

End to End Multi-Objective Optimisation of H.264 and HEVC Codecs

By

Maryam Mohsin Salim Al Barwani

A Doctoral Thesis

Submitted in partial fulfilment of the
requirements for the award of

Doctor of
Philosophy of
Loughborough University

October 2017

© by Maryam Mohsin Al Barwani (2017)

Supervisor: Prof. Eran Edirisinghe

Abstract

All multimedia devices now incorporate video CODECs that comply with international video coding standards such as H.264 / MPEG4-AVC and the new High Efficiency Video Coding Standard (HEVC) otherwise known as H.265. Although the standard CODECs have been designed to include algorithms with optimal efficiency, large number of coding parameters can be used to fine tune their operation, within known constraints of for e.g., available computational power, bandwidth, consumer QoS requirements, etc. With large number of such parameters involved, determining which parameters will play a significant role in providing optimal quality of service within given constraints is a further challenge that needs to be met. Further how to select the values of the significant parameters so that the CODEC performs optimally under the given constraints is a further important question to be answered.

This thesis proposes a framework that uses machine learning algorithms to model the performance of a video CODEC based on the significant coding parameters. Means of modelling both the Encoder and Decoder performance is proposed. We define objective functions that can be used to model the performance related properties of a CODEC, i.e., video quality, bit-rate and CPU time. We show that these objective functions can be practically utilised in video Encoder/Decoder designs, in particular in their performance optimisation within given operational and practical constraints. A Multi-objective Optimisation framework based on Genetic Algorithms is thus proposed to optimise the performance of a video codec. The framework is designed to jointly minimize the CPU Time, Bit-rate and to maximize the quality of the compressed video stream. The thesis presents the use of this framework in the performance modelling and multi-objective optimisation of the most widely used video coding standard in practice at present, H.264 and the latest video coding standard, H.265/HEVC.

When a communication network is used to transmit video, performance related parameters of the communication channel will impact the end-to-end performance of the video CODEC. Network delays and packet loss will impact the quality of the video that is received at the decoder via the communication channel, i.e., even if a

video CODEC is optimally configured network conditions will make the experience sub-optimal. Given the above the thesis proposes a design, integration and testing of a novel approach to simulating a wired network and the use of UDP protocol for the transmission of video data. This network is subsequently used to simulate the impact of packet loss and network delays on optimally coded video based on the framework previously proposed for the modelling and optimisation of video CODECs. The quality of received video under different levels of packet loss and network delay is simulated, concluding the impact on transmitted video based on their content and features.

Maryam M. Al Barwani, October 2017

Acknowledgments

First of all, I would like to thank Almighty God for all blessing given to me providing the ability and patience to finish this thesis work.

I would like to thank my supervisor, Prof Eran A. Edirisinghe, for his expert guidance and continuous encouragement throughout the course of my PhD. Without his suggestions this work would not have been possible.

I want to extend my gratitude to Mr Salim Al Busaidi my dearest husband, who has always supported me and helped me overcoming the difficulties without complaining. I would also like to say a big thank you to my beloved parents, brothers and sisters for their prayers and care before and during my study at Loughborough University. My special thanks also go to my children.

I also appreciate the support and friendship provided by the academic and secretarial staff of the department of Computer Science, Loughborough University. To all I say thank you.

Finally, I would like to acknowledge the support of the Ministry of Manpower of the Sultanate of Oman for providing the funding for this research and Loughborough University UK for providing the required research facilities and support.

Table of Contents

Abstract	ii
Acknowledgments	iv
Table of Contents	v
List of Figures	viii
List of Tables	x
List of Abbreviations	xii
Chapter 1 Introduction	1
1.1 Aim & Objectives	3
1.2 Research Contributions	3
1.3 Thesis Structure	4
Chapter 2 Research Background	5
2.1 Introduction	5
2.2 Video Representation Formats and Compression	5
2.2.1 Digital video formats	6
2.3 H.246 ADVANCE video coding standard	7
2.3.1 Video Coding Concepts.....	8
2.3.2 Basic coding structure for H.264/AVC	9
2.3.3 The H.264/AVC Coding Tools.....	10
2.3.4 The H.264/AVC encoder parameters.....	11
2.4 HEVC video coding standard	13
2.4.1 Differences of HEVC from H.264/AVC.....	14
2.4.2 Compression Performance Analysis in HEVC	17
2.5 Tools for Encoding, Decoding, Modelling and Optimisation	19
2.5.1 JM Reference Software.....	19
2.5.2 The Intel® VTune™ Amplifier XE.....	19
2.5.3 WEKA machine learning toolkit.....	20
2.5.4 Matlab Optimisation Toolbox	23
2.6 An overview of End-to End video streaming	25

2.6.1 Overview of video streaming	25
2.6.2 Overview of the VideoLAN streaming solution.....	27
2.6.3 EvalVid - A Video Quality Evaluation Tool-set	27
2.6.4 Network simulator using Riverbed Modeler (Opnet).....	28
2.7 Summary	28
<i>Chapter 3 Literature Review</i>	<i>29</i>
3.1 Introduction	29
3.2 Optimisation of Video Coding	29
3.2.1 Parameter-based Optimisation	31
3.3 Theory of Multi-Objective Optimisation	36
3.3.1 Multi-Objective Optimisation using Genetic Algorithms	37
3.4 H.264 Video Coding	38
3.5 High Efficiency Video Coding (HEVC)	40
3.6 Use of machine learning in video coding	44
3.7 End-to-End Video Streaming	46
3.8 Summary & Conclusions	49
<i>Chapter 4 Parameter based Characterisation and Performance</i>	
<i>Modelling of a H.264 Video CODEC</i>	<i>50</i>
4.1 Decoder Introduction	50
4.2 Proposed framework for multi-objective optimization.....	51
4.2.1 The Profiling Experiments - Determining the Significant Coding Parameters	53
4.2.2 The Objective Functions of the H.264/AVC Encoder	59
4.3 Encoder Performance Analysis	61
4.4 Decoder Performance Analysis	64
4.4.1 The output of Decoded video.....	65
4.5 Using Advanced Machine Learning Algorithms for the Modelling of an H264 CODEC	69
4.5.1 Experiments, results and analysis	70
4.6 Summary & Conclusion	72

Chapter 5 Multiobjective Optimisation	73
5.1 Introduction	73
5.2 Setting up the Genetic Algorithm	74
5.3 Optimising the Encoder	77
5.3.1 Experimental results and analysis	78
5.4 Optimising the Decoder	86
5.5 Summary and Conclusion	88
Chapter 6 A Machine Learning based Framework for Parameter based Multi-Objective Optimisation of a H.265 Video CODEC	89
6.1 Introduction	89
6.2 Proposed Framework for Performance Modelling	90
6.2.1 Profiling Experiments/ Determining the Significant Coding Parameters 92	
6.2.2 The Objective Functions of the HEVC Encoder	97
6.3 Analysis of experimental results	97
6.3.1 Encoder Analysis	97
6.3.2 Decoder Analysis	100
6.4 Multi-Objective Optimisation of a H.265 Video CODEC	102
6.4.1 Implementation	103
6.5 Optimising the encoder	105
6.5.1 Experimental results	106
6.5.2 Discussion	109
6.6 Summary	111
Chapter 7 Impact of Packet Loss & Network Delay on Optimally Coded Video Streaming	112
7.1 Introduction	112
7.2 System Design and Implementation	114
7.2.1 Tested video sequences	116
7.2.2 Video streaming experimental setup	117
7.2.3 Client - Server communication	117

7.3 Experiments, results and analysis.....	118
7.4 Summary.....	124
<i>Chapter 8 Conclusions and Future Work</i>	<i>126</i>
8.1 Conclusions	126
8.2 Future work.....	129
<i>References</i>	<i>131</i>
<i>Appendices</i>	<i>140</i>
Appendix A: JM reference encoder / decoder configuration file.....	140
Appendix B: Selected parameters for encoder / decoder.	142
Appendix C: Pareto Plot.....	148
Appendix D: Optimal points and functional values.	149
Appendix E: HEVC HM encoder / decoder configuration file.....	156
Appendix F: Selected parameters for encoder / decoder.....	159
Appendix G: Optimal points and functional values.	161

List of Figures

Figure 2-1: A Video CODEC Process.	9
Figure 2-2: The H.264 video coding and decoding processes [3].	10
Figure 2-3: An example of sequence with I-, B- and P-frames [8].	10
Figure 2-4 Block diagram of an HEVC encoder with built-in decoder [13]	14
Figure 2-5 Video streaming diagram [29].....	26
Figure 2-6 VideoLAN streaming solution [30]	27
Figure 3-1: Multi-objective Optimisation framework used in [37].....	32
Figure 4-1: Proposed Multi-objective optimisation framework.	52
Figure 4-2: Sample image of frame 30 at (a) QP= 17 and (b) at QP= 49	65
Figure 5-1: Setting options for the optimisation task	76
Figure 5-2: Pareto front for foreman PSNR in (db) vs. Bit-Rate in (kbit/s).	80
Figure 5-3: Pareto front for foreman PSNR in (db) vs. CPU Time in (sec).	80
Figure 5-4: Pareto front for foreman CPU Time in (sec) vs. Bit-Rate in (Kbit/s).....	81
Figure 5-5: number of solutions and the number of generations.....	83
Figure 5-6: More details about the optimization	83
Figure 5-7: Pareto points 1 PSNR in (db) vs. Bit-rate in (Kbit/s).....	85
Figure 5-8: Pareto points 2 PSNR vs. CPU Time in (sec).	85
Figure 5-9: Pareto points 3 CPU Time in (sec) vs. Bit-Rate	86
Figure 5-10: Bit-Rate in in (Kbit/s) vs. CPU Time in (sec).....	87
Figure 5-11: PSNR in (db) vs. CPU Time in (sec).....	87
Figure 6-1: Proposed Multi-objective optimisation framework	91
Figure 6-2: PSNR versus Bit-rate at QP 27, 37, 45.	100
Figure 6-3: The visual artifact with different QP.	102
Figure 6-4: Pareto front for cactus video sequences PSNR in (db) vs. Bit-Rate in	

(Kbit/s).....	106
Figure 6-5: Pareto front for cactus video sequences CPU Time in (sec) vs. Bit-Rate in (Kbit/s).....	107
Figure 6-6: Pareto selected points for Cactus PSNR vs Bit-Rate.....	110
Figure 6-7: Pareto selected points for Cactus CPU Time vs Bit-Rate.....	110
Figure 7-1: Block diagram of streaming procedure.....	116
Figure 7-2: The effect of different packet loss rates on the PSNR.	121
Figure 7-3: Impact on PSNR with different levels of delay for Akiyo video.	122
Figure 7-4: (a) video frame before streaming (b) received video by client at 6% Packet loss Rate. (c) The effect of delay at 50 ms.	123

List of Tables

Table 2-1 SIF, CIF, and QCIF digital video formats [1].	7
Table 2-2: List of coding parameters	11
Table 2-3 H.265 (HEVC) and H.264 (MPEG 4 AVC) [14].	15
Table 2-4 Intra prediction techniques between H.264 and HEVC [13]	16
Table 4-1: Selected frames of video sequences [93], [94].	55
Table 4-2: The initial list of parameters used [37].	56
Table 4-3: Significant parameters and value used.	57
Table 4-4: selected set of parameters for foreman sequence	58
Table 4-5: Encoder Correlation Coefficient.	62
Table 4-6: Selected set of decoder parameters for Foreman Sequences.	67
Table 4-7: Decoder correlation coefficient	68
Table 4-8: Decoder Correlation coefficient using bagging	70
Table 4-9: Encoder Correlation coefficient with both linear and bagging	71
Table 5-1: ‘gamultobj’ settings used	75
Table 5-2: The optimal points for foreman PSNR vs. Bit-rate	82
Table 5-3: Output data describing the results of MOO with GA for Figure 5-2 to Figure 5-4.	82
Table 6-1: Settings for the Encoder in HM	93
Table 6-2: Tested Video Sequences	94
Table 6-3: Selected Set Of Parameters For Cactus Sequence	96
Table 6-4: Encoder Correlation Coefficient	98
Table 6-5: Decoder Correlation Coefficient	100
Table 6-6: Optimisation settings	104
Table 6-7: The optimal points for Cactus PSNR vs. Bit-rate	107

Table 6-8: Output data describing the results of MOO with GA for cactus.	109
Table 7-1: Video sequences and their properties	117
Table 7-2: Select optimal points from Figure 5-7 in chapter 5 for foreman video. .	119
Table 7-3: Encoded video with different parameter sets for foreman video using optimal points.....	119
Table 7-4: Packet Loss settings and the resulted PSNR (quality)	120
Table 7-5: Impact of packet-delay on the quality of the Akiyo video sequence.....	121
Table 7-6: Using defaults parameter of H.264 encoder.....	124

List of Abbreviations

AVC	Advance video coding
AVC	Advance video coding
B-frames	Bi-directional predicted frame
CABAC	Context-Adaptive Binary Arithmetic Coding
CAVLC	Context-Adaptive Variable Length Coding
CIF	Common Intermediate Format
CODEC	COder-DECoder/COMpression-DECompression
CPU	Central Processing Unit
dB	Decibels
DCT	Discrete Cosine Transform
EAs	Evolutionary algorithms
EMO	Evolutionary Multi-Objective Optimisation
FMO	Flexible Macro-block Ordering
GA	Genetic Algorithm
GHz	Gigahertz - 1 billion hertz
HEVC	High-Efficiency Video Coding
IEC	International Electro technical Commission
I-frames	Intra-frames
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union –Telecommunication
JTC	Joint Technical Committee
JVT	Joint Video Team
MOEAs	Multi-objective evolutionary algorithms

MOO	Multi-objective Optimization
MPEG	Moving Picture Experts Group
MSE	Mean squared error
NAL	Network adaptation layer
NSGA	Non-dominated sorting genetic algorithm
OM	OPNET Modeler
P-frames	Predicted frames
P-R-D	Power-rate distortion
PSNR	Peak Signal-to-Noise Ratio
QCIF	Quarter Common Intermediate Format
QoS	Quality of service
QP	Quantization Parameter
UHD	Ultra-high definition
VCEG Video	Coding Experts Group
VCL	Video coding layer
WLAN	Wireless local area network
WEKA	Waikato Environment for Knowledge Analysis

Chapter 1

Introduction

Applications that benefit from accurate video capture, efficient representation and coding, error-free transmission and subjectively optimised display, have been growing over the years due to the availability of higher network bandwidth, faster processor speed and advanced capture and display technologies. Recent studies have shown that coded video data is becoming the major part in consumer internet traffic with a predicted share of 90% by 2019. Some of the most extensively used applications include real-time video conferencing, video streaming over broadband networks and digital TV broadcasting. Most current mobile hand-held devices come equipped with a video camera that is able to capture and encode a video stream in a standard format. These devices also include video players, which can decode and playback video. All above developments continuously demand for more efficient video coding algorithms that are able to reduce the bitrate without sacrificing video quality or to enable the increase of video resolution, without increasing the bitrate. High Efficiency Video Coding (HEVC) also known as H.265 is the most recent answer to this consumer demand. The more established video coding standards used widely in practice however is, H.264, the Advanced Video Coding standard and MPEG-2.

All advanced video CODECs have many parameters that can be used to control their operational characteristics, both at the encoder and decoder ends, enabling the possibility of fine tuning their operation for maximum efficiency within environments and application scenarios that are bound by various constraints. For example the available bandwidth will have an upper limit, the network will be subjected to delays and the decoder/display unit may have limitations in processing and capabilities. In the design and implantation of international video coding standards extreme care has been taken to make the algorithmic performance optimal by designing and using (or recommending the use of) the optimal algorithms for every possible task within a video CODEC. Therefore nothing much can be done to change the algorithms. Yet the encoder, transmission medium and decoder have many parameters that can be adjusted for them to be efficiently operational under above mentioned constraints. Identifying the values of these

parameters that result in the implemented CODECs optimal performance under given constraints remains an open research problem of vital importance.

The first step of parameter based optimisation of a video CODEC is the identification of the coding parameters that have a significant impact of its key properties, such as, bandwidth, image/video quality, and CPU time, etc. Although an experienced user of a video CODEC can guess these parameters with some accuracy, when the content of the video is known, a formal scientific approach is needed to accurately decide the parameter set, with minimum error from any subjectivity in the decision making process by human users. Having obtained these parameters it is then possible to model the key properties of the video CODEC described above based on the significant parameters. These models can then be used to optimise the performance of the video codec when operated under practical constraints thus making the parameter based characterisation and modelling, practically useful.

In this research we propose a framework that is based on the fundamentals of machine learning that can be used to scientifically determine the significant coding parameters of a video CODEC. These parameters are then used to model the operational behaviour of the video CODEC for which machine learning algorithms are further utilised. We also show that this model can be used to establish the foundations of a multi-objective optimisation framework, which can be subsequently used for the parameter based optimization of H.264 and H.265 video coding standards. Although the experiments conducted are limited to H.264 and H.265 standards, the proposed framework can be used in relation to any video coding standard. Once parameters that result in the optimal operation of a video codec under constraints have been found using the framework being proposed, the coded video can be transmitted over a practical network that often subjects the video to delays and packet loss. The impact is thus the sub-optimal end-to-end performance of the CODEC although the CODEC itself was set for optimal performance. In this research we analyse the impact of packet loss and network delays on the quality of streamed videos transmitted across a network and received at the decoder. A detailed investigation of the impact of packet loss and delay for transmitting video using the H.264 standard is conducted. How the video quality is impacted for videos with different content and motion features is studied, leading to useful conclusions and recommendation for future work.

1.1 Aim & Objectives

The aim of the research presented within this thesis is to propose a parameter based Multiple Objective Optimization framework for video CODECs, based on the use of machine learning and genetic algorithms. In particular the use of the proposed framework for optimally coding video under given bit-rate, quality and CPU time constraints using H.264 and H.265 standards is demonstrated are the impact of transmitting such optimally coded video over real channels with packet loss and transmission delays is investigated.

Following are a list of research objectives to be met:

- Carry out a background study of H.264 and HEVC video CODECs and a complete literature review of existing approaches to multi-objective optimisation of video CODECs.
- Carry out profiling of H.264 and HEVC video CODECs using machine learning approaches to identify the coding parameters that have the most significant impact on the video compression rate, distortion/quality and use of computational power / CPU Time. Based on the significant coding parameters, create operational models for the same.
- Based on the operational models, carry out parameter based, multi-objective / multi-constraint optimisation of the H.264 and HEVC video CODECs, proposing use cases of optimal encoder and decoder configurations.
- Within an end-to-end video coding, transmission and delivery system where the CODEC has been configured to act optimally, investigate the impact of packet loss and network delays on the delivered video quality.

1.2 Research Contributions

The research conducted within the context of this thesis has led to the following original contributions:

1. The proposal of a novel machine learning based approach for the profiling of video CODECs that can be used to model the performance of CODECs with respect to coded bit-rate, decoded video quality and the computational cost of encoding/decoding.

2. Performance modelling and parameter based multi-objective optimisation of the CODECs of the most widely used video coding standard H.264 and the latest video coding standard HEVC.
3. Investigating the impact of packet-loss and network delay on the decoded video quality when optimally coded video data is transmitted over a real network. In respect to this, the proposal of a system that can be used to transmit video, under varying packet loss and delay settings.

The above contributions have led to the publication of the following conference/journal papers:

1. M. Al-Barwani and E. A. Edirisinghe, "A machine learning based framework for parameter based multi-objective optimisation of a H.265 video CODEC," 2016 IEEE Future Technologies Conference (FTC). pp. 553–559, 2016.
2. M. Al-Barwani, E. Edirisinghe, "A Machine Learning based Framework for Parameter based Multi-Objective Optimisation of Video CODECs", Journal of Advances in Science, Technology and Engineering Systems, vol. 2, no. 3, pp. 1515-1526 (2017).

1.3 Thesis Structure

For clarity of presentation, the remainder of this thesis is organized as follows:

Chapter 2 provides background knowledge about H.264, HEVC, machine learning tools and systems used for the end-to-end delivery of video streaming. Chapter 3 focuses on the study of literature in Multi-objective optimisation, machine learning, video codecs and video steaming. Chapter 4 presents the proposed novel machine learning based framework for the analysis of significant coding parameters of a H.264 CODEC. Chapter 5 introduces the proposed framework for the multi-objective optimisation of a H.264 video CODEC. Chapter 6 presents a machine learning based framework for the profiling, performance modelling and the parameter based multi-objective optimisation of a H.265 video CODEC. Chapter 7 presents a system that has been designed and integrated to enable the simulation of packet loss and network delay, in the end-to-end coding and delivery of optimally coded H.264 video streams. Finally chapter 8 concludes the research findings of the thesis and proposes possible future work.

Chapter 2

Research Background

2.1 Introduction

This chapter introduces the readers to the research background of this thesis. The chapter initially presents some essential basic concepts on video representation, compression and quality evaluation (see section 2.2). It then presents an in-depth analysis of the operational aspects of algorithms and systems within the two video coding standards to be optimized, i.e. H264 (in section 2.3) and HEVC (in section 2.4). Understanding the underlying algorithms that are controlled by the parameters used within the optimization process, is essential in making informed judgements, within the research conducted. In addition section 2.5 covers the software implementations, models and toolkits used within this thesis, that carries out video coding (JM reference software), performance evaluation (Intel Vtune Amplifier XE), optimization (NSGA-II) and machine learning (WEKA). Section 2.6 presents network modelling and end-to-end video delivery (VideoLAN & OPNET). Section 2.7 provides a summary of information presented in this chapter.

2.2 Video Representation Formats and Compression

Video or visual communications require significant amounts of information transmission. Video compression as considered here involves the bit rate reduction of the digital video signal carrying visual information [1]. Traditional video-based compression, like other information compression techniques, focuses on eliminating redundancy and unimportant elements of the source. The degree to which the encoder reduces the bit rate is called its coding efficiency, or equivalently its inverse is termed the compression ratio, that is,

$$\text{coding efficiency} = (\text{compression ratio})^{-1} = \text{encoded bit rate}/\text{decoded bit rate}$$

Compression can be a lossless or lossy operation. Due to the immense volume of video information, lossy operations are a key element used in video compression algorithms. The loss of information or distortion measure is usually evaluated using the mean square error (MSE), mean absolute error (MAE) criteria, or peak signal-to-reconstruction noise (PSNR) as defined in equations below.

$$MSE = \frac{1}{MN} + \sum_{i=1}^M \sum_{j=1}^N [I(i,j) - \hat{I}(i,j)]^2$$

(Equation 2-1)

$$MAE = \frac{1}{MN} + \sum_{i=1}^M \sum_{j=1}^N |I(i,j) - \hat{I}(i,j)|$$

(Equation 2-2)

$$MSE = 20 \log_{10} \left(\frac{2^n}{MSE^{1/2}} \right)$$

(Equation 2-3)

2.2.1 Digital video formats

Digital video is a representation of moving visual images in the form of encoded digital data. This is in contrast to analog video, which represents moving visual images with analog signals.

The ITU Specialist Group has recommended three formats that are used in the ITU H.261, H.263, H.264, and ISO MPEG video compression standards. They are the Standard Input Format (SIF), Common Interchange Format (CIF), and the low bit rate version of CIF called Quarter CIF (QCIF) (see Table 2-1). Together, these formats describe a comprehensive set of digital video formats that are widely used in current digital video applications [1].

Table 2-1 SIF, CIF, and QCIF digital video formats [1].

Description	SIF NTSC/PAL	CIF	QCIF
Horizontal resolution (Y) pixels	352	360(352)	180(176)
Vertical resolution (Y) pixels	240/288	288	144
Horizontal resolution (C_r, C_b) pixels	176	180(176)	90(88)
Vertical resolution (C_r, C_b) pixels	120/144	144	72
Bits per pixel (bpp)	8	8	8
Interlace fields:frames	1:1	1:1	1:1
Frame rate (fps)	30	30, 15, 10, 7.5	30, 15, 10, 7.5
Aspect ratio hor size/vert size	4:3	4:3	4:3
Bit rate (Y) Mbps @ 30 fps	20.3	24.9	6.2
Bit rate (U, V) Mbps @ 30 fps	10.1	12.4	3.1

2.3 H.264 ADVANCE video coding standard

H.264 is one of the most commonly used video coding standards, jointly developed by the international standards bodies ITU-T (International Telecommunication Union) Video Coding Experts Group (Coders) and ISO/IEC (International Organisation for Standardisation / International Electrotechnical Commission) Moving Picture Experts Group (MPEG). The standard was first published in 2003. It is also known as MPEG-4 Part 10 or AVC for Advanced Video Coding. According to [2], H.264 provides better compression efficiency from earlier standards such as MPEG-2 and MPEG-4 with flexibility in compressing, transmitting and storing video. It can give better performance and produces an average bitrate reduction of about 50% over MPEG-2 for the same video quality. According to [67] and [68], compressed video clips take up less transmission bandwidth and less storage space compared to older codecs. The H.264/AVC design consists of a video coding layer (VCL) and a network abstraction layer (NAL). VCL performs all the classic signal processing tasks and generates bit strings containing coded macroblocks. The main goal of the NAL is to adapt those bit strings in a network friendly design objective [5]. Integration of network adaptation and video coding can bring the best possible performance of a video communication system.

H.264 is an industry standard for video compression, which converts digital video into a format that takes up less capacity while being stored or transmitted. Applications such as digital television, DVD-Video, mobile TV, videoconferencing and internet video streaming are important technology in Video compression. The standardization of video compression makes it possible for products like encoders, decoders and storage media from different manufacturers to inter-operate. An encoder converts video into a compressed format and a decoder converts compressed video back into an uncompressed format [3] .

Developing and designing the standard is a challenge to the engineers or programmers who interface with an H.264 codec. H.264 has more options and many parameters than any previous standard codec. Getting the right controls and parameters is not an easy task for delivery of high compression performance. On the other hand, wrong controls and parameters result in poor-quality pictures and/or poor bandwidth efficiency [4].

2.3.1 Video Coding Concepts

Video coding standards such as H.264/AVC define converting a raw video source into a specified bitstream. A typical compliant video coding system consists of several modules that perform operations such as motion compensated prediction, transform coding, rate control, run-length coding and entropy coding. Although the encoded bitstream syntax is specified by a standard, many of the encoder modules are left open. In [4], compression involves a complementary pair of systems, a compressor (encoder) and a decompressor (decoder). Before transmitting or storing video data, the encoder converts the source data into a compressed form and the decoder converts the compressed form back into a representation of the original video data. The encoder/decoder pair is often described as a CODEC enCOder/DECOder. Data compression is achieved by removing redundancy. Redundant components are not necessary for faithful reproduction of the data. [4] Iain Richardson has contributed a lot to video compression in his books.

A video CODEC in Figure 2-1 encodes a source image or video sequence into a compressed form and decodes this to produce a copy or approximation of the source sequence.



Figure 2-1: A Video CODEC Process.

2.3.2 Basic coding structure for H.264/AVC

The typical video coding and decoding process is demonstrated in Figure 2-2 [3]. The encoder processes an input frame by splitting it into units of a Macroblock with 16x16 pixels. Macroblocks are organized into slices to represent the regions of a given frame to be encoded independently. The first image of a sequence or a random access point is typically Intra coded, without using information contained in the other pictures. A prediction of the macroblock is formed based on the current frame using intra prediction. For the other remaining frames that have already been coded and transmitted, inter prediction is used. A residual is formed by subtracting the prediction from the current macroblock. Then the block of residual is transformed using a 4x4 or 8x8 integer transform. The transform outputs a set of coefficients which is then quantized according to a quantization parameter QP.

In H.264, images in the input video are encoded in units of macroblocks, which are blocks of pixels that consist of 16 pixels in the horizontal direction and 16 pixels in the vertical direction; when a macroblock is encoded, information about how adjacent already-encoded macroblocks were encoded is utilized in order to achieve high compression performance [6].

The quantized transform coefficients are entropy coded and transmitted together with the side information for either Intra-frame or Inter-frame prediction. The quantized transform coefficients are inverse scaled and inverse transformed in the same way as at the decoder side; the decoded prediction residual is added to the prediction. The result is then fed into a deblocking filter which provides the decoded video as its output [7].

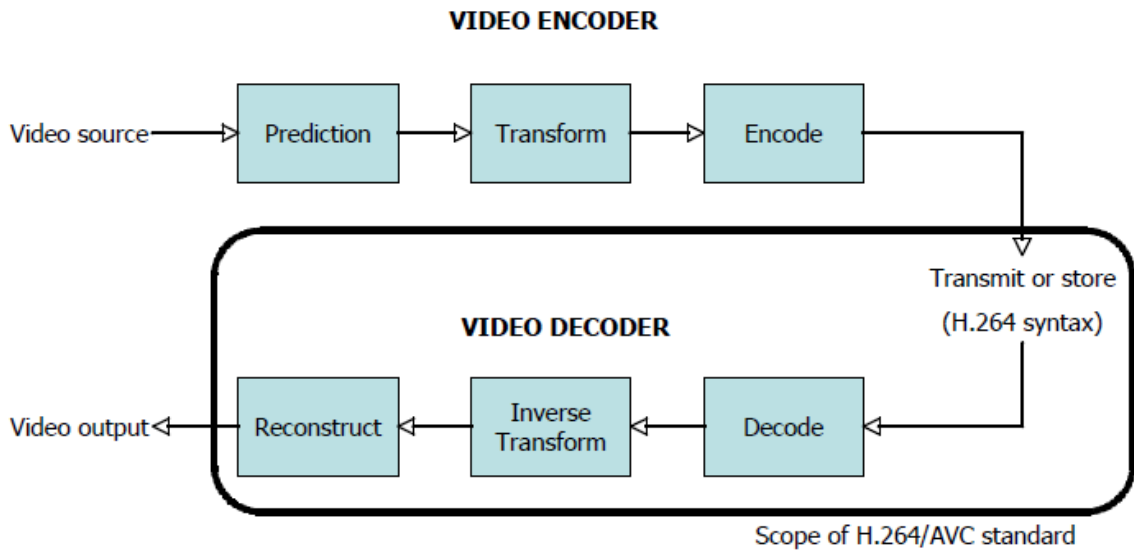


Figure 2-2: The H.264 video coding and decoding processes [3].

2.3.3 The H.264/AVC Coding Tools

This section describes Frame types and the tools that make H.264 such a successful video coding scheme. Intra coding, motion compensated prediction, transform coding, entropy coding and the adaptive de-blocking filter are discussed subsequently in the section.

Frame types

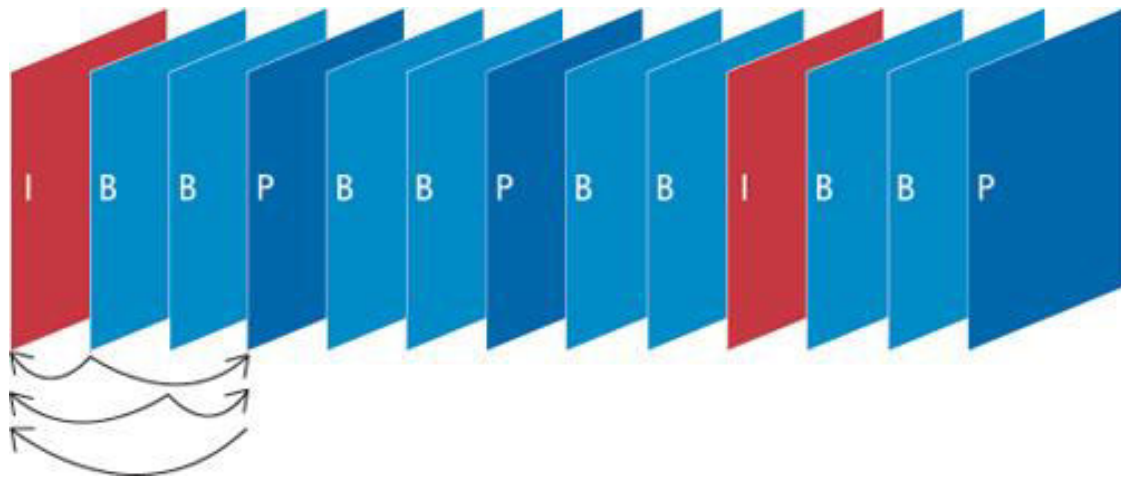


Figure 2-3: An example of sequence with I-, B- and P-frames [8].

The basic principle for video compression is the prediction between adjacent frames. There are three types of frames: Intra-frames (I-frames), forward-predicted frames (P-frames), and bi-directional predicted frame (B-frames) as explained in [8] (see Figure 2-3).

- I-frame: An Intra predicted frame is self-contained, having no dependency outside of that image. The more I-frames contained, the better the quality of the video; however, I-frames contain the greatest number of bits and therefore use more space on the storage medium.
- P-frames: Predicted-frames are predicted from the last I frame or P frame used as the reference frame. P-frames contain only the data that have changed from the reference frame.
- B-frames: Bidirectional; predicted from two references one in the past and one in the future I and P-frames. B-frames contain only the data that have changed from the preceding frame or are different from the data in the very next frame used as a reference.

2.3.4 The H.264/AVC encoder parameters

The efficiency of a coding algorithm is furthermore dependent on various parameters used. A significant number of coding options are available through the selection of various combinations of a large number of coding parameters. In [2], the H.264 CODECs have a large number of coding parameters to select from. The selection of coding parameters of a given video sequence as shown in Table 2-2, is used to achieve the optimum performance of the CODEC.

These parameters specify input/output control of the encoder, including source video and output sequence video file names, and the file format [9].

Table 2-2: List of coding parameters

Parameter	Meaning
IntraPeriod	Period of I-frames, i.e. frame will be coded using intra slices, every IntraPeriod frame.
NumberReferenceFrames	Sets maximum number of references stored in the buffer for motion estimation and compensation.
SearchRange	Sets allowable search range for motion estimation.
RDOptimization	Enables rate distortion optimized mode decision.
Quantization Parameter	Quantization parameter

IntraPeriod

IntraPeriod parameter represents the maximum period of I-coded frames in the encoded sequence. IntraPeriod can start with default values 0 which means that the first frame is coded as an I-frame and the following frames are coded as P-frames [9].

NumberReferenceFrames

NumberReferenceFrames is used to set the maximum number of references stored in Decoded Buffer for motion estimation and compensation. The number of reference frames used for inter motion search ranges from 0-16, the default value is 1 [9].

RDOptimization

RDOptimization is a method used for improving video quality in video compression. Rate-distortion optimization can be used to improve quality in any encoding video where decisions have to be made that affect both file size and quality simultaneously.

RDOptimization enables the Lagrangian based rate distortion optimized mode decision. The mode can take values from 0 to 2:

0: RD-off	Enable Low Complexity mode (default)
1: RD-on	Enable High Complexity mode
2: RD-on	Enable Fast High Complexity mode (does not support FExt profiles)

SearchRange

The search window size sets an allowable search range for Motion Estimation. It can take either of two values, 16 or 32. The default value is 16 and maximum is 32. If Rate Distortion Optimisation is enabled, the search window is centred around median predictor, not (0, 0) [9].

Quantization parameter (QP)

➤ QPISlice

Sets quantization parameter value for I slices. Allowable values are in the range of 0 to 51. Default value is 24.

➤ QPPSlice

Sets quantization parameter value for all P slices. Allowable values are in the range of 0 to 51. Default value is 24.

➤ QPBSlice

Quantization parameter used for non-stored B slices. Should be in the range [0-51]. Default value is 24.

When QP is low, almost all details are retained; however, as QP is increased, some of those details are aggregated so that the bit rate drops, but at the price of some increase in distortion and some loss of quality [9].

2.4 HEVC video coding standard

Due to the higher demand for higher resolution videos, in 2010 many proposals were submitted both from representatives of industry and academia, which in turn led to the development of the so-called High-Efficiency Video Coding (HEVC) standard during the next two and a half years. The first edition of HEVC was officially finalized in January 2013, and after that, the final aligned specification was approved by ITU-T as Recommendation H.265 and by ISO/IEC as MPEG-H, Part 2. The H.265/MPEG-HEVC standard was designed to be applicable for almost all existing H.264/MPEG-AVC applications, while putting emphasis on high-resolution video coding. Since the development process of H.265/MPEG HEVC was also driven by the most recent scientific and technological achievements in the field of video coding, dramatic bit-rate savings were achieved for substantially the same visual quality, when compared to its predecessor H.264/MPEG-AVC [11], [12].

Figure 2-4 shows a block diagram of a block-based hybrid video encoder with some characteristic ingredients of HEVC regarding its novel block partitioning concept. In a first step of this new block partitioning approach, each picture in HEVC is subdivided into square blocks of the same size, each of which serves as the root of a first block partitioning quadtree structure, the coding tree, which are therefore referred to as coding tree blocks (CTBs). The CTBs can be further subdivided along the coding tree structure into coding blocks (CBs), which are the entities for which an encoder has to decide between intra-picture and motion-compensated prediction [13].

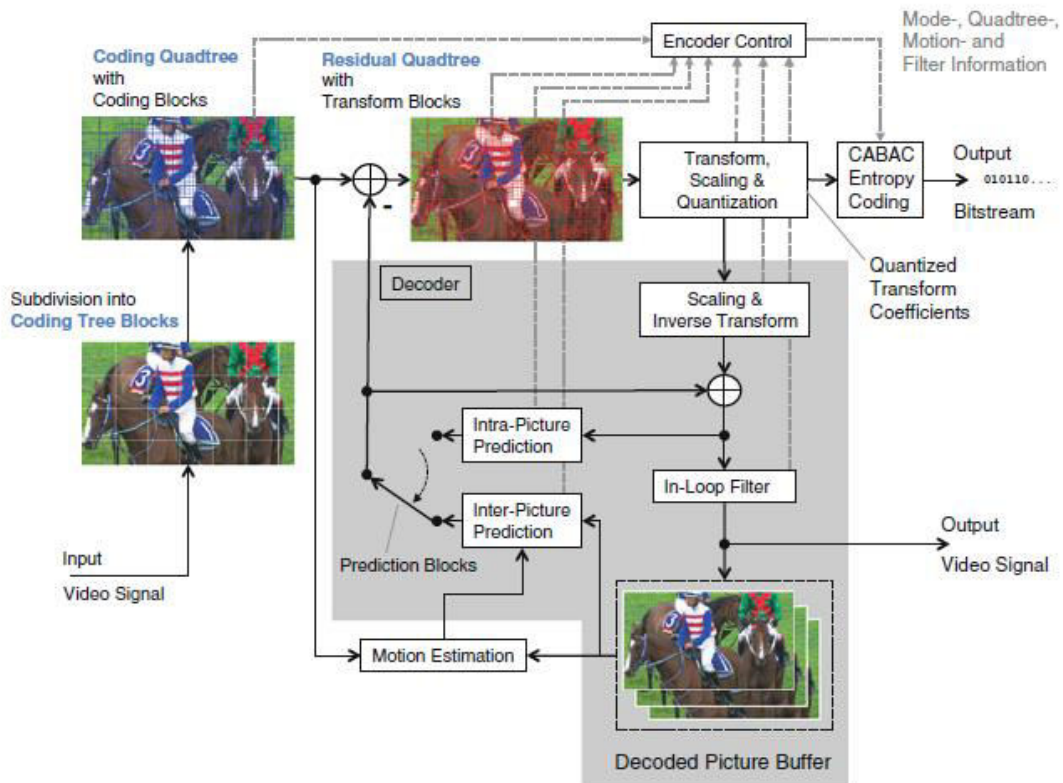


Figure 2-4 Block diagram of an HEVC encoder with built-in decoder [13]

2.4.1 Differences of HEVC from H.264/AVC

The HEVC standard has the ability of saving significant bandwidth over H.264 encoded content having similar quality. Some of the key differences between H.265 (HEVC) and H.264 (MPEG 4 AVC) are listed in Table 2-3.

Table 2-3 H.265 (HEVC) and H.264 (MPEG 4 AVC) [14].

Category	H.264	H.265
Names	MPEG 4 Part 10 AVC	MPEG-H, HEVC, Part 2
Progression	Successor to MPEG-2 Part	Successor to H.264
Key Improvement	<ul style="list-style-type: none"> - 40-50% bit rate reduction Compared to MPEG-2 with growth of HD content delivery over network. - Support Up to 4K (4,096×2,304) - Supports up to 59.94 fps 	<ul style="list-style-type: none"> - 40-50% bit rate reduction at the same quality compared to H.264. - Up to 8K UHD TV (8192×4320) - Supports up to 300 fps
Compression Model	<ul style="list-style-type: none"> - Hybrid spatial-temporal prediction model - 9 directional modes for intra prediction - Macro Blocks structure with maximum size of 16x16 - Entropy coding is CABAC and CAVLC 	<ul style="list-style-type: none"> - Enhanced hybrid spatial-temporal prediction model - 35 directional modes for intra prediction - Supporting larger block structure (64x64) - Entropy coding is CABAC only
Drawbacks	Unrealistic for UHD content delivery due to high bit rate requirements.	Computationally expensive due to larger prediction units and expensive Motion Estimation.

Both HEVC intra coding and H.264/AVC intra coding are based on spatial sample prediction [13]. Yet, the intra coding methods in HEVC can be elaborated in a number of ways.

1. First of all, the set of supported prediction block sizes is extended up to 32×32 to be aligned with the HEVC coding structures and to improve reconstruction of smooth image areas.
2. Secondly, the number of available directional modes is extended from 8 to 33 to improve modelling of directional textures. In HEVC, the whole range of directional predictors is made available for both luma and chroma blocks, while in H.264/AVC the number of available directional prediction modes for chroma is limited to two horizontal and vertical modes.
3. Another advantage of HEVC is its ability to pad the missing reference samples and allow usage of all the prediction modes independent of availability of certain reference samples.
4. The HEVC intra mode coding also uses an approach different from that of H.264/AVC. Due to the large number of intra modes in HEVC, the luma intra mode is signaled by using three of the most probable modes and the selected luma mode is always made available as one of the candidate modes for the corresponding chroma blocks.

The Table 2-4 below summarizes key differences of intra prediction in H.264/AVC and HEVC.

Table 2-4 Intra prediction techniques between H.264 and HEVC [13]

Functionality	H.264/AVC	HEVC
Prediction block sizes	4×4 , 8×8 and 16×16	4×4 , 8×8 , 16×16 and 32×32
Luma intra prediction modes	9 (4×4 and 8×8), 4 (16×16)	35
Chroma intra prediction modes	4	4 + the luma mode
Reference sample smoothing	8×8	8×8 and above
Boundary smoothing	N/A	Used for directly horizontal, vertical and DC modes
Operation when reference samples missing	Use DC mode	Reference sample substitution
Number of most probable modes in mode coding	1	3

2.4.2 Compression Performance Analysis in HEVC

To conduct HEVC performance evaluations, a well-defined encoder setting and testing environment need to be established with HEVC and AVC reference encoder software [13].

2.4.2.1 Encoder Software

In the standardization of HEVC, the reference software, named as HM (HEVC Test Model, reference software), has been developed as a common SW platform for further improvement and study.

The HM reference software is maintained at two sites [15]. HHI (Heinrich Hertz Institute) maintains the main SVN server and BBC (British Broadcasting Corporation) maintains the mirroring repository site.

- HM software repository (main at HHI)[16]
- HM software repository (mirror at BBC)[17]

2.4.2.2 Prediction Structure

The performance evaluation defines the following prediction structures.

- All Intra (AI)
- Random Access (RA)
- Low Delay P picture (LDP)
- Low Delay B picture (LDB)

In these configurations, the QP (Quantization Parameter) value can be modified by adding to it a “QP offset” value. That is, CTC defines QP of the first picture (QP of an I picture, QPI, with I picture defined below) and the QP of the following pictures are derived as QPD (QPICQP offset), with QP offset being determined according to the picture type (e.g., P & B pictures, defined below) or a picture temporal ID. An I (intra) picture refers to a picture that can be decoded independently without requiring prediction data from other decoded pictures. A P (predicted) picture, in general, requires picture sample data from one other I, P or B picture to generate each predicted sample block. A B (bi-predicted), in general, requires picture sample data from two other I, P or B pictures to generate each predicted sample block [13].

2.4.2.3 Test Sequences

Test sequences are defined according to the picture size and applications and they are classified into six classes (class A to class F). Class A is the set of sequences with higher resolution than 1080p HDTV. The sequences are used to evaluate the coding performance of 4K/8K video. To reduce computation time, picture sizes are cropped to 2,560 x 1,600 pixels.

Class B is for coding performance evaluation of 1080p HDTV and the set contains HDTV sequences, with a picture size of 1,920 x 1,080 pixels. Classes C and D are the set of test sequences with picture sizes of 832 x 480 pixels and 416 x 240 pixels, respectively. Test sequences in these two classes are for coding performance evaluation of mobile applications.

Class E is the set of test sequences with a picture size of 1,280 x 720 pixels. It is used to evaluate coding performance of low-latency applications such as visual communications. CTC, in addition, defines class F sequences for coding performance evaluation of non-camera captured content such as video screen content, containing, for example, text and computer graphics [13].

In addition, the test sequences defined are used for both objective and subjective quality performance analysis in [13].

