NTNU
Det skapende universitet

Maskinsyn-basert referansesystem for automatisk traversering av oppdrettsnot

Morten Engelhardt Olsen

Master i teknisk kybernetikk
Innlevert: juli 2013
Hovedveileder: Jo Arve Alfredsen, ITK

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

NTNU – Trondheim
Norwegian University of
Science and Technology

Camera-assisted ROV navigation in sea cages

Morten Engelhardt Olsen

Submission date: July 2013
Responsible professor: Jo Arve Alfredsen, ITK
Supervisor: Jo Arve Alfredsen, ITK
Per Rundtop, SINTEF Fiskeri og Havbruk

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU

Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for teknisk kybernetikk

MASTEROPPGAVE

Kandidatens navn:	Morten Engelhardt Olsen
Fag:	Teknisk kybernetikk
Oppgavens tittel:	Kamera-assistert ROV-navigasjon i oppdrettsmerder
Engelsk tittel:	Camera-assisted ROV navigation in sea cages

Oppgavens tekst

Industrialiseringen av moderne havbruksproduksjon har aktualisert fjernstyrte undervannsfarkoster (ROV) som plattform for ulike driftsoperasjoner i havbruksanlegg. Både not- og fortøyningsinpeksjon og vaskeoperasjoner blir nå rutinemessig utført ved bruk av ROV. Oppdrettsmerden representerer imidlertid en fleksibel og sårbar struktur, og navigasjon av farkosten i dette miljøet er en utfordring. Formålet med oppgaven er å undersøke i hvilken grad bildestrømmen fra ROVens kamera (eventuelt eget kamera) i kombinasjon med avanserte algoritmer for bildebehandling og -analyse, strukturert belysning, og andre hjelpemidler, kan benyttes til å bestemme ROVens posisjon, orientering og bevegelse i forhold til merden, dvs. som en form for kamerabasert odometri. Metodikken skal utvikles med henblikk på at målingene skal brukes til automatisk regulering av ROVens avstand, vinkel og hastighet i forhold til notveggen, samt visualisering av farkostens posisjon "på" merden. Oppgaven omfatter følgende punkter:

- Gjennomgang av problemstillingens bakgrunn, identifisering av forskjellige bruksscenarier og utarbeidelse av kravspesifikasjon for målesystemet
- Kartlegging og diskusjon av faktorer som har innvirkning på bildekvaliteten i en typisk oppdrettsmerd, slik som kamerateknologi, belysning, dybdeavhengige lys- og kontrastforhold, fisk, turbiditet, og bølger.
- Gjøre en vurdering av kamerateknologi og belysningsutstyr på aktuell ROV (Argus) i forhold til systemets behov
- Design og utvikling av algoritmer for sanntids bildeprosessering og -analyse av notpaneler, inkludert eventuelle hjelpesystemer, for ekstraksjon av informasjon om ROVens posisjon, orientering og bevegelse. Strategier for å utnytte informasjonen til å assistere ROV-pilotens navigasjon i merden skal foreslås/demonstreres.
- Planlegging og gjennomføring av tester for å studere algoritmen(e)s nøyaktighet, presisjon og robusthet, både under ideelle og realistiske betingelser
- Diskusjon av resultater og kartlegging av systemets muligheter og begrensninger

Oppgaven gitt:	14. februar 2013
Besvarelsen leveres innen:	12. juli 2013
Utført ved:	Institutt for teknisk kybernetikk, NTNU
Veileder:	Jo Arve Alfredsen, ITK, NTNU
	Per Rundtop, SINTEF Fiskeri og Havbruk

Trondheim, 14. februar 2013

Jo Arve Alfredsen

Faglærer

Abstract

This thesis is a preliminary study of the feasibility of using modern computer vision algorithms for navigation, positioning and orienting in specific and known underwater environments. The use of cameras as a source of data has long been thought of as to difficult and error prone. However, due to massive progress in the quality of digital cameras and increased computing power in the filed, this may no longer be the case.

During this thesis, several different classes of algorithms for detecting motion in an image stream are presented. Different strengths and weaknesses are discussed. A fair bit of the work done also includes establishing a new hardware platform to acquire high definition video. The hardware platform was designed to mimic the specifications used on Remote Operated Vehicle (ROV)s today.

Difficulties associated with object tracking and motion detection are discussed, both issues from the algorithmic side and issues with the environment in which the video is captured. The most robust algorithm found is presented, and some of the properties that algorithms used for tracking net and masks should have are shown.

Sammendrag

I løpet av denne masteroppgaven skal muligheten for å bruke maskinsyn til navigasjon, posisjonering og orientering for undervannsfarkoster undersøkes. Bruk av maskinsyn har lenge vært problematisk, både med tanke kvaliteten på kameraer og på regnekraften som trengs for å få gode data ut av et kamera. I nyere tid har dette blitt enklere da datakraften har økt samtidig som utviklinger i kamerateknologi har gitt tilgang til mindre og bedre kameraer.

Gjennom denne oppgaven vil forskjellig klasser av algoritmer innen maskinsyn presentert. Fordeler og ulemper med de forskjellig er tatt frem for å vise hva de forskjellige klassen spesialiserer seg på å finne. I tillegg til dette ble det kjøpt inn mye nytt utstyr for å kunne gi innsikt i høydefinisjonsvideo og tilgang til å gjøre tester med dette utstyret. Utstyret ble kjøpt etter spesifikasjoner som bruker i undervannsfartøy i dag.

De problemene som dagens datasynsalgoritmer prøver å løse er diskutert. Dette inkluderer både fra et algoritmisk ståsted og fra hvordan miljøet endrer seg over og under vann. Den mest robuste algoritmen som ble funnet blir viet litt ekstra tid, hvor egenskapene som gjør denne algoritmen bedre egnet enn de mer populære maskinsynalgoritmene blir vist.

Contents

List of Figures

List of Tables

Glossary and Terms

BNC	is a quick connect and disconnect connector often used of coaxial cables.
C++	is an open and statically typed programming language. Based on the C language, it is known for its good optimization and speed.
DirectShow	is a Microsoft™technology that is part of the DirectX multimedia framework. It gives a generic interface to operate on media and streams on the Windows platform.
ITK	Department of Engineering Cybernetics at the Faculty of Information Technology, Mathematics and Electrical Engineering, NTNU.
JIT	stands for Just-in-time compilation. Fragments of code are compiled to machine language on demand.
Marshalling	is the process of transcoding or changing the representation of an object in memory for consumption by some receiver. Also known as serialization.
NRK	Norwegian Broadcasting Corporation.

Odometry	is the technique of using data from moving sensors to estimate a change in position of a object over time.
OpenCV	is a free and open source cross-platform library that is tailored towards computer vision. It was originally developed at Intel, but is now available under a open source license.
Python	is an open and dynamically typed high-level programming language. It is mainly designed for fast development and readability.
SINTEF	was previously known as ”Stiftelsen for industriell og teknisk forskning ved Norges tekniske høgskole (NTH)”, now only SINTEF. The largest independent research organization in Scandinavia.
Windows Driver Model	is the Windows generic framework for device drivers.

List of Acronyms and Nomenclature

8N1 Common configuration for communicating over a serial port on a computer. Describes the protocol as 8 bits for data, No parity data and 1 stop bit.

AUV Autonomous Underwater Vehicle.

DVL Doppler Velocity Log.

GND Ground Pin.

HLT Hough Line Transform.

$I(x, y, t)$ Intensity at position (x, y) at time t.

$I_x(x, y, t)$ Intensity spatial derivatives for x direction at position (x, y) at time t.

LVDS Low Voltage Differential Signalling.

$\mathcal{O}(\partial^2)$ Second order and higher terms.

PHT Probabilistic Hough Line Transform.

ROI Region Of Interest.

ROV Remote Operated Vehicle.

RXD Receive Data Pin.

SDK Software Development Kit.

TTL Transistor-transistor logic. Describes the voltage levels used for signalling in integrated circuits. Usually 3.3 V or 5 V.

TXD Transmit Data Pin.

Chapter 1
Preface

Most of the hardware oriented and data related parts of this thesis has been done in collaboration with Sletta-Heimdal [19].

1.1 Acknowledgements

I would like to thank Terje Haugen at the ITK departments mechanical workshop for helping with design and the actual making of the underwater housing for the test system.

A thank also goes to Kevin Frank, Per Rundtop and Erik Høy from SINTEF for expertise and feedback on the problems of aquaculture and use of their test rig and cooperation in the field testing.

A special thanks goes to Trond Viggo Melum from NRK for help with troubleshooting the HD-SDI interface.

Thanks to Rolf Hansen at Malvik Båtforening for lending us some space in the harbour for the rigging during our field tests.

Morten Sletta-Heimdal, Stein Melvær Nornes and other involved in proofreading also deserves a small applause.

And last but not least, a big thank to Jo Arve Alfredsen from ITK for support and brilliant guidance during the rougher times of this thesis and support throughout the project.

Chapter 2
Introduction

2.1 Motivation

During the last four decades, the demand for fish has been steadily increasing. The current level of demand supersedes the amount of fish that can be fished while keeping the different stocks of fish at a sustainable level. This has lead to overfishing, eliminating for instance the fish population on the Grand Banks off the east coast of America. Many other fish populations are also either gone or rapidly diminishing. Some scientists are even predicting that with the current overfishing, the fish industry will see a global collapse during this century as pointed out by Worm et al. [23].

On way to combat the overfishing of natural fish stocks is to increase the production of fish through farming. This way of ensuring sustainability for both stocks and people has been used for millenniums on land, and is now being rapidly deployed at sea. According to FAO [8], the amount of fish that were farmed globally in 2005 was 48.1 million tons, out of a total 141.3 million tons of fish that were produced. This means that about a third of all fish sold in 2005 were farmed, with steadily increasing numbers.

The expansion of the fish farming industry causes more rigid safety frameworks to be implemented. One of the big issues with the current method of farming, which uses an open mask sea cage to contain the fish, is that the cage is subject to strain and other mechanical stresses. This can cause lacerations and tears in the net causing fish to escape. A typical construction of a sea cage can be seen in figure 2.1.

To monitor the state of the sea cage, ROVs or divers are used to inspect the net. A common sea cage has a volume between $20\,000\,\mathrm{m}^3$ and $80\,000\,\mathrm{m}^3$ as discussed in Jensen et al. [15]. Going through the whole sea cage is a time consuming job, and as people need to operate the ROV and monitor the net at the same time, it can be challenging for the ROV operator to do this multitasking.

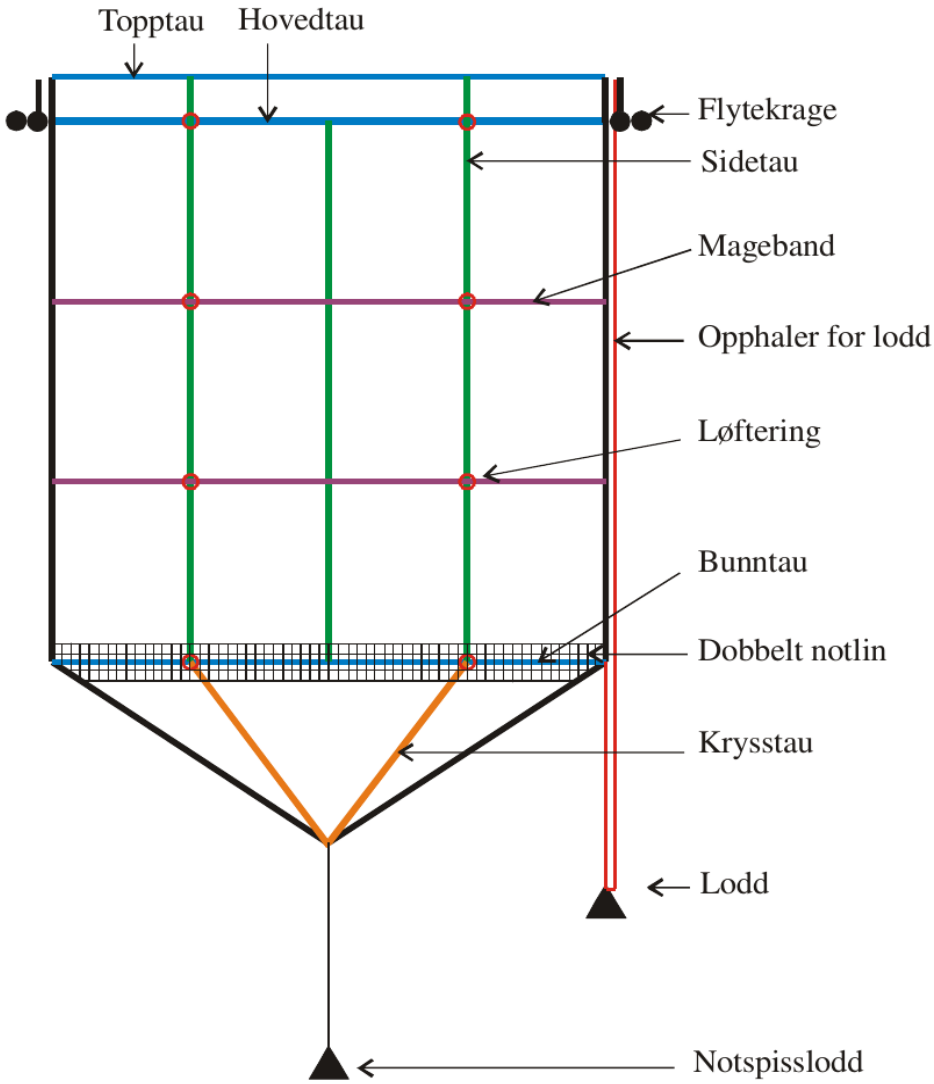


Figure 2.1: Typical construction of a sea cage. Image from Akvaplan-niva [2].

Most ROVs does have on board navigational instruments such as pressure sensors and accelerometers that can be used to find the absolute position of the ROV in the water. The problem with only having absolute positions available, is the fact that the net in the sea cage is a moving entity in the water. The net will deform and move based on the currents that goes through the net. As currents are normally changing over time, this makes the use of a relative position between the ROV and the net a more useful navigational point.

Using the on board video system used for net inspection as a data source for relative positional data would yield a much more useful location than the one available to ROV operators today. The relative movement of the ROV over the net could also be used to trace which sections of the net that has been visited, making sure to get full coverage during inspection.

2.2 Previous Work

This thesis is a further development of Carlsen [5] in cooperation with the MerdROV project at SINTEF. The main point used from Carlsen [5] was the need for modern acquisition hardware. This point was acted upon, and most of the time used on this thesis were used to get a functional hardware platform working. This work was done in cooperation with Sletta-Heimdal [19].

2.3 Limitations

Due to the fact that all the hardware used in this thesis to acquire data had to be bought, only a overview of the different classes of algorithms that may apply to the problem has been given. The main algorithm from each class has been tested, and the most promising one is highlighted. Due to the time constraints with getting both hardware and software to a working state, there were no time to do a thorough test of all algorithms.

2.4 Outline

This thesis is structured as follows;

Chapter 3 This chapter contains the main specifications of the different components of both the hardware and the software in the system.

Chapter 4 This chapter describes the effort in using a laser based orientation system.

Chapter 5 This chapter goes through the different computer vision algorithms used in this thesis, and the different classes of motion detection algorithms that were used.

Chapter 6 This chapter describes the hardware and software used and how to setup the different components.

Chapter 7 This chapter talks about the field test that were done. It also includes the different difficulties found during the test.

Chapter 8 This chapter goes through the application of the theory from chapter 5 and the data collected during the field test in chapter 7. Different problems and advantages of the different algorithms are identified, and the movement data for different test videos are shown.

Chapter 9 This chapter contains the discussion of the results found in chapter 8.

Chapter 10 This chapter concludes on the algorithm that gave the best results from the preceding chapters.

Chapter 11 This last chapter points at the classes of algorithms that were not tested, points about the found algorithm that needs to be quantified and gives different pointers of what should be done to enhance the algorithm.

Chapter 3
Specifications

The hardware and software used in this project was chosen to be as close as possible to the hardware used by Argus in their underwater vehicles. This gives us a scenario closer to the real hardware, and imposes the same restrictions on us as would be imposed in real world applications.

This chapter examines the main specifications of the hardware, its consistency with regards to what is available on a normal ROV, and comparisons where different techniques are available. An introduction is also made to the software platform that was chosen, and some of the considerations regarding that choice.

3.1 Hardware

3.1.1 Camera Specification from Argus

The hardware used by Argus is only mentioned as

- 1× Foc/Zoom camera
- Optional HDTV Camera 1080i
- 1× Lowlight Black & White camera

in Argus [3]. After some mailing with Argus, they provided us with the specific details of this HD Camera, which is what we are interested in. The most interesting parts of the camera specification is shown in table 3.1 on the following page.

3.1.2 High Definition Video Hardware

The camera used by Argus is the Sony FCB-H11 Sony [21].

Camera specification	Detail
Image sensor	1/3-type CMOS
Pixels	$\approx 2 \times 10^6$ Pixels
Zoom	10× (Optical) and 12× (Digital)
Gain	Auto and manual (−3 dB to 18 dB)
Signal Noise ratio	> 50dB
Minimum illumination	1.2 lx (F1.8 50IRE)
	1.0 lx (ICR ON F1.8 50IRE)
Video output	HD Analog component Y/Pb/Pr
	HD Digital LVDS Y/Pb/Pr 8 bit
	SD VBS 1.0 $V_{p\text{-}p}$ Negative sync Y/C
Camera control interface	VISCA TTL
Operating temperature	0 °C to 45 °C
Power consumption	9 V ± 3 V DC, 4.8 W

Table 3.1: Selected FCB-H11 Specifications.

As seen in 3.1, the camera outputs digital HD video over Low Voltage Differential Signalling (LVDS) signals. Due to the open standard used by this camera, there are quite a few different add-on cards that converts this LVDS signal to other signalling technologies meant for long distance transmissions.

VISCA Camera Control Protocol

The FCB-H11 can be controlled by the open Sony VISCA protocol. This is based on RS-232C at TTL signal levels at 9600 bauds. Newer cameras can support RS-232C at up to 38400 baud using a 8N1 configuration and no flow control. VISCA contains a rich set of commands to control professional video systems. It is even compatible with the broadcast networking specification of the RS-232C, which means that a single signal line can contain 8 devices including the PC, as it uses 1 byte for addressing.

