Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

Jon Magnus Mathisen Lauvrak
Erik Daniel Haukås Moe

Direct Methods for Detecting and Tracking Objects from Visual Images from Moving Cameras in a Marine Environment

Masteroppgave i Kybernetikk og Robotikk
Veileder: Edmund Førland Brekke
Medveileder: Lars-Christian Ness Tokle, Henrik Dobbe Flemmen, Nicholas Dalhaug

Juni 2023

NTNU
Kunnskap for en bedre verden

Jon Magnus Mathisen Lauvrak
Erik Daniel Haukås Moe

Direct Methods for Detecting and Tracking Objects from Visual Images from Moving Cameras in a Marine Environment

Masteroppgave i Kybernetikk og Robotikk
Veileder: Edmund Førland Brekke
Medveileder: Lars-Christian Ness Tokle, Henrik Dobbe Flemmen, Nicholas Dalhaug
Juni 2023

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

NTNU

Kunnskap for en bedre verden

Preface

This thesis was written in the spring of 2023 as a part of the course TTK4900 - Engineering Cybernetics, Master's Thesis and concluded our master's degree in cybernetics and robotics, with specialization in autonomous systems, at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. Both authors worked with computer vision in their specialization project, which provided a good basis for this thesis.

We want to thank our supervisors Edmund Førland Brekke, Lars-Christian Ness Tokle, Henrik Dobbe Flemmen, and Nicholas Dalhaug for their guidance and advice during the work with the thesis.

Jon Magnus Mathisen Lauvrak
Erik Daniel Haukås Moe
Trondheim, June 12th 2023

Abstract

This thesis investigates the use of optical flow for object detection and pose estimation in the context of the surroundings of the autonomous ferry milliAmpere 1. Current methods for pose estimation are often based on state estimation filters and sensor measurements. The methods used in this thesis provide accurate estimates by tracking multiple points on the objects of interest. The data set is recorded by milliAmpere 1 in Trondheimfjorden in Norway, where five image sequences are used for testing.

By comparing a state-of-the-art deep learning detection model, YOLOv8, and the optical flow method Gunnar-Farnebeck (GF) for object detection, it is found that YOLOv8 consistently outperforms the optical flow based method in both generalization and runtime, and manage to give more accurate masks. This is crucial for good estimates of the pose of the object of interest. Conversely, optical flow is able to determine whether the target is moving.

The thesis further explores the utilization of direct optical flow techniques, specifically tracking with Lucas-Kanade (LK), combined with Structure from Motion (SFM), for 3D reconstruction of objects isolated from a scene in an image. The results are promising in terms of accurately estimating relative pose, including yaw rate and position when compared with Global Navigation Satellite System (GNSS) measurements. These pose estimates rely on various factors, such as object proximity, the clarity of the features on the object, how the points cover the object, and the number of images used for reconstruction.

The obtained results suggest that tracking and pose estimation with LK and SFM have significant potential to improve control systems for autonomous vessels. However, the need for additional research and rigorous testing is emphasized before these methodologies can be implemented in real-world scenarios. Although several challenges remain, primarily related to precision, robustness, and the distribution of points on an object, this work provides a basis for further exploration into the usage of optical flow in autonomous navigation systems.

Sammendrag

Denne avhandlingen undersøker bruken av optisk flyt for objektdeteksjon og posisjonsestimering innenfor konteksten av omgivelsene til den autonome fergen milliAmpere 1. Nåværende metoder for posisjonsestimering er ofte basert på tilstandsestimeringsfiltre og sensormålinger. Metodene brukt i denne avhandlingen gir nøyaktige estimater ved å spore flere punkter på interessante objekter. Datasettet er samlet av milliAmpere 1 i Trondheimfjorden i Norge, hvor fem ulike bildesekvenser er brukt for testing.

Ved å sammenligne en en av de ledene dyp læring deteksjonsmodellene, YOLOv8, og den optiske flytmetoden Gunnar-Farnebeck (GF) for objektdeteksjon, er det funnet at YOLOv8 konsekvent yter bedre enn den optisk flyt baserte metoden både i generalisering og kjøretid, og klarer å gi mer nøyaktige masker rundt objektene. Dette er avgjørende for gode estimater av objektets posisjon og giringsrate. På den andre siden kan optisk flyt bestemme om målet er i bevegelse.

Rapporten utforsker videre bruken av direkte optisk flyt metoder, spesielt sporing med Lucas-Kanade (LK), kombinert med Structure from Motion (SFM) for 3D-rekonstruksjon av objekter isolert fra bakgrunnen i et bilde. Resultatene er lovende når det gjelder nøyaktig estimering av giringsrate og posisjon, sammenlignet med Global Navigation Satellite System (GNSS) målinger. Estimatene er vist avhengig av forskjellige faktorer, inkludert objektets nærhet, klarheten av detal- jene på objektet, hvordan punktene dekker objektet, og antall bilder som brukes til rekonstruksjon.

Resultatene antyder at sporing og posisjon- og orienteringsestimering med LK og SFM har betydelig potensial for å forbedre kontrollsystemer for autonome fartøy. Imidlertid un- derstrekes behovet for videre forskning og grundig testing før disse metodene kan implementeres i virkelige scenarier. Selv om flere utfordringer gjenstår, hov- edsakelig knyttet til presisjon, robusthet og punktdistribusjon på objektet, legger denne oppgaven et grunnlag for videre utforskning av anvendelsen av optisk flyt i autonome navigasjonssystemer.

Contents

Preface . . . . . **iii**
Abstract . . . . . **v**
Sammendrag . . . . . **vii**
Contents . . . . . **ix**
Abbreviations . . . . . **xiii**
1 Introduction **1**
1.1 Motivation 1
1.2 Litterature review 2
1.3 Problem description 4
1.4 Contribution 5
1.5 Outline 5
2 Theory **7**
2.1 Optical flow 7
2.1.1 Lucas-Kanade optical flow algorithm 8
2.1.2 Pyramidal implementation of Lucas-Kanade 9
2.1.3 Gunnar-Farneback optical flow algorithm 10
2.2 Background subtraction and detection with optical flow 13
2.3 You Only Look Once 15
2.4 Structure from motion 16
2.5 Camera model 17
2.6 The fundamental matrix 19
2.7 Motion estimation 22
2.8 Triangulation 22
2.9 Bundle adjustment 24
2.10 Finding good points to track 25
2.11 Outlier removal 26
2.12 Rotation matrices 27
2.13 Extracting yaw angle from rotation matrices 28
2.14 Coordinate systems 28
2.14.1 North East Down coordinate system 28
2.14.2 Body coordinate system 29
2.14.3 Camera coordinate system 29
2.14.4 Transformation between coordinate systems 30
2.15 Minimum Volume Enclosing Ellipsoids 31

3 Platform **35**
3.1 milliAmpere 1 and milliAmpere 2 35
3.2 External vessels 36
3.3 Environments and scenarios 38
3.3.1 Environment 1 38
3.3.2 Environment 2 39
3.4 Data 40
4 Methodology **43**
4.1 Camera calibration 43
4.2 Detection - optical flow 44
4.3 Detection - YOLOv8 45
4.4 Tracking 46
4.5 Initial tracking points 47
4.6 SFM - OpenCV 48
4.7 Correcting result from SFM 49
4.8 Positive and negative direction for yaw 51
4.9 Estimating relative yaw rate from SFM 52
4.10 Simulated real-time scenario for yaw rate estimation 52
4.11 Estimating the center of the target 53
4.12 Estimating position of the target from reconstruction 53
4.13 Ground truth 54
4.13.1 Ground truth for change in heading 54
4.13.2 Transformation of ground truth 55
4.14 Pre-processing of rotational measurements 55
5 Results: Detection **57**
5.1 Detection with optical flow (GF) 57
5.2 Detection with YOLOv8 60
6 Results: Tracking, Structure From Motion and pose estimation **65**
6.1 Tracking with multiple points (LK) 65
6.2 3D reconstruction by SFM with moving camera 68
6.3 3D reconstruction of boats with SFM with a moving camera and moving object 70
6.4 Outlier removal 73
6.5 The problem of ambiguity 73
6.6 Relative yaw rate of objects from rotation matrices 74
6.7 Tuning of reconstruction and yaw rate estimation 79
6.7.1 Maximum number of points 80
6.7.2 Minimum distance between initial points 84
6.7.3 Number of images 87
6.7.4 Tuning conclusion 90
6.8 Estimation of relative yaw rate in a simulated real-time sequence . 91
6.9 The accuracy of the isolated estimate of the yaw rate of the target . 98
6.10 Center estimation of the target 100
6.11 Estimating the position of the target 101

7 Discussion 107
7.1 Detection 107
7.2 Tracking and SFM 107
7.3 Relative yaw rate estimation 109
7.4 Reflections on ground truth for yaw angle 110
7.5 Position estimation 111
8 Future Work 113
9 Conclusion 117
Bibliography 119

Abbreviations

CNN Convolutional Neural Networks. 2, 15

DLT Direct Linear Transform. 23, 24

DSO Direct Sparse Odometry. 4

FLANN Fast Library for Approximate Nearest Neighbors. 17

GF Gunnar-Farnebeck. 7, 10, 13, 44, 45, 46, 57, 59

GNSS Global Navigation Satellite System. 5, 38, 54, 75, 96, 97, 100, 102, 103, 104, 109, 110, 111

JIPDA Joint Integrated Probabilistic Data Association. 3

LIDAR Light Detection and Ranging. 2, 36, 111, 114

LK Lucas-Kanade. 4, 7, 8, 9, 10, 16, 17, 18, 46, 47, 48, 49, 54, 65, 66, 68, 69, 71, 107, 108, 109, 113, 114, 117

MHT Multiple Hypothesis Tracker. 3

MVEE Minimum Volume Enclosing Ellipsoids. 31, 32, 33, 53, 100, 101, 104, 105, 111

NED North East Down. 28, 51, 103

ORB Oriented Fast And Rotated Brief. 16

RANSAC Random Sample Consensus. 21, 74, 78, 84

RMSE Root-Mean-Square Error. 81

SFM Structure From Motion. 4, 5, 7, 16, 17, 18, 22, 25, 26, 28, 30, 38, 47, 48, 49, 52, 53, 54, 55, 65, 66, 68, 69, 70, 71, 73, 74, 75, 77, 78, 79, 81, 82, 84, 87, 89, 90, 99, 104, 105, 107, 108, 109, 110, 113, 114, 117

SIFT Scale-Invariant Feature Transform. 16

SSD Single Shot Detector. 2

SVD Singular Value Decomposition. 20, 23, 24

SVM Support Vector Machine. 2

vSLAM visual Simultaneous Localization and Mapping. 4

YOLO You Only Look Once. 2, 4, 15, 16, 45, 61

Chapter 1

Introduction

1.1 Motivation

Navigation by sea has historically been done by the crew aboard a vessel, and by staying aware of the boat's surroundings, one would be able to avoid collision with approaching vessels. Hence, one would avoid unnecessary damage to both the boat and the people aboard. With the rising need for means of transport and the increase of people apparent both in marine and urban areas, new technology has been developed.

Today, the technology within the field of autonomy is evolving faster than ever. For instance, autonomous vehicles are starting to be accepted on the roads, and machines can now do tasks that were done earlier by humans in an even faster manner. This opens opportunities in terms of economics and efficiency and could benefit society in multiple areas. This is also the case for autonomous vessels at sea, which can be a great supplement in terms of the transport of goods or as a way for humans to get past obstacles like for instance a lake or channel. However, when removing the man in the loop, one could also stumble upon problems that did not appear before.

When allowing autonomous vehicles in society, there has to be a guarantee that the vehicle follows certain safety measures and that no humans will get hurt due to technical errors or faults. For example, guidelines, such as the EU Operational Guidelines For Trials Of Maritime Autonomous Surface Ships (MASS) [1], have been developed to make sure that autonomous vessels are suited for usage.

Regardless of guidelines that will provide safety measures for autonomous vessels, the field of autonomy is new and will naturally provide some skepticism from the general public. When new technology is introduced, especially when the technology no longer has a man in the loop, the demand for reliable and safe systems is even higher. This demand is justified because an error in a worst-case scenario can be life-threatening.

Accurate methods for detecting and tracking nearby elements are necessary to ensure safety requirements. There are multiple sensors that could be used for this purpose. For instance, one could use Light Detection and Ranging (LIDAR) or radar to recognize when an element approaches. However, this would not give too much information about the object itself. On the other hand, a camera gives a lot of valuable information about the size and looks of the element. Cameras might therefore be a good supplement to the previously mentioned sensors and could be suitable for detecting and tracking surrounding elements.

In this thesis, a camera will be used to detect vessels in the surroundings of an autonomous ferry in a marine environment. The problem of detection and tracking of close-by objects will, however, also be interesting for multiple applications such as for instance autonomous driving of cars in traffic.

1.2 Litterature review

Object detection is a problem in computer vision where the objective is to locate and recognize objects in an image. In the early days of object detection, simple techniques were used and mainly based on the extraction of features and rules that would decide the existence of an object, for instance, Support Vector Machine (SVM) [2] or decision trees [3]. The detection methods evolved until the 2000s to detect more detailed patterns like faces with the Viola-Jones algorithm [4].

Later on, in the 2000s, more advanced methods were developed, and object detection moved further toward machine learning-based methods. This came with the evolution of Convolutional Neural Networks (CNN) which were able to recognize detailed patterns in images which allowed to detect even more advanced shapes. An example of a network that surged the interest for CNN is AlexNet [5] which was introduced in 2012 and could classify images with high accuracy.

In recent years the field of object detection has developed a lot. The introduction of deep neural networks has brought new methods for detection which can be trained to recognize features of even more advanced objects. A widely used detector that has been developed in recent years is the single-stage detector which can simultaneously detect and classify an object in a single frame. Examples of such detectors are You Only Look Once (YOLO) [6] and Single Shot Detector (SSD) [7] which are both fast and accurate. In January 2023, the most recent version of YOLO, YOLOv8, was released, which has great performance in both in detection and segmentation of objects.

The methods based on neural networks for detection are promising. Nevertheless, there are reasons to make an attempt to challenge these so-called indirect methods. The mentioned methods need a good set of samples to be used for training in order to be able to detect the objects of interest. This data is used to extract features that will be identifiable in new images which is the principle of most indirect methods such as the mentioned deep learning methods. This requires a large amount of data that needs to be created for the specific field of use. Another problem is that not only boats will appear in the surroundings of the autonomous vessel and followingly unknown and new objects can be missed by the detector.

An interesting approach that can be used for object detection is optical flow. This is not as commonly used in the field of detection but can be a good tool to discover moving objects in a camera frame and can give the segment of the moving object from how its movement differs from the movement of the background. One technique that is used is background subtraction which will remove parts of the image and leave the masks of moving objects as detections as for instance used in [8]. This method could be beneficial as it can detect all kinds of objects without prior training, and such a method will be tested in this thesis. Optical flow is also a good technique for further tracking of objects.

Various methods already exist within object tracking. One category of tracking algorithms is model-based methods. These methods use a mathematical model to predict the movement of an object and combine this with the measurements of a sensor in order to find the next location of the object. Examples of such methods are the Kalman filter first presented in [9], Joint Integrated Probabilistic Data Association (JIPDA) tracker [10], and Multiple Hypothesis Tracker (MHT) [11]. These methods will, however, not use all available information in an image as they mostly use only a point measurement to estimate the position of the target.

There exist multiple methods in computer vision for tracking a target directly in an image sequence and extracting information about the movement of the target. There are two main categories for this. One of them is the so-called feature-based methods [12] that require extraction of features in the image and then use features to extract more information. Methods that do not require the same features are so-called direct methods [13]. This category of methods instead looks at patterns in the image directly and tries to match a part of one image with another image to find the transformation that best describes the movement between the frames. These methods often try to minimize an error which for instance can be the difference between brightness patterns directly in the image.

Examples of direct methods in computer vision are template matching techniques, as for instance presented in [14], and optical flow. The latter technique was tested in the specialization project of Erik Daniel Haukås Moe [15] regarding tracking with optical flow. The results showed that the optical flow algorithm

Lucas-Kanade [16] gave accurate tracking of points on a target in an image frame. The algorithm has therefore shown good potential for further pose estimation of objects that are moving in marine environments.

With point correspondences from tracking, it is possible to analyze the pose of the target further. Possible methods for pose estimation could be Direct Sparse Odometry (DSO) [17] and visual Simultaneous Localization and Mapping (vSLAM) [18] which both are able to build 3D point clouds of the surroundings. DSO is originally meant for pose estimation of the camera but would also be able to give information about the relative position of an object.

Another method for 3D reconstruction is Structure From Motion (SFM). This method has been actual since Lonquet-Higgins's publication regarding two-view reconstruction [19] in 1987. In later years there have been developed methods for performing SFM with multiple views, but also a combination of the two for instance in [20] where the results of pairs are combined to a multi-view. Also, in SFM, there are both direct and feature-based methods. Direct methods work directly on image data without the need for point correspondences, for instance, as presented in [21].

1.3 Problem description

This thesis aims to evaluate an object detection algorithm that uses optical flow by comparing it with a state-of-the-art YOLO algorithm. The goal is to determine whether optical flow can offer comparable efficiency to deep learning methods, which often operate as a black box with limited transparency in their decision-making processes. Moreover, an advantage of optical flow is its ability to detect any moving object which would be advantageous in the surroundings of an autonomous ferry.

In addition, the direct optical flow method Lucas-Kanade (LK) will be used to track objects in an image frame by masking out the object and tracking points from the mask in a sequence of images. The tracked points will be used to create a SFM where multiple views will create a 3D representation of the object. As the object is masked out from the surroundings, the relative movement between the object and the camera will be isolated and possible to analyze. This will further be used to estimate the relative change in heading between the object and the camera as well as track the position of the object during the sequence. The heading of a vessel is usually estimated with position measurements and model-based filters, which will undermine important information. The aim is, therefore, to examine if a more direct method for estimating yaw will give accurate estimates. The images also include information about the position of the target which is also attempted estimated to evaluate the performance. The data is sampled from the autonomous ferry milliAmpere 1 and is recorded in Trondheimsfjorden where five

different image sequences are used for testing. The data set also provides Global Navigation Satellite System (GNSS) measurements which will be used as ground truth for the objects to confirm the results.

1.4 Contribution

The thesis studies optical flow as a method for detection as well as using optical flow to estimate the pose of boats adjacent to the autonomous ferry milliAmpere 1. The detection method will show an alternative option of detection compared to the deep learning methods that already exist. Furthermore, the pose estimation using optical flow will show whether or not optical flow is an accurate method to use for tracking by sea. The analysis of the 3D reconstruction with SFM will show what information about the pose of an object it is possible to extract with a single camera, where a limited amount of research has been done previously. A good result will also potentially provide a more efficient method for pose estimation of close objects by sea. The code from the project could also possibly be used for further development of tracking systems.

1.5 Outline

- Chapter 2 in the thesis will give an introduction to the background theory that is used to generate the results in this thesis. This includes theory about optical flow, structure from motion, and additional theory needed.
- Chapter 3 will give an introduction to the scenarios and targets which is used for tracking and detection in this thesis to give an understanding of where the data comes from.
- Chapter 4 presents implementation choices and definitions used in the thesis as well as how ground truth is defined and generated.
- Chapter 5 presents the results obtained regarding detection with both optical flow and an alternative method.
- Chapter 6 introduces the results regarding the tracking of targets and scene reconstruction and will address potential challenges with the methods used. It also presents the results from pose estimation of targets surrounding milliAmpere 1. This mainly includes the change in heading and position estimates.
- Chapter 7 will give a discussion of the obtained results.
- Chapter 8 presents potential future work, and how to improve the results from this thesis.
- Chapter 9 will conclude the thesis.

Chapter 2

Theory

This chapter will give an introduction to the theory behind this thesis. This includes the theory behind detection, optical flow, SFM, and pose estimation. This thesis is, to some degree, a continuation of the specialization thesis of Erik Daniel Haukås Moe [15], and the following sections will therefore be related to the theory in the project: Optical flow (including LK), Camera model and Finding good points to track. All the figures in this chapter are created by the authors unless stated otherwise.

2.1 Optical flow

Optical flow is a technique in computer vision to estimate the relative movement in a sequence of images. The movement that is estimated arises either from the movement of elements in the image or the movement of the camera view. In either scenario, one can determine the movement of elements within the image frame relative to the camera. The basic idea behind optical flow is to track the movement of pixels between consecutive frames and use this information to infer the overall motion in the scene further. Optical flow does this by detecting the movement of brightness patterns between images. Optical flow methods can be useful for a range of different tasks in the field of computer vision, such as object tracking, video stabilization, action recognition, and scene flow estimation, among others. Well-known optical flow methods are the sparse optical flow algorithm LK and the dense optical flow algorithm Gunnar-Farnebeck (GF) described in later sections.

When the optical flow between two consecutive frames is found, two following main assumptions are normally considered:

- Brightness consistency, meaning that the brightness in the two frames is constant, which implies that the pixel value in the template has the same value in the image frame.
- The optical flows in an area are consistent, meaning that the movement of adjacent pixels belonging to the same target is consistent.

The first assumption is derived from the fact that the motion of an object is found by locating similar patterns in brightness in the next image. If the brightness differs significantly, between frames, then the location of similar patterns becomes hard. This is a strict assumption as changes in brightness can be significant in real-life scenarios. The second assumption comes from the fact that the optical flow is calculated by matching regions of an image in one frame to a region in the other frame, meaning that the points in that region must move together.

2.1.1 Lucas-Kanade optical flow algorithm

The Lucas-Kanade optical flow algorithm is a highly versatile algorithm, with one of its many uses being tracking [16]. It is a feature-based optical flow algorithm meaning that it estimates the optical flow for a sparse set of key points or interest points in the image. This stands in contrast to the dense optical flow methods, which estimate the movement for each pixel in the image. The ability of tracking can provide information on the movement of objects relative to their surroundings and the camera. The LK optical flow algorithm works through temple-matching, where it tries to find a match between a template T(x) and a patch in the second image I(x). The template is defined as a region surrounding a point of interest in an original image, for instance, a (15x15) patch.

The optical flow of the point of interest is given by the movement of the point x = (x, y). This is found through the matching process described above and will be defined by a warp W(x; p). This is a transformation where p is a vector that describes the movement of the point in the x and y directions. p can therefore be seen as the optical flow between the images. A simple warp that only includes translation could be defined as

$$\boldsymbol{W(x;p)} = \begin{bmatrix} x + p_1 \\ y + p_2 \end{bmatrix}. \tag{2.1}$$

The movement can also be described in a more advanced manner by including more elements in p.

The matching process is based on the minimization problem

$$\min \sum_x (I(\boldsymbol{W(x;p)}) - T(\boldsymbol{x}))^2, \tag{2.2}$$

with respect to p. This minimizes the error between T(x) and the part of the image I(x) after applying the warp on the location of the points in T(x) in the first image, denoted I(W(x;p)).

The optimization is done by an iterative process of updating p as

$$\boldsymbol{p} \leftarrow \boldsymbol{p} + \Delta \boldsymbol{p} \tag{2.3}$$

until Δp gets below a given threshold. Δp is the step length of the iterations and decides how close I(W(x,p)) is to T(x). The movement of pixels between images will in most cases be a non-linear problem. A first-order Taylor expansion of (2.3) is used to linearize the problem $I(\boldsymbol{W(x;p+\Delta p)})$ resulting in $I(\boldsymbol{W(x;p)})+\nabla I \frac{\delta \boldsymbol{W}}{\delta \boldsymbol{p}} \Delta \boldsymbol{p}$. Inserting this into (2.2), setting it equal to 0, and solving for Δp results in

$$\Delta p = H^{-1} \sum_{\boldsymbol{x}} \left(\boldsymbol{\nabla I} \frac{\partial \boldsymbol{W}}{\partial \boldsymbol{p}} \right)^T (T(\boldsymbol{x}) - I(\boldsymbol{W(x;p)})). \tag{2.4}$$

Here, $\nabla I = (\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$ represents the image gradient, or the change in pixel intensity in the image, $\frac{\partial W}{\partial p}$ is the Jacobian of the warp with respect to the elements in p, and H represents an approximation of the Hessian defined as

$$H = \sum_{\boldsymbol{x}} \left(\boldsymbol{\nabla I} \frac{\partial \boldsymbol{W}}{\partial \boldsymbol{p}} \right)^T \left(\boldsymbol{\nabla I} \frac{\partial \boldsymbol{W}}{\partial \boldsymbol{p}} \right). \tag{2.5}$$

As the derivative of the warp is included in the equations, a requirement of W(x;p) is therefore that it is differentiable with respect to p.

2.1.2 Pyramidal implementation of Lucas-Kanade

LK uses local optimization to find the optimal p, which makes the algorithm less robust to large changes. This could for instance happen if an object moves much faster than the frame rate of a camera and therefore has large movements between two images.

A tool to make the algorithm more robust to large changes by covering more of the image is the pyramidal implementation of LK [22]. This method creates layers in a pyramid where the resolution is halved for each step. The resolution is reduced by averaging and combining neighborhoods of pixels. A pyramid is illustrated in Figure 2.1, where layer 0 represents the original image. When keeping the window size for the template T(x), the same window will cover a larger area in the same image. A pyramid is also made for the original image and the template T(x) is found for each layer as it would not make sense to use the same template for all layers.

This implementation works by iteratively running LK on each layer in the pyramid, starting at the top layer, and using the resulting transformation p as the initial transformation in the next layer. This will move the starting point closer to the solution, and local optimization will have a greater chance of working.

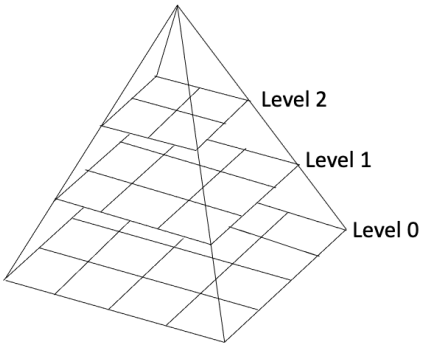


Figure 2.1: Illustration of the pyramid used in the implementation of Lucas-Kanade. This will allow the algorithm to follow larger movements of the point of interest. Originally presented in [15].

The point of interest must be the same in each layer of the pyramid. The point that is used in each layer can be expressed at

$$[x, y]^L = \frac{[x, y]}{2^L}. \tag{2.6}$$

where L is the number of the current layer in the pyramid. In addition to this, the optimal p may not always be an integer. However, when decomposing a point $[x, y]$ into $[x_0 + \alpha_x, y_0 + \alpha_y]$ where x_0 and y_0 is the integer part and α_x and α_y is the decimal part, one can achieve sub-pixel precision with following formula

$$\begin{aligned} I^L(x, y) = (1-\alpha_x)(1-\alpha_y)I^L(x_0, y_0) + \alpha_x(1-\alpha_y)I^L(x_0+1, y_0) + \\ (1-\alpha_x)\alpha_y I^L(x_0, y_0+1) + \alpha_x\alpha_y I^L(x_0+1, y_0+1), \end{aligned} \tag{2.7}$$

This pyramidal implementation of LK is used in the rest of the thesis.

2.1.3 Gunnar-Farneback optical flow algorithm

The GF optical flow algorithm is a dense optical flow algorithm developed by Gunnar Farnebäck. It was first introduced in the paper "Two-Frame Motion Estimation Based on Polynomial Expansion" which was published in 2003 [23]. It is built on the two main assumptions mentioned in the previous subsection. The optical flow is calculated using only two sequential images.

The first step in the algorithm is to estimate each neighborhood of both frames by quadratic polynomials. This means analyzing the pixels in small regions, or neighborhoods, in each image, and fitting a quadratic polynomial function that tries to approximate the pixel intensities in that region. This allows the algorithm to capture the pixel intensity pattern in each neighborhood and represent them as a mathematical signal function. This is expressed mathematically in a local coordinate system,

$$f(\boldsymbol{x}) \sim \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T \boldsymbol{x} + c \tag{2.8}$$

where $\boldsymbol{A}$ is a symmetric matrix, $\boldsymbol{b}$ is a vector and c is a scalar. The coefficients are estimated for each region using a weighted least squares fit. There are two weighting components used: *certainty* and *applicability*. The certainty weight is linked to the signal values within the region. The applicability weight is determined by the position of a pixel within its neighborhood. It is normal to weigh the center of the neighborhood higher compared to those further away from the center.

As discussed in the previous section, the goal of an optical flow method is to find the displacement between pixels between frames. In the Gunnar-Farnebäck method, the goal is to find the displacement between corresponding polynomials signal functions.

This is analyzed by considering an ideal translation whereas $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ are displaced by a displacement $\boldsymbol{d}$. This scenario is described mathematically as:

$$f_1(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A_1}\boldsymbol{x} + \boldsymbol{b_1}^T \boldsymbol{x} + c_1 \tag{2.9}$$

and

$$\begin{aligned} f_2(\boldsymbol{x}) &= f_1(\boldsymbol{x} \pm \boldsymbol{d}) = (\boldsymbol{x} \pm \boldsymbol{d})^T \boldsymbol{A_1}(\boldsymbol{x} \pm \boldsymbol{d}) + \boldsymbol{b_1}^T(\boldsymbol{x} \pm \boldsymbol{d}) + c_1 \\ &= \boldsymbol{x}^T \boldsymbol{A_1}\boldsymbol{x} + (\boldsymbol{b_1} \pm 2\boldsymbol{A_1}\boldsymbol{d})^T \boldsymbol{x} + \boldsymbol{d}^T \boldsymbol{A_1}\boldsymbol{d} \pm \boldsymbol{b_1}^T \boldsymbol{d} + c_1 \\ &= \boldsymbol{x}^T \boldsymbol{A_2}\boldsymbol{x} + \boldsymbol{b_2}^T \boldsymbol{x} + c_2 \end{aligned} \tag{2.10}$$

Using the fact that

$$\boldsymbol{A_2} = \boldsymbol{A_1} \tag{2.11}$$

$$\boldsymbol{b_2} = \boldsymbol{b_1} \pm 2\boldsymbol{A_1}\boldsymbol{d} \tag{2.12}$$

one obtains

$$c_2 = \boldsymbol{d}^T \boldsymbol{A_1}\boldsymbol{d} \pm \boldsymbol{b_1}^T \boldsymbol{d} + c_1 \tag{2.13}$$

Which, if $\boldsymbol{A_1}$ is non-singular, one can solve for $\boldsymbol{d}$ and get

$$2\boldsymbol{A_1}\boldsymbol{d} = (\boldsymbol{b_2} \pm \boldsymbol{b_1}) \tag{2.14}$$

$$\boldsymbol{d} = \frac{1}{2}\boldsymbol{A_1}^{-1}(\boldsymbol{b_2} \pm \boldsymbol{b_1}) \tag{2.15}$$

It is important to note that the assumptions that the regions can be represented by a second-degree polynomial and the perfect translation is unrealistic in practical scenarios. However, the assumptions can be made in practical scenarios,

even though errors are introduced when the assumptions are relaxed.

