NTNU
Det skapende universitet

Maskinsyn-basert inspeksjon av notintegritet i havbruksanlegg

Morten Sølvberg Sletta

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

Norges teknisk-naturvitenskapelige universitet
Institutt for teknisk kybernetikk

Project description

NTNU

Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk

Institutt for teknisk kybernetikk

MASTEROPPGAVE

Kandidatens navn:	Morten Sletta-Heimdal
Fag:	Teknisk kybernetikk
Oppgavens tittel:	Maskinsyn for sanntids deteksjon av skader i oppdrettsnot
Engelsk tittel:	Machine vision for real time detection of net damage in sea cages

Oppgavens tekst

Skader på noten i lakseoppdrettsanlegg medfører stor risiko for rømming, et problem som er assosiert med alvorlige økonomiske og miljømessige konsekvenser. I dag brukes dykkere til inspeksjon av noten, men denne aktiviteten kan være både risikofylt, tidkrevende og kostbar. ROV benyttes nå i økende grad som et alternativ til dykkerinspeksjon. Bakgrunnen for denne oppgaven er et ønske om å effektivisere ROV-operatørens arbeid med å inspisere noten og på sikt få automatisert denne arbeidskrevende og monotone operasjonen. Oppgaven tar sikte på å få utviklet robuste algoritmer med utgangspunkt i bildestrømmen fra ROVens kamera (eventuelt eget kamera og belysning) for sanntids deteksjon av skader og svakheter i noten. Oppgaven omfatter følgende punkter:

- Gjennomgang av problemstillingens bakgrunn, identifisering av forskjellige skadescenarier og utarbeidelse av kravspesifikasjon for deteksjonssystemet
- 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 bildeanalyse av notpaneler og deteksjon av typiske tilfeller av notskader og -svakheter
- Planlegging og gjennomføring av tester av algoritmen(e) med hensyn til treffsikkerhet og robusthet, både under ideelle og realistiske betingelser
- Diskusjon av resultater og kartlegging av systemets muligheter og begrensninger

Oppgaven gitt: 16. januar 2013

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

Trondheim, 16. januar 2013

Jo Arve Alfredsen

Faglærer

Acknowledgments

I had different people helping me make this a Mater thesis I am proud of. First I would like to thank my supervisors, Associate Professor Jo Arve Alfredsen at the Department of Engineering Cybernetics and Per Rundtorp from SINTEF, Fiskeri og Havbruk for giving me this interesting thesis. Especially Jo Arves help and support, and comforting words throughout this semester have been to more help than he might know.

Special thanks goes out to Terje Haugen at the cybernetics departments mechanical workshop for making the underwater camera housing, to Trond Viggo Mellum, NRK for all the help in getting the hardware to work and to Kevin Frank from SINTEF Fiskeri og Havbruk for letting us lend his test-rig. Without these three the field test may not have been a reality.

I would like to thank Rolf Hansen from Malvik Båtforening for letting us perform the field tests in Malvik småbåthavn. I would also like to thank my father Ole Sletta for the on-site help given during the field test.

During this thesis I had the pleasure of working alongside another graduate at the Department of Engineering Cybernetics, namely Morten Engelhardt Olsen. His help with getting the hardware and performing the field tests were invaluable. His knowledge of programming was a big help when trying to develop a real software program for the first time.

Last but not least I would like to thank Torild and Johannes Sletta-Heimdal, my fiancée and newborn child for their love and support. And also for letting me sleep at night ☺

Abstract

When performing damage detection on sea cage structures one usually either use divers or watch the video feed from an ROV. The video frames obtained from the ROV mounted cameras were thought to give too poor image quality to be used damage detection, (Taylor & Kelly, 2010) . However with the on-set of low cost, high quality digital cameras this may no longer be the case.

During this thesis the different damage scenarios that can happen have been looked at, and different damage types have been tested. Factors like lights, turbidity and camera technology have been tested and debated.

Several computer programs that can detect damage and check the level of growths that is present on the net was developed and the hardware needed to perform field tests were acquired. Of the programs made the one named System 2 is the best for testing different algorithms and setting while the final system is the latest version and as such has the best overall performance on the test videos. In the final system there is almost no option to be made; this is more of a finished system that a typical user can run.

The viability of using the ROVs camera system for automated damage detection is debated and indications to where further research can be done in order to make an automated system robust enough to use in real life is stated.

Sammendrag

Når man ser etter skader på notstrukturene i oppdrettsanlegg brukes det enten dykkere eller man kan se på videostrømmen fra et kamera montert på en ROV. Man har ikke villet brukt videostrømmen fra en ROV for å lage et automatisk skadedeteksjonssytem før fordi kvaliteten til bildene ikke har vært gode nok. Men med utiklingen av veldige gode og billige digitale kameraer kan dette være en løsning med stort potensiale.

I løpet av denne diplomoppgaven har forskjellige skadescenarioer blitt kartlagt og det har blitt utført tester for å se hvordan de forskjellige skadetypene ter seg under vann. Faktorer som lys, turbiditet og kamera teknologi har blitt testet og evaluert.

Det ble utviklet flere data rpogrammer som kan detektere skader på not og se etter hvor mye vekster det er på notstrukturene. Utstyret man behøver for å få gode testvideoer ble også anskaffet. Av de programmene som ble utviklet er det som blir kalt System 2 det som er best å bruke hvis man vil teste forskjellige fargerom og segmenteringsalgoritmer og endre parameterne til disse underveis i testene. Det siste systemet som ble utviklet er tenkt for å brukes av en sluttbruker og har således få valg. Man har mulightene til å velge program modus og velge hvor i prosessen man vil se bildet fra.

Det blir debatert om man kan bruke kamera systemet som sitter på ROVer idag til å utføre automatisk skadedeteksjon og forslag til videre forskning er presentert.

Contents

List of figures

List of tables

List of equations

Nomenclature

Additive color space – A color space where a color is described by the amount of the three primaries needed to combine make that color.

CIE –	International Commission on Illumination
GUI –	Graphical user interface
IDE –	Integrated development environment
Smolt –	Juvenile fish, in this stage the salmon changes to be able to survive going from fresh water to salt water

1. Introduction

1.1. Background and motivation

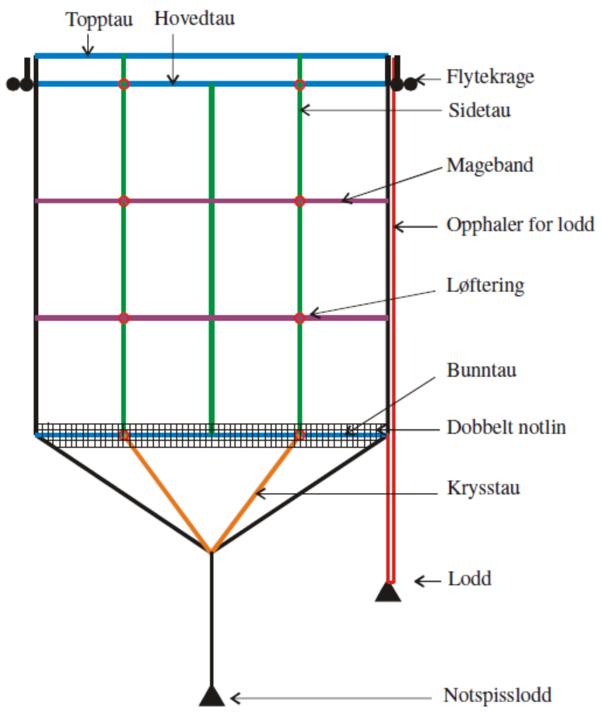


Figure 1: Principal schematic sea cage (Akvaplan-niva AS, 2005)

Fish escaping from the aquaculture industry is a big economic and environmental problem. According to (Jensen, Dempster, Thorstad, Uglem, & Fredheim, 2010) 3.93 million salmon and 0.98 million rainbow trout escaped in the span 2001-2009 in Norway, while 1.05 million cod escaped in the span 2004-2009. This seems to be a lot, but as an example there was at any given time in 2009 approximately 310 million Atlantic salmon and rainbow trout in sea cages in Norway. It seems that cod is especially hard to prevent from escaping. While the salmon escapes mainly after structural failures of containment equipment the cod escapes more often through holes in the net. The cod also bites on the net and thereby makes holes from which it escapes.

In (Taylor & Kelly, 2010) escaped incidents in Scottish aquaculture are reviewed. In Scottish aquaculture the most common cause of escape incidents is due to holes in the net. Of the reported incidents from May 2002 to October 2009, 57% of the incidents were due to holes in the net, which led to a total of 700.359 escaped fish. Incidents caused by predation and chafing are the two most frequently occurring causes for holes in the net.

In the sea cage there can be tears due to local overload when raising the cage net. From (Akvaplan-niva AS, 2005) it is stated that tearing of the net during handling is a big source for escape incidents. Figure 2 shows how a tear may develop due to wrong handling is shown.

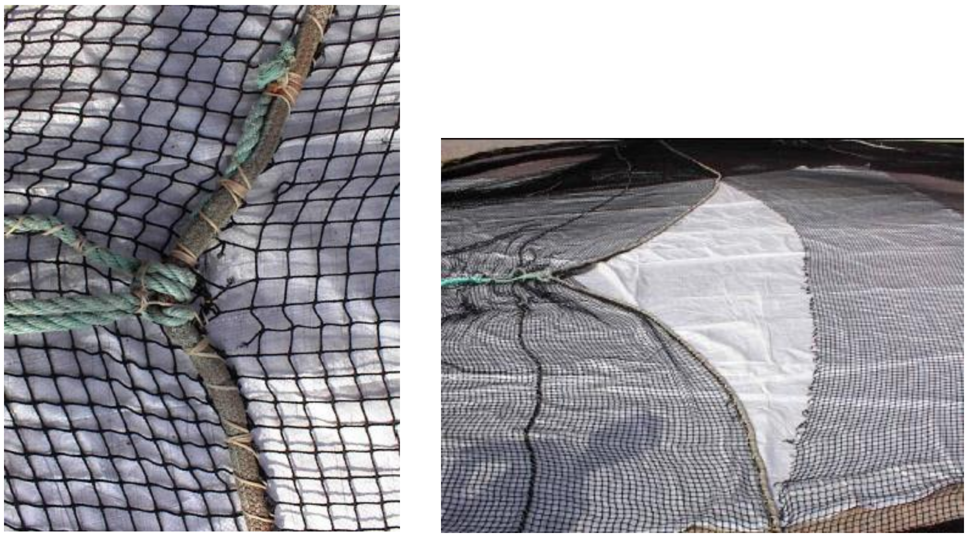


Figure 2: Development of a tear due to too much load on the side-rope (green) (Akvaplan-niva AS, 2005)

According to (Heide & Moe, 2004) the worst kind of tear is the L tear as shown in Figure 3. With a tear like this the fish have a high probability to find a way out of the sea cage. Horizontal tears can also make gaping holes that the fish can escape through while vertical tears don't make gaping holes so the fish is least likely to escape through a vertical tear. Because of the tension in the sea cage vertical tears tend to close up, this is good since the fish won't escape from them, however this also can make them harder to see.

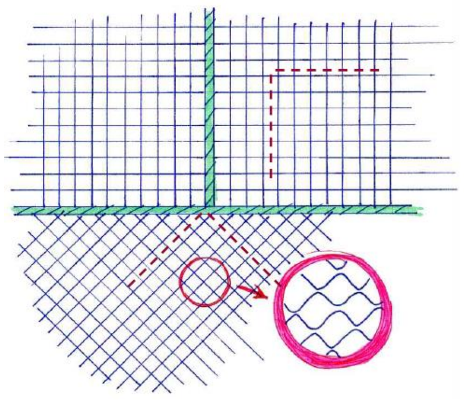


Figure 3: L-tear in sea cage (Heide & Moe, 2004)

Making a system to help detect structural damage to sea cages can have both economical end environmental benefits. Escaped fish is both a direct loss in form of money but there is also the problem of the farmed fish mating with the wild fish or spreads disease. Mating with wild salmon leads to the degeneration of the wild fish genetic material. According to (Rønning, 2011) and (Klima- og forurensningsdirektoratet, 2012) the problem is that the farmed salmon is bred based on criteria's such as the ability to grow fast, fat content, flesh color and disease resistance. The qualities needed to survive in the wild however are quite different and more varied. A salmon that has a genetic material closer to the farmed salmon will not be as resistant to changes in its environment. Also the smolt production have been found in controlled experiments to be 30% lower for escaped farmed salmon compared to wild salmon. If the wild salmon gets a decreased smolt production and weaker genetic material the damages to the wild salmon population may be irreversible.

Another problem that comes from escaped farmed fish is diseases, and in particular the *Lepeophteirus salmonis* lice. According to (Direktoratet for naturforvaltning, 2012), *Lepeophteirus salmonis* is a crustacean that attack salmon, sea trout and char. It is a severe threat to the wild salmon, and in some areas it threatens to eradicate the wild salmon and sea trout population. With the huge increase in overall fish numbers due to fish farming the lice population have also increased. The lice release its eggs into the water where the eggs develop into larvae that follow the current and attack wild salmon and sea trout. As few as 9 to 11 lice is considered deadly for the young salmon that comes from the rivers and are entering the sea. In the fish farming facilities the lice population is kept down with chemicals and wrasse. There is a concern that the lice are getting resistant to the chemicals that are used and the fish farms thereby losing their means to combat this threat.

To inspect the cage net divers can be used for visual inspection. Divers can also perform tasks such as removing dead fish, report on the fish condition and inspect moorings. The divers can identify net damage before the damage leads to escape incidents. Sea cages are rather big structures. They are either square cages with 20 to 40m sides which are 35 m deep, or circular cages with a circumference of 90-157m that are 15-48m deep. Total cage volumes range from 20.000 to 80.000 m^3 (Jensen, Dempster, Thorstad, Uglem, & Fredheim, 2010). Using motorized aids like a ROV or AUV is wanted because of this large size. ROVs are now used to inspect the sea cage as well as performing cleaning operations. An operator drives the ROV and watches a video feed from the ROV in order to inspect the sea cage. In the future there is hope that the ROV can automatically detect and repair holes. This is where this thesis comes in. Many different approaches are possible to do this detection. In (Taylor & Kelly, 2010) they recommend that research should be undertaken into devices that could detect holes in the net. However they stated that cameras have been found difficult to use for this task.

There are not only holes in the net that can be detected using computer vision. A major concern for fish farmers is growths on the net that disturbs the water flow through the sea cage. Seeing the progress of growths on the net can help the fish farmers to better control when net cleaning should be performed. One threat for the fish is a little hydroid named *Ectopleura larynx.* This little creature and how to combat it have been the focal point in (Guenther, Fitridge, & Misimi, 2011). According to this report biofouling is a serious problem for fish farmers. It has a negative impact on the water exchange across the net and the water quality within the cages. It also can lead to cage deformation and structural fatigue. To remove the organism the sea cages are cleaned with high pressure salt water, this method removes most of the hydroids.

1.1.1. Today's solution

Today divers or ROVs are used to inspect the sea cages. Using a ROV an operator needs to drive the ROV around the sea cage and inspect the video taken from the on-board camera on the ROV to see if there is any damage to the sea cage. In Figure 4 a frame from a ROV inspection video is shown. As of now the operator does not have any software to help with detecting damages.

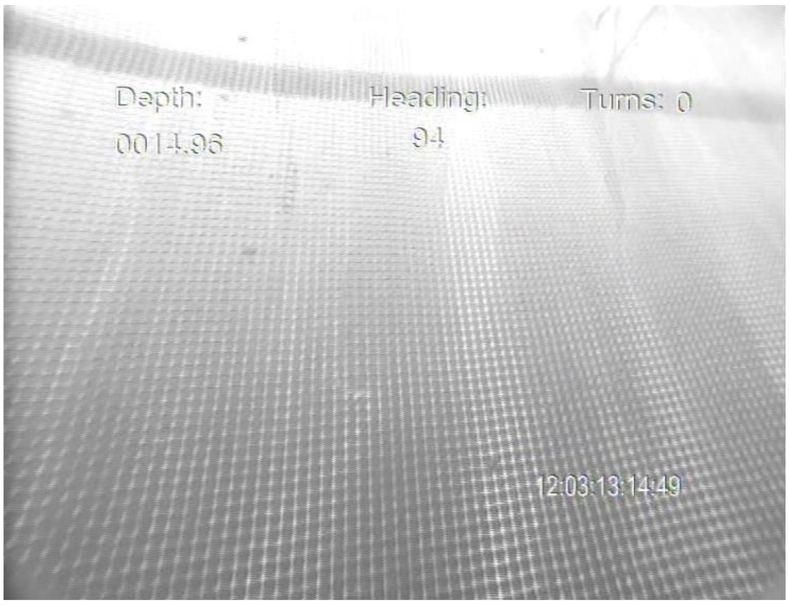


Figure 4: ROV inspection video frame

1.1.2. Previous work

In 2010-2011 Rolf Arne Hult Jakobsen wrote his Master thesis where he performed a preliminary study of using AOVs in aquaculture. Although not much have been used from his work his thesis have been important in making the department see that new hardware were needed to improve the usefulness of field tests and the results obtained from the field tests.

1.2. Limitations

During this thesis I was limited to use hardware bought by the department although it would have been better to get to borrow the actual hardware used by Argus.

Due to problems with acquiring the hardware needed and getting it to work I only got about a month of development using actual useful data which of course limited the progress I was able to make with my system.

1.3. Contributions

During my work I have made two computer programs that can be used for further testing. The program named System 2 can be used as a test system where different algorithm can be tested using different color spaces, image enhancement techniques and parameters. The final system is more of a finished system so the parameters are set. The performance of this system on an actual sea-cage could be tested.

The edge detection method of performing segmentation developed during this thesis seems to be a fairly robust method given that the edges are clear in the image.

During this thesis new hardware has been acquired by the school that I, with the help of Morten Engelhardt Olsen got up and running. This new camera hardware is a big upgrade to the camera system previously used.

1.4.Outline

Chapter 2 Theory: In the theory chapter the theory behind the color spaces, segmentation methods, net damage detection and growth detection methods used are presented.

Chapter 3 Software development: In the software design chapter the evolution of the finished damage detection program is shown, from early prototyping in Matlab to the final system.

Chapter 4 Field testing: In this chapter the hardware used for field testing and the actual testing performed is described. Also the problem of motion blurring is covered in this chapter.

Chapter 5 Testing captured video: In this chapter the systems developed are tested using videos obtained during the field test.

Chapter 6 Discussion: Here what I have found during my work are discussed.

Chapter 7 Conclusion: In this chapter conclusions are drawn about the feasibility of a damage detection system using the ROVs on-board camera.

Chapter 8 Future work: In this chapter possible future work to be done based on the Master thesis is presented.

2. Theory

In this chapter the background theory of some of the important algorithms and methods used are presented.

2.1. Color Spaces

The first thing to do when considering computer vision algorithms is to determine the color space to use. There are different reasons why different color spaces can be useful.

- One reason is that when obtaining color images the images generally have three channels of color information, for example red, green and blue (RGB). Processing all the pixels in an image over all three channels adds potential excessive computational time. If it is possible to use single- or double-channel color spaces that obtain all the information needed this can reduce computational time significantly.
- Another reason to use different color spaces is to use the color space that gives the maximum amount of information for the rest of the system. If a color system that enhances the sea cage structure in the image would make operations like segmentation and subsequently damage detection easier.

In (Kerr, 2010) it is stated that when the eye perceives color tiny elements of what we see is observed on the retina which has three kinds of cones. Each cone reacts differently to light at each wavelength. When an area of the retina is bathed in light the spectrum of the light on each cone is multiplied by the cones spectral response. The three products are then added together to give the output from the cone. The eye cone response curve can be seen in Figure 5.

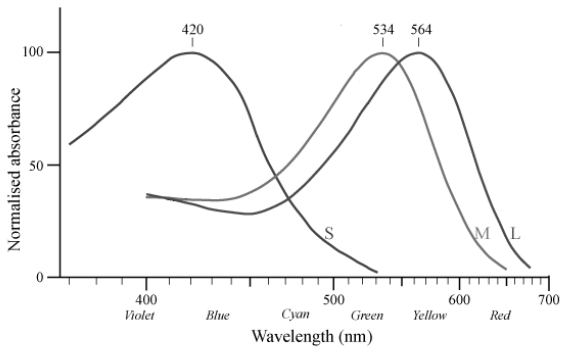


Figure 5: Eye cone response curve (Kerr, 2010)

2.1.1. CIE RGB

To characterize the response of the human eye a series of experiments were performed in 1931. The basic experiment used a small screen where one half of the screen had the light of a test color while the other half was made up of a composition of three primaries. The primaries had spectrums consisting of only a single wave length. The test person adjusted the three primaries to match the test light. The value of the primaries was recorded and another test light shown. Using this method with multiple observers they accumulated a vast amount of data, (Kerr, 2010).

In Figure 6 functions for the primaries which is shown. Here the amounts of each primary color that are needed to make a specific color wave-length are shown. While they are not equivalent of the curves of Figure 5 there is a mathematical relationship between any set of matching functions and the eye response curves which can be exploited.

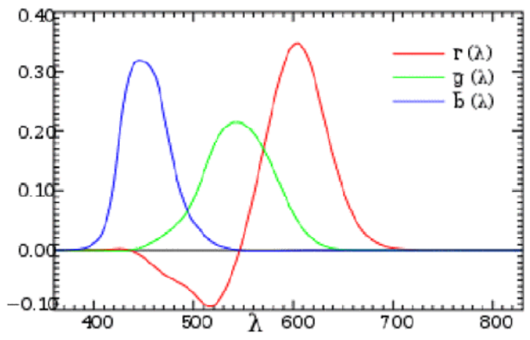


Figure 6: CIE RGB color matching functions (Kerr, 2010)

The three primaries could be described as red, green and blue and consisted of single wavelengths of 200nm, 546.1nm and 435.8nm respectively. This additive color space came to be known as the CIE RGB color space. Each pixel consists of three values, one for red, one for green and one for blue. When used in computer vision algorithms one can use the channels individually, combine of two channels or combine all three channels. In (Jakobsen, 2011) it was found that the red color space could perform well for our purposes. This will be tested during this thesis, however using a color space which is so bound to a single color may not be robust enough seeing as the color in the light is different close to the surface compared to further down.

2.1.2. CIE XYZ

The XYZ color space is defined by the International Commission on Illumination (CIE). Just as the RGB color space the XYZ space is an additive color space.

A disadvantage of the RGB color space is that the red primary matching function contains negative values. In the time where the analytical work of performing multiplications of the light spectrum values by the curve values at different wavelengths were done with calculators and on paper they did not want to use both positive and negative values. Since doing that could increase the risk of an error in the calculation. They needed a new color space, and thus the XYZ color space was created.

The RGB color space was transformed in to the new XYZ color space. The XYZ color space was created such that Y corresponds to how the human eye perceives lightness, Y therefore represents the luminance of the colors. Z is almost equal to the blue function in Figure 6 and X is a nonnegative approximation to the red function in Figure 6. The matching functions for the CIE XYZ color space can be seen in Figure 7.

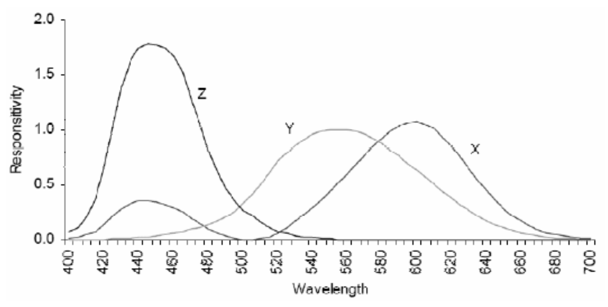


Figure 7: CIE XYZ color matching functions (Kerr, 2010)

2.1.3. CIE xyY

CIE xyY is a color space called a luminance-chrominance color space. In this kind of color space one coordinate gives information about the luminance while the two other coordinates gives information about the chromaticity. The chromaticity specifies the quality of a color and is not luminance dependent.

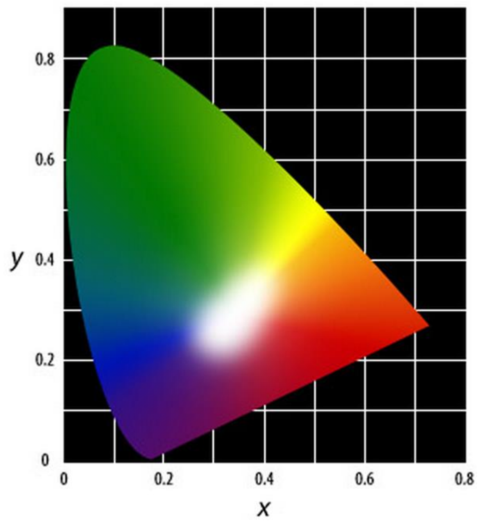


Figure 8: CIE xyY Chromaticity Diagram from 1931 (Adobe Systems Incorporated, 2000)

The CIE xyY color space is based on the CIE XYZ color space defined as:

$$x = \frac{X}{X+Y+Z} \quad y = \frac{Y}{X+Y+Z} \quad Y = Y$$

Equation 1: From CIE XYZ to CIE xyY color space

2.1.4. CIE L*a*b

L*a*b* is a nonlinear transformation of the XYZ color space. In this color space a change in color-coordinate would give an equal perceptual difference. (Hoffmann, 2013). With L*a*b* CIE had a color space which was intended to provide a standard, approximately uniform color space. This color space could be used by everyone to make it easy to compare colors.

The L*a*b* is organized as a cube as shown in Figure 9. The L* channel is the luminance channel which ranges from 0(black) to 100(white). The a* channel represents the color on a scale from green to red where positive a* numbers represent red and negative a* numbers represent green. The b* channel represents color in the range from blue to yellow where positive b* numbers represent yellow and negative b* numbers represent blue. In theory the a* and b* axis doesn't have a maximum and minimum scale, although for practical purposes they are usually limited to a range from -128 to 127 thereby consisting of 256 levels, 8 bits.

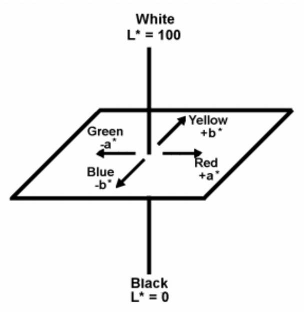


Figure 9: CIE L*a*b* (HunterLab, 2008)

2.2.Segmentation methods

One crucial task of a computer vision algorithm is to separate the interesting objects (foreground) form the rest of the image (background). There are many different methods to achieve this. Some of these approaches include global approaches which use histograms of the pixel intensities to separate the image, approaches that uses edges or texture to separate the image and local approaches that use masks of different types to separate items in the image.

From (Gonzalez , Woods, & Eddins, 2009) it is stated that global thresholding methods typically fails when the images are subjected to non-uniform background illumination. When this occurs one can use local statistics to improve the thresholding. In this thesis some of the segmentation methods have been developed both as global approaches and as local approaches where the image has been divided into several sub-images. The segmentation methods have then been performed on the sub-images individually.

2.2.1. Naïve Thresholding

Thresholding can be done as simple as declaring that all the pixels that are over a certain threshold are foreground and the other pixels are background. This is a naïve approach that to a large degree is prone to error from disturbances. Naïve thresholding is computational cheap to perform and with some smart preprocessing and optimal choice of color space it can be a viable option.

2.2.2. Adaptive thresholding

Adaptive thresholding uses local characteristics to determine the threshold value. This can be achieved by looking at the neighborhood of each pixel and use the mean value as the threshold value. It is also possible to cross-correlate the neighborhood with a Gaussian window and using that to determine the threshold value.

