Complexity Management of H.264/AVC Video Compression

Chaminda Sampath Kannangara

A thesis submitted as partial fulfilment of the requirements of
The Robert Gordon University
for the degree of Doctor of Philosophy

October 2006

Abstract

The H.264/AVC video coding standard offers significantly improved compression efficiency and flexibility compared to previous standards. However, the high computational complexity of H.264/AVC is a problem for codecs running on low-power hand held devices and general purpose computers. This thesis presents new techniques to reduce, control and manage the computational complexity of an H.264/AVC codec.

A new complexity reduction algorithm for H.264/AVC is developed. This algorithm predicts "skipped" macroblocks prior to motion estimation by estimating a Lagrange rate-distortion cost function. Complexity savings are achieved by not processing the macroblocks that are predicted as "skipped". The Lagrange multiplier is adaptively modelled as a function of the quantisation parameter and video sequence statistics. Simulation results show that this algorithm achieves significant complexity savings with a negligible loss in rate-distortion performance.

The complexity reduction algorithm is further developed to achieve complexity-scalable control of the encoding process. The Lagrangian cost estimation is extended to incorporate computational complexity. A target level of complexity is maintained by using a feedback algorithm to update the Lagrange multiplier associated with complexity. Results indicate that scalable complexity control of the encoding process can be achieved whilst maintaining near optimal complexity-rate-distortion performance.

A complexity management framework is proposed for maximising the perceptual quality of coded video in a real-time processing-power constrained environment. A real-time frame-level control algorithm and a per-frame complexity control algorithm are combined in order to manage the encoding process such that a high frame rate is maintained without significantly losing frame quality. Subjective evaluations show that the managed complexity approach results in higher perceptual quality compared to a reference encoder that drops frames in computationally constrained situations.

These novel algorithms are likely to be useful in implementing real-time H.264/AVC standard encoders in computationally constrained environments such as low-power mobile devices and general purpose computers.

Acknowledgements

I take this opportunity to record my sincere thanks to all who helped me to successfully complete this work.

First, I wish to thank my supervisor, Dr. Iain Richardson for the constant guidance, encouragement, support and most importantly for giving me the opportunity to carry out this work.

I thank my supervisory team, Dr. Yafan Zhao, especially for the friendship and support and Dr. Tony Miller, for useful discussions, comments and feedback.

My past and present colleagues at the Centre for Video Communications (CVC), Rob, Yafan, Laura, Wei Wei, Joey, Jose, Abhilash, Abhi, Andrew and Kang, with whom I built lasting friendships, helped me in many ways during the past three years.

I would like to thank Prof. Maja Bystrom (Boston University) for critically reviewing my work and providing useful feedback and Mike Nilsson (British Telecom) for the support given to me during the two month project at BT.

I also wish to thank all the volunteers who took part in subjective tests and all others who directly and indirectly helped me during the project.

I take this opportunity to thank my parents, my sisters and finally, I owe much of the success of this work to my wife, Nirosha, for constantly encouraging and supporting me in every possible way through the years.

I dedicate this work to Nirosha and Shanilka Kannangara

Contents

List of Figures

List of Tables

Abbreviations

3GPP	3rd Generation Partnership Project
ACR	Absolute Category Rating
ASO	Arbitrary Slice Order
AVC	Advanced Video Coding
CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CBR	Constant Bit Rate
CDF	Cumulative Density Function
CIF	Common Intermediate Format
DCR	Degradation Category Rating
DCT	Discrete Cosine Transform
DP	Data Partitioning
DPB	Decoded Picture Buffer
DSL	Digital Subscriber Line
DVB	Digital Video Broadcast
DVD	Digital Versatile Disk
FMO	Flexible Macroblock Ordering
FRExt	Fidelity Range Extensions
GOB	Group Of Blocks
HD	High Definition
HDTV	High Definition Television
ISDN	Integrated Services Digital Network
ISO/IEC	International Organization for Standardization, International Electrotechnical Commission

ITU-T	International Telecommunications Union, Telecommunication Standardization Sector
JM	Joint Model (H.264/AVC reference test model codec)
JVT	Joint Video Team
LAN	Local Area Network
MB	Macroblock
MBAFF	Macroblock Adaptive Frame/Field coding
MBE	(Number of) Macroblock Encodings
MC	Managed Complexity
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
MSE	Mean Squared Error
MV	Motion Vector
MVD	Motion Vector Difference
MVP	Motion Vector Predictor
NAL	Network Abstraction Layer
NNS	Nearest Neighbour Search
PC	Pair-wise Comparison Method
PDF	Probability Density Function
PicAFF	Picture Adaptive Frame/Field coding
PPS	Picture Parameter Set
PSNR	Peak Signal to Noise Ratio
QCIF	Quarter – Common Intermediate Format
QP	Quantisation Parameter
RBSP	Raw Byte Sequence Payload
RD	Rate-Distortion
RDO	Rate-Distortion Optimisation

RS	Redundant Slices
RTP	Real Time Protocol
SAD	Sum of Absolute Differences
SA(T)D	Sum of Absolute (Transform) differences
SDSCE	Simultaneous Double Stimulus for a Continuous Evaluation
SDTV	Standard Definition Digital Television
SI	Switching Intra
SIMD	Single Instruction Multiple Data
SNR	Signal to Noise Ratio
SP	Switching Predictive
SPS	Sequence Parameter Set
SSD	Sum of Squared Differences
TCP	Transmission Control Protocol
TMN	Test Model Number (Video codec test model version)
UDP	User Datagram Protocol
UFQ	User Feedback Quality Method
UVLC	Universal Variable Length Codes
VCEG	Video Coding Experts Group
VCL	Video Coding Layer

1 Introduction

1.1 Problem Statement

Advances in digital communications have changed the communication industry over the past decade. Digital TV, DVD video, HDTV, internet video streaming, video conferencing and mobile technology have expanded the boundaries of communication systems to include a rich visual dimension. Video compression has played a significant part in the realization of these technologies by bridging the gap between the demand for quality, performance and limitations of current storage and transmission capabilities.

The function of a video codec (encoder/decoder) is the compression (and decompression) of video image data into a concise form suitable for storage and/or transmission. In the past, most video codecs were implemented on specifically designed hardware platforms, mostly due to the computational complexity of the process requiring a large amount of calculations. However, in recent years general purpose processors have significantly improved in performance and reliability. They have also become less expensive and increasingly available. Therefore, implementation of software only video codecs for real time applications such as video conferencing has become feasible (for example, Apple iChat [1]). However, current software codecs offer limited compression performance and functionality due to limitations in processing resources.

Video codecs running on general purpose processors such as PCs generally compete with other processes for resources. In recent years, hand held devices such as PDAs and mobile phones are becoming quite versatile by supporting multiple functions such as mobile telephony, video recording and messaging, mobile computing and entertainment. Therefore, video codecs running on these devices are also increasingly competing with other processes for processor time. Additionally, processing power is constrained by battery power utilisation for handheld mobile devices.

The new H.264/AVC video compression standard [2] can deliver significantly improved compression efficiency compared with previous video coding standards (up to 50% more) [3] supporting higher quality video over lower bit rate channels. Due to its improved compression efficiency, error resilience features and increased flexibility of coding and

transmission, H.264/AVC has the potential to enable new video services such as mobile video telephony and multimedia streaming over wireless networks [4]. However, the performance gains of H.264/AVC come at a price of increased computational complexity [3] [5]. The processing overhead required to implement H.264/AVC is likely to be a major problem for power constrained mobile devices since higher computational loading leads to increased power consumption. There is therefore a need to develop low complexity implementations of H.264/AVC that offer the performance and flexibility advantages of the standard without an excessive computational cost.

An H.264/AVC compliant video encoder typically carries out a number of encoding processes to remove the spatial, temporal and statistical redundancy of a video signal. A significant amount of compression is achieved by the transformation of each macroblock (basic coding unit corresponding to a 16x16 block of displayed pixels) from the spatial domain to a transform domain, which is a representation of spatial frequency components (or coefficients), followed by the quantisation of transform domain coefficients. The complexity of this process in H.264/AVC is relatively low compared to other encoding functions due to the small transform block size and the availability of efficient integer implementations [6]. To achieve high compression by removing temporal and spatial redundancy, H.264/AVC is equipped with tools to closely predict each macroblock from regions of the current image or previous images that have already been coded. The standard supports a large number of modes and options to carry out this prediction process. This flexible choice of coding parameters leads to efficient compression at the expense of computational complexity because the encoder needs to evaluate a large number of mode options in order to determine the coding parameters that produce maximum compression efficiency.

Low complexity or variable complexity algorithms have been proposed for a number of aspects of the H.264/AVC encoding process (chapter 7). However, the low complexity or variable complexity algorithms developed to date lack the ability to control the computational complexity in an adaptive and scalable manner. Scalable complexity control (reducing or increasing the complexity in small steps) is especially important for software encoders on platforms with limited processing resources, for example where there may be other processes competing for the same processor cycles or where the complexity may have to be reduced in order to prolong the battery life of a mobile device.

This scalable control of computational resources should ideally have minimal effect on the compression performance of the encoder.

1.2 Research Objective

The aim of this research is to develop novel algorithms to effectively manage the computational complexity of an H.264/AVC encoder. These algorithms should enable the encoder to make efficient use of available processing resources to maximise the rate constrained video quality.

This research is particularly aimed at managing the complexity of the encoder, because, (a) an increasing number of low-power handheld devices such as mobile phones and PDAs carry out video capturing, where the captured video needs to be compressed before storage or transmission and, (b) compression efficiency depends on the coding tools and the decision making process employed by the encoder, which carries a significantly higher computational burden compared to decoding.

1.3 Project Outline

The research project was structured into several stages in order to achieve the main objective within the specified period. Figure 1.1 shows the structure of the project.

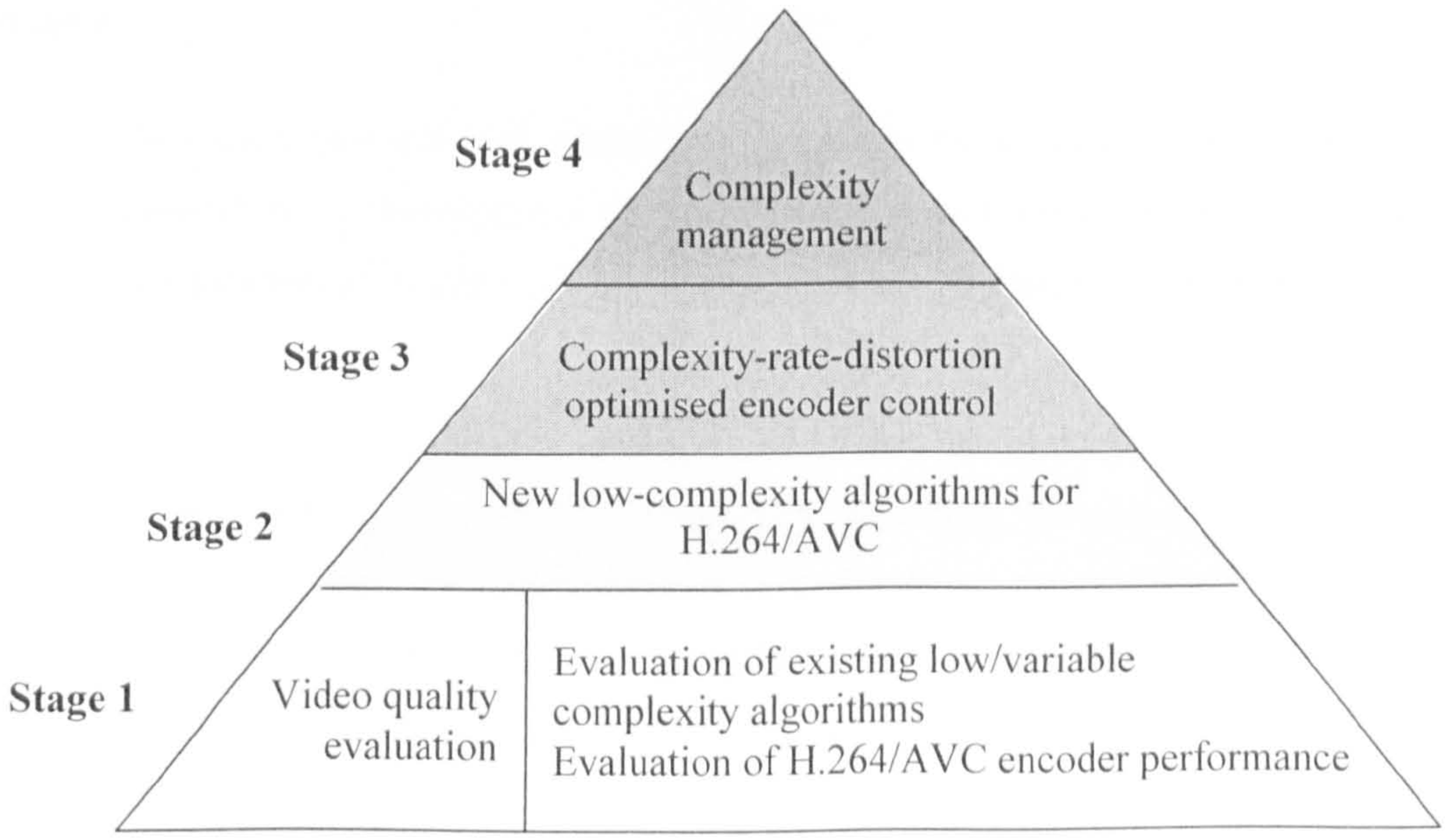


Figure 1.1: Evolution of the research project

The pyramid structure represents the evolution of the project towards its final goal. The layers represent different stages of the project as the work is progressed upwards from the bottom of the pyramid. The lower layers provide the necessary foundation for the upper layer stages.

Each stage of the project (starting from the bottom of the pyramid) is briefly described as follows:

Stage 1

1. Investigate and evaluate video coding standards and analyse the rate-distortion performance of an H.264/AVC encoder with different coding parameters and identify the main contributors to the computational complexity of the encoder.

2. Study subjective and objective video quality measurement techniques to explore the possibility of using such methods to evaluate new low-complexity algorithms.

3. Investigate and apply existing variable/low complexity algorithms to an H.264/AVC codec and analyse the trade-off between complexity, bit rate and video quality.

Stage 2

4. Develop a new reduced complexity algorithm for an H.264/AVC encoder and evaluate the performance of the algorithm. The new algorithm should reduce the computational complexity with minimal loss in rate-distortion performance.

Stage 3

5. Further develop the reduced complexity algorithm (developed in stage 2), in order to control the computational complexity of the encoder. Apply rate-distortion-complexity optimisation techniques to ensure that best rate-distortion performance is achieved at each complexity level.

Stage 4

6. Develop a complexity management framework for an H.264/AVC encoder to manage the encoder complexity in order to maximise rate-constrained video quality with limited processing resources.

1.4 Novel Contributions and Publications

This research aims to develop techniques to manage the computational complexity of an H.264/AVC encoder. Novel algorithms were developed during the project for complexity reduction, complexity control and management of the encoding process. Key contributions of this work to the advancement of video coding can be summarised as follows:

- The development of a low complexity macroblock skip prediction algorithm for an H.264/AVC encoder. Early prediction of skipped macroblocks is made by estimating a Lagrangian rate-distortion cost function. This novel algorithm significantly differs from existing algorithms (discussed in chapter 7) because, (a) this algorithm is based on a firm theoretical foundation in contrast to existing, mostly heuristic approaches, (b) the algorithm does not depend on any arbitrary thresholds (c) this algorithm is adaptive to different video sequence statistics. The development of this new algorithm led to a journal publication [7] and a conference paper [8].

- The development of an adaptive model for the Lagrange multiplier parameter associated with rate. The Lagrange multiplier is modelled as a function of quantisation parameter and sequence statistics. Techniques widely used in current practice model the Lagrange multiplier only as a function of quantisation parameter. Therefore, these current techniques do not achieve optimal performance for video sequences with different statistics. This model is described in [7] and [8] in detail.

- The novel classification of sequence statistics by an activity factor (F). The activity factor 'F' is an estimation of the amount of motion and detail present in a particular area such as a macroblock, number of frames or in a complete video sequence (also introduced in [7, 8])

- The achievement of scalable computational control of the encoding process by extending a Lagrangian cost function to incorporate computational complexity. A target level of computational complexity is met by directly controlling the Lagrange multiplier associated with complexity using a feedback algorithm. Existing techniques in the literature do not demonstrate effective 'complexity-control' of the encoding process. This work was presented in a conference paper [9].

- The development of a complexity management approach that uses a real-time encoder control algorithm, in conjunction with the complexity control algorithm to manage the computational resources utilised by an H.264/AVC encoder (presented in [10]). This is a significant contribution because the new approach addresses the problem of managing the computational complexity in a real-time computation constrained scenario, where the objective is to maximise perceptual video quality. The current literature does not provide any effective solution to this problem. Additionally, this approach can be used as a complexity management framework for other complexity control algorithms.

- The demonstration that, higher perceptual quality ratings can be achieved by the managed complexity encoder, compared to an encoder that reduces the frame rate when processing resources are limited. The managed complexity encoder maintains smooth video frame rate whilst ensuring that the frame quality is not degraded unacceptably, when available processing power is limited. This work was originally presented in [10] and also led to a UK patent application [11] and is under review for a journal [12].

- The development of a new subjective video quality assessment method which incorporates user feedback to control the video quality. This method produces comparable results in a fraction of the time normally required to carry out a standard subjective video quality assessment test. This method was published in a journal [13] and also led to a European patent application [14].

A list of publications can also be found in appendix A. Although this research is mainly focused on the encoder, some work has been carried out on the low-complexity implementation of a H.264/AVC decoder. Appendix B contains an abstract of a multi-

threaded implementation of a H.264/AVC decoder for real-time High Definition (HD) decoding in a multiprocessor PC. This work was carried out at BT Group Chief Technology Office (formerly known as BT Exact) as a placement project.

1.5 Organisation of the Thesis

The thesis is organised as follows:

Chapter 2 – This chapter provides some essential background knowledge on video coding. The key concepts and fundamental terms used in video compression are introduced. Main functions of a typical block-based video codec are briefly explained. A brief introduction to video coding standards is also presented.

Chapter 3 – Provides an overview of the H.264/AVC video compression standard. The structure of the standard and video coding tools that contribute to high compression efficiency, flexibility and robustness are briefly explained. This chapter also explains why a large amount of computational resources are needed to implement the H.264/AVC standard.

Chapter 4 – Introduces the rate-distortion optimisation problem. This chapter explores the use of the Lagrange multiplier method to optimise the rate-distortion performance of video encoders and explains the theory and assumptions relating to practical application of the Lagrange multiplier method in video compression. Algorithms developed during this research project are primarily based on the Lagrange multiplier method. Therefore, this chapter plays a very important role in understanding later chapters that describe these algorithms.

Chapter 5 – This chapter explains the experimental methods used in this work.

Chapter 6 – Presents a new subjective video quality assessment method which incorporates user feedback to control the video quality. Although this work is slightly outside the main research theme, the new method proved to be useful in rapidly assessing subjective quality of different coding algorithms or parameters.

Chapter 7 – The first part of this chapter presents a performance evaluation of different tools and coding parameters of the H.264/AVC standard with respect to rate, distortion and computational complexity. The second part constitutes a review of some of the low

complexity video coding algorithms developed for H.264/AVC and previous coding standards such as H.263 [15]. Finally, this chapter establishes the case for low-complexity implementations of H.264/AVC, identifies the lack of techniques to effectively manage the computational complexity and explains the motivation behind this work.

Chapter 8 – This chapter presents the performance of a macroblock skip-prediction algorithm (an algorithm that predicts macroblocks that are normally skipped after processing), which is adapted from a threshold based algorithm developed for H.263. This part of the work was carried out as an evaluation of existing complexity reduction algorithms.

The main algorithms developed during the research project are described in chapters 9, 10 and 11. These chapters correspond to the work carried out during stage-2, 3 and 4 of the project respectively.

Chapter 9 – Describes a new complexity reduction algorithm for H.264/AVC which uses Lagrangian cost function estimation. Computational savings are achieved by predicting skipped macroblocks prior to motion compensation by estimating a Lagrangian cost function. The Lagrange multiplier is adaptively modelled based on local sequence statistics. This is the main low-complexity algorithm developed for H.264/AVC and further developments have been primarily based on this algorithm.

Chapter 10 – Complexity reduction achieved by the algorithm described in chapter 9 depends on the video sequence statistics. This chapter explains a complexity control algorithm that extends the Lagrangian cost function by incorporating computational complexity in order to achieve a target computational complexity level.

Chapter 11 – Describes a complexity management framework for an H.264/AVC encoder running on process/power constrained environment. This new approach uses a real-time coder control algorithm in conjunction with the complexity control algorithm (described in chapter 10) in order to maintain smooth frame rate without unacceptable degradation of frame quality.

Chapter 12 – This final chapter contains the discussion and the conclusion. A summary of the main algorithms and a critical review of the results are presented. The advantages and disadvantages of proposed methods are discussed. Ideas for further investigation are

also presented. Finally, the thesis is concluded by emphasising the relevance of this work to the research problem and the original contributions made.

Appendix A – Contains a list of publications related to this research.

Appendix B – Provides an overview of a multi-threaded implementation of an H.264/AVC software decoder for a multi-processor platform. This software decoder implementation is aimed at real-time decoding of high definition content using a dual-processor PC.

Appendix C – Contains a statistical analysis that compares the results of the new subjective quality method (explained in chapter 6) with the results obtained using a standard subjective quality evaluation method.

2 Video Coding Background

2.1 Introduction

This chapter provides some essential background information on video coding. It starts by briefly describing digital video sampling, colour spaces and common picture formats. The main concepts related to block based video coding are presented by giving an overview of the basic functions of a typical block based video codec. A brief introduction to some of the popular video compression standards is also presented.

2.2 Digital Video

Digital video consists of a stream of images captured at regular time intervals. The images are represented as digitized samples containing visual (colour and intensity) information at each spatial and temporal location.

2.2.1 Sampling & Resolution

Figure 2.1 shows the sampling process of digital video. The number of horizontal and vertical samples (which can also be referred to as picture elements or in short 'pixels') in the image determines the spatial resolution of the image. The frequency at which each image is captured (temporal sampling) determines the motion smoothness of the video.

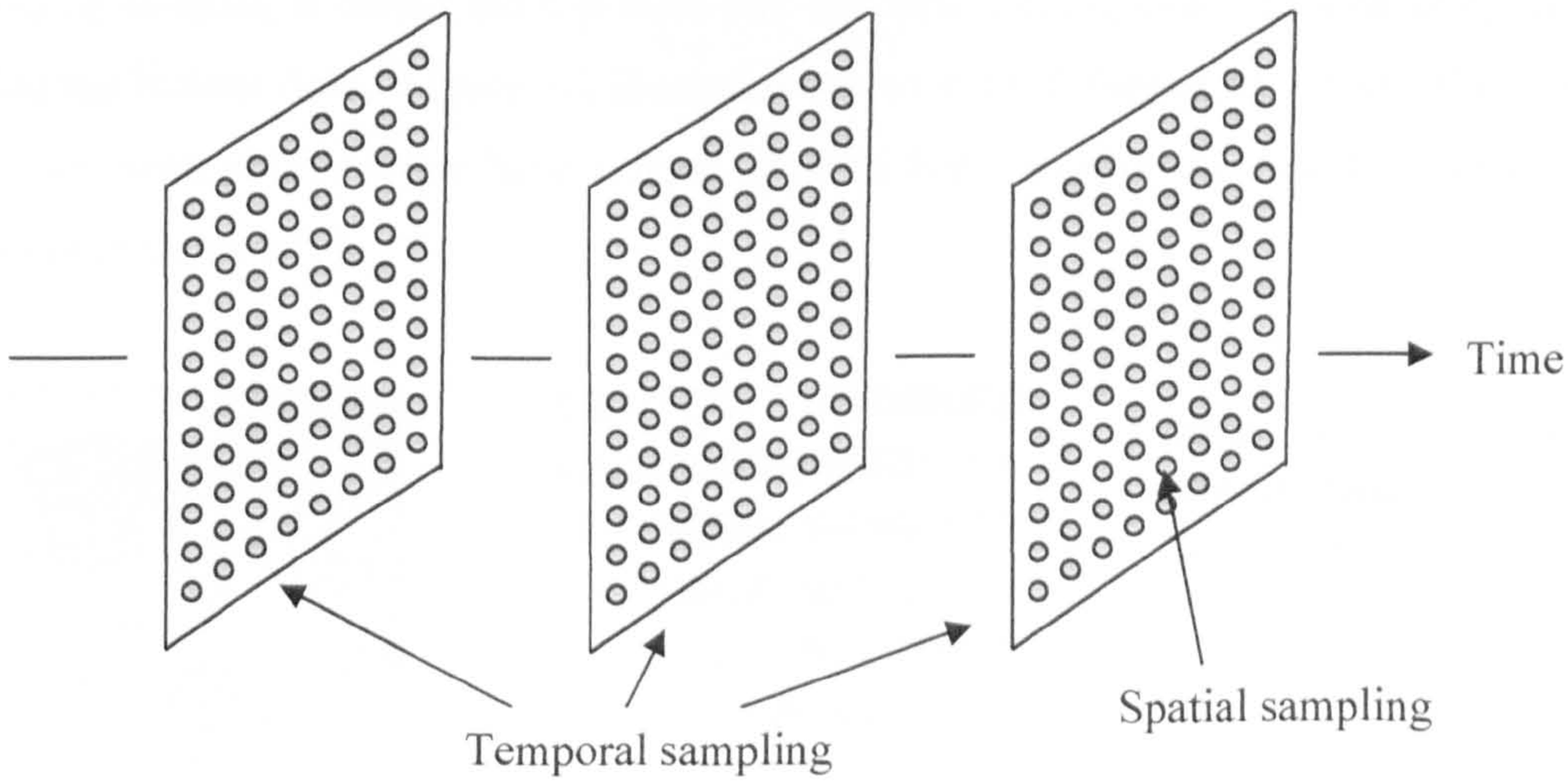


Figure 2.1: Digital video sampling

Typical temporal sampling frequencies (frame rate) are 25 Hz and 30 Hz . The frame rate determines the motion smoothness of the video, where motion appears smoother at higher frame rates.

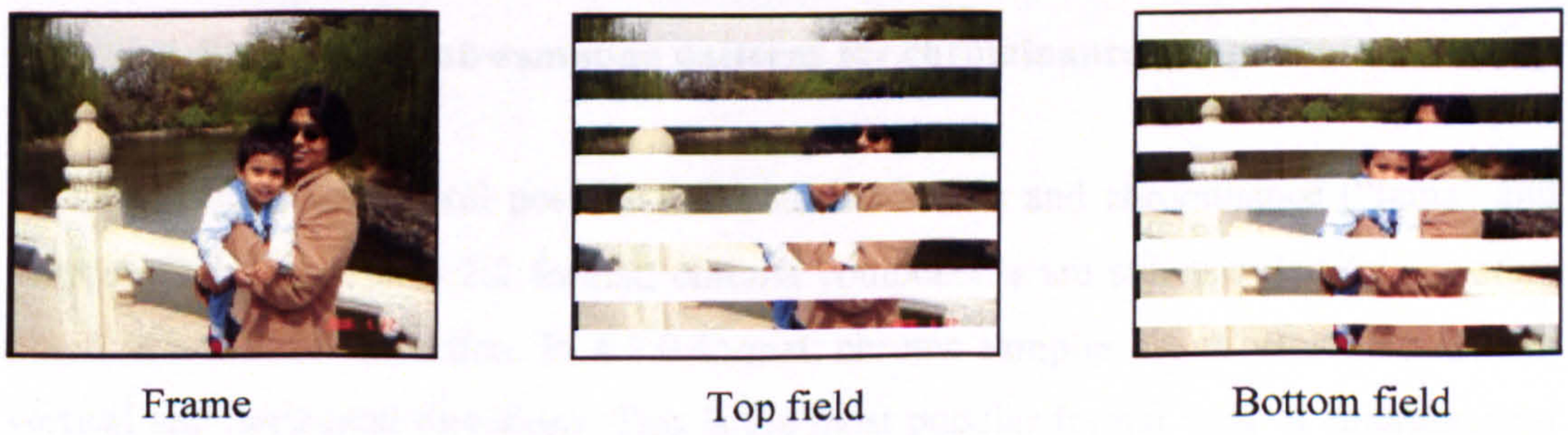
In digital video processing, different spatial resolutions are used depending on the target application. Some of the most widely used formats based on CIF (Common Intermediate Format) are shown in Table 2.1.

Table 2.1: CIF based formats and resolutions

Format name	Pixel resolution (Horizontal x Vertical)
SQCIF (Sub-Quarter-CIF)	128 X 96
QCIF (Quarter–CIF)	176 X 144
CIF	352 X 288
4CIF	704 X 576

2.2.2 Frames and Fields

A video signal can be sampled in either frames (progressive) or fields (interlaced). In progressive video, a complete frame is sampled at each time instant. In interlaced video only a half of the frame is captured (either odd or even rows of samples) at a particular time instant which are called fields. The field which has the first row (and subsequent odd rows) of samples is called the top field and the field having even rows of samples are called the bottom field. Figure 2.2 illustrates the concept of frames and fields. (However, note that rows in the picture have a depth of more than one pixel in order to illustrate the concept more clearly).



Frame Top field Bottom field

Figure 2.2: Progressive and interlaced video

2.2.3 Colour Spaces

Visual information at each sample point may be represented by the values of three basic colour components Red (R), Green (G) and Blue (B). This is called the RGB colour space. Each value is stored in an 'n'-bit number. For example; an 8-bit number can store 256 levels to represent each colour component.

The YCrCb colour space is widely used to represent digital video. The luminance component 'Y' is extracted using a weighted average of the three colour components R, G and B. The components Cr and Cb are called the chrominance (or colour difference) components. Cr is the red chrominance component (Cr = R - Y) and the blue chrominance component is Cb, where Cb = B - Y. The derivation of YCrCb colour space from RGB colour space can be found in [16]. The human visual system has less sensitivity to colour information than luminance (light intensity) information [17]. Therefore, with the separation of luminance information from the colour information, it is possible to represent colour information with a lower resolution than the luminance information (see Figure 2.3).

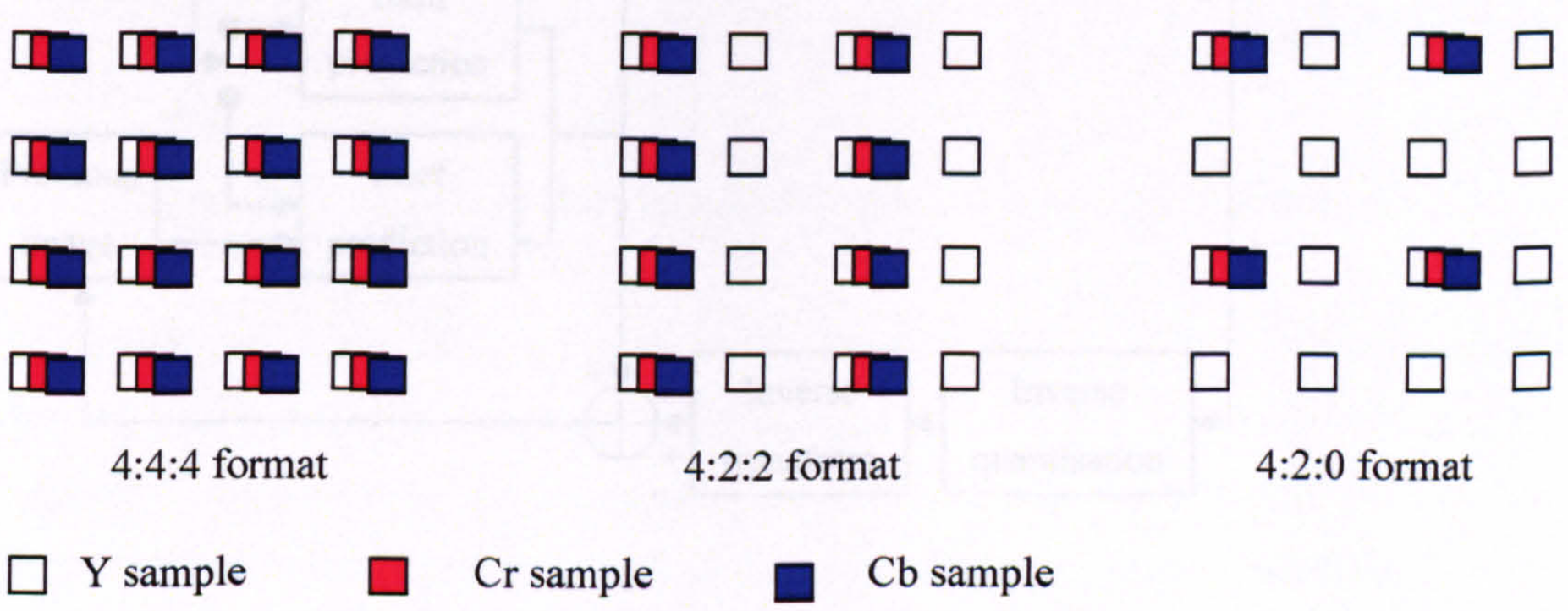


Figure 2.3: Sub-sampling patterns for chrominance components

In 4:4:4 format, each pixel position has both luminance and chrominance ("luma" and "chroma") samples. In 4:2:2 format, chroma components are sub-sampled (every other pixel) in horizontal direction. In 4:2:0 format, chroma samples are sub-sampled in both vertical and horizontal directions. This is the most popular format used in entertainment quality applications such as DVD video because the human eye does not easily recognise

missing colour information. The work carried out in this project deals with 4:2:0 sampled video.

2.3 Block Based Video Coding

In block based video coding, the basic unit of coding is a block containing NxN (e.g. 16x16) array of luma samples and corresponding chroma samples. The image is divided into an integral number of blocks and processed in raster scan (from left to right of each row and top to bottom row by row) order. Figure 2.4 shows a block diagram of a typical block based video encoder. The encoder has two data flow paths. The forward path (marked with solid lines) represents the encoding process of coding units and the reverse path (grey lines) shows the decoding (reconstruction) of the coded units within the encoder. Major elements of block based encoding are inter and intra prediction processes, transform, quantisation and entropy coding. These are described below.

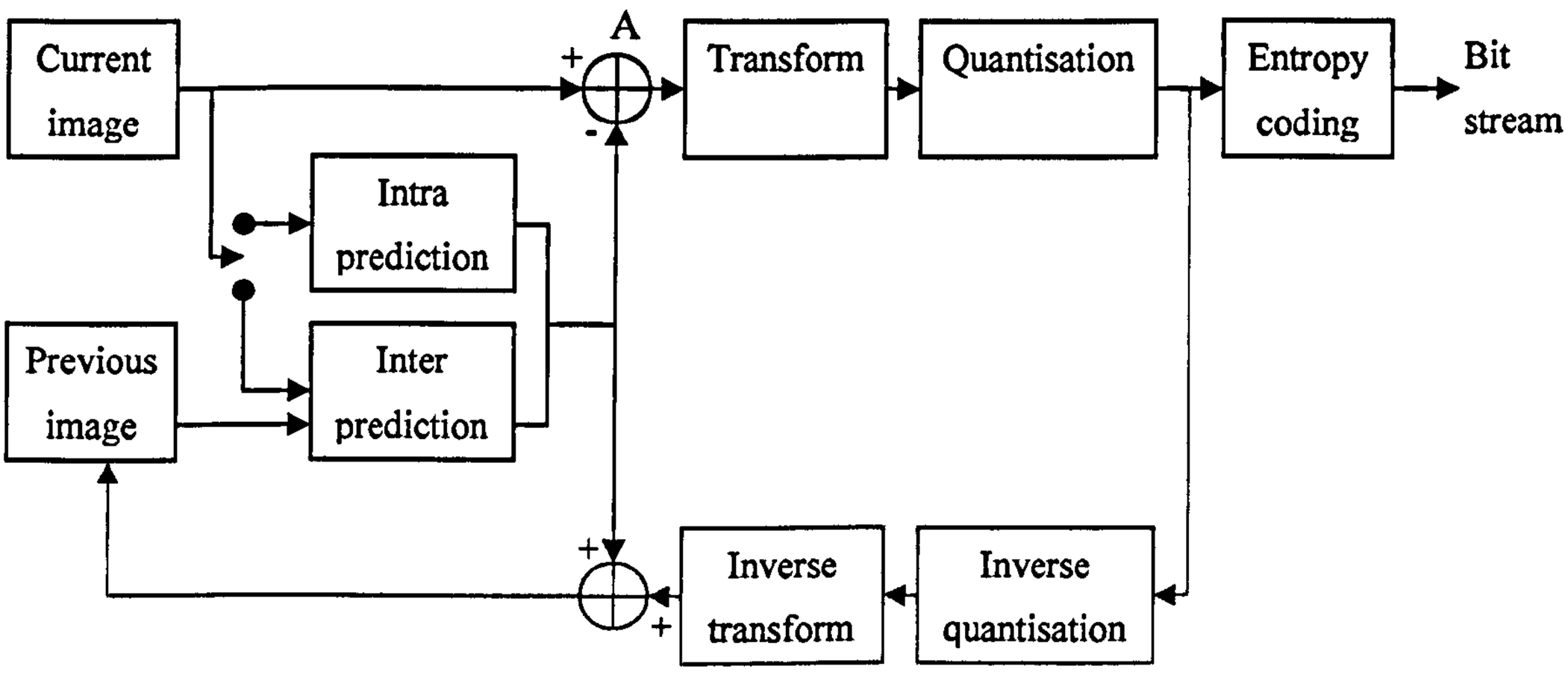


Figure 2.4 Block based encoder block diagram

2.3.1 Intra Prediction

Block based video encoders use prediction as a tool for removing redundant information. A prediction signal is obtained from previously coded samples for the coding unit and it is subtracted from the original coding unit to create a residual signal that has much less data than the original coding unit. It is the residual signal that is encoded and transmitted (see node 'A' in Figure 2.4) to the decoder. The decoder obtains the same prediction

signal using previously decoded samples, decodes the residual signal and adds them together to reconstruct the coding unit.

In intra prediction each coding unit is predicted using the surrounding pixels (which have been already coded and decoded) in the same image. Intra coding is used in the first image of a sequence. Intra coding is also very useful in coding uniform regions where surrounding pixels of the block has similar value as the pixels inside the block. Intra prediction is only used in recent video coding standards such as H.263 (Annex I) [15] and H.264/AVC [2]. Figure 2.5 shows an example intra prediction of a 2x2 sample block using the average (or 'DC' value) of surrounding sample values (shown in shaded area) that have already been coded.

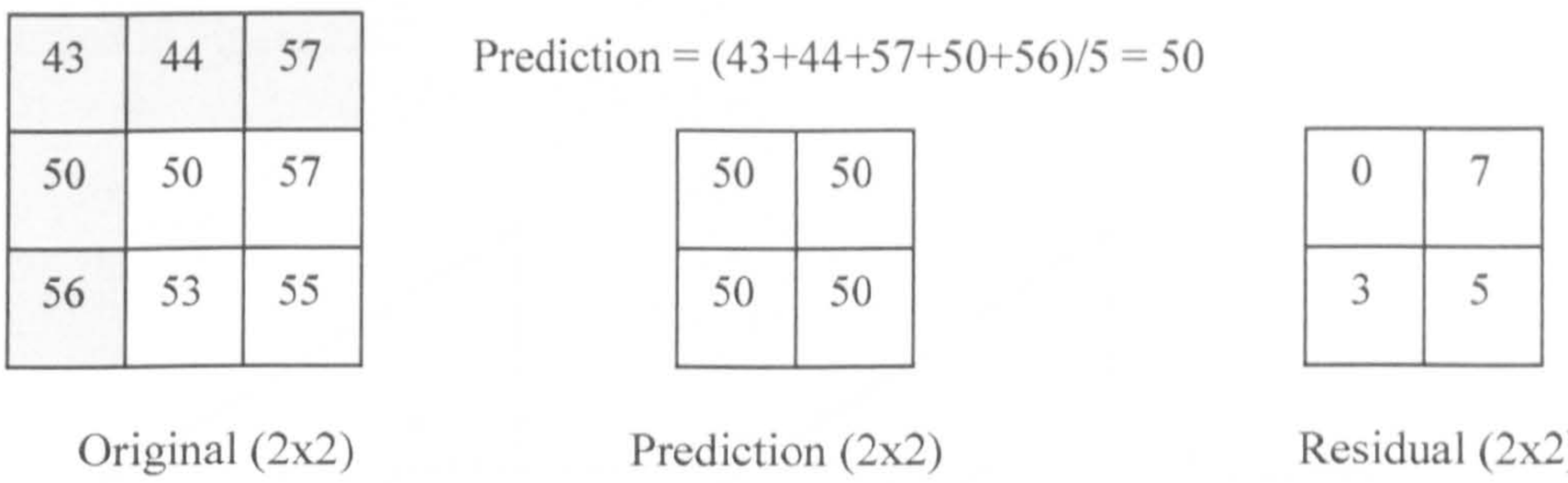


Figure 2.5 Example of intra prediction of a 2x2 block of luma samples

The advantage of using intra prediction (or predictive coding in general) can be explained using the above simplified example. If the samples in the original 2x2 block were to be transmitted, six bits are needed to represent each sample value and therefore a total of 24 bits are needed to transmit the samples in the block. However, if the spatial redundancy between the neighbouring samples is exploited (taking into account the fact that neighbouring samples have similar values as the samples in the original block), the residual block can be obtained by subtracting the average (or the 'DC' value) of neighbouring samples from the original samples. Now the smaller numbers (the difference of each sample from the average) can be represented with 3 bit numbers and only a total of 12 bits are needed to transmit the samples in the block.

2.3.2 Inter Prediction

In general, consecutive video images are very similar to each other and the differences mostly arise due to the movement of the objects in the video scene. Inter prediction is used to remove this temporal redundancy of video images. The prediction signal of a coding unit is obtained from a previously encoded and reconstructed image, for example the preceding image. The aim is to find a good match for the current block from the previously coded image. For example, if the current block forms a part of a moving object in the video scene, a good match can be found if the same part of the moving object is found from the previous image. This can be done by following the motion of the object over time between the two images. Usually it is very difficult to find an exact match by precisely following the motion. However, a reasonably accurate match can be found by searching for a similar block within a restricted region of the image. This process is illustrated in Figure 2.6.

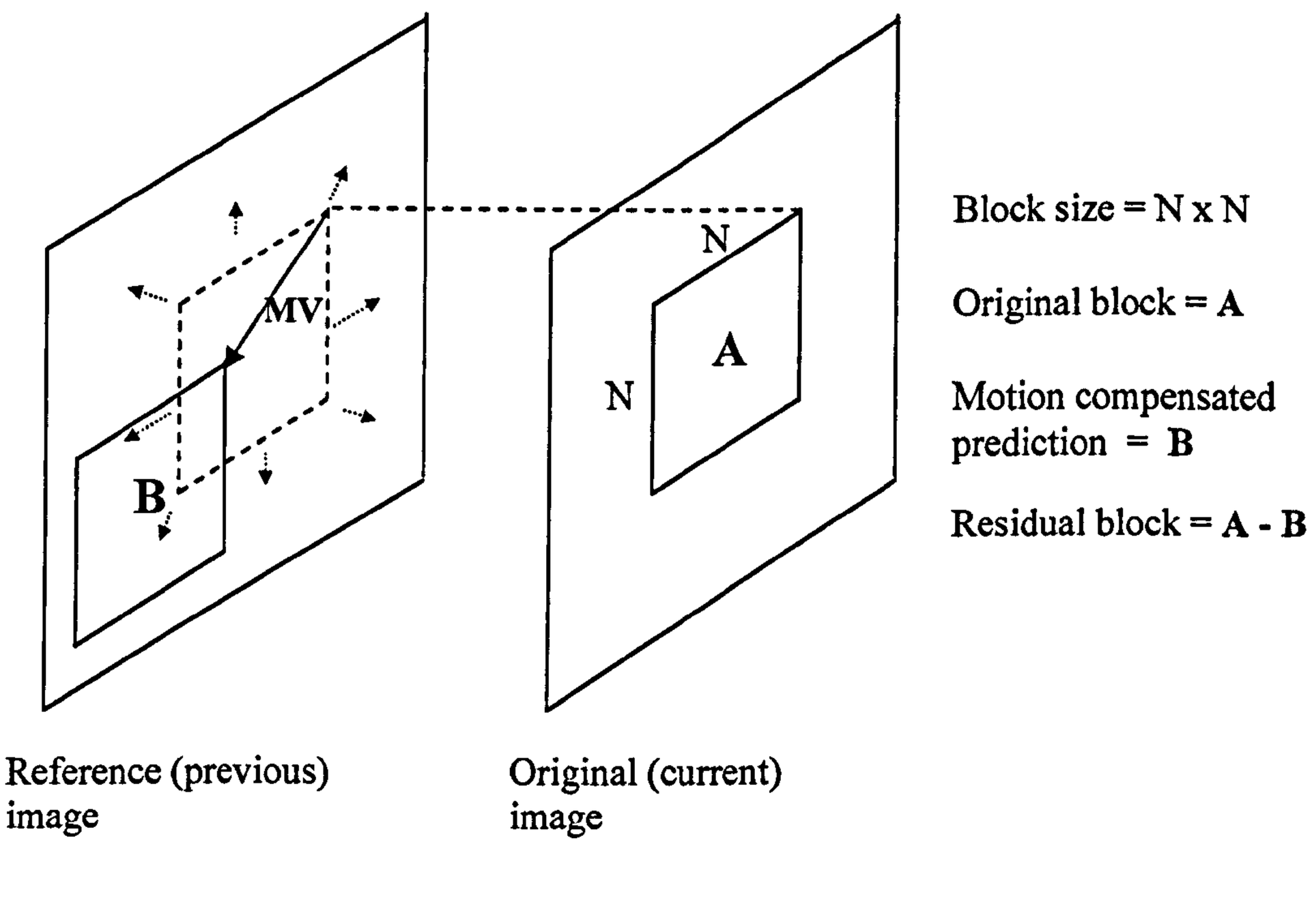


Figure 2.6: Motion estimation and compensation of an "m x n" block

Common terms related to inter prediction process can now be introduced as follows:

Reference Image – is the previously encoded and reconstructed image that is used for the prediction of blocks in the current image.

Motion Estimation – is the process of searching and finding the closest matching block (B) from the reference image to the current block (A).

Motion Compensation – is selecting the best matching block as the prediction and obtaining the residual by subtracting the prediction from the original block.

Motion Vector (MV) – is the vector representing the displacement (horizontal and vertical) of the matching block from the position of the original block. Some video coding standards (discussed later) support sub-pixel accurate (e.g. half pixel accurate) motion vectors. In this case the motion vector is pointing to a block of pixels in between the actual pixels in the image. Therefore, the prediction block is obtained by interpolating the actual pixels in order to obtain a block of pixels at a sub pixel position.

For inter predicted coding units, the residual signal is encoded and transmitted to the decoder along with the motion vector values. The decoder uses the motion vector values to find the correct prediction block and the decoded residual is added to reconstruct the coding unit.

2.3.3 I, P and B Pictures

In I-Pictures all the coding units are predicted using intra prediction only (without using previously coded pictures for prediction). These are used for the first picture of a sequence and are also used as random access pictures for reversing and fast forwarding without the need for decoding all the pictures. P-Pictures are inter predicted pictures with the reference as the nearest previously coded picture. They can not be used for random access, because of the dependency on previously coded pictures. However, they are used as reference pictures. B-pictures are bi-directionally predicted pictures which require two reference pictures for inter prediction, one from past and one from future in display order. They typically have high compression efficiency; however they are not used for reference and can not be used for random access.

2.3.4 Transform Coding

The residual block (from node 'A' in figure 2.4) is transformed from spatial domain into transform domain using a two dimensional block transform process where, the transform of a "NxN" block of image residuals results in a "NxN" block of transform coefficients. These transform coefficients represent the residual image block as magnitudes of basis patterns of the transform (typically a representation of different spatial frequency components). The transform needs to be reversible (inverse transform) in order to obtain the image residuals from the transform coefficients. The transform process is similar to the transformation of time domain signals into the frequency domain using a Fourier Transform in signal processing. The transform in itself does not achieve any compression; however, it serves the following two purposes:

1. Energy compaction – Concentrate most of the energy within a small number of large coefficients.

2. De-correlation of data – The transform coefficients should ideally have minimal inter-dependency between each other.

Therefore it is possible to reconstruct (inverse transform) the original block using only the significant transform coefficients by setting the insignificant coefficients to zero (typically through quantisation as explained shortly) without significantly affecting the perceptual quality. For example, setting some small high frequency coefficients to zero would lose some high frequency spatial detail in the image block; however (because transform coefficients are de-correlated), this does not distort other more prominent features of the image block such as average brightness or low frequency gradients. This is similar to low pass filtering an audio signal to obtain different quality signals at different frequency bandwidths.

The most widely used block based transform in image and video compression is the Discrete Cosine Transform (DCT) [18] where the transform of a 'NxN' block is given by:

$$F_{x,y} = C(x)C(y)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f_{i,j} \cdot \cos\left(\frac{(2i+1)x\pi}{2N}\right)\cos\left(\frac{(2j+1)y\pi}{2N}\right) \qquad 2\text{-}1$$

Where,

$$C(n) = \begin{cases} \sqrt{\frac{1}{N}}, for & n = 0 \\ \sqrt{\frac{2}{N}}, for & n > 0 \end{cases}$$

$f_{i,j}$ represents the samples of input block where $f_{0,0}$ is the top left hand corner sample of the block. $F_{x,y}$ represents the coefficients of the transform block where $F_{0,0}$ (the 'DC' coefficient or the average of the samples in the 'NxN' block) is at the top left corner of the block.

2.3.5 Quantisation

Quantisation is the process of converting a continuous range of values to a finite range of discrete levels. For example, in digital video an 8-bit colour sample is obtained by approximating the signal level (or the voltage level) of the colour component from the camera into one of the finite discrete levels, in this case 256 levels that can be represented by an 8-bit number. Some of the colour information is lost and can not be recovered due to approximation and therefore more levels are needed (e.g. 10-bits to approximate the same continuous range into 1024 levels) to retain more information.

In video compression, lossy compression is achieved by quantisation. The quantisation process consists of two stages; forward quantisation is carried out during encoding and rescaling is carried out during decoding. The two stages are also referred to as quantisation and inverse quantisation. In forward quantisation, the original transform coefficient value is typically divided by the "quantisation step" and rounded to the nearest integer. Information is lost during the rounding process. The quantisation step size represents the interval between discrete quantisation levels (the larger the quantisation step size, the larger the rounding error). These integer values are transmitted to the decoder along with the quantisation step used. Rescaling is carried out at the decoder, where the received integer is multiplied by the quantisation step in order to obtain the actual quantised transform coefficient (the original value approximated into a multiple of the quantisation step). However, note that common use of the phrase "quantised transform coefficient" refers to the integer value obtained after forward quantisation. The quantisation step size determines the amount of compression achieved because, dividing by a large quantisation step will result in more small and zero coefficients. Lower bit

rates can be achieved at higher quantisation levels at the expense of a large approximation error and therefore higher image distortion.

Transform and quantisation processes of a 4x4 block of image residual data are shown in Figure 2.7 .

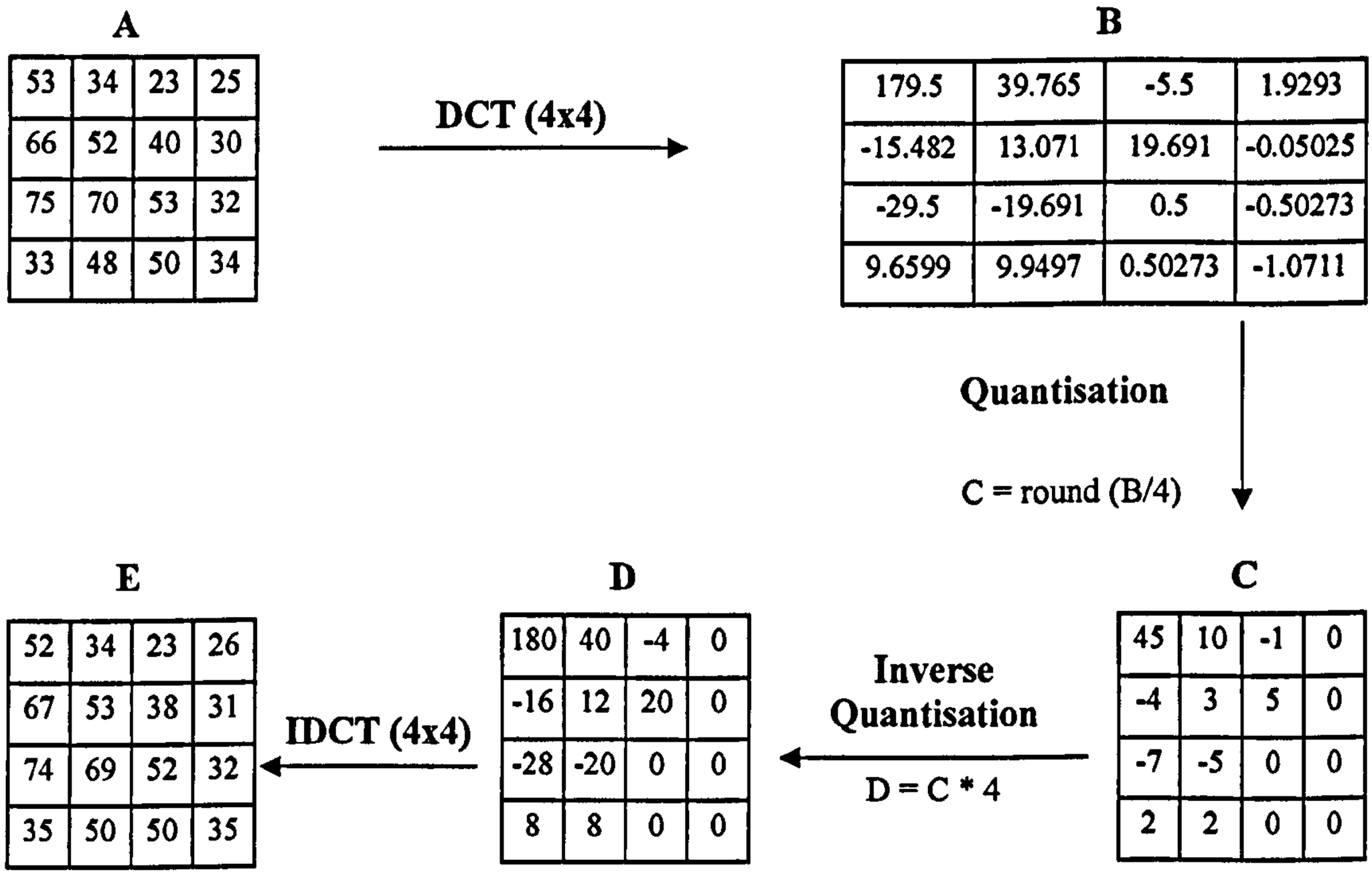


Figure 2.7: Forward and inverse transform & quantisation of a 4x4 block

Note the following points:

- DCT transform of block A (using equation 2-1) is given by block B. The inverse transform of block B results in block A.

- Most of the energy in B is concentrated on the top left corner of the block.

- Block C is transmitted to the decoder. C is obtained by "quantising" or more accurately forward quantising (quantisation step = 4) the transform coefficients in block B.

- Now the data to be transmitted (block C and the quantisation step) is very small compared to the original block A.

- Inverse quantisation (rescaling) is carried out on C at the decoder to obtain block D.

- Due to quantisation loss, transform coefficients are different from the original. Therefore, D≠B.

- Inverse transform of D results in E. Note that E is not identical to A; however, the difference is quite small because quantisation is carried out in the transform domain where the energy is concentrated in few significant coefficients.

Note that some video coding standards such as H.264/AVC (High Profiles) [19], support perceptually weighted quantisation matrices to account for the spatial frequency sensitivity of the human visual system. Other video coding standards such as H.263 [15] only support scalar quantisation which is similar to the scaling operation shown in the above example.

The forward quantised coefficients in block C (to be transmitted to the decoder) are re-ordered and coded before the entropy coding stage. The forward quantised coefficients in block C are re-ordered into a one dimensional array, typically by zigzag scanning where, the coefficients are read from the top left corner to the bottom right corner of the block in an effort to group together non zero and zero coefficients separately. Then the array is coded into a compact representation (typically in run level pairs, where run is the number of zeros before a non zero value which is indicated by the level, along the one dimensional array) in order to minimise the number of bits required to represent the numbers.

2.3.6 Entropy Coding

The encoder needs to transmit data such as residual quantised transform coefficients, quantisation values, motion vectors and other overhead information such as coding parameters to the decoder. Entropy coding is carried out to reduce the statistical redundancy of the transmitted data. This is a lossless compression technique where data with high probability of occurrence is coded with a smaller number of bits and data with lower probability of occurrence is coded with a larger number of bits. Commonly used entropy coding methods are Huffman coding and Arithmetic coding [20].

2.3.7 Decoder

The decoding process is identical to the reverse path of the encoder in Figure 2.4. The bit stream received from the encoder is first entropy decoded and then, inverse quantised and inverse transformed to create the residual. This residual is added to the prediction signal to construct the image. Due to lossy coding (quantisation) the reconstructed image is not identical to the original image; however, the reconstructed images of the encoder and decoder are identical to each other because the decoding process at the decoder and the processing carried out by the reverse path of the encoder are identical. Since the reconstructed images are used for inter prediction, they need to be identical at the encoder and decoder. Otherwise any error will propagate into the inter prediction of the next frame and added to the subsequent frames, thus progressively distorting the reconstructed images at the decoder.

2.4 Video Coding Standards

Standardization of video coding technology has played a major role in the advancement of digital video communication technologies over recent years [21]. Standardization enables interoperability between different manufacturers and is a major requirement for the communications industry. The two international standardization bodies are namely,

1. Video Coding Experts Group (VCEG) of International Telecommunications Union – Telecommunication Standardization Sector (ITU-T) and

2. Motion Picture Experts Group (MPEG) of International Organization for Standardization – International Electrotechnical Commission (ISO/IEC)

The standards released by the ITU-T have been named H.26x series and ISO/IEC has released the MPEG series of standards. The MPEG standards have been mainly aimed at media storage and distribution while the H.26x standards have been aimed at real-time video communication applications. Some of the popular standards are named below (the descriptions are based on the overviews found in [21] and in [22]).

MPEG-1 [23]

The draft MPEG-1 standard was released in 1993. Although this is a generic video coding standard (not constrained for a specific application) it was primarily designed for storage

on digital media such as CD-ROM supporting bit rates up to 1.5 Mbit/s. The standard employs a block based hybrid coding (predictive and transform coding) algorithm similar to block based video coding described in section 2.3. The standard supports flexible picture types: I-Pictures, P-Pictures and B-Pictures in order to provide good compression efficiency and added functionality such as fast forward and rewind.

MPEG-2 [24]

The MPEG-2 standard (1995) was aimed at broad variety of applications such as media storage, satellite terrestrial TV broadcasting. It builds on MPEG-1 algorithm including new tools for better quality and functionality such as interlaced video and scalable video coding[1] for applications such as digital TV and HDTV. This is the first standard to introduce the concept of "profiles" and "levels" as means of implementing compliant decoders that support only a subset of syntax (profiles, e.g. particular set of tools) with restriction on capability (levels) such as maximum supported bit rate.

MPEG-4 Visual [25]

The MPEG-4 Part 2: Visual (1998) supports a wide variety of applications including internet video streaming and digital TV broadcasting as well as applications with combined real world video scenes and computer generated graphics. The standard can

---

[1] In scalable video coding, the video is encoded in two or more layers of bit streams, for example a base layer and an enhancement layer(s). In spatial scalable coding, the base layer bit stream may be used to decode a QCIF resolution video whilst the enhancement layer (combined with the base layer) may provide the bits necessary to decode a CIF version of the sequence. Similarly, temporal scalability may provide an enhancement layer in order to decode the sequence at a higher frame rate while SNR scalability may be used to deliver video at different quality levels. One of the main design objectives of a scalable video coding structure is to maintain the rate-distortion performance at each layer (to minimise the overhead of enabling scalable decoding) at a comparable level to non scalable coding of the same decoded information. A typical scalable video coding example is video streaming, where the scalable coding of a sequence can be used to deliver only one or few layers depending on the available bandwidth

support lower bit rates than MPEG-1 and MPEG-2. MPEG-4 Visual supports object based video coding where a video scene is divided into different video objects that can be coded independently of each other; for example, foreground and background objects can be coded differently to each other.

H.261 [26]

This standard (approved in 1993) is aimed at, and was widely used for, videophone and video conferencing applications over sub-primary ISDN channels (p x 64 kbps where p = 1...30). The H.261 standard utilises hybrid coding algorithm (similar to section 2.3 and in MPEG-1) for efficient coding at lower bit rates using relatively a computationally simple algorithm. The H.261 standard only supports QCIF and CIF (optional) resolution non-interlaced video.

