NTNU

Norwegian University of
Science and Technology

Saliency based methods for camera orientation in aquaculture

Magnus Conrad Harr

Master of Science in Cybernetics and Robotics
Submission date: June 2018
Supervisor: Annette Stahl, ITK
Co-supervisor: Christian Schellewald, SINTEF Ocean

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem description

Today, cameras in sea-cages are controlled manually by an operator. From a computer-vision perspective this raises a few challenges, including how the camera orientation and position are unsupervised at night and whenever the operator is performing other tasks. For this reason, there is no guarantee that a camera system running computer-vision algorithms will be oriented optimally with respect to its desired target. As such, an algorithm able to automatically orient the camera such that interesting regions are always captured is desired.

This thesis aims to provide insight into a saliency based approach for such algorithms. Existing algorithms for this purpose are not suitable for separating interesting and non-interesting objects in a sea-cage. Therefore, modifications/additions to these algorithms will be tested. For the performance comparison, several saliency estimation techniques will be used, combined with different additions aiming to rectify the mentioned problems. The results will be based on footage from an underwater camera system developed by Sealab.

This project will lay the foundations for future 24/7 surveillance in sea-cages using computer-vision algorithms. Such algorithms will also provide an image quality guarantee to operators with remote system access, even when the site in unmanned.

In cooperation with Sealab AS, Trondheim. Contact: Oscar Markovic, CEO, oscar@sealab.no, +47 46912421

Preface

This report, written from January 4th 2018 to June 4th 2018, forms my masters thesis in engineering cybernetics at the Norwegian University of Science and Technology. Thank you to Annette Stahl and Christian Schellewald for agreeing to counsel me.

The work was done in collaboration with Sealab Ocean Group, who provided all necessary video material as well as an office space to work from. They have also, both during work on the masters thesis and the pre-project, provided knowledge on the aquaculture industry that has proven extremely useful for my work. I would like to thank the employees and my fellow students at Sealab for their input, making me consider things I may not have thought of otherwise.

Magnus Conrad Harr

June 4, 2018

Abstract

This thesis looks into the applicability of visual saliency as a basis for autonomous camera orientation. Autonomous re-orientation is expected to be a key component in future computer vision-based monitoring systems: a fish-cage is a highly dynamic environment, and even a well-placed camera may no longer be optimally oriented for data gathering if left static over time. The underwater performance of visual saliency algorithms are tested and discussed. A few different attempts at addressing a major concern related to the cage net itself are made. Most notable is a filtering scheme based on an optical flow algorithm by Gunnar Farnebäck, the use of which successfully solves the presented challenge but also introduces a new problem. In conclusion, visual saliency can provide a viable basis for camera orientation, but only under certain conditions.

Sammendrag

Denne oppgaven studerer anvendbarheten av visuell *saliency* for bruk til autonom kameraorientering. Automatisk re-orientering av kameraer antas å spille en viktig rolle i fremtidens datasyn-baserte overvåkningssystemer for oppdrettsmerder: fordi omgivelsene i en merd er svært dynamiske vil selv et velplassert kamera måtte justeres over tid for å gi optimal datafangst. Resultat av analyse av fremtredende objekter under vann testes og diskuteres. Det avdekkes en utfordring relatert til notveggen som forsøkes utbedret på flere ulike måter. Den viktigste er et filter basert på en optical flow algoritme utviklet av Gunnar Farnebäck. Bruk av dette filteret løser problemet relatert til notveggen, men innfører også et nytt. Det konkluderes med at automatisk kameraorientering vil kunne gjøres basert på visuell saliency, men kun under gitte betingelser.

Contents

1 Introduction **1**

2 Visual saliency **3**

2.1 Fine grained saliency 4

2.2 Motion saliency 5

2.3 Comparison and applicability 7

2.3.1 Share of fish considered salient 8

2.3.2 Resolution and accuracy 10

2.3.3 Dynamic range 12

2.4 Challenges 13

3 Saliency map filtering **15**

3.1 Thresholding 15

3.2 Masking based on frame logs 19

3.3 Optical flow 22

3.3.1 Motivation 22

3.3.2 Farnebäck 22

3.3.3 Applying Farnebäck 24

3.3.4 Threshold estimation using Otsu's method 26

3.3.5 Adaptive threshold 30

4 Discussion **33**

4.1 Frame log filter 33

4.2 Optical flow with Otsu filter 33

4.2.1 Fine grained saliency 34
4.2.2 Unprocessed video 37
4.3 Optical flow with adaptive filter 38
4.4 Optical flow filter comparison 40

5 Conclusion **43**

Appendices

A Unprocessed images **i**

B Farnebäck motion magnitudes **iii**

C Actual vs. perceived speed **v**

List of Figures

2.1.1 Illustration of fine grained saliency maps. In c) the fine grained saliency map is shown. Image by Montabone and Soto[17]. 5
2.1.2 Examples of underwater fine grained saliency maps, with differing camera angles. Unprocessed images are shown in appendix A. 6
2.2.1 Examples of underwater saliency maps using motion saliency, with differing camera angles. Computed from the same frames used in 2.1.2. Unprocessed images in appendix A. . 7
2.2.2 Illustration of motion saliency maps. Image by Wang and Dudek[3]. 8
2.3.1 Video frames (left), and their corresponding saliency maps (right). 11
2.3.2 Fine grained saliency of the cage net. 12
2.3.3 Two images from a sea-cage taken only a few seconds apart. 13

3.1.1 Side-by-side comparison of original saliency maps (left), and the results of applying a global threshold (right). 16
3.1.2 Original saliency map related to figure 3.1.3 17
3.1.3 Comparison of adaptive threshold with mean and gaussian kernels. $k = 31$. 18
3.2.1 Result of a frame log filter with $n = 2$. 20
3.2.2 Result of a frame log filter with $n = 5$. 21
3.2.3 Binary saliency map corresponding to figures 3.2.1 and 3.2.2. 21

3.3.1 Flow magnitudes estimated by Farnebäcks algorithm. Original frame in figure 3.3.4. 24
3.3.2 Example of a bimodal histogram. 26
3.3.3 Histogram (blue) showing the optical flow magnitude distribution from the image in figure 3.3.4. The red line marks the Otsu threshold. 28
3.3.4 Image from which the optical flow illustrated in figure 3.3.3 is estimated. 28
3.3.5 A fine grained saliency map (left), and a mask created by Otsu from the flow magnitudes (right). 29
3.3.6 A motion saliency map (left), and a mask created by Otsu from the flow magnitudes (right). 29
3.3.7 Mask created by adaptive threshold on flow magnitudes from figure 3.3.1. $k = 501$, $C = 10$. 31

4.2.1 Example frames for each case to be examined. 34
4.2.2 A fine grained saliency map from within a shoal of salmon (left), and a mask generated by the optical flow method (right). 35
4.2.3 Flow magnitudes related to the case with only fish. 35
4.2.4 A fine grained saliency map of a net (left), and a mask generated by the optical flow method (right). 36
4.2.5 Saliency map containing fish and a camera system (left), and its corresponding optical flow-based mask (right). . . . 37
4.2.6 Optical flow from original images with applied Otsu threshold. 38
4.3.1 Results of optical flow with adaptive threshold. Frame contains only fish. 39
4.3.2 Results of optical flow with adaptive threshold. Frame contains only net. 39
4.3.3 Results of optical flow with adaptive threshold. Frame contains fish, net, and a camera system. 40

A.0.1 Frame corresponding to the saliency maps in figures 2.1.2b and 2.2.1b. Time of day: 09:30 i

A.0.2Frame corresponding to the saliency maps in figures 2.1.2a and 2.2.1a. Time of day: 09:30 ii
A.0.3Frame corresponding to the saliency maps in figures 2.1.2c and 2.2.1c. Time of day: 15:30 ii

B.0.1Histogram of motion magnitudes related to figure 4.2.1a. Generated by Farnebäck optical flow. iii
B.0.2Histogram of motion magnitudes related to figure 4.2.1b. Generated by Farnebäck optical flow. iv
B.0.3Histogram of motion magnitudes related to figure 4.2.1c. Generated by Farnebäck optical flow. iv

C.0.1Field of view of DX format NIKKOR lenses [13]. *Photo by: Lindsay Silverman* v

Chapter 1

Introduction

Today, fish farms are generally monitored by operators by means of a single, manually controlled camera in each cage. In recent years, the aquaculture industry has seen a huge interest in modernization: several companies of all sizes have grown interested in joining this effort, especially related to artifical intelligence (AI) and computer vision[1, 8, 11]. Interest in this field is wide-spread, and both external companies as well as the fish-farming companies themselves are working on AI within aquaculture. One example of such a project is Aquacloud[5]. This is a joint project between several large actors within Norwegian aquaculture, aimed at analysing and predicting the development of salmon lice. The potential for savings is huge, as costs connected to salmon lice were estimated to 5 billion NOK in Norway alone in 2015[15].

The physical platform for data gathering can be realized in two ways. Either there will be added a separate camera to each cage, or the computer vision algorithms will be sharing a camera with the operators. Taking into account that operators on site report that there are already too many ropes and wires in and around the cages, it is likely that adding a separate camera for computer vision purposes will not be well received. Because of this, it is assumed here that any computer vision algorithms will be run on the same camera controlled by on-site operators. This poses a new challenge; people working on the farming installations are unlikely to have a background in computer vision or AI. They are therefore unlikely to know

how to position and orient a camera when they are not using it, in order to maximize video quality for the computer vision systems. In this thesis, it will be discussed whether or not a saliency-based approach for automatically orienting cameras in a sea-cage is viable. The findings will be an indication as to how well visual saliency would work as a basis for an orientation algorithm.

In chapter 2, two different saliency algorithms will be presented and compared. The comparison will focus on how each saliency algorithm performs in a sea-cage environment, and how well they are expected to perform as a basis for camera orientation. Chapter 3 introduces several operations performed on the saliency maps in an attempt to improve the data. Most notable of these is an optical flow-based filter based on Gunnar Farnebäcks algorithm presented in the 2003 paper *"Two-Frame Motion Estimation Based on Polynomial Expansion"*[7]. In chapter 4 the findings will be discussed and evaluated with respect to camera orientation.

The end result of this thesis will be a ruling into the pros and cons of basing a camera orientation scheme on existing saliency algorithms. Limitations and pre-requisites for the approach will be taken into account.

All software used throughout the project was implemented in C++ using OpenCV 3.3. Notable OpenCV modules that were used include libraries for visual saliency estimation, optical flow, background segmentation, and CUDA image filtering.

Chapter 2

Visual saliency

This chapter explains the concept of visual saliency, and provides an explanation of the differences between two algorithms used in this project. Some parts of the theory and explanations in this section were covered in the pre-project for this thesis[10]. However, they will be covered here as well for ease of reading, and because the pre-project is not yet publicly available.

Saliency algorithms are generally meant to estimate how interesting each pixel in an image is. Research into such algorithms are under no means finished, and several algorithms have been presented in recent years[3, 9, 17, 19]. The problem of detecting salient image regions is one that can be solved in many different ways; attempts have been made using a variety of different approaches, including background subtraction[3], and contrast evaluation[9]. The algorithm designed by Montabone and Soto in 2010 is based on convolution of differing filter-sizes, a center-surround computation, and spatial evaluations to compute pixel saliency[17].

In this project, two algorithms will be used: Montabone and Sotos fine grained saliency presented in 2010[17], and a motion saliency algorithm by Wang and Dudek from 2014[3]. As these are presented, note how the example photos provided by the algorithms developers are taken in air. Because this project is concerned with underwater images from fish-cages, the image material is very different and it is expected that the saliency

algorithms will perform differently. As a result, the algorithms must be evaluated based on images from a sea-cage.

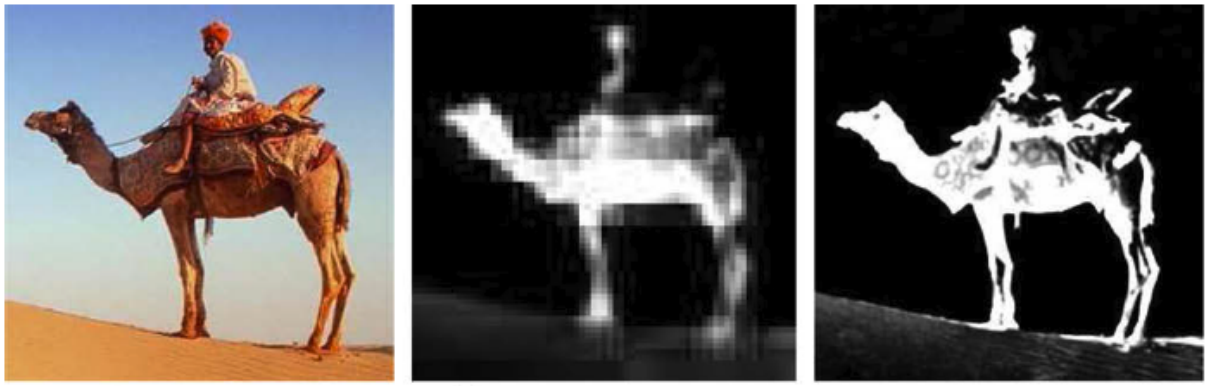
2.1 Fine grained saliency

The fine grained algorithm by Montabone & Soto, also referred to as *fine grained saliency*, is really an improvement on VOCUS[16, 17]. As explained in the pre-project [10]:

> "In VOCUS, given the original frame i_0, a $3x3$ Gaussian filter is applied and the image is scaled down to half the size on each axis to create i_1. This is repeated 4 times to create images i_0, i_1, i_2, i_3, i_4. From these images, 12 intensity submaps are generated in 6 filter windows and then summed up".