In Equation 2 an example of adaptive thresholding using the mean value is shown. Here the process is only shown for the yellow and green pixels. If a 3-by-3 mask is taken over the yellow pixel and the pixels that fall within in that mask is considered, this is called the neighborhood of the yellow pixel. The values within the mask are 0-0-0-30-30-50-100-100-100 giving the median as 50. Since the pixel in question has a value of 50 and the median value is 30 the yellow pixel is considered a foreground pixel and set to 255. The same process for the green pixel gives a median value of 50 and since the green pixel value is 30 it is considered as background and set to 0.

$$\begin{bmatrix} 0 & 30 & 100 & 50 & 0 \\ 0 & 50 & 100 & 50 & 0 \\ 0 & 30 & 100 & 30 & 0 \\ 0 & 50 & 100 & 50 & 0 \\ 0 & 50 & 100 & 50 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 50 & 100 \\ 0 & 50 & 100 \\ 0 & 50 & 100 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 50 & 100 & 50 & 0 \\ 0 & 255 & 100 & 50 & 0 \\ 0 & 50 & 100 & 30 & 0 \\ 0 & 50 & 100 & 50 & 0 \\ 0 & 50 & 100 & 50 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 30 & 100 & 50 & 0 \\ 0 & 50 & 100 & 50 & 0 \\ 0 & 30 & 100 & 30 & 0 \\ 0 & 50 & 100 & 50 & 0 \\ 0 & 50 & 100 & 50 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 100 & 50 & 0 \\ 100 & 30 & 0 \\ 100 & 50 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0 & 50 & 100 & 50 & 0 \\ 0 & 255 & 100 & 50 & 0 \\ 0 & 50 & 100 & 0 & 0 \\ 0 & 50 & 100 & 50 & 0 \\ 0 & 50 & 100 & 50 & 0 \end{bmatrix}$$

Equation 2: Using adaptive mean thresholding. Thresholding on the yellow pixel(top row) and on the green pixel(bottom row)

If the pixel tested is over the found threshold it is considered a foreground pixel, else it is considered a background pixel. Typically a constant is deducted from the mean or Gaussian value to optimize the threshold value. This constant value is normally found empirically.

2.2.3. Otsu's method

Otsu's method is a method for segmentation that always finds foreground and background pixels. Otsu's method is based on choosing an optimal threshold value k. The threshold value is optimum in the sense that it chooses the value that maximizes the between-class variance $\sigma_B^2(k)$ defined in Equation 3.

$$\sigma_B^2(k) = P_1(k)[m_1(k) - m_G]^2 + P_2(k)[m_2(k) - m_G]^2$$

Equation 3: Otsu's method. P_x is the probability of set C_x occuring. m_x is the mean intensities of the pixels in the set C_x and m_G is the global mean intensity of the entire image. (Gonzalez , Woods, & Eddins, 2009)

The idea behind this approach is that the larger the between-class variance is, the higher is the likelyhood that a proper segmentation is obtained. Otus's method is vulnerable to non-uniform lighting conditions. Using Otsu's method globally can therefore perform worse than performing the method on sub-images to get a better m_G value, and hence a better segmentation result, for each part of the image. A problem with dividing the image up is that since Otsu's method always finds foreground and background pixels it can divide even sub-images with for example only sea into foreground and background. This is unwanted behavior bacause it makes operations like damage detection and growth detection harder. Another problem with the method arises when Otsu's method segments the sub-images and defines the foreground object in one sub-image as the background object in the next sub-image. In other words it can define the sea cage as foreground in one sub-image and as background in another. This must be acounted for if Otsu's method is considered a viable method.

2.2.4. Using edge detection directly for segmentation

When segmenting out a sea cage from the surrounding sea the cage appears as lines in the image. Edge detection can be a valid choice to use for segmentation since the lines of the sea cage will be found as edges in the image. Finding edges can be done by taking the gradient of the image. To do so one can take a kernel like the Sobel operator in Equation 5 and convolve it with an image. The result will be an image with strong lines where the edges in the image are. In this thesis the object to be segmented is an sea cage, the masks in the sea cage should be strongly seen in the resulting image after performing an edge detection algorithm.

The reason for this I that an edge can be viewed as a change in the pixel value. To explain the principal behind the edge detectors used in this thesis the 1D case will be presented. In Figure 10 an edge with its first and second derivative can be seen. The edge in the top left image will be visible as a high positive or negative value in the image of first derivatives(top right). This is what is found using the Sobel and Scharr operator. They can be used as approximations of the first derivative. As approximations of the second derivative the Laplacian or Sobel operator can be used. In the bottom image in Figure 10 the second derivative is shown, in the second derivative an edge can be seen as a zero crossing. Zeros however will not only occur where edges are but also in seemingly meaningless locations, this may need to be filtered out.

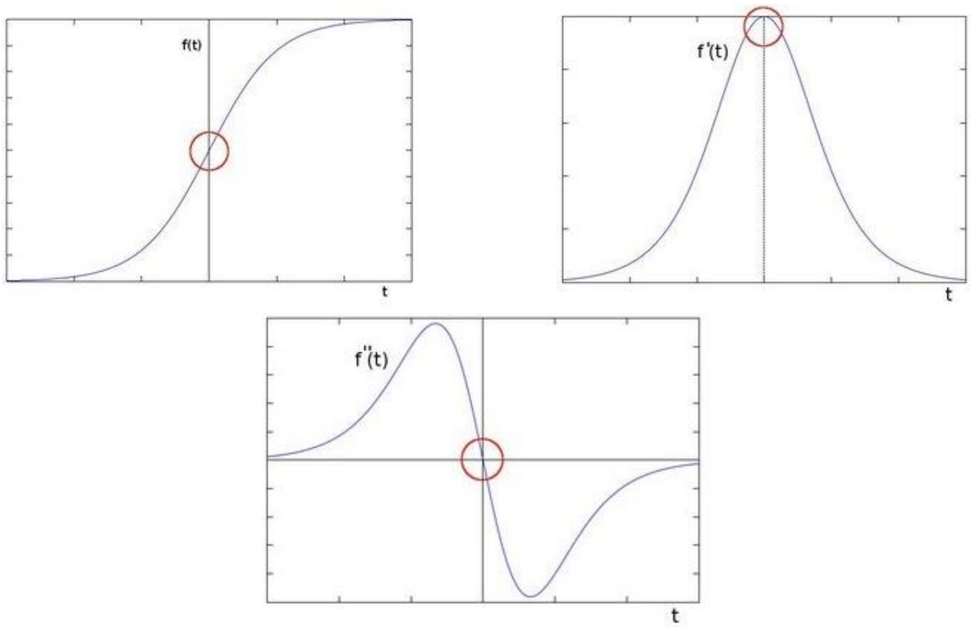


Figure 10: Using derivatives for edge detection, edge (top left), first derivative(top right) and second derivative(bottom), (OpenCV dev team, 2013)

The Sobel operator is separable. Separable means that one can split the kernel up into a column and a row vector and filter the image first with one of them and then that result is filtered with the other vector. This saves computing time. For example using a M-by-N image with a P-by-Q filter kernel one uses around MNPQ multiplications and additions, if the filtering is done in two steps, filtering with one vector uses roughly MNP operations, filtering with the second filter uses roughly MNQ operations for a combined MN(P+Q) operations. This saves $PQ/(P+Q)$ operations (Eddins, 2006).

Using a Sobel operator one can take the derivative in the dx and dy directions in an image and either look at the results separately or combined. When combining the results one can get the edge strength which can be calculated as $\sqrt{x^2+y^2}$ or simply as |x|+|y| (Sonka, Hlavac, & Boyle , 2008). In OpenCV there is a function that calculates the first, second, third or mixed derivatives for an image. The formula used to calculate the output image from this technique is found in Equation 4.

$$output = \frac{\partial^{xorder+yorder}}{\partial \mathrm{x}^{xorder}\ \partial \mathrm{y}^{yorder}}\ input$$

Equation 4: Sobel equation, (OpenCV dev team, 2013)

The function can also use the Scharr operator which is a 3x3 first x- or y- derivation filter kernel that can be more accurate than a 3x3 first order Sobel filter. In Equation 5 and Equation 6 some of the different filter masks used for Sobel and Scharr edge detection in OpenCV is shown.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ 2 & 0 & -2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

Equation 5: Sobel operators: first second and third derivative kernels, x-direction (top row) and y-direction (bottom row)

$$\begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} \qquad \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix}$$

Equation 6: Scharr operator: x-direction kernel(left) and y-direction kernel(right)

Another edge detection kernel that can be used is the Laplacian. The Laplacian approximates the second derivative and gives the magnitude of the gradient as its result (Sonka, Hlavac, & Boyle , Image Processing, Analysis, and Machine Vision, 2008). The Laplace operator is invariant to rotation if it is implemented as in Equation 7.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Equation 7: Laplacian operators, 4-connectivity kernel(left) and 8-connectivity kernel(right), (Sonka, Hlavac, & Boyle , 2008)

One of the drawbacks of the Laplacian operator is that some edges may get a double response (Sonka, Hlavac, & Boyle , Image Processing, Analysis, and Machine Vision, 2008). The Laplacian operator will also enhance noise in the image since noise will be found as gradients in the image.

2.2.5. Using edge detection to enhance Otsu's method

In (Gonzalez , Woods, & Eddins, 2009) chapter 11.3.5 an algorithm that uses edges to improve Otsu's method is presented. As shown in Figure 11 this algorithm first computes the edges for an input image. To do that edge detectors like a Sobel detector can be used. After obtaining the edges the edge image is thresholded using a threshold value. This threshold value is found by using a percentile function which thresholds the edge image so that the edge pixels with a value larger than the given percentile are kept. Now we have a binary image where the ones are the strong edges. The binary image is laid on the original image and a histogram is calculated using only the pixels in the original image corresponding to ones in the binary image. Lastly this histogram is segmented using Otsu's method.

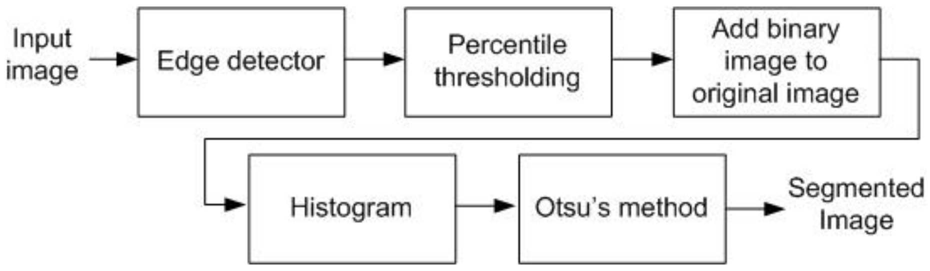


Figure 11: Using edge detection to enhance Otsu's method

2.3. Image and segmentation enhancement techniques

Enhancing the image in different places along the computer vision pipeline can be the difference between a total failure and a success.. Different functions were tested during this thesis to improve the end result.

2.3.1. Morphological operations

Mathematical morphology is a tool used for extracting image components that can be used for representing or describing a region shape. From (Gonzalez, Woods, & Eddins, Morphological Image Processing, 2009) it is stated that "morphology is a cornerstone of the mathematical set of tools underlying development of techniques that extract "meaning" from an image".

Morphology is based on set theory where logical operations like union and intersection are used. Dilation and erosion are fundamental operations in morphology, they use the logical operations union and intersection to manipulate images.

2.3.1.1. Dilation

Dilation is used to thicken foreground pixels in an image. It is essentially the union between a binary image and a structuring element that gets swept over the image. The structuring element is taken pixel-by-pixel over the entire image thereby thickening every foreground pixel in the image by the structuring element. The dilation of an image A by a structuring element B is denoted A ⊕ B. In Equation 8 the result of dilation on a small image with a column vector with three ones is shown.

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix} \oplus \begin{matrix} 1 \\ \mathbf{1} \\ 1 \end{matrix} = \begin{matrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{matrix}$$

Equation 8: Dilation

In Figure 12 the effects of dilation can be seen. In the top row is one round of dilation used on a dot; in the bottom row is one round of dilation used on the word “frame”.

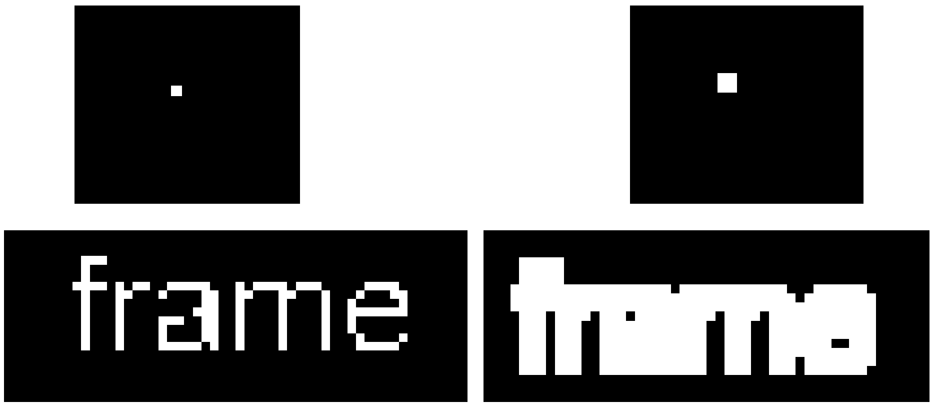


Figure 12: Morphological dilation. Before(left column) and after(right column)

Dilation can be a very important function to enhance segmentation results.

2.3.1.2. Erosion

While dilation thickens lines erosion thins foreground objects in the image. Erosion is essentially the intersection of an image A and a structuring element B. The erosion of an image A by a structuring element B is denoted A ⊖ B. In Equation 9 the result of erosion on a small image with a column vector with three ones is shown.

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix} \ominus \begin{matrix} 1 \\ \mathbf{1} \\ 1 \end{matrix} = \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

Equation 9: Erosion

2.3.1.3. Thinning

Is an operation that as the name suggests thins, or skeletonize structures. It is normally applied to binary images to make the lines found using an edge detector a single pixel wide while still keeping the length of the line intact, (Perkins, Fisher, Wolfart, & Walker, 2003). In Figure 13 the effects of morphological thinning can be seen.

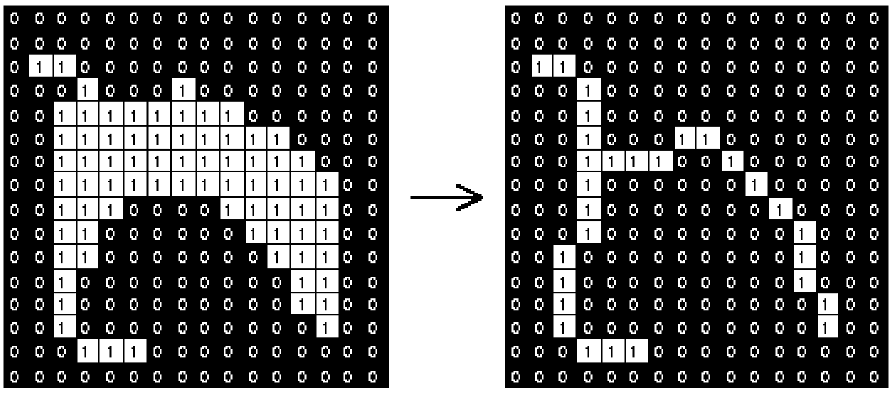


Figure 13: Morphological thinning, (Perkins, Fisher, Wolfart, & Walker, 2003)

2.3.1.4. Spurring

Spurring is an operation one can set the bwmorph function in MATLAB® to perform. The operation removes end point of lines without removing small objects in the image completely.

2.3.1.5. Opening and closing on gray-scale images

Opening and closing are operations using combinations of dilation and erosion. Opening is performed by first performing erosion and then dilating that result. The opening of an image A by a structuring element B can be written as $A \circ B = (A \ominus B) \oplus B$. When performing the opening operation one first uses erosion to remove small objects and then one uses dilation to restore the remaining objects. This operation was used to remove salt and pepper noise in the final system.

Closing is performed by first performing dilation and then eroding that result. The closing of an image A by a structuring element B can be written as $A \bullet B = (A \oplus B) \ominus B$. Closing can be used to fill up small holes in objects. When performing the closing operation one first uses dilation to close up the small holes and then the objects overall shape is restored using erosion.

2.3.1.6. Top hat

One very interesting way of using morphological opening in gray-scale images is to perform a top hat transformation. The top hat transformation can be used to extract light objects from an image, (Sonka, Hlavac, & Boyle, 13. Mathematical morphology, 2008). It can also be used to equalize uneven background illumination in a grey-scale image. When using a top hat transformation one subtracts the opening of an image from the image.

2.3.1.7. Bottom hat

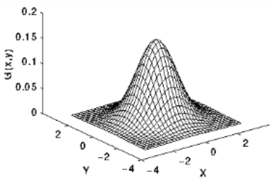
The bottom hat of an image is another way to use closing in gray-scale images. When using the bottom hat transform one subtracts the closing of an image from the image. The bottom hat transform is often called the black hat transform because it can be used to extract dark regions from an image.

2.3.2. Gaussian smoothing

Gaussian smoothing is used to blur images and thus remove sharp details and noise, (Fisher, Perkins, Walker, & Wolfart, 2003). As with the Sobel operator Gaussian smoothing convolves a kernel with the image. The kernel is an approximation to the Gaussian function in Equation 10. The Gaussian distribution and the kernel approximating the Gaussian function can be seen in Figure 14.

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

Equation 10: 2-D isotropic Gaussian with σ – standard deviation and x and y - coordinates.



$$\begin{bmatrix} 0.018 & 0.049 & 0.049 & 0.018 \\ 0.049 & 0.134 & 0.134 & 0.049 \\ 0.049 & 0.134 & 0.134 & 0.049 \\ 0.018 & 0.049 & 0.049 & 0.018 \end{bmatrix}$$

Figure 14: Gaussian distribution (left), (Fisher, Perkins, Walker, & Wolfart, 2003) and Gaussian kernel approximation with variance = 1 (right), (Schellewald, 2012)

2.4.Net damage detection using region growing

A crucial task of the system is to detect damage to the sea-cage. The system should detect not only full-fledged holes and tears but also weaknesses that can lead to major holes and tears. To identify a damaged part from the video feed a region growth algorithm can be used. For an algorithm to work the net must have been identified as foreground while the holes in the net masks are identified as background. Because of fish, debris and other objects in the image this will not necessarily always be the case. However it can still be possible to grow an area for each region of background pixels and see it any of the areas found are suspiciously large. A suspect region can then be either holes and tears or other objects like fish, debris in the water or debris stuck on the camera.

To find out if one of the region grown areas is suspiciously large one can make a list over all the regions found in an image and then mark out the regions that are over a certain size compared to the rest. The distribution of mask areas in terms of number of pixels will have to be tested. Partial masks at the image borders and distorted masks due to rippling of the net will have to be accounted for.

2.5.Growth detection

Using the system to identify the amount of growth in the sea-cage is another potential use of the ROVs video feed. A rather simple and computational cheap approach should be able to check the growth. If the net together with all growth is segmented as foreground growth can be found as easily as comparing the number of foreground pixels in the image with a reference image. The growth can be given as a percentage of growth where the number of foreground pixels in the reference frame is 0% and an image with only foreground pixels are 100%. This can be calculated as shown in Equation 11.

$$diff = video\_frame\_percentage - reference\_video\_percentage$$
$$percentage\_growth = (diff * 100)/(100 - reference\_video\_percentage)$$

Equation 11: Growth percentage

3. Software development

When developing the software it was important that the segmentation method should be able to correctly segment out the net from the sea and find damages present in the net. The segmentation method should be robust so it can operate under varying light conditions and should work independent of factors like the color of the net, the color of the sea, small parts of debris.

In the system development process the system were developed over four development cycles. In the first cycle some potential methods was prototyped in MATLAB®. These methods are described in 3.1 Prototype.

After prototyping the first test system were made. This system were made to test video obtained during the field test so it has a lot of parameters that the user can change so the user can test what methods works best. The first system is described in 3.2.

After assessment of the performance of the first system on the videos obtained in the field test (chapter 4.2) a second system were made. In system 2 the algorithms of system 1 were enhanced, obsolete functionality were removed and some new algorithms were added. The system is described in 3.3

At the end a more user friendly system was made where the best working algorithms were implemented. The user of the system only need to choose input source, program mode and which image the user wants to see.

Test system 1 and 2 and the final system were made using OpenCV 2.4.0 and Qt 5.0.1 in C++. With OpenCV many different computer vision algorithms are readily available. The algorithms are also optimized so they perform well in a real-time sense. Qt is a cross-platform application and UI framework that made the development of the GUI and OS interaction easier. Qt Creator is the supporting IDE for Qt.

In System 1, System 2 and the final system the video frames get processed via the pipeline shown in Figure 15. The image first gets converted to the chosen color space, then image enhancement techniques are applied such as Gaussian smoothing. The frame then gets segmented using the chosen segmentation method then the segmentation result is enhanced using operations like dilation and removing salt-and-pepper noise before the image is sent to the chosen Program mode. The program mode algorithm is run and the appropriate frames are shown in the GUI. For the final system only one image is shown in the GUI, the user can choose which video frame he/she wants to see at any given time.

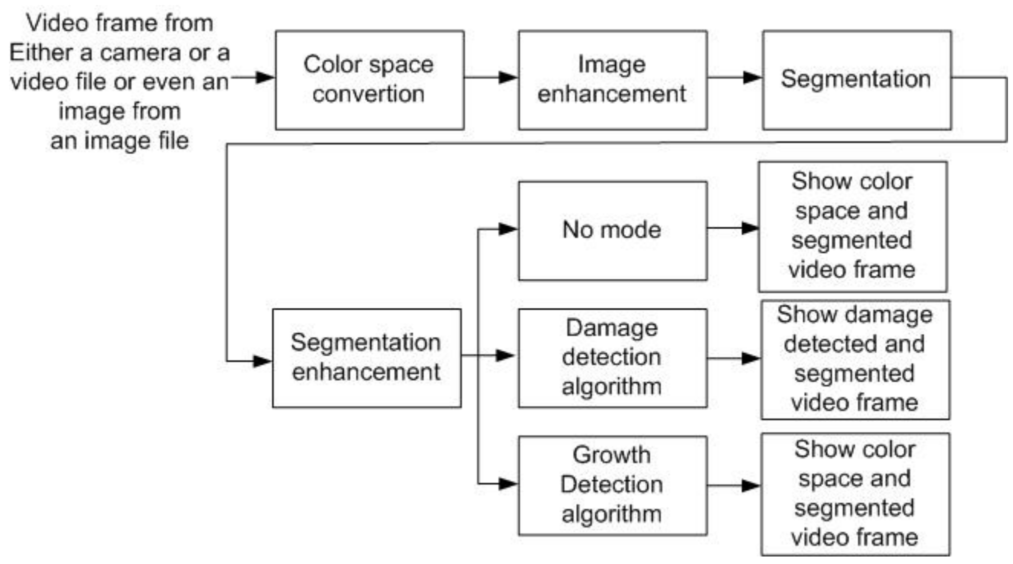

Video frame from Either a camera or a video file or even an image from an image file
Color space convertion
Image enhancement
Segmentation
Segmentation enhancement
No mode
Show color space and segmented video frame
Damage detection algorithm
Show damage detected and segmented video frame
Growth Detection algorithm
Show color space and segmented video frame

Figure 15: Software pipeline

3.1.Prototype

To get an understanding of the problem at hand it was decided to start the design process by testing different methods in MATLAB® before making a working system using Qt with OpenCV. Videos given by Per Rundtop was used for the prototyping stage.

After testing different approaches the most promising one were implemented as a MATLAB® program that is enclosed in the attached zip-file. It is in the “Prototype files” folder and the main file is named “Prototype.m”. The program uses the Y channel from the XYZ color space and segmentation using Sobel edge detection and histogram percentile thresholding. To find damaged areas region growing is used. The program when used on the image seen on top in Figure 16 finds the hole as seen in the bottom images in Figure 16. Since the quality of this image was quite poor the program also found more masks that were not intact but when studying the image one can see that there are so many pixels missing so there are a lot of seemingly damaged masks. The system seems to find the damaged masks and holes it should.

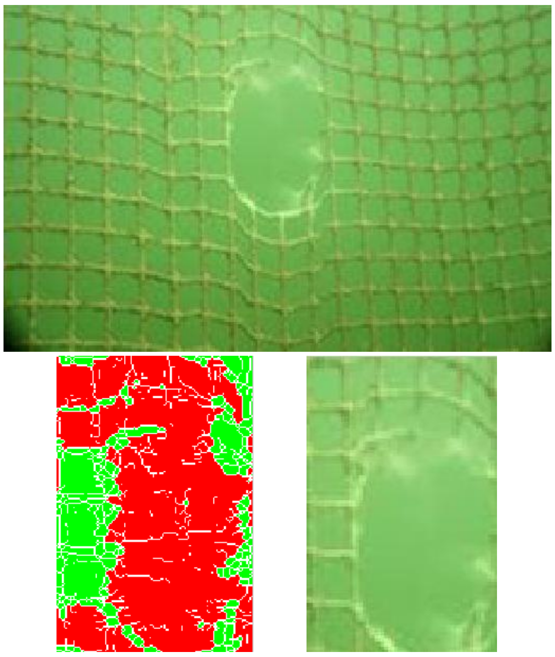


Figure 16: Poor quality image with hole (top). Output from program (bottom) with the found suspecting area (left) and its corresponding area in the input image (right)

3.1.1. Color Spaces

In the prototype stage the following color spaces were implemented:

- R, G and B from the original RGB image.
- X, Y and Z from the XYZ color space from (Paschos & Valavanis, 1999)
- x, y and Y from the xyY color space from Equation 1,page number 8.

Of the different color spaces the red from RGB and Y from the XYZ and xyY color spaces were found to be the two most promising ones.

3.1.2. Segmentation methods

Different segmenting methods were tried to in order to segment out the sea-cage from the background.

3.1.2.1. Otsu’s method

To test Otsu’s method the image seen in Figure 17 were used.

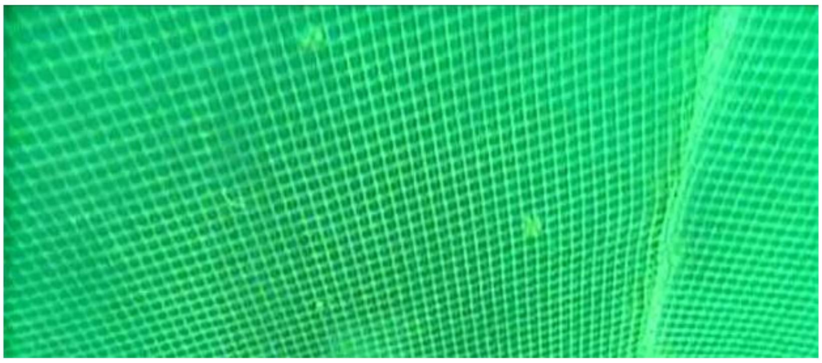


Figure 17: Image used for testing Otsu's method

The image was first filtered with a Gaussian filter and then segmented using Otsu’s method. When using Otsu's method it became apparent that using it on the whole image led to bad segmentation results. It was therefore tried to segment the image in smaller sub-images. The image was divided into 25 sub-images, which was independently segmented using Otsu’s method and stitched together. The differences between using a global approach or using sub-images can be seen in Figure 18 where in the globally image to the left the many of the sea-cage masks are not whole which when further processed will lead to the system believing the net to have holes. In the image segmented with sub-images the masks are better and the thick rope going vertically in the right of the image is more distinct. The differences can be more easily seen after using morphological operations like shown in Figure 19. Here the segmented images have been through one round of dilation followed by thinning and spurring (both operations performed until no further changes occurred)

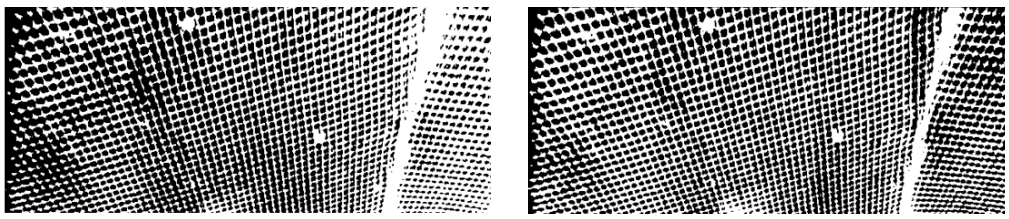


Figure 18: Thresholded image using Otsu's method globally(left) and using Otsu's method on sub-images(right)

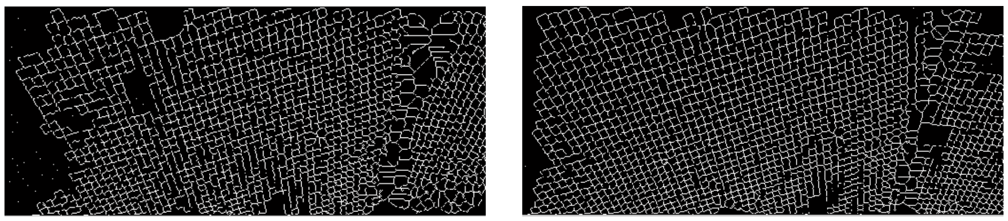


Figure 19: Thresholded image after morphological operations have been applied. Global Otsu's method(left) and Otsu's method on sub-images(right)

Using Otsu’s method the Y color space gave better results compared to the Red color space when it comes to the masks, the Red color space could however see the rope to the right in the image more clearly. The different color spaces are compared in Figure 20.