2.4.2.4 Rate Distortion Curves

When evaluating the coding performance of a video codec, a graph of R–D curve (Rate–Distortion Curve) is used. R–D curve is generated by plotting the encoded results, in terms of bit rate versus the resulting quality, in a graph. The horizontal axis denotes the bit rate and the vertical axis denotes a measure of distortion or quality of encoded video. In general, a higher compression ratio results in a lower bit rate; however, picture quality is generally reduced. Low compression ratio, on the other hand, improves picture quality but at the cost of an increase in bit rate. Since a high coding efficiency codec can achieve higher quality at lower bit rates, the R–D curve moves toward the upper left.

As an objective measurement of picture quality, PSNR is widely used. PSNR can be calculated by the following (Equation 2-4).

$$PSNR = 10 \log_{10} \frac{(2^{bitdepth} - 1)^2 \times W \times H}{\sum_i \{O_i - D_i\}^2}$$

(Equation 2-4)

2.5 Tools for Encoding, Decoding, Modelling and Optimisation

2.5.1 JM Reference Software

Joint Model (JM) is H.264/AVC codec implementation publicly made available by the Joint Video Team (JVT) H.264/AVC Reference Software, 2009. The current software version is JM 18.6. This software includes means for setting the encoder and decoder input parameters and the software package contains a Visual Studio workspace. The user has to select the appropriate solution according to his/her .NET package. These workspaces include the following projects:

- lencod H.264/AVC reference encoder
- ldecod H.264/AVC reference decoder

After selecting the desired project and the appropriate compilation mode, a compilation will create the binaries lencod.exe or ldecod.exe in the bin directory [9]. Microsoft Visual Studio is a complete set of development tools for building ASP.NET Web applications, XML Web Services, desktop applications, and mobile applications. Visual Basic, Visual C#, and Visual C++ all use the same integrated development environment (IDE), which enables tool sharing and eases the creation of mixed-language solutions. In addition, these languages use the functionality of the .NET Framework, which provides access to key technologies that simplify the development of ASP Web applications and XML Web Services. [18]

2.5.2 The Intel® VTune™ Amplifier XE

The Intel VTune Amplifier XE is a commercial application for software performance analysis on Windows and Linux operating systems. On Windows systems, the VTune Amplifier XE integrates into Microsoft Visual Studio software and has both GUI and command line interfaces or can be accessed as a Standalone VTune Amplifier XE GUI. Although AMD and Intel hardware work on basic features, Intel-manufactured CPU supports advanced hardware-based sampling. Therefore, it provides information on code performance for users developing serial and multithreaded applications. Intel VTune provides basic Hotspot analysis to identify where and how an application is spending time, determine the most time-

consuming program units, and detect how they were called. The time taken by the instructions is indicative of any stalls in the pipeline during instruction execution. The tool can also be used to analyse thread performance [19].

2.5.3 WEKA machine learning toolkit

Waikato Environment for Knowledge Analysis [20] (WEKA) is a machine learning toolkit introduced by Waikato University, New Zealand. Used for research, education and projects, it is open source software written in Java (GNU) Public License and it can be run on Windows, Linux and Mac. Data mining tasks are implemented using machine learning algorithms. There are several versions of WEKA: a GUI version adds graphical user interfaces, and a book version is command-line only. WEKA has the capability to read in ".csv" format files, where many databases or spreadsheet applications can save or export data. Actually, once data is loaded into WEKA, the data set can be saved into an ARFF format. Therefore a series of operations are performed using WEKA's attribute. Many classification methods have been developed with the aid of learning algorithms. All these classifiers are basically learning methods and adopt sets of rules. Regression is a technique used to predict a value of a numerical class, in contrast to classification, which predicts the value of a nominal class. Given a set of attributes, the regression builds a model, usually an equation that is used to compute the predicted class value [21].

Regression analysis is a statistical approach that can be used to investigate the relationship between variables; typically, the relationship between a dependent variable and one or more independent variables [22]. It is used for many purposes like forecasting, predicting and finding the causal effect of one variable on another. There are many features of regression analysis that make it a popular tool:

- Can handle multiple co-related predictor variables
- Used for continuous and categorical variables
- Addresses unknown parameters
- Studies the effect of one predictor variable on a dependent variable
- Higher-order terms can be used for modelling and data analysis

2.5.3.1 Function Based Approaches

There are a number of statistical software solutions that provide different kinds of regression techniques including linear regression.

Linear regression can be simple linear regression with only one variable or multiple linear regressions with multiple explanatory variables. This research uses linear predictor functions for data modelling wherein unknown parameters are estimated from the data. The mathematical technique is to find the straight line that best-fits the values of a linear function and plotting it on a scatter graph as data points. If a 'best fit' line is found, it can be used as the basis for estimating the future values of the function by extending it while maintaining its slope.

As per [23], linear regression is a natural technique to consider as a main method in statistics. When the outcome or class is numeric, and all the attributes are numeric, the class is expressed as a linear combination of the attributes with predetermined weights:

$$x = w_0 + w_1a_1 + w_2a_2 + \dots + w_ka_k$$

where x is the class, a_1, a_2, \dots, a_k are the attribute values; and w_0, w_1, \dots, w_k are weights.

2.5.3.2 Ensemble Classifiers

In [23] it has been demonstrated that whether the learning algorithm is appropriate to the problem at hand or not, an obvious approach to making decisions more reliable is to combine the output of several different models. Several machine learning techniques do this by learning an ensemble of models and using them in combination.

These schemes are called bagging, boosting, and stacking. They are general techniques that are able to be applied to classification tasks and numeric prediction problems.

Therefore new methods have arisen for example, shaking up bagging by adding random variants of classifiers to improve performance. This realization has led to improved procedures. In the past few years methods have been developed that combine the performance benefits of committees with comprehensible models.

Bagging: Bagging stands for Bootstrap Aggregating, an ensemble method that creates separate samples of the training dataset and creates a classifier for each sample. The results of these multiple classifiers are then combined. The trick is that each sample of the training dataset is different, giving each classifier that is trained a subtly different focus and perspective on the problem.

However, it turns out that bagging produces a combined model that often performs significantly better than the single model built from the original training data, and is never substantially worse. [23]. Bagging can also be applied to learning schemes for numeric prediction—for example, model trees. The only difference is that instead of voting on the outcome, the individual predictions, being real numbers, are averaged. The bias–variance decomposition is applied to numeric prediction by decomposing the expected value [23].

The popular base classifiers that can be used with bagging are the following:

weka.classifiers.trees.REPTree

This is a fast decision tree learner. It builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning (with back-fitting). It only sorts values for numeric attributes once. Missing values are dealt with by splitting the corresponding instances into pieces.

weka.classifiers.trees.RandomForest

Class for constructing a forest of random trees. The Random-Forest classifier is located in WEKA's trees package

weka.classifiers.meta.AdditiveRegression

Meta classifier that enhances the performance of a regression base classifier. Each iteration fits a model to the residuals left by the classifier on the previous iteration. Prediction is accomplished by adding the predictions of each classifier. Reducing the shrinkage (learning rate) parameter helps prevent overfitting and has a smoothing effect but increases the learning time.

weka.classifiers.meta.RandomSubSpace

This method constructs a decision tree based classifier that maintains the highest accuracy on training data and improves on generalization accuracy as it grows in complexity. The classifier consists of multiple trees constructed systematically by pseudo-randomly selecting subsets of components of the feature vector; that is, trees constructed in randomly chosen subspaces.

2.5.3.3 The Validation Metrics

In order to evaluate the accuracy of the prediction models, in the proposed research one of the possible objective metrics have been used, namely the Correlation Coefficient (CC), the metric popularly used to compare accuracy in modelling.

To determine the linear relationship between input variables (X) and target variables (Y). It takes values between -1 and 1. The Correlation Coefficient is defined as follows:

A positive value of Correlation Coefficient means that the two variables move in the same direction with respect to their means. A negative value means they move in opposite directions with respect to their means. A value close to 0 means the two variables has little linear dependency [24].

This means, for the predictions of the proposed work in this thesis, the Correlation Coefficient should be maintained close to 1 as much as possible, as this would facilitate training accurate models.

2.5.4 Matlab Optimisation Toolbox

The MATLAB Optimization Toolbox is an application of MATLAB which provides algorithms for searching solution spaces for an optimal solution. The Optimization Toolbox can be used to either minimise or maximise objectives while satisfying user supplied constraints. The MATLAB Optimization Toolbox contains many different types of functions for searching for solutions. The function chosen for this thesis was the gamultobj Fitness function to optimize [25].

The multi-objective solver uses a genetic algorithm (gamultobj), available in the MATLAB optimisation tool-box R2014b 64-bit.

According to [26], the optimisation procedure was implemented by a multiobjective genetic algorithm (GA). Starting from an initial population of randomly created individuals representing candidate solutions, in this scenario a heat exchanger of specific configuration and conforming to the design specifications, the GA uses the concept of survival of the fittest to produce more desirable individuals in subsequent evolutionary generations of the population. The cost value of each candidate solution denotes the fitness function of the individual, which is a measure of its quality relative to the entire population.

The evolution starts from a population of randomly generated individuals and is an iterative process, with a new population in each iteration, called a generation. In each generation, the fitness of every individual in the population is computed; the fitness being the value of the objective function.

The new generation of candidate solutions is then utilized in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

The genetic algorithm uses three main types of rules at each step to create the next generation from the current population [25].

- Selection rules select the individuals called parents that contribute to the population at the next generation.
- Crossover rules combine two parents to form children for the next generation.
- Mutation rules apply random changes to individual parents to form children.

The multi-objective genetic algorithm (`gamultiobj`) works on a population by using a set of operators that can be applied to the population. The initial population is generated randomly by default. The next generation of the population is derived from the non-dominated rank, which is assigned to each individual using the relative fitness, and a distance measure of the individuals in the current generation.

The multi-objective GA function “`gamultiobj`” uses a controlled elitist genetic algorithm (a variant of NSGA-II [27]). While the elitist GA always favours individuals with better fitness value (lower rank), the controlled elitist GA also favours individuals that tend to increase the diversity of the population even if they have a lower fitness value. In order to ensure the convergence to an optimal Pareto front, it is very important to maintain the diversity of population. This is done by controlling the elite members of the population as the algorithm progresses, by using the options, 'ParetoFraction' and 'DistanceFcn'. The Pareto fraction option limits the number of individuals on the Pareto front (elite members) and the distance function helps to maintain diversity on a front by favoring individuals that are relatively far away on the front.

The optimization problem intention is minimizing undesirable effects and/or maximizing desirable effects. Any or both of these will form the objective of

optimization from which the objective function is formulated. An optimization problem could be a single objective or MOO. For MOO, the final solution of the objective functions will represent a compromise (tradeoffs) between different objectives that may be totally conflicting, partially conflicting or non-conflicting [28].

Maximizing vs. Minimizing

Global Optimization Toolbox optimization functions minimize the objective or fitness function. That is, they solve problems of the form

$$\min_x f(x).$$

If it is required to maximize $f(x)$, $-f(x)$ should be minimised, as the point at which the minimum of $-f(x)$ occurs is the same as the point at which the maximum of $f(x)$ occurs.

2.6 An overview of End-to End video streaming

When video signals are transported over an IP network, they are most often compressed. In this context, compression means to reduce the number of bits that are required to represent the video image. Video technology users are free to choose whether or not to employ compression for their video signals. However, it is important to understand that the choice of a compression method can sometimes mean the difference between success and failure of a video networking project. Today, most communication systems depend on compression technology. So an understanding of video and audio compression is important; also, using modern video transport systems, including video over IP networks. [29]

2.6.1 Overview of video streaming

Streaming video is a method for delivering content over an IP network that can be used for a variety of purposes. Streaming is a process for sending video and audio content to a user that is watched as the same time, just like watching a network broadcast television. An active network connection requires delivering video content to the playback device, and the content is not normally stored after playback is complete. On the other hand, when podcasting and video file-sharing technologies are used, the content is stored on the received device as shown in Figure 2-5. This output can be stored or transmitted over a network. Before the data

file can be used, it must be restored to its original size, via a decompression engine. Note that a compression engine is often called an encoder, and the decompression engine is commonly called a decoder.

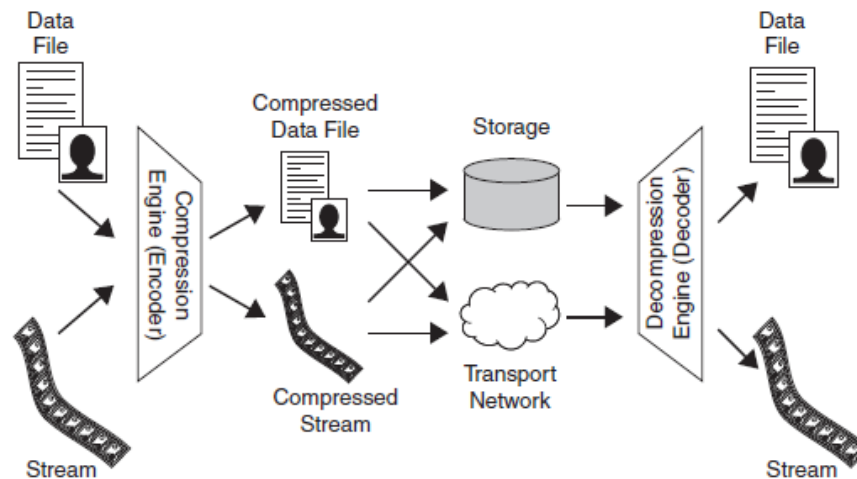


Figure 2-5 Video streaming diagram [29]

Once the video is compressed, the compressed bitstream is then encapsulated into IP packets, adding the headers and other data required to comply with a specific protocol. A bitstream consists of a sequence of data units called network abstraction layer (NAL) units, each of which contains an integer number of bytes. The first bytes of a NAL unit produce the NAL unit header, while the rest of the NAL unit contains the payload data. There are two classes of NAL units in HEVC – video coding layer (VCL) NAL units and non-VCL NAL units. Each VCL NAL unit carries one slice segment of coded picture data while the non-VCL NAL units contain control information that typically relates to multiple coded pictures. Some NAL units carry parameter sets containing control information that apply to one or more entire pictures, while other NAL units carry coded samples within an individual picture. [13].

Transport protocols are used to control the transmission of data packets in conjunction with IP. UDP, or User Datagram Protocol, is often used for video and other data that is very time sensitive.

2.6.2 Overview of the VideoLAN streaming solution

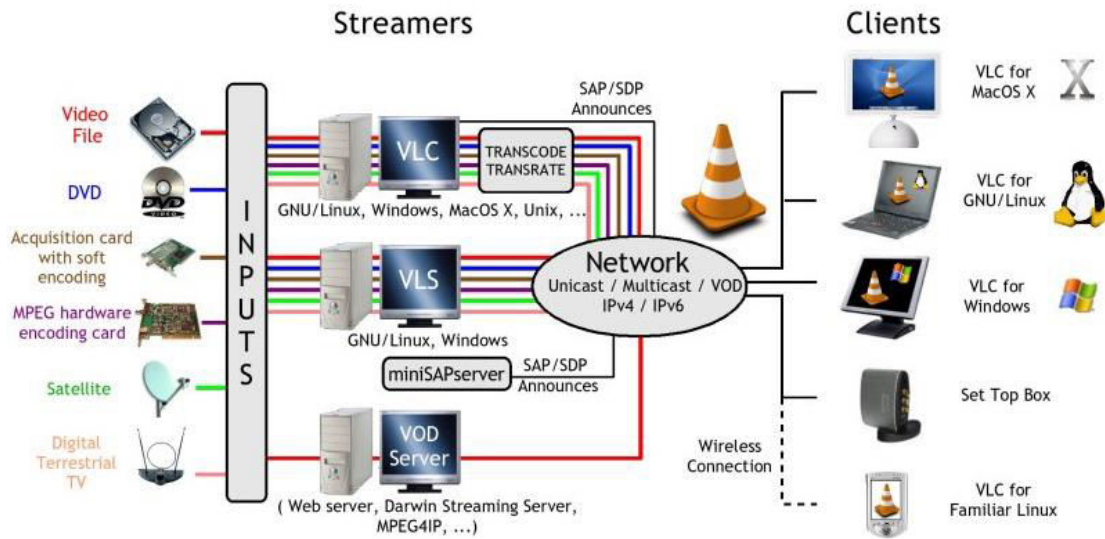


Figure 2-6 VideoLAN streaming solution [30]

VLC media player can be used as a *server* and as a *client* to stream and receive network streams. VLC is able to stream all that it can read [30]. The network on which you set up the VideoLAN solution can be as small as one Ethernet 10/100Mb switch, hub or direct peer to peer connection (see Figure 2-6).

2.6.3 EvalVid - A Video Quality Evaluation Tool-set

EvalVid [31] is a framework and tool-set for evaluation of the quality of video transmitted over a real or simulated communication networks. It is targeted at researchers who want to evaluate their network designs or setups in terms of user perceived video quality. As well as measuring QoS parameters of the basic network, like loss rates, delays, and jitter, standard video quality metrics like PSNR and a subjective video quality evaluation metric of the received video are provided.

Some useful tools from EvalVid are as follows:

- MP4-Container, the tool used to produce MP4-files, e.g., *MP4Box* is available.
- Creating Reference Videos from mp4 *ffmpeg* is used.
- Sender and Receiver per RTP/UDP to a specified destination host. *mp4trace* is able to send a hinted mp4-file. And a *trace file* is generated.

- Evaluation process is the calculation of the reference PSNR. That is the PSNR of the coded and then decoded in relation to the uncoded raw video source.

2.6.4 Network simulator using Riverbed Modeler (Opnet)

Since the design of networks is important, the software Opnet is a type of simulation tool which designs a network that demonstrates how the infrastructure would look if put in place. OPNET is now part of Riverbed, with Network simulator Technologies. This simulator provides comprehensive development features, which eases the process of designing the real world scenario and simulating the network models [32].

Riverbed Modeler provides a modeling and simulation environment for designing communication protocols and network equipment. The simulations run by representing real world devices such as configured nodes and links in the designed topology, and results are analysed after running. In riverbed, a project can be created and edited in project editor, where nodes, links, utilities, subnets and application traffic can be included for the study of simulation. This reduces the need for time-intensive and expensive real hardware devices. In addition, the modeler incorporates audio, video traffic, HTTP, FTP, and email traffic.

2.7 Summary

This chapter represented the fundamental background of H.264, HEVC video coding standards that are essential in carrying out the optimisation of the CODECs. The chapter also presented the software tools and systems used for video CODECs, optimisation, performance analysis, modelling, optimisation, network simulation and end-to-end video delivery. The understanding of the operational aspects of these tools is essential in carrying out the research presented in the chapters that follow.

The following chapter presents the literature review study of the previous research.

Chapter 3

Literature Review

3.1 Introduction

The literature review chapter presents an overview of previous research that has taken place in relation to the current research study. Many fields are involved in this research. This chapter introduces the reader to the literature of H.264 and HEVC Encoders/decoders, optimisation of video coding/decoding and the systems associated with the end to end delivery of video; it also explains the tools and methods that have been employed within this research. This chapter also provides a review of literature on two different approaches to the Optimisation of video CODECs, i.e. parameter based and algorithmic based. The two approaches are discussed in section 3.2. The theory and techniques of multi objective optimisation are discussed in section 3.3. Section 3.4 presents the literature of H.264 encoder and decoder. Section 3.5 presents the literature of High Efficiency Video Coding (HEVC), the next generation video coding standard. Furthermore, machine learning models are presented in section 3.6. The relevant literature on the End-to-End delivery of video is presented in section 3.7. Finally, a summary is given in section 3.8 drawing upon a number of conclusions that justifies the novel and contributory research conducted within the research context of this thesis.

3.2 Optimisation of Video Coding

In the literature, many works have been conducted on Optimisation of video compression algorithms. They can be broadly classified into two categories; Algorithmic-based Optimisation and Parameter-based Optimisation. The state-of-the-art of the two optimisation methods is introduced in this chapter in the following subsections. Algorithmic based Optimisation focuses on the uses of the best possible motion estimation techniques, rate-distortion algorithms, and mode-prediction algorithms etc., which are essential parts of a video compression system for direct performance optimisation of a given video CODEC. In parameter-based Optimisation, the key focus is to select values for the massive number of parameters

approximately 150 parameters that are associated with a CODEC so that the CODEC's performance will be optimised. No changes are made to the algorithms. Choosing the right parameter set is extremely important.

The algorithm-based optimisation methods focus on the direct performance optimisation of a given algorithm. The techniques and algorithms required to implement a standard are well-defined. Based on the previous research, it is well known that motion estimation is the computational bottleneck of a video encoder. In [33] the optimisations include adaptive diamond pattern based motion estimation, fast sub-pel motion vector refinement and heuristic Intra prediction. Several platform independent optimisations for a real-time H.264 encoder were proposed.

This information is used for further research areas of motion estimation. In this research by setting NumberReferenceFrames will Sets maximum number of references stored in the buffer for motion estimation and compensation. And SearchRange allowable search range for motion estimation.

The research conducted by [34], designed a memory optimisation technique for a H.264 video decoder and proposed an optimised memory management decoding method to reduce memory accessing overhead of the H.264/AVC decoder. Improved motion compensation and motion vector prediction modules were designed and implemented. Result and analysis showed that the proposed methodology does not cause any quality degradation in image PSNR (Peak Signal to Noise Ratio) performance and improves memory accessing performance of the decoder by reducing memory bottlenecks in a software H.264/AVC decoder.

The work presented in [35] proposed Rate-Distortion Optimized Distributed Packet scheduling of multiple video streams over shared communication resources, which enables the multiple senders to coordinate their packet transmission schedules, so that the overall video quality across all streams is maximized for the given available data rate on the shared channel.

A joint complexity-distortion optimisation approach for real-time H.264 video encoding under a power-constrained environment was proposed in [36]. In this paper, the authors proposed Complexity Configurable Motion Estimation (CAME) and Complexity Configurable Mode Decision for computation allocation model (CAMD) algorithms for H.264 video encoding to allocate the computational

resources. Moreover, the proposed algorithms can be easily integrated into any existing H.264 encoder as well as other standard video encoders.

The current research has demonstrated the use of a multi-objective optimisation framework based on carrying out investigations related to a H.264 CODEC. Through the use of this framework it was demonstrated how optimal configurations for the encoder and decoder performance could be obtained.

3.2.1 Parameter-based Optimisation

Parameter based optimization approaches have received attention after the CODECs were standardised, Video optimisation research has mainly been focused around standardization of video CODECs, e.g. H.262, HEVC etc. For H.264 in particular, a significant amount of effort has been put into the optimisation of the associated algorithms, during the international standardization. However, once a CODEC has been standardised, in order to comply with the standardization, the algorithms cannot be changed or modified. Hence the standardization of a video CODEC stops the opportunity that exists for further optimisation of the algorithms. However the presence of a large number of parameters that can be set and will have an impact on how the CODEC will perform, provides one an opportunity to further optimise, in particular when the operational constraints are known. Parameter based optimisation is useful at this stage.

The work of [37] proposed a novel framework for the multi-objective optimisation of a video CODEC based on genetic algorithms. The framework focuses on the development of joint complexity-memory-rate-distortion (C-M-R-D) optimisation of a H.264 video CODEC. An important aspect of the proposed framework is that it jointly considers the optimisation of multiple objectives in both the encoder and decoder. The framework in Figure 3-1 illustrates the optimisation procedure. SPSS categorical regression was used to create the objective functions and to determine the coefficients of each significant polynomial term of the objective. The polynomial terms are defined to include all possible combinations of the decision variables. These functions were then fed to the NSGA-II optimisation tool along with the quantified values of decision variables. A Genetic algorithm has the ability to find multiple optimal solutions in a single simulation run.

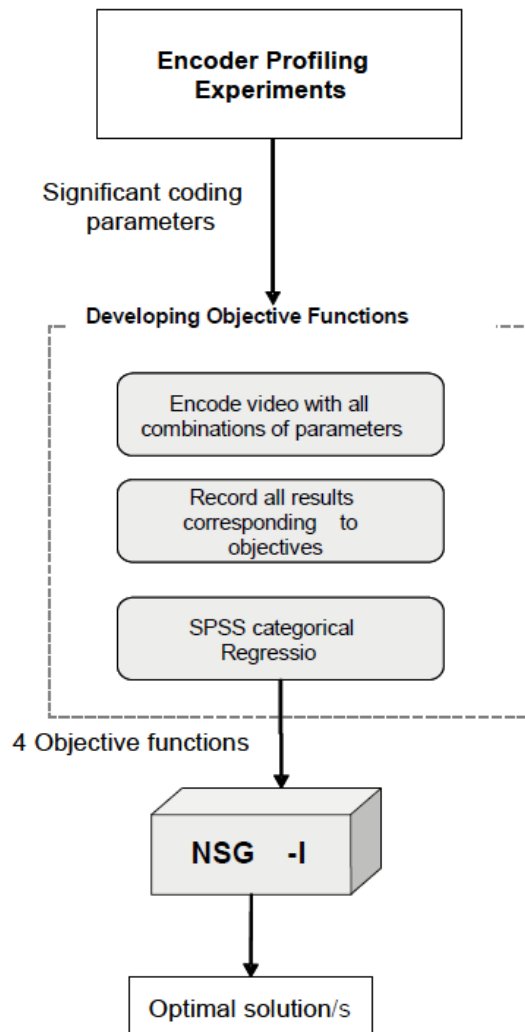


Figure 3-1: Multi-objective Optimisation framework used in [37].

The performance of the optimisation framework was used in the H.264/AVC codec JM 15.1, made by the Joint Video Team (JVT). The proposed framework is designed to jointly minimize the complexity, memory usage bit rate and to maximize the quality both on the encoder and decoder, of the compressed video stream, while achieving maximum visual quality. According to this paper’s results, the framework can produce an optimal coding parameter set for video sequences.

Comparing to the current thesis machine learning was used to create the objective functions rather than SPSS for the determination of significant coding parameters of video CODECs. Based on the results conducted it’s concluded that algorithm Linear Regression as the best practical solution.

In [38] the challenge was to determine H.264 parameter settings that result in low complexity but still offer high video quality. Two fast algorithms for finding the H.264 parameter settings were proposed: the generalized Breiman, Friedman, Olshen, and Stone [39] algorithm called GBFOS-basic and the GBFOS-iterative

algorithm that take about 1% and 8%, respectively, of the number of tests required by an exhaustive search. Both algorithms perform within a maximum PSNR difference of 0.71 dB when using the same training and test data set. The generalized BFOS algorithm is an extension of an algorithm for optimal pruning in tree-structured classification and regression to coding. The x264 encoder has been compared with different commercial H.264 (x264 2006). The research concludes with the statement: “Choosing the right set of encoder parameters results in efficiently coded video while an inappropriate selection of parameters wastes bits, sacrifices quality, and takes longer to encode.” [38].

An improvement was proposed in [39] where two algorithms for finding additional parameter settings for the GBFOS-basic algorithm were presented. It allows the encoder to choose a parameter setting that yields higher PSNR, while satisfying the encoding speed constraint. It was demonstrated that the PSNR improved by up to 0.71 dB and 0.43 dB, respectively. The algorithms were tested on both Linux and PocketPC platforms.

[40] developed a power-rate distortion (P-R-D) analysis framework; the paper analysed the encoding mechanism of typical video coding systems, and developed a parametric video encoding architecture which is scalable in computational complexity. The two major contributions in this work were firstly, to develop a parametric video encoding architecture which is fully scalable in power consumption; secondly, to successfully extend the traditional R-D analysis by considering another dimension, the power consumption, and establish the P-R-D analysis framework for mobile video encoding and communication under energy constraints. The investigation was carried out on the R-D performance of the complexity control parameters to establish an analytic P-R-D model. The author showed that the power-scalable video encoder is able to find the best configuration of complexity control parameters to maximize the video quality. In [41] it was observed that this analytical approach cannot be easily extended to other video encoders, such as H.264 video coding, since the video encoding mechanism used by such algorithms are more sophisticated. In this work, power-rate distortion (P-R-D) optimisation was proposed to minimize energy consumption for delay-tolerant over portable video communication applications. Using the proposed P-R-D optimisation technology, the energy consumption of video encoding can be significantly reduced (by up to 50%), especially in delay-tolerance.

A joint power-distortion model has been presented in [42] and analysed from two aspects: power consumption and video quality. In other words, how to find out the optimized encoding parameters with minimum power consumption? And how to calculate the optimized encoding parameters? So that the video quality is best. The framework was developed to solve the joint power and distortion optimisation problem, based on the investigation of the power consumption caused by the encoding and transmission. Using the proposed model, given any quality level, appropriate encoding parameters can be estimated so that the total power consumption is minimized. Unlike being given any available power level, the proper encoding parameters are also calculated so that the video quality is optimized. The analysis method can be easily extended to other video encoder and transmission configuration.

In this thesis it is shown that the proposed framework is flexible on the number of objectives that can jointly be optimized. The NSGA-II provides all sets of optimal results that jointly minimize CPU Time, bit-rate and maximizes quality.

One of the most critical issues in portable multimedia devices is to minimize the energy consumption and extend the operational lifetime of the system while still maintaining the required video quality. [43] Proposed a power-rate-distortion (P-R-D) model of a video encoding system to maximize its lifetime. An analytic model for P-R-D optimisation was proposed to find the optimum trade-off between power consumption and video encoding performance in [41]. Unfortunately, these works were not being applied on the hardware-based video encoder. As the power consumption becomes an important factor in portable devices, a hardware-based video encoder is often more suitable than a software-based video encoder to reduce the power consumption down to the acceptable level. In [43], the power-rate-distortion (P-R-D) model was discussed regarding the relations among power consumption, bit rate, and distortion of the video encoding process to reduce the power consumption of the video code while maintaining the video quality. The authors concludes that the proposed P-RD can be utilized by a designer to reduce the energy consumption of the video encoding system while maintaining the distortion using proper allocation of power to video encoder.

The study in [44] proposed a novel distortion prediction equation, optimisation of quantization parameter QP selection and a joint rate-distortion optimisation for the H.264 rate control algorithm. The authors demonstrated the improvement in image

quality and a computational saving of at least 34%, which was the recommended rate control algorithm in the H.264 reference software JM15.01. The author concluded that the algorithm could be implemented for real time applications requiring the optimisation of rate-distortion.

The main contribution in [45] is the insertion of an encoding parameter capable of controlling the encoding complexity and the possibility to select the desired encoding speed. The framework implemented was developed through an open source software implementation of the H.264/AVC, the x264 encoder. The results of the study showed that tight complexity control is attainable in practice, with very little loss in RD performance.

High Efficiency Video Coding (HEVC) is the next generation standard of coding being developed, the newest video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group [11]. HEVC in [46] provides significantly improved compression performance to reduce 50% bit rate compared to all existing video coding standards under the same visual quality. The paper proposed a hardware-friendly method for RDO of HEVC intra coding. The results of the study show that the proposed RD cost function provides 85.8% area reduction and 1260% throughput improvement in hardware design, with slight loss of bitrate and PSNR, which is very suitable for real-time encoder application.

The proposed framework in this thesis a Multi-Objective Optimisation was developed to determine the optimum coding parameters for a H.265 video CODEC.

The literature review reveals that generally two approaches are used in developing video coding algorithms. The algorithm base approach used at the development of algorithms aim to produce optimal performance. In contrast parameter based approaches are useful in optimising the performance of a video codecs which has been already finalised. This is important as the algorithm cannot be changed.

However the focus of this thesis is parameter-based Optimisation due to its relevance and nature of research and optimisation methodology adopted in this thesis. It important to understanding and to learn the parameters that can be set on a CODEC, parameters have impact on video CODEC performance, these would provide one of the opportunity to further optimise, in particular when the operational constraints are known. Parameter based optimisation is useful at this stage.

3.3 Theory of Multi-Objective Optimisation

An optimization problem could be a single objective or Multi-Objective Optimisation MOO. Many real-life scenarios where multiple objectives need to be satisfied in the course of optimization. Finding a single solution in such cases is very difficult, it may also happen that optimizing one objective leads to some unacceptably low value of the other objective(s). In such problems, referred to as multiobjective optimization problems (MOOPs). In the remaining part of this chapter, the focus on such methods of optimization multiple objectives.

In the past 15 years, evolutionary multi-objective optimisation EMO has become a popular and useful field of research and application. The purpose of multi-objective optimisation in a mathematical programming framework is to optimize different objective functions subject to a set of constraints. As the name suggests, a multi-objective optimisation problem (MOOP) involves optimizing a number of objectives simultaneously, which are to be minimized or maximized. The problem becomes challenging due to a conflict of objectives, and the optimal solution of an objective function is different from that of the other. Instead, there are several solutions called feasible solutions where the property of an improvement in an objective from one point to the other happens only due to a sacrifice in at least one other objective. Moreover, the problem usually has a number of constraints that any feasible solution must satisfy [47] and [48]. The entire decision variable space need not be feasible. The set of all feasible solutions is called the feasible region. In other words, the feasible region not only contains optimal solutions, but also solutions that are non-optimal. In general, optimisation refers to finding the best possible solution to a problem given a set of limitations or constraints. Moreover, the problem usually has a number of constraints that must be satisfied by any feasible solution. Since 2002 the multi-objective optimisation problem has been specified in its general form. Therefore, the general form of the MOOP may be stated as it appears in [47] and [48] as follows:

$$\begin{array}{lll}
 \text{Minimize/Maximize} & f_m(x), & m = 1, 2, \dots, M; \\
 \text{subject to} & g_j(x) \geq 0; & j = 1, 2, \dots, J; \\
 & h_k(x) = 0; & k = 1, 2, \dots, K; \\
 & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \dots, n.
 \end{array}$$

3.3.1 Multi-Objective Optimisation using Genetic Algorithms

During 1993–1995, a number of different evolutionary algorithm EAs were suggested to solve multi-objective optimisation problems. These algorithms demonstrated the necessary additional operators for converting a simple EA to a MOEA [49]. Over the past decade, a number of multi-objective evolutionary algorithms (MOEAs) have been suggested to find multiple Pareto-optimal solutions in one single simulation run. Simple evolutionary algorithms EA can be extended to maintain a various set of solutions. An EA can be used to find multiple Pareto-optimal solutions in one single simulation run. The non-dominated sorting genetic algorithm (NSGA) proposed in [50] was one of the first such EAs. In [49] the author proposed an improved version of NSGA, which was called NSGA-II. The NSGA-II outperforms two other existing MOEAs: Pareto-archived evolution strategy (PAES), strength-Pareto EA (SPEA). NSGA-II has been compared with another recently suggested constraint-handling strategy. These results encourage the application of NSGA-II to more complex and real-world multi-objective optimisation problems [49].

NSGA-II is also implemented in this thesis in the genetic algorithm It is shown that the proposed framework is flexible on the number of objectives that can jointly be optimized. Practical use of the proposed framework is described using the six videos mathematical formulation.

The features in [48] stated that the NSGA-II procedure is one of the popularly used EMO procedures which attempt to find multiple Pareto-optimal solutions in a multi-objective optimisation problem and has the following three features:

- It uses an elitist principle,
- It uses an explicit diversity preserving mechanism,
- It emphasizes non-dominated solutions.

The optimisation of H.264 and HEVC codecs in this research is carried out using implementation of NSGA-II. Two reasons have contributed to proposing to use a GA. The first reason is that a GA has the ability to find multiple optimal solutions in a single simulation run. The second reason is the presence of a well-established, popular, public domain, GA software tool, Non-dominated Sorting Genetic Algorithm NSGA-II [49] that can effectively be utilized in the proposed work.

Gamultiobj uses a controlled, elitist genetic algorithm, a variant of NSGA-II [47]. An elitist GA always favours individuals with better fitness value (rank). A controlled elitist GA also favours individuals that can help increase the diversity of the population even if they have a lower fitness value. It is important to maintain the diversity of population for convergence to an optimal Pareto front. Diversity is maintained by controlling the elite members of the population as the algorithm progresses. Two options, ParetoFraction and DistanceFcn, control the elitism. ParetoFraction limits the number of individuals on the Pareto front (elite members). The distance function, selected by DistanceFcn, helps to maintain diversity on a front by favoring individuals that are relatively far away on the front. The algorithm stops if the *spread*, a measure of the movement of the Pareto front, is small.

The literature review above demonstrated studies have been conducted for a generalised framework for multi objective optimisation of video CODEC. However there is potential for the further improvement and development of these generalised frameworks. The research conducted within this thesis will extend the state-of-art further and also for the first time carry out multi-objective optimisation of the latest video coding standard, HEVC.

3.4 H.264 Video Coding

A significant number of research articles have been published, in particular during its standardization period, on various aspects of H264 video coding. In [51] the H.264 software encoder engine that was developed by the NTT Cyber Space Laboratories were presented. Technologies for fast encoding, high compression performance, and reduced encoder operation cost, for video delivery services were proposed. The CODEC achieved a 20% reduction in the bitrate of the compressed video at the same encoding speed as compared to the original CODEC. The CODEC supported the 4:2:2 video formats, which is important for professional use, and reduces the cost of online video delivery services.

According to [52], JM 15.1 reference software and Intel IPP Integrated Performance Primitives H.264 codec are compared in terms of execution time and video quality of the output decoded sequence. PSNR, motion estimation time, encoding time, decoding time and the compression ratio of the H.264 file size encoded output were

used. Intel IPP H.264 implementation outperformed the JM 15.1 in all measures except for the compression ratio of H.264 file size obtained. The motion estimation time, encoding time and decoding time are much less in Intel IPP H.264 compared to that of JM 15.1.

The research presented in [53] conducted an investigation on the relationship between the bit rate and the CAVLC/UVLC decoding complexity. Understanding this relationship helped the researchers to choose the best encoding parameters to yield the best trade-off between the rate, distortion, and the decoding complexity performance. The proposed complexity control scheme of the H.264/AVC encoder generated a bit stream that is most suitable for a receiver platform with a power/hardware constraint. The experimental results showed that the H.264/AVC encoder can generate bit streams according to different entropy decoding complexity requirements accurately. It was shown that the resultant bit streams can be decoded at much lower complexity at the cost of small PSNR loss.

In [27] and H.264 decoder model was used to investigate the efficiency of a decoding system under various conditions for bitstreams, coding features and bitrates. H.264 achieves improved compression efficiency at the cost of increased computational complexity. Real-time execution of the H.264 decoding process provides a challenge on mobile devices due to low processing capabilities. The resulting idle time curves provide a powerful tool for extracting optimal buffer sizes and for estimating PSNR values where the decoder performs most efficiently. The simulation results obtained allowed for optimizing a decoding system for specific application aspects such as quality, memory requirements or core usage [54].

A novel method was proposed in [55] for Selective Encryption SE of H.264/AVC (CABAC) and HEVC-compressed streams. The authors tackled the main security challenge of SE, the limitation of the information leakage through protected video streams, and the improvement in the visual distortion induced by SE approaches.

The technology in [56] discussed was behind the new H.264/MPEG4-AVC standard, focusing on the main distinct features of its core coding technology and its first set of extensions, known as the fidelity range extensions (FRExt). In addition, this article also discusses the current status of adoption and deployment of the standard in various application areas.

The important differences are as follows:

- Enhanced motion-compensated prediction and spatial intra prediction capabilities
- Use of 4×4 and 8×8 (FRExt only) transforms in integer precision
- Use of a content-adaptive, in-loop, de-blocking filter
- Use of enhanced entropy coding methods

It was demonstrated that when used well together, the features of the new design provide significant bit rate savings for equivalent perceptual quality relative to the performance of prior standards. This is especially true for the High profile related coding tools.

3.5 High Efficiency Video Coding (HEVC)

High Efficiency Video Coding (HEVC) is the next generation video coding standard being developed, the newest video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group [57].

In [58], it was shown that HEVC provides significantly improved compression performance, i.e. an approximately 50% reduced bit rate as compared to the best existing video coding standards, under the same visual quality. The paper proposed a hardware-friendly method for RDO of HEVC intra coding. The results of the study showed that the proposed RD cost function provides 85.8% area reduction and 1260% throughput improvement in hardware design, with slight loss of bitrate and PSNR, which is suitable for real-time encoder application.

A performance comparison of H.265, VP9 and H.264 encoders was presented in [12]. According to the experimental results, obtained for a whole test set of video sequences by using similar encoding configurations for all three examined representative encoders, H.265/MPEG-HEVC was shown to provide significant average bit-rate savings of 43.3% and 39.3% relative to VP9 and H.264/MPEG-AVC, respectively.

In [59], it was shown that for resolutions of up to HD (1920x1080), code optimizations including heavy use of single instruction multiple-data (SIMD) instructions are sufficient to achieve HEVC real-time software decoding. It was further shown that, when it came to decoding UHD video (3840x2160), single threaded execution with code optimization was not enough.

According to [60], HEVC has been designed to focus on increasing video resolution and increasing the use of parallel processing architectures. Therefore, this approach merges all traditional configuration files used in the encoding process into only one configuration file without removing any parameters used in the traditional methods. The proposed approach in terms of encoding time as opposed to the traditional methods reduces the access time by half by reducing the data exchange between the configuration files used and without changing the rate-distortion (RD) performance or compression ratio. There is no change in the rate-distortion and compression ratio.

As per [61], the intra prediction part of the newest video compression standard H.265/HEVC was considered. That covers general HEVC dataflow. A series of experiments was conducted on different coding configurations and video sequences. The statistics presented using each intra prediction mode and the statistics of modes becoming part of the most probable mode array obtained in the experiments. The obtained statistic data are probably not the objective characteristics of the test video sequences, but only the illustration of the current intra coding practice. It is of interest to obtain objective statistical data which might contribute to the improvement of the existing approach.

An approach [62] compared the video coding standards by means of peak signal-to-noise ratio (PSNR) and subjective testing results. A joined approach is applied to the analysis of designs, including H.262/MPEG-2 Video, H.263, MPEG-4 Visual, H.264/MPEG-4 Advanced Video Coding (AVC), and High Efficiency Video Coding (HEVC). As per the results, HEVC encoders can achieve equivalent quality as encoders that conform to H.264/MPEG-4 AVC when using approximately 50% less bitrate on average. The HEVC design is shown to be especially effective for low bit rates, high-resolution video content, and low-delay communication applications.

High Efficiency Video Coding (HEVC) adopts 35 intra prediction modes with larger Coding Unit (CU) size to improve the intra encoding efficiency, causing high computational complexity. In [63], two fast intra-prediction algorithms are proposed to reduce the number of candidate modes for rate-distortion (RD) optimization. By improving the RMD and MPM process, the number of candidate modes to do SATD calculation and RDO calculation is reduced and computational complexity can be reduced. Experimental algorithms show it could save about 27.3% of encoding time with negligible performance loss compared to HM14.0.

The HEVC standard aims to provide a doubling in coding efficiency with respect to the H.264/AVC high profile, delivering the same video quality at half the bit rate. According to [64], complexity-related aspects that were considered in the standardization process are described. Furthermore, profiling of reference software and optimized software gives an indication of where HEVC may be more complex than its predecessors and where it may be simpler. Overall, the complexity of HEVC decoders does not appear to be significantly different from that of H.264/AVC decoders; this makes HEVC decoding in software very practical on current hardware. HEVC encoders are expected to be several times more complex than H.264/AVC encoders and will be a subject of research in years to come.

When it comes to transmitting high resolution video such as of resolution 4K, over the internet or in broadcast, the 50% bitrate reduction is essential. [65] Shows that real-time decoding of 4K video with a frame-level parallel decoding approach using four desktop CPU cores is feasible. It has been shown that real-time software decoding of 4K 50Hz video with HEVC is feasible on current desktop CPUs using four CPU cores. Encoding 4K video in real-time on the other hand remains a challenge. Therefore, first use cases of 4K video coded with HEVC are expected to be limited to offline encoded material for internet services like video on demand.

[66] presented a new parallelization approach for HEVC decoding named an Overlapped Wavefront (OWF). It is based on wavefront processing and improves its parallelization efficiency by allowing overlapped execution of consecutive pictures. In this strategy the de-coding, steps are performed on a CTB basis rather than on a picture basis which improves data locality. The implementation achieves between 29.6%, 42.4%, and 66.6% higher frame rates compared to previous results and

11.3%, 21.0%, and 38.0% higher frame rates compared to Tiles, for 2160p, 1600p, and 1080p, respectively.

Iterative intra prediction search in [67] was proposed for the H.265/HEVC encoder to reduce the number of prediction modes for estimation. There was about a 40% encoding time reduction for HM 10.1 intra-only coding with negligible bitrate increase and PSNR quality degradation. Additional speed-up techniques including fast prediction error estimation were offered.

[68] proposed and evaluated a parallelization strategy for the emerging HEVC video coding standard. The strategy is based on entropy slices which allow exploiting parallelism in the entropy decoding stage while maintaining high coding efficiency. The author approach requires encoding videos with one entropy slice per Largest Coding Units LCU row in order to decode multiple LCU rows in a wavefront parallel manner. Evaluations were performed on a PC with 12 Intel Xeon cores running at 3.3 GHz show that it is possible to achieve real-time performance for 1920×1080p50 (53.1 fps) and 2560×1600 (29.5fps) video resolutions with speedups of 5.2× and 6.3× compared to sequential execution, respectively.

