Master's thesis

2019

Johan Loe Kvalberg

Master's thesis

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

NTNU
Norwegian University of
Science and Technology

Johan Loe Kvalberg

Monocular Visual Odometry for a Mini ROV

June 2019

NTNU
Norwegian University of
Science and Technology

Monocular Visual Odometry for a Mini ROV

Johan Loe Kvalberg

Marine Technology
Submission date: June 2019
Supervisor: Professor Martin Ludvigsen

Norwegian University of Science and Technology
Department of Marine Technology

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MASTER THESIS IN MARINE CYBERNETICS

SPRING 2019

FOR

STUD. TECH. JOHAN LOE KVALBERG

Title: Monocular Visual Odometry for a mini ROV.

Background

There has been a huge rise of small sized Remotely Operated Vehicles (ROVs), both aiming for the low-cost market and private consumers, and as a substitution for costly inspection operations with work class ROVs or divers. In addition to be used for underwater discoveries and recreational use, the mini ROVs can be used in various tasks, such as underwater inspection, environmental monitoring and research. The low cost and easy handling has made it possible to avoid hiring professional divers or ROV ships with operators.

Even though many mini ROVs are promoted with easy operation, the user can experience a lack of control due to operation with a manual joystick. In addition, currents and drag forces from the umbilical can largely affect the small vehicles. The option of dynamic positioning of the ROV would be beneficial in many settings for easily handling. For small sized ROVs, the use of computer vision is one of the most promising methods for navigation and motion estimation. The limitations of a single camera, limited amount of sensors, and low computational power sets high demands to effective, accurate and robust algorithms.

Work Description

1. Background and literature review to provide information and relevant references on:
    a. Navigation and control with the help of computer vision.
    b. Limitations and challenges associated with a monocular camera scheme.
    c. Available and open source graphical software packages.
    d. Feature extraction and description in camera images.
2. Development of a monocular visual odometry algorithm:
    a. Development with use of the Blueye Pioneer ROV and camera.
    b. Focus on real time application implementation
    c. Sensor fusion with IMU and/or depth sensor.
    d. Explain the choices of the core components in the visual motion estimation algorithms.
3. Experimental work;
    a. Testing on prerecorded datasets as a proof of concept.
    b. Implementation and testing on the Blueye Pioneer ROV

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

The report shall be written in English and edited as a research report including literature survey, description of mathematical models, description of algorithms, simulation results, model test results, discussion and a conclusion including a proposal for further work. Source code should be provided on a CD, memory stick or similar. It is supposed that the Department of Marine Technology, NTNU, can use the results freely in its research work, unless otherwise agreed upon, by referring to the student's work. The thesis should be submitted in within June 2019.

Supervisor: Martin Ludvigsen

Abstract

This thesis presents a real-time motion estimation of a small sized remotely operated vehicle (ROV) using a single camera and a pressure sensor. This in order to increase the level of automation and ease for navigation for underwater vehicles. The motion estimation is performed through so-called monocular visual odometry (MVO), which estimates the change in position and attitude with the use of camera images. This is done by estimating the cameras relative motion, and thus the ROV motion, from one camera image to the next. The estimations is to be used for dynamic positioning.

The motion information from the images is extracted with a feature-based method using oriented Features from Accelerated Segment Test and rotated Binary Robust Independent Elementary Features (ORB). ORB is used to extract and describe features in the images, while matching of the features is performed with Brute-Force matching comparing the Hamming distances of the binary descriptors. Based on the matches between two consecutive frames, Nistér's five-point algorithm with Random Sample Consensus (RANSAC) is used to obtain the essential matrix, from which the relative translation and rotation is computed.

The implemented MVO algorithm is able to run with at an average of 10 frames per second. The performance of the monocular visual odometry algorithm is tested on both prerecorded datasets and real-time tests in the Marine Cybernetics Laboratory. The results shows that the algorithm is able to estimate the relative motion in air, but not in the underwater environment. Due to high errors in the rotational estimates, the resulting position estimates becomes too unstable to be used in a dynamic positioning implementation.

A visual inertial odometry method using the inertial measurement unit (IMU) onboard the ROV was tried implemented. Due to an uncalibrated accelerometer, the motion estimations and IMU measurements were fused through an error state Kalman Filter (ESKF). Not being able to distinguish the measurement bias from the real values in the ESKF, the implementation ended up with not being used in the final presented algorithm.

In order to improve the motion estimation, a structure from motion or simultaneous localisation and mapping approach should be considered. Estimating the vehicles position with respect to known features or objects would reduce the errors and drifts accumulated through a visual odometry approach.

Sammendrag

Denne oppgaven presenterer en estimering av bevegelsen til en liten fjernstyrt undervannsfarkost (ROV) i sanntid, ved bruk at et enkelt kamera og en trykksensor. Dette, for å øke nivået av automasjon, og for å forenkle navigasjon av undervannsfarkoster. Estimering av bevegelse er gjort gjennom såkalt monokulær visuell odometri, som estimerer endringen i posisjon og orientering ved bruk av kamerabilder. Dette gjøres ved å estimere kameraets relative bevegelse, og dermed ROV-bevegelsen, fra ett kamerabilde til det neste. Estimatene er ment å brukes for dynamisk posisjonering.

Bevegelsesinformasjonen blir hentet ut gjennom en metode basert på kjennetegn i bildet, ved bruk av ORB (oriented Features from Accelerated Segment Test and rotated Binary Robust Independent Elementary Features). ORB blir brukt til å hente ut og beskrive kjennetegn i bildet. Sammenligningen av kjennetegn er gjort gjennom Brute-Force matching ved å sammenligne Hamming-distansen til de binære beskrivelsene. Basert på sammenligning av kjennetegn mellom to påfølgende bilder brukes Nistér's fem-punkts algoritme med Random Sample Consensus (RANSAC) til å beregne den essensielle matrisen som relativ translasjon og rotasjon kan hentes ut ifra.

Den implementerte MVO-algoritmen klarer å kjøre med et gjennomsnitt på 10 bilder i sekundet. Ytelsen til algoritmen er testet både på ferdige datasett og i Marin kybernetikk sitt laboratorium. Resultatene viser at algoritmen klarer å estimere relativ bevegelse over vann, men ikke under vann. På grunn av store feil i rotasjonsestimatene blir posisjonsestimatene for ustabile til å bli brukt i en dynamisk posisjoneringsimplementasjon.

En visuell odometri metode ved bruk av treghetssensoren om bord i ROVen ble forsøkt implementert. På grunn av et ukalibrert akselerometer er bevegelsesestimatene og måledata fra treghetssensoren slått sammen gjennom et error-state Kalman Filter (ESKF). Ettersom det ikke ble klart å skille mellom bias og målinger i den forsøkte ESKF implementeringen ble denne tilnærmingen ikke brukt i den endelige implementasjonen av algoritmen.

For å forbedre bevegelsesestimatene bør en tilnærming gjennom "structure from motion" eller en simultan lokalisering og karlegging (SLAM) vurderes. Beregning av farkostens posisjon i forhold til kjente kjennetegn eller gjenstander vil redusere avvik og drift akkumulert gjennom en visuell odometri-tilnærming.

Preface

This master thesis is a part of the study program Marine Technology at the NTNU, and it was carried out during the spring semester of 2019. The objective of the thesis is to use visual input from a camera together with computer vision to estimate the motion of a small sized ROV. The resulting estimates are to be used for dynamic positioning.

The author of this thesis did not have any experience in the field of computer vision before the work was embarked. The learning process have therefore been steep, yet exciting. The use of computer vision in the underwater environment for various tasks is truly a research topic with vast possibilities.

The work has been carried out at the Department of Marine Technology and in cooperation with Blueye Robotics, with Professor Martin Ludvigsen as supervisor.

Trondheim, June 20, 2019

[illegible]

Johan Theodor Loe Kvalberg

Acknowledgement

During the work and research related to this thesis, there have been different people that deserves the authors gratitude. I would like to thank my supervisor Professor Martin Ludvigsen at NTNU for helpful guidance and support, in addition to informative discussions during meetings. I would also like to thank Oscar Pizarro and Petter Norgren for taking the time to discuss the topic presented and guidance with technical issues, and Professor and Dean Ingvald Strømmen for letting me borrow his private Blueye Pioneer for finalising of the results.

The tests conducted in the MC-lab at the Department of Marine Technology at NTNU had not been possible without help with the experimental setup, both in the lab and in Qualisys, from Mikkel C. Nieslen and Torgeir Wahl.

This thesis has been carried out in cooperation with Blueye Robotics. I would like to thank the entire Blueye team for helpful guidance and support, and for lending out necessary equipment to carry out different experiments.

J.T.L.K

Table of Contents

List of Figures

List of Tables

Acronyms

APS	Acoustic positioning system
AUV	Autonomous underwater vehicle
BRIEF	Binary Robust Independent Elementary Features
DOF	Degrees of freedom
DVL	Doppler velocity log
ECEF	Earth centred earth fixed frame
ECI	Earth-centred inertial frame
EKF	Extended Kalman Filter
ESKF	Error-state Kalman filter
FAST	Features form Accelerated Segment Test
IMU	Inertial measurement unit
INS	Inertial Navigation Sytems
KF	Kalman filter
MC-lab	Marine Cybernetics Laboratory
MEMS	Microelectromechanical systems
MES	Measurement frame
MVO	Monocular Visual Odometry
NED	North east down Frame
RANSAC	Random sample consensus

ROV Remotely operated vehicle

SfM Structure from Motion

SNAME Society of Naval Architects and Marine Engineers

TMS Tether management system

VIO Visual Inertial Odometry

VO Visual Odometry

Chapter 1

Introduction

1.1 Background

1.1.1 ROVs

The worlds oceans covers more than 70 percent of the planet's surface. In many ways, the ocean is the lifeblood of Earth by driving weather, regulating temperature, and support all living organisms [1]. Still, the depth of the oceans holds vast unexplored, unmapped and unobserved areas. In order to explore these distant areas, different underwater vehicles are used and is under constant development. Remotely operated vehicles (ROVs) and autonomous underwater vehicles (AUVs) are two types of unmanned underwater vehicles which dominate this branch.

An ROV normally consists of a frame with buoyancy elements, a camera for visual control, and thrusters in order to make the vehicle move. Many ROVs have typically manipulator arm in order to perform different tasks. The ROV is powered through a cable, called umbilical or tether, which allows data, including audio, images or video, navigation information and sensor readings to be transmitted back to the operator. This allows the ROV to be controlled from a safe, dry and comfortable place, normally a ship or a platform [2].

Exactly who to credit for developing the first ROV is unknown, however some of the first use cases of a tethered ROV can be traced back to 1953, developed by Dimitri Rebikoff. The ROV was named POODLE, and was used primarily for archeological research. Its impact on ROV history was not tremendous [3]. Still, it was a start. In the 1950s the United States Navy began developing vehicles that contributed towards an operational system. Their problem was to recover lost torpedoes and mines. They developed a manoeuvrable underwater camera system, which eventually became the Cable-controlled Underwater Research Vehicle (CURV). The CURV´s sister vehicle, CURV II, is shown in figure 1.1

The use cases for ROVs are many, and spans from site surveys, inspections and operations assistance, to exploration and recreational use. Before offshore activities such as drilling or in-

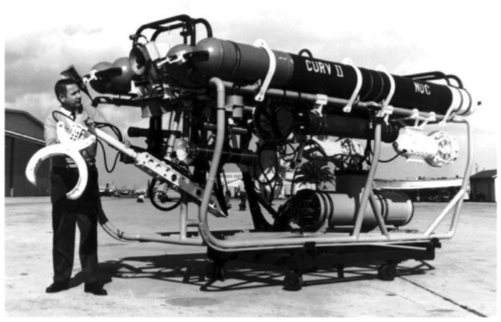

Figure 1.1: The US Navy's CURV II vehicle [3]

stallation, an inspection of the seabed is needed. This can be preformed with use of an ROV. ROVs are normally used for observation and verification, and for engagement and release of guide wires and hooks. In operation assistance, the ROV can perform flow control by chokes and valves. In the last years, small ROVs have emerged in the market, focusing on exploration and recreational use. Due to their small size, low weight and ease of use, they can be transported and controlled by one person only.

ROV classification

The International Maritime Contractors Association (IMCA) have made a classification of ROVs. The vehicles can be classified into five categories dependent on their size and field of use:

- Class I - Observation ROVs. This class consists of small vehicles fitted with camera, light and sonar only. These vehicles are intended for pure observation, even though they may be able to carry an additional sensor, as well as an additional camera.
- Class II - Observation ROVs with payload option. These vehicles are fitted with two cameras or sonars which are viewed/used simultaneously. They also have a basic manipulative capability. These vehicles should not loose original functionality while carrying two additional sensors or manipulators.
- Class III - Work class vehicles. These ROVs are able to carry additional sensors and manipulators. They commonly have a multiplexing capacity that allows additional sensors and tools to be operated without the need of hard-wiring them through the tether system. These vehicles are typically larger and more powerful than vehicles that belong to class I and II.
- Class IV - Towed and bottom-crawling vehicles. These large and heavy vehicles may have limited propulsion power, and some have the need of being towed or pulled through the water by a surface vessel or a winch. Some have only a limited manoeuvrability and are

able to swim limited distances. Bottom-crawling vehicles use a track or wheel system to move around. They are often designed for one specific task, e.g. cable burial.

- Class V - Prototype or development vehicles. Vehicles that are still being developed and those regarded as prototypes are included in this class. Special purpose vehicles that does not fall into one of the other classes is also assigned. AUVs are also included in this class.

In order to fully define the full range of vehicles, a more robust naming convention is used to categorise the vehicles even further. The categories are based on the vehicles weight. The lightest vehicles (up to 91 kg) belongs to the OCROV class. Within the OCROV category, three subcategories are again presented where Mini (or medium) OCROVs are vehicles with submersible weight between 4.5kg and 32kg. In other words, the limit of single-person hand deployment [3]. The vehicle used in this thesis, the Blueye Pioneer, belongs to the Mini OCROVs (herby denoted as mini ROVs) and is further presented in section 2.1.

Mini ROVs entering the market

Today most subsea inspections, maintenance and repair operations are performed with he support of offshore vessels. In addition to the offshore vessel itself, the operations are dependent on ROV systems, tools and experienced operators. The efficiency in the operations is highly dependent on the experience of the ROV operator. The operations of most ROVs are manually controlled, with little or no automatic control functions nor autonomy. The day rate for a support vessel is in the range of hundreds of thousands of dollars [4].

With the use of mini ROVs as an option to dive teams and work class ROVs, a cost-effective solution is available. From a vessel, small boat or quay, these vehicles can be steered by the means of a hand console. Focusing on ease of use, the mini ROVs does not require a professional operator. The last decade, several mini ROVs have entered the market. Some are targeting the professional market, while others are focusing on recreational use and the consumer market. This has lead to that the Mini ROV is the highest volume market segment [5].

1.1.2 Computer Vision

In able to gather data and comprehend its surroundings, the vehicles are dependent on sensing technology. For us humans, eyes are a very effective sensor for recognition, navigation, obstacle avoidance and vision manipulation [6]. Therefore, due to the high content of information they deliver, cameras have become an important sensor for navigation and sensing. How computers can deal with the information given by video images is through computer vision. This field has therefore been of interest to robotic researchers for a long time. The use of computer vision have been important in the development of positioning control of robots, and this technological field have developed drastically the last decade due to the access of higher computational power.

A fundamental problem in positioning control with computer vision is the ability to compute the relative motion between the camera and an object. Humans process visual information in semantic space, meaning that we recognise and classify objects based on their shape, boundaries, line-segments etc [7]. Since these features cannot be detected robustly by computers, the process methodology is quite different in computer vision. Instead informative features such as colour and texture is used to extract information from digital images. Distinct features such as corners, edges and ridges can be used as key points for tracking algorithms.

The problem of estimating a vehicles motion based on visual input started in the early 1980s by Moravec [8]. Most of the early research in VO was encourraged by the NASA Mars exploration program in order to measure 6 degree-of-freedom motion on their all-terrain rovers [9]. The work by Moravec stands out for not only presenting the first pipeline for motion estimation, whose main function blocks are still used today, but for also describing one of the first corner detectors. A predecessor for, among others, the Harris corner detector [10].

Today, computer vision algorithms and software libraries with advanced functions are available. The development have been great, resulting in functionalities such as rendering a 3D model from overlapping images, advanced tracking of objects, automatic object detection and recognition and even self driving cars.

The use of camera images in order to determine odometry information is defined as visual odometry (VO). Information from one or several cameras can be used to estimate a vehicles 3D motion. The term was first defined in the paper by Nistér et al. [11], where they present both a monocular and stereo scheme for a feature-based VO-method. The most common ways to extract motion information from camera images is feature-based or appearance-based methods. Feature-based methods use salient and repeatable features extracted, or tracked, across the images. Features are point of interest in the image that are invariant to scale, illumination, rotation and viewpoint. The points are often described by the appearance of patches of pixels surrounding the point location [12]. With the use of these visual descriptors, comparisons can be established when looking for corresponding points between images.

Today, there are many different feature based algorithms, some of which have become standards, both when it comes to robust feature detection and description. The detector-descriptor which is best suited for a certain application needs to be determined based on a trade-off between accurate and robust detection, and low computational time [13]. Among the well known feature detector-descriptors are SIFT, SURF, ORB, BRISK, AKAZE and KAZE.

Appearance-based methods use the intensity information of all the pixels, e.g. to compute the optical flow [14] which is the velocity distribution of the motion in an image. Optical flow calculations make use of the assumption that the intensity of a pixel defining and object does not change between consecutive frames, and that neighbouring pixels have similar motion. Through the years, the optical flow technique has improved since the early stages at [14]. Different objective functions, optimisation methods used, and modern implementation practices influence the accuracy as stated in [15]. The Lucas-Kanade method is a differential method for optical flow estimation, and is a widely used method in computer vision. The method assumes that the flow is essentially constant around the pixel under consideration, and through the least squares criterion solves the basic optical flow equations for all pixels in the neighbourhood [16].

As for the feature-based method, a lot of optical flow algorithms have also been developed in order to perform tracking or motion estimation. The choice of method is highly dependent on the accuracy required, computational time and the application for the method. However, appearance-based methods rely on constant brightness [15], which is a condition hard to achieve in underwater environments when operating in deeper waters, and with artificial lighting moving together with the camera.

Visual odometry estimates can be used in conjunction with information from other sources such as GPS, IMU, wheel encoders etc. If an inertial measurement unit is used within the visual odometry system, it is commonly referred to as Visual Inertial Odometry (VIO). VIO is widely used in localisations where GPS measurements its not available, e.g. indoor or underwater. Fusing of VO estimates and IMU measurements is either done through a filtering approach, or a smoothing approach through nonlinear optimisation. Filtering approaches enables a fast inference, but the accumulation of linearisation errors aggravates the accuracy. Full smoothing approaches on the other hand are more accurate, but computational demanding [17].

In VIO the accuracy of the IMU is of great importance in order to perform precise motion estimation. Nowadays cheaper and smaller IMUs, such as the MEMS IMU, are being used, e.g. in commercial quadcopters and smartphones [18]. A precondition for many of these VIO applications is that they need calibrated IMU measurements. Bonin-Font et. al presents in [19] a solution for inertial sensor self-calibration. The data given by the MEMS IMU, pressure sensor and a stereo camera rig is fused, adjusted and corrected in a multiplicative error state Kalman filter. Including the biases of the inertial sensors in the state vector results in self-calibration and stabilisation of the sensors, improving the estimates of the robot orientation.

Even though most research in VO has been done using stereo cameras, monocular visual odometry (MVO) is also an important aspect in VO. The difference from the stereo scheme is that 3D motion estimations have to be done from 2D data. Another important difference is that the absolute scale unknown. Without the scale, camera poses are determined with respect to the relative scale between the first two frames, which is usually set to one. Different approaches of scale estimation have therefore been published.

Creuze presents in [20] an online estimation of the scale factor. The VO method is aided by inertial and pressure measurements. The odometry algorithm calculates the optical flow using the iterative Lucas-Kanade method with pyramids. Compensation of yaw, pitch and roll is applied to the feature points extracted from the images in the odometry algorithm. A zooming ratio is also calculated based on the difference in depth between the image frames, which is included in the online estimation of the scale factor and altitude estimation.

During the last decade, a variety of different visual odometry and visual simultaneous localisation and mapping (SLAM) techniques have been presented, whereas several of them are open source packages. A benchmark comparison of publicly-available visual-inertial odometry algorithms have been made in [21]. A lot of these type of comparisons consider high quality visual data which is not applicable in the underwater domain. Even though a lot of the open source packages are supported by impressive demonstrations, the demonstration scenarios are limited to a specific set of conditions. In [22] they make a cross validation of the most promising open source packages by testing them on the datasets provided for each package, and on self produced datasets. These datasets are from both indoor and outdoor, captured by flying, terrestrial and underwater vehicles.

Introduction of vision sensors for control have several advantages compared to classical positioning sensors. E.g. magnetic sensors suffers from a low update rate and are strongly effected when operating close to man-made metallic surfaces. Translation sensors such as accelerometers and DVLs are integrating sensors, hence they are subject to drift and not well suited for station keeping [23]. The use of vision sensors for station keeping is a narrow topic which, to the knowledge of the author, is not widely researched the last couple of years. In order to robustly determine scale or depth of feature coordinates in the image, the methods are either presented with the assumption of planar surfaces [23], or downward looking camera in combination with a depth sensor [24] [25].

Recent years, machine learning has found its way into different computer vision problems. Most applications deals with standard problems like image classification, object detection and segmentation, [26], [27] and [28]. Some research has also been introduced in VO applications, such as [29] and [30], to better predict velocity and direction changes, and estimation of the vehicles trajectory and current pose.

In addition to visual odometry, there are different areas where underwater computer vision is used. An example is Narimani et al. [31], which performs target tracking of an underwater pipeline. Through an image processing method including edge detection, they are able to estimate the angle between the AUV and a pipeline on the seabed. In [32] they present a design of an AUV built to study marine animals by automatically track and follow them. Through image feature matching techniques, different species is identified from instantaneous images captured by the vehicles camera. Li, Dejun et al. [33] present a vision guidance algorithm for terminal docking of an AUV using one camera and one light.

