Complexity Management for Video Encoders

Yafan Zhao

This thesis submitted as part of the requirement for the degree of Doctor of Philosophy awarded by the Robert Gordon University

March 2004

To my parents

I

Abstract

Software implementation of block-based video coding standards has been used in a wide range of applications. In many cases, such as real-time multimedia systems or power-constrained systems, the coding performance of software-only video encoders and decoders is limited by computational complexity. This thesis presents research work to develop techniques to manage computational complexity of video encoders. These techniques aim to provide significant complexity saving as well as adaptively controlling the computational complexity.

This thesis first investigates experimentally the most computationally intensive functions in a video encoder. Based on the results of profile tests, several functions are selected as candidates, on which complexity reduction algorithms will be performed. These functions include discrete cosine transform and related functions as well as motion estimation.

Adaptive complexity-reduction algorithms are proposed for computationally expensive functions: discrete cosine transform and motion estimation functions respectively. It is shown that these algorithms can flexibly control the computational complexity of each function with negligible loss of video quality.

The inherent characteristics of coded macroblocks are investigated through experimental tests and they are categorized into "skipped" and "unskipped" macroblocks based on two parameters. An innovative algorithm is developed to reduce the computational complexity by predicting "skipped" macroblock prior to encoding and not carrying out the coding process on these macroblocks.

The approaches described in this thesis can not only achieve adaptive control of the computational complexity of a video encoder, but also can manage the trade-off between complexity and distortion.

These proposed algorithms are evaluated in terms of complexity reduction performance, rate-distortion performance and subjective and objective visual quality by experimental testing. The advantages and disadvantages of each algorithm are discussed.

Acknowledgements

Writing this part of thesis gives me a formal opportunity to thank the people who have supported me and consequently had influence on the accomplishment of this work within three years.

The first acknowledgement goes to Dr Iain Richardson who gave me a chance to do this interesting work. I appreciate very much for his constant guidance, suggestions, help and encouragement during this research work. Special thanks are dedicated for the hundreds of discussions we had on almost every working day in the past three years and for reviewing my thesis twice.

I also want to thank Dr Tony Miller, Dr Lin Shan and Dr Christos Grecos for being the members of the supervision team and the talks I had with them. They provided useful comments and suggestions.

Some people had spent a lot of time on reading my thesis. For this, I would like to thank my colleagues: Laura Muir and Sampath Kannangara.

During my three years study in School of Engineering, I am grateful for the company and happiness from Anita Teo and my colleagues in ICT, who are Wei Wei Li, Ying Zhong, and Jose Solera.

I want to thank the volunteers, some of them I know and others I don't know, for their kindness and help with my subjective testing.

Personally, I owe thanks to Ching Shen Tye, who always sorted out the hardware or software problems I met during my work and writing my thesis. Without this, there is no possibility I can accomplish my work on time.

Finally, I would like to take this unique chance to thank my parents and grandma, who are always in my heart and give me mental support.

Contents

List of Figures

List of Tables

Abbreviations

ADSL	Asymmetric Digital Subscriber Line
ATM	Asynchronous Transfer Mode
AVC	Advanced Video Coding
BMA	Block-Matching Algorithm
CABAC	Context-Adaptive Binary Arithmetic Coding
CAVLC	Context Adaptive Variable Length Coding
CBR	Constant Bit Rate
DCR	Degradation Category Rating method
DCT	Discrete Cosine Transform
DVD	Digital Versatile Disc
EOB	End-of-Blocks
FFT	Fast Fourier Transform
FSA	Full Search Algorithm
GOB	Group of blocks
HVS	Human Visual System
IDCT	Inverse Discrete Cosine Transform
ISDN	Integrated Services Digital Network
ISO/IEC	International Standards Organisation, International Electrotechnical Commission
ITU-T	International Telecommunications Union, Telecommunications Standardization Sector
JVT	Joint Video Team
MAD	Mean Absolute Difference
MB	Macroblock
MPEG	Motion Picture Experts Group
MSAD	Minimum Sum Absolute Difference

MSE	Mean Squared Error
MV	Motion Vector
NAL	Network Abstraction Layer
NNS	Nearest Neighbour Search
PC	Pair Comparison method
PSNR	Peak Signal Noise Ratio
PSTN	Public Switched Telephone Network
RTP	Real-Time Protocol
SAD	Sum of Absolute Difference
SI	Switching Intra
SP	Switching Prediction
SRC	Scalable Rate Control
SSE	Sum of Squared Error
STD	Standard Deviation
TDL	Two Dimensional Logarithmic search
TSS	Three Step Search
VBR	Variable Bit Rate
VCA	Variable Complexity Algorithm
VCEG	Video Coding Experts Group
VCL	Video Coding Layer
VLC	Variable-Length Code
VM	Verification Model
VO	Video Object
VOL	Video Object Layer
VOP	Video Object Plane
VS	Visual Object Sequence

Chapter 1 Introduction

1.1 Statement of the problem

In the past decade, there has been an explosive growth in the field of multimedia communications, where digital video plays an important role. The advance in digitising visual information has led to the rapid development of multimedia technology and applications due to the compatibility of digital video with other types of information such as text or voice. Digital video contains huge amount of data, and despite the increases in processor speed and disc storage capacity, it is still necessary to represent it in a more concise format. The emergence of video compression techniques addresses this problem, making it feasible to store high quality video on a limited-storage disc or transmit it within the limited-bandwidth that today's network can provide. It also drives the development of many widely used applications, such as video conferencing, Digital Versatile Disc (DVD), High Definition Television (HDTV) and so on.

The compression of digital video is accomplished by a video CODEC (Section 2.2), consisting of an encoder that compresses the original video signal and a decoder that reconstructs the original video based on the coded video sequence. Formerly, for the real-time video application, video CODECs were implemented in specially designed hardware because coding and decoding is an extremely computationally complex process (requiring a large amount of calculations), and only dedicated hardware was capable of dealing with it. With the continuous improvement in processor performance, software-only implemented video CODECs running on a general-purpose processors, embedded processors and Digital Signal Processors (DSP) became feasible. It can provide acceptable performance with benefits in terms of cost, flexibility and availability. Consequently, it is now widely used in video applications, such as video conferencing, video telephony, video streaming and mobile video phones.

The performance of a software-only video CODEC may be limited by the amount of processing power available as well as, or rather than, the available transmission bandwidth. This is particularly true in a real-time video communication system and/or power-constrained system. In a desktop video conferencing system, the CODEC runs on a general-purpose PC and has to share processing resources with other applications. In a

mobile video handset, power consumption is closely related to processor utilisation and it may be necessary to restrict computational processing in order to maximise battery life [1]. Current software video applications typically control processor utilisation by dropping frames during encoding, leading to intermittent and "jerky" motion in the decoded video sequence. Computational complexity of video CODECs becomes a major constraint on the coding performance and it is therefore important to develop methods of managing the computational complexity of video CODECs.

In the literature, the existing contributions of reducing the computational complexity of software video CODEC focus on speeding up specified operations during encoding or decoding, such as fast algorithms for computationally expensive functions (especially motion estimation, Discrete Cosine Transform (DCT) or Inverse Discrete Cosine Transform (IDCT)). The computation reduction by this type of algorithm can vary significantly depending on the scene content of the video sequence. The amount of complexity reduction and its influence on video quality and bitrate is difficult to predict.

The aim of this work is to flexibly manage the computational complexity of a video encoder and control the tradeoffs between the complexity and rate-distortion performance. In contrast to conventional solutions, the approaches proposed in this thesis are designed to provide comprehensive and predictable control of the computational complexity of video encoder rather than simply decreasing it. Given a target complexity, the video quality and rate distortion performance will be optimised within this computation limit. These approaches aim to provide nearly the same subjective video quality as the encoders without complexity reduction. This is based on a fact that a small loss in video quality caused by reduction of computational complexity may not be perceived by audiences due to the idiosyncrasy of the human visual system.

By developing complexity management techniques to decrease and control the computational complexity, the coding performance of software-only video encoders will be improved, resulting in good perceived video quality in real-time multimedia systems and prolonging the battery life in mobile video devices while maintaining the same video quality.

1.2 Objectives

In order to fulfil the aim of this work, the first step that needs to be performed is examining and analysing the computational cost of each operation involved in the encoding process. Based on these investigations, techniques are proposed to reduce and manage the most computationally complex functions, including DCT and related functions as well as the motion estimation function.

The first innovatory approach to the complexity problem is developed for the DCT function. The objective is to achieve near-constant complexity reduction of the DCT function throughout the entire video sequence at the expense of a controlled loss in video quality. This approach should also be applied to other related functions of DCT, including IDCT, quantisation and inverse quantisation.

The method for controlling motion estimation complexity has a similar objective to that for the DCT function: reducing computational complexity to a target and maintaining it at this level with negligible loss of video quality. The origin of this method starts from the investigation of a fast motion estimation algorithm.

Under the motivation of providing a comprehensive and more significant computation reduction in the encoder, the objective is to investigate approaches that can decrease and control the complexity of the entire coding process. This method should be able to achieve the best rate-distortion performance within the complexity limit.

The objectives of this work are fulfilled by developing novel approaches, which are implemented and evaluated through simulation. The strengths and weaknesses of them are also discussed based on the experimental results.

1.3 Organization of the thesis

The organization of this thesis is as follows:

Chapter 2 presents a comprehensive description of the block-based video coding technologies and most popular video coding standards. It starts with a brief introduction to the fundamental terms used in block-based video coding, then it concentrates on the main operations and techniques employed during encoding. The video coding standards

(H263 and H.263+), on which the video CODECs used in the experimental work in this thesis are based, are introduced. The concepts of complexity, bitrate and distortion are introduced and the complexity problem is discussed.

Chapter 3 describes the test environment for the simulation, including equipment, CODECs and testing sequences. The video quality assessment approaches to evaluate the quality of decoded video sequences and the methods used to measure the computational complexity are also introduced.

The development of innovative approaches and algorithms to solve the computational complexity problems are presented in Chapter 4 - Chapter 7.

Chapter 4 analyses the computational complexity of each function in the video encoder and finds out the most complex functions, which will be investigated in subsequent few chapters in this thesis. It also illustrates the rate-distortion performance and complexity-distortion performance of example CODECs.

In Chapter 5, the development of an approach used to manage the computational complexity of DCT and related functions are described. An adaptive algorithm that consists of a basic method for reducing computations and a feed-back mechanism is proposed to achieve the objective of DCT complexity control that is described in Section 1.2.

Chapter 6 investigates a new method to control the computation of motion estimation in a video encoder. This method is based on a fast motion search algorithm and it can maintain the computational complexity at scalable target levels by controlling the motion estimation search pattern.

Typically, many macroblocks are skipped during encoding of block-based video CODEC, particularly at low bitrates. Chapter 7 describes an algorithm that predicts the occurrence of skipped macroblocks prior to encoding, making it possible to save significant computational effort by not coding these macroblocks.

A summary of experimental results of the proposed approaches from Chapter 4 - Chapter 7 and the main contributions of this work are discussed and evaluated in Chapter 8.

Possible extensions and new directions of this research work are also indicated. Finally, Chapter 9 concludes this thesis.

1.4 Original contributions

This work addresses the computational complexity problem of software-only video encoders. Several approaches are developed for different computationally expensive functions in a video encoder, resulting in significant reduction and flexible control of computation costs. The original contributions to the body of knowledge in video coding are listed below.

- Analysing computational complexity of the entire coding process of a video encoder.
- Proposing the concept of controlling the computational complexity, i.e. keeping constant and stable complexity in an encoder.
- Investigation of methods to predict end of block by modelling DCT coefficients.
- Development of an adaptive algorithm for controlling DCT complexity, achieving constant complexity target.
- Investigation of a complexity-reduction approach for motion estimation, maintaining complexity at the pre-determined target.
- Examination of characteristics of macroblocks and development of a method to classify macroblocks into various types.
- Invention of a macroblock classification approach to save significant computational cost with improved accuracy.

Chapter 2 Block-based video coding fundamentals

2.1 Background

Digital video has taken the place of traditional analogue video in a wide range of applications due to its compatibility with other types of data (such as voice and text). However, at the same time, the high bitrate required to represent digital video dramatically increases the burden on storage space, processing ability and transmission bandwidth. For example, using a video format of 352×240 pixels with 3 bytes of colour data per pixel, playing at 30 frames per second, 7.6 Megabytes of disc space is needed for one second of video and it is only feasible to store around 10 minutes of video in a 4.6 Gigabytes DVD. When it is transmitted in real time through the internet, it requires a channel with 60.8 mbps, which is 118 times of the bandwidth (500 kbps) of Asymmetric Digital Subscriber Line (ADSL) for current broadband internet service. Even if high bandwidth technology is able to provide sufficient transmission speed and the storage problems of digital video are overcome, the processing power needed to handle such massive amounts of data would make video processing hardware very expensive. Although significant progress in storage, transmission and processing technology is being made, it is primarily compression technology that has made the widespread use of digital video possible.

Generally speaking, there is a large amount of statistical and subjective redundancy in digital video sequences. Video compression techniques are designed to reduce the size of information for storage and transmission. Through exploiting both statistical and subjective redundancy, a compact representation of video data is achieved and important information is kept. The performance of compression depends not only on the amount of redundancy in the video sequence but also on compression techniques used for coding. There are two classes of techniques for image and video compression: lossless coding and lossy coding. Lossless coding techniques compress the image and video data without any loss of information and the compressed data can be decoded exactly the same as original data; however, these techniques obtain a very low compression ratio and result in large files. Consequently, they are appropriate for applications requiring no loss introduced by compression, for example, medical image storage. On the other hand, lossy coding

methods sacrifice some image and video quality to achieve a significant decrease in file size and a high compression ratio. Lossy coding techniques are widely used in digital image and video applications due to the high compression ratio provided.

The growing interest in digital image and video applications has led academics and industry to work together to standardize compression techniques in order to meet the requirement of various applications. Several series of standards have been successfully developed by two organizations: International Organisation of Standardization, International Electrotechnical Commission (ISO/IEC) and the International Telecommunications Union, Telecommunications Standardization Sector (ITU-T). These standards address a wide range of video applications in terms of bitrate, image quality, complexity and so on. In the following section, the fundamental video coding techniques and popularly used compression standards are introduced.

2.2 Block-based video coding

Most of the popular video coding standards utilize block-based video coding techniques, the basic terminology and abbreviations of which are listed below:

- **Luma:** a sample or array representing a video luminance signal, often symbolized as Y.
- **Chroma:** a sample or array representing a blue or red video colour difference signal, often symbolized as C_b and C_r.
- **Pixel:** A colour element at one position in a displayed image.
- **Sample:** A luma or chroma component at one position in a video frame.
- **Block:** An M×N matrix of samples or transform coefficients. M is the number of columns and N is the number of rows. In this thesis, a block is defined as 8×8 samples or transforms coefficients (unless otherwise stated).
- **Macroblock (MB)**: 16×16 luma samples (Y) and associated chroma samples (C_b and C_r). In this thesis, the chroma components of a macroblock are assumed to each consist of 8x8 samples (unless otherwise stated).

- **Frame:** A set of samples representing a single time instant of a progressive video signal. A video frame consists of one array of luma samples and two arrays of chroma samples.

- **Picture**: In this thesis, a picture is defined as a coded video frame.

- **Discrete Cosine Transform (DCT):** A transform converting a set of samples from spatial domain to frequency domain.

- **Motion estimation:** The process of finding a match in one or more reference frame(s) for a macroblock in a video frame.

- **Motion compensation:** Computing the difference between a macroblock and a matching area in one or more reference frame(s).

- **Motion vector (MV):** The offset between a macroblock and a matching area in a reference frame.

- **Quantisation:** Mapping a signal with N possible values to a signal with M possible values, where $M \leq N$.

- **Entropy coding:** The process of representing data symbols (e.g. transform coefficients, motion vectors) in a compressed form.

- **Encoder:** Compresses a series of video frames into a concise representation ("coded video") before transmission and/or storage.

- **Decoder:** Decompresses coded video prior to display and/or storage.

- **CODEC:** An abbreviation of video encoder and decoder.

Within an image or a single frame of a video, there are usually similarities between neighbouring pixels, referred to as spatial redundancy. The compression of a single frame is accomplished by replacing the same image information with a smaller-size representation. Many techniques have been developed to allow the replacement of whole segments of an image with a set of data representing the image in a transformed state, among which, the Discrete Cosine Transform (DCT) is widely used in block-based coding. The DCT transform is commonly applied to an 8×8 block.

The movement and detail in a video scene tends to vary gradually, and thus adjacent video frames are often similar. This is temporal redundancy. If an area of a frame is compared with a previous frame, it is possible that an identical or similar area can be found. Temporal redundancy is reduced by replacing the area with a corresponding area derived from one or more reference frames. The basic area for comparison and replacement is typically a macroblock in block-based coding. If the replacement is not exact, DCT-based coding can be performed to further reduce the spatial redundancy.

The block-based video coding technique can achieve a good compression ratio while it is also computationally efficient, which has led to its wide-spread use in many coding standards, such as H.263 [2], MPEG-2 [3] and MPEG-4 [4].

2.3 Video coding techniques

2.3.1 Motion estimation and compensation

A video sequence typically contains temporal redundancy: that is two successive pictures are often very similar except for changes induced by object movement, illumination, camera movement and so on. Motion estimation and compensation is used to reduce this type of redundancy in moving pictures. The Block-Matching Algorithm (BMA) for motion estimation has been proved to be very efficient in terms of quality and bit rate; therefore it has been adopted by many standards based CODECs. In this section, the basic principle of block matching motion estimation and compensation is introduced and fast motion search algorithms are addressed.

Principle of block matching motion estimation and compensation

In block matching motion estimation, a single image is subdivided into non-overlapping N×N blocks, where N is usually 16 or 8. Each block in the current frame is compared with blocks of the same size in reference pictures in order to find out the best match, which meets an error criterion based on the measurement. The location of the block is defined by co-ordinates (x,y) of top-left corner of the block. The vector pointing from the current block to the best match block is chosen as Motion Vector (MV). The residual "difference" between current and reference frames is computed by the process of motion compensation, and then coded and transmitted with motion vectors.

Two error measurements are commonly used for block matching criteria: Sum of Squared Error (SSE) [5] and Sum of Absolute Difference (SAD)[6], which are described in Equation 2-1 and Equation 2-2 respectively.

$$SSE(d_x,d_y)=\sum_{x,y=0}^{N-1}(f_c(x,y)-f_r(x+d_x,y+d_y))^2$$

Equation 2-1

$$SAD(d_x,d_y)=\sum_{x,y=0}^{N-1}\left|f_c(x,y)-f_r(x+d_x,y+d_y)\right|$$

Equation 2-2

$f_c(x,y)$ is luminance pixel value of N×N block in current frame and $f_r(x+d_x,y+d_y)$ is the block in the position of (d_x,d_y) in reference frame. For a block size of 16×16, SSE requires 16×16=256 multiplications and 2×16×16=512 additions, whereas, SAD only need 2×16×16=512 additions. Compared with SSE, SAD is much less computationally demanding and is consequently more widely utilized.

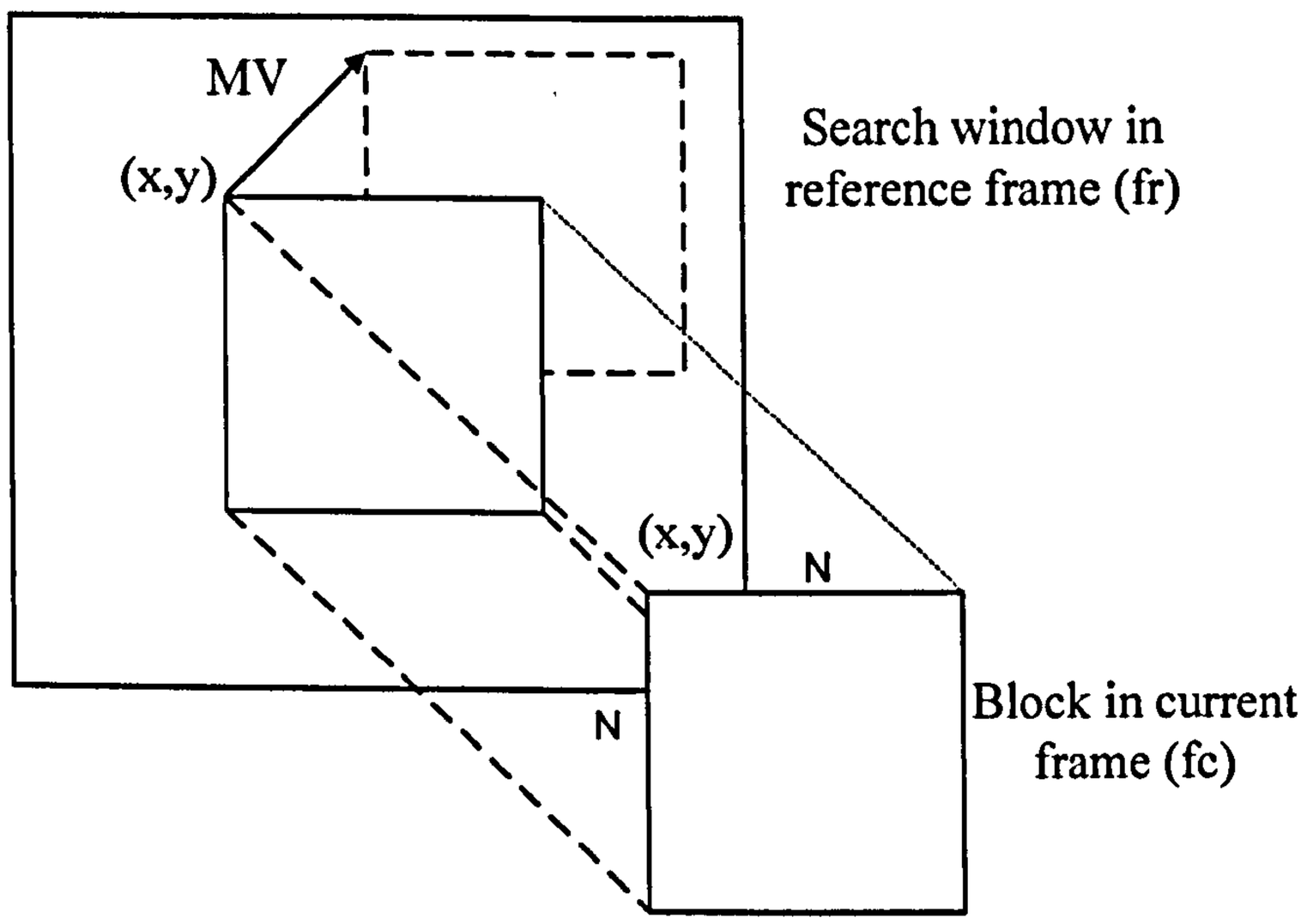


Figure 2-1 Example of block matching motion algorithm

Figure 2-1 illustrates the process of block-based motion search. An N×N block in the current frame (f_c) located by the co-ordinates (x, y) is compared with a same-size block in the search range of reference frame (f_r). SAD is computed for each search and the best motion vector is the one with the minimum SAD.

Searching algorithms for block matching

Full search algorithm----The most straightforward search method for block matching is Full Search Algorithm (FSA), which examines exhaustively all positions in the search area. The motion vector which predicts the best matching block can be definitely found, so FSA can be considered as an optimal solution to motion estimation. However, if the search range in either direction is w with step size of 1 pixel and assume the search range is square, there are in total $(2w+1)^2$ times of displacement in order to find a motion vector for each block, requiring a large amount of computations, especially for a large search window. The high computational requirements of FSA make it unacceptable for real-time software-implemented video applications.

In order to solve the computation problem of FSA, many fast algorithms have been proposed to decrease the computational complexity of motion estimation by reducing the number of candidates for motion searching, such as three-step search [6], two-dimensional logarithmic search [7], cross-search [8], diamond search [9] and so on. The difference between these fast algorithms is the way they skip those candidates that are unlikely to have minimum error. Moreover, there are some proposed algorithms which vary the number of pixels calculated for error measure, known as sub-sampling [10] and normalized partial distortion search algorithm [11].

Two-dimension logarithmic search (TDL) ----The TDL search is the first fast block match algorithm [7], illustrated in Figure 2-2. The search typically starts from the location corresponding to zero displacement, which has the same co-ordinates as current block (x, y). At each step, four locations in a "+" shape are examined as well as the central point and the location with smallest SAD is chosen as the centre of the next search. The radius of "+", called search step size (S), is reduced by this equation (S=S/2) for the next step if the best match is the central point. Otherwise, the step size remains the same. When S is reduced to 1 pixel, it searches all the locations surrounding the last best match, and then the best match is the motion vector. In the example shown in Figure 2-2, the

search window is 7 pixels and the initial step size is 2. The motion vector (2, 3) is found through best matching of first three steps: (0, 0), (2, 0) and (2, 2).

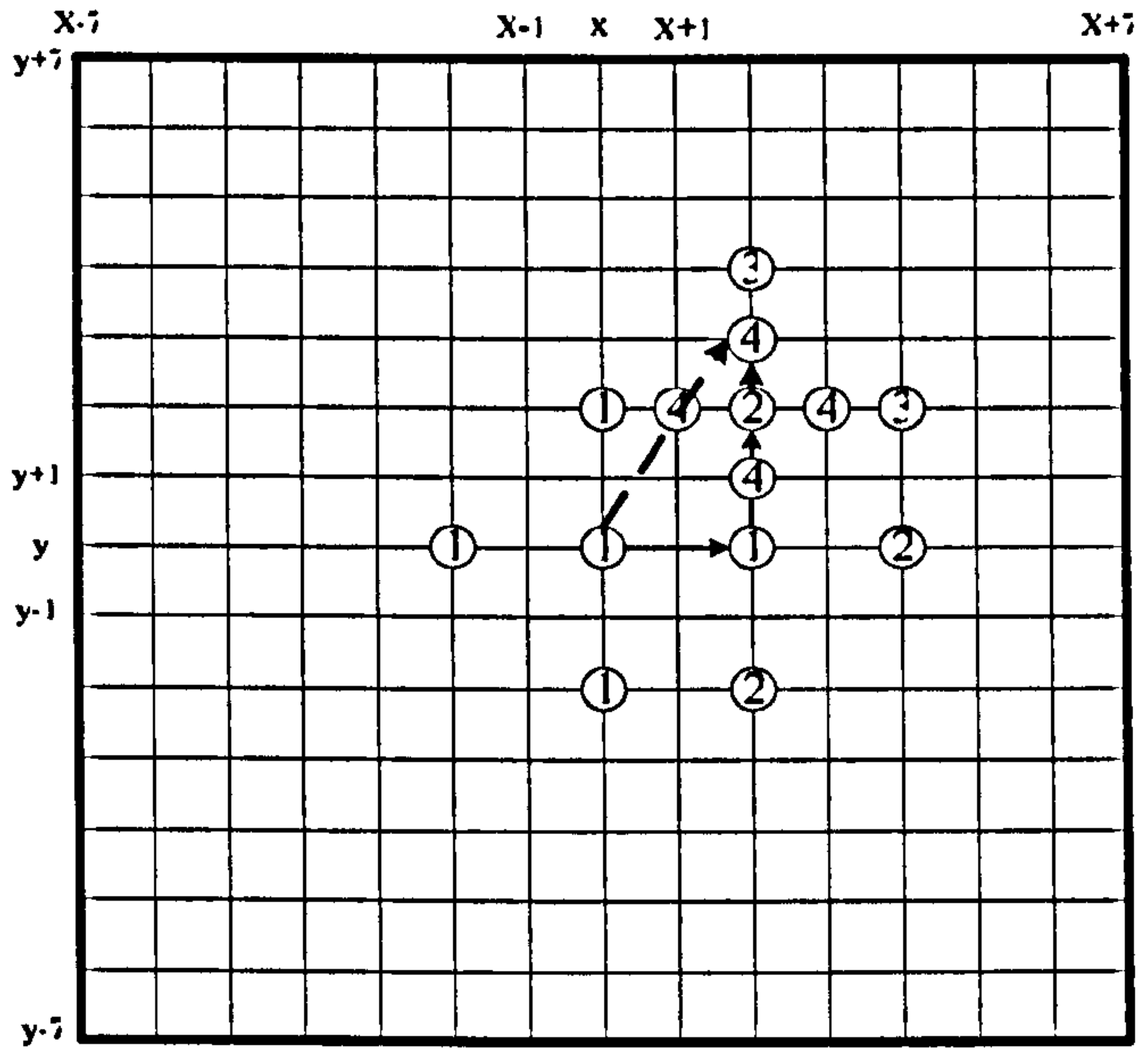


Figure 2-2 Example of 2-D logarithmic search with search step size =2

Three step search----Another popular fast search algorithm is Three Step Search (TSS) [12], which is similar to TDL. It also starts from searching the zero displacement location, but it tests 8 surrounding points in a square shape instead of four points. The initial search step size depends on the search window size. If the search window size is +/- $(2^N -1)$, search step(S) is set to $2^{(N-1)}$ at the first stage. Like TDL, the best match of this step becomes the centre point of the next step. After each stage, the search step size is reduced by the equation (S=S/2) until S is equal to 1. Figure 2-3 illustrates the search procedure with search window size +/-7 = +/- $(2^3 -1)$. The search starts with first step size equal to (2^{3-1}) =4; the best matching locations found for first two steps are (0, 4), (2, 4) accordingly and the final MV is (3, 3).

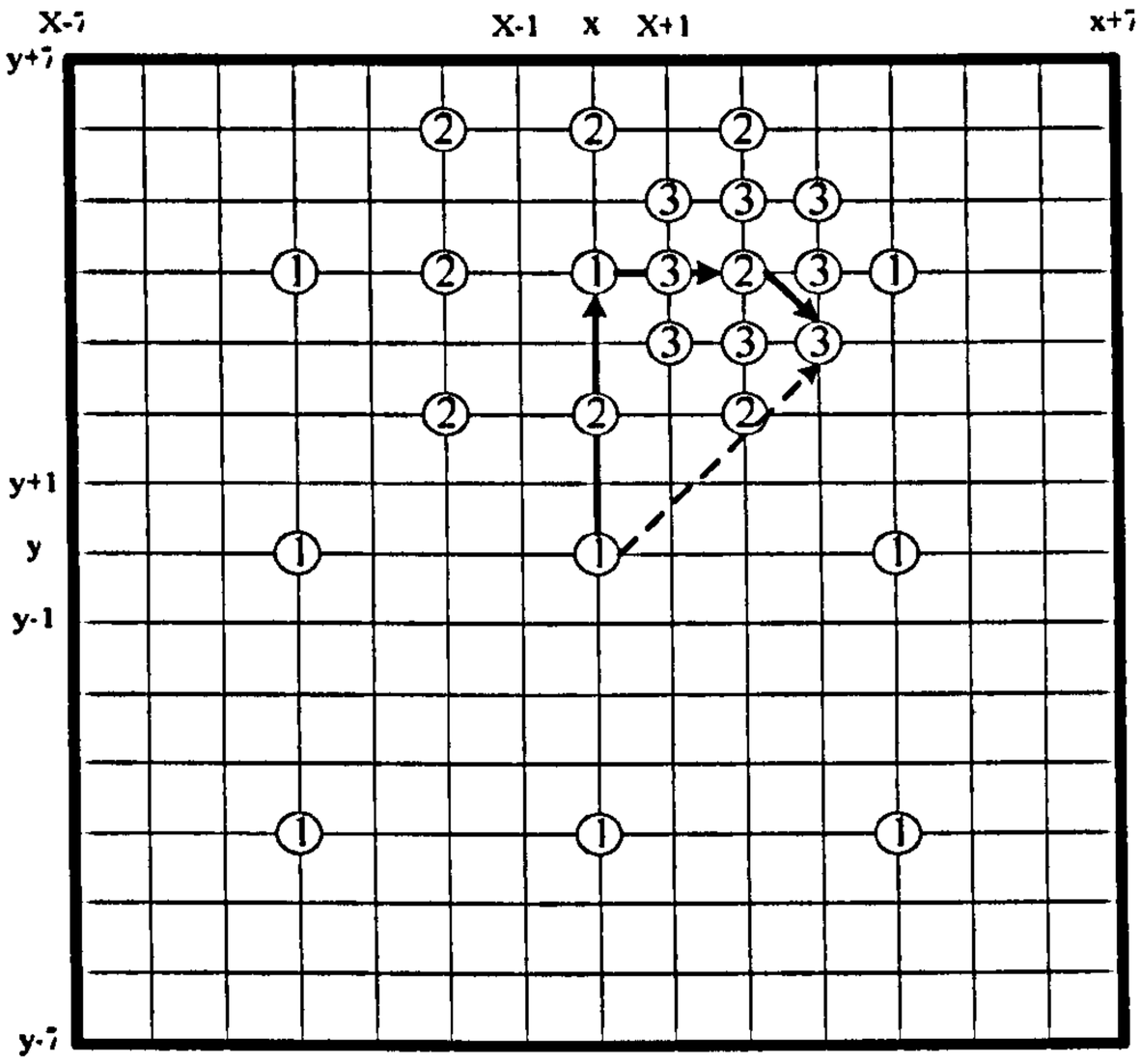


Figure 2-3 Example of three step search

Comparison of block matching algorithms----TDL and TSS fast search algorithms reduce the computations by reducing the number of candidates in search range. Table 2-1 compares the maximum and minimum number of search points required by full search algorithm, TDL (search step size =2) and TSS with search window = +/-7. It is clear in the specified search window that TSS has a fixed number of candidates, but the number of points varies case by case with TDL. Both of the two fast algorithms dramatically reduce the computations when compared with full search algorithm.

Table 2-1 Comparison of block matching algorithm with search window = +/-7

Block matching algorithm	Number of search points	
	Minimum	Maximum
Full Search	255	255
2-D logarithmic	13	26
Three-step search	25	25

Sub-pixel motion estimation

The fast block-based motion estimation algorithms detailed above assume that the best match is located exactly in a position with integer pixel offset in the search area. However, in many cases, the best match may be between pixel positions, so there is a need for motion search at sub-pixel level. The values in the sub-pixel position are created by interpolating the surrounding integer pixels; then the motion search is carried out on both integer-pixel and sub-pixel locations to find the best match. Obviously, sub-pixel motion estimation is much more computationally complex than integer-pixel motion search as extra interpolation points have to be computed, but it can significantly increase the accuracy of motion compensation. In order to reduce the computation, sub-pixel search is commonly executed only around the best integer-pixel match.

2.3.2 DCT-based transform

Through motion estimation and compensation, the temporal redundancy in the current frame is reduced and a residual frame is generated by subtracting blocks in the current frame from corresponding best match in reference blocks. There is still a certain amount of spatial redundancy in the residual frame and Discrete Cosine Transform (DCT) is applied, followed by quantisation to further reduce the redundancy. DCT is a mathematical method transforming image data from the spatial domain to the frequency domain. An N×N block of samples is converted into transform coefficients. The block size for DCT transform in image and video coding is usually chosen to be 8 because an 8×8 block provides good spatial correlations between pixels and it does not put tremendous burden on processing ability and memory storage [13]. Figure 2-4 illustrates the basis functions of an 8×8 block. The top-left one is the "DC" basis function, which represents zero spatial frequency. The spatial frequency of other "AC" basis functions increases horizontally along the top row and vertically down the left column. The weighted value of basis functions is added or subtracted to each other to generate the 8×8 transformed coefficients.

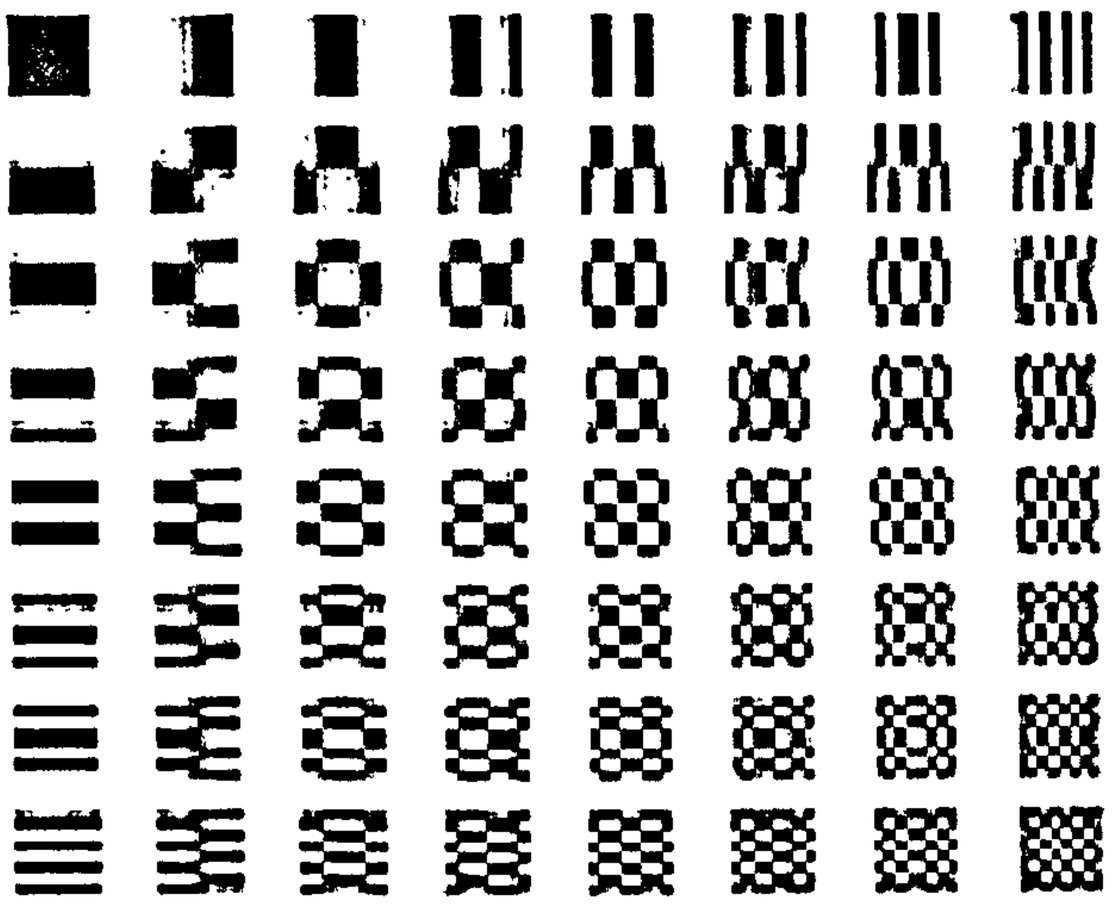


Figure 2-4 Basis functions of DCT transform

The general equation of N×N two-dimensional DCT can be defined by the following equation, where f is N×N block of original pixels and F is the matrix of transformed coefficients.

$$F(u,v) = \frac{2}{N} A(u)A(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\cos(\frac{(2i+1)u\pi}{2N})\cos(\frac{(2j+1)v\pi}{2N})f(i,j)$$

Equation 2-3

where $A(x) = \frac{1}{\sqrt{2}}$ for x = 0, 1 otherwise.

The DCT transform involves costly matrix multiplication. In practice, fast DCT (FDCT) algorithms are proposed to replace DCT operation to reduce computations. Mathematically, the DCT is perfectly reversible without any loss of information. If an inverse DCT is applied to the transform coefficients, the reconstructed image data is exactly as same as the original.

The transformed coefficients represent how much of each basis function is present in the original block. A large value of a coefficient means that the original image data vary corresponding to a specific frequency. In a typical scenario, an N×N block contains spatially related samples and the values of these samples do not vary dramatically, which

can be represented by low frequency basis coefficients. Consequently, after DCT transform, most coefficients are small values. The high valued ones are often those that represent the lower frequency functions and cluster around the DC coefficient position. Although compression has not been achieved at this stage, DCT transform makes a good preparation for next stage of coding with many low value coefficients.

2.3.3 Quantisation

There are typically a lot of near zero coefficients in the transformed block. Quantisation is used to discard these less important coefficients by dividing each coefficient by an integer. Only significant DCT coefficients are left after quantisation so that compression of the image or residual is obtained. Quantisation is an irreversible process, the data loss caused by which can not be recovered.

The amount of discarded coefficients may be varied by using a quantisation step size (Q). Large Q tends to throw away most of the coefficients, retaining only the most important large-value coefficients. On the contrary, more coefficients are kept in the quantised block by using small Q. The level of Q decides the number of zero-coefficient generated by quantisation and affects video quality and final compression rate.

2.3.4 Entropy coding

There are only a few non-zero coefficients left, typically low frequencies around the "DC" one, after quantisation. The quantised DCT coefficients are further coded by three steps: reordering, run-level coding and entropy coding. The quantised coefficients are reordered into a one-dimensional array by scanning them in zigzag order. DC coefficient is at the first position of the array, followed by the remaining AC coefficients from low frequency to high frequency. Since most of the high frequency coefficients tend to be zero, this arrangement separates the non-zero and zero coefficients. The rearranged coefficients array is coded as a series of run-level pairs: "run" is the number of consecutive zero before the next non-zero coefficient represented by "level". The run-level pairs and other coding information (such as motion vector and prediction types) are further compressed by entropy coding such as Variable Length Codes (VLC). The more frequently occurring pairs are represented by shorter codes whereas the infrequently

occurring pairs are represented by longer codes. The most popular statistical algorithms used in encoding are Huffman or modified Huffman coding and arithmetic coding.

2.3.5 Generic video CODEC

The architecture of a generic block-based DCT CODEC is depicted in Figure 2-5. At the encoder, motion estimation and compensation are performed on every block or MB to find the best motion vector and to produce the minimum residual frame of current frame n. Each block of the residual frame is DCT transformed, quantised and then entropy coded together with motion vectors from motion prediction, and finally sent to an output buffer for transmission. Meanwhile, the encoder carries out inverse Quantisation and inverse DCT to regenerate the residual frame, which is combined with reference frame to reconstruct frame n. The reconstructed frame n is not exactly the same as initial input frame due to the data loss during quantisation; however, it is a replica of the decoded frame at the decoder and is saved in the frame store for motion estimation for the next frame. Using reconstructed frames in motion estimation and compensation for the next frame assures that decoder performs motion compensation based on the same reference frame as encoder. Thus it avoids the accumulated error caused by different reference frames at both encoder and decoder.

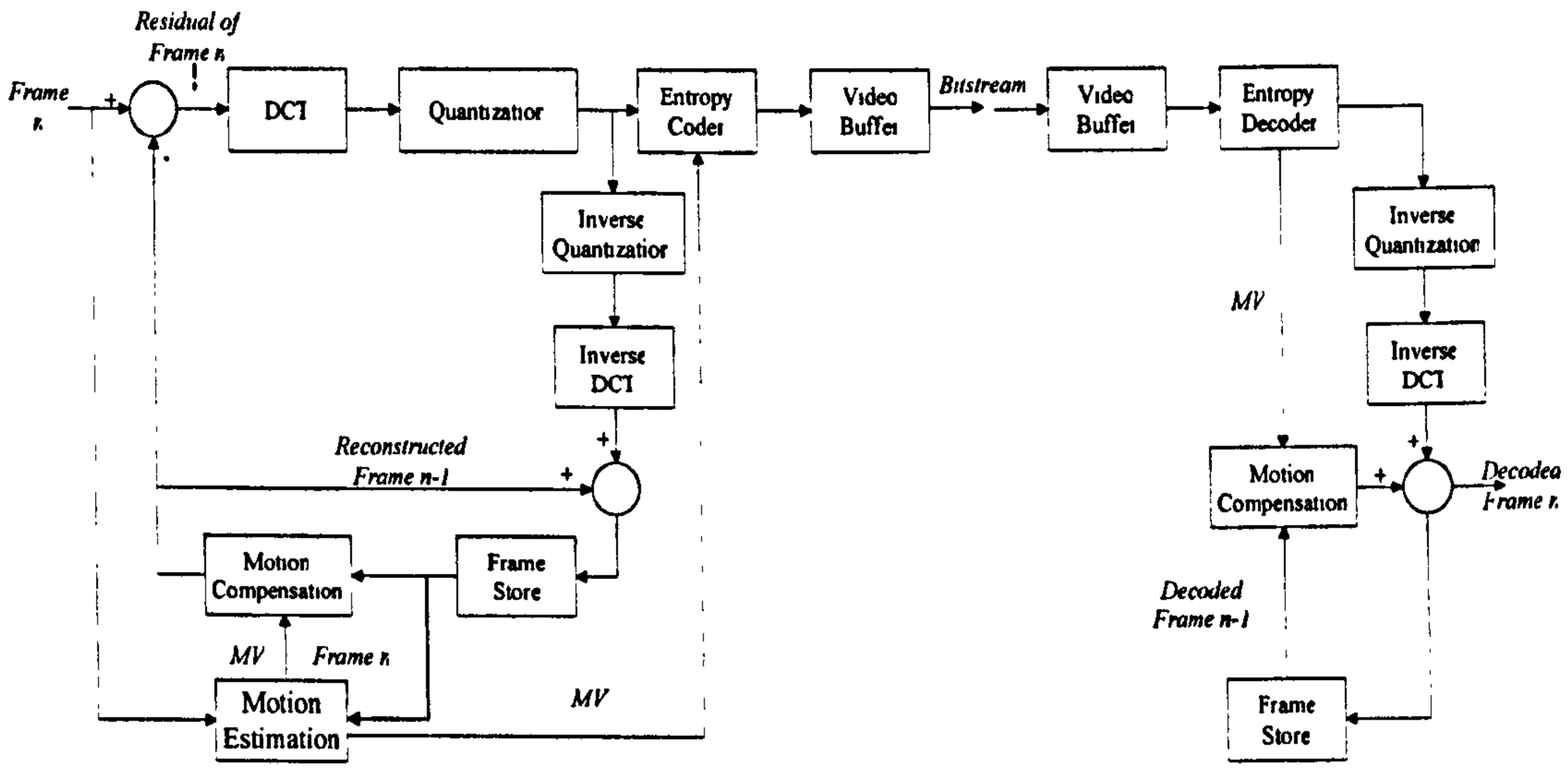


Figure 2-5 Generic DCT coder and decoder

The decoder performs inverse operations, receiving coded video and decoding it to obtain quantised coefficients. With inverse quantisation and subsequent inverse DCT, the residual pixel values of a frame are produced. These pixel values are added to the reconstructed previous frame created using motion compensation based on the received motion vector to reproduce the entire decoded frame. The decoded frame is kept in a frame store for use in motion compensation of the next frame.

2.4 Video coding standards

ITU-T Video Coding Experts Group (VCEG) and ISO Motion Picture Experts Group (MPEG) are two formal organizations that develop video coding standards. These standards are designed for a variety of video applications ITU-T standards are called Recommendations and H.26x series (H.261 [14], H.262 [3], H.263 [2], and H.264 [15]) , are designed for applications, such as video conferencing and video telephony. Meantime, ISO/IEC MPEG is responsible for the MPEG-x series: MPEG-1 [16], MPEG-2 [3], MPEG-4 [17], MPEG-7 [18] and MPEG-21 [19]. They address the problem of video storage, broadcasting video and video streaming through internet and mobile networks.

2.4.1 H.263

H.263 [20], originally standardised by ITU-T in 1993, is a video coding standard for low bit rate video communication over Public Switched Telephone Network (PSTN) and mobile networks with transmission bitrates of around 10-24kbps or above. It follows a preceding standard H.261, which is dedicated for video conferencing over Integrated Services Digital Network (ISDN) operated at bit rates of k×64kbps, where k is an integer from 1 to 30. The limitation on bit rate has been removed and H.263 can be applied for a wide range of bitrates (up to 20Mbps). H.263 employs a similar coding framework to H.261 (block-based motion estimation and DCT transform coding), but it significantly improves coding performance and error recovery with the help of new techniques and optional coding modes.

Picture format

In H.263, each picture is represented by luminance component (Y) and chrominance components (C_r and C_b). Luminance and chrominance components are sampled with the

4:2:0 sampling pattern, according to CCIR Recommendation 601 [21]. C_r and C_b have half the resolution of the Y component in both horizontal and vertical directions and are positioned in the centre of the four Y samples. As shown in Figure 2-6, for every four pixels, there is only one C_r and C_b sample , but four Y samples.

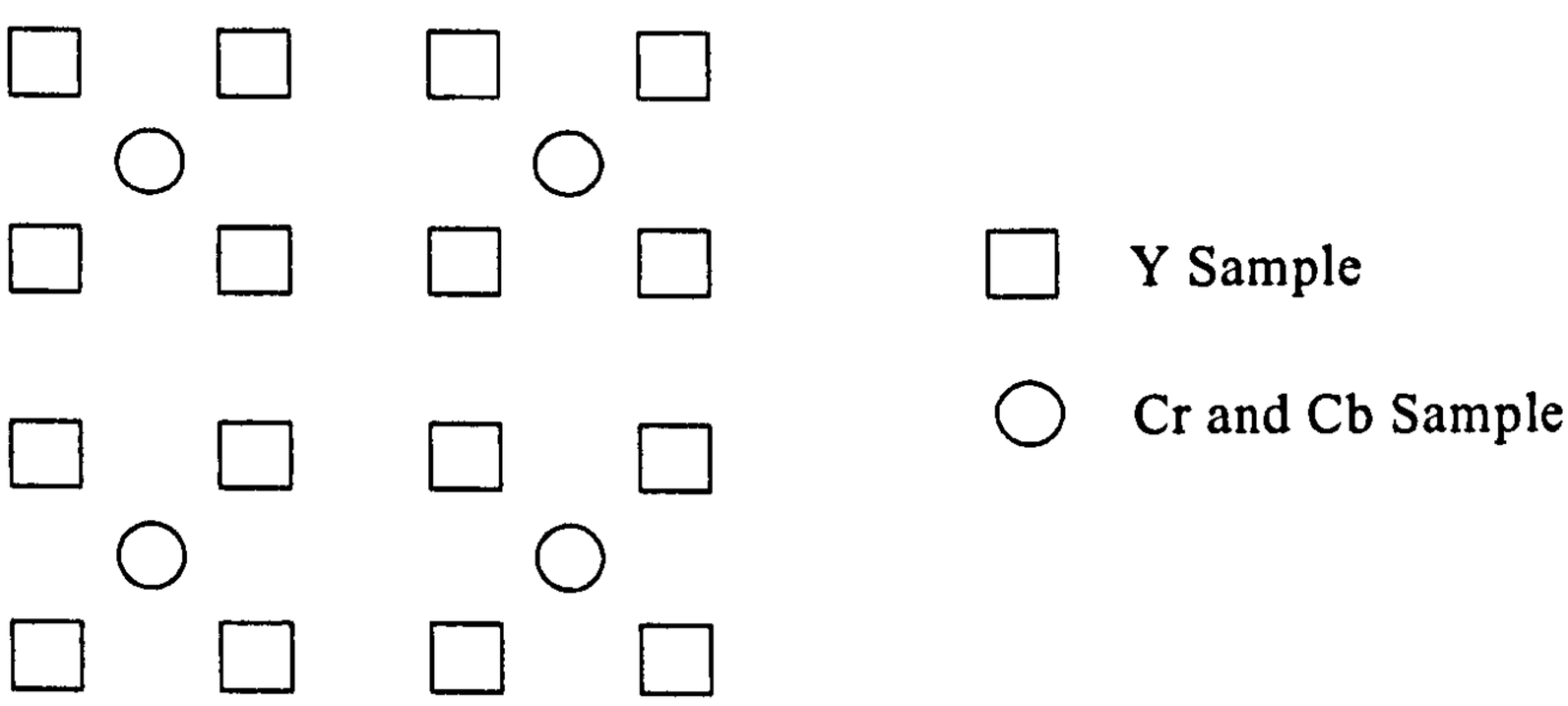


Figure 2-6 4:2:0 sample pattern for H.263

H.263 supports 5 standardized picture formats, sub-QCIF, QCIF, CIF, 4CIF and16CIF, the picture sizes of which are shown in Table 2-2. For instance, in QCIF, 176 luminance components are sampled at every line and 144 lines are scanned in each frame. Chrominance values are sampled with half the resolution of luminance, so its chroma resolution is 88×72 samples.

Table 2-2 H.263 picture format

Picture Format	Sub-QCIF	QCIF	CIF	4CIF	16CIF
Number of samples per line for luminance	128	176	352	704	1408
Number of lines for luminance	96	144	288	576	1152
Number of samples per line for chrominance	64	88	176	352	704
Number of lines for chrominance	48	72	144	288	576

Picture structure

H.263 uses a block-based video coding technique, where a single picture is divided into number of 16×16 macroblocks. Each macroblock contains four 8×8 Y blocks and one corresponding 8×8 C_r and C_b block, as shown in Figure 2-7.

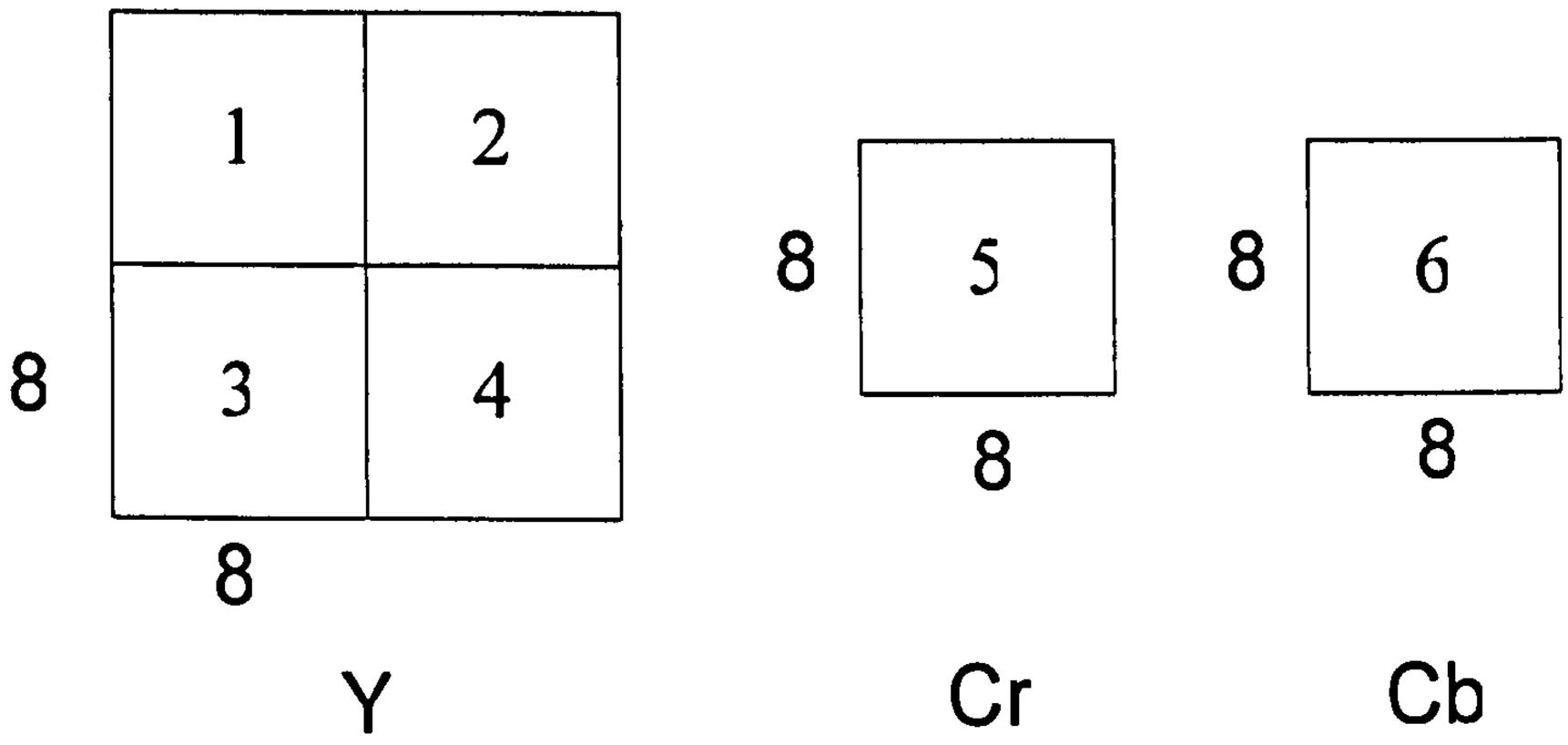


Figure 2-7 Macroblock

The MB is the basic unit in the coding process. A group of blocks (a GOB) contains k rows of MB where k = 1 for sub-QCIF, QCIF, CIF, two for 4CIF and k = 4 for 16CIF. The number of GOBs for various picture formats can be computed by dividing the number of luminance lines by 16K. Figure 2-8 shows the GOB structure of a QCIF picture.

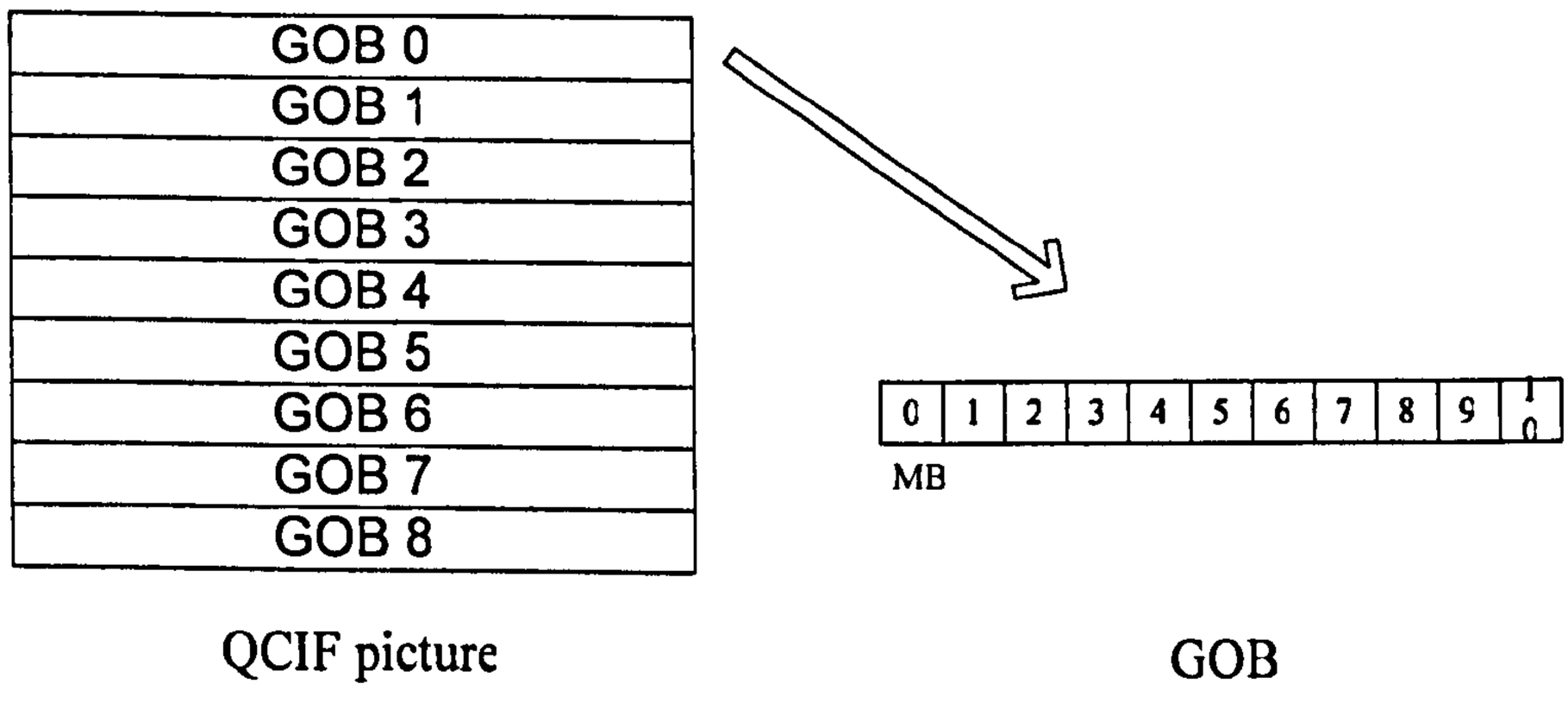


Figure 2-8 GOB structure of QCIF

There are two basic coded picture types in H.263: intra and inter coded pictures. For an intra-picture, no prediction is applied and the original sample values are coded with DCT transform, quantisation and entropy coding. An inter-picture is predicted from a previous intra or inter picture, then the "residual" difference is coded together intra-picture with motion vector.

Half-pixel motion compensation

H.263 supports motion compensation with half-pixel precision as well as integer–pixel precision. As shown in Figure 2-9, half pixel values are created by interpolating neighbouring integer-pixel values. Point 'c' is computed from integer-pixel point 'A', 'B', 'C', 'D'. Point 'a' is calculated from 'A' and 'B'. H.263 can transmit motion vectors that fall into not only integer-pixel but also half-pixel positions. This dramatically improves the coding performance when compared with H.261, where only integer motion compensation and a loop filter are utilized.

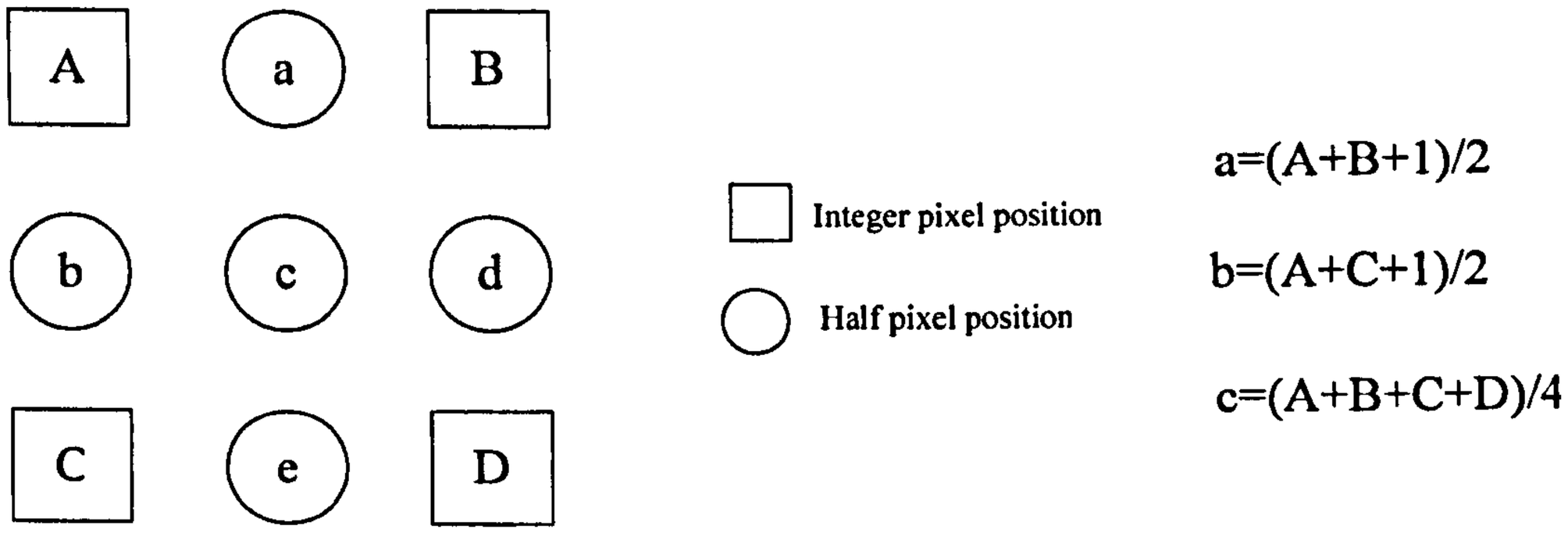


Figure 2-9 Interpolation of Half-pixel value

Predictive coding of Motion vector

The motion vectors in H.263 are encoded by means of predictive coding. As shown in Figure 2-10, motion vectors from previous coded MBs, which are to the left (MV_1), top (MV_2) and top right (MV_3) of the current MB, are used for prediction. The median of MV_1, MV_2 and MV_3 is the predictor (MV_p) for current motion vector (MV_c) and the difference between MV and predictor is calculated according the following equation.

$$MV_d = MV_c - MV_p$$

where $MV_p = Median(MV_1, MV_2, MV_3)$

Equation 2-4

If the current MB is located along the boundary of the picture or GOB, one or two previous MVs may not exist. As illustrated in Figure 2-10, if only one previous MB is outside the boundary, a zero motion vector takes its place. When two previous MBs are missing, the remaining MV is used as predictor.

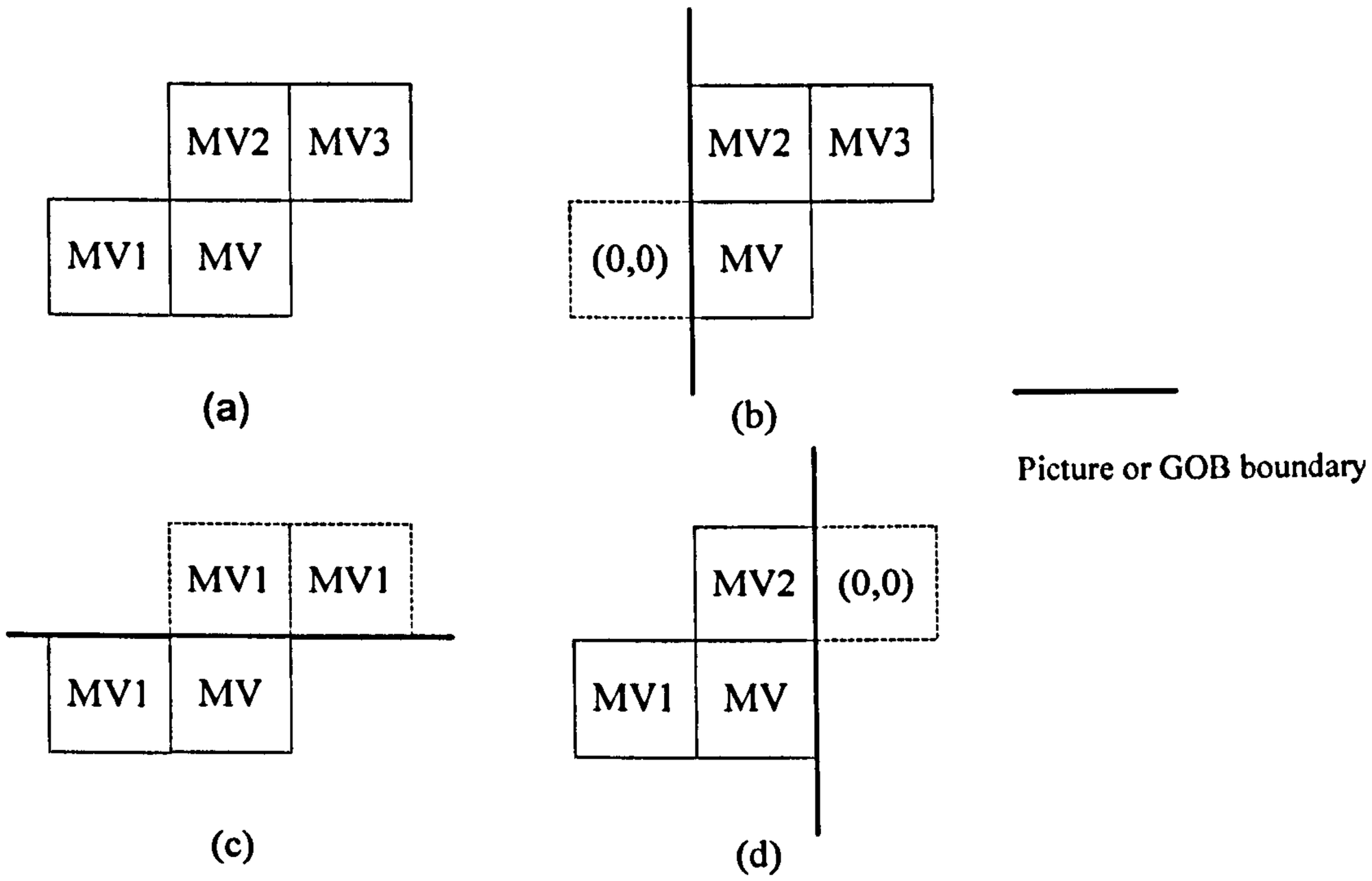


Figure 2-10 Prediction of motion vector

Advanced optional modes

H.263 has four advanced options designed to improve compression performance. The encoder and decoder must negotiate the use of optional modes. An encoder may utilize all or some of the options that the decoder supports. The four optional modes are as follows:

Unrestricted motion vector Mode (Annex D) ----In this mode, motion vectors are allowed to point outside the boundaries of the picture. A pixel position outside the picture takes the values of the nearest boundary pixel. This mode extends the motion search range from default [-16, 15.5] to [-31.5, 31.5]. A significant gain can be obtained for videos where an object moves into or out of the picture or around the border.

