NTNU

Norwegian University of
Science and Technology

Visual Estimation of Motion for ROVs

Increasing Accuracy for ROV Navigation

Einar Nonås Agdestein

Marine Technology
Submission date: June 2018
Supervisor: Martin Ludvigsen, IMT

Norwegian University of Science and Technology
Department of Marine Technology

Abstract

This project thesis presents how a remotely-operated vehicle (ROV) can increase its situational awareness and positioning accuracy based on visual inputs from a stereo camera and vision-based ego-motion estimation. This is attempted to increase the level of autonomy and to increase the ROV capabilities, repeatability and efficiency during ROV operations. The ROV in question is equipped with a stereo camera set and based on a pinhole model of the camera, 3D camera frame position of the objects in the image can be computed and tracked across the image frame. This is used to estimate the motion of the camera, and thus the motion of the ROV. Objects in the image are detected as features, and classified using the Binary Robust Invariant Scalable Keypoints (BRISK) method. Each feature in the image is described using a vector and compared to descriptors in another frame to match features.

The feature matching is implemented as a circle matching procedure, where features in the previous left frame is matched with features in the previous right, then current right, current left and back again to the previous left. This way, we can ensure that all features used in the motion estimation algorithms are matched both spatially for stereo vision and temporally for two consecutive frames. The 3D camera frame coordinates are computed based on the locations of the feature in the image frame and the intrinsic camera parameters. With a stereo set of cameras the depth of the features in the image can also be computed.

The camera frame coordinates of the features of the previous frame are then reprojected to the current image frame, using the camera intrinsic parameters. There will be an error between the locations of the features in the current image frame and the reprojected camera frame coordinates of the same features. This error is minimised using a Gauss-Newton optimisation in order to estimate the motion of the camera from the previous to the current frame. The optimisation is initialised N times for three random points, giving N estimates of the camera motion. These estimates are then compared and adjusted to get the final best estimate of the camera motion, which also includes a Kalman Filter to smooth the signal and predict the motion when the features are badly detected or matched.

The vision-based ego-motion estimation is tested on an image set from a previous mission and compared to the navigation data from the mission. The results show that the algorithm estimates the overall motion of the ROV, with some smaller oscillations around the measured value. The oscillations are most likely due to the optimisation algorithm, which sometimes overestimate the rotations over the translations.

The visual motion estimation output is merged with the measurements from the other sensors on-board the ROV and included in a Kalman Filter for state estimation. The objective is to improve underwater localisation and manoeuvring close to the seabed or close to man-made installations. To validate the results, the system has been simulated on the image set mentioned above. The simulations show good results, the VME algorithm is able to output velocity estimates at an average update rate of 2 Hz, meaning the VME can run at approximately 2 frames per second. The results from the simulation show a slightly

improved motion estimate, especially for the velocities. The performance of the Kalman Filter for position estimates was harder to evaluate, as the transponder measurements included many wild points and included an offset from the true position, and the Kalman Filter had not converged when running simulated scenarios.

This project shows how computer vision techniques can improve underwater navigation by using a stereo camera rig to estimate the ROV motion. The results from the VME implementation proved that by using feature based method of estimating the camera frame motion, the output correspond with the actual ROV motion. The feature detection with spatial and temporal matching of consecutive stereo image pairs was implemented using the BRISK algorithms, reducing the computational effort compared to other algorithms. Compared to the SURF method, the BRISK showed comparative accuracy at a considerably reduced computational time.

The next step of development could be to look at the possibility of a SLAM approach. Using the images taken to update a reconstructed map of the current surroundings. Then the ROV would recognise its position when returning to a previously visited location, while continuously updating the map. This would also considerably improve the position estimate as mentioned above. There is also some areas of improvement regarding the algorithm developed for this project, both in terms of accuracy and computational time.

Preface

This master thesis is a continuation of the effort made during the work on the project thesis in the fall semester of 2017. The objective of the master project is to use computer vision and visual input from stereo cameras to estimate the motion of a remotely operated vehicle (ROV) close to the seabed or installations. The system has been developed from a Matlab programmed off-line application only suitable for simulation, to a system that can be deployed in the ROV control system developed here at NTNU.

The visual motion estimation algorithm is programmed in C++, using the computer vision toolbox OpenCV. The author of the thesis did not have extensive experience with the C++ programming language before embarking on this project, and thus the learning curve during the semester has been very steep. This means the algorithms are not perfect, but does work for the application they are designed for.

The goal was to test the final product on the newly acquired Minerva 2 ROV, however, because of delays on delivery, the system has only been simulated on data and images from previous missions on another ROV at NTNU, the SUBfighter-30k.

The work on the thesis has been performed at the Department of Marine Technology, NTNU, with Professor Martin Ludvigsen as the main supervisor. I would like to thank Professor Ludvigsen for feedback and discussion on the topics of this thesis, both during the group meetings with the Minerva 2 project and the one-on-one meetings regarding this thesis.

I would also like to thank PhD candidate Stein Melvær Nornes for support on technical issues and discussions during meetings and drop-ins this last semester. His help with the software LabVIEW has been instrumental for the completion of this thesis. A special thanks to Stein for technical assistance with the stereo camera rig, without which the project could not have been completed.

In addition, I would like to thank friends and family for support and help with the proofreading of the final report.

Trondheim, June 11, 2018

Einar Nonås Agdestein

Table of Contents

List of Tables

List of Figures

Abbreviations

AUV	=	Autonomous Underwater Vehicle
ROV	=	Remotely Operated Vehicle
IMU	=	Inertial Measurement Unit
DVL	=	Doppler Velocity Log
VME	=	Visual Mostion Estimation
SNAME	=	Society of Naval Architects and Marine Engineers
ECI	=	Earth Centered Inertial frame
ECEF	=	Earth Centered Earth Fixed frame
NED	=	North East Down frame
DOF	=	Degrees of Freedom
APS	=	Acoustic Positioning System
LBL	=	Long Baseline
SBL	=	Short Baseline
SSBL	=	Super-Short Baseline
SIFT	=	Scale-Invariant Feature Transform
SURF	=	Speeded Up Robust Features
ORB	=	Oriented FAST Rotated BRIEF
BRIEF	=	Binary Robust Independent Elementary Features
FAST	=	Features from Accelerated Segment Test
BRISK	=	Binary Robust Invariant Scalable Keypoints
KF	=	Kalman Filter
EKF	=	Extended Kalman Filter
ECF	=	Explicit Complimentary Filter
AUR-lab	=	Applied Underwater Robotics Lab
SLAM	=	Simultaneous Localisation And Mapping
AMOS NTNU	=	Centre for autonomous marine operations and systems at NTNU
GNSS	=	Global Navigation Satellite Systems

Chapter 1

Introduction

1.1 Background

1.1.1 ROV

The oceans of the world are still for the most part unexplored areas, and to explore distant areas and great depths, underwater vehicles are instrumental in further development. Two classes of underwater vehicles dominate the nomenclature, namely the Autonomous Underwater Vehicle (AUV) and the Remotely-Operated Vehcile (ROV). The main differences between these two reside in their name, the one being autonomous and the other remotely-operated by an ROV pilot.

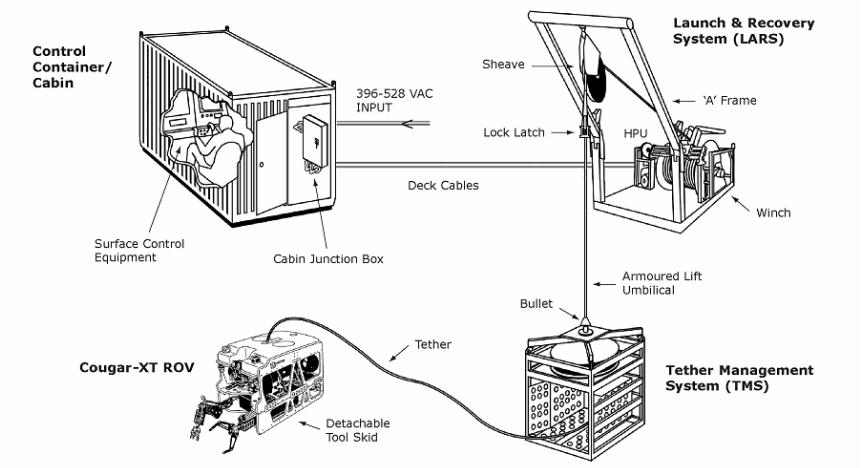


Figure 1.1: *ROV system with control room, launch and revovery system, tether and tether management system, umbilical and ROV. Courtesy of SAAB*

The ROV is here an underwater remotely operated vehicle, which consists of a frame with buoyancy elements, thrusters for motion, video cameras for control and typically manipulator arms. All communication with the ROV is sent through the umbilical that connects the ROV to the surface. An example of a typical ROV - surface system is presented in Figure 1.1

The first use of ROVs can be traced back to Royal Navy in the 1950s to recover torpedoes and mines from the sea floor. An example is shown in Figure 1.2a, where the Royal Navy's Cutlet ROV is pictured. Further on in the first few decades the ROV, it was mainly the navies of the world that was driving the development of the unmanned underwater vehicle. When the oil industry went offshore, however, the ROVs finally became available for commercial use. [1]

The first to explore the commercial side of the ROV market was the US-based Hydro Products in the 1970s. The Hydro Products RCV 255 and RCV 150 ROVs are pictured in Figure 1.2b. However, the offshore industry still used manned submersibles and saturation divers for installations and maintenance. In the 1980s, the ROV industry grew rapidly, inspired by the technological advancements, such as miniaturisation of the electronics industry. [1]

(a) *Royal Navy's Cutlet ROV from the 1950s, which was used for recovery of torpedoes and mines*

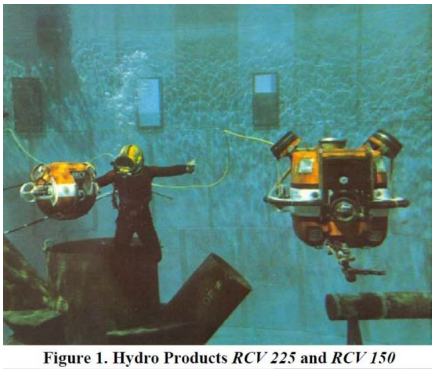


Figure 1. Hydro Products *RCV 225* and *RCV 150*

(b) *Hydro Products RCV 255 and RCV 150 ROVs, one of the first ROVs for the commercial market*

Figure 1.2: *Two examples of the ROV in its early technological development*

The ROV became smaller and more portable, and the price was pushed down such that civil organisations and academic institutions now could afford it. These observation class vehicles soon began in the 80s to perform tasks like civil engineering, dam and tunnel inspections, fisheries inspections and oceanography. Together with the already growing offshore market, the ROV had come to stay.

In the 1990s, the ROV industry finally reached full maturity and records were broken one by one. The Japanese Kaiko ROV reached the deepest point in the Mariana Trench, 10 909meters below sea level, a record impossible to break. The offshore oil industry was now also exploring fields way beyond diver depth (up to 3000 m), enabling ROV developers and the oil industry to team up and design integrated systems for installation,

operation and maintenance of offshore oil fields. By the end of the 1990s, it is estimated that more than 3000 vehicles were in operation by more than 100 users.[1] The amount of ROVs in operation today is a number hard to determine due to the exploding number of smaller ROVs.

1.1.2 ROV Autonomy

The control system development for the ROV has also followed the main technological development described above. In the beginning the ROV pilot had to individually control each thruster on the vehicle. However, soon the automatic depth control eased the operation, as most ROVs are not neutrally buoyant. With proper thrust allocation and compasses, automatic heading control also became possible.

Depth and heading control was for many years standard functions, and only in later years, the scientific community has been focused at making the underwater vehicle more autonomous, occluding the line between the previously mentioned AUV and the ROV. The level of autonomy is then a good way to divide the underwater vehicles into different classes. There are several definitions of the levels of autonomy that includes the different levels of human-robot interaction. Most are defined from manual or remote control, teleoperation, semi-autonomous to highly autonomous operation. A definition proposed in [2], which is also recommended here at NTNU is:

1. **Automatic operation (remote control):** The system operates automatically, but the human controller controls high-level mission-planning functions. Also called human-in-the-loop or human operated.

2. **Management by consent (teleoperation):** The system can now make recommendations for specific functions, but still notify the human controller with information and asks for decisions. The system can also perform some functions delegated by the human controller. Also called human-delegated system.

3. **Semi-autonomous or management by exception:** The system executes mission-related functions automatically when the response time is too short to prompt the human operator. The operator can still override or cancel decisions made by the system. Also called human-supervisory control.

4. **Highly autonomous:** The system executes mission-related functions without notifying the human controller, and is also able to plan and re-plan the mission. The human operator can still be informed of the progress, but decisions are left to the independent system. Also called human-out-of-the-loop.

In the last couple of decades, many more automated functions has been developed, such as station keeping, path tracking and velocity control. Global positioning systems (GPS) does not work underwater, thus other means of measuring the position and velovity are needed. A similar absolute position system underwater has always been the acoustic positioning system (APS), however, the update rate can be quite slow, and the readings can be erratic and inaccurate. A suitable alternative is the inertial navigation system (INS) used on submarines, but these have been too large and expensive for smaller ROVs.

The main sensors for the ROV today are typically the doppler velocity log (DVL) measuring the velocity, an inertial measurement unit (IMU) measuring the accelerations and a transponder to measure the position by acoustics. For the ROV control system developed at NTNU AMOS ([3], [4], [5]) these three sensors are the main source of navigational data.

The problem with this is the accuracy of the sensors mentioned. Manoeuvring and localisation of underwater vehicles is likely the biggest challenge for autonomous operation of underwater vehicles. The receivers of the Global Navigation Satellite Systems (GNSS) does not work underwater, thus the global position is not easily acquired as it is for surface vessels. As mentioned above, acoustic positioning systems, IMU and DVL have traditionally been used to find and estimate the position, velocity and attitude in a reference frame.

The problems with the sensors for underwater navigation and localisation are problematic for the accuracy. The acoustic system can be inaccurate and has a slow update rate, the acceleration measurements have to be integrated to compute the velocity and position, which can cause drift. The DVL is quite accurate, especially close to the seabed. However, when talking about centimetre accuracy at depths ranging to a few thousand metres, more intelligent systems are needed.

ROV autonomy is an important research area at NTNU Amos [2], and with increased accuracy of underwater navigation, many ROV operations can be made autonomous. Tasks such as manoeuvring, inspection, sampling and manipulation can be automated, facilitating ROVs that can live at the seabed for an extended amount of time, reducing the cost of deploying a ROV supply vessel each time the ROV is needed. One step towards increased autonomy in ROV missions is thus improved underwater navigation, a field in which computer vision techniques can offer parts of a solution.

1.1.3 Computer Vision

As the literature review in Section 1.4 will explain in more detail, the use of visual motion estimation has not been widely explored for the subsea environment. There are a lot of reasons for this, but mainly poor lighting conditions and a poor medium for cameras are important issues. Above sea, the development in recent years have been great, and there are a lot of open-source algorithms out there, exploring everything from security around and in buildings, to self-driving cars. Later sections will describe this in detail, in addition to what has been done in the subsea environment.

As humans, we can perceive the 3D world around us with ease, having the ability to distinguish depth, shape and translucency of objects we see. Over the last 40-50 years, researchers in computer vision have been developing mathematical models to facilitate computers to have the same visual perception as humans. Now, there are techniques that can render a 3D model of a complete scenery using thousands of overlapping photographs. This includes building surface models using stereo vision, tracking a person or an object or even name people in a photograph using recognition techniques on faces, clothing and hair. [6]

Despite all these advances, a computer cannot interpret an image to the degree even a 2-year old kid can today. Computer vision is a difficult subject to study, in part because it is an inverse problem, where we have insufficient information to find some unknowns. To

solve this problem, physics-based and probabilistic models are used to remove uncertainties in the possible solutions. Modelling the visual world turns out to be a quite dificult task to accomplish.

The possible applications for computer vision are many. Examples are character recognition (letters and numbers), inspection of machinery such as cars or aeroplanes, medical imaging, automotive security up to self-driving cars, motion capture for computer animation, image stitching, 3D modelling and more. Some of the applications are presented in Figure 1.3.

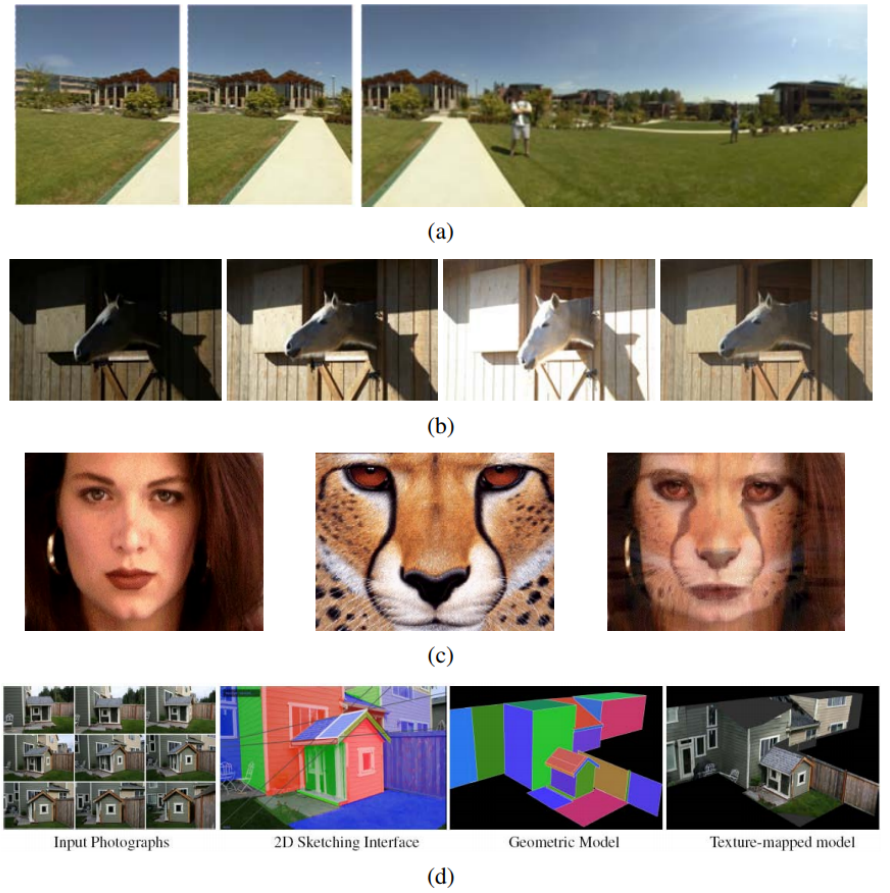


Figure 1.3: *Some examples of applications for computer vision: (a) image stitching; (b) merging different exposures; (c) morphing, blending between two photographs; (d) turning several images into a 3D model of the scene. [6]*

The first mention of computer vision dates back to the 1960s, when Marvin Minsky at MIT wanted an undergraduate to spend his summer linking a camera to a computer and describe what it saw. Of course it would never be that easy. In the beginning of the 1970s computer vision was perceived as an agenda to mimic human intelligence in robots.

Digital image processing had already been around for some time, and computer vision was distinguished from this field as a desire to recover a 3D structure of the world towards full artificial understanding of a scene.

The first attempts at this was directed at extracting edges from the topological 2D lines in the image. Both a qualitative approach to understand intensities and a more quantitative approach to feature-based stereo correspondence were developed at this time. Later in the 1980s, much of the attention was directed at the mathematical techniques for doing analyses of images and scenes. Researchers discovered that if the algorithms could be posed as variational optimisation problems, many of them could be unified in a single mathematical framework.

The 1990s saw many improvements regarding optical flow methods and multi-view stereo algorithms. The stereo algorithms could be used to produce complete 3D surfaces of a scene, a field of research which continue to be active today. Another accomplishment at this time was the tracking algorithms, which improved a lot by using intensity-based techniques, often applied to tracking faces. Computer vision also became more and more interactive with the field of computer graphics, as the multi-view stereo algorithms became better, the computer graphics had to keep up in order to animate the 3D models.

This development has only continued, and the last few decades has seen rendering, image stitching and light-field capture being brought back as computational photography as the interaction between computer vision and graphics has developed. The last trend in the field of computer vision is the application of machine learning techniques to computer vision problems. This has exploded lately as more and more information has been made available on the internet, including labelled data that can be used to learn object categories for computers.

The computer vision techniques used in this thesis is mainly feature detection and matching. In [6] the feature detection and matching are suggested split into four stages:

1. **Feature detection:** Each image is searched for features that are suitable to match with features in other images.

2. **Feature description:** The region around each feature is described in a compact and stable manner to make matching easier.

3. **Feature matching:** The features are matched with features in other images comparing the feature descriptors.

4. **Feature tracking:** An alternative to stage three more suitable for video processing as the algorithm searches in the immediate neighbourhood of the feature.

This is similar to the method used in this thesis and will be explained in detail in Chapter 3

1.2 Objectives

This master thesis is a continuation of the project thesis of the fall 2017 and is about how a remotely-operated vehicle (ROV) can increase its situational awereness and accuracy of

positioning based on visual inputs from a camera. To work on man-made installations on the seabed, the ROV will not be able to rely solely on absolute global navigation. To find and interact with small and delicate instruments, spacial references with centimetre accuracy are required. In most ROV operations the topside pilot is controlling the vehicle, through video taken from the ROV.

The key problem for this thesis is the accuracy of underwater navigation. This problem must be solved to automate control of underwater vehicles. ROV autonomy is an important research area at NTNU AMOS in order to increase safety for personnel involved in ROV operations and to reduce costs. There is also a high motivation to increase the ROV capabilities, repeatability and efficiency to move towards the intervention AUV [2]. This is especially true for ROV operations far offshore, where the vessel and vessel personnel is a key factor for both security and costs. The improvement of underwater navigation is crucial for close range navigation for tasks such as docking, manoeuvring inside structures or autonomous control of manipulators.

The challenges of underwater navigation today is closely connected to the limitations of acoustic positioning and dead-reckoning navigation and must be overcome in order to automate ROV tasks such as manoeuvring, inspection, sampling and manipulation, etc. Computer vision techniques, developed in the last few centuries, can offer a part of the solution to these challenges. Thus, the development of the VME, and including the output in the ROV state estimator is given much attention in this thesis.

1.3 Scope and Limitations

The scope of this master thesis will be a direct continuation of the work done on the project thesis. The project includes deriving motion from stereo imagery online and combining the visual motion estimation (VME) with the other sensors using state estimators. To test the performance of the system it should be simulated with the ROV control system. In the project thesis the VME was developed in Matlab and the output was compared with the data from the ROV. In this thesis the VME will be implemented in C++ using the open source library for computer vision and machine learning, OpenCV. The VME algorithms will only output the change in position from frame to frame, thus it is not a SLAM approach where the 3D point clouds are used to form a map of the surroundings.

The output from the VME will be compared with measurements from other sensors on the ROV and then be simulated in the ROV Kalman Filter state observer in the ROV control system. The Kalman Filter will be altered to include the measurements from the VME as a 6 DOF velocity vector. The images used for simulation originate from a previous mission, thus the simulation will not be run in the ROV control system developed in LabVIEW, but in a similar Matlab setup with the raw measurements from the same mission and the output form the VME

The project thesis was a preliminary work into this master thesis project. Thus, some theory has been taken from that thesis, as well as some of the methods used. Especially for the theory and methods regarding the feature detection and matching across consecutive stereo frames. This is, however a larger analysis of the computer vision application to ROV motion estimation. The theoretical background from ROV modelling and hydrodynamics and ROV control system with the Kalman Filter state observer is also included.

The VME will be included in the ROV control system developed at NTNU AUR-lab for the LabVIEW object-oriented programming tool. The system was meant to be tested online in the ROV control system with the newly acquired Minerva 2 ROV. However, due to delays from the manufacturer, sea tests of the system was not accomplished.

1.4 Literature Review

Autonomous underwater vehicles have developed over the past decades as one of the main drivers of exploring the oceans [7]. As mentioned in Section 1.1, the two main classes of underwater vehicles, are the AUV and the ROV. The former is already autonomous, while the latter is a vehicle traditionally deployed form a ship or from land, monitored and controlled through an umbilical between the ROV control room and the vehicle. Both regarding the ROV control system and computer vision, studies are made to increase the level of autonomy for the ROV, although the research regarding surface vehicles has been greater. Here we will look at progress made in both areas.

1.4.1 ROV Control System

The ROV control system developed at NTNU is presented in [3], with the additions for path planning in [4]. The problem of navigation, as stated by Dukan [3], is one of the main problems to be solved for automation of underwater vehicles. In transit the basic sensors are the transponder, giving the position of the ROV from acoustic positioning. The inertial measurement unit (IMU) gives the ROV estimates of the vehicle attitude and can provide estimates of the velocities based on the integration of the accelerations in 6 degrees of freedom [3].

However, for approach and intervention, more detailed sensor measurements will be required to avoid undesirable situations. The DVL uses the Doppler shift to calculate the velocity of the ROV based on acoustic signals sent form transducers to the seabed or water column. Thus a second sensor for obtaining the close-range localisation and situational awareness based on the camera equipped on the ROV could increase the navigational accuracy during approach and intervention.

As already mentioned in Section 1.1, ROV autonomy is an important research at NTNU AMOS [2]. An autonomous ROV could facilitate for increased capabilities, efficiency and repeatability in underwater operations. The research are driving the ROV towards the concept of an intervention AUV. The advances in recent years have been great, including the work of [3] and [5] as already mentioned, as well as the semi-autonomous agent architecture for a ROV [8].

1.4.2 Computer Vision

Several studies and papers have been written on vehicle localisation and ego-motion based on visual odometry on land, however, the research base on the same problem underwater is narrower. Some work has been done on photomosaic in underwater operations, such as the work by [5], [9] and [10]. The photomosaic technique can provide a representation of the underwater habitat, which can be used in mapping applications and monitoring missions of

underwater areas. The individual pictures taken are stitched together, providing a seamless picture of the terrain covered.

The process is done by acquiring images, extract features and descriptors, and match these across multiple images. When stitching the pictures together, this will create a seamless image of the motive. This is similar to the method developed in this thesis, however, with the included motion estimation between frames. As photomosaic techniques focus more on the construction of an underwater map or for monitoring purposes, not the navigation data that can be extracted from this, the land-based methods will be given more attention in this thesis.

The work done by [11] and [12] both address the problem of using visual features in the image for vehicle localisation and ego-motion estimation. By ego-motion estimation we mean the estimation of the camera based on movements of the features in the image. Another renowned project that have implemented this type of system is NASAs Mars Rover [13]. The Mars Rover uses input from its multiple cameras to estimate its motion, as a global navigation satellite system is yet to have been developed for the planet of Mars.

The two most common approaches to extract information from a camera to compute motion is the optical flow method and the feature descriptor method. Optical flow is, according to the early works of [14], the distribution of velocities of the motion in an image. This motion is caused by the relative movement of the objects in the image and the camera. Thus, optical flow can give information about the spatial arrangement and the rate of change of this spatial arrangement.

Features are points of interest in an image that are invariant to scale, rotation, illumination and viewpoint. Features is instrumental in many computer vision applications, such as motion estimation, 3D modelling, video tracking, photomosaics and object recognition to name a few. A feature can be a specific point in an image with a local intensity maximum or minimum, a corner or an edge. The process of finding these features usually includes investigating the gradient, i.e. the change of intensity in both the vertical and horizontal direction [6].

