Ultra-narrow Band Satellite Visual Monitoring of High-sea Fishing Vessels

Pedro Miguel Garcia Perdigão

Thesis to obtain the Master of Science in

Electrical and Computer Engineering

Supervisor: Prof. Fernando Manuel Bernardo Pereira
Prof. João Miguel Duarte Ascenso

Examination Committee

Chairperson: Prof. José Eduardo Charters Ribeiro da Cunha Sanguino
Supervisor: Prof. Fernando Manuel Bernardo Pereira
Member of the Committee: Prof. Jorge Dos Santos Salvador Marques
Eng. Pedro Miguel da Conceição Santos Lousã

June 2018

Acknowledgements

First of all, I would like to thank my parents João and Teresa, for keeping me positive, and my brother João, for always being there for me.

I would like to express my deep gratitude to Professors Fernando Pereira and João Ascenso, for their availability, advices and support throughout the course of this Thesis.

A special thanks Xsealence, for all the provided support.

I would also like to thank IPMA for providing the video resources that were the base for the tests.

To all my friends, especially the ones from the IT room, thank you for being there for this last year.

Finally, I wish to thank IT for being my host institution during the course of this Thesis.

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Resumo

O sector das pescas desempenha um papel muito importante em todo o mundo, especialmente em comunidades costeiras, contribuindo fortemente para o desenvolvimento local, emprego e para a manutenção de uma série de outras actividades económicas fundamentais. No entanto, a sobrepesca, isto é, o acto de pescar peixe em maior quantidade do que este se consegue reproduzir, tem consequências extremamente nefastas, afectando o equilíbrio natural dos oceanos bem como o bem-estar social económico das sociedades costeiras que dele dependem para sobreviver. É, portanto, de extrema importância combater a sobrepesca, de maneira a garantir a existência desta importantíssima fonte de alimento e bem-estar económico para futuras gerações.

De maneira a promover a sustentabilidade da vida marinha, várias técnicas de monitorização têm sido empregues ao longo dos anos. Uma destas técnicas de monitorização consiste na colocação de uma 'caixa preta' nos navios de pesca que, através de comunicação via-satélite, transmite informação referente a estes, tal como a sua posição e velocidade, para um Centro de Controlo em terra. De maneira a aumentar a eficácia deste tipo de monitorização, a possibilidade de instalar uma câmara a bordo do navio tem sido alvo de estudo, de forma a possibilitar a videovigilância a bordo dos navios. No entanto, o custo extraordinariamente elevado da comunicação via-satélite constitui uma forte barreira, tendo inviabilizado até agora a implementação de sistemas deste tipo.

Neste contexto, esta Tese de Mestrado teve dois objectivos: o primeiro prendeu-se com o desenvolvimento de um sistema automático de actividades piscatórias com base no sucesso recente das tecnologias *deep-learning*, nomeadamente usando as designadas redes neuronais convolucionais para fazer classificação de vídeo. Para treinar esta rede, foi criada uma base de dados contendo mais de 100 horas de vídeo obtido a partir de uma câmara colocada a bordo de um navio de pesca. O resultante sistema automático de vigilância de actividades piscatórias provou ter um excelente desempenho na detecção deste tipo de actividades, permitindo alertar para a ocorrência de actividades de pesca ilegal, bem como a transmissão de imagens relevantes desses eventos. O segundo objectivo prendeu-se com a necessidade de comprimir fortemente as imagens a transmitir para reduzir os custos de transmissão, nomeadamente usando codecs sem custos de licenciamento. Para melhorar a qualidade da experiência final foi também criado um sistema de melhoramento de imagens a ser colocado no Centro de Controlo, possibilitando assim uma maior compressão das imagens no navios e consequente redução adicional dos custos de transmissão. Este sistema, também baseado em *deep-learning*, usa uma rede neuronal convolucional que retira artefactos de compressão da imagem fortemente comprimida, melhorando substancialmente a qualidade final da imagem.

Palavras-chave: sobrepesca, sistemas de monitorização de navios de pesca, vídeo-vigilância, processamento de imagem, compressão de imagem, *deep-learning*, redes neuronais convolucionais, melhoramento de imagem.

Abstract

The fisheries sector plays a very important role throughout the world, making a major contribution to local development, employment and the sustainability of a number of other key economic activities. In this context, overfishing, i.e. fishing more fish than can be reproduced, has extremely harmful consequences, affecting the natural balance of the oceans as well as the economic and social well-being of the coastal communities that depend on it for survival. It is, therefore, extremely important to fight overfishing in order to guarantee the existence of this very important source of food and economical welfare for future generations.

In order to promote the sustainability of marine life, several monitoring techniques have been employed over the years. One of these monitoring techniques is to place a 'blackbox' in the fishing vessels which, via satellite communication, transmits relevant information, such as the boat position and speed, to a Control Center at shore. To increase the effectiveness of this type of monitoring, the implementation of an on-board camera has been subject of study to enable video surveillance on board of fishing vessels. However, the extraordinarily high cost of satellite-based communications is a major barrier, seriously hindering the implementation of such systems.

In this context, this MSc Thesis had two main objectives: the first concerned the development of an automatic system of fishing activities based on visual data. This system was based on the increasingly popular deep-learning networks, notably using the so-called Convolutional Neural Networks to provide video classification. To train this network, a dataset containing more than 100 hours of video obtained from a camera placed on board of a fishing vessel was created. The resulting automatic detection of fishing activities proved to have an excellent performance, thus allowing to alert the occurrence of illegal fishing activities, as well as the transmission of relevant images of those events. The second objective was the efficient compression of the images to be satellite transmitted, notably using a royalty free codec. To increase the final user experience, an image enhancement system to be placed in the Control Center was also developed, to allow performing a lower rate based image compression at the vessels and consequently reducing the transmission costs. This system, also based on deep-learning, uses a Convolutional Neural Network to remove the compression artifacts from the highly compressed image, substantially improving the final image quality.

Key-words: overfishing, Vessel Monitoring Systems, video-surveillance, image processing, image compression, deep-learning, Convolution Neural Networks, image enhancement.

Table of Contents

Acknowledgements iii

Declaration v

Resumo vii

Abstract ix

List of Figures xiv

List of tables xx

List of Acronyms xxi

1. Introduction 1

1.1 Context and Motivation 1

1.2 Objectives and Thesis Structure 3

2. Brief Review on Relevant Background Technologies 5

2.1 Deep Learning: Background and Main Concepts 5

2.1.1 Motivation 5

2.1.2 Background 6

2.1.3 Convolutional Neural Networks 8

2.2 Still Image Codecs: a Brief Review 12

2.2.1 JPEG Compliant Codecs 13

2.2.2 JPEG 2000 Codec 17

2.2.3 Daala Codec 18

2.2.4 VP9 Codec 21

2.2.5 Thor Codec 23

2.2.6 Still Image Coding Performance 24

3. Application Scenario: Requirements and Architecture 29

3.1 Vessel Monitoring Systems 29

3.1.1 VMS Solutions in the Market 30

3.1.2 VMS Architecture 31

3.2 Iridium Satellite Constellation 34

3.2.1 Short Burst Data Service 34

3.2.2 MO Messaging Operation 34
3.2.3 Iridium NEXT 36
3.3 Application Scenario Operation and Requirements 37
3.3.1 Operation 37
3.3.2 Requirements 41
4. Fishing Activity Detection and Classification Solution 43
4.1. Video Dataset and Ground Truth Definition 43
4.1.1 Video Acquisition 43
4.1.2 Video Dataset Characterization 44
4.1.3 Ground Truth Definition 46
4.2. Architecture and Walkthrough 48
4.3. Frame-level Fishing Event Detection 52
4.3.1 CNN Architecture Selection 52
4.3.2 AlexNet Architecture Overview 53
4.3.3 AlexNet Training 55
4.4. Performance Assessment 56
4.4.1 Training and Test Material 56
4.4.2 Performance Assessment Metrics and Procedures 58
4.4.3 Detection and Classification Results Analysis 63
4.4.4 Coding Results and Analysis 69
4.5 Proposed Solution Demonstration 73
5. Image Enhancement 75
5.1 Context and Motivation 75
5.2 Using CNNs for Image Enhancement 76
5.2.1 Enhancing by Super-resolution 77
5.2.2 Enhancing by Compression Artifacts Removal 77
5.3 Adopted Image Enhancement CNN 78
5.3.1 DnCNN Architecture 78
5.3.2 Double-Stage DnCNN-based Image Enhancement Configuration 79
5.4 Performance Assessment 80
5.4.1 Test Dataset 80

5.4.2 Single-Stage Image Enhancement Configuration Performance Assessment 81
5.4.3 Double-Stage Image Enhancement Configuration Performance Assessment 83
6. Conclusions and Future Work 89
6.1 Summary and Conclusions 89
6.2 Future work 90
Appendix A 93
Appendix B 105
Appendix C 108
Appendix D 114
References 116

List of Figures

Figure 1 – (a) Example of commercial fishing [4]; (b) Greenpeace activists protesting [5]. 1

Figure 2 – (a) Global trends in the state of world marine fish stocks since 1974 [6]; (b) Total fish catches in millions of tons reported to the UN Food and Agriculture Organization [7]. 2

Figure 3 – (a) Biological neuron [15]; (b) artificial neuron [15]. 7

Figure 4 – Non-linearity functions used in neural networks. (a) Sigmoid; (b) Hyperbolic Tangent (TanH); (c) Rectified Linear Unit (ReLu). 7

Figure 5 – Artificial Neural Network [16] architecture. 7

Figure 6 –Example of a simple Convolutional Neural Network architecture for image classification. C: convolutional layer; P: pooling layer; FC: fully connected layer. 9

Figure 7 – (a) input image; (b) 7 different activation maps obtained from 7 different filters in the first convolution layer [17]. 10

Figure 8 – A *single* filter convolving a given input image. Input image has dimensions of 5x5x3 (implying $W_1 = H_1 = 5$ and $D_1=3$). Filter has dimensions of 3x3x3 (implying a filter depth F = 3). This particular convolution is accomplished using a stride S= 2 and padding P = 1. The convolutional results in an activation map with W_2xH_2x1 dimensions (depth 1), where $W_2 = 3$ and $H_2 = 3$ [15]. 10

Figure 9 – Features visualization by using deconvolutional layers [18] at (a) lower layers; (b) medium layers; (c) higher layers [19]. 11

Figure 10 – Max-pooling operation [15]. 11

Figure 11 – A Convolution Neural Network targeting handwritten digit recognition [23]. Filters with stride S = 1 and zero-padding P = 0. 12

Figure 12 – Basic JPEG coding architecture [24]. 13

Figure 13 – Left: DCT bidimensional basis functions [25]; Middle: DCT coefficients [24]; Right: decoded DCT coefficients after quantization [24]. 14

Figure 14 - Basic JPEG 2000 coding architecture [34] 17

Figure 15 – Example of dyadic decomposition [36]. 18

Figure 16 – Basic Daala intra encoding architecture [49]. 19

Figure 17 – Left: original image; Right: image after pre-filter [41]. 20

Figure 18 – Left: image with ringing artifacts; Right: left image after applying the directional deringing filter [55]. 20

Figure 19 – Basic VP9 intra encoding architecture [38]. 21

Figure 20 – Left: prediction splitting; Right: Transform splitting (solid lines represent superblock) [59]. 22

Figure 21 – ICIP 2016 image test set. First row, from left to right: *honolulu, p08, p26* (1280x800); Second row, from left to right: *bike, cafe, woman* (1152x800). 24

Figure 22 – Specific scenario image, inside a fishing vessel: *boat* (5120x2944)..... 24

Figure 23 – Vessel Monitoring System..... 29

Figure 24 - Monicap Bluebox [76]. 30

Figure 25 – Example of VMS equipment [78]; Left: VMS unit; Right: Human to Machine Interface (HMI) terminal [79]. 31

Figure 26 - VMS software example [80]. 31

Figure 27 – VMS high level architecture. 33

Figure 28 - Monicap's architecture. 33

Figure 29 - Short Burst Data's architecture [95]. 35

Figure 30 – System operation detailed flowchart. 40

Figure 31 – (a) and (b) *Noruega* fishing vessel; (c) Trawling. 44

Figure 32 – (a) Camera installation; (b) Boat area seen by the camera. 44

Figure 33 – Two video examples: one without relevant segments/events and another with a single relevant segment/event. 44

Figure 34 – Video frames with no fishing activities: (a) Crew just standing on the deck; (b) Crew sweeping the deck; (c) Placement of the net on the floor..... 45

Figure 35 – Stages of throwing the fishing net: (a) nothing particular is happening; (b) the green door allowing the fishing net to go into the sea is opened; (c) the net starts to be slowly thrown into the sea; (d) the net is halfway into the sea; (e) the net is still visible; (f) the net stops being visible as fully under water. 45

Figure 36 – Stages of pulling the fishing net: Two sequences with different endings are represented: (a)-(b)-(c)-(d)-(e)-(f), for the situation when a large amount of fish is caught and (a)-(b)-(c)-(g)-(h)-(i), for the situation when a small amount of fish is caught. The fishing net: (a) is still not visible; (b) starts to be visible; (c) is visibly pulled; (d) stops being pulled and is full of (visible) fish; (e) is emptied and fish put on the floor; (d) is not visible and some fish is put back into the sea. In case of a small amount of caught fish, the fishing net: (g) stops being pulled and has a small amount of fish; (h) is being opened to take out the fish; (i) is taken to the side of the vessel. 46

Figure 37 – Fishing Activity Detection and Classification architecture. 48

Figure 38 – Temporal filtering example for $Sliding\_window\_size = 7/framerate$..... 50

Figure 39 – Event classification on the fly: as more filtered frame-level event probabilities are received the *Event Detection Ratio* (*EDR*) is computed for a specific time interval: (a) $EDR = 0.0$; (b) $EDR = 0.16$; (c) $EDR = 0.76$; (d) $EDR = 0.96$; (e) $EDR = 0.75$; (f) $EDR = 0.02$..... 51

Figure 40 – Overlapping of the input and output of the event classification module, for $event\_threshold = 0.75$. The blue line represents the filtered frame-level event probabilities while the dashed, black line, represents the output of the event classification module, where are clear the starting and ending event moments. 51

Figure 41 – AlexNet architecture: C means convolutional layer; P means pooling layer; N means local response normalization layer; FC means fully connected layer. 53

Figure 42 – ROC curves examples [131]: (a) ROC curves of classifiers with different accuracies; (b) discrimination threshold variation within a ROC curve. 59

Figure 43 – Creation of segments and approximation of boundaries for ground-truth. (a) before; (b) after. 60

Figure 44 – Two examples of the several steps required to compute the error boundaries, for a given ground-truth relevant event. The blue line is the output of the event classification module, this means the detected events; the dashed black line regards the ground-truth events. (a) and (d) Before procedure. (b) and (e) After true positive filtering. (c) and (f) Boundary errors computation based on the true positive detected events boundaries. 63

Figure 45 – ROC curves for different *sliding_window_width* parameter values..... 64

Figure 46 – Smoothed frame-level relevant event probability for some relevant test videos. 65

Figure 47 – Relevant test video results: (a) smoothed frame-level relevant event probability; (b) frame extracted from a problematic moment. 65

Figure 48 – Relevant test video results: (a) smoothed frame-level relevant event probability; (b), (c) and (d) frames extracted from problematic moments..... 66

Figure 49 – Non-relevant testing videos with worse results: (a) and (c) output of the temporal smoothing module; (b) and (d) frames extracted from each of the (somewhat) problematic non-relevant testing videos. 67

Figure 50 – ROC curves for different *event_time_window* values, by varying the *event_threshold* parameter. 68

Figure 51 – Image degradation process followed by a restoration/enhancement step. 75

Figure 52 – (a) A residual block [145]; a skip connection is added, therefore allowing the learning of only the residual between two or more layers, instead of learning a new mapping function from one layer to the next [143]; (b) A fully convolutional network [143]: an activation/pixel at the output, 'sees' only a portion of its input, the so-called receptive field; the bigger the receptive field, the better. 76

Figure 53 – DnCNN architecture [150]. 79

Figure 54 –Double stage image enhancement configuration. SF means scaling factor. 80

Figure 55 – Example images from each sub-dataset; (a) landscapes; (b) faces; (c) fishing boats. 81

Figure 56 – JPEG on Steroids decoded image after compression with a quality factor of 6 and a spatial resolution of 320x180; luminance only (a) before enhancement; (b) after deblocking only enhancement. 82

Figure 57 – Image with spatial resolution of 106x60 up-sampled to a spatial resolution of 318x180 (scaling factor of 3): (a) before enhancement; (b) after super-resolution only enhancement. 83

Figure 58 – Scaling factor optimization using RD performance. Each point in each curve corresponds to a particular quality factor. (a) PSNR; (b) SSIM. 83

Figure 59 – RD performance for four different enhancement approaches for a scaling factor of 2 and varying quality factor. The left-most point was obtained using a quality factor of 3; the right-most point was obtained using a quality factor of 61. 85

Figure 60 – RD performance for four different enhancement approaches for a scaling factor of 4 and varying quality factor. The left-most point was obtained using a quality factor of 5; the right-most point was obtained using a quality factor of 96. 85

Figure 61 – Results obtained for an image (of a fishing event) with a spatial resolution of 640x360 processed with a scaling factor of 3 and encoded with a bitrate of 0.062/1.76kB: (a) original; (b) compressed with Q = 15; (c) enhanced with conventional methods (spp and bicubic); (d) DnCNN image enhancement. 87

Figure 62 – Results obtained for an image with a spatial resolution of 640x360 processed with a scaling factor of 3 and encoded with a bitrate of 0.083/2.32kB. (a) original; (b) compressed with Q = 18; (c) enhanced with conventional methods (spp and bicubic); (d) enhanced with DnCNN. 87

Figure 63 – $PSNR_Y$ metric comparison for *bike* (left) and *boat* (right) compression – medium spatial resolution set. 93

Figure 64 – $PSNR_Y$ metric comparison for *cafe* (left) and *Honolulu* (right) compression – medium spatial resolution set. 94

Figure 65 – $PSNR_Y$ metric comparison for *p08* (left) and *p26* (right) compression – medium spatial resolution set. 94

Figure 66 – $PSNR_Y$ metric comparison for *woman* compression (medium spatial resolution set). 94

Figure 67 – $SSIM_Y$ metric comparison for *bike* (left) and *boat* (right) compression – medium spatial resolution set. 95

Figure 68 – $SSIM_Y$ metric comparison for *cafe* (left) and *honolulu* (right) compression – medium spatial resolution set. 95

Figure 69 – $SSIM_Y$ metric comparison for *p08* (left) and *p26* (right) compression – medium spatial resolution set. 95

Figure 70 – $SSIM_Y$ metric comparison for *woman* compression – medium spatial resolution set. 96

Figure 71 – $PSNR_Y$ results comparison for *bike (left)* and *boat (right)* compression – low spatial resolution set. 96

Figure 72 – $PSNR_Y$ results comparison for *cafe* (left) and *honolulu* (right) compression – low spatial resolution set. 96

Figure 73 – $PSNR_Y$ results comparison for *p08* (left) and *p26* (right) compression – low spatial resolution set. 97

Figure 74 – $PSNR_Y$ results comparison for *woman* compression – low spatial resolution set. 97

Figure 75 – $SSIM_Y$ results comparison for *bike* (left) and *boat* (right) compression – low spatial resolution set. 97

Figure 76 – $SSIM_Y$ results comparison for *cafe* (left) and *honolulu* (right) compression – low spatial resolution set. 98

Figure 77 – $SSIM_Y$ results comparison for *p08* (left) and *p26* (right) compression – low spatial resolution set. 98

Figure 78 – $SSIM_Y$ results comparison for *woman* compression – low spatial resolution set. 98

Figure 79 – $PSNR_Y$ results comparison for *Daala* (left) and *Guetzli* (right) – medium spatial resolution set. 99

Figure 80 – $PSNR_Y$ results comparison for *JPEG 2000* (left) and *JPEG 2000 Layers* (right) – medium spatial resolution set. 99

Figure 81 – $PSNR_Y$ results comparison for *JPEG Standard* (left) and *JPEG on Steroids*. 99

Figure 82 – $PSNR_Y$ results comparison for *Thor* (left) and *VP9* (right) – medium spatial resolution set. 100

Figure 83 – $SSIM_Y$ results comparison for *Daala* (left) and *Guetzli* (right) – medium spatial resolution set. 100

Figure 84 – $SSIM_Y$ results comparison for *JPEG 2000* (left) and *JPEG 2000 Layers* (right) – medium spatial resolution set. 100

Figure 85 – $SSIM_Y$ results comparison for *JPEG Standard* (left) and *JPEG on Steroids* (right) – medium spatial resolution set. 101

Figure 86 – $SSIM_Y$ results comparison for *Thor* (left) and *VP9* (right) – medium spatial resolution set. 101

Figure 87 – $PSNR_Y$ results comparison for *Daala* (left) and *Guetzli* (right) – low spatial resolution set. 101

Figure 88 – $PSNR_Y$ results comparison for *JPEG 2000* (left) and *JPEG 2000 Layers* (right) – low spatial resolution set. 102

Figure 89 – $PSNR_Y$ results comparison for *JPEG Standard* (left) and *JPEG on Steroids* (right) – low spatial resolution set. 102

Figure 90 – $PSNR_Y$ results comparison for *Thor* (left) and *VP9* (right) – low spatial resolution set. 102

Figure 91 – $SSIM_Y$ results comparison for *Daala* (left) and *Guetzli* (right) – low spatial resolution set. 103

Figure 92 – $PSNR_Y$ results comparison for *JPEG 2000* (left) and *JPEG 2000 Layers* (right) – low spatial resolution set. 103

Figure 93 – $SSIM_Y$ results comparison for *JPEG Standard* (left) and *JPEG on Steroids* (right) – low spatial resolution set. 103

Figure 94 – $SSIM_Y$ results comparison for *Thor* (left) and *VP9* (right) – low spatial resolution set. 104

Figure 95 – Compressing times comparison for *bike* (left) and *boat* (right). 105

Figure 96 – Compressing times comparison for *cafe* (left) and *honolulu* (right). 105

Figure 97 – Compressing times comparison for *p08* (left) and *p26* (right). 106

Figure 98 – Compressing times comparison for *woman*. 106

Figure 99 – Compressing times comparison for *bike* (left) and *boat* (right). 106

Figure 100 – Compressing times comparison for *cafe* (left) and *honolulu* (right). 107

Figure 101 – Compressing times comparison for *p08* (left) and *p26* (right). 107

Figure 102 – Compressing times comparison for *woman*. 107

Figure 103 – Architecture of the proof-of-concept system demonstrated to *Xsealence*. The blocks in orange were developed by the author of this Thesis. 108

Figure 104 – (a) Raspberry Pi Model B; (b) Raspberry Pi Camera. 109

Figure 105 – (a) live chart with relevant event probability ,the chart has a width of 300 seconds (5 minutes) and is updated from right to left; (b) image currently processed by AlexNet; (c) state of the event detection algorithm. 111

Figure 106 – JPEG 2000 compressed image received at the Control Center. 112

Figure 107 – (a) Setup; (b) Setup zoomed in. 112

Figure 108 – Performance comparison between 20 and 1 frames per second. (a) Frame-level; (b) Event-level. 113

Figure 109 – Landscape datasets images. Spatial resolutions for all image is 576x400. 114

Figure 110 – Faces dataset images. Spatial resolution are: (a) 480x 480; (b) 400x576; (c) 576x400 (d) 400x576; (e) 576x400. 115

List of tables

Table 1 – Iridium transceivers using SBD [95]. 35

Table 2 – Data plans for various Iridium services. 36

Table 3 – Video characteristics table excerpt. 48

Table 4 – Overall video dataset statistics. 48

Table 5 – CNN benchmarking [116]. 53

Table 6 – Event boundary error metrics results. 69

Table 7 – Average values of PSNR, for each codec with and without pos-processing and QP. 71

Table 8 – Cost comparison between the three codecs; JPEG codecs are pos-processed. 72

Table 9 – Compressed images for each codec and QP. 72

Table 10 – PSNR and SSIM comparison for a JPEG on Steroids decoded image before and after only deblocking. The quality factor for JPEG coding ranges from 0 to 100; higher is better. 81

Table 11 – PSNR and SSIM comparison for bicubic interpolation, Lanczos [158] interpolation with a kernel size of 3 interpolation as implemented in matlab, and DnCNN super-resolution enhancement. 82

Table 12 – Optimal scaling factors for each bitrate interval by presenting the upper and lower bounds of the intervals where a given scaling factor is optimal (in bitrate (bpp) and storage file size (kilobytes, kB)) 84

Table 13 – Comparison between no enhancing (besides bicubic interpolation), conventional enhancement and DnCNN enhancement. 86

Table 14 – PSNR and SSIM comparison for a JPEG on Steroids decoded image before and after only deblocking. The quality factor for JPEG coding ranges from 0 to 100; higher is better. 114

Table 15 – PSNR and SSIM comparison for bicubic interpolation, Lanczos [165] interpolation with a kernel size of 3 interpolation as implemented in matlab, and DnCNN super-resolution enhancement. 114

Table 16 – PSNR and SSIM comparison for a JPEG on Steroids decoded image before and after only deblocking. The quality factor for JPEG coding ranges from 0 to 100; higher is better. 115

Table 17 – PSNR and SSIM comparison for bicubic interpolation, Lanczos [165] interpolation with a kernel size of 3 interpolation as implemented in matlab, and DnCNN super-resolution enhancement. 115

List of Acronyms

AUC	Area Under the Curve
CNN	Convolutional Neural Network
CPU	Central Processing Unit
EDR	Event Detection Ratio
EMC	Electromagnetic Compatibility
EU	European Union
FAO	Food and Agriculture Organization
FC	
FDMA	Frequency Division Multiplexing Access
FN	False Negative
FP	False Positive
GIS	Geographic Information System
GLONASS	*Globalnaya navigatsionnaya sputnikovaya*
GNSS	Global Navigation Satellite System
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HMI	Human-Machine Interface
HR	High Resolution
IPMA	Instituto Português do Mar e da Atmosfera (*Portuguese Institute for Sea and Atmosphere*)
ISU	Iridium Subscriber Unit
IUU	Illegal, Unreported and Unregulated
LEO	Low Earth Orbit
LR	Low Resolution
M2M	Machine to Machine
MCS	Monitor Control and Surveillance
MO	Mobile Originated
MO	Mobile Originated
NN	Neural Networks
PSNR	Peak Signal-to-Noise Ratio
QF	Quality Factor
QP	Quality Parameter
RD	Rate-Distortion
ROC	Receiver Operating Characteristic

SBC	Single Board Computer
SBD	Short Burst Data
SF	Scaling Factor
SMS	Short Message Service
SSIM	Structural Similarity
TDMA	Time Division Multiplexing Access
TN	True Negative
TP	True Positive
VMS	Vessel Monitoring System

Chapter 1

1. Introduction

This chapter intends to introduce the topic of overfishing and its consequences, thus justifying the need for monitoring fishing vessels activities. It will also detail the objectives of this Thesis and how it proposes to improve existing fish monitoring methods, notably through visual data acquisition and processing. At the end of this chapter, the structure of this MSc Thesis is outlined.

1.1 Context and Motivation

For centuries, fishing has taken a part on many people's lives that relied on this source of food to survive. If at the beginning our seas and oceans were thought as a limitless source of food, today the reality is much different.

In the mid of the last century, international efforts were made to overcome protein-rich food scarcity, which led to a continuously increase of fishing capacity [1]. As commercial fleets aggressively scoured the oceans with unregulated ever growing, sophisticated fishing methods, with nothing but profit in mind, fishing came to a point where it started to be more and more unsustainable. This has brought serious consequences, not only for the balance of life in the oceans but also for the economic and social security of coastal communities, whose life depends on fishing. For example, in 1992, in Newfoundland, Canada, a cod fishery ingeniously thought as inexhaustible collapsed, causing 40,000 people to lose their jobs [2]. In the Pacific Nations, where the general population gets 50 to 90 percent of their protein from fish and for millions for which fishing is the only source of income, fish scarcity has much more profound implications. If this problem is not addressed rapidly, serious consequences will be faced in the future (see Figure 1).

(a) (b)

Figure 1 – (a) Example of commercial fishing [3]; (b) Greenpeace activists protesting [4].

Faced with the collapse of fish stocks, commercial vessels kept expanding deeper into the ocean, discovering new fish stocks to exploit. By 1996, where fishing peaked at 86.4 million tons [5], few were the fisheries left to be discovered, which led to a decline in catches – not due to countries fishing less,

but because fisheries had been exhausted, one after the other [6]. In 2013, total fish catches yielded 80.9 million tons, most of which caught on Northwest Pacific (27% of global fish catch), followed by the Western Central Pacific (15%), the Southeast Pacific (11%), and finally the Northeast Atlantic (10 percent) [5]. The continuation of overfishing led to an expected continuous depletion of fish stocks (see Figure 2). By the end of 2013, the number of fish stocks fished at biologically unsustainable levels reached 31.4%. In Portugal, the trend remains – catches exceed in 37% the scientific recommendations (on average), making it one of the European countries disrespecting scientific advice the most [7].

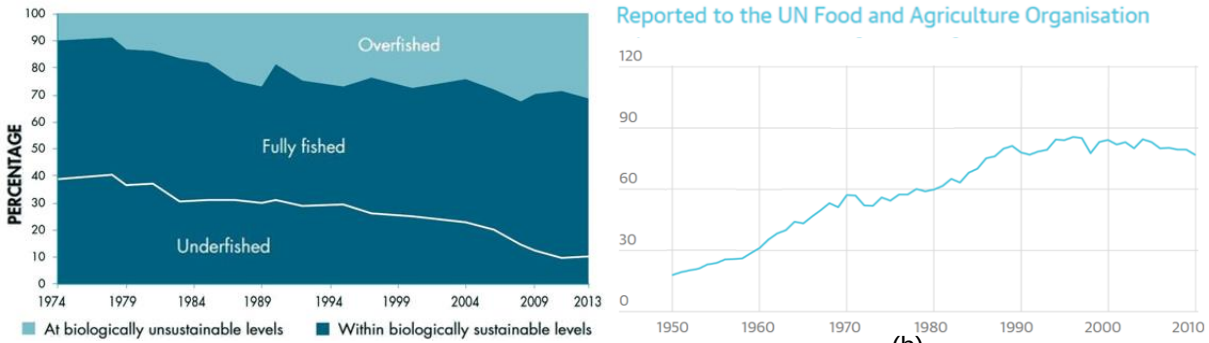


Figure 2 – (a) Global trends in the state of world marine fish stocks since 1974 [5]; (b) Total fish catches in millions of tons reported to the UN Food and Agriculture Organization [6].

To stop overfishing and its consequences, governments started to enforce fishing-policy rules, aiming to collect the necessary data for managing fishing opportunities, thus being able to ensure that only a certain quantity of fish is caught. The rules must be applied to all fishing vessels (above a certain length) – uncompliant vessels should be heavily sanctioned [8].

As technology evolves, new ways of enforcing such fishing-policy rules are developed. In fact, a Portuguese company name *Xsealence* was responsible for developing the first Vessel Monitoring System (VMS) back in 1988 [9]. This system, whose operation and architecture will be covered later in this work, is based on a “black box” placed on the fishing vessel that is able to acquire the boat position and other statistics and transmitting them to a Control Center. Currently, *Xsealence* is studying the possibility of adding a video camera to its “black box”, to provide the Control Center with the ability to monitor vessels through the acquisition and processing of visual data.

There are, however, complex problems inherent to such a visual-based solution. For instance, while mobile communications are getting increasingly faster and cheaper, in the high-seas the mobile network coverage is virtually nonexistent. As such, the only solution is to use satellite communications. However, the available channel bandwidth can be very narrow and expensive – something as simple as an e-mail must be sent through a specific e-mail client in order to be duly handled, e.g. compressed. This poses an obvious problem to video transmission, as even very low bitrate video would completely saturate the available channel, at prohibitively high costs. Nevertheless, there should be a way to transmit at least a cleverly selected set of images, providing that the user is willing to pay a reasonable extra cost.

1.2 Objectives and Thesis Structure

The main objective of this MSc Thesis is to overcome the problems described in the previous section by developing a fully automatic solution with the purpose of assessing if a fishing activity is taking place in a given vessel at any given moment, through the processing of appropriate visual data. Such a visual processing solution would enable a careful selection of the images to be sent through the satellite channel, among other benefits, thus also limiting the associated cost. Moreover, this work attempts to further reduce the amount of data to be transmitted through the typically used low-rate satellite channels by analyzing royalty free still image codecs and their compression capabilities, as well as introducing a way of enhancing the heavily compressed images received at the Control Center.

To achieve these goals, this Thesis will adopt twice a deep-learning approach. First, to solve the problem of detecting fishing activities, a well-known Convolutional Neural Network architecture designed for image classification is used. Then, to solve the problem of enhancing heavily compressed images, a different Convolutional Neural Network architecture suitable for image enhancement/restoration is also used.

The structure of this work is as follows: in Chapter 2, a review on relevant background technologies will be presented, notably basic concepts and tools related to machine/deep learning classification and image coding. Then, Chapter 3 describes the scenario where the proposed solution shall be implemented, as well as the application requirements. In Chapter, 4 the actual fishing detection solution is proposed, and its performance assessed under relevant and meaningful conditions. Also, the solution is physically implemented as a proof-of-concept in an environment mimicking a fishing activity scenario. Afterwards, in Chapter 5, a solution is proposed with the purpose of enhancing the heavily compressed images available at the Control Center with the target to improve the final experience. Finally, Chapter 6 summarizes the work done in this Thesis, presents some conclusions and provides some suggestions for future work.

Chapter 2

2. Brief Review on Relevant Background Technologies

The goal of this chapter is to introduce and address key concepts for the context of this work. With this purpose in mind, it starts by providing a background on deep learning, which turned out to be essential in the development of this MSc Thesis. Then, a review on several royalty free still image codecs will be given, followed by a performance assessment of each of them with the goal of comparing both RD-performance and computational complexity.

2.1 Deep Learning: Background and Main Concepts

The objective of this section is to introduce some essential deep learning background and concepts that will be used throughout this Thesis. This section begins with some motivation, then some background on Neural Networks and, finally, addresses Convolutional Neural Networks, which is currently one of the most popular types of Neural Networks used to address many computer vision and image processing problems.

2.1.1 Motivation

The availability of visual data has grown at a dramatic rate in the past decades. This has allowed images and videos to become ubiquitous in the Internet and new systems and applications to take advantage of this abundancy of visual data. Several high-level computer vision tasks such as image recognition and object detection for example, have been studied and improved for many years, but the many inherent different and difficult conditions and challenges, notably associated to color and texture variations, background clutter and occlusion [10], proved hard to overcome and thus reach high performances. Overcoming some of these difficulties would imply making large performance leaps for applications such as automated driving, medicine, and others.

During the 2000's, object recognition and other similar problems were solved by using hand-crafted feature extractors such as Scale Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG) coupled with machine learning classifiers. For example, features were extracted and then compactly represented using the Bag of Visual Words model, which is essentially a vector of occurrence of local image features [11] for an image. Finally, machine learning algorithms such as Support Vector Machines (SVM) were used for classification [12]. While this was a rather successful approach in its early years for image recognition, the performance of these algorithms was only as good as the discrimination capabilities of the hand-crafted features, which quickly became rather complex [12]. This increasingly higher complexity made it difficult to design better features [12], which led to rather minor performance improvements over the following years.

More recently, machine learning algorithms able to automatically learn the features appropriate for a specific task have come back with reinforced strength. Most of these methods are based on what is known today as *deep learning*. Essentially, deep learning methods in the field of computer vision work by using multiple learnable feature associated to multiple extraction layers. With the help of powerful GPUs and large image datasets, efforts on improving such methods eventually culminated in the winning of a Convolutional Neural Network solution in the image classification category of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 by a large margin. This performance leap started a revolution in the computer vision domain, with a large variety of applications, notably image classification, object detection, object localization, pose estimation, etc., resorting to deep learning methods. Eventually, deep learning found its way also in low-level computer vision and image processing, thus resulting in a very significant performance improvement in areas like image enhancement and restoration.

2.1.2 Background

Before diving into the deep learning methods typically used in computer vision tasks, the basic foundations of Neural Networks (NN) are briefly reviewed. Neural networks are a class of machine learning techniques inspired by the way biological nervous systems, such as the brain, work. A Neural Network exploits an architecture with a large number of highly interconnected neurons [13] that work together to solve specific problems, inspired by the way neurons interact and process information in the human brain.

The neuron is the basic operation unit of the brain. Each neuron receives input signals from the dendrites (see Figure 3, (a)) and produces an output signal along its only axon (a physical structure carrying signals away from the cell body toward the post-synaptic cell) that branches out and thus passes the signal that is transmitted via synapses (which are small gaps between axon terminals and the dendrites from a following neuron) to other neurons. The connections between neurons are based on the synaptic strength, which ultimately carries the 'importance' that the neuron gives to a signal arriving from a previous neuron. The dendrites carry the signals to the neuron body where they are all processed together. If the result is above a given value, the neuron fires, or *activates*, and passes the resulting signal to the following neurons [14].

The idea behind an artificial neuron was inspired by the biological model described above. An artificial neuron is connected to several other artificial neurons via connections. These connections have *weights* (mimicking the synapse strengths) associated to them which are learnable parameters that control the 'importance' that a neuron gives to each connection. Analogously to the real model, incoming signals are processed according to

$$O = f\left(\sum_{i=0}^{n} w_i x_i + b\right) \tag{2.1}$$

where O refers to the signal transmitted to the following neurons, x_i refers to the signal arriving from the connection i, w_i to the weight of each connection and n to the number of incoming signals/connections;

b refers to a *bias* learnable parameter, which essentially adds a constant value to the weighted output signal. Finally f refers to the activation function, also called non-linearity function.

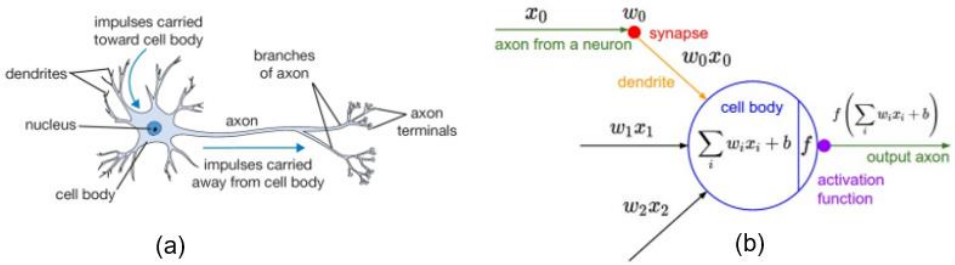


Figure 3 – (a) Biological neuron [14]; (b) artificial neuron [14].

Analogously to what happens in a biological neuron, the activation/non-linearity function determines the strength of the outcome for a specific set of incoming connections. The purpose of these functions is to introduce non-linearities into the neuron model by performing a certain pre-defined mathematical operation over its input [14]. Commonly used non-linearity functions are the Sigmoid, Hyperbolic Tangent (TanH) and the Rectified Linear Unit (ReLu) functions [15], illustrated in Figure 4. Without these functions, the output of the neural model would be just a linear function of the input; using a non-linearity function ensures that the model is more expressive, this means more capable of learning complex functional mappings, than a linear model [12].

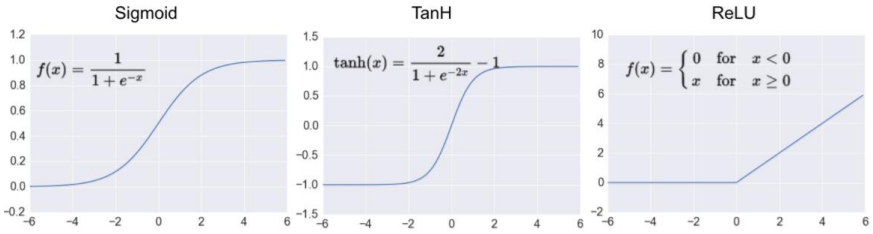


Figure 4 – Non-linearity functions used in neural networks. (a) Sigmoid; (b) Hyperbolic Tangent (TanH); (c) Rectified Linear Unit (ReLu).

Architecture

Regular Neural Networks behave essentially as directed graphs where nodes are neurons. These networks are composed by several layers and, the more layers a Neural Network has, the deeper the network is. The neurons at each layer can be fully connected to the following layer, as depicted in Figure 5, i.e. each neuron of a layer is connected to every neuron in the next layer.

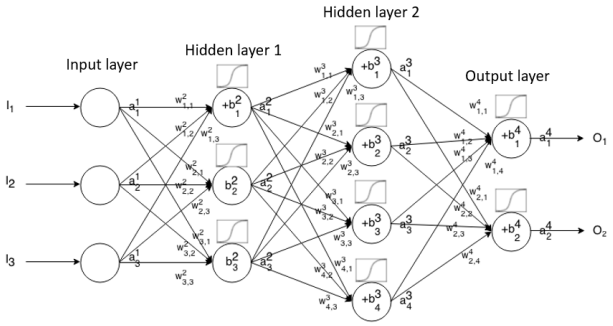


Figure 5 – Artificial Neural Network [15] architecture.

The architecture of a Neural Network is as follows:

- **Input layer** – Receives data/features suitable for the problem to be solved (not necessarily image-related) and passes this information to the next layer;
- **Hidden layer(s)** – Hidden layers combine the information received from the previous layers. Usually, when the number of hidden layers increase, the Neural Network becomes more powerful. However, the higher is the number of hidden layers, the hardest it is to train these networks.
- **Output layer** – Receives the signals computed by the hidden layers. It is usually composed by several neurons, and each neuron has some associated meaning, e.g. for classification problems each output neuron corresponds to a certain class. The neuron outputting the highest value indicates the class which is predicted by the Neural Network as the most probable, for some input data.

Training

The training of a Neural Network is performed by the so-called *backpropagation algorithm* using a set of training data, which includes a set of data examples and the corresponding desired outputs, this means the labels. This algorithm uses the training data to feed-forward it throughout the network. After one piece of the training data is processed by the network, the output result is likely to be far from the desired ground truth value, especially at the beginning of the training procedure. To measure the error between the output and the ground-truth, a loss function is used. This error/loss is computed at the output and then back-propagated through the entire network such that the weights and biases are updated according to this error, targeting the reduction of the loss error.

A non-image related example use case for Neural Networks would be predicting the price of a house given some historical information, e.g., area, age, number of bedrooms and bathrooms and location, and the corresponding price (the ground truth). The Neural Network would then learn a model (this means the network weights and biases) using the training data at its input and the corresponding real prices at its output; after, the trained network will be able to predict the prices for house given their characteristics. The better the training (both in terms of algorithms and training data), the more accurate the predictions.

2.1.3 Convolutional Neural Networks

This section focus on how Convolutional Neural Networks (CNNs) are typically used in the field of computer vision in a way that is relevant for the purposes of this Thesis.

Although there are some practical uses for pure fully connected Neural Networks, such as the one described in Section 2.2.2, for a wide range of image analysis tasks, they are not applied, especially due to the large number of weights that needs to be estimated. For example, for a 200 x 200 spatial resolution image [12], the first layer would have 40,000 neurons (one for each pixel). If followed by a hidden layer of 40,000 neurons, this would result in 800 million connection weights/parameters between neurons. With more layers, the number of parameters would escalate dramatically, and thus this type of networks cannot be straightforwardly applied due to its complexity.

To overcome these limitations, convolutional layers, which are comprised by a set of filters, are used. This radically changes the architecture of the Neural network and adds an extremely important characteristic: feature learning, i.e. the network learns filters that had to be hand-crafted in traditional algorithms.

Convolutional Neural Networks are thus a special kind of multi-layer Neural Networks as these networks still keep the forwarding processing and training algorithms as in the regular Neural Networks. The CNN processes the image data at the input (as will be detailed next) layer by layer with a set of filters until reaching the output layer. The error between the output and the ground-truth is computed (using the loss function) and back-propagated throughout the network while updating the model this means the weights and bias.

As regular Neural Networks, a CNN also consists of an input and an output layer, as well as multiple hidden layers. The hidden CNN layers, however, can vary significantly, depending on the target application. For example, CNN architectures suitable for solving imaging problems such as denoising, super-resolution, compression artifact removal and image enhancement in general, usually use convolutional layers only. However, CNN architectures for image classification related tasks tend to use additional types of layers, such as pooling and fully connected layers. For such tasks, these layers are typically divided into two main parts: one responsible for feature extraction and another responsible for the classification itself (based on the extracted features). Figure 6 depicts a simple and typical example of this type of CNN architectures. Each of these types of layers will be explained next for an image classification context.

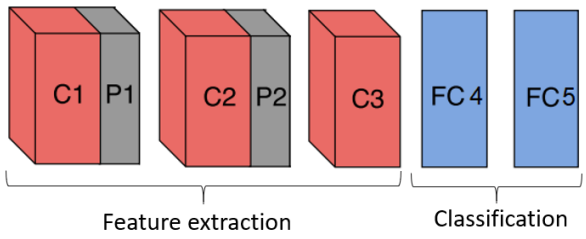


Figure 6 –Example of a simple Convolutional Neural Network architecture for image classification. C: convolutional layer; P: pooling layer; FC: fully connected layer.

Convolutional layers

Convolutional layers are made by several filters that convolve the typically 3D data matrix at its input, this means a matrix with a depth/dimensionally greater than one. For instance, RGB images have 3 channels, therefore implying a depth of 3. A convolution takes place when a filter with some specified size slides over the input matrix (multiplying the filter weights by the values of the input matrix). Each multiplication results in an activation value. When a filter finishes convolving the matrix, the resulting activation values constitute an activation map, which contains the features extracted by that same filter (see Figure 7).

Figure 7 – (a) input image; (b) 7 different activation maps obtained from 7 different filters in the first convolution layer [16].

A convolutional layer can be defined by a set of hyper-parameters: the number of filters, *K*, the depth of the filter, *F*, corresponding to the number of filter channels, the stride of the filter, *S*, corresponding to the size of the step that a filter takes when convolving its input matrix, and the amount of zero-padding, *P*, corresponding to the number of rows/columns with zero values that are added around the input matrix (see Figure 8). The steps taken in a convolutional layer are as follows [14]:

1. A convolutional layer receives a 3D matrix with dimensions $\mathrm{W}_i \times \mathrm{H}_i \times \mathrm{D}_i$, where W_i is the width, H_i the height and D_i the depth of the matrix. For the first layer, this matrix corresponds to the input image to the network; for the remaining layers, it corresponds to activation maps from previous layers.
2. Then, each of the filters in the convolutional layer convolves throughout the 3D matrix, thus requiring that the matrix depth the filter depth match (see Figure 8). For each filter convolution, a $\mathrm{W}_{i+1} \times \mathrm{H}_{i+1} \times 1$ activation map (also called feature map) is produced. This activation map contains the features extracted by that same filter. This process is represented in Figure 8, where a single filter convolves a 3D matrix, producing an activation map. W_{i+1} and H_{i+1} can be computed as:

$$\mathrm{W}_{i+1} = \frac{\mathrm{W}_i - \mathrm{F} + 2\mathrm{P}}{\mathrm{S}} + 1 \qquad (2.2)$$

and

$$\mathrm{H}_{i+1} = \frac{\mathrm{H}_i - F + 2P}{\mathrm{S}} + 1 \qquad (2.3)$$

3. Stacking the activation maps together produces an activation volume of $\mathrm{W}_{i+1} \times \mathrm{H}_{i+1} \times \mathrm{D}_{i+1}$, where $\mathrm{D}_{i+1} = K$. This activation volume is then forwarded to the next layer of the network.