H.263 [15]

This standard was originally aimed at low bit rate video communications. The core algorithm is based on the H.261 standard. However, it supports a broad range of video formats and advanced coding tools such as half pixel precision motion compensation (the motion vectors pointing to the prediction block have half pixel accuracy) and a variety of negotiable coding tools such as "unrestricted motion vectors", where the motion vector points to a region outside the picture boundary (non existent pixels are filled by the pixels at the picture boundary) and "advanced prediction", where the macroblock (the basic unit of coding, a 16x16 block of luma samples and associated chroma samples) is divided into four blocks and each block is motion compensated using individual motion vectors, resulting in higher degree of compression efficiency and flexibility. The baseline profile of H.263 and the simple profile of MPEG-4 are functionally identical.

H.264 / MPEG-4 Part 10: Advanced Video Coding [2]

The new video coding standard commonly known as H.264/AVC was jointly developed by the ITU-T VCEG and the ISO/IEC MPEG. The H.264/AVC is capable of achieving significantly improved compression efficiency and flexibility compared with all previous video coding standards. The increase in performance is due to the variety of coding tools and options available in the standard which, however, increases the computational complexity significantly.

This research work is aimed at managing the computational complexity of a H.264/AVC encoder and therefore a good understanding of the H.264/AVC standard is required. The next chapter provides an overview of the features and the coding tools available in the H.264/AVC standard.

3 An Overview of H.264/AVC Standard

3.1 Introduction

The H.264/AVC [2] [27] video coding standard is jointly developed by the ITU-T as recommendation H.264 and the ISO/IEC as international standard 14496-10 (MPEG-4 Part 10) Advanced Video Coding (AVC). The H.264/AVC standard is intended to provide significantly better compression efficiency compared to the previous standards. This chapter provides an overview of some of the main features of the standard.

The chapter is organised as follows. The development history of the standard is briefly discussed in Section 3.2. Next, a high level overview of H.264/AVC is provided in Section 3.3. Sections 3.4 and 3.5 discuss the Network Abstraction Layer (NAL, which handles transportation of coded video data) and Video Coding Layer (VCL, which carries out core video compression) respectively. Section 3.6 focuses on some of the specific coding tools available in the video coding layer. Error resilience tools available in H.264/AVC are discussed in section 3.7. Section 3.8 summarises the H.264/AVC profiles followed by a discussion on codec implementation.

3.2 Standard Development

In 1998 a call for proposals was issued by ITU-T Video Coding Experts Group (VCEG) for a new video coding standard with the objective of doubling the compression efficiency compared to any video coding standard available at the time. The new proposal was referred to as H.26L. As a result of similar interest by ISO/IEC, the Joint Video Team (JVT), consisting of ITU-T VCEG and ISO/IEC Moving Picture Experts group (MPEG), was formed in 2001 to make the development of the new standard a combined effort. The standard was finalised and the draft was approved in May 2003 [2].

The H.264/AVC standard was originally developed for "entertainment quality" video where sampling format is limited to 4:2:0 with 8 bit sample accuracy. An amendment was added to the standard in July 2004 called the Fidelity Range Extensions (FRExt) [28] [19] [29] which introduced the so-called "High Profiles" (discussed later in the chapter) in order to address "professional" applications and to enhance the compression performance. The high profiles can support up to 4:4:4 sampling format and 12 bit sample accuracy.

Recently, an 'Advanced 4:4:4 Profile' has been proposed to code 4:4:4 format video [30] [31]. The new proposal includes coding of chroma components in 4:4:4 with luma coding tools and is reported to be out-performing the 'High 4:4:4' profile. At the time of writing, work is being carried out to include scalable video coding support in H.264/AVC.

The H.264/AVC standard was designed for high compression efficiency, error resilience and flexibility so that it could support a wide variety of applications and different transport environments such as wired and wireless networks. The H.264/AVC standard is intended to support a wide range of applications such as:

- Video conferencing and video telephony services over networks such as LAN, DSL, wireless and mobile networks.
- Video on demand and multimedia streaming services.
- Digital broadcasting services.
- Video storage on media
- Multimedia messaging services

A few examples of the widespread industry adoption of the H.264/AVC video coding standard in the above and other relevant application areas (extracted from [32] and [33]) are:

Video Conferencing: Companies like Polycom and Tandberg have included H.264/AVC into their video conferencing products.

Mobile Networks: The 3RD Generation Partnership Project (3GPP) [34], which creates mobile multimedia standards, has incorporated H.264/AVC as the primary video codec in specification release 6.

Video Storage: H.264/AVC (Particularly the 'High' profiles introduced by FRext) has been adopted as a supported codec in both HD-DVD [35] and Blu-Ray [36] disk formats.

Television Broadcasting: Digital Video Broadcasting project (DVB) [37], a consortium for digital television broadcasting standards, has adopted H.264/AVC for HDTV and SDTV services.

Similar to previous standards, H.264 only specifies the syntax structure of the bit stream and the decoding process of the syntax, effectively specifying a decoder as opposed to the encoder. This ensures high flexibility in encoder implementation as long as the generated bit stream conforms to the syntax, while guaranteeing interoperability and correct decoding of content. However, the decoder is also flexible to some extent since the decoder is allowed to decode the syntax in any way as long as the decoding process produces numerically identical results to the process specified in the standard. The flexibility enables the optimisation of the encoding process to suit different applications. For example, in a video storage and reproduction application such as DVD, more emphasis can be given to maximise the video quality, whereas in a video telephony application, more emphasis can be given to complexity and implementation costs.

3.3 H.264/AVC Overview

The H.264 standard consists of various features and coding tools that contribute to the high compression efficiency, flexibility and robustness. This section describes the structure and some of the high level features of the standard.

3.3.1 Layer Structure

H.264 is designed to be flexible and customizable to handle a variety of applications and transport methods. To achieve the flexibility, the standard was designed to contain two layers.

1. The Video Coding Layer (VCL) represents the core video encoding process (which carries out actual video compression) and the VCL data consists of coded bits.

2. The Network Abstraction Layer (NAL) handles the transportation of VCL data and other header information by encapsulating them in NAL units.

The separation of video coding and transportation into two layers ensures that the video coding layer provides an efficient representation of video content, while the network abstraction layer transports the coded data and other header information in a flexible manner by adapting to a variety of delivery frameworks.

3.3.2 Profiles and Levels

Profiles and levels are used to specify the tools and capabilities of the decoder that is needed to support different applications and to provide interoperability points between different decoder implementations. Each profile is designed to have particular coding tools to support various coding requirements. The H.264/AVC standard originally specified the following three profiles (herein after referred to as "basic profiles").

1. **Baseline**: The coding tools are intended for low-latency, low-complexity, error resilience and robustness. Applications: video conferencing.

2. **Main**: For high compression efficiency. Applications: video storage and broadcasting

3. **Extended**: A superset of the baseline profile with enhanced error resilience and video stream switching capabilities. Applications: internet video streaming.

The fidelity range extensions introduced a new set of profiles called the "High" profiles intended for high quality applications (e.g. HD-DVD, HDTV) and professional applications like studio editing. They are namely, High (HP), High 10 (Hi10P), High 4:2:2 (Hi422P) and High 4:4:4 (Hi444P) profiles. A list of features and coding tools supported by each profile can be found in section 3.8.

Levels are defined as performance limits for decoders supporting each profile. Performance limits generally apply to processor load, memory capabilities and the maximum bit rates which in turn affect the frame sizes, frame rates and number of reference frames supported by a compliant decoder.

3.3.3 Picture Format and Structure

The source video is coded as a stream of pictures. The colour spaces and the sampling formats of the pictures and the process of dividing a picture into coding units comprised of slices and macroblocks are discussed in this section.

3.3.3.1 Colour space and sampling

The basic profiles support YCrCb '4:2:0' sampling format with 8-bit sample accuracy whilst the high profiles support '4:2:2' (Hi422P) and '4:4:4' (Hi444P, which also supports RGB colour space) with up to 10 and 12 bit sample accuracies. All the high

profiles also support '4:0:0' monochrome video format. The width and height of the luma sample array of a picture should be a multiple of 16, while the width and height of the chroma sample array is a multiple of 8 or 16 depending on the sampling format, so that the picture includes all the chroma samples associated with the luma samples. Both progressive and interlaced video are supported.

3.3.3.2 Macroblocks

The smallest coding unit in a picture is a Macroblock (MB). A macroblock contains data belonging to a region of 16x16 luma samples along with the associated Cr and Cb component samples. A picture should contain an integral number of macroblocks.

3.3.3.3 Slices

A picture consists of one or more slices. Each slice contains an integral number of macroblocks which should be processed in raster scan order. H.264 has following slice types.

- **I-Slices:** All the macroblocks in the slice are coded using intra prediction. The macroblocks are coded using data already coded within the same slice (Intra).

- **P-Slices:** Contains inter coded macroblocks using one reference picture and/or intra coded macroblocks (Predictive).

- **B-Slices:** Contains inter coded macroblocks using two reference pictures as well as macroblock types in P-slices (Bi-predictive).

- **SP and SI-Slices:** Special types of slices, Switching Predictive (SP) and Switching Intra (SI), for efficient switching between different video streams, random access and error resilience [38].

Figure 3.1 shows how a picture (using 4:2:0 sampling format) is divided into slices and macroblocks. The number of slices and the number of macroblocks in each slice is flexible and therefore, the encoder can decide on an appropriate size depending on the coding requirements. Slices are processed independently of each other, i.e. no information from any other slice is used to decode any one slice of the picture. The independent decoding of slices adds robustness against data loss because loss of data belonging to a slice does not affect the decoding of the rest of the picture.

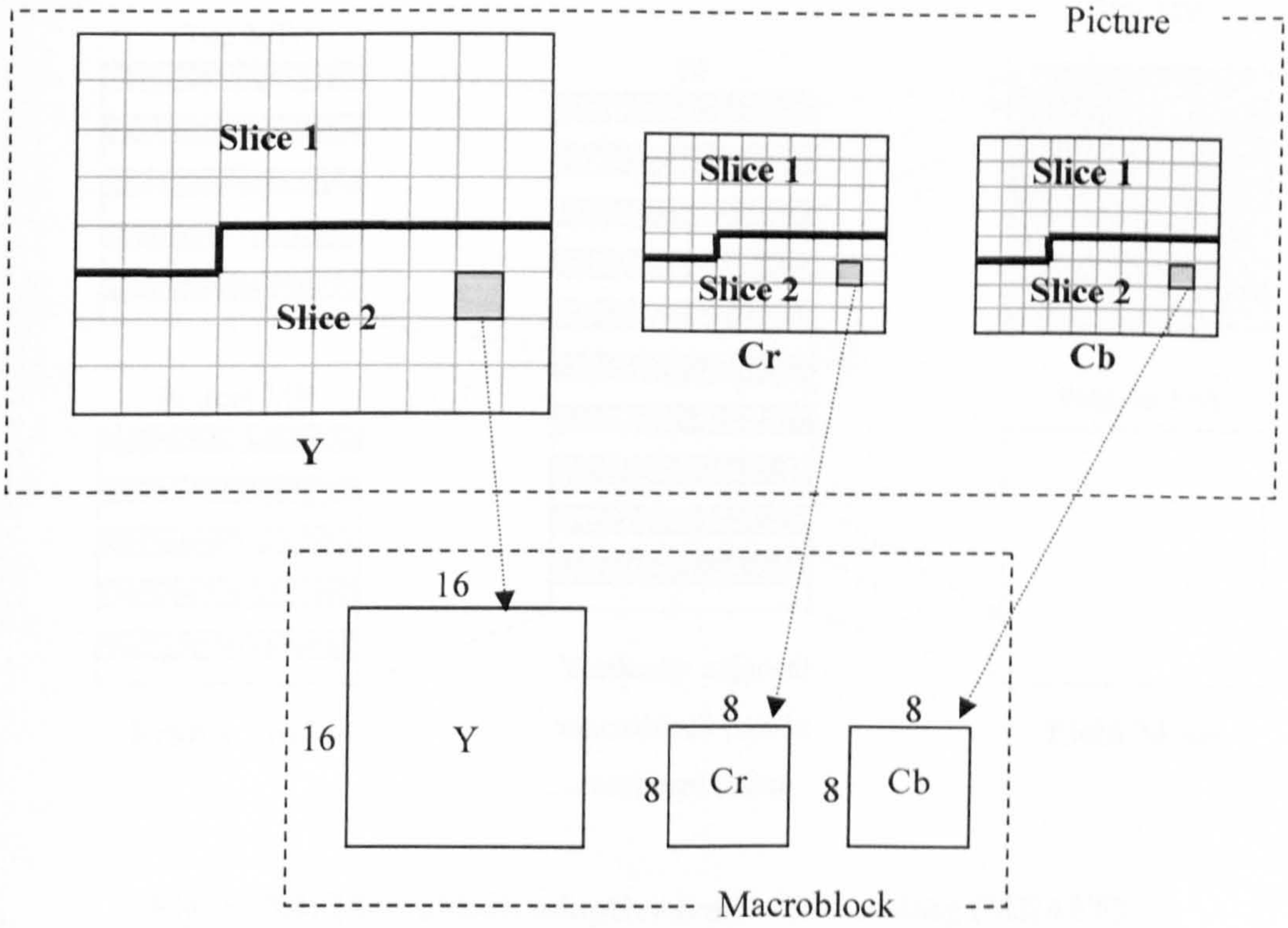


Figure 3.1 Picture structure – Slices and macroblocks (4:2:0 sampling)

3.3.3.4 Interlaced coding

Coded pictures in H.264/AVC consist of frames in progressive video and frames or fields in interlaced video. To maximise coding efficiency, H.264 can adaptively encode each frame of interlaced video as follows:

- Picture Adaptive Frame/Field coding (PicAFF) – The encoder can decide to combine the two fields and encode as one frame (frame picture) or encode as two different fields (two field pictures).

- Macroblock Adaptive Frame/Field coding (MBAFF) – the two fields are combined and encoded as one frame. The frame is split into vertically adjacent macroblock pairs. Each of these macroblock pairs in the picture can be adaptively coded either in frame mode or field mode. In field mode the top macroblock (of each vertical pair) contains the top field and the bottom macroblock contains the bottom field of the region covered by the macroblock pair. In frame mode, the top and bottom macroblocks are encoded as frame macroblocks (see Figure 3.2).

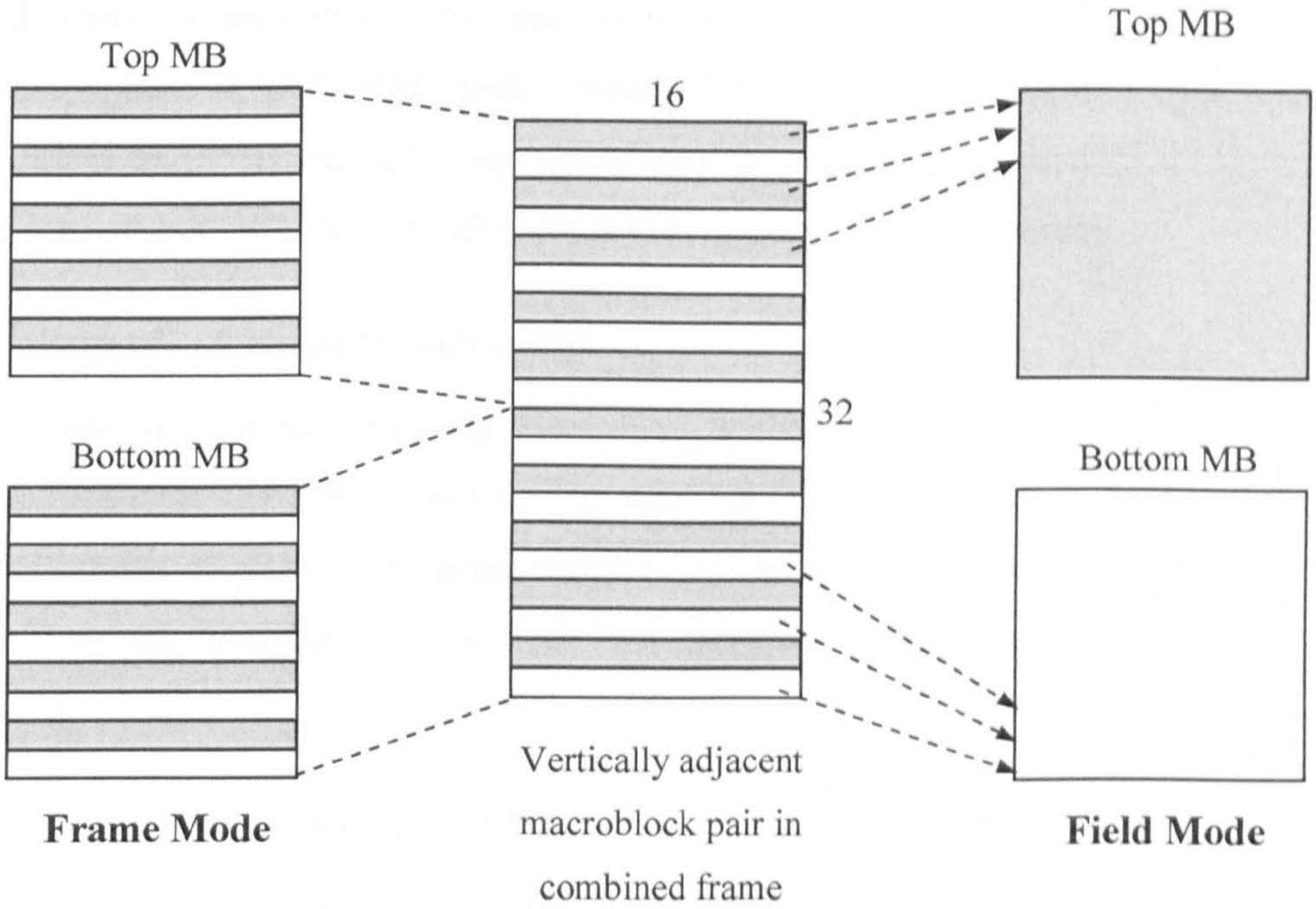


Figure 3.2: Macroblock adaptive frame/field coding (MBAFF)

3.4 Network Abstraction Layer

Coded video data (VCL) and all other information are encapsulated in NAL units before they are transmitted (or stored). Therefore, a coded video sequence contains a sequence of NAL units. In packet based networks, NAL units are transported in packets and sequenced in the correct order at the receiving end. In circuit switched networks, the sequence of NAL units is transmitted as a byte stream where the start of each NAL unit is identified using a start code prefix. The start code prefix is a unique code that is made to occur only at the beginning of a NAL unit.

Figure 3.3 shows the general form of a NAL unit. The NAL unit contains a header to identify the type of the NAL unit and the data (VCL or other information, explained below) is called the Raw Byte Sequence Payload (RBSP).

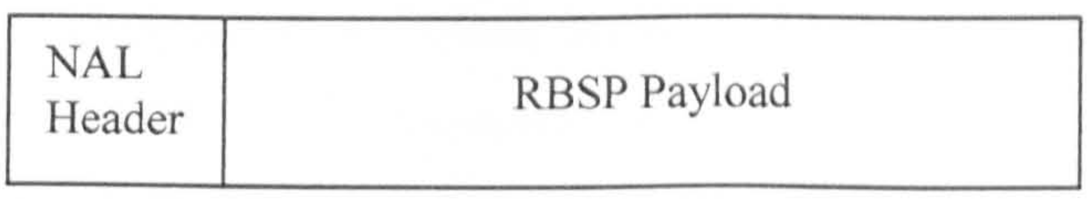


Figure 3.3: NAL unit

The NAL units are divided into two categories as VCL-NAL units and non-VCL-NAL units. The payload of VCL-NAL units contains VCL data, i.e. coded slice data such as coded I-Slices and P-Slices. All other information is contained in non-VCL-NAL units. The contents of non-VCL-NAL units include (among others) the following.

3.4.1.1 Sequence Parameter Sets (SPS)

The parameter sets contain common, important, infrequently changed information. The Sequence Parameter Set (SPS) contains common information required to decode pictures of a whole video sequence. Examples include the profile and the level that should be supported by the decoder, picture size and whether the frames are interlaced or progressive.

3.4.1.2 Picture Parameter Sets (PPS)

These are parameters needed to decode one or more pictures in the video sequence. Examples include entropy coding mode, initial quantisation parameters and whether the default parameters of the de-blocking filter is modified by slice header.

Figure 3.4 shows a possible NAL unit order of a coded video sequence. SPS and PPS should be transmitted prior to coded slices in order to decode pictures (or slices) of the sequence, because decoding of each slice is linked to a particular PPS and SPS for the required common information.

Figure 3.4: An example NAL unit sequence

The organisation of the NAL units permits the use of parameter sets PPS and SPS to represent important, common and infrequently changed data. Therefore robustness can be improved by transmitting these parameter sets using a more reliable channel than the video channel. For example, in a packet based network VCL-NAL units can be transported using RTP/UDP while parameter sets can be delivered using TCP for robustness against the loss of important data.

The standard does not specify how the NAL units should be customized to suit different transport networks. However, the network abstraction layer acts as a customisable middle layer between the transport network and the video coding layer. It enables H.264/AVC to be flexible to accommodate a broad range of transport frameworks.

3.5 Video Coding Layer

The core video compression is carried out by the Video Coding Layer. The Video Coding Layer compresses the pictures by dividing the pictures into one or more slices which contain an integral number of macroblocks. Macroblocks are the basic coding units of the H.264/AVC encoder. The basic architecture of H.264/AVC is similar to previous coding standards such as MPEG-2 and H.263 where a motion compensated block based transform is used to achieve compression.

Functional block diagrams of H.264 encoder and decoder are shown in Figure 3.5 and Figure 3.6 respectively. These figures show the high level functional elements which should be present in an encoder and a decoder which complies with H.264/AVC. The operation of the H.264/AVC encoder and decoder is briefly described here. Specific features and coding tools available in each functional element will be discussed in the next section.

3.5.1 H.264/AVC Encoder

The macroblocks in the current picture are processed as either intra or inter coded macroblocks. The encoder consists of a forward path (represented with thick black arrows in Figure 3.5) for the encoding and a reverse path (grey arrows) for decoding and reconstruction of the current picture. A prediction signal for the macroblock is calculated using either intra prediction or inter prediction. In intra prediction, the current macroblock is predicted from the neighbouring samples in the current slice which have been already encoded, decoded and reconstructed by the encoder. In inter prediction the prediction signal is obtained through motion estimation and compensation using one (or two) reference picture(s) from the reference picture buffer. The reference picture buffer contains previously coded and decoded pictures that can be selected for inter prediction.

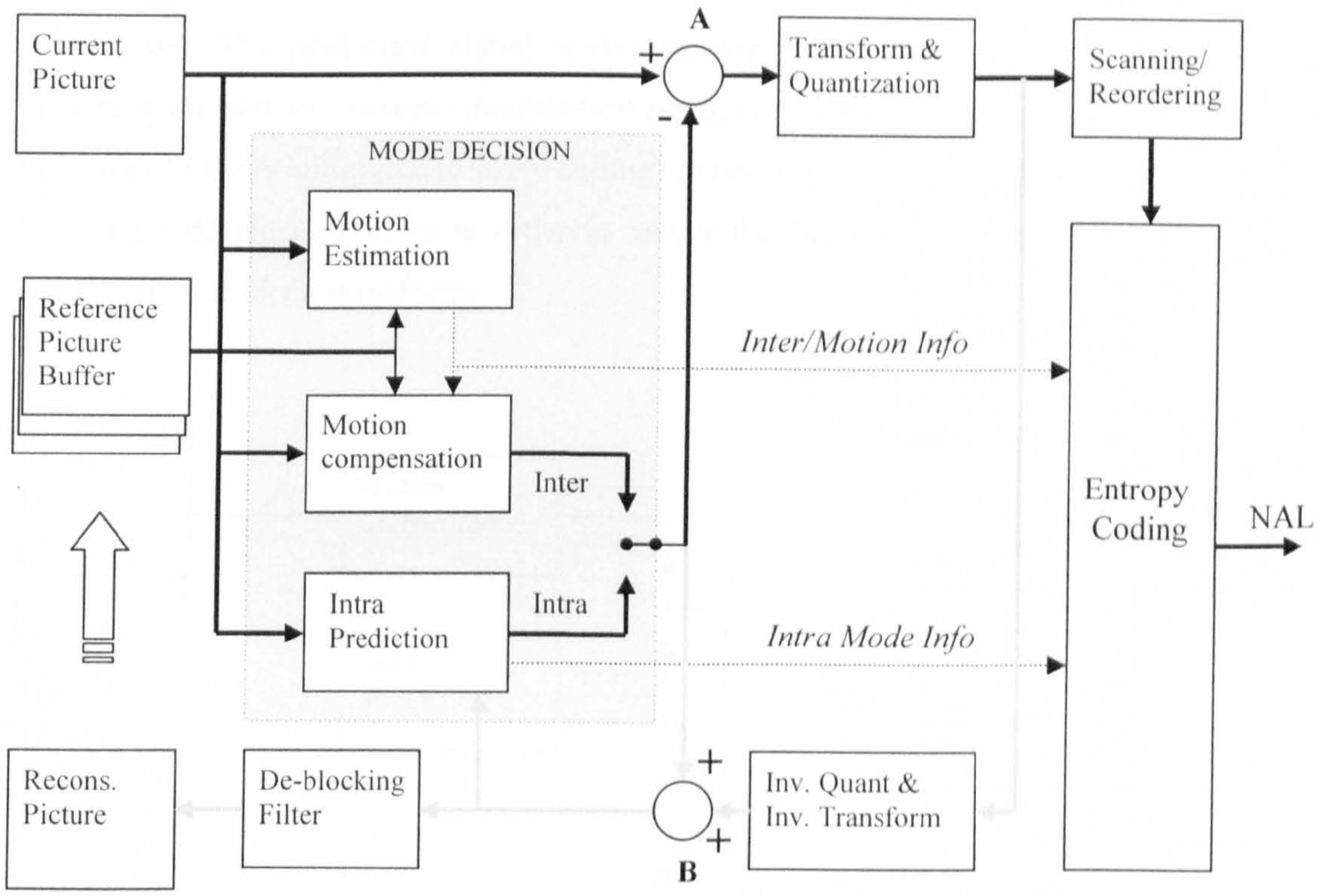


Figure 3.5: Functional block diagram of H.264 encoder

The prediction macroblock is then subtracted from the original macroblock to create a residual macroblock at node A. The residual macroblock is transformed, quantised and reordered before entropy coding. Entropy coding (lossless coding) is done to remove the statistical redundancy of the data. The entropy coder also processes other information (dotted arrows) necessary for correct decoding of the residual data such as the quantisation parameter, macroblock partition modes (see section 3.6.2: inter prediction), the reference frame(s) used, motion vector information for inter coded macroblocks and Intra mode information (see section 3.6.1: intra prediction) for intra coded macroblocks. The output of the entropy coder is compressed video bits which are encapsulated in NAL units before transmission or storage.

The objective of the reverse path (marked by thin arrows) is to reconstruct the lossy coded picture (after quantisation) exactly as the decoder. The reconstructed samples of the neighbouring macroblocks in the current slice may be used for intra prediction of macroblocks and the current reconstructed picture may be used for inter prediction of future pictures. The exact match between the reconstructions at the encoder and the

decoder is essential to avoid any error between the prediction signals at the encoder and the decoder. The prediction signal calculated during the forward path is added to the inverse quantised and inverse transformed residual to create a reconstructed macroblock (different to the original due to lossy coding) at node B. The picture is reconstructed after applying a de-blocking filter in order to reduce the blocking artefacts appearing due to quantisation of block transforms.

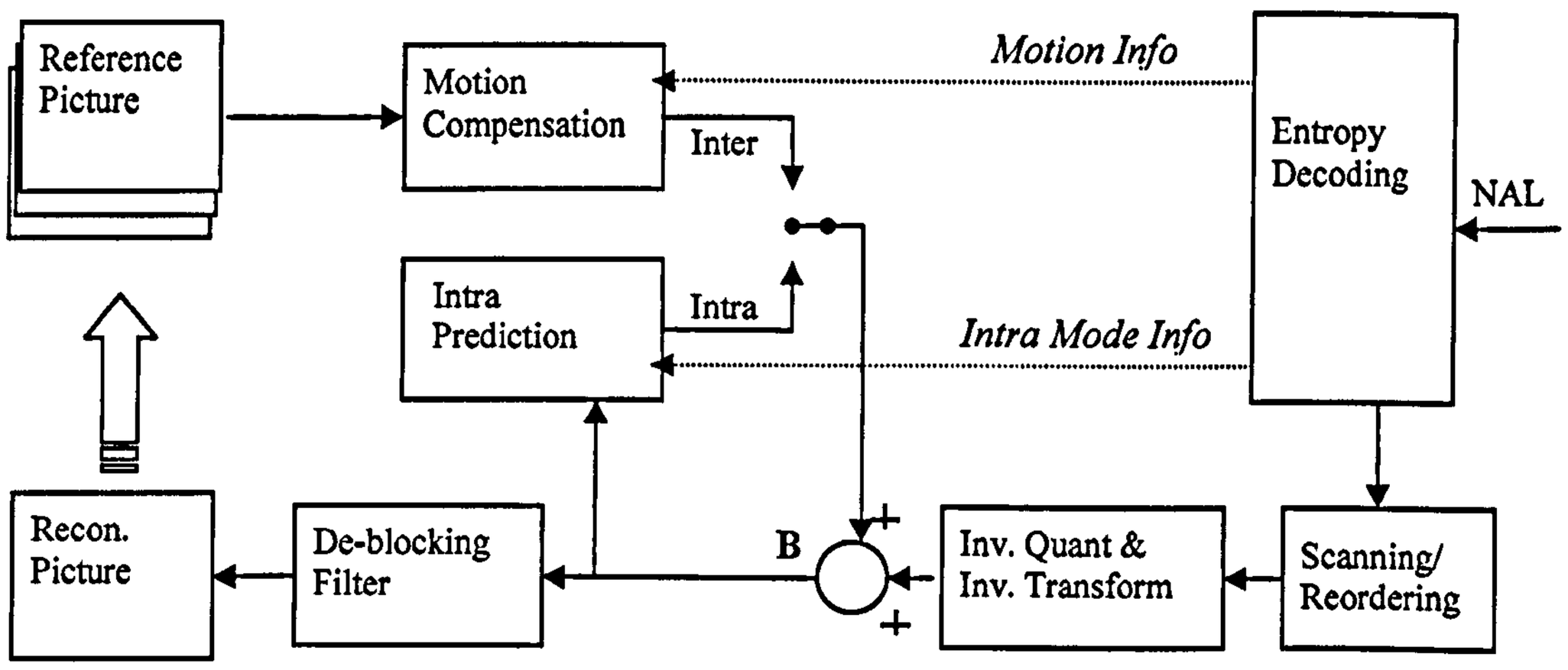


Figure 3.6: Functional block diagram of H.264 decoder

3.5.2 Decoder

The decoder block diagram is shown in Figure 3.6. Starting from the right hand side, NAL units are input to the decoder. The NAL units are first entropy decoded to obtain the quantised coefficients and other information necessary to reconstruct the macroblocks using the quantised coefficients. The coefficients are then rescanned into the luma and chroma arrays of the macroblock. Similar to the reverse path of the encoder, inverse quantisation and inverse transform are applied to the coefficients to produce the residual macroblock. For inter coded macroblocks, a prediction is obtained by carrying out motion compensation using the decoded information such as macroblock partition modes, reference picture(s) and motion vectors. Intra coded macroblocks are predicted using the decoded intra mode information and previously decoded pixels of neighbouring macroblocks. The macroblock is reconstructed by adding the prediction to the residual at node B. The de-blocking filter is applied to reconstruct the current picture. The

reconstructed picture is displayed and may also be used as a reference picture for decoding future pictures.

3.6 Video Coding Tools and Functions

The H.264/AVC standard offers a wide range of coding tools to achieve a high level of compression efficiency. Some of the important coding tools and functions of the H.264/AVC standard will be briefly discussed in this section[2].

3.6.1 Intra Prediction

In intra prediction, the prediction signal is produced using the neighbouring samples of previously encoded and reconstructed blocks which are located on the left of and/or above the current block (see below). Therefore, intra prediction exploits spatial correlation of image pixels. The following intra coding modes are supported in all slice types. Note that intra prediction is not carried out across slice boundaries. Therefore, slices can be decoded independently of each other to limit error propagation.

3.6.1.1 Intra 4x4 prediction for luma samples

Intra prediction is carried out for each individual 4x4 block of the macroblock. The small prediction block sizes are particularly useful for areas which have high detail. There are nine different 4x4 intra prediction modes that can be selected. The pixel values of each 4x4 block are predicted from the neighbouring pixel values. Figure 3.1 illustrates the prediction modes. Prediction directions are indicated by arrows and the shaded squares represent the already decoded neighbouring pixels. For example: in mode 1 (Intra_4x4_Horizontal) the pixels in each row in the 4x4 block are predicted from the pixel on the left. Details of the prediction process can be found in [2].

---

[2] The H.264/AVC standard only specifies the syntax and the decoding process of the syntax and therefore the coding tools and functions are presented in the standard in the point of view of the decoder. However, in this thesis, the coding tools and functions are presented in the point of view of the encoder as in many other literature. This is mainly due to the ease of explanation/understanding and because the work presented in the thesis is primarily concerned with the encoding process.

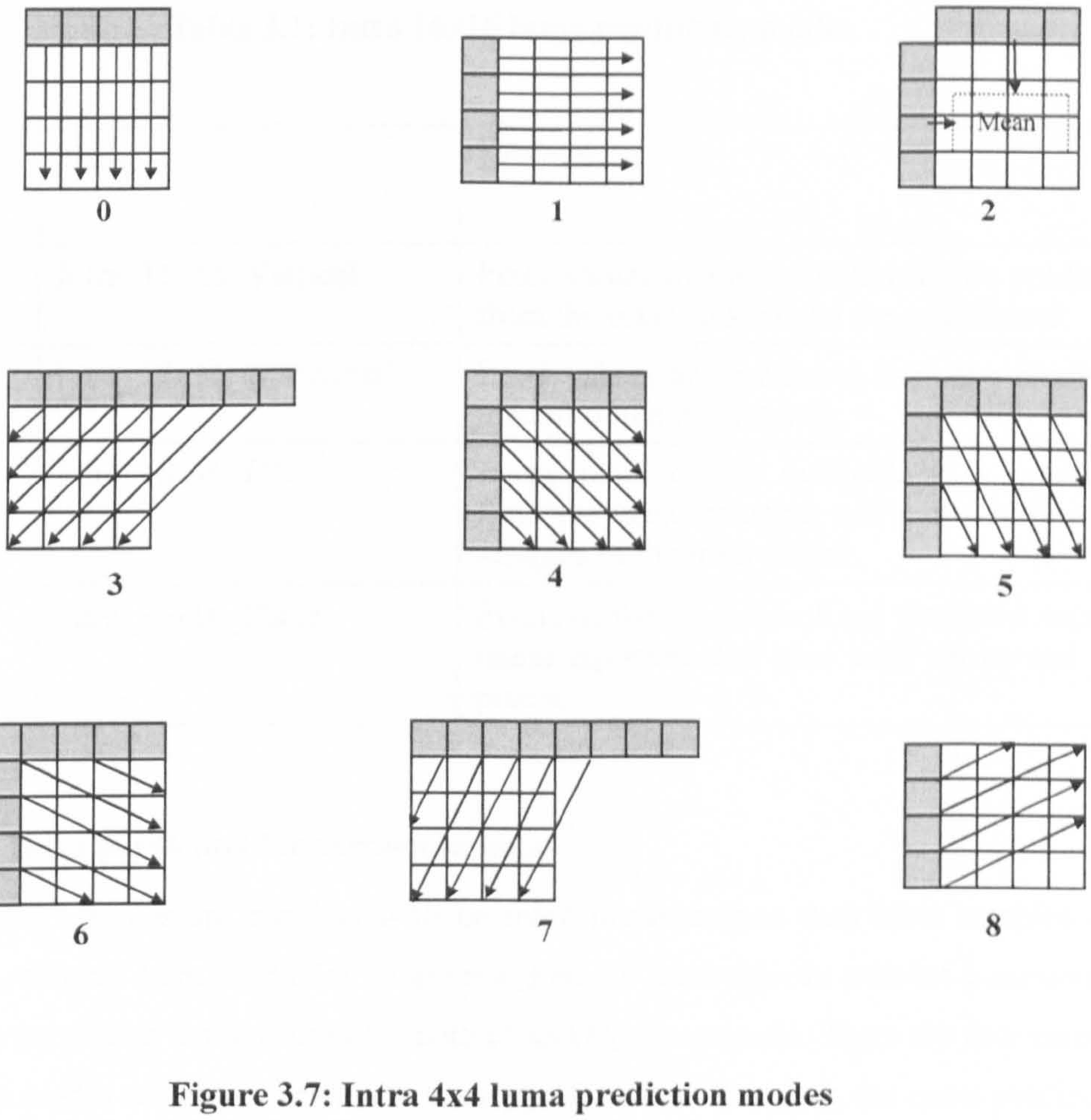


Figure 3.7: Intra 4x4 luma prediction modes

3.6.1.2 Intra 8x8 prediction for luma samples (High profiles only)

In 'High' profiles, the encoder can select an 8x8 block size for intra prediction of luma macroblocks. The intra 8x8 modes are similar to the nine intra 4x4 prediction modes shown earlier. The reference pixels are low-pass filtered in order to improve the prediction (see [19] for details).

3.6.1.3 Intra 16x16 prediction for luma samples

The samples of the macroblock are predicted without partitioning. This is useful for homogeneous areas that do not contain much detail. The four intra 16x16 prediction modes are specified as in Table 3.1. The prediction modes are similar to corresponding modes of intra 4x4; however, a block of 16x16 samples and the corresponding left and above samples are used for the prediction process.

Table 3.1: Intra 16x16 luma prediction modes

Mode	Name	Description
0	Intra_16x16_Vertical	Pixel values of the macroblocks are predicted from the pixels just above the macroblock
1	Intra_16x16_Horizontal	Pixel values are predicted from the pixels to the left of the macroblock.
2	Intra_16x16_DC	Every pixel of the macroblock is predicted from the mean of upper and left neighbouring samples of the macroblock.
3	Intra_16x16_Plane	Pixels of the macroblock are predicted using a linear equation that uses both above and left pixels.

3.6.1.4 Intra prediction for chroma samples

The chroma samples are considered to be more homogeneous than luma samples and therefore, chroma intra prediction is always done on macroblocks without partitioning. The same prediction mode is used for both C_r and C_b components. There are four chroma prediction modes which are similar to intra 16x16 modes. However, the exact prediction process is specified for different chroma formats due to the difference in chroma macroblock size. For '4:2:0' sampling format the chroma macroblock size is 8x8; for '4:2:2' the size is 8x16 (column x row) and the chroma macroblock size is 16x16 for the '4:4:4' format.

3.6.1.5 I_PCM

This is a lossless coding mode where the image sample values are transmitted directly without prediction, transform or quantisation. Although this is a very inefficient method of coding, this method is useful to represent image regions without any loss[3].

---

[3] The Fidelity range extensions introduced a transform-bypass mode (available in Hi444P profile) for lossless video coding. This mode uses prediction and entropy coding without carrying out transform and quantisation to represent the sample data without any loss.

3.6.2 Inter Prediction

Inter prediction is carried out to exploit the temporal redundancy between pictures. Block based motion estimation and compensation is carried out in order to create the inter prediction signal. The inter prediction tools contribute significantly to the improved compression efficiency of the H.264/AVC standard over previous coding standards. Some of these tools are discussed here.

3.6.2.1 Variable block size motion compensation

Motion compensation is carried out for macroblocks by dividing the macroblocks into partitions and sub-macroblock partitions. Figure 3.8 shows how the luma component of a macroblock can be partitioned for motion compensation. Each macroblock can be partitioned into one 16x16 (whole macroblock), two 8x16, two 16x8 or four 8x8 partitions. Further, if four 8x8 partitions are chosen each of them can be motion compensated as a whole 8x8 partition, two 4x8 sub-partitions, two 8x4 sub-partitions or four 4x4 sub-partitions. Each partition (or sub-partition) is individually motion compensated using a separate motion vector.

Figure 3.8: Macroblock and sub-macroblock partitions

The chroma sample arrays contain the chroma samples associated with the luma samples of the macroblock and therefore the size of chroma partitions varies depending on the sampling format. For example, in '4:2:0' format, the chroma sample arrays contain half of the number of luma samples in vertical and horizontal directions. Therefore, chroma partitions and sub-partitions are quarter the size of corresponding luma macroblock partitions and sub-partitions.

These variable block sizes enable the encoder to maximise the compression efficiency by choosing the appropriate partitions to create a closely matching (to the original macroblock) motion compensated prediction. However, for each macroblock, there are 259 partitioning combinations to choose from. Therefore, the encoder needs to evaluate all the possible combinations to achieve the maximum compression efficiency at a high computational cost. This issue is introduced in section 3.9 and further discussed in chapters 4 and 7.

3.6.2.2 Quarter pixel accurate motion vectors

Motion estimation and compensation is carried out by generating a prediction signal for each macroblock or sub-macroblock partition from the reference picture. Motion vectors indicate the relative position of the matching area in the reference picture. In H.264/AVC, motion vectors have luma quarter pixel accuracy. Therefore, the reference picture is interpolated to represent sub sample and quarter sample pixel positions (see Figure 3.9). For luma sample interpolation, a 6-tap filtering process is carried out to obtain half pixel positions from the neighbouring samples. The quarter sample positions are obtained by linear interpolation between half pixel and / or full pixel positions. Luma quarter pixel accuracy for the motion vectors means chroma pixels may need to be interpolated up to $1/8^{th}$ pixel positions in horizontal and/or vertical directions depending on the sampling format. These pixel positions are obtained by linear interpolation between four neighbouring samples.

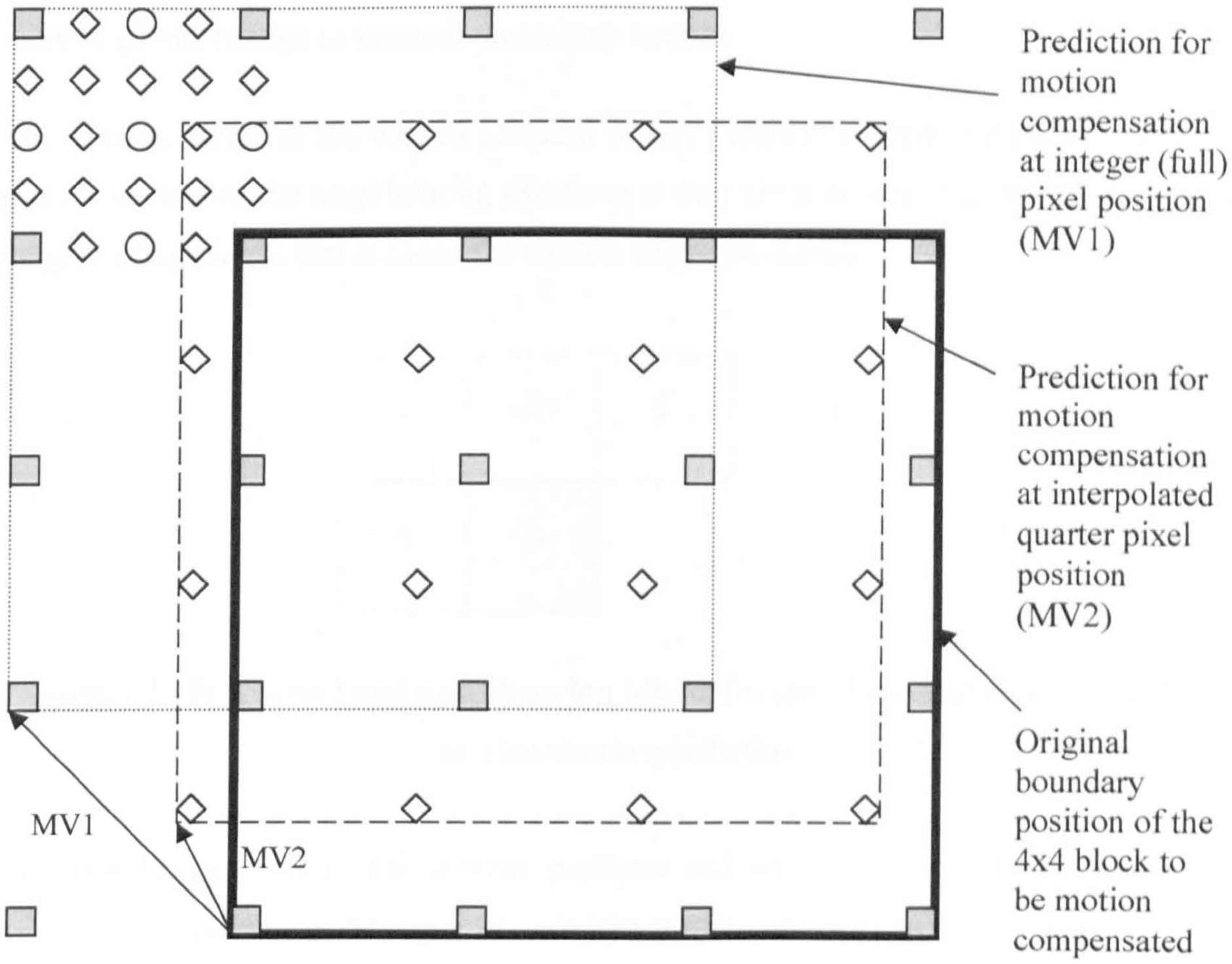


Figure 3.9: Quarter pixel accurate motion compensation

3.6.2.3 Motion vectors over picture boundaries

H.264/AVC allows motion vectors to point to regions outside the picture boundary. The pixels of the outside region are obtained by extrapolating the pixel values at the picture boundary. This allows for effective motion compensation of objects moving in or out of the picture boundary.

3.6.2.4 Motion vector prediction

Encoding of motion vectors may result in large number of bits, in particular because there can be a number of motion vectors corresponding to a number of small (sub) partitions

used for motion estimation. Therefore, motion vector prediction is used to reduce the number of bits needed to transmit the motion vectors.

The motion vector of the current partition or sub partition is predicted (MVP) from the motion vectors of the neighbouring partitions if they are available. Figure 3.10 shows the neighbouring blocks that are used for motion vector prediction.

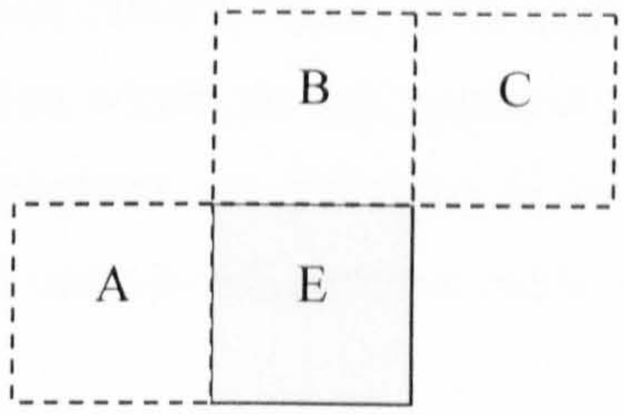


Figure 3.10: Current and neighbouring blocks (macroblock partitions) used for motion vector prediction

The shaded block E is the current partition and the blocks A, B and C are the neighbouring partitions. As a general rule, the median of the motion vectors of A, B and C is used as the motion vector prediction of current block E. However, specific rules apply depending on the availability of the neighbouring partitions, partition types and the reference pictures used.

Only the Motion Vector Difference (MVD), which is the difference between the actual motion vector for the current partition and the Motion Vector Prediction (MVP), is transmitted. The number of bits needed for the motion vectors can be reduced due to high correlation between the motion vectors of neighbouring blocks.

3.6.2.5 Reference pictures

A Decoded Picture Buffer (DPB), consisting of previously encoded and decoded pictures, is maintained by the encoder and the decoder. Pictures from the DPB are used as reference pictures for inter prediction. This enables the encoder to choose from a number of candidate reference pictures in order to obtain the best prediction.

Two reference picture lists, 'list0' and 'list1', are maintained by the encoder and decoder. However, baseline profile uses only I-Slices and P-Slices and therefore only the 'list0' is maintained (see below). Each list contains pictures in the DPB which may be past or

future pictures in the display order relative to the current picture. The position of a picture in each list is called the reference picture index. Typically, the index 0 of 'list0' contains the closest past picture in display order and index 0 of 'list1' contains the closest future picture in the display order relative to the current picture position. The reference pictures in 'list0' are used for inter prediction in P-Slices whilst both 'list0' and 'list1' can be used for inter prediction of B-Slices. The reference pictures can be selected individually for each macroblock partition (16x16, 16x8, 8x16 and 8x8). However, if 8x8 partitions are chosen, each sub-partition within an 8x8 partition must use the same reference picture. The chosen reference pictures are indicated to the decoder by sending the reference picture index along with macroblock partition mode and motion vector information.

3.6.2.6 P and B slices

Macroblocks or macroblock partitions in P-slices are inter-predicted using one reference prediction with a reference picture selected from the reference picture 'list0'. Macroblock partitions or sub-partitions in B-Slices can have one or two motion vectors. Macroblock partitions can be predicted from a reference picture in 'list0' or a reference picture from 'list1' where only one motion vector and reference index is used. Macroblock partitions can also be bi-predicted from two reference pictures, one from 'list0' and one from 'list1' and therefore two motion vectors and reference indices are used. The average (when weighted prediction is not used) pixel values of the two reference predictions are used as the bi-prediction signal. If weighted prediction is used, bi-prediction pixel values are obtained as the weighted average of the two reference predictions. The weighting factors can be either encoder defined (explicit) or implicit, where reference predictions are weighted according to the temporal distance of the reference pictures from the current picture. (Note: explicit weighted prediction can also be used in P and SP slices). There is also a special mode called the direct mode (only for the 16x16 partition size) where no motion vectors or reference picture indices are sent. They are derived from the macroblocks that have already been decoded.

3.6.3 Transform and Quantisation

A residual macroblock is generated by subtracting the prediction from the original macroblock (at node A of the encoder block diagram in Figure 3.5). The residual macroblock is transformed to remove the spatial correlation. The 'Baseline', 'Main' and 'Extended' profiles only use an 4x4 integer transform [6] [39] which is based on the DCT,

to transform the residual data of the macroblock by dividing the macroblock into 4x4 blocks. In 'High' profiles, an 8x8 integer transform [29] can also be adaptively used for the inter macroblock partitions that are equal to or larger than 8x8. For intra macroblocks in high profiles, the 8x8 transform is used if and only if the macroblocks are predicted using an intra 8x8 mode. The integer transforms can be carried out using integer arithmetic and are less complex than the DCT [29, 40]. Since no floating point arithmetic is used, the possible mismatch between the forward and reverse transform is eliminated. For macroblocks predicted using intra 16x16 mode, the DC coefficients of the 4x4 transform blocks are further transformed using a 4x4 Hadamard transform [41]. The DC coefficients of each chroma component (four 4x4 transform coefficient blocks for each 8x8 chroma component) are further transformed using a 2x2 Hadamard transform.

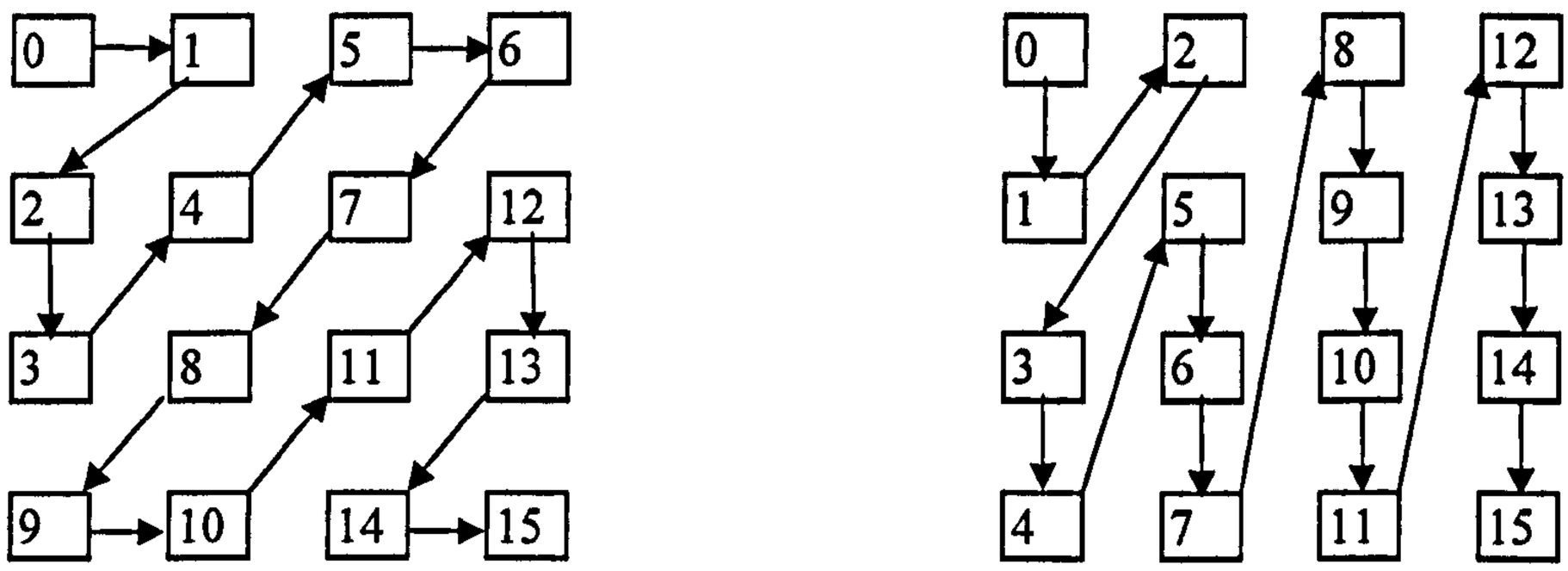
Lossy compression is achieved by quantising the transformed residual data. The Quantisation Parameter (QP) specifies the quantisation step size used. The QP can be varied from -(Offset) to 51, where, Offset = 6*(Bit depth – 8). The "Bit depth" is the sample accuracy. Therefore, for 8-bit samples the QP ranges from 0 to 51 and the range increases by 6 for every additional bit in bit depth. The value of the quantisation step size doubles for every 6 increments of the QP, allowing a wide range of bit rates and image qualities. Typically the luma quantisation value is specified and the chroma quantisation value is derived from the luma quantisation value. Each macroblock can be encoded using different quantisation parameter values. The QP is differentially coded and therefore only the change in QP is transmitted to the decoder. When using high profiles, the encoder can choose different QP values to encode Cr and Cb components independently and "Perceptual-based Quantisation Scaling Matrices" can be used where the quantisation scaling factors for frequency components (or transform coefficients) can be modified by the encoder in order to improve the subjective quality of the picture.

3.6.4 Reordering/Scanning

The transformed and quantised coefficients of the macroblock are zigzag scanned in 4x4 blocks if the transform size is 4x4 and are scanned in 8x8 blocks if the transform size is 8x8 ('High' profiles). Figure 3.11 shows the scanning process for a 4x4 block. The scanning order varies depending on whether the macroblock is a frame macroblock or a field macroblock. The coefficients of the macroblock are scanned in 4x4 blocks (4x4 transform) as shown, to maximise the number of consecutive zeros after quantisation. A

similar zigzag scanning order is specified for the 8x8 residual blocks when using the 8x8 transform.

The luminance 4x4 blocks are scanned as shown in Figure 3.11. If the macroblock is coded using an intra 16x16 mode, the DC coefficients of 4x4 luma blocks are scanned separately in the same pattern. The rest of the coefficients in the luma blocks are scanned as above, starting from the first AC coefficient. The DC coefficients of the chroma components are scanned in raster scan order and the AC coefficients are scanned in the zigzag order.



Frame Macroblocks **Field Macroblocks**

Figure 3.11: Scanning order of 4x4 coefficients in a macroblock

3.6.5 Entropy Coding

Entropy coding uses lossless coding techniques to reduce the size of coded data. Examples of coded data include data such as sequence/picture parameter sets, slice headers, slice data and macroblock layer data such as residual coefficients, motion vectors and macroblock modes. Data is entropy coded differently depending on the type of data.

The slice headers and higher layer syntax elements are encoded as fixed length codes or Universal Variable Length Codes (UVLC). Slice data and other macroblock layer syntax are encoded using either Context Adaptive Variable Length Coding (CAVLC) or Context Adaptive Binary Arithmetic Coding (CABAC) depending on the chosen entropy coding mode.

CAVLC – Variable length codes are assigned to data elements depending on the probability of occurrence. Shorter codes are assigned data elements that occur more frequently and longer codes are assigned to less frequent ones. Context adaptive means that different code tables are used according to local sequence statistics, so that more efficient coding can be achieved. Only the residual coefficients are coded context adaptively.

CABAC [42]– Arithmetic coding can generally achieve higher compression efficiency than variable length coding, but, at a higher computational complexity. Typically CABAC involves the following:

1. Binarisation – Syntax elements having non-binary values are converted to a string of binary values, which are uniquely decodable code words. Each position in the binary string is called a 'bin'.

2. Context Modelling - Each bin is assigned a probability model (probability of the bin being 0 or 1) depending on the context, i.e. the probability estimate of each bin depends on the context such as the type of the syntax element and the values of previous syntax elements. These context models are updated by feeding back the actual bin values after encoding each bin.

3. Binary Arithmetic Coding – Arithmetic coding is carried out by selecting the sub-range using the probability estimates of the current bin and the bin value.

Context modelling in CABAC is not only limited to coding residual coefficients, but also other syntax elements such as motion vector differences and macroblock modes, whereas in CAVLC, only the residual coefficients are coded context adaptively. CABAC reportedly achieves 9% - 14% higher compression efficiency compared to CAVLC [42].

3.6.6 Skipped Macroblocks

H.264/AVC specifies a special type of macroblocks called skipped macroblocks. For skipped macroblocks, no coded information is sent to the decoder. A syntax element in slice data indicates the skipped macroblocks to the decoder. Skipped macroblocks in P-slices are called P-Skip and in B-Slices they are called B-Skip.

When a particular macroblock is coded as P-Skip (for inter predicted macroblocks in P-Slices), the following conditions are assumed by the decoder:

1. The chosen macroblock partition size is 16x16 (i.e., a single motion vector for the macroblock).

2. The X and Y components of the MVD are zero (i.e. motion vector is the predicted motion vector). The motion vector prediction is calculated slightly differently to the median prediction normally used for the 16x16 partition size. For example, the prediction vector is set to zero if either the left of the top macroblock is not available.

3. The chosen prediction reference is the previous frame in display order (list0 reference index is 0).

4. All coefficients are zero after transform and quantisation.

If a macroblock in B-Slices are coded as B-Skip, the decoder assumes that the macroblock is coded as a direct mode macroblock having zero quantised transform coefficients.

The decoder does not receive any motion information or residual data for the skipped macroblock. Since the motion vector differences are zero, the motion vector prediction (or direct mode motion vectors for B-Skip) becomes the actual motion vector(s) used to obtain the predicted macroblock. Therefore, the prediction macroblock is simply copied as the reconstructed macroblock.

Typically, skipped macroblocks occur in regions with low (or similar movement as neighbouring macroblocks) movements and/or detail and therefore, the predicted macroblock (having MVD = (0, 0)) is very similar to the original macroblock. The residual data of this type of macroblocks is low resulting in zero coefficients after transform and quantisation.

When the entropy coding mode used is CAVLC, the skipped macroblocks are signalled to the decoder using the parameter "mb_skip_run" to specify the number of macroblocks that are consecutively skipped since the last coded (not skipped) macroblock. When

CABAC is used, a flag "mb_skip_flag" is used to indicate whether a macroblock is skipped or not.

3.6.7 De-blocking Filter

The quantisation of block transform coefficients can lead to "blockiness" in the reconstructed picture. The H.264/AVC standard specifies an in-loop de-blocking filter to minimise the blocking artefacts. The de-blocking filter is applied in-loop, meaning that the reconstructed and filtered pictures are used as reference pictures for inter prediction. The same filter parameters are used at both the encoder and the decoder to avoid any prediction errors. Typically a filtered picture provides a closer match to the original picture than the unfiltered reconstruction. Therefore a better prediction can be obtained using the filtered reference picture, resulting in higher objective and subjective quality (a detailed explanation of the de-blocking filter can be found in [43]). The filter is applied over 4x4 block boundaries in macroblocks and the filter strength depends on the quantisation parameters, prediction modes of neighbouring blocks and the actual pixel values across the boundary. Also, the filter strength can be explicitly changed or the filter can be completely turned off by the encoder.