In VOCUS, computation over a filter of size *n-by-n* is $\mathcal{O}(n^2)$[16, 17]. The improvement made in *fine grained saliency* is that integral images are used for feature evaluation, meaning the same computation can be performed in $\mathcal{O}(1)$[17]. This allows to avoid reducing image resolution by instead increasing the filter sizes. The end result is that *fine grained saliency* creates a higher resolution saliency map than VOCUS, as can be seen in figure 2.1.1. The fact that this algorithm can provide such high-resolution saliency maps means more potentially important image information is retained.

The algorithms performance in air is illustrated in figure 2.1.1, while the underwater performance can be seen in figure 2.1.2. There is a clear difference between these two examples: In the example image of a camel a large portion of the animal is considered highly salient, whereas in a sea-cage the fish are considered significantly less salient. From this, the algorithm has its pros and cons: On the positive side, the high resolution displayed in air is retained in an underwater sea-cage environment. This implies that the fine grained saliency map will be well suited for any further processing[10]. A negative effect arises from the fact that pixel saliency varies greatly over a given salmon: In figures 2.1.2a and 2.1.2c it is mostly the fins and dot



(a) Original Image (b) VOCUS Off-Center Surround (c) Proposed Off-Center Surround

Figure 2.1.1: Illustration of fine grained saliency maps. In c) the fine grained saliency map is shown. Image by Montabone and Soto[17].

pattern on the backs of fish that is considered salient, meaning the average saliency over a given fish is low. The total saliency of a fish is significantly lower than optimal, which may prove a challenge for camera orientation. In an ideal case, every pixel representing part of a salmon would have max intensity in the saliency map.

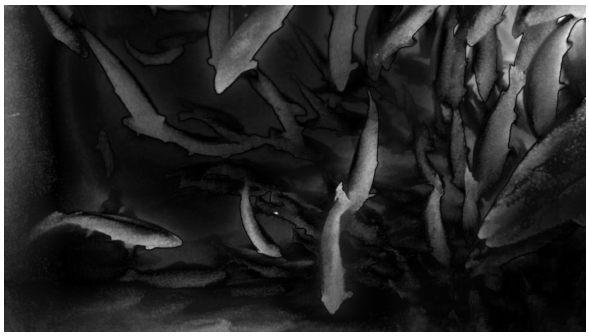
2.2 Motion saliency

The algorithm by Wang & Dudek referred to as a motion saliency algorithm is really based on a background subtractor. As explained in the pre-project[10]:

> "Wang and Dudeks background subtraction algorithm "follows the general scheme of a pixel-based background detector"[3]. It uses a set of adaptive templates for the background model, and selects which to discard based on an efficiency measure. This approach means less computation is required compared to algorithms using larger sets of values, and therefore makes it suitable for use on video feeds and in systems with limited computational power".

The underwater performance of motion saliency is shown in figure 2.2.1. Compared to the authors examples in figure 2.2.2, this shows that saliency

(a) Facing down 45°



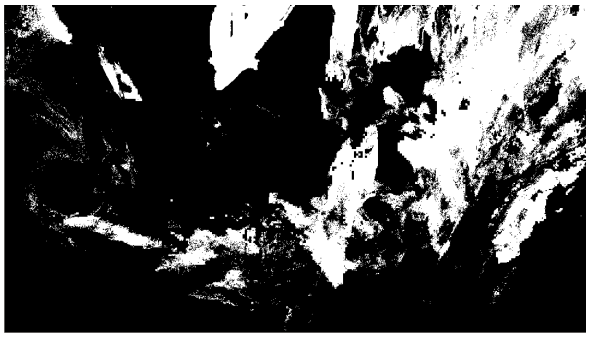
(b) Facing up 45°

(c) Facing horizontal

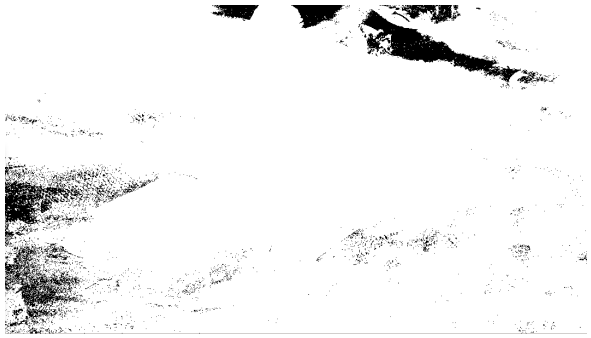
Figure 2.1.2: Examples of underwater fine grained saliency maps, with differing camera angles. Unprocessed images are shown in appendix A.

maps produced by this algorithm in a fish-cage is significantly less accurate than in air, and contain more noise. This additional noise originates from the fact that this algorithm is essentially a background subtractor. All the example images in figure 2.2.2 have clear, textured, perfectly static backgrounds, whereas in a sea-cage, as can be seen from the unprocessed images in appendix A, determining what is background and foreground is not as straight forward. First of all, the true background in a fish cage is a body of water and has very little texture, if any. Second, as the fish appear at varying distances from the camera there is intuitively some distance limit where fish should be considered background instead of foreground. This is because studying fish far from the camera is not productive, as too little detail is available (assuming a resolution of 1920x1080 pixels). Finally, as the distance between the camera and the fish increases, the increasing amount of water between camera and fish means the fish will gradually appear less clear because of particles suspended in the water[20]. These three phenomenon make accurately detecting salient image regions a diffi-

(a) Facing down 45°



(b) Facing up 45°



(c) Facing horizontal

Figure 2.2.1: Examples of underwater saliency maps using motion saliency, with differing camera angles. Computed from the same frames used in 2.1.2. Unprocessed images in appendix A.

cult task for a background subtraction algorithm, which in turn introduces significant noise in the resulting saliency map.

2.3 Comparison and applicability

There are three main topics to consider in this section: 1) The share of "fish-pixels" considered salient, 2) saliency map resolution, and 3) the dynamic range of the total amount of salient pixels in a given image. The comparison in this section will provide a preliminary indication as to which saliency algorithm is better suited as a basis for camera orientation.

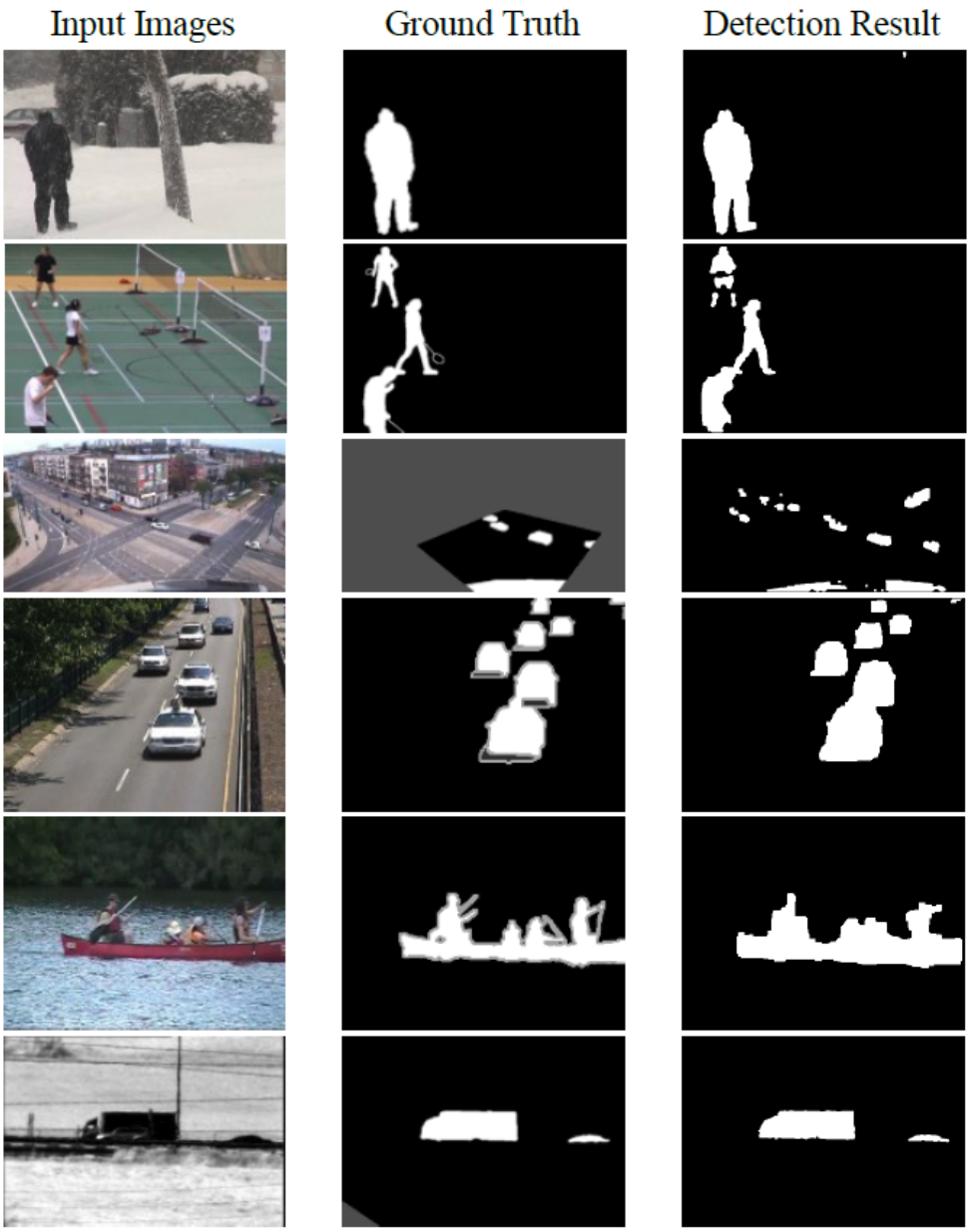


Figure 2.2.2: Illustration of motion saliency maps. Image by Wang and Dudek[3].

2.3.1 Share of fish considered salient

Naturally, the share of fish considered salient in an image is one of the main performance factors for a saliency algorithm to be used for camera orientation. For an automatic orientation scheme to successfully orient toward fish, the saliency algorithm it is built on must reliably classify fish as salient regions. It can however be claimed that this parameter is not actually critical: If the saliency estimator only determines the salmons head as

salient, an orientation algorithm focused on wherever the saliency density is highest would still orient the camera toward max fish density. From this perspective it can be argued that this parameter in itself is not important, and as long as only salmon are considered salient there is no problem. However, as will be discussed in detail later, saliency estimators in a sea-cage also define other objects as salient regions. Additionally, the case where only a given body-part is salient could prove challenging because from the cameras two-dimensional point of view the fish appear in layers. This can be seen from the unprocessed images in appendix A, where parts of fish are covered behind other fish. For maximum reliability, a saliency-based orientation scheme should therefore be based on a saliency algorithm that considers every part of a fish salient.

Secondly, this parameter affects another factor for a reorientation algorithm. Assume there exists a "perfect" saliency algorithm which accurately classifies every pixel on every fish as salient. This will have some major implications for total saliency in a given frame, based on how the fish is positioned relative to the camera: when fish appear as larger objects it means more image pixels will be salient. Any fish that is rotated 90° relative to the camera, with the camera pointing at its side, will appear as larger objects and represent a much larger total saliency compared to a fish swimming toward or away from the camera. Likewise, a fish that is closer to the camera will cover a larger image region than one that is further away. These will also represent a greater total saliency. The implications of this is that an orientation algorithm will prefer to orient such that the fish are close, and swimming perpendicular to the camera.

Is this desired functionality? The short answer: Yes. Recall from section 1, that the main purpose of a camera orientation system in a sea-cage will be to improve the data gathered for other computer vision algorithms. Examples of such algorithms can be lice detection, gill motion frequency estimation, or recognition of individuals. All these example cases are solved more easily given footage where the fish are close to the camera, as more detail is visible in the video. Also, because salmon lice tend to attach to

the dorsal and pectoral fins of the fish[4, 12], this behaviour in a saliency estimator is expected to provide better videos with respect to salmon lice detection as well.

