NTNU
Norwegian University of Science and Technology

Exploring

Visual Egomotion Estimation In Aquacultural Fish Cages

Localization In Challenging Environments

Lars Olav Libjå

Master of science in Engineering Cybernetics

Submission Date:	June 20, 2022
Supervisors:	Annette Stahl
	Rudolf Mester
	Christian Schellewald

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

By now we are used to realtime position estimating, with systems such as GPS. These types of systems have given quick and easy access to our position. However, we can not rely on these systems everywhere. Under water for instance. There arises a challenge with the desire for positioning of underwater vehicles in aquacultural fish cages. How do we find positions without the use of GPS?

To start it is produced a dataset. This is done in order to present results from the position estimates, and control the quality of the data. The dataset is a so called synthetic dataset, which means it is produced digitally. By producing a synthetic dataset we obtain the actual positions of the camera.

Then a method for position estimating is presented. Using Fourier Analysis, a technique that uses the frequencies in an image, the motion of the camera can be estimated. In comparison, equivalent systems on land uses features in the image to obtain a position estimate. In this context features are corners, edges and other contrasting regions in the image. The use of Fourier Analysis is not unheard of in regards to position estimation, new here is the use in an aquacultural fish cage.

The results with this method shows some promise. There are still steps to be made in the development of this method in order for it to be robust and reliable, but the results show possibilities for further work.

Sammendrag

Sanntidsestimering av posisjon er en teknologi vi i dag er godt vant med. Systemer som GPS har gitt oss muligheter til å kunne finne posisjonen vår raskt og enkelt. Men det er ikke alle steder vi kan være avhengige av slike systemer. Et slikt sted er under vann. Og når det da er et ønske om å kunne finne posisjonen til undervanns fartøyer i fiskemerder, oppstår det et problem. Hvordan finne posisjoner uten bruk av GPS?

Først er det laget et dataset for å kunne lage of presentere resultater og for å få høy kvalitet på dataen. Datasettet er syntetisk, som betyr at det er laget digitalt. Ved å lage ett syntetisk datasett får vi også med den faktiske posisjonen som kameraet bevegde seg.

Deretter er en metode for å posisjonsestimering presentert. Ved bruk av Fourier Analyse, en teknikk som bruker frekvensene i bildet, kan bevegelsen til kameraet estimeres. Til sammenligning bruker tilsvarende systemer på land egenskaper i bildene brukes for å oppnå posisjonsestimater. Disse egenskapene er gjerne kanter, hjørner og andre steder med stor kontrast. Bruk av Fourier Analys er ikke uhørt i sammenheng med posisjonsestimering, det som er nytt her er bruken i en merde konstruksjon.

Resultatene fra testene med denne metoden viser at bruken av denne typen metode kan ha noe for seg. Det er fortsatt en vei å gå med utviklingen for at systemet skal være robust og pålitelig, men resultatene viser at det er en mulighet å fortsette med.

Acknowledgments

I remember a conversation I had about 14 years ago with one of my schoolmates at the time. We had started middle-school, and were talking about what we wanted to do later in life. I had already then set my sights on NTNU, and engineering studies. When I 7 years later moved to Trondheim that childhood dream had become a reality. What 13 year old me had not expected was how hard it would be. It have been the hardest years in my life. Frustratingly hard at times. I am a little angry with that 13 year old boy for having his ambitions set so high, but I am also glad he did.

There have been times, to be honest there still are times, where it have seamed like I would not be able to finish what I've started. These have been some challenging years, and to top it all, the last two years have had the world turned on its head. Life have thrown a lot of curveballs, at last it seams to be looking up.

This is where I would like to thank the people who have always supported me. To mom and dad, always supportive, always understanding. It's not been easy to be away for long periodes with injuries and surgeries. Hopefully, now I can give back some of the support you have given me.

To my siblings, Ragnhild, Sverre and Asbjørn. Thank you for being who you are. Keep being yourself. You are a joy to everyone that gets to know you. You show me that life has more to offer. Thank you for being the best siblings I could wish for, and for reminding me of who I am. You all make me strive to be better.

To my grandmother, Astrid, who doesn't really understand what it is that I study. To be fair to her, not a lot of people do. Thank you for begin supportive, interested and always proud of me.

To my niece, Astrid. I am looking forward to get to know you. I absolutely intend to be your favourite uncle.

To Gjert Ingar for all the years I've known you, 23 and counting. And thank you for the years in Trondheim. Now I'm looking forward to being your colleague.

To Arild for all the hours spent in various labs, for all the feedback, discussions and generally for being a great friend trough our time at university. I'm looking forward to our future undertakings.

And last, thanks to my supervisors, Annette, Rudolf and Christian for the guidance and input during the work on this thesis.

Lars Olav Libjå

Lars Olav Libjå
Fyresdal, 20 June 2022

Contents

Chapter 1

Introduction

There are some questions arising from the title, firstly, what does egomotion estimation in aquacultural fish farms entail? Why egomotion estimation in aquacultural fish farms? And perhaps the most important question, how? The answers to the first and second questions will serve as an introduction to this thesis, and an attempt at an answer will be given in this chapter. The same problem were also the topic of my pre-project [1], and some of the lessons from that work comes into this thesis.

To try to answer these questions, lets start by having a look at the Norwegian aquacultural industry.

1.1 Norwegian Aquaculture

The Norwegian aquacultural industry had its humble beginnings in the 1970s, the industry have had a large growth since then, and is today norways second largest exporting industry - after oil and gas[2]. While norway is a relaive small producer of farmed fish in comparison to other countries, norway still are the largest producer of atlantic slamon in the world[3]. With large exports and high salmon prices, the fish farming companies have very good margins.

It is not, however, just sunshine and rainbows. Like with any other industry where there are large growth in a relative short time, there are bound to some growing pains. Lice and illnesses are one of the problems that have come about as the amount of farmed fish have increased. Another problem that fish farmers face are escapes. In 2020 about 500 000 wild atlantic salmon came back to the norwegian shore, which is half of the numbers in the 1980s [4]. One of the threats to the wild salmon population is the escaped farmed fish, which have been bred primarily for fast growth. Mixing these genes into the wild population is not desirable. Thus the fish farmers have an obligation to limit the number of escaped fish. There are also increasing preassure from different interest groups and also the norwegian government to limit the escaped fish.

There are more than one way for fish to escape the cages. Errors while moving the fish beging one of them, but the one we would like to focus on here is holes in the cages. Larger holes than intended, that is. These types of damages to the cage happens from time to time, some are due to boats and other equipment tearing holes, others can be caused by weather or flaws in the constructions of the cages. Regardless of how they come about it is important to get them fixed before large amount of escapes can happen.

Today most of the inspection of cages are done with Remotely Operated Vehicle (ROV)s, and often in conjunction with cleaning operations. These are hard, manual tasks for the operator, having to both control the ROV and inspect the state of the cage. So here is where the *why* question comes in. Estimation the position of the ROV in the cage can be of benefit to the inspection. If damages are reported, the position can be pinpointed for easier rapairs. Also, if a cage have damages in the same region on multiple occations, this might be due to defects in the cage construction, and could therfore be identified and fixed. With this technology, automatic inspection could also be implemented, possibly resulting in more frequent inspections and smaller chance of escapes.

1.2 Visual sensory information

To at all be able to estimate the position of an agent in our environment, some sort of sensory information is needed, be it to humans or computers. In this thesis we will be working with image data, or visual information from a camera to be more precise.

Using cameras as a means for position estimation is very relatable, it's what most of us humans do. We deal with visual sensory information from the moment we wake, until we go to sleep. Even when we sleep our visual cortex is active [5]. Computer science research is heavily inspired by biological systems, and this is no different.

The human visual sensory system is a complex bit of biology, and it is perhaps our most relied upon sensory system (maybe except for our tactile senses), especially when it comes to navigation. Would you be able to get to the bathroom, assuming you are not reading this in the bathroom, from where you are sitting without your eyes? Perhaps, but the loss of sight would impare your ability to navigate alot more than the loss of small or taste.

There is another good argument for using cameras as a source for localization data, price and availability. Cameras have become very cheap, compared to more complex sensor systems such as LIDAR they are only a fraction of the cost. And then there is the availability. Cameras are everywhere. Look around, I would guess there are at least two cameras less than a meter away. Being able to use an ordinary, cheap and easily accessible sensor to perform localization is one of the reasons extensive research has been done in the field of Visual Simultaneous Localization and Mapping (VSLAM) (see section 2.2.3).

1.3 Synthetic data

All research needs good data in order to produce good results. When dealing with computer vision, this data have historically been collected using cameras in the real world. Recently, however, there have been a shift in how we view data collection. Large amount of data is often needed, and this can be impractical to manually collect. Often, as in case of this thesis, ground-truth camera trajectories are vital for evaluating the results. Producing accurate ground-truths when dealing with underwater environments are very hard if not outright impossible. With recent advancements in photorealistic imaging algorithms, and great leaps in hardware accelerated graphics rendering[6], producing realistic synthesized image data have become much more practical than in the past.

In his thesis, Zwilgmeyer [7], creates an underwater dataset, focused on seabed pipe inspection. The thesis describes the different data collection techniques, from the egomotion of the camera, Inertial Measurment Unit (IMU) simulation, to the modelling and rendering of the underwater scene. His work describes techniques for creating realistic underwater data with an accurate ground-truth. The dataset produced by that work is presented in Zwilgmeyer et al. [8], and is a benchmarking dataset for underwater vision algorithms.

1.4 Problems with underwater positioning systems

Positioning systems on land have an advantage that underwater systems don't have, Global Navigation Satellite Systems (GNSS), GNSS systems such as Global Positioning System (GPS) have become the staple of positioning on land. But the attenuation of the signals is too large in water to rely on for underwater positioning. There exists acoustic systems underwater positioning, such as the system presented in [9]. One problem with such systems is the need for extra infrastructure. There needs to be placed transducers to detect acoustic signals in the area where positioning is needed.

When it comes to underwater Visual Odometry (VO) and Simultaneous Localization and Mapping (SLAM) there are some complexities that do not apply to the same extent when dealing with egomotion estimation on land. Firstly there is the problem that underwater environments are complex, there are lots of particles clouding the scene, so the noise in the data will be a lot higher than it would be without the water [10].

Then there is the problem of data association [10], which will be the main problem this thesis deals with. Data association is how we correlate the current information from a scene to previous data collected. For underwater SLAM applications this can be a big problem, and specially for the environment this thesis is dealing with. Highly monotone environments, where features are the same, struggles with data association.

Chapter 2

Previous Work

At its core, this thesis deals with position estimation, more precisely egomotion estimation, to the effect of relative position estimation. Egomotion is the motion of an agent in an environment, the agent can really be anything moving in a n-dimensional environment. In the context of computer vision however, the agent is a camera, and the egomotion is defined relative to static landmarks in a three dimensional environment.

The literature on egomotion estimation is quite extensive, and some of the techniques will be presented later in this chapter (section 2.2). Early work presented in the literature are, understandably, focused primarily on applications on land, and not on underwater environments. However in later years, there have been more publications on underwater systems that deal with the same problem in underwater environments [10]. There are complications when it comes to underwater environments that will be further discussed in section 1.4

When it comes to work on the specific topic of this thesis, positioning in an aquacultural cage, the literature is very sparse. This chapter will introduce the literature and techniques used in this thesis.

2.1 Camera

When dealing with computer vision we need to define a mathematical model of the camera. In a lot of computer vision tasks a so called *pinhole* camera model is used. The name comes from the fact that the apature of the camera is a *pinhole*, and in the mathematical models have an infinitesimal size. Such cameras were an early technology, and some of the first to ever exist. The fact that such cameras don't have lenses, and larger apature size are what makes them ideal for our model.

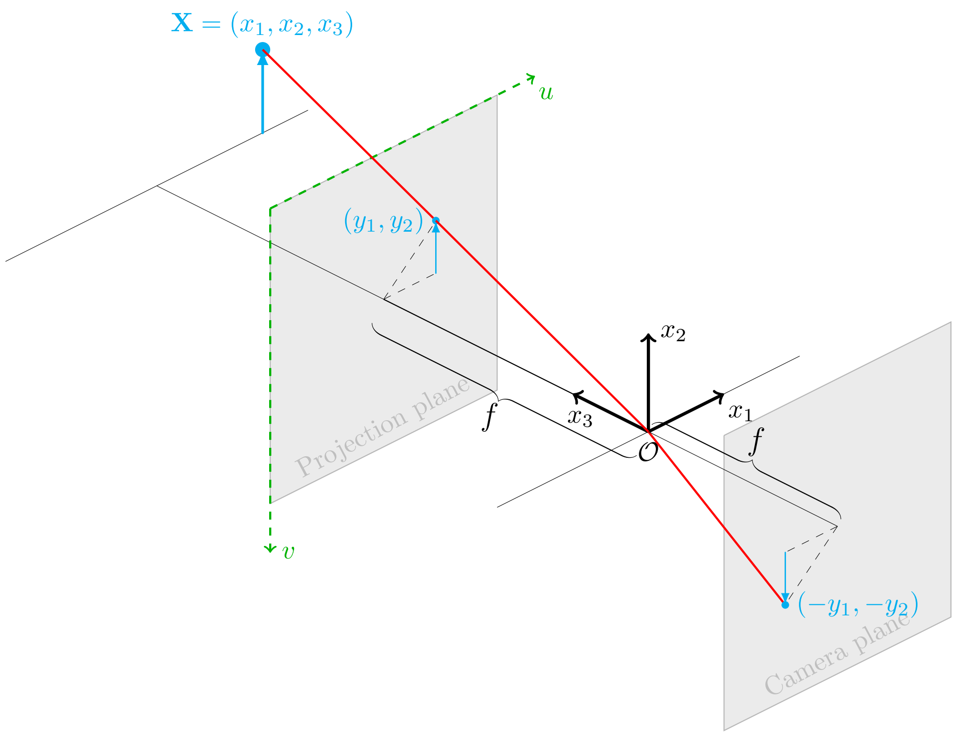

Figure 2.1

From figure 2.1 the camera model can be expressed quiet easily with the use of similar triangles. The point $\mathcal{O}$ is the location of the camera apature, where every ray is projected trough to the camera plane, this results in the image being flipped, to make calculations easier a projection plane is used. This is a virtual plane the same distance from the apature as the camera plane, but in fron of the camera. The coordinates u and v are image coordinates, meaning they are discrete. First the point X is projected to continous camera coordinates, $\tilde{x}_1$ and $\tilde{x}_2$.

$$\tilde{x}_1 = \frac{x_1}{x_3} f, \quad \tilde{x}_2 = \frac{x_2}{x_3} f \tag{2.1}$$

written as vectors

$$\begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} = \frac{f}{x_3} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.2}$$

Writing this relationship with homogeneous coordinates we get the following set of linear equations

$$\begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} \tag{2.3}$$

Transforming the origin of the camera coordinate system, which is at the center of the camera, to the origin of the image coordinate system, we can express the camera model as

$$\tilde{\mathbf{x}} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{X}_c \tag{2.4}$$

where c_x and c_y is the offset from the center of the camera to the origin (upper left corner) of the image coordinate system. To get the image points to pixle distance a scaling factor is introduced $f_x = s_x f$ and $f_y = s_y f$

$$\tilde{\mathbf{x}} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \mathbf{X}_c \tag{2.5}$$

This model transforms 3D coordinates to image coordinates. One problem with this model is that the 3D coordinates is represented in respect to the camera coordinate frame, where a better solution would be to represent the 3D coordinates in a world frame. The model can be written as

$$\tilde{\mathbf{x}} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \left[\mathbf{R}|\mathbf{t}\right] \mathbf{X}_w \tag{2.6}$$

where $\mathbf{R}$ is the rotation matrix, and $\mathbf{t}$ is the translation vector from the camera frame to the world frame, and $\mathbf{K}$ is the camera intrinsic matrix. Writing $\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$ the equation can be written as

$$\tilde{\mathbf{x}} = \mathbf{P}\mathbf{X}_w \tag{2.7}$$

2.2 Visual Odometry and Simultaneous Localization and Mapping

The core concepts used in this thesis are technologies which have been part of the literature for more than three decades. The fundamental theory was developed by Smith, R., Self, M., and Cheeseman, P. in 1986 [11; 12]. They contributed the early theory to what we today know as Visual Odometry (VO) and Simultaneous Localization and Mapping (SLAM).