In [43] HEVC HM software was compared for both the coding performance and the coding speed of practical HEVC encoders for high resolution video sequences. [69] conducted a comprehensive evaluation of the latest high performance H.265/MPEG-HEVC encoders, including the open source encoder—x265 and the commercial encoder—DivX265, based on default parameters and a new open 4K video database. Such comparison shows that the latest HEVC encoders, open source x265 and commercial DivX265, have achieved significant progress compared to the reference codec HM. However, there is further potential for optimization to address outstanding challenges.

In [70] a general review of new video compression standards HEVC, VP9 [71] and Daala [71] were conducted and their compression efficiency was compared. The experimental results in [70] showed that the Daala video encoder is still far from being competitive. While HM provides 31% better compression rates in key-frame only mode and about 40% improvement in inter-coding mode compared to JM, VP9 is only 18% better than JM in both modes. It is worth mentioning that the VP9

encoder does not have an efficient RDO model, so the VP9 encoder itself may potentially have better performance.

The performance evaluation metrics that are used to report HEVC efficiency results are mainly based on PSNR, especially for resolutions beyond HDTV. [72] provided subjective evaluation results to assess the performance of the current HEVC codec for resolutions beyond HDTV, comparing with the objective measured calculated in terms of PSNR.

The HEVC standard is the current state of the art video compression framework. The first version was published in 2013. Since then, several open source implementations have been developed, among others, the reference software model, the videoLAN encoder and the OpenHEVC decoder. HEVC codecs will replace H.264 ones inside the consumer electronic devices in the near future. In [73], the open source OpenHEVC decoder has been modified to support parallel decoding at the slice level using OpenMP instead of pthreads. The advantage of this unthreaded decoder is that it can be used with any architecture, providing it supports OpenMP. Tests have been carried out with three different multicore chips and the performance results are similar to those obtained with the threaded OpenHEVC decoder.

3.6 Use of machine learning in video coding

In Chapter-1 it was mentioned that the research presented in this thesis aims to use machine learning in the process of developing a multi-objective optimisation framework for H264 and HEVC video CODECs. In particular the use of Ensemble Learning Approaches is investigated. This section reviews past literature where machine learning was used for the purpose of enhancing or investigating the performance of video CODECs. Further literature that has investigated the capabilities of Ensemble Learning Algorithms as compared many popular single learning algorithms is also presented as further justification of the proposed research's intention to make effective use of Ensemble Learning Algorithms.

In [74] a non-traditional use of machine learning was proposed in the area of video encoding and transcoding. Video encoding and transcoding are computationally intensive processes and this complexity increases significantly with compression standards such as H.264. Video encoders and transcoders have to manage the quality vs. complexity tradeoff carefully. The author proposed the use of machine

learning in video coding and transcoding to overcome complexity related challenges in 1) MPEG-2 to H.264 video transcoding, 2) H.263 to VP6 transcoding, 3) H.264 encoding and 4) Distributed Video Coding (DVC). The results show that use of machine learning significantly reduces the complexity of encoders/transcoders and thus can enable efficient video encoding on resource constrained devices such as mobile devices and video sensors. The results also demonstrated that the proposed approach is general enough and can be used effectively in high complexity video coding and transcoding applications.

In [74] the use of machine learning to reduce the complexity of macro block mode computation from a search operation was proposed. The authors developed a methodology based on machine learning that computes the MB coding mode instead of searching for the best match thus reducing the complexity of Intra 16x16 coding by 17 times and Intra 4x4 MB coding by 12.5 times. The proposed approach uses simple mean value metrics at the block level to characterize the coding complexity of a macro block. The J4.8 classifier is used to build the decision trees. The results show that intra MB mode can be determined with over 90% accuracy. The approach can also be used for determining MB prediction modes with an accuracy varying between 70% and 80%.

According to [75], ensemble methods are learning algorithms that construct a set of classifiers and then classify new data points by taking a vote of their predictions. The concept of combining classifiers is proposed as a new direction for the improvement of the performance of individual classifiers. The original ensemble method is Bayesian averaging, but more recent algorithms include error-correcting output coding, bagging, and boosting. This paper reviews these methods and explains why ensembles can often perform better than any single classifier. Some previous studies comparing ensemble methods are reviewed, and some new experiments are presented.

The problem of combining classifiers which use different representations of the patterns to be classified was studied by [76], developing a common theoretical framework for classifier combination and showing that many existing schemes can be considered as special cases of compound classification where all the pattern representations are used jointly to make a decision. The two main reasons for combining classifiers are efficiency and accuracy.

In conducting the above review of literature it was found that the use of machine learning in enhancing the performance of video COCECs still remains in its early stages, regardless of the revolutionary changes machine learning has already resulted in video processing and quality enhancement supported by the recent developments ensemble learning systems and deep learning system.

3.7 End-to-End Video Streaming

The growth of the Internet has created a vast demand for multimedia communications. Video Services account for a very large portion of the network traffic. Transmission of real-time video typically has bandwidth, delay, and loss requirements. Therefore the subject of end-to-end video delivery and the associated performance analysis has become an area of significant research interest in the recent past.

In [77], the author presented results for evaluating MPEG-4 video quality in the presence of packet losses. The results show that a single packet loss in an I-frame can already result in a video impairment and significantly degrade the video quality. The impact of single packet loss was evaluated with different frequencies as well as loss distances within a short period. It was found that more than two times single-losses in a short period will lead the video quality to be unacceptable.

[78] used H.264 coded video over best-effort IP networks, using RTP as the real-time transport protocol. Novel, joint source and channel coding techniques were proposed. After a description of the environment, the error-resilience tools of H.264 and the draft specification of the RTP payload format were introduced. The video coding layer has been shown to significantly improve the performance of H.264 in the challenging best-effort IP environment.

The paper [79] focused on video services for 3G networks with provided traces of long MPEG-4 and H.263 encoded videos in the QCIF format resulting in low bandwidth video streams. The pre-encoded video sequences are encoded by different users and they differ in video settings in terms of codec, quality, format, and length. Compared to earlier traces, the new traces are suitable for the network performance evaluation of WLANs. The peak to mean ratios of the frame sizes of the new traces are typically in the range of 15 to 35, whereas a range from 7 to 18

was typically observed before. The traces were publicly available at [80] and provide instructions for using the traces in network evaluations.

Another study has attempted to address the problem of Multi-Objective Optimisation for Video Streaming. [81] Presented cross-layer optimized video rate adaptation and scheduling scheme for wireless video streaming over packetized networks aiming for maximum quality of service (QoS) for each user, maximum video throughput, and QoS fairness among users for wireless video streaming. The proposed framework aims to serve the user with the least remaining playback time, highest video quality and the highest video throughput.

A different approach was introduced in [82] using a pre-roll delay-distortion optimisation (DDO) framework while ensuring continuous playback for on-demand video streaming over limited bitrate networks using AVC/H.264 encoding. The input video is first divided into temporal segments. The system then encodes the input video according to the specified relevance-distortion policy, where encoding parameters are selected for each temporal segment. The optimal encoding parameters are computed using a novel, multi-objective optimisation formulation, using linear programming. What was accomplished in this paper is that the technique was developed to reduce the waiting time to much lower levels than downloading and playing; in addition, by maintaining the relevant quality at a suitable level over low bandwidth channels.

A novel and complete tool-set for evaluating the delivery quality of MPEG video transmissions in simulations of a network environment was presented in [83]. The proposed integration of EvalVid and NS2 provided a novel generalized and comprehensive tool-set for evaluating the video quality performance of network designs in a simulated environment. Therefore, for researchers to encode their own test video sequences in order to evaluate the delivered video quality in a simulated network environment, the proposed QoS assessment framework would be a good choice.

To meet these QoS requirements, researchers have developed a specific multimedia mechanism to enhance the performance of video transmission. Therefore, a new simulation tool-set called MyEvalvid_RTP to achieve more realistic simulations was proposed in [84]. With MyEvalvid_RTP, researchers can evaluate both the video delivered quality and the audio delivered quality.

The simulation model for H.264 video streaming was developed using OPNET Modeler, an advanced network modeling and simulation tool. The simulation results show that based on the high level characteristics in the time domain, a H.264 stream is very similar to an MPEG2 stream. Under stressed network conditions, the more advanced H.264 standard shows better results: lower queuing delays and less packet-loss as described in [85].

A Streaming Video Content Over IEEE 802.16 / WiMAX Broadband Access was studied in [86] to simulate bandwidth intensive, delay sensitive, video traffic representative of IPTV over WiMAX and ADSL. These video streams are encoded using MPEG-2 or MPEG-4 codecs. The simulation using the OPNET modeler with integrated WiMAX support has been adopted. As a result, ADSL showed behavior that approached the ideal values for the performance metrics, while WiMAX demonstrated promising behavior within the bounds of the defined metrics.

Analysing the wireless local area network's performance with streaming H.263 standard based video under mobility occurred in [87]. Various simulations were performed using H.263 video traffic. Study results verified the successful H.263 video traffic import in OPNET and recommendations included making video clients in WLAN somewhat more intelligent by adapting to the variations in network resources due to mobility.

The performance in [88] compared of WiMAX and ADSL by streaming audio and video contents. File Transfer Protocol (FTP), Hyper Text Transfer Protocol (HTTP), and electronic mail have also been used for comparison. The OPNET Modeler versions 15.0 and 16.0 were used to evaluate packet loss, delay, delay jitter, and throughput with various design parameters to determine whether WiMAX exhibits performance comparable to ADSL. The OPNET Modeler provided a suitable environment to design and characterize computer networks.

Real time video transmission is one of the biggest challenges of communication networks. The work in [89] deals with the transmission of video encoded with H.264/AVC coding standard through WLAN using an OPNET Modeller (OM). It is noted that the OM simulation environment allows for designing the transmission systems, which would be difficult to establish in laboratory conditions.

In [90] introduced a comprehensive toolset for streaming and evaluating video streams encoded using the H.265 standard including its scalable extension in simulated network environments. A toolset facilitates the transmission and

evaluation of HEVC/H.265 and SHVC encoded video on the popular open source NCTUns simulator.

The effect of Packet Drop and Jitter on perceived video quality for various encoded video, over Streaming Networks, using tools such as OPNET and EvalVid was investigated in [64]. The results helped with choosing an appropriate delay buffer size and packet repair techniques for various types of video, which will further help to improve the user experience in the field of multimedia as demonstrated in [91].

3.8 Summary & Conclusions

The subject of optimisation of the performance of video CODECs has been investigated both within and outside the standardization activities of all video CODECs in particular the well-established standard H264 and the latest standard HEVC. Whilst algorithmic optimisation has been the key focus of optimisation related research activity before and after the standardization of such CODECs, parameter based optimization approaches have received attention after the CODECs were standardised, in particular during their practical usage under resource constrained conditions.

Machine Learning has found widespread applications in particular in video processing and enhancement. However its use within video coding and representation has been minimal. The latest advances in machine learning such as Ensemble Learning and Deep Learning systems provides the means of utilising machine learning for enhancing the performance of video CODECs.

The literature review conducted in this chapter has shown that there has been no attempt to use machine learning for parameter based optimisation of video CODECs. Further there has been no attempt to propose approaches and/or frameworks for multi-objective optimisation of video CODECs. The research conducted within the context of this thesis and presented in chapters 5, 6, 7 and 8 contributes towards closing this research gap as briefly highlighted in Chapter 1.

The following chapter demonstrates the Machine Learning based Framework for the analysis of significant coding parameters of H.264.

Chapter 4

Parameter based Characterisation and Performance Modelling of a H.264 Video CODEC

4.1 Decoder Introduction

Applications that benefit from accurate video capture, efficient representation and coding, error-free transmission and subjectively optimised display, have been growing over the years due to the availability of higher network bandwidth, faster processor speed and advanced capture and display technologies. Some of the most extensively used applications include real-time video conferencing, video streaming over broadband networks and digital TV broadcasting. Most current mobile hand-held devices come equipped with a video camera that is able to capture and encode a video stream in a standard format.

These devices also include video players which can decode and playback video. Such video CODECs have many parameters that can be used to control their operational characteristics, both at the encoder and decoder ends, enabling the possibility of fine tuning their operation for maximum efficiency within environments and application scenarios that are bound by various constraints. For example, the available bandwidth will have an upper limit, the network will be subjected to delays and the decoder/display unit may have limitations in processing and display capabilities. Yet the encoder, transmission and decoder have many parameters that can be adjusted for them to be efficiently operational under above mentioned constraints. Identifying the values of these parameters that results in the CODECs optimal performance under given constraints remains an open research problem of vital importance.

The first step of parameter based optimisation of a video CODEC is the identification of the coding parameters that have a significant impact on its key performance related parameters, such as, bandwidth usage, image/video quality, and CPU time, etc. Although an experienced user of a video CODEC can guess these parameters with some accuracy when the content of the video is known, a formal scientific approach is needed to accurately decide the parameter set, with minimum subjective error.

In this chapter a framework that is based on linear regression is proposed for both the identification of significant coding parameters and performance modelling. In Chapter-5 it is demonstrated how these linear models can be used for multi-objective optimisation of a H.264 video CODEC using Genetic Algorithms. Together Chapters 4 & 5 provides a framework for the multi-objective optimisation of a H.264 video CODEC.

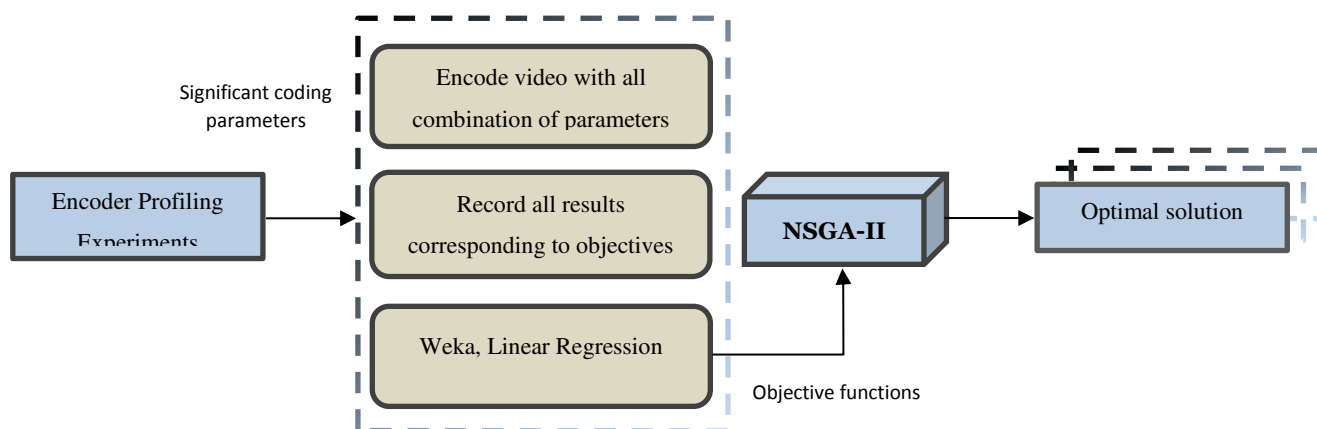
For clarity of presentation this chapter is divided several sections. Apart from this section which is an introduction to the research being carried out, section 4.2 presents an overview of the proposed framework, in particular the experiments conducted for the determination of the significant coding parameters and derivation of Linear Regression based models for the performance related measures. Section 4.3 presents a comprehensive analysis of the results of the characterisation and the performance modelling of the encoder. Decoder analysis is presented in section 4.4. Section 4.5 investigates the use of other, more sophisticated machine learning models such as Ensemble Learning Algorithms, e.g. Bagging, in the performance modelling of a H.264 video CODEC in order to justify the conclusion within this research to finally use of Linear Regression for performance modelling. Finally section 4.6 concludes providing a summary of the contributions made.

4.2 Proposed framework for multi-objective optimization

The primary focus of Chapter 4 and 5 is to propose a framework for Multi-Objective Optimisation of a H.264/AVC video CODEC, when working under multiple constraints. The MOO framework is intended to minimize the CPU time, CPU time is the total time it takes for the encoder and decoder to finish, Bit-rate and to

maximize the quality of the compressed video stream. MOO framework being proposed is accomplished by following the three steps below.

1. Profiling experiments on the encoder and decoder are carried out to determine the coding parameters that have a significant impact on each of the performance related objectives, namely, rate, and distortion and CPU Time. (Chapter-4)
2. The objective function/ model for each performance objective, subject to constraints, based on the above significant parameters is constructed, by using a suitable regression procedure. (chapter-4)
3. These objective functions were then used within a genetic algorithm (GA) based multi-objective optimisation framework to determine optimal parameter values. The focus of this chapter is the first two steps above, i.e. determining the significant coding parameters and establishing the corresponding objective functions. (Chapter-5)



A system level block diagram of the proposed framework is illustrated in Figure 4-1.

Figure 4-1: Proposed Multi-objective optimisation framework.

In a practical multimedia application scenario a device captures a video, encodes it and transmits it via a network to another device that decodes and displays the content to a viewer. Assuming that the network has bandwidth constraints and the device in which the encoder is placed has compute power constraints and the potential viewers of content may demand best quality levels, a situation in which the proposed MOO framework can be used, arises. The significant number of encoder parameters that control the encoders bit rate, quality and computational power

requirements can be selected, to ensure the encoder performance is optimal, under the given multiple constraints.

However, this requires the modelling of the encoders bit rate, quality and CPU time, based on the large number of selectable encoder parameters. If mathematical objective functions can be derived for each of the above, a standard approach to optimisation can be used (see Chapter-5). Deriving objective functions, for example using mathematical regression, will lead to the determination of the significant coding parameters, the key focus of the research presented below. The same explanation can be applied to the selection of decoder parameters that results in optimal decoder performance.

Within the research context of this chapter, the data transmission network is assumed to be perfect, i.e. no delays, no bit loses, no errors etc. Therefore, the bit stream generated by the encoder is transmitted without any loss or alteration to the decoder, real-time. The following section proposes the experimental process adopted to determine the significant coding parameters for both the encoder and decoder.

4.2.1 The Profiling Experiments - Determining the Significant Coding Parameters

This experiment was carried out using the configuration file published by Dolby Laboratories Inc. (2009). The H264 encoder software JM (Joint Model) reference software version 18.6 was used. In each profiling experiment each video sequence (only the use of a set of six popular test video sequences is presented in this chapter) was encoded using selected combinations of possible parameter values of initial set of encoder parameters. In other words, each encoding instance corresponds to a combination of coding parameter values selected from the possible exhaustive set that can be determined by varying each parameter within its entire range. For example instead of using quantization parameter variations between 1-51 (that is the exhaustive set), only two sample values, 17 and 49, were used (for further examples see bellow Table 4-3).



For each coding instance above, determined by the combination of the selected values of each coding parameter, the distortion (measured as PSNR), bit-rate and the number of CPU time were recorded. Subsequently the results were fed into the Linear Regression Analysis tool of WEKA [92] with the coding parameters as the


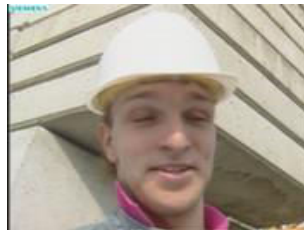
independent variables and rate, distortion and CPU time values (in separate experiments) as the dependent variables. A feature selection approach is used to remove the insignificant coding parameters, leaving the significant coding parameters as coefficients of the regression model. The resulting objective functions for the Bit-rate, distortion and CPU time are the final outcomes of the proposed research. Separate experiments are performed for each of the sample test videos.

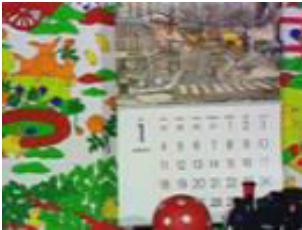

Table 4-1 shows the set of six test video sequences. *Claire* is a video sequence that has a foreground with simple motion and a non-moving area in background. In this video a news presenter is appears talking while moving her head, eyes and mouth, slowly. The *Coastguard* video sequence has fast movement on both foreground and background regions simultaneously. A boat appears to move fast in the foreground, with water showing waves and a second boat appearing after few frames, in the background. The *Football* video sequence has complicated fast motion, with a number of players moving very fast, simultaneously in the foreground. The *Foreman* video has minimal movement in the background. The Forman appears talking and his head appears to move quiet rapidly. The *Mobile* video sequence has fast background and foreground movement simultaneously. A calendar appears to be moving upward, a ball and a train is moving towards the left side of the scene with the camera showing signs of panning. Finally the *Tennis* table video sequence has a slow motion foreground and gentle movement in the background. The players hand is moving and the racket is bouncing a ball. The camera is slightly zooming out in the few last few frames. Given the above descriptions of the videos it is shown that they have different properties and features.

Note that all of the six video sequences are of QCIF resolution (176x144 pixels) and of 4:2:2 format. Note that low resolution videos are used in all experiments only to save simulation time. Without any restrictions the proposed framework can be used in relation to a video sequence of any resolution, in particular HD and full-HD. It is noted that the above, six selected video sequences have different properties of object motion, both in the foreground and background. Further differences exist in the scene content.

Table 4-1: Selected frames of video sequences [93], [94].

	
Claire 494 frames	Coastguard 300 frames

	
Football 260 frames	Foreman 300 frames

	
Mobile 300 frames	Tennis 150 frames

The CPU time was measured using the Intel VTune Amplifier XE. The experiment was performed on a HP computer, running Microsoft Windows 8.1 (64-bit), having an Intel Core i5 CPU 4200Y @ 1.40 GHz and 4.00GB RAM.

The encoder experiment can be outlined as follows:

1. The Six video sequences are encoded using a finite set of values for each encoding parameter and using all parameters defined in the configuration file. (See Appendix A.)
2. The values obtained for the three objectives were recorded for each sequence. PSNR, bitrates and CPU time.
3. Using WEKA feature selection algorithms, a parameter reduction process is carried out starting from the initial parameter used in [37] (see Table 4-2.) that resulting in four of the parameters originally used, being selected as significant coding parameters.

4. Only the Luminance Y component of the colour videos are used in the performance modelling since human eyes are more sensitive to luma components than to the chroma components. The PSNR SNR_Y is measured in decibels (dB)
5. The total encoding CPU time in seconds for each video sequence was recorded using Intel VTune Amplifier XE
6. Following are the parameters used, each video contains $3 \times 2 \times 2 \times 3 = 36$ Total Number of Instances as shown in Table 4-4. In each experiment one parameter will be changed while the rest parameters are fixed. This will help to observe the effect all parameters on each Objective. (See Appendix B.)
 - a. IntraPeriod
 - b. SearchRange
 - c. Quantization Parameter
 - d. NumberReferenceFrames
7. Then results shown in Table 4-4 are then fed in Weka software a machine learning tool to generate the linear regression function for each objective.

Table 4-2: The initial list of parameters used [37]

Parameter	Meaning
Resolution	Image width and height.
NumberReferenceFrames	Used for motion estimation.
Use FME	Fast motion estimation algorithms.
SearchRange	Sets allowable search range for motion estimation.
RDOptimisation	Enable rate distortion optimized mode decision.
SliceGroup	Number of slice group to be used.
IntraPeriod	Period of I-frames.
Quantization Parameter	Sets quantization parameter value.
DisableThresholding	Disable Thresholding of Transform Coefficients.

Table 4-3 tabulates the significant coding parameters selected. The significant parameters include Intra Period, Search Range, Quantization Parameter and Number of Reference Frames.

The Table 4-3 also tabulates the sample values used in our experiments for each parameter from within their corresponding value ranges. Note that the four parameters are given variable names $x(1)$, $x(2)$, $x(3)$ and $x(4)$. The control variable

Intra Period - IP ($x(1)$), can take values: 0 (means that the first frame is coded as an I-frame and subsequent frames are coded as P-frames), 5 and 8. The Search Range - SR ($x(2)$) is assumed to take either of the two values 16 or 32. The control variable Quantization Parameter - QP ($x(3)$) is assumed to take two possible values 17 or 49. The Number of Reference Frames – NRF ($x(4)$) can take values 2, 5 and 8.

Table 4-3: Significant parameters and value used

Variables	Parameters	Values Range	Variable Type
IP= $x(1)$	Intra-Period	(0,5,8)	Numeric
SR= $x(2)$	Search-Range	(16, 32)	Numeric
QP= $x(3)$	Quantization Parameter	(17,49)	Numeric
NRF= $x(4)$	Number-Reference-Frames	(2,5,8)	Numeric

Table 4-4 presents 36 data instances of the Foreman video sequence that were used in the final stage of modelling the PSNR, image quality and CPU time. These are the inputs to the linear regression based modelling process that will result in the three objective functions. PSNR is measured in decibels (db), Bit-rate in (kbit/s) and CPU time in seconds (sec).

Table 4-4: selected set of parameters for foreman sequence

IP	SR	QP	NRF	PSNR	Bit-rate	CPU time
$x(1)$	$x(2)$	$x(3)$	$x(4)$	in (db)	in (kbit/s)	in (sec)
0	16	17	2	44.606	547.62	61.627
0	16	17	5	44.635	473.74	75.058
0	16	17	8	44.636	471.7	84.624
0	16	49	2	22.806	9.63	41.619
0	16	49	5	23.384	9.09	57.455
0	16	49	8	23.382	9.28	70.711
0	32	17	2	44.62	547.1	61.189
0	32	17	5	44.688	472.77	74.527
0	32	17	8	44.691	471.59	83.665
0	32	49	2	22.871	9.98	41.24
0	32	49	5	23.449	9.21	57.3
0	32	49	8	23.405	9.38	71.303
5	16	17	2	45.751	819.97	55.727
5	16	17	5	45.819	712.66	69.253
5	16	17	8	45.822	714.2	75.729
5	16	49	2	23.53	16.77	37.336
5	16	49	5	23.881	16.44	53.122
5	16	49	8	23.881	16.51	63.869
5	32	17	2	45.764	820.33	69.276
5	32	17	5	45.83	714.45	80.19
5	32	17	8	45.825	714.3	86.417
5	32	49	2	23.531	16.78	37.594
5	32	49	5	23.881	16.44	51.472
5	32	49	8	23.881	16.5	64.64
8	16	17	2	44.818	618.74	57.825
8	16	17	5	44.906	544.33	69.269
8	16	17	8	44.899	540.87	77.812
8	16	49	2	22.955	13.09	39.142
8	16	49	5	23.532	12.67	52.936
8	16	49	8	23.54	12.73	65.502
8	32	17	2	44.821	618.66	58.211
8	32	17	5	44.91	544.98	70.837
8	32	17	8	44.902	542.31	78.66
8	32	49	2	22.958	13.12	39.35
8	32	49	5	23.532	12.67	53.45
8	32	49	8	23.54	12.73	66.653

4.2.2 The Objective Functions of the H.264/AVC Encoder

Based on the output of the linear regression algorithms applied as explained above, the objective functions for distortion (PSNR), Bitrate and CPU time for each video sequence are found in (Equation 4-1 - 4-18) for each sequence video. These functions provide one the means to discuss in detail the significance of each parameter and how they affect the PSNR, rate and CPU. The following section provides an analysis of the experimental results. In particular, the analysis considers the test videos separately and discusses the impact of each coding parameter given the known properties of the contents of each video.

Following are the obtained models for each video sequence, with $f(1)$ representing PSNR, $f(2)$ Bit-rate and $f(3)$ CPU time. It was shown above that these so-called dependent parameters depend basically on four independent parameters namely $x(1)$, $x(2)$, $x(3)$ and $x(4)$, which are respectively the significant coding parameters Intra Period, Search Range, Quantization Parameter and the Number of Reference Frames. Ideally in an implementation of a CODEC, especially when Rate-Control has been disabled (see Chapter 3), there should not be any dependence of $f(1)$ or $f(2)$ or vice-versa. Unfortunately due to the specific implementation adopted of the H.264 CODECs Encoder in JM reference software [95], our preliminary investigations suggested that there is likely to be a slight dependency between the two performance measures. Therefore in the modelling that was carried out we introduce two further parameters, i.e. $x(5)$ representing PSNR as a parameter and $x(6)$ presenting Bit-rate as a parameter. In the modelling of $f(1)$ and $f(2)$ we therefore use the four significant parameters $x(1)$ - $x(4)$ and a further parameter, $x(5)$ PSNR and $x(6)$ Bitrate.

Claire Linear Regression Model

$$f(1)_{PSNR} = 0.0208 * x(1) - 0.6072 * x(3) + 0.0322 * x(4) + 0.0061 * x(6) + 56.7263$$

(Equation 4-1)

$$f(2)_{Bitrate} = 4.7979 * x(1) - 6.0443 * x(3) + 285.5205$$

(Equation 4-2)

$$f(3)_{Enc_Time} = -0.5329 * x(1) + 0.1518 * x(2) - 0.1223 * x(3) + 3.4701 * x(4) + 33.658$$

(Equation 4-3)

Coastguard Linear Regression Model

$$f(1)_{PSNR} = -0.0805 * x(3) + 0.2737 * x(4) + 0.0174 * x(6) + 26.9452$$

(Equation 4-4)

$$f(2)_{Bitrate} = -3.8562 * x(3) - 15.4109 * x(4) + 41.7428 * x(5) - 748.8825$$

(Equation 4-5)

$$f(3)_{Enc_Time} = -0.7994 * x(1) - 0.3255 * x(2) - 0.7908 * x(3) + 4.6741 * x(4) + 75.6884$$

(Equation 4-6)

Football Linear Regression Model

$$f(1)_{PSNR} = -0.4367 * x(3) + 0.0802 * x(4) + 0.0047 * x(6) + 41.066$$

(Equation 4-7)

$$f(2)_{Bitrate} = -68.407 * x(3) - 15.2422 * x(4) + 3452.3615$$

(Equation 4-8)

$$f(3)_{Enc_Time} = -0.7234 * x(1) - 1.3867 * x(3) + 5.2292 * x(4) + 106.0023$$

(Equation 4-9)

Foreman Linear Regression Model

$$f(1)_{PSNR} = -0.5926 * x(3) + 0.0798 * x(4) + 0.0046 * x(6) + 52.0225$$

(Equation 4-10)

$$f(2)_{Bitrate} = 6.8133 * x(1) - 18.5023 * x(3) + 890.0319$$

(Equation 4-11)

$$f(3)_{Enc_Time} = -0.5347 * x(1) + 0.1297 * x(2) - 0.5646 * x(3) + 4.0201 * x(4) + 60.362$$

(Equation 4-12)

Mobile Linear Regression Model

$$f(1)_{PSNR} = -0.0249 * x(1) - 0.6193 * x(3) + 0.0713 * x(4) + 0.0031 * x(6) + 49.5504$$

(Equation 4-13)

$$f(2)_{Bitrate} = 16.1344 * x(1) - 52.0468 * x(3) - 18.1129 * x(4) + 2602.1383$$

(Equation 4-14)

$$f(3)_{Enc_Time} = -0.616 * x(3) + 4.7227 * x(4) + 69.4487$$

(Equation 4-15)

Tennis Linear Regression Model

$$f(1)_{PSNR} = -0.4868 * x(3) + 0.0795 * x(4) + 0.0032 * x(6) + 47.9843$$

(Equation 4-16)

$$f(2)_{Bitrate} = -39.7315 * x(3) - 20.461 * x(4) + 2061.2244$$

(Equation 4-17)

$$f(3)_{Enc_Time} = -0.7551 * x(1) - 0.9888 * x(3) + 4.323 * x(4) + 78.527$$

(Equation 4-18)

4.3 Encoder Performance Analysis

Experimental analysis was conducted separately for the encoder and decoder. The Encoder objective functions obtained as a result of the experimental procedure presented in (Equation 4-1 - 4-18) enables one to discuss the significance of each of the coding parameters.

Table 4-5 tabulates the correlation coefficients of the objective functions. They range between 0-1. A value closer to 1 represents the fact that the dependant variable (in this case Bit-Rate, PSNR or CPU time) can be predicted very accurately from the coding parameters that play a role and has thus been included within the objective functions. In analysing the objective functions above, higher positive coefficients of coding parameters indicate higher positive dependency and higher negative coefficients represent higher negative dependency. If a certain parameter is not present in the objective function that means that the objective is independent of that parameter.

Table 4-5: Encoder Correlation Coefficient.

Video	PSNR In (db)	Bit- rate	CPU time in (sec)
Claire	1	0.9424	0.9460
Coastguard	0.9865	0.9865	0.8917
Football	0.9997	0.9967	0.9849
Foreman	0.9998	0.9678	0.9746
Mobile	0.9999	0.9809	0.9588
Tennis	0.9998	0.9757	0.9883

A careful analysis of the coding parameters that have non-zero weighting factors in the objective functions obtained and a comparison of relative magnitudes of the coefficients can lead to a direct correspondence with the properties of the video, for e.g., the presence of motion in foreground and background, the speed of movement of objects, sudden scene changes, camera pan/tilt/zoom effects and the general characteristics of the content of the video as well. For example, the analysis of the linear regression equations obtained for Foreman video sequence identifies all four parameters to have significant impact on CPU time, namely:

- IntraPeriod
- Searchrange
- Quantization parameter
- NumberReferenceFrames

For the same video the following parameters were identified to have a significant impact on Bit-rate.

- IntraPeriod
- Quantization

The parameters that are identified to have a significant impact on PSNR are:

- Quantization parameter
- NumberReferenceFrames

A more detailed and video sequence specific analysis can be presented as follows.

1. CPU Time Analysis Experiment:

The objective functions obtained for all six video sequences for CPU time indicates that the parameter that has the most significant impact on CPU time is the number of reference frames. This is expected due to the need to repeat the motion estimation process when NRF increases.

It has a significant impact in increasing the CPU time. This is expected given the computational cost of the motion estimation algorithm implemented within H.264. The next significant impact is from the Quantization parameter. The impact from search range (SR) and Intra Period (IP) is relatively insignificant. For most videos with fast movement of objects (i.e. Football and Mobile) there is no impact from the Search Range. This is true given the fact that for videos with fast moving objects, best matches will not be found quickly, i.e. without having to scan the entire video. All objective functions include a similar constant term indicating that a fixed computational cost for encoding is present, which is independent of the selection of coding parameters. This is expected given the processes that exist, which are independent of the coding parameters.

- 2. Bit-Rate Analysis Experiment:** The parameter that has the most significant impact in the bit-rate is distortion or the PSNR value. This is expected given the fact that the rate-distortion optimisation happens in a combined manner. The number of reference frames (NRF) has a negative correlation to the bit rate and is significant. The negative correlation is due to the fact that when the number of reference frames increases the chances of finding a better match in motion estimation increases, thus reducing the bit rate. The significance of it is due to the same reason. As all video sequences have various amounts of motion between frames, its bit rate will have an impact on the quantization parameter. The quantization parameter (QP) has a very important impact on the compression rate of H.264.

It is noted that in the modelling of Bit-Rate PSNR was used as a fifth independent parameter, $x(5)$. However the modelling process has dropped $x(5)$ indicating that PSNR has no direct impact on the target Bit-Rate.

3. PSNR Analysis Experiment: The parameter that has the most significant impact on PSNR is bit-rate. It is noted that these two parameters are highly dependent. The number of reference frames and quantization parameter also has an impact on the PSNR for all video sequences. The PSNR results tabulated in Table 4-5 indicate that the two videos with the least amount of movement/changes, namely Mobile and Claire have the best correlation coefficients. This is expected due to the stability of the CODEC during the encoding of the individual frames of the coded sequence.

It is noted that target Bit-Rate used as parameter $x(6)$ has a slight direct impact on the PSNR obtained. This is indicated by the presence of $x(6)$ in the models/equations (see equation-1) of the Bit-Rate with a very low valued coefficient as compared to the coefficients of the other four significant coding parameters. This demonstrate the issue raised by our preliminary investigations about the specific implementation we have adopted in our experiments. However in a proper implementation of a CODEC this should not be the case (see Chapter-6 for the H.265 implementation we have used.).

4.4 Decoder Performance Analysis

To analyse the performance of the decoder, the encoded video sequences should be decoded using different combinations of decoder parameters. The process involved can be outlined as follows:

1. The Six encoded video sequences were decoded using a configuration file containing input parameters to the JVT H.264/AVC decoder as shown in Appendix A.
2. For each encoded video, the total decoding CPU time is recorded using Intel VTune Amplifier XE.
3. The decoding is conducted under 36 (=3x2x2x3) different value combinations of the decoder significant parameters, listed below:
 - a. IntraPeriod
 - b. SearchRange
 - c. Quantization Parameter
 - d. NumberReferenceFrames

4. The data instances thus gathered is sampled in Table 4-6. The full data gathered is then fed into WEKA where a machine learning tool generates the linear regression function for each objective. The analysis of the decoder is limited to decoder parameters that have significant effect on only the decoder's CPU time.

4.4.1 The output of Decoded video

A H.264 decoder takes an encoded .264 file as input and outputs a raw YUV video stream. As an example of coastguard video

InputFile = "test.264" is the H.264/AVC coded bitstream file

OutputFile = "test_dec.yuv" is the Output file, YUV/RGB

The output in Figure 4-2 shows the output video artifact of frame 30 with quantization parameter (QP) of 49 that gives very low quality with PSNR of 24.189 db and 5.83 Bitrate compared to QP 17 that has 43.418 db and 979.02 Bitrate.

It is noted that the visual quality of the video reduces when quantization parameter increased.

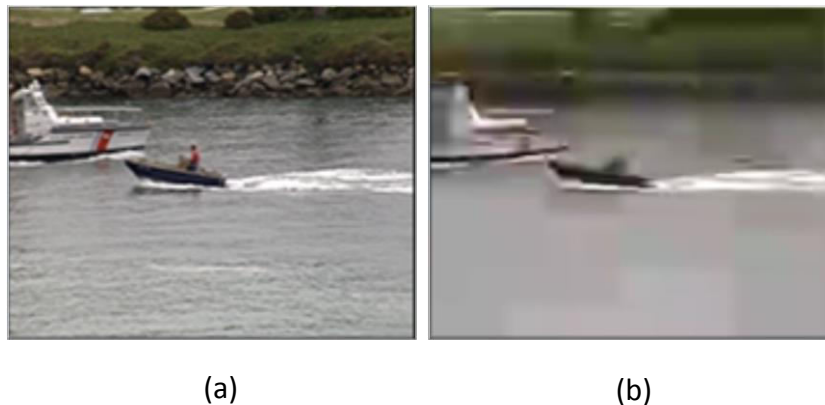


Figure 4-2: Sample image of frame 30 at (a) QP= 17 and (b) at QP= 49

Note that the Decoder parameters have no impact on Bit-rate and PSNR as these are determined by the encoder. In the proposed framework the quality and the bit-rate received by the decoder are the same as the encoder output.

The CPU time of the decoder is analysed using the same method used at the encoder end. For the six given video sequences, experiments were performed in order to find

out those coding parameters that can significantly influence CPU time. The objective functions thus obtained are listed with in (Equation 4-19 - 4-24).

Claire Linear Regression Model

$$f(1)_{Dec_Time} = 0.004 * x(1) + 0.0008 * x(2) - 0.0039 * x(3) + 0.373$$

(Equation 4-19)

Coastguard Linear Regression Model

$$f(1)_{Dec_Time} = -0.0042 * x(2) - 0.0118 * x(3) + 0.8945$$

(Equation 4-20)

Football Linear Regression Model

$$f(1)_{Dec_Time} = 0.0016 * x(1) - 0.022 * x(3) - 0.0033 * x(4) + 1.3455$$

(Equation 4-21)

Foreman Linear Regression Model

$$f(1)_{Dec_Time} = -0.0146 * x(3) + 0.9353$$

(Equation 4-22)

Mobile Linear Regression Model

$$f(1)_{Dec_Time} = 0.0071 * x(1) - 0.022 * x(3) - 0.0059 * x(4) + 1.3396$$

(Equation 4-23)

Tennis Linear Regression Model

$$f(1)_{Dec_Time} = -0.0158 * x(3) - 0.0072 * x(4) + 1.0304$$

(Equation 4-24)

Table 4-6: Selected set of decoder parameters for Foreman Sequences.

IP $x(1)$	SR $x(2)$	QP $x(3)$	NRF $x(4)$	Decoder Time in (sec)
0	16	17	2	0.659
0	16	17	5	0.659
0	16	17	8	0.639
0	16	49	2	0.209
0	16	49	5	0.205
0	16	49	8	0.203
0	32	17	2	0.659
0	32	17	5	0.635
0	32	17	8	0.628
0	32	49	2	0.223
0	32	49	5	0.207
0	32	49	8	0.205
5	16	17	2	0.741
5	16	17	5	0.721
5	16	17	8	0.729
5	16	49	2	0.243
5	16	49	5	0.237
5	16	49	8	0.227
5	32	17	2	0.804
5	32	17	5	0.857
5	32	17	8	0.792
5	32	49	2	0.236
5	32	49	5	0.227
5	32	49	8	0.229
8	16	17	2	0.656
8	16	17	5	0.646
8	16	17	8	0.622
8	16	49	2	0.221
8	16	49	5	0.213
8	16	49	8	0.214
8	32	17	2	0.658
8	32	17	5	0.637
8	32	17	8	0.626
8	32	49	2	0.232
8	32	49	5	0.213
8	32	49	8	0.214

Table 4-7 tabulates the correlation coefficients of the objective functions. The Football video sequence has the highest correlation coefficient closely followed by mobile. The analysis of the linear regression equations obtained to identify parameters that have significant impact on CPU time (Equation 4-24) reveals that the quantization parameter has the most significant impact. QP has an impact in all the video sequences as evidenced by its presence in all objective functions and being the parameter having the highest magnitude coefficient.

Table 4-7: Decoder correlation coefficient

Video Sequences	CPU time in (sec)
Claire	0.9593
Coastguard	0.9217
Football	0.9984
Foreman	0.9786
Mobile	0.9958
Tennis	0.9873

The Encoder and Decoder analysis indicates that the objective functions obtained as a result of using the proposed framework is able to accurately define the significant coding parameters and further detail the level of significance of each parameter. They can also be related to the motion and content information of the videos.

It is noted that our preliminary investigations revealed that the specific implementation of the H.264 CODEC that was used in the above modelling, did not suffer from the dependence of $f(1)$ and $f(2)$ on $x(6)$ and $x(5)$ respectively. Therefore parameters $x(5)$ and $x(6)$ were not considered during the Decoder performance modelling above.

4.5 Using Advanced Machine Learning

Algorithms for the Modelling of an H264

CODEC

In sections 4.3 and 4.4 the use of Linear Regression in performance modelling of the encoder and decoder were presented. However Linear Regression is a simple approach that may not be the most effective approach/method to model the performance of a CODEC.

The recent advances of machine learning algorithms, especially tree based algorithms, ensemble learning algorithms etc., provides further possibilities to be considered beyond using purely mathematical models. Given this observation, in this section we carry our experiments to model the performance of a H264 CODEC using advanced machine learning algorithms. The aim is to compare the accuracy of performance modelling obtainable via such approaches to the accuracy of performance modelling already shown above via the use of Linear Regression based modelling.

WEKA is an ideal platform for investigating the use of different machine learning algorithms as it consists of implementations of a large number of data-pre-processing, classification, modelling and clustering algorithms. It provides a graphical user interface for exploring and experimenting with machine learning algorithms on datasets.

The key focus of the experiments conducted in this section is to model the CPU utilization via measuring the encoding/decoding times, when the encoder/decoder parameters are respectively set to different combinations of possible values, within specified practical constraints. After collecting the encoding/decoding times under changes to parameter values, different machine learning algorithms can be used to model the recorded times. The correlation coefficient between the actual and predicted times can be used to determine accuracy of prediction that each model can provide. Details of the experiments conducted and an analysis of the results is presented in the following sub-sections.

4.5.1 Experiments, results and analysis

Bagging (Bootstrap Aggregating) is an ensemble method that creates separate samples of the training dataset and creates a classifier for each sample with the aim of reducing variance. Bagging can be used to perform both classification and regression depending on the base learner selected. Bagging has been demonstrated to be the most effective ensemble learning approach and hence the one used within the proposed research context.

In the experiments conducted the following base classifiers were used with Bagging:

- REPTree
- RandomForest
- AdditiveRegression
- RandomSubSpace

The base classifiers are combined with bagging to achieve very high classification accuracy / modelling accuracy.

Table 4-8: Decoder Correlation coefficient using bagging

CPU Time Decoder Correlation coefficient					
Videos	Linear-Regression	REP-Tree	Random-Forest	Additive-Regression	Random-SubSpace
Claire	0.9593	0.9877	0.9961	0.9818	0.9773
Coastguard	0.9217	0.9554	0.9806	0.9302	0.9476
Football	0.9984	0.9992	0.9991	0.9996	0.9987
Foreman	0.9786	0.9966	0.996	0.9906	0.9907
Mobile	0.9958	0.996	0.998	0.9958	0.9959
Tennis	0.9873	0.997	0.9979	0.994	0.9957

Bagging produces a combined model that often performs significantly better than the single model built from the original training data, and is never substantially worse. Best performance is achieved by combining both, different feature sets and different classifiers.