By comparing figures 2.1.2 and 2.2.1, it is clear to see that this demand is best satisfied by motion saliency, as motion saliency classifies a significantly larger portion of the fish salient. Figure 2.2.1c exemplifies how much total saliency the motion saliency algorithm can attribute to a frame from within a shoal. It is worth noting that this saliency map is taken just 10 frames after a fish passed in front of the camera, close enough that it covered the entire frame. The center region of the image where there are no fish is still considered salient because it was salient a few frames earlier. This trailing effect originates from the fact that the motion saliency algorithm is based on a background subtractor. On the other hand, from figure 2.2.1a it becomes clear that neither this algorithm is perfect, as it indicates that motion saliency defines the backs of fish to be non-salient.

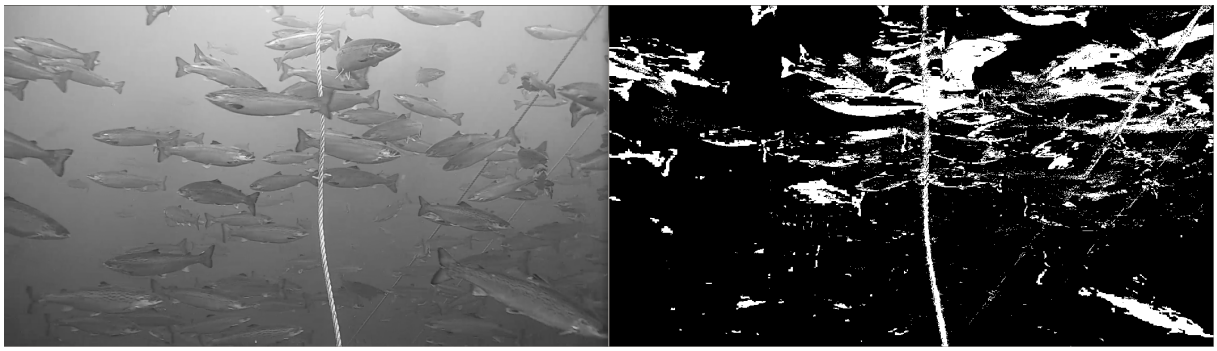
Finally, note how the fine grained saliency algorithm is better at detecting fish that are far from the camera, even though the attributed saliency is low. For some purposes this may be interesting, but for camera orientation it is less likely to be relevant. As mentioned previously, it is desired to orient the camera such that fish are as close to the camera as possible. With this in mind, it is intuitively less important to detect fish far away. In conclusion, the superior performance of Wang and Dudeks motion saliency algorithm when fish are close to the camera outweighs the fact that fine grained saliency is better at range.

2.3.2 Resolution and accuracy

The resolution of the saliency map may not intuitively seem like an important factor for this type of application. However, it was shown in the pre-project that in specific cases a high-resolution saliency map allows post-processing to make major improvements on the data for an orientation al-

gorithm[10]. One such case; whenever there are objects in the image that are not fish. Such objects includes the net itself, ropes going into the cage, and any additional pieces of technology submerged in the cage. As can be seen from figures 2.3.1 and 2.3.2, these objects are evaluated as salient. This fact will prove to be a challenge.

(a) Fine grained saliency



(b) Motion saliency

Figure 2.3.1: Video frames (left), and their corresponding saliency maps (right).

A high resolution is expected to ease the process of addressing the problem illustrated in figure 2.3.2. When this image is compared to any other fine grained saliency map of fish, it is clear that Montabone & Soto's algorithm will generate saliency maps such that more salient features exist in an image of a net than a shoal of salmon. Ultimately, this could lead an orientation algorithm to orient the camera such that the video never contains anything but the cage net. Different approaches have been attempted to address this issue, which will be explained in detail in chapter 3. It will become clear that a high-resolution saliency map facilitates more methods of filtering out

these unwanted regions, making an orientation algorithm ignore them. The fact that this point was made clear in the pre-project, is one of the reasons why the work on this thesis focused on the algorithms by Montabone & Soto, and Wang & Dudek: they both produce saliency maps of the same resolution as the original images.

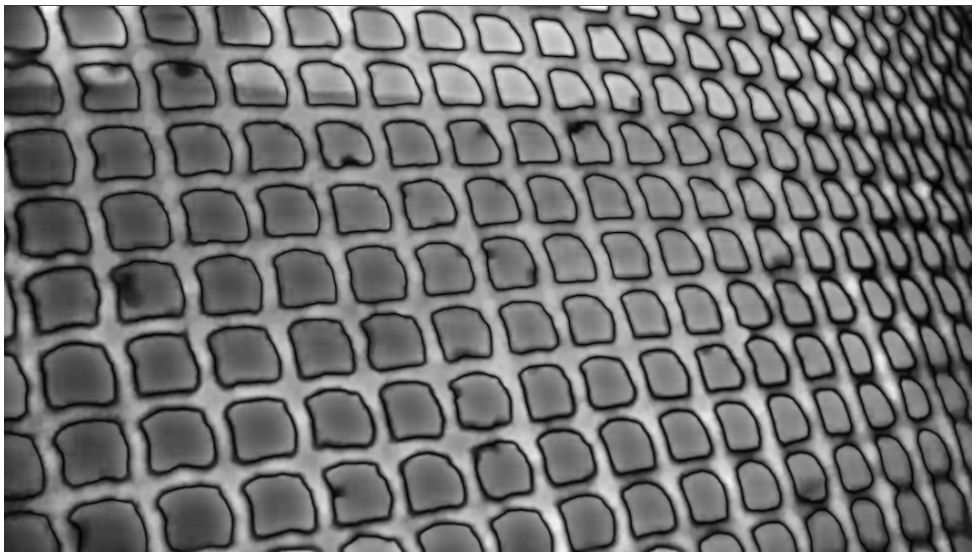


Figure 2.3.2: Fine grained saliency of the cage net.

Another point to be made is the accuracy of the saliency map. In other words, the probability that a salient pixel is on an object intended for study. This parameter must be considered because some post-processing methods, including the local binary pattern texture operator used in the pre-project[10] and the optical flow method described in chapter 3.3, analyse every salient pixel. Inaccuracies and noisy saliency maps may result in unwanted behaviour in these filters. By looking at figure 2.1.2c, it is clear that the *fine grained* algorithm is far superior in this regard. Figure 2.1.2c contains far less "misses" than figure 2.2.1c.

2.3.3 Dynamic range

In photography, dynamic range refers to the range of light intensities that are captured in an image[2]. For saliency maps however, we will use it describe the ratio between the lowest and highest amount of salient pixels

in a saliency map. In the context of a sea-cage, it becomes a measure for how much the total saliency in a frame increases as fish cover larger portions of the image. While this parameter can be viewed as a subset of the *share of fish considered salient* from section 2.3.1, it is important to consider for one specific reason: The environment is highly dynamic, and the position of the camera relative to the shoal varies over time. Especially if the camera is moved by an operator. An example of how large the differences can be, even without moving the camera, can be made by comparing the images in figure 2.3.3. It is clear that in figure 2.3.3a, where the fish are closer to the camera and cover a larger portion of the image, the number of salient pixels should be comparatively higher than for figure 2.3.3b. Based on figures 2.1.2c and 2.2.1c, Wang & Dudeks motion saliency better fulfils this requirement.

(a) (b)

Figure 2.3.3: Two images from a sea-cage taken only a few seconds apart.

2.4 Challenges

Consider figures 2.3.1 and 2.3.2, illustrating how saliency algorithms tend to consider ropes and net highly salient. This is a problem because of scenarios where the saliency estimators evaluate the net as more salient than the fish. A camera orientation algorithm based on this data would likely turn the camera such that the video contains only net. This must be avoided. During the pre-project for this thesis, an attempt at rectifying this was made by filtering contours based on the local binary pattern texture operator[10]. It was shown that this worked to some extent, but

not well enough to be viable for actual use. In chapter 3, a few other approaches to solving this problem will be presented.

It is expected that the output of an orientation estimator based on visual saliency will return a desired orientation that changes constantly. Again, this originates from the fact that the environment is highly dynamic. It can be assumed that the distribution of salient pixels in the image will vary greatly, especially when fish are close to the camera. Figure 2.3.3 contains two frames showing how much this can change over the scope of just a few seconds. A camera that is constantly oriented toward some measure of "max saliency", is likely to rotate in such a way that it appears unstable. Intuitively, for the purposes of optimizing the data gathering for other computer vision algorithms, it is desired that the camera be as stable as possible and move only when necessary.

Chapter 3

Saliency map filtering

This section describes the the attempts made at creating filters that would address the challenge described in section 2.4 related to salient cage nets.

3.1 Thresholding

Image thresholding is the operation of transforming an image into a binary one. This section describes how the fine grained saliency maps with a depth of 8 bits, is transformed to a binary map. While this operation is not needed to evaluate the saliency itself, it facilitates further processing of the saliency maps.

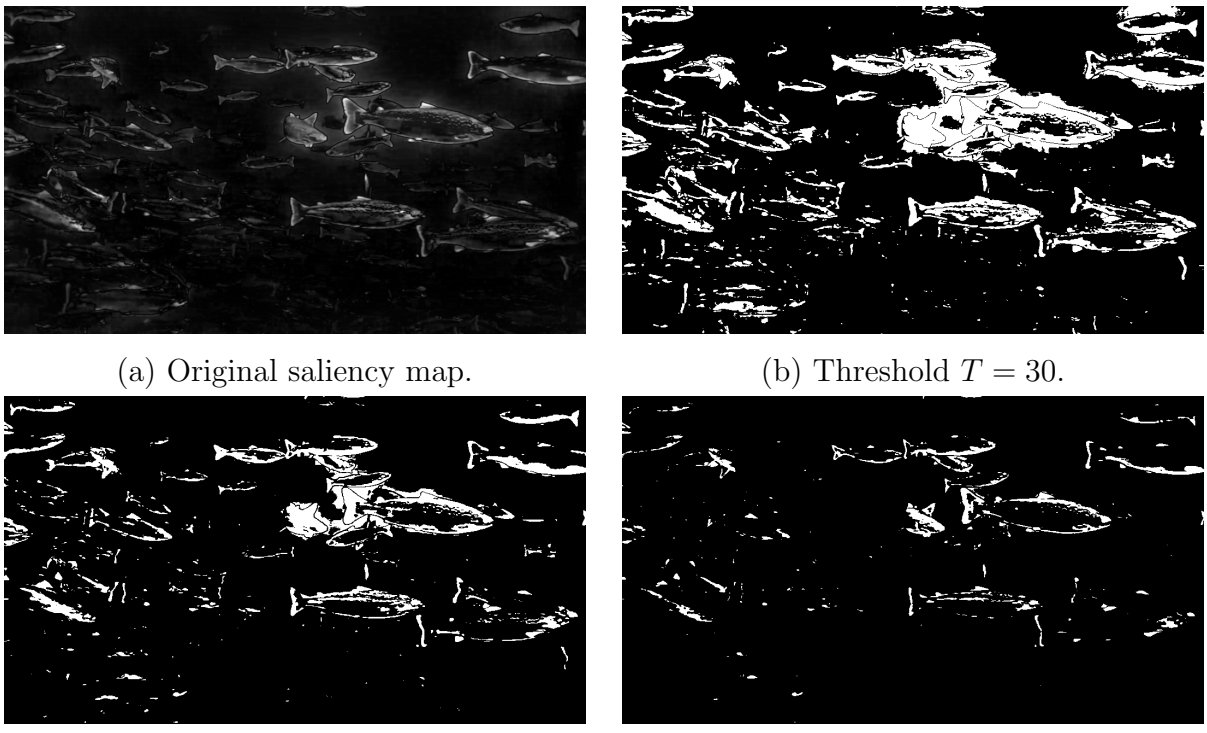
An intuitive method for binarizing an image is by applying a global threshold. This operation can be described by (3.1), where $maxValue$ is the maximum possible pixel intensity in I.

$$I(x,y) = \begin{cases} maxValue, & src(x,y) > T \\ 0, & otherwise \end{cases} \tag{3.1}$$

Results of this global threshold can be seen in figure 3.1.1. Consider first figure 3.1.1b: Most of the fish in the image are included to some degree. Also a considerable area around some fish are included in the binary saliency map. When increasing the threshold T, as illustrated in figures 3.1.1c and 3.1.1d, less unwanted area around the fish is included, at the

cost of also including less of the fish themselves. It is also noticeable how the fish in the upper half are considered more salient than those in the bottom half: these images are taken at only a few meters depth, which means the background color is lighter in the upper half of the image. The fish in the bottom of the image are therefore considered less salient, mostly because they are located in front of a darker background than those in the top of the image, and there is therefore less contrast.

Because significant areas of the fish must be excluded in order to avoid including unwanted regions, a global threshold is not suited for binarizing underwater saliency maps as it fails to uphold the primary requirement from section 2.3.1. Ideally, a method that is capable of keeping as many fish contours as possible while excluding any area around the fish should be employed.



(a) Original saliency map.

(b) Threshold $T = 30$.

(c) Threshold $T = 50$.

(d) Threshold $T = 80$.

Figure 3.1.1: Side-by-side comparison of original saliency maps (left), and the results of applying a global threshold (right).