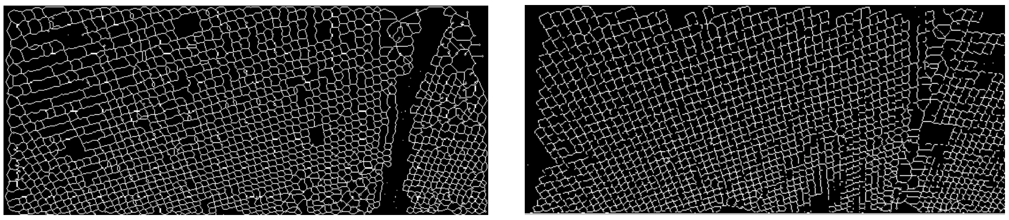


Figure 20: Otsu's method using dilation, thinning, and spurring. Red color space (left) and Y color space (right)

3.1.2.2. Using Sobel edge detection for segmentation

Based on the algorithm described in 2.2.5 Using edge detection to enhance Otsu's method, a simplified method was tested using the binary image found after performing percentile thresholding as the segmentation result.

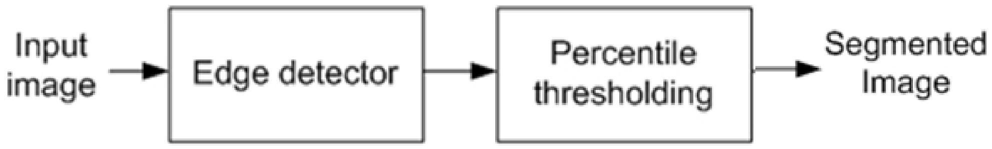


Figure 21: Sobel edge detection for segmentation

Using this approach the net will be separated as foreground since it essentially is edges. Strong edge pixels should be found from background to net and from net to background and the net should stand out strongly in the segmented image.

For this approach the input image were processed into two different images. One where Sobel edge detection had been performed in the x-direction and one in the y-direction. These edge detected images were then merged. Since the edges between the sea cage and the sea are pronounced the other edges found in the image can be ignored by using a percentile thresholding function that thresholds a given percentile of the edge pixels as foreground based on edge strength. The given percentile to be used is a parameter that could be adjusted at run-time since the same percentile won't necessarily work for different environmental differences like different cameras, lighting conditions and sea cages. When using this method it wasn't necessary to dilate the segmented image before thinning it. This introduced less corrupted masks compared to the other methods. In Figure 22 the difference with and without dilation is shown. With dilation more masks are found, however they don't have the appearance as the original image. Using dilation together with thinning doesn't seem a viable choice.

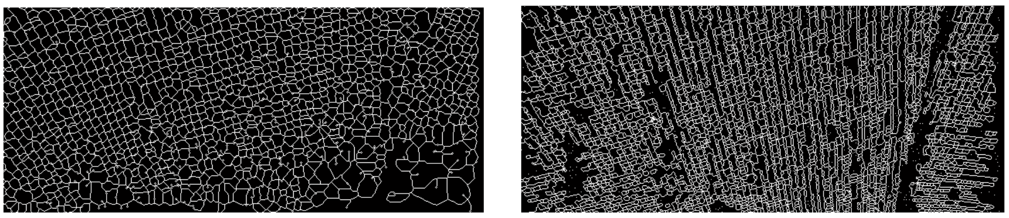


Figure 22: Foreground and background using segmentation with edge detection. Image dilated before thinning and spurring (left) and image not dilated (right)

3.1.3. Net damage detection

To spot damages to the net the following algorithm was developed:

1. Scan the image, when a background pixel is found grow a region of background pixels from that seed pixel using 4-connectivity. Give the pixels found a new value to mark them as foreground or background.
2. If the region is abnormally large it is saved as a suspect region and marked in red, if it is not suspect it is colored in green as shown in Figure 23
3. Repeat until the whole image is scanned.
4. Search through the suspect regions, Display the largest suspect area that is not on the edge of the image with the same area in the original image as shown in Figure 24

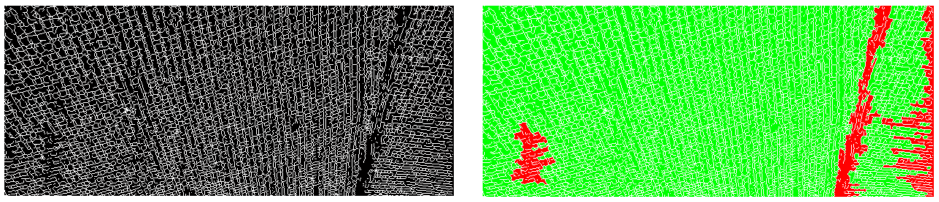


Figure 23: Input to region growing algorithm (left), output from region growing (right)

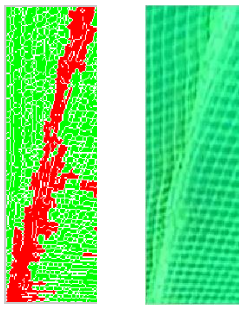


Figure 24: Biggest suspect region (left) and the same area in the original image (right)

3.1.4. Cross-correlation detection of net

Using cross-correlation to find different objects in the image can have multiple uses. It can be used to see if there is fish present in the image, check for holes, tears and ropes or to find areas with double net.

To test this approach in MATLAB® an image as shown to the left in Figure 25 was used. To see how the cross-correlation method works a mask from that image was taken out as shown the middle image in Figure 25. The resulting point on the original image found to have maximum correlation was indeed the correct mask.

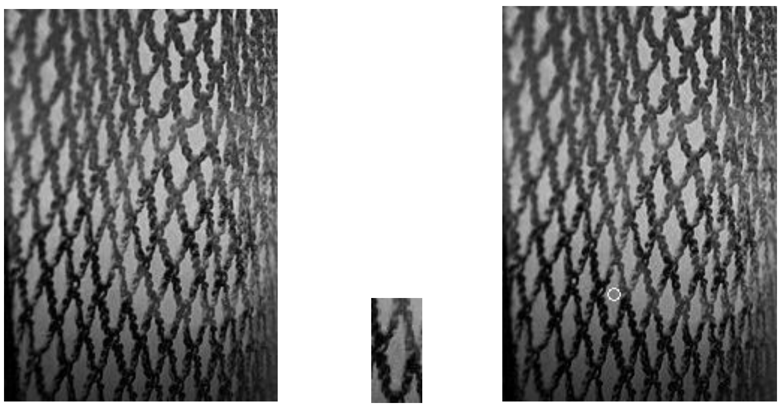


Figure 25: Test image (left), test structure (middle) and point with maximum correlation (right)

After that an algorithm that would segment the input image, perform thinning and spurring and then search for a given test structure were tried. The result can be seen in Figure 26, the algorithm does not seem to find what seem to be good masks in the input image such as the masks marked in red in the left picture in Figure 26. This can be because the structuring element is too small. The system needs the structuring element to be of the same size as the structure in the image. This makes this approach potentially fragile. Because of this the method weren't further enhanced. From what have been found in this test it became clear that to use this technique to find each mask doesn't have much promise, it can however be used to find different objects in an image as stated above.

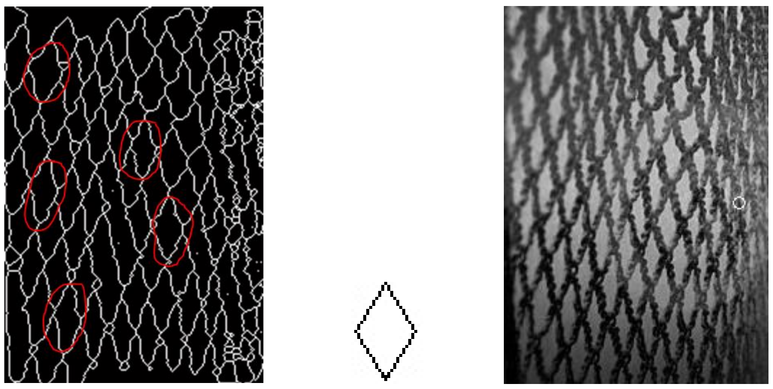


Figure 26: Thinned test image (left) with some good masks structures encircled, test structure (middle) and point with maximum correlation (right)

3.1.5. Boundary tracing using shape number

Using the shape number of a structure is another way of classifying the structure. A shape number is a representation of a shape using a chain code generated as shown in Figure 27.

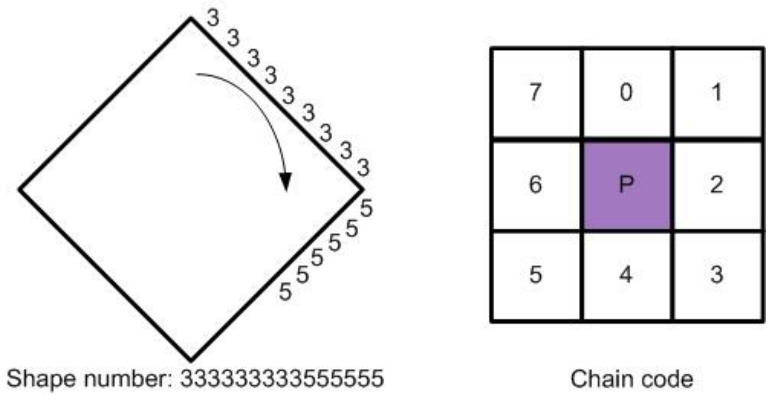


Figure 27: Generation of shape number (left) and Chain code (right)

Having obtained the shape number of a structure it can be compared to shape numbers of known structures to classify the unknown structure. In the MATLAB® program it was tried to identify valid net masks as an option to region growing, it is also believed that this can be used to classify other things in the image like fish or damage types. For this program a diamond shape was represented with its shape number as:
333333333333333333333333333333555555555555555555555555555555777777777777777
7777777777777111111111111111111111111111111

The algorithm transverses a segmented image and when it finds a white pixel it traces the shape of the structure. The algorithm always searching inwards first so as not to trace multiple masks in one search. If the shape number found is of a valid size, the shape found is compared to the diamond shape. Too small and too large sizes are not valid. Too small sizes are not part of a mask but may be disturbances in the image and too large sizes are most likely due to a lost trace. A large shape number can also mean that a hole is found but on this stage that scenario wasn't treated. If the cost of making the found shape into the diamond shape is not to large the algorithm acknowledges the shape and draws it in an output image. The cost is found by setting three parameters, the cost of deletion, substitution and insertion. Then a dynamic spelling corrector was developed to calculate a cost chart which finds the lowest possible cost of making one shape number into another.

The segmented image and output image can be seen in Figure 28.

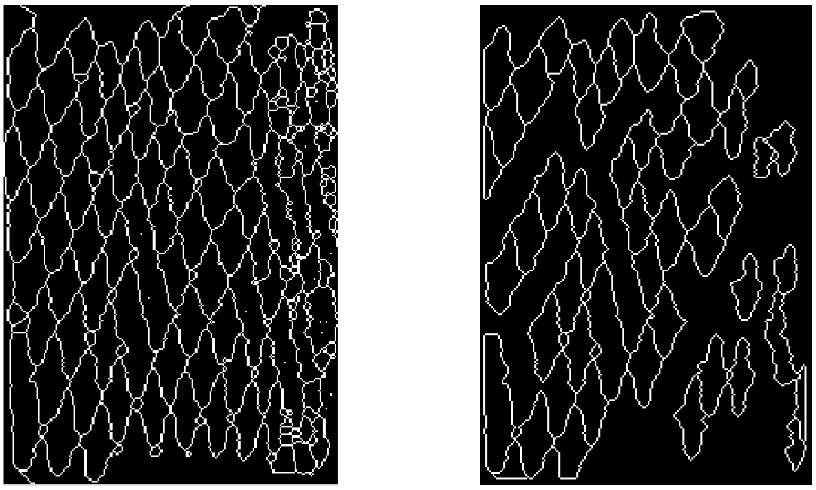
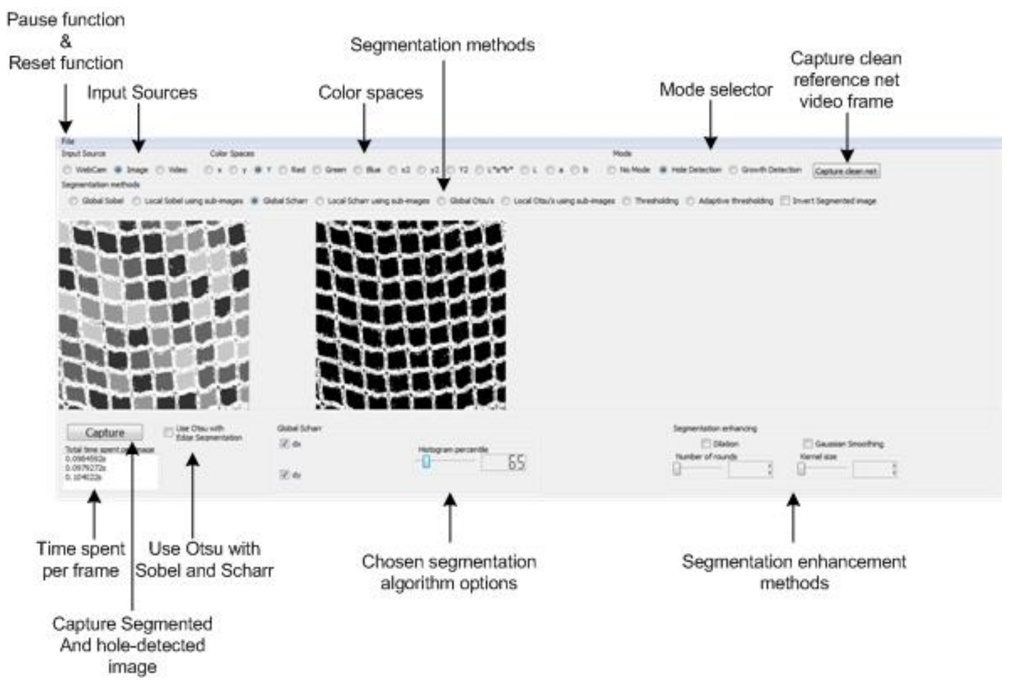


Figure 28: Segmented image (left) and valid net masks found based on shape number (right)

As can be seen most of the “good” masks were found, however if the images are as bad as this input image this approach seems too fragile to be used to find net masks. The power of an approach like this seems to lie more in classifying an interesting object. It can be used to see if a suspicious region in the image is for example a hole, a tear, a fish or debris. The code developed to make the shape code and the dynamic spelling corrector is found in the Shapetester folder in the attached files.

3.2.System 1

The first system made was intended to be used in during the field test and on videos obtained from the field test. It has a lot of different color spaces and algorithms that shall be tested during the field test. In Figure 29 the GUI developed for System 1 is shown.



Pause function & Reset function
Input Sources
Color spaces
Segmentation methods
Mode selector
Capture clean reference net video frame
Time spent per frame
Use Otsu with Sobel and Scharr
Capture Segmented And hole-detected image
Chosen segmentation algorithm options
Segmentation enhancement methods

Figure 29: GUI, System 1

The user has the following options :

- Pause program and reset the program.
- Choose the input mode. The system can use web cameras, the camera used in the field test, images or videos as input.
- Choose the desired color space.
- Choose the desired segmentation method and change parameters specific to the chosen segmentation option
- Choose the desired segmentation enhancing methods.
- Select the program mode.

The user can also capture the segmented image and the hole detected image using the Capture button. The total computing time spent per video frame is shown to the bottom left.

3.2.1. Color Spaces

In the first version of the system the following single channel color spaces were implemented

- x, y and Y from the xyY color space from Equation 1, page number 8.
- R, G and B from the BGR2RGB transform in OpenCV.

- x, y and Y made using the RGB2XYZ transform in OpenCV then converted to xyY as explained in 2.1.3 CIE xyY.
- L*, a* and b* using the RGB2Lab transform in OpenCV.

3.2.2. Segmentation Methods

In the first version of the system the following segmentation methods were implemented:

- Segmentation using edge detection, Sobel from the prototype.
- Segmentation using edge detection, Scharr.
- Segmentation using Otsu's method, similar to prototype.
- Segmentation using simple thresholding.
- Segmentation using adaptive thresholding.

Different segmentation techniques were made to be able to test different kinds of parameters. The different methods and their respective changeable parameters will be listed here.

3.2.2.1. Segmentation using edge detection, Sobel

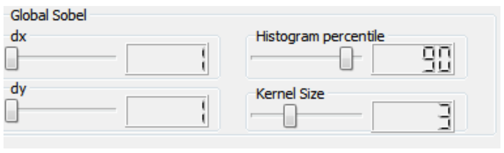

Global Sobel
dx 1
dy 1
Histogram percentile 90
Kernel Size 3

Figure 30: Global Sobel segmentation parameters, System 1

In edge detection using global Sobel there are four parameters to adjust. It is possible to take the derivative in both x and y direction (dx and dy), the system can handle from 1st to 6th derivatives. The Kernel size can also be adjusted. It is required that the derivatives are less than the kernel size, i.e. a kernel size of 3 can have 1st and 2nd derivatives. The last parameter is the histogram percentile parameter explained in 2.2.5 with a range from 60-99, note that for a percentile value of 60 the 40% strongest edge pixels are kept.

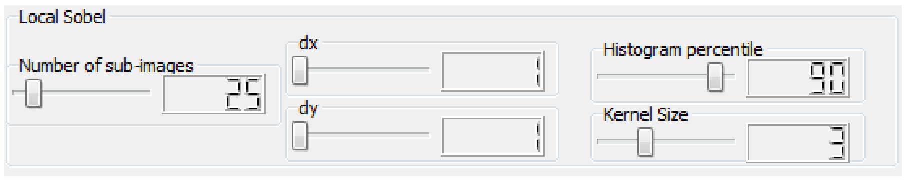

Local Sobel
Number of sub-images 25
dx 1
dy 1
Histogram percentile 90
Kernel Size 3

Figure 31: Local Sobel segmentation parameters, System 1

For the local Sobel approach the image is divided into sub-images, the number of sub-images is then naturally a parameter that can be adjusted. It has a range from 1(global) to 1600 sub-images. The default value is 25 sub-images. The sub-images are arranged in a grid pattern so with the default 25 sub-images the sub-images are arranged in a 5x5 pattern.

3.2.2.2. Segmentation using edge detection, Scharr

The Scharr operator is a special 3x3 mask that can be made using the OpenCV Sobel function. It can either have the first derivative in the x-direction or the first derivative in the y-direction or both at the same time. In the GUI this is chosen with the dx and dy check boxes. If both are chosen both directions are run and the results are then merged. In the local Scharr method one can select the number of sub-images. The last parameter is the histogram percentile parameter explained in 2.2.5 with a range from 60-99.

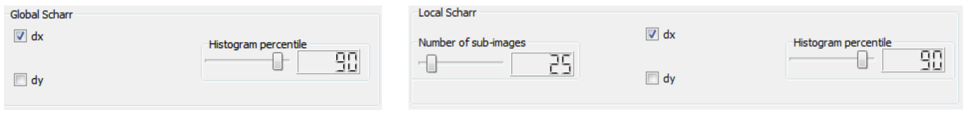


Figure 32: Global Scharr segmentation parameters (left) and Local Scharr segmentation parameters (right) , System 1

3.2.2.3. Segmentation using Otsu's method

Using Otsu's method there really are not all that many parameters to set. In a real environment it is believed that light gradients either from lighting or from the sun will affect the segmentation. There the option of splitting the segmentation process into sub-images was added. The sub-image splitting is believed to be vital for good performance. This however is a doubled edged sword, since Otsu's method always finds background and foreground it can separate an image into foreground and background when there really isn't a reason to do it. For example if a sub-image only contains a part of the image that is inside a net mask it will segment some of the pixels as foreground and some as background. The foreground pixels wrongly added here may be a hindrance to further processing of the image. It can also decide that the sea cage is foreground in one sub-image and that it is background in another sub-image. How this approach will behave under realistic conditions is a question to be answered during testing.

In Figure 33 the parameter box for the local Otsu method is shown. For the global Otsu there are no parameters to set.

Figure 33: Local Otsu segmentation parameters, System 1

There was also added the option to use the Sobel and Scharr edge detectors to enhance Otsu's method as described in 2.2.5 Using edge detection to enhance Otsu's method. This method however did not work as well as using the strong edges found directly for segmentation and will therefore be removed from the system.

3.2.2.4. Segmentation using simple thresholding

With this segmentation method the pixels are segmented into foreground and background depending on if they are above or below the adjustable threshold parameter.

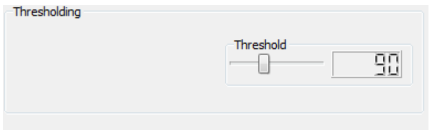


Figure 34: Simple Thresholding segmentation parameters, System 1

3.2.2.5. Segmentation using adaptive thresholding

With this segmentation method the user can choose between the Gaussian or mean adaptive method. In addition to choose the adaptive method the user can change the kernel size that is used to calculate a threshold value. The user can also change C, which is the value that gets subtracted from the weighted sum (if Gaussian) or mean of the pixels neighborhood when calculating the threshold value.

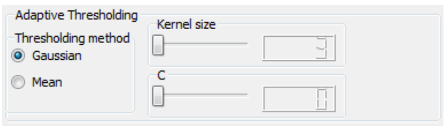


Figure 35: Adaptive Thresholding segmentation parameters, System 1

3.2.3. Segmentation enhancing functions

To make the segmentation results better a couple of functions were made that can be switched on if needed.

Morphological dilation may be needed because it is believed that especially the segmentation methods using edge detection will not create closed contours. The system will therefore need to thicken the lines found to recreate the net masks correctly. If chosen the dilation will occur after segmentation. The thinning and spurring functions used during the prototyping were dropped since they altered the shape of the sea cage masks and did not work well with dilation.

Gaussian smoothing may be needed if there is noise in the image or if the segmentation methods create noise in the image. If chosen the Gaussian smoothing operation will occur before segmentation since the Gaussian smoothing will smooth the segmentation result which will not create the binary image wanted for further processing if used after segmentation.

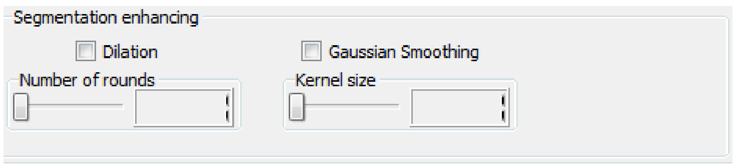


Figure 36: Segmentation enhancing functions,, System 1

3.2.4. Program modes

At this point the system has three different modes. A cage damage mode which detects damaged areas in the sea cage, a growth detection mode which detects how much growths there are on the sea-cage and a “no-mode” which only display the color space image and segmentation image.

3.2.4.1. Cage damage detection

To find damages in the net an algorithm have been developed based on the algorithm described in 3.1.3 that works like this:

1. Search for background pixels, when one is found start region growing from that pixel.
2. Grow region and paint each region with a different shade of gray. If a region stretches all the way from the top to bottom or from left to right in the image the region is assumed to be outside of the sea cage test panel. As such it is deemed invalid and the area size is set to zero.
3. If area is valid and over 500 pixels set the area and coordinates in a list
4. Sort the list after area size, if there is any unusually large areas mark them on the image

In Figure 37 an example image from System 1 using region growing is shown. As can be seen each of the different regions are marked by a different shade of gray and unusually large areas are marked with a circle. The big area in the middle are not marked as intended seeing as that region contains pixels from the top and bottom row. There are 4 gray levels used to mark individual regions. Point 3 and 4 are at this point in time based on hopeful unqualified guesses and. The 500 pixel area and what counts as a suspect area have to be further investigated when real data is available from the field test.

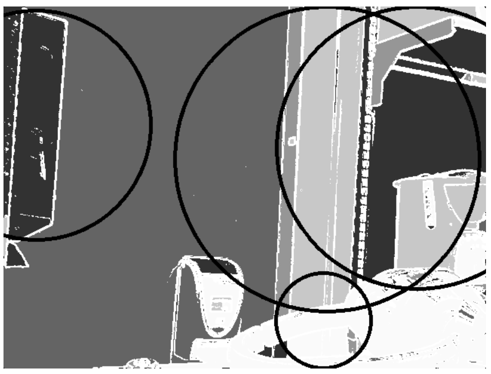


Figure 37: Example image after region growing

3.2.4.2. Growth detection

For growth detection an algorithm based on comparison between the video stream and a reference image was developed. The user presses a button to capture an image of a clean part of the net and when the system runs in Growth detection mode the algorithm compares the number of foreground pixels in the video frame and the reference frame. Growh is found as explained in the theory chapter.

3.2.5. Real Time Considerations

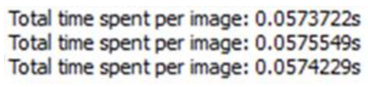
When writing software there are other things than just the complexity of the algorithms that affect the processing time used per image. How data is managed matters a great deal.

When System 1 was first made pixels in the image were accessed like this:

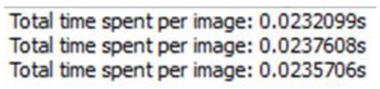
```
for(int i = 0;i< image.cols;i++)

        for(int j = 0; j<image.rows;j++)

        {

            //Do something with the pixel (j,i) using .at,
example:

             XYZ_image.at<cv::Vec3b>(j,i)[1] =
             0.299*image.at<cv::Vec3b>(j,i)[0] +
             0.587*image.at<cv::Vec3b>(j,i)[1] +
             0.114*image.at<cv::Vec3b>(j,i)[2];
        }
```

This works, but it can be made to go faster. The .at method can be used for random access of image pixels but according to (Laganière, 2011) it should never be used to scan whole images. The .at method is however easy to implement so all the loops were done this way when first developing them for System 1.

When comparing the self-written XYZ method to the one in OpenCV it was evident that changes needed to be made to the pixel accessing method. In Figure 38 the run time of the different transformations are compared.



```
Total time spent per image: 0.0573722s
Total time spent per image: 0.0575549s
Total time spent per image: 0.0574229s
```



```
Total time spent per image: 0.0232099s
Total time spent per image: 0.0237608s
Total time spent per image: 0.0235706s
```

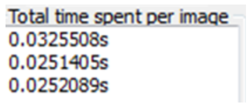
Figure 38: Run time for loading in image and using self-written XYZ transformation using .at(left) or OpenCV's XYZ transformation(right)

From (Laganière, 2011) different methods for accessing all the pixels in an image are presented. When using a pointers like this:

```
    XYZ_image.create(image.size(), image.type());
    int nl = image.rows;
    int nc = image.cols;
    if(XYZ_image.isContinuous())
    {
        nc = nc*nl;
        nl = 1; //1D array;
    }

    //loop exectued only once if the image is continious
    for(int j = 0; j<nl;j++)
    {
        uchar* data = XYZ_image.ptr<uchar>(j);
        uchar* old_data = image.ptr<uchar>(j);
        for(int i = 0; i<nc;i++)
        {
```

```
        data[3*i+1] = 0.299*old_data[3*i] +
        0.587*old_data[3*i+1]+0.114*old_data[3*i+2];//Y
        channel

      }
  {
```

The run time using the self-written XYZ transformation decreased to almost half the previous used time. The run time can be seen in Figure 39.



```
Total time spent per image
0.0325508s
0.0251405s
0.0252089s
```

Figure 39: Run time for loading image and using self-written XYZ transformation using pointers

The part of the program that eats up a lot of processing time is the segmentation algorithms and the damage detection algorithm. The damage detection algorithm uses by far the most computing time, over 0.5 seconds per video frame. At the present time it will not be made to go faster. It is mainly operations like accessing bits in the image and the like that makes the damage detection algorithm go slow and they are performed as fast as possible.

3.3.System 2

After testing System 1 with video obtained during the field test many improvements were made and there was also added some functions to try and increase the performance of the system. In the following the functions and algorithms that have been added or changed will be explained.