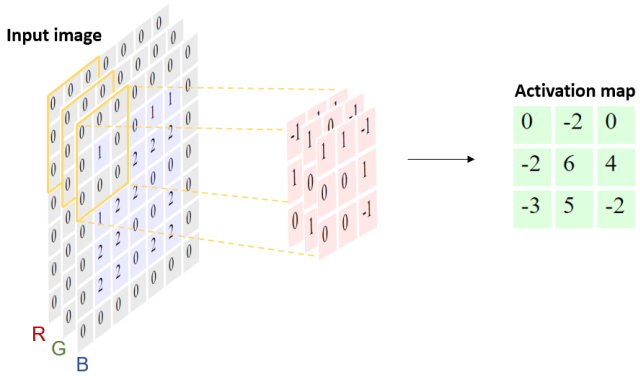


Figure 8 – A *single* filter convolving a given input image. Input image has dimensions of 5x5x3 (implying $W_1 = H_1 = 5$ and $D_1=3$). Filter has dimensions of 3x3x3 (implying a filter depth F = 3). This particular convolution is accomplished using a stride S= 2 and padding P = 1. The convolutional results in an activation map with $W_2 x H_2 x 1$ dimensions (depth 1), where $W_2 = 3$ and $H_2 = 3$ [14].

The filter parameters correspond to the weights of a neuron, since a convolution corresponds to the sum of the multiplications of the output of a previous layer (or the input RGB image for the first layer) by the weights. Thus, the training of a Convolutional Neural Network is made as for any other Neural Network.

This means that, at the beginning of the training process, filters are unable to extract important features but, as training data is being fed to the Neural Network, the weights are updated and converge to practically meaningful values, i.e. creating filters able to extract important features for a specific problem.

Convolutional layers are applied one after another (with some other layers in between), meaning that filters are applied consecutively on top of already extracted features. This causes the extracted features at each layer to have an increasingly higher level of abstraction, as depicted in Figure 9. For example, while the lower layers extract low-level features such as edges, higher layers extract higher-level features, such written characters, car wheels or even an average human face.

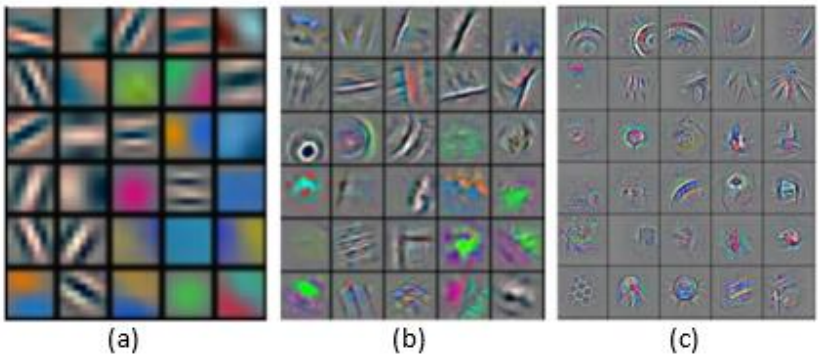


Figure 9 – Features visualization by using deconvolutional layers [17] at (a) lower layers; (b) medium layers; (c) higher layers [18].

After each convolutional layer, a non-linearity function is applied to each value in the activation map. In the context of CNNs, from the several available non-linearity functions, the Rectified Linear Unit has enjoyed the most success, since it has been shown to significantly accelerate the learning process [19], essential to train deep CNNs. Using this function on an activation map, essentially replaces negative values by zero.

Pooling layers

Pooling layers are usually inserted every few convolutional layers. The goal is to progressively reduce the amount of parameters in the network, thus resulting into a more compact model and also reducing the problem of overfitting [14], which refers to lack of capability of a model to be able to generalize well for data outside its training data. Pooling layers also help the network to be invariant to small spatial variations in the input image such as translations [12].

The most usual form of pooling is max-pooling; this method uses a filter with a dimension of 2x2, with a stride of 2 to selects the activation value with the highest value. In practice, the original feature map is subsampled by a factor of 2 in both directions, as shown in Figure 10.

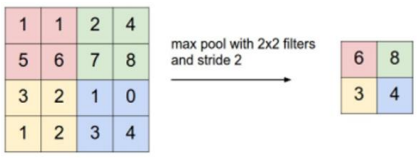


Figure 10 – Max-pooling operation [14].

Fully connected layers

In a fully connected layer, each neuron is connected to all neurons from previous and following layers, very similar to what is represented in Figure 5 (hence the term 'fully connected'). As previously described, the convolutional layers extract high-level features from the input image, i.e. several distinct activation maps. In image classification, it is up to the fully connected layers to receive these extracted features and classify the input image as belonging to a specific class, e.g., cat, dog, car, etc. The way the high-level features are fed to the first fully connected layer is by connecting each of the activation map values to each of the first fully connected layer neurons. The information is then combined and propagated forward and the last layer output is essentially a vector with different activation values, where each vector element corresponds to a class. This vector is passed through a Softmax function [20] that 'squashes' it in a way that the sum of all its elements is equal to 1. Thus, each element of the output vector contains the probability of the input image belonging to a specific class.

To help the reader visualize how the previously detailed layers are connected, a representation very similar to the LeNet CNN [21], which is a network developed for handwritten digit recognition, is presented in Figure 11.

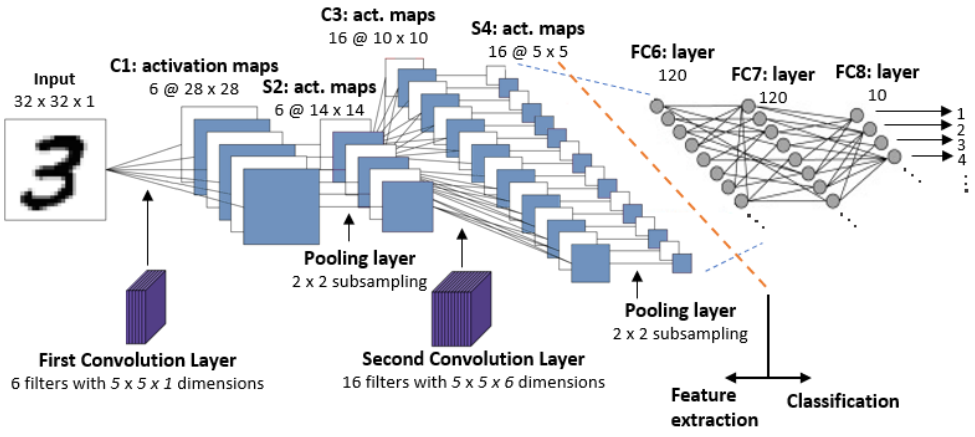


Figure 11 – A Convolution Neural Network targeting handwritten digit recognition [22]. Filters with stride S = 1 and zero-padding P = 0.

2.2 Still Image Codecs: a Brief Review

This section will briefly review the most relevant still images codec available, since these are the most suited for the context of this MSc Thesis. It will start with JPEG compliant codecs and then move on to video intra-coding solutions (e.g. VP9 Intra) which, currently, are those offering the best rate-distortion (RD) performance. It is important to understand that one same coding solution/standard may have many implementations with rather different performance, since some important encoding tools, e.g. quantization, are either non-normative or may be parameterized in more or less efficient ways. This is the reason why we are here reviewing codecs, this means implementations, and not algorithms. Afterwards, the performance of each of the presented codecs will be assessed.

2.2.1 JPEG Compliant Codecs

First introduced in 1992, the JPEG codec was developed by the Joint Photographic Experts Group (JPEG) [23]. Nowadays, is still the most widely used coding solution for lossy still imaging compression, although several coding solutions with better RD performance have been released since. Due to its popularity, many JPEG codecs have been developed targeting to improve its RD performance always in a JPEG compliant way, which is essential to benefit from the huge number of JPEG decoders available.

After a brief overview on the basic coding algorithm, this section provides a short description on the improvements adopted by some JPEG codecs to reach better RD performance. The associated encoders *tweak* the standard JPEG encoder, using techniques that have been developed throughout the years, attempting to reach a better perceptual image quality while maintaining the same file size/rate. As it is important to give the reader a general idea of the basic JPEG coding algorithm, a quick overview of the JPEG baseline sequential process (the most used JPEG coding solution) will be given first.

JPEG overview

The basic JPEG coding architecture is shown in Figure 12, followed by a brief explanation of each of the main modules.

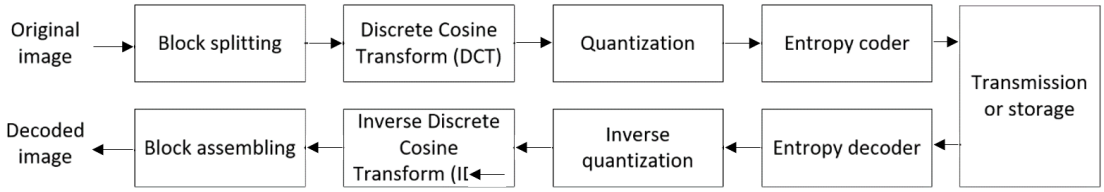
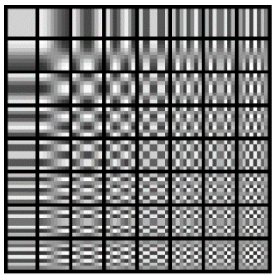


Figure 12 – Basic JPEG coding architecture [23].

- **Block splitting** – Each component of the original image, e.g. RGB or YCrCb, is sub-divided into 8x8 samples blocks.
- **Discrete Cosine Transform (DCT)** – The DCT transforms the block information from the spatial domain into the frequency domain while exploiting the spatial redundancy. To achieve this, DCT basis functions are used (see Figure 13, left). These functions represent from slower to rapid texture variations, which together are able to accurately represent the block information. The obtained transform coefficients represent how much each DCT basis function contributes for the information in the block. For example, a block only with vertical variations will have only non-zero DCT coefficients in the first row. A pure flat block will only have one non-null DCT coefficient. An example of a DCT coefficients matrix, before quantization, is given in Figure 13, middle.



$$\begin{bmatrix} -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \begin{bmatrix} -416 & -33 & -60 & 32 & 48 & -40 & 0 & 0 \\ 0 & -24 & -56 & 19 & 26 & 0 & 0 & 0 \\ -42 & 13 & 80 & -24 & -40 & 0 & 0 & 0 \\ -42 & 17 & 44 & -29 & 0 & 0 & 0 & 0 \\ 18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figure 13 – Left: DCT bidimensional basis functions [24]; Middle: DCT coefficients [23]; Right: decoded DCT coefficients after quantization [23].

- **Quantization** – After the transformation of the original block into DCT coefficients, quantization will take place to control the target quality and rate. This will effectively involve losing/discarding information. If transparent quality is targeted, it is of most importance to only discard information that is perceptually irrelevant. This process is based on a quantization matrix that may give more importance, this means quantize less, the lower frequency coefficients and less to the higher frequency ones (i.e., the quantization step is lower for the lower frequencies and vice-versa) to take into account the Human System Visual (HVS) sensitivity to each DCT coefficient. Since the higher frequency coefficients are typically quantized with higher quantization steps and natural images tend to be smooth, the chances are higher that they were set to zero or have lower quantized amplitudes. The resulting quantized DCT coefficients matrix is then passed to the entropy coder which should generate the final bitstream, taking into account the probability of each symbol to code. At the decoder, the DCT coefficients are restored by multiplying the received quantized coefficients by the quantization step matrix; naturally, the restored DCT coefficients are not equal to original DCT coefficients due to information lost in the quantization process, as shown in Figure 13, right.
- **Entropy coder** – This module takes advantage of the statistical redundancy in the symbols, e.g. some DCT coefficient quantized amplitudes are more likely than others. It performs a zig-zag serialization of the quantized coefficients, each represented by means of a run-length pair which define its and quantized amplitude, followed by Huffman coding [25].

2.2.1.1 JPEG on Steroids Codec

The JPEG on Steroids codec [26] was one of still image coding solutions proposed to Image Compression Gran Challenge organized in the context of the International Conference on Image Processing 2016 [27]. This was a competition where the participants should bring image coding solutions that were compared using several objective quality metrics and also subjective tests [28]. JPEG on Steroids competed under the label *JPEG (visual)*.

Objectives

The goal of the JPEG on Steroids codec is to code an image resorting to techniques that have been developed and applied throughout the years targeting a good RD performance while still being JPEG compliant.

Technical Approach

There are several techniques available to reduce the JPEG-compressed stream size, notably playing with the quantization. However, the JPEG on Steroids codec attempts to find a compromise between complexity and RD performance, thus avoiding to focus on highly complex techniques, or rather simplifying them:

- **Subsampling** – The HVS has less acuity for colors than for luminance. Subsampling is a technique that takes advantage of this fact by lowering the resolution of both chrominances. While this is a good strategy for low quality images, as quality increases, it may introduce noticeable distortion for higher bitrates;
- **Soft-Threshold Quantization** – Usually, JPEG encoders use quantization matrices that are based on the example quantization matrix provided in Annex K [29] of the JPEG standard. By choosing the right quantization matrix for a given content, the perceptual quality may increase substantially. JPEG on Steroids makes use of different optimized quantization matrixes, obtained from existing publications and, furthermore, attempts to quantize the DCT coefficients properly using a technique denoted as soft-threshold quantization, which works by minimizing the function

  $$J(I, I') = D(I, I') + \lambda R(I') \qquad (2.1)$$

  which evaluates the contribution of *each* DCT coefficient by means of a cost function *J* which combines the associated distortion and rate, and adjusts the quantization step accordingly. In (1.1), *I* is the original image, *I'* the decoded one, *D* a distortion measure, *R* the bitstream rate and λ the control parameter for a desired image quality. By doing this, the encoder analyses if it would be beneficial for a given DCT coefficient to be zeroed out at the block level (e.g. if a DCT coefficient is preceded by a set of a zeros and it precedes another set of zeros). While performing this process, the distortion may increase to benefit from a rate reduction [26].
- **Entropy coding** – Usually, JPEG encoders use the codewords alphabet that are specified in the JPEG standard, avoiding the additional complexity required to determine a better set of codewords. However, JPEG on Steroids uses an adaptive Huffman table (the alphabet has to be sent in the codestream) that allows a better adaptation to the symbol statistics for each image. By doing this, the performance for lower bpp conditions can be improved [26].
- **Progressive mode** – In the previous subsection, the JPEG baseline, sequential oding process was described. However, there are more operational modes available, one of which being the progressive mode which codes the image by performing multiple scans which should offer increasing quality: by performing several scans over the image, not only allows to skip over multiple empty blocks at once, it also enables the use of customized Huffman tables for each scan, therefore improving the entropy coding gain [26].

By using these techniques JPEG on Steroids is able to achieve a considerable higher rate-distortion performance while keeping complexity low [26].

2.2.1.2 Guetzli JPEG Codec

Guetzli is a new open-source perceptually-guided JPEG codec developed by Google and released in 2017 [30]. It is stated to be able to produce JPEG images with file sizes 35% smaller than currently available JPEG codecs.

Objectives

As all JPEG optimized codecs, Guetzli aims to reduce the JPEG images file size, thus reducing the amount of data needed to be stored and the time it takes to transmit, while maintaining compatibility with existing browsers, image processing applications and thus with the JPEG standard. Guetzli compression methods mostly target higher quality images.

Technical Approach

As Guetzli targets very high perceptual qualities, it performs a close-loop optimization where several JPEG candidates are produced. The selection of the best candidate is where the feedback is given by Butteraugli [31], a quality metric for image and video compression developed by Google [31] that targets to mimic the human visual system and takes into account its properties, some of which aren't usually taken into consideration by other JPEG codecs [32]. To achieve a better rate-distortion performance, Guetzli uses the following techniques:

- **Subsampling** – This technique is the most obvious tool since it instantly provides a more compressed image with almost no perceptual quality loss. However, its use starts to be perceptible for high quality images. Since Guetzli targets higher quality images, chroma subsampling is not used [32].
- **Quantization** – Images encoded with usual JPEG encoders typically have inhomogeneous quality, i.e., images often exhibit alarming artifacts in some areas of the image. Guetzli attempts to cause a degradation in visual quality that is more homogeneous by using more aggressive quantization in areas less prone to exhibit alarming artifacts and less aggressive quantization in other areas more prone to such artifacts. This yields smaller JPEG images with better perceptual qualities [32].

  The first intervention by the quantizer occurs by tuning the global quantization tables into coarser ones, decreasing the magnitude of the stored coefficients. The second occurs by aggressively replacing small-magnitude coefficients by zero (similar to JPEG on Steroids) or, in case the coefficient is not small enough, decreasing its magnitude, thus further compressing the image – the successfulness of this encoder comes from choosing the right coefficients to zero out. This process takes place several times with different parameters, therefore producing several JPEG coding alternatives. Then, using the Butteraugli metric, the best JPEG coding alternative is chosen as the final output. This is a slow process, therefore making Guetzli a rather slow encoder.

Through a handful of techniques Guetlzli is able to achieve a higher perceptual quality than typical JPEG encoders. However, since its operation is based on a closed-loop optimization, the inherent computational complexity makes it not suitable for low-complexity applications.

2.2.2 JPEG 2000 Codec

JPEG 2000 is an image coding standard developed by the Joint Photographic Experts Group (JPEG) and released in 2000 . It was intended to address several limitations of the original JPEG standard, e.g. blocking effect for highly compressed images, inefficient scalability, absence of error resilient techniques and object-based representation [33].

Objectives

JPEG 2000 standard aims to specify an image coding solution for different types of images with different characteristics. Its architecture enables its use in a wide variety of fields, from digital cameras to image archives/databases, medical imaging, among many others [34]. The main JPEG 2000 requirements were a better compression than JPEG, especially for highly compressed images, both lossless and lossy compression, random assess to spatial regions of a given image, error resilience and both quality and spatial scalability [33].

Technical Approach

Although similar to the JPEG architecture, JPEG 2000 uses the Discrete Wavelet Transform, instead of the DCT, as shown in Figure 14.

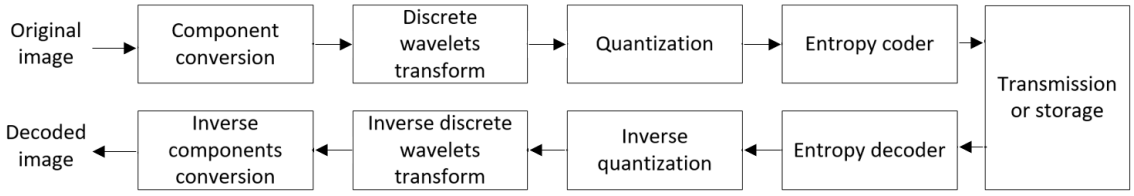


Figure 14 - Basic JPEG 2000 coding architecture [33]

The encoding procedure works as follows:

- **Original Image** – The original image may be encoded as a whole or equally divided into rectangles, named tiles; [33].
- **Component conversion** – The original image components are converted into a more suitable color space, notably YCrCB. Two component conversions are available: a reversible one (for lossless or lossy compression) and an irreversible one (only for lossy compression) [33];
- **Discrete wavelet transform** – The Discrete Wavelet Transform (DWT) works by decomposing each tile (or the complete image) into a set of sub-bands. This process is accomplished by processing the high and low frequency parts of the image independently. This happens by filtering the image along the horizontal direction using a low and high-pass filter (followed by a sub-sampling in both directions). Then, the two resulting filtered images are again filtered using a low and a high-pass filter (followed again by sub-sampling), but this time along the vertical direction, thus obtaining four sub-bands (or sub-images, which is visually more understandable), see Figure 15. The sub-band composed by the horizontal and vertical low frequency information can be seen as a low-resolution version of the original image, and the remaining sub-bands as the missing details. This implies that the previously described process can be applied any number of times to this low-resolution image, further improving the energy compaction, see Figure 15. This recursive process

is called dyadic decomposition [35]. When decoding, a low resolution image (or a full resolution but low quality image) may be easily extracted. If desired, same further sub-bands can be extracted, thus successively enhancing the quality/resolution of the first image, and so on, this providing quality and resolution scalabilities [36].

Figure 15 – Example of dyadic decomposition [35].

- **Quantization** – Process in which the DWT coefficients are quantized, i.e. the precision of the coefficients is reduced, thus further compressing the image [33]. This process sets the rate-distortion trade-off.
- **Entropy coder** – Uses the Embedded Block Coding with Optimized Truncation(EBCOT) algorithm, which is responsible for entropy coding the DWT coefficients, thus generating the corresponding binary stream [33].

JPEG 2000 surpasses the RD performance of its predecessor by using a wavelet-based method. However, it lacks support in many applications, which can cause compatibility problems.

2.2.3 Daala Codec

Daala is the name of a relatively new video coding technology, which resulted from a collaboration between the Mozzilla Foundation [37], Xiph.Org Foundation [38] and others [39]. Its first (alpha) release occurred in mid-2013, and it aspired to surpass the state-of-the-art image codecs at that time [40], notably the MPEG HEVC (High Efficiency Video Coding) standard [41] and Google's VP9 [42]. The project was declared finished by the end of 2015 [43] because it was decided to join efforts with the emerging Alliance for Open Media which targets the development of a high efficient and royalty free video coding solution well-known as AV1 (AOMedia Video 1) [44]. Despite Daala's video coding project being discontinued (or rather, moved to AV1), the still image codec may still continue to be improved [43].

Objectives

Daala is aimed to be a suitable proposal for a new video coding standard for the Internet and real-time applications [45]. Its main objectives include:

- **Avoid traditional coding techniques (to avoid licensing complications)** – Current state-of-the-art video codecs are based on techniques that date back to the ITU.T H.261 recommendation, which have been further refined since then at the cost of exponentially increasing the complexity, especially the encoder complexity. Daala tries to step away from this trajectory by adopting a new

codec design and new coding techniques [40]. By doing so, it should also avoid licensing complications [46], which is of major importance.

- **Optimize for perception** – When using a quality metrics-based coding approach (i.e., a higher metric-rated image is considered to have better quality than a lower rated one, which is not necessarily true), companies optimize for that given metric (e.g. Peak Signal-to-Noise Ratio (PSNR), Structured Similarity (SSIM, etc.) in order to push their intellectual property onto the standard and not being left out of the royalty payments, thus optimizing the codec for a given metric and not for a better perceptual quality [47]. Daala moves away from such approaches by optimizing for perception instead for a given metric;
- **Open source** – To increase its success and adoption, Daala is available free of any royalties and its reference implementation is being developed as open-source software [45];
- **Superior compression efficiency** – By using new coding techniques, Daala aimed to outperform state-of-the-art codecs, as HEVC and VP9.

Technical Approach

The avoidance of traditional coding techniques forced the developers to come up with a rather different architecture that turned out to provide very good results. The encoding architecture is presented in Figure 16.

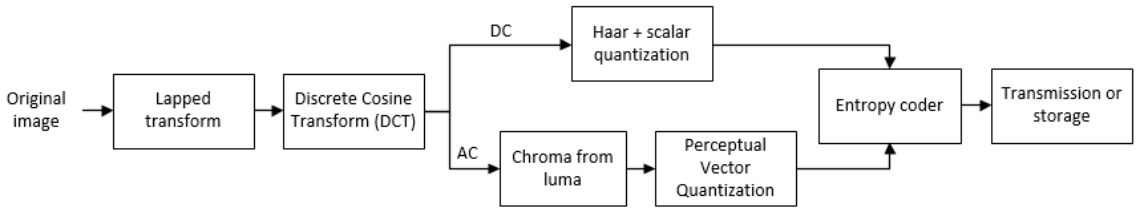


Figure 16 – Basic Daala intra encoding architecture [48].

Daala's most relevant still image coding related tools will be considered in the following:

- **Lapped Transform** – A lapped transform is any transform where each block overlaps its neighbouring blocks. Many image and video codecs use the DCT transform, which may result in strong discontinuities between the blocks (blocking artifacts), especially for high compression, which is also the reason why many image and video codecs use adaptive deblocking filters [40], therefore smoothing the discontinuities and improving the subjective quality.

  The implementation of the lapped transform in the Daala codec uses a non-adaptive post-filter similar to a deblocking filter. However, it also includes a pre-filter that is the exact opposite of the post-filter (i.e., the filter is reversible): by processing the block boundaries, it removes inter-block correlation, making the picture very blocky, as seen in Figure 17. At this stage, the blocks become very flat, greatly reducing the amount of high frequency DCT coefficients, thus providing a significantly higher coding gain. This technique has a lot less patents on it, making it easier to work with when a royalty free codec is targeted.

  The downside is that the use of a lapped transform disables the use of standard intra-prediction since, to decode a given block, it is necessary to access the decoded neighbouring blocks, but the neighbouring blocks also need to access the block in question to be decoded. However, this enables

the use of a technique called Haar DC which, by hierarchically coding DC coefficients using the haar transform [49], it compensates from not having an intra-predictor [50].

Figure 17 – Left: original image; Right: image after pre-filter [40].

- **Chroma from luma** – To minimize the components correlation, image and video codecs mostly use the YUV color space. However, these component channels are still somewhat correlated. Daala takes this into consideration and explores the correlation between channels by using the frequency domain, further increasing its compression gain by estimating chroma coefficients from luma ones. No other modern codec takes this into account although it almost made it to be included in the HEVC standard. Although significantly improving the subjective visual perception, it actually translates in a lower PSNR rating.
- **Perceptual Vector Quantization (PVC)** – PVC offers a lot more flexibility when compared with regular scalar quantization. This facilitates the implementation of activity masking thus taking into account that noise in areas lacking texture being easily perceived, than in more textured areas. This is performed by i) increasing the amount of quantization noise in high-contrast parts of the images, where the noise is less perceptible, and decreasing it less-contrast ones; and ii) keeping some AC coefficients in low-contrast regions [51]. Chroma from luma is also facilitated by PVC, since it treats the gain (contrast) separately from the shape (details) [52].
- **Directional deringing filter** – Ringing artifacts [53] are artifacts that appear near high frequency transitions. Although there are filters available to reduce this artifact, there is a high risk of simultaneously blurring the image. Daala's directional deringing filter mitigates this risk by first attempting to find the direction for the edge/pattern (in each block), and then applying a conditional replacement filter (that averages a given sample by looking at nearby samples, except those that differ by more than a certain threshold) along that direction. Then it applies the same filter (with a lower threshold) along the most orthogonal direction (horizontal or vertical). Daala's deringing filter provides a significant subjective quality improvement for Daala. In fact, it has also been adopted in AV1 [54].

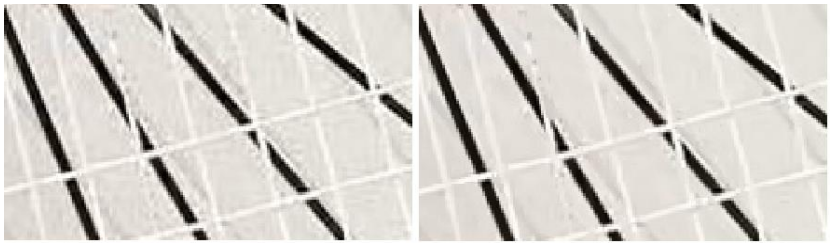


Figure 18 – Left: image with ringing artifacts; Right: left image after applying the directional deringing filter [54].

2.2.4 VP9 Codec

VP9 is an open and royalty free video codec that follows its predecessor VP8, which was developed in the WebM project [55] and further developed by Google. The codec was defined in June 2013 (although subject to bug-fixes), and it was designed to address overall online video consumption growth and H.264/AVC and VP8's bandwidth limitations, while attempting to compete with the HEVC video codec (also released in 2013), notably for ultra-high resolution videos [56]. Unlike Daala, VP9 further improves H.264/AVC and VP8 techniques, therefore continuing to move in the same technical direction. Its successor, VP10, will, just like Daala, be incorporated into AV1, which targets a royalty free codec.

Objectives

VP9 is a video codec developed for Web use; therefore, its objectives are similar to the Daala codec. The VP9 main objectives include:

- **Open source** – To increase its success and adoption, the codec software should be open for anyone to improve [55];
- **Web optimized** – Focus on addressing the needs of serving video on the web, i.e., low complexity, minimal codec profiles and sub-options and a simple container format [55].
- **Superior compression efficiency** – VP9, as other video codecs, aims to surpass the performance of the HEVC codec.

Technical Approach

VP9 uses the same coding framework as HEVC. Its encoding architecture is represented in Figure 19.

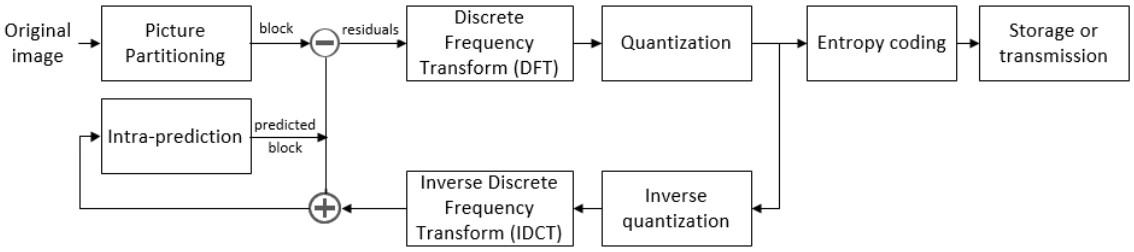


Figure 19 – Basic VP9 intra encoding architecture [38].

The main VP9 modules when performing Intra coding are:

- **Picture partitioning** – The image is first divided into Superblocks of 64x64 samples, which, in turn, can be subdivided down to 4x4 sub-blocks using a quadtree structure [57], like HEVC. Each of these sub-blocks is divided into prediction and transform blocks.
    - **Prediction block** – It can either be the whole sub-block or a smaller block resulting from the division of the sub-block, going down to 4x4 (see Figure 20, left). The prediction block is where the predicted data is stored i.e., the encoder goes through a variety of modes (whether they are Intra or Inter), and finds out what prediction mode provides a prediction that resembles more the block that is being coded; after, only the residue (this means the difference between the original block and the prediction block associated to the chosen

mode) needs to be coded, if any. Unlike HEVC, where the prediction block has to have the same size of the sub-block in Intra coding, VP9 allows any sub-division (see Figure 20, left);

- **Transform block** – This is another possible division (or not as the transform block can match the size of a sub-block) of the original sub-block that is associated to the transform of the prediction residue for a division (or the whole) of a given prediction block (compare the two figures in Figure 20) to the frequency domain. Transforms with dimensions of 32x32, 16x16, 8x8 and 4x4 are allowed [58];

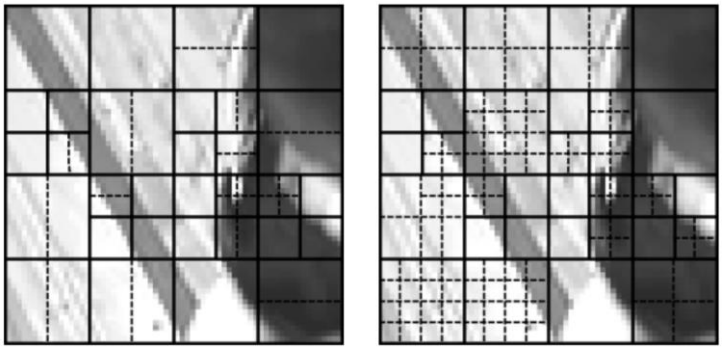


Figure 20 – Left: prediction splitting; Right: Transform splitting (solid lines represent superblock) [58].

- **Intra-prediction** – Intra-prediction relies on coding the minimum prediction error possible by exploring similarities between the block being coded and adjacent pixels of the adjacent blocks within the same frame. This is done by "pulling" the information from the adjacent pixels, effectively filling the block with as 'similar' information as possible. To achieve this, ten intra-coding modes are available: eight angular, one DC and another called True Motion (an alternative to the planar mode in HEVC, useful for smooth blocks). If the chosen angular mode is the horizontal one (several can be chosen), then pixels from the left will be "pulled", effectively filling pixels of the block being coded, and so on. The prediction error is then transformed into the frequency domain.
- **Transforms** – For Intra modes, VP9 can use the regular two-dimensional DCT, hybrid DCT/ADST (Asymmetric Discrete Sine Transform) – one dimension using DCT and another using ADST – or two-dimensional ADST [56]. HEVC uses either two-dimensional DCT or DST (Discrete Sine Transform);
- **Entropy coding** – Uses 8-bit arithmetic encoding and does not change adaptively within a frame, like the HEVC's Context Adaptive Binary Arithmetic Code (CABAC) does.
- **Post-processing** – For post-processing, VP9 only includes a deblocking filter which first smooths the vertical edges and then the horizontal ones. It is able to detect flat regions, thus varying the filter's parameters accordingly [58].

Despite the techniques provided by VP9, this codec is still not able to surpass the compression rates achieved by HEVC [38]. This was expected due to the use of less complex techniques such as far inferior number of intra prediction modes and a less advanced entropy coding procedure.

2.2.5 Thor Codec

Thor is a video codec that has been developed by Cisco Systems [59]. Its structure is similar to modern video codecs, although with less features than HEVC. It was designed to achieve high video compression with moderate complexity, using only royalty free Intellectual Property Rights (IPR); this is the main reason why lots of HEVC features are missing (as it happens in VP9). Together with VP9 and Daala, its development team has shifted its efforts to AV1, where the best practices of each codec will be put together to develop a royalty free video codec capable of surpassing HEVC [60].

Objectives

Thor aimed to be a video codec designed for web use, just like VP9 and Daala. Hence, its objectives are very similar to VP9, i.e., Thor aims to build a royalty free video codec with moderate complexity, suitable for web use [61].

Technical Approach

Thor also uses a block-based hybrid video coding architecture, i.e., it calculates a prediction residue through Inter/Intra modes and transforms it to the frequency domain, similar to other widely used video coding standards (VP9, HEVC, H.264/AVC). Considering only Intra encoding, the following modules are relevant:

- **Picture partitioning** – Picture partitioning is very similar to VP9. The image is divided into Superblocks that can have either a resolution of 64x64 or 128x128 pixels (a larger block size usually leads to better efficiency for higher resolutions [58]). In turn, each Superblock can be further divided into coding blocks down to 8x8 samples, in a quadtree structure [57]. Each coding block is divided into Transform Blocks and Prediction Blocks as it happens with VP9. For a more thorough explanation of prediction blocks and transform blocks, check the previous subsection.
- **Intra-prediction** – Intra-prediction is performed as for VP9 and it is signaled at the Coding Block level. Thor provides a set of 8 Intra coding modes, against the 10 modes provided by VP9 and the 35 provide by HEVC: one is the DC mode, the others are angular modes [62].
- **Transforms** – Unlike VP9 and HEVC, Thor only uses the Discrete Cosine Transform (DCT) that is applied to the Transform Block. On the other hand, unlike VP9 or HEVC, it is capable of transforming Transform/Coding blocks of up to 128x128 size. However, for blocks of 32x32 pixels and above, only the first 16x16 lower frequency coefficients are quantized and transmitted – the other are discarded – thus reducing the amount of coefficients that need to be coded.
- **Entropy coding** – Relies on context adaptive variable length codes, which is far less sophisticated than HEVC's CABAC [63];
- **Post-processing** – Post-processing is applied with two steps: the first refers to the use of a deblocking filter (filters 2 pixels on each side of the edge); the second refers to the Constrained Low Pass Filter, which is applied after the deblocking, and is based on a rate-distortion optimization decision (replaces the sample adaptive offset filter used in HEVC) [63].

Although using techniques similar to the ones used by VP9, Thor offers less possibilities, such as less Intra coding modes and transforms. While this might yield a lower computational complexity, its RD performance is expected to fall behind VP9.

2.2.6 Still Image Coding Performance

To perform an accurate RD performance evaluation of the codecs previously described, the same set of images as well as the test conditions used for the Compression Grand Challenge organized in the context of the International Conference on Image Processing 2016 (ICIP 2016) [28] has been selected; this set of images (represented in Figure 21) has been selected by the JPEG committee, which is a good guarantee of content representativeness. Taking into account the specific purpose of this MSc Thesis, an image of the inside of a fishing vessel was also added to the set (Figure 22). The resulting image set has the following properties:

- **Spatial resolution** – 1280 x 800 or 1152 x 800 pixels, for ICIP 2016 image set; 5120 x 2944 pixels for the fishing vessel image;
- **Color space** – RGB;
- **Chroma subsampling** – None (4:4:4).

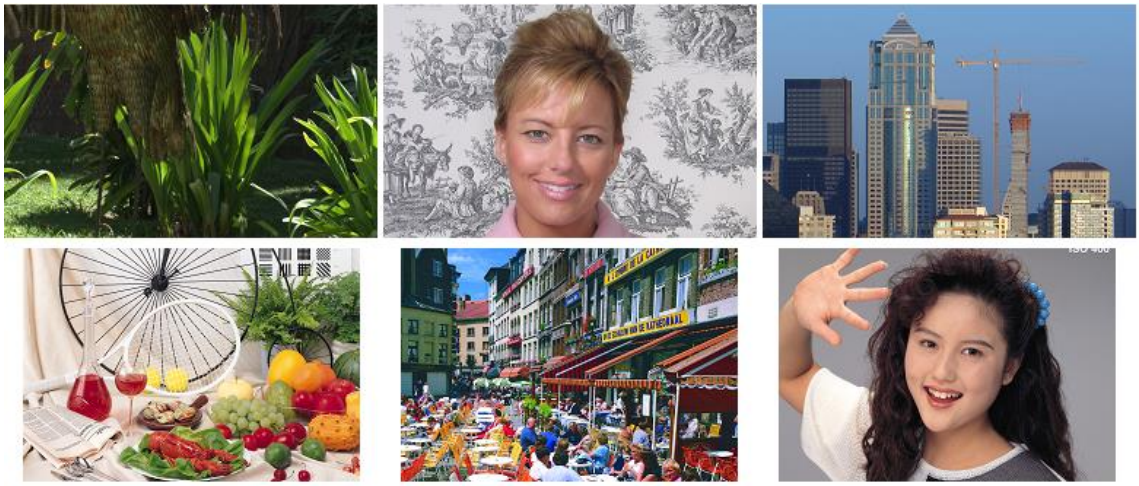


Figure 21 – ICIP 2016 image test set. First row, from left to right: *honolulu, p08, p26* (1280x800); Second row, from left to right: *bike, cafe, woman* (1152x800).

Figure 22 – Specific scenario image, inside a fishing vessel: *boat* (5120x2944)

The test material and test conditions are described in the following:

- **Spatial resolution** – Since producing small size files for a target quality is a key requirement of any codec and the original images may have a resolution which is too high for the target application, the

original images have been subsampled. Two different sets of spatial resolutions are targeted: a medium spatial resolution set (VGA-like resolution) and a low spatial resolution set (CIF-like resolution). To achieve this, both horizontal and vertical resolutions are repeatedly halved until obtaining the two previously mentioned resolution sets:

1. Medium spatial resolution set: both 640 x 400 and 576 x 400 resolution images for the ICIP 2016 and 640 x 368 for the fishing vessel image;
2. Low spatial resolution set: Both 320 x 200 and 288 x 200 resolution images for the ICIP 2016 and 320 x 184 for the fishing vessel image

- **Color space** – Each image from the two sets has been converted to YUV;
- **Chroma subsampling** – Chrominances have been subsampled to 4:2:0, i.e., the resolution of both vertical and horizontal directions has been halved;
- **Compression rates** – At ICIP 2016, each codec compressed each image for 8 different bitrates: 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75 and 2 bpp. Since the target application requires very compressed images, the images will be compressed for the bitrates of 0.10, 0.15, 0.20, 0.30, 0.40, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75 and 2.00 bpp.
- **Codec settings** – All codecs will be evaluated using default settings (unless chroma subsampling is enabled by default, in which case this setting will be disabled).
- **Codec implementations** – As it is well known, several implementations may exist for the same standard coding specification as these specifications do not rigidly specify all encoding tools and parameters, just what is essential for interoperability. The performance assessment has been made for several encoder implementations, standard and non-standard based; for JPEG, three implementations have been selected, mainly different in the way they perform the quantization. The encoder implementations used to compress the images and, therefore, being subject to performance evaluation, are:

  1. **JPEG Standard** – This corresponds to the legacy JPEG implementation. The implementation used was the one available in [64], which provides a complete implementation of the ISO/IEC 10918-1 standard [65].
  2. **JPEG on Steroids –** JPEG on Steroids is the second selected JPEG compliant solution [64]. To use this codec the implementation available in [64] can also be used with, however, the following flags:
     - "-qt 3": selects the Image Magick quantizer;
     - **"-**oz": selects the soft-threshold quantizer;
     - **"-**v": selects the progressing mode;
     - **"-**h": optimizes Huffman tables;
  3. **Guetzli –** Guetzli is the third selected JPEG compliant selection [66], which has been used with default settings. Since this codec is unable to accept YUV files, a png file had to be provided instead. Also, Guetzli uses YUV444 meaning that no subsampling is applied to the chrominances.
  4. **JPEG 2000 –** This is the OpenJPEG implementation, which is a JPEG 2000 compliant solution (release 2.1.2 [67]) used with the default settings, meaning that only one quality layer is used;

5. **JPEG 2000 Layers –** Same as the previous codec, but with two quality layers instead of one.
6. **Daala –** This is a non-standard based coding solution, which implementation is available at [68]; the default settings were used.
7. **VP9** – VP9 implementation available in the libvpx library [69] has been used with default settings;
8. **Thor –** This is another non-standard based coding solution, which implementation is available at [70]; the default settings were used.

- **Objective evaluation metrics** – To assess the performance of each codec for each bpp, two different metrics will be used:
  1. **$PSNR_Y$ (Peak Signal-to-Noise Ratio)** – Measures the ratio between the original signal and the error introduced by compression which affects the fidelity of its representation [71]; the PSNR will be measured just for the Y channel.
  2. **$SSIM_Y$ (Structural Similarity)** – Measures the similarity between two images based on structural information, where one is uncompressed/distortion free. It is designed to improve traditional methods as PSNR, since it is a perception-based model, as PSNR only measures the absolute error without any perceptual reasoning [72]; again only measured for the Y channel.

After detailing the test material and conditions, it is now possible to assess the performance of each codec using the objective evaluation metrics previously described.

2.2.6.1 RD performance

As stated earlier, two different image sets with different spatial resolution were used. The two sets of images were compressed by each codec and then the two quality metrics, $PSNR_Y$ and $SSIM_Y$, were applied, thus comparing the original set with the compressed set. Because some codecs are unable to compress below a certain quality threshold, their results only start at higher bitrates.

The obtained data are shown in two ways:

- **Multiple codecs per image** – A $PSNR_Y$/$SSIM_Y$ comparison based on the performance of all codecs when compressing each of the images, i.e., each chart represents how the chosen codecs behave when compressing a given image.
- **Multiple images per codec** – A $PSNR_Y$/$SSIM_Y$ e comparison based on the performance of each of the codecs when compressing all images, i.e., each chart represents how a given codec behaves when compressing different images.

Due to the space limitations, all the results are presented in Appendix A using appropriate charts.

The results in Appendix A show that VP9 is consistently the codec offering the best results, regardless the metric used. The PSNR results show that Thor is consistently in second place, closely followed by JPEG2000 and Daala, although for highly detailed images Daala seems to significantly fall behind. However, the SSIM metric consistently places Daala in second place; this is due to Daala being a perceptually optimized codec which penalizes its performance when measured with a non-perceptually biased quality metric like the PSNR.

As expected, JPEG compliant codecs offer the worse results. Still, by applying a number of techniques, JPEG on Steroids offer significantly better results than the JPEG Standard codec. Moreover, the JPEG eco-system is nowadays huge and its benefits may compensate the RD performance penalty.

As for Guetzli it presents a strange behavior, being only able to compress images for higher levels of quality. This is due to Guetzli being a codec that targets higher quality images only. Because of this, it ignores requests for encoding lower-quality images, thus encoding only with high quality. Also, Guetzli presents a poor performance for the $PSNR_y$ and $SSIM_y$ metrics. This is due to Guetzli encoding images with YUV444 (this means no chroma subsampling) only; while this can increase the overall quality, such quality improvements are reflected in the metrics results, as they only analyze the luminance channel.

2.2.6.2 Complexity Performance

Considering the target application in this Thesis, it is especially important that the selected image codec offers low complexity implementations. Therefore, a comparison regarding the processing time of each codec was made and is presented in Appendix B. The complexity measurements have been made using a PC with an Intel Core i7-3770 CPU @ 3.40GHz – 3.90GHz. Due to the extremely high processing times, it was decided not to represent the Guetzli results.

By analyzing the complexity results in Appendix B, it is easy to conclude that the VP9 codec is, by far, the most complex codec, followed by Daala. Although VP9 is the codec providing the best RD performance according to the results in Appendix A, this good performance is penalized with complexity and thus this codec is not the best choice for systems requiring a low complexity image coding solution. On the other hand, JPEG 2000 and Thor offer very good RD performance with far lower complexity.

Now that some of the technologies to be used throughout this thesis have been addressed, a detailed description of the scenario where the proposed solution shall be implemented, as well as the application requirements, will be given in Chapter 3.

Chapter 3

3. Application Scenario: Requirements and Architecture

This chapter will focus on how to apply a visual data processing system to Vessel Monitoring Systems. It will start by focusing on the operation and architecture of such systems and then move to operation and architecture of the solution itself, this means integrating a visual data processing system with the VMS system provided by *Xsealence*.

3.1 Vessel Monitoring Systems

A Vessel Monitoring System (VMS) is a solution used to help fisheries managers and enforcement authorities to monitor the activities of licensed vessels. In this system, vessels are equipped with a GNSS (Global Navigation Satellite System) “black box” capable of transmitting both position and speed to a Control Center on shore. This enables a simple but effective monitoring of the vessels’ activities. Fishing licenses are issued only upon the installation of the “black box” on board.

This “black box”, the so-called VMS unit, incorporates a transmitter/transceiver and has a unique identifier that matches a specific vessel. The device uses a GNSS receiver, thus obtaining its position, speed and heading. Then, using a satellite transceiver, the VMS unit sends periodic reports to the Control Center. These reports include not only position related reports, but can also sensors’ information, electronic catch reports, alarms and other useful information. These reports will then be assessed at the Control Center for regulation enforcement, notably checking if the fishing vessels are respecting their fishing licenses. It must be noted that the complete Vessel Monitoring System is composed by all the elements that play a role in its operation. Therefore, the GNSS and communications satellites, the Control Center and the VMS unit are all part of the Vessel Monitoring System, as shown in Figure 23.

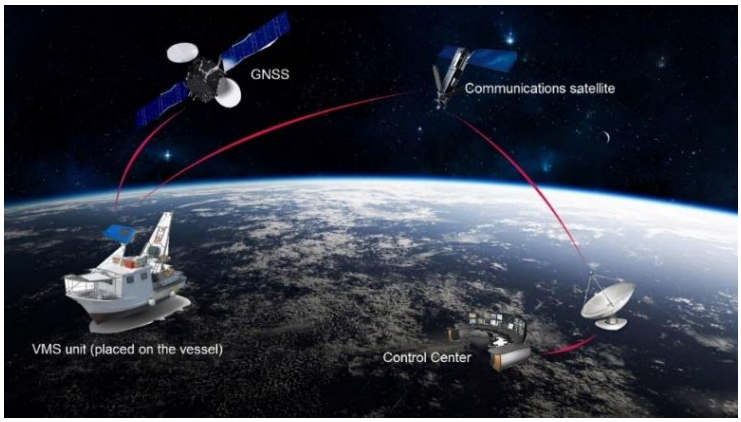


Figure 23 – Vessel Monitoring System.

It is also important to stress that the VMS is not the only tool in fisheries management. It is just one of many tools such as surface and aerial patrols, on-board observers, logbooks or dockside interviews, that are together combined in order to effectively control fishing activities. This technology is used today in almost all major fishing banks worldwide. Indeed, if a given country does not fulfill international rules, its fisheries products are banned, e.g. forbidden to be exported to the European Union [73].

The first Vessel Monitoring System was developed in Portugal as a result of a PhD work carried out at INESC. Today, Xsealence, Sea Technologies, S.A. is commercially exploring and expanding the portfolio associated with this system. Currently, it is studying the possibility of adding a video camera to its Monicap “black box” (see Figure 24), to provide the Control Center with the capability to order the capture of images or real-time video on-board to confirm (or not) that the vessel is following the regulations.