Table 4-8 tabulates the Decoder correlation coefficients when using Linear Regression and Bagging (with four different base classifiers) in the modelling of decoder CPU time. The results clearly demonstrate the ability of bagging to improve

the modelling accuracy, given the improved values of correlation coefficients obtained.

The performance modelling of the encoder was also carried out using the same Bagging based classifiers and compared with the accuracy of the Linear Regression method proposed. Results show marginal improvement in most cases over the modelling accuracy results obtained with Linear Regression in Table 4-9.

It is noted that for the Encoder modelling is performed for not only CPU time but also PSNR and Bit-Rate as illustrated in Table 4-9.

Table 4-9: Encoder Correlation coefficient with both linear and bagging

Video	Objectives	Encoder Correlation coefficient				
		Linear Regressi on	REP- Tree	Random- Forest	Additive- Regression	Random- SubSpace
Claire	PSNR	1	0.9999	0.9997	1	0.9995
	Bit-Rate	0.9424	0.9969	0.999	0.9977	0.9972
	CPU time	0.946	0.9398	0.9821	0.9699	0.9771
Coastguard	PSNR	0.9865	0.9985	0.9966	0.9989	0.9983
	Bit-Rate	0.9865	0.9939	0.9984	0.9969	0.9905
	CPU time	0.8917	0.9344	0.9708	0.9307	0.9411
Football	PSNR	0.9997	0.9998	0.9997	1	0.9997
	Bit-Rate	0.9967	0.9994	0.9997	0.9996	0.9991
	CPU time	0.9849	0.9793	0.9934	0.9885	0.9828
Foreman	PSNR	0.9998	0.9998	0.9996	0.9999	0.9994
	Bit-Rate	0.9678	0.9971	0.9991	0.9981	0.9962
	CPU time	0.9746	0.9671	0.9899	0.975	0.9752
Mobile	PSNR	0.9999	0.9999	0.9998	1	0.9995
	Bit-Rate	0.9809	0.9984	0.9996	0.9993	0.9982
	CPU time	0.9588	0.9673	0.9884	0.9763	0.9749
Tennis	PSNR	0.9998	0.9999	0.9998	1	0.9993
	Bit-Rate	0.9757	0.996	0.9988	0.9988	0.9947
	CPU time	0.9883	0.9782	0.9942	0.9918	0.9927

The encoder/decoder analysis above has shown that bagging produces more accurate models. Unfortunately, the bagging technique and its classifiers do not generate a mathematical formulation, i.e. an objective function for each objective is not produced when using bagging procedures. Therefore this stops us using Bagging

as a means to model performance of the CODEC in the optimisation work presented in Chapter-5. Therefore, in the CODEC optimisation work presented in chapter-5 the Linear Regression models developed above will be used. However the work in this section has demonstrated that better models exists for the modelling of performance of a video CODEC. Though the decision is to use the Linear Regression model, Linear regression implements a statistical model that, when relationships between the independent variables and the dependent variable are almost linear, shows optimal results, therefore the use of Linear Regression is justified based on its simplicity and easy to use.

4.6 Summary & Conclusion

This chapter proposed a machine learning based approach for the determination of significant coding parameters of a H264 video CODEC. In particular, the experiments conducted proposed the use of multivariate regression analysis in modelling the performance of a CODEC via defining objective functions for CPU time (both for the encoder and decoder), PSNR and the bit-rate (for the encoder only) of a video CODEC when a given video is being encoded/decoded. The analysis conducted used known information about the content and the motion present in the test videos to justify the formation and the nature of the objective functions obtained. These objective functions will be used in Chapter-5 for multi-objective optimisation of a video CODEC based on Genetic Algorithms.

The chapter also investigated the potential use of Ensemble Learning algorithms in the modelling of a H264 video CODEC. The results concluded that the Ensemble Learning algorithm Bagging, when used with a suitable single base classifier improves modelling accuracy beyond what can be achieved by a Linear Regression model. However the difference in accuracy is marginal that justifies the decision to use Linear Regression as the best practical solution.

Chapter 5

Multiobjective Optimisation

5.1 Introduction

The study in Chapter 4 has investigated the parameters that have a significant impact on the encoder and decoder performance in terms of CPU time. The study further resulted in a number of Linear Regression models being developed that can effectively be used to accurately model the CODEC's performance related properties such as PSNR, Bit Rate and CPU time.

This chapter presents a framework for multi-objective optimisation of video CODECs. Specifically, an optimization scheme is proposed to determine the optimum coding parameters for a H.264 AVC video codec in a bandwidth constrained environment, which minimises codec time and video distortion. In the literature, the contributions to the optimisation of H.264/AVC have focuses on the CPU time, power consumption, rate distortion and delay. A considerable research gap exists in developing a generalized framework for a parameter based approach for optimizing end-to-end video delivery system that is capable of working under multiple objectives/constraints. This framework is very useful within present and future video delivery systems and video coding.

The encoding/decoding parameters that have a significant impact on the performance of the codec are initially obtained through experimental analysis as explained in Chapter 4. A mathematical formulation by means of linear regression is subsequently used to associate these parameters with the relevant objectives and a Multi-Objective Optimization [43] problem is defined. Solutions to the optimization problem are reached through a Non-dominated Sorting Genetic Algorithm (NSGA-II). NSGA-II is implemented in the genetic algorithm gamultobj, available in the MATLAB optimisation tool-box. It is shown that the proposed framework is flexible on the number of objectives that can jointly be optimized. Practical use of the proposed framework is described using the six videos mathematical formulation.

For clarity of presentation this chapter is divided into 5 sections. Apart from this section which is an introduction to the research problem, section 5.2 presents the setting of Genetic Algorithm using Matlab Section 5.3 presents optimising the encoder. Optimising the Decoder is described in Section 5.4. Finally section 5.5 concludes with a summary of the chapter.

5.2 Setting up the Genetic Algorithm

The proposed optimisation framework was implemented on a HP computer, running Microsoft Windows 8.1 (64-bit), having an Intel Core i5 CPU 4200Y @ 1.40 GHz and 4.00GB RAM.

The optimization toolbox in MATLAB is used for the implementation of multi-objective optimization using a Genetic Algorithm. The MATLAB based solver used is ‘gamultiobj’ and the settings are fixed as shown in Figure 5-1. MATLAB’s ‘gamultiobj’ solver attempts to create a set of Pareto optima for a multiobjective minimization. ‘Gamultiobj’ uses the genetic algorithm for finding local Pareto optima. As in the GA function, one may specify an initial population, or have the solver generate one automatically. The algorithm is first initialised by defining the population size, the total number of generations, the number of variables etc. as presented in Table 5-1.

Table 5-1: 'gamultobj' settings used

gamultobj settings

Fitness function: function that is to be minimized
 Number of variables: 6
 Population Type: Double Vector
 Creation Function: feasible population
 Population Size: 300
 Initial Population: Default
 Initial Scores: Default
 Selection Function: Tournament
 Tournament size: 5
 Crossover Fraction: 0.25
 Mutation Function: adaptive feasible
 Crossover Function: Intermediate
 Crossover Ratio: 0.23
 Migration Direction: both
 Migration Fraction: Default (0.2)
 Migration Interval: Default (20)
 Distance Measure Function: Default @distancecrowding
 Pareto Front Population Fraction: .7
 Hybrid Function: None
 Maximum Generations: 300
 Time Limit: Default (Infinite)
 Fitness Limit: Default (Infinite)
 Stall Generations: Default (100)
 Function Tolerance: 1e-4
 Constraint tolerance: 1e-3
 Plot functions: Pareto front.

The optimization is executed by clicking Start listed under 'Run solver and view results' sub-frame (see Figure 5-1). A plot appears in a figure window. This plot shows the trade-off between the two components off. It is plotted in objective function space.

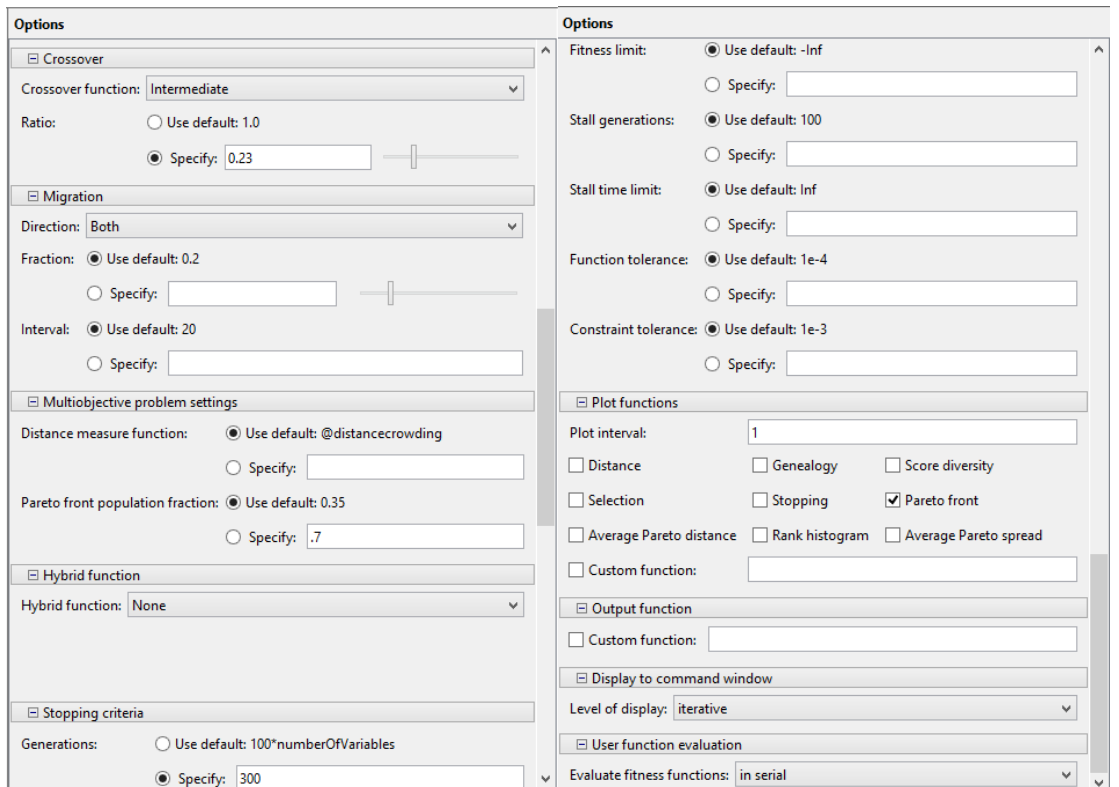
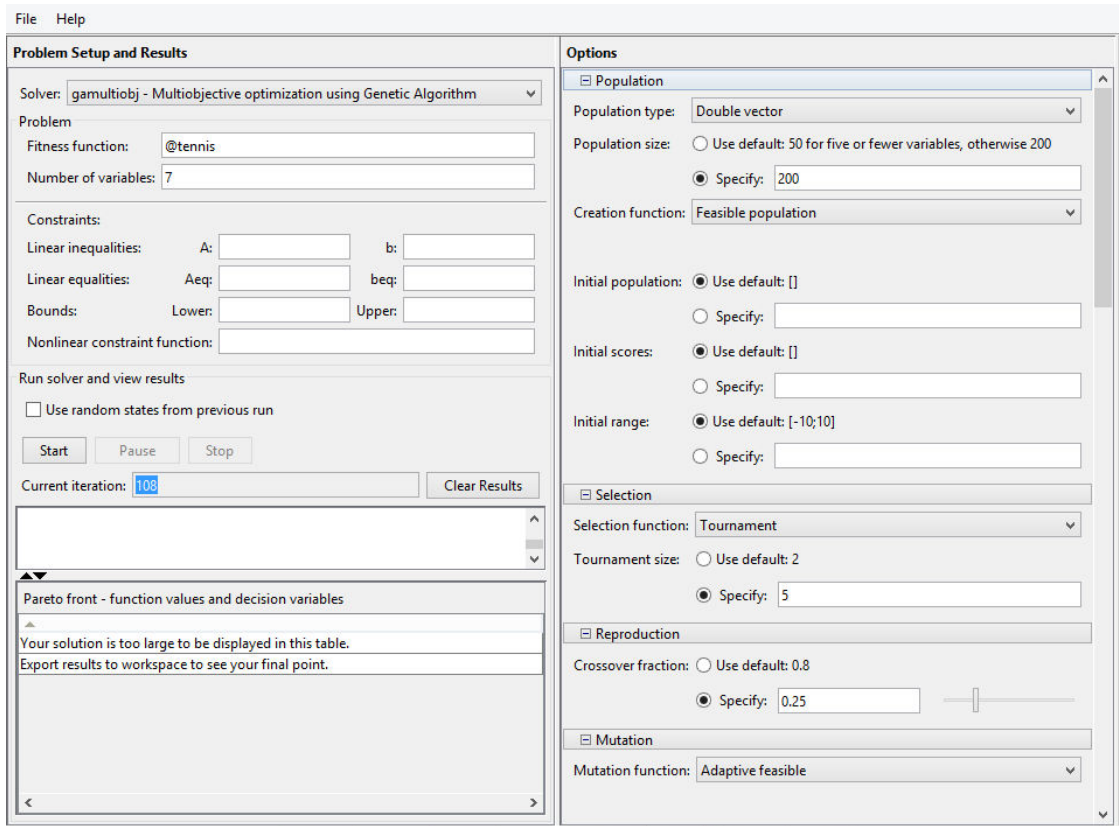


Figure 5-1: Setting options for the optimisation task

5.3 Optimising the Encoder

In this study, the objective functions to be optimized, given in Chapter 4 were used to optimise the encoder. Three functions are associated with each video namely the functions for the PSNR, bit rate and CPU time. These functions are then fed to the NSGA-II optimization tool along with the fitness function and number of variables. The NSGA-II provides all sets of optimal results that jointly minimize time, bit-rate and maximizes quality. Since a single 3D graph is complex to visualize the optimality of the results, pairs of graphs were plotted.

An optimization problem is one requiring the determination of the optimal (maximum or minimum) value of a given function, called the objective or fitness function, subject to certain defined restrictions, or constraints placed on the variables concerned.

It is noted that the MATLAB Optimization Toolbox's optimization functions minimize the objective or fitness function. That is, they solve problems of the form.

$$\min f_x(x).$$

If one requires to maximize $f(x)$, $-f(x)$ should be minimised, as the point at which the minimum of $-f(x)$ occurs is the same as the point at which the maximum of $f(x)$ occurs. In other words, to achieve optimum performance, the function is maximized by minimizing the negative of the function. In encoder (Equation 4-1 -4.18) the PSNR, i.e. the Quality of the video is to be maximised. So that maximized equation is minimized by multiplying the PSNR equation by a negative sign, i.e. creating $-f(x)$.

The objective functions depend generally on four parameters expressed as x in the MOO problem formulation, which include IntraPeriod, SearchRange, Quantization Parameter and NumberReferenceFrames. It was mentioned in Chapter-4 that due to a specific implantation issue with regards to the H.264 implementation used in the experiments there is a direct dependency of the target Bit-Rate (thus represented as a parameter $x(6)$) on PSNR. Hence in the objective function of PSNR, a fifth parameter, $x(6)$ exists. Each pair consists of two objectives functions.

In optimising the encoder two-objective, multiple constraint problems were considered. Three sets of two-objective (and multiple-constraint) optimisations were carried out, namely:

- PSNR vs. Bit-rate.
- PSNR vs. CPU.
- CPU vs. bitrate.

The following section provides the experimental results and detailed analysis of the results.

5.3.1 Experimental results and analysis

The results of MOO and Pareto set analysis are presented in Table 5-2 and Table 5-3 and Figure 5-2 to Figure 5-4 respectively. The Figure 5-2 shows the Pareto front or set of non-dominated solutions for Bit-Rate and PSNR, and the corresponding numerical values related to the optimal performance points on the Pareto curve are tabulated in Table 5-2.

The following summarises and lists the objective function pairs considered in multi-objective optimisation of the H.264 encoder when coding the Foreman and Football sequences. The equation is minimized by multiplying the PSNR equation by a negative sign, i.e. creating $f=-f$.

A. The Foreman Video

function f = foreman(x)

% PSNR versus Bit-rate

% f(1) represent psnr

$$f(1) = -0.5926 * x(3) + 0.0798 * x(4) + 0.0046 * x(6) + 52.0225; f=-f;$$

% f(2) represent bitrate

$$f(2) = 6.8133 * x(1) - 18.5023 * x(3) + 890.0319;$$

% -----

% PSNR versus cpu

% f(1) represent psnr

$$f(1) = -0.5926 * x(3) + 0.0798 * x(4) + 0.0046 * x(6) + 52.0225; f=-f;$$

```

% f(2) represent cpu
f(2) = -0.5347* x(1)+ 0.1297* x(2)- 0.5646 * x(3)+4.0201* x(4)+ 60.362;

% -----
% cpu versus bitrate

% f(1) represent cpu
f(1) =-0.5347* x(1)+ 0.1297* x(2)- 0.5646* x(3)+4.0201* x(4)+ 60.362;

% f(2) represent bitrate
f(2) =6.8133 * x(1) - 18.5023 * x(3) + 890.0319;

End

```

B. The Football Video

```

function f = football(x)

% PSNR versus Bit-rate

% f(1) represent psnr
f(1) = -0.4367* x(3)+ 0.0802 * x(4) + 0.0047 * x(6) + 41.066;f=-f;

% f(2) represent bit-rate
f(2) = - 68.407 * x(3) - 15.2422 * x(4) + 3452.3615;

% -----

% PSNR versus cpu

% f(1) represent psnr
f(1) = -0.4367 * x(3) + 0.0802 * x(4) + 0.0047 * x(6) + 41.066;f=-f;

% f(2) represent cpu
f(2) = - 0.7234 * x(1) -1.3867 * x(3) + 5.2292 * x(4) + 106.0023;

% -----
% cpu versus bitrate

% f(1) represent cpu
f(1) = - 0.7234 * x(1) -1.3867 * x(3) + 5.2292 * x(4) + 106.0023;

% f(2) represent bitrate
f(2) =- 68.407 * x(3) - 15.2422 * x(4) + 3452.3615;

End

```

Figures 5-2, 5-3 and 5-4 plots the dual-objective Pareto fronts obtained for the Foreman video. These plots are generated as output when the ‘gamultobj’ function

[96] of MATLAB is run based on the equation pairs listed above. Similar graphs are created for 'Football' sequence and these are included in Appendix C.

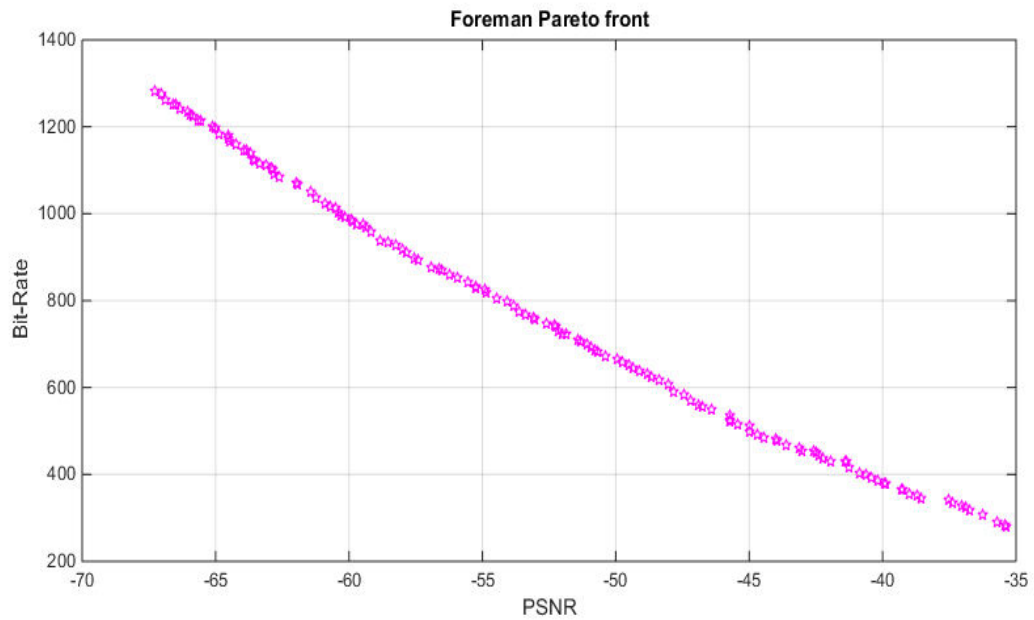


Figure 5-2: Pareto front for foreman PSNR in (db) vs. Bit-Rate in (Kbit/s).

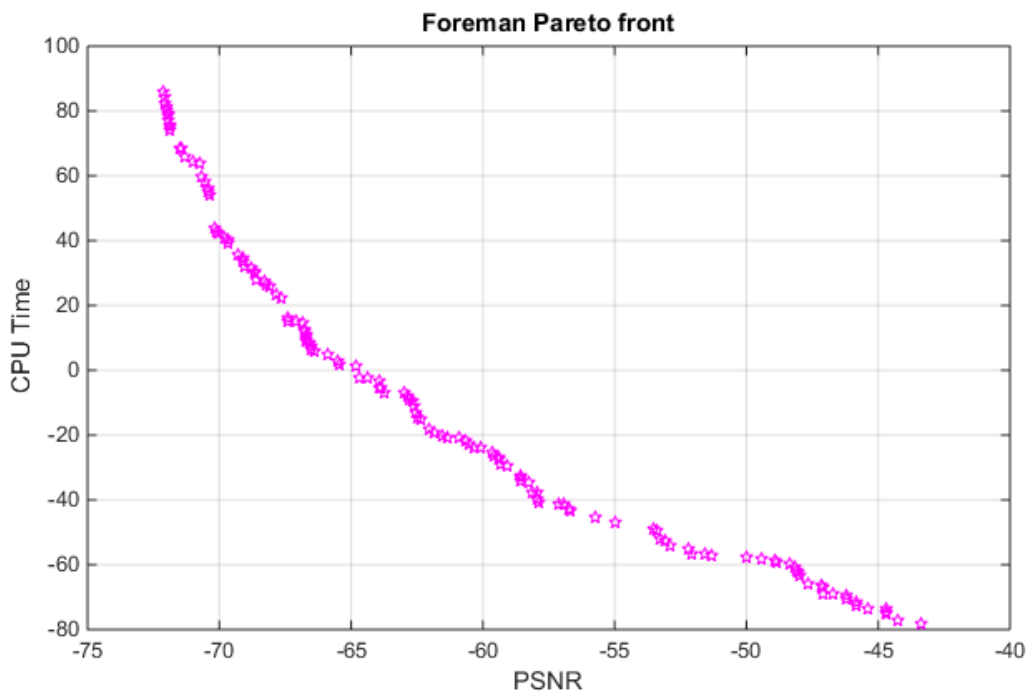


Figure 5-3: Pareto front for foreman PSNR in (db) vs. CPU Time in (sec).

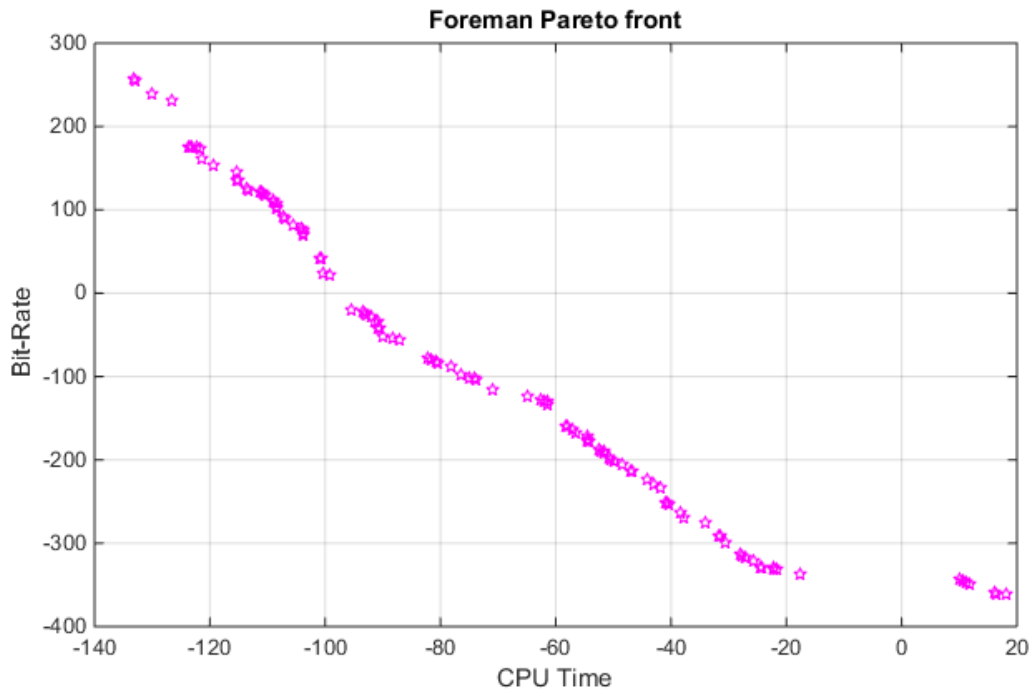


Figure 5-4: Pareto front for foreman CPU Time in (sec) vs. Bit-Rate in (Kbit/s).

Note: since the property of Pareto chart that all data must be positive. Negative values are ignored.

The results of the Bit-Rate vs. PSNR (under multiple constraints of parameter values) optimization operation are listed in table 5-2 which contains values of both objective functions and the relevant values of the parameters that generated optimal performance. Note that each parameter has been set within constraint settings, thus making the dual-objective optimisation problem being addressed, a multi-objective (i.e. dual-objective + multi-constraint) optimisation problem. The list of a sample set of optimal feasible solutions with their functional values, have been listed in Table 5-2. It is noted that the optimal point listed in the table 5-2 is 15 out of 140 optimal points obtained. The number of rows in X is the same as the number of Pareto solutions. All solutions in a Pareto set are equally optimal. For more details, readers are referred to Appendix D.

Table 5-2: The optimal points for foreman PSNR vs. Bit-rate

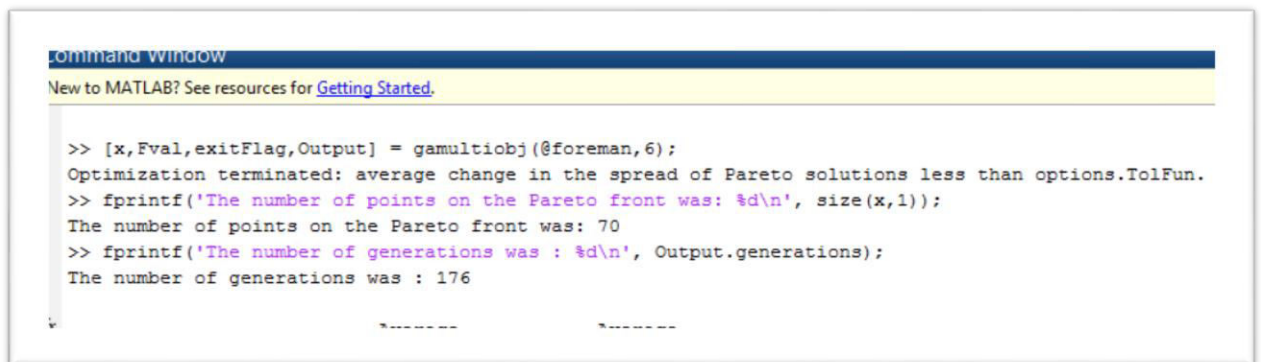
	$f(1)$	$f(2)$	X1	X2	X3	X4	X5	X6
1	-61.4345	1051.543	- 12.3536	5.948443	-13.2784	18.68428	6.922016	11.36318
2	- 56.2065	860.3077	- 15.6062	6.330999	-4.14033	21.28292	8.077764	6.978926
3	-59.1944	959.3262	- 14.4197	5.089091	-9.05509	22.12362	8.069374	8.776518
4	- 62.8529	1102.272	- 10.8373	5.983466	-15.4618	20.18949	7.297968	12.31851
5	-55.2319	827.3298	- 15.3452	7.633843	-2.26184	23.04302	10.27598	6.575443
6	-62.6273	1085.166	- 12.1723	6.900299	-15.0288	20.62279	7.015222	11.51751
7	- 58.0206	919.1406	- 14.9615	5.98795	-7.08268	22.08793	7.667866	8.319399
8	- 40.8703	404.3069	- 13.2207	14.54251	21.38374	18.91533	11.41972	2.244124
9	-57.8374	910.0747	- 14.7867	6.106682	-6.52831	23.847	8.773665	9.403458
10	-52.579	747.9986	- 15.5328	6.103938	1.956696	21.12791	7.869598	6.523647
11	-45.7358	536.7505	- 16.1877	13.29562	13.13297	18.59521	9.332297	2.607052
12	-62.8153	1089.791	- 12.1723	6.900299	-15.2788	21.12279	7.015222	11.51751
13	-56.6037	872.3414	- 14.9189	6.229737	-4.53761	23.22131	8.72742	8.51145
14	- 64.2498	1160.129	- 8.63146	7.039144	-17.7765	20.64461	8.871526	9.890705
15	-64.5445	1179.086	- 6.97241	7.537578	-18.1901	21.32292	9.585319	8.898466

Table 5-3: Output data describing the results of MOO with GA for Figure 5-2 to Figure 5-4

Problem	Number of generations	Size of population	Pareto fraction	Size of non-dominated set	Function count	Average distance	Spread
PSNR vs. Bit-rate	107	200	0.7	140	21601	0.0043	0.3793
PSNR vs. CPU	127	200	0.7	140	25601	0.0196	0.3164
CPU vs. Bit-rate	246	200	0.7	137	49401	0.0442	0.5231

Note that once the Pareto front is plotted, MATLAB's 'gamultiobj' function workspace can be used to display the number of solutions found on the Pareto front and the number of GA generations from which they resulted. The number of solutions found on the Pareto front and the number of generations are found using the following command line operations as presented in Figure 5-5, below. Further details about the optimization problem carried out can be determined by studying the 'output' of the 'gammultiobj' function, as indicated by Figure 5-6.

Figure 5-2 shows the PSNR versus bit rate values in final stage of generations being used of the optimization process. Similarly Figure 5-7 Figure 5-8 illustrates the results.



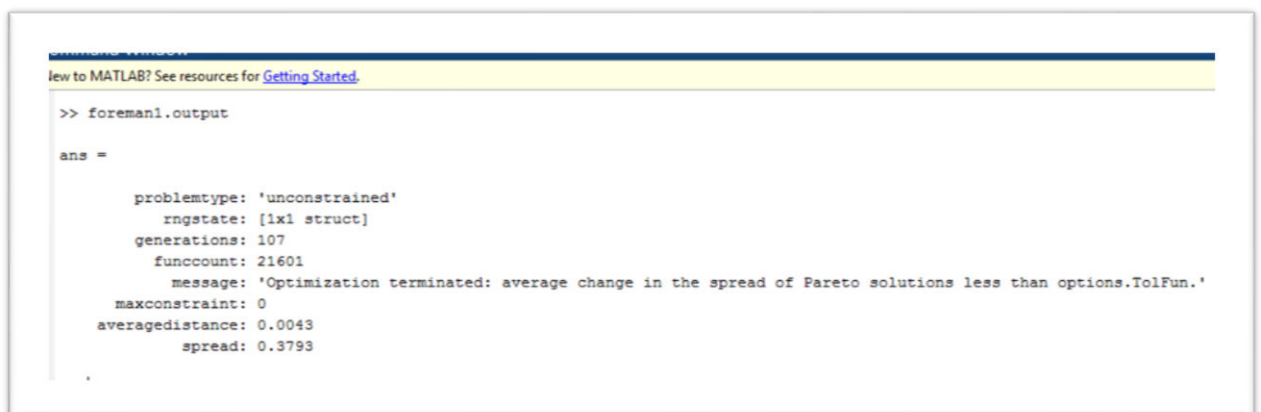
```

Command Window
New to MATLAB? See resources for Getting Started.

>> [x,Fval,exitFlag,Output] = gamultiobj(@foreman,6);
Optimization terminated: average change in the spread of Pareto solutions less than options.TolFun.
>> fprintf('The number of points on the Pareto front was: %d\n', size(x,1));
The number of points on the Pareto front was: 70
>> fprintf('The number of generations was : %d\n', Output.generations);
The number of generations was : 176

```

Figure 5-5: number of solutions and the number of generations.



```

Command Window
New to MATLAB? See resources for Getting Started.

>> foreman1.output

ans =

    problemtype: 'unconstrained'
      rngstate: [1x1 struct]
    generations: 107
     funccount: 21601
      message: 'Optimization terminated: average change in the spread of Pareto solutions less than options.TolFun.'
maxconstraint: 0
averagedistance: 0.0043
         spread: 0.3793

```

Figure 5-6: More details about the optimization

It is noted that in general, Pareto-based multi-objective approaches consider three aspects: closeness to the global Pareto front; spread along the Pareto front; number of solutions of the non-dominated set.

In the Table 5-3, population size and Pareto fraction for the GA are set at 200 and 0.7, respectively, which are considered sufficient to generate search for optimal solutions. The solver will try to limit the number of individuals in the current population that are on the Pareto front to 70 percent of the population size since the Pareto fraction is set to 0.7. For example in optimisation on PSNR vs. Bit-Rate, when the MOO algorithm of MATLAB, 'gamultiobj', is run, after 107 generations and 21601 function counts, the GA selected 140 best individuals to be considered as non-dominated solutions out of 200 individuals in the population. Average distance between individuals was found to be 0.0043, which indicates good convergence of the MOO solution. This is due to the fact that it has a distance of less than 0.05 from the nearest point in the Pareto set.

From the detailed optimisation results provided in Appendix D, Table D. 1, an example optimal point of a H.264 Encoder can be defined as follows:

IntraPeriod is -14.1153, SearchRange is 4.953125, QP is -9.39187, NRFrames is 21.66449, PSNR is 8.107256 and Bit-rate is 8.700779. Whereas the optimal values for PSNR is -59.357 and Bit-Rate is 967.6315 (see Figure 5-7). The figure 5.7 also illustrates two further examples of optimal points.

Similarly, the results showing the Pareto front of non-dominated solutions for PSNR vs. CPU is illustrated in Figure 5-8 and Bit-rate Vs. CPU time are presented in Figure 5-9. In each of the two Pareto fronts, two examples each of optimal points have been indicated.

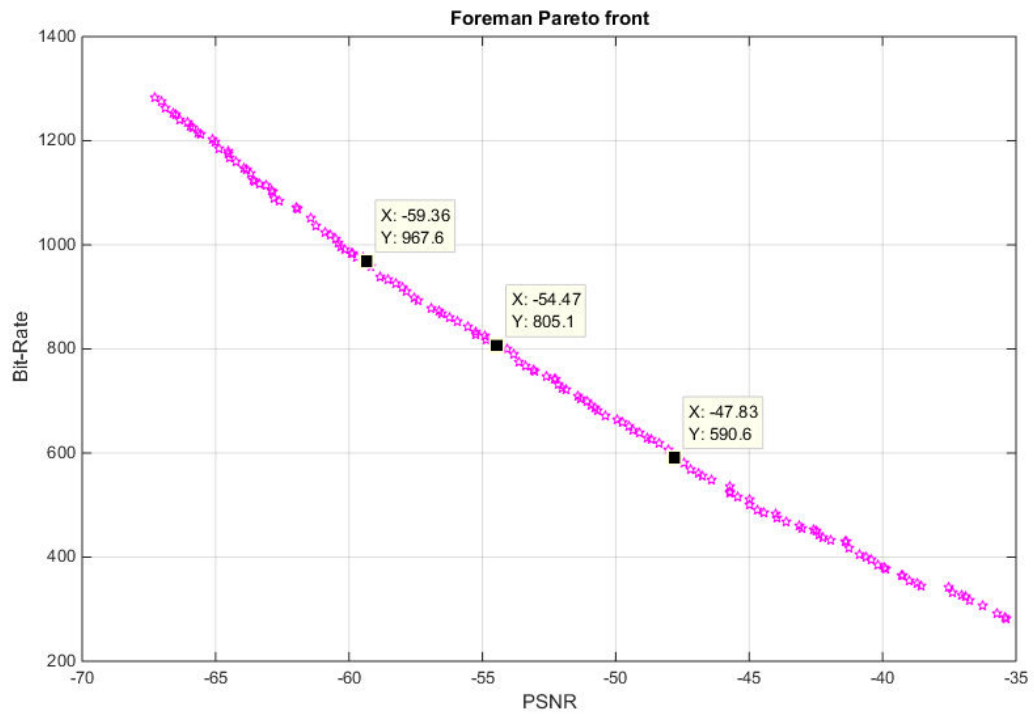


Figure 5-7: Pareto points 1 PSNR in (db) vs. Bit-rate in (Kbit/s)

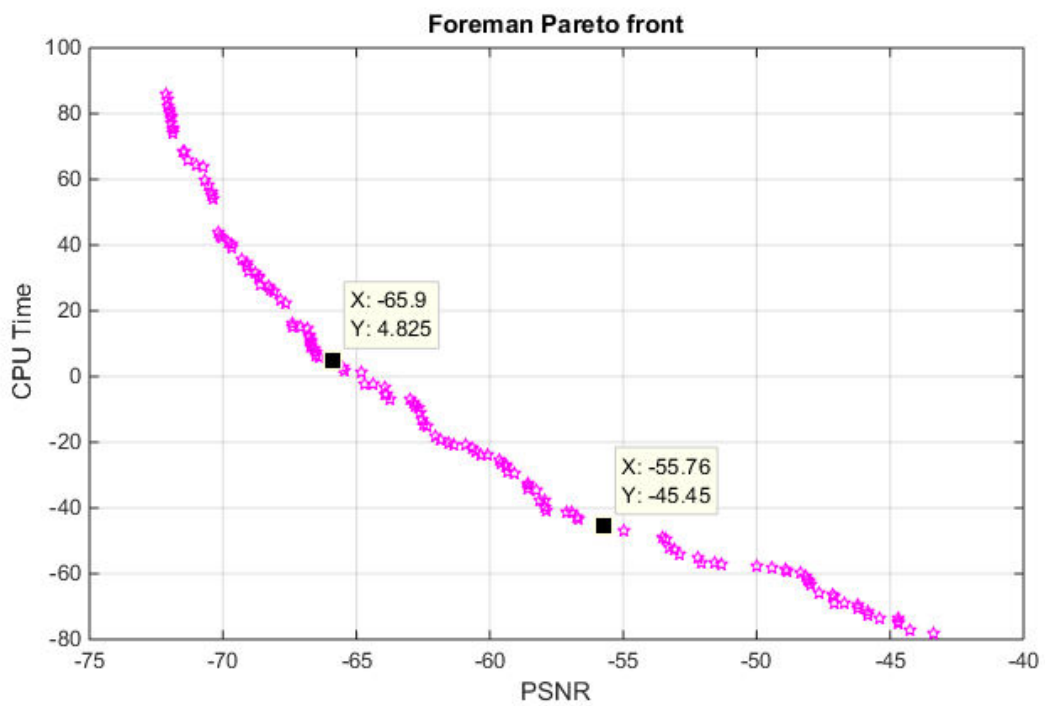


Figure 5-8: Pareto points 2 PSNR vs. CPU Time in (sec).

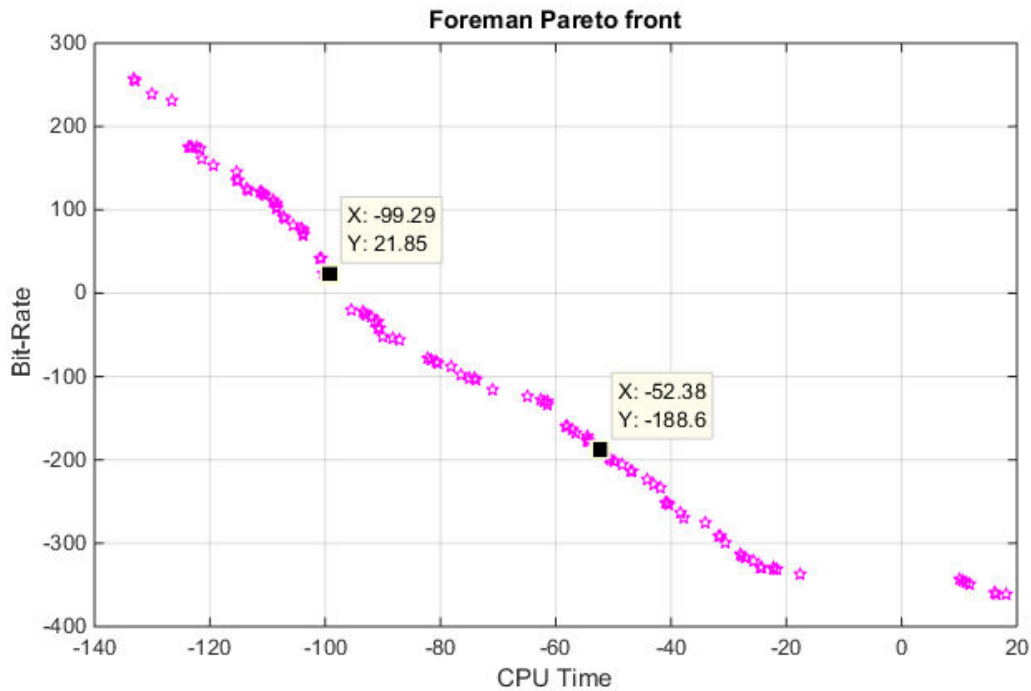


Figure 5-9: Pareto points 3 CPU Time in (sec) vs. Bit-Rate

It is noted that the optimisation procedure described above results in a number of optimal solutions. Each optimal solution defines the values of the parameter settings one should use to obtain optimal PSNR and Bit-Rate, simultaneously.

5.4 Optimising the Decoder

The analysis of the decoder is limited to decoder parameters that have significant effect on only the decoder's CPU time. It is noted that the Decoder parameters have no impact on Bit-rate and PSNR as these are determined by the encoder. In the proposed framework the quality and the bit-rate received by the decoder are the same as the encoder output. Which assumes that the decoder receives all data transmitted by encoder, at the same rate. In such cases the decoder totally depends on encoder coding parameters. The NSGA-II tool is once again used to obtain optimal parameter combinations, as described above in optimising the encoder performance. Figure 5-10 and Figure 5-11 illustrate graphs Pareto front decoder graphs for the Foreman video between Bit-Rate vs. CPU time and PSNR vs. CPU time on the way to final generation. In a manner similar to that carried out in analysing the encoder performance, optimal decoder operational points can be obtained using the same procedure adopted above in the encoder analysis.

