



NTNU – Trondheim
Norwegian University of
Science and Technology

Automatic Quality Control of Salmon

Using Machine Learning Algorithms based on
Input from a 3D Machine Vision System

Øystein Sture

Master of Science in Cybernetics and Robotics

Submission date: June 2015

Supervisor: Amund Skavhaug, ITK

Co-supervisor: John Reidar Mathiassen, SINTEF Fiskeri og Havbruk

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Summary

Quality control of Atlantic salmon is currently a task performed manually by human operators. To stay competitive in an increasingly global market, it becomes necessary to take advantage of technology to improve productivity and profitability. This is especially the case in countries with high salary levels. In this thesis, a complete machine vision system for 3D-imaging has been built and integrated for the purpose of quality control of Atlantic salmon. The system is build using off-the-shelf hardware, and has been integrated using no external proprietary tools. The software for the acquisition was implemented with real-time restrictions in mind. The end result is thus an affordable solution, which can be deployed in an industrial environment without major investments.

An experiment was then performed on Atlantic salmon of different quality classes. The obtained data was used to develop descriptors that capture enough information to separate out lower classes of Atlantic salmon based on its appearance. The thesis focuses on two primary causes of downgraded salmon; deformities and wounds. Deformities appear due to skeletal deformations and inflammation. Wounds appear due to cuts and scrapes that are infected by bacteria. Using geometric features and color information, two classifiers was developed to handle each of these cases. The classifiers have been found to reliably detect deformities and wounds in Atlantic salmon, and shows that 3D-imaging has great potential within the field of automatic quality control of fish.

Sammendrag

Kvalitetskontroll av atlantehavslaks er i dag en oppgave som blir utført manuelt. For å forsikre seg at næringen holder seg konkurransedyktig i et økende internasjonalt marked, er det nødvendig å anvende teknologi for å øke produktivitet og lønnsomhet. Dette er spesielt tilfellet i land med høye lønnsnivå. I denne masteroppgaven har et komplett maskinsynsystem for 3D-billedtakning blitt bygget og integrert med den hensikt å brukes til automatisk kvalitetskontroll av atlantehavslaks. Systemet er bygget ved bruk av lett tilgjengelige og rimelige komponenter, og integrert uten bruk av fordyrende eksterne løsninger. Løsningen er laget med fokus på effektivitet, slik at det kan tas i bruk i sanntidsapplikasjoner. Resultatet er en rimelig løsning, som kan benyttes industrielt uten større investeringer i utstyr.

Et eksperiment ble utført ved bruk av maskinsynsystemet på atlantehavslaks av forskjellige kvalitetgraderinger. Det innhentende datasettet ble da brukt til å utvikle beskrivende egenskaper med nok informasjon til å skille mellom graderingene basert på utseende. Denne masteroppgaven fokuserer primært på to årsaker til nedgradering; deformiteter og sårdannelser. Deformiteter oppstår ved deformering i skjelettet og betennelser, mens sår oppstår som følge av bakteriell infeksjon. To klassifikatorer ble utviklet til å skille mellom graderingene ved å bruke geometriske egenskaper og farger. Resultatene presentert i denne oppgaven indikerer at de to klassifikatorene er i stand til å skille ut laks av lavere kvalitet med tilfredsstillende nøyaktighet. Den anvendte metodikken for øvrig viser at 3D-maskinsyn har stort potensiale innen automatisk kvalitetskontroll av fisk.

Preface

This thesis was written during the final semester of the two-year Master's programme at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU). The thesis was written in collaboration with SINTEF Fisheries and Aquaculture, and was connected to a research project for quality grading and sorting of salmon. I was first introduced to the project during a summer internship, where some of the groundwork for this thesis was laid.

Originally, the focus of this thesis was more oriented towards developing new novel classifiers for use in quality control, but more time was needed to complete the 3D scanning and feature extraction aspects robustly. Therefore, this thesis focuses more on the latter than was originally envisioned. Nonetheless, the techniques employed are new within the industry and serves as an important step towards realizing an industrial system for grading and sorting salmon.

I would first like to thank my supervisor, Amund Skavhaug, for keeping me on track with the written material and attempting to keep my stress levels in check. I would also like to thank my co-supervisor at SINTEF, John Reidar Mathiassen for sharing his vast experience in the topic of computer vision and generally giving sound advice for any technical hurdle. Additionally, I would like to thank my classmates and the rock climbing team, for some great memories. Thanks to my parents, Edith and Helge, for being supportive. Finally, I need to thank my girlfriend, Karina, for putting up with the long hours and a general absence of mind.

Table of Contents

Summary	1
Sammendrag	2
Preface	3
Table of Contents	7
List of Tables	9
List of Figures	14
Abbreviations	15
1 Introduction	1
1.1 Background	1
1.1.1 The Norwegian Seafood Industry	2
1.1.2 Common Defects in Farmed Salmon	4
1.2 Benefits of Further Automation	6
1.3 Previous Work	8
1.4 Price of Comparable Systems	9
1.5 Related Work	10
1.6 Goals and Objectives	10
1.7 Structure of the Thesis	11
2 Literature Review	13
2.1 Classical Statistical Learning	13
2.1.1 Curse of Dimensionality	14
2.1.2 Overfitting	15
2.1.3 Cross-Validation	15
2.2 Support Vector Machines (SVM)	17
2.2.1 Linear SVM	17
2.2.2 Nonlinear SVM	19
2.2.3 VC Theory	21
2.2.4 SVM: Advantages and Disadvantages	22

2.3	Numerical Geometry and Shape Recognition	23
2.3.1	Medial Axis Transform	25
2.4	Geometric Transformations	28
3	Equipment and Acquisition	31
3.1	Line Scanner Principles	31
3.2	The Camera Rig	32
3.3	Camera Triggering	34
3.4	Calibration Routine	36
3.4.1	Direct Linear Transformation (DLT)	39
3.4.2	Non-linear Optimization	40
3.4.3	Misalignment Correction	41
3.4.4	Speed Calibration	41
3.5	Parallel Image Processing	42
3.5.1	Bayer Filtering	43
3.5.2	Coordinate Extraction	44
3.5.3	Reflectance Properties	45
3.5.4	GPU Code Optimization	46
3.5.5	Color Images	47
3.6	Polarization Filter	48
4	Experiments and Feature Extraction	53
4.1	Quality Grading	54
4.1.1	Humpback	54
4.1.2	Wounds	55
4.2	Point Cloud Post-Processing	56
4.2.1	Statistical Outlier Removal	56
4.2.2	Fin Removal	59
4.2.3	Spline Re-sampling	63
4.3	Medial Axis	68
4.4	Geometric Feature Extraction	69
4.4.1	Width	71
4.4.2	Height	72
4.4.3	Length	73
4.4.4	Skewness	74
4.5	Color Image Projection	75
4.6	Color Feature Extraction	76
5	Results	81
5.1	Method of Performance Evaluation	81
5.2	Detecting Deformities	82
5.2.1	Deformity Detection Using C-SVM with RBF Kernel	82
5.2.2	Deformity Detection Using Nearest Neighbor Classifier (NN)	87
5.3	Wound Detection	87
5.3.1	Detection of Wounds Using SVM with RBF Kernel	88
5.4	Summary of Results	93

6	Discussion	95
6.1	Methodology	95
6.1.1	Line Scanning	95
6.1.2	Camera Calibration	96
6.1.3	Laser Extraction	96
6.2	Features	97
6.2.1	Symmetry	97
6.2.2	Medial Axis	97
6.2.3	Reflective Properties	98
6.3	Attained Prediction Rates	98
6.3.1	Deformity Detection	99
6.3.2	Wound Detection	99
6.3.3	Evaluation of 3D Machine Vision in Quality Control	99
6.4	Additional Remarks	100
6.4.1	Performance	100
6.4.2	Robustness	100
7	Conclusion	103
7.1	Summary of Contributions	104
7.2	Recommendations and Further Work	105
	Bibliography	107
	Appendix	111
A	Dataset	111
B	Camera Triggering (Arduino)	115
C	Pruning and Spline Re-sampling (Matlab)	119

List of Tables

2.1	Homographic transformations.	30
5.1	Deformity Classification Results	93
5.2	Wound Classification Results	93
A1	Superior dataset 1/2	111
A2	Superior dataset 2/2	112
A3	Ordinary dataset	113
A4	Production dataset	114

List of Figures

1.1	Global statistics of commercial harvesting of aquaculture and aquatic wildlife (Source: FAO).	2
1.2	Norwegian exports of salmon and trout (Source: SSB).	3
1.3	Vertebral column of salmon without signs of deformations (from Witten et al. (2009)).	5
1.4	Multiple extreme deformations of the vertebral column of an adult salmon (from Fjellidal et al. (2012)).	5
1.5	Comparison of a regular salmon with one that has a compressed vertebral column, also known as short-tail (from Witten et al. (2005)).	6
2.1	A separation between two classes described in a two-dimensional feature space.	14
2.2	Overfitting in the case of regression, the trend in the points might be better described through a linear function than a curved function which correctly minimizes the distance to each point.	15
2.3	The optimal hyperplane is the hyperplane that separates the classes with the maximal margin.	17
2.4	The mapping Φ embeds the data into a higher-dimensional space where the nonlinear pattern appears linearly (adapted from Shawe-Taylor and Cristianini (2004)).	19
2.5	VC dimension example for linear separating functions in the plane (adapted from Vapnik (1995)).	21
2.6	The four main shape representations (Adapted from Siddiqi and Pizer (2008)).	24
2.7	Landmark points utilized by Misimi et al. (2008) in quality control of Atlantic Salmon.	25
2.8	Example Voronoi diagram for eight discrete points in R^2 .	27
2.9	The endpoints of lines at infinity forms a line in a 2D projective space, similar to an horizon.	28
3.1	The basic concept of a line-scanner using a sheet of light laser and camera.	32
3.2	Overall steps of the procedure.	32
3.3	Conceptual sketch of the camera setup. Left illustration is as seen from the front, and the right is seen from the side.	33

3.4	Photo of the camera rig whilst scanning two mackerel for a related experiment.	34
3.5	Conceptual sketch of the triggering sequence. Not to scale.	35
3.6	The calibration object designed to calibrate all three cameras.	38
3.7	The stand used to calibrate the bottom cameras. The stand includes holes for all cameras, to improve on misalignment issues that would occur by simply placing the calibration object manually.	38
3.8	An laser image taken of the calibration object as seen when calibrating the top camera. The image has been inverted to avoid the black background.	38
3.9	The misalignment correction is performed by fitting each camera to a circle and translating to match. Note that the misalignment of the right camera (in green) is worse than typical. The axes are specified in centimeters.	42
3.10	Illustration of Bayer pattern (RG) in the sensor array of a camera (adapted from Wikipedia (2006)).	43
3.11	The bilinear interpolation pattern used. Two-way interpolation is used on the red grid, and four-way (bilinear) interpolation is used in the cross-section.	44
3.12	A scan obtained of a salmon using a polarization filter. The lighter areas on the back indicate missing coordinates (the points from the bottom are seen through the gaps).	50
3.13	An image of Fish16 in the data set, obtained without a polarization filter. Increased noise is present around fins and sharp edges due to reflections of laser light. Left: Side view, Middle: back view, Right: tilted abdominal view	51
4.1	QR code pointing to the scan data located at http://ntnu-msc-oystestu.s3-website-eu-west-1.amazonaws.com/	54
4.2	Comparison of three salmon from the dataset. A: Salmon 5 with no deformities, B: salmon 32 with forward humpback, C: salmon 94 with a less protruding hump (further back)	55
4.3	All the wounds present in the dataset.	56
4.4	Histogram of 25 nearest neighbor means for fish16 in the data set	57
4.5	Fish16 after applying the statistical outlier filter.	58
4.6	The given anatomical names for the fins of salmon (adapted from an illustration by Karen Uldall-Ekamm)	59
4.7	Illustration of the fin removal steps applied to the anal fin of fish19 from the data set. In A, the fin is detected by binning the coordinates along the X-axis and calculating the spread in each bin. If enough bins are below the threshold, an ellipse is fitted to one side of the slice as in B. In C a linear regression is performed with $b = 0$ and origin at the center of the ellipse. In D, the intercept point of the line and ellipse is calculated and the tangent in that point is used to remove the fin.	61
4.8	The fin removal routine applied to a slice where a fin is detected on both sides.	63

4.9	A case where significant noise is introduced into the 3D-model due to reflections from the laser-line. The reflections hits the pelvic fin moving into the frame. The middle and bottom images is the laser-line as seen by the camera and a thresholded image, respectively. The laser images have been inverted for viewability, which is why the laser-line appears to be cyan instead of red.	65
4.10	Instances where pruning of the exterior features is necessary to obtain a good spline approximation of the main curvature. The coordinates are shown as well as the spline obtained after pruning.	66
4.11	Fig. A illustrates an instance where the spline will cut off the edge. Fig. B illustrates an instance where pruning is not performed, as the entire protruding feature is visible from the centroid.	68
4.12	Three instances of medial axis calculation. The horizontal medial axis is displayed in green, while the discarded Voronoi vertices are shown in blue.	69
4.13	The directions of width and height of a salmon, as referred to in this section.	71
4.14	The vertex containing the radius of the largest embedded circle is used in the calculation of the medial length, and as a separate feature.	72
4.15	The typical difference between the direct length from the left of the spline to right of the spline and the length by tracing the medial axis.	73
4.16	The length of the salmon is represented through the medial length, which traces through the center of the largest embedded circle for each slice.	73
4.17	The visible coordinates from each side is projected onto a 2D-plane. Left figure displays the splitting method applied to a salmon from the dataset.	76
4.18	Salmon 47 (A) and 104 (B) from the dataset, projected onto a 2D plane and interpolated to ensure equidistant pixels. A is projected from the bottom cameras, while B is projected from the top camera. Wounds are indicated by a red circle.	77
4.19	Typical histogram for RGB wound pixels. The Y-axis indicates the number of pixels with that value.	78
4.20	Typical histogram for HSL wound pixels. The Y-axis indicates the number of pixels with that value.	78
4.21	Thresholding of the hue and luminosity applied to the image containing all the wounds.	79
4.22	Morphological open operation applied to a salmon that has already been thresholded to a binary image.	80
4.23	The convex hull of each connected region within the image.	80
5.1	Overall prediction rates from an exponential grid-search for the parameters γ and C using 10x10 stratified cross-validation.	83
5.2	Fish10 from the superior dataset, where the gutting makes it appear as deformed.	84
5.3	The parameter grid-search after removing three salmon from the superior class.	84

5.4	Performing a wide-area grid search, to ensure that neighbouring regions does not provide better results. The ranges are $\log_2 C = [-16, 60]$ and $\log_2 \gamma = [-60, 16]$ for the left figure, and $\log_2 C = [60, 100]$ and $\log_2 \gamma = [-60, 16]$ for the right figure.	85
5.5	A grid-search for the same range as figure 5.1, but with a smaller step size.	86
5.6	Parameter grid search using 50x8 cross-validation for mean RGB features.	88
5.7	Parameter grid search using 50x8 cross-validation for mean HSL features.	89
5.8	Parameter grid search using 50x8 cross-validation for mean and standard deviation RGB features.	89
5.9	Parameter grid search using 50x8 cross-validation for mean and standard deviation HSL features.	90
5.10	Parameter grid search using 50x8 cross-validation for mean, standard deviation and relative size (RGB).	90
5.11	Parameter grid search using 50x8 cross-validation for mean, standard deviation and relative size (HSL).	91
5.12	Salmon 48, 88 and 103 from top to bottom. These are the salmon with wounds that are most commonly misclassified.	92
6.1	Scatter intensity for an offset of 8 for salmon 102 (top).	98
6.2	Scatter intensity for an offset of 8 for salmon 90 (bottom).	98

Abbreviations

CV	=	Cross validation
CPU	=	Central processing unit
DLT	=	Direct linear transformation
FAO	=	Food and Agriculture Organization of the United Nations
FTE	=	Full time equivalent (employees)
GPU	=	Graphical processing unit
MAT	=	Medial axis transform
MLS	=	Moving least squares
SSB	=	Statistics Norway (ssb.no)
SVD	=	Singular value decomposition
SVM	=	Support vector machine

Chapter 1

Introduction

The aquaculture industry will become increasingly important in the years to come. The industries that process the raw materials into seafood, are however faced with difficulties in countries of high salary levels. There is often more economic incentive to either sell the raw product, or process it abroad by freezing it (Egegenes (2013)). To keep the refinement of the seafood profitable locally in high-cost countries, automated systems must be used to a larger degree to reduce the man-hours. A task that is performed manually today, is the quality control of Atlantic Salmon. In this thesis a rig with multiple cameras is used to produce accurate 3D-models of fish on a conveyor belt in real-time. Properties like color, reflectance and the spatial information from the 3D-model can be utilized to devise an automatic quality control system.

This chapter starts by providing a general background on the aquatic food industry and proceeds to motivate the need to partly or fully automate these processes. It then brings the reader up to speed with the state of this project and related projects. Afterwards, the concrete goals for this thesis is listed. Finally, the overall structure of the thesis is presented.

1.1 Background

The term aquaculture loosely refers to any farming of aquatic organisms. This includes farmed fish, shellfish, shells, aquatic plants. The term farming applies when any growth enhancing actions like breeding, feeding or protection from predators are applied. This means that fishing of wild stock is not included in this term. This distinction is made because in most cases across the world, the wild populations of these categories has already reached their peak sustainable output. This is illustrated in Figure 1.1, where the global aquaculture production is plotted together with the catch obtained from wild stocks. Although these numbers are most likely inaccurate and contains a wide variety of wildlife outside the scope of this report, it shows a definitive trend. If food production from aquatic sources is to increase further, farming techniques must be employed increasingly.

China is currently the worlds largest aquaculture producer overall, and almost dominates several categories like carps and oysters. Norway and Chile are the largest producers of

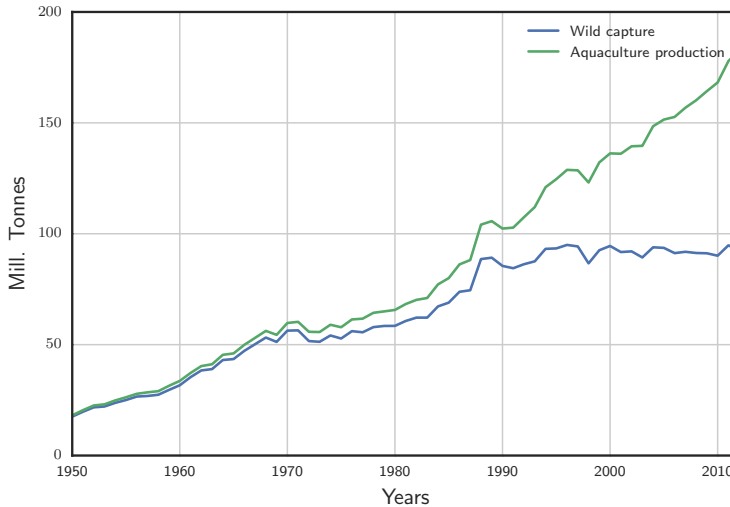


Figure 1.1: Global statistics of commercial harvesting of aquaculture and aquatic wildlife (Source: FAO).

salmon at 33% and 31%, respectively ([Subasinghe \(2005\)](#)).

Increased use of aquaculture has multiple implications for the fish/aquatic processing and refinement industries. The first one being that there is an increased amount of planning possible with farmed resources, this allows for a steady supply of raw materials. A production line or system can thus operate around the clock, increasing the profitability of the investments into automatic equipment over manual work.

The second implication is that there will be a larger degree of diseases and malformation due to the conditions of large-scale farming. One reason being that no natural selection occurs, as the salmon are protected from dangers. Another reason is simply the close quarters in which the fish reside. Some of the underlying issues can be improved by changing the farming techniques, but it is impossible to completely emulate the conditions of the wild. The systems in place, manual or automatic, must account for these imperfections. Customers are very sensitive to color, shape and smell of the products they purchase, and is thus directly linked to the value attributed to the end product.

A third implication is that the gross tonnage of fish that needs to be processed will necessarily increase considerably to meet demands and keep the farming profitable. This puts considerable strain on the parts of the supply chain where manual labor is employed.

1.1.1 The Norwegian Seafood Industry

The full seafood industry in Norway generated 46,5 billion NOK and kept about 47 400 full-time-equivalent (FTE) employees. These figures are roughly split in half between the

aquaculture and fishing industries. Specifically, the fish farming industry employs about 5700 people, working in some 1200 fish farms. These farms are dominated by salmon (Kristiansen (2014)). Norway is currently the sixth on the list of the world's largest fish farming nations. Figure 1.2 shows that the fish farming industry in Norway has had an incredible growth in the later years. These numbers reflect a similar trend as in the global aquaculture industry.

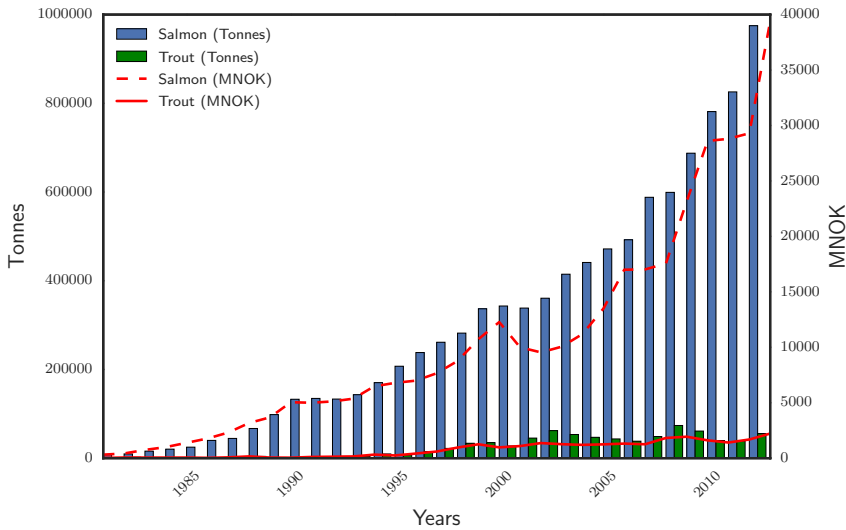


Figure 1.2: Norwegian exports of salmon and trout (Source: SSB).

While the fishing-based industry has a contribution to the gross domestic product (GDP) per FTE well above the national average of 0.83 million NOK per FTE, the industry that deals with the refinement and processing of fish does not. The contribution to GDP per FTE from fishing was 1.21 million NOK per FTE, while the fish processing industry lies at 0.68 million NOK per FTE. The aquaculture breeding industries lies comfortably at 0.97 million NOK per FTE (Sandberg (2014)). These numbers are from 2012. This clearly indicates that the part of the seafood industry that deals with the refinement and processing of the raw products are facing more economic challenges. Note that these numbers includes the full economic chain, which means that the fish farming industries are rated higher due to more third-party expenses due to fish feed.

Another statistic that reflects parts of the same problem above, is that the salary costs has increased significantly in the later years. In 2013 the salaries in Norway was 55% higher than the average among the trade partners in the EU, and as a result the processing industry has had problems with profitability for some time (Digre (2014)). Since this part of the seafood industry generates less revenue per employee, there is tough competition

from other countries that has lower salaries. Shipping whole frozen fish to another country for refinement might be more economically viable than doing the processing locally.

On a processing line with 80 to 120 tonnes salmon per shift (roughly 25000 units), two to four employees must exclusively perform the task of quality grading. They work 7 hour shifts, separating the salmon into three categories. The categories are *superior*, *ordinary* and *production*, where the highest quality is superior and worst quality is production. The classes of superior and ordinary salmon is much closer to each other than the production salmon - as that class contains the fish that is considered unfit for consumption. Both superior and ordinary salmon are used for human consumption. Typically, most of the salmon are put in the superior class - approximately 90% to 97% (Misimi et al. (2008)). In light of this, by reliable separating out this class through an automatic system - a fraction of the manual labor is needed.

1.1.2 Common Defects in Farmed Salmon

Since salmon is a biological product, it has much variance in both appearance and quality. In addition to this, the definition of the quality classes are not explicitly defined, and will likely vary from company to company. It will also vary depending on which operator is performing the grading. This section will generally define the classes of defects present in farmed salmon which is applicable to this thesis. Defects not externally visible, such as muscular defects for example, cannot be classified by a system exclusively relying on exterior imaging. The possible defects that can be caught by such a system includes skeletal deformation, sexual maturity and external blemishes - such as wounds.

Spinal Deformation

One of the possible defects in salmon is spinal deformation. The cause for the deformities cannot be attributed directly to a single factor. It has been shown that parasitic and bacterial infections, malnutrition, incubation temperature, light conditions, water quality and pollution all contribute towards the risk of developing such deformities (Vågsholm and Djupvik (1998)). While combating the conditions leading to deformations is important to improve the living conditions of farmed fish, the focus for this thesis will be major deformations that affects the shape or symmetry of the salmon. Fjellidal et al. (2007) describes three of the most common deformities in farmed salmon. They are compression, fusion and dislocation. An X-ray of a salmon without any skeletal deformations can be seen in figure 1.3, and one with an deformed spinal column in figure 1.4. Even if malformations of the spinal column is present does not necessarily mean that it is externally visible - or quality degrading. Externally visible deformations can occur when the severity causes the spine to curve in an abnormal manner, for example through fusion and dislocation. An aim in this thesis is to primarily look at two signs of deformations - humpback and short tail. The first occurs as a result of an abnormal curvature of the spine, possibly in combination with inflammation of the surrounding tissue. Short tail is the given name for compression throughout the entire vertebral column. Figure 1.5 shows a comparison of a regular salmon and one with a compressed spine. The external result is a wider (taller), stunted, appearance - which is where the name short tail comes from.

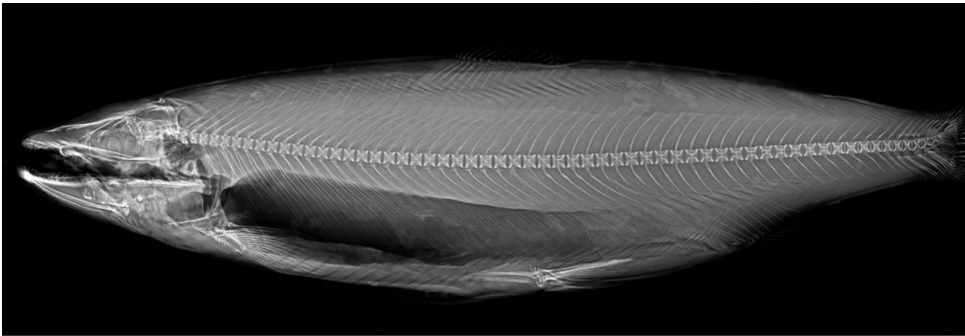


Figure 1.3: Vertebral column of salmon without signs of deformations (from [Witten et al. \(2009\)](#))

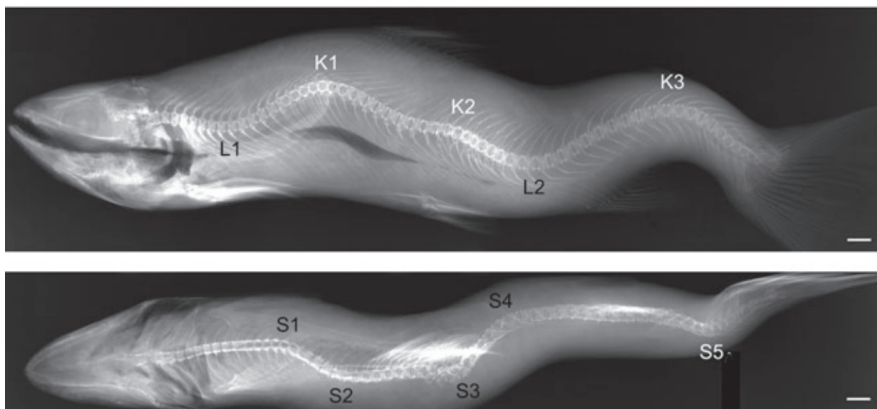


Figure 1.4: Multiple extreme deformations of the vertebral column of an adult salmon (from [Fjelldal et al. \(2012\)](#))

Depending on the severity of the deformations, the salmon is either classified as ordinary or production. Salmon in the superior class is typically free from deformations, but the separation between ordinary and superior is not cut and dry.

External Blemishes

In total there are a few common external blemishes to consider. The type that appears and is handled in this thesis is wounds. This can either be old wounds or new wounds. Wounds in salmon can appear in all sizes, and primarily occurs on the sides. When the salmon is processed, the wounds can be healed - but still causes a downgrading of the quality due to the scarring. The wounds themselves can start out as small scratches, which are then infested by bacteria, causing the wounds to expand. In waters of high salinity, such as seawater, the bacteria *Moritella viscosa* is found to be the largest cause of these wounds. The outbreak of that type of bacteria often occur during periods of cold water, and is for that reason called winter wounds or winter ulcers. The salmon can recover from these



Figure 1.5: Comparison of a regular salmon with one that has a compressed vertebral column, also known as short-tail (from [Witten et al. \(2005\)](#)).

wounds as the water gets warmer, but the scarring remains.

Other external blemishes can occur, but will not be considered in this thesis. An example of this can be sexual maturity, which changes the overall color of the salmon and changes the taste of the meat. None of the salmon in this thesis exhibited this defect.

Class Definitions

The superior grade is defined to be salmon of streamlined shape, and no external blemishes. The ordinary grade includes fish with minor defects such as unsymmetrical shape or some imperfections such as a minor external blemishes. Production constitutes the class of all salmon that does not fall into the two previous categories. The production class is not used for human consumption, and is either ground-up and used for other purposes or discarded completely. The primary cause of being labeled as production is body deformities, namely humpback and short tail ([Misimi et al. \(2006\)](#)). An aspect that makes classification into these classes difficult, is that each type of defect might appear on the same subject. The salmon might have an amount of deformity and blemishes that is within acceptable limits for the ordinary class when looking at each defect separately. When the two defects are combined on the same salmon, the verdict might change. An automatic quality control system should therefore be able to account for such sliding combinations of the various defects.

1.2 Benefits of Further Automation

Most low-hanging fruits for automation within the many industries has already been picked, and the food industries are not an exception. The tasks involving object recognition and decision making based on visual information has proven to be especially difficult. Humans are inherently skilled at quickly determining what an object is and select a course of action based on visual input. Transferring the same process to an artificial system is not simple.

An algorithm seeking to perform the same tasks would need to compare to the human operators in both quality and speed for it to be beneficial. Due to these difficulties, much of sorting and processing of fish is still done manually. Compared to many other branches of industry, the use of sensors in the fish processing industry is relatively moderate. This is of course related to the fish processing industry not being at the front of the technological development, and having a lower degree of automation than for instance the automotive industry (Buljo et al. (2013))

Automation of industrial processes has been going on for decades. This has especially been the case for high-cost countries, like Norway. Due to a high general level of salaries, the potential gains from automating a process can be significant. In the short term, this removes the need for certain jobs. In the long term the work force will adapt, however, and become more competitive with cheap labor in low-cost countries. This motivates the need for automation from the perspective of the local businesses. It also provides motivation from an environmental perspective, as transporting raw materials across the world has its toll on the environment. Refining the fish first removes the unnecessary part of the fish, and only transports the valuable parts. There is also the case of shipping the fish for processing, and then returning it back for sales in the original region - which can be avoided if it were to be processed locally.

Norwegian industries typically needs an edge other than strictly economic to be able to compete. Being close to the source of the resource was a big advantage one or two decades ago. Improved technology for freezing and storage in combination with cheap transport costs, instead allowed the industry to ship the resources abroad for processing (Egeness (2013)). This involves freezing the product once or twice, which affects the quality. As a result of this, almost 80% of the salmon exported from Norway today is fresh whole salmon to attempt to differentiate from the cheaper, frozen alternatives. Automating the processing will speed up the process from receiving a fresh fish to the packaged fish. This might yield some benefits in the shelf-life of the product. Pelagic fish, cod for example, is mostly exported frozen due to lacking automated solutions. Although automation of salmon processing has large potential, automating processing for pelagic fish is in its infancy - and a prime target for this technology if it proves to be effective. Automation of Salmon processing has come further, much due to the steadiness of supply from fish farms.

In Digre (2014) the notion is presented that introducing new technology in the fisheries will reduce the number of employees, but not the expenses in salaries. The reasoning is that the untrained labor is reduced, but that it comes with an increase in the areas of automation and process control, where the salaries are higher. The motivation to automate the processes should therefore lie in other areas, at least initially. There are potential gains by introducing new technology in the improvement of product quality and improved capacity due to the higher speed. An automated system is not subject to human error, and has inherently less variability in output product. Additionally, the increased precision can yield a better utilization of the raw product through a more correct sorting. If salmon from Norway retains an impression of quality abroad, a higher price can be justified.

The salmon processing industry in Norway has equipment available to automate a variety of single tasks. These machines are very specialized however, and requires manual

intervention to position and prepare the salmon between the various steps. Machine vision can coordinate these systems with each other. A machine vision system placed somewhere in the processing line not only provides immediate information to for example a quality control system, but also to all machines further down-line. One can imagine a robotics system being placed somewhere for example, where the visual system can be made much simpler due to the fact that analysis on higher quality imaging has already been performed earlier. The low involvement of robotics in food processing is also due to the fact that food products, and seafood products particularly, are highly variable both in shape, size and structure, which poses a major problem for the development of sensor systems (Litzenberger (2009))

In recent years, the speed of general purpose computers has steadily increased. Additionally, the cost of high-speed cameras has dropped significantly. Combining these two effects can make tasks involving visual information feasible from both economic and computational standpoints. Solving these task with general purpose computers frees the deployed systems from having specialized hardware, which would increase the cost significantly. This project uses off the shelf hardware with a computer vision system built from scratch to avoid using expensive, pre-built, systems.

