TRANSCODING OF MPEG-4 COMPRESSED VIDEO

HongQuan Chen

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

November 2003

Your file Votre référence
ISBN: 0-612-91009-1
Our file Notre référence
ISBN: 0-612-91009-1

ABSTRACT

Transcoding of MPEG-4 Compressed Video

HongQuan Chen

MPEG-4 is an ISO/IEC standard, developed by the MPEG (Moving Picture Experts Group). The transmission of MPEG-4 compressed video over channels with different capacities may require a reduction in bit rate if the transmission media has a lower capacity than the capacity required by the bitstream. There are many different approaches to this problem of bit rate conversion. A feasible and effective method is transcoding. The emphasis in this thesis is on transcoding of MPEG-4 compressed video both in the pixel domain and in the DCT domain. In the pixel domain, (joint) transcoding of MPEG-4 compressed video with drift error correction is discussed in this thesis. In the DCT domain, a new DCT Coefficient Translation and Truncation Transformation Matrix (DCTTTM) based motion composition scheme is proposed, and several suboptimal approaches to reducing the computational complexity are discussed. In addition, frame-skipping transcoding of MPEG-4 compressed bitstreams in the DCT domain is addressed as well.

Keywords: MPEG-4, (joint) transcoding, drift error correction, requantization, frame skipping, motion estimation, motion vector refinement.

Dedicated to my wife and son…

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude and appreciation to my supervisor Dr. William E. Lynch for proposing the research topic, providing a continuous support and directing this work. I would also like to thank Mr. Cheng-Yu Pai with whom I have interacted during my studies at Concordia University.

I also would like to thank my parents-in-law and my wife. Their sacrifice and support throughout all these years made the impossible possible. I am grateful to you.

Last but not least, thanks to all my friends at Concordia University for their constant moral support and for sharing with me a new area of research.

TABLE OF CONTENTS

List of Tables

List of Figures

List of Abbreviations and Symbols

B-VOP	Bidirectionally predicted Video Object Plane
CBP	Coded Block Pattern
CBR	Constant Bit Rate
DCT	Discrete Cosine Transform
DCTTTM	DCT Coefficient Translation and Truncation Transformation Matrix
DPCM	Difference Pulse Code Modulation
GOP	Group Of Pictures
IDCT	Inverse Discrete Cosine Transform
ISO	International Standard Organization
ITU	International Telecommunication Union
I-VOP	Intra Video Object Plane
MAD	Mean Absolute Difference
MB	Macroblock
MPEG	Moving Picture Experts Group
MoMuSys	Mobile Multimedia Systems
MSE	Mean Squared Error
P-VOP	Predicted Video Object Plane
QP	Quantization Parameter
PSNR	Peak Signal to Noise Ratio

TM5	Test Model document, version 5
TWDEC	Transcoding With Drift Error Correction
TWODEC	Transcoding WithOut Drift Error Correction
UPM	Universidad Politécnica de Madrid, in Spain
VLC	Variable Length Code
VLD	Variable Length Decoding
VM	Verification Model
VO	Video Object
VOP	Video Object Plane

1. Introduction

Analogue video signals are normally generated at the output of an analog camera by scanning a two-dimensional moving scene and converting it into a one-dimensional electric signal. A moving scene is a collection of individual pictures or images. Since the end user of video will be human visual system, entertainment video has been widely used by the TV/film industries to enrich our daily life. Moreover, the most important and memorable events can be recorded and stored in media storages for replay later providing a more immediate memory experience than other media provider. Thus video is an important media to study. With breakthroughs in the price and performance of digital hardware and digital computers, several digital image compression techniques have been developed to reduce the storage space and transmission rate requirements of video, so that more widespread imaging applications have become applicable. However, the transmission of stored compressed video over channels with different capacities may require a reduction in bit rate if the transmission media has a lower capacity than the capacity required by the bitstream. Transcoding provides a feasible and reliable solution to this problem. Furthermore, in the case of multiple programs sharing the same transmission channel, in order to obtain balanced performance among different sequences, joint transcoding is an efficient approach to allocating different bitrates for different sequences according to their different properties. Transcoding can be implemented either in the pixel domain or in the DCT domain. In the following section, the importance of digital compression, transcoding and joint transcoding will be briefly discussed.

1.1 Compression Is Important to Image Processing

A camera generates a component video YC1C2 signal, and from the YC1C2 signal, three primary color signals Red, Green and Blue (RGB) can be obtained. These signals may be further processed for transmission and storage. Since these three-color signals are correlated, common color standards like PAL, NTSC and SECAM [1][2] that take advantage of this correlation are often used. Each standard generates different color spaces, for example, the color space in PAL is expressed by YUV, where Y denotes the luminance, and U and V represent the two-chrominance components. In order to easily process the video signals by using computers or other digital processors, the analogue video should be digitized. The process of digitizing analogue video signals consists of three basic operations: filtering, sampling and quantization. Usually, sampled discrete signals are quantized to 8-bit resolution, which is suitable for broadcast applications.

With the advent of video conferencing, the need for digital compression is much more important because simple sampling and binary coding of the camera voltage variations will generate millions of bits per second. Hence, without compression, the video conferencing service would be very expensive. For other applications such as entertainment TV programming and storage of digital video on small CD sized disks, compression is absolutely necessary. In addition, low bit rate applications, for examples, wireless cellular video telephony, video retrieval on the Internet or World Wide Web, also require compression. Compression makes use of redundancy or superfluous information within most sensory signals. For instance, a television camera that captures 30 frames per second from a stationary scene generates very similar frames. Compression tries to remove the

redundancy such that a single frame can be represented by a smaller amount of data. Table 1.1 shows comparisons between uncompressed and compressed video bit rates[2].

Table 1.1. Uncompressed and compressed (MPEG-2 algorithm) bit rates for film, NTSC, PAL, HTDV, and videophone

	Video Resolution (pels×lines×frames/s)	Uncompressed Bit rate (RGB)	Compressed Bit rate
Film (USA and Japan)	480×480×24Hz	133Mbits/s	3 to 6 Mbits/s
NTSC video	480×480×29.97Hz	168Mbits/s	4 to 8 Mbits/s
PAL video	576×576×25Hz	199Mbits/s	4 to 9 Mbits/s
HDTV video	1920×1080×30Hz	1493Mbits/s	18 to 30 Mbits/s
HDTV video	1280×720×60Hz	1327Mbits/s	18 to 30 Mbits/s
ISDN videophone (CIF)	352×288×29.97Hz	73Mbits/s	64 to 1920 kbits/s
PSTN videophone (QCIF)	176×144×29.97Hz	18Mbits/s	10 to 30 kbits/s

We can see from Table 1.1 that uncompressed bit rates are very high and not economical in many applications, however, using MPEG-2 compression algorithms, the bit rates can be reduced significantly. We can also see from Table 1.1 that MPEG-2 can compress the raw HDTV video from 1493Mbits/s into 18 to 30Mbits/s, and compress NTSC or PAL video from more than 133 Mbits/s into 3 to 6 Mbits/s with a comparable video quality[2].

1.2 The Importance of Transcoding and Joint Transcoding

Nowadays, networked multimedia services, such as teleconferencing, video on demand, and distance learning are emerging. In these applications, it is often necessary to adjust the bit rate of pre-encoded video bitstreams so as to meet the requirement of limited bandwidth of various transmission channels. Transcoding is a method of doing this. Generally

speaking, transcoding can be defined as the conversion of one coded signal to another one. For example, it could convert one MPEG-2 coded video to an MPEG-4 coded video. This thesis focuses on converting a previously compressed bitstream to a lower bitrate bitstream without changing coding standards. There are two common architectures used for such transcoding: one is transcoding without drift error correction; the other is transcoding with drift error correction. Transcoding without drift error correction is the simplest one where the incoming bitrate is downscaled by requantizing DCT coefficients at larger quantization step sizes. In the architecture of transcoding without drift error correction, because the transcoding is implemented in partial decoding and reencoding, as a result, it is a simple and fast transcoder. However, it can generate "drift error" which leads to degradation of reconstructed video, and probably results in unacceptable video quality, especially in higher transcoding ratio applications. Drift error can be defined as the loss of high frequency data that creates a mismatch between the actual reference frame used for prediction in the encoder and the degraded reference frame used for prediction in the transcoder and decoder[3]. It is quantization error that is propagated from the previous pictures by transcoding and is regarded as a propagated vision of requantization error to the following pictures[4].

Transcoding with drift error correction is a method to compensate for drift errors by a feedback loop; therefore, it can greatly improve the performance of transcoded video. As a result, in higher transcoding ratio applications, transcoding with drift error correction is widely adopted[5][6][7].

Some telecommunication applications such as satellite broadcasting and cable TV programming require that multiple program videos should be transmitted simultaneously

over a single communication channel. These programs share the channel capacity, and the aggregate bit rate of the programs has to be equal or less than the channel rate. This can be achieved by controlling either each individual program bit rate (independent coding) or the aggregate bit rate (joint coding). Joint transcoding reallocates the bitrate among multiple programs so that the aggregate bit rate limit is respected and that there is some balance among programs of other video qualities[8].

Transcoding can be implemented either in the pixel domain or in the DCT domain. Both requantizing DCT coefficients and skipping some frames can achieve bitrate reduction. Unlike a non-frame skipping transcoder, a frame-skipping transcoder cannot simply reuse the incoming motion vectors because the previous reference frames may be skipped.

This thesis focuses on different transcoding techniques of MPEG-4 compressed video, including (joint) transcoding in the pixel domain, transcoding in the DCT domain, and frame-skipping transcoding in the DCT domain.

1.3 Figures of Merit

Transcoding from a high bitrate bitstream to a low bitrate bitstream introduces some distortions. In order to measure these distortions, two metrics are presented: one is subjective, and the other is objective.

Subjective assessment is to show the degraded pictures to a group of subjects, and their views on the perceived quality of distortions are sought. This measurement is reliable; however, it is time-consuming and expensive. In this thesis, this measurement is not considered.

The objective measurement is to employ mathematical models simulating the human observers and usually is much faster and cheaper than subjective assessments. The simplest objective measurement is called the *peak-to-peak signal-to-noise ratio* ***(PSNR)*** and defined as:

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{N}\sum_i\sum_j\left(Y_{ref}(i,j)-Y_{prc}(i,j)\right)^2} \qquad (1.1)$$

Where $Y_{ref}(i,j)$ and $Y_{prc}(i,j)$ are the pixel values of the reference and reconstructed images, respectively. N is the total number of pixels in the image. Generally, the larger the PSNR is, the better the reconstructed image is. However, there may exist some exceptions, that is, a smaller PSNR video may have a better subjective good quality image than one with a larger PSNR.

The transcoded bit rate ratio of input and output bitstream is also a measurement of the quality of a transcoder. Broadly speaking, higher transcoded bit rate ratio results in lower quality of transcoded video. But for some slow motion sequences, higher transcoded bit rate ratio could introduce almost the same quality as that of bitstreams that are generated in lower transcoded bit rate ratio.

Besides PSNR and bitrate ratio, transcoding also results in computational complexity. Therefore, the computational complexity is also used to measure the quality of transcoders. Of course, high PSNR, high bit rate ratio, with low computational complexity is preferred. However, sometimes one could tradeoff between PSNR and complexity. An example of this is the architecture of transcoder with drift error correction (more complex, better PSNR)

versus the architecture of transcoder without drift error correction (less complex, generally worse PSNR). Other tradeoffs of complexity versus PSNR are discussed in Chapter 4.

1.4 Outline of the Thesis

This thesis is organized as follows:

Chapter 2 briefly reviews the MPEG-4 video standard and verification model, including the structures of the MPEG-4 encoder and decoder; rate control models used for generating the MPEG-4 bitstreams; and motion estimation algorithms. It also reviews the general structure of transcoding and different transcoding techniques such as (joint) transcoding in the pixel domain, transcoding in the DCT domain, and frame-skipping transcoding.

Chapter 3 describes the transcoding techniques that will be used in this thesis, including the structures of transcoding of MPEG-4 compressed video with or without drift error correction in the pixel domain, the requantization selection and the rate control algorithms. Joint transcoding of multiple MPEG-4 compressed videos will also be introduced there, and three methods (TM5-like, RMSE-based, and Approximated Distortion-based) are used to achieve the joint transcoding. The corresponding simulation results of (joint) transcoding are also given in this chapter.

Chapter 4 discusses transcoding of MPEG-4 compressed video in the DCT domain. A new DCT Coefficient Translation and Truncation Transformation Matrix (DCTTTM) based motion compensation scheme has been proposed, and three suboptimal methods used to reduce the computational complexity of transcoding with drift error correction are discussed. In this chapter, frame-skipping transcoding of MPEG-4 compressed video in the

DCT domain is also introduced. In this thesis, a new suboptimal Motion Vector (MV) Refinement algorithm used for frame-skipping transcoders in the DCT domain has been presented.

Chapter 5 concludes the thesis by summarizing the proposed schemes and discussing their performance. Future work is also included in this chapter.

2. An Overview of MPEG-4 Standard and Transcoding Techniques

As discussed in Chapter 1, compression algorithms use the redundancy within each frame or between adjacent frames. In general, there are two types of redundancies, one is statistical redundancy, and the other is perceptual redundancy. Statistical redundancy is based on the fact that samples can be similar to each other so that one sample can be predicted from another sample. It can be removed either with or without destroying any information from the original uncompressed data. Perceptual redundancy utilizes knowledge of the Human Visual System (HVS) so that what is lost in compression is not perceived.

Over years, several video Coders/Decoders (Codecs) have been invented. Examples of compression standards include H.26x, MPEG-1, MPEG-2, and MPEG-4/7, which have been provided by ITU (International Telecommunication Union) and MPEG (Moving Pictures Experts Group). The success of H.261 was a milestone for low bit rate coding of video at reasonable quality. MPEG-1 was capable of accomplishing more active moving video with high quality at bit rates in range of 1.2 to 1.5Mbits/s. MPEG-2 is now used to code the video at bit rate of 4-9Mbits/s, and has been widely in such applications as digital terrestrial broadcasting, digital satellite TV, digital cable TV, Digital Versatile Disc (DVD), and High Definition Television (HDTV). MPEG-4 is aimed at very low bit rate such as 64kbits/s or less, used over Public Switched Telephone Networks (PSTN), videophones, and mobile applications. It should be noted that MPEG-4 codes images as objects, which is

different from the legacy Codecs MPEG-1/2. MPEG-4 is the compression standard that will be used in simulations in this thesis and toward which my algorithms are aimed. In this Chapter, Section 2.1 introduces the MPEG-4 video standard and verification model (**VM**), Section 2.2 derives the simplified structure of transcoder with drift error correction, and Section 2.3 reviews transcoding and joint transcoding techniques.

2.1. The MPEG-4 Video Standard and Verification Model

With the rapid convergence of the telecommunications, computer and TV/film industries, the MPEG group officially initiated a new MPEG-4 standardization phase in 1994, allowing for interactivity, high compression and/or universal accessibility and portability of audio and video content. Bit rates targeted for the video standard are between 5-64 kb/s for mobile applications and up to 2 Mb/s for TV/film applications. In the January 1996 MPEG Video Group meeting in Munich, Germany, the first version of the official MPEG-4 Video Verification Model (VM) was defined[10][11]. The MPEG-4 Video Verification Model describes a fully defined core video coding algorithm platform (encoder, decoder as well as bitstream syntax and semantics) for the development of the standard. One of the important MPEG-4 features, which is different from the MPEG-1 and MPEG-2, is that MPEG-4 allows one or more arbitrary shaped video objects to be encoded and decoded independently since an input video sequence is segmented into a number of arbitrarily shaped image regions called video object planes (VOP's)[11]. Successive VOP's (which may belong to the same physical object in a scene) are referred to as video objects (VO's). Formally VO's are a sequence of VOP's of arbitrary shape and position. The shape, motion,

and texture information of the VOP's belonging to the same VO are encoded and transmitted into a separate video object layer (VOL). Figure 2.1 shows the hierarchy.

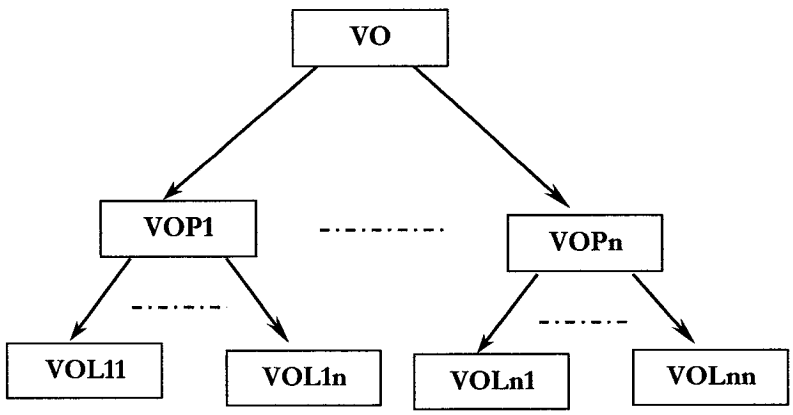


Figure 2.1 The hierarchical structure of object concepts.

In addition, relevant information needed to identify each of the VOL's and how the various VOL's are composed at the receiver in order to reconstruct the entire sequence is included in the bitstream. Figure 2.2 shows the coding of image sequences using MPEG-4's VOP's concept[11]. At the encoder, multiple VOP's can be encoded separately, and generated several bitstreams related to different VOP's. At the receiver, individual bitstreams are decoded separately and recomposed into an entire sequence.

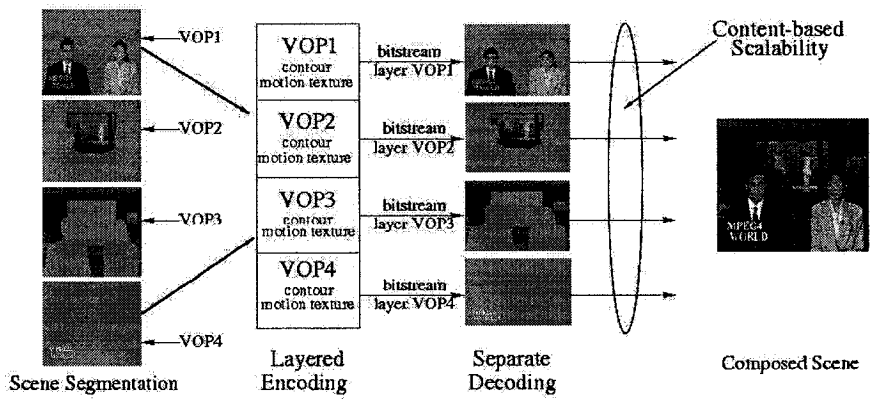


Figure 2.2 The Coding of image sequences using MPEG-4 VOP's concept[11]

As indicated in Figure 2.2, the information related to the shape, motion, and texture for each VO is coded into a separate VO Layer in order to support separate decoding of VOs. The MPEG-4 Video VM uses an identical algorithm to code the shape, motion, and texture information in each of the layers. The shape information is, however, not transmitted if the input image sequence to be coded contains only standard images of rectangular size. In this case, the MPEG-4 Video coding algorithm has the same structure as the MPEG-1/2 or H.261 coding algorithms. The MPEG-4 VM compression algorithm employed for coding each VOP image sequence (rectangular or not) is based on the successful block-based hybrid DPCM/transform coding technique that has already been employed in the MPEG-1/2 coding standards[12]. Figure 2.3 (a) shows an example of a VOP of rectangular shape[11]. The MPEG-4 coding algorithm encodes the first VOP in Intraframe VOP coding mode (I-VOP), each subsequent frames is coded using Interframe VOP prediction (P-VOP) -only the data from the nearest previously coded VOP frame is

used for prediction. In addition, the coding of bidirectionally predicted VOPs)(B-VOPs) is also supported.

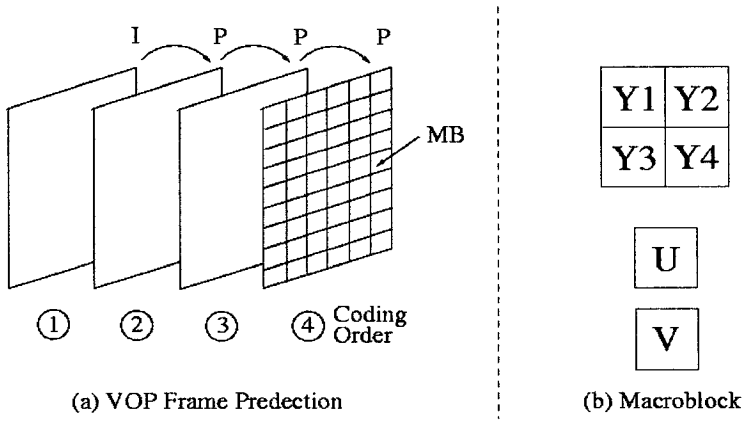


Figure 2.3 (a) Illustration of an I-VOP and P-VOP in a video sequence (b) Each Macroblock (MB) consists of four luminance blocks and two chrominance blocks, each with 8×8 pixels[11]

In general, the input image to be coded in each VOP layer is of arbitrary shape, and the shape and location of the image vary over the time with respect to a reference window. For coding shape, motion and texture information in arbitrarily shaped VOPs, the MPEG-4 Video VM introduces the concept of a "VOP image window" together with a "shape-adaptive" macroblock grid. All VOL layers to be coded for a given input video sequence are defined with reference to the reference window of constant size. Figure 2.4 shows an example of a VOP image window within a reference window and an example of a macroblock grid for a particular VOP image[11]. The shape information of a VOP is coded prior to coding motion vectors based on the VOP image window macroblock grid, and is available to both encoder and decoder. In subsequent processing steps, only the motion and texture information for the macrblocks belonging to the VOP image are coded.

In the following subsection, the structures of MPEG-4 video encoder and decoder are simply introduced.

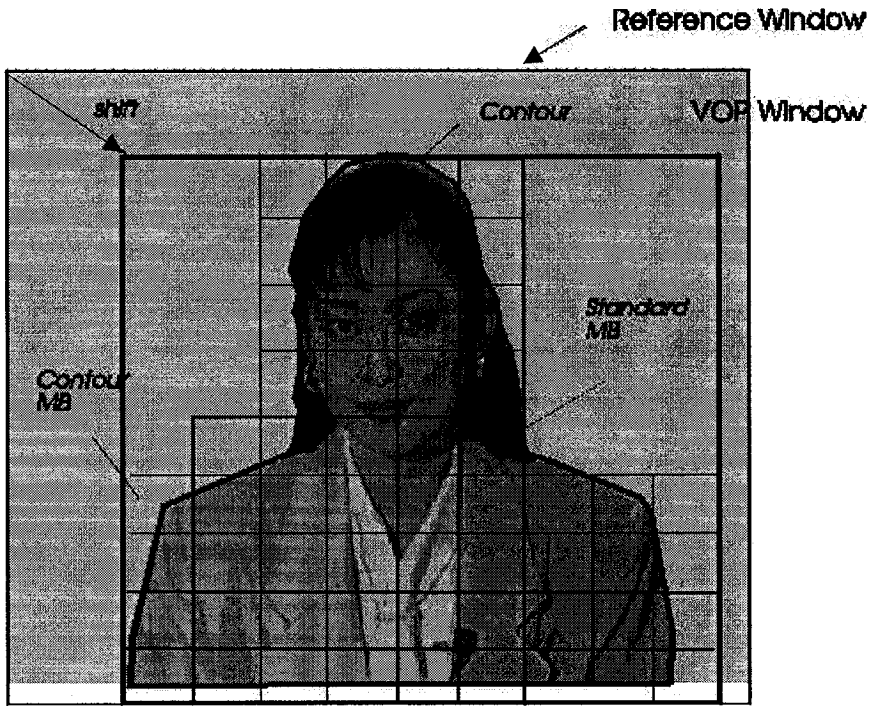


Figure 2.4 Example of an MPEG-4 VM macroblock grid for the AKIYO foreground VOP image[11]

2.1.1. MPEG-4 Video Encoder and Decoder

Figure 2.5 and Figure 2.6 show a general overview of the MPEG-4 encoder and decoder structures for each of the VOPs[13]. As we can see, the encoder is mainly composed of two parts: the *shape encoder* and the traditional *motion and texture encoder* applied to the same VOP. The decoder also consists of two parts: *the shape decoder and the traditional motion and texture decoder.*

In the following subsection, shape coding, texture coding, motion estimation, inverse quantization methods, and rate control models will be briefly described. In this thesis,

texture coding, motion estimation, inverse quantization methods, and ***rate control models*** are important parts for implementing transcoders of MPEG-4 compressed video with drift error correction. However, since only rectangular shape of objects is used in our experiments, the shape coding would not be considered in this thesis.

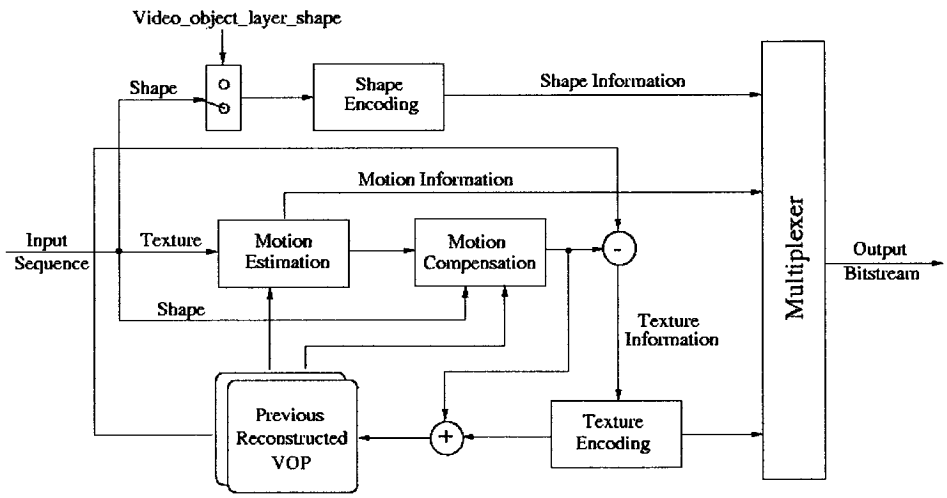


Figure 2.5 MPEG-4 VOP Encoder Structure[13]

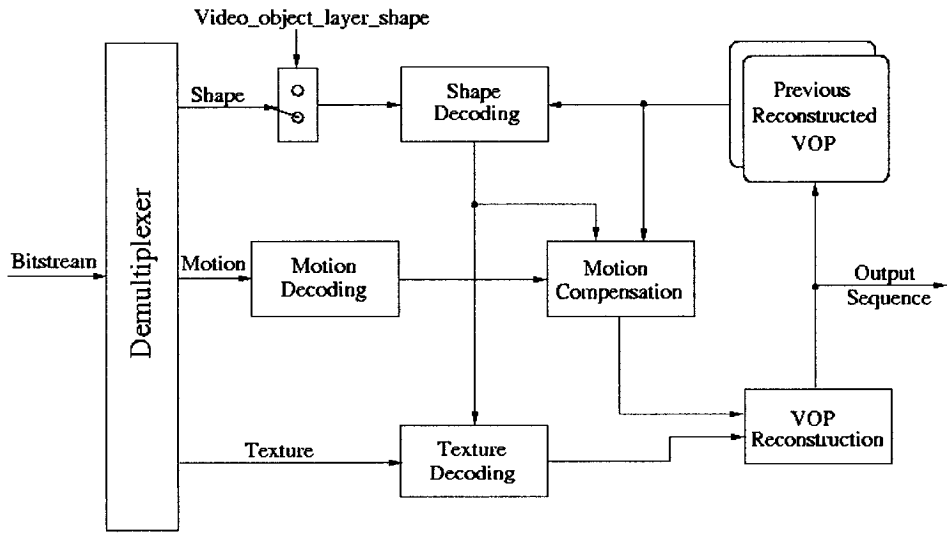


Figure 2.6 MPEG-4 VOP Decoder Structure[13]

2.1.2. Shape Coding

The shape information is referred to as an "alpha plane" in the context of the MPEG-4 VM. There are two types of shape: the *binary and grey scale shapes* or normally called *binary and grey scale alpha planes*. Binary alpha planes are encoded with one of the binary shape coding methods, while the grey scale alpha planes are encoded by motion compensated DCT similar to texture coding. A binary alpha plane can be encoded in the Intra mode for I-VOPs and the Inter mode for P-VOPs and B-VOPs. There are many coding methods used for a binary alpha plane such as *Chain coding of the object contour, Quad-tree coding, Modified Modified Reed (MMR) and Content-based Arithmetic Encoding (CAE)* [13]. The techniques to be adopted for the standard provide lossless coding of alpha-planes as well as the lossy coding of shapes and transparency information. Furthermore, it also supports both intra and inter shape coding which employs the motion compensated shape prediction - to allow both efficient random access operations as well as an efficient compression of shape and transparency information for diverse applications[13][15].

2.1.3. Texture Coding

The intra VOPs as well as the residual errors after motion-compensated prediction are coded using a DCT on 8×8 blocks similar to the MPEG-2 and H.263 standards, but there exist some differences. Because an arbitrarily shaped VOP is allowed to be coded in the MPEG-4 Verification Model, the macroblocks that completely reside inside the VOP shape are coded as normal. For the boundary macroblocks, image padding techniques[14] are used to fill the macroblock content outside of a VOP prior to applying the DCT in intra or inter VOPs. For example, if it is of Intra type, it could be padded with horizontal and vertical repetition. For inter macroblocks, the macroblock could be repeatedly padded, but

for the region outside the VOP within the block, it could be padded with zeros. The adaptive VOP window macroblock grid specified in Figure 2.4 is also employed for this purpose, but particular adaptation is required for the 8×8 blocks straddling the VOP borders. For transparent blocks (all zero), they are skipped, i.e. not coded. Scanning of the DCT coefficients followed by quantization and run-length coding of the coefficients is performed using same techniques and VLC tables defined with the MPEG-1/2 and H.263 standards[14].