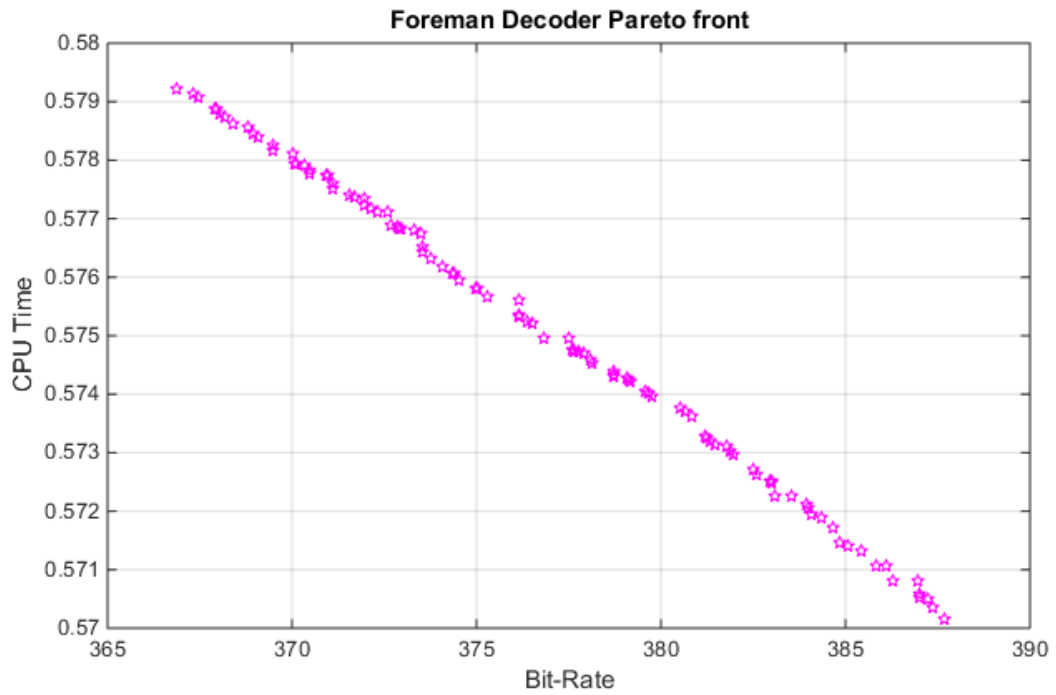


Figure 5-10: Bit-Rate in in (Kbit/s) vs. CPU Time in (sec)

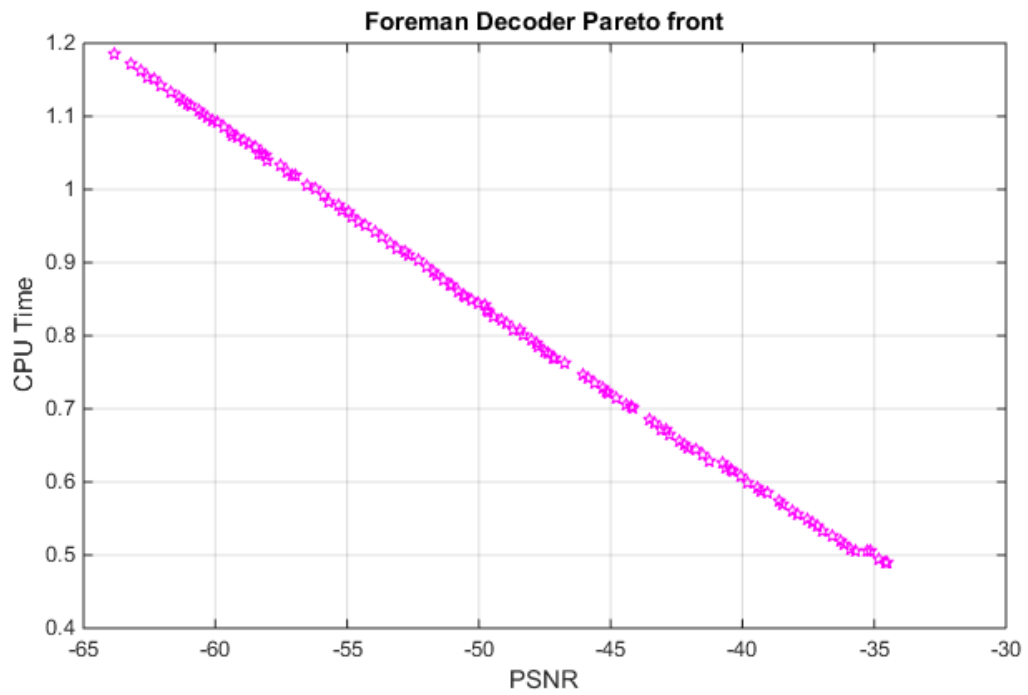


Figure 5-11: PSNR in (db) vs. CPU Time in (sec)

5.5 Summary and Conclusion

This chapter presented how the objective functions obtained for the performance objectives, PSNR, Bit-Rate and CPU time in Chapter-4 can be used to determine the optimal performance configurations of a H.264 encoder and decoder, under multiple constraints/objectives. The encoder and decoder optimal configurations were obtained using three dual-objective, multiple constrained operational conditions, namely PSNR vs. Bit-Rate, CPU time vs. Bit-Rate and CPU time vs. PSNR.

The research carried out in this chapter has demonstrated the use of a genetic algorithm based multi-objective optimisation framework based on carrying out investigations related to a H.264 CODEC. Through the use of this framework it was demonstrated how optimal configurations for the encoder and decoder performance could be obtained. Thus for practical purposes one could use this framework to determine the optimal coding parameters when the operational constraints and objectives are known.

The MOO framework proposed in Chapters 4 and 5 can also be used in relation to any other coding standards. In Chapter-6 we propose to evaluate the use of this framework in the MOO of a H.265 CODEC, the CODEC of the latest video coding standard.

Chapter 6

A Machine Learning based Framework for Parameter based Multi-Objective Optimisation of a H.265 Video CODEC

6.1 Introduction

In Chapters 4 we proposed a framework that is based on the fundamentals of machine learning that can be used to scientifically determine the significant coding parameters of a H.264 video CODEC. These parameters were then used to model the operational behaviour of the H.264 video CODEC for which machine learning algorithms were further utilised. We also showed that these models can be used to establish the foundations of a multi-objective optimisation framework. Although the experiments conducted in Chapters 4 and 5 were limited to the most widely used video coding standard H.264, it was argued that the framework proposed can be used in relation to any video coding standard.

High Efficiency Video Coding (HEVC) also known as H.265 is the most recent answer to the ever growing consumer demands. In this chapter we use the framework proposed in Chapters 4 for the characterisation, modelling and parameter based multi-objective optimisation of a H.265 video CODEC. As the coding algorithms behind H.265 video CODECs are different as compared to the coding algorithms of H.264, H.265 has different parameters and also the impact of these parameters on various performance related features can differ significantly from that of H.264. Given the above in this chapter we use the proposed parameter based multi-objective optimisation framework in the optimisation of a H.265 video CODEC.

For clarity of presentation this chapter is divided into 6 sections. Apart from this section which is an introduction to the research problem, the section 6.2 outlines the proposed framework for performance modelling and the experiments conducted for establishing the framework. Section 6.3 presents a comprehensive analysis of the results of the performance modelling of encode and decoder. Multi-Objective Optimisation Framework for H.265 is explained in section 6.4. The section 6.5 presents the results and analysis of Optimisation stages carried out using a Matlab based implementation. Finally section 6.6 summarizes the chapter by providing a summary of the contribution made.

6.2 Proposed Framework for Performance

Modelling

The proposed framework for a Multi-Objective Optimisation was developed to determine the optimum coding parameters for a H.265 video CODEC, when working under multiple constraints as shown in Figure 6-1. The MOO framework is intended to minimize the CPU time, bit-rate and to maximize the quality of the compressed video stream. MOO framework proposed is accomplished by following the steps below.

1. Profiling experiments on the encoder and decoder were carried out to determine the coding parameters that have a significant impact on each of the objectives/constraints related to rate, distortion and CPU utilization. This was achieved by measuring the impact of each parameter (while being varied) on each of the above aspects.
2. Developing the objective function for each objective/ constraint, based on the above significant parameters,
by using a suitable regression procedure.
3. These objective functions can then be used within a genetic algorithm (GA) based multi-objective optimization framework to determine optimal parameter values.

The focus of this section are the first two steps above, i.e. determining the significant coding parameters and establishing the corresponding objective functions. These

two stages enable the modelling of the performance and are subsequently used in section 6.4 for optimisation of the CODEC.

In a practical multimedia application scenario a device captures a video, encodes it and transmits it via a network to another device that decodes and displays the content to a viewer. Assuming that the network has bandwidth constraints and the device in which the encoder is placed has compute power constraints and the potential viewers of content may demand minimal quality levels, a situation in which the proposed MOO framework can be used.

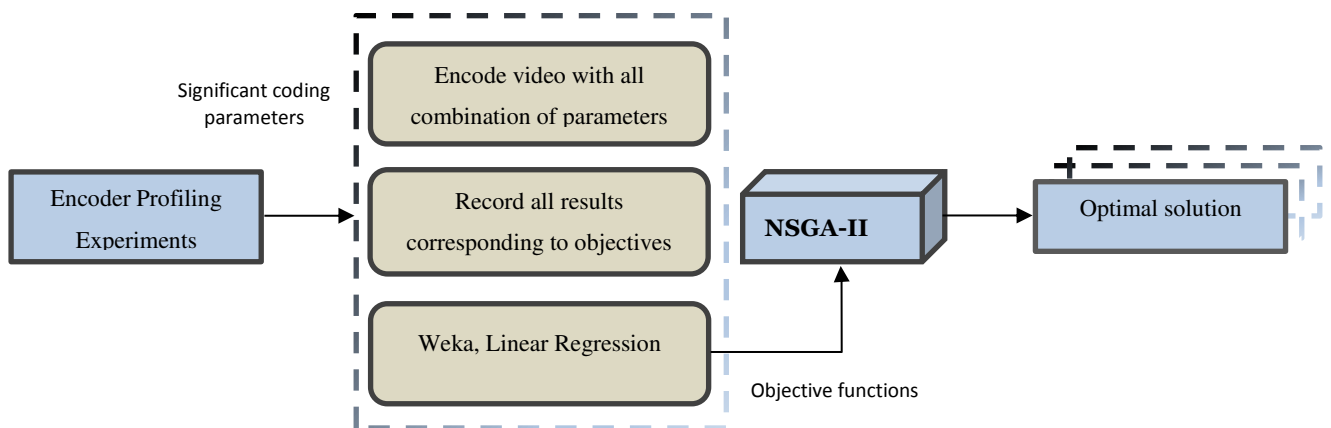


Figure 6-1: Proposed Multi-objective optimisation framework

The significant number of encoder parameters that control the encoders bit rate, quality and computational power requirements can be selected, to ensure the encoder performance is optimal, under the given multiple constraints. However this requires the modelling of the encoders bit-rate, quality and CPU time, based on the large number of selectable encoder parameters. If mathematical objective functions can be derived for each of the above, a standard approach to optimisation can be used. Deriving objective functions, for example using mathematical regression, will need the determination of the significant coding parameters, the key focus of the research presented below.

The same explanation can be applied to the selection of decoder parameters that results in optimal decoder performance. Within the research context of this chapter, the author assumed that the data transmission network is assumed to be perfect, i.e. no delays, no bit loses, no errors etc. Therefore the bit stream generated by the encoder is transmitted without any loss or alteration to the decoder in real-time. The

following section proposes the experimental process adopted to determine the significant coding parameters for both the encoder and decoder.

6.2.1 Profiling Experiments/ Determining the Significant Coding Parameters

This experiment was carried out using the Random Access (RA) configuration file of the Reference software for ITU-T H.265 high efficiency video coding named the HEVC test model (HM) version 16.8 as shown in Appendix A. Different resolutions can be used in each profiling experiment: 1080p which is representative for (Full HD) high definition systems with resolution of 1920x1080 pixels in a 16:9 aspect ratio, 2K Video a display resolution of 2560x1600 pixels with a 16:10 aspect ratio and 2160p (Ultra HD) which is representative for the next generation of high quality video. Each video sequence was encoded using a selected combinations of possible parameter values of initial set of encoder parameters.

In other words each encoding instance corresponds to a combination of coding parameter values, selected from the possible exhaustive set that can be determined by varying each parameter within its entire range. For example instead of using quantization parameter variations between 1-51 (that is the exhaustive set), only three sample values, 27, 37 and 45, were used (for further examples see Table 6-1) The table also tabulates the sample values used in our experiments for each parameter from within their corresponding value ranges.

Table 6-1: Settings for the Encoder in HM

Parameter	Meaning	Values Range
SourceWidth SourceHeight	Specifies the width and height of the input video.	1920x1080 2560x1600
FrameRate	Specifies the frame rate of the input video.	Depends on video
Internal Bit Depth	Specifies the bit depth used for coding. When 0, the setting defaults to the value of the MSBExtendedBitDepth.	8
Coding Unit Size/Depth	Maximum coding unit width in pixel Maximum coding unit height in pixel	64/4 64/4
IntraPeriod	Period of I-frames. Specifies the intra frame period. A value of -1 implies an infinite period.	(16,32,48)
GOPSize	Specifies the size of the cyclic GOP structure.	8
FastSearch	The use of a fast motion search.	1:TZ search
SearchRange	Sets allowable search range for motion estimation.	(64,128)
Fast Encoding	Fast encoder decision	(0 or 1)
Quantization Parameter	Specifies the base value of the quantization parameter. If it is non-integer, the QP is switched once during encoding.	(27,37,45)
Asymmetric Motion Partitioning	Enables or disables the use of asymmetric motion partitions.	1
Sample adaptive offset (SAO)	Enables or disables the sample adaptive offset (SAO) filter.	1
Rate Control	Rate control: enables rate control or not.	0

Table 6-2 Shows selected sample frames of a set of six video sequences with different resolutions. Note that typical resolutions used in conjunction with H.265 video coding standard, i.e., 1080p and 2K resolution videos are used in all experiments, to carry out the analysis and make the relevant conclusions of this research. However, without any restrictions the proposed framework can be used in relation to a video sequence of any resolution, in particular HD and full-HD 2K, 4k and beyond. The six selected video sequences have different properties of object motion, both in the foreground and background. Further differences exist in the scene content.

Table 6-2: Tested Video Sequences

	
YachtRide_1920x1080	Traffic_2560x1600
	
BasketballDrive_1920x1080	Jockey_3840x2160
	
Cactus_1920x1080	YachtRide_3840x2160

The experiments were initially conducted on a HP computer, running Microsoft Windows 8.1 (64-bit), having an Intel Core i5 CPU 4200Y @ 1.40 GHz and 4.00GB RAM. However it was found that coding HD resolution video is an intensive task that required for example, if encoded in the computer with the above specification, 10 hours to encode 50 frames of a 1920x1080 video at QP 37, intra period 48 and search range 64. Therefore subsequently a decision was made to make use of a High Performance Computing (HPC) facility.

Thus for all the experiments a HPC system using Redhat Enterprise Linux v6, with 20 cores of Intel Ivy Bridge Xeon E5-2670 containing 64GB RAM was used significantly reducing the execution time per experiment.

A sample of 36 data instances of the Cactus video sequence are presented in TABLE 6-3. These were used in the final stage of modelling the PSNR, Bit-rate and CPU time. These are the inputs to the [20] linear regression based modelling process that result in the three objective functions that include the significant parameters, Intra Period as x_1 , Search Rangex₂, Quantization Parameter x_3 and Fast Encoding x_4 .

The resulting objective functions for Bit-rate, PSNR and CPU time are the final outcomes of the performance modelling of the CODEC. Separate experiments are performed for each of the sample test videos. For more details see Appendix F.

Table 6-3: Selected Set Of Parameters For Cactus Sequence

x_1	x_2	x_3	x_4	Bitrate in (Kbit/s)	PSNR in (db)	CPU Time in (sec)
16	64	27	1	8422.096	36.8076	2000.97
16	64	37	1	2195.592	32.7418	1612.08
16	64	45	1	736.12	28.9058	1474.57
16	128	27	1	8424.696	36.8054	2140.34
16	128	37	1	2197.152	32.7447	1722.19
16	128	45	1	735.304	28.9092	1556.66
16	64	27	0	8414.944	36.8149	2559.02
16	64	37	0	2196.192	32.7507	2106.38
16	64	45	0	735.864	28.9111	1925.5
16	128	27	0	8414.864	36.8149	2778.27
16	128	37	0	2195.184	32.7516	2287.14
16	128	45	0	736.728	28.9173	2067.34
32	64	27	1	6993.976	36.7337	2115.22
32	64	37	1	1726.648	32.6277	1698.1
32	64	45	1	561.512	28.7824	1553.63
32	128	27	1	6991.08	36.7323	2271.37
32	128	37	1	1725.24	32.6279	1819.83
32	128	45	1	561.896	28.7877	1634.66
32	64	27	0	6994.6	36.7419	2684.76
32	64	37	0	1725.12	32.6325	2198.14
32	64	45	0	563.04	28.7949	2002.86
32	128	27	0	6991.312	36.7421	2923.43
32	128	37	0	1725.536	32.6336	2411.49
32	128	45	0	561.952	28.7952	2164.01
48	64	27	1	6914.36	36.7438	2126.42
48	64	37	1	1722.568	32.6039	1719.68
48	64	45	1	560.656	28.7335	1556.47
48	128	27	1	6911.576	36.7428	2295.48
48	128	37	1	1720.96	32.5996	1867.74
48	128	45	1	559.032	28.7422	1656.8
48	64	27	0	6911.616	36.7488	2690.7
48	64	37	0	1720.944	32.6059	2216.83
48	64	45	0	562.424	28.742	2018.13
48	128	27	0	6912.472	36.7515	2962.86
48	128	37	0	1720.176	32.6066	2437.26
48	128	45	0	560.872	28.7534	2200.32

6.2.2 The Objective Functions of the HEVC Encoder

Based on the output of the linear regression algorithms applied as explained above, the objective functions for the three objectives (for the Cactus video) are found as presented in equations (Equation 6-3) These functions provide the means to discuss in detail the significance of each parameter and how they affect the PSNR, Bit-rate and CPU encoding time. The following section provides an analysis of the experimental results. In particular the analysis considers the test videos separately and discusses the impact of each coding parameter given the known properties of the contents of each video. [Note that for each video a different model is generated based on the video's inherent properties.]

$$f(1)_{\text{Bitrate}} = -22.4664 * x(1) - 386.2482 * x(3) + 18066.616$$

(Equation 6-1)

$$f(2)_{\text{PSNR}} = -0.0039 * x(1) - 0.4404 * x(3) + 48.873$$

(Equation 6-2)

$$f(3)_{\text{Enc_Time}} = 3.9537 * x(1) + 2.5501 * x(2) - 36.2174 * x(3) + 545.1239 * x(4) + 2768.025$$

(Equation 6-3)

6.3 Analysis of experimental results

Experimental analysis was conducted separately for the encoder and decoder and can be presented as follows.

6.3.1 Encoder Analysis

The Encoder objective functions obtained as a result of the experimental procedure presented in section 6.2 enables one to discuss the significance of each of the coding parameters. Following are the obtained models for each video sequence, with $f(1)$ representing PSNR, $f(2)$ rate and $f(3)$ CPU encoding time.

Table 6-4 tabulates the correlation coefficients of the objective functions. They range between 0-1. A value closer to 1 represents the fact that the dependant variable (in this case Bit-Rate, PSNR or CPU time) can be predicted very accurately from the coding parameters that play a role and has been included within the objective functions.

Table 6-4: Encoder Correlation Coefficient

Video	PSNR in (db)	Bitrate In (Kbit/s)	CPU Time in (sec)
Cactus	0.9989	0.9551	0.988
YachtRide	0.9981	0.9532	0.9837

In analysing the objective functions (Equation 6-3), higher positive coefficients of coding parameters indicate higher positive dependency and higher negative coefficients represent higher negative dependency. If a certain parameter is not present in the objective function that means that the objective is independent of that parameter. A careful analysis of the coding parameters that have non-zero weighting factors in the objective functions obtained and a comparison of relative magnitudes of the coefficients can lead to a direct correspondence with the properties of the video, for e.g., the presence of motion in foreground and background, the speed of movement of objects, sudden scene changes, camera pan/tilt/zoom effects and the general characteristics of the content of the video as well.

For example, the analysis of the linear regression equations obtained for cactus video sequence identifies all four parameters to have significant impact on CPU time, namely:

- IntraPeriod
- Searchrange
- Quantization parameter
- Fast Encoding

For the same video the following parameters were identified to have a significant impact on Bit-rate.

- IntraPeriod
- Quantization parameter

The parameters that are identified to have a significant impact on PSNR are:

- IntraPeriod
- Quantization parameter

A more detailed and video sequence specific analysis can be presented as follows.

6.3.1.1 Analysis of the CPU Time Experiment

The objective functions obtained for all tested video sequences for CPU encoding time indicates that the parameter that has the most significant impact on CPU is Fast encoder decision. Further in selection of the Intra-Period, more I frames (smaller intra period) results in a higher processing time. The next significant impact is from the Quantization parameter. The impact from search range SR and Intra Period (IP) is relatively insignificant.

When search range Increases encoding time will slightly increase. These tests has no major impact on quality of the video. Disabling FEN will also slightly increase encoding time. However it has no major impact on quality.

6.3.1.2 Analysis of the PSNR Experiment:

The parameter that has the most significant impact on PSNR is QP. The PSNR results tabulated in TABLE 6-4 indicate that the two videos with the least amount of movement/changes, namely Cactus and YachtRide have the best correlation coefficients. This is expected due to the stability of the CODEC during the encoding of the individual frames of the coded sequence.

6.3.1.3 Analysis of the Bit-Rate Experiment:

The parameter with the most significant impact is the QP. Lower quantiser result in higher bitrate and correspondingly higher visual quality as illustrated in Figure 6-2 (QP) has a very important impact on the compression rate of H.265.

In cactus both PSNR and Bit-Rate has no impact from the Search Range. This is true given the fact that for videos with fast moving objects, best matches will not be found quickly, i.e. without having to scan the entire video. All objective functions includes a similar constant term indicating that a fixed computational cost for encoding is present, which is independent of the selection of coding parameters. This is expected given the processes that exist, which are independent of the coding parameters.

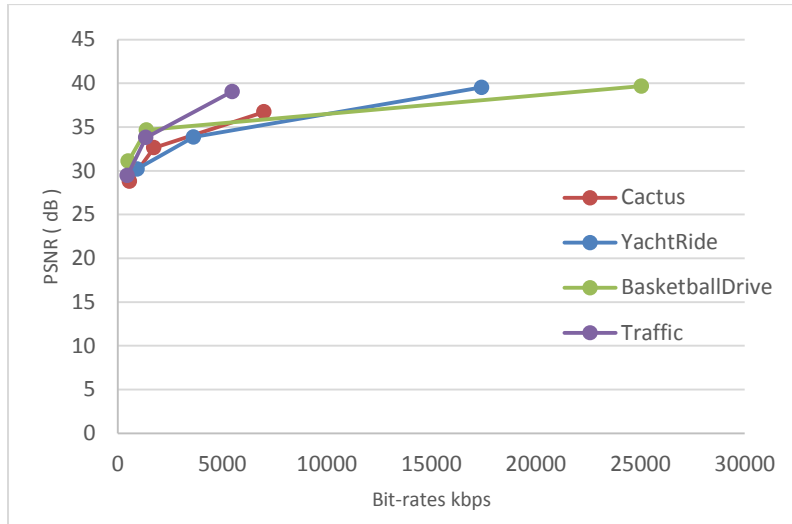


Figure 6-2: PSNR versus Bit-rate at QP 27, 37, 45.

6.3.2 Decoder Analysis

The analysis of the decoder is limited to decoder parameters that have significant effect on only the decoders CPU time. Note that the Decoder parameters have no impact on Bit-rate and PSNR as these are determined by the encoder. In the proposed framework the quality and the bit-rate received by the decoder are the same as the encoder output. The CPU time of the decoder is analysed using the same method used at the encoder end. In this section the experiments were performed in order to find out those coding parameters that can significantly influence CPU time. The objective functions thus obtained are listed within equation (Equation 6-4).

$$f_{Dec_Time} = 0.0023 * x(2) - 0.0819 * x(3) + 0.1178 * x(4) + 8.292$$

(Equation 6-4)

Table 6-5: Decoder Correlation Coefficient

Video	Decoding CPU Time in (sec)
Cactus	0.9509
YachtRide	0.9263

Table 6-5 presents the correlation coefficients of the objective functions. The cactus video sequence has the highest correlation coefficient. The analysis of the linear regression equations is carried out to identify parameters that have significant

impact on CPU time. (Equation 6-4) reveals that the Fast Encoding has the most significant impact being the highest magnitude coefficient.

The Encoder and Decoder analyses indicate that the objective functions obtained as a result of using the proposed framework are able to accurately define the significant coding parameters and further detail the level of significance of each parameter. They can also be related to the motion and content information of the videos. More importantly these objective functions model the behaviour/properties of the encoder and decoder thus allowing them to be used in multi-objective optimisation as described in the next section.

The HM decoder configuration file takes an h.265 file as input and outputs a raw YUV video stream as a reconstructed file as shown in Appendix E. The output for one example video, YachtRide, illustrated in Figure 6-3 shows coding artefacts for the video frame 30 of the sequence using a quantization parameter (QP) of 45 that gives a very low quality with PSNR of 28.7877 db and a QP 27 that gives a very good quality video with PSNR 36.7323 db. When the QP is increased during the encoding of the video, the bit rate reduces and the video loses information. The Highlighted areas show regions where some artefacts occur between the Original video at frame 30 and the Encoded video at QP 45. The blue square in the Original video shows the visible windows of a building and the red ellipse shows a big tree. On the other hand the blue square and red ellipse in the Encoded video at QP 45 shows the artefact of the two mentioned area. Where the windows and the tree are not very much clear compared to the original video.

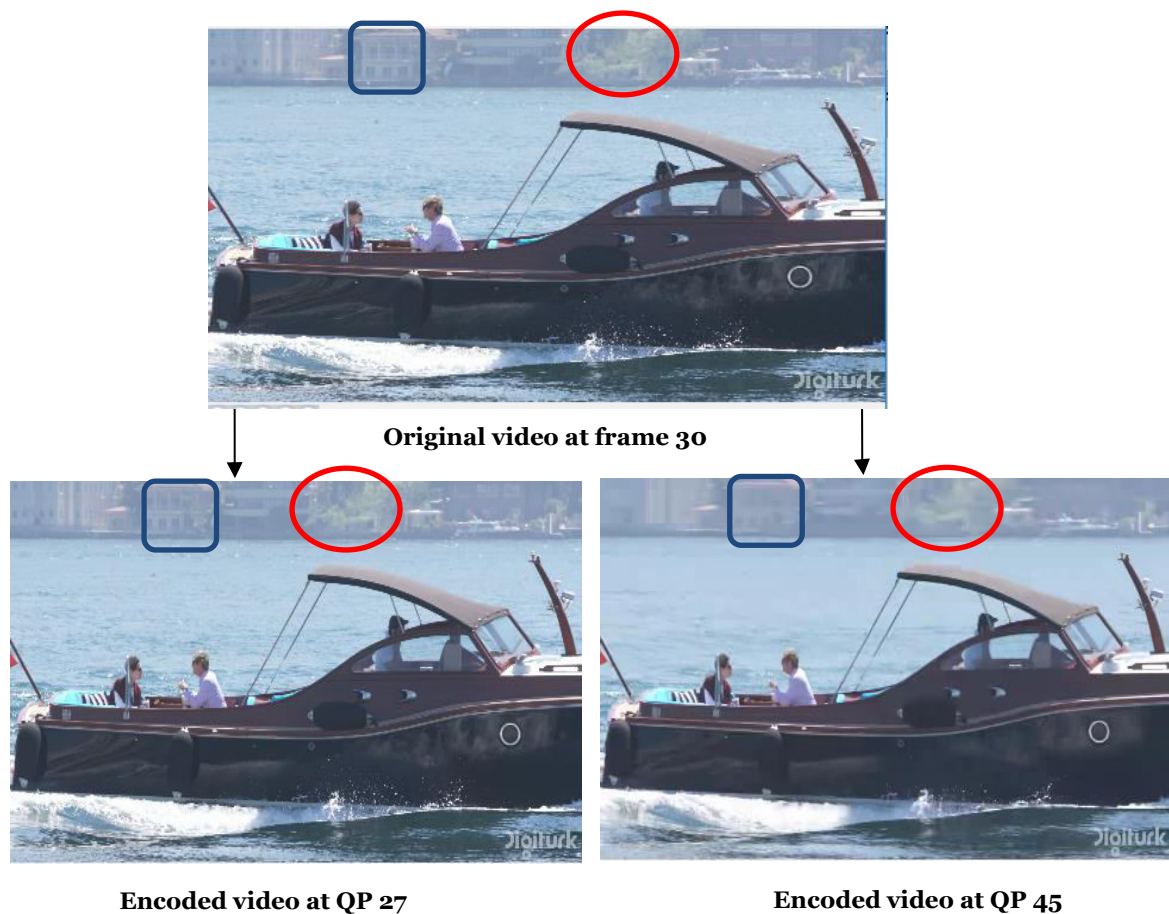


Figure 6-3: The visual artifact with different QP.

6.4 Multi-Objective Optimisation of a H.265 Video CODEC

Section 6.3 has investigated the parameters that have a significant impact on the encoder and decoder performance of a H.265 CODEC. It carried out the modelling of the H.265 codec's performance in terms of bit-rate, PSNR and computational cost (i.e. encoding/decoding time).

This Section presents multi-objective optimisation of a H.265 video CODEC. Specifically, an optimization scheme is proposed to determine the optimum coding parameters for a H.265 video codec in a bandwidth constrained environment, which minimises CODEC time and maximises video Quality.

In section 6.3 a mathematical formulation by means of regression was used to associate the significant coding parameters of a H.265 video CODEC with the

relevant performance related objectives. Solutions to the optimization problem are reached through a Non-dominated Sorting Genetic Algorithm (NSGA-II). NSGA-II is implemented in the genetic algorithm `gamultobj`, available in the MATLAB optimization tool as described in Chapter-5.

As mentioned earlier, the framework presented in this chapter aims to obtain a set of compression parameters that produces the highest image quality, while satisfying the need to minimise bandwidth requirements at the lowest possible computational cost. With the intention of addressing these objectives, a novel multi-objective optimisation framework is proposed. Therefore, the outcome of this framework is a set of all feasible solutions that represent the best trade-offs between the above mentioned objectives.

6.4.1 Implementation

The framework was implemented on a HP computer, running Microsoft Windows 8.1 (64-bit), having an Intel Core i5 CPU 4200Y @ 1.40 GHz and 4.00GB RAM.

The Multi-objective Genetic Algorithm Solver in MATLAB “`gamultiobj`” attempts to create a set of Pareto optima for a multi-objective minimization. Additionally the solver uses the genetic algorithm for finding local Pareto optima. The algorithm is first initialised by defining the population size, the total number of generations, and the number of variables. In the proposed framework we want to minimize two objectives, each having several decision variables. To achieve optimum performance, the fitness function is maximized by minimizing the negative of the function.

To make use of the MATLAB ‘`gamultiobj`’ function, one needs to provide at least two input arguments, a fitness function, and the number of variables in the problem. The fitness function required is the multi-objective (vector) function that needs minimising. The functions of three objectives of each video sequences given in equation 6.1 were used to optimise the encoder performance under multiple constraints [37]. These functions were then fed to the NSGA-II [25] optimization tool along with the fitness function and number of variables. The NSGA-II provides all sets of optimal results that jointly minimize CPU time, bit-rate and maximizes quality. Since a single 3D graph, is complex to visualize the optimality of the results, pairs of graphs were plotted. The experiment was conducted, and the following

setting of parameters for the GA was chosen (see oTABLE 6-6). The ‘gamultiobj’ function finds a local Pareto front for multiple objective functions using the genetic algorithm. To obtain a Pareto front for two objective functions were used each of four decision variables. We also impose bound constraints on the decision.

To finds a Pareto set x with the default optimization parameters, options can be created with the ‘gaoptimset’ function of MATLAB.

$$x = \text{gamultiobj}(\text{FITNESSfunction}, \text{NVARs}, A, b, Aeq, beq, LB, UB, options).$$

Linear equalities and inequalities and parameter bounds satisfy the following:

$$A^* x \leq b$$

$$Aeq * x = beq$$

$$LU \leq x \leq UB$$

Variables Bounds Constraints for Upper and Lower bounds used are as follow:

$$16 \leq x_1 \leq 48$$

$$64 \leq x_2 \leq 128$$

$$27 \leq x_3 \leq 45$$

$$0 \leq x_4 \leq 1$$

Table 6-6: Optimisation settings

gamultobj settings

Fitness function: @function.

Number of variables: 4

Bounds Constraints: lb = [16,64,27,0]

ub = [48,128,45,1]

Creation Function: Constraint dependent

Population Size: 60

Initial Population: Default

Crossover Fraction: 0.8

Mutation Function: Constraint dependent

Crossover Function: Intermediate

Crossover Ratio: 0.8

Pareto Front Population Fraction: 0.35

Maximum Generations: 300

Plot functions: Pareto front

6.5 Optimising the encoder

In this study, the objective functions were used to optimise the encoder. The functions of three objectives of each video sequences given as (Equation 6-3) are then fed to the NSGA-II optimization tool along with the fitness function and number of variables. The NSGA-II provides all sets of optimal results that jointly minimize CPU time, bit-rate and maximizes quality. Since a single 3D graph, is complex to visualize the optimality of the results, pairs of graphs where plotted.

An optimization problem is one requiring the determination of the optimal (maximum or minimum) value of a given function, called the objective or fitness function, subject to certain defined restrictions, or constraints placed on the variables concerned.

Since the Optimisation minimize the objective function or fitness function. That is, they solve problems of the form

$$\min_x f(x).$$

If one wants to maximize $f(x)$, minimize $-f(x)$, because the point at which the minimum of $-f(x)$ occurs is the same as the point at which the maximum of $f(x)$ occurs.

To achieve optimum performance, function is maximized by minimizing the negative of the function. The PSNR value as a measure of the quality of the video is to be maximised. So that the function to be maximized is minimized by multiplying the PSNR equation by minus one, *i. e.* $-f(x)$.

The objective functions depend on four parameters expressed as x in the MOO problem formulation, which include Intra-Period, Search-Range, Quantization-Parameter and Fast Encoding. Each pair consists of two objectives functions.

For example in Cactus and YachtRide two pairs were obtained independently.

- PSNR vs. Bit-rate.
- CPU vs. Bitrate.

The proposed multi-objective optimisation solution minimises the components of $f(x)$ subject to identified constraints.

6.5.1 Experimental results

This section describes a set of experiments that were designed to test the correct functionality of the proposed optimisation framework. As seen in Table 6-6, the optimisation algorithm was set to generate a population of 300 chromosomes, each consisting of four decision variables.

NSGA-II was used because of its ability to find an optimum set of solutions that is close to the Pareto-optimal set. The goal of these simulations was to obtain a set of Pareto-optimal solutions for each of the three video Sequences.

The results of Multi-Objective Optimisation Pareto set analysis are presented in Figure 6-4 and Figure 6-5. The figures show the Pareto front or set of non-dominated solutions for Bit-Rate vs. PSNR and CPU vs. Bit-Rate. The Pareto front illustrated is within a limited range and hence shows that the points lie on a straight line. In practice when the range of testing is increased the shape of the curve would represent a typical shape of a Pareto curve. The Pareto curve allows one to select optimum performance points and hence select the corresponding coding parameters that resulted in the objective function optimal values for coding the videos.

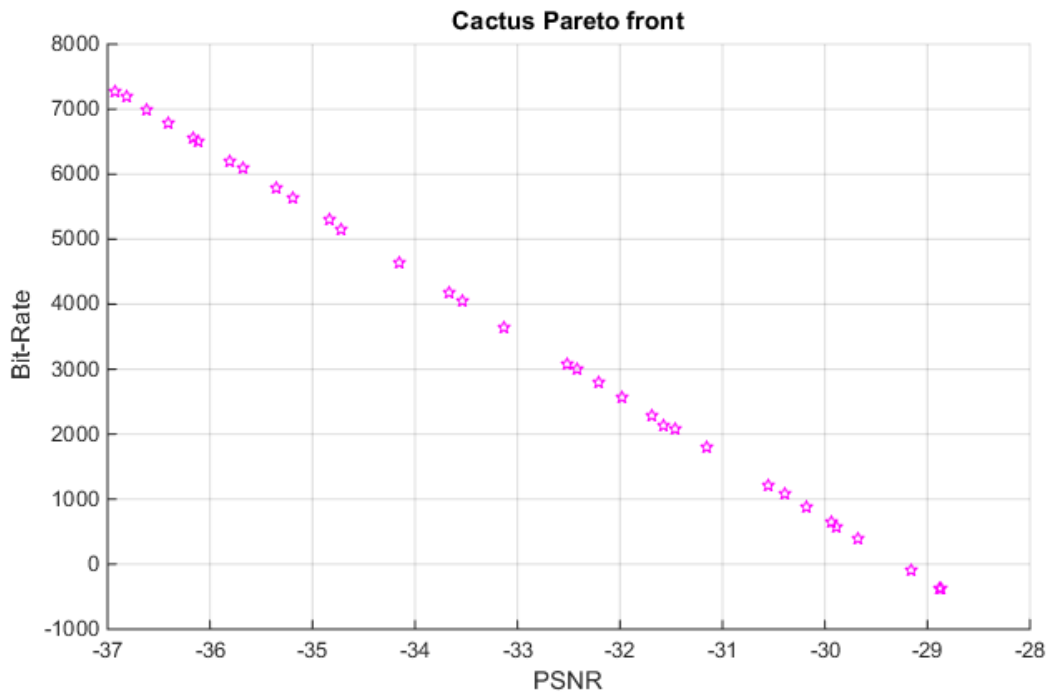


Figure 6-4: Pareto front for cactus video sequences PSNR in (db) vs. Bit-Rate in (Kbit/s).

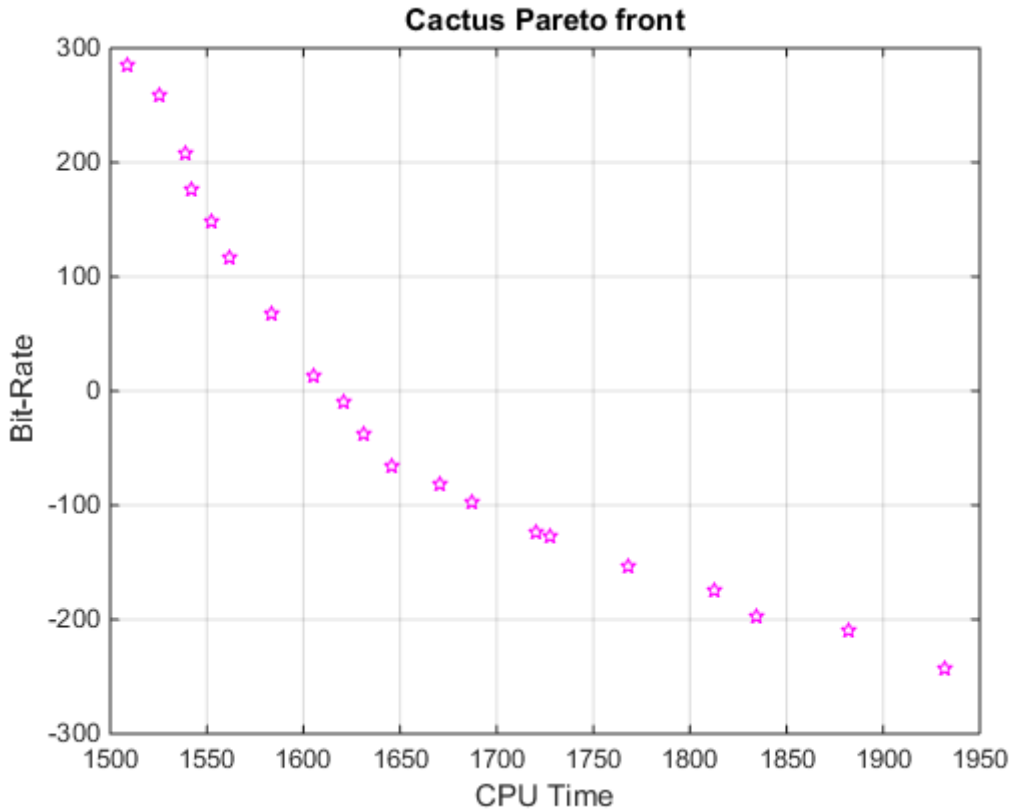


Figure 6-5: Pareto front for cactus video sequences CPU Time in (sec) vs. Bit-Rate in (Kbit/s).

The results of the optimization appear in the following table containing both objective function values and the value of the variables. The list of all feasible solutions with their functional values, the Pareto front have been illustrated in Table 6-7. The number of rows in X is the same as the number of Pareto solutions. All solutions in a Pareto set are equally optimal. For more details see Appendix G

Table 6-7: The optimal points for Cactus PSNR vs. Bit-rate

	$f(1)$	$f(2)$	$x1$	$x2$	$x3$	$x4$
1	36.9201	7278.453	16	64	27	0
2	36.6212	6984.67	17.65991	65.59725	27.66406	0.025905
3	28.8721	-382.632	47.6418	118.2422	44.99415	0.997951
4	30.5614	1223.873	41.08539	107.0511	41.21625	0.755286
5	-32.204	2800.026	33.97008	94.28146	37.54945	0.402902
6	35.8094	6208.247	21.04581	71.59679	29.47728	0.095358
7	-31.579	2126.931	40.52844	106.1184	38.91062	0.494995
8	-	1807.857	37.79216	101.8517	39.89587	0.521224

	31.1558					
	-					
9	35.1985	5622.217	23.68162	76.01039	30.8412	0.143016
	-					
10	34.7259	5144.716	26.9901	83.80625	31.88502	0.316507
	-					
11	28.8721	-382.632	47.6418	118.2422	44.99415	0.997951
	-					
12	33.1295	3651.057	31.90089	91.30083	35.46647	0.327118
	-					
13	34.1591	4637.964	27.49851	83.34469	33.16743	0.264558
	-					
14	29.9441	652.0496	42.68032	111.0691	42.60394	0.746148
	-					
15	33.5393	4061.154	29.24273	86.94946	34.55934	0.693433
	-					
16	36.9201	7278.453	16	64	27	0.25
	-					
17	29.1627	-93.9621	45.86701	115.8487	44.35001	0.837885
	-					
18	31.4594	2069.315	38.04606	102.0389	39.20418	0.660673
	-					
19	36.1234	6503.75	19.99021	71.19112	28.77362	0.274147
	-					
20	29.8873	584.6452	43.6029	111.2834	42.72479	0.841717
	-					
21	30.3913	1080.419	40.78522	108.4374	41.60511	0.687554
	-					
22	29.6739	389.8275	44.005	111.6657	43.20578	0.46095
	-					
23	-34.839	5303.11	23.88389	77.82799	31.65561	0.392679
	-					
24	30.1757	877.9415	41.48435	109.8443	42.08867	0.626643
	-					
25	28.8772	-368.384	47.1292	116.9709	44.98708	0.24776
	-					
26	-33.67	4173.746	29.35103	85.91427	34.26154	0.168342
	-					
27	36.1568	6550.884	19.05001	68.97729	28.70628	0.05654
	-					
28	36.4119	6783.901	18.56366	68.46437	28.13128	0.166284
	-					
29	31.9862	2575.949	35.70312	97.30352	38.02878	0.592786
	-					
30	32.5238	3089.451	33.49928	93.62668	36.82751	0.519083
	-					
31	-35.346	5782.687	22.05095	74.52345	30.52059	0.11781
	-					
32	31.6959	2283.743	37.67948	101.6304	38.67035	0.669224
	-					
33	35.6817	6085.708	21.59696	72.51968	29.76248	0.105324
	-					
34	32.4242	2993.15	33.97008	94.28146	37.04945	0.402902
	-					
35	-36.81	7181.891	16	64	27.25	0

Table 6-8: Output data describing the results of MOO with GA for cactus.

Problem	Number of generations	Size of population	Pareto fraction	Size of non-dominated set	Function count	Average distance	Spread
PSNRvs. Bit-rate	259	60	0.35	35	13001	0.0149	0.1487
CPU vs. Bit-rate	220	60	0.35	21	13261	0.0308	0.1050

6.5.2 Discussion

Multi-objective optimisation (gamultiobj) for tested Cactus video consists of two objective functions. Figure 6-6 shows the PSNR vs. bit rate values in final stage of generations being used of the optimization process.

The first two output arguments returned by gamultiobj are X, the points on Pareto front, and FVAL, the objective function values at the values X. A third output argument, exitFlag, that's states the reason why gamultiobj stopped. A fourth argument, OUTPUT, contains information about the performance of the solver. The fifth argument is POPULATION that contains the population when gamultiobj terminated and a sixth argument, SCORE that contains the function values of all objectives for POPULATION when gamultiobj terminated.

Pareto-based multi-objective approaches consider three aspects: closeness to the global Pareto front; spread along the Pareto front; number of solutions of the non-dominated set.

In the Table 6-8, population size and Pareto fraction for the GA are set at 60 and 0.35, respectively, which are considered sufficient to generate search for optimal solutions.

At 259 generations and 13001 function counts, the GA selected 35 best individuals considered as non-dominated solutions out of 60 individuals in the population. Average distance between individuals is 0.0149, which indicates good convergence of the MOO solution. Since it has a distance of less than 0.05 from the nearest point in the Pareto set.

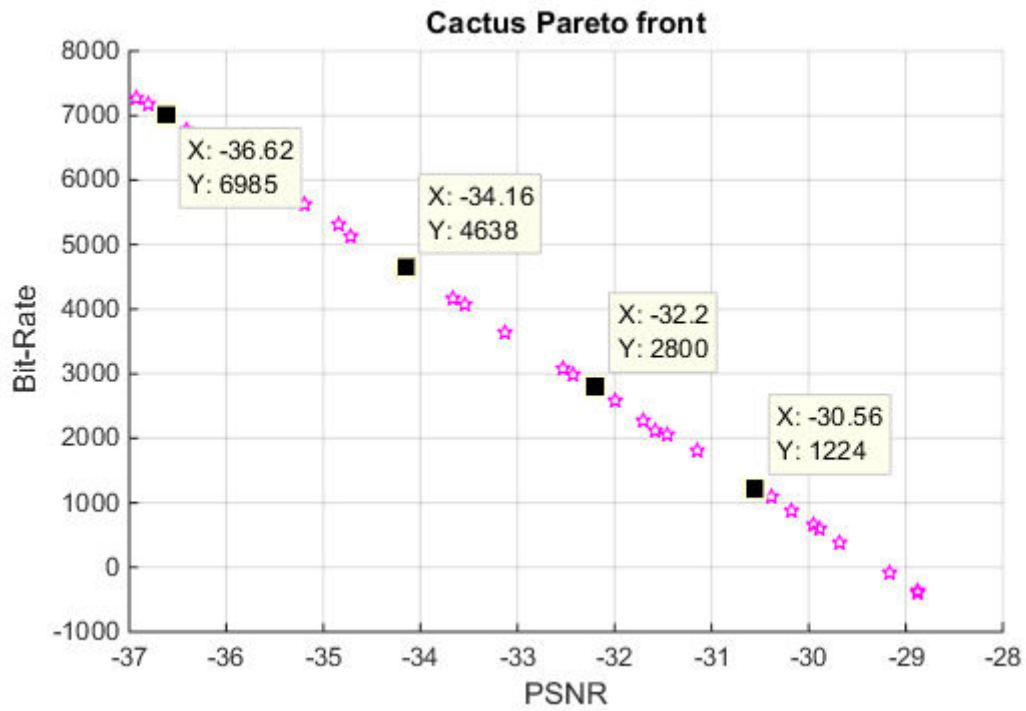


Figure 6-6: Pareto selected points for Cactus PSNR vs Bit-Rate.