The implementation of the VISCA protocol that is supported by the FCB-H11 is given in Sony [22]. The FCB-H11 does for instance not implement the broadcasting function of VISCA, and must therefore be on a single RS-232C signal line.

In addition to normal power on/off and setup data, the VISCA protocol can send different control messages. Amongst these are zoom, focus, white balancing, spot focus and different video effects as black and white or negative images that are being processed by the image processor on the camera. This makes the camera very

versatile, either by direct control from a PC using a TTL level translator or by any micro controller using USART.

Sony provides a basic program for testing and driving equipment through VISCA. In addition to this, an open source C library for Windows, Linux and the Atmel AVR micro controller called libVISCA[1] has been used. This makes it very easy to integrate the VISCA protocol into the software if needed as either a component or as a separate library. This also gives any software the possibility to control the camera to tweak the image stream.

Figure 3.1: Sony FCB-H11 High Definition Block Camera, from Sony [21].

3.1.3 High definition video signalling

Many different interface boards are available for the FCB-H11 which gives different output formats from the camera and provides different control interfaces to the camera.. The most common interface cards provides HDMI, USB, Ethernet, SDI and HD-SDI outputs in different configurations and combinations. A comparison of the different technologies is shown in 3.2 on the following page.

[1]Available at http://sourceforge.net/projects/libvisca/
[2]Still in draft

System	Bitrate	Distance	Protocol	Compression Ratio (for 720p Video)
HDMI	10.2 Gbit/s	≈ 15 m	TMDS	0%
USB 3.0	5 Gbit/s	≈ 5 m	Serial	0%
Gigabit Ethernet	1 Gbit/s	≈ 220 m	Serial	≈ 30%
SDI	360 Mbit/s	≈ 300 m	NRZI	≈ 75%
HD-SDI	1.485 Gbit/s	≈ 300 m	NRZI	0%
HD-SDI (optical)	1.485 Gbit/s	≈ 45 km	NRZI (optical)	0%
Dual Link HD-SDI	2.970 Gbit/s	≈ 300 m	NRZI	0%
3G-SDI	2.970 Gbit/s	≈ 300 m	NRZI	0%
6G-SDI	Undecided[2]	≈ 300 m	NRZI	0%

Table 3.2: Different signalling and transmission systems available for video streaming.

For this project, it was decided that we should try to get an interface that can handle a higher bitrate than the camera can produce. Looking at the different options in 3.2, the system from Argus and also thinking of the transmission length, HD-SDI was chosen as the desired technology for transferring the video stream from the camera.

HD-SDI

The HD-SDI signalling standard is a improvement to the older SDI standard. Where the SDI can only transfer images in maximum 576i format, the HD-SDI standard i capable of transferring 720p and 1080i video. Newer versions of the HD-SDI standard is also capable of 1080p and 4K video streams.

HD-SDI is a professional video standard mainly used by TV stations and in other high end systems. It is defined and maintained by the Society of Motion Picture & Television Engineers, SMPTE. More information on the SDI family can be found on the SMPTE website at https://www.smpte.org/standards/.

Some of the advantages of the HD-SDI interface, in addition to the bandwidth and transmission length, are that the interface is self-clocked, self-correcting and supports self-setup and initialization. This means that the interface is more or less plug-and-play.

This interface is coincidentally the same interface used by Argus according to the information that was received from them. However, due to the long distance of transmission when a ROV is submerged, Argus does an on-board conversion of the coaxial HD-SDI signal to an optical carrier. The use of an optical medium to transfer

Figure 3.2: Intertest EM15710 iShoot-FCB-HDSDI interface, from Intertest [13].

HD-SDI is a normal technique when the high definition signals are transferred over a distance that would lead to signal degradation using electrical signalling.

The HD-SDI interface has also been extended into Dual Link HD-SDI, 3G-SDI and 6G-SDI. The Dual Link is a simple extension allowing two HD-SDI streams on one cable. The 3G version specifies a new signalling standard without changing the bitrate relative to the Dual Link. However, the 3G version has full capability of 1080p streams at 60 Hz. Finally, the 6G version is a new and upcoming version made specifically for the new 4K, also known as Ultra HD, video format. The differences between the different resolutions can be seen in figure 3.3.

Gigabit Ethernet

During the last month of the project, two Gigabit Ethernet modules were acquired. The modules are primarily going to be used in future projects, as cabling and instrumentation over ethernet turns out to be simpler.

The interface is shown in figure 3.4 on page 13 with the FCB-H11 connected. This unit gives access to the VISCA protocol described in section 3.1.2 on page 8 over ethernet, together with the video feed. This means that the feed and control signals can be superimposed on any ethernet compatible infrastructure, which is both cheaper and more readily available, as well as easier to interface with most computers today.

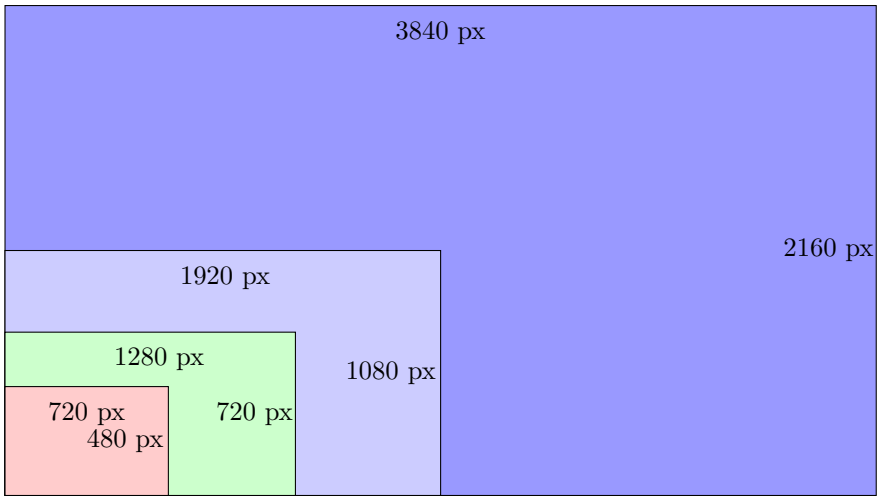


Figure 3.3: Scales of different resolutions. Red: DVD, Green: 720p, Light Blue: 1080p, Blue: UHD (4K).

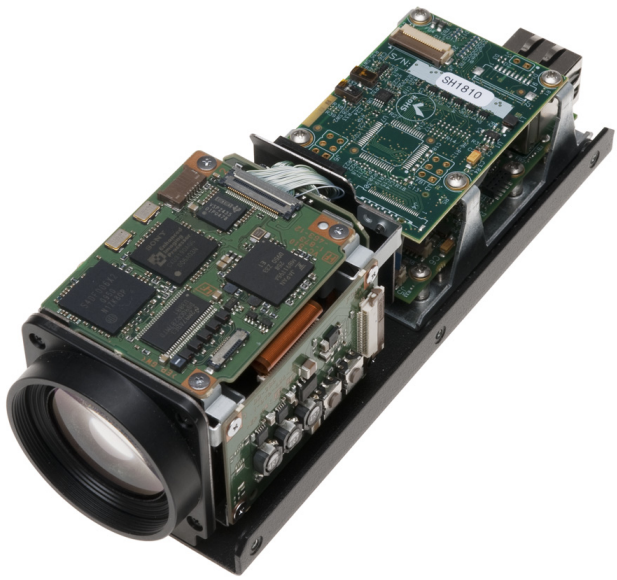


Figure 3.4: Pleora SB-Pro IP Engine for Sony FCB-H11, from Intertest [14].

Underwater Lightning

The earliest image examples provided by SINTEF shows a very clear light gradient in most images. This is due to the optical dampening that water imposes on the normal flow for light. This means that the light intensity becomes clearly dampened even in shallow waters. It also imposes difficulties during software analysis, as most of the techniques available are not designed to handle strong variation in light intensity in a single frame. The gradient will also change in accordance with the position of the ROV, and this can cause the motion analysis to skew over time.

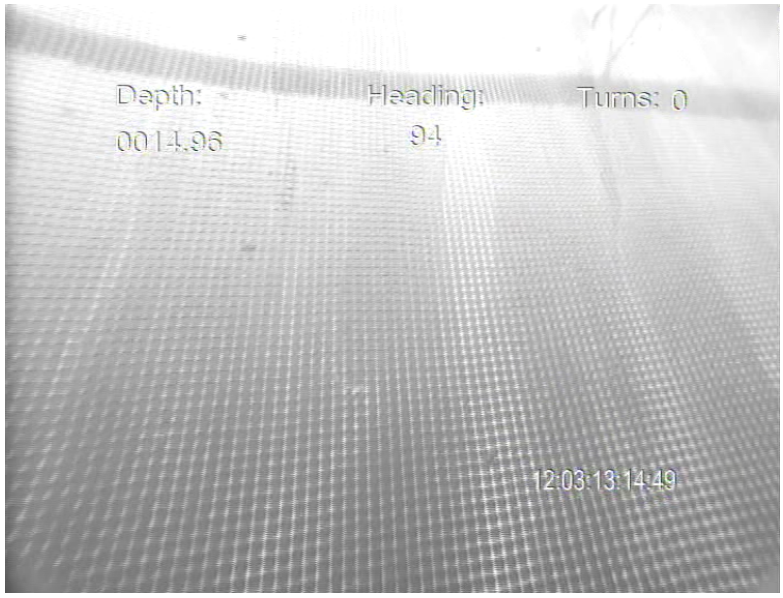


Figure 3.5: Inspection video showing a clear light gradient in underwater conditions. Video provided by SINTEF.

Figure 3.5 on the facing page shows the provided image from SINTEF. The issues with the gradient is minimized by Argus by using a strong LED and HID based light setup.

Video Stream Grabbing

The capture of video streams from a HD-SDI interface usually needs specifically designed hardware that connects to one of the internal buses in the computer to get high enough bandwidth. This is usually done using a PCI Express card that can support multiple HD-SDI streams simultaneously.

This is however not practical, as we are going to use different computers and laptops during our testing. It would therefore be better to get an external capture interface

that connects to the computers using an external computer interface. Thankfully, with the development of USB 3.0 we are able to support the bandwidth requirements for HD signal, as the USB 3.0 interface has a theoretical maximum bitrate of 5 Gbit/s.

The Ultrastudio SDI from Blackmagic Design was chosen as the stream grabber. This is a industry standard supplier the develops different high-end equipment. More information on this can be found at http://www.blackmagicdesign.com/products/ultrastudiousb3/ and the unit is shown in 3.6.

Figure 3.6: Blackmagic Ultrastudio SDI for USB 3.0. http://www.blackmagicdesign.com/products/ultrastudiousb3/.

The Ultrastudio SDI unit provides HD-SDI in together with HD-SDI out, HDMI out and USB 3.0. The extra HDMI out makes it easy to connect an external monitor while capturing to get better visual feedback.

The ethernet module described in section 3.1.3 uses multicast technology to transfer the video stream over IP. The card comes with a open Software Development Kit (SDK) to discover and stream from any unit on the local network. As an added bonus, the ethernet module allows the VISCA control commands from section 3.1.2 on page 8 to be transferred over the same ethernet connection, removing the need for extra signalling cabling.

3.2 Software

The software used is split into two different categories; the video capture software provided by Blackmagic Design or any other compatible platforms, and the video analysis software developed in this project. Fortunately, the Ultrastudio unit comes with Windows Driver Model drivers and DirectShow filters, and a software development kit which makes it somewhat easy to connect to the unit and stream the video from it in real-time. This means however that the direct capture of the video stream unfortunately is limited to the Windows operating system.

3.2.1 High Definition Video Capture

In theory, all Windows Driver Model compatible software can be used to access the video stream from the Ultrastudio unit. It turns out however that the provided software from Blackmagic has some extra interfaces which makes it a bit easier to use, for instance to change resolution on demand, jitter removal and different behavioural aspects.

Blackmagic provides the Media Express platform for capture. This is a basic but useful starting point. The main drawback is the small amount of capture formats available in the software, and the frequent crashes that were experienced during use. However, the program were stable during capture, and it is capable to capture to YUV-8 bit which means almost as raw data as possible. This format is also easy to re-encode to other formats at a later stage for storage.

It turned out that the image processing stack used in the project also was able to connect directly to the Ultrastudio unit, meaning that the video stream could be fed straight into the different algorithms. This was useful for testing on-line performance, as opposed to storing the stream and doing off-line analysis.

3.2.2 High Definition Video Analysis

The de facto standard in computer vision is the OpenCV library. The OpenCV library is implemented in C++, and it contains bindings multiple other languages. Especially the Python binding makes quick prototyping easy. Matlab was also considered, as Matlab does contain a quite powerful computer vision toolbox[3].

During early prototyping, several discrepancies between the OpenCV algorithms and the equivalent Matlab algorithms were discovered. Multiple standard algorithms would behave differently with the same source and same parameters. To keep the prototyping and the real implementation of the different algorithms as close as possible, the prototypes were implemented in Python. The modules for OpenCV in Python uses the same library source as the C++ library, ensuring coherent algorithms.

3.2.3 Real-Time Considerations

This thesis is mainly concerned with the question if the video stream from the camera can be used as a navigational or odometric data source. This gives rise to some specific real-time issues that needs to be addressed in a working system. The speed is constrained by both the amount of data that the camera produces and the computational power of the system that is doing the end analysis.

[3]Equivalent to library

For the system to be usable as a data source for navigation, it must have a couple of properties. Some of them are listed below.

- Provide data to the end user within a short time
- Provide data on a format compatible with other equipment
- Provide data that does not have negative affect on the navigation
- Provide consistent data

When the term short time is used in this context, it is meant that the data must arrive at a time after the initial capture that is not long enough for the data to be invalid. This definition is vague by design, as the time for data to invalidate is highly dependent on the system that receives the data.

Compatibility is also important, as these types of systems usually is interconnected with other systems available to the end user to provide data from multiple sources. This means that data formats should be open and easily interchangeable.

It is also paramount that the data provided should not have a bad impact on the system. If the data is not accurate or has a tendency to drift, then the inclusion of the data may cause the system to degrade.

Lastly it is important that the data does not skew or change over time and in different environments. If the system needs to be calibrated as the sun changes its position or if a fish swims in from of the camera, then the system is not a consistent source of data. This again means that the whole navigation system degrades as the data from the video becomes more unusable. It is important to have algorithms that minimizes the effect of environmental disturbances.

Chapter 4
Orientation based on Laser Projections

This chapter describes the work done to enhance the laser based guidance developed in Carlsen [5]. The use of lasers to project a known geometrical shape onto the net is thought to be a good way to get consistent information on the orientation relative to the net. It is important to check which figures that would be easy to recognize in software, and also gives a good indication on both rotation, tilting and distance relative to the net.

4.1 Projections

The use of some regular and known geometrical shape that could be projected onto the net for use for orientation was put forward quite early. The idea behind this is based on the work by Carlsen [5], who used a set of line lasers used to measure the distance between the ROV and the net.

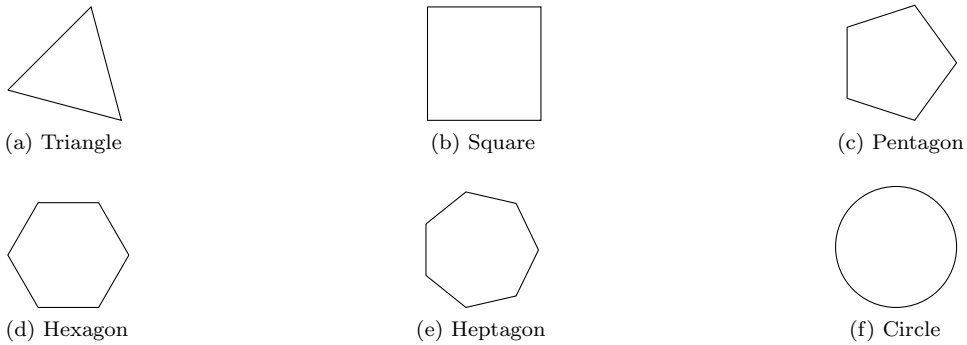
This is done by projecting the lines onto the net, so that two lines of red dots appear. Since the lines are projecting parallel lines, the dots should line up with a median distance between them. This distance can be counted in pixels and used as a proportional measure of the distance, as the distance between the points follows a inverse relationship with the distance to the net.

4.2 Patterns

There are many different patterns that may give more information about orientation of the ROV relative to the net than some simple lines. The different polygons in figure 4.1a on the next page to 4.1e were chosen as a good start for further investigation together with the circle as shown in 4.1f.

Criteria for a good pattern should contain

1. Not ambiguous when interpolating lines



(a) Triangle (b) Square (c) Pentagon

(d) Hexagon (e) Heptagon (f) Circle

Figure 4.1: Different laser patterns

2. Easy algorithm to reconstruct shape
3. Try to give orientation information
4. Avoid the aperture problem, see section 5.3 on page 26

It can be shown that amongst the patterns in figure 4.1 that all patterns with a even number of vertices is prone to the aperture problem. This means that pattern 4.1b and 4.1d can be eliminated right away. Looking at the pentagon and heptagon, they have respectively $\frac{360^\circ}{5} = 72^\circ$ and $\frac{360^\circ}{7} \approx 51.4^\circ$ angle for each corner, and the angles get closer as the number of vertices increases.

The best choice seems to be either a triangle or a circle. The triangle is easy to reconstruct. Only two of the vertices needs to be known to do a full reconstruction if the angles of the triangle is known. The orientation can also be determined within a third of a full revolution without ambiguity. However, the triangle gives ambiguity each $\pm 60^\circ$, so the problem of ambiguous orientation can not be solved using a regular pattern.

Irregular patterns are also possible, and using an irregular pattern would remove all ambiguity of the operation. A irregular pattern is however much more difficult to recreate out of partial lines. Due to this added complexity, a simple circle turns out to be the best choice.

A circle can not convey any orientation data, as there are no corners and vertices to link an angle to the net onto which it is projected. It can however give information on distance to the net by checking the amount of masks in the net that is inside the circle. The deformation of the circle will also give information on the tilt of the ROV relative to the net. The shape that would be read back in this case would be an

ellipse. The ration of the minor- and major axis in the ellipse would give estimates on these angles.

Ellipse are quite easy to detect using computer vision. The most used algorithm is described in Fitzgibbon and Fisher [10]. This algorithm fits an ellipse onto a set of points, usually >5 is needed. The algorithm optimizes the error of the found ellipse for all points. The error would then be a measure of the deformation caused by movement in the net.