Since the early introduction of optical flow to the world of computer vision [14], the technique has improved. The algorithms does not have an efficient way of dealing with outliers, as stated by [15], and many different functions have been explored since then, as [16], [17] and [18]. The choice of method highly depends on the application for the method, the accuracy required and the computational effort. However, most methods rely on constant brightness in the image [15], which is a difficult task to accomplish in the underwater environment.

Another approach is the feature based method of estimating motion. As the name implies, also this method is dependent on the image features defined above. The features are detected based on the same principle as stated above, however, for each feature detected there is a corresponding descriptor. The descriptor is comprised of a vector, describing the feature and its neighbourhood. There are several feature detectors and descriptors, and papers have been written comparing their performance, both with respect to accuracy and computational effort [19, 20].

The detectors and descriptors with most coverage are the SIFT, SURF, BRIEF, FAST, BRISK, ORB and MSER. When selecting which detector-descriptor combination to utilise for a given application, there is always a trade-off between accurate and robust detection

and computational effort [19]. In an image with a high number of potential features, the ORB and BRISK methods are the best, showing a potential of tens of thousands of features detected [20]. However, the accuracy of the SIFT and SURF methods is still the highest achievable of the methods mentioned here.

SIFT and SURF are based on obtaining vector-based descriptors of real numbers, thus requiring a large amount of computational effort as well as a large uplink bandwidth if the system is online [19]. The BRISK and ORB methods are favourable for an online system that has limited computational effort, such as an autonomous underwater vehicle. The ORB and BRISK descriptors are binary descriptors that are computed directly on the image patches. The binary descriptor consists of sampling a pattern, compensating the orientation and then sampling in pairs. This means the computational time is significantly reduced compared to the vector-based descriptors of real numbers of SIFT/SURF. When comparing the ORB and BRISK, it is clear that the ORB is faster. However, with BRISK the gains are increased repeatability and accuracy [20].

When the features are matched across several images, the motion of the camera can be estimated based on the change of the features' position in the image frame. By using the intrinsic parameters of the camera, a 3D model of the objects in the image is obtained and the movement of features to the next frame is inversely used to compute the camera's movement.

As mentioned above, the work done by [11] and [12] address the problem of vehicle ego-motion estimation. The method proposed by [12], finds the camera motion using a least squares optimisation. The basic principle here is to reproject the 3D coordinates from the previous frame to the current image frame, and finding the motion based on a Gauss-Newton least-squares optimisation.

According to [12], the Gauss-Newton optimisation only takes 4-8 iterations to converge, and is robust against outliers. The paper describes the 3D reconstruction of a scene, however, the ego-motion part of the problem is claimed to run at approximately 25 frames per second, while the depth map runs at 3-4 frames per second. This will be fast enough for online applications in an autonomous ROV.

This method of reprojecting the previous 3D points to the current image frame is also utilised in the simultaneously localisation and mapping (SLAM) approach, more particularly in the paper by [21]. The SLAM techniques constitute the most common and succesful approach to perform precise localization in unknown environments. The focus lately has been on enhancement of visual SLAM techniques, which in some cases integrate, in an Extended Kalman Filter, the dead-reckoning data, the landmark data and loop closings.

The paper proposes consecutive stereo pairs inserted into the navigation architecture. From each stereo pair of images, the system extracts image features, matches them reciprocally, and computes their 3D coordinates using the stereoscopy principle. The features corresponding to each pair are stored in a database together with their 3D points and a node identification number. The current pose of the vehicle is then calculated by computing the pose of the last graph node with the camera displacement computed by a stereo odometer.

The method is adapted in the paper as (a): recover and match 2D features of current node and a node candidate for loop-closing; (b) If number of matches is less than a given threshold, recject, otherwise; (c) extract the 3D coordinates of current node; (d) back-project the 3D coordinates of current node to 2D features of candidate node, assuming the

existence of rotation and translation, and applying RANSAC to eliminate outliers.

1.4.3 Underwater Imagery

Cameras have been standard equipment for underwater remotely operated vehicles (ROV) for a long time, and the idea of using the cameras as an aid in the ROV autonomous navigation is getting more and more attention in the underwater robotics community. The study of computer vision has been a target for the underwater robotics community for a long time already, given that video cameras are still the most important source of information for the ROV pilot. Some of the research and the corresponding challenges of underwater imagery will be presented here.

The long-range detection is still accomplished by sonar tracking, however, when close to an object or the seafloor visual sensors can be used on the final approach. ([22]) Unfortunately the lighting conditions on the seafloor are very poor, and the features of artificial light in seawater, such as absorption, scattering and distortion, make the vision task difficult. The problem of radial distortion is also more significant in underwater imagery due to the difference in refractive index between water and air. ([9]) If there is high overlap of images, which is typical for video or a slow-moving camera, the effects of the distortion is not as visible as if the overlap of images are low.

Based on the information above, the contrast in underwater imagery can be quite poor. The detection of features in an underwater image can be done in multiple ways. The colour-based underwater object recognition using water light attenuation proposed in [22] argues that colour is a simple and robust information piece in underwater imagery, and often the most reliable. Another method is a saliency-based bottom-up system, which first subtract the background in the image, leaving objects of interest in the frames. The frames are then divided into 7 channels at 6 spatial scales, which are combined again, making up the saliency map, identifying the objects in the image [23].

The final, and possibly most used, method of finding features in an underwater image, or in an image in general, is the edge and corner detectors. These algorithms investigate the intensity map on the image, and where the gradient of the intensities across the image pixels is high, an edge or corner can be identified. The corner detector used in [9], and the SURF method used in [24] all use the change of intensities across the image pixels to identify a point of interest in an image. This type of feature detection can also benefit greatly from increasing the contrast in the image, for example using the contrast-limited adaptive histogram equalisation, detailed in [25].

1.5 Structure of Thesis

The structure of the thesis is presented thematically, with the ROV and computer vision being the two main themes. In these thematic chapters both theory and methods are discussed, with the first being based on previous work and literature and the latter based on work done for this thesis. The results and discussion are presented in separate chapters. The structure is defined as:

- **Chapter 1: Introduction.** Here the background and motivation are explained in addition to introducing the objectives and scope of the thesis. A literature review

is also included here to discuss what has been done regarding computer vision and underwater imagery.

- **Chapter 2: Remotely Operated Vehicle.** The chapter introduces the ROV with modelling and hydrodynamics, forces, sensor and the ROV control system with the Kalman Filter being the main topic.
- **Chapter 3: Computer Vision.** The chapter introduces the camera model, the feature based method of estimating motion with feature detectors and matching. The optimisation algorithm to solve for the motion is introduced and discussed.
- **Chapter 4: Simulation.** The chapter explains the simulation set-up and presents the results from the simulation of the VME with the ROV Kalman Filter.
- **Chapter 5: Discussion.** Discussion of the results compared with the objectives made and the theory and methods used.
- **Chapter 6: Conclusion.** Concluding remarks and recommendations for further work

1.6 Thesis Contribution

The two main topics of this thesis is the application of computer vision techniques to estimate motion underwater, and the inclusion of these motion estimates in the ROV control system developed at NTNU. The goal of doing this is to increase the accuracy of ROV navigation to increase the level of autonomy. As discussed in Section 1.4, the computer vision techniques of estimation ego-motion of the camera frame is extensively researched and developed. However, most of these techniques have been applied on surface technology.

In this thesis these techniques have been taken underwater and installed on a stereo camera set mounted on a ROV. There have been some studies and promising results regarding SLAM for underwater vehicles, however, this has not been attempted in this thesis. The contribution of this thesis has been to investigate whether a feature-based method of estimating camera motion will function in the water medium, as it has been proven to work on the surface.

The improvement of accuracy for underwater navigation is instrumental to increase the level of possible autonomous operations for a ROV. This should as mentioned in [2] include computer vision techniques, and thus this thesis can give valuable input to the application of computer vision to improve accuracy of underwater navigation

Chapter 2

Remotely operated Vehicle

2.1 Background

A ROV is a remotely operated vehicle, in this work, an underwater vehicle, that is controlled by a trained pilot, usually in a control room on-board a larger surface vessel. The ROV consists of a frame with buoyancy elements, thrusters, video cameras, sensors and some sort of payload designated by the type of mission. A common feature of an ROV is a manipulator arm, giving the ROV the possibility to perform maintenance and intervention tasks on the seabed. Power to the vehicle, as well as all signals that are transmitted back and forth between the control room and the ROV is sent through a cable, normally called an umbilical

The ROV is not designed to be hydrodynamically efficient, thus the box-shaped design. This will however, have minimal impact on the overall drag force on the vehicle, as the umbilical connecting the ROV to the top-side control room represents most of the total drag force on the ROV. Thus, making the ROV hydrodynamically efficient will only make a tiny dent in the overall drag force experienced by the system.

2.2 Modelling and Hydrodynamics

2.2.1 Notations

The notations that are used in the thesis are taken from the SNAME convention and from Fossen's vectorial model [26, 3]. The position, orientation and linear and angular velocities are given by generalised coordinates and for an ROV there are 6 degrees of freedom presented in Table 2.1

Generalised Coordinates

The equations of motion describing the mathematical model of the ROV and hydrodynamics are presented based on the Fossen's robotic-like vectorial model [26]. The purpose of

Table 2.1: *ROV degrees of freedom*

Number	DOF	Forces & Moments	Positions & Orientation	Linear & angular velocities
1	Surge	X	x	u
2	Sway	Y	y	v
3	Heave	Z	z	w
4	Roll	K	ϕ	p
5	Pitch	M	θ	q
6	Yaw	N	ψ	r

building a model of the ROV and the hydrodynamics is to simulate the ROV motion in the observers, controllers and thruster allocation. The models are based on the modelling performed for the ROV control system developed at NTNU by [3], [5] and [4].

The generalised coordinates for position and velocity are given by (2.1) and (2.2).

$$\boldsymbol{\eta} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T \tag{2.1}$$

$$\boldsymbol{\nu} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T \tag{2.2}$$

The linear and angular position and velocity vectors are sub-vectors of the generalized coordinates and are given by (2.3).

$$\boldsymbol{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \boldsymbol{v} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \quad \boldsymbol{\Theta} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \quad \boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.3}$$

where $\boldsymbol{p} \in \mathbb{R}^{3\times1}$ is the linear position, $\boldsymbol{v} \in \mathbb{R}^{3\times1}$ is the linear velocity, $\boldsymbol{\Theta} \in \mathbb{R}^{3\times1}$ is the angular position, or the attitude, and $\boldsymbol{\omega} \in \mathbb{R}^{3\times1}$ is the angular velocity, or the turn rate of the vehicle.

Notation Norms

All matrices are expressed in boldface, upper case letters, while vectors are expressed in boldface, lower case letters. An estimate of a variable is expressed as $\hat{x}$, the estimate of the variable x, the time derivative of the variabel is expressed as $\dot{x}$, and the time derivative of the estimate of the variable is expressed as $\dot{\hat{x}}$. The cross product of two vectors $a, b \in \mathbb{R}^3$ can be calculated using the cross product operator, the skew symmetric matrix $\boldsymbol{S}$ as $\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{S}(\boldsymbol{a})\boldsymbol{b}$. The cross product operator $\boldsymbol{S}$ is defined as in (2.4).

$$\boldsymbol{S}(\boldsymbol{a}) = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}, \quad \boldsymbol{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \tag{2.4}$$

Given the skew symmetric properties; $\boldsymbol{S} = -\boldsymbol{S}^T$.

2.2.2 Kinematics

Reference Frames

There are several reference frames used for navigational purposes and reference frames to describe the motion of a vehicle. The three most common navigational reference frames are

- ECI, $\{i\}$: The Earth-centred inertial frame with axes $\{i\} = [x_i, y_i, z_i]$
- ECEF, $\{e\}$: The Earth-centred Earth-fixed frame with axes $\{e\} = [x_e, y_e, z_e]$
- NED, $\{n\}$: The North, East, Down frame with axes $\{n\} = [x_n, y_n, z_n]$

The three reference frames can be seen in Figure 2.1, where the body frame of a satellite is also present. As depicted in the figure, the ECEF frame is rotating with respect to the ECI frame with angular rate ω_e.

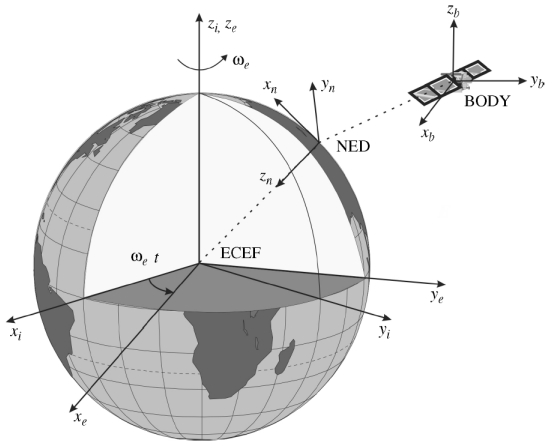


Figure 2.1: *The reference frames w.r.t. the Earth-Centered Inertial (ECI) frame. [26]*

The $\{i\}$ frame is assumed to be inertial if high accuracy is important. However, for slowly moving vehicles, the $\{n\}$ frame can be assumed inertial for most purposes and applications. The $\{e\}$ frame can be used when the flat Earth approximation of the $\{n\}$ is not applicable, i.e. for motion over long distances. GPS coordinates are given in this frame, however, it is unpractical to display the attitude in this frame.

- BODY, $\{b\}$: The body frame with axes $\{b\} = [x_b, y_b, z_b]$
- MES, $\{m\}$: The measurement frame with axes $\{m\} = [x_m, y_m, z_m]$

The two coordinate frames that are fixed to the vehicle are called the body-fixed coordinate frame and the measurement frame. The body fixed frame is fixed to the vehicle

origin and rotates with the vehicle. The measurement frame is usually moving and rotating with the body frame. Thus, the measurement from an instrument mounted on the vehicle are expressed in the measurements frame, and moving with the body frame. There can be multiple measurement frames on the vehicle, depending on the number of instruments doing measurements. The ROV body frame is presented in Figure 2.2

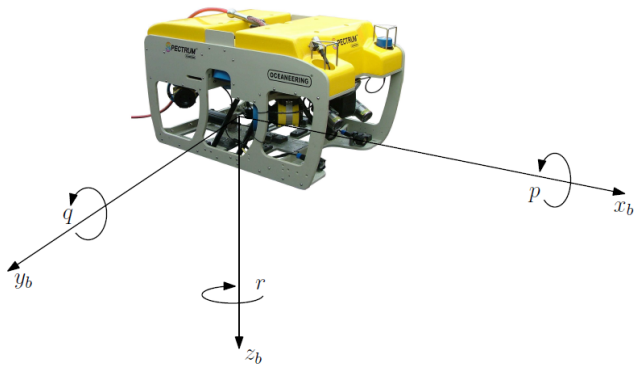


Figure 2.2: *ROV BODY frame, with axes and rotations. [3]*

The body frame of a ROV has axes defined as surge, sway and heave, and the rotations according to the right-hand rule, roll, pitch and yaw around the surge, sway and heave axes respectively.

Vector Notations

Sub- and superscripts are used to specify which reference frame the vector is decomposed in and the start and endpoint of the vector. A vector $\boldsymbol{p}$ that is decomposed in one frame (frame $\{a\}$), can be expressed in another (frame $\{b\}$) using a transformation matrix $\boldsymbol{R}_a^b(\Theta_{ba}) \in \boldsymbol{R}^{3\times3}$ as $\boldsymbol{p}^b = \boldsymbol{R}_a^b(\boldsymbol{\Theta}_{ba})\boldsymbol{p}^a$. The superscript express which frame the vector is decomposed in, and the subscript contains relevant information about the vector. The rotation of a coordinate system (frame $\{a\}$) with respect to another (frame $\{b\}$) is expressed by the rotation vector $\boldsymbol{\Theta}_{ba}$.

Transformations

Translational transformations

As explained in the section above, the transformation matrix $\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb})$ can be used to rotate a vector expressed in the $\{b\}$ frame of the vehicle to the $\{n\}$ frame. The transformation matrix is given in (2.5).

$$\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) = \boldsymbol{R}_{z,\psi}\boldsymbol{R}_{y,\theta}\boldsymbol{R}_{x,\phi} \tag{2.5}$$

where

$$\boldsymbol{R}_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}, \boldsymbol{R}_{y,\theta} = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}, \boldsymbol{R}_{z,\psi} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

s and c are short for sine and cosine as $c- = cos(-)$ and $s- = sin(-)$. The inverse transformation is given in (2.7)

$$\boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb})^{-1} = \boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb})^T = \boldsymbol{R}_n^b(\boldsymbol{\Theta}_{nb}) \tag{2.7}$$

This representation of the angles has a singularity at 90 degrees pitch, thus a quaternion representation of the attitude could ease the computations.

Rotational velocity transformations

The Euler rate vector $\dot{\boldsymbol{\Theta}}_{nb}$ is found by a transformataion of the body fixed angular velocity vector as given in (2.8).

$$\dot{\boldsymbol{\Theta}}_{nb} = \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb})\boldsymbol{\omega}_{b/n}^b \tag{2.8}$$

where the transformation matrix $\boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb})$ is given as

$$\boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb}) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \tag{2.9}$$

2.2.3 Equations of Motion

A process plant model is a detailed mathematical model of all ROV dynamics, while the control plant model is a simplified model [26], [27]. The former is mostly used in simulations, while the latter is used in controller and observer design.

Process Plant Model

The process plant model for the ROV is given in (2.10) and (2.11).

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.10}$$

$$\underbrace{\boldsymbol{M}_{RB}\dot{\boldsymbol{\nu}} + \boldsymbol{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\textit{rigid-body terms}} + \underbrace{\boldsymbol{M}_A\dot{\boldsymbol{\nu}}_r + \boldsymbol{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \boldsymbol{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r}_{\textit{hydrodynamic terms}} + \boldsymbol{g}(\boldsymbol{\eta}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{ext} \tag{2.11}$$

where $\boldsymbol{M}_{RB} \in \mathbb{R}^{6\times6}$ is the rigid-body mass matrix in CO and $\boldsymbol{C}_{RB} \in \mathbb{R}^{6\times6}$ is the rigid-body Coriolis and centripetal matrix. $\boldsymbol{M}_A \in \mathbb{R}^{6\times6}$ is the added mass matrix, $\boldsymbol{C}_A(\boldsymbol{\nu}_r) \in \mathbb{R}^{6\times6}$ is the added mass Coriolis and centripetal matrix and $\boldsymbol{D}(\boldsymbol{\nu}_r) \in \mathbb{R}^{6\times6}$ is the damping matrix. $\boldsymbol{g}(\eta) \in \mathbb{R}^{6\times1}$ is the hydrostatic restoring force vector, $\boldsymbol{\tau} \in \mathbb{R}^{6\times1}$ is the propulsion force vector and $\boldsymbol{\tau}_{ext} \in \mathbb{R}^{6\times1}$ is the external forces vector, which includes environmental forces and umbilical forces. $\boldsymbol{\nu}_r \in \mathbb{R}^{6\times1}$ is the relative velocity vector and is given by (2.12)

$$\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c \tag{2.12}$$

where $\boldsymbol{\nu}_c = [u_c, v_c, w_c, 0, 0, 0]^T$ is the current velocity in the ROV body frame.

The transformation matrix $\boldsymbol{J}(\boldsymbol{\eta})$ for the generalised coordinates consists of both the translational transformation matrix and the rotational transformation matrix as given in (2.13).

$$\boldsymbol{J}(\boldsymbol{\eta}) = \begin{bmatrix} \boldsymbol{R}_b^n(\boldsymbol{\Theta}_{nb}) & 0_{3\times3} \\ 0_{3\times3} & \boldsymbol{T}_\Theta(\boldsymbol{\Theta}_{nb}) \end{bmatrix} \tag{2.13}$$

Control Plant Model

The control plant model is a simplified version of the process plant model and is also used for the design of the observers and controllers in the ROV control system at NTNU. The assumptions for the simplified model includes:

- Slow-mowing vehicle, neglecting the Coriolis and centripetal forces
- Constant or slowly-varying current velocities, thus including the forces from the ocean currents in the bias estimate, $\boldsymbol{b}$
- Small roll and pitch motion and a neutrally buoyant vehicle with the centre of buoyancy directly above the centre of gravity, thus linearising the restoring forces $\boldsymbol{G}$

The model is given in (2.14), (2.15) and (2.16).

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.14}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} + \boldsymbol{G}\boldsymbol{\eta} = \boldsymbol{\tau} + \boldsymbol{J}^T(\boldsymbol{\eta})\boldsymbol{b} \tag{2.15}$$

$$\dot{\boldsymbol{b}} = -\boldsymbol{T}_b^{-1}\boldsymbol{b} + \boldsymbol{w}_b \tag{2.16}$$

where $\boldsymbol{M} \in \mathbb{R}^{6\times6}$ is the mass matrix and $\boldsymbol{D} \in \mathbb{R}^{6\times6}$ is the linear damping matrix. $\boldsymbol{b} \in \mathbb{R}^{6\times1}$ is the bias accounting for the unmodelled dynamics and slowly-varying loads such as current. The bias is modelled as a first order Markov process, where $\boldsymbol{T}_b \in \mathbb{R}^{6\times6}$ is a diagonal matrix with positive bias time constants and $\boldsymbol{w}_b \in \mathbb{R}^{6\times1}$ is a zero mean Gaussian white noise.

2.3 Generalised Forces

Ocean current, umbilical, manipulator and the thrusters are the contributors to the generalised forces acting on the ROV. The manipulator and umbilical forces as well as the current forces are taken care of by the bias estimation in the control plant model, while for the process plant model the current forces are included in the relative velocity vector.

2.3.1 Ocean Current Forces

The ocean current forces for use in simulations with the process plant model are generated using a model of the current speed and direction relative to the $\{n\}$ frame and is given in (2.17).

$$\boldsymbol{v}_c^n = \boldsymbol{R}_{y,\alpha_c}^T \boldsymbol{R}_{z,-\beta_c} \begin{bmatrix} V_c \\ 0 \\ 0 \end{bmatrix} \tag{2.17}$$

where V_c is the current speed and α_c and β_c are the vertical and horizontal direction respectively. The rotation matrices are computed by (2.6) with α_c and $-\beta_c$ as arguments.

2.3.2 Propulsion Forces

ROVs are usually fully actuated or over-actuated vehicles, which means they can produce force in any DOF by their thrusters. The NTNU ROV SF-30k have both single side and double-side thrusters. The force produced by the thrusters is hard to measure, thus developing a good model of the thrusters is important in the control system. The control input is the revolution speed of the thrusters, thus a mapping from revolution to force is necessary. A basic model of the thrust from a single thruster is given in (2.18).

$$f = K_T(J)\rho D^4 n^2 \tag{2.18}$$

where f is the thrust, $K_T(J)$ is the thrust coefficient, ρ is the water density, D is the propeller diameter and n is the propeller revolution speed in revolutions per second. The thrust coefficient depends on the advance number J, defined in (2.19).

$$J = \frac{V_a}{nD} \tag{2.19}$$

where V_a is the inflow water velocity to the propeller. The total thruster force, $\boldsymbol{\tau}$, in p DOFs is

$$\boldsymbol{\tau} = \boldsymbol{T}\boldsymbol{f} \tag{2.20}$$

where $\boldsymbol{T} \in \mathbb{R}^{p \times r}$ is the configuration matrix, a function of the thruster position and angles relative to the ROV body frame. The thrust vector $\boldsymbol{\tau}$ is commanded by the control system, and has to be distributed and mapped to revolution speed for each thruster. Thus, equation (2.21) is rewritten as

$$\boldsymbol{\tau} = \boldsymbol{T}\boldsymbol{K}\boldsymbol{u} \tag{2.21}$$

where $\boldsymbol{K} \in \mathbb{R}^{r \times r}$ is a diagonal matrix with thruster coefficients, with r being the number of thrusters and $\boldsymbol{u} \in \mathbb{R}^r$ is the propeller revolution speed. The thrust allocation solves for $\boldsymbol{u}$ and then the individual revolution speed for each propeller is determined. As the configuration matrix $\boldsymbol{T}$ is not invertible, a common solution is the Moore-Penrose pseudo inverse as

$$\boldsymbol{T}^\dagger = \boldsymbol{T}^T \left(\boldsymbol{T}\boldsymbol{T}^T\right)^{-1} \tag{2.22}$$

$$\boldsymbol{u} = \boldsymbol{K}^{-1}\boldsymbol{T}^{\dagger}\boldsymbol{\tau} \tag{2.23}$$

2.4 Sensors

The ROV can be equipped with a wide range of sensors to increase the accuracy of its navigation. This section will describe the most important sensors often installed on the ROV for its navigation. The measurements from these sensors will later in this thesis be combined with the output from the visual navigation system to estimate the states of the ROV. The typical sensors for a ROV are presented below and in Figure 2.3

(a) *Transponder* **(b)** *Pressure gauge* **(c)** *DVL* **(d)** *IMU*

Figure 2.3: *Typical ROV sensors. Courtesy of Kongsberg Maritime (a,b,d) and Innova AS (c)*

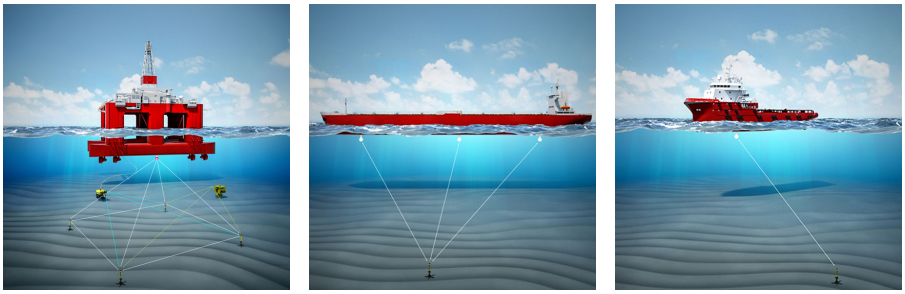
2.4.1 Sensor Description

Transponder

The transponder is mounted on the ROV and transmits and receives signals from an acoustic positioning system (APS). The signals from the APS determines the transponder position and outputs the x, y and z coordinates in the NED frame. Depending on the water depth, the update rate of the APS ranges from approximately 0.3-1.0 Hz for a typical ROV mission.

The acoustic frequency of the system is the governing factor for both the range and accuracy of the acoustic system. Higher frequencies can give good accuracy, but short waves are very vulnerable to attenuation in water, thus limiting the range of the system. Lower frequencies are not as vulnerable to attenuation and has a longer range, however with limited accuracy [28].

There are three main methods for underwater acoustic positioning, super short baseline (SSBL), short baseline (SBL) and long baseline (LBL), and these are presented in Figure 2.4. SSBL systems has a single multielement transducer mounted on the hull and uses



(a) *LBL* **(b)** *SBL* **(c)** *SSBL*

Figure 2.4: *Acoustic positioning systems, Long Baseline (LBL), Short Baseline (SBL), Super Short Baseline (SSBL). Courtesy of Kongsberg Maritime*

measurements of the range and angle to calculate the transponders position. The RV Gunnerus, where the NTNU ROVs are deployed has a Kongsberg HiPAP, a SSBL system with positioning accuracy of 0.2% of the range according to the manufacturer, with a maximum range of 4000 m. [3]

Short baseline systems typically have three transducers mounted on the hull and uses measurements of range and angle from all three to calculate the relative position of the transducer and the vessel. LBL systems are based on range measurements only. The transducer is mounted on the ROV, and its position is calculated relative to a calibrated array of transponders, typically installed on the seabed.

In deep waters the position error can be several meters, and it is thus very important to calibrate the APS for the sound velocity profile. The sound velocity will vary in the water column, dependent on conductivity, temperature and depth.