Added functions:

- light gradient equalizer using sub-images
- light gradient equalizer using morphological operations
- contrast enhancement using morphological operations
- Laplace edge detection for segmentation
- “No mode” to color space options to show the original input image/video

Enhanced functions

- Improve Sobel segmentation to segment dx and dy separately and then merge the result. The OpenCV function that performs both dx and dy at the same time is too sensitive when the light is strong.
- Improve Scharr and Sobel so that the dx and dy weights can be tuned by the user. To make the algorithms more robust and versatile.
- Improve Local Otsu so that if a sub-image appear opposite of its neighbors it inverts the color, black becomes white and vice versa.

- Remove lightest area color for the region growing algorithm so that the regions are easier to see.
- Improve the selection of suspect regions and damages in the region growing algorithm

Also the option of using edge detection to enhance Otsu's method was removed.

The GUI for System 2 can be seen in Figure 40. It should be noted that all segmentation methods that have local and global options in the GUI have been changed so that only one choice per segmentation method are available. Global option is chosen by setting sub-images to 1. So for example instead of choosing global or local Otsu one now just chooses Otsu and set the sub-images to the desired amount.

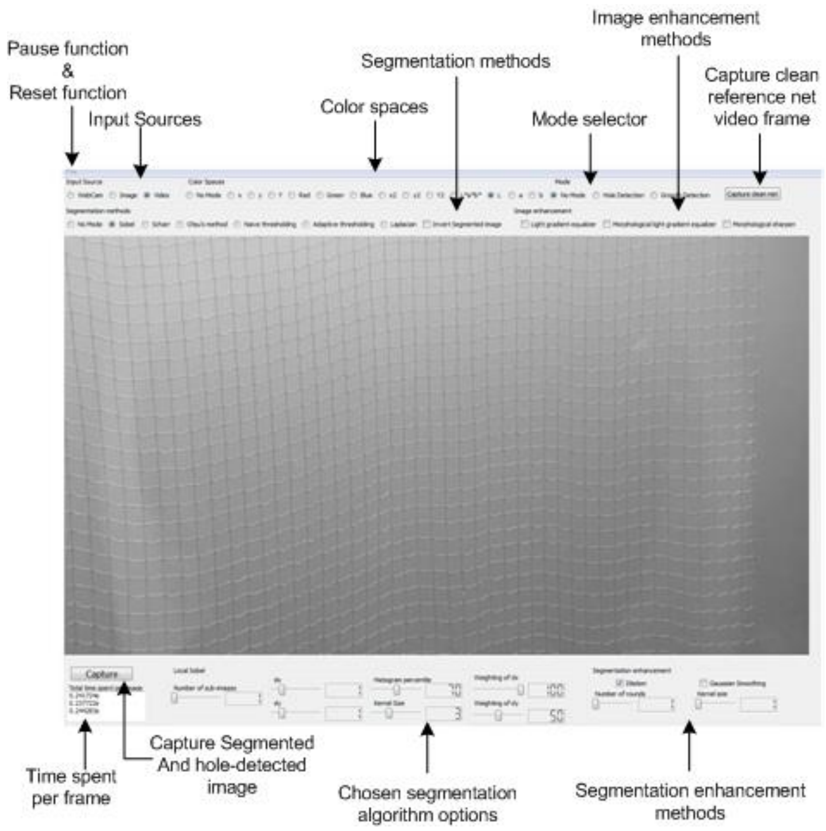

Pause function
&
Reset function
Input Sources
Color spaces
Segmentation methods
Image enhancement methods
Mode selector
Capture clean reference net video frame
Time spent per frame
Capture Segmented And hole-detected image
Chosen segmentation algorithm options
Segmentation enhancement methods

Figure 40: GUI, System 2

The user has the following options :

- Pause program and reset the program.
- Choose the input mode. The system can use web cameras, the camera used in the field test, images or videos as input.
- Choose the desired color space.
- Choose the desired segmentation method and change parameters specific to the chosen segmentation option
- Choose the desired image enhancement methods.

- Choose the desired segmentation enhancing methods.
- Select the program mode.

The user can also capture the segmented image and the hole detected image using the Capture button. The total computing time spent per video frame is shown to the bottom left.

3.3.1. Color spaces

No changes made. Although only the luminance and chromatic y color space seemed useful when used on System 1.

3.3.2. Image enhancing functions

After the field test it became clear that especially the Otsu's method would need a lot of enhancement to give anything useful. The other segmentation methods could also benefit from equalizing the highly variable light conditions experienced on the videos from the field test. Two different approaches were made to equalize the light. Also contrast enhancement functions were added to help enhance the segmentation results.

3.3.2.1. Light gradient equalizer using sub-images

This is a very easy and solution to the problem which will not completely remove the light gradients, however the algorithm will reduce the uneven light conditions. As illustrated in Figure 41 the algorithm first divides the image into twenty sub images with full column size and a twentieth row size. The average intensity of the top sub image is said to be the desired intensity and the 19 other sub images get changed so that they have the same average intensity as the top one. Then the resulting image has the same operations applied to 20 sub images with full row size and one twentieth column sizes.

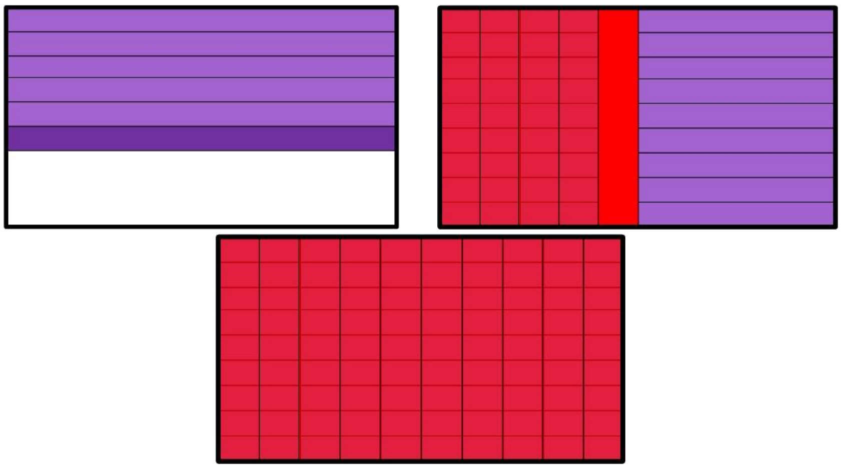


Figure 41: Light gradient equalizer process. First light equalizing the rows (top left), then light equalizing the columns (top right)

3.3.2.2. Light gradient equalizer using top hat transformation

Another approach that was implemented was using the morphological top hat transformation to remove the suns effect on the system. The top hat image is obtained using the OpenCV morhologyEX function and the resulting image is subtracted from the input image to reduce the suns effect on the sea-cage panel. It should be noted that when subtracting the top hat transformation from the original image the pixel values of the top hat image had to be halved so as not to remove too much information about the sea-cage. The wanted effect is to dampen the light horizontal lines, not to remove them completely.

A big structuring element uses a lot of computation time so after some trial and error the choice fell on a structuring element shaped as an ellipse of 5x5 elements. Although this is too small to completely remove the uneven background illumination a larger structuring element uses too much computation time to be used for a real time system. It was found during the making of this method that a structuring element with a size of 20x20 evens out the background illumination extremely well, but since it uses around 3 seconds per image it is not deemed a plausible alternative.

3.3.2.3. Contrast enhancement using morphological operations

Morphological operations can be used to enhance contrast and thereby sharpen the image. A combination of top hat and bottom hat transformation can be used to sharpen an image like shown in Equation 12.

$$enhanced\ image = image + tophat - bottomhat$$

Equation 12: Contrast enhancement using top hat and bottom hat, (Gonzalez, Woods, & Eddins, Morphological Image Processing, 2009)

This should sharpen the image if needed, it will however also sharpen the noise so if it is useful for this thesis' purpose remains to be seen.

3.3.3. Segmentation methods

The segmentation methods that seemed to perform best and the most robust during the testing of System 1 were the edge detection techniques. However a good amount of effort went into improving the Otsu's method since it is believed that it could be very useful if it works properly.

3.3.3.1. Segmentation using edge detection, Sobel

After testing the Sobel approach on the captured video it became evident that the method did not segment vertical lines good enough. In System 1 dx and dy were segmented at the same time using the OpenCV Sobel function. This approach weights dx and dy the same which does not perform adequately when effects like the strong sun makes the horizontal lines stand out more strongly than the vertical lines. So to enhance the Sobel algorithm dx and dy were calculated separately and then merged together with weights that can be tuned by the user. In Figure 42 the performance of the Sobel functions from System 1 and 2 are compared. As can be seen the performance of System 2 is a lot better than System 1.

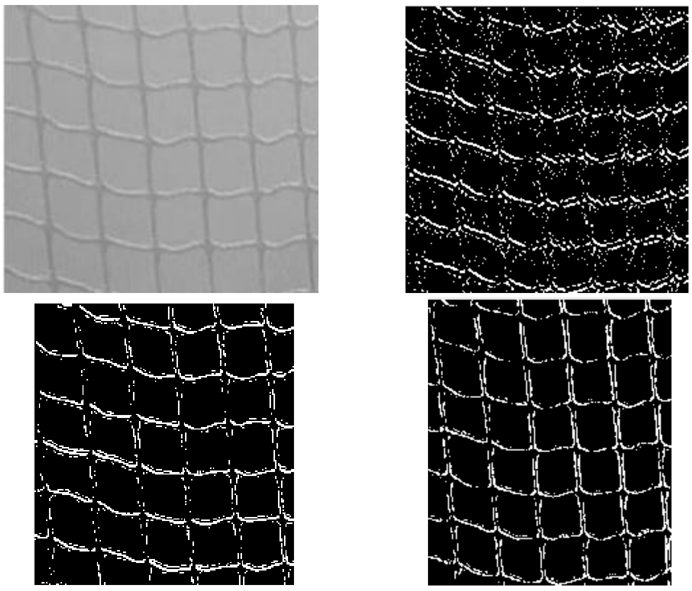
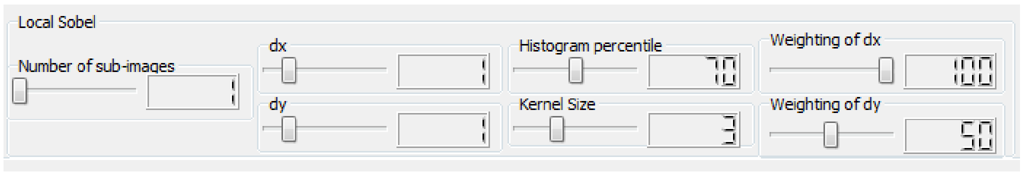


Figure 42: Improved Sobel Function: reference video, Y color space(top left), System 1(top right), System 2 with 50/50 dx/dy weights(bottom left), System 2 with 79/29 dx/dy weights(bottom right)

The new options box for the Sobel function now looks like in Figure 43. Although the dx and dy could just as well have been removed and made static as first derivatives since that works best they are kept as they are for now.

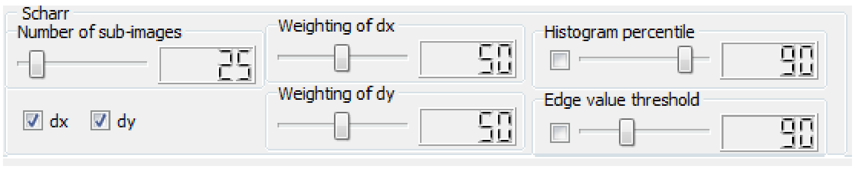

Local Sobel

Number of sub-images	1
dx	1
dy	1
Histogram percentile	70
Kernel Size	3
Weighting of dx	100
Weighting of dy	50

Figure 43: Sobel parameters, System 2

3.3.3.2. Segmentation using edge detection, Scharr

To make the Scharr algorithm more robust to sunlight weights were added so that the user can weigh the dx and dy edge detected images when they are merged. In System 1 the dx and dy results were merged with equal weight. As with the Sobel algorithm this is done to increase the performance in though light conditions. The Scharr now also have the option of using a threshold value to determine which edges are strong enough to be considered as foreground. This is meant to be used for growth detection. The new parameters box for the Scharr parameters can be seen in Figure 44.


Scharr

Number of sub-images	25
✓ dx	✓ dy
Weighting of dx	50
Weighting of dy	50
Histogram percentile	☐ 90
Edge value threshold	☐ 90

Figure 44: Scharr parameters, System 2

3.3.3.3. *Segmentation using edge detection, Laplace*

In System 2 segmentation using a Laplace operator was added. The Laplace operator is not direction dependent which is a good quality, making the system potentially more robust. However the Laplace operator generates more noise than the Sobel and Scharr operators. The Gaussian smoothing filter will be needed to remove the extra noise.

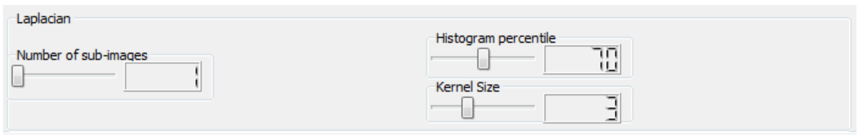

Laplacian
Number of sub-images 1
Histogram percentile 70
Kernel Size 3

Figure 45: Laplace parameters, System 2

3.3.3.4. *Segmentation using Otsu's method*

Since Otsu's method did not perform so good during the testing of System 1 some more intelligence have been added to ensure that the same structures will be segmented as foreground and background throughout the image.

When using multiple sub-images the algorithm now checks an overlapping region between a sub-image and the sub-image above it. The overlapping region is 5 rows big. A pixel-by-pixel search is made of that overlapping region and if the two sub-images are found to have opposite pixel values the sub-image in question is inverted.

3.3.4. Program modes

3.3.4.1. *Net damage detection*

When making System 1 there were not much information available about how large the areas inside net masks would be. This led to a wrong threshold being set as an acceptable region size. The guesstimate of what a damaged area was also needed to be remade using the new data. Using the videos from the field test a MATLAB® program were developed which makes a histogram of the regions found in the region growing algorithm. This MATLAB® program can be found in a subfolder called "Histogram for region growing" in the "Prototype files" folder in the enclosed zip-file.

Using this program on a list of regions from System 2 with the following settings:

Color space: L	Sub-images: 1	Percentile: 70	Dilation: 1
dx: 1	dx weight: 1	dy: 1	dy weight: 1

To get a seemingly perfect segmentation the video frame were cut so that the bottom right part of the image where the image quality wasn't good enough is cut away. This was done in order to see the distribution area size distribution in a perfect case. The video frame can be seen in Figure 46.

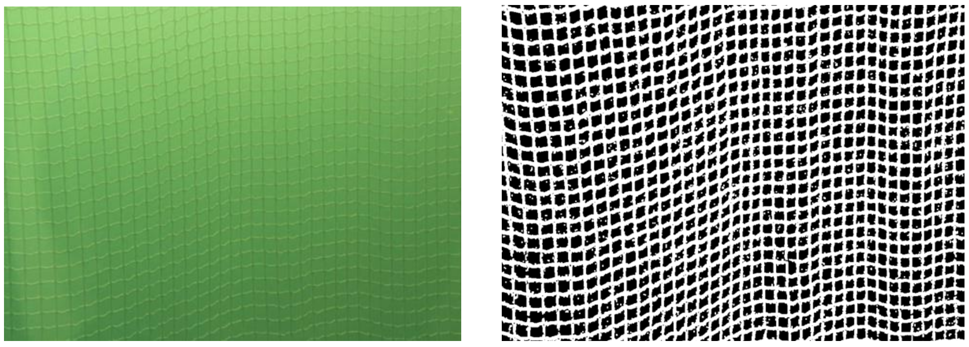


Figure 46: Reference frame used for observing area distribution(left), with corresponding segmentation result(right)

From the list of regions found from the reference frame the MATLAB® program made histograms that looks like the one in in Figure 47.

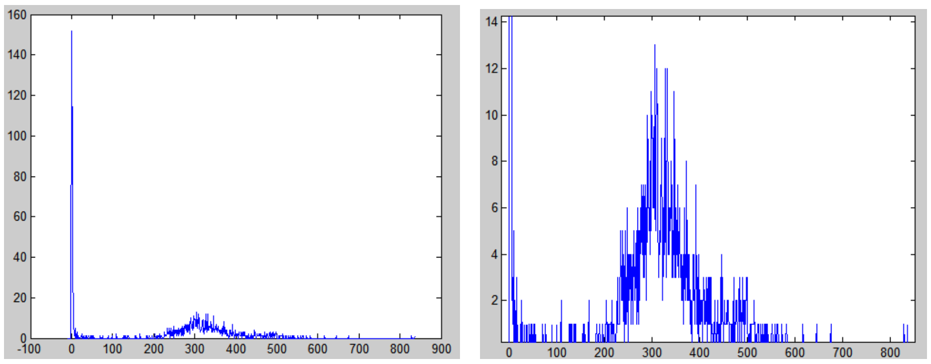


Figure 47: Histogram from region growing the reference frame

The histograms were dominated by low numbers like ones twos and threes, other than that masks appear to be mostly around 300 pixels. In Figure 48 the reason for the many small areas can be seen. The segmentation processes that uses edges has a hard time in filling in the areas where the horizontal and vertical lines cross each other. Because of this areas under 20 pixels in size where considered as noise and not taken into acount when analyzing the area size distribution.

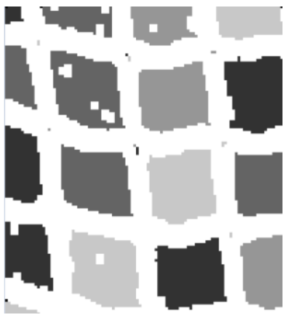


Figure 48: Small areas found inside the net

The distribution of areas around 300 pixels strecthes all the way up to 600 pixels with even some areas found to be over 800 pixels. Using the edge detection methods for segmenttion there is the off chance that a mask is not segmenetd properly and there is a gap in a mask that should be whole. This is especially likely too happen when the image quality in an area is poor. After seeing several distributions and segmentation results it became evident that it is not uncommon if two or even three masks were considered one area due to incomplete segmentation. For now small suspect areas will be marked with a black ring while big areas will be marked with a light grey ring.

In summary the new damage detection algorithm were implemented as follows:

1. Search for background pixels, when one is found start region growing from that pixel.
2. Grow region and paint each region with a different shade of gray, if region stretches all the way from the top to bottom or from left to right in the image the area is assumed invalid and set to zero. This due to the use of small sea-cage panels during the field test so some of the areas in the image does not contain sea cage.
3. Find the region size that occur the most
4. If a region is between 2-4 times that of the most occurring value the region is marked as suspect with a black ring. If the region is over 4 times as large as the most occurring region the region is considered a hole and marked with a light grey ring.

The reason for the marking the suspect areas black will be explained in the discussion chapter.

Regions that are under 20 pixels will not be counted since they are not net masks but simply small areas between edges or areas at the edge of the image. If the segmentation result is post-processed with a salt and pepper remover before running the damage detection algorithm these small areas should pretty much be gone.

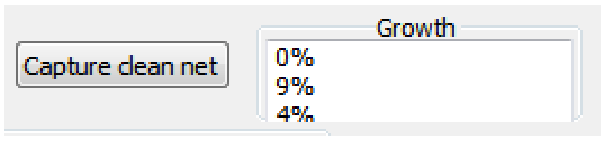
3.3.4.2. Growth detection

In System 1 the segmented image that was thought used to detect growth was a percentile thresholded Sobel or Scharr segmented image. When testing the method with videos from the field test the performance of the method was found to not be good enough. It was believed that even after enhancing the Sobel and Scharr segmentation methods based on findings from the field test a different approach should be studied for growth detection.

To try and use a slightly different approach the Scharr function now has the option of using a threshold on the edge strength value to segment the image into foreground pixels and background pixels. This way not only a given percentile of edges are kept but rather all edges that are strong enough. For this to work there must be edges within the growths so the Scharr function is able to see them.

Originally Otsu’s method was believed to be able to find growths better than the edge detection methods, however with the bad performance of the method during testing of system 1 it will not be implemented for this if the performance when testing the new and enhanced Otsu’s method doesn’t look more promising.

Also in System 2 a text box have been added next to the “Capture clean net” button to show the percentage growth as shown in Figure 49.

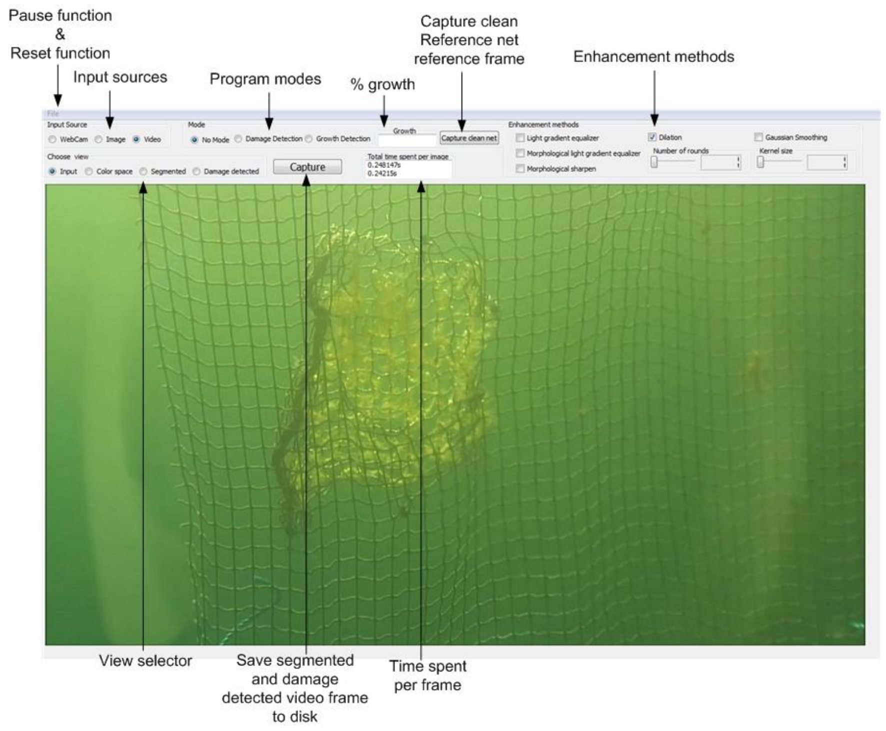


```
0%
9%
4%
```

Figure 49: Growth text box

3.4. Final system

After testing System 2 I had time to do a final iteration of the software development cycle. This final system should be more user friendly compared to the others. Here most of the choices are done for the user based on the best working algorithms from System 2. A morphological salt and pepper remover were added and the damage detection algorithm was made more robust other than that the functionality is basically the same as for System 2. In this GUI only one frame is shown at the time, the others had for example both the segmented and the damage detected video frame shown, here the user chooses which image is to be displayed.



Pause function & Reset function
Input sources
Program modes
% growth
Capture clean Reference net reference frame
Enhancement methods
File
Capture clean net
Capture
View selector
Save segmented and damage detected video frame to disk
Time spent per frame

Input Source: WebCam Image [✓ Video]

Mode: [✓ No Mode] Damage Detection Growth Detection

Growth

Choose view: [✓ Input] Color space Segmented Damage detected

Total time spent per image
0.248147s
0.24215s

Enhancement methods: Light gradient equalizer, Morphological light gradient equalizer, Morphological sharpen, [✓ Dilation], Gaussian Smoothing

Number of rounds

Kernel size

Figure 50: GUI, final system

In the No mode and the damage detection mode segmentation is performed using the Sobel algorithm with a histogram percentile of 70. If growth detection is chosen the segmentation is performed using the Scharr algorithm with edge strength thresholding as explained in 3.3.4.2. The threshold is set to 70.

3.4.1. Using morphological opening to remove salt and pepper noise

The occurrence of white pixels that are not part of the sea cage panel and black pixels inside the segmented sea cage panel as shown in Figure 48 led to the conclusion that a so-called salt and pepper remover had to be developed.

To remove salt and pepper morphological opening was used with a 5-by-5 structuring element thereby removing salt and pepper noise that are smaller than the structuring element.

First salt was removed using the OpenCV morphologyEx function, and then the result was inverted so that the pepper noise became white pixels and then a new round of opening was used. Finally the end result was inverted back. The result of these operations should clear out most of the salt and pepper noise.

3.5. Further enhancing the damage detection algorithm

In System 2 the damage detection algorithm looked at the region size that has the highest number of occurrences to determine what the normal region inside an undamaged mask most likely is. After testing System 2 this seemed to not be robust enough and therefore it was made more robust by seeking the region of most occurrences instead of seeking the most occurring value.

In the final system the program sorts a list of the recurrences of region sizes. Then for each region size the occurrence of that region size together with the occurrences of the five bigger and five smaller region sizes is added. The region size with the most number of occurrences found this way is considered to be the region size which represents a normal undamaged sea cage mask in the image.

This way of finding the normal undamaged mask region is based on the histograms both in Figure 47 and other similar ones found when testing area sizes. Since the undamaged mask sizes are clustered like a bell shape the correct region size should be found. Random spikes of other region sizes will now no longer make the system believe that the spike is the correct region size.

4. Field testing

In the start of the thesis it was believed that camera and lighting could be borrowed to use in the field test. This however turned out not to be the case. The hardware could not be borrowed and information about the lights was never given. This led to the need of acquiring the camera and lights and making it work. Thankfully I did not have to do this by myself. Morten Engelhardt Olsen a fellow graduate worked on a similar thesis and also needed the hardware. So we set out to obtain the hardware and to make it work together. I had originally planned to perform testing in the start of April at the latest but due to the delay caused by have to acquire the hardware ourselves and getting the hardware to work the testing wasn't performed before the middle of May.

4.1.Test set-up

During the work on the thesis equipment were acquired to send data from the camera using either HD-SDI or Gigabit Ethernet. In Figure 51 a schematic over the two different ways to connect the system is shown. One can either use HD-SDI to transmit the video feed using a separate cabling for the control signals to the camera. If one uses the Gigabit Ethernet card one can use an Ethernet cable for both video and control signals. The HD-SDI has a much higher video quality seeing as the video frames does not get compressed before transmission.

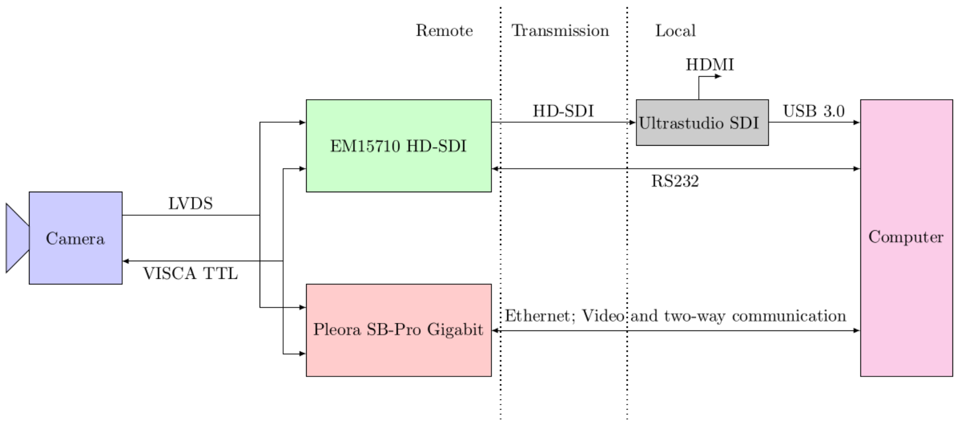


Figure 51: Test system schematic (Olsen, 2013)

4.1.1. Camera

In the testing a FCB-H11 camera from Sony was used. A datasheet of the camera can be found in the attached files. The camera is controlled via computer software communicating over the serial bus. The computer software is called FCB-HD and is enclosed in the attached files. This camera is the same as the ones used by Argus Remote Systems in their ROVs. Some of its features include:

- Approx. two million effective pixels
- 16:9 aspect ratio
- HD and SD video format
- Day/Night function
- 120x zoom ratio (10x optical and 12x digital zoom)

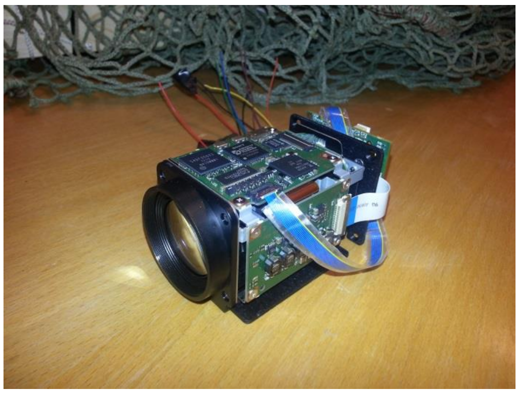


Figure 52: Sony FCB-H11 camera

The camera is controlled by Sony's VISCA protocol which is based on RS232. Using a basic program provided by Sony (in "FCB-HD Setup" folder on the DVD) the user can control a wide variety of parameters such as:

- Power on/off
- Set Baud rate
- Set video output mode
- Change focus and zoom
- Change shutter speed, gain and iris
- Effects such as negart and Black&White

The VISCA control can also be written into the program if needed, an open source C library is available at http://sourceforge.net/projects/libvisca.