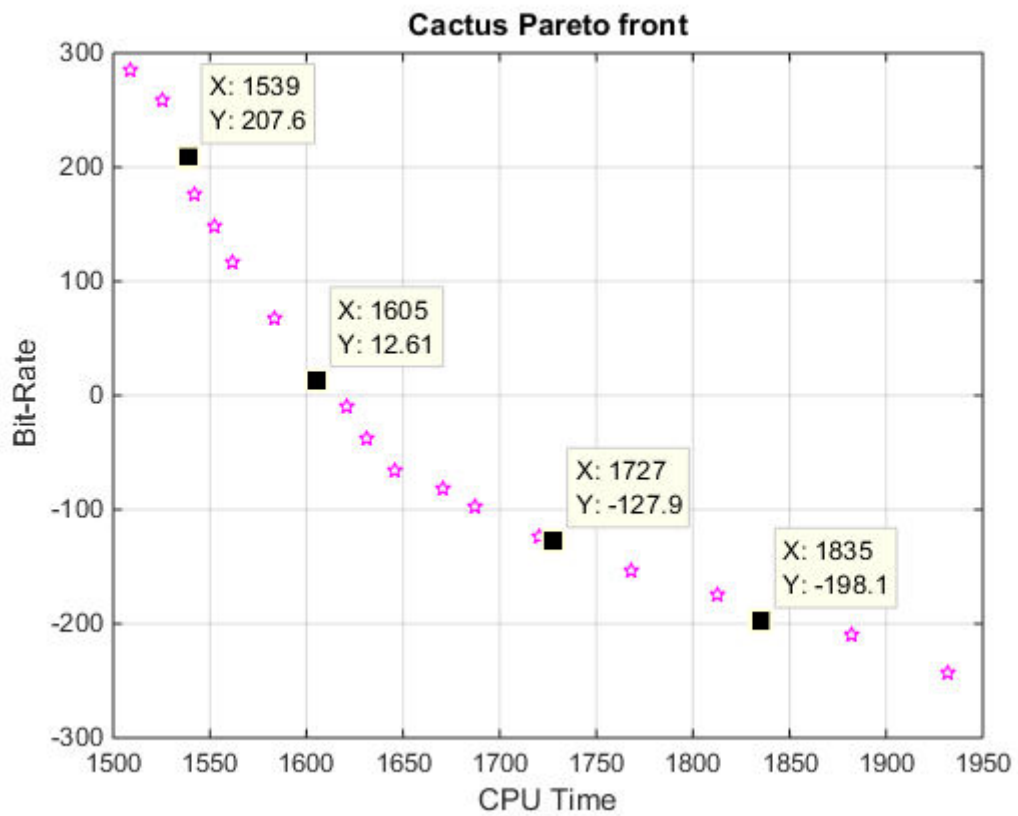


Figure 6-7: Pareto selected points for Cactus CPU Time vs Bit-Rate.

For example the Row 13 in Table 6-7 gives a set of parameters that results in an optimized H.265 CODEC performance. At this operational point the Intra-Period is 27.49851, Search-Range is 83.34469, QP is 33.16743, Fast Encoding is 0.264558. The optimal values thus obtained are PSNR (X) , -34.1591 dB, Bit-Rate (Y), 4637.964 (see Figure 6-6). It is note that under practical conditions all independent parameter values should be integers, most notably, fast encoding should be set to '0'.

Similarly, the results showing the Pareto front of non-dominated solutions for Bit-rate Vs. CPU time is presented in Figure 6-7.

6.6 Summary

In this chapter we have proposed a machine learning based approach for the determination of significant coding parameters of a H265 video CODEC. In particular we have used multivariate regression analysis in defining objective functions for CPU time, PSNR and the bit-rate of a video CODEC when a given video is being encoded/decoded. We have been able to use known information about the content and the motion present in the test videos to justify the formation of the objective functions. We have shown that these regression equations provide the means for modelling the performance of a typical H.265 video CODEC. Finally we have have used these models to optimise the performance of a video CODEC under multiple constraints. For this purpose we demonstrated the effective use of a Genetic Algorithm based appriach.

The research conducted in this chapter confirms that the proposed generalised framework for the performance analysis, modelling and multi-objective optimisation of a video CODEC previously demonstrated on H.264 as presented in Chapter 4 and 5, can be applied to any video CODEC including H.265 and provides a useful contribution to the video coding community who are often faced with the delima of selecting values for a large number of coding parameters with the intention of obtaining optimal performance of the CODEC under multiple performance constraints.

Chapter 7

Impact of Packet Loss & Network Delay on Optimally Coded Video Streaming

7.1 Introduction

Video streaming is becoming widely adopted in today's IP based networks. In video streaming, the video is played out while parts of it is being received and decoded. Video streaming applications use User Datagram Protocol (UDP), which unlike Transport Layer Protocol (TCP) used in data transmissions, provides unreliable transmissions. The UDP protocol is also infamous in terms of the video quality levels it can practically support in the presence of packet loss, network delay, jitter and out of order packet delivery. In the case of video transmission in wired networks in particular, the possibility of packet loss is very likely due to Network congestion. Losing sequential packets during transmission can result in the loss of video packets or a delay in their arrival at the destination which can affect the received video playback.

In order to cater for the above challenges within reasonable practical limits, modern video CODECs have been equipped with error-concealment algorithms and effective buffering algorithms that attempts to reduce the ultimate impact of packet loss and network delay. As these algorithms largely differ from video coding standard-to-standard and due to specifics of their implementations, the real impact of packet loss, network delay, jitter and out of order packet delivery cannot be theoretically or conceptually assessed and can only be assessed by detailed practical means. As previous researcher studies have focused on investigating the relevant impacts in detail in particular with regards to H.264 and H.265 standards.

In Chapters 5 a framework for the performance modelling and multi-objective optimisation of a video CODEC was presented and in particular tested and evaluated on H.264 (chapters 4 & 5) and H.265 (chapter 6) video CODECs. In the

work presented in chapters 5 the focus was on investigating the CODEC's performance under encoder/decoder operational constraints, such as limits of relevant coding parameters of both the encoder/decoder, expected video quality at the decoder and compression rates applied. No attempt was made to include network constraints such as packet loss or delay in the modelling processes carried out and in the subsequent optimisation of the CODEC performance. However in an end-to-end delivery of video the complete process involves, encoding of the video content at the encoder under encoder practical constraints and viewer expectations, transmission of the coded video bit-stream in a practical network under network losses/constraints and the decoding of the received video under decoder constraints. Therefore the multi-objective optimisation of end-to-end video delivery over a network (wired or wireless) should consider encoder, network and decoder constraints, altogether in performance modelling and optimisation. Due to the large number of parameters and constraints this introduces to the modelling and optimisation processes, in this thesis we carry out a detailed investigation as to the impact of optimised video delivery determined as per the video CODEC optimisations the thesis has so far investigated (i.e. ignoring network constraints), under the influence of practical network constraints. In a practical design and implementation scenario the optimal selection of Encoder and Decoder parameters can only be determined based on known or design constraints of the CODEC and the network. Packet loss and network delays are subject to constant changes due to network congestion, especially. Therefore using these network parameters in an end-to-end optimisation becomes meaningless in practice. This is a further reason that this chapter investigates the impact of packet loss and network delays, in video coded optimally as per the methods proposed in Chapters 5. The results will provide a valuable evaluation of more detailed performance and behaviour of video CODECs.

In addition to the above the practical system that was put together within the research context of this chapter to simulate the video delivery over wired networks will provide a platform for future research and the experience thus gathered documented in detail in chapter will be a contribution to researchers working in this area in general.

In this chapter, we study the scenario of the Real-time video communication of stored video. In streaming mode, the video can be played back while parts of it are

being downloaded. While streaming if the video data is not received in time, the video will experience delay which can create annoying artefacts of image/video quality during playback. Another common issue is that of the loss of video packets during transmission. When the required video packets are lost before arriving at the receiver, the quality of the video play out at the receiver is affected. Thus, it is important to understand the effect of each of these scenarios on the quality of the received video.

Within the research context of this chapter end-to-end video communication using two P2P network clients and a server base was configured to stream videos. Packet loss rate was introduced to simulate packet losses using the Clumsy [97] tool. The same tool was used to simulate network delay. Furthermore, video quality evaluation the Evalvid tool has been used to measure the PSNR of the video streaming service in the work proposed.

For clarity of presentation this chapter is divided into several sections. Apart from this section that presented the research context and the justification of the research to be conducted and presented, Section 7.2 focuses on the design and implementation of the simulation environment. Section 7.2.2 discusses the video streaming experiments, presenting results and a detailed analysis. Finally section 7.4 summarises and concludes the research conducted.

7.2 System Design and Implementation

In this section we present the detailed system design of an end-to-end video streaming and simulation environment that uses a basic network loss/delay simulator ‘Clumsy’ [97] and the widely used network simulator, OPNET [98].

Fundamentally there are five steps that need to be followed to design and evaluate a video streaming system (see Figure 7-1):

1. Prior to the streaming the raw video file (YUV-uncompressed) is encoded using H.264 JM encoder software to generate a bit-stream file. It is noted that the input file is encoded using an example encoder parameter set that resulted in an optimal coding of the video (see chapter 5). This stage is named as the pre-processing stage and is illustrated as a separate functional block in Figure 7-1

2. The Bitstream file is then received by the EvalVid tool (see Chapter-3) which transcode the bit-stream to a MP4 bit stream, producing the video Trace File [80] EvalVid tool also produces the PSNR before and after MP4 transcoding. The PSNR 'before' transcoding refers to the PSNR of the received H.264 bit-stream after local decoding and PSNR after refers to the PSNR of the MP4 bit stream after local decoding. It is noted that it is the PSNR before transcoding that depicts loss due to H.264 coding. The PSNR after depicts loss due to MP4 coding. It is noted that the transcoding to MP4 is required as it is only the MP4 format that is supported by network simulators.
3. After that the experiments performed for testing the effect of packet loss rates, was conducted using the client and server approach in which one machine was configured as a client and another was configured as a server computers are connected with each other using the cross Ethernet cable. Before transmitting the videos to the client using Mp4 file in VLC, Clumsy was used to introduce different amounts of packet loss on the server machine. The streamed video file will then be evaluation using Evalvid Tool to compare with the PSNR of the reference video. Note that the Trace File is needed for the purpose of using the OPNET simulator.
4. The coded MP4 bit stream is then sent to the server. A VLC media player embeds the bit-stream within a network abstraction layer to prepare the bit – stream for transmission over a wired network. The server incorporates an additional tool, Clumsy that can incorporate packet loss in the bit stream and/or delays in buffering the bit stream to the channel. The bit-stream is then transmitted over the network and received by a further VLC player that separated the video bit-stream from header information used in effectively transmitting the video over a network to a given destination.
5. Finally the video bit-stream is received by a second EvalVid tool that transcodes the video into a H.264 bit-stream. In a practical transmission set up the H.264 bit stream will then be decoded and will be ready to be displayed. It is noted that the PSNR due to H.264 encoding/decoding was measured at the first EvalVid tool. The PSNR due to MP4 coding is measured at the second EvalVid tool.

A video stream application runs on the client computer connected with a server by a communication medium (the network), e.g., an Ethernet Local Area Network (LAN) cable for wired communications.

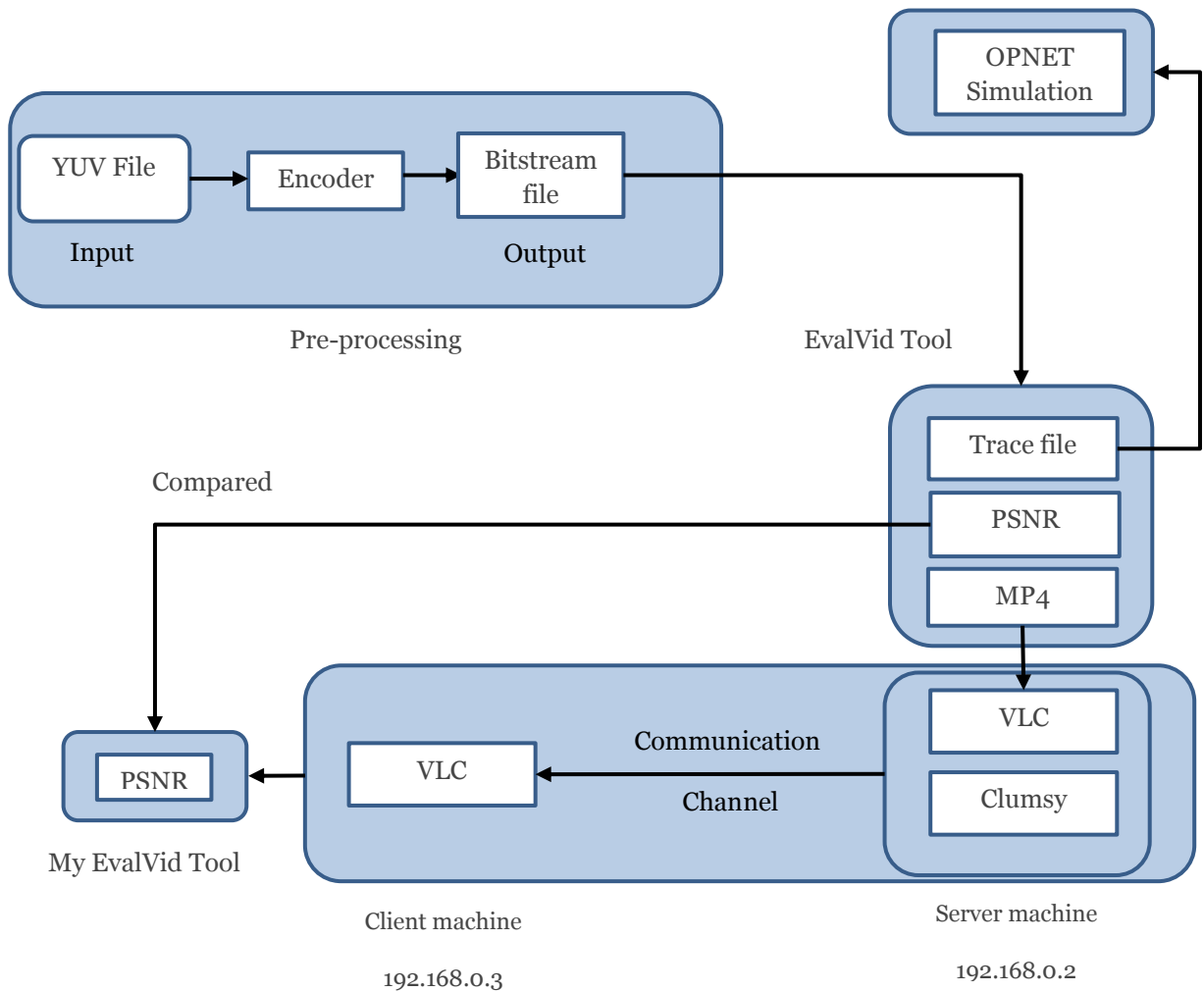


Figure 7-1: Block diagram of streaming procedure

7.2.1 Tested video sequences

The test sequences that have been used in the experiments conducted include videos with different characteristics of content and are listed in Table 7-1. Each video has objects with different motion characteristics, e.g. from slow to high phased movements. As Akiyo video sequence has objects with slow motion properties, since the background is fixed. Foreman (Qcif) has moderate motion as a man appears to be talking and the camera seems to move. Football has fast movement of players. Table 7.1 indicated the different type of videos used their resolution and number of frames used for the experiments.

Table 7-1: Video sequences and their properties

Video Sequence	Foreman	Akiyo	Football	Foreman
Resolution	76x144 (QCIF)	352x288 (CIF)	76x144 (QCIF)	1280x720
Total Frames	300	300	260	248

7.2.2 Video streaming experimental setup

Before starting the experiments, all required components or the video coding and streaming system were configured and tested using a ping command. The configurations to be conducted and steps that needs to be taken are as follows;

1. Configure IP Address in both server and client using a class C IP address.
2. Configure firewalls on both client and the streaming servers.
3. Configure packet loss on the server
4. Start VLC streaming of the video content from the server side
5. Record received video using VLC player at the client computer.

7.2.3 Client - Server communication

A client-server communication system contains three main components specified in Figure 7-1, i.e. the streaming server machine, communication channel and a receiver client machine.

On the server machine, before transmission, raw video files are encoded and compressed using H.264 encoding technique using the JM Reference software. Clumsy tool is used for emulating the lossy transmission medium, with packets being randomly dropped based on the specified percentage of packet loss rate. Video is streamed using VLC media player after receiving the coded bit-stream from server. It is noted that the VLC media player has ‘socket’ functionality that enables communications between two peer-to-peer or client-server computers.

The Processes used to start file streaming from the server machine are as follows:

1. Set the packet drop rate in the clumsy tool.
2. In the VLC media player choose the streaming video mp4 file.
3. Choose the protocol used in streaming. Note that UDP was used as a protocol in the experiments conducted.
4. Enter the IP address of the client (destination host) and the UDP port (1234 in his research). In this research the client address used was 192.168.0.3 and the port address was 1234.
5. Set transcoding option to H.264.
6. Click Stream to commence streaming the video.

On the client side the receiver component is responsible for the reception and playback of the streamed video.

For the playback at the client three further steps are required.

1. The VLC media player receives and opens the network stream.
2. In the URL Select enter UDP udp://192.168.0.3:1234 which is the IP address of the client machine and Enter port number
3. Click play

7.3 Experiments, results and analysis

The experiments were conducted using the client and server communication approach in which one machine was configured as a client and another was configured as a server. In the experiments conducted within the research context of this thesis, the client computer used was an Asus machine powered by Intel(R) Core(TM)2 Duo CPU, 2.00GHz, 2.53GHz, 32 bits, x64 processor, 300 GB Hard Drive is the storage memory. The Server is HP machine powered by Intel(R) Core(TM)dual CPU 2.00GHz, 2.00GHz, 32 bits, 150GB Hard Drive is the storage memory.

The 'Clumsy' tool was used to introduce different amounts of packet loss. The delay and packet loss settings were set using the clumsy software before transmitting. The packet loss values that were used in the experiments conducted are: 1%, 2%, 3%,

4%, 5% and 6%. Clumsy drops packets randomly once a percentage drop has been specified. The delay related experiments were concluded applying the following values: 20 ms, 30 ms, 50 ms, 80 ms, and 100 ms.

The streaming of the optimally coded video was implemented using a selected set of optimal points from the Foreman video's PSNR vs. Bit-rate pareto-optimal graph illustrated in Figure 5-7, chapter 5. Table 7-2 tabulates the selected optimal points. For example, selecting row-67 IntraPeriod is 13, SearchRange is 10, QP is 20, and NRFrames is 16. Whereas the optimal values for PSNR was 41.3778, Bit-Rate is 429.7228 as shown in Table 7-2. Note that the optimal value of PSNR represents image quality due to H.264 compression only. In the network simulation to be carried out as the H.264 bit-stream is transcoded to a MP4 bit stream first, before transmission, the transmitted video quality will be much lower than the above optimal value.

Table 7-2: Select optimal points from Figure 5-7 in chapter 5 for foreman video.

Row	f(1) _{psnr}	f(2) _{Bitrate}	X _{1IP}	X _{2SR}	X _{3QP}	X _{4NRF}
11	45.7358	536.7505	-16.1877	13.29562	13.13297	18.59521
25	36.2372	305.7522	-7.10057	8.441657	28.96404	16.3452
47	41.4123	430.5598	-12.5901	9.493237	20.19704	16.52902
67	41.3778	429.7228	-12.5433	9.524487	20.25954	16.56418
24	37.0587	327.1868	-7.61426	9.242817	27.6164	16.63976
99	37.3786	333.0412	-8.18074	9.941671	27.09138	16.83673

The Table 7-3 present the newly encoded results with coding parameters of the optimal points rounded-off, used as input encoder parameters. Note that Row-67 coding instance has been selected as the representative instance for all loss experiments below.

Table 7-3: Encoded video with different parameter sets for foreman video using optimal points.

Row	f(1) _{psnr}	f(2) _{Bitrate}	X _{1IP}	X _{2SR}	X _{3QP}	X _{4NRF}	psnr	Bitrate	Encoding Time
11	45.7358	536.7505	16	13	13	16	47.866	835.30	60.058
25	36.2372	305.7522	7	8	29	16	36.976	165.35	30.943
47	41.4123	430.5598	13	9	20	16	42.849	393.51	40.600
67	41.3778	429.7228	13	10	20	16	42.838	393.51	40.054
24	37.0587	327.1868	8	9	28	16	37.059	151.42	28.897
99	37.3786	333.0412	8	10	27	16	37.846	170.37	29.998

Table 7-4 shows the final result of the PSNR of each video sequence at the receiving end after server-to-client transmission. Note that for the QCIF Foreman video, the 23.26 dB value quoted is the quality of transmitted MP4 bit-stream at 0% loss. It should be noted that this value is significantly lower than the optimally coded H.264 bit stream quality, 42.838 dB quoted above, before being transcoded to MP4. In other words transcoding to Mp4 has created a significant loss of quality in the first place. When various levels of packet loss are introduced (1%-6%), the results of the PSNR of the received video generally reduce (when packet loss % is increased) from 15dB to 14dB, which is a significant reduction in quality. The results of transmitting other test videos incorporating increasing levels of packet loss also indicate a significant drop in image quality when packet loss percentage is increased. It is however noted that in videos with fast motion (e.g. football) the packet loss significantly reduces image quality. Such videos have significantly different adjacent frames that result in needing a significantly high percentage of the bit-stream to be allocated to coding the frame differences and motion estimations. Therefore loss of information that results from loss of packets could create more serious problems of the decoder's ability to reconstruct the frames that corresponds to the lost frames, as adjacent frames will be very much different and hence the predictions will be less accurate.

Table 7-4: Packet Loss settings and the resulted PSNR (quality)

Loss %	Foreman(Qcif)	Akiyo(Cif)	Football(Qcif)	Foreman(720p)
0%	23.26 dB	36.38 dB	37.01 dB	23.98 dB
1 %	15.52 dB	25.06 dB	15.85 dB	17.19 dB
2%	15.04 dB	23.67 dB	16.24 dB	15.04 dB
3%	15.27dB	24.12dB	15.97 dB	15.77 dB
4%	16.71 dB	24.09 dB	16.04 dB	16.41 dB
5%	16.83dB	27.19 dB	16.09 dB	15.53dB
6%	13.89dB	27.25 dB	16.29 dB	15.50 dB

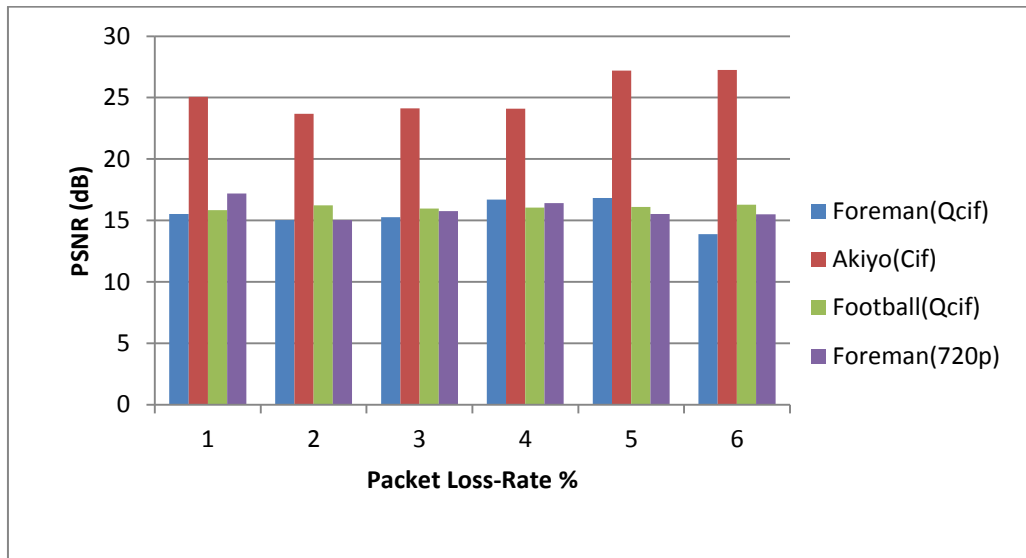


Figure 7-2: The effect of different packet loss rates on the PSNR.

Figure 7-2 illustrates a bar-chart showing how packet loss impacts the video quality of four different videos. It is seen that the Akiyo video sequence that has the least motion operates at a high level of quality and as a result indicated larger variations of quality when the packet loss rate is gradually increase from 1% to 6 %.

To check the effect of packet delay on the quality of video streamed video, the Akiyo video sequence encoded at optimal encoder settings (chapter 4 and 5) was streamed while introducing different amounts of delays in milliseconds (ms) using the clumsy software. Different delay values were used to analyse the impact of delay and the result obtained are tabulated in Table 7-5. It is seen that the PSNR reduces when the delay is increased from 10 ms to 100 ms as presented in Figure 7-3. The results show a good level of resilience to network delay that is being demonstrated by the H.264 CODEC for this video with minimal motion.

Table 7-5: Impact of packet-delay on the quality of the Akiyo video sequence

Delay in (ms)	PSNR In (db)
10	25.09
20	25.76
30	25.04
50	23.21
80	23.63
100	23.22

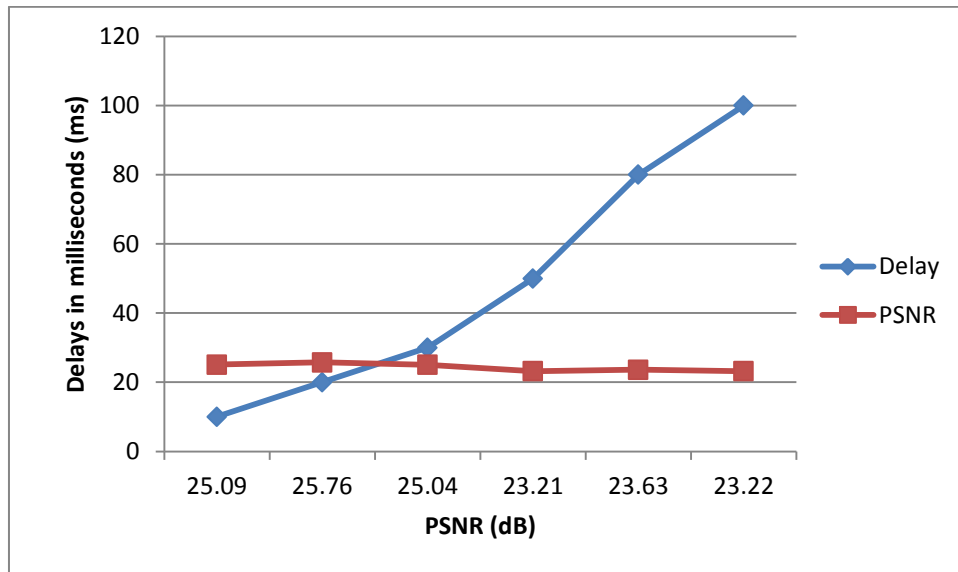


Figure 7-3: Impact on PSNR with different levels of delay for Akiyo video.

The Figure 7-4 confidently demonstrates the perceivable image quality loss that results from both packet loss and packet delay. For all videos the artefacts are more visible around the areas of motion as it is the residues of video coding (i.e. frame differences after motion prediction and correction) that dominates most of the bit stream and hence most affected due to losses or delays in the bit streams. Figure 7.4 (b) also shows that it is in the football sequence with fast motion that the impact of perceptual quality is highest, as compared to other videos, when subjected to packet loss. Figure 7.4 (c) illustrates the impact of packet delay. Note that packet delay creates artefacts that are more spread-out over the frame, rather than confined to the motion related areas. Note that in the foreman video, due to the camera motion there is some motion in the background area and this is why artefacts are shown in the background areas as well.



Foreman (Qcif)



Akiyo (Cif)



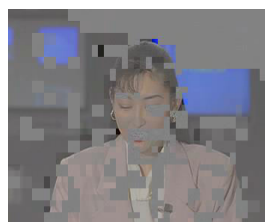
Football (Qcif)



Foreman (720p)

(a)

(b)



(c)

Figure 7-4: (a) video frame before streaming (b) received video by client at 6% Packet loss Rate. (c) The effect of delay at 50 ms.

Table 7-6: Using defaults parameter of H.264 encoder.

	Foreman(Qcif)	Akiyo(Cif)	Football(Qcif)	Foreman(720p)
PSNR	36.072 dB	40.523 dB	33.396 dB	42.038 dB
0 %	21.66 dB	32.01 dB	36.56 dB	23.36 dB
1 %	14.03 dB	23.18 dB	13.64 dB	15.38 dB

For comparison purposes of the proposed method the results obtained in Table 7-6, shows the streaming of videos encoded with default parameters of H264 encoder in JM reference software. Packet loss rate % was introduced and how packet loss impacts the video quality PSNR (dB). In particular the proposed method in this chapter performed generally better in all experiments. We see that the proposed approach has a slightly improved compared to the evaluation of the results used in default parameter.

7.4 Summary

In this chapter, we have integrated several tools to create a real network between a client and a server and have transmitted video streams compressed by an optimally configured H.264 video CODEC as per the research proposed in Chapters 4 and 5. A packet loss and network delay simulation has been used to introduce network losses as in a real network and the impact of such loss and delay on video sequences having different characteristics of content has been investigated. The research proposed has enabled us to determine the impact of packet loss and network delay on the transmission / streaming of optimally coded video. It has been shown that despite the presence of error concealment algorithms the impact on video quality is significant as high rates of packet loss can severely impact video quality of video's specifically having fast motion characteristics. Videos having larger content/object motion between frames result in a higher amounts of bits being allocated to code the motion vectors which are vital in a video's reconstruction at the decoder end. The loss of such content thus creates significant artefacts that error concealment algorithms struggle to correct.

We have observed the impact of packet losses on the transmission of video streaming through end to end devices. Due to different frame sizes and different percentages of losses and delays, a normal UDP channel has been simulated and the losses have been introduced to the network using Clumsy which is a tool that can

introduce Packet Drop with randomly discard packets and hold the packets to emulate network delay in both local and Wide Area Networks.

When amount of packet lost is introduced the result shows that Quality or PSNR of the video decreased compared to the original streamed video. In the video playback we were able to observe that, the videos were poor at different percentages of losses.

It is also observed that the low motion sequence loses quality survive due to similar frames, the better the quality of videos received.

In the fact videos are segmented into many smaller packets during transmission while the packets are dropped randomly, there is high possibility that the video suffer more losses of the very important I frames. The received video will have much more artifacts with poor PSNR. The contribution of this chapter will help in operations such as, high quality videoconferencing and online football videos.

Chapter 8

Conclusions and Future Work

In this chapter we provide a summary and conclusions of the key research outcomes of this thesis, and describe how these findings have contributed to meeting the research objectives. After presenting the summary and conclusions in section 8.1, a number of ideas to further improve the contributions of the research presented in this thesis are presented and discussed in section 8.2. The proposed further work can significantly improve the outcomes of the research presented in this thesis and will enable wider practical impact in the future.

8.1 Conclusions

The research presented in this thesis has investigated a number of original contributions made to the research area of optimisation of video codecs. In particular the optimisations have been performed on multi-objectives, to represent a generalized framework that can be applied to any video CODEC and provides a useful contribution to the video coding community who are often faced with the dilemma of selecting values for a large number of coding parameters with the intention of obtaining optimal performance of the CODEC under multiple performance constraints.

The following describes the contributions of the research conducted within the research context of this thesis:

- The thesis proposes the use of an approach to model the performance of a video CODEC based on the codecs significant coding parameters and then subsequently use these models to optimise the CODECs performance under multiple operational constraints/objectives. A framework for the above is proposed and a claim is made that this framework can be applied to any video CODEC in practice in coding any video. Therefore research is conducted to justify the use of the framework in the optimisation of CODECs of the most widely used video coding standard at present, H.264 and the latest video coding standard H.265.

- In Chapter-4 a novel approach to determining the significant coding parameters of a H.264 video CODEC using linear regression was proposed. As compared to the experimental and trial-and-error approaches used in previous literature to establish these significant coding parameters the proposed novel approach ensures a scientific methodology that allows the detailed analysis of operational constraints of specific coding algorithms implemented with the standardised CODECs as implemented. The chapter subsequently used these coding parameters to model the CODECs performance, in terms of coded bit rate, reconstructed image/video quality and encoder/decoder computational cost. Five different machine learning approaches were used for this purpose concluding that more advanced machine learning approaches such as the ensemble learning algorithms REPTree has the ability to model performance more accurately than simple linear regression based modelling. However the simplicity and applicability of Linear Regression and the closeness of obtained prediction accuracies to the highest possible accuracy resulted in a decision being made to use linear regression as the modelling approach that will be recommended for the proposed framework.
- Chapter-5 proposed the use of a Genetic Algorithm based approach to optimise the H.264 CODEC modelled in Chapter-4. Bit-rate vs. PSNR, PSNR vs. computational cost, Computational cost vs.-bit rate optimisations were carried out under further operational constraints of the parameters used. Optimal performance graphs, i.e. pareto-optimal-curves were generated leading to identifying a number of exemplar optimal operational points under the given constraints and the corresponding parameter values were established. Thus the outcome of the optimisations conducted in Chapter-5 is a collection of configurations where one is aware of the coding parameter that needs to be set in order to obtain optimal CODEC performance.
- In Chapter-6 the use of the framework tested on H.264 in chapters 4 and 5 in the modelling and optimisation of a H.265 CODEC was presented. The research conducted and the result obtained justifies the claim that the proposed parameter based multi-objective optimisation framework can be used in conjunction with and video CODEC. Due to the nature of the H.265 video CODEC's design it was found out that different set of parameters

operate as the significant coding parameters. Further due to the need of testing HD video, the computational power needed to carry out the research had to be extended beyond the use of a PC and into the use of High Performance Computing facilities.

- In Chapter 4-6 the video CODEC optimisations carried out were limited to the encoder and decoder only. However in practice encoded video is transmitted via lossy channels and the decoded video reconstructed will be subjected to packet loss, network delays etc. which will have an impact of the reconstructed image/video quality. In appreciation of this fact in Chapter-7 a framework for the transmission and network simulation of packet loss and delay were presented by integrating existing technological components. This simulation system was then used to simulate packet loss and network delay. The impact they have on the reconstructed video quality of videos having different motion characteristic were investigated and reasoning for the artefacts created were given. It was conclude that for a video streaming application and end-to-end optimal configuration parameter selection is impossible in practice as the packet-loss and delay results from external variable factors such as network congestion. Therefore the investigations carried out in this chapter was of practical importance to research community who could benefit from the work presented in chapters 4-6 by understanding the impact that optimally coded video will have when transmitted over real networks.

The above contributions have resulted in the publication of two papers (see section 1.2). A further publication is planned in the investigation of the impact of network packet loss and delays on optimally coded video.

8.2 Future work

The research conducted in this thesis was somewhat constrained by the limitations of the software implementations used for the standard functions such as those used for modelling the codec performance using machine learning and optimisation.

It was mentioned in Chapter 5 and in Chapter 6 that the MATLAB's implementation of the GA based optimisation algorithm deals with all optimisation problems as a minimisation problem. This required the negation of functions that required maximisation such as PSNR in the experiments conducted. As a result the PSNR bit-rate and CPU time values obtained are negative, making it difficult to interpret the results. A new design and implementation of this algorithm that resolves the above constrained can be recommended as future work. However this specific limitation has not had any significant overall impact of the concept being proposed in this thesis.

A further limitation of the MATLAB's multi-objective-optimisation function as implemented is that it was only able to jointly optimise two functions with multiple operational/parameter constraints. In practice this does not need to be the case and one should be able to jointly optimise more than two fitness functions. As a result of the above limitations the optimisations that were carried out were 'dual' in nature having multiple constraints in the parameter setting ranges.

In the optimisation of the H.264 CODEC our extended experiments results demonstrated that even with the CODEC's rate-control being turned off, the set bit-rate had an impact on the PSNR. This can be attributed to secondary reasons such as coding mode decisions etc. that may affect quality. Therefore in the modelling of the H.264 CODEC in Chapter-4 both PSNR and bit-rate have been attempted to be used as modelling parameters. This has only however led to the presence of bit-rate as an independent parameter in the PSNR fitness function, but not PSNR in the bit-rate fitness function. Realistically under a proper implementation of a CODEC when rate control is off, one should be able to assume that bit-rate is not a parameter that can be set and hence should not be used in the modelling. We have conducted further work since this decision and found out that if the bit-rate is not used the accuracy of the modelled PSNR slightly reduces. It can be recommended that therefore bit-rate be removed from the modelling in the future.

In extended research conducted within the research context of this thesis, we have also investigated the use of OPNET in modelling real network conditions such as packet-loss and delay. Unfortunately within the context of the research this resource that incurs licensing cost was not available to support the research conducted. In the future the proposed framework for simulation in Chapter-7 can be replaced by OPNER allowing a more comprehensive investigation to be carried out.

References

- [1] A. Bovik, *The Essential Guide to Video Processing*. 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA 525 B Street, Suite 1900, San Diego, California 92101-4495, USA 84 Theobald's Road, London WC1X 8RR, UK: Academic Press publications, 2009.
- [2] T. S. Wiegand G. J. Bjontegaard, G. Luthra, A., "Overview of the H.264/AVC video coding standard," *Circuits Syst. Video Technol. IEEE Trans.*, vol. 13, no. 7, pp. 560–576, 2003.
- [3] I. Richardson, "An Overview of H.264 Advanced Video Coding," 2013. [Online]. Available: <http://vcodex.com>.
- [4] I. E. G. Richardson, *The H.264 advanced video compression standard*. Wiley InterScience (Online Service), 2010.
- [5] S. Wenger, "H.264/AVC over IP," *Circuits Syst. Video Technol. IEEE Trans.*, vol. 13, no. 7, pp. 645–656, 2003.
- [6] N. O. Masaki Kitahara[†] and Atsushi Shimizu, "Software H.264 Encoder Engine for Online Video Delivery Service Cost Reduction," *NTT Cyber Sp. Lab. Yokosuka-shi, 239-0847 Japan*, vol. Vol. 9, no. No. 6, 2011.
- [7] T. W. and H. S. Ralf Schäfer, "The emerging H.264/AVC standard," *Heinrich Hertz Institute, Berlin, Ger.*, 2003.
- [8] A. Communications, "An explanation of video compression techniques.," *White Pap.*, 2008.
- [9] F.-I. H. H. I. Dolby Laboratories Inc. Microsoft Corporation, "H.264/AVC Reference Software." Joint Video Team, Germany, 2009.
- [10] "Italy, June , Springer: 1-15. Dolby Laboratories Inc.," *FI H Microsoft Corp.*, vol. 2000 SRC, pp. 21–23, 2009.
- [11] G. J. Sullivan, J. Ohm, H. Woo-Jin, and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *Circuits Syst. Video Technol. IEEE Trans.*, vol. 22, no. 12, pp. 1649–1668, 2012.
- [12] D. Grois, D. Marpe, A. Mulyoff, B. Itzhaky, and O. Hadar, "Performance comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC encoders,"

- 2013 *Picture Coding Symposium (PCS)*. pp. 394–397, 2013.
- [13] V. Sze and G. J. S. Budagavi, Madhukar, *High Efficiency Video Coding (HEVC): Algorithms and Architectures*. Cambridge, USA: Integrated Circuits and Systems, 2014.
- [14] N. Narang, “What is the difference between HEVC (H.265) and H.264 (MPEG-4 AVC).” *Future Of Media, Media Concepts*, 2013.
- [15] Fraunhofer Heinrich Hertz Institute, “High Efficiency Video Coding (HEVC) | JCT-VC,” *Fraunhofer-Gesellschaft*, 2016. [Online]. Available: <https://hevc.hhi.fraunhofer.de/>. [Accessed: 18-Apr-2017].
- [16] F. Heinrich, “svn_HEVCSoftware - Revision 4885: /,” 2016. [Online]. Available: https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/. [Accessed: 18-Apr-2017].
- [17] F. Heinrich, “jctvc-a124 - Revision 4885: /.” [Online]. Available: <http://hevc.kw.bbc.co.uk/svn/jctvc-a124/>. [Accessed: 18-Apr-2017].
- [18] Microsoft, “Introducing Visual Studio,” *Microsoft*, 2008. [Online]. Available: [https://msdn.microsoft.com/en-us/library/fx6bk1f4\(v=vs.90\).aspx](https://msdn.microsoft.com/en-us/library/fx6bk1f4(v=vs.90).aspx). [Accessed: 01-May-2017].
- [19] Intel, “Intel® VTune™ Amplifier XE,” 2013. .
- [20] E. F. Mark Hall Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witte, “The WEKA Data Mining Software,” 2009. [Online]. Available: <http://www.cs.waikato.ac.nz/ml/weka/>.
- [21] M. J. Trilok Chand Sharma, “WEKA Approach for Comparative Study of Classification Algorithm.,” *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 2, no. 4, 2013.
- [22] H. S. Norman R. Draper, *Applied Regression Analysis*. Wiley, 1998.
- [23] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., 2011.
- [24] R. Pierce, “Correlation,” *Math Is Fun*, 2017. [Online]. Available: <http://www.mathsisfun.com/data/correlation.html>. [Accessed: 20-Jul-2017].
- [25] MathWorks, *Global Optimization Toolbox User’s Guide*, vol. R2013a. The MathWorks, Inc.3 Apple Hill DriveNatick, MA 01760-2098, 2013.

