TRANSCODING OF H264 BITSTREAM TO MPEG-2 BITSTREAM

by

SREEJANA SHARMA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2007

ACKNOWLEDGEMENTS

The successful completion of my thesis is due to the invaluable guidance and support of a number of people.

I would like to express profound gratitude to my advisor, Dr.K.R.Rao for his invaluable support, encouragement, supervision and useful suggestions throughout this research work. His excellent ideas and dedication helped me successfully complete my research work.

I would also like to express my heartfelt gratitude to my colleagues in Intel Corporation for their invaluable industry insight and experience to help me understand and grow in the field of video processing.

I would also like to sincerely thank my thesis committee members Dr.S.Oraintara and Dr.Z.Wang for being a part of the culmination of my thesis.

I would also like to thank my alma mater, the University of Texas at Arlington, for providing me with an excellent foundation to grow and excel in my field. I would like to thank my friend Rahul for his help and guidance.

Last but not the least, I would also like to thank my parents, my friend Sandip, and above all GOD for always being my source of inspiration and guiding light.

April 9, 2007

ABSTRACT

TRANSCODING OF H.264 BITSTREAM TO MPEG-2 BITSTREAM

Publication No. ______

Sreejana Sharma, M.S.

The University of Texas at Arlington, 2007

Supervising Professor: Dr.K.R.Rao

H.264/AVC was recently developed by the JVT (Joint Video Team). This new standard fulfills significant coding efficiency, simple syntax specifications and seamless integration of video coding into all current protocols and multiplex architectures. The H.264 specification represents a significant advancement in the field of video coding technology by providing MPEG-2 comparable video quality at an average of half the required bandwidth. Since widespread use of H.264 is anticipated, many legacy systems including all Digital TVs and home receivers use MPEG-2. This leads to the need for an efficient architecture that significantly employs the lower cost of H.264 video and does not require a significant investment in additional video coding hardware [14]. An H.264

to MPEG-2 transcoder with low computational complexity and comparable quality to an MPEG-2 only encoded decoded sequence is one such effort in the above direction.

Algorithm: The proposed research divides the process of transcoding into two parts. The first part deals with the transcoding of I frames where the best method is complete encoding and decoding of the input H.264 bit stream and then re-encoding it in the MPEG-2 format. This is the best method is terms of complexity and quality too due to the advance features of H.264 which is not supported in MPEG-2, like H.264 performs intra prediction in the pixel domain and then applies a 4x4 integer transform across the residual. However in the case of MPEG-2, intra prediction is done in the transform domain and it basically involves the process of dividing every macroblock into 8x8 blocks and apply a 2D-DCT across them to get the transform coefficients. In the case of P frame transcoding, motion vectors are extracted from the H.264 decoding stage and then they are scaled and summed depending on the H.264 block sizes to get an MPEG-2 compatible motion vector for the entire 16x16 macroblock. The obtained motion vectors are then refined over a 2 pixel search window to get the best match and then reused in the MPEG-2 encoding stage. This method produces comparable quality transcoded output to simple MPEG-2 encoded decoded output with a significant saving in the motion estimation time at the MPEG-2 encoding stage.

TABLE OF CONTENTS

Chapter

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

LIST OF ACRONYMS

JVT: Joint Video Team

AVC: Advanced video coding.

HDTV: High Definition Television

DCT: Discrete Cosine Transform

M.E: Motion Estimation

M.C: Motion Compensation

MB: Macroblock

GOP: Group of pictures

VLC: Variable length coding

VLD: Variable length decoding

IQ: Inverse quantization:

IDCT: Inverse DCT.

CABAC: Context-based adaptive binary arithmetic coding

CAVLC: Context-based adaptive variable length coding

VLC: Video coding layer

NAL: Network abstraction layer

ISO: International Organization for Standardization.

ITU: International telecommunication union.

VCEG: Video coding experts group.

QP: Quantization parameter.

MV: Motion vector.

PVR: Personal video recorder.

CHAPTER 1

INTRODUCTION

1.1 Introduction

In this fast growing world of multimedia and telecommunications there is a great demand for efficient usage of the available bandwidth. With the growth of technology there is an increase in the number of networks, types of devices and different content representation formats as a result of which interoperatibility between different systems and networks is gaining in importance. Transcoding of video content is one such effort in this direction. Among its various applications my work focuses on the ability of transcoding to convert video from one format to another that is the transcoding of H.264/AVC [4][10][13] bitstream to MPEG-2 [7][11] bitstream. The main goals of an efficient transcoder are to maintain the quality of the transcoded bitstream to the one obtained by direct decoding and re-encoding of the input stream, to reuse the information available in the bitstream as much as possible so as to avoid multigenerational deterioration and last but not the least the process should be efficient, low in complexity and achieve the highest quality possible.

Having said this, now arises the question for why the need for H264/AVC to MPEG-2 transcoding [1][3]? In order to provide better compression of video as compared to previous standards, H.264/AVC was recently developed by the JVT (Joint Video Team). This new standard fulfills significant coding efficiency, simple syntax specifications and seamless integration of video coding into all current protocols and multiplex architectures.

The H.264 specification represents a significant advancement in the field of video coding technology by providing MPEG-2 comparable video quality at an average of half the required bandwidth. Since widespread use of H.264 is anticipated, many legacy systems including all Digital TVs and home receivers use MPEG-2. This leads to the need for an efficient architecture that significantly employs the lower cost of H.264 video and does not require a significant investment in additional video coding hardware [14].

1.2 Outline of the work

My work is based on the transcoding of H.264/AVC bitstream to MPEG-2 bitstream using a fast, efficient and low complexity approach. It essentially reuses a lot of information from the decoding stage of the H.264 bitstream to generate the transcoded bitstream. As I am working on transcoding of H.264 from baseline profile to MPEG-2 simple profile, there are no bi-directionally predicted (B) pictures taken into consideration. My work is divided into two parts, first is the Intra (I) frame transcoding in which I have used the brute force method that is encoding and decoding using H.264 to get the H.264 video and then re-encoding it into an MPEG-2 bitstream. Second is the Inter (P) frame transcoding in which I have reused information from the H264 decoding stage and then processed and organized them based on MPEG-2 standard.

The thesis is organized as follows: Chapter 2 provides a description of the MPEG-2 and H.264 video bitstreams; Chapter 3 describes the various transcoding architectures and their pros and cons; Chapter 4 describes the transcoding of I frames; Chapter 5 describes the transcoding of P frames and Chapter 6 concludes with a summary and topic for future research.

CHAPTER 2

VIDEO BITSTREAMS

2.1 MPEG-2

MPEG-2 [7][19] is an extension of the MPEG-1 video coding standard. MPEG-1 was designed to code progressively scanned video at a bitrate of 1.5 Mbps. On the other hand MPEG-2 is directed at broadcast formats at higher data rates of 4 Mbps (DVD) and 19 Mbps (HDTV). It provides extra algorithmic tools for coding interlaced video, supports a wide range of bit rates and provides for multichannel surround sound coding. The set of tools in MPEG-2 is known as profiles and the constraint on the parameters such as picture size and bitrate is known as Levels.

2.2 MPEG-2 video encoder

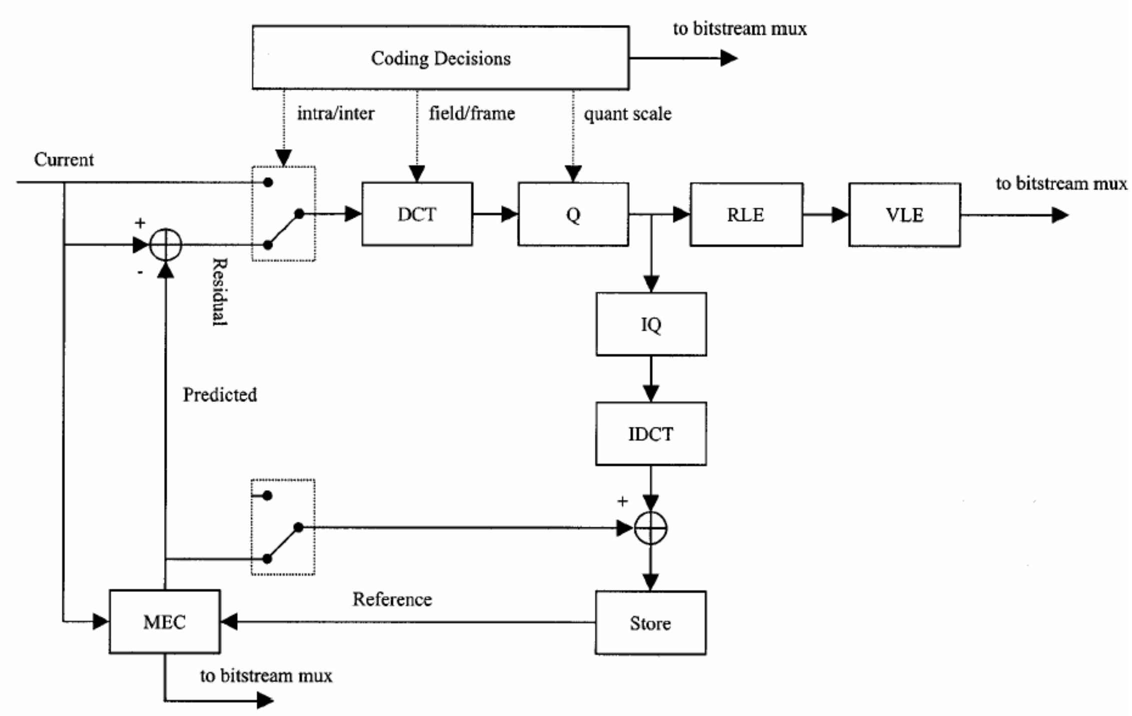


Figure 2.1 MPEG-2 encoder [18].

The basic block diagram of an MPEG-2 encoder is as shown in the figure 2.1. The different functions that are part of the MPEG-2 encoder are explained below:

DCT: The MPEG-2 encoder uses 8x8 2-D DCT [11]. In the case of intra frames, it is applied to 8x8 blocks of pels and in the case of inter frames it is applied to 8x8 blocks of the residual (motion compensated prediction errors). Since DCT is more efficient in compressing correlated sources, intra pictures DCT compress more efficiently than inter pictures.

Quantizer: The DCT coefficients obtained above are then quantized by using a default or modified matrix. User defined matrices may be downloaded and can occur in the sequence header or in the quant matrix extension header. The quantizer step sizes for DC coefficients of the luminance and chrominance components are 8, 4, 2 and 1 according to the intra DC precision of 8, 9, 10 and 11 bits respectively [7]. For the AC coefficients the corresponding header information to the quantizer scale factors are the quantizer scale type in the picture coding extension header and the quantizer scale code in the slice and MB headers.

Motion estimation and compensation: In the motion estimation process, motion vectors for predicted and interpolated pictures are coded differentially between macroblocks. Basically, the two motion vector components, the horizontal component first and then the vertical component are coded independently. In B pictures, four motion vector components corresponding to forward and backward motion vectors are transmitted. The motion compensation process forms prediction from previously decoded pictures using the motion vectors that are of integer and half-pel resolutions.

Coding decisions: There are four different coding modes in MPEG-2. These modes are chosen based on whether the encoder encodes a frame picture as a frame or two fields or in the case of interlaced pictures it can chose to encode it as two fields or use 16x8 motion compensation.

Scanning and VLC: The quantized transform coefficients are scanned and converted to a one dimensional array. Two scanning methods are available: zigzag scan which is typical for progressive (non-interlaced) mode processing, and alternate scan which is more efficient for interlaced format video.

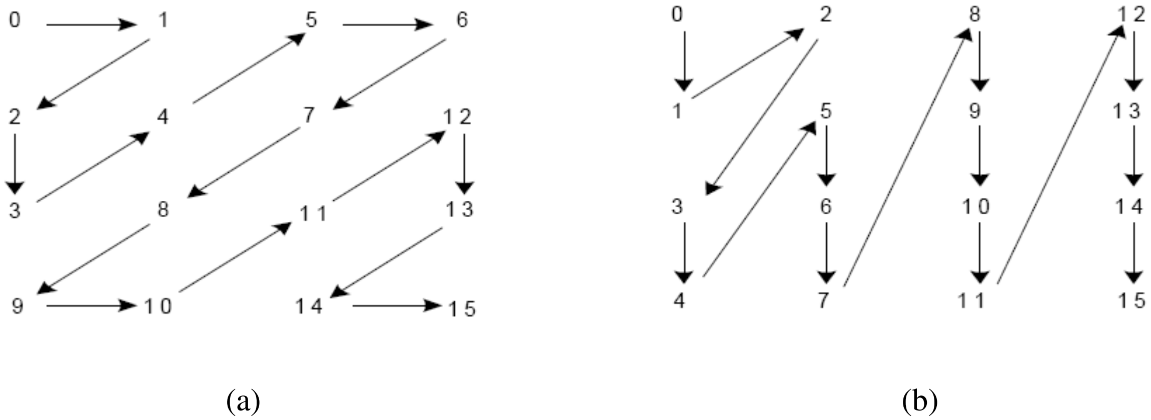


Figure 2.2 Different scan patterns for a 4x4 block: (a) zig-zag scan pattern (b) alternate scan pattern [19].

The list of values produced by scanning is then entropy coded using a variable length code (VLC).

2.3 Structure of MPEG-2 coded video data

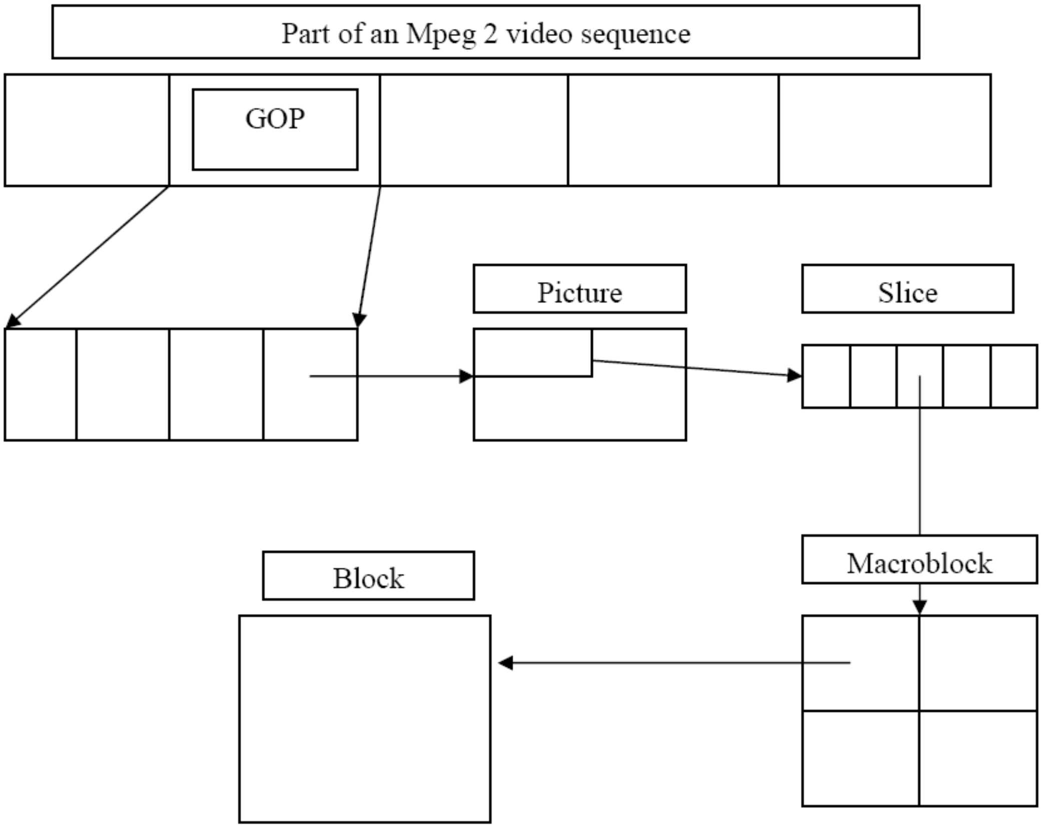
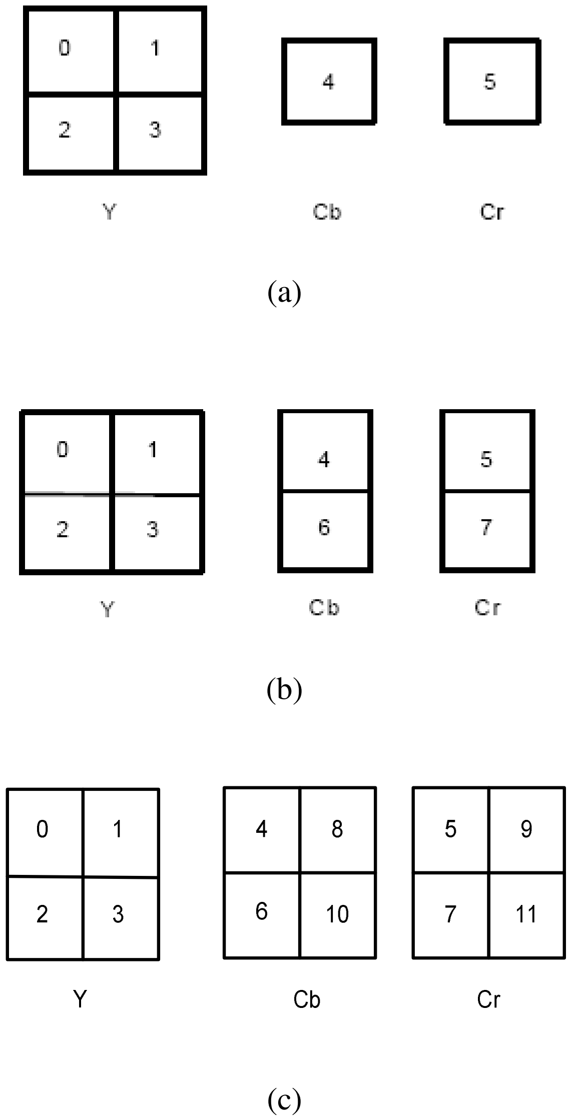


Figure 2.3 Different parts of MPEG-2 coded video data.

2.3.1. Video Sequence

An MPEG-2 video sequence is an ordered stream of bits, with a special bit pattern marking the beginning and ending of a logical section. Each video sequence is composed of a series of Group of Pictures (GOP). A GOP is composed of a sequence of pictures (frames). A frame is composed of a series of slices. A slice is composed of a series of macroblocks. A macroblock refers to source and decoded data or to the corresponding data elements. There can be three chrominance formats for a macroblock namely 4:2:0 that is 4 luminance blocks and 1 each chrominance blocks, 4:2:2 that is 4 luminance blocks and 2 each chrominance blocks and finally 4:4:4 that is 4 luminance and 4 each chrominance blocks. The three chroma formats are depicted in figure 2.4.



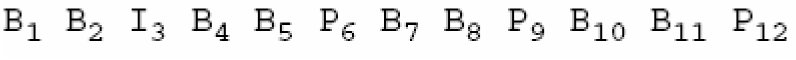
(a)

(b)

(c)

Figure 2.4 Layout of the three chroma formats: (a) 4:2:0 (b) 4:2:2 (c) 4:4:4 [19].