To address this issue, the global polynomial can be replaced with local polynomial approximations by doing a polynomial expansion of both images. This will give us expansion coefficients $\boldsymbol{A}_1(\boldsymbol{x})$, $\boldsymbol{b}_1(\boldsymbol{x})$, and $c_1(\boldsymbol{x})$ for the first image, and $\boldsymbol{A}_2(\boldsymbol{x})$, $\boldsymbol{b}_2(\boldsymbol{x})$, and $c_2(\boldsymbol{x})$ for the second image. Theoretically, this should give $\boldsymbol{A}_1 = \boldsymbol{A}_2$, but in practice, but the following approximation is used

$$\boldsymbol{A}(\boldsymbol{x}) = \frac{\boldsymbol{A}_1(\boldsymbol{x}) + \boldsymbol{A}_2(\boldsymbol{x})}{2}. \tag{2.16}$$

One can also introduce the term

$$\Delta\boldsymbol{b}(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{b}_2(\boldsymbol{x}) - \boldsymbol{b}_1(\boldsymbol{x})) \tag{2.17}$$

to obtain the primary constraint

$$\boldsymbol{A}(\boldsymbol{x})\boldsymbol{d}(\boldsymbol{x}) = \Delta\boldsymbol{b}(\boldsymbol{x}), \tag{2.18}$$

where $\boldsymbol{d}(\boldsymbol{x})$ indicates that the global displacement has been replaced with a spatially varying displacement field.

One could solve (2.18) point-wise. However, as this turns out to give noisy results, it is beneficial to make the assumption that the displacement field is slowly varying. This leads to trying to find $\boldsymbol{d}(\boldsymbol{x})$ that satisfies (2.18) while being feasible over a neighborhood I of $\boldsymbol{x}$. Mathematically this leads to the minimization problem

$$\sum_{\Delta\boldsymbol{x}\in I} w(\Delta\boldsymbol{x}) \| \boldsymbol{A}(\boldsymbol{x}+\Delta\boldsymbol{x})\boldsymbol{d}(\boldsymbol{x}), \quad \Delta\boldsymbol{b}(\boldsymbol{x}+\Delta\boldsymbol{x}) \|^2 \tag{2.19}$$

where $w(\Delta\boldsymbol{x})$ is a weight function for each neighborhood using certainty and applicability as discussed earlier. The minimum is obtained for

$$\boldsymbol{d}(\boldsymbol{x}) = (\sum w\boldsymbol{A}^T\boldsymbol{A})^{-1} \sum w\boldsymbol{A}^T\Delta\boldsymbol{b}. \tag{2.20}$$

Note that subscripts have been dropped to make the equation more readable.

The minimum value will then become

$$e(\boldsymbol{x}) = (\sum w\Delta\boldsymbol{b}^T\Delta\boldsymbol{b})\boldsymbol{d}(\boldsymbol{x})^T \sum w\boldsymbol{A}^T\Delta\boldsymbol{b}. \tag{2.21}$$

This value can be seen as a measure of the confidence of the optical flow.

The assumption made earlier that the corresponding polynomials from two frames are identical is problematic as the polynomial expansions are local models. The polynomials will therefore vary spatially, introducing errors in the constraints. This is especially problematic for larger displacement if one was restricted to comparing only polynomials at the same location. However, one can use prior

knowledge about the displacement field to compare the polynomial at $\boldsymbol{x}$ in the first signal to the polynomial at $\boldsymbol{x}+\tilde{\boldsymbol{d}}(\boldsymbol{x})$ where $\tilde{\boldsymbol{d}}(\boldsymbol{x})$ in the second signal. $\tilde{\boldsymbol{d}}(\boldsymbol{x})$ is the prior displacement field rounded to the closest integer, as to be compatible with the algorithm.

This results in being able to rewrite (2.16) and (2.17) as

$$\begin{aligned} \boldsymbol{A}(\boldsymbol{x}) &= \frac{\boldsymbol{A}_1(\boldsymbol{x})+\boldsymbol{A}_2(\widetilde{\boldsymbol{x}})}{2}, \\ \Delta\boldsymbol{b}(\boldsymbol{x}) &= \frac{1}{2}(\boldsymbol{b}_2(\widetilde{\boldsymbol{x}})-\boldsymbol{b}_1(\boldsymbol{x}))+\boldsymbol{A}(\boldsymbol{x})\widetilde{\boldsymbol{d}}(\boldsymbol{x}), \end{aligned} \tag{2.22}$$

where

$$\widetilde{\boldsymbol{x}} = \boldsymbol{x}+\widetilde{\boldsymbol{d}}(\boldsymbol{x}). \tag{2.23}$$

Here, $\frac{1}{2}(\boldsymbol{b}_2(\widetilde{\boldsymbol{x}})-\boldsymbol{b}_1(\boldsymbol{x}))$ computes the remaining displacement and $\boldsymbol{A}(\boldsymbol{x})\widetilde{\boldsymbol{d}}(\boldsymbol{x})$ adds the a priori displacement. The integration of a priori displacement field into the algorithm allows for the implementation of an iterative loop. As the quality of the a priori estimate improves, the relative displacement decreases, thereby enhancing the probability of obtaining a precise displacement estimate.

2.2 Background subtraction and detection with optical flow

The optical flow algorithm GF can be used to detect objects in an image. If an object moves in an image in a different direction than the background it will create a patch in the image with a different flow than the rest. Such a method for detection will also be able to detect multiple objects at once as it masks out movements that differ from the rest. The concept of object detection using optical flow is based on background removal, meaning that a common flow in the background is found and subtracted from the image. In many cases of detection with optical flow, the camera is not moving, meaning that the flow of the background is zero, this will not always be the case when a camera is mounted on a boat. The method used in this thesis is therefore inspired by the method from [24], mainly because it takes both the movement of a camera and the movement in the image into account. An overview of the method is illustrated in Figure 2.2.

This method first estimates the dense optical flow between two images. In this thesis, GF is used for this purpose. Some of the optical flow is caused by moving objects in the image while some is caused by movements of the camera. Defining the latter movement as $\boldsymbol{m}$ one can decompose the vector into $\boldsymbol{m}_r$ and $\boldsymbol{m}_t$ which represents rotation and translation respectively. The rotation and transition in the image are described by the parameters A, B, and C and U, V, and W respectively.

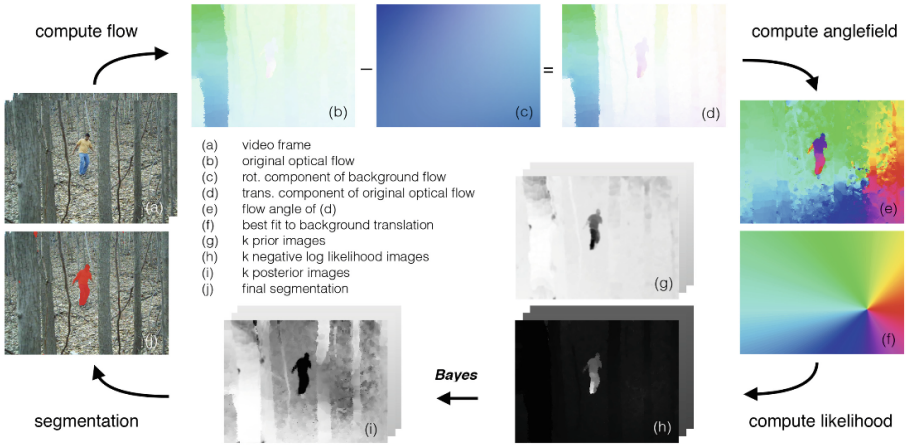


Figure 2.2: An illustration of the pipeline of the detection method. First, find the optical flow in the image and subtract the optical flow component caused by camera rotation. Then, find the angles of the flow to calculate the probability describing whether or not a set of pixels belongs to the background. The probability is then used in the process of segmentation which results in the detection of an object. The illustration is taken from [24].

The camera motion is found by the method of [25] for camera motion estimation in an image. This method finds the translation by optimizing

$$\arg\min_{U',V',W} \sum_i ||e_i(v_i, U', V', W)||, \tag{2.24}$$

where v_i is the total optical flow in a pixel and e_i is an error component orthogonal to v_i after decomposing v_i into p_i and e_i where p_i points in the direction of the estimated vector field θ_{m_t} which will be addressed later. [24], however, suggests a modified method that also finds the rotation given as

$$\hat{M} = \arg\min_{A,B,C,U',V',W} \left(\min_{U,V,W} \sum_i ||e_i(v_i, A, B, C, U', V', W)|| \right). \tag{2.25}$$

$\hat{M}$ is an estimate of the collection of θ_{m_t}'s which describes the directional optical flow for each pixel caused by translation of the camera. One can define the observed optical flow, after subtracting the rotation as v_t. By modeling the probability of observing a given optical flow v_t based on a prior translation vector field M_j denoted $\mathrm{p}(v_t \,|\, M_j)$ and a priori $\mathrm{p}(M_t)$ one can use Bayes law to obtain the probability for each translational angle field at each pixel location given as

$$p(M_j|v_t) \propto p(v_t|M_j)p(M_j) \tag{2.26}$$

This can be used to filter out unlikely movements in the image.

$p(M_j)$ is initially found by iterative methods and propagated to the next image. $p(v_t|M_j)$ on the other hand is found through

$$p(v_t|M_t) \propto p(\theta_{v_t}|||v_t||, M_j), \tag{2.27}$$

where θ_{v_t} is the angular optical flow field of v_t. This probability is well-modeled by The von Mises distribution [26] and can be found for each image. This will initially be a uniform distribution given by $\frac{1}{2\pi}$.

(2.26) can then be used to filter out pixels with a non-regular movement by

$$L = \arg\max_j p(M_j|v_t), \tag{2.28}$$

and one can segment out the moving objects in the image. For more detailed information about the method in this section, the reader is referred to [24].

2.3 You Only Look Once

YOLO is a detection and segmentation network which is used for the detection and classification of objects in an image. The YOLO network was first introduced in 2015 [27] and has been developed since with the 8th version, YOLOv8, being released in January 2023 by Ultralytics. In a traditional classification network, finding candidate regions for objects and classifying the object is done in two stages. The YOLO network differ from traditional networks as it does the same process in one stage making it faster. In this section, the general structure of a YOLO network is explained, not a specific version. It will also focus on a fundamental understanding of the network rather than a detailed presentation of each step.

The backbone of a YOLO network is a CNN which consists of convolutional layers which extract features from the input data, pooling layers that reduce the dimension of the input, and fully connected layers that are used for classification which consists of weights that give certain outputs for each class. The fully connected layers are normally applied last in the CNN. The CNN will not be explained further in this section, for more details, the reader is referred to [28].

To use the CNN, the input image in the YOLO network is divided into a grid of cells. Each cell is then responsible for detecting an object placed inside the respective cell. If an object is detected, a bounding box is created. This bounding box includes information about the location of the object and the confidence of it being an object of interest. If the object is located between multiple cells, the cells can collaborate to create the bounding box. An illustration of this process is shown in Figure 2.3. In addition to the bounding boxes, YOLO also predicts the probability of the object belonging to a certain class, and the class with the highest

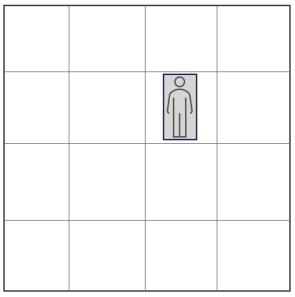


Figure 2.3: Illustration of how an image is split into a grid where a person is detected in one of the cells.

probability is assigned to the object. If a segmentation version of the network is used, additional convolutional layers are used to assign a class to each pixel in the cell to a class. One will then obtain a mask for each detected object.

A YOLO network is trained by giving the network images with objects with known localization and class and the network will train the weights to give the correct output for the class of interest. When training, a loss function gives the network feedback in three main areas: how well it localizes the bounding box, the error in class probability, and the confidence of the bounding box. This is possible as the training data also includes this information. The parameters of the network are optimized during the training.

2.4 Structure from motion

SFM is a method for reconstructing a 3D scene from a sequence of 2D images with varying camera positions as well as finding the camera positions in the images that are used. This is done by analyzing the corresponding features in the images. An illustration of the principle of reconstruction is shown in Figure 2.4. It should also be mentioned that the reconstruction of the scene only will be up to scale [29]. To find the true size and distance to objects one would have to combine SFM with for example other sensors, a depth camera, or stereo vision from two cameras.

The normal pipeline for SFM is shown in Figure 2.5 as the diagram to the left. The diagram to the right shows the version with the use of LK. The three first steps that are normally used are based on finding point correspondences in the images that are used for reconstruction. For feature detection, multiple options can be used. Some examples are Oriented Fast And Rotated Brief (ORB) [30] and Scale-Invariant Feature Transform (SIFT) [31]. These methods find the best features in the image, and will also give descriptions of the surroundings of the so-called key points that are found. Describing the surroundings of the points can be seen as the next step in the pipeline. The key points in the different images will then

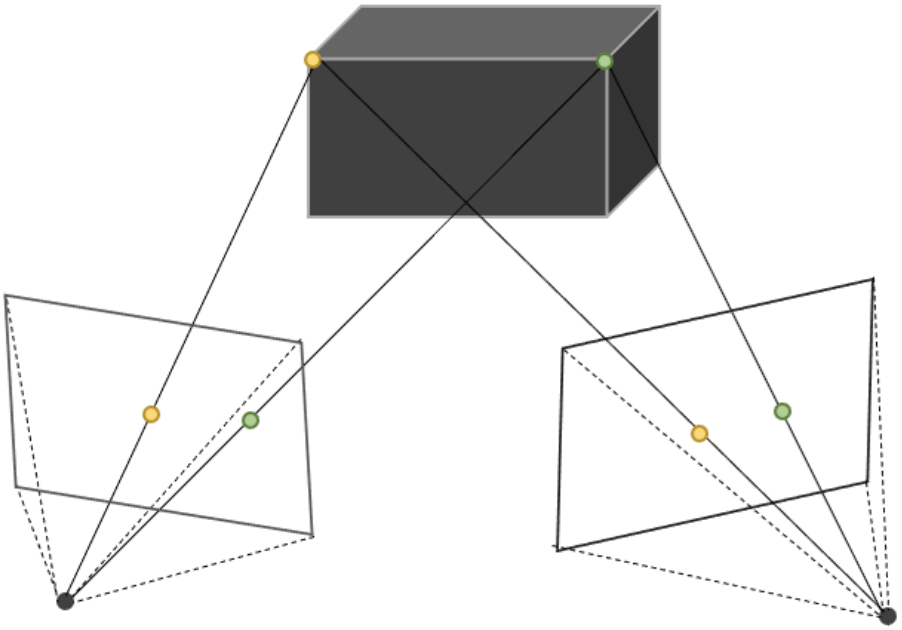


Figure 2.4: An illustration of the relation between 2D points in image frames and the corresponding 3D point. The illustration shows the principle of reconstruction in SFM. The colors of the points show the correspondences in the respective images.

be matched. This can be done by for example using the C++ library Fast Library for Approximate Nearest Neighbors (FLANN) or Brute-force matching. The three steps that are described in this section paragraph are the ones to be replaced by the LK optical flow algorithm. This will follow key points between the frames instead of finding the points in each frame and matching them. The steps will therefore not be described further in this chapter.

The next steps are motion estimation, triangulation, and bundle adjustment. In the first two, the 3D location of the points is found as well as the relative location of the camera from the different images. The bundle adjustment is then applied to adjust the points to best fit all the images used. The number of images used is arbitrary, but there should be at least two. The mentioned steps are used in this thesis, and will therefore be described further in this chapter.

2.5 Camera model

Before elaborating on the last steps in SFM, an understanding of the camera model and camera matrices is needed.

A camera model is a mathematical representation of how an image of the real

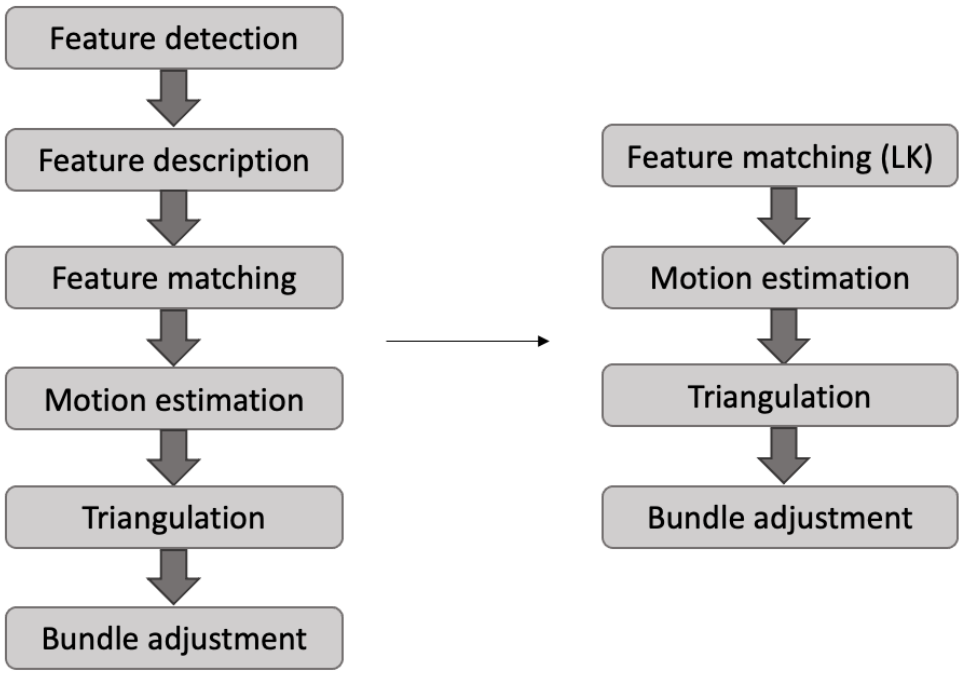


Figure 2.5: A description of SFM. On the left, is an illustration of a normal pipeline for SFM, and on the right, is an illustration of the SFM pipeline in combination with LK.

world is captured. This means that it gives the mapping of a 3D point in the real world to a 2D point in the image. A much-used model is the pinhole model [32]. This model can be described by a set of so-called intrinsics and extrinsics. The intrinsics are found through calibration for each individual camera with an object with a known pattern [33].

The intrinsic of the camera model is typically described by a 3x3 matrix called the camera calibration matrix. A general example of this matrix is shown in Equation (2.29). The matrix includes information about the focal length of the camera, the principal point, and the skew represented by the parameters f, u, and c respectively. The subscript of the parameters tells which direction the parameter is applicable to. Figure 2.6 illustrates how the world frame relates to the image frame.

$$\boldsymbol{K} = \begin{bmatrix} f_x & c & u_x \\ 0 & f_y & u_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.29}$$

The exintrics of the camera represent the relative position and orientation of the camera with respect to the world frame. The exintrics are given as a 3x3 rota-

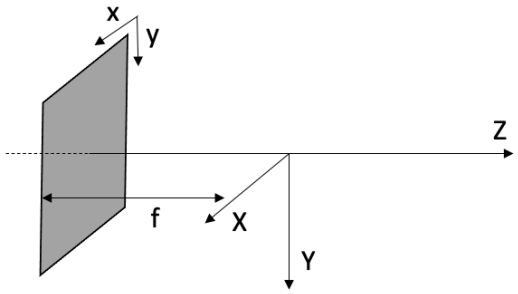


Figure 2.6: Illustration of the coordinate systems in a camera model. The dark square illustrates the image plane.

tion matrix R and a 3x1 vector t that describes the relative translation. A camera's exintrics are useful when working with multiple camera angles as they describe the relative position of the views. While the intrinsics of a calibrated camera are known and static, the exintrics describe the relative camera position for a specific case.

When combining the intrinsics and exintrics one will get the camera matrix

$$P = K[R|t], \tag{2.30}$$

which will be exclusive to each camera view in the sequence of images.

Depending on the lens used in the camera, one can obtain different versions of distortions in the image. The distortion of a camera lens describes how the pixels in an image are curved as a result of, for instance, using a wide-angle lens to capture more information about the surroundings. A camera calibration process also includes finding the distortion parameters of the camera lens. There are different types of distortion. In this thesis, the camera has a radial distortion [34] which causes straight lines in the real world to be curved when captured by the lens.

2.6 The fundamental matrix

When working with multiple camera views, it is practical to define the camera matrix P of the first camera as the starting position as

$$P = [I|0]. \tag{2.31}$$

The coordinate system will then be described in relation to the first frame as described in (2.30). The matrix R and vector t must, however, be calculated as they are not given. This can be done through the so-called fundamental matrix.

$$x'^T F x = 0. \tag{2.32}$$

The fundamental matrix, F, is a mathematical concept that in which relates corresponding points in two views of the same scene [35]. The fundamental matrix is computed by using a number of point correspondences in the two views. A point correspondence can be formulated as $\boldsymbol{x} = (x, y)$ and $\boldsymbol{x'} = (x', y')$. The fundamental matrix is then defined as in (2.32). Given a 3x3 fundamental matrix, each correspondence gives a linear equation

$$xx'f_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0, \tag{2.33}$$

where f_{xy} corresponds to an element in F, which gives F 8 degrees of freedom. when writing F as a vector in row-major order, one can, With n number of point correspondences, obtain a set of equations expressed as

$$Af = \begin{bmatrix} x_1x'_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_nx'_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} f = 0. \tag{2.34}$$

For a solution to exist for this set of equations, A must be of at least rank 8 due to the 8 degrees of freedom in f. If A has rank 8, there exists a unique solution. However, if $rank(A) = 9 > 8$ there exist multiple solutions and the solution can for instance be found through Singular Value Decomposition (SVD). This approach is called the 8-point algorithm [35]. However, by introducing the singularity constraint

$$det(F) = 0. \tag{2.35}$$

Only 7 points will be needed to find either one or three solutions for the fundamental matrix. This minimum case is called the seven-point algorithm and solves the following cubical polynomial:

$$det(\alpha F_1 + (1-\alpha)F_2) = 0. \tag{2.36}$$

The fundamental matrix is interesting as it can be used to find the camera matrix, and hence the rotation and translation of the relative camera position through epipolar geometry. [36] states that a pair of camera matrices P and P' can be connected as follows:

$$P = [I|0] \quad and \quad P' = [[e']_x F|e'], \tag{2.37}$$

where e' is an epipole in the image and $[e']_x$ is a skew-symmetric matrix of e'. The rotation and translation of the camera can therefore be extracted from the information given by the fundamental matrix from the relation to (2.30) as K is known.

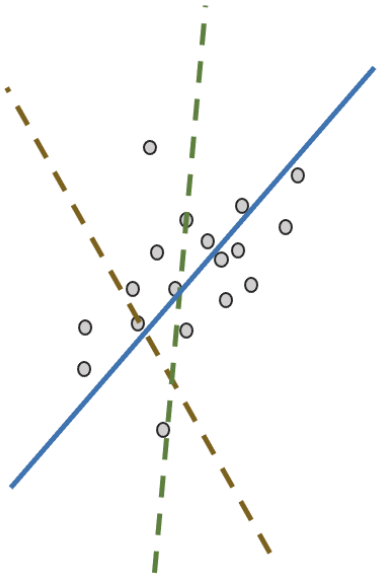


Figure 2.7: An illustration of RANSAC. The lines are randomly chosen and tested to see if they fit the points. The best-fitting line is chosen when a criterion is fulfilled. The best-fitting line is the blue line.

To find representative correspondences for the fundamental matrix (at least 7) one could for instance use a method called Random Sample Consensus (RANSAC) [37]. An illustration of the RANSAC algorithm is shown in Figure 2.7 where the principle is shown for lines that are randomly picked to fit points. In the case of fundamental matrices, 7 random points are chosen to find a matrix and then test a criterion to see whether or not it fits for most points. In the case of the OpenCV SFM module, the error that defines the criterion is the Sampson distance [35]. This is defined as

$$\sum_i \frac{(x'^T_i F x_i)^2}{(Fx_i)_1^2 + (Fx_i)_2^2 + (F^T x'_i)_1^2 + (F^T x'_i)_2^2} \tag{2.38}$$

and will tell if the fundamental matrix will work for a majority of the point correspondences or if there have been included outliers in the 7 randomly chosen points.

One problem that can arise with the fundamental matrix is the problem of ambiguity as described in Theorem 9.10 in [38]. This shows that the relation between camera poses and a fundamental matrix is non-injective, meaning that a pair of camera matrices defines a unique fundamental matrix but not vice versa. This can be a problem in this thesis when defining the movement of an object as the movement of the camera poses. One can therefore, for instance, obtain the inverse of the desired solution, where the inverse also is a correct solution in theory.

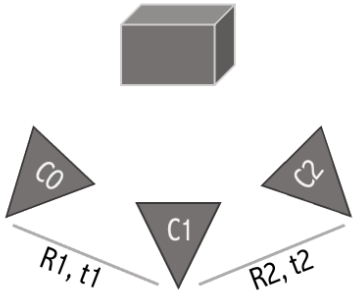


Figure 2.8: A visualization of different camera angles looking at the same object. The lines illustrate how the transformations work between the different cameras. One can move between C0 and C2 by combining the transformations.

2.7 Motion estimation

The next step in structure from motion is to use the chosen points to estimate the camera pose for the respective images that are selected for the reconstruction process. This step is needed to later be able to accurately triangulate the 3D points from 2D correspondences. The motion will be estimated between all respective frames and point correspondences should therefore be found between all images.

The position of the first camera is usually set to $[I \mid 0]$ as mentioned in the previous section. In this way, one will be able to find the rotation and translation between the frames by using the fundamental matrix as described in the previous section. The relative camera poses of the number of images one will use can then be found and will be used for further triangulation. Visualization of different camera views and the rotation and translation between them are shown in Figure 2.8.

Traditionally, SFM is predicated on the assumption that the scene under reconstruction remains static throughout the image sequence. However, in this case, this assumption does not hold, as the objects of interest in the scene are moving. To tackle this, one can mask out the object of interest to separate it from the rest of the scene. After separation, the movement of the object can be described as a camera motion. In essence, the object's relative motion can be described as a motion by the camera within its own isolated scene. This creates an equivalent condition to the original premise of SFM where the scene is static, making it possible to use traditional SFM methods.

2.8 Triangulation

After finding point correspondences and the relative position of the camera views, the next step in the process of SFM is triangulation. Triangulation is a method for

estimating the 3D position of a point by observing the point in two or more different views in 2D [39].

The process of triangulation is based on finding the 3D point X which satisfies

$$x = PX, \quad x' = P'X. \tag{2.39}$$

P and P' are the projective camera matrices in two different camera views and x and x' are the respective points in 2D in the different views. X is the 3D point that corresponds to the 2D points. The general notation for such a triangulation is

$$X = \tau(x, x', P, P'). \tag{2.40}$$

A point that match X in (2.39) would exist in perfect conditions. However, the points that are used often have some error. This could be errors in the process of finding points correspondences, but also in the camera calibration process and the estimation of camera motion. Due to noise, the triangulation process could therefore result in a small deviation as shown in Figure 2.9. The process of triangulation will therefore also include the minimization of the error that is caused by noise. The area of uncertainty when triangulating is also affected by the baseline between the images used. This is also shown in the figure. A small baseline will make it hard to give information about the depth of the 3D points. To avoid this problem one should make sure that the position of the cameras has enough variation.

There are different ways of finding solution X when doing triangulation. One way is to define a cost function, for instance for a geometric error. Another way is to use a linear triangulation method. One example is the Direct Linear Transform (DLT) method. DLT works by solving a system of linear equations. The set can be solved by for instance SVD. The set of equations to be solved in this problem can be extracted from (2.39) by taking the cross product $x \times (PX) = 0$. The following set of equations is obtained for each point:

$$x(p^{3T}X) - (p^{1T}X) = 0 \tag{2.41}$$

$$y(p^{3T}X) - (p^{2T}X) = 0 \tag{2.42}$$

$$x(p^{2T}X) - y(p^{1T}X) = 0 \tag{2.43}$$

Only two of these equations are linearly independent and one can be removed. The set of equations can then be written on the form $AX = 0$ where A is formulated as

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix}. \tag{2.44}$$

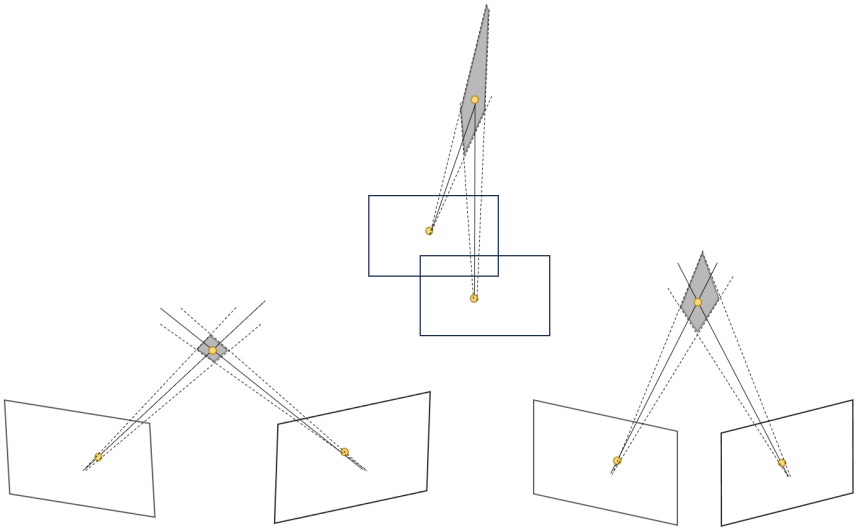


Figure 2.9: Noise in the calculation process will make it hard to find an exact solution for X. There will therefore be an error when reprojecting the point and this error is to be minimized. The yellow points show the correct solution in 2D and 3D, and the grey area is the area of uncertainty. The area of uncertainty is also affected by the baseline between the images used in the triangulation. The three figures illustrate three scenarios with the same point and a different baseline between the images.

After formulating the set of equations, the goal is to minimize the error

$$\epsilon = ||AX|| \tag{2.45}$$

which is done by using SVD. SVD gives the decomposition of A on the form

$$A = UDV^T. \tag{2.46}$$

X can be extracted from the last column in V. It should be mentioned that the solution only is up to scale as there is one more variable in the set of equations than equations. Also, the DLT method assumes small or no error in P which places all the errors in the positioning of x. After triangulating the point correspondences one will obtain a point cloud in 3D which will show the desired object.

2.9 Bundle adjustment

As mentioned in the previous section, there can occur noise that will give inaccuracies in the triangulation process. While the triangulation minimizes an error for a single point at a time, this step will minimize the reprojection error, the error after reprojecting the points to the images, for all points simultaneously. This is

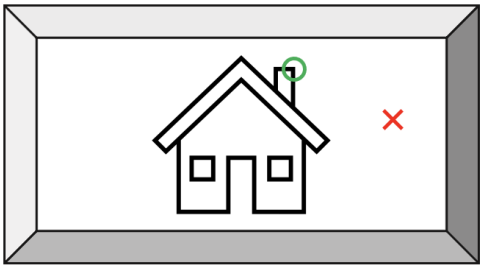


Figure 2.10: An illustration of what could be a good key point in an image. The green circle shows a corner of an object which is good while the red cross shows a bad location for a key point.

therefore a good final step in SFM to make the result even better.

In this step, both P and x are thought of as sources of error, and all parameters can be adjusted to find the best fit. The minimization problem in the bundle adjustment can be formulated as the distance between the original point in 2D, x, and the estimated reprojected point $\hat{x}$ given by $\hat{P}\hat{X}$. The mathematical expression of the problem can be formulated as

$$\min_{\hat{P}^i,\hat{X}^j} \sum_{ij} d(\hat{x}, x)^2, \tag{2.47}$$

where d($\hat{x}$, x) is the distance between the real and reprojected estimated point [39]. This problem can be solved by various methods such as the numerical minimization method Levenberg-Marquardt. The reader is referred to [40] for more detailed information on the topic.