3.7 Error Resilience Tools

Improved error resilience is an important design characteristic of the H.264/AVC standard [4]. Some of the design features which incorporate error resilience were discussed in the earlier sections. The transmission of commonly used data as parameter sets in NAL units can be used to improve the error resilience by using a more reliable transport mechanism. The visible errors due to packet losses can be minimised using multiple slices per frame and error propagation is minimised due to independent decoding of the slices. Multiple reference frame selection (with feedback from the decoder) is useful to avoid using lost pictures for inter prediction. Intra coded macroblocks (spatially predicted using only intra macroblocks) can be used to eliminate temporal error propagation. There are also some specific tools available for improved error resilience and robustness.

3.7.1 Arbitrary Slice Order (ASO)

With ASO, the slices of a picture can be sent or received and decoded in any order relative to each other. For example, robust decoding can be achieved in a packet based

network consisting of alternative paths, in a situation where, the bottom slice of the picture is received before the top slice (two slices per picture). The decoder can decode the bottom part of the picture without waiting for the top part of the picture.

3.7.2 Flexible Macroblock Ordering (FMO) / Slice Groups

Macroblocks can be grouped into slice groups where each slice group can contain one or more slices. Within each slice of the slice group, macroblocks should be coded in raster scan order. Therefore, a slice group can contain macroblocks from different parts of the picture enabling flexible ordering of macroblocks. Combined with error concealment, FMO can effectively be used as an error resilience feature. For example, consider two slice groups in checker board pattern throughout the picture. If one slice group is lost, error concealment can be carried out using the decoded macroblocks surrounding the missing macroblocks.

3.7.3 Redundant Slices (RS)

H.264/AVC allows the use of redundant slices (for whole or part of the picture) which can normally be discarded. However, redundant slices can be encoded with a different reference picture for prediction than the original. The decoder can use the redundant slice for decoding in cases where the reference picture used for prediction in the original slice is lost.

3.7.4 Data Partitioning (DP)

Slice data can be divided into different data partitions by separating header (more important) and residual data of slices and macroblocks. Data partitions can be transmitted using separate NAL units so that if a residual data partition is lost, it may still be possible to decode the slice to some extent.

3.8 A summary of H.264/AVC Profiles

The design of the H.264/AVC standard is targeted at a wide rage of applications from video conferencing to HDTV and professional studio editing applications. Therefore, as mentioned earlier in this chapter, the standard defines a set of "profiles" that include subsets of available coding tools and features targeted at different application scenarios. The following tables show the features, coding tools and possible target application areas of each profile. Table 3.2 indicates the features and coding tools contained in the baseline,

main and extended profiles. All the coding tools and features contained in the main profile are included in high profiles. The additional features and tools available in the high profiles are shown in Table 3.3.

3.9 Codec Design and Complexity

The profiles and levels specified in the H.264/AVC standard provide specific interoperability points between codec implementations. Therefore, codec designers can choose to implement only a specific profile at a specific level depending on the application requirements such as real-time (or non-real-time), bit rate, frame-size, frame rate and available processing power. The encoder designer need not be concerned about issues such as the selection of coding tools to be implemented because they are already specified for the chosen profile. Any H.264/AVC compliant decoder supporting a particular profile and level is able to decode a bit stream coded for that profile and level.

Table 3.2: A summary of 'Baseline', 'Main' and 'Extended' profiles

Features/Tools		Baseline profile	Main profile	Extended profile
Slice structure	I and P	√	√	√
	B		√	√
	SI and SP			√
Entropy coding	CAVLC	√	√	√
	CABAC		√	
Interlaced coding	PicAFF, MBAFF		√	√
Error resilience	FMO, ASO, RS	√		√
	DP			√
Prediction	Weighted prediction		√	√
Main considerations		Low complexity, high robustness and error resilience	High compression efficiency	Compression efficiency, enhanced error resilience and stream switching
Possible applications		Video telephony & conferencing, video over wireless networks	Digital TV broadcasting, digital media storage	Video streaming

Table 3.3: A summary of the 'High' profiles

Features/Tools	High (HP)	High 10 (Hi10P)	High 4:2:2 (Hi422P)	High 4:4:4 (Hi444P)
Intra 8x8	√	√	√	√
8x8 transform	√	√	√	√
Separate QP control for Cr and Cb	√	√	√	√
Perceptual-based quantisation scaling matrices	√	√	√	√
Bit depth - 9, 10		√	√	√
Bit depth - 11, 12				√
Sampling format 4:0:0 (monochrome)	√	√	√	√
4:2:2			√	√
4:4:4				√
RGB				√
Transform-bypass mode (lossless coding)				√
Main considerations	More compression efficiency than the main profile	Intended for applications which require higher sampling formats and higher sample accuracy		
Possible applications	High end consumer use and high resolution applications like HDTV / HD-DVD	Professional video editing		

Once the profile and the level are chosen, typically the main objective of the encoder design is to maximise the video quality subject to the bit rate constraint (rate-distortion optimisation). There is maximum flexibility at the encoder to use the coding tools and options in the best possible way to achieve this. Therefore, the encoders must have a

mechanism (a decision making process) to evaluate and choose from different coding options (for example, whether a macroblock should be inter coded or intra coded, inter/intra mode, quantisation parameter to be used) to suit the coding requirement. However, H.264/AVC encoders are implemented on hardware or software (or hybrid) platforms which have finite processing resources. Therefore the decision making process should consider the computational complexity of evaluating each coding mode or parameters due to the limitation in processing power. Encoder implementations seek to strike a balance between compression performance and implementation cost/complexity by employing mode/parameter selection methods that tries to optimise rate-distortion performance at manageable complexity levels.

Note that the decoder implementation is largely constrained by the syntax and the decoding process specification (however, the decoders are allowed to process the syntax in any way as long as the results are identical to the results obtained using the specified process). Therefore, decoder design does not pose any significant algorithmic issues compared to the encoder. However, the decoders can benefit from reduced computational complexity by using techniques such as multi threading for multi-processor platforms (see appendix B) or using SIMD instructions [44].

3.10 Conclusion

The new H.264/AVC video coding standard can deliver significantly improved compression efficiency compared with previous standards, supporting higher quality video over lower bit rate channels. Due to improved compression efficiency and increased flexibility of coding and transmission, H.264 has the potential to enable new video services such as mobile video phones and multimedia streaming over mobile networks. The H.264/AVC standard supports a wide range of applications from consumer applications like video conferencing to professional applications like studio editing.

The H.264/AVC standard has a range of coding tools contributing to its high compression performance, flexibility and robustness. However, the performance improvements come at a cost of significantly high computational complexity. Therefore, encoder implementations should make use of the available coding tools effectively to achieve the desired compression performance with the available processing resources.

The next chapter on rate-distortion optimised video coding, describes a mechanism used to make encoding decisions to maximise rate constrained video quality.

4 Rate-distortion Optimised Video Coding

4.1 Introduction

The H.264/AVC video coding standard specifies the coding syntax and the decoding process of the syntax, enabling a flexible encoder implementation. This has also been the case with earlier coding standards such as H.263. The flexibility is necessary to enable encoders to achieve high compression efficiency for different and changing source statistics. H.264/AVC achieves this flexibility by allowing for many different coding modes and coding parameter selections for encoding video. Therefore, it is the responsibility of the encoder to find the best way to encode a particular video sequence by choosing from a range of coding modes and parameters.

Typically, video encoders compress video signals using lossy coding techniques. Generally, the desired compression ratio is achieved through a quantisation process, resulting in some amount of distortion in the decoded video compared to the original. The goal of the encoder is to achieve optimum rate-distortion performance by selecting the best coding modes and parameters for a particular video sequence. In other words, the encoder strives to achieve minimum distortion at any particular target bit rate for a particular video sequence.

Rate-distortion Optimisation (RDO) methods used in video compression are discussed in [45], which include dynamic programming and Lagrange optimisation methods. Dynamic programming methods typically involve many calculations (or in a video coding scenario, possibly many encodings of each coding unit) that form a decision tree consisting of all the possible solutions and following the decision tree that provides the optimum solution. This type of decision making (although optimal) demands a significant amount of computational resources. On the other hand, Lagrange optimisation methods, which are also known as Lagrange multiplier methods, offer computationally less complex (although sometimes sub-optimal) solutions to the optimisation problem. Due to its less complex nature, a specific form of the Lagrange optimisation method has been used in rate-distortion optimisation of H.264/AVC and H.263 reference software encoders [46] [47]. The computational complexity involved is an important factor in choosing Lagrangian rate-distortion optimisation methods as the basis of the new low complexity

algorithms developed during the project. Therefore, this chapter is focused on Lagrange optimisation methods as means of rate-distortion optimisation.

The next section (4.2) introduces the constrained optimisation problem. Section 4.3 deals with the theoretical aspects of the Lagrange optimisation method and its application to typical video coding problems. The optimal and sub-optimal solutions in relation to video coding and the computational complexity issues relating to them are also discussed. Implementation of Lagrangian rate-distortion optimisation in H.263 and H.264 encoders is discussed in section 4.4 along with advantages, disadvantages and computational complexity related issues.

4.2 Constrained Optimisation Problem

The constrained optimisation problem deals with maximising (or in this case minimising) an objective function within a resource constraint. In video coding terms, the constrained optimisation problem can be presented as: minimising the distortion of a coding unit subject to a maximum number of bits that can be used to encode that particular coding unit. In mathematical terms, the constrained optimisation problem can be presented as follows:

Let S represent all the allowable vectors and let B an element of S, ($B \in S$). The objective function is defined for all B in S as $D(B)$ and the constraint function $R(B)$ is defined for all B in S. The constrained problem can be presented as:

Given a constraint R_c, find

$$\min_{B \in S} D(B) \tag{4-1a}$$

subject to

$$R(B) \leq R_c \tag{4-1b}$$

The solution (B*) to the problem satisfies that $R(B^*) \leq R_c$ and $D(B^*) \leq D(B)$ for all B in S^*, where $S^* = \{B | R(B) \leq R_c\}$.

That is, if the solution to the problem is B^*, then there is no other B in S which satisfies the constraint R_c, that will result in a smaller value for the objective function than $D(B^*)$. The Lagrange multiplier theory offers a way of solving the above constrained problem (i.e. finding B^*) by representing the problem as an unconstrained problem.

4.3 Lagrange Optimisation Theory

A good understanding of the Lagrange optimisation theory (widely referred to as Lagrange multiplier method) is necessary to use it as powerful optimisation tool in video coding. A generalized Lagrange multiplier method was first introduced by Everett in [48] for solving problems of optimal resource allocation. The Lagrange multiplier method for optimal bit allocation in a coding scheme was proposed by Shoham and Gersho in [49]. Their work was based on the theory introduced by Everett applied to the video coding problem. The main theory and the proof presented by Everett, was repeated in [49] . It is also summarised here as follows.

4.3.1 Main Theorem

The constrained optimisation problem was presented earlier in section 4.2, equation (4-1). The Lagrange theory represents the constrained problem as an unconstrained problem as follows:

***Theorem*:** for any $\lambda \geq 0$, the solution $B^*(\lambda)$ to the unconstrained problem

$$\min_{B \in S} \{D(B) + \lambda \cdot R(B)\} \tag{4-2}$$

is also the solution to the constrained problem (4-1) with the constraint $R_c = R(B^*(\lambda))$.

Proof of the theorem

If $B^*(\lambda)$ is the solution to the unconstrained problem (4-2), then:

$$D(B^*) + \lambda \cdot R(B^*) \leq D(B) + \lambda \cdot R(B) \qquad \text{For all } B \in S \tag{4-3}$$

Therefore,

$$D(B^*) - D(B) \leq \lambda\big(R(B) - R(B^*)\big) \tag{4-4}$$

If this is true for all B in S, it is true for a subset of B in S where,

$$R(B) \leq R(B^*) \quad \text{(i.e. } R(B) - R(B^*) \leq 0\text{)} \tag{4-5}$$

Now, for the above subset and for any $\lambda \geq 0$:

$$D(B^*) - D(B) \leq 0 \tag{4-6}$$

$$D(B^*) \leq D(B) \tag{4-7}$$

Therefore with the constraint $R_c = R(B^*(\lambda))$, the solution B^* for the unconstrained problem (4-2) is also the solution for the constrained problem (4-1).

It should be noted that the theory does not guarantee a solution for the constrained problem. It only states that for any $\lambda \geq 0$ of the unconstrained problem, there is a corresponding constrained problem which has the same solution as the unconstrained problem.

Shoham and Gersho used this theory for a bit allocation algorithm to optimally allocate bits using a set of quantisers for any general coding scheme. The following sections look at the application of the main theory in a block based encoder.

4.3.2 Rate-distortion Optimisation of a Coding Unit

To gain a better understanding of the theory, consider the application of this theory for a single coding unit. In the context of a block based encoder such as H.264/AVC, the coding unit is chosen as a single macroblock. It is somewhat unreasonable to consider a single macroblock for rate-distortion optimisation because the resulting rate and distortion of a particular macroblock typically depend on previously encoded macroblocks. Similarly, the encoding parameters of a particular macroblock generally have an influence on the outcome of the encoding of future macroblocks. However, given the current coding state (previous macroblocks already coded) and ignoring any future macroblocks, the constrained problem is defined as: minimise the distortion of the current macroblock subject to a maximum number of coded bits.

The following describes the optimisation of a single macroblock in order to gain a clear understanding of the main concepts. The rate-distortion optimisation over multiple

macroblocks (which is the main objective of the exercise) will be discussed in the succeeding section.

4.3.2.1 Optimisation problem

Consider a macroblock, for which the encoder can encode the macroblock using only one of the 'K' possible modes given by the set $m = \{m_1, m_2, \ldots, m_K\}$. Let '$M$' ($M \in m$) be the mode selected to code the macroblock. In the context of H.264/AVC, these mode allocations could be any allowable combination of macroblock partition modes, Quantisation Parameters (QP), choice of reference frames etc... so that the K possible modes will include all the possible admissible parameter combinations for the macroblock.

Define the objective function $D(M)$ and constraint function $R(M)$, where $D(M)$ and $R(M)$ are distortion and rate of the macroblock as a result of selecting a particular coding mode. If the rate constraint is R_c , the constraint problem is defined as:

Find the coding mode M^*,

$$M^* = \arg\min_{M} D(M) \tag{4-8a}$$

Subject to

$$R(M) \leq R_c \tag{4-8b}$$

This may be written as an unconstrained problem using a Lagrange multiplier:

$$M^* = \arg\min_{M}\left(D(M) + \lambda \cdot R(M)\right) \tag{4-9}$$

Where the solution to (4-9), M^*, would satisfy,

$$R(M^*) = R_c \tag{4-10}$$

The optimum coding mode M^* (if one exists) can be found by solving (4-9). That means, when the macroblock is coded in mode M^* it would satisfy the target rate R_c ($R(M^*) = R_c$). All the other modes (if they exist) that satisfy $R(M) \leq R_c$ will have a

higher distortion than $D(M^*)$. It should also be noted that, there may not be a mode that satisfies the exact rate constraint. In that case, the mode with the rate that is less than and closest to the rate constraint should be chosen.

The coding mode that satisfies the solution for the constrained problem is found by solving the unconstrained problem represented by equation (4-9). The term $D(M)+\lambda \cdot R(M)$ in equation (4-9) is called the Lagrangian rate-distortion cost. The mode that minimises the Lagrangian rate-distortion cost for a particular $\lambda \geq 0$ (which satisfies the rate constraint in the constraint problem) is selected as the solution mode for the constrained problem. This gives rise to the next problem which is, selecting the correct lambda that results in the minimum cost whilst satisfying the rate constraint. To understand the problem further, a graphical representation of the Lagrange multiplier method can be used.

4.3.2.2 Graphical representation

In practical encoders with a finite number of encoding options, the points on the minimum RD (rate-distortion) bound for a particular coding unit are discrete. This is because the finite number of coding modes results in a finite number of rate-distortion points for the coding unit. The minimum RD bound consists of the RD points which are closest to the (0, 0) point. Figure 4.1 shows a graphical representation (also refer to [45] for further explanation) of the Lagrangian multiplier method for the discrete case.

Evaluating $\arg\min_{M}\left(D(M)+\lambda \cdot R(M)\right)$ at a particular λ is equivalent to minimising the plane wave [4] with a slope of $-\lambda$. Therefore, for different increasing lambda values $(\lambda = 0, \lambda_1 < \lambda_2 < \lambda_3)$, the minimum Lagrangian cost values are at points A, B, C and D.

---

[4] A straight line with a constant slope (see the dotted lines in Figure 4.1) which has the general equation $J = D + \lambda \cdot R$. Where, D and R represent the distortion axis and the rate axis. The parameter J represents the Lagrangian cost and the value is determined by the point that the straight line crosses the distortion axis. The rate-distortion point which has the minimum cost for a particular $\lambda \geq 0$ can be found by moving the line whilst keeping the constant slope until it finds the point that has the lowest J value.

Note that:

1. In this example, there is no point in the convex hull (the points in the minimum RD bound which can be connected to form a convex, see Figure 4.1) that has a rate equivalent to R_c.

2. There is a range of lambda values that results in the same solution; e. g. slope of $\lambda_2 \pm \delta\lambda$ would have the same solution point C.

3. The solution that satisfies the rate constraint R_c would be point D since this is the closest point to the rate constraint in the convex hull with minimum distortion.

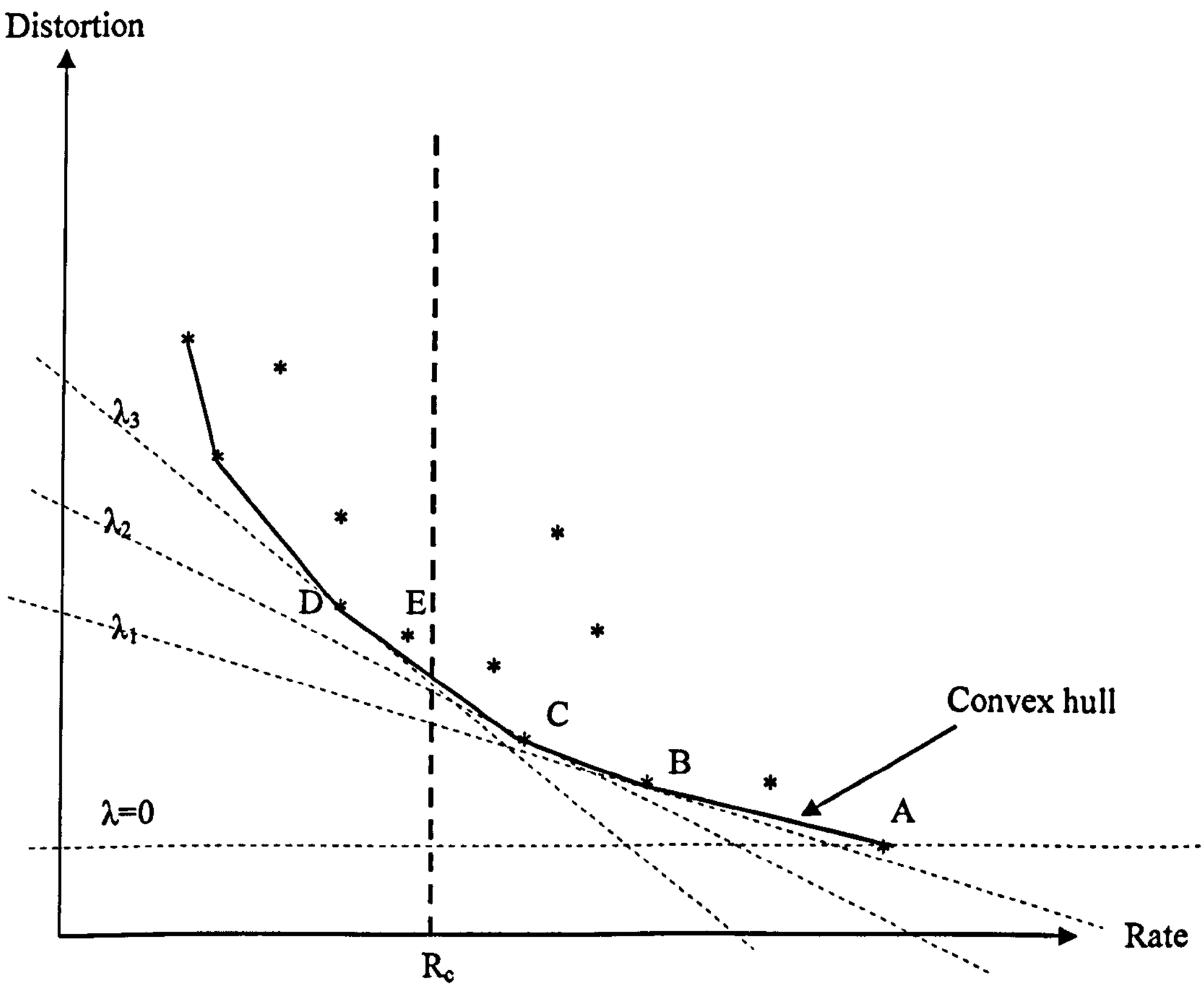


Figure 4.1: Graphical interpretation of Lagrange multiplier method

However, point E is much closer to the target rate with a lesser distortion compared with point D. Therefore, one would like to encode the macroblock using the mode corresponding to point E rather than point D. However, a limitation of using Lagrange

multiplier method for the discrete case is that only the points in the convex hull can be obtained. Typically, practical encoders will always produce discrete rate-distortion points due to the finite number of encoding options.

The above graphical explanation reveals that this method finds the optimum solutions which exist on the convex hull of all the rate-distortion points. The value of the Lagrange multiplier controls the specific trade off between rate and distortion; in effect it determines the relative weighting between rate and distortion.

4.3.2.3 Finding the solution numerically

Solving equation (4-9) by choosing correct lambda involves very high amount of computations. Whilst there are algorithms [49] to find the correct λ without incurring a very large computational overhead, the following briefly explains the "brute force" method of solving the problem.

Assuming a sufficiently large number of possible modes (approximating to a continuous RD function), it should be possible to select an appropriate λ so that an unconstrained solution (a mode that gives minimum cost with the selected λ) is available at a rate equivalent to the rate constraint.

The flow diagram in Figure 4.2 shows the "brute force" method of finding the λ value that satisfies the rate constraint and the corresponding solution.

The method shown in the flow diagram only serves as a guide to understanding the optimisation process of a single coding unit. It is quite straightforward to obtain the correct mode using the method described, provided that the points on the convex hull are virtually continuous and $\delta\lambda$ is very small, i.e. this search strategy will only find an optimum solution if it lies on the convex hull of rate-distortion boundary. It is not the objective of this chapter to describe such methods and therefore, readers can refer to [49], where the authors have described efficient algorithms to obtain optimal or near optimal points.

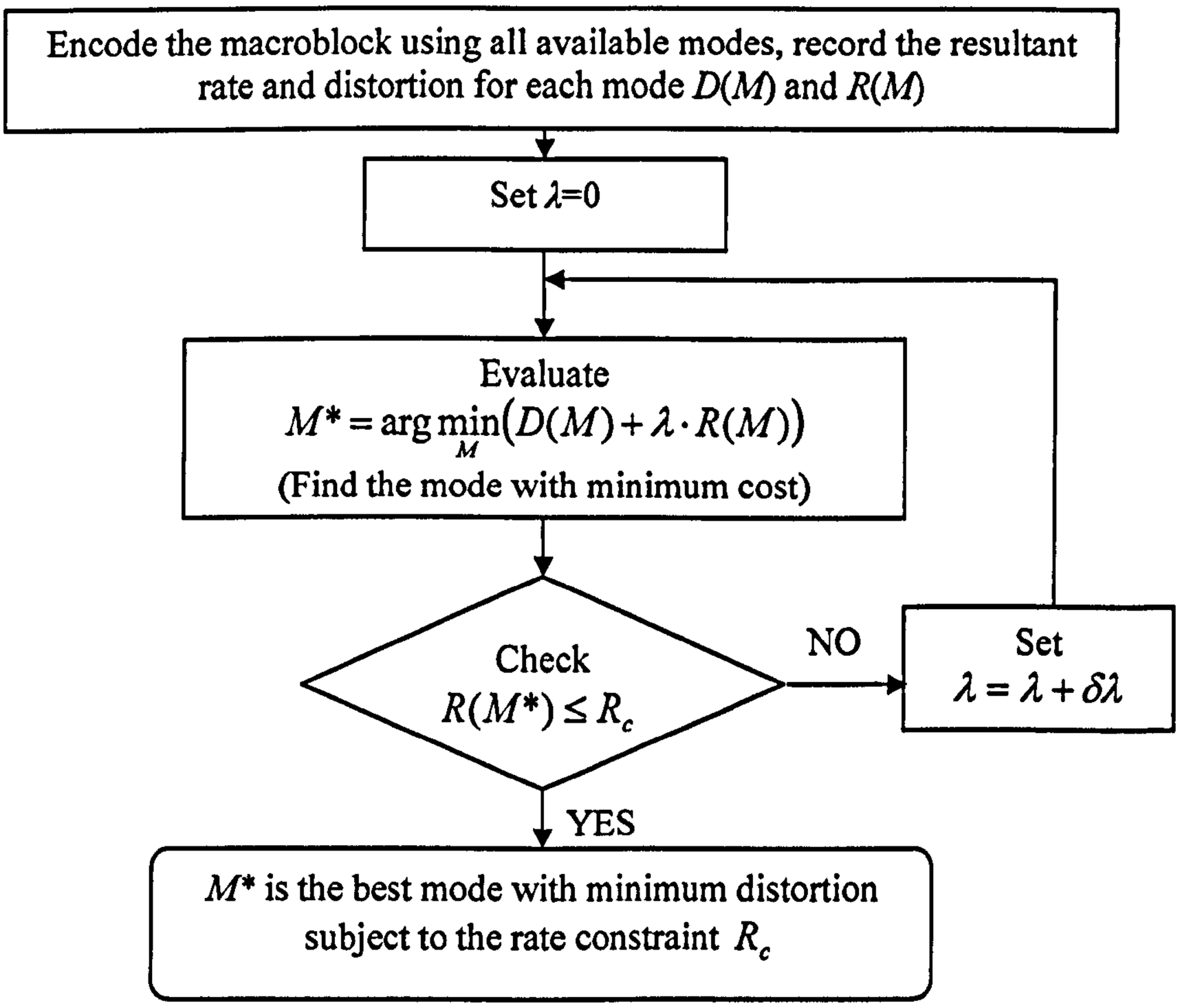


Figure 4.2: Flow chart for finding the optimum coding mode that lies on the convex hull.

4.3.3 Rate-distortion Optimisation of Multiple Coding Units

In practical encoding scenarios there is a need to optimise overall coding performance. This involves minimising the overall (or total) distortion subject to a bit rate constraint over a number of coding units-for example: minimising the total distortion of all the macroblocks in a frame subject to a bit budget in a CBR (constant bit rate) application.

4.3.3.1 Optimisation problem

Consider the rate-distortion optimisation problem over a group of N macroblocks. Let the group of macroblocks be represented by the set $\mathbf{X} = \{X_1, X_2, \ldots, X_N\}$ where, each macroblock (X_i) can be encoded using only one of the K possible modes given by the set $m = \{m_1, m_2, \ldots, m_K\}$. Let M_i ($M_i \in m$), represent the mode selected to code the macroblock X_i. The vector of macroblock coding mode allocations for the group of N macroblocks is defined as $\mathbf{M} = (M_1, M_2, \ldots, M_N)$.

If the objective function is $D(\mathbf{X},\mathbf{M})$ (the overall distortion due to the mode allocation $\mathbf{M}$ over the group of macroblocks $\mathbf{X}$) and the constraint on the total number of bits for the group of macroblocks is R_c, the constrained optimisation problem is:

$$\mathbf{M}^* = \arg\min_{\mathbf{M}} D(\mathbf{X},\mathbf{M}) \tag{4-11a}$$

Subject to

$$R(\mathbf{X},\mathbf{M}) \le R_c \tag{4-11b}$$

where $\mathbf{M}^*$ is a vector of optimal mode allocations, $D(\mathbf{X}, \mathbf{M})$ is the distortion (e.g. average distortion of the group of macroblocks) calculated for the group of coded and decoded macroblocks and $R(\mathbf{X}, \mathbf{M})$ is the coded rate (total number of bits) of the group of macroblocks. This may be written as an unconstrained problem using a Lagrange multiplier method as:

$$\mathbf{M}^* = \arg\min_{\mathbf{M}} \sum_{i=1}^{N} J(X_i) \tag{4-12}$$

Where $J(X_i)$ is the rate-distortion cost for macroblock i, and is given as

$$J(X_i) = D(X_i,\mathbf{M}) + \lambda \cdot R(X_i,\mathbf{M}) \tag{4-13}$$

To obtain the solution for the above unconstrained problem, the following two issues need to be addressed:

1. Choosing the correct Lagrange multiplier λ so that the rate constraint holds for the optimum mode allocation vector $\mathbf{M}^*$.

2. Evaluating (encoding and decoding) all the mode combinations available for the group of N macroblocks.

Assume that the Lagrange multiplier λ can be chosen, considering all N macroblocks, such that the rate constraint holds, using the exhaustive method shown in Figure 4.2 or search methods described in [49]. The task of encoding and decoding the group of

macroblocks using all the available mode combinations is computationally intensive due to macroblock dependency.

4.3.3.2 Macroblock dependency

In standard based encoders like H.264/AVC, every macroblock is predicted from the neighbouring (in space or time) macroblocks. This creates interdependency between macroblocks. For example, for inter coded macroblocks, the mode (QP, partition mode, etc...) selected for the reference macroblock and hence the quality of the reference macroblock will have an effect on the prediction signal and the mode selection of the current macroblock. For intra coded macroblocks, the current macroblock is predicted from the reconstructed samples of the neighbouring macroblocks. Therefore the prediction quality depends on the quality of the neighbouring macroblocks. On the other hand, the quantisation parameter used for the current macroblock will influence the choice of QP for the next macroblock because the QP value is differentially coded, thus influencing the bit rate.

4.3.3.3 Optimal solution

The optimal solution can be obtained by taking the interdependencies into account. This is a computationally intensive task. To explain, consider the following example:

Example 4.1

Consider a hypothetical encoder, where only 3 modes (m_1, m_2, m_3) are available to encode a macroblock and the encoding order of the macroblocks is constrained by the decoding syntax. Assume that the rate-distortion performance needs to be optimised over a group of 2 macroblocks. To find the optimum solution, every combination of modes between these two macroblocks needs to be examined as shown in Figure 4.3.

The possible mode allocations **M** are shown using a tree structure in Figure 4.3. The branches represent all the possible mode allocations for the two macroblocks. A "coding state" is defined as a group of macroblocks encoded in a particular combination of encoding modes, i.e. a particular mode allocation. Therefore, J_{CSi} represents the total Lagrangian cost of a particular mode allocation corresponding to the coding state 'i' (e.g. for i = 1, $\mathbf{M} = (m_1,m_1)$ and for i = 2, $\mathbf{M} = (m_1,m_2)$ etc...). Assuming, that the correct λ is

chosen, the optimum mode allocation is given by the branch of the tree that gives the minimum coding state cost.

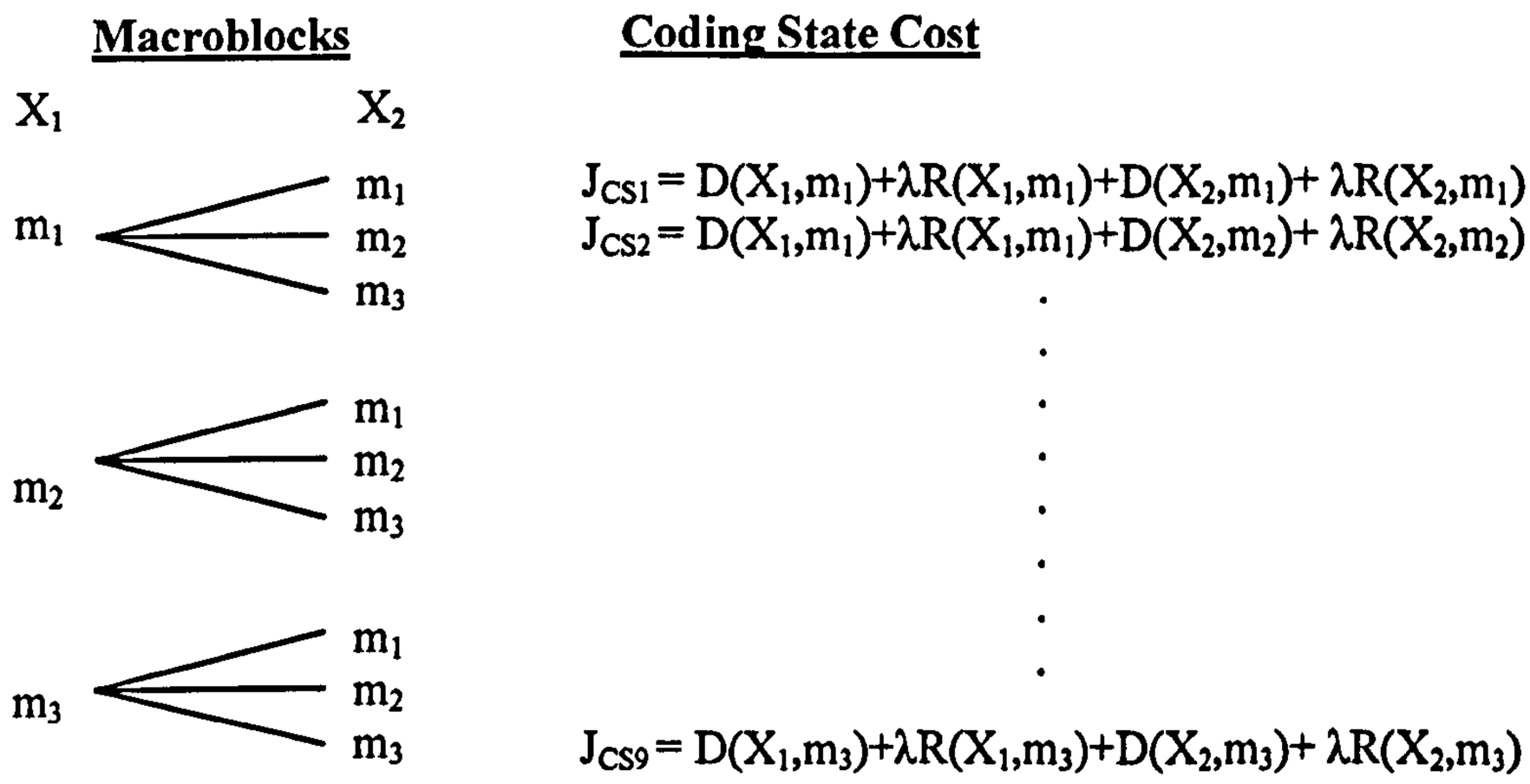


Figure 4.3: Encoding combinations for example 4.1

4.3.3.4 Computational complexity of the optimal solution

To quantify the computational complexity of obtaining the optimal solution, the cost of evaluating every possible mode combination needs to be ascertained.

Consider the number of times a macroblock is encoded, as an estimator of the computational complexity. Assume a function "ENCODE_MACROBLOCK", to be called each time a macroblock is encoded using any mode. Let the number of Macroblock Encodings (MBE) be the number of times the function is called during the optimisation process over a group of macroblocks. Other calculations such as searching for the correct lambda and cost estimation are ignored since they can be considered to be relatively insignificant compared to the complexity of encoding a macroblock. However, the computational complexity involving the search for the correct lambda will be increased when the coding options and/or macroblocks are increased. A large memory is assumed to be available to save each intermediate coding state at each node of the tree to avoid multiple coding of the same intermediate state. Note that this may not be a reasonable assumption when a large number of macroblocks (or modes) are considered.

Therefore, for the example 4.1 in Figure 4.3,

$$MBE = 3 + 3^2 = 12 \tag{4-14}$$

The average MBE per macroblock is 6 (12 / No of macroblocks). This means that, to find the optimum solution, each macroblock should be encoded 6 times on average.

Consider 3 available modes and a group of 10 macroblocks. Then,

$$MBE = \sum_{i=1}^{10} 3^i = \frac{3\left(3^{10} - 1\right)}{3 - 1} = 88572 \tag{4-15}$$

This means on average, each macroblock should be encoded 8857 times (88572 / 10) to obtain the final solution.

In general, if the number of available modes = K and number of macroblocks in the group = N,

$$MBE = \sum_{i=1}^{N} K^i = \frac{K\left(K^N - 1\right)}{K - 1} \tag{4-16}$$

Therefore, it is evident that even for a small number of available modes, evaluating the optimum mode allocations for a group of 10 macroblocks results in a very large amount of calculations.

Example 4.2

The H.264/AVC standard offers a high degree of flexibility in encoding macroblocks. There is a wide choice of encoding parameters to choose from, such as intra coding modes, inter partition modes, quantisation parameters and reference frames. For simplicity, the encoding process is constrained to a limited number of encoding parameters in view of limiting computational complexity as follows:

No of QP values	:	3 (QP-1, QP, QP+1)
No of reference frames	:	1
Prediction modes- Intra	:	13
Inter	:	259

Consider all other parameters unchanged. Note that the optimisation of the motion estimation process has been ignored.

Number of MB modes = Number of QP values*Prediction modes*Number of reference frames

= 3*(13+259)

= 816.

Now consider a group of 2 macroblocks:

$$MBE = \sum_{i=1}^{2} 816^i = \frac{816\left(816^2 - 1\right)}{816 - 1} = 666672 \qquad (4\text{-}17)$$

At this point it should be clear that obtaining an optimal solution for multiple coding units considering the interdependencies is a prohibitively complicated process. The problem is compounded when there are many encoding parameters to be tuned such as in H.264/AVC. Another issue is that the number of cost estimations (not encodings) required to be carried out for each candidate λ in the search algorithm increase in the order of K^N. The relative benefits of increased rate-distortion performance compared with the prohibitive amount of computations involved should also be considered.

4.3.3.5 Sub-optimal solution (by ignoring interdependencies)

Assuming macroblock independence, the unconstrained problem in equation (4-12) can be re-written as:

$$\mathbf{M}^* = \sum_{i=1}^{N} \arg\min_{M_i} J(X_i) \qquad (4\text{-}18)$$

Where now:

$$J(X_i) = D(X_i, M_i) + \lambda R(X_i, M_i) \qquad (4\text{-}19)$$

This means that the optimum mode of each macroblock can be found by minimising the cost function for each macroblock individually. However, the rate constraint is set for the entire group of macroblocks, in contrast to setting a rate constraint for a single

macroblock where lambda has to be chosen for each macroblock individually (according to individual rate constraints). In this case, the same λ value (assuming λ is known in advance) corresponding to the overall rate constraint should be used for all the macroblocks in the group.

4.3.3.6 Computational complexity of the sub-optimal solution

The computational complexity of the sub-optimal solution can be quantified using the same approach as before. Since interdependencies are ignored, the cost of each individual macroblock can be evaluated separately for all available coding modes. Therefore the number of encodings required for a macroblock is equal to the number of available coding modes (K). If the number of macroblocks in the group is N, then,

$$MBE = N \cdot K \tag{4-20}$$

For the earlier example 4.1 (3 modes, 2 macroblocks), the sub-optimal solution require $MBE = 3 * 2 = 6$ encodings, whereas the optimal solution requires 12. The significance of the difference in complexity is more evident when the number of macroblocks or modes increases (see Table 4.1).

Table 4.1: Computational complexity comparison between optimal and sub-optimal solutions

		Optimal solution		Sub-optimal solution	
No. of Modes	No. of macroblocks in a group	Average encodings per macroblock	Total encodings (MBE)	Encodings per macroblock	Total encodings (MBE)
3	2	6	12	3	6
3	10	8857	88572	3	30
816 (H.264)	2	333336	666672	816	1632

Note that, for the sub-optimal solution, computational complexity increases more than shown in Table 4.1 when a search for the correct Lagrange multiplier is carried out to achieve the rate constraint. This is because the search involves iteration of the encoding process until the rate constraint is met.

Due to the high complexity of the optimal solution (particularly the high complexity of evaluating a large number of modes, as in example 4.2 for H.264/AVC), practical optimisation methods have been developed around the sub-optimal solution. The Lagrange multiplier method has been used in optimisation of video encoders such as H.263 and H.264 [50] [51] [52] by ignoring the interdependencies between macroblocks, resulting in sub-optimal performance with reduced computational complexity. How close these methods approach optimality depends on underlying assumptions and the constraints imposed on the parameters to reduce the complexity of the process. These methods are discussed in the next section.

4.4 Lagrange Optimisation Methods for H.263 and H.264

Successful application of rate-distortion optimisation algorithms to practical encoders should result in achieving the highest possible performance through that particular encoding scheme. However, most encoders utilize some form of prediction from other already coded and decoded units to encode the current coding unit. H.263 and H.264 encoders use prediction from previously encoded pictures creating temporal dependency as well as spatial dependency. Therefore, a truly rate-distortion optimised encoder needs to consider dependencies between all the coding units (or macroblocks in case of H.263/H.264) in the sequence. It is evident from the previous section that this is virtually an impossible task. Therefore, practical implementations for H.263 and H.264 tend to ignore these dependencies.

4.4.1 Rate-distortion Optimisation of an H.263 Encoder

This section explores two optimisation techniques proposed for H.263. The first algorithm was proposed as a rate-distortion optimised rate control algorithm for encoding low bit rate video. The objective is to minimise distortion at very low bit rates. The second is a more widely used and less complicated algorithm used in the H.263 reference software encoder and subsequently adapted for the H.264/AVC reference software encoder.

4.4.1.1 Rate-distortion optimised mode selection and rate control for low bitrate video coding in H.263

This algorithm was the first proposal by Wiegand et al in [53] as a rate-distortion optimised mode selection algorithm. The aim of the algorithm is to minimise the

distortion of a group of macroblocks at very low bit rates. It was designed to control both instantaneous and overall bit rate of the video sequence through the Lagrange multiplier. The main features of the algorithm can be summarised as follows:

Macroblock Dependency

An image region to be optimised is selected with an objective of minimising the overall distortion of that region subject to a bit-rate budget. Only the dependencies between successive macroblocks are considered. It is necessary to select a reasonably small GOB (group of blocks) such as a single horizontal stripe of macroblocks in order to limit the computational complexity.

Macroblock Modes and Parameters

The coding mode (intra mode, inter 16x16, inter 8x8, and skip mode) for each macroblock is chosen by minimising a Lagrangian cost function over the GOB. Dynamic programming is used to reduce the computational complexity of selecting the least cost mode allocation. However, the motion estimation process is not subject to rate-distortion optimisation, instead the motion vector that produces minimum SAD (the sum of absolute difference of pixels between the original and the predicted block, refer chapter 5 for calculation of the SAD) is selected. The number of encoding options is minimised by keeping QP and other encoding parameters constant for the entire GOB.

Choice of λ

Finding the appropriate λ for a particular rate is an iterative process. However, from the Lagrange theory it is understood that, whatever value is chosen for λ, the distortion will be the minimum for the corresponding rate. Therefore, λ is chosen using a feedback algorithm to control the bit rate without losing optimality. An important contrast to later implementations of rate-distortion optimisation methods in H.263/264 encoders is that the bit rate is directly controlled by the Lagrange multiplier as opposed to by the quantisation parameter.

The results show that this algorithm substantially increases the RD performance relative to the H.263 reference test model encoder TMN5 [54]. Although evidence of required

computational resources is not presented, the algorithm was justified for very low bit rate (where distortion minimisation is crucial) and non real time applications.

4.4.1.2 Lagrange multiplier modelling for rate-distortion optimised mode selection in H.263

A new rate constrained coder control algorithm for H.263 was proposed which was subsequently incorporated into the H.263 test model TMN10 [47]. This algorithm is explained in detail in [55, 56] and also found in [50, 51]. An important aspect of the algorithm is that significant improvements in rate-distortion performance can be achieved within an acceptable level of computational complexity. Some of the main features of the algorithm are as follows.

Macroblock Dependency

A main feature of the algorithm is that it ignores the dependencies between macroblocks and therefore avoids multiple encoding of the same macroblock mode and mode combinations. However, it still retains the ability to set a rate constraint over a GOB. Even though this results in a sub-optimal solution, the computational complexity is reduced by a large extent so that it is feasible to be used in practical encoders. The relative simplicity of the algorithm has been a major factor in incorporating it into H.263 and subsequently H.264/AVC.

Motion Estimation and Mode Selection

The correct mode for each macroblock is chosen from the modes: intra, inter 16x16, inter 8x8, skip, available in H.263. Although cost calculations of intra and skip modes are straightforward, the mode cost of the inter prediction depends on the motion estimation process. The rate and distortion of the inter mode depend on the selected motion vector. Therefore, it becomes necessary to optimise the motion estimation process as well.

To evaluate the resultant rate and distortion, the encoder needs to encode, decode and reconstruct the macroblock using each mode. This evaluation is not computationally intensive for the intra mode and skip mode in H.263. However for inter macroblocks, it is a very expensive process to encode and reconstruct the macroblock for every candidate motion vector during motion estimation due to the large number of candidate motion

vectors in the search area. Therefore, the mode selection and motion estimation processes are separately optimised.

Optimising Motion Estimation

Each candidate motion vector (according to any search strategy) is evaluated by minimizing the following cost function:

$$J_{MOTION} = D_{DFD} + \lambda_{MOTION} * R_{MOTION} \quad (4\text{-}21)$$

Where, D_{DFD} is the difference between current macroblock and the displaced (predicted) macroblock corresponding to the motion vector. It is usually measured as SAD or SSD (sum of squared differences, see chapter 5 for calculation). R_{MOTION} is the number of bits required to transmit the motion vector and λ_{MOTION} is defined as the Lagrange multiplier for the motion estimation process.

D_{DFD} and R_{MOTION} are estimations used in place of the actual distortion and rate of the macroblock as a result of choosing the corresponding motion vector. Due to these estimations, the motion estimation process is optimised by minimising the block difference subject to a constraint on motion vector bits, rather than actual rate-distortion optimisation. Another weakness of the estimation is that D_{DFD} contains both distortion and rate components because it is an estimation of the energy in the residual coefficients. Therefore, the selected motion vector after motion estimation may not be the best one in a rate-distortion sense. However, these approximations are used in the algorithm in order to minimise the computational complexity.

Optimising Mode Selection

Once the appropriate motion vector is selected using the motion estimation process for inter macroblocks, the macroblock is encoded and reconstructed to obtain the actual rate and distortion for that mode. Now the mode selection is carried out by minimising the following cost function:

$$J_{MODE} = D_{REC} + \lambda_{MODE} * R_{REC} \quad (4\text{-}22)$$

Where, D_{REC} is the actual distortion between the original and reconstructed macroblock, calculated using SSD. The actual rate is denoted by R_{REC}. λ_{MODE} is defined as the Lagrange multiplier for mode selection.

The rate and distortion of a macroblock depends on the quantisation parameter as well as the mode. Therefore realistically, the rate-distortion cost of the macroblock should be evaluated for different quantisation parameter values for each mode. This means that each macroblock should be encoded number of 'modes x QP values' times to obtain the optimal solution. However, this is not carried out; the reasons are explained below.

Choice of λ and Rate Control

Lagrange optimisation typically involves minimising the rate-distortion cost for an appropriate λ value that satisfies the rate constraint. Therefore in essence, it is a rate control mechanism that ensures that the distortion is minimized for any chosen rate. For a macroblock, it is not possible to determine the resultant rate for a particular λ before encoding. Therefore, an iterative algorithm is needed to find the right λ for a particular rate constraint.

The rate constraint for a particular group of coding units should be set according to system requirements. The rate constraint may be determined by factors such as output channel transmission rate, encoder/decoder buffer or limitations in the storage media.

Rate control typically involves modifying encoding parameters to maintain a target output bit rate. Although encoding standards do not specify rate control methods, most commonly used rate control mechanisms involve changing the QP to achieve the specified rate. In general, higher quantisation results in low bit rates and vice-versa. A rate control mechanism used in H.263 test model software can be found in [47] and rate control in H.264/AVC reference software is described in [57] [58]. These mechanisms make use of models that adapt to changing macroblock statistics in order to determine the relationship between QP and bit rate. Models are necessary to avoid iterative encoding using different QP values in order to achieve the target bit rate for the macroblock(s). However, models lead to approximate results. Therefore the required rates may not be tightly achieved for each macroblock. In any case, multiple encoding is avoided to minimise the computational complexity and overall delay.

Modelling of λ

One of the most important simplifications of this algorithm comes from the modelling of the Lagrange multiplier to achieve the following:

1. To determine the Lagrange multiplier (corresponding to the required bit rate for the macroblock) in advance prior to encoding the macroblock so that there is no need for an iterative algorithm to search for the correct λ.

2. To integrate Lagrange optimisation into existing rate control methods where QP is the controlling parameter.

While the first point is straight forward, the second point needs some explanation. It is also necessary to discuss the advantages and disadvantages of this approach.

Generally, different QP values should be evaluated along with the available macroblock modes for minimising Lagrangian cost. Results were shown by the authors of [55] for the experiments carried out for various sequences by performing Lagrange optimisation where different QP values were evaluated along with available macroblock modes for selected Lagrange multiplier values. By plotting λ_{MODE} value vs. selected average QP for macroblocks, they determined that the relationship between λ_{MODE} and QP can be approximated by the following model:

$$\lambda_{MODE} = 0.85 \cdot \mathrm{QP}^2 \tag{4-23}$$

This equation implies that, when selecting the best mode-QP combination for a macroblock, a particular QP value will always be selected for a chosen λ_{MODE} (more generally, λ). Although this is not necessarily true for every sequence or different macroblocks of the same sequence, the changes in QP vs. λ relationship for different sequence statistics were considered to be insignificant so that this general rule could be applied to every sequence. However, note that using the same model for every type of sequence will result in sub optimal solutions.

With the model (equation 4-23) available, if the correct λ is known, rate-distortion costs may be evaluated for macroblock modes using only the corresponding QP. Since there is a one-to-one relationship between QP and λ, if the QP is already determined (by a rate

control algorithm), the model can be used to calculate the corresponding λ value that should be used to optimise the mode selection.

Assuming that equation (4-23) is valid for the current MB, the following two processes (see Figure 4.4) should produce the same result.

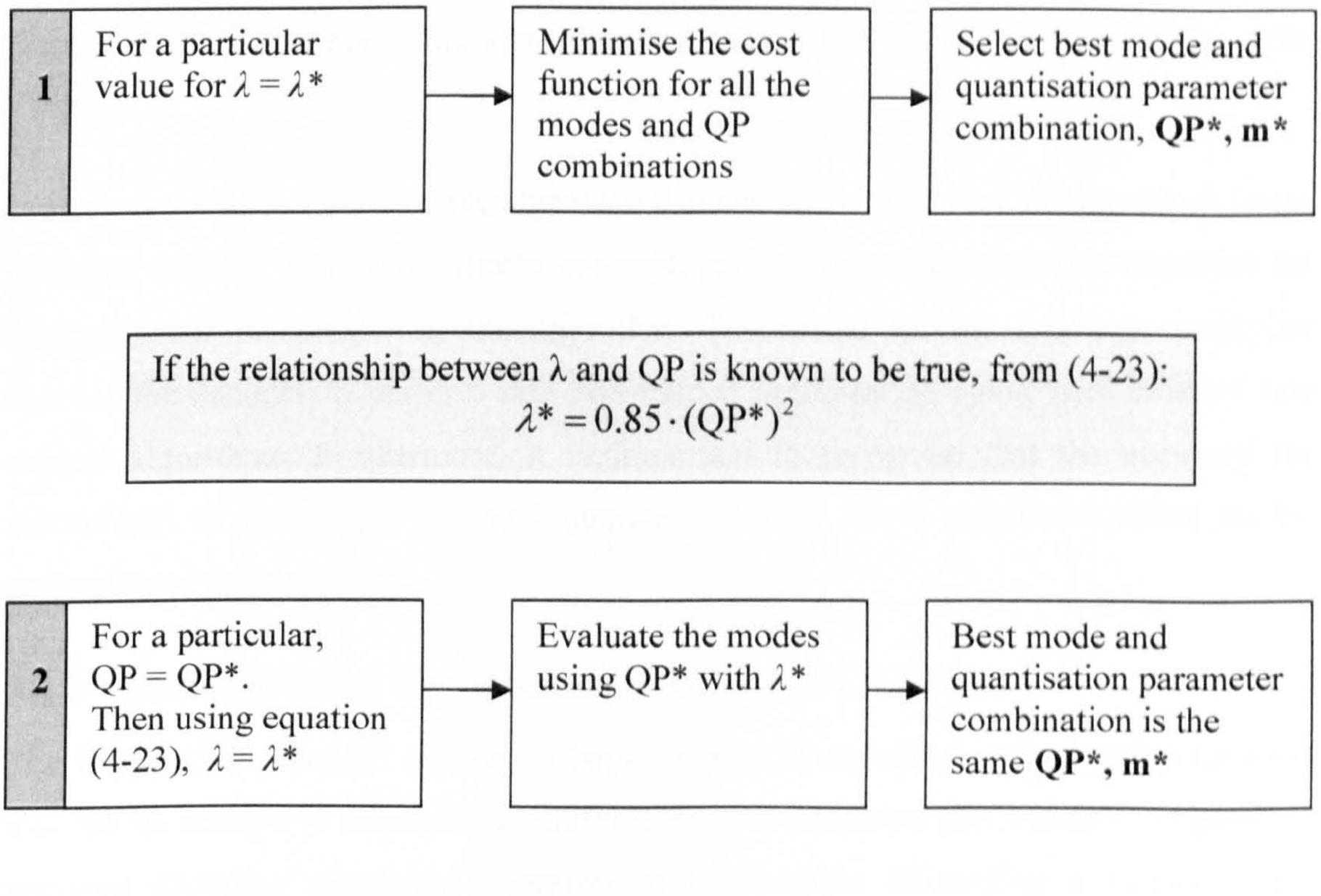


Figure 4.4: Rate-distortion optimised mode selection through Lagrange multiplier modelling

By using the model for the Lagrange multiplier, rate-distortion optimisation has been incorporated into the H.263 test model without changing the rate control mechanism. Therefore, the independent variable is QP rather than λ. The Lagrange multiplier is calculated at the beginning of encoding of each macroblock, according to the quantisation parameter selected by the rate control algorithm.

The following models for λ_{MOTION} has been obtained by authors of [55] through experimental evaluations.

When D_{DFD} is calculated using SSD:

$$\lambda_{MOTION} = \lambda_{MODE} \tag{4-24}$$

When D_{DFD} is calculated using SAD:

$$\lambda_{MOTION} = \sqrt{\lambda_{MODE}} \tag{4-25}$$

It was shown by the authors of [55] that this algorithm significantly out-performed mode selection algorithms based on thresholds by up to 10% bit rate reduction for fixed output picture quality. Therefore, this algorithm was adopted in the H.263 reference encoder TMN10.

In summary, this is a very simple algorithm that can be incorporated into any block based encoding scheme. It uses significant assumptions and approximations that minimise the computational complexity of the algorithm. The model for the Lagrange multiplier enables the encoder to perform rate constrained optimisation along with existing rate control algorithms. Furthermore, it is important to recognize that the accuracy (or inaccuracy) of the model for the Lagrange multiplier has a significant effect on the optimality of the algorithm.

4.4.2 Rate-distortion Optimisation of H.264/AVC Encoder

The H.264/AVC standard supports a large number of encoding options and parameters that can be utilized to substantially improve the rate-distortion performance compared to previous encoding standards. Therefore it is desirable to employ a rate-distortion optimisation algorithm to harness the flexibility offered by the standard. However, the flexibility and large number of encoding options brings a substantial increase in computational complexity.

The simplicity and the performance of the algorithm described in section 4.4.1.2 for H.263 led the same algorithm to be adopted for H.264/AVC. This algorithm is described in section 4.4.2.1. A Lagrange multiplier based rate control algorithm is briefly discussed in section 4.4.2.2.

4.4.2.1 Lagrange multiplier modelling for H.264/AVC

A Lagrangian rate-distortion optimisation algorithm for H.264/AVC is presented in [51]. This algorithm is based on the algorithm discussed in section 4.4.1.2, where similar experiments were carried out to obtain a new model for the Lagrange multiplier as a function of QP for the H.264/AVC reference encoder.

Macroblock Modes

Unlike H.263, a large number of encoding modes, such as 22 intra prediction modes and 259 inter prediction modes are available in the H.264 standard. Also, multiple reference frames can be used for inter prediction. Therefore a large number of modes are available to be evaluated.

Motion Estimation

The same cost estimation as in H.263 (Equation 4-21) is used here. However, D_{DFD} is calculated by using an optional Hadamard transform 'SA(T)D'. It was observed that, carrying out optimisation after transformation produces better rate-distortion performance since the residual is transformed during subsequent encoding [58]. However, the less complicated Hadamard transform is used here to minimise the complexity. R_{MOTION} is modified to include bits associated with partition mode and reference frame selection as well as motion vector values. However, the same limitation applies to motion estimation as in the previous algorithm, i.e. the motion estimation process does not produce a rate-distortion optimised solution due to the simplifying assumptions.

Model for λ

The Lagrange multipliers for H.64/AVC have been modelled using similar experiments as in [55]. The following models and a description of the rate-distortion optimised mode selection can be found in [58].

For I and P frames:

$$\lambda_{MODE,P} = 0.85 \cdot 2^{QP/3} \tag{4-26}$$

For B frames:

$$\lambda_{MODE,B} = \max\left(2, \min\left(4, \frac{QP}{6}\right)\right) \times \lambda_{MODE,P} \tag{4-27}$$

And,

$$\lambda_{MOTION} = \sqrt{\lambda_{MODE}} \tag{4-28}$$

The appropriate λ_{MODE} is used depending on the frame type.

Apart from the key differences stated above, the basic operation of the algorithm is similar to the previous one [55]. Rate control is carried out by controlling the QP rather than the Lagrange multiplier. Therefore, the optimality of the mode selection process depends on the accuracy of the models for the Lagrange multiplier. Since the effect of sequence statistics has not been taken into account in the models, the relative benefits of the rate-distortion algorithm vary depending on the sequence statistics.

4.4.2.2 Lagrangian optimised rate control algorithm for H.264/AVC

The above algorithm (section 4.4.2.1) is designed to integrate the optimisation process into a quantisation parameter based rate control algorithm. The performance of the algorithm significantly depends on the accuracy of the mapping of the correct Lagrange multiplier to QP (Note that the method is already sub-optimal because of ignoring possible dependencies). A method of avoiding this problem is to directly control the rate using the Lagrange multiplier so that QP optimisation is also carried out along with different modes. However, trying to achieve an optimal result (minimum RD bound achievable using the independent assumption) will require a large amount of computational resources. The authors of [52] propose a scalable algorithm to control the per-frame bitrate by adjusting the Lagrange multiplier. With this approach, the optimum macroblock mode-QP combination is selected during the mode selection process. Some important aspects of the algorithm are mentioned below.

Frame Level Rate Control

This algorithm sets the Lagrange multiplier for a frame to achieve a target bit rate. If the required rate is not met after the frame is encoded, the Lagrange multiplier is adjusted and the frame is encoded again. The number of iterations depends on the required accuracy of the resultant bit rate.

Macroblock Level QP-mode Selection

At the macroblock level, optimisation is carried out for different QP values as well as macroblock modes. The authors of [52] assume that the QP value of the current macroblock should be similar to the QP value of the previously encoded macroblock because the 'delta_qp' (the difference between the two QP values) is transmitted and it is

desirable to minimise the number of bits required to send 'delta_qp'. They also show that it is normally sufficient to search 3 QP values for optimum mode-QP combination. Therefore, the initial QP_m value is adapted from the chosen QP of the previous macroblock. The search for the optimum QP-mode combination is carried out for all modes within QP_m-1, QP_m and QP_m+1.

The simulation results show around 0.5dB higher PSNR performance (for the same bit rate) compared with the RD-Optimisation algorithm (algorithm in section 4.4.2.1) and rate control used in the reference encoder JM7.4 [59]. However, the required encoding time is twice that of JM7.4.

4.4.2.3 Summary of Lagrange optimisation methods for H.264/AVC

The relative benefits and drawbacks of the above two approaches may be compared. The benefit of modelling the Lagrange multiplier is that computational complexity is saved because the QP is selected by the rate control and the optimum macroblock mode can be selected using the Lagrange multiplier corresponding to the selected QP. This algorithm can be used with existing rate control algorithms. The main drawback of using the model is the possible inaccurate mapping of the Lagrange multiplier to the optimum QP. For a particular Lagrange multiplier, the selected QP will depend on the sequence statistics and other available mode options. The inaccuracies of the model lead to suboptimal solutions. This is highlighted by the performance of the second algorithm which can search for a better QP during optimisation, achieving 0.5dB better rate-distortion performance. However, the drawbacks of the second algorithm are the high computational complexity involved in the process and the inability to incorporate existing model-based rate control mechanisms.