1.1.3 Visual Odometry

By using data from sensors one can estimate changes in position and attitude over time. This is called odometry or egomotion. In mobile robot navigation, one example is measuring the wheel rotation through devices such as rotary encoders. Odometry of any type will always drift due to affecting errors, and the global error will likely diverge as the errors accumulate over time. This results in that no odometry system is stable in any sense. The errors are categorised as either systematic or non-systematic. Examples of systematic errors may be unequal wheel diameters, misaligned wheels, finite encoder resolution and finite encoder sampling rate. The non-systematic errors may occur from traveling on uneven ground or wheel slippage from over-acceleration, fast turning or interacting with external bodies.

For a flying or underwater robot the problem of odometry must be addressed with another approach. Visual odometry is the process of determining equivalent odometry information using only camera images. This is done by estimating the robot's relative motion from one camera image to the next. By assuming that features in the images is stationary, these features can be detected and matched from image to image. The motion of the camera can then be estimated by evaluating the position change of the features. These methods are described in the following sections. Any robot with a sufficiently quality camera can make use of visual odometry. One of the greatest advantages of visual odometry is that no kinematic models of the robot are needed.

1.2 Challenges with underwater imagery

The underwater environment introduces challenges with respect to imagery and computer vision which is not present on land. In many locations, there is little variation to scenery, e.g the sea floor consists manly of sand and does not offer many distinct structures and features which can be used for extracting keypoints.

The level of turbidity can cause trouble wen using cameras underwater. Seawater has generally a lot of different particles in it, resulting in feature based approaches might falsely detect particles as a feature. This can have strong effect on the algorithm's accuracy. Particles moving through the field of view due to current can cause a feature based algorithm to estimate a false translation.

Water absorbs different wavelength at different depths. This results in loss of colours, and the visual appearance of the environment can become monotonous. Colour gradients may be a valuable information source which might not be available underwater. The spectral attenuation is dependent on plankton in the water, coloured dissolved matter and suspended matter [34]. Blues and greens are transmitted quite well in pure water in comparison to red which are attenuated quite strongly. Several implications is caused by this, for example will the range of the camera from the target strongly determine the perceived colour of an object, and the high attenuation of red makes it very difficult to detect at significant ranges.

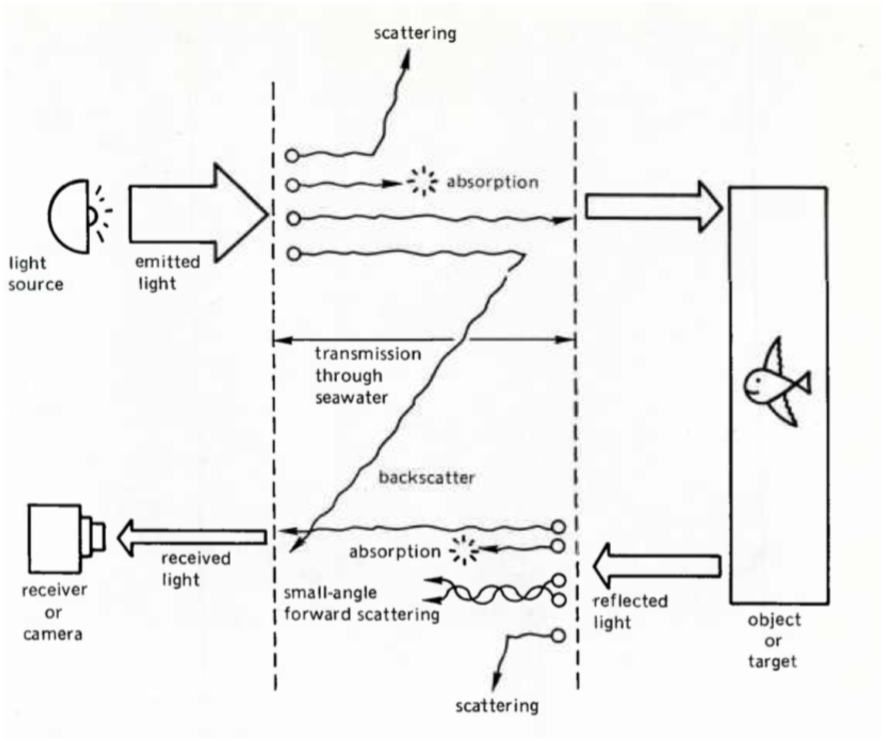

Figure 1.2: Illustration of the losses of light in an underwater imaging system [35]

In the underwater imaging process, artificial light is projected on the scene and its reflections are captured by the camera. The image is made from the registration of photons which are transmitted and attenuated through the water column before the are reflected back to the camera. Due to light-scattering, the light that is forming the imagery is reduced. The photons are scattered and absorbed by the target or the seabed. The small portion which is reflected back to the camera are again subject to scattering and absorption by the seawater and its components. Some of these photons may also be subject to scattering at a small angle, which results in a blurring effect in the image as their trajectory back to the camera is not straight [36]. The different losses of light in an underwater imaging system is illustrated in figure 1.2.

Backscatter is when a portion of the source illumination is reflected into the receiver's field of view. This is due to particles and inhomogeneities in the water. The amount of particles and inhomogeneities are often referred to as turbidity. The image contrast can be strongly degraded because of this.

1.2.1 Open CV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was first presented in 1999. Today the library consists of more than 2500 optimized algorithms, which consist of both classic and state-of-the art computer vision and machine learning algorithms [37]. It has interfaces for C++, Python and Java, and supports the most common operating systems such as Windows, Linux, Mac OS, iOS and Android. It is the most used computer vision library, and its users spans from millions of individuals to well known companies like Google, Yahoo, Microsoft, Intel, IBM etc. [37]. OpenCV has therefore a huge user community on the Internet, which makes it easy to solve upcoming problems with the help of others. The algorithms can be used e.g. to identify objects, perform different adjustments to the image, extract 3D point-clouds from stereo cameras, classify human behaviours, or track an object of interest.

In this thesis, algorithms from the OpenCV library are mostly used for feature detection, description and matching across image frames, and relative pose estimation for monocular visual odometry. In other words, only a small portion of the vast possibilities the library offers is used.

1.2.2 ROS

The framework used for writing software on the Blueye Pinoneer is the Robot Operating System (ROS). This is a flexible framework that with its collection of tools, libraries and conventions simplifies the task of creating complex and robust robot behaviours [38]. Creating robust and well-functioning robot software is hard, because problems that is trivial to us humans might strongly depend of the type of situation and surroundings. These variations is difficult to implement, and as a result ROS was built to encourage collaborative robotics software development. By the contributions of different groups that may specialise in different fields, ROS was designed such that these groups could build upon each others work.

At the lowest level, ROS acts as a communication infrastructure, where it offers an interface for message passing that provides inter-process communication [39]. This middelware, as it is commonly referred to, provides publishing and subscription of message passing, as well as recording and playback of messages. It also provides request and response remote procedure calls and a distributed parameter system. The message system in ROS manages the communication between distributed *nodes* through the publish/subscribe mechanism. The publish/subscribe system is anonymous and asynchronous, which promotes flexibility and modularity in your system. *Nodes* are responsible for the robots functions through calculation, connect and understand hardware data, communication and more. *Nodes* communicate by publishing and subscribe to *topics*. The communication service also supports synchronous request/response interaction through the use of *services*.

On top of the core middelware components, ROS provides common robot-specific libraries and tools, such as standard message definitions, a robot geometry library to manage coordinate frame transforms, a robot description language to describe and model your robot, preemptable remote procedure calls, diagnostics, pose estimation, localisation and mapping.

1.3 Problem formulation

During manually control of the Blueye Pioneer mini ROV, the user may experience lack of control due to currents and drag forces from the umbilical. Depending on the environment, these forces may change continuously. It may also induce motions and oscillations that are hard to compensate for manually. In case of an inspection or exploration the user may want the drone to stand still in respect to a certain object or area of interest. In order to achieve this we are dependent on motion measurements from the vehicle.

In underwater environments there are limited access on positioning data. The challenges of underwater navigation is connected to the limitations of acoustic positioning and dead-reckoning navigation. In order to improve underwater navigation and situational awareness, computer vision techniques are being used. The development within this field have been immense the last decades, which reveal new possibilities for underwater operations. From camera images, motion information can be extracted. This can help to increase the level of automation and autonomy, safety and efficiency in underwater operations.

This thesis will focus on how computer vision can be used to extract motion information from underwater camera images, which are to be used for position estimation of a mini ROV. The approach is more thoroughly presented in the next section.

1.4 Scope and limitations

This thesis will focus on the development and implementation of a monocular visual odometry algorithm for a mini ROV. The MVO approach was chosen as a camera assisted motion estimation of the mini ROV, where the estimates can be used for dynamic positioning. The Blueye Pioneer underwater drone is used as experimental platform, and one of the conditions during development was to only use the sensors already equipped on the vehicle. The implementation is therefore based on a monocular camera system, a MEMS IMU and a pressure sensor for depth measurements. This has led to some computational limitations, especially regarding development of state estimators through fusing of different sensor data. The MVO algorithms are to be used in a real-time application on the Blueye Pioneer. The choice of algorithms and their computational speed therefore had to be considered. The algorithms are developed in Python with the use of the computer vision and machine learning software OpenCV. ROS is the software framework used on the Blueye Pioneer, which provides a great development platform with great flexibility and robustness.

The MVO algorithms are tested on both prerecorded datasets and in the MC-lab. The positioning system software, Qualisys, used for ground truth measurements in the MC-lab had its limitations to operational space in the pool and determination of the vehicles orientation. Measurement drop outs do to movement outside its range, or wrong orientation measurements was therefore frequently experienced. However, the tests presented in this thesis clearly shows the properties of the MVO algorithm.

1.5 Structure of thesis

This thesis is organised in a classical manner with introduction, theory, method, results, discussion and conclusion. The results which contributes to the choice of method is presented in a natural manner during the description of method. The different chapters are organised in the following manner:

- **Chapter 1: Introduction**. This chapter presents the background, discussing ROVs and computer vision, in addition to what has been done in the field of computer vision used for motion estimation. The scope and limitations are also presented.
- **Chapter 2: Theory**. Here, the theory used for development is presented. Modelling and mathematical expressions for the ROV, sensors and camera is presented, as well as a description of the feature detection and descriptor used.
- **Chapter 3: Method**. This chapter presents the used method for developing and testing the implemented MVO algorithm. Choices that have been made along the process with explanatory results is also presented chronologically. A description of the different evaluation methods and test scenarios is presented.
- **Chapter 4: Results**. This chapter presents the results from the different test scenarios of the MVO algorithm.
- **Chapter 5: Discussion**. Discussion of the achieved results compared with the objectives, theory and method used is presented.
- **Chapter 6: Conclusion**. This chapter presents the conclusion with concluding remarks and recommendations for further work.

1.6 Thesis contribution

The main topic of this thesis is to investigate the possibilities of using monocular visual odometry for position estimation, keeping in mind that the position estimates should be used in a dynamic positioning application. The objective for this research is to increase the level of autonomy for ROV operation as well as the level of accuracy for ROV navigation. There has been an extensive research in the use of computer vision for navigation and odometry. However, most of these methods have been applied in areal or surface technologies.

The research conducted in this thesis will contribute to highlight the possibilities and limitations of underwater visual odometry, including feature detection, matching and underwater image processing. The research presented investigates wether a feature-based method can be used to extract motion information in an underwater environment. Many advanced approaches have been developed and presented with promising results. However, a fundamental method have been chosen as a base of development in this thesis.

Computer vision abilities are vital for increasing the level of autonomy and possible operations for ROVs. This thesis can therefore give valuable information and inputs in future development of underwater computer vision applications.

Chapter 2

Theory

2.1 Blueye Pioneer Underwater Drone

The experimental platform used in this thesis is the Blueye Pioneer underwater drone. The drone is developed by the Norwegian based company Blueye Robotics. The company is a spin-off from the Centre of Autonomous Marine Operations and Systems (AMOS) at NTNU [40]. The underwater drone can be classified as a mini ROV, with its small size and low weight. The drone is equipped with four thrusters, two rear, one vertical and one lateral. The lateral thruster makes it ideal for inspection tasks as it makes it possible to keep a constant heading while moving sideways. A full-HD camera is located at the front of the drone, together with a 3300 lumen LED light located below the camera. The drone is also equipped with an IMU, consisting of a three-axis accelerometer, angular velocity sensor and magnetometer, as well as a temperature sensor. Pressure sensors is included to measure both depth and internal pressure. A picture of the drone is shown in figure 2.1, while the main specifications are presented in table 2.1. A detailed list of the different specifications is listed in appendix A.

Table 2.1: Technical specifications

Technical specifications	Value
Dimensions	485 × 257 × 354 mm (L x W x H)
Weight in air	8.6 kg (with salt water ballast)
Maximum rated depth	150 m
Forward speed at normal use	2 m/s (4 knots)
Weight	8.6 kg
Thrusters	4 x 350 W

Figure 2.1: Illustration of the Blueye Pioneer. Photo courtesy of Blueye Robotics

2.2 Notations

The notation used in this thesis is based on the SNAME convention and from Fossen's vectorial model [41]. The ROV operates in six degrees of freedom. The position, orientation and linear and angular velocities are given in generalised coordinates. The notation is given in table 2.2.

Table 2.2: SNAME notation

DOF	Forces and moments	Linear and angular velocity	Positions and Euler angles
1 Surge	X	u	x
2 Sway	Y	v	y
3 Surge	Z	w	z
4 Roll	K	p	ϕ
5 Pitch	M	q	θ
6 Yaw	N	r	ϕ

Generalised coordinates

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

$$\boldsymbol{\eta} = \left[x,\ \ y,\ \ z,\ \ \phi,\ \ \theta,\ \ \psi\right]^\top \tag{2.1}$$

$$\boldsymbol{\nu} = \left[u,\ \ v,\ \ w,\ \ p,\ \ q,\ \ r\right]^\top \tag{2.2}$$

Sub-vectors from (2.1) and (2.2) are defined to express linear and angular position and velocity vectors which are given by (2.3)

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

where $\mathbf{p} \in \mathbb{R}^{3\times 1}$ is the linear position, $\mathbf{v} \in \mathbb{R}^{3\times 1}$ is the linear velocity, $\boldsymbol{\Theta} \in \mathbb{R}^{3\times 1}$ is the attitude or angular position, and $\boldsymbol{\omega} \in \mathbb{R}^{3\times 1}$ is the angular velocity.

Notation norms

All matrices are expressed in boldface uppercase letters and vectors are expressed in boldface lowercase letters. An estimate of the variable x is expressed as $\hat{x}$, while the time derivative of x is denoted as $\dot{x}$. The time derivative of the estimate is expressed as $\dot{\hat{x}}$. The error, defined as the difference between the estimate of a variable and its true value, is expressed as $\tilde{x}$.

The skew-symmetric matrix $\mathbf{S}$ can be used as cross-product operator to calculate the cross product, $\times$, between two vectors $\mathbf{a}$ and $\mathbf{b} \in \mathbb{R}^{3\times 1}$:

$$\mathbf{a} \times \mathbf{x} := \mathbf{S}(\mathbf{a})\mathbf{b} \tag{2.4}$$

where $\mathbf{S}$, with the skew symmetric properties, is defined as

$$\mathbf{S}(\boldsymbol{\lambda}) = -\mathbf{S}^\top(\boldsymbol{\lambda}) = \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix}, \quad \boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} \tag{2.5}$$

2.3 Kinematics

2.3.1 Reference frames

When analysing the motion of a marine craft, it is convenient to define it with respect to a reference frame. There are different reference frames used for navigational purposes, and the most common ones are

- ECI: The Earth-centred internal frame $\{i\} = (x_i, y_i, z_i)$. The origin o_i is located at the Earth's centre with axes as shown in figure 2.2
- ECEF: The Earth-centred Earth-fixed frame $\{e\} = (x_e, y_e, z_e)$. This also has its origin o_e located the Earths centre, but its axes rotates with angular rate ω_e relative to the ECI frame, which is fixed in space.

- NED: North-East-Down frame $\{n\} = (x_n, y_n, z_n)$. The origin o_n is defined relative to Earth's reference ellipsoid [41]. This is the reference frame we normally refer to.
- BODY: The body-fixed reference frame $\{b\} = (x_b, y_b, z_b)$. This coordinate frame is fixed to the vehicle, which means that it moves together with the vehicle, and its origin o_b and orientation is described relative to the inertial reference frame.
- MES: The measurement frame $\{m\} = (x_m, y_m, z_m)$. This frame is usually moving and rotating with the body frame. The measurements given from an instrument on the vehicle is expressed in this frame.

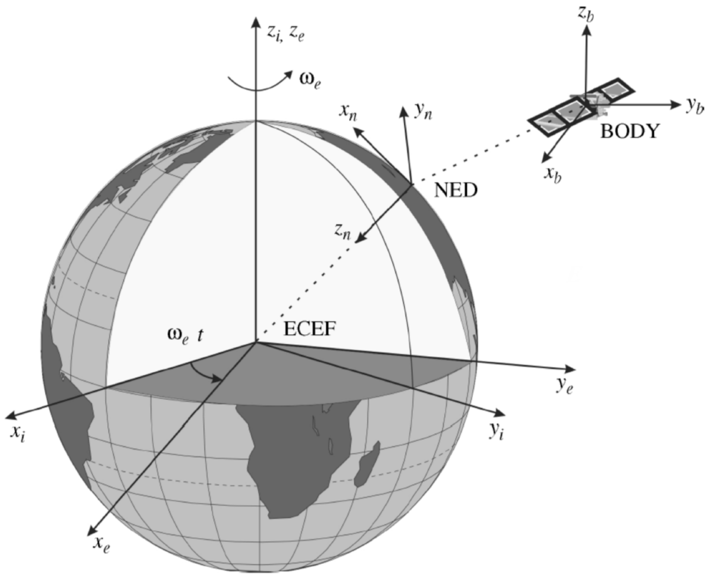

Figure 2.2: The reference frames with respect to the inertial frame (ECI), [41]

Dependent on accuracy, both $\{e\}$ and $\{n\}$ can be assumed inertial. For slowly moving vehicles, and marine crafts operating in a local area with approximately constant longitude and latitude, $\{n\}$ can be assumed inertial. If moving far away from the origin o_e, the flat earth approximation in $\{n\}$ will not be applicable. In this case $\{e\}$ can be used.

For the Blueye Pioneer, the body frame $\{b\}$ is defined as shown in figure 2.3

2.3.2 Transformations

A vector that is given in one frame, can be expressed in another using a transformation matrix. Sub and superscripts are used to determine which reference frame a vector is decomposed in. The following notation is used when transforming a vector from one coordinate frame to

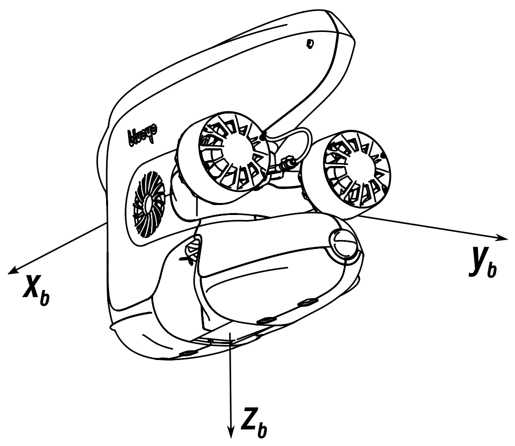

Figure 2.3: Definition of body frame on Blueye Pioneer

another.

$$\mathbf{v}^{\text{to}} = \mathbf{R}^{\text{to}}_{\text{from}} \mathbf{v}^{\text{from}} \tag{2.6}$$

$\mathbf{v}^{\text{from}}$ denotes a vector that is transformed to a new reference frame by applying the rotation matrix $\mathbf{R}^{\text{to}}_{\text{from}}$. The result is the vector $\mathbf{v}^{\text{to}}$.

E.g. in order to decompose the body-fixed velocity vector in NED reference frame, we make use of a rotation matrix $\mathbf{R}^n_b(\mathbf{\Theta}_{nb})$. Here, $\mathbf{\Theta}_{nb}$ is the rotation vector with the Euler angles roll (ϕ), pitch (θ), and yaw (ψ) as argument, $\mathbf{\Theta}_{nb} = [\phi, \theta, \psi]^\top$. The notation with subscript nb means that the coordinates are given in frame $\{n\}$ with respect to frame $\{b\}$. The transformation matrix is given as

$$\mathbf{R}^n_b(\mathbf{\Theta}_{nb}) = \mathbf{R}_{z,\psi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\phi} \tag{2.7}$$

where

$$\mathbf{R}_{x,\phi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix}, \quad \mathbf{R}_{y,\theta} = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix}, \quad \mathbf{R}_{z,\psi} = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

here $s\cdot$ and $c\cdot$ are short for sin(·) and cos(·) respectively. The inverse transformation is given as

$$\mathbf{R}^n_b(\mathbf{\Theta}_{nb})^{-1} = \mathbf{R}^b_n(\mathbf{\Theta}_{nb}) = \mathbf{R}^\top_{x,\phi}\mathbf{R}^\top_{y,\theta}\mathbf{R}^\top_{z,\phi} \tag{2.9}$$

One can see from (2.8) that this representation gives a singularity at $\phi = 90°$. One can therefore make use of quaternions to avoid this.