If the approach used in this task proves to be successful, it could be an enabling technology for other applications. If the 3D-profiles can be used for visual inspection of fish, there is nothing that prevents the same approach from being used in other applications. One possibility is using the classifier to not only determine the quality of the fish, but also output the concrete regions of poor quality. This information can be passed along to another system capable of removing these regions, possible with the help of robotics.

Successful large scale development of automated solution can also be an attractive technology to export to other nations. Exports of oil related technology is today the second largest industry in Norway. Additionally, increasing the general activity level within the field of machine vision can be a positive trend to further the general field of expertise in Norway. The problems faced by the seafood industry are not unique, at least in Norway. Many of the concepts explored in this thesis might also be directly or indirectly applicable to other industries.

1.3 Previous Work

This section goes through the initial work done on this project by the author during his summer internship at SINTEF and part-time during the fall of 2014. The intention is to bring the reader up to speed with the state of the project when the undertaking of this thesis was started. This is important, as the computer vision system was not in a state where an experiment could immediately be scheduled. Due to this, much time was spent on hardware interoperability and software development.

The primary purpose of the summer internship at SINTEF was to attempt to create a system that could obtain a height profile of an object moved through a laser-line. Previously, the research done at SINTEF had used commercially developed systems for the

same purpose. They realized that if broad implementation of their solutions was to be obtained, the cost of the machine vision system would need to be brought down. Building an in-house solution allowed the system to be tweaked to the needs of the application. The goal during the summer internship of 2014 was to develop such a scanner from scratch utilizing a single camera. This involved developing a camera calibration routine and a library capable of computing the world coordinates in real-time. The goals were met, but the performance obtained was lower than desired. While this is acceptable in an lab environment, where the speed of the conveyor belt can be reduced - it is not acceptable in the industry. Additionally, only scanning from one side was implemented. The old software routines was the result of prototyping, and much needed to be rewritten or done another way during the work on this thesis.

The calibration routine did not work as expected when applied to the bottom cameras. The accuracy from cameras positioned at an angle, as opposed to directly from above, was not within the desired limits and had to be improved during the thesis. Features critical to the success of this thesis was lacking. The camera system used monochrome images and did not output color images, which is needed in the classification of wounds. The camera system was set at the same, fixed, frame-rate for each camera. This needed to change to a system using explicit triggering, to properly synchronize the three cameras.

1.4 Price of Comparable Systems

The price of comparable, commercial, systems was an important factor in the decision to create a 3D- scanner from scratch. An price estimate of an industrial range-scanner camera is somewhere in the range 40 000 - 70 000 NOK without a lens. The upper part of this range would be required to get a system that includes both range and color imaging. For 3D-scanning with full coverage using three cameras as in this thesis, the cost exceeds 200 000 NOK just for the equipment. The software needed to tie these cameras together adds additional cost. The cost for the equipment used in this thesis, including three lasers, three cameras and LED lights for illumination is well below 70 000 NOK.

If the price of the system can be reduced to a mere fraction of comparable systems, with an acceptable accuracy, the threshold for implementing the system is lowered drastically. Not only can the system be taken into use by different companies, but also at multiple locations on the same processing line. This enables the availability of 3D-imaging as the salmon is being processed. There is also nothing keeping the system from being used in other industries with similar needs, but perhaps different margins than the salmon industry.

Another factor in the decision to develop the system from scratch, was that a much higher degree of flexibility. If a feature one needs is missing from the industrial system, there is not much one can do - except performing an expensive upgrade if possible. With a custom designed system, one must invest time in development - but this is largely a one-time cost. This allows to specialize the choice of optics and algorithms to the needs. Additionally, having a cheaper system allows for more rapid upgrades as optics and camera interfaces improve.

1.5 Related Work

In general, the Salmon can be divided into three groups according to external quality. These are superior (no external blemishes), ordinary (minor degree of blemishes) and production - where parts of the Salmon cannot be used for human consumption. Sorting between the two classes superior/ordinary and production was performed at SINTEF Fisheries and Aquaculture previously. A quality grading system that utilized a single image per Salmon from the side was used to obtain an estimated sorting reliability of 87% (Misimi et al. (2006)). Only the shape, which was extracted from the outline of the 2D-image - was considered. This was later applied to sorting between superior and ordinary using the same approach, as the production class was found to appear in small quantities. The sorting reliability between superior and ordinary was estimated to be 90% (Misimi et al. (2008)). It should be noted that the difference between the classes of superior and ordinary are magnitudes smaller than the distance to the production class, which makes it much more difficult to separate. These two cases are similar in the type of fish and the quality grading classes.

A related project used a sheet-of-light laser and camera to obtain a height-profile of pelagic fish from one side (Mathiassen et al. (2006)). The project used a commercial system. It was a proof of concept for the technology, or approach, used in this thesis. The conclusions of that study was that to obtain the speed necessary, image processing needed to be offloaded on an GPU, which led to the work preceding this thesis.

1.6 Goals and Objectives

The following section lists the goals and objectives of this thesis.

- Implement a complete 3D machine vision system with emphasis on applicability to industrial purposes, and not just laboratory experiments. This places constraints on the real-time performance of the implementation.
- Resolve any practical issues with regards to the extracted point cloud, that makes analysis of the geometry feasible. Emphasis is placed on the robustness and correctness of the point cloud with respect to feature extraction. There are no concrete goals for this point, but justifies time spent on improving the approach as a whole for future applications.
- Perform an experiment by scanning salmon, to obtain a dataset.
- Develop features that are able to describe the deformities present in salmon, and use a classifier to separate between superior and the deformed salmon in ordinary/production from the dataset.
- Implement detection of externally visible wounds, either as a separate routine or as a part of the classifier used for deformity detection.
- Evaluate the methods used, and suggest future improvements to the general approach or implementation.

A big part of the challenge with the task at hand will be to compensate for the inherent variation present in biological objects.

1.7 Structure of the Thesis

- **Chapter 2:** Contains a short literature review of machine learning, with emphasis on support vector machines. Some discussion of why SVM is suitable for this project is also found here. Some concepts within shape representation and geometrical transformations are also introduced.
- **Chapter 3:** The equipment used and the steps necessary to acquire 3D-coordinates of objects, is detailed.
- **Chapter 4:** In this chapter, the steps taken to improve the quality of the raw 3D point cloud and the extraction of features for classification is described.
- **Chapter 5:** Using the geometric and color features, classifiers are trained and the results of these are presented in this chapter along with some discussion and commentary where necessary.
- **Chapter 6:** This chapter contains general discussion regarding the results obtained and the strengths and weaknesses of the methods employed.
- **Chapter 7:** This final chapter summarizes the work and draws conclusions regarding the importance of the findings. Recommendations for further work is presented.
- **Appendices:** There are three main appendices. A table of the dataset, with comments, and two smaller code snippets performing specific functions. The remainder of the implementation is too large to embed in the thesis.

Chapter 2

Literature Review

This chapter briefly introduces some of the key concepts used in this thesis.

2.1 Classical Statistical Learning

Generally, machine learning considers a set of n samples and from this tries to find patterns or properties of this data. These patterns or properties can then be used to draw conclusions regarding new samples. The type of machine learning considered in this thesis, is supervised learning - where each of the samples has a corresponding class. The overall goal of this type of machine learning therefore becomes the problem of how to best separate these classes. For most applications, simply separating the classes in the known, labeled, set is not enough. One typically wants to find a pattern that provides the separation that *generalizes* best to new samples.

$$\mathbf{x} = [x_1, x_2, \dots, x_d]^T \tag{2.1}$$

The samples are described by a vector of *features*, which each describe some property of the sample. These features can for example be measurements. The problem of separating the classes becomes a problem of creating a d -dimensional boundary between the samples. In this thesis, binary classification is considered, namely the separation of two classes. In the two-dimensional case, a simple separation can be obtained through a linear discriminant function, as shown in figure 2.1. Classical supervised learning finds such a line, or hyperplane in a general dimensional space, such that the separation is optimal with regards to some metric.

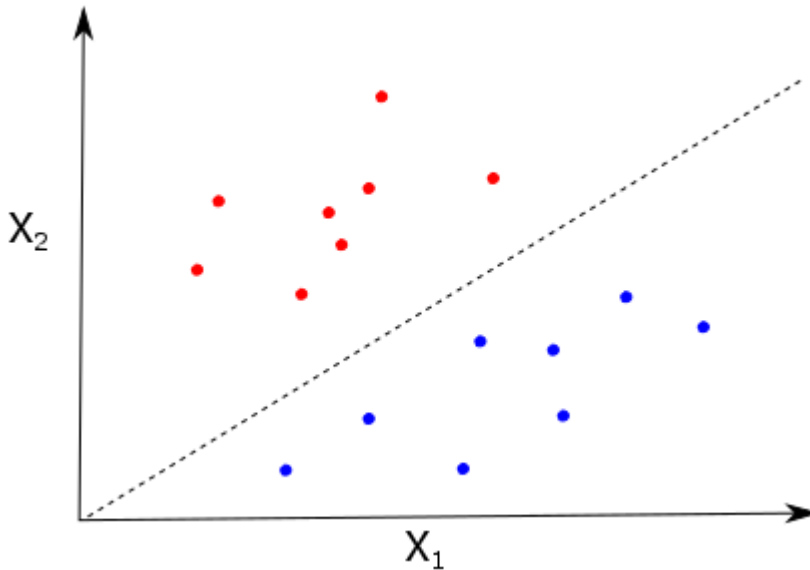


Figure 2.1: A separation between two classes described in a two-dimensional feature space.

Supervised learning can be roughly divided into two areas, one based on statistics and one based on optimization. The class of learning algorithms in the statistical class often assumes that the features follow some probability distribution, and the separating hyperplane is optimally found with regards to this distribution. The class based on optimization makes no such assumption, but rather attempts to minimize some objective function. This can for example be geometric distance between the samples. This thesis employs the second strategy.

2.1.1 Curse of Dimensionality

A large part in developing a successful machine learning application is finding the best features to describe the difference between the classes. One might think that adding new features improves the performance of the resulting classifier indefinitely. This is unfortunately not the case, even if each feature provides new information, due to an effect called the curse of dimensionality. Adding new features to the feature vector can add new information, but also increases the dimensionality of the feature space. Consider a two-dimensional feature space. If a feature is added to form a three-dimensional feature vector, the feature space becomes cubic. If the number of samples stays the same, the effective sample density is reduced exponentially. The predictive power of machine learning algorithms decreases as the sample densities becomes lower. Some algorithms are better at handling low density feature spaces, but all are afflicted by it to some degree. One therefore not only looks for features that provides new information regarding the two classes, but the feature combination that provides the most information collectively within a reasonable dimension.

2.1.2 Overfitting

Another important concept in machine learning is that of overfitting. This occurs when the machine learning algorithm starts to describe random error or noise instead of the actual pattern. This makes the classifier become very adept at separating the samples in the training set, but performs poorly when applied to new samples. That is, the classifier memorizes the dataset instead of learning from it. A good example of overfitting can be found from regression. Assume that one wants to fit a noisy dataset to some function. If the degrees of freedom is high enough, and is fitted to the data without any form of regularization - the result might become something like figure 2.2. The basic premise of overfitting is the same for machine learning as for regression. Fortunately there are methods that helps to manage this problem, as well as learning algorithms resistant to its effect. One general method that can be applied is cross-validation.

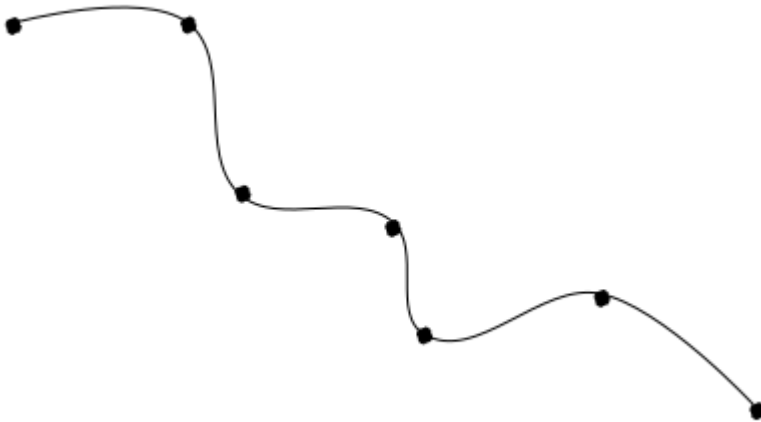


Figure 2.2: Overfitting in the case of regression, the trend in the points might be better described through a linear function than a curved function which correctly minimizes the distance to each point.

2.1.3 Cross-Validation

When training a classifier, one needs some way to assess how that classifier will generalize to new data. This is useful both for selecting the optimal model parameters, and estimating the performance of the overall classifier once the model parameters has been fixed. The goal is therefore to predict the accuracy of a given model. Training a classifier and estimating the accuracy on the same dataset introduces a bias into the calculation, as the algorithm training the model knows the full set beforehand. In that instance, the accuracy reflects how well one can specialize the model to the dataset, not how well it generalizes. This leads directly to an overfitted model. To mitigate this one usually employs a scheme

which divides the dataset into training sets and validation sets, where a subset of the data is used exclusively for validation. This process can be repeated several times using a random split of the data in for example 80% training set, and 20% validation set. The drawback with this approach is that there is no guarantee that each sample in the dataset appears both in the training and validation process.

Cross-validation approaches the problem slightly differently. It too, splits the data into different subsets, but ensures that each subset is used both as a validation set and training set. It does this by splitting the dataset into k subsets, which is used once as a validation set. The model training is then performed on the remainder of the dataset. This is called k -fold cross validation. Often chosen equal to 10, which means that the model training is performed 10 times on 90% of the dataset, and validated on 10%. Each sample is then in the validation set exactly one time. The predicted classes is then compared with the true classes to form an accuracy estimate.

The 10-fold cross-validation is by many considered to be the standard procedure, which works fairly well given that the size of the dataset is large compared to the variability of the samples. When this is not the case, one periodically experience a bad split, where perhaps all the samples that describes a certain trait is placed into the validation set simultaneously. In the face of this issue, one typically has two choices. Either repeat the experiment a certain amount of times and average the results, to reduce the effect of a few bad splits. The bad splits that do occur does of course reduce the overall accuracy slightly, but this is better than getting an overly optimistic estimate. The second approach is to take the cross-validation to the extreme by selecting $k = 1$. This is known as leave-one-out (LOO) cross-validation, and involves training a model on all the samples except one - which is then used to validate. This completely negates the effect of a bad split, unless a single sample is substantially different from the rest. For many applications, this is not feasible - as the time it takes to train all those models might be unrealistically high. In this thesis, the final model parameters is validated an additional time using LOO cross-validation. The prototyping and model selection is done using other cross-validation strategies however.

Not too many conclusive studies has been performed to evaluate the methods of assessing classifier performance, as the error in part depends on the data set. [Bouckaert and Frank \(2004\)](#) finds that performing a 10x10 cross-validation, yielded the best replicability - in which the bad splits were minimized. The drawback of this is that the model needs to be trained 100 times.

When the number of samples representing each class in the dataset is not of the same number, one sometimes wants to ensure that each class is equally represented in both the training and validation sets. This is called stratified cross-validation, and is simply done by dividing the dataset by class, performing random splits, and merging the classes back together. This avoids the situation where many samples from the class of least size is placed into the validation set simultaneously. If the remaining samples from that class in the training set is too low, the quality of the model will be poor and yield an unrealistically low prediction rate for that class.

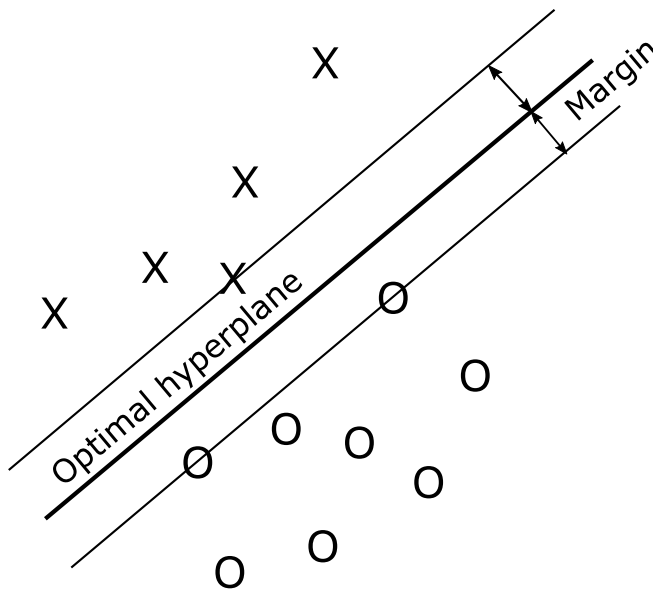


Figure 2.3: The optimal hyperplane is the hyperplane that separates the classes with the maximal margin.

2.2 Support Vector Machines (SVM)

Support vector machines (SVM) is a type of classifier developed in the 1990s originally intended for character and text recognition. SVM can be applied both to regression (SVR) and classification of binary classes (SVC). Multiple classes can be taken into account by utilizing multiple binary classifiers in so-called one-vs-all classification. When applied to binary classification, the support vector machines seek to maximize the margin of the an optimal hyperplane. The optimal hyperplane for a linearly separable set is defined as the hyperplane that separates the two classes with maximal margin, as illustrated in in figure 2.3. The support vectors are the data points that coincide with the margin. The assumption is that the larger this margin becomes, so does the classifiers ability to generalize to new data.

2.2.1 Linear SVM

The optimization problem for support vector classification will first be introduced for the linear case, and then extended to the nonlinear case. Note that while the equations presented here are correct, they are not solved directly in practice. Implementing a solver that effectively treats large datasets and features with many dimensions is difficult, and requires different approaches and heuristics. The equations presented here best illustrates the objective of SVM, without delving into too much detail.

SVM uses the dot product as a similarity measure. For linear SVM, this is simply the dot product between two vectors in the input space, R^N .

$$\langle \mathbf{w}, \mathbf{x} \rangle + b, \quad \text{where } \mathbf{w}, \mathbf{x} \in \mathbb{R}^N, b \in \mathbb{R}, \quad (2.2)$$

Where $\langle \cdot, \cdot \rangle$ denotes the inner product, thus yielding a scalar value. The sign of this can be used to realize a decision function

$$f(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b), \quad (2.3)$$

Given the training data \mathcal{X} with binary class assignments \mathbf{y}

$$\mathcal{X} = (\mathbf{x}_1, y_1) \dots (\mathbf{x}_L, y_L) \quad x \in \mathbb{R}^n, \quad y \in \{+1, -1\} \quad (2.4)$$

A convex quadratic optimization problem for the optimal hyperplane in the case of a linearly separable data set can be written as

$$\begin{aligned} & \underset{w, b}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{w}\|^2 \\ & \text{subject to} \quad y_i [\langle \mathbf{w}, \mathbf{x}_i \rangle + b] \geq 1, \quad i = 1 \dots L \end{aligned} \quad (2.5)$$

The support vectors are the data points where the constraint is at equality. The inequality constraint ensures this by scaling \mathbf{w} and b . The *margin* which is the distance from the hyperplane to the closest points equals $\frac{1}{\|\mathbf{w}\|}$. This result can be obtained by projecting points on the margin of either side of the optimal hyperplane onto the normal vector $\frac{\mathbf{w}}{\|\mathbf{w}\|}$ of the hyperplane. The distance from the margin on one side to the other is then $\frac{2}{\|\mathbf{w}\|}$. It follows that maximizing this margin involves minimizing the length of \mathbf{w} , which is reflected in the objective function.

The constant on the right hand side of the inequality can be any positive number. The positive constant is needed to ensure that both sides of the equations are scaled under multiplication. Minimizing $\|\mathbf{w}\|$ using a constraint with > 0 would not yield meaningful results, as the left hand side could be scaled freely. This can be seen by multiplying with a number $[0, 1]$, which can reduce the objective function indefinitely (Smola and Schölkopf (1998)).

The dataset could be linearly *unseparable* however - perhaps due to noise, mislabeling or nonlinear patterns. This causes the optimization problem to be unable to satisfy the constraint - and becomes infeasible. To make the problem feasible, slack variables are introduced which allows the constraint to be violated under a penalty to the objective function. This is known as a soft margin hyperplane. A parameter $C > 0$ is a given constant that weights the penalty of samples on the wrong side of the hyperplane. Changing this both changes the orientation and bias of the hyperplane, and is used as a tuning parameter to reject outliers. A normal strategy is to keep $C = 1$ and then reduce it further if the results are poor due to outliers or noise.

$$\begin{aligned} & \underset{w, b, \xi}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^l \xi_i \\ & \text{subject to} \quad y_i [\mathbf{w} \cdot \mathbf{x}_i + b] \geq 1 - \xi_i \\ & \quad \quad \quad \xi_i \geq 0 \\ & \quad \quad \quad i = 1 \dots l \end{aligned} \quad (2.6)$$

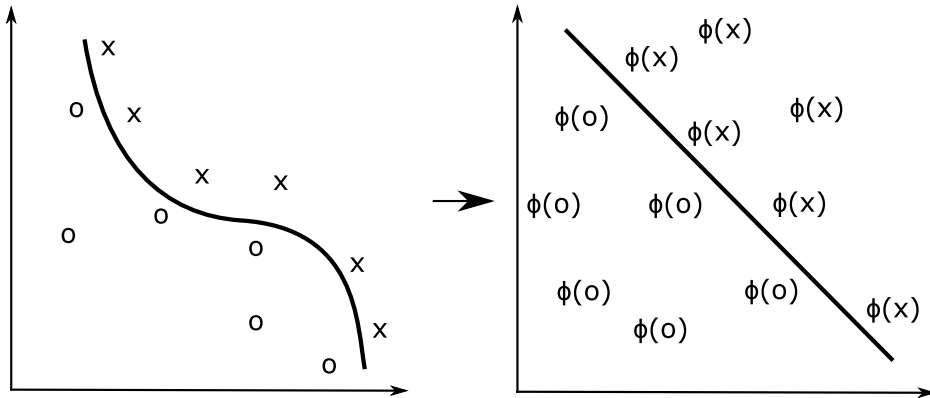


Figure 2.4: The mapping Φ embeds the data into a higher-dimensional space where the nonlinear pattern appears linearly (adapted from [Shawe-Taylor and Cristianini \(2004\)](#)).

Many solvers can be used to solve this convex quadratic optimization problem. Usually it is solved on its dual quadratic form, by maximizing the Lagrangian multipliers. Although the optimization problem can be solved using conventional quadratic solvers, more heuristics are usually needed to form an efficient SVM training algorithm.

2.2.2 Nonlinear SVM

One of the great aspects of SVM is its ability to diversify. The same optimization can be performed to find a nonlinear support vector classifier. The equations presented for the linear case are remarkably similar to the nonlinear case in appearance. The major change is that the dot product, which was performed on two vectors in \mathbb{R}^N for the linear case, instead is performed in a general dot product space \mathcal{H} . The type of dot product space is defined by a mapping $\Phi : \mathcal{X} \rightarrow \mathcal{H}$, which maps the input space \mathcal{X} to the feature space \mathcal{H} . This mapping is typically nonlinear. The similarity measure is defined through the dot product of such a mapping of each element, which is called a kernel.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \quad (2.7)$$

This kernel dot product appears directly in both the dual optimization problem and the resulting decision function. A simple way to think of kernels, is that a nonlinear kernel transforms the data from its' input space to another higher dimensional space - where the feature vectors of the classes are linearly separable. Hyperplanes are still used for nonlinear SVM, it is the input space that is mapped to an alternate feature space. Figure 2.4 illustrates this point. In theory, any dataset of two categories can always be separated linearly by a hyperplane if the data is mapped nonlinearly to a sufficiently high dimension ([Duda et al. \(2012\)](#)).

Below, three of the most common kernels are listed, where γ , r and d are kernel parameters.

1. *Linear:* $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$

2. *Polynomial*: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d, \quad \gamma > 0$

3. *Radial basis function (RBF)*: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \quad \gamma > 0$

The kernel and its parameters are specified manually, and are not free variables in the optimization problem. These parameters are usually found by performing a exhaustive grid-search of a wide range of parameter combinations. Cross-validation is performed at each parameter combination to determine the best combination. For RBF, one would search the parameter space of the regularization parameter C and kernel parameter γ . Heuristics exist that attempts to point the search in the right direction. This however, serializes the grid-search - at least to a certain degree. It is therefore more common to do the exhaustive search by for example employing distributed computing.

RBF is often recommended as the first kernel to try, but its success depends on the dataset itself. If the number of features is sufficiently high or the dataset is linearly separable, other approaches might be better. The best combination of parameters for RBF are guaranteed to not yield worse performance than linear SVM however (Hsu et al. (2003)). This of course assumes that the best parameter combination is found through the grid-search. The polynomial kernel is normally not used, as it has three tunable parameters compared to two for RBF and one for linear SVM.

By defining the inner products in terms of the original space, the computational overhead of moving to a higher dimensional space is kept much lower. The Gaussian RBF kernel for example, transforms the input space to an Hilbert feature space of infinite dimensions. Only the dot product of this space is utilized in the computations, however. To avoid confusion, a quick example using the RBF kernel for an input space in \mathbb{R} is now included.

$$\begin{aligned} K(x_i, x_j) &= e^{-\gamma \|x_i - x_j\|^2} \\ &= e^{-\gamma (x_i - x_j)^2} = e^{-\gamma (x_i^2 + 2x_i x_j - x_j^2)} \end{aligned}$$

The middle term involving both elements can now be separated out and expanded as a Taylor series.

$$= e^{-\gamma (x_i^2 - x_j^2)} \sum_{n=0}^{\infty} \frac{(2\gamma x_i x_j)^n}{n!}$$

It follows that this is result of an inner product of two vectors of infinite dimension. The mapping applied to each input vector of the inner product is shown below.

$$\Phi(\mathbf{x}) = e^{-\gamma (x^2)} \left[1, \sqrt{\frac{2\gamma}{1!}} x, \sqrt{\frac{(2\gamma)^2}{2!}} x^2, \dots \right]$$

This is known as the kernel trick, which allows the input vectors to be mapped to a high-dimensional feature space without explicitly computing anything in that space. The dot product in this space represents what the similarity value of the two vectors converges to as the dimension goes to infinity. In fact, the Gaussian RBF kernel is a combination of all polynomial kernels of positive degree. The linear kernel is actually a special case of RBF.

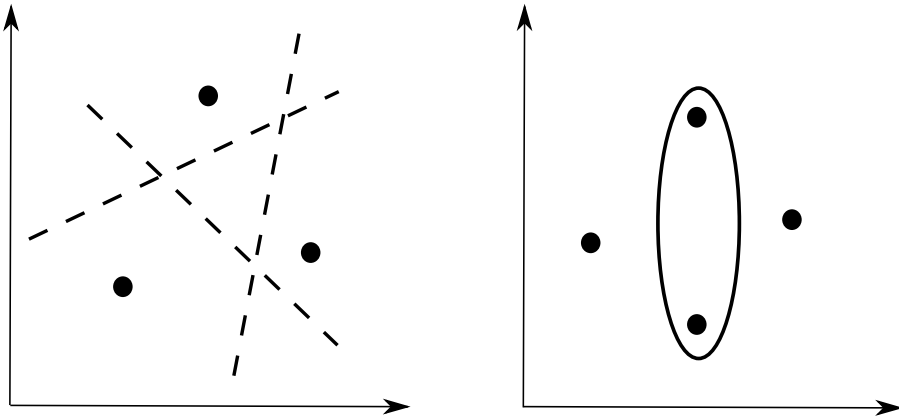


Figure 2.5: VC dimension example for linear separating functions in the plane (adapted from [Vapnik \(1995\)](#)).

2.2.3 VC Theory

An important property of SVM is that the complexity of the classifier is characterized by the number of support vectors instead of the dimensionality of the high-dimensional feature space. As a result, SVMs tend to have less problems with overfitting due to dimensionality than other methods.

The development of this class of learning algorithm was motivated by advances within statistical learning theory. In particular, the introduction of the Vapnik–Chervonenkis (VC) dimension, which can estimate to which degree a classifier can generalize to unseen data. The VC dimension itself is a scalar which places a bound on the risk achieved by a learning algorithm ([Vapnik \(1995\)](#)), also referred to as the capacity of a classifier. In plain terms, the VC dimension of a classifier model is the maximum number of points or vectors that can be separated (shattered) into any combination of two classes. Figure 2.5 shows an example using a linear discriminant function in the plane. The VC dimension in both instances is equal to 3, as the points inside the ellipse cannot be separated from the left and right points by a single, straight, line. By controlling the length of the weight vector $\|\mathbf{w}\|$ in SVM, the VC dimension can be controlled irrespective of the dimension of the space (theorem 5.5 [Smola and Schölkopf \(1998\)](#)). The minimization of the weight vector therefore also controls the capacity of the classifier. This is a central point, which explains why the SVM has the ability to classify without overfitting even in conditions where the dimension size outnumber the number of samples. SVM only uses a subset of the training points closest to the hyperplane, the support vectors, to generate a decision boundary. It therefore has a powerful ability to generalize.

Using VC theory, one can in theory estimate the kernel and kernel parameters that best suits the dataset in terms of risk minimization. Normally, cross-validation is used instead for the same purpose - as it does the same empirically.

2.2.4 SVM: Advantages and Disadvantages

Advantages

- Effective in high dimensional spaces (resistant to overfitting)
- Effective in cases where number of dimensions is greater than the number of samples
- Memory efficient, as it only uses a subset of the training points for classification
- Kernels can change the behavior completely without changing the classifier itself
- No need to reduce dimensionality through PCA for example as opposed to other algorithms that does not handle the curse of dimensionality (overfitting) inherently

Disadvantages

- Poor results if number of features much greater than samples
- No probability estimates, must be estimated by itself - which is computationally expensive
- Dataset must be scaled before classifying and predicting
- Only directly applicable to binary classification
- Performance can vary greatly depending on the type of kernel chosen, and there is not one best kernel for all problems.

The main reason for choosing this as the primary classifier type for this thesis is that the number of features present in this thesis is quite large compared to the number of samples. Other classifiers could still be utilized by using dimension reduction techniques. The number of samples in total would still be much too low for most other classifiers, however. One of very few alternatives would be the K-Nearest neighbor classification, which simply uses the label of the K closest samples to determine the most probable label. For further research, other classifiers can be considered - as the data set will increase when the system is implemented industrially.

Another advantage is that there exists software that implements advanced solvers especially for training SVMs. In particular, an implementation by the name libsvm ([Chang and Lin \(2011\)](#)) is widely recognized to be one of the best SVM implementations. It has been under active development since 2000, and has stood the test of time. Simply applying traditional methods like Quasi-Newton directly does not scale well with the size of the data set in terms of memory usage. To scale well, the problem must be decomposed into multiple subproblems, which are solved one at a time for a few variables. Libsvm bases itself on a solving strategy known as Sequential Minimal Optimization, which decomposes the large problem into smaller subproblems. At each iteration a small problem of a few variables is solved, without needing advanced optimization software. It includes heuristics that both reduce training time by pruning unneeded elements (shrinking) and intelligent caching of previous kernel values. The library itself is written in C++, but has a wide range of interfaces. In this project, LabVIEW has been used for camera acquisition,

general data processing and ties the various components developed in C++ and Matlab together. Unfortunately, the provided LabVIEW interface to libsvm was both outdated and poorly designed. During this thesis a new interface was therefore developed and freely released (available at <http://www.github.com/oysstu/LabVIEW-libsvm>).

2.3 Numerical Geometry and Shape Recognition

One of the aspects that is considered in this thesis to determine the quality of the salmon, is the geometric data. The range-cameras provide a description of the shape of the salmon in 3-dimensional space. This data must be interpreted in an invariant way to be able to compare one with another. This interpretation should be invariant to the scaling of the object, as the quality classes contain varying sizes. To be able to analyze and recognize the shape of objects, one must first decide on a suitable geometric representation.

The main methods of shape representation in 2D and 3D can be divided into four distinct groups (Siddiqi and Pizer (2008)). Each of these groups will briefly be introduced in the context of shape recognition. One of the representations is then chosen based on its applicability to shape recognition. Illustrations of each group can be seen in figure 2.6.

Boundary representation

This method represents an object as a collection of surface points, surface elements and curves. The normal along the surface and rate of change in curvature can thus be used to distinguish between shapes. Typically, one utilizes a standard object of the same topology and computes a smooth continuous mapping from the standard to the object in question. This approach facilitates analysis of the shape from its exterior, but does not take interior geometry into account. One can easily see applications in recognition of rigid shapes based on this, but it faces problems in biological recognition - where objects of different size and non-rigid deformation appear. An example can be to track the movements of an object through rigid transformations of a point cloud boundary.

Voxel displacement

The second representation also uses a standard object, which is deformed. The difference from the boundary representation, is that the interior of the object is divided into a grid of points (voxels). The deformation of these voxels are then represented by a displacement vector field. Information about the deformation of both the surface and interior is therefore available. This representation is very dense, as each voxel must be stored. Additionally, the displacement is typically modeled through differential equations - which is computationally expensive. Voxel displacement has been successfully used in alignment of 3D medical images (Studholme et al. (1999)). These applications deal, not only with the surface of the patient, but also the interior composition obtained with CT/MRI-scans. The voxel representation is therefore well suited for these applications.