2.2.1 Visual Odometry

The word *odmetry* comes from the greek words *odos* meaning "route" and *metron* meaning "measure"[13]. It translates to route-measuring, which is quiet accurate. Odometry is the method of using data from sensors in order to estimate position changes. By adding the word visual to it we specify that the estimation is done using visual information. By utilizing the information provided in a sequence of images a VO algorithm can estimate the change in position of the camera from the start.

The next section will be talking about SLAM, which in some cases can be seen as an extention to the more traditional odometry. With the main difference being that SLAM methods produce a map and perform optimization of previous estimated positions.

2.2.2 SLAM

As mentioned, positioning systems on land have the added advantage that they can rely on already existing infrastructure, and it is an easily accessible global system. However, more advanced systems use more than just their global coordinates, they rely on other sensor data to give a more accurate position, cameras being one of them.

There are traditionally two different visual slam systems, direct and feature-based methods. The difference comes down to how the images are processed. To be able to both build a map, and estimate the location in that map, the images need to be processed in such a way that this information can be extracted. To a human this might seem like a trivial task, but in reality it is a complex task. How do we distinguish between objects? How do we say that one previously observed object is the same as another? The simple answer is easily recognized shapes, corners, edges and contrasting regions.

SLAM is the computational problem of figuring out where an agent is located, whilst building a map of the same surrounding environment. Understanding the problem is trivial, however a general solution to the problem is quiet hard to produce. Although the literature is mature, and SLAM have a lots of practical applications in many fields[14], there are still cases where state-of-the-art solutions struggle to give satisfying results[15]. One example of where state-of-the-art SLAM, and specially VSLAM falls short, is in environments where there is not a lot of structure, or where the structure has low variance. Structure in VSLAM implies edges, corners and other contrasting features. In this thesis we will be working in environments where the structure has low variance.

All SLAM methods start with some sort of sensor data. This can be radar, lidar, sonar, camera, IMU, encoders, etc., or a combination. The sensors used are dependent the application. For most applications some sort of environment sensing is needed. Cameras are frequently used because of the low price of these sensors compared to e.g. lidar systems. The research of these types of SLAM algorithms, often refereed to as VSLAM, are therefore also extensive.

If high precision localization is necessary, or when we are dealing with challenging environments, we would like to augment the camera or other visual data with other sensor data e.g. IMU data, in a sensor fusion algorithm. Sensor fusion is the process of combining two or more data sources in order to decrease the uncertainty in the measurements from only one of these data sources[16].

2.2.3 Visual SLAM

When dealing with cameras in a VSLAM method, we need to extract some sort of information from the image sequence. There are two main methods of doing this, what we call direct and feature-based methods. Direct methods use, as the name implies, the images directly. Image intensities are used to minimize the photometric error between frames, this is then used to construct a map of the environment, and place the agent in this map.

Direct methods usually produce dense maps, meaning the map geometry have more definition. Information about the surfaces in the environment are included in dense maps. Compared to sparse maps, where key points define the map. These key points are usually edges, corners or other contrasting features in the environment. This leads to the other mathod of VSLAM, feature-based methods, which usually produce sparse maps. This distinction isn't a the hard line separating direct and feature based methods. Some feature-based methods can produce dense maps, and direct methods can produce sparse maps, but it is a good rule of thumb.

Feature-based methods, use features in each image to construct the geometry of the environment. A feature in a image consists of a key point and a descriptor. Key points are points of interest in the image, usually occurring on edges and corners. A descriptor, describing the key point, is then calculated for each key point. These descriptors can then be used to find the same key points in later frames. If we have enough matching key points in two frames, we can construct the epipolar geometry of these key points.

2.2.4 State of the art VSLAM methods

Recent developments to the state-of-the-art VSLAM methods have consisted of optimizing the algorithms, and decreasing the computational cost of the SLAM problem, to where online computation is possible.

ORB-SLAM

One of the most known SLAM methods is ORB-SLAM[17], and its successors ORB-SLAM2[18] and ORB-SLAM3[19]. The first one, published in 2015, being a novel approach to the VSLAM problem. Setting out to develop a SLAM system capable of large-scale real-time operation[17]. ORB-SLAM is a feature based method, using Oriented BRIEF Rotated FAST (ORB) features[20] in all stages of the algorithm. ORB features have a low computational cost associated with them. The binary descriptors being easy to compute and match, and results show they are an alternative to what have been the industry standards Scale Invariant Feature Transform (SIFT) and Speeded[sic]-Up Robust Features (SURF). The low computational cost is obviously important for the goals set for ORB-SLAM. Another novel approach ORB-SLAM took was to separate the different stages of the SLAM system in separate threads, making the real-time goal of the system possible.

Later iterations of ORB-SLAM improves on the original, by including more options for sensor data integration[18; 19]. The first version only uses monocular data, whereas the second release implements more camera setups, such as RGB-D and stereo cameras. The latest version includes visual-inertial slam methods, augmenting the visual data with IMU information.

The different ORB-SLAM versions have shown good performance on the industry standard banchmarking datasets, KITTI[21] and TUM RGB-D[22], these datasets, however, are terrestrial in structured environments.

2.3 Coordinate Systems

This thesis will be working with different coordinate systems, and it is necessary to know the relative relation between these. One coordinate system can be expressed in another, and the following sections will define each coordinate system this thesis will be working with.

2.3.1 World Coordinate System

The world coordinate system is defined with its origin in the center and at the surface of the fish-cage. The axes of the coordinate system is defined with the z-axis being normal to the surface of the cage, the x-axis is the magnetic north, and the y-axis east.

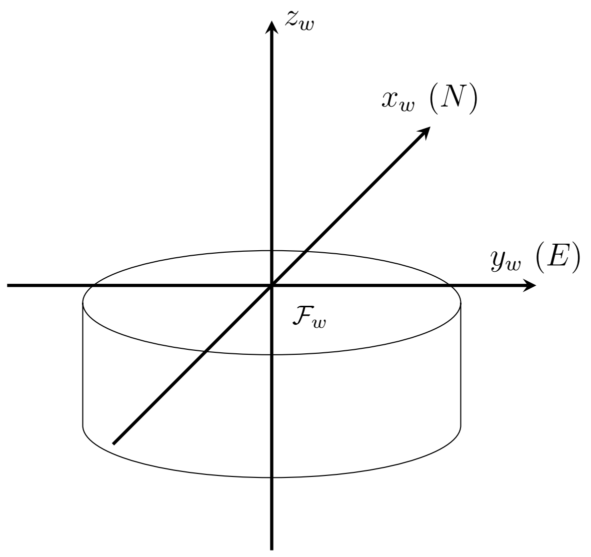

Figure 2.2: The world coordinate system plotted in a cylinder representing the cage.

2.3.2 Camera Coordinate System

The camera coordinate system is defined, as cameras usually are, with the z-axis normal to the camera sensor at its origin, pointing out of the camera. The x and y axes are defined the same direction as image coordinates usually are, with x pointing to the right, and y pointing down, looking out of the camera. Section 2.1 describes the relation between the camera frame and the world coordinate frame.

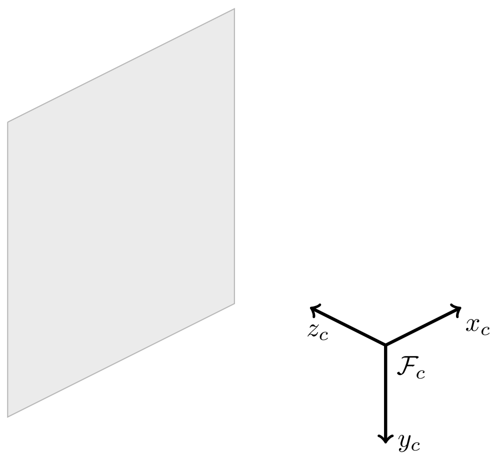

Figure 2.3

2.4 2D to 3D reconstruction

When doing VSLAM we want to construct an accurate map of the environment we are tracking. When using just visual information to this effect, we need a way to construct a three-dimensional map from a two-dimensional image. In principle this is quiet straight forward. Assuming the environment is just a planar face; with only three distinct, recognizable points, we can find the plane going through these points and then obtain the rotation between the two images. In practice, the assumption that the environment is planar is very limiting. To deal with non-planar environments, the theoretical minimum point-correspondences needed is five, in practice however, the most common method, aptly named eight-point algorithm[23], uses a minimum of eight point-correspondences.

2.5 Image Preprocessing

Tasks that use Computer Vision (CV) often need to process images to be used in different vision algorithms. Image preprocessing can include cropping, filtering, and binarization. This section will introduce the image preprocessing techniques used in the later parts of this thesis.

Using the gray-scale image, or the image without color channels, is common when working with the structure in a image, as it is easier and less memory required to work with, and the underlying structure is preserved in the gray-scale image. To convert a RGB image to gray-scale a weighted mean value of the pixel value is calculated for each pixel.

2.5.1 Binarization

Binarization is a technique relying on the pixel-intensities of the image. Making dark pixels black (0) and light pixels white (1). A simple implementation to achieve this is to use a threshold value and make all pixels above that value white, and all those below black. A more sophisticated method is to use what is called *adaptive thresholding*. The adaptive thresholding methods works by using a small region to determine the threshold value for each pixel, and then determine if it should be white or black. OpenCV [24] implements two version of this method, one using a mean thersholding method, and one using a Gaussian-weighted sum method. A third method is the Otsu's Binarization method [25], which is specially useful on bimodal images. Where the threshold value is determined to be the middle of two peaks in the image histogram.

Figure 2.4 shows the three methods described above. The Otsu thresholding and the global thresholding is very similar, this is due to the the selected value,

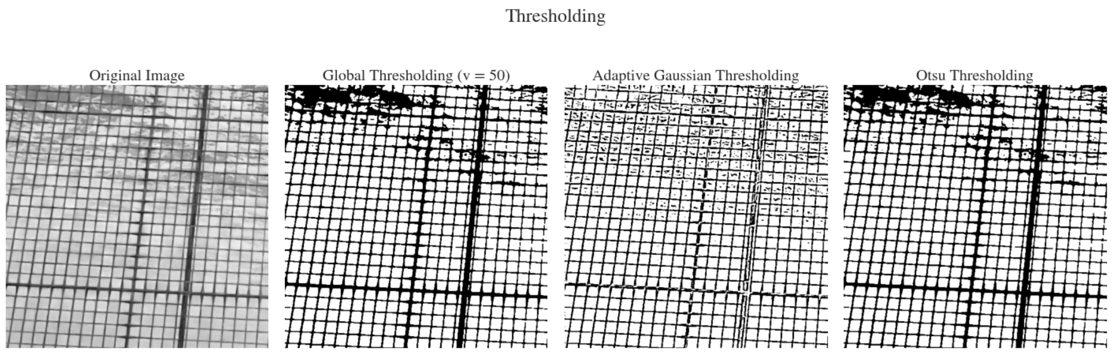

Figure 2.4: Different thresholding methods showed on the same original image.

($v = 50$), being close or equal to the value calculated by the Otsu method. Apart from the fact that the Otsu method calculates the threshold value, the two methods are identical. The global threshold value from 2.4 was chosen to get the desired effect, but deciding a static threshold value for multiple frames in a sequence where the illumination change would not yield good results, so the adaptive methods are preferable in those cases.

2.5.2 Windowing

In signal processing, windowing functions are useful to isolate the parts of the signal to be analysed. Windowing functions have a real-valued *window*, and are zero-valued outside of this window. Simple element-wise multiplication can be used to apply a window in the spatial domain of an gray scale image.

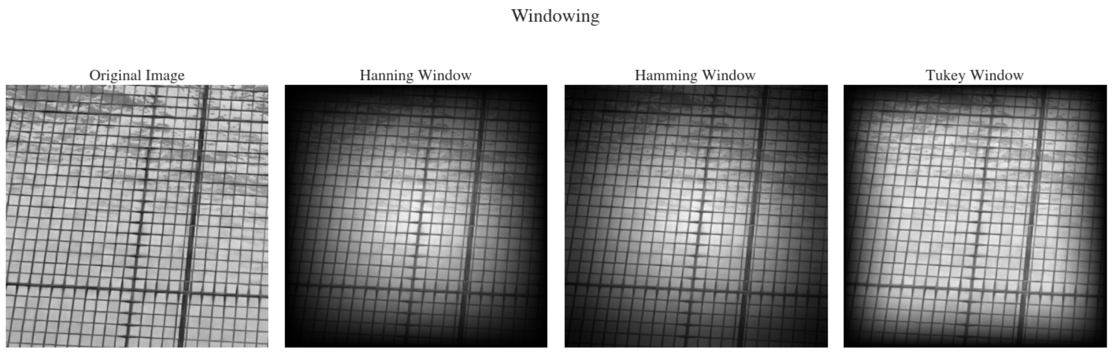

Figure 2.5: Three different windowing functions applied to the same original image.