Pressure Gauge

The pressure gauge calculates the depth of the ROV given a measured pressure. The accuracy of a good pressure gauge is typically 0.01% of full scale with an update rate of approximately 8 Hz. This means the accuracy and update rate of the depth measurement from the pressure gauge is far superior to the depth measurement given by the transponder. [3]

DVL

The doppler velocity log (DVL) uses the shift in the echo from an acoustic signal sent from the ROV. Typically 4 transducer heads pings downward towards the seabed or the water column and uses the shift in the echo to calculate the velocity vector of the DVL relative to the sea floor or the water column. The DVL outputs a 3-DOF velocity measurement, u, v and w in the DVL frame. The DVL has an update rate similar to the pressure gauge, with maximum rate at 7 Hz [29].

Inertial Measurement Unit

The inertial measurement unit (IMU) mounted on an ROV typically has 3 accelerometers, 3 gyroscopes and 3 magnetometers, measuring the 3-DOF accelerations, turn rates and magnetic field components. Given recent advances the IMUs are today very accurate, small and inexpensive and thus well suited for smaller vehicles, such as the ROV. [3]

A typical IMU has an update rate ranging from 100-1000 Hz, which is much faster than the update rate of the APS or the DVL. Calibration of the IMU is very important to increase the accuracy. Despite the advances in the technology the gyroscope and accelerometer still suffer from drift and noise. The IMU readings thus have to be adjusted by a GPS reading for a surface vessel or an acoustic reading for subsea vessels.

2.4.2 Measurements

The sensors are mounted on the ROV in different positions and alignment. To combine the measurements and express them in a common reference frame, the sensor readings have to be transformed. Figure 2.5 below shows the frame of the ROV and the locations of the different sensors. All measurements are transformed to the CO of the ROV body frame or another origin of interest, and the equations describing these translations and transformations are given in the following. [30]

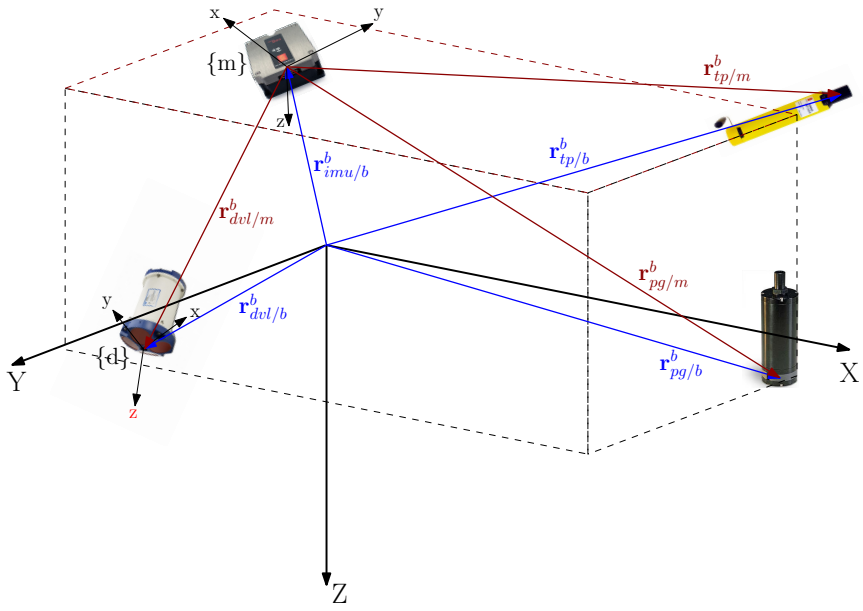


Figure 2.5: *ROV frame with sensors and their positions*

The position of the sensors relative to the CO of the ROV are given by the vectors $\boldsymbol{r}^b_{tp/b}$, $\boldsymbol{r}^b_{dvl/b}$, $\boldsymbol{r}^b_{imu/b}$ and $\boldsymbol{r}^b_{pg/b}$ for the transponder, DVL, IMU and pressure gauge respectively. Figure 2.5 shows both the arm of the sensors with respect to the body frame and the arm with respect to the IMU frame.

Sometimes it is useful to express the observer equations in the IMU frame, their position relative to the IMU is needed. These are noted $\boldsymbol{r}^m_{tp/m}$, $\boldsymbol{r}^m_{dvl/m}$ and $\boldsymbol{r}^m_{pg/m}$ for the transponder, DVL and pressure gauge respectively. The calculations of these arms relative to the IMU frame are given in (2.24)-(2.26).

$$\boldsymbol{r}^m_{tp/m} = \boldsymbol{R}^m_b(\boldsymbol{\Theta}_{bm})(\boldsymbol{r}^b_{tp/b} - \boldsymbol{r}^b_{imu/b}) \tag{2.24}$$
$$\boldsymbol{r}^m_{dvl/m} = \boldsymbol{R}^m_b(\boldsymbol{\Theta}_{bm})(\boldsymbol{r}^b_{dvl/b} - \boldsymbol{r}^b_{imu/b}) \tag{2.25}$$
$$\boldsymbol{r}^m_{pg/m} = \boldsymbol{R}^m_b(\boldsymbol{\Theta}_{bm})(\boldsymbol{r}^b_{pg/b} - \boldsymbol{r}^b_{imu/b}) \tag{2.26}$$

IMU

The measurement equations for the IMU accelerometer, gyro and magnetometer are given in (2.27), (2.28) and (2.29). The equations are given in IMU frame $\{m\}$, as explained above.

$$\begin{aligned}\boldsymbol{a}^m_{imu} = \dot{\boldsymbol{v}}^m_{m/e} + \boldsymbol{\omega}^m_{m/i} \times \boldsymbol{v}^m_{m/e} + \boldsymbol{R}^m_n(\boldsymbol{\omega}^n_{e/i} + \boldsymbol{\omega}^n_{n/e}) \times \boldsymbol{v}^m_{m/e} - \boldsymbol{R}^m_n \boldsymbol{g}^n_l \\ + \boldsymbol{b}^m_{acc} + \boldsymbol{w}^m_{acc}\end{aligned} \tag{2.27}$$
$$\boldsymbol{\omega}^m_{imu} = \boldsymbol{\omega}^m_{m/n} + \boldsymbol{R}^m_n(\boldsymbol{\omega}^n_{e/i} + \boldsymbol{\omega}^n_{n/e}) + \boldsymbol{b}^m_{gyro} + \boldsymbol{w}^m_{gyro} \tag{2.28}$$
$$\boldsymbol{m}^m_{imu} = \boldsymbol{R}^m_n \boldsymbol{R}^n_e \boldsymbol{m}^e + \boldsymbol{b}^m_{mag} + \boldsymbol{w}^m_{mag} \tag{2.29}$$

where $\boldsymbol{a}^m_{imu} \in \mathbb{R}^3$ is the measured acceleration vector, $\boldsymbol{b}^m_{acc}$ is the accelerometer bias vector and $\boldsymbol{w}^m_{acc}$ is the accelerometer noise vector. $\boldsymbol{\omega}^m_{imu} \in \mathbb{R}^3$ is the measured turn rate vector, $\boldsymbol{b}^m_{gyro}$ is the gyro bias vector and $\boldsymbol{w}^m_{gyro}$ is the gyro noise vector. $\boldsymbol{m}^m_{imu} \in \mathbb{R}^3$ is the measured magnetic field, $\boldsymbol{b}^m_{mag}$ is the magnetometer bias vector and $\boldsymbol{w}^m_{mag}$ is the magnetometer noise vector.

The IMU measurement can also be expressed in the vehicle frame, $\{b\}$. When simulating IMU measurements from the ROV motion, these equations are used. The equations are given in (2.30), (2.31) and (2.32)

$$\begin{aligned}\boldsymbol{a}^m_{imu} = \boldsymbol{R}^m_b(\dot{\boldsymbol{v}}^b_{b/e} + \boldsymbol{\omega}^b_{b/e} \times \boldsymbol{v}^b_{b/e} + \dot{\boldsymbol{\omega}}^b_{b/e} \times \boldsymbol{r}^b_{imu/b} + \boldsymbol{\omega}^b_{b/n} \times (\boldsymbol{\omega}^b_{b/n} \times \boldsymbol{r}^b_{imu/b}) \\ + \boldsymbol{R}^b_n(\boldsymbol{\omega}^n_{e/i} + \boldsymbol{\omega}^n_{n/e}) \times \boldsymbol{v}^b_{b/e} - \boldsymbol{R}^b_n \boldsymbol{g}^n_l) + \boldsymbol{b}^m_{acc} + \boldsymbol{w}^m_{acc}\end{aligned} \tag{2.30}$$
$$\boldsymbol{\omega}^m_{imu} = \boldsymbol{R}^m_b \left(\boldsymbol{\omega}^b_{b/e} + \boldsymbol{R}^b_n(\boldsymbol{\omega}^n_{e/i} + \boldsymbol{\omega}^n_{n/e})\right) + \boldsymbol{b}^m_{gyro} + \boldsymbol{w}^m_{gyro} \tag{2.31}$$
$$\boldsymbol{m}^m_{imu} = \boldsymbol{R}^m_b \boldsymbol{R}^b_n \boldsymbol{m}^n + \boldsymbol{b}^m_{mag} + \boldsymbol{w}^m_{mag} \tag{2.32}$$

DVL

The equation relating the DVL velocity vector to the ROV body frame is given in (2.33).

$$\boldsymbol{v}^d_{d/e} = \boldsymbol{R}^d_b(\boldsymbol{\Theta}_{bd}) \left(\boldsymbol{v}^b_{b/n} + \boldsymbol{\omega}^b_{b/n} \times \boldsymbol{r}^b_{dvl/b}\right) + \boldsymbol{w}^d_{dvl} \tag{2.33}$$

where $\boldsymbol{v}_{d/e}^{d} \in \mathbb{R}^3$ is the measured velocity vector and $\boldsymbol{w}_{dvl}^{d}$ is the DVL noise vector. For use in the control system, the equation must be solved for $\boldsymbol{v}_{b/n}^{b}$, the ROV velocity vector in body frame.

Pressure Gauge

The equations relating the pressure gauge depth measurement to the ROV body frame relative to the NED frame is given in (2.34) and (2.35).

$$p_{pg} = p_{atm} + \rho g z_{pg/n}^{n} + w_{pg} \tag{2.34}$$

$$z_{pg/n}^{n} = z_{b/n}^{n} + \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \boldsymbol{R}_{n}^{b}(\boldsymbol{\Theta}_{nb}) \boldsymbol{r}_{pg/b}^{b} \tag{2.35}$$

where p_{pg} is the measured pressure, p_{atm} is the atmospheric pressure at the surface, ρ is the water density, g is the acceleration of gravity, $z_{pg/n}^{n}$ is the depth of the pressure gauge and $\boldsymbol{w}_{pg}$ is the pressure gauge noise, while $z_{b/n}^{n}$ i the depth of the ROV.

Transponder

The equations relating the transponder position measurement to the ROV body frame relative to the NED frame is given in (2.36).

$$\boldsymbol{p}_{tp/n}^{n} = \boldsymbol{p}_{b/n}^{n} + \boldsymbol{R}_{b}^{n}(\boldsymbol{\Theta}_{nb}) \boldsymbol{r}_{tp/b}^{b} + \boldsymbol{w}_{tp}^{n} \tag{2.36}$$

where $\boldsymbol{p}_{tp/n}^{n} \in \mathbb{R}^3$ is the measured transponder position in NED frame, $\boldsymbol{p}_{b/n}^{n}$ is the ROV position relative to the NED frame and $\boldsymbol{w}_{tp}^{n}$ is the transponder noise vector.

2.5 ROV Control System

Development of the control system for the ROV at NTNU started as early as in 2004, when ROV Minerva was acquired. Through several master thesis and the PhD thesis of Dukan [3] the motion control system was taken from the early PC and Matlab implementation to the compactRIO and Labview implementation of today. The control system was restructured in 2012 to object-oriented programming in LabVIEW, and has since been an important tool for many disciplines at NTNU. The architecture of the control system makes it suitable as a test bed for PhD candidates and MSc students.

The modes and functions included in the motion control system range from station keeping, maneuvring from A to B, trajectory tracking, joystick closed-loop control, altitude control and more. Later an autonomy mode has also been implemented, having functions such as obsticle avoidance. The main control loop consists of signal processing, the observer, controller and thruster allocation. The structure of the motion control system with blocks and connections is seen in Figure 2.6.

In this thesis the test bed for control system implementation is the observer block. The output from the visual motion estimation serves as measurement input to the observer in the control system. The role of the observer is to take measurements in from different sensors and output smooth estimates of the ROV position and velocity to the controller block.

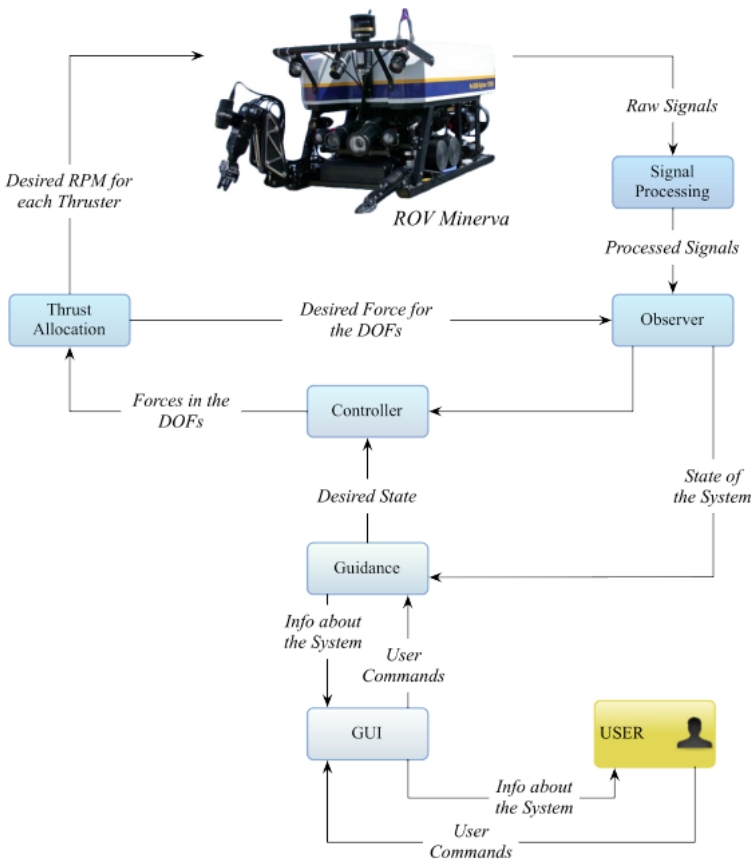


Figure 2.6: *ROV Minerva motion control system architecture. Courtesy of Mauro Candeloro*

The sensors have different update rates, noisy measurements, periods of no measurements. The observer, however, still has to provide the controller with estimates of the states of the system.

The default observer in the ROV motion control system is a Kalman Filter linearised around heading sectors. However, the attitude is estimated from the measurements from IMU prior to the observer in an explicit complimentary filter (ECF) [3]. The changes made to the observer for this thesis is the added measurements from the visual motion estimation. These come in as velocity measurements to the Kalman Filter and is supposed to increase the accuracy of the estimated states and to some degree attempt to minimise drifting.

2.5.1 ROV Kalman Filter

The Kalman filter used in the ROV control system and in this thesis is a sector Kalman Filter, where the observber equations are linearised about 36 heading angles, starting at 5° with 10° intervals. The filter is based on the equations from [26]. As the sensor measurements come in at different rates, especially the VME measurement, the measurement matrix $\boldsymbol{H}$ is adjusted accordingly. The Kalman filter in this thesis is 6 DOF Kalman filter for surge, sway, heave, roll, pitch and yaw.

The control plant model from (2.14), (2.15) and (2.16) is rewritten to the equations given in (2.37), (2.38) and (2.39).

$$\dot{\boldsymbol{\eta}} = \boldsymbol{J}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.37}$$

$$\boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{D}\boldsymbol{\nu} = \boldsymbol{\tau} + \boldsymbol{J}^T(\boldsymbol{\eta})\boldsymbol{b} + \boldsymbol{w}_m \tag{2.38}$$

$$\dot{\boldsymbol{b}} = -\boldsymbol{T}_b^{-1}\boldsymbol{b} + \boldsymbol{w}_b \tag{2.39}$$

where the restoring forces are neglected and a zero mean Gaussian white noise is added, $\boldsymbol{w}_m$. As before, the unmodelled dynamics are accounted for in the bias $\boldsymbol{b}$. Equations (2.37), (2.38) and (2.39) are expressed in state space form as given in (2.40) and (2.41).

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\boldsymbol{w} \tag{2.40}$$

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{v} \tag{2.41}$$

where

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{\eta} \\ \boldsymbol{\nu} \\ \boldsymbol{b} \end{bmatrix}, \quad \boldsymbol{u} = \boldsymbol{\tau}, \quad \boldsymbol{w} = \begin{bmatrix} 0 \\ \boldsymbol{w}_m \\ \boldsymbol{w}_b \end{bmatrix} \tag{2.42}$$

The number of inputs to the system is determined p, the number of states is n, while the number of outputs, or measurements, is m. Then for 6 DOFs, $\boldsymbol{y} \in \mathbb{R}^{m\times 1}$ is the measurement vector, $\boldsymbol{H} \in \mathbb{R}^{m\times n}$ is either zero or one, depending on the available measurements and $\boldsymbol{v} \in \mathbb{R}^{m\times 1}$ is a vector with sensor measurement noise. The matrices in the state-space equations (2.40) and (2.41) are given in (2.43).

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{J}(\boldsymbol{\eta}) & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{M}^{-1}\boldsymbol{D} & \boldsymbol{M}^{-1}\boldsymbol{J}^T(\boldsymbol{\eta}) \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{T}_b^{-1} \end{bmatrix}, \boldsymbol{B} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{M}^{-1} \\ \boldsymbol{0} \end{bmatrix}, \boldsymbol{E} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{2.43}$$

where the transformation matrix $\boldsymbol{J}(\boldsymbol{\eta})$ is linearised about the current heading, measured in the compass. A function to avoid chattering between the different sectors is also implemented. The state-space model in (2.40) and (2.41) is discretised as given in (2.44) and (2.45).

$$\boldsymbol{x}(k+1) = \boldsymbol{\Phi}(\boldsymbol{\eta}_i)\boldsymbol{x}(k) + \boldsymbol{\Delta}(\boldsymbol{\eta}_i)\boldsymbol{u}(k) + \boldsymbol{\Gamma}(\boldsymbol{\eta}_i)\boldsymbol{w}(k) \tag{2.44}$$

$$\boldsymbol{y}(k) = \boldsymbol{H}(k)\boldsymbol{x}(k) + \boldsymbol{v}(k) \tag{2.45}$$

where the matrices of the continuous model are discretised as

$$\boldsymbol{\Phi}(\boldsymbol{\eta}_i) = e^{\boldsymbol{A}(\boldsymbol{\eta}_i)h} \simeq \boldsymbol{I} + \boldsymbol{A}(\boldsymbol{\eta}_i)h \tag{2.46}$$

$$\boldsymbol{\Delta}(\boldsymbol{\eta}_i) = \int_0^h e^{\boldsymbol{A}(\boldsymbol{\eta}_i)s} ds \boldsymbol{B} \simeq \left(\boldsymbol{I}h + \frac{1}{2}\boldsymbol{A}(\boldsymbol{\eta}_i h^2\right)\boldsymbol{B} \tag{2.47}$$

$$\boldsymbol{\Gamma}(\boldsymbol{\eta}_i) = \int_0^h e^{\boldsymbol{A}(\boldsymbol{\eta}_i)s} ds \boldsymbol{E} \simeq \left(\boldsymbol{I}h + \frac{1}{2}\boldsymbol{A}(\boldsymbol{\eta}_i h^2\right)\boldsymbol{E} \tag{2.48}$$

k is the step number, and h is the time step in seconds for the discretisation. Based on these equations, and on [26], the following Kalman filter is proposed. The Kalman filter gain matrix $\boldsymbol{K}$ is defined in (2.49). $\boldsymbol{R}$ is the measurement covariance matrix and $\boldsymbol{Q}$ is the model covariance matrix. $\boldsymbol{P}$ is the state covariance matrix, a measure of the estimated accuracy of the state estimate.

$$\boldsymbol{K}(k) = \bar{\boldsymbol{P}}(k)\boldsymbol{H}^T(k)\left[\boldsymbol{H}(k)\bar{\boldsymbol{P}}(k)\boldsymbol{H}^T(k) + \boldsymbol{R}(k)\right]^{-1} \tag{2.49}$$

The corrector equations are given in (2.50) and (2.51).

$$\hat{\boldsymbol{x}}(k) = \bar{\boldsymbol{x}}(k) + \boldsymbol{K}(k)\left[\boldsymbol{y}(k) - \boldsymbol{H}(k)\bar{\boldsymbol{x}}(k)\right] \tag{2.50}$$

$$\hat{\boldsymbol{P}}(k) = \left[\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}(k)\right]\bar{\boldsymbol{P}}(k)\left[\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}(k)\right]^T + \boldsymbol{K}(k)\boldsymbol{R}(k)\boldsymbol{K}^T(k) \tag{2.51}$$

The predictor equations are given in (2.52) and (2.53)

$$\bar{\boldsymbol{x}}(k+1) = \boldsymbol{\Phi}(\boldsymbol{\eta}_i)\hat{\boldsymbol{x}} + \boldsymbol{\Delta}(\boldsymbol{\eta}_i)\boldsymbol{u}(k) \tag{2.52}$$

$$\bar{\boldsymbol{P}}(k+1) = \boldsymbol{\Phi}(\boldsymbol{\eta}_i)\hat{\boldsymbol{P}}(k)\boldsymbol{\Phi}^T(\boldsymbol{\eta}_i) + \boldsymbol{\Gamma}(\boldsymbol{\eta}_i)\boldsymbol{Q}(k)\boldsymbol{\Gamma}^T(\boldsymbol{\eta}_i) \tag{2.53}$$

The $\hat{\boldsymbol{x}}$ is the estimated state vector which is sent to the control block. The filter always runs at the same frequency, thus if a measurement is unavailable at time step k, the corresponding element in the measurement matrix $\boldsymbol{H}(k)$ is set to zero. This way the corrector only runs when there is an available measurement, otherwise, the predictor outputs the estimated states based on the mathematical model of the ROV.

There is no proof of stability or convergence when the system is linearised as it is in the sector Kalman Filter. The extended kalman filter could fix this problem, however, it is more meticulous to implement and more costly computationally than the linearised version. For a slowly-moving ROV the Coriolis force is going to be small, and the quadratic damping less important, thus the sector Kalman filter is implemented in the ROV control system at NTNU and in this thesis.

In order to accommodate the measurements from the visual motion estimation, the measurement matrix is expanded to $\boldsymbol{H} \in \mathbb{R}^{18\times 18}$ as the number of states remain the same, but the number of outputs is increased from 12 to 18. The measurement matrix is given in (2.54).

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{I}_{6\times 6} & \boldsymbol{0}_{6\times 6} & \boldsymbol{0}_{6\times 6} \\ \boldsymbol{0}_{6\times 6} & \boldsymbol{I}_{6\times 6} & \boldsymbol{0}_{6\times 6} \\ \boldsymbol{0}_{6\times 6} & \boldsymbol{I}_{6\times 6} & \boldsymbol{0}_{6\times 6} \end{bmatrix} \tag{2.54}$$

For $p = 6$ number of inputs, $n = 18$ states and $m = 18$ outputs, the measurement noise covariance matrix is defined as a diagonal matrix $\boldsymbol{R} \in \mathbb{R}^{18\times 18}$, while the matrices $\boldsymbol{A} \in \mathbb{R}^{18\times 18}$, $\boldsymbol{B} \in \mathbb{R}^{18\times 6}$, $\boldsymbol{E} \in \mathbb{R}^{18\times 18}$ are defined as in (2.43). The process noise covariance matrix $\boldsymbol{Q} \in \mathbb{R}^{12\times 12}$ is also unchanged from the control system from [3].

The visual motion estimation runs on a separate program than the control system, thus the Kalman Filter is implemented as a case structure. This means the Kalman Filter changes from its original version to the modified version depending on the existence of an input from the visual motion estimation. When the Kalman Filter runs without input from the visual motion estimation the measurement matrix is defined as:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{I}_{6\times 6} & \boldsymbol{0}_{6\times 6} & \boldsymbol{0}_{6\times 6} \\ \boldsymbol{0}_{6\times 6} & \boldsymbol{I}_{6\times 6} & \boldsymbol{0}_{6\times 6} \end{bmatrix} \tag{2.55}$$

The measurement noise covariance matrix $\boldsymbol{R}$ is expanded to include the measurements from the VME. The first 12 diagonals are given as before in the ROV control system, the last 6 for the VME measurements are given in (2.56)

$$\boldsymbol{R}(12-18, 12-18) = \mathit{diag}\left\{0.1, 0.1, 0.1, 100, 100, 10\right\} \tag{2.56}$$

Chapter 3

Computer vision

Computer vision deals with how digital images and video can be used to gain understanding of the surroundings. In the following the theory of computer vision will be used to gain understanding of motion, based on the visual input from a stereo camera. The field of computer vision has really gained ground the last decade as computational power has become more available to run the expensive algorithms in computer vision. See Section 1.1 and Section 1.4 for more about computer vision.

The method of estimating vehicle motion based on visual input is already used widely above the sea, especially as the development of the driver-less car is gaining traction. Also space probes, as the Mars Rover has included computer vision as one of its navigational sensors when exploring Mars. [13]

In this section we will first look into the model of the camera, then briefly discuss the different methods of extracting information from the camera, before going more deeply into the method used in this thesis and the different steps toward having an estimate of motion from one frame to the next.

3.1 Camera Model

3.1.1 Pin-hole Camera Model

The camera model applied to this problem is the pin-hole camera model. The pin-hole model centres around the optical axis drawn from the centre of the camera, through the centre of the image plane into the 3D coordinate frame, called the camera reference frame. The distance from the camera centre to the image plane is called the focal length, while the centre of the image plane $c = (c_x, c_y)$ is called the principle point. The pin-hole camera model for a stereo camera set is presented in Figure 3.1.