Landmark representation

This representation has the same idea as boundary representation, except points are selected intelligently. The points can represent important geometric locations along

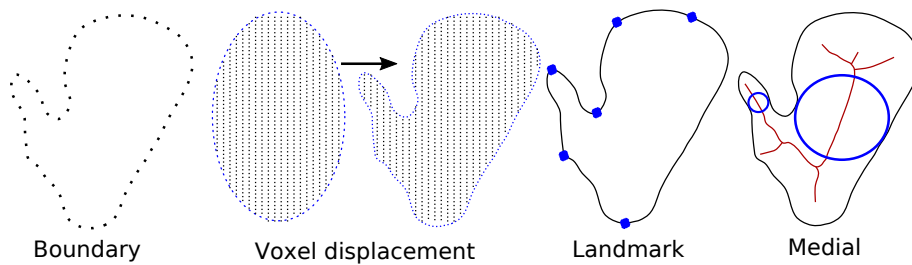


Figure 2.6: The four main shape representations (Adapted from Siddiqi and Pizer (2008)).

the boundary. Examples of such landmarks can be peaks and dips. Depending on how these landmarks are selected, they can be applied in shape recognition. This representation requires much heuristics for extraction of the landmarks, and is very application-specific. Due to the sparseness of the landmarks, a bare minimum of information is retained. The landmark representation has been applied successfully in face recognition (Zhao et al. (2003)). In general this representation depends heavily on the presence of good landmark candidates, which differ from application to application.

Medial axis

The final representation presented is that of the medial axis transform. The result of this transformation is a line inside the object, which branches outwards towards protruding parts of the object. A point on the axis is a location where a hypersphere that is tangent to the boundary in two or more locations can be inscribed within the object. In other words, the medial axis is the points where the largest possible hyperspheres can be placed inside the object. The full medial axis is the union of all these points. Like the voxel displacement representation, the medial axis contains information about both the surface and interior of an object.

The obvious choice would either be the landmark representation or the medial axis representation. The landmark representation has been successfully applied previously in automatic quality control of salmon from 2D-images (Misimi et al. (2008)). The landmarks chosen can be seen in figure 2.7. They were the widest and narrowest part of the fish, the width mid-way between these, and the lengths between them. This, in conjunction with the area of the front and back parts, contained enough information to separate the classes. The same approach could be used for the 3D-model, possibly extended by calculating the volumes instead of areas.

Medial transforms has been the subject of much research over the recent years. Medial descriptions of shapes have gained significant momentum due to their invariance to translation, rotation, scale and their ability to cope with moderate amounts of within-class deformation (Macrini et al. (2008)). It additionally has, as a method, a much greater level of generality to it than the landmark approach, which is very application-specific. The implications of this is that one to a greater degree can use results from other applications and can with more ease use the knowledge obtained in related applications.

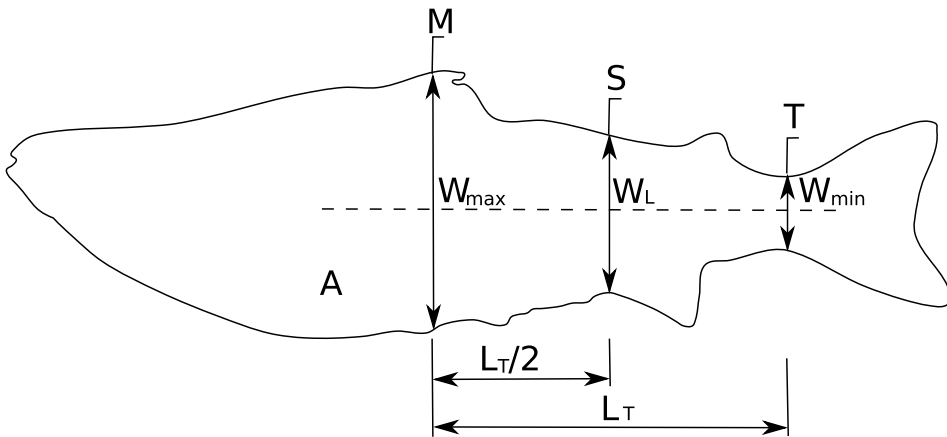


Figure 2.7: Landmark points utilized by Misimi et al. (2008) in quality control of Atlantic Salmon.

Although the work done in this thesis strives to represent the same information contained in the landmark features, the 3D-model provides much more information about the shape of the salmon than the 2D-images or one-sided 3D imaging used previously. Exploring a representation that captures more information is therefore interesting, especially since the method itself contains a larger degree of generality.

2.3.1 Medial Axis Transform

In 1967, Blum suggested the notion of a medial loci, or medial axis as a representation of the shape in 2D-objects extracted from images (Blum (1967)). Medial stems from the latin word *medialis*, which roughly translated means "in or near the middle". It is used in anatomy to describe the centerline of a body, much like a skeleton. It is this shape description that Blum aimed to obtain. The original definition of the medial axis transform (MAT) was formulated through the concept of a grass-fire, or wave propagation originating from the borders of an object. If the borders of an object is set alight simultaneously, the wave-fronts of the object would at some point meet near the middle. The collection of all these fronts would form a continuous line with branches, which would be the medial axis - or medial loci as it also is called.

A related idea is that of skeletonization, which can be computed by iteratively thinning the boundary until a centerline remains. In image processing this can be done by traversing the edge of the object, removing one pixel at a time. The result is not identical to the medial axis, however. Additionally, it has limited utility in analyzing morphology because the number of branches and configuration of the branches in the skeleton is highly sensitive to boundary noise (Yushkevich (2009)). The order in which the pixels are processed in a thinning operation, also affects the end result of the skeleton - which is undesirable. The medial axis is also susceptible to boundary noise, but methods has been developed that mitigates this problem (Katz and Pizer (2003)). Skeletonization can be affected by rotation and scaling, which the medial axis transform is invariant to.

The idea of wave propagation has been used directly to compute the medial axis by simulating the partial differential equation (PDE) in Equation 2.8. Here, W is the boundary, and $-n$ is the inward normal. The union of all points where the fronts meet will form the medial axis. Another popular method involves using an Euclidean distance transform on the interior of the object, and calculating its gradient vector field. Thresholding this vector field yields an approximation to the medial axis.

$$\frac{\partial W}{\partial t} = -n \quad (2.8)$$

A third method involves calculating the Voronoi diagram for the shape. Voronoi diagrams has been intensively researched for other applications within computational geometry. It is therefore been well developed in terms of algorithms, both in terms of speed and geometric stability. In plain terms, it divides the plane into regions according to the nearest-neighbor rule. This is formally stated in Equation 2.9, where $d(x,y)$ denotes the Euclidean distance function.

$$V_k(p) = \bigcap_{q \in S-p} \{x \in R^n \mid d(x,p) \leq d(x,q)\} \quad (2.9)$$

The expression to the right of the intersection operator defines a closed half plane bounded by the perpendicular bisector of p and q . The plane separates the points closest to q and those closest to p . The intersection of such regions gives a convex polotype, or convex polygon in two dimensions. Calculating this for all the points gives a structure as seen in figure 2.8. Each point on an edge is equidistant from exactly two sites, and each vertex is equidistant from at least three points. This property causes the edges and vertices of the individual regions to match the neighboring regions exactly, which causes the full region to be partitioned. This partition is called the Voronoi diagram, $V(S)$, of the finite point-set S . Although n sites give rise to $O(n^2)$ separators, only a linear amount contributes to an edge in $V(S)$ (Aurenhammer (1991)). Voronoi diagrams are used for many purposes, one of them being a discrete approximation to the medial axis.

When applied in the context of the medial axis, the focus is on the edges interior to the border defined by the points. The union of all the discrete vertices of the interior constitutes a discrete medial axis. As the sampling density along the boundary uniformly approaches infinity, the discrete medial axis will converge to the continuous skeleton (Schmitt (1989)), and thus the medial axis obtained by the grass-fire flow. Extracting the medial axis from the Voronoi diagram involves pruning the branches that represents insignificant boundary information.

The approach using Voronoi will be used in this thesis. The primary reason for this is that the shape will be obtained sparsely in the form of a point cloud. If the distance transform were to be used, this sparseness would be lost - as the distance transform would need to be computed for the interior of the object. The Voronoi diagram is calculated directly from discrete points sampled along the boundary, and is therefore well suited to applications involving point clouds. The calculation of the Voronoi diagram is done through the qhull library (Barber et al. (1996)), which is widely used for this purpose. It runs in $O(n \log v)$ for 2D and 3D in the worst-case given a balance condition, where n is the number of input points and v is the number of output vertices (processed points).

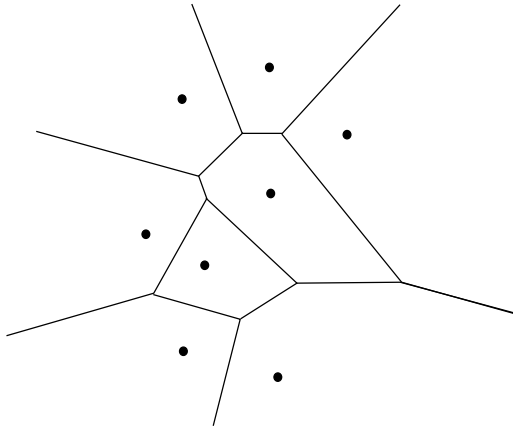


Figure 2.8: Example Voronoi diagram for eight discrete points in R^2

The balance condition imposes restrictions on the number of new facets and partitioned points per processed point. Should the balance condition be broken the algorithm runs in $O(nf_r/r)$, where n is the dimension, f_r maximum number of facets of r vertices, and v still the output vertices. In summary, the algorithm is effective up to 8 dimensions if the balanced condition is upheld. It should still be efficient at 2D, and 3D given that the number of points is relatively small. The conditions can be evaluated statistically.

Comparing the algorithmic complexity with other approaches is difficult in general, as other algorithms not only operates on the boundary - but also the interior. The computational complexity is therefore specified in terms of interior points in addition to the boundary points. The approach is considered algorithmically robust and efficient compared to other methods, however.

A weakness in the Voronoi approach, is that it is sensitive to the sampling density along the boundary. There are two recommended sampling strategies to mitigate this problem. The brute-force approach is to uniformly sample the boundary densely enough to capture the most protruding features. Another approach is to dynamically sample the boundary based on the rate of change along the edge - so-called non-uniform sampling. A sampling theorem exists that places a lower bound on the number of samples in relation to the curvature ([Asada and Brady \(1986\)](#)). In this thesis, the uniform sampling approach will be used, as the boundary of a salmon is relatively smooth. The gain obtained by reducing the amount of samples using the non-uniform approach is therefore limited. The Voronoi diagram has good properties for 2D-shapes, and can be applied directly. It faces difficulties when moving to 3D-shapes, however. That is not an issue, as this thesis will treat 2D cross-sections of the salmon.

There is literature which suggests that shape representation of objects using medial loci has similarities to the cognitive processing performed by the human brain when identifying shapes.

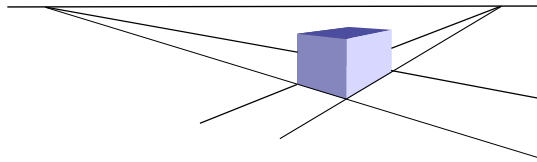


Figure 2.9: The endpoints of lines at infinity forms a line in a 2D projective space, similar to an horizon.

2.4 Geometric Transformations

Camera calibration in general usually deals with the problem of estimating parameters internal to the camera which distorts the output image in some way. A good calibration is important as it can be used to correct for these imperfections, making the image more similar to the actual scene. This is important in a wide range of applications - rigid shape recognition to mention one. Rigid shape recognition is often done by extracting a series of object keypoints, which is then matched against a database of known objects by rigidly transforming the object. If the camera distorts the keypoints, the search for a known object would suffer. Camera calibration is a wide topic, therefore this review section will mostly focus on theory necessary to understand the thought behind choices made and the implementation presented later.

An image represents a projection of the world onto a two-dimensional plane. One must therefore first decide on a coordinate representation that is valid in both spaces. The Euclidean representation is a familiar representation that is well suited to describe lengths, angles and shapes. It does not represent the nature of projective spaces however. Two lines that are parallel in an Euclidean space will remain so for eternity. In a projective space, on the other hand, parallel lines meet at infinity. The Euclidean space \mathbb{R}^n can be extended to a projective space \mathbb{P}^n by utilizing homogeneous coordinates. The Euclidean representation in the plane, $(x, y)^T$, thus becomes $(x, y, 1)^T$. Using homogeneous coordinates, all scalar multiples of these coordinates are equivalent. The Euclidean representation can thus be obtained by normalizing with respect to the z -coordinate. Points at infinity is represented by the coordinate triple $(x, y, 0)^T$, as normalizing this brings the Euclidean representation to infinity. In computer vision, the projective space is just a convenient representation of the real world. The notion of points at infinity is there as limits, not because they actually exist. In two-dimensional projective space, the points at infinity form a line, while in three-dimensional projective space a plane is formed at infinity. To illustrate this, see figure 2.9, where a 2D projective space is shown. Points that meet at the same point at infinity are defined to be parallel. The geometry of the projective plane coupled with a certain line at infinity is known as *affine geometry*, and a transformation relating this geometry to another projective space with a different line at infinity is known as an *affine transformation* (Hartley and Zisserman (2003)). A projective transformation does not preserve angles, distances and ratios of distances - but does preserve straight lines. A straight line will remain straight after a projective transformation. The importance of this is of course that a relationship needs to be defined between the image and world coordinates.




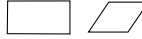
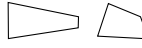
Normally, the process of projecting a 3D world onto a 2D image is modeled by assum-

ing that all light is projected through a single point in space which then intersects a 2D plane - which is the image plane. This model is based on a camera where a ray of light goes through an lens, and onto a film or sensor array. Assuming that all light goes through the center of the lens is a reasonable simplification as the lens is much smaller than the landscape itself. The result is a projection from \mathbb{P}^3 to \mathbb{P}^2 in which a dimension is lost. There is clearly some ambiguity in this transformation, as one cannot reduce the dimension without losing a degree of freedom. In fact, using central projection through a point $(0, 0, 0, 1)$, all points with different final coordinate, the homogeneous coordinate, maps onto the same point. That is, a 3D projective coordinate $(X, Y, Z, T)^T$ maps to $(X, Y, Z)^T$ - dropping the final coordinate. This specific mapping can be represented through a 3 x 4 matrix $P = [\mathbb{I}_{3 \times 3} | \mathbf{0}_3]$. This matrix is known as the camera matrix. Knowing the camera matrix exactly means that any projection of a point in space can be transformed to camera coordinates through the linear mapping $(x, y, w)^T = P_{3 \times 4}(X, Y, Z, T)^T$. In general, the camera matrix can take on any value, and will vary depending on the camera parameters and position of the camera in relation to the world frame. Knowing the full camera matrix is clearly a powerful tool to have. The ambiguity mentioned before stems from the fact that a 2D image cannot reflect the depth in an image. If there are two cameras, both of which has known camera matrices, the ambiguity can be resolved however. This is precisely what is done within stereoscopic imaging, where multiple cameras sees a scene from multiple angles, and generates a range estimate. This is done by computing a relationship between the two camera matrices. The camera matrix P has 11 degrees of freedom, some of which can be specified from the cameras data-sheets.

Homographic Transformations

Although the camera matrix fully describes the projection from world coordinates to image coordinates, it is not needed for all camera projections. Take the imaging of a 2D plane in a camera, for example. Because both the source and destination are in the plane, a mapping relating \mathbb{R}^2 to \mathbb{P}^2 is needed. This leads to a group of transformations known as projective transformations, or homographies. These transformations are presented in table 2.1, and will briefly be introduced. The Euclidean transformation rotates and translates, and preserves the Euclidean metric space while doing so. The notation in the table indicates that the upper left 2×2 matrix must be a valid rotation matrix, namely an orthogonal matrix such that $R^T R = R R^T = I$. The rotation and translation is also performed by similarity transforms, but this transformation can also apply an uniform scaling to the axes. The affine transformation is the first transformation where the upper left 2×2 sub-matrix can vary freely. Using the new degrees of freedom, the affine transform can deform an object. It can be considered a composite transformation consisting of a rotation, scaling, rotation back by the same angle, and finally another rotation $A = R(\theta)R(-\phi)DR(\phi)$, $D = \text{diag}(\lambda_1, \lambda_2)$. This result is obtained through the singular value decomposition (SVD) of A (Hartley and Zisserman (2003)). Compared to the similarity transform, the affine transformation can apply individual scaling to rotated axes, as opposed to an equal scaling applied to the original axes. This transformation preserves parallel lines, scaling of parallel lines, ratio of areas. Finally, the transformation with the most degrees of freedom in this group of transformations is the projective transformation. This transformation applies a general linear transformation and a translation. The most important invariant

Table 2.1: Homographic transformations.

Group	Matrix	# DoF	Distortion	Invariant properties
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	2		Length, area, orientation
Euclidean	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$	3		Length, area
Similarity	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$	4		Shape
Affine	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$	6		Parallelism, ratio of areas
Projective	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$	8		Collinearity

property for these purposes of this thesis is that of collinearity. The other invariant properties of the projective transformation requires more knowledge of the geometry of the projective space than the scope of this text. Collinearity is the property that points that can be joined by a single line, remains so under the transformation. Basically, it means that straight lines remains straight - but not necessarily in relation to other lines.

A projective transformation between two planes can be determined uniquely from four point correspondences, with no three collinear on either plane. This stems from the fact that there are eight degrees of freedom, and each point correspondence has two components.

Chapter 3

Equipment and Acquisition

This chapter takes the reader through the equipment used, and the steps necessary to transform images into a 3D point cloud. First, the general principles behind line scanning is introduced. The equipment and its configuration is then outlined. The calibration routine to find the transformation from pixels in the images taken by the camera to real-world coordinates is then described. Lastly the on-line processing, which is executed on a graphical processing unit (GPU), is described conceptually. The post-processing applied to the raw coordinate data is covered in [chapter 4](#).

3.1 Line Scanner Principles

An illustration a basic line-scanner is shown in [Figure 3.1](#). A laser that emits a line, known as a sheet of light laser, is used to project a laser-line on the contours of the object. A camera positioned at an angle is used to take an image of the laser-line. The angle is necessary to give the camera perspective of the elevation, and thus a change in height of the laser is reflected in a change in the image. The camera does not need much perspective to be effective, and an angle of about 15 degrees was deemed to be sufficient for this project. Designing with a greater angle increases the resolution by utilizing more pixels to represent the same height. There is always a trade off between increased resolution and computational load, as more pixels means more data to be processed. As this is the first iteration of this type of equipment, the goals set were conservative - as the computational load was unknown.

Each image taken by the camera corresponds to a collection of image coordinates in the x-y plane at a discrete position along the z-axis. Scanning an object therefore involves taking multiple such images and merging them to form the profile along the z-axis, which is the axis along the direction of movement. The transformation from image coordinates to world coordinates is done through a mathematical transformation which scales, translates and rotates the image coordinates correctly. This transformation is determined through a calibration routine, which is covered later.

The goal in building the camera rig is not to beat the commercial systems in terms of accuracy, but to compete on price and flexibility. Additionally, the primary application

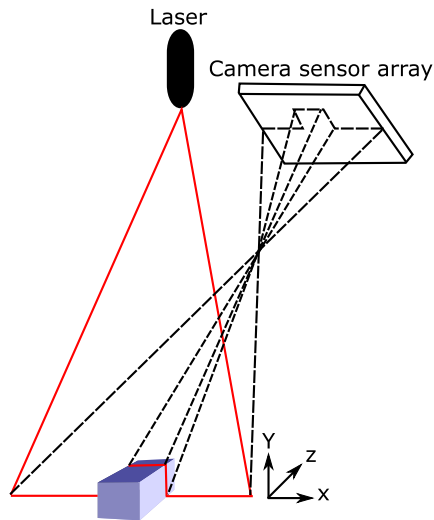


Figure 3.1: The basic concept of a line-scanner using a sheet of light laser and camera.

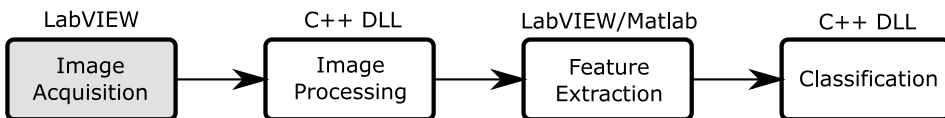


Figure 3.2: Overall steps of the procedure.

area is for fast-moving objects. This places further constraints on which commercial systems can be utilized due to the required frame-rate. The resolution of the scan along the direction of movement z is determined by the acquisition speed of the system in relation to the rate of movement of the object. Having a relatively uniform resolution along all dimensions are normally preferable for most applications. The goal was to obtain a better resolution than 2 mm per pixel along all dimensions.

The desired resolution in the xy -plane can be obtained by changing camera parameters like the resolution, angle and distance from the object. An increase in the resolution along the z -axis is directly correlated with how fast the system can process the frames, assuming that the frame-rate in the cameras is not the limiting factor. The resolution along the y axis in the image plane is hard to quantify as the first moment of intensity along the cross-section of the laser-line is utilized to obtain sub-pixel accuracy for the height profile. More details follows under section 3.5.

3.2 The Camera Rig

The cameras used are manufactured by Point Grey Research, the model is GS3-U3-23S6C-C. The product sheet states that this camera can run at 162 frames per second (FPS) at full resolution (1920x1200). This FPS increases when a smaller region of interest is specified,

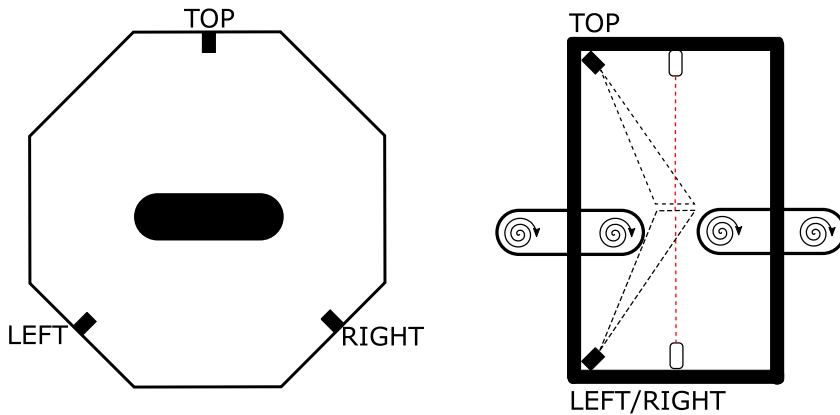


Figure 3.3: Conceptual sketch of the camera setup. Left illustration is as seen from the front, and the right is seen from the side.

and for this project the camera hardware can output well over 1000 FPS at the image sizes used. The camera hardware is therefore not a limiting factor in this respect. Additionally, the type of sensor used (Sony IMX174 CMOS) has very good noise rejection properties, even when the exposure time is set very low.

The camera rig consists of three cameras and three lasers placed to cover an object moving on the conveyor belt from all angles. A conceptual sketch of the layout is shown in figure 3.3. The reason for opting for three cameras is not just the theoretical full coverage of the object, but also out of necessity. Placing a camera directly beneath the conveyor belt would pose problems in a practical situation since substances would be more likely to hit the camera when falling down.

Each laser is placed perpendicular to the gap in the conveyor belts, while the cameras are placed at an angle. The cameras are not spaced exactly 120 degrees however, as salmon are roughly elliptical perpendicular to the direction of travel. Since the top camera covers one entire side, the bottom cameras can be closer together as long as they still manage to capture the sides of the object. To obtain better images from the bottom side of the salmon, the two cameras on the bottom are closer together - around 90 degrees.

In addition to the lasers, the rig is equipped with a LED strip around the gap in the conveyor belts. The LEDs and lasers are triggered every other frame. When color images are obtained, the LEDs are on to provide enough illumination. A photo of the rig with the LED strip active can be seen in figure 3.4. The LEDs are turned off again when an image is taken of the laser-line to increase the contrast of the laser against the background. This enables the system to keep the cameras running at a much lower exposure time than what would be possible when not turning them off and on, thus increasing the frame rate. The exposure time used for the experiment is set at 300 μ s.

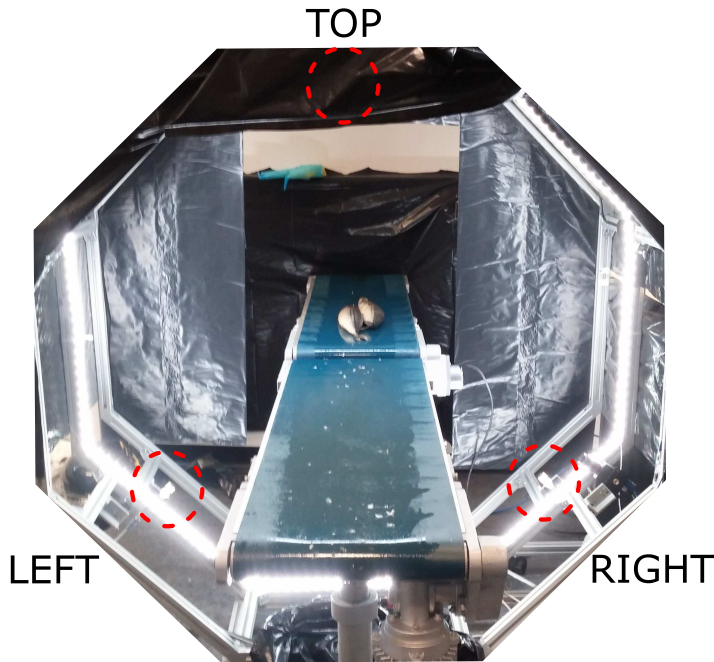


Figure 3.4: Photo of the camera rig whilst scanning two mackerel for a related experiment.

3.3 Camera Triggering

Explicit camera triggering is necessary for several reasons. The first reason being that the conveyor belt might operate at variable speeds in industrial use. This is not by itself enough to justify the implementation for this thesis alone, as that modification could be added at a later point. The second reason, which is more substantial, is the fact that the separate cameras has overlapping views and overlapping laser-lines. As there is a laser projected from the direction of each camera, depending on the size of the object, the laser lines might have significant overlap. Herein lies the problem, as the amount of overlap varies, and is thus hard to correct for. Properties of the laser light reflection contains valuable information about the surface of the object, and will be used to detect wounds. This is possible due to different scattering of light in fleshy areas. A non-uniform intensity across the laser line complicates the analysis using these features greatly, if not makes it impossible. The third reason is a potential decrease in accuracy of the coordinates, as a double laser line can saturate the camera pixels in the region of overlap - thus potentially reducing the accuracy when calculating the mid-point of the laser. The effect of this is untested, but avoiding it completely is wise as the problem varies with laser intensity, exposure and camera gain. Additionally, if the lines are not completely overlapped, the center of the laser line will be shifted from its location with respect to the true laser line.

The triggering is implemented on an Arduino Uno microcontroller board, which features an Atmel ATmega328 chip. The board has 14 digital I/O pins, each of which outputs 5 V at a maximum of 4 mA. Each camera has an optical isolated input circuit designated

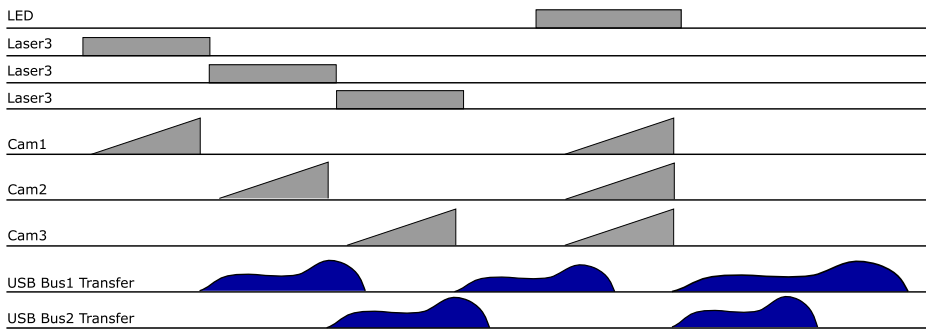


Figure 3.5: Conceptual sketch of the triggering sequence. Not to scale.

for triggering signals. Each triggering input is connected to a separate digital output on the Arduino. An additional three pins are connected to each laser to be able to turn them on slightly before the triggering signal is sent, as there might be some delay in turning them on. Two LED strips, which covers a circle around the conveyor belt, are triggered by a single output connected to a transistor per strip. This is needed because the digital outputs of the Arduino are not capable of supplying enough current. The LED strips are supplied externally by a 20V DC source, with a current of about 12 mA. The lasers are powered directly from the Arduino digital outputs.

Triggering sequence

Figure 3.5 shows a sketch of the triggering sequence. The sequence itself is quite basic, the full code is therefore included in full in appendix B. Each sequence captures one LED image and one laser image per camera. The frame rates mentioned below therefore needs to be halved to get the actual frame rate of the output coordinates. The laser frames are obtained by sequentially enabling one of the lasers and cameras, thus avoiding the issues with overlap. After the last camera has captured its laser image, all three cameras take color image simultaneously with the LEDs enabled. The hard part is not implementing the sequence itself, but getting the timing right, as the cameras spend an indeterminate amount of time transferring the images. Time fluctuations can occur due to scheduling on the computer, as a standard Windows 8 operating system is used in place of a real-time system. The process is however set to higher than normal priority, to mitigate this effect. If a triggering signal is sent before the camera is ready, the camera takes a new image as soon as it is able. This makes tuning of the timing a matter of trial and error, while looking at the output frame rates.

The resolution used for the experiment is set at 1280x192 pixels. The height was originally intended to be smaller, but was increased due to changes to the calibration object. The calibration object used a larger space than intended, it was increased rather than re-designing the object itself. In a future revision, a height of 128, or even 96, is obtainable. This is important, as the width of the conveyor belt is expected to increase when applied industrially. With 400 frames per second, each camera uses $400 \cdot 1280 \cdot 192 = 98.3 \text{ MB s}^{-1}$ of continuous bandwidth to transfer the images. This is more than sufficient seeing as the

USB3 bus is theoretically rated at an effective bandwidth of 500 MB s^{-1} . A more conservative estimate is 450 MB s^{-1} due to overhead, although the peak bandwidth can be significantly lower at times. The cameras are connected to two distinct USB3 buses, one internal bus and one external PCIe 2.0 bus through an expansion card. Camera 1 and 3 therefore share the same bus to minimize the reduction in obtainable frame rates due to the sequential capture. The two cameras on the same bus are additionally left idle when the other camera is taking a laser image due to overlap. By experimentally pushing the frame rate as high as possible, nearly 600 frames per second was obtained before dropped frames occurred. An estimate of the required bandwidth in that instance is $600 \cdot 1280 \cdot 192 \cdot 2.5 = 369 \text{ MB s}^{-1}$. Using a second expansion card could potentially increase the frame-rate further due to a higher peak bandwidth available per camera. The frame rate for the experiment was still set at 400 frames per second, as that is both safely below the limit and provides more than enough resolution along the z-axis. At a reasonable speed for the conveyor belt in the lab, the resolution along the z-axis was measured to be 0.953 mm with 400 FPS, which is more than sufficient.

Image Embedding

As mentioned, every other frame received by the computer is a LED and laser frame. To determine which is which, one has several options. The naive approach is to determine what the first frame is by using the average intensity, and then assume that the frames appear in alternating order. The reason for this being naive is that it is possible that a frame is missed at some point, due to noise or other errors. Checking the intensity of every frame introduces unnecessary computations and is not an option. The camera model used supports embedding certain information, like timestamps, into the first bytes of the image. It also has a few general purpose inputs in addition to the triggering input. The state of this input can therefore be embedded into the image data. By connecting the power signal of the lasers to the cameras in addition to the lasers themselves, the on/off status of the laser is directly embedded in the image. This binary flag can then be extracted from the image. After this state has been read, the bytes are set to zero to avoid interfering with further processing.

3.4 Calibration Routine

To relate the extracted laser-line image coordinates to world coordinates, the cameras must be calibrated. General calibration routines for 3D geometry normally utilize a plane with a checkered pattern or similar, which is imaged multiple times - often at different angles by rotating the object. This yields enough information to not only estimate the internal parameters of the camera (intrinsic parameters) and parameters of camera position in relation to the world frame (extrinsic parameters). Radial distortion is an effect that occurs due to optics, which causes straight lines to be increasingly bent out of shape towards the sides of the image. This effect will not be considered in this thesis, but can be added to the developed routine in further work. The lenses utilized in this thesis has been found to have little radial distortion, and the center of the image is utilized - minimizing this effect.

The calibration routine has been developed under the following simplifying assumptions. The world coordinate XY -plane is chosen to be perpendicular to the direction of travel along the conveyor belt. The laser light is assumed to be projected along this line onto the object in question. The laser-line in world coordinates therefore does not utilize the third dimension along the Z -axis, which is described only by the traveled distance in between two captured frames. The world coordinate plane can therefore be assumed to not use three dimensions, but only two. This simplifies the calibration greatly, as one can look for a transformation relating 2D points to 2D points. That is, we are looking for a transformation that expresses the relationship from image coordinates to world coordinates, $X_w = HX_c$. To solve for the transformation matrix, H , one must know points in both the world frame and image frame. This is where a calibration object comes in.