With the assumption that ψ and θ is small, which is a good approximation for underwater vehicles [41], we can write $\mathbf{R}^n_b(\Theta_{nb}) \approx \mathbf{R}_{z,\phi}$

Quaternions

An alternative to Euler angle representation is quaternions. This is a four-parameter method where a quaternion $\mathbf{q}$ is defined as a complex number with one real part η and three imaginary parts given by the vector

$$\boldsymbol{\varepsilon} = [\varepsilon_1, \varepsilon_2, \varepsilon_3]^\top \tag{2.10}$$

Quaternions can be used in order to prevent representation singularity of the Euler angels. They are usually given in unit form, called unit quaternions, which then satisfies $\mathbf{q}^\top\mathbf{q} = 1$. A quaternion is expressed on the form

$$\mathbf{q} = \begin{bmatrix} \eta \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{bmatrix} \tag{2.11}$$

If the Euler angles Θ_{nb} are known, the corresponding unit quaternions can be computed as follows

$$\mathbf{q} = \begin{bmatrix} \cos(\psi/2)\cos(\theta/2)\cos(\phi/2) + \sin(\psi/2)\sin(\theta/2)\sin(\phi/2) \\ \cos(\psi/2)\cos(\theta/2)\sin(\phi/2) - \sin(\psi/2)\sin(\theta/2)\cos(\phi/2) \\ \sin(\psi/2)\cos(\theta/2)\sin(\phi/2) + \cos(\psi/2)\sin(\theta/2)\cos(\phi/2) \\ \sin(\psi/2)\cos(\theta/2)\cos(\phi/2) - \cos(\psi/2)\sin(\theta/2)\sin(\phi/2) \end{bmatrix} \tag{2.12}$$

Translation of e.g. a linear velocity vector from an inertial reference frame to a body fixed reference frame can be expressed as

$$\dot{\mathbf{p}}^n_{b/n} = \mathbf{R}^n_b(\mathbf{q})\mathbf{v}^b_{b/n} \tag{2.13}$$

where the rotation matrix $\mathbf{R}^n_b(\mathbf{q})$ is given as

$$\mathbf{R}^b_n(\mathbf{q}) = \begin{bmatrix} 1-2(\varepsilon_2^2+\varepsilon_3^2) & 2(\varepsilon_1\varepsilon_2-\varepsilon_3\eta) & 2(\varepsilon_1\varepsilon_3+\varepsilon_2\eta) \\ 2(\varepsilon_1\varepsilon_2+\varepsilon_3\eta) & 1-2(\varepsilon_1^2+\varepsilon_3^2) & 2(\varepsilon_2\varepsilon_3-\varepsilon_1\eta) \\ 2(\varepsilon_1\varepsilon_3-\varepsilon_2\eta) & 2(\varepsilon_2\varepsilon_3+\varepsilon_1\eta) & 1-2(\varepsilon_1^2+\varepsilon_2^2) \end{bmatrix} \tag{2.14}$$

2.3.3 Thrust allocation

The thruster layot on the Blueye Pioneer is as presented in figure 2.4. The thrusters on the drone is fixed, meaning that all control forces are produced by thrusters in fixed directions. The control force inputs are given in surge, sway, heave and yaw, which is the same number as degrees of freedom for the drone, meaning it is possible to find an "optimal" distribution of control forces $\mathbf{f}$ for each DOF by using an explicit method [41].

The thrust allocation implemented allocates the given vehicle force in body frame to a dshot signal, representing a desired thrust from each thruster. The thrust allocation consist of a set of lever arms and direction of force for each thruster as a matrice **B**. The force to thrust relation is given as

$$\begin{aligned} \boldsymbol{\tau}_d &= \mathbf{B}\mathbf{f} \\ \Rightarrow \mathbf{f} &= \mathbf{B}^{-1}\boldsymbol{\tau}_d \end{aligned} \tag{2.15}$$

Where the matrix **B** and the desired thrust $\boldsymbol{\tau}_d$ is given as

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -l_{y,star} & -l_{y,port} & l_{x,sway} & 0 \end{bmatrix} \qquad \boldsymbol{\tau}_d = \begin{bmatrix} X \\ Y \\ Z \\ N \end{bmatrix} \tag{2.16}$$

The constants $l_{y,star}$, $l_{y,port}$ and $l_{x,sway}$ specify the distance from the centre of origin in body frame to each thruster.

The thrust force is linearised by the third degree polynomial to approximate resulting force at a specific thruster permil. The polynomial constants are found by fitting the loadcell data to the polynomial. There are different constants for positive and negative physical directions.

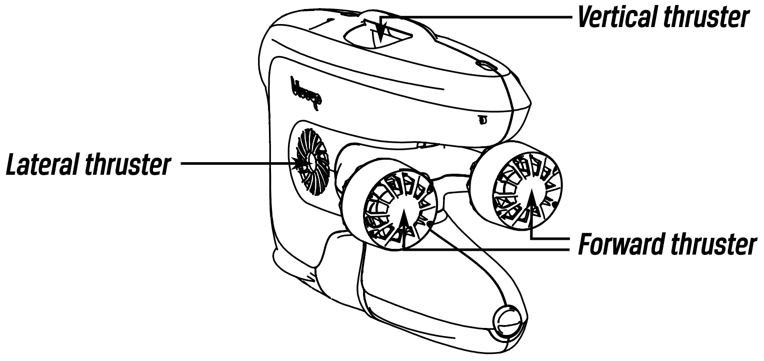

Figure 2.4: Thruster setup on the Blueye Pioneer. Photo courtesy of Blueye Robotics

2.4 Sensors

2.4.1 Inertial Measurement Unit

The Blueye Pioneer is equipped with an inertial measurement unit (IMU) consisting of a three-axis accelerometer, three-axis gyroscope and a three-axis magnetometer measuring 3-DOF acceleration, angular rate and magnetic field components. Thanks to a significant reduction in price during the last decades, inertial measurement technology is available for commercial use. This, together with the reduction in size makes them well suited for smaller vehicles such as mini ROVs [42].

The IMU has an update rate normally ranging from 100 - 1000 Hz. This is a higher update rate than other positioning sensors such as the doppler velocity log (DVL) or acoustic positioning systems (APS). Limitations in MEMS technology results in that the IMU still suffer from bias instability, noisy output and drift. In order to compensate for these effects, calibration is a necessary step prior to application of appliance.

The measurement equations for the three-axis accelerometer, gyro and magnetometer are given in (2.17), (2.18), (2.19)

$$\mathbf{a}^b_{\text{imu}} = \mathbf{R}^b_n(\Theta)\left(\dot{\mathbf{v}}^n_{m/n} - \mathbf{g}^n\right) + \mathbf{b}^b_{\text{acc}} + \mathbf{n}^b_{\text{acc}} \tag{2.17}$$

$$\boldsymbol{\omega}^b_{\text{imu}} = \boldsymbol{\omega}^b_{m/n} + \mathbf{b}^b_{\text{gyro}} + \mathbf{n}^b_{\text{gyro}} \tag{2.18}$$

$$\mathbf{m}^b_{\text{imu}} = \mathbf{R}^b_n(\Theta)\,\mathbf{m}^n + \mathbf{b}^b_{\text{mag}} + \mathbf{n}^b_{\text{mag}} \tag{2.19}$$

where $\mathbf{a}^b_{\text{imu}} \in \mathbb{R}^{3\times1}$ is the measured acceleration vector, $\boldsymbol{\omega}^b_{\text{imu}} \in \mathbb{R}^{3\times1}$ is the turn rate vector from the gyroscope, and $\mathbf{m}^b_{\text{imu}} \in \mathbb{R}^{3\times1}$ is the magnetic field components. The accelerometer and gyro biases are denoted as $\mathbf{b}^b_{\text{acc}}$ and $\mathbf{b}^b_{\text{gyro}}$ while $\mathbf{b}^b_{\text{mag}}$ is the local magnetic disturbance. $\mathbf{n}^b_{\text{acc}}$, $\mathbf{n}^b_{\text{gyro}}$ and $\mathbf{n}^b_{\text{mag}}$ are additive zero-mean sensor measurement noise.

In order to perform more accurate visual odometry, sensor readings from an IMU is often included in some sort of way. We then characterise it as a visual-inertial odometry. Combining inertial and visual measurements has become popular as they offer complementary characteristics. Traditionally, sensor reading from the IMU is fused with the estimates from the visual odometry algorithm through filtering, like the work presented in [43] and [44]. Now, optimisation based methods are on the rise, such as [45] and [46]. A prerequisite for many of these applications is that they need a pre-calibrated IMU. If not, they need to perform an online self-calibration. Bonin-Font et al. presents in [43] an online calibration method done through a multiplicative error-state Kalman Filter (MESKF). A calibration is needed in order to compensate for drift and biases in the sensor measurements. The drift is accumulated through the samples and generates an important deviation of the final pose estimation integrated from the INS measurements with respect to the ground truth. Some manufacturers provide an estimation of these systematic errors, assuming they are constant. But in practice they are not. Consequently, it turns out to be very difficult to compensate for their effect [43].

2.4.2 Pressure Sensor

The Blueye Pioneer is equipped with a pressure sensor which makes it able to calculate the vehicles current depth based on a measured pressure. The measurements from the sensor gives depth down to millimetre precision, which makes it suitable to use as an additional input in the odometry algorithm. The update rate is approximately 40 Hz. This is superior compared to the update rate in the implemented MVO algorithm, which is 7-10 Hz.

2.5 Error state Kalman Filter (ESKF)

The error state formulation of the Kalman filters deal with nominal variables and their errors. Both ESKF and EKF should lead to the similar results, but the ESKF generates lower values for the covariance, and the error variables are better at being represented as linear as the their variations are much slower than changes on the nominal navigation data. [43].

The EKF state representation is supported by the fact that the estimated state vector, $\tilde{\mathbf{x}}$ at time instant k consists of the true state, $\mathbf{x}$, and an error, $\delta\mathbf{x}$:

$$\tilde{\mathbf{x}}_k = \mathbf{x}_k + \delta\mathbf{x}_k \tag{2.20}$$

The nominal and error state vector are involved in the process. The nominal state vector $\mathbf{x}_k$ at time k is expressed as

$$\mathbf{x}_k = \left[\mathbf{p}_k, \mathbf{q}_k, \mathbf{v}_k, \boldsymbol{\omega}_k, \mathbf{a}_k, \mathbf{d}_k \mathbf{b}_k\right]^\top \tag{2.21}$$

where $\mathbf{p}_k$ is the position of the ROV with respect to the inertial global frame, $\mathbf{q}_k$ is the orientation expressed in quaternions, $\mathbf{v}_k$ is the linear velocity, $\boldsymbol{\omega}_k$ is the angular rate, $\mathbf{a}_k$ is the acceleration, $\mathbf{d}_k$ is the the gyroscope drift and $\mathbf{b}_k$ is the accelerometer bias. The biases in the accelerometer and the gyroscope are assumed to be constant. It is also assumed that the nominal states does not include noise or model imperfections.

The error state vector at time k, $\delta\mathbf{x}_k$, including the errors caused by sensor biases and random noise is denoted as:

$$\delta\mathbf{x}_k = \left[\delta\mathbf{p}_k, \delta\mathbf{q}_k, \delta\mathbf{v}_k, \delta\boldsymbol{\omega}_k, \delta\mathbf{a}_k\right]^\top \tag{2.22}$$

where $\delta\mathbf{p}_k$ is the error in position, $\delta\mathbf{q}_k$ is the error in attitude in the form of a rotation vector, $\delta\mathbf{v}_k$ is the error in linear velocity, $\delta\boldsymbol{\omega}_k$ is the error in angular rate, and $\delta\mathbf{a}_k$ is the error in linear local acceleration. The sensor bias and noise in acceleration and angular rate are included in the errors.

Nominal State Prediction

The sensor readings from the inertial sensors are used at this stage to estimate the relative displacement of the vehicle between two filter iterations. These measurements contain a bias and a noise, assumed to be additive and zero-mean Gaussian [43]. By assuming that the inertial sensor is located approximately in the centre of the vehicle, and that the centripetal acceleration is negligible, the uncalibrated acceleration measurements can be given as

$$\mathbf{a}_m = (\mathbf{a} - \mathbf{g}^b) + \mathbf{b} + \mathbf{n}_a \tag{2.23}$$

Here, $\mathbf{a}_m$ is the measured acceleration, $\mathbf{a}$ is the real body acceleration, $\mathbf{g}^b$ is the gravity expressed in the body frame, $\mathbf{b}$ is the accelerometer bias and $\mathbf{n}_a$ is the measurement noise.

The discrete equations that govern the nominal states are given as:

$$\hat{\mathbf{p}}_k = \hat{\mathbf{p}}_{(k-1)} + \mathbf{R}_{(k-1)}\hat{\mathbf{v}}_{(k-1)}\Delta t + \frac{1}{2}\mathbf{R}_{(k-1)}\hat{\mathbf{a}}_{(k-1)}\Delta t^2 \tag{2.24a}$$

$$\hat{\mathbf{q}}_k = (\mathbf{q}_m * \hat{\mathbf{q}})_{(k-1)} \tag{2.24b}$$

$$\hat{\mathbf{v}}_k = \hat{\mathbf{v}}_{(k-1)} + \hat{\mathbf{a}}_{(k-1)}\Delta t \tag{2.24c}$$

$$\hat{\boldsymbol{\omega}}_k = (\boldsymbol{\omega}_m - \hat{\mathbf{d}})_{(k-1)} \tag{2.24d}$$

$$\hat{\mathbf{a}}_k = (\hat{\mathbf{b}} - \mathbf{g}^b - \mathbf{a}_m)_{(k-1)} \tag{2.24e}$$

$$\hat{\mathbf{d}}_k = \hat{\mathbf{d}}_{(k-1)} \tag{2.24f}$$

$$\hat{\mathbf{b}}_k = \hat{\mathbf{b}}_{(k-1)} \tag{2.24g}$$

where $\mathbf{R}$ is the rotation matrix from local to global coordinates, and Δt is the time step interval between the current and predicted nominal state. The sign, $*$, denotes the quaternion product.

Error State Prediction

The error is defined as a difference between the real value of a variable and its estimate. The estimation of the error state vector at time k is defined as:

$$\delta\tilde{\mathbf{x}}_k = f(\mathbf{x}_{k-1}) + \boldsymbol{\epsilon}_k \tag{2.25}$$

where $f(\mathbf{x}_k)$ is the set of functions that predicts the error state (2.26). The error of the prediction model is modelled as a zero-mean Gaussian, $\boldsymbol{\epsilon}_k = N(0, Q_k)$, where Q_k is the model noise covariance.

The estimated error states are given by the following discrete equations:

$$\begin{aligned}\delta\tilde{\mathbf{p}}_k = \delta\tilde{\mathbf{p}}_{(k-1)} + \mathbf{R}\delta\tilde{\mathbf{v}}_{(k-1)}\Delta t - \mathbf{R}\left(\tilde{\mathbf{v}}_{(k-1)} \times \delta\tilde{\mathbf{q}}_{(k-1)}\right)\Delta t \\ - \frac{1}{2}\mathbf{R}\left(\tilde{\mathbf{a}}_{(k-1)} \times \delta\tilde{\mathbf{q}}_{(k-1)}\right)\Delta t^2 + \frac{1}{2}\mathbf{R}\delta\tilde{\mathbf{a}}_{(k-1)}\Delta t^2\end{aligned} \tag{2.26a}$$

$$\delta\tilde{\mathbf{v}}_k = \delta\tilde{\mathbf{v}}_{(k-1)} + \mathbf{g}^b \times \delta\tilde{\mathbf{q}}_{(k-1)}\Delta t + \delta\tilde{\mathbf{a}}_{(k-1)}\Delta t \tag{2.26b}$$

$$\delta\tilde{\boldsymbol{\omega}}_k = \delta\tilde{\boldsymbol{\omega}}_{(k-1)} \tag{2.26c}$$

$$\delta\tilde{\mathbf{a}}_k = \delta\tilde{\mathbf{a}}_{(k-1)} \tag{2.26d}$$

$$\begin{aligned}\delta\tilde{\mathbf{q}}_k = \mathbf{R}_m\delta\tilde{\mathbf{q}}_{(k-1)} + \Big(-\mathbf{I}\Delta t + \frac{1-\cos\left(|\boldsymbol{\omega}_{(k-1)}|\Delta t\right)}{|\boldsymbol{\omega}_{(k-1)}|^2}\mathbf{V} \\ - \frac{|\boldsymbol{\omega}_{(k-1)}|\Delta t - \sin\left(|\boldsymbol{\omega}_{(k-1)}|\Delta t\right)}{|\boldsymbol{\omega}_{(k-1)}|^3}\mathbf{V}^2\Big)\delta\tilde{\boldsymbol{\omega}}_{(k-1)}\end{aligned} \tag{2.26e}$$

Since the values forming the error states are all reset to zero after the nominal state correction, the values computed by equations (2.26) are always zero. However, the system covariance matrix $\mathbf{P}_k$ propagates according to the Kalman filter equation, since the Jacobian matrix $\mathbf{F}_k = \frac{\partial f(\mathbf{x}_k)}{\partial\delta\mathbf{x}_k}|_{\delta\tilde{\mathbf{x}}_k}$, at time k is different from zero.

Error Kalman Update

The measurement error $\delta\mathbf{z}_k$ is denoted as the difference between a sensor reading $\mathbf{z}_k$ and its prediction $\hat{\mathbf{z}}_k$ computed in the nominal state:

$$\delta\mathbf{z}_k = \mathbf{z}_k - \hat{\mathbf{z}}_k \tag{2.27}$$

The classical Kalman Filter equations are used to calculate the updated error state vector $\delta\tilde{\mathbf{x}}_{k|k}$ and its covariance:

$$\begin{aligned}\tilde{\mathbf{y}}_k &= \delta\mathbf{z}_k - \mathbf{H}_k\delta\tilde{\mathbf{x}}_k \\ \mathbf{K}_k &= \bar{\mathbf{P}}_k\mathbf{H}_k^\top\left(\mathbf{H}_k\bar{\mathbf{P}}_k\mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1} \\ \delta\tilde{\mathbf{x}}_{k|k} &= \delta\tilde{\mathbf{x}}_k + \mathbf{K}\tilde{\mathbf{y}}_k \\ \hat{\mathbf{P}}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\bar{\mathbf{P}}_k \\ \bar{\mathbf{P}}_{(k+1)} &= \mathbf{F}_k\hat{\mathbf{P}}_k\mathbf{F}_k^\top + \mathbf{Q}\end{aligned} \tag{2.28}$$

The nominal state vector is then corrected with the values from the updated error state vector.

2.6 Camera Model

The process of image formation, whether it is in a camera or an eye, involves a projection from the three-dimensional world onto a two-dimensional surface. Camera models is used to describe the mathematical relationship between three-dimensional coordinates in space and its projection onto a two-dimensional image plane. Different camera models exists for different camera types, wether it is a wide field-of-view camera equipped with a fisheye lens, a catadioptric camera which consists of a camera lens and a mirror two broaden its field of view, or a more normal type of camera used as a web camera or the ones equipped in a smartphone. One of the widely used mathematical models in computer vision which describes the perspective transformation is the pinhole camera model.

2.6.1 Pinhole Camera Model

The central perspective imagine model, also known as the pinhole camera model, shown in figure 2.5 is common to use in computer vision. The lighting rays converge at the origin of the camera frame C, resulting in inverted image is projected onto the image plane at $z = -f$. The camera is modelled as a box, with a small hole on one side and a photographic plate on the opposite side. The model is centred around the optical axis, which coincides with the z-axis of the coordinate frame which is located at the centre of projection, known as the focal point C, or principle point.

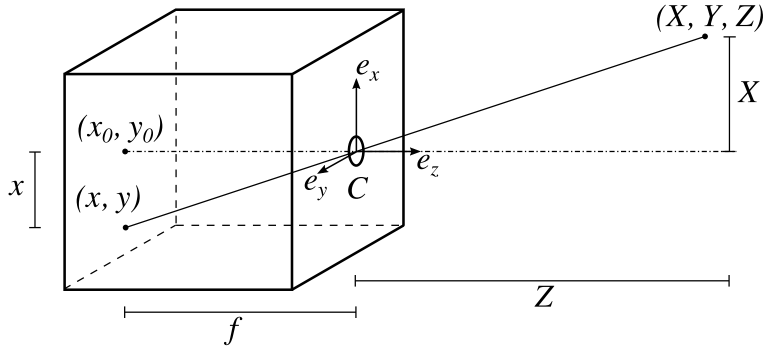

Figure 2.5: Pinhole camera model

The distance from the image plane to the camera centre is the focal length f. Geometrical relations gives:

$$\frac{x}{f} = \frac{X}{Z} \quad \text{and} \quad \frac{y}{f} = \frac{Y}{Z} \tag{2.29}$$

In computer vision we make use of homogenous coordinates. This is done by introducing one more dimension to the coordinate system. For a point in a 2D image frame, the point will have three coordinates, and for a point in a 3D camera frame, the point will have four coordinates. In computer vision, the extra coordinate will have the value of one. Mapping a 2D point from $(x, y) \to (x, y, 1)$ defines an inclusion from the Euclidean plane to the projective plane. Writing equation (2.29) in matrix form with the use of homogenous coordinates, we get:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.30}$$

where the depth, λ, is equal to Z. By introducing the notation

$$\mathbf{K} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.31}$$

in equation (2.30), we get

$$\lambda \mathbf{x} = \mathbf{K}[\mathbf{I}_{3x3}|\mathbf{0}_{3x1}]\mathbf{X} = \mathbf{P}\mathbf{X} \tag{2.32}$$

where $\mathbf{P} = \mathbf{K}[\mathbf{I}_{3x3}|\mathbf{0}_{3x1}]$. In this manner, the matrix $\mathbf{P}$ relates extended image coordinates $\mathbf{x} = (x, y, 1)$ to extended object coordinates $\mathbf{X} = (X, Y, Z, 1)$ through the equation

$$\lambda \mathbf{x} = \mathbf{P}\mathbf{X} \tag{2.33}$$

The matrix $\mathbf{P}$ is called the camera matrix, and equation (2.33) is called the camera equation [47]. In a redefined camera model, the matrix $\mathbf{K}$ is replaced by

$$\mathbf{K} = \begin{bmatrix} f & sf & c_x \\ 0 & \alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.34}$$

This matrix consists of the intrinsic parameters, as they model the intrinsic properties of the matrix. f is the focal length, α is the aspect ratio, s is the skew parameter, and c_x and c_y is the principle point coordinates. For most cameras, the skew parameter is close to zero ($s \approx 0$) and the aspect ratio is close to one ($\alpha \approx 1$). The focal length can also be given for both axes as f_x and f_y where the relation $f_x = \alpha f_y$ applies.