The implementation shown in figure 4.2 on the following page proved to be quite accurate and stable. A test pattern was generated where a random number of points were laid out on an elliptical path where the points were allowed to move in all directions. As figure 4.2b on the next page shows, points that diverge much from the elliptical path does not affect the found ellipse significantly.

4.3 Algorithm

The algorithm used to generate the ellipse in 4.2b is based on a chain of standard computer vision algorithms and filters. The chain consists of

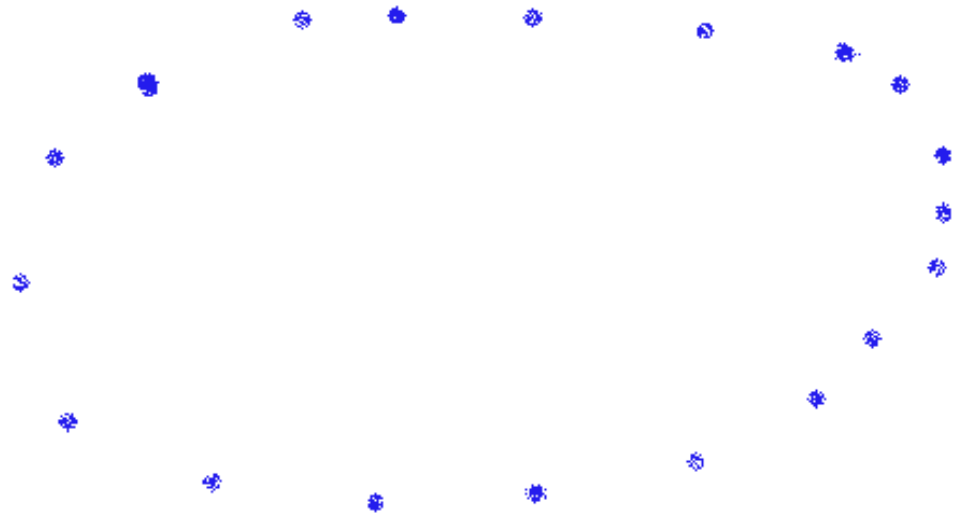
1. Colour conversion to greyscale.
2. Gaussian smoothing. See section 5.1.3 on page 25 for the algorithm.
3. Detection of points using the Hough Circle algorithm. See section 5.2.3 on page 26 for the algorithm.
4. Discarding of points outside a Region Of Interest (ROI).
5. Use the Fitzgibbon and Fisher [10] algorithm to fit an ellipse over the remaining points.

4.4 Hardware

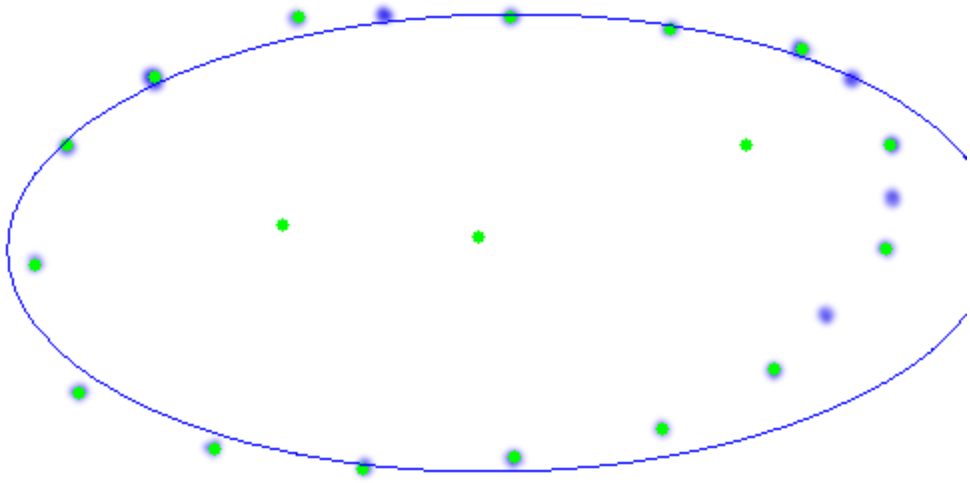
To manufacture the hardware, a specification was developed together with Diode Laser Concepts, INC[1]. This is a reseller that specializes in all sorts of laser equipment, and has been use by the institute before. It turned out that circular lasers are not a common configuration, so after some discussion we were referred to Frankfurt Laser Company[2] which is a laser manufacturer based in Europe.

After even more discussions with Frankfurt Laser Company, it became clear that manufacturing such a specialized laser as the one required would be difficult. The

[1] www.diodelaserconcepts.com
[2] www.frlaserco.com



(a) Simulated points



(b) Detected ellipse

Figure 4.2: Simulation of a projected ellipse and the detected ellipse. Blue point sets are set to vibrate in a random pattern, green dot is the detected point and blue circle is the fitted ellipse.

quote from Frankfurt Laser Company had a minimum order of 1,500 units. Due to the high manufacturing cost for single units, the efforts for acquiring a laser were cancelled.

The lack of laser hardware means that the algorithm used to detect ellipse was never proven to work outside of simulations. Real life data on the performance of the detector were therefore not available. However, based on the dynamic simulations, the ellipse were always detected successfully. The quality of the fitting is very reliant on the number of points which are detected. Even though the algorithm claims to work as long as it has 5 points or more, it is obvious that the fit gets better with more points.

Even with detection errors as shown in figure 4.2b where points are being detected on the background and some points that should have been detected are not, the fitting seems quite robust. This is partly because of the minimization that the Fitzgibbon and Fisher [10] algorithm does, and partly the fact that the points simulated clearly describes an elliptical shape. If many points start to diverge from the approximate path, the fit gets increasingly worse.

Chapter 5
Computer Vision and Algorithms

The use of computer vision has seen a substantial growth over the last 40 years, as camera technology matured and the analysis of data became cheaper and faster. The wish to enable the usage of video streams to measure the environment is old, and has driven a spectacular development in algorithms within this field. In this chapter we are going to look at the main algorithms that were tested and applied in the project. Some more in-depth functionality of the algorithms and mathematics behind each of them are also provided.

Most of the theory in this chapter is collected from Sonka et al. [20], Schellewald [18] and Davies [6].

5.1 Filtering

Smoothing is a common and necessary technique in computer vision to remove artefacts and noise from images. The normal way of smoothing a image is to apply a filter to each pixel of the image through convolution. This makes all filter knowledge from the field of signal processing available.

The use of convolution also standardizes the structure of filters, and makes them easily interchangeable. Some of the most used filter kernels are shown in the following sections.

5.1.1 Image Convolution

The normal structure for different filters is usually described using a convolution kernel. This convolution kernel describes how neighbouring pixels will affect the pixels that is about to get filtered. This description is given using the convolution

formula in (5.1).

$$f(i,j) = \sum_{(m,n)\in\mathcal{P}} \sum h(i-m, j-n)g(m,n) \tag{5.1}$$

In equation (5.1) the f denotes the resulting pixel value, g is the original pixel and h is the convolution mask or kernel. In this context, $\mathcal{P}$ denotes the set of pixels in the neighbourhood of (i,j) for which the kernel is non zero.

The easiest example of a convolution kernel is the averaging kernel. The averaging coefficient is dependent on the dimensions of the kernel. A averaging kernel of dimension 3×3 is shown in (5.2). This kernel gives a pixel the average weight of all neighbouring pixels.

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{5.2}$$

The convolution operation is quite an expensive operation. This has led to a favouring towards separable kernels. Kernels that are separable have the property that they can be decomposed into two one-dimensional filters. The computational gain of this can be quite significant.

If an image containing $n \times m$ pixels are to be convolved with a $p \times q$ kernel, this takes approximately $nmpq$ multiplication and additions. If however, the kernel can be separated, the first dimension takes approximately nmp adds and multiplications, and the second uses nmq. This results in a complexity of $nm(p+q)$ which usually is a smaller number than $nmpq$. The ratio between this is shown in equation (5.3).

The approximations that leads to equation (5.3) is from Eddins [7] and is somewhat crude on small dimensions. It is however asymptotically correct for larger dimensions of the kernel.

$$\frac{nmpq}{nm(p+q)} = \frac{p \times q}{p+q} \tag{5.3}$$

This causes a 9×9 kernel to have a computational gain of $\frac{9\times 9}{9+9} = 4.5$ using this principle of separable kernels. The ratio will increase in benefit for separability if the dimensions increase.

5.1.2 Normalized Box Filter

The box filter with normalization is the simplest of all filters. Each pixel is convolved with a kernel that weights all neighbouring pixels equally. One example of this kernel has already been shown in (5.2).

5.1.3 Gaussian Filter

This is the most used filter. The kernel used in this filter approximates a Gaussian function. This kernel is usually generated based on a set size with given mean and variance which is then normalized.

This kernel would be a discrete approximation to the two dimensional Gaussian filter equation shown in equation (5.4).

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{5.4}$$

5.1.4 Median Filter

This filter is almost as easy as the Normalized Box filter in section 5.1.2. The only difference is that in stead of a normalized identity kernel, this kernel is the median of the pixel values of the neighbourhood.

5.2 Hough Algorithms

The set of Hough algorithms uses a trick to find lines in a image. The trick is instead of using the Cartesian description $y(x) = ax + b$ for lines, to rather use the a polar parametrization in equation (5.5).

$$r(\theta) = x_0 \cos\theta + y_0 \sin\theta \tag{5.5}$$

This (r, θ) space from equation (5.5) is usually denoted as the Hough space.

The following sections describe different specific implementations of the Hough algorithm. For deeper details on these algorithms, please see Sonka et al. [20] for a full description and a good description of the different sub algorithms.

5.2.1 Hough Line Transform

The Hough Line Transform (HLT), usually called just the Hough Transform, uses equation (5.5) together with a global accumulator matrix. For each pixel in the image that is being analysed, the equation is being applied to each pixels and its neighbours.

If there are enough evidence that there is a line at that point, the accumulator is incremented at the position that indicates the found line.

When the whole image has been analysed, the accumulator is inspected. Lines should travel through multiple points, each which would increment the accumulator leading to peaks. Each peak in the accumulator corresponds to a line in the (r, θ) space, and can be extracted and recreated.

5.2.2 Probabilistic Hough Line Transform

The Probabilistic Hough Line Transform (PHT) is an extension to the HLT. The extension consists of some sort of probabilistic function that is applied after the accumulation has been done. This function can for instance be an a-priori function that describes the probability of the existence of lines at a given point.

Another common version uses a least square approach. This does not need any a-priori knowledge, and is therefore the preferred way of implementing this function. The benefit of using this type of algorithm over the HLT is that the lines defined by the accumulator no longer have to be global lines. This comes from the fact that a least square solution can terminate a line if the error between the described line and the probability of the line in the image goes beyond a given threshold.

5.2.3 Hough Circle Transform

To use the algorithm previously shown to detect circles, a bit more information is needed. The circle with centre (x_0, y_0) will be defined by $C : \{x_0, y_0, r\}$ where r is the radii. Other than that, the algorithm is the same as the one in section 5.2.1.

There is also a Cartesian implementation of this algorithm, in which each pixel is assumed to be the centre of a circle.

5.3 Aperture Problem

The aperture problem in computer vision usually arises during optical flow problems. The issue is that any differential flow algorithm is only able to determine flow that is normal to the edge which is being traced. This means that other flow components that is not normal to a edge are not possible to determine.

Figure 5.1 on the facing page attempts to illustrate how the aperture problem arises. If the area that are being tracked only contains a single edge, then it is impossible to say if there is movement in any direction that is not parallel to the normal direction of the edge. This means that tracing a single edge is not favourable, and all modern

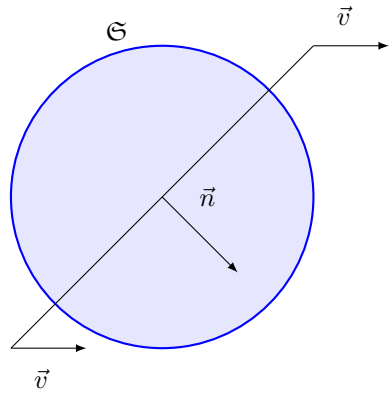


Figure 5.1: Aperture problem, figure inspired by Yazdanbakhsh and Gori [24]. If the line defined by $\vec{n}$ is observed only in the area defined by $\mathfrak{S}$, then any movement along any $\vec{v}$ will be observed as movement along $\vec{n}$.

algorithms will try to trace on corners, as a corner by definition consists of at least two non-parallel edges

5.4 Optical flow

Optical flow is a term that covers the apparent motion of objects, specifically its edges and surfaces relative to an observer. The term was first coined for this by James Gibson in the 1940s to describe the visual stimulus that animals get from movement in Gibson [11].

5.4.1 Theory

The foundation of optical flow theory is based on the normal flow constraint equation in (5.6). In equation (5.6), I is the intensity at a given point ant at a given time. It is quite obvious that a normal optical flow problem will be of three dimension as it will include spatial coordinates in addition to a temporal part. The deltas shown is an indefinite movement in the three dimensions.

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{5.6}$$

If the deltas in equation (5.6) are assumed to be small, then the approximation in equation (5.7) is valid.

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \mathcal{O}(\partial^2) \quad (5.7)$$

Equation (5.7) contains a higher order collection term $\mathcal{O}(\partial^2)$. This can be thought of as the error in this first order Taylor approximation. As the deltas are assumed small, then the Taylor expansion terms with order higher than one are negligible.

Substituting equation (5.6) into (5.7) it is obvious that equation (5.8) must hold. By dividing equation (5.8) with the change of time we get (5.9), where V_x and V_y are the spatial velocities in the x and y direction.

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \quad (5.8)$$

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = \frac{0}{\Delta t}$$

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \quad (5.9)$$

By defining the intensity spatial derivatives as I_x, I_y and I_t, we get equation (5.10). Using the dot product on the derivatives leads to equation (5.11), which is the usual way of writing the optical flow problem.

$$I_x V_x + I_y V_y = -I_t \quad (5.10)$$

$$\nabla I^\top \cdot \vec{V} = -I_t \quad (5.11)$$

Equation (5.11) shows the big problem of optical flow problems. It is a single equation with two unknowns, and is therefore not solvable without adding additional constraints to the analysis. The problem is known as the aperture problem, and all of the following algorithms introduces some additional condition that infers constraints to the flow problem.

5.4.2 Phase Correlation

As the name of this method implies, this method uses shift and correlation in the phase plane between to frames as a measure of motion. Given that correlation is

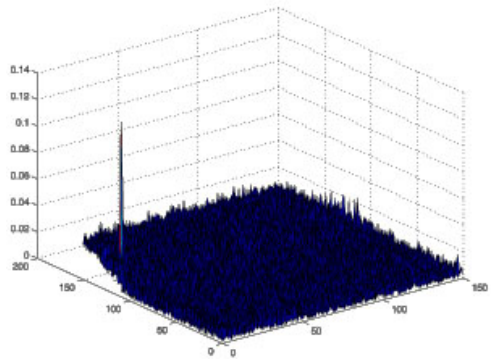


Figure 5.2: Image showing spatial result of phase correlation. Image collected from ee3 [1].

a global operation, the method is also image global. The first step is to apply the Fourier transform to both frames, $\mathbf{G}_a = \mathcal{F}\{g_a\}$ and $\mathbf{G}_b = \mathcal{F}\{g_b\}$. The cross-power spectrum can then be calculated with equation (5.12).

$$R = \frac{\mathbf{G}_a \circ \mathbf{G}_b^\star}{|\mathbf{G}_a \mathbf{G}_b^\star|} \tag{5.12}$$

By inverse transforming R, we get the normalized cross-correlation, $r = \mathcal{F}^{-1}\{R\}$, which can be though of as a heat map of the movement between the two frames. A example of this can be seen in figure 5.2. After the inverse transform there are usually some statistical estimation of the centre of the strongest peak that was detected. This ensures that the noise in the peak are minimized.

There are however some drawbacks with using the phase correlation method, mainly the global aspect. The global aspect infers smooth and equal translation of all points between the frames. The method has its base in the Fourier shift theorem, which implies that the images are assumed to have an even shift.

Figure 5.2 shows the spatial result of the phase correlation algorithm. The peak seen gives the coordinates of the assumed translation that was detected. If multiple peaks of equal strength are detected, then some other movement than linear translation is starting to become significant. An example of the phase correlation during rotation can be seen in figure 5.3.

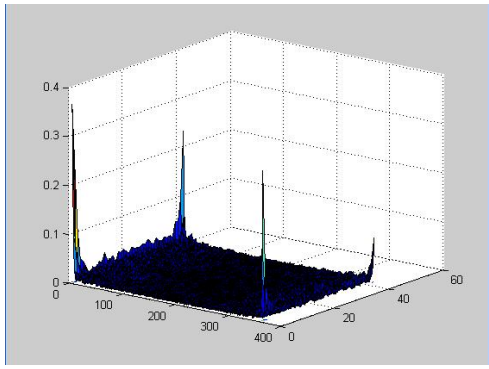


Figure 5.3: Image showing spatial result of phase correlation. The multiple strong peaks implies that some rotation also took place. Image collected from ee3 [1].