To address this problem an adaptive thresholding scheme was applied. This

can be described by (3.2).

$$I(x,y) = \begin{cases} maxValue, & src(x,y) > T(x,y) - C \\ 0, & otherwise \end{cases}, \tag{3.2}$$

where $T(x,y)$ can be calculated either as 1) the average pixel intensity in a k-by-k neighbourhood around *(x,y)*, or 2) a weighted sum of the pixel intensities in a k-by-k neighbourhood using a Gaussian kernel as in (3.3). The standard deviation of the Gaussian kernel is computed using (3.4)[6].

$$G_i = \alpha * e^{-(i-(k-1)/2)^2/(2*\sigma^2)} \tag{3.3}$$

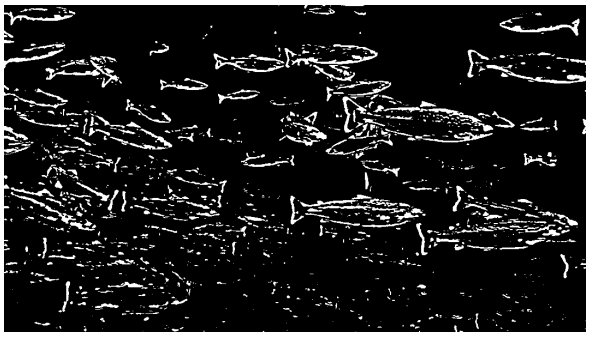
$$\sigma = 0.3 * ((k-1) * 0.5 - 1) + 0.8. \tag{3.4}$$

In (3.3), $i = \{0, 1, ..., (k-1)\}$ and α is a constant such that $\sum_i G_i = 1$.

Figure 3.1.2: Original saliency map related to figure 3.1.3

.

This method was applied with both mean and Gaussian kernels. Intuitively, from comparing (3.1) to (3.2), the adaptive threshold will generate a very different binary image if the original image has varying average intensity. Based on the saliency map generated by the fine grained saliency algorithm in figure 3.1.2, where the average pixel intensities on fish and



(a) Gaussian kernel, $C = 10$.

(b) Mean kernel, $C = 10$.

(c) Gaussian kernel, $C = 15$.

(d) Mean kernel, $C = 15$.

Figure 3.1.3: Comparison of adaptive threshold with mean and gaussian kernels. $k = 31$.

background varies depending on image region, it is expected that an adaptive threshold will include a greater fraction of the fish than the previously discussed global threshold[18].

Figure 3.1.3 demonstrates the result of applying an adaptive threshold on the same images as in figure 3.1.1. Not only are a larger fraction of the fish included after binarization, but there are very few regions included in the binary map that do not represent fish. The binary maps include multiple small white regions in the bottom quarter of the image; however, these also map to contours of salmon that are barely visible in the original saliency map. It is clear that an adaptive threshold is better suited than a global one.

3.2 Masking based on frame logs

This section will discuss the applicability of a filtering solution based on masking out static contours. This type of filter depends on the binary saliency maps generated by the threshold operations from section 3.1. Assume, initially, that the images are made up of two classes of contours: *static* and *moving*. Also assume that fish are always swimming, and that static contours mainly represent ropes, nets, or other objects that should not be considered by an orientation algorithm. From these assumptions it can be reasoned that an algorithm for computing optimal camera orientation should include moving contours while ignoring any static ones. An intuitive approach for removing static contours was attempted by performing the following operations:

Given the images $\{i_1, i_2, ..., i_n\}$, that make up a log of inverted binary saliency maps from the n previous frames in the video feed, and frame i representing the binary saliency map of the current frame.

1. Create an inverse of i and call it i_{inv}.
2. Push the frame log such that $i_k \rightarrow i_{k+1}\ \forall\ k\ \epsilon\ \{1, n-1\}$, and set $i_1 = i_{inv}$.
3. Let $i_{mask} =$ *fully black frame.*
4. $i_{mask} = bitwise\_or(i_{mask}, i_k)\ \forall k\ \epsilon\ \{1, n\}$.
5. $i_{filtered} = bitwise\_and(i_{mask}, i)$.

This 5-step algorithm creates a mask i_{mask} such that any pixel that is white in any of the inverted frames $\{i_1, ..., i_n\}$ is also white in i_{mask}. Or, more intuitively, i_{mask} is a mask where a given pixel is white only if the same pixel has been black in any of the last n binary saliency maps. When i_{mask} is used to mask the binary saliency map i in step 5, it removes any pixels in i that have been white for n or more consecutive frames. Ideally, this means any contours that have not moved in the last n frames will be

removed from the binary saliency map.

Figure 3.2.1: Result of a frame log filter with $n = 2$.

As can be seen in figure 3.2.1 the results achieved in a sea-cage are far from perfect. This originates from the fact that in a sea-cage the objects previously assumed to be static are not perfectly static from the cameras point of view. Because of the physical setup the camera is always in motion due to forces from waves, sea currents, collisions with fish, etc. This means that, from the cameras point of view, any objects not moving perfectly synchronized with the camera, are non-static. Perfectly synchronized motion is highly unlikely, so it is realistic to say that for a camera in a sea-cage there are no perfectly static contours. In figure 3.2.1 it is shown how this manifests as regions without fish still being included in the filtered image. Notice how even though n is set as low as 2 frames, neither the other camera system or the net in the bottom right corner is properly filtered out.

This approach proves to be even less applicable when a comparison is made between the contours of salmon in the filtered image and the unfiltered image in figure 3.2.3. Notice how the attempt to remove the static objects also removes significant regions of the fish. This is because the frame log length n is so small that even the moving contour of a fish holds the same pixels white for more than n consecutive frames. A way to address this

issue would be to increase n, which would improve upon the problem of filtering out part of the fish but also introduce a drawback. As n increases, i_{mask} becomes a more liberal filter because the maximum allowed movement for an object to be considered stationary is proportional to n. This is illustrated in figure 3.2.2, where the camera system is much more visible compared to figure 3.2.1.

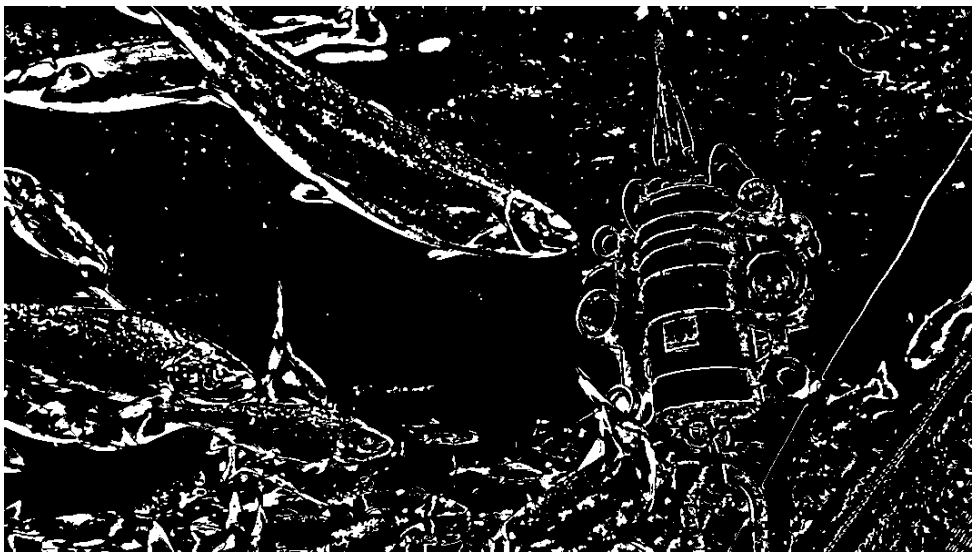


Figure 3.2.2: Result of a frame log filter with $n = 5$.

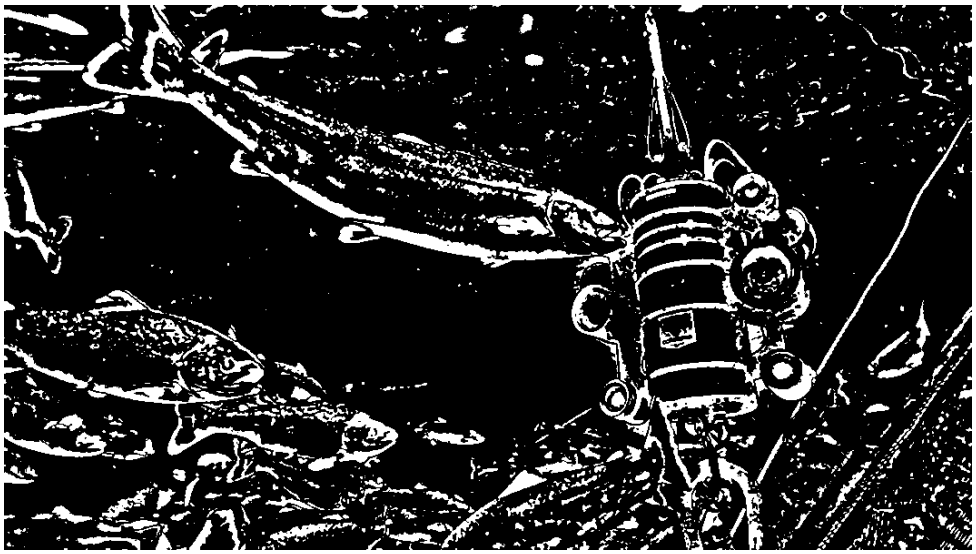


Figure 3.2.3: Binary saliency map corresponding to figures 3.2.1 and 3.2.2.

3.3 Optical flow

This section explains the motivation for using optical flow in this project, the theory behind the chosen algorithm, why this specific algorithm was chosen, and how it was implemented. It will also detail the implementation of two optical flow-based filters for addressing the issue of removing unwanted objects from saliency maps.

3.3.1 Motivation

Optical flow was used in this project in an attempt to address a challenge mentioned in section 2.4, where it is described how an orientation algorithm based on raw saliency maps is likely to prioritize nets over fish. The idea behind using optical flow for this purpose is based on the assumption that the fish will always move, and that salmon close to each other in a shoal is likely to display similar motion. The goal for using optical flow was to be separate salmon from any non-fish objects in the cage. The hope was that an optical flow-based filtering scheme would perform better than the masking approach from section 3.2.

Because the purpose of optical flow in this project is to separate fish from other objects, the algorithms accuracy is not critical. The core parameter is the resolution of the optical flow estimator, and with that, its ability to estimate flow in every pixel. Because of this, a polynomial-based algorithm by Gunnar Farnebäck was used.

3.3.2 Farnebäck

This algorithm was proposed by Gunnar Farnebäck in 2003 in the paper *"Two-Frame Motion Estimation Based on Polynomial Expansion"*[7]. As explained in the abstract of the conference paper:

> "The first step is to approximate each neighborhood of both frames by quadratic polynomials, which can be done efficiently using the polynomial expansion transform. From observing how

> an exact polynomial transforms under translation a method to estimate displacement fields from the polynomial expansion coefficients is derived and after a series of refinements leads to a robust algorithm."

The quadratic polynomial in question is represented on the form

$$f(\mathbf{x}) \sim \mathbf{x}^T \mathbf{A}\mathbf{x} + \mathbf{b}^T \mathbf{x} + c \tag{3.5}$$

where $f(\mathbf{x})$ is a polynomial approximation to a neighbourhood around a given pixel. The optical flow approach is based on the idea that motion is estimated by comparing the difference between two subsequent frames. By defining the frames such that

$$\begin{aligned} f_1(\mathbf{x}) &= \mathbf{x}^T \mathbf{A}_1 \mathbf{x} + \mathbf{b}_1^T \mathbf{x} + c \\ f_2(\mathbf{x}) &= f_1(\mathbf{x} - \mathbf{d}) \end{aligned} \tag{3.6}$$

it is shown by Farnebäck[7] that solving the equation with respect to the displacement $\mathbf{d}$, gives

$$\mathbf{d} = -\frac{1}{2}\mathbf{A}_1^{-1}(\mathbf{b}_2 - \mathbf{b}_1) \tag{3.7}$$

$$\mathbf{b}_2 = (\mathbf{b}_1 - 2\mathbf{A}_1\mathbf{d})^T. \tag{3.8}$$

However, this solution is based on an assumption that the signal in an entire image can be described as a single polynomial. According to Farnebäck, this is an unrealistic assumption. In his paper, he goes on to detail how to get around this issue by introducing approximations of $\mathbf{A}$, estimating $\mathbf{A}$ over local neighbourhoods, and presenting parametrized equations for the displacement fields. This will not be detailed here. From the results presented in Farnebäcks paper it is clear that with respect to accuracy, this algorithm is inferior to other alternatives. For this project however, as explained in section 3.3.1, this is not critical.