The calibration object, seen in figure 3.6, is used to relate the world coordinates to image coordinates. The laser light is projected on to the jagged edges of the object. This is then perceived by the camera, which can be used to extract the corresponding points in the image plane. The peaks and bottoms of the spikes are used as coordinates. The position of these extremities has been done quite robustly in the image plane by dividing the image into sections with straight laser lines, which is then fitted to a line using least squares regression. The intersect points of these lines are then found, which is used as the corresponding point in the image plane to the known coordinate in the world plane. This also gives the additional benefit of sub-pixel accuracy, as the intersect point is calculated as a floating point number instead of an integer pixel location. An image taken of the laser-line on the calibration object can be seen in figure 3.8, which has been inverted and re-colored to remove the blackness of the background. Originally, the intent was to use the stair-formed shape on the underside of the object for calibration of the bottom cameras. During the early work on this thesis, it became clear that the short edges was too short for the interpolation routine to yield good results. Since it was established that the top of the calibration object worked quite well to calibrate the top camera, a stand was 3D-printed so that the calibration object could be flipped upside-down towards the camera in question. This stand can be seen in figure 3.7, along with the calibration object in the position used to calibrate the lower left camera.

Another important aspect of designing the calibration object, is that collinearity is preserved through an projective transformation. This means, that to provide new information between the world and camera frames - no more than two points should be collinear. This is yet another reason the bottom stair-shape of the calibration object would not work well to determine the projective transformation, as the points are relatively collinear. The top side of the calibration object, has been designed with this in mind however.

The transformation that needs to be found is a projective transformation with 8 degrees of freedom. An affine transformation, where h_{31} and h_{32} is set to zero could also be used. An affine transformation preserves parallelism however, and can therefore not be used to describe the pinhole projection in a pinhole camera model. We need projective geometry to represent such transformations. Another way to look at this is that an affine transformation preserves the ratio of areas throughout the image. Thinking of this intuitively, the affine transformation will not be able to correctly transform these areas from the camera view to the world coordinates. This is due to the perspective of the camera. Areas closer to the camera will for this reason appear larger than those further away. Apply-

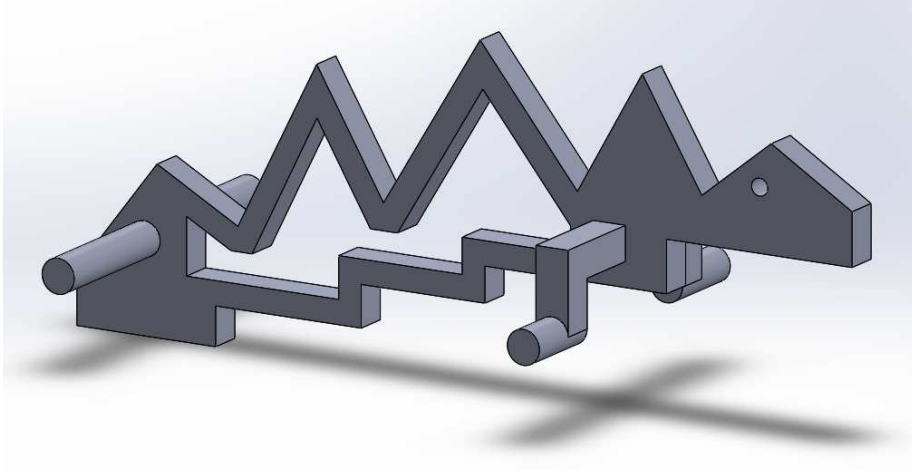


Figure 3.6: The calibration object designed to calibrate all three cameras.

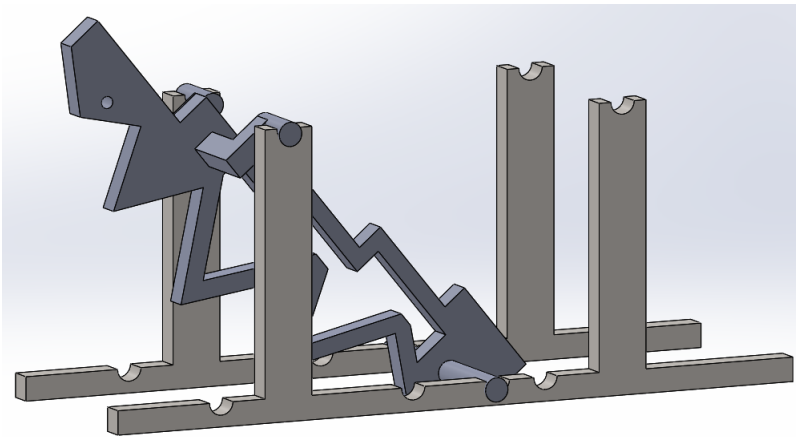


Figure 3.7: The stand used to calibrate the bottom cameras. The stand includes holes for all cameras, to improve on misalignment issues that would occur by simply placing the calibration object manually.



Figure 3.8: An laser image taken of the calibration object as seen when calibrating the top camera. The image has been inverted to avoid the black background.

ing an affine transformation to these areas will scale the areas by an equal amount thus preserving the perceived difference in size. As the name indicates, the projective transformation is not afflicted by this, and must be used. During the prototyping stage, the affine transformation was tested, and was able to obtain decent results from the top camera. The two bottom cameras, which are placed at an angle, yielded worse results as expected due to the increased perspective (or depth) in the image.

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$$

The calibration object yields 7 point correspondences, which in turn yields 14 unique variables since each point has an X and Y coordinate. Since we are solving for 8 variables in the transformation matrix, the problem is overdetermined. One must therefore use methods in numerical optimization to find the transformation that minimizes the point correspondences over a chosen metric. The algorithms of the numerical optimization routine will be presented. It has two main steps. First, a linear system of equations is solved for an initial transformation matrix. This matrix is then used as an initialization in a nonlinear solver that further reduces the error by minimizing a cost function.

3.4.1 Direct Linear Transformation (DLT)

Assuming that we have exactly four points, the transformation matrix H can be solved for exactly by the following linear procedure, known as the direct linear transformation. First, the matrix H is written as a vector by specifying its row vectors as \mathbf{h}_j . Note that i denotes the point correspondence, while c and w denotes the camera and world coordinates, respectively. $\mathbf{X}_{c,i}$ denotes the full vector for point number i , while its individual components is written as $(x_{c,i}, y_{c,i}, w_{c,i})^T$.

$$\mathbf{H}\mathbf{X}_c = \begin{pmatrix} \mathbf{h}_1^T \mathbf{X}_{c,i} \\ \mathbf{h}_2^T \mathbf{X}_{c,i} \\ \mathbf{h}_3^T \mathbf{X}_{c,i} \end{pmatrix}$$

Writing the cross product of the world coordinates $\mathbf{X}_{w,i} = (x_{w,i}, y_{w,i}, w_{w,i})^T$ and the vector-notation above gives.

$$\mathbf{X}_w \times \mathbf{H}\mathbf{X}_{c,i} = \begin{pmatrix} y_{w,i} \mathbf{h}_3^T \mathbf{X}_{c,i} - w_{w,i} \mathbf{h}_2^T \mathbf{X}_{c,i} \\ w_{w,i} \mathbf{h}_1^T \mathbf{X}_{c,i} - x_{w,i} \mathbf{h}_3^T \mathbf{X}_{c,i} \\ x_{w,i} \mathbf{h}_2^T \mathbf{X}_{c,i} - y_{w,i} \mathbf{h}_1^T \mathbf{X}_{c,i} \end{pmatrix} = 0$$

This cross-product is clearly equal to zero as they are linearly dependent, and can be set equal to zero. This expression can now be rewritten as below, by considering that $\mathbf{h}_j^T \mathbf{X}_{c,i} = \mathbf{X}_{c,i}^T \mathbf{h}_j$.

$$\begin{pmatrix} \mathbf{0}^T & -w_{w,i} \mathbf{X}_{c,i}^T & y_{w,i} \mathbf{X}_{c,i}^T \\ w_{w,i} \mathbf{X}_{c,i}^T & \mathbf{0}^T & -x_{w,i} \mathbf{X}_{c,i}^T \\ -y_{w,i} \mathbf{X}_{c,i}^T & x_{w,i} \mathbf{X}_{c,i}^T & \mathbf{0}^T \end{pmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0. \quad (3.1)$$

This final expression has the form $\mathbf{A}_i \mathbf{h} = \mathbf{0}$ where each submatrix of \mathbf{A}_i is equal to the 3×9 matrix above when $\mathbf{X}_{c,i}$ is expanded. The sub-matrices \mathbf{h}_j is also expanded to form a vector with 9 elements, one for each element in the transformation matrix. This matrix is linear in the elements of \mathbf{H} , and can thus be solved using linear algebra. In reality, the final row of equation 3.1 is not linearly independent, and is removed as it does not provide any new information. Each point correspondence therefore gives an 2×9 matrix. Using the concatenated set of equations from four points, this can be solved explicitly as an 8×9 linear system. The problem at hand is however, overdetermined. Although given no noise, this could still be solved explicitly.

The overdetermined DLT problem is normally solved by minimizing the norm $\|\mathbf{A}\mathbf{h}\|$, subject to the constraint $\|\mathbf{h}\| = 1$. The solution to this has been found to be the unit eigenvector of $\mathbf{A}^T \mathbf{A}$ with the smallest eigenvalue. This can be found directly from the singular value decomposition (SVD) of \mathbf{A} ($\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$), by selecting the column of \mathbf{V} that corresponds to the smallest singular value. The elements of that column can then be rearranged to form the eight elements of the transformation matrix \mathbf{H} . DLT minimizes the algebraic error, which may not be the optimal solution with respect to geometric error. DLT can be computed quickly due to the linear algebra and gives fairly good results, and is for that reason often used as a initial step before improving the geometric distance through nonlinear optimization.

Normalization

Before moving on to the next step in the calibration routine, normalization must be discussed. Algorithms that minimize geometric errors, are invariant to scaling which can be represented through similarity transforms. The DLT algorithm, which bases itself on algebraic minimization, is not. Therefore, before solving the system of equations in 3.1, a normalizing transformation must be applied to the set of coordinates. After the transformation matrix \mathbf{H} has been computed, it must be de-normalized to return to the original space. The procedure below is applied to the world coordinates and image coordinates individually, apart from DLT itself.

1. Translate coordinates so that the centroid (mean) is at the origin.
2. Scale points so that the average distance from the origin is $\sqrt{2}$.
3. Scale coordinates individually using the similarity transform obtained from the two previous steps $\mathbf{T} = \mathbf{T}_{scale} \mathbf{T}_{translate}$
4. Calculate DLT
5. De-normalize \mathbf{H} by calculating $\mathbf{H} = \mathbf{T}_w^{-1}(\mathbf{H}\mathbf{T}_c)$, where \mathbf{T}_w and \mathbf{T}_c is the similarity transformation from before for world and camera coordinates, respectively.

3.4.2 Non-linear Optimization

Using the result from DLT as an initial starting point, a non-linear solver can be applied to improve the result by minimizing some geometric error cost. In this thesis, we are strictly

interested in the transformation from image coordinates to world coordinates. The natural choice is to minimize the one-directional transformation error cost $\sum_i d(X_{w,i}, \hat{X}_{w,i})^2$.

Where d is the Euclidean distance norm and $\hat{X}_{w,i}$ are the coordinates obtained through the projective transformation below.

$$\hat{\mathbf{X}}'_{w,i} = \mathbf{H}\mathbf{X}_{c,i}$$

Which is then projected onto a 2D-plane by normalizing with respect to w -element.

$$\hat{\mathbf{X}}_{w,i} = \begin{pmatrix} \mathbf{x}'_{w,i}/\mathbf{w}'_{w,i} \\ \mathbf{y}'_{w,i}/\mathbf{w}'_{w,i} \\ 1 \end{pmatrix} \quad (3.2)$$

The one-directional cost is minimized by computing the world coordinates as above using the Levenberg-Marquardt algorithm, which is an iterative non-linear solver for minimization of a sum of squares. It utilizes a combination of the Gauss-Newton method and steepest descent. Any non-linear solver should be able to minimize the problem however, provided that DLT provides an adequate starting-point.

3.4.3 Misalignment Correction

As mentioned previously, the original intent was to calibrate the bottom cameras using the bottom of the calibration object. Part of the idea with this approach was that the top and one bottom camera could be calibrated simultaneously, thus reducing the misalignment. Possibly also using the top camera calibration twice to relate the calibration of the two bottom cameras. Unfortunately, this was not possible as the approach was scrapped. Instead, a routine was developed where a circular object is imaged in the laser after the initial calibration. The coordinates from each camera individually, can then be fitted to a circle. The center of the two bottom circles is then translated to the centroid of the circle fitted to the top camera. The assumption here, is that the top camera has the most correct calibration, due to having the least perspective distortion. The amount of misalignment was found to be minor however, as the coordinates mapped quite well to the same coordinate system. An image of this procedure can be seen in figure 3.9. Note that this example has slightly worse misalignment on one of the cameras than what is typical, but it illustrates the procedure better than almost perfectly overlapping points. Typically, the misalignment between the bottom cameras and the top camera is in the range of 0.01-0.05 centimeters. Since the calibration routine involves moving the calibration object to a new position in the stand, errors can easily be introduced if not careful. This is one of the drawbacks of the current calibration routine. This can be improved on by having a fixed stand, instead of the portable stand used for laboratory tests.

3.4.4 Speed Calibration

For this thesis, a conveyor belt of constant speed was used. The speed of the conveyor belt was calibrated by scanning an object of known length. That length was divided by the number of frames taken of the object to obtain the spacing between each frame. The

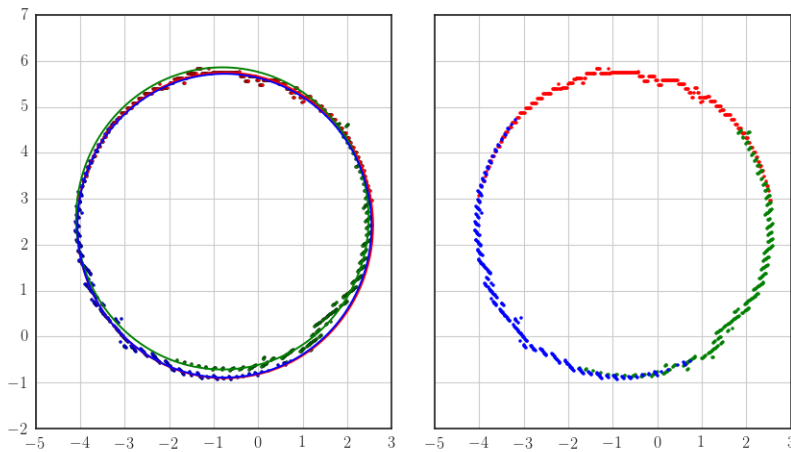


Figure 3.9: The misalignment correction is performed by fitting each camera to a circle and translating to match. Note that the misalignment of the right camera (in green) is worse than typical. The axes are specified in centimeters.

calculated value was found to be 0.0953 mm between each frame. Industrially, this will be replaced by a rotary encoder which passes the speed or distance traveled to the triggering system, to ensure that the frames is taken at equal distance should the conveyor belt speed vary. While the speed of the conveyor belt is supposed to run at a constant speed, it was stopped and started between each scan - and might have a transient starting phase. Additionally, the band might slip if the motors accelerate to quickly. The salmon was put as far back on the conveyor belt as possible to attempt to mitigate this. If there is an effect of this, it has not been large enough to be visible.

3.5 Parallel Image Processing

The resolution along the z-axis is directly linked to how fast the system can capture and process the images from all three cameras. In controlled experiments, the speed of the conveyor belt can be set lower to adjust the resolution as needed without changing acquisition speed. The results of this project is very much intended to be put into practice on a real processing line, where this is not an option. Parts of this project might also be re-used for imaging of falling objects. In an effort to press the resolution as high as possible for objects at high speeds, the parallel processing power of a graphical processing unit (GPU) has been utilized. Not only would a CPU not be sufficient, even with all CPU cores working in parallel, it would also steal processing power from other tasks - like analysis of the data.

Programming for a GPU requires a slightly different mindset than that of conventional code. This section will introduce the reader to some concepts within GPU-programming. The code will not be detailed in depth, but all the steps will be explained conceptually. The implementation itself has been written using C++ AMP, which is a fairly high-level

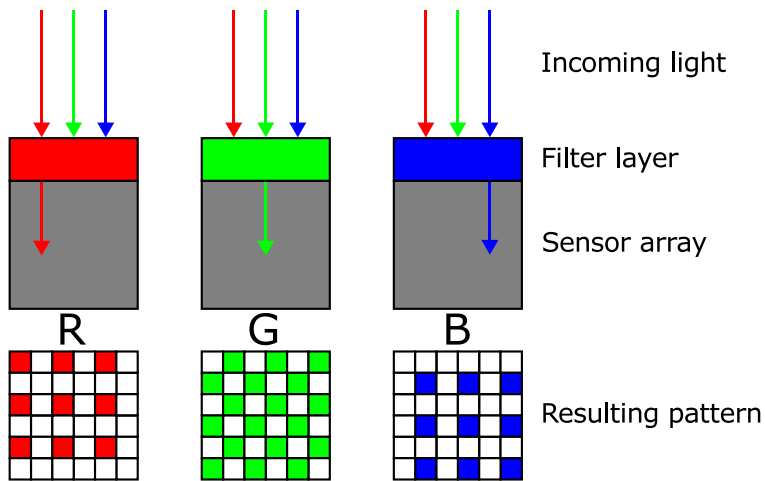


Figure 3.10: Illustration of Bayer pattern (RG) in the sensor array of a camera (adapted from Wikipedia (2006)).

GPU interface.

3.5.1 Bayer Filtering

When the work on this thesis started, some GPU code was already implemented by the author and a colleague (Andreas Ulvøen). It took monochrome (greyscale) images from the camera, extracted a laser-line and transformed the image coordinates to world coordinates. To enable the extraction of color images in addition to laser images, the monochrome mode could not be used. Switching between the two modes every other frame would not be feasible. Color images in cameras are normally captured using an array sensors, where different sensors capture each of the color planes. This can be done by using the same light-sensitive sensor, and filtering out the unwanted colors for each sensors. In RGB cameras, green is normally overrepresented to mimic the physiology of the human retina - which is more sensitive to green light. There are many arrangements of these color pixels possible. A common arrangement is the Bayer filter - seen in figure 3.10.

To obtain color images in full resolution, various forms of interpolation can be used to calculate the color values not obtained by that sensor. Cameras normally support this operation in hardware on the camera itself using field-programmable gate arrays (FPGA). This is convenient, as it frees the computer from performing the operation. Unfortunately, this mode requires three times as much bandwidth due to the three color planes present at each pixel. The result of an increase in the image size results in a reduction of attainable frame rate, as the bus shared by the two cameras were already pushing the limit. For this reason, a raw mode was instead used - where the intensity of each sensor is transferred. This results in additional work for the computer, but can be accelerated by a GPU to make this negligible.

As mentioned, to obtain a color image in full resolution - interpolation must be per-

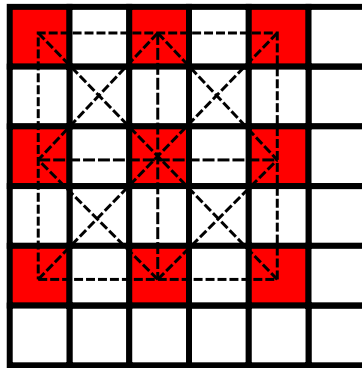


Figure 3.11: The bilinear interpolation pattern used. Two-way interpolation is used on the red grid, and four-way (bilinear) interpolation is used in the cross-section.

formed. This is known as debayering or demosaicing. The simplest, and fastest, method is the nearest-neighbor method - which sets the missing color planes equal to an adjacent pixel. While this could produce acceptable results, it would mean that every other red or green column is an copy of the one before it (or interleaved from two columns). The laser-line dominates the red spectrum, and could therefore be affected by this in terms of accuracy. The next method on the complexity-ladder is interpolation, which is greatly simplified when applied to elements of a rectangular equidistant grid, because the average can be used in place of a weighted interpolation. The interpolation pattern used for the red spectrum can be seen in figure 3.11. The pixels that fall on the rectangular grid of the red pixels, are simply interpolated from the two closest neighbors. The pixels in the cross-sections are calculated using four-way bilinear interpolation. The bottom row and rightmost column are set using nearest-neighbor. More advanced methods available that uses gradients, splines and bi-cubic interpolation to preserve more smoothness or finer details in an image. For this application the bilinear interpolation is sufficient however, as the more advanced methods for the most part only improve the image in terms of aesthetics.

The laser frames are sent to the GPU to calculate the coordinates, among other things. For this reason debayering of the red color plane is implemented on the GPU. This both offloads much calculations from the CPU, and reduces the amount of data needed to be transferred to the GPU. The color images are debayered on the CPU, which will be justified in section 3.5.5.

3.5.2 Coordinate Extraction

Once the red spectrum of the images has been debayered, the extraction of the laser-line can commence. The rotation of the cameras about their axis has been done so that the laser-line is as horizontal as possible in the images, that is - perpendicular to the direction of travel. The extraction of the laser-line coordinates for this reason performed on a per-column basis in the image. Doing this not only makes sense because of the aligned laser-line, but also provides an excellent divisible unit on which the GPU can operate in parallel. A single thread iterates over all the pixels in a column. It then calculates the

weighted average for the intensities and the pixel positions, ignoring any pixels below a certain threshold. The threshold is necessary to remove background noise, and some of the unwanted reflections. In this thesis, a threshold of 14 was utilized (where the maximum value is 255). Adjusting this would however depend on the camera sensor noise and the gain applied by the camera to the raw sensor values. In addition to this per-pixel threshold, a column is required to have at least a certain pixels over the threshold to be used in a coordinate calculation. This is done to avoid calculating a coordinate if a single pixel is above the threshold. The multiplier is set at 5 in this thesis. The weighted average on the pixels above the threshold is calculated according to equation 3.3, where w_i is the intensity of a pixel and y_i is the index.

$$\bar{x} = \frac{\sum_{i=1}^n w_i y_i}{\sum_{i=1}^n w_i} \quad (3.3)$$

The result is the mean of the pixel indices, weighted by its intensity. This is then used as the y-coordinate in the image plane. The x-coordinate is chosen as the index of the column. Note that this yields a sub-pixel accuracy for the y-coordinate, unless all of the intensity pixels are saturated. Now that an X and Y coordinate has been obtained for the image plane, they can be transformed to world coordinates through the projective transformation matrix found in the calibration, $X_w = HX_c$. This matrix product, along with the division required to normalize with respect to the last element is performed to obtain the coordinates.

3.5.3 Reflectance Properties

The spatial coordinates have now been extracted from the laser image. There is, however, more information contained in the raw image. How the laser light reflects or scatters when it hits the surface, can provide valuable insight into the reflective qualities of the surface. Three descriptors has been extracted per column which each convey some aspect of the reflective properties.

1. **Offset:** The raw intensity value in the image a specified number of pixels from the weighted mean. This provides information of how strong the laser light is on the side of the actual peak of intensity.
2. **Total:** The sum of all pixel intensities above the threshold.
3. **Beamwidths:** The width returned is the smallest intensity width possible, in which a user-specified percentage of the total intensity lie within. This yields a measure of how centered the laser beam is at its region of highest intensity. Much of the same information as the offset intensity is obtained from this, but more robustly.

These properties can be used in the detection of wounds, as the reflective properties of the surface changes in the presence of a wound, open or not. Basically, any lack of the regular scaled surface will yield a different scattering of light. More discussion regarding these properties follow in the section about wound detection.

3.5.4 GPU Code Optimization

In GPU code, there are certain aspects that affect the performance of the program more than on their CPU counterparts. To write an effective GPU program, these must be considered. While the implementation details is not specifically covered in this thesis, these aspects has been considered thoroughly to minimize the computational requirements of the GPU processing.

Global Memory Access

A GPU has multiple memory layers. The top layer is the global memory layer, which is typically large - but the slowest to read and write to and from. One of the primary causes of performance degradation in GPU code is poorly planned global memory access. If a piece of data is needed once throughout the calculations, one cannot optimize the memory access. If the very same element is used multiple times, however, one can cache the value in a faster, more local, memory to avoid reading the same value from the slowest memory. Reading local memory can be read in as little as four clock cycles, while a global memory access can take as much as 400-600 cycles (Miller and Gregory (2012)). When this occurs often, it starts to add up. Within the group of local memory, there is a type of memory named shared memory. This is a type of memory which can be shared across threads. This has specifically been utilized in the debayer implementation, where each red pixel is read eight times during the interpolation of its neighboring pixels, and once for the pixel location. The size of the shared memory is of course limited, and the image is therefore divided into units of threads, called tiles, which cooperate on a certain region of the image at a given time.

Coalesced Memory Access

In addition to reducing the total number of global memory accesses, one must also consider if the reads or writes are coalesced or not. Reads from memory is typically performed in blocks or chunks of a size determined by the architecture, a coalesced read is a read where consecutive elements are read simultaneously. This is familiar to those proficient in programming CPUs, as one typically tries to read an array row-wise because it is stored that way in memory. The same consideration is necessary on the GPU, but becomes slightly more involved when multiple threads reads memory. A specific example from this is the column-wise division of the processing when extracting the laser line coordinates. One might think that since each thread reads elements from a column, this principle is violated. It turns out, that this is actually not the case, because multiple threads are executing the code simultaneously. The threads iterate through the code in a synchronized manner. When a thread reads a pixel in its column, so does the neighboring threads. The compiler is smart enough to realize that these reads are being done from the same region in memory, and utilizes the same read for multiple threads. Iterating column-wise is therefore the optimal pattern for this case. Of course, this assumes that the indexing is performed equally for each thread.

Divergent Code

To continue from the last point, there is another aspect that is important. GPUs are different from CPUs in the way they handle instructions. Conditional statements, like the if sentence, introduces branching in the instructions. A GPU typically executes the same instruction across on multiple data (SIMD), as the hardware is optimized for this pattern. Specifically, the logic that handles instructions are typically shared across a certain number of threads. This means that if one thread branches, all other threads sharing the same instruction unit must wait for the instructions in that branch to complete before continuing. Reducing the use of divergent code means that less threads are left idle.

Performance

By using the raw images from a salmon obtained in the experiment, the performance of the GPU library can be stated by running the frames through as fast as possible. Using a salmon consisting of 848 frames, from three cameras, the processing time took 0.76 s including transfer to and from the GPU. By extrapolating from this result, the library should be capable of well above 1000 frames per second from each camera, which is just the frames containing coordinates. The cameras must run at double that to output coordinates at that rate. By analyzing the GPU workload, it can be concluded that most of the time is spent transferring data over the PCI-e bus. The images are sent to the GPU in batches of 16 frames from each camera. One such batch takes 6.6 ms on average. Of this time, only 0.2 ms was spent processing. The rest of the time was spent on data transfer and overhead. This means that more advanced computations can be put on the GPU without affecting performance much, as long as the amount of data does not increase. The above timings was obtained on a NVIDIA GTX 970 graphics card. The graphics card on the camera rig is AMD R295x2, which should have better performance.

3.5.5 Color Images

Every other frame taken is one where the laser is turned off, and the LED strips surrounding the conveyor belt is turned on. This is referred to the color image. The laser-frames are also color images, but since the ambient light is turned off - only the laser-light appears in relative darkness. The problem at hand is to somehow extract the color of the salmon and imbue the spatial information with color information. A method where the spatial information and color is not linked, is by simply taking one row from each color frame and concatenate them to form the full image. Unfortunately, this means that the perspective of the camera is preserved. While not problematic from the top camera, the bottom cameras will have a skewed perspective. This becomes problematic for the image processing related to wound detection. Due to the overlap between the bottom cameras, it becomes hard to draw a line between the cameras to avoid processing the same region twice. Additionally, the perspective causes the areas far away to become smaller. Instead of doing the simple merging of images, one instead can assume that the closest pixel to the weighted center of the laser-line does not change much from the laser-frame to the color-frame. By picking out the same pixel in the color frame, one has the approximate color of the spatial coordinate found in the previous frame. One can even compensate for the shift

from the laser-frame to the color-frame by interpolating between two laser-frames. This has not been done in this thesis, as the frame-rate is quite high in relation to the speed of the conveyor-belt, which minimizes this effect. If the coordinates become more sparse, this might become necessary.

When picking out the corresponding pixel in the next color frame, there is still the issue of debayer filtering to be resolved. This frame is naturally also in the raw format, except now all three colors are needed instead of just red. At this stage, one could either bayer-filter the entire color image, for example by utilizing the GPU again. Debayering the full color image on the CPU would not yield the desired performance. What was done instead, was to debayer on the CPU - but only the specific pixels needed. Since the GPU processes the image column-wise, and the image has dimensions 1280x192, one only need to debayer 1280 pixels. This is a massive reduction. The performance by doing this on the CPU was more than acceptable, and avoids having to transfer the color images to the GPU in addition to the laser images. The bulk of the time the GPU spends is on the data transfer. This is therefore a good alternative. The debayering of specific pixels was implemented by a colleague at SINTEF (Andreas Ulvøen).

3.6 Polarization Filter

A linear polarization filter is an optical component often used in photography to reduce unwanted reflections of a certain type. Light can be regarded as a flow of photons, or an electromagnetic wave. The waveform oscillates in an arbitrary direction perpendicular to the direction of travel. A polarization filter is placed in front of the camera lens, and only allows the waves traveling in the same direction as the polarization direction to pass through unaltered. All other waves will be attenuated by a factor of $\cos(\theta)$, where θ is the angle between the wave direction and the polarization direction. In an evenly illuminated space, the net result is a reduction of light by half. Rotating the filter adjusts the polarization direction, which can be used to determine which reflections to remove. The polarization filter on the lens is thus rotated so as to have a polarization direction perpendicular to the polarization direction of the laser beam. This almost removes the specular reflections - which for example occurs on wet, shiny or metallic surfaces. Only light that has actually entered the surface, and thereby been diffused and depolarized, is thus imaged. The effect of utilizing a polarization filter is a reduced dynamic range, but more robustness with respect to wet and shiny surfaces. One can therefore focus on the interesting illumination, e.g direct or scattered laser light.

The wet scales of Atlantic salmon polarizes reflected light. A polarization filter can therefore be utilized to reduce the glare, which is caused by reflections aimed directly at the camera lens. This is similar to the effect photographers experience while taking images of water, or the skies. By orienting the filter perpendicular to the laser-line, much of the direct reflections can be reduced. Initially, the camera rig was outfitted with a polarization filter. Later in the project, it was decided that the experiment should proceed without such filtering. The reason for this can be plainly seen in figure 3.12, where large parts of the backside of the salmon is missing. This is due to the different reflective properties on the backside, where there are no scales and a darker color. This effect can be mitigated slightly by reducing the threshold utilized when calculating the weighted mean of the laser-line.

Figure 3.12 has the lowest threshold possible before background noise increases rapidly, however.

The decision was therefore made to remove the polarization filters, and accept an increase in noise levels and glare. The primary focus of this thesis is the analysis of the shape features, and losing parts of the curvature of the back could potentially reduce the effectiveness of the algorithms developed. One could of course assume that the curvature approximately follows a circle or ellipse, but it is better to obtain true data instead of extrapolating. Figure 3.13 displays a raw scan of the salmon denoted as fish16 in the dataset, taken without a polarization filter. The patches that was previously void of coordinates, is now filled. The increased noise levels is plainly visible around sharp edges, however. The reason for this is that the camera to a larger degree captures light reflected off the fins and the edge of the abdominal cut. In addition to the noise around the edges there is also increased noise along the side of the salmon facing the top camera. The top camera has a more direct orientation towards the scales of the salmon, and is thus more susceptible to glare. This can be seen by tracing the middle image along the left edge, where the surface appears to be wavy when the curvature should be smooth. Fortunately the noise does not affect the entire top side, due to the lateral curvature. This noise can therefore be handled in software with relative ease. The two bottom cameras are not affected by the glare effect since they are positioned at an angle, when the most reflective scales are on the sides. A possible future remedy for the glare effect is to add another camera and capture the upper side similar to the bottom configuration, or increase the strength of the laser. In an industrial version of the scanner, it is suggested to have the polarization filter perpendicular to the laser and a polarization filter on each lens, as originally intended, and to use more powerful lasers so as to ensure a strongly observable laser line.

It should be noted that some of what appears to be noise behind the anal fin (hind abdominal fin) of fish16 is due to innards captured while hanging down between the conveyor belts.



Figure 3.12: A scan obtained of a salmon using a polarization filter. The lighter areas on the back indicate missing coordinates (the points from the bottom are seen through the gaps).



Figure 3.13: An image of Fish16 in the data set, obtained without a polarization filter. Increased noise is present around fins and sharp edges due to reflections of laser light. Left: Side view, Middle: back view, Right: tilted abdominal view

Chapter 4

Experiments and Feature Extraction

This chapter outlines the execution of the experiment and the treatment of the data obtained. The data must be treated to extract *features* which can be used to separate the classes. The goal here is to keep as much of the features that distinguishes the classes as possible, while rejecting irrelevant data. A challenge in all machine learning tasks is to find those features that best separate the samples. First, how the features are extracted will be outlined and justified from previous work and practical considerations. After this, the raw features are analyzed, which provides insight that can be used when deciding on the type of classifier and its configuration. Some of the algorithms and features presented here, are performed per frame. That is, a single slice in the XY-plane. The terms slice and frame are used interchangeably to describe whenever this is done.

Note that all the salmon from the data set is available to be viewed online. The web page features both still images and an interactive point cloud 3D-viewer. These point clouds have a reduced density compared to the full point cloud (roughly 80% of the points). The QR code in figure 4.1 can be used, or the URL in the caption. Additionally, a list of all the salmon and their numbering with comments can be found in appendix A.



Figure 4.1: QR code pointing to the scan data located at <http://ntnu-msc-oystestu.s3-website-eu-west-1.amazonaws.com/>.