2.10 Finding good points to track

To be able to find good points to track for use as point correspondences in SFM, a key point detector should be used. Clearly defined points are also key to tracking with greater accuracy. A key point is a point in an image that sticks out from its surroundings and will therefore be easier to recognize in a new image.

There are different aspects of a point that can qualify it as a key point. For instance, the key point could be variant to scale and intensity which will make it possible to recognize in a new image even if the conditions and camera position change or the image is subject to processing. It could also, for instance, be located where the color in the image changes significantly, like on an edge or corner of an object. I.e. the point should be robust regarding changes. An illustration of a potentially key point and a bad point for tracking is illustrated in Figure 2.10.

An example of a method for detecting such a key point is the Shi-Tomasi corner detector [41]. For this method, the decision if a point is a good candidate for a key point is based on the information in the gradient of the image. The gradient of an image describes the change in brightness patterns. The method looks for small patches in the image with a large change in the gradients. This means that a small positional change of the point will result in a large change in gradients, hence, a large change in brightness. A cost function is then used to evaluate the quality of the key point. For the Shi-Tomasi detector, the cost function is defined as

$$R = \min(\lambda_1, \lambda_2), \tag{2.48}$$

where λ_1 and λ_2 are the eigenvalues of the sum of the Jacobians in the area around the point of interest. The point is defined as a key point if R exceeds a threshold. The Jacobians are expressed as

$$\boldsymbol{J} = \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}, \tag{2.49}$$

where I_x and I_y are the directional gradients of the patch around the point.

2.11 Outlier removal

An outlier is a data point that significantly differs from other data points in the data set. This is illustrated in Figure 2.11. In the case of this thesis, outliers can exist due to inaccurate tracking of bad features or by using the resulting bad features to find the camera motion in SFM. An outlier can affect the performance of the methods and should therefore be removed before generating results.

Figure 2.11: An illustration of an outlier marked with red.

In this thesis, the Mahalanobis distance will be used which takes both the variance in the data set and the distance between points into account.

Mahalanobis distance squared can be defined as

$$D^2 = (X_1 - X_2)^T C^{-1} (X_1 - X_2), \tag{2.50}$$

where C is the covariance matrix of the distribution of the points and X_1 and X_2 are two different points of interest to measure the distance between them [42].

When using Mahalanobis distance for outlier detection in a point cloud, X_2 can be defined as the center. One can in this way find points that are located unusually far away from the point cloud compared to the rest of the points, and then filter them out with a defined threshold.

2.12 Rotation matrices

The are multiple ways of describing rotation in a coordinate system. One of them is rotation matrices. A 2D rotation matrix R can rotate a vector in the xy-plane with a rotation θ and is defined as

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{2.51}$$

When rotating objects in three dimensions, the mathematics becomes somewhat more cumbersome but is given in an intuitive way. When introducing a new dimension the rotation matrix must be expanded to 3x3 and the rotations can happen in three directions. The rotation matrix can be decomposed in three different directions, R_x, R_y, and R_z, and can be defined as

$$R = R_x R_y R_z \tag{2.52a}$$

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.52b}$$

$$R_y = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.52c}$$

$$R_z = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.52d}$$

and will rotate a vector in the directions x, y and z with the respective angles ϕ, θ and ψ [43]. This is possible because applying the total rotation is equal to applying one rotation at a time. Another property of the rotation matrices is that the inverse of the matrix is equal to the transposed matrix such as

$$R^{-1} = R^T. \tag{2.53}$$

Information about rotation matrices will be a central part of the calculation of camera positions and further calculations of for instance heading of the objects of interest.

2.13 Extracting yaw angle from rotation matrices

The yaw angle, or heading, is, in a x, y, z coordinate system, defined as the rotation around the z-axis. From the different camera views from SFM it is possible to calculate the change in the yaw angle of the object. Each camera position is described by a rotation matrix which all are related to the same coordinate system. If the respective camera matrix is defined as

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{2.54}$$

as a result of the complete matrix R in (2.52a), where r_{11} corresponds to $\cos(\psi)\cos(\theta)$ and r_{21} corresponds to $\sin(\psi)\cos(\theta)$, the rotation in yaw, ψ, can be extracted by

$$\begin{aligned} \psi &= \text{atan2}\left(\frac{r_{21}}{r_{11}}\right) = \text{atan2}\left(\frac{\sin(\psi)\cos(\theta)}{\cos(\psi)\cos(\theta)}\right) \\ &= \text{atan2}\left(\frac{\sin(\psi)}{\cos(\psi)}\right) = \text{atan2}\left(\tan(\psi)\right) \end{aligned} \tag{2.55}$$

This describes the orientation of the object in the xy-plane, or in other words, the direction around the z-axis.

2.14 Coordinate systems

A world can be represented by three dimensions, and for this purpose, a reference frame would be handy. It will therefore be necessary to define a coordinate system that describes the relative position of points and give a description of how points, for instance, different point clouds are located in relation to each other. In this thesis, the data will be presented in different coordinate systems. This section will therefore give a short introduction to the definitions in the ones that are relevant.

2.14.1 North East Down coordinate system

A commonly used coordinate system in for instance navigation is the North East Down (NED) coordinate system [43]. In this coordinate system, the orientation of the axes is set to the celestial directions north and east as well as pointing towards the center of the earth. This definition will allow for easy navigation for instance for vessels on the sea. Figure 2.12 illustrates the direction of the axes in this coordinate system. The origin can be defined for the given use case. In this thesis, ground truth will be defined in NED before being transformed and used in other reference frames.

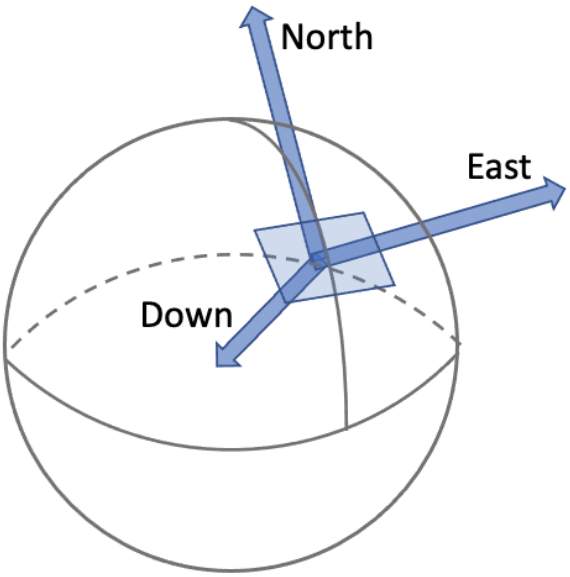


Figure 2.12: An illustration of a NED coordinate system. The figure illustrates how the axes point in the north and east direction as well as down toward the center of the earth. The globe in the figure illustrates the Earth.

2.14.2 Body coordinate system

Another coordinate system that will be used in this thesis is a body coordinate system [43]. This is a coordinate system that is fixed to a specific body or a specific object. The positive direction of the x-axis of this coordinate system will be pointing in the same direction as the heading of the object. The z-axis is defined as down. Figure 2.13 illustrates the coordinate system fixed on a boat.

A body coordinate system is often used to describe the position, orientation, and motion of the object relative to its own reference frame. The object can rotate in 3 dimensions around each of the 3 axes. Rotation around the x-axis is called roll, rotation around the y-axis is called pitch, and rotation around the z-axis is called yaw. The yaw angle of an object will be the most relevant in this thesis. This is because it described the heading of the object of interest, and therefore, for instance, can say if a boat is on a collision course or not. Roll, pitch, and yaw are illustrated in Figure 2.13 as arrows around the axes. The positive direction of the orientation follows the arrows.

2.14.3 Camera coordinate system

The last coordinate system that will be used in this thesis is the camera coordinate system. A camera coordinate system is commonly used in computer vision

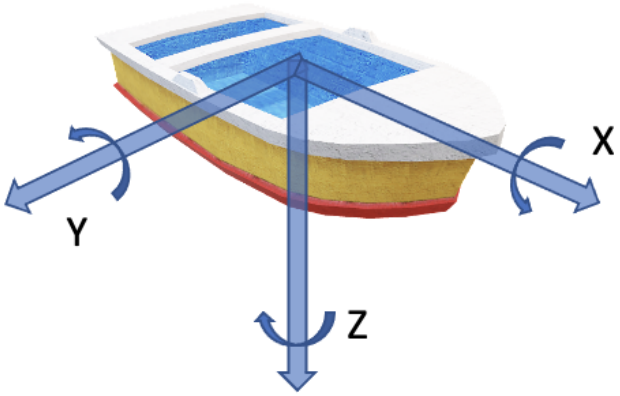


Figure 2.13: An illustration of a body coordinate system. The figure shows how x points in the same direction as the heading of the boat. The rotation of the boat is illustrated with arrows and is defined by the rotation of a right-handed coordinate system.

to describe the position and orientation of a camera in 3 dimensions [32]. In this coordinate system, the origin is typically located in the center of the camera. The way the camera coordinate system differs from, for instance, the body coordinate system is that the z-axis is aligned with the optical axis of the camera, meaning that the z-axis points directly out of the camera describing the depth in the image. The reason for this is that the depth in the camera can be visualized by moving elements further away from the camera. Followingly, the y-axis points downwards and the x-axis is perpendicular to the two axes. The coordinate system described in this section is illustrated in Figure 2.14. The same coordinate system can be recognized in Figure 2.6 which illustrates the camera model.

By using this coordinate system when working with a camera, it is possible to determine the three-dimensional location of objects in the camera's field of view. This will come in handy when using SFM for scene reconstruction.

2.14.4 Transformation between coordinate systems

When working with different coordinate systems, it is often desirable to transform between them. The difference between coordinate systems is how they are rotated and translated in relation to each other. To rotate between a coordinate system, A, and another, B, one would need the relative rotation matrix, R, and the relative translation, t. The transformation between the coordinate systems can then be applied as

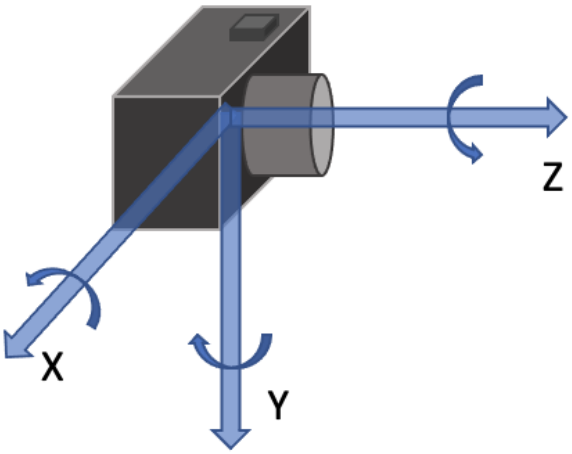


Figure 2.14: A normal right-handed coordinate system from a camera point of view. This will for instance be used when projecting points from an image to 3D coordinates. The rotation around the axes is illustrated with arrows.

$$x' = Rx + t, \tag{2.56}$$

where x is the point in A and x' is the same point in B [43]. Followingly, the same matrices can be used to transform back to the original coordinate system with the inverse transform

$$x = R^T x' - R^T t. \tag{2.57}$$

2.15 Minimum Volume Enclosing Ellipsoids

An ellipse is a geometric shape with the form of a stretched circle. An ellipsis has two axes which can have different lengths, and the shape of the ellipsis is decided by two foci which are placed on the longest axis between the center and the end-points. An ellipsis may be used to circle a point cloud and can be better than a regular circle as it has more directions of freedom.

If an ellipsis is used to find the shape of a point cloud, it is desirable to create an ellipsis that includes all points of interest. An ellipsis with these properties is called a Minimum Volume Enclosing Ellipsoids (MVEE) [44] which covers a set of points in $\mathbb{R}^n$ with the minimal ellipsoid. An example of a minimal ellipse in two dimensions is shown in Figure 2.15. The problem of finding this ellipsoid can be scaled to three dimensions with point clouds from boats which will be done in this thesis. This can be used to estimate the shape and center of the vessel when not being able to see the whole boat from all angles. This can be used to give more

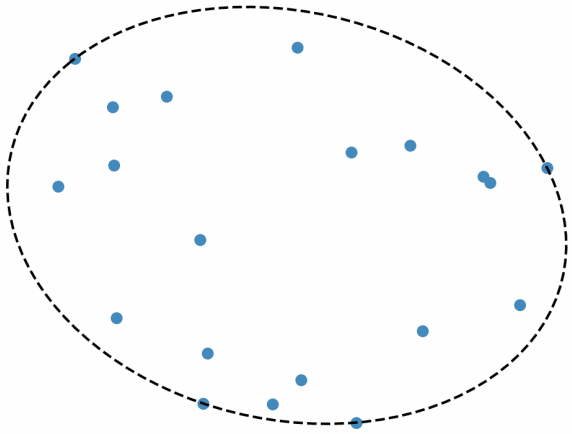


Figure 2.15: An example of a minimal ellipse that covers a set of points in two dimensions. The ellipse is defined by the points furthest away from the center.

accurate position measurements of the object.

The ellipse in the figure can be formulated on a so-called center-form as

$$(x-c)^T A(x-c) = 1, \tag{2.58}$$

for all points x on the ellipsoid, where c is the center of the ellipsoid and A is a description of the shape of the ellipsoid.

One of many methods for solving the MVEE problem is Khachiyan's algorithm [45] which is used in this thesis. Khachiyan's algorithm is based on iteratively updating the weights of points in a set of points, where the weights represent the probability of a point belonging to the ellipsoid.

The weights, u, will be initialized uniformly as 1/N for each point where N is the number of points. Also a matrix $Q = [(p^i), 1]$ is initialized with all points p in a data set P with an extra 1 added to each point. For each iteration, a matrix X is found by

$$X = Qdiag(u)Q^T. \tag{2.59}$$

This matrix is used to find

$$M = Q^T X^{-1} Q, \tag{2.60}$$

which gives a measure for the distance of each point to the center of the ellipsoid. The index of the largest element in M is found as $idx = \arg\max M$ and the step size, β, at the current iteration, is a scalar, decided by

$$\beta = \frac{M[jdx] - d - 1}{(d+1)(M[jdx] - 1)}. \tag{2.61}$$

u is then updated as $u = (1-\beta)u$ and $u[idx] = u[idx] + \beta$, and the iteration continues until β gets under a desired threshold.

After convergence, the matrix A, which describes the shape of the ellipsoid and the center, c, is given as

$$A = \frac{1}{d}(P^T diag(u)P - c^T c)^{-1}, \tag{2.62}$$

and

$$c = uP, \tag{2.63}$$

which is used to describe the MVEE in (2.58).

Chapter 3

Platform

To evaluate the results of direct methods such as optical flow, it is helpful to analyze the tracking of actual data in the marine environment. In this thesis, the methods introduced have the purpose of detecting and tracking objects in marine environments, especially for the autonomous ferries milliAmpere 1 and its successor milliAmpere 2; two autonomous ferries that have been developed and tested in Trondheim, Norway. This chapter includes an introduction to milliAmpere 1 and milliAmpere 2, a description of the vessels that are apparent in the data, and a description of different scenarios in which the data has been recorded. Parts of the descriptions in this chapter will be related to the corresponding chapter in [15] as the same data is used.

3.1 milliAmpere 1 and milliAmpere 2

The milliAmpere ferries are autonomous ferries that are developed at NTNU by students and employees [46]. The ferries are among the first of their kind, and the autonomous ferries could potentially contribute to for instance urban marine environments in terms of efficiency. They could also replace bridges which can be time-consuming and expensive to build. This could also make transport more available. Another benefit of an autonomous ferry, compared to regular ferries, is that fewer humans need to interact with the system, and one person alone can observe multiple ferries at once as the ferries likely will be monitored at land. This will lead to lower costs in terms of salaries and could reduce the risk of human error if the ferries are made safe. milliAmpere 2 was first tested by the public in the fall of 2022 between Fosenkaia and Ravnkloa in Trondheim.

The ferries are 5 and 8 meters long respectively, which makes them significant in size in smaller channels, like for instance in the channel in Trondheim. The velocity of the ferries is 5 knots which is the speed limit close to the shore. Nevertheless, the speed and size make it necessary to make sure that the ferries travel safely in their environments. To fulfill the safety measures for such a vessel and

be aware of its surroundings, it is necessary with a variety of sensors. The ferries are equipped with optical cameras, infrared cameras, LIDAR, and radar which all can be used for tracking. In this thesis, however, only the optical cameras will be regarded as only direct optical methods are used. In milliAmpere 1, there are 5 optical cameras [47]. These create a 360 degrees view, and one can therefore track vessels behind, in front of, and next to the ferry. In this thesis, the front camera will mainly be used as the target for detection and tracking mostly is apparent in this camera. However, for further use, all cameras should be used.

Figure 3.1: milliAmpere 1 docked in environment 2, the channel in Trondheim. Image from [47].

3.2 External vessels

In the recorded data, different vessels are used to create scenarios that are suited for tracking. The vessels are of different sizes and have different properties that will represent vessels that are likely to meet in the common surroundings of the ferry.

One of the vessels used is pictured in Figure 3.2 and is called Gunnerus. This is a large boat that is owned by NTNU and is often used for research purposes. A boat of this kind may be a common sight in the surroundings of autonomous ferries. Gunnerus is large in size, making it easy to recognize and separate from the surroundings. The speed of Gunnerus is relatively slow compared to smaller boats. The movement of the boat between different frames will therefore be smaller, which may make it easier for tracking algorithms to keep track of the boat. The size of Gunnerus may also result in occlusion, as it can cover smaller boats, which is an interesting and important scenario to analyze.

Figure 3.2: Gunnerus in environment 1. Image from [47].

A second vessel that appears in the scenarios used in this thesis is Havfruen. Havfruen is a smaller boat than Gunnerus and is owned by a student organization at NTNU called Mannhullet. Due to the smaller size of the boat, it can maneuver at higher speeds than Gunnerus and may be harder to notice when further away in the image frame. Havfruen can be seen in Figure 3.3. The color of the boat makes it stand out from the background in many cases, and can therefore be an interesting target when tracking close to buildings and other detailed backgrounds.

The third vessel that will be present in the data is a jet boat which can be seen in Figure 3.4. This boat is smaller than the previous one and therefore includes fewer important features when tracking. The vessel's color will also make it blend in with the surroundings, making it harder to detect and track.

Figure 3.3: Havfruen in environment 2. Image from [47].

Figure 3.4: Jetboat seen in environment 2. Image from [47].

3.3 Environments and scenarios

The data that is used is sampled from two different environments. Both environments are located in Trondheim, but in other surroundings varying the conditions around milliAmpere 1. milliAmpere 1 meets different challenges in different environments which gives interesting problems for tracking algorithms. For all the scenarios, ground truth is provided from GNSS antennas mounted on the vessels.

3.3.1 Environment 1

Environment 1 is situated outside Piren in Trondheim. This environment includes open water in Trondheimsfjorden, where waves can disturb the tracking algorithm. Besides this, the background in this environment has few details, which makes it easier for boats to stick out from the background. This will make tracking and detection of different vessels easier. A map of environment 1 is shown in Figure 3.5 described by the green area. The targets that are involved in the image sequences in environment 1 are Gunnerus and Havfruen. The recording in this environment happened on the 4th of May 2021 from 09:44 to 13:41.

From the examination of the data from Environment 1, a particular scene stands out and will be referred to as Scenario 6 from this point forward. The scenario is illustrated in Figure 3.6. This scenario is particularly in focus as it most of the time represents a simple scenario with only Gunnerus present. The scenario also includes a significant maneuver by Gunnerus where a relatively large directional change is performed in a small time interval. This will be a good scenario to study later on with motion estimation and SFM. Along with Gunnerus, Havfruen is also present in this scenario. However, this boat will play a smaller role and be

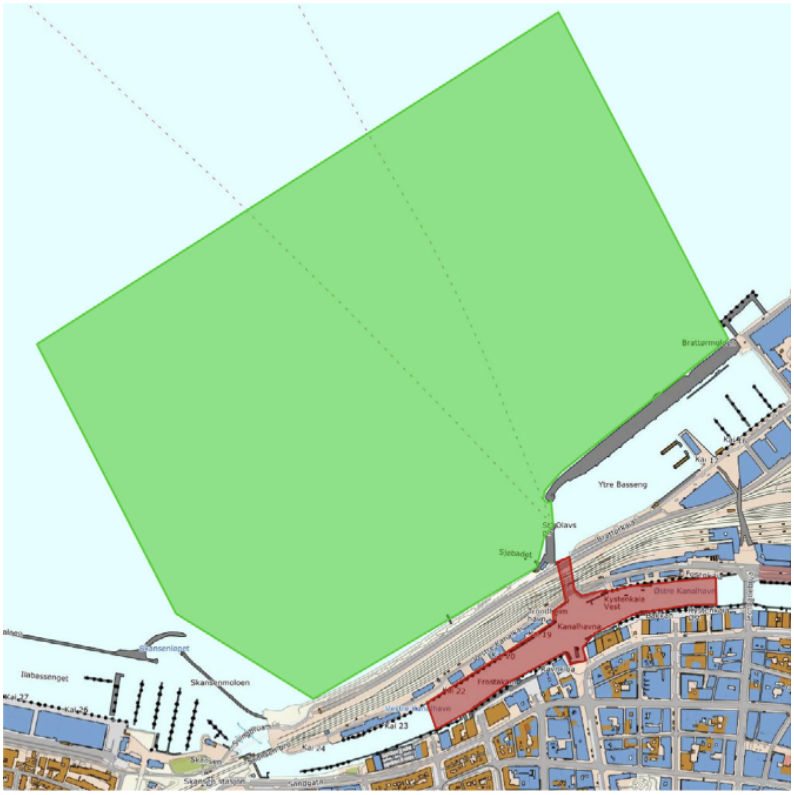


Figure 3.5: A map showing the two environments where the data is recorded. The green area shows environment 1 and the red area shows environment 2. Image from [47].

less visible.

3.3.2 Environment 2

Environment 2 is located in the channel in Trondheim. The data from this scenario was recorded on the 5th of May 2021 from 10:52 to 11:57. In this environment the background has significantly more features than the previous environment which may cause disturbances when detecting and tracking. As the channel is somewhat shielded from the open water, the water is much more still in this environment, and waves are likely less of a problem. The vessels that are used in this scenario are Havfruen and a jet boat. Gunnerus is not included due to its large size. In addition to this, boats are docked in the channel which is an interesting scenario for detection. The environment can be seen in Figure 3.5 as the area covered in red.

For this environment, one scenario has been chosen to analyze the detection and tracking performance. The scenario is referred to as Scenario 13. The scenario is shown in Figure 3.7. Here, both Havfruen and a jet boat are involved.

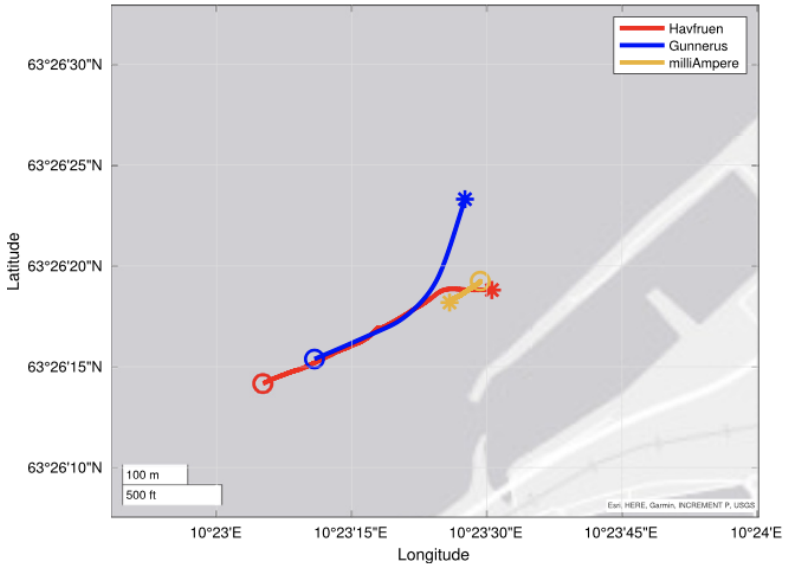


Figure 3.6: Scenario 6, showing environment 1 and the tracks of the included vessels. Circle illustrates the starting point and the star shows the endpoint. Image from [47].

Both boats are small in size and have unclear features which will make tracking harder than in Scenario 6. The colors of the boats are also different which may be interesting to analyze.

Also, the backgrounds are different in the two scenarios. The background in Scenario 13 is more detailed than in Scenario 6 which may cause problems when trying to recognize the objects of interest.

3.4 Data

At land, there already exist multiple data sets for the purpose of detection and tracking. One example is the KITTI data set [48] which is specially built for autonomous vehicles in traffic. In the marine environment, however, the amount of data is smaller, which can indicate that the focus has not been on autonomous vessels yet. When detecting and tracking by sea, one finds themselves in a different setting and can meet new problems like waves and new movements in the camera. The given data set was therefore made for this purpose and is recorded in Trondheimsfjorden.

The data is sampled as sequences of images. An example of an image is shown in Figure 3.8 which shows Gunnerus in environment 1. In addition, there is provided ground truth of the vessels with respective transformations for each frame.

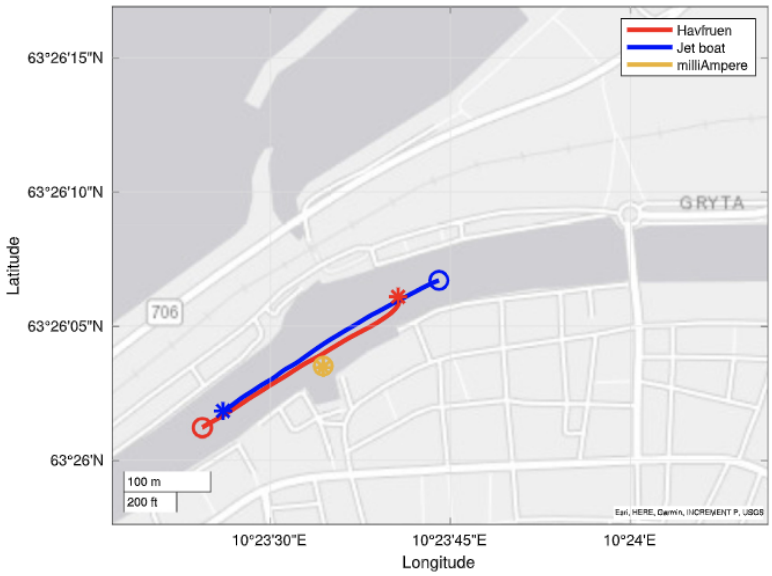


Figure 3.7: Scenario 13. The background shows Environment 2, circles illustrate the start of the track, and stars show where the tracks end. Image from [47].

This makes it possible to transform the ground truth of a vessel into the image frame for comparison.

Figure 3.8: An example of an image frame that is used for testing. The image shows Gunnerus in environment 1 and is recorded by milliAmpere 1.

Chapter 4

Methodology

4.1 Camera calibration

The camera projection matrix, as described in Section 2.5, for the camera used to create data in this thesis is already given in the data set. This matrix is normally found through a camera calibration process where an object with a known shape is used to get the correct output from the camera. This is, however, not necessary for this camera as it has already been given. On the other hand, the lack of insight in the calibration process can be a source of error but is not considered critical.

The lens that is used to create the data set causes distortion in the images. The distortion parameters are also given for the camera in this data set. The images before and after correcting the images in the data set are shown in Figure 4.1. The black area around the undistorted image will be cropped when displaying the results of this thesis.

Figure 4.1: A distorted image (left) and the same image without distortion (right).

4.2 Detection - optical flow

The technique in Section 2.2 is tested for background removal and detection of moving objects in the image frame. The code for this method is provided with the paper and the method is therefore not implemented from scratch. The implementation has been done in MATLAB. The implementation which originally included the Horn-Schunck dense optical flow method[49], was updated by replacing the optical flow method with GF. This was done due to the computation time of the algorithms as the performance of both algorithms was similar based on testing of both algorithms.

The method includes different variables which can be used as tuning parameters. A von Mises distribuiton can be formulated as $V(\mu, \kappa)$. This distribution directly affects the segmentation process of the method as $p(v_t, M_j)$ is modeled with von Mises. This is the probability of the appearance of a flow vector in a flow field. The parameter μ is decided by the estimated optical flow angle caused by the translation of the camera. κ is, on the other hand, a tuning parameter that describes the expected magnitude of the flow in the background. This can be tuned for the purpose of this task and will come in handy when avoiding the detection of waves in the water. Also, the new motion detection probability $p(M_j)$ can be used for tuning. When lowering this parameter, it tells the algorithm that it should be very certain that a new movement is a new object and not a wave or reflection in the water.

The implementation of the Gunnar Farneback (GF) optical flow algorithm for this research relies on the calcOpticalFlowFarneback function from OpenCV. It is essential to adjust the parameters of this function to ensure its suitability for marine environments. This function takes two images as well as the previous optical flow used for initialization. In addition, the function takes in a set of tuneable parameters.

The first parameter is the number of layers in the pyramid. This is intrinsically linked to the iterative loop process introduced at the end of Section 2.1.3, where the prior displacement field ($\tilde{\boldsymbol{d}}(\boldsymbol{x})$) is integrated into the algorithm. The pyramid structure helps to handle large displacements that could occur in dynamic marine environments. The number of layers in the pyramid directly corresponds to the number of iterations we perform to improve the estimate of the displacement field.

Next, the scale for each layer in the pyramid is essential for determining the relative sizes of the displacement fields that are being compared at each level. This helps adjust the algorithm to handle various spatial frequencies in the images, thereby assisting with the assumption of the second-degree polynomial representation of pixel regions.

The window size parameter aligns with the concept of neighborhood. Larger window sizes imply broader neighborhoods being considered for polynomial fitting and displacement estimation, which in turn influence the level of detail captured in the optical flow. More extensive windows may yield a smoother, more generalized flow, while smaller windows can capture more intricate flow patterns. A well-chosen window size helps balance between capturing the details of the flow and smoothing out the noise.

The $poly_\sigma$ parameter adjusts the blur of the image. This helps to reduce the effect of noise in the marine environment. Noise, in this setting, is meant small disturbances which are unimportant and can cause inaccurate overall estimations. This is disturbances such as water reflections and waves.

Lastly, the $poly_n$ parameter governs the degree of the polynomial used for approximating the flow of each pixel. It directly impacts how effectively one can align with the assumption of a second-degree polynomial representing the region. A larger $poly_n$ results in a smoother flow, albeit with the possibility of missing out on more detailed aspects.

In this research, an interactive tuning interface as shown in Figure 4.2 is used to adjust these parameters in real-time, examining their impact on the resulting optical flow. This empirical tuning process plays a crucial role in aligning our implementation of the GF algorithm with the theoretical principles discussed, and in optimizing its performance for our specific marine context.

4.3 Detection - YOLOv8

When using optical flow for detection it will be useful to compare the results with another different method. An example of a state-of-the-art method that uses machine learning to detect objects is YOLOv8 which is created by Ultralytics and is the eighth version of the YOLO networks. There is not yet released an official paper on this network. As the currently most efficient and accurate method, it is viewed as the most relevant method for comparison.