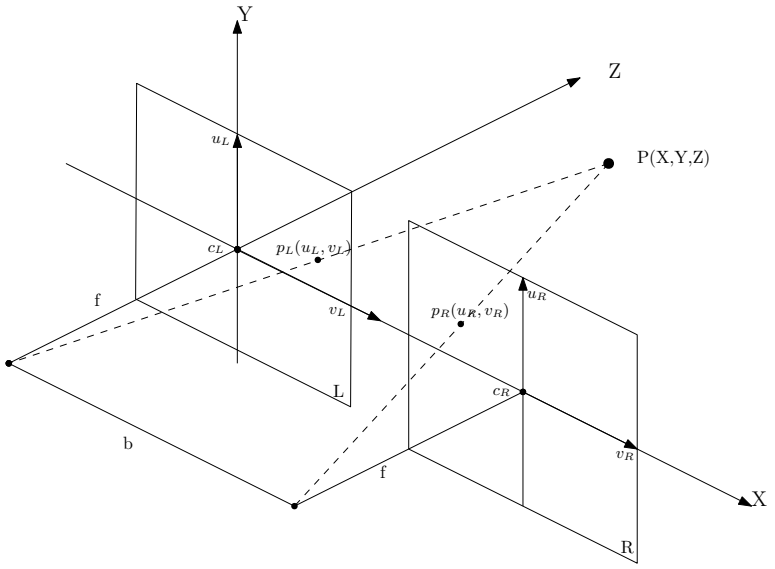


Figure 3.1: *The pin-hole camera model for a stereo pair of cameras*

We see the point $P = (X, Y, Z) \in \mathbb{R}^3$ relative to the camera reference frame, where the optical axis is the Z-axis. By taking the geometrical relations we derive that the point projected onto the image plane is

$$x = f\frac{X}{Z},\ y = f\frac{Y}{Z} \tag{3.1}$$

where x, y are the coordinates on the image plane. For only one camera, the depth of the image is difficult to determine. In order to calculate the depth of an object in the image, it is necessary to have a separate camera. Then by looking at the geometry between the camera centres, through the image planes to a point in 3D space the depth in the image can be calculated as:

$$Z = f\frac{b}{x_l - x_r} \tag{3.2}$$

where x_l, x_r is the x-coordinate of the point in the image plane of the left and right camera respectively and b is the baseline distance between the camera centres of the left and right camera. The origin of the camera reference frame is chosen to be in the camera centre of the left camera. By applying the equations of (3.1) and expressing the points in homogenous coordinates and relative to the principal point, the relation can be expressed in compact and more general form as:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z}\begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{bmatrix}\left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} - \begin{bmatrix} s \\ 0 \\ 0 \end{bmatrix}\right) \tag{3.3}$$

where (u, v) is the projected point to the image frame, f is the focal length, (c_u, c_v) is the principal point in the image frame and (X, Y, Z) is the point in the camera reference frame and s is the baseline. This model holds for both cameras, where $s = 0$ for the left camera and $s = b$ for the right camera.

In order to implement this simplified pin-hole camera model square pixels are assumed. The specifications from the camera manufacturer usually specifies this. There is also assumed no skew in the pixels. The distortion in a camera is usually not a problem on land, however, underwater, this could lead to some errors. The calibration method and how to account for distortion in the underwater images is described in the next section below.

3.1.2 Camera Calibration

Camera calibration is the process of finding the coefficients explained in the section above, the intrinsic camera parameters. The intrinsic parameters are mainly the focal length, the skew coefficient and the principal point coordinates, ideally in the centre of the image. The skew coefficient is as mentioned above assumed to be zero. The camera calibration is usually done by picturing a motive with direct lines and known geometry, such as a chessboard. By looking at the final geometry, the intrinsic parameters are determined. [31]

Most programming tools with computer vision ability can calibrate the camera, such as Matlab and OpenCV for C++. As mentioned, underwater imagery can be quite more challenging with respect to lighting and lens distortion especially. The refractive index of light travelling in water will change w.r.t. air due to particles. Thus a careful calibration of both the camera intrinsic parameters and distortion coefficients is necessary to get accurate results.

A method widely used, among others in the Agisoft software used for the NTNU ROV, is the Brown distortion model [32]. The distorted image coordinate (x, y) is related to the undistorted coordinates (x', y') by the equation given in (3.4) (3.5)

$$x' = x\left(1 + K_1 r^2 + K_2 r^4 + K_3 r^6\right) + P_2\left(r^2 + 2x^2\right) + 2P_1 xy \tag{3.4}$$

$$y' = y\left(1 + K_1 r^2 + K_2 r^4 + K_3 r^6\right) + P_1\left(r^2 + 2y^2\right) + 2P_2 xy \tag{3.5}$$

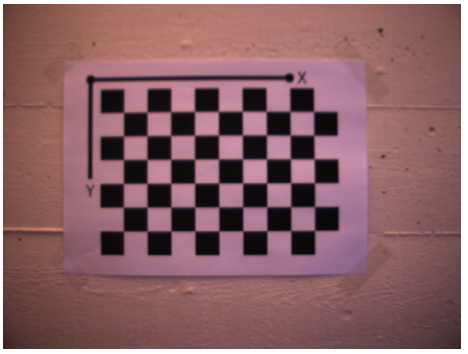
$$\text{where} \quad r = \sqrt{x^2 + y^2}$$

where K_1, K_2, K_3 are radial distortion coefficients, and P_1, P_2 are tangential distortion coefficients. These coefficients, together with the camera intrinsic parameters are all estimated by the camera calibration.

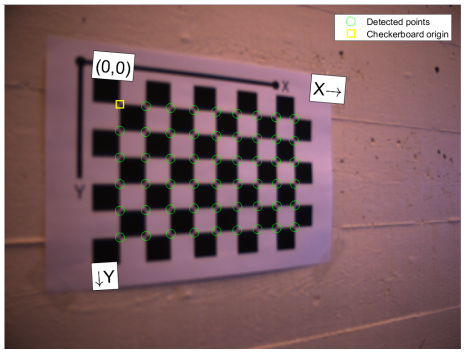
The calibration procedure is usually performed on a flat-surface, with known geometrical shapes. The most common flat-surface is the chessboard with fixed square sizes. The Matlab calibration toolbox is an easy and applicable tool for camera calibration. The procedure is shown in Figure 3.2 and will be explained here.

The Matlab toolbox for camera calibration takes in a series of images of a chessboard, as exemplified in Figure 3.2a. For the most optimal results a series of 10-15 images of the chessboard is recommended, taken at different angles and distances. The algorithm then detects all the corners of the chessboard as presented in Figure 3.2b. The algorithm

calculates the focal length, principal point and distortion coefficients based on the images and chessboard corners.



(a) *Chessboard used for camera calibration*



(b) *Corners and origin detected on chessboard*

Figure 3.2: *Camera calibration by chessboard using the Matlab camera calibration toolbox*

The results from the calibration in air with the stereo cameras listed in Appendix B is presented in Table 3.1.

Table 3.1: *Intrinsic parameters and distortion coefficients after camera calibration in air*

	Left Camera	**Right camera**
Focal length (f)	1275.8 pixels	1264.35 pixels
Principal point horizontal (c_u)	685.79 pixels	705.24 pixels
Principal point vertical (c_v)	509.81 pixels	528.75 pixels
Distortion coefficient (K_1)	-0.1383	0.1730
Distortion coefficient (K_2)	0.2929	0.4676
Distortion coefficient (K_3)	-0.4180	-0.3607

3.2 OpenCV Library

The OpenCV (Open Source Computer Vision Library) is a computer vision and machine learning library. The library is open sorce, meaning everyone can utilise the library for all sorts of applications. OpenCV is written natively in C++, but has interfaces with C++, Python, Matlab and Java, and supports the operating systems Windows, Linux, Android and Mac OS.

OpenCV has more than 2500 optimised algorithms that can be used for among others face recognition, object identification, human actions classification, tracking camera movements or moving objects, 3D model extraction, points cloud production from stereo

cameras, image stitching, etc. The library is used by several well-established companies, such as Google, Yahoo, Microsoft, Intel, Ibm and more. [33]

The use of the library span the entire world, being used for intrusion detection in Israel, detection of swimming pool drownings in Europe, checking runways for debris in Turkey and rapid face detection in Japan. [33] In this project the OpenCV algorithms used are mostly for feature detection and description in underwater images and matching features across frames. Thus, utilising only a small part of the vast opportunities the library present.

The point cloud production from stereo cameras can be an interesting next step for this type of project. Then the algorithm could gradually move over to a SLAM approach, where the ROV creates a map of the surroundings online, and can receive an absolute position reference when moving in these surroundings.

3.3 Feature Detection and Matching

Features in an image is very important in the study of computer vision. They are instrumental in motion estimation, 3D reconstruction, image mosaicking etc. An interest point in the image can be an edge, a corner, a local intensity maximum or minimum, or line endings. A key condition for an interest point is that it is distinguishable, meaning recognised in different lighting, different viewpoint or orientation. This is important in the field of object detection and tracking as the features must be recognised in different uncorrelated images. In Figure 3.3 the feature detection and matching across two images is presented, using the BRISK feature detector.

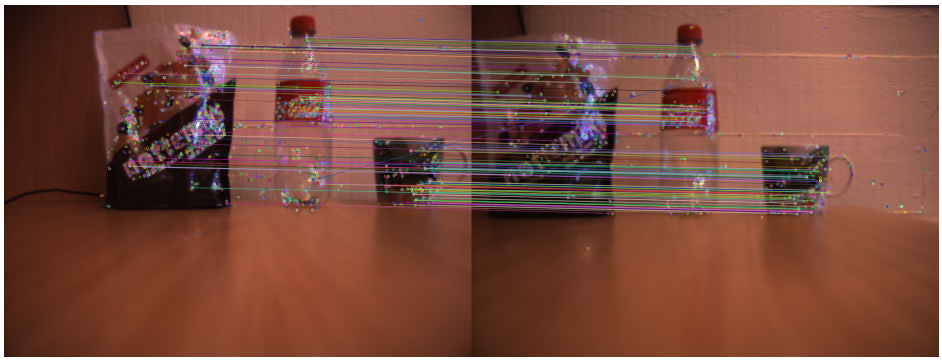


Figure 3.3: *Features are retrieved in two different images and matched from left image to right image*

The goal of extracting information about features in the image is to recognise the same feature in a different image. Thus, by computing the transformation matrix of a specific feature from frame to frame, the camera's motion can be estimated. A key parameter to match features in different images is the feature descriptor. The descriptors are usually a vector of N unique numbers describing the feature. Two features are matched when the norm of the corresponding descriptor vectors are sufficiently similar.

There are multiple feature detectors and descriptors developed for computer vision. The different methods have different advantages and are developed for different applications. The two most important criteria is speed and accuracy. The speed criteria is important for whether or not it is for online applications or processing applications. Regarding online implementation, one of the most common applications is navigation, especially vehicles on land, including the Mars rover [13]. In order to receive accurate and up-to-date information about the vehicles location, the feature detectors must be computationally fast [19], [20].

For post-processing, like constructing maps based on images, the speed is not as important. However, especially for map making, high accuracy is instrumental for a good result. Thus different feature detectors and descriptors have been developed with advantages and disadvantages for the different applications. In particular the SIFT and SURF methods proposed in the project thesis leading to this work, and the more computationally efficient binary descriptors, such as the ORB or BRISK methods.

3.3.1 Feature Detector SIFT/SURF

The scale-invariant feature transform (SIFT) method was introduced by [34] and is comprised of four basic steps. First the interest points in the image are detected, then their locations in the image are calculated. Then for each point the orientation is determined based on intensity gradients, before the descriptors are obtained by their gradient vectors.

In the first step the image is scanned for potential interest points. These interest points must satisfy the invariant criteria for both scale and orientation. The detection of features that are invariant to scale is done by searching for interest points across many scales using a scale-space function [34]. The scale space is obtained from the convolution of a variable-scale Gaussian filter $G(x, y, \sigma)$ with an input image $I(x, y)$.

$$L(x, y, \sigma) = G(x, y, \sigma) \circledast I(x, y), \quad G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{3.6}$$

To detect the location of a stable interest point, the difference of two nearby scales is computed $D(x, y, \sigma)$, separated by a multiplication factor k, $D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$. Then the local maximum and minimum of D is detected by comparing the sample point to its neighbours. To be selected as a detected interest point, the local maximum or minimum of the sample point must be bigger or smaller than all its neighbours [34].

If the interest point is a valid candidate, the point's location is found. The point is rejected if the contrast is poor or if it is badly located along an edge. This is either done by simply extracting the information from the original sampling point [35], or by fitting a function to the sample point and interpolating the location of the maximum or minimum [36]. In order for the feature to be invariant to rotation, the orientation of the feature is also assigned. The orientation is assigned based on the properties of the neighbouring pixels, such that even if the image is rotated, so is the neighbourhood of the feature.

Based on the information computed up to this point, a descriptor of the feature is formed. When extracting the descriptor, the gradient magnitude is computed at each sample point in a 4×4 region around the interest point. For each sample point the gradient histogram is computed in 8 orientation, thus creating a descriptor of $4 \times 4 \times 8 = 128$

elements for each feature. To avoid the effect of brightness and illumination change, the vector is normalised and reduced based on a given threshold. An example showing a 2×2 region of the interest point and the 8 orientations of the gradient histogram is presented in figure 3.4

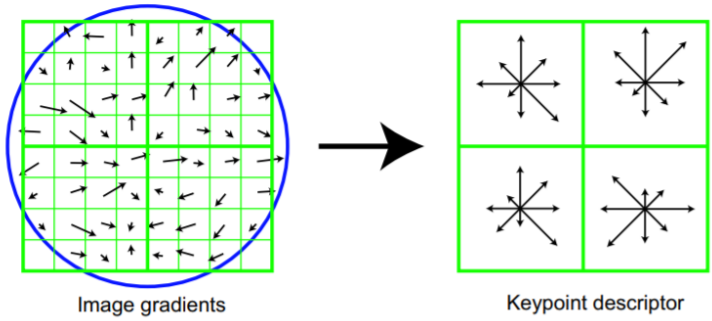


Figure 3.4: *Left shows the computation of gradient magnitude and orientation for an 8×8 set of samples weighted by a Gaussian window. Right shows the samples accumulated into orientation histograms in a 2×2 descriptor array. [34]*

The SURF method was introduced in [24] to reduce computation time and includes two new concepts to the SIFT method. Box filters are used to detect interest points and the vector containing the descriptors are reduced to half the size of the SIFT method [37]. The 128-vector long descriptor in the SIFT method is reduced to a 64-vector descriptor in the SURF method.

When extracting the descriptor, a square region is first centred around the interest point. This region is then divided into subsequent smaller 4×4 sub-regions. For each sub-region simple features are computed, and filters in horizontal and vertical direction are applied to the computed features. The filter responses are then summed over the sub-regions, giving a descriptor vector for all 4×4 sub-regions of length 64 [24]. The SURF method is faster than the SIFT, however still quite slow for online applications.

3.3.2 Feature Detector BRISK

The goal of the binary robust invariant scalable keypoint (BRISK) method is to reduce the computational cost, while still requiring high-quality descriptions of features in an image [38]. The method is mainly structured in the steps of detecting scale-space interest points and describing them, much like the SIFT and SURF methods. An advantage with the BRISK method of keypoint detection is that it allows for the combination of a BRISK detector with another feature descriptor and vice versa [38].

The BRISK methodology is inspired by the AGAST method in [39], which uses the FAST corner detection algorithm. In addition to searching for maximum and minimum in the image plane, the BRISK method also searches in the scale-space. The image is divided into pyramid layers, achieved by half-sampling the original image. The pyramid usually consists of 4 octaves and 4 intra-octaves. The sample point for detection must fulfil a

maximum condition compared to all of its 8 neighbouring pixels. This is done by a score, which is defined as the maximum threshold of an image point detected as a corner. The maximum condition will then have to be applied to all layers both above and below the active layer.

The feature description is composed as a binary string by assembling results of simple comparison of brightness. The characteristic direction of the interest points is also identified in order to allow features invariant to rotation. As mentioned before, invariance to scale and rotation are two very important abilities for a robust feature detection and description. BRISK uses a pattern for sampling the neighbourhood of a feature. N locations are placed around the feature spaced on circles, expanding from the center. To avoid aliasing when sampling the intensity of a point in the pattern, a Gaussian smoothing filter is applied. An example of the sampling pattern with $N = 60$ points is presented in figure 3.5.

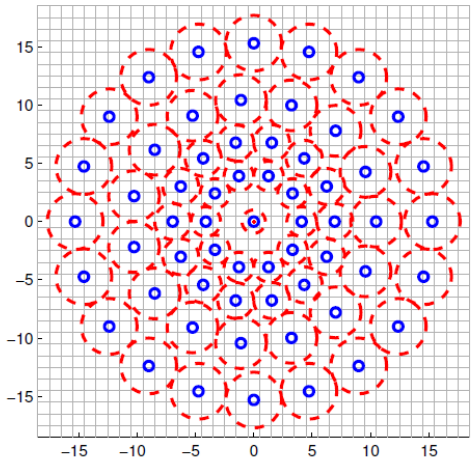


Figure 3.5: *BRISK sampling pattern with 60 points. Blue circles are sampling locations, red circles corresponds to standard deviation of Gaussian kernel applied to the sampling points, The scale here is 1. [38]*

The intensity values after the Gaussian smoothing are then used to estimate the local gradients. The algorithm looks at the intensity values for sampling-point pairs $(\boldsymbol{p}_i, \boldsymbol{p}_j)$, with their intensity values $I(\boldsymbol{p}_i, \sigma_i)$ and $I(\boldsymbol{p}_j, \sigma_j)$, where σ is the standard deviation of the Gaussian filter. The local gradient is then determined as given in (3.7)

$$\boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{p}_j) = (\boldsymbol{p}_j - \boldsymbol{p}_i)\frac{I(\boldsymbol{p}_j, \sigma_j) - I(\boldsymbol{p}_i, \sigma_i)}{||\boldsymbol{p}_j - \boldsymbol{p}_i||^2} \tag{3.7}$$

By iterating over the points, the overall characteristic direction of the feature is calculated. Only the long distance point pairs are used here, given a previously set threshold. The descriptor is then formed by rotating the sampling pattern around the feature and the descriptor is formed with each bit b satisfying (3.8)

$$b = \begin{cases} 1, & I(\boldsymbol{p}_j, \sigma_j) > I(\boldsymbol{p}_i, \sigma_i) \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

The matching of descriptors is a simple calculation of the Hamming distance between the two descriptors. The Hamming distance is essentially the number of positions in the descriptors that are not equal. Thus, a match is awarded if this number is below or above some user-specified threshold. In Figure 3.6 the BRISK feature detector is used to examplify the robustness of the algorithm to rotations.

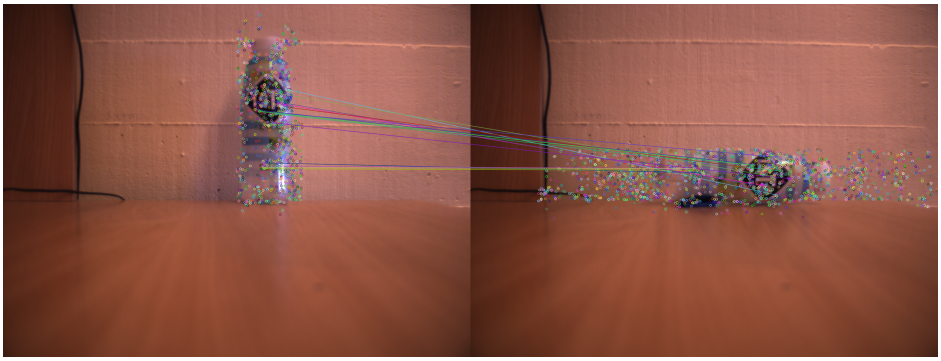


Figure 3.6: *Example of BRISK feature detection and matching algorithm and its robustness to rotation*

3.3.3 Feature Matching

The matching procedure of the VME is performed using the brute-force matcher. The brute-force means it will look through all feature descriptors in the second image for a match with every descriptor in the first image. For each match the distance between the descriptors is computed. As mentioned above, the Hamming distance is the one used when applying binary feature descriptors as with the BRISK feature detector and descriptor. Thus for all features the brute-force matcher will find a match and allocate it a Hamming distance. The Hamming distance is faster to compute than the euclidean distance as used in SIFT/SURF, making it another advantage w.r.t. computational effort [40].

Not all matches will be a correct match, and to sort out the incorrect matches the matching procedure also includes a nearest-neighbour search. This means that for each descriptor, the algorithm will find, in this case, two descriptors that are a match. The idea, that is explained in [34], that the closest neighbour is the actual matching feature, and second-closest is defined as the closest neighbour that is known not to be match. Thus, for each feature there will be two matches and two Hamming distances computed.

The next step is to sort out the matches that are incorrect. To do this, the ratio between the two matches is computed. If the closest neighbour is a correct match, the distance will be much lower than the distance of the second-closest neighbour. Thus the match will be accepted as a correct match. If the closest neighbour is an incorrect match, then by the

definition in [34], both distances will be quite similar. Thus by excluding all matches with a ratio above some threshold will eliminate most incorrect matches. This threshold is set to 0.65 for the implementation of the VME in this project.

3.4 Comparative Feature Detector Analysis

3.4.1 Feature Detection

In the project thesis leading up to this master thesis, the SURF feature detector was applied. However, as that part of the project was only applied to offline simulation, the computational speed of the solution was not emphasised. For online applications the BRISK feature detector is chosen based on its lower computational demand with comparative accuracy to SIFT/SURF as stated in [38]. The feature detection utilises the BRISK feature toolbox from the OpenCV library. For every iteration features are detected in 4 images, left and right from the current frame and from the previous frame. The BRISK feature detector class in the OpenCV library has functions for both detecting features and extracting the feature descriptor as explained in Section 3.3.

The feature detector returns a keypoint class of objects, representing the features in the image. The information returned include the location, orientation and the size of the feature's immediate neighborhood. The extractor returns the binary descriptor vector for each feature, the scope of which depends on the accuracy wanted. A comparison of the SURF and BRISK feature detection and description follows.

When deciding whether to use SURF or BRISK for the feature detection and description both methods were tried on the underwater images captured on the Stokkbergneset image set. The two algorithms were tested for two sets of 4 images in two consecutive stereo frames. The first set includes images with low activity, meaning the expected number of features is low. The second set includes images with high activity, thus expecting a large number of features.

The SURF algorithm was as expected quite slow, one run of the circle matching as described in Section 3.3 took on average 1.80 seconds for the first set, and 3.49 s for the second set. The SURF algorithm beats the BRISK on both sets of images, however comparatively much more on the first set of images. The BRISK algorithm was significantly faster, on average one run took 0.28 seconds for the first set and 0.74 s for the second set. The performance of the SURF and BRISK feature detection is presented and compared in Table 3.2.

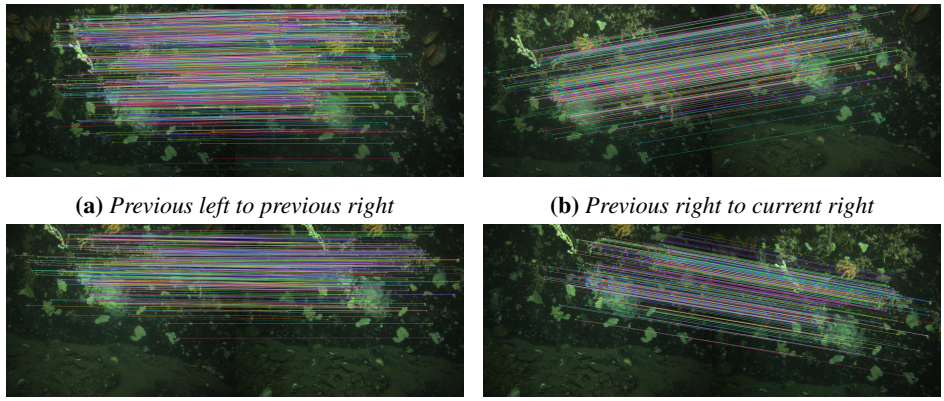
3.4.2 Feature Matching

The feature matching utilises the matching class from the OpenCV library. The procedure for both BRISK and SURF follows the procedure explained in Section 3.3. The difference between the two is that the BRISK feature matching computes the Hamming distance, as the descriptors are binary, while the SURF matching computes the euclidean distance, as the descriptors contain real numbers.

A problem with this approach is the classification of the matched features in the 4 images. To make use of the Gauss-Newton optimisation and the reprojection error of a

Table 3.2: *Feature detection results comparing SURF and BRISK algorithms.*

	SURF set 1	SURF set 2	BRISK set 1	BRISK set 2
Features previous left	4009	7518	980	6595
Features previous right	3080	7902	1108	9647
Features current right	2721	7157	886	6860
Features current left	3739	7017	902	4500
Average time	1.80 s	3.49 s	0.28 s	0.75 s



(a) *Previous left to previous right*

(b) *Previous right to current right*

(c) *Current right to current left*

(d) *Current left to previous left*

Figure 3.7: *Circle matching of features detected using BRISK and matched using the Hamming distance across 4 images, from a stereo camera in two consecutive frames*

feature, as will be discussed in Section 3.5, it is important to compare the same feature in four separate images. Thus, a circle matching is performed on the 4 images present at each iteration.

The features are detected in two consecutive stereo frames, meaning there will be 4 feature objects for each iteration, previous left and right frames and current left and right frames. First the feature descriptor of the previous left is matched with the feature descriptor in the previous right frame. The matched features is then extracted from the class of feature points in the previous right frame. These descriptors are then matched with the feature descriptors in the current right frame. These features are extracted and then matched with the feature descriptors in the current left frame.

To make sure the features matched through the circle to the current left are also present in the previous left frame, the circle is fully connected by matching the feature descriptors in the current left with the extracted descriptors from the previous left frame. This way it is ensured that all features or objects in the image are matched across all 4 images and can

be used for the Gauss-Newton optimisation algorithm. The circle matching procedure is presented in Figure 3.8.

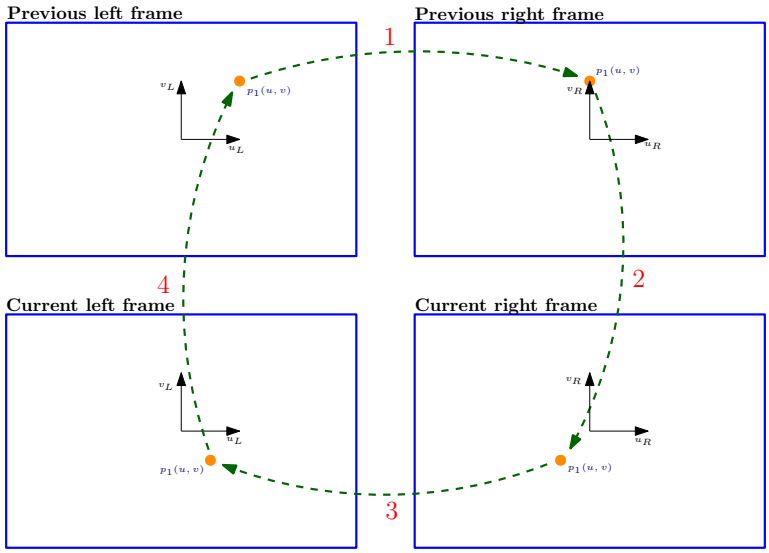


Figure 3.8: *Circle matching of features across two consecutive stereo frames*

A feature, in this case p_1 is matched across all four images and logged as a point for the subsequent algorithms. This way the features are matched in the temporal and spatial domain simultaneously. A comparison of the SURF and BRISK matching procedures follows.

Despite the comparatively low number of features, the BRISK and SURF algorithms computes a similar number of matched features through the whole circle for the first set. However, in the second set, the BRISK algorithm outperforms the SURF by almost doubling the number of matched features. The results from the feature matching is presented and compared for the two methods in Table 3.3. An example of the circle matching is also presented in Figure 3.7, where the BRISK algorithm is used on a set of images from the Stokkbergneset survey.

The accuracy of the two algorithms was studied by looking at the standard deviation on the matched features form image to image in both horizontal (x) and vertical (y) direction. The relative distance between the matched features for two images was computed and compared with all other matched features. The standard deviation was then computed based on the features matched across all 4 images as explained above. The result is presented in Figure 3.9, where the standard deviation is listed as distance in mm in the image frame.