Advanced prediction mode (Annex F) ----Using this mode, motion compensated prediction can be performed on the basis of a block instead of macroblock. Therefore, there are four motion vectors for the four luminance blocks in a MB. For each chrominance block, only one motion vector is used for motion compensation. It is derived by calculating the mean of the four motion vectors of luminance blocks; the result has to be rounded to the nearest half-pixel position. Four motion vectors per MB provide better motion compensation for MBs at the edge of the objects or in areas with complex motion because they are much more precise than one motion vector for each MB. This mode also includes overlapped block motion compensation, which can smooth block edges during motion compensation. Furthermore, motion vectors are permitted to point outside the picture in the same way as Unrestricted Motion Vector Mode.

PB-frame mode (Annex G) ----In PB-frame mode, two adjacent frames are treated as a unit: a PB-frame comprising a P-frame and a B-frame. Figure 2-11 illustrates the construction of PB-frame. The P-frame is predicted from the previous decoded I-frame or P-frame. B-frame is the second frame predicted bi-directionally from previous I or P-frame and the P-frame in this unit. The motion vectors of B-frames are derived from motion vectors of both of the P-frames. For a PB-frame, each macroblock consists of 6 blocks from the P-frame and 6 blocks from the B-frame. The 6 P-frame blocks are transmitted first and followed by the 6 B-frame blocks. The advantage of PB- frame mode is that the frame rate can be doubled without significantly increasing the bit rate since bi-directional prediction is more efficient than unidirectional prediction.

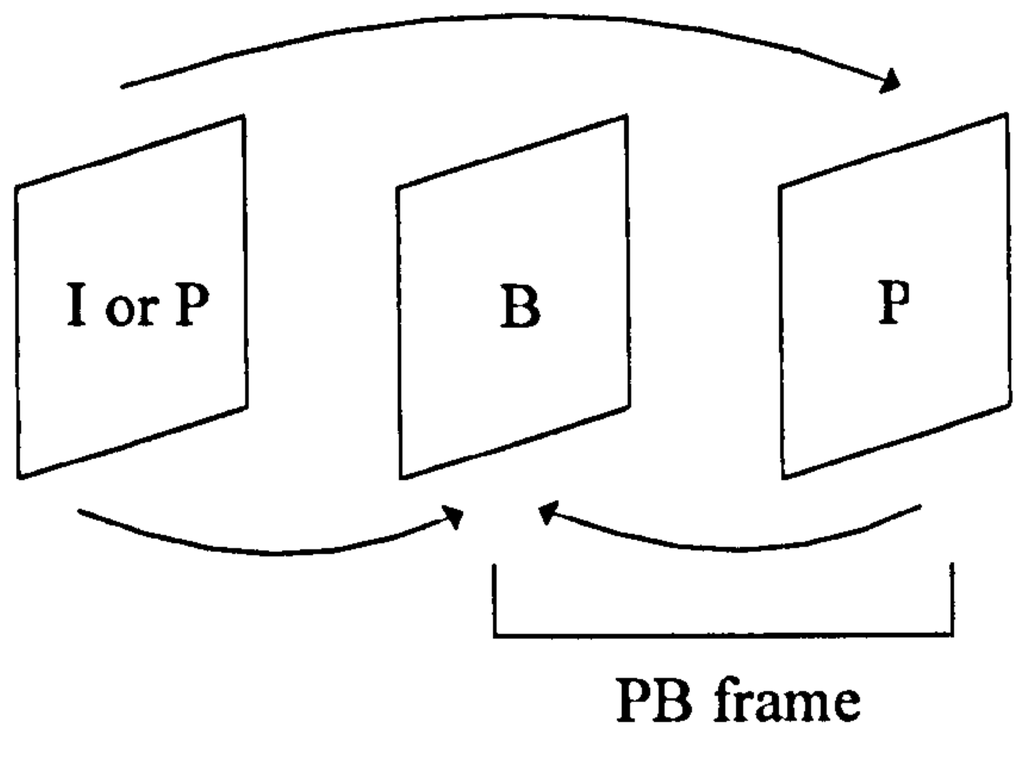


Figure 2-11 PB-frame prediction

Syntax-based arithmetic coding mode (Annex E) ----Arithmetic coding is utilized instead of variable length coding in this mode. It is shown from experiments that bitrate reduces around 3-4% for inter-coded MBs and nearly 10% for intra-coded MBs [22].

2.4.2 H.263+ and H.263++

Following the wide use of the first version of the H.263 standard, new negotiable options were added, leading to the second version of the standard, known as H.263+ [2]. H.263+ provides 12 new optional modes, grouped into three types based on their functions.

Enhancement of coding efficiency

1. **Advance INTRA coding mode (Annex I):** This annex includes a modified inverse quantisation and a separate Variable-Length Code (VLC) table for intra-block coefficients. Besides, the DC coefficients, the first column and first row of AC coefficients may be predicted spatially from DCT coefficients of neighbouring blocks.

2. **Deblocking filter mode (Annex J):** A special block edge filter is utilized to reduce the blocking artifacts of decoded frames and improve the performance of motion compensation.

3. **Improved PB frame mode (Annex M):** Besides bi-directional prediction of B picture in Annex G, this mode also supports forward and backward motion prediction for B pictures.

4. **Reference picture resampling (Annex P):** The current video frame can be predicted from a "warped" picture, which is generated by resampling and/or interpolating the decoded reference frame. This mode specifies the alteration between the current frame and the reference frame in terms of resolution, shape, size and location. It can used to adaptively change the resolution of the frames during the encoding process.

5. **Reduced resolution update mode (Annex Q):** This annex allows the encoder to code some macroblocks at a lower spatial resolution than the reference frame, by which the encoder can maintain an adequate frame rate for high motion video sequence.

6. **Alternative inter-VLC mode (Annex S):** The encoder may use the alternative VLC table used in Annex I for coding quantised coefficients in inter-coded blocks.

7. **Modified quantisation mode (Annex T):** This mode improves the flexibility of controlling quantisation step size during encoding.

Error robustness

1. **Slice structured mode (Annex K):** This annex groups a series of macroblocks (in raster order or in rectangle area of a picture) into slices. A slice is coded independently of any other slices, which makes it a resynchronization point in the bitstream, useful for limiting error propagation.

2. **Reference Picture Selection mode (Annex N):** The encoder is allowed to choose a reference picture from several previous coded pictures for macroblock motion prediction in a GOB, slice or picture.

3. **Independent segment decoding mode (Annex R):** In this annex, each segment in a video picture is decoded independently of any other video picture segments and motion vectors can not be pointed to outside of the current segment in the reference picture. When the sliced structured mode is not applied, a segment consists of a GOB or a consecutive number of GOB. A slice forms a segment if sliced structured mode is in use.

Increased flexibility

1. **Temporal, SNR and spatial scalability mode (Annex O):** This annex enables the coded video sequence to be decoded at various quality levels by using a hierarchy of pictures and enhancement pictures.

2. **Supplemental enhancement information specification (Annex L):** This annex provides some enhanced display capabilities, such as frame freeze and zoom.

H.263++ is the third version of H.263, completed in 2001. It includes three extra new modes to enhance the coding efficiency, reduce delay and improve error resilience.

1. **Enhanced reference picture selection (Annex U)** [23]: This optional mode improves coding efficiency and error resilience of Annex N by reducing the memory requirement for storing previously coded pictures and by predicting a macroblock from any pictures stored in memory.

2. **Data partitioned slice (Annex V)** [24]: This annex modifies slice structured mode (Annex K) to enhance the error resilience by rearranging the macroblock data in a slice as macroblock headers, followed by the motion vectors and all the coefficients.

3. **Additional supplemental enhancement information (Annex W)** [25]: In addition to the supplemental enhancement information defined in Annex L, this mode supports the use of an approximate IDCT and allows the user to insert a message into the coded bitstream.

2.4.3 H.264

H.264 [15] [26], previously known as H.26L, was initially started by ITU-T VCEG in 1998. In 2001, VCEG and ISO MPEG established the Joint Video Team (JVT) to take the responsibility of developing it into a standard. The standard is now finished and called officially Advanced Video Coding (AVC), also known as ITU-T H.264 and ISO MPEG-4 part 10. The main objective of the emerging H.264 is to improve coding performance and efficiency with a simple syntax specification. The basic video coding approach used in H.264 is very similar to that adopted in previous standards, such as H.263. However, new features and enhanced prediction methods make it able to provide low bit rate, low

coding delay and high complexity video coding [27]. H.264 can be applied in a variety of video applications: internet video streaming, mobile video, high definition TV, video storage on DVD and so on.

Layer structure

Figure 2-12 illustrates the layer structure of the H.264 standard, which comprises a Video Coding Layer (VCL) and Network Abstraction Layer (NAL) [26]. VCL is used to process the original video sequence and generate compressed information by discarding temporal and spatial redundancy. It is similar in principle to previous video coding standards, based on motion compensated prediction and transform coding. The compressed video data from VCL, together with some control data is sent to the NAL for formatting. The NAL reorganizes these data in a suitable format and adds headers for conveying to the transport layer. NAL has a generic format for both packet-oriented transport and bit streaming. The transport layer that NAL is connected to can be the Real-Time Protocol (RTP), systems for broadcasting or storage media.

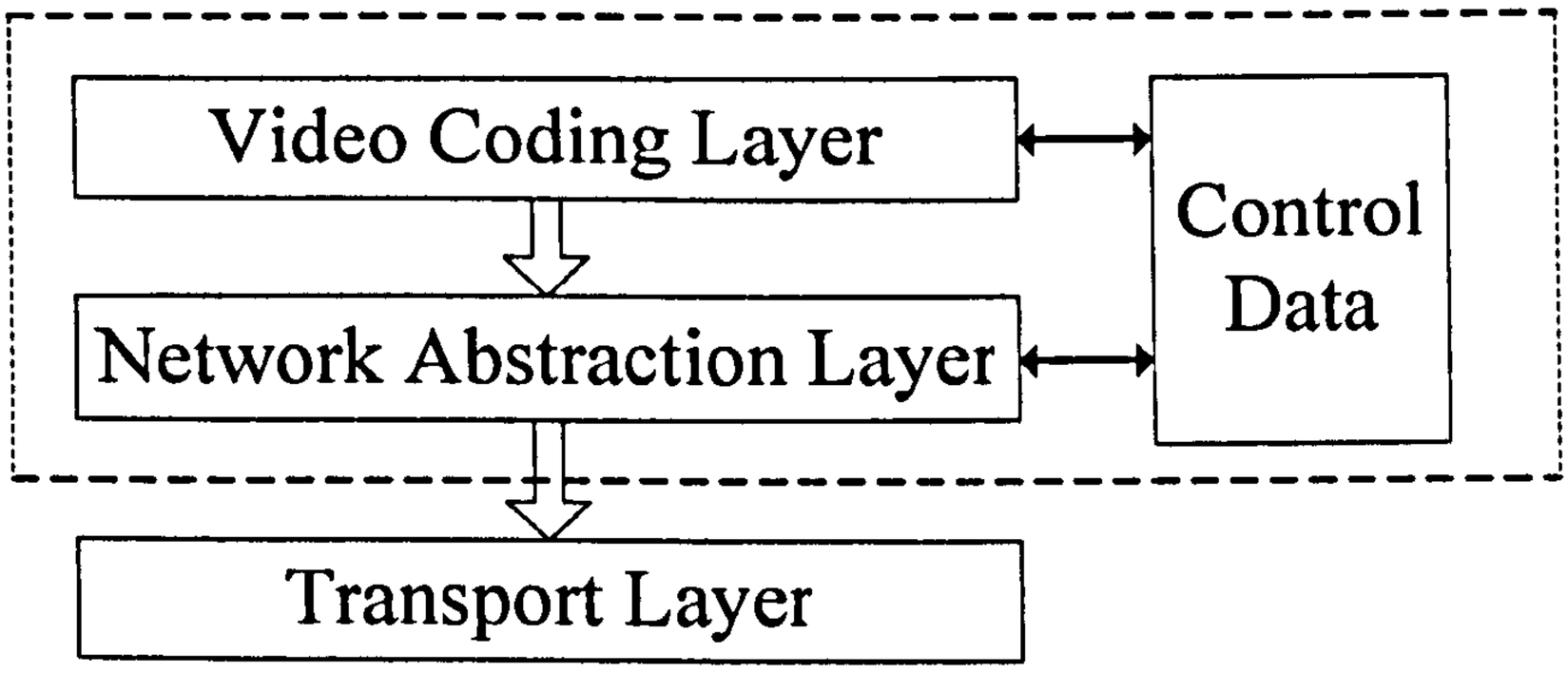


Figure 2-12 H.264 layer structure

Picture structure

In H.264, the basic unit is a 16×16 pixel matrix, called a Macroblock. A macroblock contains a 16×16 luma block and two 8×8 chroma blocks as H.264 employs 4:2:0 sampling at present. However, compared with H.263 (Section 2.4.1), chroma samples are positioned horizontally shifted by 0.5 pixel as shown in Figure 2-13. An entire picture

comprises of one or several non-overlapped slices, each containing an integer number of macroblocks. Slices in a picture are coded independently so that data damage or loss in one slice has minimal effect on decoding of the others. H.264 supports five types of slices. The first one is I slice, in which macroblocks are coded without prediction from other pictures in the sequence. P (predicted) and B (bi-predicted) slices are predicted from one previously coded picture and two pictures respectively. The two remaining slice types are Switching Prediction (SP) and Switching Intra (SI) [28] used to facilitate switching between video streams.

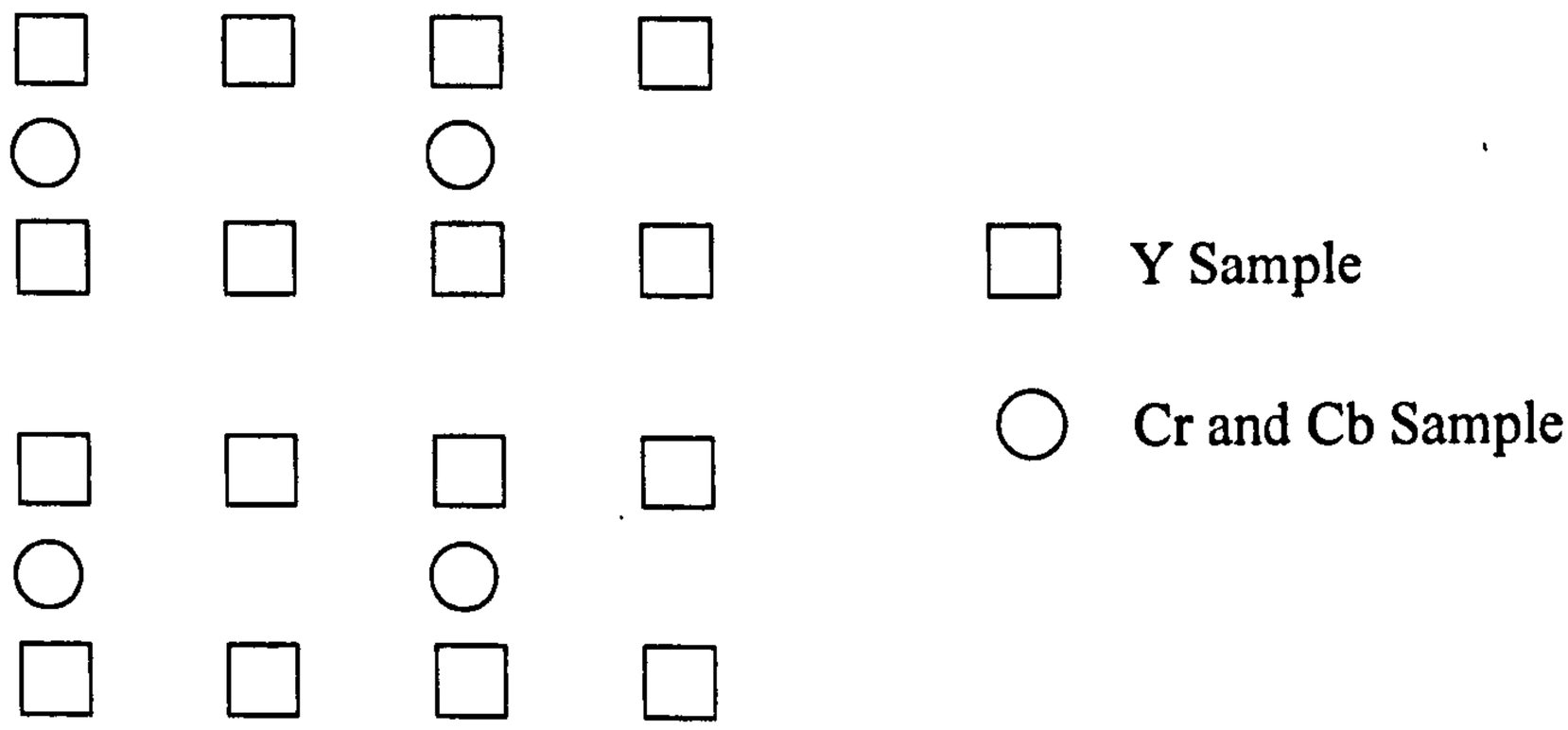


Figure 2-13 4:2:0 sampling pattern for H.264

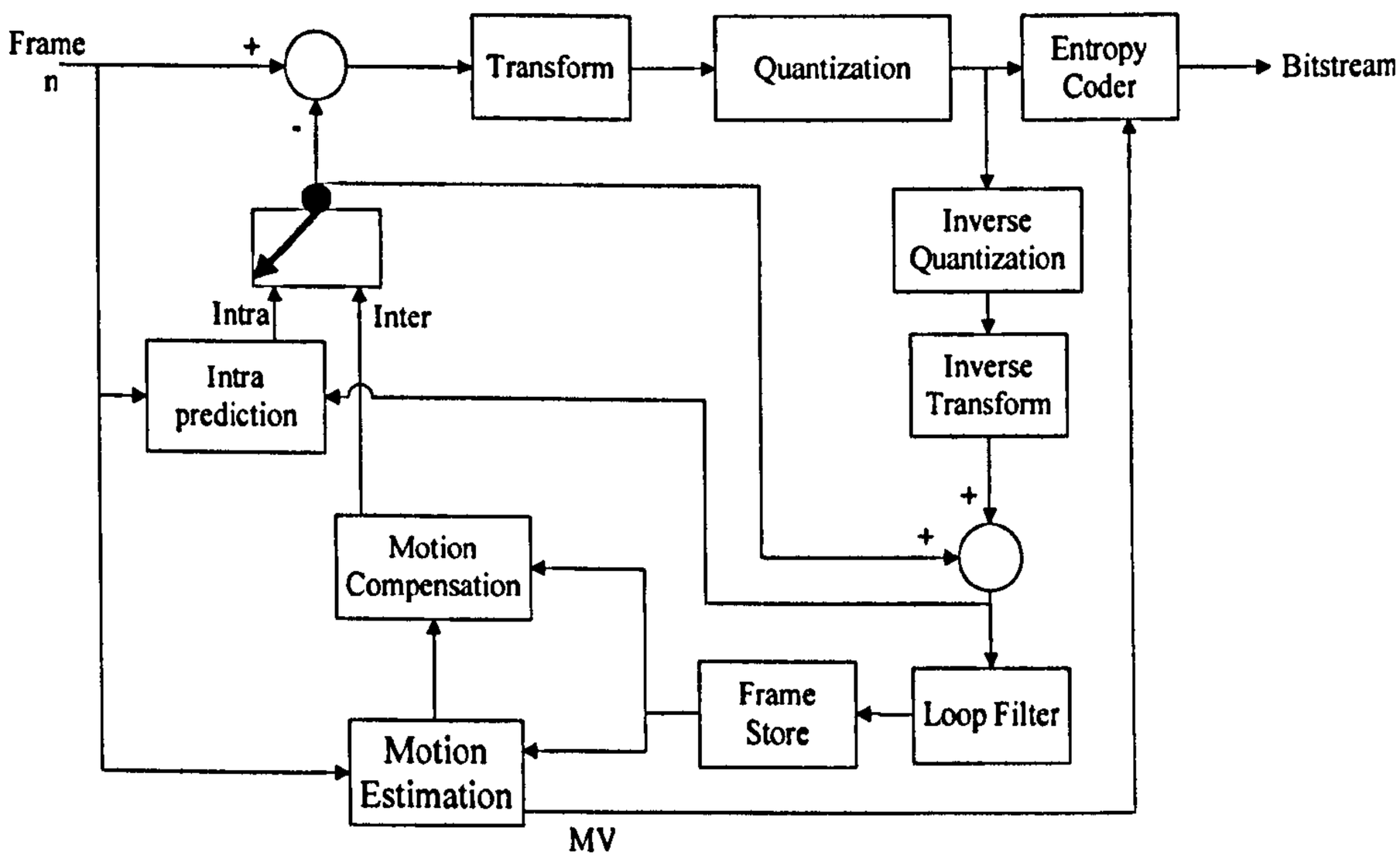


Figure 2-14 H.264 encoder

AVC encoder

Figure 2-14 is a block diagram of an encoder used in the video coding layer. It has some similarities to the generic video encoder in Figure 2-5, but the details have been changed to improve coding performance. Enhancements have been made in motion compensated prediction, intra prediction, transform and entropy coding, which will be explained in the following sections.

Enhanced motion compensation

AVC supports tree structured motion compensation, which segments a luma macroblock into a number of varying-size areas and employs these areas as the basic unit for motion estimation. At the first step, a 16×16 macroblock may be divided into small areas of one of the four types: 16×16, 8×16, 16×8, 8×8, as shown in the top row of Figure 2-15, and each small area is called a macroblock partition. When the 8×8 mode is chosen, a further segmentation may be applied to each 8×8 macroblock partition and split into macroblock sub-partitions according to sub-partition types, which are shown in the bottom row of Figure 2-15.

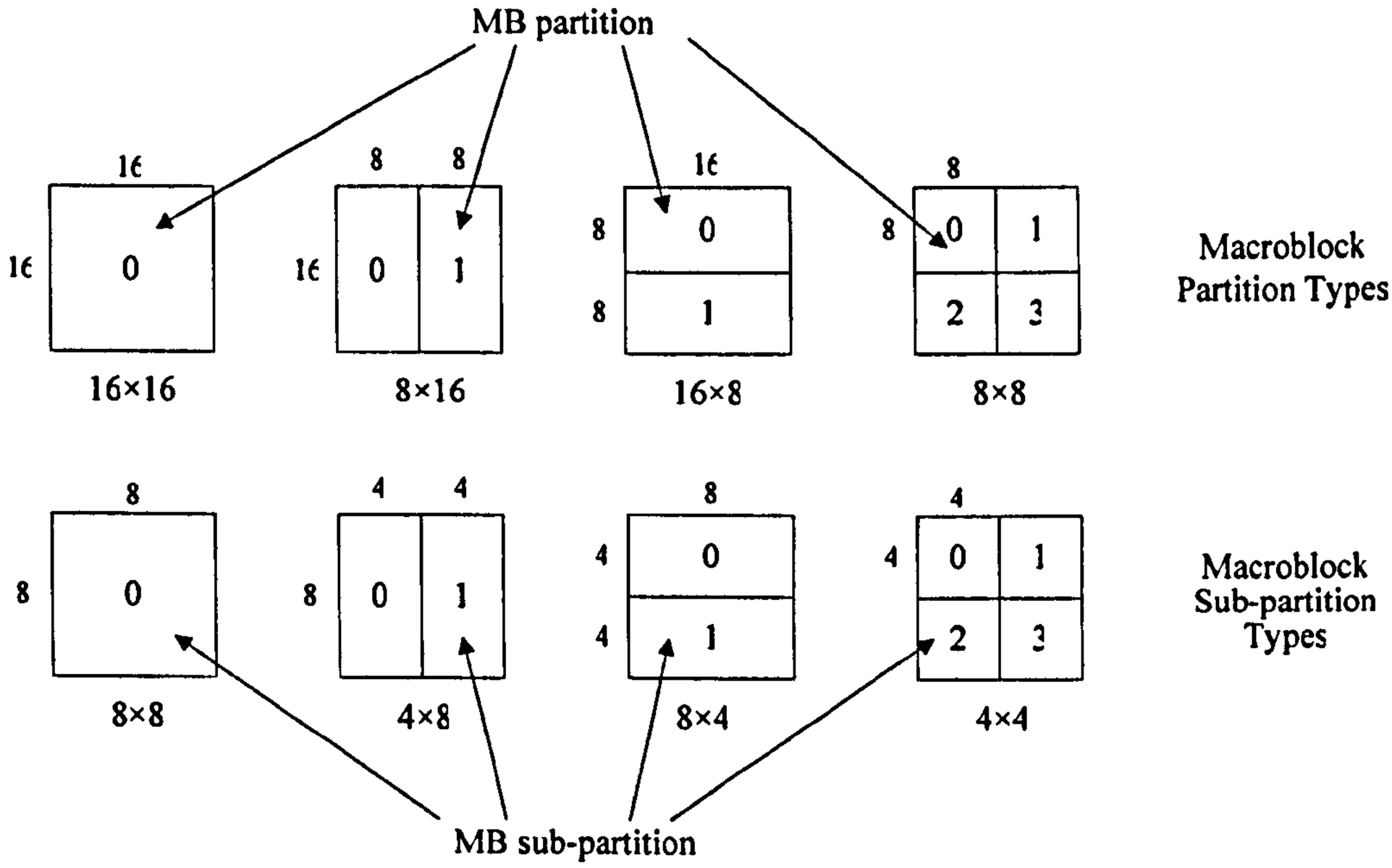


Figure 2-15 Macroblock segmentation for motion compensation

A wide range size of partitions (from 16×16 pixel to 4×4 pixel) in inter-coded MB can be used for motion estimation. H.264 employs sub-pixel motion compensation with the accuracy of ¼ luma samples. The value of ¼ and ½ sample positions are interpolated from nearby integer samples. H.264 supports multi-picture motion-compensated prediction; i.e. more than one picture can be used as reference for motion-compensated prediction. Each MB or MB partition in P-slices can only be predicted from one previously coded picture (either past or subsequent picture), which are stored in the picture buffer at the encoder and replicated by the decoder. The prediction of MB or MB partitions in B-slices can be performed from two pictures from the pre-coded pictures buffer, which may be (1) two past pictures or (2) two future pictures or (3) one past picture and one future picture.

Intra prediction

In addition to inter-coded macroblock types, H.264 has two intra coding types: INTRA-4×4 and INTRA-16×16, where each 4×4 or 16×16 luma block is predicted from previously coded neighbour blocks. The reference blocks have to be in the same slice as the current block in order to maintain independent coding of each slice. For INTRA-4×4 modes, there are nine prediction modes available for luma blocks. When INTRA-16 × 16 modes are used, four types of modes are supported and prediction is performed on an entire luma macroblock. INTRA-16×16 is suitable for smooth image areas without too much detail. Each 8×8 Chroma block in a macroblock has four types of prediction modes which are very similar to the four INTRA-16×16 modes for luma macroblocks.

Integer transform, Quantisation and Entropy coding

After inter and intra prediction, the residual macroblocks are transformed, quantised and entropy coded. Instead of the 8×8 DCT transform, H.264 uses a 4×4 integer transform [29, 30], which is based on the DCT transform, operating on every 4×4 residual block. If a macroblock is predicted using INTRA-16×16, an extra 4×4 transform is applied to all DC coefficients of luma 4×4 block. For chroma blocks, an additional 2×2 transform of DC components is applied. Compared with the conventional DCT transform, the integer transform does not produce any loss of data as it is defined exactly by integer arithmetic operation, so that inverse transform mismatch is avoided.

In the quantisation stage, a scalar quantiser is used. H.264 has 52 quantiser steps (Q) and this large dynamic range of Q results in flexible bitrate control.

H.264 supports two methods of entropy coding depending on the coding mode. For default entropy coding mode, Exp-Golomb codes are utilized for all syntax elements except for transformed coefficients, which are coded by a more sophisticated Context Adaptive Variable Length Coding (CAVLC) method. The efficiency of coding can be improved further using the optional alternative to Exp-Golomb/CAVLC: Context-Adaptive Binary Arithmetic Coding (CABAC) [31].

In-loop deblocking filter

Blocking artifacts, visible block edges in a decoded picture, are considered to be the most obvious artifact appearing in block-based video coding. H.264 employs an adaptive in-loop deblocking filter [32] to smooth the blocking around the boundary of each macroblock without affecting the sharpness of the picture. Therefore, subjective video quality is dramatically improved. Moreover, motion estimation predicted from filtered macroblocks can produce smaller residual macroblocks so that the bit rate of the filtered sequence can be reduced when compared with a non-filtered sequence.

2.4.4 MPEG-4 visual

MPEG-4 [4, 33], an ISO/IEC standard developed by MPEG, defines techniques that address a wide range of multimedia applications with high coding efficiency, interactivity, universal access, scalable complexity, error resilience and bitrates from 5kbps to above 1Gbps. It supports object-based coding of natural image, video and audio, synthetic 2-D, 3-D graphic and audio, as well as text. MPEG-4 is comprised of several parts and the visual part provides techniques and profiles for coding of natural, synthetic and hybrid natural/synthetic video. Natural video coding will be introduced in this section.

Description of visual scene

An MPEG-4 visual scene is made up of a number of Video Objects (VO), which form a hierarchical representation of an entire video sequence as shown in Figure 2-16. This representation provides easy access to video content and allows high interactivity between the coding process and user. The highest level of this structure is Video Object

Sequence (VS), which represents the complete video scene and consists of several video objects. A video object has three-dimensions: two spatial dimensions in a frame and a time dimension. Regardless of the time dimension, a VO can occupy the entire frame size or rectangular area or even an arbitrary shaped object, which is coded independently by using motion, shape, and texture techniques. A video object has at least a base Video Object Layer (VOL) and optionally a number of enhanced VOLs supporting scalable coding, resulting in a wide variety of bitrates and computational complexity levels. A time sample of a VOL is called a Video Object Plane (VOP), which defines all the information related to a video object at an instant time. Consecutive VOPs are grouped into a Group of Video Object Plane (GOV).

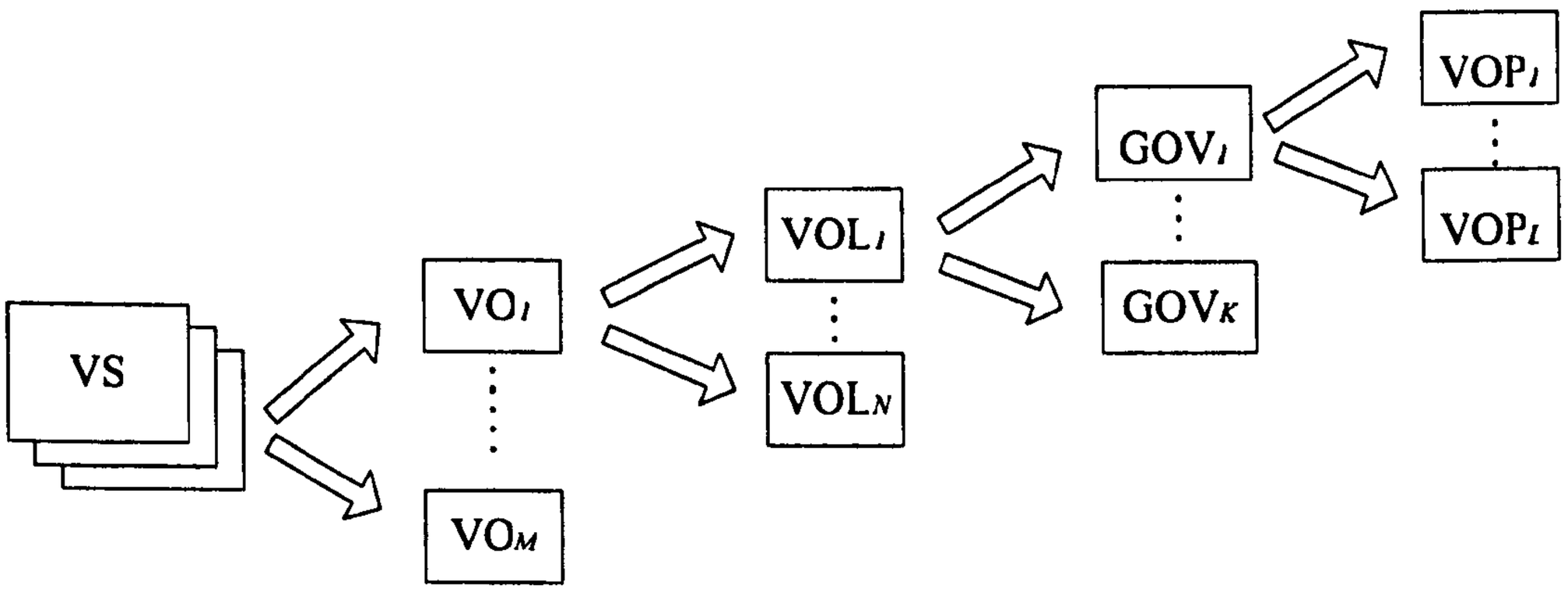


Figure 2-16 Hierarchical structure of MPEG-4 visual scene

Visual coding tools

Shape coding----Shape coding tools [34] are used to represent the borders of arbitrarily shaped objects by using extra information, which is described in a map of the same resolutions as luma signals. There are two types of shape coding tools in terms of level of transparency: binary shape coding and grey-scale shape coding. In binary shape coding, shape information of pixels belonging to a VOP is defined as "opaque" and as "transparent" for pixels outside a VOP, whereas grey-scale shape coding employs 8 bits to represent the transparency level of each pixel. Binary shape information for a macroblock is referred to as a binary alpha block (BAB), coded by using Context based Arithmetic Encoding (CAE) and motion estimation independently of the corresponding

macroblock. BABs inside a VOP have all opaque pixels and BABs crossing the boundary of a VOP have both opaque and transparent pixels.

Texture coding—The coding of motion compensated residual values is called texture coding, which includes DCT transform, quantisation and variable-length coding, in a similar way to H.263. For a boundary MB that has both opaque and transparent pixels, the transparent pixels are filled with some values based on pre-determined rules (boundary block padding), and then the boundary MB is coded in the same way as other MBs. Alternatively, a block in a boundary MB can be transformed by using shape-adaptive DCT which only processes opaque pixels and ignores transparent pixels.

Error resilience—Several techniques may be exploited to increase coding robustness. First of all, unique markers are inserted into the bitstream so that the decoder can stop decoding until the next marker when an error is found. This is called resynchronisation. In MPEG-4, texture data and motion data may be coded separately by means of data partitioning, which keeps the different types of data independent and makes it easy to apply error concealment methods on error-corrupted texture data. Finally, in the case of errors, reversible VLCs allow decoding of the bitstream both forward and backward to minimize the effect of an error.

Scalability—MPEG-4 supports four types of scalable coding. Firstly, it provides object-based scalability: the decoder can select a number of VOs instead of a full set of VOs. It is achieved naturally by the data structure used and is independent of the coding of each VO. The use of base and enhancement layer VOLs enables spatial and temporal scalability. Spatial resolution obtained from the base VOL can be improved by using one or more enhancement layers. For adjacent frames in a video sequence, temporal scalability enhances temporal resolution and smoothness of movement by increasing the frame rate. In order to provide flexible bitrates in video streaming applications, Fine Granular Scalability (FGS) truncates the bitstream during coding according to the bandwidth of the transmission channel.

Profiles and levels

Profiles [35] define the types of video objects present in the scene and a subset of corresponding coding tools that can be used to code these video objects. MPEG-4 visual

has 5 types of rectangular video objects, 7 types of arbitrarily shape video objects and 12 kinds of profiles, which use one or more video objects. For example, the Simple Profile only includes simple video objects, but the Advanced Simple Profile has advanced simple and FGS video objects. The Simple profile is almost identical with the baseline CODEC of H.263 described in 2.4.1. Levels in a profile specify several parameters of video bitstream, such as scene size, bit rate, maximum number of objects and so on.

2.4.5 Other MPEG standards

MPEG-1

The first standard developed by MPEG was MPEG-1 [36], coding video and associated audio information. MPEG-1 consists of three parts: system, video and audio and has been optimised for video quality at bitrates of 1.1 to 1.5 Mbps, to provide a mean of storing digital audiovisual data on CD (with a playback rate of around 1.4 Mbps) with nearly the same quality as VHS cassettes.

MPEG-2

In order to improve the quality of MPEG-1, MPEG-2 [3] was developed to support a wide range of resolutions and bitrates, including digital television and high definition television. The MPEG-2 standard has various parts, which cover video, audio and all aspects related to representation and delivery.

MPEG-7

MPEG-7 [37], formally called the "Multimedia Content Description Interface" is aimed at developing a standard method to describe multimedia information. Unlike previous MPEG standards, it concentrates on representing the content of the information, but not on the information itself. Therefore, multimedia information can be searched in a similar way to text.

MPEG-21

The aim of MPEG-21 [38] is to define a standardized open framework for multimedia delivery and consumption by all communities across a wide range of networks and devices. This framework enables transparent and augmented use of multimedia resources

and covers the entire chain consisting of content creation, production, distribution and trade.

2.5 Complexity, rate and distortion

2.5.1 Performance evaluation of video encoders

When evaluating the performance of video encoders, the bitrate of the coded video sequence, the video quality and the algorithm complexity used in the CODEC are the most important performance constraints that need to be considered. Changing coding parameters, adding optional coding modes and choosing various coding algorithms affect the output of video encoders, resulting in different levels of visual quality, bitrate and computation. In general, good quality video requires a complicated coding scheme and/or high bitrate.

Visual quality and bit rate are two related parameters. In many applications, the bit rate of a coded video sequence is limited by the characteristics of transmission channels. In order to achieve the optimum coding performance, it is necessary to obtain the highest visual quality subject to the bit rate constraint, which is called rate-distortion optimisation. Many strategies have been developed to improve the rate distortion performance. Near-optimum strategies tend to be highly complex, requiring a large amount of computation.

For an increasing number of applications, it is necessary to take into account of the computational complexity of the video encoder, such as in real-time and/or power-constrained applications. In these cases, the performance of a video encoder may be limited by the amount of processing power available as well as, or rather than, the available transmission bandwidth, and so there is a need to manage the complexity to meet the constraints of processing power. Moreover, rate-distortion strategies also have to work under the constraint of computational complexity, which leads to the emergence of complexity-rate-distortion theory [39, 40]. This theory is proposed to optimise the video quality subject to the bitrate and complexity constraints.

2.5.2 Rate-distortion control

The bit rates produced by a generic video encoder (using motion-predicted compensation and DCT) fluctuate naturally. Some existing networks, such as Public Switched

Telephone Network (PSTN) and Integrated Services Digital Network (ISDN), provide fixed bit rate channels. In order to transmit compressed video data (with its variable bit rate) through these Constant Bit Rate (CBR) channels, a channel buffer is usually added to smooth out the Variable Bit Rate (VBR) video stream and rate control techniques are applied to prevent buffer underflow and overflow. Theoretically Variable Bit Rate (VBR) networks, such as, Ethernet and Asynchronous Transfer Mode (ATM), can cope with VBR data, but in practice, the available capacity of the channel can not always satisfy the bitrate requirement of transmitted data due to practical limitations (e.g. congestion) and thus the bit rate has to remain within a certain range. In this scenario, it is necessary to reduce the bit rate variation to meet channel constraints using rate control schemes.

The rate control problem has been investigated for nearly 50 years in the literature [41-43] and the main concern is to trade off bitrate and obtained video quality. That is, given the desired bit rate, how to encode the video sequence to achieve the highest objective and subjective video quality? A classical rate-distortion problem is shown in Equation 2-5 : minimize distortion (D) under the bitrate constraint (R_t), which can also be interpreted as how to efficiently allocate a limited bit rate to each frame and macroblock.

$$\textbf{Min (D), When } R < R_t$$

Equation 2-5

Many practical algorithms have been developed to effectively keep bit rate within the target whilst optimising video quality by adjusting quantisation step size (Q). The operations of these algorithms generally can be split into three steps: (1) Calculate and allocate the available bit rate for each frame, or macroblock (MB). (2) Compute Q for each frame or MB based on allocated bit rate. (3) Apply Q in encoding of a frame or MB to achieve the target. Several algorithms that have been proved to be efficient and applied in the popular video compression standards are introduced below.

TMN5

The Test Model 5 (TMN5) [44] rate control scheme proposed in H.263 mainly focuses on low bit rate and real-time CBR video applications with low latency. Due to tight buffer size imposed by strict requirements on delay, it is necessary to control the bit rate within a small amount of variation. TMN5 rate control consists of output buffer regulation and

updating of quantisation step size, the details of which are stated in the following procedures:

1. Allocate the same number of bits (B_t) (equal to target bit rate (R_t) divided by target frame rate) to each frame.

2. Update quantiser (Q) for current frame at the beginning of each new group of macroblocks based on Equation 2-6. $\overline{Q_{i-1}}$ is average quantiser for previous frame. B_{i-1} is the number of bits occupied by the last frame, the number of bits spent for current frame so far is represented by B_{imb}, mb is the present Macroblock number and MB is the total number of Macroblocks in a frame.

$$Q = \overline{Q_{i-1}}\left(1 + \frac{B_1}{2B_t} + \frac{12B_2}{R_t}\right),$$

$$B_1 = B_{i-1} - B_t \text{ and } B_2 = B_{imb} - \frac{mb}{MB}B_t$$

Equation 2-6

After encoding each frame, the number of bits spent on this frame is added to the buffer. If the buffer size is bigger than a threshold, the next frame is skipped.

This method can prevent buffer overflow, but at same time, it can not guarantee a minimum frame rate and delay.

TMN8

The rate control scheme employed in H.263 test model 8 (TMN8) [45] is based on a Lagrange optimisation technique presented in [46]. TMN8 rate control consists of two layers: frame layer and macroblock layer rate control. In the frame layer, a number of bits is allocated to a single frame. The new quantisation step size is computed for each macroblock at the macroblock layer. The operation of TMN8 rate control is very similar to that of TMN5 except for the following points:

1. TMN8 updates the same number of bits allocated to the next frame based on the number of remaining bits in the buffer, whereas TMN5 assigns the same target to every picture.

2. The selection of the quantiser is made for each macroblock in TMN8 rather than each GOB (used by TMN5).

TMN8 allocates the number of bits based on past coding information and provides more accurate updating of the quantiser than TMN5, which makes it achieve better rate-distortion performance.

VM18

The rate control scheme used in MPEG-4 verification model 18 [47] consists of three parts: frame layer, macroblock layer and multiple video object rate control. The frame layer rate control scheme, also called Scalable Rate Control (SRC), is based on a quadratic rate distortion model described in [48]. SRC is designed to achieve the target bit rate for a single Video Object (VO) and the same quantisation step is used for all the macroblocks in a video object plane. In the macroblock layer, the quantiser is updated for each single macroblock. The multiple video object rate control algorithm is proposed to control the bit rate for more than one VO in a scene.

The SRC scheme utilizes a rate distortion model described in Equation 2-7:

$$R = \frac{X_1 S}{Q} + \frac{X_2 S}{Q^2}$$

Equation 2-7

X_1 and X_2 are modelling parameters, which are updated according to the coding result of the current frame. S is the Mean Absolute Difference (MAD) of the residual value of the current frame after motion estimation. Target bit rate (R) for each frame is derived from the available bits and the bits occupied by the previous frame. The Quantiser (Q) is calculated based on the above equation and applied to obtain the target bit rate.

Compared with the TMN8 rate control algorithm used in H.263, SRC has the following significant improvements:

- It supports control of various bitrates, frame sizes and frame rates, thus it can satisfy constraints of both CBR and VBR channels at frame-layer level.

- In a CBR application, it can vary the frame rate in order to target bit rate.

2.5.3 Complexity-distortion control

As one of the performance constraints, the computational complexity of video encoders has been a concern since the introduction of video compression, especially when it comes to practical implementation and applications. This is a problem of trade-off between visual quality and algorithm complexity: (a) either increasing video quality at the expense of higher computations or (b) reducing the complexity with corresponding degradation in visual quality. The choice of the level of quality and complexity are based on the requirements of the practical application. For example, when encoding a movie and storing it on a DVD, the main concern is to obtain best visual quality and meet the bitrate requirement. Coding is performed offline so that complexity does not matter. Thus, complicated algorithms can be applied to achieve the target. On the other hand, in the scenario of a real time application, coding time becomes a significant factor and therefore algorithm complexity has to be taken into account. Researchers have developed many methods and techniques in order to solve the trade-off problems (a) or (b) and some of them are listed below:

For trade-off problem (a)

- Full motion search algorithm: theoretically it can find the best match as it searches the entire search area, but requires a lot of computation.

- Variable block size in motion estimation: H.264 supports different block size for motion search from 16×16 pixels to 4×4 pixels. The variety of block size makes motion search more efficient and accurate, resulting in improvement of coding performance. However, it significantly increases complexity.

- Optional modes in H.263 as introduced in section 2.4.2: these modes can add extra computations to encoding while increasing video quality.

For trade-off problem (b)

- Fast motion estimation search algorithm: decreases the complexity by searching in a small area or reduces some of the search points, resulting in inaccuracy compared with the full search algorithm.

- Approximate DCT transform: it speeds up the transform by indirect or direct fast transform [49], filtering the input signal [50] or calculating only low frequency components [51], at the same time causing loss of video quality.

With increased processor speed and application of software CODECs, there are emerging requirements not only to reduce the computational complexity, but also to adaptively control the trade-off between complexity and video quality, which can be interpreted as: trying to minimise distortion (D) subject to complexity constraint (C_t)

$$\textbf{Min (D), When } C < C_t$$

Equation 2-8

2.5.4 Complexity-rate-distortion control

Rate-distortion achieves good decoded visual quality by efficient allocation of available bits. Various rate control algorithms have different levels of computational complexity. The performance comparison of these algorithms has to be made based on the same computation expenses. Complexity-distortion control, analogous to rate-distortion control, provides scalable control of computations and minimizes quality distortion within a complexity boundary. The reduction of computation affects not only video quality but also the bitrate of the decoded video sequence. Therefore, it is necessary to explore the relationship among the three parameters and give rise to a new theory: Complexity-Rate-Distortion control. Researchers [52-54] have addressed the tradeoffs between complexity, bitrate and quality. Coding a video sequence at various rates R and complexities C, the video quality function D(R,C) can be represented by a three-dimension convex surface[55]. Figure 2-17 shows a complexity-rate-distortion surface of "Carphone" video sequence coded by using the complexity reduction algorithm introduced in 7.3.4.

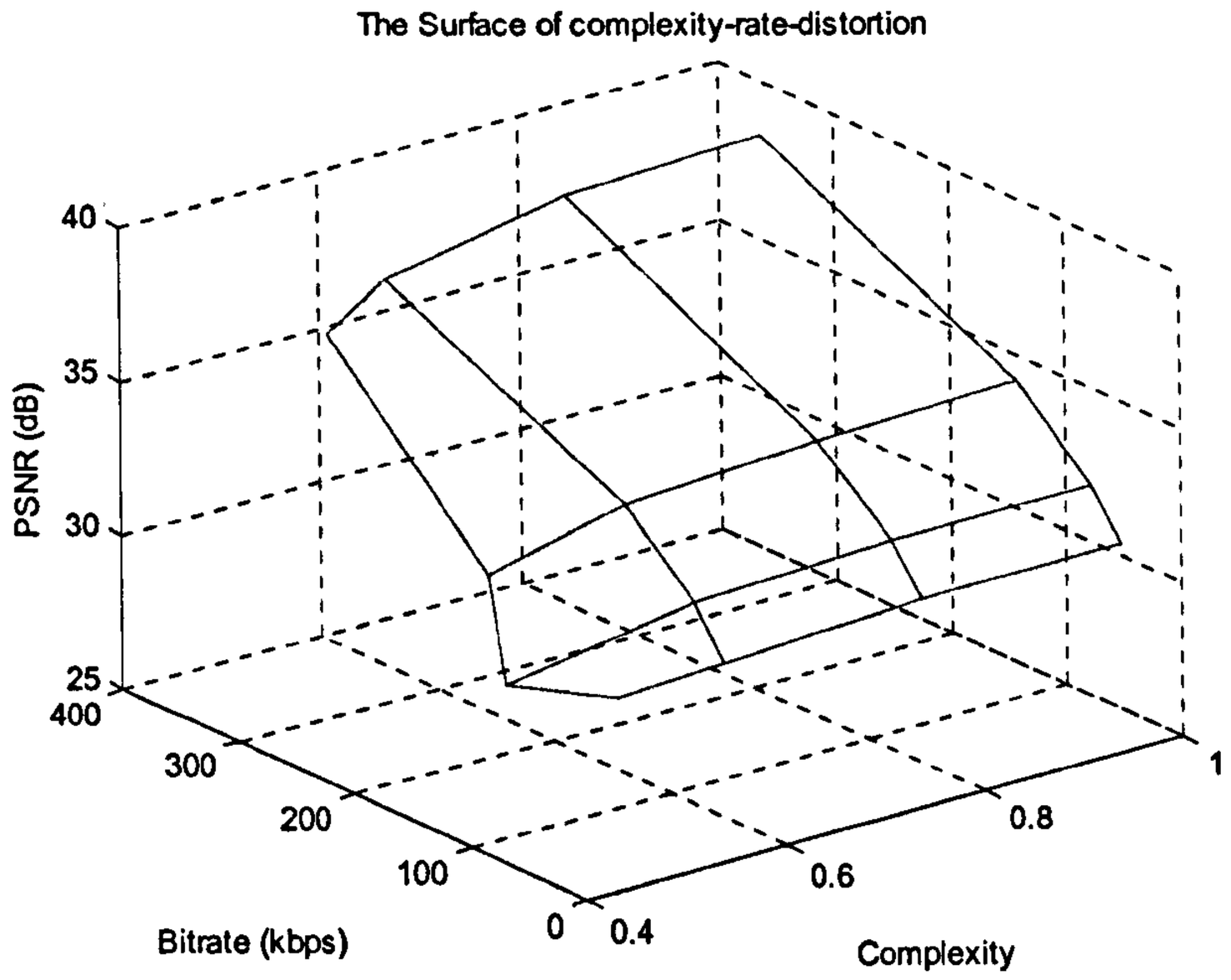


Figure 2-17 Complexity-rate-distortion surface

Given a target R_t and C_t, there is a point on the surface which satisfies the R and C requirement as well as having the highest video quality as illustrated in Equation 2-9.

$$\text{Min (D), When } C < C_t \text{ and } R < R_t$$

Equation 2-9

2.6 Summary

In this chapter, the fundamental technology for video coding and the most well known video coding standards are discussed. Software-only implemented video CODECs enable a wider range of application due to their scalability and low-cost. The most important problem in a software-based video encoder is how to optimise the achieved video quality subject to the communication bandwidth and computational complexity constraints. Many approaches have been proposed in the literature to minimize quality distortion within a bitrate limitation. The later chapters of this thesis address the trade-off problem between tcomputational complexity and video quality. The techniques developed aim to adaptively control computational complexity while achieving the maximum video quality.

Chapter 3 Experimental method

3.1 Introduction

Proposed approaches to manage the computational complexity of a video encoder are investigated in Chapter 4-Chapter 7. The experimental system employed in those chapters can be described by a generic diagram, as shown in Figure 3-1. This chapter introduces test environments and methods, including test video sequences, platform, video CODECs, video quality assessment methods and complexity measurement methods.

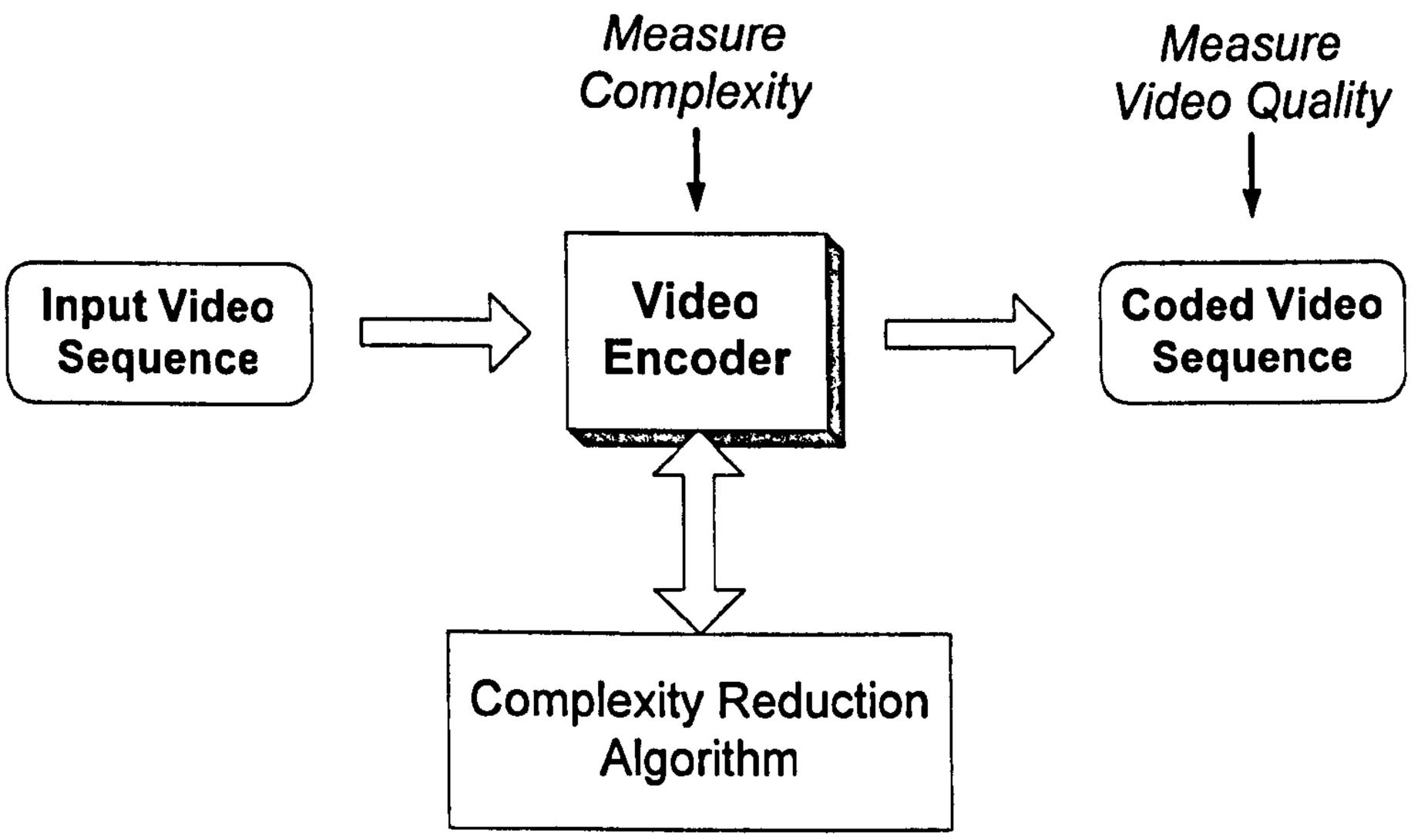


Figure 3-1 Block diagram of experiments

3.2 Test environment

Platform

The experiments described in the following chapters were carried out on two workstations:

- Dell workstation with Intel 800MHZ processor and 256Mbytes memory, running Windows 2000.

- Samsung laptop with Intel Pentium IIII-M 1.7GHZ processor and 512Mbyes Memory, running Windows XP.

Test video sequences

Test sequences are chosen from ITU test video materials widely used by researchers and scientists with the goal of covering a range of content detail, object motion and different types of background as well as camera movement. The picture format of the selected video clips is QCIF as introduced in Table 2-2.

- Claire: This is a scene with very low motion and detail. A lady sits in front of a fixed single-colour background, talking to a camera with a few facial and head movements. (Figure 3-2)

- Mother and Daughter: This scene contains a mother with a little daughter sitting on her lap with a wall and painting as background. The scene contains moderate detail and facial motion as well as head and hand movement. (Figure 3-3)

- Carphone: A man sitting in a moving car, with a lot of head and shoulder movements and exaggerated facial expressions. The interior of the car does not change significantly, but the scene seen through the car window varies while the car is moving. Moreover, the picture is affected by camera shake. (Figure 3-4)

- Foreman: A worker talking to the camera with animated facial expression and hand gestures. The camera pans from the worker (Figure 3-5 (a)) to a construction site (Figure 3-5 (b)) causing significant movement of the whole picture.

Figure 3-2 Example frame of "Claire" video sequence (frame 5)

Figure 3-3 Example frame of "Mother and Daughter" sequence (frame 5)

Figure 3-4 Example frame of "Carphone" sequence (frame 5)

(a) **(b)**

Figure 3-5 Example frames of "foreman" video sequence: (a) Frame 150 (b) Frame 220

These sequences reflect a wide range of video with different properties and behaviour: from low detail to high detail scenes, from moderate to high movement, from fixed to

changing background, etc. It is possible to systematically assess the performance of a CODEC or proposed method by using these test sequences,

3.3 Video CODECs

The encoder and decoder used in the experiments are an H.263 CODEC based on Test Model TMN5 [44] and H.263+ CODEC based on TMN8 [45] (or low complexity mode of TMN10), which were developed by University of British Columbia, Canada. These software-only CODECs are written in C language and run under the Microsoft Windows operating system.

The H.263 encoder has the following features:

- Supports the 4:2:0 Y Cr Cb format (Section 2.4.1)
- Includes a normal DCT transform (Section2.3.2) and a fast DCT transform as described in [56].
- Motion estimation including integer motion search (full spiral search algorithm [57]) and half-pixel search (Section 2.4.1).
- Two rate control methods: offline rate control and TMN5 (Section 2.5.2)

Compared with H.263, the H.263+ encoder includes additional features:

- Use Nearest Neighbour Search (NNS) as optional fast motion estimation algorithm, which is introduced in Section 6.2.
- Three rate control methods: offline rate control (Quantiser step size is updated once per frame), TMN5 and TMN8 (Section 2.5.2)

3.4 Video quality assessment

In order to assess the quality of decoded video sequences, two types of quality measurement are employed in the experiments. Objective testing is used to give a numeric assessment of video quality, compared with a reference video sequence. Subjective testing is carried out to measure perceived video quality based on opinions of representative users.

3.4.1 Objective test methods

Objective video quality test methods are based on statistics measured between the two video sequences, such as average difference, maximum error, Mean Squared Error (MSE) and so on, among which, MSE is the most popular one. As shown in Equation 3-1, MSE calculates the average of squared difference between test video sequence (V_t) and (V_o) original video sequence, where N is the total number of pixels in a single picture. For a colour video, MSE can be calculated separately for luminance component, or each of the chrominance components.

$$MSE = \frac{\sum_{i=1}^{N} (V_o - V_t)^2}{N}$$

Equation 3-1

In practice, most video coding systems more often use Peak Signal Noise Ratio (PSNR) rather than MSE to assess the video quality, which is illustrated in Equation 3-2, where n is the number of bits used to represent a sample. In this thesis, PSNR of the luminance component is utilized to measure objective video quality.

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

Equation 3-2

The advantage of PSNR is its simplicity and ease of computation, but it is not always consistent with perceived video quality because it only considers the MSE of part of signal (either luma or chroma) of a frame without taking account the Human Visual System (HVS) factor. Experimental results show that many HVS features influence perceived video quality; consequently, incorporating these features into video quality assessment has been suggested in the literature [58-60]. However, due to the high computational complexity of the HVS based measurement and the difficulty of reliably simulating 'real' HVS response, it has not been widely implemented. Therefore, in addition to PSNR, subjective testing is also carried out to test video quality in this thesis.

3.4.2 Subjective test methods

The aim of the subjective testing in these experiments is to obtain a clear opinion from audiences about the perceived quality of test video sequences (coded and decoded by our algorithms). ITU-T recommendation P.910 [61] defines subjective video quality assessment methods for multimedia applications, such as video telephony, video conferencing etc. Two methods selected from P.910 are employed because the specified requirement of digital images (picture format, frame rate and bit rate) are suitable for the experiments in this work. One is the Degradation Category Rating method (DCR), where a reference sequence (source sequence without coding) and test video sequence are assessed in pairs. When the QCIF picture format is used, it is suggested that they should be presented on the same monitor as shown in Figure 3-6: the reference sequence is always positioned to the left of the sequence under test. After seeing each pair of sequences, a non-expert viewer will be asked to grade the degradation of the test sequence relative to the reference sequence. The levels of the degradation are listed in Table 3-1.

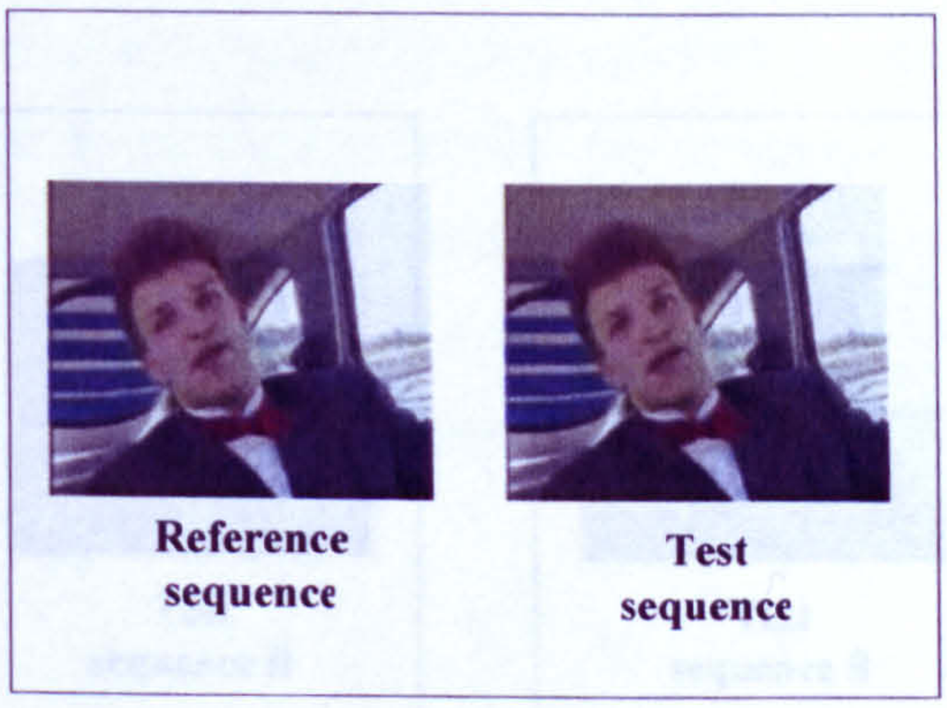


Figure 3-6 Reference and test sequences position of DCR method

Table 3-1 Grading table for Degradation Category Rating method

Grades	Descriptions
5	Imperceptible
4	Perceptible but not annoying
3	Slightly annoying
2	Annoying
1	Very annoying

The Pair Comparison (PC) Method also tests a pair of video sequences on the same screen, but they are shown in two relative positions: (1) test sequence A on the left and test sequence B on the right (2) test sequence A on the right and test sequence B on the left as illustrated in Figure 3-7. The viewer chooses the sequence he/she prefers in terms of perceived quality.

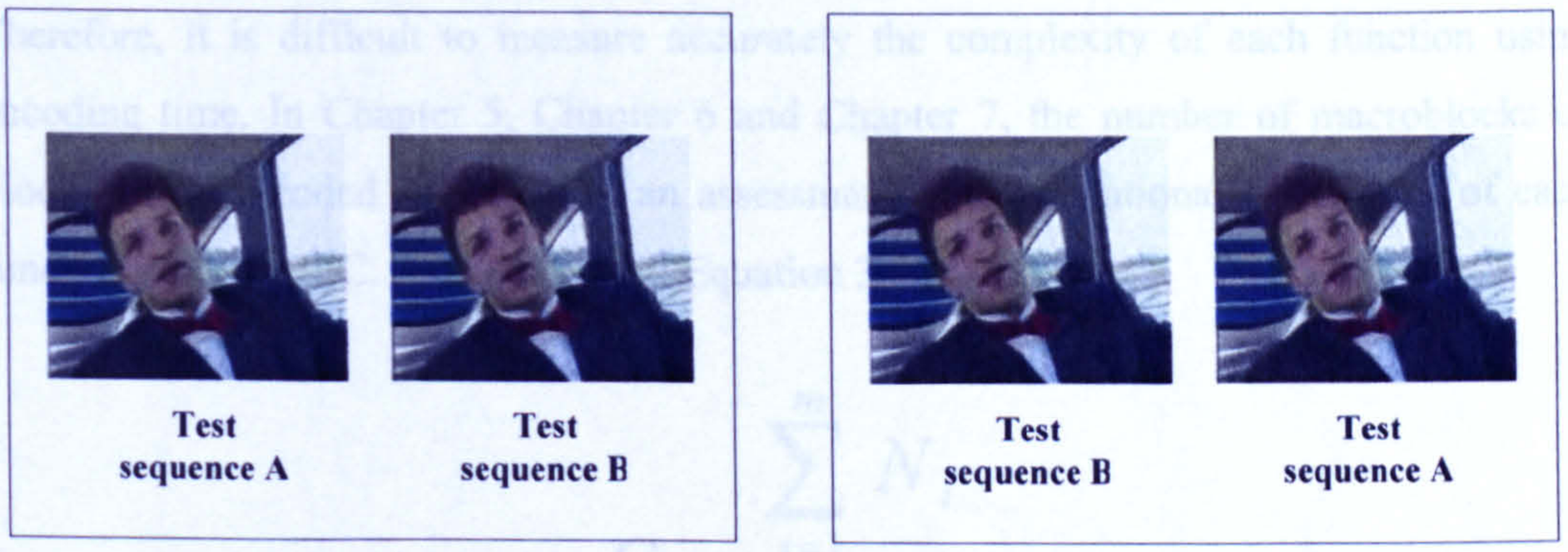


Figure 3-7 Reference and test sequence position of Pair Comparison method

In the assessment of either DCR or PC, both reference sequence and test sequence are displayed on the same screen simultaneously. They are the same length and start and stop at the exactly same video frame. The viewers have to make their decision in less than ten seconds. Generally, the number of viewers required for testing each pair of video sequence will be from 4 to 40. In the subjective testing of this work, 15 viewers

participated in the experiments for evaluating each pair of scenarios and the test conditions are set according to the rules described in [61].

3.5 Computational complexity measurement

Computational complexity is another important performance-related parameter that needs to be evaluated. Three different methods have been chosen to calculate complexity depending on the experimental situation.

Encoding time

Encoding time is a direct measurement of the algorithm complexity in software-only encoders. In order to estimate the complexity of different types of encoder, the time spent on a encoding video sequence (in milliseconds) is recorded and utilized to compare the complexities in Chapter 4.