2.1.4. Motion Estimation and Compensation

Besides macroblock-based motion estimation and compensation, the MPEG-4 VM employs block-based motion estimation and compensation techniques to efficiently explore temporal redundancies of the video content in the separate VOP layers. In general, the techniques used are somewhat similar to those used in MPEG-1/2 or H.261/3. The MPEG-4 VM also supports the coding of both forward-predicted VOP (P-VOP) and bidirectionally predicted VOP (B-VOP). Motion vectors are predictively coded using standard MPEG-1/2 or H.263 VLC code tables. It should be noticed that the MPEG-4 VM still supports the coding of standard MPEG I-frames, P-frames and B-frames for the special case of image input sequences (VOP's) of rectangular shape.

2.1.5. Inverse quantization methods

In a transcoder, the incoming quantized DCT coefficients should be first dequantized, and then requantized by the second quantizer with larger step size to achieve much lower bit rate. Therefore, the inverse quantization is necessary. For texture object decoding, the MPEG-4 standard allows two types of inverse quantization: an MPEG-1/2 like method, or

an H.263 like method. For the first, two matrices are used: one for intra blocks, and the other for non-intra blocks. The default matrix for intra blocks is[2]:

$$\begin{bmatrix} 8 & 17 & 18 & 19 & 21 & 23 & 25 & 27 \\ 17 & 18 & 19 & 21 & 23 & 25 & 27 & 28 \\ 20 & 21 & 22 & 23 & 24 & 26 & 28 & 30 \\ 21 & 22 & 23 & 24 & 26 & 28 & 30 & 32 \\ 22 & 23 & 24 & 26 & 28 & 30 & 32 & 35 \\ 23 & 24 & 26 & 28 & 30 & 32 & 35 & 38 \\ 25 & 26 & 28 & 30 & 32 & 35 & 38 & 41 \\ 27 & 28 & 30 & 32 & 35 & 38 & 41 & 45 \end{bmatrix} \quad (2.1)$$

The default matrix for non-intra blocks is:

$$\begin{bmatrix} 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \\ 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \\ 18 & 19 & 20 & 21 & 22 & 23 & 24 & 25 \\ 19 & 20 & 21 & 22 & 23 & 24 & 26 & 27 \\ 20 & 21 & 22 & 23 & 25 & 26 & 27 & 28 \\ 21 & 22 & 23 & 24 & 26 & 27 & 28 & 30 \\ 22 & 23 & 24 & 26 & 27 & 28 & 30 & 31 \\ 23 & 24 & 25 & 27 & 28 & 30 & 31 & 33 \end{bmatrix} \quad (2.2)$$

Suppose that $QF[i][j](0 \le i, j \le 7)$ is quantized DCT coefficient, and $F'[i][j](0 \le i, j \le 7)$ is the reconstructed DCT coefficients. The inverse quantization is essentially a multiplication by the quantizer step size. The quantizer step size is modified by two mechanisms; a weighting matrix (Equ 2.1 or Equ.2.2) is used to modify the step size within a block and a scale factor is used in order that the step size can be modified at the cost of only a few bits (as compared to encoding an entire new weighting matrix). However, the DC coefficients of intra-coded blocks shall be inversely quantized in a

different manner to all other coefficients. In intra blocks, DC coefficient $F'[0][0]$ shall be obtained by multiplying $QF[0][0]$ by a constant multiplier. The reconstructed DC values are computed as follows:

$$F'[0][0] = dc\_scaler \times QF[0][0] \tag{2.3}$$

The *dc_scaler* could be computed from Figure 2.7(a) according to the quantization parameters QP.

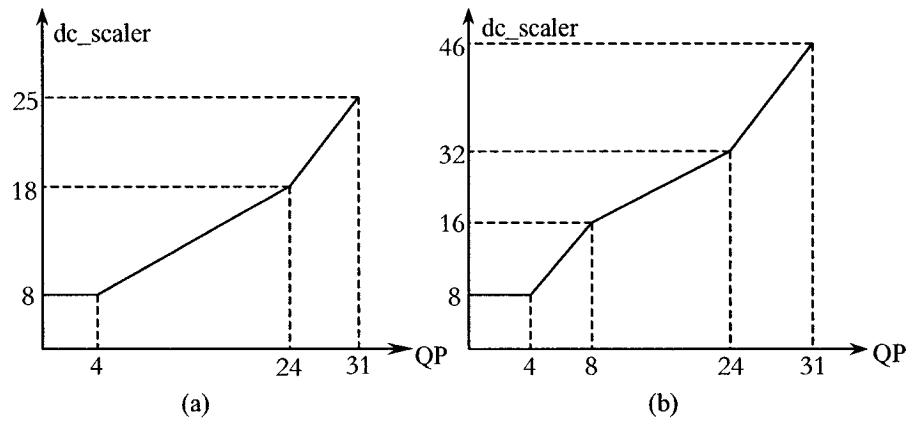


Figure 2.7 (a) piecewisely linear mapping from *QP* to *dc_scaler* for MPEG-1/2 quantization type; (b) piecewisely linear mapping from *QP* to *dc_scaler* for H.263 quantization type

All other coefficients other than the DC coefficient of an intra block shall be inversely quantized as specified in the MPEG-1/2 standards[20]. Two weighting matrices in equation (2.1) and (2.2) are used. One shall be used for intra macroblocks and the other for non-intra macroblocks.

Let the weighting matrices be denoted by $qmat[l][i][j]$ where l takes the value 0 or 1 indicating which of the matrices is being use. For instances, $qmat[0][i][j]$ is for

intra macroblocks, and $qmat[1][i][j]$ is for non-intra macroblocks. Then the reconstructed DCT coefficient can be obtained by:

$$F'[i][j] = \begin{cases} 0 & if\ QF[i][j] = 0 \\ ((2 \times QF[i][j] + k) \times qmat[l][i][j] \times qscaler)/16 & if\ QF[i][j] \neq 0 \end{cases} \quad (2.4)$$

The value of *qscaler* is determined by VOP quantization parameters (QP) for luminance and chrominance. As for k, it is defined by:

$$k = \begin{cases} 0 & \text{if it is Intra block} \\ Sign(QF[i][j]) & \text{if it is non Intra block} \end{cases} \quad (2.5)$$

The second inverse quantization method is used for all the coefficients other than the DC coefficient of an intra block. In the second inverse quantization method, the DC coefficient of an intra block is quantized using the same method as in the first inverse quantization method but *dc_scaler* is calculated from Figure 2.2(b) according to the quantization parameters QP. Unlike the first method, there are no weighting matrices required, and the parameter *qscaler* may take integer values from 1 to $2^{quant\_precision} - 1$. The quantization step size is equal to twice the *qscaler*. The reconstructed DCT coefficient of both Intra AC quantized DCT coefficient and all non-Intra quantized DCT coefficients $F'[i][j]$ can expressed as:

$$F'[i][j] = \begin{cases} 0 & if\ QF[i][j] = 0 \\ Sign(QF[i][j]) \times (2 \times |QF[i][j]| + 1) \times qscaler & if\ QF[i][j] \neq 0,\ qscaler is\ odd \\ Sign(QF[i][j]) \times (2 \times |QF[i][j]| + 1) \times qscaler - 1 & if\ QF[i][j] \neq 0,\ qscaler is\ even \end{cases} \quad (2.6)$$

2.1.6. Rate Control Models

In[32], there are two-rate control algorithms employed in MoMuSys Codec, i.e. Q2 and UPM rate control models. Q2 rate control only independently allocates bitrate for every VOL, while UPM is global rate control. UPM rate control model consists of four other different algorithms. Among four UPM rate control algorithms, UPM1 is used in this thesis, while other three algorithms will be briefly introduced in Chapter 3. UPM1 allocates bits counts among the VOLs by minimizing the sum of weighted distortions-per-pixel of the VOLs. Here, Q2 and UPM1 rate control mechanisms are discussed in the following sections.

1) Q2 Rate Control Model

Q2 rate control model is based on the quadratic rate distortion model introduced in [33] to estimate the quantization parameter used in MPEG-4 encoder. The Q2 Rate Control Model can be expressed as follows:

$$R = S\left(\frac{X_1}{Q} + \frac{X_2}{Q^2}\right) \tag{2.7}$$

Where, R is the encoding bit count for texture object, S represents the encoding complexity which is the mean absolute difference (***MAD***), the quantization parameter is denoted as Q, and X_1 and X_2 denote the first and second modeling parameters. Given the target bit rate R, by solving the equation (2.7), we can get quantization parameter Q used for the next VOP:

$$Q=\begin{cases}\dfrac{SX_1}{R} & \text{if } X_2=0 \text{ or } (SX_1)^2+4RSX_2<0 \\ \dfrac{2X_2S}{\left(\sqrt{(SX_1)^2+4RSX_2}-SX_1\right)} & \text{otherwise}\end{cases} \tag{2.8}$$

Where, X_1 and X_1 can be derived from linear regression method by using the past n frames[33]:

$$\begin{cases} X_2 = \dfrac{n\sum_{i=1}^{n}\dfrac{R_i}{S_i}-\left(\sum_{i=1}^{n}Q_i^{-1}\right)\left(\sum_{i=1}^{n}\dfrac{Q_iR_i}{S_i}\right)}{\left[n\sum_{i=1}^{n}Q_i^{-2}-\left(\sum_{i=1}^{n}Q_i^{-1}\right)^2\right]} \\ X_1 = \dfrac{\sum_{i=1}^{n}\dfrac{Q_iR_i}{S_i}-X_2Q_i^{-1}}{n} \end{cases} \tag{2.9}$$

Where

n: the number frames observed in the past;

S_i: the mean absolute difference in the past;

Q_i, R_i: the actual encoding average quantization parameter and bit count in the past;

2) UPM1 Rate Control Model

The UPM1 rate control model is similar to the Q2 rate control model, but tries to minimize the sum of weighted distortions-per-pixel of the VOLs. In order to calculate the quantization parameter for the next VOP, the rate count and distortion should be

considered together. The Rate control model can be written as:

$$\begin{cases} R = Pixels \times S \times \left(\dfrac{X_1}{Q} + \dfrac{X_2}{Q^2} + X_3 \right) \\ D = Pixels \times \left(Y_1 \times Q + Y_2 \times Q^2 + Y_3 \right) \end{cases} \tag{2.10}$$

where,

R, D are the target bit rate and distortion;

Pixels is the number of pixels per VOP;

S is the Mean Absolute Difference (***MAD***), which is the same as that in Q2 rate control model;

X_i, Y_i are the modeling parameters for rate and distortion models that can also be derived from linear regression method by using the past n frames.

By using Equation (2.10), the optimal value of Q must meet the following criteria:

$$\begin{cases} \left\{ \overrightarrow{R^*}, \overrightarrow{D^*} \right\} = \underset{R_i, D_i}{\arg\min} \sum_{i=1}^{M} \left(R_i + D_i \xi_i \right) \\ \sum_{i=1}^{M} R_i \leq R \end{cases} \tag{2.11}$$

Where ξ_i are configuration parameters defined in each VO configuration file ($*.cfg$), which are used to identify the importance of each VO. For example, if there are four VOs in a scene, say VO_0, VO_1, VO_2, and VO_3, if $\xi_0 = 0.5$, $\xi_1 = 0.8$, $\xi_2 = 1.0$ and $\xi_3 = 0.3$, that means that VO_0 is the third important, VO_1 is the second important, VO_2 is the most important and VO_3 is the least important. Because the complexity of each VO in the scene is different, the procedure for setting the ξ_i allocates more bits to more complex VOs.

2.2. Structures of Transcoder

Conceptually, a transcoder[9] consists of a cascaded decoder and an encoder. Often the bitstream is only partially decoded. For instance the motion vectors from the original encoding are reused in the transcoder, which greatly reduces computational complexity. Figure 2.8 shows the basic configuration of a system including a cascaded transcoder; Figure 2.9 shows the structure of cascaded transcoder including decoder and encoder in more details.

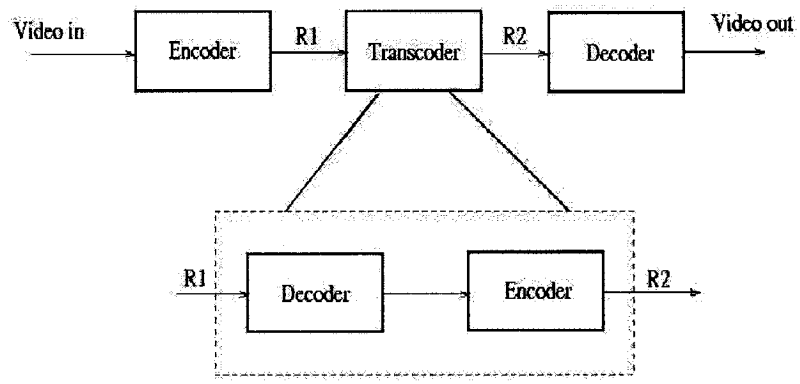


Figure 2.8 Basic Configuration of a system including a transcoder

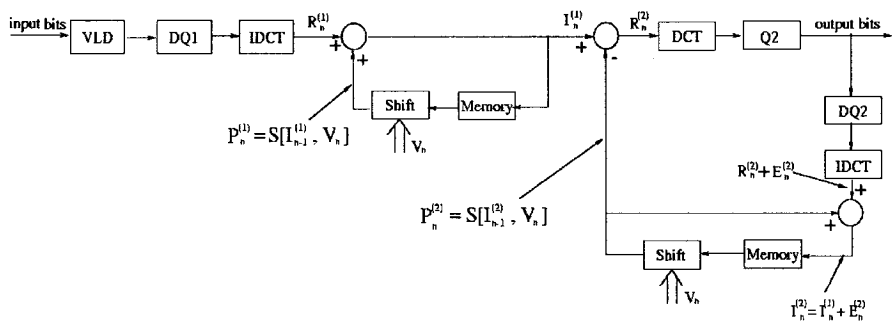


Figure 2.9 The structure of cascaded decoder and encoder as a transcoder

Transcoding by means of a cascaded decoder and encoder may lead to two types of problems: one is that of computational complexity and the other is that of performance. As we can see from Figure 2.9, the cascaded transcoder needs 3(I) DCT, 2 frame memories and 3 additions.

Let V_n represent the vector field of current picture n, and assume that the position of a pixel in a picture is denoted by x, the displacement is denoted $V_n(x)$ the motion-compensated signal is denoted $S[I_{n-1}^{(1)}, V_n]$, where $S[\ ]$ is referred to as the shift operation, $n-1$ stands for the previous frame, the superscript 1 means the decoder in the transcoder, and superscript 2 is donated as the encoder in the transcoder. For example, $I_{n-1}^{(1)}$ is the previous decoded signal from the decoder.

Therefore, the motion compensation yields the prediction $P_n^{(1)} = S[I_{n-1}^{(1)}, V_n]$ as follows:

$$P_n^{(1)}(x) = S[I_{n-1}^{(1)}, V_n](x) = I_{n-1}^{(1)}[x + V_n(x)] \qquad (2.12)$$

Where $V_n(x)$ is equal displacement for each pixel within a macroblock.

Since the shift operator is linear, in Figure 2.9, the prediction for the second encoder is given by the following equation:

$$P_n^{(2)} = S[I_{n-1}^{(1)} + E_{n-1}^{(2)}, V_n] = S[I_{n-1}^{(1)}, V_n] + S[E_{n-1}^{(2)}, V_n] \qquad (2.13)$$

Where $E_n^{(2)}$ is the requantization error caused by the second quantizer Q2. Thus, the residual $R_n^{(2)}$ can be written as:

$$R_n^{(2)} = I_n^{(1)} - S[I_{n-1}^{(1)} + E_{n-1}^{(2)}, V_n]$$
$$= I_n^{(1)} - S[I_{n-1}^{(1)}, V_n] - S[E_{n-1}^{(2)}, V_n] \quad (2.14)$$

And the decoded picture $I_n^{(1)}$ can be written as:

$$I_n^{(1)} = R_n^{(1)} + S[I_{n-1}^{(1)}, V_n] \quad (2.15)$$

Substituting Equ.(2.15) into Equ.(2.4), we get:

$$R_n^{(2)} = R_n^{(1)} - S[E_{n-1}^{(2)}, V_n] \quad (2.16)$$

Then, according to Equ (2.16), the residual $R_n^{(2)}$, which is used to produce the output bitstream, can be obtained directly by subtracting the motion-compensated quantization error $E_{n-1}^{(2)}$. Here, $S[E_{n-1}^{(2)}, V_n]$ is so called drift error in the transcoder. From Equ (2.16), only $E_{n-1}^{(2)}$ need be stored and the store used for $I_n^{(1)}$ can be dropped. Figure 2.10 shows a more simplified structure of the transcoder by moving the DCTs and IDCTs around in the block diagram[9], where 1 IDCT, 1 Memory and 1 Add are saved as compared to the typical cascaded transcoder shown in Figure 2.9. In this thesis, transcoding of MPEG-4 compressed video will adopt this simplified structure to implement the transcoder with drift error correction.

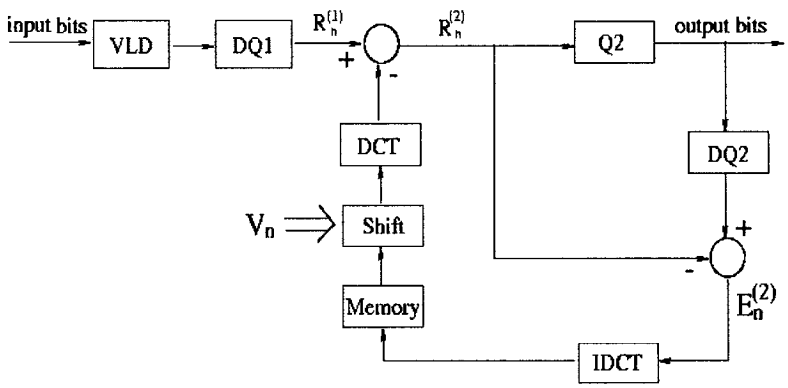


Figure 2.10 Transcoding system with a simplified structure

2.3. Review of Transcoding and Joint Transcoding Techniques

If the transmission media has a lower capacity than the capacity required by an already compressed bitstream, then bit rate may need to be further reduced. Up until now, there have been many different approaches to solving this problem such as motion-vectors reduction, frame dropping, reducing the spatial resolution of the video, discarding high-frequency DCT coefficients, requantizing the DCT coefficients, and fully decoding/re-encoding of the video[5][6][7][16][18][19][24]-[29]. Among these approaches, full decoding/re-encoding of a compressed video involves much more computational complexity related to the MPEG encoding algorithm.

There are several mechanisms that govern how much degradation occurs in transcoding. Several that are present in multigeneration (full decoding, recoding at the same bit rate) and that are related to the space domain are outlined in [16][18][19]. These mechanisms would also be present in a full decoding/encoding transcoding scenario, but would be avoided in a transcoding scheme where only a partial decoding/encoding occurred. At low transcoding ratios (small reduction in bits), these partial decoding/encoding schemes are superior. However, for high ratio transcoding, the full decoding/encoding is likely to be better as there is more benefit from motion vector reestimation.

Generally speaking, similar to transcoding of MPEG-2 compressed video, transcoding of MPEG-4 compressed video from a high bit rate to a low bit rate can be implemented either in the pixel domain or in the DCT domain. In the pixel domain, DCT and IDCT operations can be considered in the transcoder with drift error correction, while in the DCT domain, these two operations can be avoided, but compositing one DCT block

from four adjacent DCT blocks involves more computational complexity of matrix multiplications and additions [20][22][23][24]. Transcoding from a high bitrate bitstream to a low bitrate bitstream may result in the degradation of performance. No matter in which domain the transcoder is implemented, drift error correction is an efficient method to improve the performance by compensating the requantization error, and has almost the same performance as the cascaded full decoding/re-encoding algorithm. For the rest of this thesis, we will refer to these methods as Transcoding WithOut Drift Error Correction (**TWODEC**) and Transcoding With Drift Error Correction (**TWDEC**). Note that several strategies are examined to do TWDEC.

Although in most cases, the motion vectors are reused in frame-skipping transcoding, the incoming motion vectors cannot be simply employed to reconstruct the current frame because the previous reference frames may be dropped. In this case, the motion vectors should be re-evaluated. Motion estimation can result in large amounts of computational complexity; there are some techniques[25][26][27] to solve this problem. The most popular scheme is first to composite the dominant motion vectors, and then refine the motion vectors with a small search area size. This scheme is usually called the Motion Vector (MV) refinement scheme.

In the past several years, there has been a great deal of work on transcoders[4][5][6] [7][28][29]. In some applications, several programs are transmitted over the same transmission channel. Hence, in order to make full use of transmission resources, and to obtain overall good video quality, the bit rate for each program is intelligently reallocated. This introduces another transcoding technique called joint transcoding[8]. All above-mentioned transcoders are implemented with MPEG-2 compressed video.

With regards to MPEG-4 transcoders, in [30], two approaches are discussed about object-based transcoding for adaptive video content delivery: a Dynamic Programming approach; and a Meta-Data based approach. The first approach selects the requantization parameter QP2 for each object based on the distortion increases and rate budget. The second approach is based on the availability of meta-data, where several transcoding hints could be used for different purposes. In [30], they do not consider the influence of the ratio of requantization parameters to the first encoding quantization parameters on the transcoding performance. In addition, they do not implement the transcoder with drift error correction, which can effectively improve the quality of transcoded video sequences. In this thesis, various transcoding schemes of MPEG-4 compressed bitstream in the pixel domain and in the DCT domain will be discussed. In the pixel domain, a transcoder of MPEG-4 compressed video with drift error correction and selective requantization parameter method has been presented. Also, a new approximate distortion based bit rate allocation algorithm has been proposed for joint transcoding of multiple MPEG-4 compressed bitstreams in order to take advantage of limited transmission channel. In the DCT domain, a new DCT Coefficient Translation and Truncation Transformation Matrix (**DCTTTM**) based motion composition scheme has been proposed, and several suboptimal approaches to reducing the computational complexity are analyzed. Finally, a frame-skipping transcoder in the DCT domain is discussed, and a new suboptimal motion vector refinement method is also introduced.

3. Transcoding and Joint Transcoding of MPEG-4 Compressed Video in the Pixel Domain

Because MPEG-4 consists of shape coding and texture coding, theoretically, in the transcoder of MPEG-4 compressed video, two parts should be considered: one for shape information, the other for the texture information. Since only rectangular shaped video objects are employed in our simulations, the shape information is equal to zero, and it is ignored in the process of our implementation; therefore, only the texture content of MPEG-4 compressed video is transcoded.

In this Chapter, (joint) transcoding of texture content of MPEG-4 compressed video in the pixel domain will be discussed. In section 3.1, the structure of transcoder regarding to MPEG-4 is first presented, then some issues related are discussed. The last part of this section gives our simulation results. In section 3.2, joint transcoding of MPEG-4 is introduced, a new joint bit rate allocation algorithm called approximated distortion method is proposed, and the corresponding simulation results are also given. Section 3.3 summarizes this Chapter.

3.1. Transcoding of MPEG-4 Compressed Video

With regard to transcoding of MPEG-4 compressed video, two transcoding schemes are presented and implemented: one is without drift error correction and the other is with drift error correction. Figure 3.1 and Figure 3.2 show the two object-based transcoding systems of MPEG-4 compressed video without and with drift error correction, respectively. In both structures, shape information is passed through, i.e. decoded and

re-encoded into the output bitstream. Similarly, motion vectors of the input bitstream are reused. This avoids motion estimation, the most computationally intensive operation of the MPEG-4 encoding algorithm, and significantly reduces the complexity of the system, as compared to a cascaded decoder-encoder system.

However, macroblock types may be changed after requantization. For example, a motion-compensated coded macroblock may be changed to a motion-compensated not coded macroblock if all the DCT coefficients are zero after requantization. In Figure 3.1, the transcoder without drift correction simply decodes the texture and requantizes it according to the target bit rate without considering the error introduced by requantizer Q2. The advantages of this structure are its lower complexity and smaller memory requirements. This transcoder without drift correction does not require any frame buffer. It is more suitable for low delay and low cost applications, and for video bitstreams having a short Group Of Pictures (GOP) structure, which reduce the effects of drift. However, the performance (in PSNR) of the transcoded bitstream will be worse, especially in the case of a large transcoding recompression ratio ($R1/R2$ where $R2$ is the output bitrate, $R1$ is the input bitrate). In order to overcome this problem, in Figure 3.2, the transcoder with drift correction uses a feedback loop to compensate for drift errors that result from requantizing a motion-compensated compressed video. The residual $R_n^{(2)}$ for the second quantizer can be written as:

$$R_n^{(2)} = R_n^{(1)} - S[E_{n-1}^{(2)}, V_n] \tag{3.1}$$

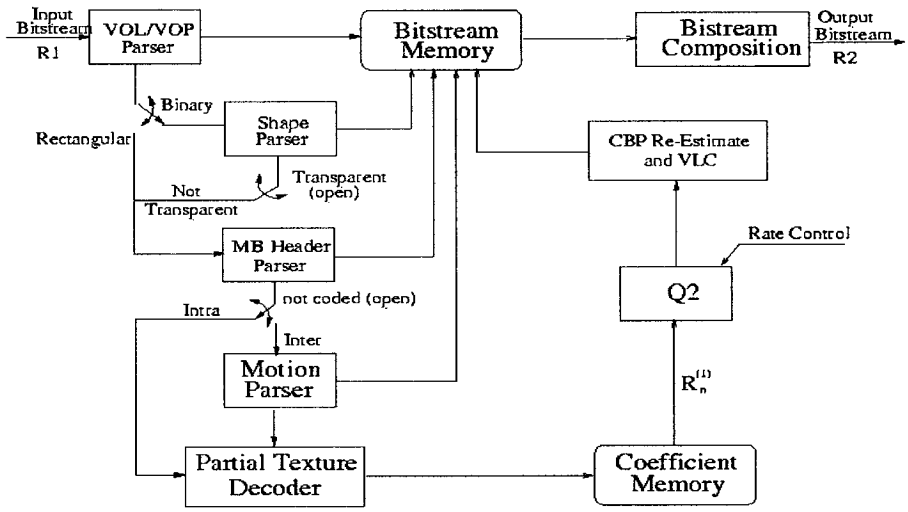


Figure 3.1 Object-based transcoding system of MPEG-4 without drift error correction[30]

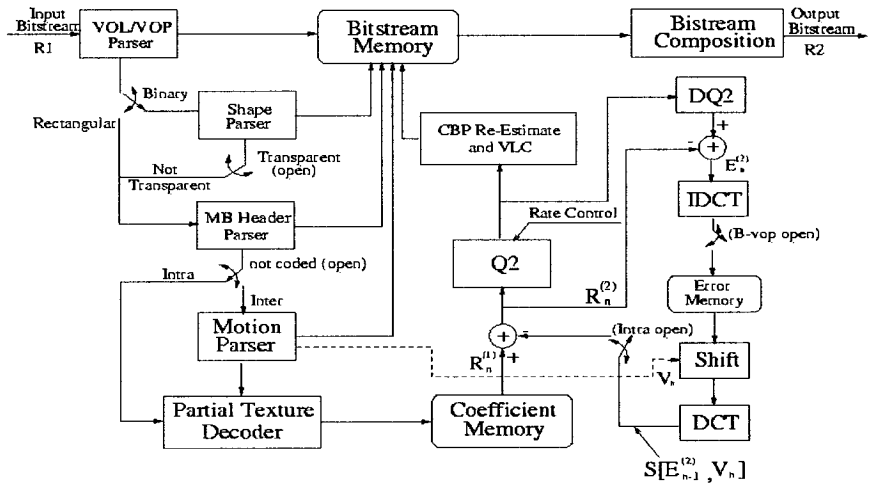


Figure 3.2 Object-based transcoding system of MPEG-4 with drift error correction

The new texture object residual signal $R_n^{(2)}$ is obtained by subtracting the motion-compensated error $S[E_{n-1}^{(2)}, V_n]$ from the decoded residual texture object residual $R_n^{(1)}$. It should be pointed out that for I-VOP, we cannot compensate the requantization errors, but those requantization errors, which generate the drift errors, could be computed and stored for compensating the future inter coded VOP; for P-VOP's, using the same motion vectors of the first encoder of current VOP and for the future VOP prediction respectively. For B-VOP, because it is not used for any further picture, there is no need to store it. Simulation results in Section 3.1.3 show the quality with drift correction is improved over that without drift error correction, as expected

As we can see from Figure 2.10 and Figure 3.2, although MPEG-4 transcoding is similar to MPEG2 transcoding, there are still some differences. Because MPEG-4 video includes not only texture information but also shape information, in practice, in order to accurately predict the requantization parameter $Q2$ for texture, we should subtract the bits used for coding shape as well as motion vectors and VOP header information from the target bitrate per frame. Furthermore, the requantized coefficients are different from those of the first encoding stage, thus at MB layer, the CBP (Coded Block Pattern) of each macroblock should be recomputed. The CBP not only affects the structure of the bitstream, but also has an impact on intra AC/DC prediction (the parameter ACpred_flag). Due to their different syntax, in MPEG-2, quantization parameter (QP) is directly included at MB layer, while in MPEG-4, the VOP header includes QP parameter, each macroblock uses the bias value $DQUANT$ $(-2 \le DQUANT \le 2)$ to slightly adjust the real QP used for each MB. Here, we only do transcoding VOP by VOP (one quantization

parameter for each VOP), not macroblock by macroblock.

3.1.1. Issues Related to Transcoding of MPEG-4 Bitstreams

Transcoding of MPEG-4 compressed video with and without drift correction involves a requantization of the originally quantized DCT coefficients. The cascaded dequantization and requantization lead to extra distortion as compared to direct quantization of the original DCT coefficient with the second (coarser) quantizer. As the step size of the finer quantizer is encoded in the bitstream and is available to the transcoder, it is possible to use this information to reduce the requantization errors and achieve a higher transcoding quality. In[5], a study of the requantization problem in transcoding of MPEG-2 Intra frames with emphasis on designing the quantizer of the transcoder is presented. The paper provides two approaches for adjusting the decision levels of the transcoder's quantizer to improve picture quality. The first uses a mean-squared error (***MSE***) cost function, and the second uses a maximum a posteriori (***MAP***) cost function. Both **MSE** and **MAP** quantizers result in a higher picture quality in transcoding than the TM5 quantizer[31]. In[7], strategies were introduced to avoid critical ratios (Q2/Q1) between the finer (encoder) and coarser (transcoder) quantizers in order to reduce the requantization errors and achieve a higher transcoding quality. In this thesis, these criteria are also applied to MPEG-4 transcoders.