Figure 24 - Monicap Bluebox [75].

To explain how the VMS system works, a use case can be used. Let’s suppose that a fishing vessel is heading towards high sea and at some stage it crosses a zone where fishing is prohibited. Although the crossing of prohibited fishing zones is allowed, fishing activity inside is not. If the fishing vessel decides to start fishing, there is no alarm based only on position and/or speed report. At the Control Center, although an experienced eye can spot an eventual fishing activity based on the speed/position pattern (which also depends on what kind of fishing the vessel is equipped to do), it is hard to prove that the vessel was actually fishing. So, currently, the Control Center cannot prove that an illegal activity is occurring and thus fine the vessel owner. On the other hand, if a camera is placed on the vessel, an order for an image capture would be able to prove without any doubt if the vessel is actually fishing. Furthermore, if the camera is able to recognize fishing activities, it can automatically alert the Control Center when an illegal fishing activity is taking place.

3.1.1 VMS Solutions in the Market

While searching for VMS solutions, one can find different approaches, depending on the business models of the several VMS solutions providers:

1) **VMS unit hardware only** – The VMS provider develops the VMS unit hardware only (the VMS equipment to be installed on the vessel, with all the required functionalities) and uses a third-party software solution (see Figure 25).

2) **Control Center software (and possibly equipment) only** – The VMS provider develops the Control Center software only and uses market available commercial mobile units that are sold together with the software or bought separately by the client (see Figure 26).

   Some companies provide a complete Control Center with all the necessary equipment for the client to monitor their fleet, while others offer a hosted system, in which the VMS provider itself hosts the Control Center, giving the client a web-based application to access the data. This enables high installation (and especially maintenance) savings, since it is not necessary to install and maintain a Control Center at the client.

   A cloud solution may also be used – a contract is made with a cloud provider, based on the size of the fleet to be monitored and range of services offered [76].

3) **Fully proprietary solution** – The VMS provider may also develop both the VMS unit hardware and the Control Center hardware and software, offering the client a fully customized solution.

Figure 25 – Example of VMS equipment [77]; Left: VMS unit; Right: Human to Machine Interface (HMI) terminal [78].

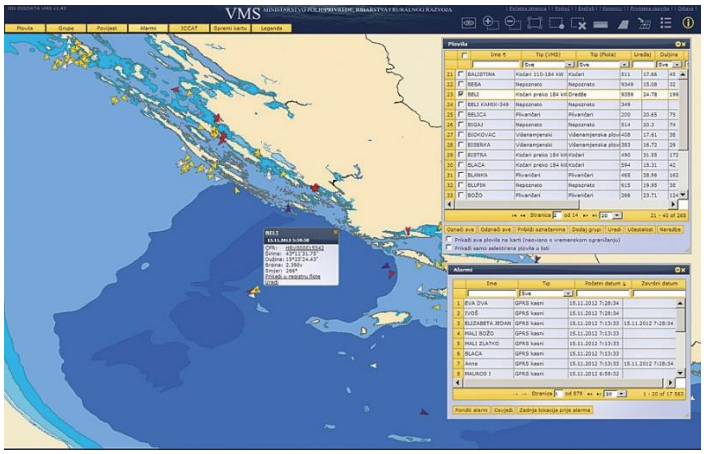


Figure 26 - VMS software example [79].

3.1.2 VMS Architecture

Although there are many VMS providers with their own solutions, these are mostly based on the same high level architecture. In the first subsection, the high level architecture of the VMS system will be presented with the purpose of introducing each of the VMS elements. The second subsection will focus on the Xsealence's VMS unit solution, named Monicap (and not on the whole system), particularizing its modules.

VMS High Level Architecture

A Vessel Monitoring System is composed by several elements, as shown in Figure 27:

- **VMS unit** – Senses and stores various types of data, e.g. position, date/time, integrity, status, etc., and sends it to the Control Center. The reporting frequency and data types are established by the management authorities. Each VMS unit has a unique identifier. It is extremely important that it is tamperproof, since tampering the unit may result in substantial financial gains. The VMS unit is composed by the following modules:
  - **Position module** – Responsible for acquiring the device's position; it connects to a GNSS constellation and can also be part of the Communication module.
  - **Communication module** – Responsible for establishing communications with the nearest communication satellite; some solutions may include more than one communication module to save communication costs (e.g. using a GSM/GPRS module so that data can be uploaded when near the coast).
  - **Central Processing Unit (CPU)** – It is in charge of the VMS unit control and information processing.
  - **Human to Machine Interface (HMI)** – Usually in the form of a tablet or a simple on-board display, it enables the user to interact with the equipment. Through this module, the user can check the VMS unit status, ask for help (SOS), check location, check weather, fill electronic catch reports and more, depending on the VMS unit.
  - **Power supply** – Due to the harsh environment and unreliable electric generator, it must be sturdier and more sophisticated than usual. For example, it must endure voltage peaks and be EMC (Electromagnetic Compatibility) compliant; most systems have a battery installed, keeping the system powered in case of failure.
- **Communication satellite** – Responsible for establishing the connection between the vessel and a Satellite Earth Station, through the satellite network. Among the most used satellite constellations are Iridium [80], Inmarsat [81] and Argos [82]. Due to full global coverage and lower prices, Iridium is being increasingly used.
- **Satellite Earth Station** – Terrestrial radio station responsible for connecting a satellite network with the Internet.
- **GNSS constellation** – Through multiple satellites, the constellation enables the position module to determine the vessel position. Examples of constellations are GLONASS [83], GPS [84] and Galileo [85]. GPS is by far the most preferred method, due to its high level of accuracy, availability and low equipment cost [74].
- **Control Center** – This is where all communications and data related to the vessel monitoring system are centered. It controls the VMS unit (reporting frequency, fishing zone's management, etc.) and it is able to gather information regarding each vessel on demand. It collects the data received from the VMS unit and stores it for further assessment. The data is displayed using a Geographic Information System (GIS).
- **User:** The human interacting with the VMS unit.

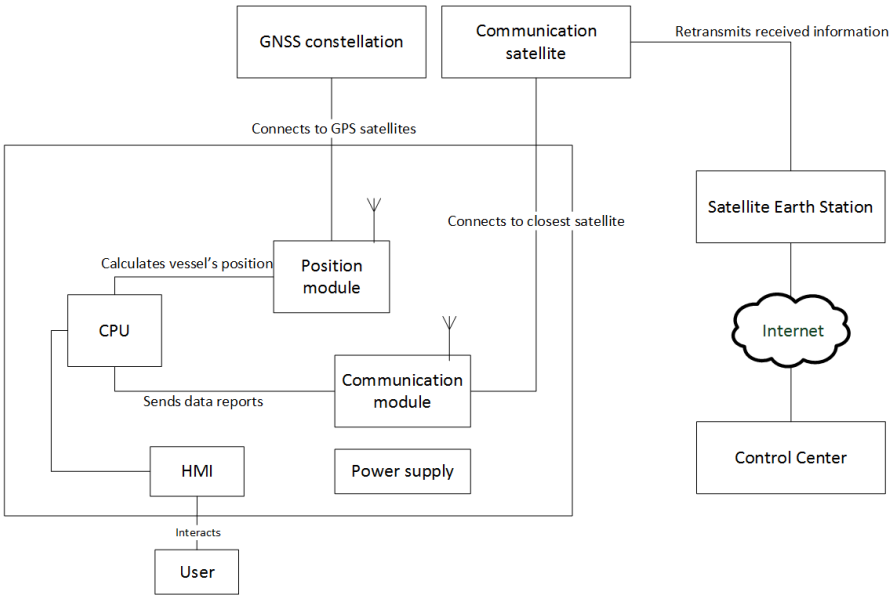


Figure 27 – VMS high level architecture.

Monicap's High Level Architecture

This section particularizes only the modules of the Xsealence's VMS unit Monicap to avoid repeating the full VMS's modules description. The communication satellite and the GNSS constellation solutions are referred on the corresponding Monicap's module. Therefore, Monicap is currently composed by the following modules, as shown in Figure 28:

- **Position module** – Implemented using a GPS module that connects to the GPS satellite constellation.
- **Communication module** – Implemented using an Iridium modem that connects to the Iridium satellite constellation.
- **CPU** – Implemented using a Single Board Computer (SBC).
- **HMI** – On-board display, as it may be seen in Figure 24.
- **Power supply** – To keep the system powered in case of failure, Monicap uses a high-capacity smart battery.

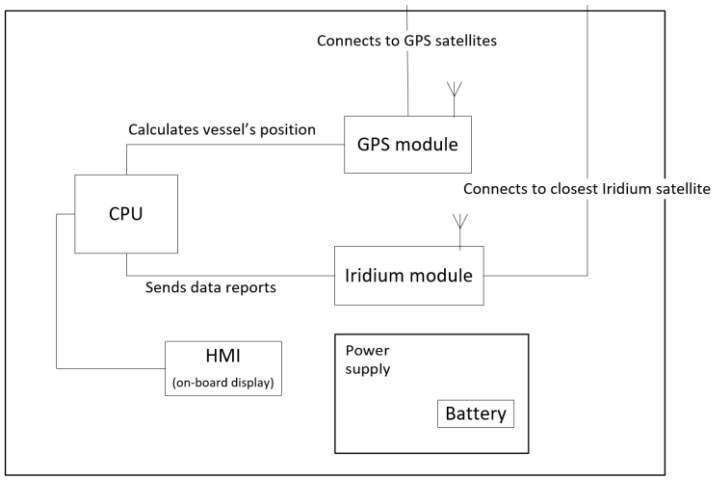


Figure 28 - Monicap's architecture.

3.2 Iridium Satellite Constellation

Iridium is a satellite constellation released back in 1998 [86] that provides global coverage with its 66 active Low Earth Orbit (LEO) satellites (spread over 6 planes), offering also lower power consumption and delay. The operation of the VMS system relies on these satellites, terrestrial gateways, and Iridium Subscriber Units (ISU). ISU are the devices that connect to the Iridium satellite network, allowing the transmission/reception of voice or data, e.g. an Iridium phone or modem. The satellite communication system uses Frequency Division Multiplexing Access (FDMA) with carrier frequencies in the range of 1616 to 1626.5 MHz, and Time Division Multiplexing Access (TDMA) with a frame length of 90 ms [87]. When an ISU connects to a satellite, a timeslot within a given channel is assigned for communication [88].

Several Iridium communication services are available, from M2M (Machine to Machine) to circuit switch voice and data services, and even broadband data. Circuit switch voice and data services can reach rates up to 2.4 kbps [89] with a latency averaging at around 1800 ms round-trip time (RTT) for data [90]). Broadband services can reach up to 134 kbps using Iridium Pilot, the latest Iridium broadband capable hardware [91].

3.2.1 Short Burst Data Service

The Iridium modem used by Xsealence only supports the Iridium's Short Burst Data (SBD) service, meaning that it does not support voice, circuit switched data or short message services (SMS) [92]. This service is an M2M type of service and aims to efficiently transmit small bursts of data (e.g. information regarding position, in the case of VMS) between field equipment and the centralized host computer [93]. It targets applications that need to periodically transmit small size messages, making it ideal for a VMS system.

The Iridium SBD service distinguishes between sent and received messages (their sizes differ, mainly). Since the goal is to transmit data from the Iridium modem placed on the vessel to the Control Center, more focus will be given to Mobile Originated (MO) messaging in the following sections.

3.2.2 MO Messaging Operation

The user uses a Field Application that is associated to the hardware and software that 'talks' with the ISU, sending/receiving data to/from it and where the desired message is 'written' and sent. The message is then loaded into the ISU's MO buffer, where it is stored. If the Iridium satellite network is available (that is, if the ISU can "see" an Iridium satellite), the ISU will connect to it and establish an SBD session with the Iridium Gateway, followed by the MO message to be sent. Only one MO message may be sent per SBD session. The MO message will be queued at the Iridium Gateway, which will then establish a TCP/IP connection to the Vendor Application, e.g. the hardware and software at the Control Center, although a standard email protocol can also be used. Again, only one MO message is sent per socket connection. After the Vendor Application receives the MO message, the Iridium Gateway closes the

socket connection, thus successfully ending the process [94]. This is the communication solution to be used for image/video transmission in this Thesis.

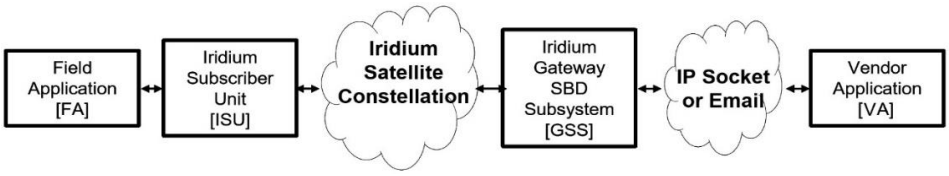


Figure 29 - Short Burst Data's architecture [94].

SBD's Service Characterization

The SBD service characteristics and behavior are described below:

- The interface between the Iridium Gateway and the Vendor Application uses either standard mail protocols or an IP socket type interface (DirectIP protocol) to send/receive messages [94]; using the latter will provide lower latency [94];
- Depending on the Iridium modem used, the message size can range from 340 to 1960 bytes (see Table 1). In the latter case, the message includes a first 70 bytes segment followed by up to 14 segments of 135 bytes;
- Global latency for a message is stated to range from 5 seconds (for 70 bytes messages) to 20 seconds (for 1960 bytes messages) [95]. This implies a data rate of approximately 126 bytes per second (~ 1k bit/s), which matches unofficial information found online (which mentions 125 bytes per second) [87];
- The user is billed by the byte (providing that a minimum is exceeded, e.g. 10 bytes) [94];
- Messages sent from the ISU to the Iridium Gateway are guaranteed to be delivered error free [94];
- For Mobile Originated (MO) messages, the Iridium Gateway acts as a client, seeking to establish a connection to the Vendor Application;
- DirectIP protocol uses bi-directional TCP/IP socket connections; for MO messages, no acknowledge is expected from the vendor server [94];
- The Iridium Gateway queues up to 10,000 MO messages using a FIFO model [94];
- The SBD ISU's MO buffer will maintain all MO messages (until it is powered off, overwritten with new data or erased explicitly by a command) [94];
- If the ISU is unable to connect to an Iridium satellite, it will keep trying to establish a connection, with a periodicity that increases until it reaches its maximum, notably 120 seconds [88].

Table 1 – Iridium transceivers using SBD [94].

Transceiver Name	Maximum Mobile Originated Message Size (Bytes)
9522A	1960
9522B	1960
9601	340
9602	340

Pricing

The Iridium SBD service was built around applications that use a very low amount of data. As such, one expects monthly fees to be lower than other services that need higher bandwidths. Indeed, SBD's

monthly fees are lower than those found for voice, circuit switch data and broadband services. However, when comparing the price per byte of the SBD service with the others, the difference is quite substantial, as expected, given the nature of the SBD service.

On Table 2, data plans for three different Iridium services are presented to give an idea about pricing differences, one of which being the SBD service. The remaining two are:

- **Iridium GO!** [96] – Service based on a device that connects to an Iridium satellite much like an ordinary Iridium phone, with the difference that it behaves like a hotspot, enabling users to connect their smartphone to the device. Using a smartphone application, the user can make phone calls that are routed through the hotspot and into the Iridium network, like an ordinary Iridium phone would. Therefore, the user can only use circuit switch voice or data services at rates up to 2.4 kbps. Voice is not represented in Table 2 but its monthly included minutes and overage charge is, in most plans, the same as for data;
- **Iridium OpenPort** [97] – Iridium's broadband service with a plan which comprises only the service, being the hardware purchased separately. The Iridium Pilot [98] is the most recent piece of broadband capable hardware. Table 2 shows data plans as an example of pricing. If voice is required, an additional plan needs to be purchased. Depending on the plan, prices can range from $0.99/min to $0.39/min [99].

Table 2 – Data plans for various Iridium services.

	SBD [100]		**Iridium GO!** [101]		**Iridium OpenPort (using Iridium Pilot) – data plans only** [99]		
Monthly fee	$15.99	$33.99	$46.95	$99.95	$54	$237	$1,500
Data included	None	30 kB of data	5 data minutes	150 data minutes	0 MB included	30 MB included	1500 MB included
Data overage charge	$1.09 / kB	$1.09 / kB	$1.29 / min @ 2.4 kbps (up to 18 kB / min)	$1.09 / min @ 2.4 kbps (up to 18 kB / min)	$13.95 / MB	$8.50 / MB	$1 / MB

From Table 2, it is evident that, although for small amounts of data, SBD is a cheaper choice, for bigger amounts it is useful to consider other options.

3.2.3 Iridium NEXT

Iridium NEXT is the second generation of Iridium satellites. Having successfully launched the first set of satellites on January 14th, 2017, Iridium plans to have the network fully operational by mid-2018 [102]. It aims to fully replace the current constellation, retaining the same architecture while increasing the capacity, data rates, and introducing new services, including:

- Higher quality voice at 2.4 kbps for standard quality and 4.8 kbps for high quality, with better voice compression [103], against the current rate of 2.4 kbps [104];
- Extended capacity and operational flexibility for M2M [105];

- Data rates up to 88 kbps for mobile class equipment (currently, data rates are up to 2.4 kbps for either voice or data) and up to 1.4 Mbps for broadband terminals (currently, Iridium OpenPort offers data rates up to 134 kbps) [104];
- Broadcast (undirected regional services at up to 64 kbps) [105];
- Data rates of up to 8 Mbps, for fixed/transportable terminals (using Ka-band) [105];
- Service continuity and backward compatibility [106].

These improvements will not only enhance current services and add new ones, but will also lower the costs, thus providing new service opportunities. With increased bandwidth, it might be possible to implement actual real-time video, or at least something closer to it, thus providing the users with increased means for communication.

3.3 Application Scenario Operation and Requirements

When designing a solution for a problem, it is of most importance to carefully define and address all aspects of its operation, by making a detailed description of the application scenario operation. With this in mind, the technical requirements can then be established. As such, this section will present a detailed description of the application scenario operation, followed by a list of the identified requirements.

3.3.1 Operation

By carefully analyzing the application scenario, a flowchart of its operation may be made as shown in Figure 30. To better describe the operation at the two sides of the communication system, the application scenario operation was split into two flowcharts: a *'At the vessel'* flowchart and a *'At the Control Center'* flowchart. While the former aims to define the operation on the vessel's side (controlled by the VMS unit), notably the steps taken by the event detection and coding modules and related footage storage and transmission, as well as answering the Control Center requests, the latter aims to explain the operation at the Control Center on the shore.

To better understand how the system operates, a brief walkthrough of each flowchart is given below. The flowchart features three different box types: A *rectangle* implies a process, a *diamond* implies a decision (as in an answer to a question), and a *rectangle with curved edges* implies a starting/ending process or passive element (not directly taking an action). Some processes/decisions will be explained individually, while others will be explained in groups.

At the vessel

The 'At the vessel' operation proceeds as follows:

1. **Starting** – The system at the vessel is switched on.
2. **Visual data acquisition** –The camera continuously acquires visual data at a certain frame rate and spatial resolution which is provided to the event detection and coding modules.
3. **Pending requests checking** – To avoid having pending requests, the system checks if there are any pending requests from the Control Center. If there are, the VMS unit addresses them. There are four possible types of requests:

- ***Event history request*** – The VMS unit checks the stored event history and transmits it back to the Control Center;
- ***Erasure/protect-from-erasure request*** – The storage management module at VMS unit either erases some specific visual content or protects it from erasure;
- ***Event's frame request*** – The VMS unit transmits a low-quality frame from the event in question. If a frame from that event was already sent, the VMS unit transmits a different one distanced in time and with medium quality, thus providing a better understanding of that event to the Control Center.
- ***Live visual data request*** – The VMS unit responds to this request by ranking current visual data as very relevant for a certain pre-defined period of time, implying that it is coded, stored and a low quality representative frame initially transmitted to the Control Center.
- ***Event time-lapse request –*** The VMS unit transmits a time-lapse of a given past event, this means a set of equally spaced images.

4. **Visual event detection** – The automatic event detection algorithm detects relevant events using the visual data received from the camera and determines its relevance by associating a ranking score, i.e., if fishing activities might be taking place on the vessel. If the event detection algorithm decides that the visual data is not relevant, the visual data is discarded. If an event is detected and considered relevant enough (this means its relevance is above a given *relevance threshold*), the associated data has to be further processed. The precise meaning of what an event is will be determined later while compromising the detailed semantics of an event versus the complexity of the associated detection mechanism.
5. **Visual data coding** – If considered relevant, the visual data is encoded with two different qualities, to be eventually transmitted to the Control Center (a low quality frame is immediately transmitted). This grants the Control Center the possibility of quickly receiving a low-quality footage frame, while keeping the transmission cost prices low; if after receiving the low-quality frame the Control Center requires more information, medium quality frames can then be transmitted. The visual data is also stored with a third quality, this one being of very high quality (see next bullet), with the purpose of retrieving high-quality data when the vessel arrives at the port.
6. **Visual data storage and transmission management** – This module has to manage the visual data stored at the VMS unit, as well as its eventual transmission. After coding, the visual footage is stored with the maximum available resolution and quality. If the storage is full, the oldest least relevant event is overwritten. Management actions may also result from Control Center requests. To reduce the involved complexity, it may be decided to store only images with good enough quality.

At the Control Center

The 'At the Control Center' operation proceeds as follows:

1. **Starting** – The system at the Control Center is switched on;
2. **Vessel communications monitoring –** The Control Center continuously monitors the vessel communication channel to check if any information is being transmitted (either visual data, warnings or event lists). While operating, three things may can happen: the Control Center receives data from

the VMS unit (step 3), requires additional information (step 4) or gives some command to the visual data storage module (step 5).

3. **Data reception** – The Control Center receives data from the VMS unit. One of four types of data can be received:
   - ***Event history reception*** – The Control Center receives a previously requested event history list. In this case, the Control Center may decide to request visual data regarding any event in this list by transmitting a visual data request.
   - ***Low-quality frame reception*** – If the Control Center receives a low-quality frame from a given event, it should decide if another frame is necessary to clearly understand what is happening on the vessel. If no further frames are needed, the Control Center has the possibility of either erase or erase-protect the visual data in the vessel memory by sending appropriate requests to the visual data storage management module at the vessel.
   - ***Medium-quality frame reception –*** The Control Center may also receive medium-quality frames following a previous request or reception of a low-quality frame; also in this case, the Control Center may request the erasure or protection of this content by sending an appropriate request to the visual data storage management module at the vessel.
4. **Request transmission** – The Control Center can transmit requests to the VMS unit at the vessel, notably:
   - ***Live visual data request*** – A message is sent to the VMS unit requesting it to transmit as soon as possible the visual data associated to what it is happening at that precise moment, independently of the relevance ranking.
   - ***Event history request*** – A request is sent to the VMS unit at the vessel requesting the event history, i.e., a list of relevant past events.
   - ***Medium-quality frame request –*** After receiving a low quality frame from a specific event, the Control Center may request for a medium quality frame from that specific event (distanced from the first frame by a certain amount of time).
   - ***Event's frame request*** – After a relevance warning (without transmitted visual data), the Control Center may ask for a frame from that specific event; this frame will be transmitted in low quality. The same happens when the Control Center asks for the transmission of a visual frame from the event history.
   - ***Erasure/protect request*** – A request to the VMS unit is issued to definitely erase or permanently protect from erasure some specific visual footage which is defined using a specific time period.
   - ***Event time-lapse request –*** A request is sent to the VMS unit requesting a time-lapse of a given past event.

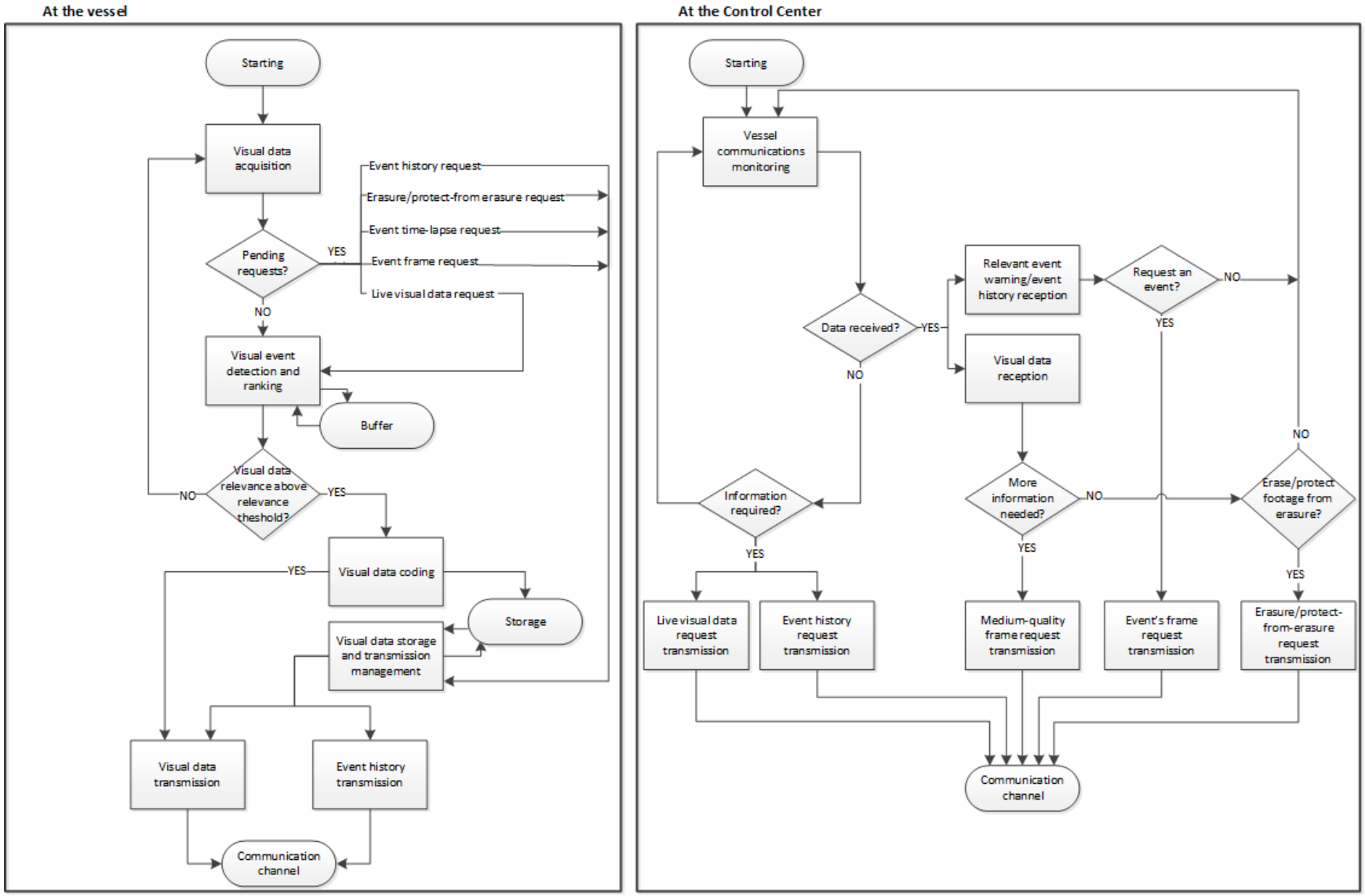


Figure 30 – System operation detailed flowchart.

3.3.2 Requirements

To properly address the user's needs, the solution must fulfill a number of identified requirements. To speed-up future referencing and for organization purposes, a code will be defined for each requirement, as recommended by the European Space Agency (ESA) [107]. The code adopts the following format: REQ.[CAT].[SEQ], where:

- *REQ* is the requirement label;
- *CAT* is a code associated to the category (group) of the requirement;
- *SEQ* is a two-digit number which guarantees a unique requirement identifier.

In the following, the requirements will be categorized into three sets of requirements.

1. **Visual data processing requirements**

   This set of requirements refers to all the visual data processing related needs and refers both to the sending and receiving sides; important issues related to the bandwidth, processing power and cost limitations are addressed here. As video transmission was excluded due to the involved bandwidth, visual data will mostly refers to images.

REQ.VIS.10	**Royalty free image codec** The adopted image codec must be royalty free.
REQ.VIS.20	**High image compression efficiency** Given the very narrow bandwidth and the air-time costs of the satellite channels, the image data to be transmitted shall be compressed as much as possible.
REQ.VIS.30	**Two quality storage** The solution shall encode the relevant visual data with two different qualities, thus providing the possibility of transmitting a low-quality frame and later, if deemed necessary, a medium quality one.
REQ.VIS.40	**Video-based event detection** The system should be able to identify and score the importance of detected events based only on the available video data.
REQ.VIS.50	**Storage visual data resolution** Footage shall be stored with the maximum resolution available, this means, in principle, the resolution made available by the camera.
REQ.VIS.60	**Low computational complexity** Required processing operations at the VMS Unit should have as low as possible computational complexity in order not to compromise its regular operation, namely its response time and power consumption.
REQ.VIS.70	**Image post-processing** An image post-processing tool should be used at the receiving side (Control Center) to process the decoded images and provide the users with an improved visual experience (e.g. frame interpolation and denoising).

2. Non-visual data related VMS unit requirements

These requirements refer to other VMS unit functionalities, non-directly visual data related, which should be also fulfilled for the correct operation of the system.

REQ.MBU.10 **Event visual data recording**
The VMS unit shall record the visual data associated to a given event every time the event detection tool classifies an event as minimally relevant.

REQ.MBU.20 **Critical event transmission**
The VMS unit shall automatically transmit a visual footage frame associated to an event which relevance rating is above a certain predefined threshold.

REQ.MBU.30 **Normal event transmission**
The VMS unit shall have the capability to process and transmit visual footage frames requested by the Control Centre.

REQ.MBU.40 **Event history**
The event detection tool shall keep a list with the history of stored events and their relevance ratings. This will allow the Control Center to select and request specific past events footage.

REQ.MBU.50 **Event history rollover**
If the visual data storage capacity is full, the visual data associated to new events shall overwrite previous ones, starting by the oldest ones.

3. Non-visual data related Control Center requirements

These requirements are related to functionalities to be offered at the Control Center to fulfill the needs for a correct operation of the system.

REQ.CC.10 **Event history request**
The Control Center shall have the possibility of requesting the event history to a given VMS unit.

REQ.CC.20 **Event visual data request**
The Control Centre shall have the possibility of requesting visual data related to a certain selected event from the event history, with a selectable quality level.

REQ.CC.30 **Additional visual data frames request**
The Control Center should have the possibility of requesting additional visual data frames distanced by some time from the previously received frames and with higher quality.

REQ.CC.40 **Critical event non-erasure control**
The Control Center shall be able to indicate the VMS unit that the visual data of a specific event must not be erased.

REQ.CC.50 **Visual footage erasure**
The Control Center shall be able to indicate the VMS unit to erase some specific visual footage.

REQ.CC.60 **Live visual data request**
The Control Center shall have the possibility of requesting the VMS unit to transmit the live visual data as soon as possible so that it can see what is momentarily happening at the vessel.

Chapter 4

4. Fishing Activity Detection and Classification Solution

Chapter 3 has described the importance of Vessel Monitoring Systems to control the state of global fishing stocks, which are already in shortage conditions. Beside the current monitoring solutions, adding a video camera to fishing vessels may lead to very significant improvements of the efficacy of the vessels monitoring process in terms of fishing activities, notably due of the difficulty to detect fishing activities using only geo-positioning data.

In this context, this chapter proposes an automatic solution for the detection and classification of fishing activities based on the processing of real-time video information acquired with a camera well positioned in the vessel. This automatic detection and classification solution must not only detect fishing events with varying relevance but should also choose well representative video frames to be sent to the Control Center in order further human analysis is made almost in real-time. The following subsections will present the designed solution as well as its performance assessment and analysis.

4.1. Video Dataset and Ground Truth Definition

Naturally, the design and assessment of a fishing detection and classification solution can only be made if appropriate and representative test material is available. The presentation of the video dataset and the definition of the fishing activities ground truth are the main targets of this section.

4.1.1 Video Acquisition

The Instituto Português do Mar e da Atmosfera (IPMA) [108] often performs sea expeditions with the goal of assessing the state of the ocean's resources, among other purposes. In one of these expeditions, IPMA has kindly allowed the installation of a video camera inside *Noruega*, a vessel used in bio-oceanography and fisheries campaigns, therefore making possible the recording of video data regarding the vessel activity, notably with fishing purposes. The expedition used a fishing method called *trawling* [109] to assess the state of fishery stocks. Trawling is a fishing method that involves pulling a fishing net through the water behind one or more boats. The net that is used for trawling is called a *trawl*, see Figure 31. The expedition where the test video dataset was collected lasted one month, and the crew has recorded the events that they deemed relevant, including fishing and non-fishing activities.

Figure 31 – (a) and (b) *Noruega* fishing vessel; (c) Trawling.

The video camera installed in *Noruega* was placed in a position where it should easily enable the capture of any fishing activity, notably pointing at the back of the vessel, where the fishing net goes in and out of the boat, see Figure 32. The video camera has the following properties [110]:

- **Model** – HikVision VFIR-Bullet Network Camera
- **Spatial resolution** – 2688x1520 pixels
- **Chroma subsampling** – 4:2:0 YUV
- **Frame rate** – 20 frames per second
- **Video compression** – H.264/AVC Main profile compliant video codec

Figure 32 – (a) Camera installation; (b) Boat area seen by the camera.

4.1.2 Video Dataset Characterization

The acquisition of the visual footage has resulted in a dataset with 242 videos. The videos have an average duration of 30 minutes. Although some videos are sequential, i.e., one ends where the following starts, all videos are treated in an independent manner. In each video, the footage can be either *relevant* or *non-relevant*, depending if fishing activities are happening or not. The exact definition of what a fishing activity is, is to be specified later when defining the ground truth for the labelling of the training data.

Each video may or may not contain a relevant event and there is, at most, one relevant event somewhere within each video, as exemplified in Figure 33, whether it is at the beginning, middle or end of the video (this happened to be the way the videos were recorded).

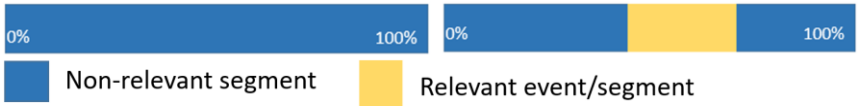


Figure 33 – Two video examples: one without relevant segments/events and another with a single relevant segment/event.

Since the reader might not have the chance to see the actual video footage, a brief description is presented below, backed by screenshots to make it easier to understand:

1. Most footage shows an empty deck, with nothing really relevant happening.
2. From time to time, one or more crew members appear to perform non-relevant activities, usually having a conversation, to do some activity such as sweep the floor or do some preparation for an eventual fishing activity, such as placing the net on the floor after being pulled (see Figure 34).
3. There are two major visually relevant fishing actions, which are described below:
    - **Throwing the fishing net into the water:** The fishing net is already placed on the floor. At some stage, the green door at the back of the boat opens, see Figure 35 (b), enabling the fishing net to start slowly falling into the ocean. Eventually, with some help of the crew, the net is swallowed by the sea, finally stopping being visible.
    - **Pulling the fishing net from the water:** The net pulling process requires some preparation, notably the crew moves to the deck to perform a few necessary actions. The several stages of pulling the fishing net are represented in Figure 36. The fishing net, initially underwater, is pulled continuously until its end reaches the deck. If loaded with a large amount of fish, its content is placed on the deck and then returned to the water (as represented in Figure 36 (d), (e) and (f)) since these scientific expeditions do not really target fishing. However, in case of a small amount of caught fish, the fishing net is taken to the side of the vessel (not shown in the footage) and its content is directly put back into the water (see Figure 36 (g), (h) and (i)).

Figure 34 – Video frames with no fishing activities: (a) Crew just standing on the deck; (b) Crew sweeping the deck; (c) Placement of the net on the floor.

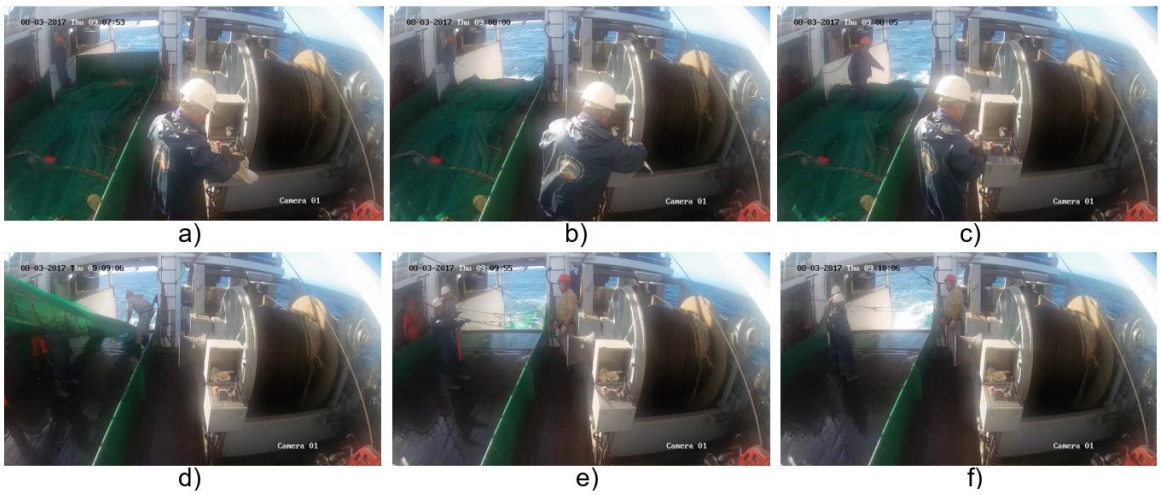


Figure 35 – Stages of throwing the fishing net: (a) nothing particular is happening; (b) the green door allowing the fishing net to go into the sea is opened; (c) the net starts to be slowly thrown into the sea; (d) the net is halfway into the sea; (e) the net is still visible; (f) the net stops being visible as fully under water.

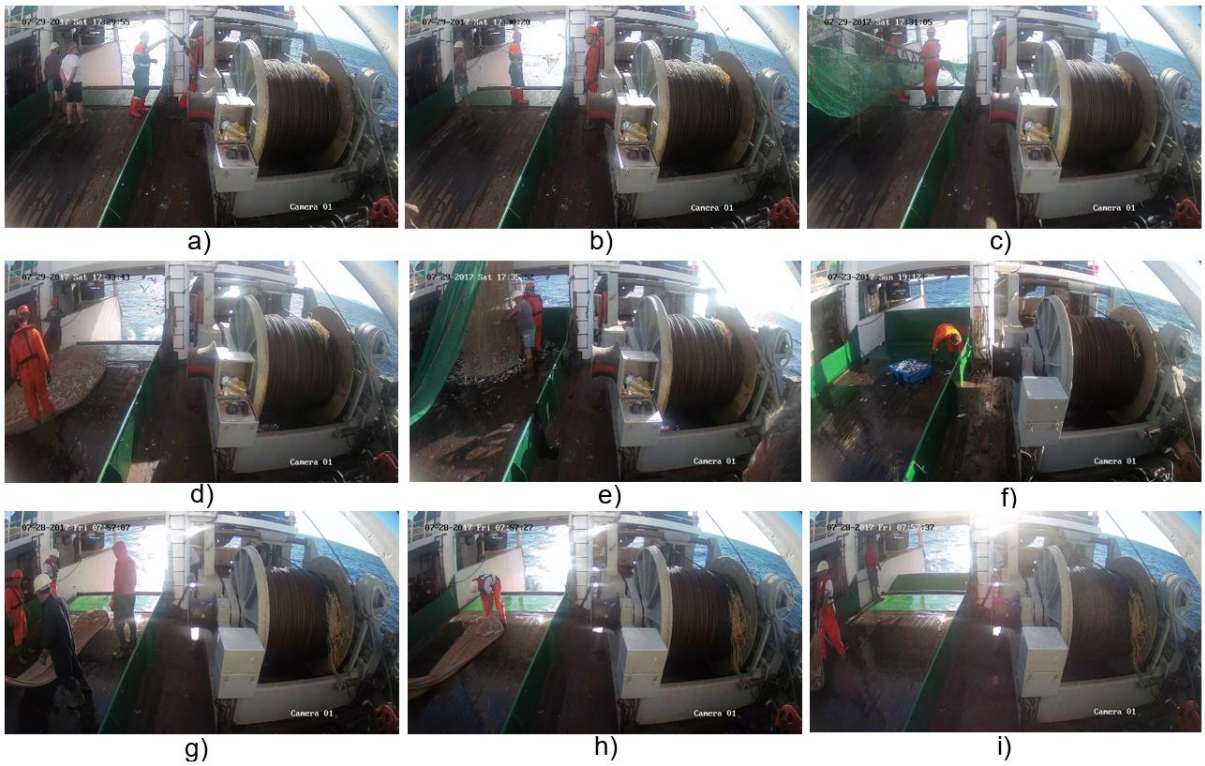


Figure 36 – Stages of pulling the fishing net: Two sequences with different endings are represented: (a)-(b)-(c)-(d)-(e)-(f), for the situation when a large amount of fish is caught and (a)-(b)-(c)-(g)-(h)-(i), for the situation when a small amount of fish is caught. The fishing net: (a) is still not visible; (b) starts to be visible; (c) is visibly pulled; (d) stops being pulled and is full of (visible) fish; (e) is emptied and fish put on the floor; (d) is not visible and some fish is put back into the sea. In case of a small amount of caught fish, the fishing net: (g) stops being pulled and has a small amount of fish; (h) is being opened to take out the fish; (i) is taken to the side of the vessel.

4.1.3 Ground Truth Definition

The videos in the dataset include events that change throughout a given video. As mentioned above, the objective of the automatic fishing detection and classification system to be developed implies defining in a rather precise way what means precisely ‘fishing’ in this context in order some video ground truth may be defined for the training and performance assessment when testing the designed solution. This implies classifying video segments (and thus their frames) as relevant or non-relevant in terms of fishing activities (Figure 33 exemplifies this well). The non-relevant fishing periods will include not only the non-relevant fishing events but also the periods where no events are happening. In this context, the precise boundaries (at frame level) separating a relevant event/period from a non-relevant one are not very critical as the fishing activity does not have a sudden start but rather evolves during a certain period of time where the precise starting and stopping boundaries are somehow difficult to define. But the focus of this work is to detect the fishing event which lasts at least a few minutes and thus the frame precise definition of its starting and ending is not critical.

In practice, the fishing activity comprises three major parts, notably throwing the fishing net into the water; having the fishing net under the water; and pulling the fishing net out of the water. Naturally, from an automatic visual detection point of view, the fishing net throwing and pulling are the critical events as they correspond to the parts where there is relevant visual activity shoot by the video camera that is critically useful for the automatic detection of the fishing activities.

In the process of defining the fishing activity ground truth, the definition of event boundaries, this means the starting and ending of throwing in and pulling out the net from the sea are critically important as they define in this work the precise meaning of 'fishing activity'. Between these two events (when the net is underwater as part of the fishing activity) will not be classified as relevant as there is nothing that can visually distinguish this period from no fishing activity.

As it was previously mentioned, the ground truth will be based on two classes only: relevant fishing events and non-relevant events (including also the event-free periods). After analyzing the dataset and discussing with *Xsealence*, the company which brings the practical requirements to this Thesis, it was decided to define as relevant the following events:

- **Throwing the net into the sea** – The start of this task is here associated to actions performed with the net targeting its launching into the sea. The throwing net task is considered finished when the net is no longer visible, as fully under water. Using Figure 35 as an example, (a) and (b) are considered non-relevant, since the fishing net is not yet being thrown into the sea. From (c) to (e), the net is clearly being thrown into the sea, and thus the frames are considered relevant. In (f), the fishing net stops being visible, which implies that the frames are not relevant anymore. All the frames between the start and the end of the throwing process are by definition defined as relevant even if at a specific time the net and fish are not visible, e.g. because something, notably a fisherman, is in front of the camera.

Pulling the net from the sea – The start of this task is here associated to the visibility of the fishing net on the boat deck and thus the pulling out start corresponds to the first frame where the fishing net is minimally visible. The pulling net task is considered finished when the net is totally removed from the boat deck, together with any fish. Using Figure 36 as an example, (a) is considered non-relevant as the net is not yet visible while (b) is relevant as the net starts being visible. From (b) onwards, there is a relevant event happening, including (f) where there is still fish on the deck and (i) which shows the net with some fish being taken out to the side of the vessel. When all the fish has been removed, and the net is not visible anymore, the relevant event ends and thus the frames are not relevant anymore. Again, all the frames between the start and the end of the pulling process are by definition defined as relevant even if at a specific time the net and fish are not visible, e.g. because something, notably a fisherman, is in front of the camera.

All other parts of the video, including the time the net is underwater, are defined as non-relevant. Considering this definition of ground truth, the obtained dataset was characterized, organized and classified using an Excel table, for easier understanding of its features. For each video, a specific ID (video stream number) was given, together with the shooting date and time, and duration; moreover, for each video, the fishing relevant parts were characterized, notably in terms of duration and number of frames. A column with observations was also added for relevant information regarding the activities occurring in the video. An excerpt of this table is presented in Table 3. To have a good overview of the complete dataset, some global statistics are presented in Table 4, where it may be seen that the global percentage of relevant frames is slightly below 4%.

Table 3 – Video characteristics table excerpt.

# video	Date&time	Video duration [hh:mm:ss]	Number of frames (per video)	Relevant events' duration (per video) [hh:mm:ss]	Number of relevant frames (per video)	% of relevant frames (per video)	Observations
1	08/07/2017 06h58	00:23:49	28 589	00:00:00	0	0,00%	
2	08/07/2017 10h08	00:28:28	34 150	00:00:00	0	0,00%	
3	08/07/2017 10h36	00:28:28	34 150	00:00:00	0	0,00%	
⋮							
237	03/08/2017 16h22	00:28:28	34 150	00:02:40	3 200	9,37%	net thrown into the sea;
238	03/08/2017 16h33	00:28:28	34 150	00:00:00	0	0,00%	
239	03/08/2017 17h02	00:35:07	42 132	00:06:16	7 520	17,85%	net pulled from the sea; net being placed on the deck;
240	03/08/2017 18h50	00:28:28	34 150	00:00:00	0	0,00%	little or no activity;
241	03/08/2017 19h19	00:28:27	34 140	00:00:00	0	0,00%	
242	03/08/2017 19h47	00:15:18	18 369	00:04:38	5 560	30,27%	net pulled from the sea;
TOTAL		**120:55:54**	**8 707 084**	**04:38:15**	**333 900**	**3,83%**	

Table 4 – Overall video dataset statistics.

Total number of videos	242
Number of videos without relevant events	176
Number of videos with relevant events	66
Total video duration [hh:mm:ss]	120:55:54
Relevant events' duration [hh:mm:ss]	04:38:15
% of relevant frames (total)	3,83%
Average video duration [hh:mm:ss]	00:29:59
Average % of relevant frames in videos with relevant frames	14,87%

4.2. Architecture and Walkthrough

In Sub-section 3.4.1, the overall architecture of the system operation was presented. The key module in this architecture is the Fishing Activity Detection and Classification engine which is responsible for the core functionality of this system, this means the detection and classification of fishing events. In this module, the captured visual data is analyzed to assess if a fishing event is taking place or not at some stage. In this section, the architecture and walkthrough of this engine is presented, see Figure 37, while the key module, this means the Frame-level event detection module, is presented in detail in the next sub-section.

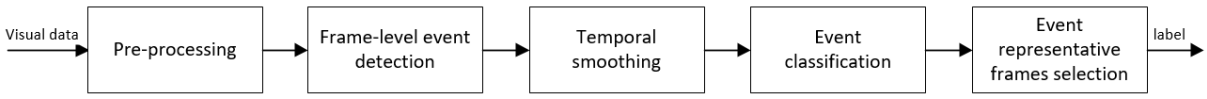


Figure 37 – Fishing Activity Detection and Classification architecture.