5.4.3 Lucas-Kanade

The Lucas-Kanade method is a widely used method for providing the needed constraints to solve (5.11). This method assumes that the motion between two frames is small and more or less constant in the neighbourhood of a point, known as the velocity smoothness constraint. This means that (5.11) holds for all points within some window with the point that is being considered. This means that the local velocity vector should satisfy (5.13).

$$\begin{aligned} I_x(q_1)V_x + I_y(q_1)V_y &= -I_t(q_1) \\ I_x(q_2)V_x + I_y(q_2)V_y &= -I_t(q_2) \\ &\vdots \\ I_x(q_N)V_x + I_y(q_N)V_y &= -I_t(q_N) \end{aligned} \tag{5.13}$$

Rewriting (5.13) on matrix form using the normal $Av = b$ structure, we get the matrices in (5.14).

$$A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_N) & I_y(q_N) \end{bmatrix} \quad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix} \quad b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_N) \end{bmatrix} \tag{5.14}$$

The equation set in (5.14) is overdetermined. The Lucas-Kanade method now uses

the least squares method to find the minimal solution to this set. The system that then must be solved usually takes the form of (5.15).

$$v = \left(A^\top A\right)^{-1} A^\top b \tag{5.15}$$

If equation (5.15) is expanded back to its original matrices and using sums, we have the system in (5.16)

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x^2(q_i) & \sum_i I_x(q_i) I_y(q_i) \\ \sum_i I_y(q_i) I_x(q_i) & \sum I_y^2(q_i) \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix} \tag{5.16}$$

5.4.4 Horn-Schunck

The Horn-Schunck method is another global algorithm that was first developed in Horn and Schunck [12]. It assumes that the flow in a frame is smooth and will minimize any error in this assumption. It formulates the flow as an energy function on the form (5.17), which can be minimized using the corresponding Euler-Lagrange equations in (5.18) with $L = \int E \, \mathrm{d}t$.

$$E = \int \int \left(\left(I_x u + I_y v + I_t\right)^2 + \alpha^2 \left(||\nabla u||^2 + ||\nabla v||^2 \right) \right) \mathrm{d}x \, \mathrm{d}y \tag{5.17}$$

$$\begin{aligned} \frac{\partial L}{\partial u} - \frac{\partial}{\partial x}\frac{\partial L}{\partial u_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial u_y} &= 0 \\ \frac{\partial L}{\partial v} - \frac{\partial}{\partial x}\frac{\partial L}{\partial v_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial v_y} &= 0 \end{aligned} \tag{5.18}$$

The solution to the system in (5.17) and (5.18) uses the same arguments as those given in Khalil [16] for energy based system analysis. The full analytical steps and solution, which will be omitted here, can be found in Horn and Schunck [12]. The result usually takes the form as the iterative equations in (5.19).

$$\begin{aligned} u^{k+1} &= \bar{u}^k - \frac{I_x \left(I_x \bar{u}^k + I_y \bar{u}^k + I_t \right)}{\alpha^2 + I_x^2 + I_y^2} \\ v^{k+1} &= \bar{v}^k - \frac{I_x \left(I_x \bar{v}^k + I_y \bar{v}^k + I_t \right)}{\alpha^2 + I_x^2 + I_y^2} \end{aligned} \tag{5.19}$$

5.4.5 Farnebäck

The algorithm proposed by Farnebäck [9] is a method that is quite new, but it has started to get a firm foothold in the field of optical flow analysis. It has replaced the Horn-Schunck algorithm mentioned in section 5.4.4 in the OpenCV library.

The algorithm uses quadratic polynomials to approximate the neighbourhood of each pixel. This approximation is then used to estimate the displacement fields for that neighbourhood by observing the how the exact polynomial transforms during the translation. The algorithm is shown to be robust and reasonably inexpensive to compute compared to the Horn-Schunck and the Lucas-Kanade algorithms.

The Farnebäck algorithm does not contain much advanced mathematics, but the theory behind it is somewhat complex. It is not something that will be given in this thesis, but can be read in Farnebäck [9]. The main idea, however, of the algorithm is to find the `flow` so that equation (5.20) holds.

$$\mathrm{P}_{t-1}(y,x) \sim \mathrm{P}_t\big(y + \mathrm{flow}_y(y,x), x + \mathrm{flow}_x(y,x)\big) \tag{5.20}$$

The equation in (5.20) would have been similar to the flow constraint equation in (5.6) if the similarity in (5.20) is replaces with a equality. This similarity constrain comes from the quadratic approximation done in Farnebäck [9], and shows that the mathematics of this method is quite different from the other and older approaches to the optical flow problem.

To keep this algorithm local, the image is usually split up into a grid of sub-images where the equation (5.20) is applied. This does not only keep the algorithm local, it also makes it possible to run the flow estimation in parallel as all sub-images are independent. The number of sub-images must be chosen so that the algorithm contains its local perspective. Too many sub-images can however have a negative effect on the calculation time, as more time would be spent dividing and reassembling the image.

5.4.6 Nyquist-Shannon Sampling Theorem and Optical Flow

The movement in this thesis is centred about movement over a repetitive and uniform surface. The edges of the masks in the net gives the sampling points that the motion tracking algorithm uses. The distance which these lines can move between each time the data is sampled, e.g. a frame is saved, is covered by the Nyquist-Shannon sampling theorem. More on the theory behind this can be found in Proakis and Manolakis [17].

The part of the theorem that has impact on the repetitiveness of the object that is going to be tracked, is the fact that if the sampling is happening too slowly, then the net will start to show aliasing. As most of the theorems are greedy, they always assume that the movement is the smallest possible. This means that if a mask travels more that half of the mask height or breadth, the algorithms will fail. The mask that were tracked will then be assumed to be the next mask in the opposite direction of movement. This will again cause the algorithm to always be predisposed to give the wrong direction of travel if the movement is to large.

The change of direction is the classic term of aliasing, and can be seen in everyday situations. Wheels on cars can for instance look like they are travelling slowly backwards, when they in fact are travelling forwards. Examples of such aliasing can be seen in Proakis and Manolakis [17].

Chapter 6
System Implementation

6.1 Hardware Setup

6.1.1 Camera

The camera is described in section 3.1.2 on page 7. There were not done any modifications to the camera itself during the test.

The connections from the camera are quite simple. One main flat flex cable provides power to the camera and Transistor-transistor logic. Describes the voltage levels used for signalling in integrated circuits. Usually 3.3 V or 5 V (TTL) level serial signal for the VISCA protocol. In addition to power and control signal, the cable also gives access to a analogue HD and SD output. The SD output is usually disabled, unless it is explicitly activated in software. The analogue HD uses default Y/Pb/Pr component signalling, and is compatible with most analogue HD monitors and screens.

6.1.2 HD-SDI Interface Card

The HD-SDI card can be mounted onto a small bracket on the back of the camera. This gives a small footprint for the unit. The card is mounted as shown in figure 6.1, where all the connectors are described in the figure caption.

The interface has a couple of different functions in the setup. Mainly it does the conversion and packing of the LVDS signal from the camera into HD-SDI according to both the HD-SDI protocol and the electrical specification. The video data is then available on a MCX7 connector which can be connected to normal BNC connector using a provided adapter.

There is also a big flat flex cable between the camera and the interface card. This cable is mainly used for sending and receiving the VISCA commands to and from the camera. In this situation the interface card provides a level translator so that the signal can be connected directly to a serial port on a computer. This serial signal is

available on a breakout cable, together with ground references and the analog video outputs from the camera.

The card also has a single status led to indicate what operating mode the HD-SDI bus is in. On startup the led will be red. This means that the video input can not be streamed out to the HD-SDI interface. This usually resolves itself during the setup of the HD-SDI interface between the card and the recipient on the other side of the cable. When the setup process is completed, the led will change to green indicating that the card is now streaming data.

During power up, the camera will close and open the iris. This indicates that at least power is getting through. During HD-SDI setup, the camera will continue to do this, and it will make clicking sounds. This is a good way of getting feedback without having to constantly looking at the interface card.

6.1.3 Underwater Housing

As the camera needs to be submerged in water, a waterproof housing for the camera was made by the mechanical workshop at the institute. The housing can be seen in figure 6.2 on page 38. The housing is made up of a tube of clear acrylic plastic with two end pieces. The back piece has a sliding bracket that the camera can be mounted to. It also has two nipples for the two cables that is going to carry the video signal and the power and control signals.

As seen in figure 6.2 the housing also comes with a outer mounting bracket in white acrylic. This is used to attach the housing to the underwater part of the rig as described in section 6.1.8 on page 44. Both the front and back cover of the housing is removable using three screws. To ensure that the housing is waterproof there are also a slot on each cover that has a o-ring that gives a tight seal between the inside of the housing and the covers.

6.1.4 Underwater Lighting

To mimic the setup from Argus described in section 3.1.3 on page 12, some cheaper LED based lights were bought. The lights are two casings which delivers 27 W. Each casing has 9 LEDS, each rated to 3 W. The lights are shown in figure 6.4 on page 40.

The lights are originally intended to be mounted on a ships hull to light up the surrounding waters of the boat. This means that the lights are designed for mounting on top of a flat surface, which makes things easier for us. It is also delivered with a 12 VDC driver which means that it can be connected to any 12 V source.

Figure 6.1: EM15710 connected to the camera. Blue flat cable on the top is LVDS from camera. Gray flat flex to the left contains analog HD, communication and power to the camera. Multicolor molex in the bottom middle contains the analog video signals and level translated communication signals. Molex on the top is power. Gold MCX/BNC is HD-SDI out. Status led on the top left corner is red during signal locking and green during normal operation.

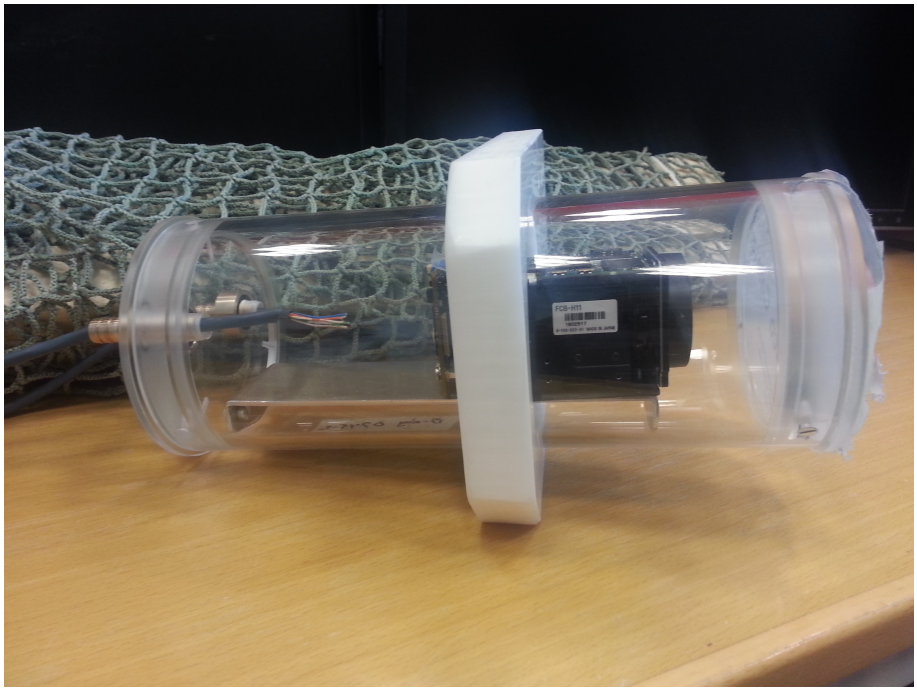


Figure 6.2: Underwater housing designed by Terje Haugen at the mechanical workshop at the institute with camera seated on the holding bracket. Nipples for the water tight cable connection using a normal 75 Ω coaxial cable for the video and a Ethernet CAT-5E twisted pair cable for power and serial data.

6.1.5 Cabling

The cabling used in the test are divided between the video stream and the other cable that provides power and communication with the camera. The video stream was transferred using a off-the-shelf 75 Ω coaxial cable with standard BNC connectors attached to it. This connects between the HD-SDI interface and the Ultrastudio SDI unit as depicted in figure 6.5 on page 41.

The secondary cable used was a CAT5E Ethernet Twisted Pair cable. As this is a cable made from eight leads that are twisted into four pairs, this had enough leads to transport both power and communication. Two of the pairs were used to carry the power to the camera. The last four leads were connected to the RX, TX and GND lines on a USB-RS232 adapter. This meant that the camera could be controlled over a USB port which is easier on a modern laptop.

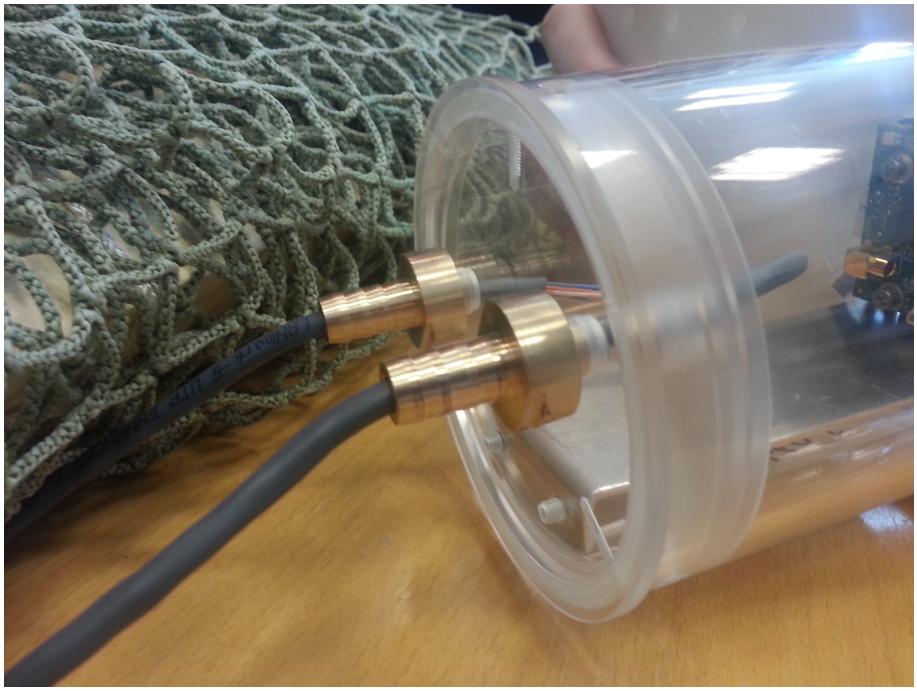


Figure 6.3: Same housing as shown in figure 6.2 but showing the nipples for the waterproof cable entries.

HD-SDI Interface	CAT5E Cable	RS232 Adapter
Pin 1, RXD (Brown)	Green/White	Pin 3, TXD
Pin 2, TXD (Black)	Blue	Pin 2, RXD
Pin 5, GND (Yellow)	Green	Pin 5, GND
Power 1, 9 V ± 3 V (Red)	Orange & Orange/White	
Power 2 & 3, GND	Brown & Brown/White	

Table 6.1: Signal cable connections for communication and power

Figure 6.4: Underwater LED lights mounted on a plywood backing plate.

6.1.6 Hardware Schematic

Figure 6.5 on the next page describes how the different parts of the system interconnects. The figure includes the Ethernet interface in addition to the HD-SDI interface. The different domains of the system are also shown. This is marked as Local, Transmission and Remote.

Remote in this context means all the components that are either connected directly to the camera or needs to be within a short distance of the camera. In figure 6.5 this includes the two interface cards used. Transmission is the part containing the entire distance between the camera and the receiving computer. Local points are the parts of the systems that are in close vicinity of the receiving computer.

Figure 6.5 also shows all the different signals and directions of the signals. This in addition to table 6.1 on the preceding page should give a fairly complete picture of how the components are interconnected.

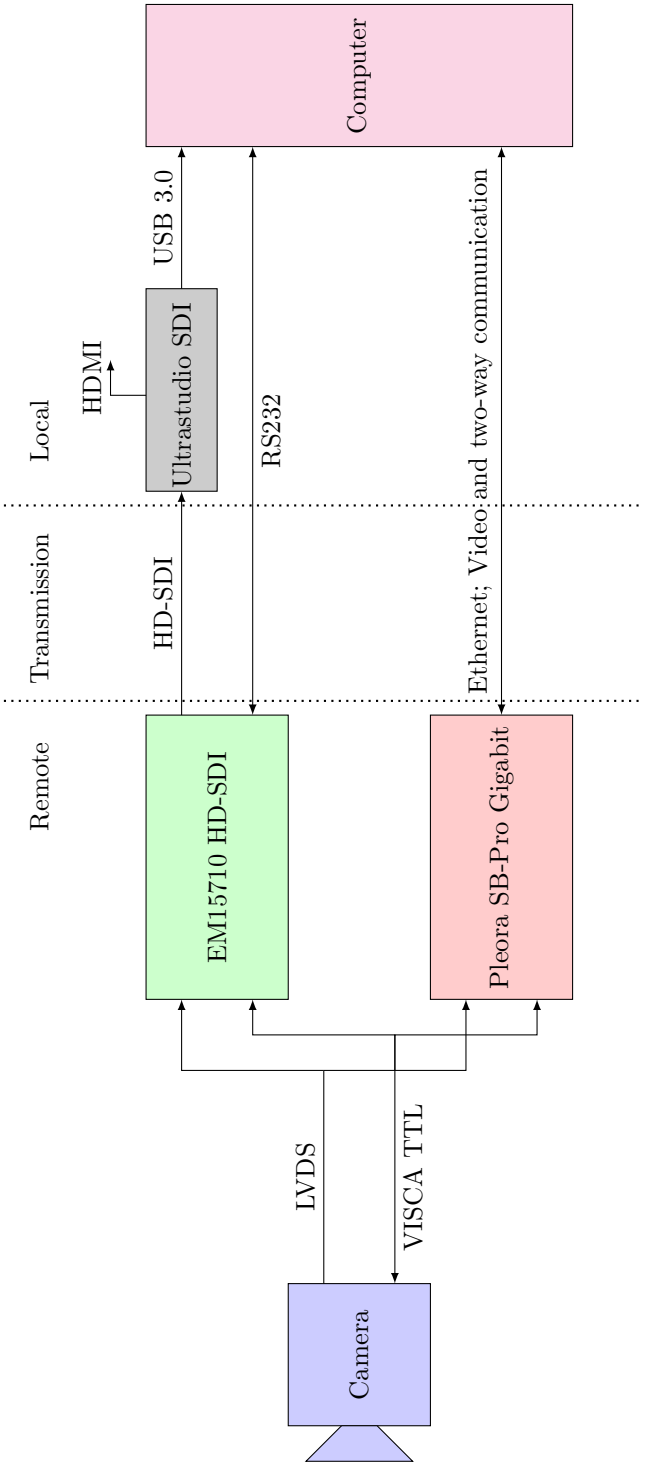


Figure 6.5: Schematics of the hardware, showing both the HD-SDI and the Ethernet pipeline. The different signals and directions are indicated. Vertical lines express different domains in the system; Remote components on the camera, Transmission for the place where the long distance signals are sent and Local for what is near the computer and the receiving end.

6.1.7 Hardware Setup

Due to the problems experienced during the initial setup and design of the different components in the system, a step by step list is provided with the normal response of the system.

1. Install the Ultrastudio drivers and software. The drivers are installed as part of the Blackmagic Media Express software.