4.1 Quality Grading

We will briefly re-iterate some aspects of the quality-degrading cases from the introduction, with salmon from the actual dataset. It should be noted that not all of the salmon from the dataset have been useful with the goals of this thesis in mind. Some were damaged in the gutting process itself, which was not considered in the design of the features. While bad gutting technically can be detected by computer vision, the number of samples is low and variability of the type of bad gutting large - causing that aspect to be put aside in this thesis. Others contained melanin-spots in the fillets - which cannot be detected by external vision, as one need to look inside the gutted portion of the salmon to see them. The focus of this thesis has been externally observable deformities and wounds.

There are 45 salmon of the type superior, which serves as a base of comparison. These are relatively free of deformities and completely free of wounds. From the ordinary class, 15 salmon contains deformities. The production class contains 10 salmon with deformities. In total, 15 salmon in the dataset contained wounds, all of them in the class production. Fashioning wounds by cutting a wound-free salmon was attempted, but does not work, as the appearance of the wounds changes due to long exposure with water, bacteria and healing. The appearance of the wounds are dark, as opposed to the flesh - which is light pink or red. Sometimes the wounds appear dark green, sometimes dark red.

4.1.1 Humpback

One of the most visually recognizable form of deformity, is that of the humpback. This occurs when the vertebra of the spine fuses together in a curved shape with a greater roundness than normal. Although the humpback itself is one deformity, it is really its affect on the symmetry of the salmon that causes it to be downgraded. This is also a safe assumption with regards to healthy salmon, as they are streamlined to reduce resistance in water. In captivity, there is no natural selection that singles out the salmon with bad backs, so to speak. Figure 4.2 contains a direct comparison of one of the superior salmon, and

two salmon from the dataset. A is a streamlined, well shaped salmon. B is one of the most obvious cases of humpback in the dataset. C is another salmon with humpback, to show that not all salmon with humpback in the dataset are close to the extreme of salmon 32. In reality, the humpback deformity can be quite subtle, and not necessarily exclusive to the front of the salmon. This is one of the aspects that makes the classification challenging.



Figure 4.2: Comparison of three salmon from the dataset. A: Salmon 5 with no deformities, B: salmon 32 with forward humpback, C: salmon 94 with a less protruding hump (further back)

4.1.2 Wounds

Wounds in salmon can appear in all sizes, and primarily occurs on the sides. When the salmon is processed, the wounds can be healed - but still causes a downgrading of the quality due to the scarring. The wounds themselves can start out as small scratches, which are then infested by bacteria, causing the wounds to expand. In waters of high salinity, such as seawater, the bacteria *Moritella viscosa* is found to be the largest cause of these wounds. The outbreak of that type of bacteria often occur during periods of cold water, and is for that reason called winter wounds or winter ulcers. The salmon can recover from these wounds as the water gets warmer, but the scarring remains.

The salmon with wounds in this thesis was obtained 10th of June, and therefore consists of these type of healed wounds, not open wounds that are still infested. For that reason, the work in this thesis on the detection of wounds should be revised in time using a dataset consisting of both types of salmon. One can also devise a classifier which changes depending on the time of year to account for the type of wounds most common. There will naturally be a gradual shift in appearance depending on the temperature, which can be accounted for. The wounds present in this thesis can be seen in figure 4.3.

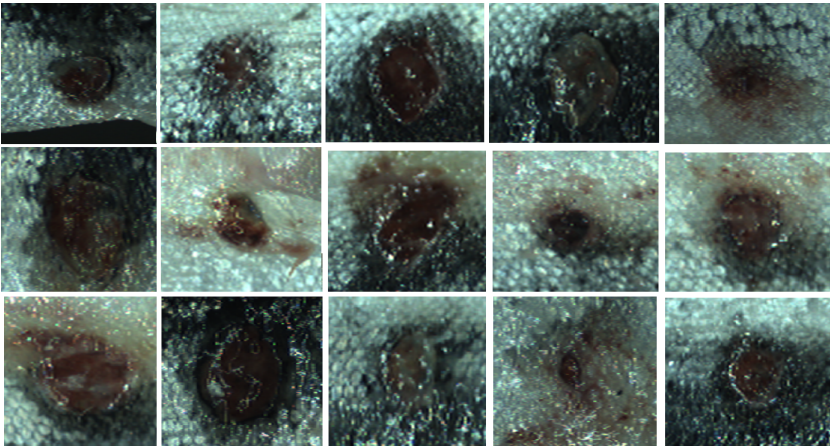


Figure 4.3: All the wounds present in the dataset.

4.2 Point Cloud Post-Processing

This section details the post-processing performed on the point cloud. The first step in any application utilizing point clouds is to reduce the effects of noise. In this thesis there are two distinct types of noise. Statistical noise which materializes as isolated outliers, and systematic noise - which occurs as a side effect of unwanted light. Statistical noise can for example occur due to random anomalies in the camera sensors. This type of noise typically affects each pixel independently. Since the extraction of the centerline of the laser is performed by calculating the first moment, the effect of this is greatly reduced. The threshold applied to each pixel keeps the average background noise at bay. While the threshold multiplier, which enforces the requirement of a certain number of pixels above the threshold, mitigates some of the effect of outliers. These two factors reduce the statistical noise pertaining to the optics to an unnoticeable level. This leaves the effect of systematic noise due to unwanted reflections. Additionally, actual physical characteristics of the salmon might be unwanted in the point cloud for further analysis. As the salmon scanned are already gutted, innards has the tendency to occasionally hang down from the side of the salmon. The innards typically flaps about relative to the direction of movement, causing a larger spatial impact on the point cloud than its size.

4.2.1 Statistical Outlier Removal

The first step used to filter unwanted noise is a statistical outlier filter. The basic premise of the filter is quite simple. For each point in the dataset, the filter computes the mean distance from that point to its K nearest neighbors - which is stored. The collection of the mean distances for all the points is then assumed to follow a Gaussian distribution. The mean and standard deviation for the collection of mean distances is then computed. All points which lie outside a specified standard deviation multiplier is considered outliers, and is removed.

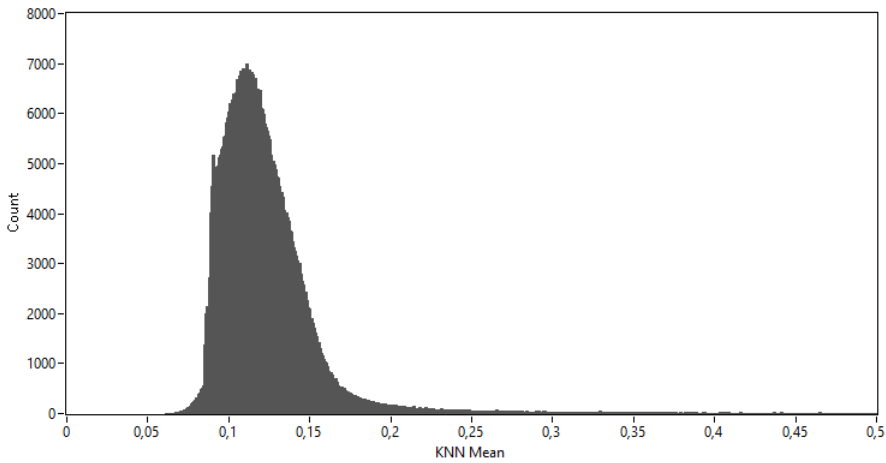


Figure 4.4: Histogram of 25 nearest neighbor means for fish16 in the data set

Figure 4.4 contains a histogram of the mean nearest neighbor distances for $K = 25$ for a salmon in the dataset. Generally, it seems that the assumption of a normal distribution is not far fetched. The selection of the K parameter is a matter of tweaking the number of neighbors in relation to the density of points across the surface. It adjusts in a sense the density required in a local feature for it to become permanent. A high K -value increases the smoothness of the filter, but does increase the computational complexity. Higher values of K does not necessarily produce better results, as the difference in density between the actual surface and outliers is large. For the standard deviation threshold, an value of 2.5 was selected, as values lower than 2 caused areas of lower density (on the back) to disappear. Figure 4.5 displays the same salmon as in figure 3.13, after it has been processed by the statistical outlier filter. The end result is a salmon where the spray of coordinates surrounding the edges is almost completely removed, and the coordinates around the back is preserved as opposed to the polarization filter.

The implementation of the statistical outlier filter used is from the point cloud library (Rusu and Cousins (2011)), which is an open source initiative for large scale point cloud processing. While the basic concept of this filter is easy to implement, the PCL library does so by first sorting the point cloud into a tree structure - thus speeding up the search process. Processing fish16, which consists of more than 1 million coordinates, takes 2.6s on an Intel i5-4670K 4.1GHz processor. It is possible to improve on this by either implementing the algorithm on the GPU, or by utilizing the fact that the point cloud is semi-sorted. The point cloud is completely sorted along the z-axis, since each frame results in a single slice. The coordinates for the top camera is sorted along the x-direction as the image columns is perpendicular to the conveyor belt. The two bottom cameras are also likely to be partially sorted along the x-axis depending on the curvature of the object. By utilizing this information it is possible to speed up the nearest neighbor search significantly.



Figure 4.5: Fish16 after applying the statistical outlier filter.

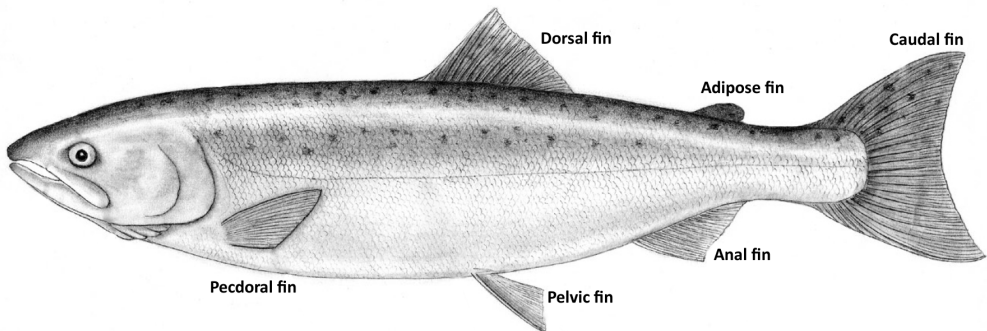


Figure 4.6: The given anatomical names for the fins of salmon (adapted from an illustration by Karen Uldall-Ekamn)

4.2.2 Fin Removal

It became clear early on that the fins posed a problem in the extraction of geometric features. Both because of the irregularities introduced into the point cloud due to the sharp, protruding, nature of the fins and its variability. The fins can move, curl in on itself, or vary in size altogether. This causes an issue in the geometric analysis, as they introduce a varying geometric element outside of the actual curvature or form of the salmon. For this reason, an algorithm has been developed to remove the fins consistently, and taking the curvature of the salmon into account while doing so. This algorithm is important to facilitate proper analysis of the symmetry, or lack thereof - as the fin interferes with the general curvature of the salmon. The algorithm is only applied to the region of the salmon where the fins are located. This is to avoid interpreting the gutted part of the abdomen as a fin. Only the anal fin, dorsal fin and adipose fins are considered, as those are the most protruding ones. The pelvic fin and pectoral fins are normally more aligned with the body, and does not pose a problem. The fins in question can be seen in figure 4.6.

The algorithm consists of an initial step where a naive detection of the fins is performed by looking at how spread the coordinates in a single slice is along the Y -axis. This is computed by first sorting the coordinates in increasing X -direction. The coordinates are then placed into bins of 0.2 cm. For each of these groups, the interquartile range along the Y -axis is computed. The interquartile range is the difference between the upper and lower quartile. This is known as a trimmed or truncated estimator of range, as it basically discards 25% of the data in both directions. It is thus one of the most basic forms of robustness for this type of calculation. For the case of the fins, the interquartile range is roughly equal to taking the difference of the top and bottom of the fin. If five consecutive bins on either side of the slice is below a threshold of 1.5 cm, the frame is assumed to contain a fin. The reason why this basic detection works, is that the curvature of the body of the salmon, excluding the fin, increases rapidly thus exceeding the interquartile threshold.

Once a fin has been detected, the coordinates that falls in the bins and are under the threshold are temporarily removed from the active set. The remaining coordinates is then split about the mean X -value. The coordinates on the side of the mean facing the fin is then

fitted to an ellipse. The `fitEllipse` routine in the OpenCV library (Bradski (2000)) is used for this purpose, which implements an algorithm that minimizes the algebraic distance of the ellipse. This algorithm is quite fast, but breaks down completely in the face of high levels of noise or a large difference in the two principal directions. The side of the salmon facing the back of the salmon maintains a high level of roundness throughout, usually close to a circle. Fitting an ellipse to this was never found to be a problem. The side facing the abdomen, which is gutted, was sometimes found to be problematic in the elliptical fit. This occurred when the gutted part was long and thin, without a roundness to the curvature (i.e. two lines that almost meet at a point). The solution found was to include coordinates past the mean X value, thus including more of the curvature towards the back. No breakdowns was observed after doing this in the dataset. Further experimentation is necessary to determine if moving to a slower more robust algorithm is warranted.

After the elliptical fit, a linear regression is performed on the binned coordinates that was removed, with the constraint that it should pass through the center of the ellipse. The idea here is to determine the general direction of the fin. The linear regression is performed using a standard least squares procedure. The intersect point (x_0, y_0) of the ellipse and the line is then found using the equation in 4.1, where a and b are the major and minor axis of the ellipse. Since the quadrant is unknown, the intersect point with the closest metric distance is used. Note that this equation finds the intersect point for a non-rotated ellipse centered at the origin. The slope found through the linear regression is therefore evaluated at a point either to the far left or right (along the x-axis) depending on which side the fin is on. The corresponding point is then translated by (x_c, y_c) , which is the center-point of the ellipse. The point is then rotated clockwise by the angle of the ellipse found in the elliptical fit. This ensures that the point from the linear regression is in the same coordinate frame as the unrotated, centered ellipse.

$$\begin{aligned} x_p &= \pm \frac{ab}{\sqrt{a^2y_0^2 + b^2x_0^2}}x_0 \\ y_p &= \pm \frac{ab}{\sqrt{a^2y_0^2 + b^2x_0^2}}y_0 \end{aligned} \quad (4.1)$$

The intersect point should be roughly positioned at the location where the fin meets the body. The tangent of the ellipse in that point is then found using equation 4.2. Here (x_p, y_p) is the intersect point on the ellipse found previously.

$$y_{tan} = -\frac{b^2x_p}{a^2y_p}x + \frac{b^2}{y_p} \quad (4.2)$$

The tangent found is the tangent of the non-rotated, non-translated ellipse. Instead of rotating all (x, y) points to this reference frame, the line is rotated back to the original reference frame. This is done by simply rotating the orthogonal basis vectors and re-solving for the parametric equation of the line in the new reference frame.

$$\begin{aligned} a' &= -\frac{a\cos(\theta) + \sin(\theta)}{-a\sin(\theta) + \cos(\theta)} \\ b' &= y_c - a'x_c \\ y' &= a'x + b' \end{aligned} \quad (4.3)$$

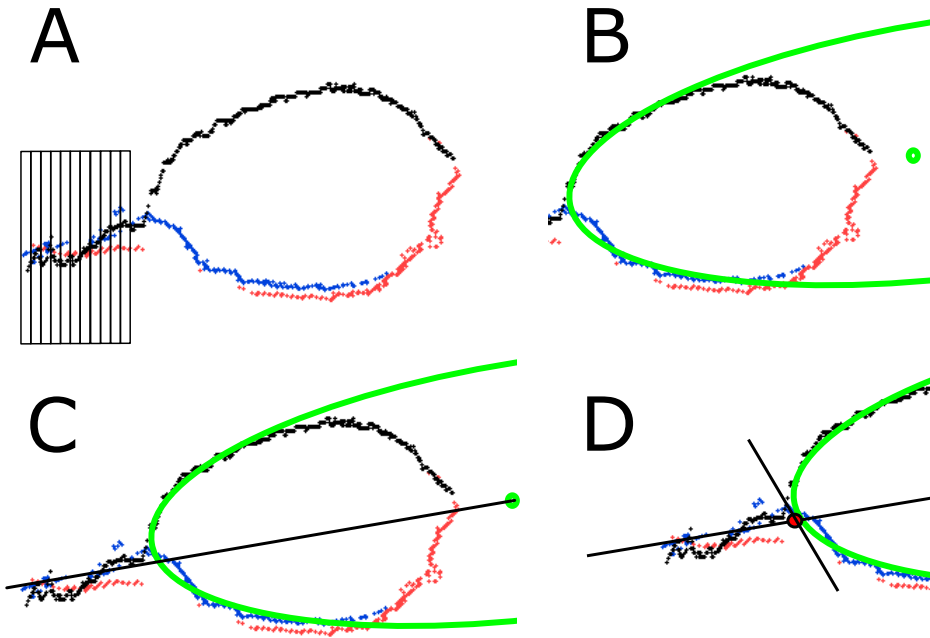


Figure 4.7: Illustration of the fin removal steps applied to the anal fin of fish19 from the data set. In A, the fin is detected by binning the coordinates along the X-axis and calculating the spread in each bin. If enough bins are below the threshold, an ellipse is fitted to one side of the slice as in B. In C a linear regression is performed with $b = 0$ and origin at the center of the ellipse. In D, the intercept point of the line and ellipse is calculated and the tangent in that point is used to remove the fin.

Finally, this tangent is used to define a cutting plane on the original coordinate set containing both the binned and un-binned coordinates.

This process is applied to all the slices which has been selected due to being below the binned threshold. There is, however, a high likelihood that some frames has not been processed due to being above the threshold - even if a fin is present. The solution to this is to find the closest slices that has been processed on both sides, and use their tangents to define the cutting for the frames in between. The most straightforward way to accomplish this is to define an average plane between the two tangents, thus spanning an approximate cutting plane. A better method, which is used instead, is to interpolate the angle of the normals of the two nearest tangents. The intercept points are also interpolated to calculate the bias of the tangent. If multiple frames are missing, this forms a twisted plane - which follows the movement of the fin frame-to-frame. Additionally, this interpolation procedure is also applied between the edges of the fin-region and the nearest bin to the edge. This removes the fin as it gradually starts to form.

Algorithm 1 One-Sided Single Frame Fin Removal (Left)

```

1: procedure Q = QUARTILEBINS(X, Y)
2:   Q ← {}
3:   bins ← divide coordinates into equally sized bins by X
4:   for all bin ∈ bins do
5:     Q1 ← Compute lower quartile (25th percentile)
6:     Q3 ← Compute upper quartile (75th percentile)
7:     if (Q3 − Q1) < QThreshold then
8:       Q ← Q ∪ bin
9:     else
10:      break
11: return
12: procedure INDICES = FINREMOVAL(X, Y)
13:   I ← {} // Index set of indices to keep
14:   X, Y ← Sort (X,Y) by X in ascending order
15:   Q ← QuartileBins(X, Y)
16:   if SIZE(Q) > 5 then
17:     width, height, angle, cx, cy ← FitEllipse({X \ Q} ≤ Mean(X \ Q))
18:     slope ← LinearLeastSq(Q − {cx, cy}) with constraint b = 0
19:     xp, yp ← Calculate intersect point of slope and ellipse according to 4.1
20:     a, b ← Calculate tangent at (xp, yp) according to 4.2
21:     arot, brot ← Rotate a, b according to 4.3
22:     for i ← 0 to length(X)-1 do
23:       // Evaluate tangent with each X-coordinate in the set
24:       y ← arotX(i) + brot
25:       // Evaluate which side of the tangent the coordinate is
26:       isLeft ← arot ≥ 0 ∧ y ≥ Y(i) ∨ arot ≤ 0 ∧ y ≤ Y(i)
27:       // Keep coordinate if it is on the correct side of the tangent
28:       if isLeft then
29:         I = I ∪ i

```

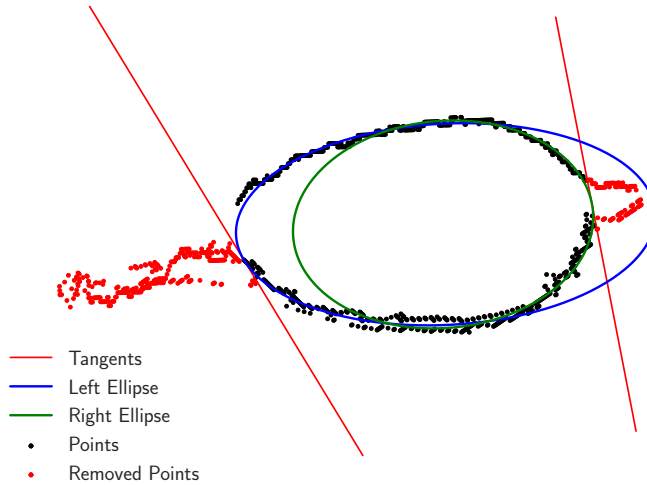


Figure 4.8: The fin removal routine applied to a slice where a fin is detected on both sides.

4.2.3 Spline Re-sampling

As mentioned in the literature review section, the Voronoi diagram is sensitive to both noise and sample density along the boundary. The sample density varies depending on the perspective and overlap between the cameras. This is a common problem in point cloud processing, and a technique often employed is moving least squares (MLS). MLS is based on local fitting of the surface to a polynomial in the least squares sense, and can be used to interpolate and re-sample irregularly distributed points. While this method has the potential to work very well, it was found that the computational complexity far exceeds the real-time requirements of this application. Part of the reason for this is that the local search radius had to be set quite high to counteract the effect of the induced noise due to glare and reflections. The implementation of MLS used was from the Point Cloud Library, which does not exploit the inherent structure due to the laser-scanner obtaining one slice at a time. An future implementation exploiting this property, implemented on a GPU, could obtain enough throughput for a real-time application on dense point clouds.

Instead of the MLS, a re-sampling and data-smoothing is performed in the form of a spline interpolation. This is done for all 2D-slices in the XY-plane where the Voronoi diagram is needed. The drawback of this is that one of the spatial dimensions is not utilized. A counterpoint to this argument is that impact of noise is worst in regions of reflections and glare, and neighboring slices will also inhibit the same structured noise characteristics - thus limiting the usefulness of that spatial dimension. This is also the reason the search radius had to be set quite high with the MLS routine.

To be able to perform the re-sampling using splines, one must first place the points into its correct sequence along the boundary. Two main approaches was considered in this regard. The first being an iterative procedure, where the order of the points is determined

by traversing the edge. This can be done by employing an nearest neighbor procedure, which can be enhanced to select the correct path by considering the curvature of the previous points to determine in which direction to move. This has been done in [Liu and Ye \(2011\)](#), where a snake model is used to follow a gradient flow by attributing the points with a certain attraction. With a real time application in mind, a simpler alternative was followed. Due to the removal of the fins, the coordinates as seen from the center of a slice is fairly regular. In light of this the coordinates can be parameterized using polar coordinates centered at the centroid of the slice. For the most part, this gives a good indication of the sequence. The exception to this is where the curvature bounds back on itself as seen from the inside. This occurs rarely after the removal of the fins, but three main exceptions to this has been observed.

1. The laser-line can be reflected off its actual position onto something further up in the image. This has been observed to sometimes happen if the pelvic fin is directed outwards, as opposed to flat alongside the body as seen in [figure 4.9](#). This might be a result of the fish being frozen with the fin in an abnormal position. What happens specifically, is that the reflection of laser-light on the fin is mistaken as *height* by the coordinate transformation. This therefore produces a spray of coordinates directed upwards. It should be noted that this was not a problem before removing the polarization filter, as the reflections were reduced sufficiently to be below the threshold. Part of the reason for this problem might be that the fleshy part of the cut is exposed as a flat, light surface which causes the light to scatter more.
2. Sometimes the innards of the salmon will hang out due to the gutting. In the experiments this was done intentionally to simulate real-world conditions. Coordinates below the conveyor belt plane in the Y-direction are removed, but some remains is still left over that can make an impact on the spline. This can be seen in [figure 4.10](#) under subfigures A, C, E, F, G and I. The coordinates facing downwards on the left side ends abruptly because something has been hanging down and been removed at a threshold (subfigure A has lower threshold for illustration purposes). Although some of these will produce a perfectly good spline without modifying the dataset, it will skew the spline slightly outwards.
3. The final disturbance in the point cloud occurs due to the pectoral fin being angled outwards as a opposed to along the body, similar to the first case. The pectoral fin does not produce the same type of reflections as in the first case however. The reason for this is that the pectoral fin has a darker color, and thus reflects less light. Additionally, it is angled more towards the laser-line. The disturbance is therefore mostly due to the presence actual fin in the laser-line, as opposed to a reflection. It could potentially be handled similarly to the dorsal, adipose and anal fin with the tangent approach. Since the problem is intermittent and has less of an impact, it could be overkill however. Additionally, the curvature surrounding the pectoral fin has less regularity than the three fins handled by the ellipse-tangent algorithm due to gutting.

In these cases, the sequence sorted by the angle θ would no longer coincide with the order of points along the boundary (in that region). These features must therefore be pruned,

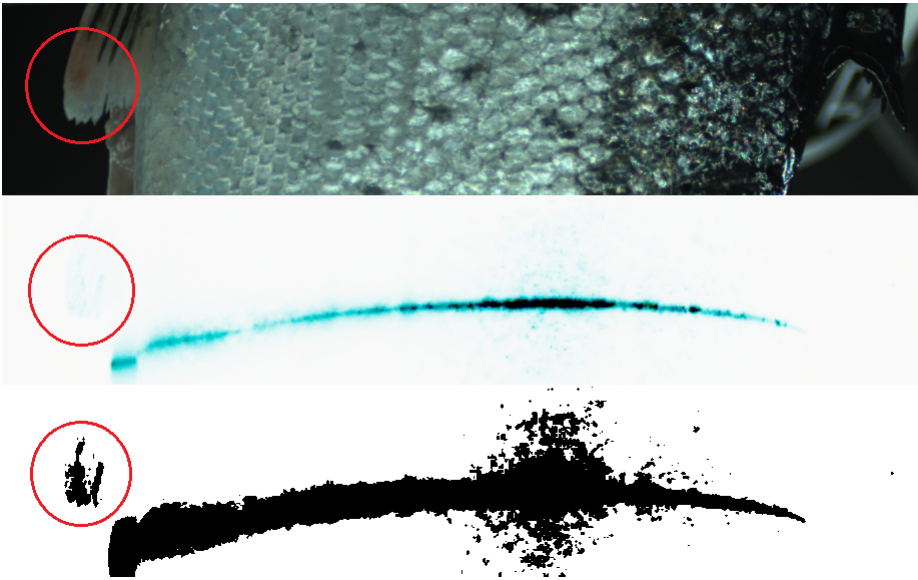


Figure 4.9: A case where significant noise is introduced into the 3D-model due to reflections from the laser-line. The reflections hits the pelvic fin moving into the frame. The middle and bottom images is the laser-line as seen by the camera and a thresholded image, respectively. The laser images have been inverted for viewability, which is why the laser-line appears to be cyan instead of red.

as they are unneeded and unwanted in further analysis. Figure 4.10 displays some examples of features which must be pruned, and the spline result after successfully pruning. Algorithm 2 contains pseudocode for the procedure, and the implementation in Matlab can be found in appendix C. Note that the pseudocode does not handle the polar coordinates wrapping around.

The basic premise is to detect the regions of the XY-slice where following an increasing sequence of θ causes ρ to change abruptly. This is an indication of multiple surfaces, or layers being present in that direction. If that region is above a certain size, measured by the angle from region start to end, an outwards pruning is performed. Since these unwanted features are based on the presence of actual objects such as fins, the observation is that the unwanted coordinates are never directed inwards. This observation is used to make the spline follow the inner path, ignoring the outer surface. This is done naively by calculating the mean ρ for the region, and discarding the points outside that threshold. This procedure is only performed once. The remaining set of coordinates is then used to calculate a spline approximation. The residuals of the spline approximation is studied to look for both outliers and inliers. These are iteratively removed while the spline is retrained and residuals recomputed. Note that the threshold for removing these are set quite high to avoid removing too many points and the performance penalty of training many splines. In fact, this iterative procedure has become largely redundant after adding the statistical outlier removal.

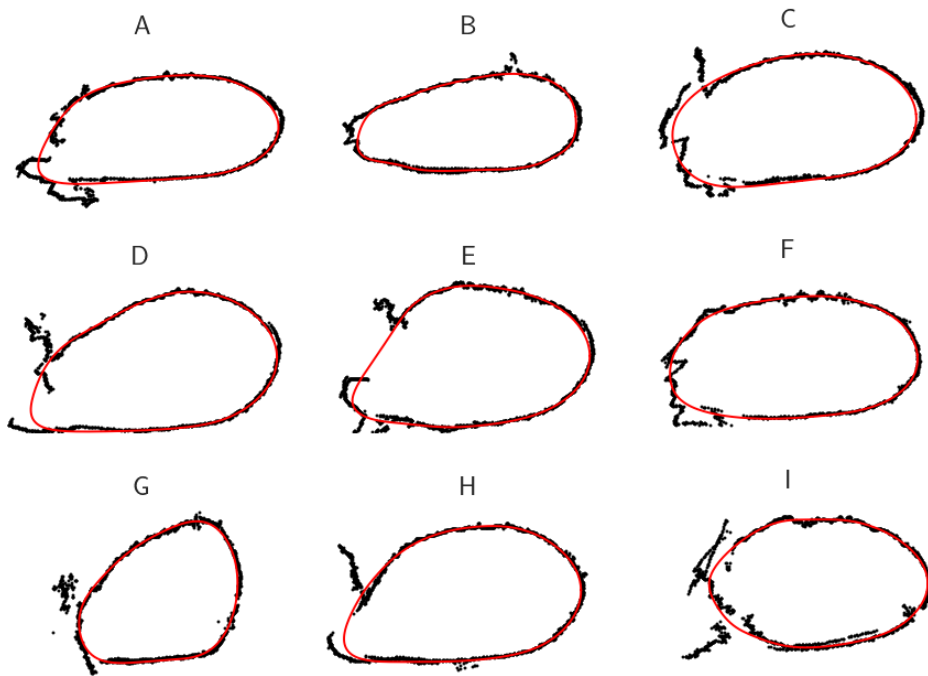


Figure 4.10: Instances where pruning of the exterior features is necessary to obtain a good spline approximation of the main curvature. The coordinates are shown as well as the spline obtained after pruning.

Algorithm 2 Spline Pruning and Spline Re-sampling

-
- 1: **procedure** $(X_{spl}, Y_{spl}) = \text{SLICESPLINE}(X, Y)$
 - 2: Calculate centroid of X,Y and translate all points so that it is at the origin
 - 3: Express the coordinates on polar form ρ and θ
 - 4: Sort by increasing θ and apply same indexing to ρ, X, Y
 - 5: Calculate difference between each subsequent value of ρ
 - 6: Smooth local variations by running $\Delta\rho$ through an averaging filter
 - 7: $I \leftarrow \{\}$
 - 8: **if** any $\Delta\rho > \rho$ threshold **then**
 - 9: Divide into regions of θ that exceeds the threshold
 - 10: Merge regions that are within a small distance of each other ($\Delta\theta < 2^\circ$)
 - 11: Ignore small regions ($\Delta\theta < 5^\circ$)
 - 12: **for all** remaining regions **do**
 - 13: Calculate mean ρ for the region
 - 14: $I \leftarrow I \cup \{i \mid \rho > \text{mean}(\rho)\}$
 - 15: $X \leftarrow \{X \setminus X(I)\}$
 - 16: $Y \leftarrow \{Y \setminus Y(I)\}$
 - 17: Pad end of X,Y with coordinates from the start to ensure a smooth start/end
 - 18: Fit spline to data with the ordering of points found
 - 19: **while** abs(spline residuals) > residual threshold **do**
 - 20: Remove points exceeding residual threshold
 - 21: Fit spline to new dataset
 - 22: $X_{spl}, Y_{spl} \leftarrow$ Evaluate spline at equidistant points
-

Finally, the spline is sampled equidistantly with sufficient samples to calculate the medial axis. The theory behind splines is a large topic by itself. For the purposes of this thesis, considering the splines as an interpolation that ensures smooth derivatives of a certain order is sufficient. The representation chosen in polar coordinates from the centroid definitely has its benefits in speed and simplicity. There are however, some cases where this representation will not yield good results. Figure 4.11A illustrates the main case for this. If the upper abdominal flap curves in on itself quite sharply. The lower flap is then cut off by the pruning procedure, due to not being visible. Even if it was not cut off, the spline would not follow the curvature perfectly due to the coordinates being in the wrong sequence. Sub-figure B is not a case of the spline failing, but shows that the pruning will not be performed from any visible part as seen from the centroid. This case is not a problem in further analysis, as shall be seen as the extraction of features are detailed.



Figure 4.11: Fig. A illustrates an instance where the spline will cut off the edge. Fig. B illustrates an instance where pruning is not performed, as the entire protruding feature is visible from the centroid.