The overall walkthrough for the Fishing Activity Detection and Classification architecture is as follows:

1. **Pre-processing** – Since the next Frame-level event detection module, which is based on deep learning, requires the input data to have certain characteristics, this module has the main task to pre-process the acquired data to convert it to the appropriate data for the next module. In this context, this module is responsible for applying the necessary processing actions to assure that the following conditions are met:
    - **Cropping** – As the next module requires images with a square resolution, this module crops each frame to a square. By analyzing the fishing dataset, it was observed that nothing

important ever happens on the right part of the image, which corresponds to a less relevant part of the boat; hence, the cropping has been made to keep the left part of the image where the key events take place. Since the videos in the fishing dataset have a spatial resolution of 2688x1520 pixels, the output is a square image with a resolution of 1520 x1520 pixels.

- **Resizing** – As the following module accepts only images with a spatial resolution of 227x227 pixels, the OpenCV resize function with the INTER_AREA interpolation flag [111] has been applied to down-sample the input 1520 x1520 pixel images to 227x227 pixel images.
- **Mean subtraction** – To process zero-mean data, which may have some advantages for processing, the 0-255 (8-bit) valued images are mean subtracted using the mean computed for each channel/component across all images from the dataset.

2. **Frame-level event detection** – This module is responsible for individually processing each frame captured by the camera and, on a frame-by-frame basis, computing the probability of a fishing event taking place at that moment. This is achieved using a Deep Learning based approach, more precisely, a Convolutional Neural Network (CNN). This type of solution was chosen as it has been recently proved to provide excellent performance in terms of image detection, classification and recognition related tasks.

3. **Temporal smoothing** – As the output of the previous module results from a frame-by-frame approach, and thus does not take into account the temporal evolution by using past and future frames, it may happen that these probabilities show rapid and large fluctuations, which is undesirable in terms of event detection and classification. Thus, a temporal filtering operation is performed to smooth the probability-based detection output of the previous module, thus allowing to obtain a more stable and accurate classification, since the fishing events cannot successively change at frame level. This module corresponds basically to a temporal low-pass filter that operates based on a sliding window that slides, frame by frame, thus smoothing the already available frame-level probabilities, averaging them to obtain a smoothed probability for the central frames of each sliding window according to:

$$P_i = \sum_{i=-n/2}^{n/2} p_i \tag{4.1}$$

where p_i represents the probability of a fishing event taking place in a specific frame as output by the previous Frame-level event detection module and P_i is the smoothed frame-level event probability corresponding to the average of the probabilities p_i for a specific sliding window size, $Sliding\_window\_size$, measured in seconds. Figure 38 depicts an example of temporal filtering with $Sliding\_window\_size = 7/framerate$, where $framerate$ is the framerate of a given video.

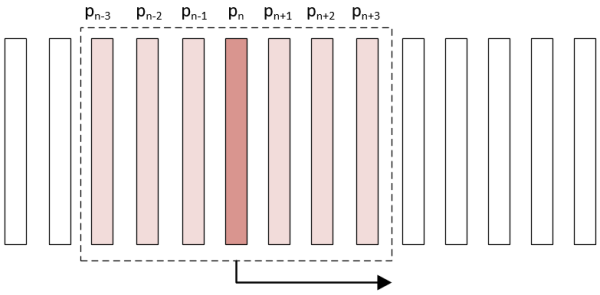


Figure 38 – Temporal filtering example for $Sliding\_window\_size = 7/framerate$.

To perform the sliding window averaging using the past and future neighboring frames, knowledge regarding future frames is necessary and thus a delay of half the size of the sliding window is involved. This delay is not critical for this application as the scale of this delay is typically much smaller than the scale of the events themselves. The performance of the proposed solution will be assessed in Section 4.5 considering several sliding window sizes in order to determine the optimal size.

4. **Event classification –** The goal of this module is to classify the content in terms of fishing events after assessing whether a relevant event, this means fishing activity, is happening or not, naturally based on the output of the Temporal smoothing module.

   The event classification process where relevant fishing events are detected and classified is based on the so-called *Event Detection Ratio* (*EDR*), which is the ratio between the so-called *Area Under the Curve* (AUC), where the curve is here the filtered frame-level event probabilities within a given time interval (portraited in blue in Figure 39) and the maximum possible area in that same time interval (portraited in grey in Figure 39). Thus, EDR comes:

$$EDR = \frac{AUC}{max\_area} \tag{4.2}$$

   As this module continuously receives filtered frame-level event probabilities, EDR is successively computed for each frame. At some point, EDR becomes larger than a pre-defined event threshold, the *event_threshold,* and thus the event starts to be considered a candidate for a relevant event; when EDR is back to less than *event_threshold*, the current candidate event is considered finished and finally classified as a *relevant event* if longer than a pre-defined minimum length. This minimum length shall be called *minimum_relevant_event_duration* and has been chosen to be of 1 minute, as all relevant events have a duration of more than 1 minute. Take for example $event\_threshold = 0.75$; from Figure 39 (c) to (e), the current event would be considered relevant if long enough.

   In short, the algorithm works as follows:

   1. For each newly received frame, compute the *event detection ratio (*EDR*)* as defined in equation 4.2 using a specific time interval, *event_time_window*.
   2. While $EDR > event\_threshold$, the current time period is classified as a candidate event.
   3. If the event is longer than 1 minute, then the event is finally classified as *relevant*. An event that the algorithm classifies as relevant shall be called a *detected event*.

Figure 39 shows several examples of the event classification process, notably the proposed algorithm continuously receiving filtered frame-level event probabilities and deciding on events based on them.

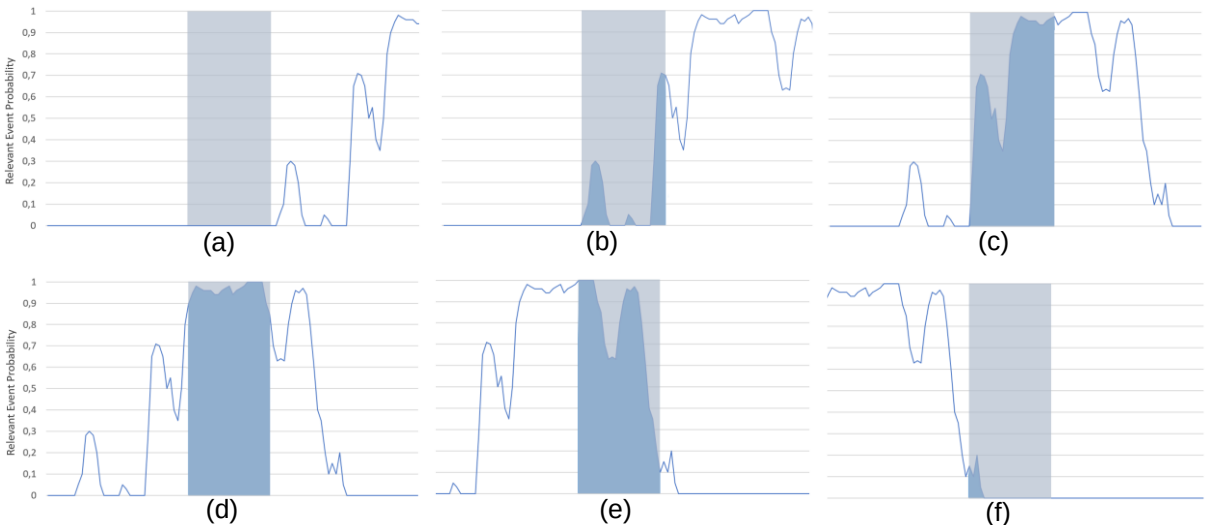


Figure 39 – Event classification on the fly: as more filtered frame-level event probabilities are received the *Event Detection Ratio* (*EDR*) is computed for a specific time interval: (a) $EDR = 0.0$; (b) $EDR = 0.16$; (c) $EDR = 0.76$; (d) $EDR = 0.96$; (e) $EDR = 0.75$; (f) $EDR = 0.02$.

The exact moment/frame at which the algorithm defines the starting and ending detected event points depends also on the defined $event\_time\_window$ as follows:

- ***Starting point*** – The first moment the condition $EDR > event\_threshold$ is valid, the algorithm defines as the event starting point, labeled as *event_time_window* leftmost frame.
- ***Ending point*** – The first moment the condition $EDR > event\_threshold$ is no longer valid, the algorithms defines as ending point, labeled as *event_time_window* rightmost frame.

An example of the frame level definition of a relevant event starting and ending points is represented in Figure 40 for $event\_threshold = 0.75$.

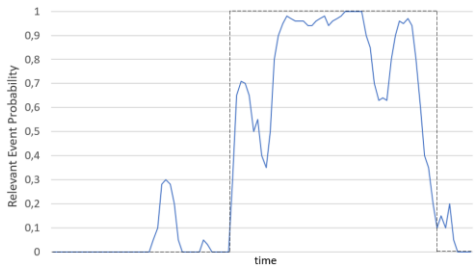


Figure 40 – Overlapping of the input and output of the event classification module, for $event\_threshold = 0.75$. The blue line represents the filtered frame-level event probabilities while the dashed, black line, represents the output of the event classification module, where are clear the starting and ending event moments.

The optimal values for *event_threshold* and *event_time_window* (the time interval for which AUC and *max_area* are computed) will be determined in Section 4.5.3, when assessing the performance of the proposed solution.

5. **Event representative frames selection** – After detecting a relevant event, it is necessary to identify which frames represent the event better to send at least one to the Control Center (initially, only one is sent; more may follow, if requested).

   The representative frames are then chosen as follows:

   - Once a relevant event has been detected, the frame with the highest filtered probability within the last interval corresponding to half of the *event_time_window* parameter is selected.
   - In the likely scenario of several frames having the same probability, the most recent one is chosen.
   - Other representative frames are randomly chosen only after the event has finished, with the constraint of being separated in time by 20% of the total detected event duration from other representative frames.

4.3. Frame-level Fishing Event Detection

As mentioned above, this is the key module in the overall architecture with the objective to detect fishing events at frame level. For the past few years, deep learning based solutions have provided very significant leaps and determined the state-of-the-art for detection, classification and recognition tasks, notably surpassing by far the performance of previous technical approaches and solutions. This is especially true for Deep Convolutional Neural Networks (CNN), which have revolutionized image classification (as well as detection and segmentation), sometimes even surpassing the human accuracy [112] [113].

The success achieved by CNNs in image classification has inspired the use of deep CNN to also improve video classification performance. Although video classification may imply the exploitation of features involving time, it has been shown that this temporal exploitation might not play such an important role and video may be also treated as a sequence of frames. For example, in [114] several approaches to video classification are analyzed to conclude that the best approach is one using information across several frames, this means exploiting the correlation in time, which achieves a classification accuracy of 60.9% on top-1 predictions. However, there is also a single-frame approach, which consists in simply forward-passing a frame through a regular CNN architecture, thus not exploiting temporal information, which achieves an accuracy of 59.3% on top-1 accuracy predictions. The very small difference between the two performances suggests that motion might not play such a significant role, at least for some classification tasks. After analyzing our fishing detection problem, it was concluded that this may be the case for our problem as the key information is not related to the temporal evolution of the moving elements in the scene but rather of the spatial positions of key elements like the fishing net and the boat doors. Based on this analysis, it was decided to adopt a single-frame approach to initially attribute a fishing event detection probability to each frame. For this, an available CNN classification architecture needs to be chosen as accomplished in the next section.

4.3.1 CNN Architecture Selection

While current state-of-the-art CNN solutions achieve exceptionally good results, these are often very computationally expensive. This feature does not go well with a low complexity requirement as it is the

case for the solution targeted in this Thesis. To address this requirement, a comparison of well-known CNNs in terms of computational complexity (forward-pass speed) and accuracy is presented in Table 5, using information from [115]. For a set of relevant CNN, Table 5 shows the number of layers involved, the top-5 error as accuracy metric and the task running time as complexity metric. The running times were obtained by forward-passing a minibatch of 16 images on a machine with a Pascal Titan X GPU, an Intel Core i5-6500 CPU and 16GB RAM running Ubuntu 16.04 with the CUDA 8.0 release and using the Torch deep learning framework [116]. The top-5 error scores, corresponding to the percentage of times that the top-5 predictions of a CNN fail to contain the right class, were taken from published papers associated to each CNN when using single-cropped images (usually several crops of the same image are fed to CNNs to improve their performance), except for VGG-19 which instead is applied over 256x256 test images as described in [117].

Table 5 – CNN benchmarking [115].

CNN	Layers	Top-5 error	Task running time (ms)
AlexNet [118]	8	18.20%	5.04
Inception-V1 [119]	22	10.07%	12.06
VGG-19 [117]	19	9.00%	55.75
ResNet-152 [113]	152	4.49%	75.45

From Table 5, it is possible to conclude that the CNN architecture offering lower computational complexity is AlexNet [118]. This was expected since its architecture is much simpler than the studied alternatives as it includes only 8 layers. Although achieving a lower accuracy performance, AlexNet can still do a remarkable job in image classification. In conclusion, AlexNet is the CNN architecture selected for the detection and classification solution to be developed in this Thesis.

4.3.2 AlexNet Architecture Overview

Now that a specific CNN architecture has been chosen, it is time to provide some insight about it, starting with the architecture which is presented in Figure 41.

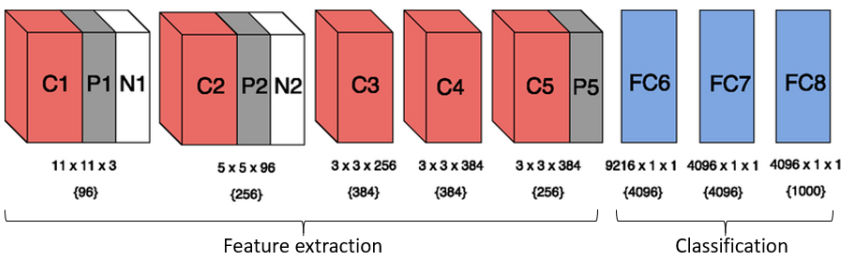


Figure 41 – AlexNet architecture: C means convolutional layer; P means pooling layer; N means local response normalization layer; FC means fully connected layer.

As shown in Figure 41, AlexNet includes several types of layers, notably convolutional, pooling, local response normalization and fully connected layers. Although not depicted in Figure 41, an activation layer (composed by Rectified Linear Units [120]) is also present after every convolutional and fully

connected layers. Also, pooling, local response normalization and activation layers perform fixed operations, which means that only the convolutional and fully connected layers have trainable parameters, which is the reason why this network is said to have only 8 layers, notably 5 convolutional and 3 fully connected layers.

A detailed explanation of the role of each type of layer has been presented in Chapter 2. The exception is the local response normalization layer which was believed to aid the generalization of the network but whose contribution has been shown to be minimal and thus, less and less used [121].

As all CNNs, the AlexNet network is divided into two main parts: the feature extraction part and the classification part. A walkthrough on both of these parts is presented below:

- **Feature extraction part** – Essentially composed by convolutional layers, this part is responsible for extracting the most relevant image features. These layers are usually followed by other layers such as pooling layers and activation layers, which improve the network operation. AlexNet, in particular, also uses local response normalization layers. As explained in Chapter 2, the feature extraction part works by extracting low-level features in its first layers; as the network progresses, filters are applied to the output of previous layers, therefore progressively extracting higher-level features. An explanation of its operation, based on Figure 41, is given below:
    - ***C1, P1, N1*** – An image with a spatial resolution of 227x227x3 (depth 3 corresponding to R, G and B image channels/components) is fed to the network. The first convolutional layer which is comprised of 96 filters with a shape of 11x11x3 convolves the image across its channels with a stride of 4, therefore resulting in an output with a 55x55x96 shape. The output of this layer is then max-pooled to reduce the amount of parameters and computation of the network and also control overfitting, with 3x3 filters with a stride of 2, resulting in an output with a shape of 27x27x96 and then normalized.
    - ***C2, P2, N2*** – This layer includes 256 5x5x96 filters with stride 1. After normalization and pooling, the output as a shape of 13x13x256.
    - ***C3*** – This layer includes 384 3x3x256 filters with stride 1, leading to an output with a shape of 13x13x384.
    - ***C4*** – This layer includes 384 3x3x384 filters with stride 1, leading to an output with a shape of 13x13x384.
    - ***C5, P5*** – This layer includes 256 3x3x384 filters with stride 1, leading to an output with a shape of 13x13x256. The pooling layer operation results in an output with a shape of 6x6x256.
- **Classification part** – This part is essentially composed by fully connected layers and it is responsible for combining the extracted features to model more complex patterns. An explanation of its operation, based on Figure 41, is given below:
    - ***FC6*** – This layer includes 4096 neurons, each one connected to each “pixel” of the output of the feature extraction last layer (6x6x256 = 9216 “pixels”).
    - ***FC7*** – This layer includes 4096 neurons, each one connected to all the neurons from the previous layer.

- ***FC8*** – This layer is composed by a number of neurons corresponding to the number of classes, $n\_classes$, that the network has been trained for, where each neuron is connected to all neurons from the previous layer. The output of this layer is then fed to a softmax function [122], which "squashes" the output of the network's last layer into an n-dimensional array that add up to one, and thus where each element contains the probability of the original image belonging to the class represented by that very element.

It is interesting to note that out of the 60 million parameters to be learned in this network (from filters in the convolutional layers and connections in the fully connected layers), each convolutional layer contains no more than 1% of these parameters [118], meaning that the vast majority is present in the classification part.

4.3.3 AlexNet Training

This process of training requires many *different* images for it to converge and generalize properly. However, our available dataset is rather limited, as it was taken from a single boat and thus the features present are always the same. In fact, very few people train an entire CNN from scratch, simply because the availability of datasets large enough to provide proper generalization is scarce [123]. For example, AlexNet was originally trained using the Imagenet database containing 1.2 million images spread over 1000 categories. One way to solve this problem is to use transfer learning, where a network trained on a very large dataset is used for a slightly different task by re-training and thus fine-tuning some layers using some additional content.

To determine which layers should be fine-tuned in order to obtain the best classification performance, the insights from [114] have been used. In this paper, the authors experimented using transfer learning to fine-tune their network (which is rather similar to AlexNet, layer-wise) with a smaller dataset than the one originally used for training. Through experimenting, it was found that the fine-tuned network achieved better results when the three fully connected layers were fine-tuned, as opposed to fine-tuning all layers or only the last fully connected layer. Because of this, it was decided that the three fully connected layers in AlexNet will be fine-tuned. This corresponds in practice, in training the classification part of the network, somehow assuming that the extracted features are appropriate even if the classification task changes slightly. However, the classification layers have to be re-trained as the labels are different and thus the extracted features have to be combined in a different way.

When training a CNN, there are a number of parameters and choices to be taken into account, notably:

- **Deep Learning framework** – The framework should allow to easily build and implement different types of machine learning algorithms, such as neural networks, in a simplified way. Due to high community adoption, very good documentation and good performance, TensorFlow was the chosen framework.
- **Pre-trained weights** – The pre-trained weights to initialize the AlexNet were taken from Model Zoo [124] and converted with *Caffe to Tensorflow* [125]. These weights were originally obtained when training using the ImageNet 2012 dataset.

- **Number of classes** – For the purposes of this Thesis, the network was trained for two classes: relevant and non-relevant. Hence, when a video-frame is forward-passed through the network, the output will be the probability of the frame belonging to a relevant event and the probability of the frame belonging to a non-relevant period. The latter will not be considered in the following.
- **Dropout probability** – The 'dropout probability' expresses the probability of setting to zero the output of a neuron in a fully connected layer, meaning that the dropped neuron will not contribute to neither the forward pass nor the backpropagation [118]. This forces the network to learn more robust features. The dropout probability used was 0.5, as used in the original AlexNet training.
- **Batch size** – The batch size is the number of training images that the CNN will forward-pass before the network updates its parameters (through backpropagation). The batch size used was 128, as used in the original AlexNet training.
- **Loss function** – To measure the loss at the output of the network the cross entropy loss function is used, in order to quantify the difference between probability distributions [126] at the output of the network.
- **Learning rate** – This rate refers to how large are the steps taken that will eventually lead the network parameters to converge. Too high learning rates prevent the network from converging at all, while too small learning rates slow down the training process. When fine-tuning, it is common to use smaller learning rates, since it is not a good idea to distort too substantially the pre-trained weights [123]. Originally, AlexNet was trained initially with a learning rate of 0.01. However, since AlexNet is being fine-tuned, a learning rate of 0.001 was used, which was found to provide a good training convergence.
- **Number of epochs** – Training occurred over 10 epochs, this means the number of times that the full training set was forwarded through AlexNet.

Now that the proposed solution has been described it is time to assess its performance, which will be detailed in the next section.

4.4. Performance Assessment

This section targets to assess the performance of the fishing events detection and classification solution proposed in the previous sections. To achieve this objective, the process of defining the material used for training and testing will be presented as well as the adopted performance metrics. After assessing the solution performance, a summary on its advantages and limitations will be given.

4.4.1 Training and Test Material

In Section 4.2, the fishing activity dataset has been described. It was seen that this dataset is comprised of 242 videos with an average duration of 30 minutes each, where 176 of them do not contain any relevant events and 66 videos do. Only a small portion of each video with relevant events is, in fact, a relevant event, and no video has more than one relevant event.

When training a CNN, as in machine learning in general, the trained model is required to be able to generalize well the learned knowledge, i.e., actually learning from data and not just memorizing it, what

is commonly known as *overfitting* [127]. To check whether the model is generalizing well, one should test the model with data never seen by it. Hence, instead of using the whole dataset for training, it is necessary to split the dataset in, at least, two different sets: a *training set*, whose data will be used for training (usually composed by the largest portion of the available dataset); and a *test set*, whose data will be used to check if the model is behaving properly, notably detecting the fishing events with high accuracy.

Since the AlexNet deals only with images, the training material must be composed of labelled video frames, notably frames which can 'teach a lot' in terms of fishing events. It is then necessary to first divide the videos from the dataset between the training set and the test set, and only then, extract from the training test the frames to be used to re-train the last three AlexNet layers as proposed in the previous section. However, the test videos will be fully processed since full videos with and without relevant events are needed to assess the event detection capabilities of the proposed solution.

Hence, selecting the training and test material involves two major steps:

1. **Dividing the overall dataset into training and test sets** – A division of 80% of the total video duration for training and 20% for testing was found to be appropriate considering the recommendations in the literature. Since the fishing activity dataset videos have time durations that can differ substantially, it is not possible to simply divide the full dataset on a video basis since the total video length split would not match the target percentages. Hence, the video division between the training and test sets was made as follows:
    a. Divide the overall dataset into two pools: *relevant pool* which contains videos with relevant events, and *non-relevant pool*, which contains non-relevant footage only.
    b. Randomly select a video from the *relevant pool* and put it in the training set until the training set contains 80% of the total number of relevant event frames; the remaining relevant videos are put in the test set.
    c. Randomly select a video from the *non-relevant pool* and put it in the training set until the training set contains 80% of the total number of non-relevant frames; the remaining non-relevant videos are put in the test set.

2. **Select frames from the training set videos** – A CNN requires images to be trained. As such, frames need to be extracted from the training set beforehand. However, when training a CNN, the different classes should be trained with roughly the same amount of training images. Not complying with this can have a severe negative impact on the CNN performance, as shown in [128]. Hence, it is not possible to simply use all the frames from the videos in the training set for training as the percentage of fishing relevant and non-relevant frames would be very different. Thus, it is necessary to perform the training with a similar number of fishing relevant and non-relevant frames. To achieve this target, the frames for training are extracted from the videos in the training set as follows:
    - All frames from the relevant event parts of the videos in the training set are selected and used for training.

- The same number of frames are randomly extracted from the non-relevant footage part, this means non-relevant videos and the non-relevant parts of videos that incorporate relevant events.

By applying the process above a total number of 266 778 frames from relevant events and the same number from non-relevant footage has been selected to re-train the last three AlexNet layers for fishing event detection. This process resulted in 47 test videos, out of which 15 containing relevant events and 32 containing non-relevant parts only.

4.4.2 Performance Assessment Metrics and Procedures

To assess the performance of the proposed solution as detailed in Section 4.3, some performance metrics are required. This section defines the metrics that are appropriate to measure the performance of the proposed solution.

4.4.2.1 Receiver Operating Characteristic Curve

A Receiver Operating Characteristic (ROC) curve is a graphical representation of the accuracy of a binary classifier as its discrimination threshold is varied [129]. In this context, a binary classifier is an algorithm with the purpose of, given a stimulus, classifying it as either a “positive event”, i.e. the type of event for which the algorithm was designed to detect, or a “negative event”, i.e. all other uninteresting events. The ROC curve allows to assess the accuracy of a binary classifier (or several) by plotting the True Positive rate versus the False Positive rate for several discrimination thresholds, for some testing set, as depicted in Figure 42 (b). A lower, more lenient discrimination threshold leads to a larger TP rate but at the cost of FP rate increase, and vice-versa. This fundamental tradeoff in these two components will be naturally different from one classifier to another. The aim is to have high TP rate starting as low as possible in terms of FP rate, i.e. a high-performance classifier should have its ROC curve closer to the TP rate axis getting closer to the maximum TP rate as fast as possible, as seen in Figure 42 (a). Therefore, a way to select the best binary classifier among several other classifiers is to plot them all in the same chart and then selecting the one performing closest to the vertical axis. The TP and the FP rates are defined as follows:

- **True Positive rate** – Ratio between the number of positive events correctly classified as positives (true positives) and the total number of actual positive events (which is the sum of the true positives and false negatives) computed as:

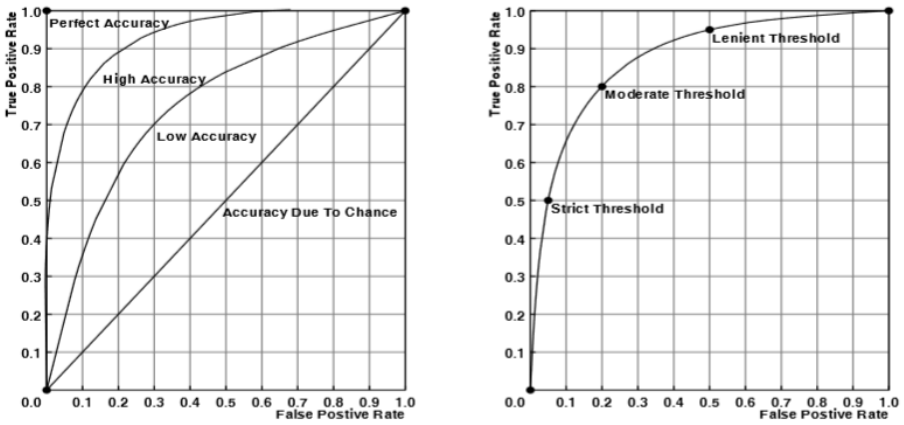
$$TPR = \frac{TP}{TP + FN} \tag{4.3}$$

where TPR means the True Positive rate, TP the number of True Positives for a given threshold and FN the number of False Negatives for the same threshold.

- **False Positive rate** – Ratio between the number of negative events wrongly classified as positives (false positives) and the total number of actual negative events (which is the sum of the false positives and true negatives) computed as:

$$FPR = \frac{FP}{FP + TN} \tag{4.4}$$

where FPR means the False Positive rate, FP the number of False Positives for a given threshold and TN the number of True Negatives for the same given threshold. The actual definitions of TP, FP, TN and FN are described next.



(a) (b)

Figure 42 – ROC curves examples [130]: (a) ROC curves of classifiers with different accuracies; (b) discrimination threshold variation within a ROC curve.

The goal of using ROC curves is to measure the proposed solution ability to identify the fishing events which are first detected at the frame level and, after some processing, at the event level.

Frame-level Performance Assessment Procedure

In this section, the assessment procedure is performed at the frame level, thus evaluating the output of the architectural module working at this level. More precisely, the performance impact of the temporal filter for several values of the *sliding_window_width* parameter will be studied, therefore choosing the value offering the best performance. For computing the TP and FP rates, for a given discriminative threshold, it is necessary to define:

- **True positives** (TP) – Frames that, for a given discriminative threshold, are classified as part of a relevant event and are indeed part of a relevant event according to the ground-truth.
- **False positive** (FP) – Frames that, for a given discriminative threshold, are classified as part of a relevant event but are, in fact, not part of a relevant event according to the ground-truth.
- **True negatives** (TN) **–** Frames that, for a given discriminative threshold, are classified as not part of a relevant event and are indeed not part of a relevant event according to the ground-truth.
- **False negatives** (FN) **-** Frames that, for a given discriminative threshold, are classified as not part of a relevant event but are, in fact, part of a relevant event according to the ground-truth.
- ***TP + FN*** – All frames that belong to relevant events as defined by the ground-truth.
- ***FP + TN*** – All frames that do not belong to relevant events as defined by the ground-truth.

With these definitions, the TPR and FPR can be calculated with Equations (4.3) and (4.4).

Event-level Performance Assessment Procedure

The final output of the fishing activity detection and classification framework (in terms of detection and classification, neglecting the representative frame selection) needs also to be evaluated with a suitable procedure. As this output is event based and not directly aligned to the ground-truth data, some delay in the detection of the event can occur or even some misclassification during a relevant (or non-relevant) event. Due to this misalignment and the need to precisely define what is a true/false positive/negative for the event-level performance assessment, it is proposed to divide, both the ground-truth and the output of the classifier, into temporal segments. For this, the following procedure is applied:

1. ***Ground-truth event boundaries processing*** – Each test video is divided into equal segments with a duration equal to the *minimum_relevant_event_duration* (in this case 60 seconds). Then, each time boundary of a relevant event in the ground-truth (manually defined) is rounded to the limits of the closest segment, as depicted in Figure 43.
2. ***Detected events boundaries processing*** – Detected events, which are the events labeled as relevant by the event classification module, are assigned into the same intervals previously defined by also rounding each event boundary to the closest segment boundary.

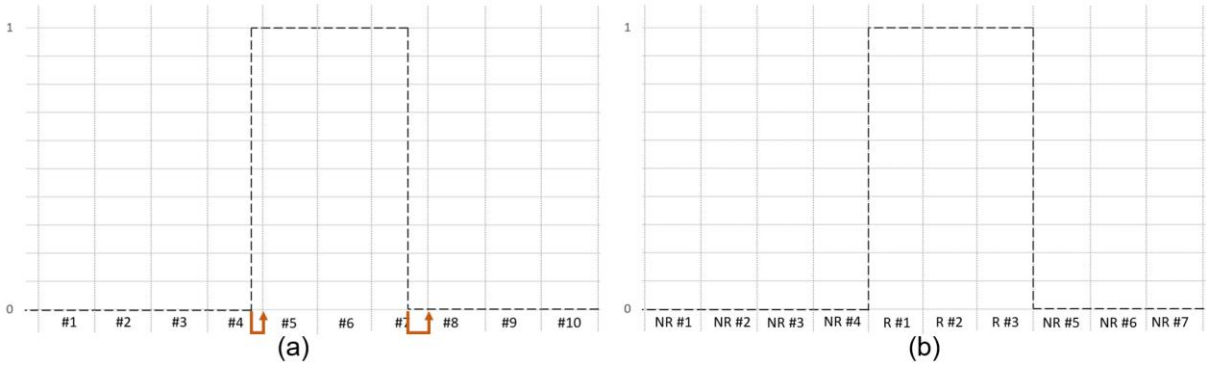


Figure 43 – Creation of segments and approximation of boundaries for ground-truth. (a) before; (b) after.

These two steps allow the boundaries of the relevant events in the output of the detection and the ground-truth to be represented with the same temporal precision and thus aligned despite some temporal localization (quantization) error. The localization errors are always lower than ½ the *minimum_relevant_event_duration* which is considered a negligible error for the target application. Now that all (detected and ground truth) events are referred to the same segments, the computation of the true positive and false positive rates can be made, for a given discrimination threshold. The following definitions come rather naturally:

- **True positives** – Event segments that, for a given *event_threshold*, the event detection module classifies as relevant and are indeed relevant according to the ground truth.
- **False positive** – Event segments that, for a given *event_threshold*, the event detection module classifies as relevant but are, in fact, not relevant according to the ground truth.
- **True negatives –** Event segments that, for a given *event_threshold*, the event detection module classifies as not relevant and are indeed not relevant according to the ground truth.

- **False negatives –** Event segments that, for a given *event_threshold*, the event detection module classifies as not relevant but are, in fact, relevant according to the ground truth.
- **TP + FN** – All relevant segments within the test videos.
- **FP + TN** – All non-relevant segments within the test videos.

4.4.2.2 Event Boundaries Localization Error

The previously described metrics are able to describe how well events can be detected. However, it is also important to measure the precision of the boundaries that separate both types of events. Therefore, a procedure to measure the boundaries errors between the detected events and the ground-truth's relevant events is proposed. This section describes the procedure aiming to measure the detected events precision (temporally), especially because the definition of the ROC curves above introduces some localization errors.

In a scenario where only one event is detected (that mostly intersects the ground-truth), the computation of these boundaries errors is the difference between the left boundary of the ground-truth's relevant event and the left boundary of the detected event, and the right boundary of the ground-truth's relevant event and the right boundary of the detected event. However, there are more complex cases that can occur, such as the possible existence of more than one detected event and false positives that may or may not partially overlap with the ground-truth and must be filtered out (true positive filtering). Thus, a procedure that is able to measure the error boundaries for all possible cases is proposed below, followed by an example to facilitate its understanding.

This procedure assumes that all detected events boundaries and ground-truth's relevant events boundaries are known. The left and right boundary of each of the detected events is called `detected_left[i]` and `detected_right[i]`, respectively, where `i` denotes the i-th detected event, i.e., `i=0` for the first detected event within a given testing video, `i=1` for the second and so forth. The left and right boundaries of each of the ground-truth's relevant events are called `gt_left[j]` and `gt_right[j]`, respectively, where `j` denotes the `j-th` ground-truth's relevant event, i.e., for a given testing set `j=0` means the first relevant event, `j=1` the second, and so forth. The procedure is described next. For each ground-truth relevant event, `j`, the following procedure is applied:

1. **True positive filtering:** As the Event Boundaries Localization Error metric should consider only detected events that somehow overlap temporally with a ground truth relevant event some true positive filtering of events has to be performed. Thus, it is first computed the overlap in time (`overlap[i]`) of all the detected events with respect to the ground-truth's relevant event `j` with the following pseudo-code. The `minimum_relevant_event_duration` corresponds to the minimum event duration while the `true_positives_total` is the number of true positives for each ground-truth's relevant event, j. The algorithm starts by computing the size of the overlapping between the detected event and the ground-truth. If more than 50% of the detected event overlaps with ground-truth events or the overlapping is greater than `minimum_relevant_event_duration`, then the detected event is considered a true positive.

```
1. if(detected_left[i] and detected_right[i] < gt_right[j]) or if(detected_left[i] and
   detected_right[i] > gt_left[j]), then overlap[i] = 0.
   else, overlap[i] = min(detected_right[i], gt_right[j]) – max(detected_left[i]-gt_left[j])

2. if(overlap[i] > 0.5*(detected_right[i] - detected_left[i]) or overlap[i] >
   minimum_relevant_event_duration)) the detected event is considered true positive. Increment
   true_positives_total by one.
   else, it is considered a false positive.

3. if (true_positives_total == 0) go to the next event j.
```

2. **Computation of left and right boundary errors, $e_{left,j}$ and $e_{right,j}$, respectively**: Now the errors between the left and right boundaries are computed only considering the detected events that were considered true positive for the ground truth event j. The algorithm starts by finding the index, `i`, of the left boundary of the leftmost true positive detected event, `idx_left`, and the index, `i`, of the right boundary of the rightmost true positive detected event, `idx_right`. The left and right boundary errors can then be computed as the difference between the left boundary of the leftmost detected event and the ground-truth's relevant event left boundary, and the right boundary of the rightmost detected event and the ground-truth's relevant event right boundary, respectively.

```
1. For all detected events i, idx_left = index(min(detected_left[i]), where index is a function
   that returns the index. Ignore events with overlap[i] equal to 0.

2. For all detected events i, idx_right = index(max(detected_right[i]). Ignore events with
   overlap[i] equal to 0.

3. e_left,j = | detected_left[idx_left] − gt_left[j] |

4. e_right,j = | detected_right[idx_right] − gt_right[j] |
```

3. **Computation of boundary error metrics:** With the left and right boundaries errors computed, it is possible to measure the following metrics to assess the temporal precision of the detected events:
   - **Mean Absolute Boundaries Error (MABE)** – The purpose of the Mean Absolute Boundaries Error metric is to assess how precise are the detected event boundaries in comparison to the ground-truth. This metric is based on the average of the absolute errors between the detected event(s) boundaries and the corresponding ground-truth's relevant event, as shown in (4.5):

$$MABE = \frac{1}{2 \cdot \text{gt\_rel\_events}} \sum_{j=0}^{\text{gt\_rel\_events}} \left(|e_{left,j}| + |e_{right,j}| \right) \tag{4.5}$$

   where gt_rel_events is the total number of ground-truth's relevant events for which boundary errors were successfully computed, notably $e_{left,j}$ and $e_{right,j}$.

   - **Mean Relative Boundaries Error (MRBE) –** Like the MABE metric, the Mean Relative Boundaries Error also evaluates how precise are the detected event boundaries, but in this case in a relative way, thus considering the duration of the relevant event in question. This allows understanding how precise is the proposed solution relatively to the duration of the event, as shown in (4.6):

$$MRBE = \frac{1}{2 \cdot \text{ gt\_rel\_events}} \sum_{j=1}^{\text{gt\_rel\_events}} \left(\frac{|e_{left,j}| + |e_{right,j}|}{\text{gt\_event\_duration}_{\text{j}}} \right) \cdot 100 \tag{4.6}$$

where gt_rel_events is the total number of ground-truth's relevant events for which boundaries errors were successfully computed, notably $e_{left,j}$ and $e_{right,j}$, and $gt\_event\_duration_j$ is the duration of the j-th ground-truth's relevant event.

To understand the boundary error metric calculation procedure above, Figure 44 presents some illustrative examples. In Figure 44 (a), four events were detected. In *step 1* of the proposed procedure, the two rightmost events are filtered since they don't overlap enough with the ground truth and thus only the two leftmost events are considered true positives (see Figure 44(b)). In Figure 44 (c), the boundary errors are computed as described in *step 3*, i.e. the leftmost boundary and the rightmost boundary are compared with the corresponding ground truth boundaries for the error computation (Figure 44, (c), in yellow). Figure 44 (d) shows a similar case with just two detected events. In this simple scenario, only one true positive detected event (which is, hopefully, the most likely situation) is obtained and the resulting boundaries errors are as shown in Figure 44, (f).

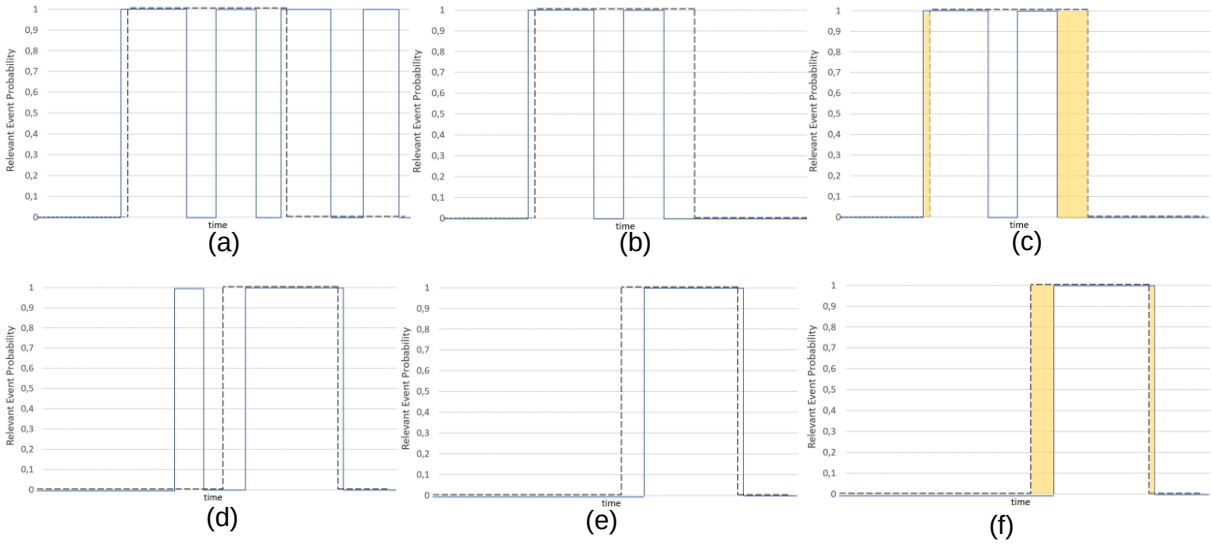


Figure 44 – Two examples of the several steps required to compute the error boundaries, for a given ground-truth relevant event. The blue line is the output of the event classification module, this means the detected events; the dashed black line regards the ground-truth events. (a) and (d) Before procedure. (b) and (e) After true positive filtering. (c) and (f) Boundary errors computation based on the true positive detected events boundaries.

4.4.3 Detection and Classification Results Analysis

In this section, the previously defined performance assessment metrics are used to evaluate the performance of the proposed solution and to find the best values for some relevant parameters. The performance assessment is made at the frame-level and then at the event-level.

4.4.3.1 Frame-based Performance Assessment

As shown in Section 4.3, the proposed solution performs a frame-level classification without exploiting past or future frames, which can lead to a rather inconsistent and highly fluctuating detection output. To address this problem, the output probability of the frame-level classification is used by a temporal

smoothing filter that is controlled by the *sliding_window_width* parameter: the highest its value, the smoother the signal at its output.

To find the optimal value for the *sliding_window_width* parameter, the smoothed output probability is thresholded and the accuracy is measured at the frame level. Thus, several thresholds were applied, in which frames with a probability higher than a threshold are considered positives, this means a frame belonging to a relevant event, and frames with a probability lower than a threshold are considered negatives and thus non-relevant. The following thresholds were applied: 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.

As shown in Figure 45, several ROC curves (this TPR versus FPR metric) were obtained for several *sliding_window_width* parameter values, notably 0 (meaning that no smoothing is applied), 5, 20, 40 and 60 seconds.

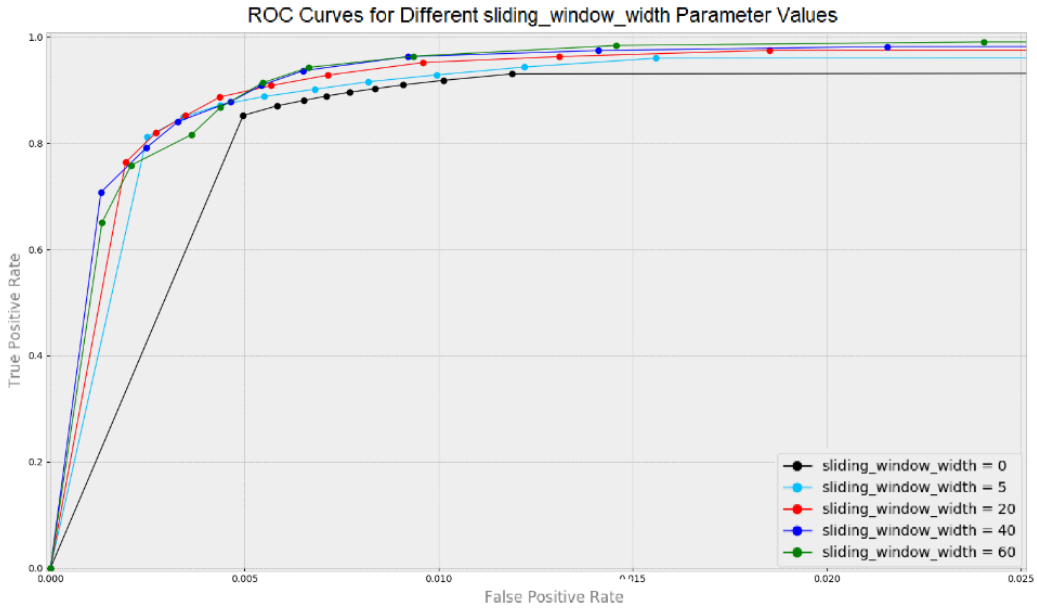


Figure 45 – ROC curves for different *sliding_window_width* parameter values.

From Figure 45, it can be concluded that high performance was obtained for the *sliding_window_width* parameter values of 20, 40 and 60 seconds, with slightly different tradeoffs in terms of TPR/FPR. The value of 20 seconds has been selected as the optimal one as it allows to lower the delay imposed by the temporal smoothing module, and it also allows to obtain (slightly) better TPR/FPR trade-offs for stricter thresholds (0.5 and up).

From the ROC curve, the performance of the frame-level detection module can be evaluated. However, to complement this analysis, the output probability of the temporal smoothing module for some test videos is shown next. Since there are two types of test videos in this dataset, i.e., relevant test videos which contain both relevant and non-relevant footage/events, and non-relevant test videos which contain non-relevant footage only, a separate analysis is done for each type. In the following analysis, the smoothed frame-level probability of a frame belonging to a relevant event is plotted for each video and its behavior analyzed.

Relevant Test Videos Analysis

Figure 46 shows the output of the temporal smoothing module for some of the 15 relevant test videos, for the selected *sliding_window_width* parameter value (20 seconds) along with the ground truth labelling. In Figure 47 and Figure 48, the output of the temporal smoothing module for two more relevant test videos are also presented, together with some frames extracted from problematic moments, this means parts of the video that yielded high frame-level event probabilities in non-relevant footage. Then analysis of the results in Figure 46 indicates that the frame-level event detection module, which is based on the well-known Convolutional Neural Network, AlexNet, has a rather good performance for this dataset. However, for the cases in Figure 47 and Figure 48, a significant amount of frames have a higher probability of belonging to a relevant event when in fact there is no relevant event taking place (false positive). The worst case is shown in Figure 48. In these cases, the relevant event frame-level probabilities spike near a relevant event, since there is usually a lot of activity going on, even after the fishing activity has finished.

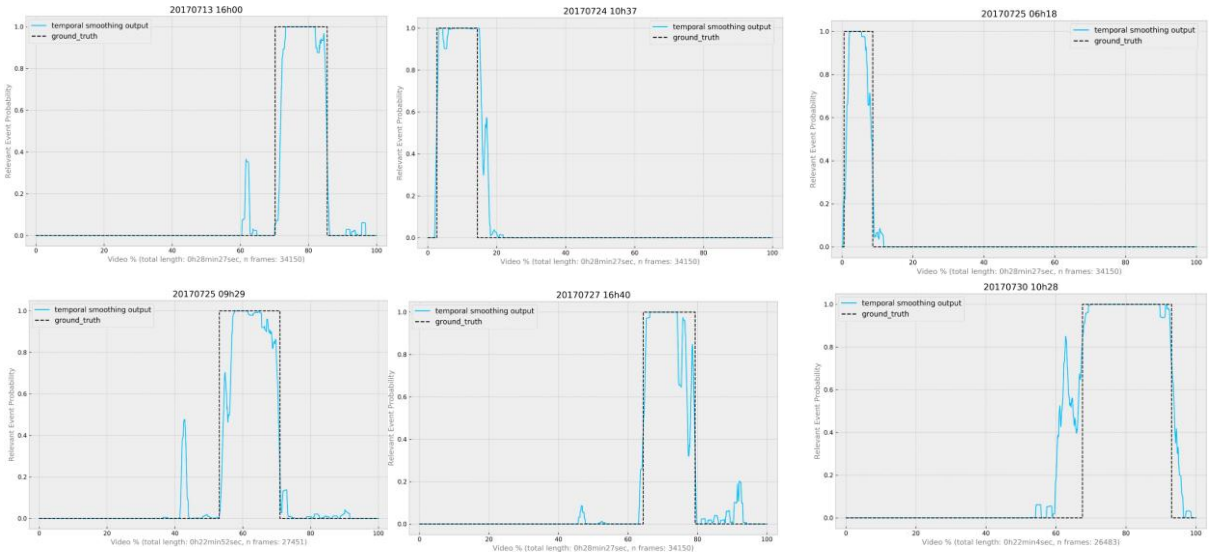


Figure 46 – Smoothed frame-level relevant event probability for some relevant test videos.