4.5 Conclusion

The H.264/AVC video coding standard specifies only the coding syntax and the decoding process of the coded bit stream. This enables flexible encoder implementations. H.264 has a wide range of encoding modes and parameters that contribute to the high compression efficiency of the standard. However, the encoder needs to choose the correct encoding modes and parameters during the encoding process to achieve optimum rate-distortion performance.

The Lagrange multiplier method is a powerful optimisation tool for optimal allocation of constrained resources to maximise the output. In video coding, the Lagrange multiplier method can be used to optimally allocate coding modes and parameters over a set of coding units to minimise the overall distortion subject to a bit rate constraint. However, obtaining the optimal solution is a highly complex task due to interdependencies between coding units. Therefore, practical Lagrange optimisation algorithms tend to ignore these interdependencies, in order to keep the computational complexity at a manageable level.

This chapter discussed algorithms which use Lagrange optimisation, developed for both H.263 and H.264/AVC coding standards. The most successful algorithms involve modelling of the Lagrange multiplier as a function of the QP because of the lower computational complexity and because they can be used in conjunction with existing rate control mechanisms.

5 Experimental Method

5.1 Introduction

The experimental method used in this research project is outlined in this chapter. The following sections describe the method of implementation and testing of the algorithms, test video sequences used and performance evaluation methods.

5.2 Implementation and Testing

The implementation and testing of the algorithms are carried out by software simulation. A software video codec running on a PC is used to test the algorithms developed during the project.

5.2.1 Video Codec

In this work, the H.264/AVC Reference software codec JM7.3 [46] (and JM7.4 [59] for some experiments, referred to as the JM codec) is used as the reference video codec. The JM codec is commonly used to test new algorithms in the video coding community. The use of this reference software enables realistic comparison of the performance of different algorithms developed by different researchers. The source code (in the C programming language) for the JM codec can be downloaded from http://iphome.hhi.de/suehring/tml/download/. The earlier and the later versions of the JM codec and the revised manual [60] for the H.264 reference software can also be found at the above website.

The JM encoder reads input parameters from a configuration file. A wide range of encoding parameters can be changed using the configuration file. These include (but are not limited to):

- Input video sequence (concatenated YCrCb 4:2:0 format)
- Quantisation parameters for I, P and B slices
- Available macroblock partition modes
- I, P and B picture sequence

- Number of reference frames

- Rate-distortion optimisation - ON/OFF (see section 4.4.2.1)

The JM codec also provides useful encoding statistics such as bit rate of the encoded bit stream (for a given frame rate), video quality in PSNR (discussed later) of luminance and chrominance components of the coded video and encoding time. The encoder has an option to output a trace file which contains the syntax elements of the coded bit stream. The trace contains the name of the syntax element, the value (in decimal) and the actual bits. This is very useful for understanding the coded bit stream and detecting bugs in algorithms.

The decoder access a configuration file to read data, such as the file name of the encoded bit stream. The decoder provides statistics such as decoding time and video quality along with the output video sequence.

5.2.2 Test Platform

A personal computer with the following specifications is used as the test platform for the software video codec.

Processor	: Intel Pentium IV CPU 3.00 GHz
Memory	: 512 MB
Operating System	: Microsoft Windows XP Professional (Service Pack 2)

5.2.3 Development Environment

The JM codec source code is compiled and built using Microsoft Visual C++ Professional Version 6.0. This development environment is used to modify the source code of the JM codec to incorporate the algorithms to be tested.

5.2.4 Testing the Algorithms

Figure 5.1 shows a typical test scenario. The reference software is modified to incorporate the algorithm to be tested. A test video sequence is encoded using both the original reference encoder and the modified reference encoder. Performance indicators such as computational complexity, video quality and bit rate are measured. For experiments that

require subjective video quality evaluations (see section 5.4.3.1), the encoded test video sequences are decoded using the JM decoder.

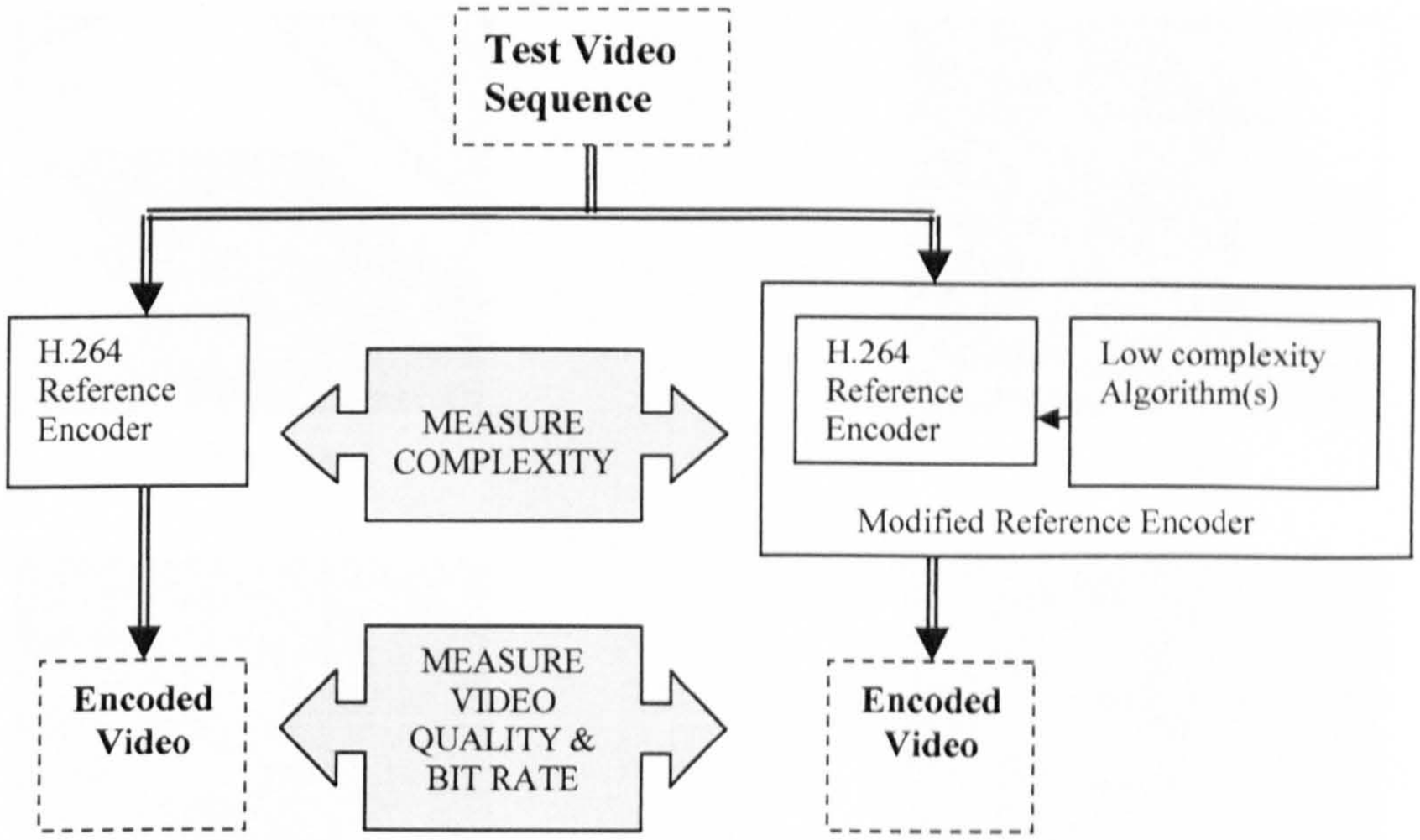


Figure 5.1: A typical test scenario

The following sections describe the test video sequences used for the experiments and the methods used to evaluate performance such as video quality, bit rate and computational complexity.

5.3 Test Video Sequences

The test video sequences are mostly chosen from widely used video material in video coding research. The test sequences include different foreground and backgrounds, motion, detail and camera movements. Some of the test sequences can be downloaded from http://www.vid-labs.com/resources/resources.html. Sample frames from sequences (in QCIF format) which are used to test low complexity algorithms are shown in Figure 5.2. The main video sequences are described below.

(a) Foreman: This video clip shows a person talking to the camera at a construction site. The man gives an animated talk by moving his head and hands and with facial expressions. There is a considerable amount of camera shake because of the handheld camera. Towards the end of the clip, the camera rapidly moves away from the man's face

and shows the construction site. This video sequence is characterised by high motion and detail.

(a) Foreman

(b) Carphone

(c) Mother & daughter

(d) Claire

Figure 5.2: Sample frames from test video sequences in QCIF format- (a) Foreman, (b) Carphone, (c) Mother & daughter and (d) Claire

(b) Carphone: This video is characterised by moderate to high motion and medium detail. A man is talking to the camera inside a moving car. He uses a high amount of facial expressions and body movement. The background through the car window changes as the car moves. The camera moves slightly.

(c) Mother and daughter: This is a scene of a woman and a child sitting in a room. The woman talks to the camera while stroking the child's hair. There are some head and hand movements with moderate amount of detail. The camera is static. This video sequence has low to moderate movement.

(d) Claire: This video sequence has very low motion and detail. A woman is talking to the static camera. Only small movements of the head and the face are visible. The background is plain and static.

These video sequences are chosen so that the performance of the algorithms can be assessed for varying input video statistics ranging from low to high motion and detail. The use of these well known test video sequences helps to evaluate and present the results in a way that is comparable to other research in the industry.

Chapter 6 of this thesis presents a new subjective video quality assessment technique. The following two test video sequences in CIF format are used for evaluating this technique.

(a) Mobile and Calendar (b) Violin

Figure 5.3: Sample frames from CIF video sequences – (a) Mobile and calendar, (b) Violin

(a) Mobile and calendar – This video sequence has large amount of detail where, the background consists of a detailed colourful picture. The calendar (also with high detail) is moved vertically (up and down) and the toy train moves horizontally while pushing a ball. The camera zooms outs and pans, following the movement of the train. This results in a very high motion because objects are moving in many directions simultaneously (this video sequence is also commonly referred to as "Mobile").

(b) Violin – This clip is of a girl playing the violin. Although the girl does not move much, there is significant camera motion with zooming out, camera shake (hand held camera) and panning. The music sheet in the background provides fine detail which can be immediately noticed.

In addition to the test video sequences described so far, the well known sequences Akiyo, Coastguard, Container, Hall objects, News, Silent, Suzie, and Table Tennis are also used during the development of the algorithms.

5.4 Performance Evaluation

The performance of the algorithms is evaluated by measuring the computational complexity, coded bit rate and video quality. The video compression performance or "rate-distortion performance" is generally assessed by plotting the video quality against the bit rate as in Figure 5.4. In this example, the results of two algorithms are plotted in the same figure. The plots show that algorithm 'A' achieves a particular video quality ('Q') with a smaller number of bits than algorithm 'B'. Therefore, the rate-distortion performance of the algorithm 'A' is better than the rate-distortion performance of the algorithm 'B'.

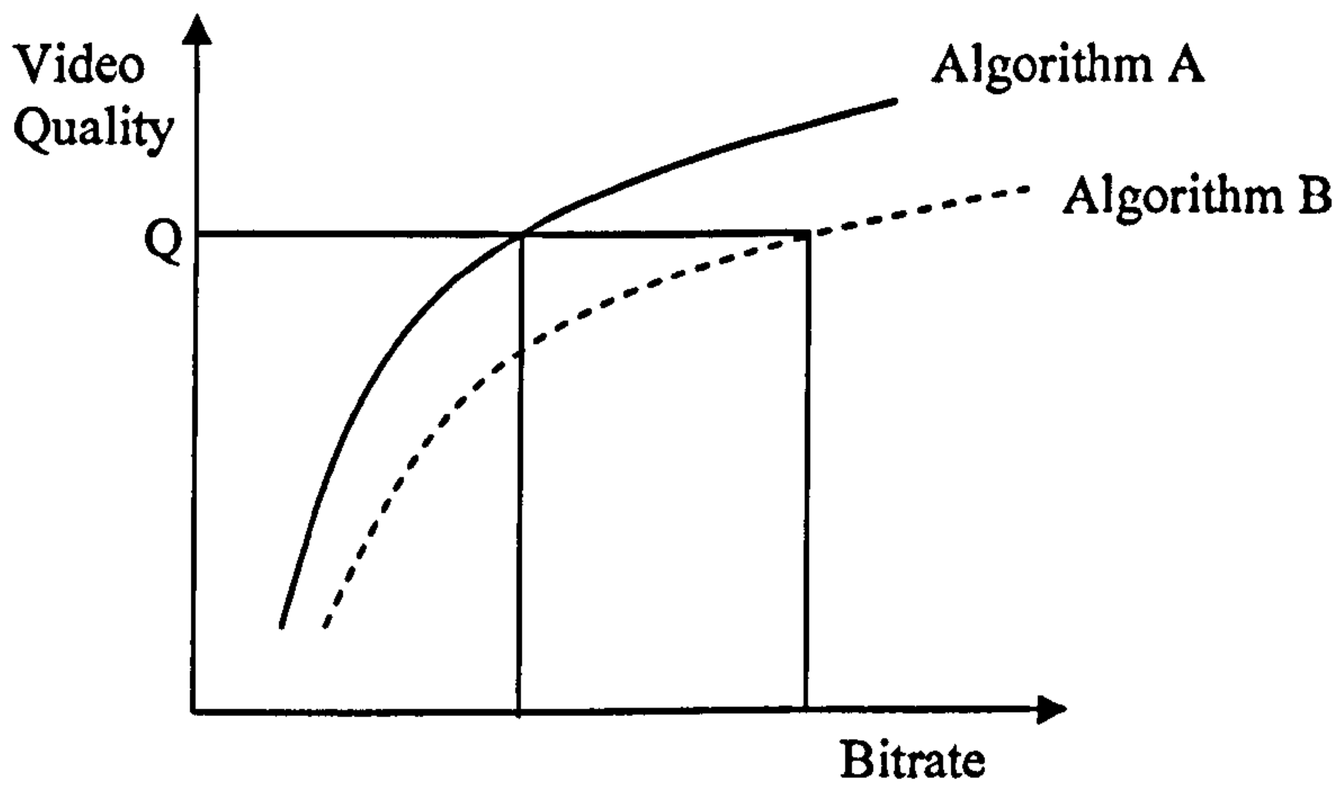


Figure 5.4: An example rate-distortion plot

When the performance of low-complexity algorithms is evaluated, typically the rate-distortion performance of the low-complexity algorithm is plotted, with the rate-distortion performance of the reference encoder. The rate-distortion performance of the algorithm and the reference encoder is compared, along with the actual computational complexity savings achieved by the low complexity algorithm. For variable complexity algorithms, the rate-distortion plot can be extended into a 3-dimensional surface by plotting the rate-distortion performance against a third complexity axis (complexity-rate-distortion

performance). Methods of evaluating computational complexity, bit rate and video quality are discussed in the following sections.

5.4.1 Computational Complexity

The computational complexity can be measured in number of different ways according to the requirements of the experiments and the analysis.

5.4.1.1 Encoding time

The time spent by the encoder encoding a particular number of frames, the average encoding time per frame or the time spent on a particular function such as motion estimation, can be used as a measure of computational complexity. The encoding time can be compared between the reference encoder and the modified (with the algorithm to be tested) encoder or between encoders with different algorithms.

However, care should be taken to maintain the same test conditions for each encoder. For example, for an encoder running on a PC, each encoder must be given the same amount of processing resources by making sure that only the same essential system processes are running during each test.

5.4.1.2 Software profiling

Information about processor utilization can be obtained by using software profiling tools such as the one included in Microsoft Visual C++ Professional Version 6.0. The tools provide information about the time spent by each function of the program such as motion estimation and intra prediction.

5.4.1.3 Number of processed elements

This is a useful measurement of computational complexity when evaluating algorithms developed to reduce the number of elements processed. For example, later chapters (chapters 8, 9, 10 and 11) of this thesis discuss macroblock skip prediction algorithms aimed at reducing the number of macroblocks processed by the encoder. Therefore, the number of macroblocks processed per picture by the modified encoder (with the skip-prediction algorithm) can be compared with the reference encoder in order to measure the complexity reduction of the algorithm compared to normal encoding.

5.4.2 Bit Rate

Typically, the average bit rate (bits per second) is calculated by measuring the number of bits in the encoded bit stream, the number of frames and the frame rate (frames per second) as follows:

$$BitRate(bps) = \frac{FrameRate(fps) \times NumberOfBits}{NumberOfFrames} \quad (5\text{-}1)$$

The statistical data of the encoded video sequence can be obtained from the JM encoder either to a standard output or to a text file. This data includes the average bit rate of the encoded video sequence.

5.4.3 Video Quality

Video quality is an important measure of performance for video coding algorithms because most video compression algorithms are "lossy", i.e. compression is achieved at the expense of video quality. Video quality is a subjective measure: different people perceive video quality differently. Therefore, it is difficult to measure video quality in a reliable manner. In this section the subjective and objective measures which have been devised to measure the video quality are discussed.

5.4.3.1 Subjective video quality measurement

Subjective video quality measurements are carried out to assess the perceived quality of a video sequence. These involve a number of observers assessing the quality of the video and using these assessments to form a general conclusion about the video quality, called the Mean Opinion Score (MOS). The perceived video quality can be influenced by number of factors such as the picture quality, frame rate and jitter (the variation in frame rate). Subjective tests can capture all these factors and, therefore, forms a realistic opinion about the actual video quality. There are a number of standard subjective video quality assessment methods specified in [61] for multimedia applications. Methods for subjective assessment of television pictures are recommended in [62]. This work is concerned with subjective quality assessment methodology specified in [61] as it deals with algorithms developed for multimedia applications such as video conferencing and mobile video.

The following three methods are commonly used in subjective video quality assessment.

Absolute Category Rating (ACR)

The test video sequences are presented one at a time ("single stimulus" method) and the observer is asked to rate them according to the following 5 level scale:

5 – Excellent

4 – Good

3 – Fair

2 – Poor

1 – Bad

The observers are given a voting time of less than or equal to 10 seconds between each sequence. Typical presentation timing is shown in Figure 5.5.

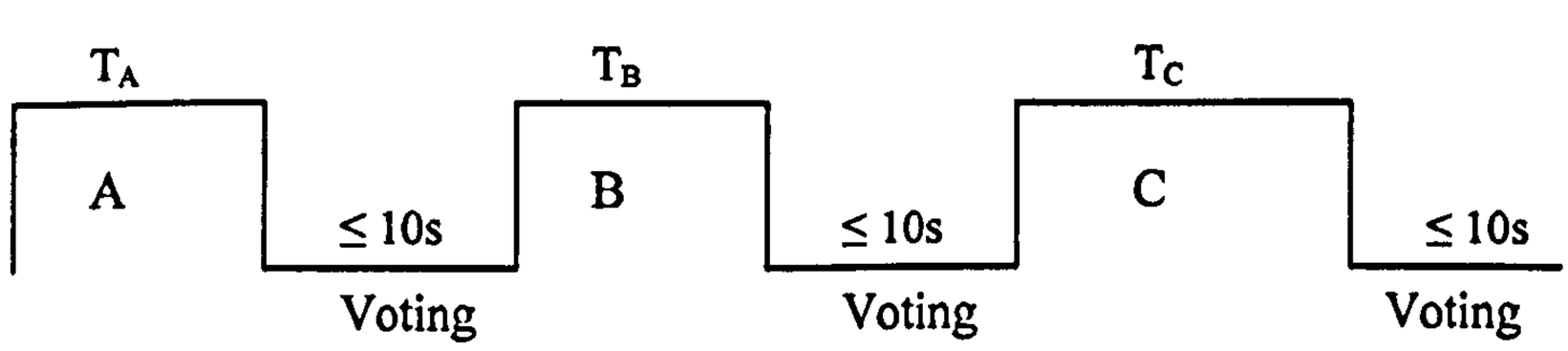


Figure 5.5: Presentation timing of sequences A, B and C

It is recommended to repeat each sequence at a different time to obtain more reliable ratings. This method is useful to obtain an absolute measure (how acceptable the video sequence is) of the quality of video sequences.

Degradation Category Rating (DCR)

The test video sequences are presented in pairs, where the first video sequence is the reference and the second is the video sequence under test. The sequence under test is compared with the reference and the quality is rated on a degradation category scale.

5 – Imperceptible

4 – Perceptible but not annoying

3 – Slightly annoying

2 – Annoying

1 – Very annoying

The reference and the test sequence pairs can be shown simultaneously (reference on the left and sequence under test, on the right) for smaller picture sizes such as CIF and QCIF. It is desirable to synchronise the left and right had sequences. Similar to ACR a voting time of 10 seconds or less is allowed between sequence pairs. This method is useful to assess the video quality of coded video using a particular algorithm, compared with a reference video.

Pair Comparison Method (PC)

This method is useful to compare number of video sequences produced using different systems under test. For example, to compare video encoded through 3 different coding algorithms (say A, B and C).

With this method, test sequences are shown in pairs (one after the other or side by side for smaller sizes like CIF and QCIF). These pairs are shown in all possible orders as below:

AB, BA, AC, CA, BC, CB

However, the sequence of pairs should not necessarily be in this order; they can be in random order. In general, for 'n' systems under test, there are possible n(n-1) combinations to be tested.

The observers are asked to mark the preferred sequence in every pair. Here, each sequence pair is automatically replicated. Presentation timing should be similar to previous tests with voting time limited to 10 seconds.

Experimental Considerations

The recommendation [61] also provides general guidelines for designing the tests. The number of observers (or users/viewers) should be between 4 and 40. However, it is recommended to have at least 15 observers for the test. The observers should not be experts in picture quality evaluations or should not be involved in similar work in day to day life. There should be at least 2 repetitions of each test condition (for ACR and DCR) in each subjective test, for the results to be reliable. Furthermore, viewing conditions, environmental conditions and guidelines for the instructions given to the viewers are also specified.

The statistical accuracy of the results of subjective video quality assessment methods discussed above depends on employing a sufficient number of observers and using repetitions of test sequences. This is a time consuming and costly process. A new method is discussed in chapter 6, which employs user feedback and produces similar results in a fraction of time required for a standard subjective test such as the pair comparison test.

5.4.3.2 Objective video quality measurement

Objective video quality measurements are used to measure the video quality, typically in situations where fast (sometimes online) and repeatable measurements of the distortion or the difference between the video under test and a reference video are needed.

PSNR

The Peak Signal to Noise Ratio (PSNR) is the most commonly used objective measure of video quality. PSNR is measured as follows:

$$PSNR(dB) = 10 \cdot \log_{10} \frac{(2^n - 1)^2}{MSE} \tag{6-2}$$

Where n is the bit depth and MSE is the Mean Squared Error between corresponding pixel values of the original image and the current image of the sequence under test.

For $M \times N$ array of pixels, MSE is given by:

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (P_O(i,j) - P_t(i,j))^2 \tag{6-3}$$

Where $P_O(i,j)$ denotes a pixel from the original image and $P_t(i,j)$ denotes the corresponding pixel from the test image. The parameters 'i' and 'j' point to a position in the pixel arrays. The MSE in itself can be a measure of distortion. However PSNR is preferred because the log scale provides a more realistic mapping to quality variations. Note that it is assumed that the original (perfect) video sequence is available for PSNR calculation. The PSNR of a complete video sequence is generally taken as the average PSNR over all the frames.

A higher PSNR generally relates to higher video quality and lower PSNR relates to lower quality. However, there are instances where PSNR does not correlate well with subjective video quality as shown in [63] (also see the example shown in [64]). Another drawback of PSNR is that it does not take into account temporal aspects such as frame rate and jitter. Different approaches for objective video quality measurement are discussed in [65] [66]. These methods try to accommodate the aspects of human visual system that influence how people perceive video quality. However, they fail to provide accurate measures that can be calculated without committing a large amount of computational complexity. Therefore, PSNR continues to be the most commonly used objective quality measure.

In addition to the PSNR, the difference between an original picture (or a region) and test picture (or a region) can also be evaluated using the following measures.

SAD

The Sum of Absolute Differences (SAD) is calculated as follows.

$$SAD = \sum_{i=1}^{M}\sum_{j=1}^{N}\left|P_O(i,j) - P_t(i,j)\right| \tag{6-4}$$

The same notation as in equation (6-3) is used. The SAD can be calculated with low computational complexity using only additions and subtractions.

SSD

The Sum of Squared Differences (SSD) is calculated as:

$$SSD = \sum_{i=1}^{M}\sum_{j=1}^{N}(P_O(i,j) - P_t(i,j))^2 \qquad (6\text{-}5)$$

The same notation as above is used. SSD can be calculated without using divisions. Note that the difference between SSD and MSE is that MSE is the average of SSD over all the pixels in the region.

Both SSD and SAD calculate the amount of residual energy between the two pictures or regions. Therefore, they are used as low complexity approximations to the objective video quality.

5.4.3.3 Summary of video quality measurement

Subjective video quality assessment methods tend to be time consuming and expensive because they involve a number of observers watching video sequences and rating them according to a particular scale. However, they are needed for a realistic measure of perceived video quality. Objective measures are faster and produce more consistent results; however, they do not always correspond to actual subjective quality. PSNR is the most commonly used objective measure but is known to correlate poorly with subjective quality in some scenarios.

5.5 Conclusion

This chapter presents the experimental methods used in this research project. The algorithms developed during the project are tested using software simulation. The H.264/AVC reference software codec is used as the reference codec and the algorithms are implemented by modifying the source code of the reference encoder. Standard test video sequences are used to evaluate the performance of the algorithms. The performance of the algorithms is evaluated by measuring the computational complexity, bit rate and subjective and/or objective video quality.

The next chapter presents a new subjective video quality assessment technique aimed at significantly reducing the time required to carry out subjective quality tests, compared to the standard methods described in this chapter.

6 Video Quality Measurement with User Feedback

6.1 Introduction

A new subjective video quality assessment technique named "User Feedback Quality (UFQ)" method is presented in this chapter. This new method was presented in [13] and is also the basis for a European patent application [14]. The UFQ method, as the name suggests, incorporates the feedback from user (or observer / test subject) to control the output video quality as part of a rapid video quality measurement method. This method is well suited to test scenarios where the user is required to select the preferred video sequence from a range of candidates. A feedback mechanism enables the user to control the display of the video and to converge on the preferred video sequence within a short period of time. Compared with the standard pair comparison (PC) method, the UFQ method produces comparable results in a fraction of the time normally required to carry out subjective video quality assessment.

This chapter is organised as follows. Section 6.2 explains the background and the motivation behind the development of the UFQ method. The user feedback quality (UFQ) method is described in section 6.3. The experimental method is presented in section 6.4 followed by results & discussion in 6.5 and conclusion in section 6.6.

6.2 Background

The earlier chapter on experimental methods (chapter 5) described subjective and objective video quality measurement methods. Objective methods such as PSNR provide automatic and repeatable results. However, they do not accurately measure the perceptual quality. Subjective methods provide a realistic guide to actual perceived video quality. Obtaining an accurate outcome from subjective tests typically involves carrying out many time-consuming tests with a large number of test subjects, making them costly and difficult to carry out. The UFQ method described in this chapter addresses the problem of reducing the time taken to carry out subjective video quality evaluation.

A typical subjective test requirement is to determine the preferred choice from a set of alternative versions of a video clip, for example, versions of the same clip encoded with different encoders, alternative strengths of a post-processing filter and alternative trade-offs between frame-rate and image quality for a given bit rate. In this type of scenario, the conventional Pair Comparison (PC) method can be used. As described in the previous chapter (section 5.4.3.1), the alternative versions of a video clip must be compared with each other using all the possible combinations and graded by the test subjects. The time required to carry out a test increases linearly with the number of alternatives to be tested. Table 6.1 shows the number of pair combinations and the estimated time to carry out a subjective test for varying number of alternative versions of video clips. It is assumed that the running time of each video clip is 15 seconds, the video sequence pairs are displayed simultaneously side by side and the voting time for each pair is 10 seconds.

Table 6.1: Estimated test durations for PC method

No. of alternative versions (n)	No. of pair combinations n(n-1)	Estimated test duration
3	6	2 mins, 30 secs.
5	20	8 mins, 20 secs.
10	90	37 mins, 30 secs.

Therefore, in some cases (for example, choosing a preferred trade-off between frame rate and image quality), when there is a large number of possible outcomes, the test designer is faced with a choice between:

a) running a very large number of time consuming tests in order to obtain a fine-grained result or;

b) limiting the number of tests at the expense of the granularity or accuracy of the result.

Experience in carrying out subjective tests shows that most test subjects get tired and lose concentration after the first 10 - 15 minutes of the test. Therefore, time consuming tests may not produce accurate results.

The User Feedback Quality measurement (UFQ) method described in the next section is particularly well-suited to the test scenario described above, i.e. choosing between a number of alternative coding or processing options. The UFQ method incorporates feedback from the user, such that the user actively selects the preferred alternative.

6.3 The User Feedback Quality Measurement Method

The User Feedback Quality (UFQ) method requires the user to interactively choose a preferred video clip from a set of alternative options. The user is presented with a continuous video display (either a long video sequence or a short sequence presented in a loop) and interacts with the display using an input control (for example, an on-screen slider bar or a slider-type input device). The slider position controls the display of the video clip, either by controlling video processing in real time or by selecting from a number of pre-processed video clips. In the situation where the slider selects from a number of video clips, the software maintains the time position in the video sequence so that the clip is not interrupted. The user moves the slider until the preferred video clip is selected. The slider position is recorded (sampled) at regular time intervals.

When the test ends (typically at a point chosen by the user), the "optimum" choice of parameters for this user is determined from the recorded final slider position. The final slider position reflects the user's preferred video quality. A record of the slider's position during each test may also give some insight into the user's response to the video clip, for example an oscillating position may indicate uncertainty on the part of the test subject.

The UFQ method has the potential to deliver subjective test results in a fraction of the time required for traditional methods, since the preferred outcome of a particular user is reached in a single "pass" of the test (where a "pass" means that the user steps through the different video quality levels in a preferred manner that may be unique to each user). The UFQ method differs from continuous evaluation methods such as "Simultaneous Double Stimulus for a Continuous Evaluation" (SDSCE, described in [61]) because in UFQ, the test subject interacts with the displayed video sequence in contrast to SDSCE, where the slider is used only as a voting mechanism. The use of a slider to allow a user to control the rate of transmission errors in coded audio data is described in [67] [68] but the authors do not address the application of user feedback to measure subjective video quality.

6.4 Experimental Method

An experiment was carried out in order to evaluate the performance of the UFQ method in comparison with Pair wise Comparison (PC) method. Two CIF video clips (Mobile – 300 frames and Violin – 300 frames) were encoded using the H.263 test model encoder TMN8 [69]. Four different versions of each video clip were encoded, each with a different frame rate and a constant coded bit rate. This is a typical trade-off experienced in coding scenarios, i.e. choosing between temporal and spatial quality, constrained by a maximum bit rate. Table 6.2 & Table 6.3 lists the mean quantisation parameters and obtained bit rates at different frame rates for the two sequences.

Table 6.2: Mobile – different frame rates at constant bit rate

Frame Rate (fps)	Mean Quantiser	Achieved Bit Rate (kbps)
30	27.77	384.23
15	19.88	385.74
10	15.67	386.66
7.5	13.08	386.60

Table 6.3: Violin – Different frame rates at constant bit rate

Frame Rate (fps)	Mean Quantiser	Achieved Bit Rate (kbps)
25	23.45	128.22
12.5	15.65	128.94
8.33	12.54	128.46
6.2	10.77	128.89

Using the UFQ software application, the user switches between different versions of the sequence using an on-screen slider bar. The user is presented with a continuous, repeating video clip and moves the slider bar until the preferred video clip is selected. Switching between video clips is synchronized so that there are uninterrupted smooth transitions. Therefore, viewers only notice the change in video quality with continuous playback of video. The software records the slider position (and hence the chosen video clip) at regular time intervals. The user is instructed to choose the final slider position that

corresponds to the preferred video clip. Figure 6.1 shows a screen shot of the UFQ software application showing the "Violin" video sequence.

In this experiment, each participant assessed one set of alternative versions of a clip (either Violin or Mobile) with the UFQ method and the other set of clips with the PC method. There were 22 participants in total, i.e. 11 sets of test results for each method with each video clip.

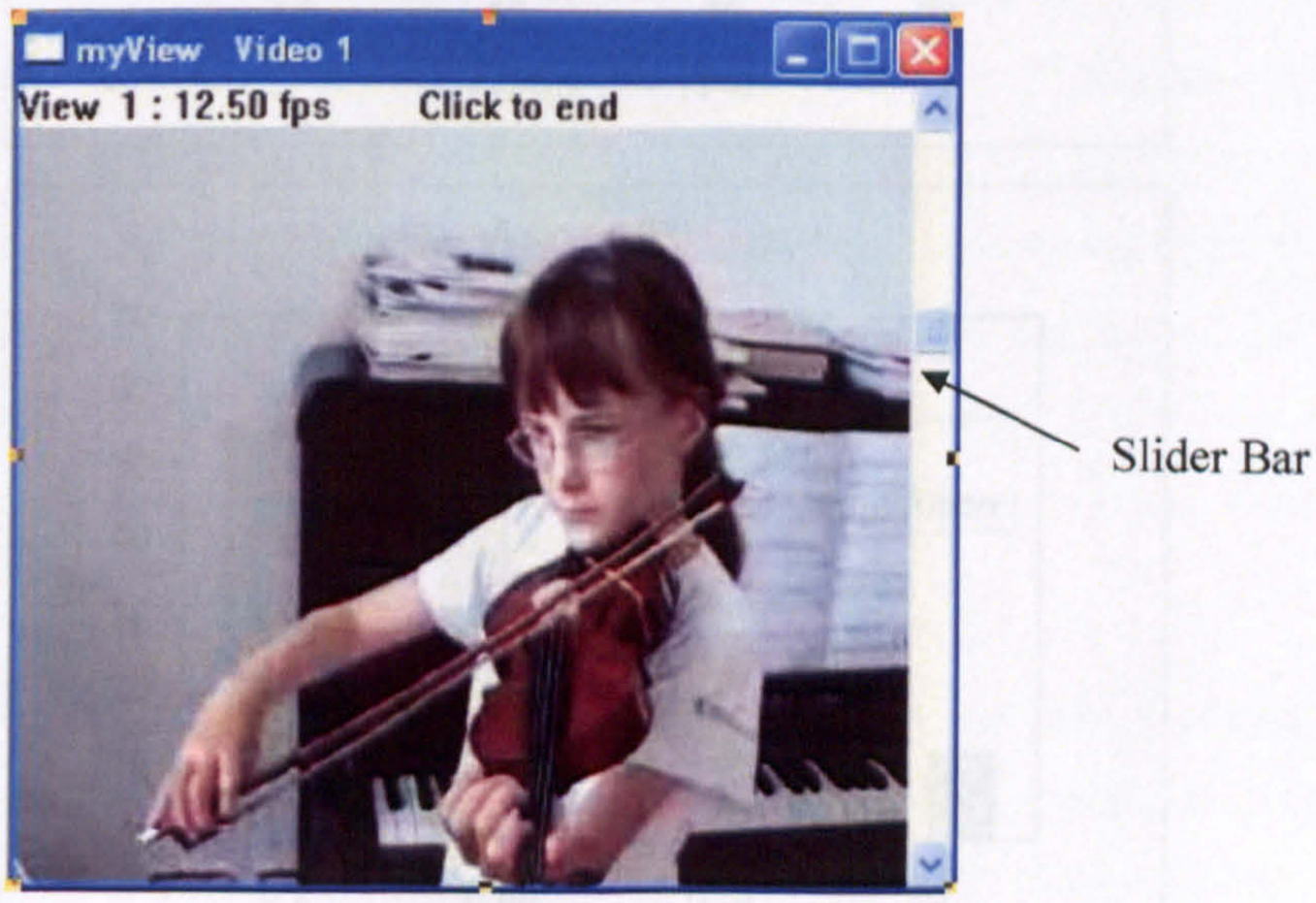


Figure 6.1: Screenshot of UFQ software application

The PC assessment was carried out according to the procedures described in [61]. Pairs of clips (two versions of the same clip coded at the same bit rate but with different frame-rates) were presented side by side on a grey background and the user was asked to choose the clip with the "best" video quality. The UFQ assessment was carried out as described above, i.e. the test subject was presented with a continuously running clip and was asked to select the slider position that gave the "best" video quality. The total time required to carry out each test (PC and UFQ) was recorded.

6.5 Results and Discussion

The preferred choice of video frame rate for each of the two sequences is shown in Figure 6.2. Table 6.4 lists the average time (in seconds) required to carry out each test (PC and UFQ) together with the time saving achieved by the UFQ method.

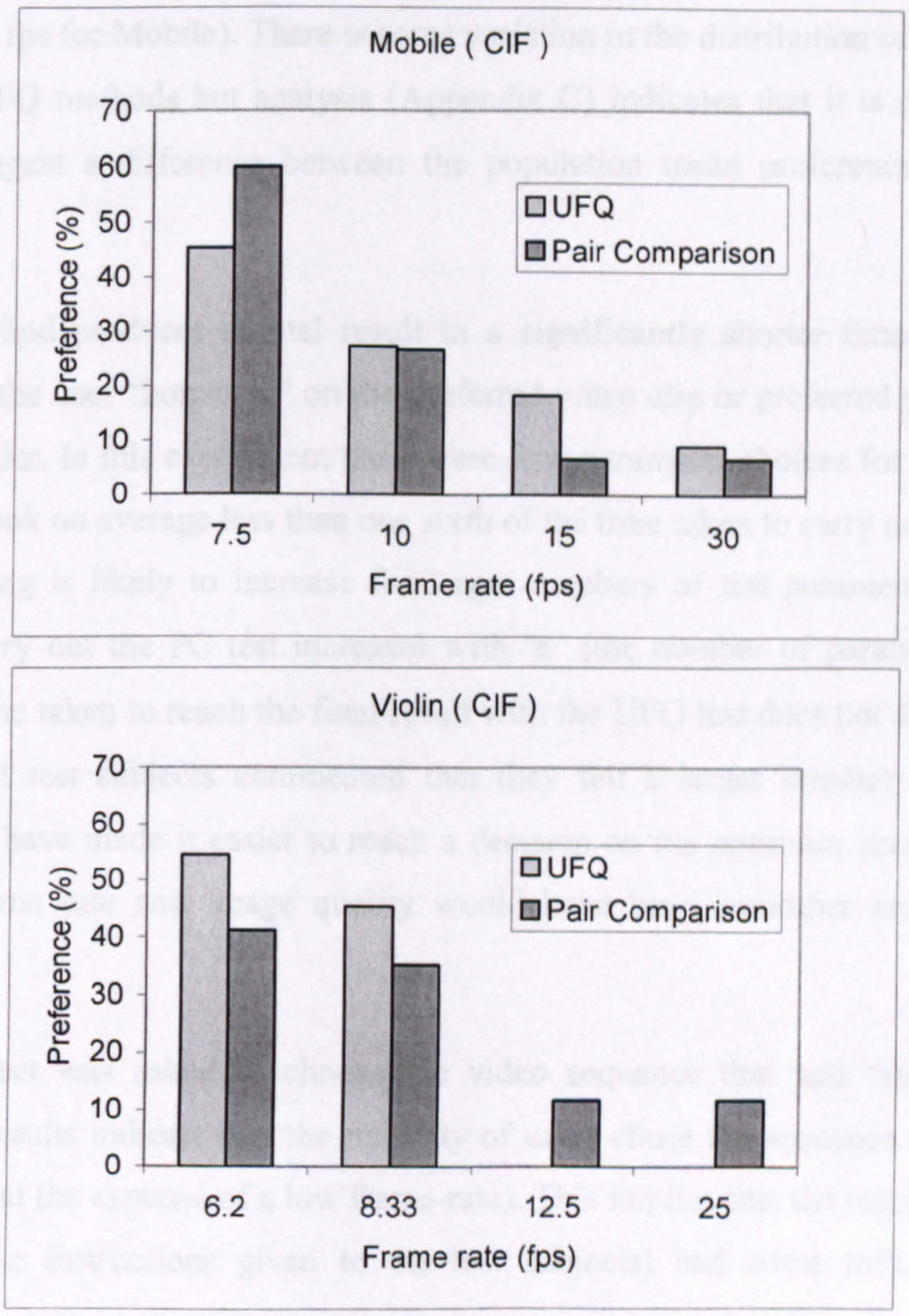


Figure 6.2: Distribution of user preferences

Table 6.4: Average time taken to complete each test

Video Sequence	PC	UFQ	Time Saving
Violin	348 seconds	47 seconds	86%
Mobile	388 seconds	65 seconds	83%

Figure 6.2 demonstrates that there is a clear similarity between the responses obtained from the two test methods. In each case, the sequence with the lowest frame rate and hence the highest image quality has the highest frequency of preference (6.25 fps for

Violin and 7.5 fps for Mobile). There is some variation in the distribution of responses for the PC and UFQ methods but analysis (Appendix C) indicates that it is not significant enough to suggest a difference between the population mean preferences of the two methods.

The UFQ method produces a final result in a significantly shorter time than the PC method, since the user "homes in" on the preferred video clip or preferred parameters by moving the slider. In this experiment there were four parameter choices for each clip and the UFQ test took on average less than one sixth of the time taken to carry out the PC test. This time saving is likely to increase for larger numbers of test parameters. The time required to carry out the PC test increases with 'n' (the number of parameter choices) whereas the time taken to reach the final result with the UFQ test does not depend on 'n'. In fact, several test subjects commented that they felt a larger number of parameter choices would have made it easier to reach a decision on the optimum choice, since the changes in frame rate and image quality would have been smoother and more fine-grained.

Each test subject was asked to choose the video sequence that had "the best video quality". The results indicate that the majority of users chose the sequence with the best **image** quality (at the expense of a low frame-rate). This implies that the test protocol (and in particular the instructions given to the test subjects) had some influence on the subjective test outcome.

6.6 Conclusion

This chapter describes a new method of subjective quality evaluation for digital video, the User Feedback Quality (UFQ) method. The new method enables the user to control the video display and select the preferred video quality using a slider control. The experimental results indicate that the UFQ method achieves the same result as established subjective quality assessment methods such as the pair comparison method, in a fraction of the time. The method significantly reduces the time taken to carry out subjective quality assessment and therefore has the potential to reduce the cost of testing and/or enable an increased number of tests to be carried out. The UFQ method is particularly well-suited to measure visual quality where there is a choice between a number of parameters such as different frame rates, alternative coding algorithms or alternative post-

processing algorithms and has the added benefit of supporting a very fine-grained quality assessment result.

The primary objective of this research is complexity management of H.264/AVC compliant video encoders. The following chapters describe the main contributions of this research in relation to the primary objective. The next chapter (chapter 7) presents a performance evaluation of the H.264/AVC standard, followed by a review of existing low-complexity algorithms for H.264/AVC. This chapter identifies the lack of techniques for effective computational management.

7 Low Complexity Video Coding

7.1 Introduction

The H.264/AVC standard supports a variety of coding tools and options in order to achieve high compression efficiency. However, gains in compression efficiency are achieved at the cost of increased computational complexity. The first part of this chapter (section 7.2) evaluates coding tools and coding options in H.264/AVC in terms of rate-distortion performance and computational complexity. The second part of this chapter (section 7.3) investigates existing low complexity and variable complexity algorithms with particular emphasis on algorithms developed for H.264/AVC encoders. Finally, section 7.4 explains the motivation behind the low-complexity algorithms developed during this project.

7.2 Performance Evaluation of H.264/AVC

This research is aimed at low complexity implementations of H.264/AVC in processor/power constrained environments. A typical target application is video over wireless networks with encoders running on mobile handheld devices. The baseline profile of H.264/AVC suits these applications because of its comparatively low complexity and the availability of error resilience tools such as ASO and FMO. Therefore this research is primarily focused on the implementation of baseline profile encoders. The analysis in this chapter is limited to the baseline profile of the H.264/AVC standard. The JM7.4 reference software is used with the following "default" configuration (see Table 7.1):

Table 7.1: JM7.4 – Default configuration

Parameter	Value
Slice structure	1 slice per frame (IPPPP..)
Rate-distortion optimisation	OFF
Number of reference frames	1
Motion estimation (full) search range	16
Macroblock partition modes	All

All other input parameters of JM7.4 are kept constant. Experiments are carried out to:

- Compare the rate-distortion performance and computational complexity between inter and intra prediction,
- Measure the computational complexity of motion estimation and mode decision and
- Evaluate the rate-distortion and complexity performance of different macroblock partition modes, multiple reference frames and the use of rate-distortion optimised mode decision [58].

Comprehensive performance evaluations of H.264/AVC can be found in [3] and [5].

7.2.1 Intra vs. Inter Prediction

In this first experiment, the compression performance and the computational complexity of inter predictive and intra predictive coding of H.264/AVC are investigated. The coding options available for inter and intra prediction coding in H.264/AVC are described in Chapter 3.

The Foreman sequence (QCIF, 300 frames @ 30 frames per second) is encoded using the above default parameters to obtain six inter coded sequences at constant QP values of 12, 18, 24, 30, 36 and 42. Note that for inter coded slices the JM7.4 encoder considers both intra modes and inter modes to encode each macroblock (as the standard allows). Therefore the set of encoded sequences are named as "Inter & Intra". A set of "Intra Only" versions of the same Foreman sequence is obtained by encoding the video sequence (at same QP values as above) with all the inter macroblock partition modes disabled, and thereby, forcing the encoder to encode the whole sequence using intra modes only. The bit rate and the average PSNR of the luminance component of each version of the sequence (at each constant QP level) is recorded along with the time taken to encode.

Figure 7.1 plots the rate-distortion performance of inter ("Intra & Intra") coding alongside the rate-distortion performance of "Intra only" coding. These plots suggest that inter prediction achieves more than 50% increase in compression efficiency compared with "Intra only" prediction. For example, the dotted lines show the approximate resultant bit

rates (calculated using linear approximation between adjacent points) of inter and intra coding at a luminance PSNR of 40dB. The bit rate of "Intra only" coding is nearly 1275 kbps whereas bit rate of "Inter & Intra" is 350 kbps, resulting in a 72.5% increase in compression efficiency. Note that the improvement in compression efficiency is calculated as the percentage decrease in bit rate at the same level of quality (luminance PSNR).

The computational complexity is measured as the encoding time spent by the encoder. The encoder spends on average 5.71 seconds to encode the sequence as intra only, while inter coding requires on average 50.08 seconds. In this case the inter prediction process is approximately 8 times more computationally complex than intra prediction.

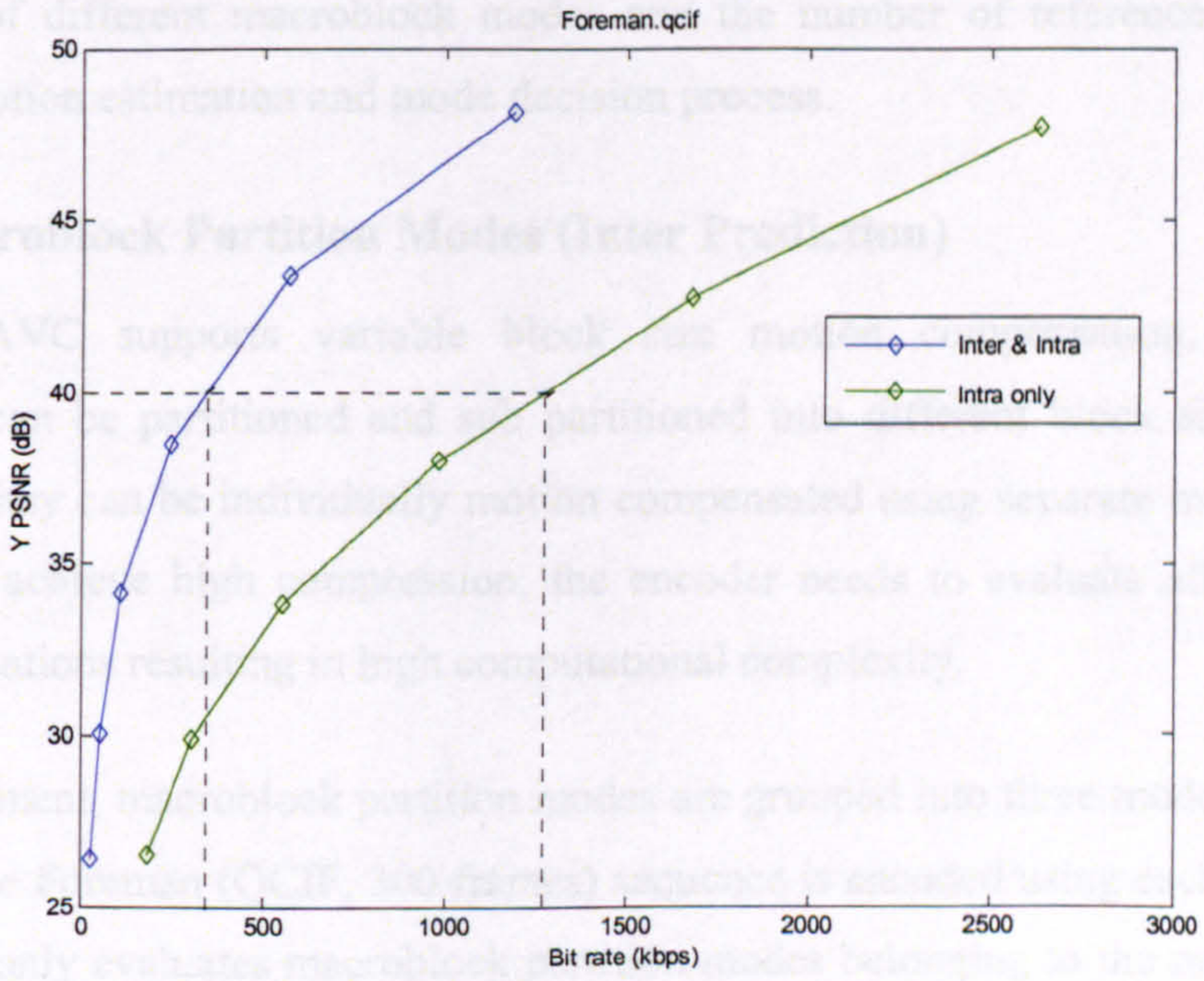


Figure 7.1: Comparison of rate-distortion performance between "Intra only" coding and "Inter & Intra" coding for "foreman" sequence.

7.2.2 Motion Estimation and Mode Decision

The H.264/AVC standard does not specify any motion estimation method to obtain the best prediction or any strategies for selecting best macroblock coding modes. These remain the responsibility of the encoder implementation. However, motion estimation and mode decisions play a major role in achieving high compression efficiency.

Software profiling is carried out for the JM7.4 reference software using the Microsoft Visual C++ profiling tool for function timing, which can be used to measure the time spent on each function during the execution of a software program. The first 100 frames of the Foreman (QCIF) sequence are encoded at QP = 24 with the default configuration. Results show that motion estimation and mode decision functions take up more than 65% of the total processing time of the encoding process. The rest of the processes such as transform and quantisation, entropy coding and other encoder management overhead take up only 35% of the total processor time.

This indicates that the encoder spends a large proportion of available processing resources on the motion estimation and mode decision process. The following experiments provide more insight into the benefits and associated computational complexity of different macroblock modes and the number of reference frames used during the motion estimation and mode decision process.

7.2.3 Macroblock Partition Modes (Inter Prediction)

The H.264/AVC supports variable block size motion compensation, where each macroblock can be partitioned and sub partitioned into different block sizes (partition modes) and they can be individually motion compensated using separate motion vectors. However, to achieve high compression, the encoder needs to evaluate all the possible block combinations resulting in high computational complexity.

In this experiment, macroblock partition modes are grouped into three mode groups as in Table 7.2. The Foreman (QCIF, 300 frames) sequence is encoded using each mode group (the encoder only evaluates macroblock partition modes belonging to the mode group) at six constant QP levels (12, 18, 24, 30, 36 and 42).

Table 7.2: Macroblock partition mode groups.

Mode Group	Partition/Sub partition block sizes
1	16x16
2	16x16, 16x8, 8x16, 8x8
3	16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4

Figure 7.2 shows the rate-distortion performance of the three macroblock partition mode groups. The dotted lines point to the approximate bit rates of each mode group at a constant PSNR of 40dB, with the bit rates calculated using linear approximation (see Table 7.3).

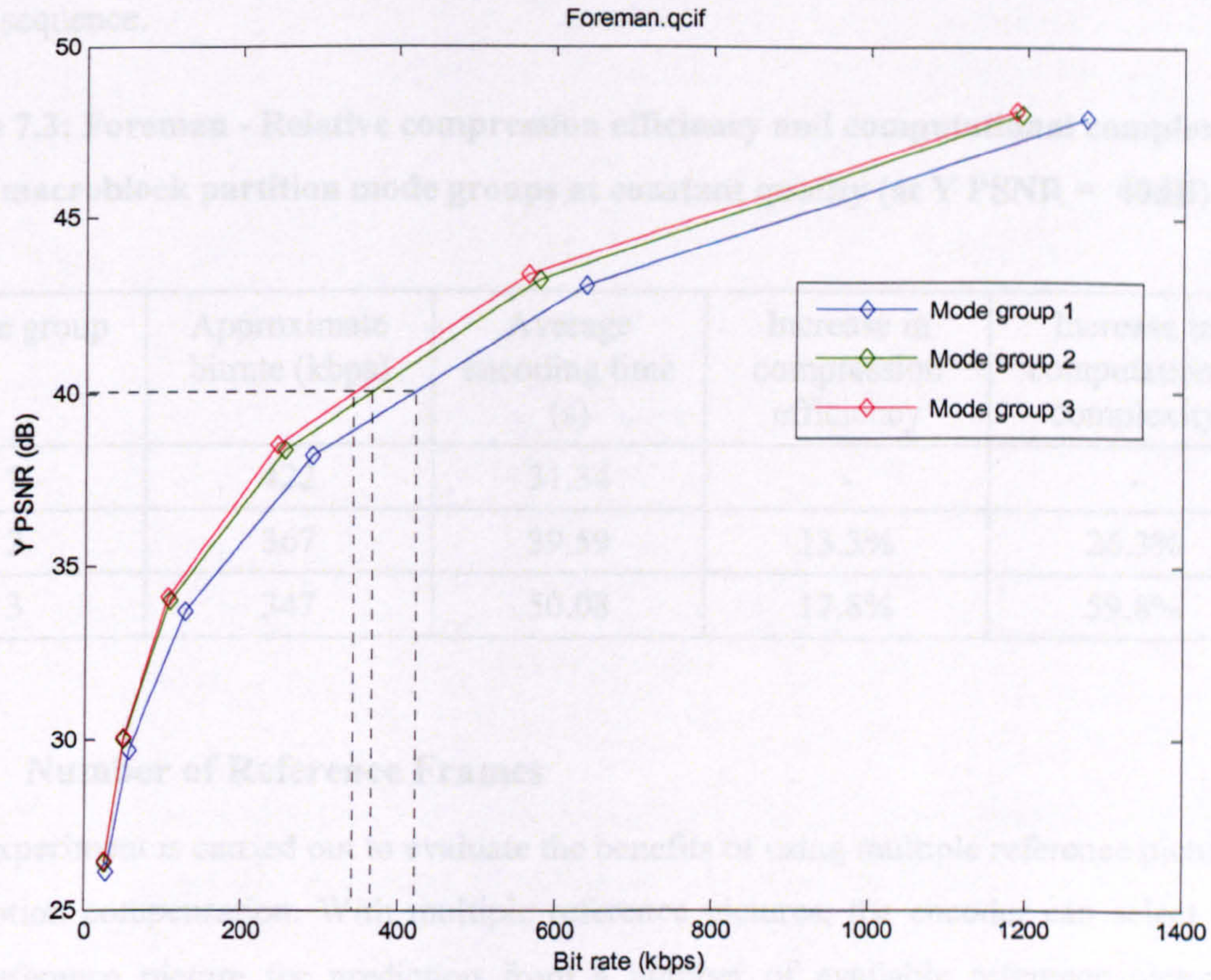


Figure 7.2: Rate-distortion performance of different macroblock partition mode groups

The increase in compression efficiency is calculated as the percentage bit rate saving relative to mode group 1 (using only 16x16 partition mode). Similarly, the increase in computational complexity due to evaluation of additional modes is calculated as the percentage increase in encoding time relative to mode group 1.

According to the results, compression efficiency is improved as the number of macroblock modes evaluated is increased. Using partition modes from 16x16 down to 8x8 increases the compression efficiency by 13.3% relative to just using 16x16 mode. In this case however, going down to 4x4 level only increases the compression efficiency by

a further 4.5% while the increase in computational complexity is doubled. The selection of different partition modes typically depends on the amount of detail and activity present in a video sequence. For example, a better prediction signal may be obtained by using 4x4 sub partition sizes in scenes with high motion and detail. Therefore the relative rate-distortion increases contributed by different partition modes may depend on the type of video sequence.

Table 7.3: Foreman - Relative compression efficiency and computational complexity of macroblock partition mode groups at constant quality (at Y PSNR = 40dB)

Mode group	Approximate bitrate (kbps)	Average encoding time (s)	Increase in compression efficiency	Increase in computational complexity
1	422	31.34	-	-
2	367	39.59	13.3%	26.3%
3	347	50.08	17.8%	59.8%

7.2.4 Number of Reference Frames

This experiment is carried out to evaluate the benefits of using multiple reference pictures for motion compensation. With multiple reference pictures, the encoder can select the best reference picture for prediction from a number of available reference pictures. However, the computational complexity increases because motion estimation and mode decision should be carried out for each candidate reference picture in order to select the best prediction.

The Foreman sequence (QCIF, 300 frames) is encoded by allowing the encoder to use 5 reference pictures for inter prediction. The sequence is encoded using same constant QP values as before. The rate-distortion performance and computational complexity is compared with the Foreman sequence encoded with 1 reference picture in Figure 7.3. The increase in compression efficiency of using 5 reference pictures compared with using 1 reference picture is found to be 10.4% (at PSNR = 40dB). This increase is achieved at the cost of increasing the computational complexity by almost 5 times (total time required for 1 ref. picture encoding is 50 seconds while average time for 5 ref. pictures coding is 246 seconds).

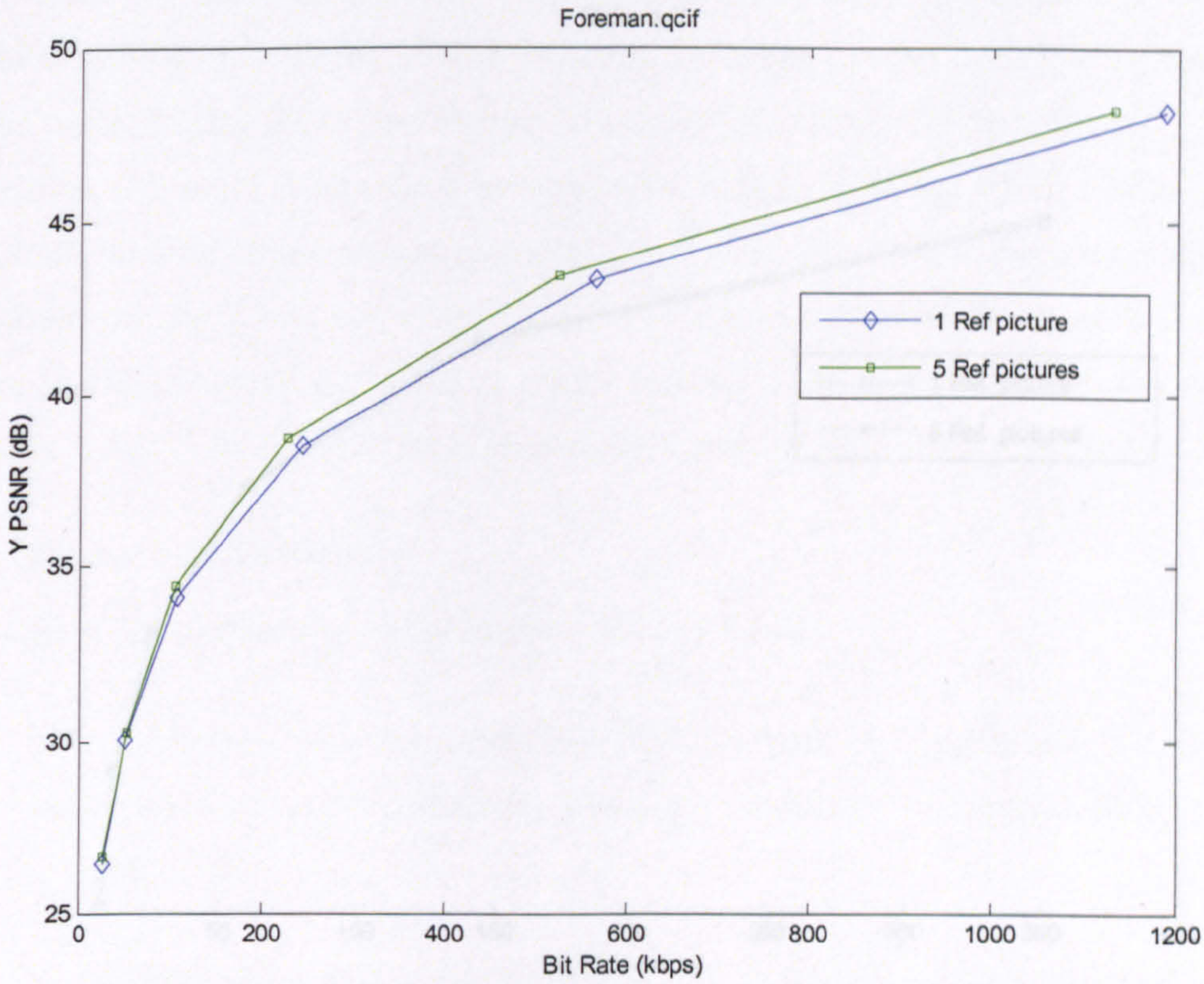


Figure 7.3: Foreman sequence - Comparison of rate-distortion performance between using 1 reference picture and 5 reference pictures

A similar experiment is carried out for the Claire sequence (QCIF, 494 frames). The performance between 1 reference frame and 5 reference frames is compared in Figure 7.4. In this example, the relative increase in compression efficiency of using 5 reference pictures is only 1.35% whilst computational complexity is increased by 5 times.

