NTNU
Norwegian University of
Science and Technology

Classification of fish body parts in an underwater environment

Thorbjørn Sømod

Master of Science in Cybernetics and Robotics
Submission date: July 2017
Supervisor: Annette Stahl, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Acknowledgements

I would first like to thank my supervisor Assistant Professor Annette Stahl of the Department of Engineering Cybernetics at the Faculty of Information Technology and Electrical Engineering at the Norwegian University of Science and Technology (NTNU). Her knowledge within the fields of machine learning, mathematics and computer vision, along with her guidance and support throughout the semester is what has made this thesis possible. Further thanks go to my co-supervisors Per Rundtop and Christian Schellewald at SINTEF Ocean. I would also like to thank Milan Markovic, Oscar Markovic and my other colleagues at Sealab AS for their knowledge on underwater imaging, supply of technical equipment and dry humor. A special thank you is also given to Navid Nourani Vatani, who over an email correspondence on the topic of his software library on Local Binary Pattern (LBP) has helped me immensely in my work. Finally, I would like to express gratitude to my parents and my girlfriend for providing me with support and encouragement throughout my years of study, and also over the course of writing this thesis. Thank you.

Author

Thorbjørn Sømod

Summary

This master thesis is the result of work conducted over the course of a semester with the goal of invstigating a possible approach to recognizing fish parts in a video stream from a camera system situated in an underwater environment. This task is seen as the first part of a three-step scheme for implementing an automatic system for fish health assessment in the fish farming industry. This thesis describes work done in setting up an interface to an IP camera that is situated in an underwater environment, collecting and labelling image material from the camera system for training and testing object classifiers, and training the object classifiers for multi-class object recognition based on image descriptors suitable for an underwater environment. Finally, a complete object recognition framework is implemented and performance tests are performed based on the pre-trained classifiers, and the results are analyzed. The results of this thesis show that it is possible to create a system that is able to perform this classification by relying on Support Vector Machine (SVM) classifiers based on adaptations of the Local Binary Pattern (LBP) image patch descriptor. By using a linear SVM classifier good results are achieved.

Sammendrag

Denne masteroppgaven er resultatet av arbeid utført i løpet av et semester med mål om å utvikle en mulig tilnærming til å gjenkjenne kroppsdeler av fisk i en bildestrøm fra et kamerasystem som opererer i et undervannsmiljø. Denne oppgaven er sett på som den første delen av en tre-trinns løsning for implementering av et automatisk system for vurdering av fiskens helsetilstand i oppdrettsindustrien. Denne oppgaven beskriver arbeidet med å sette opp et grensesnitt til et IP-kamera som opererer i et undervannsmiljø, samling og merking av bildemateriale fra kamerasystemet for opplæring og testing av objektklassifikatorer, og trening av objektklassifikatorer for multiklasse objektgjenkjenning basert på bildebeskrivelser egnet for et undervannsmiljø. Et rammeverk er konstruert for gjenkjenning av objekter og prestasjonstester utført basert på de tidligere trente klassifikatorene. Resultatene blir deretter analysert. Resultatene av denne avhandlingen viser at det er mulig å opprette et system som kan utføre denne klassifiseringen ved å bruke SVM klassifikatorer basert på tilpassninger av LBP koder. Ved å bruke en lineær SVM klassifikator oppnås gode resultater.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This chapter serves as an introduction to the main body of the thesis. Section 1.1 describes the motivation behind the work conducted. Section 1.2 defines the objectives and scope of the project, while Section 1.3 comments on related work. Because there exists an excessive amount of research articles on the topic of object recognition this section mentions important object recognition milestones, but focuses on work in the sub-field of object recognition in an underwater environment and work that is related to the approaches proposed in this thesis. Finally, Section 1.4 provides an outline of the thesis.

1.1 Motivation

Over the last decades global population growth has led to an increasing demand for food, and a focus on sustainable food sources. One group of products that are often associated with sustainability are those provided by the aquaculture industry, such as farmed fish, crustaceans, molluscs, aquatic plants and algae. Because of this companies in the aquaculture industry are now under great pressure to deliver the quantities of product requested by the consumers at the desired quality. This increasing demand for aquatic products has, however, revealed shortcomings at many stages in current industry operations. Outdated technology and/or manual labor is often employed at many stages of production, placing a limit on processing efficientcy and product quality. Manual labor, by virtue of requiring wages for employees, is also less cost-effective than utilizing automation. Because of this many companies in the aquaculture industry are now exploring ways of integrating new advanced technology into the production pipelines at their facilities.

Considering that this work is still in development there are many challenges that have yet to be met. One such challenge is ensuring fish welfare in the fish farming industry. Within this sub-field of aquaculture the fish stock is often exposed to a number of hazards at the different stages of production. This includes physical hazards and stress during transport between facilities and diseases that thrive in the large quantities the fish are kept in at these facilities. Salmon during two different stages of production in fish farming can be seen in Figure 1.1. Ensuring fish welfare is important for companies in the aquaculture industry, both economically and ethically. Economically, because flawed stock will yield less revenue when it is sold, and ethically, because keeping sick or damaged fish alive is immoral if it can not be treated. According to a study conducted by Veterinærinstituttet (2017) financial losses as a result of sickness or damages in the fish stock, in addition to lost fish, can be as much as 20% of the approximated value of the stock.

(a) Relocation of adolescent salmon. **(b)** Salmon in an enclosure at a slaughterhouse.

Figure 1.1: Salmon at different stages of the production.

An important part of ensuring fish welfare is performing health assessments by mapping occurances of sickness and damages in the fish stock. Today, this task is often realized by manual sampling of the fish at the various stages of production. This manual approach is inefficient and is also considered invasive as the fish is often physically removed from the water in order to perform measurements and testing. By using new technology to make this task automated health assessments could become notably more accurate, whilst also making the task non-invasive and reducing the costs of the current manual labor.

To automate this task a scheme based on using a high definition underwater camera system with computer vision and machine learning algorithms is proposed. The scheme comprises of three steps. First, parts of the fish body that are susceptible to sickness or damages are recognized in the frames of a video stream. Secondly, the recognized fish parts are assessed based on features such as contours and/or color spectrums to determine their condition. Finally, the condition assessments are used to calculate statistics on the fish stock to the benefit of the aquaculture companies. In addition to making this task automated and non-invasive the proposed scheme would also yield real-time statistical data on the fish stock, in comparison to ordinary scheduled health assessments.

This thesis investigates a possible approach to the first part of the aforementioned three-step scheme; recognizing fish body parts in frames acquired from a camera system situated in an underwater environment. The proposed approach utilizes SVM classifiers included in LIBSVM by Chang and Lin (2011) and LIBLINEAR by Chang and Lin (2011). To overcome recognition restrictions present in an underwater environment, such as varying levels of illumination, while preserving location, scale and rotation invariance both approaches utilize various forms of the LBP operator for image patch description. This thesis describes the mathematical theory behind this approach and how the approach is implemented programatically. The thesis also describes how image material for training and testing of the classifiers was acquired from the camera system, how the image material was pre-processed, and how the respective classifiers were trained. A performance evaluation of the proposed implementations was also conducted, and the test results are presented and discussed.

1.2 Objectives and scope

The aim of the work conducted for this thesis has been to investigate possible approaches to recognizing fish parts in a video stream from a camera system situated in an underwater environment using computer vision and machine learning algorithms. This task can be seen as the first part of a three-step scheme for implementing an automatic system for fish health assessment in the fish farming industry, as described in Section 1.1. The part of the scheme described in this thesis could be seen as comprising of five main objectives.

- Construct an interface to the camera system situation in an underwater camera system. The camera system used for the experiments conducted for this thesis will be the Sony FCB-EV7520 IP camera enclosed in housing suited for an underwater environment provided by Sealab AS.
- Collect image material for training and testing of object classifiers from the camera system operating in an underwater environment at a field location and pre-process this material.
- Train classifiers for a multi-class object recognition system based on image descriptors suitable for an underwater environment.
- Implement a functioning object recognition system based on pre-trained classifiers.
- Perform performance testing based on pre-trained classifiers and analyze the results.

So that the reader can properly appreciate the work that has resulted in this thesis, some additional information on the scope of the work should be mentioned.

- Annotated image material for this project was nearly non-existent. Consequently some object recognition schemes that rely on large amounts of image material, i.e neural nets, were not considered when deciding the type of classifiers that the approaches would implement.
- Because the majority of the fish farming industry in the native country of the author is focused on the breeding of salmon the work conducted for this thesis has focused on the recognition of salmon body parts. The proposed systems could easily be extended to handle other species of fish, but this will not be explored further.
- The focus of the object recognition approaches implemented for this thesis has been achieving high object recognition accuracy over increased recognition speed. Because the completed system will be left operating at fish farming locations for long periods at a time it is not important to achieve real-time recognition.

1.3 Related work

Object recognition in digital imagery is a field within computer vision that has received considerable attention in the recent decades. This has primarily been the result of technological advances in computing which have allowed efficient object recognition to be achievable. This development has resulted in an extensive amount of research being conducted on the topic, with many different approaches being investigated. The different approaches can roughly be grouped into the classes of appearance-based methods, feature-based methods, template-based methods and methods based on artificial neural nets, as noted by Achatz (2016). This section presents a selection of important contributions to the field of object recognition for three of these classes, along with contributions to the more specific field of underwater object recognition. The class of template-based methods is not included in this review as most of these methods are deemed outdated.

An important milestone in the field of object recognition was the introduction of the Scale-Invariant Feature Transform (SIFT) descriptor by Lowe (1999). This feature-based method allows for the extraction of local features that are invariant to image scale and rotation, and that exhibits a high tolerance for noice and change in illumination. Additionally, the features based on SIFT are also highly distinctive, making them well suited for object recognition methods that rely on feature matching. However, the cost of extracting features based on SIFT is high, and the features are thus ordinarily not suited for real-time applications. The high extraction cost can be minimized by using a classification approach based on cascade filtering in which the computationally heavy operations are only applied at filtered locations that have passed the earlier blocking stages of the cascade. The original SIFT has been revisited a number of times, such as by Lowe (2003). It has also been combined with other feature descriptors to produce even greater robustness to image deformations and faster feature matching, as attempted by Ke and Sukthankar (2004).

The work presented by Viola and Jones (2001) was another turning point in the field of object recognition. In their paper they show that a cascade of weak classifiers based on the extraction of Haar-like features can be used to obtain a capable real-time object detector for face detection. This feature-based method, which has later come to be known as the Viola & Jones algorithm, offers efficient feature selection and a scale and location invariant detector. It is also a generic solution capable of adaptation for other types of image descriptors. The original algorithm was further developed by Lienhart and Maydt (2002), and most recently by Li et al. (2012). This last version of the original algorithm is the one adopted by the cascaded adaptive boosted classifiers supplied by OpenCV. However, training a classifier based on Haar-like features on a sufficiently large training set to give adequate recognition accuracy is extremely time consuming. The Haar-like features are also sensitive to changes in illumination and rotation, making them unsuitable for object recognition in many environments.

The Speeded Up Robust Features (SURF) descriptor and detector presented by Bay et al. (2006) addressed the limitations of object recognition approaches of the early 2000s in terms of speed and accuracy. Inspired by the original SIFT operator SURF was able to approximate and frequently outperform other contemporary solutions by using a Hessian matrix-based measure for the detector, a distribution-based descriptor, and relying on integral images for image convolutions. The original publication was later revisited by Bay et al. (2008). This version of the feature-based detector and descriptor was the field standard of object recognition in images for many years, and has subsequently been patented alongside SIFT.

Descriptors that rely on binary information to describe images have also become popular for use in object recognition. This is a result of the binary descriptors computational simplicity, and their inherent ability to store image information in a compact representation. An approach to object recognition that utilizes binary descriptors to consider both the shape and texture information of an entire image patch is presented by Ahonen et al. (2004). In their work the LBP histograms proposed by Ojala et al. (1995) are exploited to store image patch information in a compact fashion. Additionally, because the LBPs are calculated for each pixel in the image patch spatial information is also preserved. The compact representation of the LBPs enables the implementation to provide both a fast training process and fast recognition. The approach was further developed by Liao et al. (2007) to produce the Multi-scale Block Local Binary Pattern (MB-LBP). In this approach the LBPs are calculated based on averaged values of block sub-regions instead of individual pixels. This enables MB-LBPs to capture both micro- and macro-structures in the image patch, resulting in a more complete image representation. OpenCV provides MB-LBPs as one feature alternative in their implementation of the cascaded adaptive boosting classifiers.

A feature-based approach to object recognition that relies on binary feature descriptors is suggested by Calonder et al. (2010) with their proposed Binary Robust Independent Elementary Features (BRIEF). BRIEF is computed using simple intensity difference tests, allowing for fast computation. Similarity between descriptors can also be computed using the Hamming distance, which is more efficient than using other distance metrics to compare descriptors. In their work it is shown that BRIEF is capable of achieving results comparable to SURF while running at a fraction of the time. Another approach based on binary feature descriptors is the Binary Robust Invariant Scalable Keypoints (BRISK) proposed by Leutenegger et al. (2011). BRISK is both scale- and rotation-invariant to a significant degree, and offers performance and speed comparable to BRIEF. The Oriented FAST and Rotated BRIEF (ORB) binary feature descriptor presented by Rublee et al. (2011) combines the Features from Accelerated Segment Test (FAST) keypoint detector proposed by Rosten and Drummond (2006) with BRIEF. The resulting descriptor is further enhanced by applying a fast and accurate orientation component to the FAST keypoints. In their work they show that ORB displays similar performance to SURF in terms of rotation-invariance while operating at the speed of BRIEF. Additionally, ORB is not patented.

An appearance-based strategy for overcoming the obstacle of large data dimensionality in object recognition by utilizing Principal Component Analysis (PCA) was proposed by Nayart et al. (1996). In this implementation PCA is used to convert an image patch to a lower-dimensional representation that is the projection of the patch when viewed from the most informative viewpoint, instead of explicitly extracting features. This dimensionality reduction allows for faster comparison of data, as long as the reduction by PCA is fast. PCA also results in a dramatic reduction in data size which reduces the amount of memory needed by the application that utilizes it. In the work by Nayart et al. (1996) it is shown that an effective course grained multi-class classifier can be implemented in this way. PCA has also been successfully exploited in other object recognition frameworks such as by Malagón-Borja and Fuentes (2009) for pedestrian detection and by Dashore (2012) for face detection. PCA has been combined with many other object recognition strategies, such as by Ke and Sukthankar (2004), mentioned above. However, the dimensionality reduction performed by PCA is often too expensive for many applications because the complexity of the reduction scales cubicly with the number of features of the original data set. Methods based on PCA are also highly sensitive to partial occlusion, which further limits their use. Additionally, PCA is not suited for fine grained classification problems as the dimensionality reduction makes it hard to separate similar objects.

An alternative appearance-based method to PCA for dimensionality reduction of feature data in object recognition was introduced with the use of Linear Discriminant Analysis (LDA) by for their face detection framework. While PCA is an unsupervised technique, LDA takes into account the class labels of the training and test samples in an attempt to find a linear combination of features that best separates the classes. Thus, LDA is more suited for the classification tasks that are often found in object recognition of multiple objects. LDA has been used in many object recognition frameworks in recent years such as by Hariharan et al. (2012) and by Tian and Zhang (2017). LDA was also combined with the adaptive boosting (AdaBoost) algorithm used in the Viola & Jones object detection scheme by Nunn et al. (2009) to yield significantly improved discriminative power.

The focus of research in the field of object recognition changed altogether with the introduction of AlexNet by Krizhevsky et al. (2012). In their approach utilizing deep Convolutional Neural Nets (CNNs) they achieved detection error rates 11% less than any other entry to the ImageNet 2012 competition on object recognition whilst preserving scale, rotation, and illumination invariance. AlexNet was also able to handle partial occlution well in comparison to earlier strategies. The Region-based Convolutional Neural Net (RCNN) presented by Girshick et al. (2014) addressed the issue of object localization by introducing a region proposal layer to the neural net hierarchy. The RCNN was further developed by Girshick (2015) and Ren et al. (2015), greatly increasing both classification accuracy and speed. Other contributions over the last years include MultiBox by Erhan et al. (2014) as a region proposal solution, Single Shot MultiBox Detector (SSD) by Liu et al. (2015), and Residual Block Net (ResNet) by He et al. (2015). MultiBox was also used as the foundation for You Only Look Once (YOLO) by Redmon and Farhadi (2016). ResNet achieved extraordinary low error rates at 3.6%, being the first object recognition framework to surpass human performance. CNNs show great promise for the future of object recognition, however, the large amounts of image data needed to train a CNN for object recognition can pose problems in cases where annotated image material of the object is sparse.

Many possible approaches based on the aforementioned methods have been proposed for the more specific task of underwater object recognition. In the work presented by Kim and Yu (2017) the original Viola & Jones algorithm using Haar-like features is used to implement a real-time underwater object detection scheme for sonar images. An object recognition strategy that uses SURF for interest point detection and LBP for feature extraction and description is introduced by Prabhakar and Kumar (2012). The LBP operator is also exploited by Nagaraja et al. (2015) in which it is further developed to result in the Robust Local Binary Pattern (RLBP). The RLBP operator is then used to describe interest points for object recognition. Two fine-grained object recognition frameworks for fish species are presented by Spampinato et al. (2016); one based on an approximation of Local Ternary Patterns (LTPs) by the use of two LBPs and another that is a two-layer hybrid approach that employs either SIFT or LTPs at the upper level and Fisher Vectors or Sparse Coding in the second layer as an encoder. The work by Ren et al. (2015) was used by Li et al. (2015) to create a successfull real-time fish recognition framework. A comparison of the performance between a method based on a deep CNN and a method employing an SVM to learn Histogram of Oriented Gradients (HOG) features for recognizing fish in underwater images was conducted by Villon et al. (2016). In their work they show that the method based on a deep CNN significantly outperforms the method based on classifying HOG features with a SVM.

1.4 Outline

This thesis is organized as follows:

- **Chapter 2 - Theory:** This chapter describes the mathematical theory behind the image features, classifiers, and localization methods used in the object recognition implementations described in this thesis.
- **Chapter 3 - Implementation:** In this chapter an overview of the support vector machine classifier implementations is given. The chapter also explains the data flow for each implementation.
- **Chapter 4 - Experiments:** This chapter recounts each part of the experiments performed for this thesis. This chapter also explains how the training and test data was acquired and what pre-processing had to be completed before training.
- **Chapter 5 - Results:** This chapter presents the results of the experiments described in Chapter 4.
- **Chapter 6 - Discussion:** In this chapter a discussion of the results presented in Chapter 5 is given.
- **Chapter 7 - Further work:** This chapter comments on possible improvements to the proposed implementations. The chapter also discusses other further work.
- **Chapter 8 - Conclusion:** Finally, this chapter contains a conclusion to the work conducted for this thesis.

Chapter 2

Theory

This chapter defines key concepts within object recognition and describes the mathematical theory behind the image patch descriptors, image data classifiers and computer vision algorithms utilized in the object recognition frameworks presented in this thesis. Section 2.1 supplies definitions of terminology used throughout this thesis. The mathematical theory behind the image patch descriptors used in the object recognition frameworks presented in this thesis is described in Section 2.2. Section 2.3 explains the theory behind the Support Vector Machine (SVM) and how it is used in classifying data samples. Finally, the theory behind computer vision algorithms important to this thesis is described in Section 2.4.

2.1 Terminology

Because of the extensive amount of research papers that exist on the topic of object recognition a number of different definitions for key concepts have previously been used. This section seeks to clarify the definitions of such concepts that are used in this thesis.

Definition 2.1.1 (Object detection)**:** *Within computer vision object detection is the task of determining whether an object of a certain class is present within an image. This normally translates to processing an image to output a true/false response.*

Definition 2.1.2 (Object classification)**:** *Object classification within computer vision is the task of assigning semantic labels to a objects present in an image. Object classification is typically separated into two categories, binary object classification and multi-class object classification.*

Definition 2.1.3 (Object localization)**:** *In computer vision object localization is the task of identifying where in an image a certain object is located. The objective of object localization is often to place a bounding box around an object to visualize the location of the object in the image.*

Definition 2.1.4 (Object recognition)**:** *Object recognition within computer vision can be seen as the task of supplying a single system that performs all of the above defined tasks* *of object detection, binary or multi-class object classification and object localization. The objective of such a system is to place bounding boxes visualizing the location of all objects of certain classes within an image and label the boxes with the correct class label.*

2.2 Local binary pattern features

This section contains the theory behind the LBP image patch descriptors used in the object recognition frameworks presented in this thesis. The theory behind the original LBP image patch descriptor is detailed in Subsection 2.2.1. The theory behind the further developed Uniform Local Binary Pattern (LBP-U2), Rotation Invariant Local Binary Pattern (LBP-RI), Rotation Invariant Uniform Local Binary Pattern (LBP-RIU2) and Local Binary Pattern Fourier Histogram (LBP-HF) image patch descriptors is then presented in the subsequent subsections, and arguments for and against the use of these different adaptations are given.

2.2.1 LBP

The Local Binary Pattern (LBP) image patch descriptor, originally introduced by Ojala et al. (1995), is a descriptor that relies on binary texture information to describe the surroundings of a pixel. This is achieved by comparing the intensity values of the surrounding pixels to that of the center pixel. The intensity value of a pixel is a single number signifying the brightness within a certain color spectrum of that pixel. The LBP image patch descriptor is usually applied to 8-bit grayscale images in which each pixel of the image is represented by an integer value between 0 and 255 specifying the pixel intensity. The same image in color and grayscale formats can be seen in Figure 2.1.

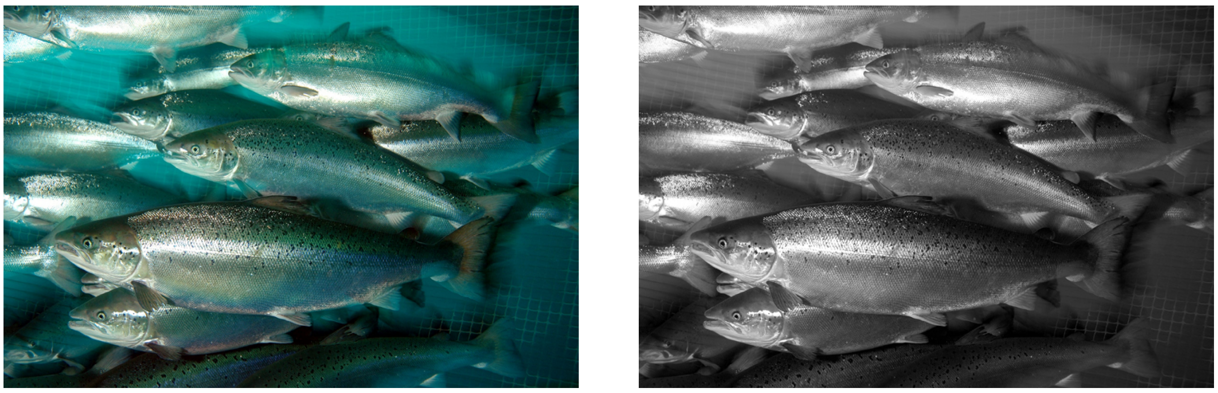
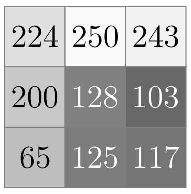
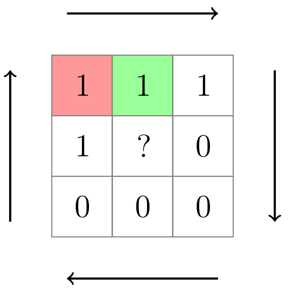

(a) Color format. **(b)** Grayscale format.

Figure 2.1: Image taken by Svensen (2016) showing color and grayscale formats of a scene.

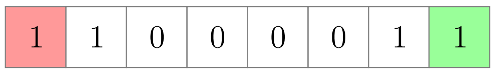
The simplest form of the LBP image patch descriptor is computed by considering the three-by-three neighbourhood of a center pixel in a grayscale image. The intensity values of the neighbouring pixels are each compared to that of the center pixel in a clockwise or counter-clockwise manner, starting with the top pixel. If the intensity value of a pixel is greater than or equal to that of the center pixel a 1 is returned. If the intensity value of a pixel is less than that of the center pixel a 0 is returned. This comparison is shown in Figures 2.2a and 2.2b. The values returned by the comparisons are used to form a 8-bit binary string by setting the bits of the string as the values are returned, starting with the Least Significant Bit (LSB). This binary string representation is shown in Figure 2.2c. The binary string is then converted into an integer decimal value between 0 and 255 which is used as the new intensity value for the center pixel. This results in a compact image patch representation where the location and intensity information of the patch can be stored as a single unsigned 8-bit value. The conversion from binary string to decimal number and the new representation of the center pixel is shown in Figures 2.2d and 2.2e respectively.



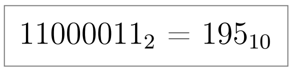
(a) Pixel intensity representation.



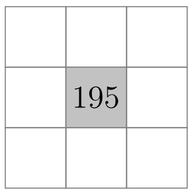
(b) Comparison of pixel intensity values.



(c) Binary string representation.



(d) Binary to decimal number conversion.



(e) Represented as a single integer.

Figure 2.2: Computation of the original LBP image patch descriptor.