Number of processed blocks

Encoding time depends on the processing speed of the workstation used for simulation and the time cost for a single function during encoding a single frame is very small. Therefore, it is difficult to measure accurately the complexity of each function using encoding time. In Chapter 5, Chapter 6 and Chapter 7, the number of macroblocks or blocks actually coded is chosen as an assessment of computational complexity of each function in a CODEC. As described in Equation 3-3,

$$C = \frac{\sum_{i=1}^{m} N_i}{\sum_{i=1}^{m} B_i}$$

Equation 3-3

Where B_i is the total number of MBs or blocks in frame i, N_i represents the number of MBs or blocks of frame i that have been coded and *m* is the total number of frames in a video sequence.

The computational complexity (C) is defined as the result of the sum of N_i in a video sequence over the total number of macroblocks or blocks of this entire sequence. For the reference video CODEC without our computation reduction algorithms, the complexity is equal to one for every function as coding is carried out on every macroblock. If the complexity of the CODEC with our proposed algorithm is less than 1, a computation saving has been achieved.

Profiling

The third way to calculate the computational complexity of the main functions is profiling. The profiling function included in Microsoft Visual C++ is used to measure the total time coverage of each individual functions whilst encoding a video sequence. This provides accurate information about processor utilization.

The extra complexity of each proposed algorithm is calculated in terms of basic operations used in the computer, including addition, multiplication, shift and comparison.

Chapter 4 Computational complexity of video encoders

4.1 Introduction

Popular video coding standards (Chapter 2) employ block based video coding techniques, where the encoding process consists of motion estimation and compensation, DCT, Quantisation and Entropy coding. A video encoder also includes a decoding process (inverse DCT and inverse quantisation) in order to decode the current frame and reconstruct it for motion compensation of the next frame. Some of these functions are computationally complex, requiring significant processing power. Real-time implementation of a video encoder with a normal frame rate (25 frames/s) and good visual quality can be achieved by using hardware, such as a dedicated video processor. With the increase in speed of general-purpose processors, the implementation of software-only video encoders becomes more and more feasible. However, it is still the computational complexity of these functions that determines the frame rate and video quality that a software-implemented video encoder can achieve. Compared with encoding, the decoding process is considerably simpler because it does not require motion estimation, DCT or quantisation. In this chapter, the computational complexity of software-only video encoders is investigated in order to find out the computation cost of encoding functions. The Rate-distortion performance of H.263 and H.263+ are examined and the correlations between algorithm complexity and rate-distortion performance are discussed.

4.2 Computational complexity analysis of video encoder

Profile tests (Section 3.5) were performed on a PC with an Intel Pentium IIII mobile processor (1.7 GHZ) in order to examine the computational complexity of the main functions in a video encoder. The actual time spent on each function was measured in milliseconds. Processor utilization (P_u), is given by dividing the time cost of each function ($T_{function}$) by total encoding time (T_{total}) (Equation 4-1), and indicates the computational complexity of each function. There are two types of video encoders employed in the experiments: an H.263 encoder based on TMN5, and an H.263+

encoders based on TMN8 (see Section 3.3). The baseline encoder of each is almost identical: they both use the fast forward and inverse DCT algorithm described in [56] and full motion estimation search algorithm. The following fast motion estimation algorithms are selected as an alternative to full search for each type of encoder: the Three Step Search algorithm [12] is chosen for the TMN5 encoder and TMN8 encoder utilizes the nearest neighbour search described in Section 6.2. Integer search window size is set to +/- 7 Pixels for both full search and fast search algorithms and half pixel motion search function is switched on, resulting in +/-7.5 pixels total search area.

$$P_u = \frac{T_{function}}{T_{total}}\%$$

Equation 4-1

The "Carphone" video sequence (300 frames) is coded by TMN5 and TMN8 with Q = 8 and various motion search algorithms. The total time spent on encoding the video sequence and the percentage of processing power spent on the main functions are summarized in Table 4-1 and Table 4-2 for TMN5 and TMN8 respectively. Since DCT, quantisation, inverse DCT and inverse quantisation are related functions (processing the residual data from motion compensation), they are grouped together for evaluation of processor utilization in the table. Motion estimation in the table includes both integer and half-pixel motion search. Except for the file input/output functions and statistical information calculation functions (as they are not included in a real-time encoder), other functions are grouped together as "Remaining functions".

Table 4-1 Profiling result of H.263 (TMN5)

	Processor Utilization	
	With Full search	With Three-step search
DCT, IDCT, Quant and Dequant	15.9%	26.8%
Motion estimation (integer and half-pixel)	63.8%	38.2%
Remaining functions	20.3%	35.0%
Total coding time(Seconds)	14.766	8.796

Table 4-2 Profiling result of H.263+ (TMN8)

	Processor Utilization	
	With Full search	With Nearest neighbour search
DCT, IDCT, Quant and Dequant	12.6%	21.4%
Motion estimation (integer and half-pixel)	63.6%	38.7%
Remaining functions	23.8%	39.9%
Total coding time(Seconds)	14.924	8.807

It is clear that motion estimation takes up more than half of the processing power for both TMN5 and TMN8 from Table 4-1 and Table 4-2 when the full spiral search is employed. Thus motion estimation is an extremely computationally expensive function. By using a fast search algorithm, the encoding time decreases from 14.766 seconds to 8.796 seconds for TMN5 and from 14.924 seconds to 8.807 seconds for TMN8. The total processing power consumed by DCT and related functions as well as motion estimation is nearly 65% for TMN5 with three-step search and 60% for TMN8 with nearest neighbour fast

motion search; therefore, they are the most computational complex functions in a video encoder.

The profiling results depend partly on the chosen processor and video sequences, but the experimental results clearly demonstrate that DCT and related functions and motion estimation are the most computationally expensive functions in a video encoder. These functions are therefore good candidates for complexity optimisation.

4.3 Rate-distortion performance

The video sequence "Carphone" was encoded using H.263 (TMN5) with either full motion estimation search algorithm or three-step search. The quantiser step size (Q) was changed from 4 to 16 with increments of 4 and the PSNR and bitrate were recorded at each setting. Similar experiments were carried out for H.263+ (TMN8) with the full search algorithm and the nearest neighbour search algorithm.

Figure 4-1 and Figure 4-2 illustrate the rate-distortion curve (PSNR against bit rate) for TMN5 and TMN8 respectively. It is obvious that the full search algorithm outperforms fast search algorithms through the entire range of bit rate for both TMN5 and TMN8. Since TMN5 and TMN 8 employ the same baseline CODEC and the same kind of full motion search algorithm, the full search curves for both of them are almost identical. Compared with three-step search, NNS is much closer to the rate-distortion curve of full search, so it is clear that NNS achieves better rate-distortion performance than the three-step search.

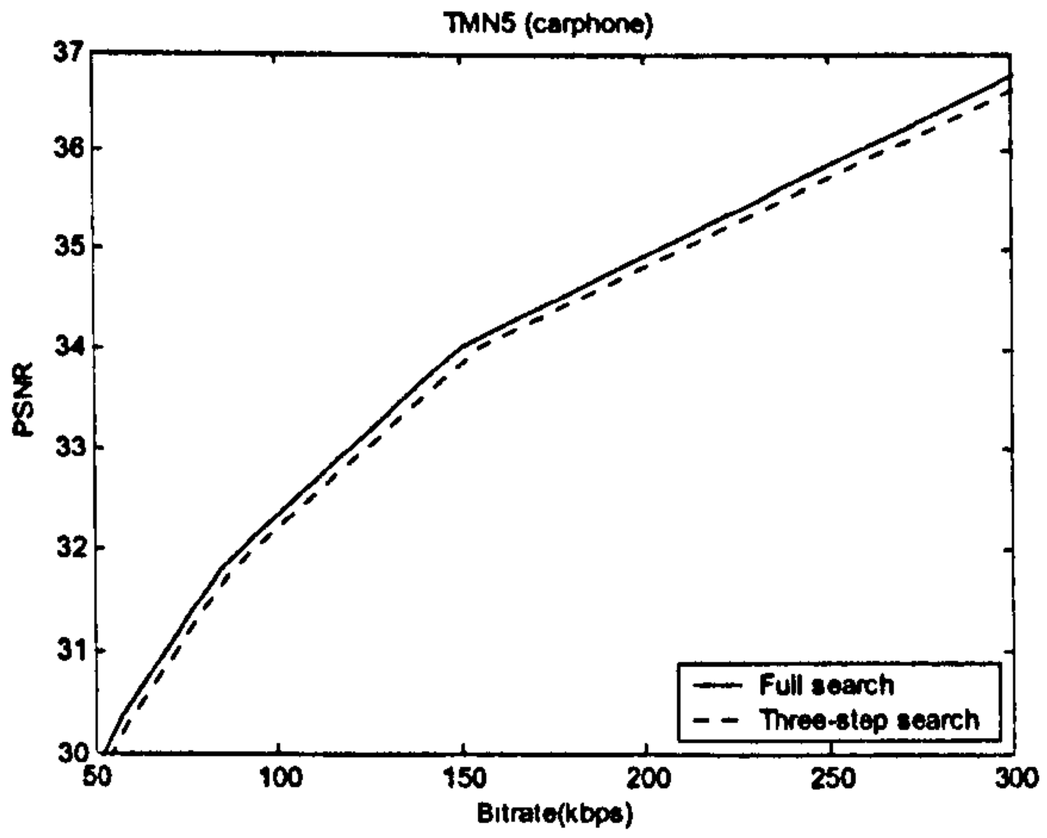


Figure 4-1 Rate-distortion performance of TMN5

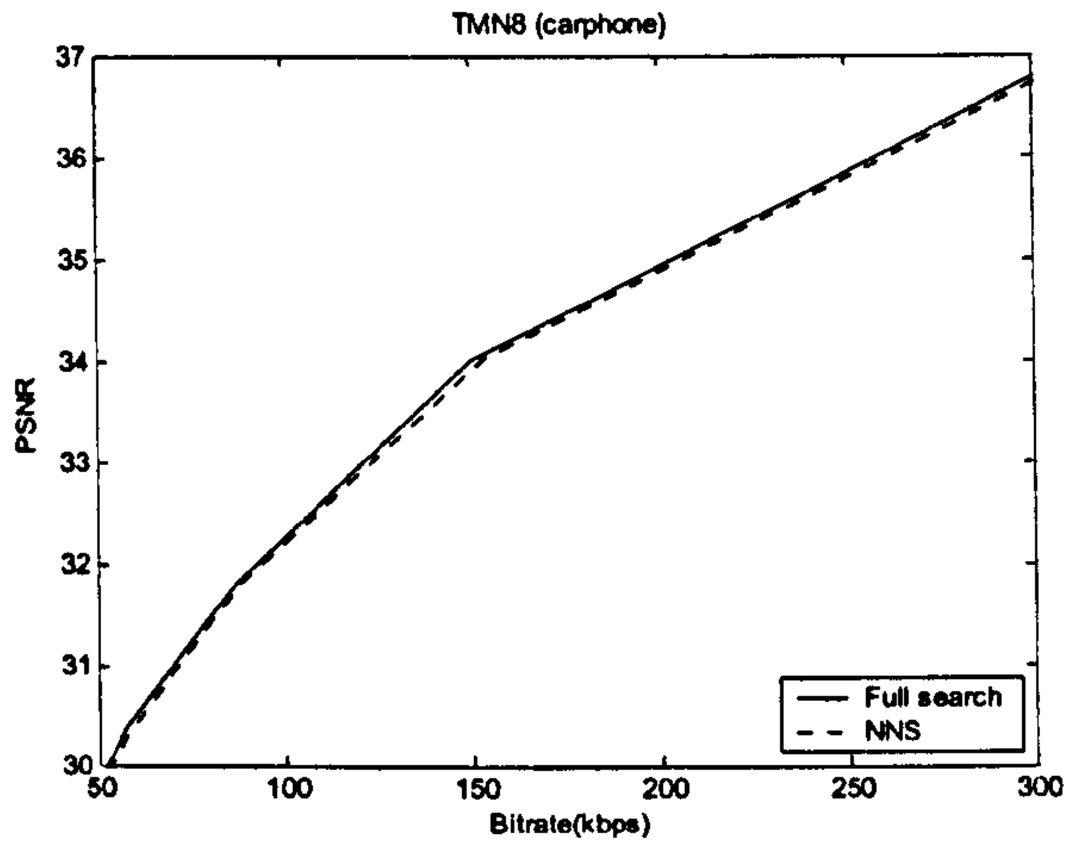


Figure 4-2 Rate-distortion performance of TMN8

4.4 Discussion of computational complexity and rate-distortion performance

Figure 4-1 shows that the full search algorithm has better coding performance than the three-step search, but has a high computational cost: 14.926 seconds (in Table 4-1) for encoding "Carphone" with 300 frames (a video sequence with a length of 12 seconds when the frame rate is 25 frames per second), and is therefore unsuitable for real-time encoding on this platform. The Three-step algorithm enables real-time coding by reducing the complexity of motion search with a small loss in rate-distortion performance: it takes only 8.796 seconds (less than 12 seconds, the requirement for real-time) to complete encoding of the same video clip. Similar results are illustrated in Figure 4-2 and Table 4-2 for Nearest Neighbour Search. These experimental results show there is a relationship between coding performance and computational complexity. High coding performance may require a complicated algorithm; fast coding algorithms can speed up the coding process at the expense of coding performance, which can be described as the trade-off problem between complexity and coding performance introduced in Section 2.5. A reduction in the computational cost of coding is often at the expense of a loss in coding performance.

Figure 4-1 and Figure 4-2 shows that the nearest neighbour search algorithm provides better coding performance than the three-step search and Table 4-2 shows that its

encoding time (8.807 seconds) is nearly identical to that of three-step search (8.796 seconds) listed in Table 4-1. Therefore, when trying to decrease the computational complexity of a video encoder, the loss of coding performance may be minimized by careful algorithm development and selection.

4.5 Summary

This chapter has demonstrated that the most computationally intensive functions in software-implemented video encoders are the DCT and related functions as well as motion estimation. Complexity reduction affects rate-distortion performance. However, the distortion can be minimised by developing effective algorithms. In the next chapter, approaches to reduce the complexity of the DCT and related function are introduced.

Chapter 5 Computational complexity control of DCT

5.1 Introduction

In Chapter 4, the Discrete Cosine Transform is demonstrated to be a complexity intensive function in software-implemented video encoders. This chapter starts by summarizing previous methods to reduce the DCT complexity in the literature and concentrates on the development of an algorithm for adaptive complexity management of the DCT function.

Discrete Cosine Transform (DCT) based video coding standards (such as H.263 and MPEG-4) continue to be widely used. Software implementations of these coding standards are used in a wide range of applications as they can provide low-cost, efficient and compatible encoding and decoding functionalities. In the applications of real-time multimedia communication systems and/or mobile-video systems (where power is a significant constraint), the performance of a video CODEC may be limited by the amount of processing power available as well as, or rather than, the available transmission bandwidth. When the processor can not cope with encoding video at the required frame rate, it starts to drop frames, resulting in variable performance. The resulting "jerky" video seriously damages perceived video quality.

The DCT (introduced in Section 2.3.2) is a widely-used transform technique in digital signal processing. Its main applications include image and video compression. The DCT operation requires a significant number of calculations. A major breakthrough in speeding up the DCT process was made in 1974 [62] with the invention of the first fast DCT algorithm, which is based on the Fast Fourier Transform (FFT). Since then, a lot of research work has been conducted to investigate efficient DCT algorithms, which can be mainly classified into two types: indirect approaches and direct approaches. The indirect approach speeds up the DCT process by mapping cosine sequences to other kinds of transformation sequences (for example: sine and cosine sequences in the Fourier Transform), and then making use of existing fast algorithms (for example, the FFT) [49]. [63-65] belong to the second category, which recursively decompose a matrix of original signals into matrix factors to decrease the required computations. These fast DCT

algorithms achieve computational complexity reduction by means of mathematical approaches, resulting in an output exactly the same as that of the conventional DCT.

The experimental results presented in Section 4.2 show that fast DCT algorithms are nevertheless complicated, requiring a large amount of coding resources. In a computation-constrained system, it may be necessary to resort to approximate calculation of the DCT so that computational complexity can be reduced further, at the expense of loss of information.

Subband-DCT approaches have been proposed to achieve faster computation of DCT [50, 66, 67]. In [50], the input sequence is decomposed into a series of sub-bands by a low-pass filter and a high-pass filter. The high-frequency bands are discarded because they contain less important information and DCT coefficients are only calculated from the low-frequency bands. If the decomposition is repeated n times, 2^n sub-bands are generated. A pre-determined complexity budget can be roughly achieved by adjusting the number of decompositions. Subband-DCT can produce an approximate DCT because some energy in the signal is thrown away prior to transform. The error caused by the approximation depends on the level and type of decomposition as well as the input sequence. If the input video sequence tends to have more energy in low-frequency bands, the output of subband DCT is similar to that of the conventional fast DCT.

Pruned DCT techniques [51, 68-70] are based on statistical characteristics of the DCT output: low-frequency DCT components have high values and high-frequency components tend to be near zero or zero. These techniques suggest computing a low-frequency subset of DCT coefficients (e.g. a 4x4 or 2x2) instead of the entire 8x8 DCT. High-frequency DCT coefficients are set to zero directly without being calculated. In a video frame, the blocks have various distributions of DCT coefficients and applying the same size of pruned DCT to all the blocks will result in loss of information for those blocks that have frequently high coefficient values, causing a loss of the decoded video quality.

Subband DCT and pruned DCT reduce computational complexity by approximately calculating the DCT coefficients. The accuracy of the DCT output depends very much on the input video signal: the more energy in low-frequency components, the less loss of

video quality. These techniques can decrease the amount of computation, but they can not adaptively control complexity and video quality.

New research work has led to the development of Variable Complexity Algorithms (VCAs) which aim to flexibly manage the computational complexity of a video CODEC independent of input video signal. It is necessary to trade off the complexity reduction and the video quality to meet the requirement of the specific applications. Generally speaking, VCAs aim to achieve a scalable reduction in computational complexity with acceptable loss of video quality.

A simple VCA has been proposed to reduce the complexity of IDCT function [71] by detecting the zero input coefficients and ignoring them in computation. Similarly in [72], a VCA for the DCT function saves computation resources by detecting a subset or the entire input signal in a block which will become zero after quantisation. An image analyser [73] based on hierarchical vector quantisation [74] has been suggested to vary the DCT complexity of the JPEG image by indicating sub-blocks containing all zero coefficients. This method is also applicable to MPEG video sequences. The VCAs for DCT and IDCT require additional calculations to classify the inputs, which may offset the computational savings.

The distribution of quantised DCT coefficients can be represented by the position of the last nonzero coefficient: End-of-Block (EOB). It is possible to predict the EOB of each block by using a model and the DCT operations with reduced-computations can be applied to those blocks that have low EOB values. In [75], the authors have exploited the relationship between EOB and the quantiser step size (Q) used for each block in several video sequences and have found that the EOB decreases with the increment of Q. Consequently, it is proposed to use the quantiser step size to predict the EOB. When Q is larger than a pre-determined threshold, a 4×4 subset of DCT coefficients is calculated. Otherwise, the normal 8×8 DCT operation is carried out. This model only considers the influence of the quantiser on the output DCT coefficients, but does not take into account the variation of the input signal. It "sacrifices" blocks which have high detail and/or motion because a lot of important information is thrown away. This model is improved in [76] by including the residual energy of each macroblock after motion compensation (represented by Sum Absolute Difference (SAD)). The quantiser and SAD are compared

with a threshold to decide whether to calculate a full-DCT, to compute only the DC component of the DCT coefficients or to skip the entire DCT process. The threshold is identical for any input video sequence. Applying the same threshold to video sequences with various levels of detail and movement results in different levels of complexity reduction and video quality degradation. This method can reduce DCT computations, but is not able to provide adaptive control of the computational complexity. Appendix J in MPEG-4 Video Verification Model (VM) 18 [77] proposes comparing the SAD of each block (rather than macroblock in [76]) with a threshold to predict all-zero quantised DCT-coefficients. It only lists the upper bound of the threshold that guarantees no loss of video quality: i.e. non-zero coefficients blocks are assured with full DCT operation and there is no further introduction of the selection of threshold.

In this chapter, the following steps are taken to develop an "intelligent" method of managing computational complexity. The models used to predict the EOB are compared and the correlation between complexity and threshold is investigated. An adaptive algorithm is proposed to flexibly control the complexity of the DCT and related functions, including quantisation, IDCT and inverse quantisation. Experimental and profiling results demonstrate this algorithm can achieve and maintain a "target" level of computational complexity. This work has been published in [78], which is presented in full in an Appendix.

5.2 Predicting End of Block and decision thresholds

5.2.1 Predicting End of Block

During block-based video coding, an encoder performs motion estimation to find the most similar area in reference frame(s) for each macroblock in the current video frame. The selection is made based on error measurements, the most popular one of which is Sum of Absolute Difference (SAD) (shown in Equation 2-2), due to its low computation cost. The macroblock with minimum luminance SAD in the reference frame(s) is chosen as the best match for the current MB and the difference between each sample of these MBs is DCT transformed and encoded.

The luminance SAD value of a MB (SAD_{MB}) has been suggested to predict the position of the last non-zero quantised coefficient (End of block: EOB) of every block in a MB

[76]. The EOB depends on the energy of the DCT coefficients in each block, which is decided by the residual value of each block after motion-compensated prediction. The SAD_{MB} represents the average value of the luminance residual signal in a MB and it therefore can approximate the position of EOB. However this is based on an assumption that every block in an MB, including four luminance blocks and two chrominance blocks, has the same value of EOB. If a macroblock is positioned at the edge of a high detail and/or motion area, some blocks will have high signal energy and others have low energy, so that the EOB of these blocks in the same MB will not be identical. The SAD_{MB} of each macroblock is calculated during motion estimation, so this method adds no extra computation to the encoding process.

Appendix J in MPEG-4 video VM 18 suggests using the SAD of each 8x8 block (SAD_B) to model the EOB, which is described as follows:

$$SAD_B = \sum_{i=0}^{7}\sum_{j=0}^{7}|C(i,j)|$$

Equation 5-1

Where C(i,j) are the residual samples of luminance block or chrominance block.

SAD_B gives a more precise prediction of the EOB for each block than SAD_{MB} because it represents the energy of a single block more accurately. SAD_{MB} is computed for each macroblock during motion estimation. SAD_B can be obtained without additional computations by ensuring the encoder calculates SAD_{MB} from the sum of SAD_B of the luminance blocks. Additional computations for calculating SAD_B of the chrominance blocks are required as they are not used in motion estimation.

In order to compare the performance of predicting EOB of SAD_{MB} and SAD_B, the "Carphone" video sequence was coded by H.263 encoder with Q = 8 and no optional modes. SAD_{MB} for each macroblock, SAD_B for each block in a MB and the corresponding EOB value are recorded. The EOB represents the position of the last non-zero quantised coefficients. EOB equal to zero means all of the coefficients are zero, EOB equal to one means that only the DC coefficient is non-zero and so on. Since SAD_{MB} is the total of absolute value of 256 residual samples in a 16×16 MB and SAD_B is

that of 64 residual samples of each 8×8 block, SAD_{MB} is normalized by dividing each value by 4 so that SAD_{MB} and SAD_B can be compared on the same basis.

The performance of SAD_{MB} and SAD_B are compared in Figure 5-1. The line "P(EOB=0)" plots the probability of EOB=0 against SAD (normalised SAD_{MB} or SAD_B). It is clear for both SAD_{MB} and SAD_B that low SAD corresponds to a high probability of EOB being equal to zero. Furthermore, SAD_B gives much higher P(EOB=0) than SAD_{MB} at the same value through the entire range of SAD. The cumulative distribution of SAD_{MB} and SAD_B are shown in the lines labelled with "CDF", which have a similar trend and are very close to each other. This indicates that the distribution of SAD values for SAD_B and normalized SAD_{MB} are approximately the same.

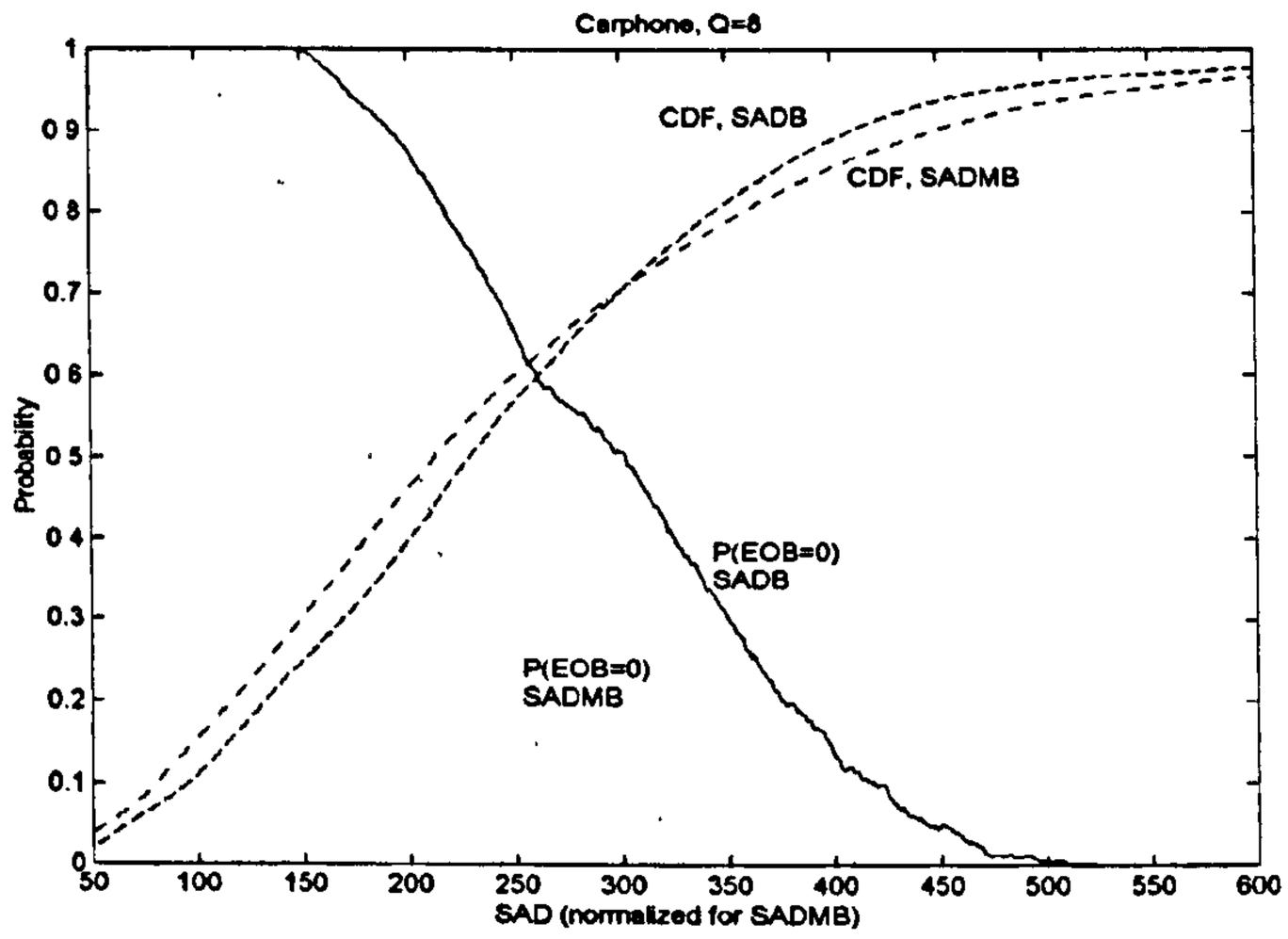


Figure 5-1 Probability of EOB=0 and cumulative distribution (CDF) (Carphone, Q=8)

Table 5-1 lists P(EOB=0), CDF (proportion of blocks) and SAD value of some examples. For instance, when SADB is equal to 200, there is 90% probability that EOB will be zero (i.e. all coefficients in this block will become zero after quantisation); however, there is only a 55% probability that EOB will be zero for normalized SADMB equal to 200. Correspondingly, there are 40% blocks which have SAD less than or equal to 200.

Table 5-1 Probability of EOB=0 and block distributions (Carphone, Q=8)

SAD threshold	P(EOB=0)	Proportion of blocks
$SAD_B = 200$	90%	40%
$SAD_{MB} = 200$	55%	40%
$SAD_B = 250$	70%	60%
$SAD_{MB} = 250$	20%	60%

The computational complexity of DCT function may be reduced using either SAD_{MB} or SAD_B to predict all-zero DCT coefficients. Given a pre-determined threshold (T_0), if the SAD value of a MB or block is less, DCT and quantisation are not performed. For example, when the threshold is set to 200 and SAD_{MB} is used as a predictor, 40% blocks will not go through the DCT, i.e. 40% of the computational complexity of DCT and related functions (including quantisation, IDCT and inverse quantisation) are saved. Within those 40% blocks, 55% are correctly predicted and the coefficients of the remaining 45% are wrongly set to zero. Those 45% mis-predicted blocks will cause degradation of video quality. However, when SAD_B is employed, for the same threshold (200), there is the same percentage of computation reduction (40%), but the mis-prediction rate drops to 10%. This indicates that SAD_B can achieve better video quality than SAD_{MB} with the same reduction of DCT complexity. This is because the number of non-zero quantised coefficients in an individual block depends on the energy in that block and SAD_B gives a more accurate measure of block energy than SAD_{MB} (which is proportional to the average energy across all 4 luminance blocks in the current macroblock).

5.2.2 Decision threshold

The P(EOB=0) curve in Figure 5-1 indicates that P(EOB=0) depends on the SAD value. The P(EOB=0) also depends on the quantisation step size because a higher quantiser results in a high probability that all coefficients will be zero after quantisation. Figure 5-2 plots the probability of EOB=0 against SAD_B for the Carphone sequence with varying quantiser step size.

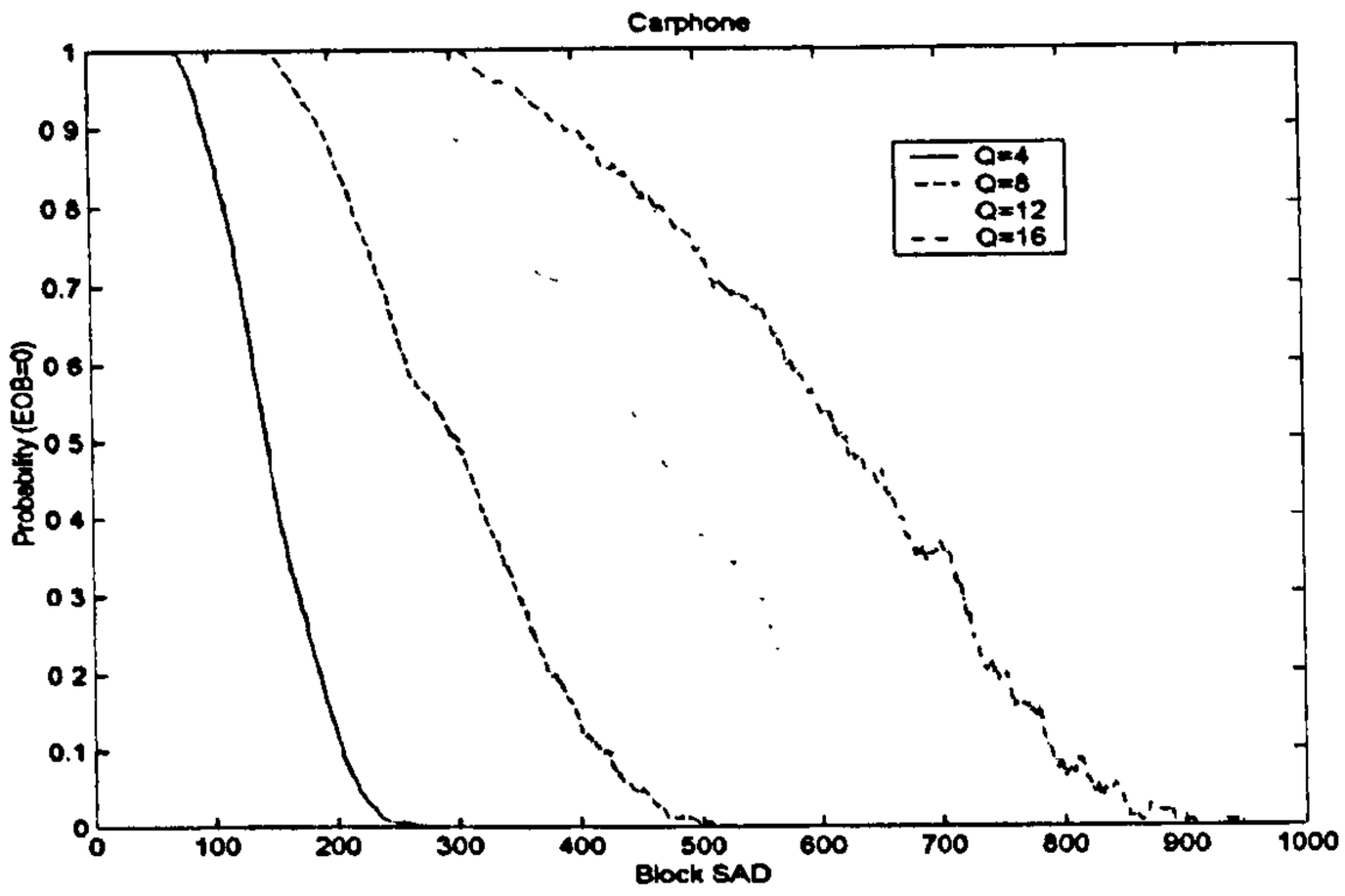


Figure 5-2 Probability of EOB=0 against block SAD for Carphone with varying Q

The value of SAD_B at which a given P(EOB=0) occurs is approximately related to quantiser step size Q, i.e.

$$P(EOB = 0) \approx \frac{SAD_B}{Q}$$

Equation 5-2

In order to determine whether the DCT should be calculated for a block, SAD_B/Q may be compared with a threshold T_0. A complexity-reduction algorithm to reduce the complexity of DCT can be described in pseudocode as follows:

IF $SAD_B/Q < T_0$

THEN Do not perform DCT and quantisation

Set quantised coefficients to zero

ELSE Perform DCT and quantisation

A lower value of the threshold T_0 will tend to give a more accurate prediction of zero EOB, resulting in almost no loss of video quality. For example, if the threshold T_0 is set

at about 25, EOB will be correctly predicted for approximately 90% of blocks when quantiser step size is 8. A higher value of T_0 should provide high computational complexity reduction (i.e. fewer DCTs are calculated) at the expense of poorer prediction accuracy.

5.3 Variation in computational complexity

5.3.1 Variation of complexity with fixed threshold

By using the complexity-reduction algorithm described in Section 5.2.2, video sequences "Carphone" "Mother and daughter" and "Claire" were coded with fixed threshold T_0=30 and a fixed quantisation step size = 8. At each frame, the number of blocks transformed by DCT (N_i) was recorded. The DCT complexity of a frame i (C_i) is defined as N_i over B_i (the total number of blocks in a frame), as shown in Equation 5-3. The DCT complexity of the original video encoder (without the complexity-reduction algorithm) is equal to one as all of the blocks go through the DCT process. It is possible that the DCT complexity of each frame may be less than one after employing the complexity-reduction algorithm in the encoder.

$$C_i = \frac{N_i}{B_i}$$

Equation 5-3

Figure 5-3 compares the DCT complexity of the first 200 frames of the "Carphone", "Mother and daughter" and "Claire" sequences for fixed T_0=30. There is a clear variation of DCT complexity between sequences: sequences with little detail and low activity (such as Claire) tend to have lower complexity than high movement and detail sequences ("Carphone", "Mother and daughter", etc.). "Claire" has lower value of residual signal causing a high probability of EOB=0. Given the same threshold T_0, the DCT will be skipped for more blocks in "Claire" than for a high-activity sequence. There is also variation between frames in the same video sequence (e.g. frame 170-200 in "Carphone"), which is caused by changes in scene activity.

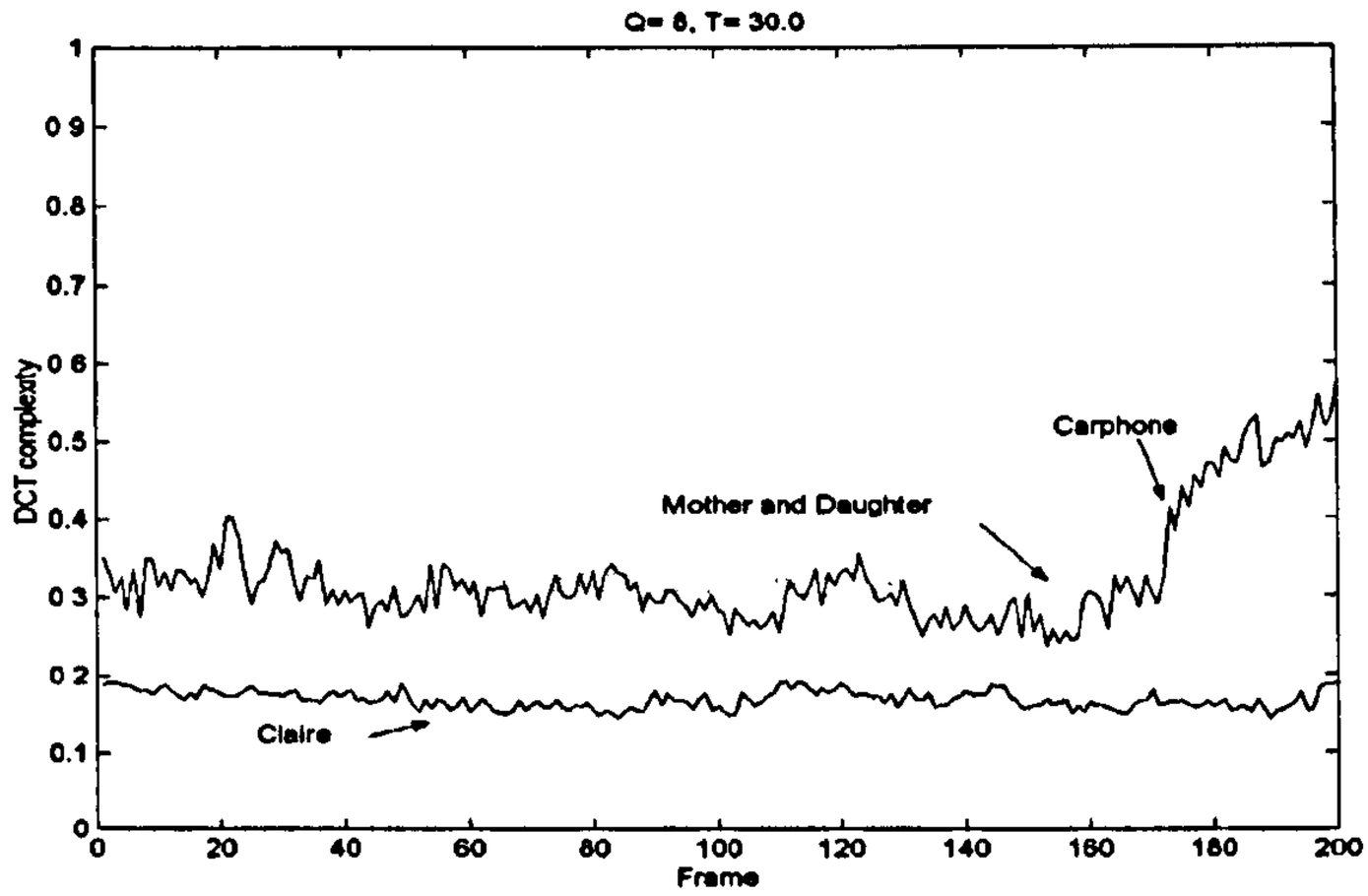


Figure 5-3 Variation of DCT complexity

With a fixed threshold, the computational complexity of the DCT depends on the scene content of the input video, so it is difficult to predict how much complexity reduction can be achieved. It is also impossible to predict the error of skipping the DCT process (mis-prediction) and its effect on video quality. Therefore, it is difficult to manage the processor utilization required to carry out the DCT for an unknown input video sequence. The fixed threshold causes variation of the DCT complexity of each frame, resulting in a varying requirement of processing power. If available processor capacity is limited, it may not be possible to process the video sequence when the DCT complexity increases significantly. An intelligent complexity-control approach should be able to maintain a predictable level of computational complexity that does not depend on the content of the video sequence.

5.3.2 Correlation between complexity, threshold and quantiser

The video sequences "Claire", "Mother and daughter" and "Carphone" were coded using a H.263 encoder integrated with the complexity-reduction algorithm described in Section 5.2.2 with a range of fixed threshold (T_0) and quantiser step size (Q). The corresponding computational complexity of the DCT function (C) was recorded for each pair of T_0 and Q and Figure 5-4 plots the relationship of complexity (C), threshold (T_0) and quantiser (Q) for each test sequence. There is a clear relation between the complexity, threshold and quantiser: at a fixed threshold, complexity decreases when Q increases; a large value of

T_0 tends to reduce the DCT complexity with fixed Q. Figure 5-4 also shows that varying the threshold results in a greater change of complexity than varying the quantiser. The shape of the T/Q/C surfaces for various video sequences is very similar and only the actual values of complexity are different, which indicates that there is a consistent relationship between C, T_0 and Q regardless of the content of the input video sequences.

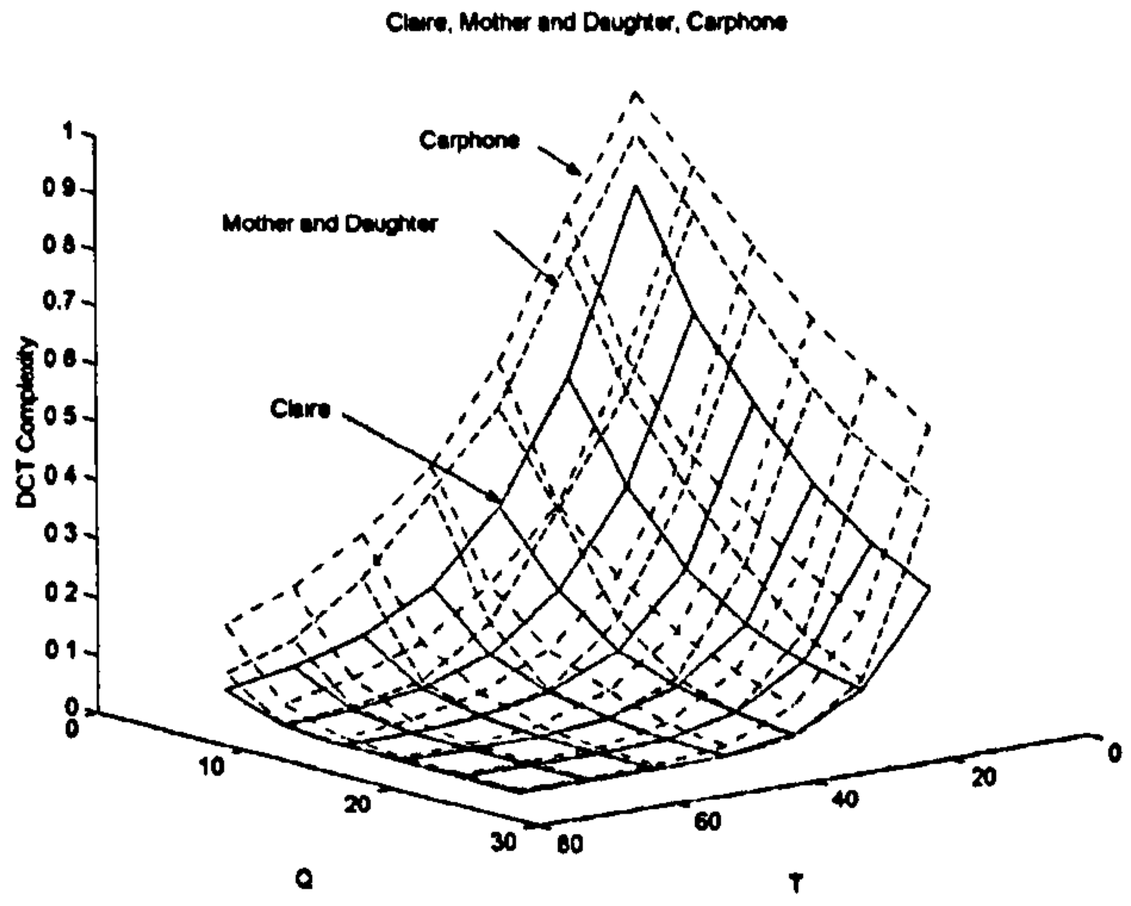


Figure 5-4 Surface of DCT complexity, T0 and quantiser

In order to examine the effect of changing the threshold and quantiser on the coded bit rate, Figure 5-5 plots the mean bit rate, T_0 and Q of the above video sequences. Higher values of Q reduce the coded bit rate. This is because high Q produces more zero coefficients after quantisation and therefore less information is left for entropy coding. T_0 also influences the coded bit rate (for a fixed value of Q): higher values of T_0 result in lower bit rates, as fewer blocks are actually coded. However, the effect is small compared with the effect of quantiser step size Q on bit rate.

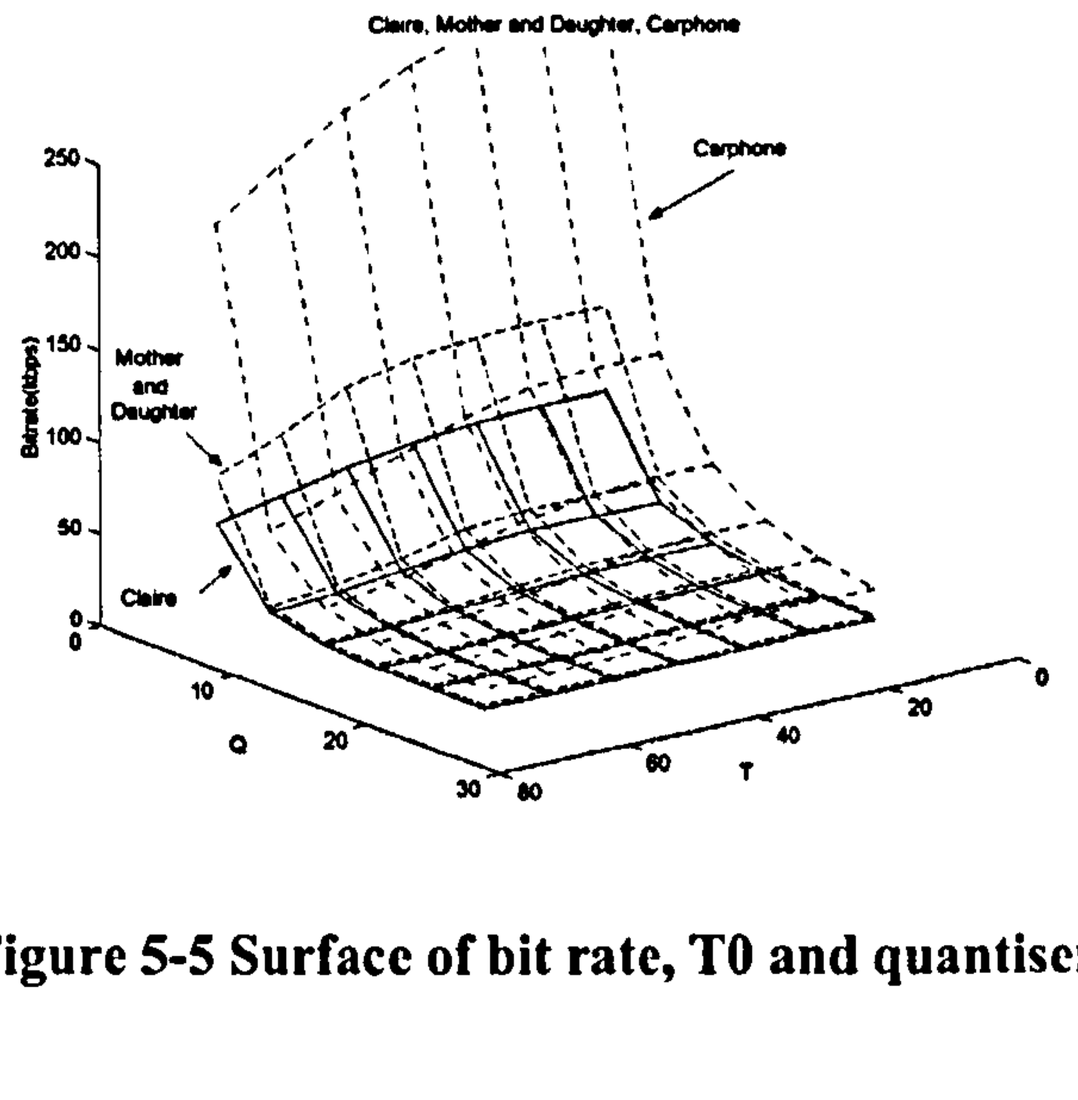


Figure 5-5 Surface of bit rate, T0 and quantiser.

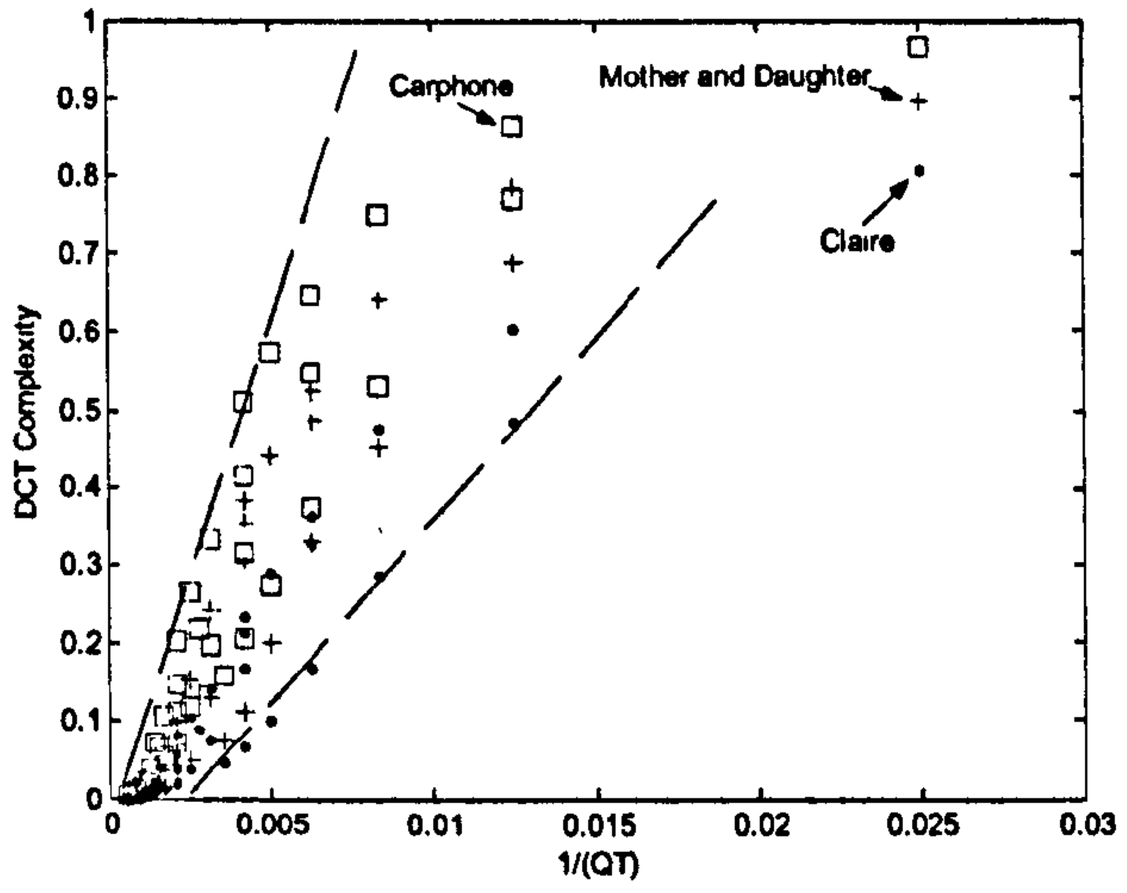


Figure 5-6 DCT complexity against $1/QT_0$

Figure 5-4 demonstrates that computational complexity of DCT is closely related to the threshold and quantiser and large values of T_0 and Q tend to produce low complexity. Figure 5-6 plots the DCT complexity (C) against $(1/QT_0)$ for the three test video sequences described above. Most of the markers fall into the range specified by the two dashed lines, which indicates an approximately linear relationship between complexity and $(1/QT_0)$. Those markers who are outside this range tends to have higher value of $(1/QT_0)$, corresponding with lower value of Q and T. It is not necessary to take into account of these points as lower value of Q and T will result in almost no skipping of

DCT function on blocks. Consequently, C can be represented as a linear function of $(1/QT_0)$ for each video sequence, i.e.

$$C \approx \frac{v}{T_0 Q}$$

Equation 5-4

Parameter v varies with the content of the video sequence.

5.4 Adaptive control of DCT complexity

5.4.1 Introduction

With a fixed threshold, there is significant variation in the computational complexity of DCT processing between sections of the same video sequence due to the changing scene content. Equation 5-4 demonstrates the approximately linear relationship between complexity (C) and $1/T_0$, indicating that it may be possible to adaptively change T_0 to reduce variation in complexity. Scene content in a video sequence tends to vary gradually (i.e. neighbouring frames are often similar in terms of detail and activity), thus it is proposed to update T_0 based on the measured computational complexity C of one or more recently-encoded frames. It is proposed to update T_0 after encoding each frame in order to maintain a near-constant "target" computational complexity (C_t) throughout the entire video sequence. Modifying T_0 once per frame should be sufficient for the purpose of managing processor resources: a practical video encoding application will usually buffer one or more frames of coded data prior to transmission and so a short-term variation in computational complexity during the encoding of a frame is not likely to be significant. For practical computational management, it is more important to smooth out the computational complexity over a longer period.

The development process of an adaptive complexity-reduction algorithm is investigated in this section, the goals of which are to:

1. Maintain a mean target complexity C_t

2. Minimize the variation in frame complexity C_n

3. Minimize distortion

5.4.2 Updating threshold

After encoding frame n, the actual value of DCT complexity for frame n (C_n) can be measured based on the proportion of blocks processed by the DCT. Based on the relationship between complexity, threshold and quantiser described in Equation 5-4, v can be represented as:

$$v = C_n Q_n T_{0n}$$

Equation 5-5

where T_{0n} and Q_n are the threshold and mean quantiser for frame n and C_n is the measured DCT complexity of frame n (i.e. the proportion of blocks for which the DCT and quantisation functions were calculated as defined in Equation 5-3).

For the next frame n+1, the threshold $T_{0(n+1)}$ is required to achieve the target complexity C_t. The relationship between $T_{0(n+1)}$, C_t and $Q_{(n+1)}$ (the quantiser step size at the frame level chosen for frame n+1) corresponds with Equation 5-4. Consequently, $T_{0(n+1)}$ can be calculated from Equation 5-4:

$$T_{0(n+1)} = \frac{v}{C_t Q_{(n+1)}}$$

Equation 5-6

Assuming that the proportionality term v does not vary significantly between frame n and frame n+1, combining Equation 5-5 and Equation 5-6:

$$T_{0(n+1)} = T_{0n} \frac{Q_n}{Q_{(n+1)}} \frac{C_n}{C_t}$$

Equation 5-7

Rearranging gives:

$$T_{0(n+1)} = T_{0n} \frac{Q_n}{Q_{(n+1)}} \left(1 + \frac{C_n - C_t}{C_t} \right)$$

Equation 5-8

Equation 5-8 contains a term proportional to the previous value of T_0 and a term proportional to the error between the actual and target computational complexity (C_n-C_t). The sensitivity of Equation 5-8 to this error term may be varied by scaling the error term with a factor *k*:

$$T_{0(n+1)} = T_{0n} \frac{Q_n}{Q_{(n+1)}} \left(1 + \frac{C_n - C_t}{kC_t} \right)$$

Equation 5-9

A small value of *k* means that the update is sensitive to small errors (C_n-C_t): it should therefore converge rapidly to a target complexity C_t at the expense of possible instability. A large value of *k* should give better stability but slower convergence.

If quantiser (Q) is kept constant throughout each sequence, which means that the term $Q_n/Q_{(n+1)}$ becomes unity, Equation 5-9 can be simplified to:

$$T_{0(n+1)} = T_{0n} \left(1 + \frac{C_n - C_t}{kC_t} \right)$$

Equation 5-10

5.4.3 Adaptive complexity-reduction algorithm

Based on the method of updating the threshold (T_0) in Equation 5-10, the adaptive DCT complexity-reduction algorithm can be described as:

1. Initialisation (set C_t, k and a initial value of T_0);

2. IF $SAD_B/Q < T_0$

THEN *Do not perform DCT and quantisation*

Set quantised coefficients to zero

ELSE *Perform DCT and quantisation*

3. Update T_0 *after encoding each frame*

4. Go to step 2

5.5 Experimental results

5.5.1 Video quality, convergence time and complexity stability

The adaptive complexity-reduction algorithm for the DCT function proposed above was tested in terms of video quality, convergence time and the stability of the complexity. The quality of each video sequence coded by the adaptive algorithm is evaluated using PSNR: mean PSNR drop (compared with a "baseline" encoder) gives the average quality of entire video sequence and the PSNR value of each frame shows the variation of the quality throughout the sequence. Convergence time and complexity stability indicate the complexity-control ability of the adaptive algorithm. This algorithm can use any threshold for the first frame and updates it for the next frame after encoding the current frame. It usually takes a few frames to reach the target complexity level, and the convergence time is defined as the number of frames that are coded before the algorithm reaches target complexity (within +/-10% of C_t). Changing term k in Equation 5-10 will affect the convergence time of the algorithm. The stability of the adaptive algorithm may be estimated by measuring the Standard Deviation (STD) of the DCT complexity, which is described in the following equation. A lower STD value indicates high stability of the algorithm.

$$STD = \sqrt{\frac{\sum_{i=1}^{n}(C_i - C_{ave})^2}{n-1}} \text{, where } C_{ave} = \frac{1}{n}\sum_{i=1}^{n} C_i$$

Equation 5-11

The sequences "Carphone", "Claire" and "Mother and Daughter" were encoded by H.263 TMN5 encoder integrated with the adaptive DCT complexity-reduction algorithm described by Equation 5-10, with a range of quantiser step size Q, convergence factors k and target DCT complexities C_t. Q was kept constant throughout each sequence and the threshold T_0 was initially set to an arbitrary value of 30 in each case.

"Mother and daughter" video sequence

Figure 5-7, Figure 5-8 and Figure 5-9 plot the variation of DCT complexity (C) against frame number for the Mother and Daughter video sequence with Q=8 and convergence factor k=2, 6, 10 respectively. The target complexity (i.e. the proportion of blocks for which the DCT is calculated) is set to 0.1, 0.3, 0.5. Figure 5-7 shows the obtained complexity of every frame when k is equal to 2. It is clear that the complexity quickly converges to the target value. The adaptive algorithm achieves constant and stable complexity for C_t = 0.3 and 0.5. However, the obtained complexity seriously oscillates if the target is set to 0.1. The result of k=10 is shown in Figure 5-9: the algorithm is stable for all the target values but converges slowly to each target, especially for C_t=0.5. A good compromise between stability of the algorithm and convergence time is given by k=6 shown in Figure 5-8: the algorithm remains stable for $C_t \geq 0.1$ and maintains the target complexity despite changes in sequence characteristics.

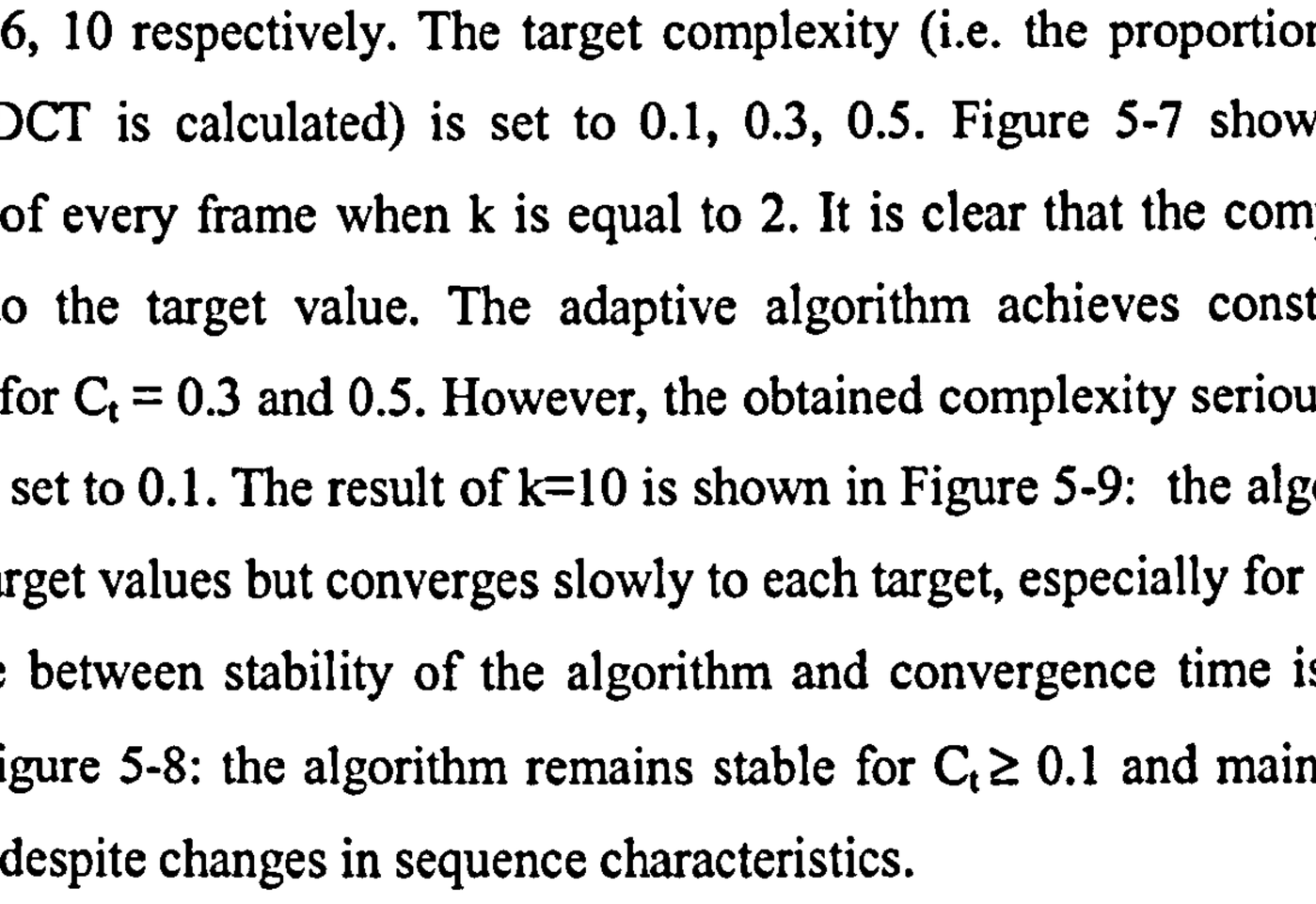


Figure 5-7 DCT complexity for mother and daughter with k=2

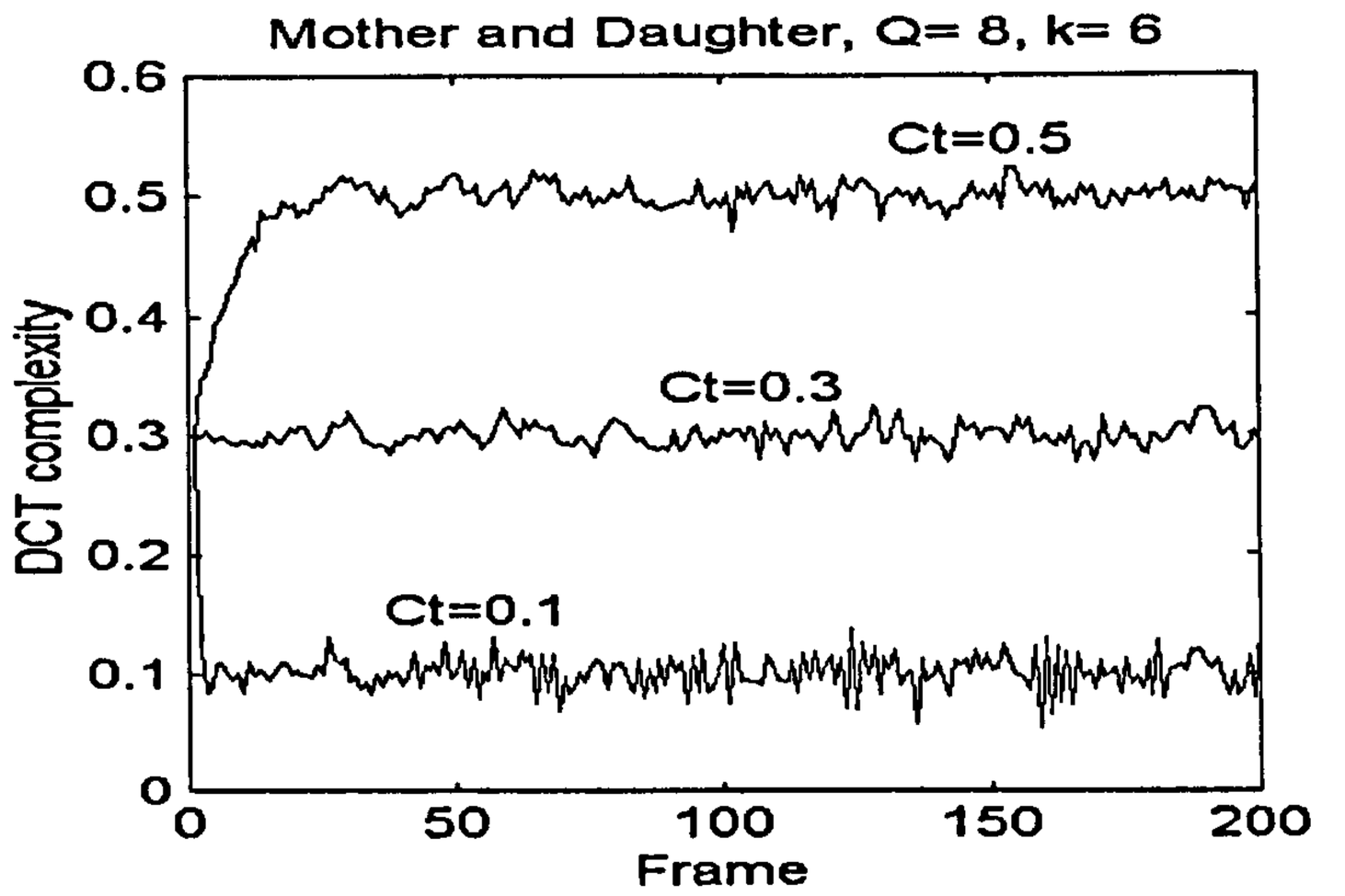


Figure 5-8 DCT complexity for mother and daughter with k=6

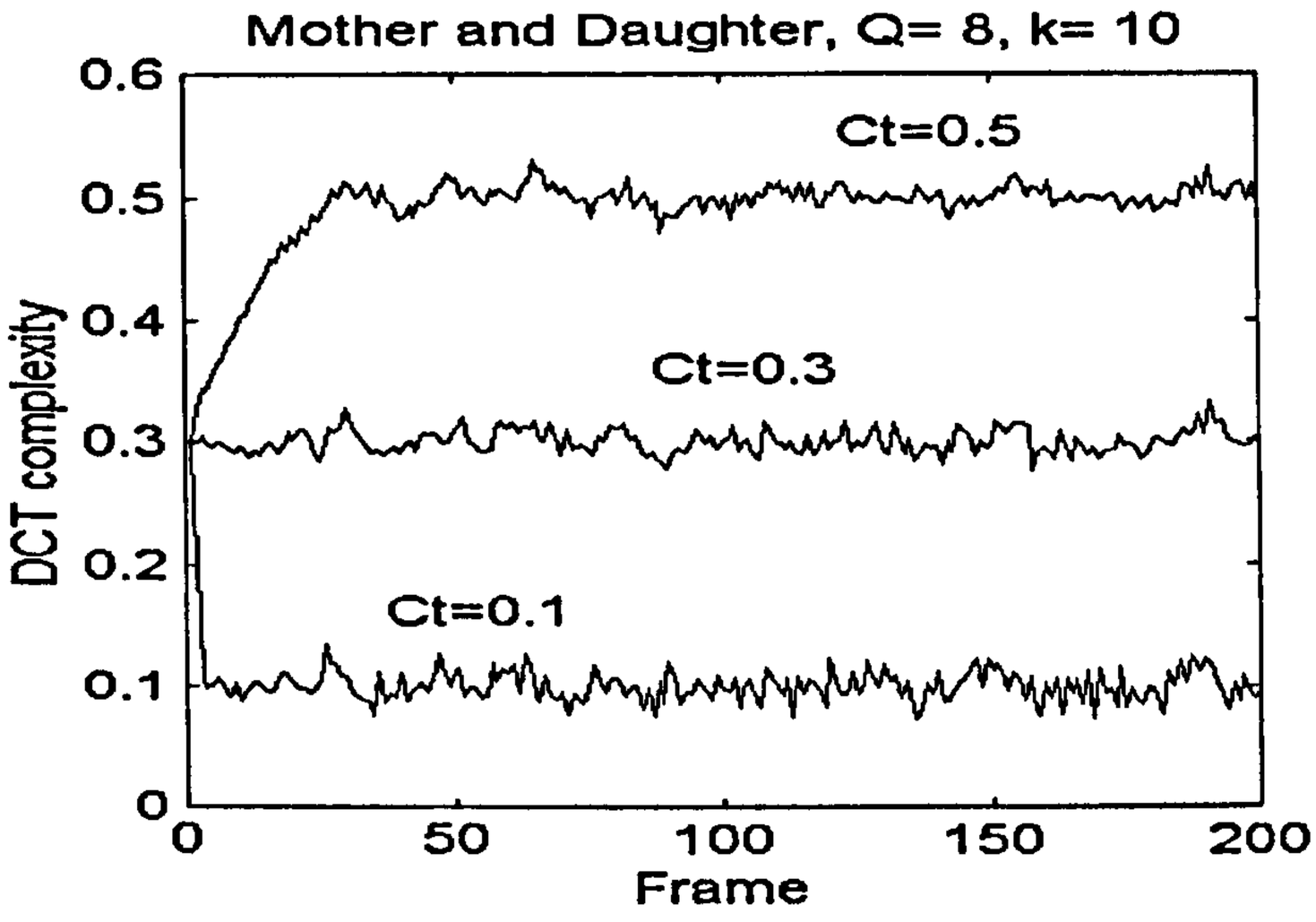


Figure 5-9 DCT complexity for mother and daughter with k=10

The corresponding PSNR of each frame in "mother and daughter" video sequence with Q= 8 and k =2, 6, 10 are shown in Figure 5-10, Figure 5-11and Figure 5-12 respectively. For k=6 in Figure 5-11, there is a negligible drop in PSNR when C_t=0.5 (not shown in the figure), and a slightly larger drop when C_t=0.3. PSNR drops more significantly when C_t=0.1, but the decoded sequence is still clearly recognizable. Figure 5-10 shows for k=2

a large drop and oscillation of PSNR occurs when target complexity is equal to 0.1 due to the unstable threshold as shown in Figure 5-7.