The above examples show that although the compression efficiency is increased by increasing the number of reference pictures, the benefit is sometimes limited when compared with the amount of computational complexity involved. However, the use of multiple reference frames is generally useful for video scenes containing oscillating or repetitive movements. For example, a better match for the current block may be available in a non-adjacent picture (a few pictures away from the current one) where the object or a part of the object contained in the current block reappears in nearly the same orientation and position. Therefore, a more significant increase in compression efficiency can be expected for these types of sequences.

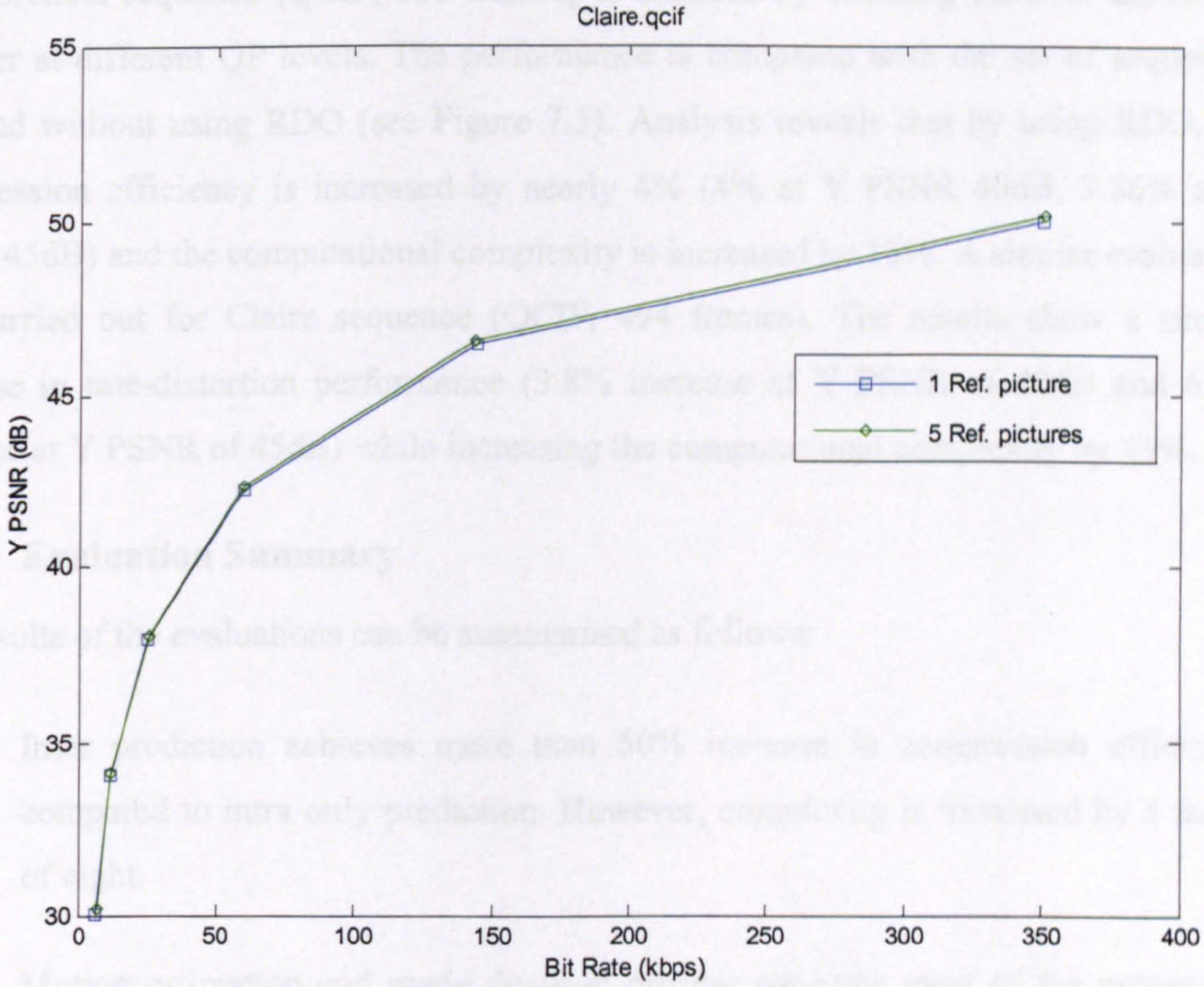


Figure 7.4: Claire sequence - Comparison of rate-distortion performance between using 1 reference picture and 5 reference pictures

7.2.5 Rate-distortion Optimisation

The JM7.4 reference software carries out rate-distortion optimised mode selection when Rate-distortion Optimisation (RDO) is enabled. The encoder selects the best macroblock mode by evaluating a Lagrangian cost function for each macroblock. This evaluation is carried out using the coded rate and distortion of each macroblock mode, which requires the macroblock to be encoded and decoded using all the possible modes before selecting the best mode and therefore, increasing the computational complexity.

This mode decision strategy does not form a part of the H.264/AVC standard. However, it was described in chapter 4 how rate-distortion optimisation can be used to achieve good rate-distortion performance in video coding. Furthermore, computational complexity issues associated with large number of possible encoding modes such as in H.264/AVC were also described. The following experiment evaluates the performance of RDO mode selection in the JM7.4 reference encoder.

The Foreman sequence (QCIF, 300 frames) is encoded by enabling RDO in the JM7.4 encoder at different QP levels. The performance is compared with the set of sequences encoded without using RDO (see Figure 7.5). Analysis reveals that by using RDO, the compression efficiency is increased by nearly 4% (4% at Y PSNR 40dB, 3.86% at Y PSNR 45dB) and the computational complexity is increased by 58%. A similar evaluation was carried out for Claire sequence (QCIF, 494 frames). The results show a similar increase in rate-distortion performance (3.8% increase at Y PSNR of 40dB and 6.7% increase at Y PSNR of 45dB) while increasing the computational complexity by 45%.

7.2.6 Evaluation Summary

The results of the evaluations can be summarised as follows:

- Inter prediction achieves more than 50% increase in compression efficiency compared to intra only prediction. However, complexity is increased by a factor of eight.

- Motion estimation and mode decision process consume most of the processing resources during encoding.

- Compression performance can be increased by using more macroblock partition modes. According to the results for the Foreman sequence, the relative increase in performance tends to be marginal when smaller block sizes are used. However, the computational complexity tends to increase with the increase in partition modes.

- Similarly, the performance can be increased by using more than one reference picture for prediction. However, the increase is sometimes hardly justifiable when compared to the increase in computational complexity.

- A rate-distortion optimised mode decision strategy increases the rate-distortion performance, however with a significant increase in computational complexity.

Note that the results may vary for different video sequences having different levels of motion and detail.

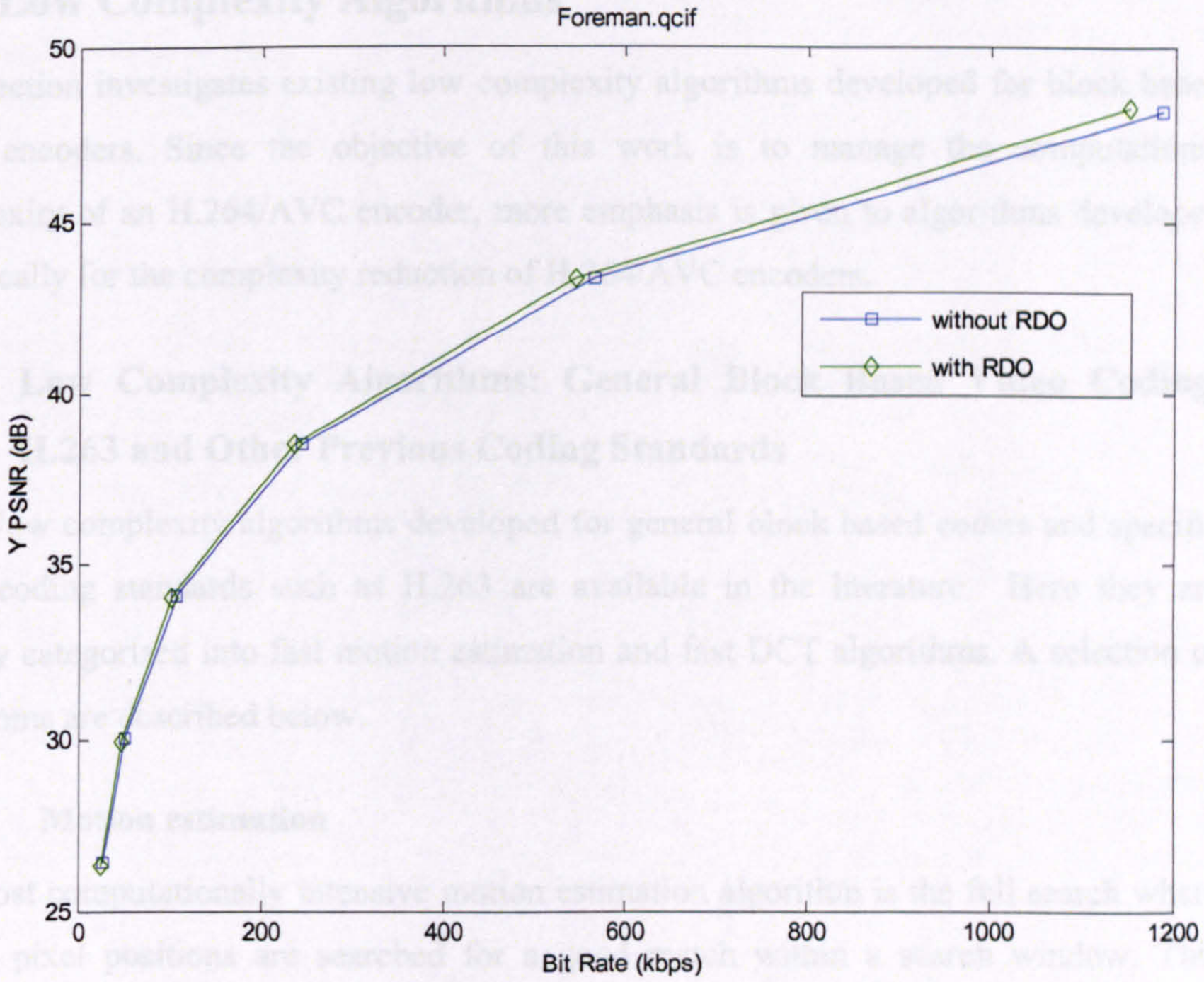


Figure 7.5: Foreman sequence – Rate-distortion performance, with and without rate-distortion optimisation

The H.264/AVC standard supports a number of coding modes and options. The coding modes and options individually provide varying amounts of compression gains depending on the encoded sequence. The significant gains in compression efficiency over previous coding standards are achieved by the combined effect of using various coding modes tools and options. However, the computational complexity is significantly increased by using many coding modes and options. Therefore, low-complexity algorithms have been developed to reduce the computational complexity of encoding by typically using only a subset of coding modes and options (or using more efficient methods of coding), whilst trying to maintain the compression efficiency of utilising the full set of modes and options.

7.3 Low Complexity Algorithms

This section investigates existing low complexity algorithms developed for block based video encoders. Since the objective of this work is to manage the computational complexity of an H.264/AVC encoder, more emphasis is given to algorithms developed specifically for the complexity reduction of H.264/AVC encoders.

7.3.1 Low Complexity Algorithms: General Block Based Video Coding, H.263 and Other Previous Coding Standards

Many low complexity algorithms developed for general block based coders and specific video coding standards such as H.263 are available in the literature. Here they are broadly categorised into fast motion estimation and fast DCT algorithms. A selection of algorithms are described below.

7.3.1.1 Motion estimation

The most computationally intensive motion estimation algorithm is the full search where all the pixel positions are searched for a good match within a search window. This method generally achieves the best rate-distortion performance since all the positions are evaluated. Fast motion estimation algorithms tend to reduce the number of positions searched for the best match while trying to maintain the same compression efficiency as full search. A number of fast motion estimation algorithms have been presented in the literature. Some of these algorithms can be found in [70]. This discussion will be limited to only one of the general algorithms. Fast motion estimation algorithms for H.264/AVC are discussed in section 7.3.2.1.

A block based fast motion estimation algorithm called the Nearest Neighbour Search (NNS) is described in [71]. The algorithm carries out a localised search from a search centre based on a motion vector prediction. The motion vector prediction is typically calculated as the median of the motion vectors of the neighbouring left, top and top right macroblocks. Each search layer carries out a diamond shape search at neighbouring pixel positions until the minimum SAD (or other best match criteria between the original and the predicted block) is in the middle of the diamond (see Figure 7.6). The algorithm also provides a flexible termination criterion based on Lagrangian complexity-distortion cost and a cost difference threshold in order to control the computational complexity by controlling the number of positions searched. It has been shown that this algorithm

matches the rate-distortion performance of a full search with a fraction of the computational resources.

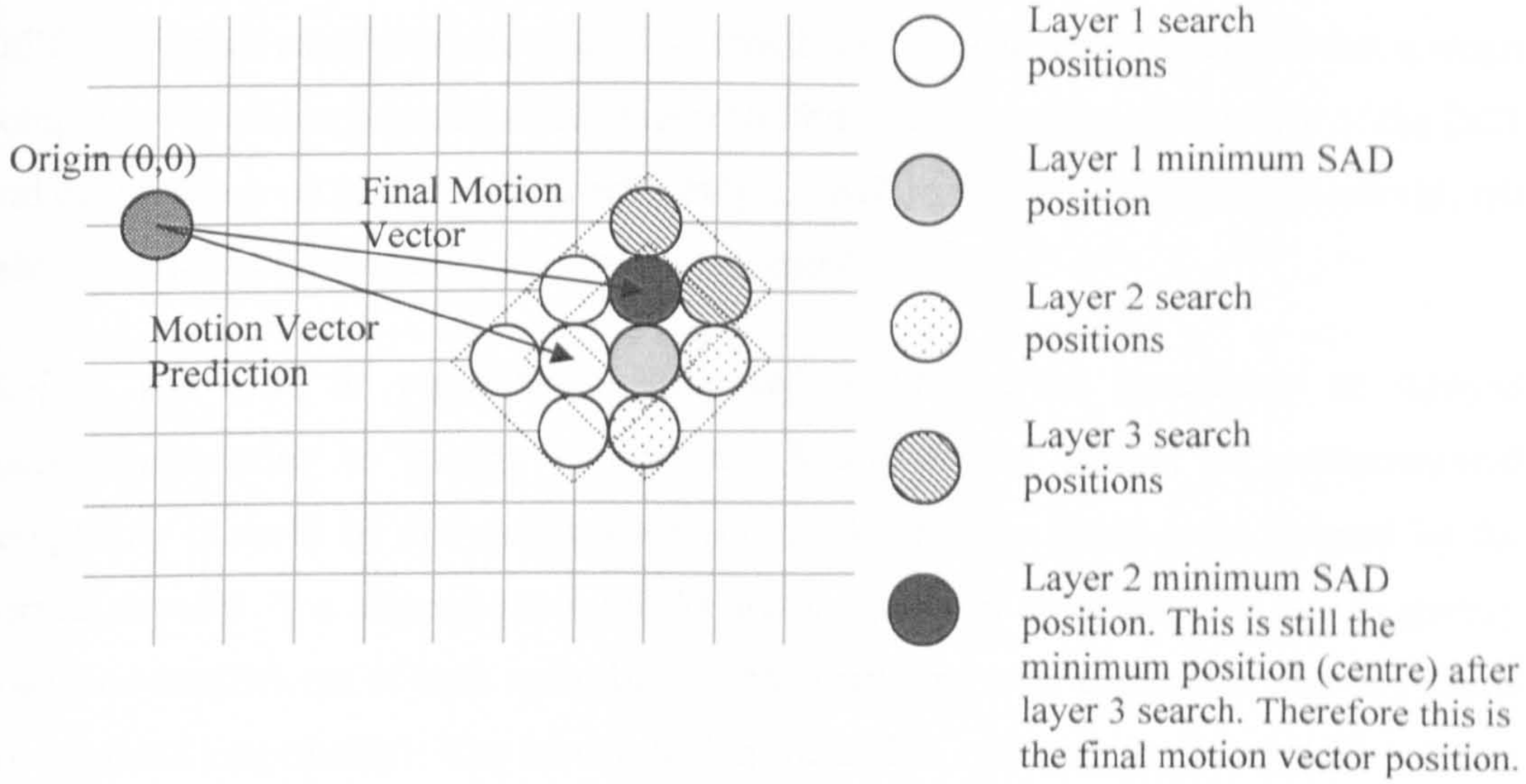


Figure 7.6: Nearest Neighbour Search (NNS) algorithm

7.3.1.2 DCT

In encoders such as H.263, the DCT is carried out on 8x8 blocks of samples. The DCT calculations typically require a significant amount of computational complexity compared with other encoding functions [72] [73]. Therefore, low complexity algorithms have been developed to reduce the computational complexity of the DCT calculations.

An adaptive algorithm was presented in [73] to reduce the complexity of DCT and quantisation process. The algorithm calculates the SAD value between the original block and the motion compensated prediction block. This is compared against threshold values in order to determine whether to skip the DCT and set all coefficients to zero, calculate only the DC component, calculate only low frequency components or to carry out the full DCT calculations. The thresholds are obtained using a statistical model for the DCT coefficients and the quantisation parameter. However, a single statistical model is used for all sequences resulting in variable performance for different types of sequences. The results showed a significant drop in computational complexity with little or no loss in video quality.

An algorithm to control the computational complexity of an H.263 encoder (or any block based encoder using DCT transform) by controlling the number of DCT calculations has been presented in [72, 74]. The SAD of each block (between the original and the motion compensated prediction) is compared with a threshold to determine whether to skip the DCT and set all coefficients to zero. The threshold is updated adaptively so that a target complexity level can be maintained. The algorithm can reduce the complexity of the DCT and quantisation up to 70% - 90% with only a small loss in image quality. However, the rate-distortion performance is not clearly presented.

In [75] and [76], an algorithm is proposed to predict the occurrence of skipped macroblocks prior to motion estimation. A significant amount of computational complexity is saved by not coding the macroblocks that are likely to be skipped by the normal encoder. The skipped macroblocks are predicted by estimating the low frequency quantised coefficients of each macroblock and comparing with thresholds (the thresholds are obtained empirically). The results (using the H.263 reference encoder [69]) indicate that significant computational savings can be achieved with only a small loss in rate-distortion performance. This algorithm is explained in chapter 8 of this thesis.

7.3.2 Low Complexity Algorithms: H.264/AVC

This section reviews some of the low complexity algorithms designed specifically for H.264/AVC standard encoders. In previous coding standards such as H.263 the processing resources have been more or less equally distributed among motion estimation/compensation, transform/quantisation and other encoding functions [72]. However, in H.264/AVC the DCT transform is replaced by an efficient integer transform [6] and due to the large number of encoding modes and options, the motion estimation and mode selection process tends to consume most of the processing resources during encoding. Therefore the main research efforts have been concentrated on low complexity implementations of the motion estimation and mode selection process.

7.3.2.1 Motion estimation

A fast motion estimation algorithm for H.264/AVC was initially proposed in [77] and has been later incorporated into the H.264/AVC JM reference encoder [58]. The Hybrid Unsymmetrical-cross Multi-Hexagon-grid Search (UMHexagonS) algorithm (as incorporated into JM reference software) estimates an initial motion vector prediction

using information such as motion vectors of neighbouring macroblocks (median), motion vectors of larger partition modes than the current mode (up-layer) and the motion vector of the current partition using previous reference frame (neighbouring reference frame). Using the best motion vector prediction as the centre, the integer pixel motion search is carried out in various shapes such as an unsymmetrical cross, local square full search, hexagonal grid and a hexagon based search patterns to avoid getting trapped in local minimum points. Once the best integer position is obtained, the sub-pixel motion estimation is carried out in a diamond based pattern around a sub pixel motion vector prediction which is obtained from the initial motion vector prediction. The results presented in [77] indicate that the algorithm achieves from 41% to 67% total encoding time saving compared with the full search while keeping the same rate-distortion performance. Further modifications to this algorithm, such as using simple motion vector prediction (using only the spatial median and up-layer prediction) and replacing the local square full search with a hexagonal search have been proposed in [78]. The presented results indicate a further 46% to 57% increase in computational savings compared to [77], with improved rate-distortion performance.

A fast algorithm for motion estimation using multiple reference pictures (long term motion estimation) has been presented in [79]. Complexity savings are achieved by adaptive motion search window location based on neighbouring motion vector information and motion estimation of lower resolution pictures with temporal and spatial search range reduction. The authors report significant complexity reductions compared to full search. However, no comparison is made with other fast motion estimation algorithms.

7.3.2.2 Mode decision

Fast macroblock mode selection algorithms have been proposed and incorporated into the H.264/AVC reference test model software (JM – Joint Model software) [58].

A fast algorithm for rate-distortion optimised intra mode decision in an H.264/AVC encoder was proposed in [80] and presented in [81]. This algorithm has been incorporated into the H.264 JM reference software [58]. The low complexity intra mode decision is based on local edge directional information where only the most probable intra modes are evaluated according to an edge direction histogram of the macroblock (in intra prediction directions).

An algorithm for fast inter mode decision is proposed in [82] [83]. The algorithm detects homogeneous and stationary regions to determine the inter prediction block size. For example, larger block sizes are selected if the region is more homogeneous. The homogeneous regions are detected using edge detection carried out in fast intra mode decision algorithm in [80] (discussed above). Stationary regions are detected using the SAD between the current and co-located macroblock in the reference picture. For example, if the 16x16 mode has zero motion and the macroblock is stationary (SAD < Threshold), inter 16x16 is chosen without evaluating other modes. This fast inter mode decision algorithm was also incorporated into the mode decision process of the JM reference software [58]. Two further modifications to the fast mode decision in JM reference software were introduced in [84]. First, an early skip mode decision was introduced for macroblocks in P-slices, where the motion estimation of the 16x16 partition is carried out first. If the transform coefficients are all zero and the motion vectors are equal to motion vector prediction, skip mode is chosen without evaluating any other mode. Second, the selective intra mode decision carries out intra mode prediction of a macroblock in a P-slice only if the spatial correlation between the current macroblock and the neighboring (already decoded) pixels is higher than the temporal correlation of the current macroblock with the pixels in the reference picture. The hypothesis is that if the current macroblock has a larger correlation in the spatial direction, it has a higher probability of being coded as an intra predicted macroblock. The experimental results indicate an average of 30% reduction in computational complexity with only a small drop in rate-distortion performance. The early skip decision was extended to B-slices in [85]. In a similar way to P-slices, the direct mode is evaluated first in B-slice macroblocks. The mode is selected as B-Skip if all the coefficients are zero, without evaluating any other mode. The results indicated an average reduction of 66% in encoding time for sequences coded in main profile with B-Slices.

Some of the other fast mode decision algorithms proposed for H.264/AVC encoder are as follows.

In [86], the authors present two algorithms for fast rate-distortion optimised mode selection of an H.264/AVC encoder. First, a number of candidate modes are selected (without evaluating all the modes) by performing non-optimised mode selection (low complexity mode decision described in [58], also see section 9.3.1) and choosing the

modes with lowest mode costs. The second algorithm carries out rate-distortion optimised mode selection using an estimated rate rather than calculating the actual rate for the selected modes. The presented results indicate only a small loss in PSNR with a significant drop in computational complexity. However, the changes to the rate and therefore the actual rate-distortion performance are not clearly shown. The algorithm uses a fixed number of candidate modes without considering sequence statistics which can lead to a loss in performance for some sequences. A similar algorithm is described in [87], where the number of candidate modes for rate-distortion optimisation is reduced by calculating a simple RD cost similar to the low-complexity mode selection described in JVT-K049 [58]. Although the authors claim that the number of candidates is selected according to the QP, the method of selection has not been presented.

A fast block size selection algorithm for inter frame coding was presented in [88]. The algorithm calculates the spatial complexity of a macroblock by estimating the energy of the transformed AC coefficients and selects the block sizes accordingly. An early skip decision is made if the SAD between the current and co-located macroblock in the reference picture is below a threshold. However, the decision making process is largely heuristic and the development of the algorithm is not supported by any analytical or experimental evidence. In [89], the original picture and the reference picture are down-sampled and pre-encoded to find the best prediction modes. The best prediction modes of the down-sampled picture are used to select subsets of candidate modes for inter prediction of the original picture. The authors of [90] present a hierarchical approach for fast mode decision using thresholds and cost comparison of already calculated modes (a largely heuristic approach).

A fast mode decision for intra only encoding based on the macroblock correlation between current and previous pictures has been presented in [91]. The results depend on the selection of a threshold. However, the threshold is selected using trial and error methods.

A skip mode selection algorithm for H.264/AVC has been proposed in [92] for rate-distortion optimised mode selection of an H.264/AVC encoder (JM9.0 [93]). This skip selection is carried out using Bayesian decision criterion by modelling the Probability Density Functions (PDF) of rate-distortion cost difference between "coding" and "skipping" a macroblock and the probability of the macroblock being coded or skipped.

The PDFs are modeled based on the QP and sequence activity. The concept of measuring and using sequence activity levels was introduced during this work [7] and are explained in Chapter 10. The results shown indicate good rate-distortion performance results with significant time savings[5].

7.4 Summary

Low-complexity algorithms have been developed to reduce the computational complexity of encoding functions such as motion estimation, mode selection and transform & quantisation. The hypothesis behind these algorithms is that the computational complexity of video compression can be reduced without significantly reducing video compression efficiency.

In H.264/AVC standard encoders, the most computationally intensive process tends to be motion estimation and mode selection and therefore the main focus of recent research has been to reduce the complexity of this process. In general, the following characteristics are common to most of the algorithms discussed in the previous sections.

1. The achieved complexity reductions and rate-distortion performance typically depend on the activity of the video sequence.

2. The low complexity algorithms are mostly threshold based. Many algorithms derive these thresholds empirically using trial and error methods.

3. Video sequence statistics are not typically taken into account for modelling and threshold calculations, leading to performance degradation for sequences with different activity levels.

4. The algorithms tend to be mostly heuristic and lack an underlying theoretical basis for the decision making process.

---

[5] Scalable complexity reduction is not carried out in this method. However, scalability can be achieved by shifting the decision threshold away from equal probability points with a similar effect to introducing a Lagrangian complexity cost in the complexity control algorithm described later in chapter 10.

5. No significant attempt has been made to control the computational complexity and to manage the complexity of a real-time encoder in order to maximize the performance with limited processing resources.

The algorithms developed during this project are based on the hypothesis that the computational complexity of the encoding process can be reduced by using efficient coding methods and/or utilising only a subset of coding modes and options whilst maintaining the original rate-distortion performance. However, the scope of this project is not limited to reducing the computational complexity of an H.264/AVC encoder. The objective is to manage the computational complexity of a real-time H.264/AVC encoder in order to produce maximum rate vs. video quality performance with limited available processing resources.

In the next chapter, an existing low-complexity algorithm is integrated into an H.264/AVC encoder. The algorithm skips the motion estimation and mode decision process (and subsequent processing stages) for some macroblocks and therefore eliminates the computational cost of processing these macroblocks. The coding time of the encoder is reduced without significantly reducing rate-distortion performance.

8 Macroblock Skip-mode Prediction

8.1 Introduction

This chapter presents a complexity reduction algorithm for an H.264/AVC encoder. The algorithm predicts macroblocks that are likely to be skipped after processing, i.e. macroblocks coded as P-Skip in H.264/AVC baseline profile. The skipped macroblocks are predicted prior to motion estimation by calculating the residual energy of the macroblock, therefore saving the computational effort of carrying out motion estimation/compensation and further processing stages for these macroblocks. The results indicate that the skip-mode prediction algorithm can achieve a significant reduction in computational complexity with only a small drop in rate-distortion performance.

8.2 Hypothesis

The variety of coding tools available in the H.264/AVC standard contribute to increased rate-distortion performance, however with increased computational complexity compared with previous coding standards. For example, the H.264 standard supports variable block size motion compensation, multiple reference frames for motion compensation prediction, quarter pixel accurate motion vectors and motion compensation over picture boundaries to improve the accuracy of the motion estimation process resulting in higher compression efficiency. The performance evaluation of some of the above coding tools revealed that these compression gains are achieved with a significant increase in computational complexity. In particular, the analysis shows that motion estimation and mode decision process consumes more than 65% of the processing resources during encoding, thus making it the most computationally expensive process in an H.264 encoder (section 7.2.2). Therefore, any complexity reduction algorithm proposed for an H.264 encoder should benefit from concentrating on reducing the complexity of the motion estimation and mode decision process rather than subsequent processing stages.

Low complexity or variable complexity algorithms have been developed for the main coding stages in a video codec and are implemented in standards-based encoders such as H.263 and H.264/AVC encoders. Some of these algorithms were reviewed in chapter 7. In [75] and [76], the authors propose an algorithm for managing the computational

complexity of an H.263 encoder by predicting and not processing macroblocks that are likely to be skipped after the motion estimation process. The algorithm estimates the energy of low frequency quantised coefficients in the residual macroblock (compared to the same position in reference picture, i.e. without motion estimation) to classify it as "skipped" or "not skipped". If the macroblock is classified as "skipped", the algorithm indicates the presence of a skipped macroblock in the bit stream and no further encoding is carried out. If the macroblock is classified as "not skipped", the usual encoding functions are carried out for the macroblock. The results presented show a significant complexity reduction without any significant loss in compression efficiency.

The rationale behind the skip prediction algorithm in [75] [76] is,

- The encoder normally skips a significant proportion of macroblocks, especially for low activity sequences, during encoding. The authors have shown that an H.263 encoder typically skips from 13% to 76% of the macroblocks depending on the sequence type and quantisation used. Therefore, a significant proportion of computational complexity can be saved by identifying and avoiding the processing of these macroblocks before the motion estimation process.

- By accurately predicting skipped macroblocks (macroblocks that the encoder would choose to skip after motion estimation process) it is possible to avoid degradation in rate-distortion performance.

Figure 8.1 shows a simplified block diagram of a macroblock skip prediction algorithm. If a macroblock is predicted as "skip", further processing of the macroblock is skipped and the presence of a skip macroblock is signalled to the bit stream. Otherwise, the macroblock is coded as normal. The advantage of the macroblock skip prediction algorithm is that all the computational effort required to carryout motion estimation and further processing stages are saved for the macroblocks that are predicted as skip. This advantage makes the macroblock skip-prediction an attractive technique for reducing the computational complexity of an H.264/AVC encoder because the algorithm can be applied before the motion estimation process. However, significant complexity savings can only be achieved if a significant number of macroblocks are skipped by an H.264 encoder after evaluating all the possible modes during motion compensation.

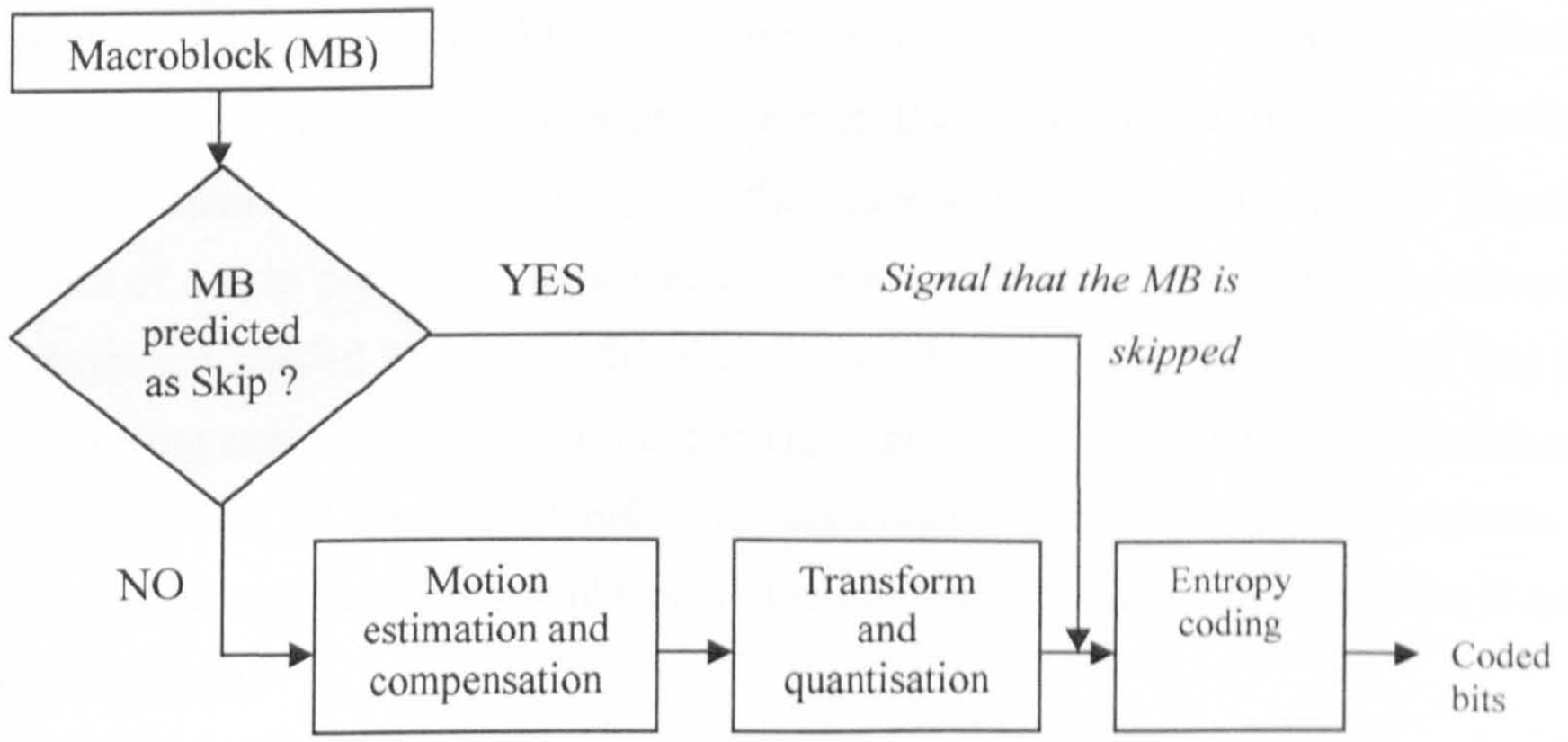


Figure 8.1: Macroblock skip-mode prediction

Experiments were carried out to measure the number of macroblocks skipped by the H.264/AVC reference encoder JM7.3 for different video sequences. The encoding parameters are set to the default configuration used for the baseline encoder evaluation in chapter 7 (Baseline profile, one slice per picture, IPPP, RDO disabled, one reference frame). These parameters are chosen to reflect likely choices for an H.264 implementation on a power- and/or computation-constrained device. The results are shown in Table 8.1. A significant amount of macroblocks are skipped during encoding, especially in lower activity sequences such as "Claire".

Table 8.1: Macroblock skip percentages

Test sequence	QP	PSNR-Y(dB)	MB skip %
Carphone (QCIF, 250 frames)	28	36.30	30.94%
	32	33.26	44.07%
Foreman (QCIF, 300 frames)	28	35.32	24.04%
	32	32.54	35.82%
Claire (QCIF, 494 frames)	28	39.43	78.63%
	32	36.40	84.96%

Therefore it can be hypothesized that macroblock skip-prediction can be used to bypass the processing of these macroblocks and hence significantly reduce the computational complexity of an H.264/AVC encoder. However, the rate-distortion performance of the algorithm depends on the accuracy of the skip-prediction algorithm. The possible outcomes of a skip prediction algorithm are shown in Table 8.2. It shows that accurate predictions are needed to achieve the objective of the algorithm. It is assumed that the pre-processing cost of the prediction algorithm is negligible compared to the complexity of the processing of the macroblock. The algorithm should minimise the chances of predicting a macroblock that should be coded as "skip" because this will incur a rate-distortion penalty.

Table 8.2: Possible prediction outcomes of the skip-prediction algorithm

Normal encoding result	Prediction	Rate-distortion performance	Computational complexity
Skip	Skip	unchanged	Reduced
Code	Code	unchanged	Slightly increased due to pre-processing of the prediction algorithm
Skip	Code	Unchanged, because the macroblock is skipped after normal processing	Slightly increased due to pre-processing of the prediction algorithm
Code	Skip	Reduced	Reduced

8.3 Skip-mode Prediction

This section explains the skip-prediction algorithm [75] [76] in detail and its implementation in the H.264/AVC reference encoder JM7.3. This work has been presented in [94].

8.3.1 Skip Macroblock Selection in JM Encoder (JM7.3)

The H.264/AVC JM encoder carries out a mode decision process to select the best macroblock coding mode and identifies certain macroblocks as skipped during encoding. When Rate-Distortion Optimised (RDO) mode selection is enabled, the skip mode is evaluated together with other possible coding modes as described in chapter 4 by

encoding the macroblock using each mode (including skip mode) and selecting the mode which minimizes the Lagrangian rate-distortion cost.

When RDO mode selection is disabled, skip mode is selected for a macroblock if the following conditions are met:

1. The chosen macroblock partition size is 16x16 (i.e., a single motion vector for the macroblock).

2. The X and Y components of the MVD (motion vector difference) are zero, relative to the skip mode motion vector prediction. Note that the skip mode motion vector prediction is slightly different from the median prediction [2] (refer section 3.6.6).

3. The chosen prediction reference is the previous frame in display order.

4. All coefficients are zero after transform and quantisation.

The macroblock partition sizes (or intra modes) are chosen during motion estimation and mode decision according to the following cost minimisation [58]:

$$\min\{SA(T)D0 + SA(T)D\} \tag{8-1}$$

The flow chart for motion estimation for each macroblock partition mode is shown in Figure 8.2.

The parameter $SA(T)D0$ is a weighting factor based on the quantisation parameter, macroblock mode and (for inter predicted macroblocks) an estimation of the number of bits required to encode the motion vectors and reference frame(s). $SA(T)D$ is the block difference (or the residual energy) between the original block and the motion compensated block using the motion vector. The (T) in $SA(T)D$ refers to optionally using the Hadamard transform to calculate the block difference. When the Hadamard transform is not used, the $SA(T)D$ is calculated as the SAD between the original and the motion compensated block. If the Hadamard transform is used, the pixel differences are transformed (using a 4x4 Hadamard for each 4x4 block within each block) and sum of the absolute values are taken.

For each partition mode, motion estimation is carried out by evaluating and selecting the motion vectors that minimises equation 8-1. The mode which results in the lowest '*min*{*SA*(*T*)*D0* + *SA*(*T*)*D*}' cost is selected as the best mode.

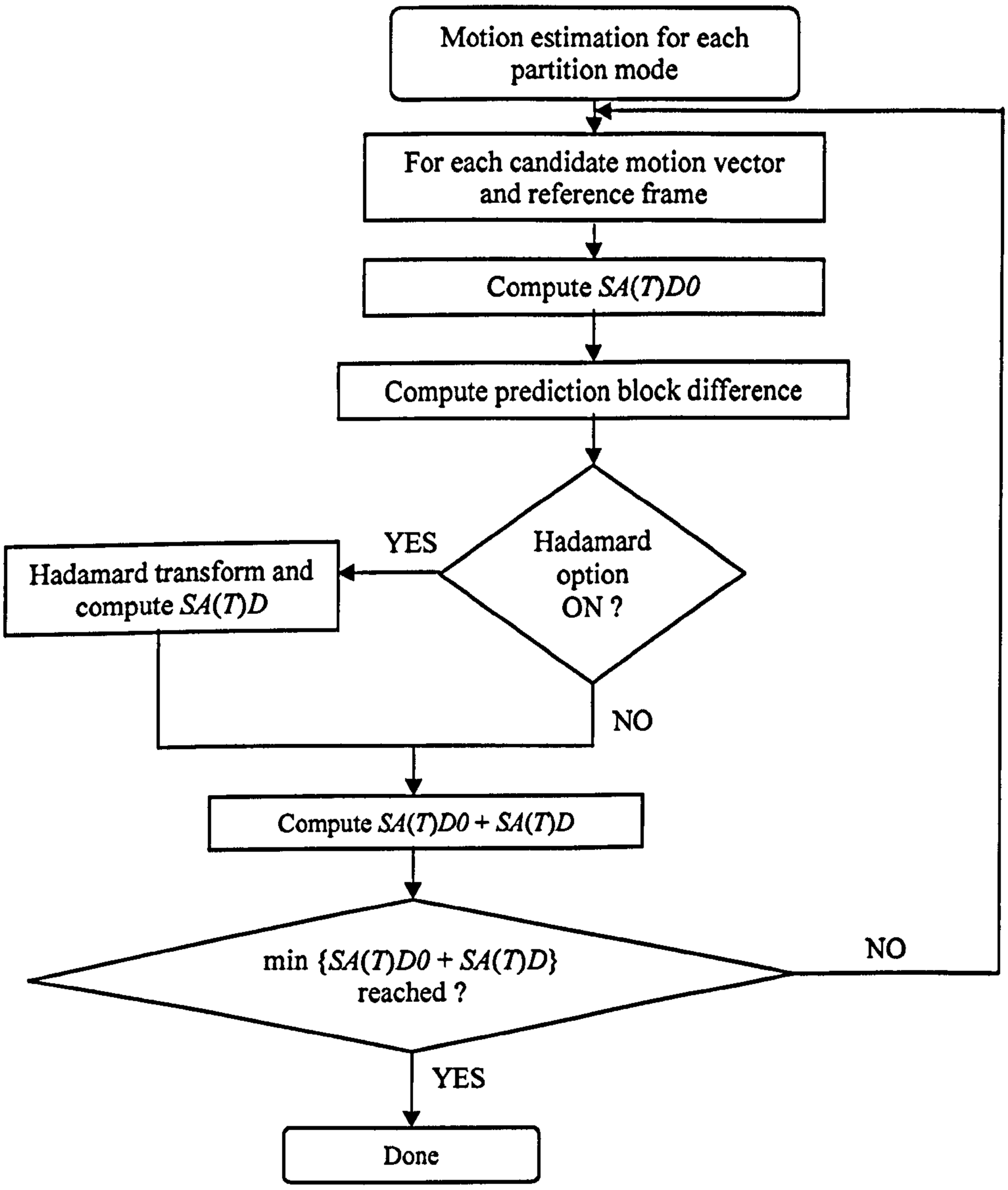


Figure 8.2: Flow chart for motion estimation

The weighting factor $SA(T)D0$ is biased to favour the skip mode when the partition size is 16x16 and the motion vector difference is zero. The encoder starts the motion estimation process using the motion vector prediction as the base. Therefore, zero motion vector differences mean that motion vectors are equal to the motion vector prediction. If a macroblock is skipped by the encoder, the decoder reconstructs the skipped MB by

motion-compensated prediction from the current reference picture, using a motion vector predicted from previously-decoded motion vectors.

8.3.2 Skip-prediction Algorithm

8.3.2.1 Basic prediction

Skipped macroblocks tend to have low residual energy at zero motion vector position (SAD_{00}). In [75], the authors have shown that SAD_{00} correlates well with the probability of a macroblock being skipped and that this relationship depends on the quantisation step size. The skip prediction algorithm forms a basic prediction using the high correlation between the quantised residual energy of a macroblock at zero motion vector and the probability of that macroblock being skipped. In [75], the residual energy is normalised by the quantisation step size in order to obtain similar relationships for all quantisation step sizes.

Experiments were carried out by encoding a video sequence (Foreman - QCIF, 300 frames) at different QP levels using JM7.3 (default configuration, without using Hadamard transform) and recording the residual energy $SA(T)D_{00}$ of each macroblock for 16x16 mode with zero motion vector difference (motion vector is the motion vector predictor). The selected coding mode of each macroblock was also recorded.

Figure 8.3 shows probability of macroblocks being skipped (y-axis) against residual energy $SA(T)D_{00}$ normalised by the quantisation step size Qstep (x-axis). The two plots show the Foreman sequences encoded at QP=26 and 32, using the H.264 reference software JM7.3. The dotted region plots each macroblock which has a particular $SA(T)D_{00}$/Qstep against the probability of the macroblock being skipped. The solid line represents the Cumulative Density function (CDF) of the macroblocks and indicates the distribution of macroblocks having various $SA(T)D_{00}$/Qstep values. The CDF values are normalised to one (equals to 100%) so that the values correspond to the y-axis.

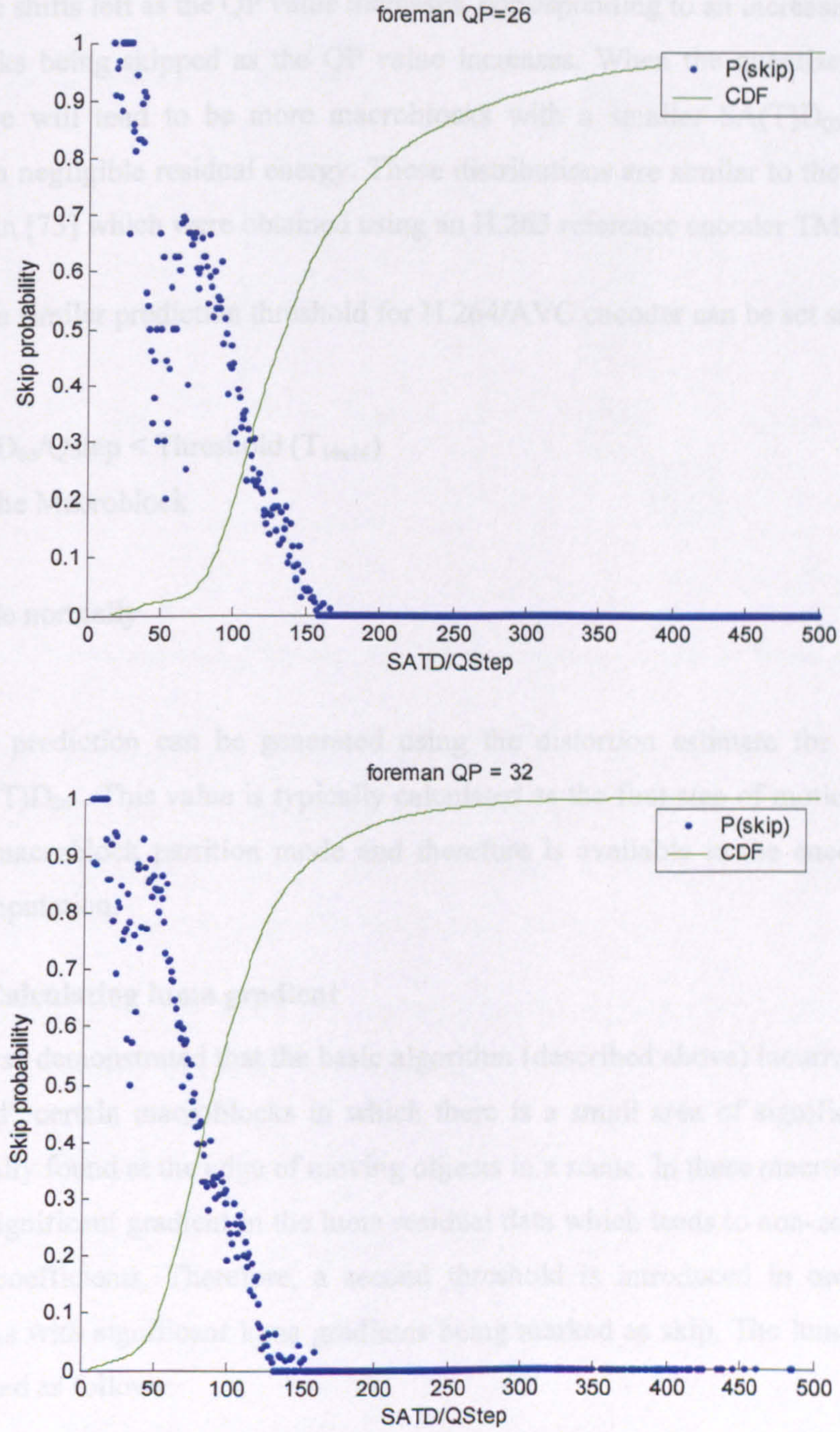


Figure 8.3: Distribution of skipped macroblocks

The plots indicate that low $SA(T)D_{00}$/Qstep values correlates well with high skip probability 'P(skip)' indicating that skipped macroblocks have low residual energy. The skip-probability distributions of normalised residual energy at both QP=26 and 32 show similar behaviour with clear cut off points (zero probability) at approximately 150. The

CDF curve shifts left as the QP value increases, corresponding to an increasing number of macroblocks being skipped as the QP value increases. When the quantiser step size is large, there will tend to be more macroblocks with a smaller $SA(T)D_{00}/Qstep$ value resulting in negligible residual energy. These distributions are similar to the distributions presented in [75] which were obtained using an H.263 reference encoder TMN10 [47].

Therefore a similar prediction threshold for H.264/AVC encoder can be set such that;

If $SA(T)D_{00}/Qstep < Threshold\ (T_{16x16})$

Skip the Macroblock

Else

Encode normally

This basic prediction can be generated using the distortion estimate for zero motion vector $SA(T)D_{00}$. This value is typically calculated as the first step of motion estimation of 16x16 macroblock partition mode and therefore is available to the encoder without further computation.

8.3.2.2 Calculating luma gradient

In [75] it was demonstrated that the basic algorithm (described above) incorrectly predicts as "skipped" certain macroblocks in which there is a small area of significant residual data, typically found at the edge of moving objects in a scene. In these macroblocks, there is often a significant gradient in the luma residual data which leads to non-zero quantised transform coefficients. Therefore, a second threshold is introduced in order to avoid macroblocks with significant luma gradients being marked as skip. The luma gradient is approximated as follows:

The macroblock is partitioned into four 8x8 blocks and each 8x8 block is further divided into four 4x4 blocks as shown in Figure 8.4.

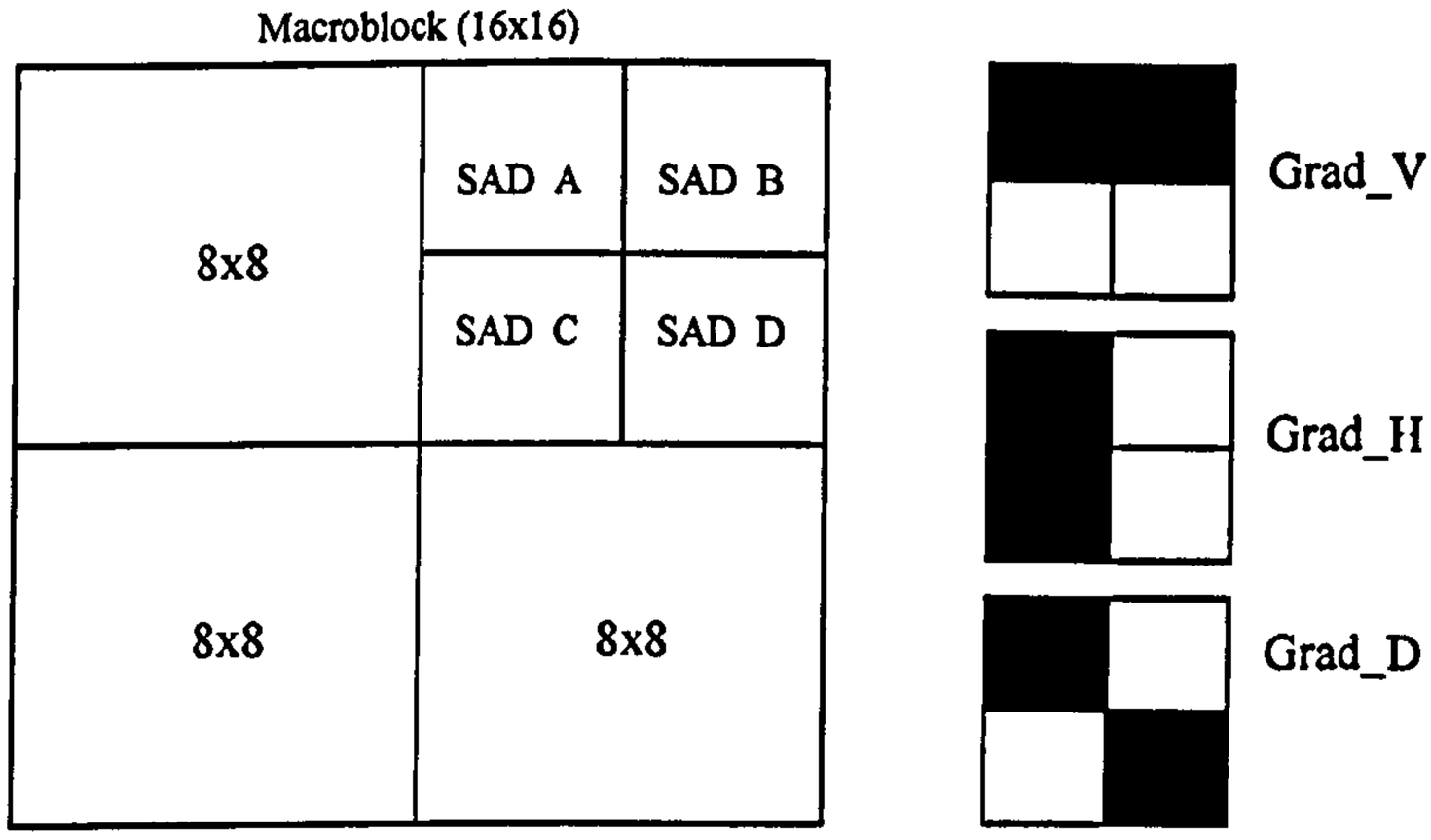


Figure 8.4: Macroblock partitioning for the approximation of luma gradients

The luma gradient within each 8x8 luma block is approximated as follows. First, the SAD value of each 4x4 luma block within the 8x8 block is calculated. These are marked in the figure as SAD_A, SAD_B, SAD_C and SAD_D which correspond to the top left, top right, bottom left and bottom right 4x4 blocks respectively.

Then, the absolute vertical, horizontal and diagonal luma gradients (denoted by Grad_V, Grad_H and Grad_D respectively) are calculated as follows:

$$Grad\_V = abs(SAD\_A + SAD\_B - SAD\_C - SAD\_D) \quad (8\text{-}2)$$

$$Grad\_H = abs(SAD\_A - SAD\_B + SAD\_C - SAD\_D) \quad (8\text{-}3)$$

$$Grad\_D = abs(SAD\_A - SAD\_B - SAD\_C + SAD\_D) \quad (8\text{-}4)$$

The maximum luma gradient within an 8x8 block is calculated as:

$$Grad\_8x8\_max = max(Grad\_H, Grad\_V, Grad\_D) \quad (8\text{-}5)$$

These calculations are carried out for all four 8x8 blocks of the macroblock. The maximum luma gradient value out of all 8x8 blocks (Grad_MAX) is chosen as the maximum luma gradient present in the macroblock. Note that this method is used to roughly approximate the luma gradients present without using a significant amount of

computational complexity, since the computational complexity introduced by the skip-prediction algorithm should be minimised.

8.3.2.3 The complete algorithm

This algorithm adapted from [75, 76], predicts skip-mode macroblocks in an H.264 encoder by estimating the residual energy and the maximum luma gradient within a macroblock prior to coding. The algorithm is summarised as follows:

For each macroblock in the picture:

1. Calculate the predicted 16x16 motion vector for the macroblock.
2. Calculate the total luma SA(T)D for zero motion vector difference (this is $SA(T)D_{00}$).
3. Calculate the maximum luma gradient of the macroblock as described in section 8.3.2.2.
4. Make the skip prediction decision as follows.

If $\{SA(T)D_{00}/Qstep < Threshold\ (T_{16x16})\}$ AND $\{Grad\_MAX < T_{Grad}\}$

Skip the macroblock

Else

Encode normally

8.4 Results

Table 8.3 shows the performance of the skip mode prediction algorithm. It lists data for "Carphone" sequence (QCIF, 250 frames) encoded with QP = 28 and 32. In both cases T_{16x16} and T_{Grad} were selected so that the total number of macroblocks skipped (macroblocks predicted as skipped as well as the macroblocks skipped after encoding) are approximately the same as the number of macroblocks skipped during the original encoding process. The thresholds were set using a trial and error method, by encoding the sequence with different threshold levels and selecting the thresholds which satisfy the above criterion with a good rate-distortion performance. The complexity reduction was calculated using the actual coding time of the sequence. The table demonstrates that the

algorithms can significantly reduce computational complexity with a small drop in PSNR (0.14dB).

Table 8.3: Carphone -Performance of the skip-prediction algorithm

Algorithm	QP	PSNR-Y (dB)	Bit rate Kbps	Complexity reduction		PSNR drop
				Time	Skipped macroblocks	
Original	28	36.30	150.58			
Algorithm	28	36.16	149.26	**25%**	**27%**	0.14
Original	32	33.26	81.74			
Algorithm	32	33.12	80.66	**31%**	**33%**	0.14

Figure 8.5 shows the PSNR performance on a frame-by-frame basis for the algorithm with a complexity reduction of 25%, for the Carphone sequence encoded with a QP value of 28. The graph shows a small PSNR drop (average 0.14 dB) when the sequence is encoded with the macroblock skip prediction algorithm.

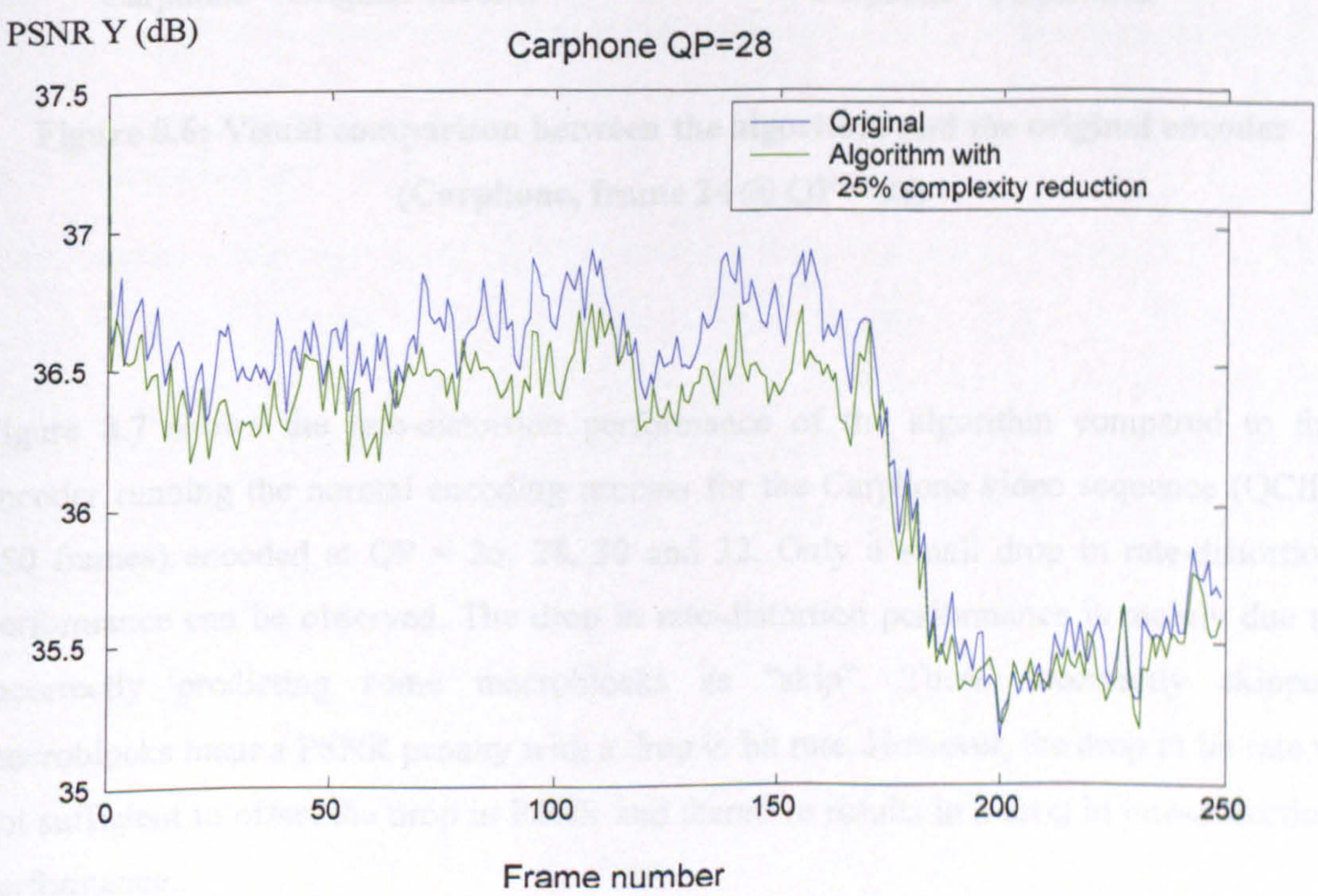


Figure 8.5: PSNR performance of the algorithm with 25% complexity reduction

This small PSNR drop could be attributed to the algorithm incorrectly skipping some macroblocks that should have been coded. For example, macroblocks containing edges of moving objects could produce high frequency AC coefficients (that are not detected by the algorithm) when matched with the reference macroblock with zero displacement relative to the prediction.