By studying equation (2.30), we see that in order to determine the distance to a 3D point we need two pinhole cameras, since every image point is the projection of all infinite 3D points lying on the ray passing through the same image point and the centre of projection, C, in figure

2.5.

Since an image is a two-dimensional projection of a three dimensional scene, depth recovery from a single image, without additional knowledge of the scene, is mathematically impossible. For an image sequence of a rigid three dimensional scene taken from a single moving camera, it is possible to compute the motion up to a scale factor. In order to determine the 3D position for a feature, the moving camera must observe it repeatedly each time, capturing a ray of light from the feature to its optic centre. The angle between the captured rays from different viewpoints is the feature's parallax. This parallax is what allows it's depth to be estimated [48].

2.6.2 Digital image representation

A digital image is represented through a matrix where each cell corresponds to a pixel. Each pixel usually consists of three values defining its colour. A video with a resolution of 1920x1080 pixels and a frame rate of 30fps therefore consist of 6,220,800 values 30 times pr second, which clearly explains why computational time is strongly effected by the resolution. Optimised algorithms are therefore critical, and before the algorithms are applied, the images are usually downsized or cropped. Through manipulation of the different values in the image matrix, specific tasks can be performed, such as masking out a feature, blur the image, change the contrast or perform edge detection. This can be done through kernels or different filter operations.

Kernels is simply small matrices, and the wanted effect is accomplished through convolution between the kernel and the image. It is related to a form of the mathematical convolution, where each element of the image is added to its local neighbours, weighted by the kernel. For example sharpening the image can be done through a kernel given as:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{2.35}$$

The sharpening effect is shown in figure 2.6

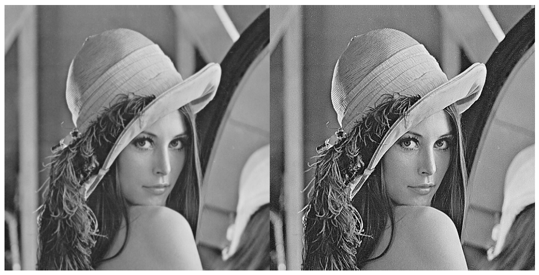

Figure 2.6: The resulting effect of applying a sharpening kernel as given in (2.35). Before (left) and after (right)

The pixel is defined by its colour values, and there are many different colour models used in order to describe the colour. For computer- and TV-displays RGB and HSV are used. The RGB model is an additive colour model that consists of the three colours red, green and blue. The amount of each colour is defined by a number between 0 and 255. In this manner a broad array of colours can be produced. In OpenCV, the BGR colour definition is used. It is the same as RGB, only the numbers defining the resulting colour is given as blue, green and red instead of red, blue and green, respectively. A visualisation of the RGB representation is given in figure 2.7. The HSV (Hue Saturation Value) model is an alternative representation to the RGB colour model, designed to be more representative to the way humans perceive colours. Hue determines the main colour, saturation the intensity and value the brightness.

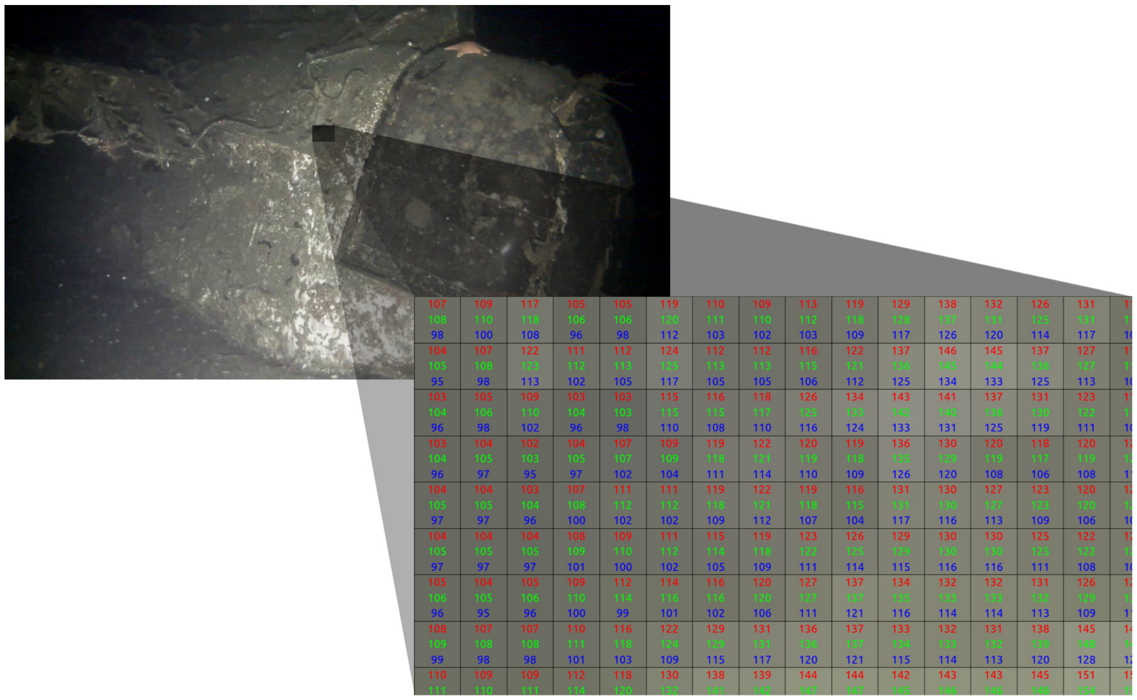

Figure 2.7: RGB representation. A pixel intensity is represented by three values which describe its colour.

2.6.3 Distortion

Due to imperfections in lenses, a variety of distortions is introduced. This imperfections can be geometric distortions causing points on the image plane to be displaced from where they should be, spherical aberration or astigmatism (variation of focus across the scene) or chromatic aberration (colour fringing).

Geometrical distortion are the major problems we encounter in robotic applications [6]. The main types of geometrical distortion is radial distortion and tangential distortion. Radial distortion causes straight lines to appear curved, and the distortion becomes larger the farther points are from the centre of the image. The radial error can be approximated by a polynomial

$$\delta r = k_1 r^3 + k_2 r^5 + k_3 r^7 + \dots \tag{2.36}$$

where r is the distance of the image point from the principle point.

A point with coordinates (u, v) after distortion is given by

$$u^d = u + \delta_u, \qquad v^d = v + \delta_v \tag{2.37}$$

where the displacement is

$$\begin{bmatrix} \delta_u \\ \delta_v \end{bmatrix} = \underbrace{\begin{bmatrix} u\left(k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots\right) \\ v\left(k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots\right) \end{bmatrix}}_{\text{radial}} + \underbrace{\begin{bmatrix} 2p_1 uv + p_2\left(r^2 + 2u^2\right) \\ p_1\left(r^2 + 2v^2\right) + 2p_2 uv \end{bmatrix}}_{\text{tangential}} \tag{2.38}$$

Two common types of radial distortion are positive and negative distortion, also known as barrel distortion and pincushion distortion [6]. Barrel distortion causes lines at the edge of the image to curve outwards which occurs when magnification decreases with distance from the principle point. The opposite effect, when magnification increases with distance from the principle point, causes pincushion distortion, which results in that straight lines curves inward. The two distortion effects is shown in figure 2.8.

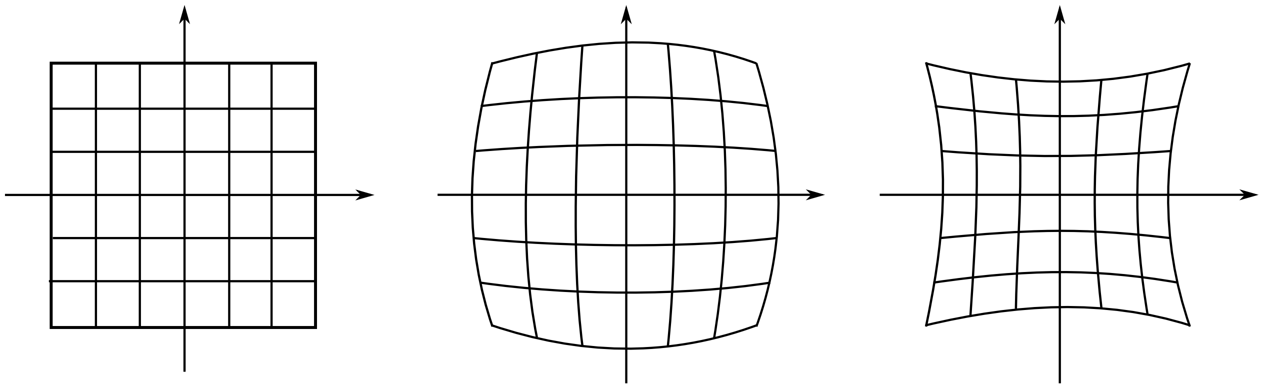

Figure 2.8: Distortion effects. Barrel distortion (middle) and pincushion distortion (right).

The distortion coefficients are parameterised by $(k1, k2, p1, p2, k3)$ which are considered additional intrinsic parameters.

2.7 Feature detection and description

In order to perform motion estimation, 3D reconstruction, image mosaicking etc, we need interest points in the image that can help us compare different images. An interest point can be an edge, a corner, or a local intensity maximum or minimum. An important factor is that interest points needs to be distinguishable, meaning that they can be recognised in different lighting, viewpoint or orientation. For example in the field of object detection and tracking is this important as the features must be recognised in uncorrelated images.

In able to compare features from one image with the next, each feature is given a descriptor describing the feature. The descriptor is usually a vector of N unique numbers. Two features can then be matched if their corresponding descriptors are sufficiently similar.

There are may different feature detection algorithms and descriptors developed. The different methods are developed for different applications and have therefore different advantages. When comparing different feature detection algorithms the main criteria is speed and accuracy. For online applications speed is an important factor, whereas for processing application accuracy may be more critical. For navigation, online feature detection algorithms is used. In order to get up-to-date information about the vehicles orientation and position, the algorithm needs to be fast [13] [49]. For post processing such as mapping, high accuracy is more important than speed in order to get a good result.

2.7.1 ORB (Oriented FAST and Rotated BRIEF)

ORB (Oriented FAST and Rotated Brief) [50] is a binary descriptor based on BRIEF [51] and FAST [26].

FAST keypoint detector

The FAST (Features form Accelerated Segment Test) keypoint detector [26] operates by considering a circle of sixteen pixels around the corner candidate p. If there exists a set of n contiguous pixels in the circle around p which are all brighter than the intensity of the candidate pixel I_p, plus a threshold t, or darker than $I_p - t$, p is classified as a corner. This is illustrated in figure 2.9. Instead of checking 12 contiguous corner, the FAST implementation used in ORB checks 9. Hence the name FAST-9. This algorithm shows good performance [26], [50].

Since FAST does not produce a measure of cornerness, it has large responses along edges. A Harris corner measure [10] is therefore applied to order the FAST keypoints. To find a target number N of keypoints, the threshold is set low enough to get more than N keypoints. The keypoints are then ordered according to the Harris measure, and the top N points are picked.

A scale pyramid of the image is employed where FAST features filtered by Harris measure is produced at each level of the pyramid. This is done since FAST does not produce multi-scale

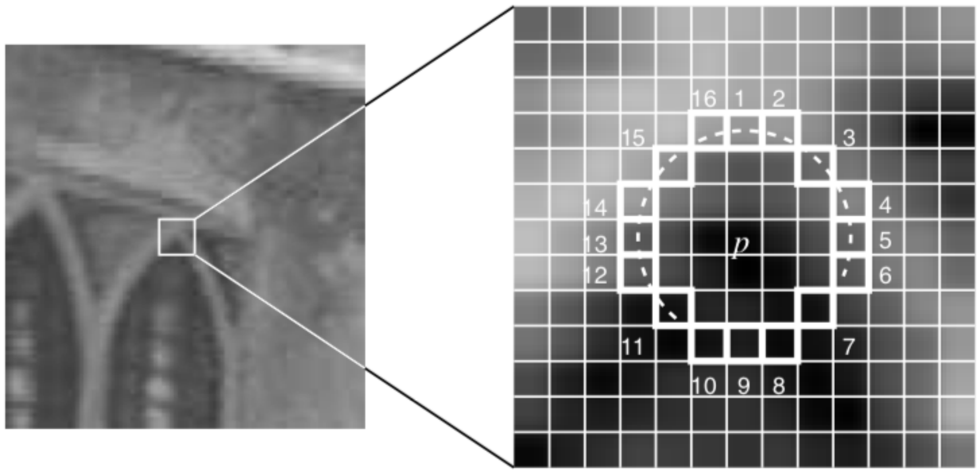

Figure 2.9: Corner detection test in an image patch. The highlighted squares are the pixels used in the corner detection. [26]

features.

FAST features also does not have an orientation component. An effective measure of corner orientation is applied, based on the intensity centroid technique by Rosin [52]. This technique assumes that a corner's intensity is offset from its centre, and by finding the vector from the corners centre, **o**, to the centroid **c**, the orientation of the patch can be determined. The moment of the patch is defined by Rosin as:

$$m_{pq} = \sum_{x,y} x^p, y^p I(x,y) \tag{2.39}$$

where p and q are the order of the moments. The centroid can then be found by

$$\mathbf{c} = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \tag{2.40}$$

A vector, $\vec{\mathbf{oc}}$, from the corners centre to the centroid can then be constructed. The orientation of the patch is then:

$$\theta = \text{atan2}\,(m_{01}, m_{10}) \tag{2.41}$$

where atan2 is the quadrant-aware version of arctan.

The rotation invariance of this measure is improved by making sure that moments are computed with x and y remaining within a circular region of radius r. r is empirically chosen to be the patch size, so that x and y run from $[-r, r]$.

rBRIEF descriptor

The modified BRIEF (Binary Robust Independent Elementary Features) descriptor used in ORB is a rotation aware BREIF in order to allow BREIF to be invariant to in-plane rotation.

The BRIEF descriptor [51] is constructed from a set of binary intensity tests. From these tests a bit string description of an image patch can be made. For a smoothed image patch **p**, a binary test τ is defined by:

$$\tau(\mathbf{p}; x, y) := \begin{cases} 1, & : \mathbf{p}(x) < \mathbf{p}(y) \\ 0, & : \mathbf{p}(x) \geq \mathbf{p}(y) \end{cases} \tag{2.42}$$

where $\mathbf{p}(x)$ is the intensity of **p** at a point x. The feature is defined as a vector of n binary tests:

$$f_n(\mathbf{p}) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(\mathbf{p}; x_i, y_i) \tag{2.43}$$

the length of the vector is chosen as $n = 256$.

In order to make BRIEF invariant to in-plane rotation, a method to steer BRIEF according to the orientation of keypoints is applied [50]. A $2 \times n$ matrix, **S**, is defined for any feature set of n binary tests at location (x_i, y_i):

$$\mathbf{S} = \begin{bmatrix} x_1, \ldots, x_n \\ y_1, \ldots, y_n \end{bmatrix} \tag{2.44}$$

A "steered" version $\mathbf{S}_\theta$ of **S** is constructed by using the patch orientation θ and the corresponding rotation matrix $\mathbf{R}_\theta$:

$$\mathbf{S}_\theta = \mathbf{R}_\theta \mathbf{S} \tag{2.45}$$

The steered BREIF operator then becomes

$$g_n(\mathbf{p}, \theta) := f_n(\mathbf{p}) | (x_i, y_i) \in \mathbf{S}_\theta \tag{2.46}$$

A lookup table of precomputed BRIEF patterns are constructed where the angle is discretised to increments of $2\pi/30$(= 12degrees). The correct set of points $\mathbf{S}_\theta$ will be used to compute its descriptor as long as the keypoint orientation θ is consistent across views [50].

An important property of BRIEF is that each bit feature has a large variance and a mean close to 0.5. But once BRIEF is oriented along the keypoint direction, the means are shifted to a more distributed pattern. A high variance is desirable as it makes the feature more discriminative because it responds differentially to inputs. Also, if the tests are uncorrelated, each test will contribute to the result. Therefore, ORB runs a greedy search among all possible binary test to find the ones that have means close to 0.5 and high variance, as well as being uncorrelated. The result is called rBRIEF, which has a significant improvement in variance and correlation over steered BRIEF. [50]. The combination of oFAST and rBRIEF is what is called ORB.

Chapter 3

Method

3.1 Accelerometer calibration

Since the accelerometer equipped on the Blueye Pioneer is not calibrated, either a pre-calibration of the accelerometer or an online calibration method is needed. As the Pioneers are equipped with the same sensor-suite, the calibration parameters should in theory be equal for all of them. But this is not the case in practice. The same sensors may have differences when it comes to manufacturing, such as misalignments of the axes, or that the orientation may not be exact. If the MEMS accelerometer is used in environments with variations with respect to temperature, the user may need to frequently recalibrate [53]. An online calibration method is therefore desired in order to improve the measurement accuracy.

An online calibration method with the use of an error state Kalman Filter, based on the work in [43], was tried implemented. The chose of an error state Kalman filter over a total state Kalman filter (also known as indirect and direct Kalman filters, respectively) has multiple benefits when implemented in conjunction with an inertial navigation system (INS). Being in the INS loop and using the total state representation, a dynamic model would have to be incorporated such that the filter would maintain explicit, accurate awareness of the vehicle's motion. It would also have to attempt to suppress noisy data at a relatively high frequency [54]. A dynamic model often consists of complex non-linear equations which are hard to determine. Some models may require a large number of states, which can lead to large computational demands.

Simulation results

The ESKF implementation is developed through simulation. In this manner the filter implementation can be controlled by applying known states. The implemented ESKF filter consists of the stages and states presented in section 2.5. The IMU on the Blueye Pioneer has an update rate of 100 Hz, so the time step in the simulation is set to 0.01 seconds. The sensor readings, $\mathbf{z}_k$, sent to the filter at time k consists of the following:

$$\mathbf{z}_k = \begin{bmatrix} \mathbf{p}_k \\ \psi_k \\ \boldsymbol{\omega}_{m,k} \\ \mathbf{a}_{m,k} \end{bmatrix} \tag{3.1}$$

where $\mathbf{p}_k$ is the position of the vehicle with respect to the global frame, ψ_k is the current measured heading, and $\boldsymbol{\omega}_{m,k}$ and $\mathbf{a}_{m,k}$ is the measured angular velocity given by the gyroscope and measured acceleration given by the accelerometer, respectively. In the simulation the current position, is calculated as given in (2.24a) and (2.24c), based on the known acceleration. The position measurements are given by the visual odometry algorithm in a final implementation.

The noise variance in acceleration, $\sigma^2_{n_a}$, and angular velocity $\sigma^2_{n_\omega}$, as well as drift and bias is given in table 3.1.

Table 3.1: Parameters used in the ESKF simulation

Parameter		Value
Accelerometer	$\sigma^2_{n_a}$	0.004
	b_x	0.02
	b_y	0.02
	b_z	0.02
Gyroscope	$\sigma^2_{n_\omega}$	0.004
	d_x	0.0
	d_y	0.0
	d_z	0.0

The system is simulated with an acceleration step input of 0.05m/s^2 after 5 seconds which also last for 5 seconds. The results is shown in figure 3.1. The goal of the EKSF implementation is to correctly estimate the bias of the accelerometer measurements and at the same time filter out the noise. As shown by the results, the filter does not estimate the bias and true acceleration correctly. When the system is in steady state, i.e. when the true acceleration is zero, the filter estimates the bias correctly. The wrong estimates occur when the the step input is applied. The change in acceleration is considered a bias and not true acceleration. The true acceleration is therefore always estimated to zero.

The ESKF may have several advantages compared to a standard extended Kalman filter when implemented together with an IMU. But for a low cost IMU, as the one used in this thesis, it may lead to difficulties. One of the reasons why this implementation did not work properly is because it is a sensor based observer instead of a model based observer. For cheap sensors which may not be accurate, the distinction between correct measurements and noise may be harder to determine. This can clearly be seen in in the results, where the true change in acceleration is considered as bias.

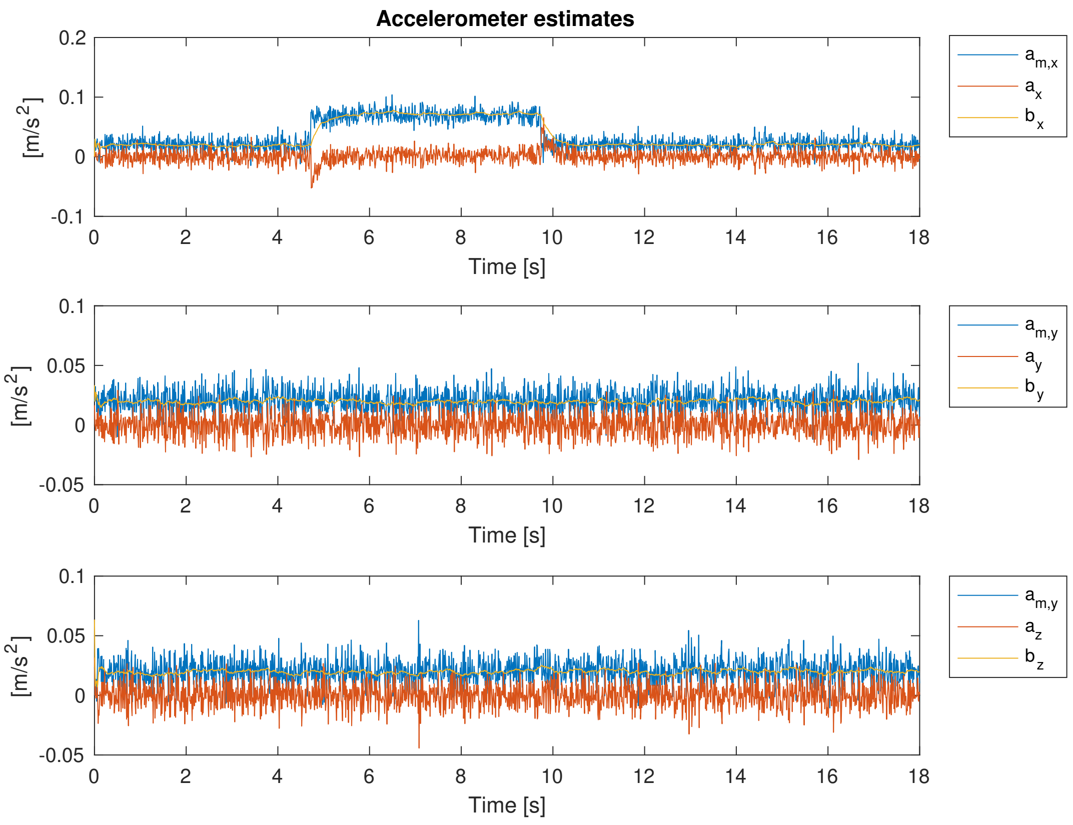

Figure 3.1: Acceleration estimates from the EKSF simulation

Implementing a well functioning ESKF for this application is both time consuming and challenging since it needs to work on multiple devices in order to be implemented as a universal calibration method on all units of the Blueye Pioneer. The implementation was therefore considered as too time consuming, and was decided not to be implemented in the final MVO algorithm in order to research and finalise the other needed parts of an implementation.