4.1.2. HD-SDI interface module

To combine high quality video transfer with a long range, HD-SDI was chosen as the way to transfer the video signal from the camera. HD-SDI is the common term for a standard named SMPTE 292M (The Society of Motion Picture and Television Engineers). It uses coaxial cables to transport uncompressed digital video and has a range of about 300 meters. HD-SDI is mainly used in TV studios but it is an ideal standard for our use, providing sharp images with zero compression of the data.

To interface the camera an iShot FCB HD-SDI Interface Module from InterTest were used. This interface module acquires digital images directly from the camera in real-time. Instead of re-digitizing the analog output from the camera this module directly connects to the cameras digital output. The datasheet for the iShot can be found in the attached files.

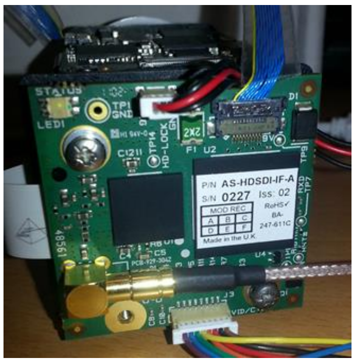


Figure 53: iShot FCB HD-SDI Interface Module

4.1.3. Gigabit Ethernet interface

In the later stages of the thesis a Gigabit Ethernet interface module were also acquired. While we didn't have the time to implement and test it on our systems it can be a nice addition to the department's hardware portfolio.

The iPORT SB-Pro IP engine can be used to make the Sony FCB-H11 into a GigE Vision® compliant camera. Using CAT5e/6 cabling it can transmit a distance up to 100 meters, and with switches this range can increase further.

The image quality from this solution would not be the same as from the HD-SDI solution, but we tried to hook it up and get it to work. Unfortunately it turned out that all the components needed to connect the module were not sent from the supplier, we therefore did not have the time to get it up and running.

4.1.4. Frame grabber

To connect the camera to a computer an UltraStudio SDI from Blackmagic Design was used. This unit takes in a HD-SDI signal and passes it to a computer via the USB 3.0 port. Although it is mostly found in TV-studios, it is perfect for our purposes since it can be connected to almost any computer that has a USB 3.0 port. This unit can also send out the HD-SDI again this can be used to daisy chain multiple UltraStudio SDI together. In addition the UltraStudio SDI has an HDMI port. With the HDMI port a monitor can be plugged in and the video stream can be viewed directly from the camera. The data cannot be processed from the HDMI port but it is an easy way to see the raw video stream.

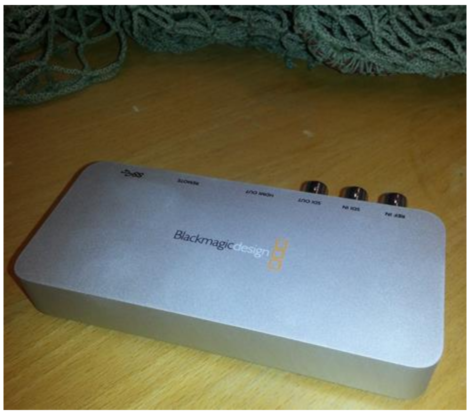


Figure 54: Blackmagic Design UltraStudio SDI

Using the UltraStudio SDI box the video could either be run using their software or processed directly in the software developed during this thesis. When using their software the video stream can be recorded and saved to disk as raw data. This consumes a lot of disk space so these videos needs to be converted to a more compressed format to be readily shared. The videos obtained in the field test were converted to h264 compressed video files.

4.1.5. Waterproof camera housing

To withstand being submersed a waterproof camera housing was made by Terje Haugen at the departments mechanical workshop, Figure 55. It consists of a tube of clear plastic with a plug in either end. The camera is mounted on a steel bracket and sees through one end of the tube while the cables come in from the other end. Inserted into the housing there is a 25m long coaxial cable transmitting the HD-SDI signal and a 50m long cat5e. cable supplying power and control signals. The cables did of course not need to be so long during our testing but operation the equipment with long cables was a test in itself to see that it is possible to operate the camera from such lengths.

The housing has a sturdy white plastic bracket to fasten it to the test rig. This can be clearly seen in the right image in Figure 55. The bracket was screwed into a plate in the test rig and submerged as shown in Figure 62.

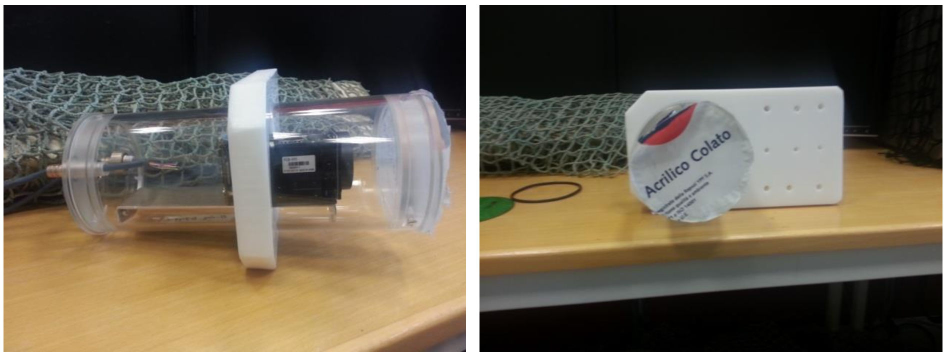


Figure 55: Camera housing

The housing has been tested down to 2 meters, but it should be tight even further down. Since the camera only was submerged down to 1.5-2 meters during the field test the housing has submerged lower to see how far it can go.

4.1.6. Lights

Sufficient lighting is necessary to obtain sharp images. For our test two 27W LED lights were used. They were supplied with 12V led-drivers so they can be connected to any 12V DC source. The lights are originally made for lighting up under boats and can be surface-mounted on ship hulls. This design made them well suited for our purpose. Each light were fastened to a 2 meter long pole so that they could be positioned as needed.

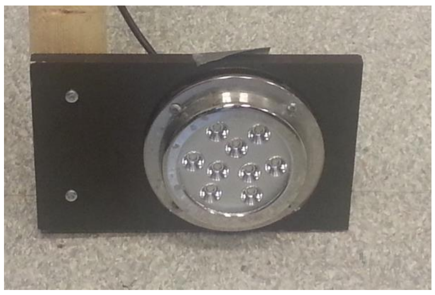


Figure 56: 27W LED light

4.1.7. Hardware problems

After obtaining the camera and all the components needed to get video on the computer the setup unfortunately didn’t work. Some tweaking and fixing is expected when connecting together something new but all-in-all it took four weeks to get the system to work properly. The components used were chosen for their plug-and-play ability. In a perfect world the hardware should have worked together when properly connected together, this however is far from a perfect world. When plug-and-play doesn’t work it is really hard to figure out the error source. Combine that with the fact that the UltraStudio SDI doesn’t give any indication of whether it sends video through the USB 3.0 port or even gets viable video in through the SDI port. In addition if the UltraStudio SDI is connected to the wrong USB 3.0 controller or a controller with old drivers it simply doesn’t work.

After many weeks of rigorous testing we sought the help from Trond Viggo Melum at NRK. Using NRK’s equipment we got to test that every individual piece of hardware worked properly. The error was revealed when trying to get the camera to work. The SDI output from the camera was plugged into Melums laptop via one of the UltraStudio SDI boxes. However it didn’t work unless we plugged in the combined coax into another UltraStudio SDI box that was hooked up to our laptop. The hardware configuration then looked like in Figure 57. Why the coaxial cable from the combined coax and power cable had to be inserted into the UltraStudio SDI was a mystery. It turned out that there were ~0Ω between the minus power cord and the shield on the coax. So for the camera to get the required stable power supply the coax shield had to be plugged into something stable. This led to us not getting video even after we switched out the coax from the combined cable because it still was being used to supply power to the camera. After replacing this defect cable with a new coax and running the power together with the serial data through a Cat 5e cable the system worked as it should.

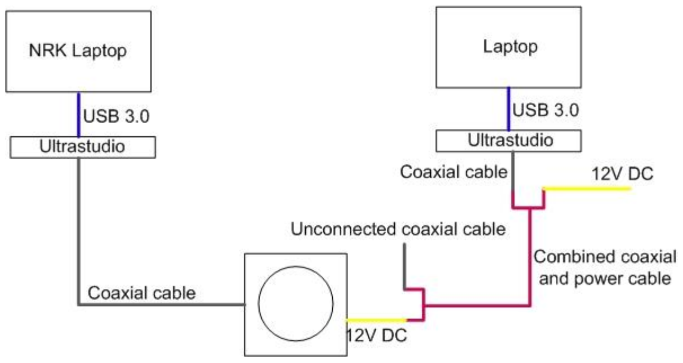


Figure 57: First hardware set-up that worked

4.1.8. Sea-cage panels

Five panels were made as shown in Figure 58 and Figure 59. The idea was to pass the panels one at the time across the camera field of vision and use the developed software to detect the structures in the panels. Either those structures are holes, tears, growth or double sea cage structures. For the growing test a smaller net that simulates growing was attached over a hole of the top left panel in Figure 58 during the growing test. The growing were made to look like *Ectopleura larynx* by tying yarn to the net and letting it go through a few cycles of soaking in water and drying. The end result seems to be close enough for our purposes as can be seen in Figure 60.

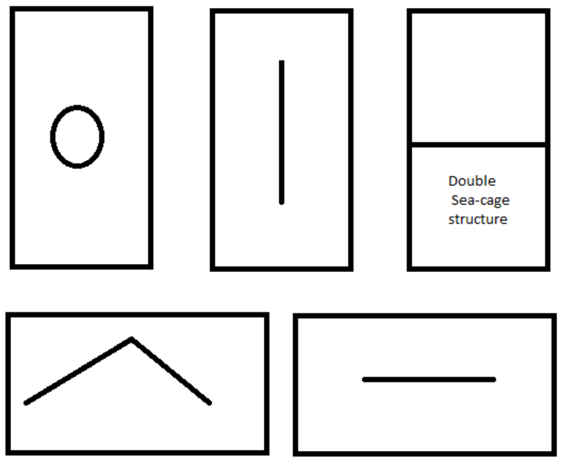


Figure 58: Sea cage panels. Top left: Panel with hole. Top middle: Panel with vertical tear. Top right: Panel with double net. Bottom left: Panel with L-tear. Bottom right: Horizontal tear.

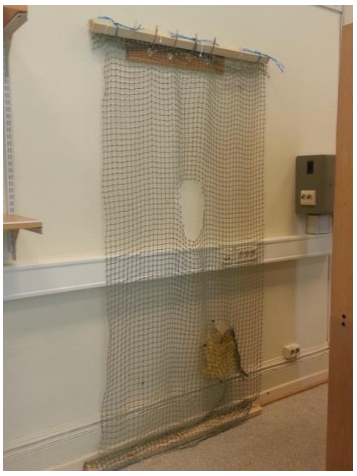


Figure 59: Actual sea cage panel used

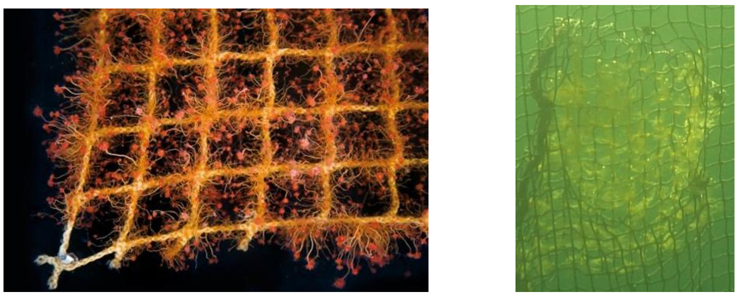


Figure 60: Ectopleural larynx growth. Actual (Benjaminsen, 2012) (left) and simulated (right)

4.1.9. Test Rig

The test rig illustrated in Figure 62 was borrowed from Kevin Frank at SINTEF Fiskeri og Havbruk. It consists of scaffolding pieces put together, a winch and a pulley system. With this rig the net can be moved up and down with the winch and equipment like the camera can be placed with various distances from the net, approximately from one up to two meters.

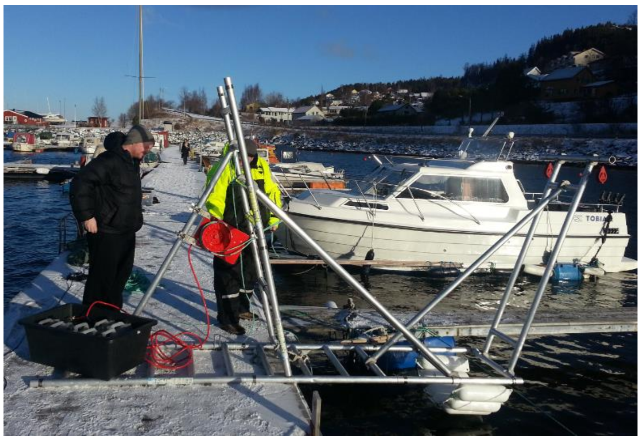


Figure 61: Test rig used in February

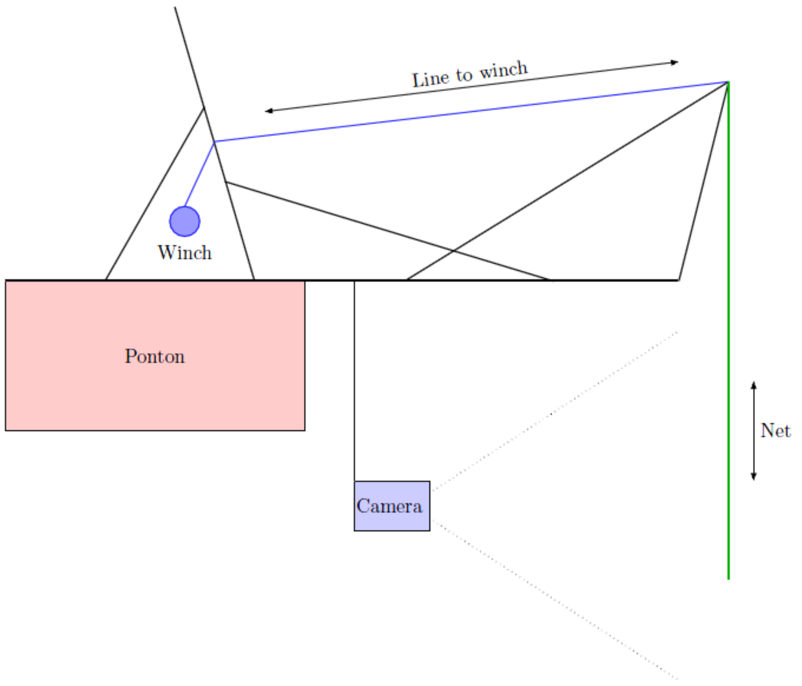


Figure 62: Test rig illustration, black lines is the scaffolding, blue lines is ropes and green is the sea cage panel, (Olsen, 2013)

4.2.Field test

The field test was performed in Hommelvik, Sør-Trøndelag, at a port belonging to Malvik Båtforening. Due to the many delays the test was performed in mid-May as opposed to the start of April as was originally wanted. The purpose of this field test was to acquire realistic data to test the computer vision system. It was a clear sky and it was very sunny. It fast became evident that the strong sun made our lights unnecessary. With the strong sun we got some troublesome effects in the videos like a vertical light gradient and also reflection of the sunlight on the viewport on the camera housing. During the testing a strong wind also blew up so we got some turbidity in the sea as well. All in all we encountered seemingly realistic environmental conditions during the test.

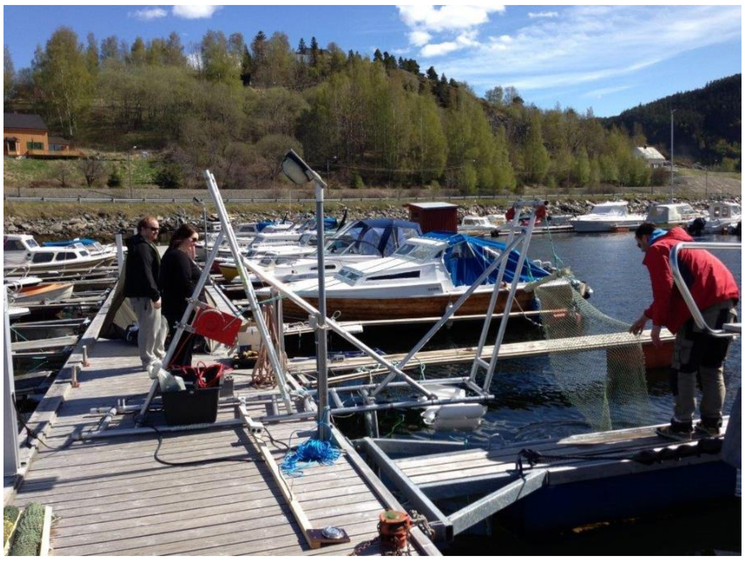


Figure 63: Field test

During the field test the camera were mounted from the rig and down into the water approximately 1.5meters from the sea cage panel. The camera sent the HD-SDI signals through a 25meter long coaxial cable to the Blackmagic UltraStudio SDI box and from there via USB3.0 to a laptop. The camera was controlled via the same laptop. A 50meter long cat.5e cable were used both for the control signals and to supply the camera with power. The setup of the hardware is shown in Figure 51.

The sea cage panels were fastened as shown in Figure 63 and were lowered into the sea and raised again using the winch to simulate motion.

During the field test an attempt to use System 1 was tried, however it proved unsuitable to use on the small laptop screen as the video frames was too large. The system worked fined with the camera, the screen of the laptop was just too small so that it was hard to see the results to be of any real use at that time. Instead videos were recorded during the field test which the systems can be tested on.

The videos taken were captured as raw data and therefore took up a lot of memory. In some videos there are frames missing because the laptop used couldn’t save the frames to disk fast enough. This is something that should be addressed when the test set-up is in use next time. On the attached zip-file h264 compressed videos are found from some of the tests in the “Videos from field test” folder.

4.3.Further motion testing

After testing the videos obtained during the field test it became evident that there is a big problem concerning motion. In Figure 64 it can be seen that in the color space image there is no visible horizontal lines in the masks when the net is moved vertically.

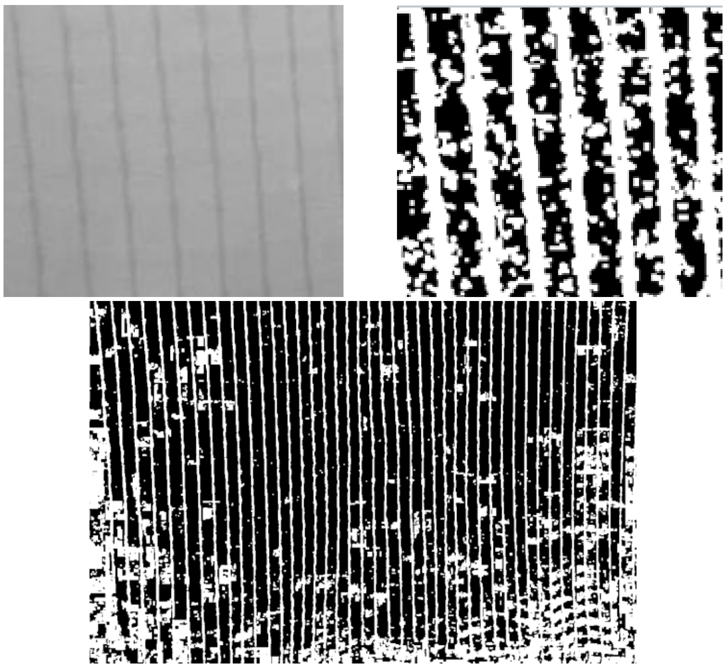


Figure 64: Scharr segmentation on moving reference net, y color space (left), Scharr segmented image (right) and Otsu segmented (bottom).

In the videos recorded during the field testing the horizontal mask lines is blurry and even disappears during vertical movement. When this effect was discovered it was believed that it was due to a too large shutter speed. With a too large shutter speed during capturing the horizontal lines in the masks moved across the frame while the shutter was open. This causes an effect called motion blurring. While motion blurring is an effect often desired to make visual effects in images or to make motion in video games look more realistic it is highly unwanted when the video feed is used in computer vision algorithms.

To avoid the problem of motion blurring one can either develop software algorithms that help reduce the problem, this however takes time to make and have to be used on every frame leading to longer processing time for each processed frame. Another way to solve this is to simply decrease the shutter time on the camera, given that the camera can be set to a lower speed. Decreasing the shutter time will make the video feed look less natural and jerkier because there is no visual indicator to how the movement is occurring. This will however lead to better images for the computer vision algorithms.

During the field test the camera was set to “Full Auto” exposure mode, this mode has a fixed shutter speed of 1/50 which seemingly was too large. However the camera has a total of 21 different shutter speeds available, ranging from 1/2 second to $1/10000$ seconds, (Sony, 2008). This gives a wide range of shutter speeds to choose from. When the shutter speed is reduced one needs a higher aperture size or stronger lights to get the amount of light needed to get a clear image during the open time of the shutter. The camera automatically sets iris and gain according to the brightness of the frame, but if that is enough or not have to be tested.

To test what shutter speed is needed to avoid motion blurring the sea cage panels were put up against a wall. The camera was moved while filming the sea cage panel from a distance of 1 meter and 3 meters. During these tests the camera was set to wide mode, i.e. no zoom. It is important to note that since these tests were performed in the office under fluorescent lights there are shadows from the net on the wall that degrades the segmentation result. In the right part of the shadows made that area murky and hard to segment. It was therefore important to see clear evidence that missing lines were due to motion and not to incomplete segmentation. The images taken from the recording made from 1 meter are therefore exclusively of the top left part of the net where the conditions were best.

During these test the camera was operated "free-hand" and moved vertically, horizontally and in a zigzag pattern, with an approximate speed of 0.5m/s.

First the camera was set to Full Auto which gives a shutter speed of 1/50. Here the motion blurring effects are strong as can be seen in Figure 65.

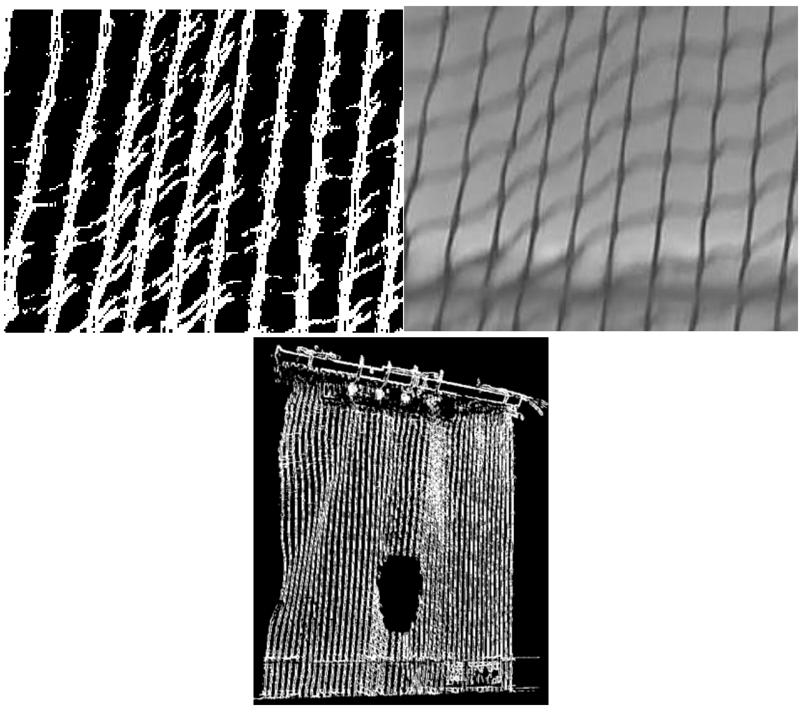


Figure 65: Net filmed with shutter speed 1/50s: 1m distance vertical motion(top), 3m distance vertical motion(bottom)

When using a shutter speed of 1/300s the camera captures the motion adequately from 1meter, however from 3 meters it had problems. As shown in the bottom image in Figure 66 the system "looses" the vertical lines during horizontal motion.

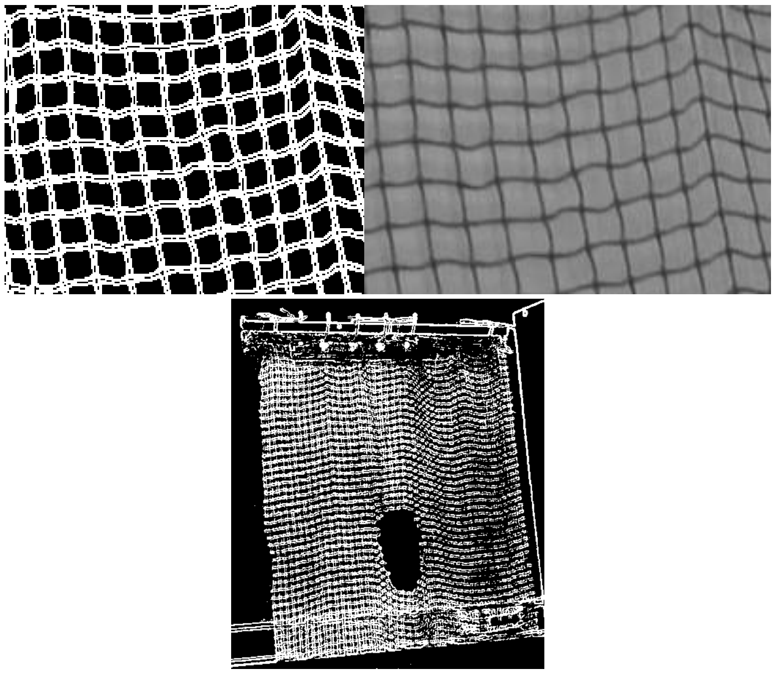


Figure 66: Net filmed with shutter speed 1/300s: 1m distance vertical motion(top), 3m distance horizontal motion(bottom)

When using a shutter speed of 1/600s the camera seems to capture motion well from both 1 and 3 meters as shown in Figure 67.

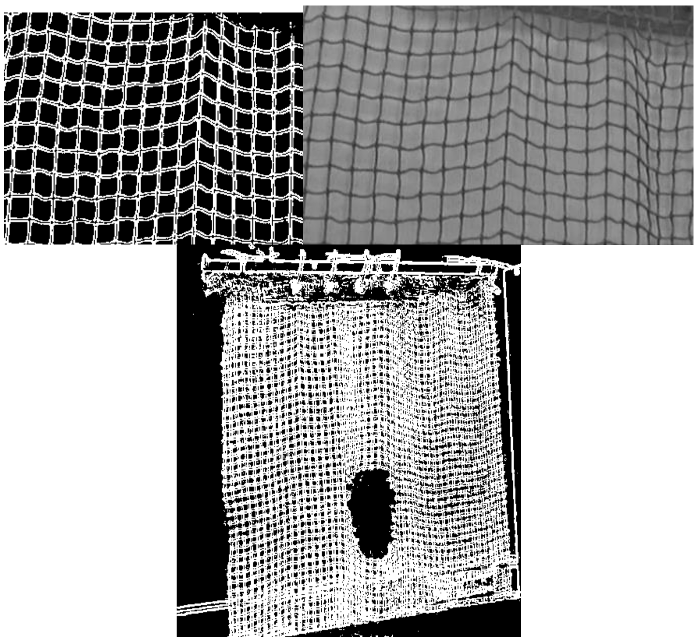


Figure 67: Net filmed with shutter speed 1/600s: 1m distance vertical motion(top), 3m distance horizontal motion(bottom)

Using a shutter speed of 1/1250s the performance should have been better than for 1/600 however that is not the case. As can be seen in the bottom images of Figure 68 the vertical lines are not clearly visible along a good portion of the sea cage panel. This is due to the light condition not being good enough for this low shutter speed. Even with vertical movement where the vertical lines should stand strongly out they are missing. The horizontal lines however are strong independent of the motion of the camera because of how the light illuminates them.