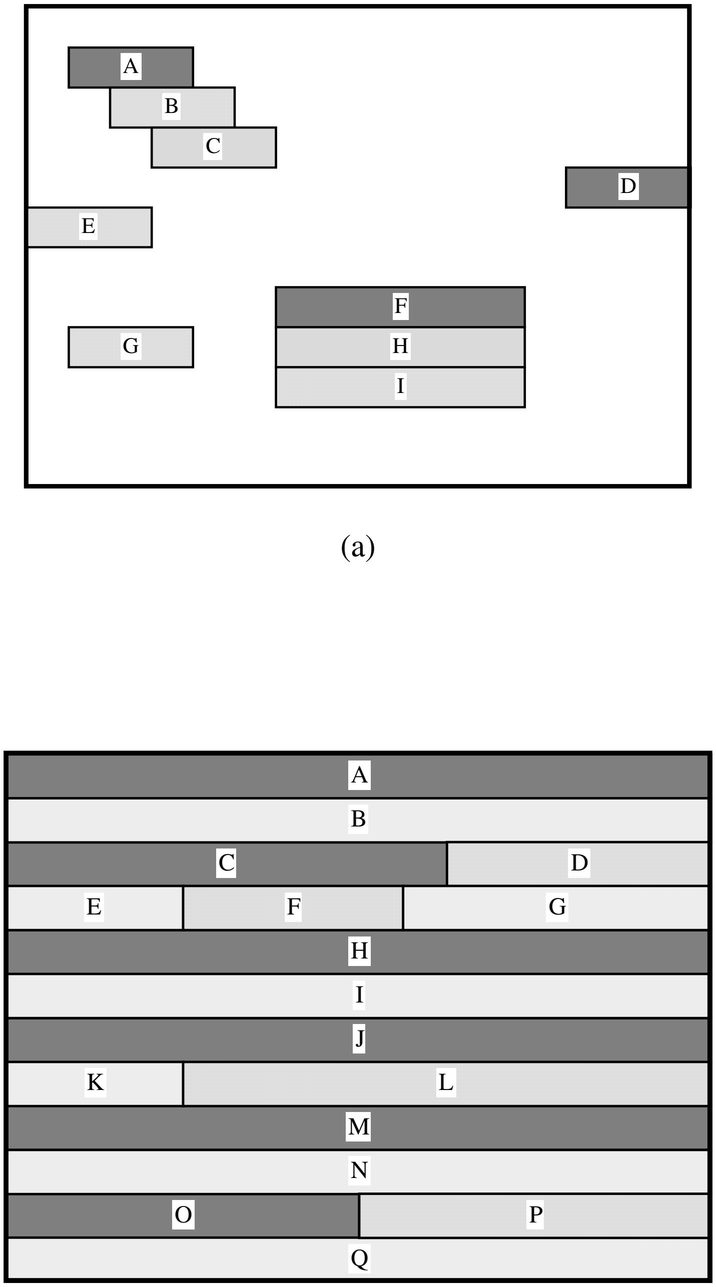
The GOP structure is intended to assist random access into a sequence. A GOP is an independently decodable unit that can be of any size as long as it begins with an I-frame. (Sequences are a higher level structure than GOPs, and may contain information about quantization tables. Their information is needed to decode all following GOPs.) GOPs are independently decodable if they are closed, for example a GOP with the pattern IBBP is closed, but the pattern IB is not. A typical GOP pattern is as shown in figure 2.4 [11].



B_1 B_2 I_3 B_4 B_5 P_6 B_7 B_8 P_9 B_{10} B_{11} P_{12}

Figure 2.5 Typical GOP structure where I is Intra picture, P is predicted picture and B is bi-predicted picture.

Each slice is (in a sense) an independently decodable unit too. There can be 1 slice per frame, 1 slice per macroblock, or anything in between. The slice structure is intended to allow decoding in the presence of errors. Serendipitously, it also allows parallel encoding/decoding at the slice level.



(a)

(b)

Figure 2.6 Slice structures: (a) Most general slice structure (b) Restricted slice structure [19].

In software [19], a coded video sequence commences with a sequence header, which may optionally be followed by a sequence extension and group of pictures header. The GOP header is an optional header that can be used immediately before a coded I frame to indicate to the decoder if the first consecutive B frame immediately following the coded I frame can be reconstructed properly in the case of random access. If the preceding reference frame is not available the B frames, if any, cannot be reconstructed properly unless they use backward prediction or intra coding. A GOP header also contains time code information that is not used by the decoding process. The order of the coded frames is the order in which the decoder processes them but that is not necessarily the correct order for display. The video sequence is terminated by a sequence end code.

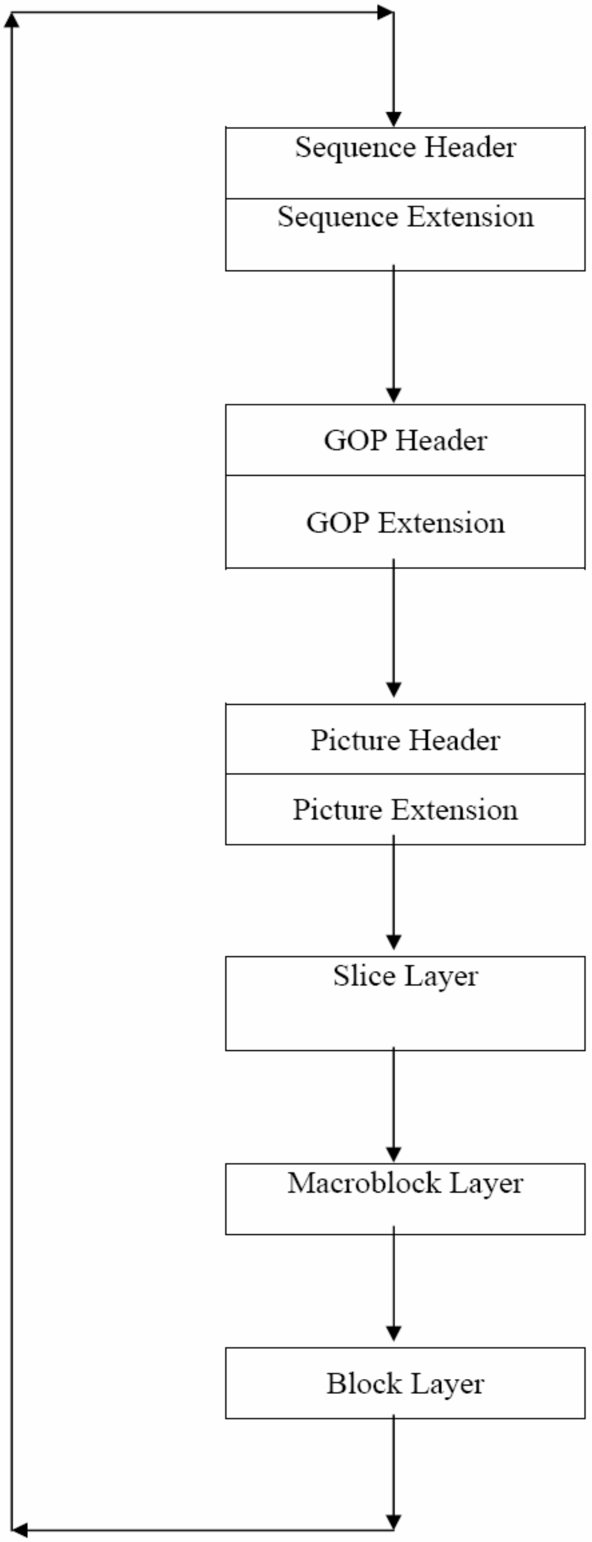


Figure 2.7 Hierarchy of MPEG-2 coded video data [19].

2.3.2. Picture Types

There are two different picture types in MPEG-2 namely Intra pictures and Inter pictures. The picture types are classified on the basis of the prediction modes used [7][19].

Intra pictures (I frame): Intra pictures are coded on the basis of spatial correlation and they do not refer to any reference pictures. As a result of only spatial redundancy reduction, these pictures achieve only moderate compression. These pictures are periodically used to provide access points in the bitstream where the decoding begins.

Inter pictures (P frame and B frame): Inter pictures are coded on the basis of temporal correlation as well as spatial correlation. There are two type of inter pictures in MPEG-2 known as predicted (P) pictures and bi-directionally predicted (B) pictures.

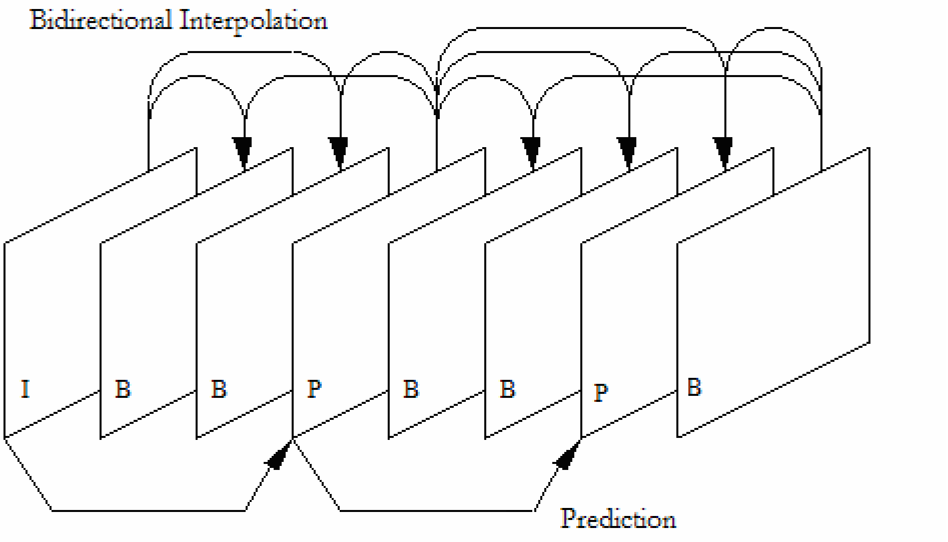
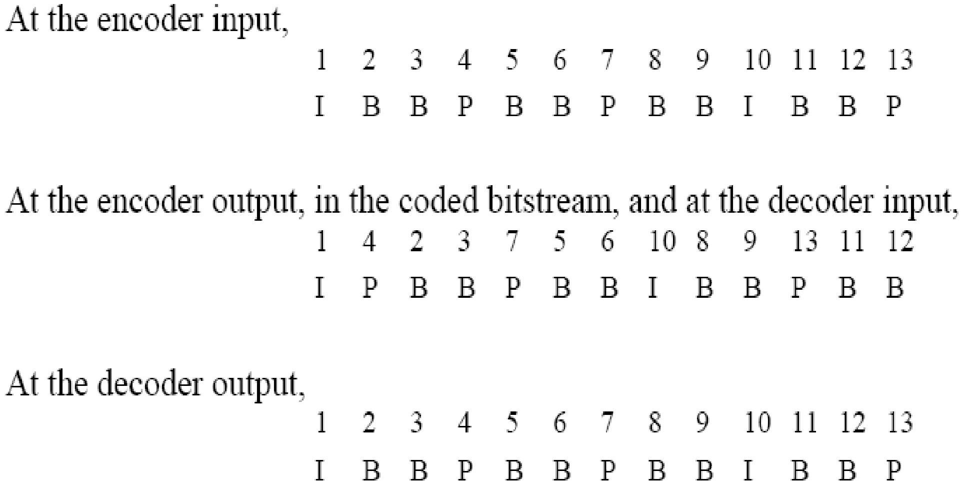


Figure 2.8 Illustration of temporal prediction[24].

P pictures are based on the concept of temporal redundancy reduction and so they can use the previous I or P pictures for motion compensation and may themselves be used as a reference picture for further prediction. Since P pictures use both spatial and temporal redundancy reduction techniques, they provide a higher level of compression than I pictures. B pictures also use inter and intra prediction like P pictures with an addition that B picture can use previous and next I and P pictures as reference pictures leading to more compression than P pictures. However in the case of B pictures, picture reordering is required from display order to decoding order so that the B picture is transmitted after the previous and next reference pictures. This type of reordering results in a delay depending on the number of consecutive B pictures.



At the encoder input,

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	I	B	B	P

At the encoder output, in the coded bitstream, and at the decoder input,

1	4	2	3	7	5	6	10	8	9	13	11	12
I	P	B	B	P	B	B	I	B	B	P	B	B

At the decoder output,

1	2	3	4	5	6	7	8	9	10	11	12	13
I	B	B	P	B	B	P	B	B	I	B	B	P

Figure 2.9 Illustrating sequence re-ordering in the presence of B pictures [19].

2.3.3. Profiles and Levels

A profile is defined as a set of algorithmic tools that can be used in video coding and a level sets the constraints on the picture parameters such as bit rates, picture sizes etc. Profiles are divided into two types in MPEG-2, non-scalable profile and scalable profile. Under non scalable profile there are two types, simple profile and main profile [7][19].

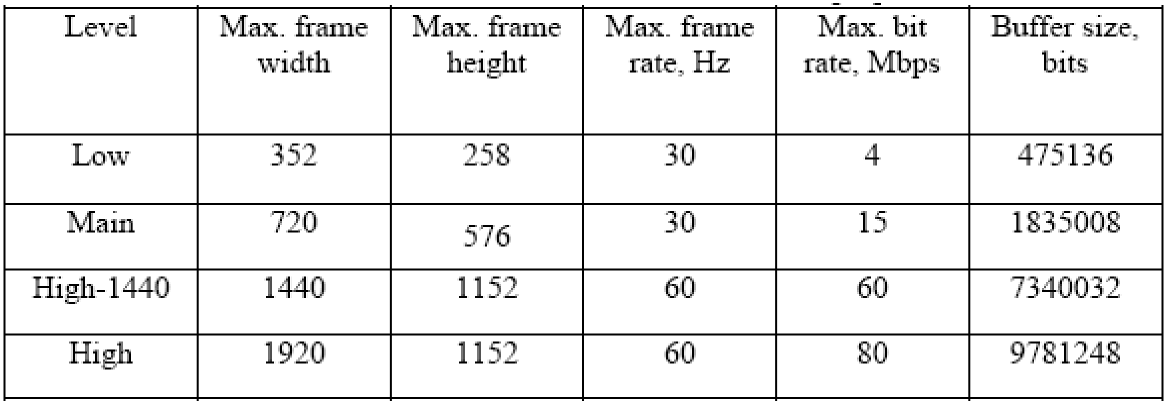
Simple profile: There are no B pictures in simple profile and hence no coding delay. As a result of this feature, this profile is suitable for low delay applications such as video conferencing where the acceptable overall delay is up to 100ms.

Main profile: This is the most widely used profile. B pictures are present in this profile leading to the introduction of coding delay of about 120ms in order to allow for picture reordering.

Scalable profiles are used in the case of scalable video coding and it is out of the scope of this work.

MPEG-2 defines four levels of coding constraints.

Table 2.1 Level constraints in MPEG-2 [19]

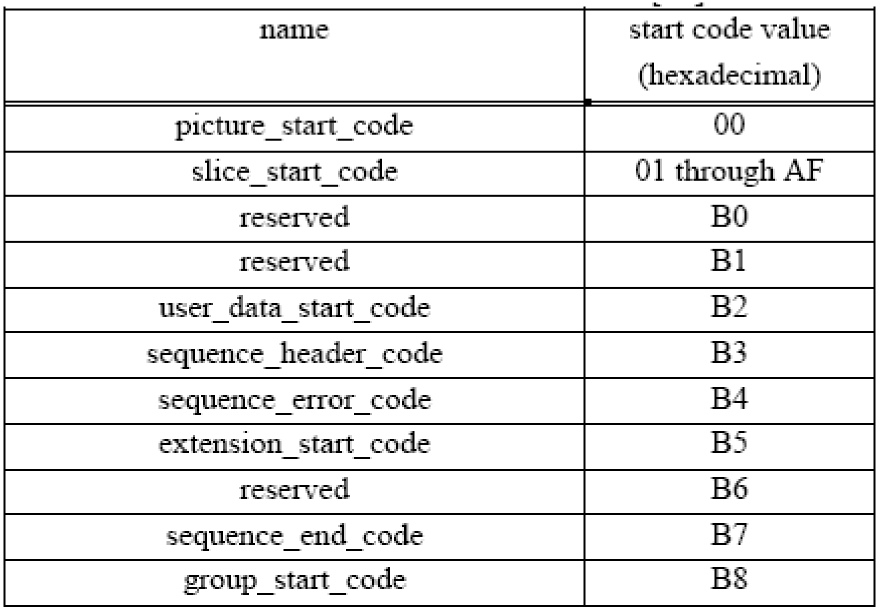


Level	Max. frame width	Max. frame height	Max. frame rate, Hz	Max. bit rate, Mbps	Buffer size, bits
Low	352	258	30	4	475136
Main	720	576	30	15	1835008
High-1440	1440	1152	60	60	7340032
High	1920	1152	60	80	9781248

2.4 MPEG-2 video bitstream syntax

As described earlier we know that the video sequence starts with a sequence header then the group of pictures and so on. In the MPEG-2 bitstream, start codes are used to signal the start of the different parts of the bitstream. Start codes are specific bit patterns that do not otherwise occur in the video bitstream. Each start code has a start code prefix which is a string of 23bits with value zero and the last bit with value one. All start codes are byte aligned that is the first bit of the start code is also the first bit of a byte. The various MPEG-2 start codes are given in Table 2.2 [19]:

Table 2.2 MPEG-2 start code values [19]



name	start code value (hexadecimal)
picture_start_code	00
slice_start_code	01 through AF
reserved	B0
reserved	B1
user_data_start_code	B2
sequence_header_code	B3
sequence_error_code	B4
extension_start_code	B5
reserved	B6
sequence_end_code	B7
group_start_code	B8

2.5 MPEG-2 video decoder

At the decoder side, the quantized DCT coefficients are reconstructed and inverse transformed to produce the prediction error. This predicted error is then added to the motion compensated prediction generated from previously decoded picture to produce the reconstructed output.

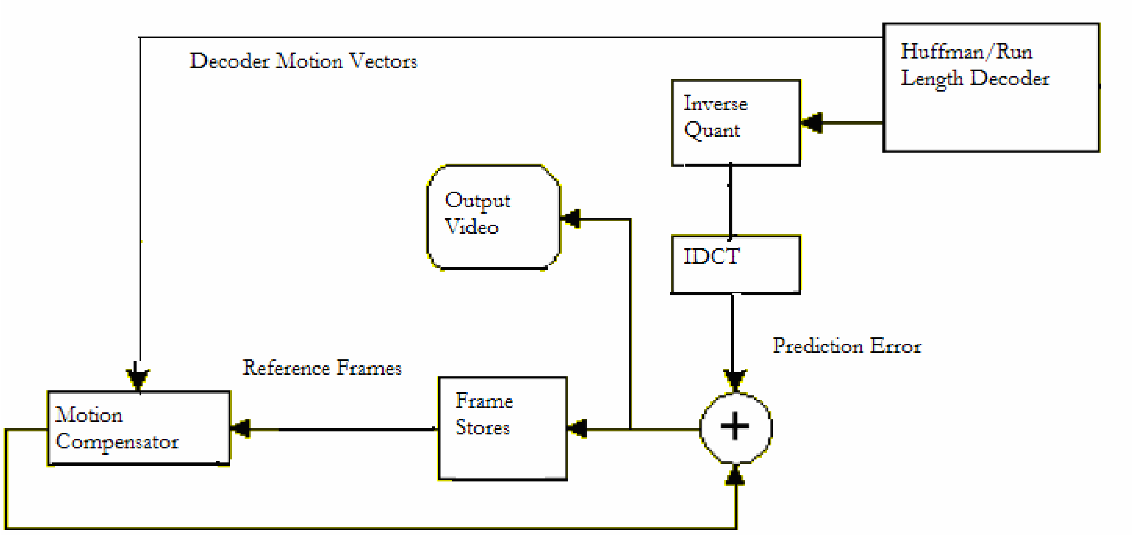


Figure 2.10 MPEG-2 decoder block diagram [11].

The different parts of an MPEG-2 decoder are as described below:

Variable length decoding: This process involves the use of a table defined for decoding intra DC coefficients and three tables one each for non intra DC coefficients, intra AC coefficients and non intra AC coefficients. The decoded values basically infer one of three courses of action: end of block, normal coefficients and escape coding.

Inverse scan: The output of the variable length decoding stage is one dimensional and of length 64. Inverse scan process converts this one dimensional data into a two dimensional array of coefficients according to a predefined scan matrix as shown in figure 4.4.