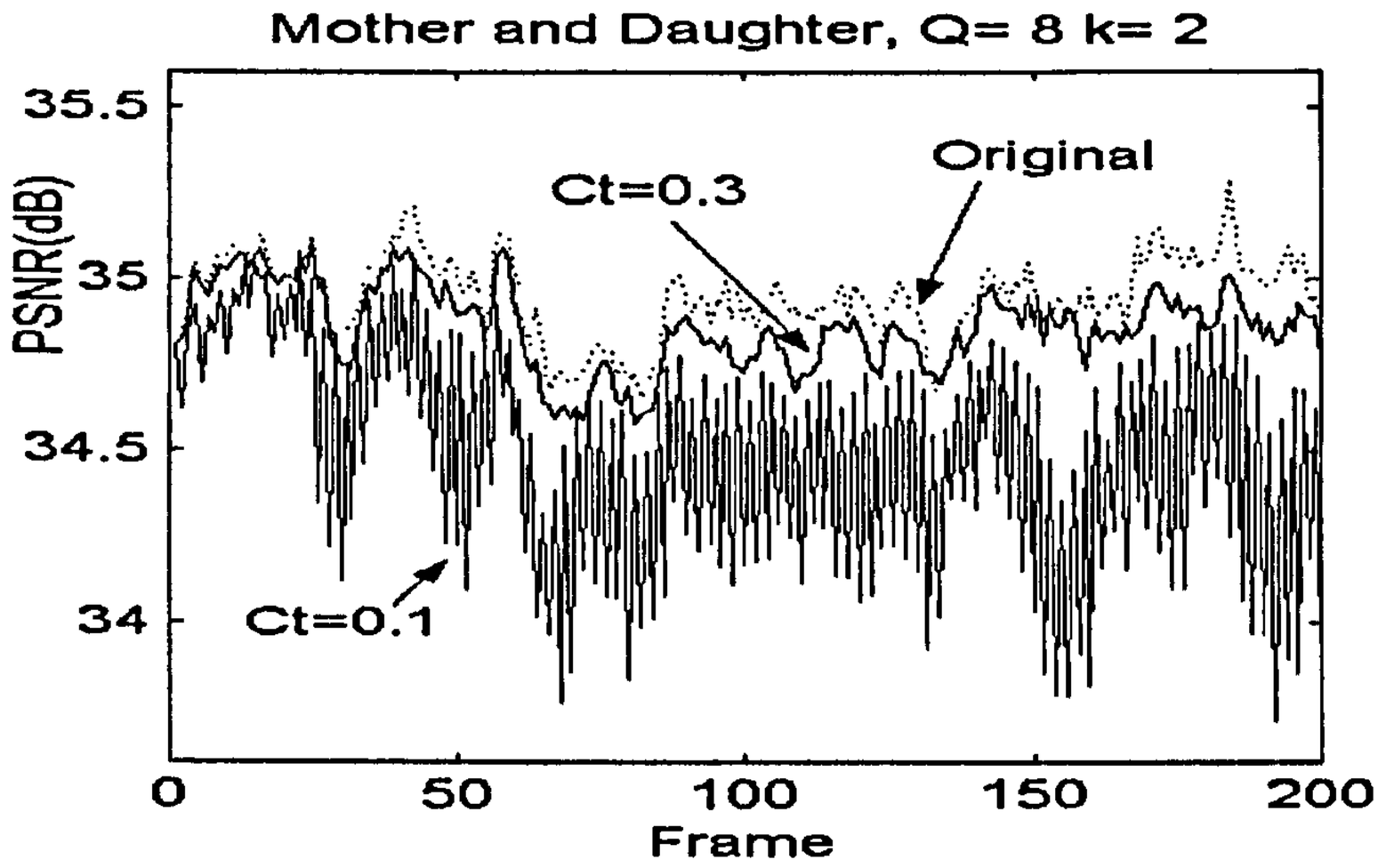


Figure 5-10 PSNR against frame number of mother and daughter with k=2

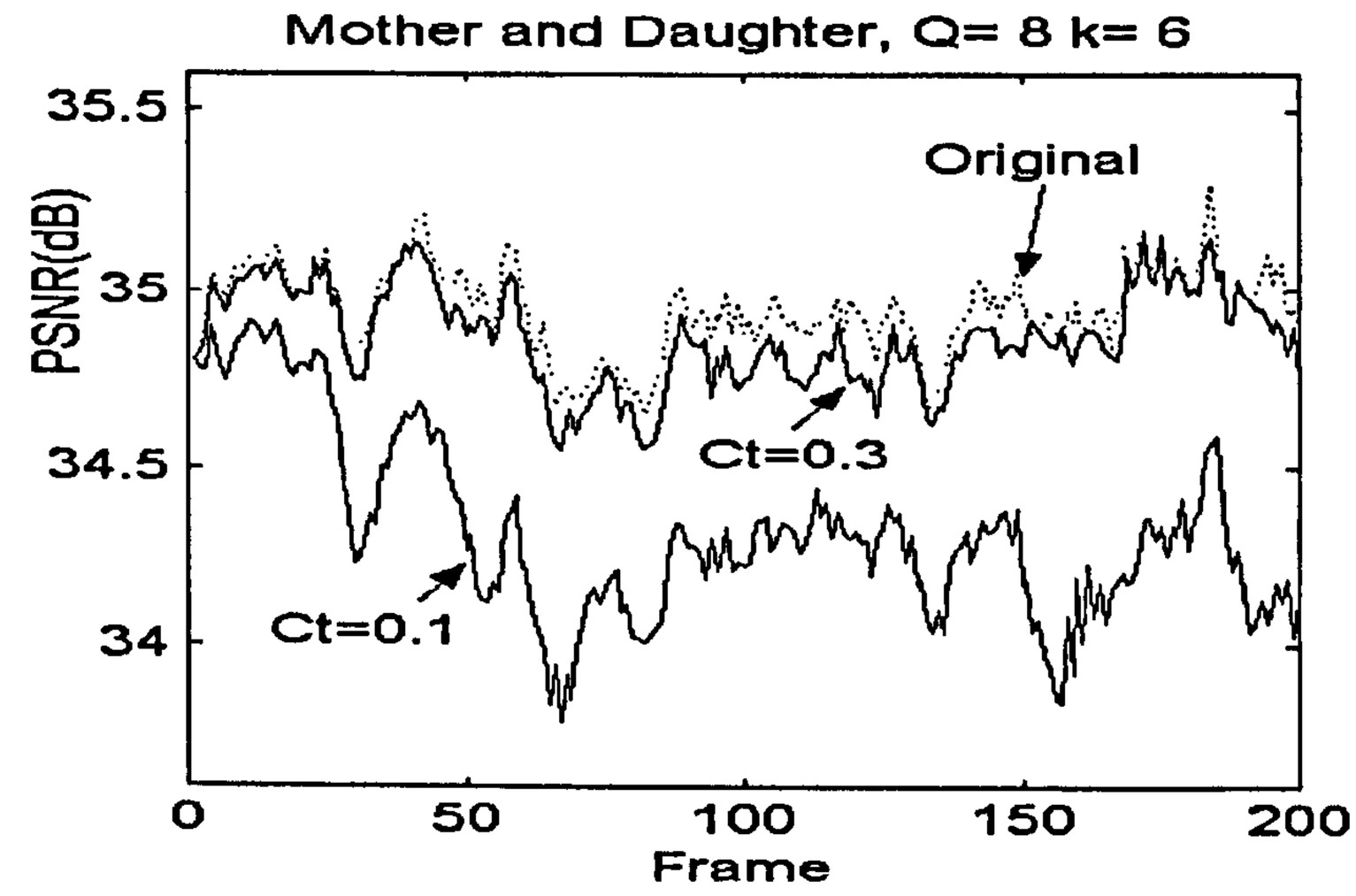


Figure 5-11 PSNR against frame number of mother and daughter with k=6

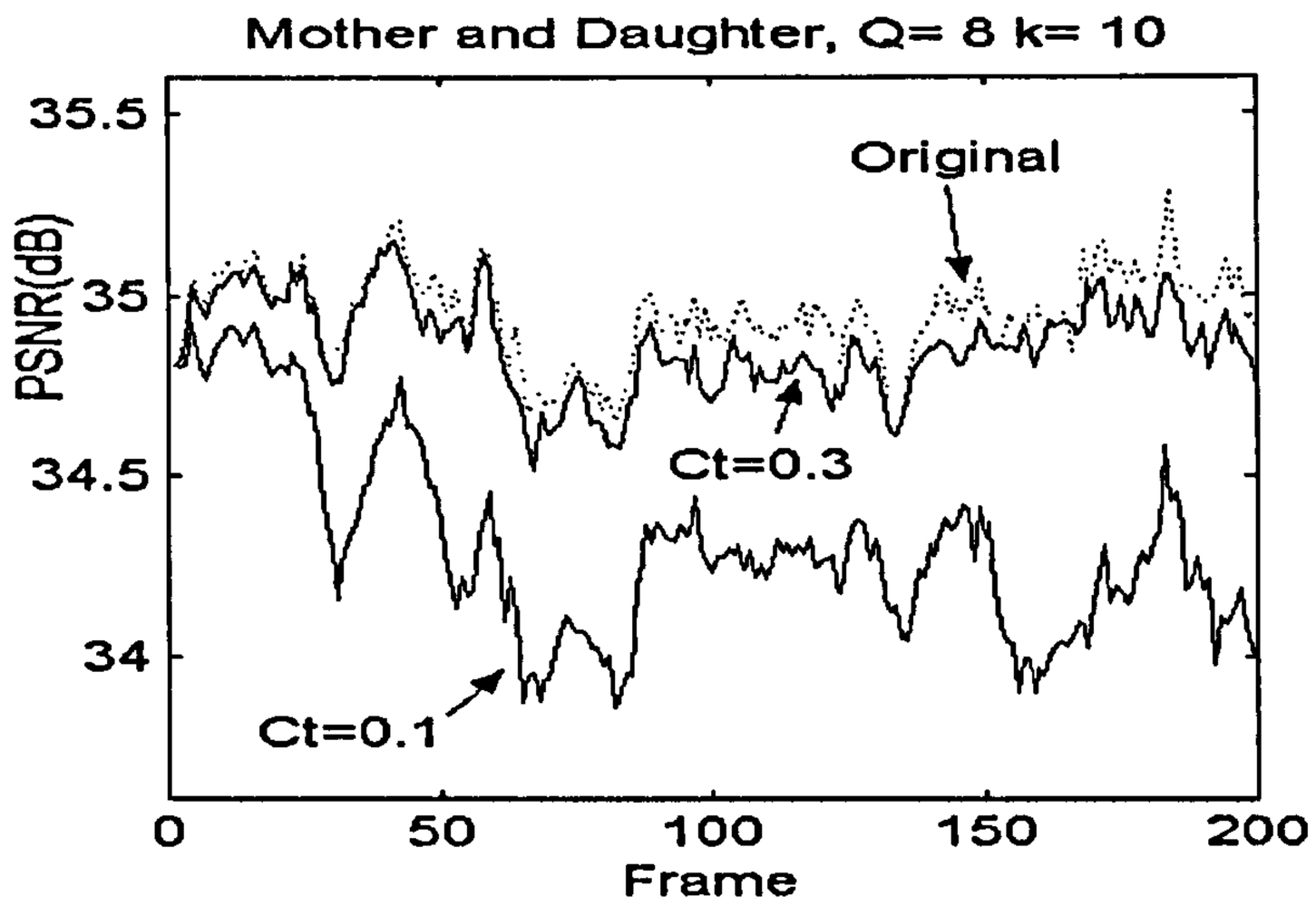


Figure 5-12 PSNR against frame number of mother and daughter with k=10

Similar results were obtained for the other sequences (Carphone and Claire). Convergence factor k=6 gives acceptable performance for all sequences and all values of Q.

Convergence factor k=6

Table 5-2 summarizes the experimental results for each sequence coded by the adaptive algorithm with a convergence factor k=6. The mean DCT complexities listed in this Table are nearly the same as the target set for each case and the standard deviation of the measured DCT complexity ("STD Complexity") is very small. These results show that the adaptive algorithm can achieve the target DCT complexity and remain stable throughout the video sequence. Table 5-2 also shows that the PSNR drop is very small for each sequence and quantisation step size when $C_t = 0.3$ and 0.5 (the highest drop is only 0.2411dB for "Carphone" with Q=8 and $C_t = 0.3$). When $C_t = 0.1$, the adaptive algorithm can maintain the target complexity with only a small drop in PSNR except in the case of "Carphone" with Q=8. This is because "Carphone" contains more motion and detail than the other two sequences. Hence it is more difficult to achieve a target complexity of C_t=0.1 (i.e. to discard 90% of blocks) without a significant loss of image quality. The last column in this table is convergence time and it occurs within 23 frames in the worst case.

Table 5-2 Performance of adaptive algorithm (k=6)

Sequence	Ct	Q	Mean DCT complexity	STD DCT complexity	Mean PSNR drop (dB)	Convergence (frames)
Claire	0.1	8	0.1009	0.0113	0.4300	3
		16	0.0997	0.0087	0.0161	7
	0.3	8	0.2968	0.0137	0.0276	17
		16	0.2928	0.0368	0.0	21
	0.5	8	0.4883	0.0467	0.0026	22
		16	0.4792	0.0760	0.0	23
Carphone	0.1	8	0.1051	0.0277	1.8294	4
		16	0.1027	0.0121	0.7028	2
	0.3	8	0.3044	0.0247	0.2411	10
		16	0.3007	0.0281	0.0087	10
	0.5	8	0.5007	0.0322	0.0227	8
		16	0.4952	0.0576	0.0007	16
Mother and Daughter	0.1	8	0.1017	0.0202	0.6269	5
		16	0.1002	0.0128	0.1475	5
	0.3	8	0.3002	0.0093	0.0772	1
		16	0.2965	0.0353	0.0019	10
	0.5	8	0.4942	0.0285	0.0384	18
		16	0.4872	0.0068	0.0088	22

5.5.2 Profiling results

The actual processor utilization was measured by profiling an H.263 encoder (based on H.263 TMN-5) that incorporates the adaptive update algorithm (Equation 5-10). The encoder uses a fast motion estimation algorithm, three-step search, with a search window of +/-7.5 luminance samples. The Forward and Inverse DCTs are calculated using the algorithm described in [56]. Figure 5-13 shows the actual coding time spent on each function for 200 frames of the "Mother and Daughter" sequence, with k=6, Q=8 and C_t =0.3. Only the most computationally intensive functions are listed. "SAD_Macroblock" is the function that calculates SAD for integer-accuracy motion estimation and "FindHalfPel" calculates the half-pixel component of the motion vector.

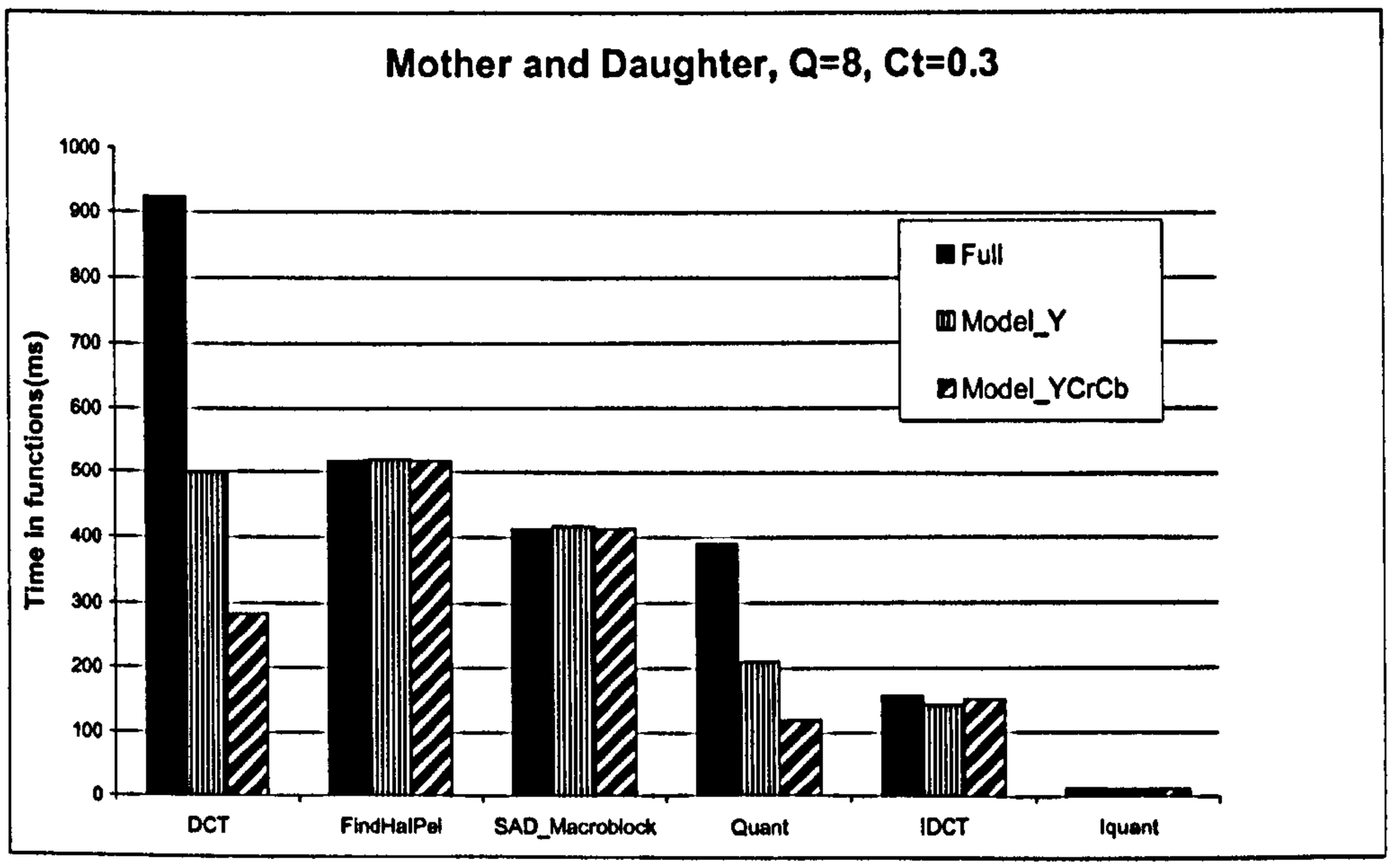


Figure 5-13 Profiling result of Mother and daughter with k=6, Q=8 and Ct=0.3

The black bars show the result when a full FDCT operation is carried out for each block and the DCT is clearly the most computationally intensive function. The "Model_Y" bars show the result of applying the proposed adaptive algorithm to the luminance blocks only, with a target complexity of 30%. The algorithm successfully reduces the number of calculations for luminance blocks by around 70% but the total computation time for the forward DCT (and also the Quantiser function) is only reduced by about 47% because DCT and Quantisation are still calculated for every chrominance block.

The "Model_YCrCb" bars show the results when the adaptive algorithm is applied to both luminance and chrominance blocks. There is a small extra computational overhead because SAD_B is calculated for the Cr and Cb blocks in each macroblock (this value is not normally calculated during motion estimation). However, the results clearly show that the complexity reduction outweighs this extra overhead and the encoder achieves around 69% reduction in complexity of the DCT and Quantiser functions.

5.6 Summary

The DCT and related functions have been demonstrated to be computationally complex functions in a software video encoder. In this chapter, a complexity-reduction algorithm is proposed to control DCT complexity by comparing statistical information about each block with a threshold in order to determine whether to perform DCT and quantisation operations on the block. The efficiency of the algorithm depends on how accurately it can estimate the block information and it has been demonstrated that block SAD can provide more accurate prediction of block coefficients than macroblock SAD. Using a fixed decision threshold in the algorithm results in unpredictable performance, with variable complexity and quality depending on the time-varying content of the encoded video sequence. In order to achieve adjustable control of complexity, the adaptive algorithm presented in this chapter uses a feedback loop to update the threshold on a frame basis. The changing of threshold compensates for variation in scene content so that the complexity control algorithm becomes independent of the input video signal statistics. The proposed algorithm enables the encoder to maintain a near-constant target level of computational complexity throughout the video sequence.

The complexity, quality and convergence performance of the proposed algorithm have been tested by applying it to video sequences with various levels of activity and a range of quantisation step sizes. Results demonstrate that it works well, enabling the encoder to reduce the complexity of DCT and quantiser functions to a target level and keep it at this level throughout the entire video sequence with little deviation. The adaptive algorithm can achieve complexity reduction by up to 70-90% with minimal video quality loss. Profiling results show that the extra computation for calculating the SAD value of chrominance block is small compared with the total complexity reduction and therefore can be neglected.

Chapter 6 Computational complexity control of motion estimation

6.1 Introduction

Chapter 5 investigates an adaptive algorithm to manage the computational complexity of the DCT and related functions. This chapter will address the complexity problem of motion estimation function.

Motion estimation has been demonstrated to be a computationally expensive function since the invention of video coding techniques. Many attempts have been made to speed up the process of block-based motion search in the last two decades, resulting in some efficient fast search algorithms as described in Section 2.3.1. Most of these algorithms save computations by reducing the number of candidates for motion search (such as three-step search [6], four-step search [79]) or by reducing the number of luminance samples used in the block matching process (SAD or SSE introduced in Section 2.3.1) [10, 11, 80] . In [10], the fast block matching process can be obtained by sub-sampling the luminance components in horizontal and vertical directions. Computation scalable algorithms [81-83] have been proposed to further the complexity reduction of the specified fast motion search algorithms. In [81], the authors proposed a method to control the complexity of a motion estimation algorithm (3-D recursive search block-matcher) [84]. The computation-constrained motion estimation search algorithm described in [82] is reported to reduce computation of the Nearest Neighbour Search (NNS) by varying a control parameter of search termination criterion. The value of the control parameter is chosen based on experiments and pre-determined for each sequence, and is therefore not able to provide flexible, predictable control of computational complexity or tradeoffs between complexity and rate-distortion performance.

Most existing "fast search" and "computation scalable" motion estimation algorithms are input dependent, i.e. the computational cost of motion estimation varies depending on the content of the input video sequence. Typically, complex or high-motion scenes lead to an increased number of search operations and can therefore place an unacceptably high burden on the processor. When the computational burden is too high, the only solution is

for the software encoder to "skip" frames and reduce the encoded frame rate. This in turn leads to "choppy" video at the decoder. It would be preferable in many cases to manage the computational complexity of motion estimation in order to reduce the number of skipped frames and maintain a smoother video sequence, perhaps at the expense of a slight reduction in image quality.

The adaptive algorithm proposed in this chapter provides efficient and predictable control of the complexity of motion estimation based on a particular fast search algorithm, Nearest Neighbour Search (NNS) [82] . This algorithm can obtain a pre-determined target complexity regardless of the content of the input sequence. This allows the encoder to adjust complexity to meet the available processor utilization. A feed-back controller is added to this algorithm to maintain the complexity of each frame at a target level by flexibly adjusting the motion estimation search pattern. Experimental results show that flexible management of computational complexity can be achieved with only a small degradation in rate-distortion performance.

6.2 The "Nearest Neighbour Search" fast search algorithm

Nearest Neighbour Search (NNS) is a very efficient integer-pixel motion search algorithm and can be adopted as an alternative to the full search in H.263 / MPEG-4 video encoders. The search origin is the median-predicted motion vector that is derived from previously transmitted motion vectors of spatially or temporally neighbouring macroblocks. Sum of Absolute Differences (SAD) described in Equation 2-2 is employed as the block match measurement. Figure 6-1 illustrates the search procedure of NNS: this algorithm always searches the four neighbouring locations in an equal size diamond-shaped "layer". Then, the neighbour location with minimum SAD is chosen to be the centre point for the next search "layer" and its four neighbours are processed. The basic rule is that layer $i+1$ is centred at the location with smallest SAD of layer i and it always contains four immediate neighbours. After the first layer, there are still four candidates in each diamond-shaped layer, but only 2 or 3 of them need to be tested at each layer (because the rest of the candidates belong to previous layers). The search is terminated (a) when all candidates of the current layer have been considered and the minimum SAD value of the current layer is larger than that of the previous layer or (b) when the next layer would extend outside the search area. In addition to searching around

the median-predicted position, the (0,0) position (the position of the current macroblock) is also tested.

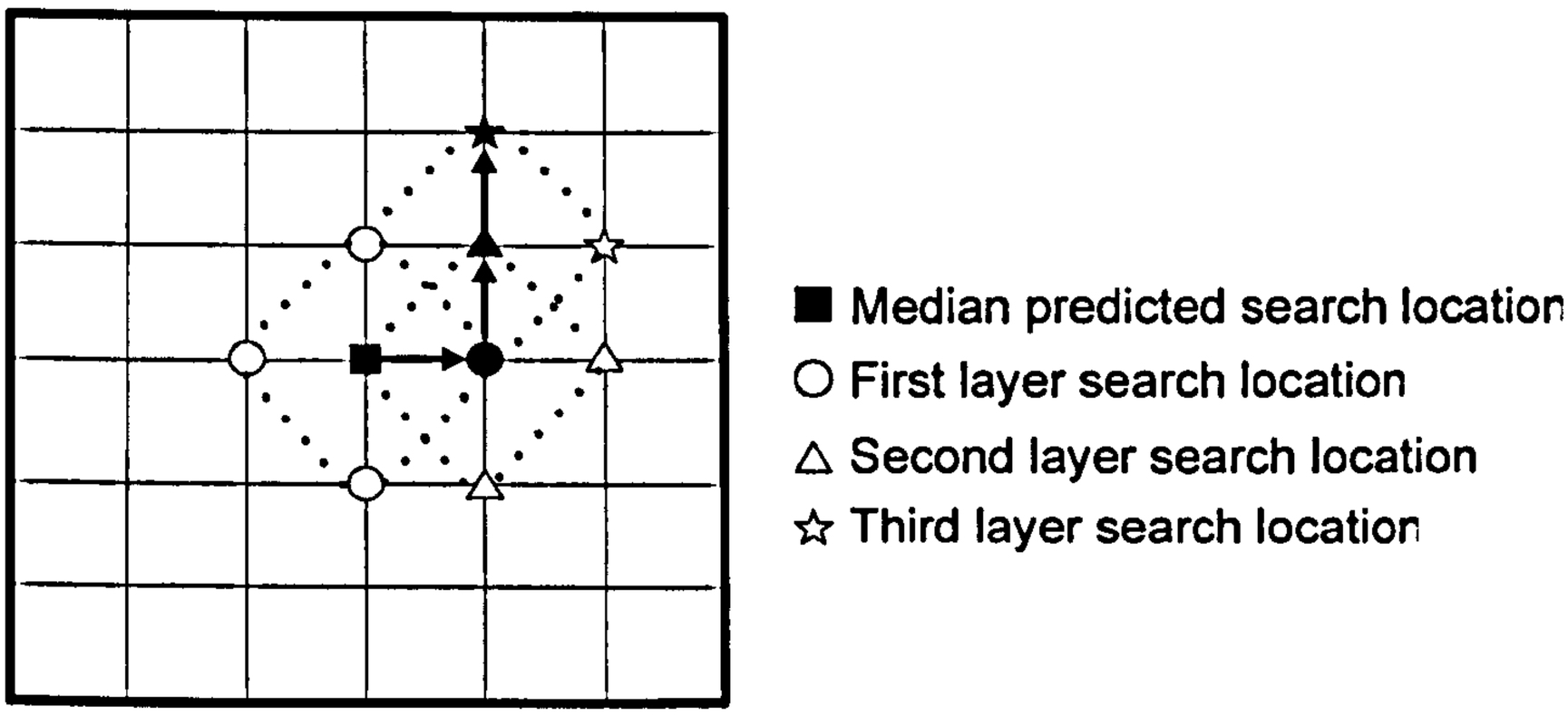


Figure 6-1 Search path of Nearest Neighbour Search

The practical NNS algorithm used in TMN8 H.263+ encoder model [45] requires to be performed on at least three layers for each macroblock because the termination rule is only switched on after three layers have been searched. In other words, the search continues until after the third search layer and when the termination rules are met, which can be described as:

$$SAD_L > SAD_{L-1} \quad \text{AND} \quad L \geq 3$$

Equation 6-1

L is the number of current search layer, SAD_L represents the minimum SAD value of the current layer and SAD_{L-1} is the minimum SAD value of the last search layer. The search origin of NNS is located by the median-predicted motion vector, which is close to the global minimum (the best matching position) in the search area. Searching at least three layers for every macroblock reduces the possibility of getting trapped in local minima around the median-predicted motion vector. However, for those macroblocks whose best matching motion vector is at (0,0) position or within three searching layers, additional operations will slow down the coding process. Generally, L will not become very large when the search is terminated.

6.3 Reducing motion estimation complexity

6.3.1 Complexity of motion estimation

In block-based motion estimation, the Sum of Absolute Difference (SAD) needs to be calculated at each search location for the current macroblock (MB). During the calculation of SAD for a 16×16 MB, each pixel requires three operations: a subtraction, an absolute-value calculation and an addition, so the total number of operations for one SAD calculation is 16×16×3. The overall computation cost for calculating SAD per macroblock is (number of search locations) ×16×16×3 operations. The computation of SAD takes up most of the computation resources spent in motion search so that the number of SAD operations can be used to represent the complexity of motion estimation function.

In the NNS motion search algorithm, there are at most four search candidates at each layer; the number of SAD operations (i.e. the complexity of motion estimation) is approximately linearly related to the number of layers for motion search. The more layers are searched, the more SAD operations are required, hence the more complex the motion search algorithm is. Therefore, it may be possible to reduce the number of SAD operations by controlling the search layer. Unnecessary search locations (which will not result in global minima) can be ignored by terminating the search at a proper time. This may provide a trade-off between complexity and rate-distortion performance.

6.3.2 Upper-layer and lower-layer termination algorithm

A straightforward way to discard the SAD operations is limiting the upper bound of the layer (L_{upper}) that may be searched. Motion search has to stop when the current search layer (L) is greater than the pre-determined L_{upper}, even though the normal termination rules ($SAD_L > SAD_{L-1}$) are not met. This upper-layer termination algorithm can be defined as:

$$SAD_L > SAD_{L-1} \quad \text{OR} \quad L \geq L_{upper}$$

Equation 6-2

By using this upper-layer termination rule, there is no complexity reduction or video quality loss for macroblocks whose best match motion vector can be found within L_{upper}. However, for those macroblocks whose best motion vector goes beyond L_{upper}, the use of upper layer bound L_{upper} will decrease the number of SAD operations (i.e. complexity of motion estimation), with a corresponding loss of accuracy of the motion vector.

In order to test the influence of fixing the upper-search layer on the motion search result, the NNS in H.263+ TMN8 encoder is integrated with the upper-layer termination algorithm. 200 frames of video sequence "Foreman" are coded at QCIF resolution with L_{upper} =1, 2, 3, 4 and fixed quantiser step size Q = 8. Rate control is not switched on.

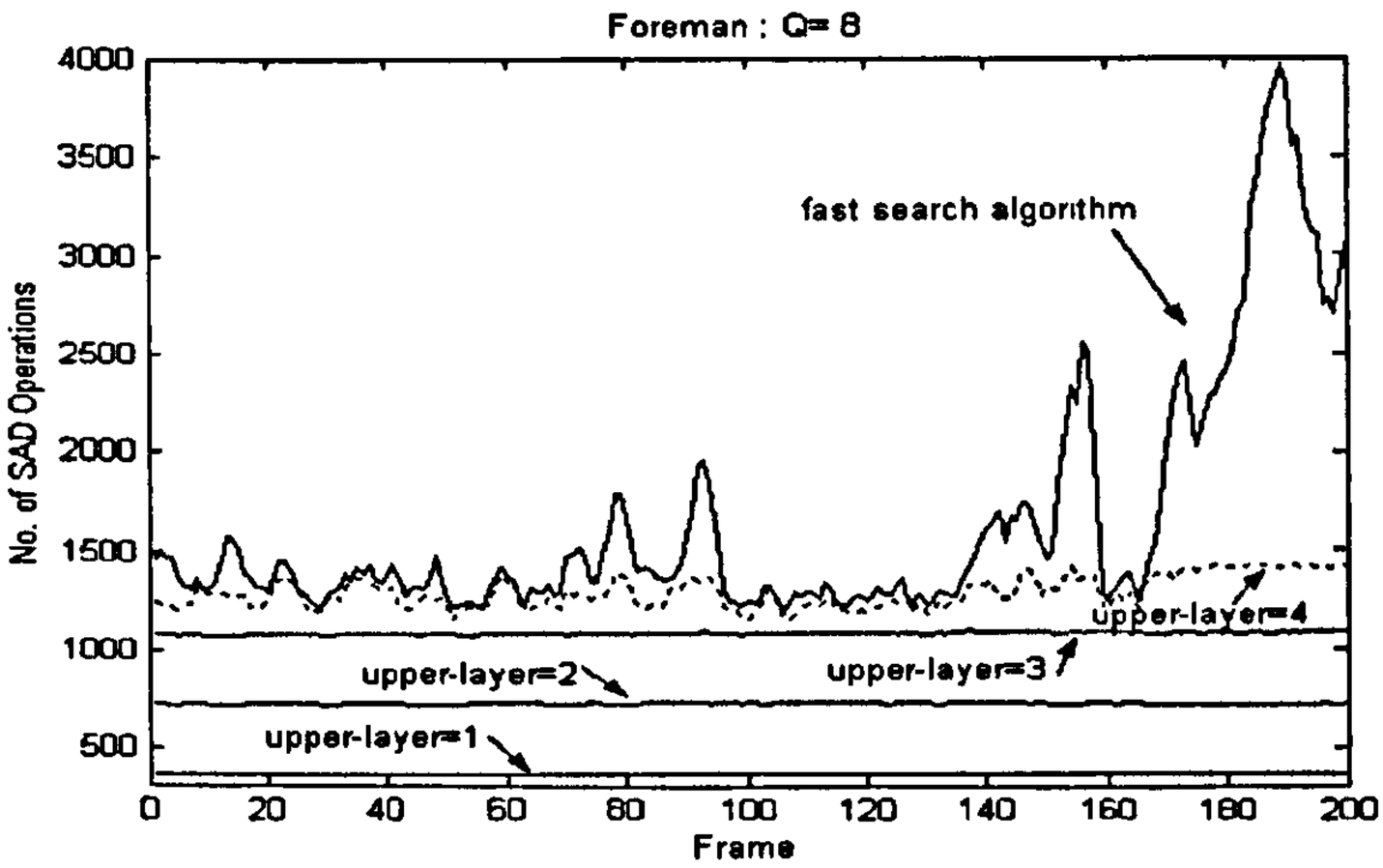


Figure 6-2 Complexity Comparison of upper-layer search algorithm and fast search algorithm

Figure 6-2 shows the computational complexity of motion estimation (represented by total number of SAD operations per frame) with various L_{upper} values. The complexity of the "full-complexity" NNS fast search algorithm (where no algorithm is applied) varies with the content of the input video; in contrast, the complexity of the upper-layer search algorithm is kept at a certain level and oscillates slightly. From this figure, it can be seen that the complexity of L_{upper} =4 is very close to the curve of "full complexity" NNS in

most range of frames, which indicates the motion vector for most of macroblocks can be found in a small range and only some macroblocks with high activities will go beyond this range. Figure 6-3 compares the rate-distortion performance of "full complexity" NNS and upper-layer search algorithm. It is clear that the significant reduction of computational complexity is achieved at the expense of poorer rate-distortion performance.

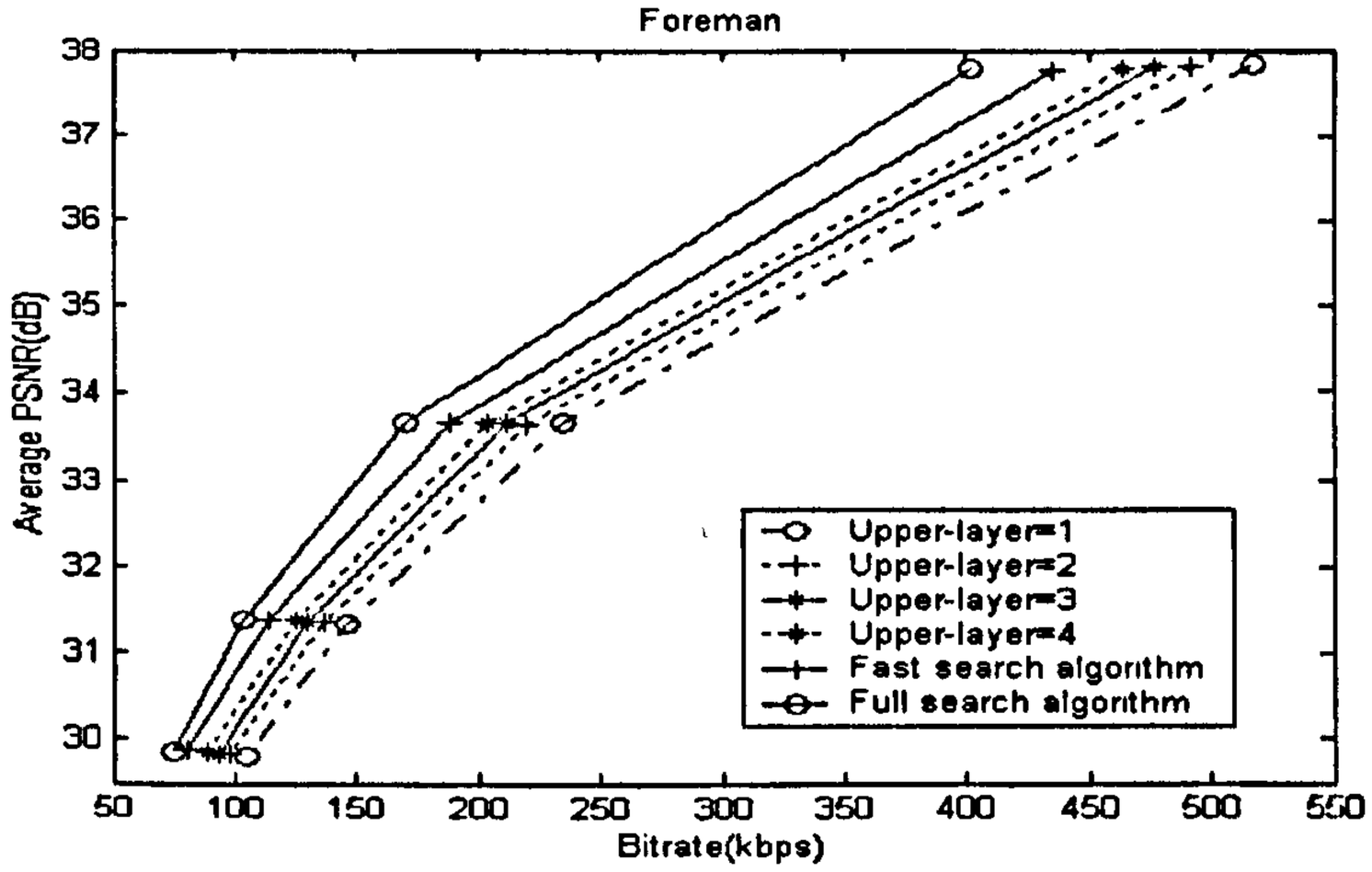


Figure 6-3 Rate-distortion performance comparison of upper-layer search algorithm and fast search algorithm

The implementation of NNS described in [45] does not end motion search when the termination rules are satisfied until it reaches the third layer. Searching until the third layer will waste computation resources for macroblocks whose best motion vector is in the first layer or in the second layer. The lower-layer search algorithm allows the motion search to stop after the first or second layer and the search continues when:

$$SAD_L > SAD_{L-1} \quad \text{AND} \quad L \geq L_{lower}$$

Equation 6-3

Where L_{lower} =1, 2.

The complexity performance and rate-distortion performance of the "Foreman" sequence, coded by H.263+ Tmn8 CODEC incorporated with the lower-layer search algorithm, are illustrated in Figure 6-4 and Figure 6-5 respectively. From Figure 6-4, it can be seen that the shape of complexity curves of L_{lower} =1 or 2 is very similar to the "full complexity" one and significant complexity (the number of SAD operations) has been reduced. This means that a lot of motion searches naturally stop at the first or second layer. Results in Figure 6-5 show that there is only a little drop in rate-distortion performance when the lower layer termination algorithm is applied.

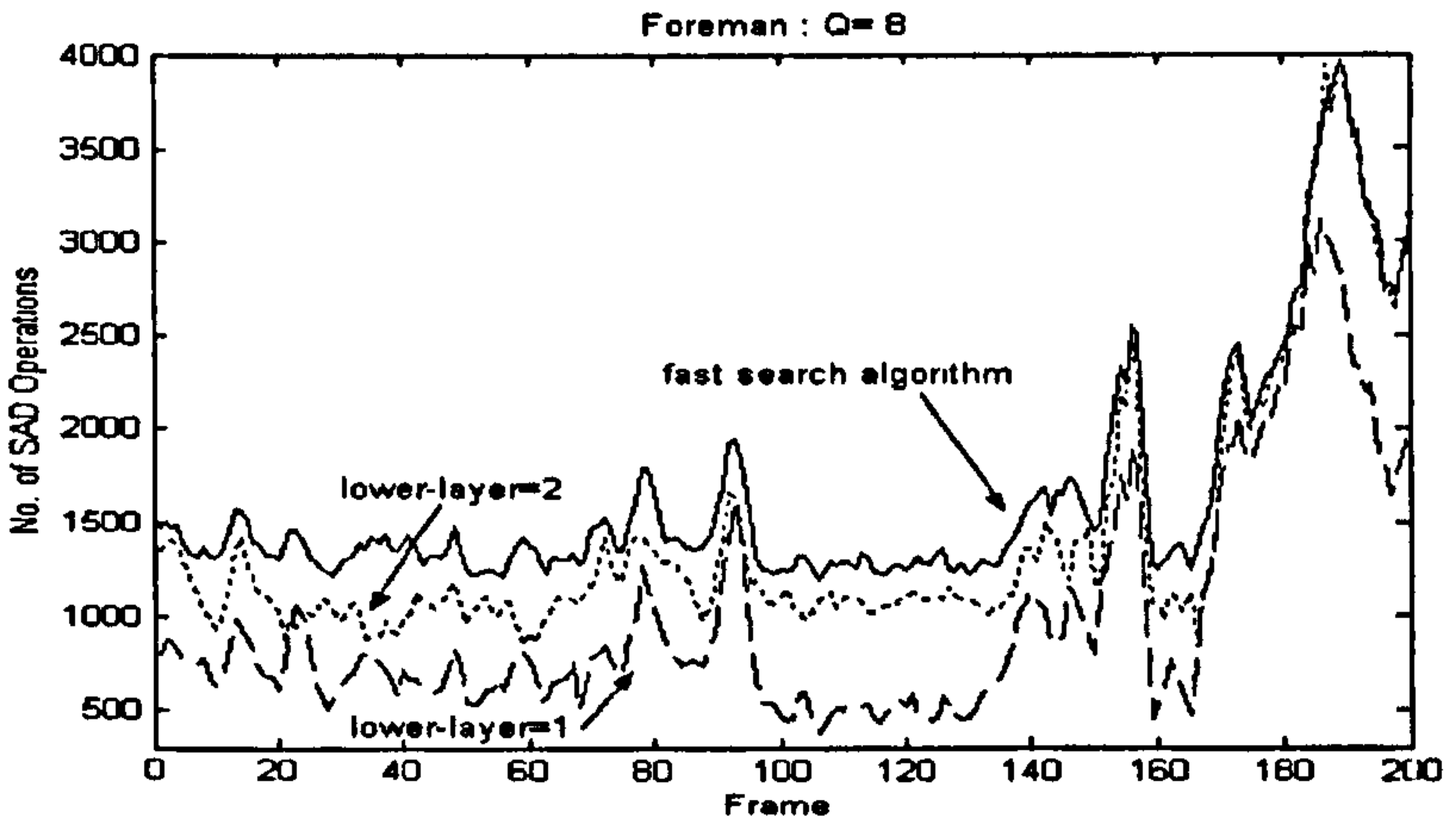


Figure 6-4 Complexity comparison of lower-layer search algorithm and fast search algorithm

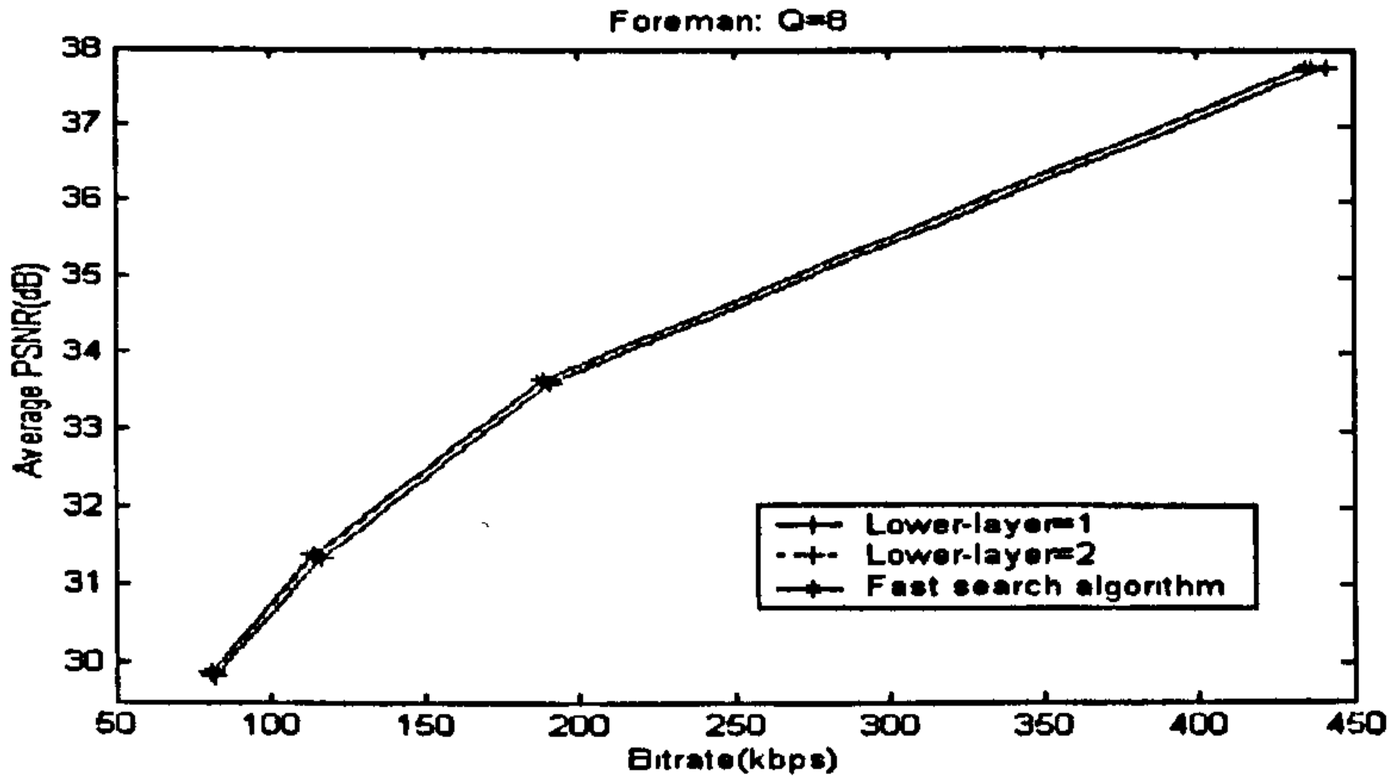


Figure 6-5 Rate-distortion performance comparison of lower-layer search algorithm and fast search algorithm

6.3.3 Relationship of complexity, bit rate and video quality

Experimental results of upper-layer and lower-layer termination algorithms show a certain range of trade-offs between complexity and rate-distortion performance. Combining both upper-layer and lower-layer algorithms to control the available search range can clearly demonstrate the detailed variation of complexity and its effect on video quality and bitrate. The new termination algorithm is as follows:

$$SAD_L > SAD_{L-1} \quad \text{AND} \quad L \geq L_{lower} \quad \text{OR} \quad L \geq L_{upper}$$

Equation 6-4

Where, L_{lower} =1, 2, 3 and $L_{upper} \geq L_{lower}$.

Two video sequences "Foreman" and "Mother and Daughter" are coded with a range of quantiser (Q=4, 8, 12, 16) and combinations of lower-layer and upper-layer search algorithms listed in Table 6-1. Figure 6-6 and Figure 6-7 plot the relationship of computational complexity, bit rate and video quality (represented by PSNR) for "Foreman" and "Mother and Daughter" respectively. A wide range of computational complexity of motion estimation is obtained by using both upper-layer and lower-layer algorithms. The rate-distortion performance varies with the different combinations of lower-layer and upper-layer. Through selecting the appropriate combination of lower-layer and upper-layer, it should be possible to obtain near-optimum quality whilst ensuring that computational complexity does not exceed a "target" value. For the high movement sequence "Foreman" in Figure 6-6 and the moderate movement sequence "Mother and Daughter" in Figure 6-7 the algorithm with $L_{lower} = 1$ outperforms the others in terms of rate-distortion performance versus computational complexity reduction.

Table 6-1 The combination of upper-layer and lower-layer

Lower Layer	Upper Layer
1	1,2,3,4,5,6…
2	2,3,4,5,6…
3	3,4,5,6…

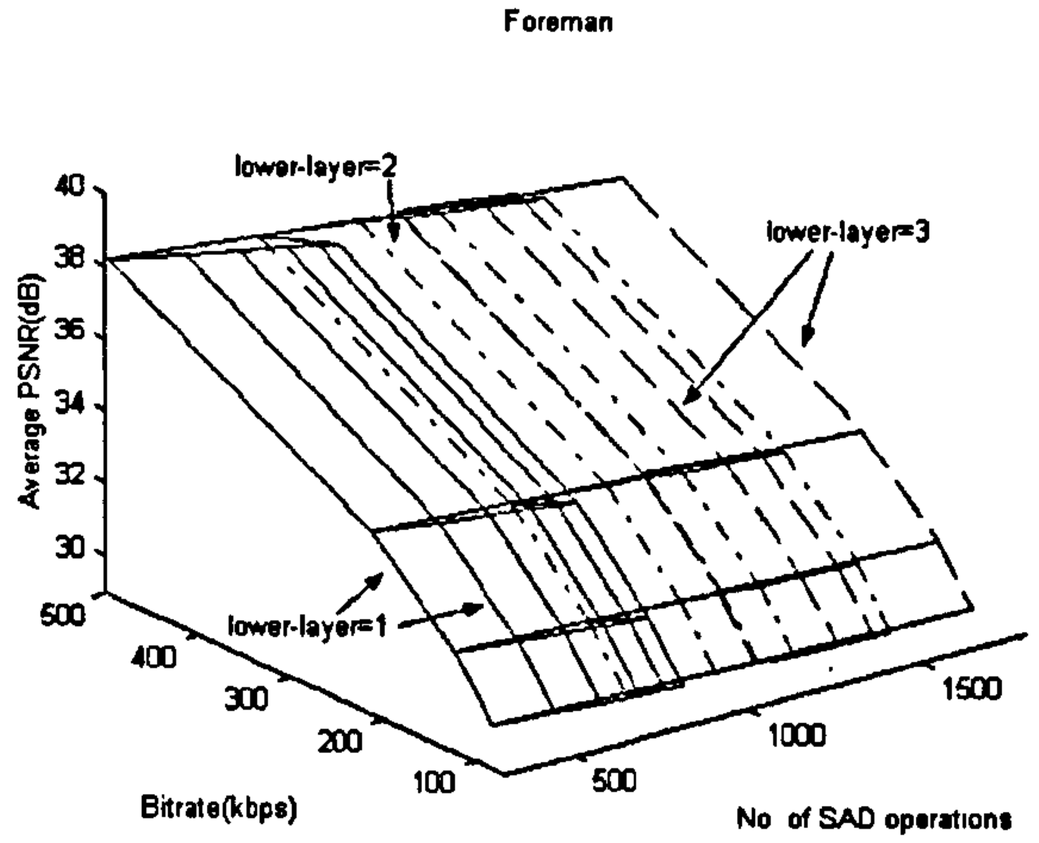


Figure 6-6 Correlation of complexity, bit rate and PSNR for "Foreman"

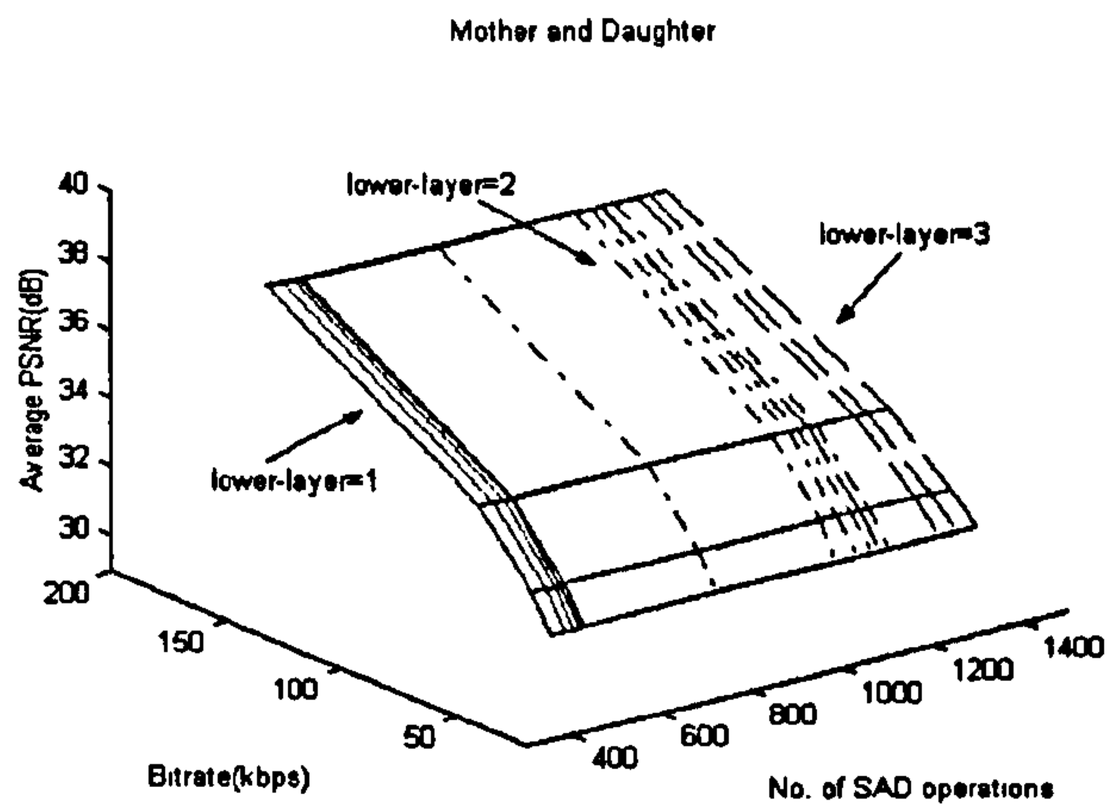


Figure 6-7 Correlation of Complexity, bit rate and PSNR for "Mother and daughter"

Operating with a fixed lower and upper layer enables a reduction of computational complexity but the algorithm is still "input-dependent", i.e. the computational cost depends on the input sequence and is therefore not fully predictable. In the next section methods are examined for adaptively controlling computation in order to achieve a "target" computational complexity.

6.4 Adaptive control of motion estimation complexity

Four video sequences "Foreman", "Carphone", "Mother and Daughter" and "Clair" are coded with $L_{lower} = 1$, L_{upper} from 1 to 10 and quantiser step sizes (Q) = 4, 8, 12 and 16. These sequences offer a combination of various types of video content. Figure 6-8 plots the complexity of motion estimation (represented by number of SAD operations) against L_{upper} for each video sequence and Q. It can be seen for the same video sequence that complexity curves with various values of Q are approximately co-located, which indicates that the relationship of complexity and L_{upper} is independent of quantisation step size. When L_{upper} is less than or equal to 5, it is clear that the number of SAD operations increases quickly with L_{upper} ; however, the rate of increase slows down after L_{upper} becomes greater than 5. This means there are only fewer macroblocks whose best motion vector is located in the high search layer, especially for low-movement video sequences. Setting L_{upper} to a high value will not contribute significantly to the total computational complexity.

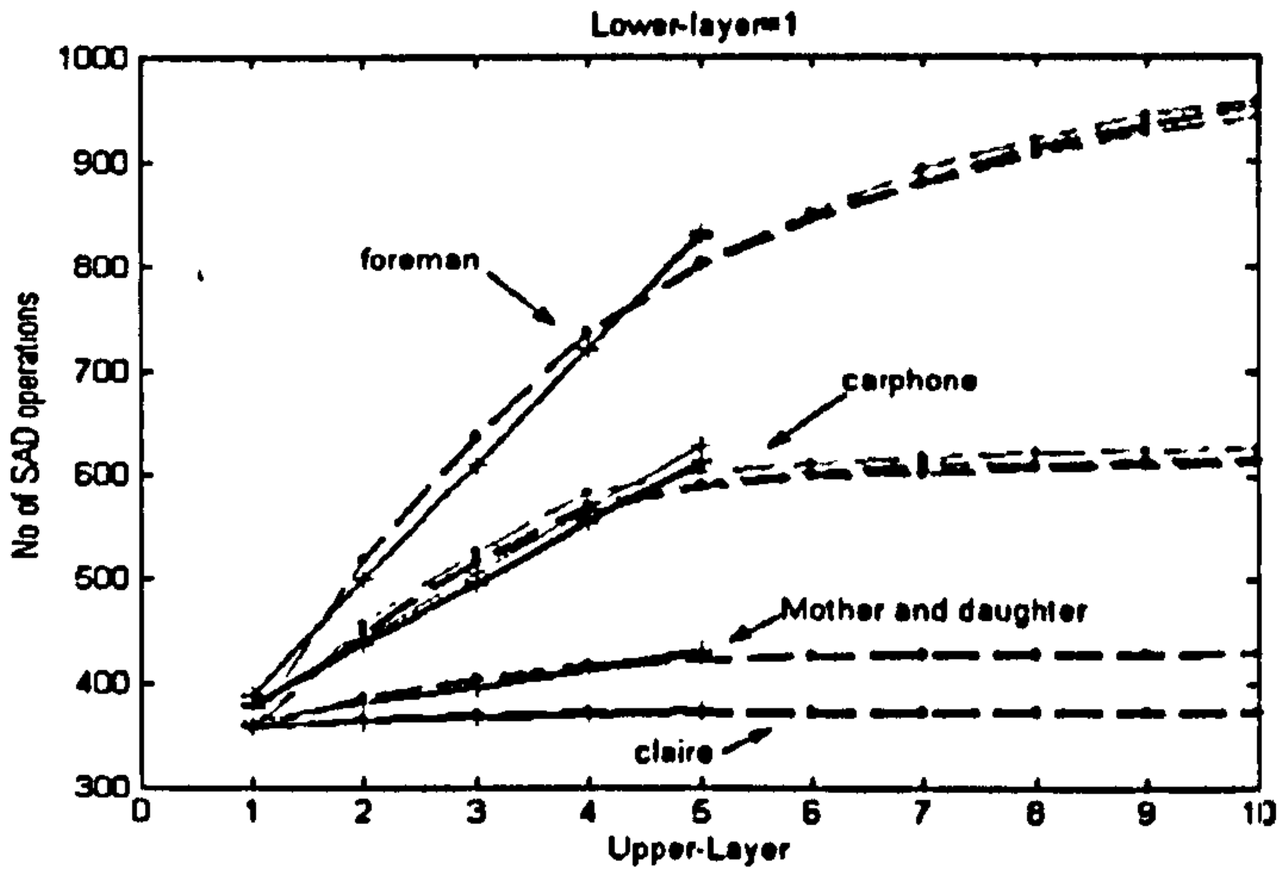


Figure 6-8 Result of polynomial fit for "Foreman", "Carphone", "Mother and Daughter" and "Claire" with lower-layer=1

The computational complexity curve can be described approximately by a polynomial equation, which is plotted superimposed on the complexity curves in Figure 6-8. The slope of the straight-line approximations varies depending on the amount of activity and level of detail in the sequence. Therefore, the relationship between L_{upper} and the number of SAD operations (S) can be approximately represented by a linear relationship for lower values of L_{upper} :

$$S = K \times L_{upper} + A$$

Equation 6-5

L_{upper} is equal to 1,2,3,4,5 and K, A are determined by the characteristics of the video sequence.

Given a target number of SAD operations (S_t) for motion estimation, the objective of the adaptive algorithm is to achieve and maintain the target complexity for any video sequence whilst minimizing any increase in distortion.

After completing encoding frame n, the actual number of SAD operations for motion estimation in frame n (S_n) can be obtained. According to Equation 6-5, the relationship between S_n and the upper-layer of frame n ($L_{upper(n)}$) can be described as:

$$S_n = K \times L_{upper(n)} + A$$

Equation 6-6

Therefore, parameter k is obtained by arranging this equation:

$$K = \frac{S_n - A}{L_{upper\ (n)}}$$

Equation 6-7

For frame n+1, upper-layer has to be chosen in order to achieve the target complexity (number of SAD operations), so the relationship of complexity and L_{upper} for frame $n+1$ is given by:

$$S_t = K \times L_{upper(n+1)} + A$$

Equation 6-8

Hence,

$$L_{upper(n+1)} = \frac{S_t - A}{K} = L_{upper(n)} \times \frac{S_t - A}{S_n - A}$$

Equation 6-9

This adaptive algorithm proceeds as follows:

1. Choose initial value for A
2. Set initial L_{upper} to achieve target number of SAD operations S_t
3. Encode frame n and measure the actual number of SAD operations S_n
4. Update L_{upper} using Equation 6-9
5. Go to step 3

Equation 6-9 will usually produce a non-integer value of $L_{upper(n+1)}$ and so the result is rounded to the nearest integer in the range 1-5.

6.5 Experimental Results

In the simulation, a software H.263+ encoder (based on TMN8) is used to evaluate the adaptive complexity-reduction algorithm described in Equation 6-9. Test video sequences are "Foreman", "Carphone" and "Mother and Daughter", which provide various scenes with different levels of object movement. The picture format is QCIF (176×144) and frame rate is set to 30 frames per second. Test sequences are coded with a range of quantiser values (Q=4,8,12,16) and target SAD (S_t),(i.e. number of SAD operations for motion search) from 500 to 800 increasing by 100 at a time. The search window size is set to +/-7.5 pixels and rate control and optional modes are disabled. By experiment, the parameter A is set in the adaptive algorithm to be 100. This value gives good results for all the sequences and it does not appear necessary to vary A adaptively.

Figure 6-9, Figure 6-10 and Figure 6-11 illustrates the corresponding computational complexity, video quality (PSNR) and rate-distortion of "Foreman" video sequence with Q=8 and $A = 100$. "Foreman" is a video sequence with high object motion and rapid changes of scene starting from frame 160, which results in a large number of SAD operations in motion estimation, especially in frames 160 to 180. The target complexity (number of SAD operations) is set to 500 or 800 and, at each target, the obtained complexity of every single frame and corresponding PSNR are plotted in Figure 6-9 and Figure 6-10 respectively. It is clear that the complexity is reduced to the target levels (S_t= 500 and S_t= 800) and maintained reasonably closely to these levels throughout the sequence. For the scene changing area (from frame 160 to frame 200), it can be seen that the obtained complexity oscillates in order to achieve the target due to the adjustment of the adaptive algorithm. PSNR shown in Figure 6-10 is largely unchanged throughout the video sequences for both the targets (as would be expected with a fixed quantiser step size). Compared with the "full complexity" NNS fast search algorithm, the rate distortion performance of the adaptive algorithm in Figure 6-11 is slightly reduced by around 0.5-0.7dB. Clearly, the curve of S_t= 800 gives better rate-distortion performance than that of S_t= 500, but it achieves less complexity reduction. This demonstrates the relationship between coding performance and complexity as introduced in Section 4.4.