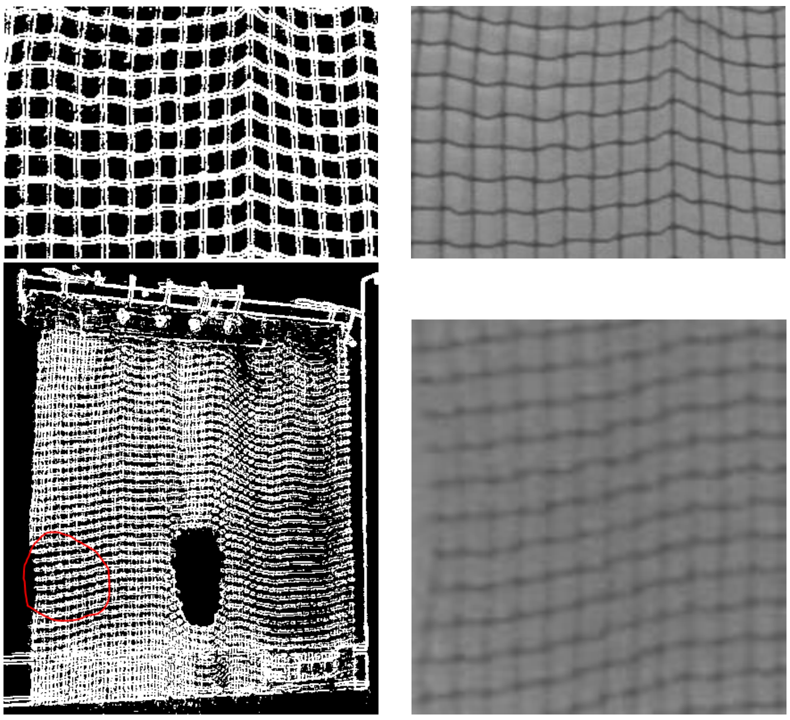


Figure 68: Net filmed with shutter speed 1/1250s: 1m distance vertical motion(top), 3m distance vertical motion(bottom)

To show the effects of too little illumination compared to shutter speed shutter speeds of 1/6000s and 1/10000s were also tested. With so low shutter speeds the camera tries to improve the light in the image so much that it introduces a lot of noise. In Figure 69 a shutter speed of 1/6000s were used. The top image is when the camera is stationary while the bottom images are when moving the camera diagonally downward and to the left. As can be seen most of the net disappears when moving the camera.

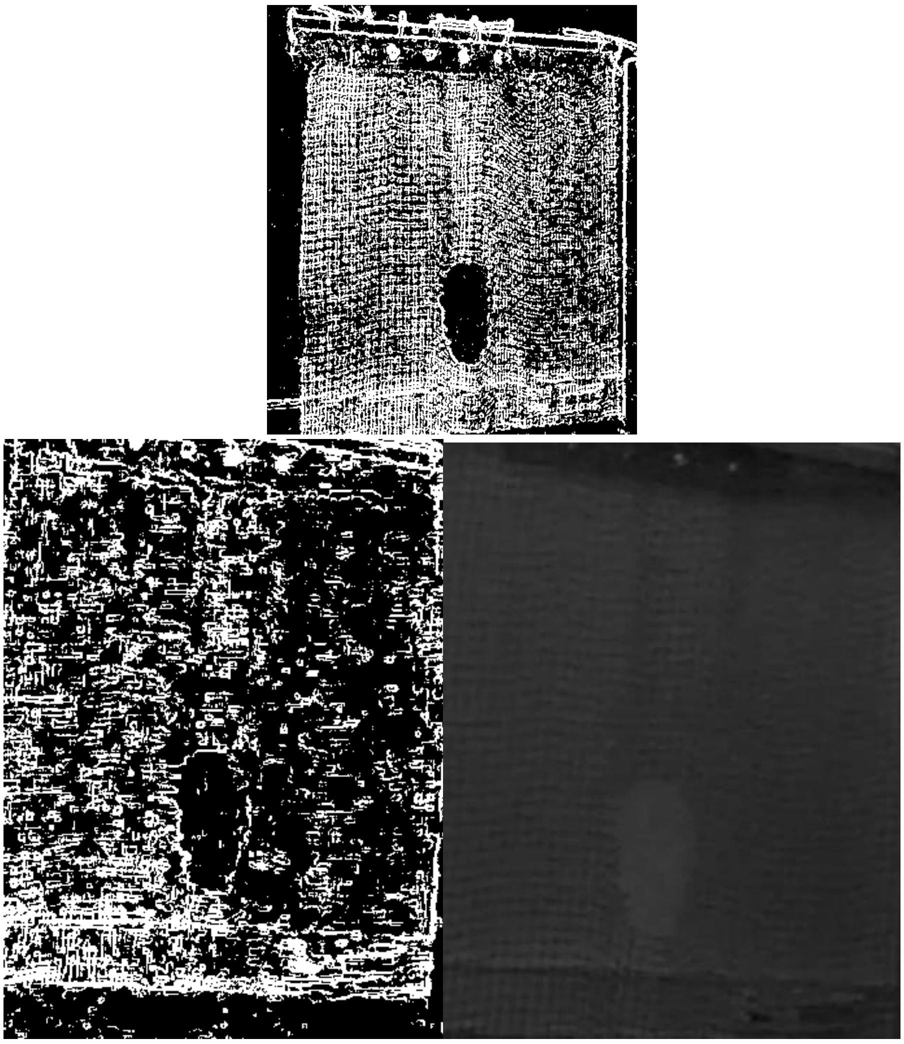


Figure 69: Net filmed with shutter speed 1/6000s: 3m distance no motion(top), 3m distance vertical motion(bottom)

In Figure 70 the camera has been set to a shutter speed of 1/10000s. As can be seen the footage is virtually useless. For this super-fast shutter speed to work very strong illumination is required.

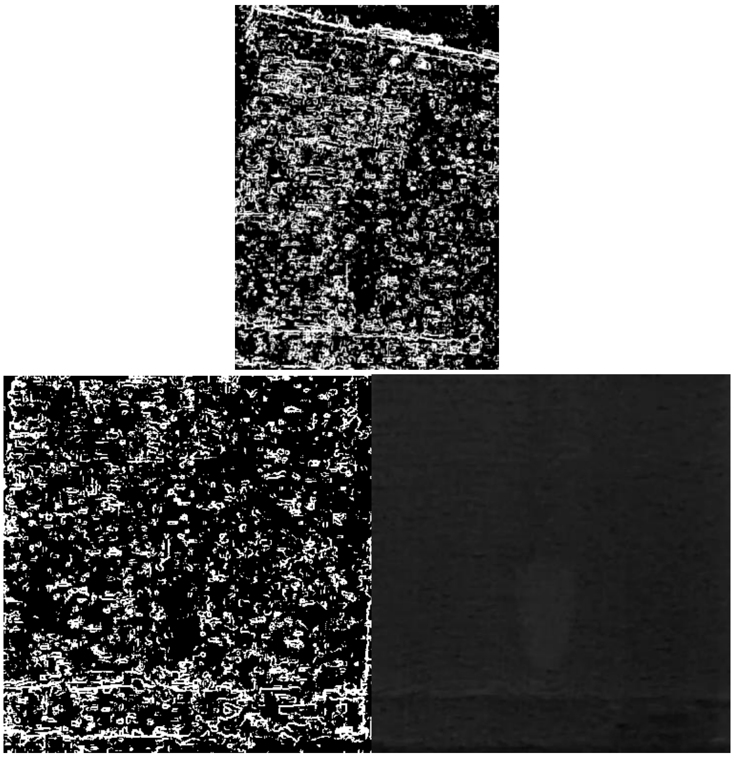


Figure 70: Net filmed with shutter speed 1/10000s: 3m distance no motion(top), 3m distance vertical motion(bottom)

5. Testing captured video

After obtaining the videos from the field test they were used to test and improve the software being developed. The results from each iteration of testing gave results that were used for making a new version of the software and gave a good basis to discuss and conclude on whether the algorithms developed during this thesis could be used for automatic surveillance of sea cages.

5.1. Testing videos on System 1

Using the video material obtained during the field test the various functions developed in System 1 were tested to check their viability and usefulness. The functions are described in chapter 3.2.

To test the various functions a reference video were used. In this video there is a flawless net that stands relatively still. An image from this video using the L color space is shown in Figure 71

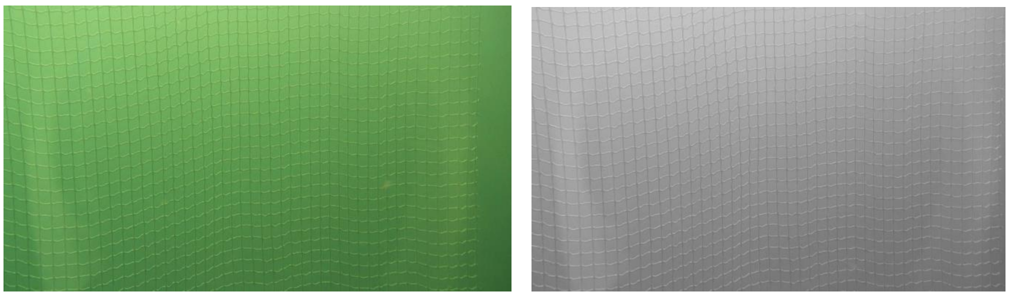


Figure 71: Reference video. Video frame (left)and L color space (right)

The halo of light that is visible on the images in Figure 71 makes the segmentation process harder; the big problem however is in the bottom right of the image. Here the sun isn’t as strongly present as in the rest of the image, which combined with the halo makes it hard to distinguish the vertical mask apart from the surrounding sea. This is shown in Figure 72. During situations like this it is extremely difficult to determine if the seemingly missing parts of the net is due to damage or just camera issues.

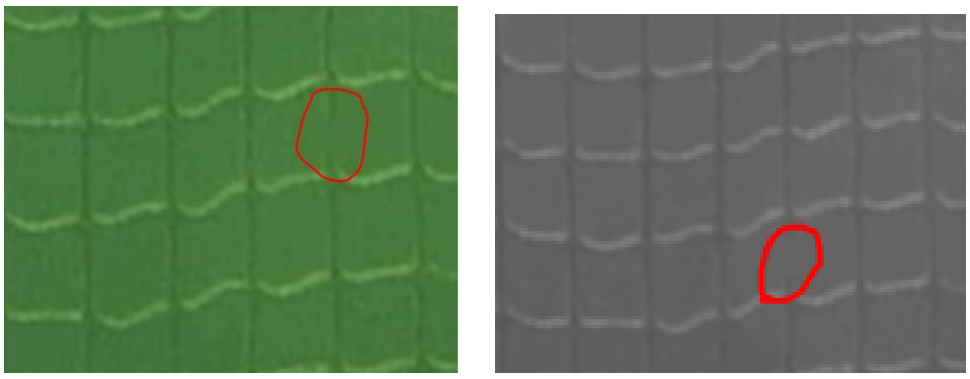


Figure 72: Bottom right part of the net.

During the testing it became evident that the light from the sun was not as strongly present in the bottom right side of the image. This was due to a large boat that was docked in the marina next to the test site. The problems occurring with sub-standard lighting conditions affects the system behavior. For our purposes this is actually a wanted behavior because it makes the segmentation process harder and it illustrates the problems that arise with bad and uneven illumination.

5.1.1. Color spaces

From Figure 73 it can be seen that the two xyY color spaces behaved more or less alike. All the illumination color spaces gave a clear view of the net, the same did the RGB color spaces but since they are considered less robust than the illumination color spaces they will not be used further for testing. Of the color spaces tested the illumination color spaces should work with the edge detection methods while the chromatic y color space could work together with Otsu’s method.

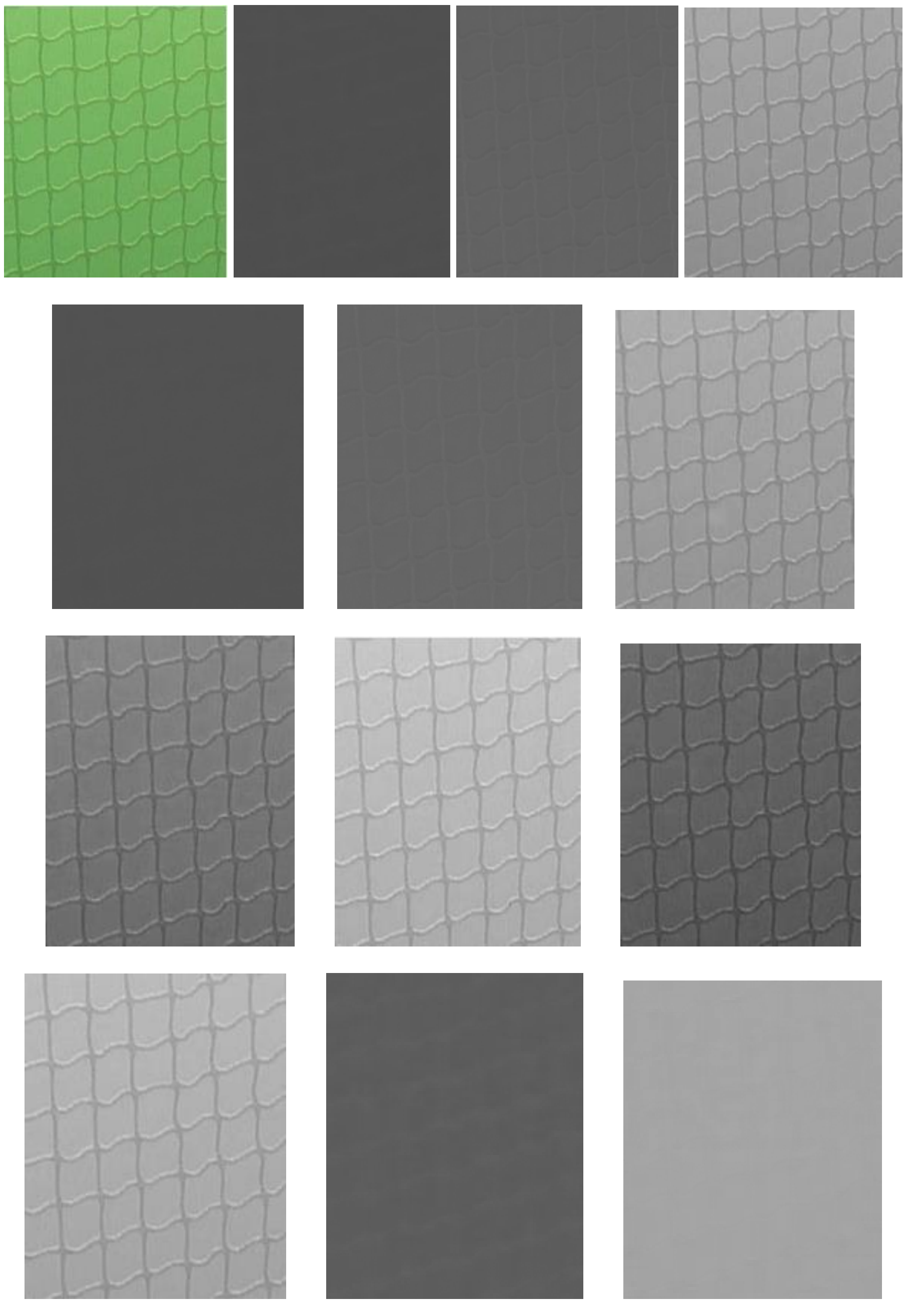


Figure 73: Images showing the different channels of the different color spaces. Top row is original image and the xyY color space from OpenCV's RGB2XYZ function. Second row is the xyY color space made using Equation 1. Third row is the RGB color space and the last row is the L*a*b* color space

5.1.2. Segmentation methods

During the system testing it became evident that edge detection methods performed best with the luminance color spaces. Otsu's method worked best with the y color space from xyY.

5.1.2.1. Segmentation using edge detection, Sobel

Using the Sobel segmentation method it became evident that the OpenCV Sobel function didn't perform as well as was expected. When using the Sobel algorithm on the frame from Figure 71 it becomes evident that the strong sun reflection on the horizontal part of the sea cage masks makes the OpenCV Sobel function disregard the vertical direction of the masks. When trying to lower the histogram percentile to include more edges the image just gets fuzzy without adding sufficient enough points where the vertical lines are. This effect is shown in Figure 74.

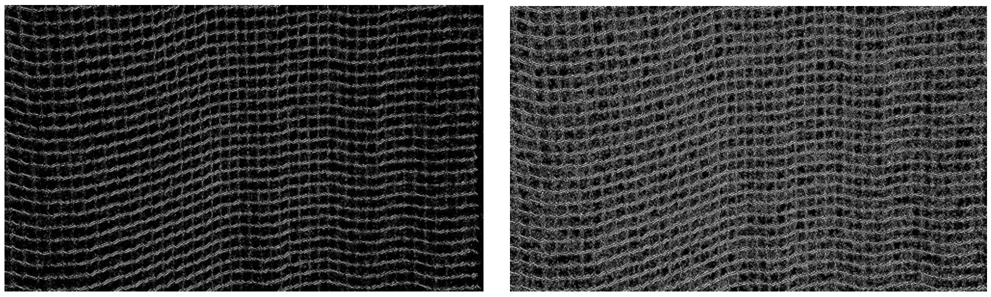


Figure 74: Using Sobel for segmentation. Histogram percentile = 90 (left) and 70 (right)

5.1.2.2. Segmentation jusing edge detection, Scharr

Using the special Scharr mask in the OpenCV Sobel function led to better results than the Sobel segmentation, Figure 75. When using the Scharr mask the OpenCV Sobel function must first be run in the x-direction then the y-direction the two results was then merged using the OpenCV function Addweighted(). With this function the two results can be weighted when merged together, tuning this variable for varied conditions can increase performance.

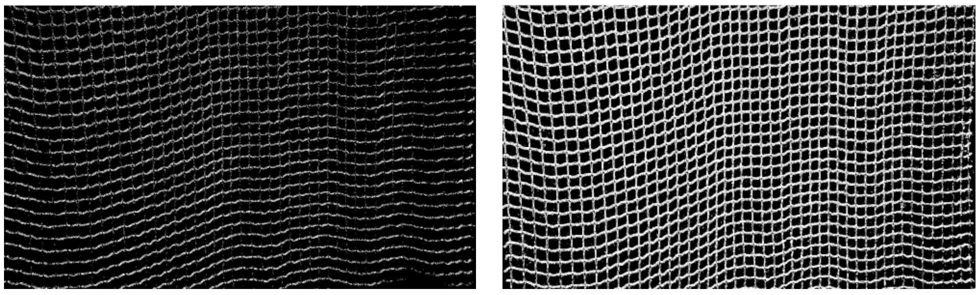


Figure 75: Using Scharr for segmentation. Histogram percentile = 90 (left) and 70(right)

5.1.2.3. Segmentation using Otsu's method

Using Otsu's method did not work especially well with the luminance color spaces. It is very sensitive to the light gradient from the sun and the suns effect on the horizontal lines of the masks. Using the local approach gave a better result. For this method to be viable it must be remade so that when segmenting a sub-image the result must be inverted if the sub-image has defined foreground and background differently than its adjacent sub-images.

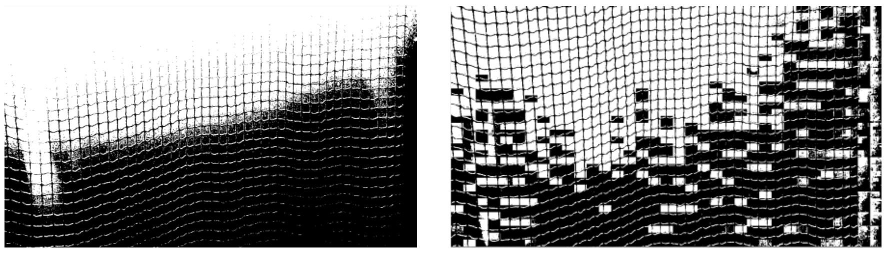


Figure 76: Using Otsu's method for segmentation, L color space. Global approach (left) and Local approach with 1225 sub-images (right)

When using the chromatic y color space the results seemed more promising. In Figure 77 the local Otsu approach with 1600 sub-images is shown. To see the details only a part of the video frame net is shown. Using the y color space and enhancing the method as explained above may make Local Otsu a viable choice. It uses slightly less computational time compared to global edge detection methods so it would be very interesting to get it to work properly.

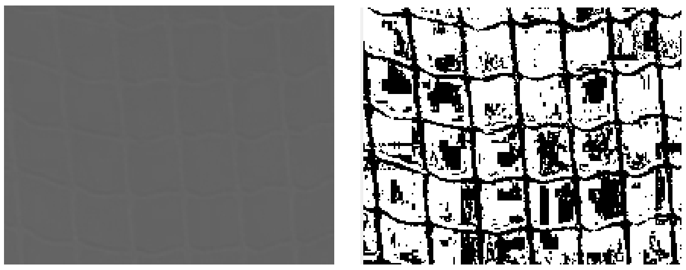


Figure 77: Using Otsu's method for segmentation, y color space and 1600 sub-images

Using edge detection to enhance Otsu's method was also tried. This however did not work at all as good as hoped and is therefore, for the time being, considered as unfit for this purpose.

5.1.2.4. Using naïve thresholding for segmentation

Using the naïve approach works up to a point in video frames converted to the y color space. The strong sun still makes too much problems, however at lower depths where sun is not an issue this very simple approach could be used in combination with a master system that adjusts the threshold to get maximal "sea cage structure" in every frame or by manually adjusting it.

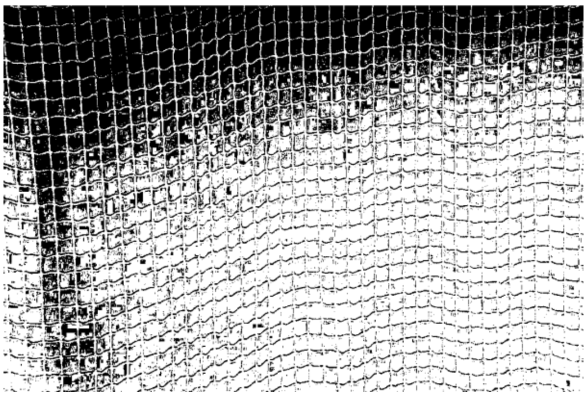


Figure 78: Using naive thresholding for segmentation, y color space

5.1.2.5. *Adaptive thresholding*

From 2.2.2 it is stated that the constant C value is deducted from the mean or Gaussian threshold value to optimize the threshold value. Using the reference video with the chromatic y color space it was only possible to use a C value of 1 and 0. Using the L color space instead led to a constant value C of 6 to perform the best and a wider range of C values gave adequate results. In Figure 79 the result from using the y color space is shown. With this approach the segmentation method finds the edges of the net but since it segments both the sea and the net as foreground. This is not desirable since a round of dilation is used when pixels are missing along the borders to close the gaps.

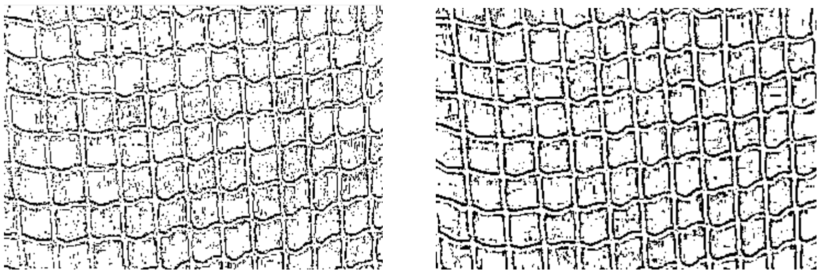


Figure 79: Adaptive thresholding with Kernel size of 7 and C set to 1, y color space. Gaussian (left) and Mean (right)

When using the L color space it was possible to segment out the net as solid lines of foreground pixels. In Figure 80 the performance of the region growing algorithm used on the reference video with the L color space is shown. It detects almost all masks expect for some masks in the bottom right

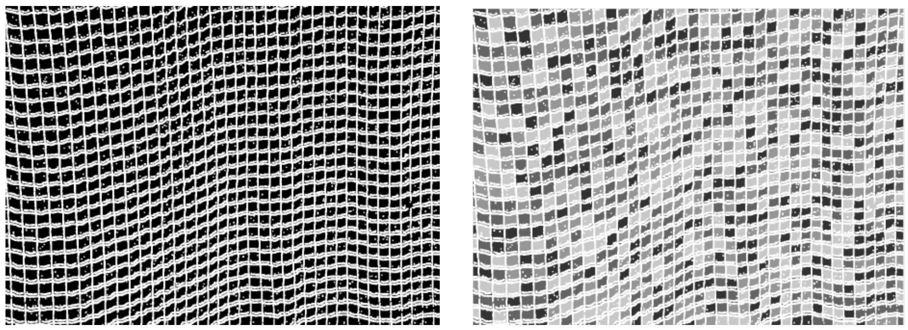


Figure 80: Mean adaptive thresholding with Kernel size 7 and C set to 3, L color space

The mean method seems to give stronger segmentation of the net which makes it easier to get closed contours for each mask. The difference is however not large and both methods could be used.

5.1.3. Improvements to be made on functions from System 1

- Remove Local and Global option for segmentation methods, global can be chosen as the segmentation method with 1 sub-image.
- Improve Sobel segmentation to segment dx and dy separately and then merging. The OpenCV function that performs both at the same time is too sensitive when the light is so strong.
- Improve Scharr and Sobel so that the weights in AddWeighted() can be tuned by the user.
- Improve Local Otsu so that it segments the whole image correctly
- Improve Local Otsu so that if a sub-image appear opposite of its neighbors it inverts the color(i.e. black becomes white and vice versa)
- Remove lightest area color the grown region gets colored in for the region growing algorithm. The lightest color was too close to white so hard to visually see.
- Remove enhanced Otsu using edge detection

5.2.Testing of captured video on System 2

After the field test System 1 was improved in various ways. System 2 is still very much a test system where the different functions can be tested and adjusted, the functionality however is much improved.

5.2.1. Image and Segmentation enhancing functions

In addition to dilation and Gaussian smoothing a few new enhancement methods were added. Two different approaches to equalize light were made along with a sharpening function.

5.2.1.1. Light gradient equalizer using sub-images

This approach is explained in chapter 3.3.2.1. When trying this algorithm on the reference video the light got evened out fairly well. Otsu performed better with this turned on, however there was seemingly little or no improvement to the Sobel and Scharr segmentation methods.

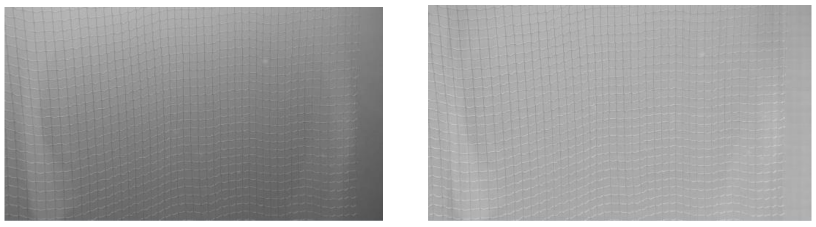


Figure 81: Using light gradient equalizer using sub-images, input frame(left) and output frame(right)

5.2.1.2. Light gradient equalizer using top hat transformation

Using the top hat transformation to reduce the suns effect on the sea cage worked quite nicely when used on the reference video as can be seen in Figure 82. Otsu performed better with this turned on, however there was seemingly little or no improvement to the Sobel and Scharr segmentation methods.

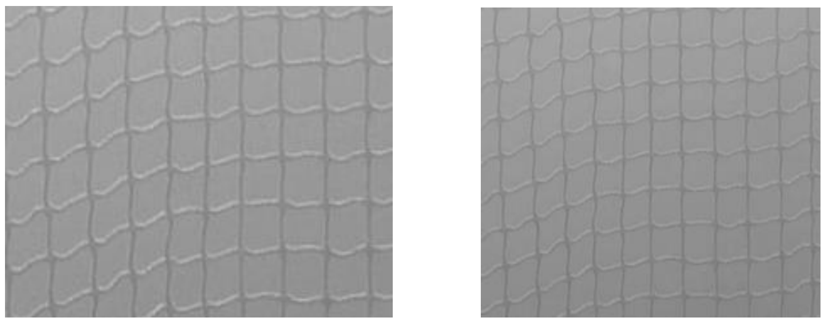


Figure 82: Top-hat transformation to remove suns effect on sea-cage panel. Before operation(left) and after operation(right)

5.2.1.3. Contrast enhancement using morphological operations

The result of applying this method to the image can be seen in Figure 83. Since the function enhances contrasts it naturally enhances noise as well. As of now it doesn’t seem to help the segmentation process.