Figure 8.6 shows two sample frames from the Carphone video sequence (frame 24, QP = 28) coded using the original encoder and the reduced complexity encoder. No significant difference can be observed between the two coded video frames.

Carphone – Original encoder

Carphone – Algorithm

Figure 8.6: Visual comparison between the algorithm and the original encoder (Carphone, frame 24 @ QP = 28)

Figure 8.7 shows the rate-distortion performance of the algorithm compared to the encoder running the normal encoding process for the Carphone video sequence (QCIF, 250 frames) encoded at QP = 26, 28, 30 and 32. Only a small drop in rate-distortion performance can be observed. The drop in rate-distortion performance is mainly due to incorrectly predicting some macroblocks as "skip". These incorrectly skipped macroblocks incur a PSNR penalty with a drop in bit rate. However, the drop in bit rate is not sufficient to offset the drop in PSNR and therefore results in a drop in rate-distortion performance.

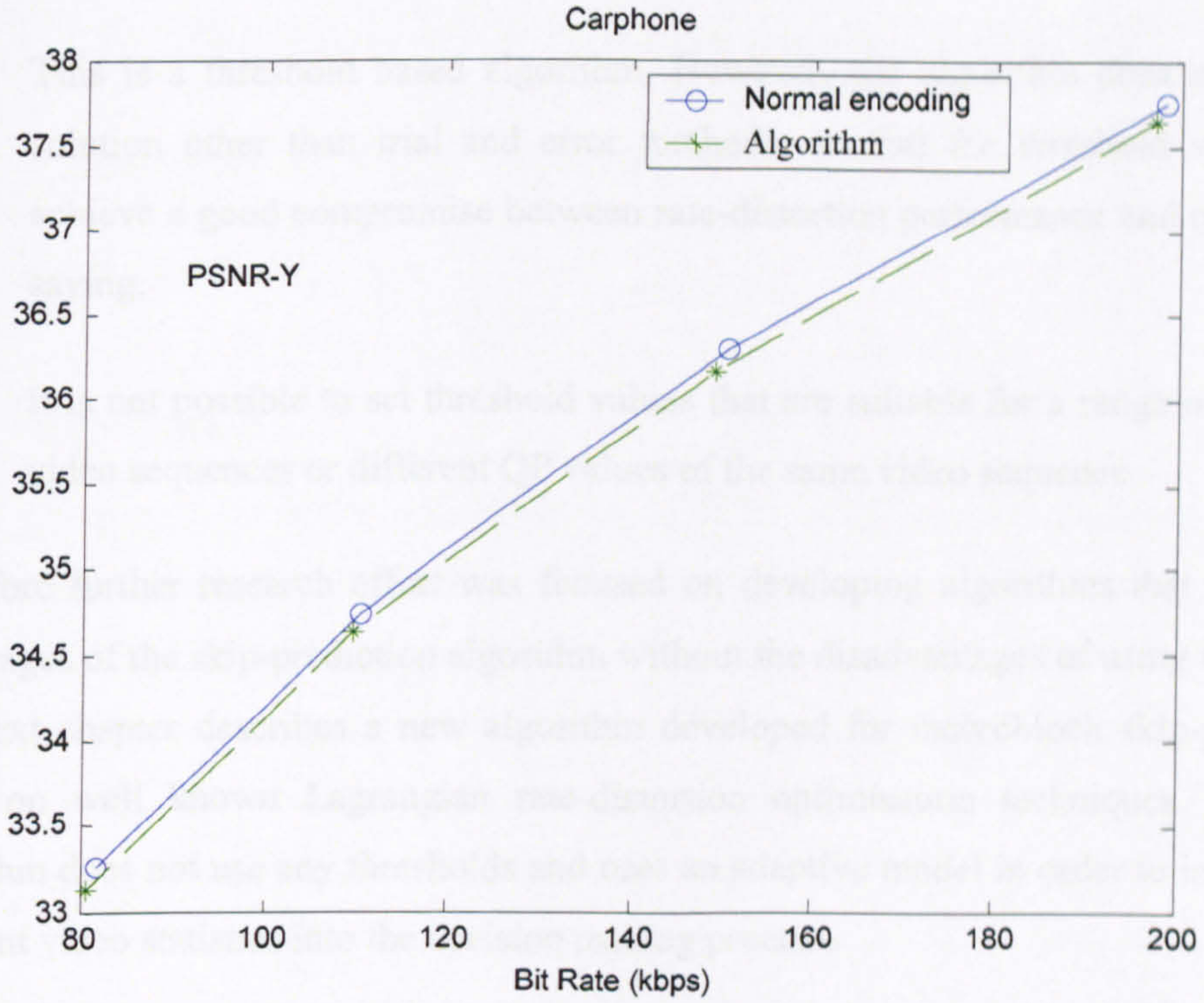


Figure 8.7: Rate-distortion performance of the complexity reduction algorithm

8.5 Conclusion

This chapter investigates a complexity reduction algorithm for an H.264/AVC encoder which is adapted from an existing algorithm [75, 76] developed for H.263. The algorithm predicts skipped macroblocks by calculating the residual energy before performing the motion estimation. Macroblocks predicted as "skipped" are not processed further, saving all further computation. The advantages and the disadvantages of this algorithm can be summarised as follows:

Advantages

- Significant reduction in computational complexity can be achieved by predicting skipped macroblocks before processing.
- Complexity savings can be achieved without a significant loss in rate-distortion performance by carefully selecting the threshold values.

Disadvantages

- This is a threshold based algorithm. However, the algorithm does not offer a solution other than trial and error methods, to find the threshold values that achieve a good compromise between rate-distortion performance and complexity saving.

- It is not possible to set threshold values that are suitable for a range of different video sequences or different QP values of the same video sequence.

Therefore further research effort was focused on developing algorithms that retain the advantages of the skip-prediction algorithm without the disadvantages of using thresholds. The next chapter describes a new algorithm developed for macroblock skip-prediction based on well known Lagrangian rate-distortion optimisation techniques. The new algorithm does not use any thresholds and uses an adaptive model in order to incorporate different video statistics into the decision making process.

9 Complexity Reduction of an H.264/AVC Encoder using Lagrange Cost Function Estimation

9.1 Introduction

A new complexity reduction algorithm for an H.264/AVC encoder is presented in this chapter. This is a skip-macroblock prediction algorithm where computational savings are achieved by identifying, prior to motion estimation, macroblocks that are likely to be skipped and hence saving further computational processing of these macroblocks. In this new algorithm, the early skip-macroblock prediction is made by estimating a Lagrangian rate-distortion cost function. The algorithm employs a model of the expected rate-distortion cost of skipping versus coding the macroblock to determine whether the macroblock should be coded or skipped. The Lagrange multiplier parameter is adaptively modeled in order to incorporate local sequence statistics into the decision making process. Simulation results demonstrate that the algorithm can achieve computational savings of 19-67% (depending on the source sequence) with no significant loss of rate-distortion performance, compared with a "baseline" non-optimised encoder. The development of this new algorithm has led to a journal publication [7] and a conference paper [8].

The Lagrangian optimisation theory and its applications to rate-distortion optimisation in video coding were explained in chapter 4 in detail. Section 9.2 briefly revisits some important concepts used in the Lagrangian rate-distortion optimised mode selection algorithm [51] employed by the H.264/AVC reference encoder JM7.3 (herein after referred to as the "JM Encoder"). The Lagrangian rate-distortion cost based low-complexity skip prediction algorithm is described in section 9.3 followed by results and conclusion.

9.2 Rate-distortion Optimised Mode Selection

The problem of rate-distortion optimised mode selection over a group of N macroblocks and the use of Lagrange optimisation methods to solve this problem are discussed in

chapter 4 in detail. Assuming macroblock independence (see section 4.3.3.5), the sub-optimal solution is given by:

$$\mathbf{M}^* = \sum_{i=1}^{N} \arg\min_{M_i} J(X_i) \tag{9-1}$$

Where now:

$$J(X_i) = D(X_i, M_i) + \lambda R(X_i, M_i) \tag{9-2}$$

The mode selection algorithm incorporated into the JM encoder [51] assumes independence between macroblocks and estimates a suitable value of λ for every macroblock as a function of the current QP. In the JM encoder, the macroblock mode is selected by first encoding the macroblock in every mode and then selecting the mode that has the minimum R(D) cost given the estimated Lagrange multiplier λ for the selected QP value. The modes include 13 intra modes, which use spatial prediction from previously-coded samples of the current frame, the inter mode with 259 possible combinations of motion compensation block sizes, motion compensated prediction from previously coded reference frame(s), and the skip mode in which no information (other than informing the decoder that the macroblock is being skipped) is transmitted for this macroblock.

Therefore, despite the simplifying assumption of macroblock independence and the estimation of λ, rather than iterative evaluation for different values of λ, the computational cost of the rate-distortion optimised (RDO) algorithm is high due to the requirement to encode each macroblock multiple times.

Equation (9-2) expresses the rate-distortion cost of coding a macroblock, and is the basis for the complexity reduction algorithm described in this chapter. The Lagrange multiplier estimation based approach used in the JM encoder is utilized in order to reduce the computational complexity. First, the Lagrange multiplier to be used is estimated as a function of the video sequence and the quantisation parameter and then, the estimated multiplier is used to trade off the rate-distortion cost of coding a macroblock with the rate-distortion cost of skipping the macroblock. In the following sections, the skip mode selection algorithm is described in detail and the results of applying this algorithm within the framework of the JM encoder are presented.

9.3 Low Complexity Skip Prediction

The skip-prediction algorithm reduces computational processing through early identification of macroblocks to be skipped. The skip prediction model aims to reduce computation by not processing a proportion of macroblocks whilst maintaining or improving rate-distortion performance. The new model is compared with the JM encoder operating with the following default parameters (hereinafter described as the "baseline encoder") similar to the encoder configuration used in chapter 8:

1. Baseline profile
2. One reference frame, one slice per frame, P-slices only (after an initial I-slice)
3. Rate-distortion optimised mode selection disabled.

These parameters are chosen to reflect likely choices for an H.264/AVC implementation on a power- and/or computation-constrained device.

9.3.1 Skip Prediction Model

Let M_i be the coding mode chosen by the baseline encoder for macroblock X_i and let $M_i = m_K$ represent the skip mode. The rate-distortion cost of coding an MB is given in (9-2) while the cost of skipping a macroblock is:

$$J(X_i, m_K) = D(X_i, m_K) \tag{9-3}$$

The rate term is ignored since the rate associated with a skipped macroblock is effectively zero.

Macroblock X_i should be skipped (that is, not coded) if

$$D(X_i, M_i) + \lambda R(X_i, M_i) \geq D(X_i, m_K) \tag{9-4}$$

This is the decision making criteria for skip-macroblock prediction.

9.3.2 Model Parameter Estimation (D, R)

In equation (9-4), $D(X_i, m_K)$ is the distortion between the current MB and the motion-compensated MB from the reference picture with zero displacement from the predicted vector MVP. The distortion is calculated as the MSE. This effectively is the actual distortion of the macroblock if the macroblock is skipped. $D(X_i, m_K)$ may be calculated prior to coding the current macroblock, i.e., its calculation does not depend on any outputs of the coding process.

$D(X_i, M_i)$ is the MSE between the current MB and the decoded, reconstructed MB while $R(X_i, M_i)$ is the number of bits required to code the current macroblock using coding mode $M_i \in \{m_1, ..., m_{K-1}\}$. The actual values of $D(X_i, M_i)$ and $R(X_i, M_i)$ are not available prior to coding and therefore these parameters for each macroblock in the current frame (n) are estimated using the following models:

$$\hat{D}^{(n)}(X_i^{(n)}, M_i^{(n)}) = \alpha_d D^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)}) \tag{9-5}$$

And

$$\hat{R}^{(n)}(X_i^{(n)}, M_i^{(n)}) = \alpha_r R^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)}) \tag{9-6}$$

Where $D^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)})$ is the MSE between the original and reconstructed macroblocks in the same position (co-located macroblock) in the previous frame (n-1) and $R^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)})$ is the number of bits required to code the MB in the same position in the previous frame. The error 'ε' between the cost of coding a macroblock in the current frame and the cost of coding using the estimated parameters can be calculated as,

$$\begin{aligned} \varepsilon = &\left(D^{(n)}(X_i^{(n)}, M_c^{(n)}) + \lambda R^{(n)}(X_i^{(n)}, M_c^{(n)})\right) - \\ &\left(\alpha_d D^{(n-1)}(X_i^{(n-1)}, M_c^{(n-1)}) + \alpha_r \lambda R^{(n-1)}(X_i^{(n-1)}, M_c^{(n-1)})\right) \end{aligned} \tag{9-7}$$

The values of α_d and α_r that minimises the estimation error, $(\alpha_d, \alpha_r)^*$, are chosen according to the following:

$$(\alpha_d, \alpha_r)^* = \arg\min_{\alpha_d, \alpha_r} E\{\varepsilon^2\} \quad (9\text{-}8)$$

To avoid re-introducing complexity into the coding process, the optimal choices for the parameters are estimated prior to coding, and average values are used. Twelve different video sequences are encoded at several QPs and the error $E\{\varepsilon^2\}$ values are calculated for α_d and α_r between 0.1 and 2.0 (in steps of 0.1). The Lagrange multiplier λ is calculated as described in section 9.3.3. The twelve sequences are chosen to reflect a wide range of video content and activity. The minimising values $(\alpha_d, \alpha_r)^*$ for each combination of sequence and QP are recorded and averaged (see Table 9.1).

The averages of the minimising values are found to be $\alpha_d \approx 1.2$ and $\alpha_r \approx 0.5$. Note that the variance between $R^{(n)}$ and $R^{(n-1)}$ is found to be significantly higher than the variance between $D^{(n)}$ and $D^{(n-1)}$, hence the minimum of $E\{\varepsilon^2\}$ occurs for a relatively low value of α_r. In order to achieve computational saving, the parameters α_d and α_r are not estimated for every MB; rather, fixed values fitting a wide range of sequences with varying degrees of motion and detail were selected. The fixed values of the parameters α_d and α_r are applied in the context of the algorithm described in section 9.3.4. The average values obtained ($\alpha_d \approx 1.2$ and $\alpha_r \approx 0.5$) minimize the estimation error. However, when applied to the skip prediction algorithm (see next section), the parameter value $\alpha_d = 1.2$ results in error propagation. This is because any error introduced by the skip algorithm increases with an estimation parameter which is larger than one. That is, if a macroblock is erroneously skipped, the probability of the co-located macroblock in the next frame being skipped is increased. Therefore, fixed parameters $\alpha_d = 1.0$ and $\alpha_r = 0.5$ were chosen to provide an acceptable trade-off between rate-distortion and computational performance, i.e. these values approximately satisfy equation (9-8) and give acceptable performance across a range of video sequences and QPs. These fixed parameters are chosen to provide computational savings with minimal loss of rate-distortion performance (see results in section 9.4).

Table 9.1: Minimising values of α_d and α_r for selected video sequences.

Sequence	QP	α_d^*	α_r^*
Foreman	25	1.3	0.6
	30	1.2	0.5
	33	1.1	0.6
Carphone	25	1.3	0.6
	30	1.2	0.5
	33	1.2	0.4
Claire	25	1.3	0.6
	30	1.3	0.3
	33	1.2	0.2
Mother & daughter	25	1.2	0.6
	30	1.2	0.4
	33	1.2	0.3
Akiyo	25	1.2	0.6
	30	1.1	0.4
	33	1.1	0.3
Coastguard	25	1.2	0.7
	30	1.1	0.5
	33	1.1	0.3
Container	25	1.2	0.4
	30	1.1	0.2
	33	1.1	0.1
Hall objects	25	1.1	0.7
	30	1.1	0.4
	33	1	0.4
News	25	1.2	0.6
	30	1.1	0.5
	33	1.1	0.4
Silent	25	1.1	0.8
	30	1.1	0.7
	33	1.1	0.6
Suzie	25	1.3	0.6
	30	1.2	0.5
	33	1.3	0.4
Table	25	1.1	0.8
	30	1.1	0.6
	33	1	0.6
Mean values		**1.16**	**0.49**

9.3.3 Model Parameter Estimation (λ)

For a non-increasing, convex rate-distortion function, the Lagrangian parameter λ is the (negative) derivative of distortion with respect to rate [95]. The algorithm is designed to

match the rate-distortion performance of the baseline encoder in order to achieve low-complexity. The Lagrange multiplier λ is estimated with the assumption that distortion is a differentiable function of rate at all points. Since λ is the negative of the slope of a *D(R)* curve at a given coding point it can be estimated as:

$$\lambda = -\frac{d\overline{D}}{d\overline{R}} \tag{9-9}$$

Where $\overline{D}$ and $\overline{R}$ are the mean distortion and rate, respectively, for a set of coded macroblocks. The distortion and rate for a given QP value are clearly dependent on macroblock statistics and the quality of motion-compensated prediction.

In general, the number of bits required to encode a macroblock tends to be inversely proportional to the macroblock distortion so that lower bit rates can be achieved at the expense of higher distortion and vice versa. However, the number of coded bits achieved for a particular level of distortion also depends on sequence statistics such as the degree of motion (activity) and image detail. The rate-distortion curves are modelled with a simple polynomial model $\overline{D} = \overline{F} \cdot \overline{R}^{-1}$, where the value of the parameter $\overline{F}$ is determined by the activity of the sequence.

Therefore, an activity factor F_i is defined for the i[th] macroblock as:

$$F_i = D(X_i, M_i) \cdot R(X_i, M_i) \tag{9-10}$$

Using the baseline encoder, a set of 50-frame video sequence segments were encoded at a range of QP values. The set of video sequence segments encompasses a wide variety of activity, from low to high motion and from low to high detail. Figure 9.1 plots average distortion $\overline{D}$ against average coded bits per macroblock $\overline{R}$ for three sequence segments, coded at selected QP values. The average activity in the sequence is calculated as $F = \frac{1}{N_{MB}} \sum_{i=1}^{N_{MB}} F_i$, where N_{MB} is the total number of macroblocks in the segment. The three curves represent different activity levels *F*, corresponding to the sequence segments, Foreman (frames 275-325) with F = 1191, Foreman (frames 100-150) with F=778 and Carphone (frames 100-150) with F=337. Figure 9.1 shows that higher sequence activity and higher quantisation parameter result in higher distortion for a fixed average bitrate.

As illustrated in Figure 9.1 by the coding points corresponding to QP=27, the $\overline{D}(\overline{R})$ curve and its slope at each QP is clearly dependent on both F and QP; therefore an estimate for λ was developed based on these two parameters.

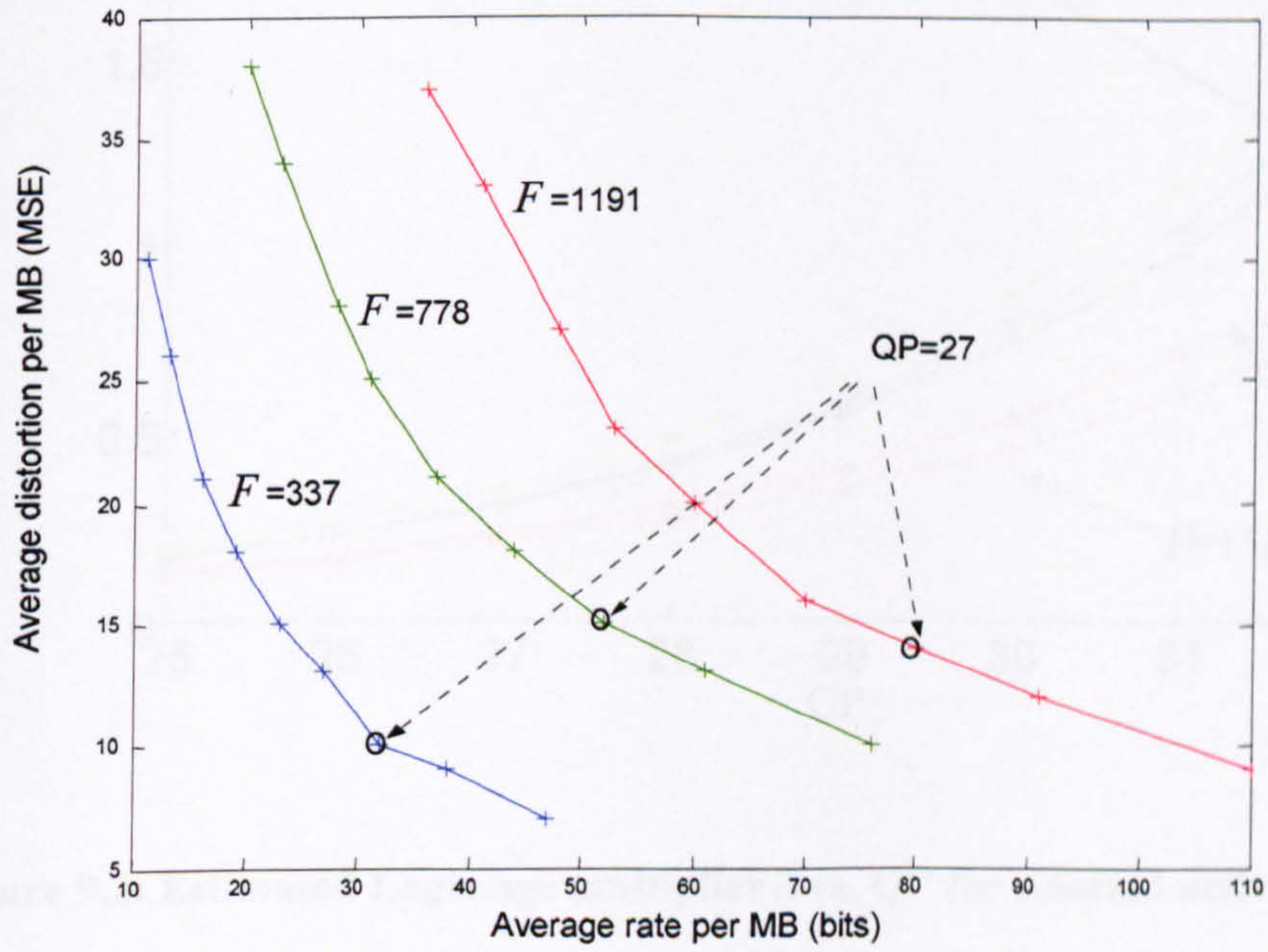


Figure 9.1: Average distortion vs. rate per MB for selected activity factors, *F*.

For a given activity factor F, λ is estimated as the central difference using neighbouring $D(R)$ points:

$$\lambda(QP) = -\frac{1}{2}\left(\frac{\overline{D}(QP)-\overline{D}(QP+1)}{\overline{R}(QP)-\overline{R}(QP+1)}+\frac{\overline{D}(QP-1)-\overline{D}(QP)}{\overline{R}(QP-1)-\overline{R}(QP)}\right) \tag{9-11}$$

The $\lambda(QP)$ calculated for different activity levels (using different sequence segments) is used to obtain an approximate relationship for $\lambda(QP)$ at each activity level. The curves for different activity levels are shown in Figure 9.2. The results show that the sequence activity has a more significant effect at high QP values.

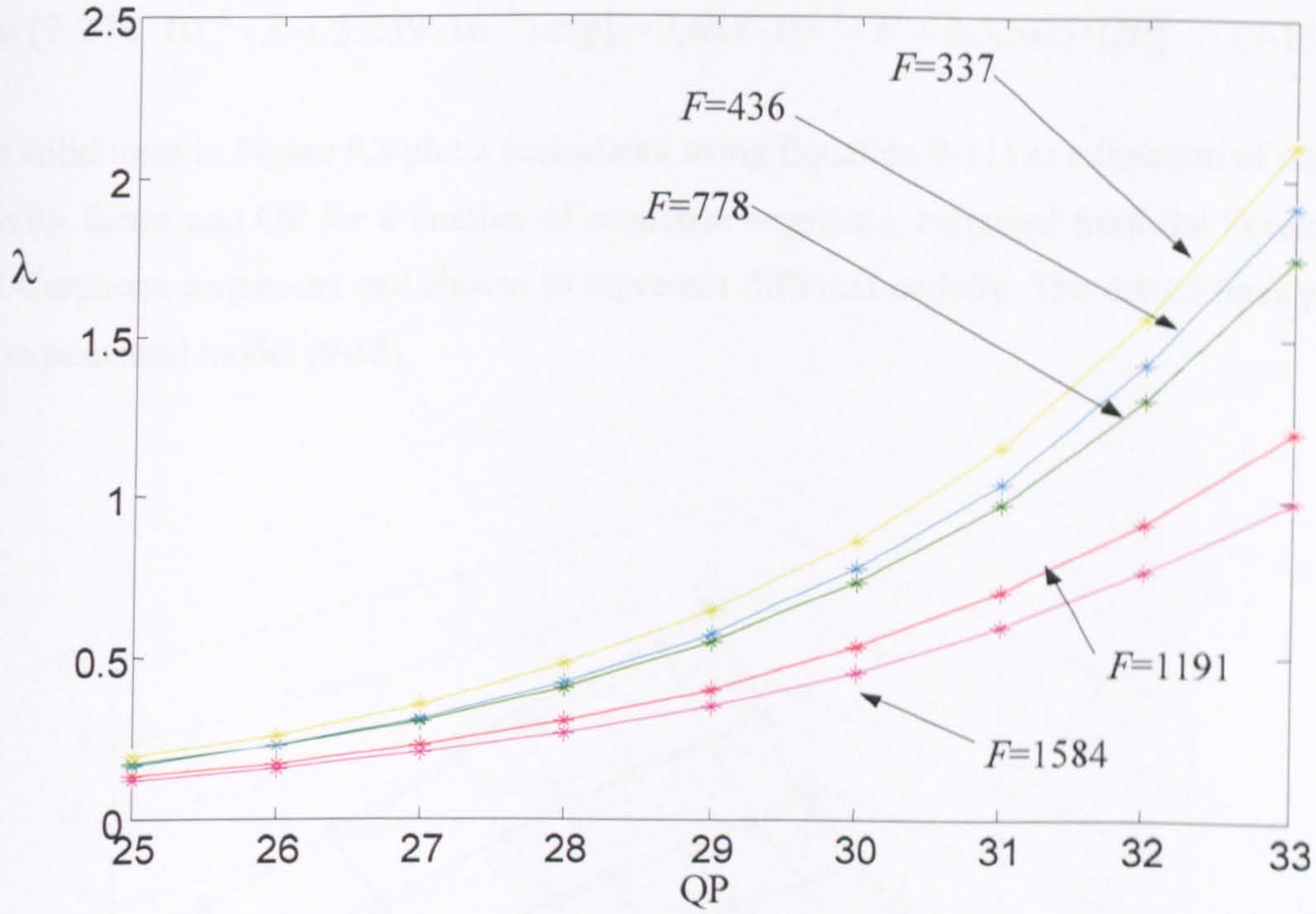


Figure 9.2: Estimated Lagrange multiplier λ vs. QP for selected activity levels, F.

These curves are obtained by fitting the estimated λ values (using equation 9-11) to an exponential model,

$$\lambda = A \cdot \exp(B \cdot QP) \tag{9-12}$$

Where the coefficients A and B depend on the average activity level of a sequence segment. By plotting the coefficient values A and B against corresponding activity factors and using linear regression, the following relationships between the coefficients A and B and the average activity factor F is obtained.

$$A = 7.374 \cdot 10^{-8} F + 5.239 \cdot 10^{-5} \tag{9-13}$$

and

$$B = -3.688 \cdot 10^{-5} F + 0.3203 \tag{9-14}$$

Therefore, λ can be estimated as a function of activity factor, F, and QP,

$$\hat{\lambda} = (7.374 \cdot 10^{-8} \cdot F + 5.239 \cdot 10^{-5}) \exp[(-3.688 \cdot 10^{-5} \cdot F + 0.3203) \cdot QP] \quad (9\text{-}15)$$

The solid lines in Figure 9.3 plot λ (calculated using Equation 9-11) as a function of mean activity factor and QP for a number of sequence segments, extracted from the Foreman and Carphone sequences and chosen to represent different activity. The dotted lines plot the exponential model (9-15).

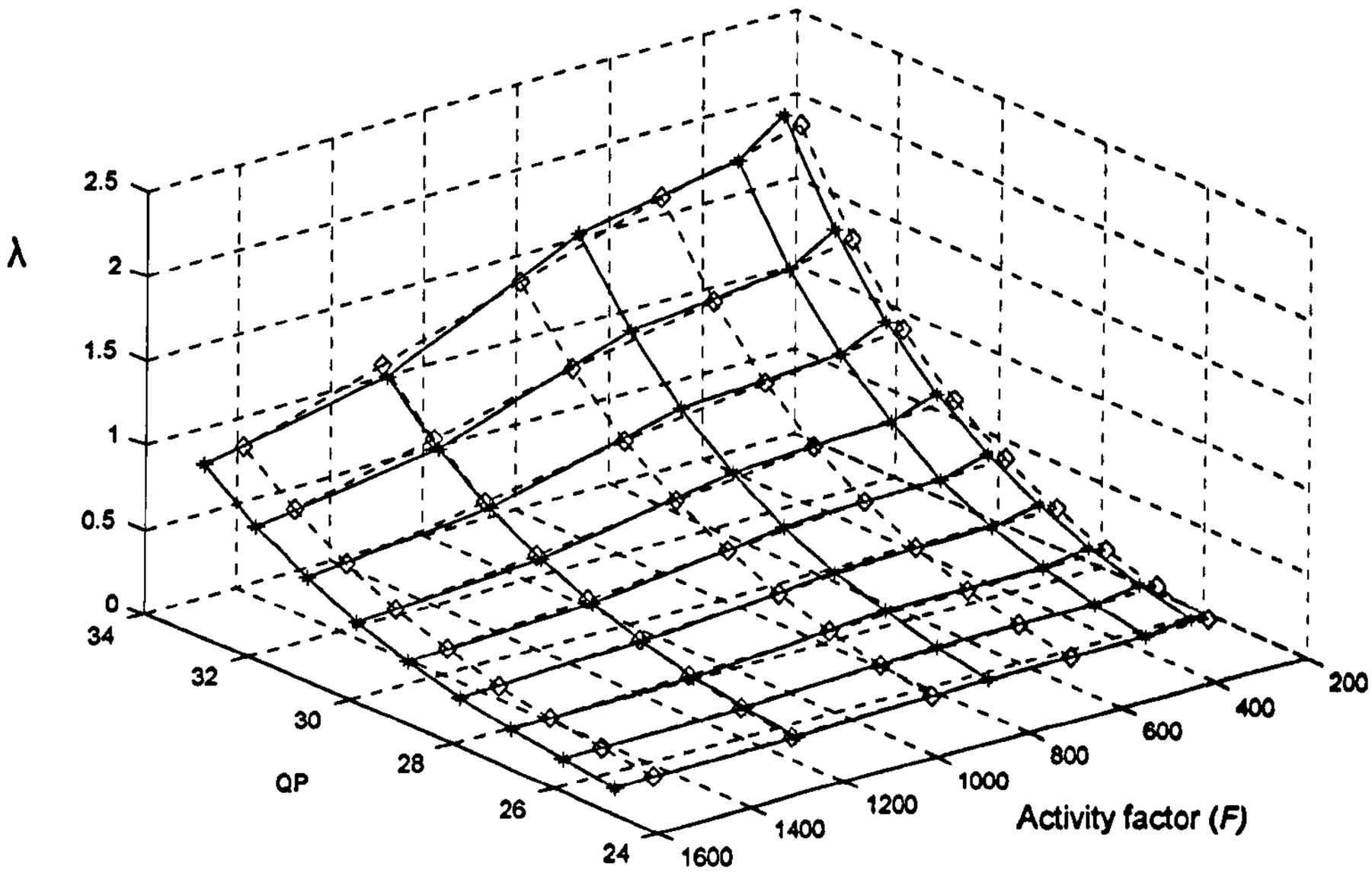


Figure 9.3: The parameter λ as a function of QP and F (solid lines: experimental results, dotted lines: model).

It is clear from Figure 9.3 that this model (equation 9-15) closely matches the measured performance across a variety of video sequence segments, from low activity to high activity, and a range of quantisation values. Figure 9.4 shows the error between the model (equation 9-15) and measured values for parameter λ, for a range of quantisation values and activity levels. The mean absolute error is found to be only 4.82%. The sequence activity factor and desired quantisation parameter are known and therefore an estimate of λ can be made prior to coding.

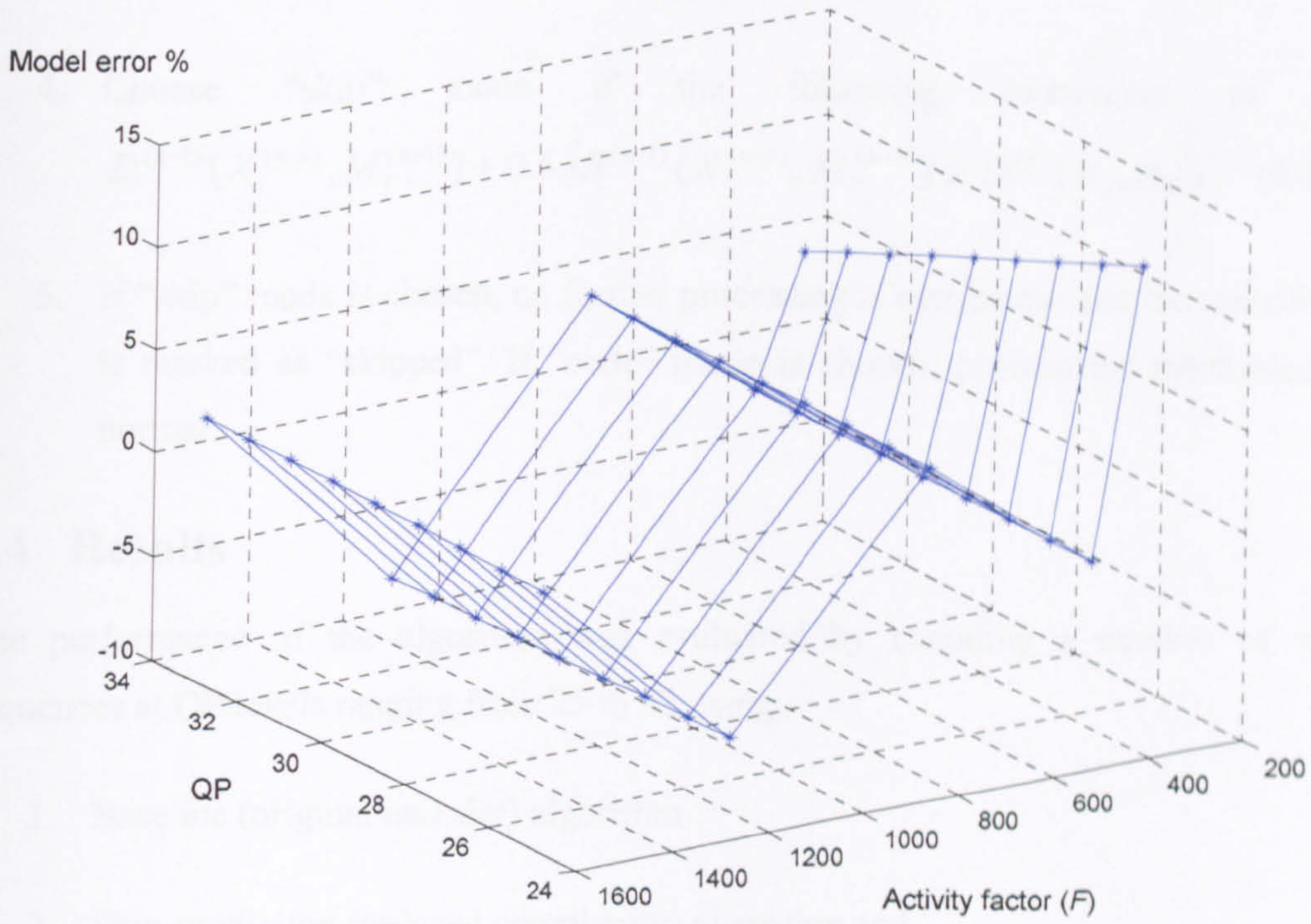


Figure 9.4: Percentage error between the model and experimental results for parameter λ.

9.3.4 Skip Prediction Algorithm

The skip prediction algorithm proceeds as follows:

1. For every macroblock, calculate $D^{(n)}(X_i, m_K)$ and read previously stored values $D^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)})$ and $R^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)})$.

2. Calculate the activity factor for the current macroblock (which is an estimated value using the rate and distortion values of the co-located macroblock in the previous picture):

$$F_i = D^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)}) \cdot R^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)}) \quad (9\text{-}16)$$

3. Calculate $\hat{\lambda}$ using equation (9-15) by substituting F_i for F.

4. Choose "skip" mode if the following expression is true:

$$D^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)}) + 0.5\hat{\lambda}R^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)}) \geq D^{(n)}(X_i, m_K) \quad (9\text{-}17)$$

5. If "skip" mode is chosen, no further processing is carried out and the macroblock is marked as "skipped". If "code" mode is chosen, process the macroblock as normal.

9.4 Results

The performance of the algorithm was evaluated by encoding a number of video sequences at QP levels ranging from 25 to 33 using;

1. Baseline (original encoder) algorithm

2. Skip-prediction (reduced complexity) algorithm and,

3. High complexity, rate-distortion optimised (RDO enabled) algorithm.

The resultant bit rates, average distortion and coding times were recorded. The rate-distortion performance of the three algorithms is shown in Figure 9.5. These results show that the reduced complexity algorithm has negligible effect on rate-distortion performance compared with the baseline algorithm.

Figure 9.6 shows sample frames from the Foreman (frame 16) and Mother and Daughter (frame 86) sequences; coded using the baseline and reduced complexity encoders. It is evident that there is no perceivable difference between frame quality of the baseline and reduced complexity encoders.

Table 9.2 shows the complexity reduction achieved by the reduced complexity encoder, both in terms of the percentage of macroblocks processed and coding time. The coding time is measured for the reduced-complexity encoder and the rate-distortion optimised encoder with respect to the time required by the unmodified baseline encoder. Note that the measured coding time includes the time taken to calculate the parameters of Equation (9-17). The results in Table 9.2 indicate that coding time is reduced by 19% to 67% compared with the baseline encoder due to the algorithm skipping (not processing) a

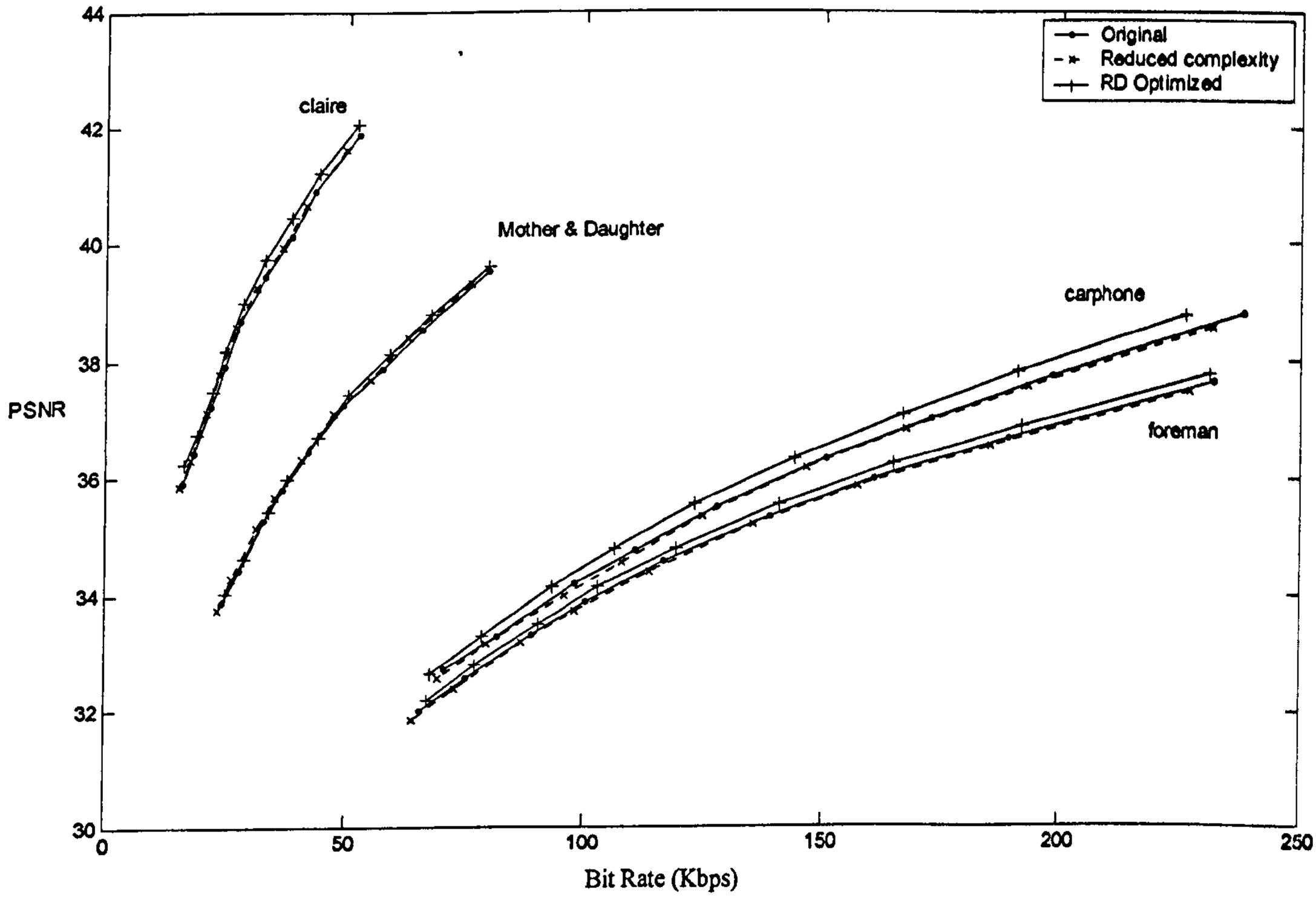


Figure 9.5: Rate-distortion performance of the baseline (original) algorithm, proposed skip prediction (reduced complexity) algorithm and high-complexity "rate-distortion optimised" algorithm used in the JM reference encoder for the first 250 frames of Carphone, Foreman, Claire and Mother and Daughter sequences.

significant number of macroblocks. The computational saving is higher for low-activity sequences such as Claire and Mother & daughter and lower for the high-activity sequence Foreman. The coding time for a particular sequence varies with the quantisation parameter QP and is typically higher for smaller QP.

Baseline - Foreman

Reduced complexity - Foreman

Baseline – Mother and daughter

Reduced complexity – Mother and daughter

Figure 9.6: Visual comparison between baseline and reduced complexity encoder output (Foreman, frame 16 @ QP =25 and Mother and Daughter, frame 86 @ QP = 25).

Table 9.2: Complexity reduction

Sequence name	Percentage of MBs processed by the reduced complexity encoder	Coding time (relative to baseline encoder)		
		Reduced complexity encoder	Baseline encoder	RD Optimised encoder
Carphone	64 – 71 %	0.64 – 0.73	1.0	1.54 – 1.77
Foreman	69 – 78 %	0.71 – 0.81	1.0	1.56 – 1.87
Claire	31 – 36 %	0.33 – 0.38	1.0	1.50 – 1.60
Mother & Daughter	44 – 45 %	0.46 – 0.47	1.0	1.51 – 1.69

Since the skip prediction algorithm makes skip decisions based on an estimate of $D(X_i, M_i)$ and $R(X_i, M_i)$, for a given QP value, the algorithm produces a lower mean bit rate than the unmodified encoder. The approximations in the skip mode decision tend to produce an increase in the number of skipped macroblocks and hence a drop in the coded bit rate. At the same time, these approximations produce a drop in PSNR due to the algorithm occasionally choosing to skip macroblocks that should have been coded. The resulting rate-distortion behaviour closely matches that of the unmodified baseline encoder. In some cases (e.g. Claire and Mother & daughter sequences), the skip prediction algorithm actually performs better than the baseline encoder. Mode decisions made by the "baseline encoder" (refer section 9.3) are sub-optimal in a rate-distortion sense and in some cases the mode decisions made by skip-prediction algorithm (which are rate-distortion optimised but are based on estimated parameters and are restricted to the two mode choices code or skip) produce better overall performance.

Coding time is substantially reduced because a significant number of macroblocks are not processed by the encoder. This reduction is particularly high for low-activity sequences, where typically a high proportion of macroblocks are skipped during the normal coding process, but is still evident for high-activity sequences such as Foreman.

9.5 Conclusion

This chapter describes a new algorithm for skip-macroblock prediction in an H.264/AVC encoder. The rate-distortion costs of coding or skipping a macroblock are estimated prior to processing and a decision is made whether to code the macroblock or stop further processing based on a Lagrangian cost function. The significant aspects of this algorithm can be summarised as follows:

- Encoder computational complexity is reduced by 19-67% with no significant loss of rate-distortion performance.
- The computational saving depends on the activity of the video sequence and is particularly significant for sequences with low/medium activity.
- The algorithm automatically adapts to different sequence statistics (without the need for any thresholds) by adaptively modelling the Lagrange multiplier based

on QP and sequence activity. An activity factor (F) is used to classify the sequence activity.

The next chapter describes an algorithm to control the computational complexity of an H.264/AVC encoder. The complexity reduction algorithm described above is further developed by extending the Lagrangian rate-distortion cost function to incorporate computational complexity so that the encoding process can be maintained at a target level of complexity.

10 Complexity Control of an H.264/AVC Encoder

10.1 Introduction

This chapter presents a complexity control algorithm for an H.264/AVC encoder. The complexity control algorithm is based on the complexity reduction algorithm described in chapter 9 which achieves varying degrees of complexity savings depending on the source video statistics. The contribution of the complexity control algorithm is to enable control of the computational cost of H.264/AVC encoding. In a computation or power-constrained application it is important to be able to control and manage the computational complexity of key components such as the video encoder. In the context of this algorithm, the level of complexity is measured by the number of skipped macroblocks.

The Lagrangian rate-distortion cost estimation carried out during skip macroblock prediction (chapter 9) is extended to incorporate computational complexity. Computational savings are achieved by early prediction of skipped macroblocks prior to motion estimation through estimating a Lagrangian complexity-rate-distortion cost function. A feedback control algorithm ensures that the encoder maintains a pre-defined target computational complexity. Simulation results demonstrate that the algorithm can effectively control the complexity of the encoding process whilst maintaining good rate-distortion performance. This work has been presented in [9].

10.2 Lagrangian Complexity-rate-distortion Cost

The goal of complexity-rate-distortion optimised mode selection is to choose, from K available coding modes, the mode that minimizes distortion subject to a computational complexity constraint and a rate constraint. Keeping the same notation as in the previous discussions for a group of N macroblocks and K possible modes, an additional constraint on computational complexity is introduced as C_c. Assuming computational complexity and rate are independent resources, the constrained optimisation problem can be expressed as:

$$\mathbf{M}^{*} = \arg\min_{\mathbf{M}} D(\mathbf{X},\mathbf{M}) \qquad (10\text{-}1)$$

Subject to

$$R(\mathbf{X},\mathbf{M}) \le R_c \text{ and } C(\mathbf{X},\mathbf{M}) \le C_c$$

where $\mathbf{M}^*$ is a vector of optimal mode allocations, $D(\mathbf{X}, \mathbf{M})$ is the distortion, $R(\mathbf{X}, \mathbf{M})$ is the coded rate and $C(\mathbf{X},\mathbf{M})$ is the computational complexity of carrying out mode allocation process for the group of macroblocks. This may be written as an unconstrained problem using a Lagrange multiplier method as:

$$\mathbf{M}^* = \arg\min_{M} \sum_{i=1}^{N} J(X_i) \tag{10-2}$$

Where, $J(X_i)$ is the complexity-rate-distortion cost for macroblock i, and is given as

$$J(X_i) = D(X_i,\mathbf{M}) + \lambda_r \cdot R(X_i,\mathbf{M}) + \lambda_c \cdot C(X_i,\mathbf{M}) \tag{10-3}$$

Where, λ_r is the Lagrange multiplier associated with rate and λ_c is the Lagrange multiplier associated with computational complexity. The proof of the Lagrange multiplier theorem in chapter 4 can easily be extended to incorporate the second resource constraint. The Lagrange multipliers λ_r and λ_c are chosen, considering all N macroblocks, such that the rate constraint and the computational constraint holds respectively.

Assuming macroblock independence, the sub-optimal solution can be written as

$$\mathbf{M}^* = \sum_{i=1}^{N} \arg\min_{M_i} J(X_i) \tag{10-4}$$

Where now:

$$J(X_i) = D(X_i,M_i) + \lambda_r \cdot R(X_i,M_i) + \lambda_c \cdot C(X_i,M_i) \tag{10-5}$$

This equation represents the complexity-rate-distortion cost of coding a macroblock and is the basis of the complexity control algorithm described in the following sections.

10.3 Complexity Control using Macroblock Skip-Prediction

The complexity control algorithm controls computational processing through early identification of macroblocks to be skipped. Prior to coding each macroblock, the encoder estimates the complexity-rate-distortion cost of coding or skipping the macroblock. Based on these estimates, the macroblock is either coded as normal (i.e. the encoder processes the macroblock and selects an appropriate coding mode) or skipped (i.e. no further processing is carried out). The skip prediction model (explained in the next section) aims to control computation through processing a proportion of macroblocks according to a complexity budget, whilst trying to maintain optimum rate-distortion performance. The algorithm's performance is compared with the "baseline encoder" (JM7.3) described in the previous chapter.

10.3.1 Skip Prediction Model

Let M_i be the coding mode (one of K possible modes) chosen by the baseline encoder for macroblock X_i. The complexity-rate-distortion cost of coding the macroblock X_i is given in equation (10-5). If the skip mode is $M_i=m_K$, then the rate-distortion cost of skipping a macroblock is:

$$J(X_i, m_K) = D(X_i, m_K) \tag{10-6}$$

Note that the rate associated with a skipped macroblock is effectively zero. The computational complexity associated with a macroblock identified as "skipped" prior to coding is assumed to be negligible, since no further processing is carried out.

The following simplifying assumptions are made:

- The computational complexity of all coded macroblocks is approximately the same. The JM Encoder (un-optimised low complexity mode) evaluates all possible macroblock modes (described in chapter 8 – 8.3.1) and selects the mode that minimises a cost function. Therefore, approximately the same amount of processing resources is used regardless of the selected coding mode, hence $C(X_i, M_i) = 1$.

- The effect on the bitrate due to complexity reduction is negligible. However, note that since macroblock skip prediction is used for complexity control, a significant

number of macroblocks are expected to be skipped (over and above the percentage of macroblocks normally skipped by the baseline encoder) to meet high complexity reduction targets. Therefore the bitrate may be reduced while trying to achieve high complexity reduction targets (see results in section 10.4).

Macroblock X_i should be skipped (not coded) if,

$$D(X_i,M_i)+\lambda_r \cdot R(X_i,M_i)+\lambda_c \geq D(X_i,m_K) \qquad (10\text{-}7)$$

This is the decision-making criteria for the skip prediction model. Note that the difference between this equation (10-7) and the decision making criteria for the complexity reduction algorithm (chapter 9, equation 9-4) is the introduction of λ_c. Computational complexity control is achieved by selecting the appropriate λ_c value, i.e. finding the λ_c value that matches the desired complexity constraint (see next section).

The estimation of the parameters, $D(X_i,M_i)$, λ_r, $R(X_i,M_i)$ and $D(X_i,m_K)$ are carried out as in the complexity reduction algorithm (chapter 9) where the rate constraint is met by estimating the Lagrange multiplier for rate as a function of QP and sequence statistics.

10.3.2 Complexity Control (*frame level*)

The objective of the algorithm is to achieve a computational complexity target for each frame (in terms of target number of processed macroblocks) by selecting an appropriate λ_c whilst trying to maintain good complexity-rate-distortion performance.

Finding the appropriate λ_c to match a particular target complexity can be an iterative process because the complexity saving (skipped macroblocks) depends on the QP and the sequence statistics. In [53], a feedback algorithm has been used to estimate the Lagrange multiplier without using an iterative process. This is possible without losing optimality because, from the Lagrange multiplier theory it is understood that whatever Lagrange multiplier is chosen, the objective function is minimised for the corresponding resource constraint.

Complexity control is achieved by choosing the parameter λ_c via an adaptive proportional feedback algorithm. A complexity reduction budget (specified as a percentage of average

skipped macroblocks per frame) is allocated at the beginning of the encoding process. This budget is converted to a target number of macroblocks to be skipped per frame depending on the frame size. The parameter λ_c is updated for the current frame (n) according to the following formulae.

$$\lambda_c^{(n)} = \lambda_c^{(n-1)} + \delta\lambda_c^{(n)} \qquad (10\text{-}8)$$

Where,

$$\delta\lambda_c^{(n)} = 0.2(1-C')^2(S_{target}^n - S_{actual}^{(n-1)}) \qquad (10\text{-}9)$$

$$\text{and, } C' = (\text{complexity reduction target})/100 \qquad (10\text{-}10)$$

The parameter S_{target}^n represents the target number of macroblocks to be skipped in the current frame and the parameter $S_{actual}^{(n-1)}$ is the actual number of macroblocks predicted as skipped during the encoding of the previous frame. The proportionality constant $0.2(1-C')^2$ was experimentally obtained and changes adaptively according to the complexity budget in order to stabilize the feedback loop at a range of target complexity budgets. The parameter $\lambda_c^{(0)}$ is initialised to zero at the start of the encoding process.

10.3.3 Skip Prediction Algorithm

<u>At the beginning of encoding</u>

Read the target complexity level set by the user and calculate S_{target}^n by multiplying (and rounding to nearest integer) the complexity reduction target with the number of macroblocks in a frame.

<u>At the beginning of each frame:</u>

If current frame = first frame

Set $\lambda_c^{(n)} = 0$

Else

Calculate $\lambda_c^{(n)}$ for the current frame using the equation (10-8)

Prior to coding each macroblock:

(The steps, 1, 2 and 3 are identical to the complexity reduction algorithm in chapter 9.)

1. Calculate $D^{(n)}(X_i, m_K)$ and read previously stored values $D^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)})$ and $R^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)})$.

2. Calculate the activity factor as in chapter 9, equation (9-16).

3. Calculate $\hat{\lambda}_r$ as in chapter 9, equation (9-15).

4. Choose "skip" mode if the following expression is true:

$$D^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)}) + 0.5\hat{\lambda}_r R^{(n-1)}(X_i^{(n-1)}, M_i^{(n-1)}) + \lambda_c^n \geq D^{(n)}(X_i, m_K) \quad (10\text{-}11)$$

5. If "skip" mode is chosen, no further processing is carried out and the macroblock is marked as "skipped". If "code" mode is chosen, process the macroblock as normal.

At the end of each frame

Record the actual number of macroblocks skipped by the algorithm during encoding of the current frame. This is used as $S_{actual}^{(n-1)}$ for the calculation of $\lambda_c^{(n)}$ for the next frame.

10.4 Results and Discussion

A number of QCIF test sequences were encoded using the "baseline" encoder using the above algorithm. Figure 10.1 shows the rate-distortion-complexity performance of the baseline codec (dashed lines) and the complexity control algorithm (solid lines) for the low-activity Claire sequence (QCIF, first 250 frames). The sequence is encoded at QPs from 26 to 32 with complexity reductions from 10% to 80% of skipped macroblocks. Note that a frame in a QCIF sequence consists of 99 macroblocks; therefore the target complexity reduction is used directly as the target number of macroblocks to be skipped.

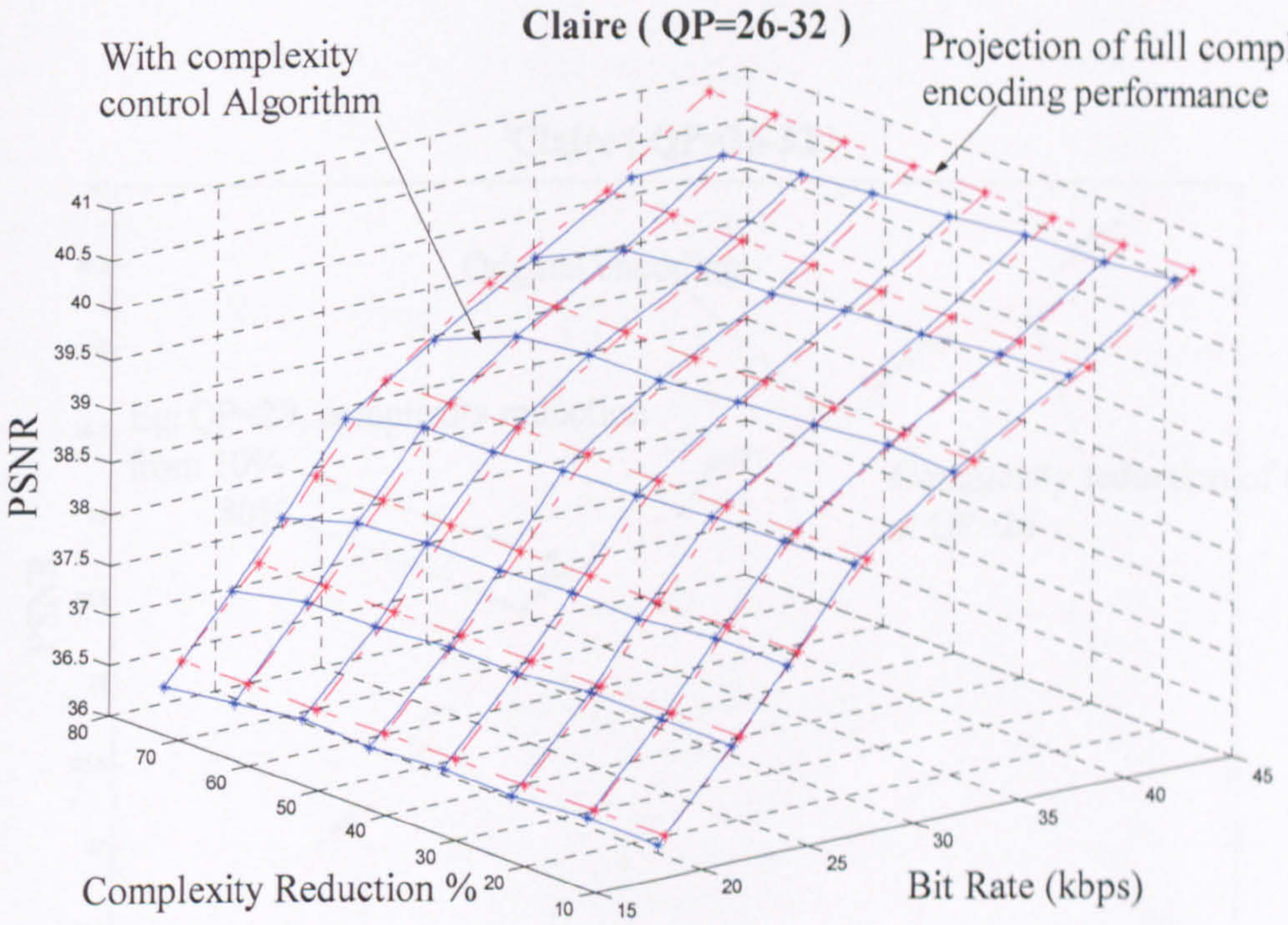


Figure 10.1: Claire – Distortion, rate and complexity performance

Figure 10.2 is the projection of the surfaces of Figure 10.1 on to the Rate-Distortion plane. It is evident that the complexity control algorithm performs better than the baseline encoder for most target complexities. For low activity sequences, the Lagrangian skip-prediction produces better overall rate-distortion performance, compared with the un-optimised baseline encoder (explained in section 9.4).