Windowing is a useful tool when dealing with edge-effects, such as in phase-correlation (section 2.6.1), where the technique only works on a circular shifted image, and not on simple linear shifts. Windowing functions are used in this case to avoid edge effects, and produce better results.

2.6 The Fourier Transform

Spatial analysis is often the most approachable, straight forward method for signal analysis. However, in some cases it does not yield the desired results. In that instance another tool is the Fourier Transform (FT) and frequency analysis[26].

In the case of image analysis the two dimensional Fourier transform is used, and it can be stated mathematically as

$$\hat{\mathbf{x}}(u,v) = \mathcal{F}(\mathbf{x}) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} \mathbf{x}(x,y)e^{-i2\pi(\frac{ux}{N}+\frac{vy}{N})} \tag{2.8}$$

where $\mathbf{x}$ is the NxN image, u and v are image coordinates, and N is typically a power of two to increase the performance of the Fast Fourier Transform (FFT) [27]. $\hat{\mathbf{x}}(u,v)$ is the Fourier transformed signal. Applying this transform to an image will give the frequency transform of the image. The frequency representation of a signal have many properties, and some of those properties have practical applications in the context of this thesis.

2.6.1 Phase Correlation

Phase Correlation is a technique that can be used to find translation, rotation and scale variation between two images [28]. Phase correlation makes use of the Fourier transform of two images, computing the cross-power spectrum and determining the displacement from one image to the other.

The cross-power spectrum of two images $\mathbf{x}$ and $\mathbf{x}'$ with FTs $\mathcal{F}(\mathbf{x}) = \hat{\mathbf{x}}$ and $\mathcal{F}(\mathbf{x}') = \hat{\mathbf{x}}'$ is defined as

$$\hat{\mathbf{p}} = \frac{\hat{\mathbf{x}} \odot \bar{\hat{\mathbf{x}}}'}{|\hat{\mathbf{x}} \odot \bar{\hat{\mathbf{x}}}'|} \tag{2.9}$$

where $\bar{\mathbf{x}}$ is the complex conjugate of $\mathbf{x}$ and $\odot$ is the element-wise product.

If now $\mathbf{x}_1(x,y)$ and $\mathbf{x}_2(x,y)$ are two image signals where $\mathbf{x}_2(x,y) = \mathbf{x}_1(x-x_0, y-y_0)$ is the shifted signal of $\mathbf{x}_1$. Their Fourier transforms will be

$$\mathcal{F}(\mathbf{x}_1), \quad \text{and} \quad \mathcal{F}(\mathbf{x}_2) = \mathcal{F}(\mathbf{x}_1)e^{-i2\pi(\frac{ux_0}{N}+\frac{vy_0}{N})} \tag{2.10}$$

calculating the cross-power spectrum give

$$\hat{\mathbf{p}} = \frac{\hat{\mathbf{x}}_1 \odot \bar{\hat{\mathbf{x}}}_\mathbf{1} e^{i2\pi(\frac{ux_0}{N}+\frac{vy_0}{N})}}{|\hat{\mathbf{x}}_1 \odot \bar{\hat{\mathbf{x}}}_\mathbf{1} e^{i2\pi(\frac{ux_0}{N}+\frac{vy_0}{N})}|} = e^{i2\pi(\frac{ux_0}{N}+\frac{vy_0}{N})} \tag{2.11}$$

applying the inverse Fourier transform to $\hat{\mathbf{p}}$ give a peak $\mathbf{p}(x,y) = \delta(x+x_0, y+y_0)$

at the displacement.

The same principles applied above can be used to determine rotation and scale, by using log-polar coordinates, as it can be shown that scaling is displacement on a logarithmic scale, and rotation is displacement with polar coordinates [28].

2.6.2 Net Pose Estimation

In their article Schellewald, Stahl and Kelasidi [29] describe a technique for estimating the pose of the net relative to the camera in an aquacultural fish cage. This technique uses the Fourier space to calculate the pixel size of the net grid. With prior knowledge of the actual size of the grid, and the calibrated camera matrix, the net pose can be estimated using a PnP-solver.

The repeating pattern of the net grid will show up as peaks in the FT. With regularity checks of these peaks, presence of a grid can be determined and two vectors in the frequency space $\mathbf{k}_2$ and $\mathbf{k}_2$ can be found. These vectors relate to the spatial grid vectors $\mathbf{s}_1$ and $\mathbf{s}_2$, which is the pixle dimentions of one grid cell. In section 4.1.2 an example with figures show these vectors in both the spatial and frequency domain. The relationship between the vectors in the frequency domain and the spatial domain is as follows

$$\mathbf{s}_i = \frac{N}{|\mathbf{k}_i|^2}\mathbf{k}_i' \tag{2.12}$$

where $\mathbf{k}_i'$ is orthogonal to $\mathbf{k}_i$, the FT is NxN, see their article for an full explanation of this technique [29].

2.7 Modern Methods

Artificial Inteligence (AI) and Machine Learning (ML) are becoming more mature, and are getting a larger and larger place in research and development. This is specially true for CV, in fact a lot of the technology in todays computer vision techniques are based on some sort of neural network, such as Convolutional Neural Network (CNN)s. Theses types of networks specialises on object detection[30] and clasification[31]. There have also been attempts at replacing feature detection and matching methods with methods based on machine learning.

2.7.1 SuperGlue / SuperPoint

One of the novel approaches to feature detection and matching is SuperPoint[32] and SuperGlue[33]. These methods uses machine learning to find, describe and match features. SuperPoint are the framework for feature extraction and description, and SuperPoint is the method for feature matching using the SuperPoint descriptors.

Chapter 3

Video Material

Most of the challenges in the pre-project arose from the quality of the video material available. A majority of the data available came from cleaning operations in fish-cages, and the data was not collected in a systematic way. Much of the data contained debris from the cleaning operation, and it were not really suitable for localization. The data didn't contain calibration data for the cameras either, making it practically useless.

Another issue with the data from the pre-project were the lack of a ground-truth. This made giving accurate results tough, and the results were based on eye-observations of the real trajectory compared to the predicted path of the SLAM algorithms. Obtaining a ground truth trajectory in underwater environments is a hard task in and of itself, and most of todays datasets for underwater SLAM with a ground truth is synthetic data, like the VAROS dataset [8].

3.1 Synthesizing Data with Blender

Blender [34] is a open source 3D modelling software, primarily developed for game asset modelling, VFX and animation. But it is also being used more frequently for creating datasets for machine learning and other applications needing accurate image and video data, such as VO and SLAM. It has a flexible Python API for controlling the 3D scene and rendering, making it suitable for producing high quality datasets. The accurate ground truth acquired provides good conditions for validating the results of the VO and SLAM algorithms.

3.1.1 Fish cage modelling

In order to get a model as realistic as possible, dimensions from were obtained from Egersund Nets website[35]. The dimensions of the net structure were decided to be modelled as 70x70mm squares. This grid size is not to any industry standard, due to computer-memory limitations the grid size were decided to be modelled larger than one would find in a real cage.

The first step was to model a realistic fish cage. It could be argued that it is a little excessive to model the whole cage, and not just the net, that would to some extent be true. However, to get as close to reality more of the structure were modelled, such as the side ropes, possibly helping discern differences in the structure. The table below describes the different modelling parameters used for the cage.

Table 3.1: Dimensions used for modelling the fish cage. Based on technical data from Egersund Net [35].

Description	**Dimension**	**unit**
Circumference	150	m
Depth to bottom rope	20	m
Total depth	35	m
Number of side ropes	60	
Grid size	70x70	mm

The modelled cage in figure 3.1 is based on the parameters in table 3.1, the figure shows the wireframe of the cage.

Further a model of the environment is needed, trying to replicate a underwater environment as best as possible, with haze, and light-scattering. Figure 3.2 shows the the scene with landscape, the landscape consists of a surface, a seabed and water. In the figure, the water is not shown, because it obsures the cage, and would just show up as a box around the cage.

The shader and camera settings closely follows the Zwilgmeyers master-thesis [7], except from some of the color settings. Since that thesis primarily focuses on deep underwater exploration, it does not take into consideration the light absorption of algae and phytoplankton. The bio-optical characteristics of the scene is added to the shader, to produce a blue-green hue in the image. Figure 3.3 shows the effects of the shader in the viewport, this is not the final render however, and is just showing a preview of the shader on the scene.

Figure 3.4 shows a rendering from the dataset. The camera is placed inside the cage, capturing the grid structure of the net. Comparing the colors of the water from figure 3.3 the render have colors more consistent with actual water, this is a result of the light scattering calculations done by the rendering-engine in blender.

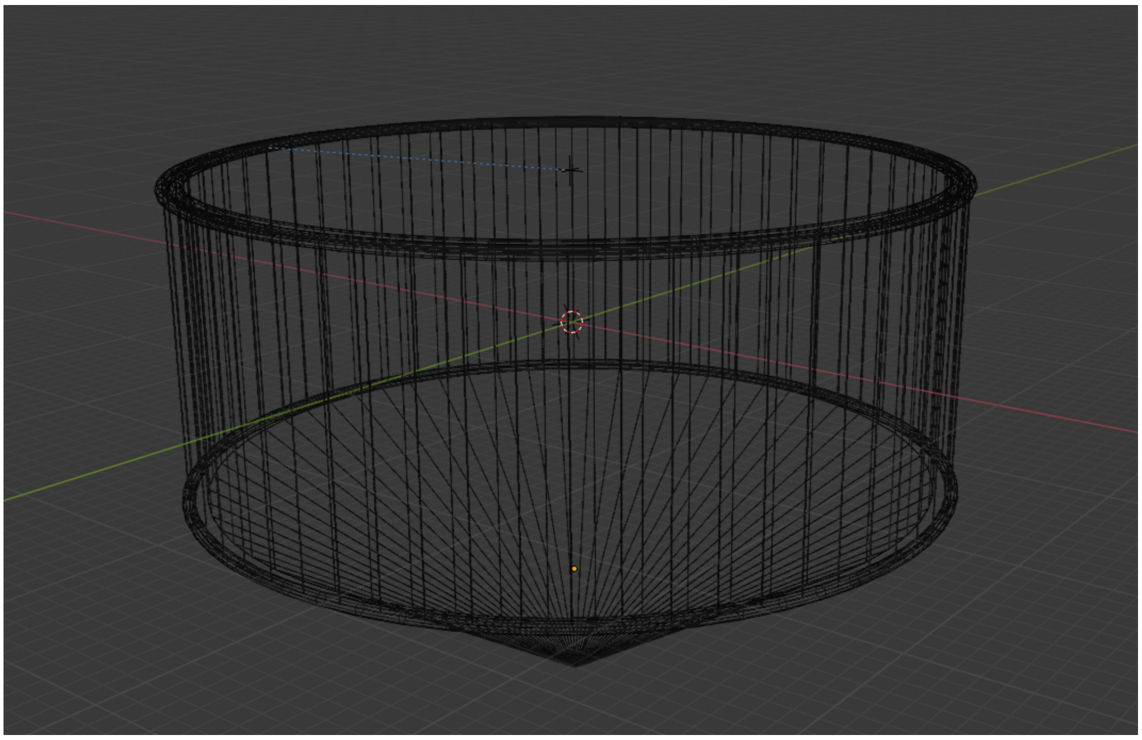

Figure 3.1: Wireframe outline of the modelled fish cage. Here only the supporting structure of the fish cage is modeled. The net is not seen in this figure.

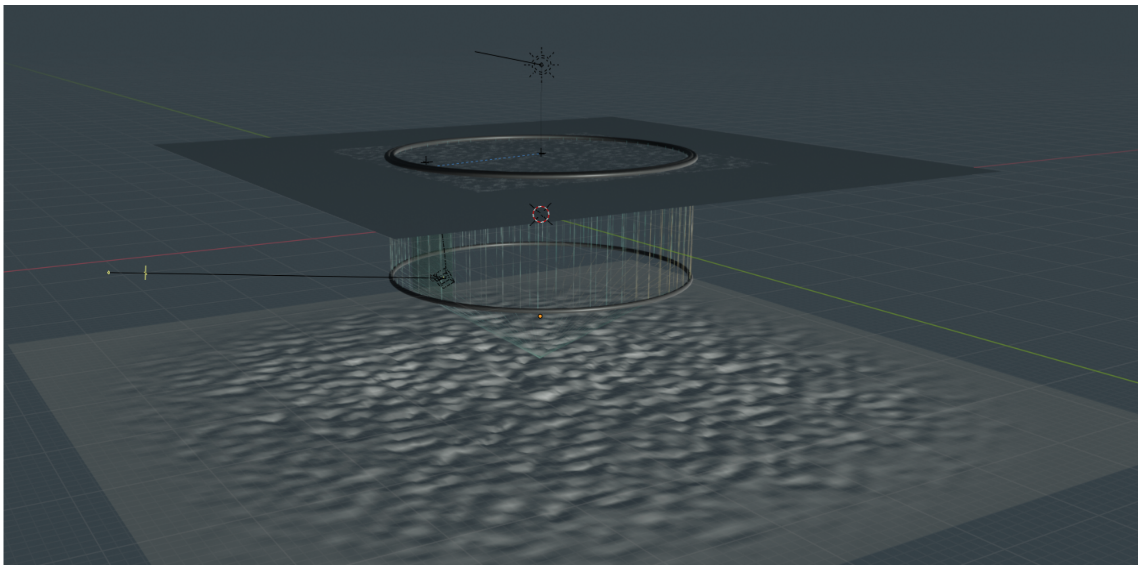

Figure 3.2: Viewport render of the fish cage and the environment. Here the "water" does not render. But a box around the cage has shaders that acts as hazy water