However, the simple form specified for the three-by-three neighbourhood is only a special case of the general LBP image patch descriptor, which can describe a neighbourhood consisting of an arbitrary number of pixels situated at any radius around a center pixel. To derive the general LBP image patch descriptor a neighbourhood N of a pixel is first defined by Equation 2.1

$$N = [p_c, p_0, ..., p_i], \qquad i \in [0, P-1] \tag{2.1}$$

where p_c are the coordinates of the center pixel, p_i are the coordinates of the neighbouring pixels, and P is the number of pixels in the neighbourhood. The intensity values V of the neighbouring pixels are given by Equation 2.2

$$V = [v_c, v_0, ..., v_i], \qquad i \in [0, P-1] \tag{2.2}$$

where v_c is the intensity value of the center pixel, v_i are the intensity values of the neighbouring pixels and P is the number of pixels in the neighbourhood. The neighbouring pixels p_i are located on the radius of distance R from p_c and are evenly spaced according to Equation 2.3.

$$p_i = [x_i, y_i] = \left[x_c + R \cdot \cos\left(\frac{2\pi \cdot i}{P}\right), y_c + R \cdot \sin\left(\frac{2\pi \cdot i}{P}\right)\right] \tag{2.3}$$

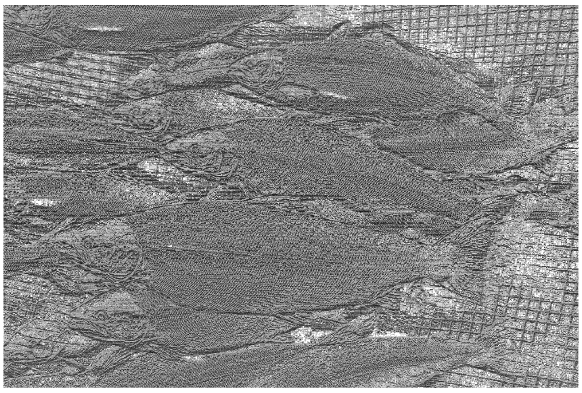
The comparison of pixel intensity values as described above for the three-by-three neighbourhood is then performed on the pixels in the neighbourhood N of p_c. If the coordinates of a neighbouring pixel are located between real pixel values the pixel intensity value v_i of that pixel is interpolated, i.e approximated, according to an interpolation algorithm depending on the neighbouring pixels of p_i. There exists a number of different interpolation algorithms with varying accuracy and computational complexity, but because an extensive review of these is beyond the scope of this thesis they will not be explored further. For future reference it is assumed that the bilinear interpolation scheme as defined by Wikipedia (2017) is used if not otherwise stated. The binary values are returned as defined by the step function shown in Equation 2.4.

$$s(v) = \begin{cases} 1, & \text{if } v \geq v_c \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

The step function is employed in order to retain the texture information of the image patch while discarding what is regarded as scaling of that information. This allows the LBP descriptor to describe different texturer such as lines, curves and edges while also remaining invariant to changes in illumination. The integer decimal value describing the P pixels evenly spaced at a distance R from a center pixel p_c is defined by the LBP code given by Equation 2.5

$$\mathrm{LBP}_{\mathrm{P,R}}\left(p_c\right) = \sum_{i=0}^{P-1} s\left(v_i\right) \cdot 2^i, \qquad v_i \in V \tag{2.5}$$

where P are the number of pixels in the neighbourhood, R is the distance from the center pixel to the neighbouring pixels and v_i is the intensity value of the neighbouring pixel with coordinates p_i. LBP codes computed over an entire image using different values for the radius R and number of neighbouring points P are shown in Figures 2.3 and 2.4 respectively. The image is first divided into a number image patches defined by the chosen radius R. The LBP code using P neighbours for each image patch is then computed and the original image is updated to show the grayscale intensity values corresponding to the codes.


(a) Radius = 1, number of points = 8.

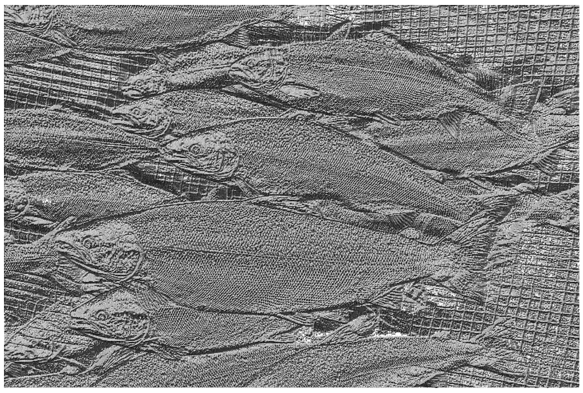
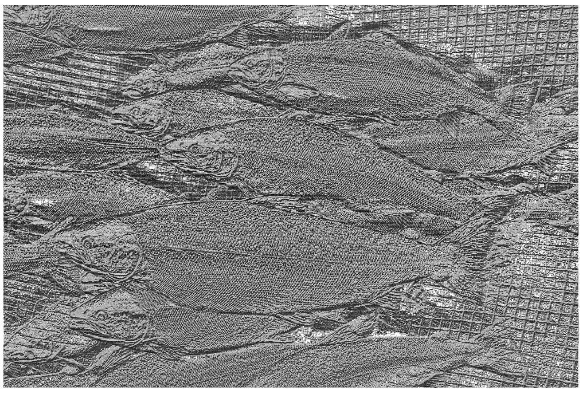

(b) Radius = 3, number of points = 8.

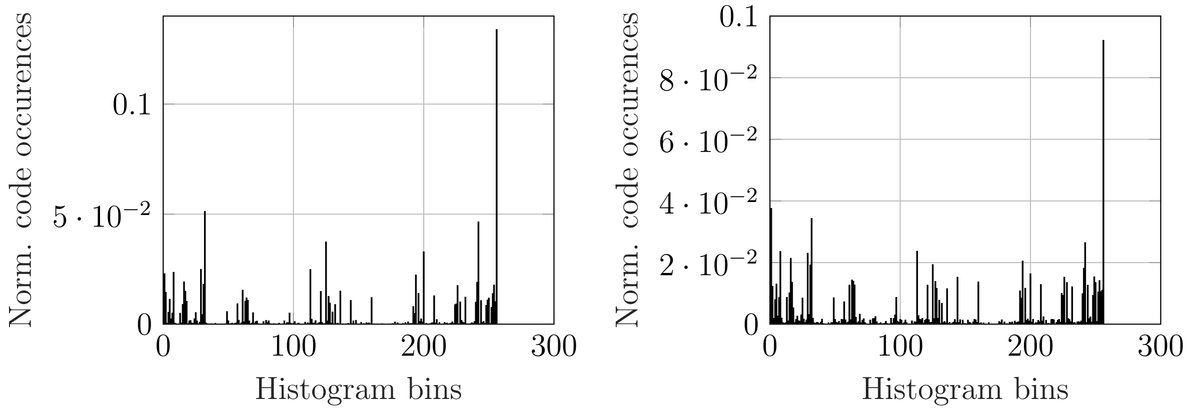
Figure 2.3: LBP images computed using different radii. Number of pixels used are constant.


(a) Radius = 2, number of points = 8.

(b) Radius = 2, number of points = 24.

Figure 2.4: LBP images computed using different numbers of pixels. The radius is constant.

Clearly, the image type representation of the LBP codes shown in Figures 2.3 and 2.4 is of too high dimensionality to be usefull in a object recognition framework. By forming a histogram consisting of 256 bins, one bin for each gray level of a 8-bit LBP image a more suitable representation can be acquired. This histogram representation can also be used as a feature vector that can be used in object recognition tasks. However, this representation has the cost of loss of information in terms of image patch locations. Histograms displaying the LBP code distribution of an image where the LBP codes are calculated using different values for the radius R and the number of neighbouring pixels P are kept constant are shown in Figure 2.5.



(a) Radius = 1, neighbouring pixels = 8. **(b)** Radius = 2, neighbouring pixels = 8.

Figure 2.5: Histograms displaying the distribution of LBP codes when calculated with different radii and constant number of neighbouring pixels.

Even though the LBP image patch descriptor delivers a compact and representative description of a grayscale image at a low computational cost it is limited by the fact that the binary code returned for a center pixel p_c only supplies information on the neighbourhood of pixels situated at a specific distance R from the center pixel. This means that a LBP code created with a smaller radius will be unable to detect texture patterns of a lower frequency, while a LBP code created with a larger radius will be unable to detect texture patterns of a higher frequency. An obvious solution to this limitation is to combine LBP codes of different radii into a single representation that is able to describe texture patterns of both high and low frequency. However, this approach is often restricted by the computational complexity of LBP codes created using a larger radius because the number of individual LBP codes double with the addition of each neighbouring point.

2.2.2 LBP-U2

The Uniform Local Binary Pattern (LBP-U2) image patch descriptor presented by Topi et al. (2000) is based on the important observation that some LBP codes have a much higher rate of occurence in natural images than others. The greater part of circular defined LBP codes, i.e codes defined as in Subsection 2.2.1, contain at most two $1 \rightarrow 0$ or $0 \rightarrow 1$ transitions. The LBP codes containing at most two of these transitions are defined as LBP-U2s. An example of a uniform and a non-uniform LBP code is shown in Figure 2.6.

0	1	0	0	0	0	0	0

(a) Uniform Local Binary Pattern code.

0	1	0	1	0	1	0	1

(b) Local Binary Pattern code.

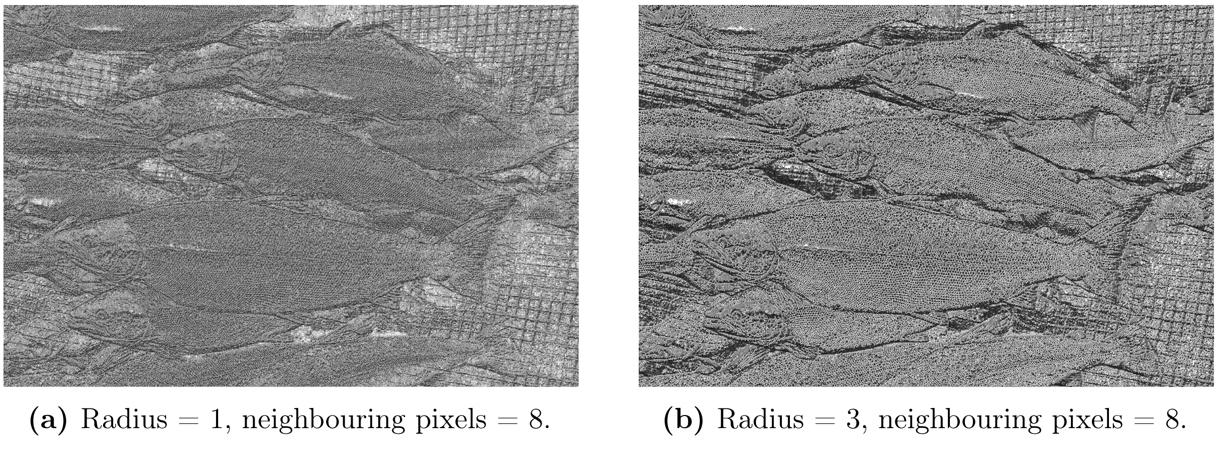
Figure 2.6: Examples of LBP-U2 and LBP codes.

A LBP code can be checked for uniformity by summarizing the absolute value of the difference between the code and the code circularly shifted one bit. An algorithm for testing for uniformity is included in the work by Lindahl (2007) and is defined by Equation 2.6

$$U\left(LBP_{R,P}\right) = \begin{cases} \text{true}, & \text{if } \left|s\left(v_{P-1}\right) - s\left(v_0\right)\right| + \sum_{i=1}^{P-1} \left|s\left(v_i\right) - s\left(v_{i-1}\right)\right| \leq 2 \\ \text{false}, & \text{otherwise} \end{cases} \tag{2.6}$$

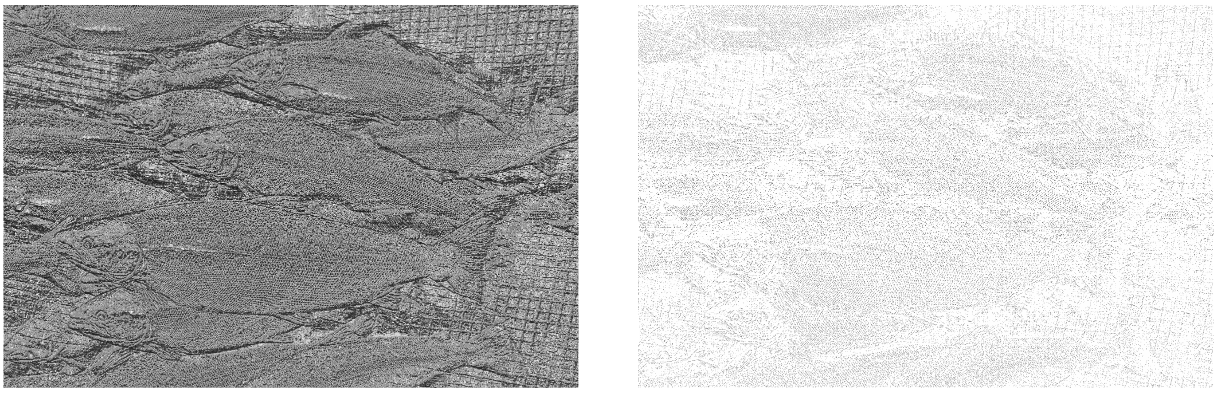
where $LBP_{R,P}$ is the LBP code for a grayscale image patch computed using P neighbouring pixels situated at the radius of distance R from a center pixel p_c, v_i are the intensity values of the neighbouring pixels, and $s(v)$ is the step function as defined by Equation 2.4.

By requiring that LBP codes be uniform it is possible to reduce the number of possible LBP codes from 2^P to $P \cdot (P-1) + 2$. Because the LBP codes that satisfy the requirement of being uniform have a much higher occurence rate than other codes this can be done without a significant loss in discriminative power. The use of LBP-U2 codes in the three-by-three case mentioned in Subsection 2.2.1 reduces the number of required bins in the LBP code distribution histogram from 256 to 59, one for each unique LBP-U2 code and one bin for the remaining LBP codes. This enables the comparison of LBP-U2 codes to be much faster than for LBP codes and also reduces the risk of high frequency noise contaminating the codes. LBP-U2 codes computed over an entire image using different values for the radius R and number of neighbouring pixels P are shown in Figures 2.7 and 2.8 respectively. It should be noted that these images are almost identical to Figures 2.3 and 2.4, another testament to the high occurence of LBP-U2 codes in natural images.



(a) Radius = 1, neighbouring pixels = 8. **(b)** Radius = 3, neighbouring pixels = 8.

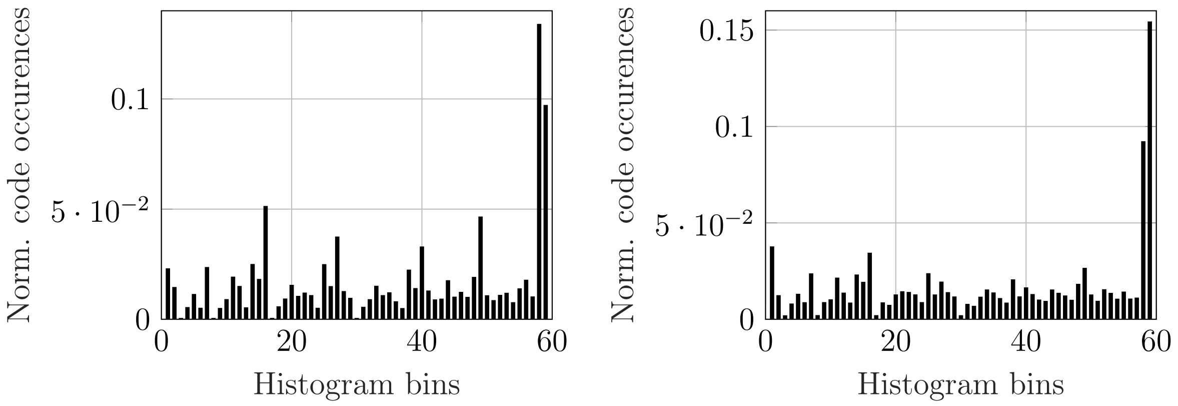
Figure 2.7: LBP-U2 images computed using different radii. Number of pixels used are constant.



(a) Radius = 2, neighbouring pixels = 8. **(b)** Radius = 2, neighbouring pixels = 24.

Figure 2.8: LBP-U2 images computed using different numbers of pixels. The radius is constant.

Histograms displaying the LBP-U2 code distribution of an image where the LBP-U2 codes are calculated using different values for the radius R and a constant number of neighbouring pixels P are shown in Figure 2.9.



(a) Radius = 1, neighbouring pixels = 8. **(b)** Radius = 2, neighbouring pixels = 8.

Figure 2.9: Histograms displaying the distribution of LBP-U2 codes when calculated with different radii and constant number of neighbouring pixels.

2.2.3 LBP-RI

One of the disadvantages of the LBP and LBP-U2 image patch descriptors is that they do not provide a rotation invariant description. When a grayscale image patch is rotated, the intensity values of the pixels lying on the radius of distance R from a center pixel p_c will move along the radius with the rotation. Thus, the LBP code of that image patch changes because the pixels used in the computation of the LBP code are changing while the LBP code is calculated in the same way regardless of rotation. The only LBP codes that do not change under rotation are codes consisting of only one type of binary value i.e codes consisting of only 1's or 0's. To eliminate the effects of rotation upon the LBP codes an adaptation of the LBP image patch descriptor is proposed by T. et al. (2002) called Rotation Invariant Local Binary Pattern (LBP-RI).

In their approach the LBP codes computed by performing the comparison described in Subsection 2.2.1 are submitted to a bit-wise right shift of the Most Significant Bit (MSB) for each neighbouring pixel used in the computation of the code i.e P times. The code that has the maximum number of MSBs set to 0 is chosen as the new representation of the LBP code. The execution of the bit-wise right shift is defined in the original paper by T. et al. by Equation 2.7

$$LBP^{ri}_{R,P} = \min\{ROR\left(LBP_{R,P}, i\right), \qquad i \in (0, 1, ..., P-1)\} \tag{2.7}$$

where $LBP^{ri}_{R,P}$ is the rotation invariant form of the previously calculated LBP code $LBP_{R,P}$, P are the number of pixels used to compute the original code and $ROR(x, i)$ is the rotational OR operation. An example of a LBP code before and after the bit-wise right shift is shown in Figure 2.10

0	1	0	0	0	1	0	1

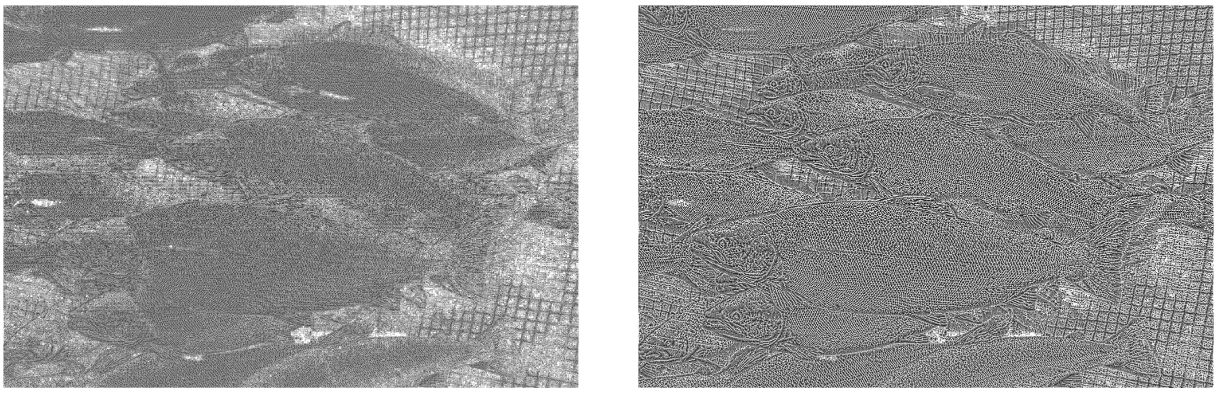
(a) LBP code.

0	0	0	1	0	1	0	1

(b) LBP-RI code.

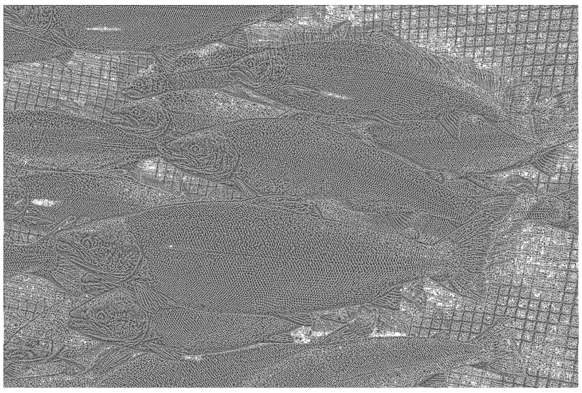
Figure 2.10: Examples of LBP and LBP-RI codes.

The bit-wise right shift of the original LBP codes has the additional effect of reducing the number of possible LBP-RI codes to $\frac{P}{2} \cdot (P+1)$, further reducing complexity of comparisons between LBP-RI codes, albeit at a loss to general discriminative power. For the three-by-three image patch described in Subsection 2.2.1 this results in 36 unique LBP-RI codes. LBP-RI codes computed over an entire image using different values for the radius R and number of neighbouring pixels P are shown in Figures 2.11 and 2.12 respectively.



(a) Radius = 1, neighbouring pixels = 8. **(b)** Radius = 3, neighbouring pixels = 8.

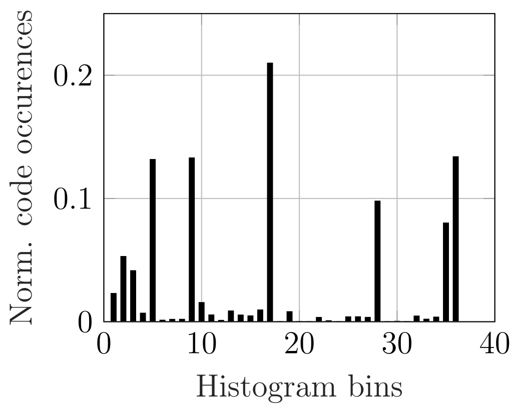
Figure 2.11: LBP-RI images computed using different radii. Number of pixels are constant.


(a) Radius = 2, neighbouring pixels = 8.

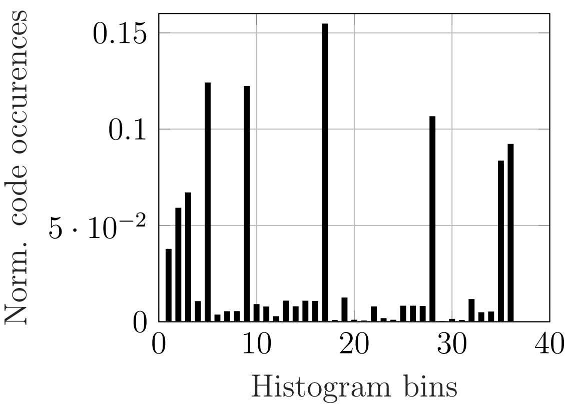
(b) Radius = 2, neighbouring pixels = 24.

Figure 2.12: LBP-RI images computed using different numbers of pixels. The radius is constant.

Histograms displaying the LBP-RI code distribution of an image where the LBP-RI codes are calculated using different values for the radius R and a constant number of neighbouring pixels P are shown in Figure 2.13.



(a) Radius = 1, neighbouring pixels = 8.



(b) Radius = 2, neighbouring pixels = 8.

Figure 2.13: Histograms displaying the distribution of LBP-RI codes when calculated with different radii and constant number of neighbouring pixels.

2.2.4 LBP-RIU2

The Rotation Invariant Uniform Local Binary Pattern (LBP-RIU2) image patch descriptor, also presented by T. et al. (2002), combines the advantages of the previously defined LBP-U2s by Topi et al. (2000) with the approach to rotation invariance proposed with LBP-RI. To compute the LBP-RIU2 code of an image patch the LBP-RI code of the image patch is first computed. If the LBP-RI code is uniform as defined by Equation 2.6 the code is considered a LBP-RIU2 code. The process of computing LBP-RIU2 codes is shown in Equation 2.8

$$LBP_{R,P}^{riu2} = \begin{cases} \sum_{i=0}^{P-1} s\left(v_i\right), & \text{if } U\left(LBP_{R,P}^{ri}\right) \\ P+1, & \text{otherwise} \end{cases} \tag{2.8}$$

where $LBP_{R,P}^{riu2}$ are the uniform set of precomputed LBP-RI codes, $LBP_{R,P}^{ri}$ are the precomputed LBP-RI codes, P are the number of pixels used in the computation of each LBP-RI code, $s(v)$ is the step function as defined by Equation 2.4, v_i are the intensity values of each neighbouring pixel and $U()$ is the check for uniformity as defined by Equation 2.6.

By definition there exists $P+1$ unique uniform codes for a circularly symetric neighbourhood consisting of P pixels around a center pixel p_c. Thus $P+2$ codes are needed for a complete description, one for each of the possible LBP-RIU2 codes and one code for the remaining non-uniform LBP-RI codes. For the three-by-three neighbourhood defined in Subsection 2.2.1 this results in 10 unique codes. This results in a highly compact image patch representation and enables the possibility of fast comparisons of LBP-RIU2 codes. LBP-RIU2 codes computed over an entire image using different values for the radius R and number of neighbouring pixels P are shown in Figures 2.14 and 2.15 respectively.