2. Connect all cables according to figure 6.5. Note that the figure describes two different component paths. For HD-SDI, the following cables should be connected;

   a) Ensure that the LVDS cable and the flat flex cable is connected between the camera and the HD-SDI interface board.

   b) BNC connector from camera should be connected to the "HD-SDI In" port on the Ultrastudio unit.

   c) USB 3.0 connector from the Ultrastudio unit should be connected to a computer.

   d) Power should be connected to the correct leads in the Ethernet cable. Due to losses in the cable, 12 V should be applied.

   e) Optional: Connect the Receive Data Pin (RXD), Transmit Data Pin (TXD) and Ground Pin (GND) leads from the Ethernet cable to a serial port on the computer. This is used for operation control of the camera.

3. After power has been applied, the camera module will give some clicking noises. This is the opening and closing of the iris and the initialization of the zoom. By looking into the lens during power up, the inner lenses can be seen moving.

4. After the camera has finished, the status led on the HD-SDI interface board on the back of the camera should be glowing red. If it is green, then the Ultrastudio unit has already initiated the video stream. This should usually not interfere with continued use.

5. Check that the small white led beside the USB 3.0 connector on the Ultrastudio unit is glowing white. If it is not, then unplug the unit and try another USB port.

6. Open the Media Express software. The Ultrastudio unit should automatically get selected under the Device menu.

7. Change view in the Media Express software from Playback to Capture.

8. The Status led should now turn green, indicating that the Ultrastudio unit has initiated the video stream. The stream from the camera should now appear in the Media Express software, and can be captured.

9. Optional: If the camera is controlled over serial;

   a) Install and open the attached `FCBH11` program provided by Sony.

   b) Choose the correct serial port and 38400 baud.

   c) The main window of the program should appear. The rightmost box should contain name, firmware version and other information about the camera. If the rightmost box is empty, try clicking refresh to update the information. The send and receive command buffers can be seen below the information box.

   d) When the connection is active, try using the Zoom functionality. The send command buffer should fill up with the Zoom steps. The receive buffer should contain the acknowledgement packets sent from the camera. More on the packets and what can be sent over VISCA can be found in Sony [22].

A point regarding the above list, is that the use if the Blackmagic Media Express software is only necessary during initial setup. Any software that registers as a capture sink for the Ultrastudio unit should cause the unit to initiate the video stream. As the Blackmagic drivers registers with Windows Driver Model, most software capable of streaming should be able to initiate this. Both the OpenCV library, as well as for instance VLC[1] proved able to do this.

The side effect of the fact that the Ultrastudio unit does not activate until it is requested by software is that the unit does not stream any video until such a request is made. The status led on the HD-SDI board will also indicate error until the stream is completely initiated, as the HD-SDI protocol does not activate until both the receiver and sender have agreed on the parameters of the stream. As the schematic in figure 6.5 indicates, a separate HDMI output is available from the Ultrastudio unit. This output does not activate until some software initiates the stream, and can be quite confusing.

To access the video stream without a computer, it is necessary to connect the screen or input device directly to the analogue video output. The output is available on breakout leads from the HD-SDI board. The header and leads for this is described in Intertest [13]. The analogue HD is always active, as these signals are provided directly from the camera.

[1] www.videolan.com/vlc

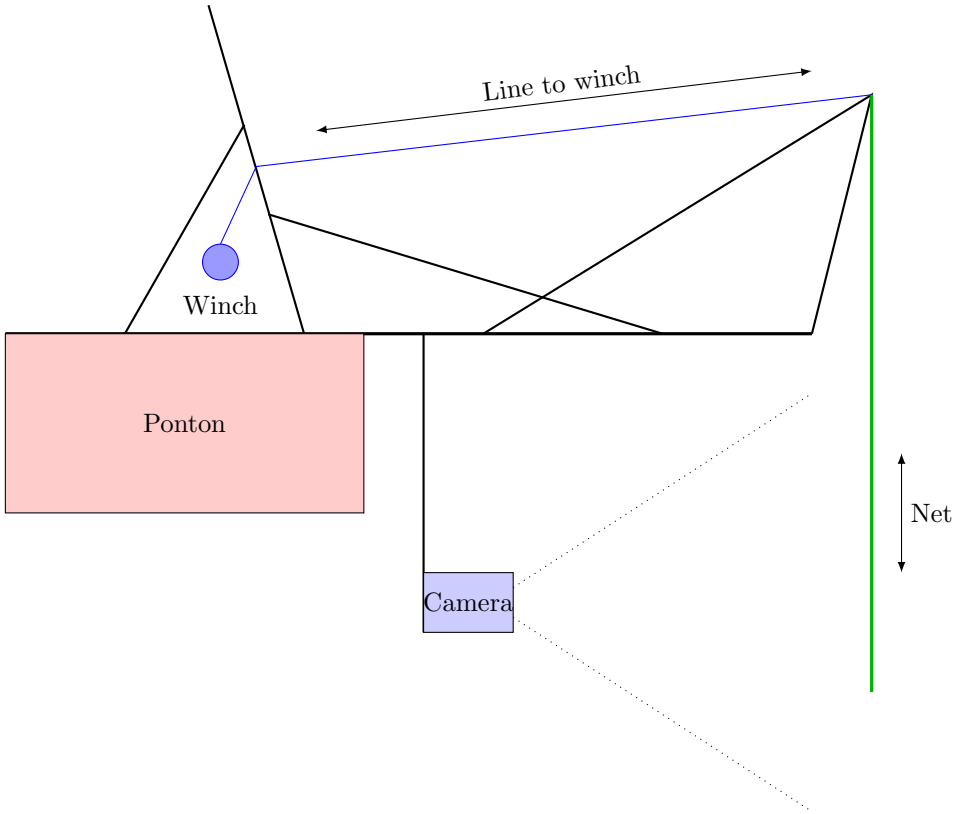


Figure 6.6: Illustration of the test rig from SINTEF. Black lines denote scaffolding, blue ropes and green the net. Counterweights are stacked on the back of the rig to keep it steady and horizontal. The camera is submerged and kept in place by scaffolding approximately 1.5 meters below the surface and 2 meters from the net.

6.1.8 Test Rig

The test rig is the same as the one shown in figure 7.2 on page 47. The rig consists of a scaffold that is extended over water. The rig has a winch that connects to the net and gives the power to lower and raise the net. The speed of the net movement can also be controlled to some degree using the winch. A schematic of the rig is shown in figure 6.6.

Chapter 7
Field Tests

Computer Vision algorithms can only give as good results as the source videostream it is being fed. After talking with SINTEF, some test videos were provided by them. However, as seen in figure 7.1 on the next page, the quality leaves much to be desired. The video provided had a resolution of 854×480 pixels with big black box padding around it. This does make the video unsuitable for use in a computer vision algorithm.

Due to the poor nature of the quality of the video it was decided early on that a field test was needed, since we were going to use equivalent hardware as to what is available on the ROV.

7.1 SINTEF DVL Test

As part of the cooperation with SINTEF during the preliminary testing, we were asked to help with a DVL test using a rig for controlling the movement of the net. As a favour for us helping out, we got to borrow the rig at a later time when our hardware was ready to do a field test.

During this test, we learned enough about the rig and operation of it to be able to operate it ourselves. The field test also gave some valuable information on how to get everything set up correctly, and what preparations would be needed for test.

7.2 HD Video Test

The rig was configured to mimic the approximate distance to the net in figure 7.1. We were however not able to tilt the camera to an angle due to environmental constraint from an anchor chain situated below the camera. We approximated the distance between the ROV and the net in figure 7.1 on the following page to be 1.5 m. This lead to the image in figure 7.5 on page 49.

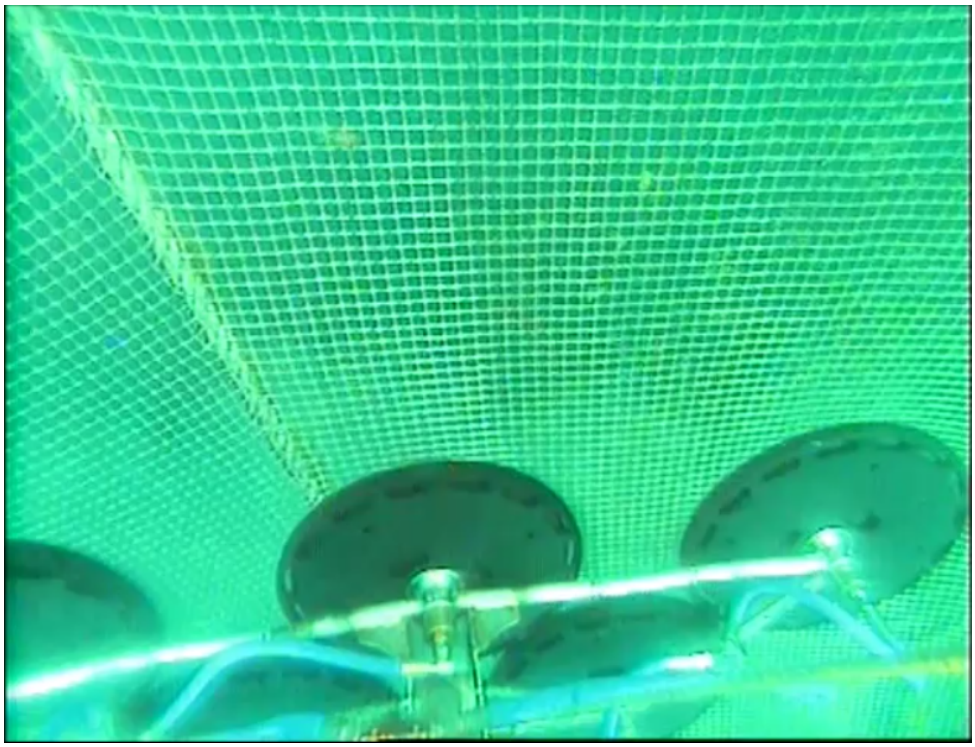


Figure 7.1: Original video provided by SINTEF.

Net setup	Description
Reference	Net in static position. No forced movement.
Reference with movement	Net lowered in front of the camera.
Circular hole	Net with a circular cut-out lowered.
Vertical tear	Net with a narrow vertical tear lowered.
Horizontal tear	Net with narrow horizontal tear lowered.
L-shaped tear	Net with horizontal L-shaped tear lowered.
Growth	Net with imitation growths lowered.
Double sea-cage	Two nets laid on top of each other.

Table 7.1: Test setup, all at 1.5 m

Figure 7.2: Field test for SINTEF using DVL.

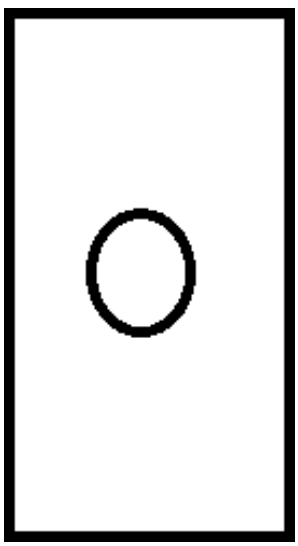
Comparing image 7.5 on page 49 and 7.1 on the preceding page, it seems that the top row of masks is approximately the same in both images. Due to the tilt of the camera in image 7.1 we get a prominent vanishing point in that image. There are also some growth on the net in figure 7.1, which for obvious reasons does not appear in figure 7.5.

The test was done in accordance with the test patterns described in figure 7.3 on the following page. Nets with different holes were lowered in front of the camera, left hanging for some time, and then lifted up again. In addition to the different nets with holes, one net without holes was also used as a clean reference.

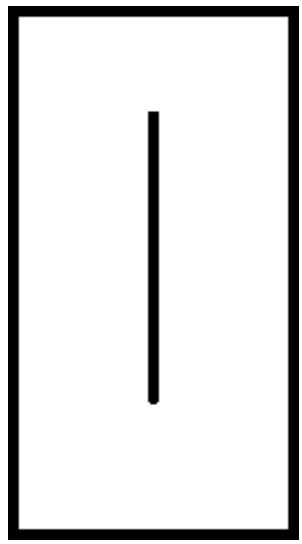
7.3 Video Stream and Quality

The video collected during the test showed off the superb quality of the camera that was acquired. The camera deals with the low light conditions and the clear light gradient in the image. A still image of the test of the net that contained growth can be seen in figure 7.7 on page 50.

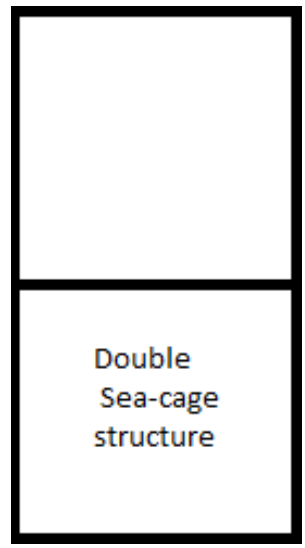
The image in figure 7.7 also shows some of the issues with the underwater conditions. The lower right part of the image has a very different light and contrast ratio, as a



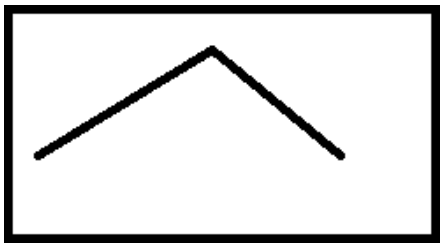
(a) Circular Hole.



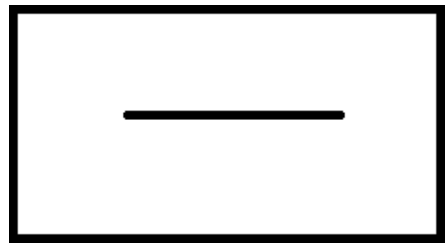
(b) Vertical tear.



(c) Double net.



(d) L-shaped tear.



(e) Horizontal tear.

Figure 7.3: Different net configurations with holes and tears, excluding normal single net. Images collected from Sletta-Heimdal [19].

halo from the sun can be seen. This adds complexity to the algorithms that are to be used, and can in worst case lead to a total loss of data points.

The zoom level on the camera was also tested. A normal camera will struggle to keep the same colour level during zoom, as the light and colour changes during zoom. As seen in figure 7.8 on page 51, the software in the camera seems to deal with this in an acceptable way. Figure 7.8 shows the edge of the growth region with full zoom. The details in the image is difficult to appreciate when it is shown as still images. For the full effect, the reader should have a look through the video `reference_growth.mp4` that is attached to this report.

Another problem with the video stream that was discovered at a later stage, is

Figure 7.4: Field test on a sunny day in May.

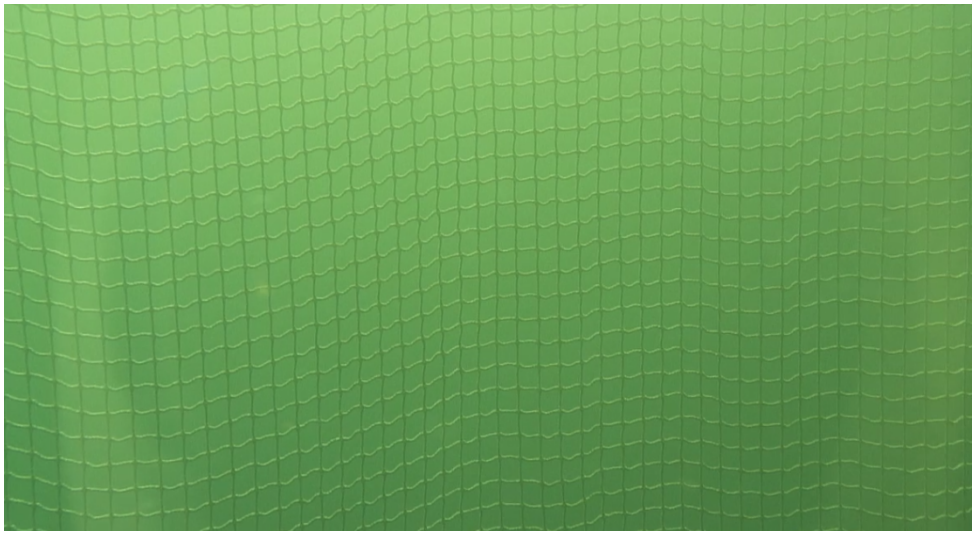


Figure 7.5: View of the net from 1.5 m. This is the default distance for the rig used.

Figure 7.6: View of masks in the net at 1.5 m. Digitally zoomed in post processing.

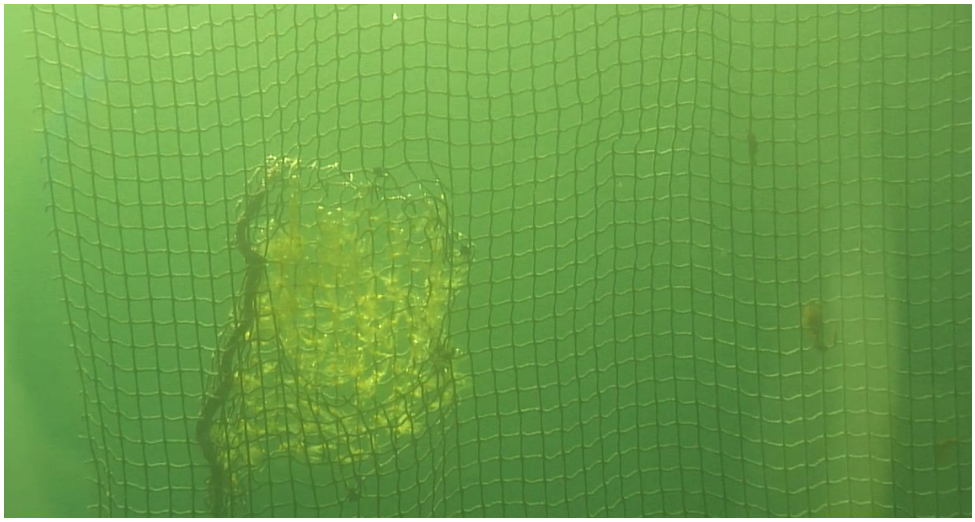


Figure 7.7: View of masks in and growth simulation on net at 1.5 m.

Figure 7.8: Same view as in figure 7.7, but using optical zoom.

how the camera behaves during rapid movement. Due to the perceived low light conditions, it seems that the camera increases the shutter speed to allow more light into the camera sensor.