3.2 Camera calibration

In order to determine the cameras intrinsic parameters, a calibration of the camera is necessary. The intrinsic parameters consists of the focal length, the optical centre which is determined by the principle point coordinates, the skew coefficients, and the distortion coefficients which are considered additional. The intrinsic parameters is needed in order to determine the matrix **K**, given by (2.34). The calibration process is done by picturing a model with a known geometrical pattern, such as classical black-white chessboard or a symmetrical circle pattern. The intrinsic parameters are then determined by looking at the final geometrical pattern [55]. The calibration procedure for the camera in the Blueye Pioneer is done in water which is shown in figure 3.2. Most computer vision software or programming tools with computer vision ability, e.g. OpenCV or Matlab, has the ability to perform a calibration process based on predetermined camera models.

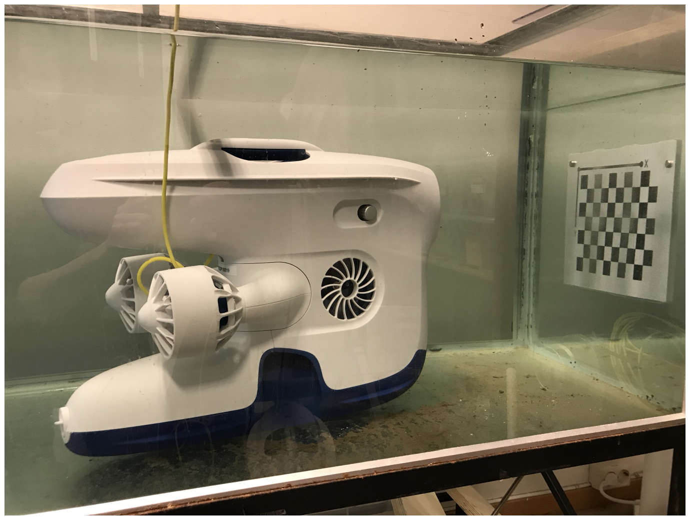
Figure 3.2: Calibration of camera using a checkerboard pattern

The results from the calibration procedure is presented in table 3.2

Table 3.2: Underwater calibration results: Intrinsic parameters and distortion coefficients

f_x	1354.45761 pixels
f_y	1379.97405 pixels
c_x	891.06972 pixels
c_y	756.19288 pixels
k_1	-0.2708139
k_2	0.2005247
p_1	0.0208302
p_2	0.0002806
k_3	-0.1013460

3.3 Motion estimation

Consider a point correspondence represented by two image points $\mathbf{q}_1$, $\mathbf{q}_2$ given by homogenous coordinates in two successive image frames, I_k and I_{k-1}. The geometric relation between the

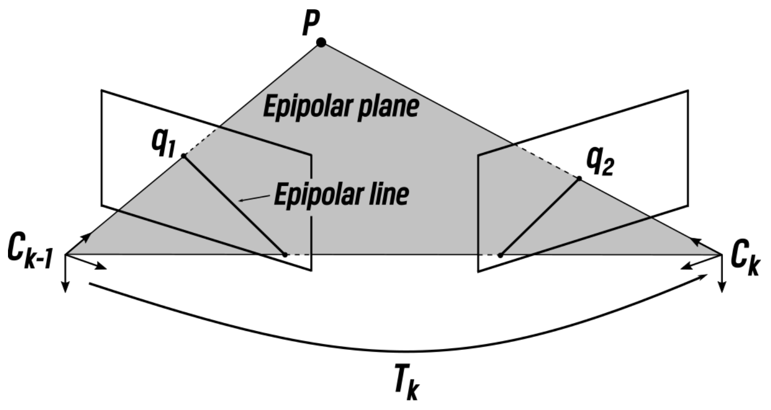

Figure 3.3: Illustration of the epipolar constraint

two image frames are given by the so called essential matrix **E**. The camera motion parameters can be extracted from the essential matrix up to an unknown scale for the translation. The essential matrix is given on the form:

$$\mathbf{E}_k = \mathbf{R}_k \mathbf{S}(\mathbf{t}_k) \tag{3.2}$$

where **R** is the rotation matrix, and **S**(**t**) is the matrix representation of the cross product with **t**, given as

$$\mathbf{S}(\mathbf{t}) = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{3.3}$$

The essential matrix can be computed using the 2D-to-2D correspondences between the image points. The epipolar constraint is the main property of the 2D-to-2D based motion estimation [9]. This constraint determines the line in which the corresponding point lies in the other image, called the epipolar line and shown in figure 3.3. The epipolar constraint is described by:

$$\mathbf{q}_2^\top \mathbf{K}^\top \mathbf{E} \mathbf{K} \mathbf{q}_1 = 0 \tag{3.4}$$

In order to determine the the essential matrix, Nistér's five-point algorithm [56] is used. The algorithm consists of computing the coefficients of a tenth degree polynomial in closed form and, subsequently, finding its roots. The algorithm requires the minimum number of points possible, since the essential matrix has only five degrees of freedom.

The rotation matrix **R** and translation **t** are recovered from the essential matrix based on theorem 3 given in [56] which states that given a singular decomposition of the essential matrix as

$$\mathbf{E} \sim \mathbf{U}\mathrm{diag}(1,1,0)\mathbf{V}^\top \tag{3.5}$$

where ~ denotes equality up to scale. **U**, and **V** are chosen such that det(**U**) > 0 and det(**V**) > 0.

Then:

$$\mathbf{t} \sim \mathbf{t}_u \equiv [u_{12}, u_{23}, u_{33}]^\top \tag{3.6}$$

and **R** is equal to

$$\mathbf{R}_a \equiv \mathbf{U}\mathbf{D}\mathbf{V}^T \qquad \text{or} \qquad \mathbf{R_b} \equiv \mathbf{U}\mathbf{D}^\top\mathbf{V}^\top \tag{3.7}$$

Any combination of **R** and **t** according to conditions above satisfies the epipolar constraint in (3.4).

In order to resolve this, it is assumed that the first camera matrix is $[\mathbf{I}_{3x3}|\mathbf{0}_{3x1}]$ and the translation **t** is of unit length. There are then four possible solutions. In order to determine which choice corresponds to the true configuration, we make use of the cheirality constraint which says that scene points should be in front of the camera.

By denoting the rotation and translation between the two last camera frames to be $\mathbf{R}_k$ and $\mathbf{t}_k$, the current camera pose expressed by $\mathbf{R}_{tot,k}$ $\mathbf{t}_{tot,k}$ can then be written by:

$$\mathbf{R}_{tot,k} = \mathbf{R}_{tot,k-1}\mathbf{R}_k \tag{3.8}$$

$$\mathbf{t}_{tot,k} = \mathbf{t}_{tot,k-1} + \lambda\mathbf{R}_{tot,k}\mathbf{t}_k \tag{3.9}$$

where λ is the scale parameter.

Finally the set of camera poses $\mathbf{C}_{0:n} = \{\mathbf{C}_0, \dots, \mathbf{C}_n\}$ where $n \in [0, k]$ contains the transformations of the camera with respect to the initial coordinate frame at $k = 0$. A visualisation of the motion estimation is shown in figure 3.4.

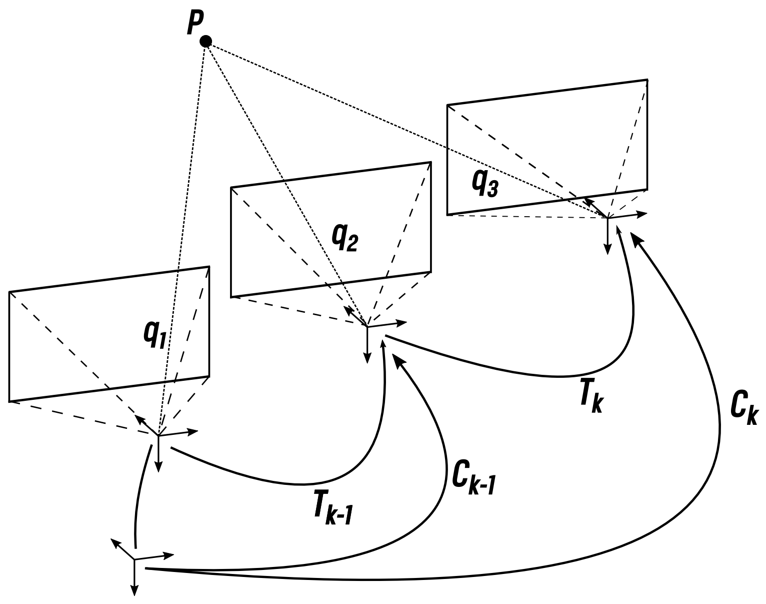

Figure 3.4: Graphical representation of monocular visual odometry

3.4 Feature matching

Tracking of detected features are done by matching features in two subsequent frames. The matching is done by the Brute-Force matcher. Brute-Force matching is simple as it match features based on distance calculation. The closest feature is returned as a match. The distance between two binary string descriptors is obtained by calculating the Hamming distance. The Hamming distance between two strings of equal length is the number of positions which are not equal. E.g. the Hamming distance between $4\underline{1}95\underline{8}24$ and $4\underline{3}95\underline{3}24$ is 2. Calculating the Hamming distance is faster than computing the Euclidean distance, which is an advantage with respect to computational effort [51]. Figure 3.5 shows feature matching between two image frames.

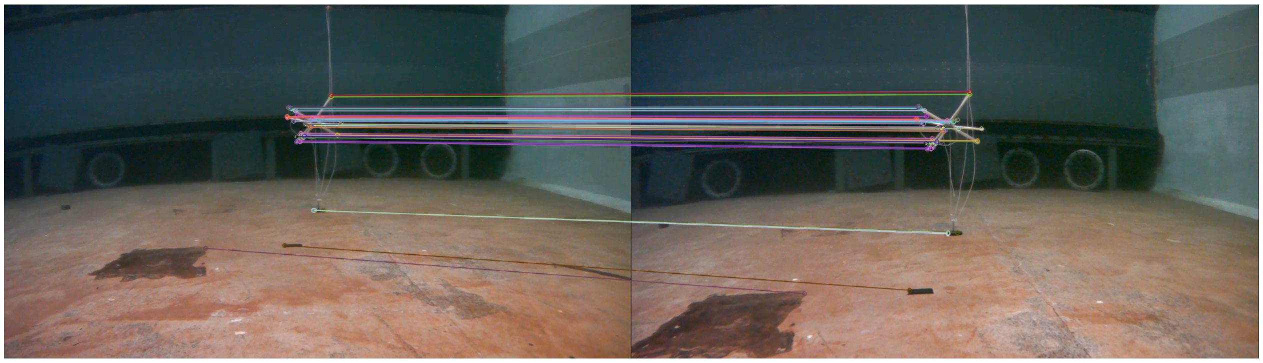

Figure 3.5: Matching of features detected using ORB and matched with use of the Hamming distance

3.5 Feature detector comparison

Timing and accuracy of feature-based tracking is important in a real-time application in order to end up with reliable position estimates. Therefore, correct choice of feature-detector-descriptor is critical in feature-matching applications. Computational efficiency and robustness is two key characteristics of a good feature-detector-descriptor. A reasonable quick computational time is critical in order to implement a feature-based tracking algorithm real time.

SIFT, SURF, KAZE, AKAZE, ORB and BRISK are among the fundamental feature-detectors [49]. Both SIFT and SURF are patented, which results in that these feature-detector algorithms are not considered further in this thesis. Also, because of the patent, these algorithms are part of what OpenCV calls "non-free" modules in newer versions of the computer vision library. A comparison of ORB, BRISK, AKAZE and KAZE follows.

In order to distinguish which feature-detector to use, the different algorithm have been tested on datasets that simulates the environment for where a dynamic positioning feature for the Blueye Pioneer would be beneficial. One ability of a robust feature-detector for use in un-

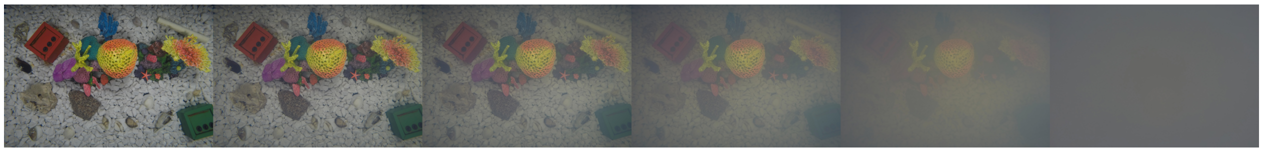

Figure 3.6: The Milk subset of the TURBID dataset (6 out of 20 images). Degradation is increased in each image by adding a controlled amount of milk between each image.

derwater environments is good performance in different levels of turbidity. Ocean turbidity, also known as "cloudiness" or "haziness", is caused by individual particles in the seawater. This particles can be too small to be seen without magnification. Scattering and attenuation of light cause the loss of water transparency, and results in turbid images. As light is backscattered, a visibility condition similar to when headlights on a car is used in fog arises.

The TURBID dataset consists of a collection of underwater images, generated to contribute to the underwater research area [57]. The turbidity, and consequently the amount of degradation in each image, is increased by adding a specific amount of whole milk into the water tank. The Milk subset is used for comparison analysis in this thesis. In this subset, a set of 20 images is taken. The images are taken in a water tank of a man made scene. The first image is taken in clear water and works as a reference image. A collection of the subset is shown in figure 3.6. The subset is herby referred to as the TURBID dataset.

A comparison of number of features found by AKAZE, KAZE, ORB and BRISK in each image of the TURBID dataset is shown in figure 3.7. The first image is the reference image, taken in clear water. The average computational time for detection features in each image by the feature-detector algorithms is presented in table 3.3. The ORB algorithm shows an extreme superiority when it comes to feature detection. The oriented FAST detector used in ORB, has an overall better performance compared to the other detectors. Also, ORB detect features in the images with high turbidity where the other algorithms does not. However, matching time for such a large number of features, prolongs the total image matching time.

The computational time for feature detection in the TURBID dataset shows that BRISK and ORB has a clear advantage compared to AKAZE and KAZE. KAZE uses an average of 0.4770 seconds to find features in each image. Together with matching, this algorithm has too long computational time to be implemented in a real-time application. The same applies for AKAZE. There was not to be found any correlation between turbidity level and computational time of feature detection, which is worth to mention as the features found in each image is naturally affected by the turbidity. For this dataset, ORB has the fastest computational time for feature detection.

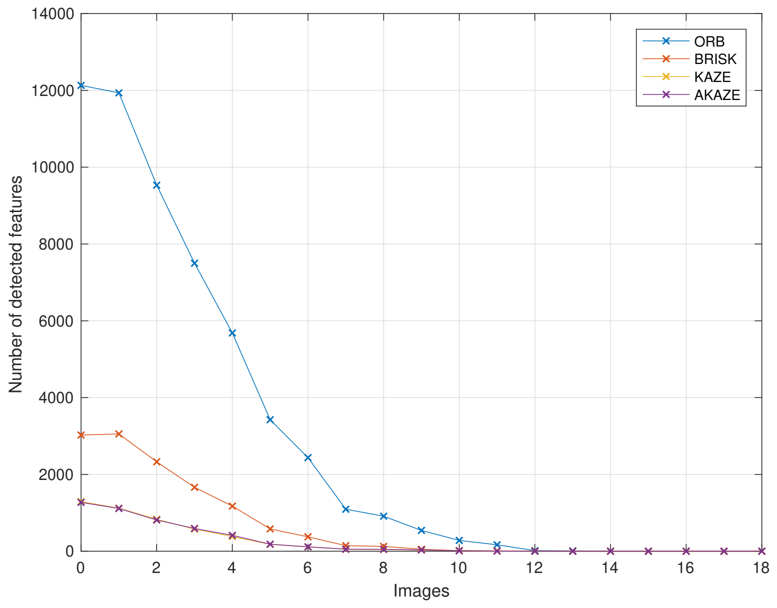

Figure 3.7: Comparison of features detected in the TURBID dataset

Table 3.3: Computational time of feature detection in the TURBID dataset

	ORB	BRISK	KAZE	AKAZE
Detection [s]	0.0158	0.0170	0.4770	0.0689

A comparison of the feature-detectors is also done on images taken from the test facility in Marine Technologies MC-Lab. The dataset consists images taken with the Blueye Pioneer, where the drone performs a forward motion towards a floating cross. Two of the images can be seen in figure 3.5. The dataset consists of 160 images. The dataset is herby referred to as the POOL dataset. In addition to feature detection, a matching between the features found in two following pictures throughout the dataset is done to compare the tracking abilities of the algorithms. ORB in an unbounded state detects a large number of features, as seen in the results from the TURBID dataset. Hence, an increased computational time for feature-matching. The feature matching cost can be reduced by using its detector in a bounded state. The max amount of features is therefore put to 500 in this thesis. BRISK, ORB and AKZE have binary descriptors, so matching is preformed as described in section 3.4. KAZE uses a string based descriptor, which means that the matching has to be done differently using L1-norm or L2-norm. In this comparison, L2-norm is chosen. The results is presented in figure 3.8 and table 3.4.

ORB detects most features in this dataset as well. The number of features found in these images are much lower than the number of features found in the TURBID dataset. This clearly affects the total computational time. BRISK shows a faster total computational time than ORB. However, the difference is so small that it is not considered advantageous in a real-time imple-

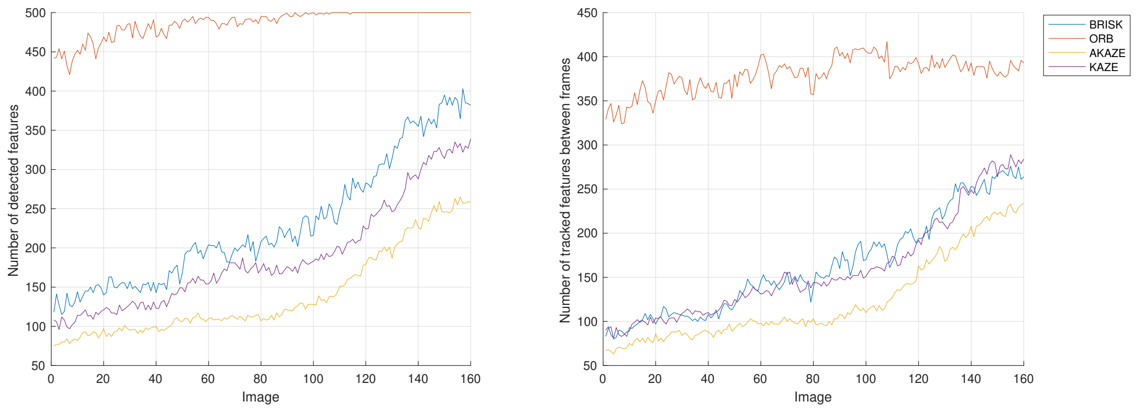

Figure 3.8: Number of detected features (left) and tracked features between images (right) in the POOL dataset

Table 3.4: Computational time of feature detection and matching in the POOL dataset

	ORB	BRISK	KAZE	AKAZE
Detection [s]	0.0094	0.0090	0.3444	0.0752
Detection + matching [s]	0.0217	0.0195	0.3534	0.0837

mentation, and the number of features detected by ORB makes the algorithm more accurate in terms of pose estimation.

By the comparison of the feature detector and descriptor algorithms, ORB is chosen as the algorithm used in the MVO implementation in this thesis. Comparison of the algorithms are also done by [49], however not in water. The results are similar, with ORB being the preferred algorithm.

3.6 Program Implementation

The monocular visual odometry implementation conducted in this thesis uses the theory previously described in chapter 2 and the methods described in section 3.3 and 3.4. The implementation is also based on results from the method described earlier in this chapter.

For two successive image frames captured at time step $k-1$ and k, features in both images are extracted, as well as given a descriptor, using ORB. Before the features are extracted, the images are corrected for distortion. The found features are matched using the Brute-Force matcher comparing their Hamming distance as described in section 3.4. The found matches are then sorted based on the distance, as the matches characterised as "best" are the ones with the shortest Hamming distance.

Unfortunately, not all matches will be a correct match. A way to sort of the incorrect matches is through a nearest-neighbour search. The idea is presented in [58]. This is done by returning two descriptors that are a match for each feature. By comparing their distance, the closest neighbour is the actual match, while the second-closest is defined to be the closest neighbour not to be a match. For each feature, two matches and two Hamming distances will then be computed. To sort of the matches that are incorrect, the ratio between the closest and the second closest neighbour is compared. To be accepted as a correct match, the closest neighbour must have a lower distance than the second closest. As defined in [58], the distances of the two neighbours will be quite similar if an incorrect match occurs. Thus, the closest neighbour will only be accepted as a correct match if the distance is lower than a threshold defined. For the MVO implementation in this thesis, this closest neighbour will be accepted as match if the distance is lower than 80% of the second closest.