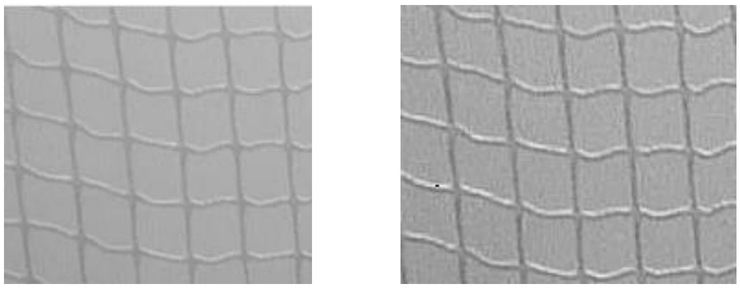


Figure 83: Contrast enhancement using top hat and bottom hat

5.2.2. Segmentation methods

With the enhancement to the Sobel algorithm it now seems like the best alternative. In System 2 Laplace edge detection was also tried to use for segmentation. The Laplace operator works fairly good combined with Gaussian smoothing but a well-known issue with the Laplace operator is that it can produce double edges and with the sun shining on the net a double edge gets segmented. A lot of work has been done into making the Otsu algorithm work better. It still doesn't not look like a viable choice although the performance of the algorithm have been greatly improved.

5.2.2.1. Segmentation using edge detection, Sobel

After enhancing the Sobel algorithm it worked a lot better. In Figure 84 the difference between System 1 and System 2 can be seen. System 2 is infinitely better, however the bottom right part of the sea cage panel is not segmented correctly due to the image quality not being good enough as shown in Figure 72 on page 59. Using the settings in Table 1 seems to give the best performance of the Sobel algorithm

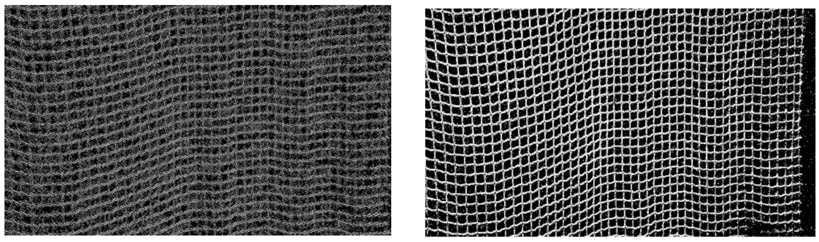


Figure 84: Sobel algorithm, System 1(left) and System 2(right)

dx	dy	Histogram percentile	Kernel size	dx Weight	dy weight
1	1	70	3	90	50
Color space	Dilation	Gaussian smoothing	Light equalization	Sharpening	
Luminance	1 round	off	off	off	

Table 1: Sobel parameters for best performance, System 2

The remainder of the noise in the right image in Figure 84 can be removed using the Gaussian smoothing function with a kernel size of 5. This however worsens the performance of the bottom right part of the image.

5.2.2.2. Segmentation jusing edge detection, Scharr

For the Scharr algorithm the option of choosing the weights of the dx and dy Scharr results when merging them were added. This does not seem to give a huge increase in performance when assessing the reference video.

Another thing that was added to the Scharr algorithm was the option of setting a threshold value where edge pixels that are higher than the threshold are considered to belong to the net and are kept as foreground, while all other pixels are background. A threshold value of about 75 is found to give the same results as setting the histogram percentile to 70. When segmenting the net to use for damage detection the histogram percentile is still considered the best option since thresholding the pixel value of the edges is a less robust way to do the segmentation. Pixel values might change with varying environmental conditions whereas the 70th percentile will always be the 70th percentile. If something with stronger edges comes in front of the camera then of course that will take some of the edge pixels away from the percentile method this is however believed to be a smaller problem than using a set threshold.

Comment [WU1]: flytt til no diskusjonsgreir?

dx	dy	Histogram percentile	dx weight	dy weight
checked	checked	70	90	50
Color space	**Dilation**	**Gaussian smoothing**	**Light equalization**	**Sharpening**
Luminance	1 round	off	off	off

Table 2: Scharr parameters for best performance, System 2

5.2.2.3. Segmentation jusing edge detection, Laplace

The Laplacian algorithm produces more noise than the other edge detection methods as expected. Also because the sun reflects off the horizontal lines the lines is segmented as two lines. This somewhat ruins the current way of detecting damages to the net so for the Laplacian to be of any use either the double line must be removed or the damage detection algorithm must be tweaked. In Figure 85 the best image obtained from the reference video is shown. In Table 3 the settings used to obtain the best result is listed.

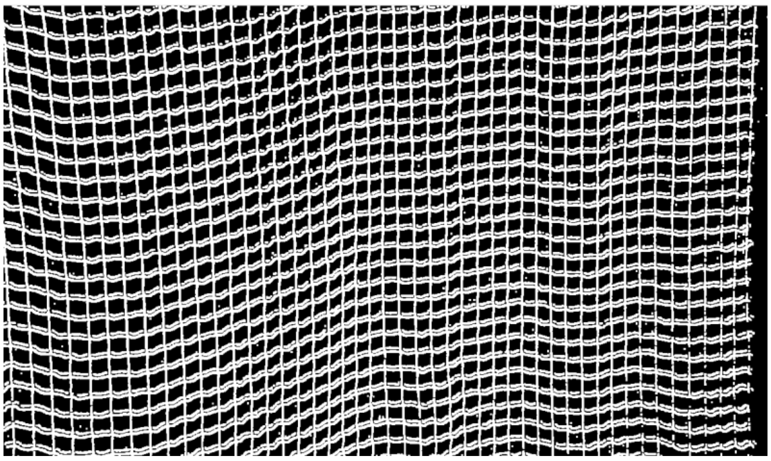


Figure 85: Laplace edge detection

Histogram percentile	Kernel size	Color space	Dilation
82	5	Luminance	1 round
Gaussian smoothing	**Light equalization**	**Sharpening**	
On with kernel size 5	off	off	

Table 3: Laplace parameters for best performance, System 2

5.2.2.4. *Segmentation using Otsu's method*

A lot of things were tried to enhance Otsu's method. Both trying to make the segmentation of foreground and background smarter and trying to pre-process the image before segmenting. In Figure 86 the best results using the reference video on System 2 is shown. The settings are explained in Table 4.

One thing that makes it very hard to segment the net from the sea correctly is the halo of light shown in Figure 71. In the left image in Figure 86 one can see one sub-image to the left where the net still is segmenetd as background because of the halo of light.

Comment [WU2]: Mer? Bedre? Olsen sier JA

Using the chromatic y color space the negative effect of the halo is decreased there is however still problems with large white patches turning up and masking the sea cage.

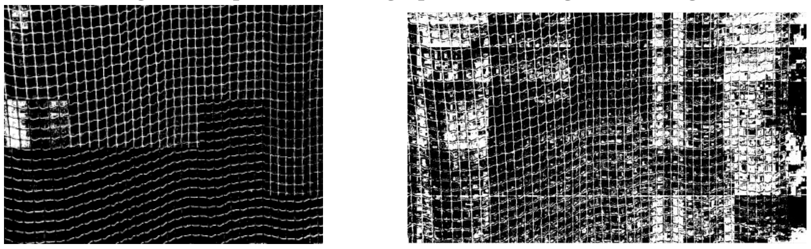


Figure 86: Segmentation using Otsu's method, using Y color space(left) and y color space (right)System 2

Sub-images		Color space	Dilation
25		Luminance	none
Gaussian smoothing	Light equalization	Sharpening	
On with kernel size 5	off	off	

Table 4: Otsu parameters for best performance, System 2

Program modes

5.2.2.5. Net damage detection

Using segmentation using the Sobel algorithm a fairly good segmentation is performed. As can be seen in Figure 87 some suspect regions have been found in the right side of the net and marked with a black circle. Most of them are due to the bad image quality in that part of the image but not all. It still happens that a few net masks are not segmented whole and therefore the region growth algorithm detects them as double or triple mask areas, i.e. suspect area sizes.

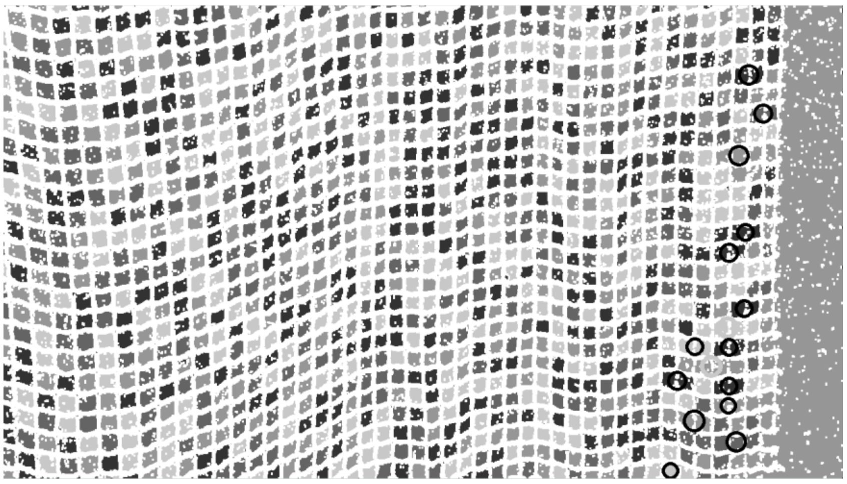


Figure 87: Net damage detection on reference video using Sobel

To test the damage detection algorithm under hard conditions the hole_2 video were used. In Figure 88 the performance of the damage detection algorithm can be seen. Since there is a growth from an anchor chain lying under the camera coming up into view the autofocus on the camera starts to focus on that instead of the net. Autofocus should really not be used when running the algorithms as things like this may happen. In the middle image in Figure 88 the algorithm can be seen to work fairly well. Many suspect areas are found in the right side where the image quality is poor and the hole in the net is found to be a hole. The growth that sticks up is also found to be a hole because it has no edges.

In the bottom mage in Figure 88 a case where the algorithm has broken down can be seen. Here the algorithm went from taking 280 pixels to be the correct mask size to take that 30 pixels is the correct mask size. Some memory should be added to the function so that it can’t switch like this.

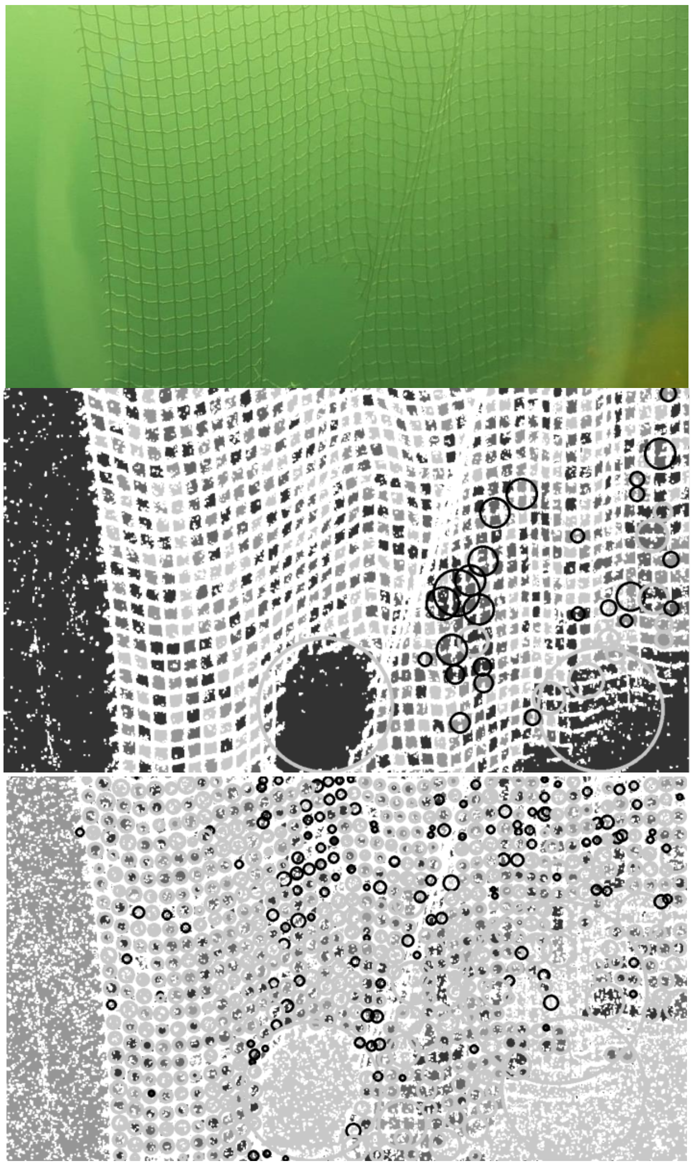


Figure 88: Damage detection under hard circumstances, camera video frame(top), resulting damages found(middle) and broken down algorithm(bottom)

Using the damage detection algorithm the system uses approximately 1.2 seconds per frame on the computer in the office.

5.2.2.6. Growth detection

For System 2 another option were made available to the Scharr function. Instead of using histogram percentile to determine which pixels is foreground and which pixels is background pixels one can now use an edge value threshold. Using a frame from the reference video as a reference to a clean net and the growth video it was found that using an edge pixel threshold value of 90 the system found a growth of 25% on the growth video. Using the histogram percentile option the growth found was only 10%. Images from the reference video and growth video can be found in Figure 89. Although this seems promising most of the increase in foreground pixels may only be due to the double layer of sea cage that is present in the bottom part of the growth patch. It does not seem that the edge detection segmentation methods perform very well when growths cover a too large space and are too uniform in color and brightness.

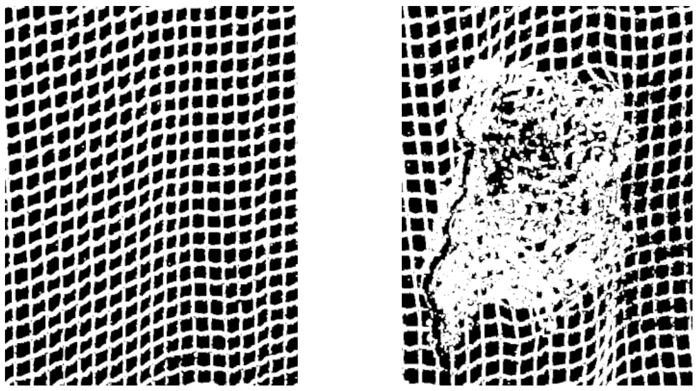


Figure 89: Segmentation result comparison of reference video(left) and growth video(right)

Using Otsu’s method together with the y color space a better result was seemingly obtained. This way the patch with growths on it is seen more clearly, however since there are many white patches all around the image this method cannot be used in the developed growth detection algorithm.

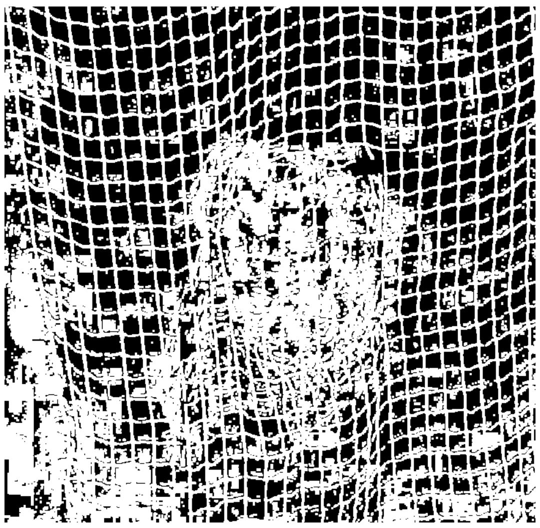


Figure 90: Otsu's method used to find growths

Using the growth detection algorithm uses approximately 0.2 seconds per video frame. This is very efficient and basically the same run time as running on the “No mode” setting.

5.3.Testing of captured video on the final system

After testing System 2 a final system were made. Where System 2 can be used for experimentation the final system is more of an end product, this leads to fewer choices for the user, only using the best performing algorithms with their best settings.

5.3.1. Using morphological opening to remove salt and pepper noise

In Figure 91 the performance of the salt and pepper remover is shown. The top row is taken from the top left of the image where the image quality is the best. Here the salt and pepper remover works perfectly. The bottom row shows the bottom right part of the image, here the salt and pepper remover does remove some parts of the sea cage panel, however those parts belong to broken masks and therefore it is not a big problem. It should be noted that all the images are taken from the program running the reference video and as such it is not exactly the same video frame being processed in the four images.

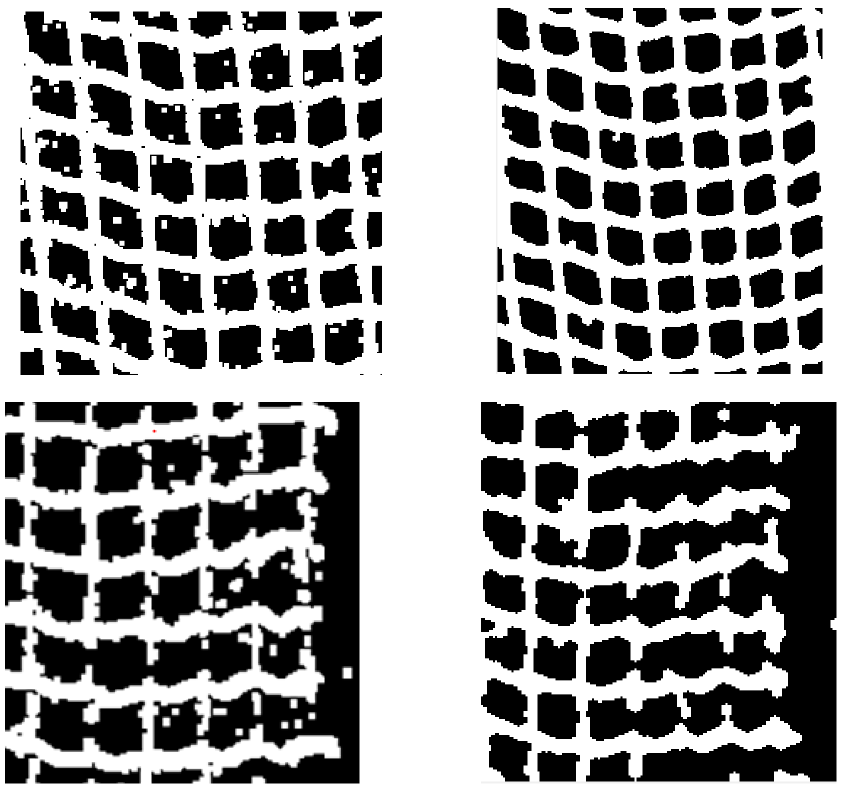


Figure 91: Using opening to remove salt and pepper noise. Before and after on a part with good segmentation result(top row) and a bad area(bottom row)

Using this method to remove salt and pepper noise seems to work fine, however caution must be used since the kernel size used to remove the salt and pepper noise can potentially close up the net if the masks are small enough. This can happen when the distance from the net is larger than during this test. If this morphological approach is used either information about the distance to the net is needed to change the kernel accordingly or the camera must only film a given size of the net in each frame, i.e. zoom in if the ROV is further from the net.

5.3.2. Improved damage detection algorithm

With the new improvements to the damage detection algorithm it behaved better. To test out the damage detection algorithm and the system as a whole the different videos from the field test were run on the final system.

5.3.2.1. Reference video

When testing the final system on the reference video one can see in Figure 92 that the system finds double masks that it marks with a black ring, the system also finds an region where four masks is taken as one region, and correctly marking that with a light grey ring.

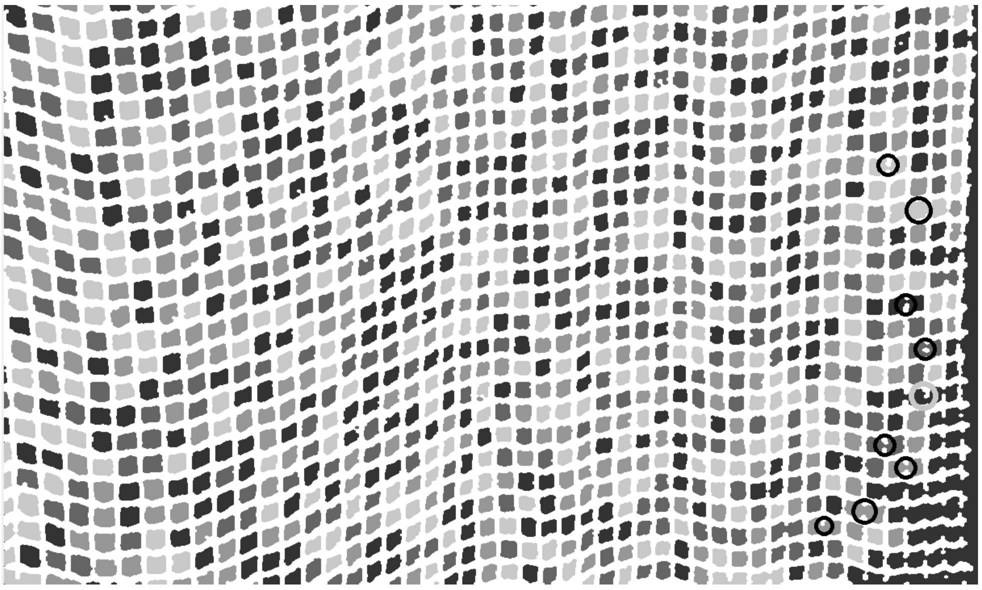


Figure 92: Damage detection of the reference video, final system

5.3.2.2. Sea-cage panel with hole

In 5.2.2.5 the video named hole_1 was used, now the video named hole_1 was used to test both videos. In the hole_1 video there was as was the case with the other video growths from an anchor chain coming up into the frame and after segmentation it looks like a big tear.

In the bottom left image in Figure 93 the area in the bottom is detected as a damaged area since it is inside the sea cage panel and it is a relatively big area. As such it is considered a valid area and marked with light grey as a damaged area. The small part of the hole in the net that is visible in the top part of the image is correctly detected and marked.

In the bottom right image in Figure 93 the seemingly teared area is part of the outside of the sea cage panel and therefore is ignored. The hole in the top of that image is detected as a damaged area and marked with a light grey ring as it should.

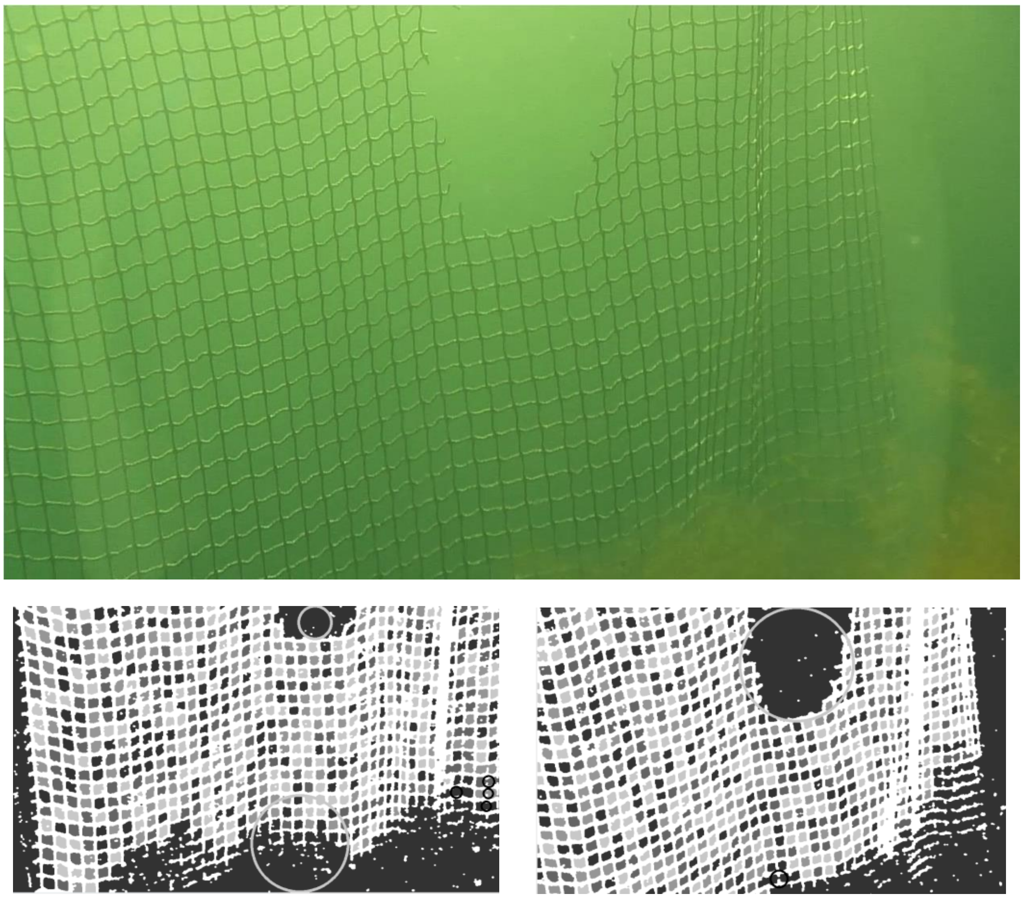


Figure 93: Damage detection on hole_1 video, final system

5.3.2.3. *Sea-cage panel with growth*

As before the algorithm is far from perfect, however using Scharr for segmentation using an edge strength threshold value of 70 the area where there is single net and growth marked with a red ring is much more pronounced than before. This can be seen in Figure 94.The part with double layers of net and growths is an almost solid mass of foreground pixels which coincides well with the reality seen in the left image.

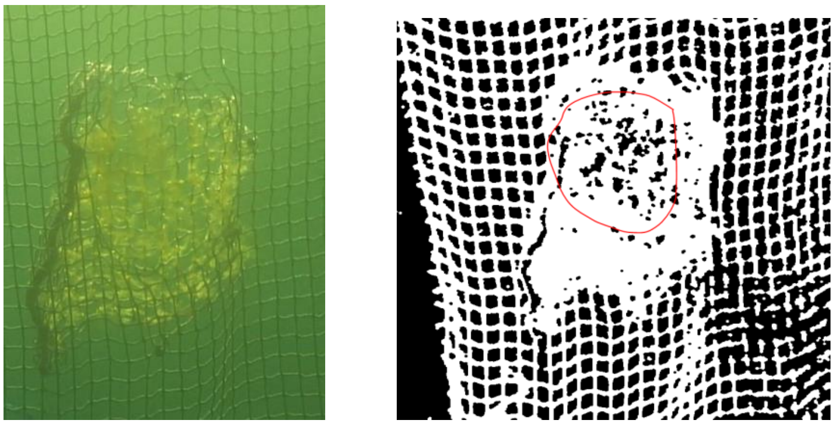


Figure 94: Growth detection, final system

Sea-cage panel with vertical tear

The result of running damage detection on the video named vertical_tear_1 is shown in Figure 95. The system finds the tear but in the video the movement of the net and less than ideal light conditions makes the system not segmenting all the masks correctly.