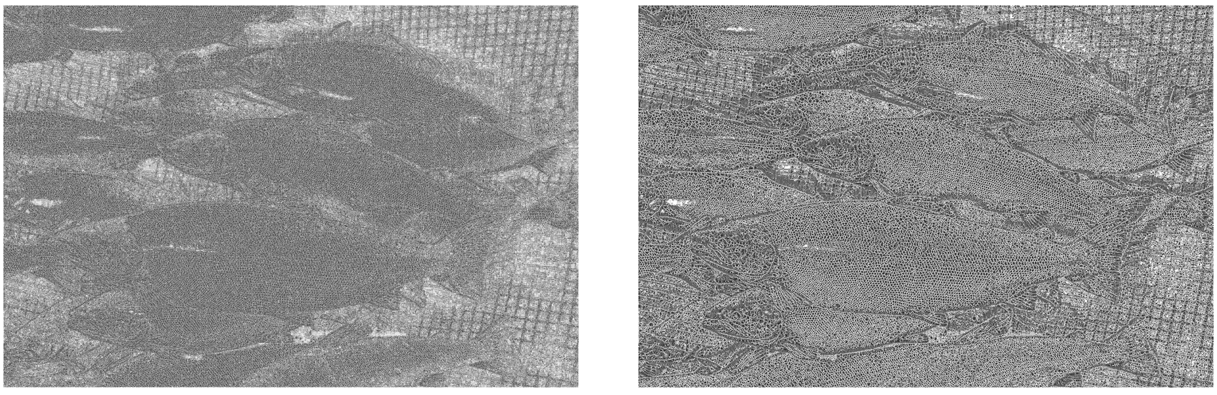

(a) Radius = 1, neighbouring pixels = 8. **(b)** Radius = 3, neighbouring pixels = 8.

Figure 2.14: LBP-RIU2 images computed using different radii. Number of pixels are constant.

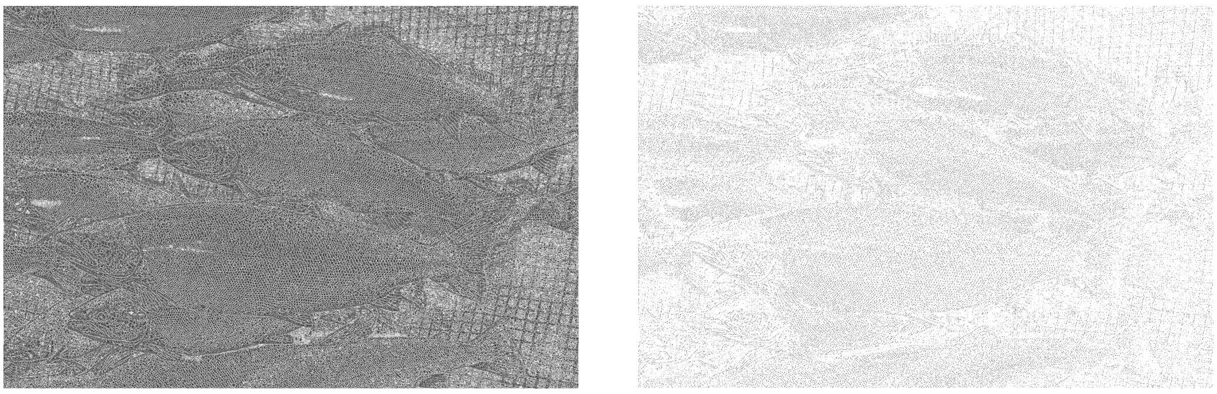
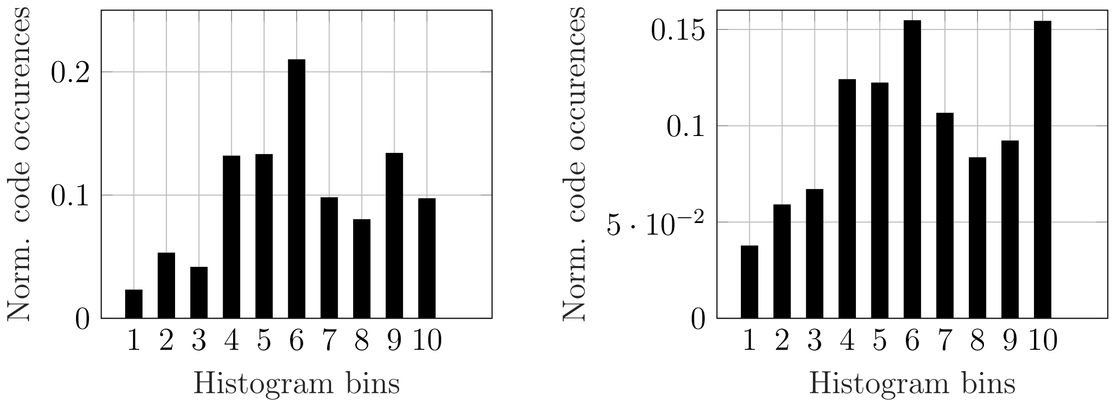

(a) Radius = 2, neighbouring pixels = 8. (b) Radius = 2, neighbouring pixels = 24.

Figure 2.15: LBP-RIU2 images computed using different numbers of pixels. The radius is constant.

The low number of unique LBP-RIU2 codes used in the description of each image patch is reflected in the number of bins used for the LBP-RIU2 code distribution histograms. Histograms displaying the LBP-RIU2 code distribution of an image where the LBP-RIU2 codes are calculated using different values for the radius R and a constant number of neighbouring pixels P are shown in Figure 2.16.



(a) Radius = 1, neighbouring pixels = 8. (b) Radius = 2, neighbouring pixels = 8.

Figure 2.16: Histograms displaying the distribution of LBP-RIU2 codes when calculated with different radii and constant number of neighbouring pixels.

However, there is one drawback to the approach used in achieving rotation invariance for the LBP-RI and LBP-RIU2 image patch descriptors; the quantization of the angular space is dependent on the number of neighbouring pixels P used in the computation of the LBP-RI code. Since the number of possible LBP-RI and LBP-RIU2 codes grow for each neighbouring pixel used in the computation of the codes achieving a high level of rotation invariance comes at the cost of higher computational complexity. For the three-by-three neighbourhood defined in Subsection 2.2.1 only a coarse 45° level of rotation invariance is possible when calculated using the eight neighbouring pixels of a center pixel p_c.

2.2.5 LBP-HF

An alternative approach to achieving a rotation invariant adaptation of the LBP-U2 image patch descriptor is proposed by T. et al. (2009). In their work they show that their Local Binary Pattern Fourier Histogram (LBP-HF) image patch descriptor is capable of outperforming the LBP-RIU2s proposed by T. et al. (2002) in terms of being invariant to rotation while retaining the highly descriptive power of the LBP-U2 image patch descriptor.

In the work by T. et al. the LBP-HF is derived by first denoting a specific LBP-U2 code by $U_P(n, r)$. The pair (n, r) specifies the LBP-U2 so that n is the number of bits set to 1 in the pattern and r is the rotation of the pattern. For a LBP-U2 computed using P neighbouring pixels n has values from 0 to $P + 1$, where the label $P + 1$ is the label used for all non-uniform patterns. Consequently, the rotation of the pattern describing an image patch is in the range $0 \leq r \leq P - 1$ when $1 \leq n \leq P - 1$. The rotation of a point (x, y) to the location (x', y') is then denoted by $I^\alpha(x, y)$ where α is the rotation in degrees. Thus, a circular sampling neighbourhood placed around $I(x, y)$ will also be rotated by α degrees if $I(x, y)$ is rotated. The allowed rotations of the sampling neighbourhood are limited to integer multiples of the angle betwen two sampling points by requiring that $\alpha = a \cdot \frac{360\,\text{deg}}{P}, a = 0, 1, ..., P - 1$ which allows rotation of the sampling neighbourhood by a discrete steps. This requirement in turn results in that the LBP-U2 code denoted $U_P(n, r)$ at point (x, y) is replaced by the LBP-U2 code denoted $U_P(n, r + a \mod P)$ after a rotation to point (x', y').

The histogram showing the LBP-U2 code distribution for a image is then denoted as $h_I(U_P(n, r))$. By the reasoning of the above paragraph a rotation of the input image I by $\alpha = a \cdot \frac{360\,\text{deg}}{P}$ causes a cyclic shift in the LBP-U2 histogram along n such that $h_{I^\alpha}(U_P(n, r + a)) = h_{I(U_P}(n, r))$. The LBP-HF approach is based on exploiting this property by using the Discrete Fourier Transform (DFT) to construct the new features. The DFT is used to transform a sequence of N complex numbers into another equally long sequence that is a complex-valued function of frequency. The DFT is defined mathemtically as shown by Equation 2.9

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{\frac{-i2\pi kn}{N}} \tag{2.9}$$

where $x_0, x_1, ..., x_{N-1}$ is the initial set of complex numbers and $X_0, X_1, ..., X_{N-1}$ is the DFT of that set. $H(n, \cdot)$ is then defined as the DFT of the set of LBP-U2 codes in $h_I(U_P(n, r))$ containing n 1's. This is defined as shown by Equation 2.10.

$$H(n, u) = \sum_{r=0}^{P-1} h_I(U_P(n, r)) \cdot e^{\frac{-i2\pi ur}{P}} \tag{2.10}$$

It can be shown that a cyclic shift of the input vector of the DFT results in a phase shift of the coeffisients of the DFT. By defining the LBP-U2 histogram after a rotation to be $h'\left(U_P\left(n,r\right)\right) = h\left(U_P\left(n,r-a\right)\right)$, then

$$H'\left(n,u\right) = H\left(n,u\right)\cdot e^{\frac{-i2\pi ua}{P}}$$

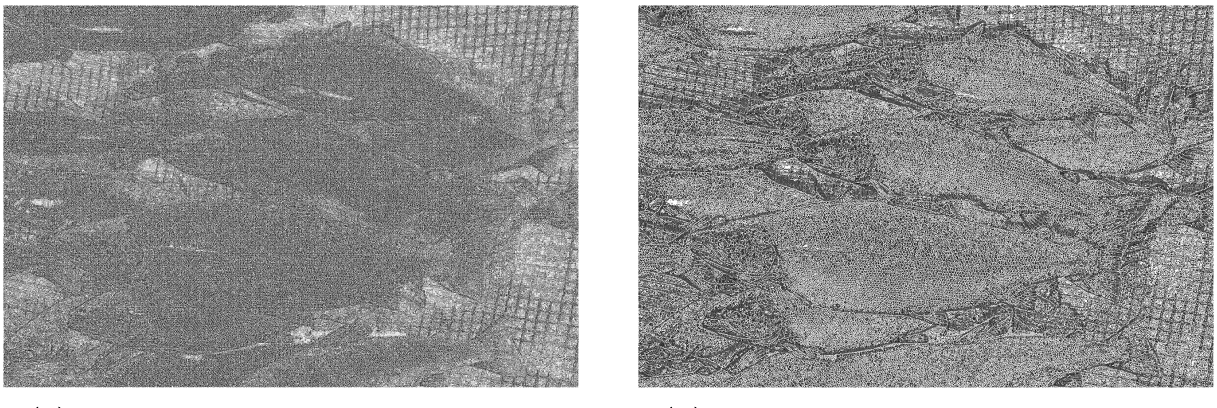
and, as a consequence, for any $1 \leq n_1$ and $n_2 \leq P-1$,

$$H'\left(n_1,u\right)\overline{H'\left(n_2,u\right)} = H\left(n_1,u\right)\cdot e^{\frac{-i2\pi ua}{P}}\overline{H\left(n_2,u\right)}\cdot e^{\frac{i2\pi ua}{P}} = H\left(n_1,u\right)\overline{H\left(n_2,u\right)}$$

in which the complex conjugate of $H\left(n_2,u\right)$ is denoted by $\overline{H\left(n_2,u\right)}$. The above reasoning shows that the features calculated from LBP-U2 codes and defined by Equation 2.11

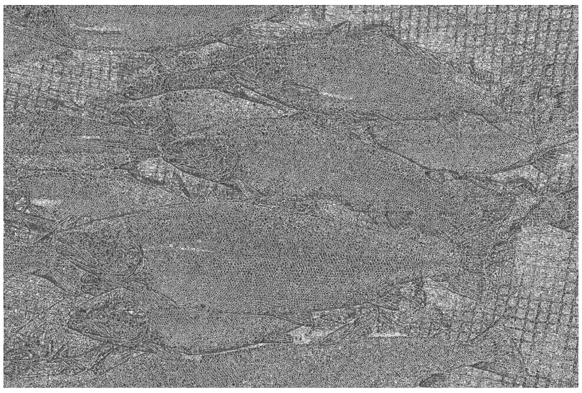
$$\text{LBP-HF}\left(n_1,n_2,u\right) = H\left(n_1,u\right)\overline{H\left(n_2,u\right)} \tag{2.11}$$

with any $1 \leq n_1,\ n_2 \leq P-1$ and $0 \leq u \leq P-1$ are invariant to cyclic shifts of the placements of 1's in $h_I\left(U_P\left(n,r\right)\right)$ and thus also invariant to rotations of the input image. The transformation from LBP-U2 codes to LBP-HF codes reduces the number of unique codes needed to describe a grayscale image patch from $59 \rightarrow 36$ whilst retaining the high level of description of the LBP-U2 image patch descriptor. LBP-HF codes computed over an entire image using different values for the radius R and number of neighbouring pixels P are shown in Figures 2.17 and 2.18 respectively.



(a) Radius = 1, neighbouring pixels = 8. **(b)** Radius = 3, neighbouring pixels = 8.

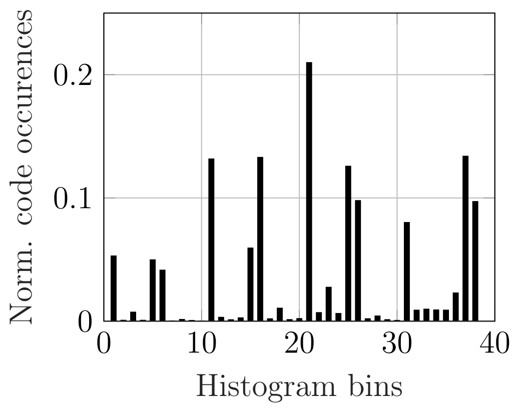
Figure 2.17: LBP-HF images computed using different radii. Number of pixels are constant.


(a) Radius = 2, neighbouring pixels = 8.

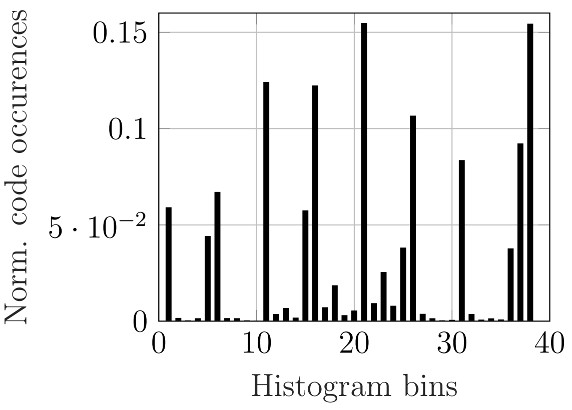
(b) Radius = 2, neighbouring pixels = 24.

Figure 2.18: LBP-HF images computed using different numbers of pixels. The radius is constant.

The LBP-HF code distribution of an image where the LBP-HF codes were calculated using different values for the radius R and a constant number of neighbouring pixels P are displayed in the histograms shown in Figure 2.19.



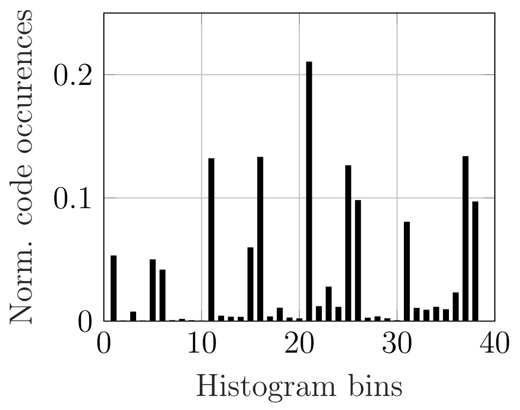
(a) Radius = 1, neighbouring pixels = 8.



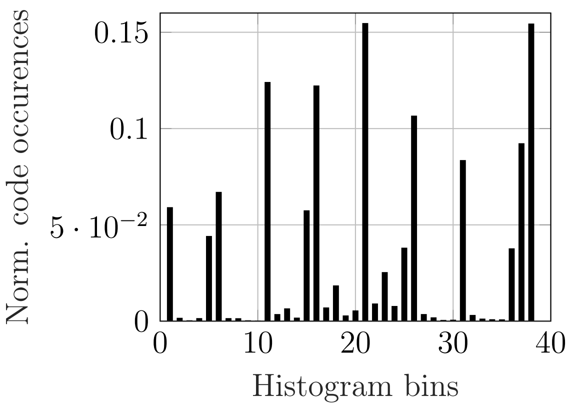
(b) Radius = 2, neighbouring pixels = 8.

Figure 2.19: Histograms displaying the distribution of LBP-HF codes when calculated with different radii and constant number of neighbouring pixels.

In Figure 2.20 histograms displaying the LBP-HF code distribution of the same image as in Figure 2.19 calculated using the same parameters after the image had been rotated −90° is shown. The histograms before and after the rotation are nearly identical, supplying a visual demonstration of the robustness to rotation of LBP-HF codes.



(a) Radius = 1, neighbouring pixels = 8.



(b) Radius = 2, neighbouring pixels = 8.

Figure 2.20: Histograms displaying the distribution of LBP-HF codes when calculated with different radii and constant number of neighbouring pixels after the original image was rotated $-90°$.

2.3 Support vector machine classifiers

This section describes the concepts and mathematical theory behind the SVM, and explains how SVMs can be utilized to classify image patches for the object recognition framework presented in this thesis by utilizing feature vectors as the input vectors to the SVM. Subsection 2.3.1 provides a general overview of the SVM approach, while the linear and non-linear adaptations of the SVM are explained in Subsections 2.3.2 and 2.3.3 respectively. Different approaches to solving the mathematical problem constructed by the SVM are detailed in Subsection 2.3.4. En explanation of how multi-class classification can be achieved from binary SVM classifiers is included in Subsection 2.3.5.

2.3.1 General SVM classification

The Support Vector Machine (SVM) is a type of supervised learning model that within machine learning is used along with associated learning algorithms to analyze data used for tasks such as classification. SVMs were first introduced by Vapnik and Lerner (1963), but has later been revised a number of times. In general, SVMs are defined as *non-probabilistic binary linear classifiers*, as the SVM training algorithm will attempt to build a model that assigns one of two labels to a query sample by relying on a set of s labeled training samples each consisting of p real numbers. The model that the SVM training algorithm outputs is a representation of the training samples as points in a space of some dimension that allows the training samples to be separated by a gap with the maximum width possible. Classification is then performed by assigning one of the two labels to a query sample based on which side of the gap the query sample is located after being mapped to the same dimension as the training samples. Thus, the task of classifying a query sample develops into the task of finding a hyperplane in a dimension that separates the labeled samples in a way so that the gap between them is maximized.

Figure 2.21 displays the space of a two-dimensional SVM model. The labels of the training samples are here shown as the colors black and white, and the unclassified query sample q_1 is shown in green. Hyperplanes in the form of lines crossing the model space are shown in h_1, h_2 and h_3. Clearly, h_3 does not seperate the labeled training samples and so is not a solution. h_3 separates the labeled training samples, but is not the optimal solution as the gap between the training samples is not maximized. The optimal solution is shown by h_1. Thus, the query sample q_1 will be labeled black after classification. The maximum gap between the labeled training samples is shown as consisting of the margins m_1 and m_2 on either side of h_1.

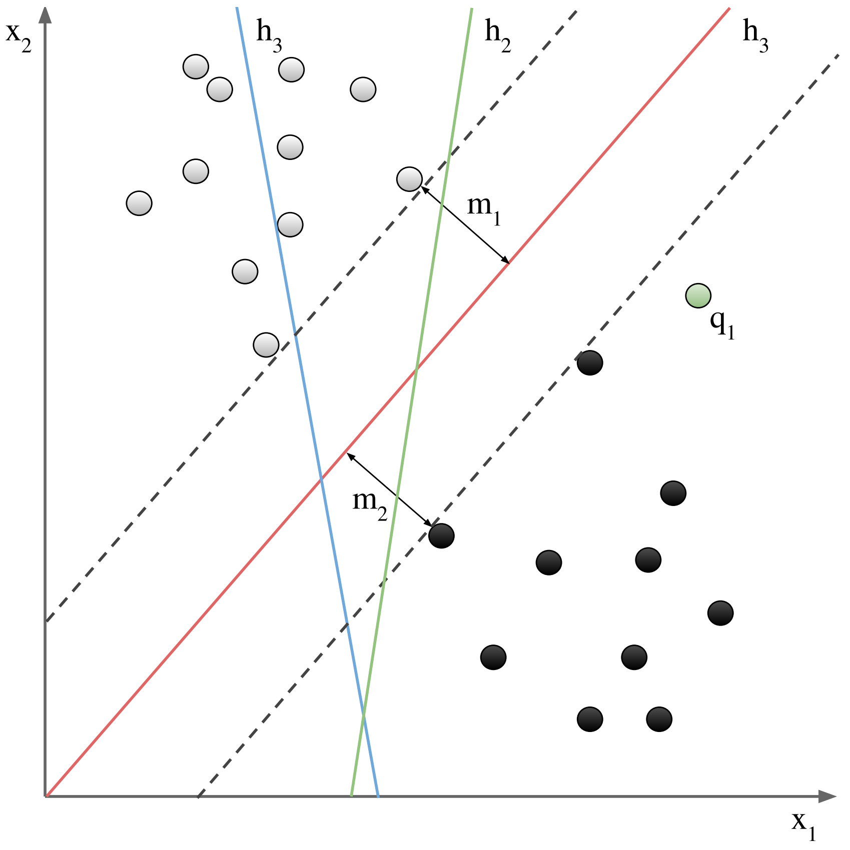


Figure 2.21: Support Vector Machine example plot.

2.3.2 Linear SVM classification

Linear SVMs are defined as SVMs where a set S of n labeled training samples each consisting of p numbers are separated by a hyperplane of dimension $p-1$. The example detailed in Subsection 2.3.1 and shown by 2.21 is an example of a linear SVM. The behaviour characterized by this example can be described formally by deriving the mathematical definition of linear SVMs. Suppose that a set S of labeled training samples is defined by

$$S = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \dots, (\vec{x_n}, y_n)\} \tag{2.12}$$

where y_i is the class of the training sample x_i represented by the integer value 1 or -1 and

$\vec{x_i}$ is a vector of p real numbers so that the training samples are linearly seperable. The objective of the SVM is then to find the hyperplane dividing the training samples $\vec{x_i}$ into groups according to whether they have the label $y_i = 1$ or $y_i = -1$ that has the largest distance on either side of the hyperplane to any training sample. This is known as the *maximum margin hyperplane*. Hyperplanes of any dimension can be defined as

$$\vec{w} \cdot \vec{x} - b = 0$$

where $\vec{w}$ is the normal vector to the hyperplane. The offset of the hyperplane from the origin along the normal vector $\vec{w}$ is determined by the parameter $\frac{b}{||\vec{w}||}$. Let two hyperplanes that separate the two classes of samples labeled as 1 or -1 be defined by

$$\vec{w} \cdot \vec{x} - b = 1 \tag{2.13a}$$

and

$$\vec{w} \cdot \vec{x} - b = -1 \tag{2.13b}$$

so that the distance between the two hyperplanes is as large as possible. The maximum margin hyperplane as defined above is then the hyperplane that lines halfway between them. The distance from the maximum margin hyperplane to the two defined hyperplanes is defined as the margin of that hyperplane. To make sure that no training samples fall inside the margin of the maximum margin hyperplane contraints are added for each training sample in S

$$\vec{w} \cdot \vec{x_i} - b \geq 1, \qquad \text{if } y_i = 1 \tag{2.14a}$$

and

$$\vec{w} \cdot \vec{x_i} - b \leq -1, \qquad \text{if } y_i = -1 \tag{2.14b}$$

so that all training samples from either class lie on the correct side of their respective margins on each side of the maximum margin hyperplane. The above constraints can then be rewritten as

$$y_i \left(\vec{w} \cdot \vec{x_i} - b\right) \geq 1, \qquad \forall \quad 1 \leq i \leq n \tag{2.15}$$

where n is the number of training samples in S.

Maximizing the distance between the two defined hyperplanes, and thus maximizing the margins of the hyperplane that lies halfway between them, amounts to minimizing $\vec{w}$, because the distance between the two defined hyperplanes is given by $\frac{2}{||\vec{w}||}$. The problem defined above can then be rewritten as

$$\min ||\vec{w}|| \quad \text{s.t} \quad y_i \left(\vec{w} \cdot \vec{x_i} - b\right) \geq 1, \quad i = 1, 2, \ldots, n \tag{2.16}$$

where the classifier is determined by the $\vec{w}$ and b that solves the problem so that

$$\vec{x} \to \operatorname{sgn}\left(\vec{w} \cdot \vec{x} - b\right) \tag{2.17}$$

where the maximum margin hyperplane is determined by the training samples $\vec{x_i}$ that lie closest to it. The training samples that determine the position of the maximum margin hyperplace are called *support vectors*. The above problem formulation defines what is formally called a *hard margin* linear SVM.