A YOLO network has to be trained to give reasonable results. YOLOv8 comes with different versions of pre-trained models which include the classes from the COCO data set [50]. In the COCO data set, classes for different vessels are included, and the pre-trained weights are considered sufficient for the detection in this thesis. One can pick only the desired classes in the COCO data set to make the detector only detects boats. There are different versions of the pre-trained models that come with YOLOv8 which vary in size and followingly also run time and accuracy. As this potentially will be used in a real-time system, the smallest model, named yolov8n-seg.pt, is used due to its small runtime.

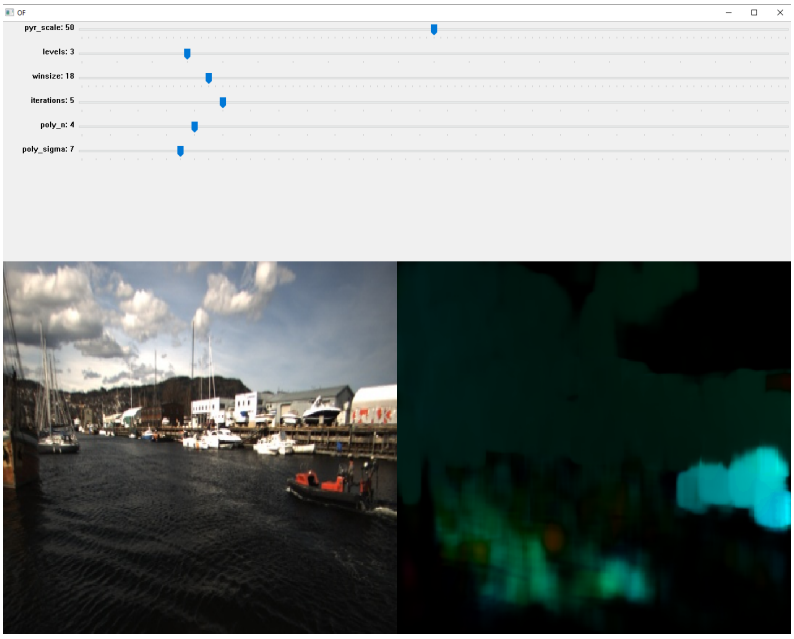


Figure 4.2: Tuning of the parameters in the GF function. The parameters can be changed along the image sequence, and one can see how the parameters change the result. The jet boat is standing out from the surroundings in the current frame.

4.4 Tracking

Tracking points between images in a sequence is a central part of this thesis. The tracking is done with the optical flow method LK described in Section 2.1.1. The LK algorithm that is used in this thesis is the OpenCV function calcOpticalFlowPyrLK. This function takes a list of points in the first image, the first image itself, and a second image as parameters. It also lets the user adjust the window size and the number of layers in the pyramid. It followingly uses the pyramid implementation which will handle large changes in the image well.

The tuning parameters of the LK function will have an impact on the performance of the tracking of the points. The parameters that will be used in this thesis are a window size of 25x25 and the number of layers in the pyramid is 4. The numbers that are used follow from the analysis done in the specialization project of Erik Daniel Haukås Moe [15]. In summary, a larger window size will decrease the accuracy of the track as it may include parts of the image that move with a different optical flow than the object itself. A smaller window size will increase the accuracy but will, on the other hand, make it less robust to large changes in the position of the point. The number of layers that are used in the pyramid will affect how the algorithm handles large changes in the image. However, it can be argued that the number of layers needed rarely exceeds 5 since, for instance, an image of shape 2000x2000 would be reduced to 125x125 in the 5th layer.

The value minimized by the LK algorithm in (2.2) can be used to measure the error of a tracked point. This error can be used to filter out points that are tracked inaccurately. After every point was tracked throughout the entire image sequence, points with a significantly higher maximum error were filtered out. More specifically, points with a maximum error higher than the mean maximum error plus one standard deviation were removed.

4.5 Initial tracking points

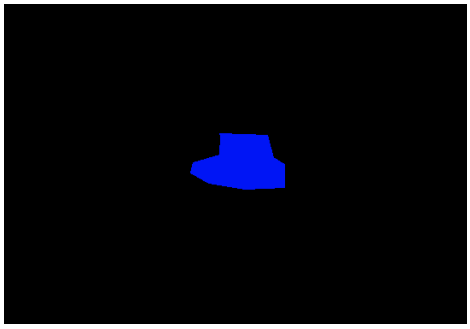
Initial points are needed to be able to locate points in an image sequence. The points are easier to track if the points are set to clear features on the object of interest. To find the features that are used for tracking, the OpenCV function goodFeaturesToTrack is used which is based on the Shi-Thomasi feature detection method as explained in Section 2.10. This function takes the respective image as an input parameter and has the possibility to add a mask to the image. This makes it possible to mask out the target and only find points that belong to it. When masking the objects of interest manually, it is important to ensure that the mask only includes the object and not the background which would give outliers. It may therefore be a better approach to have a small margin inside the object than to mask it exactly on the border.

Further, the function allows the user to specify the minimum distance in pixels allowed between the points and the threshold for the quality of the points. This makes it possible to spread the points out on the object and only chose points over a specific quality level. It will, however, always choose the best possible points, and it may therefore be a better approach to adjust the number of points parameter, which is also included in the function, rather than the desired quality. Also, the minimum distance between points has a great correlation with the maximum number of points used. This is because there are fewer possible points in a mask the larger the minimum distance gets.

When analyzing the pose of a target, the latest information is the most important information in the latest frames. When choosing points that will later be used for SFM, one will assure that the results are most correct for the latest frames. The initial points are therefore chosen in the latest image and tracked back in the earlier images to assure that the best points in the most recent frame are used. This is done to give the best results when analyzing the performance of the methods.

The masks that are being used are created by using functions from OpenCV that allows you to create a best-fitted polygon to points. The points are chosen manually and the polygon and mask are then made. The mask can then be stored for further use and will provide consistent initial data for the tracker. An example of how a mask is provided is shown in Figure 4.3. Here, the points are chosen

(a) Choose points.



(b) Create mask.

Figure 4.3: The process of making a mask. First, choose points around the object and then create a mask from the resulting polygon.

manually by the user, and a mask is created from the respective points.

4.6 SFM - OpenCV

The OpenCV SFM module is used to do reconstructing in 3D. This module is chosen as it can be used in Python which was seen as convenient as it would be easier to use with the rest of the project as well as being efficient. It is also possible to use the SFM module in C++ which is the language OpenCV is written in. This makes it versatile for combining with new projects in the future.

The SFM module is not in the standard OpenCV library. To access the module one would have to include the OpenCV contrib library. This requires the user to build OpenCV from source while including the path to the OpenCV contrib where the contrib modules are. This is a cumbersome process as there is not much documentation of either the module or the installation process. Nevertheless, while building the OpenCV library, one can include the path to Python and will get access to the desired functions. For the SFM module to work the dependencies, Eigen, Google Flags (gflags), Google Log (glog), and Ceres-solver must be installed.

The SFM module includes four different versions of the reconstruct function which includes the pipeline for reconstructing a scene in 3D. In two of the functions, the input parameters are the desired images and a K matrix. These two functions control the whole pipeline themselves. They, therefore, find their own point correspondences using SIFT and will not give the user the option to find their own correspondences with another method. In this case, it is desired to use LK to find the point correspondences between frames. Therefore, other versions of the function reconstruction are more desirable to use.

In the two other functions, it is possible to find point correspondences in advance and give them as an input parameter along the K matrix. In this way, one

can combine LK with SFM. When using this function directly, it will change the K matrix in the bundle adjustment process with a refinement function. As the K matrix already is defined in the data set, this process will give a wrong result. The final refinement step is therefore omitted to avoid this. It should be mentioned that bundle adjustment still is performed. The output from this function is the 3D point cloud as well as the respective R and t matrices for each camera view.

The approach for reconstruction with multiple views used in the OpenCV function is incremental. An incremental approach means that it initializes the reconstruction with two views, desirably to have the greatest baseline, and then continues to add more views incrementally by finding correspondences in the new view and performing camera motion estimation, triangulation, and bundle adjustment [51]. The desired amount of views is then added to the reconstruction.

An additional change has been made to the original reconstruct function in the SFM module. The original function used the two first frames, and point correspondences, as a base for the result. This will not fully exploit the possible baseline made with a larger number of images. In an attempt to make the reconstruction more accurate, a new input parameter has therefore been added where one can select the desired keyframes. The keyframes will make a basis for the reconstruction where new images are added. These are later bundle adjusted to fit the point cloud.

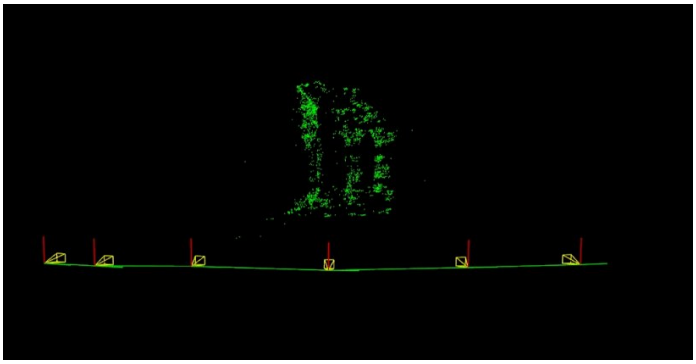
The OpenCV SFM module comes with some examples of its use. To give an indication of how the result will be looking, an example of a use case from OpenCV is shown in Figure 4.4. Here, the function that takes images as input is used. The functionality of the functions is however the same and will give equal results with equal points. The different camera views are shown in the image and the reconstruction indicates the shape of the temple.

A visualization library in OpenCV called Viz is used to display the 3D point clouds before the cloud is transformed into new coordinate systems. This is because the library easily can display intuitive camera views together with the point cloud. Viz is included in the OpenCV contrib and is installed together with SFM. This is the same library as used in Figure 4.4. After being transformed, the point cloud will be displayed with the Python library matplotlib.

4.7 Correcting result from SFM

Figure 4.5 illustrates the setup of the camera that is used for the data in this thesis. The camera is mounted with a 15 degrees slope pointing downwards toward the ocean. This would give a deviation when calculating the change in yaw. However, this error is removed with a fixed transformation from the camera to the body frame. This transformation also converts the result from camera coordinates

(a) The input images given to the reconstruct function.



(b) The resulting 3D reconstruction of the object. The elements connected by lines are the different camera views from the images above.

Figure 4.4: The illustrations show images and the reconstruction of the Middlebury temple which is a model of the "Temple of the Dioskouroi" in Agrigento. Both images are taken from [52].

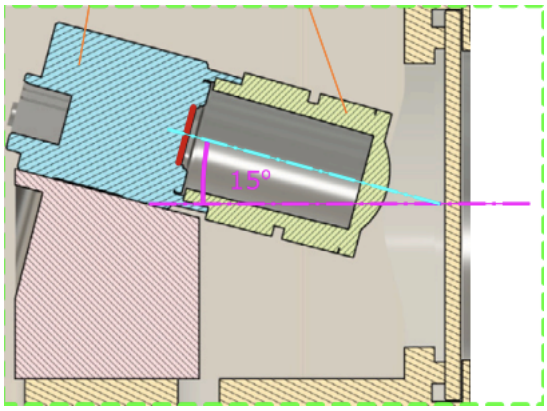


Figure 4.5: An illustration of the tilt of the camera on milliAmpere 1. The image taken from [47].

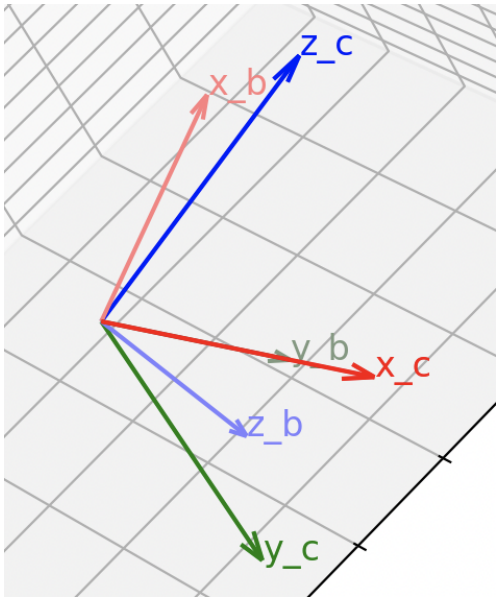


Figure 4.6: The result of the transformation from the camera coordinate system (subscript c) to the body coordinate system (subscript b). The 15 degrees slope can be recognized. Also, notice how the z-axis (body) replaces the y-axis (camera).

to regular NED coordinates. By correcting the slope, one will get more correct estimates when looking at movements relative to milliAmpere 1. The new body coordinate system after the transformation is applied to the camera coordinate system is shown in Figure 4.6.

There is also a 20 cm translation in the x-direction between the camera mounted on milliAmpere 1 and the center of milliAmpere 1. This may cause a small deviation between the estimated yaw angle and ground truth. However, as the results only are up to scale, it is not possible to correct this deviation. Some of the error may therefore be caused by this, but will, however, only make the results slightly inaccurate.

4.8 Positive and negative direction for yaw

When looking at the change in yaw angle, a consistent notation is important. Figure 4.7 illustrates the interpretation of a positive and negative change in heading. This also means that 0 degrees in the heading are defined as moving along the north axis. When working in a body coordinate system fixed at milliAmpere 1, 0 degrees means that the target is moving or heading in the same direction as milliAmpere 1.

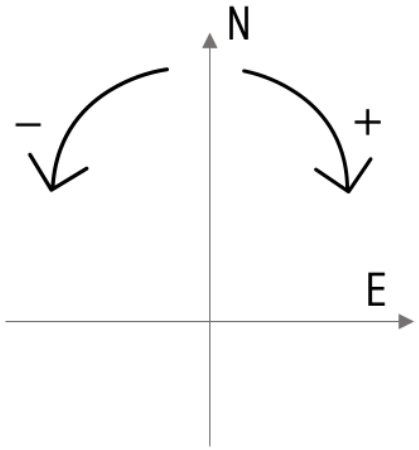


Figure 4.7: An illustration of how positive and negative yaw is defined in this thesis in the NED frame.

4.9 Estimating relative yaw rate from SFM

SFM gives rotation matrices as output. These matrices give information about the relative yaw angle between milliAmpere 1 and the target. Without scale, it is hard to decide the absolute heading of the target. However, it is possible to find the relative change in the target heading by the information from the rotation matrices.

Section 2.13 presents how yaw can be extracted from a rotation matrix. The relative change in the heading, or relative yaw rate, can then be extracted by finding the yaw from the resulting rotation matrix after multiplying the inverse rotation of the first rotation matrix with the second rotation such as $R_1^T R_2$. The relative yaw rate can be caused by both a camera rotation and a rotation of the object of interest. The frame rate of the camera is 0.2 seconds. As the yaw angle per second gives the yaw rate, the results are multiplied by 5 to get the rate from the change between the rotation matrices.

4.10 Simulated real-time scenario for yaw rate estimation

When estimating the relative yaw rate in a real-time situation, one would like to always estimate the yaw rate for the latest image at all times to have the newest information about the target. To obtain this one would have to include the newest image in the SFM reconstruction. With the OpenCV reconstruct function one will only be able to perform SFM on a newly updated set of images. By using a moving window of images, iterated by one at each iteration, one will be able to simulate

a scenario where a new image is received and a new reconstruction is done to retrieve the information about the latest relative yaw rate. This is a cumbersome process where the information is calculated multiple times, and the usage in a real situation may demand a more efficient algorithm. However, as the points are found in the last image one can assume that the last result would be accurate and the method could give a good representation of how the performance would be in a real-time scenario.

4.11 Estimating the center of the target

When using an image sequence to reconstruct a target with SFM, one would usually not have a 360 degrees view of the target. This will give a point cloud that does not match the shape of the real target and it will be hard to find the center of the vessel. An error in the vessel center can propagate when trying to estimate the position of the target, and one will try to find a candidate point that is located as close to the volumetric center of the object as possible.

The method that is used in this thesis to estimate the center and shape of the target is the MVEE. This will create an ellipsoid in 3D which will use the current knowledge of the target to estimate the shape, and hence be able to find the real center. The method that is implemented in this thesis is Khachiyan's algorithm which is described in Section 2.15.

Another approach could for instance be to use the mean of the point cloud. This gives the mass center of the point cloud which could be a good approach if the point cloud was uniformly distributed. In this case, it will, however, emphasize the parts of the vessel with the most features which will not necessarily be in the center.

A method that is based on including all points in a data set may have some weaknesses. For example, the method may not be very robust if outliers appear in the 3D reconstruction. This would give a wrong ellipsoid and hence a bad estimate of the center. Nevertheless, outlier removal is used in this thesis to prevent this from happening. Also, since the results presented in this thesis will be highly visual, one can, by inspection, see how well this method works.

4.12 Estimating position of the target from reconstruction

In addition to heading, one can also obtain information about the position of the target by looking at the results from SFM. The position of the target is given as relative to milliAmpere 1 up to scale. The position of the target is best estimated

as the center of the object. The center of the object will be found with the method presented in the Section 4.11. Information about the course will also be given as the direction between two positional measurements.

To determine the scale, or the actual distance to the object, one would have to use techniques beyond what is used in this thesis. However, since the ground truth gives the actual distance from the target to milliAmpere 1, one can use this information to obtain scale in the coordinate system of the reconstructed point cloud. This will make it possible to obtain a scale for the resulting point cloud and views, making it possible to analyze the position estimates given by SFM and the tracking from LK.

An alternative way of determining the target's position is to track a point in the image frame using LK and directly project the point to world coordinates. This is possible if the scale is available and will be tested and compared to the previous method with SFM.

4.13 Ground truth

A ground truth is needed to analyze the pose estimation performance. The ground truth will in this thesis be based on the measurements from the data set given and will be used for comparison with the estimates. The ground truth originates in GNSS measurements for the position of the targets and milliAmpere 1 in the world frame related to Piren, as well as IMU measurements for the orientation of milliAmpere 1. These measurements may have some errors due to inaccurate filtering which will give a varying quality of the data used at ground truth in this thesis. The uncertainty in the equipment also gives oscillations which can make the ground truth inaccurate for slowly moving objects in terms of heading when the distance between the consecutive position measurements is small. This will be attempted smoothed out to make the data more accurate. Nevertheless, as the ground truth may be somewhat inaccurate, the more important it is to see if the estimations follow the same trends as the ground truth which will give information if the methods are working or not.

4.13.1 Ground truth for change in heading

The vessels that are used as targets in this thesis do not have any IMU measurements and therefore lack any information about the heading. The data from the GNSS measurements is therefore used to create ground truth for the heading of the targets. It is assumed that the target is moving in a forward direction, meaning that it is going in the same way as the heading in the data in this thesis. The heading can therefore be calculated by looking at the angle of the vector between

two consecutive position measurements. This will make the heading and course the same. One factor that can make the heading differ from the course, which is more or less what is used in this thesis, is currents in the ocean that can somewhat change the course. However, this is not regarded as a big enough problem, and the lack of data makes it hard to deal with. In the case of milliAmpere 1, the available data gives the heading of the ferry based on data from an IMU. The same assumptions will therefore not be made for milliAmpere 1.

4.13.2 Transformation of ground truth

When using SFM to get information about a target, the results will only be up to scale. This means that, for example, the distances between camera angles will not be representative of the world coordinates and the size of the target will not be correct. To compare the results in the world frame would therefore not be possible as one for instance cannot apply the static translation from the camera to the body of milliAmpere 1 to the result.

What is possible, however, is to move the ground truth from world coordinates to the coordinate system fixed on milliAmpere 1. This will describe the movement of the target from the perspective of milliAmpere 1 and one can find relative information from the camera and compare it to the relative ground truth.

4.14 Pre-processing of rotational measurements

As mentioned, in [46], the rotational measurements regarding the pose of milliAmpere 1 are measured by an RTK-GNSS receiver with two antennas providing position and heading. In addition, it is also equipped with an Inertial Measurement Unit. milliAmpere 1's navigation system fuses these measurements to use in an alpha-beta filter. This intends to produce a smooth and accurate pose estimation in six degrees.

Chapter 5

Results: Detection

The following chapter will present the result obtained from testing the setup for detection on objects close to milliAmpere 1 with both optical flow and YOLOv8. The methods will then be compared to see which will perform best. The performance will mainly be evaluated based on visual inspection of the masks and detections of the surrounding objects, but also other metrics such as runtime and compatibility.

5.1 Detection with optical flow (GF)

The results of detection with optical flow come from tuning and testing as described in Section 4.2. The values of the parameters which has been used in the OpenCV GF function are shown in Table 5.1, and are found by empirically analyzing which values gave the best flow estimate for detection purposes.

Table 5.1: Tuning parameters for GF for detection with optical flow. The values are found empirically.

Parameter	Value
Layers in pyramid	3
Pyramid scale	0.14
Window size	16x16
Iterations	4
poly_sigma	14
poly_n	5

The tuning is done for the sequence illustrated in Figure 5.1. In this sequence, the movements of milliAmpere 1 are small and consistent, meaning that there are no rapid changes in movement, and the movements of the boat will differ from the movements of the background. The resulting detection shows that the method manages to locate the moving boat well and keeps track of it during a longer se-

quence. One notices, however, that the mask is somewhat outside the boundaries of the boat. This results from a larger window size where some of the water is given the same flow as the boat.

Figure 5.1: The detection of Havfruen in a sequence of images. The left image shows an image early in the sequence while the image to the right shows how Havfruen has moved closer to milliAmpere 1 later in the sequence. The masks are covering Havfruen well, with some deviations. The camera movements are more or less uniform during the sequence.

A part of the algorithm is to estimate the movement of the camera to subtract the flow in the background. An interesting case to analyze would therefore be to look at a scenario where the camera has a rapid change in movement, for instance, when milliAmpere 1 rotates in a new direction. This is illustrated in Figure 5.2. In the leftmost image, the movement of milliAmpere 1 has been continuous between multiple consecutive frames. In the next image, to the right, the movement changes direction. The method then struggles to estimate the movement of the camera accurately, and followingly, some movement in the background is included in the detection. This is not desired behavior from a detector. One positive side of this example is that it detects the jet boat, which does not completely stand out from the surroundings.

To further address the problem of a moving camera, the detection of Gunnerus in environment 1 is attempted in Figure 5.3. In this environment, waves affect the stability of the camera to a higher degree than in environment 2. The figure illustrates how the detection is affected by waves. Also, the resulting mask of Gunnerus is somewhat incomplete which is caused by the size of the areas with similar color on the hull of the boat. The detected spots in other parts of the image are likely caused by changes in brightness or reflections of light in the water and the change in camera motion. Change in brightness is also a possible source of error when using optical flow for detection, and, for instance, a light spot in the water can be misdirected as a moving object. The masks which do not mark the moving object could be filtered out by tuning, for instance, the probability of

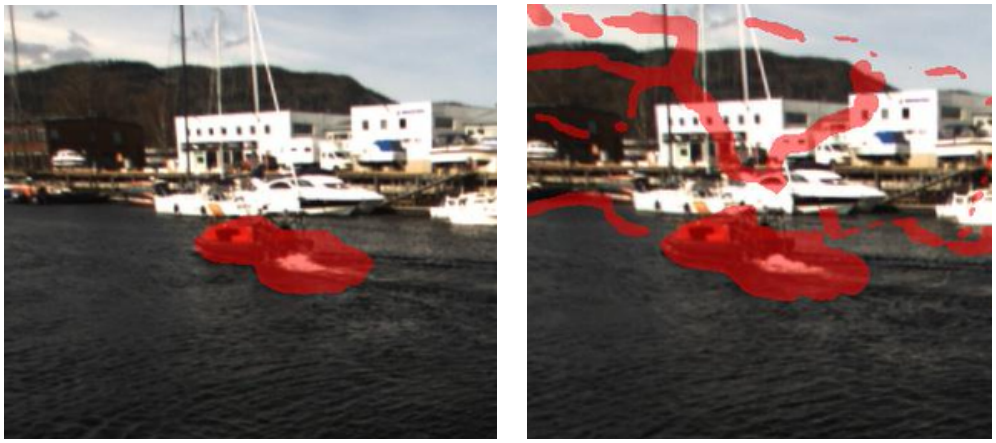


Figure 5.2: An illustration of the detection of the jet boats with the optical flow based method. The images are consecutive in sequence. The detection in the left image is more or less accurate. Between the images, the rotation of milliAmpere 1 changes direction, and the method does not handle this well.

detecting new objects or the GF algorithm differently. Nevertheless, it is hard to find a generalized tuning that would work for scenarios, and one would have to use some case-specific parameters for it to work well.

The efficiency of the algorithm is a bit varying depending on the size of the input images that are used. When using the true size of the images, the runtime varies from 3 to 5 seconds on a normal CPU. When reducing the size of the image to half the size, the runtime is lowered to an average of 0.5 seconds at the expense of the detection quality. If the system is used in real-time, these runtimes can be considered insufficient and will cause a delay in the system. However, the runtime could be lowered by optimizing the algorithm and running on GPU. Nevertheless, this is out of the scope of this thesis.

A positive side of the optical flow based method is that no training or information is needed in advance before running the algorithm. This means that it can detect everything that may appear in the water without any previous information about what an object may look like. The method is therefore versatile when it comes to the detection of objects. It also only detects objects that are moving. Docked and still-standing objects may not be equally important to analyze and may only be considered a general obstacle that should be avoided.

Figure 5.3: The detection of Gunnerus using optical flow in environment 2. The movements in the camera are highly affected by waves which are reflected in the detection result.

5.2 Detection with YOLOv8

YOLOv8 is also tested for comparison with the method from the previous section. A YOLOv8 segmentation model is used which allows for displaying both the detection in the form of a bounding box and the segmentation of the object. The segmentation can further be used as a mask for feature extraction and tracking. As a pre-trained model has been used in this thesis, no results from training will be included.

In Figure 5.4, the YOLOv8 network is tested on data from environment 2 in the channel in Trondheim. The results show two images where Havfruen is far away and close to milliAmpere 1. In the image to the left, the confidence level of the detection of Havfruen is much lower than in the image to the right, with 33% and 48%, respectively. The percentages are still low, but this shows how the detector's confidence increases as the object becomes clearer. This is expected as the detection model is based on features from a training data set which becomes more visible in the right image.

Another way this result differs from the detection algorithm based on optical flow is how boats that stand still also are detected. This will be beneficial as it gives more information about the surroundings. It can, however, be argued that it is necessary to care about boats that are docked. Nevertheless, more information is always desired, and the tracking of docked boats will not be problematic. In the given detection results, the quality of the detection of the docked boats varies. A general observation is that the boats in the areas with varying detections blend in with the background. This is an example of how YOLOv8 can miss out on information when, for instance, lighting condition and color makes the objects

less visible. This could, however, be fixed by using larger pre-trained models with better detection accuracies. The size of the models will give a trade-off between speed and accuracy. In addition, the smallest model often manages to classify the objects, even with some flaws, while the larger models are somewhat more strict when making decisions.

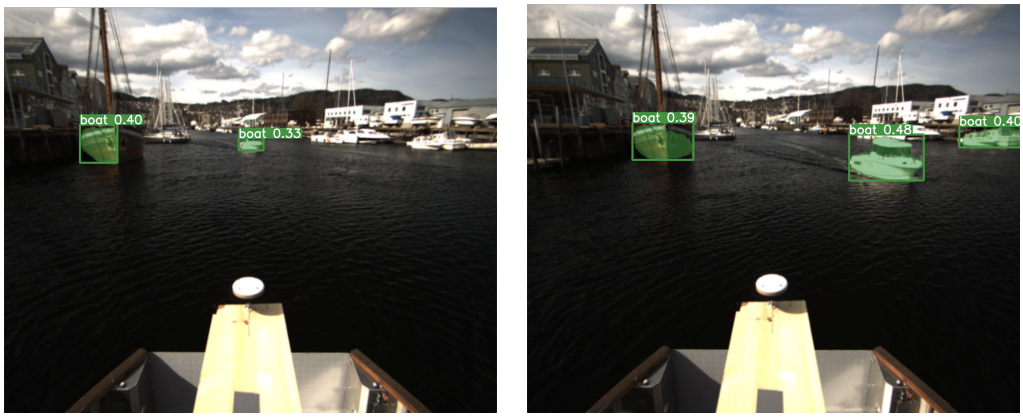


Figure 5.4: Detection of boats in the channel in Trondheim. Both Havfruen and the still-standing boats are detected and segmented with high precision. It is observed that the confidence of Havfruen being a boat is increased as it approaches milliAmpere 1. The boats in the first frame are generally of low confidence as they are less clear.

In Figure 5.5, another example of using YOLOv8 is shown. The figure shows images of Gunnerus when located far away and close to milliAmpere 1. In both cases, the detector manages to locate Gunnerus. When looking at the detection confidence, however, one notices that it gets much more confident of the detection when Gunnerus is closer to the camera with a confidence of 50% compared to 90%. This is consistent with the result in Figure 5.4. It also shows that tracking can begin when a boat is far from the camera if lower confidence is accepted.

The mask and an illustration of the points found in the mask are shown in Figure 5.6. All the points are located on the target, indicating that such an automatic process of finding points works well.

Another important aspect when comparing the two methods is the runtime for the different algorithms. As mentioned, the method based on optical flow has a large runtime. This is due to non-optimal implementations of the different parts of the method and more heavy algorithms. The runtime of YOLOv8 is significantly less and can run in milliseconds for each image on a regular CPU. This is a huge benefit if the method is used in a real-time system, and one would like to have the information as fast as possible. Also, the YOLO network does detection on a single frame, while the optical flow method needs two frames to analyze the movement

Figure 5.5: The detection and segmentation of Gunnerus when the boat is far away and relatively close to milliAmpere 1. One notice that the segmentation is precise and would be a great mask for finding key points in both cases. The probability of the classification is, however smaller when the object is less clear. The network also classifies the visible part of milliAmpere 1 as a boat. This can, however, be filtered out, for example, by looking at the optical flow in the mask or adjusting the threshold for classification.

in the image. The detection will therefore happen faster, which is desirable.

The masks from YOLOv8 are seemingly more precise than those from the method based on optical flow. In addition, it runs faster and is, therefore, more suited for use in a real case. One benefit of the method based on optical flow is that there is no need for training data, and it can detect objects that are not included in a data set. This can make the method more robust in some scenarios. However, with a suited data set, it may be possible to train YOLOv8 to detect objects in the water in general. This is, however, a more cumbersome process and is not in the scope of this thesis.

Figure 5.6: A mask of Gunnerus and key points in the mask from the detection in the right image in Figure 5.5. The mask of Gunnerus is chosen manually, as there is more than one mask in the image. The mask fits Gunnerus well, and all points are located on the boat. The points are found with the OpenCV function goodFeaturesToTrack.

Chapter 6

Results: Tracking, Structure From Motion and pose estimation

The following chapter will present the results from tracking and structure from motion and the results for the pose estimation of targets surrounding milliAmpere 1. The pose estimates mainly regard the relative yaw rate and position of the target. The points that are used for tracking are mainly found by manually creating a mask around the target and not the results from Chapter 5.

6.1 Tracking with multiple points (LK)

To find points that could be used to initialize the track, a mask is created around the target boat using the method as described in Section 4.5. An example of a track is shown in Figure 6.1. The points used for initialization are shown in the bottom right image. The following images show the tracking of the points in a sequence of images.