Inverse quantization: At this stage the two dimensional DCT coefficients are inverse quantized to produce the reconstructed DCT coefficients. This process involves the rescaling of the coefficients by essentially multiplying them by the quantizer step size. The quantizer step size can be modified by using either a weighing matrix or a scale factor. After performing inversion quantization, saturation and mismatch control operations are performed. Saturation is employed to keep the DCT coefficients in the range of [-2048:+2047]. After this mismatch control operation is performed to avoid large and visible mismatches between the state of the encoder and decoder.

Inverse DCT: Once the reconstructed DCT coefficients are obtained, a 2D 8x8 inverse DCT is applied to obtain the inverse transformed values. These values are then saturated to keep them in the range of [-256:+255]. In a not skipped macroblock, if the pattern code is one for a particular block in the macroblock then the corresponding coefficient data for that block is included in the bitstream. Or else if the pattern code is zero or the macroblock is skipped then that block contains no coefficient data.

Motion Compensation: During this stage, predictions from previously decoded pictures are combined with the inverse DCT transformed coefficient data to get the final decoded output.

2.6 H.264/AVC

H.264/AVC [4][10][13][22] was developed by the JVT (Joint Video Team) to achieve MPEG-2 [7][11][19] quality compression at almost half the bit rate. H.264/AVC provides significant coding efficiency, simple syntax specifications, and seamless integration of video coding into all current protocols and multiplex architectures. H.264 supports various applications such as video broadcasting, video streaming, and video conferencing over fixed and wireless networks and over different transport protocols.

2.7 H.264/AVC video encoder

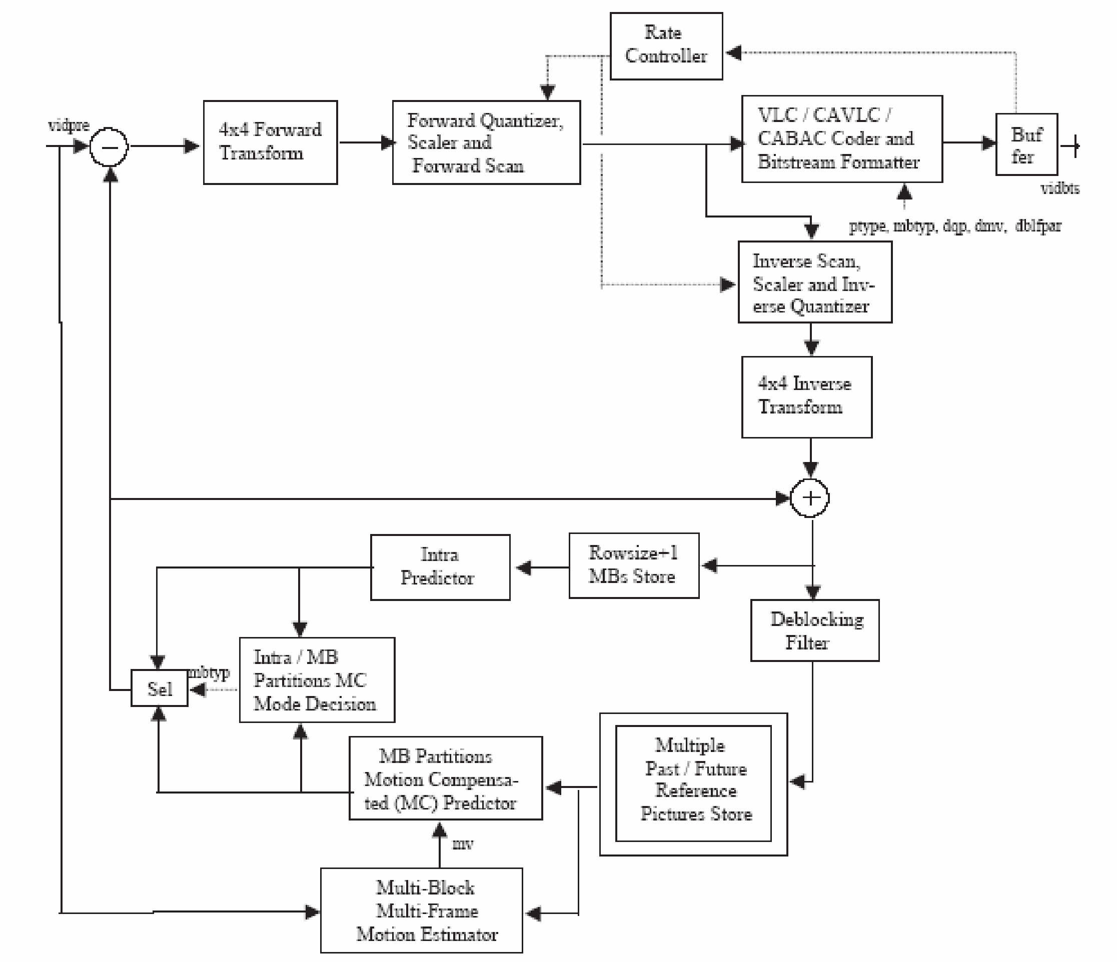

Rate Controller
vidpre
4x4 Forward Transform
Forward Quantizer, Scaler and Forward Scan
VLC / CAVLC / CABAC Coder and Bitstream Formatter
Buf fer
vidbts
ptype, mbtyp, dqp, dmv, dblfpar
Inverse Scan, Scaler and Inv-erse Quantizer
4x4 Inverse Transform
Intra Predictor
Rowsize+1 MBs Store
Deblocking Filter
Sel
mbtyp
Intra / MB Partitions MC Mode Decision
MB Partitions Motion Compensa-ted (MC) Predictor
Multiple Past / Future Reference Pictures Store
mv
Multi-Block Multi-Frame Motion Estimator

Figure 2.11 H.264/AVC encoder block diagram [10].

The basic block diagram of an H.264/AVC video encoder is as shown in Fig.2.11. The different functional blocks of the H.264 encoder are described below:

4x4 Integer transform: The H.264 employs a 4x4 integer DCT transform as compared to 8x8 DCT transformed by the previous standards. The smaller block size leads to a significant reduction in ringing artifacts. Also, the 4 x 4 transform has the additional benefit of removing the need for multiplications.

Quantization and scan: The H.264 standard specifies the mathematical formulae of the quantization process. The scale factor for each element in each sub block varies as a function of the quantization parameter associated with the macroblock and as a function of the position of the element within the sub block. The rate control algorithm controls the value of the quantization parameter. Two types of scan pattern are used for 4x4 blocks namely one for frame coded macroblocks and one for field coded macroblocks.

Context-based adaptive variable length coding (CAVLC) and Context-based adaptive binary arithmetic coding (CABAC) entropy coding: H.264 uses different variable length coding methods in order to match a symbol to a code based on the context characteristics. They are context-based adaptive variable length coding (CAVLC) and context-based adaptive binary arithmetic coding (CABAC). All syntax elements except for the residual data are encoded by the Exp-Golomb codes. In order to read the residual data (quantized transform coefficients), zig-zag scan (interlaced) or alternate scan (non-interlaced or field) is used. For coding the residual data, a more sophistical method called CAVLC is employed. Also, CABAC is employed in Main and High profiles, CABAC has more coding efficiency but higher complexity compared to CAVLC.

Deblocking filter: H.264 employs a deblocking filter to reduce the blocking artifacts in the block boundaries and stops the propagation of accumulated coded noise.

The filter is applied after the inverse transform (before reconstructing and storing the macroblock for future predictions) and in the decoder (before reconstructing and displaying the macroblocks). The deblocking filter is applied across the edges of the macroblocks and the sub-blocks. The filtered image is used in motion compensated prediction of future frames and helps achieve more compression.

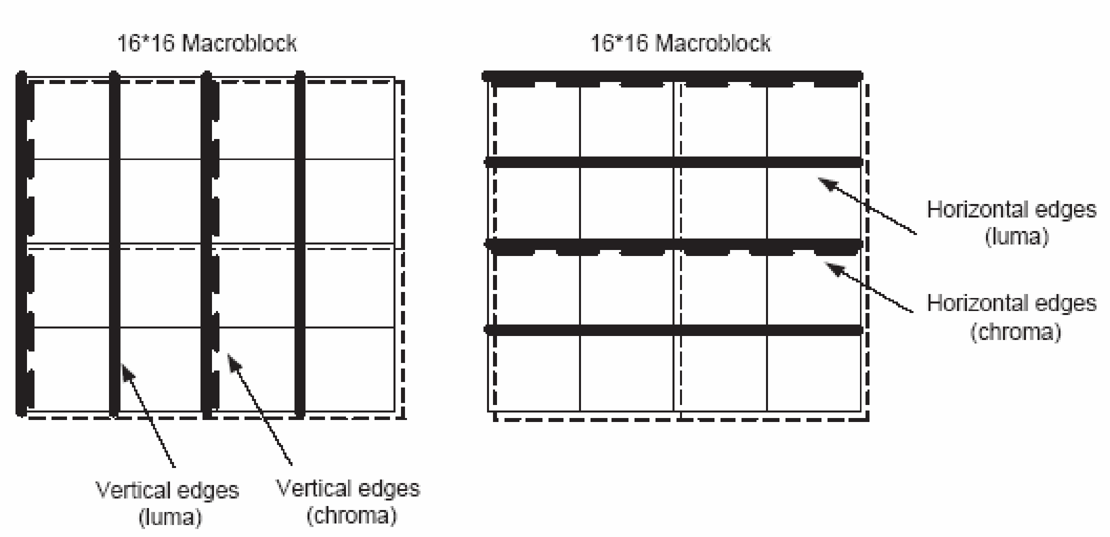


Figure 2.12 Diagram depicting how the loop filter works on the edges of the blocks and sub-blocks.

Intra prediction: During intra prediction, the encoder derives a predicted block based on its prediction with previously decoded samples. The predicted block is then subtracted from the current block and then encoded. There are a total of nine prediction modes for each 4x4 luma block, four prediction modes for each 16x16 luma block and four modes for each chroma block.

Inter prediction: Inter prediction is performed on the basis of temporal correlation and consists of motion estimation and motion compensation. As compared to the previous standards, H.264 supports a large number of block sizes from 16x16 to 4x4. Moreover H.264 supports motion vector accuracy of one-quarter of the luma sample.

Reference pictures: Unlike the previous standards that just use the immediate previous I or P picture for inter prediction, H.264 has the ability to use more than one previous reference picture for inter prediction thus enabling the encoder to search for the best match for the current picture from a wider set of reference pictures than just the previously encoded one.

2.8 Structure of H.264 coded video data

The basic coding structure of this standard is similar to that of the previous standards. Video is coded picture by picture. Each picture to be coded is first partitioned into slices. A slice is a sequence of macroblocks or when macroblock adaptive frame field coding (MBAFF) is used, a sequence of macroblock pairs.

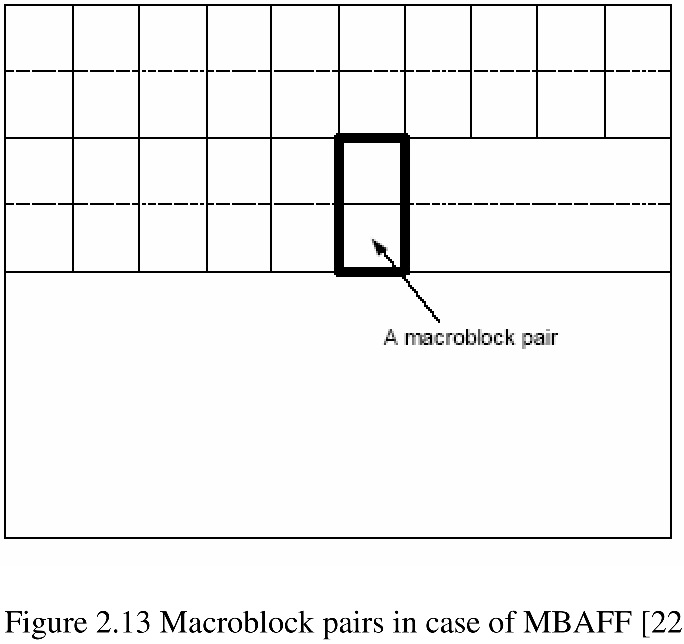


Figure 2.13 Macroblock pairs in case of MBAFF [22].

Slices are individually coded and hence are the coding units. While pictures plus associated data are access units.

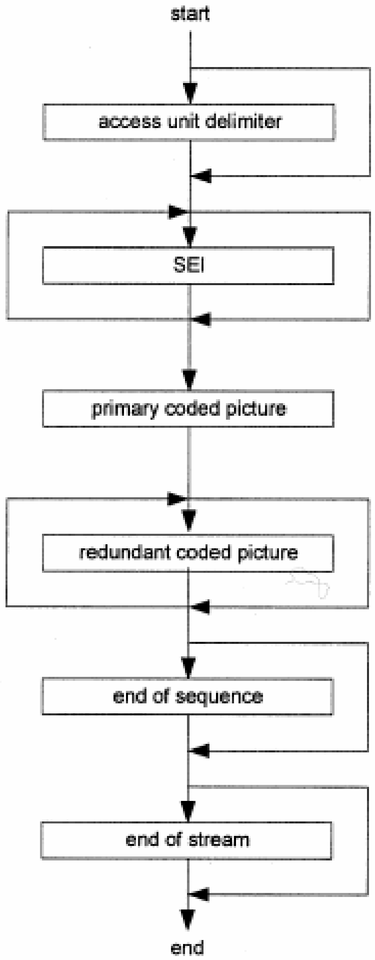


Figure 2.14 Structure of an access unit [10].

There are three basic slice types in H.264/AVC [10][22]: I (Intra), P (Predictive) and B (Bi-directionally predictive). The I slice macroblocks are compressed without using any motion prediction from the slices in other pictures. A special type of picture called instantaneous decoder refresh (IDR) picture is defined in H.264, which contains only I slices and all the pictures following an IDR picture do not use pictures prior to IDR picture as references for motion prediction. IDR pictures can be used for random access or as entry points in the coded sequence. P slices consist of macroblocks that can be compressed by motion prediction. P slices can also have intra macroblocks. P slice when using motion prediction can use one prediction only. Unlike previous standards, the pixels used as reference for prediction can either be in the past or in the future. B slices also consist of macroblocks that can be coded my motion prediction and like P slices can also have intra macroblocks. B slices when using motion prediction can use two predictions. On of the prediction can be in the past and the other in the future in the display order but unlike previous standards it is possible to have both motion predictions from the past or both motion predictions from the future. Also unlike previous standards, B slices can be used as reference for motion prediction by other slices in the past or in the future. Besides I, P and B slices there are also two derived slice types called SI and SP slices. These slices allow switching between multiple coded streams as might be needed by some applications.

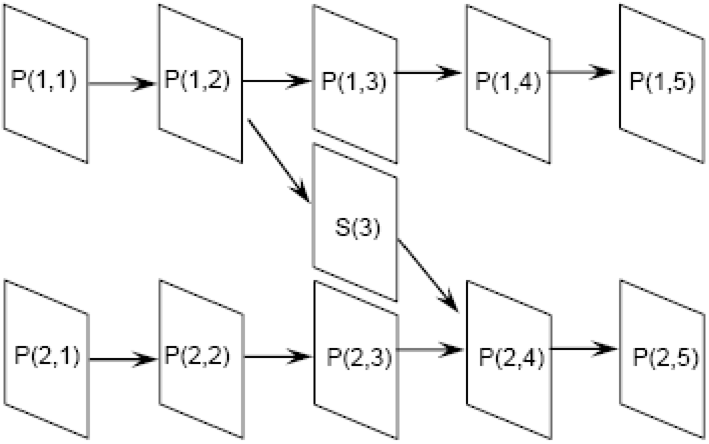


Figure 2.15 Switching using SP picture [4].

2.9 H.264 video decoder

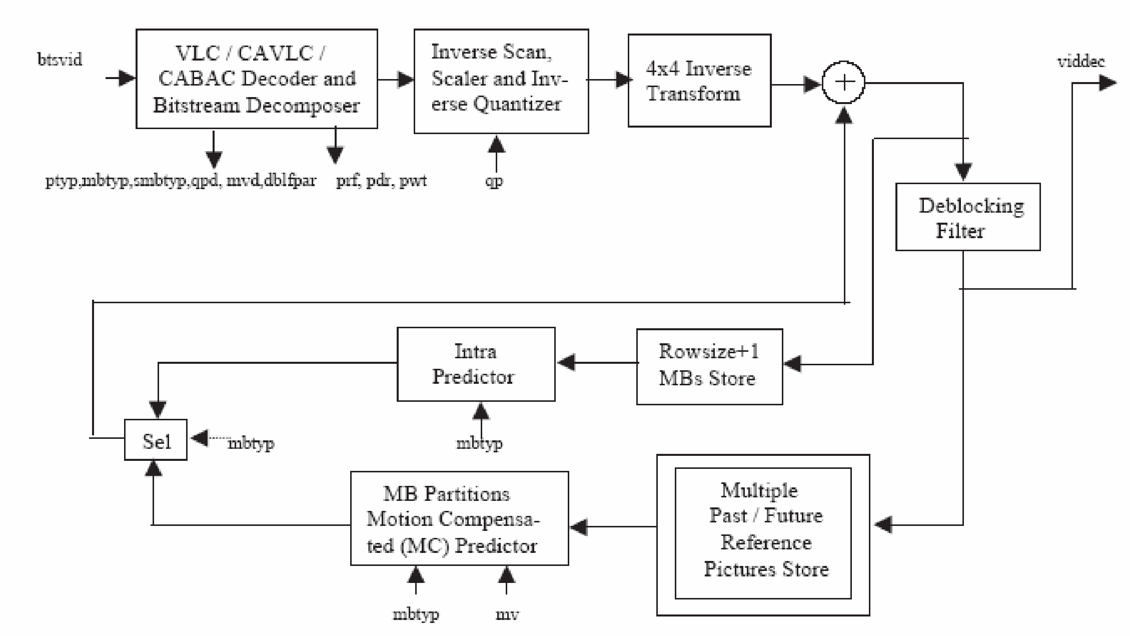

btsvid
VLC / CAVLC / CABAC Decoder and Bitstream Decomposer
Inverse Scan, Scaler and Inverse Quantizer
4x4 Inverse Transform
+
viddec
ptyp,mbtyp,smbtyp,qpd, mvd,dblfpar
prf, pdr, pwt
qp
Deblocking Filter
Intra Predictor
Rowsize+1 MBs Store
Sel
mbtyp
mbtyp
MB Partitions Motion Compensated (MC) Predictor
Multiple Past / Future Reference Pictures Store
mbtyp
mv

Figure 2.16 H.264/AVC decoder block diagram [10].

At the decoder's side, the incoming H.264 bitstream is entropy decoded to produce a set of quantized coefficients which are then inverse quantized and inverse transformation to produce the reconstructed picture. A prediction block is created by the decoder by using the header information decoded from the bitstream and this block is identical to the prediction block formed in the encoder. The predicted block is then added to the reconstructed picture to give the final decoded block.

2.10 Profiles and Levels

A profile in H.264 specifies a subset of the entire bitstream of syntax and limits that shall be supported by all decoders conforming to that profile. There are three profiles in the first version: Baseline, Main, and Extended.