Based on the matches between two consecutive image frames, Nistér's five-point algorithm with random sample consensus (RANSAC) is used to compute the essential matrix. The given matches contains outliers, hence RANSAC is used as this method is suited for applications in automated image analysis where an interpretation is based on data given by error-prone feature-detectors [59]. A simple example is fitting a line in two dimensions given a set of observations. Assuming that the observations contain both inliers and outliers, i.e. points which approximately can be fitted to the line and points which cannot be fitted to the line, a simple least square method would produce a line that fits badly to the data including both inliers and outliers. This is because the least square method produce a line that is optimally fitted to all points, including the outliers. RANSAC, on the other hand, attempts to only use the inliers in its calculation by fitting the linear models to several random samplings of the data and return only the models that has the best fit to a subset of the data. In this manner, it excludes the outliers. A random subset consisting of inliers would have the best model fit, since the inliers tend to be more linearly related than a random mixture of inliers and outliers. In addition to the matches and a fitting model given as input to RANSAC, some confidence parameters are set. The threshold parameter defines when a datapoint fits the model, while the probability defines the desired probability that we get a good sample. The threshold is set to 0.5 and the probability is set to 0.999 in the implemented algorithm.

After the essential matrix is computed, the translation and rotation between the image frames can be extracted as described through equation (3.5) to (3.7). The calculated translation and rotation between the two consecutive images are then added to the total translation and rotation, as given by equation (3.8) and (3.9). Before a new image is captured and the cycle starts over, the features found in the first image is stored in order to be matched with the new features found the incoming image.

The implemented MVO algorithm is able to deliver position estimates with an update rate of approximately 10 Hz, meaning the algorithm can process approximately 10 frames per second in the given environments. A pipeline of the MVO is shown in figure 3.9.

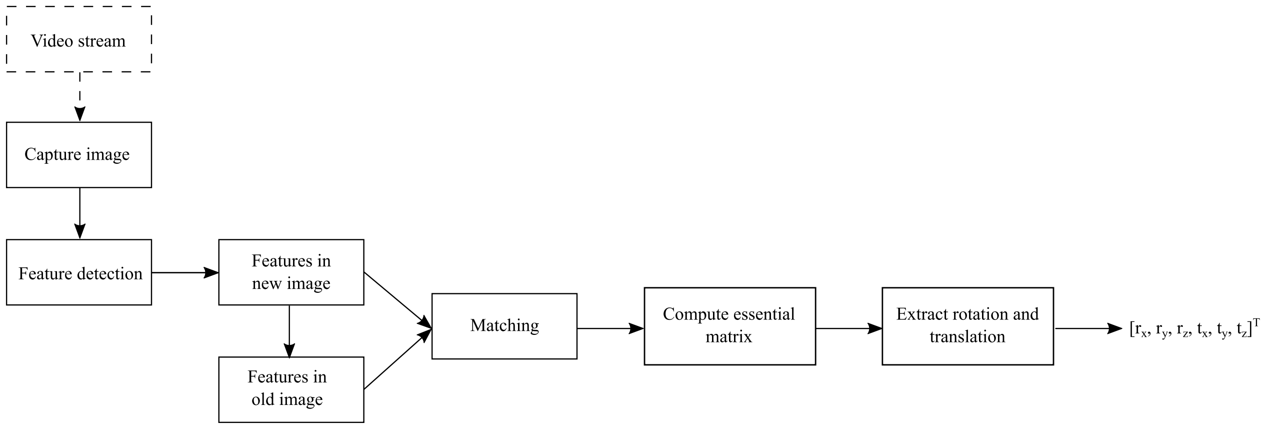

Figure 3.9: Monocular visual odometry pipeline from feature detection to motion estimation

By using one camera, we can no longer tell if we are looking at a large object in the distance or a smaller object close to the camera since we have lost the depth information. Just try to close one eye and determine an unknown distance. Therefore, a scale parameter needs to be estimated in order to add a scale to the motion estimations from the MVO-algorithm.

A translation in heave is performed to estimate the scale parameter. In this manner, a measurable translation where the distance is known can be compared to the estimated translation from the MVO algorithm. In the beginning of every run, the drone is moved 20cm in heave. The scale, λ is then calculated as the relationship between the measured and estimated translation in heave from the MVO algorithm.

$$\lambda = |\frac{z_k - z_0}{\hat{z}_k}| \tag{3.10}$$

where z_k and z_0 is the measured depth at time k and the initial depth at time $k = 0$, respectively, an $\hat{x}_k$ is the estimated translation in heave form the MVO algorithm.

3.7 Simulation and test scenarios

The implemented algorithm is tested through different simulations and real-time tests. This section will present the different scenarios, while the results with comments will be presented in the next chapter. The underwater environment offers greater challenges with respect to computer vision as stated in section 1.2. In order to verify the different aspects of the algorithm, as well as evaluate its performance and try to find its limitations, a set of different tests conducted both in-air and underwater were chosen. The next subsections presents the different simulation and test scenarios.

3.7.1 KITTI dataset

As an initial test, the MVO algorithm was evaluated on the KITTI dataset [60]. The dataset works as a benchmark suite for real-world computer vision. Different tasks of interest can be used with the dataset, such as stereo, optical flow, visual odometry, SLAM, 3D object detection and 3D tracking. The dataset is recorded from a car driving around in the city of Karlsruhe, Germany, and consists of camera images, laser scans, high-precision GPS measurements and IMU accelerations from a combined GPS/IMU system. The purpose of the dataset is to contribute to the development of computer vision [60].

The dataset has its own visual odometry subset consisting of 22 stereo sequences. 11 of them are provided with ground truth trajectories. The MVO algorithm is tested on two of the sequences and the results are presented in the following. The first sequence consists of footage from a road with smooth turns and few road changes. The second sequence is recorded in a city with several sharp and 90 degree turns, as the cars is driving around blocks and changing streets, which provides a good test of the rotational accuracy of the algorithm. The MVO algorithm is tested on a 60 second sample of both sequences.

As the sequences consists of recorded camera images with known ground truth, there is no way to perform the scale estimation procedure implemented in the MVO algorithm as stated in section 3.6. The effect of a wrong scale parameter is therefore not visible in the this comparison. The scale parameter, λ, used in the KITTI dataset is calculated as

$$\lambda = \sqrt{(x - x_{\text{prev}})^2 + (y - y_{\text{prev}})^2 + (z - z_{\text{prev}})^2} \tag{3.11}$$

where (x, y, z) and $(x_{\text{prev}}, y_{\text{prev}}, z_{\text{prev}})$ is the known ground truth coordinates for current and previous camera frame, respectively.

3.7.2 In air evaluation

The MVO algorithm was tested on the Blueye Pioneer in air. A calibration procedure as described in section 3.2 was done in air prior to the test. The calibration results is presented in appendix. A simple scenario was presented, where the drone where moved through a straight office hallway. A picture from the hallway can be seen in figure 3.10.

3.7.3 Tests in MC-lab

The final and main tests where conducted in the MC-lab at Marine technologies facilities at Tyholt. The laboratory consists of a 40m x 6.45m x 1.5m pool with a Qualisys real-time positioning system. The system consists of 6 Oqus cameras and the Qualisys Track Manager (QTM) software [61]. By applying markers on the Blueye Pioneer, the Qualisys software makes it possible

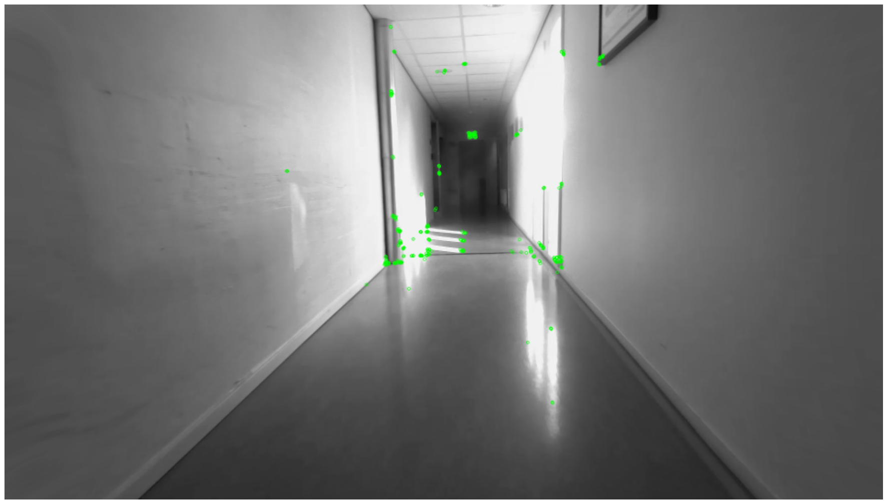
Figure 3.10: Hallway used for in air evaluation. Features detected are marked green

to record 6-DOF movement of the mini ROV. As the walls and bottom of the pool contains few features, a simple cross made of out of buoyant material was made to use as an object of interest. Figure 3.11 and 3.12 shows the experimental setup of the Blueye Pioneer with markers, and the object of interest.

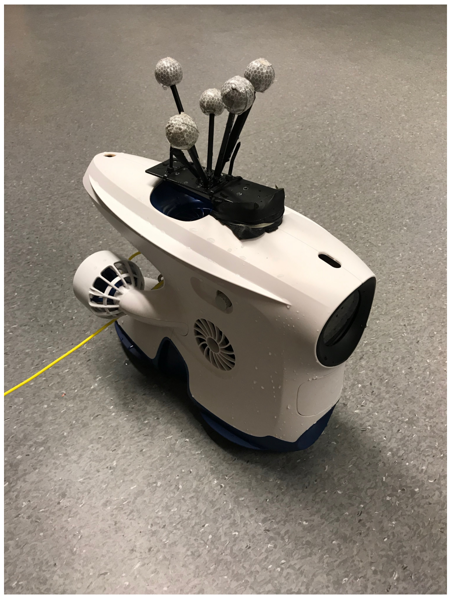
Figure 3.11: Blueye Pioneer equipped with markers to measure position and orientation

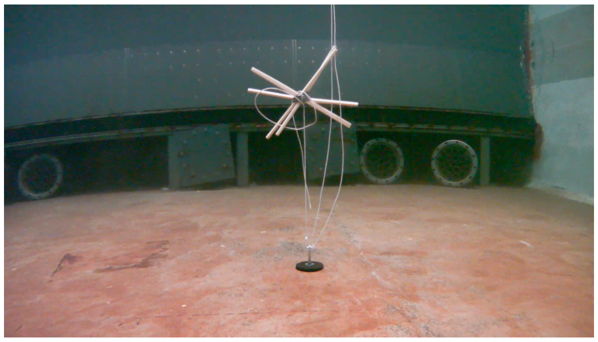
Figure 3.12: Object used as reference during testing

Different scenarios where tested in the MC-Lab to evaluate the different aspects of the MVO algorithm. Pure translation scenarios and station keeping where tested while performing motion estimation. During testing, it was soon realised that the applied algorithm did not perform well enough in able to be used for dynamic positioning. The reasons why is discussed in chapter 5. The different test scenarios were still conducted in order of trying to find the limitations of the algorithm and sources of errors of the results. The different test scenarios presented is:

1. Pure surge translation backwards from the object of interest.
2. Forward and backward translation.
3. Sway translation.
4. Station keeping.

3.7.4 Scale estimation

In addition to the different test scenarios, a series of scale estimations were performed. As stated earlier in section 3.6, the scale is estimated in the beginning of every run. Using the measurements from the depth sensor the drone is driven straight upwards 20 cm. The scale is then calculated as the relation between the measured and estimated translation in heave from the MVO algorithm.

Chapter 4

Results

This chapter presents the different results from the simulation and test scenarios presented in section 3.7.

4.1 Comparison with KITTI dataset

The results is shown in figure 4.1 and 4.2 .

From the results of the first sequence, figure 4.1, the MVO algorithm manage to estimate the trajectory well. However, the position error accumulates with distance, and the position error is approximately 14 meters after 60 seconds.

The results from the second test shows that the algorithm estimates the trajectory well while the car is driving on a straight path. The position error quickly rises as the first turn is performed, and it is clear from the presented results that the wrong rotation estimation is the cause of the accumulated error throughout the rest of the sequence. While the car is driving on a straight path, the MVO algorithm estimates a straight trajectory. Still, because of the the wrong estimates in rotation, the position error is above 80 meters after 60 seconds in this scenario.

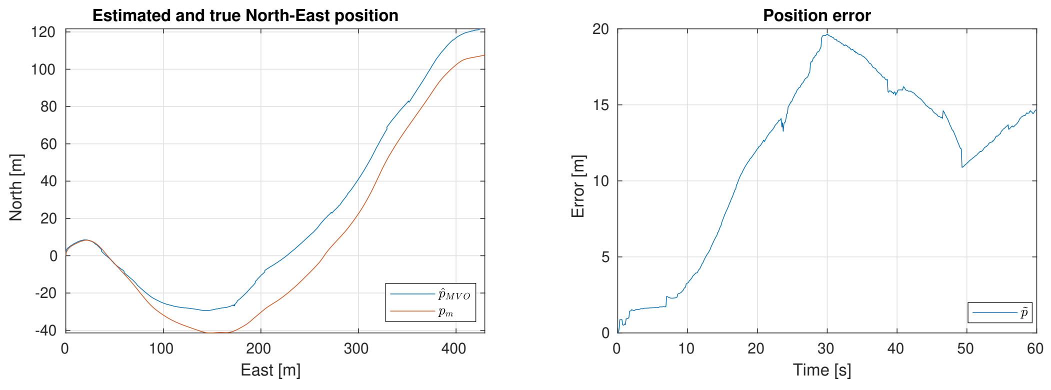

Figure 4.1: Results from comparison with the KITTI dataset, first sequence

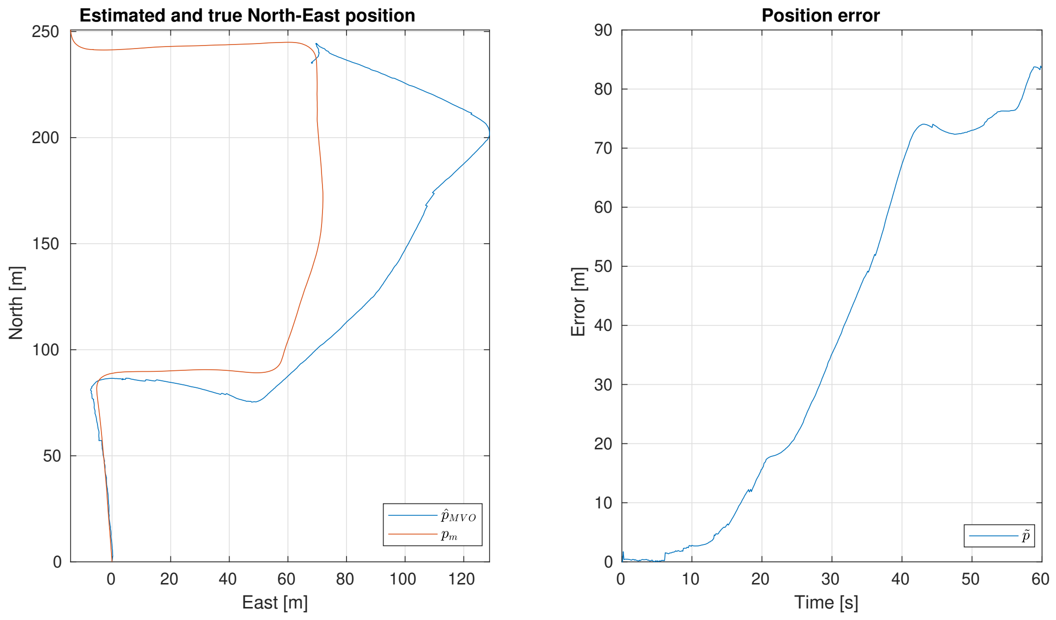

Figure 4.2: Results from comparison with the KITTI dataset, second sequence

4.2 In air evaluation

The results is presented in figure 4.3. The results have no ground truth for comparison, but as the applied movement is a pure surge translation, it is easy for the reader to visualise the expected ground truth when evaluating the results. The ROV was moved approximately 18 meters. The estimated translation in surge is therefore close to the true translation.

It is expected from a well functioning visual odometry algorithm that the result of the trajectory estimation would be a straight path. The results shows that the MVO algorithm estimates the trajectory well. It has an offset of ± 0.4 meters in sway along the trajectory.

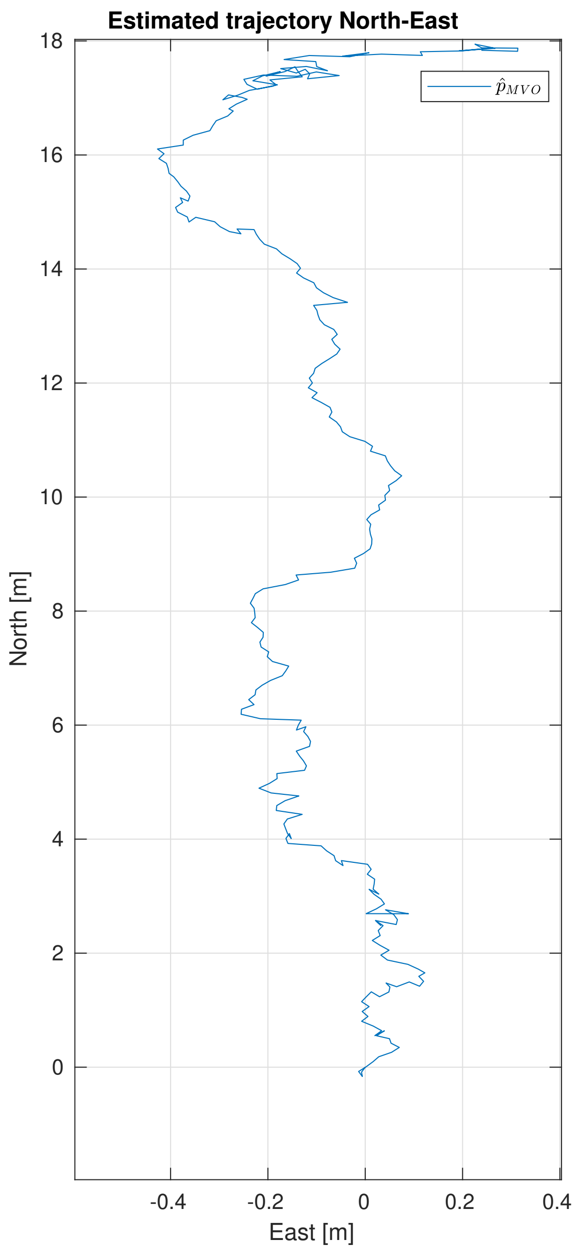

Figure 4.3: Estimated trajectory of motion through a straight hallway

4.3 Pool tests

4.3.1 Scenario 1: Surge translation

A pure surge translation were tested to analyse the algorithms ability to perform motion estimation in what was considered a simple scenario. This scenario is also comparable to the in-air evaluation presented in section 4.2, which therefore gives an indication on the challenges associated with the underwater environments. The North-East map is given in figure 4.4, the estimated change in position and rotation is given in figure 4.5 and 4.6, respectively.

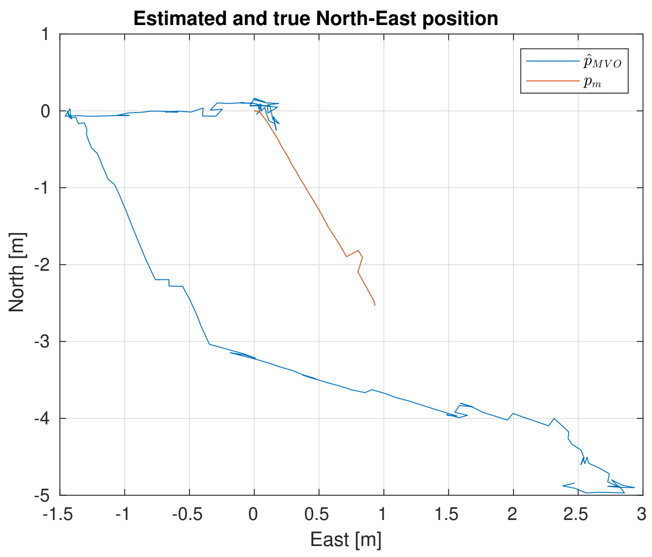

Figure 4.4: Results scenario 1: North-East plot of the estimated trajectory and ground truth

The algorithm estimates a wrong orientation from the beginning, resulting in an estimated translation in sway instead of surge during the first seconds. After 16 seconds the ground truth position data from Qualisys have a sudden change. The software looses track of some or all the markers on the ROV, and estimates an orientation which causes the sudden change. The results after 16 seconds should therefore not be considered in the evaluation.

4.3.2 Scenario 2: Forward-backward translation

In order to analyse the change in translation, a forward-backward test were conducted. The North-East plot is presented in figure 4.7, estimated change in position in figure 4.8, and estimated change in orientation in figure 4.9.