The rate control algorithm is another important issue in implementing an efficient encoder or transcoder. In[24], an approach to transcoding optimization and rate control by minimizing a Lagrangian cost has been proposed. This technique allows the selection of a trade-off point between rate and distortion subject to a rate constraint $R \leq R_{\max}$.

However, this method is computationally expensive and cannot be easily implemented on a macroblock by macroblock basis. One well-known rate control algorithm is described in the Test Model document, version 5 (*TM5*)[31]. The advantage of this algorithm is that the operations can be easily implemented on a macroblock basis, and therefore is more convenient for real time and low delay transcoding. Our simulations are implemented based on MoMuSys MPEG-4 encoder and decoder software V1.0[32]. There are several rate control algorithms that have been employed such as Q2, UPM1, UPM2, UPM3 and UPM2a rate control algorithms. The rate control algorithm is determined by the selection of rate control type in the control file and the config file. The following gives out the difference of every rate control model.

1) **Q2**: Rate control specified in [32] provides only independent control for every VOL.

2) **UPM1**: Global rate control. The distribution of the global bit-rate among the VOLs attempts to minimize the sum of weighted distortions-per-pixel of the VOLs. The weights for each VO are given by the user through "*Global RC Parameter 1*" in the configuration file of each VOL.

3) **UPM2**: Global rate control. The distribution of available bits among VOLs is made according to a priority list and a target PSNR for each VOL. The index of each VOL in the list is given by the user through "*Global RC Parameter 1*" in the configuration file of the VO and the target PSNR is given in "*Global RC Parameter 2*" in the same file. This algorithm intends to obtain a desired quality at least for the most important VO. If it is not possible because of limitations in

bit-rate, it will be coded as well as possible.

4) **UPM3**: Global rate control. The distribution of the global bit-rate among the VOLs attempts, as algorithm UPM1, to minimize the sum of weighted distortions-per-pixel and additionally keep the ratio between the distortions of the different VOPs constant.

5) **UPM2a**: Global rate control. Algorithm UPM2 with stronger restrictions. This algorithm also calculates a distribution of the global target bitrate between the different VOs under the constraint to assure a minimum quality (target PSNR) for the most important VOs. The difference to algorithm UPM2 is that no bits are assigned to lower priority VOs, if for the more important one the maximum possible distortion $D_{\max}$ still has not been achieved. Furthermore, there exists the option of a further restriction, which is, that the QP of a VO cannot be lower than the one of a VO with higher priority.

In this thesis, in order to simplify the implementation of transcoders, only Q2 and UPM1 rate control algorithms are used. In the next sections, two other important parts related to transcoders, i.e, the selective requantization method[7] and rate control models, are briefly discussed.

3.1.2. The Selective Requantization

In [7], the selective requantization method is introduced. It is based on avoiding critical ratios of $Q2/Q1$ ($Q2$: transcoder quantizer parameter, $Q1$: the first step finer quantizer parameter) that either lead to larger transcoding errors or require a higher bit budget for little or no gain in output quality. Essentially ratios that lead many breakpoints of the

second quantizer (the transcoding quantizer) to land on reconstruction levels of the first quantizer (the original compression quantizer) lead to large requantization errors. Such ratios are avoided. For a uniform midstep quantizer (suitable for Intra Macroblock), an even integer ratio of $Q2/Q1$ must be avoided; while for a midriser quantizer (suitable for non-Intra Macroblocks), any odd integer ratio of $Q2/Q1$ should be avoided. Since MPEG-4 standard allows both MPEG-2 and H.263 quantizations, therefore, the selective requantization method could be used for CBR transcoding of MPEG-4 compressed video. Let *qscale_1* and *qscale_2* be the first quantization step size and requantization step size, According to [7], the conditions applied to Intra/Inter macroblock requantization can be summarized as follows:

1. Intra Macroblock
   (a) $if((qscale\_2\%(2\times qscale\_1)) == 0)$
   qscale_2=qscale_2+2;
   (b) $if(((qscale\_2+2)\%qscale\_1) == 0)$
   $if(((qscale\_2+2)/qscale\_1)\%2)!= 0)$
   qscale_2=qscale_2+2;
   (c) $if(((qscale\_2+2)\%(2\times qscale\_1)) == 0)$
   qscale_2=qscale_2+4;
2. Non-Intra Macroblock
   (a) $if(((qscale\_2+2)\%qscale\_1) == 0)$
   qscale_2=qscale_2+2;
   (b) $if(((2\times qscale\_2)\%qscale\_1) == 0)$
   $if((((2\times qscale\_2)/qscale\_1)\%2)!= 0)$
   qscale_2=qscale_2+2;

The above items are the result of the analysis for midstep (Intra) and midriser

(Non-Intra) quantizers. For more details see [7], Item 1.(a) and Item 2.(a) are the most important to avoid larger transcoding errors for Intra and non-Intra macroblock requantization.

3.1.3. Rate Control Models in the Transcoder

In this thesis, two rate control algorithms are employed: Q2 and UPM1. As discussed above, in the encoder of MPEG-4, Q2 rate control independently allocates bitrate for every VOL, while UPM1 is a global rate control. UPM1 allocates bits among the VOLs by minimizing the sum of weighted distortions-per-pixel of the VOLs. It should be noted that in the MPEG-4 transcoder, the complexity S must be measured in a different way from that used in the encoder where we can exactly know the original input pixel value of each frame in the spatial domain. Referring to [30], we define the estimated value S' as:

$$S' = \sqrt{\frac{1}{M_c} \sum_{M_c} \sum_{i=1}^{63} \gamma(i) \times |C_m(i)|^2} \qquad (3.3)$$

Where, $C_m(i)$ are the DCT AC coefficients of each MB from the first encoding stage, M_c is the number of Macroblock (MB) per frame, and $\gamma(i)$ are weighting factors used to adjust the DCT coefficients. For the MPEG-2 quantizer, $\gamma(i)$ could be replaced with its quantization matrix, while for the H.263 quantizer, they can simply be set 1.

Figure 3.3 shows the relationship between S and S'. As we can see, with the exception of a few outliers, they are approximately linearly related. As a results, S' can be used for rate control models in the transcoder as S is employed in MPEG-4 encoder.

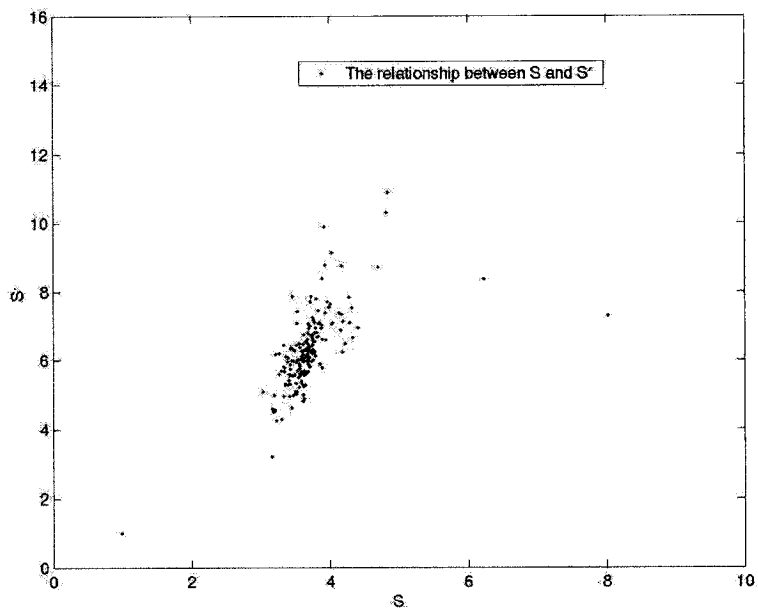


Figure 3.3. The relationship between S and S'

Based on the above conditions, the rate control model parameters can be computed and continually updated. To obtain the new quantization parameter Q', we utilize the same methodology as in the MPEFG-4 encoder, but apply the selective requantization strategy and drift error correction in order to reduce the distortion compared to the cascaded or directly encoding methods.

3.1.4. Simulation Results

The transcoding software is implemented based on MomuSys software V1.0[32]. Q2 rate control model is employed. We use *200* frames of the sequences *Foreman* (frame size: 176×144), Mobile (frame size: 352×288), *Tennis* (frame size: 352×240) and 150 frames for *Flowergarden* (frame size: 352×240). The sequences are all with *IPPP*.... GOP structure. For *Foreman*, the incoming bitrate is R1=256 kbps, and the outgoing bitrate is R2=128 kbps; for *Mobile*, R1=1024 kbps, R2=512 kbps; for *Tennis*, R1=640 kbps, R2= 480 kbps and for *Flowergarden*, R1=980 kbps, R2=480 kbps. The frame rates are all 30 frames/second. In order to compare the performance of the proposed transcoder, all sequences are tested in three ways: cascaded transcoding, transcoding without drift correction, and transcoding with drift correction. The simulation PSNR results are shown in Figure 3.4, Figure 3.5, Figure 3.6, and Figure 3.7, respectively.

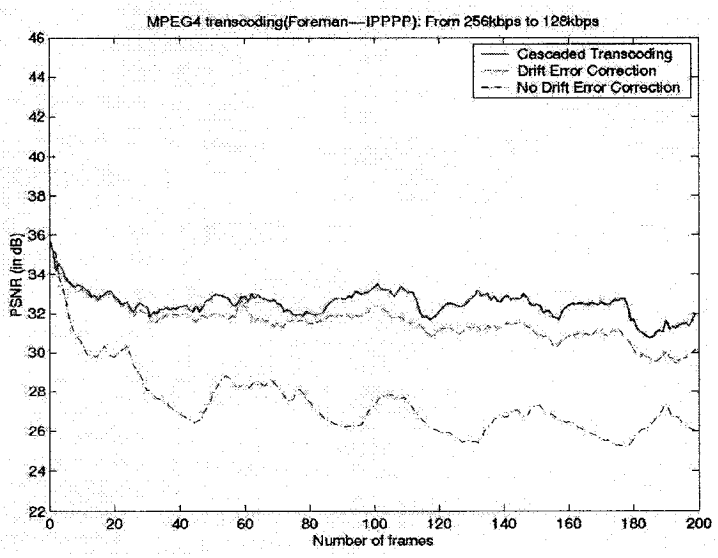


Figure 3.4 PSNR compared to cascaded transcoding and transcoding using drift error correction or no drift correction, all bitstreams are with N=200, M=1. *Foreman*: transcoding from 256kbps to 128kbps

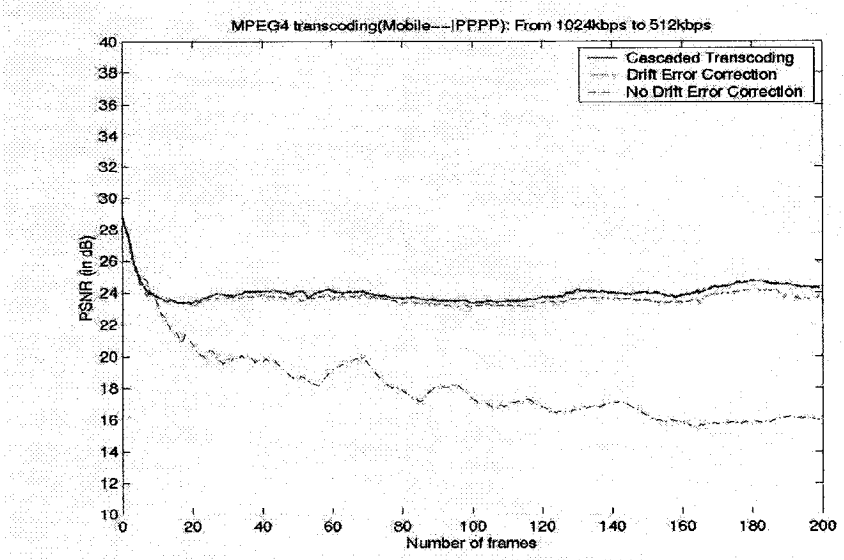


Figure 3.5 PSNR compared to cascaded transcoding and transcoding using drift error correction or no drift correction, all bitstreams are with N=200, M=1. *Mobile*: transcoding from 1.024Mbps to 512kbps

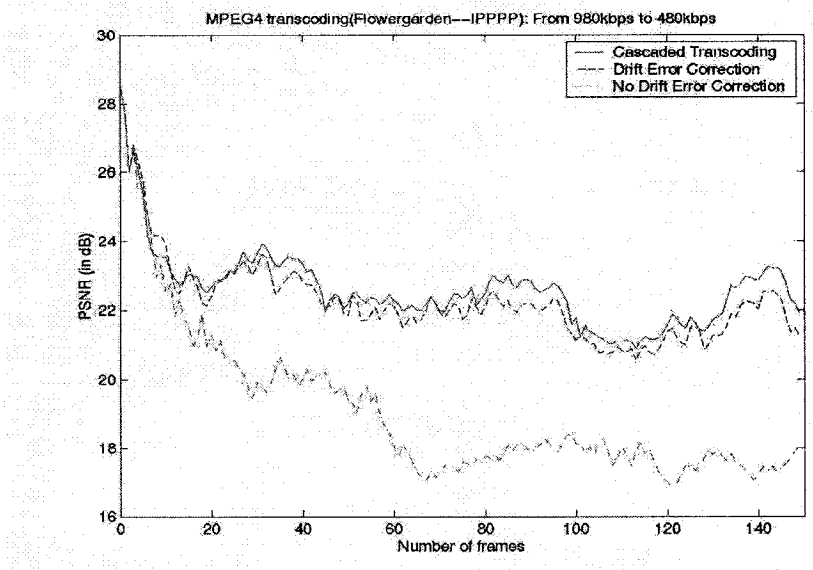


Figure 3.6 PSNR compared to cascaded transcoding and transcoding using drift error correction or no drift correction, all bitstreams are with N=150, M=1. *Flowergarden*: transcoding from 980kbps to 480kbps

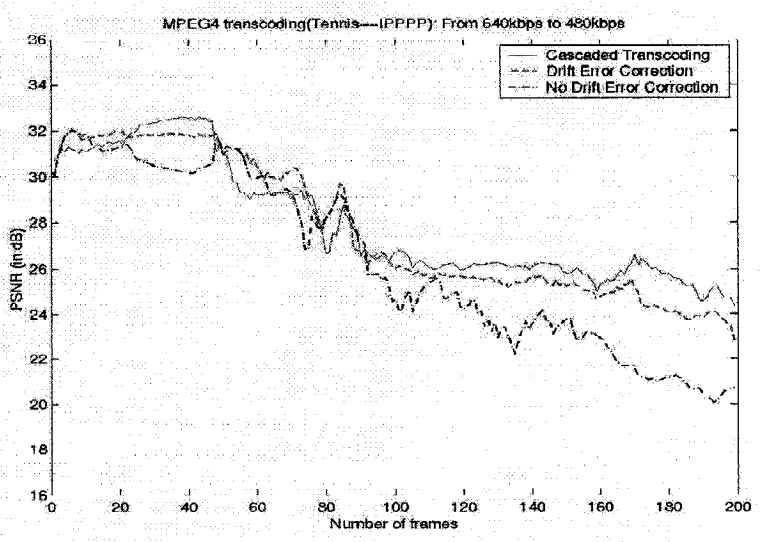


Figure 3.7 PSNR compared to cascaded transcoding and transcoding using drift error correction or no drift correction, all bitstreams are with N=200, M=1. *Tennis*: transcoding from 640kMbps to 480kbps

Table 3.1 Average *PSNR* for transcoding of four sequences with a GOP structures N=200, M=1. Cascaded transcoding $PSNR_{csd}$, Drift Error Correction $PSNR_{dec}$ and Without Drift Error Correction transcoding $PSNR_{ndc}$

Transcoding PSNR (in dB)				
Case	Foreman	Mobile	Flowergarden	Tennis
R2/R1	2.00	2.00	2.00	1.33
$PSNR_{csd}$	32.48	23.99	22.61	27.99
$PSNR_{dec}$	31.55	23.69	22.27	27.71
$PSNR_{ndc}$	27.45	18.16	19.07	26.27
$\Delta 1(PSNR_{dec} - PSNR_{csd})$	-0.93	-0.30	-0.34	-0.28
$\Delta 2(PSNR_{ndc} - PSNR_{csd})$	-5.03	-5.83	-3.54	-1.72
$\Delta 1 - \Delta 2$ (PSNR Gain)	4.10	5.53	3.20	1.44

The average *PSNR* of each transcoded sequence is analyzed in Table 3.1, we can observe that among our experiments, the performance of cascaded transcoder is the best, and the performance without drift error correction is the worst. Compared to those without drift correction, the performance of the transcoder with drift correction is improved about *3.20 dB (Flowergarden)*, and the best is high to *5.53 dB (Mobile)* when the transcoding ratio $R2/R1 = 2$. However, the difference between the transcoder with drift error correction and the cascaded transcoder is much smaller. When the transcoding ratio $R2/R1$ decreases, the difference between drift correction and no drift correction decreases. This can be seen from the transcoding of sequence *Tennis* from *640kbps* to *480kbps*. Those results show that in order to achieve the high quality of transcoded MPEG-4 compressed video, the drift error introduced by requantizer in the transcoder should be compensated, especially in the case of large ratio of transcoding. Figure 3.8-Figure 3.11 show the subjective images of the cascaded transcoding, TWDEC, and TWODEC of 4 different sequences: *Foreman, Mobile, Flowergarden, and Tennis*. Clearly, the cascade method outperforms drift error correction method since it fully decodes the incoming bitstream, then redo motion estimation, and modify macroblock coding pattern so as to compensate the drift error and more efficiently encode the sequence, as a result, the quality of reencoded video is improved. As we can see from Figure 3.8(d) to Figure 3.11(d), TWODEC results in the worst performance. With drift error correction, the performance will be improved and is comparable to the cascaded transcoding method.

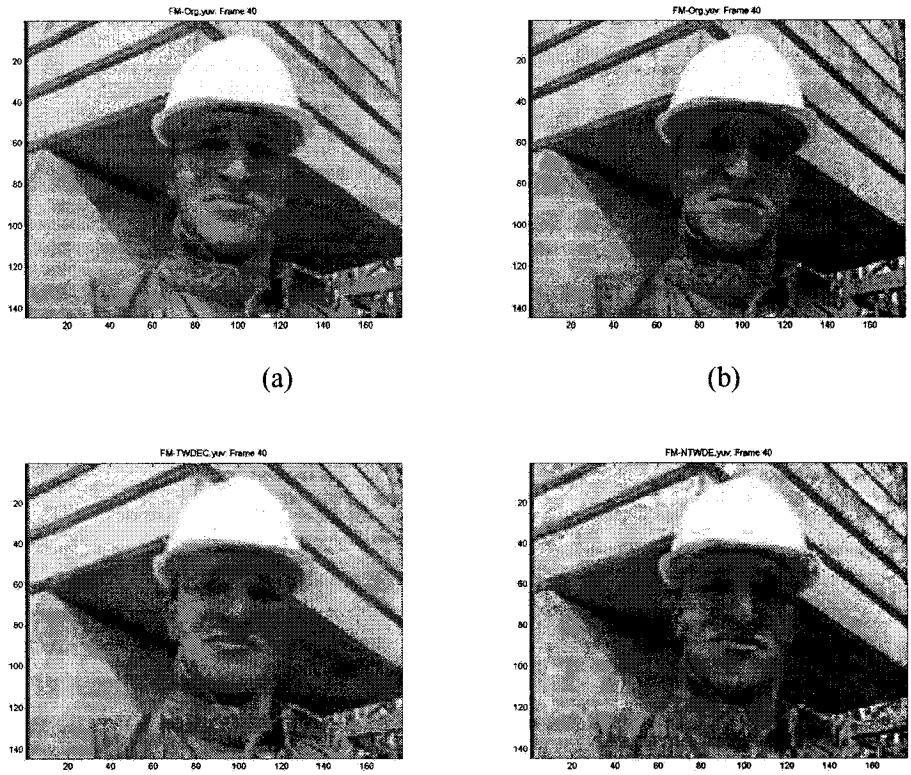


Figure 3.8 Illustration of transcoding effects by different methods: *Foreman*. (a) Original image at frame 40. (b) Reconstructed image at frame 40 by cascaded transcoding method. (c) Reconstructed image at frame 40 by TWDEC method. (d) Reconstructed image at frame 40 by TWODEC method.

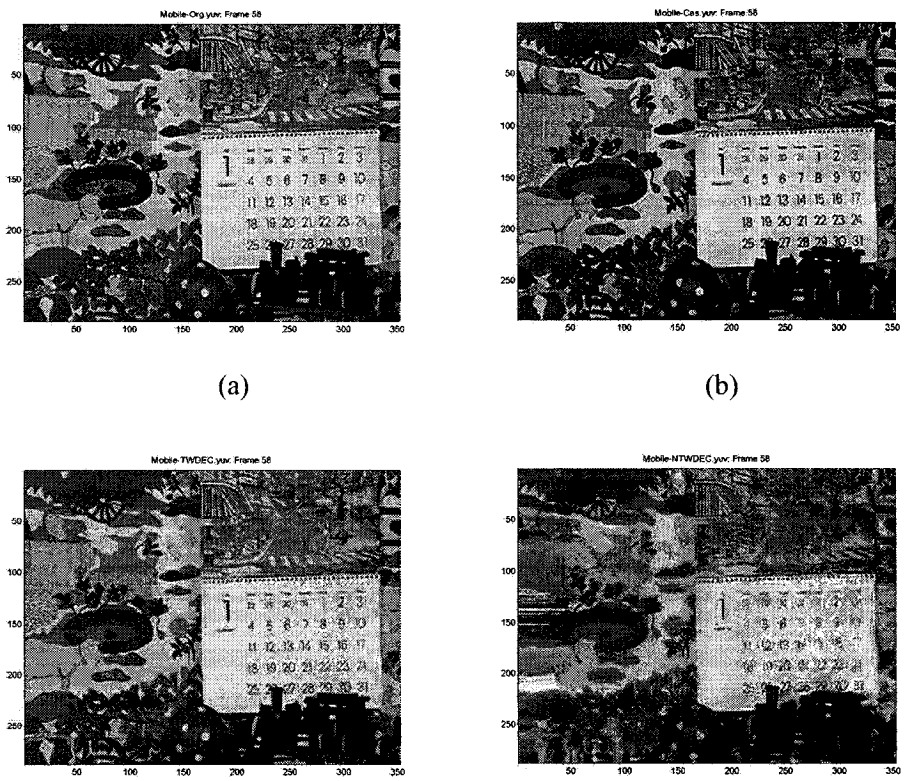


Figure 3.9 Illustration of transcoding effects by different methods: *Mobile*. (a) Original image at frame 58. (b) Reconstructed image at frame 58 by cascaded transcoding method. (c) Reconstructed image at frame 58 by TWDEC method. (d) Reconstructed image at frame 58 by TWODEC method.

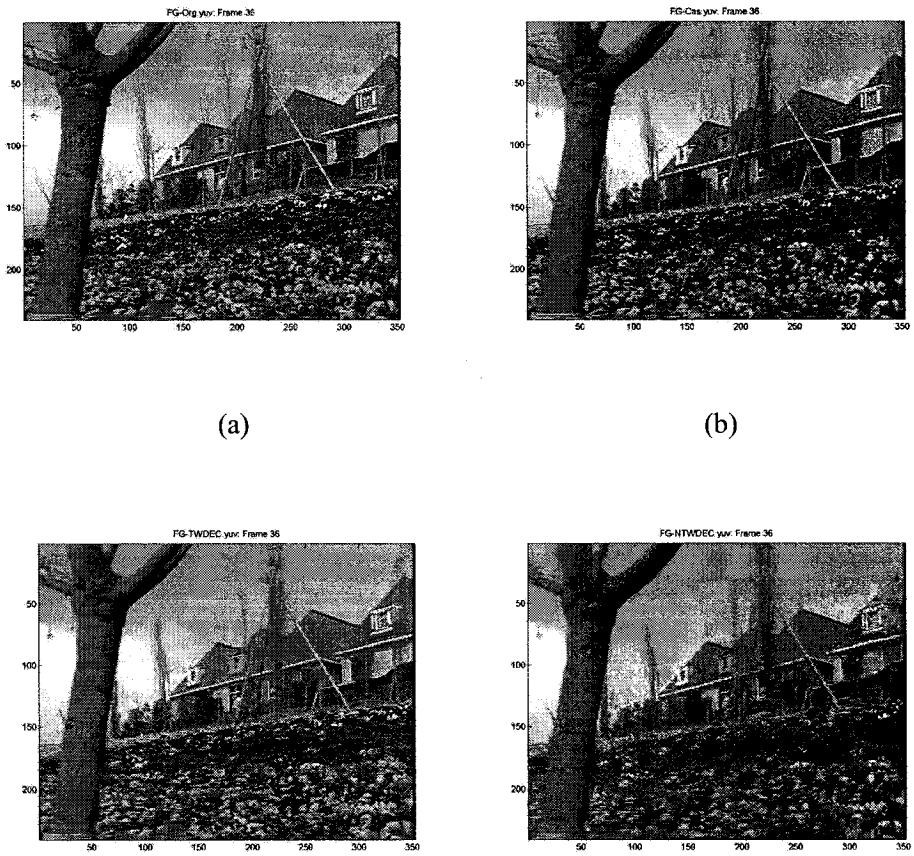
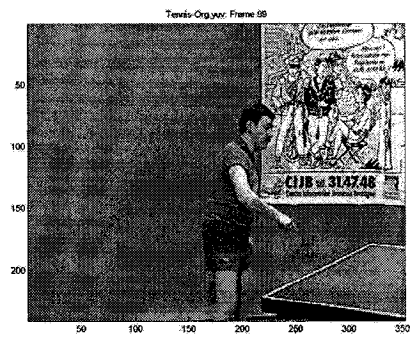
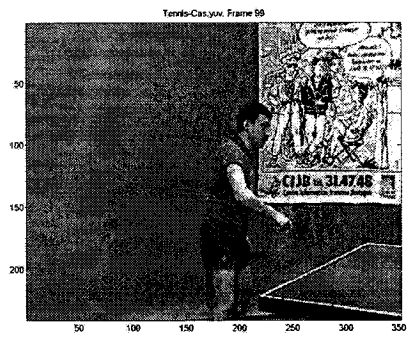


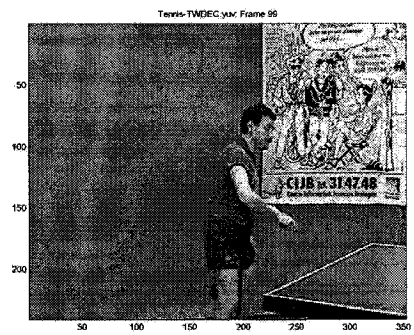
Figure 3.10 Illustration of transcoding effects by different methods: *Flowergarden* (a) Original image at frame 36. (b) Reconstructed image at frame 36 by cascaded transcoding method. (c) Reconstructed image at frame 36 by TWDEC method. (d) Reconstructed image at frame 36 by TWODEC method.



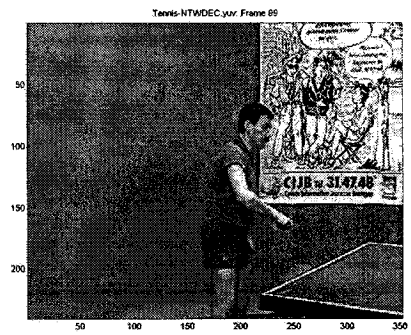
(a)



(b)



(c)



(d)

Figure 3.11 Illustration of transcoding effects by different methods: *Tennis* (a) Original image at frame 99. (b) Reconstructed image at frame 99 by cascaded transcoding method. (c) Reconstructed image at frame 99 by TWDEC method. (d) Reconstructed image at frame 99 by TWODEC method.

3.2 Joint Transcoding of Multiple MPEG-4 Compressed Video

In the previous section, we discussed transcoding of single streams of MPEG-4 compressed video from a high bitrate to a lower bitrate in order to meet the requirement of transmission channels with limited bandwidth. As we can observe from the simulation results, the drift error correction is very important in improving the quality of transcoded video, especially at larger transcoding ratio. In some applications, multiple programs should be transmitted simultaneously over a single communication channel. Consequently, transcoding techniques of multiple compressed videos are necessary for converting a higher bitrate of multiple compressed videos to a lower one. There are two approaches to solving this problem, one is independent transcoding, and the other is joint transcoding. The independent transcoding is to transcode each compressed video according to average allocated bitrate; however, joint transcoding is to allocate the bitrate for each video according to its properties such as the moving activities. In [8] , one joint transcoding scheme for minimizing the quality-variation among multiple transcoded bitstreams has been proposed. Their simulation results show joint transcoding can distribute the channel capacity among several incoming sequences according to their complexity so as to reduce the overall quality variation, compared to the independent transcoding method. In the following section, joint transcoding of multiple MPEG-4 compressed video bitstreams is discussed.