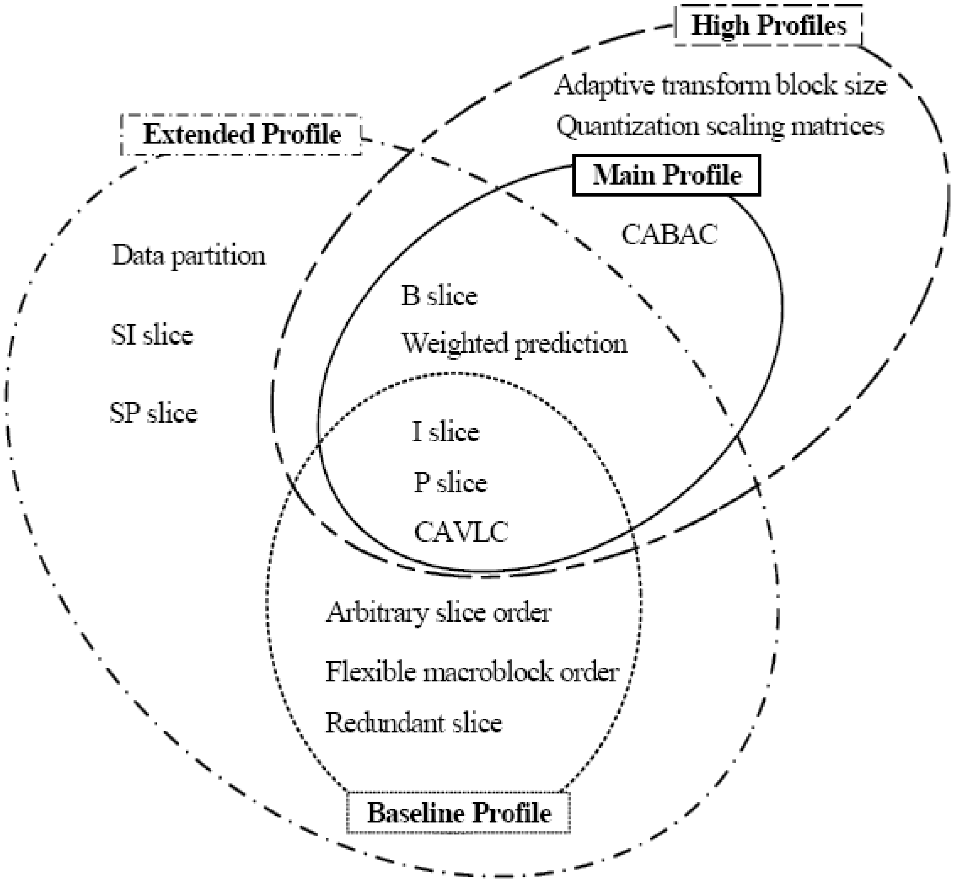


Figure 2.17 Diagram depicting the interconnection between the different profiles in H.264/AVC [4].

2.11 H.264 video bitstream syntax

The coded output bitstream of H.264 has two layers: the network abstraction layer (NAL) and the video coded layer (VCL) [10[[22]. The VCL is specified to efficiently represent the content of the video data and the NAL is specified to format that data and provide header information in a manner appropriate for conveyance across different communication channels or storage media. The NAL can be transmitted in packet oriented formats or byte stream format. The formats for NAL units are similar in both the transport patterns except that each NAL unit can be preceded with a start code prefix and some padding bytes in the byte stream format.

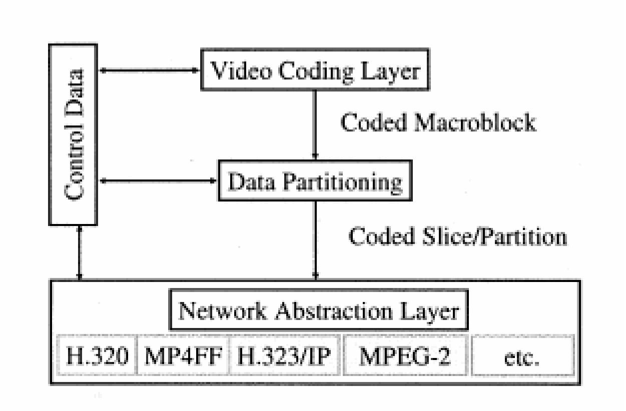


Figure 2.18 Overall structure of H.264 coded video data [10].

The VCL unit contains the core video coded data, which consists of video sequence, picture, slice, and macroblock. The video sequence has either frames or fields which are comprised of three sample arrays, one luma and two chroma sample arrays or the RGB arrays (High 4:4:4 Profile only). Also the standard supports either progressive-scan or interlaced-scan, which may be mixed together in the same sequence. Baseline profile is limited to progressive scan. Pictures are divided into slices. A slice is a sequence of macroblocks and has flexible size, especially one slice within a picture. In case of multiple slice groups, the allocation of macroblocks is determined by a macroblock to slice group map that indicates which slice group each MB belongs to. In the 4:2:0 format, each macroblock is comprised of one 16 x 16 luma and two 8 x 8 chroma sample arrays.

CHAPTER 3

INTRODUCTION TO TRANSCODING

3.1 Transcoding

Video transcoding [1][3] is the process of converting video from one format to another. A format is basically defined by the characteristics such as bit-rate, frame rate, spatial resolution etc. One of the earliest applications of transcoding is to adapt the bit rate of a precompressed bitstream to the available channel bandwidth. Hence transcoding is undertaken to meet the demands of constrained bandwidths and terminal capabilities [1]. Besides these a transcoder can also be used to insert new information for example company's logos, watermarks as well as error resilience features into a compressed video stream. Transcoding techniques are also useful in supporting VCR trick modes such as fast-forward, reverse play etc. for on-demand applications. Transcoding also leads to interoperability between different networks, devices and content representation formats.

Transcoding can be of various types [1]. Some of them are bit rate transcoding to facilitate more efficient transport of video, spatial and temporal resolution reduction transcoding for use in mobile devices with limited display and processing power and error-resilience transcoding in order to achieve higher resilience of the original bit stream to transmission errors.

To achieve optimum results by transcoding, the following criteria have to be fulfilled:

(i) The quality of the transcoded bitstream should be comparable to the one obtained by direct decoding and re-encoding of the output stream.

(ii) The information contained in the input stream should be used as much as possible to avoid multigenerational deterioration.

(iii) The process should be cost efficient, low in complexity and achieve the highest quality possible.

3.2 Transcoding Architectures

This section describes the various transcoding architectures [3]:

(1) Open loop transcoding:

The open loop transcoders include selective transmission where the high frequency DCT coefficients are discarded and requantization. They are computationally efficient, since they operate directly on the DCT coefficients. However they suffer from the drift problem. Drift error occurs due to rounding, quantization loss and clipping functions.

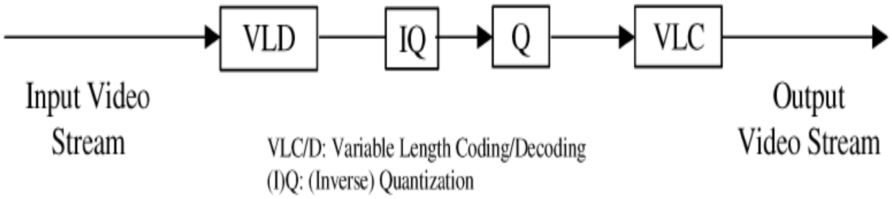

VLD
IQ
Q
VLC
Input Video Stream
Output Video Stream
VLC/D: Variable Length Coding/Decoding
(I)Q: (Inverse) Quantization

Figure 3.1 Open loop transcoding architecture [3].

(2) Cascaded pixel domain transcoding architecture:

This is a drift free architecture. It is a concatenation of a simplified decoder and encoder as shown in Fig.3.2. In this architecture, instead of performing the full motion estimation, the encoder reuses the motion vectors along with other information extracted from the input video bitstream thus reducing the complexity.

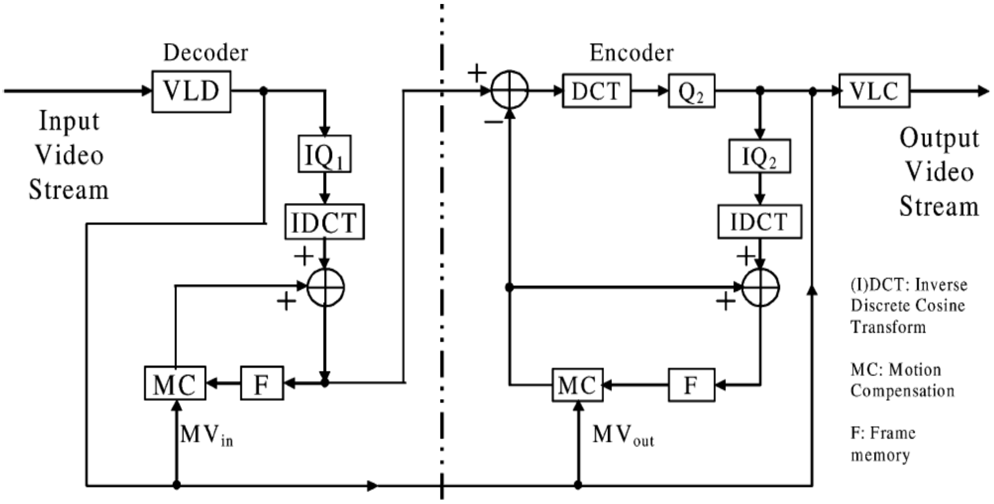


Figure 3.2 Cascaded pixel domain transcoding architecture [3].

(3) Simplified DCT domain transcoding (SDDT):

This architecture is based on the assumption that DCT, IDCT and motion compensation are all linear operations. Since in this architecture, the motion compensation is performed in the DCT domain it is a computationally intensive operation. For instance, as shown in figure 3.4, the goal is to compute the target block B from the four overlapping blocks B1, B2, B3 and B4.

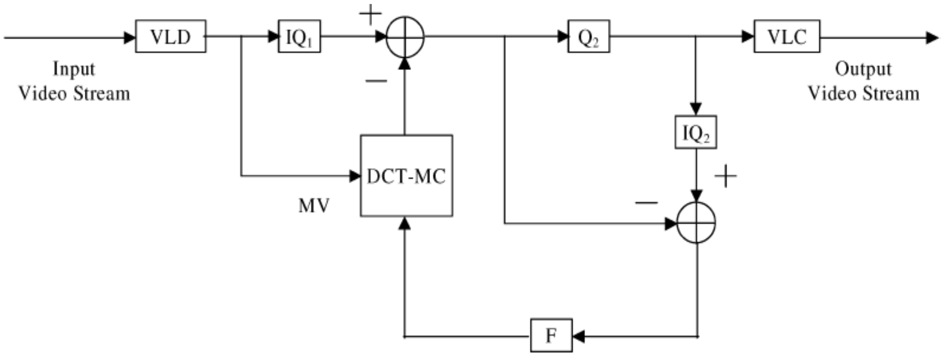


Figure 3.3 Simplified DCT domain transcoding architecture [3].

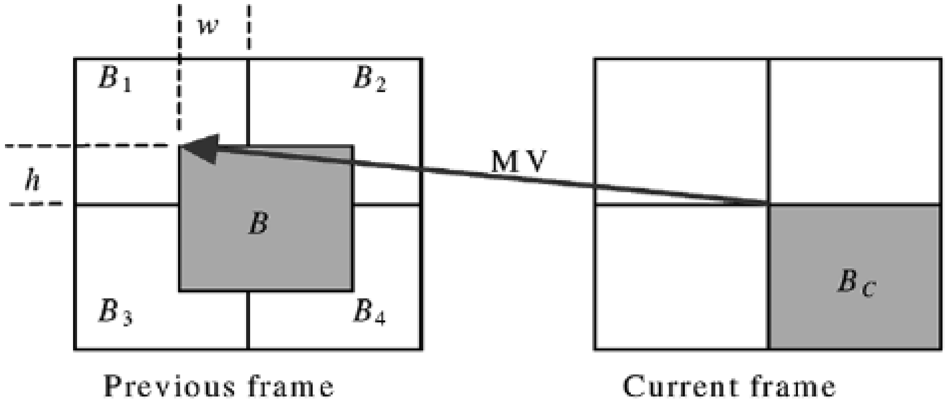


Figure 3.4 DCT- Motion compensation [3].

SDDT eliminates the DCT/IDCT and reduces the frame numbers by half as a result of which it requires less computation and memory as compared to CPDT. However the linearity assumptions are not strictly true since there are clipping functions performed in the video encoder/decoder and rounding operations performed in the interpolation for fractional pixel MC. These failed assumptions may cause drift in the transcoded video.

(4) Cascaded DCT domain transcoding (CDDT):

The cascaded DCT-domain transcoder can be used for spatial and temporal resolution downscaling and other coding parameter changes. Compared to SDDT, greater flexibility is achieved using additional DCT-motion compensation and frame memory resulting in higher cost and complexity. This architecture is adopted for downscaling operations where the encoder side DCT-MC and memory will not cost much.

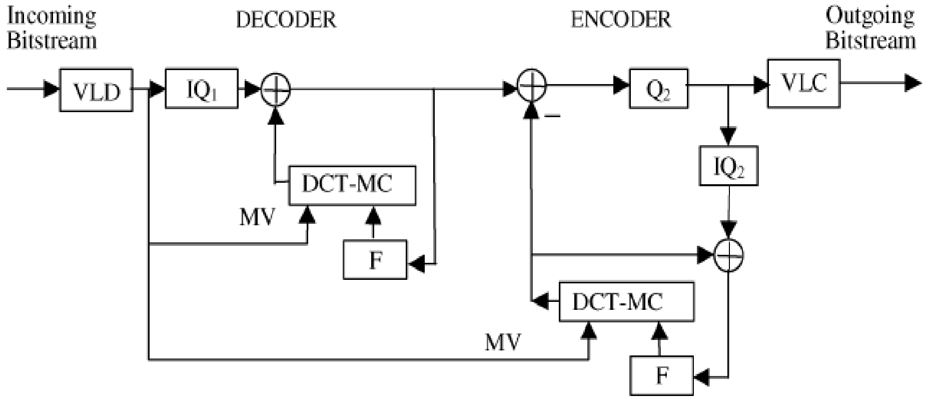


Figure 3.5 Cascaded DCT domain transcoding architecture [3]

3.3 Transcoding of H.264 to MPEG-2

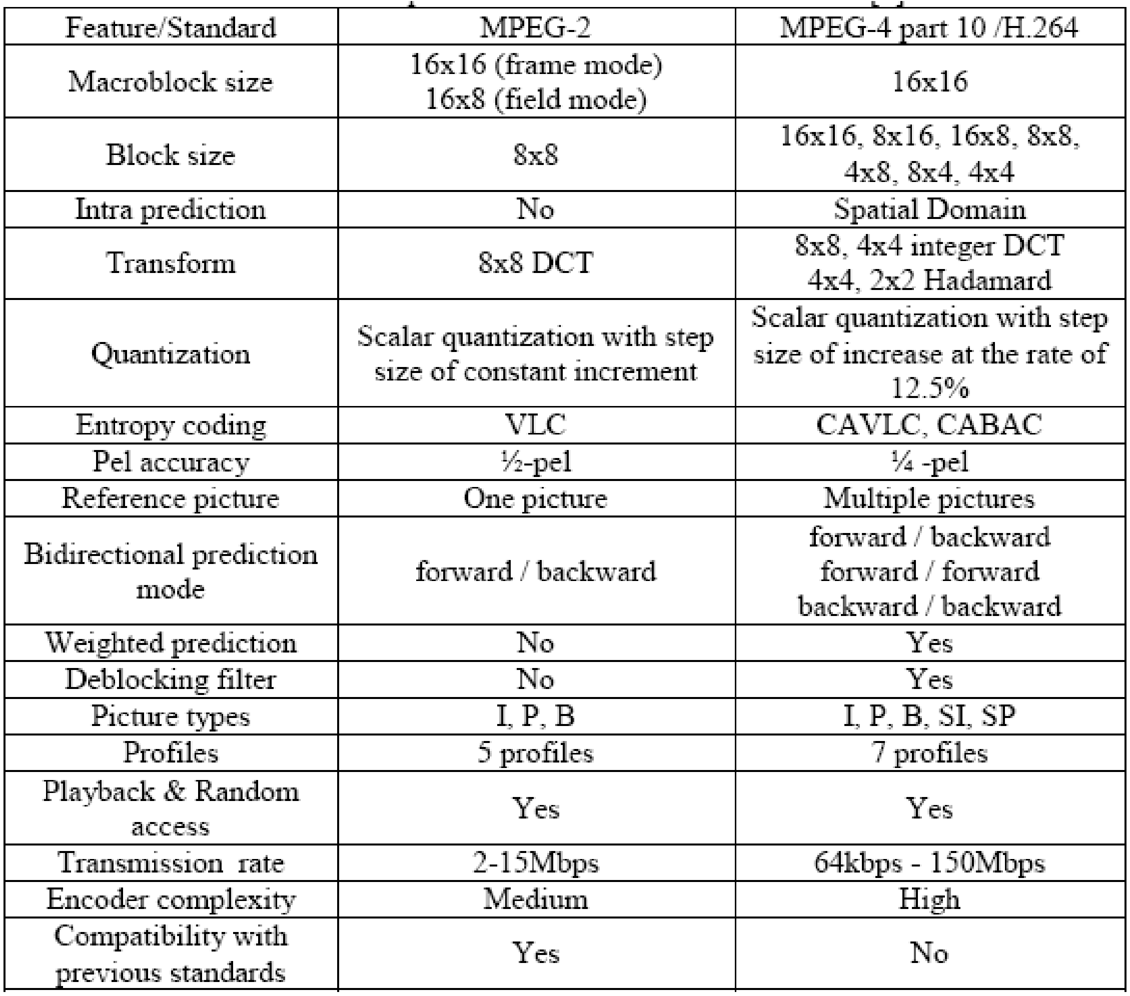
In order to provide better compression of video as compared to previous standards, H.264/AVC video coding standard was recently developed by the JVT (Joint Video Team) consisting of experts from VCEG (Video Coding Experts Group) and MPEG. This new standard fulfills significant coding efficiency, simple syntax specifications, and seamless integration of video coding into all current protocols and multiplex architectures. Thus H.264 can support various applications such as video broadcasting, video streaming, video conferencing over fixed and wireless networks and over different transport protocols. However MPEG-2 has already been widely used in the field of digital broadcasting, HDTV and DVD applications. Hence transcoding is a feasible method to solve the incompatibility problem between H.264 video source and the existing MPEG-2 decoders.

An H.264/AVC to MPEG-2 transcoder is designed to transcode the H.264 video stream to MPEG-2 format so as to be used by the MPEG-2 end equipment. It is better to transmit H.264 bitstreams on public networks to save on the much needed bandwidth and then transcode them into MPEG-2 bitstreams for local MPEG-2 equipment like a set-top box.

3.4 Considerations for H.264 to MPEG-2 Transcoding

For efficient transcoding it is essential to exploit the similarities present in the two standards and also keep in mind the differences between the two standards [6]. As it is known that all standards have various coding tools grouped into various profiles, for my proposed research I am performing transcoding from the baseline profile of H.264/AVC to the simple profile of MPEG-2 at the main level. The basic comparisons between these two standards are enumerated in the Table 3.1.

Table 3.1 Comparison of MPEG-2 and H.264/AVC [4]



Feature/Standard	MPEG-2	MPEG-4 part 10 /H.264
Macroblock size	16x16 (frame mode) 16x8 (field mode)	16x16
Block size	8x8	16x16, 8x16, 16x8, 8x8, 4x8, 8x4, 4x4
Intra prediction	No	Spatial Domain
Transform	8x8 DCT	8x8, 4x4 integer DCT 4x4, 2x2 Hadamard
Quantization	Scalar quantization with step size of constant increment	Scalar quantization with step size of increase at the rate of 12.5%
Entropy coding	VLC	CAVLC, CABAC
Pel accuracy	½-pel	¼ -pel
Reference picture	One picture	Multiple pictures
Bidirectional prediction mode	forward / backward	forward / backward forward / forward backward / backward
Weighted prediction	No	Yes
Deblocking filter	No	Yes
Picture types	I, P, B	I, P, B, SI, SP
Profiles	5 profiles	7 profiles
Playback & Random access	Yes	Yes
Transmission rate	2-15Mbps	64kbps - 150Mbps
Encoder complexity	Medium	High
Compatibility with previous standards	Yes	No

Since I am proposing to perform transcoding from the baseline profile of H.264 to the simple profile at the main level of MPEG-2, the coding tools taken into consideration are enumerated below.