Farnebäcks dense optical flow algorithm creates a matrix $\boldsymbol{A}$ with the same dimensions as the source image, where each cell holds the optical flow

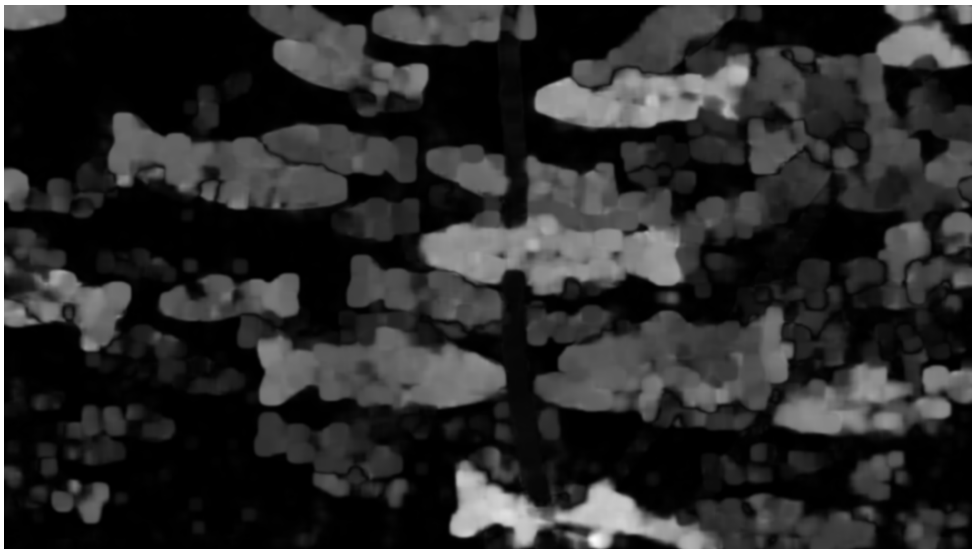


Figure 3.3.1: Flow magnitudes estimated by Farnebäcks algorithm. Original frame in figure 3.3.4.

information of the corresponding pixel as a set of two values, (x, y). With this, both angle and magnitude information is available. For the purposes of this thesis it was deemed that the angle information is of little value as it is difficult to know what types of objects are moving in different directions. This is also outside the scope of this project. Hence, only the magnitude of the optical flow vector is considered. The flow magnitude image I_{FM} illustrated in figure 3.3.1 is created by

$$I_{FM}(i,k) = \sum_{i=0}^{rows(A)} \sum_{k=0}^{columns(A)} \sqrt{A_x(i,k)^2 + A_y(i,k)^2}, \tag{3.9}$$

where A is the optical flow result matrix.

3.3.3 Applying Farnebäck

Applying Farnebäck to saliency maps in this project includes two separate cases, as there are two saliency algorithms this may be based on. Consider the two saliency maps in figure 2.3.1, which will now be used as a basis for this discussion. The fine grained saliency maps are of depth 8 bits, meaning pixel intensities are anywhere between 0 and 255. This, combined with the high-resolution nature of the saliency map, suggests that optical flow

should perform well on these. This is because fine grained saliency maps contain enough texture in local neighbourhoods for optical flow algorithms to reliably detect and track features. While Farnebäcks algorithm, as described in section 3.3.2, is not based on feature-tracking in and of itself, textured neighbourhoods are necessary for the polynomials describing local neighbourhoods to contain useful information: a polynomial describing a uniform region, will simply be a constant. As such, a textured saliency map is important.

Running Farnebäck on a stream of motion saliency maps yields vastly different results. As mentioned previously, the ability to estimate the motion of a neighbourhood of pixels requires the intensity distribution in said neighbourhood to be non-uniform. The saliency maps generated by Wangs & Dudeks motion saliency algorithm are binary[3], as illustrated in figure 2.2.1. Clearly, the only pixel-neighbourhoods in a binary saliency map with a non-uniform intensity distribution are along the edges of salient contours. This means Farnebäck will return a flow matrix where the estimated motion of any pixel is 0, except along the edges of contours. So for motion saliency, another approach was attempted to work around this problem. Instead of running optical flow on the saliency maps, the optical flow is instead run separately on the unprocessed images and applied as a mask to the saliency map. This way, an optical flow algorithm should provide a more accurate result.

The hope was that the magnitude of the optical flow can be used to determine which parts of the image should be included or not, based on similar logic as in section 3.2. However, with the optical flow approach one faces the same challenge as with the simple bitwise comparison method: Because nothing is ever stationary from the cameras point of view, and the perceived motion of the fish will vary, a static threshold for the motion magnitude cannot be defined. Instead a self-adjusting threshold needs to be implemented to include the faster objects while ignoring the slower ones. For this task, self-adjusting methods for intensity thresholding were applied on the flow magnitude images such as in figure 3.3.1.

3.3.4 Threshold estimation using Otsu's method

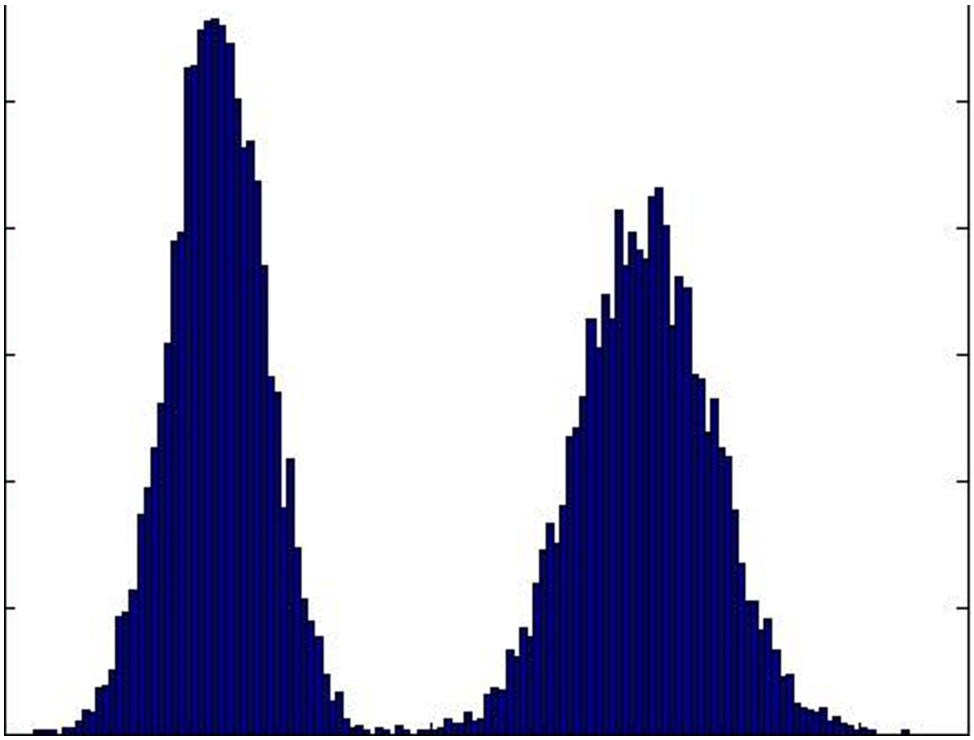


Figure 3.3.2: Example of a bimodal histogram.

Otsu's method was implemented for processing the results of optical flow estimation, based on the idea that there are only two classes to be considered in this scenario: Moving and semi-stationary. Otsu's method is originally a method for optimal threshold estimation for global thresholding of bimodal images[14]. That is, where the image contains two distinct classes. In this context, a class is the set of pixels with intensity above/below the the threshold value. An example of a bimodal histogram is illustrated in figure 3.3.2, in which case Otsu's method would place the threshold intensity at the lowest point between the two peaks. This is done by searching for a threshold t that minimizes the sum of within-class variances of the two classes. This can be expressed mathematically as minimizing (3.10).

$$\sigma_\omega^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t), \tag{3.10}$$

where

$$\begin{aligned} \omega_0(t) &= \sum_{i=0}^{t-1} p(i) \\ \omega_1(t) &= \sum_{i=0}^{t-1} p(i) \end{aligned} \tag{3.11}$$

The weights $\omega_0(t)$ and $\omega_1(t)$ represent the probability that a given pixel in the image belongs to class 0 and 1, respectively, when threshold t is applied. As proven by Otsu[14], minimizing the intra-class variance in (3.10) also maximizes the between-class variance σ_b as shown in (3.12) where σ_T is the total variance. μ_0 and μ_1 represent the mean values of their respective classes, while μ_T represents the mean of all values. Otsu's method for threshold estimation is therefore to maximize $\sigma_b^2(t)$ by equation (3.12).

$$\begin{aligned} \sigma_b^2(t) &= \sigma_T^2(t) - \sigma_\omega^2(t) \\ &= \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \\ &= \omega_0\omega_1 \left[(\mu_0(t) - \mu_1(t))\right]^2 \end{aligned} \tag{3.12}$$

The performance of this method will now be discussed. First, consider whether the assumption that a given video frame will contain relatively bimodal motion holds true. This can be tested by plotting the optical flow magnitude of every pixel into a histogram and assessing the histograms shape. If the histogram shape is similar to the example in figure 3.3.2, the assumption can be said to hold.

The histogram in figure 3.3.3 clearly does not conform to the the assumption made earlier. As such, this method of using an Otsu threshold on the optical flow magnitudes is expected to provide non-optimal results. Even though it is not perfect, the method was tested anyway in order to determine whether it is still viable. The entire process can be summarized as doing the following for every frame:

- Generate the saliency map.
- Run Farnebäck optical flow.

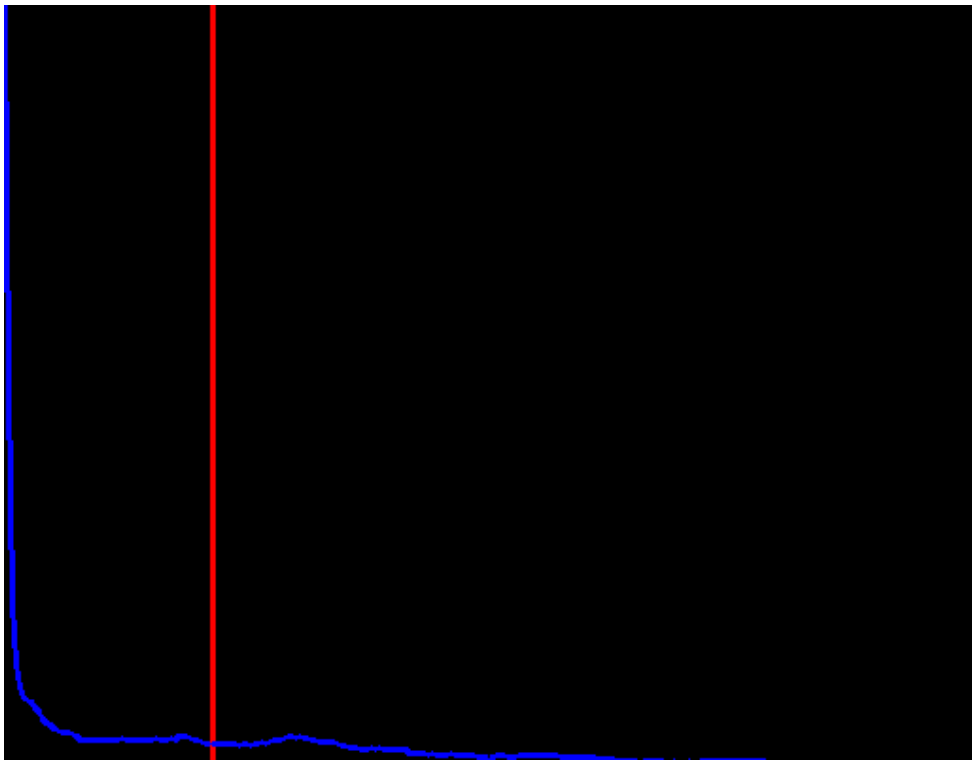


Figure 3.3.3: Histogram (blue) showing the optical flow magnitude distribution from the image in figure 3.3.4. The red line marks the Otsu threshold.

Figure 3.3.4: Image from which the optical flow illustrated in figure 3.3.3 is estimated.

- Apply Otsu's threshold estimator on the flow magnitude matrix.

- Create a binary image from the result.

Figures containing the original saliency map from each saliency algorithm, and the corresponding saliency maps processed by the suggested optical flow filter are presented to demonstrate performance. The results on a fine grained saliency map is illustrated in figure 3.3.5. Figure 3.3.6 demonstrates the correlation between the motion saliency map, and the mask generated by running Farnebäck on the original images. The reason for running optical flow on the original image instead of the motion saliency map was explained in section 3.3.3.