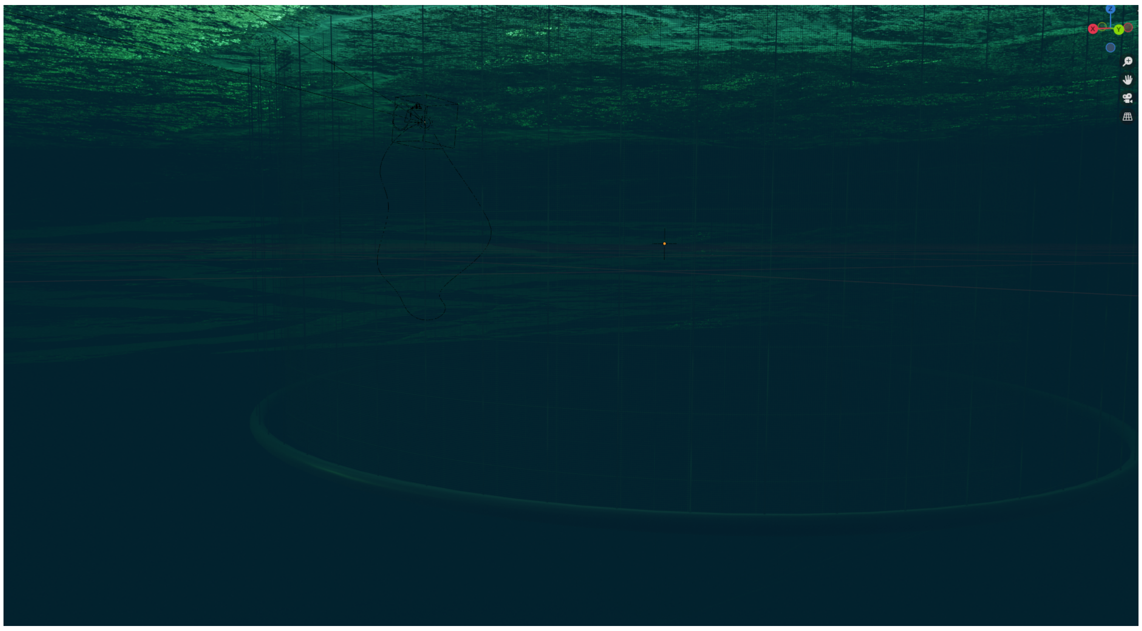

Figure 3.3: Screenshot of the underwater scene, viewport shading is turned on to show the green hue of the water.

The lighting of the scene is also modelled to get a realistic representation of a real environment. Above the surface light source representing the sun is placed, and above the camera there is placed a spotlight. Attenuation of the light coming from the sun would make the renderings almost black a few meters below the surface. ROVs also use spotlights to light up the cage when doing inspections and performing cleaning operations.

3.1.2 Dataset iterations

There have been some iterations on the datasets, but due to the amount of time needed to produce the datasets I would still describe it as a *work in progress*. The first data produced were just that, a first attempt. Figure 3.5 shows a render from this first attempt. From the figure there are very little variance in the structure, and there are not a lot of noise in in the image. These first images were rendered using blenders *eevee* rendering engine, which is a fast rendering engine, but it comes at the cost of quality.

Realising the problems with the first attempt, the rendering engine are switched to *cycles*, which is the more powerful rendering engine in blender, and the net structure are updated. This results in images as figure 3.6 shows. It is not really an improvement over the previous attempt. The problem this time around is the texturing, surrounding environment and rendering settings.

Third time's the charm. Applying other textures to the water, and changing the settings in the rendering engine, the resulting renders look lot more like what was

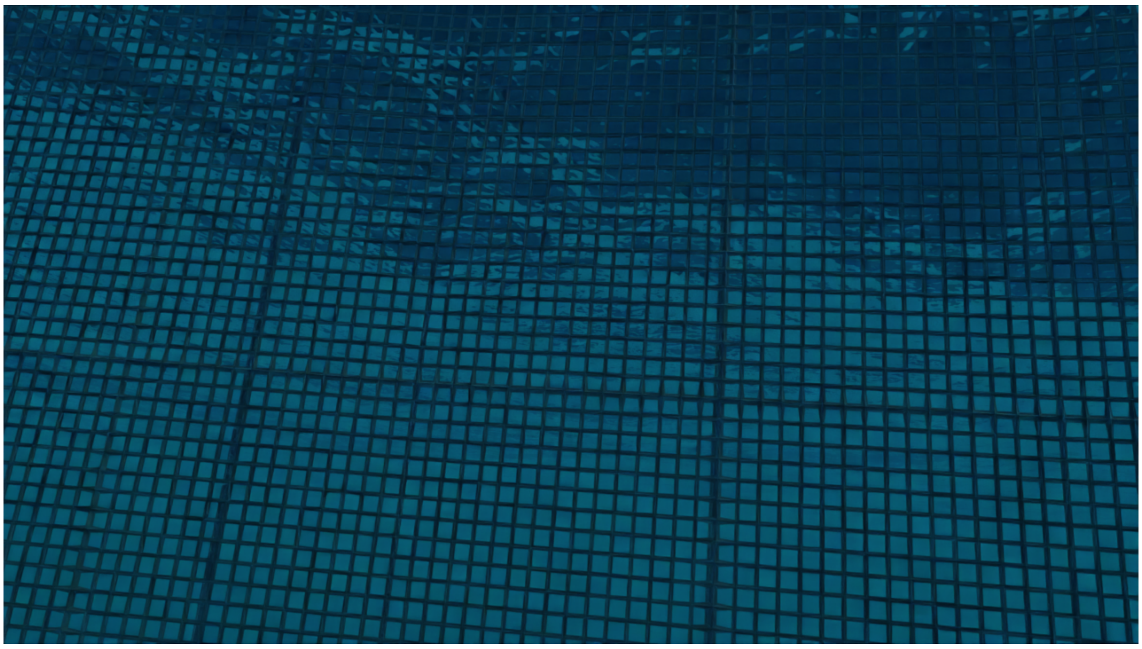

Figure 3.4: Rendering of the net structure. Thicker support ropes are shown both horizontally and vertically. Additionally there are ropes outside the net structure supporting the bottom weight ring.

intended. Figure 3.7 shows this render. This version is then used to produce the first dataset with more than just a couple of images, rendering 250 images, or just over 10 seconds of video.

However, some issues with this dataset came to light when working with it. The movement of the camera is a little too fast, and the surface of the water is static. So the images shown in figure 3.4 and figure 3.8 is the updates to the third attempt. Making the sufrace dynamic, slowing down the camera movement, and again updating some textures in order to increase the noise in the net and water.

3.1.3 Note

I am **not** a 3D-artist, and I have had to learn how to used blender in order to produce the data for this thesis. There are things I would have liked to change, but due to time and resource constraints have not been done. The data produced here, is a basis for further work. Further discussion and recommendations see chapter 6.

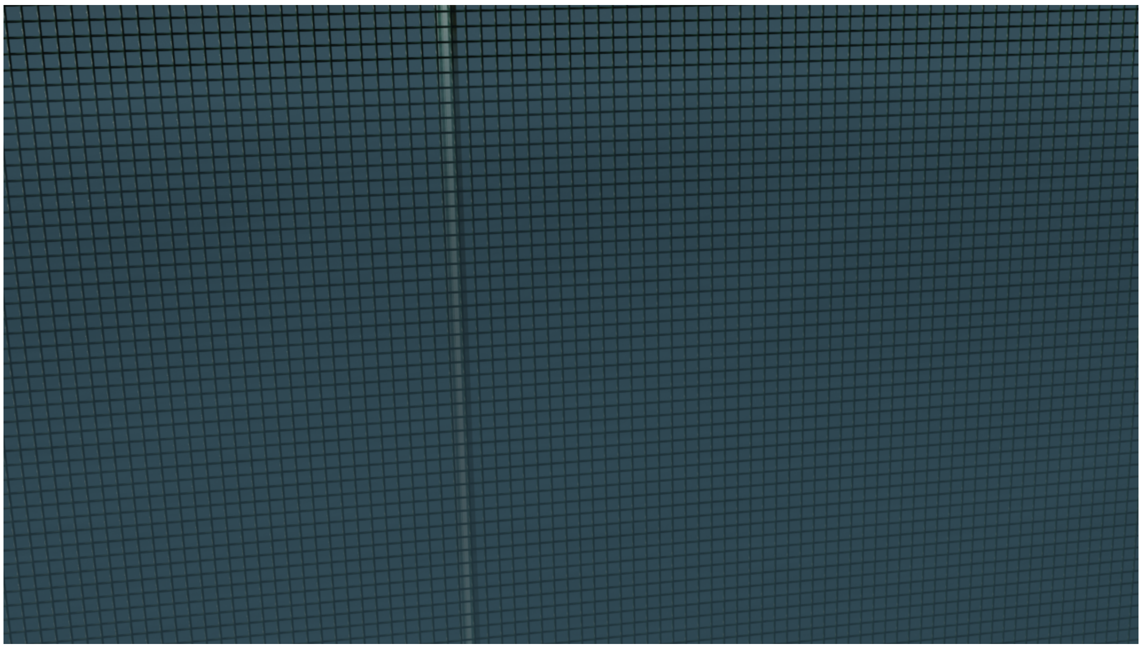

Figure 3.5: A first attempt at rendering images in the cage. It is plane, not containing enough structure, and no noise.

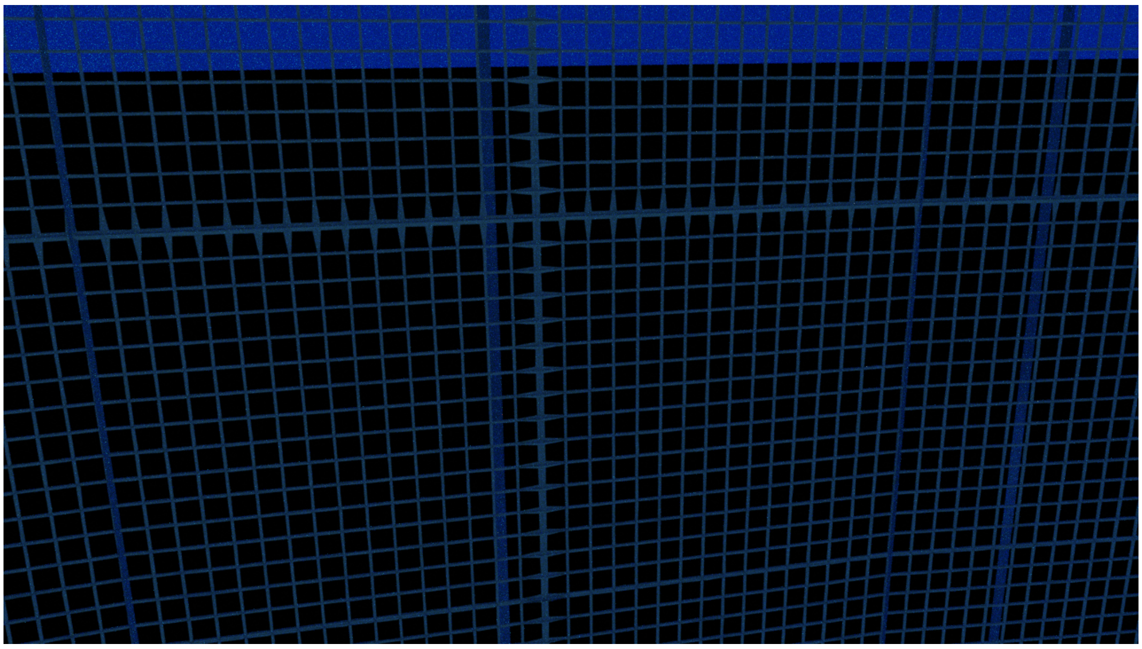

Figure 3.6: The second iteration, with updated net structure and use of cycles engine instead of eevee.

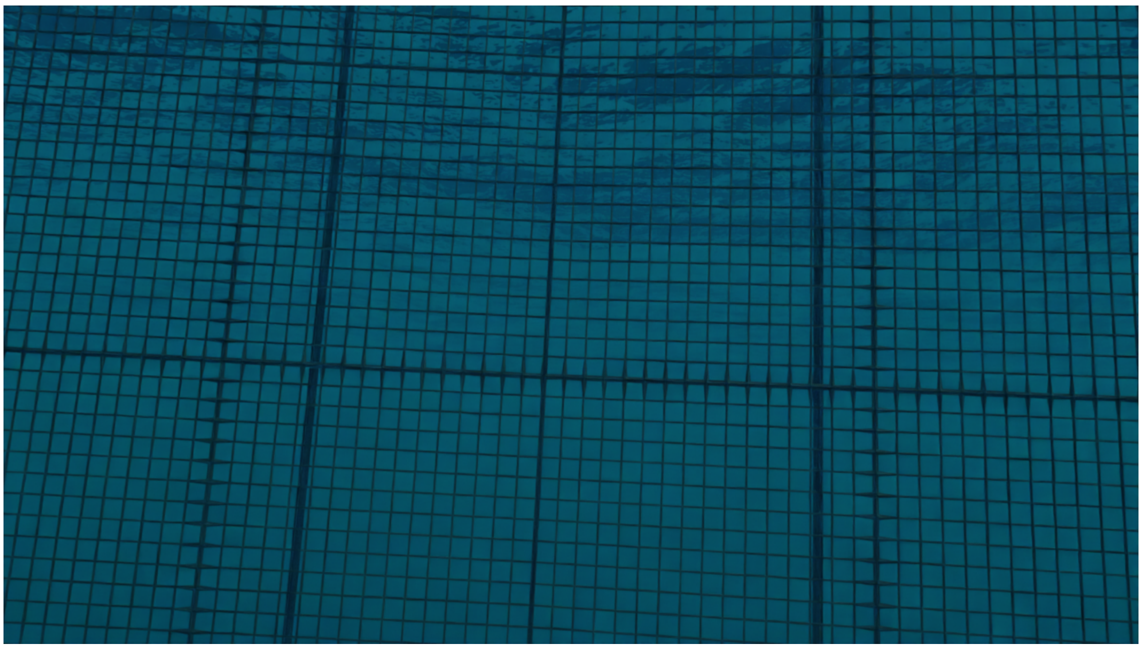

Figure 3.7: Third iteration. The water is rendered, and the looks a lot more realistic than the previous two renderings.