3.2.1. Joint Transcoding of Multiple MPEG-4 Compressed Videos

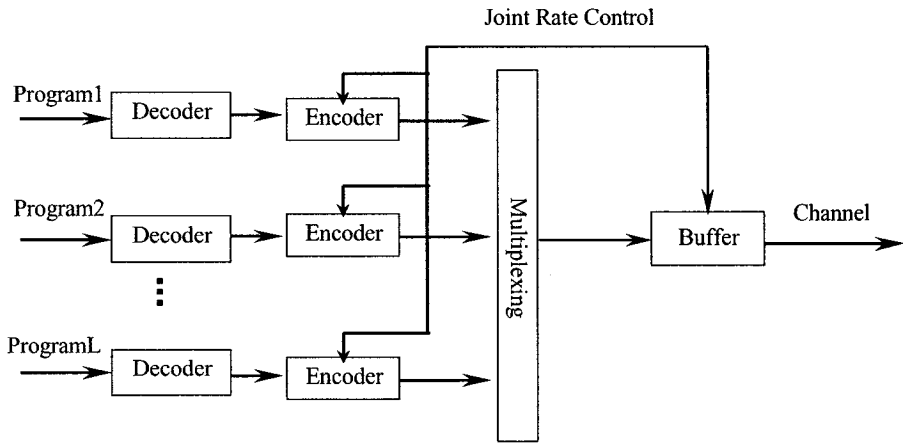


Figure 3.12 Joint transcoding: the programs are transcoded jointly; a joint rate control is included

Figure 3.12 shows the diagram of joint transcoding of multiple programs. Similar to joint coding[34][38], joint transcoding also needs to maintain the total bit rate constant, while allowing the bit rate allocated for each individual program to change. For example, some bits will be moved from the less active programs to the more active programs so as to maintain the uniform picture quality. To some extent, since transcoder consists of one decoder and one encoder, joint bitrate control algorithms used in joint coding can also be applied for that in joint transcoding.

Many researchers addressed the problem of transmitting multiple VBR videos over a shared communications channel [34][36][37] when the compression being done is the original compression. In[34], a joint bit allocation strategy for joint coding of multiple video programs is presented. It is shown that joint coding results in a uniform picture quality among programs as well as within a program. In[36], a method to transmit

multiplexed VBR video to improve video quality and/or transmission capacity is presented. The proposed method uses a multichannel rate control algorithm that allows statistical multiplexing of VBR video to regulate encoders during buffer congestion periods. The joint rate control algorithm is engaged when channel bandwidth overflow is predicted. The available bandwidth is then distributed among the video programs depending on their relative complexity. In[37], lossless aggregation, a scheme for the delivery of VBR video streams from a video server to a group of users over a shared CBR channel is presented. The transmission schedules of the individual video streams are determined dynamically depending on their relative traffic characteristics. It is shown that the aggregation method achieves buffer reduction compared to independent transmission of individual video streams over separate channels. In[8], the Root-Mean-Square-Error (*RMSE*) of requantization DCT coefficients is used to calculate the global complexity in the joint transcoding, and each transcoder produces a variable bit-rate stream. Simulation results shows it reduces the variation in picture quality between the video sequences as well as within each sequence, compared to independent transcoding.

In MPEG-4 transcoding, because it allows multiple VOPs to be independently or globally encoded, it is possible to use this feature to achieve joint transcoding of multiple MPEG-4 compressed videos (each frame of different sequence is regarded as a special "VOP", multiple such "VOPs" generate a special super "VO"). In the following section, three bit allocation schemes used for joint transcoding will be discussed.

In a Joint coding scheme, the goal is to distribute the channel capacity among programs according to video complexity, where more bits are allocated to the more demanding (from a coding point of view) programs than to the easier programs, therefore

resulting in a relatively uniform picture quality among programs. One way to do this is to use the same quantization parameter or the same video complexity for all programs at each super "VO". In order to employ the multiple-VOP concept to implement joint transcoding of multiple MPEG-4 programs, every frame of different sequences is regarded as one VOP object while each object is a rectangular picture with the same size as its original one. The new super "VO" is composed of these "VOPs". Thus, the joint transcoding method in MPEG-2[8] can be used in joint transcoding of multiple MPEG-4 compressed videos.

Assume there are L MPEG-4 compressed programs in a transmission system. We define a super "VO" as a collection of N "VOPs", one from each of the L special "VOP-like" programs at the same time instant. For the sake of simplicity, we further assume that the N "VOPs" in a super "VO" are of the same VOP type, therefore the VOPs are of the same VOP type. (In fact, these "VOP" type can be different in a super "VO" [34]). Implementation of joint transcoder mainly consists of two steps: the first step is to set a target bit number for a super "VO"; the second step is to allocate the bits to each VOP within the super "VO". In the following sections, several joint bit allocation methods are discussed briefly.

3.2.2. Bit Allocation at the Super "VO" Level

Assume there are L different programs, and at a certain time, these L frames from L different programs generates a super frame. If each program has the same length N of GOP, then a super GOP is defined as a super group of N super frames. Clearly, each super GOP contains $L \times N$ frames. Here, we treat one super frame as one super "VO", the

global bit rate is R, and the frame rate of each program is R_f, then the target number of bits T_g for one super group is given by[8]:

$$T_g = L \times N \times \frac{R}{R_f} \tag{3.4}$$

Let T_n be the target number of bits for a super "VO" *n*. Let $T(m,n,t)$ be the target number of bits for the sequence *m* of the super "VO" *n* with the "VOP" type *t* (I, B, or P frame); therefore, the sum of $T(m,n,t)$ over all sequences in super "VO" n should be equal to T_n:

$$\sum_{m=1}^{L} T(m,n,t) = T_n \tag{3.5}$$

and

$$\sum_{n=1}^{N} T_n = T_g \tag{3.6}$$

The task here is how to allocate the bit counts for different sequences in order to achieve a uniform quality of multiple programs. There are several methods such as TM5-like sequence complexity measurement[34][38] and RMSE-based measurement[8] to reallocate bit counts for different sequences.

In this thesis, a new *approximated distortion-based* complexity measurement used for joint transcoding is proposed. In the following paragraphs, these three methods will be introduced.

Assume that $X(m,n,t)$ is the global complexity of sequence *m* of super "VO" *n* with type *t (I, B, or P)*, $B(m,n,t)$ is the corresponding actual coded number of bits.

(1). TM5-like method

$$X(m,n,t) = B(m,n,t) \times Q(m,n,t) \tag{3.7}$$

Where, $Q(m,n,t)$ is the average quantization parameter for the previous coded "VOP" m of super "VO" n with the type t.

(2). RMSE-based method

$$X(m,n,t) = B(m,n,t) \times e(m,n,t) \tag{3.8}$$

Where, $e(m,n,t)$ is the Root-Mean-Square-Error (RMSE), or the quantization error, which is defined as:

$$e(m,n,t) = \sqrt{\frac{1}{64 \times N} \sum_{i=1}^{N} \sum_{u=0}^{7} \sum_{v=0}^{7} \left(Error_{m,n,t}^{i}(u,v) \right)^2} \tag{3.9}$$

Where, $Error_{m,n,t}(u,v)$ is the inverse DCT (IDCT) coefficient of the ith 8×8 DCT block of the reconstructed error between the inverse output and the input the requantizer of the transcoder, N is the total number of DCT blocks within a "VOP".

(3). Approximated distortion-based method

$$X(m,n,t) = B(m,n,t) \times MAD(m,n,t) \tag{3.10}$$

Where, $MAD(m,n,t)$ is the approximate distortion per pixel which is the *Mean Absolute Difference* of DCT Coefficients between original and reconstructed signal and given as:

$$MAD(m,n,t) = \sqrt{\frac{1}{64 \times N} \sum_{l=1}^{N} \sum_{u=0}^{7} \sum_{v=0}^{7} \left| C_{m,n,t}^{l}(u,v) \times \xi_{m,n,t}^{l} \right|^2} \tag{3.11}$$

Where, $C_{m,n,t}^{l}(u,v)$ are the lth 8×8 DCT coefficients of residual signals, which are

the inputs of requantizer; $\xi^{l}_{m,n,t}$ are the corresponding weighing factors which can be set to those of an MPEG quantization matrix or simply equal to 1[35].

Given the global complexity $X(m,n,t)$ from equations(3.7), (3.8), and (3.10), we can find the target bit counts for the sequence n of super "VO" m with "VOP" type t[8]:

$$T(m,n,t)=\frac{\frac{1}{K_{m,n}}X(m,n,t)}{\sum_{i=1}^{L}\frac{1}{K_{i,n}}X(i,n,t)}T_n \tag{3.12}$$

Clearly, the target number of bits for super "VO" m is proportional to its complexity. Note that the above equation depends on the global complexities of current and future VO. Since future complexities are not available, one uses the most recent complexity measures for each sequence and for each picture type.

3.2.3. Bit Allocation at the Super GOP Level

At the start of transcoding, T_g is assigned according to Equation (3.4). Let R_r be the remaining number of bits after this super group is coded, and then R_r should be updated in the next super GOP:

$$R_r = R_r + T_g \tag{3.13}$$

Accordingly, within a super GOP, after coding the (j-1) the super "VO", R_r should be updated as:

$$R_r = R_r - \sum_{m=1}^{N} B(j-1,m,t) \tag{3.14}$$

Where $B(n,m,t)$ is the actual number of bits coded for sequence m of super "VO" n with frame type t. It should closely equal to $T(n,m,t)$.

3.2.4. Simulation results

In the joint transcoder of this thesis, UPM1 rate control model is employed because it globally controls the bitrate for different VOPs. But some modifications have been done according to equations from (3.4) to (3.14). First of all, each sequence is regarded as one VOP. Second, the parameter that is used to indicate the importance of each VOP is replaced with the global complexity. Third, before transcoding a superframe, the target bitrate for each "VOP" is calculated according to Equ. (3.12).

The following are the simulation results based on above assumptions. Here, 150 frames (each with size 352×240) of the sequences *Flower Garden, Football, Table Tennis, Salesman, and Miss American* are used with a color sampling ratio of *4:2:0.* The GOP length is *150* with a structure *IPPP... IPPP...* .The global bit rates of input and output of transcoder are *4.80Mbps* and *2.56Mbps*. Input sequences are individually encoded at *960kbps*, and frame rate is 30frames/second. In the independent transcoding, each sequence is independently transcoded to *512kbps*. As for joint transcoding, three bitrate re-allocation algorithms (TM5-like, RMSE-based and Approximated Distortion-based) introduced in section 3.2.2 are implemented with drift error correction. Simulation results are shown in Figure 3.13-Figure 3.16 In order to quantitavely compare the performance of each joint transcoding with independent transcoding, Table 3.2-Table 3.4 show statistical PSNRs of each sequence with different joint transcoding schemes. Table 3.5 tells us their statistical reallocated bitrates for different sequence.

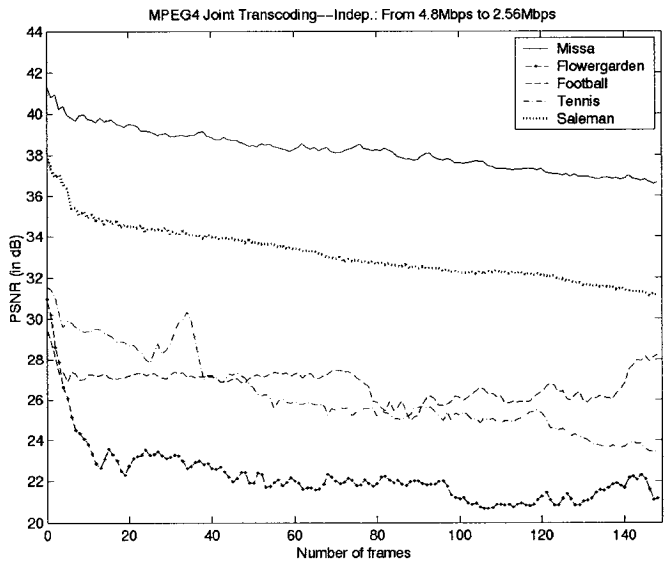


Figure 3.13 Independent transcoding of five sequences; each transcoded from 960kbps to 512kbps

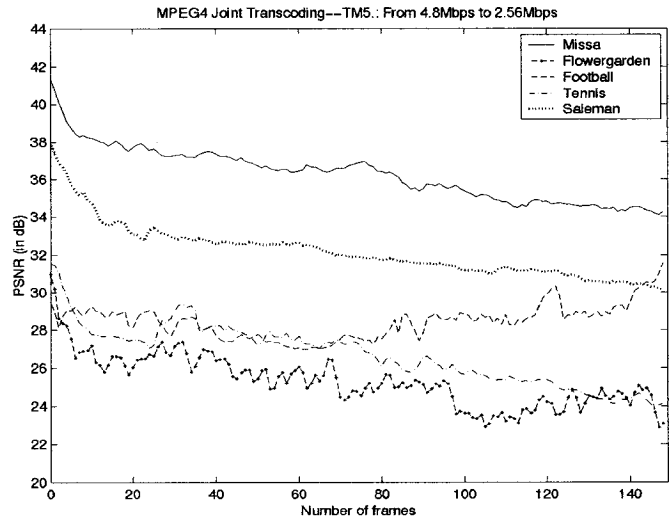


Figure 3.14 Joint transcoding of five sequences based on TM5-like complexity

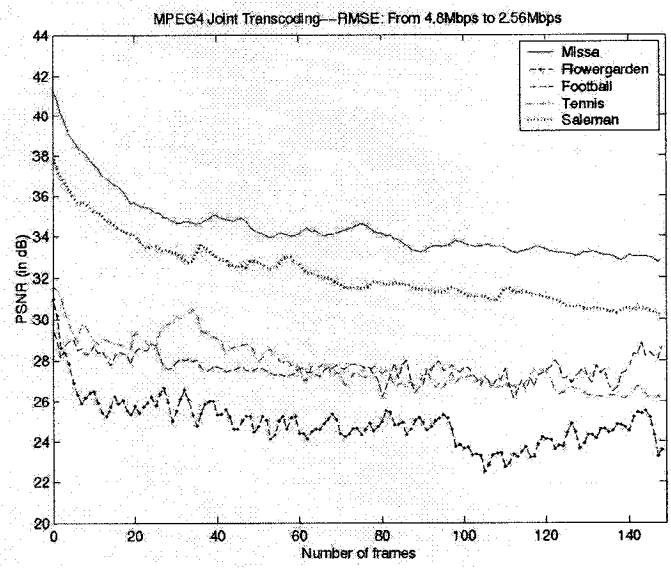


Figure 3.15 Joint transcoding of five sequences based on the Root Square Mean Error (RMSE)

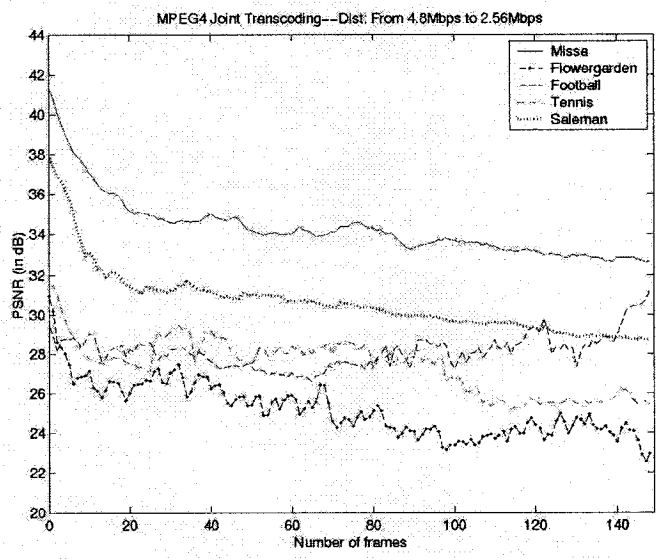


Figure 3.16 Joint transcoding of five sequences based on approximate distortion

Table 3.2 PSNR of joint transcoding with TM5-like complexity measure and Independent transcoding. The Total input and output bit rates of the five sequences are 4.80Mbps and 2.56Mbps respectively

Transcoding with drift error correction								
Case	PSNR (in dB)							
	Flr.	Ftb.	Ten.	Slm.	Miss.	Mean	Min.	Var.
Joint (TM5)	26.6	29.4	28.5	33.8	36.9	31.1	23.7	15.9
Independent	23.7	28.0	28.2	35.1	38.4	30.7	21.4	30.7
Δ (Joint-Indep)	2.9	1.4	0.3	-1.3	-1.5	0.4	2.30	-14.8

Table 3.3 PSNR of joint transcoding with RMSE-based complexity measure and Independent transcoding. The Total input and output bit rates of the five sequences are 4.80Mbps and 2.56Mbps respectively

Transcoding with drift error correction								
Case	PSNR (in dB)							
	Flr.	Ftb.	Ten.	Slm.	Miss.	Mean	Min.	Var.
Joint (RMSE)	26.1	28.8	29.4	33.9	35.3	30.7	23.0	13.5
Independent	23.7	28.0	28.2	35.1	38.4	30.7	21.4	30.7
Δ (Joint-Indep)	2.4	0.8	1.2	-1.2	-3.1	0.0	1.6	-16.8

Table 3.4 PSNR of joint transcoding with Approximated Distortion-based complexity measure and Independent transcoding. The Total input and output bit rates of the five sequences are 4.80Mbps and 2.56Mbps respectively

Transcoding with drift error correction								
Case	PSNR (in dB)							
	Flr.	Ftb.	Ten.	Slm.	Miss.	Mean	Min.	Var.
Joint (Distortion)	26.4	29.6	28.6	32.5	35.1	30.4	23.7	11.4
Independent	23.7	28.0	28.2	35.1	38.4	30.7	21.4	30.7
Δ (Joint-Indep)	2.7	1.6	0.4	-2.6	-3.3	-0.3	2.3	-18.7

Table 3.5. The individual output bit rate after transcoding (The global input bitrate is 4800kbps, the target bit rate is 2560kbps)

Case	Bit rate reallocated for each sequence (in Kbps)					
	Flr.	Ftb.	Ten.	Slm.	Miss.	Total
Independent	508	496	484	337	509	2334
Joint (TM5)	885	715	540	158	107	2405
Joint (RMSE)	876	632	678	166	90	2442
Joint (Dist.)	833	727	568	125	89	2342

From above simulation results, in terms of variation and minimum PSNR, Approximated Distortion method outperforms the other two methods, the difference of variation between Distortion and Independent is high to 18.7, the difference of minimum PSNR is up to 2.3dB, and the average PSNR is almost the same as the independent transcoding. The effect of joint transcoding is largely dependent on the complexity of

input sequences, if the difference between input sequences is larger, then the performance of joint transcoding is better.

3.3. Conclusions

In this Chapter, first, we discuss two types of architectures of transcoding of MPEG-4 compressed bitstreams: transcoding without drift error correction (TWODEC) and transcoding with drift error correction (TWDEC). TWODEC is a simple implementation of transcoding, and is suitable for low transcoding ratio (R1/R2), where low latency is required. However, for higher transcoding ratios (R1/R2), it results in degradation of performance due to large drift errors introduced by the requantization. TWDEC is a feasible and reliable solution to this problem. TWDEC differs from the cascaded transcoding in that it partially decodes the incoming bitstreams and reuses motion vectors, and therefore, reduces a computational complexity mainly by avoiding motion estimation. Simulation results show that TWDEC greatly outperforms that of TWODEC, and is close to that of cascaded transcoding. For example, when the transcoding ratio is $R2/R1 = 2$, TWDEC of *Mobile* sequence improve the performance in 5.53dB PSNR over that without drift error correction, while the difference (degradation of performance in PSNR) between drift error correction method and cascaded method is only 0.3dB.

In this Chapter, another transcoding technique introduced is joint transcoding of multiple MPEG-4 compressed bitstreams. Joint transcoding is used to improve overall video quality of multiple programs sharing the same transmission channel. According to the different complexity of each sequence, the bit rate for each sequence is allocated so as to reduce the quality variation between different programs. Besides independent

transcoding, TM5-like, and RMSE-based complexity measurements, a new Approximated Distortion-based complexity algorithm is proposed in this thesis. Simulation results indicate that by allocating bitrate for each sequence according to its complexity, the overall performance of TM5-like, RMSE-based, and Approximated Distortion-based joint transcoding outperforms that of independent transcoding in terms of variation and minimum PSNR. It should be noted that the improvement of performance of joint transcoding is dependent on the motion activity of input sequences. If the motion characteristics of each sequence are almost the same, the joint transcoding can simply be replaced by independent transcoding.

4. Transcoding of MPEG-4 Compressed Video in the DCT Domain

In Chapter 3, (joint) transcoding of MPEG-4 compressed bitstreams in the pixel domain was discussed. In this chapter, we will address DCT domain transcoding of MPEG-4 compressed as well as frame-skipping transcoder implemented in the DCT domain.

Recently, the technique of manipulation and composition in the DCT domain[20] has been applied to transcoding. This has some advantages and disadvantages. For example, in the compressed domain (DCT domain), two I (DCT) operations are removed. However, the inverse DCT motion compensation becomes the bottleneck for DCT domain transcoding. Several fast MC-DCT algorithms have been addressed[22][24]. Compared to the pixel domain approaches, the algorithm proposed in [22] can obtain 32% computational complexity savings, and the method using approximate matrices in [24] can provide a reduction of 81% in computational complexity. In [23], another local bandwidth constrained inverse motion compensation algorithm was proposed, in which only those DCT coefficients inside the estimated bandwidth are computed. This algorithm achieves 25%~55% computational improvement over the method proposed in[20]. They also present a Look Up Table (LUT)-based implementation scheme to obtain further another 31%~48% improvement in computation.

In this chapter, we will first discuss the structure of MPEG-4 transcoder and preliminary of implementing the transcoder in the DCT domain. We will also discuss some issues related to the transcoder in the DCT domain. In addition, a new DCT Coefficient Translation and Truncation Transformation Matrix (**DCTTTM**) based motion

compensation algorithm used for transcoding of MPEG-4 compressed bitstreams is proposed, and a suboptimal frame-skipping transcoding scheme is also discussed.

4.1. The Architecture of Transcoding of MPEG-4 Compressed Video in the DCT Domain

Figure 4.1 shows the structure of an MPEG-4 transcoder in the DCT domain with drift error correction. Compared to that in the pixel domain given in Figure 4.2, 2 (IDCT/DCT) operations are saved. All operations are done in the DCT domain, similar to that in the pixel domain, shape information is passed directly to the output, it is not considered to be transcoded. Although there are no (IDCT/DCT) operations, inverse motion compensation in the DCT domain is more complicated than that in the pixel domain. In the next section, composition and manipulation of blocks, which are used in the transcoding of MPEG-4 in the DCT domain is discussed.

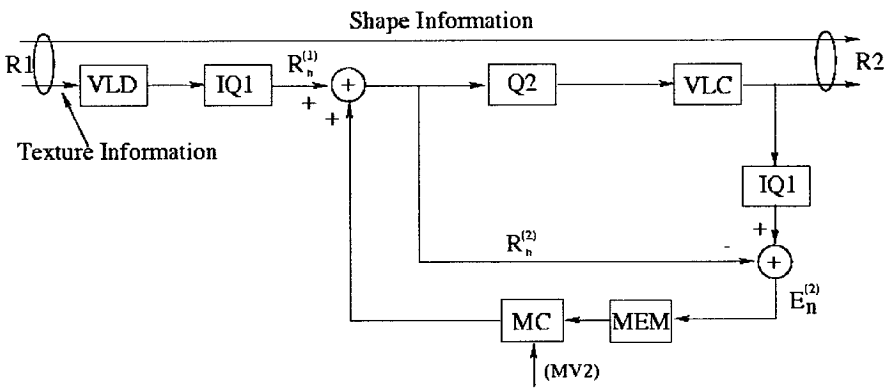


Figure 4.1 The simple structure of MPEG-4 transcoder in the DCT domain

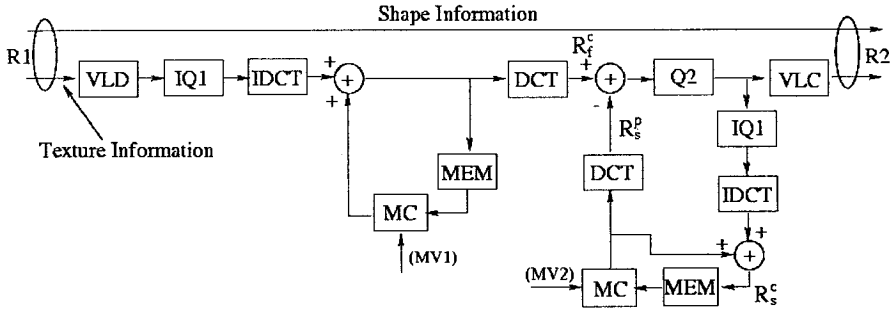


Figure 4.2 The simple structure of MPEG-4 transcoder in the pixel domain

4.2. DCT Domain Inverse Motion Compensation Algorithms

Recently, several DCT domain inverse motion compensation algorithms[22][23][24] are based on that proposed by Chang and Messerschmitt's algorithm[20]. However, in this thesis, a new motion compensation algorithm based on DCT Coefficient Translation and Truncation Transformation Matrix (**DCTTTM**) in the DCT domain is proposed. In the following subsection, these two algorithms will be introduced.

4.2.1. Chang and Messerschmitt's algorithm

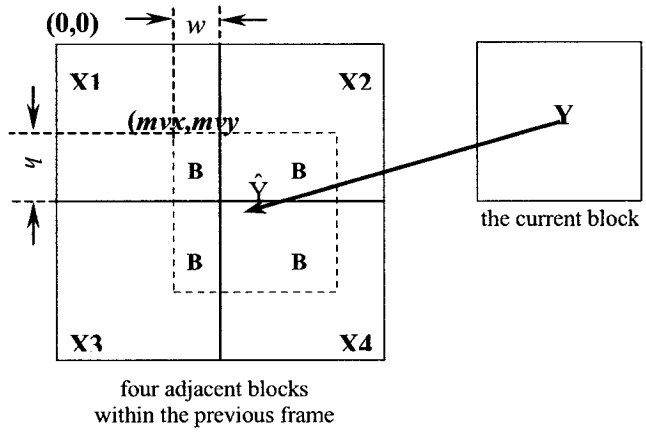


Figure 4.3 DCT domain based inverse motion compensation

In [20], Chang and Messerschmitt introduce algorithms to composite and manipulate blocks due to motion compensation. In motion compensation, because the motion vectors are usually not zero or an integer multiple of block size, the pixels from a previously encoded frame that provide an estimate for a block in the current frame usually come from four blocks in the reference block. Figure 4.3 shows the relationship of a 8×8 DCT block in the current frame Y with its motion compensated estimate $\hat{Y}$ from the reference frame. Note that $\hat{Y}$ overlaps four 8×8 DCT blocks in the reference frame. $\hat{Y}$ is composited from the four blocks X_1, X_2, X_3, and X_4. As in [20], $\hat{Y}$ can be expressed as:

$$DCT(\hat{Y}) = \sum_{i=1}^{4} DCT(H_{i1}) \times DCT(X_i) \times DCT(H_{i2}) \tag{4.1}$$

Where,

$$H_{21} = H_{11} = \begin{bmatrix} 0 & I_h \\ 0 & 0 \end{bmatrix} \tag{4.2}$$

$$H_{12} = H_{32} = \begin{bmatrix} 0 & 0 \\ I_w & 0 \end{bmatrix} \tag{4.3}$$

$$H_{31} = H_{41} = \begin{bmatrix} 0 & 0 \\ I_{8-h} & 0 \end{bmatrix} \tag{4.4}$$

$$H_{22} = H_{42} = \begin{bmatrix} 0 & I_{8-w} \\ 0 & 0 \end{bmatrix} \tag{4.5}$$

Where H's are all 8×8 matrices; I_h and I_w stand for the identity matrices with size $h\times h$ and $w\times w$, (h, w represents horizontal and vertical shift as seen in Figure 4.3). In

order to describe the above matrices more clearly, one example is that we set $h = 2$, $w = 3$, then we can derive the following matrices used for equation(4.1):

$$H_{21} = H_{11} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.6}$$

$$H_{12} = H_{32} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.7}$$

$$H_{31} = H_{41} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.8}$$

$$H_{22} = H_{42} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.9}$$

Note $1 \le h, w \le 7$. In order to compute the composited DCT block, $DCT(H_{i1})$ and $DCT(H_{i2})$ $(1 \le i \le 4)$ can be precomputed and stored in memory.

The reconstructed block Y at (x, y) in the DCT domain can be written as:

$$DCT(Y(x,y)) = DCT(\hat{Y}(x,y)) + DCT(e(x,y)) \tag{4.10}$$

Where, $e(x, y)$ is the residual signal.

In this thesis, we propose another new DCT Coefficient Translation and Truncation Transformation Matrix (**DCTTTM**) motion compensation (**MC**) algorithm that is suitable for the transcoder implemented in the DCT domain.

4.2.2. *DCTTTM*-based Motion Compensation

Since the DCT is a linear operation, the DCT of a 8×8 matrix X with 64 coefficients C_{ij} $(0 \le i, j \le 7)$ can be regarded as summations of 64 special matrices, that is:

$$X = \sum_{i=0}^{7} \sum_{j=0}^{7} C_{ij} \times DCT(Q_{ij}) \tag{4.11}$$

Where Q_{ij} is a simple 8×8 matrix, in which only its element $d'_{ij} = 1$, and the others

are set to 0. For example, when i=0, j=3, Q_{ij} can be expressed by:

$$Q_{03} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.12}$$

Thus, the IDCT of equation (4.11) can be represented as:

$$Y = IDCT(X) = \sum_{i=0}^{7}\sum_{j=0}^{7} C_{ij} \times IDCT(Q_{ij}) \tag{4.13}$$

Similar to the analysis in [20], one current block's estimate may straddle four adjacent blocks in the reference frame, and there are 256 coefficients with 256 locations used to calculate the reconstructed DCT block. In order to extract one DCT block from these four blocks X_1, X_2, X_3, and X_4 with displacements mvx and mvy, where mvx and mvy have the relationship with h, w as shown in Figure 4.3 assuming the block size is 8:

$$\begin{cases} mvx = 8 - w & ;0 \le w, mvx \le 8 \\ mvy = 8 - h & ;0 \le h, mvy \le 8 \end{cases} \tag{4.14}$$

At each location (i, j) in the block X_1, we first generate an inverse DCT of Q_{ij},

then according to the displacement values mvx and mvy, a new matrix $IDCT(Q_{ij}^{'})$ is created, and this new matrix is used to redo the DCT operation, finally a reconstructed new DCT matrix Q_{ijl} (l is the index of four adjacent block, and $1 \le l \le 4$) is derived for given mvx and mvy. Therefore, for given mvx and mvy, the extracted DCT block $\hat{Y}$ can be rewritten as:

$$\hat{Y} = \sum_{l=1}^{4} \sum_{i=0}^{7} \sum_{j=0}^{7} C_{ijl} \times Q_{ijl} \tag{4.15}$$

Where C_{ijl} and Q_{ijl} are the DCT coefficients and the responding weighting coefficients of lth block with the displacement mvx and mvy.