The shutter speed of the camera during the underwater operation seems to have been 1/50 s which caused motion blurring to be introduced. This causes the masks in the net to be blurred out, resembling that of a Gaussian filter. Back in the office, it was discovered that the camera can lock the shutter in different modes, allowing manual control of the shutter. The camera is capable of speeds ranging from 1/2 s to 1/10 000 s in 21 steps. The movement artefacts can be seen in figure 7.9.

7.4 Software

The software used during the field tests was mainly the software provided with the Ultrastudio unit. This capture software provided a stable connection to the Ultrastudio unit. It also made it possible to capture all the sequences that were done as raw video. The raw video could then be compressed at a later stage to ensure that the quality during tests were as close to the raw stream as possible.

7.4.1 Capture Constraints

During the initial tests, it was discovered that the throughput of the capture was far greater than that the laptop was able to save to disk. This meant that the capture

was done to RAM, and pushed to the disk at a later stage. The size restriction of the RAM meant that the length of each sequence needed to be quite short, as the laptop that was used would exhaust its memory in approximately 30 seconds of capture. This caused frames to be skipped in some of the captured files.

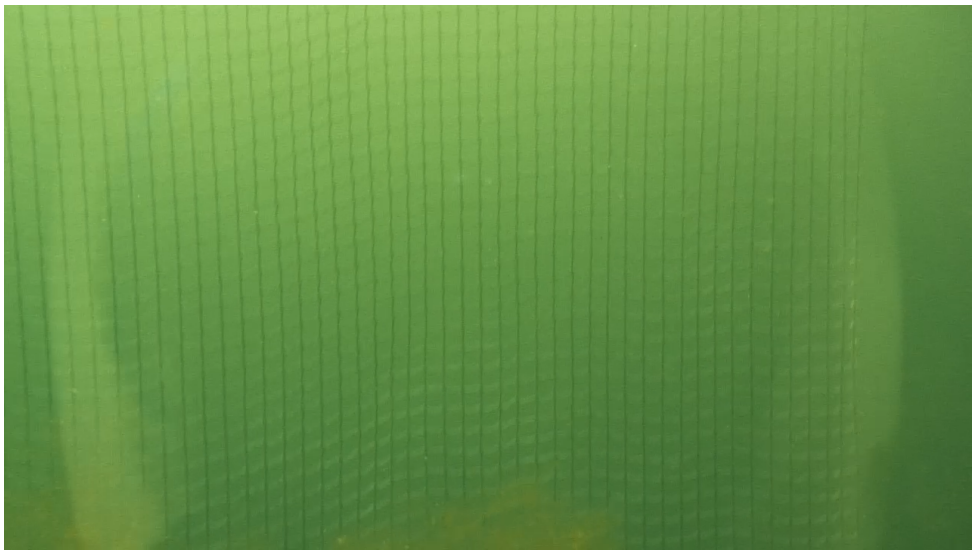


Figure 7.9: The raw video during rapid movement.

Chapter 8
Application of Algorithms

In this chapter the main algorithms that were covered in chapter 5 are going to be examined in a more practical fashion. The algorithms will be prototyped against the navigation scenario in a net.

As mentioned in section 3.2.2 on page 15, all implementations done uses the open source OpenCV library. This is done through the Python bindings made available by the project for prototyping, and through the C++ API to see how the runtime of the algorithms compare. By design, Python will be slower than the C++ versions as Python uses an interpreted and JIT based system for running a program[1]. This means that the Python runtime needs to perform Marshalling which gives a negative speed impact.

The main video used in this chapter is `reference_move.mp4` which is attached to this thesis. This video was a capture of a single net being first lowered and then raised in front of the camera. The result of this video should be a movement pattern that is mainly in the vertical direction, with some minor movement in the horizontal direction. The movement in the horizontal direction is mainly caused by motions in the water moving the net back and forth.

8.1 Optical Flow Algorithms

The video that were mainly used to compare the different algorithms below where the video `reference_move.mp4`. Three other videos are also attached, showing different kinds of disturbances. These videos were also used to test if the algorithm skewed if noise and foreign objects were introduced.

As the movement of the net were not precisely defined during the test, the result of the different algorithms does not contain any specific error measurement. However,

[1] The runtime

by visual inspection of the video side by side with the tracking output gives some rough estimates.

8.1.1 Flow Fields

Lucas-Kanade is the main algorithm that tries to identify optical flow by modelling the difference between two frames as a flow field. This algorithm is described in section 5.4.3 on page 30.

The version of Lucas-Kanade that was implemented is based on the pyramid version of the algorithm. This makes the algorithm less computational expensive. The pyramid is built up based on the `cv::goodFeaturesToTrack` function in the OpenCV library. This function uses a Canny edge detector to detect features in a frame. The Canny edge detector is thoroughly discussed in Sonka et al. [20] and Canny [4]. This makes use of the fact that edge intersections normally does not deform between two frames. Features that are traceable will by that argument be in the concave section of two or more intersecting Canny lines.

The initialization of the feature points yields the image in figure 8.1. The red circles shows where the Canny detector has found strong enough line intersections.

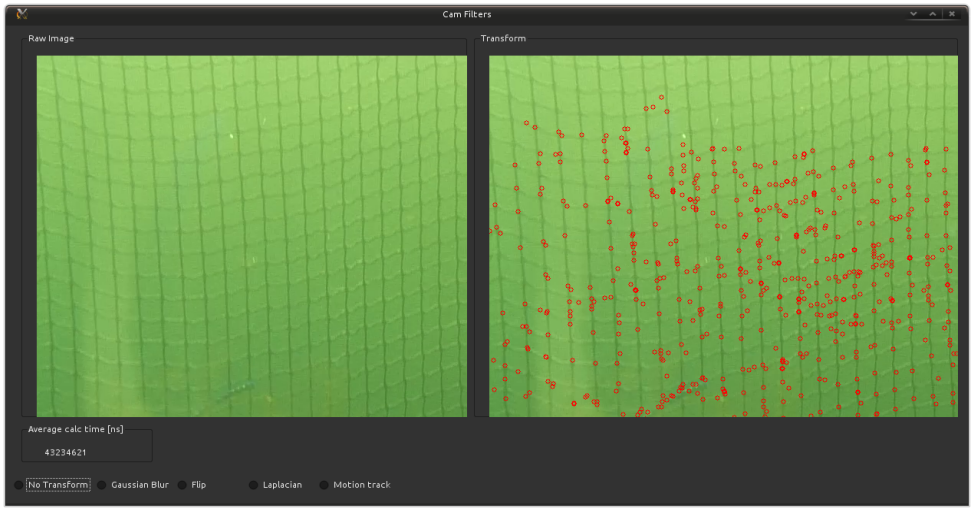


Figure 8.1: Initialization point of Lucas-Kanade. Red dots show the tracking result from the Canny Edge Detector.

The problem with finding good enough features becomes evident after some time. Figure 8.2 on the facing page shows the tracking result after 1 s. Since the features need to be constant to be tracked, no new features are found as masks in the net

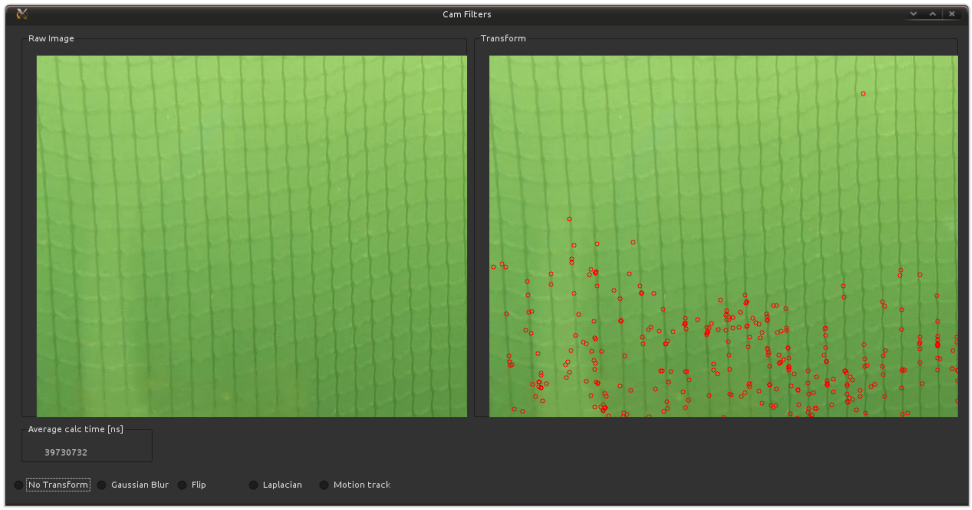


Figure 8.2: 1 second of tracking using the Lucas-Kanade algorithm with the initialization from 8.1.

move off the screen. This is mainly rooted in the normal application of algorithms such as this. Normally, the algorithm would be used to track either a single object or some discrete number of objects in a stream of frames.

Applying the Lucas-Kanade algorithm however to the net scenario yields bad results. One reason is that the track points start to slip on the features that were chosen to be tracked. The reason for this is the relative uniformity of the environment, which is not something that the Lucas-Kanade algorithm was designed to track. This is rooted in the velocity smoothness constraint mentioned in section 5.4.3 on page 30

8.1.2 Energy Fields

Horn-Schunck

The main algorithm that uses energy fields to model the flow between two frames is the Horn-Schunck algorithm described in section 5.4.4 on page 31. This algorithm has been more or less replaced by the Farnebäck [9] algorithm, as described in section 5.4.5.

On this background, the Horn-Schunck algorithm was not implemented or tested. This is mainly because of the statistical data presented by Farnebäck [9] showing that the Farnebäck algorithm never gives a worse result than the Horn-Schunck algorithm, while proving to be less computationally expensive.

Farnebäck

Farnebäck became the algorithm with a base in the theory of energy fields that was implemented. This is a quite a new algorithm in the field of optical flow, but it has proven itself to be a robust and attractive algorithm. The implementation is based on Farnebäck [9] by using the OpenCV library.

The image in figure 8.3 shows the field calculated by the algorithm during linear translation in a realistic scenario. This means that even though the majority of movement is found to og in the downward direction, there are some horizontal movement as well. A close-up view can be seen in figure 8.4.

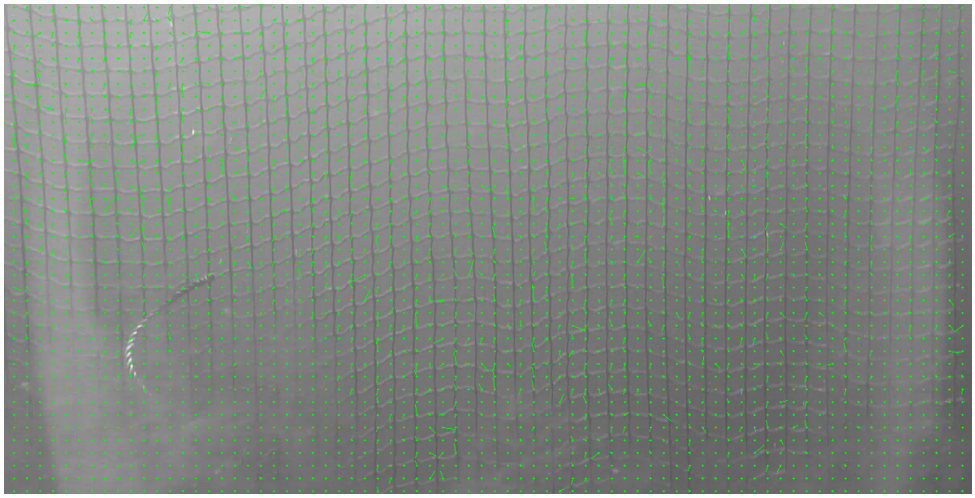


Figure 8.3: Using Farnebäck [9] to track movement when net is lowered. Each point is the middled motion found in each sub-image as described in section 5.4.5.

Since this algorithm estimates the field in a grid, it can be sensitive to weak references and smoothing. An example of real life conditions that may lead to a loss of the field in areas with low contrast is shown in figure 8.5.

Based on how the algorithm calculates the flow, it is also obvious that the Farnebäck algorithm is susceptible to other foreground objects that will distort the measurement of the flow for the net. This is based in the local grid that this algorithm uses, and it therefore does not use any global measurement to detect the mean flow. These features can be seen in figure 8.6 and 8.7.

The hard distortions seen in figure 8.6 and 8.7 stems from the intended use of the algorithm. It is optimized to find either global flow when the image only contains background, or to track objects in the foreground.

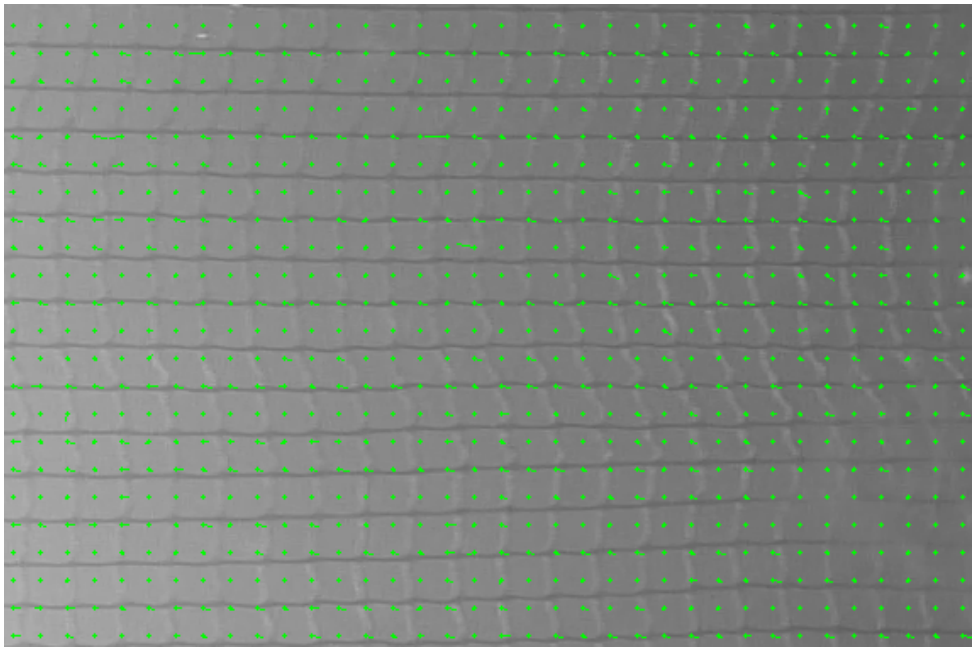


Figure 8.4: Closer look at the output from Farnebäck [9] on a subset of masks. Almost uniform flow is detected in the points where the net movement happens. Up direction is to the left in the image.

The Farnebäck algorithm was tested on a sample video. The movement detected by the algorithm has been plotted in figure 8.8 on page 60. Note that the dimensions for the movement are related by some scaling factor to the real movement in the image stream.

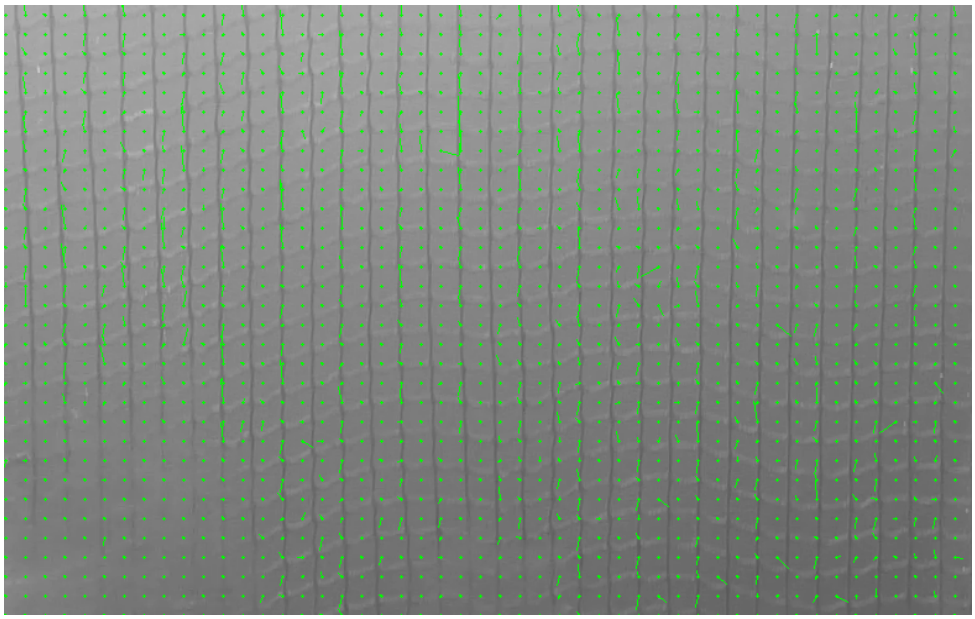


Figure 8.5: Overview of tracking. Note good tracking in points that contains the vertical parts of the masks and the loss of tracking in the lover left corner.

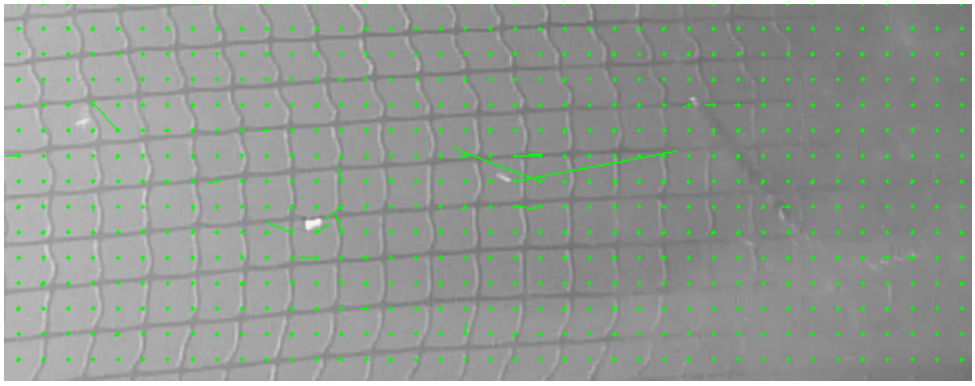


Figure 8.6: Showing Farnebäck [9] tracking other unwanted objects in the stream. Here some small air bubble are being tracked. Upwards direction is to the left in the image.