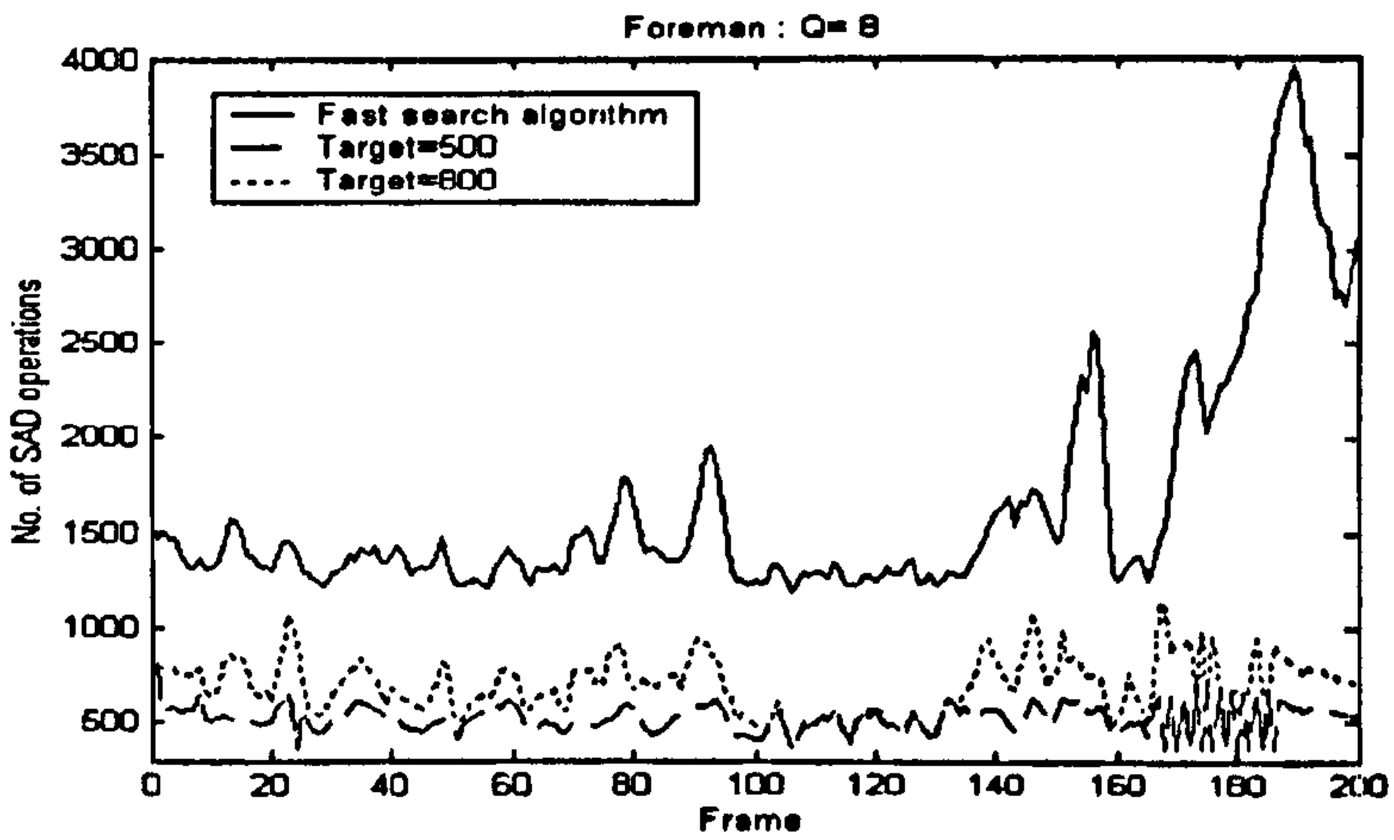


Figure 6-9 Computational complexity of "Foreman" with Q=8

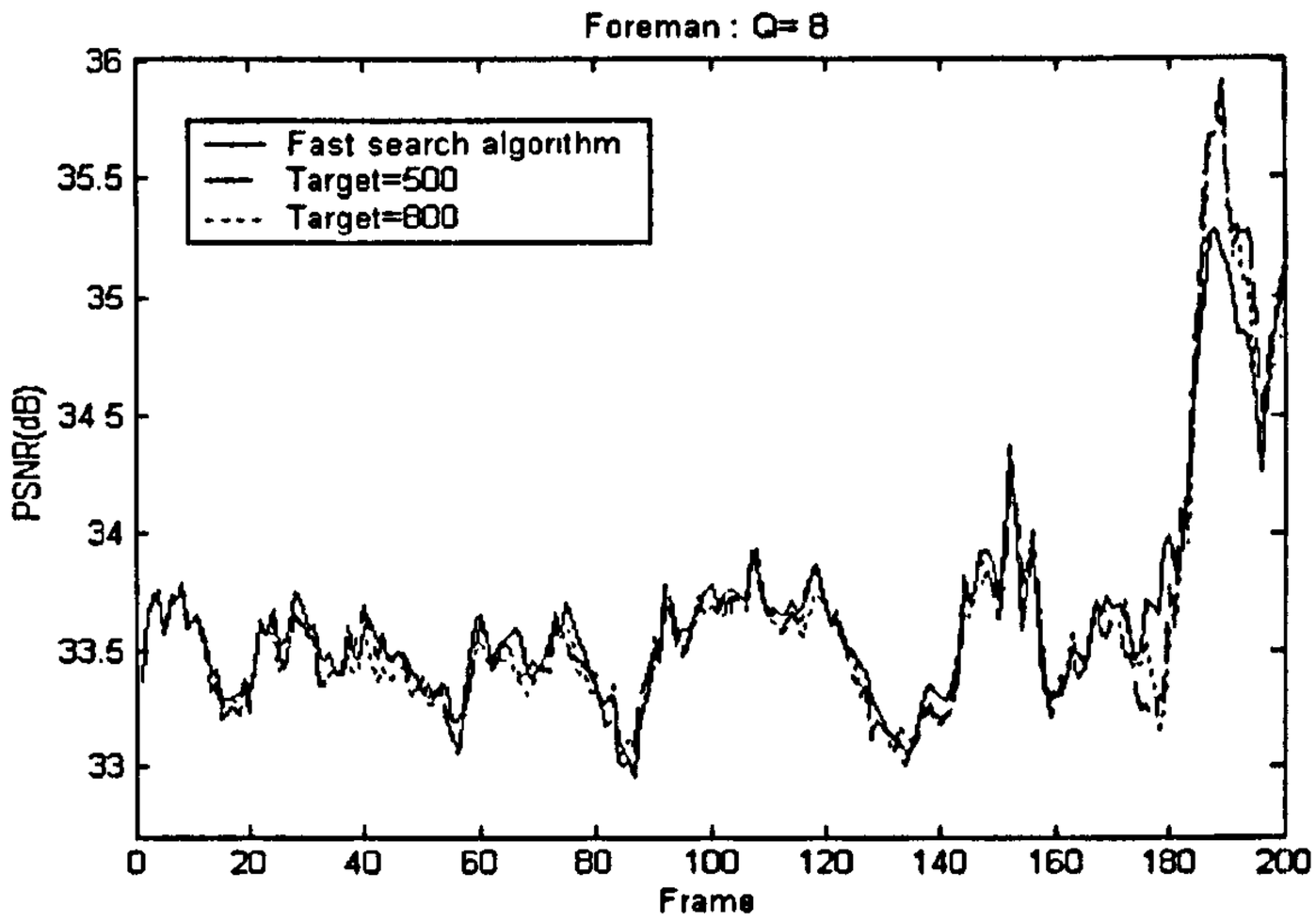


Figure 6-10 PSNR of "Foreman" with Q=8

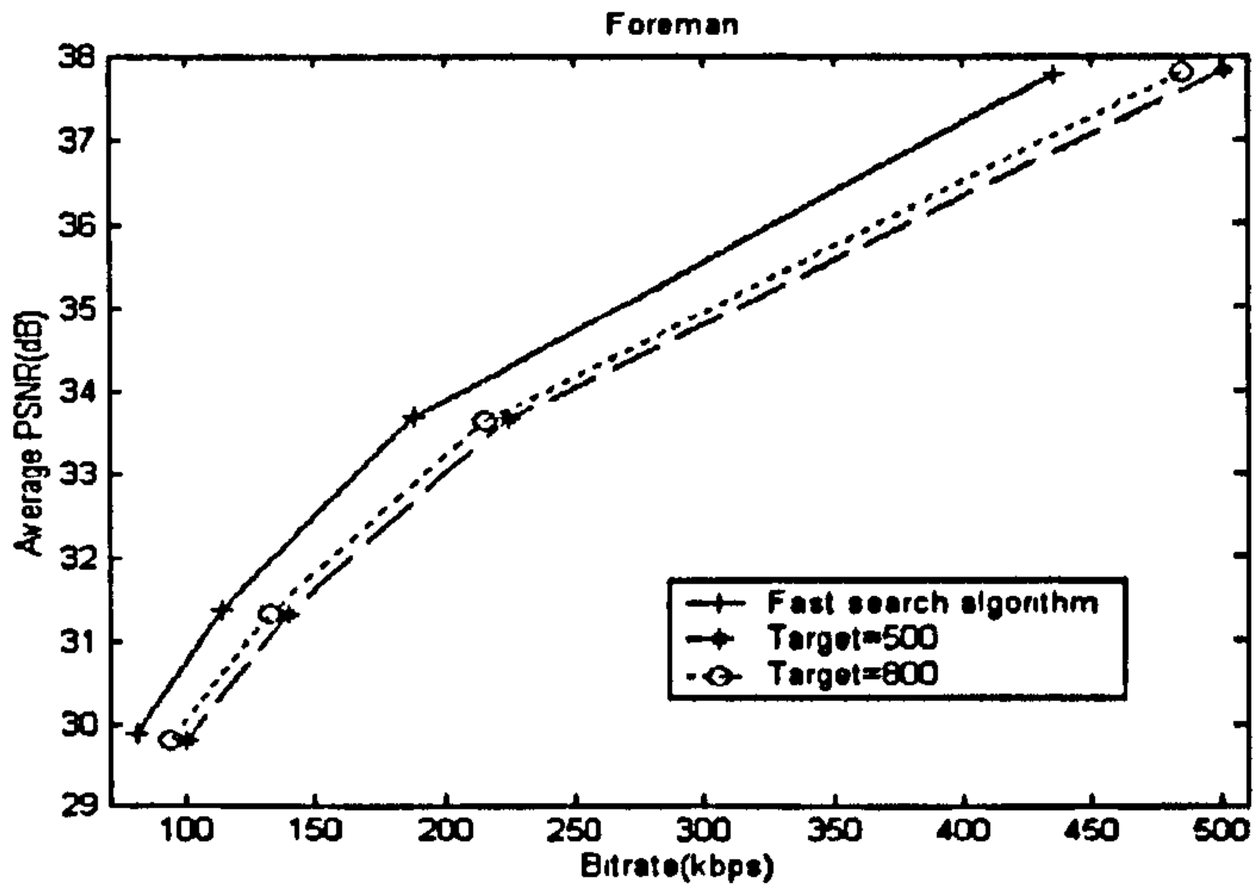


Figure 6-11 Rate-distortion performance of "Foreman" with Q=8

Experimental results for "Mother and Daughter" are shown in Figure 6-12, Figure 6-13 and Figure 6-14. Figure 6-12 plots the computational complexity of each frame at Q=8. The complexity drops to the same level (around 430) for both "targets" (S_t= 500 and S_t= 800) because "Mother and Daughter" is a moderate-movement sequence and the SAD operations can be reduced dramatically by setting lower-layer to 1. The corresponding PSNR for both of the targets (shown in Figure 6-13) is almost the same as that of "full" NNS fast search throughout the sequence, which shows there is negligible drop in video

quality even with the large amount of complexity reduction. From Figure 6-14, the rate-distortion curve of target complexity =500 or 800 can not be seen clearly as they overlap that of "full" NNS fast search. This indicates that rate-distortion performance is maintained by using the adaptive complexity-reduction algorithm.

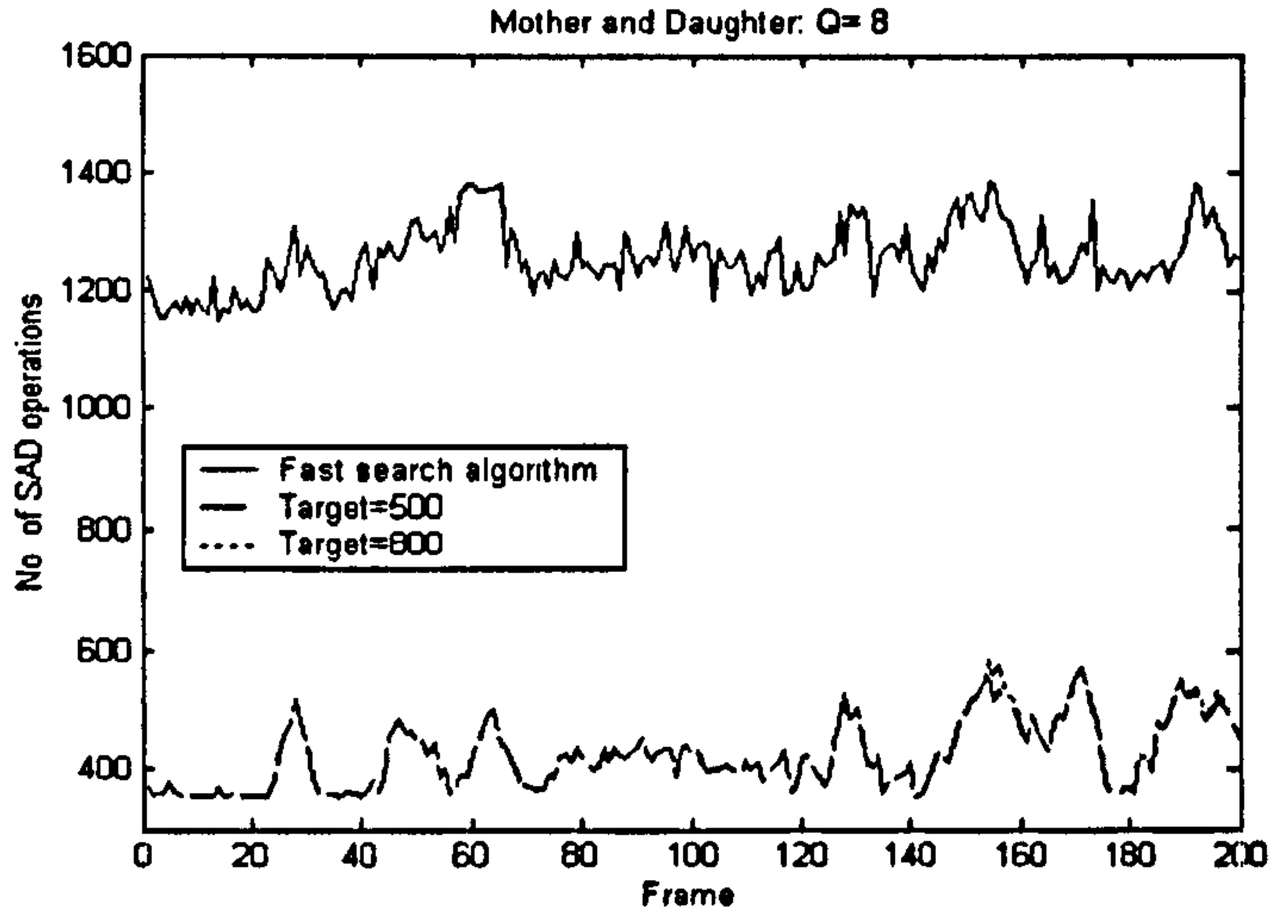


Figure 6-12 Computational complexity of "Mother and daughter" with Q=8

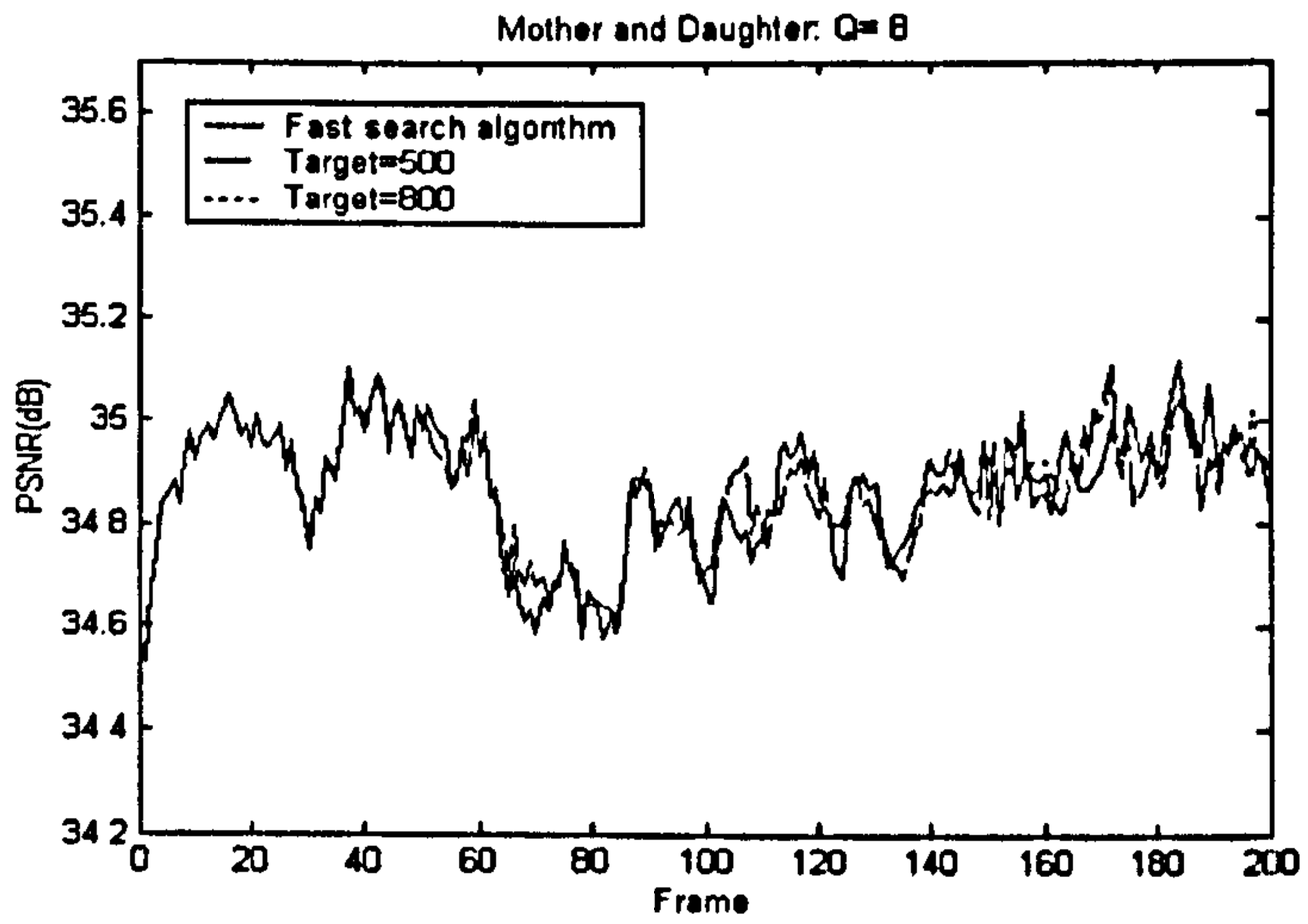


Figure 6-13 PSNR of "Mother and daughter" with Q=8

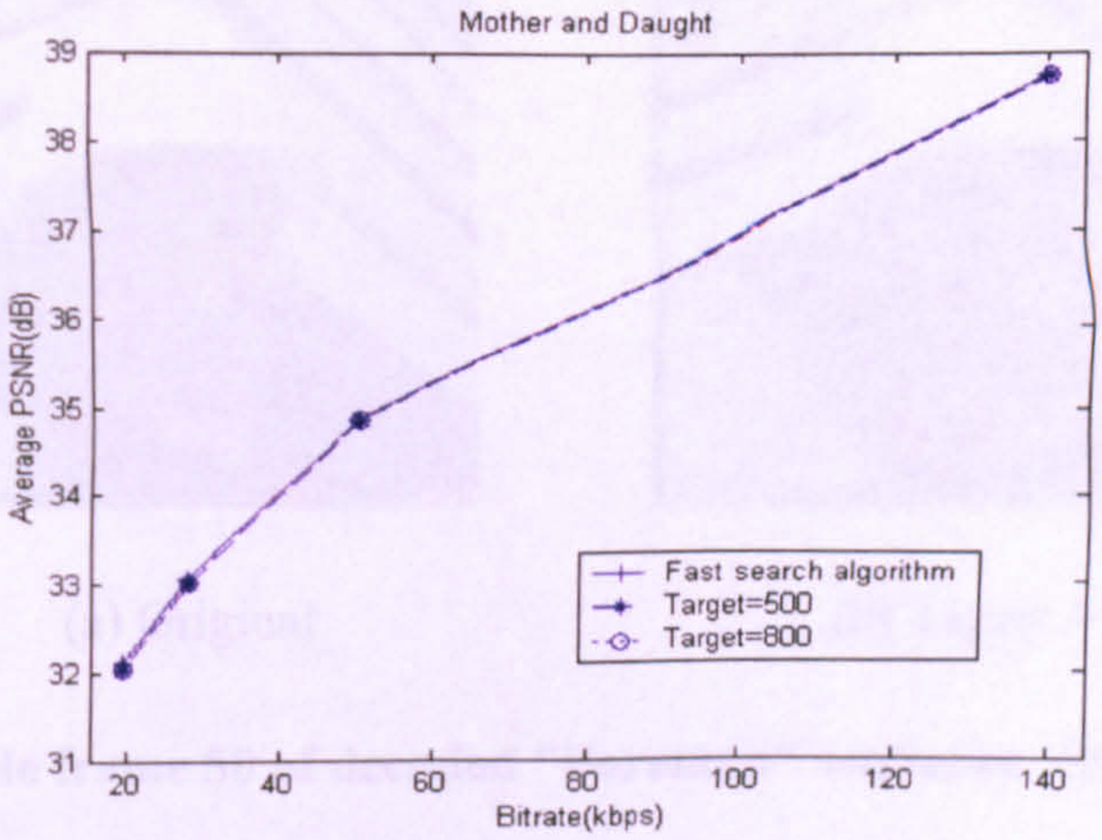


Figure 6-14 Rate-distortion performance of "Mother and daughter" with Q=8

Figure 6-15 compares frame 50 of the "Mother and daughter" video sequence without complexity reduction and with target = 800. It is very clear that there is no difference between the two sample frames, indicating the video sequence with complexity reduction in motion estimation has the same subjective video quality as the original video sequence. Figure 6-16 shows the same result for the "Foreman" video sequence.

(a) Original (b) Target = 800

Figure 6-15 Sample frame 50 of decoded "Mother and Daughter" sequence with Q=8 and target=800

(a) Original (b) Target = 800

Figure 6-16 Sample frame 50 of decoded "Foreman" sequence with Q=8 and target=800

Table 6-2, Table 6-3 and Table 6-4 summarize the results for each test sequence with various Q and target complexities. The computational complexity of motion estimation in each sequence is represented by the mean number of SAD operations. Mean PSNR drop (compared with the "full" complexity NNS motion search) is used to measure the video quality and bitrate is given in terms of kilo-bits per second. These tables also list the original bit rate and mean number of SAD operations of the "full complexity" NNS fast search, for comparison with the adaptive algorithm. The detailed results of "Foreman" and "Mother and Daughter" are also listed in Table 6-3 and Table 6-4 respectively and are similar to that of "Carphone".

The results of "Carphone" in Table 6-2 show that target complexity has been achieved: the mean number of SAD operations has significantly dropped from 1416 (when Q=8) and 1438 (when Q=16) to the target level (492 for Q=8 and 489 for Q=16) when S_t= 500. For other targets (S_t= 600, 700 or 800), the complexity has been reduced to less than the pre-determined target because the number of SAD operations can be easily decreased to a low level by setting lower-layer =1. The mean PSNR drop of "Carphone" is less than or equal to 0.09dB, a drop in video quality that is not likely to be obvious to viewers [85]. The obtained bit rate of "Carphone" coded with Q=8 and target = 500 is 124.32 kbps by an increase of only 4.05kbps (3%) when compared with the original bit rate of "Carphone" with "full" NNS fast search. This is because the adaptive algorithm saves complexity whilst decreasing the accuracy of the motion vector, resulting in higher energy in the residual macroblock, and therefore higher bit rate. Similar results can be found for "Carphone" with other combinations of Q and target except for Q=16 and target

= 500, where the bit rate is reduced from 46.57 kbps ("full complexity" fast search) to 45.58 kbps. The detailed results of "Foreman" and "Mother and Daughter" are also listed in Table 6-3 and Table 6-4 respectively and they are similar to that of "Carphone".

Table 6-2 Performance of adaptive algorithm on "Carphone"

Sequence	Bit rate (kbps) (fast search)		Mean no. of SAD operations (fast search)		Target	Q	Mean no. of SAD Operations	Mean PSNR drop (dB)	Bit-Rate (kbps)
	Q=8	Q=16	Q =8	Q=16					
Carphone	120.3	46.6	1416	1438	500	8	492	0.08	124.32
						16	489	0.02	45.58
					600	8	536	0.02	124.24
						16	534	0.01	47.32
					700	8	566	0.02	123.58
						16	566	0.07	47.37
					800	8	590	0.09	123.12
						16	586	0.02	47.07

Table 6-3 Performance of adaptive algorithm on "Foreman"

Sequence	Bit rate (kbps) (fast search)		Mean no. of SAD operations (fast search)		Target	Q	Mean no. of SAD operations	Mean PSNR drop (dB)	Bit rate (kbps)
	Q=8	Q=16	Q =8	Q=16					
Foreman	188.4	81	1645	1674	500	8	521	0.0	224.59
						16	521	0.01	99.76
					600	8	590	-0.01	219.86
						16	590	-0.03	97.5
					700	8	657	0.21	217.3
						16	660	0.26	95.82
					800	8	703	0.18	215.99
						16	714	0.25	94.41

Table 6-4 Performance of adaptive algorithm on "Mother and Daughter"

Sequence	Bit rate (kbps) (fast search)		Mean no. of SAD operations (fast search)		Target	Q	Mean no. of SAD Operations	Mean PSNR drop (dB)	Bit-Rate (kbps)
	Q=8	Q=16	Q =8	Q=16					
Mother & Daughter	50.1	19.7	1256	1297	500	8	425	-0.07	50.39
						16	426	0.1	19.82
					600	8	429	-0.08	50.31
						16	430	0.17	49.65
					700	8	429	-0.08	50.31
						16	430	0.17	19.65
					800	8	429	-0.08	50.31
						16	430	0.17	19.65

6.6 Summary

In this chapter, methods for decreasing the computational complexity of the motion estimation function are presented for a specified fast search algorithm: nearest neighbour search. Through examining the inherent relation between complexity and upper search layer (L_{upper}), an adaptive algorithm is proposed to flexibly control the computational complexity by varying L_{upper}, which is updated on a frame basis based on the obtained complexity of the previous frame. The experimental results show that motion estimation complexity can be successfully controlled by the proposed algorithm. The adaptive algorithm described here enables a flexible trade-off between computational complexity (in this case, the number of SAD operations) and rate-distortion performance. It could form part of a video encoder that can optimise performance in scenarios where computational resources are restricted.

Chapter 7 Macroblock classification and complexity control

7.1 Introduction

Methods to manage the complexity of computationally expensive functions have been investigated in Chapter 5 and Chapter 6. The proposed adaptive complexity-control algorithms for DCT and motion estimation can flexibly trade-off the computational complexity and rate-distortion performance in order to meet the specified requirement. The complexity-reduction methods for motion estimation in the literature are usually dedicated for a particular fast motion search algorithm, for example, [86] proposed a new specific block matching algorithm incorporated with complexity-distortion optimisation and the methods developed in Chapter 6 are based on Nearest Neighbour Search. This chapter will investigate approaches to reduce the computational complexity of the entire encoding process regardless of the type of motion estimation algorithm.

Sum of Absolute Differences (SAD) has been proposed to reduce the complexity of DCT and related functions by predicting non-zero coefficients in a MB [75, 77] and the algorithm proposed in Chapter 5 improves the accuracy of prediction by using block SAD instead of macroblock SAD. In [87], SAD is used to predict zero motion vectors to save the computation in motion estimation and the authors report that this method performs well together with DCT computation reduction.

In a block-based video coding system, such as H.263+/MPEG-4 Simple Profile, motion estimation and compensation, DCT transform and quantisation are the main processes to reduce temporal and spatial redundancy. After that, there are two important parameters remaining for each macroblock: (1) motion vectors (MV), which point to the most similar matching region in a reference frame and (2) quantised coefficients (QCoeff), the residual signal in a MB. Since successive video frames are similar, many coded macroblocks in an inter-coded frame have zero motion vectors and/or quantised coefficients. A MB with zero MV and no non-zero coefficients is skipped after processing. If these skipped macroblocks could be accurately predicted prior to encoding, all subsequent operations

on these macroblocks (motion estimation and compensation, DCT, quantisation, etc) could be avoided, saving considerable computational effort.

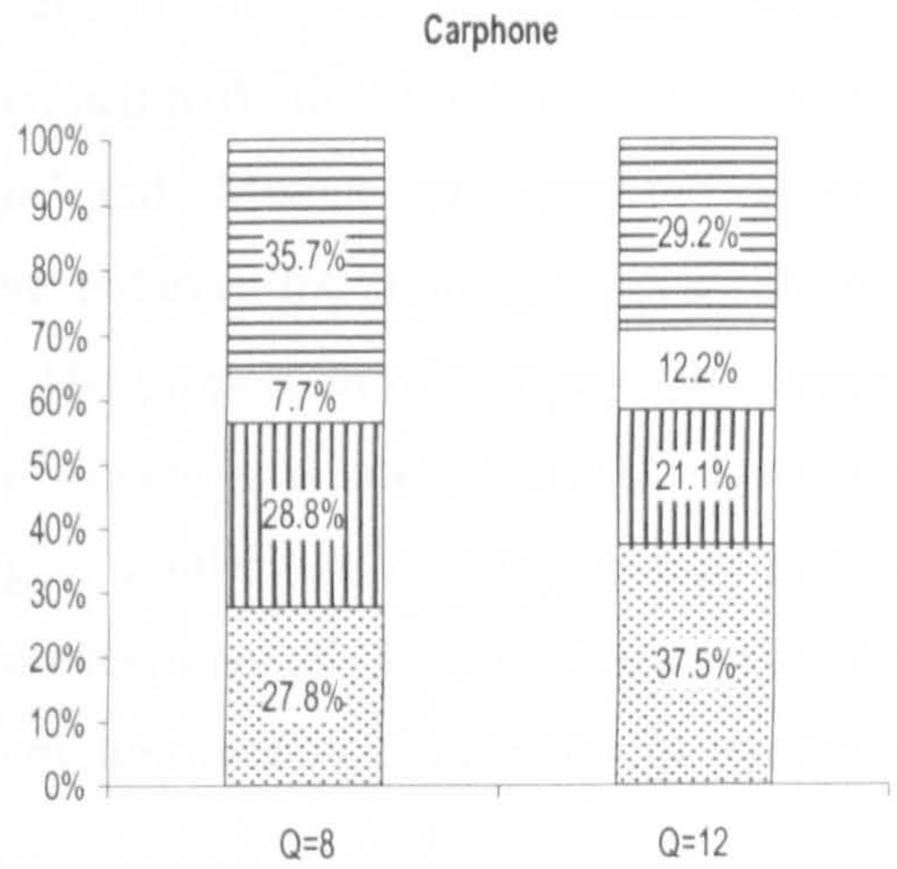
This chapter presents a pre-classification algorithm that categorises MBs into two types, "skipped" and "not skipped", prior to encoding. Computational complexity can be reduced by not processing MBs that are expected to be skipped. The proposed algorithm provides a simple, controllable and robust method of managing computational resources. It can control the amount of reduced computation whilst minimising distortion due to occasional incorrect macroblock classification.

The organization of this chapter is as follows. Section 7.2 introduces the concepts of macroblock classification and methods to reduce the computational complexity. Section 7.3 describes the development of a pre-classification algorithm based on measured characteristics of an input macroblock. Simulation results for the proposed algorithm and conclusions are presented in Section 7.4 and Section 7.5 respectively. This work has also been published in [88]; the full paper is appended with this thesis.

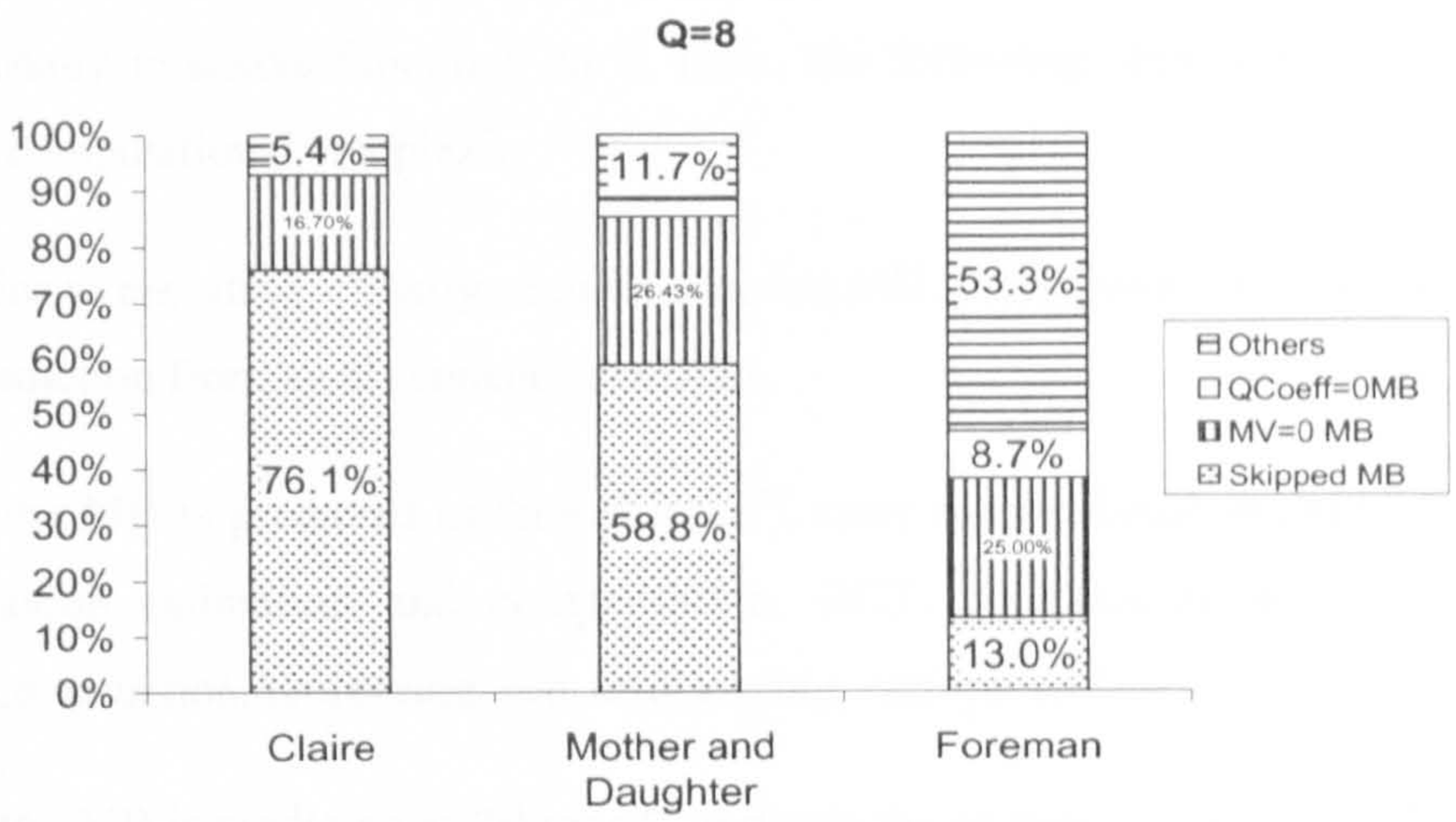
7.2 Distribution of Macroblock types

An encoder that conforms to one of the popular DCT-based video coding standards (such as H.263 or MPEG-4 Simple Profile) processes each frame in units of a Macroblock. In an inter-coded picture, motion estimation is carried out in order to find a suitable prediction for the current macroblock from reference frames. Each block of the motion-compensated residual MB is coded using the DCT, quantisation, reordering and entropy coding. Motion vectors and quantised coefficients are encoded together with side information and the macroblock is reconstructed for prediction of further pictures.

Four video sequences ("Carphone", "Mother and Daughter", "Foreman" and "Claire") were encoded using an H.263+ encoder (corresponding to the low complexity mode of test model TMN10 [89]; hereafter described as "TMN10") with a fixed quantiser step size (Q=8, 12). Coded MBs in P-pictures were categorised into four types based on the values of MV and QCoeff: (1) "skipped" (zero MV, no non-zero QCoeff), (2) "MV=0" (zero MV, some non-zero QCoeff), (3) "QCoeff=0" (non-zero MV, no non-zero QCoeff) and (4) "other" (non-zero MV and non-zero QCoeff).



(a) Carphone



(b) Q=8

Figure 7-1 Distribution of four types of MBs in video sequences

Figure 7-1 shows the distribution of four categories (a) for "Carphone" encoded with two quantiser step sizes (Q=8, 12) and (b) for the remaining three sequences encoded with a quantiser step size of 8. Figure 7-1(a) demonstrates that the proportion of skipped macroblocks increases with increment of quantiser step size (Q). When Q increases, DCT coefficients will be divided by a larger number and therefore there is a higher probability that the DCT coefficients will become zero after quantisation. It is clear from Figure 7-1

(b) that low-activity sequences such as "Claire" and "Mother and Daughter" contain a higher proportion of skipped MBs than high-motion sequences such as "Carphone" and "Foreman". For "Claire" and "Mother and Daughter", more than half of the MBs are skipped during encoding and even for "Foreman", there are still 13% MBs containing all zero MV and QCoeff.. The proportion of skipped MBs depends on the degree of detail and activities in a video sequence: in low-detail and low-motion sequences, neighbouring frames tend to have higher similarities, resulting in high probabilities of zero MV and/or zero QCoeff. With the exception of "Foreman", the majority of macroblocks in each sequence (i) contain no motion vectors, (ii) contain no coefficients or (iii) are skipped.

It is clear from Figure 7-1 that a significant proportion of MBs are skipped (not coded), particularly in low-motion sequences and/or at higher quantiser step sizes (and hence lower bitrates). Predicting the presence of a skipped macroblock prior to coding could make it possible to save considerable computational resources by not carrying out computationally intensive functions on it. Here, the following approach is proposed to reduce the computational complexity:

- Prior to encoding, classify each inter-coded MB as "skipped" or "not skipped" by prediction from local sequence statistics.
- If the MB is predicted as "not skipped", carry out the usual encoding functions (motion estimation and compensation, DCT, quantisation, rescaling, IDCT, reconstruction, reordering, run-level coding, entropy coding).
- If the MB is predicted as "skipped", indicate the presence of a skipped MB in the bit stream; no further processing needs to be carried out.

If the prediction of MB type is correct, computational complexity is reduced without any effect on decoded video quality. If a macroblock that should have been encoded (i.e. a macroblock that contains non-zero MV and/or QCoeff after encoding) is wrongly predicted as "skipped", a reduction in decoded quality is likely to occur. Computational complexity reduction may therefore lead to increased distortion. [90] demonstrates that objective video quality measurement (PSNR) does not always correlate with subjective video quality. It has been argued previously [78] that a small reduction in PSNR is an acceptable penalty for reduced computation. A limited degradation in PSNR (less than

1dB) is difficult to distinguish subjectively by audiences [85] and hence the subjective video quality will not be significantly affected. Furthermore, in a real-time video application, maintaining a consistent video frame rate through computational complexity management (at the expense of limited reduction in PSNR) is likely to be preferable to a "jerky" decoded video sequence due to an encoder dropping frames.

7.3 Macroblock classification algorithms

7.3.1 Correlation between residual energy and probability of skipped macroblock

Macroblocks that are skipped have zero MV and no non-zero QCoeff. This means that (a) the closest matching region is in the same position in the reference frame and (b) the energy of the residual MB (after subtracting the reference region from current macroblock) is low, such that there are no non-zero DCT coefficients after quantisation. Both of these conditions are likely to be met if there is a strong similarity between the current MB and the same MB position in the reference frame. Without motion compensation, the energy of the residual MB formed by subtracting the reference MB (in the same location as current MB) from the current MB is approximated by $SAD0_{MB}$ (sum of absolute differences for luminance part of macroblock, zero displacement):

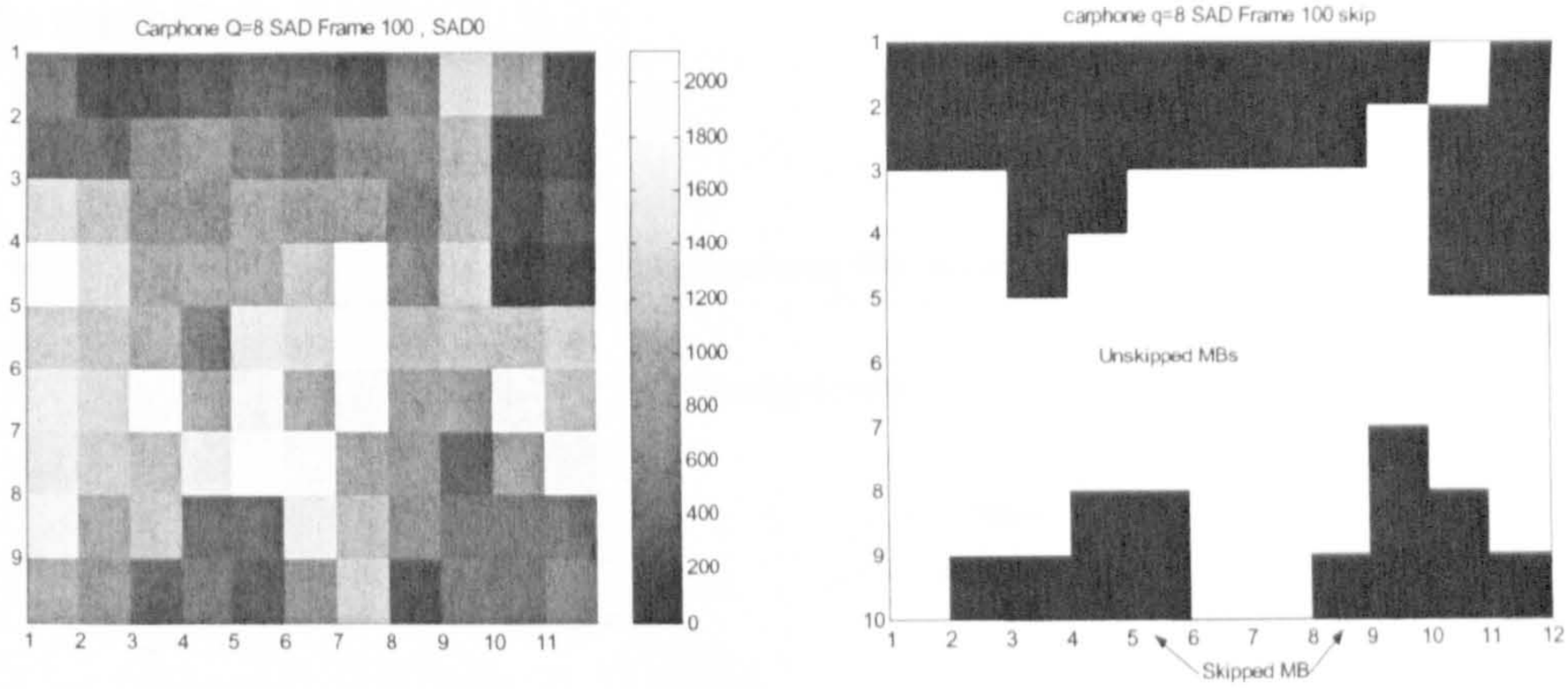
$$SAD0_{MB} = \sum_{i=0}^{15} \sum_{j=0}^{15} \left| C_C(i,j) - C_P(i,j) \right|$$

Equation 7-1

$C_C(i,j)$ and $C_p(i,j)$ are luminance samples from an MB in the current frame and in the same position in the reference frame, respectively.

$SAD0_{MB}$ represents the total energy of a residual MB. A skipped MB has low residual energy and a zero motion vector and so there may be a correlation between $SAD0_{MB}$ and P(skip), the probability of skipping the current MB. Figure 7-2(a) plots $SAD0_{MB}$ for each MB in frame 100 of the "Carphone" sequence (QCIF format, encoded with a fixed quantiser step size of 8). The dark colour corresponds to a low value of $SAD0_{MB}$. Figure 7-2(b) plots the distribution of skipped and unskipped MBs after coding this frame using

TMN10 and the black region signifies the skipped MBs. It is clear that skipped MBs tend to have low values of $SAD0_{MB}$. It may therefore be possible to use $SAD0_{MB}$ to predict whether a MB is likely to be skipped.



(a) $SAD0_{MB}$ **(b) Skipped MB**

Figure 7-2 SAD0MB and skipped MB (Carphone, Q=8, Frame 100)

7.3.2 Classifying skipped MBs using $SAD0_{MB}$

The "Carphone" sequence was coded using TMN10 with fixed quantiser step size of 8. The dotted region of Figure 7-3 plots $SAD0_{MB}$ (x-axis) against P(skip), the probability that a MB with a given $SAD0_{MB}$ will be skipped (y-axis). The solid line plots the cumulative density (CDF) of $SAD0_{MB}$ and each pair of co-ordinates in this line shows the percentage of the MBs (Y co-ordinate) whose $SAD0_{MB}$ value is less than the x co-ordinate.

Figure 7-3 indicates that $SAD0_{MB}$ correlates with P(skip): MBs with low $SAD0_{MB}$ value tend to have a high probability of being skipped. It therefore may be possible to use $SAD0_{MB}$ as a predictor to determine whether a given MB is likely to be skipped. An algorithm for macroblock classification can be described in pseudocode as follows:

IF $SAD0_{MB} < T_{sad0}$

THEN skip coding this MB and set MV and QCoeff to zero

ELSE continue coding

When $SAD0_{MB}$ is less than a pre-determined threshold (Tsad0), the encoder skips coding this MB and sets MV and QCoeff to zero directly, hence reducing the processing required for the macroblock.

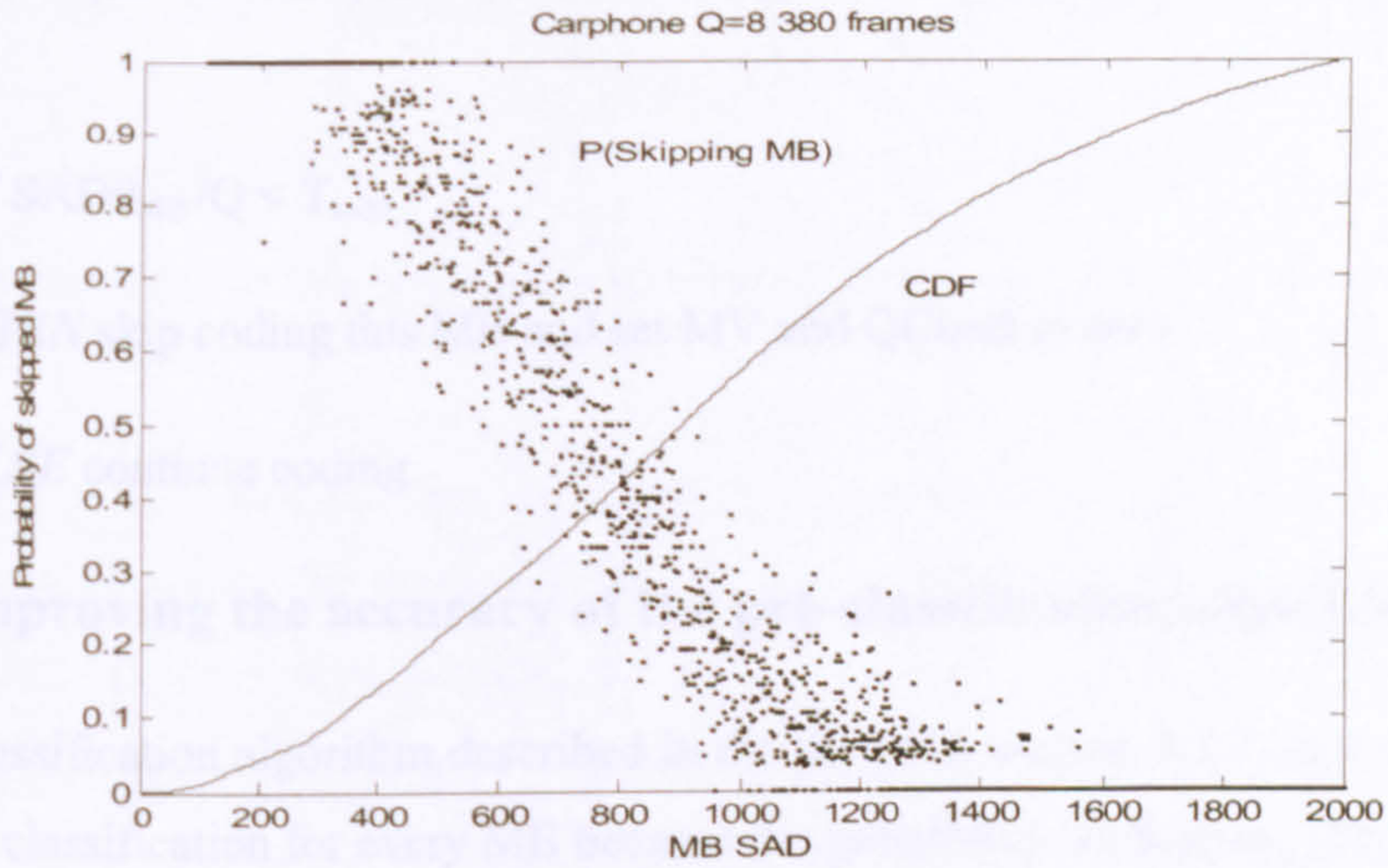


Figure 7-3 Probability of skipping MB with Q=8

Figure 7-4 Probability of skipping MB with normalized $SAD0_{MB}$

The relationship between $SAD0_{MB}$ and P(skip) also depends on the quantiser step size since a higher step size results in an increased proportion of skipped macroblocks (as shown in Figure 7-1). Figure 7-4 plots P(skip) against ($SAD0_{MB}$ / Q) for the "Carphone" sequence with various quantiser step sizes Q (Q=4,8,12,16). The plots of P(skip) in this Figure are smoothed with a moving average operator to facilitate comparison of the different sequences. Normalising $SAD0_{MB}$ with respect to Q produces a similar trend for each coded sequence. The pre-classification algorithm can therefore be changed to the following:

IF $SAD0_{MB}/Q < T_{sad0}$

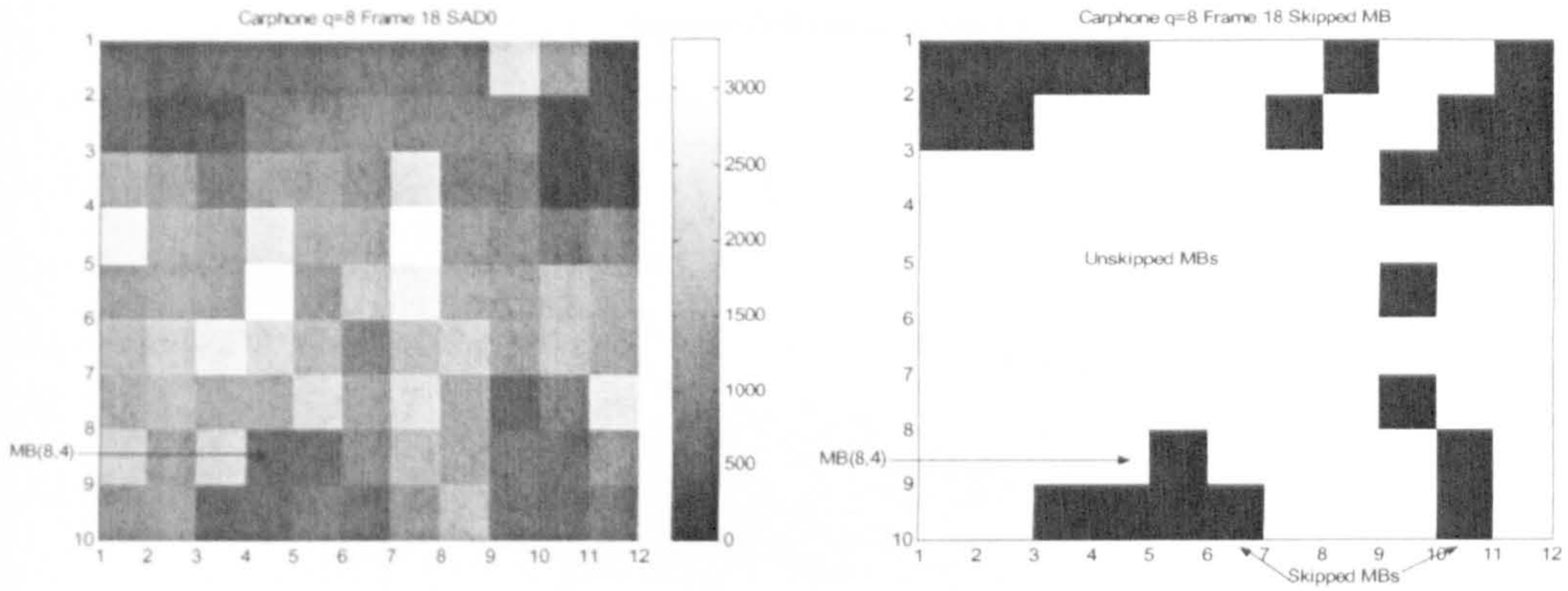
THEN skip coding this MB and set MV and QCoeff to zero

ELSE continue coding

7.3.3 Improving the accuracy of the pre-classification algorithm

The pre-classification algorithm described in the previous section 7.3.2 may not produce the correct classification for every MB because the probability of skipping MBs P(skip) is less than 1 in many cases. Whatever the value is chosen for Tsad0, there may be some MBs that have a value of $SAD0_{MB}$ below this threshold but should not be skipped. The pre-classification algorithm will fail in these cases, causing macroblocks to be erroneously skipped and increasing distortion in the decoded video sequence. Therefore, the aim of this section is to find a method to improve the accuracy of the pre-classification algorithm in order to decrease the distortion caused by incorrect prediction.

Figure 7-5 shows one example of a macroblock falling into this category. MB(8,4) in Frame 18 of "Carphone" has a low value of $SAD0_{MB}$ (535) but is not skipped as shown in Figure 7-5(b).

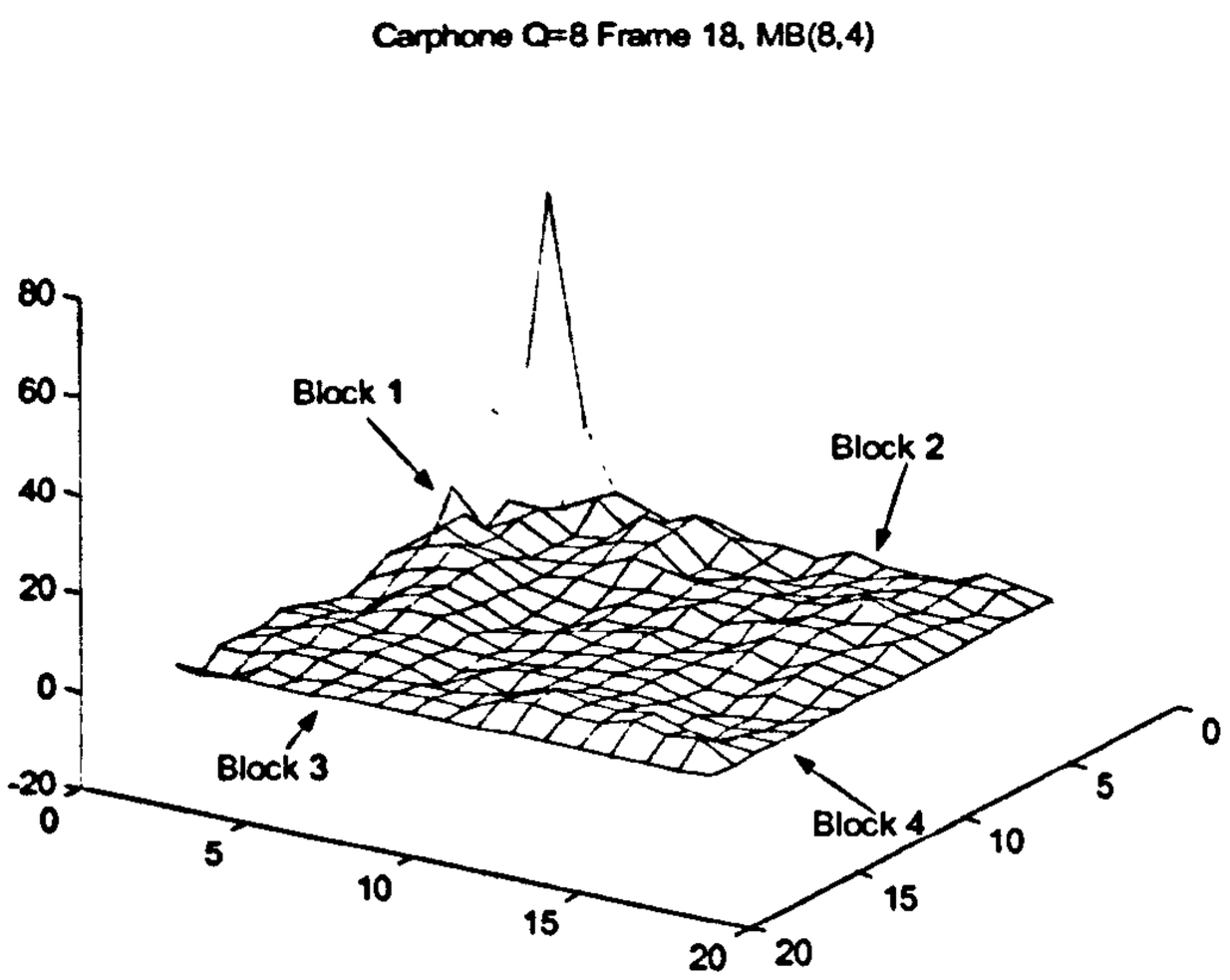


(a) $SAD0_{MB}$ **(b) Skipped MB**

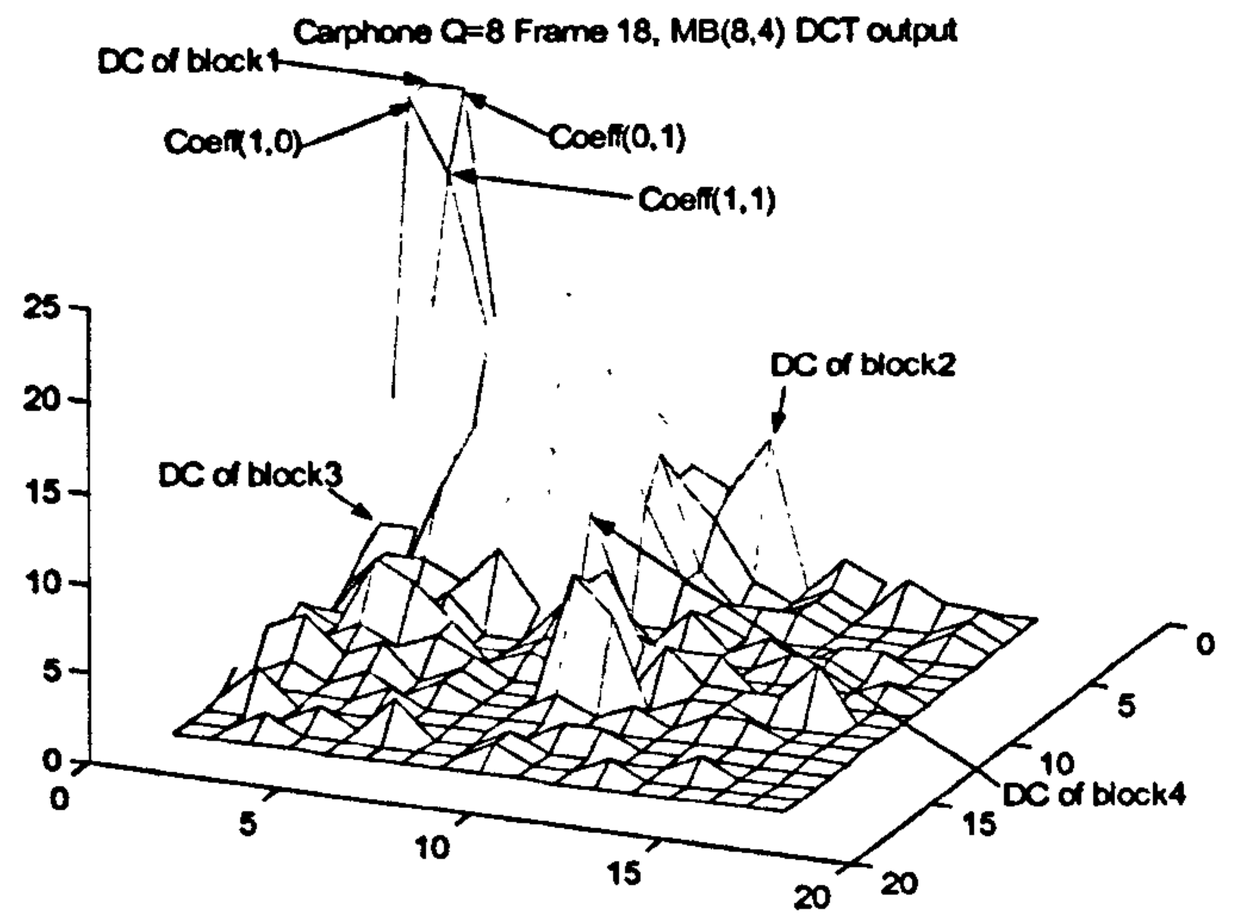
Figure 7-5 Example of low-$SAD0_{MB}$ value MB (Carphone, Q=8, Frame 18, M (8, 4))

Examination of MBs that fall into this category shows that these MBs often occur on the edges of moving areas. The residual (without motion compensation) typically contains a small number of high-valued samples, not enough to cause a significant increase in SAD but enough to produce some non-zero quantised coefficients.

Figure 7-6 shows the residual and DCT output for the luminance part of MB(8,4) in frame 18 of "Carphone", which has a low SAD value, but should not be skipped. From Figure 7-6 (a), it is clear that this is a region with a few high residual values, corresponding to a moving edge in the video sequence. Figure 7-6 (b) plots the magnitude of unquantised DCT coefficients of the four luminance blocks. It can be seen that the top-left block has a number of significant DCT coefficients. In particular, the four low-frequency DCT coefficients: the DC coefficient and the 3 lowest AC coefficients positioned in (0, 1), (1, 0) and (1, 1) have large magnitudes. The high magnitude of these low-frequency components is caused by the high value of a few samples in residual block. They are too significant to become zero after quantisation, which is the reason why this MB is not skipped.



(a) Residual value



(b) Absolute DCT output

Figure 7-6 Example of DCT process of (Carphone, Q=8, Frame 18, M (8, 4))

If the magnitudes of the DC and three low-frequency components can be approximated prior to encoding, the MBs (that have low $SAD0_{MB}$ but should not be skipped) may not be wrongly predicted as "Skipped" type so that accuracy of the pre-classification can be improved.

A high-magnitude DC coefficient tends to produce a correspondingly high $SAD0_{MB}$ (since the DC coefficient is proportional to the mean sample value of each block). Consequently, the DC coefficient can be approximately represented by $SAD0_{MB}$. However, the three lowest-frequency AC coefficient magnitudes are not reflected in the calculation of $SAD0_{MB}$. Therefore, a low-complexity method of estimating the magnitude of these coefficients (without actually carrying out the DCT) is proposed based on their basis pattern. The procedure for this method is illustrated in Figure 7-7.