In order to illustrate this idea, one example is taken as follows:

Let $mvx = 3$, $mvy = 0$, $i = 0$, $j = 3$, and $l = 1$, then Q_{03} is the same as shown in equation (4.12), its inverse DCT matrix is:

$$IDCT(Q_{03}) = \begin{bmatrix} 0.147 & -0.0345 & -0.1734 & -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 \\ 0.147 & -0.0345 & -0.1734 & -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 \\ 0.147 & -0.0345 & -0.1734 & -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 \\ 0.147 & -0.0345 & -0.1734 & -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 \\ 0.147 & -0.0345 & -0.1734 & -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 \\ 0.147 & -0.0345 & -0.1734 & -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 \\ 0.147 & -0.0345 & -0.1734 & -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 \\ 0.147 & -0.0345 & -0.1734 & -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 \end{bmatrix} \tag{4.16}$$

Because $mvx = 3$, $mvy = 0$, we regenerate a new IDCT matrix by removing out the first three columns and left shifting the remaining 5 columns and then filling zeros into the empty places. The new IDCT matrix $IDCT(Q_{03}^{'})$ is then given by:

$$IDCT\left(Q_{03}^{'}\right)=\begin{bmatrix} -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 & 0 & 0 & 0 \\ -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 & 0 & 0 & 0 \\ -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 & 0 & 0 & 0 \\ -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 & 0 & 0 & 0 \\ -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 & 0 & 0 & 0 \\ -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 & 0 & 0 & 0 \\ -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 & 0 & 0 & 0 \\ -0.0982 & 0.0982 & 0.1734 & 0.0345 & -0.147 & 0 & 0 & 0 \end{bmatrix} \qquad (4.17)$$

Now, the new DCT matrix Q_{ijl} $(i=0, j=3, l=1)$ can be obtained as:

$$\begin{aligned} Q_{031} &= DCT\left(IDCT\left(Q_{03}^{'}\right)\right) \\ &= \begin{bmatrix} 0.0609 & 0.1656 & -0.022 & -0.5256 & -0.4823 & 0.0478 & 0.1059 & -0.1521 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \qquad (4.18)$$

For each pair of given displacements mvx and mvy, there will be 256 (8×8) matrices generated. If we convert one 8×8 matrix to one 64×1 matrix or vector, then for each pair of given displacements mvx and mvy, a 64×256 Mega-matrix W is generated. Similarly, the 8×8 matrix $\hat{X}$ can also be converted to a 64×1 matrix or vector $\hat{Z}$, and then, the equation(4.15) can be rewritten as:

$$\hat{Z} = \sum_{i=0}^{255} C_i \times W_i \qquad (4.19)$$

Where, C_i ($0 \le i \le 255$) is the DCT coefficient of 4 straddled blocks X_j ($1 \le j \le 4$), and

$W_i (0 \le i \le 255)$ is the corresponding 64×1 vector derived from the *ith* column of 64×256 matrix W.

Since $0 \le mvx \le 7$ and $0 \le mvy \le 7$, there are 64 combinations of mvx and mvy; therefore, there are sixty-four 64×256 matrices generated. These matrices could be pre-calculated during the initialization, and then stored. When motion compensation is executed, according to motion vectors, a pair of displacements mvx and mvy could be found out, then, the corresponding 64×256 matrix W can be looked up. By using equation), the estimated block $\hat{Y}$ can be deduced from the four straddled blocks in the reference frame.

Theoretically, the extracted compensated DCT block $\hat{Y}$ derived from equation (4.19) would be more complicated than that from equation (4.1). However, since there are lots of elements of matrices with very small value, they would be ignored and truncated by applying a threshold. Thus, a new DCT Coefficient Translation and Truncation Transformation Matrix (**DCTTTM**) motion compensation (**MC**) algorithm is proposed in this thesis.

4.2.2.1. Analysis of Computational Complexity of DCTTTM MC

From equation (4.19), the complexity of the **DCTTTM MC** method is related to the number of nonzero elements within the 64×256 matrix. However, the motion vectors or the displacements in both horizontal and vertical directions determine the number of nonzero elements. For given displacements mvx and mvy, the 64×256 matrix W is given. Figure 4.4-Figure 4.9 show nonzero-element distributions of 64×256 matrix W in the cases of different displacements of mvx and mvy.

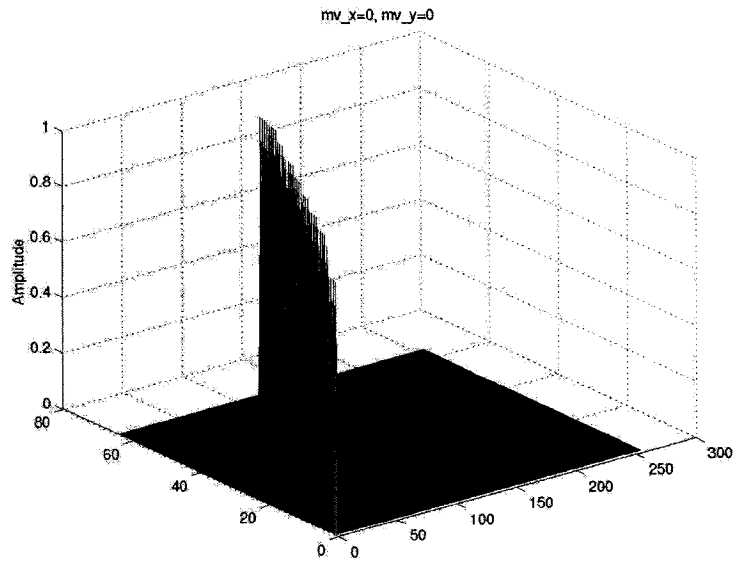


Figure 4.4 Displacements with mvx=0 and mvy=0.

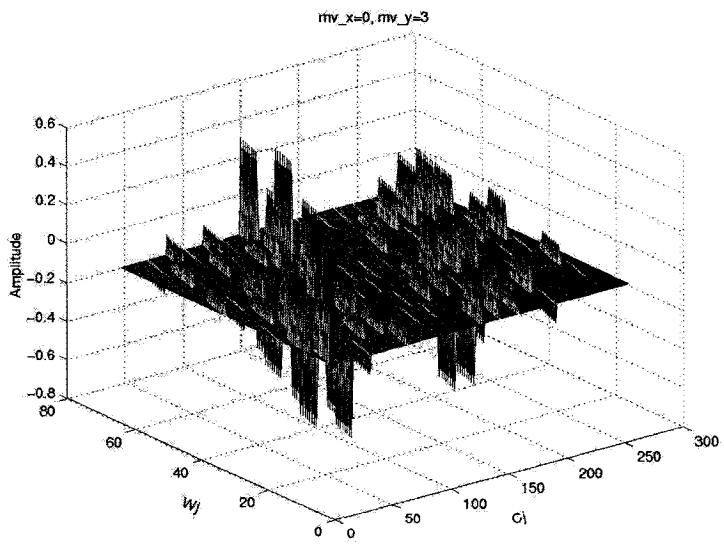


Figure 4.5 Displacements with mvx=0 and mvy=3.

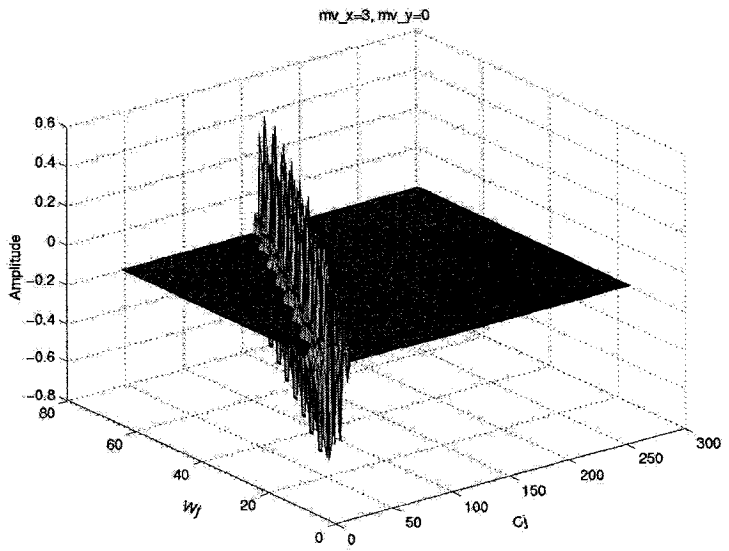


Figure 4.6 Displacements with mvx=3 and mvy=0.

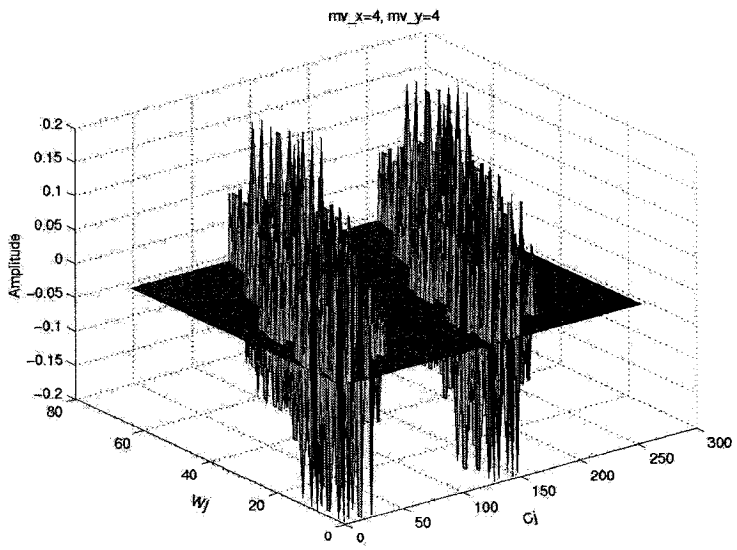


Figure 4.7 Displacements with mvx=4 and mvy=4.

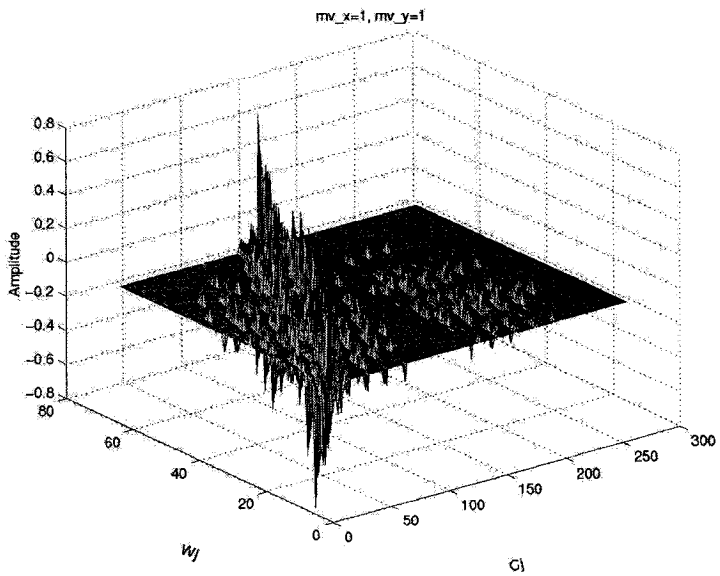


Figure 4.8 Displacements with mvx=1 and mvy=1.

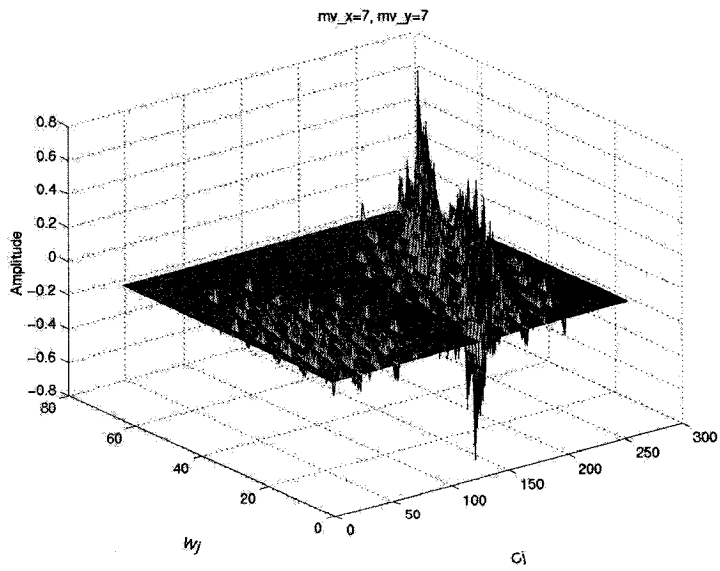


Figure 4.9 Displacements with mvx=7 and mvy=7.

As we can see from Figure 4.4 to Figure 4.9, when mvx=0 and mvy=0, only block X_1 contributes to the complexity and there are only 64 nonzero elements within the corresponding 64×256 matrix. In fact, we can ignore the complexity under this situation because we directly extract X_1 as $\hat{X}$, therefore, there no multiplications or additions at all. When there is only displacement in one direction, i.e., when $mvx \neq 0$ and mvy=0 or when mvx=0 and $mvy \neq 0$, there are exactly two blocks contributing to computational complexity, and in total 128 locations generate 128 (64×1) vectors, each vector has only at most 8 nonzero elements. In this case, there will be 1024 multiplications and 960 additions. The worst case is that neither of the displacements mvx and mvy is equal to zero. For example, mvx=1 and mvy=1, each location generates 64 non-zero element vector, 256 locations create 16384 nonzero elements within this 64×256 matrix, introducing 16384 multiplications and 16320 additions. This is a large computational complexity. However, there are many very small valued elements within each 64×256 matrix, which have less influence on the transcoded video quality. These small valued elements of matrices can be truncated to zero by different thresholds so that they are not considered to increase the number of multiplications and additions used for retrieve the compensated DCT block from the four adjacent blocks in the reference frame. Figure 4.10-Figure 4.14 show some examples of operational complexity with different thresholds, and Table 4.1 gives more detailed number of multiplications and additions with different mvx and mvy.

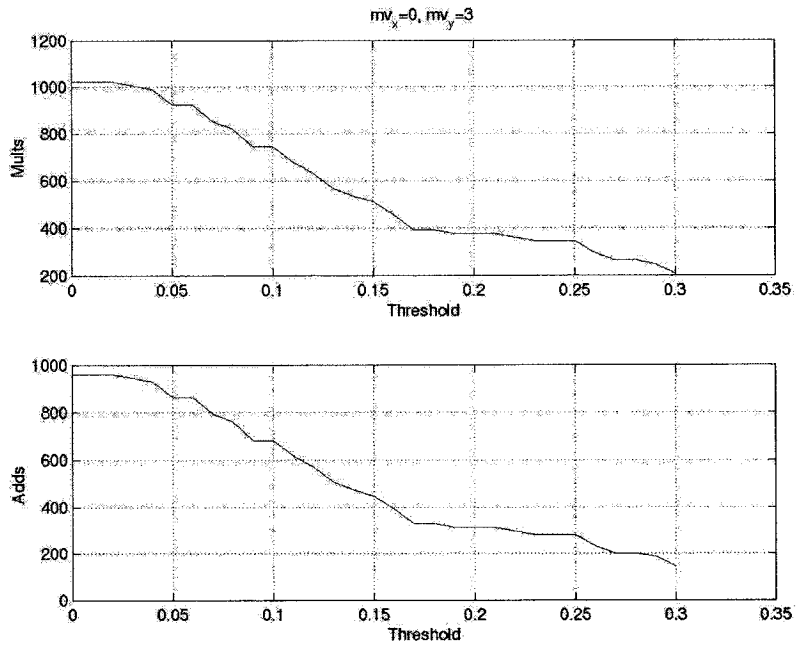


Figure 4.10 Multiplications and Additions when mvx=0 and mvy=3

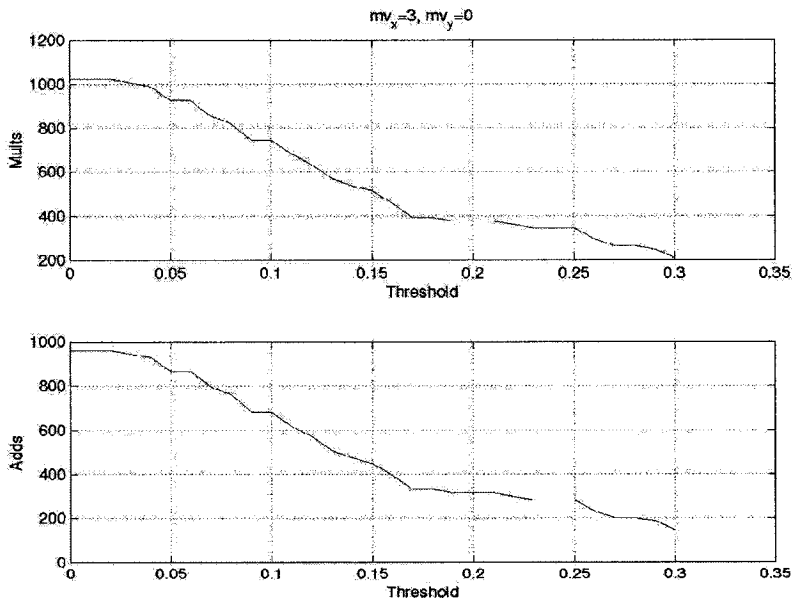


Figure 4.11 Multiplications and Additions when mvx=3 and mvy=0

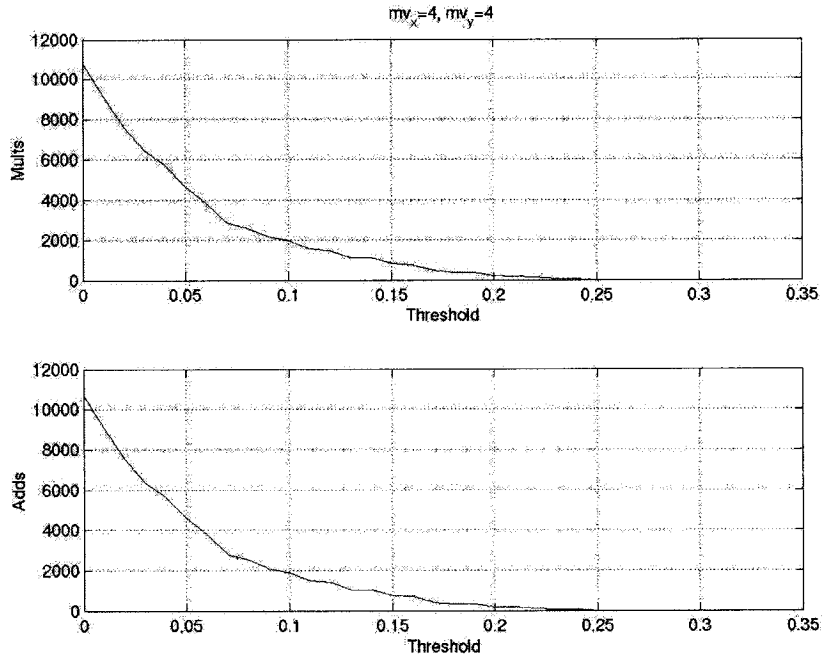


Figure 4.12 Multiplications and Additions when mvx=4 and mvy=4

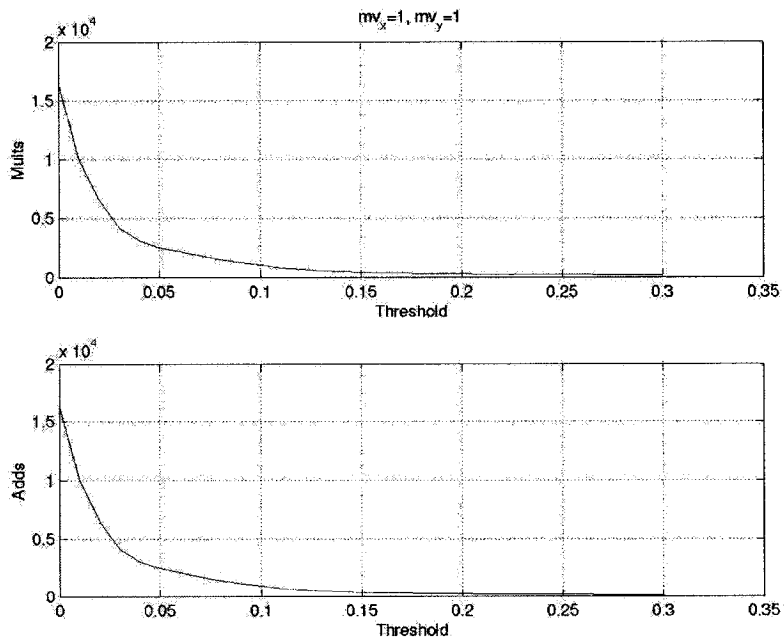


Figure 4.13 Multiplications and Additions when mvx=1 and mvy=1

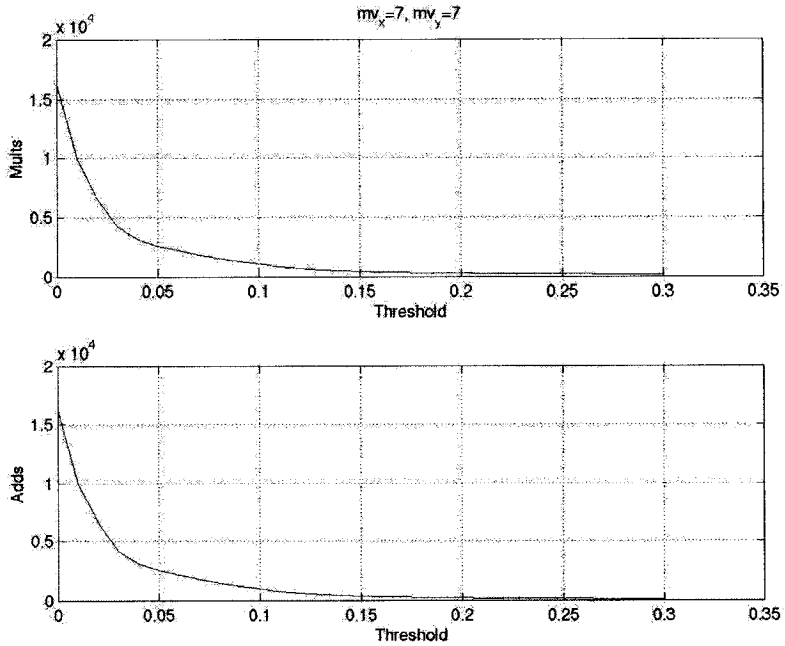


Figure 4.14 Multiplications and Additions when mvx=7 and mvy=7

Table 4.1 Multiplications and Additions with different thresholds, "Mults" means Multiplications, "Adds" means Additions.

Threshold	$mvx = 0$ $mvy = 3$		$mvx = 3$ $mvy = 0$		$mvx = 4$ $mvy = 4$		$mvx = 1$ $mvy = 1$		$mvx = 7$ $mvy = 7$	
	Mults	Adds	Mults	Adds	Mults	Adds	Mults	Adds	Mults	Adds
0.00	1024	960	1024	960	10816	10752	16384	16320	16384	16320
0.01	1024	960	1024	960	9152	9088	10069	10005	10069	10005
0.02	1024	960	1024	960	7584	7520	6564	6500	6564	6500
0.03	1008	944	1008	944	6448	6384	4229	4165	4229	4165
0.04	992	928	992	928	5712	5648	3151	3087	3151	3087
0.05	928	864	928	864	4672	4608	2585	2521	2585	2521

From Figure 4.10-Figure 4.14 and Table 4.1, we can see that the computational

complexity is symmetrical about displacements *mvx* and *mvy*. Although most of the complexity comes from those *mvxs* and *mvys* which are both not equal to zero, the values of elements within the 64×256 matrix are mostly very small, and imposing a threshold can leverage the complexity. For instance, when $mvx=1$ and $mvy=1$, when the threshold is chosen as 0.03, the number of multiplications is decreased dramatically from 16384 to 4229, and the number of additions is decreased from 16320 to 4165, respectively.

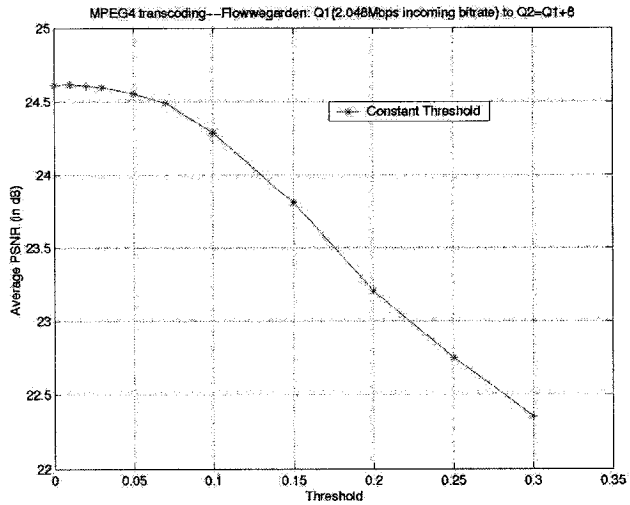
4.2.2.2. Suboptimal solutions to reducing computational complexity and simulation results

From the above discussion, as the threshold increases, the number of both multiplications and additions decreases. In order to investigate the impact of different thresholds on the performance of the transcoder, only one constant threshold has been applied to all 64 (64×256) matrices. Figure 4.15-Figure 4.20, Table 4.2, and Table 4.3 show some simulation results with sequences *Flowergarden* (352×240), *Forman* (176×144), *Bream* (176×144), *Football* (352×240), *Silent* (176×144), and *News* (176×144). All sequences are with frame rate at 30fp/s and IPPP...IPPP with GOP structure (N=15, M=1).

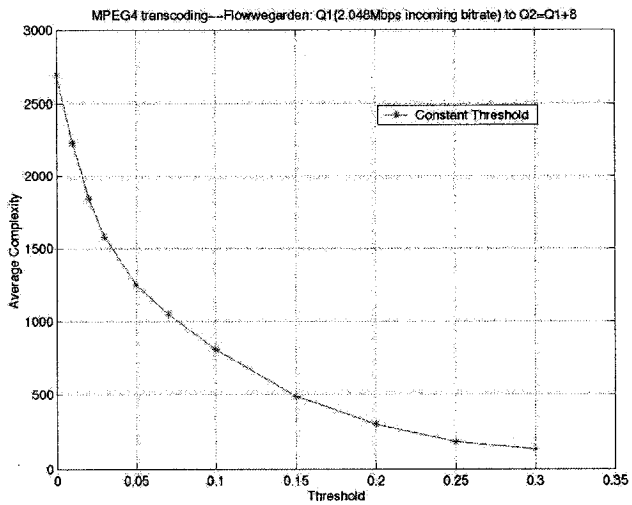
As we can see from Figure 4.15-Figure 4.20, Table 4.2, and Table 4.3, when the threshold increases, the operations needed for compositing one DCT block from the 4 blocks within the reference frame decreases; however, the performance of transcoded video degrades. As expected, the larger the threshold is, the worse the quality of the transcoded videos is, and the less the complexity is. From the six sets of simulations, we

find that the degraded quality of transcoded video is small and acceptable when the threshold is chosen to be 0.05. Figure 4.25-Figure 2.1 show the subjective qualities of *6* sequences with original, reconstructed with threshold=0.0 and threshold=0.05. We hardly distinguish the difference of subjective qualities when the threshold is chosen from 0 to 0.05.

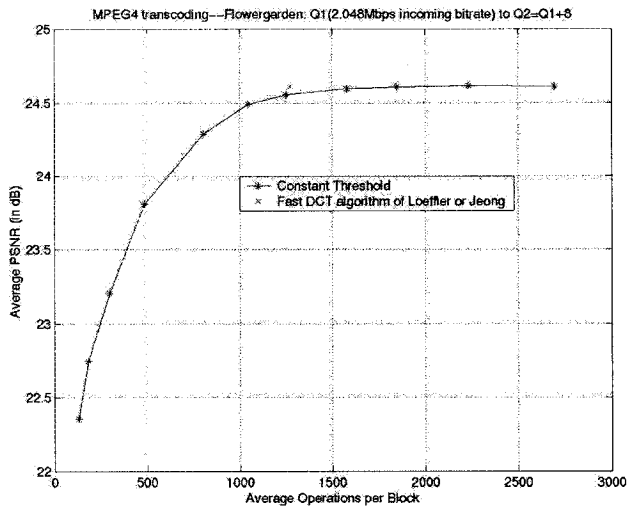
Figure 4.21-Figure 4.24 show the statistical properties of average PSNR, average complexity with different thresholds for different sequences, as we can see, when the threshold is chosen 0.05, the average PSNR of 6 transcoded videos degrades not so much as comparison with that of no thresholding, and the average complexity decreases nearly 50% or more. Therefore, the threshold with 0.05 could be a baseline for other sequences used for DCTTTM-based MC transcoding.