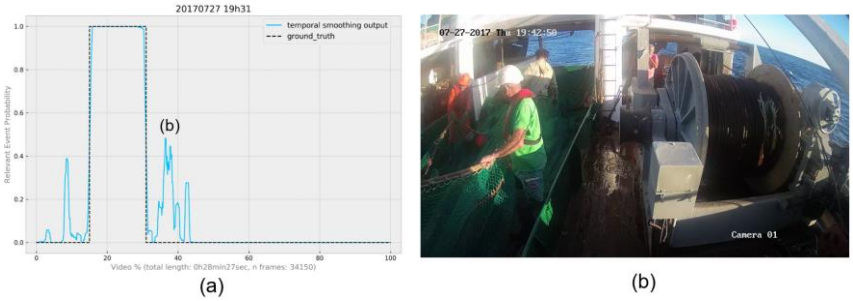


Figure 47 – Relevant test video results: (a) smoothed frame-level relevant event probability; (b) frame extracted from a problematic moment.

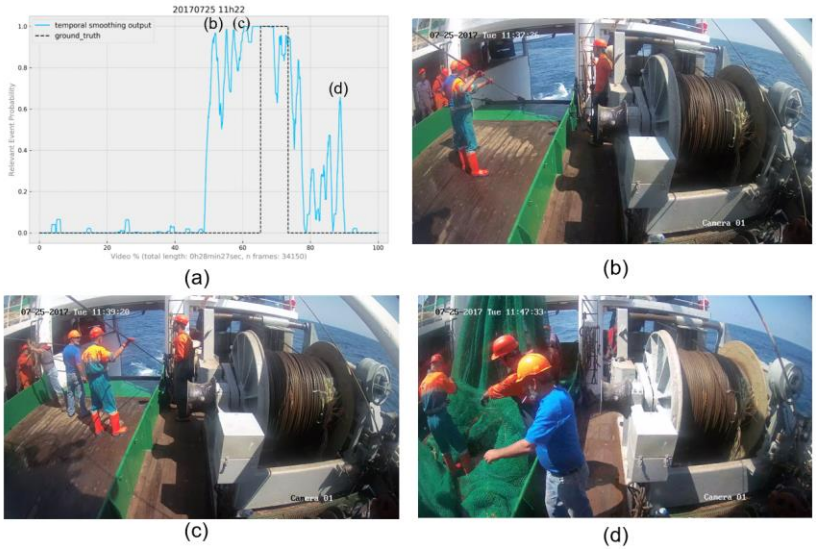


Figure 48 – Relevant test video results: (a) smoothed frame-level relevant event probability; (b), (c) and (d) frames extracted from problematic moments.

As described in Section 4.1, an event is considered relevant in two occasions: when the net is thrown and pulled out from the sea. In both occasions, the green door at the end of the vessel opens. The problematic video frame represented in Figure 47 shows the fishing men putting the fishing net on the floor, which is a normal preparation for a fishing activity (note that the green door at the end of the vessel is closed). According to the ground-truth definition, this is a non-relevant event; however, it somewhat resembles the net throwing and pulling procedures, which justifies the slight increase in the frame-level relevant event probability. As for the video represented in Figure 48, in (b) and (c) the fishing net is not yet visible and, thus, it is a non-relevant event (ground-truth labeling). However, these frames are similar to the beginning of the net pulling process when the net is visible. In this particular video, there is a problem with the net pulling process which causes the fishing men and cables to stay in that position for quite some time, thus increasing the relevant event probability much earlier than what is defined by the ground-truth. In Figure 48 (d), the fishing men are maneuvering the net, possibly untangling it, which is considered non-relevant by the ground-truth. However, as in Figure 47 (b), this resembles part of the net throwing and pulling procedures, thus spiking the frame-level relevant event probability.

Non-relevant Testing Videos Analysis

In this case, the output of the temporal smoothing module (after frame-level detection) is analyzed when no relevant events occur for quite some time. In these non-relevant test videos, there are almost no relevant event probability spikes: out of the 32 videos, only two videos have spikes in the frame-level relevant event probability, and even these spikes do not go to high values. In Figure 49, the results for these videos are presented, together with some extracted frames.

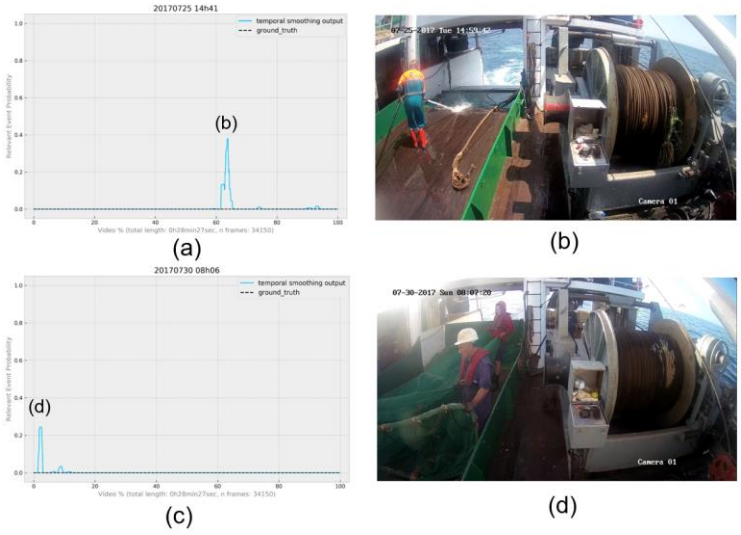


Figure 49 – Non-relevant testing videos with worse results: (a) and (c) output of the temporal smoothing module; (b) and (d) frames extracted from each of the (somewhat) problematic non-relevant testing videos.

Just for the relevant test videos, the frames causing spikes in the frame-level relevant event probability can be traced to similar frames in relevant events and thus it is natural that they occur, especially because the deep learning algorithm was trained with similar frames labeled as relevant. For example, Figure 49 (b), such as Figure 48 (b) and (c), is somewhat similar to the beginning of the net pulling process (or the end of the net throwing process). As for Figure 48 (d), where the net is just being placed on the floor by the fishing men, it is similar to the middle of the net pulling and throwing procedures, just like in Figure 47 (b).

In conclusion, the frame-level detection module followed by the temporal smoothing module performs remarkably well since it considers the positions taken by the crew when a fishing activity occurs, as well as the appearance and position of the fishing net, to output a probability value. Naturally, the fact that the training and test material are rather similar, notably resulting from the same shooting scenario, in terms of the boat and its areas, may contribute for this high performance.

4.4.3.2 Event-based Performance Assessment

The objective of this section is to study the performance of the proposed solution, notably for different values of the *event_threshold* and *event_time_window* parameters, which were introduced in Section 4.3, as well as the overall system performance. The *event_time_window* refers to the window used to compute the output of the temporal smoothing module (frame level relevant event probabilities) to assess whether a relevant event is taking place or not. A larger *event_time_window* 'looks' at more of the temporal smoothing module output before making a final decision. As for the *event_threshold*, it is a key parameter for the final detection decision as it determines if the Area Under the Curve (sum of the probabilities in a *event_time_window*) is enough (or not) for an event to be detected. The event-level performance assessment is performed with two types of metrics, both described in Section 4.4.2. Note also that the computation of the event-level ROC curves is based not on actual non-relevant/relevant events but on the fixed length segments which divide the non-relevant/relevant events, as detailed in Section 4.4.2.

Figure 50 presents several ROC curves for different *event_time_window* values, notably 30, 45, 60, 75 and 90 seconds. For each *event_time_window,* different *event_threshold* values were applied, notably 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1. In a ROC curve, the thresholds vary from the highest value at the left, to the lowest value at the right.

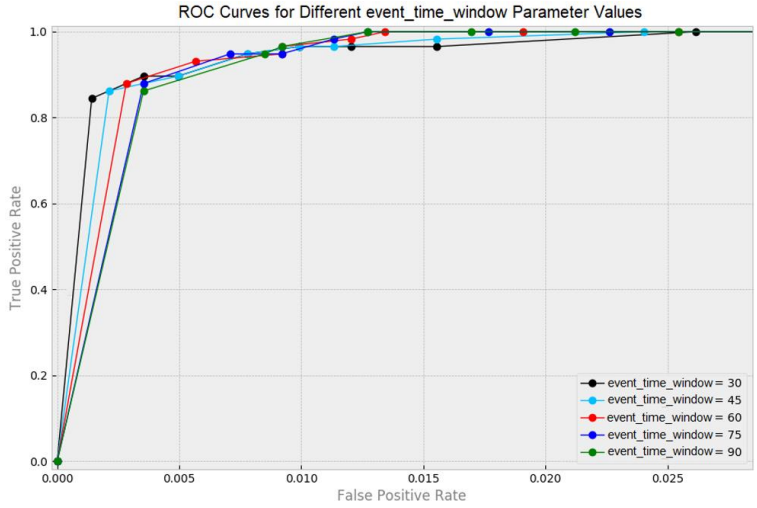


Figure 50 – ROC curves for different *event_time_window* values, by varying the *event_threshold* parameter.

As shown in Figure 50, the ROC performance for several event_time_window values is rather similar. This can be explained by the good performance of the frame-level event detection and temporal smoothing modules; thus having after a larger or smaller event_time_window does not have a big impact. However, this parameter should be large enough to suppress eventual drops in the relevant event probability and, thus, able to detect a single relevant event reliably, and not two (or more) separate/broken relevant events. Moreover, if the event_time_window is too small, a small spike could be interpreted as a relevant event when it is not although these small spikes are typically filtered by using the minimum_relevant_event_duration criterion.

Table 6 shows the results for the two previously presented error boundaries metrics, notably Mean Absolute Boundaries Error and Mean Relative Boundaries Error, for event_time_window parameter values of 30, 45, 60, 75, 90 and event_threshold parameter values of 0.6, 0.7 and 0.8 (for which the TPR/FPR values change more). These metrics measure the event boundary error in absolute and relative ways.

Table 6 – Event boundary error metrics results.

Event_time_window **Parameter (seconds)**	***Event_threshold*** **Parameter**	**MABE (seconds)**	**MRBE (%)**
30	**0.6**	25.6	14.7%
	0.7	23.3	13.3%
	0.8	21.2	11.9%
45	**0.6**	29.3	16.3%
	0.7	24.8	14.1%
	0.8	20.7	11.4%
60	**0.6**	30.4	17.1%
	0.7	27.4	15.6%
	0.8	21.0	11.5%
75	**0.6**	35.0	19.4%
	0.7	29.2	16.5%
	0.8	22.4	12.3%
90	**0.6**	40.1	21.9%
	0.7	31.4	17.6%
	0.8	23.9	13.1%

From the ROC results in Figure 50, the best performance achieved for medium to high FPRs (greater than 0.005) without sacrificing too much the performance for low FPRs is obtained for a event_time_window value of 60. A trend can be observed in Figure 50 as when event_time_window increases better performance is obtained for medium to high FPRs but the performance for low FPRs decreases. From the results in Table 6, it is possible to conclude that the errors increase (for the same *event_threshold*) when the *event_time_window* increases, for most the cases. This was expected since the larger decision windows contribute to a larger imprecision in the definition of the event boundaries. In this case, the *event_time_window* parameter value of 60 seconds represents a compromise as it does not have a large boundary error (comparing to *event_time_window* of 45s only increases 5s) and minimizes the number of false alarms comparing to other *event_time_window* ROC curves.

The *event_threshold* parameter should be selected according to the application requirements, since it represents a tradeoff between detecting more relevant events at the cost of more false positives and vice-versa. Since this application requires a low number of false positives, an *event_threshold* of 0.8 is adopted.

4.4.4 Coding Results and Analysis

As presented in Section 3.3, the fishing vessel transmits some selected images of fishing events through a satellite link. This type of channel has extremely low bandwidth as well as large delays and high cost. Thus, it is crucial to compress the image before transmission to the Control Center as much as possible, while still preserving a sufficient amount of quality to allow understanding what type of activities are going on in the vessel. Thus, the performance of some royalty free codecs has been assessed for this type of scenario and is present in Section 2.2.6. From an RD and computational complexity performance perspective the codecs that better fit the context of this work are the JPEG on Steroids codec, which is an optimized JPEG compliant codec, and the JPEG 2000 codec. As such, these codecs will be again compared, this time also with transmission costs in mind. As basis of comparison, a non-optimized JPEG is also evaluated:

- **Non-optimized JPEG**: First introduced in 1992 [131], JPEG it is still the most widely used coding solution for lossy still imaging compression. The Independent JPEG Group (release 9c) [132] implementation was used here. Default parameters were used (chrominance subsampling, 4:2:0, is used by default). The quality parameter varies from 1 to 100 where a value of 100 means the highest quality.
- **JPEG on Steroids** – A JPEG compliant codec described in [26] and available in [133]. Several optimizations are included to improve the rate-distortion (RD) performance of the reference JPEG implementation. Default settings as described in [133] were used, which include chroma subsampling.
- **JPEG 2000** – An image coding standard released in 2000 to address several JPEG standard limitations, such as blocking artifacts at low bitrates, inefficient scalability as well as to improve its RD performance [134]. However, today, few are the digital cameras and applications offering this format. The implementation used here is from OpenJPEG, release 2.3.0 [135]. Since it does not support chroma subsampling, an YUV 4:2:0 converted image was provided to the encoder.

The proposed solution as described in Section 4.3 with the adopted optimized parameters was run for the test videos, thus yielding one representative frame for each detected event. The following procedure was applied:

1. **Frame extraction:** Representative frames were extracted with a spatial resolution of 2688x1520 in the Portable Network Graphics (PNG) format, cropped to a 2560x1440 and then resized to a CIF-like 320x180 resolution, sub-multiple of the 2560x1520 spatial resolution. These were the images to be coded and transmitted. Chroma subsampling was performed for all coding solutions (4:2:0).
2. **Compression:** The selected images were compressed using each of the previously described codecs for four different quality parameters (QP). The QPs were selected to obtain the following bitrates: QP_1: 1500 bytes, QP_2: 2500 bytes, QP_3: 3500 bytes and QP_4: 4500 bytes. The non-optimized JPEG encoder was not able to compress the images for QP_1. To be able to reproduce the compression results, the actual quality parameters used for each codec are as follows:
    a. Non-optimized JPEG – QP_2: 5; QP_3: 9; QP_4: 15.
    b. JPEG on Steroids – QP_1: 5; QP_2: 10; QP_3: 17; QP_4: 25.
    c. JPEG 2000 – QP_1: 125; QP_2: 75; QP_3: 55; QP_4: 42 (for this codec, the quality parameter is actually a compression ratio).
3. **Quality evaluation:** To assess the quality degradation introduced by the lossy image compression, thus affecting the fidelity of the visual representation, the PSNR metric, which measures the error between the original data and the decoded images, was adopted. Results are only shown for the luminance component as it is the most important from the subjective point of view. Thus, for each codec, for each QP and for each image extracted from the relevant events, a PSNR value is obtained. To represent the results in a compact way, the PSNR values for the same QP/codec are averaged over all the extracted frames. The file size of the compressed images for each QP of each codec is also averaged. Also, a deblocking filter was applied to the non-optimized JPEG and JPEG on Steroids codecs, which substantially improves PSNR results. This filter is the available in

FFmpeg tool [136] and can be used by running the following flag and parameters: *-vf pp="hb/vb/dr/fq|32".* A description of this and other FFmpeg's processing filters are presented in [137]. The results, with and without pos-processing filters, are presented in Table 7.

Table 7 – Average values of PSNR, for each codec with and without pos-processing and QP.

		Non-optimized JPEG		**JPEG on Steroids**		**JPEG 2000**
		w/o deblocking	**w/ deblocking**	**w/o deblocking**	**w/ deblocking**	
QP_1	**PSNR** (dB)	-	-	24.26	24.96	27.34
	Size (bytes)	-		1472		1482
QP_2	**PSNR**	26.01	26.80	27.18	27.53	29.62
	Size	2571		2466		2574
QP_3	**PSNR**	27.97	28.59	28.73	28.90	31.21
	Size	3378		3481		3501
QP_4	**PSNR**	29.55	29.98	29.92	29.92	32.65
	Size	4472		4536		4528

An analysis on Table 7 shows that by applying a deblocking filter on the JPEG encoded images the objective quality is substantially increased. However, the use of such filter is beneficial only to some extent; as quality increases, the filter improvements start to be less obvious, actualy starting degrading the image quality at some point. There is also a difference from the improvements of the deblocking filter, due to the way the JPEG on Steroids is tweeked to 'squeeze' as much as possible an image in the encoding process. This ultimately results in reaching a saturation point before the non-optimized JPEG codec, resulting in the non-optimzed JPEG codec being able to surpass JPEG on Steroids after pos-processing. Despite performance increases, using a deblocking filter is not nearly enough for both JPEG codecs to compete with JPEG 2000. This was expected, since JPEG 2000 is a much more modern codec.

As described in Section 3 from [138] and [139], it is assumed that the data rate for the Iridium modem used by Xsealence corresponds to 125 bytes per second. Assuming this bitrate, the average transmission time can be computed for each image target size/quality, i.e., for 1500, 2500, 3500 and 4500 bytes, yielding 12, 20, 28, and 36 seconds, respectively. Also, using the price per kB indicated in [140]**,** which is 1.09$ per kB**,** the cost to transmit an image for the various target size/quality can be computed. In Table 8 is represented the cost associated with using each codec to transmit an image with a targeted quality Q1, Q2 and Q3.

Table 8 – Cost comparison between the three codecs; JPEG codecs are pos-processed.

		PSNR (dB)	**Size** (bytes)	**Cost $**
Q1	**Non-optimized JPEG**	27.27	2641	2.81
	JPEG on Steroids	27.39	2337	2.49
	JPEG 2000	27.34	1482	1.58
Q2	**Non-optimized JPEG**	29.02	3533	3.76
	JPEG on Steroids	28.98	3433	3.65
	JPEG 2000	29.03	2257	2.40
Q3	**Non-optimized JPEG**	30.57	4822	5.13
	JPEG on Steroids*	30.66	4970	5.29
	JPEG 2000	30.62	3134	3.34

Analyzing Table 8 reveals a somewhat costly solution, especially if a time-lapse (i.e., several frames) is asked by the Control Center. However, the benefits of the Control Center being able to immediately retrieve proof of an eventual illegal fishing activity can eventually make up for the additional cost. Also, it is shown that JPEG 2000 provides the lower cost, which was expected, based on the previous analysis on Table 7.

To provide an idea of the subjective quality inherent to these results, Table 9 shows the decoded images for a single selected representative image when using different codecs and QPs.

Table 9 – Compressed images for each codec and QP.

	JPEG2000	**JPEG on Steroids**	**Non- optimized JPEG**
QP_1	 PSNR: 27.34	 PSNR: 24.96	-
QP_2	 PSNR: 29.62	 PSNR: 27.53	 PSNR: 26.80
QP_3	 PSNR: 31.21	 PSNR: 28.90	 PSNR: 28.59
QP_4	 PSNR: 32.65	 PSNR: 29.92	 PSNR: 29.98

From the objective metrics in Table 7, Table 8 and the subjective examples in Table 9, it is possible to conclude that the JPEG 2000 codec is a clear winner as expected. However, as previously stated, JPEG 2000 lacks support in many applications, which can cause compatibility problems. On the other side, JPEG on Steroids, while still achieving a good performance, it is compatible with an enormous number of JPEG decoders. For these reasons, both codecs were implemented in the proposed solution, to be used as needed.

4.5 Proposed Solution Demonstration

The purpose of this work has been, from the beginning, the design and development of a proof-of-concept prototype capable of detecting relevant fishing events, taking place on a vessel, and transmitting highly compressed images classified as relevant by the detection algorithm. This proof-of-concept prototype should demonstrate its usability in the relevant field and fulfil the requirements presented in Section 3.4.2. A demonstration of such proof of concept was presented to Xsealence premises, where the work described in this Thesis was implemented (until Chapter 4, inclusive). A thorough explanation of the procedure can be found in Appendix C.

Chapter 5

5. Image Enhancement

In Section 4.4.4, a comparison between low-quality decoded images was made to understand the relationship between image quality and the cost associated to its transmission from a fishing vessel. This analysis was solely based on the bandwidth transmission cost of an encoded image and its corresponding quality after decoding and only with a simple post-processing step. This Chapter targets the enhancement of the decoded image quality using deep learning frameworks.

5.1 Context and Motivation

To maintain a low transmission cost from the fishing vessel using satellite channels, the decoded quality is rather low for the adopted coding solutions and thus, this chapter aims at proposing a powerful post-processing stage to be performed at the Control Center where the decoded image is enhanced and significant amount of computational resources are available. This stage restores a substantial amount of image detail to provide the users with a much better image quality, notably where details are more visible and less suffering from block effect and blurring.

Image enhancement is a classical problem in low-level vision and the associated technologies aim to improve/restore/enhance the overall quality of a degraded/corrupted image. This topic has been widely studied and discussed in the literature and a wide range of solutions have been proposed, notably related to operations such as denoising, inpainting, deblocking, and super-resolution, among others [141]; an example is depicted in Figure 51. Such processing operations have been traditionally based on analytical methods, which heavily rely on previous knowledge of the problem (assumptions). With the emergence of deep learning tools, several attempts to address such imaging problems have been made using this novel approach; similarly to what happened in the field of image classification, deep learning-based solutions quickly surpassed the state-of-the-art performance offered by the previous analytical enhancement approaches [142].

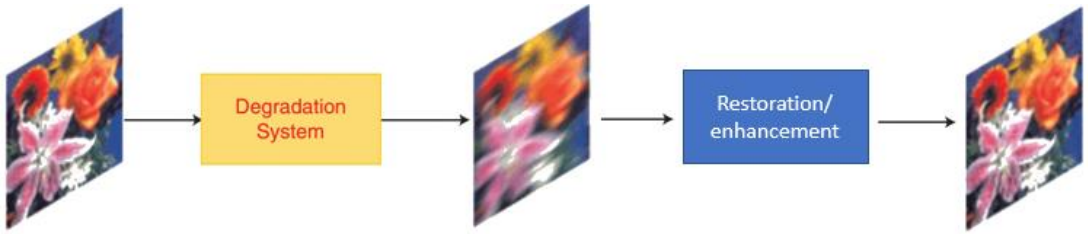


Figure 51 – Image degradation process followed by a restoration/enhancement step.

Following the fishing detection context above described, and the poor quality of the available decoded images, the last component of the entire solution proposed in this MSc Thesis, is to design and evaluate

a deep learning-based post-processing step able to enhance the decoded image quality. This brings great benefit for the success of the entire system since it allows to use higher compression factors, and thus less rate, to minimize the critical transmission costs. Naturally, this final processing operation will greatly improve the viewing experience at the Control Center as well as the effectiveness of the fishing control task.

5.2 Using CNNs for Image Enhancement

The use of CNNs for image enhancement purposes requires some architectural changes regarding image classification CNN architectural siblings. In image classification, CNNs usually have pooling layers to minimize the number of parameters in the network. On the other hand, image enhancement usually requires that the input data dimensions are equal to the output data dimensions and hence, pooling layers are usually not used. Moreover, since there is no classification in these CNNs, fully connected layers are typically not used. Hence, CNNs for image enhancement related tasks are usually comprised of convolutional layers only (followed by a non-linearity function).

Latest advances in deep learning enhancement methods have been largely due to the use of *residual learning* [142]. This approach was initially proposed in [143] for image classification and aimed to work around the problem faced when training very deep networks (e.g. with more than 100 layers) as, with the increased depth, accuracy gets saturated and it starts to degrade rapidly [143]. When residual learning is used, new "shortcut" connections are added, as depicted in Figure 52 (a); in this case, the layers within the network, learn residual functions with reference to the layer inputs, instead of learning unreferenced functions [143], i.e. the residual (difference) of input and output of some layer is learned. This approach allows the training of deeper networks for which non-residual networks struggle (e.g. weights that are difficult to be updated), but also allows improved overall performance. By being able to successfully train deeper networks, the model does not only has more representative power (the deeper the network, the more complex and thus more representative is the function mapping the input to the output) but also with an increased overall receptive field (see Figure 52 (b)) which provides more contextual information at each layer. The receptive field is the region in the input space that a particular CNN's neuron is looking at (i.e. be affected by). By doing so, the image recovery/enhancement task performance is further improved [142].

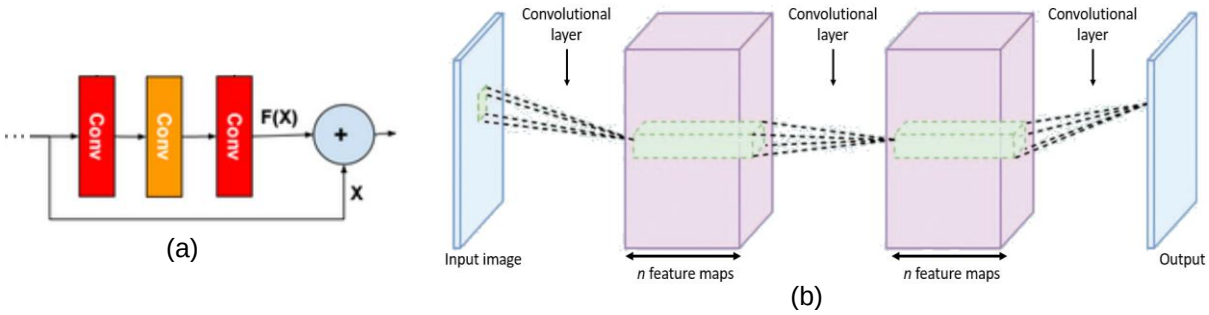


Figure 52 – (a) A residual block [144]; a skip connection is added, therefore allowing the learning of only the residual between two or more layers, instead of learning a new mapping function from one layer to the next [142]; (b) A fully convolutional network [142]: an activation/pixel at the output, 'sees' only a portion of its input, the so-called receptive field; the bigger the receptive field, the better.

When it comes to lower the image storage/transmission size, two approaches are interesting for this case: i) down-sampling, with the uncompressed image corresponding to a (reduced) number of pixels, and ii) image coding with a lower quality, i.e. employing a larger quantization step size. These processes result in two major types of artifacts: artifacts due to the upscaling interpolation method (considering that the image is up-sampled back to the original resolution) and compression artifacts due to the lossy compression method applied. The problem of enhancing the image by removing these artifacts has been addressed by the research community using deep learning models, notably using super-resolution and (lossy) compression artifacts removal approaches.

5.2.1 Enhancing by Super-resolution

Generic single image super-resolution (SISR) algorithms aim to generate high-quality high-resolution (HR) images from a *single* low-resolution (LR) input image [145]. Typically, SISR CNNs take as input an already up-sampled image obtained by a standard interpolation method (usually bicubic [146]) to the target spatial resolution, rather than having the CNN up-sampling the image itself.

A first successful attempt at using SISR with deep learning (described in [147]) uses a network relying on three convolutional layers only, which was enough to overcome the state-of-the-art on non-deep learning based solutions available at that time. However, this network had some underlying problems, notably the lack of ability to train a single model for more than one up-sampling factor, a small overall receptive field and a slow training convergence [148].

The next solutions have achieved better performance by a large margin by using residual learning, which allowed accurate training of very deep networks. In [148], a network architecture very similar to the well-known VGG-net with 20 layers was proposed. Similarly, in [149], a 16-layer network was proposed to restore several low-level images artifacts, SISR inclusive. More recently, the performance has increased by using very deep networks with residual blocks as used in the ResNet image classification CNN [150] [151].

5.2.2 Enhancing by Compression Artifacts Removal

Lossy image compression solutions, e.g. JPEG, WebP, HEVC codecs, are able to achieve high image compression ratios by discarding less perceptually relevant information for the human vision system. However, very high compression ratios lead to complex and highly visible compression artifacts, notably blocking artifacts, ringing effects and blurring [152].

In recent years, some image enhancement solutions based on CNNs have been proposed to remove such compression artifacts. Due to the huge popularity of the JPEG codec, such solutions focus mainly on JPEG compression artifacts. These solutions are usually very similar, or sometimes even identical, to those used in SISR as basically some effective image enhancement has to be performed. A first successful attempt was achieved in [152] by essentially reusing the CNN architecture used for SISR in [147], with some minor changes. Later, a deeper CNN resorting to residual learning and a multi-scale loss function, i.e. with several loss functions were computed at several points of the network, was introduced [153]. Similar results were obtained with the network proposed in [149]. Interestingly, the

latter was trained both for SISR as well as for image denoising, thus providing very good image enhancement generalization capabilities.

5.3 Adopted Image Enhancement CNN

From the several networks described in Section 5.2.1 and 5.2.2, only the network presented in [149], the DnCNN network, has been evaluated (by its authors) and the results show that works well for both super-resolution and compression artifacts removal (notably JPEGs artifacts). Initially, the main purpose of the authors was to provide image denoising capabilities with unknown Gaussian levels, thus departing from typical discriminative denoising models which require the knowledge of a given model for each Gaussian noise level. However, the authors found that a *single* trained model was able to provide state-of-the-art performance for super-resolution (for different scaling factors), JPEG deblocking (for different quality factors) and Gaussian denoising. For this reason, this was the CNN architecture adopted to restore the low-quality images obtained at the Control Center after decoding due to the low-resolution and heavy JPEG compression used to reduce the satellite transmission costs.

5.3.1 DnCNN Architecture

The DnCNN is based on the VGG-net architecture [143]. However, some of its layers were removed, notably the pooling and fully connected layers, to make it suitable for general image enhancement purposes as explained in Section 5.2.2. All convolutional layers have 64 filters with 3x3xd size, where d = 1 in the first layer and 64 in all other layers. The last layer has only one 3x3x64 filter, which is used to reconstruct the output. Since this network accepts a single channel as input, denoising, super-resolution and JPEG deblocking are performed only for the luminance channel of the images.

The DnCNN network offers three keys characteristics whose combination was the main responsible for its high performance, notably:

- **Deep architecture** – This CNN includes 16 convolutional layers. As stated in Section 5.2, using a network with increased depth further increases the overall receptive field of the model, which in turn provides more contextual information at each layer of the network, thus improving the network performance [142].
- **Residual learning** – Similarly to what has been done in the field of image enhancement using CNNs (as detailed in Section 5.2.1 and Section 5.2.2), DnCNN also adopts residual learning. However, instead of multiple residual blocks along its architecture, this network employs a single residual unit to predict the residual image. This means that upon receiving a distorted image at its input, the CNN will implicitly remove the clean image throughout its hidden layers, thus producing the distortion at its output; the output of the network, i.e., the residual is then subtracted from the input image, thus resulting in the desired, enhanced (closer to clean) image (see Figure 53) [149].
- **Batch normalization** – In the process of training Deep Neural Networks, the distribution of each of the layer's inputs changes throughout the training process, which slows down the training since it requires a lower learning rate and a careful parameter initialization [154]. Using a batch

normalization layer before the non-linearity layer/function minimizes this problem, thus allowing a faster training, superior performance and lower sensitivity to initialization [149].

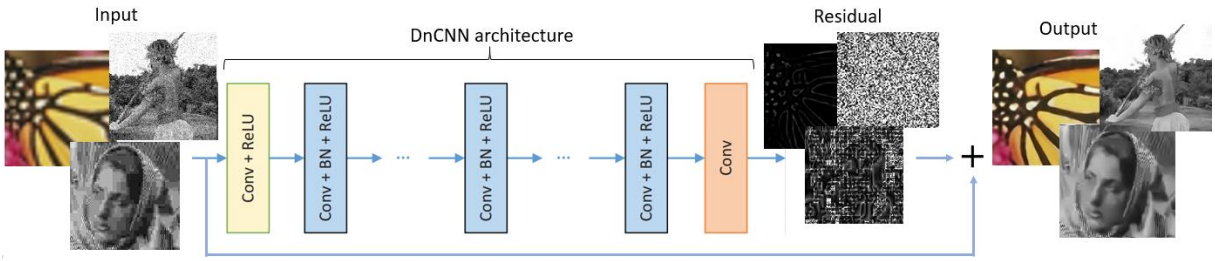


Figure 53 – DnCNN architecture [149].

The single DnCNN model with triple capabilities, this means good performance for denoising, super-resolution and deblocking (although only the last 2 are of major interest in the context of this Thesis) is available [155]. Its training was accomplished by using 291 images, as described in [149]. Several 50x50 patches were extracted from each of the images, which were then either JPEG encoded (quality factors ranging from 5 to 99), down-sampled and then up-sampled using bicubic interpolation (using scaling factors of 2, 3 and 4), or added with a certain Gaussian-noise level. Rotation and flip operations were also used for data augmentation. As for the loss function, the mean squared error was used, which naturally favors using PSNR as the final quality metric.

5.3.2 Double-Stage DnCNN-based Image Enhancement Configuration

As it was previously mentioned in this section, image storage/transmission size reduction can be obtained by down-sampling, by lossy compression, or both. Using such methods can yield very high compression factors, and thus low rates, regarding the original image but typically at the cost of a poor-quality image. However, using the DnCNN presented in the previous section some of the lost details can be restored and the overall subjective quality of the image can be improved. Since the compression artifacts that the DnCNN was trained to remove were JPEG related ones, the proposed image enhancement strategy assumes encoding performed with the JPEG on Steroids codec rather than the JPEG 2000 codec.

The proposed enhancement configuration aims to enhance a low-quality image which was produced after taking an image with high-resolution, performing spatial down-sampling with a certain pre-selected scaling factor, and then using the JPEG on Steroids encoder to finally compress the image. In the context of this MSc Thesis, this process would take place on the vessel, before transmission. Upon receiving the compressed image, the Control Center would proceed to enhance the image by first decoding the image, and after removing first the compression artifacts (in practice performing deblocking) and next restore the image to its original resolution using a super-resolution strategy (upscaling and DnCNN enhancement). This process is depicted in Figure 54. As it will be shown later, this double-enhancement strategy performs better than a single-enhancement strategy where deblocking and super-resolution are performed in a single deep learning-based stage.

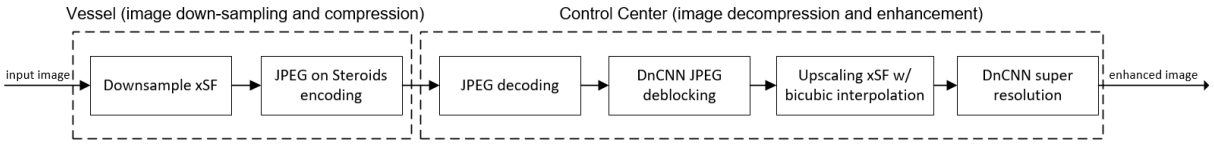


Figure 54 –Double stage image enhancement configuration. SF means scaling factor.

The proposed double-stage enhancement configuration is implemented using a single DnCNN model, i.e., only one network trained for both super-resolution and JPEG deblocking is used. The double-stage enhancement strategy presented in Figure 54 targets to remove one artifact at a time. First, the image is down-sampled and coded using the JPEG on Steroids encoder. After decoding with the JPEG on Steroids decoder, the image is passed through de DnCNN for deblocking. Since only JPEG related artifacts are present, the DnCNN removes them as best as possible. Then, after upscaling the image to its original spatial resolution with a bicubic interpolation [146], a new set of artifacts related to the upscaling of the image appears which are addressed by processing the deblocked and up-sampled image through the DnCNN again to remove these artifacts from the image; since the network was also trained for super-resolution, it should be able to also (as much as possible) remove these artifacts.

There is, however, an issue that needs careful consideration depending on the application and targets at hand: depending on the desired compression ratio, the optimal down-sampling (and consequent up-sampling) scaling factor differs. For instance, if the goal is to obtain better quality (and thus less compressed) images, then no down-sampling should be applied before coding, as doing so would result in a worse decoded quality image; however, as the need for higher compression increases, down-sampling before coding starts yielding better final quality images. This means that, for a given range of compression ratio/bitrates, there is an optimal scaling factor. The optimal down-sampling scaling factor value for the context of this Thesis will be studied in the following section.

5.4 Performance Assessment

This section starts by assessing the performance of a single stage enhancement configuration, this means where super-resolution and deblocking are performed individually and not one after the other. After, the optimal down-sampling scaling factor parameter will be studied for the double-stage image enhancement configuration proposed in the previous section; finally, the advantages/gains of using the deblocking and super-resolution stages will be highlighted.

5.4.1 Test Dataset

Several datasets have been used in the literature for testing super-resolution and deblocking enhancement solutions. Since the proposed double-stage enhancement configuration explicitly considers both types of artifacts, some images were taken from such datasets, as the LIVE1, Classic5 and Set14 datasets available in [156]. Also, some images from the dataset used for royalty free still image codecs performance assessment in Section 2.3 were used. Moreover, to appropriately test the proposed image enhancement solution for the context of this Thesis, 10 images were taken from the fishing video dataset, presented in Section 4.1.2, with the largest possible variety of events and situations. All images were organized into three sub-datasets named as faces, landscapes and fishing boats. To be able to compare the compressed images in term of storage size, all images were cropped

and down-sampled to a specific number of pixels, in this case 230400 pixels; for a square image, this implies a spatial resolution of 480 x 480 pixels. Finally, for the sake of saving space, the performance results for the faces and landscapes datasets, as well as the description of these datasets such as spatial resolution, can be found in Appendix D. The images from the fishing boats dataset have a spatial resolution of 640x360 and its enhancement performance will be reported here since they are the main target of this Thesis.

Figure 55 – Example images from each sub-dataset; (a) landscapes; (b) faces; (c) fishing boats.

5.4.2 Single-Stage Image Enhancement Configuration Performance Assessment

Before diving into the proposed double-stage image enhancement configuration proposed in Section 5.3.2, which combines the super-resolution and deblocking enhancement, an independent analysis of each of these two enhancements approaches will be performed. Since only the luminance channel is enhanced by the DnCNN, both objective and subjective results will be displayed on this channel only. Table 10 and Figure 56 show the results and gains of using the DnCNN only for deblocking purposes, notably in terms of PSNR, SSIM and subjective quality.

Table 10 – PSNR and SSIM comparison for a JPEG on Steroids decoded image before and after only deblocking. The quality factor for JPEG coding ranges from 0 to 100; higher is better.

	PSNR (dB)			SSIM		
Quality factor (Q)	**No restoration**	**DnCNN restoration**	**Gain**	**No restoration**	**DnCNN restoration**	**Gain**
Q = 6	26.09	26.95	0.86	0.68	0.73	0.05
Q = 10	27.61	28.56	0.95	0.75	0.79	0.04
Q = 15	28.83	29.94	1.11	0.80	0.84	0.03
Q = 20	29.72	30.95	1.23	0.83	0.86	0.03
Q = 25	30.44	31.72	1.28	0.85	0.88	0.02
Q = 30	31.04	32.36	1.33	0.87	0.89	0.02

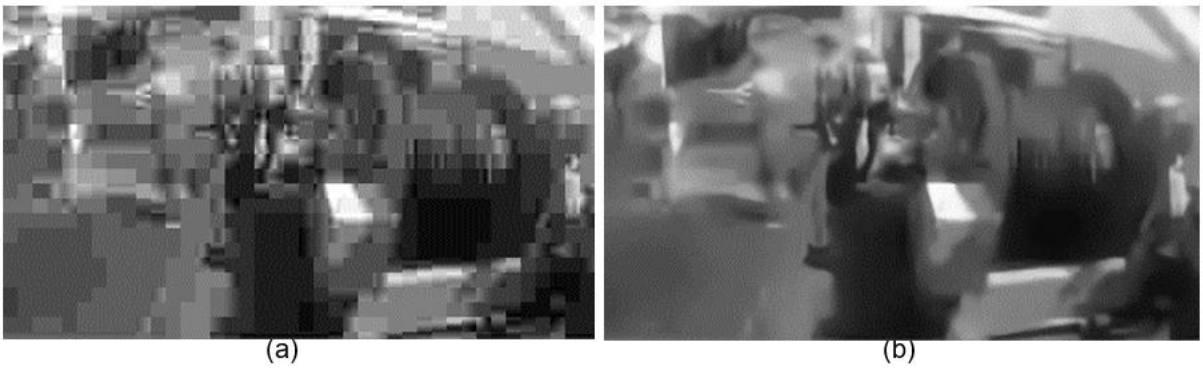


Figure 56 – JPEG on Steroids decoded image after compression with a quality factor of 6 and a spatial resolution of 320x180; luminance only (a) before enhancement; (b) after deblocking only enhancement.

The analysis of Table 10 shows substantial PSNR gains, especially for higher values of the JPEG quality factor. On the other hand, the SSIM metric shows a gain decrease as the quality factor increases. This behavior may be related to the fact that the SSIM is a more perceptual-based metric and thus, for the lower quality factors the removal of blocking artifacts provides a more substantial perceptual/subjective quality improvement; the SSIM metric reflects this behavior by yielding higher gains for the lower quality factors.

These PSNR and SSIM gains can be confirmed by the subjective analysis of Figure 56, where many blocking artifacts are successfully removed from Figure 56 (a) to (b). The resulting image offers much higher subjective quality, although is still blurry; however, this is due to the lack of detail resulting from the aggressive JPEG compression, and not due to blurring resulting from the deblocking operation, as it occurs in some traditional deblocking algorithms [152].

Table 11 and Figure 57 present similar results when using the DnCNN only for super-resolution purposes. Table 11 shows a substantial increase for the PSNR and SSIM values after applying the DnCNN super-resolution enhancement. This increase is rather noticeable when comparing Figure 57 (a) with (b); from the blurry image in (a) that results from bicubic interpolation, the DnCNN is able to restore some high-frequency details; however, the DnCNN is unable to successfully restore some finer details such as the sea undulation and the pulley system rope. This causes some parts of the image to remain blurred.

Table 11 – PSNR and SSIM comparison for bicubic interpolation, Lanczos [157] interpolation with a kernel size of 3 interpolation as implemented in matlab, and DnCNN super-resolution enhancement.

	PSNR			**SSIM**		
Scaling Factor	**Bicubic**	**Lanczos3**	**DnCNN restoration**	**Bicubic**	**Lanczos3**	**DnCNN restoration**
2x	29.37	29.64	32.04	0.88	0.89	0.93
3x	27.31	27.44	28.87	0.81	0.82	0.87
4x	26.34	26.44	27.44	0.75	0.75	0.81

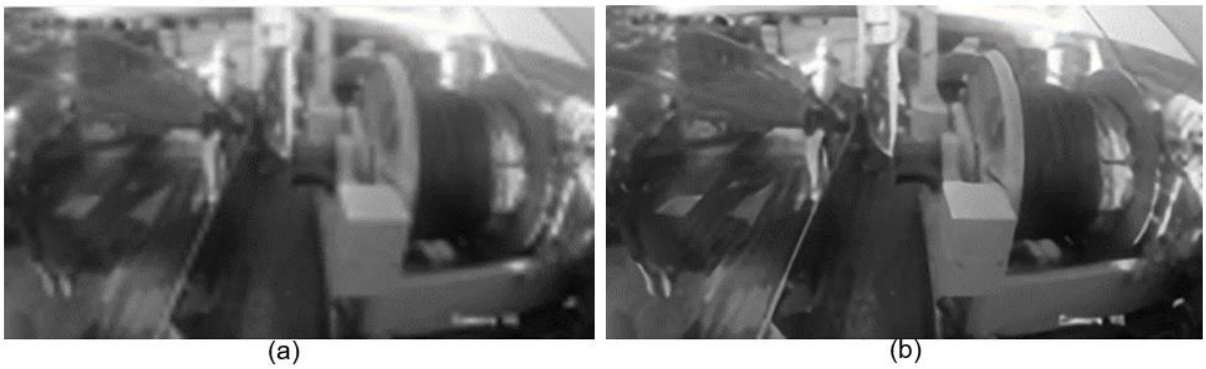


Figure 57 – Image with spatial resolution of 106x60 up-sampled to a spatial resolution of 318x180 (scaling factor of 3): (a) before enhancement; (b) after super-resolution only enhancement.

5.4.3 Double-Stage Image Enhancement Configuration Performance Assessment

After the individual analysis of the two enhancements approaches, notably deblocking and super-resolution, it is time to present the performance assessment of the double-stage image enhancement configuration proposed in Section 5.3.2.

Finding the Optimal Down-Sampling Scaling Factor Value

As seen in Section 5.3.2, when compressing an image, it might compensate to down-sample before encoding and then up-sampling after decoding. The optimal value for the down-sampling (and consequent up-sampling) scaling factor, here-after referred only as *scaling factor*, depends on the desired bitrate/compression ratio. Thus, to find the optimal scaling factor value for the double-stage image enhancement configuration, the RD performance obtained with the PSNR and SSIM distortion metrics is presented in Figure 58, for scaling factors of 1 (meaning the image is not resized), 2, 3 and 4.

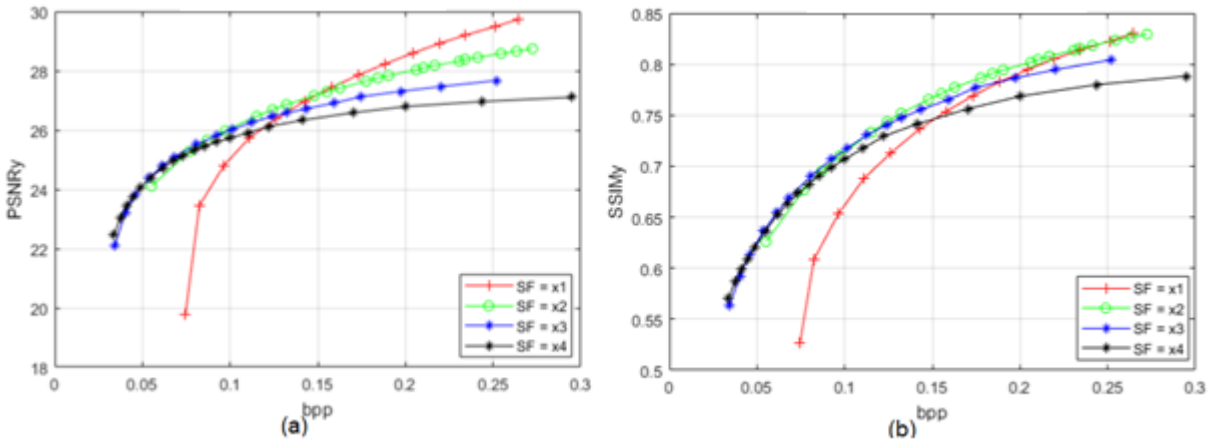


Figure 58 – Scaling factor optimization using RD performance. Each point in each curve corresponds to a particular quality factor. (a) PSNR; (b) SSIM.

From Figure 58, it is possible to conclude that, from the PSNR and SSIM metrics perspective, a scaling factor of 1, should be applied for images with a desired bitrate higher than 0.150 bpp and 0.251 bpp, respectively. Below these bitrates, a scaling factor of 2 should be used, until reaching a bpp of 0.093 and 0.113 for the PSNR and SSIM metrics, respectively. Below this rate point, a scaling factor of 3 should be used instead, until a bpp of 0.055 and 0.044 for the PSNR and SSIM metrics, respectively,

are reached; for lower bitrates, a scaling factor of 4 is preferred. In practice, this means that the lower is the target rate, the lower the spatial resolution the image has to be coded to be advantageous in terms of RD performance. This information is represented in Table 12, where the optimal scaling factors for each bitrate interval are presented.