The North-East plot clearly shows the errors of a wrongly estimated scale parameter. The algorithm estimates a 14 meter forward surge translation, which should be around 2 meters. Studying the surge plot in figure 4.8 shows that the MVO algorithm does not manage to estimate the change in surge motion, where the backward motion is not correctly estimated. The

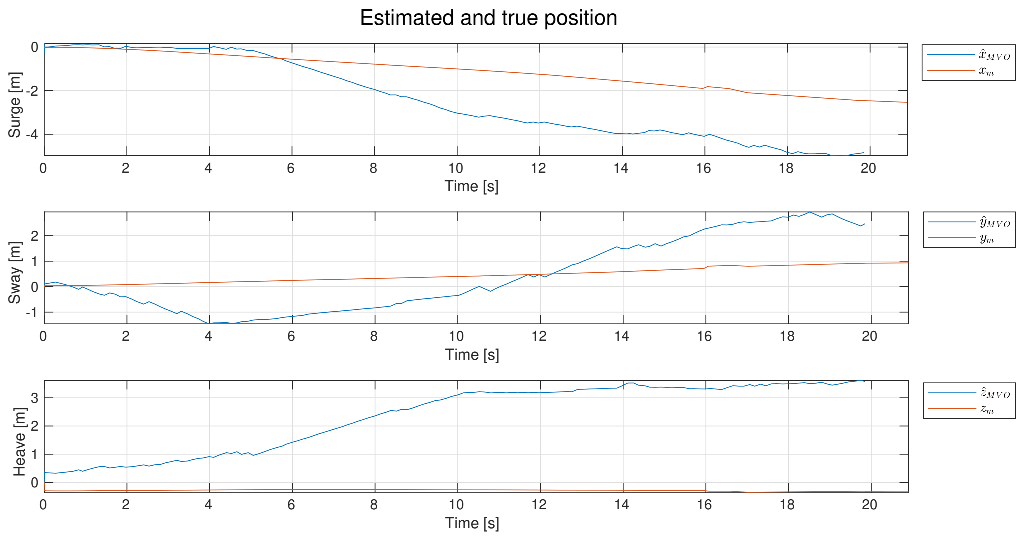

Figure 4.5: Results scenario 1: Measured and estimated movement in surge, sway and heave

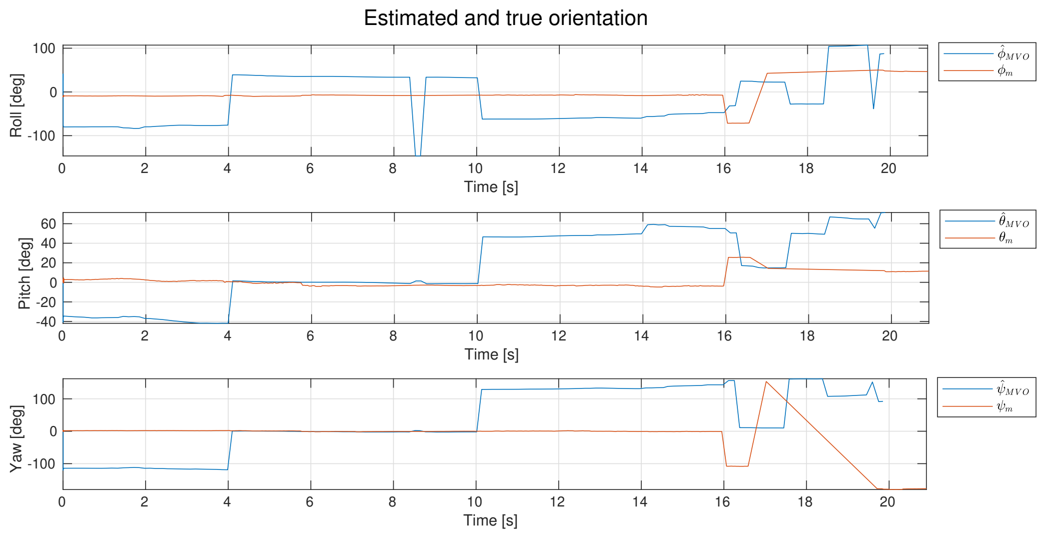

Figure 4.6: Results scenario 1: Measured and estimated orientation

transition between forward and backward motion happens after ~ 20 seconds. The orientation estimation has a sudden change in roll at the same time. The wrong estimations in rotation clearly affect the position estimation, resulting in wrong estimates because of an orientation of the drone that is not correct. This is also clearly in the heave motion. The drone is kept on a constant depth throughout the test, but the MVO algorithm estimates a depth below 6 meters after 35 seconds.

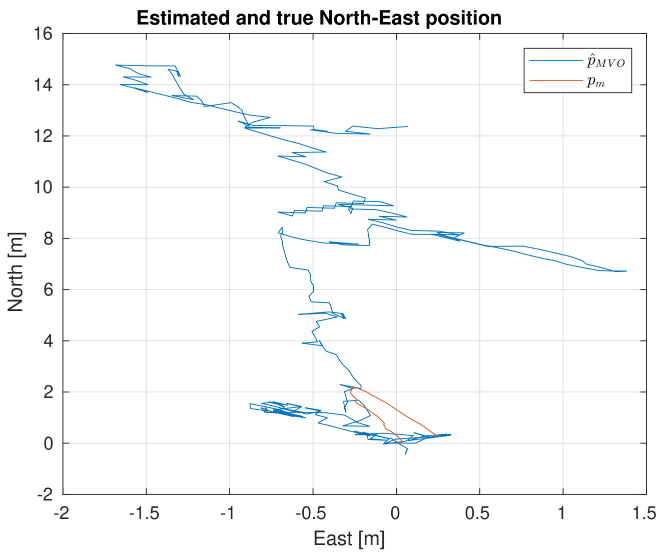

Figure 4.7: Results scenario 2: North-East plot of the estimated trajectory and ground truth

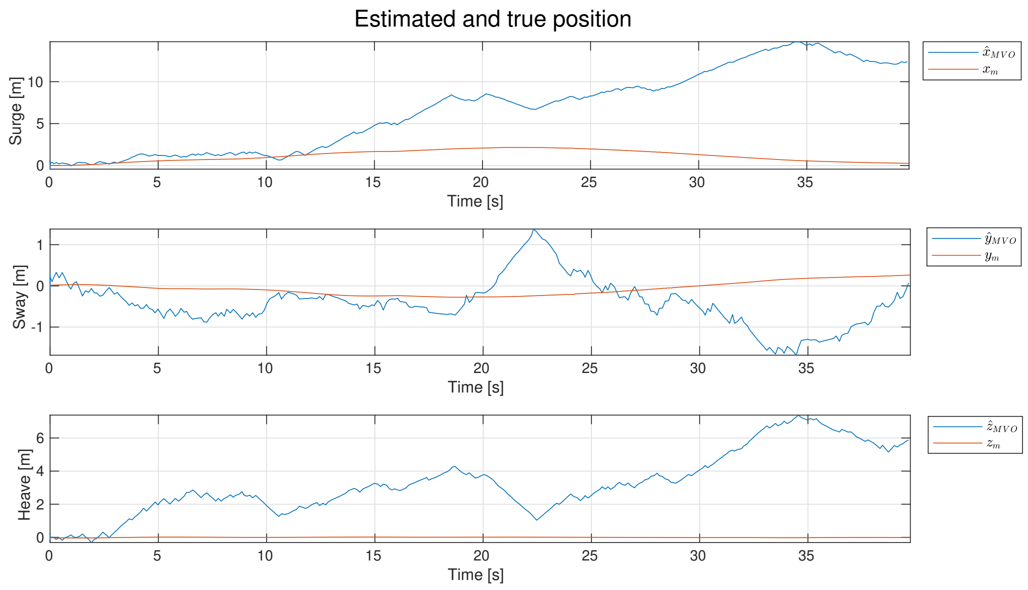

Figure 4.8: Results scenario 2: Measured and estimated movement in surge, sway and heave

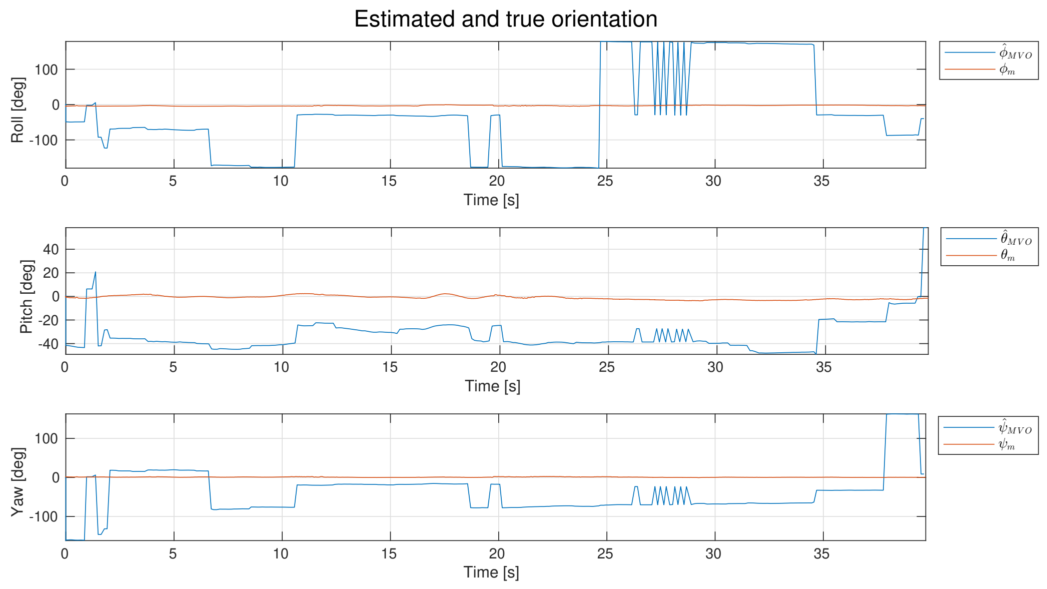

Figure 4.9: Results scenario 2: Measured and estimated orientation

4.3.3 Scenario 3: Sway translation

In the previous scenarios surge motion have been under evaluation, while in this scenario sway motion is considered. The North-East plot is presented in figure 4.10, estimated change in position is presented in figure 4.11, and estimated change in position in figure 4.12. A negative sway motion is applied, while the heading is tried kept constant throughout the test. The underwater drone will therefore move to the left with respect to the body frame, and in this manner the object of interest will move through the image frame, starting in the far left and ending up in the far right.

The results, as the previous scenarios, shows that the MVO algorithm is not capable of computing a correct motion. During the test, the underwater drone moves approximately 1.7 meters to the left, and 0.45 meters backwards while keeping an almost constant heading. Studying figure 4.11, the MVO algorithm is not capable of computing the sway motion at all. Throughout the test, the estimated sway coordinate $\hat{y}$ have small fluctuations around zero.

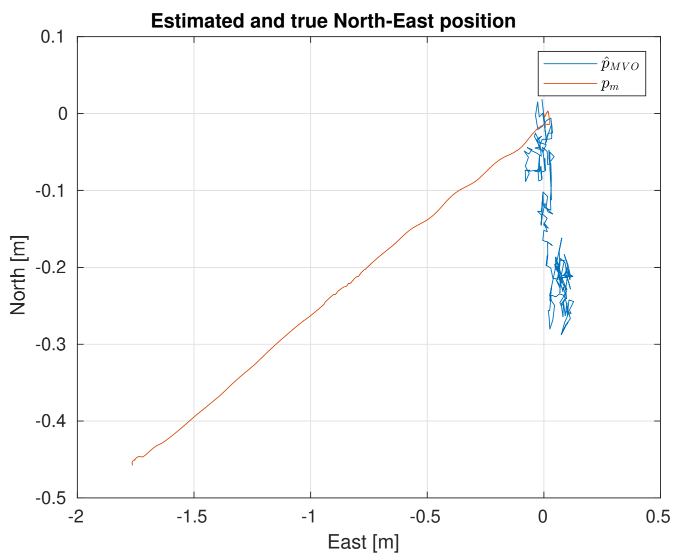

Figure 4.10: Result scenario 3: North-East plot of the estimated trajectory and ground truth

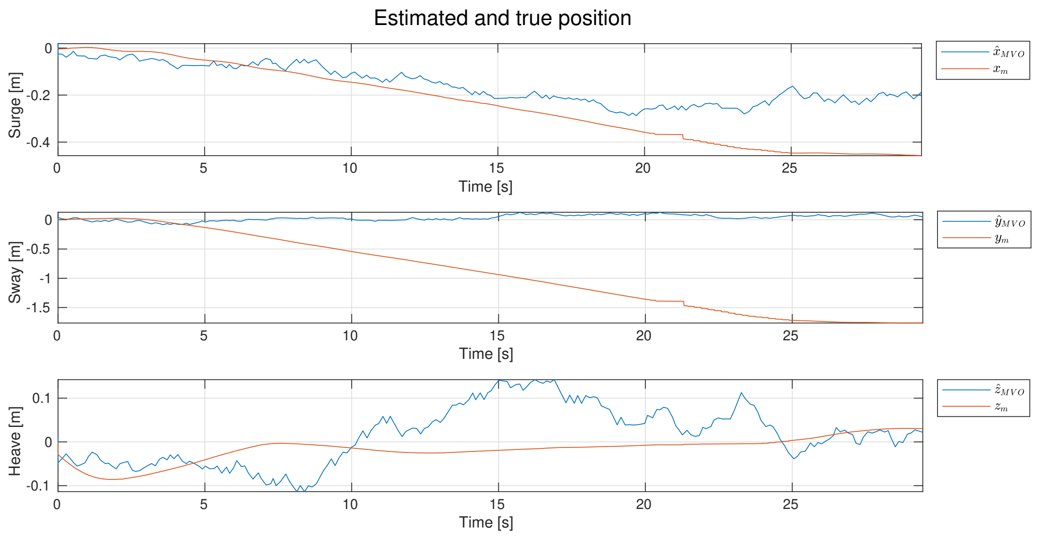

Figure 4.11: Results scenario 3: Measured and estimated movement in surge, sway and heave

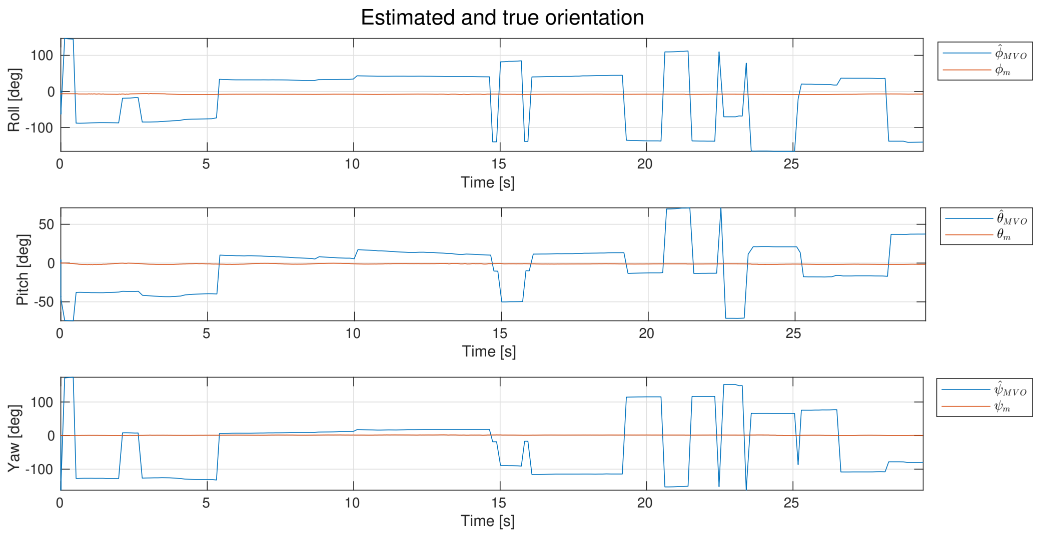

Figure 4.12: Results scenario 3: Measured and estimated orientation

4.3.4 Scenario 4: Station keeping

Since the MVO algorithm was made with intention of being used for dynamic positioning, the ability of position estimation during station keeping was tested. The resulting North-East map is shown in figure 4.13, the estimated position in surge, sway and heave together with ground truth is shown in figure 4.14, and the estimated orientation is shown in figure 4.15.

The station keeping is preformed in 60 seconds. The results shows that the drift in surge is higher than for sway and heave, with approximately 5 meters. There is high fluctuations in all 3 DOFs, making it easy to see that the position estimates from the MVO-algorithm could not be used not be used for dynamic positioning without any manipulation.

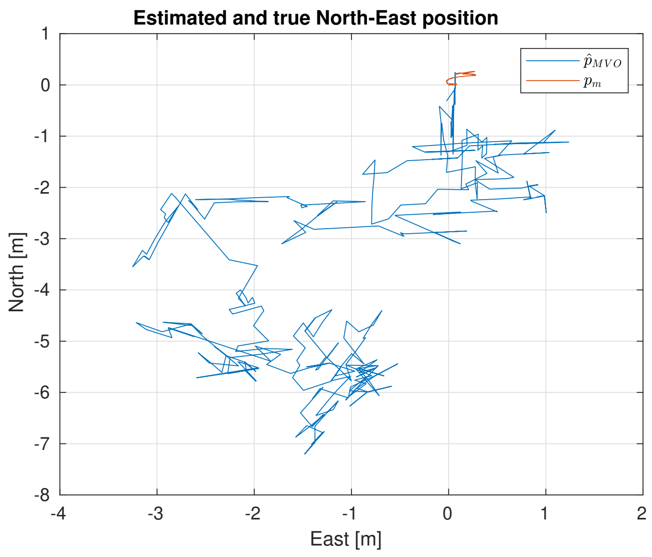

Figure 4.13: Results scenario 5: North-East plot of the estimated trajectory and ground truth

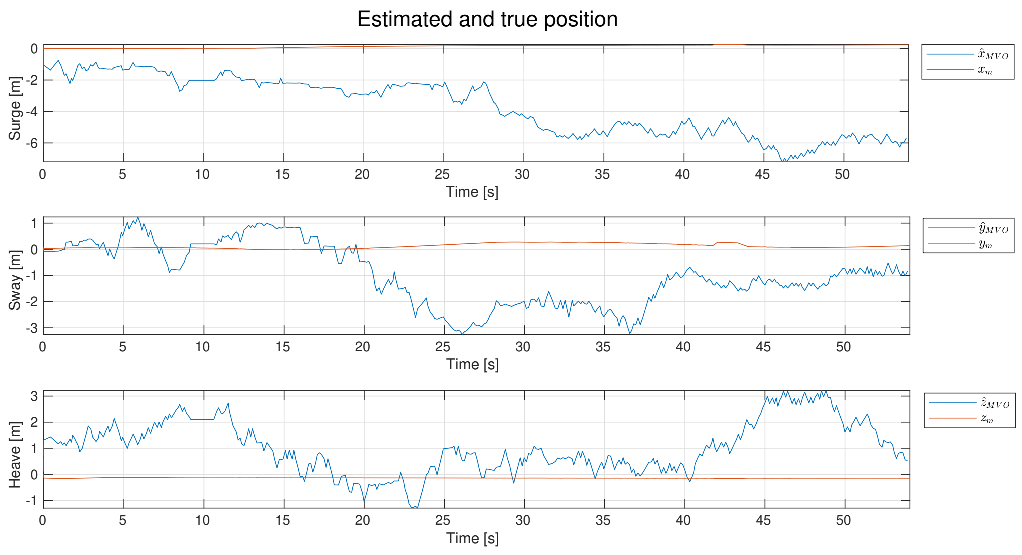

Figure 4.14: Results scenario 5: Measured and estimated movement in surge, sway and heave

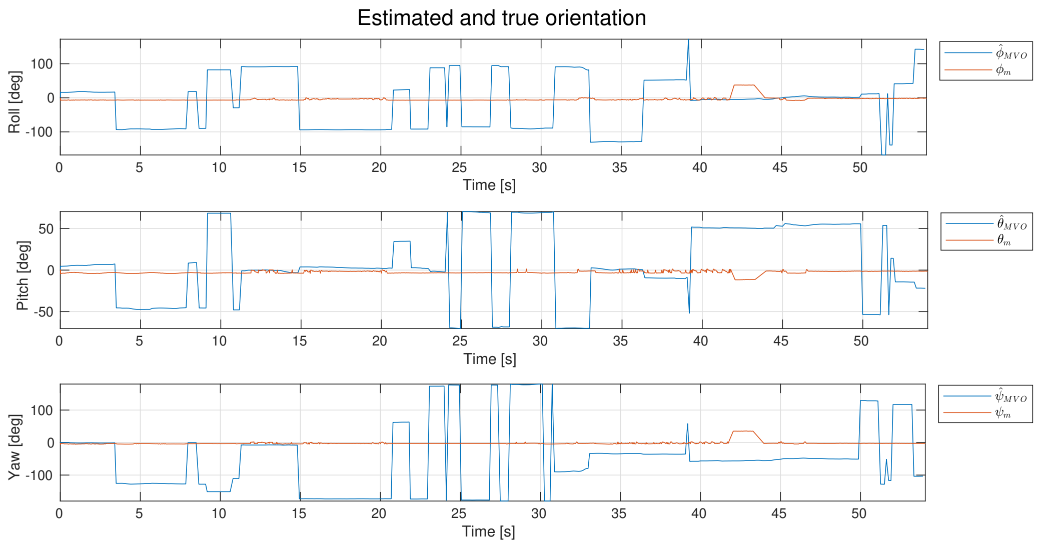

Figure 4.15: Results scenario 5: Measured and estimated movement in roll, pitch and yaw

4.4 Scale estimation

A series of 65 scale estimations is performed, where the results is given by the box plot in figure 4.16. The median is 0.088152, while the max and min values are 1.4989 and 0.018228, respectively.

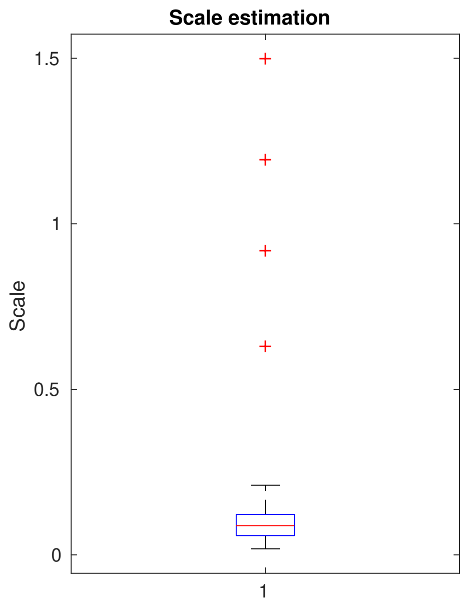

Figure 4.16: Box plot of a series of 65 scale estimations performed

Chapter 5

Discussion

5.1 Results

The results form the KITTI dataset 4.1 shows that the MVO algorithm has a potential of estimating the trajectory correctly. Still, the wrong estimation in rotation results in that the accumulated position error grows rapidly after only two ~ 90° turns, as shown from the results in sequence 2, figure 4.2. However, when evaluating the results with the final use case in mind, the mini ROV will not perform any massive rotations as the goal is to keep the camera fixed to an area or object of interest. The results shows good estimations in translation, which is key.