MPEG4 transcoding--Flowwegarden: Q1(2.048Mbps incoming bitrate) to Q2=Q1+8
Constant Threshold
Average PSNR (in dB)
Threshold
25
24.5
24
23.5
23
22.5
22
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35

(a)

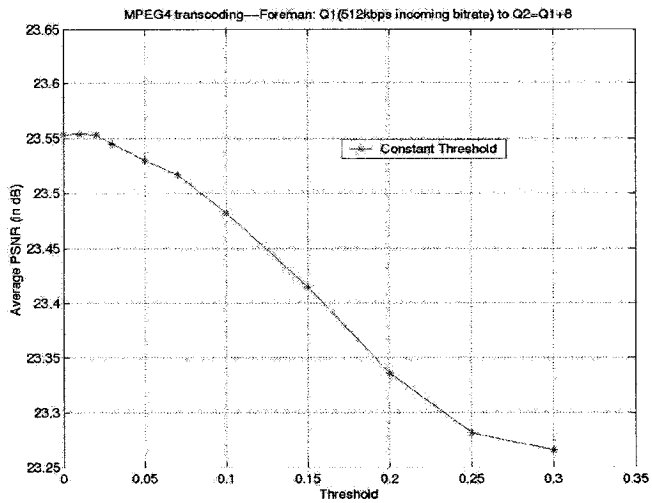

MPEG4 transcoding--Flowwegarden: Q1(2.048Mbps incoming bitrate) to Q2=Q1+8
Constant Threshold
Average Complexity
Threshold
3000
2500
2000
1500
1000
500
0
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35

(b)

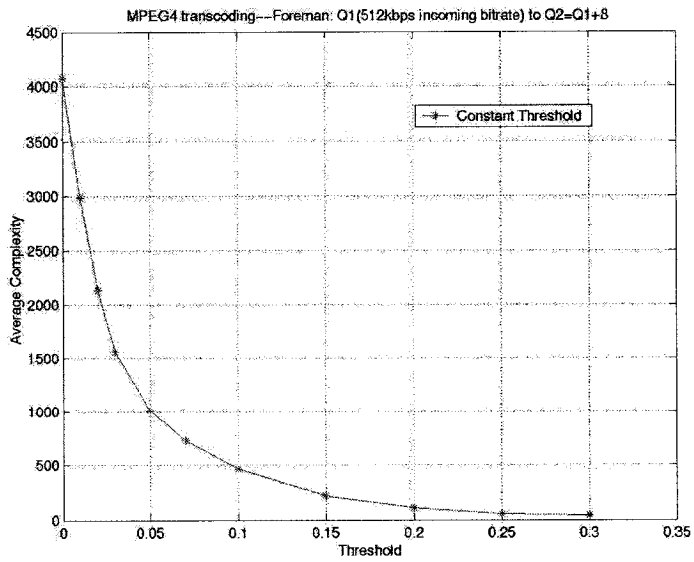


(c)

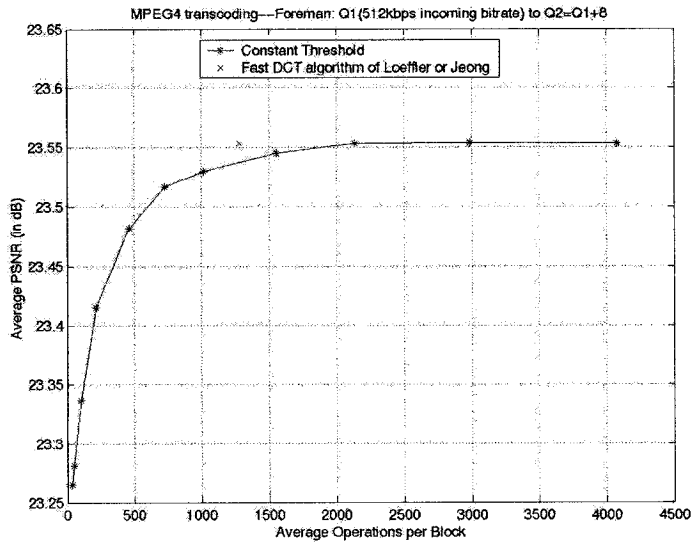
Figure 4.15 The PSNR and Complexity with different thresholds: *Flowergarden* --- transcoding from incoming bitsrate 2.048Mbps, GOP structure: N=15, M=1; frame rate 30 frame/s (Q1) to Q1+8. (a) PSNR ~ Threshold, (b) Complexity ~ Threshold, and (c) PSNR ~ Complexity.



(a)

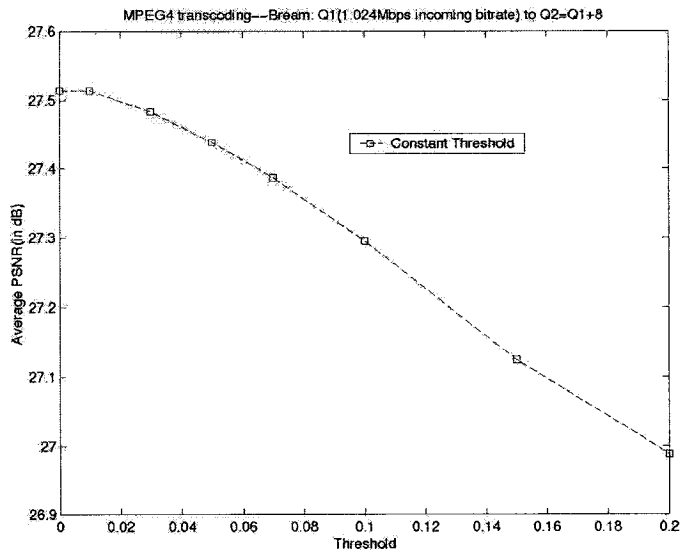


(b)

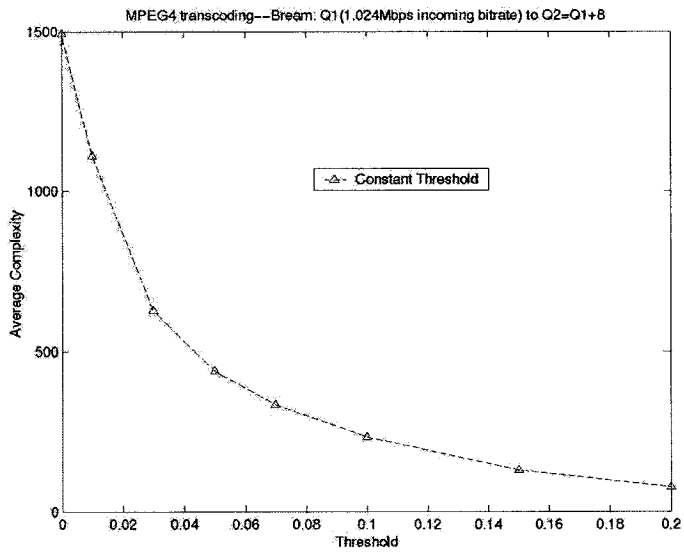


(c)

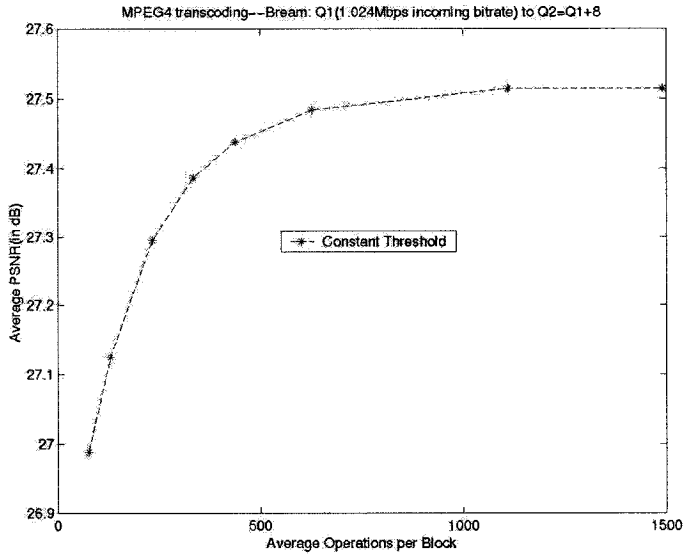
Figure 4.16 The PSNR and Complexity with different thresholds: *Foreman* --- transcoding from incoming bitsrate 512kbps, GOP structure: N=15, M=1; frame rate 30 frame/s (Q1) to Q1+8. (a) PSNR ~ Threshold, (b) Complexity~Threshold, and (c) PSNR ~ Complexity.


MPEG4 transcoding--Bream: Q1(1.024Mbps incoming bitrate) to Q2=Q1+8
Constant Threshold
Average PSNR(in dB)
Threshold
27.6
27.5
27.4
27.3
27.2
27.1
27
26.9
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2

(a)

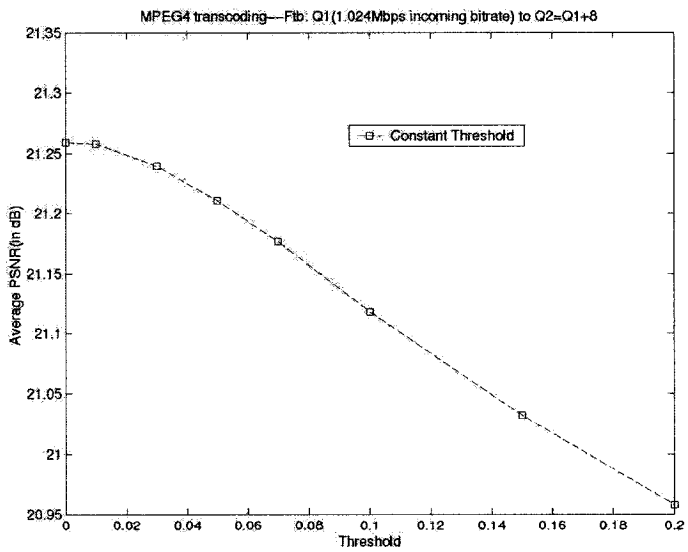

MPEG4 transcoding--Bream: Q1(1.024Mbps incoming bitrate) to Q2=Q1+8
Constant Threshold
Average Complexity
Threshold
1500
1000
500
0
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2

(b)

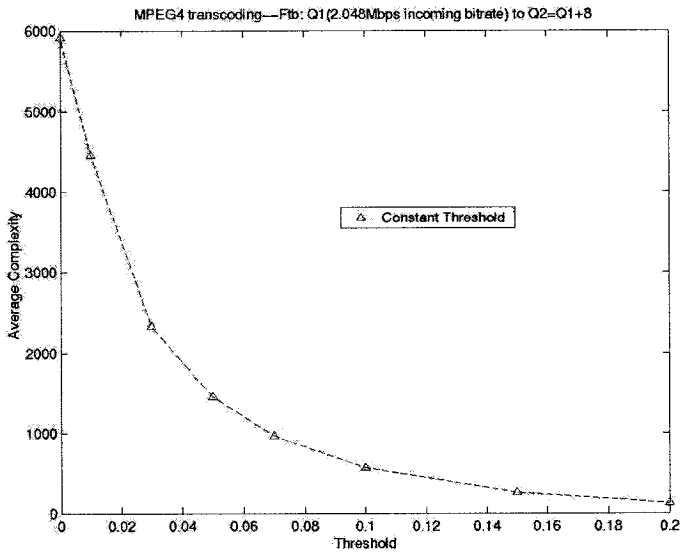


(c)

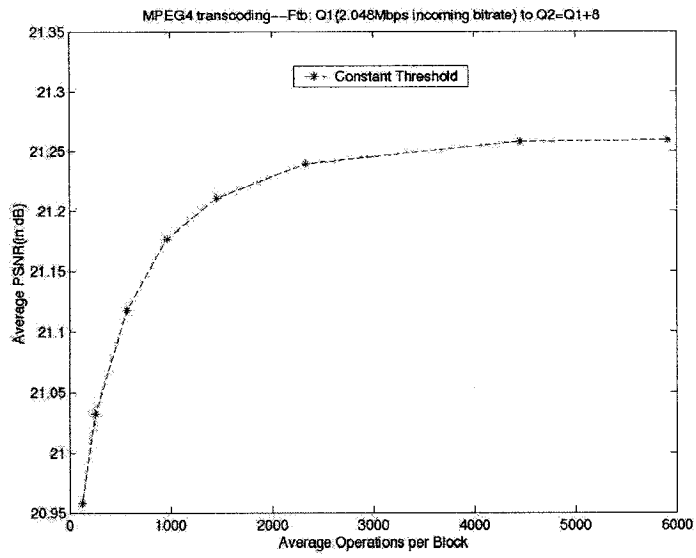
Figure 4.17 The PSNR and Complexity with different thresholds: *Bream* --- transcoding from incoming bitsrate 1024kbps, GOP structure: N=15, M=1; frame rate 30 frame/s (Q1) to Q1+8. (a) PSNR ~ Threshold, (b) Complexity~Threshold, and (c) PSNR ~ Complexity.



(a)

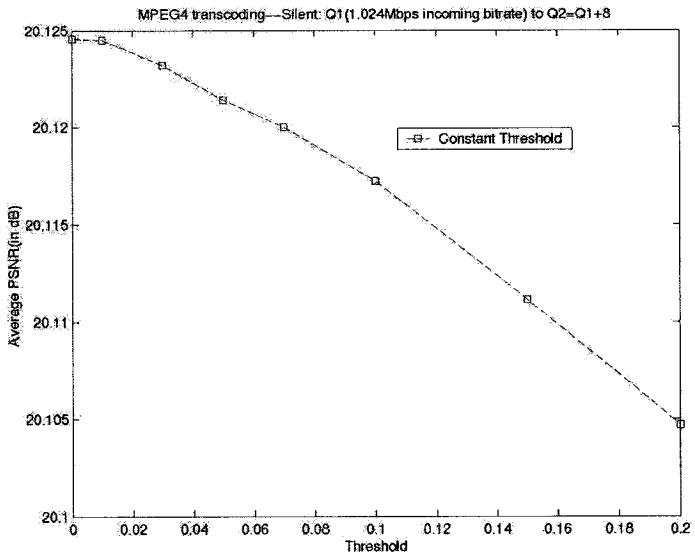


(b)

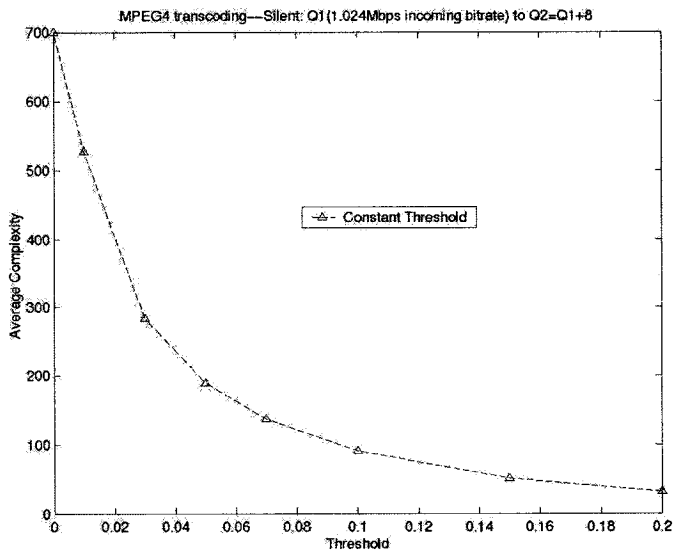


(c)

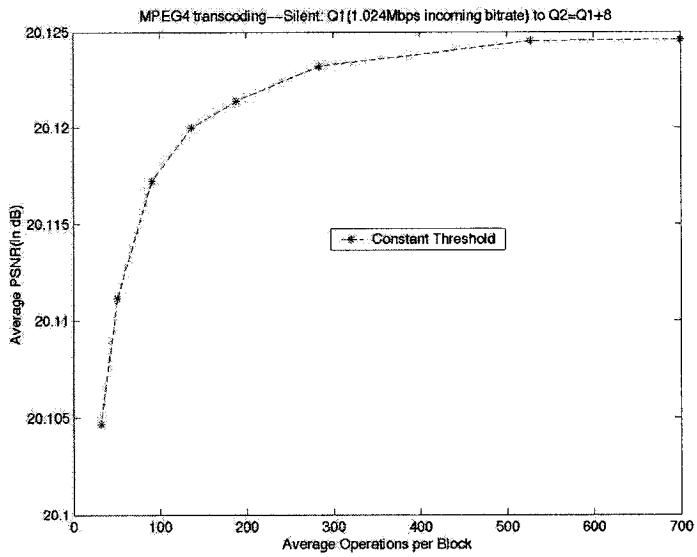
Figure 4.18 The PSNR and Complexity with different thresholds: *Football* --- transcoding from incoming bitsrate 2.048Mbps, GOP structure: N=15, M=1; frame rate 30 frame/s (Q1) to Q1+8. (a) PSNR ~ Threshold, (b) Complexity~Threshold, and (c) PSNR ~ Complexity.


MPEG4 transcoding---Silent: Q1(1.024Mbps incoming bitrate) to Q2=Q1+8
Constant Threshold
Average PSNR(in dB)
Threshold
20.125
20.12
20.115
20.11
20.105
20.1
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2

(a)

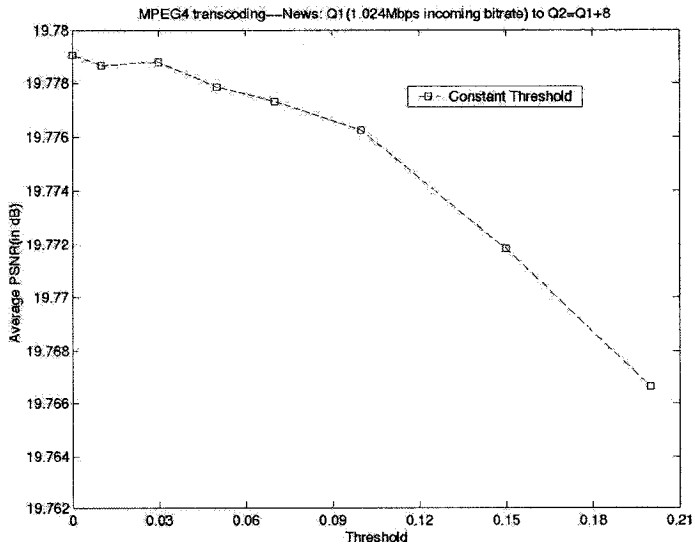

MPEG4 transcoding---Silent: Q1(1.024Mbps incoming bitrate) to Q2=Q1+8
Constant Threshold
Average Complexity
Threshold
700
600
500
400
300
200
100
0
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2

(b)

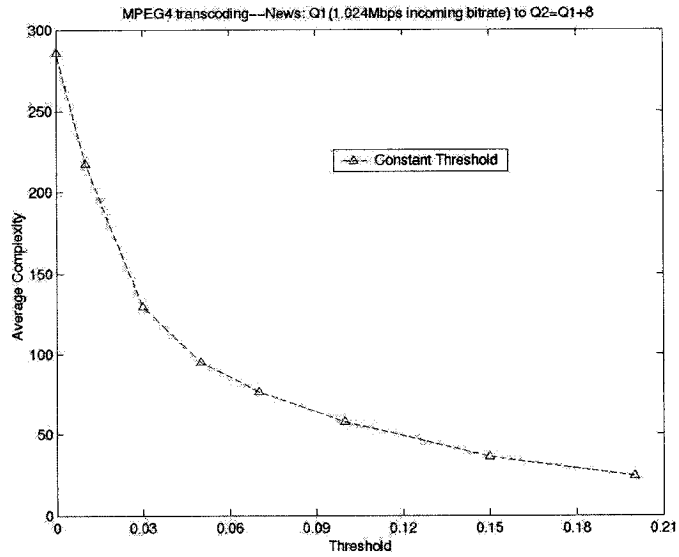


(c)

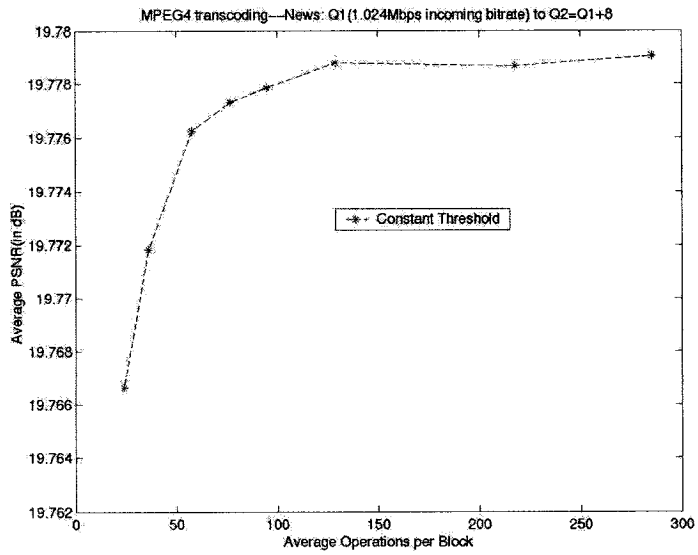
Figure 4.19 The PSNR and Complexity with different thresholds:*Silent* --- transcoding from incoming bitsrate 1024kbps, GOP structure: N=15, M=1; frame rate 30 frame/s (Q1) to Q1+8. (a) PSNR ~ Threshold, (b) Complexity~Threshold, and (c) PSNR ~ Complexity.



(a)



(b)



(c)

Figure 4.20 The PSNR and Complexity with different thresholds: *News* --- transcoding from incoming bitsrate 1024kbps, GOP structure: N=15, M=1; frame rate 30 frame/s (Q1) to Q1+8. (a) PSNR ~ Threshold, (b) Complexity~Threshold, and (c) PSNR ~ Complexity.

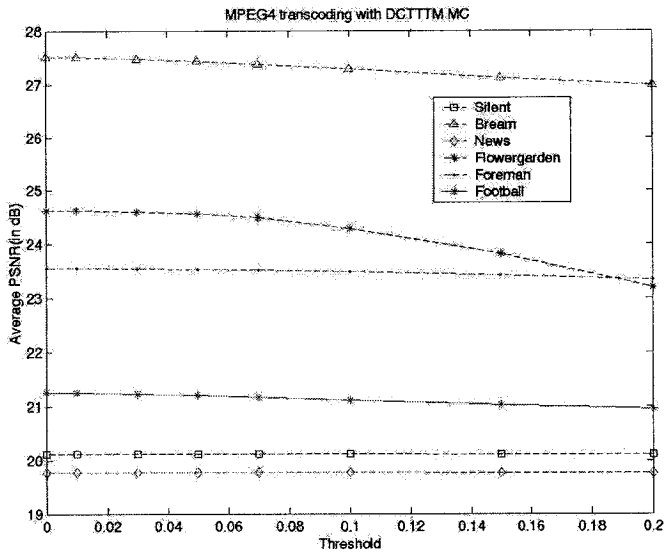


Figure 4.21 The PSNR versus different thresholds

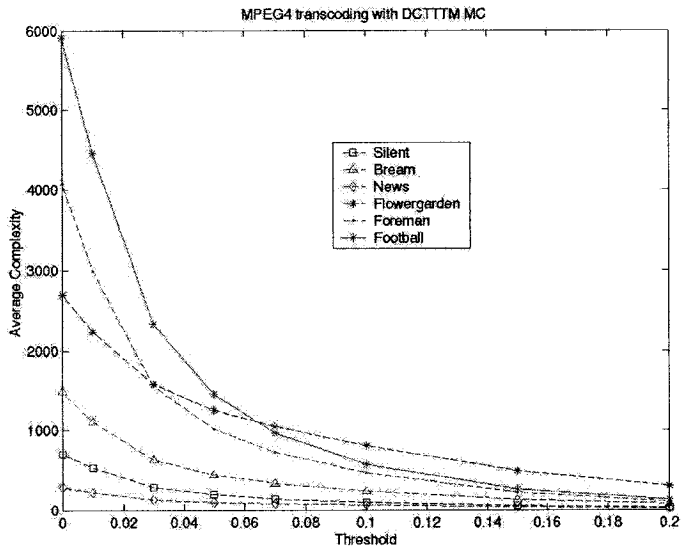


Figure 4.22 The complexity versus different thresholds

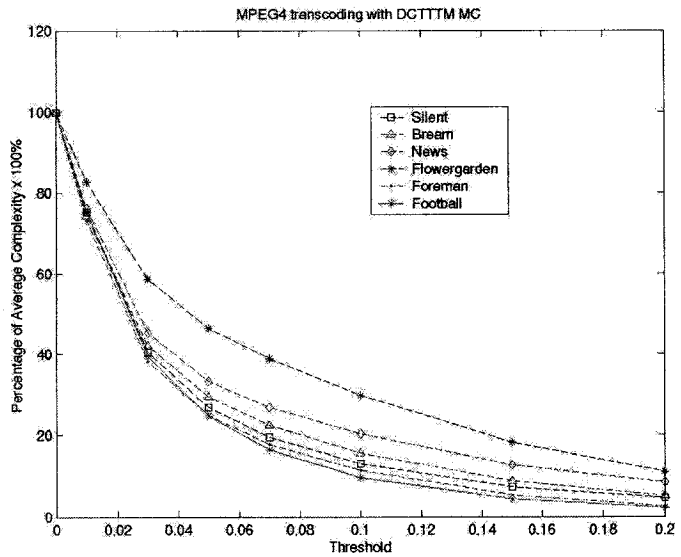


Figure 4.23 The percentage of complexity versus different thresholds

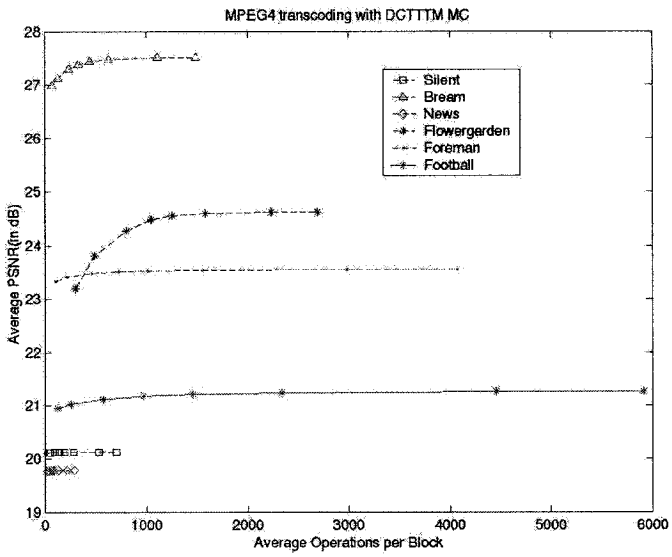


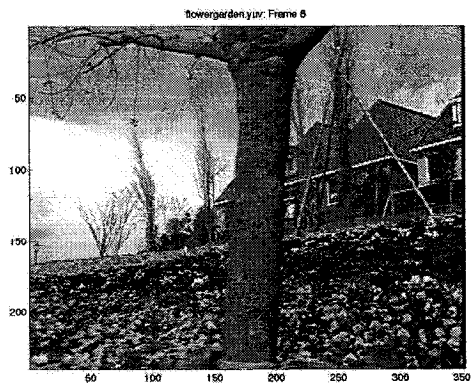
Figure 4.24 The PSNR versus complexity

Table 4.2 The average PSNR (Avg. PSNR) and average operations of multiplications and additions per block (Avg. Ops) with different thresholds. (Flowergarden, Forman, and Silent)

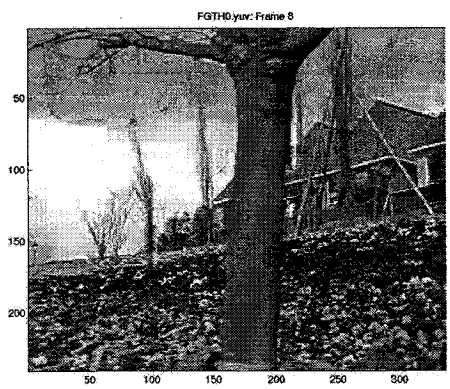
Threshold	Flowergarden		Foreman		Silent	
	Avg. PSNR	Avg. Ops	Avg. PSNR	Avg. Ops	Avg. PSNR	Avg. Ops
0.00	24.62	2695	23.55	4076	20.13	699
0.01	24.62	2233	23.55	2986	20.12	528
0.03	24.60	1581	23.54	1558	20.12	283
0.05	24.56	1254	23.53	1014	20.12	189
0.07	24.49	1048	23.52	726	20.12	137
0.10	24.28	806	23.48	464	20.11	91
0.15	23.81	490	23.42	217	20.11	51
0.20	23.20	299	23.34	106	20.10	32

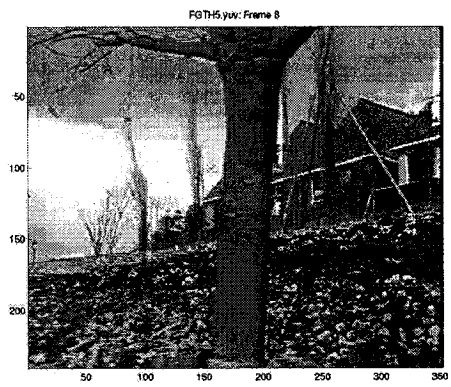
Table 4.3 The average PSNR (Avg. PSNR) and average operations of multiplications and additions per block (Avg. Ops) with different thresholds. (News, Football, and Bream)

Threshold	News		Football		Bream	
	Avg. PSNR	Avg. Ops	Avg. PSNR	Avg. Ops	Avg. PSNR	Avg. Ops
0.00	19.78	285	21.26	5914	27.51	1492
0.01	19.78	218	21.26	4460	27.51	1110
0.03	19.78	129	21.24	2337	27.48	629
0.05	19.78	95	21.21	1462	27.44	439
0.07	19.78	77	21.18	972	27.39	335
0.10	19.78	58	21.12	570	27.30	233
0.15	19.77	36	21.03	257	27.12	130
0.20	19.77	24	20.96	129	26.99	77



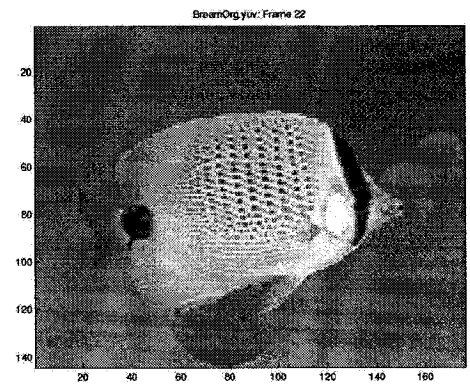
(a)



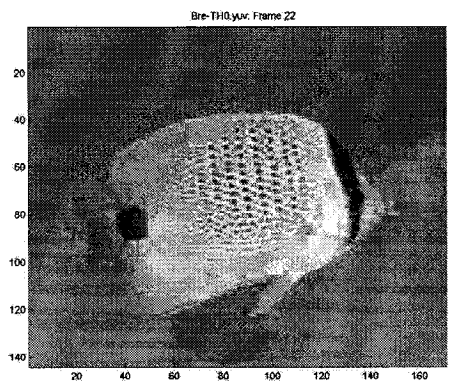


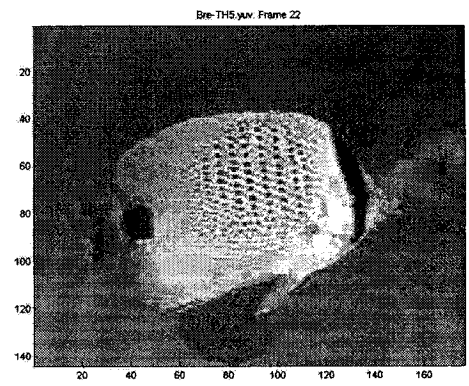
(b) (c)

Figure 4.25 Illustration of transcoding effects by using different threshold: *Flowergarden*---(a) Original image at frame 8.(b) Reconstructed image at frame 8 by transcoding with threshold=0.00.(c) Reconstructed image at frame 8 by transcoding with threshold=0.05.