The above problem formulation holds as long as the training samples of S are linearly seperable. To extend SVMs to hold for training samples that are not linearly separable Cortes and Vapnik (1995) introduced the *hinge loss function* is defined by

$$\max\left(0, 1 - y_i\left(\vec{w} \cdot \vec{x_i} - b\right)\right) \tag{2.18}$$

where the value of the function is 0 if the training sample $\vec{x_i}$ lies on the correct side of the margin. This will be the case if the constraint of Equation 2.15 is satisfied. The value of the function is determined by the distance from the margin if the training sample $\vec{x_i}$ is on the wrong side of the margin. The problem formulation with the addition of the hingle loss function becomes

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i\left(\vec{w} \cdot \vec{x_i} - b\right)\right)\right] + \lambda||\vec{w}||^2, \quad \text{s.t} \quad y_i\left(\vec{w} \cdot \vec{x_i} - b\right) \geq 1, \quad i = 1, 2, \ldots, n \tag{2.19}$$

where the arrangement between ensuring that the training samples x_i lie on the right side of the margin and a larger margin size is determined by the parameter λ. The incorporation of the hinge loss function into the original problem formulation produces what is formally called a *soft margin* linear SVM. From Equation 2.16 it can be seen that the soft margin linear SVM behaves identically to the hard margin linear SVM if the training samples $\vec{x_i}$ are linearly separable and the value of λ is small.

2.3.3 Non-linear SVM classification

In many cases the set of training samples S defined by Equation 2.12 are not linearly separable, making the problem formulation of Subsection 2.3.2 invalid. This matter was confronted in the work by Boser et al. (1992) in which the *kernel trick* is applied to maximum margin hyperplanes by replacing each dot product in the linear problem formulation by some nonlinear kernel function. The application of the kernel trick transforms the feature space of the original problem in some way specified by the kernel function so that the maximum margin hyperplane may be easier to find. By allowing the transformation specified by the kernel function to be nonlinear and the transformed feature space to be of a high dimension the classifier may be found to be a hyperplane in the transformed feature space even though it may be nonlinear in the original input space. The transformation of training samples from a input space to a transformed feature space given a polynomial kernel function is displayed in Figure 2.22

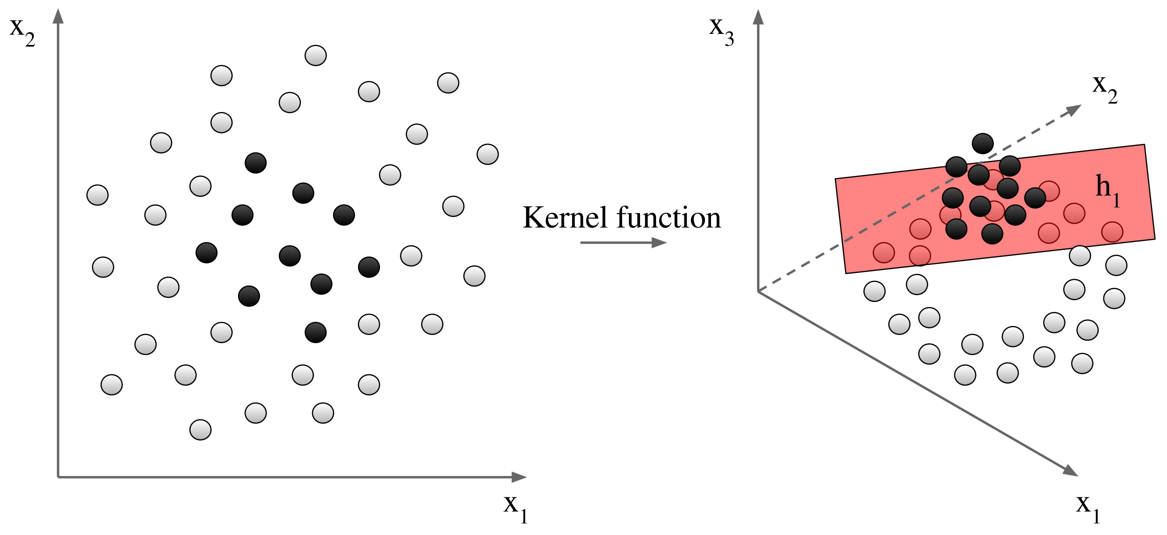


Figure 2.22: Nonlinear kernel transform example plot.

in which the hyperplane that separates the training samples x_i in the transformed feature space is given by h_1.

2.3.3.1 Kernel functions

Many different kernel functions have been applied to the problem formulations of linear SVMs. Some common kernel functions include the *polynomial kernel*, the *radial basis function kernel* and the *sigmoid kernel*.

Polynomial kernel

The polynomial kernel is a kernel function that uses a feature space over the polynomials of the original variables to represent the training samples $\vec{x_i}$ allowing nonlinear models to be learned. The polynomial kernel is defined by Equation 2.20 for polynomials of degree d.

$$K(x_i, x_j) = \left(x_i^T x_j + c\right)^d \tag{2.20}$$

Radial basis function kernel

The radial basis function kernel is a kernel function that given two training samples x_i and x_j is defined by

$$K(x_i, x_j) = e^{-\gamma ||x_i - x_j||^2}, \qquad \gamma > 0 \tag{2.21}$$

where $||x_i - x_j||^2$ is the squared Euclidean distance between the two training samples and γ is a free parameter larger than 0. Seeing as γ can be chosen freely the kernel has an infinite number of possible dimensions.

Sigmoid kernel

The sigmoid kernel is a kernal function that relies on a feature space over the hyperbolic of the original variables to represent the training samples $\vec{x_i}$ to allow the learning of nonlinear models. The sigmoid kernel is defined by Equation 2.22 where γ and r are kernel parameters.

$$K\left(x_i, x_j\right) = \tanh\left(\gamma x_i^T x_j + r\right) \tag{2.22}$$

2.3.4 Solvers

This subsection describes how the problem formulations of Subsections 2.3.2 and 2.3.3 can be solved by computing the SVM classifier. The soft margin SVM classifier will be focused on as the hard margin SVM classifier can be derived from the soft margin SVM classifier for linearly separable training samples $\vec{x_i}$ by setting a sufficiently low value for λ. The subsection first explains the classical approaches to learning linear and nonlinear soft margin SVM classification rules by reducing an expression on the form given by Equation 2.19 to a *quadratic programming problem.* Modern methods of computations such as *sub-gradient descent* and *coordinate descent* are then mentioned.

2.3.4.1 Primal form

To compute the soft margin SVM classifiers of Subsections 2.3.2 and 2.3.3 an expression on the form given by Equation 2.19 has to be minimized. This can be achieved by relying on quadratic programming algorithms to compute the solution of the expression on the form of a constrained optimization problem with an objective function that is differentiable. In practice this is conducted by introducing a variable $\zeta_i = \max\left(0, 1 - y_i\left(w \cdot \vec{x_i} + b\right)\right)$ for each $i \in \{1, 2, \ldots, n\}$ so that ζ_i is the smallest nonnegative number satisfying $\left(0, 1 - y_i\left(w \cdot \vec{x_i} + b\right)\right) \geq 1 - \zeta_i$. The optimization problem can then be written as displayed by Equation 2.23

$$\begin{gathered} \min \frac{1}{n} \sum_{i=1}^{n} \zeta_i + \lambda ||w||^2 \\ \text{s.t} \quad y_i\left(w \cdot \vec{x_i} + b\right) \geq 1 - \zeta_i \quad \text{and} \quad \zeta_i \geq 0, \qquad \forall\, i \in \{1, 2, \ldots, n\} \end{gathered} \tag{2.23}$$

which can be solved efficiently for $\vec{w}$ and b by quadratic programming algorithms as proposed by Gill and Wong (2014). Here λ is defined as in Subsection 2.3.2. This is defined formally as the optimization problem on the *primal form.* Query samples are after learning classified according to Equation 2.24.

$$\vec{x} \to \operatorname{sgn}\left(w \cdot \vec{x} - b\right) \tag{2.24}$$

The kernel trick defined in Subsection 2.3.3 can be applied to the primal form of the optimization problem specified above to learn nonlinear classification rules that correspond to linear classification rules for transformed training samples. Let some nonlinear kernel function be defined by

$$K\left(\vec{x_i}\right) = \phi\left(\vec{x_i}\right)$$

and the optimization problem is defined as by Equation 2.23. Query samples can then be classified according to Equation 2.25 after learning.

$$\vec{x} \to \operatorname{sgn}\left(w \cdot \phi\left(\vec{x}\right) - b\right) \tag{2.25}$$

2.3.4.2 Dual form

Another approach to minimizing an expression on the form given by Equation 2.19 is to solve for the Lagrangian dual of the optimization problem specified by Equation 2.23. The Lagrangian dual problem is obtained by using nonnegative Lagrange multipliers to add the constraints to the objective function to form the Langrangian as defined by Bertsekas et al. (2001). The Lagrangian dual problem is then solved for some primal variable values that minimize the Lagrangian. The solution to the Lagrangian dual problem gives the primal variables on the form of *dual variables* which are the primal variables as functions of the Lagrange multipliers. The new problem then becomes to maximize the objective function with respect to the dual variables under their derived constraints. The Lagrangian dual adaptation of the problem specified by Equation 2.23 is defined by Equation 2.26. This is defined formally as the optimization problem on the *dual form.*

$$\begin{aligned} &\max f\left(c_1, c_2, \ldots, c_n\right) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i \left(x_i \cdot x_j\right) y_j c_j \\ &\text{s.t} \quad \sum_{i=1}^{n} c_i y_i = 0 \quad \text{and} \quad 0 \leq c_i \leq \frac{1}{2n\lambda}, \qquad \forall\, i \in \{1, 2, \ldots, n\} \end{aligned} \tag{2.26}$$

This form of the linearly constrained optimization problem is seen to be a quadratic function of c_i and so can also be efficiently solved by the same quadratic programming algorithms as mentioned above. In Equation 2.26 the variables c_i are defined as

$$\vec{w} = \sum_{i=1}^{n} c_i y_i \vec{x_i}$$

so that if $\vec{x_i}$ is on the correct side of the margin according to the label y_i then $c_i = 0$. If $\vec{x_i}$ lies on the boundary of the margin then $0 \leq c_i \leq (2n\lambda)^{-1}$. Thus, it is possible to write $\vec{w}$ as a linear combination of the support vectors. By requiring that $0 \leq c_i \leq (2n\lambda)^{-1}$ so that x_i lies on the boundary of the margin the offset b can also be found. This is achieved by solving the equation $y_i\left(\vec{w}\cdot\vec{x_i}+b\right)=1 \iff b=y_i-\vec{w}\cdot\vec{x_i}$. Query samples can then be classified according to Equation 2.27.

$$\vec{x}\rightarrow \operatorname{sgn}\left(\vec{w}\cdot\vec{x}+b\right)=\operatorname{sgn}\left(\left[\sum_{i=1}^{n}c_iy_i\vec{x_i}\cdot\vec{x}\right]+b\right) \tag{2.27}$$

Learning nonlinear classification rules that correspond to linear classification rules for transformed training samples is also possible by applying the kernel trick described in Subsection 2.3.3 to the dual form of the optimization problem described above. Suppose that some kernel function is defined as

$$K\left(\vec{x_i},\vec{x_j}\right)=\phi\left(\vec{x_i}\right)\cdot\phi\left(\vec{x_j}\right)$$

and that for the transformed feature space the classification vector $\vec{w}$ is given by

$$\vec{w}=\sum_{i=1}^{n}c_iy_i\phi\left(\vec{x_i}\right)$$

where the variables c_i can be found by solving the optimization problem given by Equation 2.26 input the transformed variables $\phi\left(\vec{x_i}\right)$ and $\phi\left(\vec{x_j}\right)$. The coefficients c_i can then be found using the same procedure as above. By again requiring that $0\le c_i\le(2n\lambda)^{-1}$ such that x_i lies on the boundary of the margin in the transformed feature space it is possible to find the offset b.

$$b=\vec{w}\cdot\phi\left(\vec{x_i}\right)-y_i=\left[\sum_{k=1}^{n}c_ky_k\phi\left(\vec{x_i}\right)\cdot\phi\left(\vec{x_j}\right)\right]-y_i=\left[\sum_{k=1}^{n}c_ky_kK\left(\vec{x_k},\vec{x_i}\right)\right]-y_i$$

so that query samples $\vec{z}$ can then be classified as shown by Equation 2.28.

$$\vec{z}\rightarrow \operatorname{sgn}\left(\vec{w}\cdot\phi\left(\vec{z}\right)+b\right)=\operatorname{sgn}\left(\left[\sum_{i=1}^{n}c_iy_iK\left(\vec{x_i},\vec{z}\right)\right]+b\right) \tag{2.28}$$

2.3.4.3 Modern solvers

Contemporary algorithms for solving the optimization problems described in Subsection 2.3.4 include the methods of sub-gradient descent and coordinate descent. These approaches exhibit certain advantages over the classical approaches described in Subsection 2.3.4.

Contrary to the previously mentioned approaches to solving the optimization problems of Subsection 2.3.4 the sub-gradient descent algorithms for SVMs work directly with the expression defined by Equation 2.19. The sub-gradient descent algorithm exploits the fact that the expression defined by Equation 2.19 is a convex function to adapt the traditional gradient descent method. Instead of taking a step in the direction of the gradient of the function the sub-gradient descent algorithm takes a sted in the direction of a vector selected from the sub-gradient of the function. This allows the sub-gradient descent algorithm to be especially efficient when there are many training samples as the number of iterations of the solver will not scale with the number of training samples x_i for certain implementations. However sub-gradient descent approaches to SVMs are beyond the scope of this thesis and so will not be explored further. A thorough investigation of sub-gradient descent algorithms is provided by Boyd et al. (2003)

The coordinate descent algorithms for SVMs work with the dual form of the optimization problem described by Equation 2.26. In these approaches the coefficient c_i is iteratively adjusted in the direction of $\frac{\partial f}{\partial c_i} f$ for each $i \in \{1, 2, \ldots, n\}$. The adjusted coefficients $c_i^{'}$ are then mapped onto the nearest vector of coefficients that satisfy the constraints. A near-optimal vector can then be obtained by repeating this process. Approaches to SVMs utilizing coordinate descent algorithms are especially efficient when the dimension of the feature space is high. The coordinate descent algorithm and approaches to SVMs relying on it is also beyond the scope of this thesis. The fundamentals of the coordinate descent approach is presented by Wright (2015) and is left to the reader.

2.3.5 Multi-class SVM classification

Multi-class SVM classifiers are classifiers that rely on SVMs to assign labels to query samples where the labels are drawn from a finite set. Even though SVMs classifiers are inherently binary utilizing SVMs to perform multi-class classification is possible by reducing the single multi-class problem into multiple binary classification problems. Two approaches to reducing a multi-class problem into multiple binary classification problems are the *one-versus-all* and *one-versus-one* strategies.

For the one-versus-all strategy the classification of a query sample is determined by the classifier with the highest output value of a calibrated output function. For the one-versus-one strategy every classifier assigns a label to the query sample. The label of the query sample is then determined by a voting strategy where the label of the query sample is the label achieving the maximum number of votes.

Other approaches to reducing a multi-class problem into multiple binary classification problems include the work by Dietterich and Bakiri (1995) and more recently by Chen and Liu (2009). A comparison between different approaches to achieving multi-class classifiers based on SVMs is provided by Hsu and Lin (2002) but is left to the reader as it is beyond the scope of this thesis.

2.4 Computer vision algorithms

This section contains concepts and mathematical theory behind computer vision algorithms used in the object recognition framework implemented for this thesis. Subsection 2.4.1 describes the edge-box algorithm for generating region proposals for object recognition. The image pyramid, sliding window and non-maxima suppression techniques for locating and isolating objects by bounding boxes are then described in Subsections 2.4.3, 2.4.2 and 2.4.4 respectively.

2.4.1 Edge-boxes

An approach to generating region proposals for object recognition in digital images by relying on structured edges is presented by Zitnick and Dollár (2014). In their work the edge response for each pixel p of a digital image I is computed using the efficient structured edge detecter proposed by Dollár and Zitnick (2013) and further developed by Dollár and Zitnick (2014). Non-Maximum Suppression (NMS) orthogonal to the edge response is then performed on the computed dense edge responses to find the edge peaks. This results in a sparse edge map where each pixel p is defined as having an edge magnitude m_p and an orientation θ_p. Edges are defined as the pixels p that have an associated edge magnitude $m_p > 0.1$. Edges forming a coherent curve, line or boundary are defined as contours.

The objective of the approach proposed by Zitnick and Dollár (2014) is to identify contours that are unlikely to belong to an object contained by a bounding box by determining which contours that overlap the boundary of the bounding box. Contours that overlap the boundary of the bouding box are unlikely to belong to an object contained within the box. This is achieved for a bounding box b by computing the maximum affinity of each pixel $p \in b$ with $m_p > 0.1$ with an edge on the bounding box boundary. Edges that have a high affinity are then grouped together and the affinity between the edge groups are computed. The edge groups are created using an approach that combines eight connected edges until the sum of their orientation differences is beyond some threshold. Then the smaller edge groups are merged with other neighbouring groups. The affinity between each pair of neighbouring groups are computed for a given set of edge group $s_i \in S$. The affinity is computed for a pair of edge groups s_i and s_j based on their mean positions x_i and x_j and their mean orientations θ_i and θ_j. The affinity between a pair of edge groups is computed by

$$a(s_i, s_j) = |\cos(\theta_i - \theta_{ij}) \cos(\theta_j - \theta ij)|^\gamma \tag{2.29}$$

where the angle between the mean positions x_i and x_j of edge group s_i and s_j determines θ_{ij}, and the affinity's sensitivity to changes is orientation is adjusted by γ.

The object proposal score for a candidate bounding box b is then computed from the set of edge groups S and their associated affinities. First, the magnitudes m_p for all edges p in the edge group $s_i \in S$ is used to compute the sum m_i of the magnitudes. Then the position x_i of some random pixel $p \in s_i$ is picked. To determine whether the edge group si is wholly contained in the candidate box b or not a continuous value $w_b(s_i)$ is computed so that s_i is wholly contained in b if $w_b(s_i) = 1$ and not wholly contained in b if $w_b(s_i) = 0$. The set of edge groups s_i that overlap the boundary of the bounding box b are defined as S_b. $w_b(s_i)$ for all $s_i \in S_b$ are 0 because these edge groups are not wholly contained in b. This is also true for the mean values x_i of s_i that are not wholly contained in b. $w_b(s_i)$ is then computed for the remaining edge groups where $\bar{x}_i \in b$ and $s_i \notin S_b$ by

$$w_b(s_i) = 1 - \max_T \prod_j^{|T|-1} a(t_j, t_{j+1}) \tag{2.30}$$

where the variable T is an ordered path of edge groups that begins at some $t_1 \in S_b$ and ends at $t_{|T|} = s_i$ so that $|T|$ is the length of T. The value of $w_b(s_i)$ is defined as 0 if no such path T exists. The path T with the highest affinity between a edge group that overlaps the boundary of the candidate bounding box b and the edge group s_i is thus found by Equation 2.30. Finally, the object proposal score for a candidate bounding box b is defined as

$$h_b = \frac{\sum_i w_b(s_i) m_i}{2(b_w + b_h)^k} \tag{2.31}$$

where b_w and b_h are the width and height of the candidate bounding box and the value k is used to offset the bias of larger images having more edges on average.

To speed up the computation of $w_b(s_i)$ in Equation 2.30 the integral image trick presented by Crow (1984) can be applied. Here, the integral image is used to compute the sum of all m_i for $\bar{x_i} \in b$. The expression $(1 - w_b(s_i))$ is then sumstracted from the sum for all s_i with $\bar{x_i} \in b$ and $w_b(s_i) < 1$. Another observation by Zitnick and Dollár (2014) is that the edges in the center of a candidate bounding box b are less important than the edges near the boundary of the bounding box. This fact can be accounted for by rewriting Equation 2.31 so that the edge magnitudes of a smaller box b^{in} is subtracted from the candidate bonding box b as shown by Equation 2.32

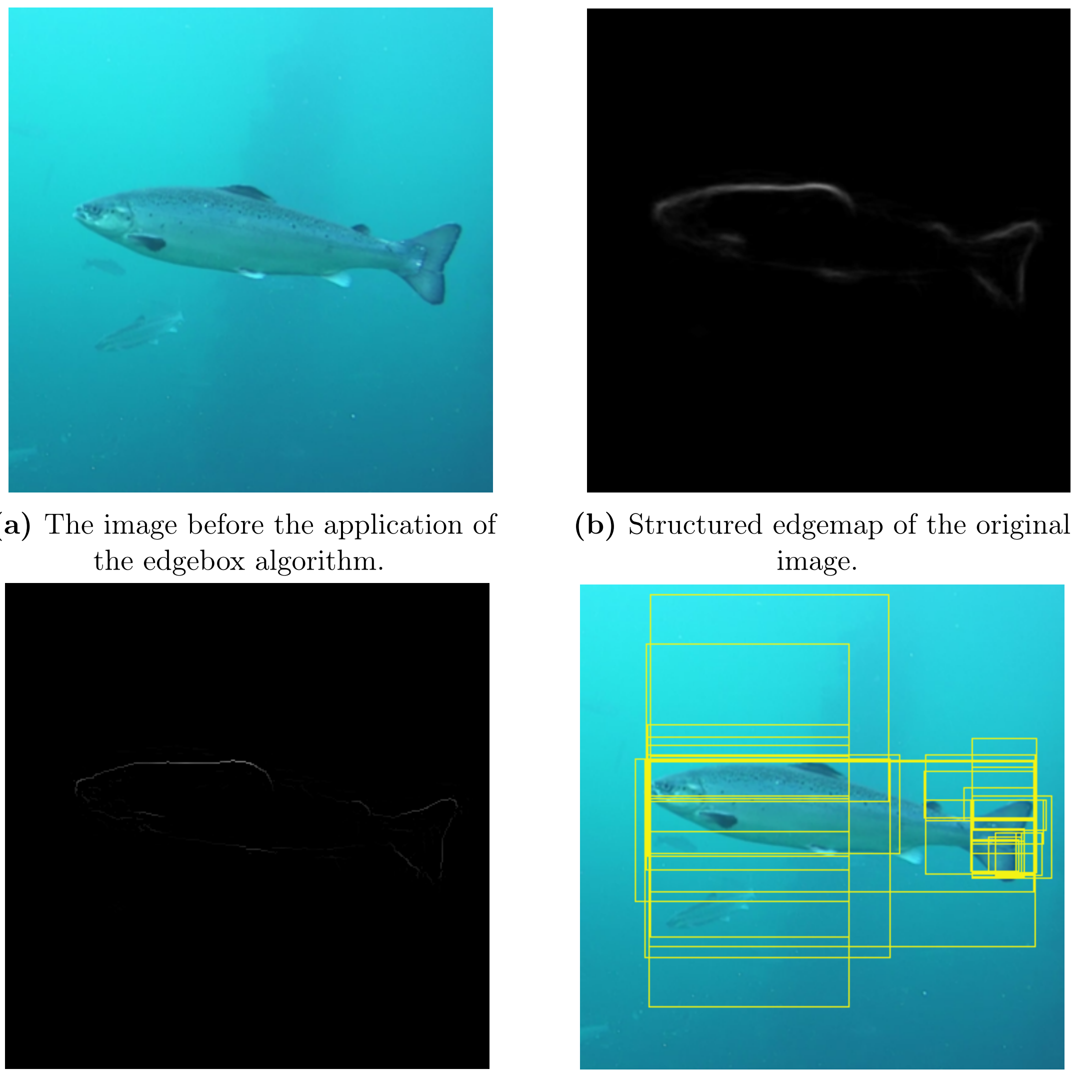
$$h_b^{in} = h_b - \frac{\sum_{p \in b^{in}} m_p}{2(b_w + b_h)^k} \tag{2.32}$$

where $\frac{b_w}{2}$ and $\frac{b_h}{2}$ specifies the width and height of b_{in}. As with the expression in Equation 2.31 the integral image trick can be used to compute the sum of the edge magnitudes in b_{in} efficiently.

In the above reasoning it is assumed that the set of edge groups S_b that overlap the boundary of the candidate bounding box b is known. Because the number of candidate boxes to evaluate is large finding an effective way to obtain S_b is crucial. In the work by Zitnick and Dollár (2014) an efficient method for finding the edge groups that overlap the boundary of the candidate bounding box b is proposed. The proposed method relies on two additional datastructures for finding intersecting edge groups for each side of a candidate bounding box. First, two datastructures are created for each row of the input image. The first datastructure stores the edge group indices of row r in an ordered list L_r. The list L_r is created by storing the order in which the edge groups occur along the row r. If the index of the edge group changes from one pixel to the next along the row r the index is added to the list L_r. This results in the list L_r being much shorter than the width of the input image. A 0 is added to the list in the case that some pixels between the edge groups are not edges. The second datastructure, K_r, has the same size as the width of the input image. K_r stores the index into the list L_r for each column c in row r.

If $L_r\left(K_r\left(c\right)\right) = i$ then the pixel with coordinates $\left(c, r\right)$ is a member of the edge group s_i. Because most of the pixels of the input image does not belong to any edge group the list of overlapping edge groups can be efficiently found by searching the list L_r from $K_r\left(c_0\right)$ to $K_r\left(c_1\right)$.

Searching the input image for possible candidate bounding boxes can be performed by passing a sliding window over the image at different scales. The sliding window technique is further discussed in Subsection 2.4.2 and so will not be explained further in this Subsection. If a bounding box is detected using the sliding window technique with a score h_b^{in} above some small threshold the bounding box is retained for further refinement. This refinement of the detected bounding boxes is achieved by performing a greedy iterative search over position, scale and aspect ratio so as to maximize h_b^{in}. The search step is reduced by halv the step for each iteration until the search is halted when the translational step size is less than 2 pixels. Images showing the stages of the edge-box approach required to find region proposals for object recognition are shown in Figure 2.23.