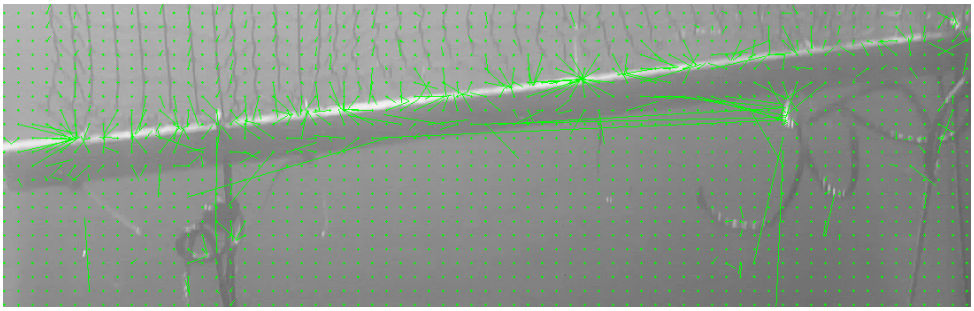
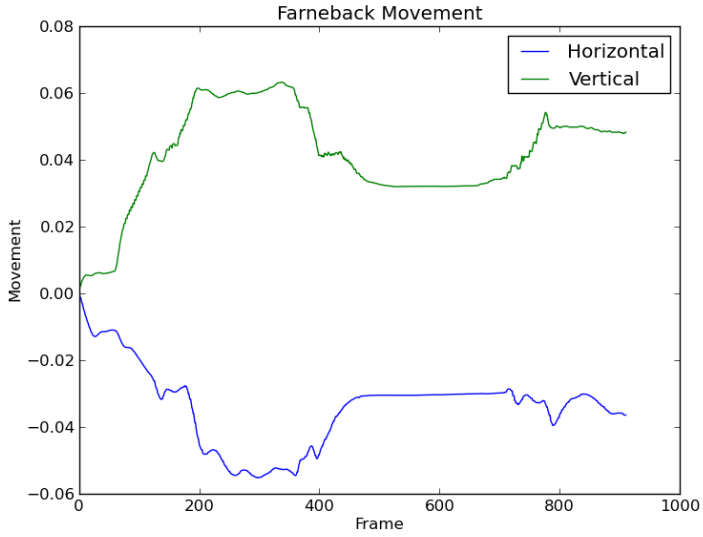
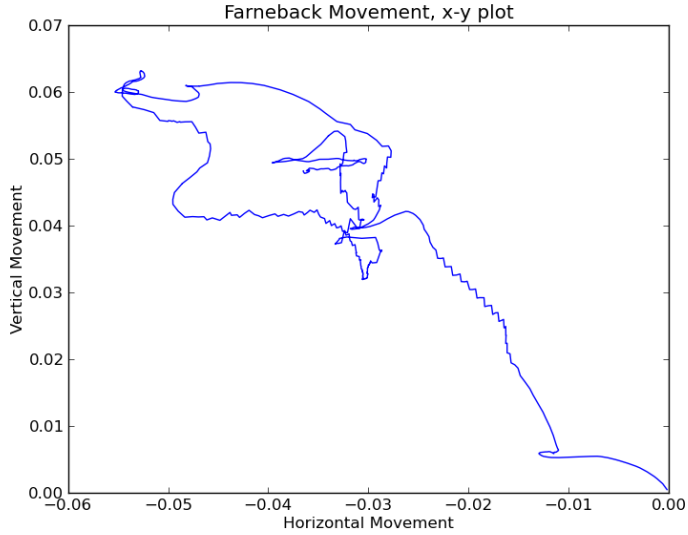


Figure 8.7: Showing Farnebäck [9] tracking the bottom line of the rig. Multiple big distortions.



(a) Result of the Farnebäck algorithm detected movement on a sample video.



(b) Result of the Farnebäck algorithm detected movement on a sample video, horizontal position plotted against vertical position.

Figure 8.8: Using the Farnebäck algorithm on a sample video. Shown as position vs frame (8.13a) and x-position vs y-position (8.13b).

8.1.3 Line Tracing

The use of Hough Transforms was tested by Carlsen [5] to detect the net and masks in the net. His trials were initially done in a static scenario with the net held in place by a metal frame. He showed that it was possible to find the lines describing the mask structure using the normal Hough Transform by setting strict parameters on the minimum distance between detected lines under those conditions. It proved however to be much worse to get stable readings during his field test, and

The image in figure 8.9 shows the results of the normal Hough Transform on the net suspended in free water. Due to the movement of the net, and the folding seen in figure 8.9 the transform fails to find a consistent number of lines. When the net starts moving up as shown in figure 8.10 are seems as most of the lines that were found in figure 8.9 is being tracked correctly. However, as the Hough Transform generates a new set of lines for each image this is not an assumption that generally holds.

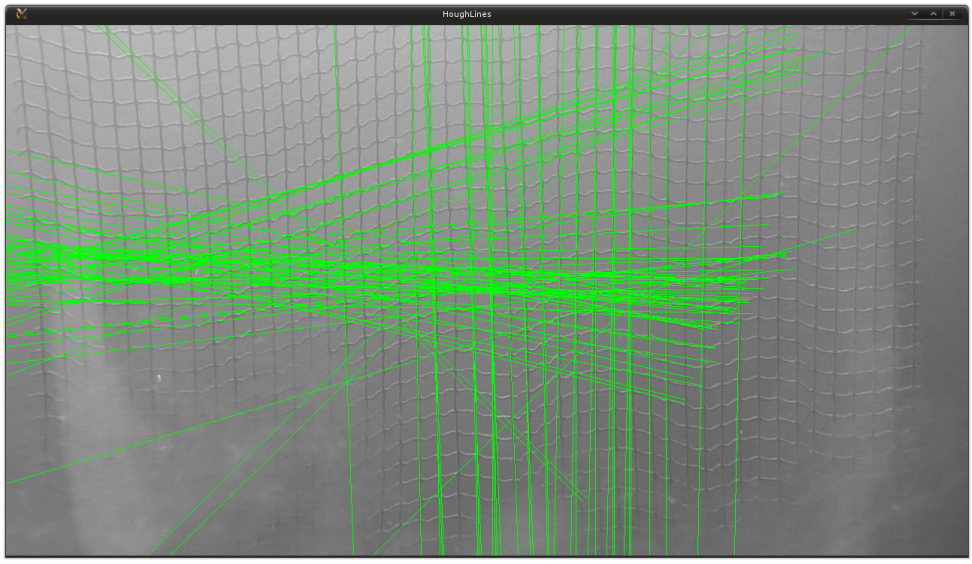


Figure 8.9: Initial result of the Hough Line Transform on a net at rest.

According to section 5.2.2 on page 26 the Probabilistic Hough transform would be a better choice in the environment shown in figure 8.9. This comes from the design of the algorithm to not assume that lines are straight for the duration of the lines. Using the probabilistic properties from section 5.2.2 the image in figure 8.11 on page 63 were obtained. This is from the same sequence as figure 8.9. The image in figure 8.12

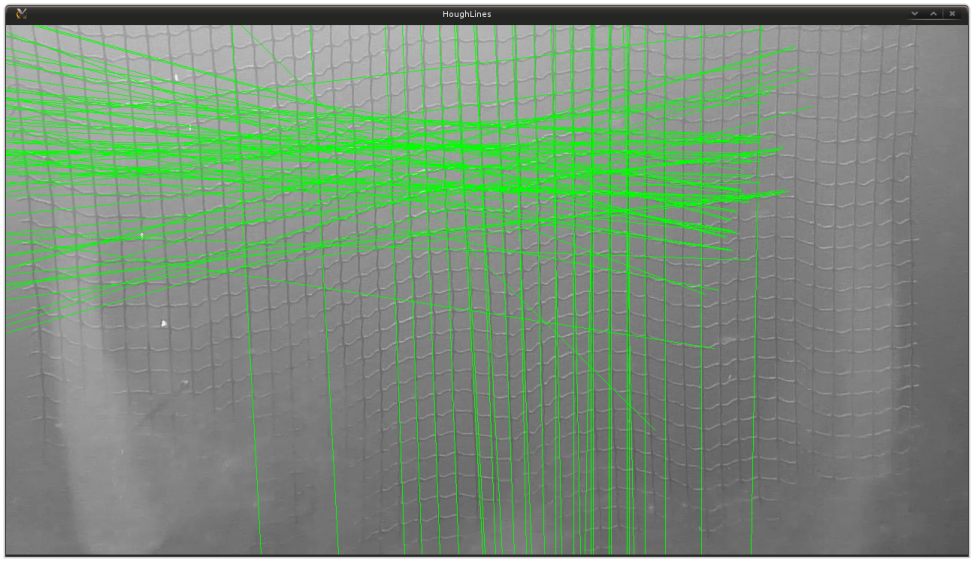


Figure 8.10: Result of the Hough Line Transform when the net is moving.

is from the same sequence as figure 8.10, and shows how the probabilistic transform detects lines during movement.

As seen in all the different figures in this section, line tracing often gives a bad or inconclusive result. This is caused by the deformation of the net, causing the nice square masks to deform. This makes consistent straight lines hard to identify, leading to errors in the line.

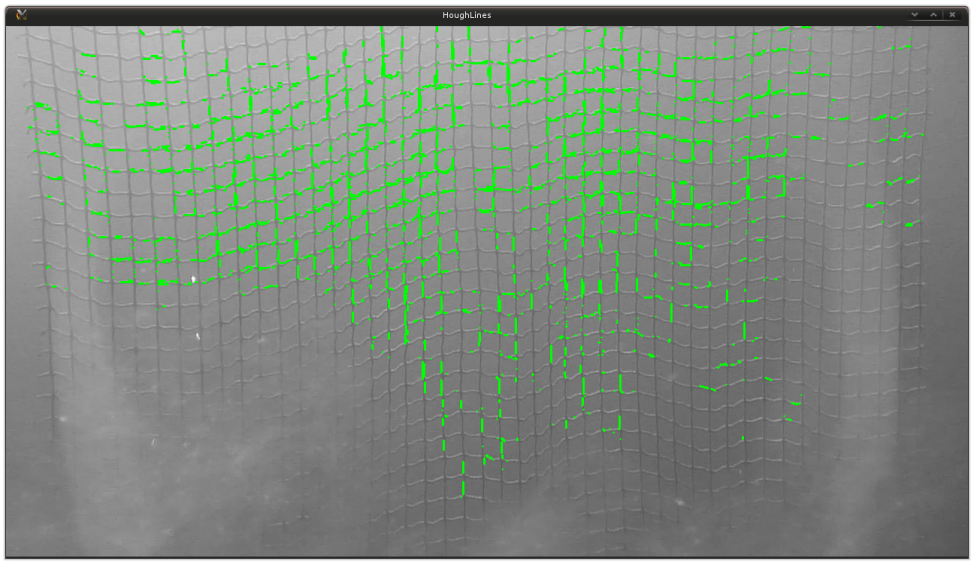


Figure 8.11: Initial result of the Probabilistic Hough Line Transform on a net at rest.

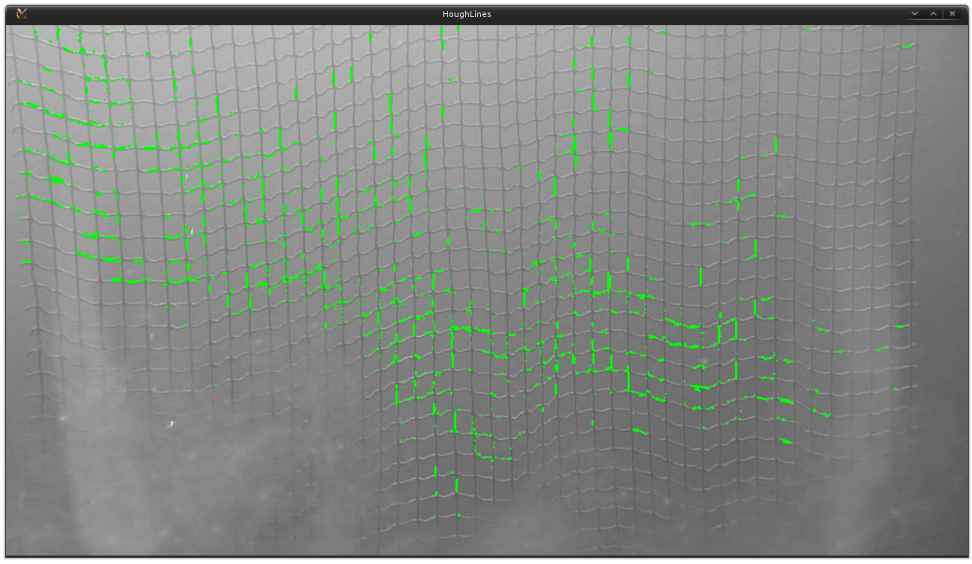


Figure 8.12: Result of the Probabilistic Hough Line Transform when the net is moving.

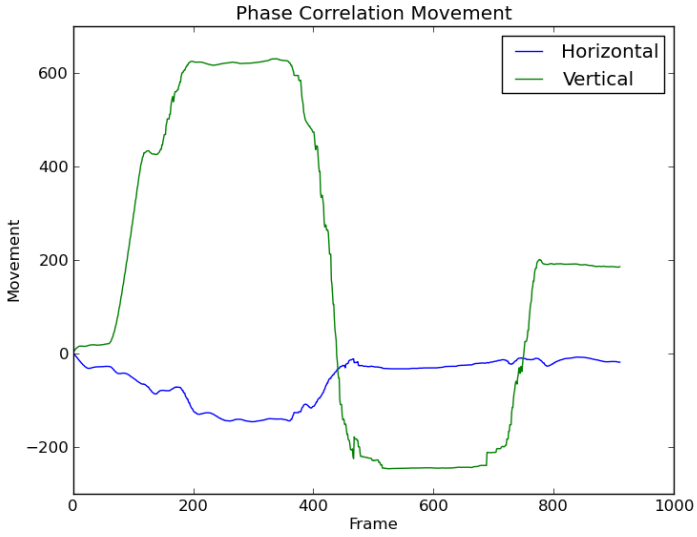
8.1.4 Phase Correlation

The Phase Correlation algorithm that was described in section 5.4.2 on page 28 was also tested. This is one of the older methods of finding flow, and it is mainly designed to find the linear translation between two frames. The use of a phase plane makes the algorithm extremely resilient to noise in the image. This is in sharp contradiction to other algorithms that work in the spatial domain.

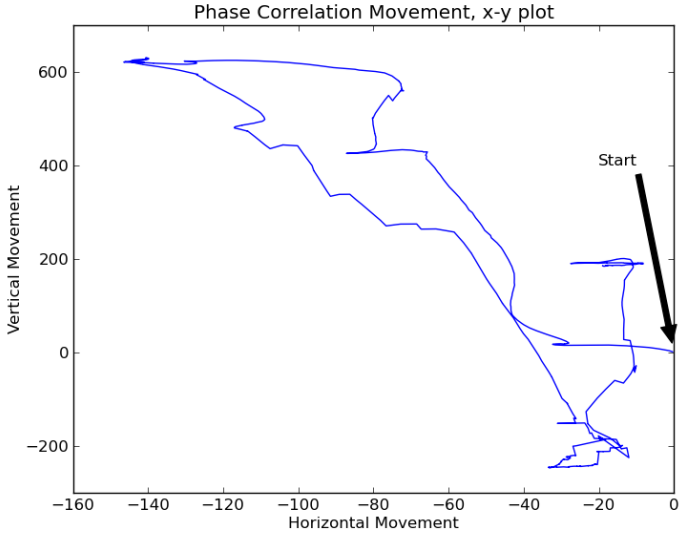
The output from the phase correlation algorithm is the primary detected movement between two frames, and some statistical data about the detected movement. This means that it is easily plotted as shown in figure 8.13.

The figures in figure 8.13 shows the same video sequence plotted in two different ways. Figure 8.13a shows the two different dimensions of the video and how it is detected to move during the full duration of the stream. This gives a temporal view of how the movement progresses. Figure 8.13b shows a polar plot of the same, where the horizontal and vertical components This plot loses the temporal component, but it shows a trace of the movement of the net.

After the testing in the field were done, some more controlled tests were also done. This consisted of simply filming some nets that were attached to a wall. The camera were moved in a simple up-down pattern. The images in figure 8.14 shows the result of phase correlation on one of these sequences. Most of the movement in the horizontal direction seen in figure 8.14a comes from the fact that the camera was moved by hand. The figures does show quite a good tracking of the vertical movement.

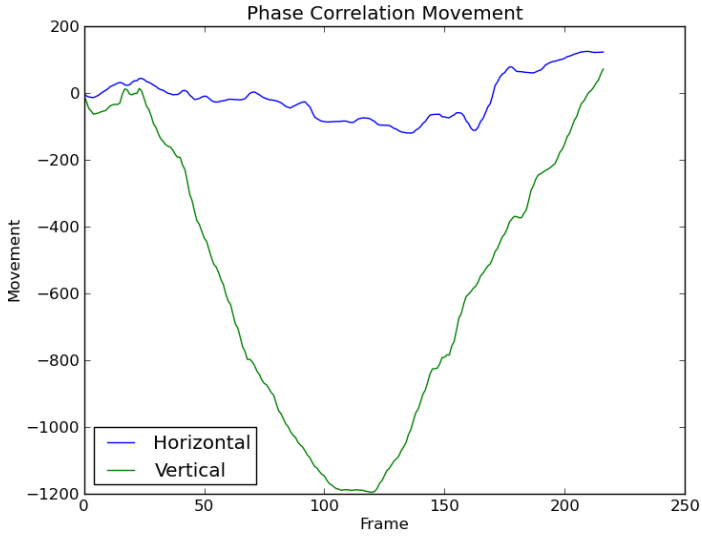


(a) Result of Phase Correlation detected movement on a sample video.

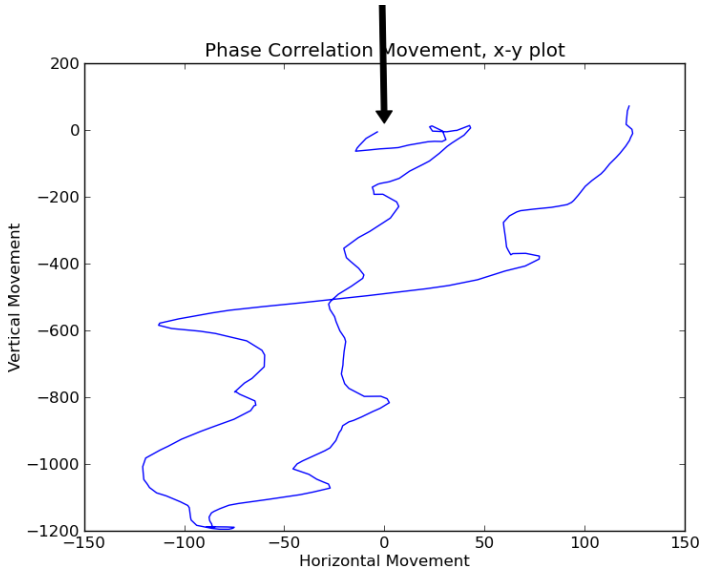


(b) Result of Phase Correlation detected movement on a sample video, horizontal position plotted against vertical position.