The results show that the standard deviation of the first set is considerably larger than the second set, thus with more features come greater accuracy. The difference from the two algorithms is also evident at this point. With a low number of features the SURF algorithm is more accurate than the BRISK. However, with the large number of features in the second set of images, the standard deviation is quite similar, even in favour of the

Table 3.3: *Feature matching results comparing SURF and BRISK algorithms.*

	SURF set 1	SURF set 2	BRISK set 1	BRISK set 2
Previous left - previous right	796	1875	454	2383
Previous right - current right	286	683	235	972
Current right - previous right	156	442	184	711
Previous right - previous left	144	397	156	608
Average time	0.42 s	1.82 s	0.04 s	0.85 s

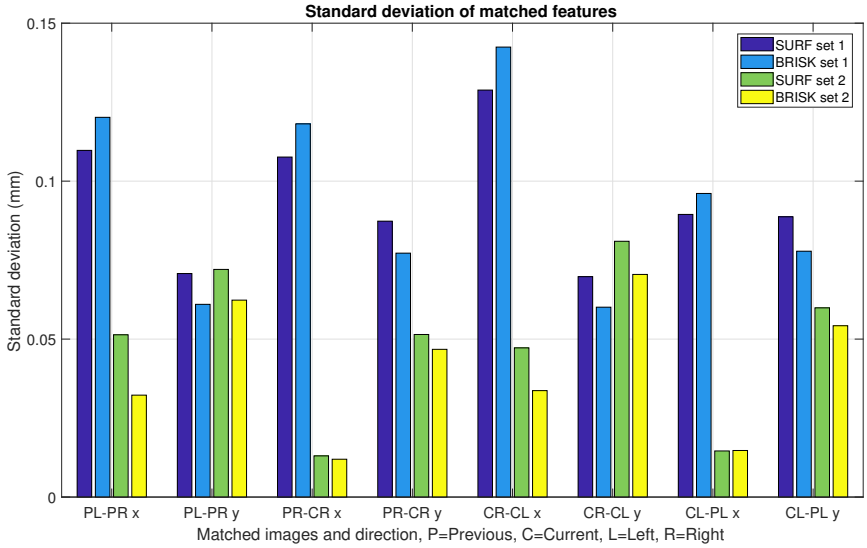


Figure 3.9: *Standard deviation of the relative distance to matched features during circle matching in both horizontal and vertical direction*

BRISK in some cases. This builds up under the conclusion made in Section 3.3 where it is said that the BRISK method has comparative performance with SURF with lower computational effort.

The standard deviation presented here will partially be dealt with by the motion estimation algorithm that includes a Gauss-Newton optimisation. As will be explained in Section 3.5 the algorithm selects the motion vector that best fit the most matched features. The algorithm will thus select the motion vector that fits the matched features around the median of the features which the standard deviation is based on.

3.5 Visual Motion Estimation

The visual motion estimation is implemented as a $.exe$ executable, programmed in C++ using the openCV computer vision library. The program outputs an estimation of the motion from one frame to the next. The visual motion estimation pipeline is presented in Figure 3.10.

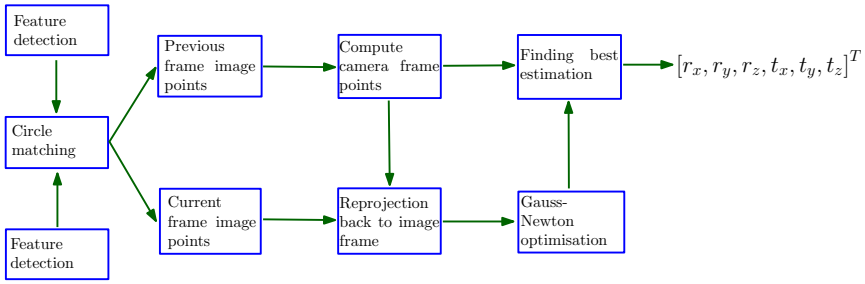


Figure 3.10: *Visual motion estimation pipeline from feature detection to motion estimation*

As presented, the pipeline consists of feature detection, feature circle matching, computing the camera frame coordinates, reprojection back into the image frame and a Gauss-Newton optimisation to solve for the camera frame motion $[\boldsymbol{r}, \boldsymbol{t}]^T$. In the following, the implementation of the algorithm will be explained, including some of the choices of method made during the development. The complete VME program is outlined in Algorithm 1.

Algorithm 1 Visual motion estimation

1: Get new current stereo images
2: Compute features and match in circle, see Algorithm 2
3: Calculate 3D location of features, see Algorithm 3
4: Gauss-Newton optimisation, see Algorithm 4
5: Export motion estimation
6: Send the current images to next iteration
7: Return to step 1

The first step of the algorithm is the feature detection and matching. The theory is discussed in Section 3.3, and the algorithm follows the steps as presented in Algorithm 2.

Algorithm 2 Feature detection and matching

```
Detect features using BRISK
Compute binary descriptors
Match features in previous left with previous right, 2 matches for each feature
for i = 1 : N, N number of matches do
    if r < 0.65, r is ratio of hamming distance of the two matches then
        Correct match
    end if
end for
Compute new descriptors for matched features in previous right
Match features in previous right with current right, 2 matches for each feature
for i = 1 : N, N number of matches do
    if r < 0.65, r is ratio of hamming distance of the two matches then
        Correct match
    end if
end for
Compute new descriptors for matched features in current right
Match features in current right with current left, 2 matches for each feature
for i = 1 : N, N number of matches do
    if r < 0.65, r is ratio of hamming distance of the two matches then
        Correct match
    end if
end for
Compute new descriptors for matched features in current left
Match features in current right with previous left, 2 matches for each feature
for i = 1 : N, N number of matches do
    if r < 0.65, r is ratio of hamming distance of the two matches then
        Correct match
    end if
end for
Convert keypoints to 2D image frame coordinates
return 2D image frame coordinates
```

3.5.1 Gauss-Newton Optimisation

For numerical optimisation of larger least-squares problems, the Gauss-Newton method is a good choice. In least-squares problems, the objective function has the form

$$f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j^2(x) \tag{3.9}$$

In this equation, each r_j is a smooth function determined a residual, and x is the parameter we want to find. For many unconstrained minimisation problems, the first and second derivative is necessary to find a solution. However, for a least-squares problem, the solution comes easier [41]. The individual residuals are formed into a residual vector

$\boldsymbol{r}(x)$, and we can thus rewrite the objective function $f(x) = \frac{1}{2}||\boldsymbol{r}(x)||_2^2$. The derivatives of f is then described through the Jacobian matrix given in (3.10).

$$\boldsymbol{J}(x) = \left[\frac{\partial r_j}{\partial x_i}\right], \quad j = 1, 2, ..., m, \quad i = 1, 2, ..., n \tag{3.10}$$

For this problem to have a solution a requirement is that $m > n$, which means there are more residuals than parameters to be optimised. Any solution for x^* must then satisfy

$$\nabla^2 f(x)x^* = -\nabla f(x)r_j \tag{3.11}$$

where we have

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x)\nabla r_j(x) = \boldsymbol{J}(x)^T r(x) \tag{3.12}$$

$$\begin{aligned}\nabla^2 f(x) &= \sum_{j=1}^{m} \nabla r_j(x)\nabla r_j(x)^T + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x) \\ &= \boldsymbol{J}(x)^T\boldsymbol{J}(x) + \sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x)\end{aligned} \tag{3.13}$$

Here we see that by computing the Jacobian matrix, we get easily both $\nabla f(x)$ and the first part of $\nabla^2 f(x)$. In addition, the first part of the second derivative is much more important than the second part. In fact, because of near-linearity close to the solution or very small residuals, the second part is usually neglected, meaning $\sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x) \approx 0$ [41]. Thus the solution is then found to satisfy the equation

$$\boldsymbol{J}(x)^T\boldsymbol{J}(x)x^* = -\boldsymbol{J}^T r(x) \tag{3.14}$$

3.5.2 Gauss-Newton For Visual Motion

As mentioned above the ego-motion estimation, meaning the estimation of the camera motion can be done in multiple ways. For underwater imagery, the feature-based method of estimating motion is the preferred, as quality of the images decreases when taken underwater. In [12] a method of minimising the reprojection error is proposed.

The motion of the camera is described in terms of both translation and rotation. As the ROV, the camera frame can move in 6 degrees of freedom, translation along the x, y and z-axis, t_x, t_y and t_z, and rotation around the three axes, r_x, r_y and r_z. In computer vision it is beneficial to talk about homogenous coordinates, meaning there is also one more dimension to the coordinate system. In a 2D image frame, the point will have three coordinates, while in a 3D camera frame, the point will have 4 coordinates. The extra coordinate will always have the value of one. With the extra coordinate, translation and rotation can fit into one 4×4 matrix, thus avoiding addition and subtraction.

The motion of the camera frame is described by the matrix $\boldsymbol{T}$, given in (3.15)

$$\boldsymbol{T}(\boldsymbol{r}, \boldsymbol{t}) = [\boldsymbol{R}(\boldsymbol{r}) \quad \boldsymbol{t}] \tag{3.15}$$

To solve for the transformation matrix $\boldsymbol{T}$, the relation between the image frame and camera frame coordinates can be utilised as in (3.3). Here, the transformation matrix is simply set equal to the homogenous identity matrix. To find $\boldsymbol{T}$ the image frame coordinates of the current frame can be compared with the camera frame coordinates of the previous frame. Thus, by solving for the transformation matrix the motion of the camara from one frame to the next is found. Equation (3.3) is expanded with $\boldsymbol{T}$ in (3.16)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{bmatrix} \boldsymbol{R}(\boldsymbol{r}) & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} - \begin{bmatrix} s \\ 0 \\ 0 \end{bmatrix} \right] \tag{3.16}$$

where $\boldsymbol{R}(\boldsymbol{r}) = \boldsymbol{R}_x(r_x)\boldsymbol{R}_y(r_y)\boldsymbol{R}_z(r_z)$ is the rotation matrix describing the rotation of the stereo camera set from previous to current frame, and $\boldsymbol{t} = [t_x, t_y, t_z]^T$ is the translation vector. The s is the baseline and is defined $s = 0$ for the left camera and $s = b$ for the right camera. The camera frame coordinates are given by (3.17)

$$Z = f\frac{b}{u_l - u_r}, \quad X = \frac{u_l - c_x}{f}Z, \quad Y = \frac{v_l - c_y}{f} * Z \tag{3.17}$$

To solve this equation for the 6 unknowns, three translations and three rotations, a Gauss-Newton optimisation can be applied. The goal of this optimisation is to minimise the sum of reprojection errors and optimise the rotation and translation vectors $\boldsymbol{r}, \boldsymbol{t}$, thus estimating the motion of the camera frame. Using least-squares optimization the sum is minimised according to the relation given in (3.18)

$$\sum_{i=1}^{N} ||\boldsymbol{x}_i^{(l)} - \pi^{(l)}(\boldsymbol{X}_i; \boldsymbol{r}, \boldsymbol{t})||^2 + ||\boldsymbol{x}_i^{(r)} - \pi^{(r)}(\boldsymbol{X}_i; \boldsymbol{r}, \boldsymbol{t})||^2 \tag{3.18}$$

Here $x_i^{(l)}$ and $x_i^{(r)}$ denote the locations of the features in the current image frame for the left and right cameras respectively. The $\pi^{(l)}$ and $\pi^{(r)}$ is the projection matrices of the 3D points in the previous camera frame from the left and right cameras respectively to the current imageframe.

3.5.3 Program Implementation

We let the previous frame be denoted $i - 1$ and the current frame denoted i. The matched features in frame $i - 1$ are used to compute the coordinates of the features in the camera reference frame, using equation (3.17). The points are then stored for the least-squares optimisation used to estimate the motion of the camera reference frame. The calculation of the camera frame coordinates are implemented according to the steps outlined in Algorithm 3.

Algorithm 3 Compute 3D camera frame coordinates

Image frame coordinates as input
for $i = 1 : N$, N number of points **do**
  Compute 3D points
  $Z = \frac{fb}{d}$
  $X = \frac{u - c_u}{f} Z$
  $Y = \frac{v - c_v}{f} Z$
  Correct for distortion
  $x = \frac{X}{Z}, \quad y = \frac{Y}{Z}$
  $x' = x(1 + K_1 r^2 + K_2 r^4 + K_3 r^6), \quad r = \sqrt{x^2 + y^2}$
  $y' = y(1 + K_1 r^2 + K_2 r^4 + K_3 r^6), \quad r = \sqrt{x^2 + y^2}$
  Compute distortion corrected 3D points
  $Z = \frac{fb}{d}$
  $X = x'Z$
  $Y = y'Z$
end for
return 3D camera frame coordintates

The motion of the camera is described by a homogeneous transformation matrix containing the rotations and translations of the camera. The rotation matrix is developed the same way the rotation matrix in (2.6), with angles defined as above. The order of multiplication for the camera frame computation is $\boldsymbol{R}(\boldsymbol{r}) = \boldsymbol{R}_x(r_x)\boldsymbol{R}_y(r_y)\boldsymbol{R}_z(r_z)$. The full transformation matrix, including the translation is defined as the homogeneous transformation matrix in (3.15).

To map the 3D camera frame coordinates back to the 2D image frame, homogeneous coordinates are implemented, as in (3.16) the camera frame coordinates are mapped to the image frame. This will be the framework for the Gauss-Newton optimisation to solve for the translational and rotational motion of the camera frame.

The 3D coordinates computed in frame $i - 1$ are reprojected into image coordinates in frame i. The expressions mapping the 3D coordinates to image frame is denoted $\pi^{(l)}$ and $\pi^{(r)}$ for the left and right camera respectively and are given in equation (3.18). The function to be minimised through the Gauss newton optimization is then given in (3.19).

$$\sum_{j=1}^{N} \left\| \begin{bmatrix} u_i^{(l)} \\ v_i^{(l)} \end{bmatrix} - \pi^{(l} \left(\begin{bmatrix} x_{i-1} \\ y_{i-1} \\ z_{i-1} \end{bmatrix} ; \boldsymbol{r}, \boldsymbol{t} \right) \right\|^2 + \left\| \begin{bmatrix} u_i^{(r)} \\ v_i^{(r)} \end{bmatrix} - \pi^{(r} \left(\begin{bmatrix} x_{i-1} \\ y_{i-1} \\ z_{i-1} \end{bmatrix} ; \boldsymbol{r}, \boldsymbol{t} \right) \right\|^2 \tag{3.19}$$

The equation is iteratively minimised with respect to optimising the variable $\boldsymbol{r}, \boldsymbol{t}$, which is the rotational and translational motion of the camera from frame $i - 1$ to frame i. The number of points in each iteration is chosen as $N = 3$ and the points are chosen at random. Hence, the optimisation takes in three random points that are matched across all four images and performs the iteration to minimise the function value.

The idea behind the reprojection and least-squares optimisation is presented graphically in Figure 3.11. The camera has here moved in one direction to make the example

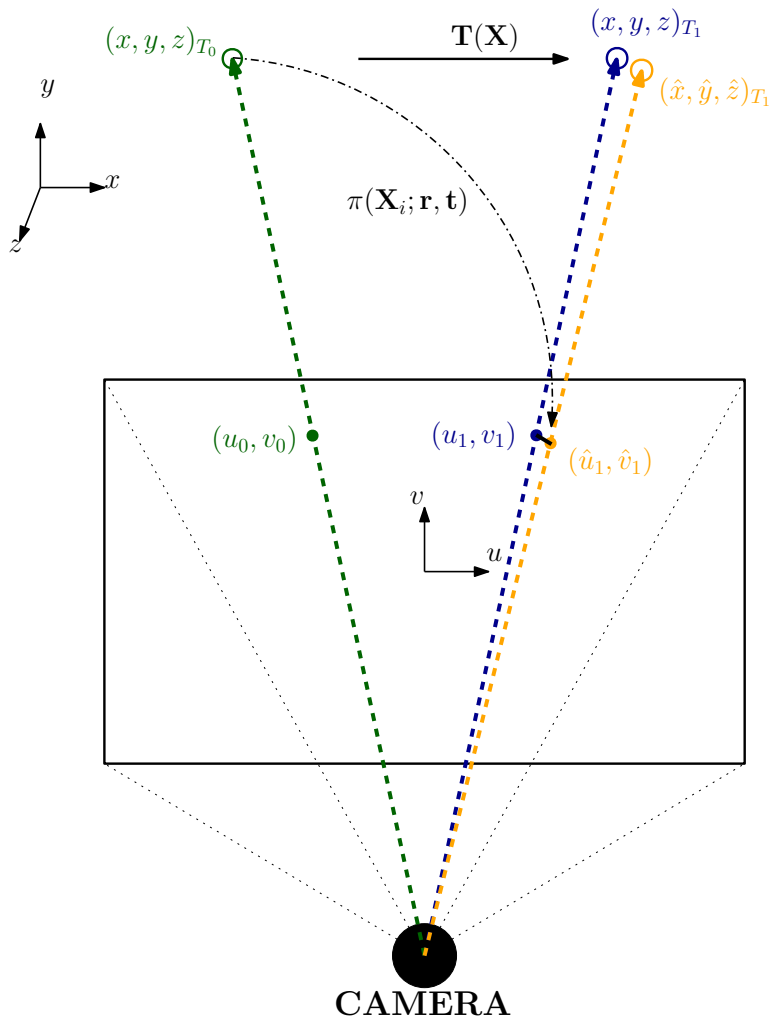


Figure 3.11: *Graphical representation of the least-squares problem with reprojection*

easy to understand. Correspondingly, the feature has moved in the opposite direction in the image plane and is detected and matched at two time instants, T_0 and T_1. The true transformation between the two time instants is not known, hence the variable describing the transformation have to be estimated.

The features have their image plane coordinates in (u_0, v_0) and (u_1, v_1) and their camera frame coordinates $(x, y, z)_{T_0}$ and $(x, y, z)_{T_1}$ respectively. By following the Figure and equation (3.19) the reprojection of the camera frame point to the image plane is recognised as $\pi(\boldsymbol{X}_i; \boldsymbol{r}, \boldsymbol{t})$. Thus, the Gauss-Newton minimises the error in the image plane $\epsilon = (u_1, v_1) - (\hat{u}_1, \hat{v}_1)$. The method used in this thesis, with the Gauss-Newton optimisa-

tion follows.

The optimisation algorithm takes in an initial guess of the variables, here named $\boldsymbol{\beta} = [r_x, r_y, r_z, t_x, t_y, t_z]^T$. The equation (3.19) is then calculated for the three randomly picked points and gives a 12×1 vector of residuals, classified $\boldsymbol{r}(\boldsymbol{\beta})$. The jacobian matrix must be computed for all 12 residuals as functions of the optimisation parameter β. Thus the Jacobian matrix is defined as:

$$\boldsymbol{J}(\beta) = \begin{bmatrix} \frac{\partial r_1}{\partial r_x} & \frac{\partial r_1}{\partial r_y} & \frac{\partial r_1}{\partial r_z} & \frac{\partial r_1}{\partial t_x} & \frac{\partial r_1}{\partial t_y} & \frac{\partial r_1}{\partial t_z} \\ \frac{\partial r_2}{\partial r_x} & \frac{\partial r_2}{\partial r_y} & \frac{\partial r_2}{\partial r_z} & \frac{\partial r_2}{\partial t_x} & \frac{\partial r_2}{\partial t_y} & \frac{\partial r_2}{\partial t_z} \\ ..... & & & & & \\ \frac{\partial r_{12}}{\partial r_x} & \frac{\partial r_{12}}{\partial r_y} & \frac{\partial r_{12}}{\partial r_z} & \frac{\partial r_{12}}{\partial t_x} & \frac{\partial r_{12}}{\partial t_y} & \frac{\partial r_{12}}{\partial t_z} \end{bmatrix} \tag{3.20}$$

The parameter iteration is then performed as presented in equation (3.21)

$$\boldsymbol{\beta}^j = \boldsymbol{\beta}^{j-i} - \left(\boldsymbol{J}^T\boldsymbol{J}\right)^{-1}\boldsymbol{J}^T\boldsymbol{r}(\boldsymbol{\beta}^{j-1}) \tag{3.21}$$

The average number of iterations to reach below the selected threshold of 10^{-10} is between 5 and 15 iterations. The optimisation parameters are initialised at 0, however, the number of iterations does on average never exceed 15. Thus, this part of the algorithm will run quite fast compared to the feature detection and matching, and will not be a delimiting factor on the overall computational effort of the system.

The inverse of the expression $\boldsymbol{A}^{-1} = (\boldsymbol{J}^T\boldsymbol{J})^{-1}$ is computed using the Gauss-Jordan method of matrix inversion. The identity matrix is set up next to the matrix as $\boldsymbol{A}_1 = [\boldsymbol{A}, \boldsymbol{I}]$. Then we find the sequence of elementary row operations that reduces $\boldsymbol{A}$ to the identity matrix. Then the same operations are performed on the identity matrix in order to obtain $\boldsymbol{A}^{-1}$. Then the matrix and its inverse will satisfy the equation $\boldsymbol{I} = \boldsymbol{A}\boldsymbol{I}\boldsymbol{A}^{-1}$

The optimisation is initialised 50 times for 3 random points, giving 50 solutions to the optimisation problem. As Gauss-Newton optimisation tool is powerful for these large problems, the average number of iterations before convergence is between 5 and 15. Thus, for each three points, the program runs 20 iterations, and by a set of conditions the best solution is chosen. For each of the 50 iterations the solution is applied to the reprojection from equation (3.18) on all matched features. The residuals are checked against a defined threshold of 10^{-4} and the solution that best fits the highest number of features are selected.

In addition the solution will sometimes emphasise the rotations over translations. As the ROV normally is relatively stable in pitch and roll, these DOFs are forced to be small in order to find a realistic estimation of the ROV motion. The algorithm for both the Gauss-Newton optimisation and the algorithms used for determining the best solution of the optimisation are implemented according to the steps outlined in Algorithm 4 and Algorithm 5.

Algorithm 4 Gauss-Newton optimisation

for $i = 1 : N$, N number of solutions **do**
  Pick three random features
  Initialise estimate (β)
  for $i = 1...M$, M iterations **do**
    Calculate residuals, $\boldsymbol{r}(\boldsymbol{\beta}^{i-1})$
    Calculate jacobian $\boldsymbol{J}$
    $\boldsymbol{\beta}^i = \boldsymbol{\beta}^{i-1} - (\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T\boldsymbol{r}(\boldsymbol{\beta}^{i-1})$
    if $(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T\boldsymbol{r}(\boldsymbol{\beta}^{i-1}) < \delta_{threshold}$ **then**
      break
    end if
  end for
  Calculate inliers, see Algorithm 5
  if Current inliers $>$ best inliers **then**
    Save estimation
    Save best inliers
  end if
end for

Algorithm 5 Caluclate inliers for the solution from Gauss-Newton optimisation

Motion solution, image frame coordinates and camera frame coordinates as input
$\text{count} = 0$
for $i = 1 : N$, N number of points **do**
  Reproject 3D coordinates to image frame
  $[u_r, v_r, 1]^T = \boldsymbol{C}[\boldsymbol{R}(\boldsymbol{r}), \boldsymbol{t}][X, Y, Z, 1]^T$, where $\boldsymbol{C}$ is camera intrinsic parameters
  if $(u_r - u)^2 + (v_r - v)^2 < t$, t is pre-defined threshold **then**
    $\text{count} = \text{count} + 1$
  end if
end for
return Number of inliers, count

The estimated motion can be ragged and prone to unrealistic spikes, thus a standard Kalman filter (KF) is placed on top of the optimisation. The states of the KF are the translational velocities and the angular velocities given by the translation and rotation vectors $\boldsymbol{t}, \boldsymbol{r}$ divided by the time between successive frames, $\boldsymbol{\nu} = [\boldsymbol{r}, \boldsymbol{t}]^T/\Delta t$. The 6 DOF accelerations are also included in the KF as states and are assumed constant.

This gives the state equation:

$$\begin{bmatrix}\boldsymbol{\nu}(k)\\ \boldsymbol{a}(k)\end{bmatrix} = \begin{bmatrix}\boldsymbol{I}_{6\times6} & \Delta t\boldsymbol{I}_{6\times6}\\ \boldsymbol{0}_{6\times6} & \boldsymbol{I}_{6\times6}\end{bmatrix}\begin{bmatrix}\boldsymbol{\nu}(k-1)\\ \boldsymbol{a}(k-1)\end{bmatrix} + \boldsymbol{v} \tag{3.22}$$

and the output equation:

$$\frac{1}{\Delta t}\begin{bmatrix}\boldsymbol{r}(k)\\ \boldsymbol{a}(t)\end{bmatrix} = \begin{bmatrix}\boldsymbol{I}_{6\times6} & \boldsymbol{0}_{6\times6}\end{bmatrix}\begin{bmatrix}\boldsymbol{r}(k)\\ \boldsymbol{t}(k)\end{bmatrix} + \boldsymbol{w} \tag{3.23}$$

In the equations above $\boldsymbol{\nu}$ is the velocity vector with translational and angular velocities, $\boldsymbol{I}_{6\times6}$ is the identity matrix, $\boldsymbol{a}$ is the acceleration vector, and $\boldsymbol{v}, \boldsymbol{w}$ are the process and measurement noise respectively. The Kalman Filter equations are adopted from the discrete Kalman Filter of [26] page 297.

The translational and angular velocities are now described in the camera frame for the left camera, so in order to compare the data with the data from the ROV, and to implement the sensor data to the ROV control system, the velocities must be rotated to the ROV body frame. This is accomplished using the relations between reference frames described in [26]. First the angular velocities are rotated to the body frame by $\boldsymbol{\Theta}_{bc} = [\phi, \theta, \psi]^T = [0, \pi/2, \pi/2]^T$, then the translational velocities are rotated as given in (3.24):

$$\begin{aligned} \boldsymbol{\omega}^b_{b/n} &= \boldsymbol{R}^b_c(\boldsymbol{\Theta}_{bc})^T \boldsymbol{\omega}^b_{c/b} \\ \boldsymbol{v}^b_{b/n} &= \boldsymbol{R}^b_c(\boldsymbol{\Theta}_{bc})^T \boldsymbol{v}^b_{c/b} - \boldsymbol{\omega}^b_{c/b} \times \boldsymbol{r}^b_{c/b} \end{aligned} \tag{3.24}$$

where the notation c refers to the camera coordinate frame and b refers to the body frame of the ROV. $\boldsymbol{R}^b_c(\boldsymbol{\Theta}_{bc})$ is the rotation matrix from the camera frame to the body frame, $\boldsymbol{\omega}^b_{c/b}$ and $\boldsymbol{\omega}^b_{b/n}$ are the angular velocities in the camera frame and ROV body frame respectively, $\boldsymbol{v}^b_{c/b}$ and $\boldsymbol{v}^b_{b/n}$ are the translational velocities in the camera frame and body frame respectively and $\boldsymbol{r}^b_{c/b}$ is the arm from the camera frame origin to the body frame origin of the ROV.

For the results below and for the simulation presented in Section 4 the cameras are located at the front of the ROV frame pointing along the x-axis of the ROV body frame. The location of the left camera from the ROV origin is given in (3.25).

$$\boldsymbol{r}^b_{c/b} = [1.21, -0.19, 0.05]^T \tag{3.25}$$

3.6 Motion Estimation Results

The vision-based ego-motion scheme is tested on an image set taken from a mission to Stokkbergneset in February 2017. An ROV is surveying the seabed with a stereo image set and the data from the mission include image sets and navigational data from the ROV. For the simulation an image set of 485×2 images with a time interval of approximately 2 seconds are used, giving a total simulation time of 770 seconds. The results presented in this section is simulated on this image set and the corresponding navigational data from the ROV control system. The cameras were not explicitly calibrated for this mission, thus the camera intrinsic parameters are estimated based on the images taken.

The image set from Stokkbergneset also included the navigational data from the ROV during the mission. The body-frame velocities ν from the navigational data was hence compared with the motion estimates obtained using the vision-based ego-motion scheme of this thesis. The results are presented in Figures 3.12 (u and p), 3.13 (v and q) and 3.14 (w and r). In these figures the motion estimate from the vision-based ego-motion is compared with the measured velocity from the ROV mission, which is based mainly on the input from the DVL and IMU sensors.