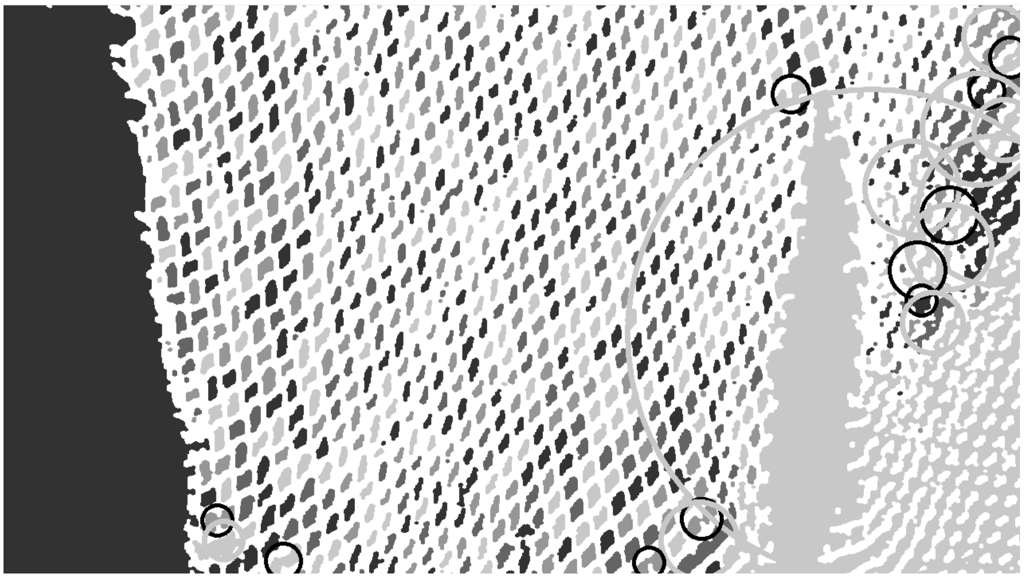


Figure 95: Damage detection on vertical_tear_1 video, final system

5.3.2.4. Sea-cage panel with horizontal tear

The video named horizontal_tear_2 were used for this test. This video is not very good since there is something to the left in the image that steals the focusing of the camera. As can be seen in Figure 96 the top image in the image quality is really bad and as such it is not a surprise that the damage detection in addition to find the damage as it should finds a lot more damaged areas due to the growth and poor segmentation. The huge light grey ring belongs to the growth to the left.

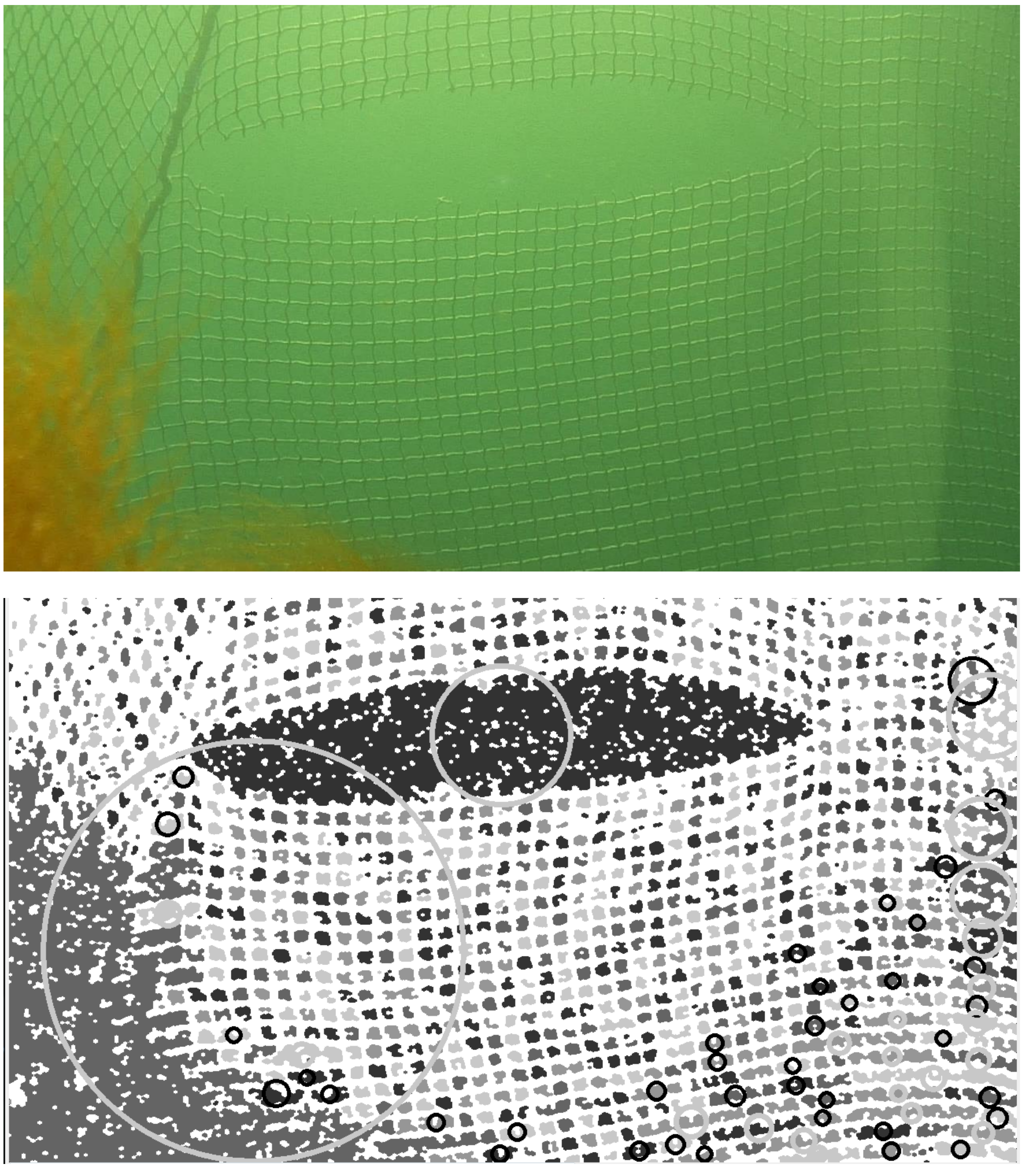


Figure 96: damage detection on horizontal_tear_2 video, final system

5.3.2.5. Sea-cage panel with L shaped tear

For this test the videos named L_tear_1, L_tear_2 and L_tear_3 were used. None of the videos had a good enough quality for the system to be able to work properly. In Figure 97 the reason for the impossible working condition for the system is shown. The growth occupies almost the whole image and the image quality in the rest of the image is also lacking.

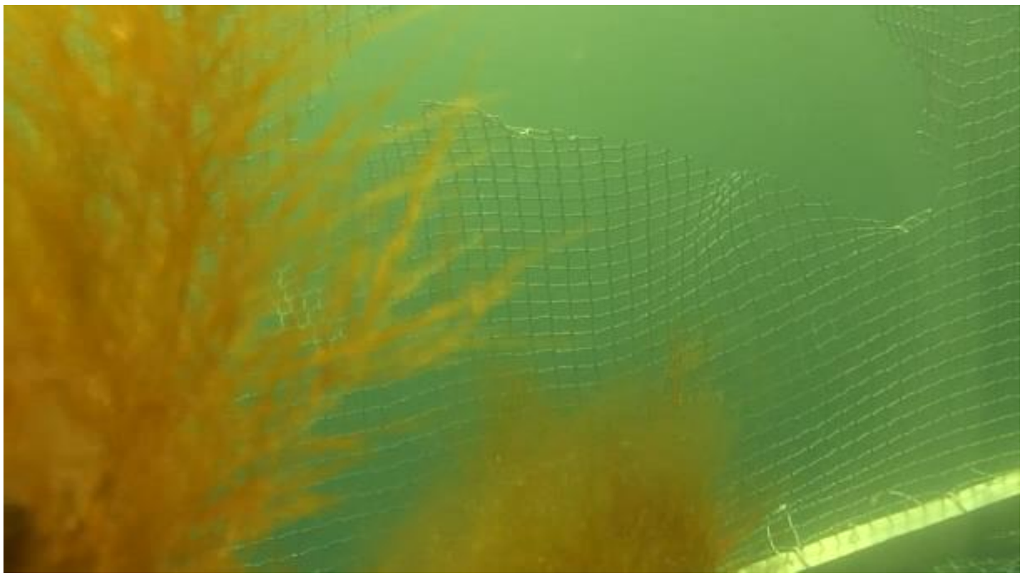


Figure 97: Typical frame from L_tear videos

6. Discussion

During this thesis edge detection was found to be the most robust method for segmentation. Under good conditions this approach seems to work fairly well. But as with other segmentation methods the results is highly dependent of the quality of the video frames obtained. In the figures in chapter 5.3.2 one can see how different the results from segmentation and damage detection was on the different sea cage panels. This difference in how successful the system were to properly segment the sea cage panel was due to different focus and light conditions affecting the quality of the video frames. Bad conditions that affect the image quality are a big problem when using a video stream from the camera to automatically detect damages. Great care must be taken in order to ensure that video frames the damage detection system gets is as good as possible.

Hardware

During this thesis a large part of the time used was spent on finding, requesting, ordering and waiting for the hardware components. The camera equipment obtained during this thesis is far better than the equipment the department had from before. In the end I was very happy with the performance of the camera equipment.

The camera seems to be an extremely useful piece of equipment. It is very versatile and is well suited to tackle the bad conditions experienced under water. However auto-focus should never be used while running the algorithms! After the proper zooming and focusing is obtained the camera should be locked to those settings, preferably camera control should be integrated into the developed software so the system automatically can adjust the camera settings and so that the user can change the camera settings from the damage detection system.

Using HD-SDI for transmitting the video stream was really good after it finally started to work properly. Getting it to work however was non-trivial, seeing as neither the camera, the HD-SDI interface module nor the frame grabber could give any indication as to what was wrong. The video stream had sharp and detailed video frames where there was enough light. In the parts of the video frames where there was bad lighting naturally issues arose.

The lights acquired were not strong enough to compete with the strong sun during the field test. Since no testing was performed where the camera was submerged further down it is uncertain how well the lights would have performed acting as the sole source of illumination. After testing the videos obtained during the field test it was clear that lights are extremely important for good imaging. Especially considering the fact that the shutter speed must be set to around 1/1000s to be able to capture motion without motion blurring. This demands more light compared to running on the 1/50s shutter speed the camera uses when in auto-focus mode. If the lights that are present on the Argus ROVs are good enough for an automated damage detection system is hard to say since I did not get to test with them and had to find some other lights for the test. The ROV lights should however be stronger than the lights acquired during this thesis so if at a later stage the lights acquired are found to be good enough then the lights on the ROVS should also be good enough. An even background illumination is important to get a best possible segmentation result.

Using edge detection for segmentation

Using edge detection like it has been used in this thesis is based upon the principle that the net is seen in the image as edges. Of the different methods tried it seemed the most robust but as with other segmentation methods it has its weaknesses.

One weakness is that since the structure to be segmented, in this case the sea-cage is only segmented based on its edges the structure will not be correctly segmented if it is too wide. This effect is shown in Figure 98 where in the top row the segmentation works as intended. The line get seen as two edges, one from crossing from blue to green and one from green to blue. Morphological dilation is added to fuse the edges found together. In the bottom row however the line is too wide. This results to the edges found not fusing together even after dilation.

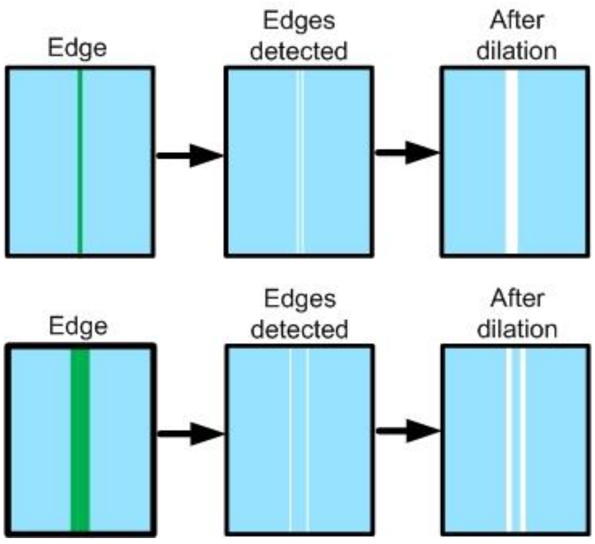


Figure 98: Edge detection used for segmentation, successful attempt(top row) and failed attempt(bottom row)

This weakness became apparent when trying to use edge detection for growth detection. Both the real growth that appeared in some of the testing videos and the simulated growth used in the growth video did not get segmented properly. Since the growths were very uniform in color and texture not enough edge pixels were found in the region where the growths were.

Another potential weakness is that other objects in the video frame like fishes or debris would look like holes or deformed areas after segmenting the sea cage. Using the proposed method from Figure 100 could help on this potential problem since the algorithm for finding damaged areas would be more robust. Since the method doesn’t try to segment the image into a certain number of parts, like foreground 1, foreground 2 and background; but instead perceives every object as edges the system should not be too confused by other objects in the environment. Of course situation can happen where debris or fish block the view of a damaged area to the net which would be bad since the method needs a clear view of the damaged area.

With very varying light conditions the weights of the dx and dy edge detected images would have to be adjusted to account for that. Although using a 50/50 weighting should work well when the sun is not messing up the segmentation.

How well the Sobel edge detection algorithm would work when the sea cage is oriented like a diamond and not flag-oriented should be studied closer. There is possible to use kernels for the Sobel operation that is not oriented in the x or y direction, they are instead oriented on the diagonal. It would be possible to perform segmentation in the x-, y-, and diagonal directions and then merge the four different results together.

At this stage the edge detection approach to segment out the sea cage structure seems to be an option with a lot of promise.

Using edge detection and region growing for damage detection

When using only region growing to detect damages it can happen that a damaged area is not seen since it is covered by another structure. For the developed method to work the damaged area must be captured dead on. One way that a damaged area can be hidden is a situation like shown in Figure 99. Here there is a double sea cage structure with a hole in one of the nets.

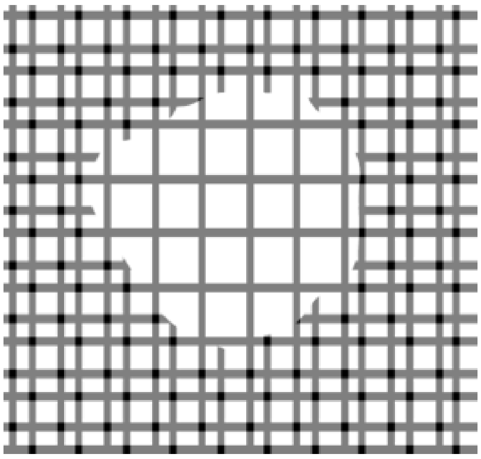


Figure 99: Undetectable hole in double sea cage structure

It could be possible to check if the vertical and horizontal lines are approximate straight along the video frame. If there is a hole or tear then all the lines will not be vertical and horizontal but the ones that goes through the hole is broken and the lines surrounding the hole are more curved and deformed than normally. When damages like the one in Figure 99 are present one could also detect the damage by detection that a double sea cage structure is present and if there in the middle of the double sea cage structure is an area with only a single sea cage structure then it is considered a damaged or suspect area.

Using edge detection and pixel-counting for growth detection

To find out how much of the mask area in the sea cage has diminished because of growths and things in the net a simple approach were used. First a reference video frame with no growth was given to the system. The system counted foreground pixels after segmentation was performed and that amount of foreground pixels were set as 0% growth. Then when the system is run in growth detection mode the system counts the foreground pixels in the video frames and compares them to the reference frame. If the net together with all the things that diminish the mask area gets segmented out as foreground this approach works like a charm. However since at this time the segmentation method did not segment the growths as needed for this approach the growth detection algorithm is unreliable.

Using the final system it was found that the improved Otsu's method performed better than the Scharr method in correctly segmenting out the growths. Although the Scharr function found that the area had growths it did not segment the growths out in a natural way, meaning that the end result did not look like the growths. If the improved Otsu's method could be made to work more robustly and not make patches of foreground pixels all around in the processed frame it would be the best option. If the sea cage and the growths gets corrected segmented it is trivial to detect the level of growth present on the sea cage.

7. Conclusion

Although in (Taylor & Kelly, 2010) it was stated that cameras have been found difficult to use for damage detection on sea cages I find that it can most certainly be done. It all comes down to segmenting the sea cage properly from the surrounding sea.

Using the proposed edge detection segmentation scheme the system would be very robust since as long as the sea cage doesn't blend completely into the sea edges will be present and can be detected. This method will find the sea cage no matter what the color of the sea cage and the sea is and will not behave badly if nets and ropes have different colors which can be the case with other segmentation approaches.

At the present time the performance of the system is in question when subject to big amounts of growths and different distances, other than that the system performs as desired given that the camera has good enough working conditions to take clear frames.

Concerning the computation time per frame it is for the final system 1.2 seconds per frame on my computer at school. In the videos obtained from the test the camera covered around $1.5m^2$if this system was used to detect damages on a circular sea cage structure which has a circumference of 120m and a depth of 30m it would take the system optimally 48 minutes. If the system is run with a 50% overlap of the frames to improve the robustness of the damage detection algorithm the system would on an near optimal run search through the sea cage in one and a half hour. With better performance of the algorithms and a more optimized computer it can use even less time per frame and thus search through the structure faster. It could also be possible to film the sea cage from a greater distance as long as the segmentation algorithms still works as they should.

8. Future work

There are many things that can be done to further improve the developed system and thereby increasing its usefulness.

Hardware

Since sufficient illumination is imperative for the system to obtain good results, more effort should be made into finding a better light system. Obtaining similar lights to the ones used on the ROVs from Argus would be an interesting option however other light schemes could be looked at instead.

The camera works very well, but using one software program to control the camera and another to control the damage detection system is unnecessary. As stated in 0 the camera control can be integrated into the program, for example by using the open source C library available at http://sourceforge.net/projects/libvisca

Robustness of the damage detection algorithm

To make the system more robust it is thought that in the future a suspect region will not be considered a damage area if that part of the net is whole on another image. As shown in Figure 100 the first time a small damage is detected it should get marked, in this case shown as a red area. With future location tracking in place it should not be a large problem to track the damaged area and check it again when the next frame is processed. In the bottom image in Figure 100 the area have been checked again in the next frame and found to still be damaged, it is now marked in purple and the two frames where the damage were detected must be seen by an operator to assess the possible damage to that area.

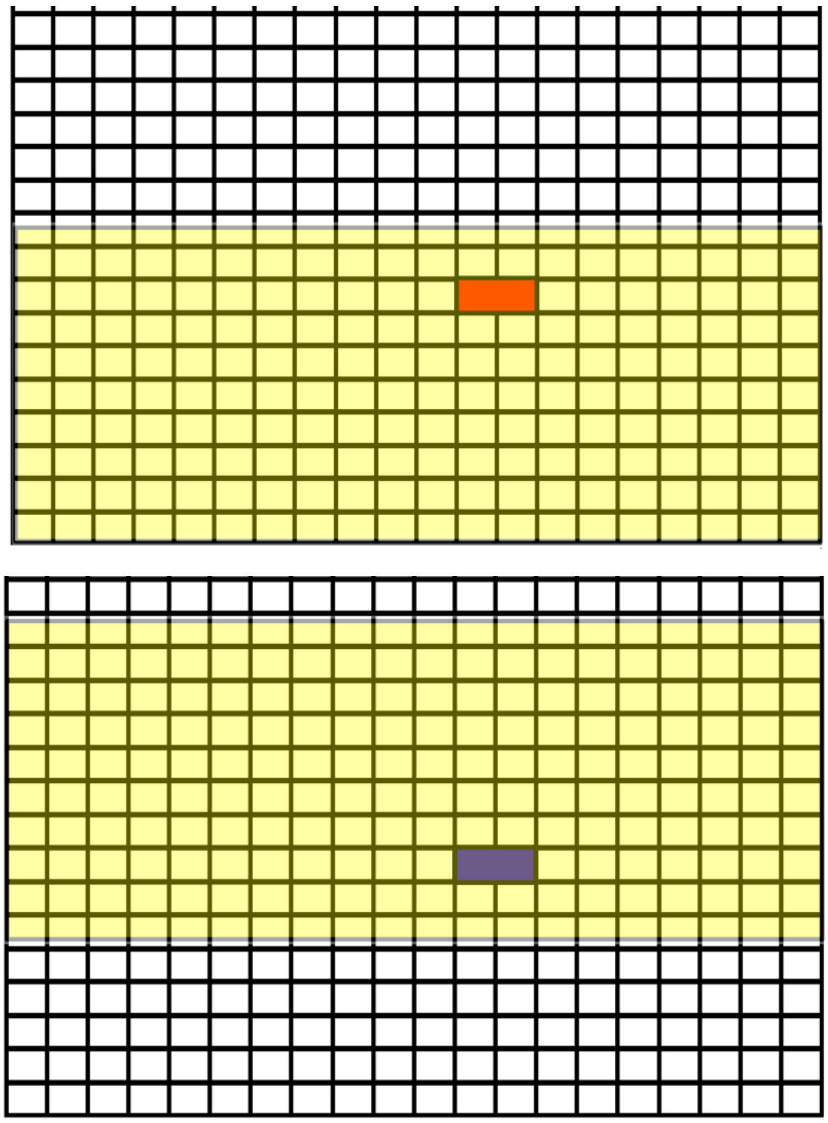


Figure 100: Future damage detection, first occurrence of small damage(top) and second occurrence of the same damage(bottom)

In one video obtained before prototyping of a net cleaning operation a small fish were present in the image. It was fairly stationary with respect to the ROVs camera so if after segmentation the fish would look like a hole that would be an example of a case where the sea cage would be believed to have a damaged area in one frame and when processing the next frame the is found to be whole. If a fish is swimming with the ROV in front of a part of the video frame the system detects a damaged area in one part of the frame repeatedly for many frames. The system should then assume that something is obscuring the camera and ignore that part of the image until the segmentation results improves.

When running the damage detection algorithm it can happen that the area value found to be the normal area suddenly jumps from for example 300 to 40. In the cases when this has happened it’s because the camera have lost focus resulting in the net not getting segmented properly. To prevent cases like this some memory should be added to prevent the system from changing the normal mask area value too much between two successive frames.

Alternatively if an external system is used to detect the distance from the ROV to the sea cage the normal sea cage mask size we should have could be calculated and known to the system before trying to find suspicious regions. Then the normal mask area would be known and the damage detection algorithm would be even more robust and stable.

Another thing one could use to improve the damage detection algorithm would be to use a cross-correlation, chain code or another approach to try and identify if a suspect region is a hole, a fish or simply some debris. To do this a big data set would be needed to give a potential identifying system a wide range of scenarios to compare the suspect region to.

Growth detection algorithm

To make the growth detection algorithm to work properly one needs to get a better segmentation of the net. One could also try other approaches than just counting foreground pixels. A possibility is to make a histogram over the area sizes found while running the damage detection algorithm and use the histogram to get information about the growths.

Other things to use the video stream for

When processing the video stream like it has been done in this thesis one can use the area sizes obtained during the damage detection algorithm to calculate the distance from the ROV to the net. This should work if the mask size is known beforehand, which in real life it is. There would be important to prevent the system from believing that for example a hole is a new mask sixe or that a part of the net that is partly occluded is a normal mask size so using the normal size found in the damage detection algorithm should make this approach more robust.

Reference list

Adobe Systems Incorporated. (2000). *CIEXYZ*. Hentet May 27, 2013 fra Adobe Technical Guides: http://dba.med.sc.edu/price/irf/Adobe_tg/models/ciexyz.html

Akvaplan-niva AS. (2005, June 30). Faktaark 2 Transport og løft av not. *30.06.05*.

Benjaminsen, C. (2012, January 12). *Small creatures do great damage*. Hentet May 21, 2013 fra Science Nordic: http://sciencenordic.com/small-creatures-do-great-damage

Direktoratet for naturforvaltning. (2012, September 17). *Lakselus*. Hentet March 26, 2013 fra miljøstatus.no: http://www.miljostatus.no/Tema/Ferskvann/Laks/Lakselus/

Eddins, S. (2006, october 4). *Steve on Image Processing*. Hentet June 5, 2013 fra MATLAB CENTRAL: http://blogs.mathworks.com/steve/2006/10/04/separable-convolution/

Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (2003). *Gaussian smoothing*. Hentet June 19, 2013 fra http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

Gonzalez , R., Woods, R., & Eddins, S. (2009). *Digital image processing using MATLAB.* Gatesmark publishing.

Gonzalez, R., Woods, R., & Eddins, S. (2009). Morphological Image Processing. I R. C. Gonzalez, R. E. Woods, & S. L. Eddins, *Digital Image Processing using MATLAB* (2nd. utg., ss. 486-534). Gatesmark Publishing.

Guenther, J., Fitridge, I., & Misimi, E. (2011, October). Potential antifouling strategies for marine finfish aquaculture: the effects of physical and chemical treatments on the settlement and survival of the hydriod Ectopleura larynx. *Biofouling: The Journal of Bioadhesion and Biofilm Research , 27*(9), 1033-1042.

Heide, M. A., & Moe, H. (2004). *Alternative notkonsepter - Delrapport i prosjekt "nye rømningssikre merdkonsepter"*. Trondheim: SINTEF Fiskeri og havbruk.

Hoffmann, G. (2013, Januray 31). *docs-hoffmann.de*. Hentet March 18, 2013 fra hoffmann.de: http://docs-hoffmann.de/cielab03022003.pdf

HunterLab. (2008). *Application notes*. Hentet March 18, 2013 fra hunterlab.com: http://www.hunterlab.com/appnotes/an07_96a.pdf

Jakobsen, R. H. (2011). *Automatic Inspection of Cage Intergrity with Underwater Vehicle.* Trondheim: NTNU- Departement of Engineering Cybernetics.

Jensen, Ø., Dempster, T., Thorstad, E., Uglem, I., & Fredheim, A. (2010, August 12). *Escapes of fishes from Norwegian sea-cage aquaculture: causes, consequences and prevention.* Retrieved february 17, 2013, from preventscape.eu: http://preventescape.eu/wp-content/downloads/2010_aei_jensen_et_al.pdf

Kerr, D. A. (2010, March 21). *The CIE XYZ and xyY Color Spaces.* Hentet March 11, 2013 fra http://dougkerr.net: http://dougkerr.net/pumpkin/articles/CIE_XYZ.pdf

Klima- og forurensningsdirektoratet. (2012, April 14). *Rømt oppdrettfisk.* Hentet March 26, 2013 fra miljøstatus: http://www.miljostatus.no/Tema/Hav-og-kyst/Fiskeoppdrett/Romt-oppdrettsfisk/

Laganière, R. (2011). *OPENCV 2 Computer Vision Application Programming Cookbook.* Birmingham: Packt Publishing Ltd.

Olsen, M. E. (2013). *Camera-assisted ROV navigation in sea cages.* IME, ITK. Trondheim: Norwegian University of Science and Technology.

OpenCV dev team. (2013, April 05). *OpenCV API Reference >> imgproc.Image Processing*. Hentet June 18, 2013 fra OpenCV 2.4.5.0 documentation: http://docs.opencv.org/modules/imgproc/doc/filtering.html?highlight=sobel#sobel

OpenCV dev team. (2013, April 05). *OpenCV Tutorials >> improc module >> Laplace Operator*. Hentet June 18, 2013 fra OpenCV 2.4.5 documentation: http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/laplace_operator/laplace_operator.html

Paschos, G., & Valavanis, K. P. (1999, February). A Color Texture Based Visual Monitoring System For Automated Surveillance. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS--PARTC: APPLICATIONS AND REVIEWS, 29, NO. 2,*, ss. 298-307.

Perkins, Fisher, Wolfart, & Walker. (2003). *Thininng*. Hentet June 25, 2013 fra http://homepages.inf.ed.ac.uk/rbf/HIPR2/thin.htm

Rønning, A. (2011, August 20). *Villaks-gener påvirket av oppdrett.* Hentet March 26, 2013 fra forskning.no: http://www.forskning.no/artikler/2011/august/295798

Schellewald, C. (2012, January 19). Computer Vision (TDT 4265) Lecture presentation. Trondheim.

Sonka, M., Hlavac, V., & Boyle , R. (2008). Image Processing, Analysis, and Machine Vision. I M. Sonka, V. Hlavac, & R. Boyle, *Image Processing, Analysis, and Machine Vision* (ss. 132-137). Cengage Learning.

Sonka, M., Hlavac, V., & Boyle, R. (2008). 13. Mathematical morphology. I M. Sonka, V. Hlavac, & R. Boyle, *Image processing, Analysis, and Machine Vision* (3rd. utg., ss. 657-693). Cengage Learning.

Sony. (2008). HD Color Camera Module - Technical Manual - FCB-H11. Sony.

Taylor, M., & Kelly, R. (2010). *Assessment of Protocols and Development of Best Practice Contingency Guidance to Improve Stock Containment of Cage and Land-based Sites Volume 1:Report.* Scottish Aquaculture Research Forum.

The Society of Motion Picture and Television Engineers. (u.d.). *SMPTE SDI Standards*. Hentet May 21, 2013 fra HDSDI High Definition Serial Digital Interface: http://www.hdsdi.info/smpte-sdi-standards/