The coding tools present in the baseline profile of H.264 at Level 2 (CIF 352X288) are:

(i) I pictures.

(ii) P pictures

(iii) Context adaptive variable length coding (CAVLC).

(iv) Flexible Macroblock ordering.

(v) Redundant and arbitrary slice order.

(vi) 4:2:0 Chroma format.

On the other hand, the coding tools present in the simple profile of MPEG-2 are as shown below:

(i) I pictures.

(ii) P pictures.

(iii) Conformance point for this profile is at the main level with a broadcast stream picture size of 720X576.

(iv) 4:2:0 Chroma format.

3.5 Choice of transcoding architecture

Open loop transcoders although the simplest and cost efficient are subject to drift and can lead to large quality degradation. On the other hand DCT domain transcoders the DCT domain transcoders are computationally intensive and also less flexible as compared to pixel domain transcoders. However the DCT domain transcoders are significantly faster as compared to pixel domain transcoder but this is offset by the fact that the pixel domain transcoders have better peak signal-to-noise ratio (PSNR) and are more flexible.

The cascaded pixel domain transcoding architecture gives optimum results in terms of complexity, quality and cost. The cascaded pixel domain transcoder offers greater flexibility in the sense that it can be used for bit rate transcoding, spatial/temporal resolution downscaling and for other coding parameter changes as well. Since in the case of standards transcoding it is required to take into consideration the different coding characteristics of MPEG-2 and H.264, flexibility is a key issue.

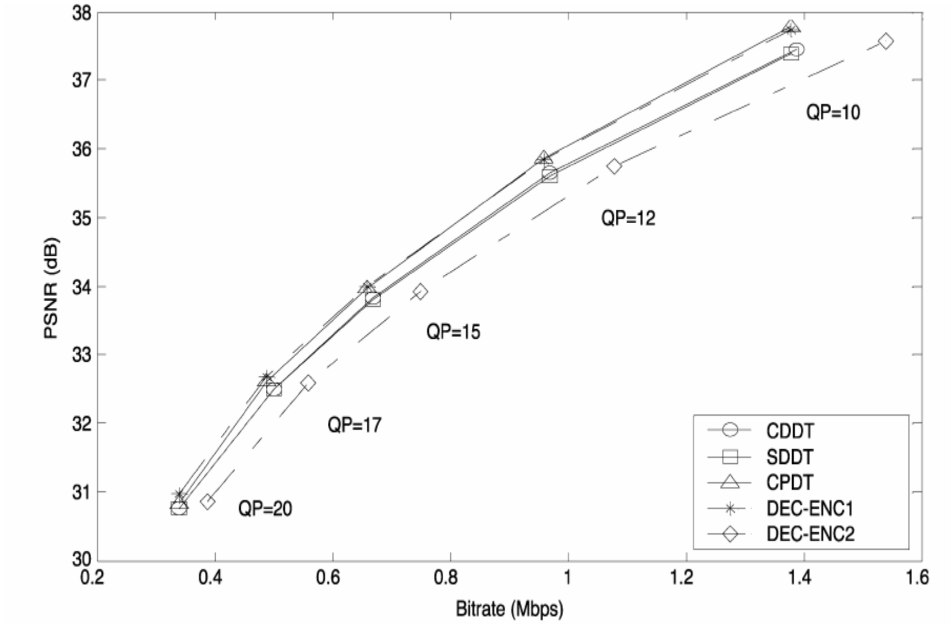


Figure 3.6 PSNR as bit rate plot for the "Foreman" sequence encoded at QP = 7, and then transcoded with different QP's respectively, GOP = (15,1), using the various transcoding architectures shown in Figs.3.1,3.2,3.3 and Fig 3.5. DEC- ENC1 is CPDT using full-search motion reestimation and DEC-ENC2 is CPDT using three-step fast search motion reestimation [3].

From Fig 3.6 it is clear that the cascaded pixel domain architecture outperforms the DCT domain architecture. Moreover for larger GOP sizes the drift in DCT domain transcoders becomes more significant leading to high performance degradation.

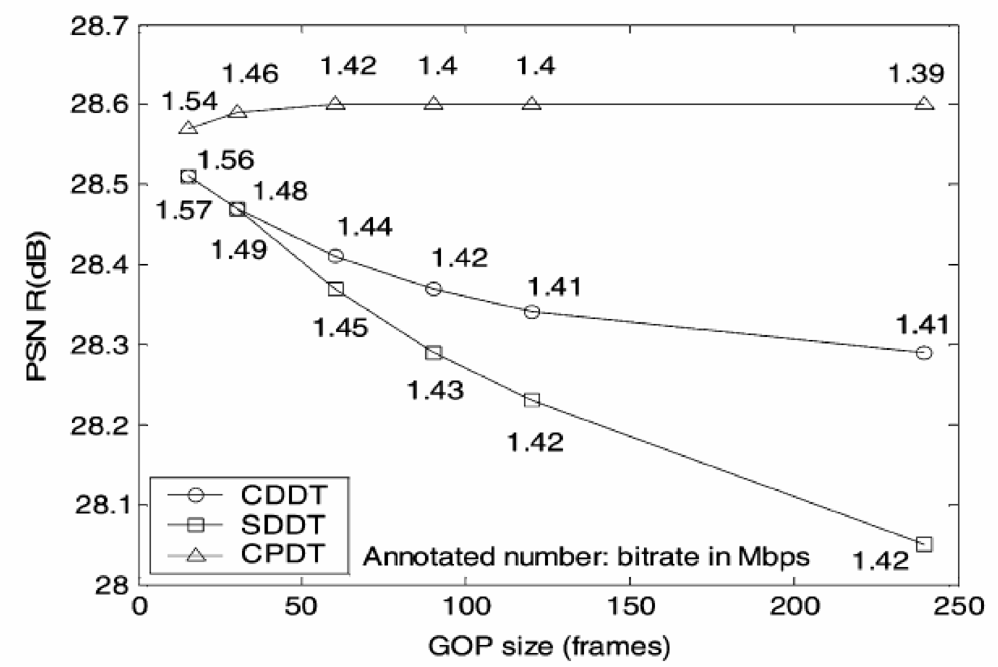


Figure 3.7 Performance comparison of average PSNR for CPDT, SDDT, and CDDT for different GOP sizes. The “Mobile & Calendar” sequence is encoded at QP = 5 and transcoded at QP = 11 [3].

Based on the comparison of the various transcoding architectures, it can be inferred that pixel domain transcoding with motion vector reuse offers the best performance in terms of PSNR, execution time and resistance to drift.

CHAPTER 4

INTRA FRAME TRANSCODING

4.1 Introduction

Before going into the details of the process involved in intra frame transcoding of H.264 to MPEG-2, it is important to understand the intra frame coding process in both the standards.

4.2 Intra Frame Coding in H.264

Intra frame coding or intra prediction is done in the pixel domain in H.264. In intra frame coding, a prediction block is formed based on the neighboring samples of the previously encoded and reconstructed blocks and this predicted block is subtracted from the current block to get the error residual which is then encoded. The H.264 standard supports 9 prediction modes for each 4x4 sub-block and 4 prediction modes for each 16x16 luma block. For the chroma blocks there are 4 predictions modes which are similar to the ones for 16x16 luma blocks. Both the chroma blocks (Cb and Cr) use the same prediction modes.

4.2.1. Intra prediction modes

Spatial prediction in H.264 is performed in three different ways Intra 4x4, Intra 16x16 and I_PCM. The use of 4x4 blocks means more accuracy and detail however at the cost of high overhead for conveying the sub-block mode decisions in the bitstream. The 16x16 block size on the other hand offers a better overall bit rate [4][13].

Intra 4x4 Luma prediction:

Each luma macroblock is divided into 4x4 sub-blocks. Each sub-block can be predicted using any of the 9 prediction modes as shown in the figure below.

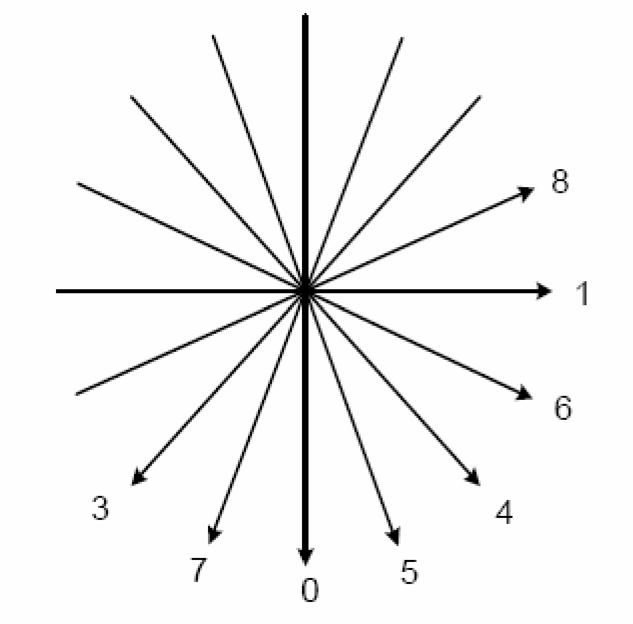


Figure 4.1 The nine directional prediction modes for Luma 4x4 [22].

Table 4.1 Specification of Intra4x4PredMode [luma4x4BlkIdx] and associated names [22]

Intra4x4PredMode[luma4x4Blkldx]	Name of Intra4x4PredMode[luma4x4Blkldx]
0	Intra_4x4_Vertical (prediction mode)
1	Intra_4x4_Horizontal (prediction mode)
2	Intra_4x4_DC (prediction mode)
3	Intra_4x4_Diagonal_Down_Left (prediction mode)
4	Intra_4x4_Diagonal_Down_Right (prediction mode)
5	Intra_4x4_Vertical_Right (prediction mode)
6	Intra_4x4_Horizontal_Down (prediction mode)
7	Intra_4x4_Vertical_Left (prediction mode)
8	Intra_4x4_Horizontal_Up (prediction mode)

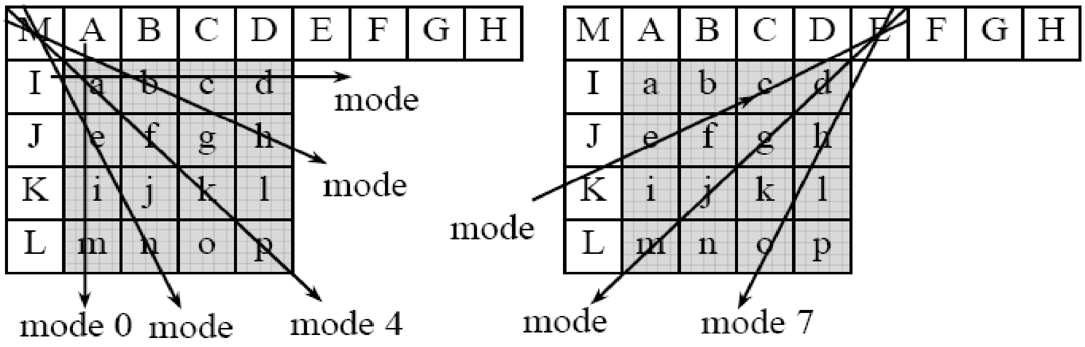


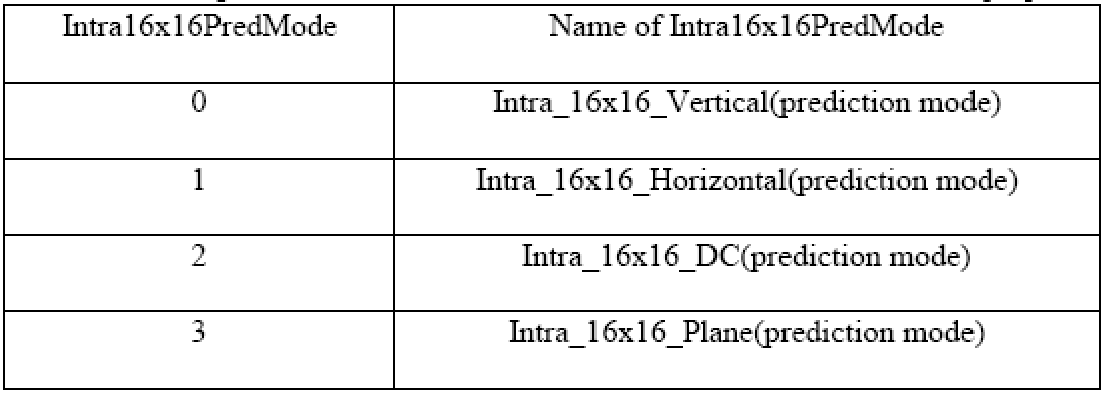
Figure 4.2 Intra 4 x 4 prediction mode directions (vertical : 0, horizontal : 1, DC : 2, diagonal down left : 3, diagonal down right : 4, vertical right : 5, horizontal down : 6, vertical left : 7, horizontal up : 8) [4].

To demonstrate how intra frame coding for 4x4 sub-blocks takes place in H.264, let us refer to the figure above. In the figure a, b, c….p are pixels of the current block and A, B, C…..M are the pixels of the previously decoded block, using vertical prediction the pixels a, e, i and m are predicted from the pixel A and similarly the pixel B is used to predict the pixels b, f, j, and n and so on.

Intra 16x16 Luma prediction:

Under this kind of prediction, each 16x16 macroblock can be predicted using four prediction modes as follows:

Table 4.2 Specification of Intra16x16PredMode and associated names [22]



Intra16x16PredMode	Name of Intra16x16PredMode
0	Intra_16x16_Vertical(prediction mode)
1	Intra_16x16_Horizontal(prediction mode)
2	Intra_16x16_DC(prediction mode)
3	Intra_16x16_Plane(prediction mode)

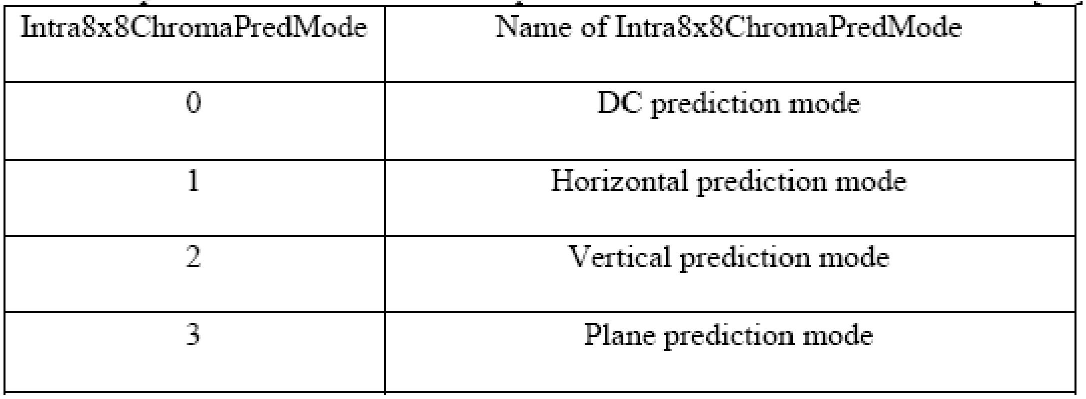
I_PCM:

Under this type, the encoder transmits 16x16 macroblocks containing raw uncompressed pixel data for the luma and chroma planes [13].

8x8 Chroma prediction:

Each 8x8 block of Chroma for both Cb and Cr are predicted from previously encoded samples above and/or to the left using four prediction modes which are similar to the prediction modes used for Intra 16x16 Luma prediction except that the numbering of the modes is different. The four prediction modes for 8x8 Chroma is as shown below:

Table 4.3 Specification of Intra Chroma prediction modes and associated names [22]



Intra8x8ChromaPredMode	Name of Intra8x8ChromaPredMode
0	DC prediction mode
1	Horizontal prediction mode
2	Vertical prediction mode
3	Plane prediction mode

4.2.2 Mode decisions

The method of mode decisions [22] in H.264 determines the coding mode for each macroblock. Mode decision for high efficiency may use rate/distortion optimization but this method can be quite complex. The outcome for mode decision is the best-selected coding mode for a macroblock.

4.3 Intra Frame Coding in MPEG-2

In MPEG-2 a simple intra prediction is performed on the DC coefficients and so it is done in the transform domain unlike in H.264 where intra prediction is done in the pixel domain. If the vale of picture coding type in the bitstream is 001 then the coded picture is intra picture.

In intra frame coding of MPEG-2, each macroblock is divided into non-overlapping 8x8 blocks. A 2D DCT [11] is then applied to each of the blocks to give the 8x8 DCT coefficients. DCT works on the concept of energy compaction that is most of the signal information is concentrated on low-frequency components. Moreover DCT is more efficient for highly correlated sources and hence gives a better compression of I frames then compared to P or B frames.

$$F(u,v) = \frac{2}{N} C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\frac{(2x+1)u\pi}{2N} \cos\frac{(2y+1)v\pi}{2N} \tag{4.1}$$

with u, v, x, y = 0, 1, 2, …. N-1

where x, y are spatial coordinates in the sample domain

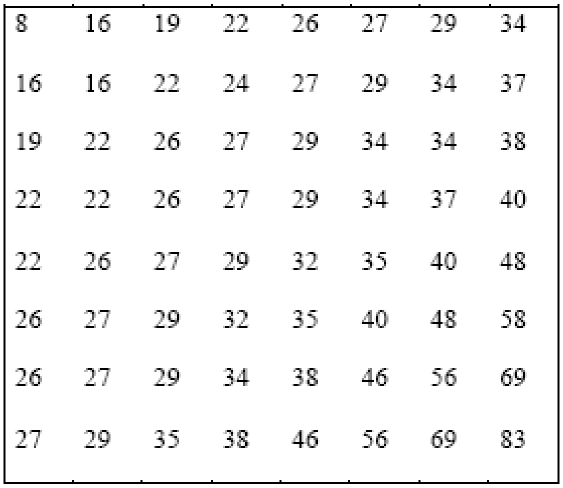
u, v are coordinates in the transform domain

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases}$$

The inverse DCT (IDCT) is defined as:

$$f(x,y)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)F(u,v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \qquad (4.2)$$

After applying the 2D 8x8 DCT across each macroblock it is observed that DCT does not directly reduce the number of coefficients instead for every 8 bit coefficient of 8x8 block it produces and 8x8 block of 11 bit coefficients. The compression however is achieved by the fact that the transform concentrates most of the energy in the low frequency components and many of the high frequency components are near zero. Most of the near zero components are then discarded and the remaining components are then quantized and entropy coded.

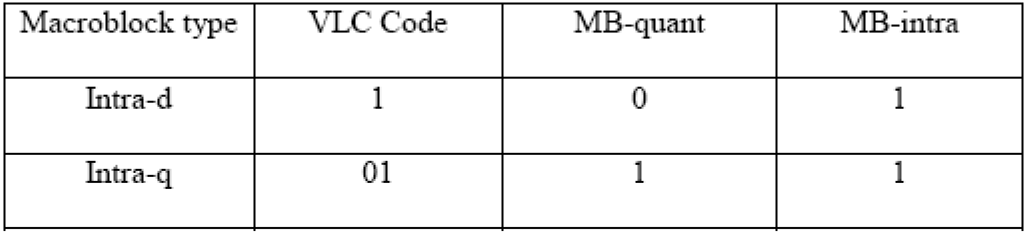


8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Figure 4.3 Default quantization matrix for Intra DCT coefficients [19].