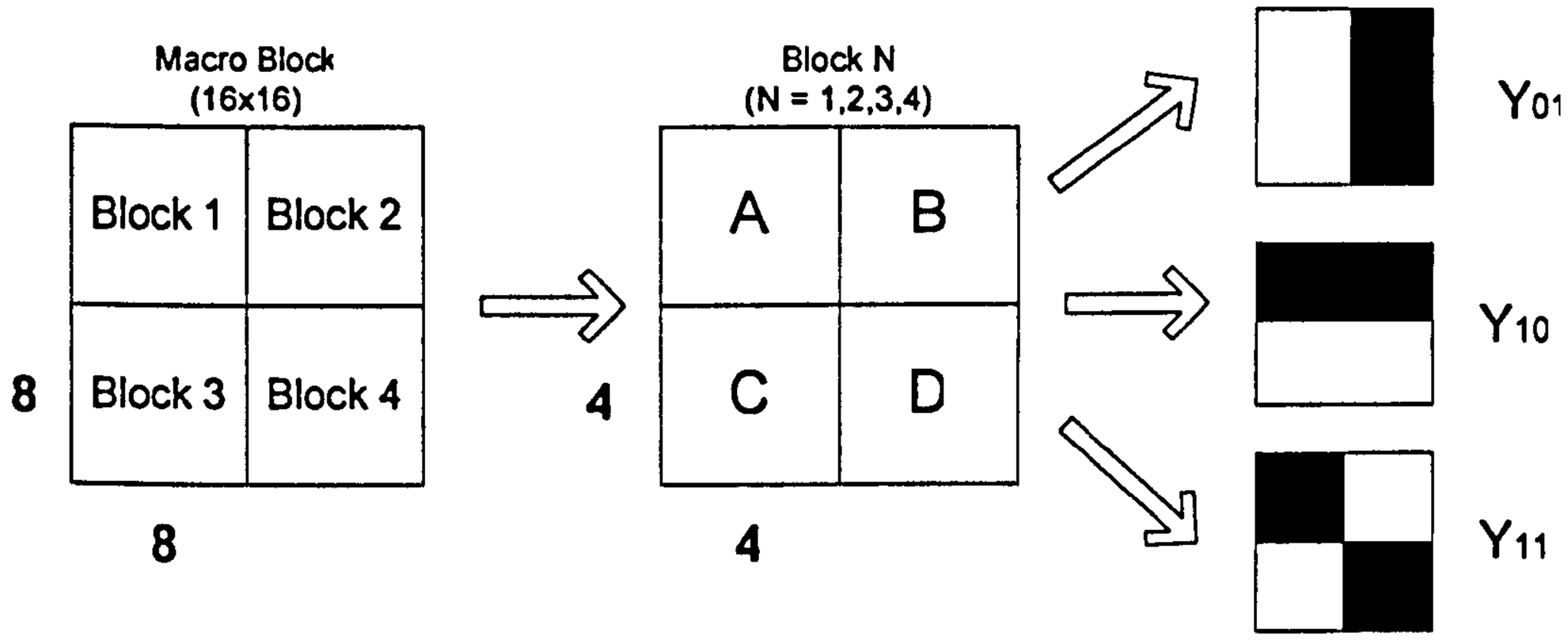


Figure 7-7 Predicting the three low frequency components

In a 16x16 macroblock, each 8x8 luminance block is divided into four 4×4 sub-blocks. A, B, C and D (shown in Equation 7-2) are the SAD values of each 4×4 block and R (i, j) are the residual pixel values without motion compensation.

$$A = \sum_{i=0}^{3} \sum_{j=0}^{3} |R(i,j)| \qquad B = \sum_{i=0}^{3} \sum_{j=4}^{7} |R(i,j)|$$

$$C = \sum_{i=4}^{7} \sum_{j=0}^{3} |R(i,j)| \qquad D = \sum_{i=4}^{7} \sum_{j=4}^{7} |R(i,j)|$$

Equation 7-2

Since the frequency represented by DCT coefficients (0,1) is changing horizontally from left to right, (A+C-B-D) (shown as Y_{01} in Equation 7-3) can be used to approximately evaluate the magnitude of the DCT coefficients (0,1). Similarly, a low-complexity estimate of the magnitudes of the other low frequency DCT coefficients coeff(1,0) and coeff(1,1) are provided by Y_{10} and Y_{11} respectively in Equation 7-3 based on their basis

patterns. If any of these coefficients have a large value, there is a high probability that the macroblock should not be skipped. Therefore $Y4\times4_{block}$ (Equation 7-4) is used to predict whether each block may be skipped. The maximum for the luminance part of a macroblock is calculated using Equation 7-5.

$$Y_{01} = abs(A + C - B - D)$$

$$Y_{10} = abs(A + B - C - D)$$

$$Y_{11} = abs(A + D - B - C)$$

Equation 7-3

$$Y4\times4_{block} = MAX(Y_{01}, Y_{10,} Y_{11})$$

Equation 7-4

$$Y4\times4_{\max} = MAX(Y4\times4_{block1}, Y4\times4_{block2}, Y4\times4_{block3}, Y4\times4_{block4})$$

Equation 7-5

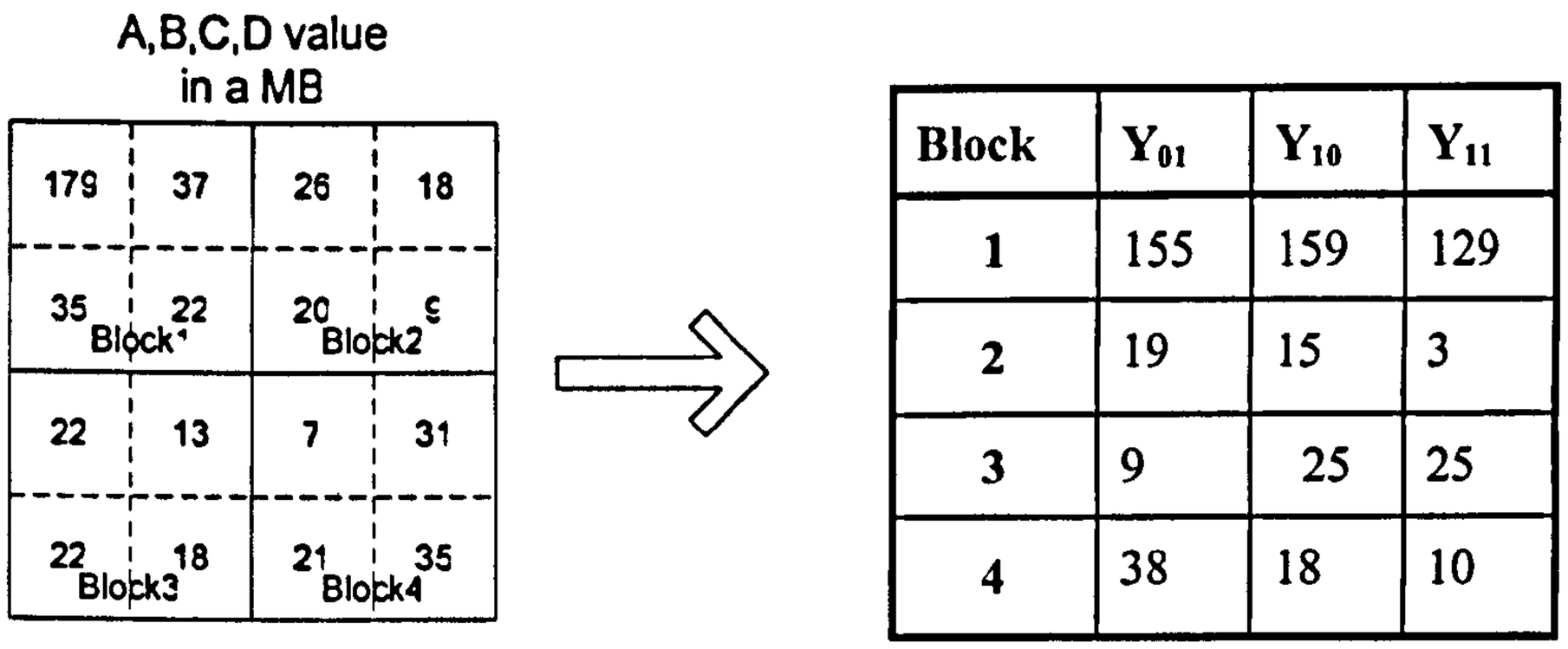
The calculated value of $Y4\times4_{max}$ is compared with a threshold T_{4by4} to improve the accuracy of skip prediction. The macroblock pre-classification algorithm becomes:

IF $SAD0_{MB}/Q < T_{sad0}$ and $Y4\times4_{max} < T_{4by4}$

THEN skip coding this MB and set MV and QCoeff to zero

ELSE continue coding

Figure 7-8 shows the results of calculating Y_{01}, Y_{10} and Y_{11} for MB(8,4) in frame 18 of "Carphone". Using Equation 7-3, Equation 7-4 and Equation 7-5, $Y4\times4_{max}$ of this MB is calculated to be 159. If it is greater than the threshold T_{4by4}, it will not be skipped during encoding.



Block	Y_{01}	Y_{10}	Y_{11}
1	155	159	129
2	19	15	3
3	9	25	25
4	38	18	10

Figure 7-8 Example of predicting the three low frequency components (Carphone, Q=8, Frame 18, MB(8, 4))

7.3.4 Choosing threshold T_{4by4}

The video sequences "Carphone", "Foreman" and "Mother and Daughter" were coded using the modified classification algorithm with various values of quantiser step size Q and T_{4by4}. For each fixed value of Q, a range of values of T_{4by4} were tested and the choice of T_{4by4} resulting in minimum distortion was recorded. The results of this experiment indicate that the optimal choice of T_{4by4} is approximately linearly related to Q (Equation 7-6).

$$T_{4by4} = 10Q + 70$$

Equation 7-6

Incorporating Equation 7-6 into the pre-classification algorithm gives the following:

IF $SAD0_{MB}/Q < T_{sad0}$ and $(Y4\times4_{max}-70)/Q < 10$

THEN skip coding this MB and set MV and QCoeff to zero

ELSE continue coding.

Applying this algorithm to MB(8,4) in frame 18 of "Carphone" sequence, where $Y4\times4_{max}$ = 159 and Q = 8:

$(Y4\times4_{max}-70)/Q = (159-70)/8 = 11.125$

This is greater than 10 and so the MB is not skipped, regardless of the value of $SAD0_{MB}$.

7.4 Experimental results

Several simulations were conducted to evaluate the performance of the macroblock classification algorithms. H.263+ TMN10 CODEC is modified with incorporating two pre-classification algorithms: (a) T_{sad0} only and (b) $T_{sad0}+ T_{4by4}$. The performances of the two algorithms are compared in terms of accuracy of prediction, decoded video quality, rate-distortion performance and computational complexity. The three test sequences "Carphone", "Mother and Daughter" and "Foreman" were used with a picture size of QCIF (176×144) and frame rate of 30 frames per second.

7.4.1 Performance of MB prediction

Each test sequence was coded using TMN10 encoder with classification algorithms (a) T_{sad0} only and (b) $T_{sad0}+ T_{4by4}$ with 280 frames and Q=8. Fixed thresholds T_{sad0} and T_{4by4} were chosen for the two algorithms to achieve the same target percentage of skipped MBs. In the experiments, the pre-determined targets for "Carphone" were set to 30% and 45% skipped MBs, and those for "Mother and Daughter" were chosen as 45% and 70%. Compared with the other two sequences, "Foreman" only has one target (30%) because it contains a lot of high motion and it is not possible to set a high target for skipped MBs without a significant reduction in decoded quality. For each sequence with a specified target and a pre-classification algorithm (either (a) T_{sad0} only or (b) $T_{sad0}+ T_{4by4}$), the total number of skipped MBs, the number of "missed" MBs (MBs that are skipped by the unmodified encoder but were not predicted as skipped MB by the proposed pre-classification algorithm) and the number of MBs wrongly skipped by the pre-classification algorithm are summarised in Table 7-1. The average luminance PSNR drop relative to TMN10 encoder with no complexity reduction, is also listed in Table 7-1.

In order to evaluate the performance of the proposed classification algorithms, 280 frames of each sequence were coded by TMN10 and MPEG-4 VM18 [47] (Simple Profile) with Q=8 and without any pre-classification algorithms. The percentage of

skipped MBs and average luminance PSNR of "Carphone", "Mother and daughter" and "Foreman" are shown in Table 7-2(a), (b) and (c) respectively.

From Table 7-2(a), 7712 MBs are skipped during encoding of "Carphone" using TMN10 (27.82% of total MBs). Using MPEG-4 VM18 with the same coding parameters, 7409 MBs (26.72% of total MBs) are skipped during encoding. MPEG-4 has less skipped MBs than TMN10, but it outperforms TMN10 by around 0.2 dB in terms of average PSNR. Similar results can be found for "Mother and Daughter" and "Foreman".

For each sequence with the same complexity target, it can be seen from Table 7-1 that the total number of MBs skipped by using (a) T_{sad0} alone is nearly the same as that by (b) T_{sad0}+ T_{4by4}, so that the complexity reduction performance of the two algorithms is comparable. Table 7-1 shows that the T_{sad0}+ T_{4by4} "misses" fewer MBs than T_{sad0} alone in each case and the number wrongly predicted skipped MBs is always lower for T_{sad0}+ T_{4by4}. Since the total number MBs skipped by both of the algorithms is the same, there are more skipped MBs correctly predicted by T_{sad0}+ T_{4by4} than by T_{sad0}. This indicates that T_{sad0}+ T_{4by4} provides a more accurate prediction than T_{sad0} only, resulting in better video quality, which is illustrated by the value of average luminance PSNR drop: T_{sad0}+ T_{4by4} has lower PSNR drop than T_{sad0} only. For the "Carphone" sequence with Q=8, the maximum percentage of skipped MBs by TMN10 is 27.82% (from Table 2(a)). Hence, when the skipped MB target is set to 45% for "Carphone", an extra 5224 MBs (number of wrongly predicted MBs) are skipped by T_{sad0}+ T_{4by4}. These extra skipped MBs cause a degradation of video quality. However, because the pre-classification algorithms select MBs with low residual energy, the loss of video quality is minimized.

Table 7-1 Performance of skipping MBs prediction using two classification algorithms

Video Sequence	Target Skipped MBs %	Classification algorithms	Total Skipped	Missed	Wrongly Predicted	Average Y PSNR drop (dB)
Carphone	30%	T_{sad0} only	8312	1585	2185	0.22
		T_{sad0}+ T_{4by4}	8329	1532	2149	0.19
	45%	T_{sad0} only	12413	523	5224	0.63
		T_{sad0}+ T_{4by4}	12389	431	5108	0.51
Mother and Daughter	45%	T_{sad0} only	8298	8209	222	0.14
		T_{sad0}+ T_{4by4}	8261	8226	202	0.08
	70%	T_{sad0} only	19411	1041	4167	1.05
		T_{sad0}+ T_{4by4}	19368	1011	4094	0.58
Foreman	30%	T_{sad0} only	8288	318	4998	0.64
		T_{sad0}+ T_{4by4}	8264	280	4936	0.41

Table 7-2 Number of MBs skipped by TMN10 and MPEG-4

(a) "Carphone" sequence

Carphone	TMN10	MPEG-4
Skipped MB	7712 (27.82%)	7409 (26.72%)
Average Y PSNR(dB)	34.10	34.32

(b) "Mother and Daughter" sequence

Mother and Daughter	TMN10	MPEG-4
Skipped MB	16285 (58.75%)	16288 (58.75%)
Average Y PSNR(dB)	34.95	35.28

(c) "Foreman" sequence

Foreman	TMN10	MPEG-4
Skipped MB	3608 (13%)	3347 (12.07%)
Average Y PSNR(dB)	33.37	33.65

7.4.2 Video Quality

7.4.2.1 Objective video quality

Table 7-1 demonstrates that T_{sad0}+ T_{4by4} achieves a lower PSNR drop than T_{sad0} only (with the same complexity reduction), therefore it can provide better video quality than T_{sad0}. Figure 7-9 shows the PSNR of each frame of "Carphone" with Q=8, 45% skipped MBs and fixed thresholds T_{sad0} and T_{4by4}. The thresholds were chosen to give a 45% mean reduction in computational complexity (i.e. 45% skipped macroblocks) in each case. It can be seen that T_{sad0} + T_{4by4} consistently achieves better video quality than T_{sad0} alone throughout the video sequences, for a comparable reduction in computational complexity.

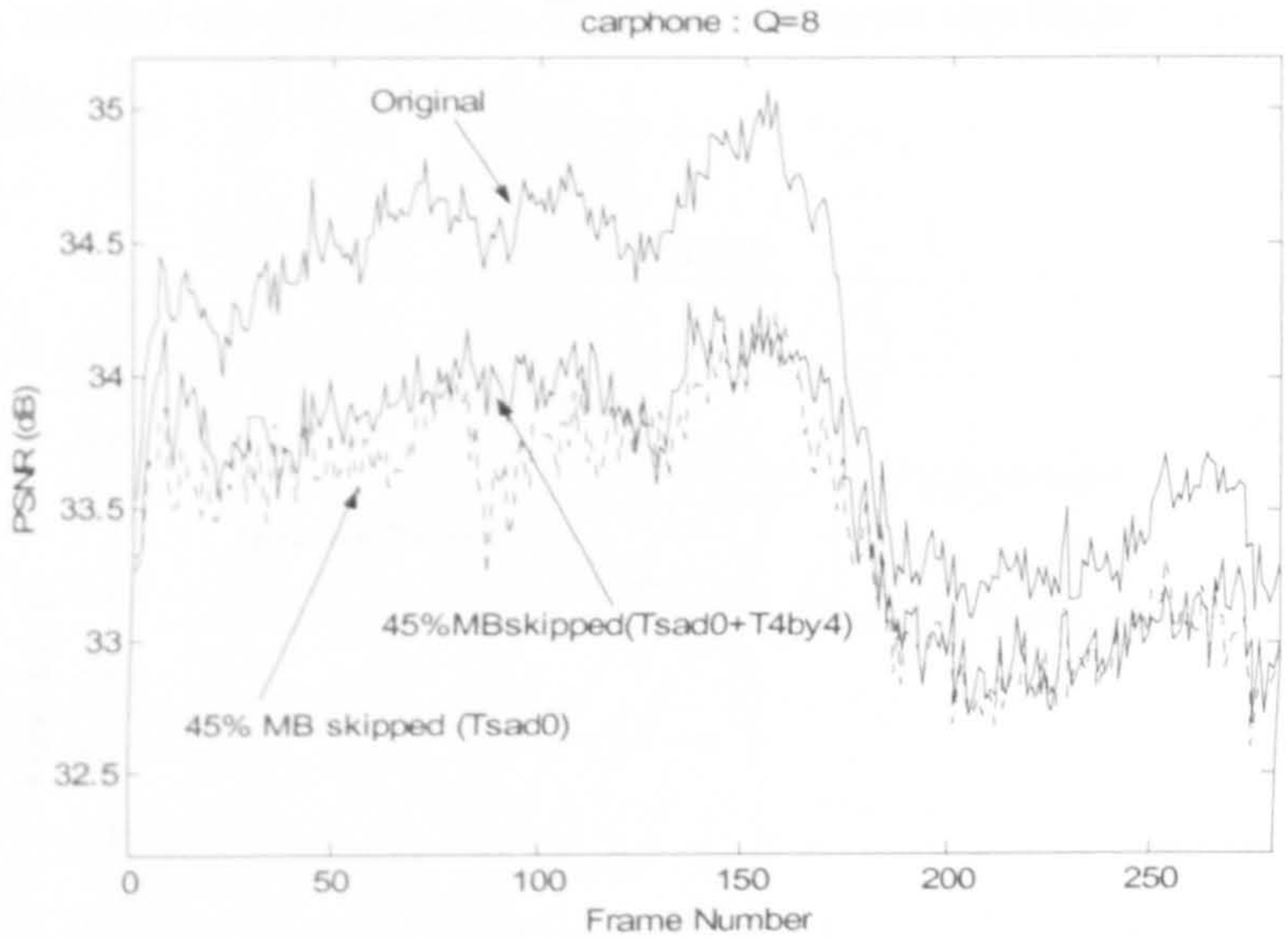


Figure 7-9 PSNR against frame number for "Carphone" with same complexity reduction by two types of classification algorithms

The "Carphone" video sequence was coded using two types of pre-classification algorithms. At each fixed quantiser, T_{sad0} was varied from low to high to parameterise the performance of the algorithm at a range of computational complexity values. Figure 7-10 plots the percentage of MBs classified as "skipped" by the proposed algorithm (x-axis) against the mean drop in luminance PSNR compared with the same sequence encoded without complexity reduction (y-axis). This drop in PSNR is caused by occasional incorrect skipping of MBs that should have been encoded. As T_{sad0} increases, the graph moves to the right and up: more MBs are skipped but the PSNR drop increases.

This Figure shows that classifying macroblocks using two thresholds, T_{sad0} + T_{4by4}, produces better video quality than T_{sad0} alone for the entire range of percentage of skipped MBs and various quantiser step sizes. For example, using only T_{sad0} to classify macroblocks, the algorithm skips 40% of macroblocks at the expense of a PSNR drop of 0.5dB (for a fixed quantiser step size Q=8). Using T_{sad0} + T_{4by4}, approximately 44% of macroblocks can be skipped without any further PSNR drop. When Q=16, skipping approximately 46% of macroblocks produces a PSNR drop of 0.5dB using only T_{sad0}; approximately 56% may be skipped for the same PSNR drop using T_{sad0}+ T_{4by4}. The

T_{sad0}+ T_{4by4} method out-performs the T_{sad0} method more significantly at higher quantiser step sizes (Q).

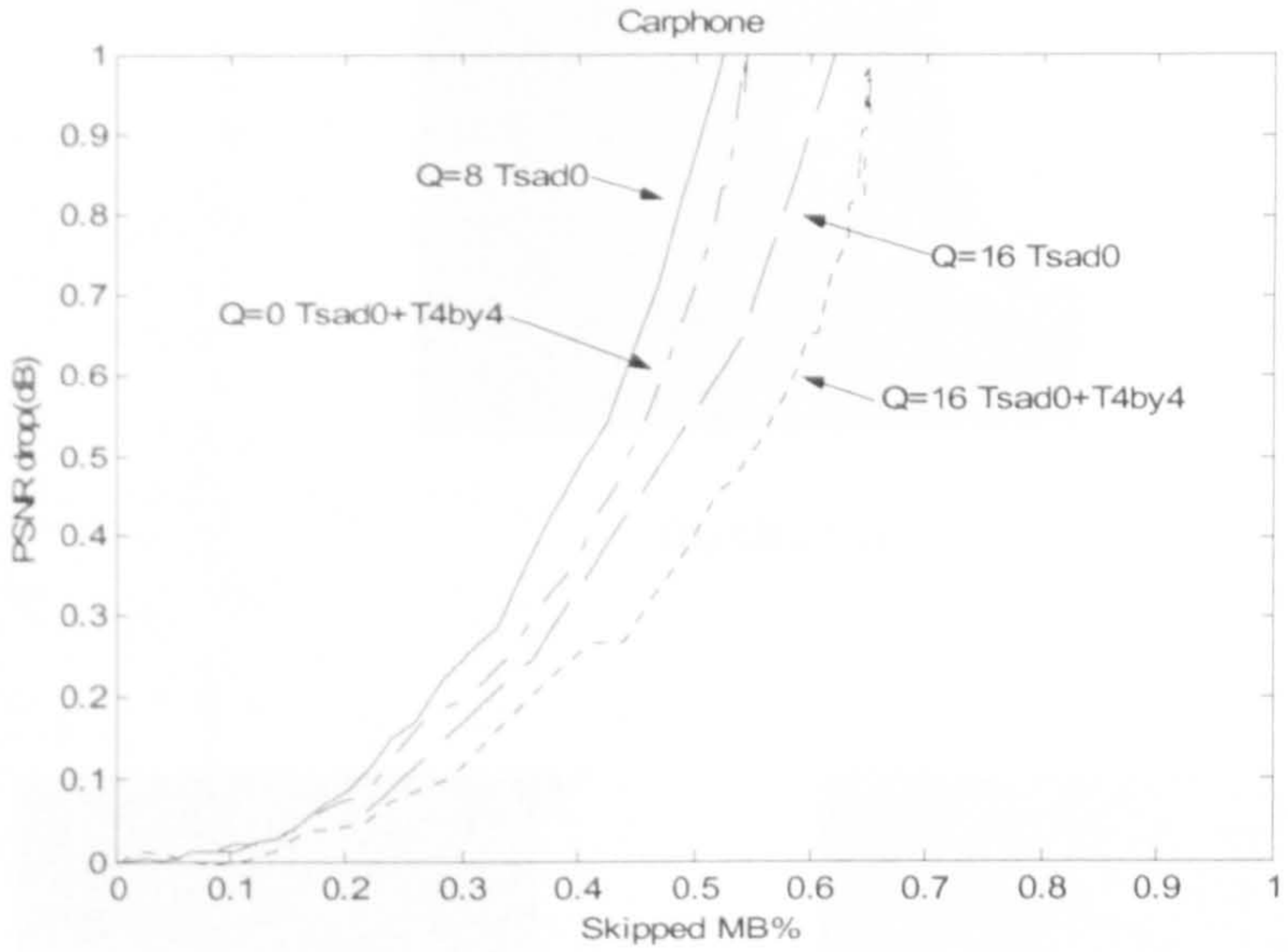
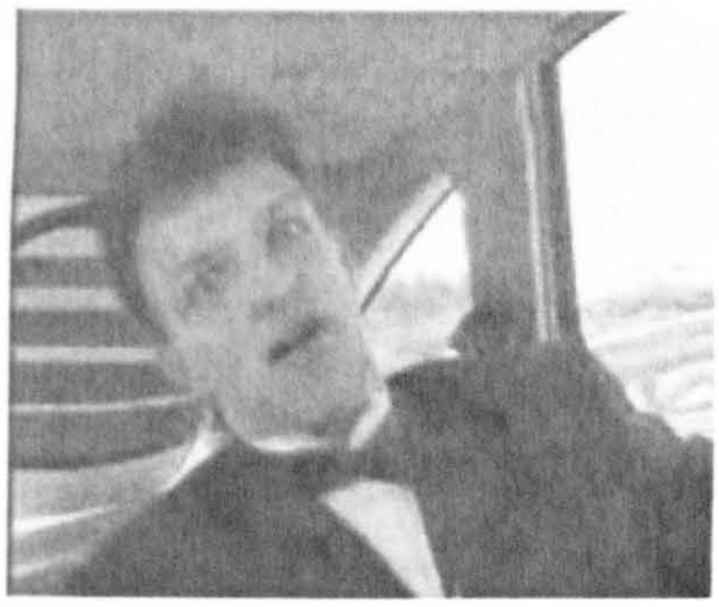


Figure 7-10 Performance comparison of various Q and two types of classification algorithms

7.4.2.2 Subjective video quality

Figure 7-11 compares frame 87 of the "Carphone" sequence after encoding and decoding using three methods: (a) without any complexity reduction, (b) after skipping 45% of the macroblocks using T_{sad0} only and (c) after skipping 45% of the macroblocks using T_{sad0}+ T_{4by4}. Figure 7-11(b) shows some visible degradation compared with Figure 7-11(a), with blocking artifacts around the top of the head introduced by the complexity reduction algorithm (note that these artifacts occur at the boundary of a moving object, as reported in Section 7.3.3). Figure 7-11 (c) is very similar to Figure 7-11 (a) with little apparent increase in distortion. When played back at 30 frames per second, there is a slight degradation in sequence (b) compared with (a) but no obvious difference between (c) and (a). An interesting feature of the (T_{sad0}+ T_{4by4}) algorithm is that MBs classified as "skipped" are typically in low-activity regions of the scene (such as the background of "Carphone") and incorrectly-skipped macroblocks in this region tend not to be noticed by the viewer. Hence the subjective quality is often better than the PSNR results might imply.

(a) Original



(b) With Tsad0 only

(c) Tsad0+ T4by4

Figure 7-11 Sample frame 87 of decoded "Carphone" sequence with Q=8 and 45% skipped MB

The "Carphone" and "Foreman" video sequences are encoded by TMN10 without and with the two pre-classification algorithms: (a) T_{sad0} only and (b) $T_{sad0}+ T_{4by4.}$ The quantiser step size is varied from 4 to 16 with an increment of 4. Fixed thresholds are chosen for T_{sad0} and T_{4by} to achieve a complexity reduction of 30% and 45% for "Carphone" and 30% for "Foreman". At each case, the coded video is decoded and tested on 15 viewers using the subjective video quality testing methods introduced in Section 3.4.1: Degradation Category Rating (DCR) and Pair Comparison (PC) methods.

During the DCR test, according to the grading levels listed in Table 3-1, viewers give their opinions about the degradation of each test sequence compared with original video

sequence (without compression). Score "5" means the degradation is imperceptible and the perceived video quality of the test sequence is exactly the same as the original. Score "1" indicates the worst visual quality, clear and very annoying degradation. The perceived video quality of sequences coded by classification algorithms is evaluated by DCR drop, which is defined as the difference between its average degradation score and that of the sequence coded with full complexity (no classification algorithm applied) at each quantiser.

Figure 7-12 and Figure 7-13 plot DCR drop of "Carphone" with 30% and 45% complexity reduction respectively. In each figure, the "triangle" marks show the actual value of DCR drop of the sequence coded by T_{sad0} only and the "square" marks show that of the sequence coded by T_{sad0}+ T_{4by4}. It is clear that there is a big variation of DCR drop between different quantiser step sizes of T_{sad0} only. This is because the subjective opinion of viewers is varied and the total number of viewers is limited. In order to compare the performance of the proposed two algorithms, linear trend lines are also plotted in the same figure, indicating the trend implied by these marks.

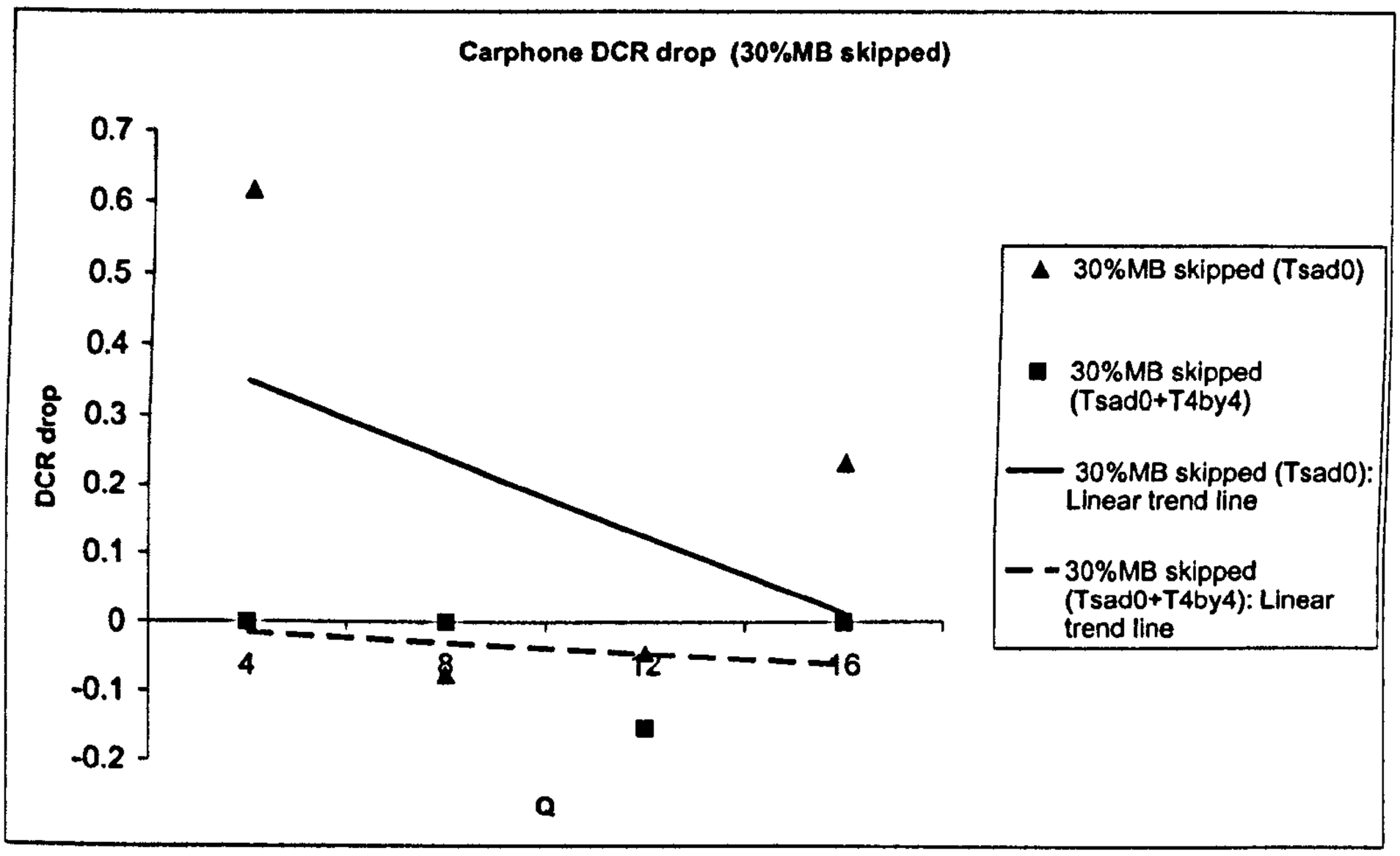


Figure 7-12 DCR drop for "Carphone" with 30% MB skipped

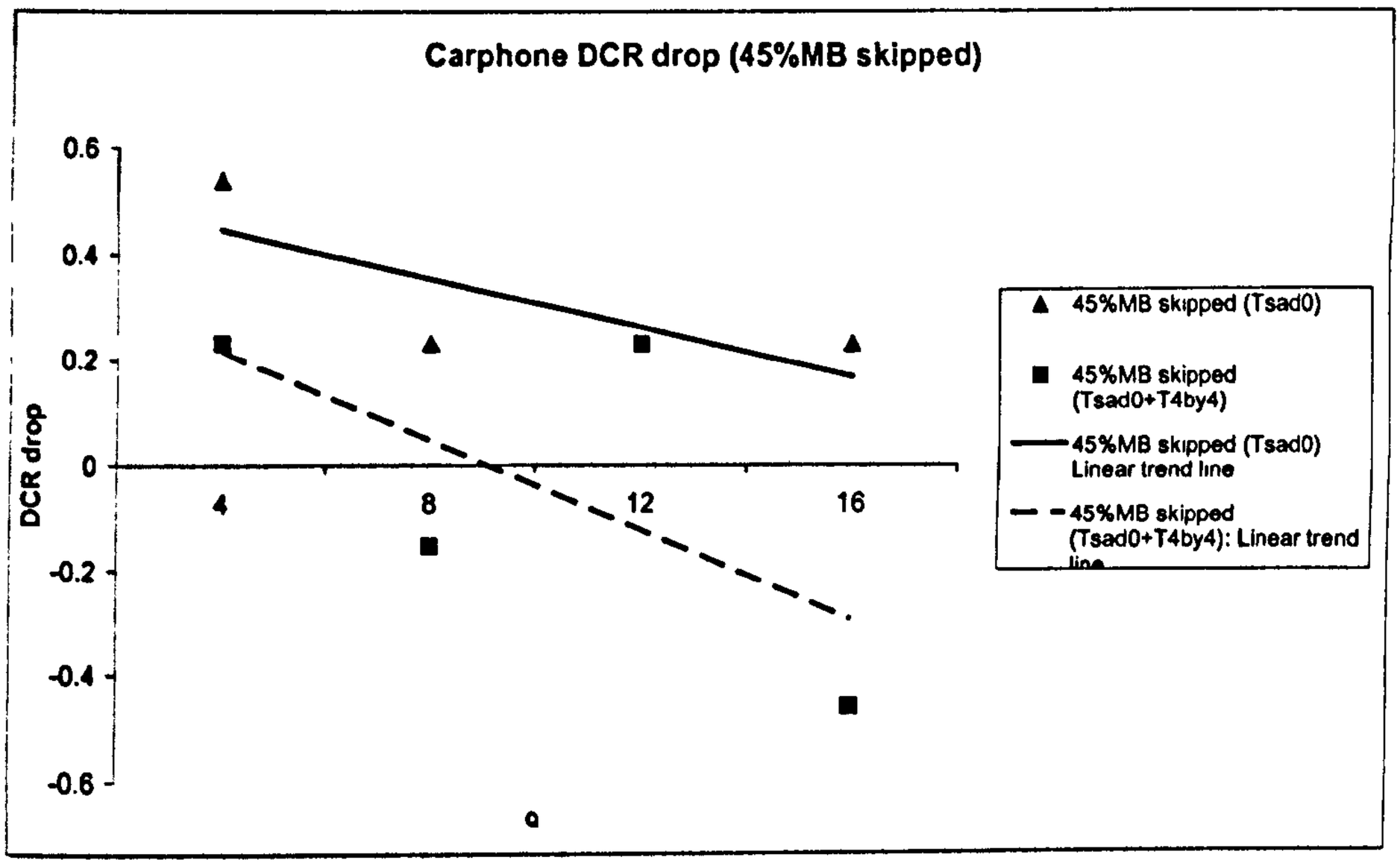


Figure 7-13 DCR drop for "Carphone" with 45% MB skipped

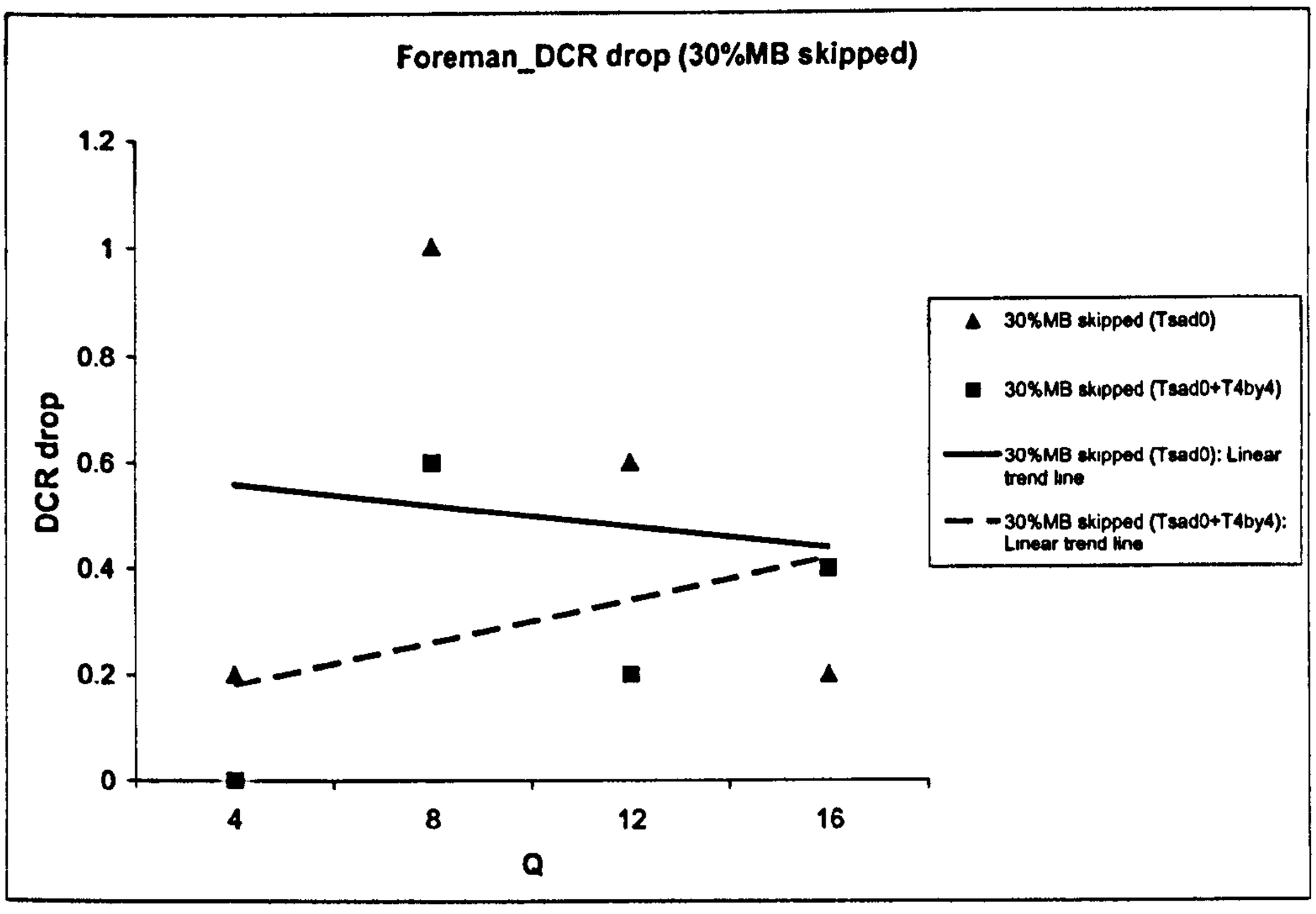


Figure 7-14 DCR drop for "Foreman" with 30% MB skipped

It is clear in Figure 7-12 that the DCR drop is around zero for Tsad0 + T4by4 , signifying no perceived visual quality loss compared with the normal coding without any

complexity-reduction algorithm. Both Figure 7-12 and Figure 7-13 illustrates that DCR drop decreases with increasing quantiser and becomes negative for Tsad0+ T4by4 with large quantiser in Figure 7-13, e.g. quantiser =16. A negative DCR drop means that the video sequence with complexity reduction has better perceived video quality than that with full complexity at the same quantiser. For both 30% and 45% complexity reduction, Tsad0+ T4by4 provides a consistently lower DCR drop than Tsad0 only along the entire range of Q. Similar results can be found in Figure 7-14 for "Foreman" with 30% skipped MBs: Tsad0+ T4by4 outperforms Tsad0 only in terms of perceived visual quality. The results shown in the above figures are the average DCR drop of all test subjects. There is a wide variation in the recorded results due to the difference in subjective opinion of the viewers. The variation can be reduced by increasing the number of viewers.

During the PC test, video sequences coded by classification algorithms are displayed together with the same video coded without complexity reduction. Viewers choose the one they prefer in terms of perceived video quality. The proportion of people, who think the reduced-complexity video has better video quality than normal video with "full" complexity (P), is recorded. Individual P points and linear trend of P against Q are plotted in Figure 7-15 for "Carphone" with 30% complexity reduction, Figure 7-16 for "Carphone" with 45% complexity reduction and Figure 7-17 for "Foreman" with 30% complexity reduction. P=50% means the subjective video quality of reduced-complexity video is the same as that of "full" complexity video, which is plotted as a dotted line in each figure.

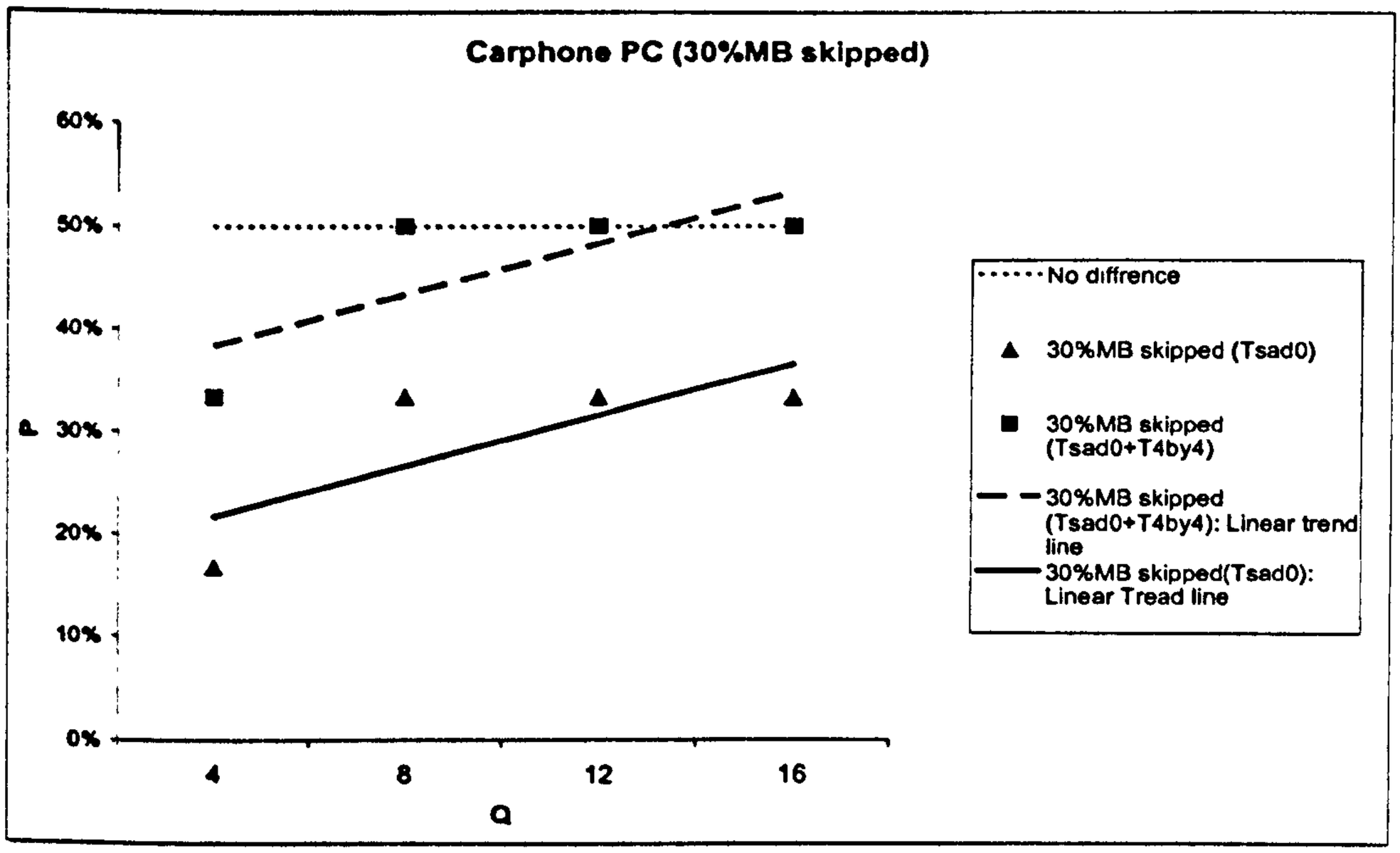


Figure 7-15 PC results for "Carphone" with 30% MB skipped

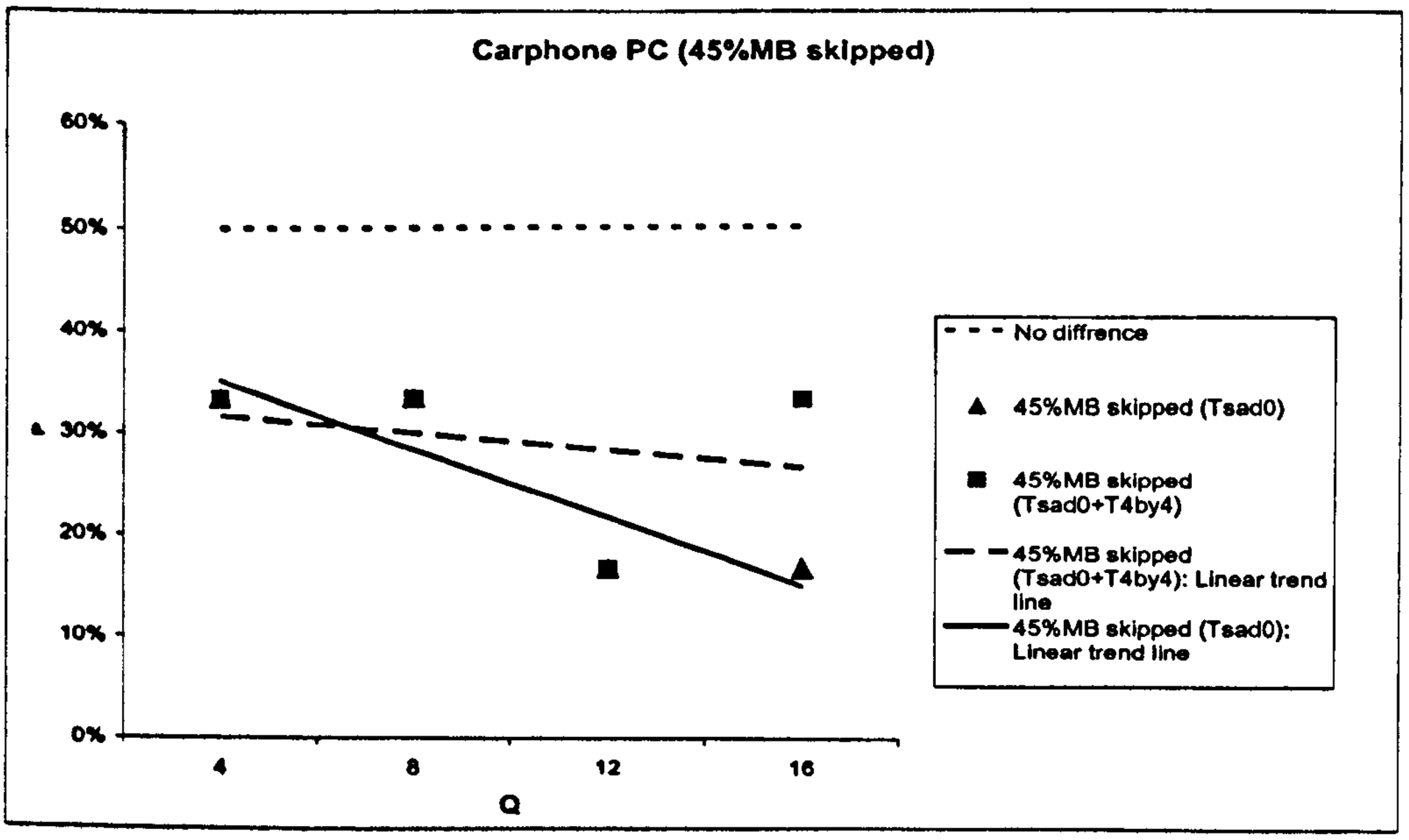


Figure 7-16 PC results for "Carphone" with 45% MB skipped

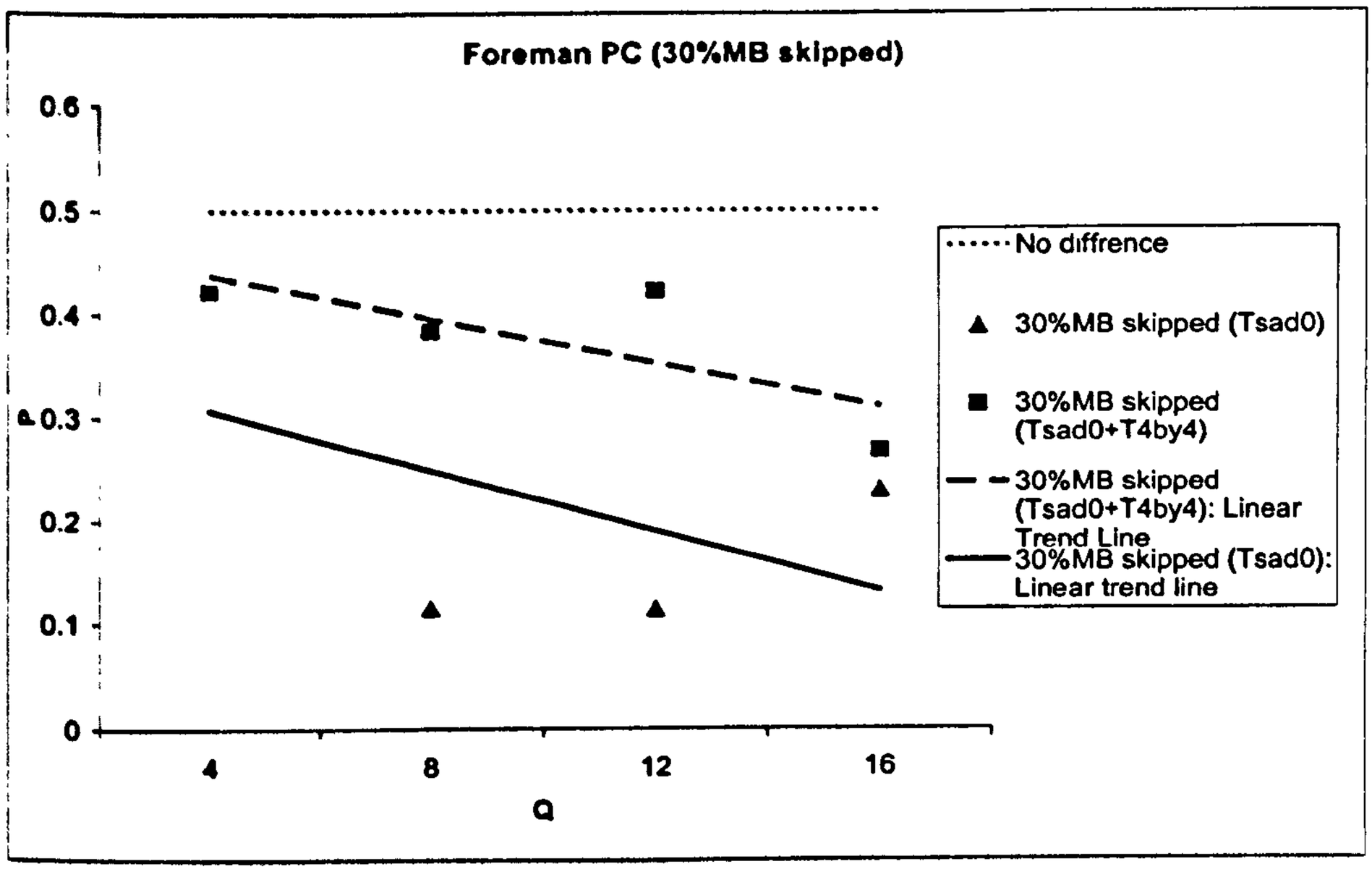


Figure 7-17 PC result for "Foreman" with 30% MB skipped

In Figure 7-17, it can be seen that nearly 50% of subjects prefer the video sequences with 30% complexity reduction by using T_{sad0}+ T_{4by4}, whereas around 30% of subjects preferred the sequence with 30% complexity reduction using only T_{sad0}. For 45% complexity reduction, Figure 7-16 shows that T_{sad0}+ T_{4by4} achieves higher P than T_{sad0} at large values of Q. Figure 7-17 demonstrates that T_{sad0}+ T_{4by4} outperforms T_{sad0} with an increase nearly 15% in P.

7.4.3 Rate-distortion performance

Figure 7-18 plots the rate-distortion performance of the "Carphone" sequence after encoding and decoding using the H.263 TMN-10 reference model operating in Baseline mode ("Original"). Two methods (T_{sad0} only and T_{sad0}+ T_{4by4}) are used to achieve two target complexity reduction levels (30% skipped macroblocks and 45% skipped macroblocks) and the corresponding rate-distortion curves are plotted in Figure 7-18. The (T_{sad0}+ T_{4by4}) algorithm outperforms the T_{sad0} algorithm at both complexity targets through the entire range of bit rate. Using the (T_{sad0}+ T_{4by4}) method the computational complexity can be reduced by 30% with a very slight degradation in rate-distortion performance.

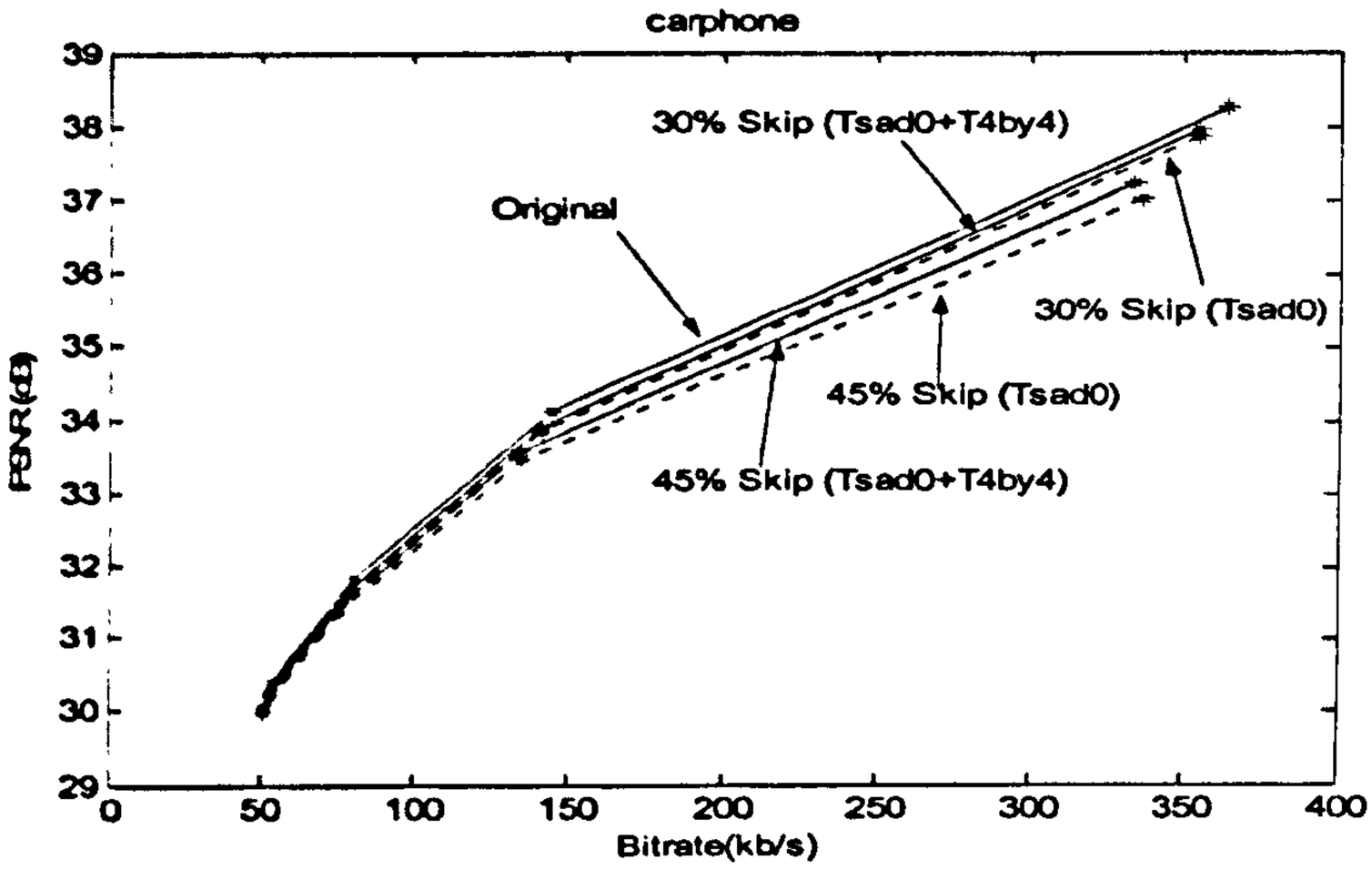


Figure 7-18 Rate-distortion performance of "Carphone"

7.4.4 Computational complexity

The purpose of the pre-classification algorithm is to reduce computational complexity during encoding and so it is important to identify any additional computational costs added by the proposed algorithm. $SAD0_{MB}$ is normally computed in the first step of any motion estimation algorithm and so there is no extra calculation required. Furthermore, the SAD values of each 4x4 block (A, B, C and D in Equation 2) may be calculated without penalty if $SAD0_{MB}$ is calculated by adding together the values of SAD for each 4x4-sample sub-block in the macroblock (Equation 7-7).

$$SAD0_{MB} = \sum_{block=1}^{4}(A+B+C+D)$$

Equation 7-7

The additional computational requirements of the classification algorithm are the operations in Equation 7-3, Equation 7-4 and Equation 7-5. For each MB consisting of four 8×8 luma blocks, there are 12 (=4×3) additions, 24 (4×6) subtractions and 12 comparisons. This is negligible compared with the computational cost of motion

estimation, DCT, etc. For each macroblock, therefore, the computational saving of our proposed algorithm significantly outweighs the small computational cost of classifying macroblocks.

7.5 Summary

This chapter describes a macroblock pre-classification algorithm that has the potential to significantly reduce computational complexity in a software video encoder. The goal of this algorithm is to avoid expensive processing of macroblocks that are destined to be skipped during encoding, whilst minimising distortion due to incorrect classification predictions. The initial SAD calculation $SAD0_{MB}$ can give an approximate prediction of macroblock type (as reported in [87]) but incorrectly predicts non-skipped macroblocks that contain significant low frequency AC coefficients after quantisation. Adding a prediction of low-frequency residual energy improves the accuracy of macroblock classification, particularly for macroblocks at the edge of moving regions.

Experimental results show that the proposed classification algorithm can deliver substantial computational savings (40-50% for a high activity sequence such as "Carphone", more for low activity sequences) with only a small reduction in rate-distortion performance. The reduction in subjective quality is actually smaller than the PSNR results imply, since the algorithm tends to classify skipped macroblocks in static, visually unimportant regions of the scene. The classification process is computationally simple and most of the operations may be integrated into the normal motion estimation process.

The classification algorithm has been applied to H.263+ but it is equally applicable to MPEG-4 Simple Profile and other macroblock-based encoders.

Chapter 8 Discussion

8.1 Introduction

This chapter summarizes the main developments and experimental results of Chapter 5, Chapter 6 and Chapter 7. The advantages and disadvantages of these proposed techniques are evaluated by analysing the experimental results and comparing with other approaches in the literature. Possible directions for further research are also indicated.

8.2 Summary of developments and experimental results

Computational complexity limits the coding performance of software-only video encoders. This work has proposed several approaches to reduce and manage complexity during the encoding process.

8.2.1 DCT complexity analysis

An adaptive algorithm used to control the computation cost of DCT and related functions is introduced in Chapter 5. This algorithm exploits the correlation between all-zero coefficients and the SAD value of a single block (SAD_B) by means of statistical analysis. Based on the relationship found by experiments, SAD_B approximately predicts the occurrence of all-zero coefficient blocks and a lower value of SAD_B indicates higher possibility of a block containing all-zero coefficients. It is obvious that SAD_B can not provide a 100% accurate estimation. However, since SAD_B represents the average energy of a block, the distortion caused by incorrect prediction is minimized. The computational complexity of DCT and related functions are reduced by comparing the SAD_B of every block with a pre-set threshold and not carrying out further processing of blocks whose SAD_B is less than this threshold.

The use of a constant threshold in the entire video sequence results in variation of complexity, and consequently variable processor utilization, which does not satisfy the objective of near-constant complexity described in Section 1.2. Through examining the relationship between complexity(C), thresholds (T_0) and quantiser step size (Q), it is found that complexity is almost linear in relation to $1/QT_0$. This approximate correlation makes it possible to choose the threshold for each frame which can achieve a target

complexity. The objective of DCT complexity control (near-constant complexity throughout the video sequence) is accomplished by adding a feed-back loop into the complexity-reduction algorithm, which updates the threshold for each frame based on previous complexity information.

The mechanism of this adaptive algorithm is based on two hypotheses (1) the correlations between complexity, threshold and quantiser and (2) successive motion-compensated frames have a similar distribution of coefficients in each block. Section 5.3.2 describes the investigation and Figure 5-6 demonstrates the correlation between these three parameters. It is well known that neighbouring frames have strong similarities (i.e. temporal redundancy); accordingly, after motion estimation and compensation, two consecutive motion-compensated frames are similar in terms of energy distribution.

The performance of the proposed algorithm is evaluated by experiments in Section 5.5 in terms of video quality, complexity stability and convergence time. The results demonstrate that the adaptive algorithm can reduce computations of DCT and related functions by 50-70% with negligible loss of video quality. When the complexity reduction increases to 90%, the video quality loss is still small for low or moderate activity video sequences. This is because they naturally contain more all-zero coefficients than high motion sequences. It is also shown that the complexity can converge quickly to a target and may be controlled at this level without significant oscillation. Profiling results demonstrate the ability of the proposed algorithm to control coding time. The extra calculations added by this algorithm have almost no effect on the total complexity.

Experimental results have demonstrated that the objectives for this part of the work described in Section 1.2 have been achieved. Compared with previous approaches to reduce the complexity of DCT function in literature, especially Sun's work [75], the innovation and advantages of this adaptive algorithm are summarized as:

- SAD_B is used to predict the all-zero coefficients block, which is more accurate than the SAD value of a macroblock.
- It is proposed not only to decrease computations but also to maintain near-constant complexity throughout the video sequence.

- A feed back loop is employed, which updates a control threshold dynamically to achieve the target complexity regardless of initial condition or sequence statistics.

However, this algorithm does not include any mechanism to stop reducing complexity in circumstances when the obtained video quality is very bad. For instance, if a low complexity target (such as 0.1) is set for a very high-motion video sequence, the adaptive algorithm will skip DCT and related functions for most of the blocks in order to achieve the target, causing a serious loss in quality. The 'best' factor k in the algorithm is chosen based on experimental results and it may be appropriate to change it dynamically, based on the characteristics of the input video. Suggestions for possible improvements and extension of this work are described in Section 8.3.

8.2.2 Motion estimation complexity analysis

Chapter 6 describes management of computational complexity of motion estimation function based on nearest neighbour search (NNS). As described in Section 1.2, the objective of this development is to achieve constant complexity of motion estimation throughout the video sequence, which is fulfilled by the proposed adaptive algorithm. Through examining the search pattern and the termination rules of NNS, it is found that the complexity of motion search can be reduced by controlling the maximum number of layers that are searched. The near linear correlations of complexity and number of search layers found in the simulations demonstrate the possibility of adjusting the maximum number of search layers for the current frame in order to obtain a target complexity. The adaptive algorithm for motion estimation is built on two foundations, which are similar to that of the adaptive algorithm for DCT, (1) the linear correlation between complexity and the number of search layers and (2) neighbouring frames have strong similarities.

The experimental results in Section 6.5 demonstrate that the adaptive algorithm for motion estimation achieves good performance in decreasing and controlling computational complexity. However, the use of this algorithm causes degradation in rate distortion performance that compensates for the reduction in computational complexity.

Conventional methods of decreasing the computation cost of motion search typically reduce the number of candidates involved in the search. In comparison with conventional methods, the approach investigated in Chapter 6 has the following innovative aspects:

- It achieves near-constant complexity of motion estimation by means of dynamically updating thresholds.

- The control of the complexity is largely independent of the input video signal.

A disadvantage of this approach is that it cannot always achieve exactly the pre-determined target because the number of layers used in changing the threshold is limited and it can not perform very fine adjustment. This approach is designed for reducing and controlling the complexity of nearest neighbour search and it cannot be directly applied to other fast motion search algorithms, but the basic concept and mechanism of this approach may be applicable to other fast search algorithms for block-based motion estimation.

8.2.3 Macroblock classification analysis

Chapter 7 introduces a pre-classification algorithm used to manage the computational complexity of the entire encoding process including motion estimation and compensation, DCT and related functions as well as variable length coding. Certain macroblocks contain all-zero quantised DCT coefficients and zero motion vectors and these macroblocks are categorized as skipped MB by a pre-classification algorithm. Encoding these skipped MBs wastes computational resources as they are skipped after processing. Significant computational can be saved by not coding these skipped MBs, so the main task of this pre-classification algorithm is to identify skipped MBs correctly prior to encoding.

The first approach presented in Chapter 7 predicts skipped MBs by using $SAD0_{MB}$ (sum of absolute difference of the luminance part of a MB without motion estimation). Since $SAD0_{MB}$ is the value of the error measurement for zero displacement in motion search and represents the average energy of a macroblock, low $SAD0_{MB}$ value indicates a high possibility of both zero motion and all-zero DCT coefficients after quantisation. Therefore, a MB is classified as a skipped MB if its $SAD0_{MB}$ value is less than a pre-determined threshold. $SAD0_{MB}$ provides a good prediction of skipped MB, but it is not 100% accurate and there are always some MBs with low $SAD0_{MB}$ values which are not skipped naturally. Through examining the characteristics of those MBs, it is found that they often fall into the area on the edge of the movement in a picture. They typically have

a few high value coefficients, which are not enough to significantly increase the value of $SAD0_{MB}$, but can produce non-zero coefficients after quantisation.

A new method is added to the pre-classification algorithm to improve the accuracy of the prediction. It identifies those unskipped MBs with a low $SAD0_{MB}$ value by estimating the low-frequency components. When the estimated low-frequency value exceeds a certain level, this MB will not be skipped even if its $SAD0_{MB}$ is less than the threshold. This method improves the accuracy of the pre-classification algorithm and the small additional computation can be ignored in comparison with a significant computational saving.

Experimental results in Section 7.4 demonstrate that prediction accuracy is improved by the new method of estimating the low-frequency components of the DCT, resulting in an increase in decoded video quality and rate-distortion performance compared with the basic method of $SAD0_{MB}$ only. The proposed pre-classification algorithm can deliver a 30-45% complexity reduction with small degradation in objective video quality. The subjective video quality is actually higher than implied by PSNR because the algorithm skips the macroblocks in unimportant areas, such as the background.

Methods of complexity reduction described in the literature are performed on single or related operations during encoding, so that the maximum complexity saving is limited. This pre-classification algorithm has the following advantages:

- This algorithm is able to decrease the computation burden for each macroblock in terms of the entire coding process, so it can obtain more significant complexity saving than other methods.

- This algorithm minimizes the incorrect prediction of skipped macroblocks by means of estimating low-frequency DCT coefficients.

A fixed threshold is utilized to obtain the complexity target and it may be possible to achieve better results by changing it adaptively. Furthermore, further work is needed before this algorithm can be applied to practical real-time video communications. Section 8.3 gives suggestions for future research.

8.3 Suggestions for future work

The techniques presented in this thesis could be extended in several ways. This section lists some possible topics for further development, some of which are directly linked with the proposed complexity-reduction algorithms; others indicate general directions for the longer term.

1) The adaptive algorithm for DCT and related functions updates the thresholds for the current frame based on one previous frame and it may be possible to achieve better stability by examining several preceding frames. The 'best' factor k is selected according to experimental results; in order to obtain the best control for various sequences with different levels of coding parameters, it may be necessary to change it dynamically, based on the characteristics of the video sequence and the current coding situation.

2) The macroblock pre-classification algorithm employs a fixed threshold to achieve complexity management. A suitable approach for extending this work might be to include a control algorithm such as the feedback-based computational complexity control mechanism described in Chapter 5 to achieve constant and stable complexity.

3) The macroblock pre-classification algorithm reduces complexity by skipping MBs with all-zero DCT coefficients and zero MV. There are still two types of macroblocks that may benefit from complexity reduction: "QCoeff=0" and "MV=0". Further work might be appropriate to integrate the pre-classification algorithm with computational management of "QCoeff=0" macroblocks (e.g. the DCT block classification algorithm described in Chapter 5) and "MV=0" macroblocks.

4) Computational management affects the rate-distortion performance of a video CODEC and so it could be useful to take bit rate into account and examine the joint control of rate, complexity and distortion [39].

5) The Advanced Video Coding standard (MPEG-4 part 10/H.264) has a significant performance improvement over previous video coding standards due to the

employment of new techniques, such as multiple references frames and variable-size blocks. However, it is also these new techniques that dramatically increase the complexity of the CODEC and make it difficult to implement in real-time. Researchers have already started to work on this problem and to date mainly concentrated on speeding up the motion estimation [91-93]. The complexity control techniques developed in this thesis are based on block-based video coding and focus on the main coding functions. It may be possible to apply these algorithms to the H.264 encoder and investigate new approaches in order to achieve adaptive control of the complexity of H.264.