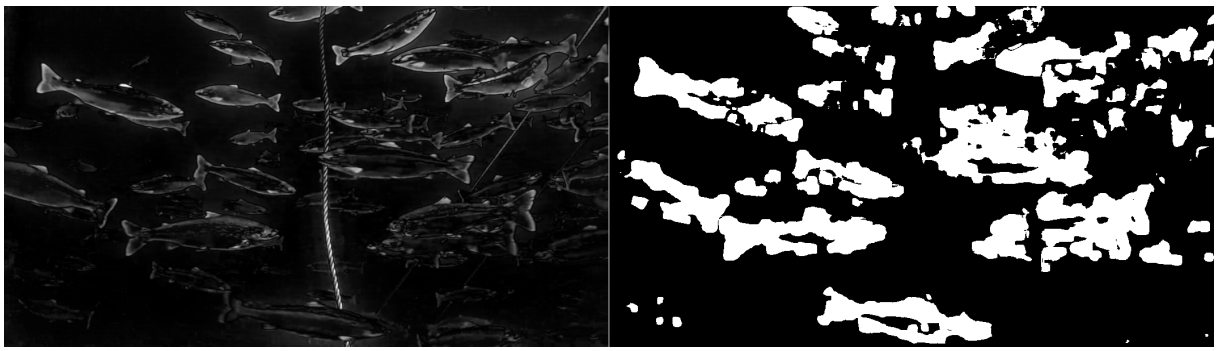


Figure 3.3.5: A fine grained saliency map (left), and a mask created by Otsu from the flow magnitudes (right).

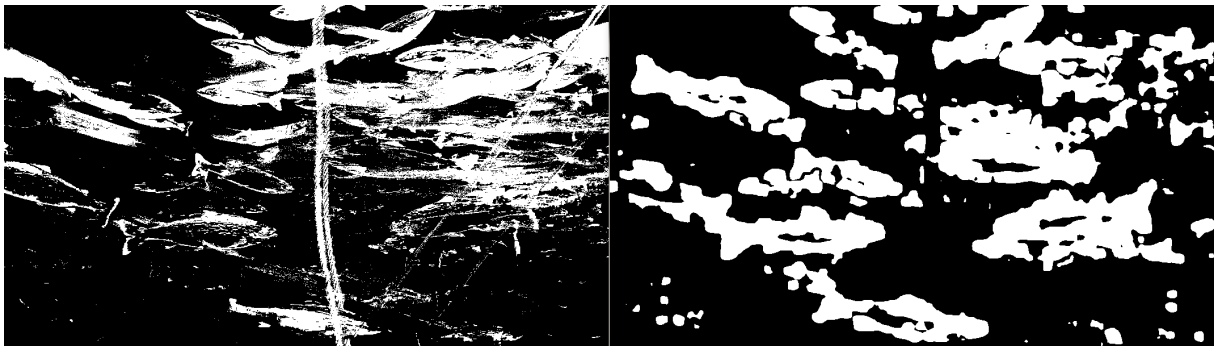


Figure 3.3.6: A motion saliency map (left), and a mask created by Otsu from the flow magnitudes (right).

3.3.5 Adaptive threshold

The same adaptive threshold scheme as described in section 3.1 was also attempted to filter the optical flow results. The overall work-flow is similar to that with Otsu's threshold; the main difference is that Otsu's threshold has been swapped with an adaptive threshold. Similar to what was done with Otsu's method, the adaptive threshold is applied on the optical flow magnitude matrices. The goal is also the same: keeping regions with interesting fish white, and everything else black.

An adaptive threshold scheme for this purpose must be run with a large value for k, meaning the subimages in which the thresholds are computed and applied are larger. If k is too small for a pixel in the middle of a fish-contour, the entire k-*by*-k area around that pixel could be inside the same contour. If this happens, the adaptive threshold would be such that only sub-regions of the fish body, with notably higher motion estimates would be included as white. The resulting filtered image could resemble a fine grained saliency map, as in figure 2.1.2c. Given an ideal motion estimator that attributes the same speed to all pixels on the same object, an adaptive threshold with a small k-value would only include edges of the fish contours as the optical flow magnitude over a the contour of a single fish would be uniform. In order to guarantee this did not happen, k was set to 501 pixels. An example result is demonstrated in figure 3.3.7.

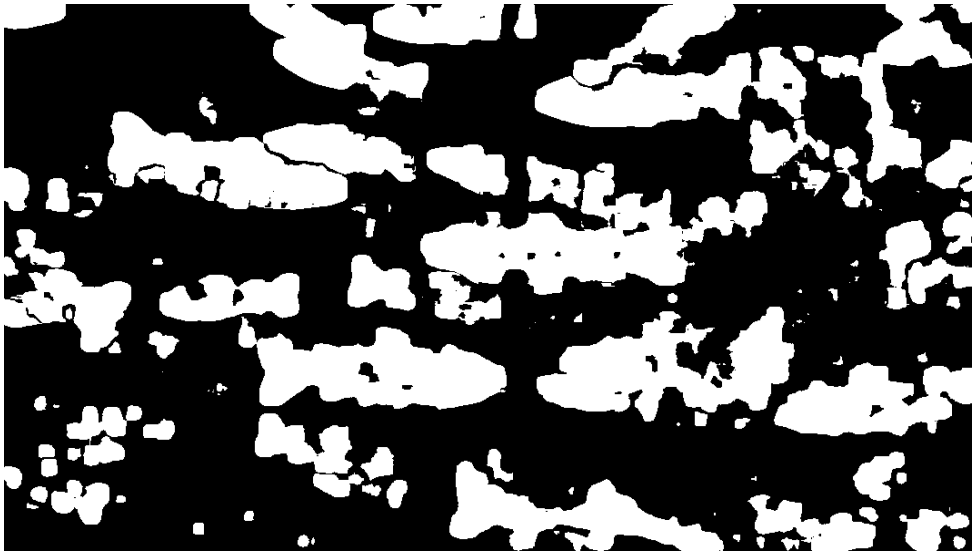


Figure 3.3.7: Mask created by adaptive threshold on flow magnitudes from figure 3.3.1. $k = 501$, $C = 10$.

Chapter 4

Discussion

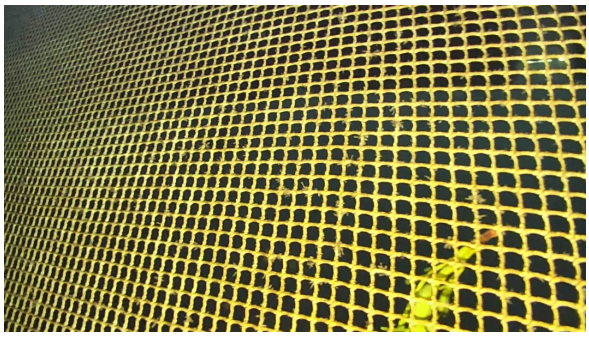
4.1 Frame log filter

The results gathered in section 3.2 clearly indicate that this approach is not suited for filtering saliency maps in a sea-cage. From what was learned through this project, there seems to be no way to meet the required assumptions for this method. This applies even in a calm setting with weak currents and low winds, as was the case when the material represented in figures 3.2.1-3.2.2 was recorded. Additionally, even if there was some way to determine the length n of the frame log such that it perfectly separates fish from other objects, this choice of n will likely be sub-optimal after a minor change in the environment. In a dynamic environment such as a sea-cage, this approach is simply not viable.

4.2 Optical flow with Otsu filter

This section will consider the applicability of the optical flow filter in three cases where the images contain 1) only fish, 2) only net/ropes, and 3) both fish and net/ropes. In each case a figure illustrating the results will be presented and discussed. Also, the corresponding raw image is presented in figure 4.2.1, and histograms showing relevant motion magnitude distributions can be found in appendix B.

(a) A typical image containing only fish.



(b) A typical image containing only net.

(c) An example image containing both fish and static objects.

Figure 4.2.1: Example frames for each case to be examined.

4.2.1 Fine grained saliency

First, consider case 1, when there are only fish in the image. The performance of the optical flow approach in this case is shown in figure 4.2.2. It is clear that some fish are excluded. This is because of the way Otsu's method calculates the threshold value: the threshold estimator is based on the assumption that there are two distinct classes in the image[14]. When there is not, such as in this case where there are only fish, the threshold must necessarily divide the single class into two. The result is that the fish with less motion are excluded from the mask, while the fish with faster motion are included. Note that the unit of perceived speed here is *pixels/frame*, and as there is no depth information available this can not reflect true speed. Instead, it is a measure for a combination of actual speed and distance from the camera. Proof of this can be found in appendix C. As a result, when the video contains only fish, this attempt at using optical flow for filtering is likely to provide data such that any orientation

algorithm based on it will focus on the fish that are closest to the camera.

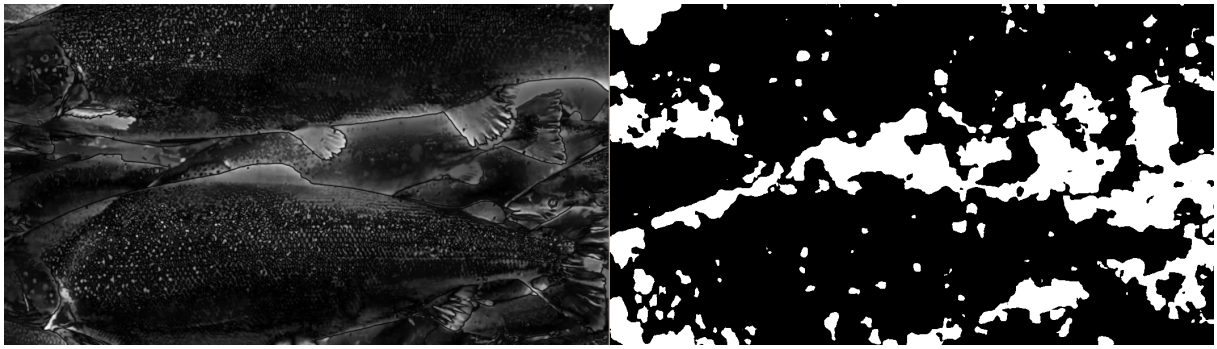


Figure 4.2.2: A fine grained saliency map from within a shoal of salmon (left), and a mask generated by the optical flow method (right).

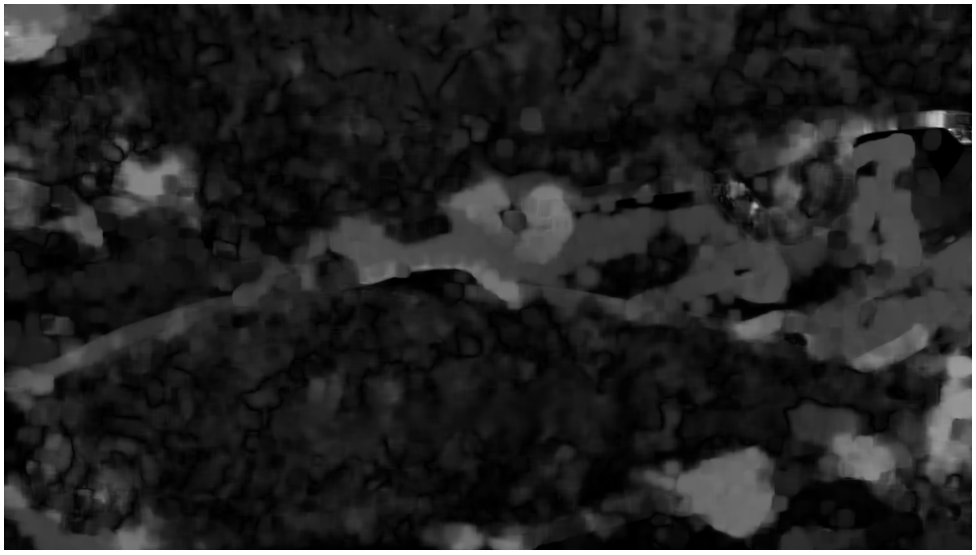


Figure 4.2.3: Flow magnitudes related to the case with only fish.

Note how this explanation is contradicted by figure 4.2.2. This is because, as indicated by the corresponding flow magnitude image in figure 4.2.3 and histogram in figure B.0.1, the optical flow magnitudes of the two fish in the foreground is lower than the rest of the frame. In reality, this is not true as the two salmon in question moved faster in the video than those further away. This phenomenon is consistent across different videos, and indicates a weakness in Farnebäcks optical flow algorithm when the camera is close to the fish. As mentioned earlier, the optical flow approach was expected to prioritize fish that are close to the lens, under the assumption that they are swimming at a certain speed. As proven here, this assumption is not

always true, indicating that the Farnebäck optical flow + Otsu approach is not a perfect one.

In the second case, the core problem is the same as for case 1, namely that there is only one class to consider: the net. However, compared to case 1 there is a big difference. In case 1 there are multiple fish swimming at slightly different speeds at varying distance from the camera lens, which means that to a camera they present as regions with different motion. In the case of a net however, where there is just a single object, the motion magnitude in each pixel will be much more similar (see figure B.0.2). Based on this similar motion and how Otsu binarization is implemented, it is expected that Otsu binarization will return a mask where roughly 50% is included. The results for this case are illustrated in figure 4.2.4. Notice how the results fully match the presented theory. The included region covers about 50% of the frame, and represents the part of the net closest to the camera.