After applying the 2D DCT across each macroblock the output will be six blocks of 8x8 DCT coefficients; four for 8x8 luma blocks and 2 for 8x8 chroma (Cb and Cr) blocks for the 4:2:0 format. The DCT coefficients are then quantized using either the default quantization matrix (intra-d) which is indicated in the sequence and slice layer or a modified matrix (intra-q) which is defined by the buffer status. The quantizer step size is derived from the quantization matrix and the quantizer scale. If the macroblock type is intra-q, then the MB header contains a five bit integer that defines the quantizer scale. The scale can be changed from 1 to 31, both inclusive. If the macroblock type is intra-d, no quantizer scale is transmitted and the decoder uses the previously set value. The quantized DCT coefficients are coded losslessly by a DPCM technique. The AC coefficients are coded using run/level coding technique. These quantized coefficients are then scanned using either a zig-zag scan method or alternate scan method creating a one dimensional array of coefficients. They are then variable length coded.

Table 4.4 Macroblock type VLC for I pictures [7]



Macroblock type	VLC Code	MB-quant	MB-intra
Intra-d	1	0	1
Intra-q	01	1	1

$$\begin{array}{cc|cccccccc}
 & & & & & & & & & u \\
 & & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
 & 0 & 0 & 1 & 5 & 6 & 14 & 15 & 27 & 28 \\
 & 1 & 2 & 4 & 7 & 13 & 16 & 26 & 29 & 42 \\
 & 2 & 3 & 8 & 12 & 17 & 25 & 30 & 41 & 43 \\
 & 3 & 9 & 11 & 18 & 24 & 31 & 40 & 44 & 53 \\
 & 4 & 10 & 19 & 23 & 32 & 39 & 45 & 52 & 54 \\
 & 5 & 20 & 22 & 33 & 38 & 46 & 51 & 55 & 60 \\
 & 6 & 21 & 34 & 37 & 47 & 50 & 56 & 59 & 61 \\
v & 7 & 35 & 36 & 48 & 49 & 57 & 58 & 62 & 63
\end{array}$$

(a)

$$\begin{array}{cc|cccccccc}
 & & & & & & & & & u \\
 & & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
 & 0 & 0 & 4 & 6 & 20 & 22 & 36 & 38 & 52 \\
 & 1 & 1 & 5 & 7 & 21 & 23 & 37 & 39 & 53 \\
 & 2 & 2 & 8 & 19 & 24 & 34 & 40 & 50 & 54 \\
 & 3 & 3 & 9 & 18 & 25 & 35 & 41 & 51 & 55 \\
 & 4 & 10 & 17 & 26 & 30 & 42 & 46 & 56 & 60 \\
 & 5 & 11 & 16 & 27 & 31 & 43 & 47 & 57 & 61 \\
 & 6 & 12 & 15 & 28 & 32 & 44 & 48 & 58 & 62 \\
v & 7 & 13 & 14 & 29 & 33 & 45 & 49 & 59 & 63
\end{array}$$

(b)

Figure 4.4 Scan matrices in MPEG-2: (a) Zig-zag scan (b) Alternate scan [19].

4.4 Transcoding of an I frame

The general block diagram of a transcoder that would implement I frame decoding in H.264 followed by encoding in MPEG-2 is as shown below:

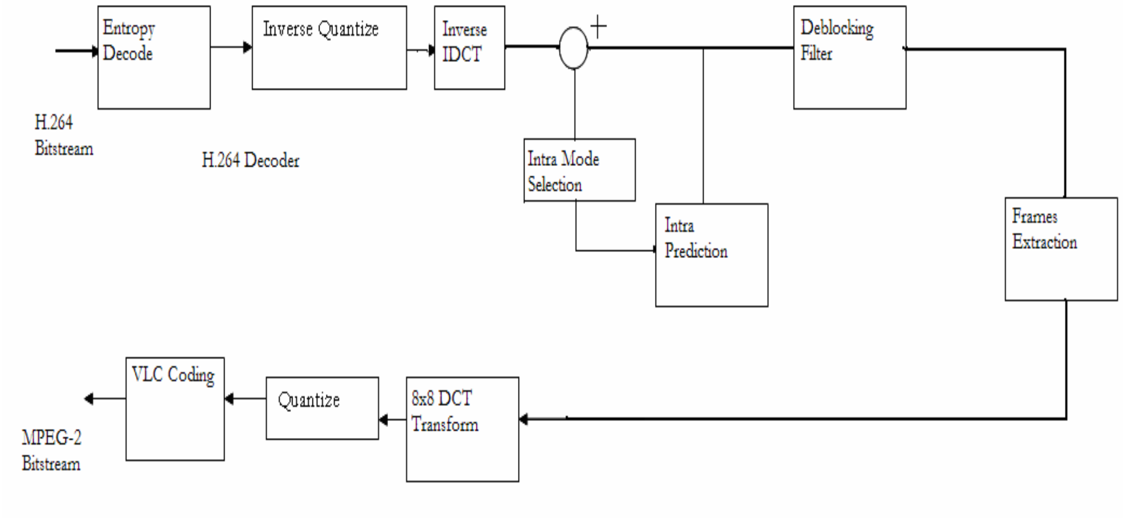


Figure 4.5 Basic block diagram for I frame transcoding.

Since Mpeg 2 does not have mode decision based intra prediction, the reconstructed frame coming out of the H.264 decoder is directly passed into an Mpeg 2 encoder where 8x8 DCT is applied across the frame and then quantized and VLC coded to give an Mpeg 2 encoded output. The PSNR of the final decoded output is then compared with the PSNR of the output obtained by direct encoding and decoding a sequence using MPEG-2 encoder and decoder. The results are plotted.

In software, the Mpeg 2 encoding phase is as shown below:

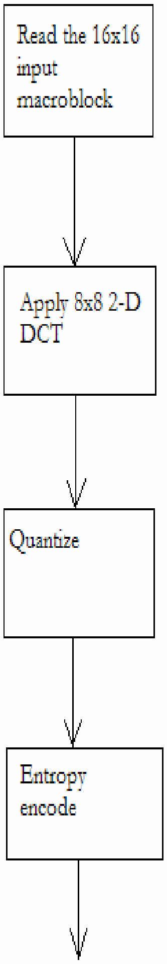
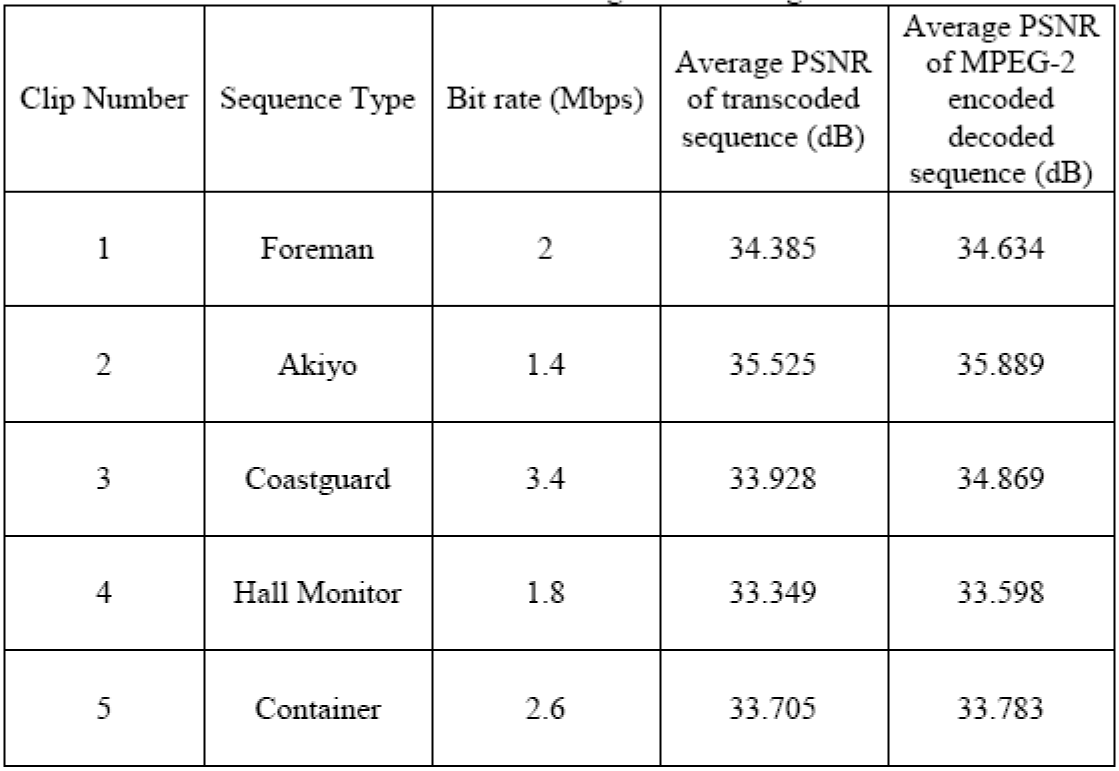


Figure 4.6 Intra coding in Mpeg 2 [7].

4.5 Results

As mentioned earlier the transcoding of I frames is done by using the brute force method that is decoding the H.264 bitstream and then encoding it in the MPEG-2 format. Since the MPEG-2 standard does not have mode decision based intra coding, the input macroblock after being decoded by the H.264 decoder is passed through the MPEG-2 encoder where it is divided into four 8x8 blocks and a 2D 8x8 DCT is applied across it followed by quantization and VLC coding.

Table 4.5 Comparison of PSNR obtained by transcoding the bitstream and MPEG-2 encoding and decoding it.



Clip Number	Sequence Type	Bit rate (Mbps)	Average PSNR of transcoded sequence (dB)	Average PSNR of MPEG-2 encoded decoded sequence (dB)
1	Foreman	2	34.385	34.634
2	Akiyo	1.4	35.525	35.889
3	Coastguard	3.4	33.928	34.869
4	Hall Monitor	1.8	33.349	33.598
5	Container	2.6	33.705	33.783

Table 4.5 lists the results obtained by transcoding the I frames when the test sequences were transcoded by using the brute force method of decoding in H.264 and

then re encoding in MPEG-2. The PSNR obtained by this method is then compared with the PSNR obtained by encoding and decoding the test sequences in the MPEG-2 format. As shown in Table 4.5, the PSNR of the transcoded bitstream is comparable to the PSNR obtained by decoding and encoding of the bitstream in MPEG-2 format.

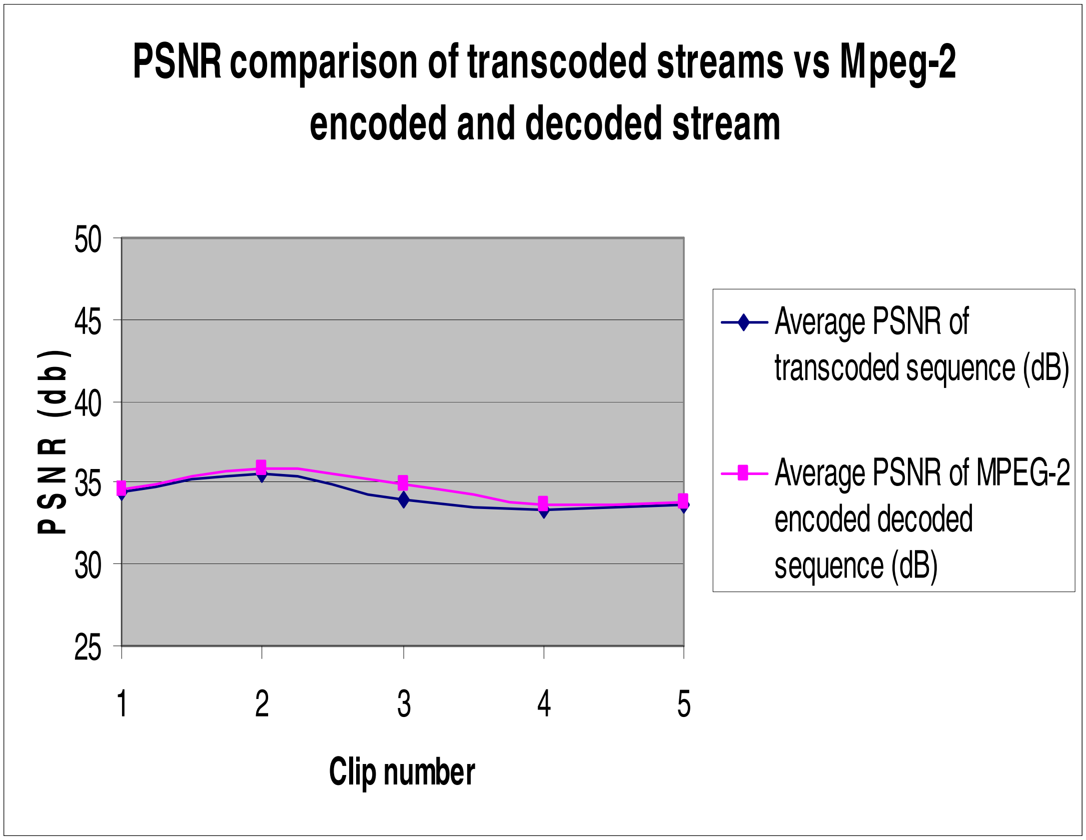


Figure 4.7 Comparison of the PSNR of the proposed method and encoding and Decoding in MPEG-2

Figure 4.7 shows the plot between the PSNR values of the transcoded sequence and the MPEG-2 encoded decoded sequence. As can be seen, the PSNR values of both these cases are very close and comparable.

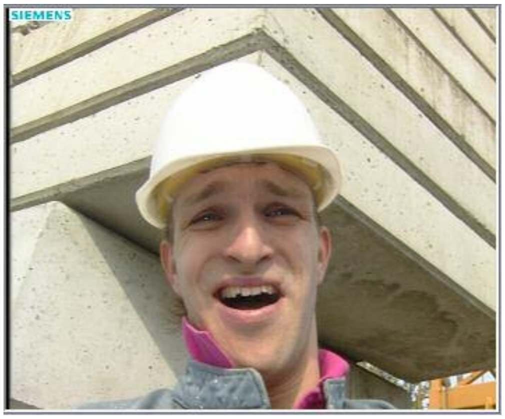


Fig 4.8 Subjective quality of original Foreman clip.

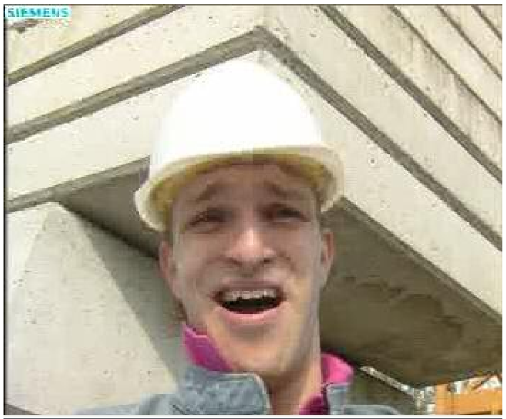


Fig 4.9 Subjective quality of an I frame of the clip foreman in 2Mbps MPEG-2 compressed stream.

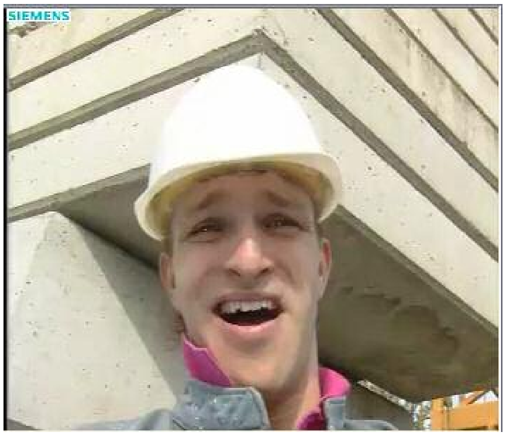


Fig 4.10 Subjective quality of an I frame of the clip Foreman in 2 Mbps MPEG-2 transcoded stream.

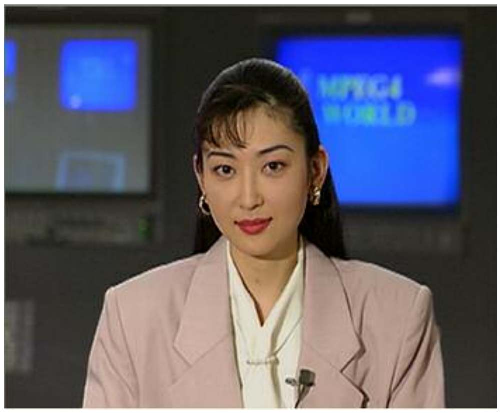


Fig 4.11 Subjective quality of original Akiyo clip.

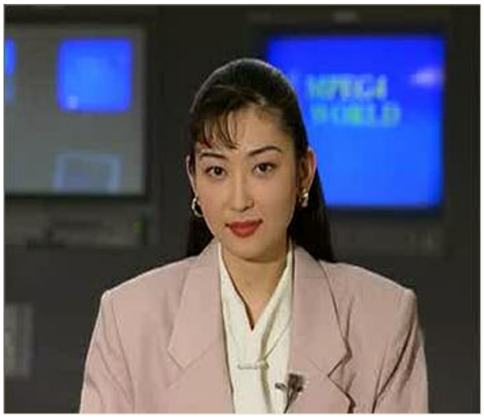


Fig 4.12 Subjective quality of an I frame of the clip Akiyo in 1.4 Mbps MPEG-2 compressed stream

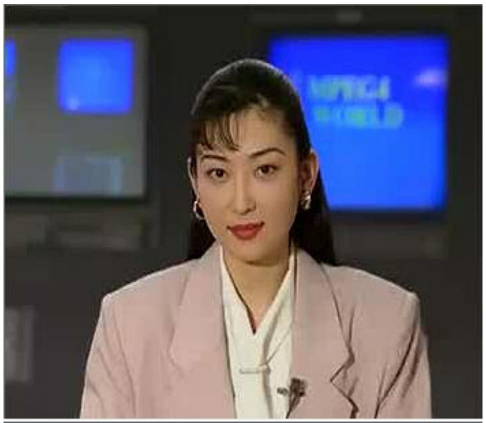


Fig 4.13 Subjective quality of an I frame of the clip Akiyo in 1.4 Mbps MPEG-2 transcoded stream.

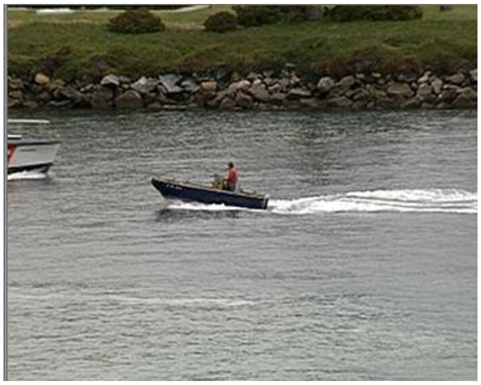


Fig 4.14 Subjective quality of original Coastguard clip.

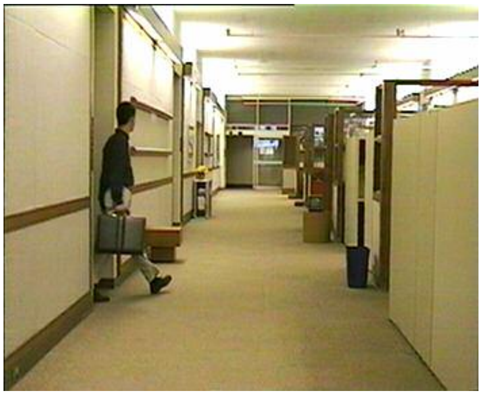


Fig 4.17 Subjective quality of original Hall Monitor clip.

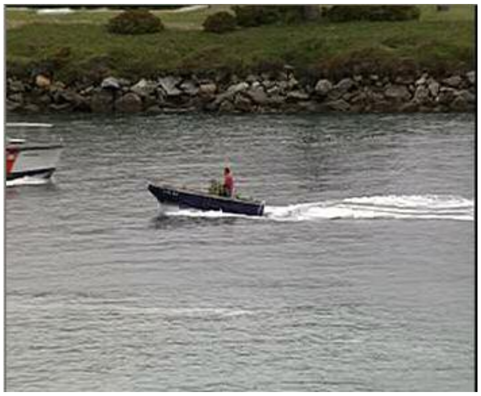


Fig 4.15 Subjective quality of an I frame of the clip Coastguard in a 3.4 Mbps compressed MPEG-2 stream.