For the Claire sequence, the baseline encoder skips 75% of macroblocks at QP=26 and 85% of macroblocks at QP=32 (i.e. 75-85% of macroblocks are skipped after coding) during normal encoding. This is reflected in the performance of the complexity control algorithm, where the rate-distortion performance deteriorates when the complexity reduction target is similar to, or higher than, the percentage of macroblocks skipped during baseline encoding, for example a complexity reduction target of 80% at QP=26. The increase in distortion due to incorrect prediction of some of the skipped macroblocks by the algorithm is offset by a reduction in rate (since a skipped macroblock does not contribute to the coded rate). The overall effect is to maintain good rate-distortion performance whilst reducing computation to a target level.

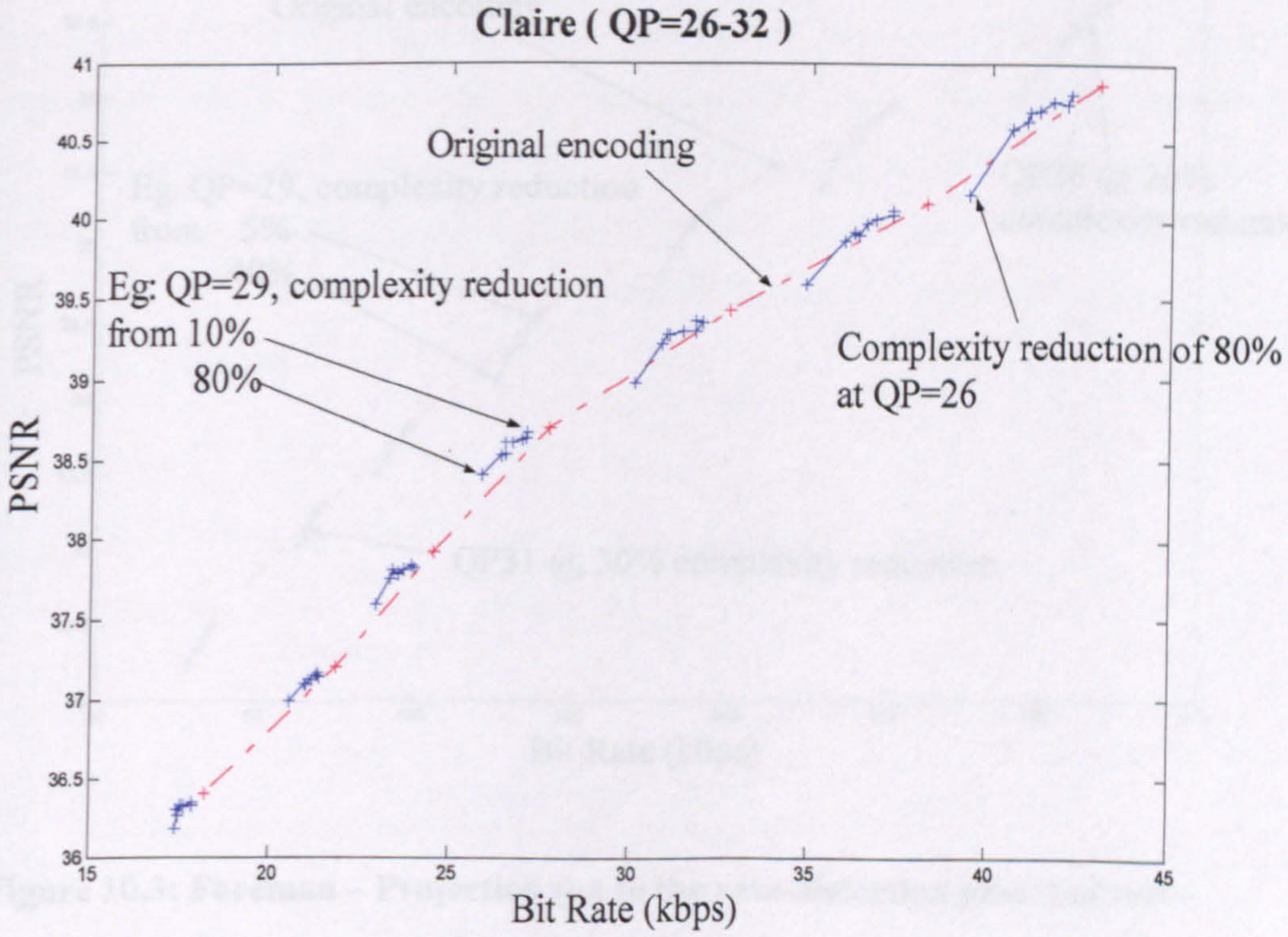


Figure 10.2: Claire – Projection (on to rate-distortion plane) of rate-distortion performance for all target complexities.

The results for the high activity Foreman sequence (QCIF, first 250 frames) are shown in Figure 10.3. The QP values are changed from 26 to 32 and the target complexity reduction changes from 5% to 40%. The baseline encoder skips 19% of the macroblocks at QP=26 and 35% at QP=32. Figure 10.3 demonstrates that the complexity control algorithm maintains similar rate-distortion performance to that of the baseline encoder, as long as the target complexity reduction is not significantly greater than the proportion of macroblocks skipped by the baseline encoder.

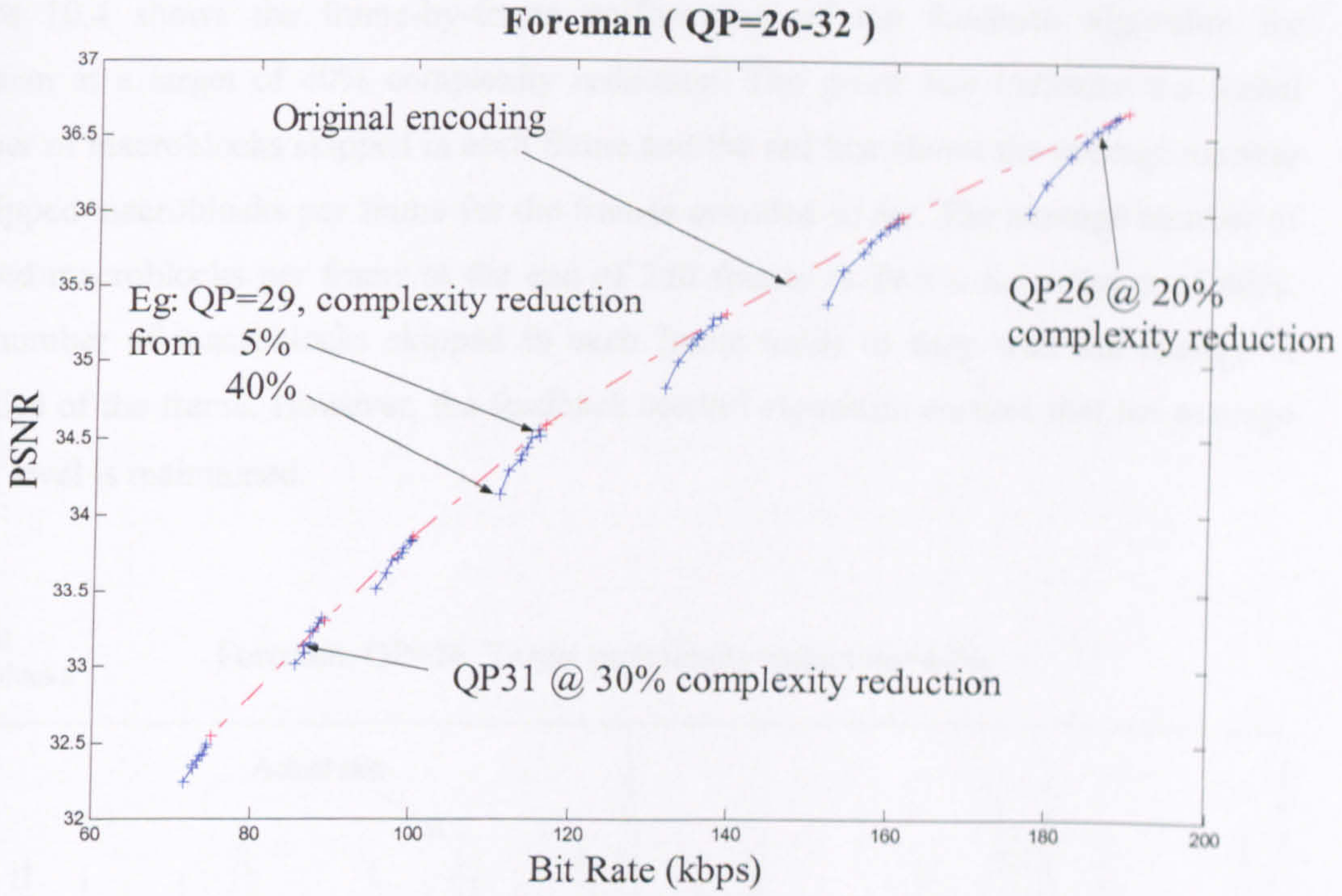


Figure 10.3: Foreman – Projection (on to the rate-distortion plane) of rate–distortion performance for all target complexities.

Table 10.1 shows the target and achieved complexity reductions using the above algorithm for the Foreman sequence encoded with QP=26. The baseline encoder (i.e. zero complexity reduction) produced a PSNR of 36.67 dB and a bit rate of 189.1 kbps. The change in rate and distortion with respect to the baseline encoder at different complexity reduction targets are also shown (the minus sign indicates a decrease). The data indicates that the average actual complexity reduction achieved is close to the target at different complexity reduction levels.

Table 10.1: Target and actual complexity reductions.

Target complexity reduction (%)	Actual complexity reduction (%)	Change in PSNR (dB)	Change in bit rate (%)
10	10.1	-0.05	-0.84
20	20.0	-0.13	-2.08
30	29.9	-0.30	-3.78
40	39.5	-0.68	-6.47

Figure 10.4 shows the frame-by-frame performance of the feedback algorithm for Foreman at a target of 40% complexity reduction. The green line indicates the actual number of macroblocks skipped in each frame and the red line shows the average number of skipped macroblocks per frame for the frames encoded so far. The average number of skipped macroblocks per frame at the end of 250 frames is 39.5% for a target of 40%. The number of macroblocks skipped in each frame tends to vary with the change in statistics of the frame. However, the feedback control algorithm ensures that the average target level is maintained.

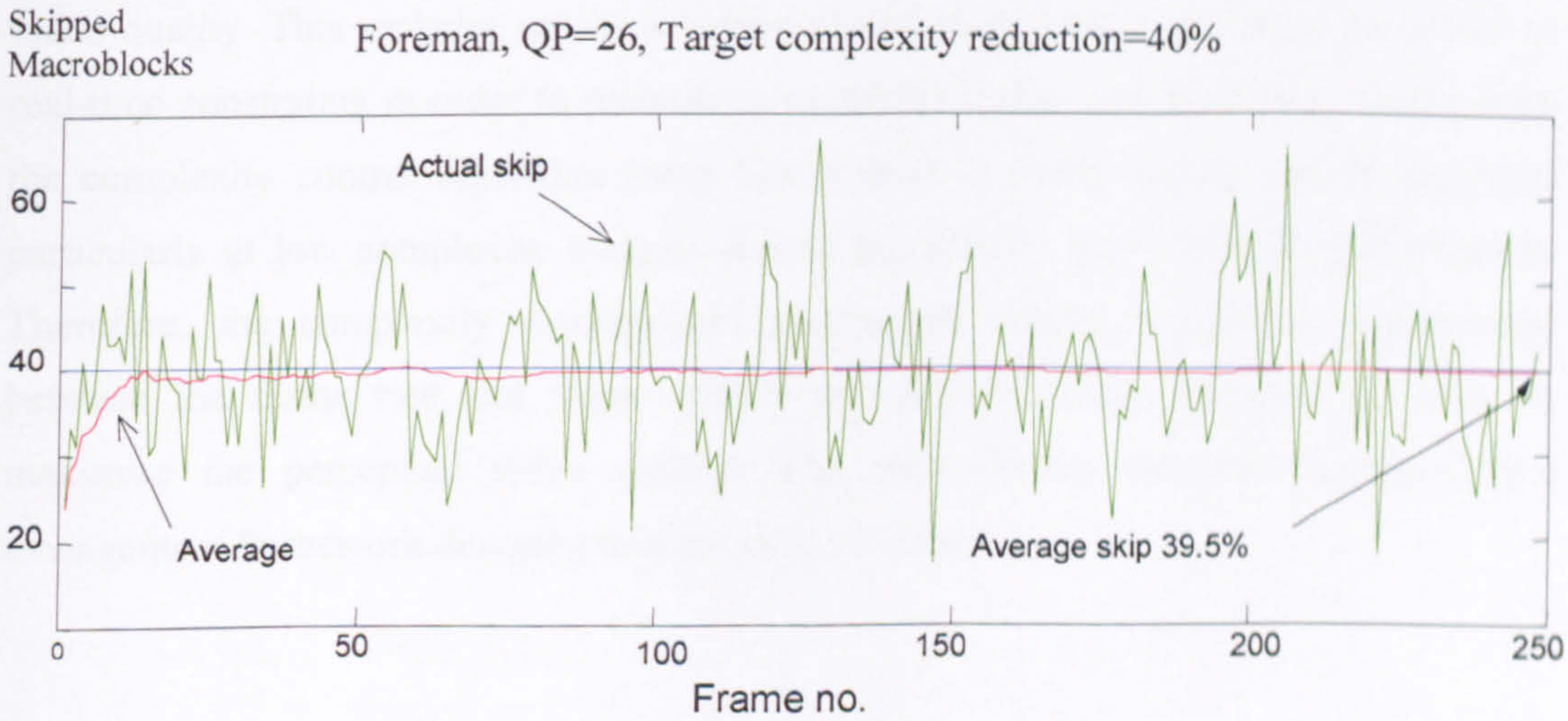


Figure 10.4: Foreman – Performance of the feedback algorithm (target skip 40%)

10.5 Conclusion

This chapter describes an algorithm for complexity control of an H.264 encoder. In contrast to normal codec operation, the complexity-rate-distortion cost of coding or skipping each macroblock is estimated prior to encoding in order to determine whether it is beneficial (in a complexity-rate-distortion sense) to process the macroblock, which is an extension of the complexity reduction algorithm described in chapter 9. Simulation results demonstrate that this algorithm can effectively control the computational complexity of encoding, meeting an average, per-frame target complexity reduction.

When the complexity reduction target (as a percentage of macroblocks to be skipped) is similar to or less than the percentage of macroblocks skipped during "normal" encoding,

rate-distortion performance is maintained at a similar level to that of the baseline reference encoder without the calculations utilised by the baseline codec. This algorithm provides a flexible means of controlling the computational cost of a video encoder whilst maintaining good rate-distortion performance. This is likely to be particularly useful for computation-constrained applications, to control the video codec complexity according to a computational budget.

However, this algorithm does not address the problem of real-time encoder management. In real-time resource constrained scenarios, a computational management mechanism is needed to utilise the available processing resources to maximise the rate constrained video quality. This includes setting a computational budget for each frame according to real-time constraints in order to maintain a particular frame rate. However, results from the complexity control algorithm show that a drop in frame quality can be expected particularly at low complexity budgets due to the drop in rate-distortion performance. Therefore, the complexity management mechanism should achieve a compromise between the frame rate and frame quality for different video statistics in order to maximise the perceptual video quality. The next chapter describes a complexity management framework designed to address these issues.

11 Complexity Management of a Real-Time H.264/AVC Encoder

11.1 Introduction

This chapter describes a complexity management approach for an H.264/AVC encoder running in a processor/power constrained environment. This complexity management approach has been published in [10], led to a UK patent application [11] and is currently being reviewed for a journal publication [12].

This work addresses the problem of maintaining acceptable coded video quality in a computation-constrained environment. The objective of this complexity management approach is to maintain a smooth video frame rate whilst ensuring that the frame quality is not degraded unacceptably. A frame-level algorithm calculates a target coding time for each frame and drops frames when necessary to maintain acceptable image quality. A per-frame algorithm controls the coding complexity of each frame in order to achieve the target coding time. The performance of the approach is evaluated by carrying out subjective tests and comparing the managed complexity encoder with a reference encoder in a computation-constrained scenario. Subjective results show that the managed complexity encoder consistently outperforms the reference encoder.

11.2 Video Coding Complexity and Perceived Video Quality

With the adoption of H.264/AVC for an increasing range of mobile and low-power applications, there is a requirement to reduce and/or manage the computational cost of video coding. A mobile platform may be required to carry out a number of processing functions with limited computational resources. The combination of limited processing capability and the high complexity of H.264 video encoding poses a problem for real-time video processing on such a platform. Typical approaches to this problem are to (a) reduce the video frame rate in order to maintain real-time coding and/or (b) reduce the complexity of video encoding.

11.2.1 Video Frame Rate and Perceived Video Quality

The perceived quality of a video sequence depends on the quality of each frame and the video frame rate. Research into user acceptability of video at different frame rates has produced somewhat mixed results. The authors of [96] have shown that a significant loss in frame rate does not proportionately reduce the perceptual quality or the acceptability of the encoded video and that the ability to understand the information content of the sequence does not degrade significantly with reduction in frame rate. The results presented in [97] indicate that when watching sports highlights (high activity) using small screens such as mobile video phones, the picture quality is more important than frame rate. However, in [98] the authors suggest that a lower frame rate has a particularly adverse effect on two-way multimedia communication. The authors of [99] report that users perceive the effect of reduced frame rate differently depending on the video content. They find that reduced frame rate leads to progressively lower acceptability ratings, particularly for scenes with low activity and movement and less so for high activity scenes (the opposite to what might be intuitively expected).

11.2.2 Reduced Complexity Video Coding

Low complexity or variable complexity algorithms proposed for a number of aspects of the H.264/AVC encoding process have been investigated in chapter 7. A common characteristic of these approaches is that (a) significant reductions in coding complexity may reduce rate-distortion performance and (b) the saving in complexity is typically content-dependent.

Chapter 9 describes a complexity reduction algorithm for an H.264/AVC encoder (publications [7] and [8]) that reduces encoder computation through early prediction of skipped macroblocks by estimating a Lagrangian rate-distortion cost function. It was demonstrated that the algorithm can achieve computational savings of 19-67% with no significant loss of rate-distortion performance. However, the actual complexity savings are dependant on source content, in particular on the amount of motion and detail in the sequence. Chapter 10 described how the Lagrangian cost function estimation based macroblock skip prediction is further extended in order to achieve computational complexity control (publication [9]).

The contribution of the new computational complexity management approach is to use a control algorithm (which uses real time coding statistics to set computational complexity targets) in conjunction with the variable-complexity algorithm described in chapters 9 and 10 to manage the computational resources utilized by an H.264/AVC encoder. The objective is to achieve an acceptably high frame rate whilst maintaining good image quality, when computational resources are limited. It is hypothesised that the perceived quality obtained using this method will be greater than the quality produced by an encoder that drops frames in order to maintain real-time coding in a computationally constrained situation.

To show this, the performance is compared between the two encoders which operate with limited computational resources: a reference encoder (that drops frames when computational resources are restricted) and an encoder which uses our "managed complexity" (MC) approach. Subjective video quality assessment tests show that the encoder using the MC algorithm consistently out-performs the reference encoder.

The remainder of this chapter is organised as follows. Section 11.3 describes the new complexity management approach in detail. The experimental method is described in section 11.4. Experimental results are presented in section 11.5 and discussion and conclusions are provided in section 11.6.

11.3 Complexity Management of H.264/AVC Encoding

Figure 11.1 shows an overview of the computational complexity management approach. Video frames to be coded are stored in an input frame buffer. A frame-level control algorithm calculates a target coding time for the next video frame based on the overall coding delay and the target video frame rate. Frames are flushed from the buffer (dropped) if the coding delay becomes so high that the required frame rate can not be maintained. A per-frame control algorithm manages the computational complexity within a video frame with the aim of encoding the frame within the target encoding time.

The objective is to maintain a smooth video frame rate where possible (i.e. minimize the number of dropped frames) whilst ensuring that frame quality is not degraded unacceptably. The following sections describe the Frame level control algorithm and Per-frame complexity control algorithm in detail.

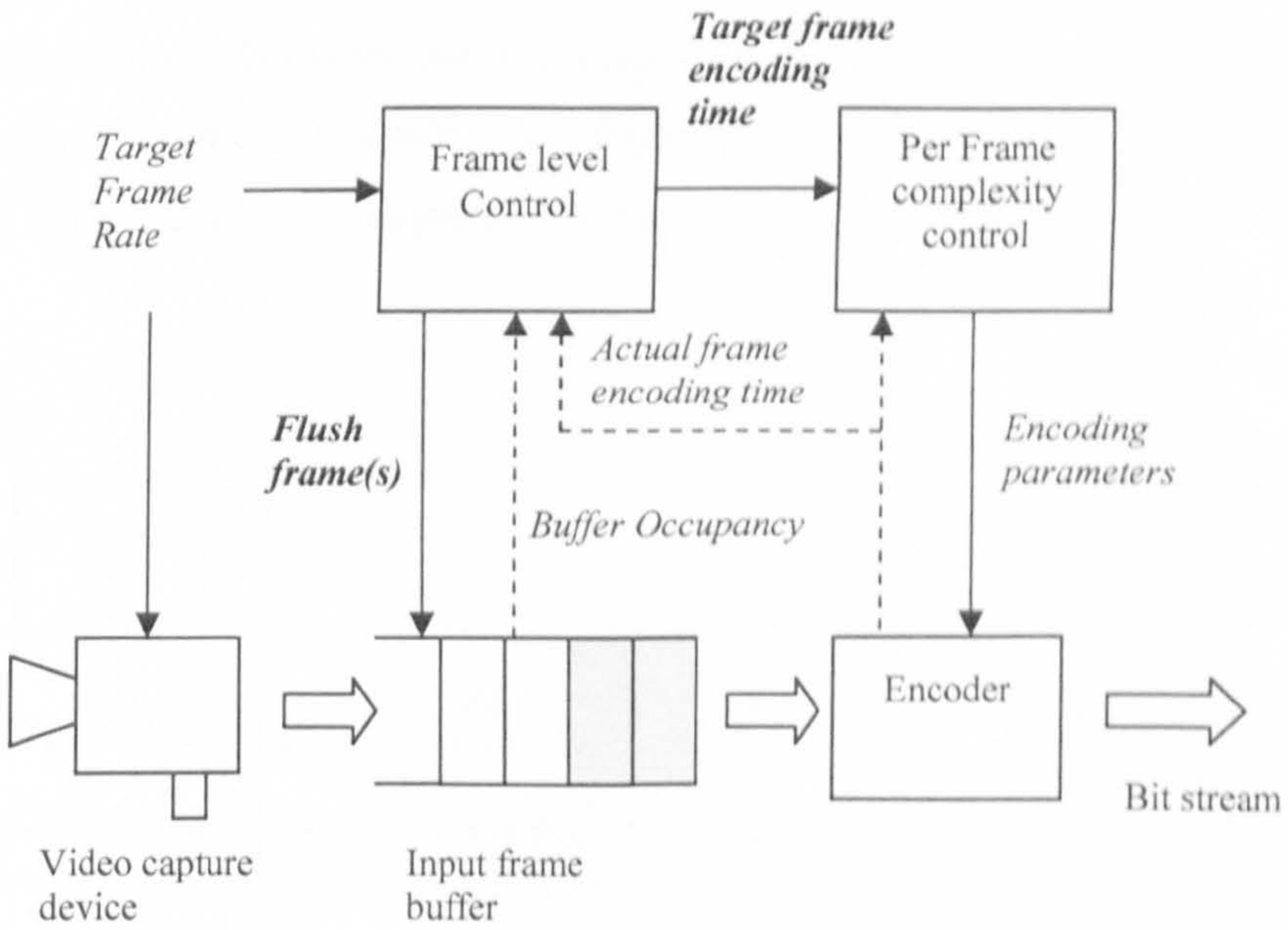


Figure 11.1: Block diagram of managed complexity encoder

11.3.1 Frame Level Control Algorithm

The main objective of the frame level control algorithm is to maintain the target frame rate of the encoder. To achieve this objective, the frame level control algorithm calculates a target encoding time for the next frame. The target encoding time for the next frame is calculated based on the total delay experienced by the frame(s) in the input frame buffer. This target time is to be used by the per-frame complexity control algorithm to control the complexity of each frame so that the frame is encoded within the allocated time

11.3.1.1 Calculation of total delay

The algorithm calculates the total delay (or waiting time) experienced by each frame at the start of encoding the frame. This waiting time is an indication of the real-time performance of the encoding process (i.e. whether the encoder has enough processing resources to maintain the frame rate at current complexity level). The total delay is used to calculate a target encoding time for each frame.

Figure 11.2 shows a possible timing scenario of the input frame buffer. The horizontal axis represents time. Input frame buffers drawn below the time axis show the buffer level just after frame inputs from the capturing device. Buffers drawn above the time axis show

the levels at the beginning and end of encoding of each frame. It is assumed that a negligible time is spent between end of encoding one frame and start of encoding the next frame (if the next frame is available) and that the buffer space of the current frame will not be freed until it has been completely coded.

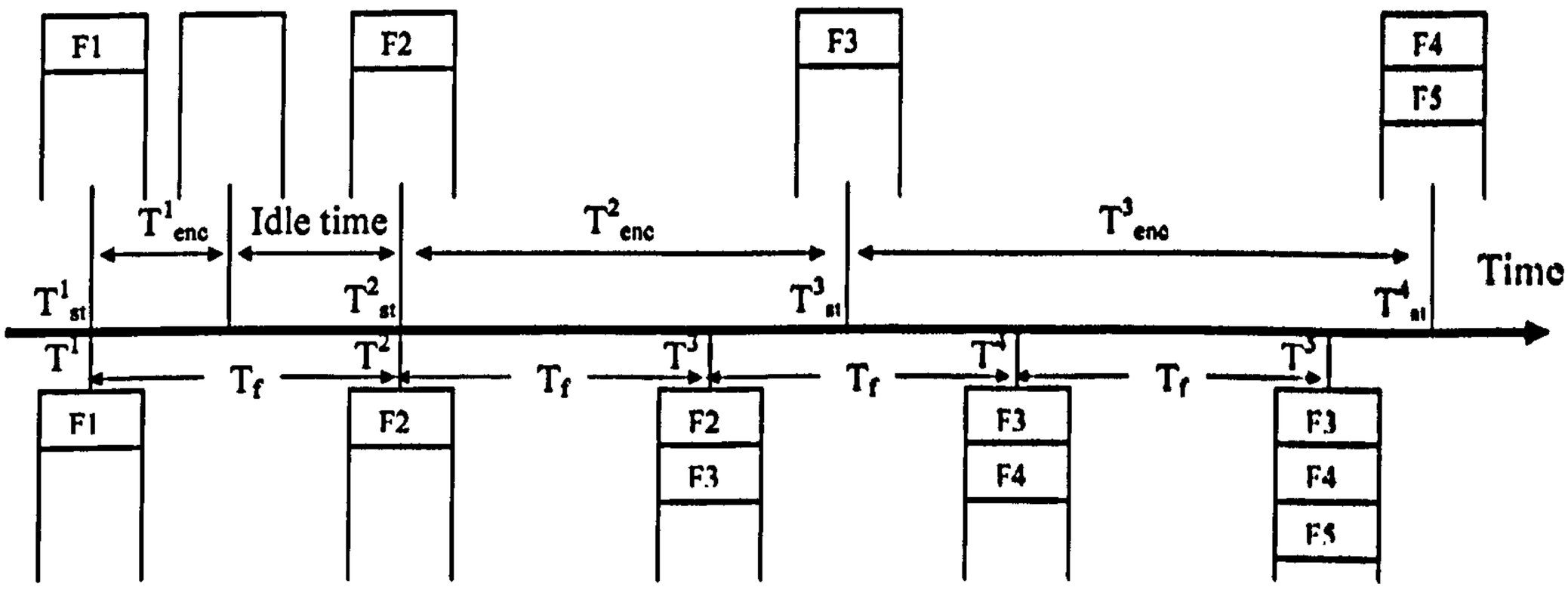


Figure 11.2: Input frame buffer timing diagram.

Labels F1, F2...F5 represents video frames in the buffer and the input times (from the capturing device to the frame buffer) are marked as T^1, T^2...T^5 respectively. The frame interval (i.e. time between frames) is T_f. The time at the start of encoding of each frame is labelled as T^1_{st}, T^2_{st}... T^4_{st} and the actual time taken to encode frames F1, F2 and F3 is T^1_{enc}, T^2_{enc} and T^3_{enc}.

Frames 1 and 2 are started as soon as they are received. However, the encoder stays idle for some time because coding of frame 1 is finished before the arrival of frame 2. The coding of frames 3 and 4 is delayed because the encoder takes a longer time to encode previous frames.

Total delay of frame F3 = $T^3_{st} - T^3$

Total delay of frame F4 = $T^4_{st} - T^5 + T_f$

In general, the total delay of frame "n" can be calculated as:

$$Total\_Delay^n = T_{st}^n - T^l + T_f \cdot (Buffer\_level - 1) \quad (11\text{-}1)$$

Where, $Buffer\_level$ is the buffer level at the start of frame "n" and T^l is the time at which the latest frame was input to the buffer.

11.3.1.2 Target encoding time for the next frame

Should the coding time of a particular frame exceed T_f, this can not be compensated for by subtracting the delay from the available encoding time of the next frame only, because this will lead to a large variation in frame quality. Therefore a linear model is defined to allocate a target time T_{target}^n for the frame "n" by distributing the delay over a number of frames as follows:

$$T_{target}^n = T_f - C \cdot Total\_Delay^n \quad (11\text{-}2)$$

Where C is a proportionality constant. When $Total\_delay^n = 0$, the target time T_{target}^n equals to the frame interval T_f.

Consider the situation where $Total\_delay^n$ has reached its maximum allowable value, i.e. when the buffer is about to overflow. The next frame has to be encoded almost instantaneously to make room for the incoming frame. In this case, $T_{st}^n - T^l = T_f$ and $Buffer\_Level = B$, where B is the maximum buffer size. Therefore equation (11-1) becomes;

$$Total\_Delay^n = T_f + T_f(B-1) = T_f \cdot B \quad (11\text{-}3)$$

At this point, the time available for the next frame T_{target}^n is zero, therefore a frame has to be skipped. Now by substituting the values for $Total\_delay^n$ into (11-2):

$$0 = T_f - C \cdot T_f \cdot B \quad (11\text{-}4)$$

$$C = \frac{1}{B} \quad (11\text{-}5)$$

Therefore, the time allocation for the next frame can be calculated by substituting values to equation (11-2):

$$T_{target}^{n} = T_f - \frac{1}{B} \cdot [T_{st}^{n} - T^{l} + T_f \cdot (Buffer\_level - 1)] \qquad (11\text{-}6)$$

The achievement of T_{target}^{n} depends on the performance of the per-frame complexity control algorithm and on sequence statistics. When processing resources are limited, it is possible to build up high delays. This can cause the input frame buffer to overflow so that frames are dropped automatically. However, when the delay increases significantly, T_{target}^{n} becomes very small, forcing the frames to be encoded with very low complexity. This may result in a very low frame quality leading to low perceptual quality even though a higher frame rate is maintained. This indicates that there should be a trade-off between the acceptable minimum thresholds for frame quality and frame rate, and that the trade-off point could be set according to the needs of specific applications. Section 11.3.2.3 describes how the per-frame complexity control algorithm is used to achieve the trade-off between maximum allowable distortion and frame rate[6].

11.3.2 Per-Frame Complexity Control Algorithm

The objective of the per-frame complexity control algorithm is to encode the frame(s) within the target encoding time provided by the frame level control algorithm. Here, the complexity control algorithm based on Lagrangian complexity-rate-distortion cost function estimation developed through chapter 9 & 10 is used (with some modifications to the control algorithm). However, other variable complexity algorithms (such as in [88]) can also be used with this framework, as long as a control parameter is available for complexity control.

---

[6] Alternatively, a minimum threshold on the target encoding time can be used as a control mechanism to achieve the trade-off between the minimum frame quality and the bit rate. This can be accomplished by dropping the next frame and moving on to the succeeding frame until the calculated target time is above the threshold.

11.3.2.1 Lagrangian cost based skip prediction

The complexity control algorithm developed in chapters 9 and 10 is adapted for the per-frame control algorithm. The computational complexity of each frame is controlled by an early macroblock skip prediction algorithm. This early macroblock skip-decision involves minimising the Lagrangian complexity-rate-distortion cost by choosing between coding or skipping a particular macroblock before the motion estimation process, based on the estimated cost of coding and the cost of skipping the macroblock.

The per-frame complexity control algorithm is identical to the complexity control algorithm (chapter 10) except for the following:

- In the per-frame complexity control algorithm, the computational complexity is measured as, and a target is set for, the encoding time. In the previous control algorithm the complexity was measured as, and a target was set for, the number of skipped (predicted) macroblocks.

- In the previous algorithm, the complexity target was set for all the frames at the beginning of the encoding process. However, in the per-frame complexity control algorithm a complexity target is set for each frame by the frame level control algorithm.

- A new feedback mechanism is used to calculate the appropriate Lagrange multiplier associated with complexity (coding time) constraint for the current frame λ_c^n, which is explained below.

11.3.2.2 Complexity control

In chapter 10 it was demonstrated that complexity can be effectively controlled by changing the Lagrangian multiplier associated with computational complexity at frame level. It was shown that a target level of complexity can be maintained by calculating the value of $\lambda_c^{(n)}$ for each frame using a feedback algorithm. This mechanism is adapted for the per-frame complexity control algorithm as follows.

$$\lambda_c^{(n)} = \lambda_c^{(n-1)} + \delta\lambda_c^{(n)} \tag{11-7}$$

Where,

$$\delta\lambda_c^{(n)} = \left| \frac{\delta\lambda_c^{(n-1)}}{T_{enc}{}^{(n-2)} - T_{enc}{}^{(n-1)}} \right| \cdot (T_{enc}{}^{(n-1)} - T_{target}{}^{n}) \qquad (11\text{-}8)$$

The time allocation for the current frame $T_{target}{}^{n}$ is supplied by the frame level control algorithm. The parameters $T_{enc}{}^{(n-1)}$ and $T_{enc}{}^{(n-2)}$ are the actual encoding times of previous frames. The proportionality variable $\left| \frac{\delta\lambda_c^{(n-1)}}{T_{enc}{}^{(n-2)} - T_{enc}{}^{(n-1)}} \right|$ represents the contribution of $\delta\lambda_c$ to a unit change in encoding time for the previous frame. This is used as the estimated rate of change for the next frame. Note that, in the situation where the numerator and/or denominator of the proportionality variable equals zero for the current frame, it is replaced by the proportionality variable used for the previous frame.

The parameter $\lambda_c{}^{(0)}$ is initialized to zero at the start of the encoding process and is used for the first P frame (assuming one slice per frame). Value $T^{(n-2)}$ is only available after encoding the 2nd P frame. Therefore, the following expression is used for the 2nd P frame:

$$\delta\lambda_c^{(n)} = 0.2(1-C')^2 \cdot (T_{enc}{}^{(n-1)} - T_{target}{}^{n}) \qquad (11\text{-}9)$$

Where C'= (complexity reduction target)/100, which is similar to the proportionality constant used in the previous control algorithm.

Note that occasionally it is possible to encounter large variations in $\delta\lambda_c^{(n)}$ due to large changes in scene content. This is because the estimated rate of change (the contribution of $\delta\lambda_c$ to a unit change in encoding time for the previous frame) may not be valid for the current frame. This can cause unwanted artefacts and large variations in distortion between consecutive frames. These include artefacts such as unwanted edges and trails of moving objects. Therefore, the value of $\delta\lambda_c^{(n)}$ is bounded by the limits 1 and -2 to avoid large perceptual quality differences between consecutive frames. The limit is biased towards the minus value to favour lower number of macroblocks to be skipped when scene content changes. These values are found experimentally by encoding the video sequences, Claire, Foreman, Carphone and Mother & Daughter at different complexity

levels, and at different boundary values (changed in 0.5 steps) and using informal observations to choose the values that minimise artefacts.

11.3.2.3 Limiting maximum distortion

The value $\lambda_c^{(n)}$ affects the complexity of the current frame by controlling the number of skipped macroblocks. When available processing time decreases, the time allocation for the current frame is decreased, resulting in increased distortion because more macroblocks are skipped due to a large increase in $\lambda_c^{(n)}$. Therefore an upper limit for $\lambda_c^{(n)}$ is introduced to limit the complexity reduction achieved by skipped macroblocks and thereby limiting the maximum distortion. Now equation (11-7) becomes:

$$\lambda_c^{(n)} = \min(Limit\lambda_c, (\lambda_c^{(n-1)} + \delta\lambda_c^{(n)})) \qquad (11\text{-}10)$$

$Limit\lambda_c$ acts as a tuning parameter which can be used to achieve a desired trade-off between frame rate and maximum allowable distortion because $\lambda_c^{(n)}$ directly controls the number of macroblocks skipped and therefore distortion. $Limit\lambda_c$ is set at 2.75 by encoding a number of sequences (Foreman, Clair, Carphone and Mother & Daughter) at 20% complexity (as explained in section 11.4), changing $Limit\lambda_c$ in 0.25 steps and choosing a $Limit\lambda_c$ value such that no obvious artefacts (due to skipped macroblocks) are observed. However, note that $Limit\lambda_c$ may also be set experimentally to satisfy a minimum mean opinion score (MOS) or PSNR depending on the application.

11.4 Experimental Method

Experiments are carried out to evaluate the performance of the managed complexity encoder compared with a Normal (full complexity) encoder operating in a real-time environment. The H.264/AVC reference software JM7.4 is chosen and modified to simulate real-time operation as follows.

11.4.1 Simulated Real-time Encoder

The following parameters are chosen to reflect likely choices for an H.264/AVC implementation on a power- and/or computation-constrained device.

1) Baseline Profile.

2) One reference frame, one slice per frame, P-slices only (after an initial I-slice).

3) Rate-distortion optimised mode selection disabled.

4) Original frame rate 15fps.

The JM7.4 encoder is modified to use multiple processing threads in order to simulate real time operation. This is achieved by updating a hypothetical frame buffer in fixed time intervals corresponding to a frame interval T_f, without interrupting the encoding process. The frame interval is used to simulate the desired complexity or processing resource level. Initially, the frame interval is experimentally obtained by calculating the average time the normal encoder spends in each frame (the "full complexity" frame interval). The frame interval is reduced by a desired percentage to achieve different levels of complexity or available processing time.

The reference encoder JM7.4 as modified above for simulated real-time operation (referred to as "Normal Encoder") is compared with a modified JM74 encoder including the complexity management algorithm (referred to as "Managed Complexity" or "MC" Encoder).

11.4.2 Subjective Tests

The MC encoder aims to achieve an acceptably high frame rate whilst maintaining good image quality compared to the normal encoder, in process constrained situations, so that the video will be more acceptable to the user.

Subjective video quality measurements are used to assess the user preference between the MC and Normal encoders. It is difficult to realistically assess video quality using objective methods such as rate-distortion performance when comparing video with different frame rates. However, objective measurements are also taken in to account to compare the MC and Normal encoders (see results). Pair Comparison (PC) and Degradation Category Rating (DCR) subjective tests are carried out using 15 participants.

11.5 Experimental Results

A number of test sequences are coded with the JM reference encoder and with a modified encoder incorporating the complexity management algorithm. The two encoders ("Normal" and "MC") are operated with identical processing resources. Starting with an original frame rate of 15fps, the processing resources are reduced as shown in Table 11.1. The "Normal" encoder drops coded frames to cope with the reduced processing resources. The "MC" encoder maintains a higher frame rate at the expense of a drop in PSNR within each coded frame.

Table 11.1: Performance of 'MC' encoder vs. 'Normal' encoder (QP =26, original frame rate 15fps).

			Normal Encoder			Managed Complexity Encoder		
No.	Sequence	Processing Resources	Frame rate	PSNR (average)	Bit Rate Kbps	Frame rate	PSNR (average)	Bit Rate Kbps
1	Claire	50%	**7.65**	40.74	23.7	**15**	40.17	30.8
2	Claire	33%	**5.10**	40.71	19.51	**11.96**	39.72	26.16
3	Claire	20%	**3.10**	40.65	14.8	**6.38**	39.74	19.9
4	Carphone	50%	**6.99**	37.56	79.9	**9.26**	37.11	91.8
5	Carphone	33%	**4.63**	37.54	59.6	**5.89**	37.17	65.95
6	Mother&Daughter	50%	**7.00**	38.55	30.37	**13.3**	38.08	37.27
7	Mother&Daughter	33%	**4.70**	38.56	24.65	**8.50**	38.13	30.53
8	Mother&Daughter	20%	**2.90**	38.53	18.54	**4.60**	38.17	22.83
9	Foreman	50%	**7.80**	36.47	91.36	**9.75**	36.08	100.71
10	Foreman	33%	**5.02**	36.50	68.22	**6.00**	36.19	75.43

Subjective tests are carried out with 15 observers using the Degradation Category Rating (DCR). Each observer is asked to rate the video quality produced by the 'Normal' and 'MC' encoders for each of the sequences and processing resource levels shown in Table 11.1 (reference as same sequence encoded at full processing resources).Figure 11.3 plots the mean score allocated to each sequence (the sequence numbers correspond to the test conditions listed in Table 11.1). These results demonstrate that the observers consistently preferred the decoded video produced by the 'MC' encoder.

For low and moderate activity sequences like Claire (sequence numbers 1, 2 and 3) and Mother and Daughter (sequence numbers 6, 7 and 8), there is a significant difference between perceptual quality of the 'Normal' and 'MC' encoders due to differences in frame rate. At lower activity levels the 'MC' encoder is able to skip a large proportion of macroblocks without degrading the image quality. However, higher activity sequences Carphone (sequence numbers 4 and 5) and Foreman (numbers 9 and 10) show a small difference in perceptual quality between 'Normal' and 'MC' encoders. At high activity levels, the difference between 'Normal' and 'MC' encoder frame rates is smaller because only a small number of macroblocks can be skipped without severely degrading the image quality.

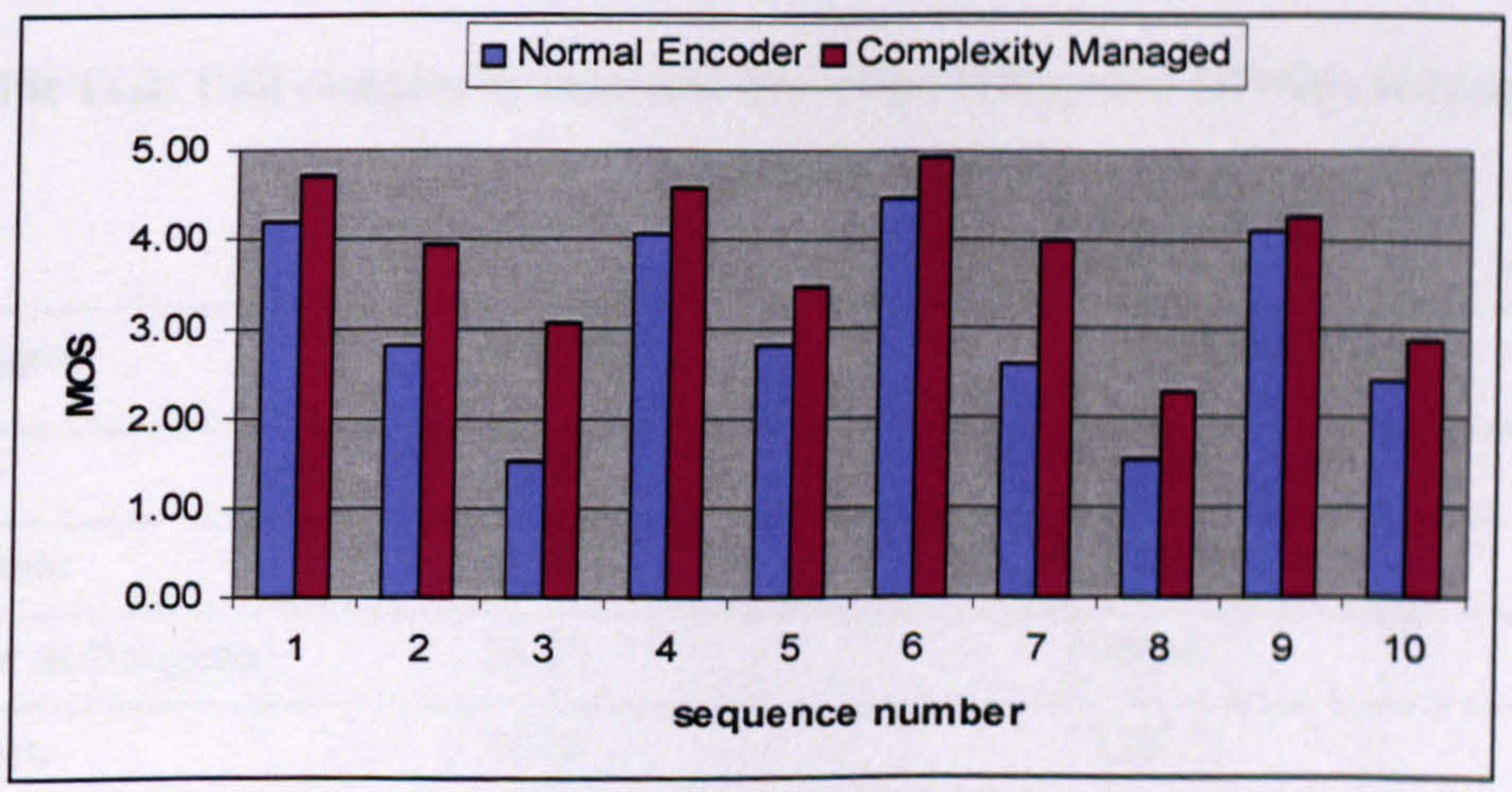


Figure 11.3: 'MC' encoder and 'Normal' encoder at various levels of complexity at same QP level (sequences are numbered as in Table 11.1)

However, note that (in Table 11.1) the average bit rate of the 'Normal' encoder is significantly less than the average bit rate of the managed complexity encoder at the same QP (QP=26) and complexity levels. This is because the Managed Complexity encoder achieves a significantly higher frame rate compared to the 'Normal' encoder.

Table 11.2 shows the rate and distortion values of the sequences encoded originally at full complexity. It is evident that the bit rate of the managed complexity encoder (from Table

11.1) is lower than the original bit rate (from Table 11.2) at all complexity levels and achieves better perceptual quality than the 'Normal' encoder at the same time. This is the desired performance in applications with a constant bit rate channel (or an upper ceiling for the bit rate) where perceptual quality should be maximised when processing power becomes limited as long as the bit rate limit is not exceeded, i.e. utilising the available bit rate to maximise perceptual quality.

It is worthwhile to evaluate the performance of the Managed Complexity encoder compared with the 'Normal' encoder, with both bit rate and complexity set at comparable levels. This enables direct comparison between the 'Normal' and 'MC' encoders for the perceptual quality.

Table 11.2: Full complexity rate and distortion (15fps and QP=26) of encoded sequences.

Sequence	PSNR	Bit Rate (Kbps)
Claire	40.76	44.75
Carphone	37.54	172.73
Mother & Daughter	38.57	59.96
Foreman	36.48	185.53

The complexity management algorithm is further tested at comparable bit rates and comparable complexity levels (see Table 11.3) for both encoders. In this experiment, the QP value for the 'Normal' encoder was set to ensure that the bit rate produced by the 'Normal' encoder was the same as, or higher than, that produced by the Managed Complexity encoder.

Table 11.3: Performance of managed complexity encoder vs. 'Normal' encoder : QP of the normal encoder is decreased to increase the overall bit rate above the bit rate of the managed complexity encoder.

			Normal encoder				Managed Complexity Encoder			
No.	Sequence	Proc. Res.	Frame rate	QP	PSNR (avg)	Bit Rate (Kbps)	Frame rate	QP	PSNR (avg)	Bit Rate (Kbps)
1	Claire	50%	7.53	24	42.13	31.25	15	26	40.17	30.8
2	Claire	33%	4.92	23	42.82	27.8	11.96	26	39.72	26.16
3	Claire	20%	2.98	23	42.8	20.79	6.38	26	39.74	19.9
4	Carphone	50%	6.83	25	38.55	93.1	9.26	26	37.11	91.8
5	Carphone	33%	4.48	24	38.51	68.41	5.89	26	37.17	65.95
6	Mother&Daughter	50%	6.8	24	40.02	39.59	13.3	26	38.08	37.27
7	Mother&Daughter	33%	4.4	24	39.99	31.19	8.5	26	38.13	30.53
8	Mother&Daughter	20%	2.8	24	39.98	23.63	4.6	26	38.17	22.83
9	Foreman	50%	7.58	25	37.45	107.75	9.75	26	36.08	100.71
10	Foreman	33%	4.88	25	37.44	80.29	6	26	36.19	75.43

Figure 11.4 shows the pair comparison results for the 'MC' and 'Normal' encoders operating at comparable complexity and bit rates. Despite the large reduction in image PSNR, the higher frame rate (and hence smoother video) obtained from the "managed complexity" encoder was considered to be subjectively superior. Similar to the previous experiment, the 'MC' encoder achieves significantly higher percentage acceptability for lower activity sequences than higher activity sequences. It is also evident that the differences between frame rates are more visible at low frame rates than at high frame rates. For example, Claire sequence at a processing resources level of 20% (sequence number 3), the 'Normal' encoder achieves a frame rate of 2.98 while the 'MC' encoder achieves 6.38 fps. At 50% resource level (sequence number 3), achieved frame rates are 7.53 (Normal) and 15 (MC). However, sequence number 3 (at 20%) shows a significant difference between 'Normal' and 'MC' encoder ratings compared with sequence number 1.

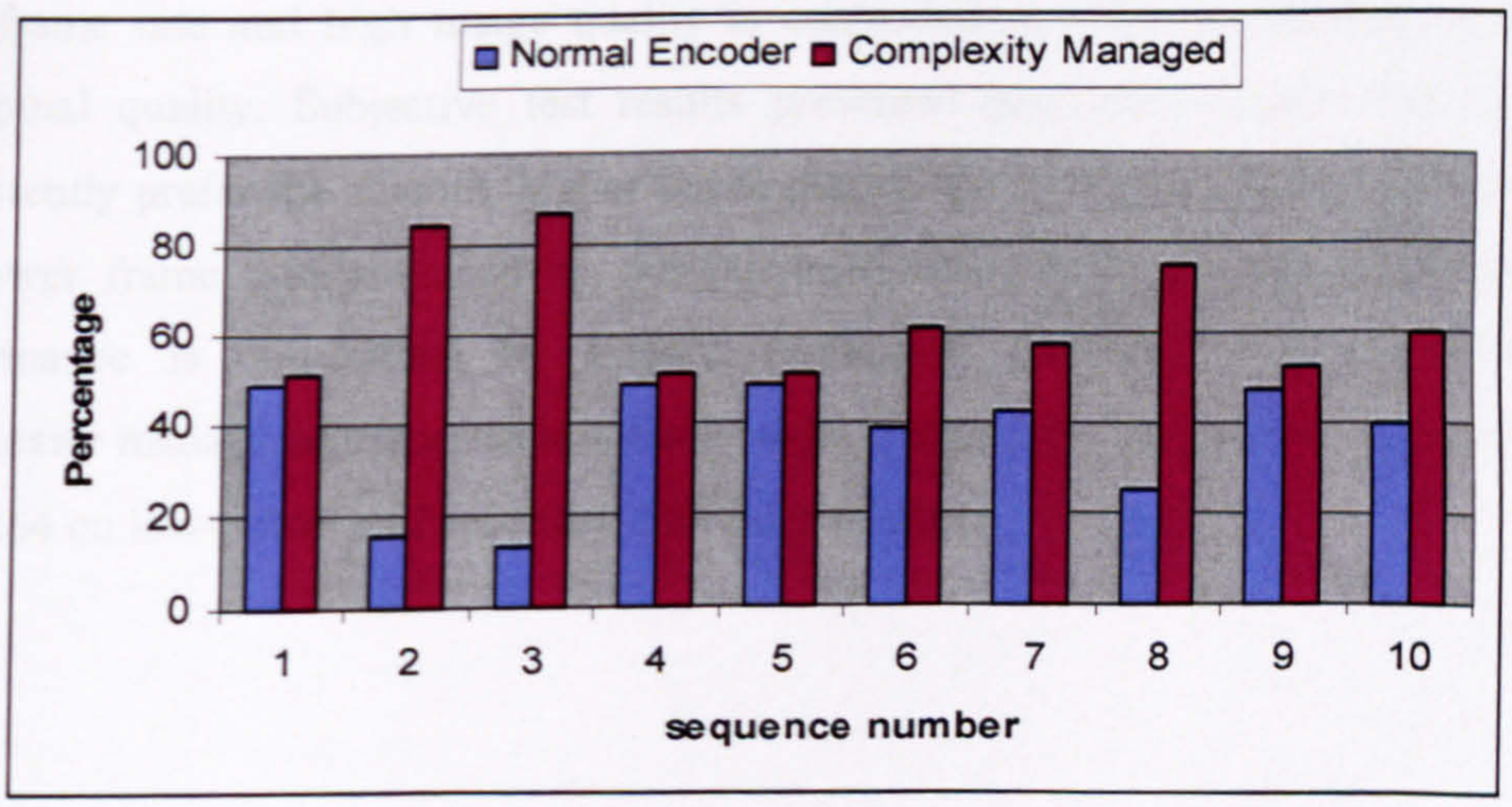


Figure 11.4: Acceptability by direct comparison – 'MC' encoder vs. 'Normal' encoder at various levels of complexity at same bitrate for each complexity level (sequences are numbered as in Table 11.3).

11.6 Conclusion

The aim of this work is to improve the perceptual quality of coded video in a computation-constrained scenario, through controlling per-frame complexity and frame rate. The results presented in Figure 11.3 (DCR test, MC encoder and reference encoder operating with the same quantisation parameter) show that the complexity management method consistently out-performs the reference encoder in a situation where performance is constrained by limited computational resources, i.e. a computation-constrained scenario. Figure 11.4 (PC test, MC encoder and reference encoder operating at the same bitrate) compares the performance of the two systems when each is operating at the same level of complexity and at the same bitrate, i.e. a computation **and** rate-constrained scenario. Even in this case, the complexity management algorithm outperforms the reference encoder. The results in Figure 11.4 also show that users consistently prefer a higher frame rate video sequence over a lower frame rate sequence, even at the expense of a drop in image quality. This is in accordance with the conclusions reached in [98] and [99].

This chapter presents a system for managing the computational complexity of H.264/AVC video encoding in a real-time scenario. The complexity management system controls the coding time of each frame of a video sequence and maintains a high frame rate where possible, whilst avoiding significant image quality loss. The trade-off between

high frame rate and high image quality is controlled in order to maintain acceptable perceptual quality. Subjective test results presented here demonstrate that observers consistently prefer the smooth, higher frame rate of the managed complexity encoder to the lower frame rate produced by the reference encoder, in scenarios where coding performance is constrained by limited processing capability. The computational complexity management approach is likely to be particularly useful for implementations of H.264 on low-power and processor-limited platforms.

12 Discussion and Conclusion

12.1 Introduction

This chapter summarises the main contributions of this work. The algorithms and experimental results are critically reviewed. The advantages and disadvantages of the main contributions are discussed. Possible directions for further research in relation to the main findings are also indicated. The thesis is concluded by highlighting the completion of each stage of the project towards the final objective, emphasising the relevance of this work to the research problem.

12.2 Main Developments and Results

The aim of this work has been to develop novel algorithms to manage the computational complexity of an H.264/AVC encoder so that the available processing resources are used efficiently in order to maximise the (rate constrained) video quality. The significant contributions of this research project can be classified into four algorithms or methods and they have been presented in four chapters.

Chapter 6 – Video quality evaluation with user feedback.

Chapter 9 – Complexity reduction of an H.264/AVC encoder using Lagrangian cost estimation.

Chapter 10 – Complexity control of an H.264/AVC encoder.

Chapter 11 – Complexity management of a real-time H.264/AVC encoder.

12.2.1 Video Quality Evaluation with User Feedback

This fast subjective video quality measurement method was developed as a more efficient alternative to the standard subjective video quality measurement methods such as Pair Comparison (PC). This method enables the user to interactively choose a preferred video clip from a set of alternative clips within a short period of time.

The test subject is presented with a continuous video display (looped in the case of short video clips) and interacts with the display using a slider bar. The slider position controls

the display of the video clip by controlling the video processing or by selecting from a number of video clips (in this case, the software displays the video uninterrupted by transferring from one clip to another while keeping the correct time position). The slider position is recorded at regular time intervals and the final slider position determines the chosen best parameters or the best clip for the test subject. The experimental results show that this method achieves comparable results to the standard Pair-wise Comparison (PC) test with a time saving of 83%-86%. The advantages and disadvantages of this method can be identified as follows.

Advantages

1. This is very useful in a test scenario where the best coding option out of a large number of coding options needs to be chosen. A significant time saving can be achieved compared with the pair comparison test where pairs with all the combinations need to be evaluated.

2. Due to the large time saving, more fine grained levels of quality can be evaluated with a larger number of observers.

3. Observers' viewing patterns can be monitored. For example an oscillating slider position may indicate uncertainty on the part of the observer.

Disadvantages

1. This method is not particularly suitable for other tests such as the Degradation Category Rating (DCR) test where every clip needs to be compared with a reference video clip.

2. When only a few video clips or coding options are evaluated, the transitions between clips only occur after the slider is moved by a large amount (e.g. when there are only two clips the transition will occur after the halfway mark). This confuses some users who expect the video quality to be changed after every slider movement. However, this problem is reduced when a large number of clips or options are evaluated.

This work was carried out during stage-1 of the research project where a study in to existing subjective and objective video quality methods was carried out. This method was proposed in particular as a faster alternative to the pair-wise comparison (PC) method.

12.2.2 Complexity Reduction of an H.264/AVC Encoder using Lagrangian Cost Estimation

This is a new complexity reduction algorithm for an H.264/AVC encoder based on macroblock skip prediction. Computational savings are achieved by identifying, prior to the motion estimation and mode decision process, the macroblocks that are likely to be skipped and therefore saving further computational processing of these macroblocks. The objective of the algorithm is to match the rate-distortion performance of the non-optimised baseline encoder while significantly reducing the computational complexity compared to the same. The skip macroblock prediction is made by estimating and comparing the Lagrangian rate-distortion cost of "coding" and "skipping" the macroblock before the macroblock is processed. If the estimated rate-distortion cost of "coding" the macroblock is more than the cost of "skipping", the macroblock is marked as skip (no further processing is carried out for this macroblock), otherwise the macroblock is coded as normal.

The rate-distortion cost of skipping the macroblock is evaluated as the distortion value because the number of bits associated with a skipped macroblock is assumed to be negligible. The distortion of the skipped macroblock can easily be calculated as the Mean Squared Error (MSE) between the original and the motion compensated prediction using the predicted motion vector for skip. However, the rate-distortion cost of coding the macroblock needs to be estimated because this is not available until the macroblock is actually been coded. The rate-distortion cost of coding is estimated as follows:

- The coded rate and distortion of the current macroblock is estimated from the rate and distortion values of the co-located macroblock in the previous frame.

- The Lagrange multiplier is adaptively estimated as a function of the quantisation parameter and video sequence statistics.

- Video sequence statistics are quantified using an activity factor, where the value of the activity factor is an estimation of the amount of motion and detail present

in a sequence. This concept of measuring and using an activity factor for adaptive modelling is successfully used in other recent algorithms [92].

The experimental results show that the algorithm achieves computational savings of 19-67%, depending on the source sequence, with no significant loss of rate-distortion performance compared to the non optimised baseline encoder. This algorithm is a novel contribution due to the following. This algorithm is based on a firm theoretical foundation in contrast to most existing heuristic algorithms. Unlike most algorithms available in literature, the performance of the complexity reduction algorithm does not depend on empirically obtained thresholds. This algorithm automatically adapts to different sequence statistics (different video sequences with a range of motion and detail) without the need for 'tuning' of any thresholds. Other advantages and disadvantages of this algorithm are summarised below.

Advantages

1. For high/moderate-high activity sequences such as Foreman and Carphone, significant reductions in computational complexity (more than 19%) can be achieved with only negligible loss in rate-distortion performance.

2. For low activity sequences, the rate-distortion performance is better than the un-optimised baseline encoder with complexity reductions of more than 50%.

Disadvantages

1. Complexity savings mainly depend on the activity of the sequence and are therefore unpredictable. The complexity saving is high for low activity (because more macroblocks can be skipped without adversely affecting the rate-distortion performance) and vice versa.

2. The Lagrange multiplier is calculated using an exponential function which might become costly to implement in an embedded-type platform. However, this problem can easily be eliminated by using a look up table (with small amount of additional memory) with pre-calculated values for a range of QP and activity levels.