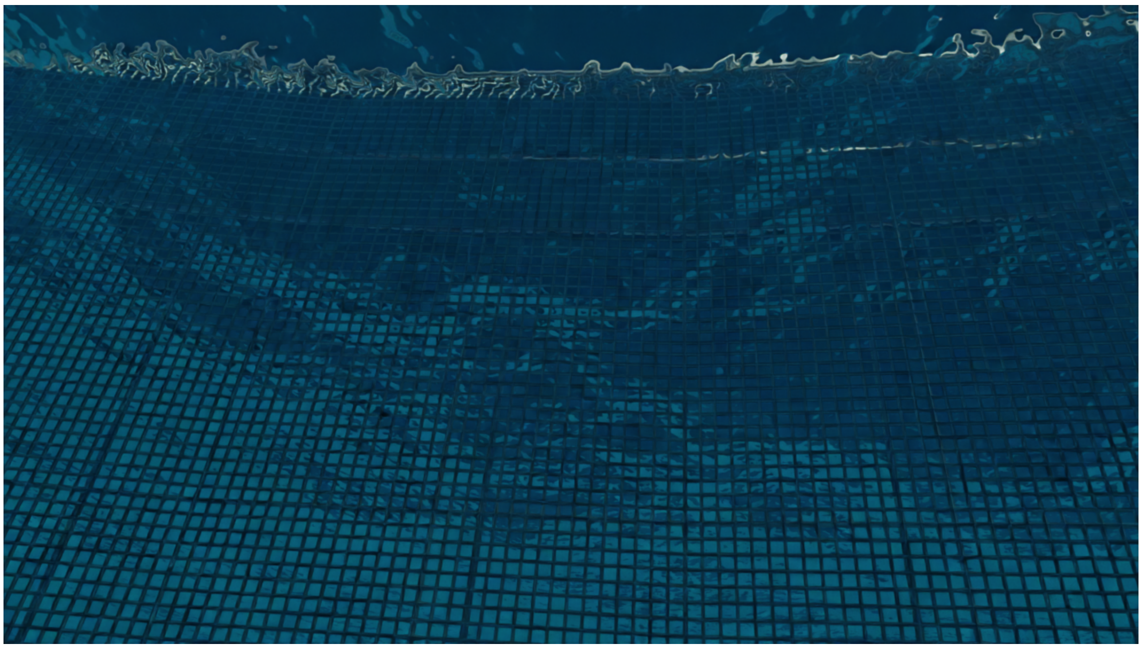

Figure 3.8: Rendering from the last updated version. Showing the camera tilted up to capture the boundary at the surface. Clearly showing the light refractions at the surface.

Chapter 4

Method

4.1 Frequency Analysis

Described in section 2.6 the two dimensional Fourier-transform yields the frequency components of an image. In this section the methods utilizing the FT from sections 2.6.1 and 2.6.2 will be explored with the synthetic data described in section 3.1.

4.1.1 Phase correlation

As described in section 2.6.1 the phase correlation method can be used to determine the rotation, scale and displacement between the same image with these transformations applied. With some care phase correlation can also be used on two images of the same area, i.e. two consecutive images in a video sequence. Since the phase correlation method assumes circular shifts, meaning the shifted region appears at the opposite edge, a windowing function can be applied to reduced the edge effects.

Figure 4.1 shows the phase correlation working for shifts on the same image, displaced by a set value, and the reporting back the same value as the shift. Whereas figure 4.2 shows two the displacement between two consecutive frames in a sequence. Here the shift is reported to be $\Delta x = 3$ and $\Delta y = -1$, which is reasonable given the observed shift. Note that there are more noise in the displacement plot than in figure 4.1.

Velocity, frame-rate and grid-size constraint

In highly repeating images, such as those this thesis is dealing with, ambiguity can arise. It is therefore important for the displacement between two consecutive frames

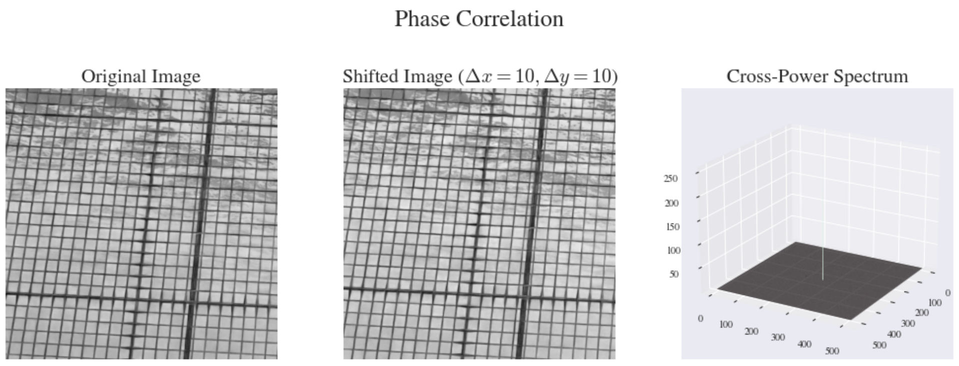

Figure 4.1: Shifted version of the same images, shifted by 10 pixels in both x and y direction. Figure on the right shows the resulting displacement plot, with a peak at $x = 266, \quad y = 266$, indicating a 10 pixel displacement in both x and y (measures from center of the frame).

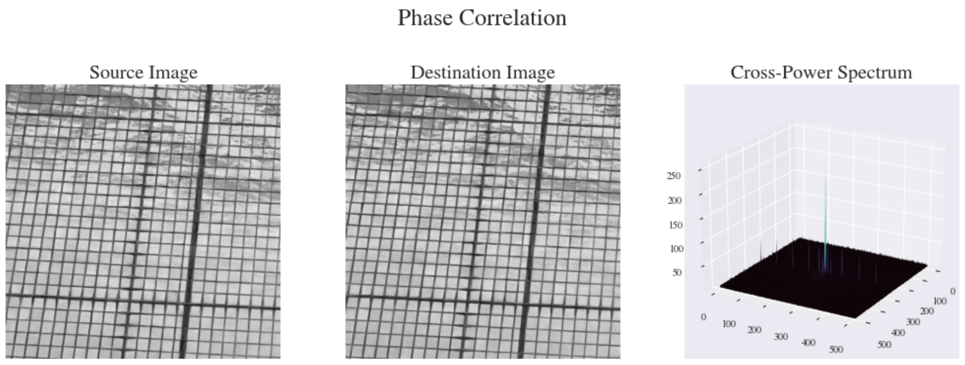

Figure 4.2: The phase correlation of two consecutive frames in a video sequence. Showing a reported shift $\Delta x = 3$, $\Delta y = -1$.

to not be too large. Formalizing this, the relationship between the velocity of the camera, frame rate, and grid size need to meet a certain criteria.

To avoid ambiguity, the pixel translation have to be less than half the pixel grid size $s_x < \frac{1}{2}k_{sx}$, $s_y < \frac{1}{2}k_{sy}$. Assuming the distance from the camera to the net is the same between each frame, we can state this formally using the pinhole camera model (see section 2.1).

$$\mathbf{s} < \frac{1}{2}\mathbf{k_s} \tag{4.1}$$

where s and k_s are defined as

$$\mathbf{s} = \frac{f}{Z}\frac{1}{T}\dot{\mathbf{x}}, \quad \mathbf{k_s} = \frac{\mathbf{g}}{Z}f \tag{4.2}$$

Table 4.1: Explanation of the different values used in equations (4.1) to (4.3).

Description	**Symbol**	**unit**
Shift	$\mathbf{s}$	[px]
Grid Size	$\mathbf{g}$	[m]
Camera Grid Size	$\mathbf{k_s}$	[px]
Camera Velocity	$\dot{\mathbf{x}}$	[m/s]
Distance to net	Z	[m]
Focal length	f	[mm]
Frames per second	T	[1/s]

resulting in the relationship between camera velocity, frame rate and grid size

$$\frac{\dot{\mathbf{x}}}{T} < \frac{1}{2}\mathbf{g} \tag{4.3}$$

As most cameras have a set frame rate which they operate at, and the grid size of the cage is a fixed value depending on the cage, this essentially limits the speed at which the Unmanned Underwater Vehicle (UUV) can operate at to get a reliable displacement estimate.

4.1.2 Net pose estimation

The method described in section 2.6.2 relies on finding the peaks in the magnitude spectrum of a FT. Figure 4.3 shows these peaks clearly in both the two- and three-dimensional representations.

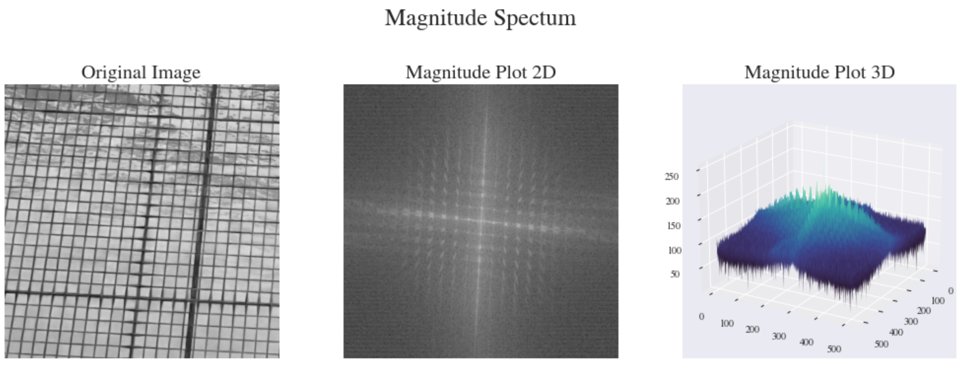

Figure 4.3: The magnitude spectrum of the image on the left, plotted in both 2D and 3D. Showing the regular peaks resulting from the regular structure in the image.

Locating the peaks in the magnitude spectrum, and finding the k_1 and k_2, as described in section 2.6.2, gives the the grid size in the original image. From the pixel grid locations, a known grid-size and a calibrated camera, a PnP-solver can be used to estimate the rotation and translation from the set origin to the origin of

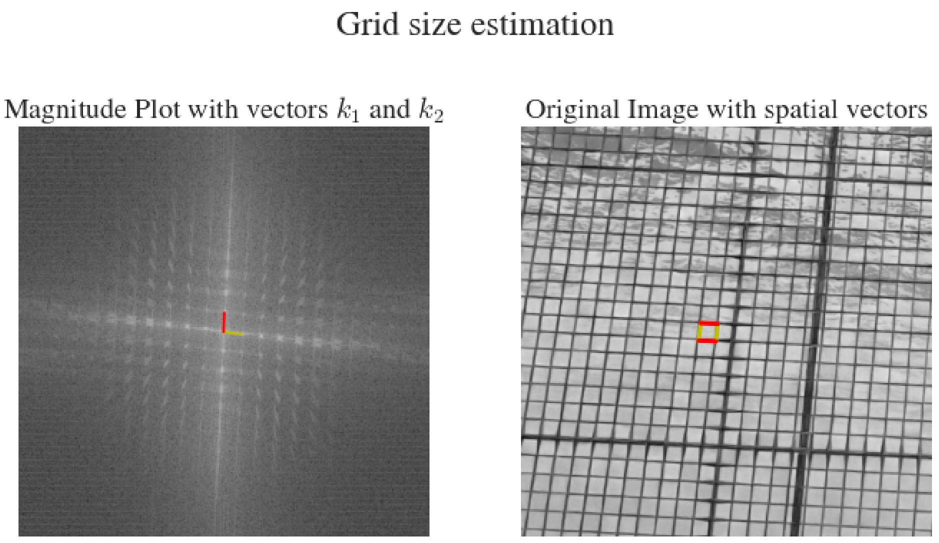
Grid size estimation
Magnitude Plot with vectors k_1 and k_2
Original Image with spatial vectors

Figure 4.4: Magnitude spectrum with vectors, and the resulting spatial grid size plotted.

the camera reference frame. The reference origin is set at the center of the grid cell. Figure 4.4 shows the estimated grid cell.

4.2 Pose Estimation

In order to acquire an estimated camera pose, there needs to be established correlation between two frames. In traditional VSLAM systems this is done with keypoints and feature-descriptors, such as ORB, SIFT or SURF. In our case this is not feasible due to the monotony of the scene, resulting in mismatched keypoints.

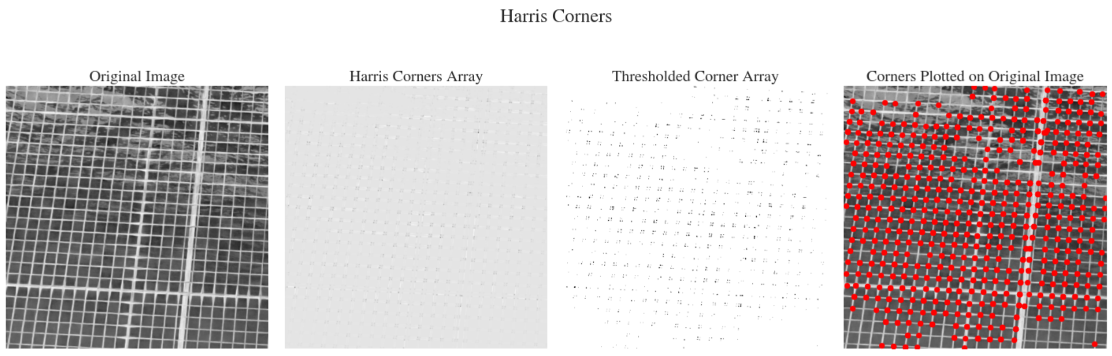

Figure 4.5: Harris corner detection, with intermediate steps. Leftmost figure shows the original image, rightmost shows the corners plotted on the original image. The figures in the center shows two intermediate steps in the corner detection.

However, we need some keypoints we can track in order to find the epipolar-geometry between two frames. A good alternative is the Harris-corner-detector[36], this method finds corners in an image. In an image with the gird structure of a cage, a lot of corners can be found. As figure 4.5 shows, three are quiet a few corners detected in a frame. And with further inspection, most of them are placed at intersections in the grid. The intermediate steps, shown in the figure shows the regions where there are detected corners. The thresholded corner array is then subject to a region algorithm, finding the center of each region, before each region is compared to the original image to find the coordinates of the corners.

Combining the methods from sections 4.1.1 and 4.1.2 in a pipeline for motion estimation, the pipeline will look something like figure 4.6.