(a) The image before the application of the edgebox algorithm.

(b) Structured edgemap of the original image.

(c) The edgemap after non-maxima suppression.

(d) Region proposals generated by the edge-box algorithm.

Figure 2.23: Stages of generating region proposals using the edge-box algorithm

2.4.2 Sliding window technique

The sliding window technique within computer vision and machine learning is a technique that allows the localization of objects in digital images by relying on a detector/classifier window that is allowed to slide over the query image in a row-column manner. For each step of some length specified by the stride variable Δ the slide the detector/classifier attempts to detect/classify the image patch covered by the detector/classifier window. When a correct detection/classificaion is made and an object is detected/classified the area covered by the detector/classifier window is often shown by a bounding box around the detected/classified. The general behaviour of sliding windows for use within computer vision and machine learning is depicted in Figure 2.24 in which the white cells of the grid are the cells that have already been visited by the sliding detector/classifier window and the red cell is the current position of the sliding window.

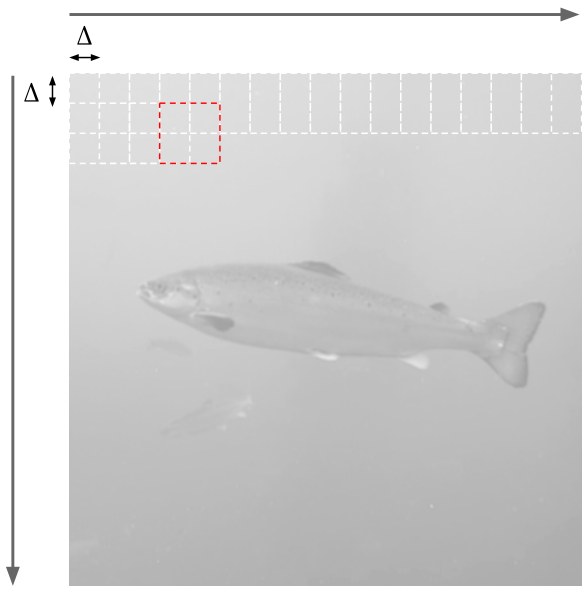


Figure 2.24: Sliding window technique example plot.

2.4.3 Image pyramid technique

Within computer vision, the image pyramid technique, is a technique used to observe an image in image pyramid representation. The image pyramid representation of an image is a collection of multiple images all derived from the same original image by smoothing the image and then downsampling it according to some factor less than 1. This process is repeated until some stopping criteria is reached resulting in a pyramid of succesively smaller images. The image smoothing performed before each downsampling of the original image is executed by convolving the image with some kernel. Some common kernels used to achieve the image pyramid representation include the Gaussian kernel and the Laplacian kernel. The Gaussian kernel for a two-dimensional space, i.e a digital image, as defined by Shapiro and Stockman (2001) is shown in Equation 2.33. The Gaussian kernel is commonly used in the downsampling of images because of the blurring effect it has on images.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.33}$$

The Laplacian kernel as defined by Paris et al. (2011) is commonly used in image compression algorithms as it is able to save the difference image of the blurred versions between each pyramid level. This enables the reconstruction of high resolution images from little data.

Within machine learning the image pyramid technique is used to achieve scale-invariance for object detection and recognition frameworks. This is accomplished by letting the detector/classifier window slide over the images at each level of the image pyramid as described in Subsection 2.4.2. The image pyramid technique relies on the fact tha object might appear too small or large for the framework at some level of the image pyramid, but could be detected/classified more easliy at another level. This behaviour is displayed by Figure 2.25 in which each level of the constructed image pyramid is a quarter of the size of the previous level. The stopping criteria while constructing the image pyramid is that the top level of the image pyramid should not be smaller than the detector/classifier window. The detector/classifier window passing over each level of the image pyramid as defined in Subsection 2.4.2 is shown by the red square.

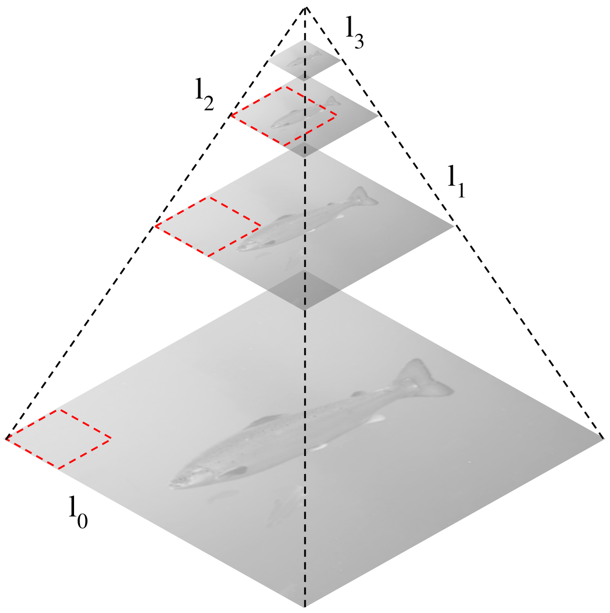


Figure 2.25: Image pyramid technique example plot.

2.4.4 Non-maximum suppression

The Non-Maximum Suppression (NMS) algorithm is a post-processing algorithm that within object recognition is used to merge detections/classifications that belong to the same object in a digital image. Multiple detections of the same object in one image is often an undesirable side effect of performing correct detection/classification that stems from the imperfect ability of detection/classification algorithm to localize the concepts of interest. The NMS algorithm works by weighing the bounding boxes enclosing a detected/classified object according to some measure so as to find the bounding box that is the most correct among the many. The objective of the NMS algorithm can thus be defined as to retain only one bounding box per group of bounding boxes corresponding to the precise location of the true object location. Many different adaptations of the NMS algorithm that utilizes different measures to optain the desired bounding box exists. Most of these approaches employ confidence measures supplied by many contemporary detection/classification frameworks to weight each bounding box in a cluster by how confident the detector/classifier is in the detection/classification. However, confidence measures are not available for all detectors/classifiers. Another approach to the NMS algorithm that does not rely on any confidence measure of the detector/classifier is weighing the bounding boxes by how much they overlap with one another. By requiring that all bounding boxes that overlap by more than a certain percentage of their own area are merged an approximation of the true object location can be achieved. This amounts to determining the expression displayed in Equation 2.34 and merging the bounding boxes that overlap by more than a certain fraction.

$$\text{Overlap} = \frac{A_{b_i \cap b_j}}{A_{b_i} \cup A_{b_j} - A_{b_i \cap b_j}} \tag{2.34}$$

The above mentioned behaviour is displaued in Figure 2.26 in which the threshold on the overlap is chosen to be 0.5.

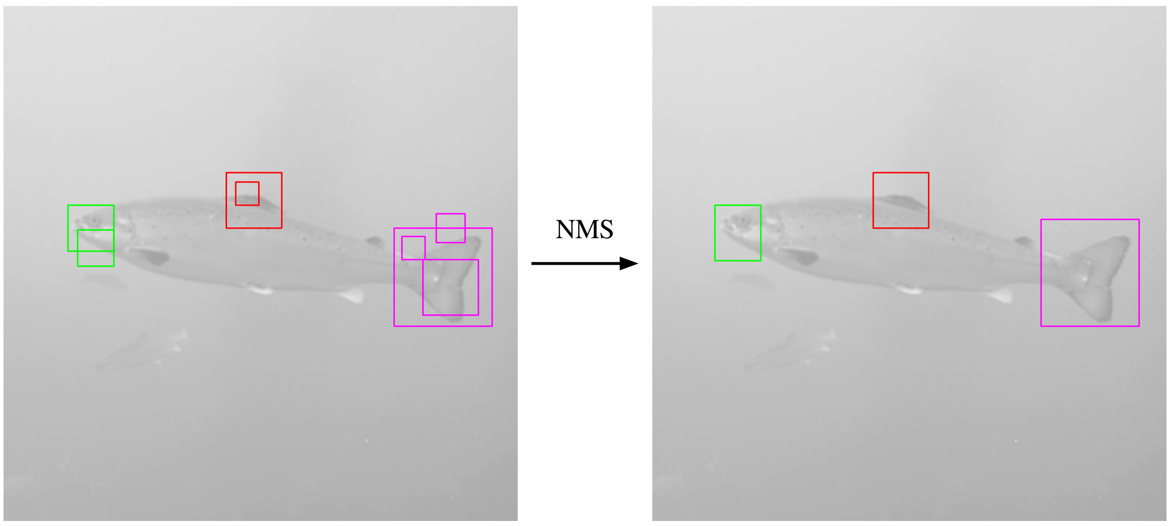


Figure 2.26: Non-maxima suppression example plot.

Chapter 3

Implementation

This chapter describes the object recognition framework implemented for this thesis and explains the flow of information for this framework. The hardware used for implementing the system is described in Section 3.1, while the programming language, operating system, development environment. software and programming libraries chosen for the work conducted for this thesis is presented in Section 3.2. The implementation itself is then presented in Section 3.3 along with an explanation of the information flow of the system.

3.1 Hardware

This section describes the hardware utilized in constructing the object recognition framework implmented for this thesis. The Sony FBC-EV7520 camera system is described in Subsection 3.1.1, while the underwater housing in the form of the Aquapod provided by Sealab AS is described in Subsection 3.1.2.

3.1.1 Sony FCB-EV7520

The FCB-EV7520 camera system produced by Sony is the camera system that was chosen for use in the object recognition framework presented in this thesis. The FCB-EV7520 is a internet protocol camera that is equipped with a 30x optical zoom lens and that is capable of recording at 1920x1080 resolution at up to 50 frames per second. The small form factor of the block camera and the high sensitivity makes the FCB-EV7520 well suited for UAV and drone use and is one of the main reasons that it was chosen for this project. The camera is also well suited for remote operation as it is capable of communication over ethernet by the internet protocol.

3.1.2 Sealab Aquapod

The Aquapod created by Sealab AS serves as the underwater housing for the Sony FCB-EV7520 camera system used in the object recognition framework implemented for this thesis. However, it also serves as one of the only physical components of the object recognition framework. The Aquapod is supplied with camera housing with dome ports for two camera systems and four external LED lights that provide ample illumination under water. It also features a water jet propulsion system capable of turning the Aquapod in the $\vec{z}$ plane for better viewing angles. The Aquapod operates on a single cable principle so that all communication and power to the electronics goes through a single umbilical. The communication from the Aquapod to a computer on the surface is handled by single mode fiberoptics so that image delays over the cable becomes a non-issue. Additionally, the Aquapod features multiple moisture and pressure sensors in both the housing for the camera system and for other electronics so as to quickly detect possible leaks while the system is submerged. The Aquapod is displayed in Figure 3.1.

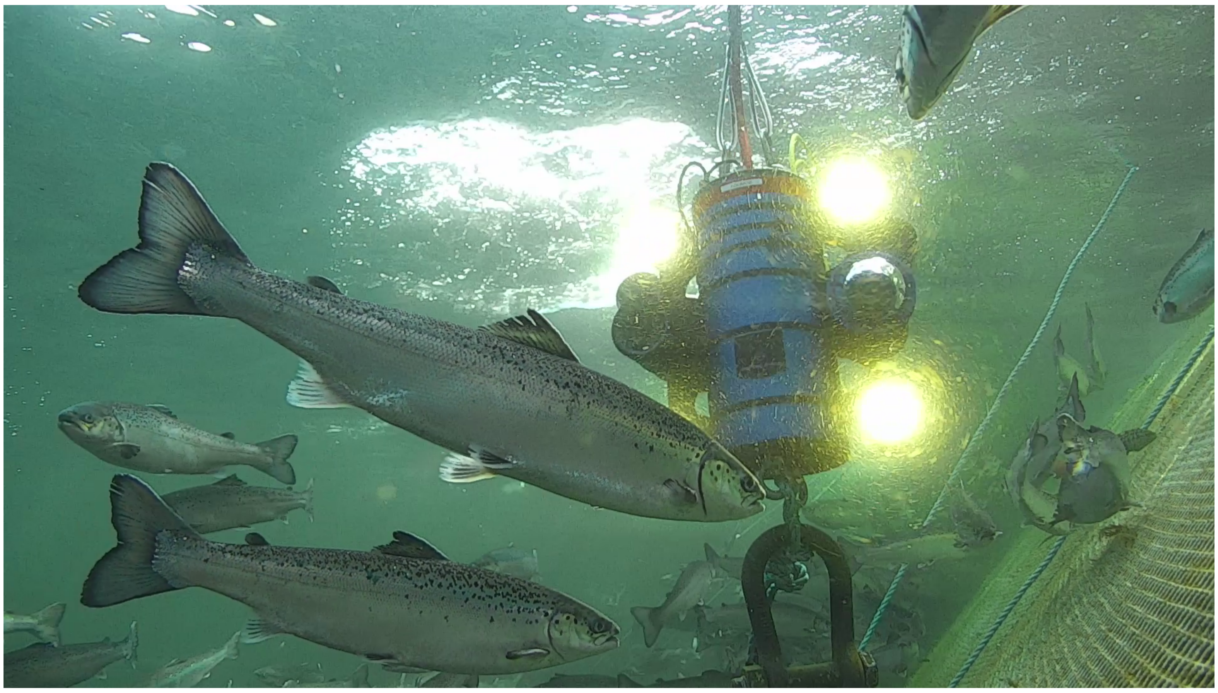


Figure 3.1: The Aquapod by Sealab AS at the Kåholmen test facility on Hitra.

3.2 Software

This section provides information on the software packages and programming libraries that the object recognition framework implemented for this thesis utilizes. Subsections 3.2.1, 3.2.2 and 3.2.3 justify the choice of programming language, computer operating system and development respectively. The OpenCV library is then described in Subsection 3.2.4 and an explanation of how it is used by the system is provided. Subsection 3.2.5 discribes the FFTW library and explains how it is utilized by the other programming libraries employed by the object recognition framework. The programming library featuring the

LBP operators utilized for classification is described in Subsection 3.2.6. Subsections 3.2.7 and 3.2.8 describe the software packages LIBSVM and LIBLINEAR that are utilized for non-linear and linear classification respectfully. The software package OpenMP is then specified in Subsection 3.2.9 it how LIBSVM and LIBLINEAR utilize it is covered. The software library LibVLC that is utilized for communicating with the Sony FCB-EV7520 camera system is finally described in Subsection 3.2.10.

3.2.1 Programming language

The programming language chosen for the work resulting in this thesis was C++. C++ was chosen as the programming language for this work because it is a general purpose programming language that incorporates both object-oriented and generic programming features, offers a bias towards performance and efficiency, whilst also boasting a low memory footprint. Many software packages are written in C++ or have C++ versions. This includes many of the software packages that are essential to the object recognition framework presented in this thesis such as OpenCV, LIBSVM, LIBLINEAR and LibVLC.

3.2.2 Operating system

The operating system chosen as the platform for the object recognition system implemented for this thesis was Ubuntu 16.04. Ubuntu is a open source Debian based Linux operating system that is based on free software packages. Ubuntu is completely modifiable and can offer many of the freedoms that a traditional Windows operating system is not capable of. Ubuntu, as a Linux based operating system, is also focused on secury which is of a large concern when operating over the internet as with the Sony FCB-EV7520 internet protocol camera system. Ubuntu is also compatible with most software packages and installation of these is mostly without any trouble. Ubuntu 16.04 also boasts a large helpfull community that is able most questions.

3.2.3 Integrated development environment

The development environment chosen for the programming tasks required for implementing the object recognition framework was the Ubuntu terminal with the Vim environment installed. The reason for this lightweight choice of integrated development enviroment is that when workin on a Linux based system the terminal is already the environment where you do most of the work. Extending the terminal to also work as the programming environment for the work conducted for this thesis came naturally.

3.2.4 Open computer vision library

The Open Computer Vision Library (OpenCV) is a open source software library that provides code and algorithms dedicated to the fields of machine learning and computer vision. OpenCV is written natively in C++ but includes interfaces for many different programming languages. OpenCV also supports a multitude of operating systems including Windows, Mac OS, Android and Linux. The Open Computer Vision Library is used as the foundation to the object recognition framework presented in this thesis because of how compatible it is to other necessary software packages, and because of the large number of optimized computer vision algorithms it provides. The OpenCV library is also free to use under the BSD license. The OpenCV contribution package and the OpenCV extra data package was also used for the object recognition framework implemented for this thesis. The OpenCV extra data package contains extra data models and data samples not otherwise included in OpenCV. The OpenCV contribution package contains implementations of computer vision algorithms that the OpenCV developers can not guarantee the stability of, but implementations that can be usefull none the less. The OpenChttps://www.scribbr.com/dissertation/preface-dissertation/V contribution package and the OpenCV extra data package are utilized in the object recognition framework implemented for this thesis by providing an implementation of the edge-box algorithm presented by Zitnick and Dollár (2014).

3.2.5 FFTW

The Fast Fourier Transform in the West (FFTW) is a subroutine library by Frigo (1999) written in C that provides a fast implementation for computing the discrete Fourier transform. FFTW is used by the software library provided by Vatani (2013) that supplies the C++ implementations of the LBP operators used for the classification task in the object recognition framework implemented for this thesis. Since FFTW 1.3 it also free to use.

3.2.6 Nourani LBP

The LBP library provided by Vatani (2013) is a GitHub repository that provides implementations of the popular adaptations of the LBP operator written in C++ that are utilized for the classification task in the object recognition framework implemented for this thesis. The adaptations of the LBP operators supplied by this repository are here used because they are written in C++ and are under no licensing demands.

3.2.7 LIBSVM

LIBSVM is a open source machine learning library that delivers a C++ implementation for solving many linear and non-linear SVM problems. LIBSVM is utilized by the object recognition framework implemented for this thesis by supplying a SVM implementation that utilizes the Radial Basis Function kernel to solve the constrained optimization problem. The implementation also supplies an algorithm for the automatic scaling of training data and an algorithm providing cross-validation in the form of a grid search on the parameters of the constrained optimization problem. LIVBSVM is also free to use under the BSD license.

3.2.8 LIBLINEAR

LIBLINEAR is an open source machine learning library created by the same people as LIBSVM. It delivers a fast C++ implementation for solving linear SVMs and used in the object recognition framework implemented for this thesis for this task. LIBLINEAR, in opposition to LIBSVM, does not an provide algorithm for the automatic scaling of trianing data nor any implementation cross-validation. LIBLINEAR, as LIBSVM, is free to use under the BSD license.

3.2.9 OpenMP

Open Multi-Processing is an application programming interface that provides multi-platfor mshared memory multiprocessing in the programming languages C, C++ and Fortran. OpenMP is utilized by both LIBSVM and LIBLINEAR by making some parts of the training of the SVM multi-threaed. In this OpenMP helps speed up the overall process of implementing the object recognition framework described in this thesis.

3.2.10 LibVLC

LibVLC is a media framework written in C++ that can be embedded into an application to achieve multimedia capabilities. LibVLC is for communicating with the Sony FCB-EV7520 internet protocol camera used in the object recognition framework implemented for this thesis. LibVLC is able to communicate of the real time streaming protocol (RTSP) which the FCB-EV7520 understands. In this way it is possible to build an entire application around a pre-made camera such as the FCB-EV7520. LibVLC is licensed under the LGPL license.

3.3 SVM implementation

This section provides a description of the object recognition framework implemented for this thesis. The section relies on the definitions of concepts and mathematical theory presented in 2 along with the description of hardware and software provided by the above sections to explain how the system works and how the data flows in the system. The section consists of two main parts; the acquisition of image material from the Sony FCB-EV7520 camera system and the classification of fish parts in these images. An overview of the system can be seen in Figure 3.2.

3.3.1 Image material acquisition

The image material that is to classified is acquired from the Sony FCB-EV7520 camera system by communication over the real time streaming protocol (RTSP). A complete explanation of how the RTSP works is beyond the scope of this thesis. However, it is

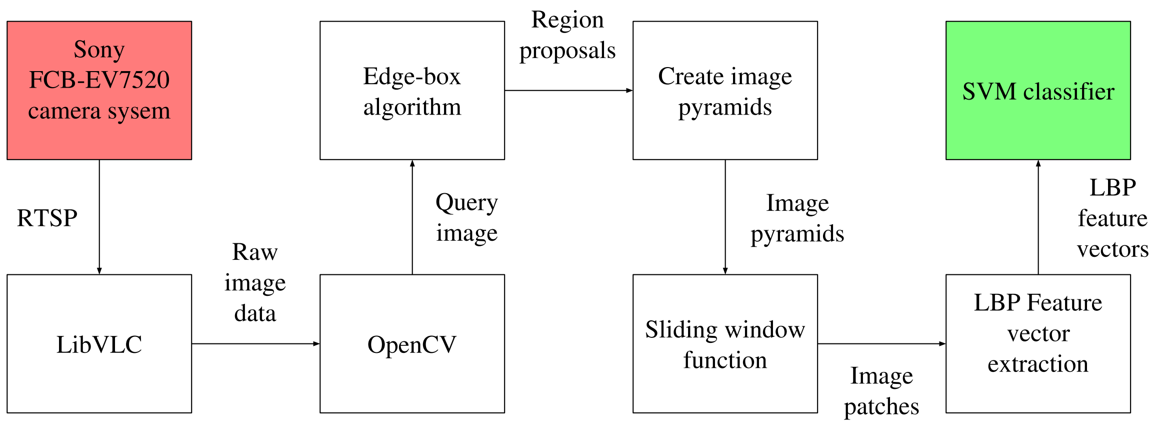


Figure 3.2: Overview of the object recognition framework.

sufficient to define it as a communications protocol that works over ethernet by utilizing the internet protocol. The retrieval of new image data is defined programatically so that LibSVM is used to continually update a data container. The raw image data ia retrieved from that data container by utilizing a mutex so that the OpenCV side of the code locks the mutex whenever it is ready to classify objects in a new query image but unlocks it while it is workin. By implementing the mutex the Sony FCB-EV7520 camera is not forced to wait on the object recognition algorithm before it can update the data container. This makes sure that the system runs in real-time even if the object recognition framework is slower than the camera. The image converted from raw image data to an OpenCV `Mat()` object is then sent to the function of the code implementing the edgebox alorithm defined in 2.4.1.

3.3.2 Classification

Given some query image the edge-box algorithm defined in 2.4.1 outputs regions proposals from the query image. These region proposals are then used to create image pyramids subject to the level constraint that is the size of the sliding window as defined in 2.4.3. The recognition window is next slid across each level of the pyramids created from the region proposals as defined by 2.4.2. For each step determined by the value Δ that the recognition window uses to slide over the scales of the query image a LBP feature vector corresponding to the LBP type of the SVM classifier is extracted. The extracted feature vectors can then be classified by the SVM classifier by determining on which side of the learned decision surface it lies.

Chapter 4

Experiments

This chapter describes the experiments that were conducted for the object recognition framework implemented for this thesis. The acquisition and preparation of image material for training and testing the SVM classifiers along with the data formats used is described in Setion 4.1. Section 4.2 then describes the training of the SVM classifiers. The performance metrics used to evaluate the SVM classifiers along with a discription of the confusion matrix for multi-class classifiers are then presented in 4.3.

4.1 Image material

This section describes the image material used for training and testing of the SVM classifier based on the different LBP image patch descriptors. To test the SVM classifiers based on the adaptations of the LBP image patche descriptor the classes that we want to classify needs to be defined. For the experiments conducted for this thesis the classes that the SVM classifiers should classify are defined as background, fish heads, dorsal fins and coidal fins.

4.1.1 Data acquisition

The image material used for the training and testing of the SVM classifiers based on the LBP image patch descriptors was acquired by recording video footage using the Sony FCB-EV7520 camera system situated inside the Aquapod provided by Sealab AS at a fish farming test facility located at Kåholmen on Hitra, Norway. As no image material was available beforehand the positive image material was acquired over the course of the semester when this thesis was written. The video footage acquired from the Sony SVMs is require to be labeled in order for a SVM classifier to learn, and so the acquired image data was first separated into individual images and then annotated to suit the needs of the SVMs. The annotation of positive samples was done semi-manually by the use of the `opencv_annotation()` tool provided by OpenCV. `opencv_annotation()` is a drag and drop tool that saves the areas marked by the mouse pointer of a computer to a text file of annotations. This text file can then be used to extract the areas bounded by the annotations from the original digital images. In total 5000 positive samples for each class were annotated to train the SVMs. Separate annotatios for the test set were also made from novel image material. However, only 500 test samples from each class were annotated. For the background classes of the training and test set random images were chosen that do not include the sought fish parts. The image material is provided as 35x35 pixel image patches that either contain the sought fish parts or not.

4.1.2 Pre-processing

Because the feature vectors based on the adaptations of the LBP described in 2 are extracted from grayscale images both the image material used for training and the image material used for testing was converted to grayscale separably. Samples of the image material used for training and testing the SVM classifiers are shown in Figures 4.1 and 4.2 respecitvely.