A visual inspection shows that LK manages to track the points well between images. The accuracy of the matching points in the different images will affect the results of the 3D point cloud from SFM, and one keeps track as accurately as possible when good features have been chosen. As seen in Figure 6.2, the points match each other well in most cases between images which indicates that the point correspondences will be suitable for SFM. Nevertheless, there will be some outliers, especially in areas with few features. An example of this is the lower green correspondence in the figure where the point is wrongly matched as it is located in a uniform area. This is a commonly known problem in computer vision known as the aperture problem [53] where it is hard to determine a movement when looking at an isolated area. This shows the importance of good features on the object and a well-tuned tracker. A poorly tuned tracker would cause drift in the points, which is not desired as the track happens over multiple images. The

accuracy will also be important for an accurate 3D reconstruction with SFM later.

Figure 6.1: A sequence of images where LK is used to track points. The initial points are selected from a mask in the bottom right image. In the sequence, images two seconds apart have been chosen to be able to see the difference between the images.

In Figure 6.3 shows Havfruen when being far away from and close to milliAmpere 1. The figure indicates that it is hard to tell, by visual inspection, if the tracked points in the two images are located in the same spots as Havfruen only appears as a white dot in the left image. Firstly, one can argue that this makes it harder to get accurate tracks and estimates of objects that are further away. Secondly, it shows how an object can change a lot during a sequence which is a reason for finding features in the last image when choosing initial points for tracking when testing SFM later on. This would not be optimal for real-time performance but will be done to test the performance of the methods. Another problem when tracking using optical flow is when a target moves so much that the features in the last image are no longer visible in the first image, for instance, 180 degrees. This will also happen in the scenario of occlusion, where another object covers some parts of the object. Such scenarios will not be included in the reconstruction results in the following sections as it is not considered in the scope of this thesis. However, one should be aware that it could be a problem.

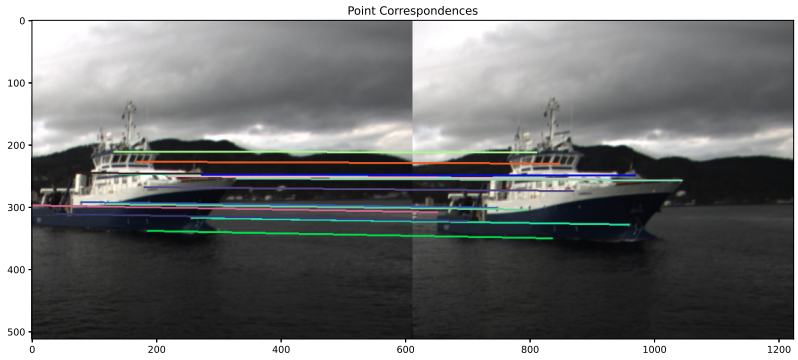


Figure 6.2: Illustration of the corresponding points in two different images. Each colored line represents a point correspondence. The images are cropped to increase the size of the target. Only a few points are selected to avoid too many lines in this illustration. In other cases, more points will be used.

Figure 6.3: Two images taken at different points in time from the same sequence. In the first image, the features are less clear than in the second. It is, therefore, hard to say if the features tracked in the sequence always correspond when the track is happening over an extended period of time. However, the points are always attached to the target and not disappearing in the surroundings. Also, the track happens over time, and the features in the left and right images will be more or less in the same spots.

Figure 6.4: The image sequence used to reconstruct the object in the center of the images with SFM. The images are taken with different angles to better view the object. The object's points are found using a mask in the first image in the top left corner and are tracked using LK. The images are taken with the camera of a phone.

6.2 3D reconstruction by SFM with moving camera

The accuracy of the point correspondences from LK shows potential for further use in SFM. SFM is, in general, a method to use when the camera is moving and the object of interest is standing still. To test the method in a normal use case, photos are taken of a random object on land while moving the camera. This will be the same as if milliAmpere 1 moves around a still-standing boat in the water.

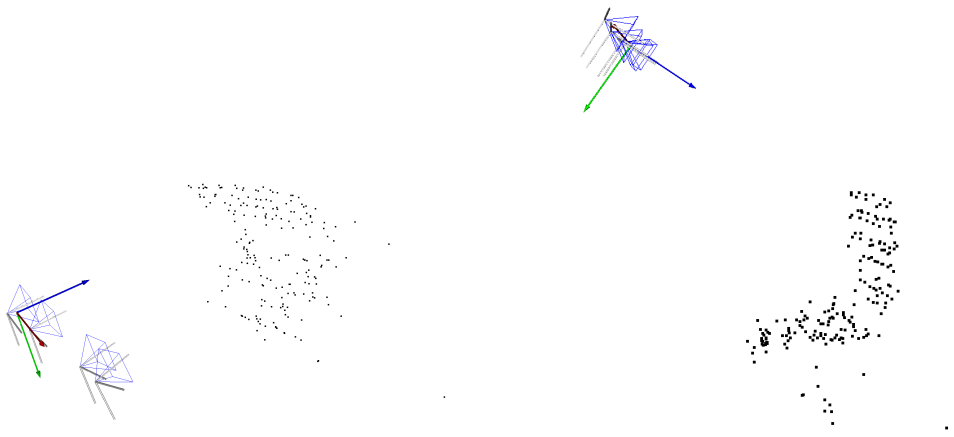


Figure 6.5: The resulting point cloud of the same object as in Figure 6.4. The reconstruction clearly shows the contour of the open box in 3D with some outliers likely to be poorly tracked points. The camera views are illustrated as the pyramids, and the coordinate system illustrates the origin.

In Figure 6.4, one can see an example of the tracked points, which seem to correspond in most cases. The points that are used are found by using a mask the same way as for the boat in the previous section, and the point correspondences are found by using LK. In Figure 6.5, the corresponding point cloud is shown with the respective camera views illustrated as pyramids. The object's shape is well preserved in the point cloud, and one can see that the camera view seems reasonable when looking at the images. One should notice that there are some outliers in the point cloud. These are poor matches or poorly tracked points and are mainly located on the black dashboard in the box. The features in this area are not as good as others, and it is reasonable that LK will struggle more when tracking these points. Where the features are more distinct, there are fewer outliers which is good. One could consider using methods for removing poorly tracked points at an earlier stage by comparing the quality of the matches between images. This would ensure that the matches used in SFM remain good which is important as the method is somewhat vulnerable to outliers. In general, the more outliers, the higher the risk of bad reconstructions and bad pose estimates.

Reconstruction of objects using SFM and LK seems like a good approach for creating point clouds of entities that operate close to milliAmpere 1. In the other case, however, one has both moving cameras and a moving object. It is, therefore, interesting to see how the algorithm performs in such a scenario.

6.3 3D reconstruction of boats with SFM with a moving camera and moving object

When creating a 3D cloud of an object in an image from the camera mounted on milliAmpere 1, both the camera and the target may cause relative movements. This will differ from the traditional use case for SFM as the scene normally would be static relative to the moving camera. In practice, however, the two scenarios will be perceived equally by the SFM module. When the target is rotating relative to the camera, this rotation could be described as a rotation of the camera around the object. The information of the relative movement both caused by the camera and the object of interest will therefore be included in the exintrics of the camera poses. This can be further used for pose estimation.

Figure 6.6: The images used for reconstruction of Gunnerus with SFM. The images are taken 3 seconds apart in the same sequence to be able to observe movement between the images. A small number of images are used to better see the camera views. A larger amount of images with a smaller interval between them will be used later with similar results.

Another difference from the scenario in the previous section is how the 3D point cloud will display the shape of the object of interest. In the previous example, the camera gets a good view of the object from multiple angles. It is then able to show the shape of the object, and one can clearly see that the point cloud looks like the object in the images. In this case, the camera, in most cases, only takes images from one side of the boat unless the boat rotates rapidly, which is rarely the case. Such a rapid change could also cause problems as LK would not be able to find matches as the boat completely changes from the first to the last image. SFM will therefore only be able to, to some degree, reconstruct the boat's shape. Nevertheless, it will be able to show the contours of the boat where the key points are registered and how the boat is positioned relative to milliAmpere 1, at least up to scale.

In Figure 6.6, one can observe the points used for SFM. These points are tracked with LK as in the previous examples, where the points in the most recent frame are found from a mask around the object using a key point detector.

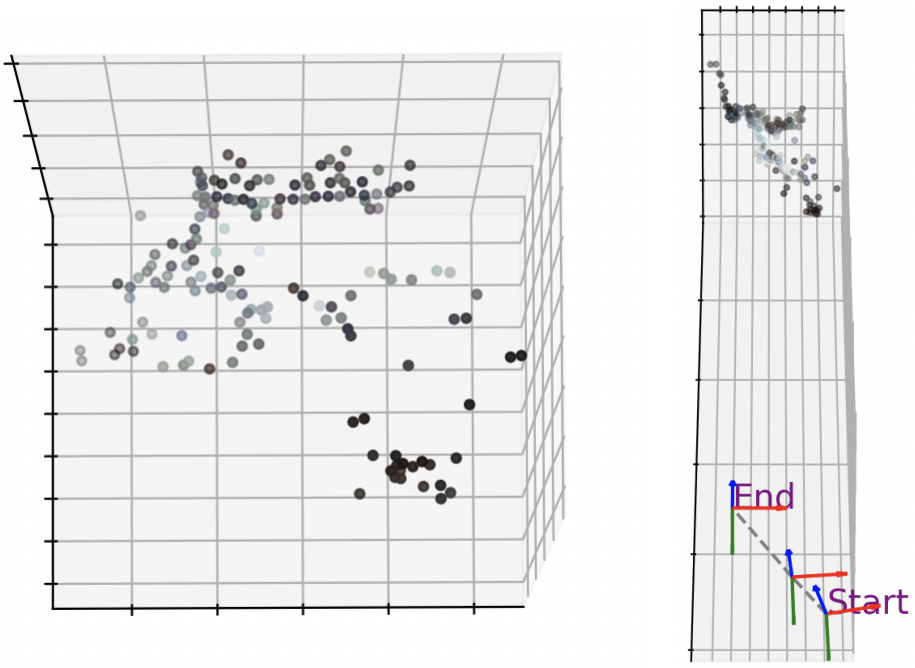


Figure 6.7: The SFM reconstruction of Gunnerus in body coordinates. The reconstruction clearly shows the boat from one side and manages to reconstruct the shape of the boat well. The camera views in the left image point towards the blue arrow. The point's color represents the colors of the points in the image. As these are key points they are often colors that are in contrast to the rest of the object. The colors are, therefore, shades of gray.

Figure 6.7 shows the resulting point cloud along the different camera views that are caused by movement in both the camera view and by the target. The contour of the boat is clear in the point cloud and the features are well represented. The quality of the point cloud will later be confirmed by looking at reprojection errors. The estimated camera views are showing the relative movement between the boat and the camera. The camera position in the first image is the one furthest away. The camera poses can be confirmed by looking at the path of Gunnerus in the image sequence as Gunnerus moves closer to milliAmpere 1 while having a relative rotation to the left. The movement of the camera views could also be illustrated as a moving point cloud with inverse movement. This will present the relative movement as a movement of the target in contrast to the movement of the camera. This is illustrated in Figure 6.8. One can recognize the large transition between the second and third camera view and the change in rotation of the first and the second view in Figure 6.7. Further analysis of the quality of the reconstruction will be done later on in this chapter.

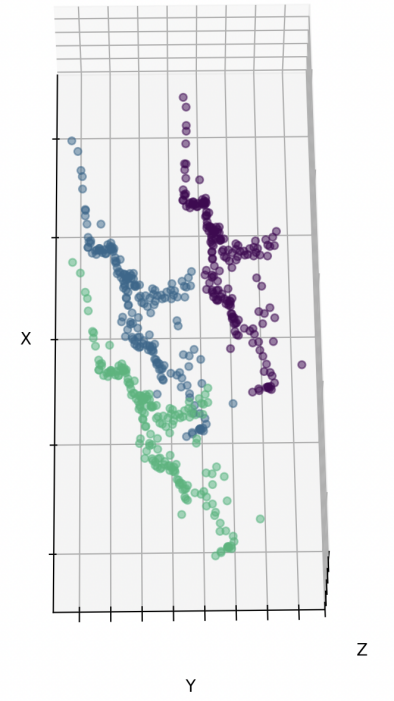


Figure 6.8: Result of describing the relative movement as a movement of the target in contrast to a movement of the camera. The point clouds show how the movement of the camera can be transformed into the relative movement of Gunnerus while keeping only the initial camera position. The colors are added to separate the different point clouds.

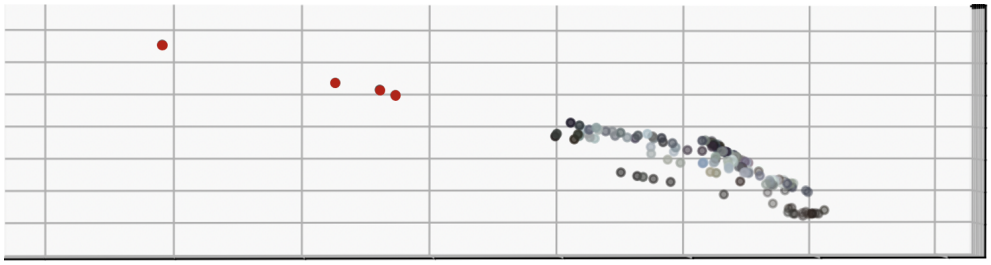


Figure 6.9: An illustration of detected outliers in a result from SFM. The outliers are marked in red. The outliers are removed when using the point cloud for estimation.

6.4 Outlier removal

When using SFM there will occur outliers in the point cloud. Figure 6.9 illustrates how outliers may appear and that they can be detected. A 3D reconstruction may not necessarily always have outliers, but if it does, it can affect the results. However, if the outliers are detected they can be removed from the set of points. This will allow, for example, the estimates of the center to be better, as the methods used may be affected by outliers, and the resulting estimates to be more accurate.

6.5 The problem of ambiguity

Ambiguity is a problem that can appear if the images that are used for reconstruction are too similar or the points that are tracked are too close such that multiple solutions exist for the extrinsics of the camera views. An example is when a majority of points are in the same plane in the image. In these cases, the movement of the object is wrongly described for the use case in this thesis, and one would get a wrong result when estimating the pose of the object.

Figure 6.10 illustrates an example of a case where ambiguity is present. The figure shows two different reconstruction processes of Gunnerus with the same amount of points. The only difference is that the minimum distance allowed between the key points is 4px higher for the images to the left. The interval between the images used is 1, meaning that the images are taken 0.2 seconds apart. This will result in a small baseline.

The figure shows two almost equal scenarios where the camera views to the left are looking at the point cloud as one would expect from the images. In the right images, however, the camera views look at the object from the opposite side. It is impossible to find points on the side of the boat which is not present in the camera. Even if the solution to the right is visually incorrect, it is theoretically correct. It is, therefore, hard to decide which one is correct in the reconstruction

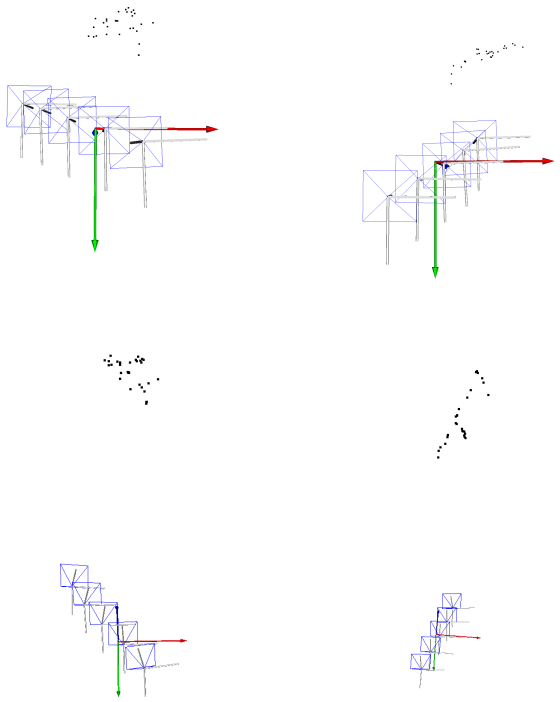


Figure 6.10: An illustration of the ambiguity problem from an attempt to reconstruct Gunnerus. The number of points is 50 in both examples, with the only difference being the spread of the points with a minimum distance for the right is 5px and 1px for the left. One should notice the difference in how the camera views are located in relation to the point cloud. The left images are considered the correct ones.

calculations without any additional information. A possible method for solving the problem is by adding more information to the process of determining the correct camera poses, for instance, that the object, in most cases, should be convex. This could, on the other hand, make the algorithm less general as one could meet concave objects by sea as well. Also, there exist different versions of RANSAC where, for instance, DEGENSAC [54] tries to solve the problem of multiple solutions for the fundamental matrix.

6.6 Relative yaw rate of objects from rotation matrices

When using the SFM method to create a point cloud, one obtains a rotation matrix and translation vector for each camera view. The rotation matrices contain important information about how the object's heading of interest changes in relation to milliAmpere 1. This is because SFM interpret the movement of both elements as

camera movement. One should recall that the ground truth used in the following sections is the change in yaw rate extracted from GNSS measurements of the position of the target relative to the rotational measurements from milliAmpere 1.

The rotation matrices of the camera views are interesting when estimating the target's relative yaw rate. One can obtain the change in yaw between images by looking at the change between the camera poses. This will directly describe the rotation between the object and milliAmpere 1, and by scaling the result with the inverse frame rate, one will obtain the rotation in yaw per second.

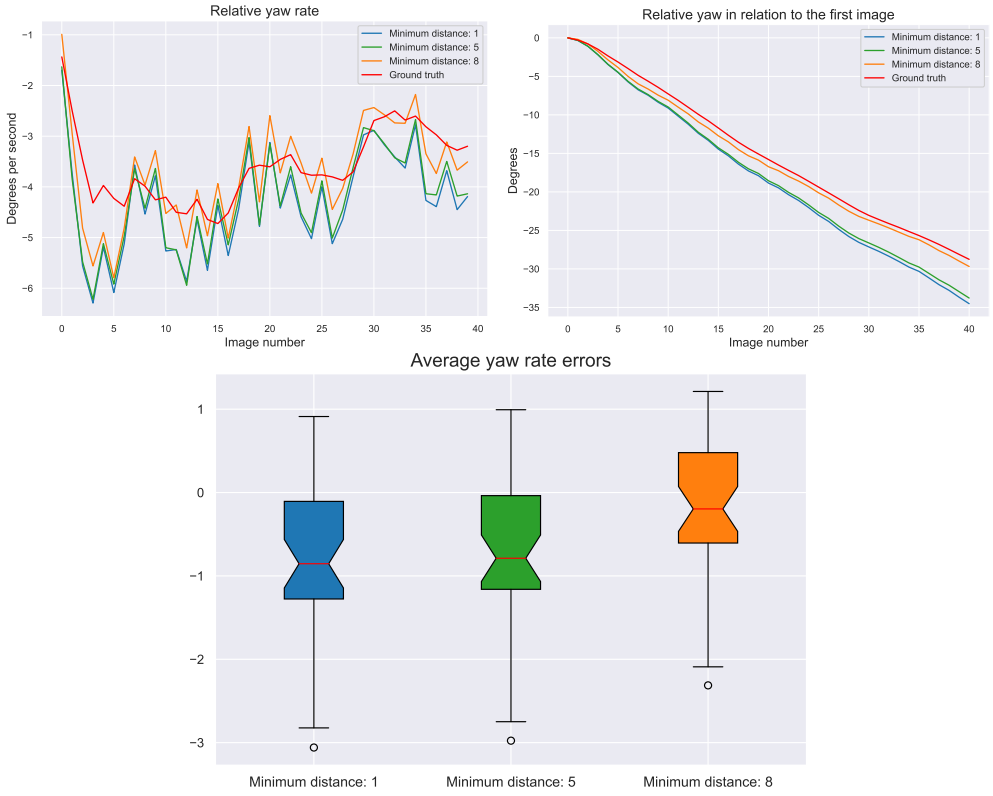


Figure 6.11: An illustration of the estimate of relative yaw rate and yaw for Gunnerus when approaching milliAmpere from a distance in Scenario 6 over 40 images with a high allowed number of points but varying minimum distance. 40 images are used to show that the method works for longer sequences. The left plot shows the estimated yaw rate between all images in the sequence used for SFM and the plot to the right shows the accumulated change in yaw, both for a different spread of the tracked points. The distribution of the error is shown in the third figure. The median of the error is marked with a red line, the colored box shows the 25% and 75% percentile of the distribution, and the black line represents the range of the error. The estimate with the most spread points seemingly performs better.

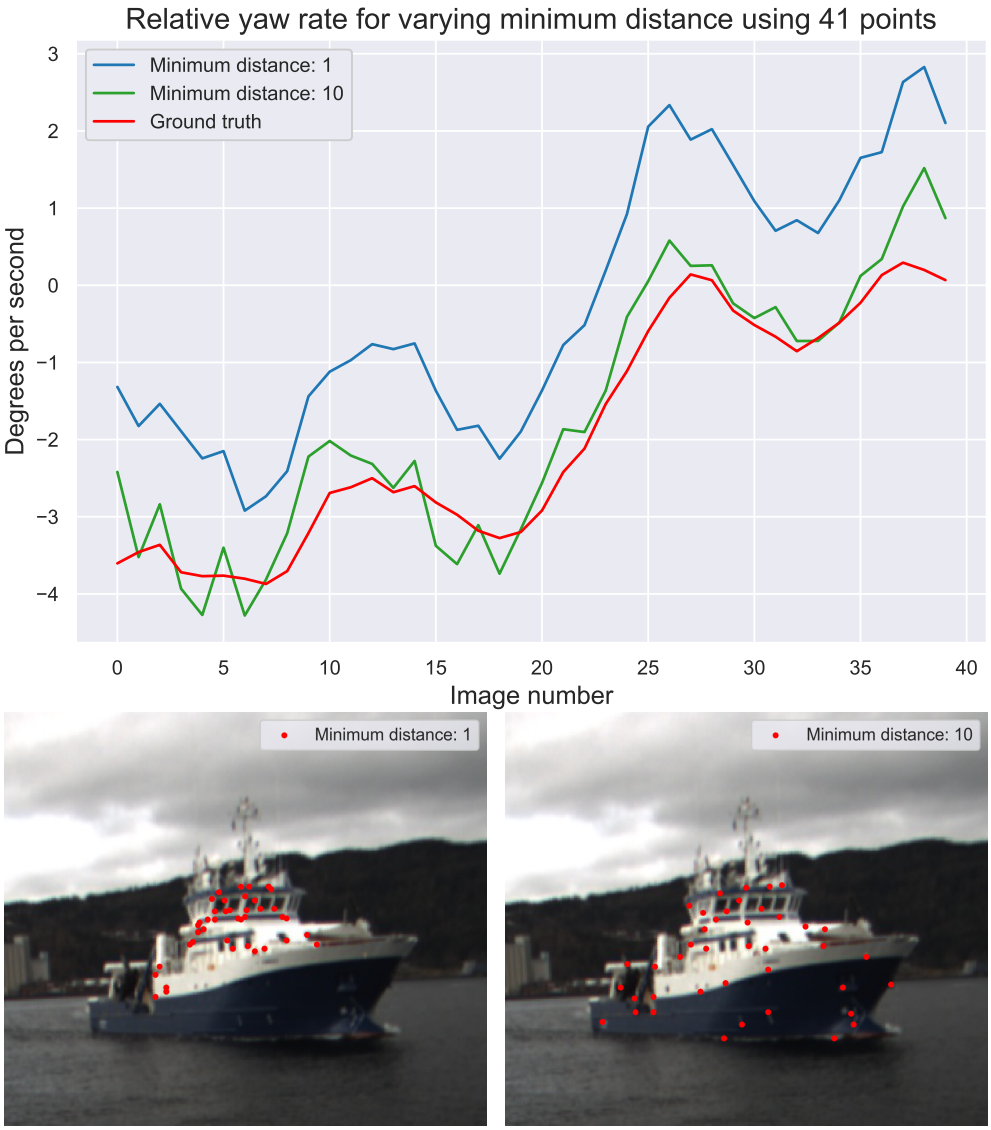


Figure 6.12: The plot shows the result of relative yaw rate estimation for the same amount of initial points but with different spreads on the object. The initial points with the most spread give better results. The respective initial points are shown in the two images above.

The 3D reconstruction of the target will only be made up to scale. This means that the point cloud translation can not directly describe the target's movement without any further information given by other sensors. However, the rotation, or change in heading, will be possible to decide as it is only described by rotation matrices.

The scenarios used for testing in the following sections are mainly based on Scenario 6 in Environment 1 shown in Figure 3.6. In Figure 6.11, a result of the

relative yaw rate from a sequence of SFM is compared to ground truth. The figures show the result for each image used in SFM. In the figures, the estimates with different minimum distances between the initial points for tracking are shown. The points are picked with a different constraint on minimum distance. This means that a trade-off between the best features available and the spread of the points on the object is evaluated. The results show that the relative yaw rate and followingly the relative cumulative yaw follow ground truth well with a small average error, especially for the most spread points. Generally, the points that are spread out give a better estimate of the relative rotation. This intends that points that in a higher degree covers the object will give a better understanding of how the object moves.

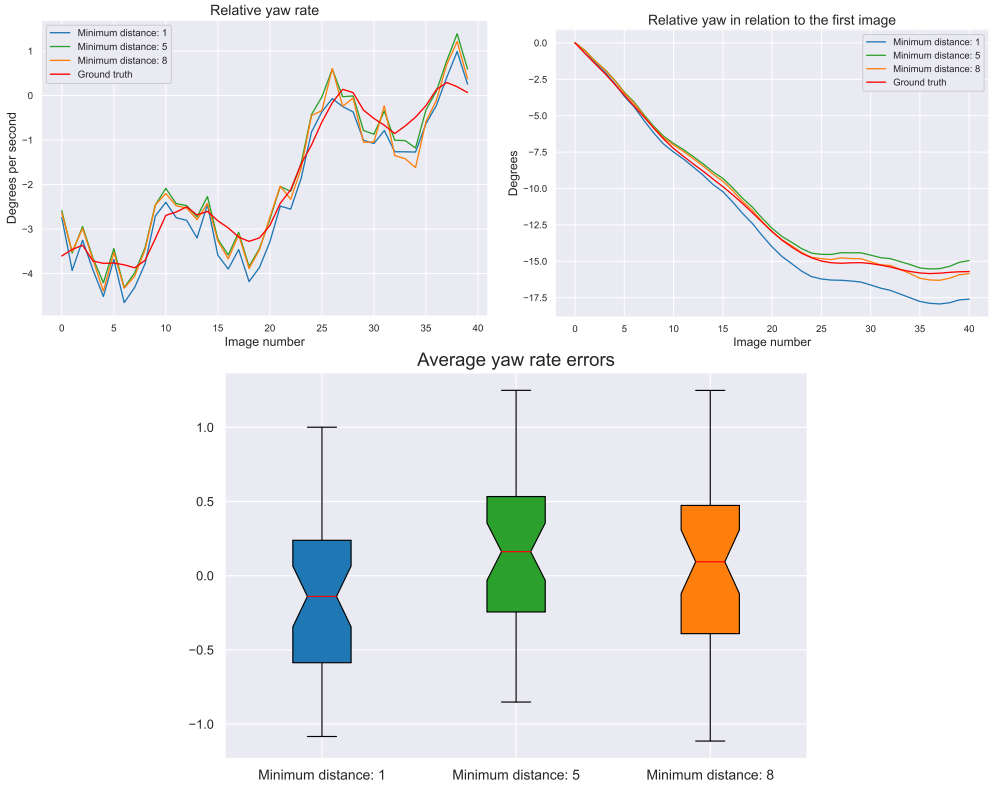


Figure 6.13: An illustration of the estimate of relative yaw rate and yaw when Gunnerus is close to milliAmpere 1 and performing a turn maneuver in Scenario 6 over 40 images with a high allowed number of points but varying minimum distance. The left plot shows the estimated yaw rate between all images in the sequence used for SFM and the plot to the right shows the accumulated change in yaw, both for a different spread of the tracked points. The distribution of the error is shown in the third figure. The median of the error is marked with a red line, the colored box shows the 25% and 75% percentile of the distribution, and the black line represents the range of the error. The estimate with the most spread points seemingly performs better.

The observation of better results for more spread points may rise from the computation of the camera poses in SFM. Recall that when computing the camera poses, the 7-point algorithm is used with RANSAC to find fundamental matrices which describe the geometry between two camera views before bundle adjustment. The minimum distance between the points will limit how many points that will be available in a mask. Also, if the maximum number of points is kept high, the points that are chosen are often also included in the same mask with a lower minimum distance because the best points available will be chosen in both cases. The chances of accepting camera view estimates based on neighboring points will therefore be greater with a smaller minimum distance as the close points are not filtered out. This may give a worse estimate of the transformations between the camera views which are directly connected to the yaw rate. So, even with a smaller number of points available, it still manages to get a better result. One could therefore also argue that the same amount of points in a more spread pattern will perform better than the equal amount of points placed in a small neighborhood on the object. This theory is tested and confirmed in Figure 6.12 where the result is displayed together with the points that are used as initial points. However, if the minimum distance is too large, one will not always have enough points to work with which must also be regarded.

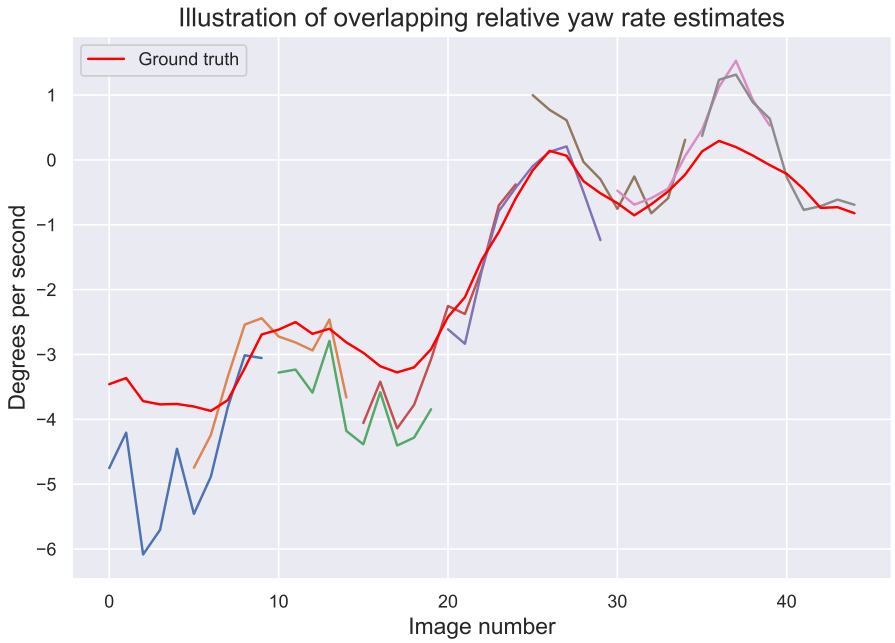


Figure 6.14: An illustration of how eight different smaller overlapping reconstruction results. The different sequences are marked with different colors and show that the results correlate. To get a better result, sequences of 20 images are used, where only the yaw rates of the last ten images are used. The sequence of images used is when Gunnerus is close to milliAmpere 1 and performing a turn maneuver.