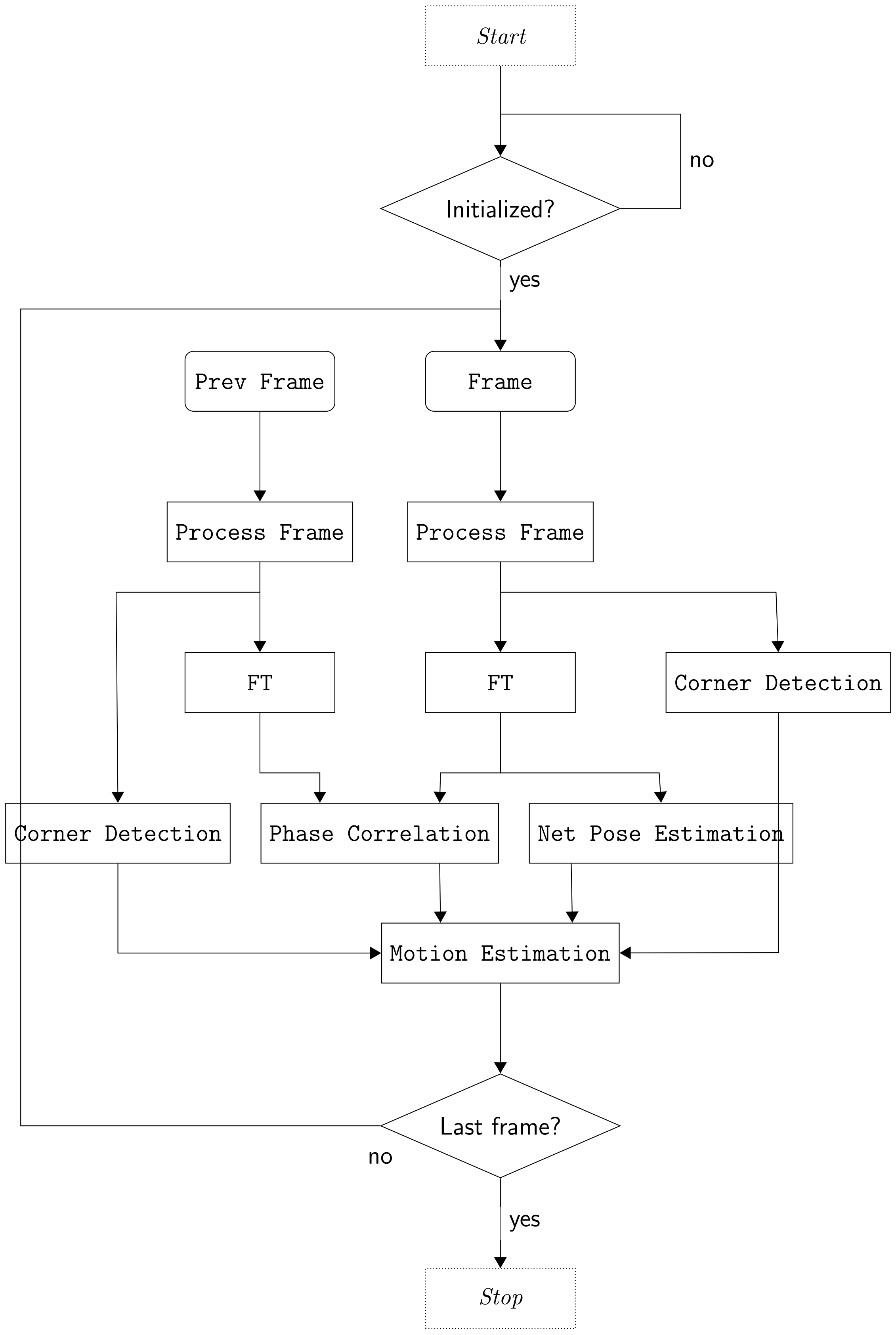

Figure 4.6: Flow diagram of the process for the front end of the motion estimation pipeline.

Chapter 5

Results

5.1 ORB feature matching

Some testing with ORB features were done at the beginning. Figure 5.1 shows the ten best matches between two consecutive frames in the dataset. Some of the matches are actual matches, but most are false positives. More frames were tested but the results were equivalent to this. With such poor data association, any form of motion estimation is next to impossible.

5.2 AI Based Feature Descriptors

The feature descriptors and feature matching mentioned in section 2.7.1, SuperPoint[32] and SuperGlue[33], were tested on the dataset. The test used pretrained networks from MagicLeap[37], but it were concluded without much tweaking of the network, that the matches were not good enough.
Figure 5.2 shows that for the start of the dataset there are quiet a few matches, but

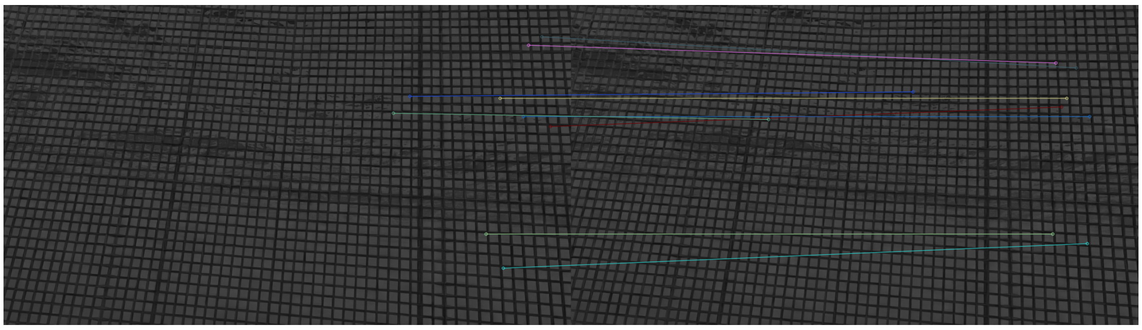
Figure 5.1: The 10 best matches between two consecutive frames in the dataset.

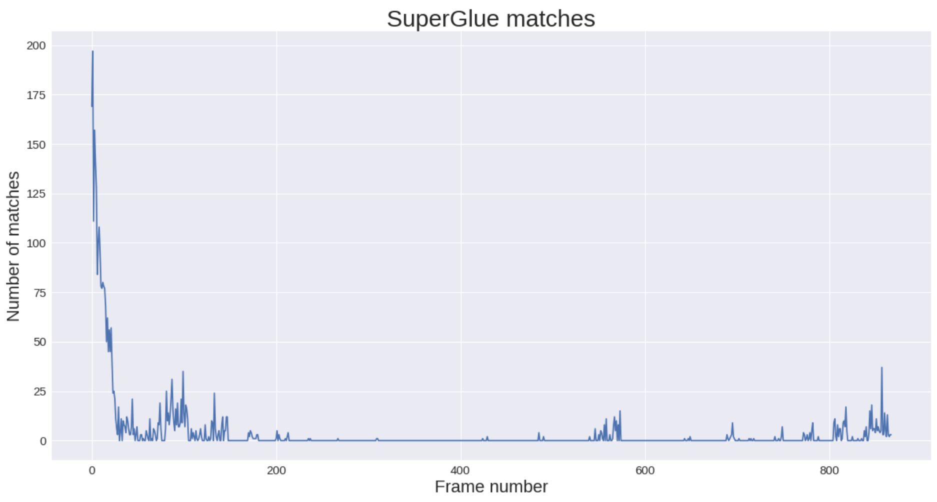

Figure 5.2: Number of matches for each consecutive frame in the dataset.

as as the sequence continues, the number of matches drops to zero for a lot of the dataset. This is not enough structure to get any kind of position estimate. This could be down to some fine tuning in the matching network, but no more time were spent pursuing this.

5.3 Frequency Analysis

5.3.1 First dataset

For the first dataset the same methods as in the section below was tried. The results were worse for the whole of the dataset. Figure 5.3 shows the resulting error plot. It shows that the compounding errors for X and Y are increasing rapid at the start. The quality of the data have some stake in this, see chapter 6.

5.3.2 Final dataset

A simple motion estimation pipeline, similar to the described in figure 4.6, was deployed on the final dataset. The difference between the pipeline in described in figure 4.6 and the one used here, is that we don't take advantage of the corner detection. Figure 5.4 shows a 3D plot of the final estimate, and figure 5.6 shows the XY-plot of the same values. Here the coordinates are in the camera coordinate system. The error for each frame is shown in figure 5.5. Figures 5.7 to 5.9 shows each axis and their estimates for each frame.

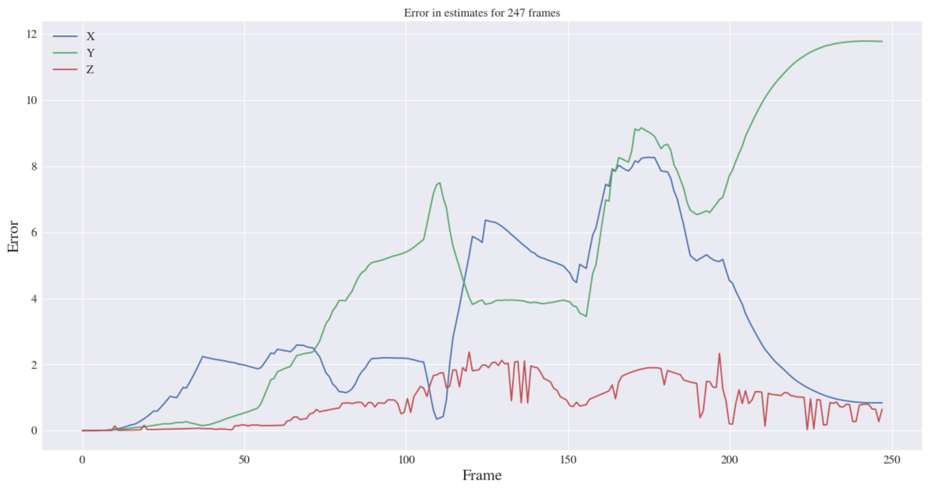

Figure 5.3: Error plot for the first dataset.

As the figures show, there errors tend to get very large. However for the x and y axes these are compounding errors. So the error from the previous steps are added at each frame.

This dataset is almost 40 seconds long at 24 frames per second. The speed of the camera changes from slow goinng slow to faster. We take a look at a smaller sample size of at the start of the sequence. Inspecting the 100 first frames in the sequence, we get the 3D plot in figure 5.10, showing the ground truth trajectory and the estimate. Plotting the same plots as for the full dataset, the error plot is shown in figure 5.11. There are some spikes in the estimate of Z. This is not a compounding error, so the estimate corrects itself after some frames. These spikes are due to wrong grid size estimation. X and Y have a relative small error, figure 5.12 shows the same plot without the Z value stetching the axis. The maximum error for the x axis in this sequence was reported to be 20%, and for the y axis it was 81%.

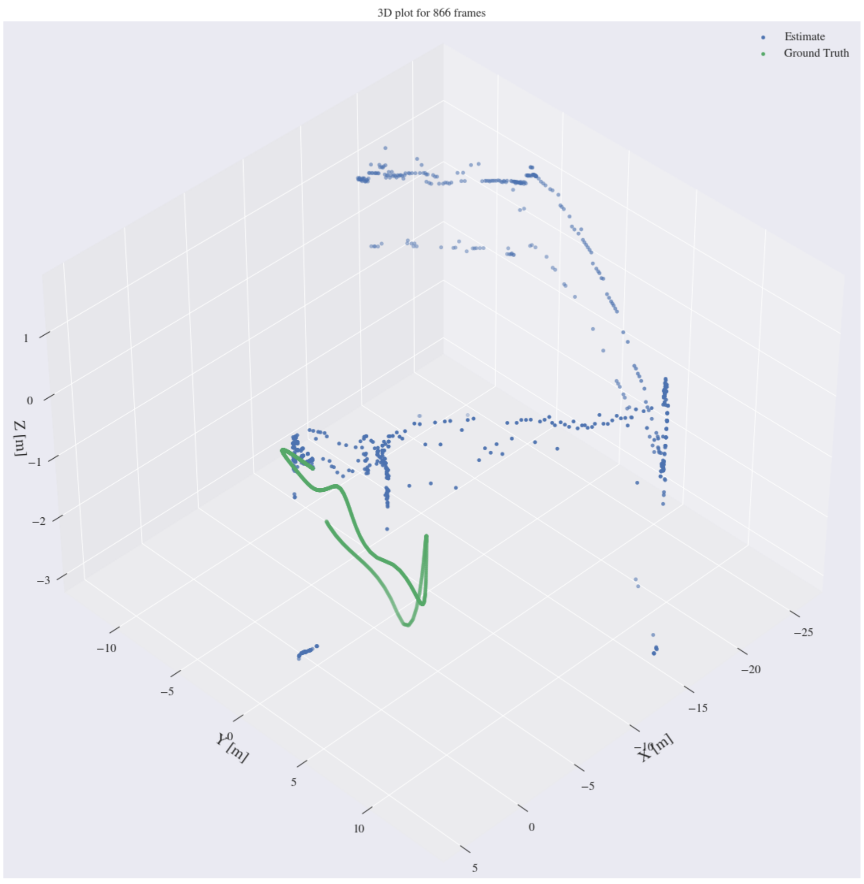

Figure 5.4: 3D plot of all the positions estimated by the simple motion estimation method.

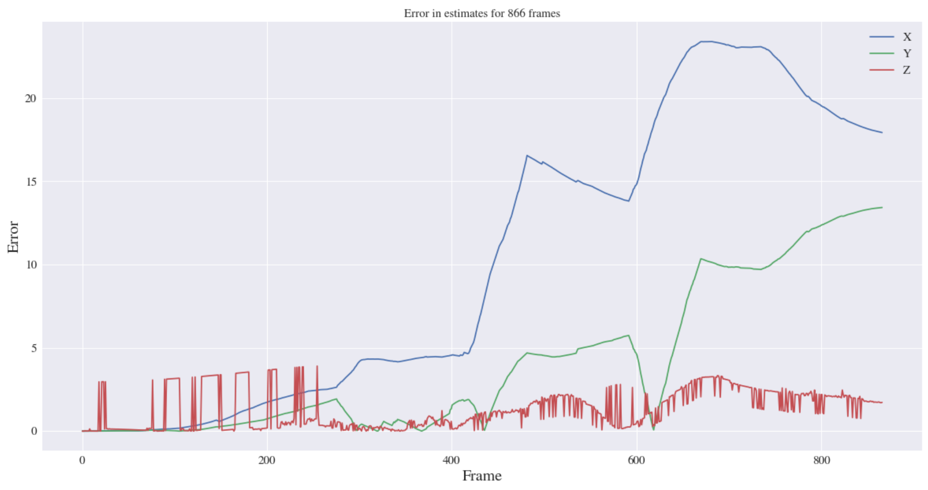

Figure 5.5: Error plot. The error is the deviation of the estimate to the ground truth.