The fluctuations experienced in the in-air test (figure 4.3) can be a result of a non-steady applied surge motion. As the test was conducted in air, the ROV could obviously not move around on its own. The surge motion was achieved by placing the drone on a trolley, and push it through the hallway. Hence, there is a high probability that the velocity was not constant throughout the test. Unwanted sway motions may also have occurred resulting in the fluctuations. A wrong scale parameter may also amplify the fluctuations in sway. The low amount of detected features on the white walls in the hallway can also affect the estimation, and contribute to the offset.

In all of the tests conduced in the MC-lab there is a big offset in heave motion, especially the first two scenarios. However, the depth estimates would not be used in a further dynamic positioning implementation, since the underwater drone is equipped with a depth sensor which is more reliable.

The unstable rotation estimations from the MVO algorithm is a clear cause of the instability in the MVO algorithm. A way of compensating for this instability would be to compute a rotation matrix from the angle measurements from the digital compass. The digital compass in the Blueye Pioneer has a drift of around 1 degree per minute. This will of course be a factor that will affect the compensation. However, while performing dynamic positioning with the Blueye Pioneer the elapsed time during a dynamic positioning maneuver will in most cases be in the magnitude of a few minutes. Hence, a few degrees offset will not affect the user experience and may be considered as good enough. Compensation of the rotation matrix was not considered in this thesis, as the main focus was to reveal the capabilities of a pure monocular visual odometry implementation.

The ability of calculating a correct scale parameter is one of the main problems when working with monocular visual odometry. The method used in the presented MVO algorithm is based on several assumptions. The effect of these is evident through the results. The scale is only calculated in heave (z - direction), and is assumed equal in the other directions. The scale should take into account movement in all directions, as the scale calculation for the KITTI dataset in equation (3.11). The scale is also kept constant throughout the different tests. During the different scenarios the drone where manually controlled, hence the velocity was not kept constant constant during the whole test. It is reasonable to assume that the calculated translations must have a higher scale parameter when the drone moves with higher velocities compared to when the drone is standing still. That will not be the case with the scale estimation procedure used.

One of the main challenges with the use of monocular visual odometry in combination with station keeping is the prevention of drift. Rotational and translational motion is estimated at the same time with the used method. However, a main source of drift in visual odometry is inaccuracy in the rotation estimate, both seen from the results obtained, and stated in [62]. This inaccuracy results in unreliable position estimates. The current position of the camera is estimated in the VO method by accumulating the motions between each image. The matched features used to determine the rotation between frames is not stored or remembered in any sort of way. Therefore, the different rotations can be characterised as individual contributions which added together results in the current camera pose. Hence, the drift accumulates as the rotation calculation have no feedback or other way of correcting itself. In a station keeping point of view, a structure from motion or SLAM method would therefore bee a better choice. In a SLAM based method, a previous visited location can be detected through global map optimisation using techniques like bundle adjustment or loop closure with pose graph optimisation [63]. This is presented more thoroughly in section 6.2.

5.2 General comments

A challenge when operating in underwater environments is the loss of features and degradation of image quality due to turbidity, scattering and attenuation. Even though the 5-point algorithm used to obtain the essential matrix, only need five distinctive points [56], the probability of a correct solution increases with more features. If a large number of features is detected within a small area of the image, the information contribution from the different features at the same area would not contribute to as accurate pose estimation as if the features where more spread out across the image frame. Comparing the results from the in-air evaluation (Figure 4.3) and the pool results (section 3.7.3), this may bee one of the causes of the wrong estimates in the results form the underwater tests. A lot of the features detected in the underwater tests, where features on the object placed in the pool. These features are therefore concentrated on a small part of the image, when the underwater drone is far away from the object, and may therefore result in an unreliable rotation estimation.

Another important aspect of the computer vision problem is with how much certainty can we guarantee a correct match between features. For the implementation presented in this thesis, there will be incorrect matches that will affect the pose estimation. In the process from feature detection to the relative pose obtained form the essential matrix, there are two mechanisms that filters out the outliers. During the matching process, the nearest-neighbour search with a ration test is performed. However, this method may lead to incorrect matches when there are similar features in the image. This may occur in underwater images as stated above, and in section 1.2.

The other mechanism is related to the outliers relevance to the pose estimation, instead of filtering them out of of the matching sequence. By applying RANSAC when computing the fundamental matrix, the possible solutions affected by the wrong matches will appear as an outlier and contribute to a solution with a lower probability value in the RANSAC algorithm. Hence, the solution consisting the wrong match will be discarded. However, if a large amount of the matches computed are wrong the matches will consist of outliers, and RANSAC will be affected. Hence, the outcome will be wrong translation and rotation calculations. There is no guarantee that a random subset of inliers will be picked by RANSAC. The probability of the algorithm succeeding highly depends on the proportion of inliers in the data, as well as the choice of the different algorithm parameters such as threshold and the probability limit.

Underwater imagery introduces challenges w.r.t. lighting and lens distortion which can cause the standard pinhole model to be inaccurate or invalid [64]. Due to air particles in the water the refractive index will change. On the Blueye Pioneer a flat-panel window is used for the underwater camera housing. Flat ports introduce significant distortions due to the refraction at the air-glass and glass-water interfaces [65]. Instead, the Pinax Model introduced by Łuczyński et al. (2017) which combines the aspects of a virtual pinhole model with the projection function for the axial camera model can be used for accurate and efficient refraction correction. This is not done in this thesis, but is something worth looking into in further development.

The performance and accuracy of the camera parameters obtained from the camera calibration procedure can also vary. If the calibration is performed with the checkerboard pattern at a constant distance, the calibration parameters is linearised around that distance. If the camera then operates with an object at a distance closer or further away than the distance used during calibration, this will affect the performance. Based on talks with experts, the influence of not calibrating the camera at the same distance as for the MVO application may vary, and may not even have any noticeable difference. However, it may be a factor that contributes to the total error.

Rolling shutter effect can cause distortions that may introduce non-trivial errors in the VO system [63]. The camera on the Blueye Pioneer has an electronic rolling shutter. This means that each row of the sensor array is exposed at a different time, resulting in the so-called rolling shutter. If the video camera or the object moves during image capturing, the rolling shutter effect can cause great geometric distortions in the image. This can result in that the object appears slanted, sheared, shrunk, elongated or even arbitrarily deformed [66]. The rolling shutter effect is more visible when the velocities are higher, or in combination with high rotational velocities. High velocities will not happen when the MVO algorithm is used for dynamic positioning, hence this effect may not be as decisive as the other topics discussed. There are also different methods to model and compensate for rolling shutter effect [66].

The video camera on the Blueye Pioneer can deliver video with 30 frames per second. For a slow-moving ROV, the features found in one image will therefore be located at a small distance from the features located in the following image. One way to speed up the MVO algorithm, and to filter out outliers, is to alter the algorithm to check for matches in area around the feature, expanding for the previously known position. This method will probably fail sometimes, resulting in that all features found in the image needs to be matched. Nevertheless, this method will be equally fast or faster than the implemented method presented in this thesis, as the search space for a match is narrowed down in most cases. This matching approach is presented in [67], although it is used in a stereo camera-scheme. An outlier that may be considered a match when the position of the feature is not taken into account may also be filtered out with this approach.

Chapter 6

Conclusion

6.1 Concluding Remarks

Motion estimation of a mini ROV is implemented through a monocular visual odometry (MVO) application. The MVO-algorithm consists of a feature-based motion estimation, with ORB as feature detector and descriptor. Through comparison with other fundamental feature-based detectors, ORB shown best results in terms of computational efficiency, robustness and accuracy in underwater environments.

The MVO-algorithm is implemented in Python using the OpenCV computer vision library in a ROS software framework. The application manage to run with a computational speed processing 10 frames per second in the given environments, which is considered successful and advantageous for accurate real-time motion estimation. The tests conducted through the different test scenarios, both above and under water and on prerecorded datasets, shows that the algorithm manages to provide a successful motion estimation in translation when performed in air. However, the errors in the rotational estimation results in high accumulated errors over time. The underwater environment adds additional challenges with respect to feature detection and robust matching, which results in that the motion estimation errors become too large too quickly for the implemented MVO-algorithm to be used for the intended dynamic positioning purpose.

The problem of determining a correct scale parameter in monocular camera scheme was solved with use of the depth sensor equipped on the mini ROV. However, the results shows that a more sophisticated solution should be considered in order to improve the accuracy of the MVO algorithm.

To improve the MVO-algorithm's motion estimation a visual inertial odometry scheme with use of a MEMS IMU was tried carried out. An online self calibration procedure with an error state Kalman Filter (ESKF) was conducted. Due to challenges of distinguish noise and correct measurements from the MEMS IMU in an ESKF manner, the implementation was nonsuccessful and not used in the presented MVO algorithm.

6.2 Further Work

This thesis has implemented a motion estimation application through monocular visual odometry. The main purpose of the motion estimation was to be used for dynamic positioning of a mini ROV. However, through the research done in this thesis, other approaches or additional features should be considered in order to get more accurate results.

The use of an IMU within the visual odometry system, so-called visual inertial odometry (VIO), could improve the motion estimation and give additional benefits. This could be done either through a filter approach attempted in this thesis, or a smoothing approach through non-linear optimisation. There are also many public VIO-pipelines available, such as the ones presented in [21], which are worth looking into even though the majority are made for arial vehicles.

Another approach of determining the scale factor then the one presented in this thesis should also be considered. Through combined measurements from an IMU and a pressure sensor, an online estimation of the scale factor could be achieved, as stated in [20]. The calculation of a correct scale parameter is a central part of an accurate MVO-algorithm.

Structure form motion (SfM) is the process of determining the 3D structure of a scene or an object, from 2D images. By moving around an object, or around in a scene, a lot of three-dimensional information can be collected. A unique metrical reconstruction of an object is known possible if four points is seen from three different perspectives [68]. Three dimensional information is obtained by looking at the correspondence between images gathered. These correspondences is found by matching features in the same manner as done in VO. The features are tracked from one image to the next to calculate the transformation and rotation of the camera.

When corresponding features are found throughout several frames, the 3D coordinates is updated in order to get a more correct representation of the scene. For a station keeping problem, we can assume that the scene is static and objects detected are stationary. When the 3D coordinates are known with a certain probability, the estimation of the camera position can be done with respect to the known features. In this manner, there is always a known reference in the image, which the pose estimation can use as a base. Each position estimate is calculated individually, avoiding the problem of accumulated errors as obtained in a standard VO approach.

As described in chapter 5, a camera calibration procedure that take into consideration the refraction at the air-glass and glass-water interfaces would result in more accurate intrinsic parameters. The resulting effect of using a different calibration model, e.g. the Pinax Model [65], may not be decisive for the end result, but worth looking into for further development.

Bibliography

[1] NOAA National Oceanic and Atmospheric Administration. How much of the ocean have we explored? `https://oceanservice.noaa.gov/facts/exploration.html`. Online, accessed 2018-12-03.

[2] Steven W Moore, Harry Bohm, Vickie Jensen, and Nola Johnston. *Underwater robotics: science, design & fabrication.* Marine Advanced Technology Education (MATE) Center Monterey, CA, 2010.

[3] Robert D Christ and Robert L Wernli Sr. *The ROV manual: a user guide for remotely operated vehicles.* Butterworth-Heinemann, 2013.

[4] Ingrid Schjølberg, Tor B Gjersvik, Aksel A Transeth, and Ingrid B Utne. Next generation subsea inspection, maintenance and repair operations. *IFAC-PapersOnLine*, 49(23):434–439, 2016.

[5] Sheldon Rubin. Mini-ROVs, going where no ROV has gone before. In *2013 OCEANS-San Diego*, pages 1–4. IEEE, 2013.

[6] Peter Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*, volume 73 of *Springer Tracts in Advanced Robotics.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[7] Bo Zhang. Computer vision vs. human vision. *Cognitive Informatics (ICCI), 2010 9th IEEE International Conference on*, pages 3–3, 2010.

[8] Hans P Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical report, STANFORD UNIV CA DEPT OF COMPUTER SCIENCE, 1980.

[9] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

[10] Christopher G Harris, Mike Stephens, et al. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[11] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. Ieee, 2004.

[12] Richard Szeliski. *Computer Vision: Algorithms and Applications.* Texts in Computer Science. London, 2011.

[13] Şahin Işık. A comparative evaluation of well-known feature detectors and descriptors. *International Journal of Applied Mathematics, Electronics and Computers*, 3(1):1–6, 2014.

[14] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[15] Deqing Sun, Stefan Roth, and Michael 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. IEEE, 2010.

[16] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[17] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2016.

[18] Joan Sola. Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.

[19] Francisco Bonin-Font, Miquel Massot-Campos, Pep Negre-Carrasco, Gabriel Oliver-Codina, and Joan Beltran. Inertial sensor self-calibration in a visually-aided navigation approach for a micro-auv. *Sensors*, 15(1):1825–1860, 2015.

[20] Vincent Creuze. Monocular Odometry for Underwater Vehicles with Online Estimation of the Scale Factor. In *IFAC 2017 World Congress*, Toulouse, France, July 2017.

[21] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2502–2509. IEEE, 2018.

[22] Alberto Quattrini Li, Adem Coskun, Sean M Doherty, Shervin Ghasemlou, Apoorv S Jagtap, M Modasshir, Sharmin Rahman, A Singh, Marios Xanthidis, Jason M O'Kane, et al. Experimental comparison of open source vision-based state estimation algorithms. In *International Symposium on Experimental Robotics*, pages 775–786. Springer, 2016.

[23] J-F Lots, David M Lane, Emanuele Trucco, and François Chaumette. A 2d visual servoing for underwater vehicle station keeping. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 3, pages 2767–2772. IEEE, 2001.

[24] Chao-Lin Kuo, Long-Yi Chang, Ying-Che Kuo, Chia-Hung Lin, and Kuei-Mei Lin. Visual servo control for the underwater robot station-keeping. In *2017 International Conference on Applied Electronics (AE)*, pages 1–4. IEEE, 2017.

[25] Xavier Cufí, Rafael Garcia, and Pere Ridao. An approach to vision-based station keeping for an unmanned underwater vehicle. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 799–804. IEEE, 2002.

[26] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[29] Kishore Reddy Konda and Roland Memisevic. Learning visual odometry with a convolutional network. In *VISAPP (1)*, pages 486–490, 2015.

[30] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. Undeepvo: Monocular visual odometry through unsupervised deep learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7286–7291. IEEE, 2018.

[31] Mehdi Narimani, Soroosh Nazem, and Mehdi Loueipour. Robotics vision-based system for an underwater pipeline and cable tracker. pages 1–6. IEEE Publishing, 2009.

[32] G. Santhan Kumar, Unnikrishnan V. Painumgal, M.N.V. Chaitanya Kumar, and K.H.V. Rajesh. Autonomous underwater vehicle for vision based tracking. 133:169–180, 2018.

[33] Dejun Li, Tao Zhang, and Canjun Yang. Terminal underwater docking of an autonomous underwater vehicle using one camera and one light. *Marine Technology Society Journal*, 50(6):58–68, 2016.

[34] Martin Ludvigsen, Tharindu D. M. Hewage, and Kristine Klingan. *Lecture Notes: AT-334 Arctic Marine Measurements Techniques, Operations and Transport.* 2016.

[35] CJ Funk, SB Bryant, and PJ Heckman Jr. Handbook of underwater imaging system design. Technical report, Naval Undersea Ceter San Diego CA, 1972.

[36] Martin Ludvigsen. *An ROV toolbox for optical and acoustical seabed investigations.* 2010.

[37] OpenCV team. About. `https://opencv.org/about/`. Online, accessed: 2019-05-19.

[38] Open Source Robotics Foundation. About ros. `http://www.ros.org/about-ros/`. Online, accessed: 2018-11-01.

[39] Open Source Robotics Foundation. Core components. `http://www.ros.org/core-components/`. Online, accessed: 2018-11-01.

[40] Blueye Robotics. Our story. `https://www.blueyerobotics.com/page/our-story`. Online, accessed: 2019-05-30.

[41] T.I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons, 2011.

[42] Fredrik Dukan. Rov motion control systems. 2014.

[43] Francisco Bonin-Font, Miquel Massot-Campos, Pep Lluis Negre-Carrasco, Gabriel Oliver-Codina, and Joan P Beltran. Inertial sensor self-calibration in a visually-aided navigation approach for a micro-auv. *Sensors (Basel, Switzerland)*, 15(1), 2015.

[44] Mingyang Li and Anastasios I Mourikis. High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[45] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[46] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[47] Eduardo Bayro Corrochano. *Handbook of Geometric Computing: Applications in Pattern Recognition, Computer Vision, Neuralcomputing, and Robotics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[48] Javier Civera, Andrew J Davison, and JM Martinez Montiel. Inverse depth parametrization for monocular slam. *IEEE transactions on robotics*, 24(5):932–945, 2008.

[49] Shaharyar Ahmed Khan Tareen and Zahra Saleem. A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*, pages 1–10. IEEE, 2018.

[50] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. Orb: An efficient alternative to sift or surf. In *ICCV*, volume 11, page 2. Citeseer, 2011.

[51] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[52] Paul L Rosin. Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291–307, 1999.

[53] L Ye, A Argha, BG Celler, HT Nguyen, and SW Su. Online auto-calibration of triaxial accelerometer with time-variant model structures. *Sensors and Actuators A: Physical*, 266:294–307, 2017.

[54] Stergios I Roumeliotis, Gaurav S Sukhatme, and George A Bekey. Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, volume 2, pages 1656–1663. IEEE, 1999.

[55] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.

[56] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):0756–777, 2004.

[57] Amanda Duarte, Felipe Codevilla, Joel De O Gaya, and Silvia SC Botelho. A dataset to evaluate underwater image restoration methods. In *OCEANS 2016-Shanghai*, pages 1–6. IEEE, 2016.

[58] David G Lowe et al. Object recognition from local scale-invariant features. In *iccv*, volume 99, pages 1150–1157, 1999.

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

[60] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets Robotics: The KITTI Dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[61] NTNU. Marine cybernetics laboratory (MC-lab). `https://www.ntnu.edu/imt/lab/cybernetics`. Online, accessed: 2019-12-06.

[62] Pyojin Kim, Brian Coltin, and Hyoun Jin Kim. Visual odometry with drift-free rotation estimation using indoor scene regularities. In *BMVC*, 2017.

[63] Nan Yang, Rui Wang, Xiang Gao, and Daniel Cremers. Challenges in monocular visual odometry: Photometric calibration, motion bias, and rolling shutter effect. *IEEE Robotics and Automation Letters*, 3(4):2878–2885, 2018.

[64] Tali Treibitz, Yoav Schechner, Clayton Kunz, and Hanumant Singh. Flat refractive geometry. *IEEE transactions on pattern analysis and machine intelligence*, 34(1):51–65, 2011.

[65] Tomasz Łuczyński, Max Pfingsthorn, and Andreas Birk. The pinax-model for accurate and efficient refraction correction of underwater cameras in flat-pane housings. *Ocean Engineering*, 133:9–22, 2017.

[66] Chia-Kai Liang, Li-Wen Chang, and Homer H Chen. Analysis and compensation of rolling shutter effect. *IEEE Transactions on Image Processing*, 17(8):1323–1330, 2008.

[67] Andreas Geiger, Julius Ziegler, and Christoph Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 963–968. Ieee, 2011.

[68] Jan J Koenderink and Andrea J Van Doorn. Affine structure from motion. *JOSA A*, 8(2):377–385, 1991.

Appendix A

Drone Specifications

A.1 Drone

Specs	Value
Ingress protection	IPX8
Dimensions	485 x 257 x 354 mm (LxWxH)
Weight in air	8.6 kg (with salt water ballast)
Construction	ABS enclosures, Aluminium pressure enclosures, Polycarbonate (PC) windows
Buoyancy material	HCP 30 Polymer Foam
Maximum rated depth	150 m
Forward speed at normal use	2 m/s (4 knots)
Thrusters	4 x 350 W
Run time at normal use	Approx. 2 hours
Operating temperature	-5 to +40 °C

A.2 Camera

Specs	Value
Sensor	CMOS, 1/3 inch
Shutter speed	1/30 s – 1/8000 s
Picture max resolution	2M (1920 x 1080)
Video resolution	FHD: 1920 x 1080 25/30 Fps, HD: 1280 x 720 25/30 Fps

Specs	Value
Video type	MP4
Video storage bit-rate	2 to 16 MBit/s
SD card	64 GB

A.3 Lens

Specs	Value
Image Circle	1/3”
Focal length	2.1 mm
Aperture	f/1.8
Iris Type	fixed
MOD	200mm
Resolution	MP
Angle of View (D / H / V)	170°/ 130°/ 96°
Back Focal Length	6.25 mm
Mount	M12x0.5-6g

A.4 LED lights

Specs	Value
Luminous flux	3300 Lumen
Colour temperature	5000 K
Colour rendering index (CRI)	70
Adjustable dimming	Yes

A.5 Sensors

Specs	Value
IMU	3 axis gyro & accelerometer & magnetometer
Depth sensor	Resolution: 0.2mbar
Depth sensor operating range	0 to 30 bar

Specs	Value
Temperature sensor	+/- 1°

A.6 In air calibration results

The in air calibration is done in the same procedure as described in section 3.2. The results is shown in table A.6

Table A.6: In air calibration results: Intrinsic parameters and distortion coefficients

f_x	978.36617202 pixels
f_y	975.69987506 pixels
c_x	985.08473535 pixels
c_y	541.52130078 pixels
k_1	-0.37370139
k_2	0.26899755
p_1	-0.00120655
p_2	-0.00185788
k_3	-0.1411856