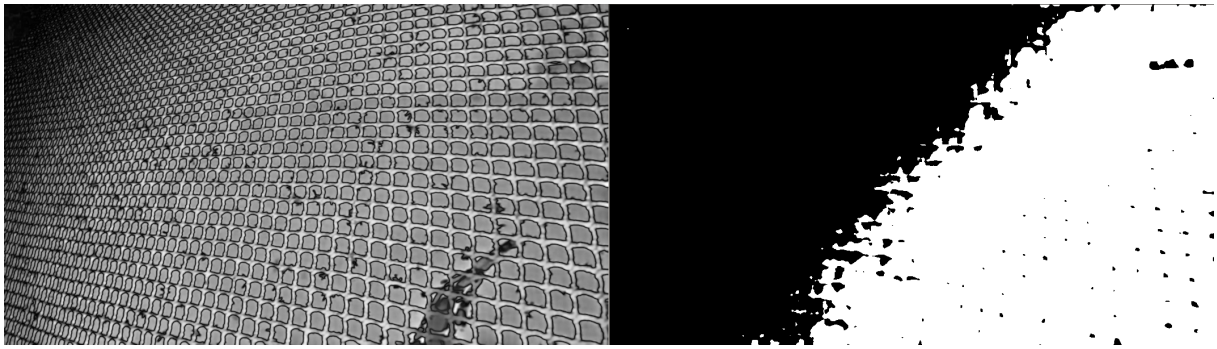


Figure 4.2.4: A fine grained saliency map of a net (left), and a mask generated by the optical flow method (right).

The third case is when there are both fish and other objects in the image. Figure 4.2.5 shows how most of the fish are included in the optical flow mask, while the camera system is ignored. The improved performance relative to the first case in figure 4.2.2, comes from the fact that in this case there are large regions with very little motion. This can be seen from figure B.0.3. Otsu's threshold estimator therefore sets the threshold lower than if the entire frame was full of moving fish. The result is that there

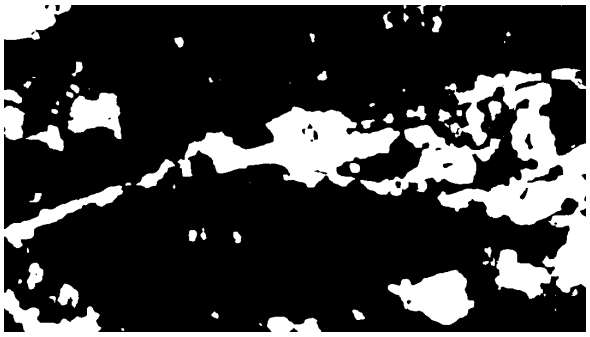
are no major regions where fish are excluded from the mask. The camera system, on the other hand, is correctly excluded. Note the improvement from the frame log approach from section 3.2.

Figure 4.2.5: Saliency map containing fish and a camera system (left), and its corresponding optical flow-based mask (right).

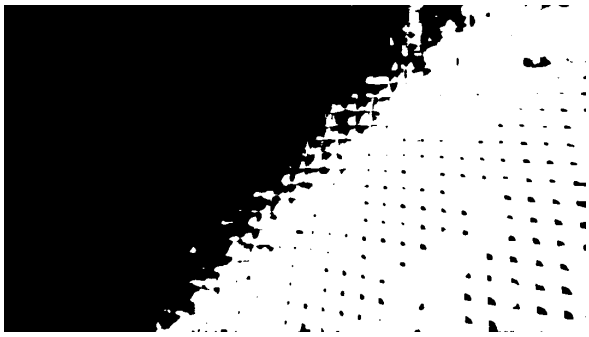
4.2.2 Unprocessed video

As explained in section 3.3.3, optical flow on motion saliency maps yields little information. Instead, an optical flow-based filter must be run on the original images. The results from the same three cases used in the previous section are shown in figure 4.2.6.

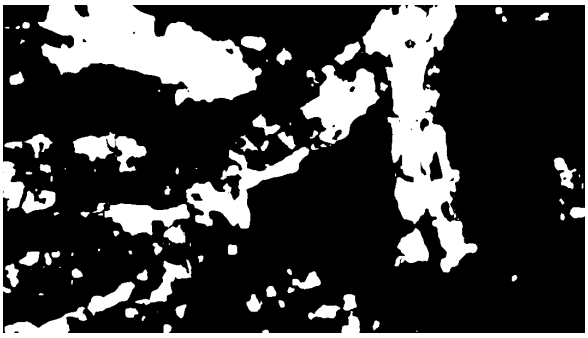
For all three cases, the results are very similar to the results for fine grained saliency. This has three implications: First, similar results mean that the discussion around the fine grained saliency results (section 4.2.1) applies to this case as well. Second, it backs up the initial claim that the fine grained saliency map retains much of the information from the original image. If not, it would be expected that the optical flow returned different results. Finally, it implies that when using optical flow for this type of filtering there is no need to run it off the saliency map, as similar results are achieved from the unprocessed video. This opens up the possibility of generating the motion estimates and saliency maps in parallel to increase efficiency.



(a) Case 1.



(b) Case 2.



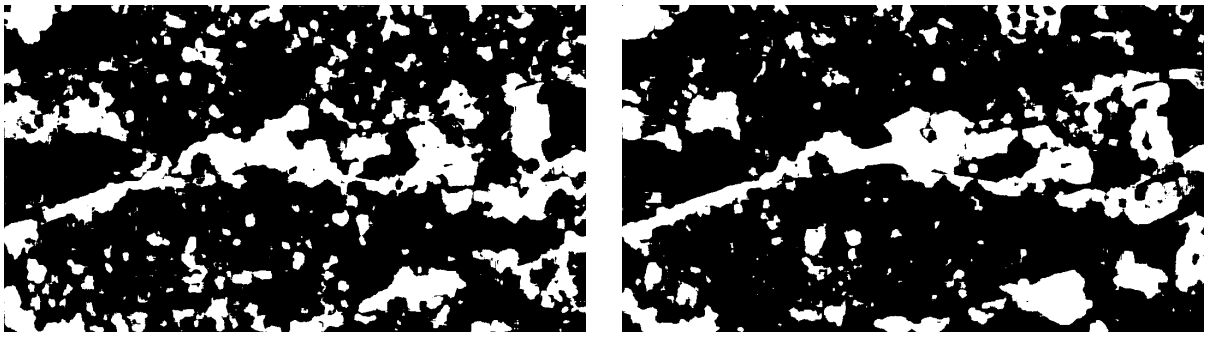
(c) Case 3.

Figure 4.2.6: Optical flow from original images with applied Otsu threshold.

4.3 Optical flow with adaptive filter

The discussion in this section will be based on the same three images as the Otsu filter (figure 4.2.1). For the first case, with only fish in the image, the results are demonstrated in figure 4.3.1. There are notable similarities between these and the results with an Otsu threshold. The main difference is that with an adaptive threshold, spots on the body of two salmon in the foreground are included. It is however clear that neither an adaptive threshold filters this data well. The reason is the same as for Otsu binarization: motion estimates for the two foreground salmon are incorrect.

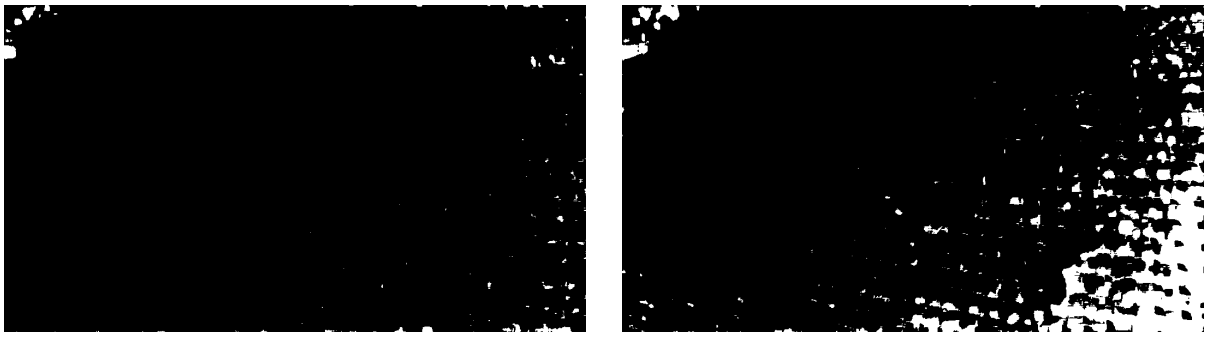
For the case in figure 4.2.1b where a net covers the entire frame, the results can be found in figure 4.3.2. The filtering performance with an adaptive threshold is far superior to that of an Otsu threshold from figure 4.2.4, where most of the image is black. The frames illustrated here also indicate



(a) From fine grained saliency maps. (b) From unprocessed video.

Figure 4.3.1: Results of optical flow with adaptive threshold. Frame contains only fish.

that the adaptive threshold performs best when based on the fine grained saliency maps. While that is true in this specific case, the comparative results of this approach based on saliency maps or unprocessed video varies with every frame. In the case where the video frame contains only nets, the optical flow approach with an adaptive threshold successfully excludes most the net from evaluation when based on either data type.

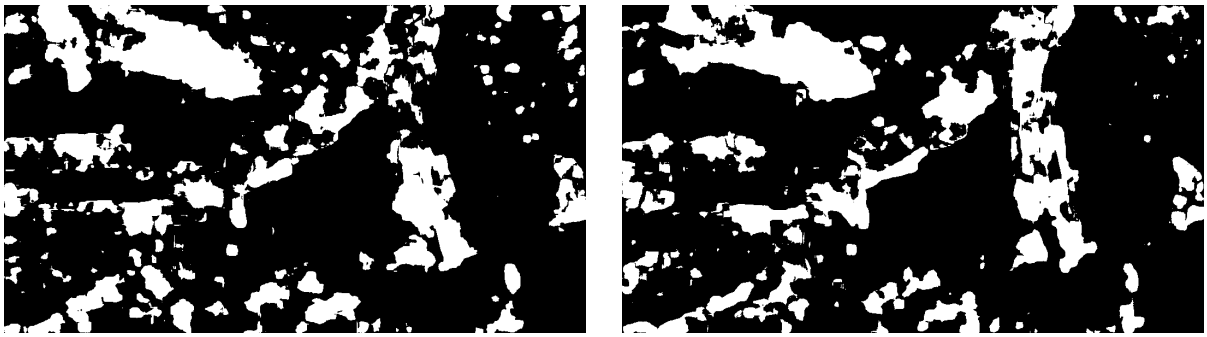


(a) From fine grained saliency maps. (b) From unprocessed video.

Figure 4.3.2: Results of optical flow with adaptive threshold. Frame contains only net.

When there are both fish and stationary objects in the frame the results are typically as illustrated in figure 4.3.3. Again, the results are similar to those when using an Otsu threshold as in figure 4.2.5. However, upon further inspection and comparison to the original frame in figure 4.2.1c, Otsu's method appears to provide slightly more accurate results. This claim is

based mainly on the differences in the top-left and top-right corners of the result images.



(a) From fine grained saliency maps. (b) From unprocessed video.

Figure 4.3.3: Results of optical flow with adaptive threshold. Frame contains fish, net, and a camera system.

The observation made in section 4.2.2 holds true also when using an adaptive threshold to filter the results: because the results based on fine grained saliency maps and unprocessed video frames are so similar, there is no indication that optical flow should be run on the saliency maps. If implemented, this type of system may run optical flow and saliency algorithms in parallel to decrease computation time.

4.4 Optical flow filter comparison

Comparing the over-all performance of the optical flow-based filter with Otsu and adaptive thresholds, the adaptive threshold is superior. It is expected that using this approach with an Otsu threshold near a cage net will not properly address the challenge discussed in section 2.4, where the net is likely to be considered the most salient object. Ultimately, an orientation algorithm based on this is expected to orient towards any cage net in the vicinity. Using the adaptive threshold however, the net was almost fully removed. The fact that the frame in figure 4.3.2 represents only a re-orientation from figure 4.3.3[1], implies that as long as the total saliency

[1]I was there when the footage in question was recorded.

in figure 4.3.3 is higher than 4.3.2 the camera would be oriented away from the net. This re-orientation would occur when using the adaptive threshold, but not with Otsu. Ultimately, using an adaptive threshold to create a filter from the magnitude of optical flow successfully removes the cage net from consideration for an orientation algorithm.

In the case where the frame contains only fish, the optical flow filter fails to perform adequately. The flow magnitude illustration in figure 4.2.3 shows how the fish close to the camera are estimated to move slower than those behind them when using Farnebäcks algorithm to estimate motion. This happens even though in reality the foreground fish moved faster through the frame. Because of this behaviour in the optical flow algorithm, the best option is to base a camera orientation system on the unprocessed saliency maps as there is no need to perform filtering when there are only fish in the frame. Optimal results would be achieved if an algorithm can determine when the camera is too close to unwanted objects, and only then enable the optical flow filter.

Chapter 5

Conclusion

In chapter 2, two algorithms for estimating visual saliency were presented: The fine grained saliency by Montabone & Soto[17], and motion saliency by Wang & Dudek[3]. These algorithms were compared with respect to their ability to facilitate an orientation algorithm in a fish-cage.