The results show general good performance of the visual motion estimation compared to the measured values from the ROV sensors. Especially for the translational velocities

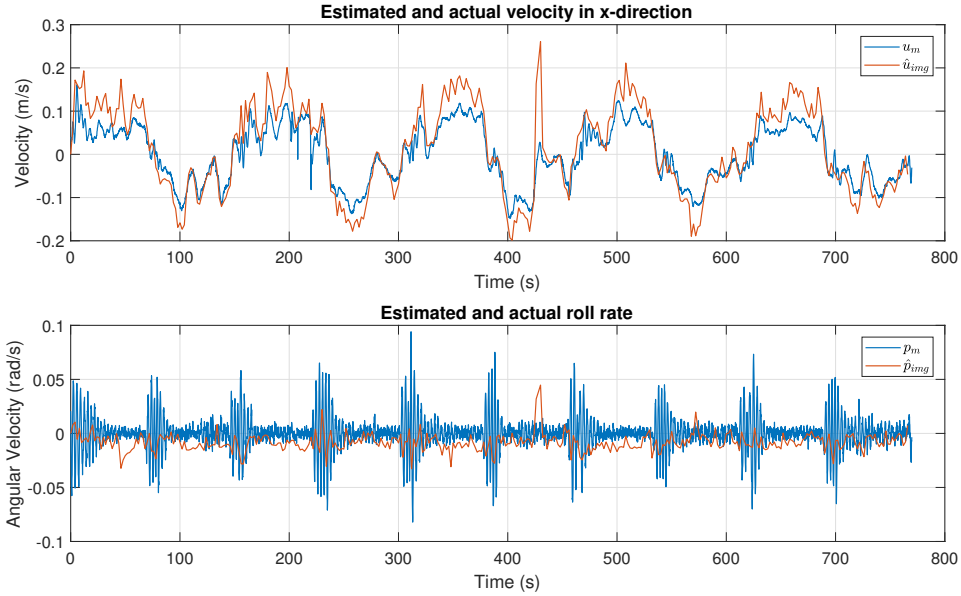


Figure 3.12: *Performance of the vision motion estimation system compared to the measured states from the ROV for velocity in x-direction and roll rate*

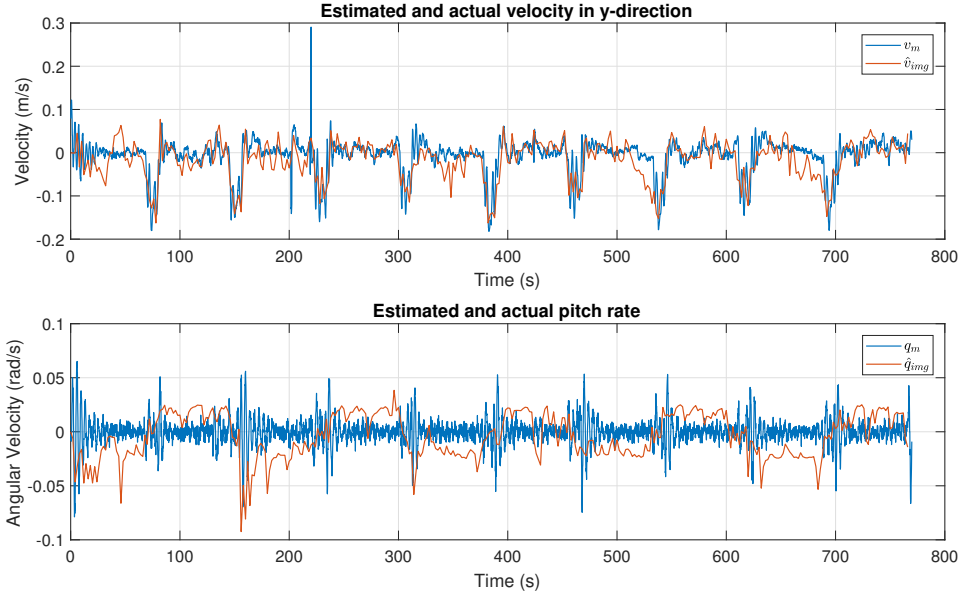


Figure 3.13: *Performance of the vision motion estimation system compared to the measured states from the ROV for velocity in y-direction and pitch rate*

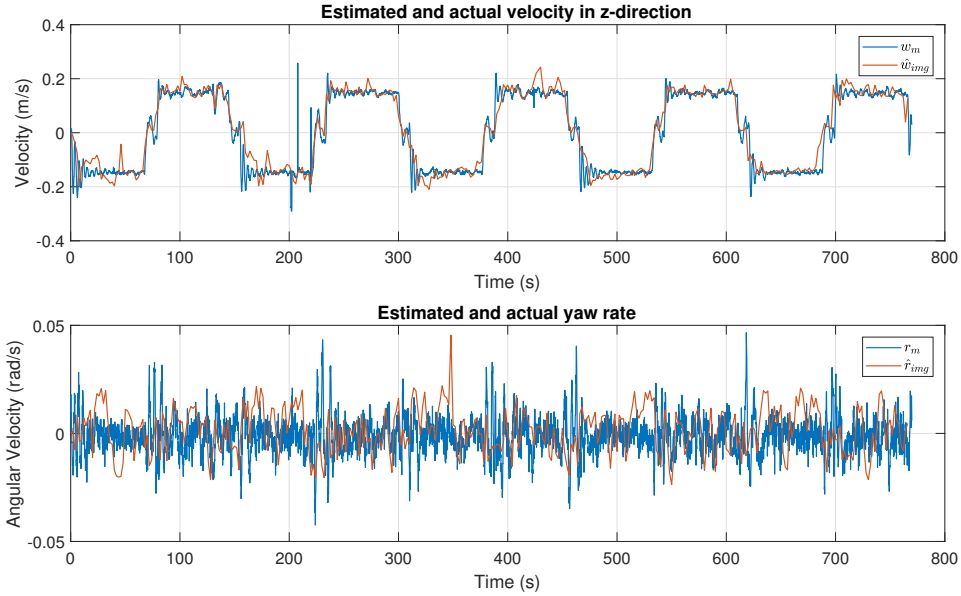


Figure 3.14: *Performance of the vision motion estimation system compared to the measured states from the ROV for velocity in z-direction and yaw rate*

the performance is very close to the sensor measurements. There are some larger oscillations in all three directions, however, there is a considerable improvement from the results in the project thesis. The process of selecting the best optimisation from the Gauss-Newton algorithm is improved, and the computational speed is reduced to a quarter.

The angular velocities deviate more from the ROV measurements. This is in line with the experience from the project thesis, the angular velocities are harder to estimate correctly than the translations. In this thesis the rotations from the Gauss-Newton optimisation have been played down, due to their inclination to emphasise rotational motion over translational, which usually is not the case for a ROV.

Chapter 4

Simulation

The simulation of the VME performance together with the Kalman Filter in the ROV control system is done based on the same image sets from the mission at Stokkbergneset mentioned in Section 3.6. The setup for simulation could not be accomplished through the ROV control system in LabVIEW, because the image data taken from previous missions could not be simulated together with the navigation data from the same mission. This was solved by exporting all data from both the ROV control system and the VME to Matlab.

The Kalman Filter was implemented as it is in the ROV control system in LabVIEW and the measurements from the VME is processed together with the raw measurements from the ROV. The results are then compared to the estimated states from the actual mission without the visual measurements and the actual position of the ROV. The actual position is not definite, as it is estimated based on the images taken by the software Agisoft after the mission. Based on this data and the map created of the environments, the ROV position is estimated.

The Kalman Filter develops over time, and gives better estimates after having converged. Thus, by instigating the Kalman Filter at a point in time during post processing will give a different result than the real time estimates. The real time Kalman Filter in this scenario has had time to converge and will thus be a better estimate. However, as the Kalman Filter converges in this simulated scenario, the estimates should converge to similar states.

4.1 Simulation Set-up

The simulation is based on image sets and navigational data from the mission to Stokkbergneset. The ROV in use is the SUB-fighter 30k with specifications described in Appendix A. The cameras in use are two Allied Vision Prosilica model GC1380 cameras. The cameras use a GigE Vision Gigabit Ethernet interface for high-speed data transmission over Ethernet networks and are set up with Schneider Kreuznac Cinegon lenses. The camera and lens specifications are described in Appendix B. The cameras are placed at the front of the ROV frame, and are pointing along the x-axes of the ROV body frame as Figure 4.1

shows. The location of the left camera, the origin of the camera frame, that is used in the rotation of the 6 DOF velocity vector is given in (3.25).

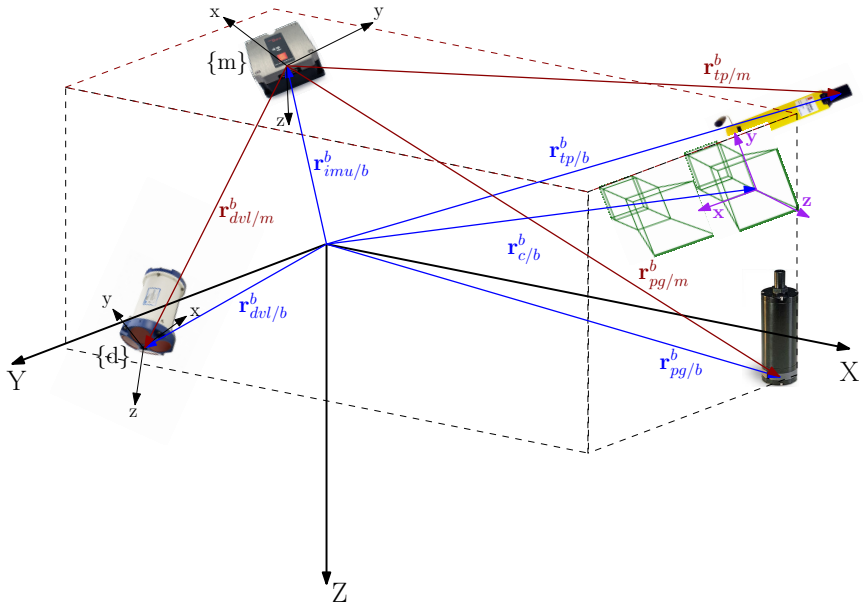


Figure 4.1: *ROV frame with sensors and their positions, including the stereo camera mounted at the front of the ROV with the camera frame in the colour purple*

The camera frame is coloured in purple to make it visible, where the depth of the camera frame is pointing in the ROV body frame x-direction, the x and y of the camera frame pointing in the y, and z-directions respectively of the ROV body frame.

The intrinsic parameters of the camera are estimated based on the images taken during the mission by the Agisoft software. Agisoft Photoscan is a software that performs photogrammetric processing of digital images. The software can also generate 3D spatial data as mentioned above. Thus a 3D map of the surroundings comprising the ”ground truth” position of the ROV is obtained based on the images taken during the mission. The calibration parameters estimated by the Agisoft software are listed in Table 4.1. [42]

Table 4.1: *Intrinsic parameters and distortion coefficients after underwater camera calibration using the Agisoft software*

	Left Camera	**Right camera**
Focal length (f)	1692.56 pixels	1701.64 pixels
Principal point horizontal (c_u)	670.21 pixels	675.04 pixels
Principal point vertical (c_v)	526.37 pixels	503.70 pixels
Distortion coefficient (K_1)	0.52	0.50
Distortion coefficient (K_2)	0.60	0.73
Distortion coefficient (K_3)	2.96	2.64

Comparing these parameters with the parameters given in Table 3.1, they are quite different. Especially the distortion coefficients differ a lot from air to water. Also the focal length changes as the cameras are submerged.

As mentioned, the mission to Stokkbergneset was meant to take pictures of a near vertical wall on the seafloor, thus the stereo cameras are pointed in the ROV body frame x-direction. The ROV moves in a lawnmower pattern up and down along the subsea wall, at a fixed distance (altitude) from the seabed. This will be visible in the plots presented in Section 4.2

4.2 Results

The simulations with the ROV observer was also performed on the image set from Stokkbergneset. The measurements corresponding to the image set was simulated with the visual motion estimation in a Kalman filter according to (2.37), (2.38), (2.39). The measurement matrix and the measurement covariance matrix were expanded according to Section 2.5 to accommodate the additional measurements, and the Kalman filter was simulated in the 4 DOFs of the control system, north, east, down and yaw for both position and velocity.

This section presents results from two scenarios, from two image sets taken during the mission to Stokkbergneset. The first scenario is simulated at a depth of 100 meters, while the second scenario is simulated at a depth of 300 meters. For most of the Figures the position plot and velocity plot in the same direction are grouped together with the velocity placed below the position. Error plots comparing the difference between the measured and estimated values are also included for all 4 DOFs. In addition a trajectory plot of the ROV in the North-East plane is included for both scenarios.

The subscripts in the legend for all plots are explained as follows: $m, sensors$ means raw measurements from the DVL, IMU, transponder and pressure gauge. ROV, nav means estimated values in the ROV control system online during the mission. ROV, IMG means simulated estimated values with the visual estimation of motion. "$true$" means the estimated ground truth from the image post-processing in Agisoft Photoscan. For the error plots, the subscript err, nav is the error between the raw measurements and the online estimation, while the err, IMG is the error between the raw measurements and the estimated values from the simulated Kalman Filter.

4.2.1 First Scenario

The results from the first scenario are presented in Figures 4.3 (N and u), 4.4 (E and v), 4.5 (D and w) and 4.6 (ψ and r). The North-East plot of the estimated trajectory of the ROV is presented in Figure 4.2

The North-East plot in Figure 4.2 with the visual motion estimation deviates from the output from the Kalman Filter during the mission. One reason for this is that the Kalman Filter on the ROV has already converged at this point. In the simulation, the Kalman Filter including the visual motion estimation has to start from the beginning each time. The time for it to converge explains the large deviation in the beginning.

In addition, the transponder measurements deviates from the estimated trajectory during the mission as well. This could be a symptom of a poorly calibrated acoustic po-

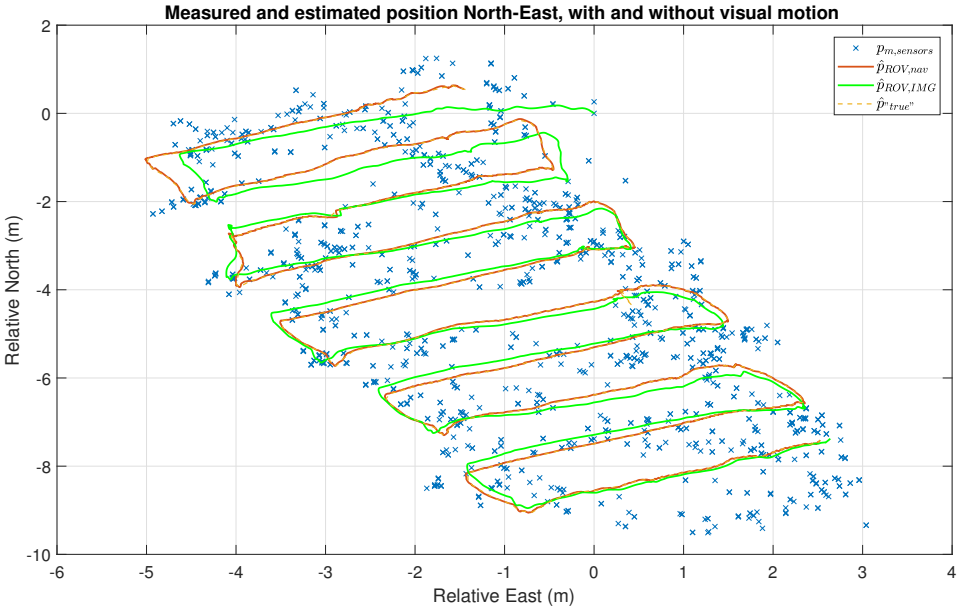


Figure 4.2: *Simulation results scenario 1: North-East plot of the estimated trajectory of the ROV with and without visual motion estimation*

sitioning system. This also becomes evident when looking at the estimated trajectory post-processed based on the images taken on the mission, which is very close to the online estimated trajectory. The ROV trajectory is simulated at positions relative to the first measured position, thus always starting in the point $(E, N) = (0, 0)$.

For the individual plots of the North and East directions in Figures 4.3 and 4.4 the simulation shows that by combining the sensor measurements from the ROV with the output from the visual motion estimation gives a similar result as the one obtained without the visual motion estimation. The difference is quite small and can be explained mostly by the problems stated above.

The results showing the performance in z-direction in Figure 4.5 is almost identical for measured, real estimated and simulated estimated. This is because the pressure gauge measuring the depth of the ROV is very accurate, thus the role of the visual motion estimation is played down in the measurement covariance matrix. This matrix will thus lead the Kalman Filter to rely more on the measurements from the pressure gauge sensor than other sensors.

The results for ROV yaw and yaw rate in Figure 4.6 shows a different picture than for the other DOFs. This is mostly due to the combination of Kalman Filter convergence and the role of the rotational velocity of the visual motion estimation as discussed in Section 3.5. The estimated and measured values for the yaw and yaw rate are identical, meaning the measured values have been passed directly through the Kamlam Filter.

In the Figures 4.7, 4.8, 4.9 and 4.10 the error plots are presented. Here the estimated states with and without the VME are compared to the measured states of the ROV. Also

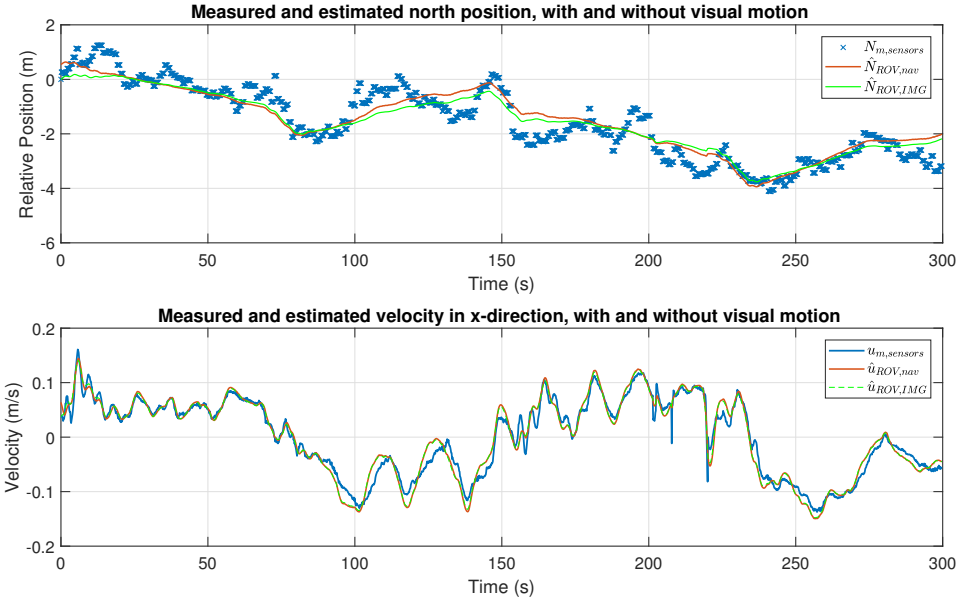


Figure 4.3: *Simulation results scenario 1: Performance of the Kalman filter with and without visual motion estimation for North position and velocity in x-direction*

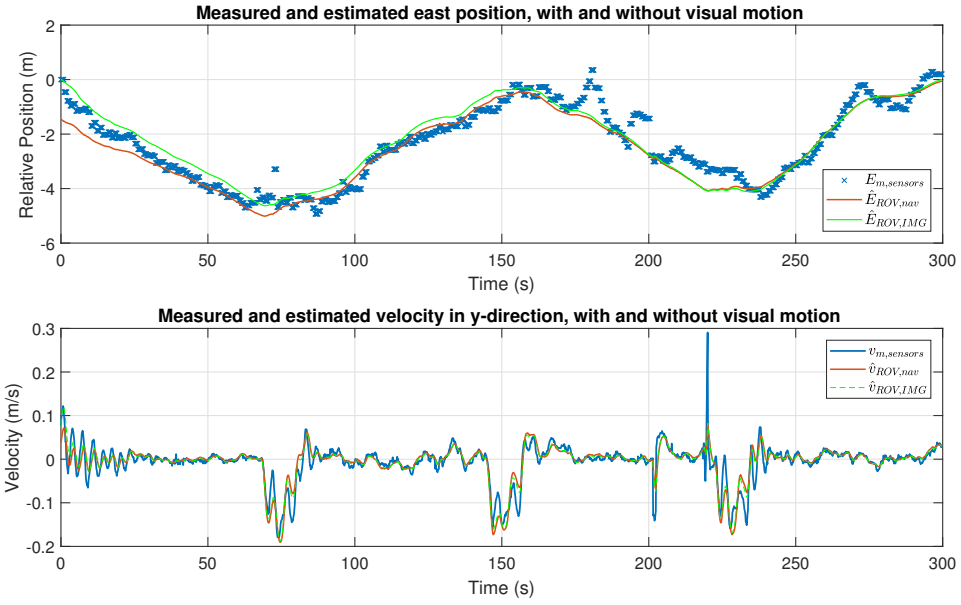


Figure 4.4: *Simulation results scenario 1: Performance of the Kalman filter with and without visual motion estimation for East position and velocity in y-direction*

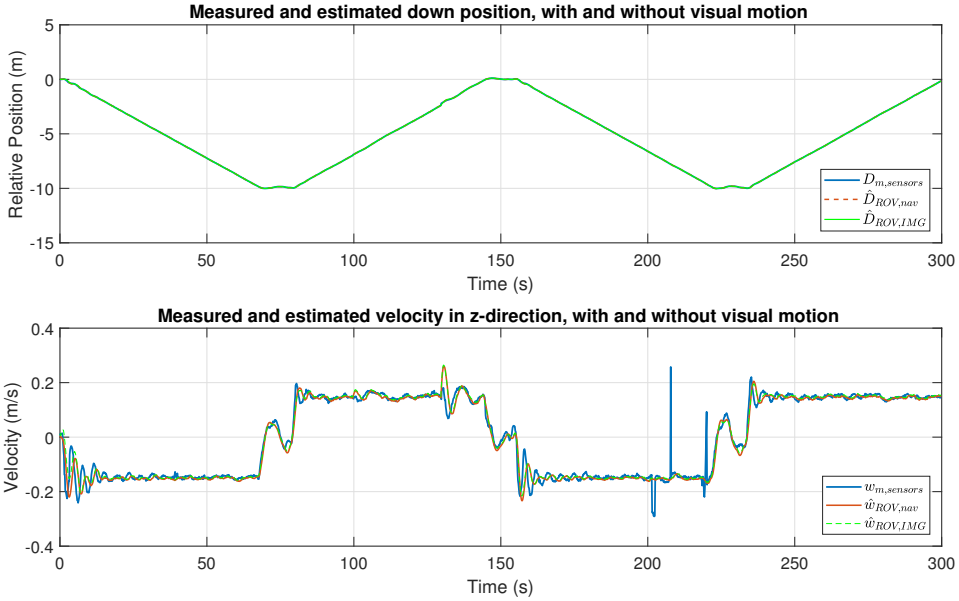


Figure 4.5: *Simulation results scenario 1: Performance of the Kalman filter with and without visual motion estimation for Down position and velocity in z-direction*

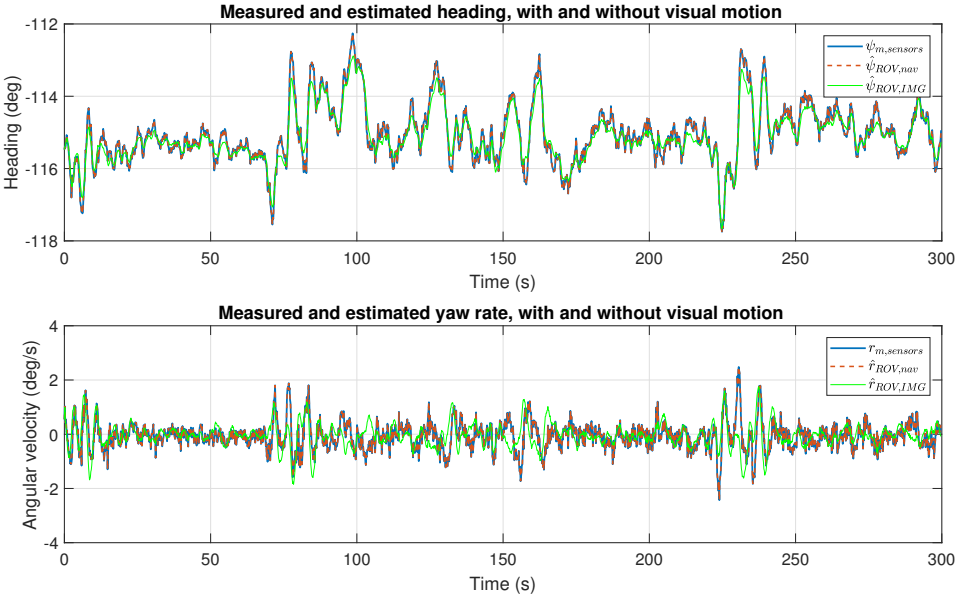


Figure 4.6: *Simulation results scenario 1: Performance of the Kalman filter with and without visual motion estimation for heading and yaw rate*

here we can see that the estimates with VME are slightly closer to the measured values than the estimates without VME.

Also worth mentioning is the error plot of the North and East position. The estimates with VME are closer to the measured values, but as mentioned above, this could have been a result of a slowly converging Kalman Filter for the simulations. As the measured position is mainly based on the transponder, there are some possibilities of errors, such as poor calibration as mentioned above.

The error plot of the heading and yaw rate in Figure 4.10, the error of the estimated value with VME is much higher than the error of the online estimate. As mentioned above, this could be because the measured heading is trusted in the online Kalman Filter, and sent through almost without filtering.