The same observations were done in Figure 6.11 are made in the results for another sequence as seen in Figure 6.13. Here Gunnerus performs a turn with milliAmpere 1 following the boat with the camera. In this sequence, Gunnerus is closer to milliAmpere 1 than in the previous example. For the result with a higher minimum distance, great accuracy is observed with the yaw angle almost ending at the same value as ground truth. This indicates that it is easier to estimate the relative yaw rate the closer the object is. Another observation is how the estimate over- and under-estimates the relative yaw rate between images, which can be caused by noise and the bundle adjustment. Still, as a total, it follows ground truth well and ends up at a similar value.

To use the estimates for a longer period of time or in a real-time scenario one would have to create new reconstructions with SFM when new images arrive to always have the newest information about the yaw rate of the target. Figure 6.14 illustrates how multiple sequences together follow the ground truth well. The scenario used is the same as in Figure 6.13 and the result does not differ much from when looking at only one reconstruction result.

6.7 Tuning of reconstruction and yaw rate estimation

This section will present and assess the effects of tuning the key initial conditions which affect the SFM-reconstruction. To test how the initial points affected the SFM, a series of tests were done with different initial points by varying the parameters of the function for finding initial points. In addition, the number of images used for the reconstruction was varied. To evaluate the effect, the reprojection error was calculated for all the tests as well as the error in the estimated relative yaw rate. The relative yaw angles between the target and milliAmpere 1 were used for testing. If the yaw rate of the target were to be used alone, one would have to subtract the measured yaw rate of milliAmpere 1 from the estimate. In addition, the runtime of a complete reconstruction process, as well as the percentage of unsuccessful reconstructions, will be presented at the end of each subsection. The reconstruction is deemed unsuccessful if it yields an ambiguous and degenerate solution. The degenerate reconstructed poses normally result in pose estimates that significantly deviate from the previous timesteps. This would therefore lead to a high yaw acceleration and give misleading error estimates. Therefore, a reconstruction was deemed successful if the maximum yaw rate difference between the two frames was less than or equal to 15 degrees.

The parameters that will be tested are, firstly, the maximum number of points used for the tracking, thereafter the minimal distance between points, and lastly, the number of images used for the reconstruction will be tested. The results from varying each of these parameters will be presented in its own subsection. The aim of the testing was to find the optimal settings for the reconstruction in terms of

accuracy, robustness as well as computational cost. For all tests, 35 different sequences of images were used. For reasons later explained in Figure 6.28, all the image sequences were gathered from scenario 6. The 35 sequences have different start images. The quality level used to find the initial point was set to 0.01 for all tests. In the end, this section will conclude with parameters that will be used for the rest of the testing.

6.7.1 Maximum number of points

To test how the maximum number of points used for the reconstruction affected the reconstruction, a series of tests were conducted. During all tests, the values of minimal distance were set to 3px and the number of images used for the reconstruction was set to 31 which gives 30 yaw rate measurements.

Figure 6.15: Illustration of projected points on an image of Gunnerus in Environment 1, for the varying maximum number of points used in the reconstruction. From top-left to bottom-right, the number of points used are 10, 50, 100, and 200, respectively. It is observed that the reprojection gets slightly worse when more points are used for reconstruction.

As seen in Figure 6.15, the points used when the maximum number of points is low are located in high contrast areas in the upper parts of the vessel. This suggests that these points could be considered the best. As a larger amount of points are allowed to be used, the points are spread more evenly throughout the ship. Keep in mind that some points that are subject to poor tracking are removed.

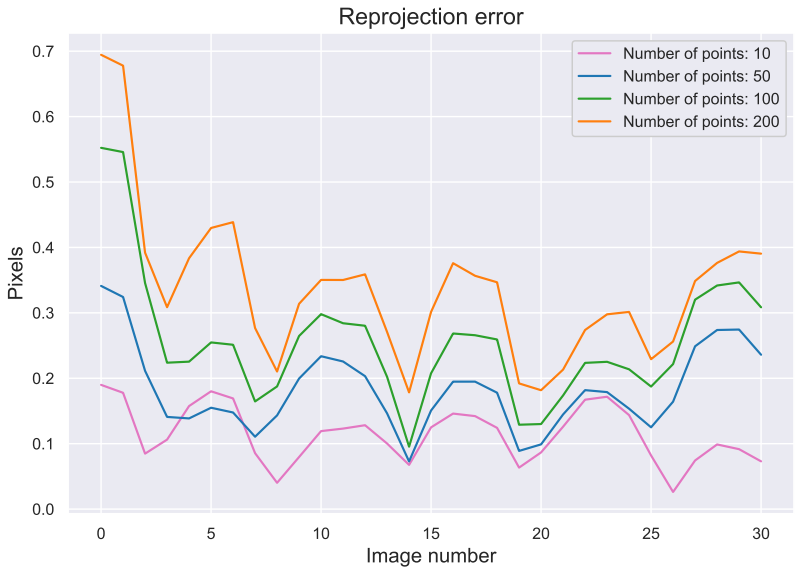


Figure 6.16: Graph depicting the variation of RMSE reprojection error as a function of the image sequence number for a reconstructed sequence using SFM. It showcases an increasing trend of error with an increase in the number of points used for reconstruction. The oscillatory nature of errors, with a period aligning with the oscillatory movement of the camera due to oceanic waves, can also be observed.

Figure 6.16 shows an example of the Root-Mean-Square Error (RMSE) reprojection error as a result of a reconstructed sequence using SFM. One can observe that the error increases with the number of points. One should also notice that the errors oscillate over a period of roughly five images. This matches the period of the oscillatory movement of the camera caused by the waves in the ocean. The images which show the lowest reprojection error correspond to the images taken when milliAmpere 1 is one of the two extremes in terms of pitch and roll angle. This is likely because when milliAmpere 1 is in this position, the camera is relatively steady between frames causing less motion blur, yielding a better reconstruction than when the camera is in the middle of the oscillatory movement.

Figure 6.17 shows a box plot of the RMSE of the reprojection for the different number of points. As with the single example in Figure 6.16, one can observe that the RMSE generally increases with an increased number of points. This is a consequence of the function used to choose the initial points. The function is always

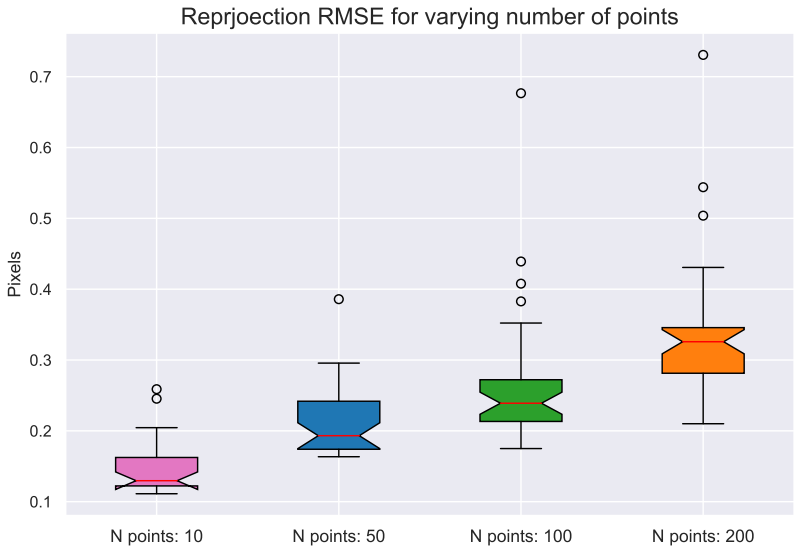


Figure 6.17: Box plot showing the distribution of RMSE of the reprojection for a different maximum number of points used in the reconstruction. It demonstrates that with an increase in the number of points, the RMSE also generally increases, leading to a poorer reprojection accuracy. The plot is based on the error distribution from 35 different image sequences. The median of the error is marked with a red line, the colored box shows the 25% and 75% percentile of the distribution, and the black line represents the range of the error. The means for 10, 50, 100, and 200 points are 0.14, 0.21, 0.26, and 0.33 pixels, respectively.

choosing the best points which fit the criteria set by the user. This also indicates that the additional points found when lowering the initial points' criteria will likely be worse than the points that have already been chosen. As the number of points increases, the likelihood increases to include worse features. These points can be hard to track accurately due to weak texture, motion blur, or other effects. Poorly tracked points will then be a source of error in SFM and will give a higher reprojection error. This is illustrated in Figure 6.15 where the points located on the hull of Gunnerus or in white areas generally have a worse reprojection error than the ones in high contrast areas, especially in the images where the number of points increases.

In addition, when there are only a few points, there are fewer constraints on the optimization problem. This can cause a situation where the poses and the point clouds are wrong, for instance, in the case of ambiguous solutions. This will give a visually wrong solution, but the reprojection error is kept low as the solution still is correct theoretically. Nevertheless, fewer constraints that follow from points that are tracked well could also make it possible to fit the camera poses better to the points and also for the correct solutions. This will give a lower reprojection error, also for visually correct solutions.

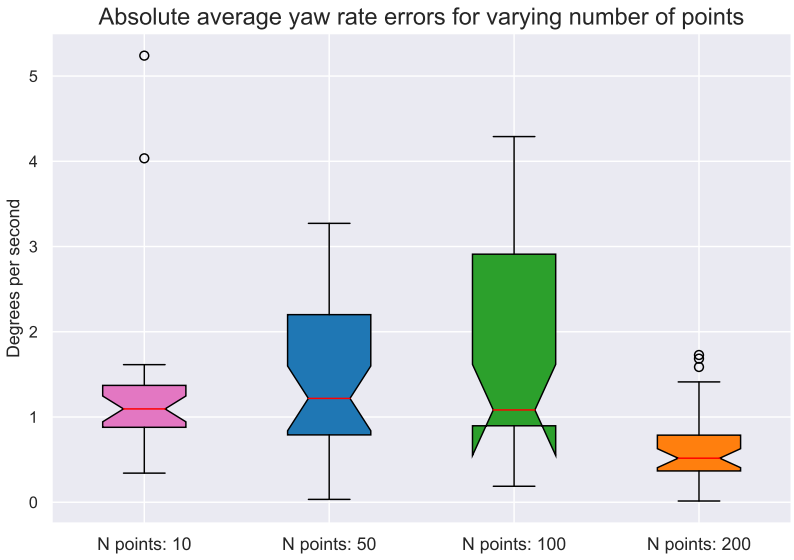


Figure 6.18: Box plot of the average absolute error for the relative yaw rate for the different maximum numbers of points used in the reconstruction. This plot provides insight into the impact of the number of points on the consistency and accuracy of relative yaw rate estimation. The plot is based on the error distribution from 35 different image sequences. The median of the error is marked with a red line, the colored box shows the 25% and 75% percentile of the distribution, and the black line represents the range of the error The means for 10, 50, 100, and 200 points are 1.33, 1.43, 1.78, and 0.64 degrees per second, respectively.

Upon examination of Figure 6.18, it is notable that the mean yaw rate errors for 10, 50, and 100 points appear approximately equal. One can also observe that the variance of the data increases significantly with the number of points for the three cases. This can be explained by judging by the illustration in Figure 6.15. One can observe that even though there are more points for the images corresponding to a maximum number of points of 50 and 100, the majority of the points are located on the same part of the ship. As the points are in the same neighborhood, they give little information about the 3D structure of the whole ship. The 3D structure of the whole ship is what is needed to give accurate estimations of the pose. Also, when adding more points from the same location on the ship, the extra points are considered worse initial points. This effectively adds more noise without adding more information about the actual 3D structure. This could lead to worse reconstruction. This can describe why the reconstruction with 100 points is worse than with 50, as the coverage of the object is similar. Therefore, the reconstruction with only ten points can lead to a good, if not better, reconstruction than the ones using more points from the same location, as the extra points do not give much new information.

Another aspect that influences the results in Figure 6.18 is the number of times the reconstruction and estimates are considered unsuccessful. As seen in Table 6.1

the failure rate resulting from reconstruction with a maximum of 10 points is significantly higher than for the other tests. As only the successful estimates are included in the plot, the error may be artificially low and one would rather prefer the reconstructions with more points as one will get more useful estimates.

Contrary to the reprojection error, the relative yaw rate error decreases when the number of points used increases. As seen in Table 6.1, the yaw rate error is significantly reduced when the maximum number of points is increased to 200. This is likely because, as seen in Figure 6.15, the points have good coverage of the whole ship. The reconstruction of the ship, therefore, has more accurate information about its 3D structure and, followingly, it also has more information about the rotation of the ship in relation to the camera. Altogether, this indicates that the reprojection error alone is a good indicator of the quality of the reconstruction but that it is not necessarily a good measure of the quality of the estimates. Also, the more coverage of the target, the better the estimates will get.

Table 6.1 shows that increasing the number of points makes the reconstruction process more robust with a lower failure rate for a higher number of points. This makes sense as the RANSAC and the 7-point algorithm, which initializes the poses, are more likely to choose more spread points and have a smaller chance of unsuccessful runs. A trade-off comes when looking at the runtime of the algorithm, where the runtime increases with the number of points used. The increase is, however, not significantly large between different amounts of points compared to the initial process when creating the reconstruction. This is based on the fact that the increase is lower than the runtime of reconstruction with 10 points.

Table 6.1: Average runtime of a single SFM result based on the number of points used for reconstruction. Failure rate describes the number of wrong solutions due to ambiguity.

Maximum points	Runtime (s)	Failure Rate (%)
10	1.42	25.71
50	1.60	2.86
100	1.74	0
200	2.00	0

6.7.2 Minimum distance between initial points

During the tests with minimum distance, the maximum number of points was set to 200. As seen in Figure 6.19, the minimum distance between the initial points affects the number of points due to the restriction of the area on the ship. The result is the higher the minimum distance, the fewer points.Figure 6.20 shows how the mean reprojection error steadily increases with the increasing minimum dis-

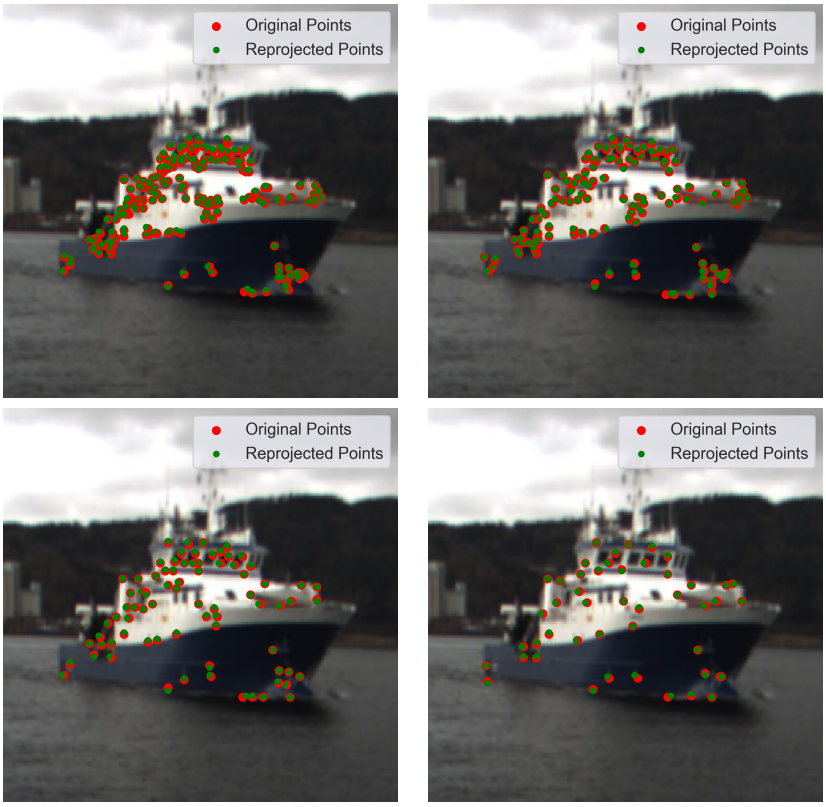


Figure 6.19: Distribution of points on the ship for different minimum distances between the initial points. Each plot represents the spatial distribution for minimum distances of 1px, 3px, 5px, and 7px, respectively.

tance between points. Contrary to the tests with a different number of points, this means that the reconstruction using the least amount of points gave the results with the highest average reprojection error. This can be explained by the fact that the restriction of a larger spread of the points makes it necessary to choose worse features from the same selection of points in all four scenarios. Worse features will result in a higher reprojection error.

One can see from Figure 6.19 that the vessel has points spread across the entire ship for all the minimum distances. This explains how the mean of the yaw rate error in Figure 6.21 is similar in all cases with the percentiles for the higher minimum distances being closer to zero. One can, however, see that the error when using a minimum distance of 1px is slightly higher than for the rest. In this scenario, the majority of points will be located close to each other. The bundle adjustment step in the algorithm will prioritize to optimize the reprojection error for the points located in the highly dense areas. This will result in the optimal poses being fitted to points that do not completely describe the whole movement of the object which results in a worse pose estimate.

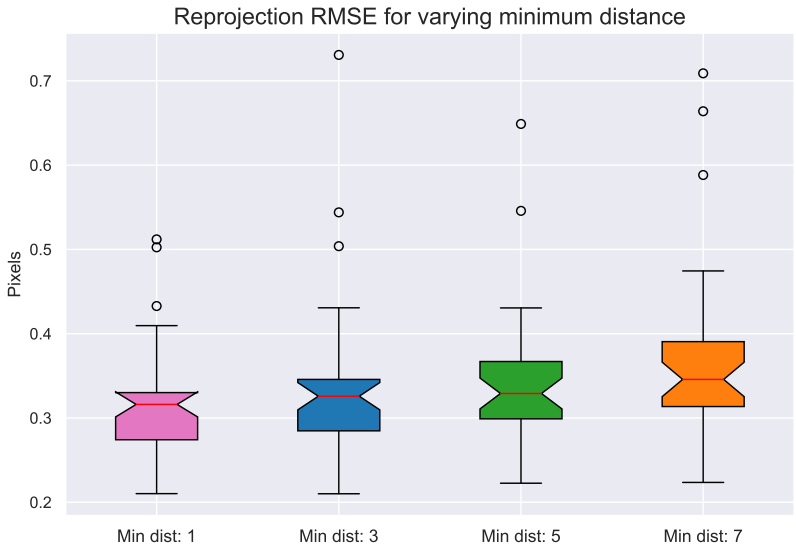


Figure 6.20: Boxplot showing the average RMSE of the reprojection for different minimum distances between the initial points. The plot illustrates an increasing trend in error with the increasing minimum distance. The plot is based on the error distribution from 35 different image sequences. The median of the error is marked with a red line, the colored box shows the 25% and 75% percentile of the distribution, and the black line represents the range of the error The means for min distances 1, 3, 5, and 7 are 0.31, 0.33, 0.34, and 0.36 pixels, respectively.

When comparing the results acquired from using minimum distances 3px, 5px, and 7px, one can see that the mean value of the errors is almost identical in all cases. However, the variance is higher for minimum distances of 5px and 7px, and the total distribution is closer to zero than for a minimum distance of 3px. This means the estimates will be closer to the ground truth the more spread the points are, as it gives more coverage of the object. This corresponds to the results obtained for a different number of points and strengthens the hypothesis that more coverage of the object gives better estimations. There is, however, a trade-off between performance and the number of points available when adjusting the minimum distance. If the minimum distance is too high, there will not be many points available, and one can more often get meet scenarios where the reconstruction will be worse due to bad track of points or noise.

Table 6.2 shows the runtime and failure rate for different minimum distances. The failure rate is generally kept low as the points are spread out on the object in most scenarios. When looking at the runtimes, the runtime decreases with the increasing minimum distance. This comes from the number of points available which matches the results from a varying number of points.

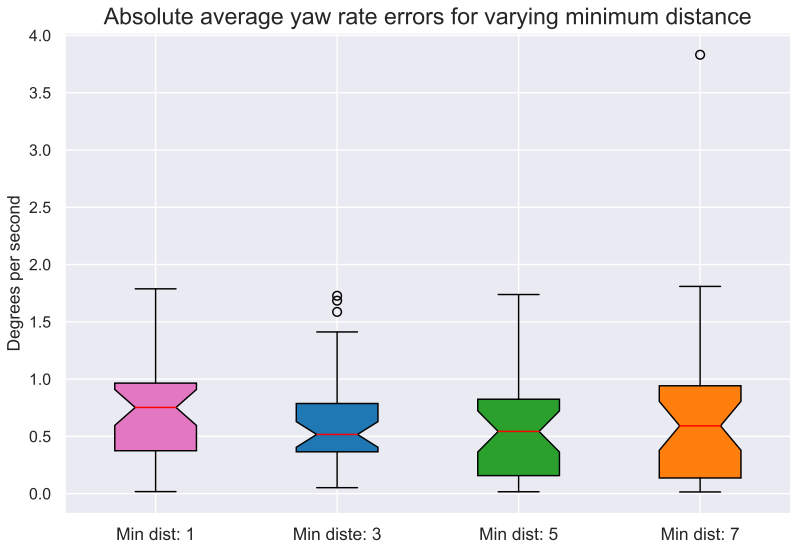


Figure 6.21: Boxplot showing the average absolute error for the relative yaw rate for different minimum distances between the initial points. The plot shows similar median values for the errors, with the total distribution being closer to zero for minimum distances 5px and 7px. The plot is based on the error distribution from 35 different image sequences. The median of the error is marked with a red line, the colored box shows the 25% and 75% percentile of the distribution, and the black line represents the range of the error The means for min distances 1, 3, 5, and 7 are 0.74, 0.64, 0.63, and 0.70 degrees per second, respectively.

Table 6.2: Average runtime of a single SFM result based on the minimum distance between the initial points used for tracking and SFM. Failure rate describes the number of wrong solutions due to ambiguity.

Minimum distance (px)	Runtime (s)	Failure Rate (%)
1	1.83	0
3	1.79	0
5	1.78	2.86
7	1.61	0

6.7.3 Number of images

During the tests of the number of images, the maximum number of points was set to 200 and the minimum distance was set to 3px to as to get high-quality and robust results. The minimum distance is set to 3px to ensure that there are enough points available in all masks. The tests done in this section are highly related to testing how different baselines between the images affect the reconstruction.

As seen in Figure 6.22, the average reprojection error increases when the number of images used for reconstruction increases. This can be explained by multiple

factors which can affect the result. In terms of tracking, it has been shown that the track not always is perfect and that there can exist a tracking error in the points used for reconstruction. This error can propagate between images and will possibly get larger the more images involved. This can make it harder to fit the camera poses to the point cloud in the bundle adjustment step, and one would get a larger reprojection error. In Section 2.8, it is explained that a smaller baseline will give more uncertainty in the 3D projection of a point in an image. When fewer images are used, there is a larger area where the point could be located, especially in depth. This could be beneficial in the bundle adjustment step when trying to fit the camera poses to the same point cloud which will make the total reprojection error smaller. Another source of the increasing reprojection error, which can also be related to the tracking error, is the fact that the more images, the more camera poses will be attempted to fit the same point cloud. This will increase the chance of images with noise and other disturbances which in general can increase the chance of a higher reprojection error.

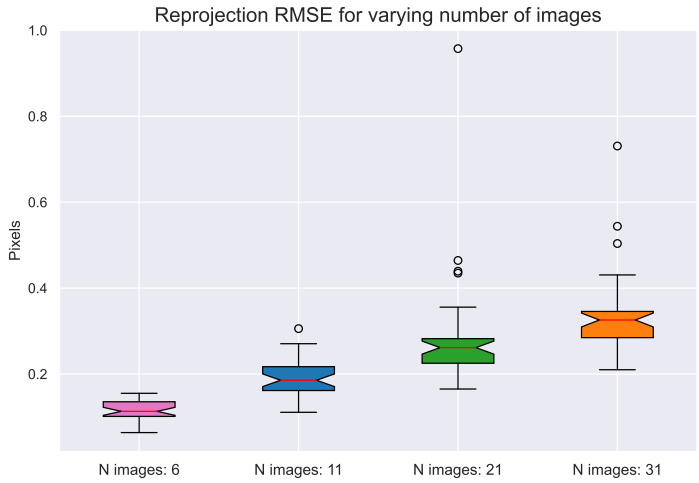


Figure 6.22: Boxplot showing the average RMSE of the reprojection for a varying number of images used for reconstruction. The plot illustrates how the error increases when more camera poses are attempted to fit the same points. The plot is based on the error distribution from 35 different image sequences. The median of the error is marked with a red line, the colored box shows the 25% and 75% percentile of the distribution, and the black line represents the range of the error. The means for 6, 11, 21, and 31 images are 0.11, 0.19, 0.28, and 0.33 pixels, respectively.

This can be explained by that when fewer images are used to initialize the point cloud, tho ships are likely to be in a similar relative position when the first and the last image are taken. This effectively makes the baseline used for the reconstruction shorter. This means that there is little 3D- information about the location of the points as illustrated in Figure 2.9. However, as the pose estimation is refined iteratively during the bundle adjustment step, the baseline between two

subsequent images is even smaller which makes it possible for the algorithm to find a pose that yields low reconstruction error, even if the 3D position of the points is highly inaccurate. The bundle adjustment process does not alter the total shape of the point cloud, meaning that all the views will assume that the warped point cloud is correct. An example of this happening is shown in Figure 6.23. One can also see in Table 6.3 that the reconstruction fails completely 5.71 percent of the times which corresponds to 2 of the 35 sequences.

Figure 6.24 illustrates the error in relative yaw rate estimation. In the figure, the relative yaw rate error seems to decrease with the increased number of images. With a higher number of points, there will be a larger baseline available, and the reconstruction is then likely to be more accurate. This will give more accurate estimates of the camera motion which reflects the movement of both the camera and the object. The results in the figure show that the higher amount of images, the better the results get. However, one can observe that the difference between the error with 21 and 31 images is smaller than the difference between 6 and 11 images. Also, the distribution of the error with 31 images has a small variance which indicates that the error will flatten out with a higher amount of images. A demand when using a large number of images is that the target is visible in all images and that the features of the object of interest are somewhat visible in all images to ensure a good track. Nevertheless, the small error indicates that if enough images are available, the estimates will follow ground truth well.

Table 6.3: Average runtime of a single SFM result based on the number of images used for SFM. Failure rate describes the number of wrong solutions due to ambiguity.

Number of images	Runtime (s)	Failure Rate (%)
6	0.40	5.71
11	0.59	0
21	1.11	0
31	1.70	0

In Table 6.3, one observes that the algorithm's runtime increases with the number of images used in the reconstruction. Compared to the runtime with different numbers of points, the difference in runtime increases more with the increase in number of images. A trade-off between runtime and accuracy is therefore present. Also, the failure rate is somewhat significant when the number of images is too low. This shows that the accuracy of the estimates will be worse when fewer images are available. This could, for instance, be just after the detection of the object.

(a) Front view **(b)** Top view

Figure 6.23: Effects of the short baseline on point cloud and consequent pose estimation, due to limited images used for reconstruction. In the front view (left), the ship appears well-structured, falsely indicating a successful reconstruction. However, the top view (right) unveils the true state of the 3D model, exhibiting a skewed orientation not discernible from the front. This demonstrates the intrinsic inaccuracies of the 3D model, manifesting the challenges posed by the short baseline and a limited number of images in capturing the genuine 3D structure of the ship.

6.7.4 Tuning conclusion

In conclusion, the results show that an essential part of getting a high-quality 3D reconstruction and, consequently, good pose estimation is to get as much coverage of the object of interest as possible. This can be done by spreading the points. This can affect the quality of the reprojection but will, on the other hand, give a more descriptive representation of the target's motion relative to the camera mounted on milliAmpere 1. The number of images used for reconstruction will also affect the reconstruction quality and the quality of the estimates. A larger amount of images will give a more significant baseline which will make the reconstruction describe the camera motion better. This will, on the other hand, give a larger reprojection error for the resulting SFM.

As a result of this analysis, the number of images will be kept high in the following sections of this chapter. This minimum distance will also be as high as possible. However, the masks of the targets are not always as big, and a trade-off must be done to find consistent parameters which can be used for multiple scenarios.

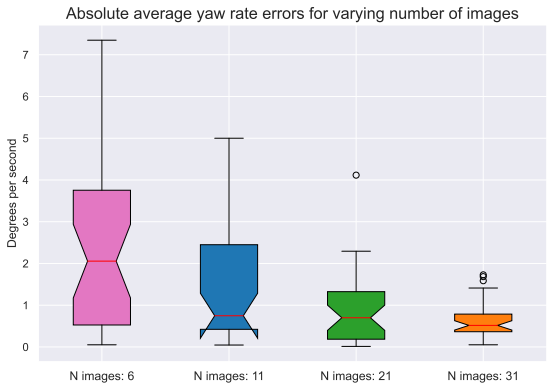


Figure 6.24: Boxplot showing the average absolute error for the relative yaw rate for a varying number of images used for reconstruction. The plot illustrates how the error decreases when more camera poses are used to describe the relative motion between the Gunnerus and milliAmpere 1. The plot is based on the error distribution from 35 different image sequences. The median of the error is marked with a red line, the colored box shows the 25% and 75% percentile of the distribution, and the black line represents the range of the error. The means for 6, 11, 21, and 31 images are 2.52, 1.58, 0.91, and 0.64 degrees per second, respectively.

6.8 Estimation of relative yaw rate in a simulated real-time sequence

In this section, as described in Section 4.10, the relative yaw rate will be plotted where the data point at each image is made from an individual reconstruction result with 20 images back in time, including the newest image, where the last yaw rate is extracted and plotted. 20 images are used instead of 30 because the target is small early in one scenario, and one would like to include the same features in the images. This will be kept consistent throughout the simulations while considering that more images could give a better result if available. The min distance used for initial points in this section is 3px to ensure that enough points are available, even for the images further away, to be able to use the same parameters for all scenarios. Also, from the previous section, these numbers will perform well while keeping the runtime low.

Figure 6.25 illustrates such a scenario when Gunnerus performs a turn maneuver when located close to milliAmpere 1. The figure shows the estimated relative yaw rate for three different amounts of points compared to ground truth. One can observe that the estimates based on the greatest number of points perform better during the whole sequence of images. The estimates are also, in general, following the ground truth well, with an average error close to zero. This is also visible in the plot. In this sequence, the boat is relatively close to milliAmpere 1. This makes the boat's features visible, making it easier to get a good estimate.