Table 12 – Optimal scaling factors for each bitrate interval by presenting the upper and lower bounds of the intervals where a given scaling factor is optimal (in bitrate (bpp) and storage file size (kilobytes, kB))

		Scaling Factor			
		1x	**2x**	**3x**	**4x**
PSNR	**Upper bound (bpp/kB)**	∞	0.150/4.06	0.093/2.28	0.055/1.38
	Lower bound (bpp/kB)	0.150/4.06	0.093/2.28	0.055/1.38	-∞
SSIM	**Upper bound (bpp/kB)**	∞	0.251/5.90	0.113/2.74	0.044/1.27
	Lower bound (bpp/kB)	0.251/5.90	0.113/2.74	0.044/1.27	-∞

Table 12 shows discrepancies between the optimal intervals for the PSNR and SSIM metrics for a given scaling factor. For example, PSNR recommends changing the scaling factor from 3 to 2 at a bpp/kB of 0.072/2.28, whereas SSIM recommends doing this at 0.095/2.74. However, as SSIM is a more perceptual-based metric, the optimal scale factors should be chosen according to its defined intervals.

Assessing the Contributions of each Image Enhancement Stage

Above, the optimal scaling factor for different bitrates was found for the double-stage image enhancement configuration, which is constituted by two image enhancement stages: deblocking and super-resolution. However, it is not yet known the individual contribution of each of these image enhancement stages to the final quality. Thus, this section aims to assess the individual contributions of each of these enhancement stages in the context of the double-stage configuration.

To achieve this target, the PSNR and SSIM results for four different configurations are compared: using both super-resolution and deblocking enhancement stages (i.e., the original double-stage configuration), using the deblocking stage only, using the super-resolution stage only and using neither of the enhancement stages.

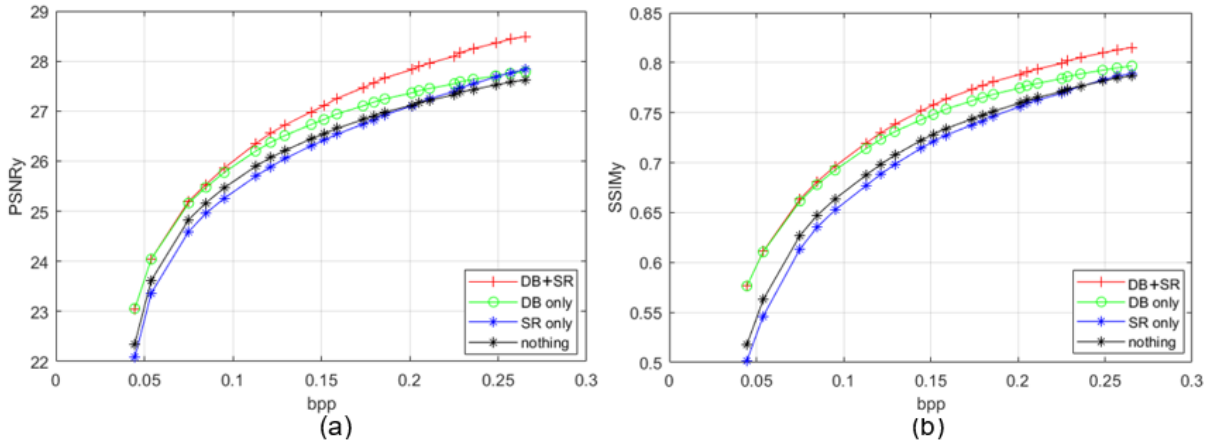


Figure 59 – RD performance for four different enhancement approaches for a scaling factor of 2 and varying quality factor. The left-most point was obtained using a quality factor of 3; the right-most point was obtained using a quality factor of 61.

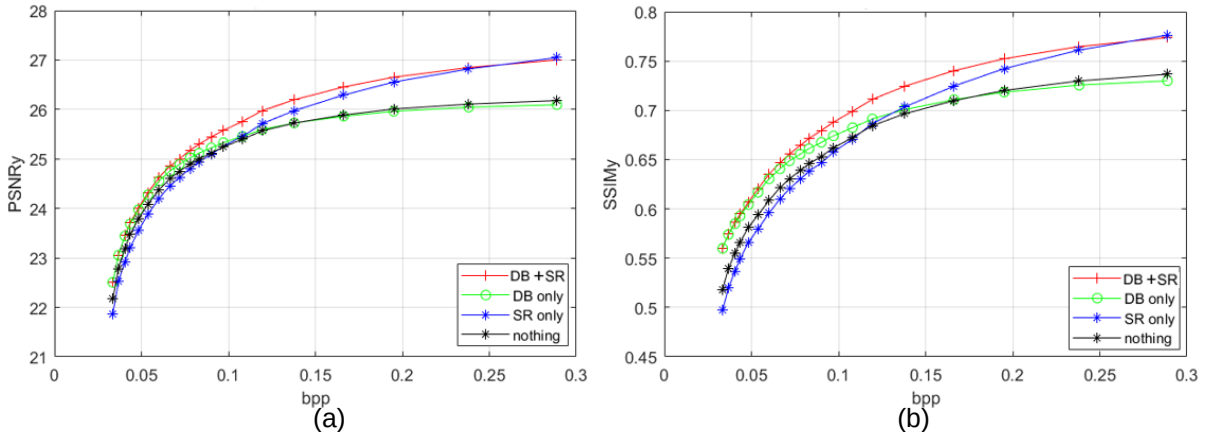


Figure 60 – RD performance for four different enhancement approaches for a scaling factor of 4 and varying quality factor. The left-most point was obtained using a quality factor of 5; the right-most point was obtained using a quality factor of 96.

The analysis of the results allows to conclude:

- **Deblocking only** – A closer look at Figure 59 and Figure 60 shows that, for the lower bitrates, the performance of this stage is the same as using the double-stage configuration, meaning that the super-resolution step has pratically no contribution. This was expected for lower bitrates and thus, lower quality factor values for the JPEG on Steroids encoder since the lack of detail due to very high compression is very intense. Since some pre-existing detail is required to efficiently perform super-resolution, very low bitrate decoded images do not benefit much from super-resolution techniques. As the bitrate increases, so does the detail, thus providing the DnCNN with more information to work with, therefore increasing the super-resolution stage contribution in terms of quality improvements. This means that, as the quality factor increases, the deblocking only stage starts to lose more and more against the double-stage configuration. At some point, for very high-quality factor values (96 and up), passing an image through the DnCNN for enhancement actually decreases its objective quality as some ‘true’ high-frequency content may be removed as noise.
- **Super-resolution only** – For lower bitrates, using the super-resolution only configuration is worse than doing nothing due to the heavily present compression artifacts. Since the decoded image is

not passed through the DnCNN right after decoding, when the image is upscaled, so are the compression artifacts. When passing the resulting image through the DnCNN, the network is unable to 'understand' the upscaled compression artifacts (they may not have been part of the training) which causes to actually have a worse image quality. Also, as the quality factor values used in the encoding process increase, so does the contribution of this stage. Since Figure 60 presents higher quality factors (images suffered down-sample with a scaling factor of 4, hence use higher quality factors to achieve the same bitrate), super-resolution contribution is thus higher.

- **Deblocking and super-resolution (proposed double-stage configuration)** – As expected, this is the configuration that provide the best performance. This is achieved by restoring a type of artifact at a time: first, the image passes through the DnCNN to remove the JPEG compression artifacts. After, the image is upscaled to its original resolution, thus resulting in some interpolation artifacts. By passing the image again through the DnCNN, the interpolation artifacts are minimized and some of the original image details are restored, resulting in a better overall quality image.

Final Performance Assessment

A double-stage image enhancement similar such as the one presented in this section could also be achieved through conventional methods, which do not require any learning. Thus, it was defined as benchmark, a solution that performs image enhancement with a conventional deblocking filter, notably the deblocking filter presented in Section 4.5.4 (FFmpeg's pos-processing filter employed by using the parameters pp="hb/vb/dr/fq|32" [137]) and the bicubic interpolation up-scaling method. This will allow to assess the improvements of the deep learning based method presented in this section. In Table 13, results are shown for a double-stage image enhancement using conventional methods and the double-stage image enhancement using the DnCNN network as presented in Section 5.3.2, for the boat dataset.

Table 13 – Comparison between no enhancing (besides bicubic interpolation), conventional enhancement and DnCNN enhancement.

			Without enhancement		**Conventional enhancement**		**DnCNN enhancement**	
Scaling factor	**Quality factor**	**Avg. bitrate (bpp/kB)**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**
1x	**Q = 3**	0.097/2.71	23.78	0.59	+0.64	+0.05	+1.02	+0.06
	Q = 5	0.126/3.54	25.50	0.66	+0.54	+0.04	+0.87	+0.05
	Q = 8	0.173/4.87	26.94	0.72	+0.39	+0.03	+0.92	+0.05
2x	**Q = 7**	0.066/1.85	24.32	0.61	+0.26	+0.03	+0.44	+0.04
	Q = 12	0.092/2.58	25.35	0.67	+0.09	+0.01	+0.47	+0.03
	Q = 20	0.128/3.59	25.82	0.70	+0.32	+0.02	+0.95	+0.05
3x	**Q = 15**	0.066/1.85	24.64	0.63	+0.00	+0.01	+0.36	+0.03
	Q = 27	0.090/2.52	25.34	0.67	-0.11	+0.00	+0.42	+0.03
	Q = 48	0.125/3.50	25.93	0.71	-0.19	-0.01	+0.54	+0.03
4x	**Q = 25**	0.064/1.79	24.51	0.63	-0.06	+0.00	+0.32	+0.03
	Q = 48	0.088/2.47	25.14	0.66	-0.16	+0.00	+0.40	+0.04
	Q = 76	0.125/3.50	25.67	0.70	-0.25	-0.01	+0.49	+0.03

Table 13 shows that the double-stage image enhancement method using the DnCNN provides better performance with respect to the benchmark. This is especially true for higher quality factors where the conventional deblocking filter improvements start to be less evident and at some point, even lowering

the image quality. At the same time, the deep learning approach keeps improving the image, especially through the super-resolution stage whose advantage is more evident for higher quality factors. A few subjective results are provided below in Figure 61 and Figure 62. By inspecting the results obtained for these methods it is possible to see that the enhancement method using the DnCNN network indeed provides sharper and visually more pleasant results in comparison with conventional methods.

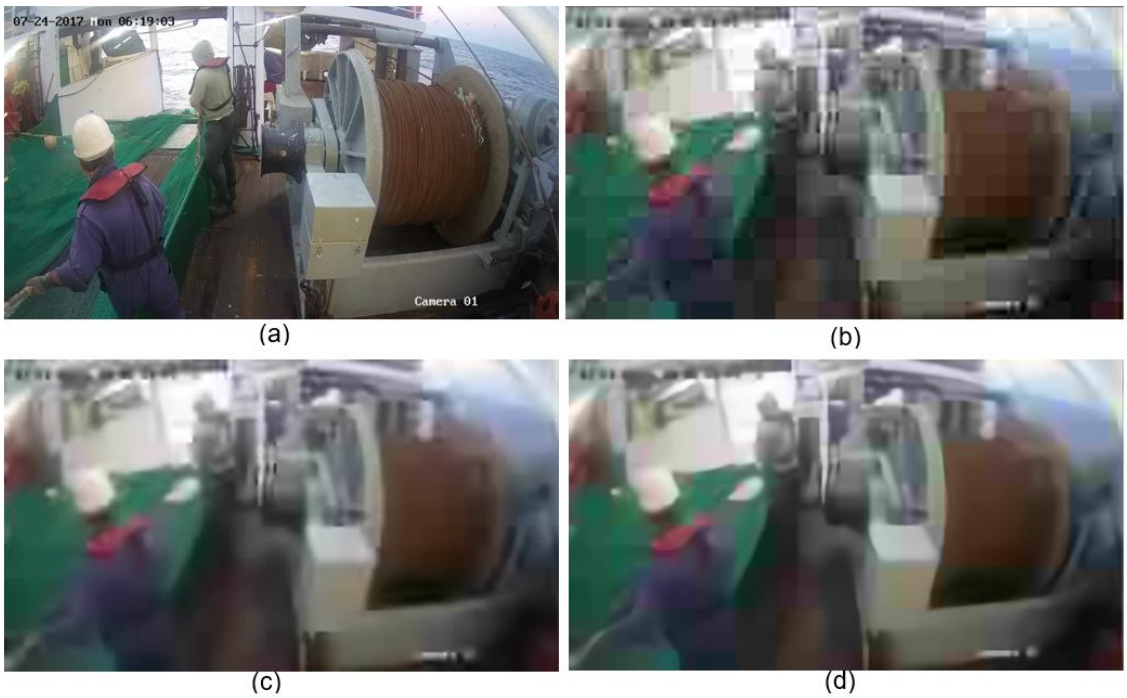


Figure 61 – Results obtained for an image (of a fishing event) with a spatial resolution of 640x360 processed with a scaling factor of 3 and encoded with a bitrate of 0.062/1.76kB: (a) original; (b) compressed with Q = 15; (c) enhanced with conventional methods (spp and bicubic); (d) DnCNN image enhancement.

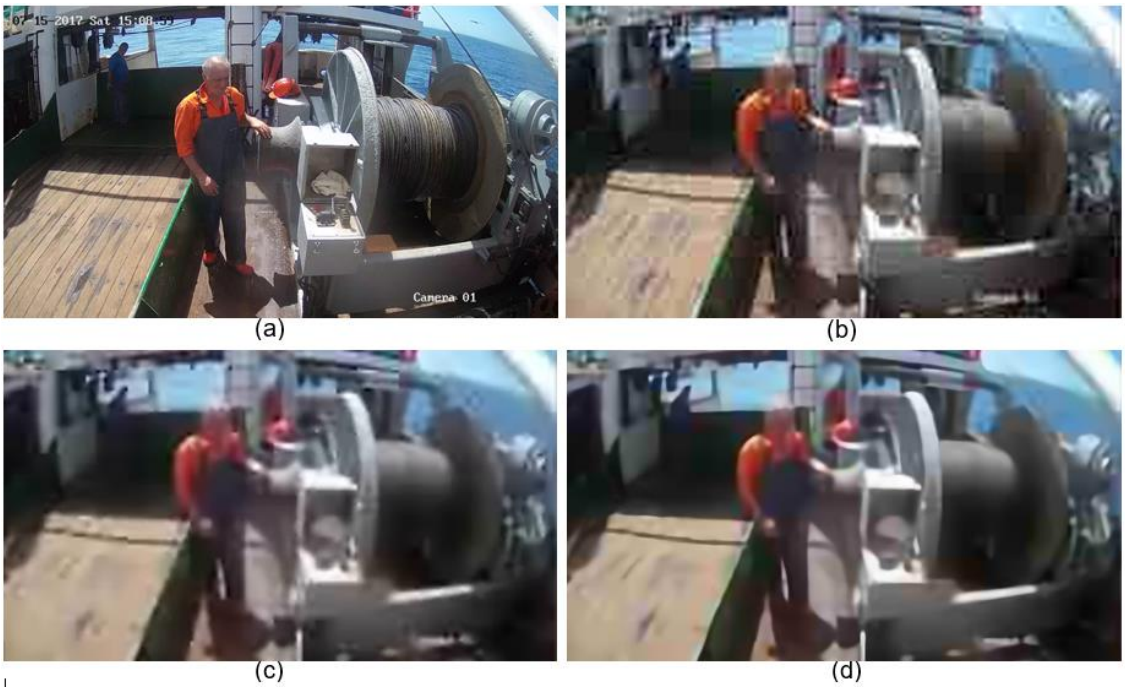


Figure 62 – Results obtained for an image with a spatial resolution of 640x360 processed with a scaling factor of 3 and encoded with a bitrate of 0.083/2.32kB. (a) original; (b) compressed with Q = 18; (c) enhanced with conventional methods (spp and bicubic); (d) enhanced with DnCNN.

This chapter has exploited deep-learning methods for image enhancement of heavily compressed (and down-sampled) images. A Convolutional Neural Network named DnCNN was used, as it provides very good deblocking and super-resolution performance using a *single* trained model. Using the DnCNN, a double-stage image enhancement configuration was proposed to allow higher compression ratios by enhancing the image quality after the decoding process. Also, for this configuration, the optimal scaling factors for some bitrate intervals were assessed. For the fishing context addressed in this Thesis, a scale factor of 3 would provide the best results, since it ideally covers an interval from 0.044 bpp/1.27 kB to 0.095 bpp/2.74 kB (for images with a spatial resolution of 640x360).

Chapter 6

6. Conclusions and Future Work

The purpose of this last chapter is to conclude this MSc Thesis by summarizing its main achievements and highlighting the main conclusions. Finally, some thoughts on possible future work are presented.

6.1 Summary and Conclusions

Nowadays, Vessel Monitoring Systems play a very important role in fishing activity monitoring, control and surveillance methods. However, there are limitations to be addressed, notably the detection of fishing activities in prohibited zones. Although position, speed and other information are provided by the VMS system, detecting fishing activities using only this information may be rather ineffective. The purpose of this work was to overcome this problem by finding an automatic solution that, by visually monitoring the vessel activities, would be able to automatically detect when fishing activities occur. After detection, the Control Center would be informed of a relevant event followed by receiving visual data that could eventually confirm or deny if any illegal activity is being performed.

To detect fishing activities, an available deep learning based solution using a Convolutional Neural Network developed for image classification was used. This solution was trained on a novel fishing dataset containing more than 100 hours of video footage. The performance assessment of the designed fishing detection solution has shown that it provides very good results in detecting relevant events, i.e., fishing activities visible to the camera; moreover, while the detection of the relevant event boundaries is not very precise as it shows an error from 10% to 20%, this is not a critical issue for the overall performance of the solution as only the beginning and ending of the fishing activity may be missed and thus the main part of the fishing activity is caught. Thus, the designed solution proves that it is possible to achieve high performance in real-time, automatic detection of fishing events using available video data.

Another objective was to compress and transmit a fishing event representative frame to the Control Center with as low as possible compressed size (for an acceptable quality) to lower the transmission costs. A study on the most relevant available royalty free codecs and their performance was presented in Section 2.2 and Section 4.4.4. reported a transmission-cost analysis using the most relevant image codecs. The results show that it is indeed possible to compress images in a way that makes its transmission for the available channels and conditions economically feasible, which is a strength of the proposed solution.

A working proof-of-concept implementing this detection and compression solution was developed. For this, a system mimicking the environment at a fishing vessel and following the architecture and requirements detailed in Chapter 3 was implemented. All necessary image processing algorithms were implemented in a Raspberry Pi with a video camera and (wired) connected to the VMS unit provided by *Xsealenc*e, which in turn was connected to the Control Center through GPRS. A description of the whole procedure was described in Section 4.5.

Finally, Chapter 5 presented and assessed the usage of a deep learning based solution for image enhancement, suitable for images with heavy compression artifacts or heavily smoothed (due to down-sampling before encoding and up-sampling at the receiver). It was shown that by using such type of solutions, notably for compression artifacts and super-resolution removal, a very substantial increase in both the objective and subjective quality may be achieved, therefore enabling higher compression factors and thus lowering the transmission costs.

6.2 Future work

Despite the encouraging results obtained for both the fishing activities detection and image compression and enhancement, the proposed solutions can still benefit from some improvements:

- **Fishing dataset augmentation** – The performance of a deep CNN, as other machine learning algorithms, depends critically on the data it has been trained with. If the visual data to classify has rather different characteristics from the training data, the image classification CNN will naturally not be able to maintain high levels of performance. In this MSc Thesis, the training data was selected from the fishing dataset presented in Section 4.1, and therefore the solution might perform less effectively for testing videos recorded in other vessels with different configurations, with different fishing methods or even with substantially different video camera positions. In the future, more video recordings of fishing activities might be available, which would enrich the current training dataset and significantly improve the generalization capabilities of the proposed solution.
- **Improved fishing detection** – The proposed fishing detection algorithm is based on a frame-by-frame individual classification. However, there are more complex approaches which could improve the detection performance. Using an image classification CNN architecture with better accuracy or using an architecture capable of exploiting temporal information by 'looking' at more than one frame at a time should lead to improved performance. Also, using deep learning object detection frameworks (such as the several region-based CNN architectures) where the appearance of certain objects (e.g. fish, net) in the video would trigger a special kind of event, may provide good performance while being more robust (and generic) to changes in the vessel configuration, camera position, etc. Although promising, all these improvements would most likely result in a computational complexity increase which may not be affordable currently in devices with a low amount of computational resources (e.g. Raspeberry Pi).
- **Improved image enhancement –** Using deep learning for low-level vision tasks is currently a hot topic. Hence, more sophisticated and more powerful CNN architectures are being developed and made available; exploring these advances would further improve the image enhancement

capabilities presented in Section 5.2, eventually further allowing to reduce the image compressed size and the transmission costs without penalizing the final quality.

In conclusion, there is still room to improve the overall performance and generalization capabilities of the proposed system. By doing so, an intelligent video-surveillance system for low bandwidth links may eventually start to be integrated in the fishing vessels, effectively surveilling their activities in a smart and somewhat low-cost way. Such achievement would significantly improve the effectiveness of Vessel Monitoring Systems, thus helping to ensure the availability and sustainability of fish stocks for generations to come.

Appendix A

A. Performance Assessment

This appendix reports the performance assessment results obtained for the image codecs, test material and test conditions presented in Section 2.2.6.2. To report the results, two different configurations are used: the *multiple codecs per image*, which compares the performance of multiple codecs when compressing a given image, and the *multiple images per codec*, which compares the performance of a given codec when compressing different images. For both configurations, the performance assessment is based on the $SSIM_Y$ and the $PSNR_Y$ metrics. Two different spatial resolution sets were used: the medium and the low spatial resolution set.

A.1 Multiple codecs per image

The results for *multiple codecs for one image* representation are presented below, for both spatial resolution sets.

$PSNR_Y$ results – medium spatial resolution

The results for the PSNR metric are represented below, from Figure 63 to Figure 66.

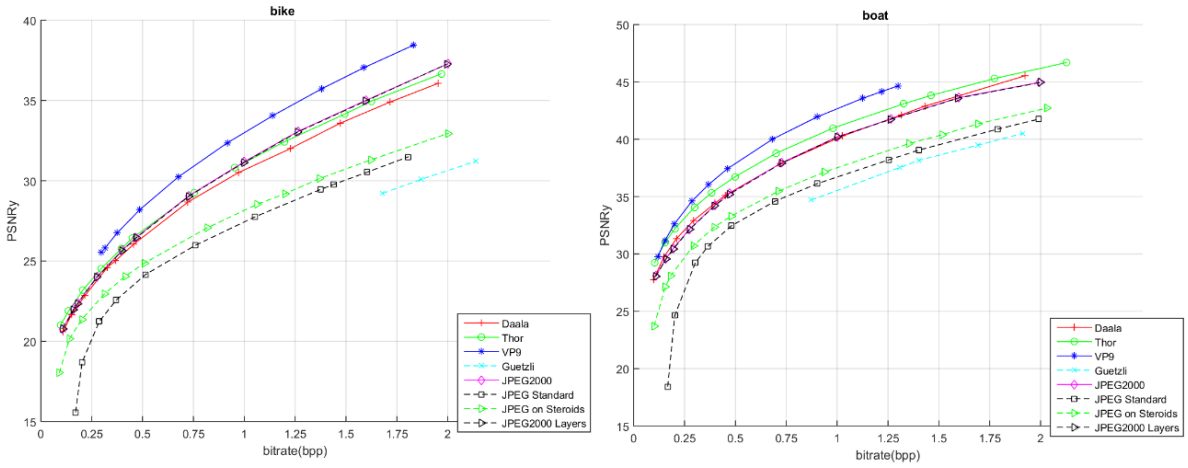


Figure 63 – $PSNR_Y$ metric comparison for *bike* (left) and *boat* (right) compression – medium spatial resolution set.

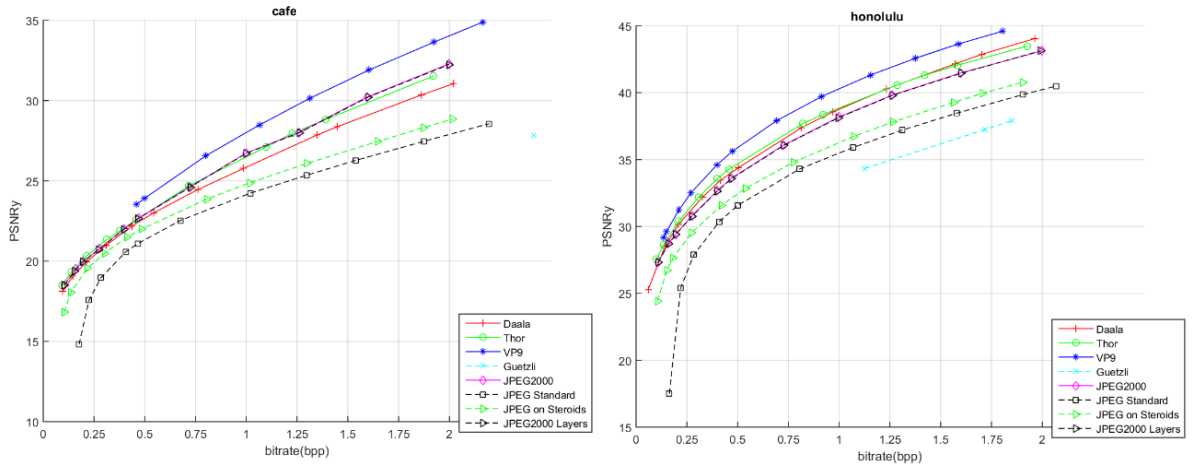


Figure 64 – PSNR$_Y$ metric comparison for *cafe* (left) and *Honolulu* (right) compression – medium spatial resolution set.

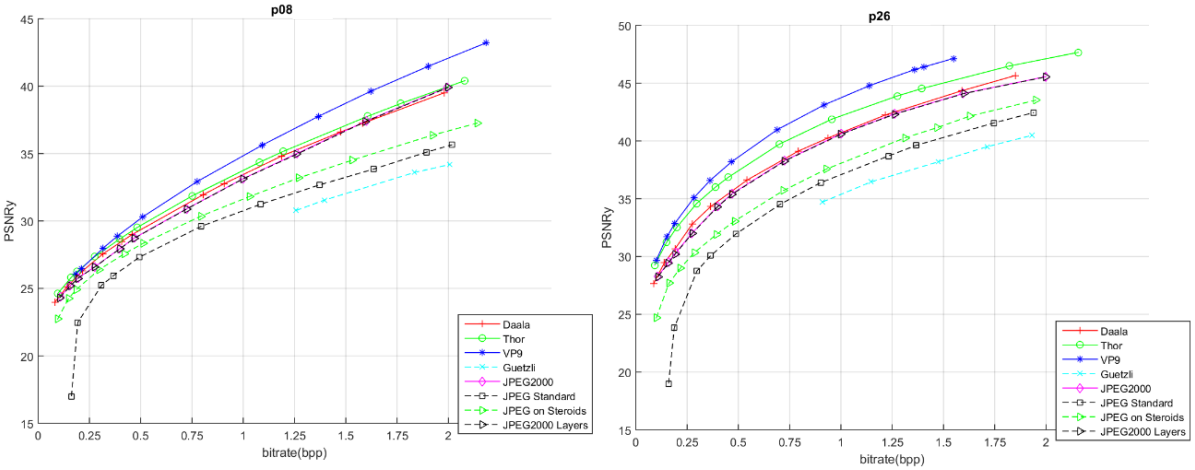


Figure 65 – PSNR$_Y$ metric comparison for *p08* (left) and *p26* (right) compression – medium spatial resolution set.

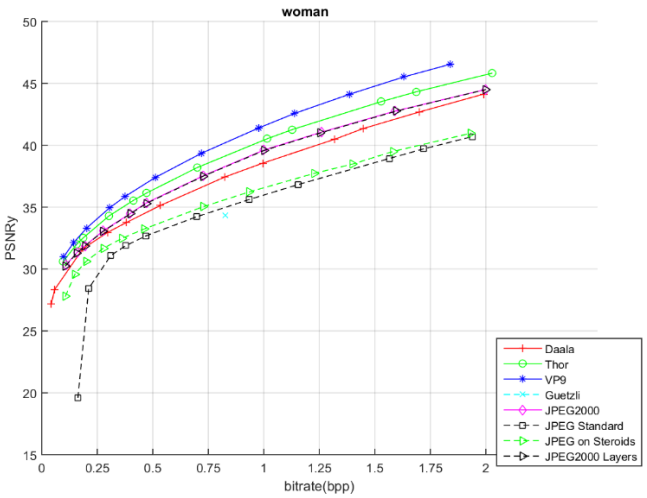


Figure 66 – PSNR$_Y$ metric comparison for *woman* compression (medium spatial resolution set).

SSIM$_Y$ metric – medium spatial resolution set

The results for the SSIM$_Y$ metric are represented below, from Figure 67 to Figure 70.

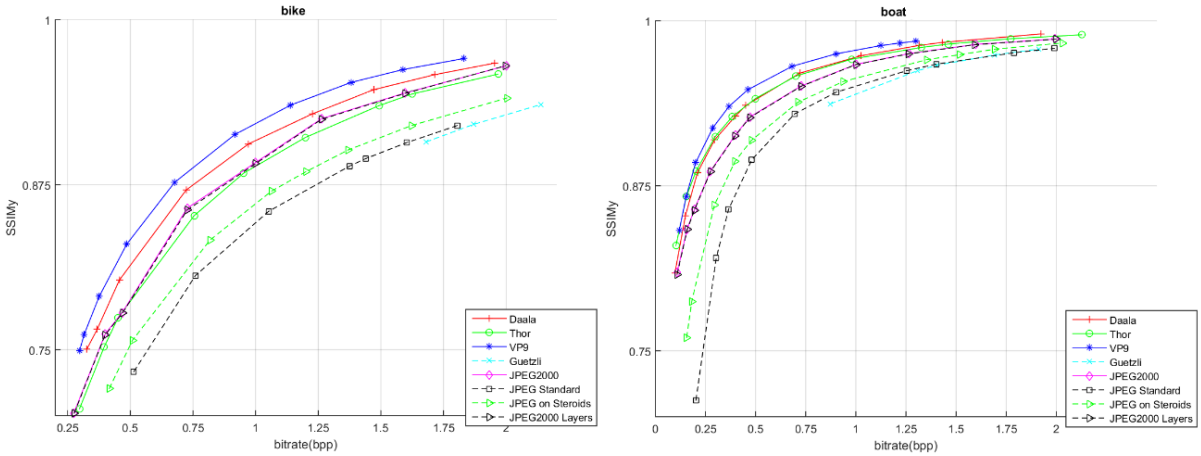


Figure 67 – $SSIM_Y$ metric comparison for *bike* (left) and *boat* (right) compression – medium spatial resolution set.

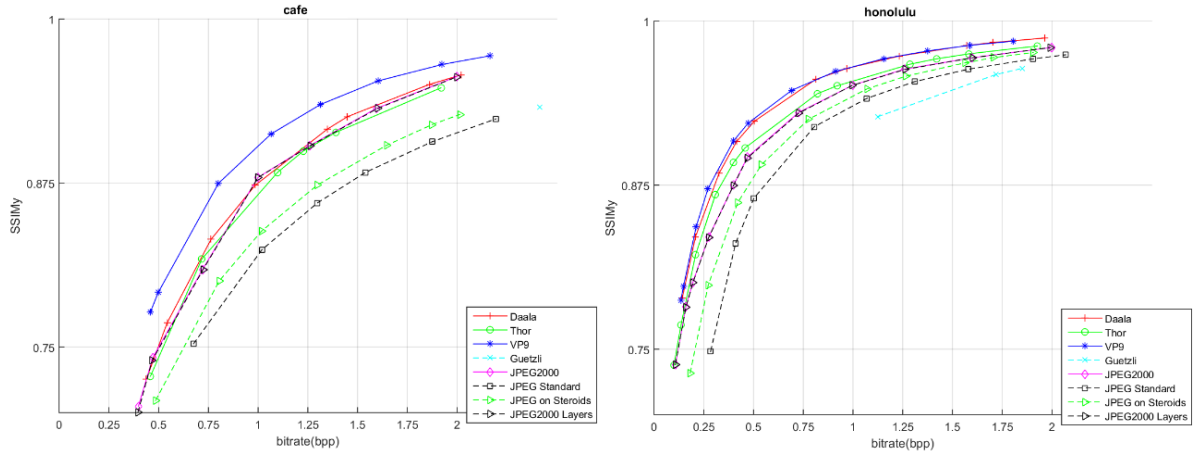


Figure 68 – $SSIM_Y$ metric comparison for *cafe* (left) and *honolulu* (right) compression – medium spatial resolution set.

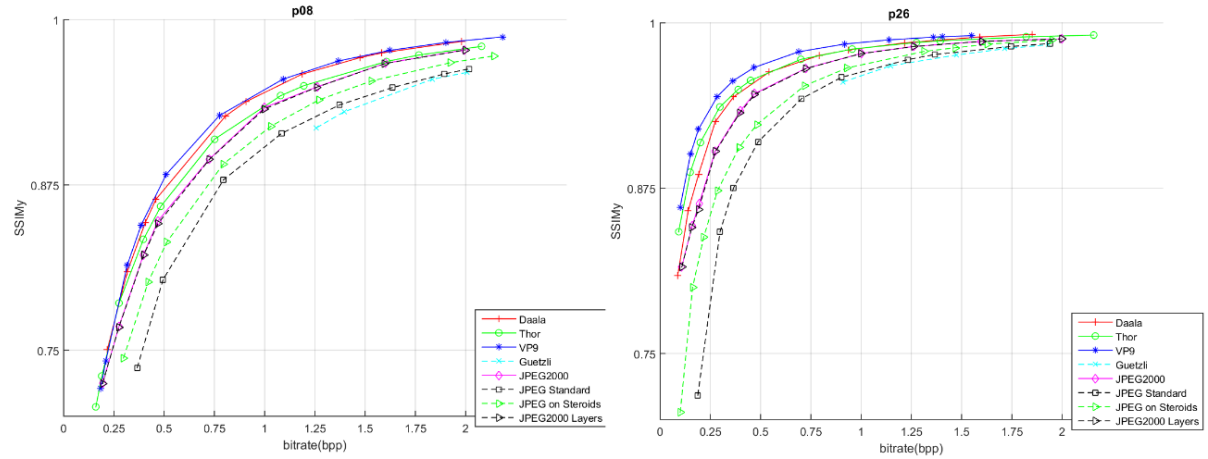


Figure 69 – $SSIM_Y$ metric comparison for *p08* (left) and *p26* (right) compression – medium spatial resolution set.

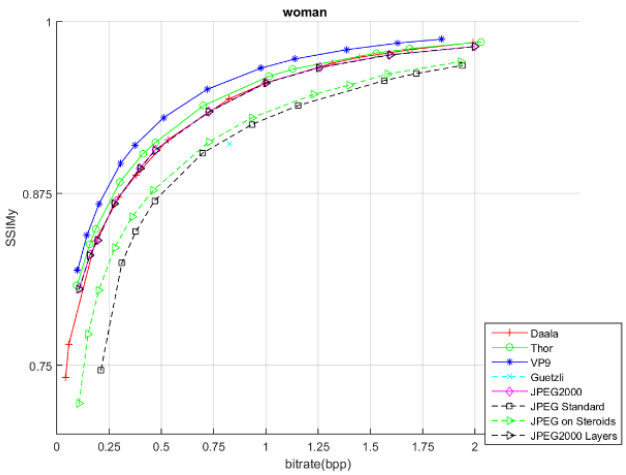


Figure 70 – SSIM$_Y$ metric comparison for *woman* compression – medium spatial resolution set.

PSNR$_Y$ metric – low spatial resolution set

The results for the PSNR$_Y$ metric are represented below, from Figure 71 to Figure 74.

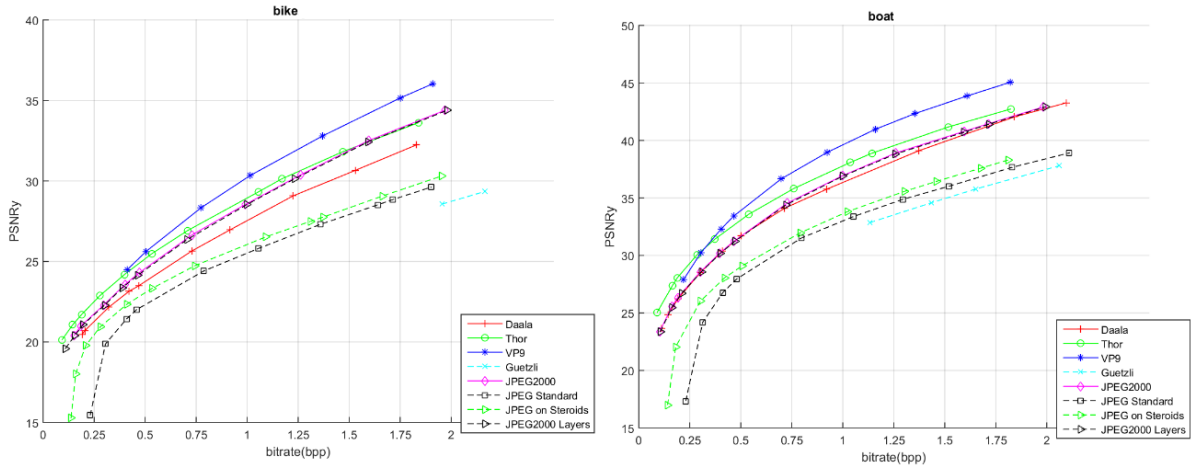


Figure 71 – PSNR$_Y$ results comparison for *bike (left)* and *boat (right)* compression – low spatial resolution set.

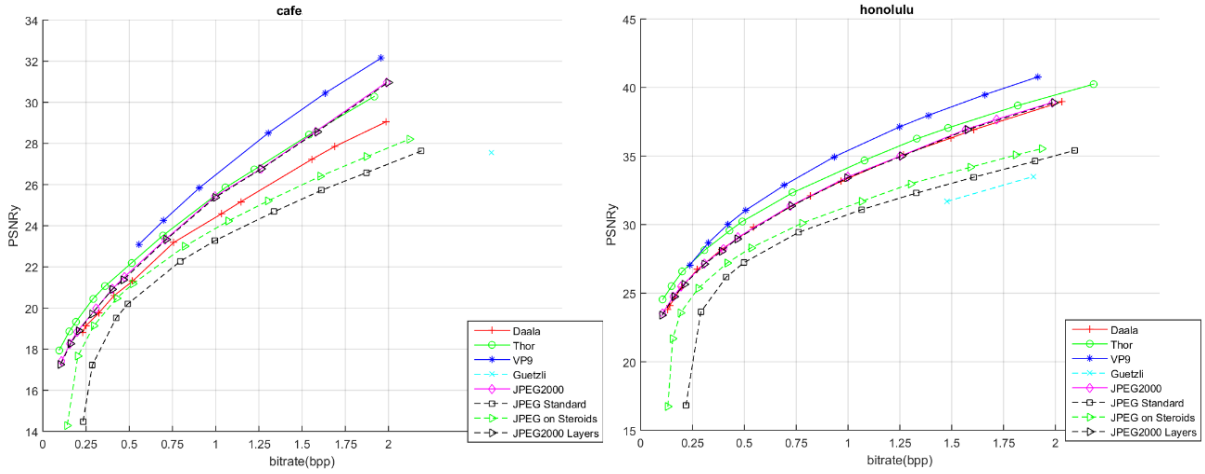


Figure 72 – PSNR$_Y$ results comparison for *cafe* (left) and *honolulu* (right) compression – low spatial resolution set.

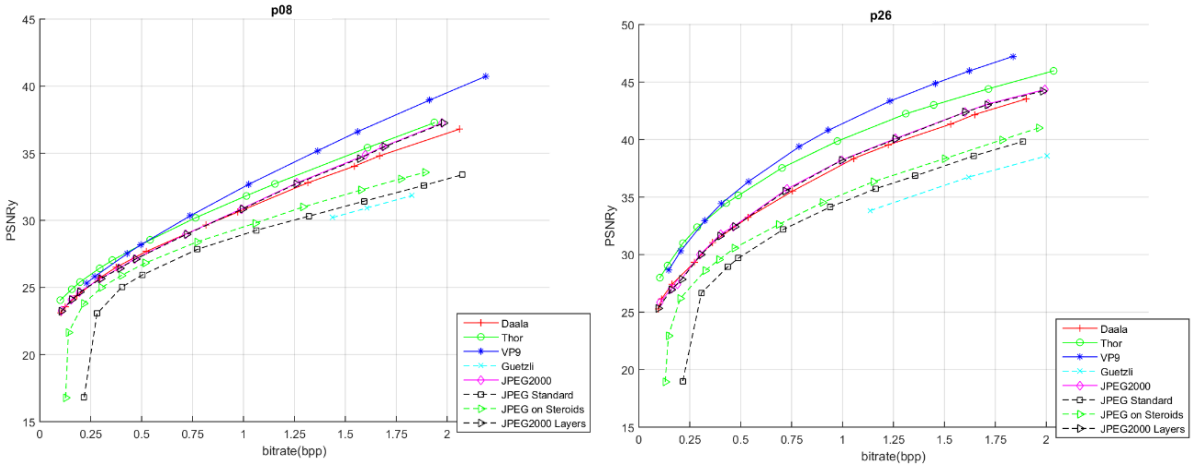


Figure 73 – $PSNR_Y$ results comparison for *p08* (left) and *p26* (right) compression – low spatial resolution set.

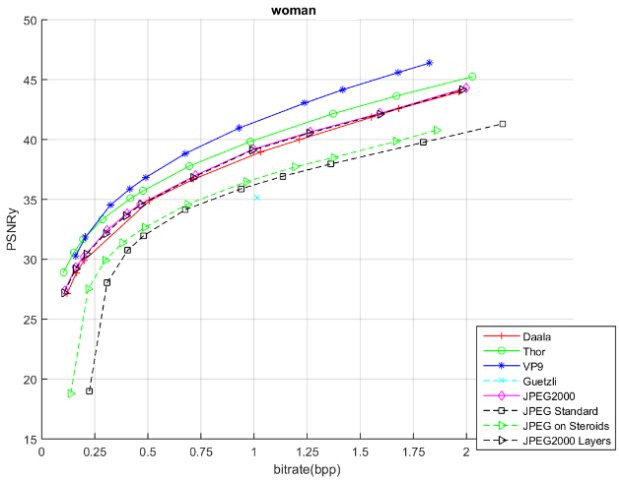


Figure 74 – $PSNR_Y$ results comparison for *woman* compression – low spatial resolution set.

$SSIM_Y$ metric – low spatial resolution set

The results for the SSIM metric are represented below, from Figure 75 to Figure 78.

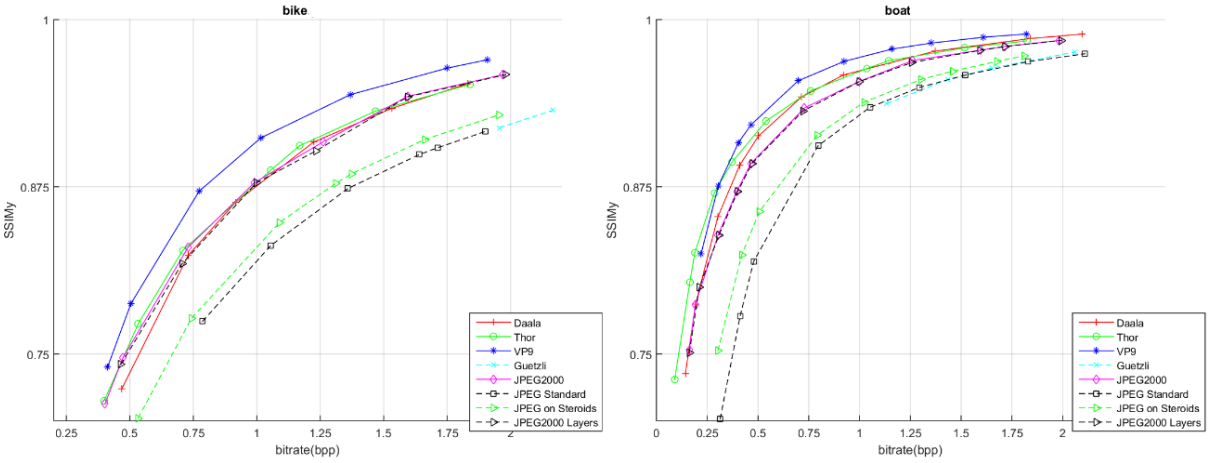


Figure 75 – $SSIM_Y$ results comparison for *bike* (left) and *boat* (right) compression – low spatial resolution set.

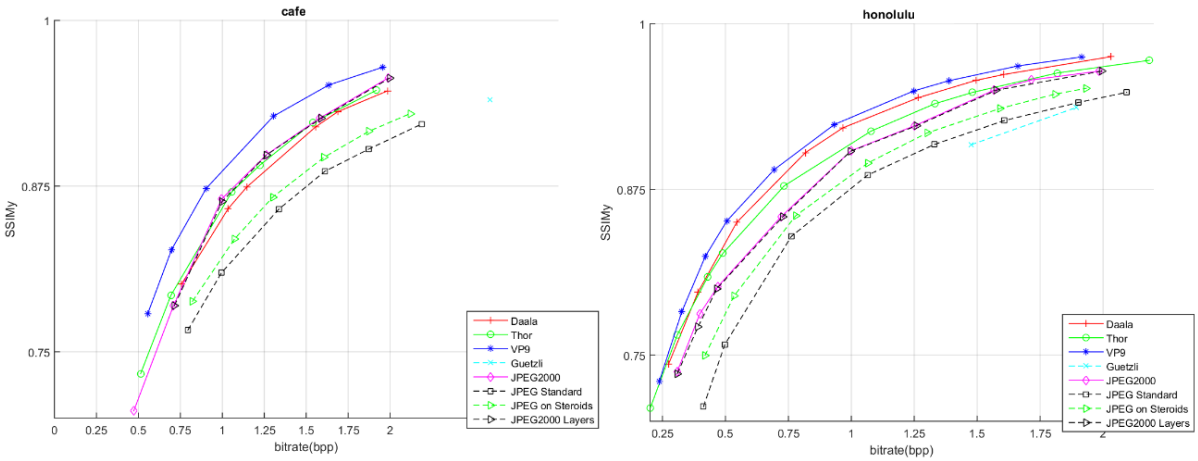


Figure 76 – $SSIM_Y$ results comparison for *cafe* (left) and *honolulu* (right) compression – low spatial resolution set.

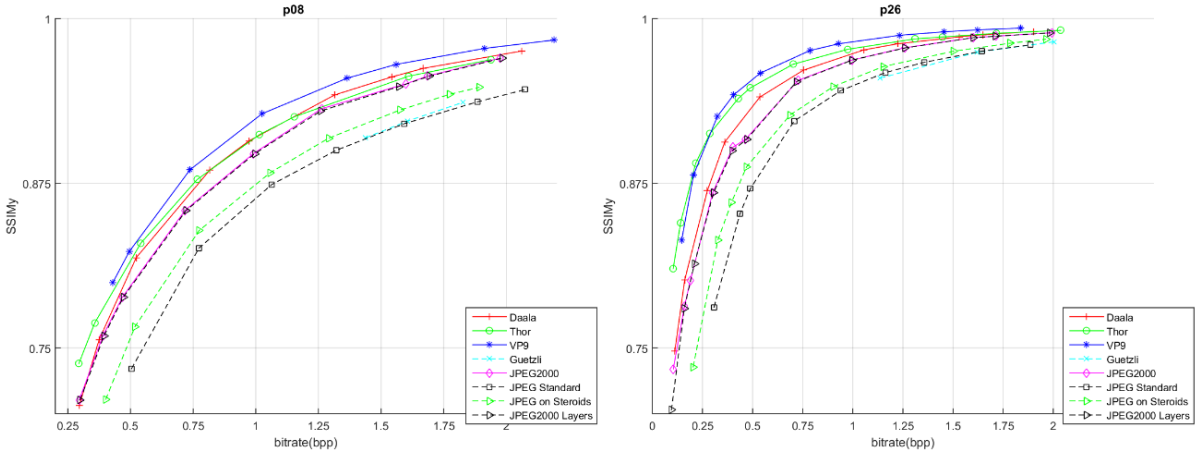


Figure 77 – $SSIM_Y$ results comparison for *p08* (left) and *p26* (right) compression – low spatial resolution set.