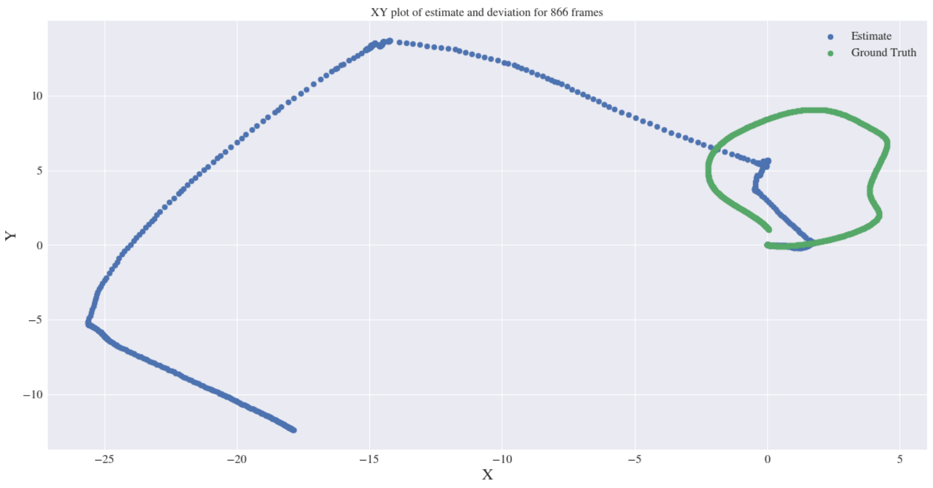

Figure 5.6: XY plot. A plot of the estimated X and Y values. The same values as in the 3D plot, but seen from the XY-axis

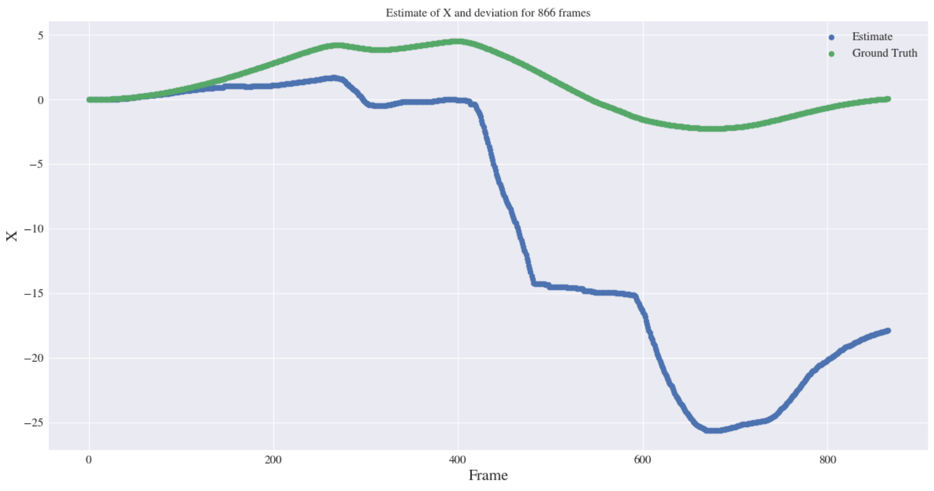

Figure 5.7: Plot of the estimated X values and the ground truth.

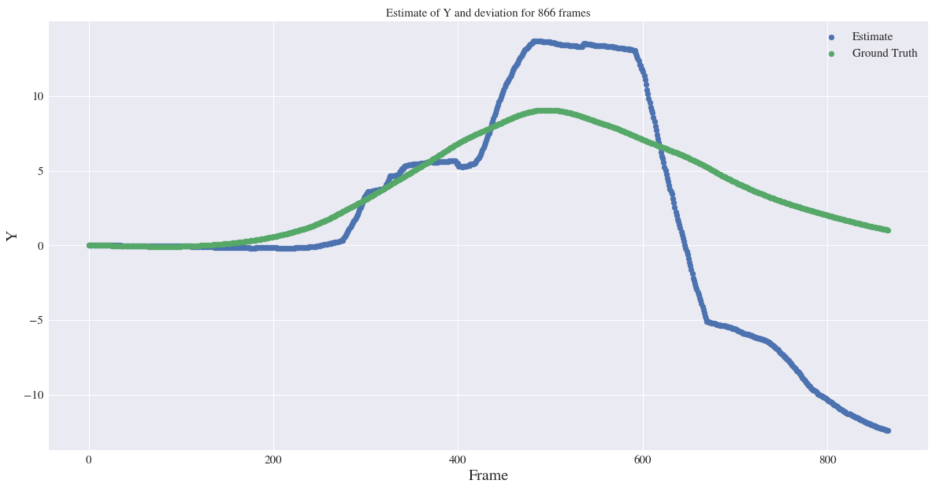

Figure 5.8: Plot of the estimated Y values and the ground truth.

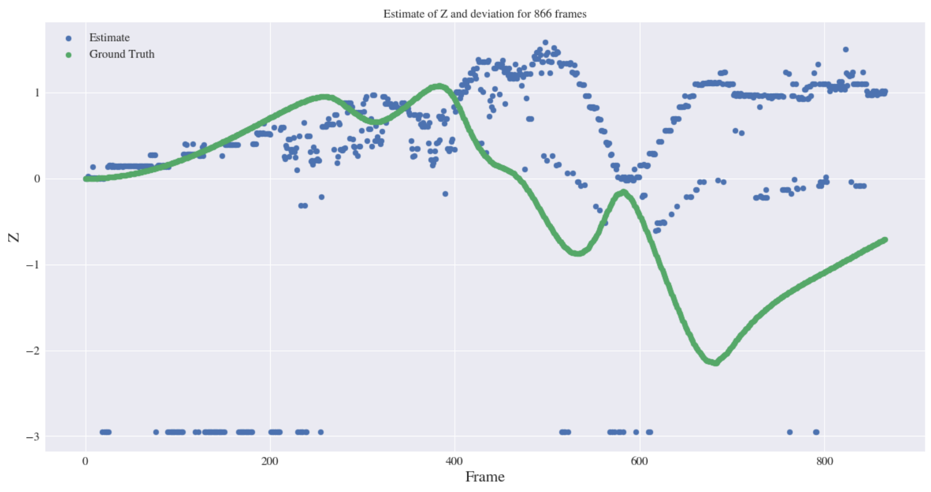

Figure 5.9: Plot of the estimated Z values and the ground truth.

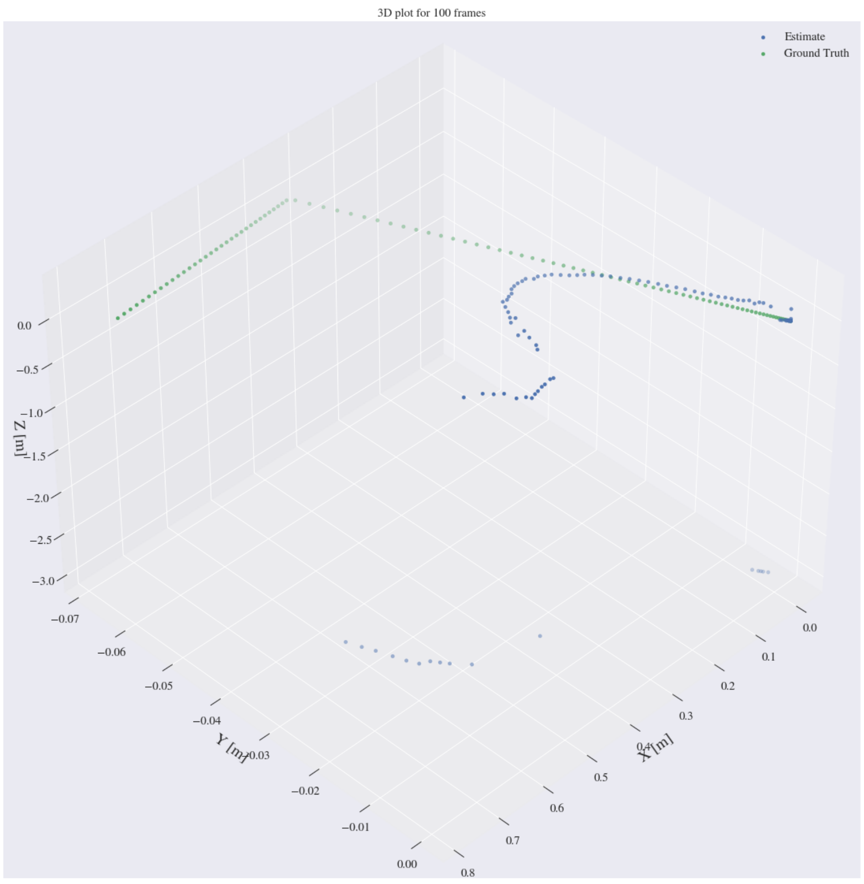

Figure 5.10: 3D plot of all the positions estimated by the simple motion estimation method.

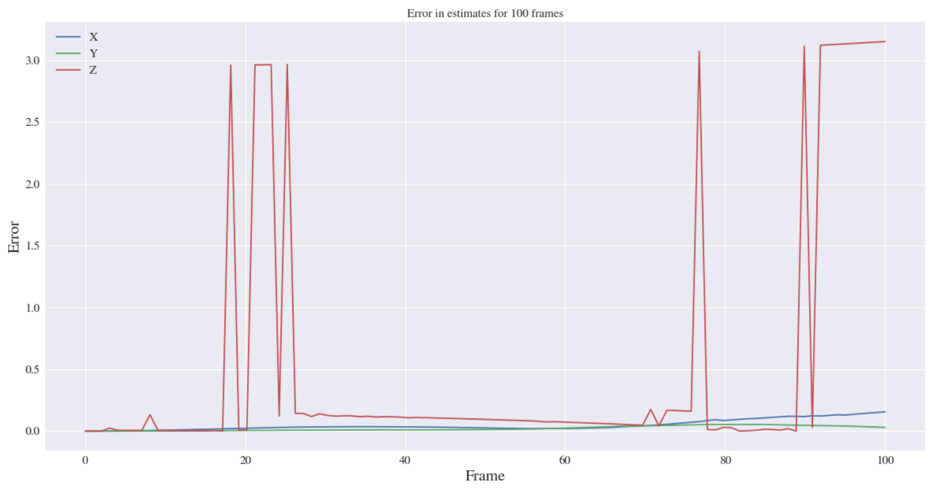

Figure 5.11: Error plot. The error is the deviation of the estimate to the ground truth.

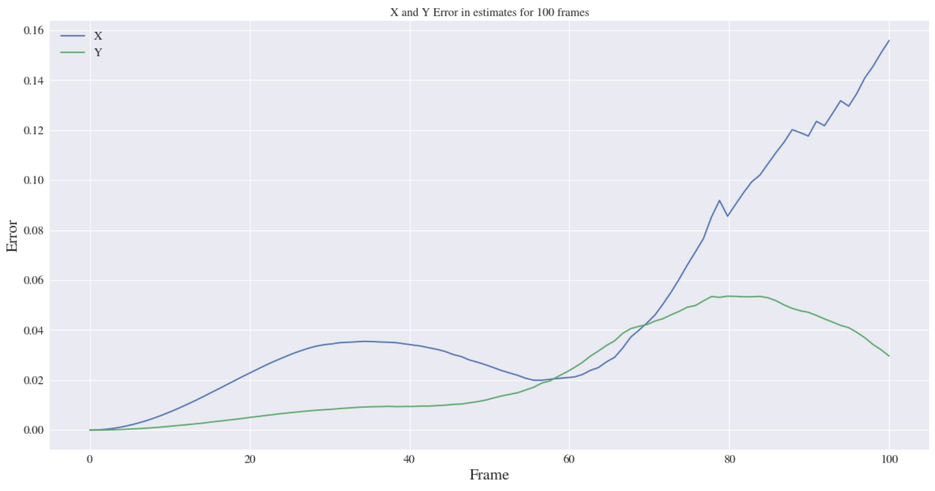

Figure 5.12: Error plot. The error is the deviation of the estimate to the ground truth.

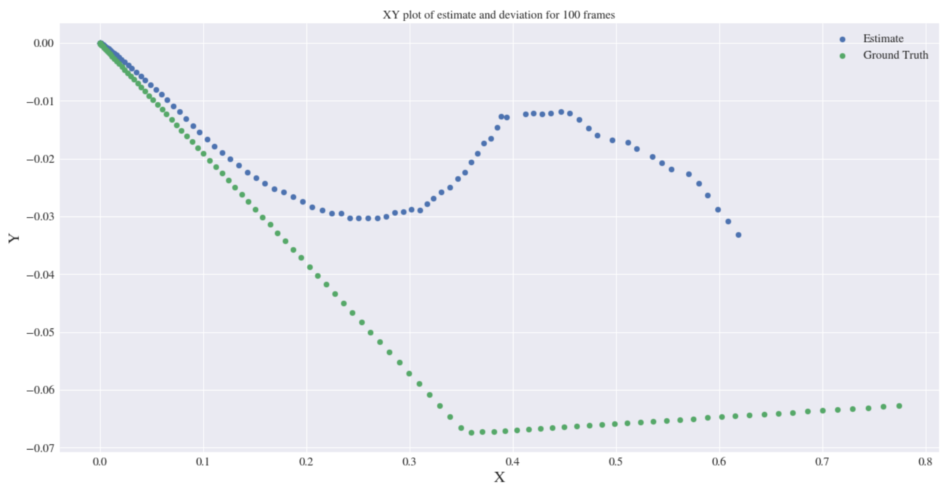

Figure 5.13: XY plot. A plot of the estimated X and Y values. The same values as in the 3D plot, but seen from the XY-axis

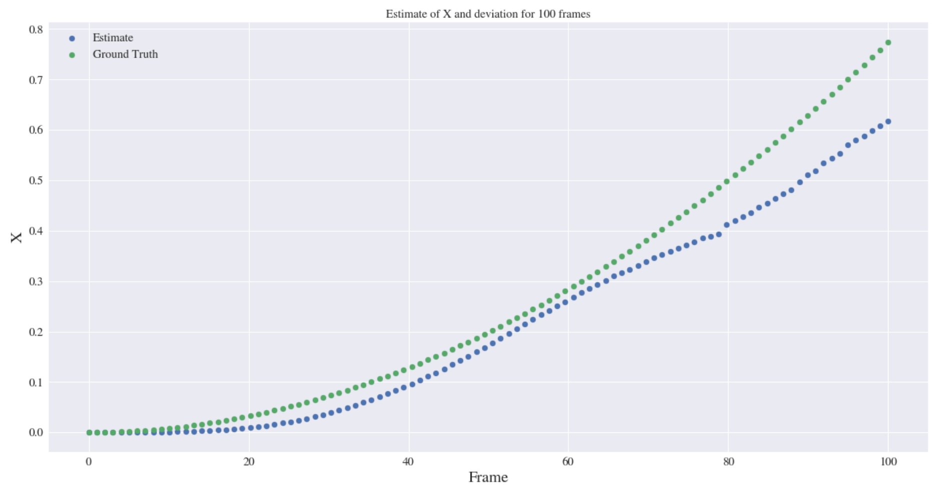

Figure 5.14: Plot of the estimated X values and the ground truth.