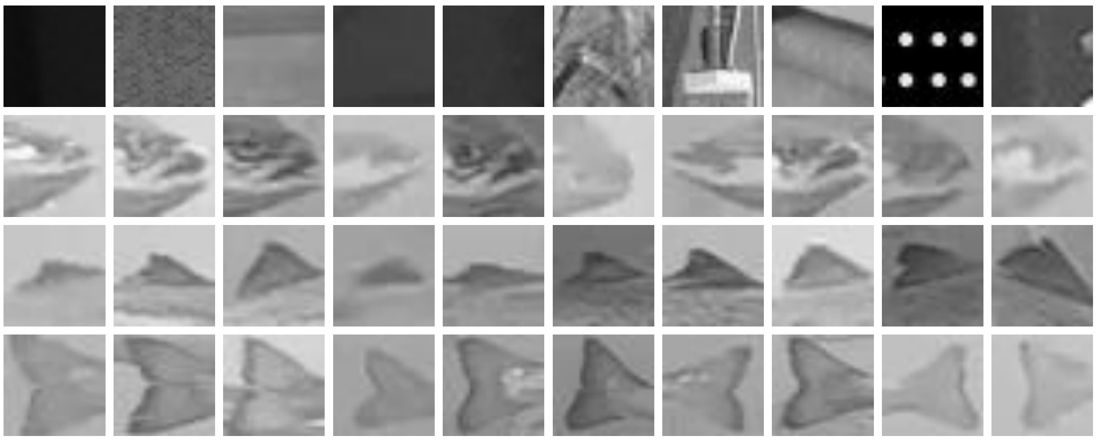


Figure 4.1: Training samples for each class.

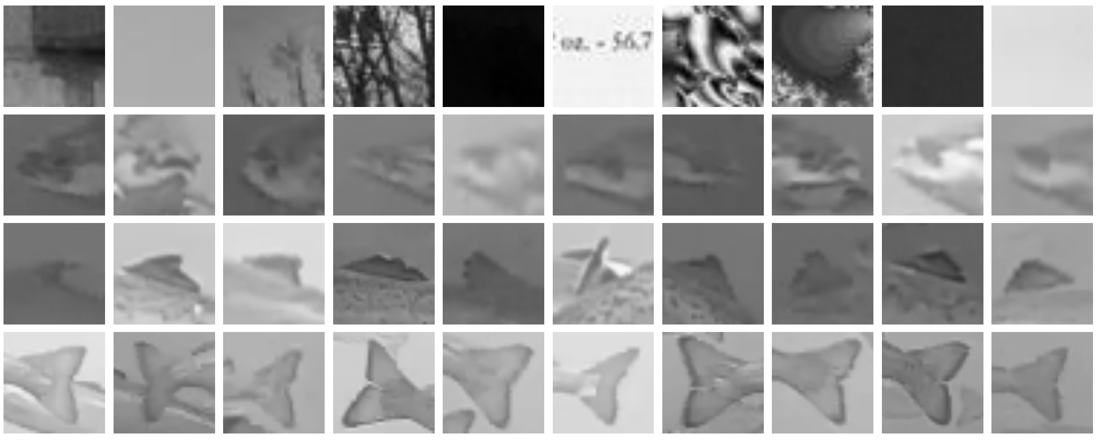


Figure 4.2: Test samples for each class.

4.1.3 Data formats

In order to learn the SVMs defined in Chapter 2 require that the image data used for training is on the form of a vector of data points where the vector is labeled as one of the classes that is to be classified, in this case the fish parts defined in 4.1. By extracting the LBP feature vectors from the training samples as defined in Section 2.2 this is achieved The feature vectors, consisting of the histogram of the distribution of LBP codes over the image patches, are then normalized and scaled between -1 and 1 according to the advice given by Chang and Lin (2011) and Fan et al. (2008) so as to maximize the performance of the trained classifier. The feature vectors of the image patch samples that are to be used for training are then added to a text file where each line signifies the feature vector of one image patch used for training.

4.2 SVM classifier training

This section describes the training of the linear and non-linear SVM classifiers utilized in the object recognition framework implemented for this thesis.

The SVM classifiers trained for the object recognition framework implemented for this thesis were based on the LBP codes defined in 2.2 and the different SVM classifier implementations defined in 2.3. For this thesis linear and non-linear SVM classifiers based on LBP, LBP-U2, LBP-RI, LBP-RIU2 and LBP-HF codes were trained. The non-linear SVM classifiers were achieved using the Radial Basis Function (RBF) kernel as defined by Equation 2.21. The classifiers were trained using a variable number of training images for each object class resulting in a total of 120 classifiers.

4.2.1 LBP parameter choices

The parameters of the different LBP codes were kept constant for the training of the classifiers to achieve consistency during testing. The radius r, number of neighbouring pixels P and block size b as defined in 2.2 were kept at 2, 8 and 5 respectfully. These values were chosen according to the parameters used in achieving the highest performing classifiers of object recognition frameworks utilizing LBP codes in non-underwater environments.

4.2.2 SVM parameter choices

The parameters of the linear SVM and the non-linear SVM utilizing the RBF kernel are defined in 2.3. For the non-linear SVM using the RBF kernel this amounts to the task of cross-validation by performing a grid search over the space spanned by the parameters C and γ that are to be found. The grid search is performed k-fold with exponentially growing parameters C and γ until suitable values for the parameters are found. For the linear SVM, which only relies in the parameter C it is possible to set the variable γ as constant and perform a line search with exponentially growing parameters instead of performing the grid search.

4.3 Performance measurements

This section details the performance measurements used for testing the SVM classifiers trained as defined in the above sections. The performance measures used for testing the SVM classifiers are defined in Subsections 4.1, 4.4, 4.2, 4.3 and 4.5. This section also includes a description of the confusion matrix, a powerfull tool in evaluating the performance of a multi-class classifier.

4.3.1 Confusion matrix for binary classifiers

The condusion matrix for a binary classifier is defined as by Table 4.1. Here the true positives (TP) and true negatives (TN) are given by the upper left and lower right fields of the table specified by $c_{0,0}$ and $c_{1,1}$ respectively. The false positives (FP) are given by

Table 4.1: Confusion matrix of a binary classifier.

		Predicted	
		Class 0	Class 1
Actual	Class 0	$c_{0,0}$	$c_{0,1}$
	Class 1	$c_{1,0}$	$c_{1,1}$

4.3.2 Confusion matrix of multi-class classifiers

The confusion matrix of a multi-class classifieris defined as by Table 4.2 in which the total number of test examples for any class is the sum of the corresponding row for that class. The total number of false negatives (FN) for a given class is the sum of values in the corresponding row, expluding the true positives (TP). The total number of false positives (FP) for a class is the sum of value in the corresponding column, excluding the true positives (TP). Finally, the total number of true negatives (TN) for a certain class is the sum of all rows and columns excluding the row and column of that class. The true positives (TP) are given by the diagonal elements of Table 4.2

4.3.3 Performance metrics for multi-class classifiers

The performance metrics used to test the multi-class classifiers implemented for this thesis are given by Sokolova and Lapalme (2009) and are defined below. In their work the performance metrics used to evaluate multi-class classifiers are divided into two classes, performance metrics that evaluate the classifiers based on macro-averaging over the classes and performance metrics that evaluate the classifiers based on micro-averaging over the classes. Macro-averaging performance petrics calculate the same metrics for all classes and then average the score. Micro-averaging performance metrics calculate the sum of

Table 4.2: Confusion matrix of a multi-class classifier.

		Predicted				
		Class 0	Class 1	$\cdots$	$\cdots$	Class N
	Class 0	$c_{0,0}$	$c_{0,1}$	$\cdots$	$\cdots$	$c_{0,n}$
	Class 1	$c_{1,0}$	$c_{1,1}$			$c_{1,n}$
Actual	$\vdots$	$\vdots$		$\ddots$		$\vdots$
	$\vdots$	$\vdots$			$\ddots$	$c_{n,n-1}$
	Class N	$c_{n,0}$	$c_{n,1}$	$\cdots$	$c_{n-1,n}$	$c_{n,n}$

counts in order to obtain cumulative TP, FN, TN and FP which is then used to calculate a performance measure. The two techniques are different in that micro-averaging will favor larger classes during classification whereas macro-averaging will treat all classes equal. Because the training and test sets used for the training and testing of the classifiers implemented for this thesis were equal in size for all classes the performance metrics used for this thesis have been the ones based on the macro-averaging technique. The macro-averaging performance metrics are marked with a subscript M.

4.3.3.1 Average accuracy

The average accuracy of a multi-class classifier is defined as the average per-class effectiveness of the classifier. This is defined mathematically as shown by Equation 4.1

$$\text{Average accuracy} = \frac{\sum_{i=1}^{N} \frac{\mathrm{TP_i+TN_i}}{\mathrm{TP_i+FN_i+FP_i+TN_i}}}{N} \tag{4.1}$$

in which N is the number of classes of the classifier, TP_i are the true positives of class i, TN_i are the true negatives of class i, FN_i are the false negatives of class i and FP_i are the false positives of class i.

4.3.3.2 Precision$_\mathrm{M}$

The precision of a multi-class classifier is defined as the average per-class agreement of the data class labels of a classifier. This relationship is defined as shown by Equation 4.2

$$\mathrm{Precision_M} = \frac{\sum_{i=1}^{N} \frac{\mathrm{TP_i}}{\mathrm{TP_i+FP_i}}}{N} \tag{4.2}$$

in which N is the number of classes of the classifier, TP_i are the true positives of class and FP_i are the false positives of class i.

4.3.3.3 Recall$_\mathrm{M}$

The recall of a multi-class classifier is defined as how effective the classifier is at identifying class labels for each class of the classifier. This relationship is defined as shown by Equation 4.3

$$\mathrm{Recall_M} = \frac{\sum_{i=1}^{N} \frac{\mathrm{TP_i}}{\mathrm{TP_i}+\mathrm{FN_i}}}{N} \tag{4.3}$$

in which N is the number of classes of the classifier, TP_i are the true positives of class and FN_i are the false positives of class i.

4.3.3.4 Error rate

The error rate of a multi-class classifier is defined as the average per-class classification error of the classifier. This is defined mathematically by Equation 4.4

$$\text{Error rate} = \frac{\sum_{i=1}^{N} \frac{\mathrm{FP_i}+\mathrm{FN_i}}{\mathrm{TP_i}+\mathrm{FN_i}+\mathrm{FP_i}+\mathrm{TN_i}}}{N} \tag{4.4}$$

in which N is the number of classes of the classifier, TP_i are the true positives of class i, TN_i are the true negatives of class i, FN_i are the false negatives of class i and FP_i are the false positives of class i. The error rate of a multi-class classifier is the inverse of the average accuracy of that classifier.

4.3.3.5 F-score$_\mathrm{M}$

The F-scores of a multi-class classifier are the relationships between the positive labels of the data and those supplied by the classifier ased on a per-class average. This relationship is defined as shown by Equation 4.5

$$\text{F-score}_\mathrm{M} = \frac{(\beta^2+1)\cdot \mathrm{Precision_M} \cdot \mathrm{Recall_M}}{\beta^2 \cdot \mathrm{Precision_M} + \mathrm{Recall_M}} \tag{4.5}$$

in which β is a free variable, the precision of the classifier is given by 4.2 and the recall of the classifier is given by 4.3. By setting the value of β to 1 the harmonic mean between the precision and the recall can be measured.

Chapter 5

Results

This chapter presents the results of the performance tests conducted for each of the classifiers trained for this thesis. The results of the measured average accuracy, precision, recall, error rate and F1-measure of the SVM classifiers is included in the following sections. Section 5.1 also includes the measured cross-validation average accuracies for the SVM classifiers. Section 5.7 shows the training duration for all classifiers. A discussion of the results presented in this chapter can be found in Chapter 6.

5.1 Cross-validation average accuracy

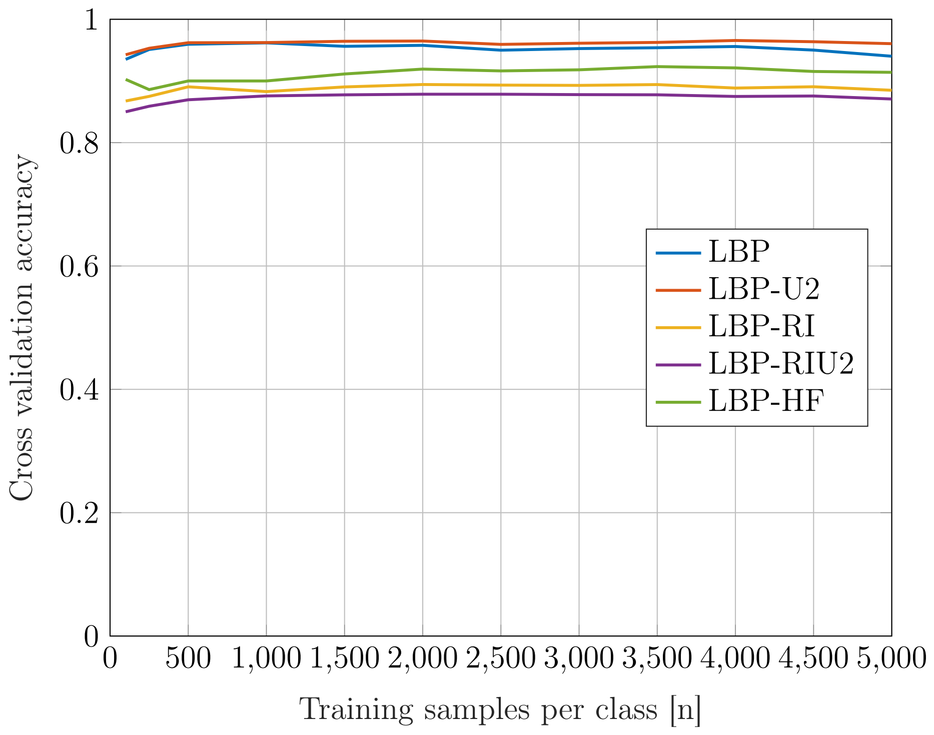


Figure 5.1: Cross-validation average accuracy of linear SVM classifiers.

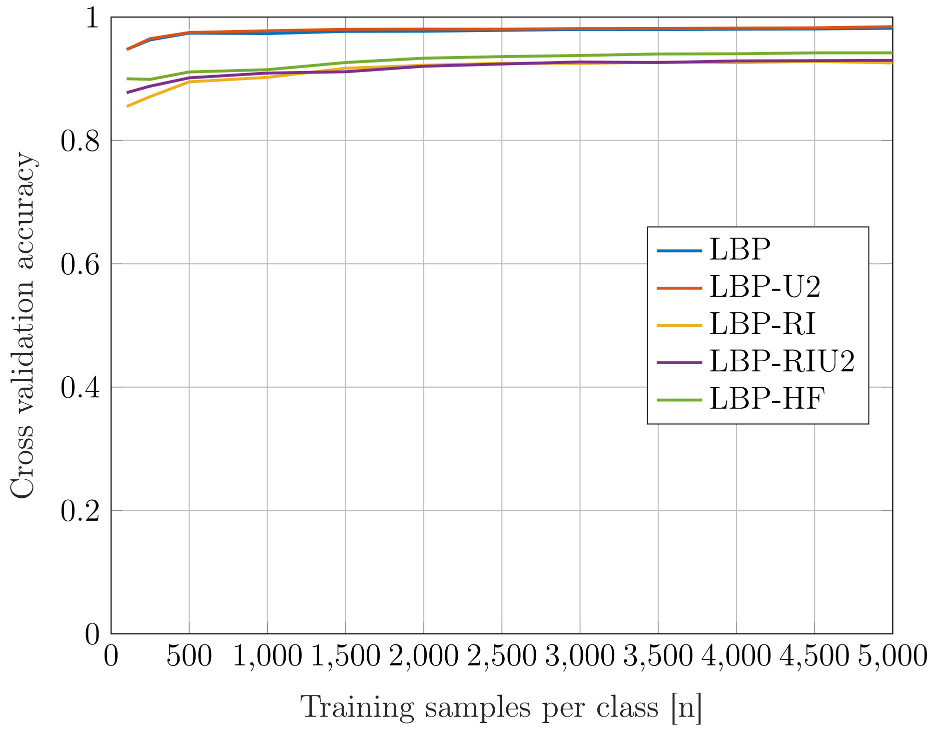


Figure 5.2: Cross-validation average accuracy of radial basis function SVM classifiers.

5.2 Average accuracy

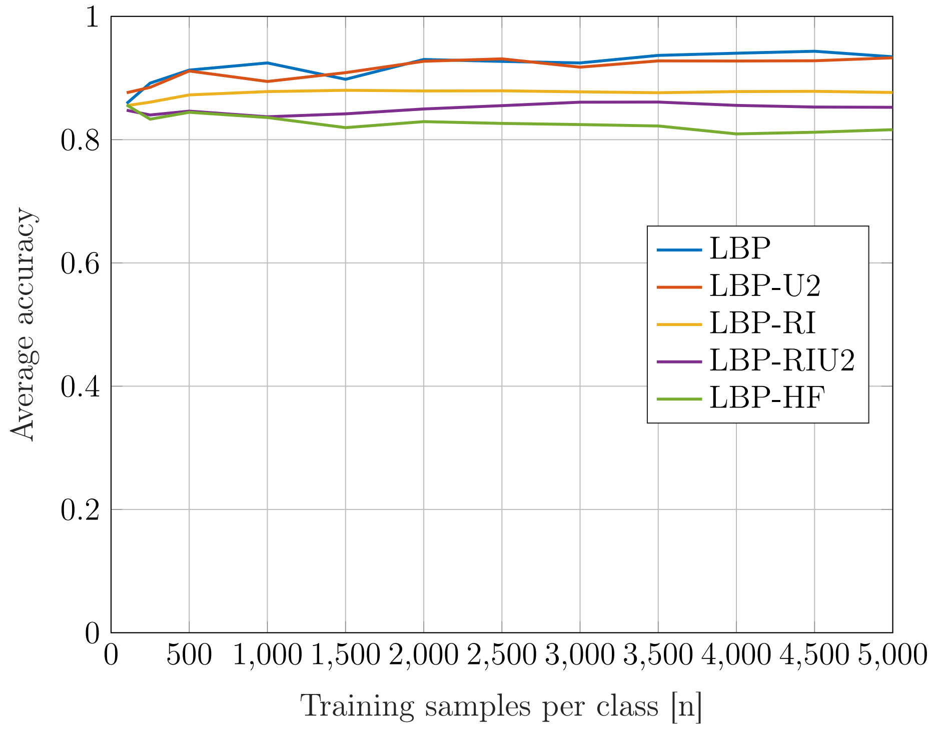


Figure 5.3: Average accuracy of linear SVM classifiers.

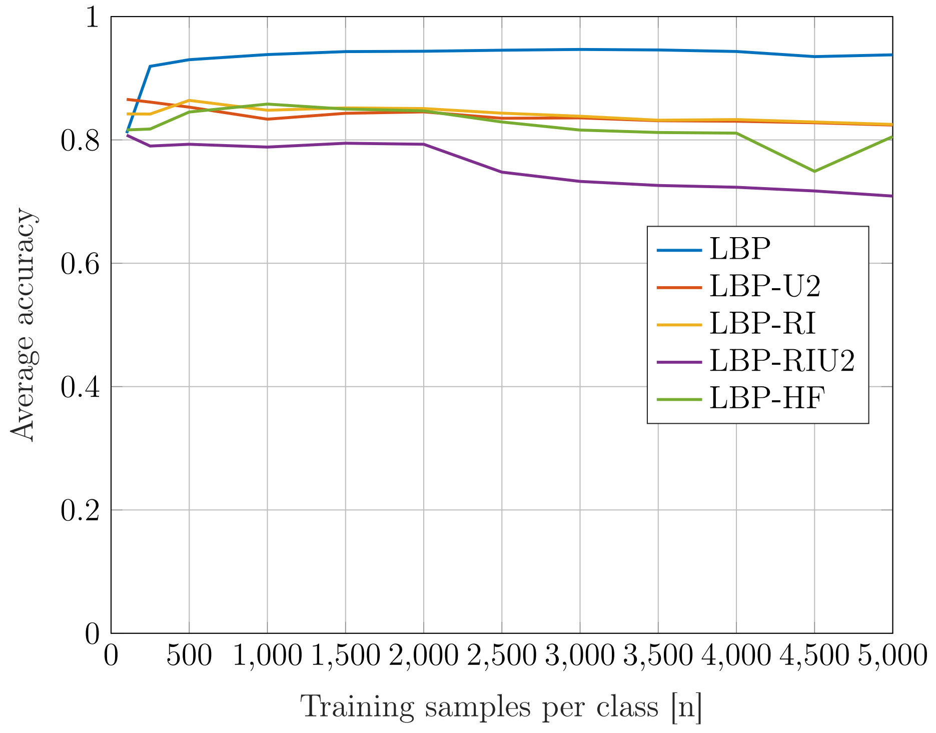


Figure 5.4: Average accuracy of radial basis function SVM classifiers.

5.3 Precision

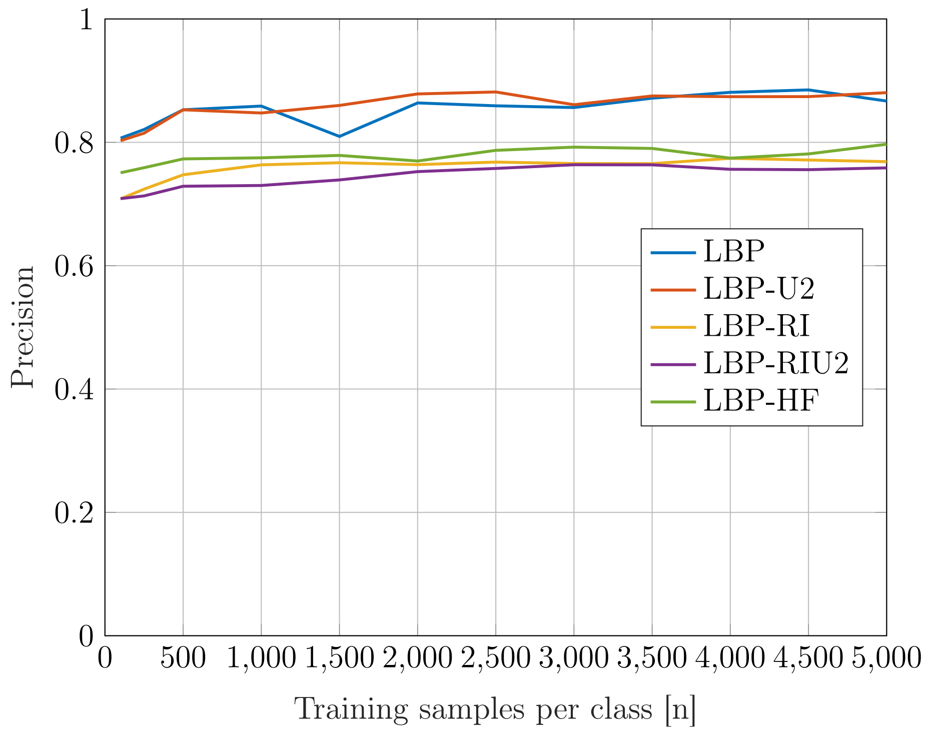


Figure 5.5: Precision of linear SVM classifiers.

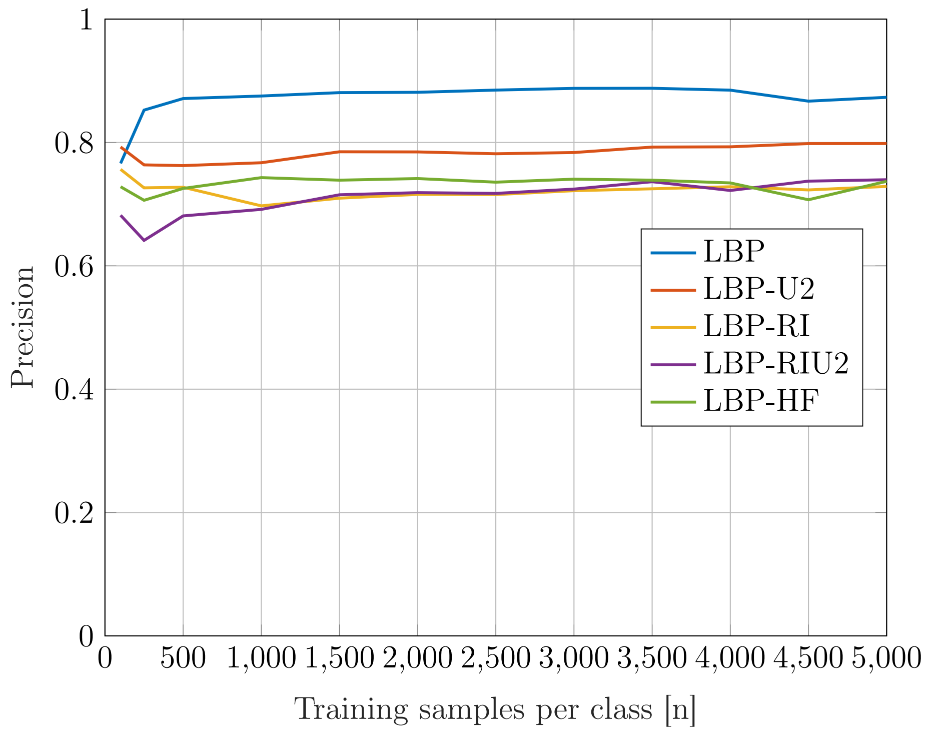


Figure 5.6: Precision of radial basis function SVM classifiers.