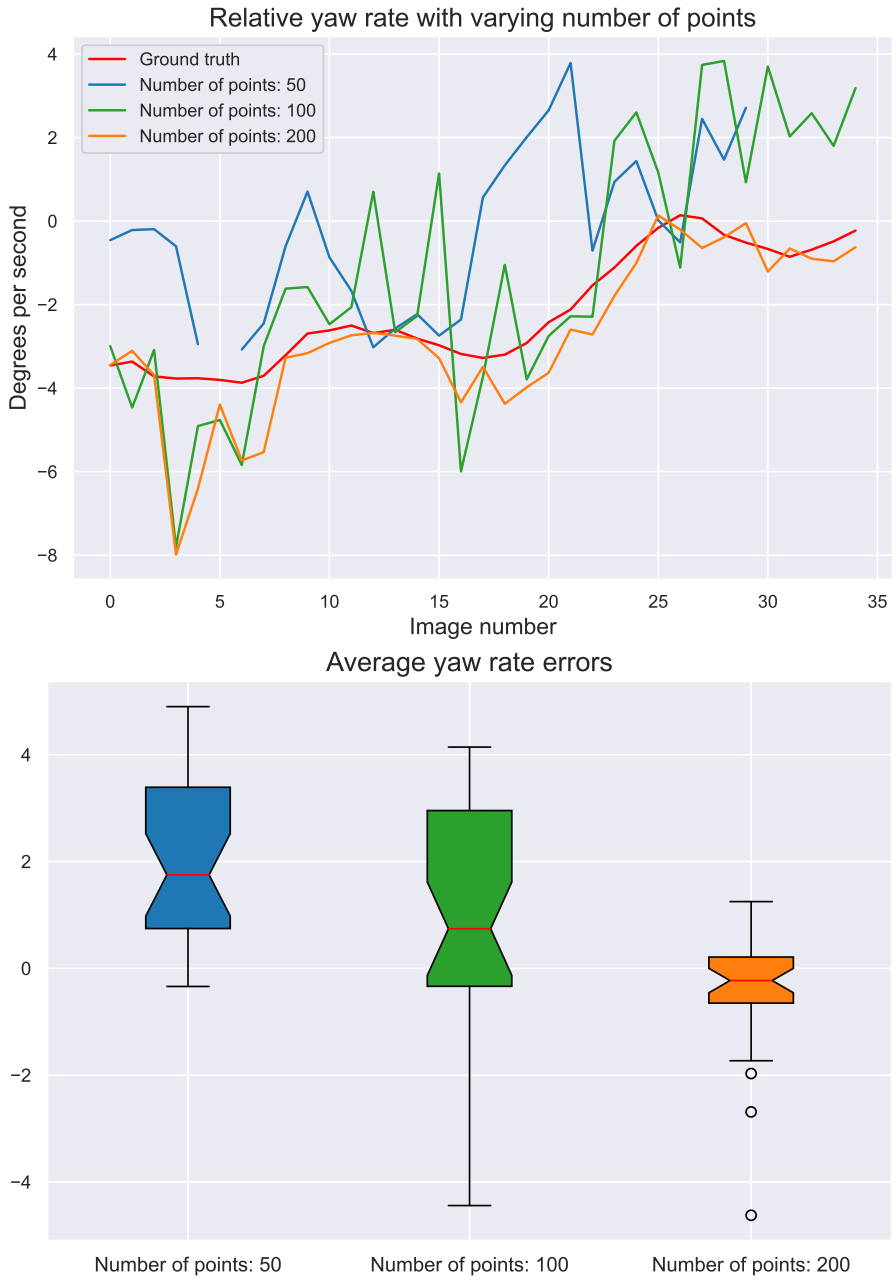


Figure 6.25: An illustration of the relative yaw rate estimate where a new reconstruction is done with 20 previous images for each new image in the sequence. In the sequence, Gunnerus is performing a turn maneuver close to milliAmpere 1. The results show how more coverage, or more points, of the boat, gives a better result. The number of points is the maximum number of points, and the real number that is used is based on the available points in the mask. The average error is small. The empty values in the graphs are a result of NaN values. The value is set to Nan when ambiguity is present. Ambiguities give a large deviation from ground truth which will make the plot less visible.

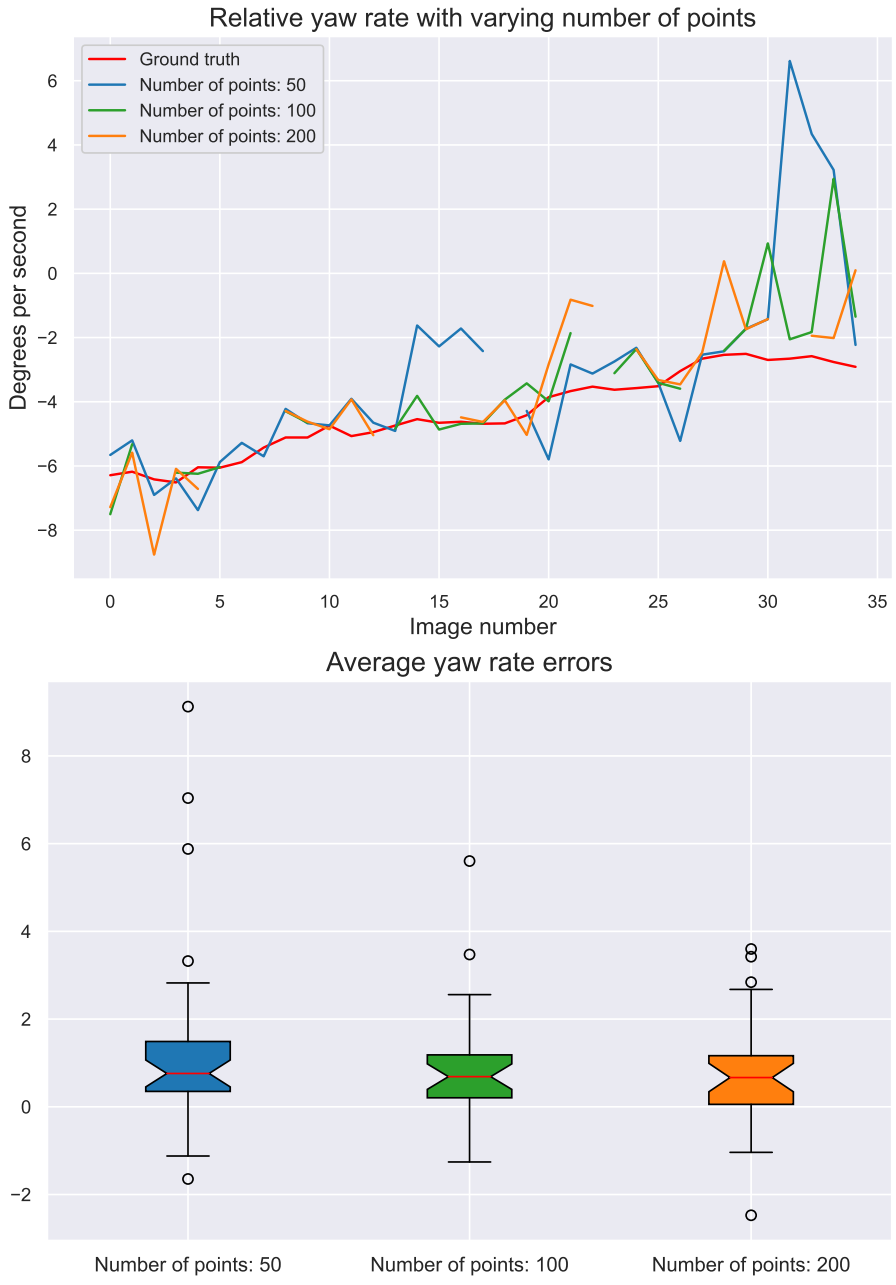


Figure 6.26: An illustration of the relative yaw rate estimate where a new reconstruction is done with 20 previous images for each new image in the sequence. In the sequence, Gunnerus drives away from milliAmpere 1 and the camera is filming from behind. The results show how more coverage, or more points, of the boat, gives a better result. The number of points is the maximum number of points, and the real number that is used is based on the available points in the mask. However, the results are getting worse the further away Gunnerus gets for all number of points. The empty values in the graphs are a result of NaN values. The value is set to Nan when ambiguity is present. Ambiguities give a large deviation from ground truth which will make the plot less visible.

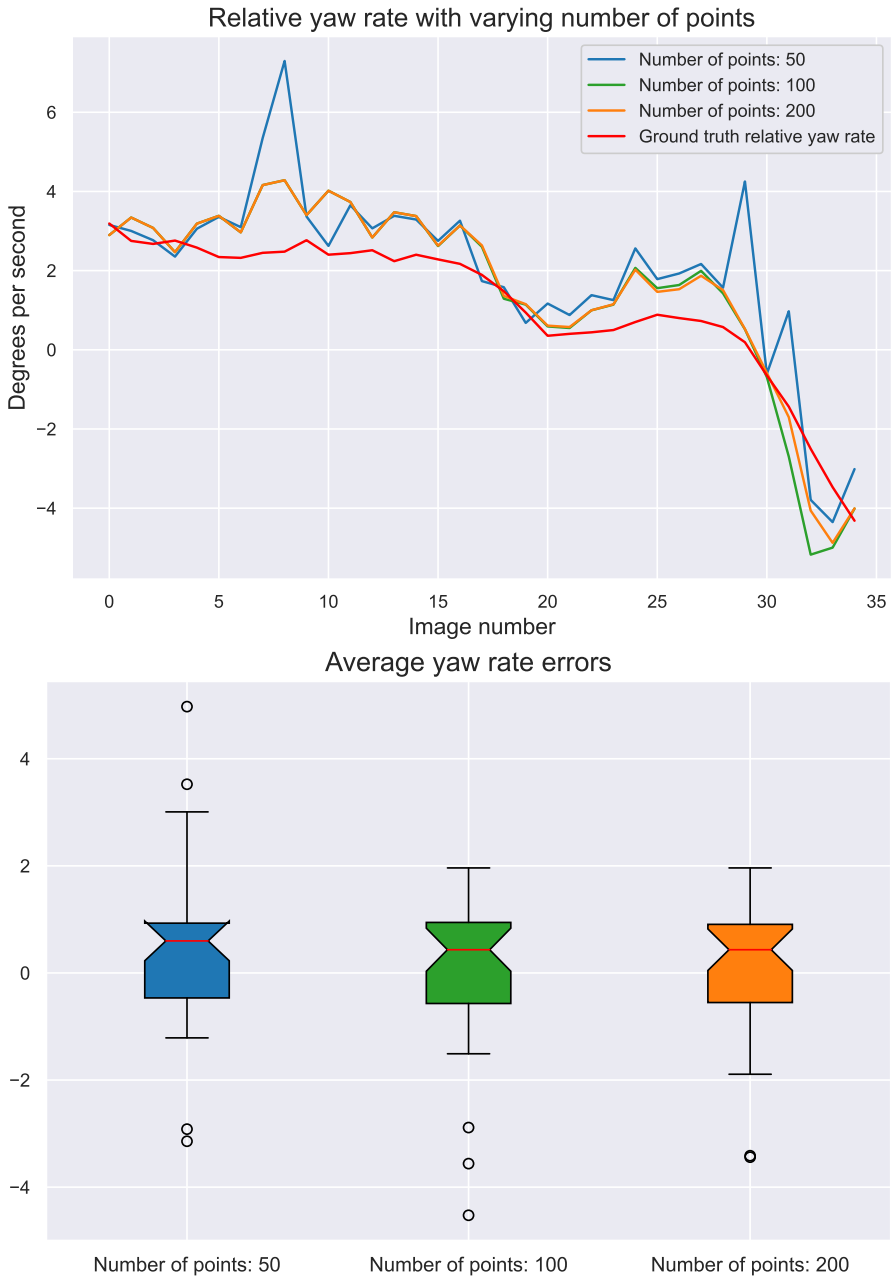


Figure 6.27: An illustration of the relative yaw rate estimate where a new reconstruction is done with 20 previous images for each new image in the sequence. In the sequence, Gunnerus approaches milliAmpere 1 from far away while filming the front of the boat. The more points on the boat, the better the estimate gets. Early in the sequence, the estimates with 100 and 200 points look similar. This indicates that there are not more than 100 points available on the target at the time due to its small size in the image. The spikes come in the areas where waves affect camera stability.

In some periods, one notices deviations for all the estimates, for instance, between image numbers 2 and 7. In this part of the sequence, milliAmpere 1 and the stability of the camera are highly affected by waves that give rotations in roll, pitch, and yaw. The sensor for ground truth is tuned in a way that does not prioritize all degrees of freedom, and as one also will observe later on, the estimates will have some deviations in these cases. However, the estimates look good in the areas with fewer movements in roll and pitch.

In Figure 6.26, one can observe the results of another sequence where Gunnerus is moving away from milliAmpere 1. The plot shows how the estimated relative yaw rate follows the ground truth well in the beginning and starts to struggle when the target gets less clear during the sequence due to the distance. This is as one would expect as it is harder to extract information from an object far away, and the number of points available will be reduced. However, the average error er kept small, especially for the plot where the most points are used. It should be mentioned that the difference in error between the graphs in this sequence is smaller than in Figure 6.25. This is because the available points on the object and the real number of points used are more similar in this sequence.

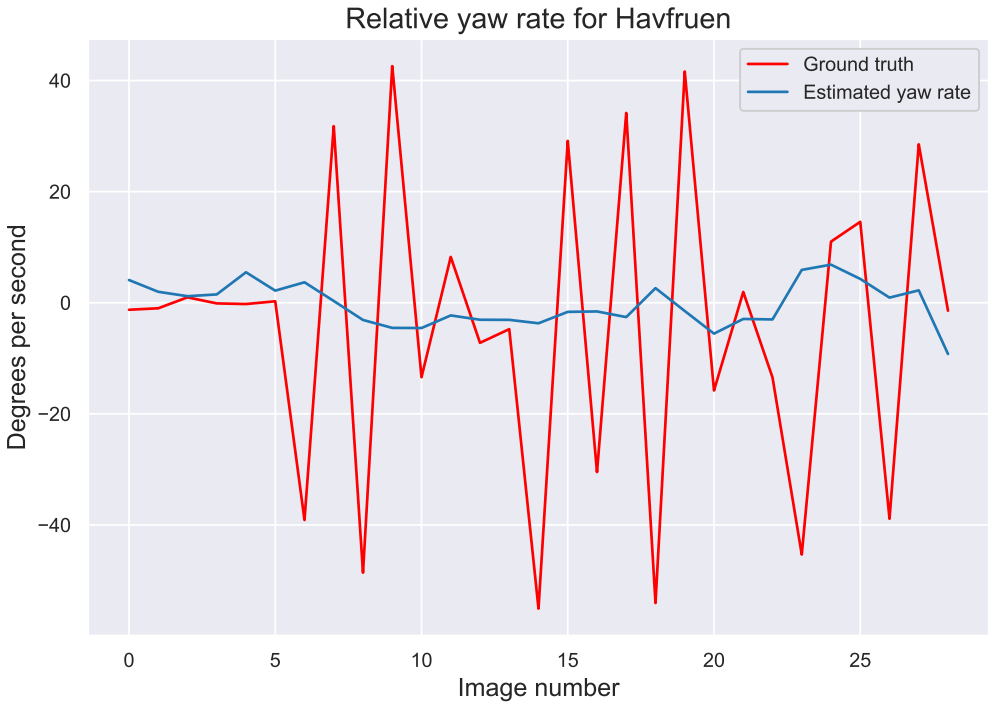


Figure 6.28: Example of a simulated real-time scenario where the relative yaw rate of Havfruen is estimated. The figure shows an oscillating ground truth due to variance in the GNSS data which makes it hard to evaluate the result. The trend of the estimate is, however, good.

Figure 6.29: An illustration of how sunny conditions affect the features on Havfruen.

The same method is tested for another sequence in Figure 6.27. In this sequence, Gunnerus is approaching milliAmpere 1 from a distance where the camera is pointed towards the front of Gunnerus. The camera is highly affected by waves in this sequence which causes sudden changes in the camera views. This is also the reason for the many spikes in the estimate. Nevertheless, the average error is kept below one degree during the sequence which is considered good. The small deviation throughout deviation from the ground truth is likely due to a slip in the heading which is not included in the ground truth as the data set lacks information about this.

Until now, only sequences with Gunnerus from Scenario 6 is tested. In Figure 6.28 an estimate of the relative yaw angle for Havfruen in Scenario 13 is shown. The figure shows a noisy ground truth with large oscillations. This comes from the uncertainty in the GNSS device and the way the ground truth for the heading is calculated from positional measurements. Nevertheless, the estimate in the figure seemingly follows the trend of the ground truth well without the possibility of further confirmations of the result. Also, Scenario 13 is filmed on a sunny day. As Havfruen is white, the reflection of the light on the boat makes the features less visible. This will reduce the number of points available and make tracking harder. This is illustrated in Figure 6.29.

During the reconstruction process of the Havfruen ship, another challenge that arose was the presence of outliers. As seen in Figure 6.30, the white color of the ship and the white capping effect on the waves created difficulties during YOLO detection. The white parts of the waves could be mistakenly identified as parts of the ship, leading to the initialization of points on the waves instead of the actual ship. As a result, these points would diverge from the ship as they were tracked, resulting in an inaccurate representation of the ship's structure.

To address this issue, several measures can be implemented. One approach is to limit the tracking process to points that consistently remain within the masks produced by YOLO detection. This requires precise and accurate masks generated by the YOLO algorithm so as not to remove good points. Another potential solution is to rely on the removal of outliers during the RANSAC algorithm. However, this may not always occur since the points may not deviate significantly from frame

(a) Image showing Havfruen.

(b) Havfruen with YOLO mask.

(c) Resulting points on the boat.

Figure 6.30: Example illustrating the problem of outliers during reconstruction. The images show (a) the original boat image, (b) the boat image with the YOLO mask overlay, and (c) the resulting points on the boat. The white parts of the waves, are mistaken as parts of the ship, resulting in being initialized on the white waves as seen in the lower right parts of the boat in (c).

to frame and will, therefore, not necessarily be interpreted as actual outliers.

6.9 The accuracy of the isolated estimate of the yaw rate of the target

In real-life applications of this estimation method, one would typically have access to the pose of milliAmpere 1 through measurements from other sensors. Therefore, it is interesting to evaluate the isolated estimates of the target's yaw rate after removing the movement of milliAmpere 1.

The estimated relative yaw rate is a composite of two motions - the rotation of milliAmpere 1 and the rotation of Gunnerus. The movement of milliAmpere 1 primarily manifests as a shift in the position of the entire point cloud across the image frames. In contrast, Gunnerus's rotation impacts how the points relate to each other within the point cloud. In terms of pixel changes within an image,

the motion from milliAmpere 1 will likely have a more significant impact, raising questions about the method's ability to accurately detect the motion caused by the rotation of Gunnerus. If Gunnerus's motion were not detected accurately, the estimated yaw rate would align more closely with the measured yaw rate for milliAmpere 1.

Figure 6.31 presents the estimated yaw rate of Gunnerus in comparison with the ground truth derived from a single SFM result. This comparison validates that the estimation method does capture the movement of Gunnerus, and the estimated yaw rate aligns well with the constant turn in Scenario 6, suggesting the method's potential accuracy in detecting motions even amidst significant shifts caused by milliAmpere 1.

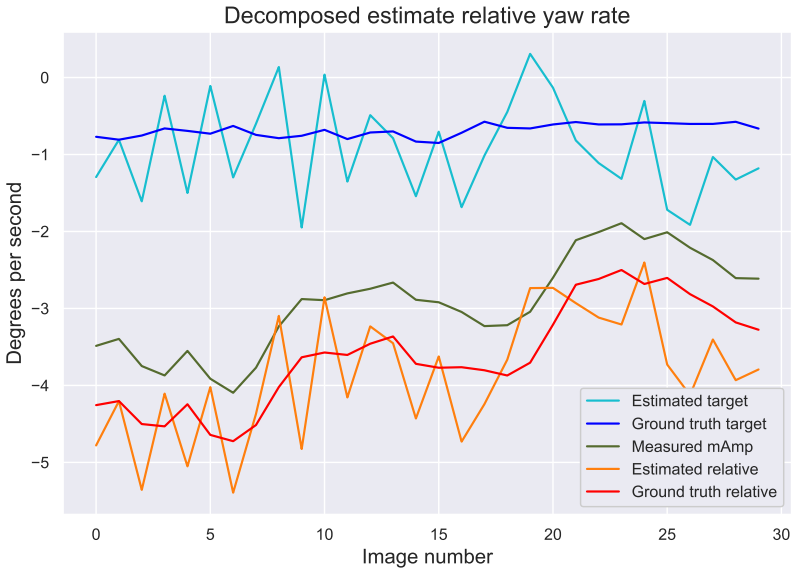


Figure 6.31: The estimated relative yaw rate, ground truth, the movement of milliAmpere 1, and the estimate of Gunnerus in when performing a turn. The estimated movement of the target follows the ground truth with some oscillations. The plot illustrates the isolated target estimate over a single SFM result.

The isolated estimates of the yaw rate of the target in the simulated real-time scenario, when Gunnerus is moving towards milliAmpere 1 from far away, is shown in Figure 6.32. Here one observes that the estimate follows the estimate of milliAmpere 1 to a higher degree than in the single SFM result in Figure 6.31. In this scenario, the target is further away. The small changes of Gunnerus are harder to analyze when the target is less visible, and the result is that the movement of milliAmpere 1 is more dominant in the estimate. However, the isolated estimate of the target is not zero which means that the method catches some of the movement of the target.

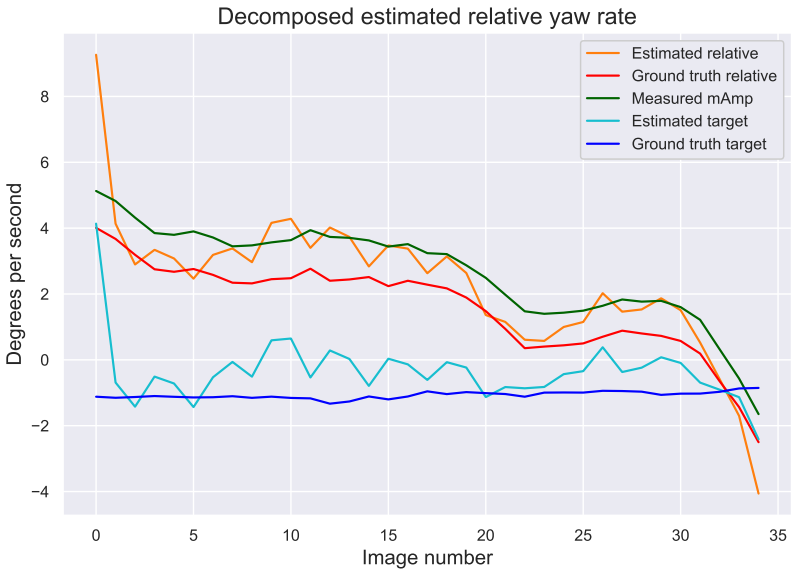


Figure 6.32: The estimated relative yaw rate, ground truth, the movement of milliAmpere 1, and the estimate of Gunnerus when moving towards milliAmpere 1 from far away. The estimated movement of the target follows the ground truth with some oscillations. The plot illustrates the isolated target estimate over a simulated real-time scenario.

6.10 Center estimation of the target

When measuring the distance between two points on an object, the distance may vary based on where the point is located on the object. Therefore, how much an object has moved between two camera angles is decided by where on the target one measures the movement. One way of estimating the average movement of the target could be by looking at the movement of the mean of the point cloud. Another method is to use a MVEE to create an ellipsoid that covers all the points in the point cloud to estimate the shape and find the volumetric center of the point cloud.

Figure 6.33 illustrates a MVEE that covers the reconstructed point cloud of Gunnerus. The reconstruction gives a good estimate of how Gunnerus would look if one could see the boat from all angles. The middle point is also seemingly in the middle of the boat, which is good for further estimation. One could assume that the mean would be closer to the areas with more points which is undesirable. This is illustrated in the same figure as an orange dot. One can see that the mean is influenced by the side of the boat seen by the camera. The ground truth is not included in the comparison. As one will see later, the GNSS is imperfect and will not follow the pitch and roll of milliAmpere 1 well. It will, therefore not give correct information about the center of the target when projected onto the image frame of the camera.

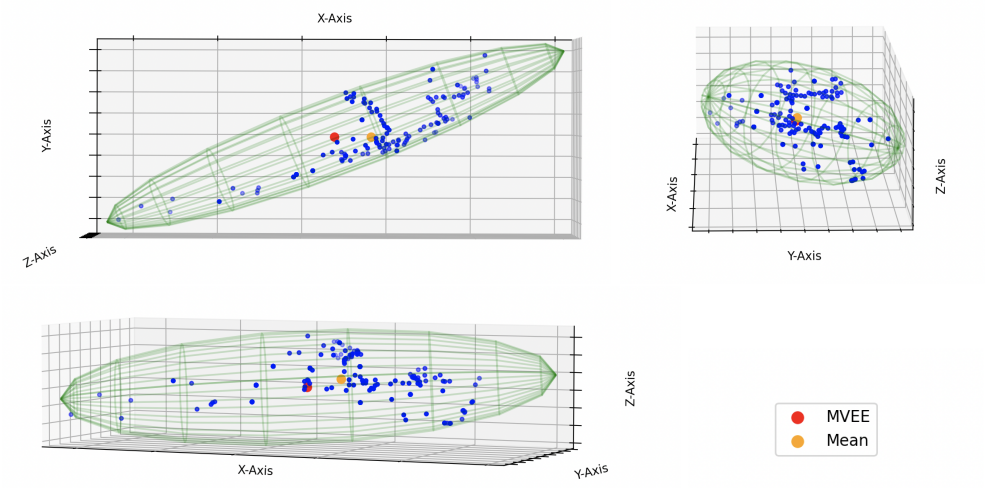


Figure 6.33: Illustration of an ellipsoid that surrounds all points related to the 3D reconstruction of Gunnerus seen from different angles. The red dot indicates the estimated center of the point cloud and the ellipsoid, and the orange dot indicates the mean of the point cloud. The views are from the top, front, and side, respectively.

Figure 6.33 confirms that MVEE may be a better approach to find the center of the target as it seemingly gives a better estimate for the volumetric center of the point cloud than the mean. This method is therefore used further in this thesis for position estimation.

6.11 Estimating the position of the target

As explained in Section 4.12 the position of a target in the NED frame can, in theory, be estimated using the reconstruction results from structure from motion in combination with the positional and rotational measurements for the main vessel. However, a prerequisite for this is that the positional and rotational measurements for the main vessel are highly accurate. As seen in Figure 6.34, one can observe large and oscillatory deviations in the z-value. This is a consequence of the pose estimation of milliAmpere 1 as explained in Section 4.14. It seems that the alpha-beta filter prioritizes the smoothing of the yaw measurement over the responsiveness of the roll- and pitch measurements.

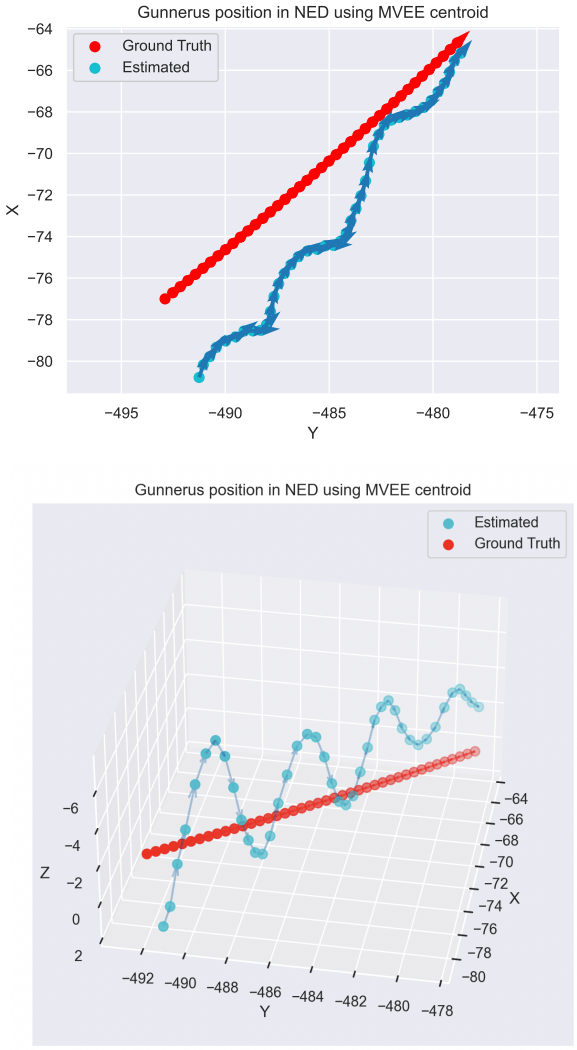


Figure 6.34: Positional estimates of Gunnerus in Environment 1 in 2D and 3D to illustrate the deviation in the z-values. The estimates are highly affected by waves that cause changes in roll and pitch which are filtered out from the measurements for milliAmpere 1. The plots are given in world NED coordinates, meaning that scale has been introduced. The trend is somewhat correct in the plots, however, it is hard to compare with the illustrated deviation.

To illustrate and confirm that the measurements used for ground truth are less correct when waves cause movements in roll and pitch, the GNSS measurements are projected onto the image plane. This is shown in Figure 6.35. One can observe how the GNSS struggles to follow the center of Gunnerus where the sensor is assumed to be located. The movements are mainly in a vertical direction. In addition, this is also evident in that the measured angles do not match with what one can observe in the images corresponding to the measurements. The measured

pitch angles relating to the three images displayed in Figure 6.35 are 1.01, 1.20, and 1.30 degrees, respectively. The measured roll angles relating to the three images are -2.50, -1.97, and -1,95. However, one can see from the horizon that these measurements are inaccurate. From visual inspection, these angles can be deemed highly inaccurate.

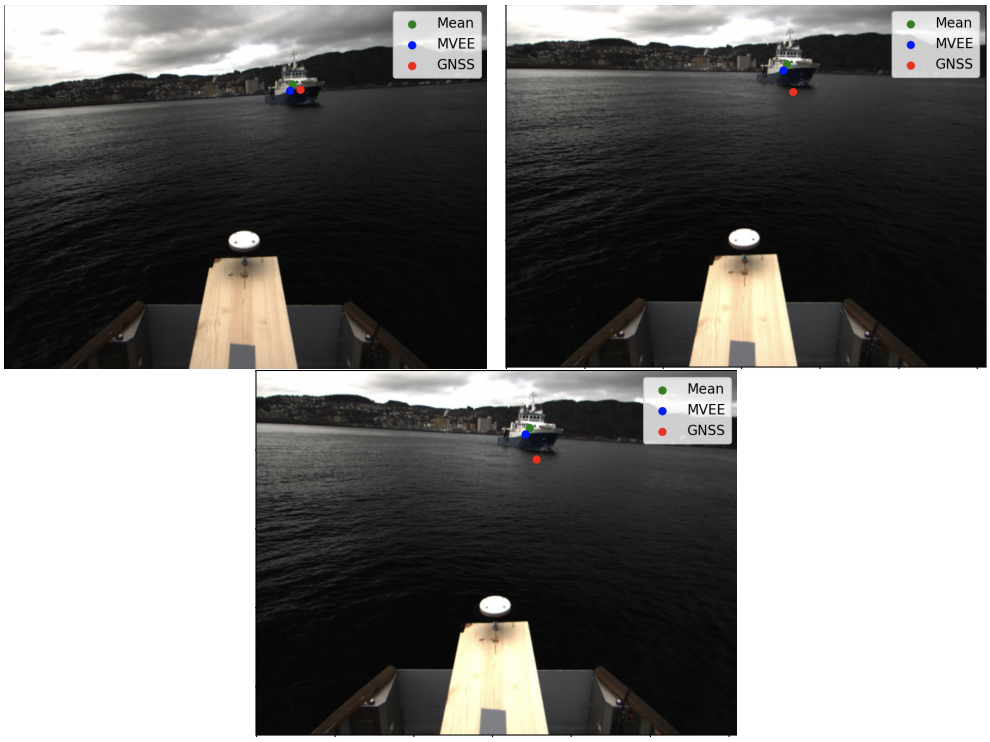


Figure 6.35: Image sequence showing how the inaccurate pitch and roll estimated values affect the projection of the GNSS measurements of Gunnerus in Environment 1. The inaccurate measurements make it hard to compare the estimated center of Gunnerus with the ground truth.