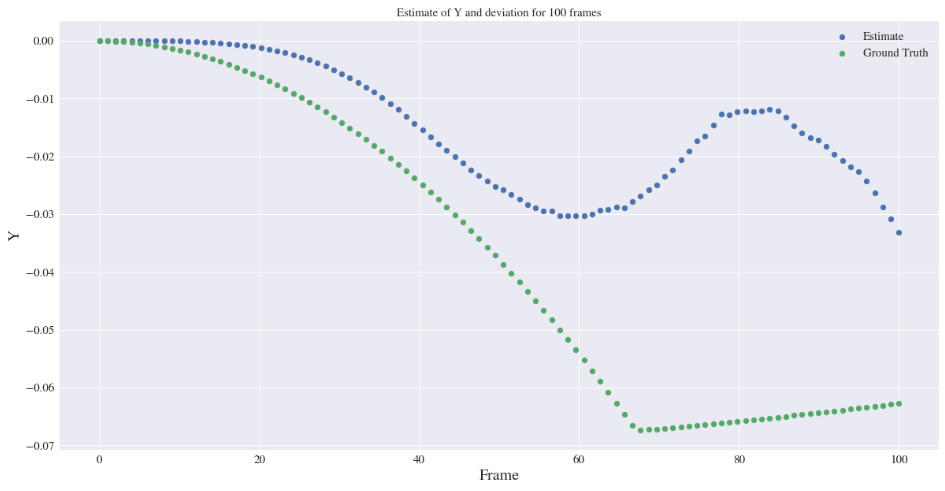

Figure 5.15: Plot of the estimated Y values and the ground truth.

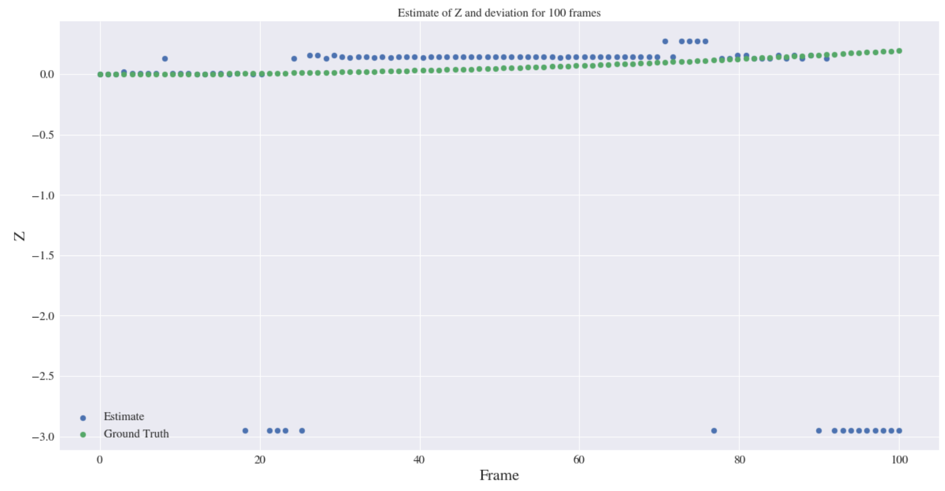

Figure 5.16: Plot of the estimated Z values and the ground truth.

Chapter 6

Discussion

It has been an experience writing this thesis. First having to produce synthetic data, and then using said data. Experimenting with more traditional methods, to realise these are not a viable solution. To then venturing into fourier transforms and frequency analysis.

6.1 Data

This thesis started with the problem of not having sufficient data available. Thus we set out to generate our own. This was an ambitous goal, but in the end the results turned out better than expected. However the fact that the data had to be produced had an impact on the time available for working on the method for solving the problem we initially set out to solve.

As breifly mentioned in section 3.1.3 there are some things that still would be good to improve with the current dataset. First and foremost, general improvements to the scene. The surface of the water is dynamic in the last version, but the light changes are too abrupt. Slowing down the simulation of the waves on the surface would improve the quality of the data. Texturing is also something that could be improved. Learning to use blender in this thesis showed me just how complex texturing can be. In the data produced the textrues are good enough for a rough draft, but improvements can definently be made.

Another thing to be improved is the camera motion. The motion of the camera is just a following a path layed out in blender. It is not currently programmable, the curve have to be manually changed in the blender model. And it does not rely upon a motion model. In order to produce better; more realistic motion, and to be able to simulate IMU data this is perhaps the first improvement that should be done.

6.2 Preprocessing

First steps in any image analysis is preprocessing. In this thesis a very basic method of image preprocessing is used. Binarization for highlighting the structure of the net, and windowing in order to avoid edge effects in the phase correlation step. It is simple, and effective to a degree. Still there are things to be desired, specially for the grid highlighting in form of the binarization. There are still artifacts from the background, which is at least not helping the final results.

More advanced preprocessing and methods existst, and should be considered fot use here. Using methods for removing the backgound so only the grid remains would also be benefitial, a method in Arild Madshavens master thesis[38] does this.

6.3 State of the Art Methods

A breif excursion in the state of the art methods were considered. This was, however, not fruitful. Feature based methods such as ORB, and SuperGlue/SuperPoint were not able to establish good correlations between features from different frames. It was however not surprising, the results from the pre-project had shown that these methods was not reliable for the monotone environment of a fish cage.

6.4 Fourier Transforms

Switching focus to the method of using fourier transforms for motion estimation, yielded promesing results in initial testing. If the translation of the grid structure was not to large, a good estimate of the image translation in pixels could be obtained. This suggested that using phase correlation for motion estimation could be a viable option. Together with the method for pose estimation of the net in the fish cage, yielding a distance from the net to the camera, a rudimentary method for motion estimation was made.

For the tests in the second dataset with a simple motion estimation model it proved that this motion estimation model were too simple. Which were expected. For the first 100 frames of the sequence it reported some deviation drom the acctual trajectory, but it was much better for this segment than for the rest of the dataset. The subsequence reported a maximum error of 20% in the X direction and 81% in the y axis. These results are not great, but it is a start.

From the outset, trying to work with feature based methods, these results are very promising. Keep in mind that the method for estimating the motion of the camera is very simple. This method could be greatly improved to increase the accuracy of the estimate. Improving the phase correlation to increase the accuracy of the translational movement of the camera is one improvement that could be made. Making a model for the multi-motion of the frame would also be advantageous.

One method that wasn't fully tested or produced results for was including a corner detector, matching these with based on the translation fromm one frame to the next. This would then produce a result more in line with normal feature based methods.

6.5 Additional Sensor Information

One of the goals at the start of the thesis were to include sensor information from an IMU. However, this did not happen. Inclusion of such sensor data and a motion model for the camera would further increase the accuracy of the estimates. Stereo imaging would also be a good improvement to the dataset, and could help with further improving the results of the motion estimation.

6.6 Conclusion

Two synthetic dataset for underwater inspection of aquacultural fish cages have been produced, these datasets include high quality rendered images from a camera in motion in front of the net structure in the cage. Ground truth information for the motion of the camera and intrinsic camera parameters are also a part of these datasets. Producing these datasets was key for taking a step in the right direction of a solution to egomotion estimation in aquacultural fish cages.

Continuing with the work from the pre-project, a basic method for camera motion estimation have been developed. This method is extremely simple, and is a first primitive approach at the problem using fourier transforms. The results reflect the simple state of this model. However, it shows some promise for further work on this problem. While traditional feature based models using keypoints and feature descriptors fail to establish any good matches between frames, the approach of using frequency analysis at least show there are possibilities for a solution to this problem.

6.7 Further Work

The work in this thesis should lay a foundation to be built upon for further exploration into egomotion estimation in aquacultural fish cages. As this chapter already have talked about, there are lots of improvements to be done to the different components of the method. Extending the method presented in this thesis with techniques such as better preprocessing, motion models and just general improvements in to the method, should see considarably better results.

Another point for further work is to improve the dataset, preferably with the inclusion of IMU data and stereo imaging. Further general improvements to the model would also be welcome. There are still room for making the renderings more photorealistic.

Bibliography

[1] Lars Olav Libjå. Localization of defects in aquacultural fish cages, January 2021.

[2] SSB. De største næringene, 2021. URL https://www.ssb.no/nasjonalregnskap-og-konjunkturer/faktaside/norsk-naeringsliv.

[3] Trond A. Steinset. Frå attåtnæring til milliardindustri. *Samfunnsspeilet*, (1/2017), 2017. URL https://www.ssb.no/jord-skog-jakt-og-fiskeri/artikler-og-publikasjoner/fra-attatnaering-til-milliardindustri.

[4] Miljødirektoratet. Miljøstatus laks, 2021. URL https://miljostatus.miljodirektoratet.no/tema/ferskvann/laks/.

[5] Mariko Igawa, Yoshikata Atsumi, Kazumi Takahashi, Shinichi Shiotsuka, Hideto Hirasawa, Ryusei Yamamoto, Atsushi Maki, Yuichi Yamashita, and Hideaki Koizumi. Activation of visual cortex in rem sleep measured by 24-channel nirs imaging. *Psychiatry and clinical neurosciences*, 55(3):187–188, 2001. ISSN 1323-1316.

[6] Dzmitry Mazouka and Viktor Krasnoproshin. Developmental milestones of graphics technologies. 2021.

[7] Peder Georg Olofsson Zwilgmeyer. Creating a synthetic underwater dataset for egomotion estimation and 3d reconstruction. Master's thesis, Norwegian University of Science and Technology, 2021. URL https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2786135.

[8] Peder Georg Olofsson Zwilgmeyer, Mauhing Yip, Andreas Langeland Teigen, Rudolf Mester, and Annette Stahl. Varos synthetic underwater data set, October 2021. URL https://doi.org/10.5281/zenodo.5567209.

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

[10] WenLong Zhao, Tao He, Abdou Yahouza M. Sani, and TingTing Yao. Review of slam techniques for autonomous underwater vehicles. In *Proceedings of the 2019*

International Conference on Robotics, Intelligent Control and Artificial Intelligence, RICAI 2019, pages 384–389, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450372985. doi: 10.1145/3366194.3366262. URL https://doi.org/10.1145/3366194.3366262.

[11] Randall C Smith. Development system for flexible assembly system. 1986.

[12] Randall Smith, Matthew Self, and Peter Cheeseman. Estimating uncertain spatial relationships in robotics. 1986. doi: 10.48550/ARXIV.1304.3111. URL https://arxiv.org/abs/1304.3111.

[13] Wikipedia. Odometry. URL https://en.wikipedia.org/wiki/Odometry.

[14] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220, 2017.

[15] Kevin Köser and Udo Frese. Challenges in underwater visual navigation and slam. In *AI Technology for Underwater Robots*, pages 125–135. Springer, 2020.

[16] Xinzheng Zhang, Ahmad B Rad, and Yiu-Kwong Wong. Sensor fusion of monocular cameras and laser rangefinders for line-based simultaneous localization and mapping (slam) tasks in autonomous mobile robots. *Sensors*, 12 (1):429–452, 2012.

[17] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5): 1147–1163, Oct 2015. ISSN 1941-0468. doi: 10.1109/tro.2015.2463671. URL http://dx.doi.org/10.1109/TRO.2015.2463671.

[18] Raul Mur-Artal and Juan D. Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33: 1255–1262, 2017.

[19] Carlos Campos, Richard Elvira, Juan J. G'omez Rodr'iguez, José M. M. Montiel, and Juan D. Tardós. Orb-slam3: An accurate open-source library for visual, visual-inertial, and multimap slam. *IEEE Transactions on Robotics*, 37: 1874–1890, 2021.

[20] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. pages 2564–2571, 11 2011. doi: 10.1109/ICCV.2011.6126544.

[21] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[23] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

[24] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[25] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[26] Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[27] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965. ISSN 00255718, 10886842. URL http://www.jstor.org/stable/2003354.

[28] B.S. Reddy and B.N. Chatterji. An fft-based technique for translation, rotation, and scale-invariant image registration. *IEEE Transactions on Image Processing*, 5(8):1266–1271, 1996. doi: 10.1109/83.506761.

[29] Christian Schellewald, Annette Stahl, and Eleni Kelasidi. Vision-based pose estimation for autonomous operations in aquacultural fish farms. *IFAC-PapersOnLine*, 54(16):438–443, 2021.

[30] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[32] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. *CoRR*, abs/1712.07629, 2017. URL http://arxiv.org/abs/1712.07629.

[33] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. URL https://arxiv.org/abs/1911.11763.

[34] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org.

[35] Egersund Net. Straight wall circular net, 2022. URL https://www.egersundnet.no/products/fish-farming-nets/straight-wall-circular-net.

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

[37] MagicLeap. Supergluepretrainednetwork. online, 2022. URL https://github.com/magicleap/SuperGluePretrainedNetwork.

[38] Arild Madshaven. A complete framework for robust fish cage hole detection in challenging environments. Master's thesis, Norwegian University of Science and Technology, 2021. URL `https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2781060?show=full&locale-attribute=no`.