Figure 8.13: Using Phase Correlation on a sample video. Shown as position vs frame (8.13a) and x-position vs y-position (8.13b).



(a) Result of Phase Correlation detected movement on a sample video.



(b) Result of Phase Correlation detected movement on a sample video, horizontal position plotted against vertical position.

Figure 8.14: Using Phase Correlation on up_down.ogv. Video of simple up and down movement. Shown as position vs frame (8.13a) and x-position vs y-position (8.13b).

Chapter 9
Discussion

During this thesis, multiple motion tracking algorithms have been implemented and tested. In the field of computer vision, most motion tracking algorithms can be classified into a couple of different classes based on the class of equations on which the algorithms are based.

9.1 Algorithms

The two main classes of motion tracking algorithm were implemented and tested. The issue with the algorithms are the scenario that they are developed for. Most applications of motion tracking today tries to detect moving objects that are relatively small compared to the background in the video. The problem in this thesis however is mainly centred around tracking the background. This causes the popular algorithms to prefer distortions and objects in front of the net.

Looking back, it turned out that one of the easiest and most basic algorithm was the best in tracking the net. This algorithm is called the Phase Correlation algorithm, and is based as the name implies on phase correlation. This algorithm is covered in detail in section 5.4.2 on page 28. This algorithms has one property that the newer and more popular algorithms avoids; Globality. Other algorithms that tries to track objects needs to be local in scope, so that multiple objects can be tracked. Globality is however a very good property when tracking one big object.

Having the global property means that smaller objects will have minor impact on the output of the tracking algorithm, as long as the net is the main object visible in the image. Phase correlation is one of the tools used in digital signal processing. The fact that phase correlation is a mature technique means that many optimizations has been done to the equations to make implementations faster.

There are some issues with the Phase Correlation algorithm. The largest issue is that it struggles during rotation. This shortcoming is of an algorithmic nature. It is

however not likely that big rotations should be a problem in the sea cage.

Another issue with all the algorithms presented is the need for calibration. The Phase Correlation algorithm for instance gives the detected translation in pixels. This number then needs to be transformed into a translation. The problem with this is that the pixel to translation transformation is dependent on the distance between the camera and the net and the zoom level of the camera. The laser system that were initially thought of would be able to detect this distance by counting the area of masks inside the laser. As the effort with lasers were halted due to engineering costs, some other mean of calculating the distance between the camera and the net is needed.

The issue of calibration could also be solved by detecting the masks in the net, and using a-priori knowledge about the masks dimension. This could however lead to errors, as pointed out in Sletta-Heimdal [19]. Detecting the masks are not always easy, as the algorithms needs to find the spine of the ropes in the mask. This number is not something that is easily measured, as a mask dimension usually is the width and breadth of the open part of the mask. As growth accumulates on the masks, this measurement degrades.

In Carlsen [5], Hough Lines were used to detect the masks in the net. The trials carried out in that report was done with the net stretched out on a wire frame. This made the Hough Lines correspond quite nicely to the masks in the net. During the tests done in this thesis, the same procedure was found to give bad results. This could come from many environmental influences, but the main issue is that the masks are not square when they are free-hanging in the water. They will deform, and the net itself can start to buckle and warp. This makes line finding very difficult to do in a coherent fashion.

9.2 Hardware

As noted by Sletta-Heimdal [19], quite a bit of time in his and this thesis was used to get the capture hardware operational. This included both deciding on and ordering hardware and building underwater housing. Getting the hardware operation proved also difficult at first, as the equipment were new and unknown. The HD-SDI interface proved particularly difficult to get operational. This mainly stems from the behaviour of HD-SDI, where no data is transmitted until end-to-end communication has been established. This makes debugging and the useful ”Divide and Conquer” method for narrowing the error source difficult.

9.3 Quality

When all the components finally started to work, the image quality produced was impressive. The quality turned however quickly to a double edged sword. The high resolution video meant that the fine details of the masks in the net were visible, even at a distance. The computational complexity of the algorithms on the other side, meant that online tracking is not feasible using the current setup. Different means to address this is covered in section 11.2.

After the underwater tests were done, it was noticed that the videos contained quite a bit of motion blurring, causing rapid movement to become unclear. This was found to be largely a result from the low light condition and the shutter speed used by the automatics in the camera. It was unfortunate that this was discovered after the filed test, as there are no way to fix this after capture.

Chapter 10
Conclusion

As most of the normative theory in the field of optical flow points out; "[...] tracking large, monotone or repetitive objects in the background is hard", from Schellewald [18]. This surfaces in most of the popular algorithms used to do motion tracking today. The used algorithms are tuned and tailored to detect objects that are relatively small compared to the area in a video frame.

The fact that most algorithms tested in this thesis is tuned to track objects makes the problem somewhat difficult to solve. Even with considerable fiddling with different parameters, the tracking data given by these algorithms works sporadically at best. The favouring of foreground objects also leads to big noise spikes when some distortion or object passes between the camera and the net.

Using one of the older approaches to detect motion was at an early stage in the thesis discarded as being to crude and not functional. This was concluded based on test data provided at the time.

When newer and better data were acquired during this thesis, it turned out that the better images collected just lead to an increase in the noise reported by the modern algorithms. As a last resort, the old algorithm using phase correlation were tested. The result from this talks for itself, and can be seen in figure 8.13 on page 65.

A algorithm based on the phase correlation principle is what ended as the better of the algorithms tested. The algorithm does not have a particularly bad response with normal noise, and is designed to find the translation between two images, which is exactly the main point of this thesis.

During this thesis, performance has been a recurring issue. The problem stems from the fact that most of the motion tracking algorithms are quite slow and complex. This is usually not a big problem when normal foreground objects are being tracked, as any of the tracking objects needs to be quite distinguishable for the tracker to

work. This means that even if a algorithm loses track of the object, it can be found again either by interpolating its position based on previously known velocity and direction, or by a reinitialization of the tracker.

Another issue is the conversion between the output from the algorithm and to a real-world equivalent. This needs some sort of calibration to be done. The calibration needs to be robust, as there are environmental changes during operation of the ROV.

Due to the repetitiveness of the net, the sampling time is critical to avoid aliasing. The movement of the net between each video frame needs to be less than half of the width of the masks in the net. This means that in stead of analysing the whole net it might be easier to carry out the analysis on a smaller area of the net, where it is known not be anything obstructing the view to the net.

In the end, it seems that using the phase correlation principle gives quite good data when it has a good view of the net and the net does not move to quickly. To quickly is in this context the speed which the tracking software is able to read in and process images. If the time delay is to great, then the algorithm will start to misbehave. The limits of this issue will need further investigation before a system can be realized.

Chapter 11
Further Work

This chapter aims to provide some pointers to guide the further development of the issues covered by this thesis.

11.1 Algorithms

Improvements and other clarifications related to the algorithms used are mentioned in the sections below.

11.1.1 Contour Based Tracking

Using contour based tracking has not been mentioned as a class of motion tracking in this thesis. This is mainly because of the common use cases for this method. Using contours for motion tracking is dependent on having clear contours in the image. This will most likely not be the case when considering the massive repetitiveness of the masks in the net.

The environmental challenges of detecting motion in a repetitive structure was the cause why this method was omitted. The assumptions for this may however not be valid, and should be checked.

11.2 Computational Complexity and Optimizations

It turned out early that the video stream outputted by the camera was much better than quality than any other video that we were given access to. The steep increase in the video size made both online algorithms difficult to implement and caused a big strain on the capture hardware. The sections below deal with different means to reduce the complexity of the problem.

Peripheral Vision

Currently, the calculation of movement between two frames takes to long. Calculating the movement does not keep ut with the default sampling rate of the camera, causing frames to be skipped. This can cause aliasing if the movement between two full frames are to long.

One way of reducing the amount of data is to use peripheral vision. This is an approach inspired by nature, where most animals uses peripheral vision to process the wanted information from the environment and discarding unwanted information.

Peripheral vision gives the perception of higher details in a region centred about the direct line of sight. The vision outside this region is naturally blurred, and does not contain much information in itself. This region outside the detailed can give information on movement and objects, but no real detail. This causes the computational complexity to decrease in these areas.

Peripheral vision inspires the method of pyramids, as described in section 11.2.1 below.

11.2.1 Pyramids

There are a broad use of pyramids in image processing today. The construction of pyramids in the sense of image processing can be seen in figure 11.1.

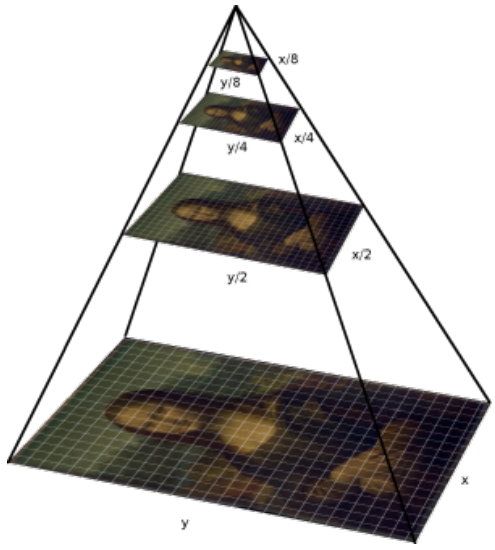


Figure 11.1: Pyramid view of an image. Image collected from http://iipimage.sourceforge.net/documentation/images/.

The use of pyramids allows for the use of different resolutions at different points in an image. This can lead to parts of an image to be discarded as uninteresting, lowering the resolution in parts of the image.

Algorithms that is to be used on pyramid images needs to be aware of the pyramid, so that it can use the parts of the image with lower resolutions to be processed more quickly. Another issue that can arise is the fact that the pyramid usually will enhance the edges between parts of an image when the different levels are combined.

Using pyramids in our problem to reduce the complexity is a path that should be investigated.

11.2.2 Parallel Computation

Being able to divide a frame into smaller chunks could also make parallelization easier, as the computational units are clearly divided in this scenario.

One of the issues that was encountered in the different implementations of vision algorithms is that they seldom give any indication on the possibility to parallelize the algorithm. The OpenCV library used also takes a conservative approach to this issue. This means that the prototypes developed were mostly single threaded.

A substantial speed-up should be possible if pyramids are used. Pyramids are briefly discussed in section 11.2.1. Using pyramids gives some clear borders between regions of interest in the image. If this region is consistent across the two frames where motion is to be calculated, then the different sub-images could be processed individually. The potential of this should be further examined.

11.3 Quantifying Bounds of Aliasing and Velocity

Aliasing is a recurring issue in this thesis. Aliasing in this context comes from the problems that arise when the net moves fast or the camera samples slowly. Due to the repetitiveness of the surface, this can cause the different algorithms to detect motion opposite the direction of movement.

The variables that is involved in this is the velocity of the net, the speed that the images are sampled which may differ form frames per second delivered from the camera and the size of each mask in the net.

The software needs to be aware of the issue of aliasing. This can either be done purely in software by using different rules to avoid sampling at a speed that causes aliasing, or some empirical data that describes when aliasing can appear can be used. Other external measurements as accelerometers may also give some information if the

direction of travel is different that the direction returned by the vision algorithm. The quantification of when aliasing can occur is important to keep the vision algorithm robust.

11.4 Distance to the Sea Cage Wall

As most of the algorithms in computer vision gives a relative translation, this number needs to be transformed into a real translation. The calculation of this either dependent on the distance between the camera and the net or on the known dimension of the masks in the net. The distance between the net and camera is going to change during operation, making the pre calculated conversion factor between pixel and metre wrong. Using the known width of the mask is also troublesome, as described in Sletta-Heimdal [19]. This is both due to difficulties detecting the edges of the mask, and also as the edges collects growths, making the dimensions non-uniform. This is paramount to sort out if the data from any vision algorithm is going to be implemented into a system.

11.5 Laser Based Orientation

The use of lasers to help with the orientation were mostly abandoned throughout this thesis, The main reason for this was the lack of appropriate hardware. Based on the experienced behaviour of free-hanging nets in the water, it would seem that a circular laser would give easy access to data on the deformation of the net, and the tilt that the camera has relative to the net.

As Jo Arve Alfredsen pointed out, projecting for instance a triangle onto the net using three line lasers would possibly yield the same result. The difficulty would then be to apply thresholding and to detect the projected lines and to recreate the geometrical figure. How robust, easy and useful this information would be is also something that will need further investigation.

References

[1] EE368/CS232 digital image processing. Lecture Foils. http://www.stanford.edu/class/ee368/.

[2] Akvaplan-niva. Faktaark 2 transport og løft av not, 2005.

[3] Argus. Argus rover observation/light work class rov, 2013. URL www.argus-rs.no.

[4] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986. ISSN 0162-8828. doi: 10.1109/TPAMI.1986.4767851. URL http://dx.doi.org/10.1109/TPAMI.1986.4767851.

[5] A. Carlsen. Navigational assistance for mini-rov. Master's thesis, Norwegian University of Science and Technology, 2010.

[6] E. R. Davies. *Machine vision : theory, algorithms, practicalities.* Morgan Kaufmann, 3rd edition, 2005.

[7] Steve Eddins. Steve on image processing. http://www.blogs.matchworks.com/steve/2006/10/04/separable-convolution, October 2006.

[8] FAO. The state of world fisheries and aquaculture. Technical report, Fisheries and Aquaculture Departement, Food and Agriculture Organization of the Inited Nations, 2006. http://www.fao.org/docrep/009/A0699e/A0699e00.htm.

[9] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, LNCS 2749, pages 363–370, Gothenburg, Sweden, June-July 2003.

[10] Andrew Fitzgibbon and Robert B. Fisher. A buyer's guide to conic fitting. In *In British Machine Vision Conference*, pages 513–522, 1995.

[11] James Jerome Gibson. *The perception of the visual world.* Houghton Mifflin, 1950.

[12] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow, 1980. URL http://hdl.handle.net/1721.1/6337.

[13] Intertest. ishot fcb hd-sdi interface module, 2011. URL www.intertest.com.

[14] Intertest. G. ishot xblock gige ip network interface board kit for sony fcb-h11 block camera, 2011. URL www.intertest.com.

[15] Ø. Jensen, T. Dempster, I. Uglem, and A. Fredheim. Escapes of fishes from norwegian sea-cage aquaculture: causes, consequences and prevention. Technical report, Aquaculture Environment Interactions, August 2010. http://preventescape.eu/wp-content/downloads/2010_aei_jensen_et_al.pdf.

[16] Hassan K. Khalil. *Nonlinear Systems*. Prentice Hall, 3rd edition, 2001.

[17] J.G. Proakis and D.G. Manolakis. *Digital signal processing*. Pearson Prentice Hall, 2007. ISBN 9780131873742.

[18] Christian Schellewald. TDT4265 computer vision. Lecture Foils.

[19] Morten Sletta-Heimdal. Machine vision for real time detection of net damage in sea cages. Master's thesis, Norwegian University of Science and Technology, 2013.

[20] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2007.

[21] Sony. Fcb-h11 high definition color block camera, 2008. URL http://pro.sony.com/bbsc/ssr/product-FCBH11/.

[22] Sony. *Sony HD Color Camera Module Technical Manual A-D3N-100-11(1)*, 2008.

[23] Boris Worm, Edward B. Barbier, Nicola Beaumont, J. Emmett Duffy, Carl Folke, Benjamin S. Halpern, Jeremy B. C. Jackson, Heike K. Lotze1, Fiorenza Micheli, Stephen R. Palumbi, Enric Sala, Kimberley A. Selkoe, John J. Stachowicz, and Reg Watson. Impacts of biodiversity loss on ocean ecosystem services. *Science*, 314(5800):787–790, November 2006.

[24] Arash Yazdanbakhsh and Simone Gori. Mathematical analysis of the accordion grating illusion: A differential geometry approach to introduce the 3d aperture problem. *Neural Networks*, 24(10):1093 – 1101, 2011. ISSN 0893-6080. doi: 10.1016/j.neunet.2011.06.016. URL http://www.sciencedirect.com/science/article/pii/S0893608011001766.

Appendix A
Content of the Attachments

The attached files are split into a couple of different folders. The full archive has been uploaded to CloudStor[1] with subject `id=8833` in accordance with the instructions given in DAIM[2].

A.1 libVISCA

This folder contains a ported version of the libVISCA library that can be used to control the camera used in this thesis. It is written in C, and its source is included.

A.2 Prototypes

This folder contains the different prototypes of algorithms implemented in this thesis. it consists of either Python or C++ source files.

A.2.1 Python

The python files requires the OpenCV python module, the `numpy` module and the `matplotlib` module. These modules can be installed under a Debian related Linux distribution by issuing the following command. The package names are mostly similar for RPM based systems.

```
apt-get install python python-{matplotlib,opencv,numpy}
```

A.2.2 Qt

To compile the C++ programs, both Qt[3] and OpenCV is needed. Windows users can download QtCreator at http://qt-project.org/downloads which includes MinGW.

[1] https://filesender.uninett.no/
[2] https://daim.idi.ntnu.no
[3] http://qt-project.org/resources/getting_started

The OpenCV library is available from http://opencv.org. The QtCreator project file needs to be updated with the library path to OpenCV.

For Linux users, the command below will install QtCreator, OpenCV and all dependencies. The project file is also configured to use `pkg-config` for library and include paths.

```
apt-get install qtcreator libopencv-dev
```

A.3 Videos

The videos folder contains the videos used in this thesis. The rest of the videos collected during this thesis are available in a subfolder.

A.4 FCBH11.zip

This archive contains a sample program provided by Sony for communicating with the camera over the VISCA protocol. It gives access to most VISCA commands in a GUI, and it also gives logging output so that the raw VISCA commands can be seen. This made the understanding of the VISCA protocol easier, as it was possible to see the different packets, commands and responses.