Chapter 9 Conclusions

This thesis addresses the computational complexity problem of software video encoders. The performance of software CODECs is constrained by several parameters including computational complexity and transmission bandwidth. In the scenarios of real-time multimedia and mobile video, computational complexity becomes a key constraint. The obtained video quality of a coded sequence depends on the available bitrate and computational capacity. The aim of this work is to limit the computational burden as well as to adaptively control complexity in order to provide optimal video quality.

The allocation of computation resources to each function during the encoding process is examined and profiling test results show that motion estimation, DCT and related functions (including quantisation, IDCT, and inverse quantisation) are typically the most computationally complex functions and therefore become candidates for complexity control.

An adaptive complexity control algorithm for DCT and related functions is described in Chapter 5, which can reduce computation to a target level and keep it at this level throughout a video sequence. Constant complexity performance can be obtained at the expense of negligible or little loss of video quality as compared with full DCT complexity. The main innovation of this approach is that (1) it provides adaptive control of computational complexity rather than a simple reduction in computation and (2) the complexity control is largely independent of the input video sequence.

Most existing "fast" motion estimation algorithms reduce computation by decreasing the number of candidates for motion search. In many cases the complexity reduction depends on the content of video sequences. In Chapter 6, a variable-complexity algorithm is described for managing motion estimation complexity based on the Nearest Neighbour Search algorithm. This algorithm has the following advantages: (1) it is capable of maintaining the complexity at a near-constant target level regardless of the input video signal, (2) it achieves small loss in rate-distortion performance.

Coded macroblocks are classified into "skipped" and "unskipped" types based on the value of quantised DCT coefficients and motion vectors. In Chapter 7, two macroblock

pre-classification algorithms have been investigated to predict the skipped MBs prior to encoding, with the potential to save significant computational effort by not coding these MBs. The accuracy of the prediction of the first proposed algorithm is improved by estimating the magnitude of low-frequency DCT coefficients. The most important advantage of this technique is that the achieved computational complexity reduction applies to the entire encoding process for a macroblock rather than single functions or groups of functions.

A reduction in computational complexity of video encoders affects decoded video quality. The proposed algorithms achieve the minimal quality distortion under a fixed encoder complexity constraint and provide controllable trade-offs between complexity and video quality. Computation resources are efficiently allocated among different frames and components of a video sequence.

The techniques described in this thesis fulfil the aim and objectives of this research work. They are practical and can be applied to video coding applications where encoding time is an important factor, such as real-time multimedia systems, to control computational complexity whilst maintaining acceptable video quality. These algorithms can also benefit power-constrained systems, for example mobile video phones, by reducing the power consumption for encoding video, hence potentially achieving longer battery life.

References

[1] M Kotadia, "3G subscriptions poor: U.K. report," http://asia.cnet.com/newstech/communications/0,39001141,39145660,00.htm

[2] ITU-T Recommendation H.263, "Video coding for low bit rate communication," 1998.

[3] ITU-T Recommendation H.262 ISO/IEC 13818-2 MPEG-2 video, "Generic coding of moving pictures and associated audio, Part2: Video," 1994.

[4] IISO/IEC JTC1/SC29/WG11 14496-2, "Information technology - Generic coding of audio-visual objects Part 2: Video," March 1999.

[5] S Kappagantula and K R Rao, "Motion compensated predictive coding," *Proc. SPIE 27th*, pp. G4-70, 1983.

[6] T Koga, K Iinuma, A Hirano, Y Iijima, and T Ishiguro, "Motion compensated interframe coding for video conferencing," *Proc. National Telecommunication Conference*, pp. G.5.3.1-5.3.5, 29 Nov. - 3 Dec. 1981.

[7] J R Jain and A K Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. Communications*, vol. 29, pp. 1799-1806, Dec. 1981.

[8] M Ghanbari, "The cross-search algorithm for motion estimation," *IEEE Trans. Communications*, vol. 38, pp. 950-953, July 1990.

[9] J Y Tham, S Ranganath, M Ranganath, and A A Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. Circuits and System. Video Technology*, vol. 8, pp. 369-377, Aug.1998.

[10] B Liu and A Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits and System. Video Technology*, vol. 3, pp. 145-157, Apr. 1993.

[11] C K Cheung and L M Po, "Normalized partial distortion search algorithm for block motion estimation," *IEEE Trans. Circuits and System. Video technology*, vol. 10, pp. 417-422, Apr. 2000.

[12] T Koga, K Linuma, A Hirano, Y lijima, and T Ishiguro, "Motion compensated interframe coding for video conference," *Proc. NTC81*, Nov. 1981.

[13] V Bhaskar and K Konstantinides, "DCT based Coding," in *Image and Video Compression Standards Algorithms and Architectures*: Kluwer Academic Publishers, 1997, pp. 74.

[14] ITU-T Recommendation H.261, "Video CODEC for Audiovisual Services at p×64 kbits," March 1993.

[15] ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services," May 2003.

[16] ISO/IEC 11172-2 MPEG-1 video, "Information technology-coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbits-part 2: Video," 1993.

[17] ISO/IEC 14492-2 MPEG-2 visual, "Information technology - Coding of audio-visual objects part 2: Visual," July 2000.

[18] ISO/IEC JTC/SC29/WG11 Doc.N37511, "Introduction to MPEG-7," La Baule, France, Oct. 2000.

[19] ISO/IEC JTC1/SC29/WG11 Doc. N4041, "MPEG-21 Overview," Singapore, March 2001.

[20] ITU-T Recommendation H.263 draft, "Video Coding for low bit rate communication," May 1996.

[21] ITU-R Recommendation BT.601-5, "Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios," 1995.

[22] T Chen, "H.263, H.263 version2, and H.26L," http://www.ece.cmu.edu/~ee899/lecture5.pdf

[23] ITU-T Recommendation H.263 - Annex U, "Video Coding for low bit rate communication - Annex U: Enhanced reference picture selection mode," Nov. 2000.

[24] ITU-T Recommendation H.263 - Annex V, "Video Coding for low bit rate communication - Annex V: Data-partitioned slice mode," Nov. 2000.

[25] ITU-T Recommendation H.263 - Annex W, "Video Coding for low bit rate communication - Annex W: Additional supplemental enhancement information specification," Nov. 2000.

[26] A Luthra, G J Sullivan, and T Weigand, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits and System. Video Technology*, vol. 13, pp. 560-576, July, 2003.

[27] 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. 13, pp. 688-703, July 2003.

[28] M Karczewicz and R Kurceren, "The SP- and SI-Frames design for H.264/AVC," *IEEE Trans. Circuits and System. Video Technology*, vol. 13, pp. 637-644, July 2003.

[29] H S Malvar, A Hallapuro, and M Karzewicz, "Low-complexity transform and quantisation in H.264/AVC," *IEEE Trans. Circuits and System. Video Technology*, vol. 13, pp. 598-603, July 2003.

[30] M Wien, "Variable block-size transform for H.264/AVC," *IEEE Trans. Circuits and System. Video Technology*, vol. 13, pp. 604-613, July 2003.

[31] D Marpe, H Schwarz, and T Wiegand, "Context-based adaptive binary coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits and System. Video Technology*, vol. 13, July 2003.

[32] M Wien, "Adaptive Deblocking Filter," *IEEE Trans. Circuits and System. Video Technology*, vol. 13, pp. 604-613, July 2003.

[33] T Ebrahimi and C Horne, "MPEG-4 natural video coding-an overview," http://ltswww.epfl.ch/~tvgroup/publications/publications_2000/PaperRev1.pdf

[34] N Brady, "MPEG-4 standardized methods for the compression of arbitrarily shaped video objects," *IEEE Trans. Circuits and System. Video Technology*, vol. 9, pp. 1170-1189, Dec. 1999.

[35] R Koenen, "Profiles and levels in MPEG-4: Approach and overview," *Signal Processing: Image Communications*, vol. 15, pp. 463-478, Dec. 1999.

[36] ISO/IEC IS11172 MPEG-1 video, "Information technology - Coding of moving pictures and associated audio for digital storage media up to about 1.5Mb/s - Part 2: Video," 1993.

[37] S F Chang, T Sikora, and A Puri, "Overview of the MPEG-7 standard," *IEEE Trans. Circuits and System. Video technology*, vol. 11, pp. 688-695, June 2001.

[38] ISO/IEC JTC1/SC29/WG11, "Coding of moving pictures and audio," Shanghai, October 2002.

[39] D Sow and A Eleftheriadis, "Complexity Distortion Theory," *Proc. ISIT 1997*, Germany, June 29-July 4, 1997.

[40] D Sow and A Eleftheriadis, "Complexity Distortion Theory," *IEEE Trans. Information Theory*, vol. 49, pp. 604-608, March 2003.

[41] G Sullivan and T Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, Nov. 1998.

[42] J I Ronda, M Eeckert, F Jaureguizar, and N Garcia, "Rate control and bit allocation for MPEG-4," *IEEE Trans. Circuits and System. Video Technology*, vol. 9, pp. 1243-1258, Dec. 1999.

[43] K Ramchandran, A Ortega, and M Vetterli, "Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders," *IEEE Trans. Image Processing*, vol. 3, pp. 533-545, Sep. 1994.

[44] ITU-T Study group 15, "Video CODEC test model, Version 5," *Jan. 1995.*

[45] ITU-T Study group 16, "Video CODEC test model, Version 8," Jan. 1997.

[46] J Ribas-Corbera and S Lei, "Rate control in DCT video coding for low-delay communications," *IEEE Trans. Circuits and System. Video Technology*, vol. 9, pp. 172-185, Feb. 1999.

[47] ISO/IEC JTC1/SC29/WG11, "MPEG-4 Video Verification Model version 18.0," January 2001.

[48] T Chiang and Y Zhang, "A new rate control scheme using quadratic rate distortion model," *IEEE Trans. Circuits and System. Video Technology*, vol. 7, pp. 246-250, Feb. 1997.

[49] K R Rao and P Yip, "Discrete Cosine Transform Algorithms, Advantages, Applications," *Academic Press*, New York, 1990.

[50] A Hossen and U Heute, "Fast approximation DCT:basic idea, error analysis, applications," *Proc. ICASSP97*, March 1997.

[51] Z Wang, "Pruning the Fast Discrete Cosine Transform," *IEEE Trans. Communications*, May 1991.

[52] V K Goyal and M Vetterli, "Computation-distortion characteristics of block transform coding," *Proc. ICASSP97*, April 1997.

[53] V K Goyal and M Vetterli, "Computation-distortion characteristics of JPEG encoding and decoding," *Proc. 31st Asilomar Conference on Signals, Systems and Computers*, 1997.

[54] M J Gormish, "Quantization and Computation-Rate-Distortion," in *Source Coding with Channel, Distortion and Complexity Constraints*, 1994.

[55] K. Lengwehasatit and A. Ortega, "Rate-Complexity-Distortion Optimization for Quadtree-Based DCT Coding," *Proc. ICIP 2000*, Sep. 2000.

[56] W Chen, C Smith, and S Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform," *IEEE Trans. Communications*, vol. COM-25, pp. 1004-1009, Sep. 1977.

[57] I Richardson, "Full Search Motion Estimation," in *Video CODEC Design*, 2002, pp. 99-101.

[58] A M Eskicioglu and P S Fisher, "Image quality measures and their performance," *IEEE Trans. Communications.*, vol. 43, pp. 2959-2965, Dec. 1995.

[59] S A Karunasekera and N G Kingsbury, "A distortion measure for blocking artifacts in images based on human visual sensitivity," *IEEE Trans. Image Processing*, vol. 4, pp. 713-724, June 1995.

[60] C H Chou and Y C Li, "A perceptually tuned subband image coder based on the measure of Just-Noticeable-Distortion profile," *IEEE Trans. Circuits and System. Video Technology*, vol. 5, pp. 467-476, Dec. 1995.

[61] ITU-T Recommendation P.910, "Subjective video quality assessment mehtods for multimedia applications," Sep. 1999.

[62] N Ahmed, T Natarajan, and K R Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, pp. 90-93, Jan. 1974.

[63] F Arguello and E L Zapata, "Fast cosine transform based on the successive doubling method," *Electronics Letters*, vol. 26, pp. 1616-1618, Sep. 1990.

[64] S C Chan and K L Ho, "A new two-dimensional fast cosine transform algorithm," *IEEE Trans. Signal Process*, vol. 39, pp. 481-485, Feb. 1991.

[65] H S Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust. Speech Signal Processing*, vol. 35, pp. 1455-1461, Oct. 1987.

[66] A N Hossen, U Heute, O Shentov, and S Mitra, "Subband DFT-part II: Accuracy, Complexity, and Applications," *Signal Processing*, vol. 41, pp. 279-294, Feb. 1995.

[67] S Jung, S Mitra, and D Mukherjee, "Subband DCT: definition, analysis and applications," *IEEE Trans. Circuits and System. Video Technology*, vol. 6, pp. 273-286, June 1996.

[68] A Skodras, "Fast Discrete Cosine Transform Pruning," *IEEE Trans. Signal Processing*, pp. 1799-1808, July 1994.

[69] C A Christopoulos, J Bormans, J Cornelis, and A N Skodras, "The vector-radix fast coine transform: pruning and complexity analysis," *Signal Processing*, vol. 43, pp. 197-205, 1995.

[70] D Kalles and A N Skodras, "Motion estimation and DCT pruning: a combined approach to video coding," *Proc. of the 3rd IEEE Int. Conf. on Electronics, Circuits and Systems (ICECS '96)*, pp. 736-739, Oct. 1996.

[71] K Lengwehasatit and A Ortega, "Distortion/Decoding Time Tradeoffs in Software DCT-based Image Coding," *Proc. ICASSP97*, Munich, Germany, Apr.997.

[72] K Lengwehasatit and A Ortega, "DCT Computation Based on Variable Complexity Fast Approximations," *Proc. ICIP98*, Oct. 1998.

[73] R L de Queiroz, "Variable complexity DCT approximations driven by an HVQ-based analyzer," *IEEE Trans. on Circuits and System. Video Technology*, vol. 12, pp. 1021-1024, Nov. 2002.

[74] R C Chang, J May, and R Gray, "Hierarchical vector quantization with table look-up encoders," *Proc. int. Conf. Communications*, pp. 1452-1455, 1985.

[75] I M Pao and M T Sun, "Approximation of Calculations for Forward Discrete Cosine Transform," *IEEE Trans. Circuits and Systems. Video Technology*, vol. 8, pp. 264-268, June 1998.

[76] I M Pao and M T Sun, "Modelling DCT Coefficients for Fast Video Encoding," *IEEE Trans. Circuits and Systems. Video Technology*, vol. 9, pp. 608-616, June 1999.

[77] ISO/IEC JTC1/SC29/WG11 Appendix J, "MPEG-4 Video Verification Model version 18.0," pp. 311-314, Jan. 2001.

[78] I E G Richardson and Y Zhao, "Adaptive management of video encoder complexity," *Journal of Real-time Imaging*, vol. 8, pp. 291-301, Aug 2002.

[79] l M Po and W C Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits System. Video Technology*, vol. 6, pp. 313-317, 1996.

[80] Y H Fok and O C Au, "Fast motion estimation using spatial decimation," *Proc. IEEE Int. Conf. on Image processing 1995*, Las Vegas, Nov. 1995.

[81] R Braspenning, G de Haan, and C Hentschel, "Complexity scalable motion estimation," *Proc. VCIP 2002*, Jan. 2002.

[82] M Gallant, G Côté, 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. 1816-1822, Dec. 1999.

[83] K Lengwehasatit and A Ortega, "A novel computationally scalable algorithm for motion estimation," *Proc. VCIP1998*, Jan. 1998.

[84] G de Haan and P Biezen, "Sub-pixel motion estimation with 3-D recursive search block-matching," *Signal processing: Image communication*, vol. 6, pp. 229-239, 1994.

[85] I Rhee, "Error control techniques for interactive low-bit rate video transmission over the internet," *ACM SIGCOMM'98*, 1998.

[86] P L Tai, C T Liu, and J S Wang, "A new fast block matching algorithm based on complexity-distortion optimization," *Journal of Imaging Systems and Technology*, vol. 12, 2002.

[87] J F Yang, S C Chang, and C Y Chen, "Computation reduction for motion search in low rate video coders," *IEEE Trans. Circuits and System. Video Technology*, vol. 12, pp. 948-951, Oct. 2002.

[88] Y Zhao and I Richardson, "Macroblock classification for complexity management of Video Encoders," *Singal Processing: Iamge Communication*, vol. 18, Oct. 2003.

[89] ITU-T Study group 16, "Video CODEC test model, Version 10," Apr. 1998.

[90] Y Zhong, I Richardson, A Sahraie, and P McGeorge, "Qualitative and quantitative assessment in video compression," *Proc. 12th European Conference on Eye Movements*, Dundee,Scotland, 20-24 Aug. 2003.

[91] A Chang, O C Au, and Y M Yeung, "A novel approach to fast multi-frame selection for H.264 video coding," *Proc. ICASSP2003*, 2003.

[92] Hye-Yeon Cheong and Alexis Tourapis, "Fast motion estimation within the H.264 CODEC," *Proc. ICME 2003*, July 2003.

[93] Y K Tu, J F Yang, Y N Shen, and M T Sun, "Fast variable-size block motion estimation using merging procedure with adaptive threshold," *Proc. ICME 2003*, July 2003.

Bibliography

1. I. E. G. Richardson, *Video CODEC Design*, John Wiley & Sons Ltd, 2002.

2. V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards Algorithms and Architectures*, Second Edition, Kluwer Academic Publishers, 1997.

3. M. J. Riley. and I. E. G. Richardson, *Digital Video Communications*, Artech House, 1997.

4. Y. Wang, J. Ostermann, Y. Zhang, *Video Processing and Communications*, Prentice Hall, 2002.

5. K. R. Rao, *Discrete Cosine Transform*, Harcourt Brace Jovanovich Publishers, 1990.

6. K. R. Rao, Z. S. Bojkovic and D. A. Milovanovic, Milovanovic, *Multimedia Communication Systems Techniques, Standards and Networks*, Prentice Hall, 2002.

7. A. H. Sadka, *Compressed Video Communications*, John Wiley &Sons, Ltd, 2002.

8. J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video Compression Standard*, Chapman & Hall, 1997.

9. G. Bhatnager, S. Mehta, S. Mitra, *Introduction to Multimedia Systems*, Academic Press, 2002.

10. T. S. Rzeszewski, *Digital Video Concepts and Applications across Industries*, IEEE Press, 1995.

11. F. Pereira and T. Ebrajimi., *The MPEG-4 Book*, Prentice Hall, 2002.

12. J. Watkinson, *The MPEG Handbook*, Focal Press, 2001.

Appendix: List of publications

The publications related with this work are listed below and the two journal papers are appended.

Journal papers

1. Y Zhao and I E G Richardson, "Macroblock Classification for Complexity Management of Video Encoders", Signal Processing: Image Communication, Vol. 18, Issue 9, Oct. 2003 pp. 801-811.

2. Iain E G Richardson and Yafan Zhao, "Adaptive management of Video Encoder complexity", Journal of Real-Time Imaging, Vol. 8, No. 4, Aug. 2002, pp. 291-301.

Conference and seminar papers

1. Y Zhao and I E G Richardson, "Macroblock skip-mode prediction for Complexity control of Video Encoders", Proc. IEE Visual Information Engineering, 7~9 July 2003, Surrey, U.K, pp. 5-8.

2. Y Zhao and I E G Richardson, "Macroblock Classification for Video Encoder Complexity Management", Proc. International Picture Coding Symposium, 23-25 April 2003, Saint-Malo, France, pp. 371-376.

3. Y Zhao and I E G Richardson, "Complexity Management for Video Encoders", Doctoral Symposium, ACM Multimedia Conference, 1-6 December 2002, Juan Les Pins, France, pp. 647-649.

4. Y Zhao, L Muir and I E G Richardson, "Implementing MPEG-4 Visual in software", IEE Seminar on Visual Media Standards, 25 April 2002.

5. Y Zhao and I E G Richardson, "Computational Complexity Management of Motion Estimation in Video Encoders", IEEE Data Compression Conference, 2-4 April, 2002, Snowbird, UT, USA, pp. 483.

6. Iain E G Richardson and Yafan Zhao, "Adaptive Algorithms for Variable-complexity Video Coding", Proc. Internal Conference on Image Processing, 7-10 October 2001, Thessaloniki, Greece, pp. 457-460.

7. Iain E G Richardson and Yafan Zhao, "Video CODEC Complexity Management", Proc. Picture Coding Symposium, 25-27 April 2001, Seoul, Korea, pp. 267-270.

Patent application

1. I E G Richardson and Yafan Zhao, "Macroblock Classification for Complexity Management of Video Encoders", UK Patent application 0229354.6, filed 18 December 2002.

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Signal Processing: *Image Communication* 18 (2003) 801–811
SIGNAL PROCESSING: IMAGE COMMUNICATION
www.elsevier.com/locate/image

Macroblock classification for complexity management of video encoders

Yafan Zhao*, Iain E.G. Richardson

School of Engineering, The Robert Gordon University, Schoolhill, Aberdeen AB10 1FR, UK

Received 7 January 2003; received in revised form 6 May 2003; accepted 13 June 2003

Abstract

Typically, many macroblocks (MBs) are skipped during encoding of H.263 or MPEG-4 SP video data, particularly at low bit-rates. In this paper, we describe an algorithm that predicts the occurrence of skipped MBs prior to encoding, making it possible to save significant computational effort by not coding these MBs. The algorithm estimates the energy of low-frequency quantized coefficients in order to classify each MB as 'skipped' or 'not skipped'. Results show that the algorithm can deliver substantial computational savings at the expense of a small reduction in rate-distortion performance.

1. Introduction

Video CODECs based on the H263 [5] and MPEG-4 [3] video coding standards are used in a wide range of applications. Software-only CODECs are becoming particularly popular, offering advantages such as flexibility, ease of upgrading and distribution. In real-time and/or power-constrained applications, the performance of a video CODEC may be limited by the amount of processing power available as well as, or rather than, the available transmission bandwidth. In a desktop video conferencing system, the CODEC runs on a general-purpose PC and has to share processing resources with other applications. In a mobile video handset, power consumption is closely related to processor utilization and it may be necessary to restrict computational processing in order to maximize battery life. Current software video applications typically control processor utilization by dropping frames during encoding, leading to intermittent and 'jerky' motion in the decoded video sequence. Hence, computational complexity can be a major constraint on coding performance. It is therefore important to develop methods of managing the computational complexity of video CODECs.

Previous work on reducing computational complexity of video CODECs has included many proposals for 'fast search' motion estimation algorithms [6,7]. Adaptive motion estimation algorithms such as Nearest Neighbour Search [1] provide coding performance that is close to that of Full Search with greatly reduced complexity. However, the computational cost of this type of algorithm can vary significantly depending on the scene characteristic of the video sequence. Several

*Corresponding author. Fax: +44-1224-262-444.
E-mail address: 9905854@rgu.ac.uk (Y. Zhao).

0923-5965/$ - see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/S0923-5965(03)00072-9

methods [10,2] have been proposed to reduce the computational complexity in the discrete cosine transform (DCT) by calculating a subset of the DCT coefficients. Applying these methods to all blocks in an image will significantly reduce the quality of the decoded image and this has led to proposals for algorithms that selectively calculate the DCT based on sequence statistics [11]. In [9], we describe an algorithm that enables flexible and accurate management of DCT complexity in a software video encoder and a similar approach to motion estimation complexity is presented in [14]. The initial calculation of sum of absolute differences (SAD) for zero motion vector (MV) is used to reduce motion estimation complexity in [13] and the authors report that this method performs well together with DCT computation reduction. The results presented in these papers show that variable-complexity algorithms can reduce computational complexity, usually at the expense of increased distortion.

Many coded macroblocks (MBs) in an inter-coded frame have zero MVs and/or quantized coefficients (Q_{Coeff}). An MB with zero MV and no non-zero coefficients is skipped. If these skipped MBs could be accurately predicted prior to encoding, all subsequent operations on these MBs (motion estimation and compensation, DCT, quantization, etc.) could be avoided, saving considerable computational effort.

This paper presents a pre-classification algorithm that categorizes MBs into two types, 'skipped' and 'not skipped', prior to encoding. Computational complexity can be reduced by not processing MBs that are expected to be skipped. The proposed algorithm provides a simple, controllable and robust method of managing computational resources. It can control the amount of reduced computation whilst minimizing distortion due to occasional incorrect MB classification.

This paper is organized as follows. Section 2 introduces the concepts of MB classification and the trade-off between computational complexity and rate-distortion performance. Section 3 describes the development of a pre-classification algorithm based on measured characteristics of an input MB and Section 4 presents simulation results for the proposed algorithm. Conclusions and suggestions for further work are given in Section 5.

2. Distribution of macroblock types

A CODEC that conforms to one of the popular DCT-based video coding standards (such as H.263 or MPEG-4 Simple Profile) processes each frame in units of an MB. In an inter-coded picture, motion estimation is carried out in order to find a suitable prediction for the current MB from a reference frame. Each block of the motion-compensated residual MB is coded using the DCT, quantization, reordering and entropy coding. MVs and quantized coefficients are encoded together with side information and the MB is reconstructed for prediction of further pictures.

Four video sequences ('Carphone', 'Mother and Daughter', 'Foreman' and 'Claire') were encoded using an H.263+ encoder (corresponding to the low complexity mode of test model TMN10 [12]; hereafter described as 'TMN10') with a fixed quantizer step size. Coded MBs in P-pictures were categorized into four types: 'skipped' (zero MV, no non-zero Q_{Coeff}), 'MV = 0' (zero MV, some non-zero Q_{Coeff}), '$Q_{Coeff} = 0$' (non-zero MV, no non-zero Q_{Coeff}) and 'other' (non-zero MV and non-zero Q_{Coeff}). Fig. 1 shows the distribution of the four categories (a) for 'Carphone' encoded with two quantizer step sizes and (b) for the remaining three sequences encoded with a quantizer step size of 8. Fig. 1(a) demonstrates that the proportion of skipped MBs increases with quantizer step size and Fig. 1(b) shows that low-motion sequences such as 'Claire' and 'Mother and Daughter' contain a higher proportion of skipped MBs than high-motion sequences such as 'Carphone' and 'Foreman'. With the exception of 'Foreman', the majority of MBs in each sequence (i) contain no MVs, (ii) contain no coefficients or (iii) are skipped.

It is clear from Fig. 1 that a significant proportion of MBs are skipped (not coded), particularly in low-motion sequences and/or at higher quantizer step sizes (and hence lower bit-rates). Predicting the presence of a skipped MB prior to coding could make it possible to save

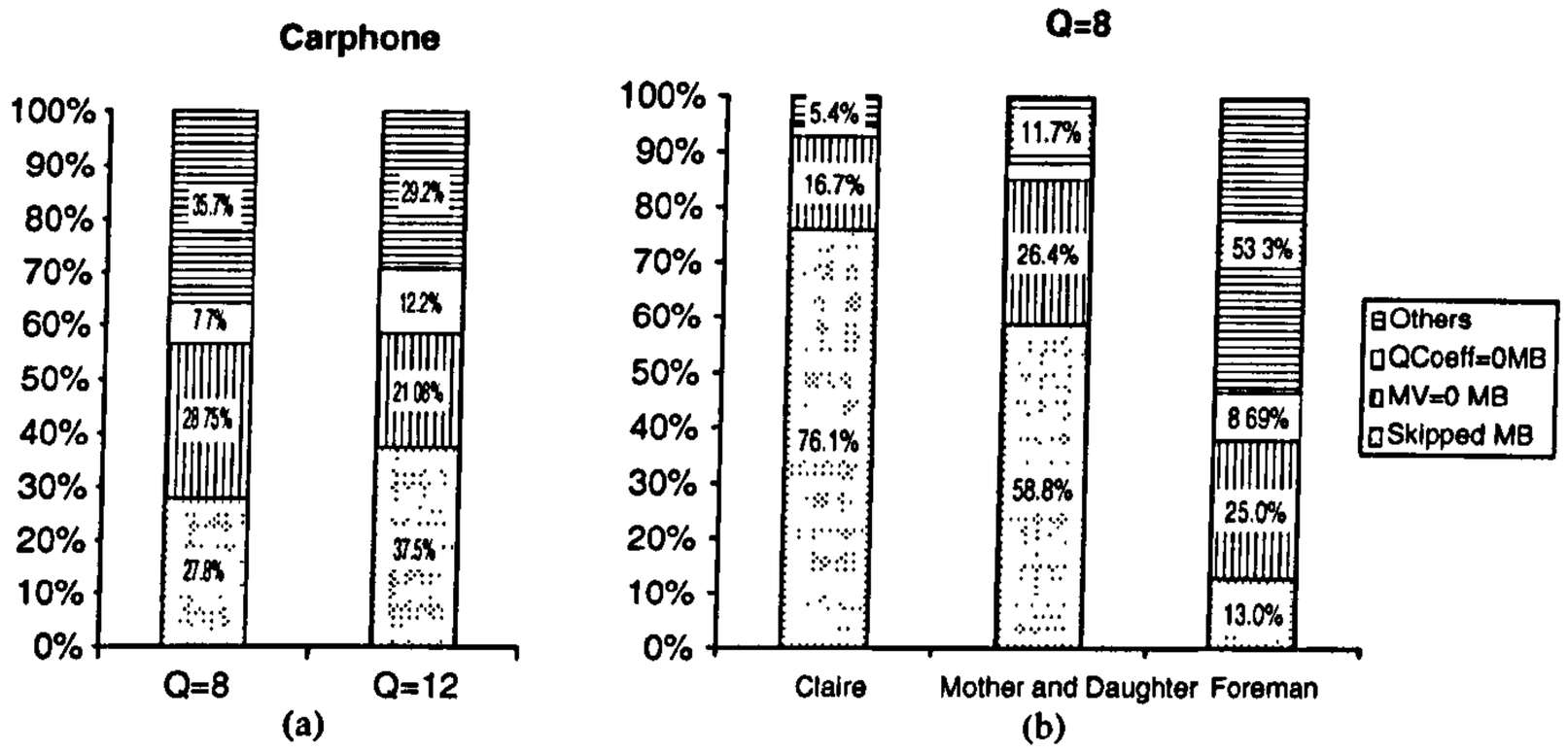


Fig. 1. Distribution of four types of MBs in video sequences: (a) Carphone; and (b) $Q = 8$.

considerable computational resources by not carrying out computationally intensive functions. We propose the following approach to MB classification:

1. Prior to encoding, classify each inter-coded MB as 'skipped' or 'not skipped' by prediction from local sequence statistics.
2. If the MB is predicted as 'not skipped', carry out the usual encoding functions (motion estimation and compensation, DCT, quantization, rescaling, IDCT, reconstruction, reordering, run-level coding, entropy coding).
3. If the MB is predicted as 'skipped', indicate the presence of a skipped MB in the bitstream; no further processing is carried out.

If the prediction of MB type is correct, computational complexity is reduced without any effect on decoded video quality. If an MB that should have been encoded (i.e. an MB that contains non-zero MV and/or Q_{Coeff} after encoding) is predicted as 'skipped', a reduction in decoded quality is likely to occur. Computational complexity reduction may therefore lead to increased distortion. We have argued previously [9] that a small reduction in PSNR is an acceptable penalty for reduced computation. A limited degradation in PSNR (less than 1 dB) is difficult to distinguish subjectively [8]. Furthermore, in a real-time video application, maintaining a consistent video frame rate through computational complexity management (at the expense of limited reduction in PSNR) is likely to be preferable to a 'jerky' decoded video sequence due to an encoder dropping frames.

3. Macroblock classification algorithms

3.1. Correlation between residual energy and probability of skipped macroblock

MBs that are skipped have zero MV and Q_{Coeff}. This means that (a) the closest matching region is in the same position in the reference frame and (b) the energy of the residual MB (after subtracting the reference region) is low, such that there are no non-zero DCT coefficients after quantization. Both of these conditions are likely to be met if there is a strong similarity between the current MB and the same MB position in the reference frame. The energy of the residual MB formed by subtracting the reference MB (without motion compensation) from the current MB is approximated by $\mathrm{SAD0}_{\mathrm{MB}}$ (SAD for luminance part of MB, zero displacement):

$$\mathrm{SAD0}_{\mathrm{MB}} = \sum_{i=0}^{15}\sum_{j=0}^{15} |C_{\mathrm{C}}(i,j) - C_{\mathrm{P}}(i,j)|, \qquad (1)$$

$C_{\mathrm{C}}(i,j)$ and $C_{\mathrm{P}}(i,j)$ are luminance samples from an MB in the current frame and in the same position in the reference frame, respectively.

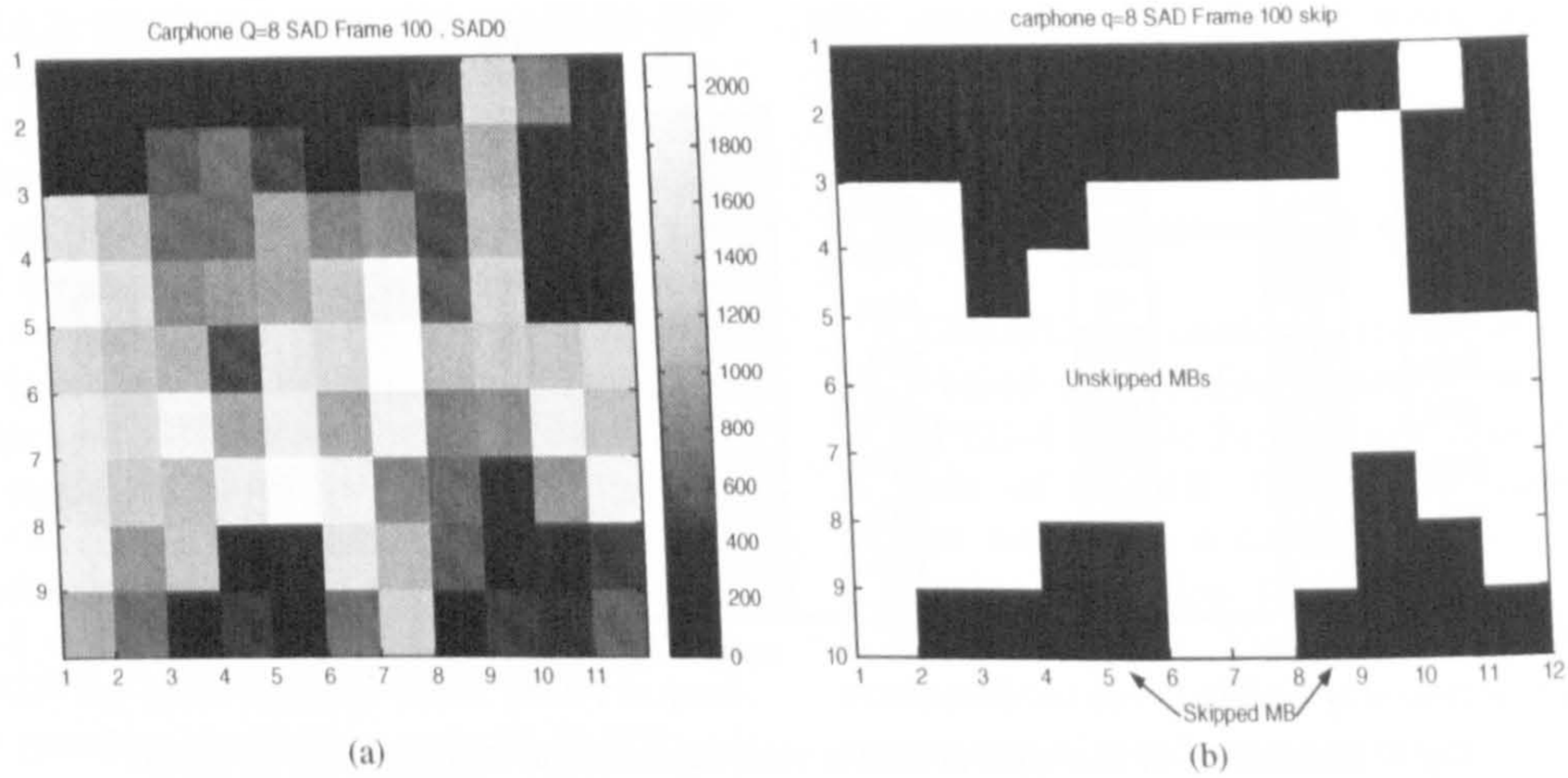


Fig. 2. $SAD0_{MB}$ and skipped MB (Carphone, $Q = 8$, Frame 100): (a) $SAD0_{MB}$: and (b) skipped MB.

$SAD0_{MB}$ represents the total energy of a residual MB. A skipped MB has low residual energy and a zero MV and so we might expect to find a correlation between $SAD0_{MB}$ and P(skip), the probability of skipping the current MB. Fig. 2(a) plots $SAD0_{MB}$ for frame 100 of the 'Carphone' sequence (QCIF format, encoded with a fixed quantizer step size of 8) and Fig. 2(b) plots the distribution of skipped MBs after coding this frame using TMN10. It is clear that skipped MBs often have low values of $SAD0_{MB}$. It may therefore be possible to use $SAD0_{MB}$ to predict whether an MB is likely to be skipped.

3.2. Classifying skipped MBs using $SAD0_{MB}$

The 'Carphone' sequence was coded using TMN10 with fixed quantizer step size of 8. The dotted region of Fig. 3 plots $SAD0_{MB}$ (x-axis) against P(skip), the probability that an MB with a given $SAD0_{MB}$ will be skipped (y-axis). The solid line plots the cumulative density (CDF) of $SAD0_{MB}$.

Fig. 3 indicates that low $SAD0_{MB}$ correlates with high P(skip). The relationship between $SAD0_{MB}$ and P(skip) also depends on the quantizer step size since a higher step size results in an increased proportion of skipped MBs (as shown in Fig. 1). Fig. 4 plots P(skip) against ($SAD0_{MB}/Q$) for the 'Carphone' sequence with various quantizer step sizes Q. The plots of P(skip) in this figure are smoothed with a moving average operator to facilitate comparison of the different sequences.

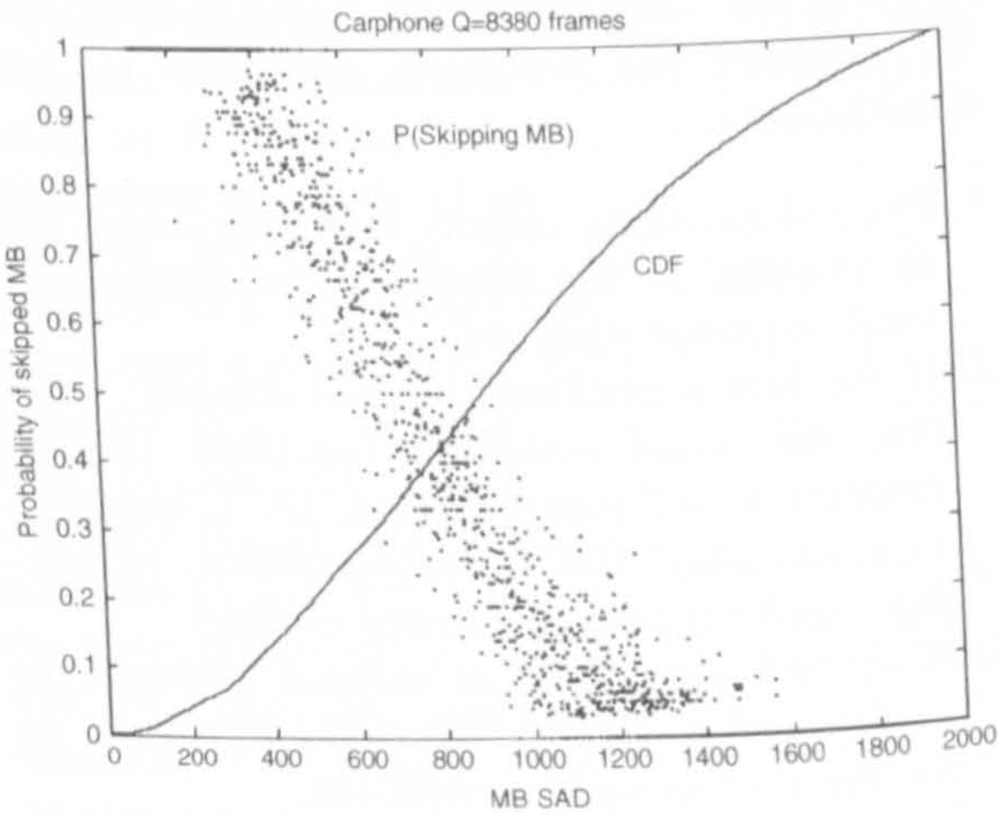


Fig. 3. Probability of skipping MB with $Q = 8$.

It may be possible to use $SAD0_{MB}/Q$ as a predictor to determine whether a given MB is likely to be skipped. An algorithm for MB classification can be described in +code as follows:

IF $SAD0_{MB}/Q < T_{sad0}$
THEN skip coding this MB and set MV and Q_{Coeff} to zero
ELSE continue coding

When $SAD0_{MB}/Q$ is less than a pre-determined threshold, the encoder skips coding this MB and sets MV and Q_{Coeff} to zero directly, hence reducing the processing required for the MB.

3.3. Improving the accuracy of the classification algorithm

The classification algorithm described in the previous section may not produce the correct classification for every MB. Fig. 3 indicates that P(skip) is less than 1.0 in many cases. Whatever threshold value T_{sad0} is chosen, there may be some MBs that have a value of $SAD0_{MB}$ below this threshold but should not be skipped. The classification algorithm will fail in these cases, causing MBs to be erroneously skipped and increasing distortion in the decoded video sequence. Fig. 5 shows one example of an MB falling into this category. MB(8,4) in Frame 18 of 'Carphone' has a low value of $SAD0_{MB}$ (5 3 5) but is not skipped.

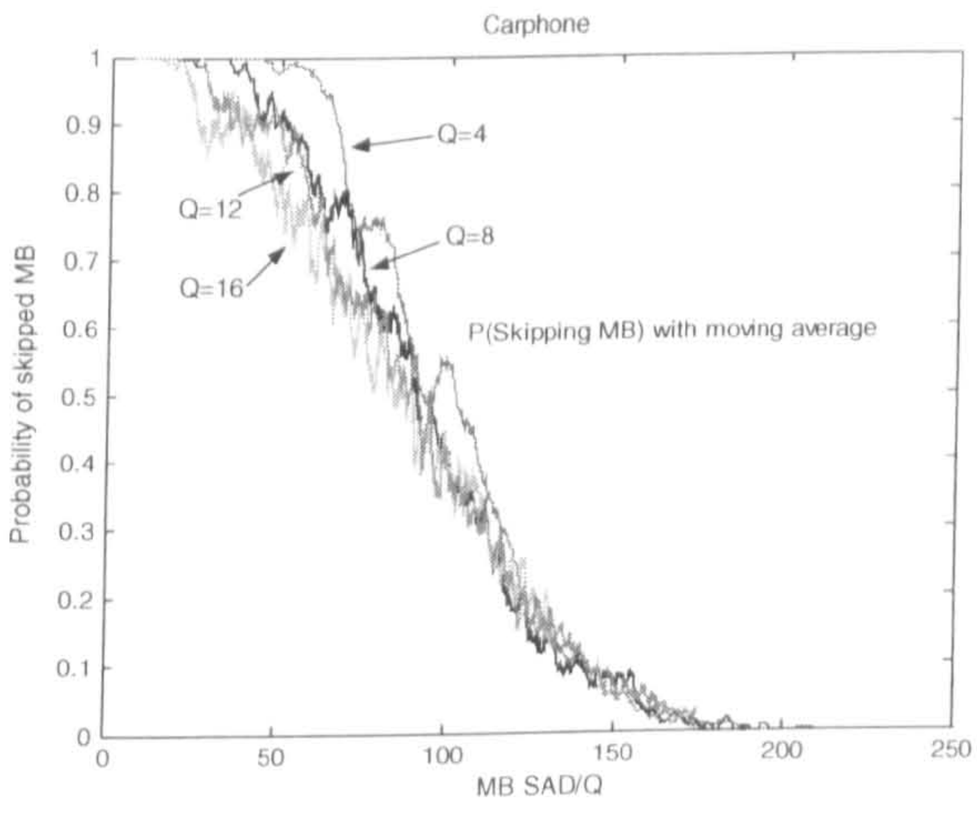


Fig. 4. Probability of skipping MB with normalized $SAD0_{MB}$.

Examination of MBs that fall into this category shows that these MBs often occur on the edges of moving areas. The residual (without motion compensation) typically contains a small number of high-valued samples, not enough to cause a significant increase in SAD but enough to produce non-zero quantized coefficients. Fig. 6 shows the residual and DCT output for the luminance part of MB(8,4) in frame 18 of 'Carphone'. The residual (a) contains a single high-valued region (corresponding to a moving edge in the video sequence). The unquantized DCT coefficient magnitude of the four luminance blocks (b) shows that the top-left block has a number of significant coefficients. In particular, the four low-frequency DCT coefficients (the DC coefficient and the three lowest AC coefficients) have large magnitudes.

A high-magnitude DC coefficient tends to produce a correspondingly high $SAD0_{MB}$ (since the DC coefficient is proportional to the mean sample value of each block). However, the three lowest-frequency AC coefficient magnitudes are not reflected in the calculation of $SAD0_{MB}$. We therefore propose a low-complexity method of estimating the magnitude of these coefficients (without actually carrying out the DCT) and this is illustrated in Fig. 7.

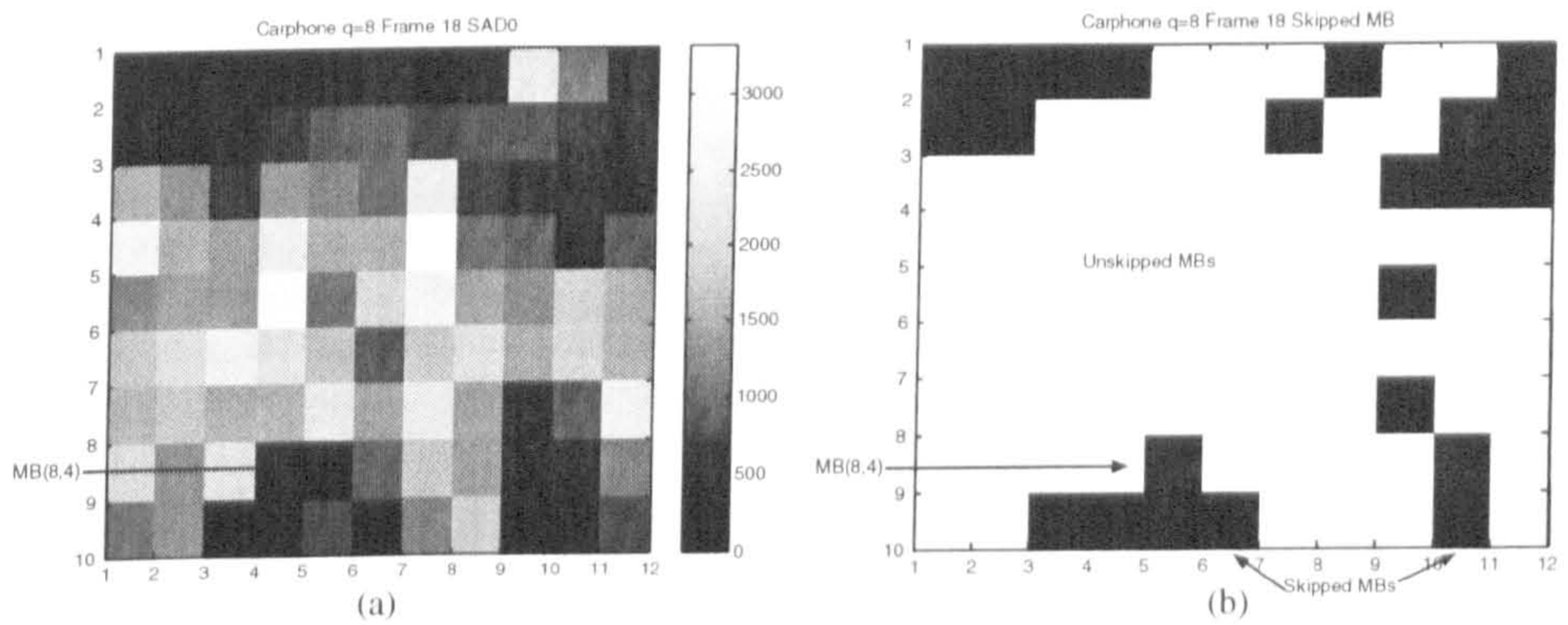


Fig. 5. Example of low-$SAD0_{MB}$ value MB (Carphone, $Q = 8$, Frame 18, MB (8, 4)): (a) $SAD0_{MB}$; and (b) skipped MB.

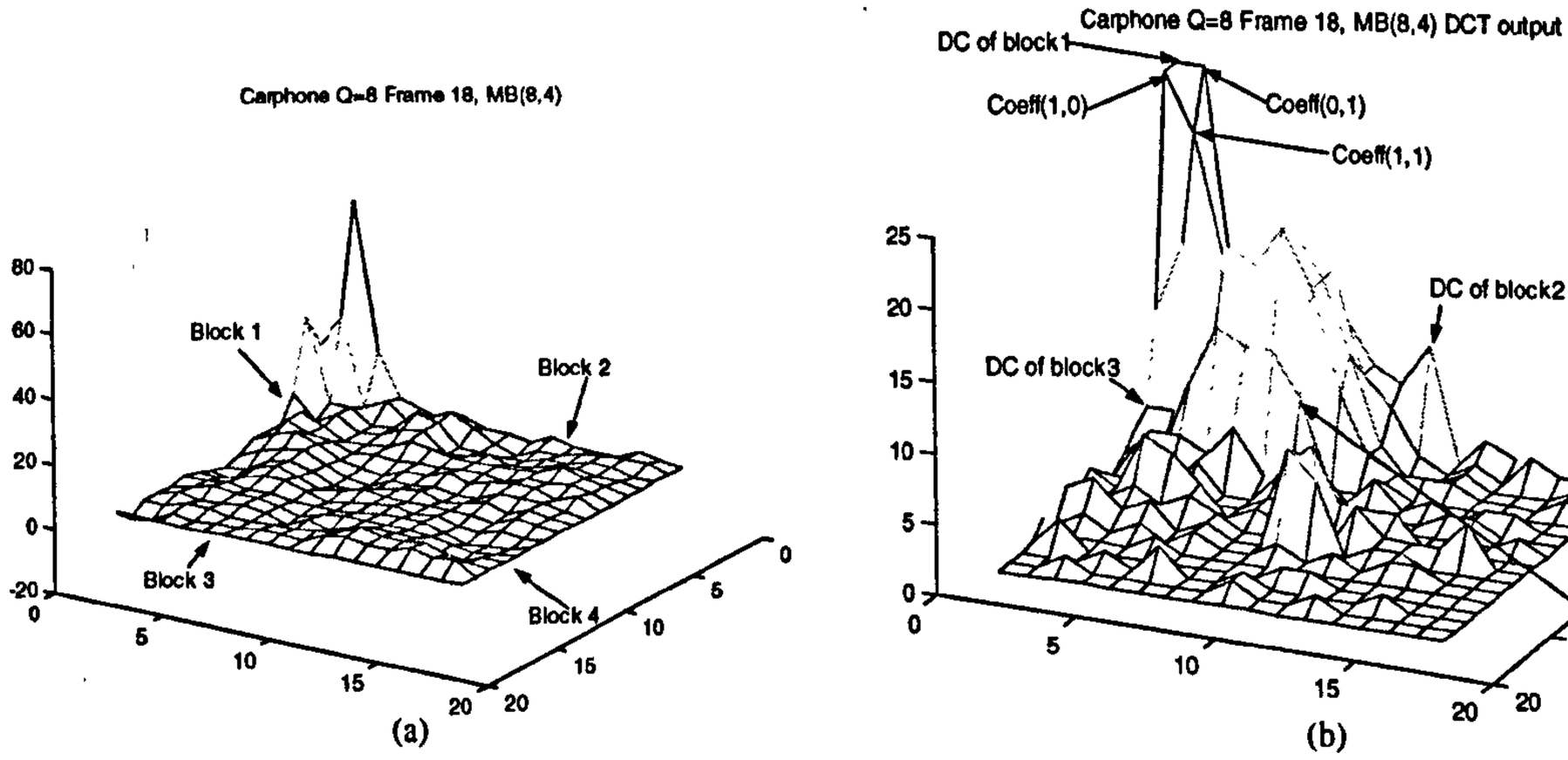


Fig. 6. Example of DCT process of (Carphone, $Q = 8$, Frame 18, MB (8, 4)): (a) residual value; and (b) absolute DCT output.

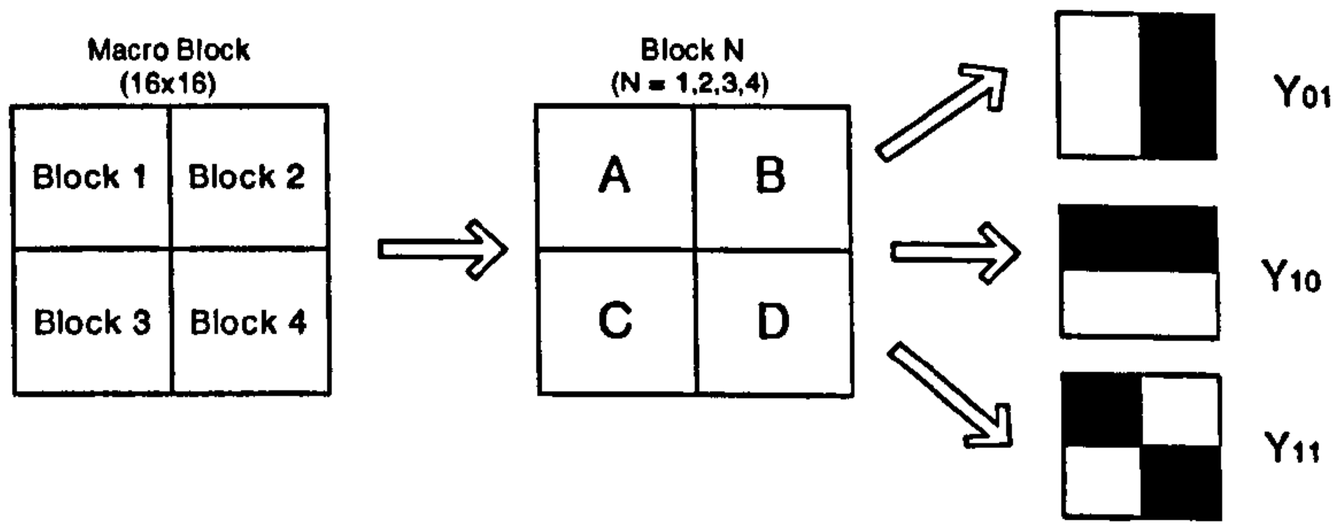


Fig. 7. Predicting the three low-frequency components.

Each 8×8 luminance block is divided into four 4×4 blocks. A, B, C and D (Eq. (2)) are the SAD values of each 4×4 block and $R(i,j)$ are the residual pixel values without motion compensation

$$A = \sum_{i=0}^{3}\sum_{j=0}^{3}|R(i,j)| \quad B = \sum_{i=0}^{3}\sum_{j=3}^{7}|R(i,j)|$$
$$C = \sum_{i=4}^{0}\sum_{j=3}^{7}|R(i,j)| \quad D = \sum_{i=4}^{7}\sum_{j=4}^{7}|R(i,j)| \tag{2}$$

Y_{01}, Y_{10} and Y_{11} (Eq. (3)) provide a low-complexity estimate of the magnitudes of the three low-frequency DCT coefficients coeff(0,1), coeff(1,0) and coeff(1,1), respectively. If any of these coefficients is large then there is a high probability that the MB should not be skipped. We therefore use $Y4 \times 4_{\text{block}}$ (Eq. (4)) to predict whether each block may be skipped. The maximum for the luminance part of an MB is calculated using Eq. (5)

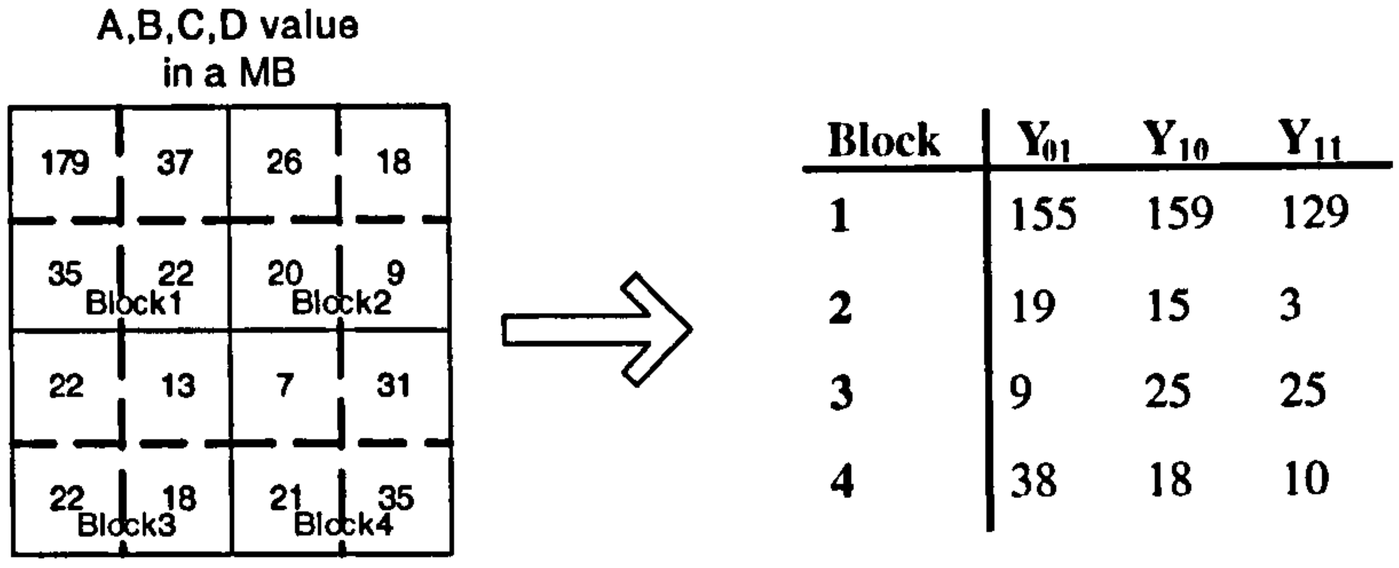
$$Y_{01} = \text{abs}(A + C - B - D),$$
$$Y_{10} = \text{abs}(A + B - C - D),$$
$$Y_{11} = \text{abs}(A + D - B - C), \tag{3}$$

$$Y4 \times 4_{\text{block}} = \max(Y_{01}, Y_{10}, Y_{11}), \tag{4}$$

$$Y4 \times 4_{\text{max}} = \max(Y4 \times 4_{\text{block1}}, Y4 \times 4_{\text{block2}}, Y4 \times 4_{\text{block3}}, Y4 \times 4_{\text{block4}}). \tag{5}$$

The calculated value of $Y4 \times 4_{\text{max}}$ is compared with a threshold $T_{4\times4}$ to improve the accuracy of skip prediction. The MB classification algorithm becomes:

IF $\text{SAD0}_{\text{MB}}/Q < T_{\text{sad0}}$ and $Y4 \times 4_{\text{max}} < T_{4\times4}$
THEN skip coding this MB and set MV and Q_{Coeff} to zero
ELSE continue coding



Block	Y_{01}	Y_{10}	Y_{11}
1	155	159	129
2	19	15	3
3	9	25	25
4	38	18	10

Fig. 8. Example of predicting the three low-frequency components (Carphone, $Q = 8$, Frame 18, MB(8, 4)).

Fig. 8 shows the results of calculating Y_{01}, Y_{10} and Y_{11} for MB(8,4) in frame 18 of 'Carphone'. Using Eqs. (3)–(5), $Y4 \times 4_{max}$ of this MB is calculated to be 159. If it is greater than the threshold $T_{4\times4}$, it will not be skipped during encoding.

3.4. Choosing threshold $T_{4\times4}$

The video sequences 'Carphone', 'Foreman' and 'Mother and Daughter' were coded using the modified classification algorithm with various values of quantizer stepsize Q and $T_{4\times4}$. For each fixed value of Q, a range of values of $T_{4\times4}$ were tested and the choice of $T_{4\times4}$ resulting in minimum distortion was recorded. The results of this experiment indicate that the optimal choice of $T_{4\times4}$ is approximately linearly related to Q (Eq. (6))

$$T_{4\times4} = 10Q + 70. \tag{6}$$

Incorporating Eq. (6) into the pre-classification algorithm gives the following:

IF $SAD0_{MB}/Q < T_{sad0}$ and $(Y4 \times 4_{max} - 70)/Q < 10$
THEN skip coding this MB and set MV and Q_{Coeff} to zero
ELSE continue coding

Applying this algorithm to MB(8,4) in frame 18 of 'Carphone' sequence, where $Y4 \times 4_{max} = 159$ and $Q = 8$:

$$(Y4 \times 4_{max} - 70)/Q = (159 - 70)/8 = 11.125.$$

This is greater than 10 and so the MB is not skipped, regardless of the value of $SAD0_{MB}$.

4. Experimental results

4.1. Performance of MB prediction

Video sequences 'Carphone', 'Mother and Daughter' and 'Forman' were coded using a modified TMN10 encoder, incorporating our MB classification algorithms (a) T_{sad0} only and (b) $T_{sad0} + T_{4\times4}$ with 280 frames and $Q = 8$. Fixed thresholds T_{sad0} and $T_{4\times4}$ were chosen for the two algorithms to achieve the same target percentage of skipped MBs. For each algorithm, the total number of skipped MBs, the number of 'missed' MBs (MBs that are skipped by the unmodified encoder but were not predicted as skipped by our algorithm) and the number of MBs wrongly skipped by our algorithm are summarized in Table 1. The average luminance PSNR drop relative to TMN10 encoder with no complexity reduction is also listed in Table 1 for each case. In order to evaluate the performance of the proposed classification algorithms, each sequence was coded by TMN10 and MPEG-4 VM18 [4] (Simple Profile) without any classification algorithms. The percentage of skipped MBs and average luminance PSNR of each sequence are shown in Table 2(a), (b), and (c) respectively.

From Table 2(a), 7712 MBs are skipped during encoding of 'Carphone' using TMN10 (27.82% of total MBs). Using MPEG-4 VM18 with the same coding parameters, 7409 MBs (26.72% of total MBs) are skipped during encoding. For the sequences listed in Table 2, TMN10 and MPEG-4 skip a similar number of MBs but MPEG-4

Table 1
Performance of skipping MBs prediction using two classification algorithms

Video sequence	Target skipped MBs (%)	Classification algorithms	Total skipped	Missed	Wrongly predicted	Average Y PSNR drop (dB)
Carphone	30	T_{sad0} only	8312	1585	2185	0.22
		$T_{sad0} + T_{4\times4}$	8329	1532	2149	0.19
	45	T_{sad0} only	12,413	523	5224	0.63
		$T_{sad0} + T_{4\times4}$	12,389	431	5108	0.51
Mother and daughter	45	T_{sad0} only	8298	8209	222	0.14
		$T_{sad0} + T_{4\times4}$	8261	8226	202	0.08
	70	T_{sad0} only	19,411	1041	4167	1.05
		$T_{sad0} + T_{4\times4}$	19,368	1011	4094	0.58
Foreman	30	T_{sad0} only	8288	318	4998	0.64
		$T_{sad0} + T_{4\times4}$	8264	280	4936	0.41

Table 2
Number of MBs skipped by TMN10 and MPEG-4

	TMN10	MPEG-4
(a) '*Carphone*' *sequence*		
Skipped MB	7712 (27.82%)	7409 (26.72%)
Average Y PSNR(dB)	34.10	34.32
(b) '*Mother and daughter*' *sequence*		
Skipped MB	16285 (58.75%)	16288 (58.75%)
Average Y PSNR(dB)	34.95	35.28
(c) '*Foreman*' *sequence*		
Skipped MB	3608 (13%)	3347 (12.07%)
Average Y PSNR(dB)	33.37	33.65

VM18 outperforms TMN10 in terms of average PSNR.

Table 1 shows that the algorithm using $T_{sad0} + T_{4\times4}$ 'misses' fewer MBs than the algorithm using T_{sad0} alone for all the video sequences, i.e. $T_{sad0} + T_{4\times4}$ provides a more accurate prediction than T_{sad0} only. For the 'Carphone' sequence, the maximum percentage of skipped MBs by TMN10 is 27.82% (from Table 2(a)). Hence, when the skipped MB target is set to 45% for 'Carphone', an extra 5224 MBs are skipped by $T_{sad0} + T_{4\times4}$. These extra skipped MBs cause a degradation of video quality. However, because our algorithms select MBs with low residual energy, the loss of video quality is minimized.

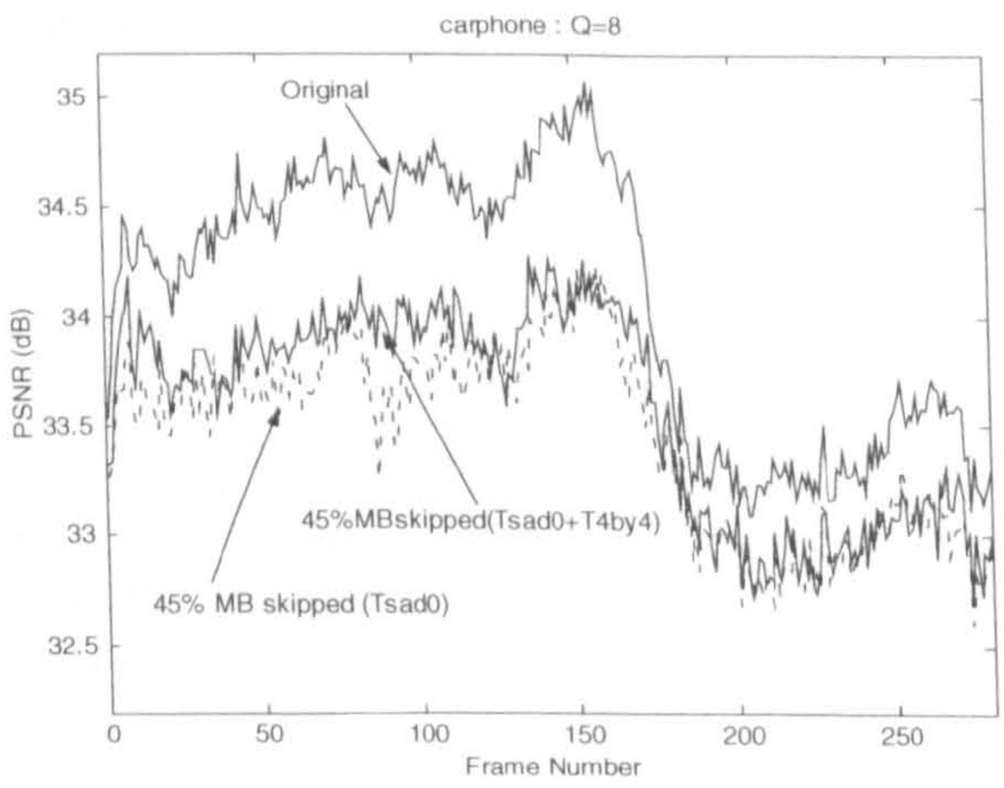


Fig. 9. PSNR against frame number for 'Carphone' with same complexity reduction by two types of classification algorithms.