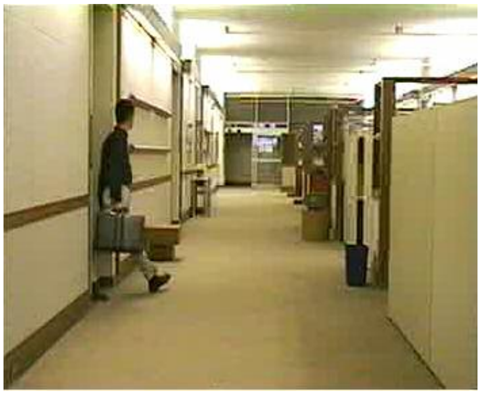


Fig 4.18 Subjective quality of an I frame of the clip Hall Monitor in a 1.8Mbps compressed MPEG-2 stream

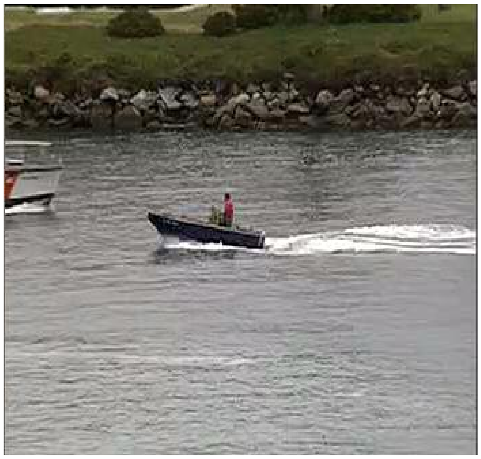


Fig 4.16 Subjective quality of an I frame of the clip Coastguard in a 3.4 Mbps transcoded MPEG-2 stream.

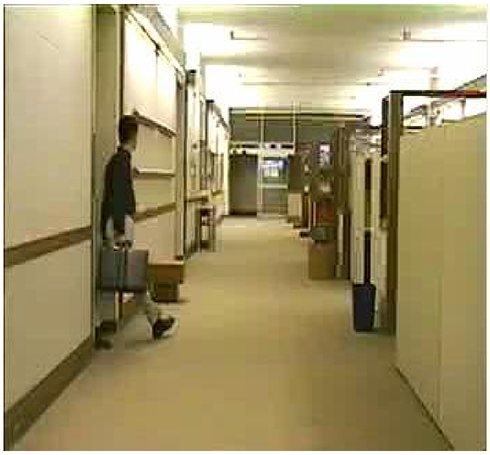


Fig 4.19 Subjective quality of an I frame of the clip Hall Monitor in a 1.8Mbps transcoded MPEG-2 stream

CHAPTER 5

INTER FRAME TRANSCODING

5.1 Introduction

Before moving into the process of transcoding on P frames from H.264 to MPEG-2 it is important to understand the process of predictive coding in H.264 and MPEG-2.

5.2 P frame coding in H.264

P frame coding in H.264 includes both motion estimation (ME) [13][22] and motion compensation (MC) [13][22]. The ME/MC process performs prediction. It generated a predicted version of a rectangular array of pixels, by choosing another similarly sized array of pixels from a previously decoded reference picture and translating the reference array to the position of the current rectangular array. In H.264/AVC, the rectangular array of pixels that are predicted using MC can have the following sizes: 4x4, 4x8, 8x4, 8x8, 16x8, 8x16, and 16x16 pixels. Hence AVC allows sub 16x16 and sub 8x8 ME and MC unlike the MPEG-2 standard. Also, AVC allows motion estimation up to quarter pixel accuracy unlike MPEG-2 which allows motion estimation to half pixel accuracy. AVC also allows the use of more than one recently decoded reference frames from the reference frame buffer. It also supports the use of B frames as reference frames but this feature is optional. These features such as multi reference prediction, quarter pixel accuracy ME and use of B frames (which is optional) as reference frames is not supported by the MPEG-2 standard.

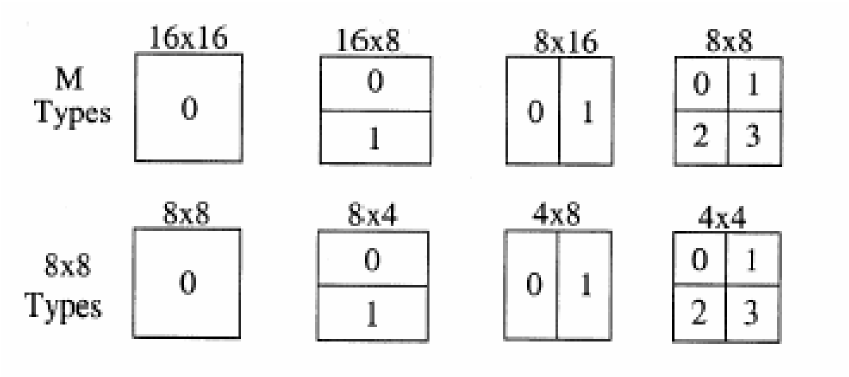


Figure 5.1 Segmentation of macroblocks for motion compensation in H.264 [22].

The selection of the size of inter prediction partitions is a result of a trade-off between the coding gain provided by using motion compensation with smaller blocks and the quantity of data needed to represent the data for motion compensation.

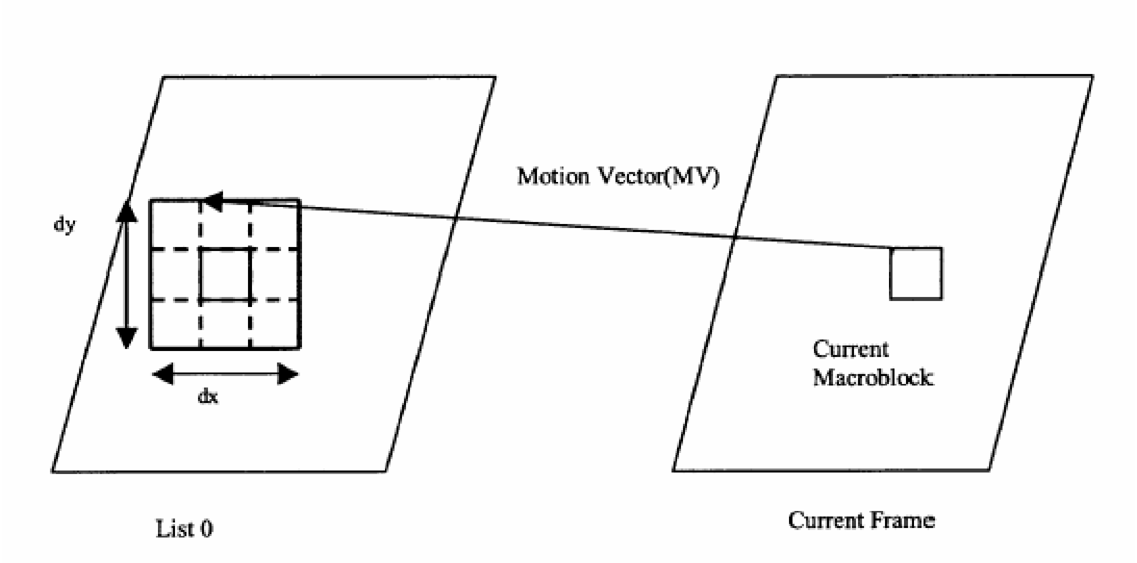


Figure 5.2 Process of P frame prediction for the current macroblock over a search window dx x dy in a reference frame of List0 [21].

Search window width, dx = 2*swx + 1

Search window height, dy = 2*swy + 1

AVC maintains two lists of reference frames, list0 and list1. P frames predict only from past decoded reference frames in List0 i.e. it uses forward prediction. For each pixel of each macroblock, it refers to each reference frame in list0 as shown in Fig.5.7 and it computes the sum of absolute difference (SAD) value at each pixel within a user defined search window (swx – x axis search window size, swy – y axis search window size) in the reference frame. It selects the pixel in the reference frame which has the minimum SAD value as a search center. It then uses this search center to perform half pixel resolution ME. Again the pixel or sub pixel with the least SAD value is selected as the search center and quarter pixel ME is performed. This process is repeated for each type of macroblock and sub macroblock partition, and each reference frame in List0 and then the best match with the minimum SAD value is selected. Depending upon the best mode or sub partition selected, the macroblock can have one motion vector or up to 16 motion vectors. This full search method achieves the best result but it is very computationally intensive.

5.3 P frame coding in MPEG-2

In the MPEG-2 standard, macroblocks in P frames can be coded using inter modes or intra modes. Macroblocks that are inter coded seek to exploit temporal correlation among frames and hence achieve compression. In P pictures, some modes available are MC/ no MC, coded/not coded, intra/inter and quantizer modification or not. The standard itself does not specify how to make these decisions [7].

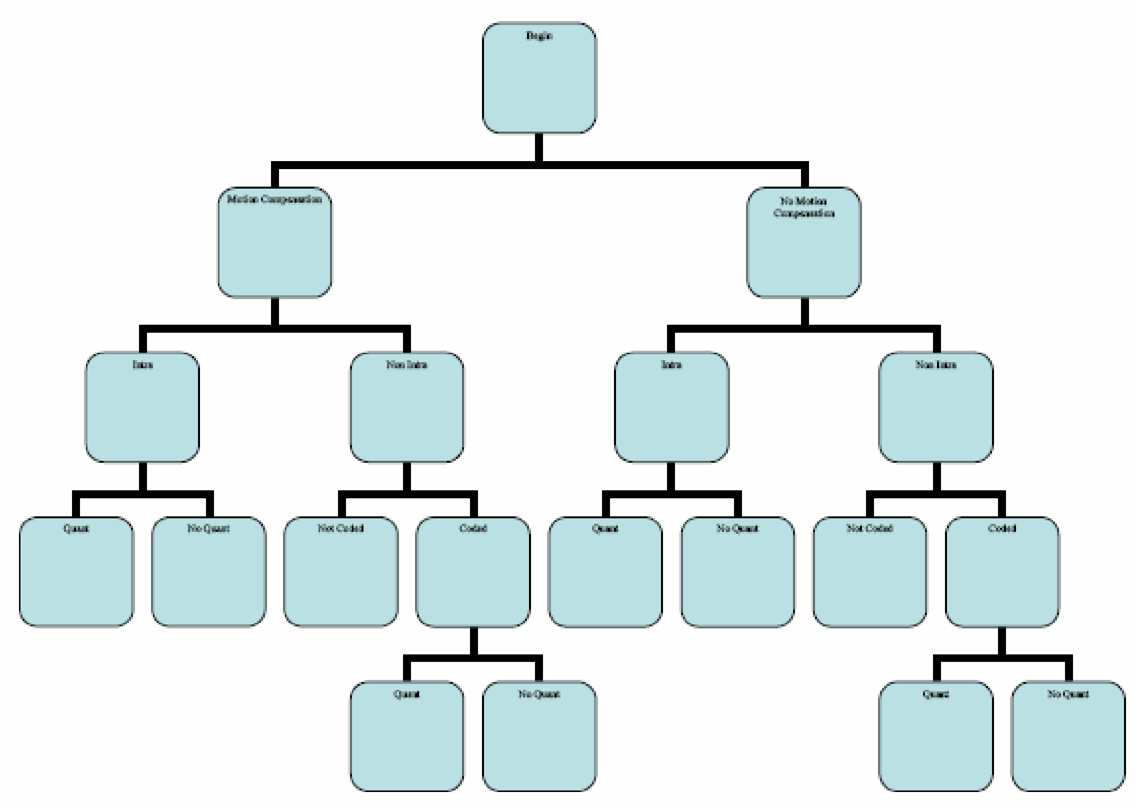


Figure 5.3 Macroblock mode selection process for P frames in MPEG-2 [7].

The MC/no MC decision is made by the encoder. The encoder can decide to transmit a motion vector or not. If the motion vector is zero due to MC decision, then some bits can be saved by not transmitting it. The intra/inter decision is based on the comparison of signal variance and difference signal variance. The coded/not coded decision is a result of quantization. If all the quantized DCT coefficients are zero then the block need not be coded. The quant/no quant decision is made as to whether the quantizer scale is to be changed or not. This is usually based on the frame content and on the buffer fullness assessment of the decoder. The decision process for each macroblock is described in figure 5.1.

5.3.1 Motion Estimation and Compensation

In the motion estimation process, motion vectors for predicted and interpolated pictures are coded differentially between macroblocks. The two motion vector components, the horizontal one first followed by the vertical one are coded independently.

The motion compensated prediction forms predictions from previously decoded pictures, using the motion vectors that are of integer or half pel resolution. For the half pel resolution, decoded motion vectors are linearly interpolated in the previous or future picture. In the case of intra macroblock, no prediction is formed. They may however carry motion vectors know as concealment motion vectors which are used to aid the decoding process when bit stream errors have occurred.

The differential motion vectors are coded using VLC. The motion vectors for the color components are obtained from the motion vectors of the luminance components as shown in equations 5.1 – 5.4

For 4:2:0;

MV (horizontal chroma) = MV (horizontal luma) / 2 (5.1)

MV (vertical chroma) = MV (vertical luma) / 2 (5.2)

For 4:2:2;

MV (horizontal chroma) = MV (horizontal luma) / 2 (5.3)

MV (vertical chroma) = MV (vertical luma) (5.4)

There are four prediction modes in MPEG-2 since it also provides field based processing unlike MPEG-1. The four modes are known as [7]:

1) Field prediction: A frame is composed of two fields, predictions are made independently for each field by using data from one or more previously decoded fields. Field prediction can be made between the same parity fields or opposite parity field. In P-pictures the two reference fields from which predictions shall be made are the most recently decoded reference top field and the most recently decoded reference bottom field. The simplest case is shown in figure 5.2. It is used when predicting the first picture of a coded frame or when using field prediction within a frame picture. In this case the two reference fields are part of the same reconstructed frame.

The case when predicting the second field picture of a coded frame is more complicated because the two most recently decoded reference fields shall be used. In this case the most recent reference field was obtained from decoding the first field picture of the coded frame. Figure 5.3 illustrates this situation when the second picture is the bottom field. Figure 5.3 illustrates the situation when the second picture is the top field.

2) Frame prediction: Predictions are made from one or more previously decoded frames. Even in a frame picture, field predictions can be selected on an MB by MB basis.

3) (16x8) Motion compensation: In this case, two motion vectors are used per macroblock. The first motion vector is used for upper 16x8 region and the second motion vector is used for the lower 16x8 region. This method can be used by field pictures only.

4) Dual prime prediction: This is present in P pictures only. Only one motion vector is encoded in the bit stream together with a small differential motion vector. In the case of field pictures two motion vectors are derived from this information. They are used to form predictions from two reference fields (one top field, one bottom field), which are averaged to form the final prediction.

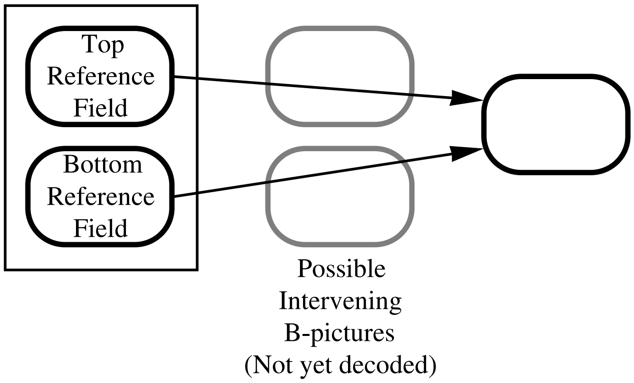


Figure 5.4 Prediction of the first field or field prediction in a frame picture [19].

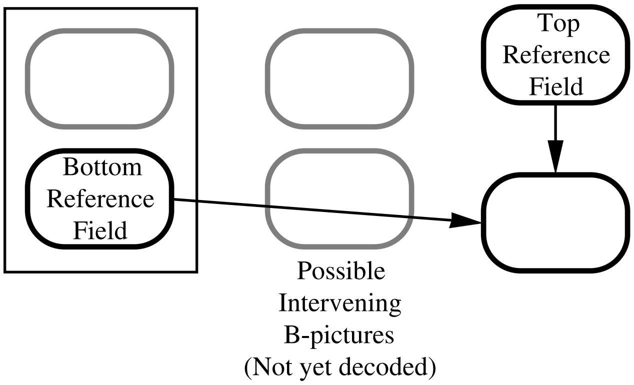


Figure 5.5 Prediction of the second field picture when it is the bottom field [19].

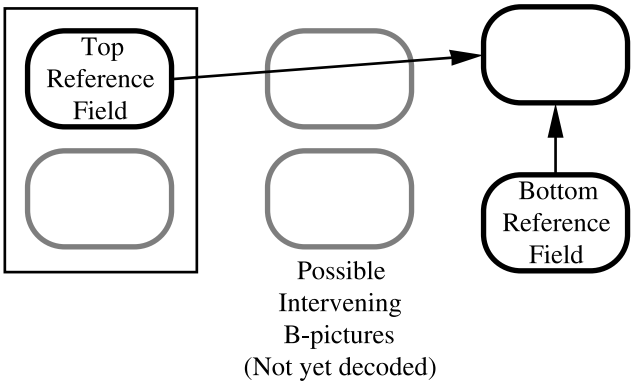


Figure 5.6 Prediction of the second field picture when it is the top field [19].

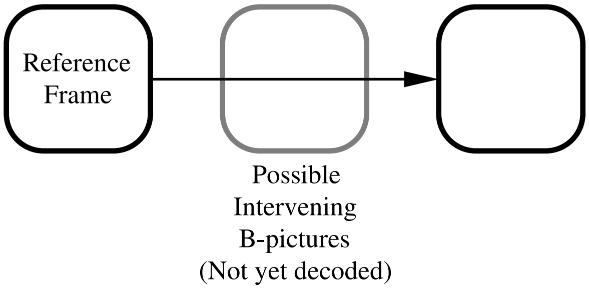
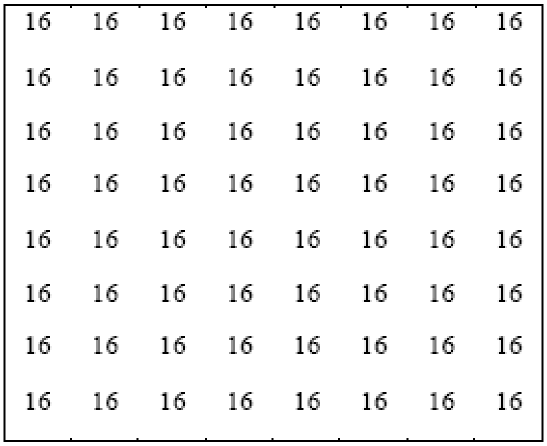


Figure 5.7 Frame prediction of a P picture [19].

5.3.2 DCT Coding

Only the DCT coefficients of the basic 8x8 blocks indicated by the coded block pattern (CBP) in a macroblock are coded. The DCT coefficients of the remaining blocks are quantized as zeros. After applying DCT, the transform coefficients are then quantized using the default quantization matrix (Fig. 5.7) for non-intra macroblocks or a user defined matrix. The transformed and quantized coefficients are then scanned and VLC coded.



16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

Figure 5.8 Default quantization matrix for 8x8 DCT coefficients in non-intra macroblocks [19].