As these changes in pitch and roll are not represented by the measurements, the changes of the center of the target in the body frame are therefore interpreted as changes in heave when transformed to the NED-frame. This can be seen in the upper plot in Figure 6.34. The lower plot in Figure 6.34 shows how this leads to that changes in roll and pitch being wrongly interpreted as changes in yaw. This ultimately causes inaccuracies in the estimated position of the target in NED. Therefore, the results cannot be easily compared.

Given the importance of accurate heading information, it seems logical in most maritime applications to prioritize heading when pre-processing the rotational measurements of the vessel. This is because the vessel's motion predominantly occurs in the horizontal plane and therefore is directly related to the course of

the vessel. However, when using rotational measurements in combination with computer vision, it is advantageous to have accurate estimations of yaw and pitch as well.

Due to difficulties when comparing estimates to ground truth due to changes in roll and pitch, it is interesting to make the same comparison when the water is still in Environment 2. In Figure 6.36, positional estimates of Havfruen in scenario 13 are shown. Both the method where one point is tracked and the method with SFM and MVEE are used and illustrated in separate plots in the figure. It is observed that the first method includes an offset as a result of the tracked point not being in the same position as the GNSS sensor. Besides the offset, the tracks with the two methods are both good with some deviations. The deviations can be caused by inaccuracies in the estimate, the way ground truth is extracted from the available data, and the method for finding the scale from the ground truth. Nevertheless, the track is accurate, and by introducing the MVEE to find the centroid of the object, the offset in position is removed from the estimate.

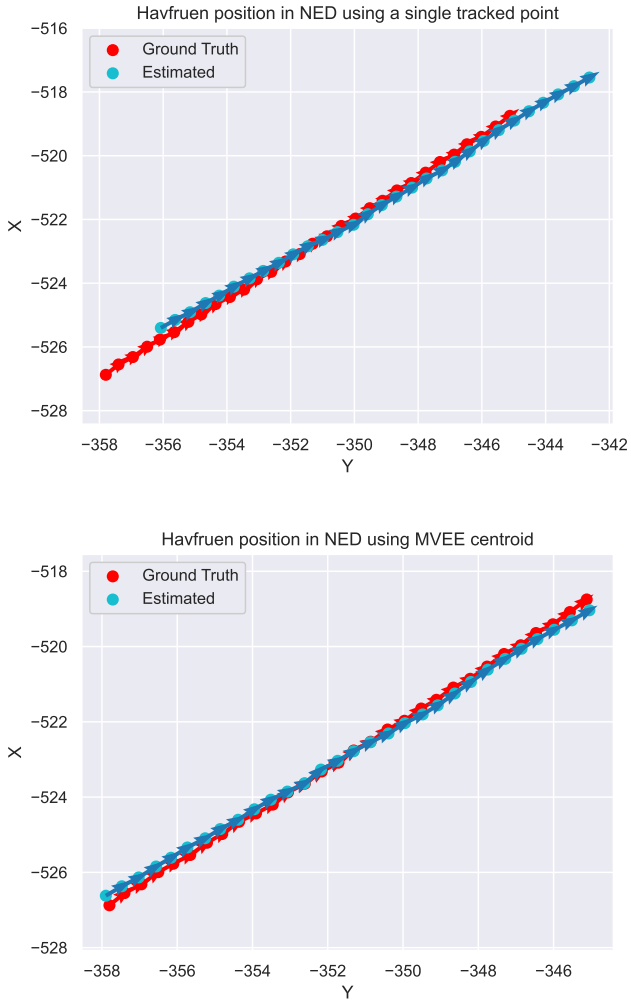


Figure 6.36: Positional estimates of Havfruen in Environment 2. The estimates are not affected as much by waves as in Environment 1. The plots are given in world NED coordinates, meaning that scale has been introduced. The upper plot shows the result from only tracking a single point on the target while the lower plot shows the result from SFM and MVEE. The first method has an offset which is removed with the second method.

Chapter 7

Discussion

7.1 Detection

In terms of detection, YOLOv8 seemingly outperforms the method based on optical flow in many areas. The masks from YOLOv8 are better and the detection is more robust in terms of detection of water and other parts of the background and is not affected by rapid changes in movement of the camera. The tuning of the optical flow will also often be case dependent which makes the algorithm hard to generalize. In addition, YOLOv8 outperforms the other method in terms of runtime where they can do detections in the same image in terms of milliseconds and a couple of seconds respectively. The runtime of YOLOv8 will however be dependent on the model that is used whereas larger models will use more time while giving more accurate detections. Nevertheless, YOLOv8 will still be able to compete with the other method in terms of runtime In addition to this, the method based on optical flow needs two images to do what YOLOv8 can do in one image.

Another difference between the two detection methods is how the method based on optical flow only detects moving objects. By ignoring objects which are still-standing, one will save some computational power while keeping track of targets that may pose a danger for milliAmpere 1. However, there is not a huge downside to tracking objects that are docked as this would be additional information. On the other hand, if one only would like to detect moving objects, one could combine the methods which would give the best of both worlds.

7.2 Tracking and SFM

In general, the combination of LK and SFM seemingly work well to estimate the relative rotation and translation of a target when the object is isolated from the rest of the image with a mask. It manages to reconstruct the object well and the camera poses are reasonable when compared with the images used, as long as the

points used are tracked well and located on the object of interest. If the point that deviates from this could be considered outliers and will give a worse result. Nevertheless, this shows that the algorithm manages to estimate both the movement and rotation of the target and milliAmpere 1 when triangulating between images in a sequence. It also shows that the method is able to describe the movement of the target as a movement of the camera view which is embedded in the relative movement between the camera and the object.

The quality of the 3D reconstruction depends on the track by LK. This can be affected by multiple factors. One of them is the quality of the camera. The camera used in this thesis has a relatively low resolution which does not capture all the details of the objects. This would also make the track better for objects further away. This also includes the lack of insight into the calibration process of the camera which could be a source of error. A better camera would also allow for better detection of features and tracking further away. Also, with a higher frame rate, the changes between the images would be smaller which would make the tracking more accurate.

In the case of occlusion and a rotation of the target that covers features that are visible in other images, it will not be possible to find correspondences. A more comprehensive method for guaranteeing that the correspondences match, or for instance, a method for only finding correspondences between two consecutive frames could be used to solve the problem. Nevertheless, this is a problem that is not regarded in this thesis and could be seen as a further improvement of the system.

The results illustrate how a baseline is needed when reconstructing a scene with SFM. There are, however, some problems that can occur when the baseline is small. This could, for instance, be at the beginning of a track, just after detection, when only a few images are available, or when the relative movement between the boat and the camera is small, regardless of how many images are used for reconstruction. This could, for instance, be if both are standing still. Such a scenario would make it hard to give a good reconstruction result. However, it can be discussed whether or not the method is needed for non-moving objects when the ferry is parked, and the user could be restricted to when the ferry is moving, which could be decided by analyzing the tracked points. This would reduce the number of bad estimates.

Ambiguity is a problem that can affect such that the estimates differ a lot from the results one would expect. Nevertheless, this is not directly handled in this thesis as it is challenging to change the implementation of the OpenCV reconstruct function, but it would be possible with a self-implemented function. The wrong estimates as a consequence of ambiguity could also be ignored. By fusing the estimates with other sensors in later stages in the estimation process, one would be

able to trust these estimates less, and they would therefore be less important.

7.3 Relative yaw rate estimation

A general observation in the relative yaw rate estimate is that the estimates follow the ground truth well with a small average error in the cases where the measurements used for ground truth are reliable. It is clear from the plots that the best estimates are found when the target is close to milliAmpere 1. The method also gives a somewhat reliable estimate when the target moves further away from the camera.

In terms of tuning, the accuracy of the results is generally based on how much coverage one has of the object. More coverage will give a better understanding of the target's movements, making it possible to give a better estimate of the relative change in yaw. Also, it is shown that a greater baseline, or more images used, will give a better result as it gives a better view of the target. This stands in contrast to the reprojection error of the 3D reconstruction. Nevertheless, as the goal is to get accurate pose estimates, one should prioritize the estimates over how well the reconstruction is reprojected.

In Section 6.6, it is illustrated how the relative yaw rate follows the images when only looking at single SFM results. When looking at the error in these estimates, the average error is close to zero and is more or less equally spread around an equilibrium. After estimating the camera views in SFM, the bundle adjustment may change the views due to noise which will make the estimates oscillatory. It is, however, observed that the estimate, in general, has a low error as the oscillations are around ground truth. This shows that the method with relative yaw rate estimation works well when using SFM and LK. It is also shown that the more spread of the points on the object, the more accurate the results get. This comes from the fact that more coverage of the target gives a better understanding of how the target moves. Subsequently, an offset or error could occur when there is insufficient coverage of the target. Also, as seen in the position estimation results, the roll and pitch affect the yaw of the target. This effect is filtered out from the GNSS measurements, making the data used for ground truth somewhat incorrect. A scenario without waves has not been tested for relative yaw rate estimation, and one can assume that some of the error also can be caused by this effect.

Further, the same method is used for simulating a real-time scenario to analyze the method's performance when extracting the last yaw result from a single reconstruction. The results are promising as the result gives an estimate which follows ground truth well. One downside with this method is that the over- and under-compensation are not included as the separate estimates do not have any information about each other. This is the advantage of using only one SFM result,

as the error in the different frames will be correlated. By excluding this information, one could possibly obtain worse results that are not desired. To solve this, one would have to change the implementation by, for instance, including the new image in a current SFM. This has, however, not been tested in this thesis. In addition to this, it can be observed that the estimates get worse as the target moves further away from milliAmpere 1. This is expected as the details of the target and its movements get less visible from a greater distance.

The method used for the simulated real-time implementation is not optimal. The information about the camera positions is calculated multiple times, and the new estimate does not use any of the information from the previous estimates. In a more optimal implementation, one could initialize a 3D construction after tracking a target for enough frames to have images with a long enough baseline effectively. After this, the new poses and yaw rates could be estimated by only using the position of the points in the newest frame. This would drastically decrease the computational time as only one pose estimation and bundle adjustment must be performed. This would also have benefits as two subsequent pose estimates would be directly correlated. Nonetheless, the current implementation provides promising indications that the method accurately estimates relative yaw rates in accordance with GNSS measurements.

In the simulated real-time scenario results, 20 images were used. The resulting runtime was approximately one second per yaw rate estimate. This includes tracking, SFM, and yaw rate extraction. The frame rate of the camera is five images per second, meaning that one is not able to give an estimate before a new image arrives. The runtime will be reduced by reducing the number of images. However, this will be at the cost of accuracy. Nevertheless, the implementations and methods used in this thesis are not optimal, so the runtime can be further reduced by doing calculations more efficiently.

7.4 Reflections on ground truth for yaw angle

In this thesis, the ground truth for the yaw angle is derived from positional measurements used to calculate the ship's course. Given Gunnerus' large size and speed, this method provided a reasonable estimation. However, an inherent delay is introduced due to the time required for a change in yaw to manifest in the ship's course.

This delay and the phenomenon known as slip, the disparity between course and actual yaw, are observable in Figure 6.31 and Figure 6.27. The results could be affected, with disparities introduced between estimated and actual yaw angles. Still, considering these effects, it's worth noting that the findings may actually be more accurate than they first appear.

The ground truth for heading, inferred from the direction of positional movements, might be affected by slip, especially when the ship changes its yaw frequently or rapidly. This is an inherent limitation of the ground truth, suggesting that a more precise method to obtain heading measurements might improve the accuracy. Future research should, therefore, consider more direct and accurate real-time yaw angle measurements, potentially from rotational sensors, to eliminate time lag and enhance the precision of optical flow techniques.

7.5 Position estimation

The results show that, by only using information from a camera, one can obtain precise position estimates compared to ground truth. There are, nevertheless, some deviations that can be explained by factors such as incorrect scale values or noise. This is the case when the GNSS measurements are reliable, as in the channel in Environment 2 without waves. In Environment 1, however, it is harder to compare the position estimates with ground truth as the GNSS measurements are smoothed in roll and pitch. Nevertheless, the good results for Havfruen show that it is possible to give a good estimate of the position of the target.

This will, however, only be possible to give correct estimates in world coordinates if the scale is available. The method used for finding scale in this thesis would not be suitable in a real scenario as one does not have GNSS measurements of the target. Methods that could be used instead could be to use LIDAR measurements to find the distance to the points in the point cloud. Another possible method could be to use stereo vision to estimate the position of the point cloud.

The method where only one point is tracked in the image gives an offset on the positional estimates while the trend of the estimates is a lot similar to the other method. This is expected as the point is located in a different position on the target than the GNSS sensor. Nevertheless, by finding the center of the target with MVEE, one can reduce this positional offset which would give a more precise estimate. One could also get information about the size and shape of the target by using a MVEE which could be interesting when analyzing the danger of the object.

The position estimates could also be useful if one would like to get course estimates of the targets. This could be interesting if one needs information about the direction of movement of the target, and could be found by looking at the direction of the change in position. This can be interesting information as it, for instance, would tell if an object is on a collision course or not.

Chapter 8

Future Work

In terms of detection, the method based on optical flow struggles to estimate the camera movements when the camera moves in different directions. These movements are already measured by other sensors on milliAmpere 1, and one could combine these with the detection algorithm for better motion estimations for the background and hence, better performance. YOLOv8 gives good detection results for marine objects. If this detection method is going to be used, an automatic pipeline should be constructed to map the right masks to the right objects, for instance, based on the tracked points. There will then be no need for manual masking which would be convenient for use in a real system. Also, if it is desirable to detect only moving objects, a combination of YOLOv8 and optical flow could be used.

The current implementation for pose estimation LK and SFM does not take problems such as occlusion and covered points as a result of large rotations into account. Solutions for this should be included for robustness before the system is usable in real-world applications. Also, to be able to get the best possible results, a dynamic tuning of minimum distance and number of points could be included based on the size of the object in the image.

In the surroundings of an autonomous ferry, there will be more than one vessel present. In order to use the estimation method, one should scale up the system, both detection, tracking, and estimation, to handle more than one object at a time. The system should, however, be scalable, and one could perform the same method for each object where an object is connected to a specific track.

The implementation of SFM used in this thesis is an OpenCV module, which limits flexibility and makes adding potential improvements difficult. The reconstruct function only returns a SFM result. A future project could focus on developing a SFM function from scratch, creating a more optimal function for reconstruction suited to this particular application. This could, for instance, be done similarly to the incremental reconstruction proposed in [55]. The proposed im-

plementation will significantly reduce runtime and possibly improve the accuracy of the estimates.

The estimates obtained from the methods presented in this thesis could be used in a larger system by fusing the estimates with other sensors. To remove some of the noise and oscillations of the measurements and disregard deviating estimates, one could filter the estimates using a Kalman filter. To successfully utilize a Kalman filter, a mathematical model of the boats' motion must first be established. This model serves as a predictor of the future state of the boat, taking into account factors such as current velocity, heading, and yaw rate. Subsequently, the estimates from our optical flow and structure from motion methods are fed into the Kalman filter as measurements, alongside potential data from other sensors like LIDAR or radar. The Kalman filter then combines these measurements with the predictions from our mathematical model to generate refined estimates of the boat's state. This can be fused with measurements from other sensors to give reliable information about the surroundings. When further fusing the estimates with other measurements at a higher level, occurrences of ambiguity and bad estimates would be disregarded, making this problem less critical.

Another important step before the methods are used in a real scenario is to test LK, SFM, and the estimation method with more data. The data in this thesis has been of varying quality, especially in terms of ground truth, and one should do some extended testing on more varied data to ensure that the methods work for multiple types of vessels in different scenarios. This is important as one needs to ensure the methods can give reliable estimates in different situations with different vessels. Also, it would be interesting to look at scenarios where the majority of the rotation is caused by the target rather than milliAmpere 1, which is rarely the case in the data used in this thesis.

One aspect not utilized in this project was the additional information estimated about the distance to the boat. If these methods were to be used for navigation and collision avoidance, this information could be used in addition to, or as a substitute for, other distance measuring techniques or equipment such as LIDAR. For example, if this was integrated into a complete system for collision avoidance, one could use the information from the reconstruction to estimate the time it would take before a collision occurs. This could be done without knowing the scale - one could observe how much closer objects get in the relative frame. However, the feasibility of this in practice is uncertain and warrants further investigation.

While this thesis has primarily focused on traditional computer vision techniques for detection and pose estimation, an intriguing area for future research lies in exploring the potential of deep learning methodologies. Deep learning methods could be utilized in a fully end-to-end manner, whereby the entire process from

detection to pose estimation is managed by a single network. As a substitute, deep learning could be applied to some areas of the current pipeline. For example, deep learning models could be used to enhance point matching or optical flow estimation, which are critical for accurate pose estimation. Exploring the integration of deep learning within the current pipeline is an exciting direction for future work.

Chapter 9

Conclusion

This thesis provides insight into the potential and challenges of optical flow-based object detection and pose estimation in autonomous sea navigation. The research compares the effectiveness of an optical flow-based method and a state-of-the-art deep learning model, YOLOv8, for object detection. Although the detection method based on optical flow gives more insight into the detection decisions, YOLOv8 outperforms the optical flow-based method in most scenarios, with a superior ability to generalize and lower runtime, making it the recommended method for detecting objects in milliAmpere 1's surroundings.

Further, a combination of LK and SFM is used for the 3D reconstruction of objects isolated from a scene. This approach yields accurate relative pose estimates, including yaw rate and position, based on various parameters such as the number of images used, the distribution of points on the boat, and the boat's coverage. A key observation is that the closer the boat is and the more features it has, the more accurate the pose estimates get.

By analyzing the results, it is observed that in situations where the features of the boat are visible, the precision of the method is sufficient to determine a boat's actual yaw rate and distinguish it from the relative yaw rate. This shows that the proposed approach has great potential to enhance control systems on autonomous vessels. Nevertheless, while the presented results are promising, it's clear that more research and testing are necessary before the proposed methodologies can be applied in real-world scenarios. Several challenges still need to be addressed. Notably, the optical flow-based method's precision and the fact that it could be significantly influenced by the number and distribution of points on a boat and the number of images used.

In conclusion, while requiring further work for an operational implementation, this thesis offers valuable insights and lays the foundation for future research. The concept and techniques presented here are promising and demonstrate their potential to contribute to the evolution of control systems for autonomous vessels.

Bibliography

[1] R. Wright, *Unmanned and Autonomous Ships: An Overview of MASS*. Mar. 2020, ISBN: 9780429450655. DOI: 10.1201/9780429450655.

[2] M. Hearst, S. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, 1998. DOI: 10.1109/5254.708428.

[3] S. B. Kotsiantis, "Decision trees: A recent overview," in *Artificial Intelligence Review*, vol. 39, 2013, pp. 261–283.

[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I. DOI: 10.1109/CVPR.2001.990517.

[5] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Neural Information Processing Systems*, vol. 25, Jan. 2012. DOI: 10.1145/3065386.

[6] J. Terven and D.-M. Cordova-Esparza, *A comprehensive review of yolo: From yolov1 to yolov8 and beyond*, Apr. 2023.

[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2.

[8] A. Kushwaha, A. Khare, O. Prakash, and M. Khare, "Dense optical flow based background subtraction technique for object segmentation in moving camera environment," in *IET Image Processing*, vol. 14, Dec. 2020. DOI: 10.1049/iet-ipr.2019.0960.

[9] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," in *Journal of Basic Engineering*, vol. 82, Mar. 1960, pp. 35–45. DOI: 10.1115/1.3662552. [Online]. Available: https://doi.org/10.1115/1.3662552.

[10] D. Musicki and R. Evans, "Joint integrated probabilistic data association: Jipda," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, 2004, pp. 1093–1099. DOI: 10.1109/TAES.2004.1337482.

[11] D. Reid, "An algorithm for tracking multiple targets," in *IEEE Transactions on Automatic Control*, vol. 24, 1979, pp. 843–854. DOI: 10.1109/TAC.1979.1102177.

[12] P. H. S. Torr and A. Zisserman, "Feature based methods for structure and motion estimation," in *Vision Algorithms: Theory and Practice*, B. Triggs, A. Zisserman, and R. Szeliski, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 278–294, ISBN: 978-3-540-44480-0.

[13] M. Irani and P. Anandan, "About direct methods," in *Vision Algorithms: Theory and Practice*, B. Triggs, A. Zisserman, and R. Szeliski, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 267–277, ISBN: 978-3-540-44480-0.

[14] R. Brunelli, "Template matching techniques in computer vision: Theory and practice," 2009, ISBN: 978-0-470-51706-2.

[15] E. D. H. Moe, "Optical flow for tracking maritime objects," Specialization project at NTNU. The project can be made available by the author upon inquiry., 2022.

[16] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework: Part 1," *International Journal of Computer Vision*, vol. 56, pp. 221–255, 2002.

[17] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, 2018, pp. 611–625. DOI: 10.1109/TPAMI.2017.2658577.

[18] N. Karlsson, E. Bernardo, J. Ostrowski, L. Goncalves, P. Pirjanian, and M. Munich, "The vslam algorithm for robust localization and mapping.," Jan. 2005, pp. 24–29. DOI: 10.1109/ROBOT.2005.1570091.

[19] H. Longuet-Higgins, "A computer algorithm for reconstructing a scene from two projections," in *Readings in Computer Vision*, M. A. Fischler and O. Firschein, Eds., San Francisco (CA): Morgan Kaufmann, 1987, pp. 61–62. DOI: https://doi.org/10.1016/B978-0-08-051581-6.50012-X.

[20] J. Thomas and J. Oliensis, "Dealing with noise in multiframe structure from motion," in *Computer Vision and Image Understanding*, vol. 76, 1999, pp. 109–124. DOI: https://doi.org/10.1006/cviu.1999.0779. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1077314299907798.

[21] P. Favaro, H. Jin, and S. Soatto, "A semi-direct approach to structure from motion," in *Proceedings 11th International Conference on Image Analysis and Processing*, 2001, pp. 250–255. DOI: 10.1109/ICIAP.2001.957017.

[22] J.-y. Bouguet, "Pyramidal implementation of the lucas kanade feature tracker," *Intel Corporation, Microprocessor Research Labs*, 2000.

[23] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*, Springer, 2003, pp. 363–370.

[24] P. Bideau and E. Learned-Miller, "It's moving! a probabilistic model for causal motion segmentation in moving camera videos," vol. 9912, Oct. 2016, pp. 433–449. DOI: 10.1007/978-3-319-46484-8_26.

[25] A. R. Bruss and B. K. Horn, "Passive navigation," in *Computer Vision, Graphics, and Image Processing*, vol. 21, 1983, pp. 3–20. DOI: https://doi.org/10.1016/S0734-189X(83)80026-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0734189X83800267.

[26] M. Evans, N. Hastings, B. Peacock, and C. Forbes, *Statistical Distributions*. Wiley, 2011, ISBN: 9781118097823. [Online]. Available: https://books.google.no/books?id=YhF1osrQ4psC.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," Jun. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

[28] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6. DOI: 10.1109/ICEngTechnol.2017.8308186.

[29] R. Hartley and A. Zisserman, "3d reconstruction of cameras and structure," in *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004, pp. 262–278. DOI: 10.1017/CBO9780511811685.015.

[30] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, 2011, pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[31] T. Lindeberg, "Scale invariant feature transform," in *Scholarpedia*, vol. 7, May 2012. DOI: 10.4249/scholarpedia.10491.

[32] R. Hartley and A. Zisserman, "Camera models," in *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004, pp. 153–177. DOI: 10.1017/CBO9780511811685.010.

[33] V. Ntouskos, I. Kalisperakis, and G. Karras, "Automatic calibration of digital cameras using planar chess-board patterns," in *Optical 3-D Measurement Techniques VIII*, vol. 1, Jan. 2007.

[34] M. M. Fleck, "Perspective projection: The wrong imaging model," in *IEEE Transactions on Reliability*, 1995.

[35] R. Hartley and A. Zisserman, "Computation of the fundamental matrix f," in *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004, pp. 279–309. DOI: 10.1017/CBO9780511811685.016.

[36] R. Hartley and A. Zisserman, "Epipolar geometry and the fundamental matrix," in *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004, pp. 239–261. DOI: 10.1017/CBO9780511811685.014.

[37] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," in *Commun. ACM*, vol. 24, 1981, pp. 381–395.

[38] R. Hartley and A. Zisserman, "Epipolar geometry and the fundamental matrix," in *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004, pp. 239–261. DOI: 10.1017/CBO9780511811685.014.

[39] R. Hartley and A. Zisserman, "Structure computation," in *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004, pp. 310–324. DOI: 10.1017/CBO9780511811685.017.

[40] R. Hartley and A. Zisserman, "Iterative estimation methods," in *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004, pp. 597–627. DOI: 10.1017/CBO9780511811685.035.

[41] J. Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600. DOI: 10.1109/CVPR.1994.323794.

[42] G. Mclachlan, "Mahalanobis distance," in *Resonance*, vol. 4, Jun. 1999, pp. 20–26. DOI: 10.1007/BF02834632.

[43] T. I. Fossen, "Handbook of marine craft hydrodynamics and motion control: Fossen/handbook of marine craft hydrodynamics and motion control," 2011.

[44] M. J. Todd, *Minimum-Volume Ellipsoids*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2016. DOI: 10.1137/1.9781611974386.

[45] M. J. Todd and E. A. Yıldırım, "On khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids," in *Discrete Applied Mathematics*, vol. 155, 2007, pp. 1731–1744. DOI: https://doi.org/10.1016/j.dam.2007.02.013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0166218X07000716.

[46] E. F. Brekke, E. Eide, B. H. Eriksen, E. F. Wilthil, M. Breivik, E. Skjellaug, Ø. K. Helgesen, A. M. Lekkas, A. B. Martinsen, E. H. Thyri, *et al.*, "Milliampere: An autonomous ferry prototype," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 2311, 2022, p. 012 029.

[47] Ø. K. Helgesen, K. Vasstein, E. F. Brekke, and A. Stahl, "Heterogeneous multi-sensor tracking for an autonomous surface vehicle in a littoral environment," in *Ocean Engineering*, vol. 252, Elsevier, 2022, p. 111 168.

[48] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," in *The International Journal of Robotics Research*, vol. 32, Sep. 2013, pp. 1231–1237. DOI: 10.1177/0278364913491297.

[49] B. K. Horn and B. G. Schunck, "Determining optical flow," in *Artificial Intelligence*, vol. 17, 1981, pp. 185–203. DOI: https://doi.org/10.1016/0004-3702(81)90024-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370281900242.

[50] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll'a r, and C. L. Zitnick, "Microsoft COCO: common objects in context," in *CoRR*, vol. abs/1405.0312, 2014.

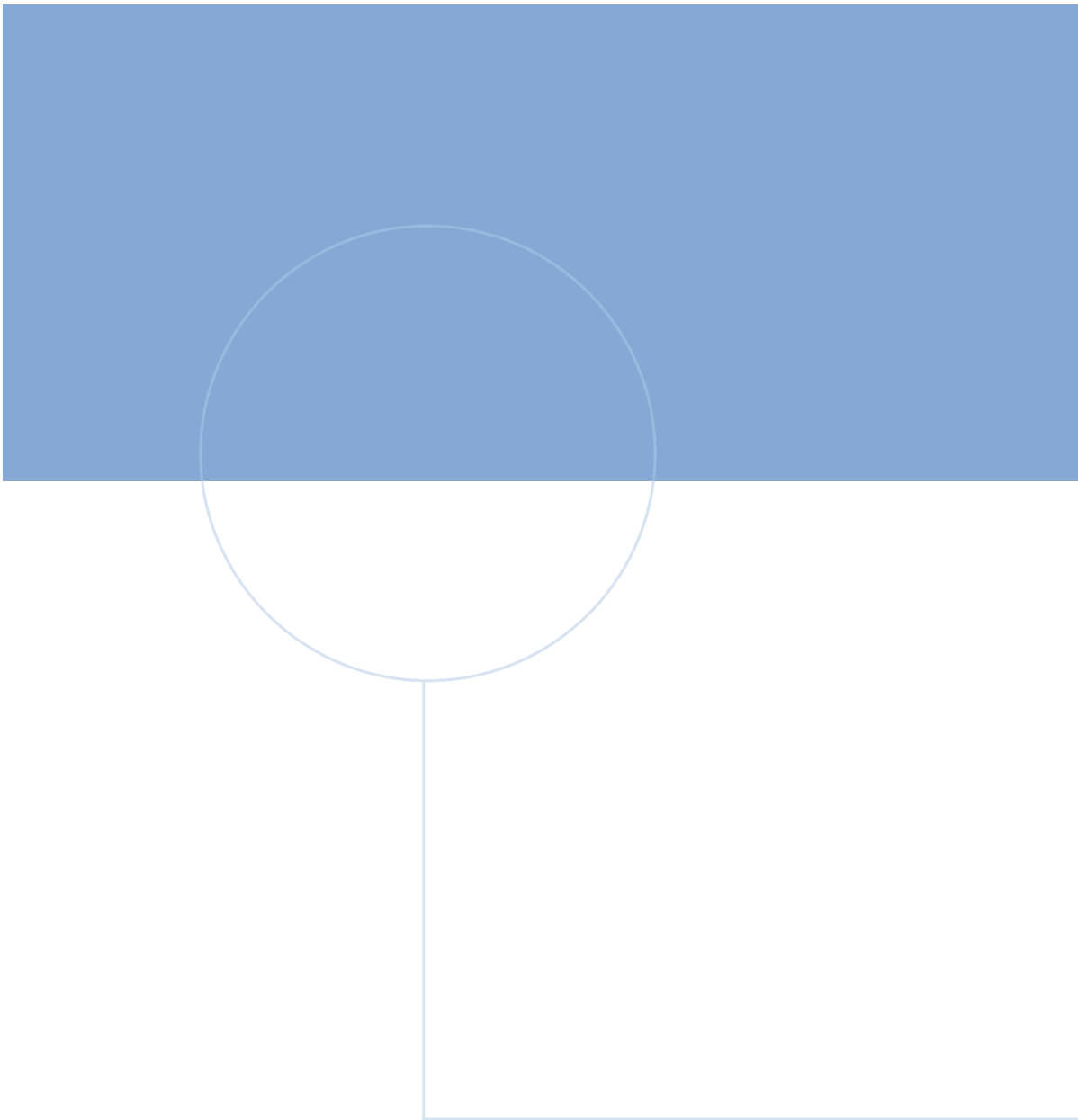
[51] D. L. Baggio, *Mastering OpenCV with practical computer vision projects*. Packt Publishing Ltd, 2012.

[52] OpenCV, *Reconstruction and images of middlebury temple.* [Online; accessed April 19th, 2023], 2023. [Online]. Available: https://docs.opencv.org/3.4/d4/d18/tutorial_sfm_scene_reconstruction.html.

[53] E. H. Adelson and J. A. Movshon, "Phenomenal coherence of moving visual patterns," in *Nature*, vol. 300, 1982, pp. 523–525.

[54] O. Chum, T. Werner, and J. Matas, "Two-view geometry estimation unaffected by a dominant plane," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, 772–779 vol. 1. DOI: 10.1109/CVPR.2005.354.

[55] A. Alouache and Q. Wu, "Evaluation of an opencv implementation of structure from motion on open source data," in *Towards Autonomous Robotic Systems: 22nd Annual Conference, TAROS 2021, Lincoln, UK, September 8–10, 2021, Proceedings 22*, Springer, 2021, pp. 158–167.



NTNU

Kunnskap for en bedre verden