5.4 Recall

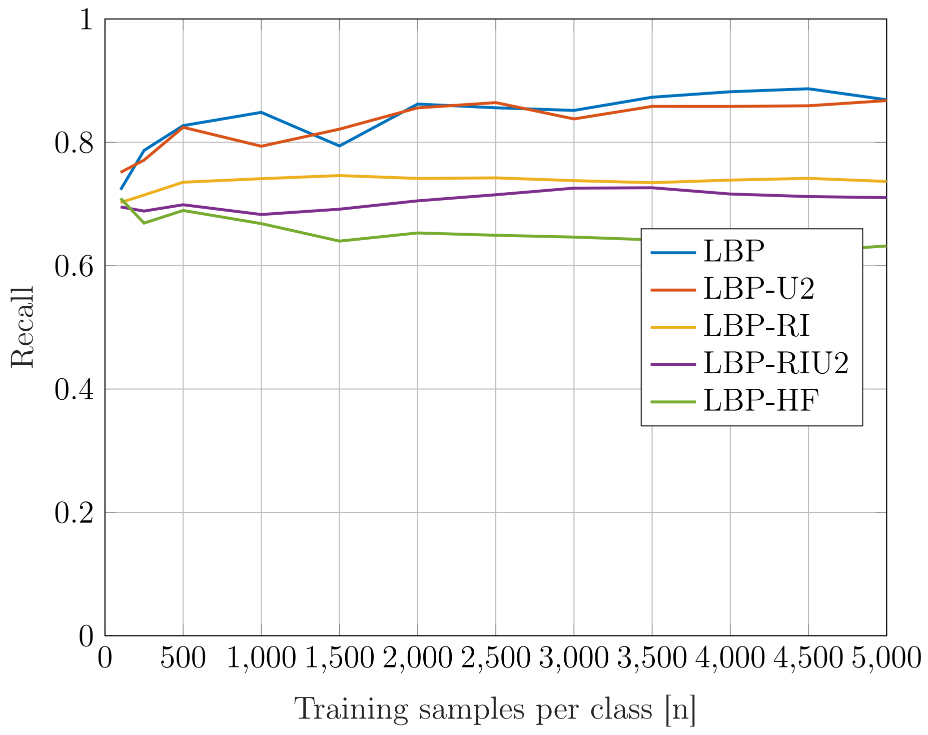


Figure 5.7: Recall of linear SVM classifiers.

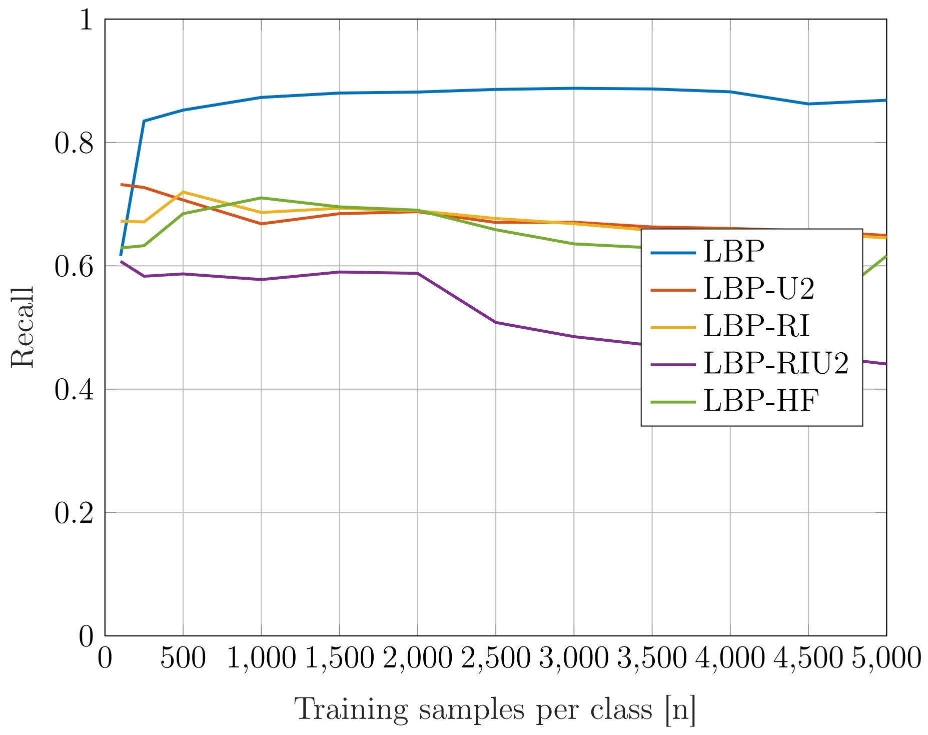


Figure 5.8: Recall of radial basis function SVM classifiers.

5.5 Error rate

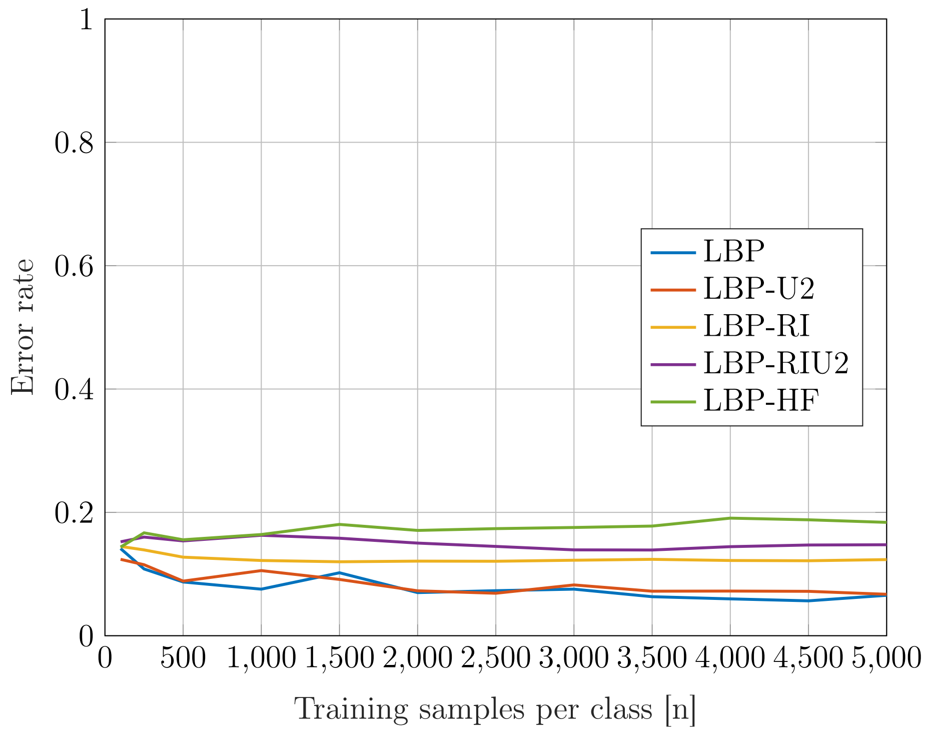


Figure 5.9: Error rate of linear SVM classifiers.

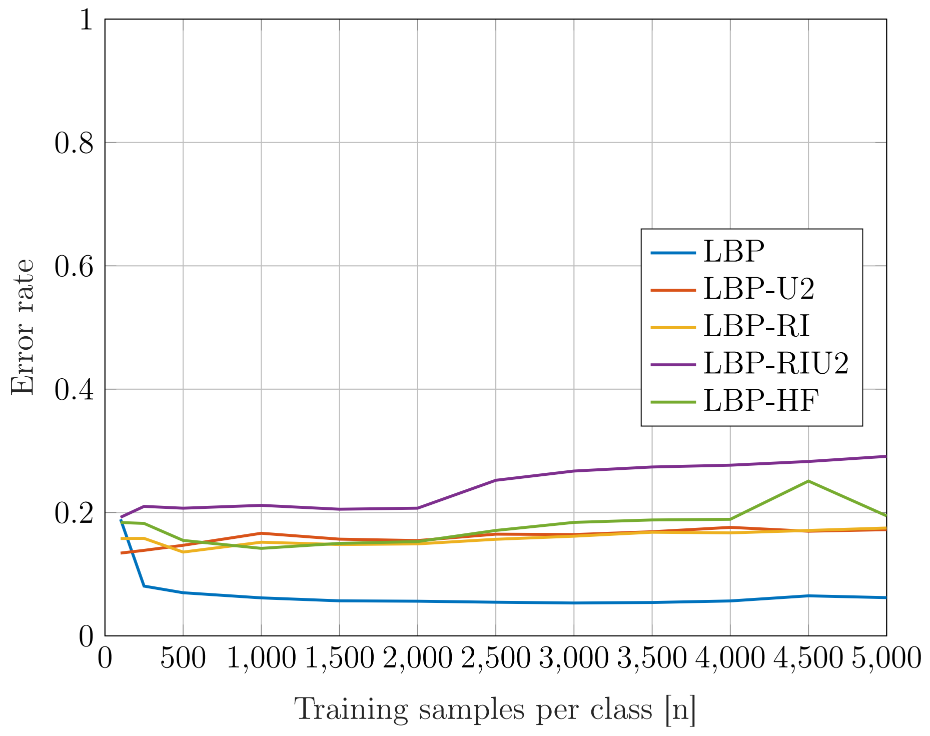


Figure 5.10: Error rate of radial basis function SVM classifiers.

5.6 F1-measure

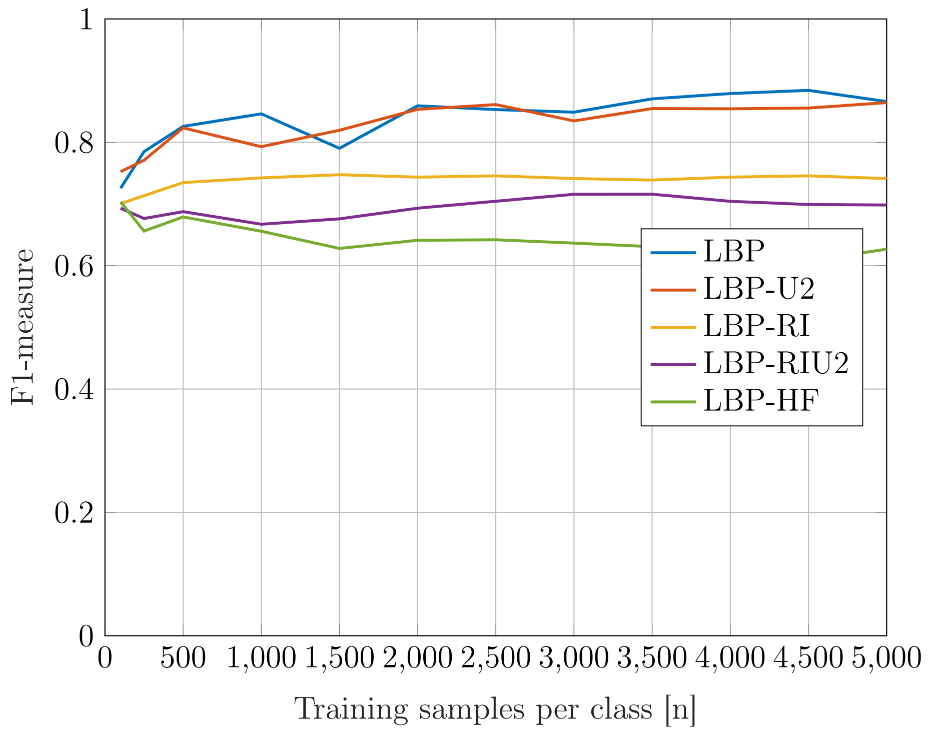


Figure 5.11: F1-measure of linear SVM classifiers.

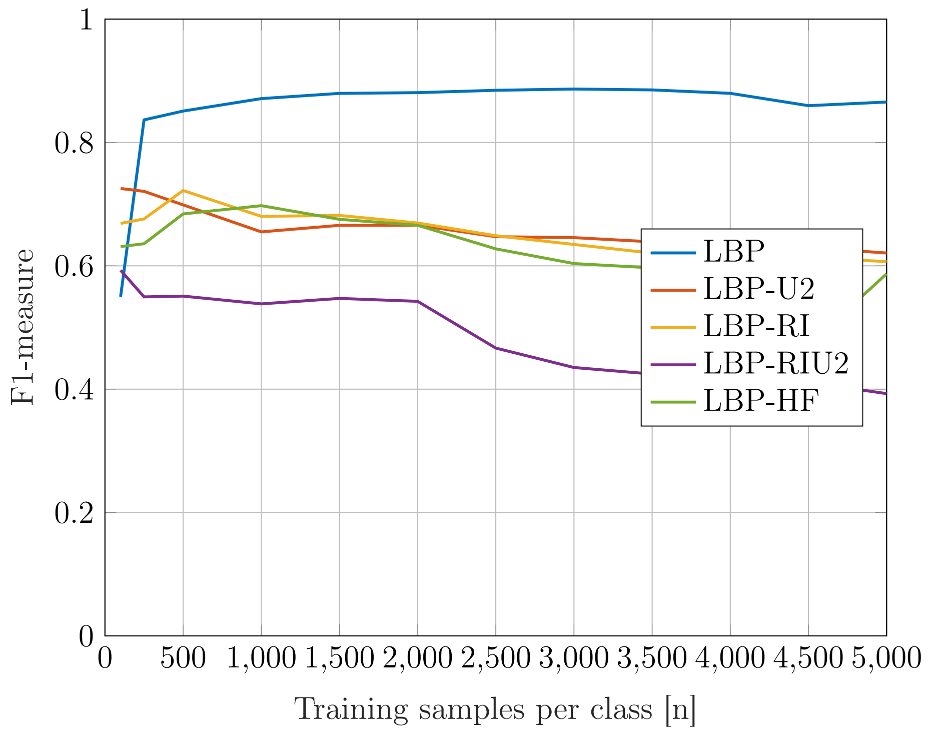


Figure 5.12: F1-measure of radial basis function SVM classifiers.

5.7 Training duration

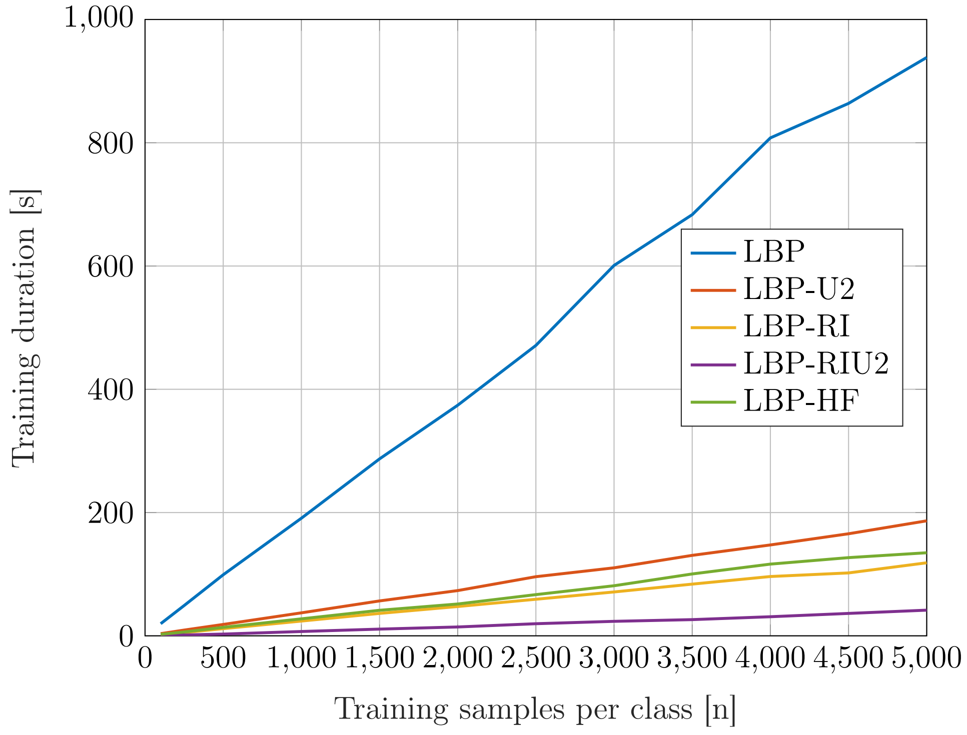


Figure 5.13: Duration of training for linear SVM classifiers.

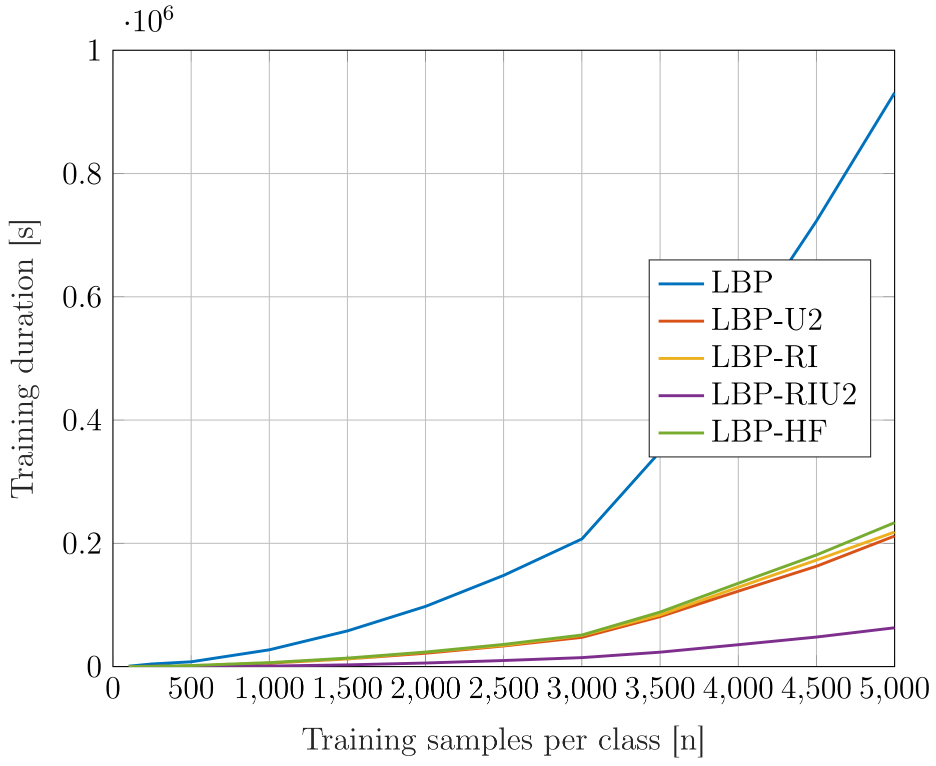


Figure 5.14: Duration of training for radial basis function SVM classifiers.

Chapter 6

Discussion

This chapter contains a discussion of the results of the performance measurements defined in Chapter 4 as presented in Chapter 5. The subsequent paragraphs discuss the results of the measurements of the average accuracy, precision, recall, error rate and F1-score. The cross-validation average accuracy of the classifiers are also discussed. The cross-validation average accuracy of a classifier is the average accuracy measured on a subset of the image patches used for training that were not used in training. The cross-validation average accuracy is used to choose the parameters of the SVM classifiers during training. Then, the training duration of the SVM classifiers and their relation to the performance metrics is discussed. Finally, a general discussion of the effectiveness of the classifiers is supplied.

The results of the measurements of cross-validation average accuracy, average accuracy, precision, recall, error rate and F1-measure on the SVM classifiers trained using a linear kernel are shown in Figures 5.1, 5.3, 5.5, 5.7, 5.9 and 5.11 respectively. It can be seen that the SVM classifiers trained using a linear kernel achieve very high cross-validation accuracies and relatively high average accuracies. The linear classifiers based on LBP and LBP-U2 codes achieve the highest cross-validation average accuracies and average accuracies at **~0.98**, **~0.97**, **~0.96** and **~0.95** respectively. The high scores are to be expected as the LBP codes used to describe the images contain the most image information among those tested. The cross-validation average accuracy and average accuracy of the linear classifiers based on the LBP and LBP-U2 are also nearly equal, as can be expected when taking into account their definitions as supplied in Subsection 2.2. Consequently the error rates of the linear SVM classifiers based on LBP and LBP-U2 codes are very low with a lowest error rate at **~0.04** and **~0.05** respectively.

The measured cross-validation average accuracies and average accuracies of the linear classifiers based on LBP-RI and LBP-RIU2 codes are somewhat lower than for the ones based on LBP and LBP-U2 codes at a highest cross-validation average accuracy and highest average accuracy of **~0.90** and **~0.87** for the classifiers based on LBP-RI codes and **~0.87** and **~0.86** for the classifiers based on LBP-RIU2 codes. However, the results of the linear SVM classifiers based on LBP-RI codes are more consistent as more image patches are used in training. This could indicate that the scheme utilized to achieve rotation invariancy for the LBP-RI and LBP-RIU2 codes is working. The overall lower average accuracies could indicate that the linear LBP-RI and LBP-RIU2 code classifiers suffer from the lower discriminative power of the shorter feature vectors. The error rates of the linear SVM classifiers based on LBP-RI and LBP-RIU2 codes are higher than for the linear SVM classifiers based on LBP and LBP-U2 codes at a lowest error rate of **~0.13** and **~0.14** respectively.

Surprisingly, the linear SVM classifiers based on the LBP-HF codes achieve high cross-validation average accuracies while achieving the lowest average accuracies at a highest cross-validation average accuracy of **~0.94** and a highest average accuracy of **~0.85**. This could indicate that the feature vectors of the query image patches in their Fourier transformed state as defined in Subsection 2.2 are not linearly seperable, or that images patches used in training are not representative enough of the query image patches. The error rates for the linear SVM classifiers based on LBP-HF codes are slightly higher than for the linear SVM classifiers based on LBP-RI and LBP-RIU2 codes with a lowest error rate at **~0.15**.

The measured precisions of the linear SVM classifiers based on LBP and LBP-U2 codes are fairly high, with a highest precision of **~0.90** for the classifiers based on LBP codes and **~0.89** for the classifiers based on LBP-U2 codes. This indicates that these classifiers mostly agree on which class label to label a query image patch with. However, these results are not very consistent, which could indicate that the classifiers are easily affected by changes in the image patches used for training.

The measured precisions of the linear SVM classifiers based on LBP-RI and LBP-RIU2 codes are considerably lower than for the linear SVM classifiers based on LBP and LBP-U2 codes. The highest measured precision of the classifiers based on LBP-RI and LBP-RIU2 codes are **~0.80** and **~0.77** respectively. This further strengthens the theory that the shorter length of the feature vectors utilized by these classifiers result in a lower discriminative power.

For the linear SVM classifiers based on LBP-HF codes the measured precisions are slightly higher than for the linear SVM classifiers based on LBP-RI and LBP-RIU2 codes. This is expected as the longer feature vector length used by the LBP-HF code scheme allows for higher discriminative power between the classes.

The linear SVM classifiers based on LBP and LBP-U2 codes achieve very inconsistent results when it comes to the measured recall of each classifier, indicating that the efficiency of the classifiers is easily affected by the image patches used for training. The highest measured recall of the linear SVM classifiers based on LBP and LBP-U2 codes are **~0.89** and **~0.86** respectively.

The measured recall of the linear SVM classifiers based on LBP-RI and LBP-RIU2 codes are consistantly lower than the measured recall for the linear SVM classifiers based on LBP and LBP-U2 codes. The results also show that the measured recall for these classifiers is more consistent when more imaga patches are used for training. The highest measured recall of the linear SVM classifiers based on LBP-RI and LBP-RIU2 codes are **~0.75** and **~0.73** respectively.

For the linear SVM classifiers based on LBP-HF codes the measured recall of each classifier is significantly lower than the measured recall of the linear SVM classifiers based on LBP-RI and LBP-RIU2. The highest measured recall of the linear SVM classifiers based on LBP-HF codes is **~0.68**, which shows that these classifiers are not very effective when assigning class labels.

The results of the F1-measures for the linear SVM classifiers based on the different adaptations of the LBP codes show the same results as those presented in the results for precision and recall of the same classifiers. The linear SVM classifiers based on LBP and LBP-U2 codes achieve higher F1-measures than the classifiers based on LBP-RI and LBP-RIU2 codes, albeit being easily affected by the image patches used for training. The highest F1-measure achieved by the classifiers based on LBP and LBP-U2 codes are **~0.88** and **~0.86** respecitvely. The linear SVM classifiers based on LBP-RI and LBP-RIU2 codes achieve higher F1-measures than the classifiers based on LBP-HF codes, again shown in the results presented for the precision and recall of the same classifiers. The highest measured F1-measure of the linear SVM classifiers based on LBP-RI, LBP-RIU2 and LBP-HF codes are **~0.75**, **~0.73** and **~0.69** respectively.

The results of the measurements of cross-validation average accuracy, average accuracy, precision, recall, error rate and F1-measure of the SVM classifiers trained using a non-linear RBF kernel are shown in Figures 5.2, 5.4, 5.6, 5.8, 5.10 and 5.12 respectively. It can be seen that the SVM classifiers trained using a non-linear RBF kernel achieve slightly higher cross-validation average accuracies but significantly lower average accuracies than the classifiers based on LBP-U2 codes trained using a linear kernel. The highest cross-validation average accuracies and average accuracies achieved by the non-linear SVM classifiers based on LBP and LBP-U2 codes are **~0.98**, **~0.94**, **~0.98** and **~0.87** respectively. This might indicate that the non-linear SVM classifiers are good at generalizing during training, but are very susceptible to changes between the training and testing data. Surprisingly the average accuracies of the classifiers based on LBP codes do not seem to be as susceptible to the differences between training and testing image data, and maintain a high average accuracy. This might be because the longer feature vectors of the classifiers based on LBP codes contain more information on the image patches and can hence be successfully both non-linearly and linearly mapped. The associated lowest error rates of the non-linear SVM classifiers based on LBP and LBP-U2 codes are **~0.06** and **~0.13** respectively.