4.2. Video quality

4.2.1. Objective quality

Fig. 9 shows the PSNR of each frame of 'Carphone' with $Q = 8$, 45% skipped MBs and fixed thresholds T_{sad0} and $T_{4\times4}$. The thresholds were chosen to give a 45% mean reduction in computational complexity (i.e. 45% skipped MBs) in each case. This figure shows that $T_{sad0} + T_{4\times4}$ consistently achieves better video quality than T_{sad0} alone, for a comparable reduction in computational complexity.

The 'Carphone' video sequence was coded using two types of MB classification algorithms. At each

fixed quantizer, T_{sad0} was varied from low to high to parameterize the performance of the algorithm at a range of computational complexity values. Fig. 10 plots the percentage of MBs classified as 'skipped' by our algorithm (x-axis) against the mean drop in PSNR compared with the same sequence encoded without complexity reduction (y-axis). This drop in PSNR is caused by occasional incorrect skipping of MBs that should have been encoded. As T_{sad0} increases, the graph moves to the right and up: more MBs are skipped but the PSNR drop increases.

This figure shows that classifying MBs using two thresholds, $T_{sad0} + T_{4\times4}$, produce better video quality than T_{sad0} alone. For example, using only T_{sad0} to classify MBs, the algorithm skips 40% of MBs at the expense of a PSNR drop of 0.5 dB (for a fixed quantizer step size $Q = 8$). Using $T_{sad0} + T_{4\times4}$, approximately 44% of MBs can be skipped without any further PSNR drop. When $Q = 16$, skipping approximately 46% of MBs produces a PSNR drop of 0.5 dB using only T_{sad0}; approximately 56% may be skipped for the same PSNR drop using $T_{sad0} + T_{4\times4}$.

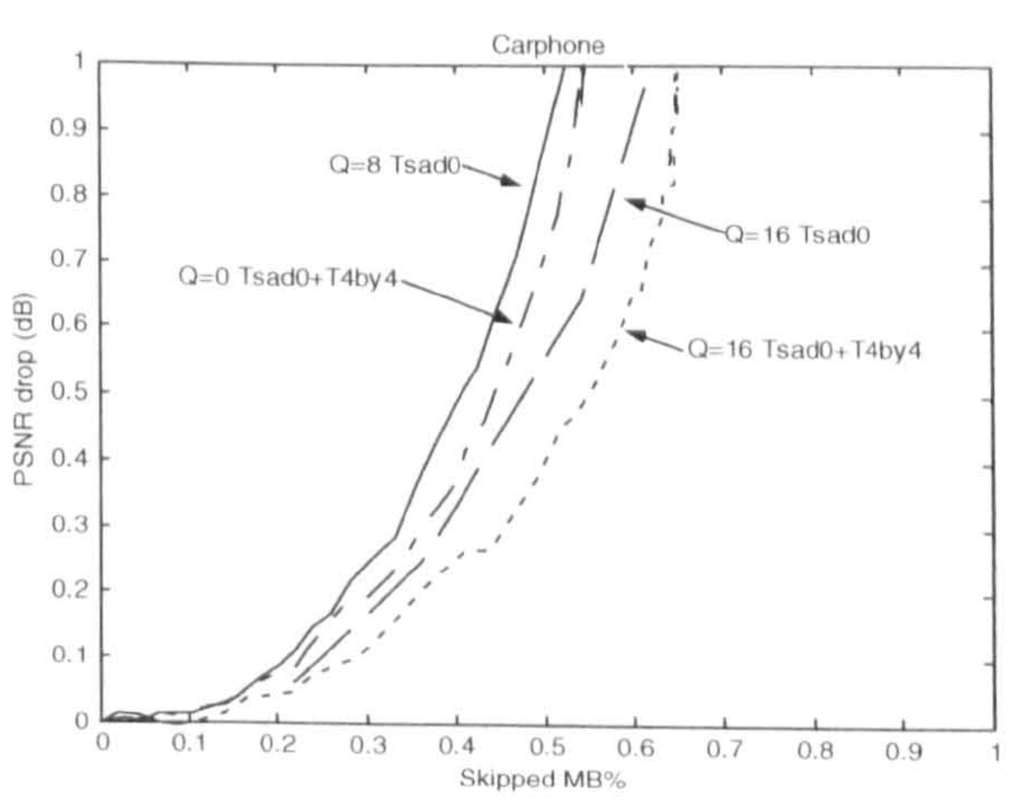


Fig. 10. Performance comparison of various Q and two types of classification algorithms.

4.2.2. Subjective quality

Fig. 11 compares frame 87 of the 'Carphone' sequence after encoding and decoding using three methods: (a) without any complexity reduction, (b) after skipping 45% of the MBs using T_{sad0} only and (c) after skipping 45% of the MBs using $T_{sad0} + T_{4\times4}$. Fig. 11(b) shows some visible degradation compared with Fig. 11(a), with blocking artefacts around the top of the head introduced by the complexity reduction algorithm (note that these artefacts occur at the boundary of a moving object, as reported in Section 3.3). Fig. 11(c) is very similar to Fig. 11(a) with little apparent increase in distortion. When played back at 30 frames per second, there is a slight degradation in sequence (b) compared with (a) but no obvious difference between (c) and (a). An interesting feature of the ($T_{sad0} + T_{4\times4}$) algorithm is that MBs classified as 'skipped' are typically in low-activity regions of the scene (such as the background of 'Carphone') and incorrectly skipped MBs in this region tend not to be noticed by the viewer. Hence the subjective quality is often better than the PSNR results might imply.

4.3. Rate-distortion performance

Fig. 12 plots the rate-distortion performance of the 'Carphone' sequence after encoding and

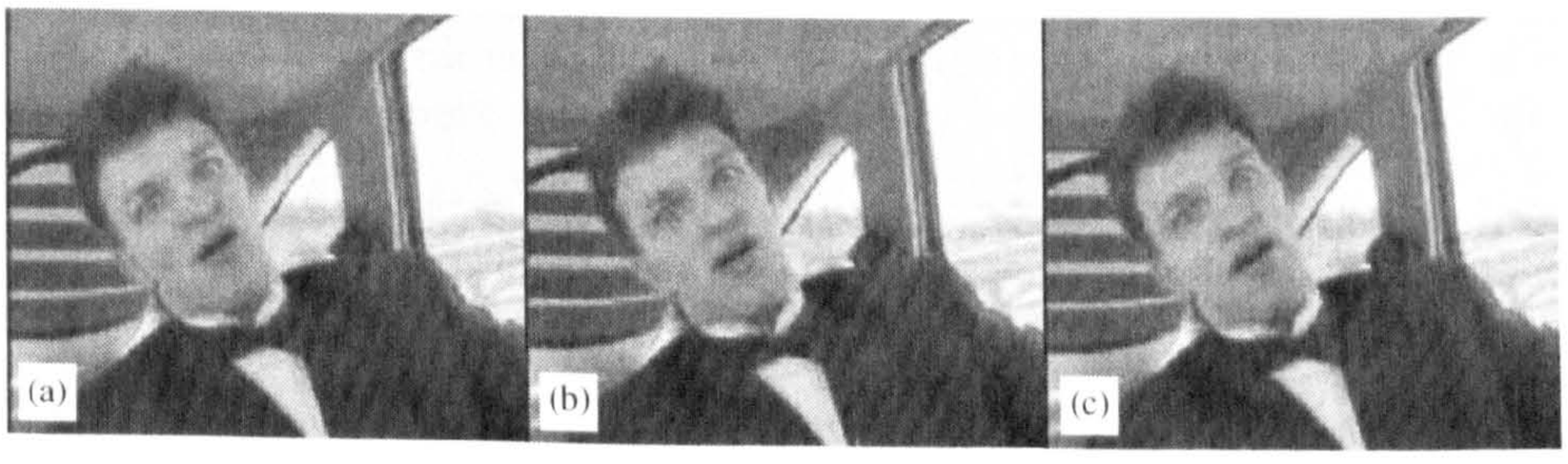


Fig. 11. Sample frame 87 of decoded 'Carphone' sequence with $Q = 8$ and 45% skipped MB: (a) original; (b) with T_{sad0} only; and (c) $T_{sad0} + T_{4\times4}$.

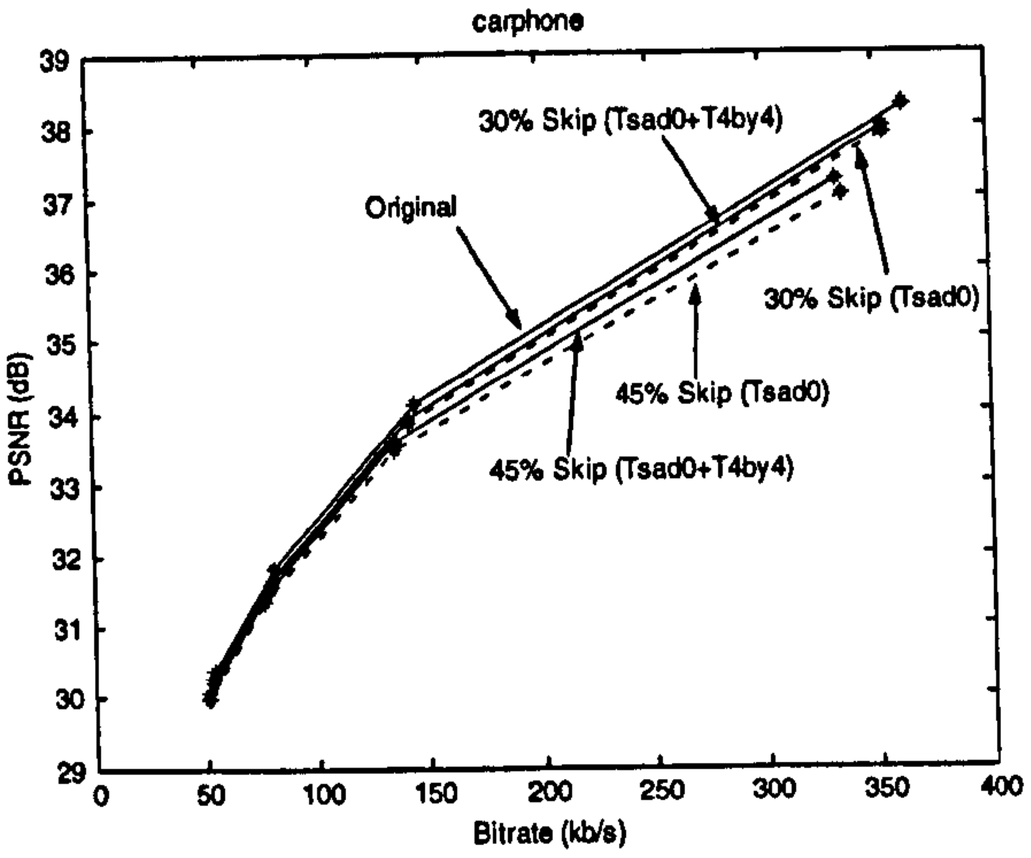


Fig. 12. Rate-distortion performance.

decoding using the H.263 TMN-8 reference model operating in Baseline mode (Original). Two methods (T_{sad0} only and $T_{\text{sad0}} + T_{4\times4}$) are used to achieve two target complexity reduction levels (30% skipped MBs and 45% skipped MBs) and the corresponding rate-distortion curves are plotted in Fig. 12. The ($T_{\text{sad0}} + T_{4\times4}$) algorithm outperforms the T_{sad0} algorithm at both complexity targets. Using the ($T_{\text{sad0}} + T_{4\times4}$) method we can reduce complexity by 30% with a very slight degradation in rate-distortion performance.

4.4. Computational complexity

The purpose of our proposed algorithm is to reduce the computational complexity during encoding and so it is important to identify any additional computational costs due to the classification algorithm. SAD0_{MB} is normally computed in the first step of any motion estimation algorithm and so there is no extra calculation required. Furthermore, the SAD values of each 4×4 block (A, B, C and D in Eq. (2)) may be calculated without penalty if SAD0_{MB} is calculated by adding together the values of SAD for each 4×4-sample sub-block in the MB.

The additional computational requirements of the classification algorithm are the operations in Eqs. (3)–(5). For each MB consisting of four 8×8 blocks, there are $12(=4 \times 3)$ additions, $24(4 \times 6)$ subtractions and 12 comparisons. This is negligible compared with the cost of motion estimation, DCT, etc. for each MB and so the computational saving of our proposed algorithm significantly outweighs the small computational cost of classifying MBs.

5. Conclusions

In this paper we have described an MB classification algorithm that has the potential to significantly reduce computational complexity in a software video encoder. The goal of this algorithm is to avoid expensive processing of MBs that are destined to be skipped during encoding, whilst minimizing distortion due to incorrect classification predictions. The initial SAD calculation SAD0_{MB} can give an approximate prediction of MB type (as reported in [13]) but incorrectly predicts non-skipped MBs that contain significant low-frequency AC coefficients after quantization. Adding a prediction of low-frequency residual energy improves the accuracy of MB classification, particularly for MBs at the edge of moving regions.

Experimental results show that our classification algorithm can deliver substantial computational savings (40–50% for a high-activity sequence such as 'Carphone', more for low activity sequences) with only a small reduction in rate-distortion performance. The reduction in subjective quality is actually smaller than the PSNR results imply, since our algorithm tends to classify skipped MBs in static, visually unimportant regions of the scene. The classification process is computationally simple and most of the operations may be integrated into the normal motion estimation process.

In this paper, the classification algorithm was applied to H.263 but it is equally applicable to MPEG-4 Simple Profile and other MB-based CODECs. It is planned to incorporate other classification methods with this algorithm (e.g. the DCT block classification algorithm described in [9]) in order to avoid unnecessary computation for MBs in other categories such as '$Q_{\text{Coeff}} = 0$' and 'MV = 0' (Fig. 1) and to examine

the effect of adding a control algorithm such as the feedback-based computational complexity control mechanism described in [9].

References

[1] M. Gallant, G. Côté, F. Kossentini, An efficient computation-constrained block-based motion estimation algorithm for low bitrate video coding, IEEE Trans. Image Process. 8 (12) (December 1999) pp. 1816–1822.

[2] A. Hossen, U. Heute, Fast approximate DCT, in: Proceedings of the ICASSP97, Munich, 1997.

[3] ISO/IEC 14496-2, Information technology—generic coding of audio-visual objects—Part 2: Visual, MPEG4, 1999.

[4] ISO/IEC JTC1/SC29/WG11, MPEG-4 video verification model version 18, January 2001.

[5] ITU-T Recommendation H.263, Video coding for low bit rate communication, 1998.

[6] J. Jain, A. Jain, Displacement measurement and its application in interframe image coding, IEEE Trans. Commun. COM-29 (December 1981) pp. 1799–1808.

[7] K.I.T. Koga, A. Hirano, Y. Iijima, T. Ishiguro, Motion-compensated interframe coding for video conferencing, in: Proceedings of the NTC 81, Vol. 28, New Orleans, December 1992.

[8] I. Rhee, Error control techniques for interactive low-bit rate video transmission over the internet, ACM SIGCOMM'98, Vancouver, Canada.

[9] I.E.G. Richardson, Y. Zhao, Adaptive management of video encoder complexity, J. Real-time Imag. 8 (4) (August 2002) 291–301.

[10] A. Skodras, Fast discrete cosine transform pruning, IEEE Trans. Signal Process. 42 (1994) pp. 1833–1838.

[11] M.-T. Sun, I.-M. Pao, Statistical computation of discrete cosine transform in video encoders, J. Visual Commun., Image Repr. 9 (1998) pp. 163–170.

[12] Video Coding Experts Group, ITU, Video codec test model, Version 10 (TMN10), April 1998.

[13] J.-F. Yang, S.-C. Chang, C.-Y. Chen, Computation reduction for motion search in low rate video coders, IEEE Trans. Circuits Systems Video Technol. 12 (10) (October 2002) pp. 948–951.

[14] Y. Zhao, I.E.G. Richardson, Computational complexity management of motion estimation in video encoders, IEEE DCC2002, Snowbird, UT.

Real-Time Imaging **8**, 291–301 (2002)
doi:10.1006/rtim.2001.0275, available online at http://www.idealibrary.com on IDEAL®

Adaptive Management of Video Encoder Complexity

In this paper, we investigate methods of reducing the computational complexity of the discrete cosine transform (DCT) in a software video encoder. The number of DCT calculations may be reduced by modeling the distribution of zero blocks. We demonstrate that the reduction in computational complexity is variable and depends on the statistics of the video sequence. We propose a new adaptive algorithm that can maintain a near-constant reduction in complexity. The proposed algorithm performs well at converging to a "target" computational complexity, at the expense of a small reduction in image quality. This algorithm provides a flexible mechanism for managing computational complexity in a video encoder.

Iain E.G. Richardson and Yafan Zhao

School of Engineering,
The Robert Gordon University,
Aberdeen AB10 1FR, UK
E-mail i.g.richardson@rgu.ac.uk

Introduction

There is a growing market for video communication applications that incorporate software-only implementations of popular video coding standards such as H.263 [1] and MPEG4 [2]. These include conferencing and streaming video applications on the PC platform as well as emerging applications for embedded processors and digital signal processors [3].

In many cases, the performance of these software-only CODECs is limited by available processing power as well as, or rather than, by available bandwidth. This is particularly true for applications using low-power embedded processors or in cases where the video CODEC must compete for processor resources with several other applications. It is therefore important to develop flexible methods of managing the computational complexity of video encoding and decoding.

In this paper, we take the following approach to support complexity management for real-time video encoding. The discrete cosine transform (DCT) [4] is identified as a computationally intensive function. Methods of predicting or modeling the output of the DCT (and therefore bypassing some computational steps) are compared. An adaptive algorithm is described and is shown to be suitable for dynamically controlling the complexity of the DCT (and related functions) to maintain a "target" level of computational complexity.

DCT Complexity

Popular video coding standards such as H.263 and MPEG4 make use of DCT encoding of motion-compensated residual frames. Within this type of video CODEC, encoding requires more processing power than decoding (due to the extra computation required for

1077-2014/02/ $35.00

motion estimation and reconstruction). In a software-based encoder, the computationally intensive operations include motion estimation and compensation, DCT, quantization and variable length encoding. For example, Table 1 summarizes the results of profiling a software H.263 encoder running on a Pentium III processor. The encoder uses the "fast" forward and inverse DCT algorithms described in [5] and a logarithmic search motion estimation algorithm with a search area of ± 7.5 pixels. In this example, the DCT, quantization, inverse DCT and inverse quantization steps take nearly 35% of the total processing resources. It is therefore useful to examine methods of reducing the computational complexity of the DCT and related functions.

Reducing DCT complexity

After motion compensation, the residual energy in many image blocks in a typical inter-coded video frame is low, such that most or all of the quantized coefficients are zero. In these cases it may not be necessary to transform and quantize all 64 coefficients. Several reduced-complexity DCT algorithms have been developed that reduce the number of arithmetic operations required to carry out the DCT [6–8]. These algorithms typically use "pruning" to calculate a subset of the complete 8×8 DCT (e.g. a 4×4 or 2×2 DCT). Applying a pruned DCT algorithm to all blocks in an image will, however, significantly reduce the quality of the decoded image.

A model of the distribution of quantized DCT coefficients in a block may be used to predict the likely position of the highest non-zero coefficient (the end of block position or EOB). Reduced complexity approximations to the DCT (and quantize) operations can be applied to blocks where the model predicts a low value of EOB. In [9], the quantization parameter selected for each image block is used to predict the maximum value of EOB with a certain confidence (e.g. 95%, 98%, etc.). A reduced-complexity DCT is applied if the predicted EOB is less than a threshold. This penalizes those blocks that do not fit the model (i.e. where the quantizer is high but there are a significant number of non-zero coefficients). These blocks tend to contain image detail and so this method is likely to degrade the quality of "active" areas of the video scene. This model is extended in [10] to include a measure of macroblock energy in the model (the minimum mean absolute error, MMAE). The predicted EOB is compared with pre-determined threshold values in order to choose either a full DCT, a reduced-complexity DCT or no DCT. Annex P of MPEG4 (Video) [2] suggests comparing MMAE with a threshold and encoding each block with either a full DCT or no DCT. In [11] a general model is applied to each of the DCT inputs to estimate whether some or all of the output coefficients are zero. This model requires additional computation to classify the inputs, which may offset the computational savings due to a reduced-complexity DCT.

Table 1. Profile of software H.263 encoder

Functions	Processor utilization (%)
DCT, IDCT, Quant and Dequant	34.8
Motion estimation (integer and half-pixel)	26.9
Remaining functions	38.3

Predicting end of block

An H.263 video encoder carries out motion estimation to select the "best" matching region from a reference frame (typically the previous reconstructed frame) for each macroblock. In practice, this selection is usually based on the minimum sum of absolute differences (SAD) or MMAE between the 16×16 luminance samples in the current macroblock and a neighboring 16×16 region in the luminance component of the reference frame. SAD is given by

$$SAD_{MB} = \sum_{i=0}^{15} \sum_{j=0}^{15} |C(i,j)| \quad (1)$$

where $C(i,j)$ are the residual luminance samples after motion compensated prediction from the reference frame.

The luminance SAD for the current macroblock (SAD_{MB}) may be used to model or predict the highest non-zero DCT coefficient after quantization [9, 10]. The position of the highest non-zero coefficient (the end of block or EOB) depends on the energy of the DCT coefficients. This in turn depends on the energy of the residual image samples after motion compensation, which is approximated by SAD_{MB}.

Annexe P of [2] recommends using the SAD of the current 8×8 block as a predictor for EOB. Block SAD (SAD_B) is given by:

$$SAD_B = \sum_{i=0}^{7} \sum_{j=0}^{7} |C(i,j)| \quad (2)$$

SAD_B for each luminance block may readily be obtained in a software video encoder without extra computation, since SAD_{MB} may be calculated from the sum of SAD_B values for each block. Additional computation is required to calculate SAD_B for the chrominance blocks in the macroblock since these values are not normally calculated during Motion Estimation: we investigate whether this extra computation is worthwhile in Section 5 of this paper.

Figure 1 compares the performance of the two predictors (SAD_{MB}, and SAD_B). The "Carphone" video sequence was encoded using an H.263 software encoder (with a fixed quantizer step size $Q=8$) and the values of EOB, SAD_{MB} and SAD_B were recorded for each block of the encoded sequence. The lines labeled "$P(EOB=0)$" plot the probability of zero EOB against SAD. (Note: SAD_{MB} is normalized by dividing each value by 4, since SAD_{MB} is the sum of 256 difference values whereas SAD_B is the sum of 64 values.) There is a higher probability that EOB for the current block is zero for lower values of SAD. Some examples are listed in Table 2 and highlighted here. For example, if SAD_B for the current block is 200, there is a 90% probability that EOB will be zero, i.e. that there will be no non-zero coefficients remaining after DCT and quantization. If SAD_{MB} (normalized) for the current block is 200, there is a 55% probability that EOB will be zero. The lines labeled "CDF" plot the cumulative density function of SAD_{MB} and SAD_B: these plots show that the distributions of SAD_{MB} and SAD_B are approximately the same. For example, $SAD \leq 200$ for approximately 40% of image blocks. The encoder may reduce computational complexity by choosing not to calculate the DCT for blocks with $SAD<200$. If SAD_{MB} is used as a predictor, computational complexity will be reduced by 40% (since 40% of blocks fall into this category) but the decoded image will be distorted due to the 45% error in predicting blocks with zero EOB. If SAD_B is used as a predictor, the complexity will once again be reduced by 40%, with lower distortion of the decoded image due to the 10% error in prediction. This indicates that SAD_B gives a more accurate prediction of EOB than SAD_{MB}. This is because the number of non-zero quantized coefficients in an individual block depends on the energy in that block and SAD_B gives a more accurate measure of block energy than SAD_{MB} (which is proportional to the average energy across all four luminance blocks in the current macroblock).

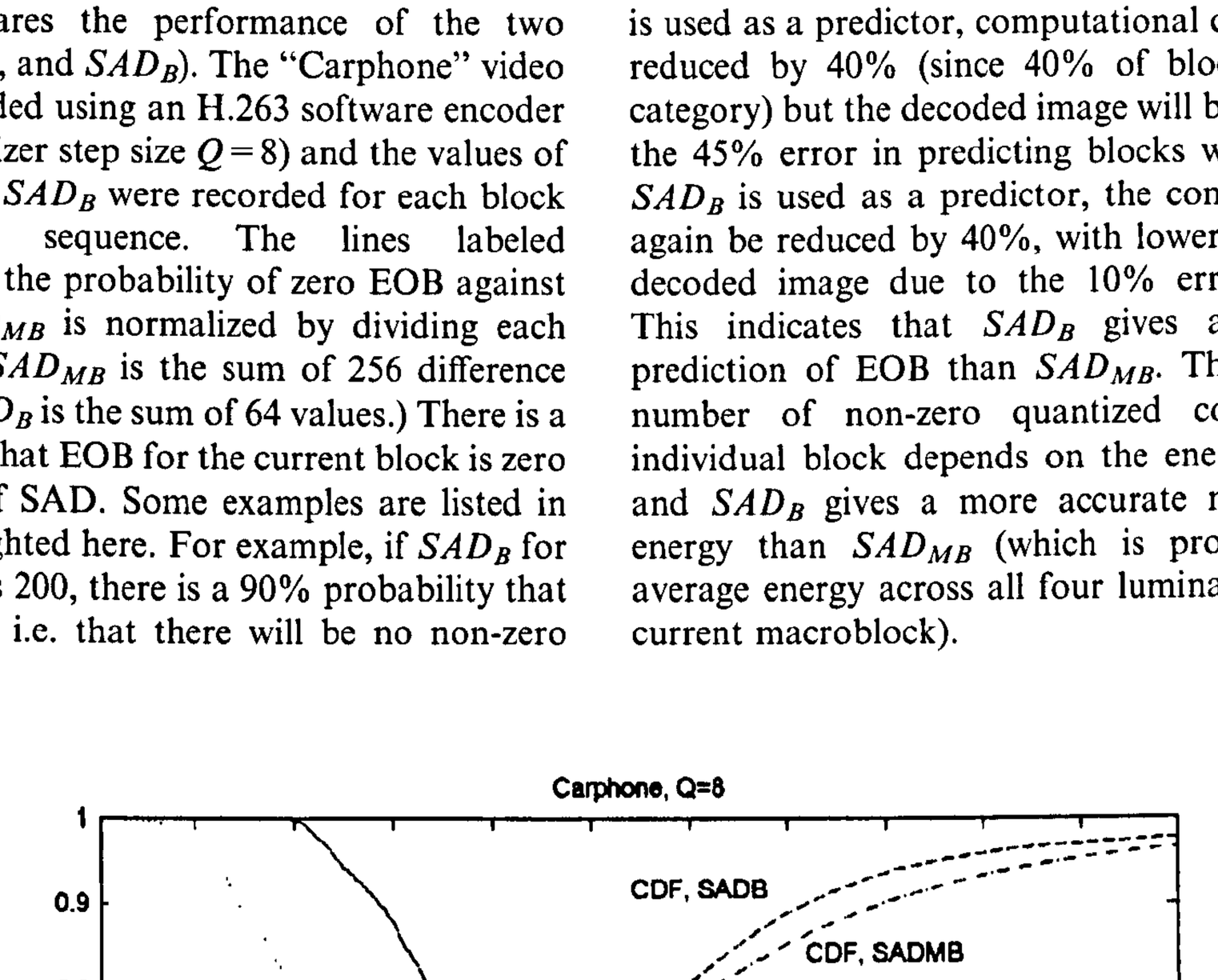


Figure 1. Probability of zero EOB and cumulative distribution (CDF) (Carphone, $Q=8$).

Table 2. EOB probability and block distributions, Carphone, $Q=8$

SAD threshold	$P(EOB=0)$ (%)	Proportion of blocks (%)
$SAD_B=200$	90	40
$SAD_{MB}=200$	55	40
$SAD_B=250$	70	60
$SAD_{MB}=250$	20	60

Decision thresholds

Figure 2 plots $P(EOB=0)$ for the Carphone sequence with varying quantizer. The value of SAD_B at which a given $P(EOB=0)$ occurs is approximately linearly related to quantizer step size Q, i.e.

$$P(EOB=0) \approx \frac{SAD_B}{Q} \tag{3}$$

In order to determine whether the DCT should be calculated for a block, SAD_B may be compared with a threshold T_0, as described in Annex P of [2]:

$$DCT\_Flag = (SAD_B/Q < T_0 ? 1 : 0) \tag{4}$$

If *DCT_Flag* is 1, the "full" DCT and quantization are calculated for the current block. If *DCT_Flag* is 0, all the quantized DCT coefficients are set to zero (i.e. the DCT and quantize operations are skipped).

A lower value of the threshold T_0 will tend to give a more accurate prediction of zero EOB. For example, if the threshold T_0 is set at about 25, EOB will be correctly predicted for approximately 90% of blocks. A higher value of T_0 should give lower computational complexity (i.e. fewer DCTs are calculated) at the expense of poorer prediction accuracy.

Variation in Computational Complexity

The sequences "Carphone", "Claire" and "Mother and Daughter" were encoded using the threshold algorithm (Eqn (4)) with a fixed threshold $T_0=30$ and a fixed quantizer step size $Q=8$. Figure 3 shows the computational complexity of the DCT calculations (compared with encoding every block using the "full" DCT) for the first 200 frames of the sequences "Carphone", "Claire" and "Mother and Daughter". There is a clear variation between sequences: in general, for a given value of SAD,

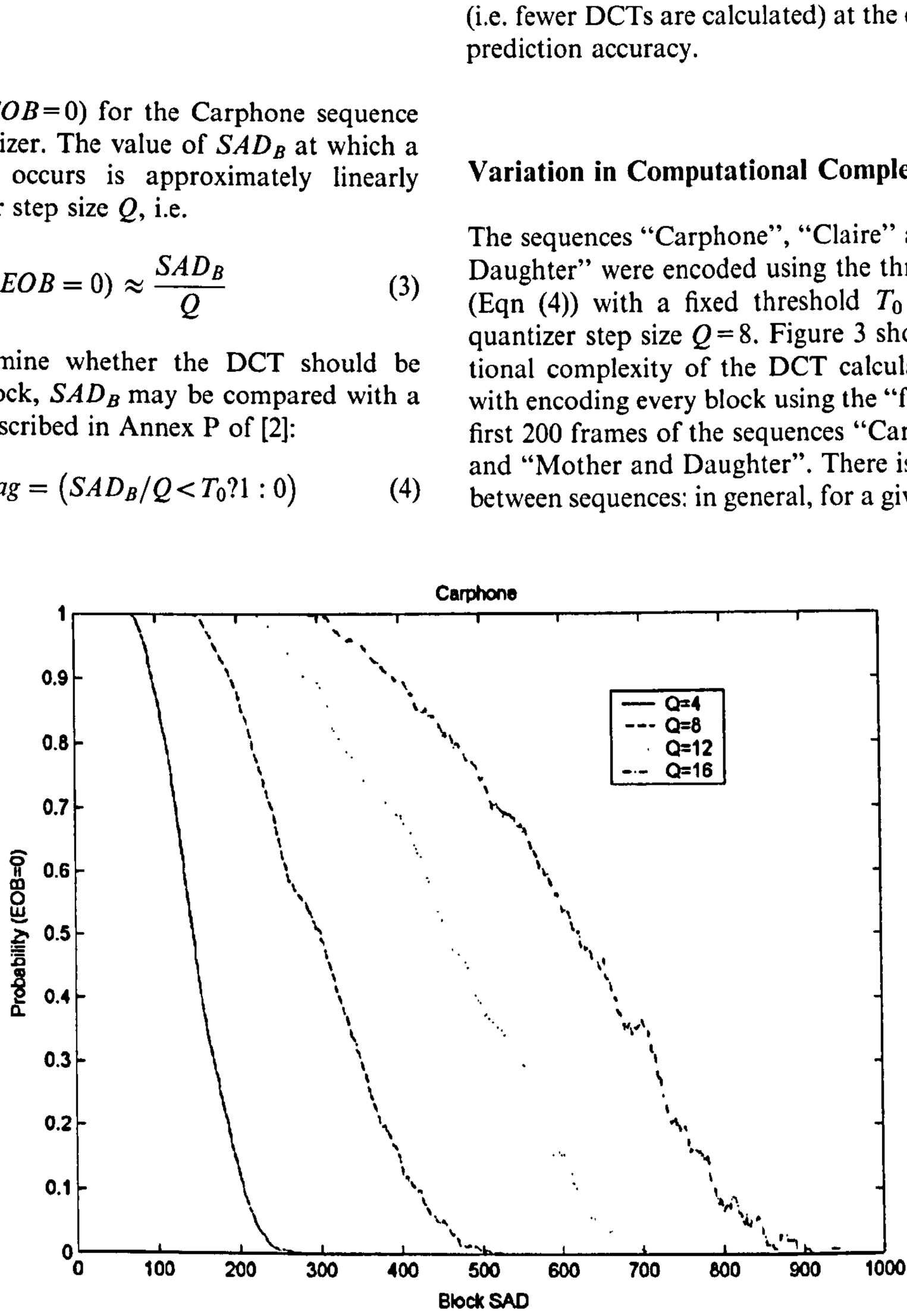


Figure 2. Probability of zero EOB ("Carphone", varying Q) (—) Q=4, (- - -) Q=8, (····) Q=12, (-···) Q=16.

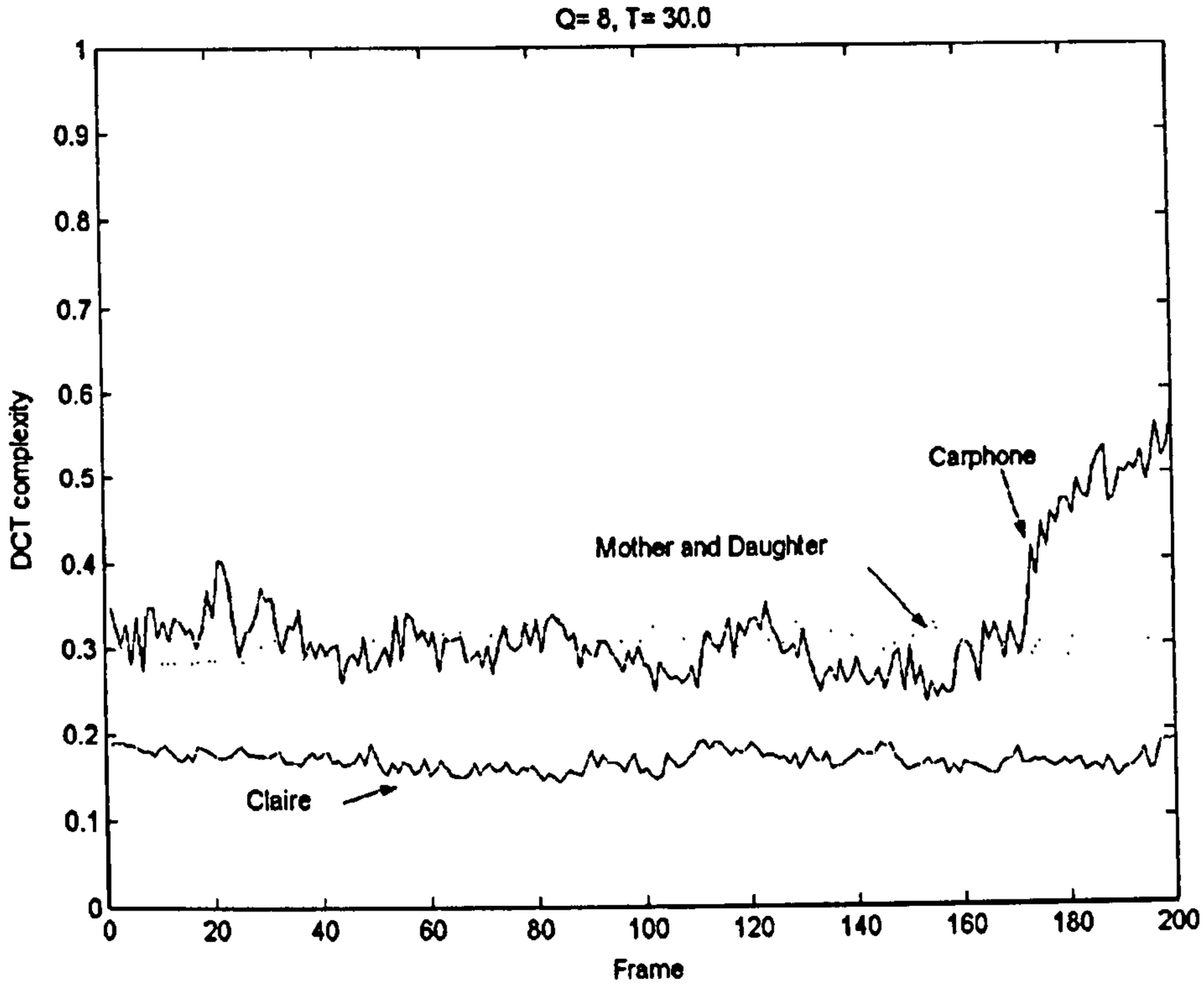


Figure 3. Variation in DCT complexity, $T_0 = 30$ (fixed).

there is a higher probability that $EOB = 0$ for a sequence with low activity (such as Claire) than for a sequence with high activity (such as Carphone) and this produces a variation in the number of blocks "skipped" by the threshold algorithm. There is also a variation in DCT complexity during each sequence (for example, in the later part of "Carphone").

With a fixed threshold, the computational complexity of the DCT (and also the prediction error and hence the PSNR) varies depending on scene content. These variations make it difficult to predict or manage the processor utilization required to carry out the DCT. A more useful approach is to maintain a predictable level of computational complexity that does not depend on the content of the video sequence.

The sequences "Carphone", "Claire" and "Mother and Daughter" were encoded with a range of threshold values T_0 and quantizer step sizes Q. Figure 4 plots the relationship between threshold (T), quantizer step size (Q) and computational complexity of the DCT and quantizer functions (C). The computational complexity C varies with T_0 and Q: larger values of T_0 and Q tend to reduce C and vice versa. The shape of the surface is similar for the three sequences, but the actual values of C vary between the sequences. Figure 4 indicates that it should be possible to control the DCT complexity C by varying T_0 and/or Q. Note that a wider dynamic range of C may be obtained by varying T_0 (with fixed Q) than by varying Q (with fixed T_0).

Figure 5 shows the mean bitrate of each of the coded sequences described above. It is clear that T_0 has a limited effect on the coded bitrate (for a fixed value of Q): higher values of T_0 produce lower bitrates, since fewer blocks are actually coded. However, the effect is small compared with the effect of quantizer step size Q on bitrate.

Figure 6 plots DCT complexity (C) against ($1/TQ$) for each of the coded sequences above. This figure shows that C is approximately linearly related to $1/TQ$:

$$C \approx \frac{v}{T_0 Q} \tag{5}$$

The term v varies with the content of the video sequence. It should be possible to vary the threshold T_0 in order to

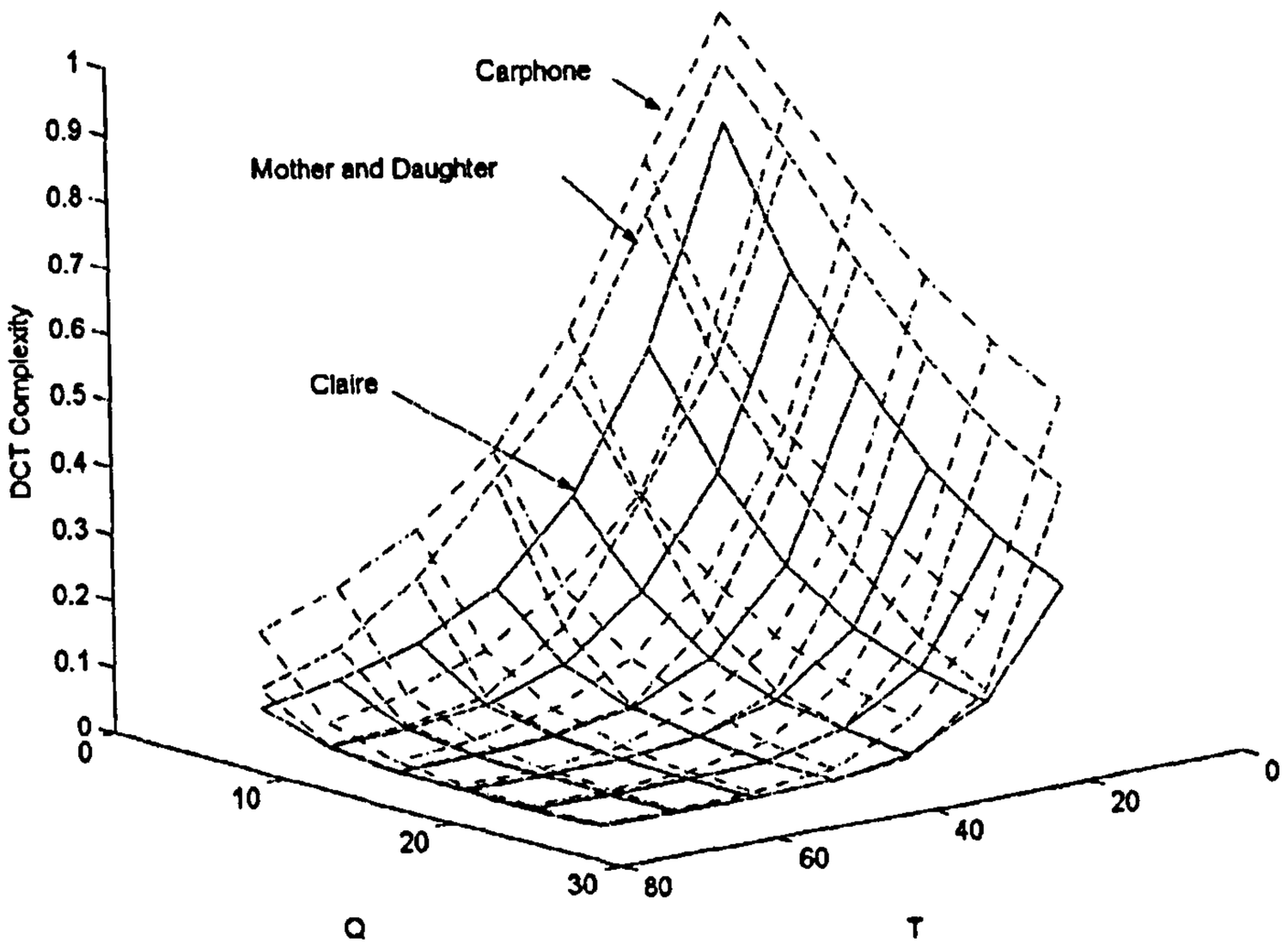


Figure 4. T_0, Q, DCT complexity surfaces.

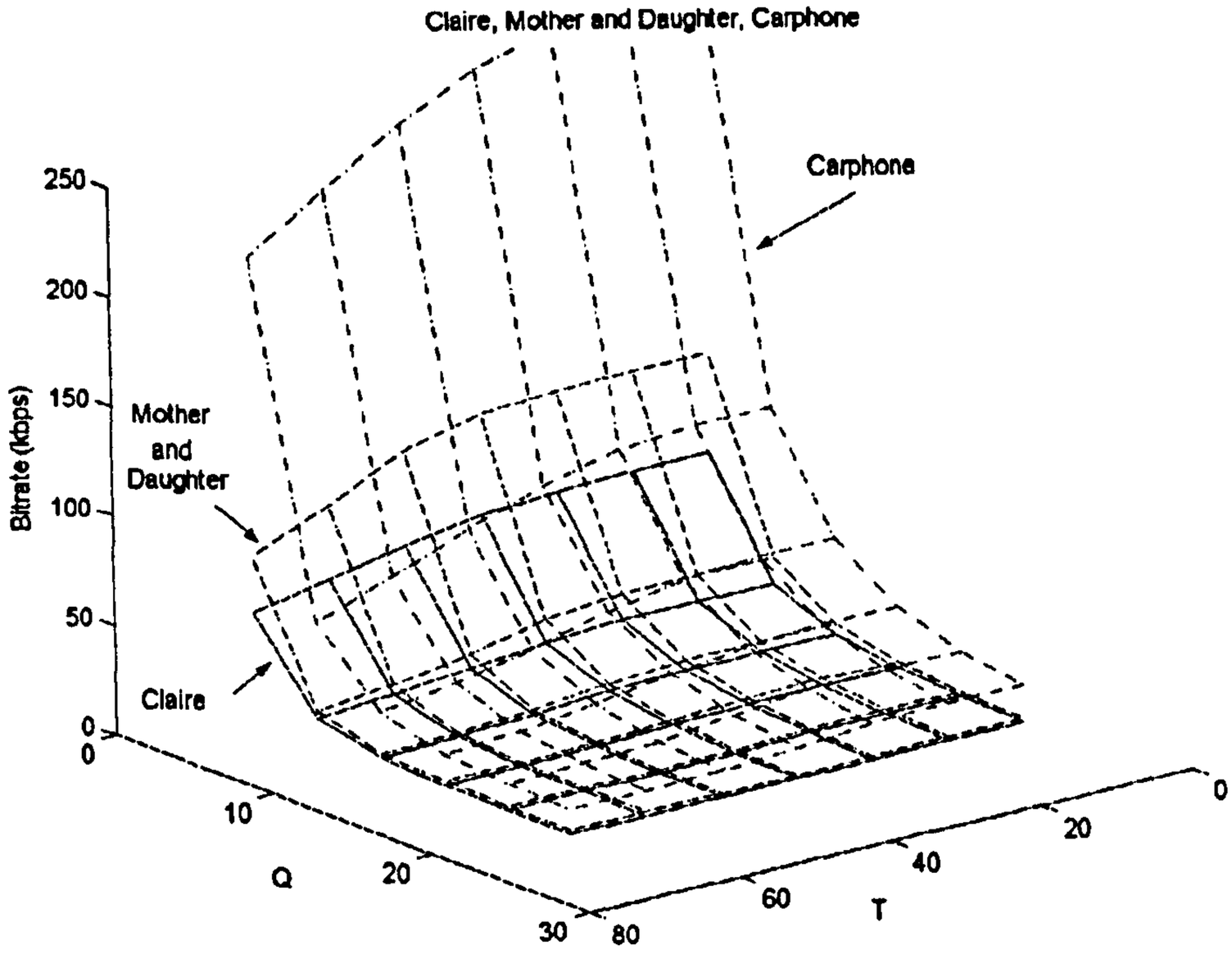


Figure 5. T_0, Q, rate surfaces.

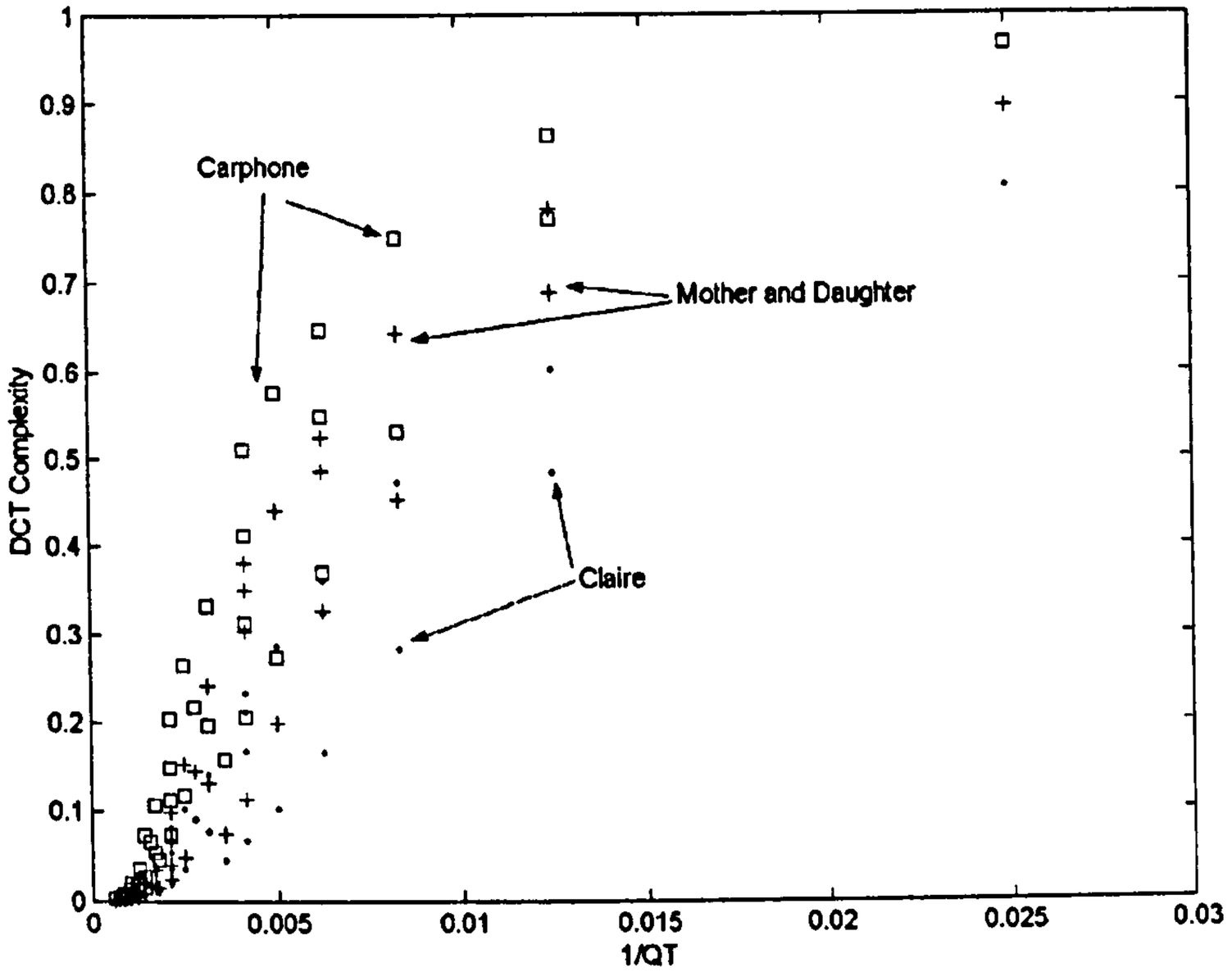


Figure 6. Plot of DCT complexity vs. $1/QT_0$.

compensate for the variation in video scene content in order to maintain a near-constant reduction in complexity.

Adaptive Modeling to Maintain Constant Complexity

With a fixed threshold, the computational complexity tends to vary gradually and so it should be possible to update the threshold T_0 based on the measured computational complexity C of one or more recently encoded frames. We propose to update T_0 after each encoded frame in order to maintain a near-constant "target" computational complexity C_t. Modifying T_0 once per frame should be sufficient for the purpose of managing processor resources: a practical video encoding application will usually buffer one or more frames of coded data prior to transmission and so a short-term variation in computational complexity during the encoding of a frame is not likely to be significant. Of greater importance is the computational complexity over a longer period. The goals of an adaptive algorithm should therefore be to:

1. Maintain a mean target complexity C_t.
2. Minimize the variation in frame complexity C_n.
3. Minimize distortion.

After encoding frame n, the actual value of v can be calculated. Rearranging Eqn (5):

$$v = C_n Q_n T_{0n} \tag{6a}$$

where T_{0n} and Q_n are the threshold and mean quantizer for frame n and C_n is the measured DCT complexity of frame n (i.e. the proportion of blocks for which the DCT and quantize functions were calculated). The threshold $T_{0(n+1)}$ required to achieve the target complexity C_t for frame $n+1$ can be calculated from Eqn (4) (assuming that v does not vary significantly between frame n and frame $n+1$):

$$T_{0n+1} = \frac{v}{C_t Q_{n+1}} \tag{6b}$$

where Q_{n+1} is the quantizer step size (at the frame level) chosen for frame $n+1$.

Combining Eqns (6a) and (6b):

$$T_{0n+1} = T_{0n} \frac{Q_n}{Q_{n+1}} \frac{C_n}{C_t} \tag{7a}$$

Rearranging yields:

$$T_{0n+1} = T_{0n}\frac{Q_n}{Q_{n+1}}\left(1+\frac{C_n - C_t}{C_t}\right) \quad (7b)$$

Eqn (7b) contains a term proportional to the previous value of T_0 and a term proportional to the error between the actual and target computational complexity ($C_n - C_t$). The sensitivity of the update equation to this error term may be varied by scaling the error term with a factor k:

$$T_{0n+1} = T_{0n}\frac{Q_n}{Q_{n+1}}\left(1+\frac{C_n - C_t}{kC_t}\right) \quad (8)$$

A small value of k means that the update is sensitive to small errors ($C_n - C_t$): it should therefore converge rapidly to a target complexity C_t at the expense of possible instability. A large value of k should give better stability but slower convergence.

Results

Convergence, stability and quality

The sequences "Carphone", "Claire" and "Mother and Daughter" were encoded using the adaptive update algorithm described by Eqn (8), with a range of quantizer values Q, convergence factors k and target DCT complexities C_t. Q was kept constant throughout each sequence. This means that the term Q_n/Q_{n+1} in Eqn (8) becomes unity, making it possible to study the relationship between complexity and T_0. The threshold T_0 was set to an arbitrary initial value of 30 in each case.

Figure 7 plots the variation of DCT complexity C, threshold T_0 and peak signal to noise ratio (PSNR) for the Mother and Daughter sequence with $Q=8$. The left-hand column of Figure 7 shows the variation of DCT complexity (C) (i.e. the proportion of blocks for which the DCT is calculated). When $k=2$, the complexity quickly converges to the target value. However, the update algorithm is clearly unstable for $C_t=0.1$ (i.e. a target complexity of 10% of the "full" DCT complexity). When $k=10$, the algorithm is stable but converges slowly to the target C_t. A good compromise is given by $k=6$: the algorithm remains stable for $C_t \geq 0.1$ and maintains the target complexity despite changes in sequence characteristics. The center column of Figure 7 shows the variation of the threshold T_0. For $k=6$, the threshold varies gradually as the content of the sequence changes. The right-hand column of Figure 7 shows the PSNR of each frame of the sequence. For $k=6$, there is a small drop in PSNR when $C_t=0.5$ (not shown) and a slightly larger drop when $C_t=0.3$. PSNR drops more significantly when $C_t=0.1$: however, the video sequence is still clearly recognizable.

Similar results were obtained for the other sequences (Carphone and Claire). Convergence factor $k=6$ gives acceptable performance for all sequences and all values of Q. Table 3 summarizes the results for each sequence with a convergence factor $k=6$. This table shows that the adaptive algorithm maintains the target complexity with only a small drop in PSNR for $C_t>0.1$. The drop in PSNR is highest for the "Carphone" sequence. This is because "Carphone" contains more motion and detail than the other two sequences and hence it is more difficult to achieve a target complexity of $C_t=0.1$ (i.e. to discard 90% of blocks) without a significant loss of image quality. The standard deviation of the measured DCT complexity ("STD Complexity") indicates the stability of the update algorithm and these results show that the algorithm remains stable in each measured case. "Convergence" is defined as the number of frames that are encoded before the algorithm reaches the target complexity (within $\pm 10\%$ of C_t) and this occurs within 23 frames in the worst case.

Profiling

The actual processor utilization was measured by profiling an H.263 encoder (based on H.263 TMN-5) that incorporates the adaptive update algorithm (Eqn (8)). The encoder uses a logarithmic search motion estimation algorithm with a search window of ± 7.5 luminance samples and the Forward and Inverse DCTs are calculated using the algorithm described in [5]. Figure 8 shows the results for 200 frames of the "Mother and Daughter" sequence, with $k=6$ and $Q=8$. Only the 5 most computationally intensive functions are listed. "SAD_Macroblock" is the function that calculates SAD for integer-accuracy motion estimation and "FindHalf-Pel" calculates the half-pixel component of the motion vector.

The black bars show the result when a full FDCT calculation is carried out for each block: the DCT is clearly the most computationally intensive function. The "Model_Y" bars show the result of applying the proposed adaptive threshold algorithm to the luminance blocks only, with a target complexity of 30%. The algorithm successfully reduces the number of calculations for luminance blocks by around 70% but the total

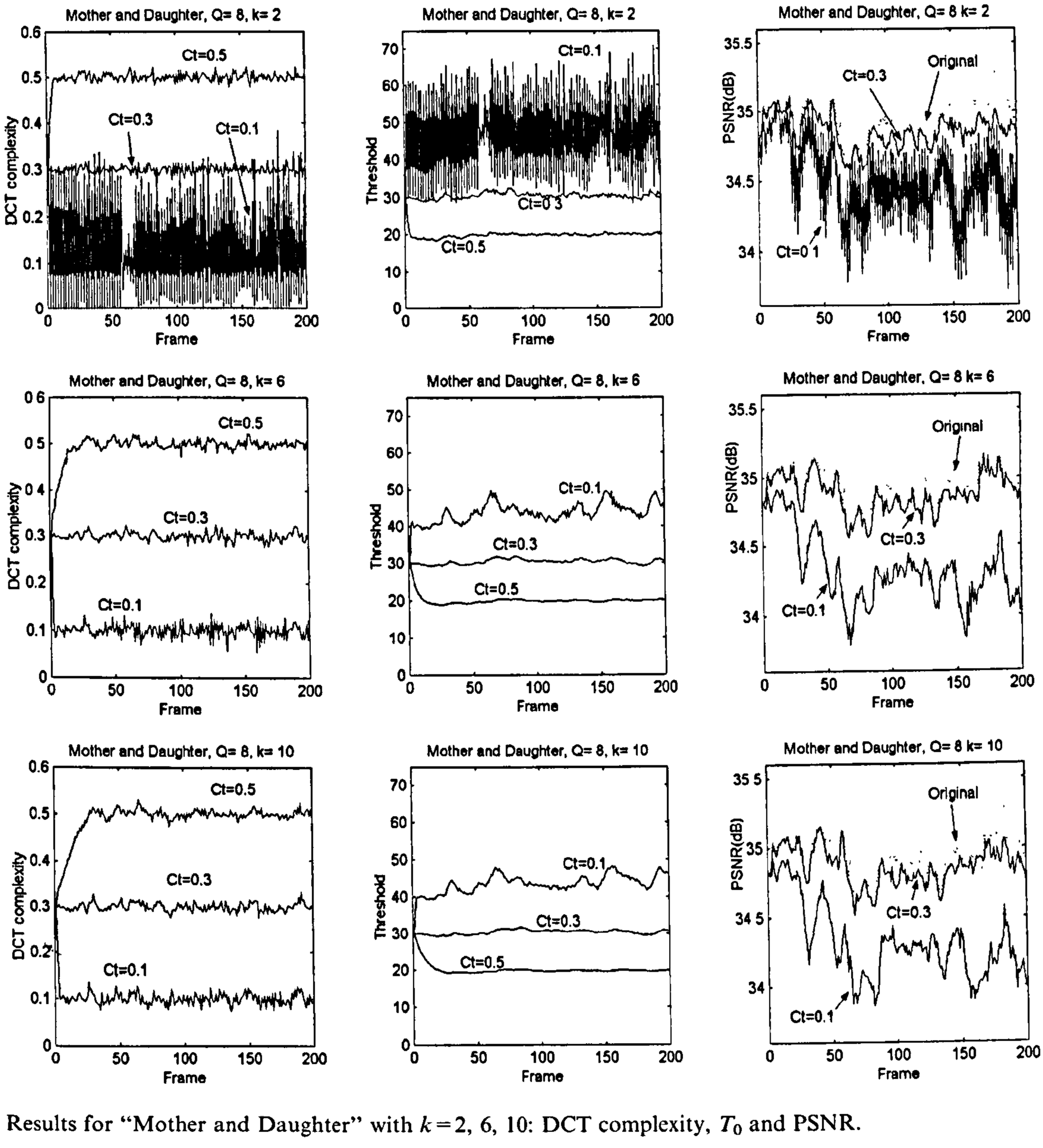


Figure 7. Results for "Mother and Daughter" with $k=2$, 6, 10: DCT complexity, T_0 and PSNR.

computation time for the forward DCT (and also the quantizer function) is only reduced by about 47% because DCT and quantization are still calculated for every chrominance block.

The "Model_YCrCb" bars show the results when the adaptive algorithm is applied to luminance and chrominance blocks. There is a small extra computational overhead because SAD_B is calculated for the Cr and Cb blocks in each macroblock (this value is not normally calculated during motion estimation). However, the results clearly show that the complexity reduction outweighs this extra overhead and the encoder achieves around 69% reduction in complexity of the DCT and quantizer functions.

Discussion and Conclusions

The results described in this paper indicate that the computational complexity of the DCT and quantizer functions in a DCT-based video encoder can be successfully controlled by comparing block SAD with a threshold in order to determine whether to encode

Table 3. Adaptive threshold performance ($k = 6$)

Sequence	C_t	Q	Mean DCT complexity	STD DCT complexity	Mean PSNR drop (dB)	Convergence (frames)
Claire	0.1	8	0.1009	0.0113	0.4300	3
		16	0.0997	0.0087	0.0161	7
	0.3	8	0.2968	0.0137	0.0276	17
		16	0.2928	0.0368	0.0	21
	0.5	8	0.4883	0.0467	0.0026	22
		16	0.4792	0.0760	0.0	23
Carphone	0.1	8	0.1051	0.0277	1.8294	4
		16	0.1027	0.0121	0.7028	2
	0.3	8	0.3044	0.0247	0.2411	10
		16	0.3077	0.0281	0.0087	10
	0.5	8	0.5007	0.0322	0.0227	8
		16	0.4952	0.0576	0.0007	16
Mother and daughter	0.1	8	0.1017	0.0202	0.6269	5
		16	0.1002	0.0128	0.1475	5
	0.3	8	0.3002	0.0093	0.0772	1
		16	0.2965	0.0353	0.0019	10
	0.5	8	0.4942	0.0285	0.0384	18
		16	0.4872	0.0068	0.0088	22

each block. A fixed decision threshold gives unpredictable performance results, with variable complexity and quality depending on the time-varying content of the encoded video sequence. The adaptive threshold update algorithm presented here enables the encoder to maintain a near-constant target level of computational complexity. This algorithm performs well, enabling the encoder to reduce the complexity of the DCT and quantizer functions by up to 70–90% with minimal quality loss.

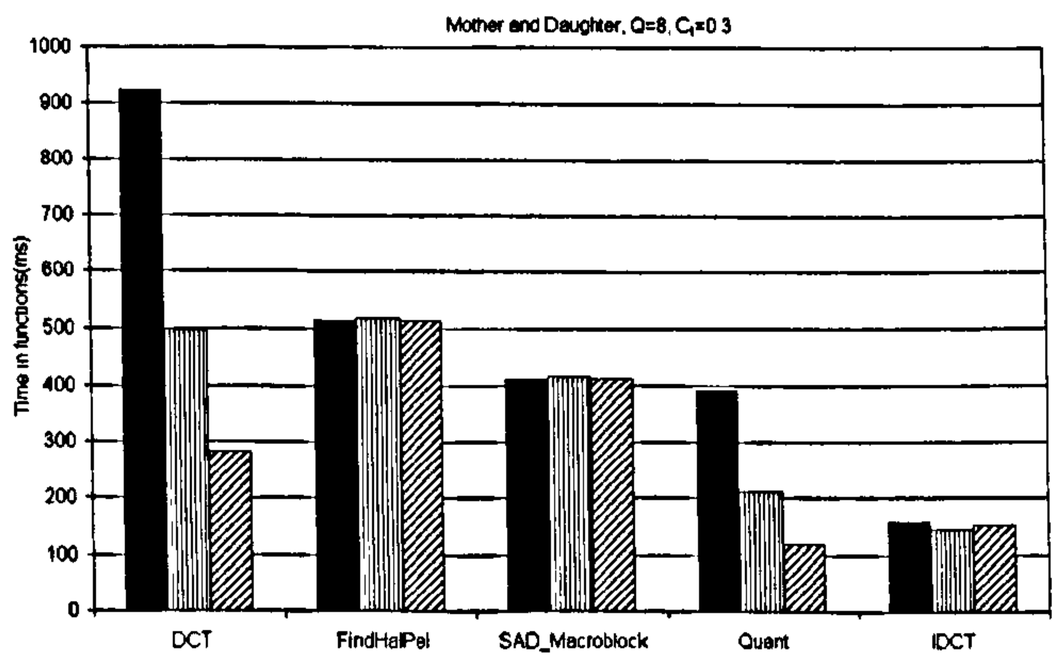


Figure 8. Result of profiling: "Mother and Daughter", Pentium III processor (■) Full, (▥) Model_Y, (▨) Model_YCrCb.

This adaptive algorithm may be a suitable component of a complexity control system in which the computational overhead of a video CODEC is managed in a similar way to the bit-rate control algorithm commonly used in a video encoder. "Coarse" complexity management may be achieved by skipping (i.e. not encoding) frames when there are insufficient processing resources available. We propose that "fine" complexity management (in which encoder complexity is controlled without dropping frames) may augment this. The aim of such a complexity control scheme should be to provide flexible, accurate control of computational overhead whilst minimizing variation in video quality. The adaptive threshold algorithm described here could form a suitable basis of such a scheme.

It may be worth investigating improvements to the adaptive update algorithm. The algorithm described here calculates the new threshold based on the results for the previous encoded frame and it may be possible to achieve better stability by examining several preceding frames. The variation in the proportionality factor v (Eqn (5)) appears to be related to the content of the video sequence (for example, v is highest for "active" sequences such as "Carphone") and the accuracy of the threshold update could perhaps be improved by incorporating a measure of scene activity such as SAD.

The choice of T_0 and Q affects the coded bit rate (as well as the computational complexity), as shown in Figure 5. Bit-rate control is an essential requirement of many video coding applications and it would be useful to extend the approach described here to provide joint bit rate and complexity control.

Finally, this approach may be extended to other computationally intensive functions in a video CODEC. For example, the variable complexity motion estimation algorithm described in [12] is reported to reduce the computation of motion estimation by a varying amount depending on a control parameter and on the statistics of the video sequence. It may be that our adaptive technique could also be applied to a variable-complexity motion estimation algorithm to provide flexible, predictable management of computational complexity.

References

1. ITU-T Recommendation H.263. (1998) Video Coding for Low Bit Rate Communication.
2. ISO/IEC 14496-2. (1999) Information Technology — Generic Coding of Audio-Visual Objects — Part 2: Visual (MPEG4).
3. Richardson, I. & Kipperman, K. (1999) Video Coding using Digital Signal Processors. *Proceedings of the DSP World Design Conference*, Orlando, November.
4. Rao, K. & Yip, P. (1990) *Discrete Cosine Transform: Algorithms, Advantages and Applications*. New York: Academic Press.
5. Chen, W., Smith, C. & Fralick, S. (1977) A Fast Computational Algorithm for the Discrete Cosine Transform. *IEEE Transactions on Communications* **COM-25**: 1004–1009.
6. Wang, Z. (1991) Pruning the Fast Discrete Cosine Transform. *IEEE Transactions on Communications* **39**: 640–643.
7. Skodras, A. (1994) Fast Discrete Cosine Transform Pruning. *IEEE Transactions on Signal Processing* **42**: 1833–1837.
8. Hossen, A. & Heute, U. (1997) Fast Approximate DCT. *Proceedings of the ICASSP97*. Munich.
9. Pao, I.-M. & Sun, M.-T. (1998) Approximation of Calculations for Forward Discrete Cosine Transform. *IEEE Transactions on Circuits and Systems for Video Technology* **8**: 264–268.
10. Pao, I.-M. & Sun, M.-T. (1999) Modeling DCT Coefficients for Fast Video Encoding. *IEEE Transactions on Circuits and Systems for Video Technology* **9**: 606–616.
11. Lengwehasatit, K. & Ortega, A. (1998) DCT Computation Based on Variable Complexity Fast Approximations. *Proceedings of the ICIP98*.
12. Gallant, M., Côté, G. & Kossentini, F. (1999) An Efficient Computation-Constrained Block-Based Motion Estimation Algorithm for Low Bit Rate Video Coding. *IEEE Transactions on Image Processing* **8**(12): 1816–1822.