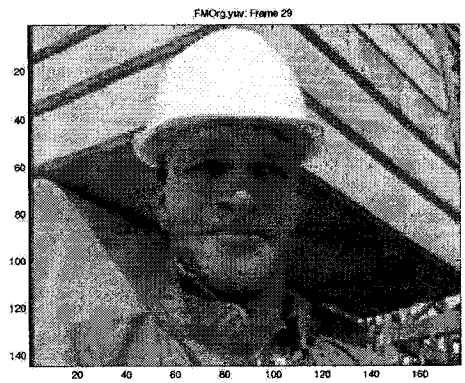
(a)



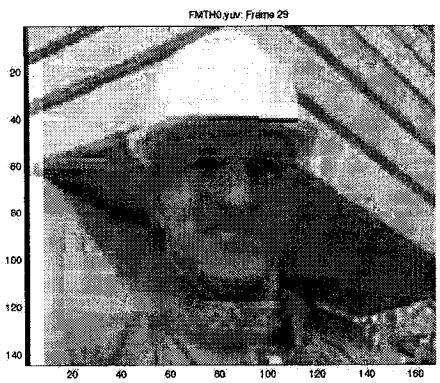


(b) (c)

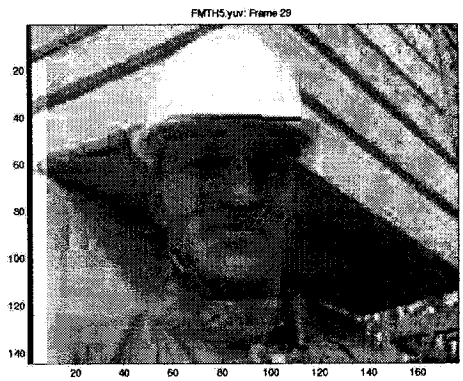
Figure 4.26. Illustration of transcoding effects by using different threshold: *Bream* ----- (a) Original image at frame 22. (b) Reconstructed image at frame 22 by transcoding with threshold=0.00. (c) Reconstructed image at frame 22 by transcoding with threshold=0.05.



(a)

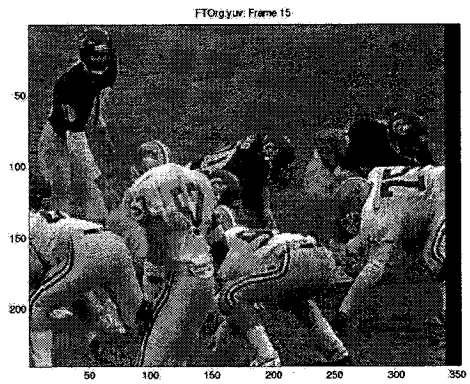


(b)

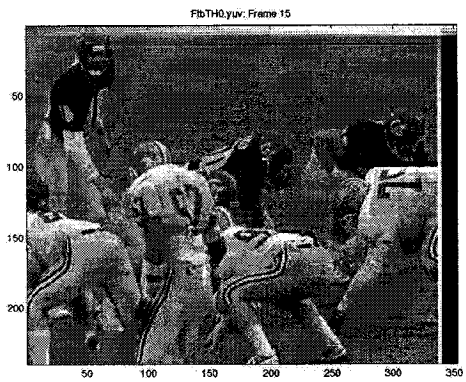


(c)

Figure 4.27. Illustration of transcoding effects by using different threshold: Foreman ---- (a) Original image at frame 29. (b) Reconstructed image at frame 29 by transcoding with threshold=0.00. (c) Reconstructed image at frame 29 by transcoding with threshold=0.05.

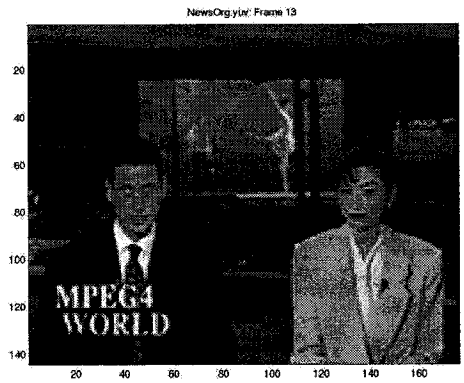


(a)

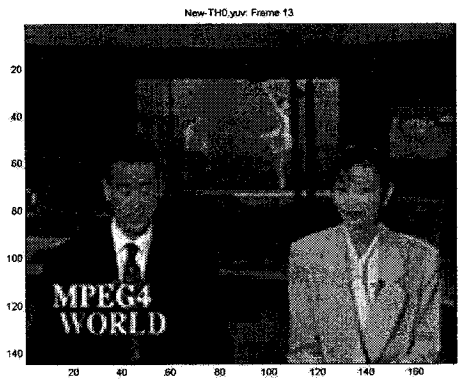


(b) (c)

Figure 4.28. Illustration of transcoding effects by using different threshold: *Football*--- (a) Original image at frame 15. (b) Reconstructed image at frame 15 by transcoding with threshold=0.00. (c) Reconstructed image at frame 15 by transcoding with threshold=0.05.

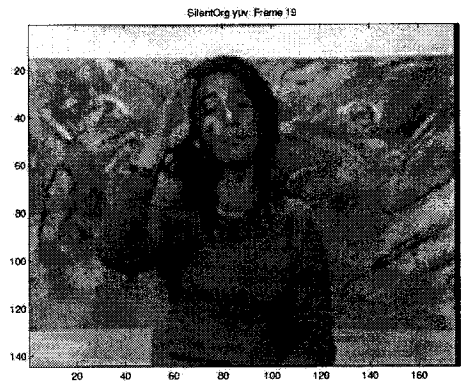


(a)

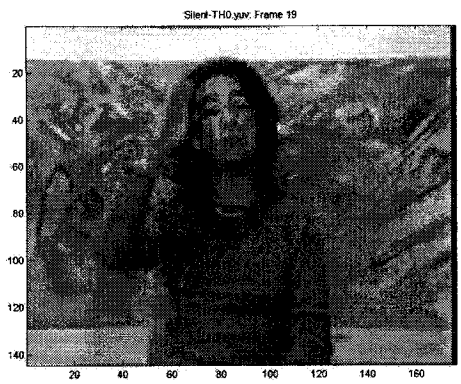


(b) (c)

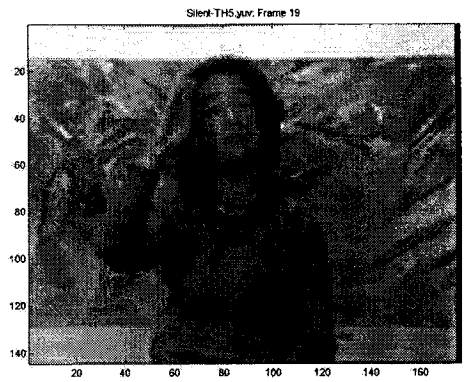
Figure 4.29 Illustration of transcoding effects by using different threshold: News----. (a) Original image at frame 13. (b) Reconstructed image at frame 13 by transcoding with threshold=0.00. (c) Reconstructed image at frame 13 by transcoding with threshold=0.05.



(a)



(b)



(c)

Figure 4.30. Illustration of transcoding effects by using different threshold: Silent---. (a) Original image at frame 19. (b) Reconstructed image at frame 19 by transcoding with threshold=0.00. (c) Reconstructed image at frame 19 by transcoding with threshold=0.05.

Although in most cases, the reference block are not aligned to an integer multiple of block size (8 in both directions), refer to Figure 4.3, the reference block might be aligned in one direction $h = 8$ or $w = 8$, which means that the right-hand of Equ(4.15) contains two terms only, or in both directions ($h = w = 8$), in which case, $\hat{X} = X_1$, therefore, no computations are needed. Having a set of equations from (4.11)- (4.15), and , and taking proper thresholding method to reducing the computational complexity, similar to that in the pixel domain, the transcoding of MPEG-4 compressed video in the DCT domain can also be implemented.

With regard to transcoders implemented in the pixel domain, there are 2 (I) DCT needed. Over years, several existing fast DCT/IDCT algorithms have been proposed[43]-[49]. Among these algorithms, the Loeffler's[48] and the Jeong's[49] algorithms have the least computational complexity. The Loeffler's requires 11 multiplications and 29 additions, and the Jeong's needs 12 multiplications and 28 additions for one 1D 8-point (I)DCT. Since 2D 8×8 (I)DCT can be implemented by 16 1D 8-point (I) DCTs, therefore, the total computational complexity for two 2D 8×8 (I)DCT is equal to 1280 multiplications and additions. As a result, the computational complexity of transcoders, where the Loeffler's or Jeong's algorithm is applied to (I)DCT, needs 1280 multiplications and additions. From Figure 4.15(c), Figure 4.16(c) and Table 4.2, when the threshold is chosen up to 0.05, the degradation of performance with DCTTTM MC algorithm in our simulations can be ignored with comparison to that of the Loeffler's and the Jeong's algorithms. Also from Figure 4.21-Figure 4.24, the threshold with 0.05 could likely be chosen as the baseline for other sequences used in the DCTTTM MC transcoding.

In the next subsection, some available suboptimal solutions to reducing the complexity are discussed.

4.2.2.3. Other Suboptimal Solutions to Reducing the Complexity of MPEG-4 Transcoder in the DCT Domain

The criteria of **DCTTTM** is that if the absolute value of each element within 64×256 matrix less than the determined threshold, then the element is set to 0, otherwise remains unchanged. Besides one constant threshold being applied to all 64 64×256 matrices as described in section 4.2.2.2, there are other ways to reduce the computational complexity used for transcoding of MPEG-4 compressed bitstreams in the DCT domain. One way is to use different thresholds on different columns of the 64×256 matrix. This may be justified by the fact that different columns correspond to different frequencies of the blocks in the reference frame. Generally, higher frequencies are more likely to be zero. Thus, the lower frequency of DCT coefficients, the lower the threshold is. For examples, the threshold **Th1** chosen for position (1,0) and (0,1) within the DCT block X1 (they create two column of 64×256 matrix) is the same, and the threshold **Th2** for position (1, 1), (2, 0), and (0, 2) within the DCT block X1 (they also create three column of 64×256 matrix) is the same. However, **Th2** is greater than **Th1**, and so on. Similarly, the same row of 64×256 matrix can be truncated by the same threshold, and the different thresholds are chosen for the different rows.

Another two algorithms used to reduce the computational complexity are that only partial DCT coefficients of four adjacent DCT blocks participated to compose the new DCT block, therefore, only partial elements of 64×256 matrix are involved in compositing new DCT block from the four reference blocks. The typical ways are triangle

and square area selections. Obviously, the larger the selected area is, the more the computational complexity, and the less the degradation of performance.

Figure 4.31 and Figure 4.32 show the influence of complexity on the performance with different suboptimal solutions. Here, two sequences are used for simulation: *Flowergarden* and *Foreman*. For *Flowergarden,* the bitrate of incoming bitstream is 2.048Mbps, the corresponding quantization parameter is Q_1 , GOP structure is IPPP...IPPP... with N=15 and M=1, the frame rate is 30 frames/s. For *Foreman,* the bitrate of the incoming bitstream is 512kbps, the corresponding quantization parameter is Q_1, GOP structure is IPPP...IPPP... with N=15 and M=1, the frame rate is 30 frames/s. All transcoded outgoing bitstreams are generated by requantization parameter $Q_2 = Q_1 + 8$. That is no rate control is used.

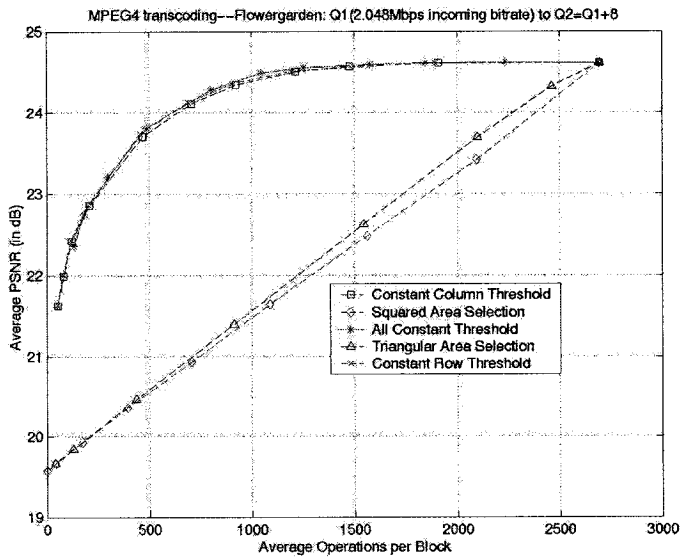


Figure 4.31 Average PSNR versus Computational Complexity with different suboptimal solutions: *Flowergarden* sequence

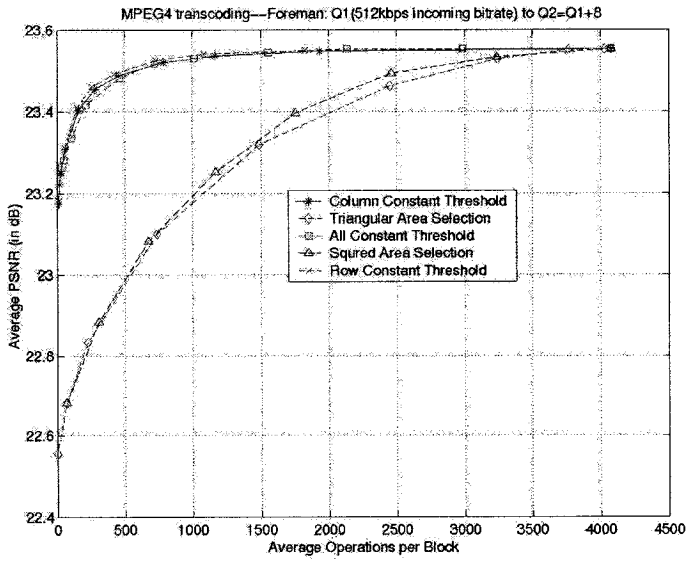


Figure 4.32 Average PSNR versus Computational Complexity with different suboptimal solutions: *Foreman* sequence

In these experiments All Constant Threshold Method (only one threshold for all 64×256 matrices) has almost the same performance along with computational complexity as that of one constant threshold applied to same column or row of 64×256 matrices (Column Constant Threshold Method or Row Constant Threshold Method). They all have much better performance than that of Triangle or Square Area Selection Method. When the target computational complexity used for compositing one DCT block from four adjacent DCT blocks in the reference frame is less than 1000, the degradation of performance reaches nearly 3dB for *Flowergarden* (more active sequence), and 0.3dB for *Foreman* (slow motion sequence).

In the next section, by using Chang and Messerschmitt's MC-DCT compositing method, frame-skipping transcoding of MPEG-4 compressed video in the DCT domain is discussed.

4.3. Frame-skipping Transcoding of MPEG-4 Compressed Video in the DCT Domain

In transcoding, there are two methods to achieve bit rate conversion: one is merely increasing requantization parameters in the second encoder in order to meet the target bitrate with the same frame rate as the incoming bitstream; the other is by skipping some frames to achieve the bitrate conversion. The first method will introduce more degradation when a higher transcoding ratio is required. In order to overcome this problem, frame skipping is usually used. However, in frame skipping, the motion vectors of the incoming bitstream cannot merely be reused for the non-skipped frame following the dropped frame because the reconstructed frame of the dropped frame does not exist. Several motion vector refinement schemes have been proposed to recompute the new motion vectors[25][26][27].

In [25], a Horizontal And Vertical Search (HAVS) scheme has been proposed. This method can reduce the computational complexity according to the base motion vector composed from the dropped frames, and achieve quite good quality of video especially for slow motion sequences. In [26], another motion estimation algorithm is presented where the dominant macroblock is determined by using the activity to compose the base motion vector, and a Variable Step Size (VSS) scheme is used to reduce the computational complexity while improving the quality of transcoded video for fast-moving objects. In [27], a new architecture for a dynamic frame-skipping transcoder in the DCT domain has been proposed where a strategy for determining the length of the

skipped frame such that it can reduce the quality degradation as well as the motion jerkiness perceived by human beings.

In the low bitrate applications, IPPP... IPPP... with larger number of GOP structure is usually employed in MPEG-4 or H.263 encoder. The quality of each frame affects the quality of the following frames. This build up of propagating error can become intolerable. In order to overcome this problem, in this thesis, a new adaptive MV refinement algorithm in the DCT domain for the frame-skipping transcoder has been proposed to improve the quality of reconstructed video while the computational complexity is not greatly increased. Figure 4.33 shows the block diagram of the cascaded object-based MPEG4 transcoder used in this thesis. Shape information is directly passed to the output of transcoder; only texture information is affected by the transcoding. The encoder compresses an input video at a bit rate $R1$, and then the transcoder converts this compressed video at a bitrate by reducing the frame rate.

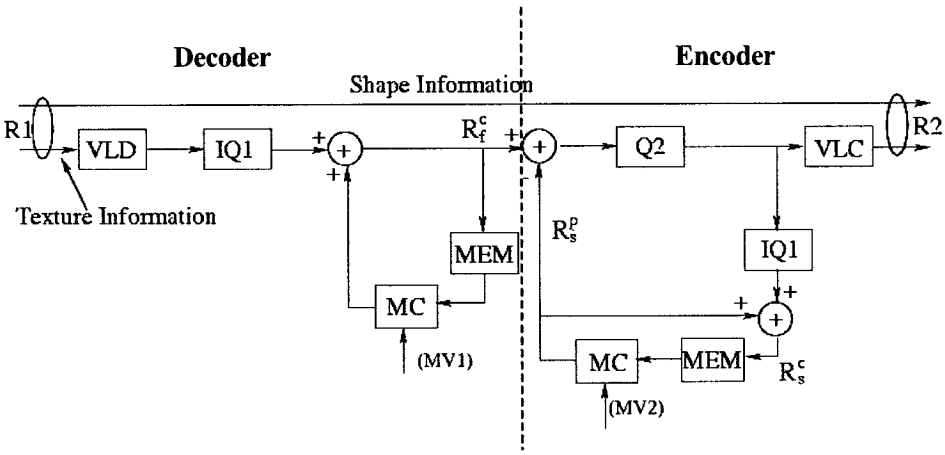


Figure 4.33 The structure of frame-skipping transcoder in the DCT domain

Where MV1 is the incoming motion vectors. MV2 is the new motion vectors for current

frame after the previous reference frames are skipped. Generally speaking, MV2 is not equal to MV1, it should be reestimated.

In order to do motion re-estimation, we have to reconstruct the reference frame and the predicted frame. As shown in Figure 4.33, the reference frame is reconstructed from the new motion vectors $MV2$, the output frame of the decoder based on the incoming motion vectors $MV1$ is regarded as the input frame of the second encoder as shown in Figure 4.33. The motion estimation is based on the luminant macroblocks or 8×8 blocks (in the MPEG-4 encoder, motion vectors based on the blocks are allowed). To reconstruct the block in the DCT domain, manipulation and composition of one DCT block is done in the same way as that introduced in [20]. The computational complexity involved in motion estimation is decided by two factors: one is the search area size; the other is the number of pixels in each block, which is used to evaluate the minimal Mean Absolute Difference (MAD) or Mean Squared Error (MSE). Generally speaking, the more the search area size and the more the pixels, the greater the computational complexity. In order to reduce complexity, one usually does motion estimation based on the composited dominant motion vectors or by reducing the number of pixels.

As discussed above, motion estimation leads to large computational complexity in the DCT domain, especially for a large search area size N. To reduce the complexity, several MV refinement schemes based on the composed MVs have been proposed which are implemented in the pixel domain[25][26], and are basically processed in the pixel domain. In this thesis, a new effective scheme based on Weighted Forward Dominant Vector Selection (WFDVS) combined with suboptimal motion estimation algorithm in the DCT domain is proposed.

4.3.1. WFDVS (Weighted Forward Dominant Vector Selection) Scheme

Many researchers have developed several efficient dominant composited motion vector schemes such as FDVS, ADVS, Bilinear Interpolation, and Bidirectional Motion Estimation Based on P Frame Motion Vectors and Area Overlap[41] Algorithms. Here, we employ the base motion vector composition scheme called Weighted Forward Dominant Vector Selection (WFDVS), which is further developed based on FDVS, Area Overlap[41] and bilinear interpolation for every block.

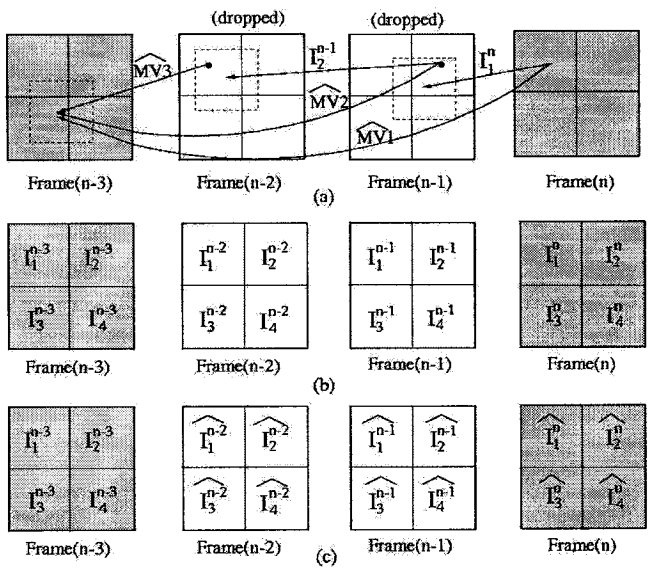


Figure 4.34 (a) WFDVS (Weighted Forward Dominant Vector Selection (b) Incoming motion vector (c) Composed base motion vector

As shown in Figure 4.34, assume Frame (n-2) and Frame (n-1) are dropped, then the base motion vectors called $\hat{I}_1^{n-2}, \hat{I}_2^{n-1}$, and $\hat{I}_1^{n}$ for Frame (n-2), Frame (n-1) and

Frame (n), respectively, can be composited as:

$$\begin{cases} \hat{I}_1^n = I_1^n + \dfrac{\sum_{i=1}^{4} \hat{I}_i^{n-1} \times S_i^{n-1}}{64} \\ \hat{I}_2^{n-1} = I_2^{n-1} + \dfrac{\sum_{i=1}^{4} \hat{I}_i^{n-2} \times S_i^{n-2}}{64} \\ \hat{I}_1^{n-2} = I_1^{n-2} \end{cases} \tag{4.21}$$

Where, S_i^{n-2} and S_i^{n-1} (i=1...4) are weighted coefficients, these are the area overlapped by the current block within the reference frame. I_i^{n-2}, I_i^{n-1}, and I_i^n are the incoming motion vectors for the Frame (n-2), Frame (n-1) and Frame (n), respectively. Since the MPEG-4 standard enables motion estimation based on either (16×16) macroblock (*INTER16*) or (8×8) block (*INTER8*). For *INTER16* coding mode, four motion vectors for 4 blocks within a macroblock are regarded as the same; while for *INTER8* coding mode, they can be different. Here, we don't change the coding modes which are determined in the pre-encoder used to generate the incoming bitstream, therefore, when the skipped frame is coded in *INTER8*, and the current incoming frame is coded in *INTER16*, then the four composed base motion vectors for each block within a macroblock may be different. In such a situation, we should generate a new motion vector for one macroblock from 4 different motion vectors of 4 blocks within this macroblock. The simplest method used in simulation is to average these four MV's. In addition, if the composed motion vector is beyond the search range, it can be simply replaced by the incoming motion vector.

4.3.2. The Suboptimal MV Refinement algorithm in the DCT Domain

Usually, we do motion estimation based on MAD (Mean Absolute Difference) or MSE (mean squared error), here, a block-based MSE is considered for the matching criteria. To obtain the motion vectors for the current block, we find a pair of motion vectors (mv_x, mv_y) for each block with the minimal MSE within the search area S:

$$(mv_x, mv_y) = \arg_{(m,n)\in S} \left(MSE_f(m,n)\right) \tag{4.22}$$

Where:

$$MSE_f(m,n) = \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M}\left|R_f^c(m,n) - R_s^c\left(i+mv_{bx}+m, j+mv_{by}+n\right)\right|^2 \tag{4.23}$$

Where $R_f^c(i,j)$ and $R_s^c(i,j)$ are output frame of the predecoder and reconstructed reference frame in the second encoder, and mv_{bx} and mv_{by} are the composed dominant motion vectors, and the block area size is $M \times M$. Assume the frame size is $W \times H$, the search range is $(m,n) \in [-N, +N]$, if we employ full search method, then from Equ.(4.22) and (4.23), there nearly requires $W \times H \times (2N+1)^2$ multiplications and $2W \times H \times (2N+1)^2$, or $3W \times H \times (2N+1)^2$ mults/adds operations. It is very expensive.

There are many approaches to reduce complexity by reducing the number of motion vectors searched. For example Three Step Search (TSS)[39], Diamond Search[40]. However, in the DCT domain, it is possible to further reduce the complexity by partially calculating the *MSE* of DCT coefficients. Because usually the whole energy of one DCT block concentrates on the DC and low frequency AC DCT coefficients, we propose a suboptimal motion re-estimation method in which we replace the 8×8 DCT block with a smaller DCT subblock to evaluate the minimal *MSE*. As in [50], a triangular shaped

DCT subblock can be used to evaluate the minimal *MSE*, however, there are still some DCT coefficients which can be intelligently ignored, and not considered to participate in motion estimation so that the computational complexity involved motion reestimation could be reduced. Hence, in this thesis, a new adaptive suboptimal motion estimation algorithm is proposed, where the number of DCT coefficients is employed to estimate the motion vector depending on their distributions.

Usually, the distribution of DCT AC coefficients of Intra macroblocks and non-Intrablocks can be represented by a two-sided Laplacian distribution[5][42], however, the distribution of reconstructed macroblocks Y=Y1+Y2 (Y1 is the reference value, Y2 is the residual, Y2 can be represented by a two-sided Laplacian distribution) cannot be simply represented by a two-sided Laplacian distribution because Y1 and Y2 are dependent. In order to simplify our method, a threshold *TR* is chosen such that if the probability of the AC DCT coefficient $X(i,j)(i,j \in [0,7])$ with its absolute value less than *TR* is more than 80%, then a symbol $Token[i][j](i,j \in [0,7])$ is set to 1; otherwise set to 0. When we later do motion reestimation, if $Token[i][j]$ is equal to 1, the corresponding AC DCT coefficients are not used to evaluate the *MSE*. Thus, a smaller subblock area $S'(\leq 64)$ is also determined. The value of $Token[i][j]$ could be updated every 10 frames to save the extra computational budget. The algorithm can be summarized as follows:

Step 1: Initialize all Num[i][j] to be 0, and set Token[i][j]=0, where $0 \leq i,j \leq 7$.

Step 2: Statistically calculate the number Num[i][j] of each DCT coefficient, which is less than given threshold TR per frame.

Step 3: Calculate the percentage of each DCT coefficient less than given threshold per frame, and determine whether it is participated in motion re-estimation according to the percentage. If this DCT coefficient is not chosen to do motion re-estimation, $Token[i][j]$ is set to 1, otherwise $Token[i][j]$ is equal to 0.

Step 4: According to $Token[i][j]$, the suboptimal motion vector for each macroblock or block is obtained.

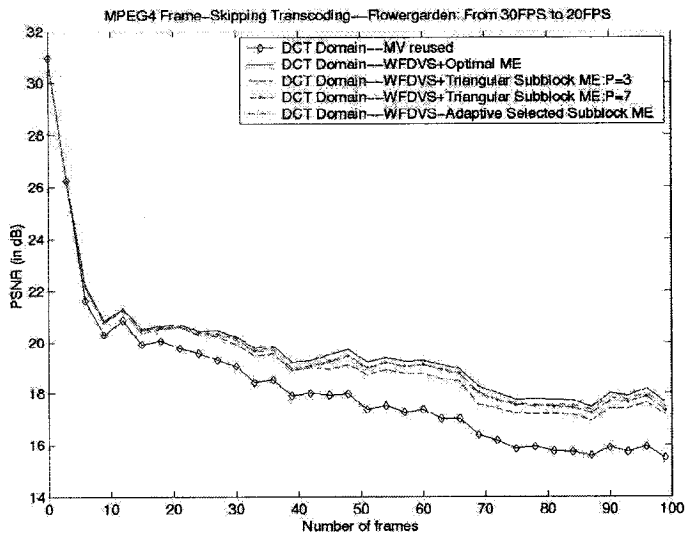
In our scheme, the threshold *TR* is determined by the quantization parameter QP. Since MPEG-2 and H.263 quantization methods are adopted in the MPEG-4 encoder; therefore, if MPEG-2 quantization type is used, TR=2QP; otherwise TR=2.5QP.