4.3 Medial Axis

Using the re-sampled coordinates, the Voronoi diagram can be computed. The vertices of the Voronoi diagram that are internal to the boundary forms the discrete medial axis. Even though the spline smooths the boundaries significantly, any slightly unevenness will cause the medial axis to branch out. This is simply one of the properties of the medial axis. The part of the medial axis that is of interest in this thesis, however. It is the medial axis that extends horizontally that will be extracted. The idea is to use the medial axis as a more correct measurement of height of the salmon than simply utilizing a the direct distance from the back to abdomen. Using this definition, the pose of the salmon does not matter - as tracing the horizontal medial axis will approximately yield the true height. The abdominal cavity contains open space after being gutted. The idea presented is that even if the abdominal flap is compressed, the medial axis will in turn curve more upwards due

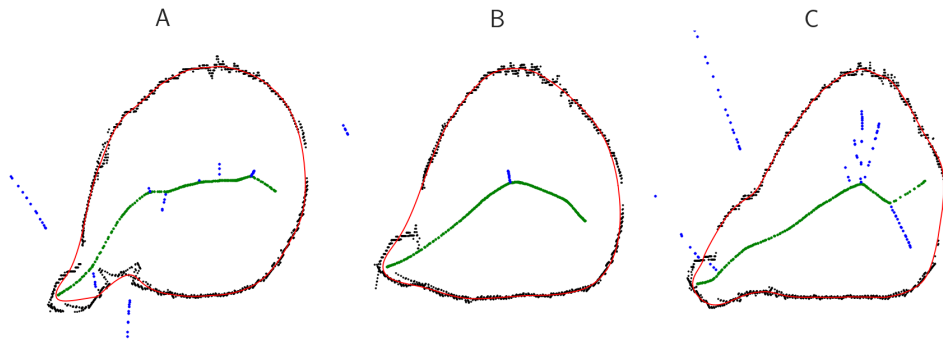


Figure 4.12: Three instances of medial axis calculation. The horizontal medial axis is displayed in green, while the discarded Voronoi vertices are shown in blue.

to the increased width (Y-axis), thus producing a longer horizontal medial axis than the direct distance. This will then yield a height that is more similar to the salmon, when fully extended - as opposed to lying on a conveyor belt.

To obtain the horizontal medial axis, the branches of the medial axis that extend to the upper and lower boundary must be pruned. For this purpose, algorithm 3 was developed. First, the Voronoi vertices external to the boundary of the slice is removed. A search is then started for the horizontal axis around the middle, in a node that has both an entry and exit node close to the horizontal plane. This avoids starting the algorithm off on a vertical branch of the medial axis. A search is then performed in both directions, choosing the path that minimizes the difference in angle from the previous node to the next one. This causes the search to follow the medial axis that extends along the major axis. Additionally, there are some constraints imposed on the overall angle, to avoid choosing a branch that is close to vertical. Figure 4.12 displays some examples of the medial axis calculation and horizontal selection. Sub-figure B shows an example of a more compressed form when compared to sub-figure A. The medial axis clearly curves upwards as a result, thus producing a longer medial compared to the direct distance. Sub-figure C shows an instance where the curvature of the salmon is cut a bit short due to missing data points. This causes the medial axis to branch out to the points where the curvature changes. While not a large problem, as shall be seen in the extraction of the height feature. The extraction of geometrical features from the medial axis is covered in the next section.

4.4 Geometric Feature Extraction

This section covers the extraction of the concrete geometric features used in the classification, based on the medial axis and re-sampled spline. Width and height are referred to frequently in this section, and applies to the distance in Y and X directions respectively (see figure 4.13).

Algorithm 3 Selection of Horizontal Medial Axis

```

1: Input: Voronoi Vertices (V), Voronoi Edges (E)
2: Output: Horizontal Medial Axis Vertices (H)
3: procedure (H) = HORIZONTALMEDIALAXIS(V, E)
4:    $H \leftarrow \{\}$ 
5:   // Find starting node
6:    $V, E \leftarrow$  Voronoi vertices and edges inside the polygon defining the boundary
7:    $i \leftarrow$  calculate index of mean(V.x)
8:   while no  $\angle E(i)$  within both  $\pm 45^\circ$  and  $\pm 45^\circ + 180^\circ$  do
9:      $i \leftarrow$  next vertex below mean
10:  // Iterate downwards from starting node
11:   $i_{start}, i_{prev} \leftarrow i$ 
12:   $i \leftarrow$   $\angle E(i)$  closest to an angle of  $180^\circ$ 
13:  while any  $\angle E(i) - \angle E(i_{prev}) < \text{threshold}$  do
14:     $H = H \cup i$ 
15:     $i_{prev} \leftarrow i$ 
16:     $i \leftarrow$  next vertex with least difference in angle
17:  // Iterate upwards from starting node
18:   $i_{prev} \leftarrow i_{start}$ 
19:   $i \leftarrow$   $\angle E(i_{start})$  closest to an angle of  $0^\circ$ 
20:  while any  $\angle E(i) - \angle E(i_{prev}) < \text{threshold}$  do
21:     $H = H \cup i$ 
22:     $i_{prev} \leftarrow i$ 
23:     $i \leftarrow$  next vertex with least difference in angle

```

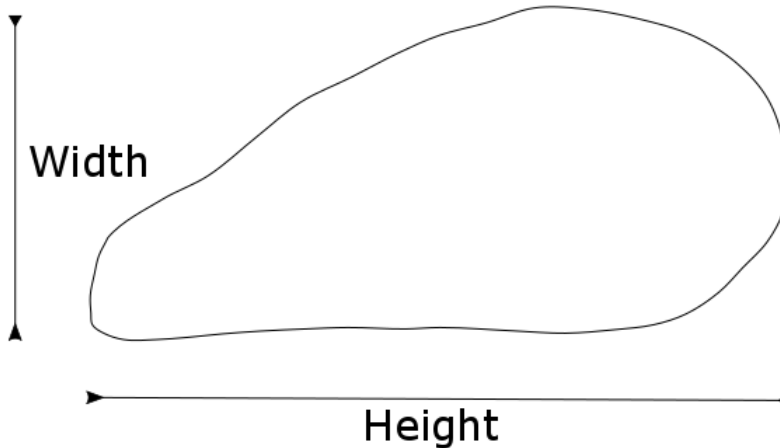


Figure 4.13: The directions of width and height of a salmon, as referred to in this section.

4.4.1 Width

As the salmon is lying on its side, one could compute the width quite accurately by taking the difference between the upper and lower coordinate of the Y-axis. Alternatively computing the difference between the top coordinate and the plane of the conveyor belt, possibly from the spline as a robustness measure. An alternative route was taken, in which the maximum inscribed circle of the medial axis was found instead. An example of this, for a single slice, can be seen in figure 4.14. The width is thus extracted as the diameter of the maximum inscribed circle. When the smoothness of the spline approximation is high, as in this thesis, the benefit of using this definition of width is reduced. It does however provide more robustness in the instances where the spline curves outwards locally. It is therefore a more general robustness measure for the width than simply relying on the spline smoothness. Additionally, the maximum inscribed circle needs to be computed regardless as it is used in the extraction of other features.

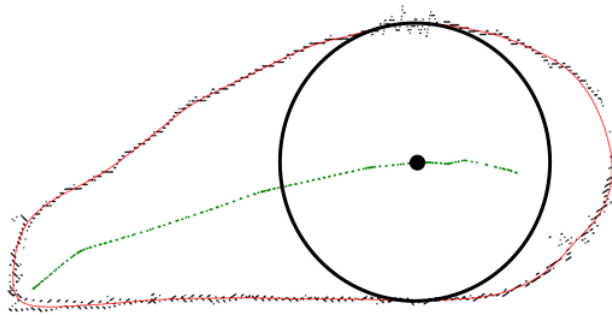


Figure 4.14: The vertex containing the radius of the largest embedded circle is used in the calculation of the medial length, and as a separate feature.

4.4.2 Height

The height of the salmon is the distance from the abdomen to its back. It is only used sparingly as a direct feature, as composite features are preferred to keep the dimensionality of the feature space low. It is aggregated in further features that attempts to describe the symmetry of the salmon, however. The medial height is therefore calculated for all the slices, although a sub-sampling the slices will most likely also be sufficient. In addition to this, the medial vertex that embeds the largest circle within the boundary is computed. Figure 4.14 displays an example of this circle. The direct medial height and the radius of the largest embedded circle is used as a direct feature for equidistant slices across the salmon. The reason for this is that the other features developed are intended to be invariant to the size of the salmon, and does not detect if a sample should be downgraded due to its absolute size. Examples of this are sample number 40 and 76 in the dataset, which are instances where the size of the salmon is too small to be usable even though no deformities are present. During the trials, as few as 3 to 5 equidistant slices throughout the length of salmon was an sufficient number to provide the desired effect.

The typical difference between the length of the medial axis and the direct length between the sides of the salmon is shown in figure 4.15. The length along horizontal medial axis typically is the same as the direct length at the start and beginning of the salmon, as the slices there are more circular. In the middle, the length by tracing the horizontal medial axis becomes longer due to the gutting and pose. This is the whole point of using the medial axis for this purpose. By calculating the length along medial axis, an attempt is made to compensate for the effect lying on a flat surface has on the measurement.

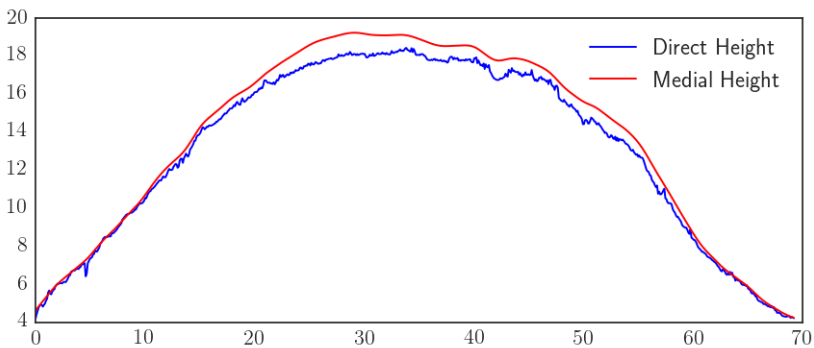


Figure 4.15: The typical difference between the direct length from the left of the spline to right of the spline and the length by tracing the medial axis.

Note that at each end of the horizontal medial axis, the radius is added to the length calculation.

4.4.3 Length

The length of the salmon is another feature extracted. This is perhaps the feature where the pose of the salmon has the most impact, due to the salmon possibly being bent in either direction while traveling across the conveyor belt. To counteract this effect, the length of the salmon is calculated by adding the distances between the center of the largest embedded circles for all slices. This is performed from the tip of the salmon, to the slice which has the smallest area towards the end. Finding an ending slice that is fairly constant across the salmon is important, as the fin itself can be curled and bent. Depending on the stiffness of the fin, it can also fall down slightly while passing over the space between the conveyor belts. Since the segmentation of the salmon depends on the object being above a certain height, as seen from the top camera, it introduces a variability in where the last frame will be. It might cut off before the actual fin has passed as a result. This is avoided by utilizing the slice of smallest area, which provides a relatively static point of reference. An illustration of the medial length can be seen in figure 4.16, along with the starting and ending slices. The length itself is calculated by linearly interpolating from center to center between the frame and summing the lengths between all frames.

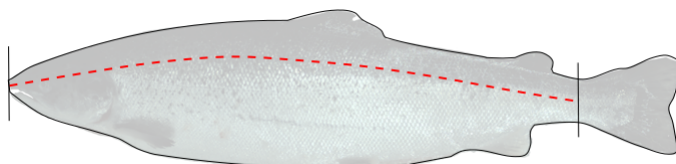


Figure 4.16: The length of the salmon is represented through the medial length, which traces through the center of the largest embedded circle for each slice.

Given enough samples in the dataset, one could potentially feed SVM with a large-dimensional feature vector - and leave it up to RBF to find a nonlinear pattern among the samples. As it is, this is an option that cannot even be considered due to the small size of the current dataset. It might be a viable solution when employed in an industrial scanner with access to 3D scans of thousands of salmon in each quality grad. For this thesis, it is better to direct the classifier in the right direction, as the pattern RBF finds might not be the one that generalizes best with so few samples.

Medial Contour Length

In addition to the medial length that passes through the interior of the salmon, the length along the contour is also extracted. This is done by using the medial heights and the known distance between each frame to sum over linear interpolation as in the previous length feature. The idea with this feature is that the less symmetry the salmon has, the longer the trace of the heights becomes in relation to the direct length. With the fins removed, this becomes viable as a feature. In fact, this feature is one of the more contributing features towards classifying deformities.

4.4.4 Skewness

As the lack of symmetry is important in detecting deformities, another feature was developed to attempt to fill the gaps where the medial contour length could not separate the classes. This feature revolves around the skewness of the height instead of the contour length. Skewness is a measure of asymmetry of probability distributions in statistics. Primarily for unimodal, asymmetrical distributions - as the results becomes hard to interpret in multi-modal distributions. There are many different formulations that attempt to describe skewness. A recurring theme is the use of the third central moment. Pearson's moment coefficient of skewness is defined in equation 4.4.

$$\gamma_1 = E\left[\left(\frac{X - \mu}{\sigma}\right)^3\right] = \frac{\mu_3}{\sigma^3} \quad (4.4)$$

Replacing the third central moment and the standard deviation with their sample counterparts yields the following expression for sample skewness.

$$\beta_1 = \frac{m_3}{s^3} = \frac{\frac{1}{N} \sum_{i=1}^n (x_i - \bar{x})^2}{\left[\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2\right]^{3/2}} \quad (4.5)$$

This definition of skewness is intended to be calculated for a randomly distributed variable. It does not, however assume that the variable follows a specific distribution. Calculating this for all of the heights will not yield the desired results, as the notion of front and back of the salmon will be lost. Instead, the salmon is divided into front and back at the middle of the length calculated previously. The skewness is then calculated for each of the halves, and used as features.

While healthy salmon are symmetrical about its width (lateral), it is not completely symmetrical along the length axis (longitudinal). The back part of the salmon is typically

slightly more stretched than the front part. One can imagine that this is a result of natural evolution to reduce the resistance in water. Regardless of the reason, this means that what we are looking for in these symmetric features are not really a perfect symmetry of the salmon. Instead, we are looking for some sign of abnormalities - which typically means that the shape is uneven as opposed to streamlined.

4.5 Color Image Projection

The color and reflectance information has been extracted and designated a coordinate in the point cloud. Analyzing the color composition of the cloud directly proves difficult due to the spatial components. Instead, the point cloud can be projected into one or more 2D-images for further analysis. Operating on a regular image improves the speed of the algorithms, but more importantly - allows usage of image processing algorithms such as morphology and filtering without reinventing the wheel. Optimally, the projection would preserve the surface area of the salmon in the 2D-projection. Doing this would map the spatial component to the 2D-plane in addition to the spectral information. This could be used to obtain an accurate notion of size in the analysis, for example the size of a wound. Practically, this could be implemented by estimating the surface normals and scale the 2D image locally based on the angle between the plane and the normal. An alternative approach could be to project the surface onto the inside of a cylinder containing the salmon - which can be unwrapped to form a 2D-surface. In this thesis, a simpler strategy is followed. This is partly due to time constraints, but also due to the fact that all salmon with wounds in the dataset was classified as production. Obtaining an accurate size of the wounds is therefore not of great importance. The important aspect is to detect the wounds regardless of the size. The strategy employed is to project the side facing upwards to one plane, and the bottom to another. For the top plane this involves exclusively the top camera, while on the bottom one must combine the two cameras placed at an angle. The projection itself is quite simple, as the plane of projection is aligned with the Cartesian coordinate system. The projection is therefore done by simply stripping the Y-coordinate.

To determine which of the two planes the pixels are projected onto, a line is drawn between the leftmost and rightmost points of the spline. This works quite well as the spline has a high degree of smoothness, and thus the turning point at the edges will be the best representation for where the curvature bends back on itself. The pixels above the line are projected to the top plane, while the bottom pixels are projected downwards. Only the top camera is considered for the upwards projection, since its perspective should cover the side in its entirety. This keeps noise from the bottom cameras interfering with the top projection. A split is also performed between the bottom cameras, for different reasons. The bottom cameras are adjusted so that the middle of the conveyor belt and towards the camera is in focus. Instead of using the pixels from each camera to effectively double up on the density, a split is performed around $x = 0$ (with some added overlap). This avoids utilizing blurred pixels that are the furthest away, and instead use the pixels of the closest camera. The pixel density is high enough regardless.

Once the projection is done, linear interpolation is performed on the scattered coordinates in the XZ-plane, to obtain an equidistant grid suitable for image analysis. For convenience, the sample density along the X-axis is chosen to be the same as along the Z-axis

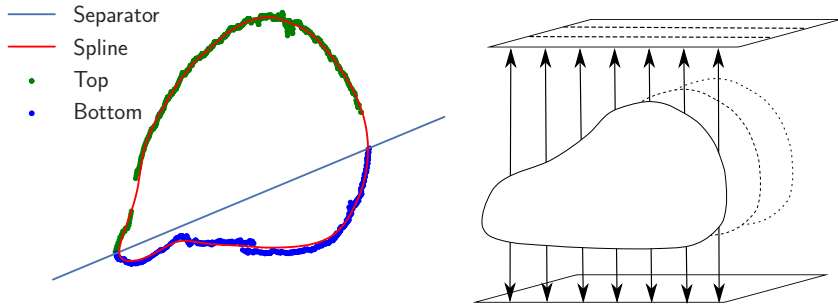


Figure 4.17: The visible coordinates from each side is projected onto a 2D-plane. Left figure displays the splitting method applied to a salmon from the dataset.

so that a re-sampling along the Z-axis is unnecessary. This means that a 1D-interpolation is performed per slice for each projection. This approach is quite fast, but means that the spatial component along the length of the salmon is not utilized. General scattered 2D-interpolation routines are based on a triangulation of all points to decide which pixels to interpolate from. This is dreadfully slow when applied to the number of points present in the 3D-model, even if performed on a section at a time. Alternatives to the 1D-interpolation could be to implement a scattered interpolation on the GPU, once again by utilizing the sorted z-axis. Another approach could be to project the points onto a fixed grid of fine resolution, and use a convolution filter to smooth out missing pixels. The 1D-interpolation has proven to yield acceptable results due to the denseness of the samples, and an effective 2D-interpolating implementation is therefore left to future work. The result of the interpolation and projection from the bottom and top can be seen in figure 4.18, although for two different salmon.

4.6 Color Feature Extraction

In this section, the steps taken to extract features used in the classification of wounds will be detailed. There are two separate images to process for each salmon, as the projection in the previous section extracted two sides. The first step will be to reduce the number of pixels that needs to be processed in a classifier. While the machine learning algorithms used in this thesis is quite fast, classifying all the pixels individually would be too computationally expensive. The strategy is therefore to first reduce the full image to a few regions where the likelihood of a wound existing is higher. This is done by imposing a threshold on the color values, thus excluding regions not wound-like.



Figure 4.18: Salmon 47 (A) and 104 (B) from the dataset, projected onto a 2D plane and interpolated to ensure equidistant pixels. A is projected from the bottom cameras, while B is projected from the top camera. Wounds are indicated by a red circle.

By looking at the color images produced by the projection, the observation was made that the wounds are relatively dark. When looking at a histogram of the RGB values in a wound, one can see that all three are fairly close together. The colors range from a slight red to a brown, with some instances of green and yellow. Simply thresholding based on the red spectrum would not obtain good results, as the entire red spectrum can appear in the rest of the fish, even if the color is not dominated by red. One needs to threshold the redness, but take the other two spectrum into account while doing so. Hue-saturation-luminosity (HSL) is a cylindrical-coordinate representation of the RGB colors. Of these, hue describes the color in terms of an angle, where red, green and blue are spaced furthest apart. A dark brown color will for example mostly be in the red-yellow region of the color wheel. Typical histogram for the RGB and HSL values for wounds can be seen in figure 4.19 and figure 4.20, respectively.

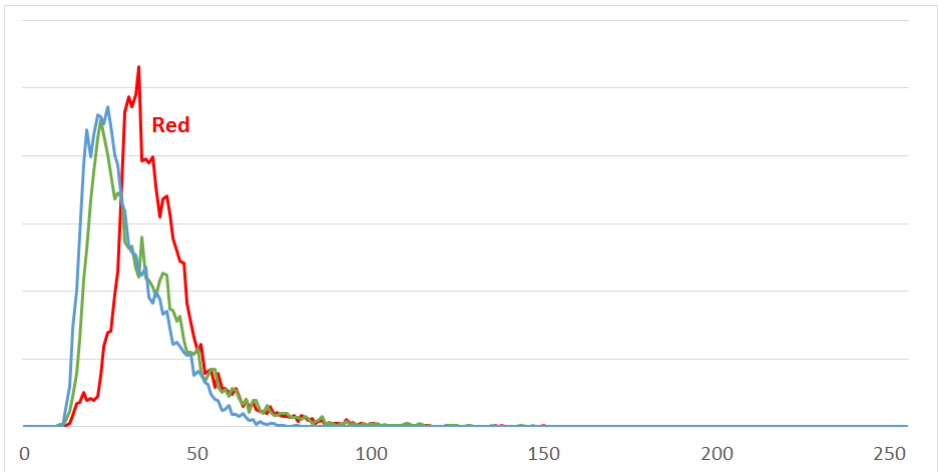


Figure 4.19: Typical histogram for RGB wound pixels. The Y-axis indicates the number of pixels with that value.

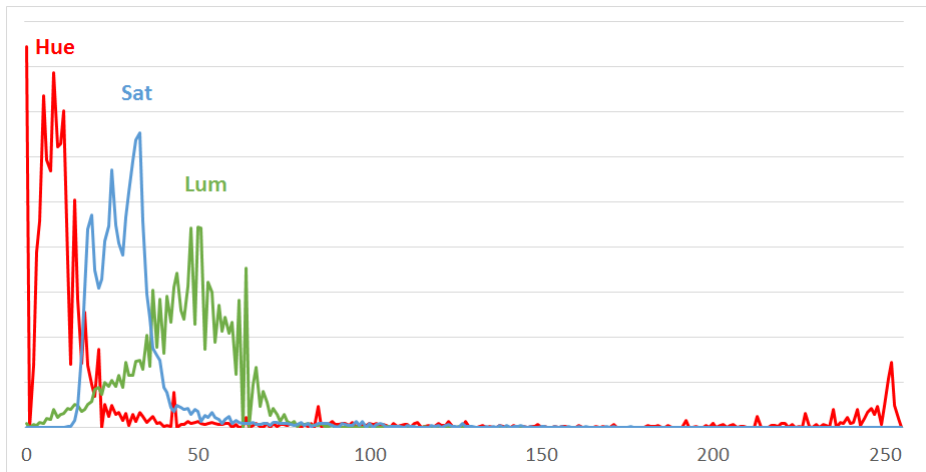


Figure 4.20: Typical histogram for HSL wound pixels. The Y-axis indicates the number of pixels with that value.

Thresholding hue to be in the range 0 to 60 and 230 to 255 yields the result in figure 4.21. The upper range of this threshold is to include the pixels that tends towards the yellow and green hue. An threshold on 0 to 100 for luminosity also gave good results, as it removed some of the brighter spots (i.e a direct reflection). An alternative to using the hue would be to normalize the red spectrum with respect to the the green and blue, seeing that the red spectrum is either equal or dominant relative to the other two for wounds.

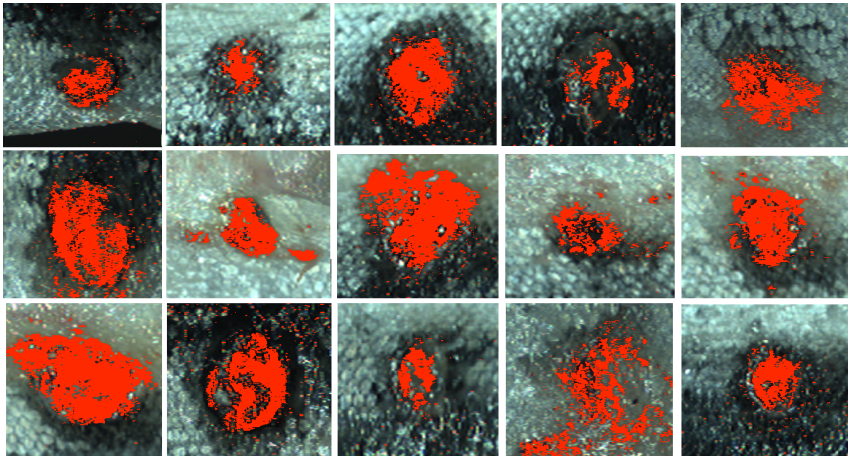


Figure 4.21: Thresholding of the hue and luminosity applied to the image containing all the wounds.

The rest of the salmon typically has scattered values that are within the range, with the occasional exception. The exceptions occur mainly around the fins, head or sometimes the abdominal flap, where sometimes a small clustering of values will occur. One cannot assume that a simple threshold will yield a perfect separation due to reflections and variability of the colors present. Before worrying about the clustered values, let us remove the small scattered points. By considering the pixels within the threshold and binary true, and the values outside the threshold as false, this can be done by using image morphology. Erosion is a morphologic operation where a pixel is set to the minimum of its surrounding pixels, the size of the filter (size of neighborhood) determines how many coarse the removal is. Dilation is the opposite operation, where a point is set to the maximum of its neighborhood. In essence, these two operations either shave off pixels around edges or add pixels around edges, respectively. The erosion is clearly capable of removing the small pixels scattered about. By itself it would also remove the clustered values partially, however. The solution is to perform an open operation, where an erosion is followed by a dilation. This is performed iteratively. The end result is the removal of small regions and protruding features - but the larger objects are kept. The result of this operation on a salmon can be seen in figure 4.23.

Now, these regions must be separated in some manner. This could be done by for example looking at the connectivity. Imagine that the clustered values are close together, but not connected. This would be problematic. To counter this, the convex hull is computed for each of the connected regions. The convex hull for 2D is the smallest polygon that encloses the object without curving inwards. That is, any straight line joining the edges of the polygon does not leave the region. The convex hulls that overlaps are merged, and thus resolves the problem with dis-connectivity of composite regions.

Before proceeding to the analysis of each region, to attempt to determine if it is a wound or not, some areas can be pruned by looking at some features of the convex hulls. Firstly, the regions below a certain area can be pruned. This is primarily to remove the scattered points remaining after the open operation. Secondly, the wounds are typically

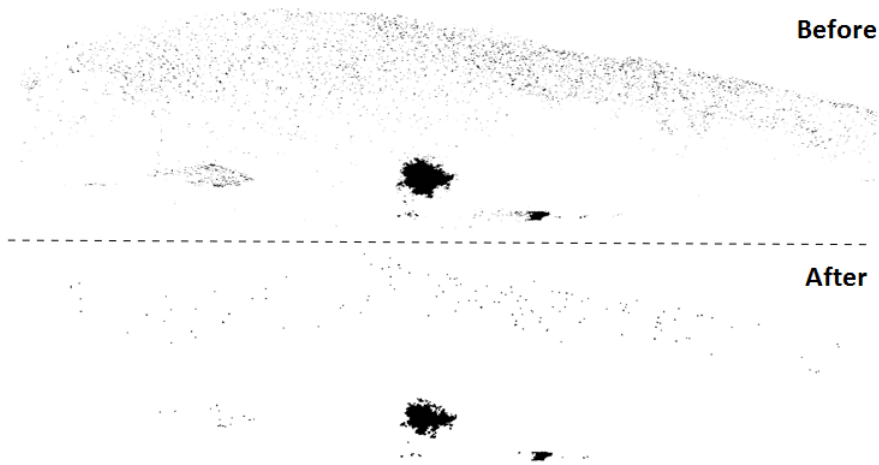


Figure 4.22: Morphological open operation applied to a salmon that has already been thresholded to a binary image.

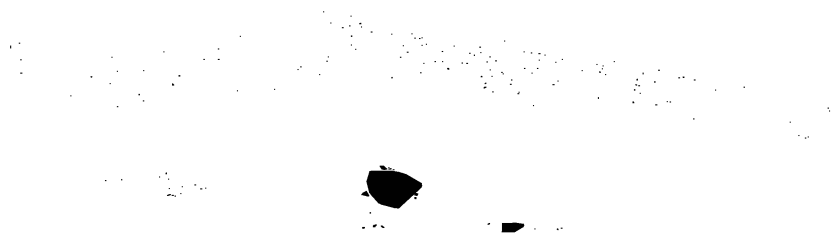


Figure 4.23: The convex hull of each connected region within the image.

fairly circular due to the nature of bacterial growth. Using this, areas that are more elongated along one direction can be pruned. Elongation can for example be represented as the ratio of the largest diameter to the smallest diameter, which is what is done in this thesis. An upper threshold for the elongation ratio is set at 3. Typically, the wounds are much lower - but some margin is factored in to avoid cutting off wounds that are a result of merged convex hulls. The purpose is mainly to remove some of the regions that are sometimes detected along the abdominal flap and along the gills.

For each of the remaining regions, the mean and standard deviation of each color spectrum is extracted (RGB/HSL). Additionally, the relative size of the pixels below threshold in a region and the total size of the image is computed. Unless the wound is recent formed it should have fairly large impact on the surrounding tissue. This can be seen in figure 4.3, where one clearly can see that even the small wounds has an surrounding area that is affected by the bacterial growth. Using the size as a feature is therefore not intended to filter out smaller wounds, but to include a sense of the amount of impact on the salmon. The classification will be done both with and without this feature for completeness.

Chapter 5

Results

The following chapter presents the results obtained by training a classifier using the features outlined earlier. In the following chapter the class containing salmon with deformities is labeled as true, and the superior class as false. Each feature is scaled from -1 to 1, as both classifiers used are sensitive to scaling. Not performing scaling could result in one feature dominating the others.

5.1 Method of Performance Evaluation

To select model parameters and evaluate the classifier performance, one must ensure that the estimates are unbiased. This becomes challenging for such a small data set, as dividing the dataset up in a validation and training set can reduce the performance of the classifier due to the decreased training set. Still, it should be done to provide a pessimistic rate. The dataset is therefore split into 80% training set and 20% validation set in some of the tests. The model parameters for SVM with the RBF kernel, γ and C , is selected by performing 10x10 fold stratified cross-validation on the training set. The optimal combination of γ and C is chosen as the ones that produce the highest average rate. The grid search is performed exponentially using base 2 (2^x) over the range $\log_2 C = [-4, 40]$ and $\log_2 \gamma = [-30, 4]$ with a step size of 2 for both parameters.

The selected parameter combination is used to train a model using the full training set, which is validated on the 20% set. The primary point of doing this is to decouple the selection of model parameters and the validation set, otherwise the accuracy calculated would be unrealistically high as the model parameters are specialized to the entire dataset. To minimize the effect of a few good or bad splits, the process of training and validation is repeated 100 times. This is most likely overkill, but the training process is quite fast due to the small dataset. The prediction rates are finally averaged and used as a performance metric.

Both datasets are unbalanced, which means that an overall rate would favor the class with the most samples. The average rate is calculated by averaging the prediction rate for each of the classes. This is preferred in this case, as the performance of the class with least samples is as important - if not more important - than the class with the most

samples. In this thesis, a high detection rate of the deformities and wounds are preferred. Downgrading a salmon too many is better than allowing salmon of poor quality into the superior class. Ultimately, the downgraded class would be sorted manually to decide the possible applications. The downgraded class consists of a fraction of the total salmon moving through the processing facility, which means that the amount of manual processing is already reduced by singling out poor quality salmon. In each trial, the performance of the best average case is stated through true positive rate and true negative rates, which reflects the rate of correct prediction within each class. The percentage of misclassification per class can from this be calculated by subtracting the true rates from one.

5.2 Detecting Deformities

This section presents the prediction rates of the classifiers trained to detect deformities. The feature vector used throughout this section can be seen below. Trials using subsets of this feature vector was attempted, but delivered inferior results - and will not be presented.

$$\mathbf{x} = [L_{\text{medial}}, L_{\text{contour}}, \beta_{\text{back}}, \beta_{\text{front}}, \mathbf{R}_{\text{max}}, \mathbf{H}_{\text{medial}}]^T \quad (5.1)$$

Where

- L_{medial} : The medial length found by tracing the centers of the largest embedded circles.
- L_{contour} : The length of the contour found by tracing the medial widths along the salmon.
- $\beta_{\text{front}}, \beta_{\text{back}}$: The skewness of the largest embedded circles, calculated from the front and back parts of the salmon. Divided at the middle of the medial length.
- \mathbf{R}_{max} : The radius of the largest embedded circles (widths), extracted at five equidistant points along the medial length.
- $\mathbf{H}_{\text{medial}}$: The medial heights, extracted at five equidistant points along the medial length.

5.2.1 Deformity Detection Using C-SVM with RBF Kernel

The dataset is both unbalanced in numbers, since the superior class consist of 45 samples and the downgraded class 25 samples. To counter this while training a model, one typically ensures that the misclassification penalty evens this balance out. This can be done by setting the misclassification cost per class so that it fulfills $N_1/N_2 = C_2/C_1$, where N is the class size and C the misclassification cost attributed to the class.

The above approach is correct if one wants to maximize the overall prediction rate. A problem with this approach is that one then trains a model for the relative sizes of the classes present in the training set, which might not reflect the relative class frequency in the real world. In fact, the fraction of the various classes can have much variation depending on time of year and which facility it originates from. Since the dataset at hand does not

reflect the a-priori probability, which is an unknown quantity anyway, one cannot train a model so that the classes are completely equally weighted anyway. We do know, however, that the downgraded class will have a much higher misclassification cost attributed with it to ensure that the superior class is free from deformities. For this reason, the cost for the class with deformities is set to 10 times the superior class. This causes a drop in the correctness of the samples labeled superior, but yields a better true positive rate. In the future, this can be revised when the training set becomes larger - as a few outliers has less of an affect on overall performance.

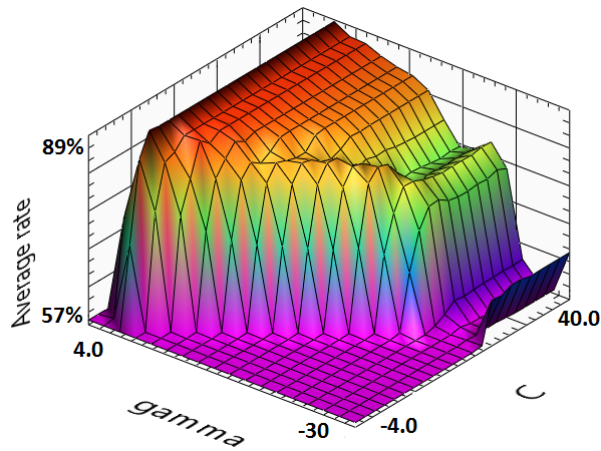


Figure 5.1: Overall prediction rates from an exponential grid-search for the parameters γ and C using 10x10 stratified cross-validation.