The classifiers trained using a non-linear RBF kernel based on LBP-RI and LBP-RIU2 codes again achieve slightly higher cross-validation average accuracies and lower average accuracies than their linear kernel counterparts. The highest cross-validation average accuracy of the non-linear SVM classifiers based on LBP-RI and LBP-RIU2 codes are **~0.94** and **~0.94** respectively. However, the average accuracies of the non-linear SVM classifiers based on LBP-RI and LBP-RIU2 codes are different than in the linear case. It can be seen that the average accuracies of the classifiers based on LBP-RI codes are now more similar to the ones achieved by the classifiers based on the LBP-U2 and LBP-HF codes. The non-linear SVM classifiers based on LBP-RIU2 codes achieve considerably worse than in the linear case with the average accuracies being consistantly low until a certain point at which the classifiers start to overfit, and the performance is further lowered. Overfitting is a phenomenon that happens when a classification model becomes excessively complex and starts to describe random noise or errors instead of the underlying relationships that it should describe. This can happen when the classification model is trained on the same type of image data for too long. The highest average accuracy of the non-linear SVM classifiers based on LBP-RI and LBP-RIU2 codes are **~0.86** and **~0.81** respectively. Consequently, the lowest error rates of the same classifiers are **~0.14** and **~0.19**.

The measured cross-validation average accuracy and average accuracy of the non-linear SVM classifiers based on LBP-HF codes are both slightly higher than for the classifiers trained using a linear kernel. The non-linear SVM classifiers based on LBP-HF codes achieve higher cross-validation average accuracies than both the non-linear classifiers based on LBP-RI and LBP-RIU2 codes with a highest cross-validation average accuracy of **~0.95**. The non-linear SVM classifiers based on LBP-HF codes achieve a highest average accuracy of **~0.87**. The lowest error rate of the same classifiers is **~0.13**.

The measured precisions of the non-linear SVM classifiers based on LBP and LBP-U2 codes show much of the same results as that of the cross-validation average accuracy and average accuracy mentioned above. For the classifiers based on the LBP codes the precision is consistently high, even with a larger number of image patches used for training. The highest measured precision of the non-linear SVM classifiers based on LBP codes is **~0.90**. For the classifiers based on the LBP-U2 codes the measured precisions are considerably lower with a highest measured precision of **~0.80**. These results reassure the theory that the longer feature vectors of the classifiers based on LBP codes contain more image information that makes them easier to separate in both linear and non-linear space.

The measured precisions of the non-linear SVM classifiers based on LBP-RI and LBP-RIU2 codes are also lower than when the classifiers were trained using a linear kernel, albeit not as drastically as with the classifiers based on LBP-U2 codes. The results show the same consistensy as more image patches are added to the training set as when the classifiers were trained using a linear kernel. The highest measured precision of the non-linear SVM classifiers based on LBP-RI and LBP-RIU2 are **~0.73** and **~0.72**.

For the non-linear SVM classifiers based on LBP-HF codes the measured precisions show the same trend when more image patches is added to the training set as or the classifiers trained using a linear kernel, although the measurements are consistently lower. The highest measured precision of the non-linear SVM classifiers based on LBP-RI is **~0.73**, achieving the same results as for the classifiers based on LBP-RIU2 codes.

The measured recall for each of the non-linear SVM classifiers based on LBP and LBP-U2 codes show that the trend from the above sections continues. The classifiers based on LBP codes achieve considerably higher recall than the classifiers based on LBP-U2 codes with a highest score measured recall of **~0.88**. This is again on par with the results from the classifiers trained using a linear kernel. However, these results are much more consistent as more image patches are added to the training set. The non-linear SVM classifiers based on LBP-U2 codes achieve a highest measured recall at **~0.73**, but the results drop considerably when more image patches are added to the training set.

For the classifiers based on LBP-RI codes that were trained using a non-linear kernel the measured recall of each classifier follows the trend of the non-linear classifiers based on LBP-U2 codes. The highest measured recall of the classifiers based on LBP-RI is found when few image patches are used for training, at **~0.71**. For the non-linear SVM classifiers based on LBP-RIU2 codes the measured recall of each classifier is more or less consistent until a certain amount of image patches are added to the training set, and the classifiers are overfit. The highest measured recall of the classifiers based on LBP-RIU2 is **~0.61**.

The measured recall of each classifier for the non-linear SVM classifiers based on LBP-HF codes is higher than for the classifiers trained using a linear kernel. This might be because the Fourier transformed histograms of LBP-HF codes used as feature vectors are more easily seperable in a more high dimensional space, such as the ones used by the RBF kernel. The highest measured recall of the non-linear SVM classifiers based on LBP-HF codes is **~0.70**.

The non-linear SVM classifiers based on the different adaptations of the LBP codes show much of the same results as for the classifiers trained using a linear kernel. The classifiers based on LBP achieve considerably higher F1-measures than for any of the other classifier types. The highest measured recall of the non-linear SVM classifiers based on LBP codes is **~0.89**. The F1-measures of the non-linear SVM classifiers based on the LBP-U2, LBP-RI and LBP-RIU2 codes follow the same trend as described above for the precision and recall of the same classifiers. It is again evident that the classifiers based on LBP-RIU2 codes are overfit after a certain number of image patches are added to the training set. The highest measured F1-measure of the non-linear SVM classifiers based on LBP-U2, LBP-RI, LBP-RIU2 and LBP-HF codes are **~0.73**, **~0.72**, **~0.69** and **~0.59** respectively.

The training duration in seconds is shown in Figures 5.13 and 5.14 for each of the linear and non-linear SVM classifiers that were trained for the object recognition framework implemented for this thesis. From this it is clear that the SVM classifiers trained using a non-linear kernel suffer from exponentially growing training durations, which get very long when the number of image patches in the training set gets large. It is also clear that the SVM classifiers trained using a linear kernel do not suffer from this problem, and can be trained in linear time based on how many image patches are in the training set. The reduced training duration is one benefit of using a linear kernel, although it might come at the cost of less discriminiative power. Luckily, this does not seem to be much of a concern with the SVM classifiers based on adaptations of the LBP codes, as the results were actually worse when utilizing a non-linear kernel than when a linear kernel was used.

In summary the SVM classifiers trained using a linear kernel function achieved overall better scores than the classifiers trained using a non-linear kernel function on the performance measures used for this thesis. The two exceptions to this were the higher cross-validation average accuracies shown by the classifiers trained using the RBF kernel and the consistantly higher scores of the non-linear SVM classifiers based on the LBP-HF codes. There could be a number of reasons for this behaviour. The most obvious is that the feature vectors based on LBP, LBP-U2, LBP-RI and LBP-RIU2 codes could already be easily separable in a linear space, and so a non-linear feature space is redundant except in the case of the classifiers based on the LBP-HF codes. Also, the low average accuracy of the non-linear SVM classifiers based on the LBP-U2, LBP-RI and LBP-RIU2 codes in comparison to their respective cross-validation accuracies could be an indication that the classifiers get easily overfit when utilizing the non-linear RBF kernel. The high scores of the non-linear classifiers based on the LBP codes could also indicate that the RBF kernel demands feature vectors of a certain size that are separable in certain dimensions in order to be successfull. The SVM classifiers trained using a linear kernel also benefit from significantly reduced training durations in comparison to the classifiers trained using a non-linear kernel.

Chapter 7

Further work

This chapter contains suggestions for further work on the object recognition framework implemented for this thesis. It includes approaches to speeding up the overall data flow of the framework, approaches to achieving higher recognition rates and suggestions what the next step to completing a system for monitoring fish welfare in an underwater environment should be.

One flaw of the object recognition framework implemented for this thesis is the speed at which it operates. The framework is currently able to handle a maximum of one frame per second for 1920x1080 pixel images, which is not enough for many purposes. The current framework uses a mutex to allow for real-time capture of image data, but the processing of the captured image is not done in real-time. To achieve higher framerates two approaches could be taken. The first is simply to parallelize the processes at relevant places in the code, so as to allow faster processing of the images. This could for example be done at the level of feature extraction as defined in 2.2 for the number of blocks in each block-row of the image patch that is to be classified. If the number of blocks for each image patch is 5x5 this could speed up the feature extraction process five-fold. Paralellization of the code could also be done at the level of the image pyramids as defined in 2.4.3 where the search through each level of the pyramids could be paralellized to achieve faster processing of the images. If the number of levels in the pyramids is large this could also speed up the framework considerably. The other approach to speeding up the object recognition framework is to completely rewrite the code to allow for computation on a graphics processing unit (GPU). A GPU consist of hundreds of computing cores and is able to significantly speed up many tasks, especially processes that work on images, such as the recognition framework implemented for this thesis.

An approach to achieving higher recognition rates is simply to fine-tune the parameters that decide how the image material is treated at each stage of the framework. For the performance evaluation conducted for this thesis the parameters R, P and b that decide how the extraction of LBP codes should be completed was set constant for the sake of consistensy when performing the performance tests, as mentioned in Chapter 4. These parameters could be unsuited for some or all of the adaptations of the LBP codes. The parameters for C and γ used in training the linear and non-linear SVM classifiers were chosen by completing five-fold cross validation in the form of a line/grid search over the parameter space. By fine tuning the above mentioned parameters higher recognition rates could be achieved. Another approach could be to simply choose another non-linear kernel for the non-linear SVM classifiers. A non-linear kernel that operates in a feature space of lower dimension could improve the recognition rates. An approach to achieving higher recognition rates by reducing the number of false positives for each class is to perform hard negative mining. Hard negative mining works by feeding images through the object recognition framework and looking at the prediction that the framework provides for each image. If the framework predics wrong according to a human supervisor the wrongly predicted image is added to the training set of the class that it correctly belongs to, and the framework is trained again. The process is repeated until the the framework can be improved no further or until some sufficient recognition rate is achieved. This technique has been shown to improve recognition rates for many recognition frameworks.

The next step in constructing an automated system for the monitoring of fish welfare based on machine learning and computer vision algorithms should actually be to take a step back. Although the object recognition framework implemented for this thesis achieves the task it was constructed to do there are some aspects of the framework that should be improved before moving forward on the overall system. One of those aspects is the image features on which the object recognition is based. Even though the LBP codes utilized for object recognition by the framework presented in this thesis have many desirable traits such as illumination and rotation invariance they are slow to compute in comparison to other contemporary image features. One way to improve the object recognition framework could be to first use the current slow framework to acquire large amounts of correctly labeled image data. This correctly labeled image data could be used to train another faster framework based on a contemporary neural net architecture. This could lower the processing time and increase recognition rates of the overall system considerably. Only when recognition rates and the processing time of the object recognition framework are at sufficient levels should the project move into the next stage of the three-step plan mentioned in Chapter 1, that of detecting damage and sickness on fish parts from the correctly classified image material.

Chapter 8

Conclusion

This master thesis has seen the proposition and implementation of an object recognition system for use in underwater environments. The proposed approach combines the illumination, localization, rotation and scale-invariance of different adaptations of the LBP image patch descriptor with the high discriminative power of SVM classifiers to achieve high classification accuracies.

Specifically, the aim of the work conducted for this master thesis has been to investigate a possible approach to recognizing fish parts in a video stream from a camera system situated in an underwater environment by using machine learning and computer vision algorithms. In this it can be said that the project has succeeded. An object recognition framework has been implemented that is able to take in images from an IP camera communicating over RTSP. The same system is able to classify novel image patches not used for training with a high accuracy by relying on a linear SVM for classificaton. However, the goal of a complete system for the recognition of fish parts in a live video stream has not been reached. The classifers trained on the training images acquired at the test facility at Kåholmen on Hitra, Norge is not able to classify image patches aquired from a video stream taken at a later time than that of the training image patches. It is suspected that this is because the image patches that have been used for training are not representative enough of this new case. Surprisingly, the non-linear SVM classifiers relying on the RBF kernel achieve much lower performance than most of the linear SVM classifiers.

That being said the use of computer vision and machine learning algorithms in the task of ensuring fish welfare in fish farms is an exciting possibility that should be continued to be worked either in the fashion suggested by this thesis or with some other approach.

Bibliography

Achatz, S., 2016. State of the art of object recognition techniques.
URL https://www.nst.ei.tum.de/fileadmin/w00bqs/www/publications/as/SS2016_AS_ObjectRecogn.pdf

Ahonen, T., Hadid, A., Pietikäinen, M., 2004. Face Recognition with Local Binary Patterns. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 469–481.
URL http://dx.doi.org/10.1007/978-3-540-24670-1_36

Bay, H., Ess, A., Tuytelaars, T., Gool, L. V., Leuven, B. K. U., 2006. Speeded-up robust features (surf).

Bay, H., Ess, A., Tuytelaars, T., Van Gool, L., Jun. 2008. Speeded-up robust features (surf). Comput. Vis. Image Underst. 110 (3), 346–359.
URL http://dx.doi.org/10.1016/j.cviu.2007.09.014

Bertsekas, D. P., Nedic, A., Ozdaglar, A. E., 2001. Convexity, duality, and lagrange multipliers.

Boser, B. E., Guyon, I. M., Vapnik, V. N., 1992. A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory. COLT '92. ACM, New York, NY, USA, pp. 144–152.
URL http://doi.acm.org/10.1145/130385.130401

Boyd, S. P., Xiao, L., Mutapcic, A., 2003. Subgradient methods.

Calonder, M., Lepetit, V., Strecha, C., Fua, P., 2010. Brief: binary robust independent elementary features. In: Proceedings of the 11th European conference on Computer vision: Part IV. ECCV'10. Springer-Verlag, Berlin, Heidelberg, pp. 778–792.
URL http://dl.acm.org/citation.cfm?id=1888089.1888148

Chang, C.-C., Lin, C.-J., 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chen, P., Liu, S., 2009. An improved dag-svm for multi-class classification. In: Proceedings of the 5th International Conference on Natural Computation. ICNC'09. IEEE Press, Piscataway, NJ, USA, pp. 460–462.
URL http://dl.acm.org/citation.cfm?id=1797096.1797193

Cortes, C., Vapnik, V., Sep 1995. Support-vector networks. Machine Learning 20 (3), 273–297.
URL https://doi.org/10.1023/A:1022627411411

Crow, F. C., Jan. 1984. Summed-area tables for texture mapping. SIGGRAPH Comput. Graph. 18 (3), 207–212.
URL http://doi.acm.org/10.1145/964965.808600

Dashore, G., 2012. An efficient method for face recognition using principal component analysis (pca). In: International Journal of Advanced Technology & Engineering Resear.

Dietterich, T. G., Bakiri, G., 1995. Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2, 263–286.

Dollár, P., Zitnick, C. L., 2013. Structured forests for fast edge detection. In: Proceedings of the 2013 IEEE International Conference on Computer Vision. ICCV '13. IEEE Computer Society, Washington, DC, USA, pp. 1841–1848.
URL http://dx.doi.org/10.1109/ICCV.2013.231

Dollár, P., Zitnick, C. L., 2014. Fast edge detection using structured forests. IEEE Trans. Pattern Anal. Mach. Intell. 37 (8), 1558–1570.
URL http://dblp.uni-trier.de/db/journals/pami/pami37.html#DollarZ15

Erhan, D., Szegedy, C., Toshev, A., Anguelov, D., 2014. Scalable object detection using deep neural networks. In: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. CVPR '14. IEEE Computer Society, Washington, DC, USA, pp. 2155–2162.
URL http://dx.doi.org/10.1109/CVPR.2014.276

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., Lin, C.-J., 2008. LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research 9, 1871–1874.

Frigo, M., May 1999. A fast fourier transform compiler. SIGPLAN Not. 34 (5), 169–180.
URL http://doi.acm.org/10.1145/301631.301661

Gill, P. E., Wong, E., jul 2014. Methods for convex and general quadratic programming.

Girshick, R., 2015. Fast r-cnn. In: International Conference on Computer Vision (ICCV).

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Computer Vision and Pattern Recognition.

Hariharan, B., Malik, J., Ramanan, D., 2012. Discriminative decorrelation for clustering and classification. In: ECCV (4). Vol. 7575 of Lecture Notes in Computer Science. Springer, pp. 459–472.
URL http://dblp.uni-trier.de/db/conf/eccv/eccv2012-4.html#HariharanMR12

He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep residual learning for image recognition. CoRR abs/1512.03385.
URL http://arxiv.org/abs/1512.03385

Hsu, C.-W., Lin, C.-J., mar 2002. A comparison of methods for multiclass support vector machines. Trans. Neur. Netw. 13 (2), 415–425.
URL `http://dx.doi.org/10.1109/72.991427`

Ke, Y., Sukthankar, R., 2004. Pca-sift: A more distinctive representation for local image descriptors. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 506–513.

Kim, B., Yu, S. C., Feb 2017. Imaging sonar based real-time underwater object detection utilizing adaboost method. In: 2017 IEEE Underwater Technology (UT). pp. 1–5.

Krizhevsky, A., Sutskever, I., Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C. J. C., Bottou, L., Weinberger, K. Q. (Eds.), Advances in Neural Information Processing Systems 25. Curran Associates, Inc., pp. 1097–1105.
URL `goo.gl/36FuDW`

Leutenegger, S., Chli, M., Siegwart, R. Y., Nov. 2011. BRISK: Binary Robust invariant scalable keypoints. In: Computer Vision (ICCV), 2011 IEEE International Conference on. IEEE, pp. 2548–2555.
URL `http://dx.doi.org/10.1109/iccv.2011.6126542`

Li, Q., Niaz, U. F., Merialdo, B., 06 2012. An improved algorithm on viola-jones object detector. In: CBMI 2012, 10th Workshop on Content-Based Multimedia Indexing, June 27-29, 2012, Annecy, France. Annecy, FRANCE.
URL `http://www.eurecom.fr/publication/3745`

Li, X., Shang, M., Qin, H., Chen, L., Oct 2015. Fast accurate fish detection and recognition of underwater images with fast r-cnn. In: OCEANS 2015 - MTS/IEEE Washington. pp. 1–5.

Liao, S., Zhu, X., Lei, Z., Zhang, L., Li, S. Z., 2007. Learning Multi-scale Block Local Binary Patterns for Face Recognition. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 828–837.
URL `http://dx.doi.org/10.1007/978-3-540-74549-5_87`

Lienhart, R., Maydt, J., 2002. An extended set of haar-like features for rapid object detection. IEEE ICIP 2002, 900–903.

Lindahl, T., 2007. Study of local binary patterns. Master's thesis, Linköbing University.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., Berg, A. C., 2015. SSD: single shot multibox detector. CoRR abs/1512.02325.
URL `http://arxiv.org/abs/1512.02325`

Lowe, D. G., 1999. Object recognition from local scale-invariant features.

Lowe, D. G., 2003. Distinctive image features from scale-invariant keypoints.

Malagón-Borja, L., Fuentes, O., 2009. Object detection using image reconstruction with pca. Image Vision Comput. 27, 2–9.

Nagaraja, S., Prabhakar, C. J., Kumar, P. U. P., 2015. Extraction of Texture Based Features of Underwater Images Using RLBP Descriptor. Springer International Publishing, Cham, pp. 263–272.
URL `http://dx.doi.org/10.1007/978-3-319-12012-6_29`

Nayart, S. K., Nenet, S. A., Murase, H., 1996. Real-time 100 object recognition system. In: International Conference on Robotics and Automation.

Nunn, C., Kummert, A., Muller, D., Meuter, M., Muller-Schneiders, S., Oct 2009. An improved adaboost learning scheme using lda features for object recognition. In: 2009 12th International IEEE Conference on Intelligent Transportation Systems. pp. 1–6.

Ojala, T., Pietikäinen, M., Harwood, D., 1995. A comparative study of texture measures with classification based on featured distributions. Pattern Recognition 28, 50–60.

Paris, S., Hasinoff, S. W., Kautz, J., Jul. 2011. Local laplacian filters: Edge-aware image processing with a laplacian pyramid. ACM Trans. Graph. 30 (4), 68:1–68:12.
URL `http://doi.acm.org/10.1145/2010324.1964963`

Prabhakar, C. J., Kumar, P. U. P., 2012. Lbp-surf descriptor with color invariant and texture based features for underwater images. In: Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing. ICVGIP '12. ACM, New York, NY, USA, pp. 23:1–23:8.
URL `http://doi.acm.org/10.1145/2425333.2425356`

Redmon, J., Farhadi, A., 2016. Yolo9000: Better, faster, stronger. arXiv preprint arXiv:1612.08242.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (NIPS).

Rosten, E., Drummond, T., 2006. Machine learning for high-speed corner detection. In: In European Conference on Computer Vision. pp. 430–443.

Rublee, E., Rabaud, V., Konolige, K., Bradski, G. R., 2011. Orb: An efficient alternative to sift or surf. In: Metaxas, D. N., Quan, L., Sanfeliu, A., Gool, L. J. V. (Eds.), ICCV. IEEE Computer Society, pp. 2564–2571.
URL `http://dblp.uni-trier.de/db/conf/iccv/iccv2011.html#RubleeRKB11`

Shapiro, L., Stockman, G., 2001. Computer Vision. Prentice-Hall, Upper Saddle River, NJ.

Sokolova, M., Lapalme, G., Jul. 2009. A systematic analysis of performance measures for classification tasks. Inf. Process. Manage. 45 (4), 427–437.
URL `http://dx.doi.org/10.1016/j.ipm.2009.03.002`

Spampinato, C., Palazzo, S., Joalland, P. H., Paris, S., Glotin, H., Blanc, K., Lingrand, D., Precioso, F., Feb 2016. Fine-grained object recognition in underwater visual data. Multimedia Tools and Applications 75 (3), 1701–1720.
URL `http://dx.doi.org/10.1007/s11042-015-2601-x`

Svensen, R., Sep 2016. Laks i merd.
URL https://www.vetinst.no/nyheter/effektiv-beskyttelse-mot-lus-ved-produksjon-av-l
_/image/7521c54d-3490-4c4c-9671-a2cad4bffa39:57e55ce6ab0bf121ca426654a0e6c7deadfcd0l
max-2400/Laks%20i%20merd.%20Foto%20Rudolf%20Svensen.jpg

T., A., J., M., C., H., M., P., 2009. Rotation invariant image description with local binary pattern histogram fourier features. In: In: Image Analysis, SCIA 2009 Proceedings, Lecture Notes in Computer Science 5575, 61-70. ISBN 978-3-642-02229-6.

T., O., Äinen M., P., T., M., 2002. Multiresoluion gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7):971 - 987.

Tian, J., Zhang, T., April 2017. Sparse null space lda for object recognition. In: 2017 IEEE 2nd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA). pp. 316–320.

Topi, M., Timo, O., Matti, P., Maricor, S., 2000. Robust texture classification by subsets of local binary patterns. In: Proceedings 15th International Conference on Pattern Recognition. ICPR-2000. Vol. 3. pp. 935–938 vol.3.

Vapnik, V., Lerner, A. Y., 1963. Pattern recognition using generalized portrait method. Automation and Remote Control 24 (6), 774–780.

Vatani, N. N., 2013. Lbp. GitHub Repository.
URL https://github.com/nourani/LBP

Veterinærinstituttet, 2017. Fiskehelserapporten 2016. Fiskehelserapporten 2016.
URL https://www.vetinst.no/rapporter-og-publikasjoner/rapporter/2017/fiskehelserapporten-2016

Villon, S., Chaumont, M., Subsol, G., Villéger, S., Claverie, T., Mouillot, D., 2016. Coral Reef Fish Detection and Recognition in Underwater Videos by Supervised Machine Learning: Comparison Between Deep Learning and HOG+SVM Methods. Springer International Publishing, Cham, pp. 160–171.
URL http://dx.doi.org/10.1007/978-3-319-48680-2_15

Viola, P., Jones, M., 2001. Robust real-time object detection. In: International Journal of Computer Vision.

Wikipedia, July 2017. Bilinear interpolation.
URL https://en.wikipedia.org/wiki/Bilinear_interpolation

Wright, S. J., jun 2015. Coordinate descent algorithms. Math. Program. 151 (1), 3–34.
URL http://dx.doi.org/10.1007/s10107-015-0892-3

Zitnick, C. L., Dollár, P., 2014. Edge Boxes: Locating Object Proposals from Edges. Springer International Publishing, Cham, pp. 391–405.
URL https://doi.org/10.1007/978-3-319-10602-1_26

Abbreviations

SVM	Support Vector Machine
LBP	Local Binary Pattern
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
MB-LBP	Multi-scale Block Local Binary Pattern
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Invariant Scalable Keypoints
ORB	Oriented FAST and Rotated BRIEF
FAST	Features from Accelerated Segment Test
PCA	Principal Component Analysis
LDA	Linear Discriminant Analysis
AdaBoost	adaptive boosting
CNN	Convolutional Neural Net
RCNN	Region-based Convolutional Neural Net
SSD	Single Shot MultiBox Detector
ResNet	Residual Block Net
YOLO	You Only Look Once

RLBP	Robust Local Binary Pattern
LTP	Local Ternary Pattern
HOG	Histogram of Oriented Gradients
LBP-U2	Uniform Local Binary Pattern
LBP-RI	Rotation Invariant Local Binary Pattern
LBP-RIU2	Rotation Invariant Uniform Local Binary Pattern
LBP-HF	Local Binary Pattern Fourier Histogram
LSB	Least Significant Bit
MSB	Most Significant Bit
DFT	Discrete Fourier Transform
NMS	Non-Maximum Suppression
RBF	Radial Basis Function