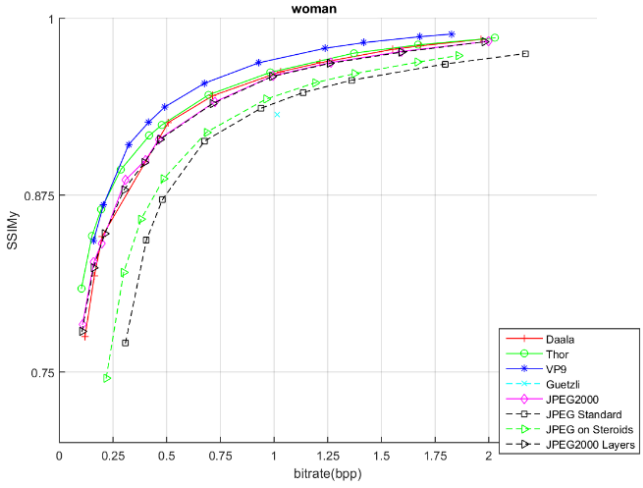


Figure 78 – $SSIM_Y$ results comparison for *woman* compression – low spatial resolution set.

A.2 Multiple images per codec

The results for *specific codec for multiple images* representation are presented below, for both spatial resolution sets.

PSNR$_Y$ metric – medium spatial resolution set

The results for the PSNR$_Y$ metric are represented below, from Figure 79 to Figure 82.

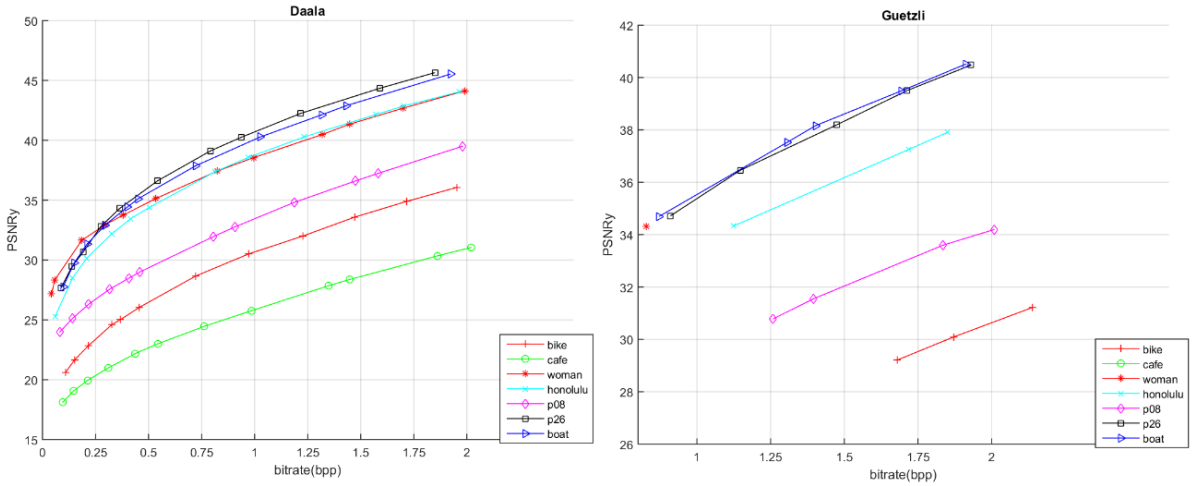


Figure 79 – PSNR$_Y$ results comparison for *Daala* (left) and *Guetzli* (right) – medium spatial resolution set.

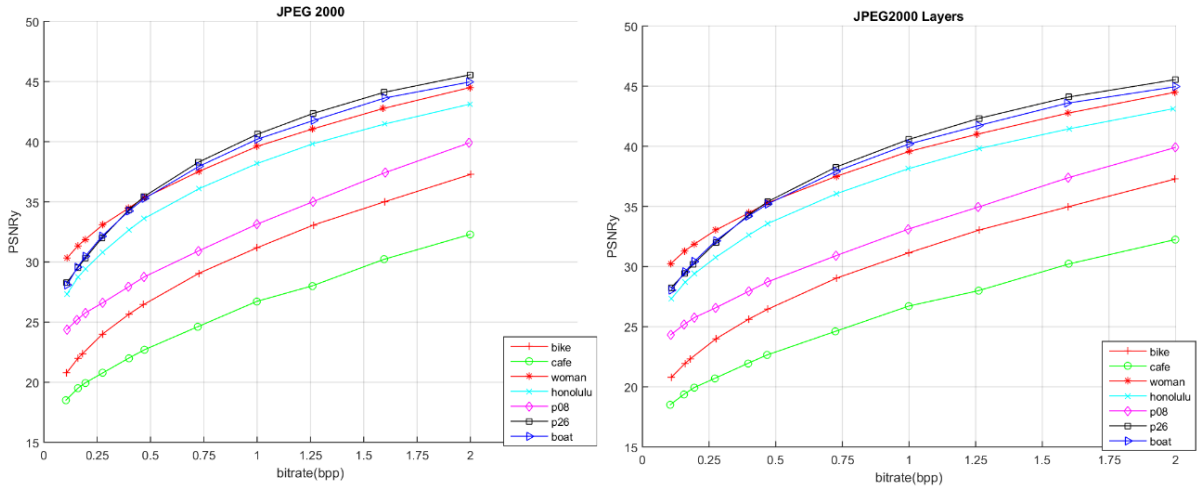


Figure 80 – PSNR$_Y$ results comparison for *JPEG 2000* (left) and *JPEG 2000 Layers* (right) – medium spatial resolution set.

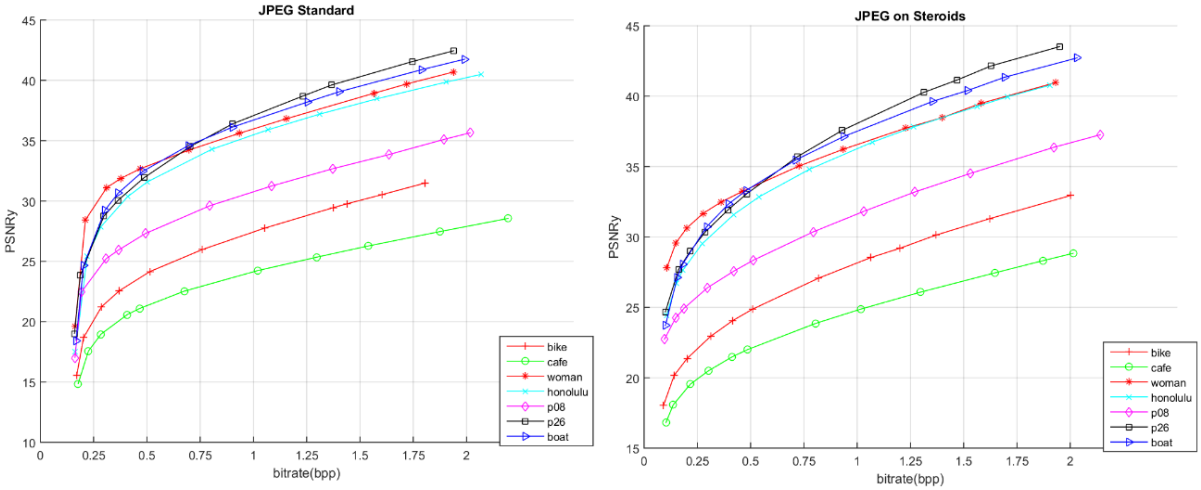


Figure 81 – PSNR$_Y$ results comparison for *JPEG Standard* (left) and *JPEG on Steroids*.

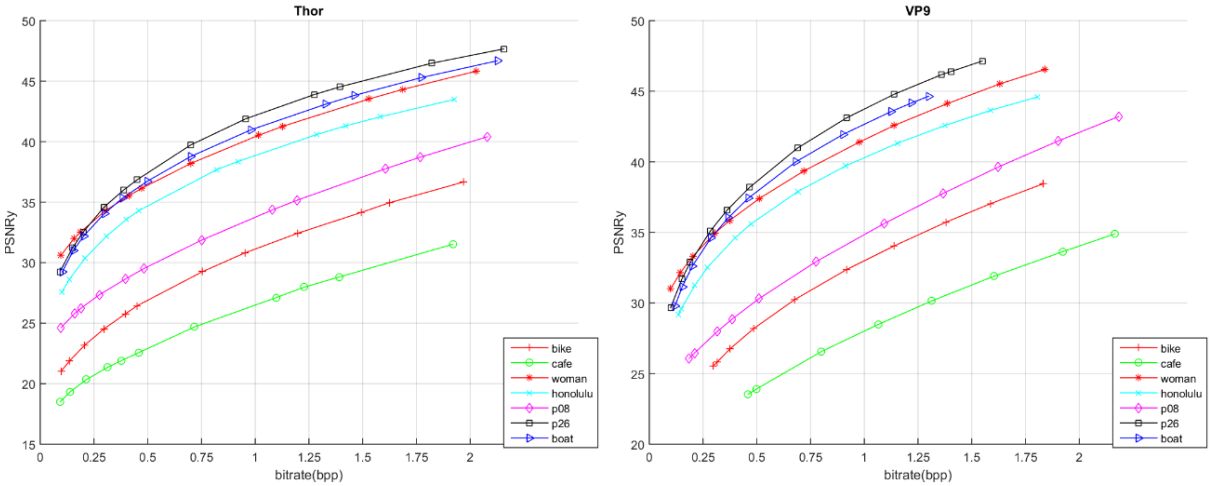


Figure 82 – PSNR$_Y$ results comparison for *Thor* (left) and *VP9* (right) – medium spatial resolution set.

SSIM$_Y$ metric – medium spatial resolution set

The results for the SSIM metric are represented below, from Figure 83 to Figure 86.

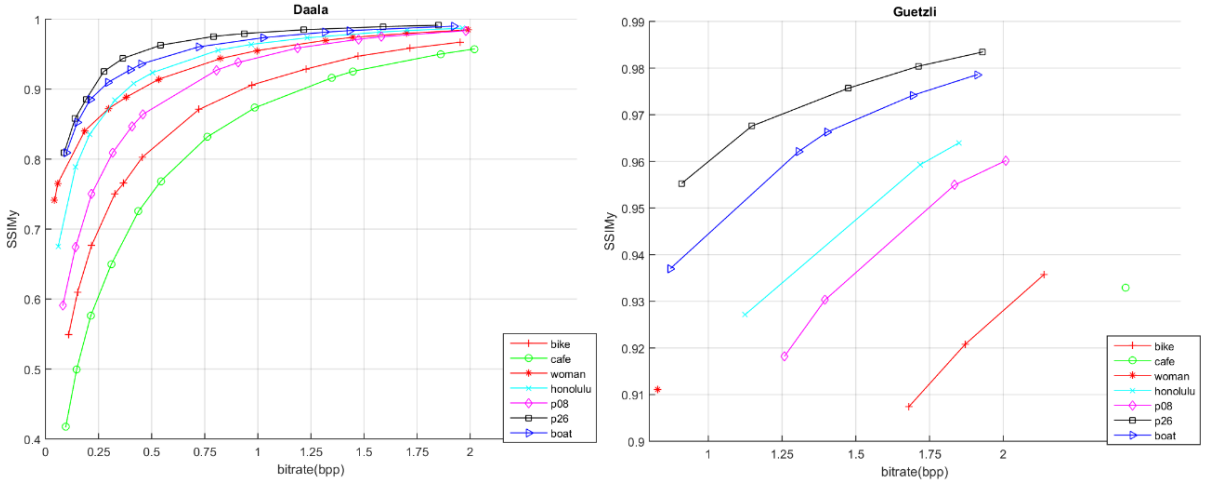


Figure 83 – SSIM$_Y$ results comparison for *Daala* (left) and *Guetzli* (right) – medium spatial resolution set.

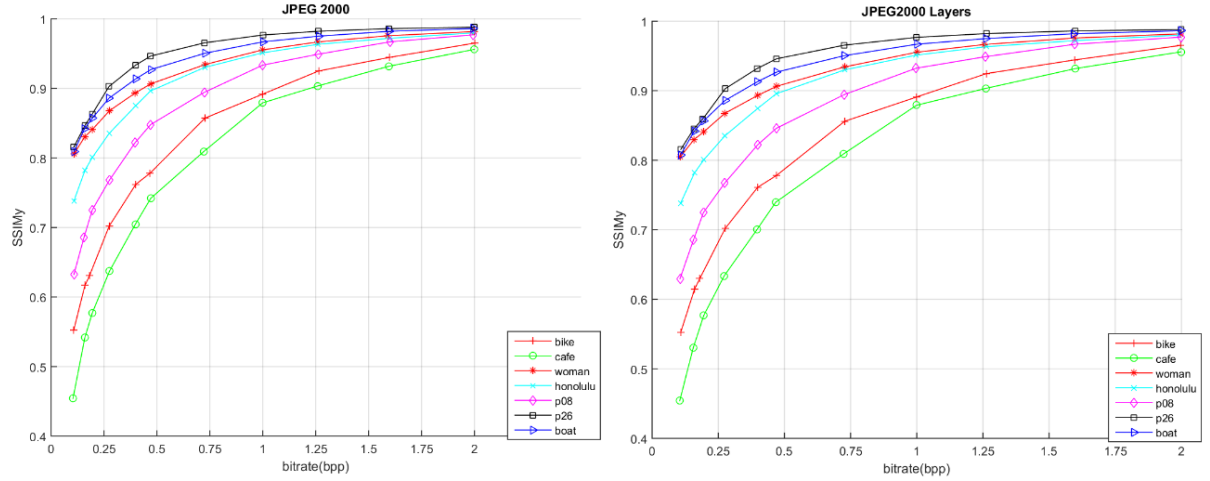


Figure 84 – SSIM$_Y$ results comparison for *JPEG 2000* (left) and *JPEG 2000 Layers* (right) – medium spatial resolution set.

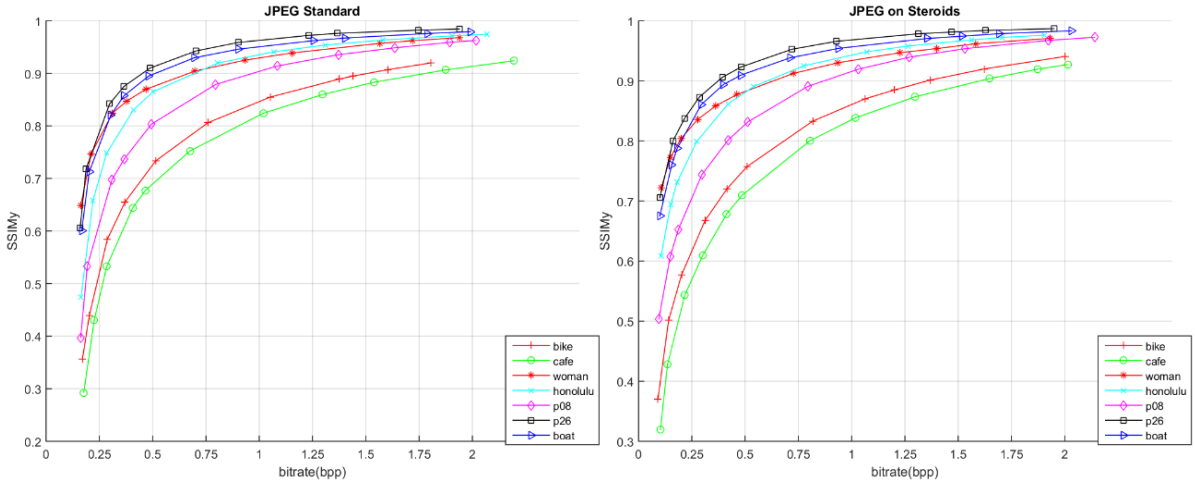


Figure 85 – SSIM$_Y$ results comparison for *JPEG Standard* (left) and *JPEG on Steroids* (right) – medium spatial resolution set.

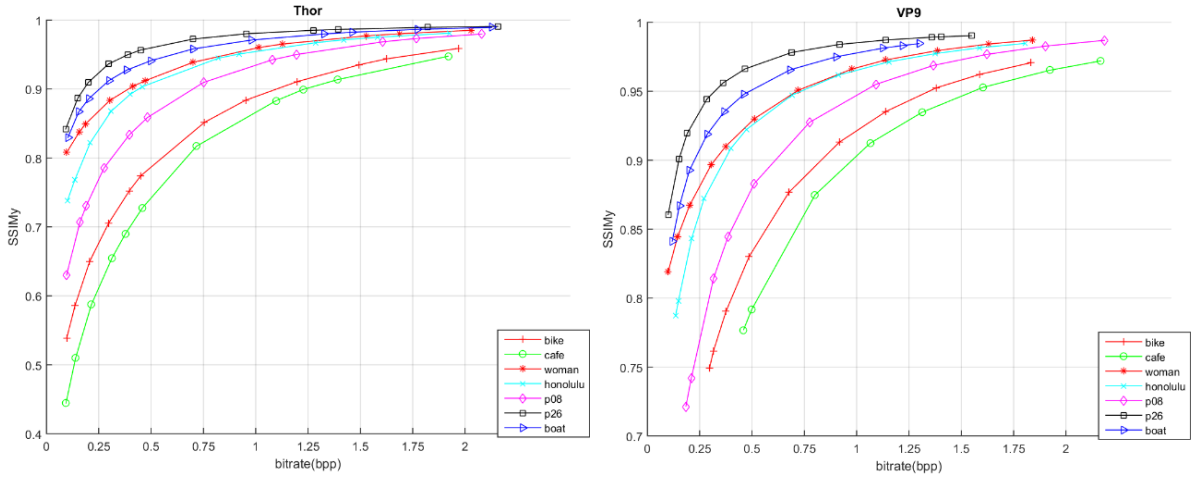


Figure 86 – SSIM$_Y$ results comparison for *Thor* (left) and *VP9* (right) – medium spatial resolution set.

PSNR$_Y$ metric – low spatial resolution set

The results for the PSNR$_Y$ metric are represented below, from Figure 87 to Figure 90.

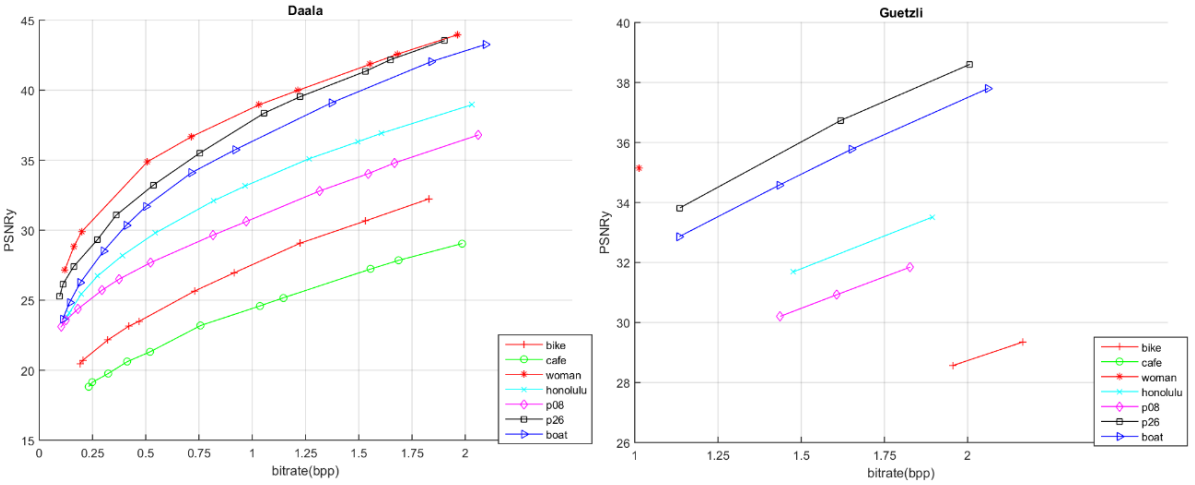


Figure 87 – PSNR$_Y$ results comparison for *Daala* (left) and *Guetzli* (right) – low spatial resolution set.

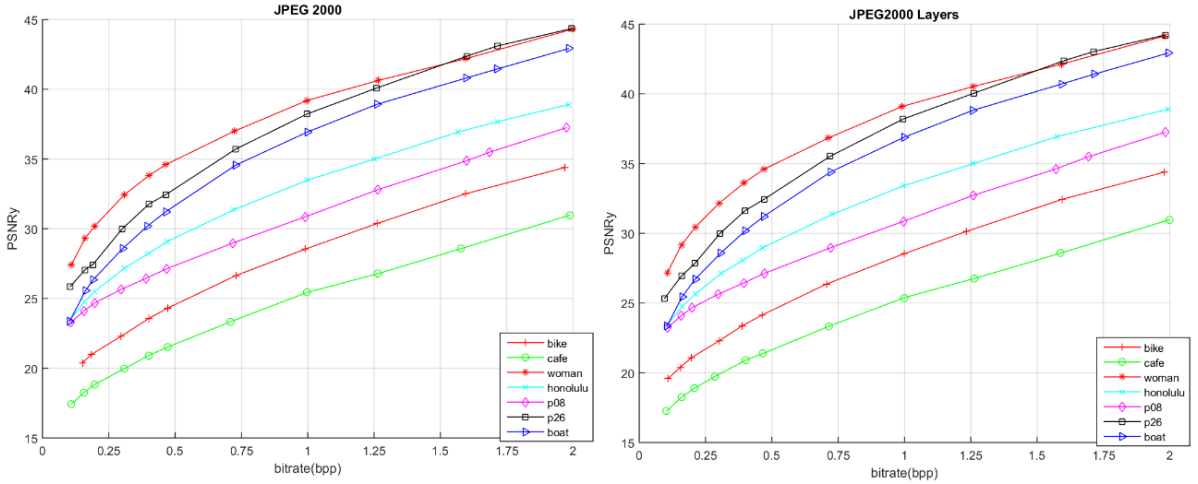


Figure 88 – PSNR$_Y$ results comparison for *JPEG 2000* (left) and *JPEG 2000 Layers* (right) – low spatial resolution set.

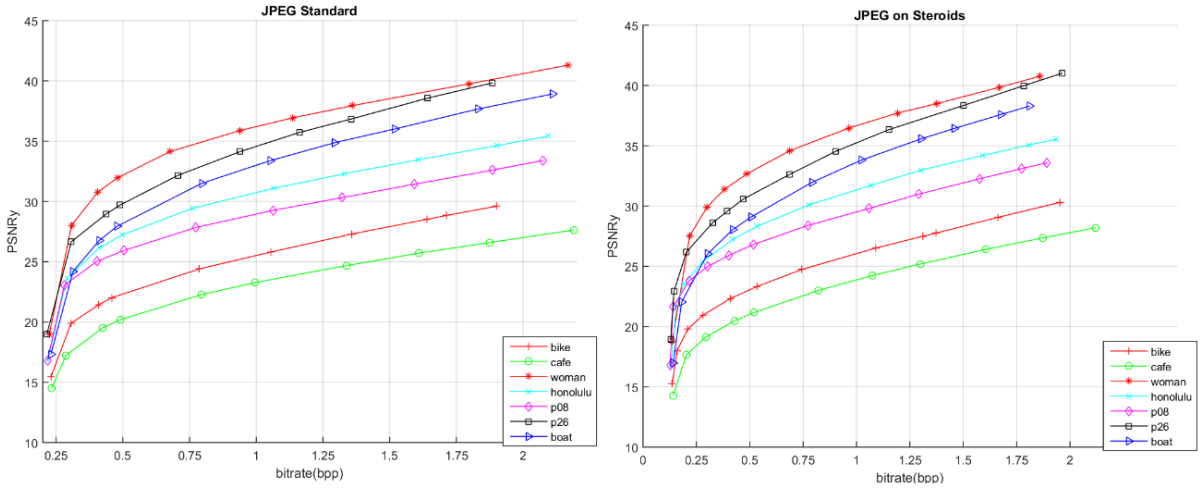


Figure 89 – PSNR$_Y$ results comparison for *JPEG Standard* (left) and *JPEG on Steroids* (right) – low spatial resolution set.

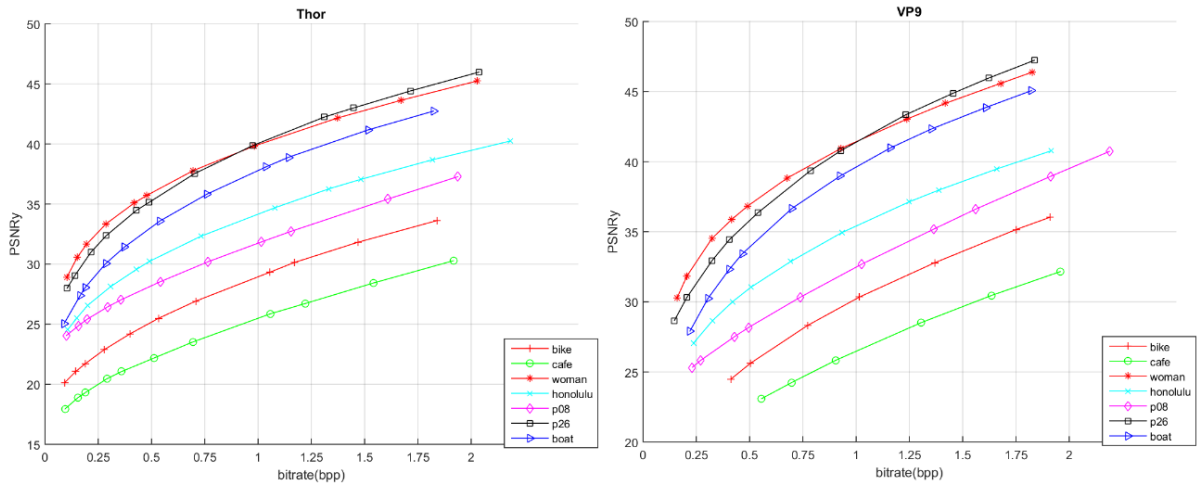


Figure 90 – PSNR$_Y$ results comparison for *Thor* (left) and *VP9* (right) – low spatial resolution set.

SSIM$_Y$ metric – low spatial resolution set

The results for the SSIM$_Y$ metric are represented below, from Figure 91 to Figure 94.

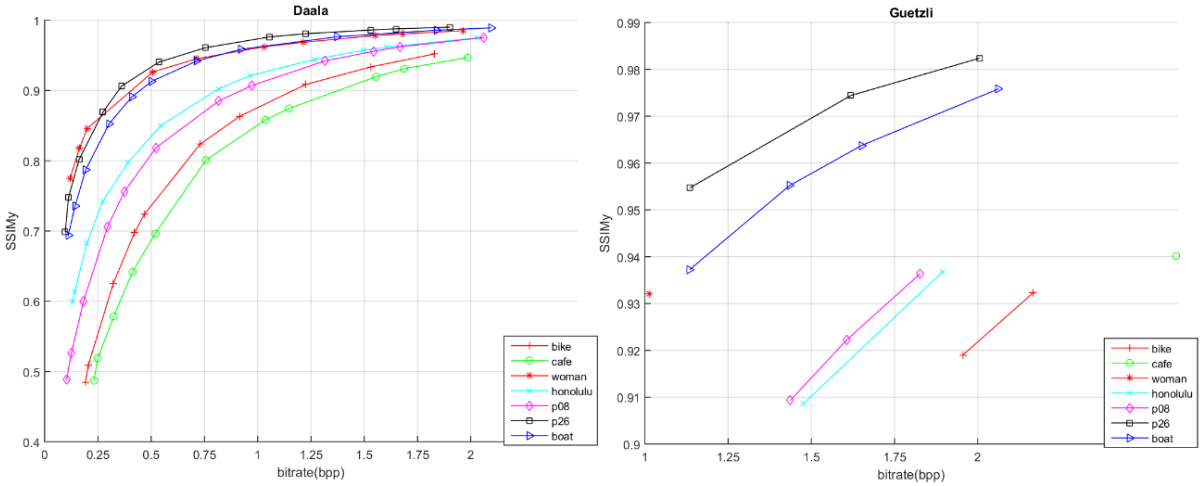


Figure 91 – SSIM$_Y$ results comparison for *Daala* (left) and *Guetzli* (right) – low spatial resolution set.

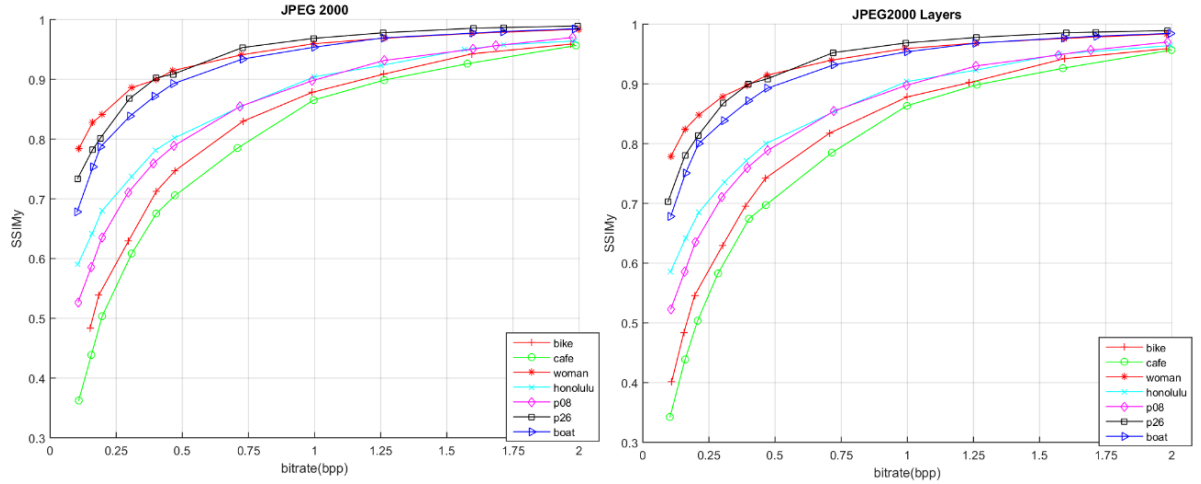


Figure 92 – PSNR$_Y$ results comparison for *JPEG 2000* (left) and *JPEG 2000 Layers* (right) – low spatial resolution set.

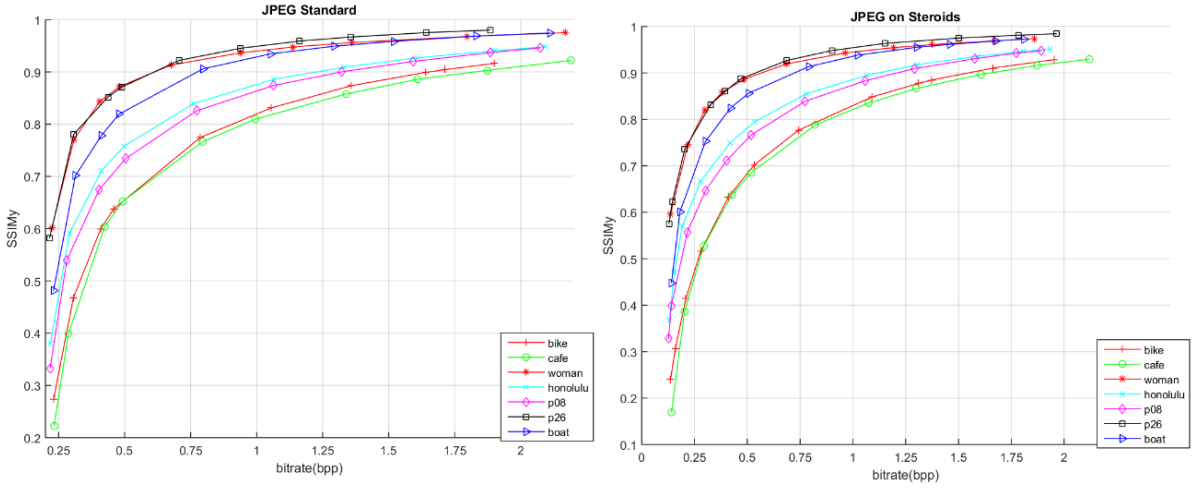


Figure 93 – SSIM$_Y$ results comparison for *JPEG Standard* (left) and *JPEG on Steroids* (right) – low spatial resolution set.

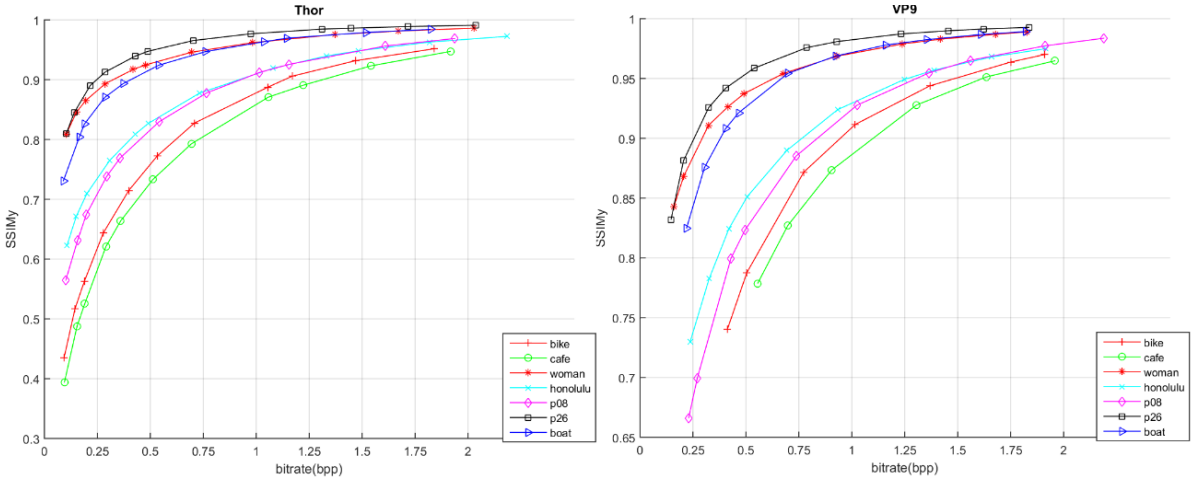


Figure 94 – SSIM$_Y$ results comparison for *Thor* (left) and *VP9* (right) – low spatial resolution set.

Appendix B

B. Image Compression Complexity Performance Assessment

This appendix reports the complexity assessment results obtained for the conditions presented in Section 2.2.6.2. A comparison of the compressing times for each codec is presented below, for both spatial resolution sets. Due to the extremely high processing times (approximately 3 and 10 seconds for the low and medium spatial resolution set, respectively), Guetzli results could not be effectively represented in the same charts. These measurements were achieved using a PC with an Intel Core i7-3770 CPU @ 3.40GHz – 3.90GHz.

Compressing time comparison – medium spatial resolution set

For the medium spatial resolution set, the complexity assessment results are presented below.

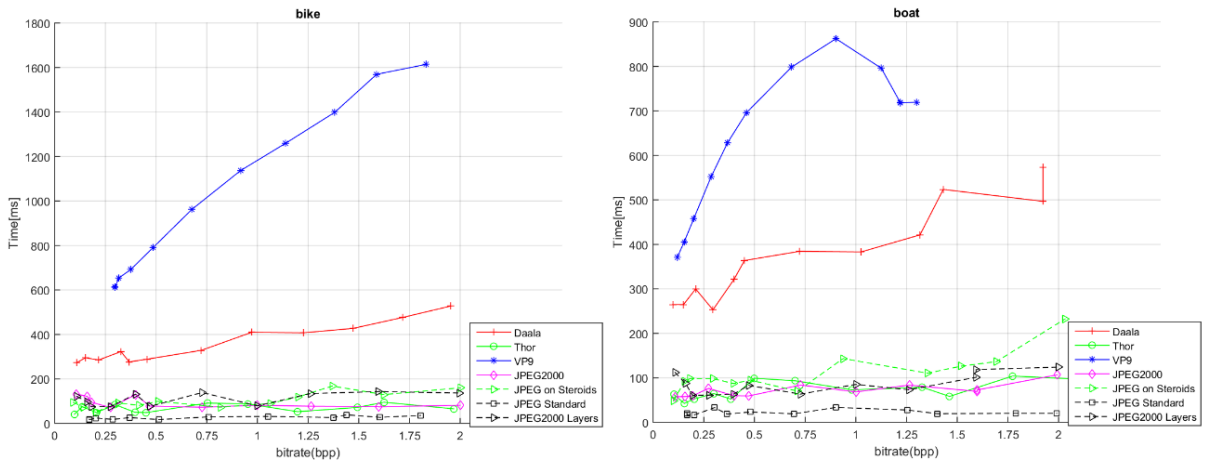


Figure 95 – Compressing times comparison for *bike* (left) and *boat* (right).

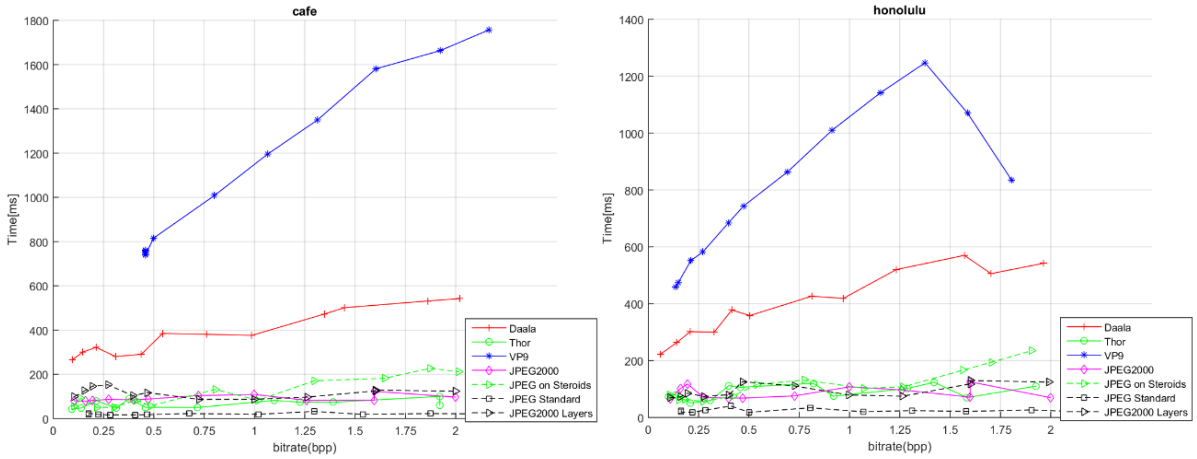


Figure 96 – Compressing times comparison for *cafe* (left) and *honolulu* (right).

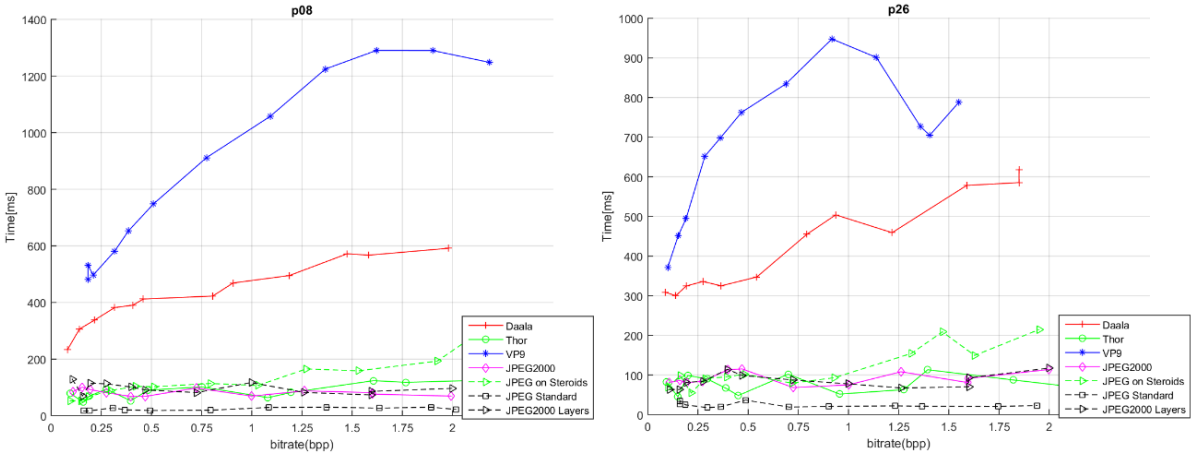


Figure 97 – Compressing times comparison for *p08* (left) and *p26* (right).

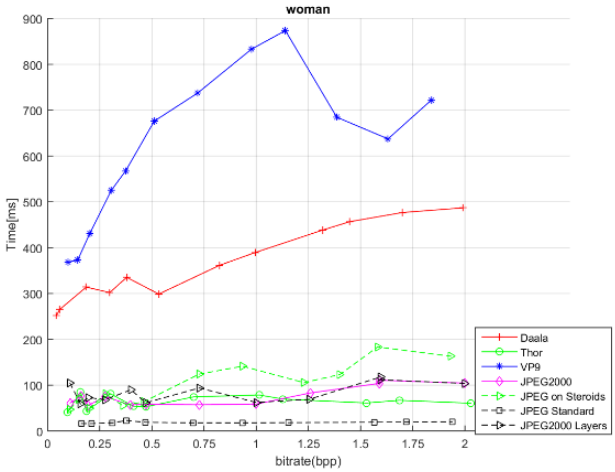


Figure 98 – Compressing times comparison for *woman*.

Compressing time comparison – low spatial resolution set

For the medium spatial resolution set, the complexity assessment results are presented below.

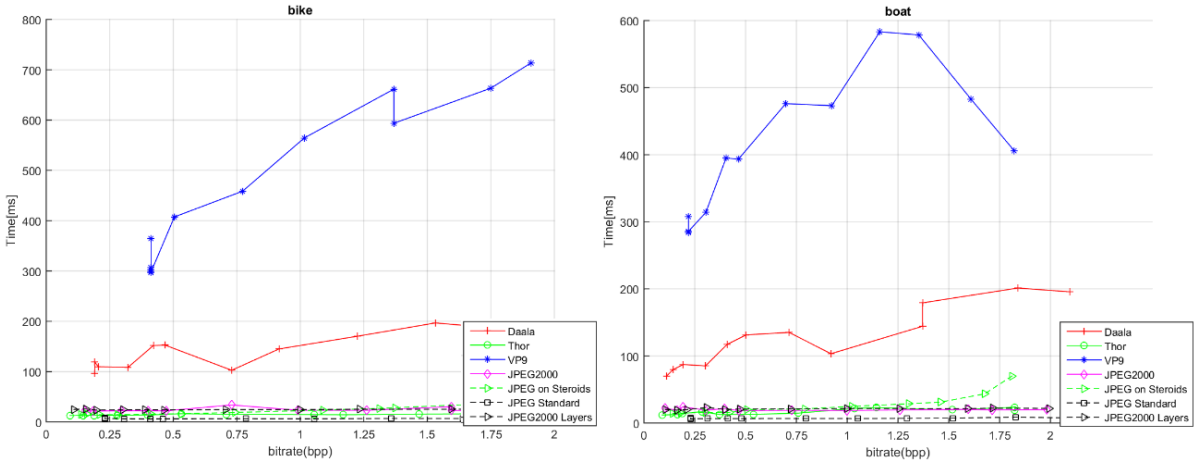


Figure 99 – Compressing times comparison for *bike* (left) and *boat* (right).

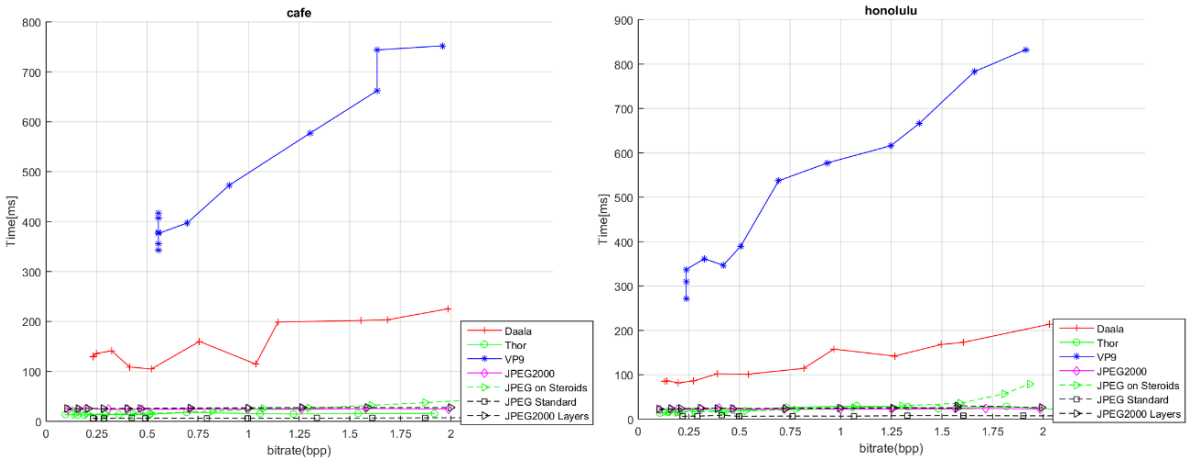


Figure 100 – Compressing times comparison for *cafe* (left) and *honolulu* (right).

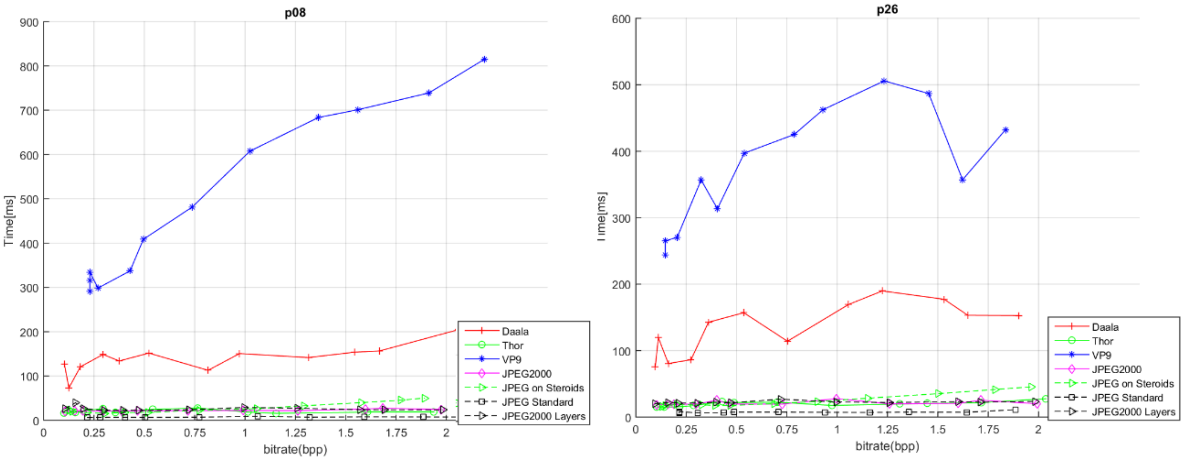


Figure 101 – Compressing times comparison for *p08* (left) and *p26* (right).

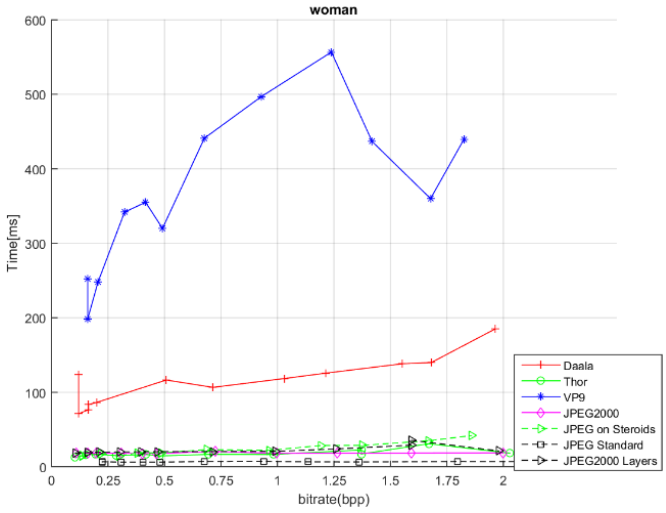


Figure 102 – Compressing times comparison for *woman*.