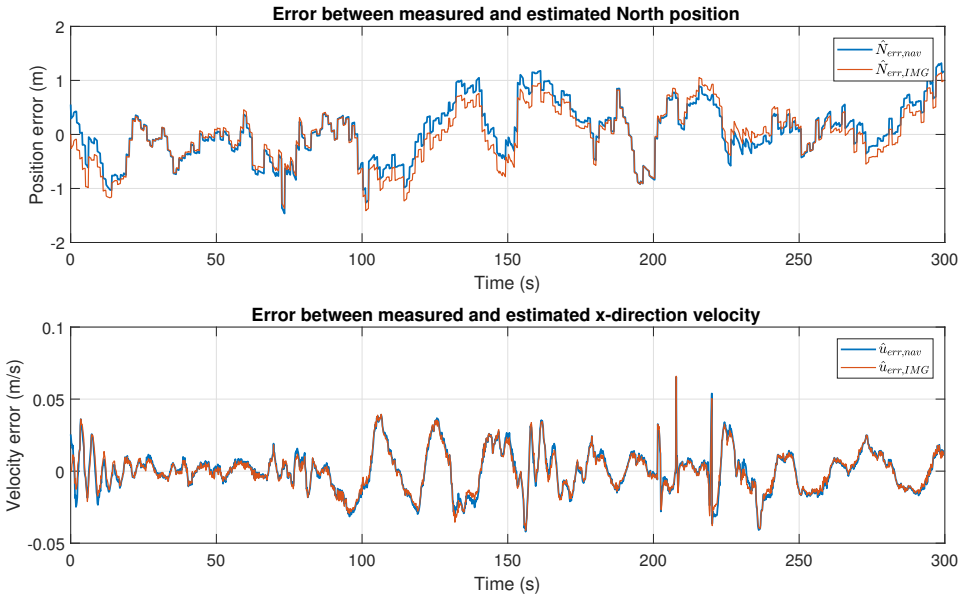


Figure 4.7: *Simulation results scenario 1: Error plot comparing the error between the two estimated values and the measured values for North position and velocity in x-direction*

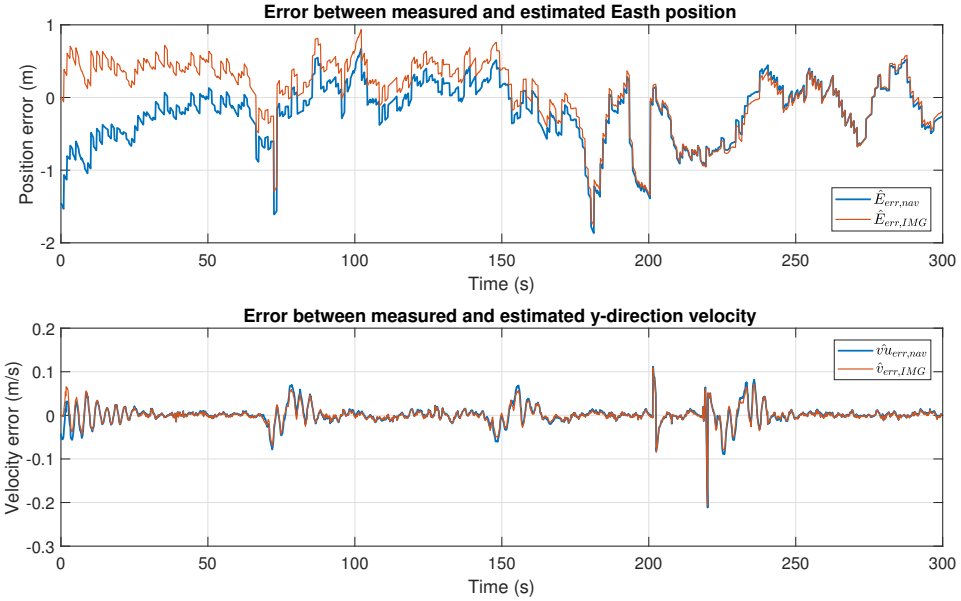


Figure 4.8: *Simulation results scenario 1: Error plot comparing the error between the two estimated values and the measured values for East position and velocity in y-direction*

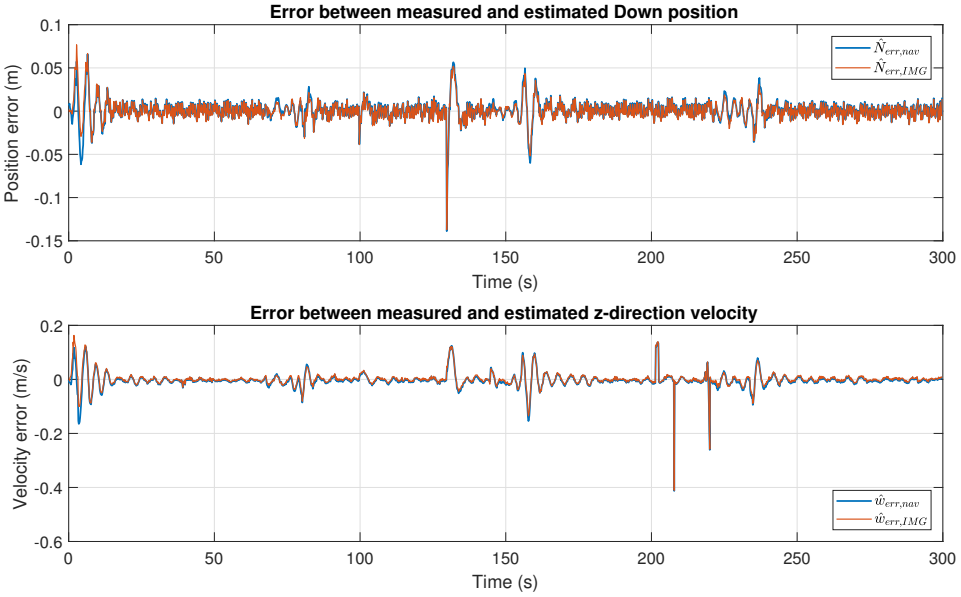


Figure 4.9: *Simulation results scenario 1: Error plot comparing the error between the two estimated values and the measured values for Down position and velocity in z-direction*

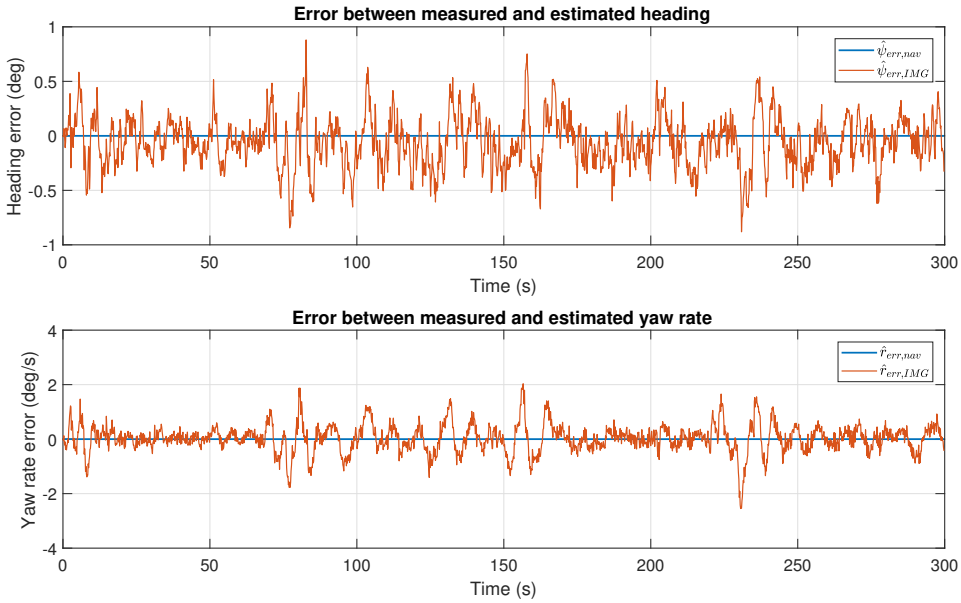


Figure 4.10: *Simulation results scenario 1: Error plot comparing the error between the two estimated values and the measured values for heading and yaw rate*

4.2.2 Second Scenario

The results from the second scenario are presented in Figures 4.12 (N and u), 4.13 (E and v), 4.14 (D and w) and 4.15 (ψ and r). The North-East plot of the estimated trajectory of the ROV is presented in Figure 4.11

The second scenario shows similar results as the first one. The North-East in Figure 4.11 has the same deviations between the measured and online estimated and between the online estimated and the states estimated with the VME. However, in this scenario the measurements are more scattered due to the increased depth of the ROV and longer travel time for the acoustic signals. However, the simulations are clearly able to filter out the wild points of the transponder measurement and estimate a trajectory similar to the one computed online in the control system.

For the individual plots of the North and East directions in Figures 4.12 and 4.13 this scenario is also similar to the first. The more scattered behavior of the transponder measurements are also clearly visible in these plots.

The results showing the performance in z-direction in Figure 4.14 is almost identical as in the first scenario. As in the first scenario this is because of the highly accurate pressure gauge sensor.

The results for ROV yaw and yaw rate in Figure 4.15 has the same tendency as in the first scenario. The online estimated heading and yaw rate are identical to the measured values, leading again to the assumption that these states are passed directly through the Kalman Filter.

As with the first set, the error is slightly smaller for the new estimates than the old

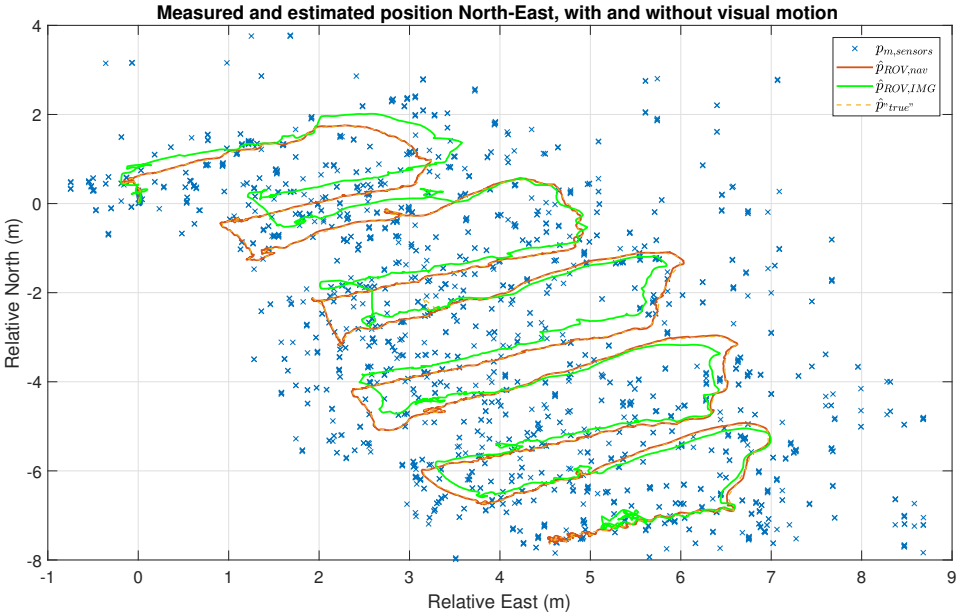


Figure 4.11: *Simulation results scenario 2: North-East plot of the estimated trajectory of the ROV with and without visual motion estimation*

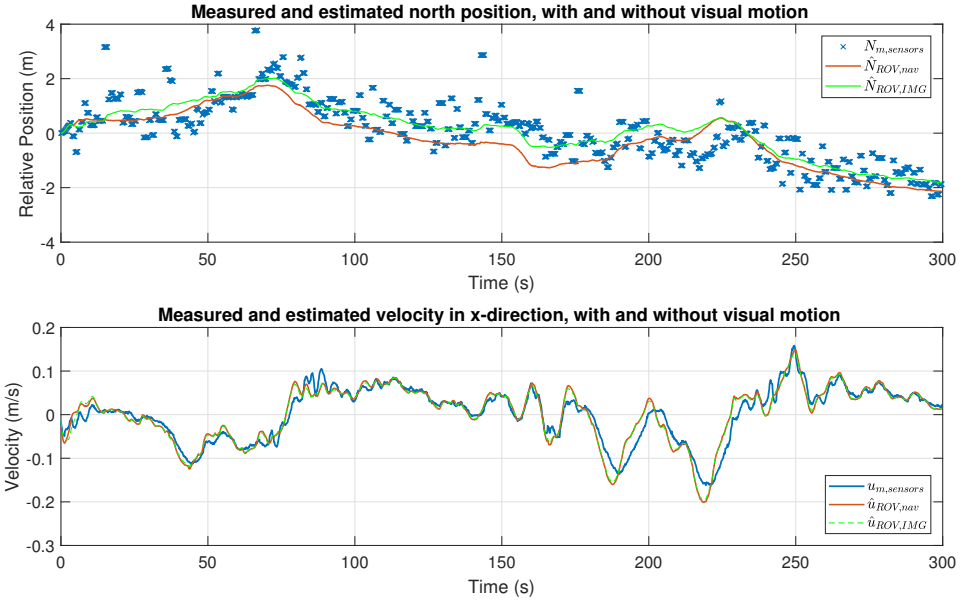


Figure 4.12: *Simulation results scenario 2: Performance of the Kalman filter with and without visual motion estimation for North position and velocity in x-direction*

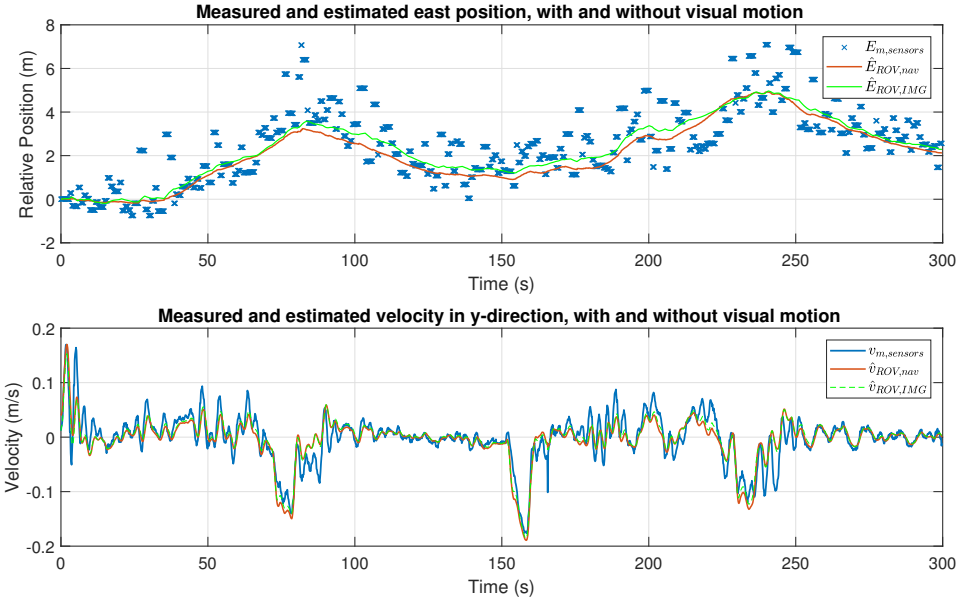


Figure 4.13: *Simulation results scenario 2: Performance of the Kalman filter with and without visual motion estimation for East position and velocity in y-direction*

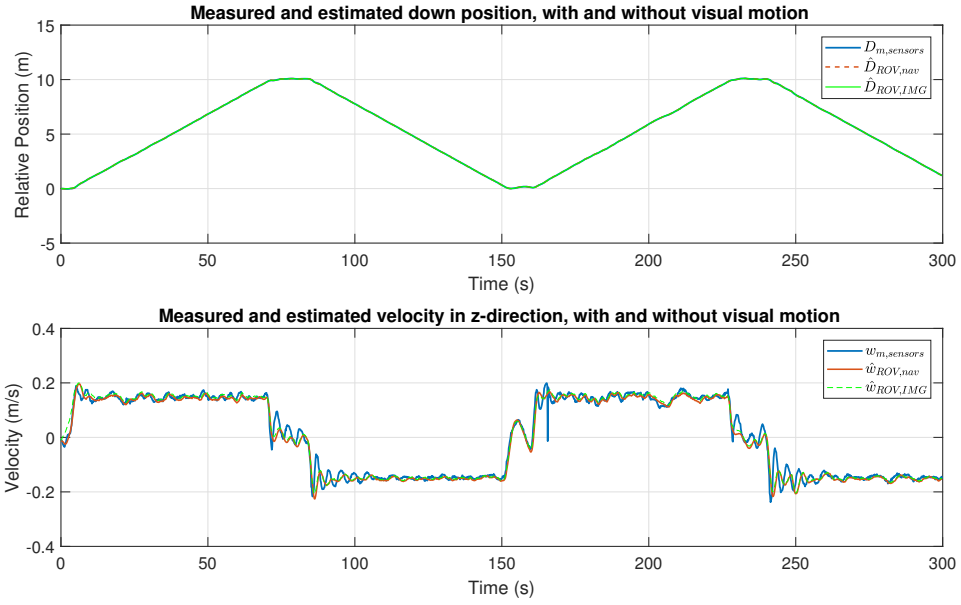


Figure 4.14: *Simulation results scenario 2: Performance of the Kalman filter with and without visual motion estimation for Down position and velocity in z-direction*

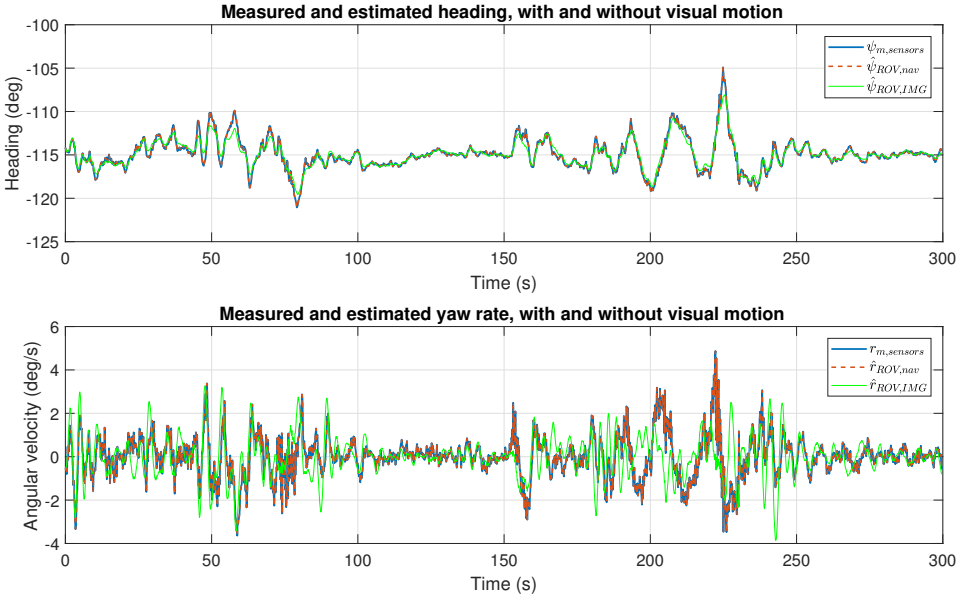


Figure 4.15: *Simulation results scenario 2: Performance of the Kalman filter with and without visual motion estimation for heading and yaw rate*

estimates. This is most notably true for the velocities, as these will be impacted the most from the VME output, being a 6 DOF velocity vector. The error plots are presented in Figures 4.16, 4.17, 4.18 and 4.19

The most notable difference in the error plots from the first to the second scenario is the fact that the error in the heading and yaw rate is doubled in the second compared to the first (Fig. 4.15). The increasing depth may be a large contributing factor to this.

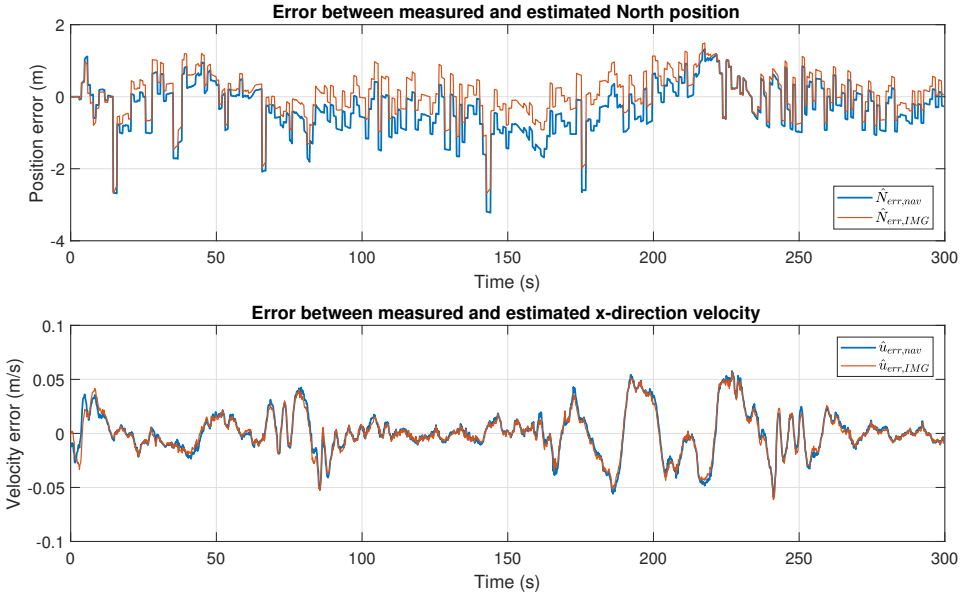


Figure 4.16: *Simulation results scenario 2: Error plot comparing the error between the two estimated values and the measured values for North position and velocity in x-direction*

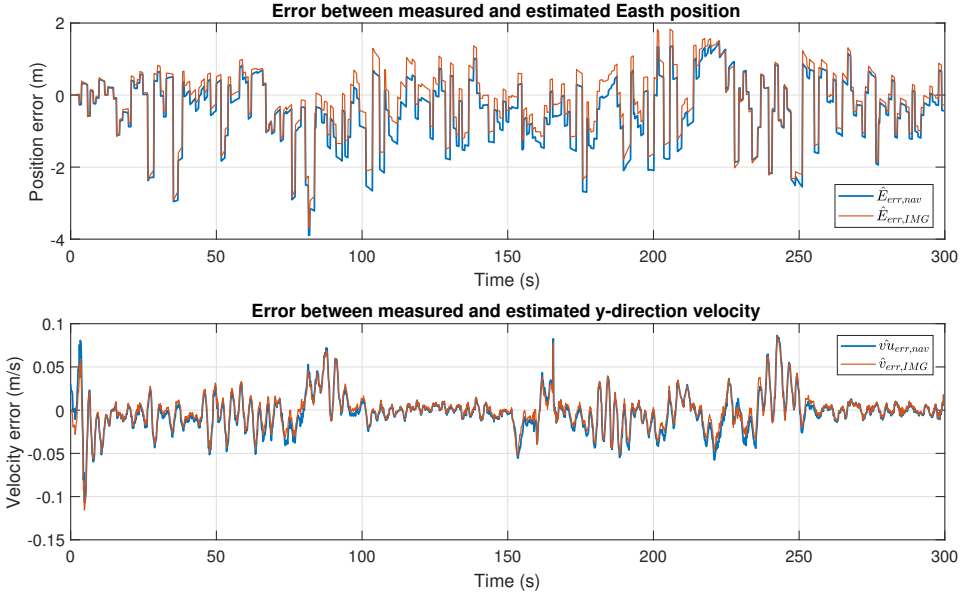


Figure 4.17: *Simulation results scenario 2: Error plot comparing the error between the two estimated values and the measured values for East position and velocity in y-direction*

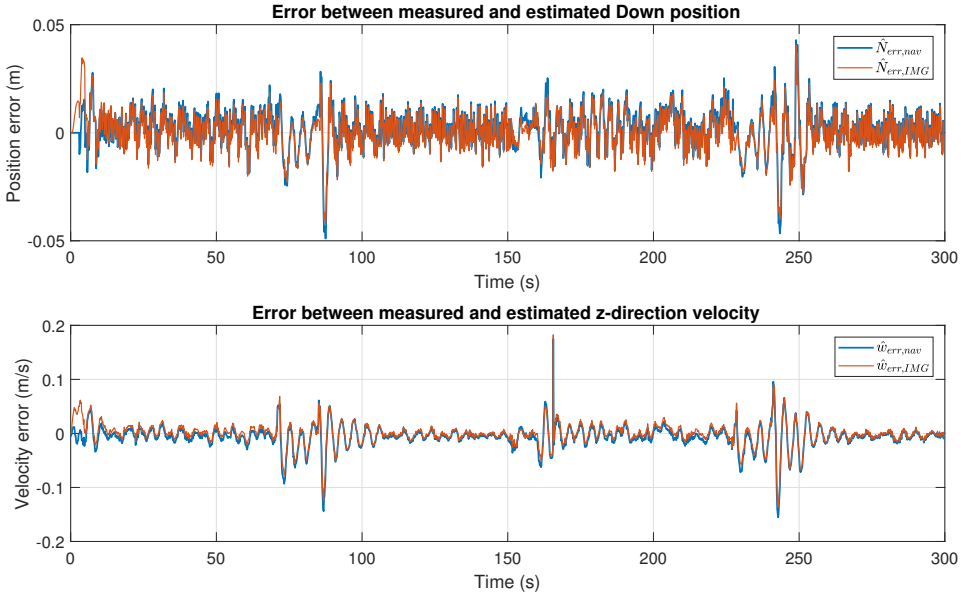


Figure 4.18: *Simulation results scenario 2: Error plot comparing the error between the two estimated values and the measured values for Down position and velocity in z-direction*

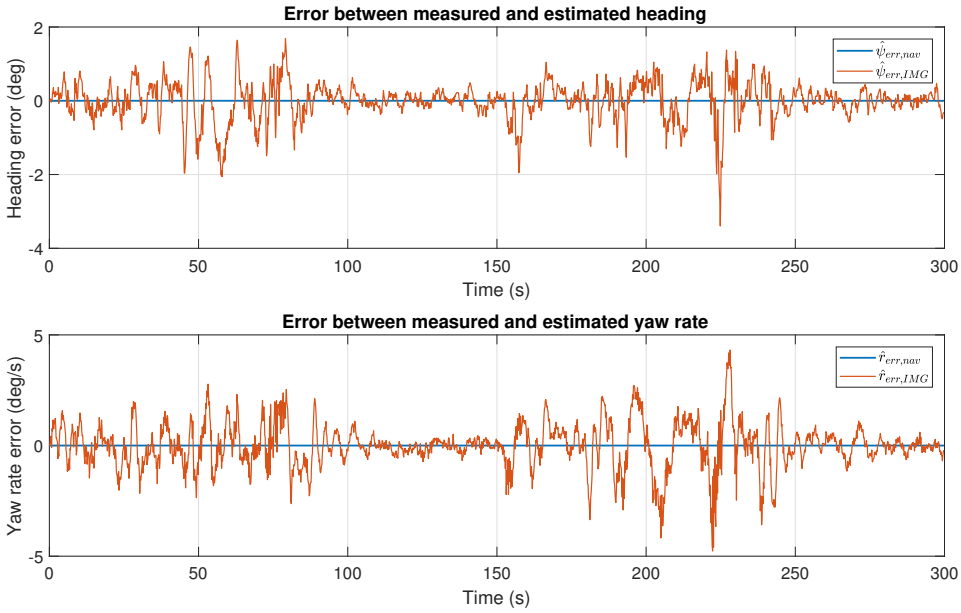


Figure 4.19: *Simulation results scenario 2: Error plot comparing the error between the two estimated values and the measured values for heading and yaw rate*

Chapter 5

Discussion

5.1 Computer Vision

During the work on the project thesis last year, leading up to this master thesis, the SURF feature detection and description method was applied to the VME for an underwater ROV. The results from that project showed that the computational load from the algorithms was too high to run the VME program online in the control system. Thus, for the master thesis work, another method is proposed. The BRISK feature detection and description has a big advantage over the SURF algorithm with regards to computational speed. The results from the comparison show that the performance of the two methods are comparatively good.

The binary descriptors of the BRISK method is often 5-6 times faster than the SURF and provided a higher number of matched features in the comparison presented in Section 3.4. To check the accuracy of the methods, the position change of the features in the image frame were compared. The standard deviation of this distance for all features matched were computed, and the results plotted in Figure 3.9. As expected, the accuracy increases with the number of features detected. The difference of standard deviation between the two methods is also very low, having similar standard deviation in both directions. This confirms the findings in [19, 20] discussed in Section 3.3.

A key point to the computer vision problem is the concept of probability in the solutions, with how much certainty can we guarantee a match. For this type of feature detection and matching there will be outliers, incorrect matches that will cause problems in solving for the camera motion. In this project there are two main mechanisms to avoid outliers clouding the solution. The first resides in the feature matching itself, as explained in Section 3.3. However, the nearest-neighbour search for matches and the threshold of the ratio between the two nearest neighbours will sometimes lead to an incorrect match when there are similar features in the image. This can be a possibility underwater, where lighting is difficult.