Figure 5.1 shows examples of the result obtained by performing a grid-search over the parameters using a 10x10 cross-validation. The prediction rates obtained are 89.1% true positive and 88.4% true negative. The observation was made, by re-training with new random splits multiple times, that some of the salmon in the superior set was classified consistently as production/ordinary. Upon inspection of the salmon in question, traits that might be considered similar to those in the downgraded classes was found. This stems from the fact that the classes are not based on hard criteria, but a rough description of what is acceptable quality and not. The boundary between superior and lower grades is therefore not an exact science. With a large dataset, these outliers will be placed less emphasis on - as the majority of the class has a good separation. In interest of seeing if the detection rate of the deformed salmon can be increased, the salmon in question were removed from the superior set. This of course skews the result, but feels justified as these samples looked very similar to the milder cases in ordinary/production.



Figure 5.2: Fish10 from the superior dataset, where the gutting makes it appear as deformed.

One particular example which should be pointed out is salmon 10 in figure 5.2, where the gutting is irregular towards the head. The cut is deeper, causing the front part of the abdominal flap to protrude more than usual. The features and classifier mistakes this for a wider front, which causes a misclassification. The features has not been developed to handle this, and must be handled in further work. Possibly by using the volume of the front part instead of the widths as a feature.

In addition to salmon 10, number 11 and 19 were also removed due to having a slight humpback. These can be seen in the online viewer. When these three samples were removed, the true positive prediction class (deformities) increased significantly. This occurred because the classifier has an easier time finding a separating hyperplane in the high-dimensional space given less outliers. This could also be observed through the training times for the classifier, which was reduced significantly after removing these three samples. Running a new 10x10 stratified cross-validation yields a true positive rate of 95.2% and true negative rate of 94.5%. Clearly an improvement. The new parameter grid-search results can be seen in 5.3.

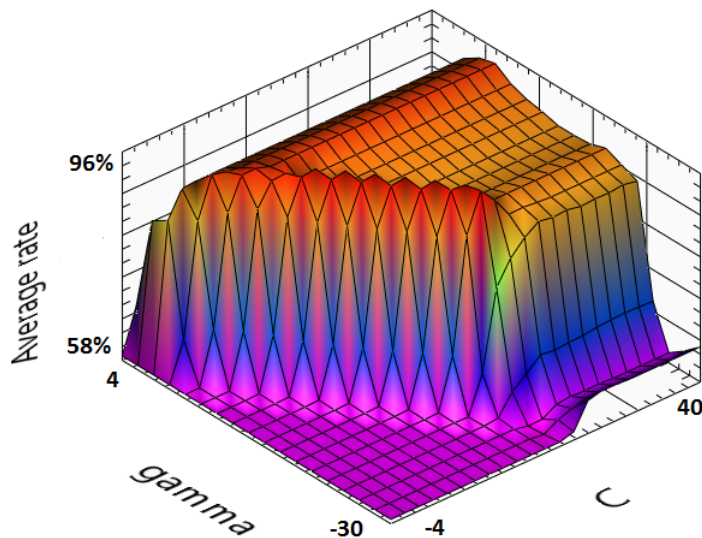


Figure 5.3: The parameter grid-search after removing three salmon from the superior class.

Separate validation set 80/20 and 90/10

Now, the same experiment is performed with a decoupled model training set and completely separate validation set. The dataset is split 80/20, and cross-validation is performed repeatedly on the training set to select the best parameters on average. Those parameters are then used to train a model using the full training set, and validated on the smaller validation set. The expectation is that the prediction rates will drop, both because the model parameter selection and validation set is decoupled, but also due to the reduced size of the training set. The obtained prediction rates are 78.3% and 94.4% for true positive and true negative, respectively. The class consisting of the least samples is hit the hardest, which comes as no surprise seeing as that class was quite small to begin with.

Attempting to increase the weighting of that class beyond 10 to attempt to skew the results towards a higher true positive rate yields the following results. Increasing the weight to 100, yields slight increase in true positive rate to 81,5% and true negative remains at 94.4%. The worry is that the data set is too small to be split into multiple parts, getting a few bad splits will reduce the prediction rates significantly. Re-running the experiment with a 90% training set and 10% validation also only yielded a slight increase. The rates increased to 82.9% and 94.6%, true and false positive rates respectively. Performing an extended grid-search with a reduced resolution gave the grid-search results in figure 5.4. The step size used was set at 8, as even this resolution took several hours to complete. The left figure shows the same area as previously, just a bit extended. The figure to the right shows the curvature of the grid-search past the cutoff point of C. This proves that increasing the value of C, which is where the previous grid-search cut off the plateau, does not increase the rates.

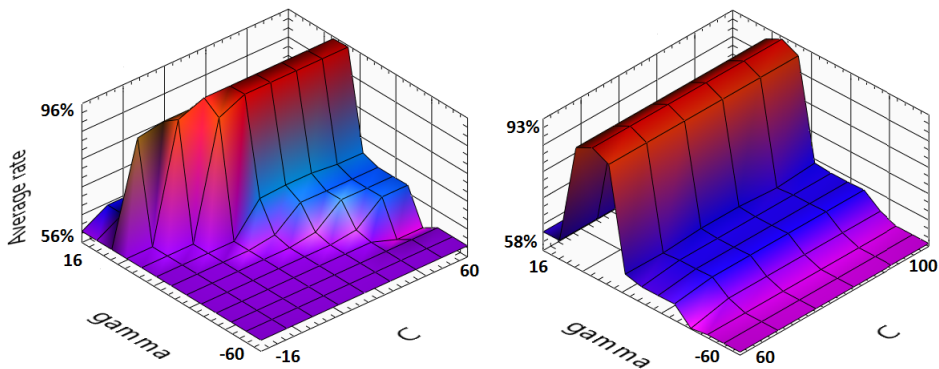


Figure 5.4: Performing a wide-area grid search, to ensure that neighbouring regions does not provide better results. The ranges are $\log_2 C = [-16, 60]$ and $\log_2 \gamma = [-60, 16]$ for the left figure, and $\log_2 C = [60, 100]$ and $\log_2 \gamma = [-60, 16]$ for the right figure.

While the results obtained so far are acceptable, some explanation for the much lower rate was searched for. Performing the grid-search with the same range as originally, but with a finer grid yields the resulting parameter-space in figure 5.5. Looking at this result, it appears that the parameter combinations providing the highest prediction rates are placed at the very edge of the region that yields acceptable rates. The optimal model parameters

are simply chosen as the maximizer of the average predicted rate, which would select these locations. This does not seem very robust in face of a changing training set, such as when doing the 80/20 or 90/10 splits. This is because a changing training set might shift the peak away from the view, which would cause the result to be very suboptimal on the training set.

Instead, the model parameter selection should select a value that provides a high prediction rate, while being robustly placed. There are many ways this could be implemented. One approach could be to make the parameter combinations prediction rates be dependent on the surrounding parameter combinations, similar to a 2D-filter in image processing. Instead of implementing a general approach, we will simply pick a value manually which has sufficient surrounding support. The parameters picked are $\log_2 C = 23$ and $\log_2 \gamma = -8$, as these are placed on the highest ridge interior to the plateau. Running a 90/10 split for this using the same amount of iterations as before (100), yields a true positive rate of 86.2% and true negative rate of 92.5%. One can immediately see that choosing the safe parameters yielded a better overall result, even though it had a suboptimal value on the grid-search. This obviously stems from the fact that the parameter space is skewed somewhat under some of the random splits, causing the rate to plummet for the entire validation set. Using the optimal parameters calculated using cross validation for the entire dataset ($\log_2 C = 16, \log_2 \gamma = -12$), the 90/10 split yields 94.4% true positive and 94.3% true negative. These rates are biased, as the model parameters are tailored to the dataset. The maximum, stable, rate most likely lies somewhere between the two results.

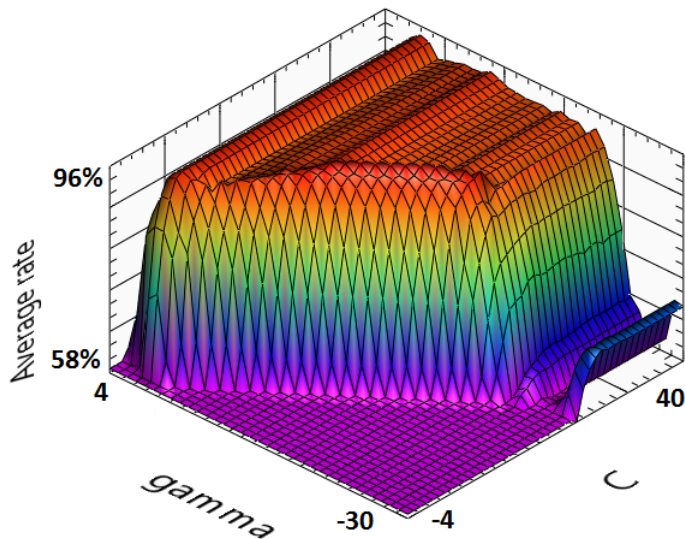


Figure 5.5: A grid-search for the same range as figure 5.1, but with a smaller step size.

It should also be said, that with a larger dataset the optimal parameter would most likely be more robust in a grid search. In this case however, with an dataset of this size

that has a high degree of variability, it causes problems. Especially the samples of the true class (deformities) seeing as they are assigned a higher misclassification penalty.

Leave-one-out cross-validation

Leave-one-out cross-validation is not afflicted with a bad split, and utilizes the dataset to the maximum - as it uses every single element once as a validation sample. First, the LOO cross-validation is performed on the safe values of $\log_2 C = 23$ and $\log_2 \gamma = -8$ from before. This yields the rates 86.3% true positive and 92.7% true negative.

Running the leave-one-out cross-validation with the biased parameter values of $\log_2 C = 16$ and $\log_2 \gamma = -12$ yields the following rates. True positive 95.5% and true negative 95.1%. This is from the parameter combination on top of the ridge, which represents an optimistic separation rate.

The LOO cross-validation yields slightly better results due to having more samples available for each training of the model. These results were also obtained using an added misclassification weight for the true class of 100. Higher weights than this does not improve the rates.

5.2.2 Deformity Detection Using Nearest Neighbor Classifier (NN)

The results will now be validated by using a different machine learning algorithm, the nearest neighbor classifier. This classifier does not have the same good generalization properties as SVM when the size of the feature vector increases due to the curse of dimensionality, but can in many cases yields good results. For datasets that are small in comparison to the feature size, the nearest neighbor classifier is often considered along with SVM as the best choice. The distance measure used is the Euclidean norm. Running a leave-one-out classification using NN yields a true positive rate of 85.2% and true positive of 97.5%. As we can see, the true positive rate is very high. This is partly due to not being able to weight the true class more than the false class, both to make up for the size difference and to skew the result towards the detection of deformities. One can for example add a weight to the Euclidean distance depending on the class to skew the result towards a positive rate. Alternatively, one can subsample the larger dataset to provide an even comparison. Regardless, the results obtained here clearly indicate that the result obtained previous using SVM are largely correct.

Using a K-Nearest neighbor classifier (KNN) with $K > 1$ did not yield better results. Most likely because the density in the feature space is already low, which means that there might not be enough clustering of the two classes.

5.3 Wound Detection

This section deals with the detection of wounds, which is treated through a separate classifier. The wounds has less variability than the deformations, and we can therefore start out with a few features, and add more features as necessary. For the deformation this was not possible, as the diversity of the samples required a certain number of features to be anywhere near acceptable rates. The samples that has wounds are assigned the true class

label, indicating a downgraded salmon. The number of wounds present in the dataset is 15. Due to the smaller size of this dataset, decoupling the parameter selection and validation set will not be performed. A regular repeated cross-validation should give a good indication of the attainable rates, albeit a bit optimistic. The number of regions detected that are not wounds is 125, which makes up the false class. The dataset is therefore unbalanced. The false regions are obtained by performing the thresholding and pruning as outlined in section 4.6 on all the salmon in the dataset without wounds. That is, the salmon in production/ordinary without wounds are also used in obtaining the data for the false class.

5.3.1 Detection of Wounds Using SVM with RBF Kernel

First, the SVM classifier will be used once again with the RBF kernel. An 8-fold cross-validation will be performed, which means that only one validation set contains a single sample from the wound-class. The 8-fold cross-validation is repeated 50 times for each trial, to minimize the effect of bad splits of the dataset. All models in this section are trained using class weights only to make up for the difference in size ($N_1/N_2 = C_2/C_1$), without any additional added weight as done for the deformity detection.

Mean RGB/HSL values

First, the classification will be done using mean RGB values. The feature-vector is thus as follows.

$$\mathbf{x} = [\mu_R, \mu_G, \mu_B]^T \quad (5.2)$$

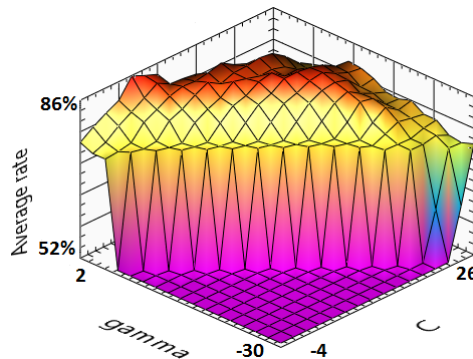


Figure 5.6: Parameter grid search using 50x8 cross-validation for mean RGB features.

The per class prediction rates for the best average rate are 89.5% true positive and 82.2% true negative. Next, cross-validation is repeated using the HSL representation.

$$\mathbf{x} = [\mu_H, \mu_S, \mu_L]^T \quad (5.3)$$

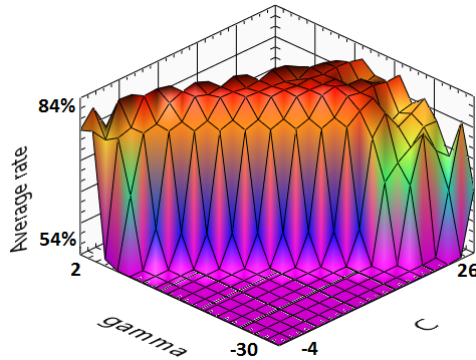


Figure 5.7: Parameter grid search using 50x8 cross-validation for mean HSL features.

The prediction rates for each of the classes are 92.4% true positive, 73.7% true negative. We can see that the wound detection for both RGB/HSL is fairly successful, but many regions are falsely detected as wounds, especially for HSL.

Mean RGB/HSL values and standard deviations

The standard deviation adds a description of how spread the color values are within each region. A wound should dominate within the area, causing less spread in the color values.

$$\mathbf{x} = [\mu_R, \mu_G, \mu_B, \sigma_R, \sigma_G, \sigma_B]^T \quad (5.4)$$

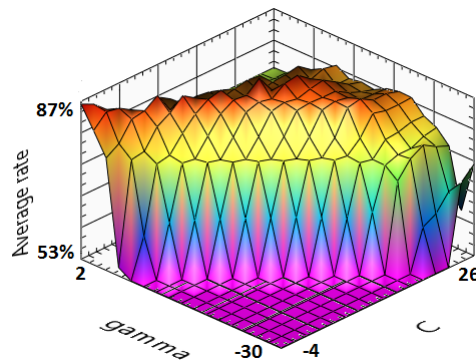


Figure 5.8: Parameter grid search using 50x8 cross-validation for mean and standard deviation RGB features.

The per-class rates are 98.1% true positive and 74.3% true negative. The wound detection has improved, but comes with a number of misclassified regions that are not wounds.

Once again, cross-validation is repeated with standard deviations using the HSL representation.

$$\mathbf{x} = [\mu_H, \mu_S, \mu_L, \sigma_H, \sigma_S, \sigma_L]^T \tag{5.5}$$

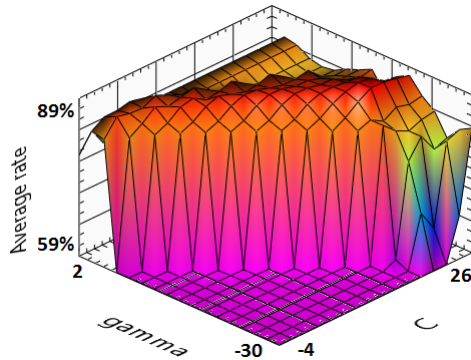


Figure 5.9: Parameter grid search using 50x8 cross-validation for mean and standard deviation HSL features.

The result using HSL is less skewed towards the true class, with 93.0% true positive and 83.7% true negative. This causes the average rate to increase.

Mean RGB/HSL values and standard deviations with relative size

The relative size of the regions compared to the total size of the salmon is added as an additional feature.

$$\mathbf{x} = [\mu_R, \mu_G, \mu_B, \sigma_R, \sigma_G, \sigma_B, \frac{N_{\text{wound}}}{N_{\text{total}}}]^T \tag{5.6}$$

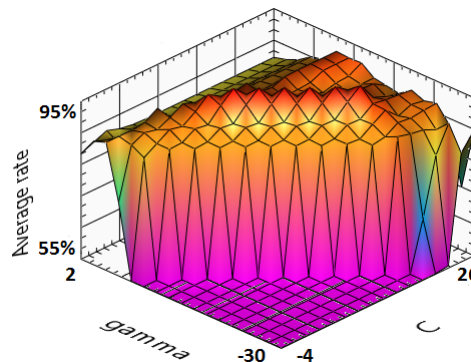


Figure 5.10: Parameter grid search using 50x8 cross-validation for mean, standard deviation and relative size (RGB).

Adding the size parameter results in a general increase in the rates to 95.6% true positive and 92.9% true negative. And now, for the HSL representation.

$$\mathbf{x} = [\mu_H, \mu_S, \mu_L, \sigma_H, \sigma_S, \sigma_L, \frac{N_{\text{wound}}}{N_{\text{total}}}]^T \quad (5.7)$$

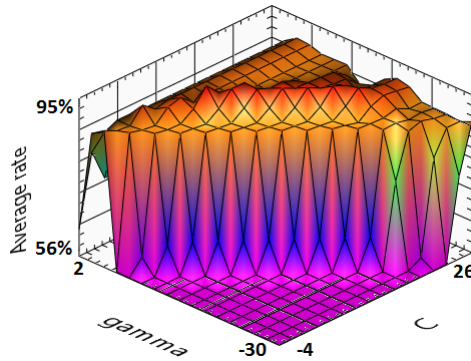


Figure 5.11: Parameter grid search using 50x8 cross-validation for mean, standard deviation and relative size (HSL).

The prediction rates are 97.8% true positive and 89.8% true negative. Due to the high amounts of repetitions per cross-validation, these results are stable even if the cross-validation is repeated. Three of the salmon with wounds have much higher misclassification counts than the rest. These are number 48, 88 and 103. An image of these salmon and their respective wounds can be seen in figure 5.12. All three are fairly small, but the most likely reason for being misclassified is the lack of a redness. Salmon 49 has an almost black area without any redness, which is rather tricky to separate from the black areas on the black. Salmon 88 has redness in the wound, but the size is fairly small. Salmon 103 has the probably most serious misclassification, due to the size of the wound. This wound has more of a milky quality than red, and is for that reason misclassified. This could perhaps be redeemed through a larger dataset with more wounds of this type.



Figure 5.12: Salmon 48, 88 and 103 from top to bottom. These are the salmon with wounds that are most commonly misclassified.

Due to the strongly unbalanced nature of the dataset, the nearest neighbor classifier did not produce acceptable results using the Euclidean distance norm. A LOO was also performed for the final feature set for RGB and HSL, using an conservative parameter combination of $\log_2 C = 23$ and $\log_2 \gamma = -8$. This is the same as the safe parameters for the deformity classifier, as they seemed to be in a safe location for the wounds as well. Note that this dataset is even smaller than the deformity dataset, and the 90/10 split was therefore not performed as the class containing the wounds is simply too small. Also, as was pointed out - the method of selecting the optimal parameter combination does not work if the parameter space changes significantly for some data splits. The LOO parameters placed sub-optimally, and is not expected to be the true rate, but is supplied as a lower bound on the rates. This is better than simply stating the optimistic rates, with parameters tailored to the dataset. The rates for the LOO cross-validation can be found in the table of results.

5.4 Summary of Results

The main results found for both the deformities and wounds will now be summarized in tables, for convenience.

Table 5.1: Deformity Classification Results

Description	Method	Training Set %	Tr. Positive	Tr. Negative
Full dataset	10x10 CV	100%	89.1%	88.4%
Reduced dataset	10x10 CV	100%	95.2%	94.5%
First Split	10x10 CV	80%	78.3%	94.4%
$C_1 = 100$	10x10 CV	80%	81.5%	94.4%
$C_1 = 100$	10x10 CV	90%	82.9%	94.6%
Using safe parameters	10x10 CV	90%	86.2%	92.5%
Using biased parameters	10x10 CV	90%	94.4%	94.3%
Using safe parameters	LOO CV	90%	86.3%	92.7%
Using biased parameters	LOO CV	90%	95.5%	95.1%
Nearest Neighbor	LOO CV	100%	85.2%	97.5%

Table 5.2: Wound Classification Results

Color Space	Features	Tr. Positive	Tr. Negative
RGB	μ	89.5%	82.2%
HSL	μ	92.4%	73.7%
RGB	μ, σ	98.1%	74.3%
HSL	μ, σ	93.0%	83.7%
RGB	$\mu, \sigma, N_w/N_T$	95.6%	92.9%
HSL	$\mu, \sigma, N_w/N_T$	97.8%	89.8%
RGB LOO (suboptimal)	$\mu, \sigma, N_w/N_T$	85.7%	96.8%
HSL LOO (suboptimal)	$\mu, \sigma, N_w/N_T$	80.0%	93.9%

Chapter 6

Discussion

This chapter briefly discusses aspects of the thesis, with focus on the general approach, methods employed, and the results.

6.1 Methodology

This section discusses the general approach and methods used in this thesis. Throughout this thesis, many choices have been made regarding which method to use. Some choices were done experimentally by implementing the alternatives, and weighing them directly. Other choices had to be made without attempting the alternatives, due to time constraints. As this thesis consists of many individual boxes that need to be ticked to form a full, functional system, simplicity was often favored over complex solutions. The simpler solutions also often made sense from a performance perspective.

6.1.1 Line Scanning

The decision of using the principal approach of line scanning preceded this thesis, and as such the discussion of which technology to use has not been an integral part of the work itself. The primary considerations for choosing this technology over other technologies, are cost, simplicity and applicability. The line scanner has no need for specialized sensors or equipment, only off-the-shelf cameras and lasers, this drives the cost down to a reasonable level. One of the primary objectives of this thesis is to provide a cheaper alternative to commercial systems, after all. The simplicity of the method was an important consideration, as the system needed to be implementable without having a doctorate in the respective field. The method also needed to be applicable to objects moving on a conveyor belt. Many methods run at a slower acquisition rate, but with a wider field of view. That would not be applicable to capture the underside of the salmon, as the space between the conveyor belts is tight. Additionally, color information is needed to determine the quality in many cases. Using technologies with other sensor types would need to have cameras in addition to the range sensors, to capture this information. By switching between the color and laser, both

is obtained - and in the same coordinate system. Other technologies does not tick these boxes as well as the line scanning approach.

6.1.2 Camera Calibration

The calibration routine uses state of the art algorithms. The implementation is therefore founded in solid theory. The original idea was to use a calibration object in which multiple cameras could be calibrated simultaneously. This was both the intention for the purpose of simplicity, but also to ensure that the coordinate space from each of the cameras coincided. Moving the calibration object about introduces a degree of human error, although minimized by having a stand with fixed points. The misalignment correction was introduced to mitigate this problem. While it does so successfully, it might be an idea to revise the calibration object such that it works with the original goals in mind. One approach could be to have a larger calibration object in the distance of the conveyor belt, which then could be moved through the laser-line by the conveyor belt. This would not only introduce the possibility of calibrating all cameras at the same time, but also improve on the number of point correspondences. The current calibration object yields 7 coordinate correspondences in the XY-plane, which gives 14 points. The projective transformation has 8 degrees of freedom. The problem *is* overdetermined, but more could potentially improve accuracy.

In this thesis, speed calibration is performed by scanning an object of known size and calculating the correct spacing of frames in relation to frame rate. An automatic routine for this was not developed, as future applications will undoubtedly utilize a rotary encoder to be able to deal with changing speeds of the conveyor belt. As the rotary encoders typically can be calibrated separately, such a feature is unnecessary.

Accuracy

Not much discussion has been performed throughout this thesis regarding the accuracy of the calibration routine. While the accuracy of the calibration routine itself has been measured to be in range of ± 0.02 to 0.04 cm, this accuracy will not be the case for the point cloud. While it might hold true for certain surfaces, it most certainly is not for the salmon. The curvature and reflective properties of the salmon causes variations in the laser line that cannot be calibrated away. This is due to optical issues, and will largely be improved by reintroducing the polarization filters - but with a stronger laser.

6.1.3 Laser Extraction

Extracting the center of the laser-line using a weighted first moment works very well. It is possible however, that other strategies could provide a better result. One thought is to use higher order moments, or ensuring that the laser-line the calculation is performed on consists of connected pixels, i.e outliers are ignored. One can also consider other image processing techniques. As the image is already on a GPU, the cost of performing additional image operations is very low.

6.2 Features

Some discussion of the features used in the classification follows. A general remark is that one needs to be careful when evaluating the effectiveness of each individual feature for small datasets. A higher rate might simply mean that the feature is more tailored to the dataset itself.

6.2.1 Symmetry

The skewness and medial contour length attempt to capture the asymmetry in the salmon. The salmon are not completely symmetrical about the longitudinal direction however, as the back-part is smaller. What these features are looking for is a lack of a streamlined shape. This means that one cannot simply deduce the symmetry from mapping the front half onto the back half. Experiments were made where this was attempted by scaling and re-sampling one half onto the other, but initial results were poor and further attempts were stopped. The basic idea is maybe worth pursuing in the future however, perhaps also by introducing concepts from fluid dynamics to describe how streamlined the salmon is. The natural shape of salmon should minimize the resistance in water. Using the medial axis, one could straighten the salmon to compensate for pose, and run a simplified simulation. This might not be feasible for gutted salmon, however.

Neither of skewness and medial contour take the placement of the asymmetry into account. Having such a feature would require some kind of spatial component in addition to the symmetry feature, which would increase the number of features drastically. Due to the small amount of samples, the symmetry needed to be represented through as few degrees of freedom as possible.

6.2.2 Medial Axis

Experiments with and without features based on the medial axis resulted in a lower prediction rate without the medial axis. Based on the dataset available it seems to have an advantage. Tests were performed replacing the medial width, height and length with its corresponding direct measurements. Collectively this reduced the prediction rate to 74% true positive. If the features based on the medial axis generalize to larger datasets remains to be seen. During the testing of removing and combining various geometric features, the medial contour length was found to be the absolute best single feature in the feature vector for describing the deformities. It seems that the length around the contour describes deformities well. This intuitively stems from the fact that lack of a streamlined shape causes unnecessary curvature, which causes the contour length to increase.

Although the medial axis took much time to implement correctly, especially considering the spline interpolation necessary. It was completely worth it. One of the primary goals of the thesis was to compensate for the inherent variation present in biological subjects. The medial axis causes the size metrics used to be largely invariant to the pose and curl of the salmon, which would otherwise be hard to do robustly.

6.2.3 Reflective Properties

Originally, the intention was to use reflective properties of the laser line in the detection of wounds. Features that could be considered in this regard is the sum of intensities per column, the intensity at a certain offset and the width of the most intense of the laserline. Unfortunately, removing the polarizing filter caused these to be unstable, and was left unused. Figure 6.1 shows an instance where the scatter value (offset) would still be usable - which illustrates the usefulness of these reflective properties. The wound is clearly visible as a bright spot. Figure 6.2 shows an instance where it would not be so useful, and might in fact do more harm than good. It seems that the offset, as imaged here, suffers more on the bottom of the salmon due to the angle of the cameras. The beamwidths on the other hand, suffered more on top due to the direct glare towards the top camera causing the width of the beam to seem much wider. Both of these reflective properties were found to be very useful when the polarization filter was used, which is why it is mentioned here.



Figure 6.1: Scatter intensity for an offset of 8 for salmon 102 (top).

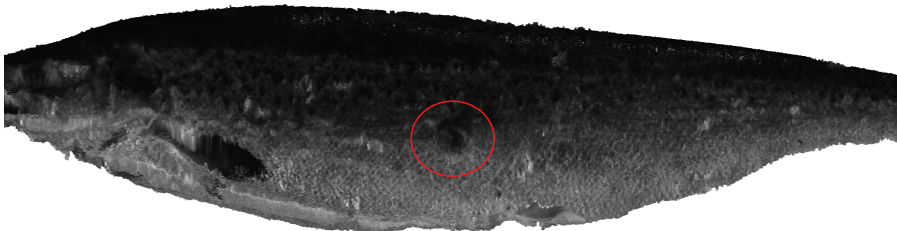


Figure 6.2: Scatter intensity for an offset of 8 for salmon 90 (bottom).

6.3 Attained Prediction Rates

Generally, the obtained prediction rates are quite satisfactory. Caution has been advised several times however. The most optimistic rates are obtained when the model parameters are not decoupled from the validation set. What this essentially does is leave two degrees of freedom to improve the performance of the classifier to known samples. The C parameter is adjusted to optimally account for outliers in the dataset, and γ adjusts how much influence a single training sample has. The cross-validation decouples the training of the model from the validation set, but the parameters are selected as the overall result

of the entire cross validation. That is, for each parameter combination - a cross-validation is performed on the entire available dataset.

The 80/20 and 90/10 splits performed for the deformation classification yields more correct rates - but are most likely a bit pessimistic due to the small size of the dataset. Due to the small size of the dataset, there was also problems with the training set occasionally selecting an optimal parameter combination that performed poorly for the validation set. This will be much more stable with a larger dataset, as the number of outliers and clustering will be more settled. The proof for this is clear, as selecting a suboptimal parameter combination for the full dataset yielded better results than selecting the optimal parameter combination as the maximizer of the average rate at each repeated 80/20 split.

6.3.1 Deformity Detection

The performance can be thought to lie somewhere between the biased parameters and safe non-optimal parameters. The true positive rate therefore lies somewhere between 95.5% and 86.3%. The true negative rate lies somewhere between 95.1% and 92.7%. The results are satisfactory, but it remains to see if the same results will be obtained for a larger dataset.

6.3.2 Wound Detection

The results obtained from the wound detection are also quite good. The attained rates seem to be the best for RGB. As with the deformity detection, the rates lie somewhere between the optimistic and pessimistic rates. This means that the true positive lies somewhere between 95.6% and 85.7%. The true negative lies between 92.9% and 96.8%. The pessimistic rates yielded a true negative that is higher than the optimistic rates. This comes from the fact that the optimistic parameter combination is chosen as the average of the true positive and true negative rates. The choice of safe parameters simply happened to favor the true negative rate. If one truly wants to maximize the true positive rate, one should perhaps place extra emphasis on the true negative rate. As mentioned previously, the overall rate, which is the maximizer of the prediction rate of all samples regardless of class - cannot be used for an unbalanced dataset. The result would be skewed towards the larger class every time. The rates found also needs to be verified on a larger dataset before one can be absolutely sure of the attained rates are correct.

The results for wound detection are generally also satisfactory, but can definitiely be improved on. One aspect that was not used in the rejection of wounds is the placement. One can most likely improve the rate significantly by ignoring the head and the abdominal flap. That is, by either removing these regions previous to the classification - or including some distance metric in the feature vector.

6.3.3 Evaluation of 3D Machine Vision in Quality Control

While performing quality control using 3D machine vision generally introduces more complexity into the extraction of features and system as a whole, it is also an enabling technology. It is an important step along the way to replace human workers completely. While the features extracted from the point cloud in this thesis might not be the features that end up being optimal in the long run, the information is there. The first step to replace

the human component in any machine vision task, is to provide the same sensory input as humans. Humans have stereoscopic vision, which enables us to perceive depth. The line-scanner is just an alternative route to obtain this information. Having full coordinate coverage around the salmon means that one can obtain the same information as a human picking up the object to study its underside.

Having the full coordinate profile enables the use of methods that would otherwise be inaccessible. The fin removal procedure would not be possible to the same degree of confidence without knowing the curvature on both sides. The medial length, tracing the center of the salmon to account for the pose and curvature would not be possible without depth information.

It is when this system is implemented in a factory, scanning several thousand salmon per day, one truly can explore the possibilities with respect to different features and classifiers. One benefit of this machine vision system is that it is non-intrusive. It can be implemented in a factory and then gradually take over as the machine learning algorithms has been improved to the point where it can take over.

6.4 Additional Remarks

6.4.1 Performance

While the performance of the acquisition-part of the thesis has been covered briefly, the performance of the analysis and feature extraction has not been covered. While the methods employed has partially been selected with respect to their computational complexity, the implementation itself has not been optimized to a large degree. The features has been a result of experimentation and prototyping, and pre-maturely optimizing them was therefore avoided. The spline re-sampling routine was perhaps the component that had the worst performance. It was found however, that this was largely due to the interface used in LabVIEW to communicate with Matlab (ActiveX). This should therefore not be a problem once it is implemented in LabVIEW or C++. There are also many parameters that can be tweaked to make the analysis quicker. Reducing the spline sample density will for example cause the Voronoi-diagram to be computed quicker. The current implementation samples the boundary quite densely. Another possibility is to not perform the full analysis for every frame, as has been done here. The features should not require the amount of density along the length the salmon for them to be effective.