3. Currently the algorithm only classifies "skip" and "code (all the other modes in one group)" and therefore there is potential for further complexity reductions by predicting other modes (see Future Work).

This algorithm forms the foundation for further research that resulted in the development of complexity control and complexity management algorithms, described in chapters 10 and 11.

12.2.3 Complexity Control of an H.264/AVC Encoder

This algorithm enables the computational complexity control of an H.264/AVC encoder. This algorithm is based on the complexity reduction algorithm described in chapter 9. The aim of the algorithm is to achieve a specified complexity target (specified as a percentage of macroblocks to be processed per frame) during encoding whilst maintaining good rate-distortion performance. Here, the Lagrangian cost estimation is extended to incorporate computational complexity. The prediction of skipped macroblocks is carried out by estimating the Lagrangian complexity-rate-distortion cost of "coding" and "skipping" a macroblock. The estimation of the rate and distortion for "code", the Lagrange multiplier associated with rate and distortion for "skip" are identical to the complexity reduction algorithm in chapter 9. Complexity control is achieved by updating the Lagrange multiplier associated with computational complexity at frame level, using a feedback algorithm based on a user-specified complexity target for the encoder. The simulation results show that using this algorithm, the encoder can achieve a specified average per-frame computational target. Therefore, this algorithm achieves scalable complexity control, which other existing algorithms have failed to demonstrably achieve so far. The advantages and disadvantages of this algorithm are listed below.

<u>Advantages</u>

1. This algorithm provides a flexible way of adaptively controlling the computational cost of an encoder whilst keeping good rate-distortion performance. This eliminates a drawback of the complexity reduction algorithm (chapter 9) where computational savings are dependant on the level of sequence activity.

2. Good rate-distortion performance is maintained as long as the target complexity reduction is similar to, or less than, the percentage of macroblocks skipped during normal encoding. For low activity sequences like Claire, for this case, the rate-distortion performance is better than the baseline encoder.

Disadvantages

1. The rate-distortion performance deteriorates when the complexity reduction target is significantly more than the percentage of macroblocks normally skipped by the encoder. Therefore it is difficult to achieve large computational reductions for high-activity sequences without adversely affecting the rate-distortion performance.

2. Although an average target per-frame complexity is achieved, there are variations in per-frame complexity at frame level. This implies that the feedback algorithm can be further developed in order to achieve more stable performance.

3. The Lagrange multiplier for complexity (complexity control parameter) is selected for a whole frame. Better performance may be achieved if the Lagrange multiplier is selected at macroblock level or for a group of macroblocks.

12.2.4 Complexity Management of a Real-time H.264/AVC Encoder

This is a new complexity management approach for an H.264/AVC encoder designed to maintain acceptable video coding performance (to maintain good perceived video quality) in a computation-constrained (power/processing resource constrained) environment. In a real-time computationally constrained situation, the managed complexity encoder maintains real-time operation based on the following hypothesis. Encoding at a higher/smoother frame rate whilst allowing a small loss in frame quality produces better perceived video quality than, encoding at a reduced frame rate whilst maintaining frame quality. The encoder is managed using two algorithms.

1. **Frame level control algorithm** – Maintains the target frame rate of the encoder by calculating a target encoding time for each frame based on real-time coding statistics such as input frame buffer level, frame waiting times and actual encoding times.

2. **Per-frame complexity control algorithm** – Receives a target encoding time for the current frame from the frame level control algorithm. The objective is to encode the frame within the allocated time by controlling the computational complexity of each frame. The Lagrangian complexity-rate-distortion cost estimation based complexity control algorithm is used with a new adaptive feedback control mechanism that controls encoding time rather than the percentage of skipped macroblocks.

These two algorithms are combined together so that a smooth video frame rate is maintained where possible without degrading the frame quality unacceptably. The trade-off between the frame rate and the maximum allowable distortion is achieved by limiting the maximum value of the Lagrange multiplier associated with complexity. The Lagrange multiplier for complexity controls the number of skipped macroblocks. Subjective video quality evaluation results show that the complexity management method consistently produces higher perceptual quality, outperforming the reference encoder, where frames are dropped in order to maintain real-time performance. These results prove the initial hypothesis.

<u>Advantages</u>

1. The complexity managed encoder produces higher frame rates with a small drop in frame quality and consistently out-performs the reference encoder in both:

    a. Computation constrained, constant bit rate applications: where the bit rate is only limited by an upper ceiling regardless of the available computational resources and,

    b. Computation **and** rate constrained applications: where the perceived video quality needs to be maximised for comparable rate and complexity constraints.

2. This is a flexible framework for managing computational complexity where other complexity control algorithms can also be used for the per-frame complexity control algorithm as long as a control parameter exists.

3. This method is likely to be particularly useful for encoders implemented on low-power handheld devices. Example applications include; (a) switching to a low power encoding mode to prolong the battery life while maximising the perceptual video quality for the selected power level and (b) maintaining perceptual quality of encoding while more power is diverted to background activities such as transmission (in a mobile video phone) to combat a sudden signal drop or interference.

Disadvantages

1. This algorithm achieves good computational management of the encoder. However, the performance is limited by the performance of the per-frame complexity control algorithm and therefore the disadvantages are inherited from it; for example, it is difficult to achieve higher frame rates for high activity sequences (see results in chapter 12) and therefore the gain in the perceptual quality diminishes with increasing activity.

2. The performance of these low-complexity/complexity control/complexity management algorithms are not evaluated when combined with established rate control algorithms.

12.3 Future Work

The algorithms developed during this research project were summarised and critically evaluated in the earlier section. This section presents some directions for further research, mainly aimed at reducing the disadvantages of the above algorithms and extending the algorithms to achieve better performance and flexibility.

1. The developed low complexity algorithms are based on a skip macroblock prediction with only one decision criteria for skip or code. If code is chosen, all the modes are evaluated as normal. However, it should be possible to group the coding modes into several logical groups (e.g. Intra modes, Inter16x16 only, Inter (16x8, 8x16, 8x8), Inter (8x4, 4x8, 4x4)) and carry out a hierarchical mode decision prediction based on the estimated rate-distortion cost of each mode group. This can be further developed to achieve complexity control by

quantifying the computational complexity of evaluating each mode group and estimating the Lagrangian complexity-rate-distortion cost.

2. Currently the Lagrange multiplier for complexity is determined for the whole frame. This can be extended up to macroblock or group of macroblocks level. This can be carried out by defining a perceptual quality map for each frame that classifies macroblocks according to perceptual importance. Then the complexity allocation for the frame can be distributed among the macroblocks according to perceptual importance. Experience with the current low complexity skip-prediction algorithms shows that perceptual quality should also be taken into account for the decision making process. For example, it was observed that small artefacts appearing in plain backgrounds (sometimes due to skipped macroblocks) have an adverse effect on perceptual quality although they are not significant in rate-distortion sense.

3. The algorithms presented so far are evaluated using fixed quantiser values. Experiments should be carried out to evaluate the performance of the low complexity algorithms combined with established rate control methods such as [57]. This will highlight compatibility/interoperability issues of combined rate and complexity control based on the above algorithms and may open further avenues for investigation.

4. The same principles can be used to further develop the rate-distortion optimisation strategy used in the H.264/AVC reference encoder [51]. Currently, during the RD optimised mode decision process, the Lagrange multiplier is modelled as a function of the quantisation parameter and therefore achieves varying degrees of performance (see rate-distortion results in chapter 7 and chapter 9) for different video sequences. The performance may be improved for all (or for a wide variety of sequences) sequences by extending the model for estimation of the Lagrange multiplier to incorporate video sequence statistics.

In addition to the above mentioned future research directions, it is planned to incorporate the complexity management approach into a real-time H.264/AVC software encoder. The complexity management approach is expected to produce video with high perceptual quality at reduced processing resource levels. In this case, the software encoder is initially

developed for a typical PC based video conferencing application. Therefore, it is expected to maintain good perceptual quality while processing resources are diverted to other applications.

12.4 Conclusion

The H.264/AVC video coding standard can deliver significantly improved compression efficiency and coding flexibility compared to other video coding standards. Due to increased compression efficiency and coding flexibility, H.264/AVC has the potential to enable new video services such as mobile video telephony and multimedia streaming over mobile networks. However, the performance gains of H.264/AVC come at a cost of significantly increased computational complexity and therefore the processing resources required to implement an H.264/AVC encoder in a power-constrained mobile platform are likely to be a major problem. The objective of this research project is to develop novel algorithms to manage the computational complexity of an H.264 encoder. These algorithms seek to effectively utilise the available computational resources to maximise the rate constrained video quality of the encoder.

The research project has been structured into four stages as illustrated in chapter 1. Every stage has been successfully completed. The work carried out and the algorithms developed have been presented in this thesis.

Stage-1

The stage-1 of the project involved evaluating the performance of the H.264/AVC encoder (chapter 7) with different coding parameters and identifying the main contributors to computational complexity. It was established that the motion estimation and mode decision process consumes the largest proportion of the processing resources during encoding. A critical review of existing low-complexity algorithms was carried out in chapter 7. An existing low-complexity macroblock skip prediction algorithm [75] was identified as a suitable candidate for H.264/AVC because the processing of the macroblocks predicted as skipped can be avoided and therefore the computational resources needed for motion estimation and subsequent processing for these macroblocks can be saved (chapter 8). This algorithm was applied to an H.264/AVC reference encoder. Experimental results showed that significant amount of computational complexity can be

saved with a small loss of rate-distortion performance by carefully selecting the thresholds that control the amount of macroblocks that are predicted as skipped.

Stage-2

A new complexity reduction algorithm for H.64/AVC encoder based on macroblock skip prediction was developed (Chapter 9) during stage-2. The skip prediction decision is made by estimating a Lagrangian rate-distortion cost function. This is a "threshold free" complexity reduction algorithm which adapts to different video sequence statistics using an adaptive model. The results indicate that the computational complexity of the encoding process is reduced by 19-67% with negligible loss of rate-distortion performance.

Stage-3

The complexity reduction algorithm was further developed (chapter 10) in order to enable control of the computational complexity of the encoding process at frame level. The skipped macroblocks are predicted using a Lagrangian complexity-rate-distortion cost. The results show that effective and scalable control of complexity can be achieved with minimum loss in rate-distortion performance.

Stage-4

A complexity management framework was developed for an H.264/AVC encoder (chapter 11) to effectively manage the computational complexity in a real-time computational constrained environment in order to maintain the perceived video quality of coded video. Subjective video quality measurement results show that the perceptual quality of video encoded using the managed complexity approach (higher frame rate with lower frame quality) is higher than the perceptual video quality of a reference encoder which drops frames in order to maintain real-time performance. This work achieves the main objective of this research project, which is to manage the computational complexity of an H.264/AVC encoder in order to maximise the rate constrained video quality by making effective use of the available processing resources.

In addition to achieving the main objective, a fast subjective video quality measurement technique was developed (chapter 6). The new technique enables the test subject to

interactively choose a preferred video clip from a set of alternative clips within a short period of time.

The main contributions to the body of knowledge in video coding can be summarised as: (a) The development of a new complexity reduction algorithm based on skip macroblock prediction using Lagrangian rate-distortion cost estimation, the modelling of the Lagrange multiplier associated with rate as a function of the quantisation parameter and video sequence statistics, the classification of video sequence statistics by an activity factor which is an estimation of the amount of motion and detail present in the sequence. (b) Demonstrating that effective control of computational complexity of the encoder can be achieved by extending the Lagrangian cost estimation to incorporate computational complexity and selecting the Lagrange multiplier associated with complexity for each frame. (c) The development of a complexity management framework to manage the computational resources utilised by an H.264/AVC encoder and demonstrating that higher perceptual quality can be achieved by encoding at a higher frame rate with some loss in frame quality as opposed to dropping frames in order to maintain real time performance. Finally, (d) the development of a new subjective video quality assessment method that uses user feedback to determine user preferences in a fraction of time compared with standard pair comparison test.

In contrast to other published work, these contributions are based on firm theoretical foundations, have minimal dependence on empirically obtained thresholds and are adaptive to changing sequence statistics. Additionally, the development of complexity control and complexity management algorithms address significant practical problems that have not been properly addressed by other published work. Novel contributions of this work may benefit applications like software video conferencing and mobile video recording/telephony, enabling automatic management of processing resources without degrading video compression performance. In conclusion, the novel algorithms developed and the original contributions made during this research project are likely to be particularly useful in implementing H.264/AVC standard compliant encoders on low power and processor-limited platforms.

References

[1] Apple Mac Video Conferencing - iChat AV, http://www.apple.com/macosx/features/ichat/.

[2] ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services," May 2003.

[3] S. Saponara, C. Blanch, K. Denolf, and J. Bormans, "The JVT advanced video coding standard: Complexity and performance analysis on a tool-by-tool basis," presented at IEEE Packet Video 2003, Nantes, France, April 2003.

[4] T. Stockhammer, M. M. Hannuksela, and T. Wiegand, "H.264/AVC in Wireless Environments," *IEEE Trans. Circuits and System. Video Technology*, vol. 13, no. 7, pp. 688-703, July 2003.

[5] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Weidi, "Video Coding with H.264/AVC: Tools, Performance and Complexity," in *IEEE Circuits and Systems Magazine*, vol. 4, 2004, pp. 7-28.

[6] H. S. Malvar, A. Hallapuro, and M. Karzewicz, "Low-complexity transform and quantization in H.264/AVC," *IEEE Trans. Cirduits and System. Video Technology*, vol. 13, pp. 598-603, July 2003.

[7] C. S. Kannangara, I. E. G. Richardson, M. Bystrom, Y. Zhao, J. Solera, A. MacLennan, and R. Cooney, "Low Complexity Skip Prediction for H.264 through Lagrangian Cost Estimation," *IEEE Trans. Circuits and System. Video Technology*, vol. 16, No. 2, pp. 202-208, February 2006.

[8] C. S. Kannangara, I. E. G. Richardson, M. Bystrom, Y. Zhao, J. Solera, A. MacLennan, and R. Cooney, "Complexity Reduction of H.264 using Lagrange Optimization Methods," presented at IEE VIE 2005, Glasgow, 4~6 April 2005.

[9] C. S. Kannangara and I. E. G. Richardson, "Computational Control of an H.264 Encoder through Lagrangian Cost Function Estimation," presented at VLBV 2005, Sardinia, Italy, 15-16 September 2005.

[10] C. S. Kannangara, I. E. G. Richardson, and A. J. Miller, "Computational Management of an H.264 Encoder," presented at Picture Coding Symposium 2006, Beijing, China, 24~26 April 2006.

[11] I. E. G. Richardson and C. S. Kannangara, "Encoder Control System." UK Patent application 0524663.2, December 2005.

[12] C. S. Kannangara, I. E. G. Richardson, and A. J. Miller, "Computational Complexity Management of a Real-time H.264/AVC Encoder," *IEEE Trans. Circuits and System. Video Technology*, Submitted for Review.

[13] I. E. G. Richardson and C. S. Kannangara, "Fast Subjective Video Quality Measurement with User Feedback," *IEE Electronic Letters*, vol. 40, No. 13, pp. 799-800, June 2004.

[14] I. E. G. Richardson and C. S. Kannangara, "Subjective Video Quality Measurement with User Feedback." European Patent Application EP20050252018, 2005.

[15] ITU-T Recommendation H.263, "Video coding for low bit rate communications," 1998.

[16] I. E. G. Richardson, "Colour Spaces," in *Video Codec Design: Developing Image and Video Compression Systems*. West Sussex: John Wiley & Sons Ltd, 2002, pp. 10-16.

[17] S. Westland, "Models of the visual system and their application to image-quality assessment," presented at AIC Colour 05 - 10th Congress of the International Colour Association, Spain, 8-13 May 2005.

[18] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, advantages, applications*. San Diego, CA: Academic Press, 1990.

[19] ITU T Rec. H.264 | ISO/IEC 14496 10 version 4, January 2005.

[20] R. C. Gonzalez and R. E. Woods, "Error Free Compression," in *Digital Image Processing*. New Jersey: Prentice Hall, 2002, pp. 440-459.

[21] R. Schafer and T. Sikora, "Digital video coding standards and their role in video communications," *Proceedings of the IEEE*, June 1995.

[22] I. E. G. Richardson, "Video Coding Standards: JPEG and MPEG/H.263 and H.26L," in *Video Codec Design: Developing Image and Video Compression Systems*. West Sussex: John Wiley & Sons Ltd, 2002, pp. 47-92.

[23] ISO/IEC 11172-2 MPEG1 Video, "Information technology-coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s - part 2: Video," 1993.

[24] ISO/IEC 13818-2 MPEG2 Video, "Information technology: generic coding of moving pictures and associated audio information: Video," 1995.

[25] ISO/IEC 14496-2, "Information technology - coding of audio-visual objects-part 2: Visual," 1998.

[26] ITU-T Recommendation H.261, "Video codec for audiovisual services at px64 kb/s," 1993.

[27] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits and System. Video Technology*, vol. 13, pp. 560-576, July 2003.

[28] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-L047d12, "Draft Text of H.264/AVC Fidelity Range Extensions Amendment," 12th Meeting: Redmond, WA, USA, 17-23 July 2004.

[29] G. J. Sullivan, P. Topiwala, and A. Luthra, "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions," presented at SPIE Int. Conf. Applications of Digital Image Processing, Denver, CO, August 2004.

[30] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-O013, "Performance Improved 4:4:4 Coding for MPEG4-Part10/H.264," 15th Meeting: Busan, KR, 16-22 April 2005.

[31] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-P017, "Advanced 4:4:4 Profile for MPEG4-Part-10/H.264," 16th Meeting: Poznan, PL, 24-29 July 2005.

[32] G. J. Sullivan, "The H.264/MPEG-4 AVC Video Coding Standard and its Deployment Status," presented at SPIE/VCIP 2005, Beijing, China, July 2005.

[33] Technology Brief: QuickTime and MPEG-4, http://images.apple.com/quicktime/pdf/H264_Technology_Brief.pdf.

[34] 3 rd Generation Partnership Project, http://www.3gpp.org.

[35] DVD Forum, http://www.dvdforum.org.

[36] Blu-ray Disc Association, http://www.blu-raydisc.com.

[37] Digital Video Broadcasting Project (DVB), www.dvb.org.

[38] M. Karczewicz and R. Kurceren, "The SP- and SI-Frames Design for H.264/AVC," *IEEE Trans. Circuits and System. Video Technology*, vol. 13, No. 7, pp. 637-644, July 2003.

[39] I. E. G. Richardson, "Transform and Quantisation," in *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*. West Sussex: John Wiley & Sons Ltd, 2003, pp. 187-194.

[40] D. Marpe, T. Wiegand, and S. Gordon, "H.264/MPEG4-AVC Fidelity Range Extensions: Tools, Profiles, Performance, and Application Areas," presented at International Conference on Image Processing, Genova, 11-14 September 2005.

[41] E. R. Dougherty and P. A. Laplante, "Hadamard Transform," in *Introduction to Real-Time Imaging*, vol. TT19. Washington: SPIE Optical Engineering Press, 1995, pp. 60-67.

[42] D. Marpe, H. Schwarz, and T. Wiegand, "Context-Based Adaptive Binary Arithmatic Coding in the H.264/AVC Video Compression Standard," *IEEE Trans. Circuits and System. Video Technology*, vol. 13, No. 7, pp. 620-636, July 2003.

[43] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive Deblocking filter," *IEEE Trans. Circuits and System. Video Technology*, vol. 13, No. 7, pp. 614-619, July 2003.

[44] J. Lee, S. Moon, and W. Sung, "H.264 Decoder Optimization Exploiting SIMD Instructions," presented at IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS), Taiwan, December 2004.

[45] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," in *IEEE Signal Processing Magazine*, vol. 15, no. 6, November 1998, pp. 23-50.

[46] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG, "JM7.3 Test Model CODEC," http://iphome.hhi.de/suehring/tml/.

[47] ITU-T Study Group 16, "Q15-D-65d1,Video Codec Test Model, Near Term, Version 10 (TMN10) Draft 1," Finland, April 1998.

[48] H. Everett, "Generalized Lagrange multiplier method for solving problems of optimal allocation of resources," *Operations Research*, vol. 11, pp. 399-417, May/June 1963.

[49] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of Quantizers," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-36, pp. 1445-1453, Sept. 1988.

[50] G. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," in *IEEE Signal Processing Magazine*, vol. 15, no. 6, November 1998, pp. 74-90.

[51] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and compression of video coding standards," *IEEE Trans. Circuits and System. Video technology*, vol. vol. 13, pp. 688-703, July 2003.

[52] M. Ghandi and M. Ghanbari, "A Lagrangian optimized rate control algorithm for the H.264/AVC encoder," presented at IEEE International Conference on Image Processing, Singapore, October 2004.

[53] T. Wiegand, M. Lightstone, D. Mukherjee, T. G. Campbell, and S. K. Mitra, "Rate-Distortion Optimized Mode Selection for Very Low Bit Rate Video Coding and the Emerging H.263 Standard," *IEEE Trans. Circuits and System. Video Technology*, vol. 6, no. 2, pp. 182-190, April 1996.

[54] ITU-T Study Group 15, "Video Codec Test Model TMN5," January 1995.

[55] T. Wiegand and B. Girod, "Rate-Constrained coder Control," in *Multi-Frame Motion-Compensated Prediction For Video Transmission*. Massachusetts: Kluwer, 2001.

[56] T. Wiegand and B. Girod, "Lagrange Multiplier Selection in Hybrid Video Coder Control," presented at IEEE International Conference on Image Processing, Thessaloniki, Greece, 2001.

[57] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-H017, "Proposed Draft of Adaptive Rate Control," Geneva, May 2003.

[58] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-K049, "Text Description of Joint Model Reference Encoding Methods and Decoding Concealment Methods," Munich, Germany, March 2004.

[59] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG, "JM7.4 Test Model CODEC," http://iphome.hhi.de/suehring/tml/.

[60] ISO/IEC MPEG and ITU-T VCEG Joint Video Team, "JVT-Q042, Revised H.264/MPEG-4 AVC Reference Software Manual," 17th Meeting: Nice, FR, 14-21 October 2005.

[61] ITU-T Recommendation P.910, "Subjective Video Quality Assessment Methods for Multimedia Applications," 1999.

[62] ITU-R Recommendation BT.500-10, "Methodology for the Subjective Assessment of the Quality of Television Pictures," 2000.

[63] Y. Zhong, I. E. G. Richardson, A. Sahraie, and P. McGeorge, "Qualitative and quantitative assessment in video compression," presented at 12th European Conference on Eye Movements, Dundee, Scotland, August 2003.

[64] I. E. G. Richardson, "Video Quality," in *Video codec Design, developing Image and Video Compression Systems*. West Sussex: John Wiley & Sons, 2002, pp. 16-23.

[65] C. J. v. d. B. Lambrecht and O. Verscheure, "Perceptual Quality Measure using a Spatio-Temporal Model of the Human Visual System," presented at SPIE, San Jose, CA, 1996.

[66] K. T. Tan and M. Ghanbari, "A Multi-Metric Objective Picture-Quality Measurement Model for MPEG Video," *IEEE Trans. Circuits and System. Video Technology*, vol. 10, No. 7, pp. 1208-1213, October 2000.

[67] A. Bouch, A. Watson, and M. A. Sasse, "QUASS - A Tool for Measuring the Subjective Quality of Real-Time Multimedia Audio and Video," presented at HCI '98, Sheffield, September 1998.

[68] A. Watson and M. A. Sasse, "Measuring Perceived Quality of Speech and Video in Multimedia Conferencing Applications," presented at ACM Multimedia '98, Bristol, UK, September 1998.

[69] Video Coding Experts Group, ITU, "Video Codec Test Model, Version 8 (TMN8)," June 1997.

[70] S. Mietens, "Motion Estimation," in *Complexity Scalable MPEG Encoding*. Eindhoven: Technische Universitiet Eindhoven, 2004, pp. 71-85.

[71] M. Gallant, G. Cote, and F. Kossentini, "An Efficient Computation-constrained Block-Based Motion Estimation Algorithm for Low Bit Rate Video Coding," *IEEE Trans. Image Processing*, vol. 8, pp. 1816-1823, December 1999.

[72] I. E. G. Richardson and Y. Zhao, "Adaptive Management of Video Encoder Complexity," *Real-Time Imaging*, vol. 8, No. 4, pp. 291-301, August 2002.

[73] I.-M. Pao and M.-T. Sun, "Modeling DCT Coefficients for Fast video Encoding," *IEEE Trans. Circuits and System. Video Technology*, vol. 9, No. 4, pp. 608-616, *June 1999.*

[74] I. E. G. Richardson and Y. Zhao, "Adaptive Algorithms for Variable Complexity video Coding," presented at IEEE International Conference on Image Processing, Thessaloniki, Greece, October 2001.

[75] Y. Zhao and I. E. G. Richardson, "Macroblock Classification for Complexity Management of Video Encoders," *Signal Processing: Image Communication*, pp. 801-811, 2003.

[76] Y. Zhao and I. E. G. Richardson, "Macroblock Classification for Video Encoder Complexity Management," presented at Picture Coding Symposium 2003, France, 23-25 April 2003.

[77] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-F017, "Fast Integer Pel and Fractional Pel Motion Estimation for JVT," 6th Meeting: Awaji, Island, JP, 5-13 December 2002.

[78] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-P021, "Improved and simplified fast motion estimation for JM," 16th Meeting: Pozan, Poland, 24-29 July 2005.

[79] H. Chung, D. Romacho, and A. Ortega, "Fast Long-term Motion Estimation for H.264 using Multiresolution Search," presented at IEEE International Conference on Image Processing, Barcelona, Spain, September 2003.

[80] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-G013, "Fast Mode Decision for Intra Prediction," 7th Meeting: Pattaya, Thailand, 7-14 March 2003.

[81] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, D. Wu, and S. Wu, "Fast Mode Decision Algorithm for Intraprediction in H.264/AVC Video Coding," *IEEE Trans. Circuits and System. Video Technology*, vol. 15, No. 7, pp. 813-822, July 2005.

[82] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-I020, "Fast INTER Mode Selection," 9th Meeting: San Diego, 2-5 September 2003.

[83] D. Wu, F. Pan, K. P. Lim, S. Wu, Z. G. Li, X. Lin, S. Rahardja, and C. C. Ko, "Fast Intermode Decision in H.264/AVC Video Coding," *IEEE Trans. Circuits and System. Video Technology*, vol. 15, No. 6, pp. 953-958, July 2005.

[84] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-J033, "Fast Mode Decision for H.264," 10th Meeting: Waikoloa, Hawaii, 8-12 December 2003.

[85] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG Doc. JVT-K021, "Fast Mode Decision for B Slice," 11th Meeting: Munich, DE, 15-19 March 2004.

[86] H. Kim and Y. Altunbasak, "Low-complexity macroblock mode selection for H.264/AVC encoders," presented at IEEE International Conference on Image Processing, Singapore, October 2004.

[87] A. Tanizawa, S. K. Chujoh, and Y. Kikuchi, "A Study on fast rate-distortion optimized coding mode decision for H.264," presented at IEEE International Conference on Image Processing, Singapore, October 2004.

[88] A. C. Yu and G. R. Martin, "Advanced block size selection algorithm for inter frame coding in H.264/MPEG-4 AVC," presented at IEEE International Conference on Image Processing, Singapore, October 2004.

[89] Q. Dai, D. Zhu, and R. Ding, "Fast Mode decision for Inter Prediction in H.264," presented at IEEE International Conference on Image Processing, Singapore, October 2004.

[90] J. Chen, Y. Qu, and Y. He, "A Fast Mode Decision Agorithm in H.264," presented at PCS 2004, SanFransisco, CA, December 2004.

[91] J. Xin and A. Vetro, "Fast Mode Decision for Intra-only H.264/AVC Coding," presented at Picture Coding Symposium 2006, Beijing, China, 24-26 April.

[92] I. E. G. Richardson, M. Bystrom, and Y. Zhao, "Fast H.264 Skip Mode Selection Using an Estimation Framework," presented at Picture Coding Symposium 2006, Beijing, China, 24-26 April 2006.

[93] Joint Video Team (JVT) of ISO/IEC MPEG & ITU T VCEG, "JM9.0 Test Model CODEC," http://iphome.hhi.de/suehring/tml/.

[94] C. S. Kannangara, I. E. G. Richardson, and Y. Zhao, "H.264 Complexity Management using Macroblock Skip-Mode Prediction," presented at PGNet, Liverpool, 28~29 June 2004.

[95] G. M. Schuster and A. K. Katsaggelos, *Rate-Distortion Based Video Compression*: Kluwer Academic Publishers, 1997.

[96] G. Ghinea and J. P. Thomas, "QoS Impact on User Perception and Understanding of Multimedia Video Clips," presented at Proceedings of the sixth ACM international conference on Multimedia, Bristol, United Kingdom, 13~16 September 1998.

[97] J. D. McCarthy, M. A. Sasse, and D. Miras, "Sharp or Smooth? Comparing the effects of quantization vs. frame rate for streamed video," presented at SIGCHI conference on Human factors in computing systems, Vienna, Austria, April 20-24, 2004.

[98] M. Jackson, A. H. Anderson, R. McEwan, and J. Mullin, "Impact of Video Frame Rate on communicative Behaviour in Two and Four Party Groups," presented at Proceedings of the 2000 ACM conference on Computer supported cooperative work, Philadelphia, Pennsylvania, United States, 2000.

[99] R. T. Apteker, J. A. Fisher, V. S. Kisimov, and H. Neishlos, "Video Acceptability and Frame Rate," *IEEE Multimedia*, vol. 2, pp. 32-40, 1995.

Bibliography

1. E. R. Dougherty and P. A. Laplante, *Introduction to Real-Time Imaging*, vol. TT19. Washington: SPIE Optical Engineering Press, 1995.

2. R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. New Jersey: Prentice Hall, 2002.

3. Z.-N. Li and M. S. Drew, *Fundamentals of Multimedia*. Upper Saddle River, NJ 07458: Pearson Prentice Hall, 2004.

4. D. C. Montgomery, G. C. Runger and N. F. Hubele, *Engineering Statistics*. New York: John Wiley & Sons Inc, 1998.

5. K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, advantages, applications*. San Diego, CA: Academic Press, 1990.

6. I. E. G. Richardson, *Video Codec Design: Developing Image and Video Compression Systems*. West Sussex: John Wiley & Sons Ltd, 2002.

7. I. E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*. West Sussex: John Wiley & Sons Ltd, 2003.

8. T. Wiegand and B. Girod, *Multi-Frame Motion-Compensated Prediction For Video Transmission*. Massachusetts: Kluwer, 2001.

9. Y. Zhao, *Complexity Management of Video Encoders*, PhD Thesis, The Robert Gordon University, March 2004.

10. S. Mietens, *Complexity Scalable MPEG Encoding*, Eindhoven: Technische Universitiet Eindhoven, 2004

Appendix A: List of Publications

A.1 Journal Publications

1. C. S. Kannangara, I. E. G. Richardson, M. Bystrom, Y. Zhao, J. Solera, A. MacLennan & R. Cooney, "Low Complexity Skip Prediction for H.264 through Lagrangian Cost Estimation," *IEEE Trans. Circuits and System. Video Technology*, Vol. 16, Number 2, February 2006, pp. 202-208.

2. I. E. G. Richardson and C. S. Kannangara, "Fast Subjective video Quality Measurement with User Feedback," *IEE Electronic Letters*, Vol. 40, Number 13, June 2004, pp. 799-800.

3. C. S. Kannangara, I. E. G. Richardson and A. J. Miller, "Computational Complexity Management of Real-time H.264/AVC Encoder" submitted to *IEEE Trans. Circuits and System. Video Technology*.

A.2 Conference Publications

1. C. S. Kannangra, I. E. G. Richardson and A. J. Miller, "Computational Management of an H.264 Encoder," PCS 2006, Beijing, China, 24~26 April 2006.

2. C. S. Kannangara and I. E. G. Richardson, "Computational Control of an H.264 Encoder through Lagrangian cost Function estimation," VLBV 2005, Sardinia, Italy, 15-16 September 2005.

3. C. S. Kannangara, I. E. G. Richardson, M. Bystrom, J. Solera, Y. Zhao, A. MacLennan & R. Cooney, "Complexity Reduction of H.264 using Lagrange Optimization Methods," IEE VIE 2005, Glasgow, 4~6 April, 2005.

4. C. S. Kannangara, I. E. G. Richardson and Y Zhao, "H.264 Complexity Management using Macroblock Skip-Mode Prediction," PGNet, Liverpool, 28~29, June 2004.

A.3 Patents

1. I. E. G. Richardson and C. S. Kannangara, "Encoder control system", UK Patent application 0524663.2, December 2005.

2. I. E. G. Richardson and Sampath Kannangara, "Subjective Video Quality measurement with User Feedback," European Patent application, EP20050252018, 2005.

Appendix B: H.264 Decoder Multi-threading for Real-time HD Decoding

B.1 Introduction

A software H.264/AVC decoder which uses multiple processing threads in a multi-processor platform is implemented. The objective is to employ parallel processing in order to increase the decoding speed. For example, in a dual processor platform, the multi-threaded decoder is expected to be as twice as fast as the single threaded decoder. This implementation is targeted for real-time decoding of High-Definition (HD) content. This work has been carried out at BT Group Chief Technology Office (Multimedia Analysis and Coding) as a placement project.

B.2 Background

The decoding of H.264/AVC coded HD video requires a significant amount of computational resources. This is mainly due to (a) the complexity of the decoding process, especially for Main and High profiles, and (b) the high macroblock decoding rate required due to the high resolution and frame rate of HD video. The resolutions, scanning method and the refresh rates of HD video formats are shown in the following table. For example, a 720p HD format picture contains 3600 (1280x720/256) macroblocks.

Table B.1: High Definition Formats

Standard	Resolution (H x V)	Frame rates	Scanning method
720p	1280x720	23.98, 29.97, 59.94, 24, 30, 60, 25	Progressive
1080p	1920x1080	23.98, 29.97, 24, 30, 25	Progressive
1080i	1920x1080	25(50i), 29.97(59.94i), 30(60i)	Interlaced

The H.264/AVC codec software developed by the Multimedia analysis and Coding group at BT (BT H.264 Codec) supports a range of profiles and levels (Baseline, Main, Extended and High profiles). The software is developed using C/C++ programming languages. The BT H.264 decoder has been developed to use a single processing thread

for the entire decoding process. Although this implementation is efficient on a single processor platform, it fails to utilise the processing resources of additional processors in dual or multi-processor platforms. Further, the single threaded decoder does not achieve real-time decoding frame rates for coded HD video (2.8GHz, Pentium IV processor). Therefore, this work is carried out to modify the BT H.264 decoder to use multiple processing threads in order to utilise all the available processors in a multi-processor platform. It is expected that the multi-threaded implementation will double the decoding speed of a single-threaded decoder in a dual-processor platform and enable real-time HD decoding.

B.3 Decoder Multi-threading

The multi-threaded decoder is implemented by modifying the existing BT H.264 decoder. Two alternative multi-threaded decoder implementations are tested. They are:

1. Implementing different processing threads for parsing, motion compensation and loop filtering processes of macroblocks.

2. Implementing multiple processing threads to decode slices in parallel (thread-per-slice).

The following sections present overviews and the results of the above two methods.

B.4 Separate Threads for Parsing, Motion Compensation and De-blocking Filter

This implementation separates the parsing, motion compensation and de-blocking filter functions into three distinct processes. It was observed that, in general each process consumes one third of the total processing resources. The figure B.1 shows a simplified flow chart of this method. The main decoding function retains the overall control of the program, whilst the decoding threads for parsing macroblock data, motion compensation and de-blocking filter are started and then wait for slice data. First, the parsing thread starts parsing the macroblock data in each slice. The motion compensation is started as soon as a macroblock is available after parsing. The de-blocking filter process needs to wait until the macroblock below the current macroblock position (1 row, for MBAFF: 2 rows) to be reconstructed before starting the loop filter process in case the below

macroblock is intra predicted. This is because the intra prediction is performed using unfiltered neighbouring pixels (See figure B.2).

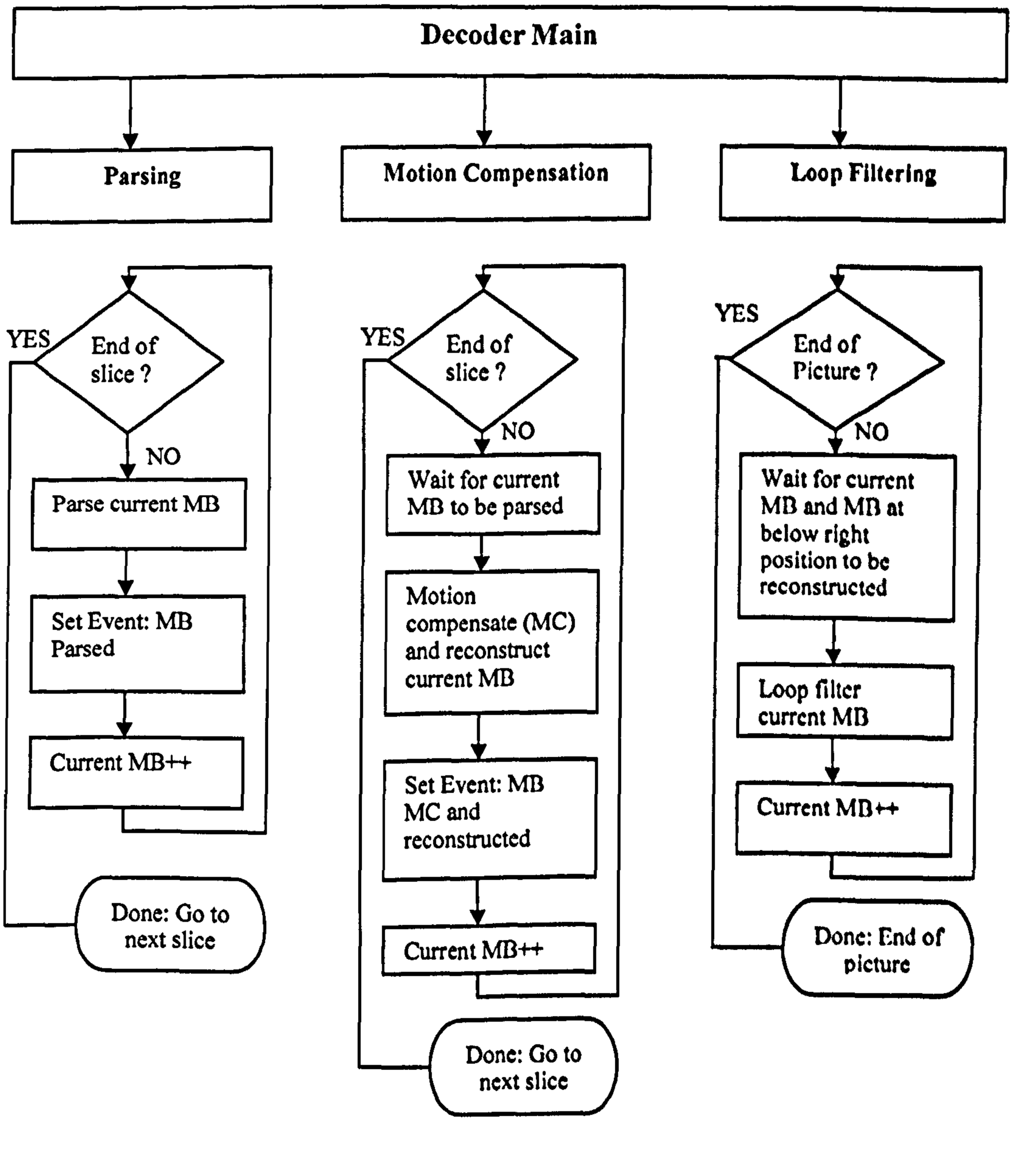


Figure B.1: Flow chart for implementing separate processing threads for parsing, motion compensation and loop filtering.

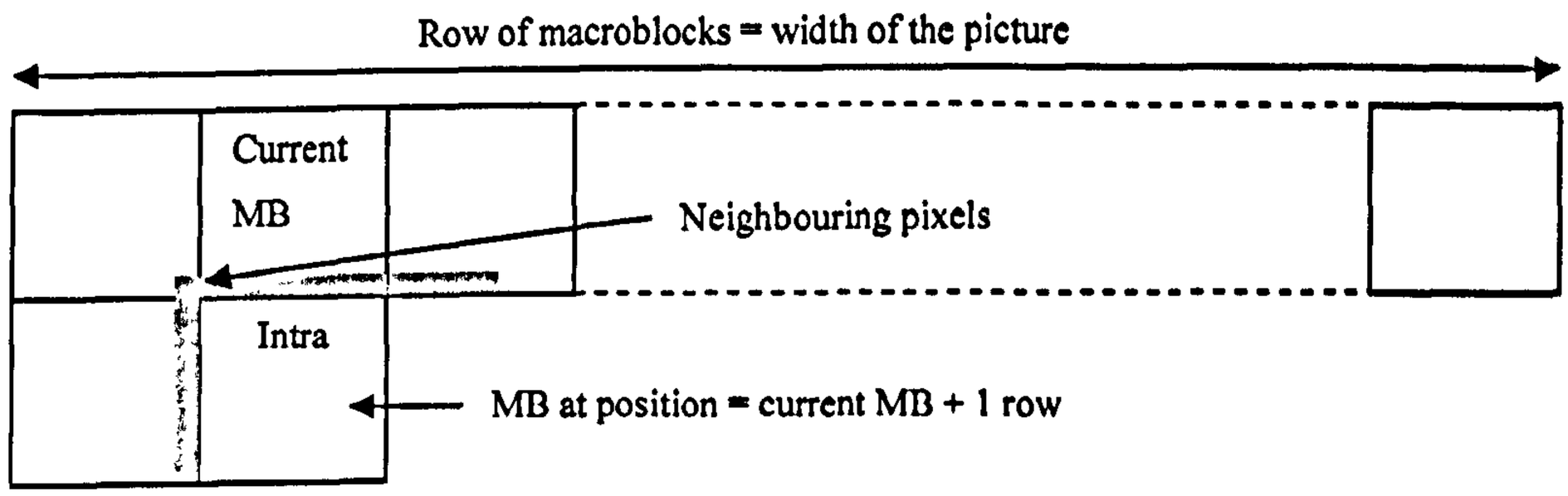


Figure B.2: Delaying of de-blocking filter to accommodate intra predicted macroblocks.

Advantages

- The maximum processor utilisation can be achieved with two or more processors because the processing threads are waiting for up-stream processes to be completed at macroblock level and the processing stages of the macroblocks are carried out as soon as data are available.

- The efficiency of this method does not depend on the slice structure of the coded video bit stream.

Disadvantages

- The synchronisation events need to be set for the completion of processing stages after every macroblock. Setting up each event costs approximately 1000 to 3000 processor cycles, whereas, parsing and motion compensation processes consume approximately 10,000 to 20,000 processor cycles. Therefore setting up and waiting for the synchronisation events consume a significant proportion of total processing resources.

- Typically, process scheduling among multiple processors is handled by the operating system (in this case Microsoft Windows). With this implementation the three processing threads are sometimes scheduled on a macroblock by macroblock basis among the two processors in a dual-processor test platform. This causes a major performance bottleneck because the usefulness of the processor cache is diminished due to repeated reloading, resulting in memory access delays.

Results

This multi-threaded decoder implementation is tested using a 2.8GHz Intel Pentium IV dual processor PC. The decoding time of the multi-threaded implementation is compared with the decoding time of the original single treaded implementation of the BT H.264 decoder. The results show only a 10-17% reduction in coding time. This is mainly due to the disadvantages discussed above.

The implementation and testing of the above method indicates that a new approach should be taken to achieve a significant reduction in processing time. The new approach is designed to use only a limited number of synchronisation events and allows longer execution of each thread without frequent re-scheduling.

B.5 Separate Decoding Threads for Decoding Each Slice (thread-per-slice)

This approach assumes that a coded picture (especially a *HD* size picture) consists of more than one slice to avoid large packet or NAL unit sizes. In this implementation, multiple decoding process threads (decoders) are initialised, typically depending on the number of processors in the system. Each decoder is given a slice by the main process to be decoded in parallel because the slices can be decoded independently to each other. The relationship between the components and the handling of different types of NAL units, with two decoding threads for a dual processor platform are shown in figure B.3. The in-loop de-blocking filter process (shown separately) is carried out using only one of the decoders for the whole picture, after decoding the last slice (see details below).

The method can be summarised as follows. Multiple decoder objects are constructed and initialised from the main decoding function (which controls the core decoder program) along with the critical sections and events to handle the decoders. The NAL units are extracted from the bit stream (or packet stream) by the application layer and they are passed on to the main decoding function. The NAL units containing sequence parameter sets (SPS) and picture parameter sets (PPS) are decoded by all the decoders and the decoders keep separate copies of the data. The NAL units containing slice data are given to available decoders such that multiple slices belonging to the same picture are decoded in parallel. However, the slice header is parsed as a critical section (one slice header at a time) in order to track the slices for ASO and error concealment. All the decoders share a

common decoded picture buffer apart from the reference picture list variables that may be changed at slice level during re-mapping. The common data structure "picture MB control list" contains the macroblock data for the whole picture. However, each decoder accesses only the macroblocks belonging to its current slice.

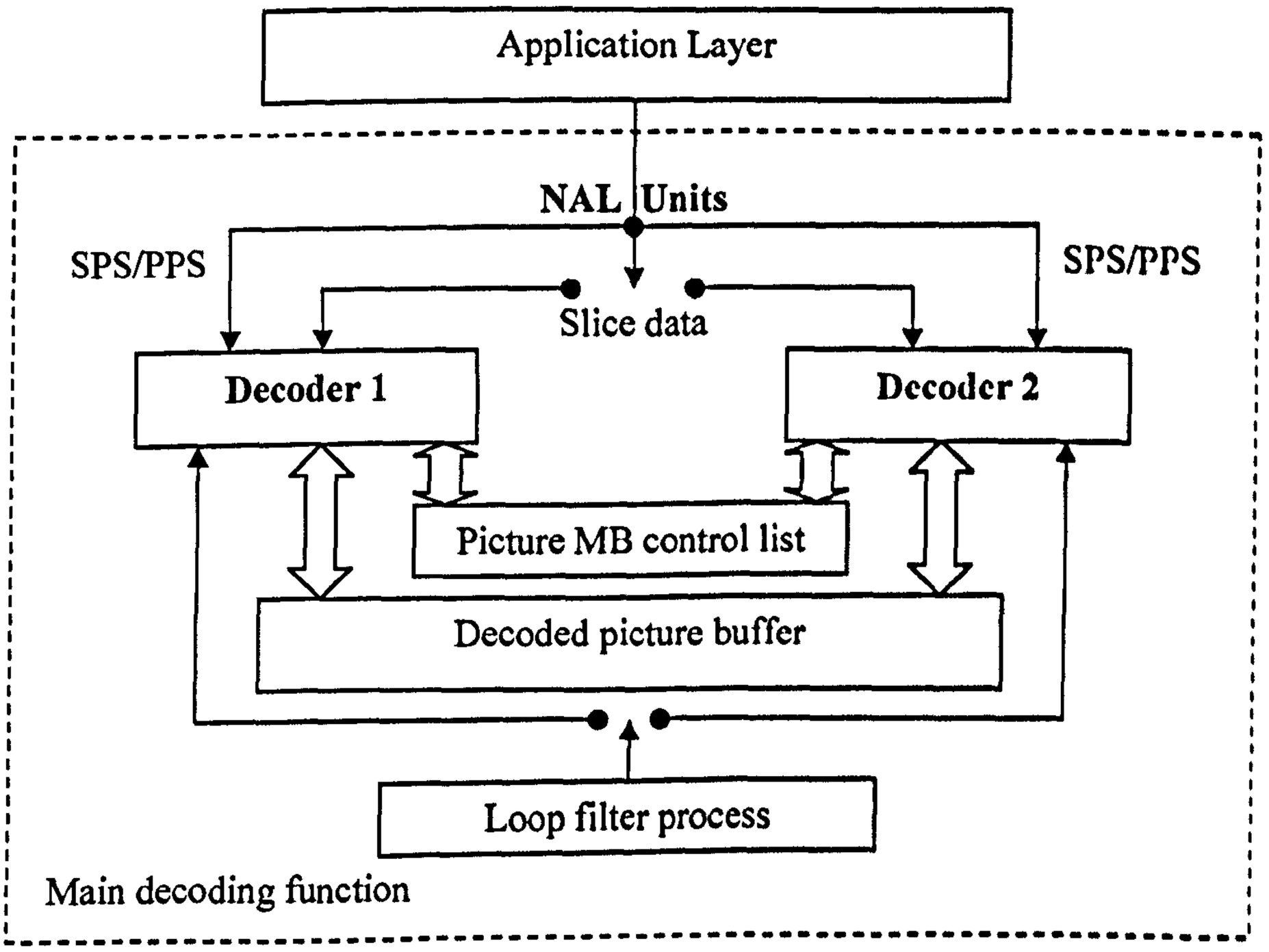


Figure B.3: H.264/AVC decoder with thread-per-slice implementation.

De-blocking filter

The de-blocking filter can only be applied in raster scan order. Typically, the de-blocking filter process consumes less processing time than the slice decoding process. Therefore the filter process is started by using the first available decoder after decoding all the slices of the current picture. The main decoder function probes into the slice header before assigning a decoding thread to decode the slice and if it is found to be the first slice of the next picture, the first available free decoder thread is used to start the de-blocking filter process for the current picture.

Once the de-blocking filter is started, it checks whether each macroblock is ready to be filtered by testing whether the current macroblock and the macroblock below at position = 'current MB + 1 row' (if MBAFF: current MB + 2 rows) is reconstructed. If not, the de-

blocking filter waits for another decoder to finish decoding a slice before checking again and continuing. This is because events are handled at slice level rather than macroblock level in order to minimise the processing overhead.

Advantages

- Most HD bit streams contain more than one slice per picture due to limitations on maximum packet size. Therefore, multiple decoding threads can be used to decode the slices in parallel.

- Processor cache reloads are minimised due to uninterrupted decoding of slices by the decoding threads.

- The synchronisation events are handled at slice level rather than macroblock level, for example an event is set after decoding a slice in contrast to a macroblock. Therefore synchronisation overhead is minimised.

Disadvantages

- Coded H.264 bit streams of video with smaller picture sizes may have only one slice per picture. In this case the multi-threaded decoder may run slower than the single threaded decoder.

- It is difficult to balance the load between processors to minimise the processor idle time. This is due to two reasons;

    - Different slices may consume different numbers of processor cycles to decode and the number of slices may not match the number of decoders.

    - The de-blocking filter is started using the first available free decoding thread after assigning the last slice of the current picture to a decoder. Therefore, other processor(s) may run idle towards the end of the de-blocking filter process.

Therefore the total processor load for the dual processor test platform does not exceed more than 90% of the total processing capacity.

Results

The performance of the thread-per-slice multi-thread decoder with two decoding threads is tested for different video sequences using a 2.8GHz Intel Pentium IV dual processor PC. The results in table B.2 compare the decoding performance of the multi-threaded decoder with the original single thread decoder. The video sequences are coded without using the de-blocking filter. Table B.3 compares the performance for sequences coded using the de-blocking filter. These results show that the decoding time can be reduced to up to 55% of the original single threaded decoding time for high definition video sequences.

Table B.2: Multi-threaded decoder performance – Dual processor platform, 2 decoding threads and without de-blocking filter.

Sequence	Decoding time on a dual processor PC (ms)		Decoding Time % $(\frac{MultiThread}{SingleThread})$
	Single Thread	Multi Thread (2 decoders)	
Foreman (2 slices, CIF)	4796	3375	70%
Park-run (7 slices, HD)	27578	15344	56%
Shields (2 slices, HD)	40953	22719	55%
Squirrel (Baseline, 2slices, HD)	51750	28296	55%
Squirrel (Main, 2slices, HD)	101953	57172	56%

Table B.3: Multi-threaded decoder performance – 2 decoder threads, 2 processors and with de-blocking filter.

Sequence	Decoding time on a dual processor PC (ms)		Decoding Time % $(\frac{MultiThread}{SingleThread})$
	Single Thread	Multi Thread (2 decoders)	
Susie (MBAFF, SD)	9328	5985	64%
Park-run (2Slices,Main profile, HD)	126218	70843	56%
Park-run (3Slices, Main, HD)	123640	68281	55%
Mobile-calendar (HD)	147140	81093	55%

The effect of performing the filter process separately does not affect the overall coding time reductions. This may be because the filter processing time is offset by varying decoding times of different slices.

Table B.4 shows the real time decoding performance of 720p HD bit streams. The results show that the decoder can achieve a decoding speed of nearly 30 frames per second for Baseline profile and at lower bit rates in Main profile.

Table B.4: Average frame rates achieved by multi thread (2 decoders) decoding of HD 720p H.264 bit streams, using a 2.8GHz Pentium 4 Dual processor PC.

Sequence	Bit Rate kbps	CABAC	Average Frame rate
1. Squirrel (Baseline, 2Slices)	9000	no	29.99
2. Squirrel (Main, 2Slices)	9000	yes	19.78
3. Park-run (High profile, 2 Slices)	7000	yes	22.00
4. Shields (Main, 2 Slices)	18,000	yes	24.99
5. Shields (Main, 2 slices)	1430	yes	27.23
5. Mobile-calendar (Main, 2 slices)	8000	yes	21.74
6. Mobile-calendar (Main, 2 slices)	1450	no	30.35

B.6 Conclusion

A software H.264 decoder which uses multiple decoding threads to enable parallel processing in a multi-processor platform is implemented. Two approaches are tested; (a) separating and using multiple processing threads for parsing, motion compensation and loop filter process, (b) parallel decoding of slices using separate decoding threads (thread-per-slice). The experimental results show that the thread-per-slice approach achieves significant reductions in decoding time in a dual processor platform (the decoding speed is almost doubled with up to 45% reduction in decoding time) and enables real-time performance for decoding HD sequences.

B.7 Acknowledgements

This work has been carried out as a placement project at British Telecom (BT) using the H.264 software codec developed by the Multimedia Analysis and Coding Group at BT

Group Chief Technology Office. Special thanks go to Mike Nilsson (BT) for providing supervision, guidance and support during the project and to Dr. Steve Appleby (BT) for supporting and for providing the opportunity to carry out the project.

Appendix C: Statistical Analysis

C.1 Objective

The objective of this statistical analysis is to investigate whether there is evidence to indicate that the two test methods, UFQ and PC, does not produce the same results. That is, to investigate whether the UFQ method produces comparable results to the PC method.

C.2 Data

Subjective tests using the UFQ method and a standard PC method are carried out for two video sequences (mobile and violin) at different frame rates. The results for the mobile sequence are shown in Table C.1. The mean preference and the standard deviation for the two methods are calculated (see Table C.2). The subjective test results and the statistical data for the Violin sequence are shown in Table C.3 and Table C.4.

Table C.1: Mobile – Subjective test results for UFQ and PC methods

UFQ Method		PC Method	
Frame rate (x)	Frequency of preference	Frame rate (x)	Frequency of preference
7.5	5	7.5	9
10	3	10	4
15	2	15	1
30	1	30	1

Table C.2: Mobile – Statistical data

Test method	Mean ($\bar{x}$)	Standard deviation (s)	Sample size (n)
UFQ	11.591	6.734	11
PC	10.167	5.860	15

Note: sample sizes differ because some users gave equal preference to some frame rates in PC method

Table C.3: Violin – Subjective test results for UFQ and PC methods

UFQ Method		PC Method	
Frame rate (x)	Frequency of preference	Frame rate (x)	Frequency of preference
6.2	6	6.2	7
8.33	5	8.33	6
12.5	0	12.5	2
25	0	25	2

Table C.4: Mobile – Statistical data

Test method	Mean ($\bar{x}$)	Standard deviation (s)	Sample size (n)
UFQ	7.168	1.112	11
PC	9.905	6.022	17

Note: sample sizes differ because some users gave equal preference to some frame rates in PC method

C.3 Hypothesis Test

Hypothesis: For a particular set of video sequence, the population mean preference of UFQ method (μ_1) is equal to the population mean preference of the PC method (μ_2). The alternative hypothesis is considered to be two sided ($\mu_1 \neq \mu_2$).

Therefore,

Null Hypothesis is $H_0 : \mu_1 = \mu_2$ or $\mu_1 - \mu_2 = 0$

Alternative Hypothesis is $H_1 : \mu_1 \neq \mu_2$

The following assumptions were made:

- The two populations are normally distributed
- Population standard deviations (σ_1, σ_2) are unknown

- Population standard deviations may not be equal ($\sigma_1 \neq \sigma_2$).

Because fewer than 30 samples are available, the hypothesis tests and confidence intervals are based on the 'Student-t' distribution (see: D. C. Montgomery, G. C. Runger and N. F. Hubele, *Engineering Statistics*. New York: John Wiley & Sons Inc, 1998).

The test statistic (for $\sigma_1 \neq \sigma_2$) is given by:

$$t_0^* = \frac{\overline{X}_1 - \overline{X}_2 - \Delta_0}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \qquad \text{(C-1)}$$

$\Delta_0 = 0$ Since $H_0 : \mu_1 - \mu_2 = 0$

Where, $\overline{X}_1$ & $\overline{X}_2$ are sample means and, s_1 & s_2 are sample standard deviations for the two samples respectively. Sample sizes are given by n_1 and n_2.

The test statistic is approximately distributed as 'Student-t' where "degrees of freedom" is given by:

$$\nu = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{\left(s_1^2/n_1\right)^2}{n_1+1} + \frac{\left(s_2^2/n_2\right)^2}{n_2+1}} - 2 \qquad \text{(C-2)}$$

For a confidence level of α the rejection criteria for null hypothesis (with two-sided alternative $H_1 : \mu_1 \neq \mu_2$) is:

$$t_0 > t_{\alpha/2,\nu} \quad \text{or} \quad t_0 < -t_{\alpha/2,\nu}$$

C.4 Analysis and Conclusions

C.4.1 Mobile Sequence

For UFQ method: $\overline{X}_1 = 11.591 \quad s_1 = 6.734 \quad n_1 = 11$

For PC method: $\overline{X}_2 = 10.167 \quad s_2 = 5.860 \quad n_2 = 15$

Difference between sample means $\overline{X}_1 - \overline{X}_2 = 1.424$

By substituting these values, from equation (C-2) we get, $\nu = 22$ (rounded to the nearest integer)

For $\alpha = 0.10$ the rejection criteria is,

$t_0 > t_{0.05,22}$ or $t_0 < -t_{0.05,22}$ where, $t_{0.05,22} = 1.717$ (from statistical tables for t distribution)

By substituting the values for UFQ and PC methods in to equation (C-1): $t_0^* = 0.562$

Conclusions

Since -1.717 < $t_0^* = 0.562$ < 1.717, at 0.10 level of significance there is not strong evidence to reject our hypothesis. Therefore, the evidence suggests 'not to reject' that the UFQ and PC methods provide comparable results.

The result for the P value is 25% < P < 40% (using statistical tables) which indicates that, given our hypothesis is true, there is a probability of 25%-40% of obtaining the above sample mean difference (1.424 fps) between the two methods.

For a population mean difference of 5 fps, the Type II error probability (β) is estimated to be $0.10 < \beta < 0.20$ suggesting that there is only a 10%-20% chance of failing to reject the null hypothesis when the actual population mean difference is 5fps.

The Power (sensitivity) of the Hypothesis test = 100(1-β) = 80%-90% and hence we can be 80%-90% sure that the difference of results from the two test methods would be within the range of 5fps.

C.4.2 Violin Sequence

For UFQ method: $\overline{X}_1 = 7.168 \quad s_1 = 1.112 \quad n_1 = 11$

For PC method: $\overline{X}_2 = 9.905 \quad s_2 = 6.022 \quad n_2 = 17$

Difference between sample means $\overline{X}_1 - \overline{X}_2 = -2.737$

From equation (C-2) : $\nu = 18$ (rounded to the nearest integer)

For $\alpha = 0.10$ the rejection criteria is,

$t_0 > t_{0.05,18}$ or $t_0 < -t_{0.05,18}$ where, $t_{0.05,18} = 1.734$

By substituting the values for UFQ and PC methods in to equation (C-1): $t_0^* = 0.483$

Conclusions

Since -1.734 < $t_0^* = 0.483$ < 1.734, at 0.10 level of significance there is not strong evidence to reject our hypothesis.

The P value is 25% < P < 40% indicating, given our hypothesis is true, there is a probability of 25%-40% of obtaining the above sample mean difference (2.737 fps) between the two methods.

For a population mean difference of 5fps, β is estimated to be in the range $0.05 < \beta < 0.10$ and the Power (sensitivity) of the Hypothesis test is 90%-95%. Therefore we can be 90%-95% sure that the difference of results from the two test methods is within the range of 5fps.

C.5 Summary

The statistical analysis was carried out using sample data for two test video sequences. Results show that the two subjective testing methods, UFQ and PC, produce comparable performance.