- [26] S. Jena, P. Patrob, and S. S. Beherac, "Multi-Objective Optimization of Design Parameters of a Shell & Tube type Heat Exchanger using Genetic Algorithm," 2013.
- [27] K. K. Mishra, A. Kumar, and A. K. Misra, "A variant of NSGA-II for solving priority based optimization problems," in *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, 2009, pp. 612–615.
- [28] O. A. Ani, H. Xu, Y. Shen, S. Liu, and K. Xue, "Modeling and multiobjective optimization of traction performance for autonomous wheeled mobile robot in rough terrain," *J. Zhejiang Univ. Sci. C*, vol. 14, no. 1, pp. 11–29, 2013.
- [29] W. Simpson, *Video Over IP, Second Edition: IPTV, Internet Video, H.264, P2P, Web TV, and Streaming: A Complete Guide to Understanding the Technology*, Second Edi. Elsevier Inc., 2008.
- [30] VideoLAN, "The cross-platform streaming solution - VideoLAN." [Online]. Available: <http://www.videolan.org/vlc/streaming.html>. [Accessed: 06-Jul-2017].
- [31] J. Klaue, "EvalVid - A Video Quality Evaluation Tool-set," 2009. [Online]. Available: <http://www2.tkn.tu-berlin.de/research/evalvid/fw.html>.
- [32] Riverbed, "University Support Center: Riverbed Modeler Aca... | Riverbed Splash," 2017. [Online]. Available: <https://splash.riverbed.com/community/product-lines/steelcentral/university-support-center/blog/2014/06/11/riverbed-modeler-academic-edition-release>. [Accessed: 06-Jul-2017].
- [33] J. Lahti, J. K. Juntunen, I. Lehtoranta, and T. D. Hamalainen, "Algorithmic optimization of H.264/AVC encoder," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, 2005, p. 3463–3466 Vol. 4.
- [34] J. Shin-Haeng, P. Jung-Wook, and K. Shin-Dug, "Optimization of Memory Management for H.264/AVC Decoder," in *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, 2006, vol. 1, pp. 65–68.
- [35] J. Chakareski and , IEEE, "P. Frossard . "Rate-distortion optimized distributed packet scheduling of multiple video streams over shared

- communication resources.,” *Multimed.*, vol. 8, no. 2, pp. 207–218, 2006.
- [36] S. Li, L. Yan, W. Feng, L. Shipeng, and G. Wen, “Complexity-Constrained H.264 Video Encoding,” *Circuits Syst. Video Technol. IEEE Trans.*, vol. 19, no. 4, pp. 477–490, 2009.
- [37] F. Al-Abri, X. Li, E. A. Edirisinghe, and C. Grecos, “A Novel Framework for Multi-objective Optimization of Video CODECs,” in *CyberWorlds, 2009. CW ’09. International Conference on*, 2009, pp. 195–202.
- [38] R. Vanam, E. A. Riskin, S. S. Hemami, and R. E. Ladner, “Distortion-Complexity Optimization of the H.264/MPEG-4 AVC Encoder using the GBFOS Algorithm,” in *Data Compression Conference, 2007. DCC ’07*, 2007, pp. 303–312.
- [39] R. Vanam, E. A. Riskin, and R. E. Ladner, “H.264/MPEG-4 AVC Encoder Parameter Selection Algorithms for Complexity Distortion Tradeoff,” in *Data Compression Conference, 2009. DCC ’09.*, 2009, pp. 372–381.
- [40] H. Zhihai, L. Yongfang, C. Lulin, I. Ahmad, and W. Dapeng, “Power-rate-distortion analysis for wireless video communication under energy constraints,” *Circuits Syst. Video Technol. IEEE Trans.*, vol. 15, no. 5, pp. 645–658, 2005.
- [41] H. Zhihai, C. Wenye, and C. Xi, “Energy Minimization of Portable Video Communication Devices Based on Power-Rate-Distortion Optimization,” *Circuits Syst. Video Technol. IEEE Trans.*, vol. 18, no. 5, pp. 596–608, 2008.
- [42] P. Wei, L. Yan, and W. Feng, “Joint Power-Distortion Optimization on Devices with MPEG-4 AVC/H.264 Codec,” in *Communications, 2006. ICC ’06. IEEE International Conference on*, 2006, vol. 1, pp. 441–446.
- [43] K. Jaemoon, K. Jungsoo, K. Giwon, and K. Chong-Min, “Power-rate-distortion modeling for energy minimization of portable video encoding devices,” in *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, 2011, pp. 1–4.
- [44] C. Fu-Chuang and H. Yi-Pin, “Rate-distortion optimization of H.264/AVC rate control with novel distortion prediction equation,” *Consum. Electron. IEEE Trans.*, vol. 57, no. 3, pp. 1264–1270, 2011.
- [45] T. A. da Fonseca and R. L. De Queiroz, “Complexity-constrained rate-

- distortion optimization for h.264/avc video coding,” in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, 2011, pp. 2909–2912.
- [46] S. Weiwei, F. Yibo, H. Leilei, L. Jiali, and Z. Xiaoyang, “A hardware-friendly method for rate-distortion optimization of HEVC intra coding,” in *VLSI Design, Automation and Test (VLSI-DAT), 2014 International Symposium on*, 2014, pp. 1–4.
- [47] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, England: John Wiley & Sons, 2001.
- [48] K. Deb, , Kanpur Indian, and , PIN, “Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction.,” *Technol. Kanpur*, vol. 208016, 2011.
- [49] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *Evol. Comput. IEEE Trans.*, vol. 6, no. 2, pp. 182–197, 2002.
- [50] K. D. Srinivas, N, “Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms.,” *Evol. Comput.*, vol. 2, no. 3, pp. 221–248, 1994.
- [51] and A. S. Masaki Kitahara[†], N. O., “Software H.264 Encoder Engine for Online Video Delivery Service Cost Reduction.” NTT Cyber Space Laboratories,” 2011. [Online]. Available: NTT Cyber Space Laboratories. [Accessed: 01-Jan-2014].
- [52] K. V. S. Swaroop and K. R. Rao, “Performance analysis and comparison of JM 15.1 and Intel IPP H. 264 encoder and decoder,” in *System Theory (SSST), 2010 42nd Southeastern Symposium on*, 2010, pp. 371–375.
- [53] S. J. K. W. and C. Lee, “H264AVC entropy decoder complexity analysis and its applications,” *J. Vis. Commun. Image Represent.*, vol. 22, no. 1 SRC-GoogleScholar FG-0, pp. 61–72, 2011.
- [54] F. H. Seitner, R. M. Schreier, M. Bleyer, and M. Gelautz, “A high-level simulator for the H. 264/AVC decoding process in multi-core systems,” in *Electronic Imaging 2008*, 2008, p. 682105.
- [55] B. Boyadjis, C. Bergeron, B. Pesquet-Popescu, and F. Dufaux, “Extended Selective Encryption of H.264/AVC (CABAC)- and HEVC-Encoded Video Streams,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 4, pp. 892–

906, Apr. 2017.

- [56] D. Marpe, T. Wiegand, and G. J. Sullivan, "The H.264/MPEG4 advanced video coding standard and its applications," *IEEE Communications Magazine*, vol. 44, no. 8. pp. 134–143, 2006.
- [57] G. J. Sullivan, J. Ohm, W. Han, and T. Wiegand, "Overview of the High Efficiency Video Coding," *Ieee Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, 2012.
- [58] W. Shen, Y. Fan, L. Huang, J. Li, and X. Zeng, "A hardware-friendly method for rate-distortion optimization of HEVC intra coding," *Technical Papers of 2014 International Symposium on VLSI Design, Automation and Test*. pp. 1–4, 2014.
- [59] B. Bross *et al.*, "HEVC real-time decoding," in *SPIE Optical Engineering+ Applications*, 2013, p. 88561R–88561R.
- [60] A. F. Eldeken, M. M. Fouad, G. I. Salama, and A. A. Youssif, "A New Implementation of High Resolution Video Encoding Using the HEVC Standard," in *Intelligent Systems in Cybernetics and Automation Theory*, Springer, 2015, pp. 163–172.
- [61] R. I. Chernyak, "Analysis of the Intra Predictions in H. 265/HEVC," *Appl. Math. Sci.*, vol. 8, no. 148, pp. 7389–7408, 2014.
- [62] J. R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards-including high efficiency video coding (HEVC)," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1669–1684, 2012.
- [63] L. Gao, S. Dong, W. Wang, R. Wang, and W. Gao, "Fast intra mode decision algorithm based on refinement in HEVC," *Proc. - IEEE Int. Symp. Circuits Syst.*, vol. 2015–July, pp. 517–520, 2015.
- [64] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "HEVC complexity and implementation analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1685–1696, 2012.
- [65] B. Bross *et al.*, "HEVC performance and complexity for 4K video," *Proc. 2013 IEEE 3rd Int. Conf. Consum. Electron. - Berlin, ICCE-Berlin 2013*, pp. 44–47, 2013.

- [66] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, V. George, and T. Schierl, “Improving the parallelization efficiency of HEVC decoding,” *Proc. - Int. Conf. Image Process. ICIP*, pp. 213–216, 2012.
- [67] M. P. Sharabayko and N. G. Markov, “Iterative intra prediction search for H.265/HEVC,” *2013 Int. Sib. Conf. Control Commun. SIBCON 2013 - Proc.*, 2013.
- [68] M. Alvarez-Mesa, C. C. Chi, B. Juurlink, V. George, and T. Schierl, “Parallel video decoding in the emerging HEVC standard,” *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 1545–1548, 2012.
- [69] Q. Huangyuan, L. Song, Z. Luo, X. Wang, and Y. Zhao, “Performance evaluation of H.265/MPEG-HEVC encoders for 4K video sequences,” *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*. pp. 1–8, 2014.
- [70] M. P. Sharabayko, “Next generation video codecs: HEVC, VP9 and DAALA,” *XI Int. Theor. Pract. Conf.*, vol. 2560, no. 1600, p. 30, 2013.
- [71] T. Gilling, *The Stream Tone : the future of personal computing?* Troubador Publishing Ltd, 2017.
- [72] P. Hanhart, M. Rerabek, F. De Simone, and T. Ebrahimi, “Subjective quality evaluation of the upcoming HEVC video compression standard,” in *Proc. SPIE*, 2012, vol. 8499, p. 84990V.
- [73] F. Pescador, M. Chavarrías, M. Garrido, J. Malagón, and C. Sanz, “Real-time HEVC decoding with OpenHEVC and OpenMP,” in *Consumer Electronics (ICCE), 2017 IEEE International Conference on*, 2017, pp. 370–371.
- [74] G. F. Escribano, R. Jillani, C. Holder, H. Kalva, J. L. M. Martinez, and P. Cuenca, “Video Encoding and Transcoding Using Machine Learning,” in *Proceedings of the 9th International Workshop on Multimedia Data Mining: Held in Conjunction with the ACM SIGKDD 2008*, 2008, pp. 53–62.
- [75] T. G. Dietterich and , MCS, “Ensemble methods in machine learning.,” *Mult. Classif. Syst. First Int. Work. Italy June Springer 115*, vol. 2000, pp. 21–23, 2000.
- [76] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, “On combining classifiers,”

- Pattern Anal. Mach. Intell. IEEE Trans.*, vol. 20, no. 3, pp. 226–239, 1998.
- [77] Q. Dai and R. Lehnert, “Impact of Packet Loss on the Perceived Video Quality,” in *2010 2nd International Conference on Evolving Internet*, 2010, pp. 206–209.
- [78] S. Wenger, “H.264/AVC over IP,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 645–656, 2003.
- [79] F. H. R. Fitzek, M. Zorzi, P. Seeling, and M. Reisslein, “Video and audio trace files of pre-encoded video content for network performance measurements,” *Int. Symp. VLSI Technol. Syst. Appl. Proc. Tech. Pap. (IEEE Cat. No.03TH8672)*, pp. 245–250, 2004.
- [80] Arizona State University, “Video Traces for Network Performance Evaluation,” 2004. [Online]. Available: <http://trace.eas.asu.edu/>. [Accessed: 10-Apr-2017].
- [81] T. Ozcelebi, M. O. Sunay, M. Tekalp, and M. R. Civanlar, “Cross-layer optimized rate adaptation and scheduling for multiple-user wireless video streaming,” *Sel. Areas Commun. IEEE J.*, vol. 25, no. 4, pp. 760–769, 2007.
- [82] T. Ozcelebi, A. M. Tekalp, and M. R. Civanlar, “Delay-Distortion Optimization for Content-Adaptive Video Streaming,” *Multimedia, IEEE Trans.*, vol. 9, no. 4, pp. 826–836, 2007.
- [83] C. H. Ke, C. K. Shieh, W. S. Hwang, and A. Ziviani, “An evaluation framework for more realistic simulations of MPEG video transmission,” *J. Inf. Sci. Eng.*, vol. 24, no. 2, pp. 425–440, 2008.
- [84] C. Yu, C. Ke, R. Chen, and C. Shieh, “MyEvalvid _ RTP : a Evaluation Framework for More Realistic Simulations of Multimedia Transmission,” *Int. J.*, vol. 2, no. 2, pp. 21–32, 2008.
- [85] F. Van Der Schueren, J. Doggen, and V. Der Schueren, “Design and Simulation of a H.264 AVC Video Streaming Model,” *Proc. Third Eur. Conf. Use Mod. Inf. Commun. Technol. ECUMICT 2008*, 2008.
- [86] W. Hrudey and L. Trajković, “Streaming Video Content Over IEEE 802.16 / WiMAX Broadband Access,” *OPNETWORK 2008*, 2008.
- [87] and M. A. Abdulghani, Amir M. Arshad, “H.263 Video Transmission in Wireless Local Area Networks using Opnet,” in *ICWN*, 2006, pp. 423–429.

- [88] and L. T. Rajvir Gill, Tanjila Farah, “Comparison of WiMAX and ADSL by Streaming Audio and Video Content,” 2011, vol. 301136804, pp. 1–38.
- [89] L. Sendrei, J. Valiska, and S. Marchevský, “H.264 video transmission over WLAN in OPNET Modeller,” *J. Electr. Eng.*, vol. 64, no. 2, pp. 112–117, 2013.
- [90] T. . Al Hadhrami, J. . Nightingale, Q. . Wang, C. . Grecos, and N. . Kehtarnavaz, “A simulator tool set for evaluating HEVC/SHVC streaming,” *Proc. SPIE - Int. Soc. Opt. Eng.*, vol. 9400, pp. 1–10, 2015.
- [91] D. J. Mandal and A. P. S. Ghimire, “Effect of Packet Drop and Jitter on Perceived Video Quality for Various Encoded Video over Streaming Network,” *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 4, no. 8, pp. 7–11, 2016.
- [92] and I. H. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, “The Mining Software An Update.” The WEKA Data Mining Software: An Update. SIGKDD Explorations, 2009.
- [93] V. Coders, “Video Coders - Research and Development on Coding Techniques for H.264/AVC, MPEG-2 and HVC,” 2010. [Online]. Available: <http://videocoders.com/>.
- [94] H. Group, “Database: Image & Video Clips,” 2006. [Online]. Available: http://see.xidian.edu.cn/vipsl/database_Video.html#video.
- [95] K. S. and G. S. Alexis Michael Tourapis, Athanasios Leontaris, “H. 264/14496-10 AVC reference software manual,” 2009.
- [96] MathWorks, “Find Pareto front of multiple fitness functions using genetic algorithm - MATLAB gamultiobj,” 2007. [Online]. Available: <https://uk.mathworks.com/help/gads/gamultiobj.html>. [Accessed: 17-Sep-2017].
- [97] Jagtt, “clumsy, an utility for simulating broken network for Windows Vista / Windows 7 and above.” 2016.
- [98] Riverbed, “OPNET Technologies – Network Simulator | Riverbed,” 2017. [Online]. Available: <https://www.riverbed.com/gb/products/steelcentral/opnet.html>. [Accessed: 06-Jul-2017].

Appendices

Appendix A: JM reference encoder / decoder configuration file.

A.1. Encoder configuration file.

```
#####
#####
# Files
#####
#####
InputFile      = "foreman_qcif.yuv" # Input sequence
InputHeaderLength = 0 # If the inputfile has a header, state it's length in byte here
StartFrame     = 0 # Start frame for encoding. (0-N)
FramesToBeEncoded = 30 # Number of frames to be coded
FrameRate      = 30.0 # Frame Rate per second (0.1-100.0)
Enable32PullDown = 0 # Enable 'hard' 3:2 pulldown (modifying the input data)
                # 0 = disabled
                # 1 = A, B, Bt|Cb, Ct|Db, D
                # 2 = A, B, C, Ct|Db, D
SEIVUI32PullDown = 0 # Enable 3:2 pulldown through VUI and SEI metadata signaling. Five
methods are supported:
                # 0 = disabled
                # 1 = A, Bt|Bb, Bt|Cb, Ct|Cb, D
                # 2 = A, B, C, C, D
                # 3 = At|Ab, Bt|Bb, Bt|Cb, Ct|Cb, Dt|Db
                # 4 = A, Bt|Bb, Bt|Cb, Ct|Db, Dt|Db
                # 5 = At|Ab, Bt|Bb, Bt|Cb, Ct|Db, Dt|Db

SourceWidth    = 176 # Source frame width
SourceHeight   = 144 # Source frame height
SourceResize   = 0 # Resize source size for output
OutputWidth    = 176 # Output frame width
OutputHeight   = 144 # Output frame height
ProcessInput   = 0 # Filter Input Sequence
Interleaved    = 0 # 0: Planar input, 1: Packed input
PixelFormat    = 0 # Pixel Format for 422 packed inputs
                # 0: UYVY
                # 1: YUY2/YUYV
                # 2: YVYU
                # 3: BGR (Unsupported)
                # 4: V210 (Video Clarity)

StandardRange  = 0 # 0: Standard range 1: Full range (RGB input)
VideoCode      = 1 # Video codes for RGB ==> YUV conversions
                # 0 = NULL,
                # 1 = ITU_REC709,
                # 2 = CCIR_601,
                # 3 = FCC,
                # 4 = ITU_REC624BG,
                # 5 = SMPTE_170M,
                # 6 = SMPTE_240M,
                # 7 = SMPTE_260M,
                # 8 = ITU_REC709_EXACT

TraceFile      = "trace_enc.txt" # Trace file
ReconFile      = "test_rec.yuv" # Reconstruction YUV file
OutputFile     = "test.264" # Bitstream
StatsFile      = "stats.dat" # Coding statistics file
```

```

NumberOfViews      = 1          # Number of views to encode (1=1 view, 2=2 views)
View1ConfigFile   = "encoder_view1.cfg" # Config file name for second view

#####
#####

# Encoder Control
#####
#####
Grayscale         = 0 # Encode in grayscale (Currently only works for 8 bit YUV 420 input)
ProfileIDC        = 100 # Profile IDC (66=baseline, 77=main, 88=extended; FREXT Profiles: 100=High,
110=High 10, 122=High 4:2:2, 244=High 4:4:4, 44=CAVLC 4:4:4 Intra, 118=Multiview High Profile,
128=Stereo High Profile)
IntraProfile      = 0 # Activate Intra Profile for FRExt (0: false, 1: true)
                  # (e.g. ProfileIDC=110, IntraProfile=1 => High 10 Intra Profile)
LevelIDC          = 40 # Level IDC (e.g. 20 = level 2.0)

IntraPeriod       = 0 # Period of I-pictures (0=only first)
IDRPeriod         = 0 # Period of IDR pictures (0=only first)
AdaptiveIntraPeriod = 1 # Adaptive intra period
AdaptiveIDRPeriod  = 0 # Adaptive IDR period
IntraDelay        = 0 # Intra (IDR) picture delay (i.e. coding structure of PPIPPP...)
EnableIDRGOP      = 0 # Support for IDR closed GOPs (0: disabled, 1: enabled)
EnableOpenGOP     = 0 # Support for open GOPs (0: disabled, 1: enabled)
QPISlice          = 17 # Quant. param for I Slices (0-51)
QPPSlice          = 17 # Quant. param for P Slices (0-51)

#####
#####
# Search Range Restriction / RD Optimization
#####
#####

RDOOptimization   = 1 # rdo-optimized mode decision
                  # 0: RD-off (Low complexity mode)
                  # 1: RD-on (High complexity mode)
                  # 2: RD-on (Fast high complexity mode - not work in FREX)

```

A.2. Decoder configuration file.

This is a file containing input parameters to the JVT H.264/AVC decoder.

```

# The text line following each parameter is discarded by the decoder.
#
# For bug reporting and known issues see:
# https://ipbt.hhi.fraunhofer.de
#
# New Input File Format is as follows
# <ParameterName> = <ParameterValue> # Comment
#
#####
#####
# Files
#####
#####
InputFile         = "test.264" # H.264/AVC coded bitstream
OutputFile        = "test_dec.yuv" # Output file, YUV/RGB
RefFile           = "test_rec.yuv" # Ref sequence (for SNR)
WriteUV           = 1 # Write 4:2:0 chroma components for monochrome streams
FileFormat        = 0 # NAL mode (0=Annex B, 1: RTP packets)
RefOffset         = 0 # SNR computation offset
POCScale          = 2 # Poc Scale (1 or 2)
#####
#####
# HRD parameters

```

```

#####
#####
#R_decoder      = 500000      # Rate_Decoder
#B_decoder      = 104000      # B_decoder
#F_decoder      = 73000       # F_decoder
#LeakyBucketParamFile = "leakybucketparam.cfg" # LeakyBucket Params
#####
#####
# decoder control parameters
#####
#####
DisplayDecParams = 0          # 1: Display parameters;
ConcealMode      = 0          # Err Concealment(0:Off,1:Frame Copy,2:Motion Copy)
RefPOCGap       = 2          # Reference POC gap (2: IPP (Default), 4: IbP / IpP)
POCGap          = 2          # POC gap (2: IPP /IbP/IpP (Default), 4: IPP with frame skip = 1 etc.)
Silent          = 0          # Silent decode
IntraProfileDeblocking = 1    # Enable Deblocking filter in intra only profiles (0=disable,
1=filter according to SPS parameters)
DecFrmNum       = 0          # Number of frames to be decoded (-n)
#####
#####
# MVC decoding parameters
#####
#####
DecodeAllLayers = 0          # Decode all views (-mpr)

```

Appendix B: Selected parameters for encoder / decoder.

Table B . 1: Selected set of parameters for Claire video sequence.

IntraPeriod	SearchRange	QP	NRFrames	PSNR	Bit-rate	CPU	CPU/Decoder
0	16	17	2	47.437	162.01	45.551	0.329
0	16	17	5	47.48	133.98	51.461	0.299
0	16	17	8	47.495	133.54	60.925	0.301
0	16	49	2	27.086	5.49	34.866	0.192
0	16	49	5	27.081	5.59	47.959	0.19
0	16	49	8	27.114	5.73	57.967	0.186
0	32	17	2	47.458	161.63	40.422	0.33
0	32	17	5	47.48	132.3	62.217	0.311
0	32	17	8	47.488	132.13	66.532	0.313
0	32	49	2	27.086	5.49	40.79	0.205
0	32	49	5	27.081	5.59	52.359	0.207
0	32	49	8	27.114	5.73	59.203	0.189
5	16	17	2	48.308	292.37	37.451	0.385
5	16	17	5	48.387	258.33	46.761	0.361
5	16	17	8	48.385	258.32	55.405	0.359
5	16	49	2	27.222	14.09	32.66	0.239
5	16	49	5	27.402	14.22	44.288	0.231
5	16	49	8	27.412	14.29	53.914	0.229

5	32	17	2	48.308	292.37	37.722	0.375
5	32	17	5	48.387	258.33	47.217	0.354
5	32	17	8	48.385	258.32	55.22	0.379
5	32	49	2	27.222	14.09	32.612	0.231
5	32	49	5	27.402	14.22	44.113	0.228
5	32	49	8	27.412	14.29	53.695	0.235
8	16	17	2	47.793	217.02	38.537	0.332
8	16	17	5	47.846	189.11	48.76	0.313
8	16	17	8	47.859	189.08	57.1	0.312
8	16	49	2	27.282	10.54	33.452	0.219
8	16	49	5	27.483	10.62	45.507	0.216
8	16	49	8	27.483	10.7	55.077	0.214
8	32	17	2	47.786	216.93	38.664	0.349
8	32	17	5	47.848	189.16	50.772	0.353
8	32	17	8	47.861	189.12	64.002	0.395
8	32	49	2	27.282	10.54	34.729	0.221
8	32	49	5	27.483	10.62	52.814	0.225
8	32	49	8	27.483	10.7	58.264	0.227

Table B. 2: Selected set of parameters for Coastguard video sequence.

IntraPeriod	SearchRange	QP	NRFrames	PSNR	Bit-rate	CPU	CPU/ Decoder
0	16	17	2	43.418	979.02	64.889	0.662
0	16	17	5	43.27	784.34	88.765	0.624
0	16	17	8	43.286	785.94	104.393	0.612
0	16	49	2	24.189	5.83	40.145	0.205
0	16	49	5	24.291	5.82	51.923	0.194
0	16	49	8	24.294	5.93	63.185	0.191
0	32	17	2	43.436	980.48	64.994	0.65
0	32	17	5	43.252	781.88	83.298	0.617
0	32	17	8	43.272	783.7	100.209	0.603
0	32	49	2	24.241	5.9	37.427	0.191
0	32	49	5	24.322	5.88	52.426	0.19
0	32	49	8	24.317	5.91	64.159	0.188
5	16	17	2	44.999	1303.7	60.007	0.735
5	16	17	5	44.782	1050.44	74.499	0.676
5	16	17	8	44.772	1048.75	89.988	0.678
5	16	49	2	24.509	10.35	34.322	0.227
5	16	49	5	24.669	10.36	47.052	0.224
5	16	49	8	24.671	10.44	57.41	0.225
5	32	17	2	32.202	508.26	34.85	0.337
5	32	17	5	33.264	507.62	46.791	0.33
5	32	17	8	33.264	507.66	57.392	0.33

5	32	49	2	24.509	10.35	34.838	0.228
5	32	49	5	24.673	10.37	47.723	0.225
5	32	49	8	24.67	10.43	58.284	0.253
8	16	17	2	43.662	1041.16	61.839	0.713
8	16	17	5	43.501	836.26	77.872	0.608
8	16	17	8	43.49	835.59	93.222	0.631
8	16	49	2	24.452	8.82	36.108	0.216
8	16	49	5	24.741	8.62	50.437	0.238
8	16	49	8	24.743	8.73	61.173	0.226
8	32	17	2	43.67	1042.05	62.295	0.656
8	32	17	5	43.503	836.23	78.095	0.615
8	32	17	8	43.49	836.42	94.709	0.607
8	32	49	2	24.452	8.82	36.389	0.219
8	32	49	5	24.739	8.62	49.087	0.217
8	32	49	8	24.741	8.71	60.517	0.226

Table B. 3: Selected set of parameters for Football video sequence.

IntraPeriod	SearchRange	QP	NRFrames	PSNR	Bit-rate	CPU	CPU/ Decoder
0	16	17	2	44.109	2242.07	89.723	0.965
0	16	17	5	44.081	2085.56	109.924	0.96
0	16	17	8	44.052	2081.78	128.955	0.937
0	16	49	2	19.663	21.35	47.255	0.247
0	16	49	5	20.043	19.32	66.317	0.237
0	16	49	8	20.061	19.43	80.083	0.231
0	32	17	2	44.076	2237.71	91.363	0.958
0	32	17	5	44.06	2081.21	112.022	0.938
0	32	17	8	44.051	2080.76	131.97	0.934
0	32	49	2	19.712	21.38	47.562	0.245
0	32	49	5	20.009	18.79	65.294	0.231
0	32	49	8	20.015	18.96	80.359	0.229
5	16	17	2	45.49	2358.85	98.877	0.979
5	16	17	5	45.49	2358.85	98.289	0.977
5	16	17	8	44.159	2104.87	103.437	0.952
5	16	49	2	20.398	31.64	43.198	0.291
5	16	49	5	20.71	29.05	58.097	0.276
5	16	49	8	20.699	29.01	71.695	0.287
5	32	17	2	45.557	2552.46	82.586	1.03
5	32	17	5	45.48	2357.46	101.71	0.997
5	32	17	8	45.475	2356.23	119.861	0.984
5	32	49	2	20.386	31.51	44.104	0.29
5	32	49	5	20.713	29.03	60.011	0.284
5	32	49	8	20.709	29.02	73.532	0.271
8	16	17	2	44.154	2261.45	85.64	0.96

8	16	17	5	44.159	2104.87	104.012	0.942
8	16	17	8	44.156	2104.7	121.461	0.945
8	16	49	2	19.774	24.93	45.184	0.262
8	16	49	5	20.167	22.02	60.455	0.248
8	16	49	8	20.151	21.98	73.405	0.244
8	32	17	2	44.139	2260.79	87.457	0.96
8	32	17	5	44.166	2106.04	105.85	0.94
8	32	17	8	44.144	2102.5	124.775	0.947
8	32	49	2	19.779	24.46	45.784	0.259
8	32	49	5	20.174	21.92	61.115	0.243
8	32	49	8	20.158	21.92	75.706	0.244

Table B. 4: Selected set of parameters for Foreman video sequence.

IntraPeriod	SearchRange	QP	NRFrames	PSNR	Bit-rate	CPU	CPU/Decoder
0	16	17	2	44.606	547.62	61.627	0.659
0	16	17	5	44.635	473.74	75.058	0.659
0	16	17	8	44.636	471.7	84.624	0.639
0	16	49	2	22.806	9.63	41.619	0.209
0	16	49	5	23.384	9.09	57.455	0.205
0	16	49	8	23.382	9.28	70.711	0.203
0	32	17	2	44.62	547.1	61.189	0.659
0	32	17	5	44.688	472.77	74.527	0.635
0	32	17	8	44.691	471.59	83.665	0.628
0	32	49	2	22.871	9.98	41.24	0.223
0	32	49	5	23.449	9.21	57.3	0.207
0	32	49	8	23.405	9.38	71.303	0.205
5	16	17	2	45.751	819.97	55.727	0.741
5	16	17	5	45.819	712.66	69.253	0.721
5	16	17	8	45.822	714.2	75.729	0.729
5	16	49	2	23.53	16.77	37.336	0.243
5	16	49	5	23.881	16.44	53.122	0.237
5	16	49	8	23.881	16.51	63.869	0.227
5	32	17	2	45.764	820.33	69.276	0.804
5	32	17	5	45.83	714.45	80.19	0.857
5	32	17	8	45.825	714.3	86.417	0.792
5	32	49	2	23.531	16.78	37.594	0.236
5	32	49	5	23.881	16.44	51.472	0.227
5	32	49	8	23.881	16.5	64.64	0.229
8	16	17	2	44.818	618.74	57.825	0.656
8	16	17	5	44.906	544.33	69.269	0.646
8	16	17	8	44.899	540.87	77.812	0.622
8	16	49	2	22.955	13.09	39.142	0.221
8	16	49	5	23.532	12.67	52.936	0.213

8	16	49	8	23.54	12.73	65.502	0.214
8	32	17	2	44.821	618.66	58.211	0.658
8	32	17	5	44.91	544.98	70.837	0.637
8	32	17	8	44.902	542.31	78.66	0.626
8	32	49	2	22.958	13.12	39.35	0.232
8	32	49	5	23.532	12.67	53.45	0.213
8	32	49	8	23.54	12.73	66.653	0.214

Table B. 5: Selected set of parameters for Mobile video sequence.

IntraPeriod	SearchRange	QP	NRFrames	PSNR	Bit-rate	CPU	CPU/ Decoder
0	16	17	2	43.964	1595.79	70.647	0.943
0	16	17	5	43.84	1388.74	88.379	1.07
0	16	17	8	43.836	1384.42	93.805	0.879
0	16	49	2	19.27	17.37	46.491	0.253
0	16	49	5	19.623	16.71	64.464	0.233
0	16	49	8	19.632	17.03	80.494	0.232
0	32	17	2	43.97	1596.36	70.257	0.936
0	32	17	5	43.837	1388.67	84.653	0.87
0	32	17	8	43.833	1383.71	94.541	0.897
0	32	49	2	19.274	17.41	47.015	0.25
0	32	49	5	19.604	16.75	64.902	0.232
0	32	49	8	19.603	17.17	81.148	0.235
5	16	17	2	45.561	2160.9	65.138	1.013
5	16	17	5	45.441	1939.38	77.751	0.957
5	16	17	8	45.452	1933.91	89.416	0.958
5	16	49	2	19.475	44.22	42.074	0.262
5	16	49	5	19.719	43.69	56.937	0.249
5	16	49	8	19.718	43.74	71.303	0.251
5	32	17	2	45.563	2161.22	65.603	1.02
5	32	17	5	45.442	1939.3	78.23	0.952
5	32	17	8	45.451	1933.65	87.185	0.95
5	32	49	2	19.475	44.22	41.903	0.275
5	32	49	5	19.715	43.67	63.003	0.287
5	32	49	8	19.718	43.74	85.226	0.272
8	16	17	2	44.085	1762.29	79.206	1.056
8	16	17	5	43.931	1553.74	86.32	0.973
8	16	17	8	43.93	1550.54	101.151	0.944
8	16	49	2	19.267	32.94	50.098	0.328
8	16	49	5	19.637	32.33	63.053	0.272
8	16	49	8	19.643	32.62	78.31	0.27
8	32	17	2	44.08	1762.55	71.846	0.996
8	32	17	5	43.931	1554.45	85.819	0.972
8	32	17	8	43.946	1550.86	96.688	1.031

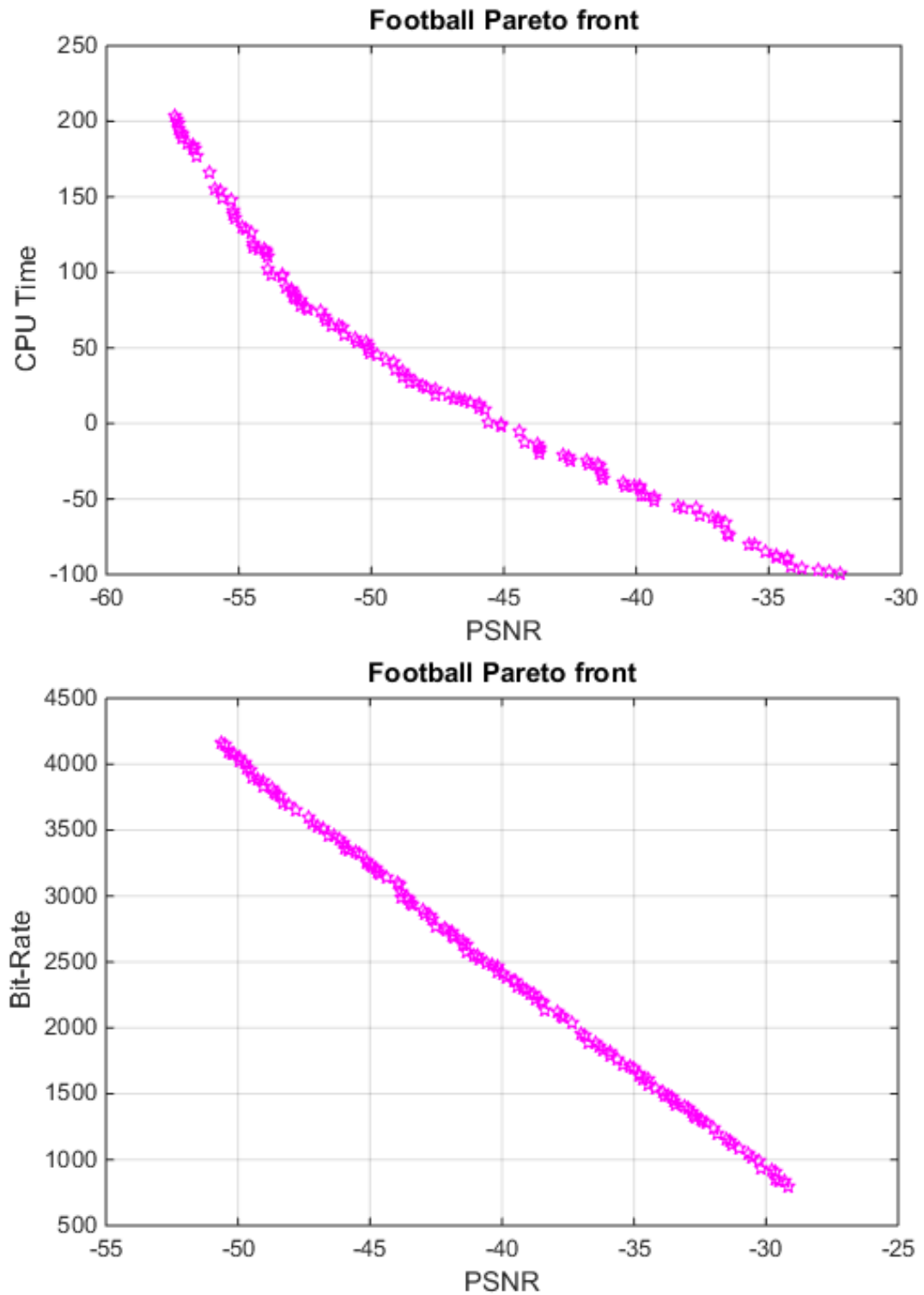
8	32	49	2	19.265	32.91	49.632	0.308
8	32	49	5	19.634	32.33	65.116	0.27
8	32	49	8	19.643	32.66	80.676	0.294

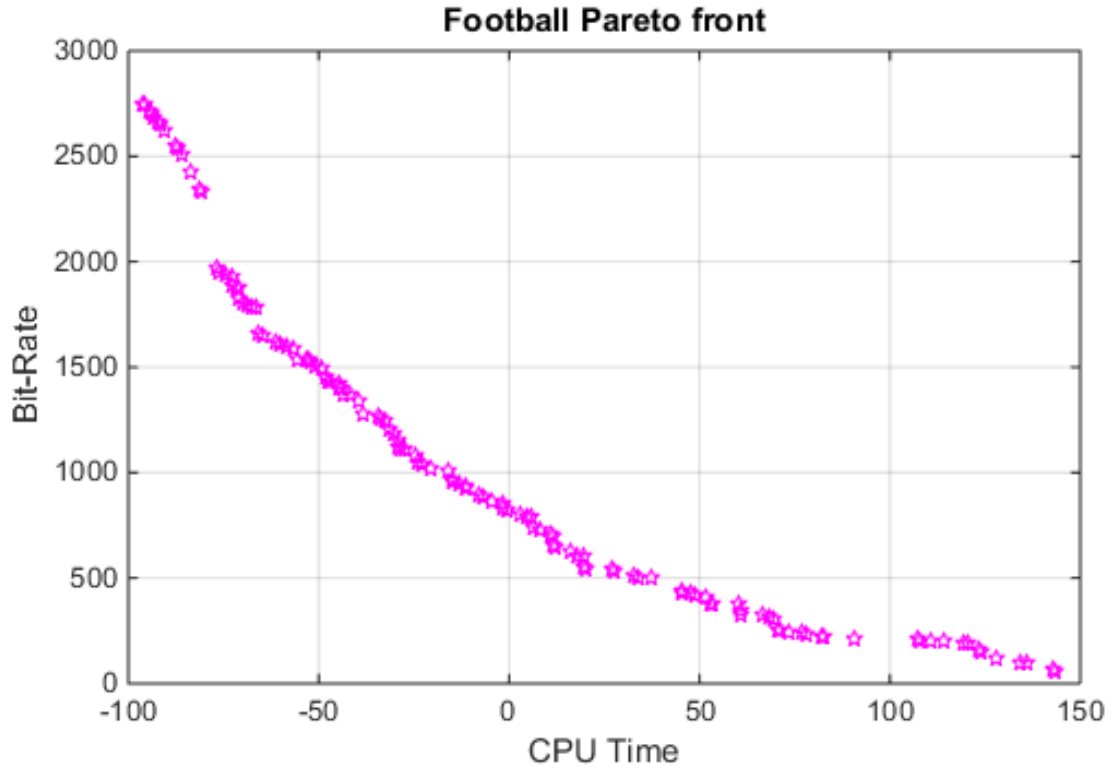
Table B. 6: Selected set of parameters for Tennis video sequence.

IntraPeriod	SearchRange	QP	NRFrames	PSNR	Bit-rate	CPU	CPU/Decoder
0	16	17	2	43.703	1258.3	73.525	0.801
0	16	17	5	43.669	1061.34	94.295	0.692
0	16	17	8	43.663	1045.54	97.927	0.662
0	16	49	2	24.284	8.56	37.576	0.211
0	16	49	5	24.462	7.96	51.811	0.202
0	16	49	8	24.475	8.04	62.768	0.2
0	32	17	2	43.7	1259.82	66.572	0.727
0	32	17	5	43.655	1062.45	83.713	0.681
0	32	17	8	43.643	1042.94	99.129	0.665
0	32	49	2	24.288	8.72	38.227	0.203
0	32	49	5	24.511	8.02	52.192	0.198
0	32	49	8	24.518	8.15	62.973	0.199
5	16	17	2	45.309	1735.94	61.859	0.853
5	16	17	5	45.066	1443.39	75.479	0.762
5	16	17	8	45.063	1432.58	89.736	0.763
5	16	49	2	24.509	15.84	34.586	0.24
5	16	49	5	24.788	15.43	47.114	0.231
5	16	49	8	24.791	15.5	57.398	0.233
5	32	17	2	45.316	1736.75	62.615	0.837
5	32	17	5	45.071	1443.81	76.39	0.762
5	32	17	8	45.079	1433.22	91.026	0.765
5	32	49	2	24.525	15.85	35.106	0.236
5	32	49	5	24.788	15.48	47.491	0.232
5	32	49	8	24.791	15.5	57.794	0.229
8	16	17	2	43.805	1330.59	64.657	0.722
8	16	17	5	43.798	1128.17	77.083	0.666
8	16	17	8	43.796	1113.46	91.711	0.659
8	16	49	2	24.337	12.65	36.02	0.222
8	16	49	5	24.674	12.23	48.242	0.215
8	16	49	8	24.682	12.25	58.291	0.217
8	32	17	2	43.807	1330.7	63.395	0.72
8	32	17	5	43.796	1129.92	78.35	0.66
8	32	17	8	43.793	1113.79	93.611	0.656
8	32	49	2	24.389	12.73	35.961	0.226
8	32	49	5	24.677	12.16	48.955	0.215
8	32	49	8	24.692	12.29	58.994	0.229

Appendix C: Pareto Plot

Plot C. 1: Pareto Plot for Football Video.





Appendix D: Optimal points and functional values.