The components of the classification was implemented individually by reading and writing data to disk, to avoid recomputing everything every time a change was made. For this reason, it is hard to provide a performance metric of the implementation of feature extraction and analysis.

6.4.2 Robustness

The robustness of the methods used has not been discussed to a great degree. One benefit of developing this system without a polarization filter, which caused more noise, is that the approach should be very robust when applied to a system with polarization filters. One aspect that was found to be problematic was once again the spline. The system is

quite sensitive to the spline smoothness. Striking a balance between a spline that rejects unwanted features and optical noise, while being flexible enough to capture the edge was difficult. Especially with respect to the abdominal flap, which is much more protruding than the other features (fins excluded). The spline representation is also not very general considering that a polar representation is used. For this reason, implementing re-sampling through another method should perhaps be considered. Previously, moving least squares (MLS) was suggested as an alternative if a fast implementation could be made, perhaps on the GPU.

Another aspect of this thesis is the application of automatic quality control to gutted salmon. Applying quality control to un-gutted salmon would reduce the variability. In fact, the polar representation used for the spline was originally designed to be used for un-gutted salmon, where the salmon is mostly elliptical in shape. A benefit of considering gutted salmon, is that applying the same techniques to un-gutted salmons should be robust. This would not necessarily be the case the other way around.

Chapter 7

Conclusion

In this thesis, a complete machine vision system for 3D-imaging has been built and integrated for the purpose of quality control of Atlantic salmon. The system is build using off-the-shelf hardware, and has been integrated using no external proprietary tools. The software for the acquisition was implemented with real-time restrictions in mind. The result is thus an affordable solution, which can be deployed in an industrial environment without major investments. An experiment was then performed on Atlantic salmon of different quality classes. The obtained data was then used to develop descriptors that capture enough information to separate out lower classes of Atlantic salmon based on appearance.

Based on the results presented in chapter 5 and the following discussion in Chapter 6, the main conclusion of this thesis is that 3D machine vision has clear benefits over one-sided imaging. While the 2D counterpart is much cheaper in terms of computational complexity, offloading the image processing to a GPU makes the increased workload manageable. Based on the available dataset, the geometric features developed attains a high level of separation between the superior and ordinary/production classes. Although the size of the dataset is at the limit of what is advisable, the results have been verified using known methods for classification accuracy estimation. The classification has primarily been performed using Support Vector Machines (SVM) with a nonlinear kernel function (RBF), which is currently one of the most popular classifiers when the dataset is small in relation to the dimensionality of the features. The geometric classification results has also been verified using nearest-neighbor classification, which provided slightly worse - but comparable results. Verifying with a second machine learning algorithm reinforces the belief that these results will generalize well to a real-world situation.

The additional effort put into the use of the medial axis, which possesses a skeleton-like approximation, was found to be well worth the effort. Features based on the medial axis are invariant to both the flex and pose of the salmon. The belief is therefore that the medial axis helped counteract the impact of the inherent variability of biological matter. One should be careful when concluding the effectiveness of features based on a small dataset, as in this thesis however. When working with a small dataset, one always runs the risk of specializing the selected features to the available data. In this thesis, the luxury of having enough available data to set aside a dataset not used for development of the features was not feasible. Determining the effectiveness of the current features and developing better

features is therefore left to further work, when a larger dataset is obtained. Based on the preliminary results found during this thesis however, the future looks optimistic.

The work performed regarding wound detection proves that it is possible to detect wounds on both sides using the color information alone. A series of simple image processing steps to segment possible wounds, followed by a classifier, proved to be a good approach. Similar to the detection of deformities, the wound detection needs a larger dataset to ultimately conclude on the effectiveness, as the color of wounds can vary greatly depending on the depth and age of the wound.

To the authors knowledge, 3D imaging has not been utilized with full 360° coverage in automatic quality control in the aquatic food industries. This statement is based on recent review papers. The 360° 3D technology enables the quality control system not only to see what the human operators see today, but also the side that is obscured from view. This has as a major impact on wound detection, as the entirety of the salmon is visible. An algorithm was for example developed to remove the effect of the most protruding fins on the shape analysis. This would not be possible with the same level confidence, from a single perspective - as the curvature on both sides of the fin would not be known. Additionally, having a full and detailed colored 3D model of the fish, coupled with deformation-invariant representation is an enabling technology. It enables practical solutions for more challenging grading tasks for salmon in the future, as well as future novel applications in fish processing as a whole.

7.1 Summary of Contributions

In this section, a short summary is presented of the contributions that is perceived to be the most important.

- An implementation that efficiently extracts and transforms the raw images obtained from the cameras to world coordinates by utilizing a graphical processing unit. While implementing code for a GPU is not difficult in itself, implementing *efficient* code on the GPU requires knowledge of how it operates. While the GPU code is not extraordinary on a global scale, it lowers the threshold for third party partners to take 3D-imaging into use.
- Algorithms for making the point cloud more suitable for analysis was implemented. This was done through statistical filtering and removal of unwanted features, such as fins. The medial axis was computed, which has clear advantages in shape recognition.
- During the development of the features used in the classification, many alternatives were attempted. The features found gives further projects a starting point, or benchmark, in the search for alternative or better features.
- Alternatives to many of the chosen strategies were presented. Some of the alternatives were not followed either because of time constraints or the fear that the algorithm would prove to be too computationally intensive for an real-time application. This provides a path of improvement for further work.

- The thesis proves that 3D scanning in 360° has a place in automatic quality control. With improvements in the performance of computers, faster and cheaper camera hardware, the concept can only improve in its applicability.
- The results obtained in deformity and wound detection proves that the features found are able to separate the classes in the dataset with an acceptable accuracy.

The software implementation responsible for acquiring images at high speed, which has been developed and refined during this thesis, will be partly or fully used by an external company for industrial use in the beginning of 2016. This would not be possible if the implementation was not optimized with real-time use in mind. The findings in this thesis will also be published in a scientific journal in collaboration with SINTEF.

7.2 Recommendations and Further Work

As this thesis spans a wide range of techniques, many of the approaches has more complex alternatives that needs to be explored. Implementation-time was weighted against the obtainable results, due to the time constraints imposed.

- The largest concern in this thesis is the size of the dataset. As this thesis deals exclusively with objects that are biological - the variations in the dataset are large compared to the number of samples. The results at this stage can therefore only serve as an indication of obtainable prediction rate, and not something conclusive. Having a larger dataset would enable further experimentation with both features and classifiers without worrying that the result is skewed due to being tailored to the samples present in the dataset.
- Experiments with more robust methods of extracting the center-point of the laser-line is recommended, at least if the polarization filter ends up not being used. The current method uses a weighted first moment to determine the center. If the laser hits a surface at an angle, this would skew the center-point towards that side. This could also be improved on by experimenting more with optics, i.e. stronger laser in conjunction with a polarization filter.
- To obtain a color image in 2D, a simple perspective projection was performed onto a plane from the bottom and top of the salmon. This works, but does not preserve the surface areas completely. Wounds at an angle will therefore appear to have a smaller surface area than in reality. An area-preserving projection can for example be done by calculating the normal vector on the surface before projecting and stretching the image to compensate.
- In this thesis, a spline interpolation was performed for each cross section of the salmon. This decouples the spline interpolation from utilizing spatial information along the length of the salmon. Moving least squares (MLS) was attempted as an alternative, but was found to perform poorly to obtain the degree of smoothness desired. An implementation of moving least squares on the GPU would be a possible course of action.

- The wound classification in this thesis proceeded using only color information. Originally, the idea was to use how the laser reflects off the surface as another means to detect wounds. Removing the polarization filter induced too much noise for this to be an option. The polarization filter should therefore be reintroduced with a stronger laser. The features used in wound classification can then be re-evaluated.
- Develop a new calibration object, that perhaps is moved along the conveyor belt as opposed to placed statically. This enables all cameras to be calibrated at once, negating any misalignment. This removes the need to perform a separate routine after the main calibration to account for misalignment.

Bibliography

- Asada, H., Brady, M., 1986. The curvature primal sketch. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (1), 2–14.
- Aurenhammer, F., 1991. Voronoi diagrams a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)* 23 (3), 345–405.
- Barber, C. B., Dobkin, D. P., Huhdanpaa, H., 1996. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)* 22 (4), 469–483.
- Blum, H., 1967. A transformation for extracting descriptors of shape.
- Bouckaert, R. R., Frank, E., 2004. Evaluating the replicability of significance tests for comparing learning algorithms. In: *Advances in knowledge discovery and data mining*. Springer, pp. 3–12.
- Bradski, G., 2000. *Dr. Dobb's Journal of Software Tools*.
- Buljo, J., Gjerstad, T., Caldwell, D., et al., 2013. Robotics and automation in seafood processing. In: *Robotics and Automation in the Food Industry. Current and Future Technologies*. Woodhead Publishing.
- Chang, C.-C., Lin, C.-J., 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2 (3), 27.
- Digre, H., 2014. Profitable refinement of seafood in norway (loennsom foredling av sjøemat i norge). Tech. Rep. A26355, SINTEF.
- Duda, R. O., Hart, P. E., Stork, D. G., 2012. *Pattern classification*. John Wiley & Sons.
- Egeness, F.-A., 2013. Chinese production of frozen filetprodukter from cod (kinesisk produksjon av fryste filetprodukter av torsk). Tech. rep., Nofirma.no.
URL <http://www.nofima.no/filearchive/Rapport%2026-2013.pdf>
- Fjelldal, P., Hansen, T., Breck, O., Ørnstrud, R., Lock, E.-J., Waagbø, R., Wargelius, A., Eckhard Witten, P., 2012. Vertebral deformities in farmed atlantic salmon (*salmo salar* l.)—etiology and pathology. *Journal of Applied Ichthyology* 28 (3), 433–440.

-
- Fjelldal, P. G., Hansen, T. J., Berg, A. E., 2007. A radiological study on the development of vertebral deformities in cultured atlantic salmon (*salmo salar* L.). *Aquaculture* 273 (4), 721 – 728.
URL <http://www.sciencedirect.com/science/article/pii/S0044848607005662>
- Hartley, R., Zisserman, A., 2003. *Multiple view geometry in computer vision*. Cambridge university press.
- Hsu, C.-W., Chang, C.-C., Lin, C.-J., et al., 2003. A practical guide to support vector classification.
- Katz, R. A., Pizer, S. M., 2003. Untangling the blum medial axis transform. *International Journal of Computer Vision* 55 (2-3), 139–153.
- Kristiansen, J. E., 2014. *This is norway 2014*. Statistics Norway, SSB, ISBN 978-82-537-8978-1.
URL <https://www.ssb.no/en/befolkning/artikler-og-publikasjoner/this-is-norway-2014>
- Litzenberger, G., 2009. *World robotics, industrial robots*. International Federation of Robotics (IFR) Statistical Department.
- Liu, S., Ye, Y., 2011. Point cloud segmentation using gradient vector flow snake. In: *Information Science and Technology (ICIST), 2011 International Conference on*. IEEE, pp. 1114–1118.
- Macrini, D., Siddiqi, K., Dickinson, S., 2008. From skeletons to bone graphs: Medial abstraction for object recognition. In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, pp. 1–8.
- Mathiassen, J., Jansson, S., Veliyulin, E., Njaa, T., Lønseth, M., Bondø, M., Østvik, S., Risdal, J., Skavhaug, A., 2006. Automatic weight and quality grading of whole pelagic fish. In: *In Proceedings NFTC 2006, the 1st Nor-Fishing Technology Conference, Trondheim, Norway*.
- Miller, A., Gregory, K., 2012. *C++ AMP*. Pearson Education.
- Misimi, E., Erikson, U., Skavhaug, A., 2008. Quality grading of atlantic salmon (*salmo salar*) by computer vision. *Journal of food science* 73 (5), E211–E217.
- Misimi, E., Mathiassen, J. R., Erikson, U., Skavhaug, A., 2006. Computer vision based sorting of atlantic aslmon (*salmo salar*) according to shape and size. In: *Proceedings of International Conference on Computer Vision Theory and Applications, VISAPP 2006. Vol. 1*. pp. 265–270.
- Rusu, R. B., Cousins, S., May 9-13 2011. 3D is here: Point Cloud Library (PCL). In: *IEEE International Conference on Robotics and Automation (ICRA)*. Shanghai, China.

-
- Sandberg, M. G., 2014. Profitability and employment in the norwegian sea-food industry (verdiskapning og sysselsetting i norsk sjoematnaering). Tech. Rep. A26088, SINTEF.
URL <http://www.fhf.no/prosjektdetaljer/?projectNumber=900899>
- Schmitt, M., 1989. Some examples of algorithms analysis in computational geometry by means of mathematical morphological techniques. In: Geometry and robotics. Springer, pp. 225–246.
- Shawe-Taylor, J., Cristianini, N., 2004. Kernel methods for pattern analysis. Cambridge university press.
- Siddiqi, K., Pizer, S., 2008. Medial representations: mathematics, algorithms and applications. Vol. 37. Springer Science & Business Media.
- Smola, A. J., Schölkopf, B., 1998. Learning with kernels. Citeseer.
- Studholme, C., Hill, D. L., Hawkes, D. J., 1999. An overlap invariant entropy measure of 3d medical image alignment. Pattern recognition 32 (1), 71–86.
- Subasinghe, R., 2005. Aquaculture topics and activities. state of world aquaculture.
URL <http://www.fao.org/fishery/topic/13540/en>
- Vågsholm, I., Djupvik, H., 1998. Risk factors for spinal deformities in atlantic salmon, salmo salar l. Journal of fish diseases 21 (1), 47–54.
- Vapnik, V., 1995. The nature of statistical learning theory. Springer Science & Business Media.
- Wikipedia, 2006. Bayer pattern on sensor profile.
URL http://commons.wikimedia.org/wiki/File:Bayer_pattern_on_sensor_profile.svg#/media/File:Bayer_pattern_on_sensor_profile.svg
- Witten, P. E., Gil-Martens, L., Hall, B. K., Huysseune, A., Obach, A., et al., 2005. Compressed vertebrae in atlantic salmon salmo salar: evidence for metaplastic chondrogenesis as a skeletogenic response late in ontogeny. Dis. Aquat. Org 64, 237–246.
- Witten, P. E., Gil-Martens, L., Huysseune, A., Takle, H., Hjelde, K., 2009. Towards a classification and an understanding of developmental relationships of vertebral body malformations in atlantic salmon (salmo salar l.). Aquaculture 295 (1), 6–14.
- Yushkevich, P. A., 2009. Continuous medial representation of brain structures using the biharmonic pde. NeuroImage 45 (1), S99–S110.
- Zhao, W., Chellappa, R., Phillips, P. J., Rosenfeld, A., Dec. 2003. Face recognition: A literature survey. ACM Comput. Surv. 35 (4), 399–458.
URL <http://doi.acm.org/10.1145/954339.954342>

Appendix

A Dataset

Red sample numbers indicates that the salmon has been completely unused. I.e a defect not considered in this thesis.

Table A1: Superior dataset 1/2

Sample number	Comments
Salmon 1	-
Salmon 2	-
Salmon 3	-
Salmon 4	Has a tendency to be wrongly classified due to an abnormal shape at the middle (high middle back). Kept in final dataset.
Salmon 5	-
Salmon 6	-
Salmon 7	-
Salmon 8	-
Salmon 9	-
Salmon 10	Deep cut during gutting at head, detected as humpback due to increased width of the abdominal flap. Removed from final dataset.
Salmon 11	Signs of humpback. Removed from the final dataset.
Salmon 12	-
Salmon 13	-
Salmon 14	-
Salmon 15	-
Salmon 16	-
Salmon 17	-
Salmon 18	-
Salmon 19	Signs of humpback. Removed from the final dataset.
Salmon 20	-
Salmon 21	-
Salmon 22	-
Salmon 23	-
Salmon 24	-

Table A2: Superior dataset 2/2

Sample number	Comments
Salmon 41	Innards hanging out at the hind part of the abdominal cut.
Salmon 42	Signs of humpback, often misclassified. Kept in final dataset.
Salmon 43	-
Salmon 44	-
Salmon 45	-
Salmon 46	-
Salmon 53	Signs of humpback.
Salmon 54	-
Salmon 55	-
Salmon 56	-
Salmon 57	Weak humpback
Salmon 58	-
Salmon 59	-
Salmon 60	-
Salmon 61	-
Salmon 62	Appears to have humpback
Salmon 63	-
Salmon 64	-
Salmon 65	-
Salmon 66	Appears to have humpback
Salmon 67	-

Table A3: Ordinary dataset

Sample number	Defects
Salmon 25	Bad gutting behind anal fin, scraped shells left side (dark regions)
Salmon 26	Slight humpback
Salmon 27	No specific deformities, generally deformed
Salmon 28	Slight humpback
Salmon 29	Humpback
Salmon 30	Possibly downgraded due to bad gutting or abnormal size
Salmon 31	Humpback
Salmon 32	Humpback
Salmon 33	Possible scaling left side
Salmon 34	Humpback
Salmon 68	Small deformation (humpback)
Salmon 69	Small deformation (wide back half)
Salmon 70	Small deformation (abnormal shape)
Salmon 71	Small deformation (humpback)
Salmon 72	Small deformation (thin, abnormal shape)
Salmon 77	Melanin spots (inside)
Salmon 78	Melanin spots (inside)
Salmon 79	Melanin spots (inside)
Salmon 80	Melanin spots (inside)
Salmon 81	Melanin spots (inside)
Salmon 92	Small deformation (thin)
Salmon 93	Small deformation (humpback, thick)
Salmon 94	Small deformation (uncertain, slight humpback and thick)
Salmon 95	Small deformation (wide back half, short back half)
Salmon 96	Small deformation (humpback, wide front)
Salmon 97	Melanin spots (inside)
Salmon 98	Melanin spots (inside)
Salmon 99	Melanin spots (inside)
Salmon 100	Melanin spots (inside)

Table A4: Production dataset

Sample number	Defects
Salmon 35	Humpback
Salmon 36	Bad gutting
Salmon 37	Humpback, scraped shells
Salmon 38	Small wound
Salmon 39	Wound, humpback
Salmon 40	Small size
Salmon 47	Wound, humpback, thin backside
Salmon 48	Humpback, small wound
Salmon 49	Wound
Salmon 50	Scraped shells
Salmon 51	Wound, humpback
Salmon 52	Scraped shells
Salmon 73	Bad gutting
Salmon 74	Bad gutting
Salmon 75	Bad gutting
Salmon 76	Bad gutting, small size
Salmon 82	Large deformations
Salmon 83	Large deformations
Salmon 84	Large deformations
Salmon 85	Large deformations
Salmon 86	Large deformations
Salmon 87	Wounds
Salmon 88	Wounds
Salmon 89	Wounds
Salmon 90	Wounds
Salmon 91	Wounds
Salmon 101	Wounds
Salmon 102	Wounds
Salmon 103	Wounds
Salmon 104	Wounds
Salmon 105	Wounds

B Camera Triggering (Arduino)

```
1 // Contains the code to trigger the laser and cameras individually
2 // Individual triggering is done to avoid laser-line overlap
3
4 // Note1: transfer overlaps with exposure in camera mode 14 for pt.↵
grey
5 // Note2: LED Transfer requires more time than the laser frames, as ↵
they are captured at the same time – while laser images are taken ↵
sequentially
6
7 typedef const unsigned int c_uint;
8
9 #define FPS 400 // Target fps
10 #define EXPOSURE_TIME 300 // Camera exposure time (microseconds)
11
12 // Estimate of the time the camera spends on data transfer after ↵
capture
13 #define DATA_TRANSFER_LASER 400
14 #define DATA_TRANSFER_LED 1000
15
16 // States
17 #define LASERREADY 0
18 #define TAKELASERIMAGE_1 1
19 #define TAKELASERIMAGE_2 2
20 #define TAKELASERIMAGE_3 3
21 #define LEDREADY 4
22 #define TAKELEDIMAGE 5
23 #define DELAY_STATE 6
24
25 // Output PIN IDs
26 #define CAM1 10 // RIGHT CAMERA PIN
27 #define CAM2 11 // TOP CAMERA
28 #define CAM3 12 // LEFT CAMERA
29 #define LASER1 3 // RIGHT LASER
30 #define LASER2 4 // TOP LASER
31 #define LASER3 2 // LEFT LASER
32 #define LED 6
33
34 #define NCAM 3 // Number of cameras
35
36 // Timing (in microseconds)
37
38 // Total requested period
39 PERIOD = (1000000UL / FPS) * 2;
40 // Small delay applied between turning laser on and capture
41 c_uint WAIT_LASER_ONOFF = 50;
42 // Allow for exposure time between laser images
43 c_uint WAIT_AFTER_LASERIMAGE = EXPOSURE_TIME + WAIT_LASER_ONOFF;
44 // Allow the laser images to be transferred
45 c_uint WAIT_AFTER_LEDREADY = DATA_TRANSFER_LASER + WAIT_LASER_ONOFF;
46 // Allow time for both transfer and exposure
47 c_uint WAIT_AFTER_LEDIMAGE = EXPOSURE_TIME + DATA_TRANSFER_LED;
48 // Delay before laser triggering begins
49 c_uint WAIT_AFTER_LASERREADY = 50;
```

```

50
51 //    Add up the delays so far, and use it to define the wait before ↔
    starting a new grab sequence
52 c_uint TIMESPENT = N_CAM * ( WAIT_AFTER_LASERIMAGE )
53                   + WAIT_AFTER_LEDREADY
54                   + WAIT_AFTER_LEDIMAGE
55                   + WAIT_AFTER_LASERREADY;
56 c_uint DELAYTIME = PERIOD - TIMESPENT;
57
58 // Wait time between states
59 int waitTimes[]= {
60     WAIT_AFTER_LASERREADY,
61     WAIT_AFTER_LASERIMAGE,
62     WAIT_AFTER_LASERIMAGE,
63     WAIT_AFTER_LASERIMAGE,
64     WAIT_AFTER_LEDREADY,
65     WAIT_AFTER_LEDIMAGE,
66     DELAYTIME
67 };
68
69 // The setup function runs once when you press reset or power the board
70 void setup() {
71     pinMode(CAM1, OUTPUT);
72     pinMode(CAM2, OUTPUT);
73     pinMode(CAM3, OUTPUT);
74     pinMode(LASER1, OUTPUT);
75     pinMode(LASER2, OUTPUT);
76     pinMode(LASER3, OUTPUT);
77     pinMode(LED, OUTPUT);
78 }
79
80 //    Initial state
81 int state = LASERREADY;
82
83 //    Initial timer value
84 unsigned long lastFrameTime = micros();
85
86 //    Loop through the state machine
87 void loop()
88 {
89     // Absolute value due to counter resetting after ~70minutes
90     if(abs(micros() - lastFrameTime) > waitTimes[state])
91     {
92         lastFrameTime += waitTimes[state];
93         StateMachine();
94     }
95 }
96
97 void StateMachine()
98 {
99     switch (state) {
100     case LASERREADY:
101         readyCamera(CAM1);
102         readyCamera(CAM2);
103         readyCamera(CAM3);
104         setLED(LOW);
105         state = TAKELASERIMAGE_1;

```

```

106     break;
107
108     case TAKELASERIMAGE_1:
109         setLaser(LASER1, HIGH);
110         delayMicroseconds(WAIT_LASER_ONOFF);
111         takePicture(CAM1);
112         state = TAKELASERIMAGE_2;
113         break;
114
115     case TAKELASERIMAGE_2:
116         setLaser(LASER2, HIGH);
117         delayMicroseconds(WAIT_LASER_ONOFF);
118         setLaser(LASER1, LOW);
119         takePicture(CAM2);
120         state = TAKELASERIMAGE_3;
121         break;
122
123     case TAKELASERIMAGE_3:
124         setLaser(LASER3, HIGH);
125         delayMicroseconds(WAIT_LASER_ONOFF);
126         setLaser(LASER2, LOW);
127         takePicture(CAM3);
128         state = LEDREADY;
129         break;
130
131     case LEDREADY:
132         readyCamera(CAM1);           // Cam1 ready for triggering
133         readyCamera(CAM2);           // Cam2 ready for triggering
134         readyCamera(CAM3);           // Cam3 ready for triggering
135         delayMicroseconds(WAIT_LASER_ONOFF); // Added delay to ensure correct←
            image embedding
136         setLaser(LASER1,LOW);        // Turn off laser1
137         setLaser(LASER2,LOW);        // Turn off laser1
138         setLaser(LASER3,LOW);        // Turn off laser1
139         setLED(HIGH);                // Turn on led
140         state = TAKELEDIMAGE;
141         break;
142
143     case TAKELEDIMAGE:
144         takePicture(CAM1);           // Trigger cam1
145         takePicture(CAM2);           // Trigger cam2
146         takePicture(CAM3);           // Trigger cam3
147         state = DELAY_STATE;
148         break;
149
150     case DELAY_STATE:
151         state = LASERREADY;
152         break;
153 }
154 }
155
156 void takePicture(int cameraNumber){
157     digitalWrite(cameraNumber, LOW);
158 }
159
160 void readyCamera(int cameraNumber){
161     digitalWrite(cameraNumber, HIGH);

```

```
162 }
163
164 void setLaser(int laserNumber, int set){
165     digitalWrite(laserNumber, set);
166 }
167
168 void setLED(int set){
169     digitalWrite(LED, set);
170 }
```

C Pruning and Spline Re-sampling (Matlab)

```
1 function [ X_spl, Y_spl] = SliceSpline( X, Y, n_eval, p, d_rho_threshold←→
   , rho_residual_threshold)
2 % Dimensions: [Cam][X], [Cam][Y]
3
4 % Flatten to 1D
5 X = reshape(X, numel(X), 1);
6 Y = reshape(Y, numel(Y), 1);
7
8 % Remove points where Y is NaN
9 X = X(~isnan(Y));
10 Y = Y(~isnan(Y));
11
12 %% Curve parameterization
13 % Translate points around centroid
14 c = [mean(Y), mean(X)];
15 cx = X - c(2);
16 cy = Y - c(1);
17
18 % Polar coordinates
19 [th,rho] = cart2pol(cx,cy);
20 th(th < 0) = th(th < 0) + 2*pi;
21 rho_mean = mean(rho);
22 n_samples = size(X,1);
23
24 % Sort by ascending theta, apply same indexing to rho, X, Y
25 [th, th_I] = sort(th);
26 rho = rho(th_I);
27 X = X(th_I);
28 Y = Y(th_I);
29
30 % Find regions of discontinuity (in rho), smooth diff result (padded to
31 % initialize filter). Extract the original sequence.
32 d_rho = diff([rho; rho(1)]);
33 windowSize = 20;
34 b = (1/windowSize)*ones(1,windowSize);
35 d_rho_smooth = filter(b, 1, abs([d_rho(end-windowSize:end); d_rho; d_rho←→
   (1:windowSize)]));
36 d_rho_smooth = d_rho_smooth(windowSize+1:end-windowSize-1);
37
38 % Find smoothed indices that exceeds the threshold
39 d_rho_threshold = d_rho_threshold * max(rho_mean, 3);
40 d_rho_above_I = find(d_rho_smooth > d_rho_threshold);
41 th_above = th(d_rho_above_I);
42
43 % Discontinuity detected?
44 if (~isempty(th_above))
45     % Find start and end of continous regions
46     d_theta_above = diff(th_above);
47
48     % Merge regions if they are separated by less than a certain angle
49     threshold_sep_angle = 5*pi/180;
50     d_theta_sameregion = d_theta_above < threshold_sep_angle;
51
```

```

52 % Check if the region bounds around
53 region_islooped = abs((th_above(end) - th_above(1) - 2*pi)) < ↔
    threshold_sep_angle;
54
55 % Indices of separation (in the original array)
56 region_indices = find(~d_theta_sameregion);
57 region_temp = zeros(size(region_indices,1)*2, 1);
58 for i = 1:2:size(region_temp,1)
59     region_temp(i) = d_rho_above_I(region_indices(uint16(i/2)));
60     region_temp(i+1) = d_rho_above_I(region_indices(uint16(i/2)) + ↔
        1);
61 end
62
63 region_separator = [d_rho_above_I(1)-1; region_temp; d_rho_above_I(↔
    end)+1];
64
65 clear region_indices region_temp
66
67 % The number of regions is equal to the number of boundaries (-1)
68 % If it loops around, the two closest areas will be merged (-1 if ↔
    true)
69 % Last index is allocated for looped_area
70 n_disc_regions = floor(size(region_separator,1) / 2);
71 if(region_islooped)
72     n_disc_regions = n_disc_regions - 1;
73 end
74
75 disc_area = cell(n_disc_regions, 1);
76
77 % Get the indices associated with each discontinuous area
78 if(region_islooped)
79     % Extract the beginning and ending indices
80     disc_area{end} = [region_separator(end):n_samples , 1:↔
        region_separator(1)];
81     region_separator(end) = [];
82     region_separator(1) = [];
83 end
84
85 % Set indices for non-looped regions
86 for i = 1:(n_disc_regions - region_islooped)
87     % Lower bound must be shifted one position due to diff
88     lower_bound = region_separator(i*2 - 1);
89     upper_bound = region_separator(i*2);
90     disc_area{i} = lower_bound:upper_bound;
91 end
92
93 % Ignore small areas (less than a certain amount)
94 threshold_min_area = 2*pi/180;
95 disc_area(all(cellfun(@isempty, disc_area),2), : ) = []
96 area_th_sz = cellfun(@(x) abs(th(x(end)) - th(x(1))), disc_area);
97 disc_area = disc_area( area_th_sz > threshold_min_area );
98
99 % Pad remaining regions a bit, to improve the regularity of the mean↔
    value
100 th_padding = 5*pi/180;
101 n_disc_padding = round(th_padding * n_samples / (2*pi));
102

```

```

103 % For each region, save indices to remove (above mean)
104 padded_area_remove = [];
105 for i = 1:size(disc_area,1)
106     disc_start = disc_area{i}(1);
107     disc_end = disc_area{i}(end);
108     padded_area = [ disc_start - n_disc_padding : disc_start-1, ...
109                   disc_area{i}, ...
110                   disc_end+1 : disc_end + n_disc_padding ];
111
112     % If padded index exceeds n_sample, roll around
113     idx_above = padded_area > n_samples;
114     padded_area(idx_above) = padded_area(idx_above) - n_samples + 1;
115     idx_below = padded_area < 1;
116     padded_area(idx_below) = padded_area(idx_below) + n_samples - 1;
117
118     % Remove points above mean in region (one-sided, outwards)
119     rho_mean_area = mean(rho(padded_area));
120     padded_area_remove = [padded_area_remove, padded_area(rho(←
121         padded_area) > rho_mean_area)];
122 end
123 % Remove the points
124 X(padded_area_remove) = [];
125 Y(padded_area_remove) = [];
126 end
127 n_samples = size(X, 1);
128
129 %% Spline fit
130
131 % Add padding to ensure smooth spline
132 n_padding = round(0.2 * n_samples);
133 X_padded = [X(end-n_padding+1:end); X; X(1:n_padding)];
134 Y_padded = [Y(end-n_padding+1:end); Y; Y(1:n_padding)];
135
136 % Set up parameterization from 0 to 1, without the padding (extend range←
137 )
138 % beyond 0 and 1 for the padded samples.
139 t_padding_down = linspace(-n_padding/(n_samples-1), -1/(n_samples-1), ←
140     n_padding);
141 t_padding_up = linspace(1 + 1/(n_samples-1), 1 + n_padding/(n_samples-1)←
142     , n_padding);
143 t = [t_padding_down, linspace(0, 1, n_samples), t_padding_up];
144
145 %% Spline fit using robustness measure (tolerance)
146 % Define tolerance in terms of rho
147 [~, avg_rho] = cart2pol(X, Y);
148 avg_rho = mean(avg_rho);$
149 tol = p * max(avg_rho, 4);
150
151 [spl1, values] = spaps(t, [X_padded'; Y_padded'], tol);
152
153 X_values = values(1,:)';
154 Y_values = values(2,:)';
155 [~, rho_values] = cart2pol(X_values, Y_values);
156 [~, rho_padded] = cart2pol(X_padded, Y_padded);
157 rho_residuals = rho_padded - rho_values;

```

```

156 X_padded_removed = X_padded;
157 Y_padded_removed = Y_padded;
158
159 % Iteratively remove samples that exceeds the residual threshold (Fast ↔
    selective RANSAC)
160 rho_residual_threshold = rho_residual_threshold * max(avg_rho, 5);
161
162 while(any(abs(rho_residuals) > rho_residual_threshold))
163     % Remove residuals further out than threshold
164     rho_residual_below_I = abs(rho_residuals) < rho_residual_threshold;
165     X_padded_removed = X_padded_removed(rho_residual_below_I);
166     Y_padded_removed = Y_padded_removed(rho_residual_below_I);
167     t = t(rho_residual_below_I);
168
169     % Re-train
170     [spl1, values_new] = spaps(t, [X_padded_removed'; Y_padded_removed←
        '], tol);
171
172     % Calculate new residuals
173     X_values = values_new(1,:)';
174     Y_values = values_new(2,:)';
175     [~, rho_values] = cart2pol(X_values, Y_values);
176     [~, rho_padded] = cart2pol(X_padded_removed, Y_padded_removed);
177     rho_residuals = rho_padded - rho_values;
178 end
179
180 % Evaluate spline within spline interval (0-1)
181 t_eval = linspace(max(1/n_eval, min(t)), min(1-(1/n_eval), max(t)), ←
    n_eval);
182 XY_spl = fnval(spl1, t_eval);
183 X_spl = XY_spl(1,:)';
184 Y_spl = XY_spl(2,:)';
185
186 end

```