4.3.3. Simulation Results

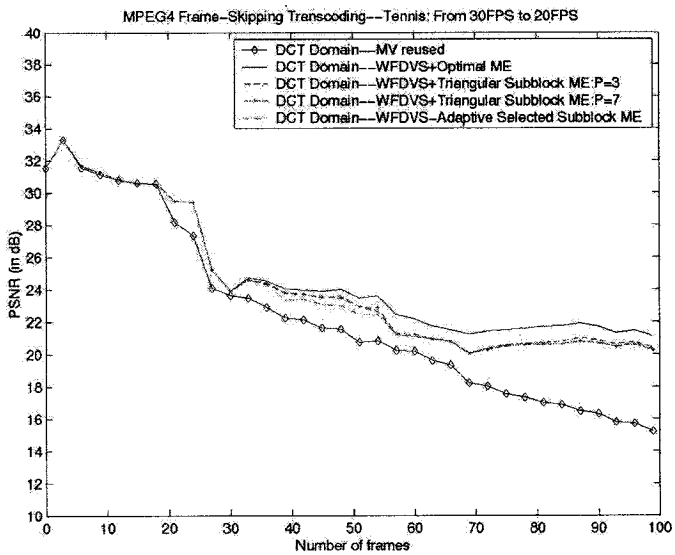
The simulation Results here demonstrate the transcoding performance of various MPEG4 compressed videos. We use 150 frames of the sequences *Flowergarden* (frame size: 352×240) and *Tennis* (frame size: 352×240). The sequences are all with IPPP...IPPP... GOP structure where N=150, M=1.

For *Flowergarden* and *Tennis*, the incoming bitrate is R1=960kbps. The incoming frame rate of all sequences is 30 frames/sec. The target frame rate is 20 frames/sec. In order to compare the performance of the proposed transcoder, all sequences are tested in five ways: the frame-skipping transcoding with the reused MV's, WFDVS+Optimal ME, FDVS+Triangular+Suboptimal ME (P=3), FDVS+Triangular+Suboptimal ME (P=7), and WFDVS+Adaptive+Suboptimal ME, their small full search area sizes are S∈ [-2,2], and all implemented in the DCT domain. P is used to determine the triangular area size as

shown in Figure 4.36.



(a)



(b)

Figure 4.35 PSNR of frame-skipping transcoding from 30 frames/s to 20 frames/s, incoming bitrate=960kbps, GOP: N=150, M=1 (a) Flowergarden: 352×240 (b) Tennis: 352×240

Table 4.4 Average PSNR and Percentage of used DCT Coefficients for motion estimation with *Tennis* and *Flowergarden* Sequences

Frame-Skipping Transcoding PSNR (in dB)					
Case	MV_Reused	P=3	P=7	Adaptive	FSS_ME
Tennis	22.30	23.82	24.19	24.12	24.74
Flowergarden	18.21	19.30	19.52	19.52	19.69

Percentage of Complexity Compared to FSS_ME		
Case	Tennis	Flowergarden
Triangular Subblock: P=7	56.25%	56.25%
Adaptive Selected Subblock	42.66%	53.98%

Results show that *PSNRs* of the MV refinement algorithms greatly outperform those of transcoding with reused MV's up to 3-4dB with average PSNRs up to 1.5dB as shown in Figure 4.35 and Table 4.4. From Figure 4.35(a) and Figure 4.35(b), we can see, when *P=3*, for slow motion sequence, the difference performance between the conventional motion estimation and our suboptimal motion estimation is less than nearly 0.2dB, for the fast motion sequence, the difference is less than 0.4dB. If we employ the adaptive method, the selected area used to evaluate *MSE* changes with different frames and different sequences by setting a proper threshold *TR*, however, for a triangular selected area, once P is determined, the complexity is also determined. For examples, when P=7, as shown in Table 4.4, for sequences *Tennis* and *Flowergarden*, the complexity of triangular selected area method is the same, and is equal to 56.25% that of conventional MV refinement scheme FSS, but for our adaptive method, the complexity of *Tennis* and *Flowergarden* is about 42.66% and 53.98%, respectively. As we can see from

the simulation results, the complexity of our adaptive suboptimal ME algorithm is reduced, however, the performance remains almost the same. Figure 4.37 and Figure 4.38 show the subjective qualities of frame-skipping transcoding with different MV refinement schemes. As we can see, if the MVs are reused, the quality of frame-skipping transcoded video would be worse than that with MV refinement schemes.

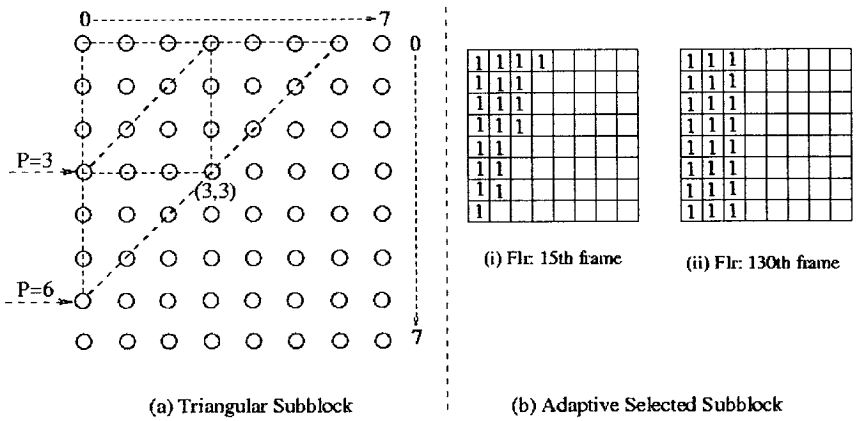
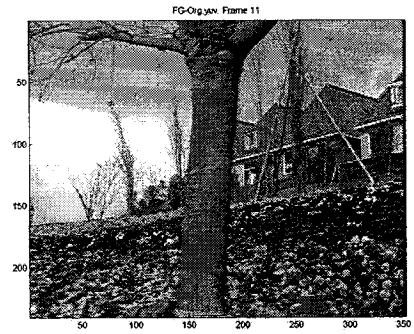
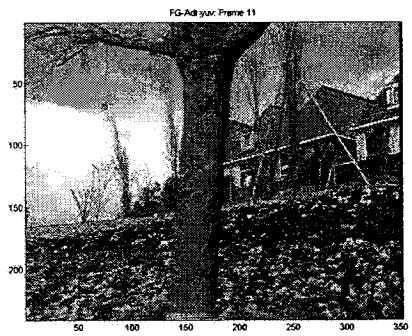


Figure 4.36 Subblock of 8×8 DCT Coefficients used for motion reestimation (a) Triangular Subblock[50], and (b) Adaptive Selected Subblock

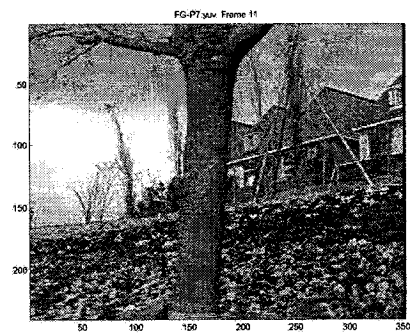


(a)

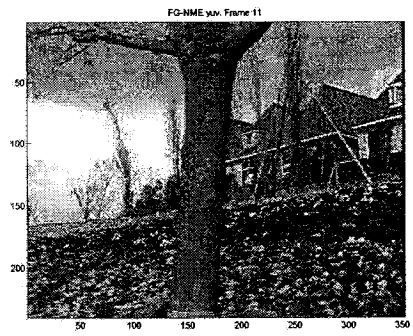
(b)



(c)

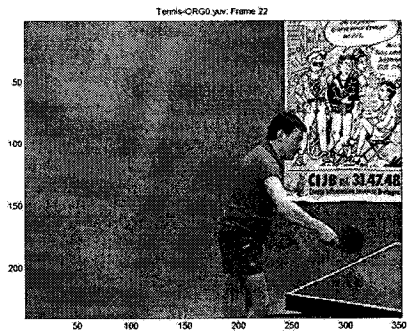


(d)

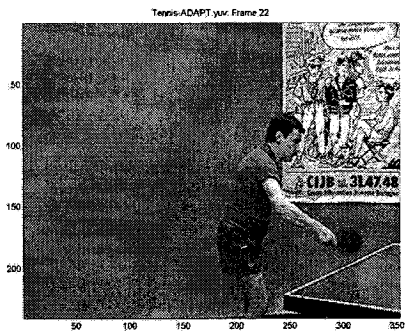


(e)

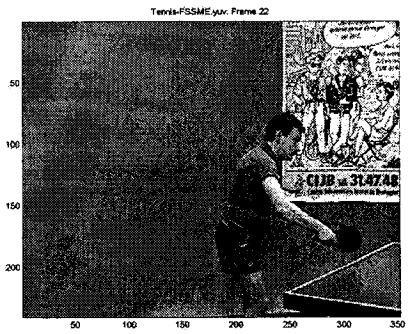
Figure 4.37. Illustration of frame-skipping transcoding by using different MV refinement schemes: *Flowergarden* at Frame 11. (a) Original Frame. (b) WFDVS +FSS_ME. (c)WFDVS_Adaptive Selected Subblock ME. (d) WFDVS+Triangular Subblock ME: P=7. (e) MV reused



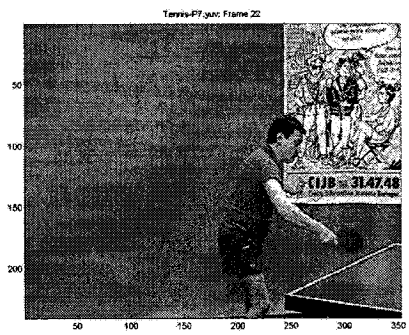
(a)



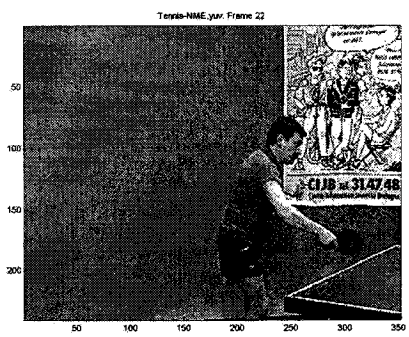
(b)



(c)



(d)



(e)

Figure 4.38. Illustration of frame-skipping transcoding by using different MV refinement schemes: *Tennis* at Frame 22 (a) Original Frame. (b) WFDVS +FSS_ME. (c)WFDVS_Adaptive Selected Subblock ME. (d) WFDVS+Triangular Subblock ME: P=7. (e) MV reused

4.4. Conclusions

In this chapter, we first propose a new DCT motion compensation algorithm other than that of Chang and Messerschmitt[20]. Theoretically, the complexity of our method is greater than that of Chang and Messerschmitt if no manipulation is employed. However, the complexity can be reduced greatly by applying the threshold of 0.05 as shown in Figure 4.10-Figure 4.20. Based on this analysis, we proposed three suboptimal DCT motion compensation algorithms: all one constant threshold, one row constant threshold, and one column constant threshold. In order to compare the performance of our proposed algorithms, the previous two algorithms are also used to do simulation: one is triangular area selection and squared area selection. Our simulation results show that our proposed algorithms have almost the same performance, however, greatly outperform those of previous methods under our assumed conditions. It should be noted that all simulations are conducted in floating-point. They can be done in integer operation by rounding floating-point coefficients in 16 bits integer, and performance would not be degraded too much. This work will be left for future.

In this chapter, we also address the computational complexity related to the motion re-estimation used for the frame-skipping transcoder in the DCT domain. Generally speaking, the complexity of motion compensation in the DCT domain is less than that in the pixel domain. However, in terms of motion estimation, when the search area size $N>2$, motion estimation in the DCT domain has more complexity than that in the pixel domain. Since the energy of 8×8 DCT block is concentrated on the DC and the low frequency index coefficients, we propose the suboptimal motion estimation algorithm by using an

adaptive DCT subblock based on the probability of each DCT AC coefficients instead of triangular selected DCT subblock to evaluate the minimal *MSE*. Experimental results show that our algorithm greatly improves the performance (in PSNR) over the reused MVs scheme, and get almost the same performance as the optimal motion estimation algorithm, while it requires less than 50% computational complexity of conventional ME algorithms. Moreover, this suboptimal ME algorithm can be combined with other ME algorithms such as TSS, Diamond Search, HSS in the DCT domain so that the computational complexity can be further reduced.

5. Contributions and Future Work

5.1. Contributions and Conclusions

Motivated by the importance of transcoding in current video technologies, the focus of this thesis is to investigate the architectures of MPEG-4 encoder and decoder, rate control models, motion estimation, motion compensation, and quantization methods. In the meanwhile, making use of MPEG-4 exclusive properties, several transcoding techniques of MPEG-4 compressed video have been discussed and implemented.

First of all, Chapter 3 addressed transcoding of MPEG-4 compressed video in the pixel domain. In this chapter, the architectures of both with drift error correction and without drift error correction transcoders for MPEG-4 video are given and implemented. Results show that the performance of TWDEC is much better than that without drift error correction, especially in cases of larger transcoding ratio applications. Similar to[8], joint transcoding of multiple MPEG-4 compressed videos is also discussed. In contrast to independent transcoding where each rate control works independently, a joint bit-allocation was used to distribute the bits between the sequences depending on their relative scenes complexities. It was shown that joint transcoding results in a more constant quality between the video sequences as well as within each sequence. Joint transcoding aims at a better utilization of the available channel bandwidth. In this Chapter, besides TM-5 like and RMSE-based joint bit-allocation algorithms, an approximated Distortion-based algorithm is presented to jointly reallocate bit rate for different sequences. A comparison between the output picture qualities of joint transcoding versus

independent transcoding was shown in Section 3.2.4. Although there is no PSNR improvement achieved in Approximated Distortion-based joint transcoding versus independent transcoding of five sequences on average, joint transcoding achieved about 2.3 dB improvements in the minimum picture-quality measure, and minimized the overall standard deviation of PSNR from 5.48 dB to 3.37 dB. Achieving a balanced picture quality by using joint transcoding is very important for many video applications, such as video on demand (VOD) services, in maintaining an acceptable picture-quality level during a worst transmission scenario.

Secondly, transcoding techniques of MPEG-4 compressed video in the DCT domain have been discussed in the Chapter 4. Many advanced video applications require manipulations of compressed video. As we know, there are two general approaches for processing compressed video bitstreams: one is the spatial-domain processing, the other is the compressed-domain processing. In the spatial domain, the video bitstream is first fully decompressed, then processed in the spatial domain, finally compressed again for storage or transmission. In the compressed domain, the pre-encoded bitstreams are partially decoded, and then processed in the DCT domain, and finally re-encoded. Compared to the spatial domain processing method, compressed domain method has several advantages such as small data to be processed, lower computational complexity, and preservation of image fidelity due to the absence of decompression and recompression operations. For transcoding of MPEG-4 compressed video with drift error correction in the DCT domain, inverse motion compensation in the DCT domain is a key to compensating drift errors in the anchor frames back to the prediction error frames. However, inverse motion compensation in the DCT domain is more complex than that in

the pixel domain, and it also becomes the neckbottle of DCT domain video processing applications.

In [20], Chang and Messerschmitt presented DCT-domain inverse motion compensation algorithm, which has been widely used for developing other compressed domain video processing methods[22][23][24]. In this thesis, a new DCT Coefficient Translation and Truncation Transformation Matrix (**DCTTTM**) based motion composition scheme has been proposed. Theoretically, it introduces more complexity than that in [20] (the performance is worse than that in[20] due to thresholding). Because linear transformations are involved these operations can be expressed as large matrix multiplications, and the complexity of inverse motion compensation based on DCTTTM is dependent on the number of nonzero elements of such large matrices. By carefully analysis and observation, most elements of such matrices would be very small with different motion vectors, and could be rounded to zero by applying suitable thresholds so that the computational complexity could be reduced. In this thesis, by offline thresholding and setting to zero of small elements in the matrices, a suboptimal low complexity scheme is developed proposed. Our simulation results show that when the threshold increases from 0.00 up to 0.10, the average PSNRs of both *Flowergarden* and *Foreman* degrade less than 0.4 dB, however, the average number of multiplications and additions needed for extracting one DCT block from the four adjacent blocks in the reference frame decreases from 2695 to 806, and from 4076 to 464, respectively. If a quality degradation of 0.2 dB is acceptable, then in our simulations, the threshold is selected as 0.05, and the average complexity savings is approximately 50%.

Finally, Chapter 4 introduced frame-skipping transcoding technique in the DCT

domain. Unlike non frame-skipping transcoding, the incoming motion vectors cannot be simply reused because the previous reference frame could be skipped according to target bitrate. Hence, motion re-estimation should be done in frame-skipping transcoding applications. However, motion estimation results in large computational complexity. In order to avoid much more complexity, several motion vector refinement algorithms for transcoding have been represented in the past years[25][26][27], and are basically processed in the pixel domain In this thesis, a new effective scheme based on WFDVS together with suboptimal motion estimation algorithm in the DCT domain was proposed. Our experimental results show that our algorithm greatly improves the performance (in PSNR) over the reused MVs scheme, and gets almost the same performance as the optimal motion estimation algorithm, while it requires less than 50% computational complexity of conventional ME algorithms. Moreover, this suboptimal ME algorithm can be combined with other ME algorithms such as TSS, Diamond Search, HSS in the DCT domain so that the computational complexity can be further reduced.

5.2. Future Work

Chapter 3 addressed rate control models used for transcoding of MPEG-4 compressed video, which are the same as those in MPEG-4 encoder. However, due to lack of the complexity S in the compressed domain, S is replaced with the approximated parameter S' defined in equation (3.3). Because S is approximated value, some requantization parameter QP2 would not be reasonable, which affects the performance of transcoding. To solve this problem, there is a need for some work to be done in the future: one is to eliminate these unreasonable requantization parameters by applying some criterion; the

other is to find out other parameters to evaluate the complexity of sequences in the compressed domain.

In Chapter 3, joint transcoding of multiple MPEG-4 compressed videos is also discussed. Joint transcoding provided a more constant quality between the sequences than independent transcoding. A new joint bit rate allocation algorithm based on approximated distortion was proposed in this thesis. Our simulation results show the average performance in PSNR doesn't outperform that of independent transcoding, while the variation in PSNR between the sequences is obviously reduced and the minimum PSNR is increased. There would be some modifications of proposed approximated distortion based joint allocation bitrate algorithm so as to balance performance in PSNR among the overall average, the minimum value, and the standard derivation of multiple transmitted sequences.

Moreover, since DCT coefficients are requantized with new quantization parameters, the coding pattern for each macroblock could be possible to determine again so as to improve coding efficiency. For example, some Intra MBs may be changed to Inter MBs or Inter MBs should be encoded in Intra mode.

Chapter 4 mainly focused on transcoding techniques in the DCT domain. Although three suboptimal motion compensation methods using thresholding have been proposed and the complexity savings of our method are substantial, future work could include differentially thresholding matrix elements so that less important ones are thresholded more heavily. As for frame skipping transcoding in the DCT domain, more work could be done.

In the DCT domain, half-pixel accuracy of motion vectors has not been considered since it will increase the complexity. Pixel accuracy of motion vector doubtlessly degrades

the performance of transcoder. Future researchers could do some work on half-pixel motion estimation in the DCT domain so as to improve the performance of transcoding and be compatible with MPEG-4 standard.

References

[1] Netravali A.N. and Haskell B.G. "Digital pictures, representation and compression and standards", 2nd edition, Plenum Press, New York, 1995.

[2] Barry G. Haskell, Atul Puri, and Arun N. Netravali, "Digital Video: An Introduction To MPEG-2", Digital Multimedia Standards Series, Chapman & Hall, 1997.

[3] Anthony Vetro, Charilaos Christopoulos, and Huifeng Sun, "Video Transcoding Architectures and Techniques: An Overview", IEEE Signal Processing Magazine, March 2003, pp18-29.

[4] I. Nagayoshi, H. Kasai, and H. Tominaga, "A study on the rate control method for MPEG transcoder considering drift-error propagation", Proceedings of International Conference on Image Processing, 2000, Volume 1, 10-13 Sept. 2000, pp960-963

[5] Oliver Werner, "Requantization for Transcoding of MPEG-2 Intraframes", IEEE Transactions on Image Processing, Vol.8, No.2, February 1999, pp179-191.

[6] Peng Yin; Vetro, A.; Bede Liu; Huifang Sun "Drift compensation for reduced spatial resolution transcoding", IEEE Transactions on Circuits and Systems for Video Technology, Volume 12, Issue 11, Nov. 2002, pp1009 -1020

[7] H. Sorial and W. E. Lynch and A. Vincent, "Selective Requantization for Transcoding of MPEG Compressed Video", ICME 2000, NY, USA, Vol.1, pp217-220.

[8] H. Sorial and W. E. Lynch and A. Vincent, "Joint Transcoding of Multiple MPEG Video Bitstreams", IEEE International Symposium on Circuits and Systems (ISCAS'99), Orlando, Florida. Vol.4, May 1999, pp251-254.

[9] G. Keesman and R. Hellinghuizen and Fokke Hoeksema and Geert Heideman, "Transcoding of MPEG Bitstreams", Signal Processing: Image Communication, Vol. 8, 1996, pp481-500.

[10] MPEG Video Group, "MPEG-4 Video Verification Model Version 14.0", ISO/IEC N2932, Melbourne, Australia, Oct. 1999.

[11] Thomas Sikora, "The MPEG-4 Video Standard Verification Model", IEEE Trans. on Circuits and Systems for Video Technology, Vol.7, No.1, Feb. 1997.

[12] R. Schafer and T. Sikora, "Digital Video Coding Standards and Their Role in Video Communications", Proc.IEEE, Vol. 83, June 1995, pp. 907-924.

[13] Mohammed Ghanbari, "Video Coding: an introduction to standard codecs", The Institute of Electrical Engineering, London, UK, 1999.

[14] "MPEG-4 Visual Working Draft Version 4.0", ISO/IEC JTC1/SC29/WG11 N1797 July 1997.

[15] Fernando Pereira and Touradj Ebrahimi, "The MPEG-4 Book", Prentice Hall PTR, Upper Saddle River, NJ07458, 2002.

[16] A. T. Erdem and M. I. Sezan, "Multi-generation Characteristics of the MPEG Video Compression Standards", IEEE Proceedings International Conference on Image Processing, Vol. 2, November 1994, pp933-937.

[17] C. Horne and T. Naveen and A. Tabatabai and R. O., Eifrig and A. Luthra, "Study of the Characteristics of the MPEG2 4:2:2 Profile-- Application of MPEG2 in Studio Environment", IEEE Transactions on Circuits and Systems for Video Technology, No. 3, Vol. 6, June 1996, pp251-272.

[18] H. Sorial and W. E. Lynch, "Degradation Mechanisms in Multigeneration of MPEG Compressed Video", Special issue on Visual Computing and Communications, Canadian Journal of Electrical and Computer Engineering, January-April 1998, No.1-2, Vol. 23, pp5-9.

[19] H. Sorial and W. E. Lynch, "Multigeneration of Transform Coded Images", Proceedings of SPIE Visual Communications and Image Processing, San Jose, California, Vol.3024, No.1, February 1997, pp1394-1405.

[20] ISO/IEC 13818(MPEG-2), "Information Technology-Generic Coding of Moving Pictures and Associated Audio Information"

[21] Shih fu Chang and David G. Messerschmitt, "Manipulation and Compositing of MC-DCT Compressed Video", IEEE Journal on Selected Areas in Communications, Vol. 13, No.1, Jan. 1995.

[22] Neri Merhav and Vasudev Bhaskaran, "Fast Algorithms for DCT-Domain Image Down-Sampling and for Inverse Motion Composition ", IEEE Trans. on Circuit Systems and Video Technology, Vol. 7, No. 3, June 1997.

[23] Shizhong Liu and Alan C. Bovik, "Local Bandwidth Constrainted Fast Inverse Motion Compensation for DCT-Domain Video Transcoding", IEEE Trans. on Circuit Systems and Video Technology, No.5, Vol. 12, May 2002.

[24] Pedro A.A. Assuncao and Mohammed Ghanbari, "A Frequency-Domain Video Transcoder for Dynamic Bit-Rate Reduction of MPEG-2 Bit Streams ", IEEE Trans. on Circuit Systems and Video Technology, No. 8, Vol. 8, Dec. 1998.

[25] Jeongnam Youn and Ming-Ting Sun and Chia-Wen Lin, "Motion Vector Refinement for High-Performance Transcoding", IEEE Trans. on Multimedia, No. 1, Vol. 1, March 1999.

[26] Mei-Juan Chen and Ming-Chung Chu and Chih-Wei Pan, "Efficient Motion-Estimation Algorithm for Reduced Frame-Rate Video Transcoder", IEEE Trans. on Circuit Systems and Video Technology, No.4, Vol. 12, April 2002.

[27] Kai-Tat Fung and Yui-Lam Chan and Wan-Chi Siu, "New Architecture for Dynamic Frame-Skipping Transcoder", IEEE Trans. on Image Processing, No.8, Vol. 11, August 2002.

[28] Y. Nakajima and H. Hori and T. Kanoh, "Rate Conversion of MPEG Coded Video by Re-quantization Process", IEEE Proceedings International Conference on Image Processing, Vol.3, October 1995, pp408-411.

[29] Kwang-deok Seo and S.Lee and J. Kim and J. Koh, "Rate Control Algorithms for Fast Bit-Rate Conversion Transcoding", IEEE Transactions on Consumer Electronics, No.4, Vol. 46, November 2000.

[30] Anthony Vetro and Huifeng Sun and Yao Wang, "Object-Based Transcoding for Adaptive Video Content Delivery", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 3, March 2001.

[31] ISO/IEC, "Test Model 5", ISO/IEC-JTC1/SC29/WG11, MPEG 93/457, April 1993.

[32] MoMuSys Codec, "MPEG4 Verification Model", ISO/IEC JTC1/SC29/WG11 Coding of Moving Pictures and Associated Audio MPEG 97, March 1997.

[33] T. Chiang and Y.Q. Zhang, "A New Rate Control Scheme using Quadratic Rate-Distortion Modeling", IEEE Trans. on Circuit Systems and Video Technology, February 1997.

[34] Wang and A. Vincent, "Joint Coding for Multi-Program Transmission", Proceedings of the IEEE International Conference on Image Processing (ICIP), September 1996, pp425-428.

[35] Anthony Vetro and Huifeng Sun and Yao Wang, "MPEG-4 Rate Control for Multiple Video Objects", IEEE Transactions on Circuits and Systems for Video Technology, No.1, vol.9, February 1999.

[36] A. Guha and D. J. Reininger, "Multichannel Joint Rate Control of VBR MPEG Encoded Video for DBS Applications", IEEE Transactions on Consumer Electronics, Vol.40, No.3, August 1994, pp616-623.

[37] S. C. Liew and H. H. Chan, "Lossless Aggregation: A Scheme for Transmitting Multiple Stored VBR Video Streams Over A Shared Communications Channel

Without Loss of Image Quality", IEEE Journal on Selected Areas in Communications, Vol.15, No.6, August, 1997, pp1181-1189.

[38] L. Wang and A. Vincent, "Bit Allocation for Joint Coding of Multiple Video Programs", Proceedings of SPIE Visual Communications and Image Processing, San Jose, CA, February 1997, pp149-158.

[39] Koga T., Iinuma K., Hirano A., Lijima Y. and Ishiguro T. "Motion Compensated Interframe Coding for Video Conferencing", Proce. National Telecomm. Conf. pp G5.3.1-G5.3.5, New Orleans, LA, Nov. 29-Dec. 3. 1981.

[40] S. Zhu and K. K. Ma, "A New Diamond Search Algorithm for Fast Block-matching Motion Estimation", IEEE Trans. Image Processing, Vol. 9, pp 287-290, Feb. 2000.

[41] W. E. Lynch, " Bidirectional Motion Estimation Based on P Frame Motion Vectors and Area Overlap", ICASSP, March 23-26, 1992, San Francisco Marriott, California, USA. pp III-445.

[42] P.A.A. Assuncao and I. Ghanbari, "Optimal Transcoding of Compressed Video", Proceedings of International Conference on Image Processing, 1997, Vol. 1, Oct. 1997, pp 739 –742.

[43] W. H. Chen, C. H. Smith, and S. C. Fralick, "A Fast Computational Algorithm for the Discrete Cosine Transform", IEEE Trans. on Communications, vol. Com-25, no. 9, pp. 1004-1009, September 1977

[44] B. G. Lee, "A new algorithm to compute the discrete cosine transform", IEEE Trans. on acoustics, speech, and signal processing, vol. ASSP-32, no. 6, pp. 1243-1245, December 1984

[45] H. S. Hou, "A fast recursive algorithm for computing the discrete cosine transform", IEEE trans. on acoustics, speech, and signal processing, vol. ASSP-35, no. 10, pp. 1455-1461, October 1987

[46] Z. Wang, "Fast Algorithms for Discrete W-Transform and for the Discrete Fourier Transform", IEEE trans. on acoustics, speech and signal processing, vol. ASSP-32, no. 4, pp. 803-816, August 1984.

[47] N. Suehiro, M. Hatori, "Fast algorithms for the DFT and other Sinusoidal Transforms", IEEE Trans. on acoustics, speech, and signal processing, vol. ASSP-34, no. 3, pp. 642-664, June 1986

[48] C. Loeffler, A. Lightenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11-multiplications", ICASSP-89, vol. 2, pp. 988 –991, 1989

[49] Y. Jeong, I. Lee, H. S. Kim, and K. T. Park, "Fast DCT algorithm with fewer multiplication stages", *Electronic Letters*, vol. 34, No. 8, pp. 723-724, April 1998.

[50] N. Merhav and V. Bhaskaran, " A fast algorithm for DCT-domain inverse motion compensation", HPL Technical Report #HPL-95-17, Feb. 1995.