Table D. 1: Foreman PSNR vs. Bit-rate

	$f(1)$	$f(2)$	X_1	X_2	X_3	X_4	X_5	X_6
1	-61.4345	1051.543	-12.3536	5.948443	-13.2784	18.68428	6.922016	11.36318
2	-56.2065	860.3077	-15.6062	6.330999	-4.14033	21.28292	8.077764	6.978926
3	-59.1944	959.3262	-14.4197	5.089091	-9.05509	22.12362	8.069374	8.776518
4	-62.8529	1102.272	-10.8373	5.983466	-15.4618	20.18949	7.297968	12.31851
5	-55.2319	827.3298	-15.3452	7.633843	-2.26184	23.04302	10.27598	6.575443
6	-62.6273	1085.166	-12.1723	6.900299	-15.0288	20.62279	7.015222	11.51751
7	-58.0206	919.1406	-14.9615	5.98795	-7.08268	22.08793	7.667866	8.319399
8	-40.8703	404.3069	-13.2207	14.54251	21.38374	18.91533	11.41972	2.244124
9	-57.8374	910.0747	-14.7867	6.106682	-6.52831	23.847	8.773665	9.403458
10	-52.579	747.9986	-15.5328	6.103938	1.956696	21.12791	7.869598	6.523647
11	-45.7358	536.7505	-16.1877	13.29562	13.13297	18.59521	9.332297	2.607052
12	-62.8153	1089.791	-12.1723	6.900299	-15.2788	21.12279	7.015222	11.51751
13	-56.6037	872.3414	-14.9189	6.229737	-4.53761	23.22131	8.72742	8.51145
14	-64.2498	1160.129	-8.63146	7.039144	-17.7765	20.64461	8.871526	9.890705

15	-64.5445	1179.086	- 6.97241	7.537578	-18.1901	21.32292	9.585319	8.898466
16	-66.0515	1235.996	- 5.41206	7.705062	-20.6914	21.56395	10.11495	10.11164
17	-52.0159	723.7014	- 15.9984	6.64863	3.098449	22.54893	6.529946	6.546711
18	-49.9581	664.811	- 16.6616	11.37207	6.037113	18.61657	10.58142	5.989593
19	-66.8771	1261.795	- 5.34666	8.862411	-22.0617	21.69734	12.40224	10.74369
20	-48.8086	629.961	- 16.6281	10.51147	7.932977	18.45426	8.760134	3.168724
21	-53.1061	760.4283	-16.219	6.067543	1.032235	20.84594	8.818547	6.91846
22	-47.1808	569.3852	- 17.9278	11.0445	10.72836	18.79524	9.431786	3.484754
23	-39.9426	379.9601	- 12.7207	15.54251	22.88374	18.41533	11.91972	2.494124
24	-37.0587	327.1868	- 7.61426	9.242817	27.6164	16.63976	4.897826	16.04133
25	-36.2372	305.7522	- 7.10057	8.441657	28.96404	16.3452	3.363182	16.17345
26	-60.8736	1022.776	-13.7537	4.488934	-12.2391	19.45727	8.899394	9.896185
27	-52.3088	743.1749	-15.1332	6.105105	2.364584	20.76614	7.856732	6.612254
28	-42.4729	450.183	- 14.2049	15.71193	18.54184	17.9197	12.23217	1.813011
29	-66.5667	1252.544	- 5.34666	9.112411	-21.5617	21.50984	11.65224	10.93119
30	-63.9595	1146.404	- 9.35309	6.526395	-17.3004	20.57108	8.819923	9.395061
31	-65.1111	1201.439	- 6.73728	8.221786	-19.3117	20.10359	10.15224	8.74369
32	-53.6093	774.305	-16.219	6.067543	0.282235	21.59594	8.818547	6.66846
33	-45.7263	522.9529	- 18.4339	13.22389	13.05152	17.92364	10.36744	1.702971
34	-62.9272	1104.152	-11.2212	5.89914	-15.7047	19.32724	7.166266	12.12836
35	-61.9056	1068.582	-11.9138	5.740745	-14.0373	18.95198	6.932407	11.34864
36	-59.9007	984.1579	-14.293	6.654113	-10.3505	21.20375	9.250422	11.4008
37	-54.9212	826.0167	- 14.4014	6.006544	-1.84334	22.30417	8.135473	5.752259
38	-63.7085	1137.851	- 9.51446	6.425226	-16.8976	20.37749	8.374987	10.0805
39	-60.1646	992.0274	- 14.2661	6.427959	-10.766	21.42701	9.083982	11.37609
40	-49.357	644.4899	-15.9591	8.778352	7.394093	21.23521	8.163604	4.717752
41	-60.2867	997.1175	-14.1183	6.483588	-10.9866	21.30807	9.221157	11.54389
42	-66.3597	1240.496	- 6.09666	9.487411	-21.1867	21.69734	12.58974	10.99369
43	-41.2395	417.5742	- 12.9707	14.54251	20.75874	18.91533	11.41972	1.994124
44	-39.2372	364.6659	-11.2387	8.539053	24.25607	19.16594	4.864439	12.90672
45	-44.9908	499.4415	- 18.6554	13.78114	14.24067	17.56389	10.46424	1.235292
46	-35.4404	283.8393	- 6.41307	8.707282	30.40154	17.0327	3.550682	16.22033
47	-41.4123	430.5598	- 12.5901	9.493237	20.19704	16.52902	4.184431	8.599632
48	-61.2323	1037.509	- 12.7009	6.766199	-12.6477	20.88092	7.283359	10.52272

49	-63.6132	1124.03	-10.7364	7.313068	-16.6005	21.31607	8.043304	11.35472
50	-52.579	747.9986	-15.5328	6.228938	1.956696	21.12791	7.978973	6.523647
51	-53.3652	767.6557	-16.219	6.067543	0.64161	21.22094	8.568547	6.41846
52	-45.7335	525.256	-18.2859	12.90914	12.98156	17.50248	10.37077	1.566837
53	-51.0542	698.1843	-16.5359	10.60957	4.279656	19.27702	10.2539	6.427399
54	-38.7187	350.7433	-11.1606	8.023428	25.03732	18.46282	5.786314	13.03172
55	-43.6302	467.1285	-16.9797	12.62711	16.60417	17.8702	9.678931	4.623302
56	-51.849	721.749	-15.8581	6.103254	3.25566	21.58818	7.558033	7.180959
57	-67.0511	1276.297	-3.86657	8.056877	-22.3004	22.16796	10.55502	9.648677
58	-36.7408	317.6449	-7.19242	7.91104	28.28745	17.64824	3.695282	15.88421
59	-63.8541	1145.035	-9.07156	6.619034	-17.1228	20.54222	8.585867	9.874549
60	-59.706	976.0243	-14.4896	6.441999	-9.98331	21.51348	8.924119	10.99764
61	-53.0171	757.1366	-15.6909	6.510954	1.404599	22.49862	6.738583	6.87452
62	-49.757	659.4709	-16.3022	10.86146	6.458069	19.33227	9.371443	4.095178
63	-54.8345	818.6735	-15.3491	6.338712	-1.79544	21.53799	8.446348	6.376447
64	-43.1229	460.5956	-15.676	11.86669	17.43735	17.66953	9.023205	5.162115
65	-40.1952	385.4652	-12.7623	14.08301	22.57088	19.16186	11.06051	4.157259
66	-55.2353	832.6873	-15.157	7.455334	-2.48209	21.41003	8.445503	7.263098
67	-41.3778	429.7228	-12.5433	9.524487	20.25954	16.56418	4.200056	8.537132
68	-65.006	1197.362	-6.53882	7.100078	-19.0182	20.92839	9.671257	9.398466
69	-35.7037	290.93	-6.47557	7.929938	29.99529	17.31786	3.425682	16.17345
70	-48.6914	625.9527	-16.4971	11.44837	8.197889	18.92573	9.561946	3.633893
71	-50.9512	691.0506	-16.1419	7.368343	4.810292	21.98412	7.272658	5.429335
72	-45.001	511.7187	-16.7857	13.12843	14.26564	17.77889	10.41123	2.953513
73	-35.3718	281.7851	-6.38573	8.504157	30.52264	17.06786	3.488182	16.31017
74	-60.4011	1002.974	-13.458	5.332504	-11.06	22.31739	8.082007	9.461304
75	-48.387	618.9329	-16.3319	10.49491	8.638115	18.38385	8.461968	3.566118
76	-44.6737	491.2763	-18.4292	13.46266	14.76531	17.45957	10.74347	1.715946
77	-63.089	1113.209	-10.3037	6.142884	-15.8564	20.26692	7.453958	11.45122
78	-60.7152	1017.983	-13.7146	4.363934	-11.9657	19.50415	8.899394	9.872747
79	-57.9944	918.2733	-14.9615	6.05045	-7.03581	22.10747	7.667866	8.331118
80	-42.6076	453.5311	-14.3924	15.64943	18.29184	17.73611	12.10717	2.070824
81	-53.8286	789.1943	-15.242	6.286032	-0.16271	21.06292	8.16361	6.272996
82	-49.5046	652.2555	-16.2193	9.541237	6.878586	19.30253	9.149291	3.913304

83	-59.357	967.6315	-14.1153	4.953125	-9.39187	21.66449	8.107256	8.700779
84	-37.5369	342.2199	-7.61426	9.367817	26.8039	16.63976	4.034545	15.3343
85	-51.3461	705.1991	-15.9984	6.14863	4.098449	21.54893	7.529946	7.109211
86	-59.8845	983.2261	-14.3555	6.654113	-10.3232	21.20375	9.250422	11.4008
87	-49.1219	638.2421	-16.6008	11.30834	7.495477	19.09879	9.510134	3.731224
88	-40.6422	398.9844	-13.1378	15.25484	21.70195	18.40704	11.61396	2.485312
89	-63.3479	1116.091	-10.8701	7.333067	-16.2207	20.81044	7.915725	11.37695
90	-58.5348	932.9666	-15.1442	6.175625	-7.89723	22.49162	7.831635	8.176308
91	-60.5158	1012.367	-12.682	5.58332	-11.2819	22.13764	8.487475	8.917931
92	-59.495	975.1789	-13.3965	6.56329	-9.53512	22.33441	8.596122	8.631024
93	-40.4136	393.7299	-12.8097	15.24747	22.10673	18.54036	11.79028	2.620046
94	-64.8572	1183.656	-7.53031	8.384754	-18.6425	21.79911	10.78283	10.34106
95	-65.5753	1213.006	-6.60513	8.770345	-19.8882	21.52129	11.00579	10.79263
96	-39.2749	365.6194	-11.7953	14.5144	23.99958	18.2185	10.6881	4.512272
97	-47.8296	590.6476	-17.8318	11.92816	9.614529	18.69078	9.948486	2.851183
98	-43.9398	476.6948	-16.9372	12.42536	16.10279	18.02288	9.58194	4.697561
99	-37.3786	333.0412	-8.18074	9.941671	27.09138	16.83673	5.676797	14.53848
100	-64.5171	1174.093	-7.47872	8.6379	-18.1067	21.51087	10.74605	10.4292
101	-65.9466	1227.164	-6.19306	8.949956	-20.5016	21.60943	11.1501	10.96584
102	-46.9021	560.7097	-18.1175	12.67267	11.1274	18.33012	10.19084	2.391024
103	-42.4088	443.0406	-14.9327	15.24097	18.65985	17.99078	11.99562	1.834861
104	-43.0446	455.1771	-16.1399	12.71003	17.55937	17.57732	10.01605	5.458591
105	-61.9838	1070.895	-11.9138	5.740745	-14.1623	18.99885	6.948032	11.44239
106	-57.5455	897.0552	-15.8703	6.191545	-6.22367	22.47326	7.958208	9.018813
107	-51.4278	708.3635	-15.8734	6.27363	3.973449	21.64268	7.561196	7.140461
108	-45.4516	515.3661	-18.3796	13.00289	13.48156	17.68998	10.9489	1.441837
109	-41.9432	431.7235	-14.4117	15.57847	19.46339	18.11843	12.29288	1.918175
110	-64.4714	1167.341	-8.50646	7.086019	-18.1202	20.86336	8.871526	9.984455
111	-56.9005	877.8957	-15.8597	6.574965	-5.18425	22.12186	8.006364	8.802045
112	-48.0335	605.96	-16.8028	10.0445	9.165856	17.90462	10.30679	3.031629
113	-63.5433	1121.717	-10.7364	7.328693	-16.4755	21.36294	7.918304	11.44847
114	-47.4635	582.2955	-17.6271	11.80343	10.14131	18.00028	9.523217	3.107143
115	-50.6987	682.8548	-16.1504	7.600527	5.250128	22.09239	7.584892	5.314675

116	-42.2605	438.415	- 14.9327	15.36597	18.90985	17.99078	11.96437	1.803611
117	-39.9057	378.5147	- 12.7207	15.55814	22.96187	18.54033	12.04472	2.369124
118	-58.241	926.5836	- 14.7398	6.137932	-7.40331	22.472	8.086165	8.262833
119	-66.4926	1250.232	- 5.34666	9.128036	-21.4367	21.50984	11.77724	10.93119
120	-36.9105	323.4129	- 7.48926	9.242817	27.8664	16.63976	4.897826	16.04133
121	-57.4184	893.2207	-15.839	6.254045	-6.00492	22.50451	7.833208	9.018813
122	-55.5572	843.3902	- 15.0914	7.032893	-3.03639	21.37213	8.580284	6.494291
123	-50.4038	672.34	- 17.4576	10.34873	5.337049	19.08276	11.19782	4.615656
124	-50.7767	685.9128	-16.041	7.678652	5.125128	22.13926	7.506767	5.345925
125	-44.4458	485.8949	- 18.1246	13.25809	15.1683	17.56818	10.70263	2.203556
126	-65.6614	1215.756	- 6.55404	8.882469	-20.018	21.63177	11.00906	10.86928
127	-46.754	555.6583	-18.18	12.56329	11.3774	18.33012	10.25334	2.391024
128	-55.9329	853.7226	-15.357	6.611018	-3.69264	21.18463	8.252159	6.869291
129	-67.2567	1282.529	- 3.83103	8.100692	-22.6242	22.33406	10.61708	9.747597
130	-65.8622	1224.851	- 6.19306	8.887456	-20.3766	21.48443	11.1501	10.87209
131	-44.0267	482.2186	- 16.6766	12.25434	15.90021	17.66651	9.879495	3.6782
132	-54.4695	805.1416	-15.617	6.37197	-1.16271	21.66839	8.429235	6.272996
133	-58.8128	939.4879	- 15.2947	5.948466	-8.30509	22.93612	7.834999	8.339018
134	-52.2783	742.0451	-15.1293	6.105105	2.427084	20.84035	7.927044	6.737254
135	-38.5502	343.4318	- 11.0669	9.886709	25.46701	19.49407	4.895689	13.87157
136	-52.1571	730.9261	- 15.5909	6.39023	2.858048	22.5433	6.871797	6.379436
137	-54.0592	799.9485	- 14.2576	5.981345	-0.38146	22.38714	8.03861	5.257371
138	-56.5326	869.1769	- 15.0439	6.245362	-4.41261	23.25256	8.72742	8.620825
139	-46.4227	548.0609	-17.9121	12.55198	11.88666	17.95687	10.49096	2.461255
140	-38.9992	355.1166	-11.5142	10.54041	24.67075	19.31855	5.690005	11.9468

Table D. 2: Football PSNR vs. Bit-rate

	$f(1)$	$f(2)$	X_1	X_2	X_3	X_4	X_5	X_6
1	-36.4775	1885.33	7.472893	3.204588	16.11738	30.47397	-5.2426	1.267363
2	-32.2449	1280.773	8.49622	0.801249	25.40222	28.46694	2.252588	-2.33526
3	-32.4568	1312.466	8.66175	1.22519	24.9281	28.51551	2.198221	-2.14312
4	-45.9977	3401.15	13.69205	11.31332	-5.8101	29.43559	8.666342	7.178861
5	-38.4406	2195.334	8.613387	4.696102	11.61588	30.33814	-2.71451	3.002132
6	-44.949	3224.616	11.21447	9.111992	-3.32792	29.87751	5.269959	7.132587
7	-46.153	3425.805	13.64624	11.44334	-6.16621	29.41626	8.722025	7.444209
8	-44.3379	3136.321	12.48134	9.658321	-1.98379	29.63779	6.054117	6.099451
9	-44.7087	3186.178	11.5581	8.944934	-2.77137	29.9015	5.153976	7.315548
10	-39.0368	2287.446	9.777756	5.256761	10.26269	30.36808	-1.59636	3.614672
11	-31.9953	1243.542	8.41577	1.078318	25.96185	28.39798	2.302892	-2.26828
12	-50.6029	4162.976	14.83888	17.27334	-16.5953	27.85814	11.77781	11.81266
13	-36.859	1943.26	7.889711	3.636809	15.25939	30.52399	-4.50985	1.865606
14	-36.3511	1862.047	7.472893	3.048338	16.42988	30.59897	-5.3676	1.267363
15	-43.9579	3101.694	11.81599	8.273249	-1.27723	28.73857	6.778175	6.224843
16	-43.5962	2988.416	12.19382	9.243413	-0.07866	30.79123	5.345989	5.607564
17	-37.6786	2079.815	8.849303	4.031209	13.33136	30.21795	-3.48742	2.327964
18	-37.3771	2035.324	7.243515	3.641932	14.00105	30.13132	-4.57905	1.883168
19	-49.9333	4057.62	14.92436	15.10836	-15.0547	27.85595	10.85719	12.53578
20	-44.7755	3204.31	13.96271	11.42298	-2.97145	29.60989	7.050213	7.901733
21	-32.3899	1296.782	8.66175	1.22519	25.1156	28.70301	2.448221	-2.14312
22	-44.9682	3237.038	13.06579	10.85335	-3.43296	29.53396	9.403721	7.312109
23	-49.7401	4004.632	15.50685	16.46084	-14.4703	28.7097	11.68406	11.14078
24	-37.3771	2035.324	6.993515	3.641932	14.00105	30.13132	-4.32905	1.883168
25	-43.8751	3079.829	12.19099	8.585749	-1.02723	29.05107	6.778175	6.521718
26	-45.0921	3251.281	13.31579	10.79085	-3.68296	29.72146	9.403721	7.249609
27	-29.325	834.0135	7.036586	0.778055	32.01901	28.08148	0.827641	-2.21941
28	-48.6552	3796.485	13.04058	12.33094	-11.7561	30.1844	10.06162	7.343364
29	-45.5101	3335.033	12.96494	10.42979	-4.75736	29.04863	7.094452	7.835925
30	-43.8512	3032.184	12.32519	8.640334	-0.68755	30.65241	5.875436	5.674094
31	-30.7033	1049.282	7.147182	2.194388	28.87483	28.06939	1.240755	-0.90525
32	-44.6105	3167.171	11.7456	9.194934	-2.52137	30.0265	5.153976	7.503048
33	-46.396	3454.317	12.61741	10.15662	-6.6521	29.72634	7.419611	8.710368
34	-50.5212	4148.907	14.8549	17.19593	-16.4001	27.90538	11.76017	11.74601
35	-38.8174	2252.786	9.176463	4.814283	10.76374	30.39327	-1.97474	3.06312
36	-32.9816	1386.757	8.172756	3.145232	23.79022	28.7483	1.974047	-0.17016
37	-48.0992	3694.802	12.64145	11.98521	-10.3863	30.70773	9.925277	7.39236
38	-38.7014	2221.917	8.863387	4.696102	11.11588	30.83814	-2.21451	3.502132
39	-42.0986	2745.807	15.3284	10.19653	3.429355	30.96421	3.743763	9.96376
40	-30.5713	1016.449	7.458158	1.245734	29.25545	28.51521	1.077295	-1.22203
41	-37.914	2120.114	7.416195	3.677159	12.7694	30.09611	-2.56258	2.281574
42	-40.347	2482.565	14.96893	10.0964	7.361436	30.58766	3.391094	9.059862
43	-39.5634	2361.579	14.64233	9.982671	9.145299	30.51922	2.680758	9.254929
44	-31.8404	1198.989	8.547355	1.364297	26.45106	29.12539	2.501441	-2.18588
45	-43.3808	2934.945	12.53757	9.743413	0.546341	31.49435	5.439739	5.857564
46	-48.5484	3779.383	13.54058	12.58094	-11.5061	30.1844	10.06162	7.843364
47	-29.4755	843.4941	7.161586	0.778055	31.76901	28.58148	0.827641	-1.96941
48	-39.1943	2295.974	9.777756	5.881761	10.01269	30.93058	-1.40886	4.302172
49	-33.0908	1403.858	8.172756	2.895232	23.54022	28.7483	2.974047	-0.17016
50	-43.5245	2975.607	12.34927	8.766206	0.0991	30.83385	5.854595	6.150291

51	-45.9696	3361.778	13.7656	11.18984	-5.51388	30.68922	8.574731	7.326088
52	-48.73	3820.37	15.28693	14.01487	-11.9946	29.68795	8.912524	9.576443
53	-45.1586	3269.67	13.2656	10.50234	-3.88888	29.43922	10.44973	7.076088
54	-33.6989	1479.399	8.463538	2.136994	22.26739	29.50476	1.622442	-1.95022
55	-33.5721	1466.107	8.713538	1.886994	22.51739	29.25476	0.872442	-1.45022
56	-29.6697	906.0785	6.97849	1.362589	31.10999	27.43319	0.540795	-2.27302
57	-37.7427	2086.461	9.302428	3.937459	13.20636	30.34295	-3.23742	2.218589
58	-39.468	2360.649	14.14233	10.02955	9.270299	30.01922	2.680758	9.098679
59	-42.7302	2850.179	15.54297	9.214795	1.928137	30.85408	4.299619	6.74947
60	-32.6484	1323.364	8.66175	1.72519	24.6156	29.20301	2.448221	-2.14312
61	-40.9222	2545.081	16.19168	10.90483	6.231085	31.55915	2.515952	9.852289
62	-41.7911	2690.691	15.3284	10.19653	4.179355	31.21421	3.993763	9.96376
63	-31.0167	1086.542	7.927211	0.586649	28.22214	28.55414	2.538579	-3.13348
64	-37.0027	1959.784	8.279241	3.413084	14.97098	30.73423	-4.11581	2.062218
65	-46.8006	3513.287	13.12145	10.70273	-7.54667	29.87231	7.4741	9.200665
66	-35.8944	1822.187	9.504408	3.346	17.26152	29.48169	2.627699	0.434286
67	-38.9598	2263.79	9.017728	4.968743	10.51598	30.78331	-1.88838	3.682996
68	-36.1528	1830.373	7.727838	3.207878	16.88863	30.61813	-4.6503	1.389986
69	-42.9549	2868.208	13.42035	10.09287	1.527483	31.46943	4.868724	6.82764
70	-41.462	2634.286	16.35077	11.17283	4.964207	31.3924	2.901758	9.825541
71	-31.534	1150.983	9.029628	3.08456	27.17183	29.0401	1.835831	1.057088
72	-40.1885	2459.696	15.13868	11.49852	7.713934	30.50601	1.442659	9.493625
73	-38.504	2201.146	8.630251	4.86196	11.4963	30.49346	-2.52863	2.728588
74	-45.4341	3321.163	12.93564	10.34783	-4.57096	29.12209	7.038619	7.73691
75	-42.6212	2822.825	15.10176	10.51848	2.260147	31.15867	3.995779	9.21031
76	-31.3811	1147.088	7.891467	0.614726	27.36757	28.41717	1.50664	-2.6622
77	-36.7409	1890.004	8.139711	4.136809	15.75939	31.77399	-3.50985	1.865606
78	-49.2852	3889.609	15.16146	14.30183	-13.1355	30.26558	12.24716	11.83458
79	-43.0257	2890.152	15.94925	8.740139	1.294125	31.07701	4.383636	6.906738
80	-39.8524	2378.873	15.69479	10.08547	8.660807	31.55903	2.267746	7.986729
81	-50.358	4096.634	15.73775	17.03272	-15.8508	28.86953	11.59578	11.62122
82	-43.3808	2934.945	12.53757	10.74341	0.546341	31.49435	5.439739	5.857564
83	-29.7943	921.147	8.552211	0.508524	30.86276	27.55414	1.163579	-0.80535
84	-39.9364	2407.928	13.65972	10.26825	8.368972	30.96255	2.788331	8.925284
85	-30.2148	932.8002	7.458158	1.245734	30.25545	29.51521	1.077295	-1.22203
86	-46.5609	3465.273	12.34439	11.50624	-6.94409	30.31798	8.259984	6.576571
87	-34.782	1634.979	10.67865	2.64746	19.894	29.94928	2.029911	0.389116
88	-32.8266	1365.189	8.66175	1.22519	24.1156	28.70301	3.448221	-2.14312
89	-35.1468	1703.793	10.08189	4.845273	18.99149	29.48511	1.86939	2.064646
90	-34.2058	1544.355	9.297726	2.413966	21.22988	29.89946	2.786377	2.752421
91	-50.1768	4073.039	15.50685	16.46084	-15.4703	28.7097	11.68406	11.14078
92	-33.4663	1445.755	8.312296	2.454434	22.7932	29.35218	3.07614	0.012247
93	-47.3186	3593.443	12.89397	10.71626	-8.72077	29.88281	7.924028	10.1401
94	-34.9848	1692.023	9.178645	3.77246	19.269	29.01178	1.904911	1.451616
95	-40.164	2427.515	15.10788	12.60335	7.957913	31.52231	1.33089	9.591937
96	-33.903	1514.162	8.312296	3.454434	21.7932	29.35218	3.07614	0.012247
97	-48.3032	3706.759	14.89758	12.50785	-10.7099	31.37591	7.519003	9.327469
98	-42.545	2766.723	15.85541	10.83148	2.737111	32.69876	2.142264	11.03649
99	-49.9213	4028.227	15.73775	18.03272	-14.8508	28.86953	12.59578	11.62122
100	-49.0522	3827.828	13.54159	13.98729	-12.4609	31.29093	10.61108	7.451975
101	-30.5713	1016.449	7.458158	0.245734	29.25545	28.51521	1.077295	-1.22203
102	-30.231	989.554	8.552211	0.508524	29.86276	27.55414	1.163579	-0.80535
103	-32.7494	1340.037	9.704497	3.244689	24.39564	29.09634	2.298761	0.737607
104	-34.4555	1610.903	9.797726	2.413966	20.47988	28.89946	2.786377	3.252421

105	-41.0899	2548.175	17.24958	11.50976	6.00554	32.36837	2.303531	10.76083
106	-49.4936	3896.235	14.54159	13.98729	-13.4609	31.29093	10.61108	8.451975
107	-33.4019	1420.041	8.36287	2.703519	23.03573	29.95073	0.655456	-1.37027
108	-41.3129	2582.379	16.99958	11.50976	5.50554	32.36837	2.553531	11.76083
109	-33.8433	1488.448	7.36287	1.703519	22.03573	29.95073	2.655456	-0.37027
110	-46.9976	3533.68	12.34439	11.50624	-7.94409	30.31798	9.259984	6.576571
111	-39.4204	2310.466	14.69479	10.08547	9.660807	31.55903	2.267746	8.986729
112	-47.8355	3646.608	12.89397	11.71626	-9.72077	30.88281	7.924028	10.1401
113	-49.5078	3949.248	14.42436	16.10836	-13.8047	29.35595	10.85719	12.53578
114	-29.155	792.1889	8.036586	0.340555	32.51901	28.58148	1.577641	-0.46941
115	-41.9114	2733.132	14.55084	9.845033	3.746521	30.37236	3.339621	9.717591
116	-45.7965	3342.156	14.64624	11.44334	-5.16621	30.41626	9.722025	7.444209
117	-31.2618	1111.646	7.605536	2.317119	27.76567	28.95574	-0.0873	-0.25798
118	-29.155	792.1889	7.036586	0.340555	32.51901	28.58148	2.577641	-0.46941
119	-47.191	3560.837	14.16223	10.14255	-8.35029	30.3593	7.779259	9.284617
120	-41.8753	2704.077	16.10541	10.33148	3.987111	31.19876	4.142264	10.28649
121	-49.0665	3866.957	13.83846	13.78416	-12.7265	29.91593	10.6267	9.2801
122	-35.4438	1724.445	7.421234	4.270633	18.48388	30.40831	0.142011	2.340608
123	-29.6672	845.3537	7.036586	1.340555	31.51901	29.58148	1.577641	-1.46941
124	-42.1831	2754.955	15.09724	11.51611	3.252407	31.15817	4.999219	8.206612
125	-49.696	3975.145	15.38185	15.21084	-14.2203	29.5222	12.55906	11.14078
126	-50.2511	4079.327	15.83888	17.27334	-15.5953	28.85814	11.77781	12.81266
127	-40.8715	2513.972	15.99958	11.50976	6.50554	32.36837	2.553531	10.76083
128	-41.47	2664.725	15.55084	9.845033	4.746521	30.37236	5.339621	8.717591
129	-41.3387	2632.562	14.5777	10.55663	5.13054	30.75899	2.616031	9.854581
130	-40.573	2495.011	18.24958	11.50976	7.00554	31.36837	2.303531	10.76083
131	-43.8254	2994.568	12.34927	8.766206	-0.4009	31.83385	5.854595	6.650291
132	-48.4368	3762.281	13.04058	13.33094	-11.2561	30.1844	11.06162	7.343364
133	-34.796	1658.397	9.797726	3.413966	19.72988	29.14946	3.286377	1.752421
134	-34.6331	1612.762	9.297726	2.413966	20.22988	29.89946	2.786377	0.752421
135	-43.4994	2948.535	12.32519	8.640334	0.312453	31.65241	6.875436	6.674094
136	-35.8852	1792.852	7.421234	4.270633	17.48388	30.40831	0.142011	3.340608
137	-50.6029	4162.976	13.83888	17.27334	-16.5953	27.85814	11.77781	11.81266
138	-35.6637	1756.957	11.08189	4.845273	17.99149	30.48511	1.86939	2.064646
139	-34.4395	1574.748	8.797726	2.413966	20.72988	30.14946	2.286377	1.752421
140	-38.3496	2138.267	8.863387	5.696102	12.11588	31.83814	-1.21451	4.502132

Appendix E: HEVC HM encoder / decoder configuration file.

E.1. Encoder configuration file.

```
#===== File I/O =====
BitstreamFile      : rand.bin
ReconFile          : rand.yuv
FrameRate          : 120    # Frame Rate per second
FrameSkip          : 0      # Number of frames to be skipped in input
SourceWidth        : 1920  # Input frame width
SourceHeight       : 1080  # Input frame height
FramesToBeEncoded  : 50    # Number of frames to be coded
#===== Profile =====
Profile            : main

#===== Unit definition =====
MaxCUWidth         : 64    # Maximum coding unit width in pixel
MaxCUHeight        : 64    # Maximum coding unit height in pixel
MaxPartitionDepth  : 4     # Maximum coding unit depth
QuadtreeTULog2MaxSize : 5   # Log2 of maximum transform size for
                          # quadtree-based TU coding (2...6)
QuadtreeTULog2MinSize : 2   # Log2 of minimum transform size for
                          # quadtree-based TU coding (2...6)
QuadtreeTUMaxDepthInter : 3
QuadtreeTUMaxDepthIntra : 3

#===== Coding Structure =====
IntraPeriod        : 32    # Period of I-Frame ( -1 = only first)
DecodingRefreshType : 1    # Random Access 0:none, 1:CRA, 2:IDR, 3:Recovery
Point SEI
GOPSize           : 8     # GOP Size (number of B slice = GOPSize-1)
#   Type POC QPoffset QPfactor tcOffsetDiv2 betaOffsetDiv2 temporal_id
#ref_pics_active #ref_pics reference pictures   predict deltaRPS #ref_idcs reference idcs
Frame1: B 8 1 0.442 0 0 0 2 3 -8 -12 -16 0
Frame2: B 4 2 0.3536 0 0 1 2 3 -4 -8 4 1
4 4 1101
Frame3: B 2 3 0.3536 0 0 2 2 4 -2 -6 2 6 1
2 4 1111
Frame4: B 1 4 0.68 0 0 3 2 4 -1 1 3 7 1 1
5 10111
Frame5: B 3 4 0.68 0 0 3 2 4 -1 -3 1 5 1 -2
5 11110
Frame6: B 6 3 0.3536 0 0 2 2 3 -2 -6 2 1 -
3 5 01110
Frame7: B 5 4 0.68 0 0 3 2 4 -1 -5 1 3 1 1
4 1111
Frame8: B 7 4 0.68 0 0 3 2 4 -1 -3 -7 1 1 -2
5 11110

#===== Motion Search =====
FastSearch         : 1    # 0:Full search 1:TZ search
SearchRange        : 64   # (0: Search range is a Full frame)
BipredSearchRange  : 4    # Search range for bi-prediction refinement
HadamardME         : 1    # Use of hadamard measure for fractional ME
FEN                : 1    # Fast encoder decision
```

```

FDM                : 1          # Fast Decision for Merge RD cost

#===== Quantization =====
QP                 : 37          # Quantization parameter(0-51)
MaxDeltaQP         : 0          # CU-based multi-QP optimization
MaxCuDQPDepth      : 0          # Max depth of a minimum CuDQP for sub-LCU-level
delta QP
DeltaQpRD          : 0          # Slice-based multi-QP optimization
RDOQ               : 1          # RDOQ
RDOQTS            : 1          # RDOQ for transform skip

#===== Deblock Filter =====
LoopFilterOffsetInPPS : 1          # Dbl params: 0=varying params in SliceHeader,
param = base_param + GOP_offset_param; 1 (default) =constant params in PPS, param =
base_param)
LoopFilterDisable   : 0          # Disable deblocking filter (0=Filter, 1=No Filter)
LoopFilterBetaOffset_div2 : 0          # base_param: -6 ~ 6
LoopFilterTcOffset_div2 : 0          # base_param: -6 ~ 6
DeblockingFilterMetric : 0          # blockiness metric (automatically configures
deblocking parameters in bitstream). Applies slice-level loop filter offsets
(LoopFilterOffsetInPPS and LoopFilterDisable must be 0)

#===== Misc. =====
InternalBitDepth    : 8          # codec operating bit-depth

#===== Coding Tools =====
SAO                 : 1          # Sample adaptive offset (0: OFF, 1: ON)
AMP                 : 1          # Asymmetric motion partitions (0: OFF, 1: ON)
TransformSkip       : 1          # Transform skipping (0: OFF, 1: ON)
TransformSkipFast   : 1          # Fast Transform skipping (0: OFF, 1: ON)
SAOLcuBoundary      : 0          # SAOLcuBoundary using non-deblocked pixels (0:
OFF, 1: ON)

#===== Slices =====
SliceMode           : 0          # 0: Disable all slice options.
# 1: Enforce maximum number of LCU in an slice,
# 2: Enforce maximum number of bytes in an 'slice'
# 3: Enforce maximum number of tiles in a slice
SliceArgument       : 1500       # Argument for 'SliceMode'.
# If SliceMode==1 it represents max. SliceGranularity-sized blocks
per slice.
# If SliceMode==2 it represents max. bytes per slice.
# If SliceMode==3 it represents max. tiles per slice.

LFCrossSliceBoundaryFlag : 1          # In-loop filtering, including ALF and DB, is across
or not across slice boundary.
# 0:not across, 1: across

#===== PCM =====
PCMEnabledFlag      : 0          # 0: No PCM mode
PCMLog2MaxSize      : 5          # Log2 of maximum PCM block size.
PCMLog2MinSize      : 3          # Log2 of minimum PCM block size.
PCMInputBitDepthFlag : 1          # 0: PCM bit-depth is internal bit-depth. 1:
PCM bit-depth is input bit-depth.
PCMFilterDisableFlag : 0          # 0: Enable loop filtering on I_PCM samples. 1:
Disable loop filtering on I_PCM samples.

```

```

#===== Tiles =====
TileUniformSpacing      : 0      # 0: the column boundaries are indicated by
TileColumnWidth array, the row boundaries are indicated by TileRowHeight array
                        # 1: the column and row boundaries are distributed
uniformly
NumTileColumnsMinus1    : 0      # Number of tile columns in a picture minus 1
TileColumnWidthArray    : 2 3    # Array containing tile column width values in
units of CTU (from left to right in picture)
NumTileRowsMinus1       : 0      # Number of tile rows in a picture minus 1
TileRowHeightArray      : 2      # Array containing tile row height values in units
of CTU (from top to bottom in picture)

LFCrossTileBoundaryFlag : 1      # In-loop filtering is across or not across tile
boundary.
                        # 0: not across, 1: across

#===== WaveFront =====
WaveFrontSynchro        : 0      # 0: No WaveFront synchronisation
(WaveFrontSubstreams must be 1 in this case).
                        # >0: WaveFront synchronises with the LCU above and to the
right by this many LCUs.

#===== Quantization Matrix =====
ScalingList              : 0      # ScalingList 0 : off, 1 : default, 2 : file read
ScalingListFile          : scaling_list.txt # Scaling List file name. If file is not exist, use
Default Matrix.

#===== Lossless =====
TransquantBypassEnableFlag : 0      # Value of PPS flag.
CUTransquantBypassFlagForce: 0      # Force transquant bypass mode, when
transquant_bypass_enable_flag is enabled

#===== Rate Control =====
RateControl              : 0      # Rate control: enable rate control
TargetBitrate            : 1000000 # Rate control: target bitrate, in bps
KeepHierarchicalBit      : 2      # Rate control: 0: equal bit allocation; 1: fixed
ratio bit allocation; 2: adaptive ratio bit allocation
LCULevelRateControl      : 1      # Rate control: 1: LCU level RC; 0: picture level
RC
RCLCUSEparateModel       : 1      # Rate control: use LCU level separate R-
lambda model
InitialQP                 : 0      # Rate control: initial QP
RCForceIntraQP           : 0      # Rate control: force intra QP to be equal to initial
QP

### DO NOT ADD ANYTHING BELOW THIS LINE ###
### DO NOT DELETE THE EMPTY LINE BELOW ###

```

Appendix F: Selected parameters for encoder / decoder.

Table F. 7: Selected set of parameters for YachtRide video sequence.

x_1	x_2	x_3	x_4	Bitrate	PSNR	Encoding Total Time	Decoder Total Time
16	64	27	1	18405.06	39.6021	2124.89	7.29
16	64	37	1	4018.464	33.9397	1650.27	5.94
16	64	45	1	1086.758	30.2936	1493.54	5.37
16	128	27	1	18390.74	39.591	2263.67	7.31
16	128	37	1	4017.139	33.9394	1753.74	5.33
16	128	45	1	1087.45	30.296	1549.16	5.44
16	64	27	0	18352.42	39.6041	2763.36	6.9
16	64	37	0	4011.59	33.9482	2171.72	5.42
16	64	45	0	1088.794	30.3024	1935.59	5.14
16	128	27	0	18363.71	39.6095	2993.42	6.71
16	128	37	0	4000.243	33.9522	2322.22	5.38
16	128	45	0	1087.718	30.3146	2036.14	5.2
32	64	27	1	17405.78	39.5309	2249.58	6.67
32	64	37	1	3622.118	33.8531	1746.47	5.23
32	64	45	1	921.0048	30.1981	1554.49	5.01
32	128	27	1	17429.05	39.5282	2434.96	6.85
32	128	37	1	3621.715	33.857	1864.24	5.7
32	128	45	1	924.7296	30.2019	1626.98	5.21
32	64	27	0	17384.99	39.5382	2893.21	7.07
32	64	37	0	3616.147	33.8585	2277.76	5.61
32	64	45	0	921.5616	30.2083	2026.42	5.31
32	128	27	0	17390.11	39.5397	3168.09	7.73
32	128	37	0	3597.85	33.8582	2465.63	5.53
32	128	45	0	921.888	30.2196	2131.71	5.05
48	64	27	1	17374.54	39.5327	2271.23	6.95
48	64	37	1	3615.149	33.8601	1763.31	5.46
48	64	45	1	911.0208	30.1996	1573.14	4.8
48	128	27	1	17379.99	39.5343	2472.95	6.89
48	128	37	1	3608.371	33.8575	1886.85	5.26
48	128	45	1	911.8656	30.2041	1635.8	4.93
48	64	27	0	17345.11	39.5417	2930.57	6.74
48	64	37	0	3605.664	33.8625	2321.1	5.42
48	64	45	0	908.9856	30.215	2031.22	5.08
48	128	27	0	17351.25	39.5417	3217.08	6.82
48	128	37	0	3594.547	33.8557	2494.55	5.84
48	128	45	0	912.4224	30.2111	2155.85	5.01

Table F. 8: Selected set of parameters for Basketball video sequence.

x_1	x_2	x_3	x_4	Bitrate	PSNR	Encoding Total Time	Decoder Total Time
16	64	27	1	5550.168	38.2859	2280.43	6.8
16	64	37	1	1481.584	34.7351	1826.42	5.69
16	64	45	1	548.64	31.2064	1643.7	5.28
16	128	27	1	5549.152	38.2857	2614.58	6.66
16	128	37	1	1478.816	34.7349	2068.29	5.7
16	128	45	1	543.608	31.2079	1806.35	5.15
16	64	27	0	5534.288	38.3075	3001.27	6.6
16	64	37	0	7973.1	36.3127	2419.69	5.7
16	64	45	0	547.176	31.2211	2157.9	5.24
16	128	27	0	5527.608	38.3064	3527.54	6.63
16	128	37	0	1479.464	34.7655	2839.84	5.57
16	128	45	0	3142.3	32.6593	2467.7	5.16
32	64	27	1	25051.2	39.6721	2437.06	6.64
32	64	37	1	1360.616	34.6565	1951.98	5.5
32	64	45	1	498	31.1013	1742.78	5.21
32	128	27	1	5130.408	38.2381	2806.42	6.8
32	128	37	1	7621.2	36.455	2223.24	5.5
32	128	45	1	494.864	31.0901	1927.68	5.05
32	64	27	0	25051.2	39.6721	3145.93	6.78
32	64	37	0	1355.232	34.6668	2551.2	6.33
32	64	45	0	496.56	31.1192	2267.28	5.99
32	128	27	0	5109.248	38.2581	3747.96	7.01
32	128	37	0	1355.432	34.6826	3033.73	5.72
32	128	45	0	495.92	31.1208	2620.51	5.5
48	64	27	1	28912	39.4279	2464.21	7.66
48	64	37	1	1362.488	34.6352	1968.15	5.55
48	64	45	1	499.056	31.0794	1751.28	4.97
48	128	27	1	5174.104	38.2385	2848.61	6.52
48	128	37	1	1357.912	34.6272	2261.67	5.63
48	128	45	1	495.6	31.0706	1953.02	4.97
48	64	27	0	5159.32	38.2557	3181.37	6.77
48	64	37	0	1360.032	34.6548	2572.77	5.42
48	64	45	0	499.256	31.094	2275.11	5.38
48	128	27	0	5151.264	38.256	3805.04	6.6
48	128	37	0	1358.008	34.6602	3071.05	5.45
48	128	45	0	496.528	31.0773	2660.61	5.69

Appendix G: Optimal points and functional values.

Table G. 1 Cactus CPU vs. Bit-rate

	$f(1)$	$f(2)$	$x1$	$x2$	$x3$	$x4$
1	1646.09	-66.3645	33.58072	119.8237	44.9932	0.127074
2	1509.326	284.8627	17.83136	117.8421	44.99995	0.000133
3	1932.2	-242.308	41.29528	125.2825	45	0.570891
4	1720.201	-123.024	36.07405	121.3606	44.99487	0.237864
5	1541.762	176.3311	22.71379	117.8431	44.99694	0.02402
6	1813.105	-174.881	38.33283	123.2134	44.99774	0.383433
7	1631.618	-37.0107	32.28606	119.4465	44.99251	0.111635
8	1687.581	-98.0855	34.97661	120.6841	44.99414	0.1891
9	1604.952	12.60939	30.04791	118.7651	44.99423	0.082253
10	1561.601	116.4379	25.39167	118.0715	44.99625	0.039877
11	1525.612	259.4696	18.96479	118.1118	44.99976	0.020516
12	1767.712	-153.442	37.40122	122.3822	44.99642	0.310719
13	1834.722	-198.094	39.3427	123.8576	44.9991	0.412839
14	1621.06	-10.2551	31.08336	119.2486	44.9932	0.101961
15	1509.326	284.8627	17.83136	117.8421	44.99995	0.000133
16	1583.464	67.7079	27.58712	118.2408	44.99471	0.063166
17	1538.525	207.6281	21.28295	118.0657	44.99914	0.027565
18	1881.706	-209.279	39.83987	124.3604	44.99914	0.493074
19	1670.614	-82.2557	34.27538	120.3385	44.99394	0.164665
20	1552.845	147.935	24.02689	117.9865	44.99408	0.033968
21	1727.486	-127.922	36.26434	121.503	44.99648	0.249289

Table G. 2: YachtRide PSNR vs. Bit-rate

	$f(1)$	$f(2)$	$x1$	$x2$	$x3$	$x4$
1	-39.4109	16336.39	16	64	27	1
2	-30.0599	-739.003	24.07144	85.33057	44.98965	0.692503
3	-31.6552	2174.09	23.45643	81.80078	41.92058	0.695639
4	-39.1046	15777.1	16.48506	65.03705	27.58924	0.852184
5	-34.0909	6621.855	20.86016	76.51778	37.23467	0.686398
6	-30.0599	-739.003	24.32144	85.83057	44.98965	0.942503
7	-38.7728	15171.24	17.09727	65.71336	28.22753	0.911814
8	-34.2337	6882.537	20.83355	76.28863	36.96003	0.823895
9	-34.7439	7814.167	20.03402	75.18035	35.97852	0.815328
10	-35.0596	8390.733	20.31216	74.49128	35.37108	0.847724
11	-31.2437	1422.713	23.2127	83.17009	42.71219	0.890661
12	-33.5976	5721.055	21.65859	77.42823	38.1837	0.59808
13	-33.793	6077.834	21.68704	76.34333	37.80782	0.705027
14	-33.9455	6356.335	20.65741	77.17123	37.51441	0.707122
15	-34.4261	7233.916	20.33019	75.58433	36.58984	0.761759
16	-32.0974	2981.675	22.96716	80.98078	41.06976	0.734344

17	-35.8391	9814.151	19.33463	72.51462	33.87145	0.744205
18	-37.4569	12768.23	18.16254	68.77927	30.7592	0.917268
19	-33.1794	4957.461	21.68115	79.02861	38.98818	0.8981
20	-30.5489	153.913	23.98381	84.75702	44.04892	0.902898
21	-33.4197	5396.253	21.39733	78.5669	38.5259	0.855767
22	-39.0207	15623.87	16.68619	64.96955	27.75066	0.984518
23	-39.222	15991.4	16.65926	64.48988	27.36346	0.910749
24	-32.4248	3579.39	22.63017	80.61827	40.44004	0.764943
25	-33.3245	5222.376	21.61679	78.08601	38.70908	0.792001
26	-36.5524	11116.71	20.16747	71.71133	32.49915	0.885192
27	-38.2545	14224.78	16.43471	66.98899	29.22467	0.947739
28	-31.7688	2381.642	23.78256	80.93746	41.70192	0.761986
29	-38.403	14495.97	16.78202	66.61411	28.93896	0.858204
30	-36.3577	10761.05	19.10031	72.01675	32.87385	0.861075
31	-30.1721	-534.074	24.00161	85.1333	44.77375	0.695595
32	-35.6106	9396.866	20.09797	73.51787	34.31108	0.848809
33	-39.4109	16336.39	16	64	27	1
34	-32.9996	4629.141	22.69956	80.43833	39.33408	0.832135
35	-37.0806	12081.21	18.41843	69.53805	31.48301	0.823904
36	-31.0648	1096.052	23.6633	83.14687	43.05634	0.69672
37	-36.6463	11288.05	18.42743	70.53262	32.31864	0.940064
38	-30.7882	590.8891	23.68367	84.04552	43.58855	0.814179
39	-32.0204	2840.917	22.52694	81.37293	41.21805	0.889274
40	-38.705	15047.42	17.12621	65.68833	28.35799	0.901741
41	-35.4252	9058.383	20.25373	73.76446	34.66768	0.819758
42	-34.819	7951.341	19.96356	74.47459	35.834	0.849
43	-36.8706	11697.69	19.01097	70.03006	31.88706	0.593025
44	-36.1488	10379.62	19.41217	71.90513	33.27571	0.934589
45	-32.6596	4008.282	21.68115	79.02861	39.98818	0.8981
46	-30.266	-362.704	24.94322	84.96834	44.5932	0.698181
47	-37.9665	13698.83	17.626	67.29745	29.77878	0.963314
48	-30.5797	210.1753	24.07144	85.33057	43.98965	0.692503
49	-37.1351	12180.64	19.6568	70.54018	31.37825	0.581954
50	-31.5776	2032.496	21.96716	80.98078	42.06976	0.734344

Table G. 3: YachtRide CPU vs. Bit-rate

	$f(1)$	$f(2)$	$x1$	$x2$	$x3$	$x4$
1	-4378.88	22547.08	-0.26978	1.983814	20.45677	-11.4804
2	-259.438	17240.74	0.496243	2.83517	26.04723	-3.92037
3	-462.411	17502.19	0.4585	2.793222	25.77177	-4.29286
4	-9021.72	30797.81	0.210769	-5.14158	11.76428	-20.1804
5	-1006.15	18082.15	0.539287	2.033017	25.16076	-5.27922
6	-3674.46	21736.31	-0.11083	1.985091	21.31095	-10.1955

7	-5268.97	24460.36	-0.03277	-0.38111	18.44105	-13.1737
8	-1792.57	19471.75	0.408045	2.778826	23.69676	-6.75843
9	-6095.92	24801.85	-0.03356	-0.67371	18.08127	-14.6302
10	573.7071	16528.86	0.496243	2.83517	26.79723	-2.42037
11	-2677.17	19704.14	0.272639	3.94169	23.45193	-8.31
12	-6520.16	27751.99	0.105996	-1.44807	14.97317	-15.613
13	558.4105	16554.4	0.496348	2.823104	26.77031	-2.44892
14	-6514.49	26611.02	0.093101	-2.0199	16.17524	-15.5033
15	-7893.23	30349.06	0.101818	-3.86481	12.23705	-18.197
16	-9021.72	30797.81	0.210769	-5.14158	11.76428	-20.1804
17	-5053.7	23304.1	0.076986	0.992593	19.65921	-12.7094
18	-7524	29128.96	0.168035	-4.65398	13.52248	-17.4522