The other mechanism does not remove outliers from the feature detection and matching, but rather minimises their relevance to the solution. The Gauss-Newton optimisation takes in three random features in the images and the corresponding camera frame coordi-

nates. The algorithm then iterates by the least-squares method until a solution is within acceptable limits. This procedure can be done several times, in this project the process is initialised 50 times, creating 50 solutions, each based on three randomly chosen features. This way the solution that fits the most features will be accepted as the correct one. If one of the features randomly chosen is an outlier match, the solution will not fit many features and be discarded. Thus, we can always ensure that that the best solution is chosen, based on correct matches.

There are still more possible improvements to the VME algorithm. As it is today, the program utilises the openCV library with its built-in functions. The feature detection is quite effective, and faster than the Matlab implementation developed for the project thesis. However, the matching procedure implemented in the master thesis is the brute-force matching algorithm. The algorithm will compare a feature with all possible features in the other image, by ”brute-forcing” its way through the features.

For a slowly-moving ROV, and a sufficiently high frame rate, the features in the one image will be located at a small distance from the features in the other image. Thus, to speed up the VME, the algorithm can be altered to check for matches in an area around the features, expanding from the previously known position. This method will of course fail sometimes, having to look through the entire image for a match. However, when failing it will work as a brute-force matching algorithm, thus either equally fast or faster than implemented in this thesis. The benefits from this setup is explained in [12], where it is claimed the algorithm runs at 3-4 frames per second, twice as fast as obtained in this project.

The image set from Stokkbergneset, used in the simulations, has an update rate of 0.5 frames per second, making it the ”slowest” sensor on the ROV. However, the stereo cameras mounted on the ROV has a maximum frame rate of 20 frames per second, meaning the potential for a higher sensor update rate is there. The final algorithm developed for this project runs at an average of 2 frames per second, which is faster than GPS and an acoustic positioning system. [30]

The program can give both an absolute relative position change from frame to frame and the average velocity between frames, depending on the application. For implementation in the ROV control system, as an additional sensor input to the Kalman Filter, the velocity between frames was chosen to be best. However, if the algorithm was to be developed into a SLAM framework, the absolute position would be used to build a map of the surroundings online. This was however not attempted in this project, but could be an interesting approach for further development.

The SLAM approach is used in [21]. The paper proposes consecutive stereo pairs, where the system extracts the image features and computes the 3D coordinates. The features and their corresponding 3D points are stored in a database with a node identification number. Thus, the 3D points will form a virtual map of the surroundings that can be used later. The SLAM approach could be a natural continuation of this project.

5.2 Motion Estimation

As explained in Section 3.5 the Gauss-Newton optimisation to estimate the motion is a very effective and fast method to find the rotations and translations of the camera. Despite

initialising the parameters at zero, the algorithm completes a solution at an average of between 5 and 15 iterations. These results are similar to the ones discussed by Geiger et. al [12] for their 3D reconstruction algorithms. This means the Gauss-Newton will only make a small dent in the overall computational effort for the algorithm. Thus, in order to speed it up further more, the feature detection and matching is the major area of improvement.

Still, the average computational time of one run is at 0.5 seconds, which means the system has an update rate of approximately 2 Hz. This is lower than most sensors on the ROV, but can compete with GPS measurements and transponders used at great depths. There is however, some differences in computational time, depending on the quality of images and environment at capture. As explained in Section 3.4, the computational time increases with increasing number of features. This could be a problem for a potential experimental implementation, as the environment and quality of the images cannot always be controlled.

The computational time for the algorithm also constitutes a delay of the measurement to the Kalman filter, compared to the other sensors. An average delay per measurement of 0.5 seconds is unfortunate in a control system, as the measurement is not fresh. This is another reason for choosing the velocity output of the VME compared to the position change output. Assuming the ROV is slowly moving, the change in velocity normally doesn't happen in 0.5 seconds, thus this delay is not catastrophic for the Kalman Filter. The delay can be fixed by applying a fusion of time delayed measurements, even with an uncertainty in the length of the time delays as in [43]. This was not attempted in this thesis.

The results from the VME on the Stokkbergneset image set are presented in Section 3.6. The results from the VME are presented and compared with the measurements from mainly the DVL sensor during the same mission. As the plots show (Figures 3.12, 3.13, 3.14), the estimated translations follow the trend of the measurements from the ROV. The VME results overshoots the measurements in the x-direction, but other than that provide good estimations of the ROV velocity when compared with the ROV measurements.

In the y-direction, the VME results are closer to the measured values. There are some wild points as can be seen from the plot. In the z-direction the VME results are almost identical, aside from the occasional wild point and some oscillations. The camera frame direction corresponding to the x-direction in the ROV body frame is the z-direction, meaning the depth in the image. This is the most difficult distance to estimate, as the image frame is ultimately in 2D. In order to accurately estimate the depth, stereo vision is needed, and the camera parameters must be calibrated accurately.

The calibration was not performed as described in Section 3.1 for the mission to Stokkbergneset. The parameters were computed by the Agisoft Photoscan software as discussed in Chapter 4 based on the images and motion after the mission. This is not as accurate as if the calibration was set up as performed in Section 3.1. As already mentioned, the distortion is an important factor for underwater imagery. Thus, without being able to validate the camera calibration for the image sets used in this mission, it could be a source of error.

The VME algorithm has a slight tendency to overestimate the rotations compared to the translations. This is most easily visible in Figure 3.13, where the velocity in the y-direction and pitch rate are displayed. The pitch rate estimates from the VME follows

the same pattern as the velocity in z-direction in Figure 3.14. As the ROV moves up and down, the VME will interpret this as rotations around the camera frame x-axis, which corresponds to pitch in the ROV body frame.

5.3 Simulation

The simulation of the ROV observer, providing the control system with estimates of the ROV position and velocity in 4 DOFS, is presented in Section 4.2. The simulation is performed for two scenarios. The online state estimates are here compared to the estimates from the off-line simulation of the Kalman Filter including the results from the VME with the same navigational data. There is a distinct difference between the results for the velocities and the position of the ROV. The estimates without VME is referred to as old estimates, while the estimates with the VME is referred to as new estimates.

Regarding the velocities, the new estimates are better than the old estimates, meaning they are closer to the measured values. This is visible in the plots showing the velocities in the x, y, and z-direction (Fig. 4.3-4.5 and 4.12-4.14). However, the new estimate of the yaw rate differs from the old estimate in a negative direction. As discussed briefly in Section 4.2, it seems the heading and the yaw rate are sent straight through the Kalman Filter, meaning the measurements are identical to the estimates. Thus, with the inclusion of the VME in the Kalman Filter, these will be somewhat different.

The ROV control system has a separate attitude observer [3], that estimates heading and yaw rate, thus the Kalman Filter state observer doesn't affect these. As the plots show, the output from the new estimates has been filtered. Most peaks have been shaved off a little, and the new estimate is smoother than the old, which is the Kalman Filter's main job in addition to predict the states.

The position estimates should not be as different from the old estimates. However, there is a big difference here. As the plots show, the new estimates are much closer to the measurements from the transponder than the old estimates. As have been already mentioned at the presentation of the results, the big discrepancy could result from the Kalman Filter convergence. The old estimates are processed online on the observer with time to converge. The new estimates are processed on a Kalman Filter that has to restart each time the simulation is started.

Another explanation of the large discrepancy is the transponder accuracy in general. A small error in the calibration can send the measurements off the actual trajectory of the ROV. For the trajectory plots in the North-East plane, the ground truth is also included. Ground truth is in this context the post-processed estimates of the ROV's absolute position based on the images taken during the mission. By stitching together the images in the Agisoft software, the images can provide very accurate position estimates.

As the results show, the ground truth is very close to the old estimates of the ROV trajectory. This is another indication that the transponder measurements are a little off from the actual trajectory of the ROV. This discrepancy is also represented in the individual plots showing the position in the North and East direction. The transponder presumably has an offset in its measurements that are not accounted for in the simulations as the offset is unknown at this point.

The two scenarios are situated at different depths, the first at 100 meters, the second

at 300 meters. This has a big impact on the accuracy of the transponder measurements. The acoustic signals have to travel from the vessel at sea level to the transponder mounted on the ROV. Thus, the accuracy is much better for the scenario at 100 meters than at 300 meters. This is most visible at the North-East plots in Figures 4.2 and 4.11, where the measurements are more scattered in the latter.

This will also have an impact on the simulated estimates in the North-East plane. The plotted trajectory of the new estimates at 100 meters is much closer to the old estimates and the ground truth, than the plotted trajectory of the new estimates at 300 meters. This is an important observation as the reliability of the acoustic measurements clearly worsens at increasing depth. There is thus a need for online tuning of the measurement noise covariance matrix to reduce the trust in the acoustic measurements at increasing depths.

The Kalman Filter simulated for the new estimates always start in the first measured position, and will then continue to base its estimates on the measurements. The bias estimate is also supposed to fix some unmodelled dynamics and environmental forces. However, as also the bias needs time to converge, the new estimates are different than both the new estimates and the ground truth. When the ROV runs online, the bias builds up and converges. When simulating offline, the bias has to build up and converge again for each run or scenario. Thus, the observer needs time for the bias to converge, and give an accurate estimate of the environmental forces, cable drag etc.

The work on the thesis is part of a larger group working on the ROV control system for the newest addition to NTNUs ROV park, the Minerva 2. Due to delays on deliverance, the experimental tests had to be cancelled. The evaluation of the performance of the VME included in the ROV control system is thus insufficient to conclude that it improves the accuracy of navigation of the ROV. However, the performance of the VME alone compared to the actual ROV movements has been proven in Section 3.6.

The system is implemented in the ROV control system in LabVIEW. The output from the VME is sent through a User Datagram Protocol (UDP) that allows messages being sent across network connections. The motion estimations are here applied to the Kalman Filter in the ROV control system according to equations in Section 2.5. As the experimental tests weren't performed, the author cannot conclude VME's successful integration with the ROV control system developed in LabVIEW.

5.4 ROV autonomy

The goal of the thesis is to increase the ROVs accuracy of positioning and increase its situational awareness. As stated in Section 1.2, the development of the VME is done to include the output from the estimation in the ROV control system and the state estimator to increase the accuracy of the ROV position and velocity estimates. Underwater navigation is one of the most important problems to solve in order to increase the autonomy of the ROV. As it stands now, the VME performance has been proven to deliver a slightly improved velocity estimate, however, without having it tested online on an actual mission, it is hard to conclude the benefit to increased accuracy of positioning.

The goal of the inclusion of computer vision in the control system is to increase the precision for close range navigation for tasks such as docking, manoeuvring inside structures or autonomous control of manipulators. As stated in [2], the limitations of acoustic

positioning and dead-reckoning navigation have to be challenged before achieving a higher level of autonomy. Computer vision techniques have matured and can offer a part of the solution to the challenges. As discussed in this thesis, the simulation on the close-range navigation in Section 4.2 show that computer vision accurately provide navigational data.

This system can be further developed into a SLAM approach where the control system can use loop-closings when revisiting areas. Then the ROV can localise itself in the map created online, which would reduce drift by applying a fixed-point reference to the control system. According to [21] the focus has been directed at enhancing the SLAM techniques with integration of an extended KF, dead-reckoning data, landmark data and loop closings. The loop closing is both considered a problem and a solution to drift. When the ROV revisits areas it closes the loop and can thus eliminate the drift from other sensors. However, if the VME fails, the ROV cannot close the loop and have to look through the entire database to re-localise in the map.

ROV autonomy is an important research area at NTNU AMOS [2]. The goal with improved autonomy for the ROV is to automate tasks such as manoeuvring, inspection, sampling and manipulation. If this is achieved, we are moving more in the direction of an intervention AUV. An intervention AUV can be based on the seafloor and perform tasks previously done by surface-controlled ROVs. This can be a huge cost benefit for businesses that operates in areas such as offshore oil and gas, monitoring, biological sampling and new areas such as deep-sea mining.

The visual motion estimation developed in this thesis is not close to fulfil the goals of ROV autonomy, but is a step in the direction of increased utilisation of computer vision techniques for underwater vehicles. This in turn can really boost underwater navigation for ROVs and facilitate for an increased level of autonomy in ROV operations.

Chapter 6

Conclusion

6.1 Concluding Remarks

This thesis has shown how computer vision techniques can improve underwater navigation by using a stereo camera rig to estimate the ROV motion. The results from the VME implementation proved that by using feature based method of estimating the camera frame motion, the output correspond with the actual ROV motion. The feature detection with spatial and temporal matching of consecutive stereo image pairs was implemented using the BRISK algorithms, reducing the computational effort compared to other algorithms. Compared to the SURF method, the BRISK showed comparative accuracy at a considerable reduced computational time.

The camera ego-motion was solved using a Gauss-Newton optimisation, solving for the translational and rotational camera motion. The optimisation only needed between 5-15 iterations on average to find a suitable solution to the problem, even when initialised at zero. This resulted in a fast algorithm, able to estimate the motion of the camera at an average of 0.5 seconds. The VME could then output motion estimates at an update rate of 2 Hz. The update rate is considerably slower than the DVL or IMU, but can compete with the transponder update rate.

A drawback with the computational time of 0.5 seconds is that the motion estimate is available 0.5 seconds after it is measured. This problem is minimised when the ROV moves at constant speed, but could cause errors if the ROV is in an accelerating or decelerating state. By applying fusion of time delayed measurements in the Kalman Filter this problem could be eliminated entirely, however, this was not attempted in this thesis.

The VME was included in the Kalman Filter state observer in the ROV control system, and simulated with raw measurements from the other sensors on a previous mission. The ROV velocities showed a slight improvement with the VME as an additional sensor. The position estimates does not give enough information to conclude with an improvement of the estimate. However, the simulated Kalman Filter was able to estimate a position similar to the one estimated online in the ROV control system.

Computer vision techniques has been proven to quite accurately estimate the motion

of the ROV, and is thus a promising technique to increase the level of autonomy in ROV operations. Computer vision is probably one of many applications necessary to accomplish complete autonomy for the ROV as outlined in Section 1.1.

6.2 Further Work

This thesis has attempted to improve the accuracy of the ROV underwater navigation in order to increase its level of autonomy. The system has been simulated with a similar state observer as in the ROV control system, however, not been tested in sea trials. Thus, in order to fully verify the possible improvement to the navigation, the system must be tested online with ROV control system on sea trials.

As discussed in Chapter 5, the output from the VME is a 6-DOF velocity vector. However, to solve the transponder inaccruacy and the drift during dead-reckoning, the VME should also give a position estimate. In addition, this can be included in a wider SLAM framework. In close-range operations, the ROV can then build a map of its surroundings and get near-absolute position fixes while submerged in operation.

The author would recommend focusing on this aspect of the computer vision techniques to further increase the accuracy of underwater navigation. The SLAM approach can also be included in the work currently being done at NTNU AMOS regarding object recognition and autonomous manipulation. This is an important step to achieve the goals outlined in [2, 8].

There are also improvements related to the feature-based visual motion estimation developed in this thesis, especially regarding computational effort. The feature detection is here accomplished through built-in functions in the OpenCV C++ library. However, by applying the techniques outlined in Section 3.3 manually, the computational time may be reduced, as accomplished in [12]. The feature matching can also be made more efficient by assuming a slowly-moving vehicle and search for matching features in close proximity to the original features.

Bibliography

[1] Robert D. Christ and Robert L. Wernli Sr. Chapter 3 - design theory and standards. In Robert D. Christ and Robert L. Wernli, editors, *The ROV Manual (Second Edition)*, pages 55 – 92. Butterworth-Heinemann, Oxford, second edition edition, 2014.

[2] Martin Ludvigsen and Asgeir J. Srensen. Towards integrated autonomous underwater operations for ocean mapping and monitoring. *Annual Reviews in Control*, 42:145 – 157, 2016. ISSN 1367-5788.

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

[4] Daniel de Almeida Fernandes. *An Output Feedback Motion Control System for ROVs: Guidance Navigation and Control*. PhD thesis, Norwegian University of Science and Technology, 2015.

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

[6] Richard Szeliski. *Computer Vision: Algorithms and Applications (Texts in Computer Science)*. Springer, 2010.

[7] Bruno Siciliano and Oussama Khatib, editors. *Springer Handbook of Robotics*. Springer, 2008.

[8] T. O. Fossum, M. Ludvigsen, S. M. Nornes, I. Rist-Christensen, and L. Brusletto. Autonomous robotic intervention using rov: An experimental approach. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–6, Sept 2016.

[9] O. Pizarro and H. Singh. Toward large-area mosaicing for underwater scientific applications. *IEEE Journal of Oceanic Engineering*, 28(4):651–672, Oct 2003. ISSN 0364-9059.

[10] N. Gracias and J. Santos-Victor. Underwater mosaicing and trajectory reconstruction using global alignment. In *MTS/IEEE Oceans 2001. An Ocean Odyssey. Conference Proceedings (IEEE Cat. No.01CH37295)*, volume 4, pages 2557–2563 vol.4, 2001.

[11] O. Pink, F. Moosmann, and A. Bachmann. Visual features for vehicle localization and ego-motion estimation. In *2009 IEEE Intelligent Vehicles Symposium*, pages 254–260, June 2009.

[12] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968, June 2011.

[13] Maimone Mark, Cheng Yang, and Matthies Larry. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.

[14] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185 – 203, 1981. ISSN 0004-3702.

[15] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2432–2439, June 2010.

[16] A. Wedel, D. Cremers, T. Pock, and H. Bischof. Structure- and motion-adaptive regularization for high accuracy optic flow. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1663–1668, Sept 2009.

[17] V. Lempitsky, S. Roth, and C. Rother. Fusionflow: Discrete-continuous optimization for optical flow estimation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[18] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In Tomás Pajdla and Jiří Matas, editors, *Computer Vision - ECCV 2004*, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[19] Sahin Isik. A comparative evaluation of well-known feature detectors and descriptors. *International Journal of Applied Mathematics, Electronics and Computers*, 3, January 2014.

[20] Shaharyar Khan and Zahra Saleem. A comparison of sift, surf, kaze, akaze, orb, and brisk. In *2018 International Conference on Computing, Mathematics and Engineering Technologies*, 03 2018.

[21] Pep Lluis Negre Carrasco, Francisco Bonin-Font, and Gabriel Oliver Codina. *Stereo Graph-SLAM for Autonomous Underwater Vehicles*, pages 351–360. Springer International Publishing, Cham, 2016.

[22] Stéphane Bazeille, Isabelle Quidu, and Luc Jaulin. Color-based underwater object recognition using water light attenuation. *Intelligent Service Robotics*, 5(2):109–118, Apr 2012. ISSN 1861-2784.

[23] D. Walther, D. R. Edgington, and C. Koch. Detection and tracking of objects in underwater video. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–544–I–549 Vol.1, June 2004.

[24] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. *SURF: Speeded Up Robust Features*, pages 404–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[25] Karel Zuiderveld. Contrast limited adaptive histogram equalization. In *Graphics gems IV*, pages 474–485. Academic Press Professional, Inc., 1994.

[26] Thor Inge Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, Chichester, West Sussex, U.K. Hoboken N.J, 2011.

[27] Asgeir J. Sorensen. *Propulsion and Motion Control of Ships and Ocean Structures. Lecture Notes*, volume UK-13-76. Department of Marine Technology, NTNU, 2013.

[28] K. Vickery. Acoustic positioning systems. a practical overview of current systems. In *Proceedings of the 1998 Workshop on Autonomous Underwater Vehicles (Cat. No.98CH36290)*, pages 5–17, Aug 1998.

[29] R. Lee Gordon. Acoustic doppler current profiler - principles of operation a practical primer, 1996.

[30] Fredrik Dukan and Asgeir J. Srensen. Integration filter for aps, dvl, imu and pressure gauge for underwater vehicles. *IFAC Proceedings Volumes*, 46(33):280 – 285, 2013. ISSN 1474-6670. 9th IFAC Conference on Control Applications in Marine Systems.

[31] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000. ISSN 0162-8828.

[32] Duane C. Brown. Close-range camera calibration. *PHOTOGRAMMETRIC ENGINEERING*, 37(8):855–866, 1971.

[33] Opencv computer vision library. `https://opencv.org/`, 2018. Accessed: 2018-05-19.

[34] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004. ISSN 1573-1405.

[35] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[36] Matthew Brown and D Lowe. Invariant features from interest point groups. In *BMVC 2002: 13th British Machine Vision Conference*, pages 253–262, September 2002.

[37] Donghwa Lee, Gonyop Kim, Donghoon Kim, Hyun Myung, and Hyun-Taek Choi. Vision-based object detection and tracking for autonomous navigation of underwater robots. *Ocean Engineering*, 48(Supplement C):59 – 68, 2012. ISSN 0029-8018.

[38] S. Leutenegger, M. Chli, and R. Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *2011 International Conference on Computer Vision*, pages 2548–2555, Nov 2011.

[39] Elmar Mair, Darius Hager, Gregory D.and Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 183–196, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[40] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 778–792, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[41] Jorge Nocedal and Stephen Wright. *Numerical Optimization (Springer Series in Operations Research and Financial Engineering)*. Springer, 2006.

[42] *Agisoft Lens User Manual*. Agisoft LLC, 0.4.0 edition, 2011. URL `http://downloads.agisoft.ru/lens/doc/en/lens.pdf`.

[43] S. J. Julier and J. K. Uhlmann. Fusion of time delayed measurements with uncertain time delays. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 4028–4033 vol. 6, June 2005.

[44] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for rgb-d cameras. In *2013 IEEE International Conference on Robotics and Automation*, pages 3748–3754, May 2013.

Appendices

A ROVs

A.1 ROV SF-30k

The SF-30k ROV is produced by Sperre ROV technology and is a standard ROV model. The specifications of the ROV are listed in Table A.1. The ROV is powered by 6 thrusters, each supplying 3000 W of power. The top speed of the ROV is 2.1 knots (1.1 m/s) in surge. The SF-30k is pictured on board NTNU research vessel Gunnerus in Figure A.1

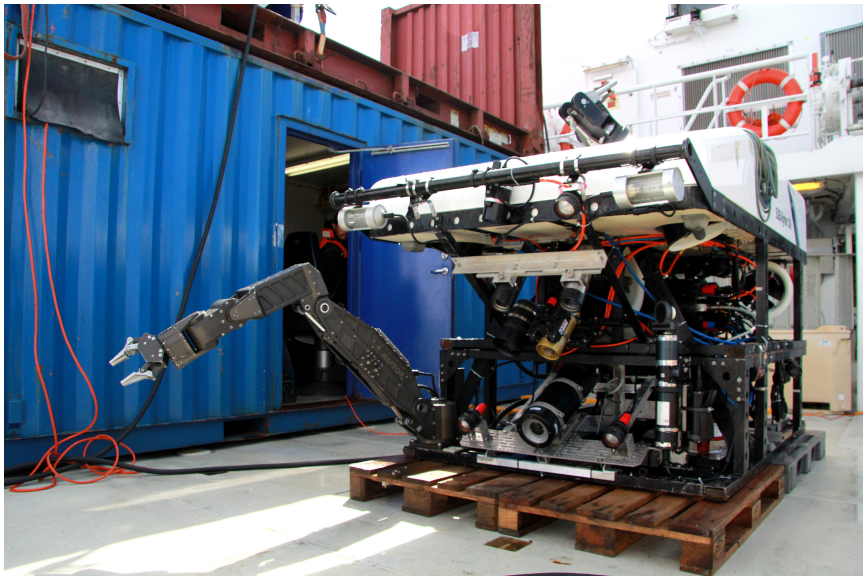


Figure A.1: *The SF-30k ROV pictured at RV Gunnerus. Courtesy of NTNU*

Table A.1: *ROV SF-30k specifications*

Make	Sperre AS
Model	SUB-fighter 30K
Year	2004
Dimensions (LWH)	260 x 150 x 160 cm
Weight (air)	1800 kg
Payload	60 kg
Max depth	1000 m
Power Input	230 VAC, 3 phase, 40 kW, 125 A
Thrusters	Horizontal: 2 x 3000 W Vertical: 3 x 3000 W Lateral: 1 x 3000 W (electrical asynchronous motors)
Manipulators	7-function hydraulic arm Kraft Telerobotics (Raptor)
Umbilical	27 mm
Lights	6 x 250 W halogen lights 2 x 400 W HMI lights

The values for mass, added mass, linear and quadratic damping of ROV SF-30k are given in (A.1), (A.2), (A.3) and (A.4) respectively.

$$\boldsymbol{M}_{RB} = \begin{bmatrix} 1862.87 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1862.87 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1862.87 & 0 & 0 & 0 \\ 0 & 0 & 0 & 525.39 & 1.439 & 33.413 \\ 0 & 0 & 0 & 1.439 & 794.199 & 2.596 \\ 0 & 0 & 0 & 33.413 & 2.596 & 691.229 \end{bmatrix} \tag{A.1}$$

$$\boldsymbol{M}_{A} = \begin{bmatrix} 779.79 & -6.8773 & -103.32 & 8.5426 & -165.54 & -7.8033 \\ -6.8773 & 1222 & 51.29 & 409.44 & -5.8488 & 62.726 \\ -103.32 & 51.29 & 3659.9 & 6.1112 & -386.42 & 10.775 \\ 8.5426 & 409.44 & 6.1112 & 534.9 & -10.027 & 21.019 \\ -165.54 & -5.8488 & -386.42 & -10.027 & 842.69 & -1.1162 \\ -7.8033 & 62.726 & 10.775 & 21.019 & -1.1162 & 224.32 \end{bmatrix} \tag{A.2}$$

$$\boldsymbol{D}_L = \begin{bmatrix} 74.82 & 0 & 0 & 0 & 0 & 0 \\ 0 & 69.48 & 0 & 0 & 0 & 0 \\ 0 & 0 & 728.40 & 0 & 0 & 0 \\ 0 & 0 & 0 & 268.80 & 0 & 0 \\ 0 & 0 & 0 & 0 & 309.77 & 0 \\ 0 & 0 & 0 & 0 & 0 & 105 \end{bmatrix} \tag{A.3}$$

$$\boldsymbol{D}_{NL} = \begin{bmatrix} 748.22 & 0 & 0 & 0 & 0 & 0 \\ 0 & 992.53 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1821.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 672 & 0 & 0 \\ 0 & 0 & 0 & 0 & 774.44 & 0 \\ 0 & 0 & 0 & 0 & 0 & 523.27 \end{bmatrix} \tag{A.4}$$

B Camera Set-up

For both the simulation and experiments a set of two Allied Vision Prosilica model GC-1380 cameras has been used. The cameras use a GigE Vision Gigabit Ethernet interface for high-speed data transmission over Ethernet networks. The cameras are set up with Schneider Kreuznac Cinegon lenses.

Table B.1: *Camera specifications Allied Vision Prosilica GC1380*

Interface	IEEE 802.3 1000baseT
Resolution	1360×1024
Sensor type	CCD Progressive
Sensor size	Type 2/3
Megapixels	1.4 MP
Pixel size	6.45 µm×6.45 µm
Max frame rate	20.2 fps
Lens mount	C-mount
Dimensions (L×W×H)	59×46×33 mm
Weight	104 g
Power consumption	3.3 W at 12 VDC
Operating temperature	0-50 °C

The camera specifications are presented in Table B.1, and the lens specifications listed in Table B.2. Both camera and lens is presented in Figure B.1.

Table B.2: *Lens specifications Schneider Kreuznac Cinegon 1.4/8*

F-numner	1.4
Focal length	8.2 mm
Image circle	11 mm
Transmission	400-1000 nm
Interface	C-mount
Weight	90 g

(a) *Prosilica GC1380 camera. Courtesy Allied Vision*

(b) *Cinegon 1.4/8 lens. Courtesy Schneider Kreuznach*

Figure B.1: *The camera and lens used in the stereo camera setup on the ROV*