Appendix C

C. Proposed Solution Demonstration

From the start, the purpose of this work is the design and development of a proof-of-concept prototype capable of detecting relevant fishing events, taking place on a vessel, and transmitting highly compressed images classified as relevant by the detection algorithm. This proof-of-concept prototype should demonstrate its usability in the relevant field and fulfil the requirements presented in Section 3.4.2. To prove that the proof-of-concept prototype works in the desired way, a presentation followed by a demonstration was made at *Xsealence* premises, the company sponsoring this work. The demonstration consisted in showing the audience the event detection algorithm developed while working using visual data acquired from a given test video displayed on a monitor, thus also simulating the acquisition process. When the detection algorithm classifies an event as relevant, an encoded frame is automatically sent to the Control Center as described in Section 4.2. In the demo, this frame is also displayed to the audience.

Initially, all necessary processing was intended to be made by the Single Board Computer (SBC) available in the VMS unit, Monicap. However, this unit has rather limited capabilities making it unable to run complex algorithms such as the deep Convolutional Neural Network, AlexNet, adopted in this work. To work around this problem, it was decided to use an external processing board, in this case a Raspberry Pi [158].

The architecture of the system used for the demonstration is represented in Figure 103. The design and implementation of the demo was made with the help of *Xsealence*, which took care of the Monicap, GPRS and Control Center modules, not central for the work developed in this Thesis.

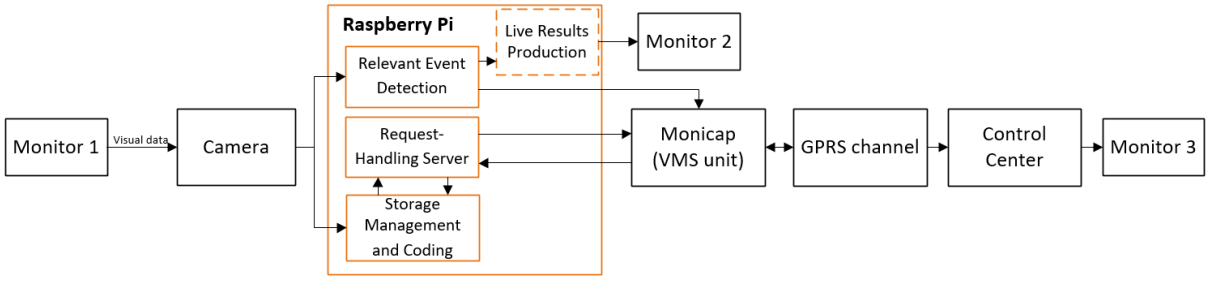


Figure 103 – Architecture of the proof-of-concept system demonstrated to *Xsealence*. The blocks in orange were developed by the author of this Thesis.

The main elements in the system architecture adopted for the demonstration were:

- **Monitor 1** – Used to mimic the environment at a fishing vessel by displaying a fishing test sequence.
- **Monitor 2 –** Used to display the decoded version of the frames selected for transmission within a relevant fishing event.

- **Monitor 3** – Used to display the detected event representative image received from Monicap by the Control Center.
- **Camera** – Used to capture visual data from Monitor 1. The camera used was a Raspberry Pi Camera, version 1.3 (see Figure 104, (b)) which was directly connected to the Raspberry Pi via a Camera Serial Interface (CSI) port [159] and programmed to run in parallel, capturing images at a constant frame rate of 1 image per second. These images are acquired with a spatial resolution of 640x480 pixels.
- **Raspberry Pi –** The Raspberry Pi is a series of Single Board Computers (SBC) developed by the Raspberry Pi Foundation with the goal of promoting the teaching of basic computer science in schools and developing countries [158]. Since its launch, several models have been released. For this work, the Raspberry Pi 3 Model B was used (see Figure 104 (a)). This model provides a Quad Core 1.2GHz Broadcom BCM2837 64bit CPU and 1GB of RAM. The operating system used was the Debian-based Raspian Stretch [160], which was installed on a 64GB microSD card.

Figure 104 – (a) Raspberry Pi Model B; (b) Raspberry Pi Camera.

As stated earlier, the machine learning framework used was TensorFlow. Since the official repository [161] does not currently contain binaries built for ARM processors and unofficial ARM binaries include only earlier versions of TensorFlow that use too much memory (more than the available 1GB), the most recent and efficient version (1.5) was built from source code in [162] and installed into the Raspberry Pi. Since Python was the first language supported by TensorFlow and is currently the most used and with most available features [163], Python was the selected language used to program the Raspberry Pi Model B (version 3.5.2).

Given the software specifics detailed above, a script for image classification with the AlexNet network fine-tuned with the fishing dataset as detailed in Section 4.3.3 takes approximately 65 seconds to load (this includes importing necessary libraries, building AlexNet's graph and importing its weights). The time that AlexNet takes to process a single image/frame and output its relevant event probability is 0.88 seconds per image. This is an average value obtained by processing a batch of 60 images. The Raspberry Pi is responsible for four main tasks, which were all programmed into a single program:

  - **Relevant event detection** – The relevant event detection process works as described in Section 4.2, by continuously classifying the most recent frame received from the camera. Although classifying a single image yields an average classification time of 0.88 seconds, the required computing power to run the whole program, this means the four main tasks for which the Raspberry Pi is responsible, increases its classification time to approximately 1.4 seconds. If the results are not displayed in Monitor 2, the classification time falls to

approximately 1.0 seconds. As detailed in Section 4.2, the moment a relevant event is detected, a low quality frame is automatically coded and transmitted to the Control Center.

- **Storage management and coding** – The Raspberry Pi continuously receives and stores frames obtained from the camera (whether in the presence of a relevant fishing event or not) on the microSD card. As explained in Section 3.4.1, the frames are encoded with 3 different qualities: 1) good quality (using the OpenCV JPEG encoder with default quality) with a spatial resolution of 640x480 – used to store visual data within the VMS unit for eventual retrieval by fishing authorities upon the arrival of the fishing boat to a port; 2) medium and 3) low quality for eventual satellite transmission, both with a spatial resolution of 320x240. Since for the last two these qualities, the goal is to compress as much as possible, either the JPEG on Steroids or JPEG 2000 codecs are used, since these were chosen as the best candidates for image compression in the context of this Thesis. Although only one of these encoders should be used, both are implemented in the program and can be easily switched from one to the other. These codecs are not used when saving in good quality because their software requires to first save the image in a lossless manner and only then encode with JPEG 2000 or JPEG on Steroids, which would increase the computational complexity. For the demonstration, the JPEG 2000 codec was used as it achieves better subjective image quality for the available rate.

  To lower the CPU usage, the medium and low quality frames are encoded only every ten seconds. Also, since all frames are continuously saved (at 1 fps), it is necessary to keep erasing the older ones, once enough time has passed to be sure that they do not belong to a relevant event. Thus, frames older than 2 minutes that do not belong to a relevant event are deleted.

  When a relevant event ends, corresponding frames are moved to a specific directory for that relevant event, and a new entry is recorded into the history file. To keep track of the frames that are transmitted, a transmission record file is used with one entry per transmitted image.

- **Request-handling server** – Responsible for listening requests coming from the Control Center. At startup, the program opens a socket server to listen for incoming connections. This module waits until receiving a message of 32 bytes from the Control Center, which contains the information about the specific request. The structure of the request messages coming from the Control Center is as follows: "*-keyword additional_field*", where *keyword* refers to the request in question and *additional_field* refers to eventual needed information regarding that specific request (e.g. number of images in an event time-lapse). Spaces are added until reaching the desired 32 bytes message size. A more detailed description of how the Raspberry Pi handles each Control Center request is as follows:
  - *Event time-lapse request* – Request for a time-lapse set of frames which must include the numbers of images that should be transmitted. Since the transmission of information between the VMS unit and the Control Center relies on fixed size 340 bytes messages, it is best to aggregate all images into one file instead of

transmitting one by one, since this would cause the last message of each image to waste a few bytes as the probability of the image size being a precise multiple of 340 bytes is low. Thus, the time-lapse set of images is zipped into a single file and the zip file transmitted.

- *Live visual data request* – Request for the most recent frame independently of its classification. The system verifies the last saved low quality frame and transmits it;
- *Event's frame request* – Request for an additional frame from the most recently detected event; the system searches for already transmitted frames and then randomly transmits a medium quality one that is temporally distant from those already transmitted by 20% of the total relevant event duration;
- *Event history request* – Request for the full history file which is located and transmitted back to the Control Center.

The *erasure/protect-from-erasure request* listed in Section 3.4.1 was not implemented.

- **Live results production** – This module was developed only for the demonstration. Since the purpose is to make a proof-of-concept, it is important to show how the detection algorithm responds as the video progresses. Thus, a live chart with the relevant event probability, as well as a window displaying what the deep learning framework is "seeing" at the moment and information regarding the state of the event detection algorithm (nothing happening, possible event detected, and event detected) are shown (see Figure 105 (a), (b) and (c), respectively).

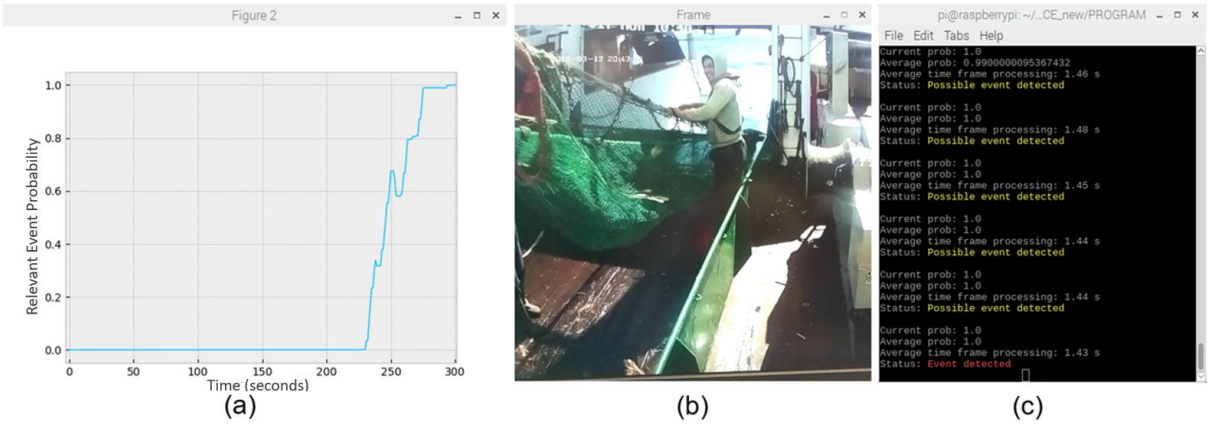


Figure 105 – (a) live chart with relevant event probability ,the chart has a width of 300 seconds (5 minutes) and is updated from right to left; (b) image currently processed by AlexNet; (c) state of the event detection algorithm.

- **Monicap** – Corresponds to the VMS unit from *Xsealence* to be installed in the vessel. Its function in this system is the data framing and wrapping of the information coming from the Raspberry Pi and the following transmission to the Control Center via an Iridium or GPRS channel. It also receives the requests from the Control Center and forwards them to the Raspberry Pi. This device has a port that is continuously listening for incoming connections and connected to the Raspberry Pi via USB using *Ethernet over USB*.
- **GPRS channel** – Responsible for establishing the connection between Monicap and the Control Center. For the demonstration, a GPRS channel was used instead of an Iridium one, essentially

because of the high costs of the Iridium communication. However, transmission tests using the Iridium channel were also successfully completed.

- **Control Center** – A laptop connected to the Control Center was used to decode and display the frames transmitted to the Control Center. A *print screen* of the interface used to retrieve and display the frames is shown in Figure 106 and a picture of the overall setup is shown in Figure 107.

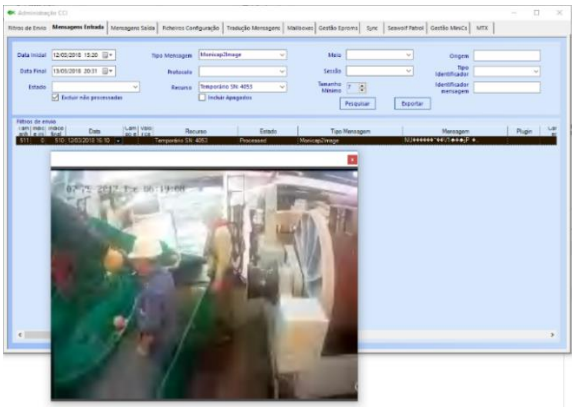


Figure 106 – JPEG 2000 compressed image received at the Control Center.

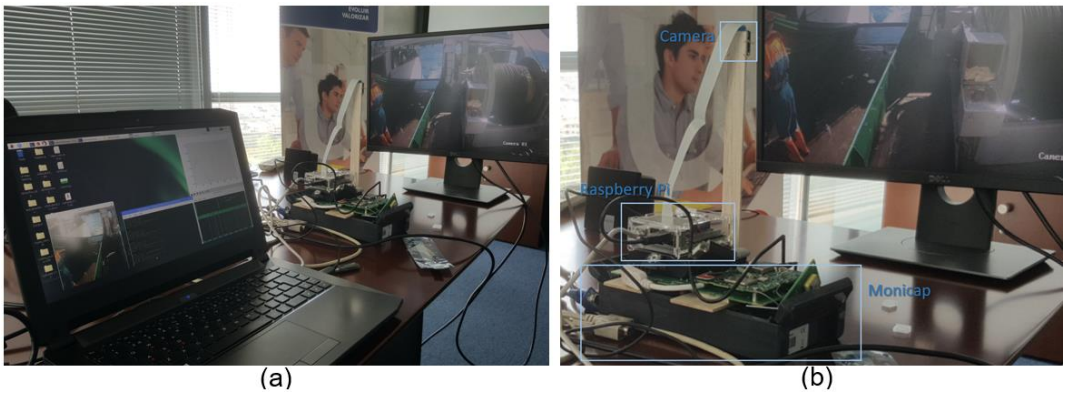


Figure 107 – (a) Setup; (b) Setup zoomed in.

The performance assessment made in Section 4.4.3 was achieved by classifying all frames depending on their relevance event probability. Since the acquisition frame rate of the videos from the fishing dataset is of 20 frames per second (fps), it can be said that the performance assessment was accomplished using a classification rate of also 20 fps, since all frames from the test videos were classified. Lower classification rates classify less images, and thus are bounded to provide a lower performance.

Due to the low computing power available, the Raspberry Pi is only able to provide a classification rate of approximately 1 fps, which may result in a reduced event detection rate performance. To understand and evaluate eventual performance differences regarding different classification rates, some ROC curves are presented in Figure 108 (a) and (b), thus allowing to compare the frame-level and event-level performances at 1 fps (approximated classification rate of the Raspberry Pi) and 20 fps (classification rate used in Section 4.4.3). The frame-level performance comparison represented in Figure 108 (a) uses a *sliding_window_width* parameter value of 20 seconds, which was the optimal value selected in Section 4.4.3.1, while the event-level comparison represented in Figure 108 (b) uses an *event_time_width* parameter value of 60 seconds, which was the optimal value selected in Section 4.4.3.2.

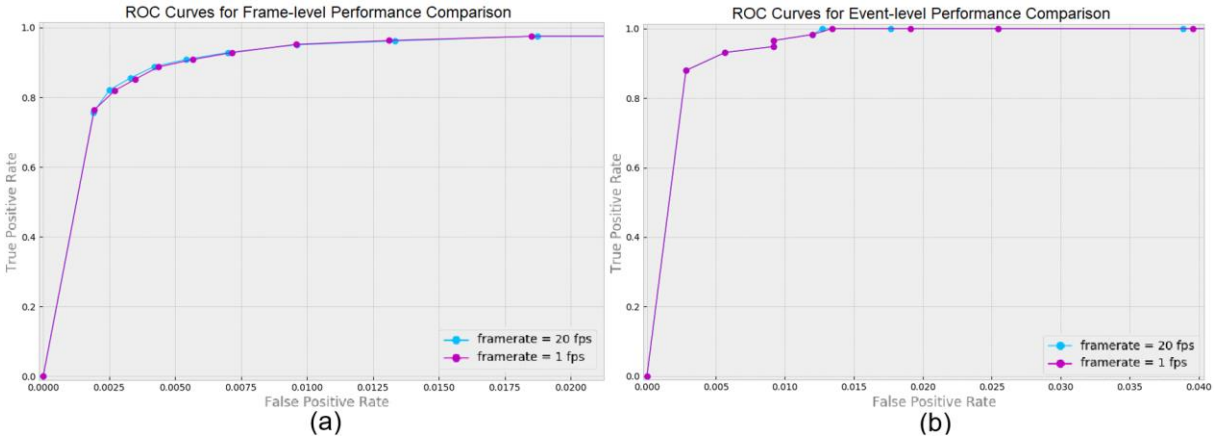


Figure 108 – Performance comparison between 20 and 1 frames per second. (a) Frame-level; (b) Event-level.

A quick look at the results in Figure 108 (a) shows that the frame-level performance is very similar for the two classification rates, thus allowing to expect that also the event-level performance is very similar. Figure 108 (b) shows the event-level performances for the two classification rates, confirming that they are in fact almost identical.

Appendix D

D. Proposed Solution Demonstration

The purpose of this appendix is to show some of the performance results as shown in Section 5.4, for the faces and landscapes datasets.

Landscape Dataset

(a) (b) (c)

(d) (e) (f)

Figure 109 – Landscape datasets images. Spatial resolutions for all image is 576x400.

Table 14 – PSNR and SSIM comparison for a JPEG on Steroids decoded image before and after only deblocking. The quality factor for JPEG coding ranges from 0 to 100; higher is better.

	PSNR (dB)			SSIM		
Quality factor (Q)	**No restoration**	**DnCNN restoration**	**Gain**	**No restoration**	**DnCNN restoration**	**Gain**
Q = 6	22.72	23.42	0.70	0.66	0.68	0.02
Q = 10	24.31	25.15	0.84	0.75	0.77	0.02
Q = 15	25.62	26.59	0.97	0.81	0.83	0.02
Q = 20	26.58	27.62	1.04	0.85	0.87	0.02
Q = 25	27.37	28.45	1.08	0.87	0.89	0.02
Q = 30	29.18	28.06	1.12	0.89	0.91	0.02

Table 15 – PSNR and SSIM comparison for bicubic interpolation, Lanczos [164] interpolation with a kernel size of 3 interpolation as implemented in matlab, and DnCNN super-resolution enhancement.

	PSNR			SSIM		
Scaling Factor	**Bicubic**	**Lanczos3**	**DnCNN restoration**	**Bicubic**	**Lanczos3**	**DnCNN restoration**
2x	25.30	25.63	28.01	0.84	0.86	0.92
3x	23.01	23.17	24.62	0.73	0.74	0.81
4x	21.86	21.98	22.99	0.63	0.64	0.72

Faces Dataset

Figure 110 – Faces dataset images. Spatial resolution are: (a) 480x 480; (b) 400x576; (c) 576x400 (d) 400x576; (e) 576x400.

Table 16 – PSNR and SSIM comparison for a JPEG on Steroids decoded image before and after only deblocking. The quality factor for JPEG coding ranges from 0 to 100; higher is better.

	PSNR (dB)			SSIM		
Quality factor (Q)	**No restoration**	**DnCNN restoration**	**Gain**	**No restoration**	**DnCNN restoration**	**Gain**
Q = 6	26.69	27.57	0.88	0.68	0.72	0.05
Q = 10	28.21	29.02	0.81	0.75	0.78	0.03
Q = 15	29.45	30.25	0.80	0.80	0.83	0.03
Q = 20	30.35	31.22	0.87	0.84	0.86	0.02
Q = 25	31.06	31.95	0.89	0.86	0.87	0.01
Q = 30	31.67	32.57	0.91	0.87	0.89	0.01

Table 17 – PSNR and SSIM comparison for bicubic interpolation, Lanczos [164] interpolation with a kernel size of 3 interpolation as implemented in matlab, and DnCNN super-resolution enhancement.

	PSNR			SSIM		
Scaling Factor	**Bicubic**	**Lanczos3**	**DnCNN restoration**	**Bicubic**	**Lanczos3**	**DnCNN restoration**
2x	31.14	31.45	32.52	0.88	0.89	0.91
3x	29.09	29.28	30.28	0.82	0.82	0.85
4x	27.98	28.14	29.08	0.77	0.77	0.81

References

[1] "Wikipedia on Overfishing," [Online]. Available: http://ocean.nationalgeographic.com/ocean/explore/pristine-seas/critical-issues-overfishing/. [Accessed 13 April 2017].

[2] "Unsustainable fishing," [Online]. Available: http://wwf.panda.org/about_our_earth/blue_planet/problems/problems_fishing/. [Accessed 13 April 2017].

[3] "Commercial Fishing," [Online]. Available: http://www.ramweb.org/commercial-fishing.html. [Accessed 24 March 2017].

[4] "Banner on Philippine Purse Seine Vessel," [Online]. Available: http://www.greenpeace.org/seasia/ph/multimedia/slideshows/The-Vergene-Purse-Seiner/Banner-on-Philippine-Purse-Seine-Vessel/. [Accessed 12 April 2017].

[5] "The State of World Fisheries and Aquaculture," 2016. [Online]. Available: http://www.fao.org/3/a-i5555e.pdf. [Accessed 14 April 2017].

[6] D. Carrington, "The Guardian - Overfishing causing global catches to fall three times faster than estimated," [Online]. Available: https://www.theguardian.com/environment/2016/jan/19/overfishing-causing-global-catches-to-fall-three-times-faster-than-estimated. [Accessed 12 April 2017].

[7] "Pesca portuguesa é a que mais captura acima do recomendado pelos cientistas," December 2015. [Online]. Available: https://www.publico.pt/2015/12/02/economia/noticia/pesca-portuguesa-e-a-que-mais-captura-acima-do-recomendado-pelos-cientistas-1716267. [Accessed 13 April 2017].

[8] "The Common Fisheries Policy," [Online]. Available: https://ec.europa.eu/fisheries/cfp_en. [Accessed 23 March 2017].

[9] "Fishing Monitoring Systems: Past, Present and Future," [Online]. Available: http://www.imcsnet.org/imcs/docs/fishing_vessel_monitoring_systems_past_present_future.pdf. [Accessed 27 March 2017].

[10] F. Li, "Machine Learning in Computer Vision," Princeton University, [Online]. Available: https://www.cs.princeton.edu/courses/archive/spring07/cos424/lectures/li-guest-lecture.pdf. [Accessed 28 April 2018].

[11] "Wikipedia on Bag-of-words Model in Computer Vision," [Online]. Available: https://en.wikipedia.org/wiki/Bag-of-words_model_in_computer_vision. [Accessed 16 March 2018].

[12] S. Srinivas, R. Sarvadevabhatla, K. Mopuri, N. Prabhu, S. Kruthiventi, and R. Babu, "A Taxonomy of Deep Convolutional Neural Nets for Computer Vision," *Frontiers in Robotics and AI,* vol. 2, pp. 1-18, 2016.

[13] M. Maind, "Research Paper on Basic of Artificial Neural Network," *nternational Journal on Recent and Innovation Trends in Computing and Communication,* vol. 2, pp. 96-100, 2014.

[14] "CS231n Convolutional Neural Networks for Visual Recognition," Stanford University, [Online]. Available: http://cs231n.github.io/convolutional-networks/. [Accessed 13 April 2018].

[15] "Mathematical foundation for Activation Functions in Artificial Neural Networks," [Online]. Available: https://medium.com/autonomous-agents/mathematical-foundation-for-activation-functions-in-artificial-neural-networks-a51c9dd7c089. [Accessed 13 April 2018].

[16] "Deeply Supervised Nets," [Online]. Available: http://vcl.ucsd.edu/~sxie/2014/09/12/dsn-project/. [Accessed 14 April 2018].

[17] M. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *Computer Vision–ECCV 2014,* vol. 8689, pp. 818-833, 2014.

[18] Y. Lecun, "ICML 2013 tutorial on Deep Learning," New York University, [Online]. Available: https://cs.nyu.edu/~yann/talks/lecun-ranzato-icml2013.pdf. [Accessed 17 April 2018].

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems,* vol. 25, pp. 1097-1105, 2012.

[20] "Wikipedia on Softmax Function," [Online]. Available: https://en.wikipedia.org/wiki/Softmax_function. [Accessed 13 April 2018].

[21] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE,* vol. 86, p. 2278–2324, 1998.

[22] "P. Zemcík, “Analysis of Wear Debris through Classification Analysis of Wear Debris Through Classification,” 13th Int. Conf. ACIVS, no. October 2014, pp. 273–283, 2011.".

[23] F. Pereira, "Photographic Imaging," in *Audiovisual Communications*, Instituto Superior Técnico, 2015/2016.

[24] "The JPEG Standard," [Online]. Available: http://www.ual.es/~vruiz/Docencia/Apuntes/Coding/Image/03-JPEG/index.html. [Accessed 5 May 2017].

[25] "Huffman coding," [Online]. Available: https://rosettacode.org/wiki/Huffman_coding. [Accessed 6 May 2017].

[26] T. Richter, "JPEG on STEROIDS: Common optimization techniques for JPEG image compression," *Proceedings - International Conference on Image Processing, ICIP,* vol. August, pp. 61-65, 2016.

[27] "ICIP 2016," [Online]. Available: http://2016.ieeeicip.org/. [Accessed 3 May 2017].

[28] "Overview and Benchmarking Summary for the ICIP 2016 Compression Challenge," [Online]. Available: https://jpeg.org/downloads/aic/wg1n73041_icip_2016_grand_challenge.pdf. [Accessed 5 May 2017].

[29] "Information Technology - Digital Compression and Coding of Continous-tone still images . Requirements and Guidelines," [Online]. Available: https://www.w3.org/Graphics/JPEG/itu-t81.pdf. [Accessed May 2017].

[30] "Announcing Guetzli: A New Open Source JPEG Encoder," [Online]. Available: https://research.googleblog.com/2017/03/announcing-guetzli-new-open-source-jpeg.html. [Accessed 8 May 2017].

[31] "Butteraugli Quality Metric," [Online]. Available: https://github.com/google/butteraugli. [Accessed 8 May 2017].

[32] J. Alakuijala, R. Obryk , O. Stoliarchuk, Z. Szabadka, L. Vandevenne and J. Wassenberg, "Guetzli: Perceptually Guided JPEG Encoder," 2017. [Online]. Available: https://arxiv.org/pdf/1703.04421.pdf. [Accessed 7 May 2017].

[33] F. Pereira, "JPEG 2000," in *Comunicações Audiovisuais: Tecnologias, Normas e Aplicações*, 2009, pp. 482, 483, 484.

[34] "Applications for JPEG 2000," [Online]. Available: https://jpeg.org/jpeg2000/applications.html. [Accessed 9 May 2017].

[35] "How does JPEG 2000 work?," [Online]. Available: http://faculty.gvsu.edu/aboufade/web/wavelets/student_work/EF/how-works.html. [Accessed 27 April 2017].

[36] K. Nahrstedt, "JPEG 2000," 2012. [Online]. Available: https://courses.engr.illinois.edu/cs414/sp2012/Lectures/lect9-jpeg2000.pdf. [Accessed 4 May 2017].

[37] "Mozilla Foundation," [Online]. Available: https://www.mozilla.org/en-US/foundation/. [Accessed 10 May 2017].

[38] "Xiph.Org Foundation," [Online]. Available: https://xiph.org/. [Accessed 10 May 2017].

[39] "Daala Video Compression," [Online]. Available: https://xiph.org/daala/. [Accessed 10 May 2017].

[40] "Introducing Daala," [Online]. Available: https://people.xiph.org/~xiphmont/demo/daala/demo1.shtml. [Accessed 10 May 2017].

[41] "Wikipedia on HEVC," [Online]. Available: https://en.wikipedia.org/wiki/High_Efficiency_Video_Coding. [Accessed 10 May 2017].

[42] "Wikipedia on VP9," [Online]. Available: https://en.wikipedia.org/wiki/VP9. [Accessed May 2017].

[43] "VideoLAN Dev Days 2016: Daala," [Online]. Available: https://www.youtube.com/watch?v=AOssZFJ0EdI&t=200s. [Accessed 10 May 2017].

[44] "Alliance for Open Media wiki," [Online]. Available: https://en.wikipedia.org/wiki/Alliance_for_Open_Media. [Accessed 10 May 2017].

[45] "Wikipedia on Daala," [Online]. Available: https://en.wikipedia.org/wiki/Daala. [Accessed 29 April 2018].

[46] J. Valin et al, "Daala: A Perceptually-Driven Still Picture Codec," 2016. [Online]. Available: http://arxiv.org/abs/1605.04930. [Accessed 9 May 2017].

[47] "Developing the Opus and Daala codecs," [Online]. Available: https://lwn.net/Articles/571560/. [Accessed 11 May 2017].

[48] N. E. Egge, J. Valin, T. B. Terriberry, T. Daede, and C. Montgomery, "Using Daala Intra Frames for Still Picture Coding," in *Picture Coding Symposium*, Cairns, Australia, 2015.

[49] "Wikipedia Haar Wavelet," [Online]. Available: https://en.wikipedia.org/wiki/Haar_wavelet#Haar_transform. [Accessed 12 May 2018].

[50] J. Valin et al, "Daala: Building a Next-Generation Video Codec from Unconventional Technology," [Online]. Available: https://jmvalin.ca/slides/mmsp2016_poster.pdf. [Accessed 12 May 2018].

[51] "Daala: Perceptual Vector Quantization (PVQ)," [Online]. Available: https://people.xiph.org/~jm/daala/pvq_demo/. [Accessed 13 May 2017].

[52] "Revisiting Daala Technology Demos," [Online]. Available: https://people.xiph.org/~jm/daala/revisiting/. [Accessed 11 May 2017].

[53] "Wikipedia on Ringing Artifacts," [Online]. Available: https://en.wikipedia.org/wiki/Ringing_artifacts. [Accessed 13 May 2017].

[54] "A Deringing Filter for Daala... and beyond," [Online]. Available: https://people.xiph.org/~jm/daala/deringing_demo/. [Accessed 11 May 2017].

[55] "About WebM," [Online]. Available: https://www.webmproject.org/about/. [Accessed 14 May 2017].

[56] "A Technical Overview of VP9," [Online]. Available: http://files.meetup.com/9842252/Overview-VP9.pdf. [Accessed 13 May 2017].

[57] "The Quadtree and Related Hierarchical Data Structures," [Online]. Available: http://www.cs.umd.edu/~hjs/pubs/SameCSUR84-ocr.pdf. [Accessed 15 May 2017].

[58] M. Sharabayko, O. Ponomarev and R. Chernyak, "Intra Compression Efficiency in VP9 and HEVC," *Applied Mathematical Sciences,* vol. 7, no. 137, pp. 6803-6824, 2013.

[59] "Cisco Systems," [Online]. Available: http://www.cisco.com/c/pt_pt/index.html. [Accessed 11 May 2017].

[60] "AV1 Bitstream Analyzer," [Online]. Available: https://medium.com/@mbebenita/av1-bitstream-analyzer-d25f1c27072b. [Accessed 15 May 2017].

[61] "Thor High Efficiency, Moderate Complexity Video Codec using only RF IPR," July 2015. [Online]. Available: https://www.ietf.org/proceedings/93/slides/slides-93-netvc-4.pdf. [Accessed 16 May 2017].

[62] "Thor Video Codec," [Online]. Available: https://tools.ietf.org/html/draft-fuldseth-netvc-thor-03. [Accessed 15 May 2017].

[63] L. Fiorentin et al, "Comparison Between HEVC and Thor Based on Objective and Subjective Assessments," in *International Conference on Systems, Signals and Image Processing (IWSSIP)*, Bratislava, Slovakia, 2016.

[64] "JPEG on Steroids Libjeg Implementation," [Online]. Available: https://github.com/thorfdbg/libjpeg. [Accessed 2 May 2017].

[65] "ISO/IEC 10918-1:1994," International Organization for Standardization, [Online]. Available: https://www.iso.org/standard/18902.html. [Accessed 4 May 2018].

[66] "Guetzli Implementation," [Online]. Available: https://github.com/google/guetzli. [Accessed 27 April 2017].

[67] "OpenJPEG," [Online]. Available: http://www.openjpeg.org/. [Accessed 14 May 2017].

[68] "Daala Implementation," [Online]. Available: https://github.com/xiph/daala. [Accessed 13 May 2017].

[69] "Libvpx Implementation," [Online]. Available: https://github.com/webmproject/libvpx. [Accessed 13 May 2017].

[70] "Thor Video Codec Implementation," [Online]. Available: https://github.com/cisco/thor. [Accessed 11 May 2018].

[71] "Wikipedia on Peak Signal-to-noise Ratio," [Online]. Available: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio. [Accessed 21 May 2017].

[72] "Wikipedia on Structural Similarity," [Online]. Available: https://en.wikipedia.org/wiki/Structural_similarity. [Accessed 21 May 2017].

[73] "Tackling illegal, unreported and unregulated (IUU) fishing," [Online]. Available: https://ec.europa.eu/fisheries/sites/fisheries/files/docs/publications/2015-04-tackling-iuu-fishing_en.pdf. [Accessed 13 April 2017].

[74] "Fishing Monitoring Systems: Past, Present and Future," [Online]. Available: http://www.imcsnet.org/imcs/docs/fishing_vessel_monitoring_systems_past_present_future.pdf. [Accessed 12 March 2017].

[75] "Monicap - Xsealence," [Online]. Available: http://www.xsealence.com/portfolio/monicap/. [Accessed 12 March 2017].

[76] "Vessel Monitoring Systems and their Role in Fisheries Management and Monitoring, Control and Surveillance," [Online]. Available: http://www.iotc.org/sites/default/files/documents/2016/05/IOTC-2016-CoC13-Inf08_-_PEW_VMS_Brief-Whitepaper_2016_0.pdf. [Accessed 13 March 2017].

[77] "Consortium wins UK fisheries VMS tender," [Online]. Available: http://www.worldfishing.net/news101/industry-news/consortium-wins-uk-fisheries-vms-tender. [Accessed 14 April 2017].

[78] "Bluetraker Touch-screen Terminal," [Online]. Available: http://www.bluetraker.com/products/bluetraker-hmi-7-touch-screen-terminal. [Accessed 14 April 2017].

[79] "Tracking Vessels Supports EU Sustainable Fishing Policy," [Online]. Available: http://www.esri.com/esri-news/arcnews/winter1213articles/tracking-vessels-supports-eu-sustainable-fishing-policy. [Accessed 14 March 2017].

[80] "Iridium Global Network," [Online]. Available: https://www.iridium.com/network/globalnetwork. [Accessed 21 April 2017].

[81] "Inmarsat," [Online]. Available: http://www.inmarsat.com/. [Accessed 21 April 2017].

[82] "Argos," [Online]. Available: http://www.argos-system.org/. [Accessed 18 April 2017].

[83] "Wikiedia on GLONASS," [Online]. Available: https://en.wikipedia.org/wiki/GLONASS. [Accessed 19 April 2017].

[84] "Wikipedia on Global Positioning System," [Online]. Available: https://en.wikipedia.org/wiki/Global_Positioning_System. [Accessed 20 April 2017].

[85] "European Global Navigation Satellite System Agency," [Online]. Available: https://www.gsa.europa.eu/european-gnss/galileo/galileo-european-global-satellite-based-navigation-system. [Accessed 20 April 2017].

[86] "Rise and F," The Rise and Fall and Rise of Iridium, [Online]. Available: http://www.airspacemag.com/space/the-rise-and-fall-and-rise-of-iridium-5615034/. [Accessed 27 April 2017].

[87] "Iridium Technical Details," [Online]. Available: http://www.decodesystems.com/iridium.html. [Accessed 23 March 2017].

[88] "General Description of Model 9602-I," [Online]. Available: http://www.nalresearch.com/Info/General%20Description%20of%20Model%209602-I.pdf. [Accessed 21 March 2017].

[89] "Iridium Bandwidth and Internet Download Speeds," [Online]. Available: http://www.mailasail.com/Support/Iridium-Bandwidth. [Accessed 9 April 2017].

[90] "Measuring Latency in Iridium Satellite Constellation Data Services," [Online]. Available: http://dodccrp.org/events/10th_ICCRTS/CD/papers/233.pdf. [Accessed 9 March 2017].

[91] "Iridium Pilot - Iridium Broadband," [Online]. Available: http://www.mailasail.com/Communication/Iridium-Pilot. [Accessed 10 April 2017].

[92] "Iridium 9602 SBD Transceiver Developer's Guide," [Online]. Available: https://satphoneshop.com/downloads/9602DevelopersGuide.pdf. [Accessed 29 March 2017].

[93] "Iridium SBD," [Online]. Available: https://www.iridium.com/services/details/iridium-sbd. [Accessed 24 March 2017].

[94] "Iridium SBD service guide," [Online]. Available: http://www.satcom.pro/upload/iblock/757/IRDM_IridiumSBDService_V3_DEVGUIDE_9Mar2012.pdf. [Accessed 21 March 2017].

[95] "Short Burst Data Service," [Online]. Available: https://www.beamcommunications.com/services/short-burst-data-sbd. [Accessed 22 March 2017].

[96] "Iridium GO!," [Online]. Available: https://www.iridium.com/products/details/iridiumgo. [Accessed 2 April 2017].

[97] "Iridium OpenPort," [Online]. Available: https://www.iridium.com/services/details/iridium-openport-broadband-service. [Accessed 3 April 2017].

[98] "Iridium Pilot," [Online]. Available: https://www.iridium.com/products/details/iridiumpilot. [Accessed 2 April 2017].

[99] "Iridium OpenPort Service," [Online]. Available: http://www.groundcontrol.com/Iridium_Openport.htm. [Accessed 15 April 2017].

[100] "IRIDIUM SBD RATES," [Online]. Available: http://www.satphonestore.com/airtime/iridium-sbd.html. [Accessed 23 March 2017].

[101] "Iridium GO! Rate Plans," [Online]. Available: http://www.bluecosmo.com/iridium-go/rate-plans. [Accessed 2 April 2017].

[102] "Iridium Announces Successful First Launch of Iridium NEXT Satellites," [Online]. Available: http://investor.iridium.com/releasedetail.cfm?releaseid=1007978. [Accessed 25 March 2017].

[103] "Iridium NEXT: The Bold of Satellite Communications," [Online]. Available: http://systems.fastwave.com.au/media/1767/irdm_iridiumnext_brochure_hr_16mar2015.pdf. [Accessed 10 April 2017].

[104] "Evolution to Iridium NEXT," [Online]. Available: https://static1.squarespace.com/static/57a8878837c58153c1897c2c/t/582c7e4837c5811822b4a439/1479310932768/1.KyleHurst.pdf. [Accessed 14 April 2017].

[105] "Iridium NEXT - a presentation," [Online]. Available: http://www.argo.ucsd.edu/sat_comm_AST13.pdf. [Accessed 24 March 2017].

[106] "Iridium NEXT," [Online]. Available: https://www.iridium.com/network/iridiumnext. [Accessed 24 March 2017].

[107] "European Space Agency," [Online]. Available: http://www.esa.int/ESA. [Accessed 19 April 2017].

[108] "IPMA," [Online]. Available: http://www.ipma.pt/en/oipma/. [Accessed 21 November 2017].

[109] "Trawling," [Online]. Available: https://en.wikipedia.org/wiki/Trawling. [Accessed 22 November 2017].

[110] "HikVision VF IR-Bullet Network Camera," [Online]. Available: http://www.hikvision.com/en/Products_accessries_760_i6150.html. [Accessed 4 December 2017].

[111] "OpenCV - Geometric Image Transformations," [Online]. Available: https://docs.opencv.org/3.1.0/da/d54/group__imgproc__transform.html#gga5bb5a1fea74ea38e1a5445ca803ff121acf959dca2480cc694ca016b81b442ceb. [Accessed 14 January 2018].

[112] O. R. e. al., "ImageNet Large Scale Visual Recognition Challenge," *Int. J. Comput. Vis.,* vol. 115, p. 211–252, 2015.

[113] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Multimedia Tools and Applications,* vol. 77, no. 9, pp. 10437-10453, 2015.

[114] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li, "Large-scale video classification with convolutional neural networks," in *Computer Vision and Pattern Recognition (CVPR)*, Columbus, Ohio, 2014.

[115] "CNN Benchmarks," [Online]. Available: https://github.com/jcjohnson/cnn-benchmarks. [Accessed 29 November 2017].

[116] "Torch," [Online]. Available: http://torch.ch/. [Accessed 1 December 2017].

[117] A. Zisserman, K. Simonyan, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Information and Software Technology,* vol. 51, no. 4, pp. 769-784, 2015.

[118] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Neural Information Processing Systems (NIPS)*, Lake Tahoe, Nevada, 2012.

[119] C. Szegedy et al., "Going deeper with convolutions," 2014. [Online]. Available: arXiv:1409.4842.

[120] "Wikipedia on Rectifier," [Online]. Available: https://en.wikipedia.org/wiki/Rectifier_(neural_networks). [Accessed 12 December 2017].

[121] "Convolutional Neural Networks," [Online]. Available: http://cs231n.github.io/convolutional-networks/#norm. [Accessed 24 November 2017].

[122] "Wikipedia on Softmax Function," [Online]. Available: https://en.wikipedia.org/wiki/Softmax_function. [Accessed 3 December 2017].

[123] "Transfer Learning," [Online]. Available: http://cs231n.github.io/transfer-learning/. [Accessed 6 December 2017].

[124] "Caffe Model Zoo," [Online]. Available: http://caffe.berkeleyvision.org/model_zoo.html. [Accessed 9 December 2017].

[125] "Caffe to Tensorflow," [Online]. Available: https://github.com/ethereon/caffe-tensorflow. [Accessed 12 December 2017].

[126] "Wikipedia on Cross Entropy," [Online]. Available: https://en.wikipedia.org/wiki/Cross_entropy#Cross-entropy_error_function_and_logistic_regression. [Accessed 21 January 2018].

[127] "Wikipedia on Overfitting," [Online]. Available: https://en.wikipedia.org/wiki/Overfitting. [Accessed 12 December 2017].

[128] P. Hensman, D. Masko, "The Impact of Imbalanced Training Data for Convolutional Neural Networks," in *Degree Project in Computer Science*, KTH Royal Institute of Technology, 2015.

[129] "Wikipedia on Receiver Operating Characteristic," [Online]. Available: https://en.wikipedia.org/wiki/Receiver_operating_characteristic. [Accessed 7 December 2017].

[130] R. Roberts, R. Maxion, "Proper Use of ROC Curves in Intrusion_Anomaly Detection," in *Technical report*, School of Computing Science, University of Newcastle, 2004.

[131] "Overview of JPEG," [Online]. Available: https://jpeg.org/jpeg/. [Accessed June 2017].

[132] "Independent JPEG Group - libjpeg," [Online]. Available: http://www.ijg.org/. [Accessed May 2017].

[133] "JPEG on Steroids libjeg release," [Online]. Available: https://github.com/thorfdbg/libjpeg. [Accessed 3 May 2017].

[134] F. Pereira, "JPEG 2000," in *Comunicações Audiovisuais: Tecnologias, Normas e Aplicações*, IST PRESS, 2009, pp. 482, 483, 484.

[135] "OpenJPEG," [Online]. Available: http://www.openjpeg.org/. [Accessed 4 May 2017].

[136] "FFmpeg," [Online]. Available: https://ffmpeg.org/. [Accessed 11 May 2018].

[137] "FFmpeg - Post Processing Filters," [Online]. Available: https://trac.ffmpeg.org/wiki/Postprocessing. [Accessed 19 February 2018].

[138] "Short Burst Data Service," [Online]. Available: https://www.beamcommunications.com/services/short-burst-data-sbd. [Accessed 27 March 2017].

[139] "Iridium Technical Details," [Online]. Available: http://www.decodesystems.com/iridium.html. [Accessed March 2017].

[140] "IRIDIUM SBD RATES," [Online]. Available: http://www.satphonestore.com/airtime/iridium-sbd.html. [Accessed March 2017].

[141] J. Dong, X. Mao, C. Shen, and Y. Yang, "Learning Deep Representations Using Convolutional Auto-encoders with Symmetric Skip Connections," 2016. [Online]. Available: https://arxiv.org/abs/1611.09119.

[142] A. Lucas, M. Iliadis, R. Molina, and A. Katsaggelos, "Using Deep Neural Networks for Inverse Problems in Imaging," *IEEE Signal Processing Magazine,* vol. January, p. 20–36, 2018.

[143] A. Zisserman, K. Simonyan, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *Information and Software Technology,* vol. 51, no. 4, pp. 769-784, 2014.

[144] "Convolutional Neural Network - ResNet," [Online]. Available: http://sqlml.azurewebsites.net/2017/09/12/convolutional-neural-network/. [Accessed 21 April 2018].

[145] C. Yang, C. Ma, and M. Yang, "Single-Image Super-Resolution: A Benchmark," in *European Conference on Computer Vision (ECCV)*, Springer, 2014.

[146] "Wikipedia on Bicubic Interpolation," [Online]. Available: https://en.wikipedia.org/wiki/Bicubic_interpolation. [Accessed 2 May 2018].

[147] C. Dong, C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 38, no. 2, pp. 295-307, 2016.

[148] J. Kim, J. K. Lee, and K. M. Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," in *Conference on Computer Vision and Pattern Recognition*, Nevada, EUA, 2016.

[149] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing,* Vols. 26, no. 7, p. 3142–3155, 2017.

[150] C. Ledig et al., "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, Hawai, EUA, 2017.

[151] B. Lim, S. Son, H. Kim, S. Nah, and K. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work,* vol. 2017–July, p. 1132–1140, 2017.

[152] C. Dong, Y. Deng, C. Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," *Proceedings of the IEEE International Conference on Computer Vision,* p. 576–584, 2015.

[153] L. Cavigelli, P. Hager, and L. Benini, "CAS-CNN: A Deep Convolutional Neural Network for Image Compression Artifact Suppression," in *IEEE Conference on Neural Networks*, Anchorage, USA, 2017.

[154] C. Szegedy and S. Ioffe, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International Conference on Machine Learning*, Lille, France, 2015.

[155] "DnCNN Implementation," [Online]. Available: https://github.com/cszn/DnCNN. [Accessed 3 May 2018].

[156] "LIVE1, Classic5 and Set14 Datasets," [Online]. Available: https://github.com/cszn/DnCNN/tree/master/testsets. [Accessed 8 April 2018].

[157] "Wikipedia on Lanczos Resampling," [Online]. Available: https://en.wikipedia.org/wiki/Lanczos_resampling. [Accessed 21 April 2017].

[158] "Wikipedia on Raspberry Pi," [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi. [Accessed 3 March 2018].

[159] "Wikipedia on Camera Serial Interface," [Online]. Available: https://en.wikipedia.org/wiki/Camera_Serial_Interface. [Accessed 3 March 2018].

[160] "Wikipedia on Raspbian," [Online]. Available: https://en.wikipedia.org/wiki/Raspbian. [Accessed 3 March 2018].

[161] "Tensorflow Repository," [Online]. Available: https://github.com/tensorflow/tensorflow. [Accessed 28 April 2018].

[162] "Installing TensorFlow from Sources," [Online]. Available: https://www.tensorflow.org/install/install_sources. [Accessed 3 March 2018].

[163] "TensorFlow in Other Languages," [Online]. Available: https://www.tensorflow.org/extend/language_bindings. [Accessed 5 March 2017].

[164] "Wikipedia on Lanczos Resampling," [Online]. Available: https://en.wikipedia.org/wiki/Lanczos_resampling. [Accessed 21 April 2017].