Further, in chapter 3, an attempt was made to rectify the challenge related to the net being evaluated as a salient region by both saliency algorithms. No method was found that will address this problem in a general case. The only method found that will successfully separate fish from non-interesting objects was using an optical flow-based filter mask, which does not produce viable results whenever there are only fish in the image. As this is the most common case, this performance is unacceptable. In conclusion, a camera orientation scheme for use in aquaculture cages based on visual saliency will have to be used in such a way that this problem is avoided. Either, the system must recognize and control when the filter should be enabled, or it must be designed such that an orientation algorithm is never run while the camera is positioned near the cage net.

Based on the results from section 2.3 and the fact that the optical flow filter performs equally well when used with either saliency algorithm, Wang & Dudeks motion saliency is deemed as best suited to be used as a basis for camera orientation.

The results presented in this thesis indicate that performing general camera orientation based on visual saliency in a sea-cage is a difficult problem. It is expected that for a saliency-based orientation algorithm to function it will have to either be operated only when the camera is sufficiently far from the cage net, or be used in tandem with a cage-net detector. On the other hand, if a cage-net detector is required one could implement a fish detector instead and use that as the basis for orientation. In conclusion, visual saliency can provide a basis for camera orientation. However, it is likely that other approaches would perform better.

Bibliography

[1] Berge A. *IBM bruker kunstig intelligens til å løse luseproblemet.* iLaks.no. URL: https://ilaks.no/ibm-bruker-kunstig-intelligens-til-a-lose-luseproblemet.

[2] Petersen B. *Dynamic Range Explained.* B&H Photo. URL: https://www.bhphotovideo.com/explora/photography/tips-and-solutions/dynamic-range-explained.

[3] Wang B. and Dudek P. "A Fast Self-tuning Background Subtraction Algorithm". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014).

[4] Hayward C. J., Andrews M., and Nowak B. F. "Introduction: Lepeophtheirus salmonis—A Remarkable Success Story". In: *Salmon Lice: An Integrated Approach to Understanding Parasite Abundance and Distribution.* Ed. by S. Jones and R. Beamish. John Wiley & Sons, 2011.

[5] The Seafood Innovation Cluster. *AquaCloud- The use of artificial intelligence in sea lice management.* NCE Seafood. URL: http://www.seafoodinnovation.no/article/213/AquaCloudThe_use_of_artificial_intelligence_in_sea_lice_management.

[6] OpenCV Documentation. *Image filtering - getGaussianKernel.* URL: https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html#getgaussiankernel.

[7] Farnebäck G. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Image Analysis*. Ed. by Bigun J. and Gustavsson T. Vol. 2749. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, pp. 363–370.

[8] Bakken J. B. *Tar opp kampen mot lakselus med kunstig intelligens*. Dagens Nærignsliv. URL: https://www.dn.no/nyheter/2017/06/21/1446/Havbruk/tar-opp-kampen-mot-lakselus-med-kunstig-intelligens.

[9] Cheng M., Mitra N. J., Huang X., Torr P. H. S., and Hu S. "Global Contrast Based Salient Region Detection". In: *IEEE Transactions on pattern analysis and machine intelligence* 37 (2015).

[10] Harr M. C. "Intelligent camera orientation in aquaculture sea-cages". Specialization project. Norwegian University of Science and Technology, 2017.

[11] Jenssen M. M. *AquaCloud - kunstig intelligens skal hjelpe mot lakselus*. URL: http://www.fiskerioghavbruk.no/teknologisk-utvikling/aquacloud-kunstig-intelligens-skal-hjelpe-mot-lakselus.

[12] Marine Institute (Foras na Mara), Ireland. *Life cycle of the Salmon Louse*. URL: https://www.marine.ie/Home/site-area/areas-activity/aquaculture/sea-lice/life-cycle-salmon-louse.

[13] Nikon. *Understanding Focal Length*. URL: https://www.nikonusa.com/en/learn-and-explore/a/tips-and-techniques/understanding-focal-length.html.

[14] N. Otsu. "A Threshold Selection Method from Gray-Level Histograms". In: *IEEE Transactions on Systems, Man, and Cybernetics* 9 (1 1979), pp. 62–66.

[15] Jensen P. M. *Mer enn fem milliarder kroner i lusekost i fjor*. kyst.no. URL: https://kyst.no/nyheter/mer-enn-fem-milliarder-kroner-i-lusekost-i-fjor.

[16] Frintrop S. *VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search*. 2006.

[17] Montabone S. and Soto A. "Human detection using a mobile platform and novel features derived from a visual saliency mechanism". In: *Image and Vision Computing* 28 (2010), pp. 391–402.

[18] P.K Sahoo, S Soltani, and A.K.C Wong. "A survey of thresholding techniques". In: *Computer Vision, Graphics, and Image Processing* 41.2 (1988), pp. 233 –260.

[19] Cui X., Liu Q., and Metaxas D. "Temporal Spectral Residual: Fast Motion Saliency Detection". In: *17th ACM international conference on Multimedia* (2009), pp. 617–620.

[20] Schechner Y. Y. and Karpel N. "Clear Underwater Vision". In: *Computer Vision and Patter Recognition* (2004).

Appendices

Appendix A

Unprocessed images

Figure A.0.1: Frame corresponding to the saliency maps in figures 2.1.2b and 2.2.1b. Time of day: 09:30

Figure A.0.2: Frame corresponding to the saliency maps in figures 2.1.2a and 2.2.1a. Time of day: 09:30

Figure A.0.3: Frame corresponding to the saliency maps in figures 2.1.2c and 2.2.1c. Time of day: 15:30

Appendix B

Farnebäck motion magnitudes

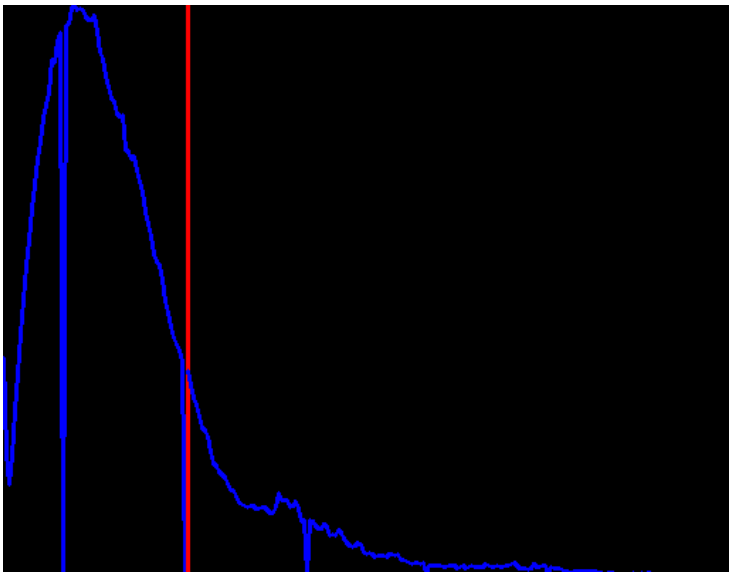


Figure B.0.1: Histogram of motion magnitudes related to figure 4.2.1a. Generated by Farnebäck optical flow.

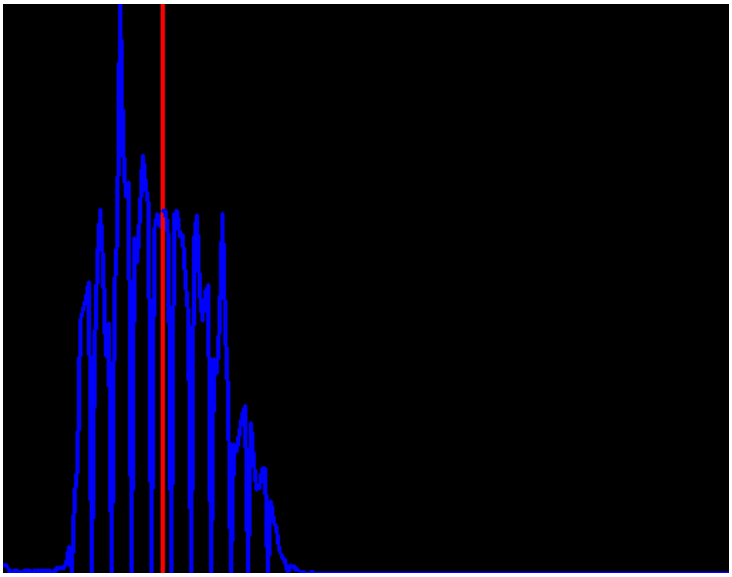


Figure B.0.2: Histogram of motion magnitudes related to figure 4.2.1b. Generated by Farnebäck optical flow.

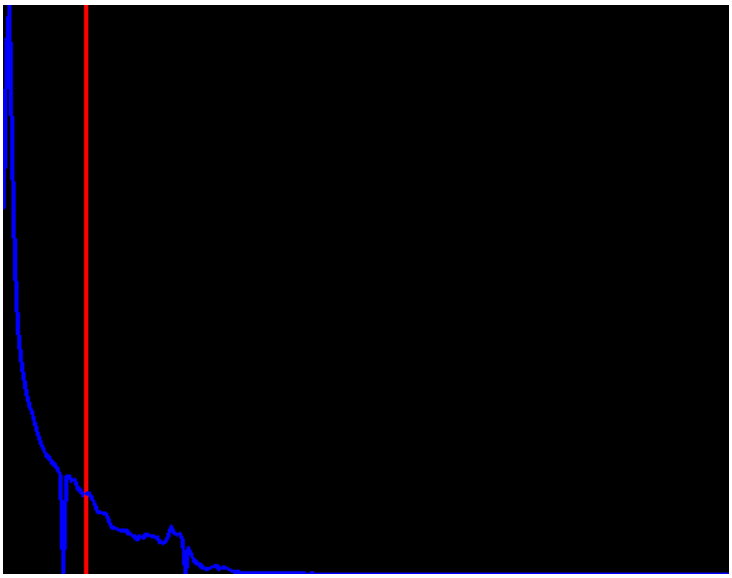


Figure B.0.3: Histogram of motion magnitudes related to figure 4.2.1c. Generated by Farnebäck optical flow.

Appendix C

Actual vs. perceived speed

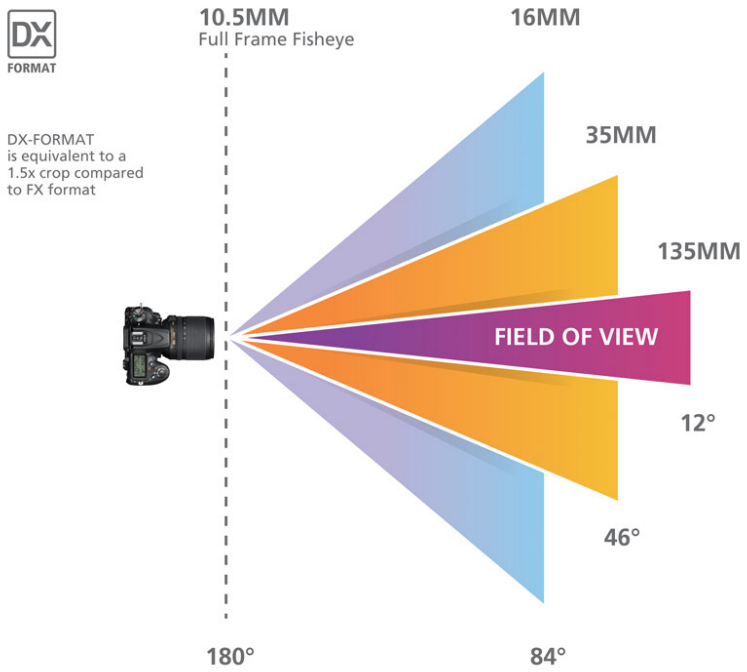


Figure C.0.1: Field of view of DX format NIKKOR lenses [13]. *Photo by: Lindsay Silverman*

This appendix aims to prove that a camera will perceive two objects moving at the same speed and different distances from the camera, to be moving a different speeds. Generally, a cameras field of view is the shape of a cone as illustrated in figure C.0.1. The angle of this cone varies between different cameras and lenses[13]. For the purposes of this discussion, assume an image resolution of 1920x1080 pixels.

A cone with angle θ and length x, has a width y at its end given by

$$y = 2x \tan(\theta). \tag{C.1}$$

Notice how the width of the field of view is proportional to the distance x. At any given distance, the entire width y must be represented by 1920 pixels. Because $y \propto x$, this means the real-life area covered by a single pixel is also proportional to x. For the conversion of speed from unit *m/s* to *pixels/frame*, this dictates the same proportionality: For any two objects moving with the same speed in *m/s*, the one closer to the camera lens will appear to move faster in a video. The difference in perceived speed depends on the ratio $\frac{x_1}{x_2}$. In conclusion, the relative perceived speed $v_{p,r}$ of two objects in a video is

$$v_{p,r} = \frac{v_1}{v_2}\frac{x_1}{x_2}, \tag{C.2}$$

where v_1, v_2 are the objects real-world speeds and x_1, x_2 are the distances from the camera to the objects.