5.4 P Frame Transcoding

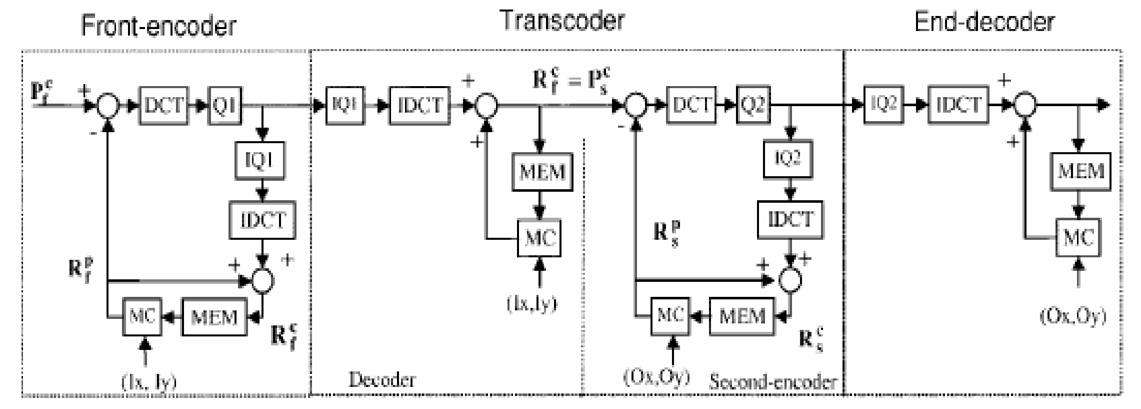


Figure 5.9 Structure of a cascaded transcoder [5].

In all of the video coding standards, motion estimation is one of the most time consuming process in the encoder. The motion estimation process is performed on the luminance macroblock and the best matching block is selected on the basis of sum of absolute difference (SAD) of the respective pixels. The block with minimum value of SAD is considered the best matching block.

To find the motion vector for a macroblock in the current frame, the best matching macroblock is selected from the previous reconstructed macroblock based on the minimum value of SAD within a predefined search window ‘s’ such that [5]:

$$(Ox,\, Oy) = \arg \min_{(m,\, n) \in S} \mathrm{SAD}_s(m,\, n) \qquad (5.5)$$

where;

$$\mathrm{SAD}_s(m,\, n) = \sum_{i}^{M} \sum_{j}^{N} |P_s^c(i,\, j) - R_s^p(i+m,\, j+n)|. \qquad (5.6)$$

where m and n are the horizontal and vertical displacements of the best matching macroblock in the previous reconstructed frame.

The subscripts ‘c’ and ‘p’ denote current and previous frames respectively as shown in Fig. 5.8. The subscript‘s’ and ‘f’ indicate second encoder and front encoder respectively. Since the reconstructed picture in the front encoder is the same as the input picture to the second encoder, the equation can be modified such that:

$$\begin{aligned}\mathrm{SAD}_s(m, n) &= \sum_i \sum_j |P_s^c(i, j) - R_s^p(i+m, j+n)| \\ &= \sum_i \sum_j |P_f^c(i, j) - R_f^p(i+m, j+n) \\ &\quad + \Delta_f^c(i, j) - \Delta_s^p(i+m, j+n)|\end{aligned} \tag{5.7}$$

Where,

$$\Delta_f^c(i, j) = R_f^c(i, j) - P_f^c(i, j) \tag{5.8}$$

Quantization error in the front encoder.

$$\Delta_s^p(i, j) = R_s^p(i, j) - P_s^p(i, j). \tag{5.9}$$

Quantization error of the previous frame in the second encoder.

As can be seen, the simple re use of the H.264 motion vectors does not take into account the quantization errors and hence it will produce non optimal results. Besides this H.264 can have up to 16 motion vectors due to adaptive block size motion estimation. H.264 also has increased pixel accuracy as compared to MPEG-2. These advanced features of H.264 make direct reuse of H.264 motion vectors in MPEG-2 produce non optimal results. Taking the average of all of the motion vectors to arrive at a single 16x16 vector becomes less reliable as the number of sub-block increases because of the possibility that a motion vector was found in a different direction.

5.5 Proposed Method

In P frame transcoding from H.264 to MPEG-2, instead of applying the brute force method of transcoding it is advisable to reuse the motion information from the H.264 decoding stage in the MPEG-2 encoding stage. This will reduce the computational time required for the transcoding process. However direct reuse of H.264 motion vectors will produce non-optimal results due to the increased motion vector resolution and up to 16 sub-macroblock motion vectors. H.264 motion vectors specify a position in ¼ pixel increments where as MPEG-2 uses ½ pixel increments. More problematic are the number of motion vectors encoded for a 16x16 macroblock. While MPEG-2 uses a full search algorithm for the entire 16x16 macroblock, an H.264 coded macroblock can have up to sixteen sub-blocks and their corresponding motion vectors. Hence arises the need to create a single motion vector from all the available H.264 motion vectors for every macroblock so as to be used by MPEG-2.

Since an H.264 macroblock can have the following block sizes: 16x16, 16x8, 8x16 and 8x8. Also each 8x8 block can be sub-divided into 8x8, 8x4, 4x8 and 4x4 hence there can be up to 16 motion vectors for a single 16x16 macroblock (Figure 5.1). For this reason it is important to arrive at a single 16x16 motion vector for the entire macroblock so that it can be re used at the MPEG-2 encoding stage. Simply taking the average of all these motion vectors can produce more than 3db degradation as in [15].

The proposed method extracts motion vectors from H.264 and scales them based on the macroblock size or sub-macroblock size. The scaled motion vectors are then summed together to obtain a single motion vector for the entire 16x16 macroblock. The algorithm for this process is described as follows.

5.5.1 Proposed Algorithm

The proposed algorithm creates a single motion vector for the entire 16x16 macroblock by using the following methods based on the different H.264 block sizes.

For the 16x16 block size, the H.264 motion vector can be applied directly with some motion vector refinement (explained later) onto the MPEG-2 encoding stage.

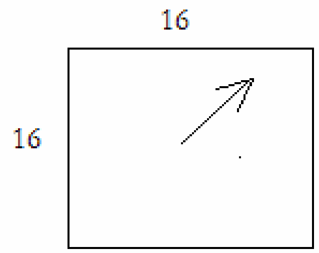


Figure 5.10 Motion vector of a 16x16 block in H.264.

For the 16x8 block size a single motion vector is arrived by the calculation as follows:

MPEG-2mv_x= (H.264mv_a$_x$ +H.264mv_ b$_x$) /2 (5.10)

MPEG-2mv_y= (H.264mv_a$_y$ +H.264mv_b$_y$) (5.11)

Where $H.264mv\_a_x$ and $H.264mv\_ b_x$ are the horizontal components of the 16x8 block sized H.264 motion vectors, while $H.264mv\_a_y$ and $H.264mv\_b_y$ are the vertical components of the 16x8 block sized H.264 motion vectors.

For the 8x16 block size a single motion vector is arrived by the calculation as follows:

$$MPEG\text{-}2mv\_x = (H.264mv\_a_x + H.264mv\_ b_x) \quad (5.12)$$

$$MPEG\text{-}2mv\_y = (H.264mv\_a_y + H.264mv\_b_y)/2 \quad (5.13)$$

In H.264 if the MB type is 3 or 4, the macroblock will be further divided into sub macroblocks. In this case to arrive at a single motion vector for the entire 16x16 block, first single motion vectors are derived for each 8x8 block and then the four motion vectors for each 8x8 block are summed together to get the final 16x16 motion vector for the entire macroblock.

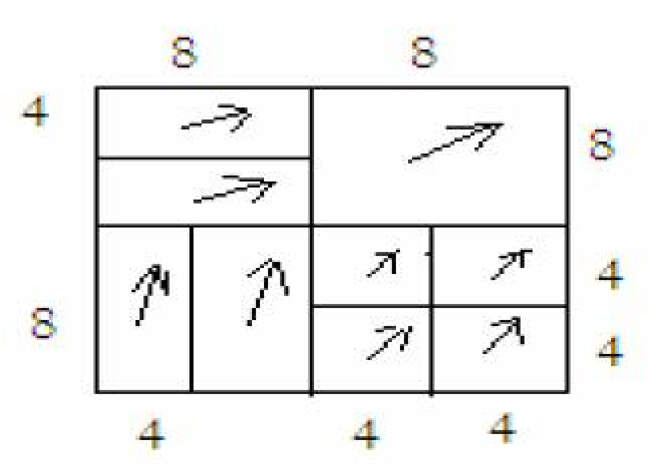


Figure 5.11 Figure depicting an H.264 macroblock with mb_type 3 or 4 and with sub-blocks and respective motion vectors.

If the sub-macroblock type is 0 then the sub-macroblock size is 8x8. A single motion vector for H.264 for this block size is arrived as follows:

$$\mathrm{H.264mv\_a}_x = (\mathrm{H.264mv\_a}_{x1}) \tag{5.14}$$

$$\mathrm{H.264mv\_a}_y = (\mathrm{H.264mv\_a}_{y1}) \tag{5.15}$$

Now in case where the sub-macroblock type is 1, the 8x8 block is split into two 8x4 sub blocks then first a single motion vector is derived for the particular 8x8 block as follows:

$$\mathrm{H.264mv\_a}_x = (\mathrm{H.264mv\_a}_{x1} + \mathrm{H.264mv\_b}_{x1})\ /2 \tag{5.16}$$

$$\mathrm{H.264mv\_a}_y = (\mathrm{H.264mv\_a}_{y1} + \mathrm{H.264mv\_b}_{y1})\ /4 \tag{5.17}$$

Where $\mathrm{H.264mv\_a}_{x1}$ and $\mathrm{H.264mv\_b}_{x1}$ are the vertical components of the motion vectors of the sub-macroblocks, while $\mathrm{H.264mv\_a}_{y1}$ and $\mathrm{H.264mv\_b}_{y1}$ are the horizontal components of the motion vectors of the sub-blocks.

Similarly when the sub-macroblock type is 2, the 8x8 block is divided into two 4x8 sub-blocks, a single motion vector for the 8x8 block is arrived as follows:

$$\mathrm{H.264mv\_a}_x = (\mathrm{H.264mv\_a}_{x1} + \mathrm{H.264mv\_b}_{x1}) \tag{5.18}$$

$$\mathrm{H.264mv\_a}_y = (\mathrm{H.264mv\_a}_{y1} + \mathrm{H.264mv\_b}_{y1})/2 \tag{5.19}$$

Again if the 8x8 sub-macroblock is divided into four 4x4 blocks then a single motion vector is arrived at for the 8x8 block as follows:

$$\mathrm{H.264mv\_a}_x = (\mathrm{H.264mv\_a}_{x1} + \mathrm{H.264mv\_b}_{x1} + \mathrm{H.264mv\_c}_{x1} + \mathrm{H.264mv\_d}_{x1})/4 \tag{5.20}$$

$$\mathrm{H.264mv\_a}_y = (\mathrm{H.264mv\_a}_{y1} + \mathrm{H.264mv\_b}_{y1} + \mathrm{H.264mv\_c}_{y1} + \mathrm{H.264mv\_d}_{y1})/4 \tag{5.21}$$

Finally when all the motion vectors for each of the 8x8 sub-blocks have been obtained then, a single motion vector for the entire 16x16 block is arrived by adding all the motion vectors of each of the 8x8 sub-block as follows:

$$\mathrm{MPEG\text{-}2mv\_x} = 1/4(\mathrm{H.264mv\_x0} + \mathrm{H.264mv\_x1} + \mathrm{H.264mv\_x2} + \mathrm{H.264mv\_x3}) \tag{5.22}$$

MPEG-2mv_y=1/4(H.264mv_y0+H.264mv_y1+H.264mv_y2+H.264mv_y3) (5.23)

Where 0, 1, 2 and 3 are the four different 8x8 blocks in a single 16x16 macroblock.

5.5.2 Motion Vector Refinement

In most cases of transcoding, the effect of quantization errors is negligible and hence performing a new motion estimation will give the same result as the incoming motion vector (i.e. the incoming motion vector is optimal) [5]. However since in general there is no guarantee that the effect is negligible all the time, there are non-zero probabilities that the quantization errors may cause the incoming motion vector to be non-optimal. Although the optimized motion vector can be obtained by new motion estimation, it is not desirable because of its high computational complexity. The reuse of the incoming motion vector [5] has been widely accepted because it is generally thought to be as good as performing new full-scale motion estimation and has been assumed in various transcoding architectures. However simply reusing the incoming motion vector is not optimal especially in this case where a single motion vector has been derived from up to 16 motion vectors.

The differential reconstruction error causes the incoming motion vector to deviate from optimal values. In most macroblocks the deviation is within a small range and the position of the optimal motion vector will be very near to that of the incoming motion vector. Therefore, the optimal motion vector can be obtained by refining the incoming motion vector within a small search range as opposed to performing full-scale motion estimation.

Keeping this in mind, once a single 16x16 motion vector is obtained for the entire macroblock from the incoming H.264 motion vectors, a small search range of -2 pels and +2 pixels around the incoming motion vectors is used to search for the motion vector with the minimum SAD and use it as the optimum motion vector. This will not only give an optimal motion vector but will also reduce the complexity involved in performing new full scale motion estimation. So in order to overcome this I have performed some motion vector refinement using the HAVS (horizontal and vertical search) method [5]. According to this scheme, instead of searching all checking points within the search window, the HAVS scheme searches first for a minimum SAD point in the horizontal line and then over the vertical line. The algorithm employed for the optimum motion vector search in the horizontal direction is as shown in Fig 5.12 and Fig 5.13.

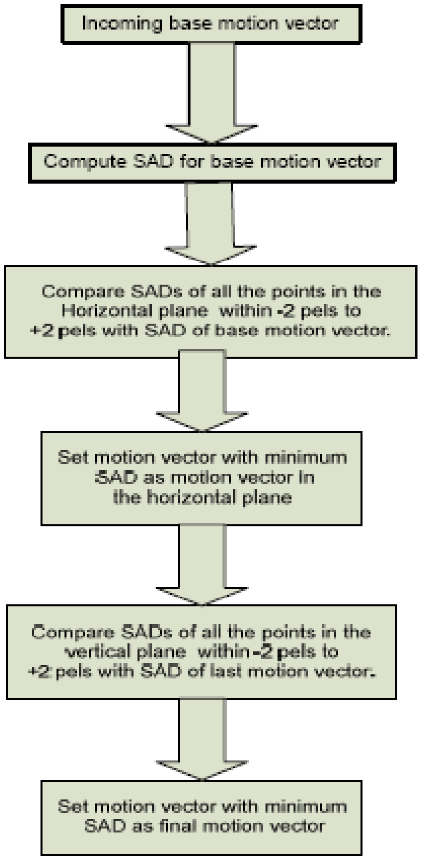

Incoming base motion vector
Compute SAD for base motion vector
Compare SADs of all the points in the Horizontal plane within -2 pels to +2 pels with SAD of base motion vector.
Set motion vector with minimum SAD as motion vector in the horizontal plane
Compare SADs of all the points in the vertical plane within -2 pels to +2 pels with SAD of last motion vector.
Set motion vector with minimum SAD as final motion vector

Figure 5.12 Motion vector refinement algorithm for P frames [5]

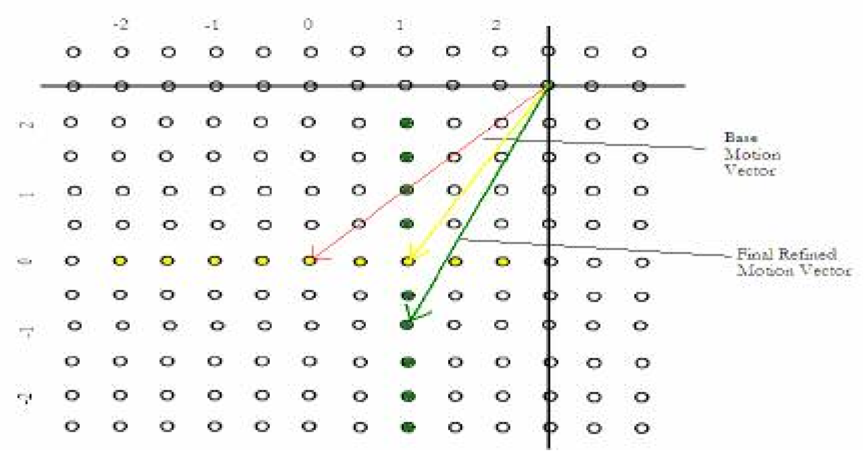


Figure 5.13 Diagram explaining the motion vector refinement scheme

5.6 Results and overview of P frame transcoding

The overall scheme of P frame transcoding can be largely divided into two parts: Extracting the frame data and motion vectors from the H.264 decoding stage and creating a single motion vector out of the extracted motion vectors for each 16x16 macroblock in H.264 so that it can be reused in MPEG-2 and then refining these motion vectors over a small search window at the MPEG-2 encoding side.

The video clips used for this experiment are standard test clips [16]. The objective of the proposed scheme for P frame transcoding is to reduce the complexity of the transcoding process without significant changes in the PSNR. It can be observed from table 5.1 that the PSNR obtained by complete decoding and re-encoding of the H.264 bit stream is comparable to that obtained by the proposed method. The PSNR also depends on other factors such as the bit rate and the motion search window size. For these experiments the bit rate was maintained constant at 1Mbps and the motion search window for the full re-encoding process was maintained at -15 pels to +15pels both in horizontal and vertical directions.

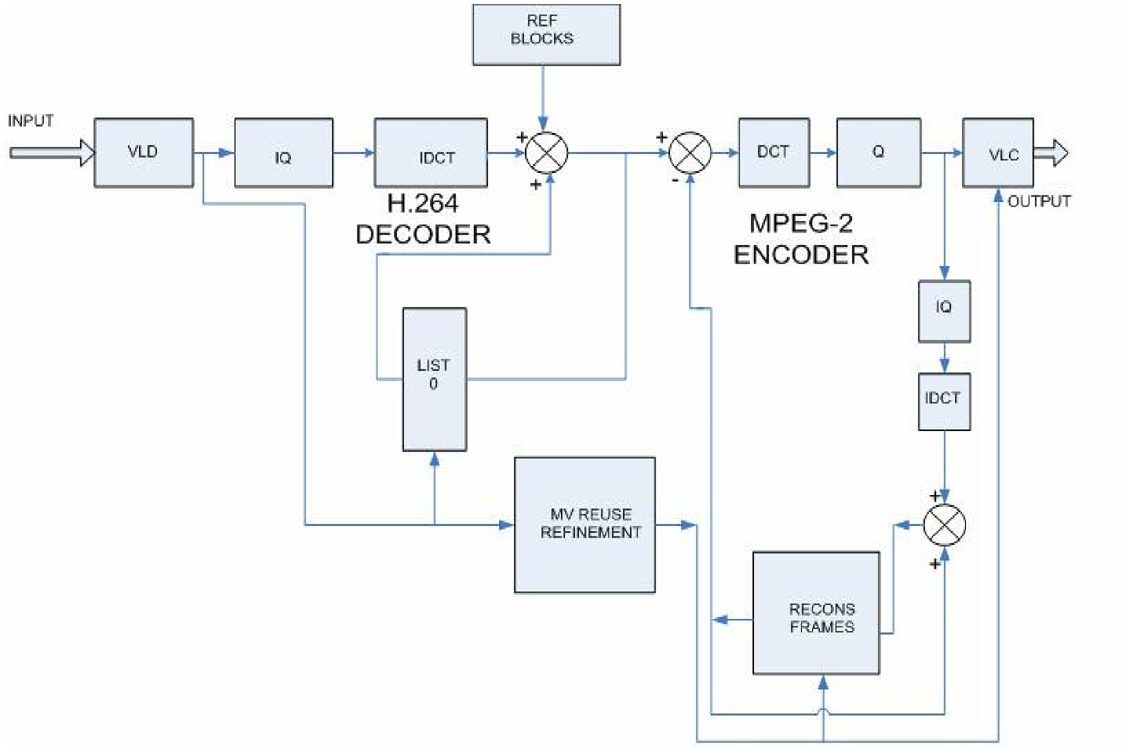
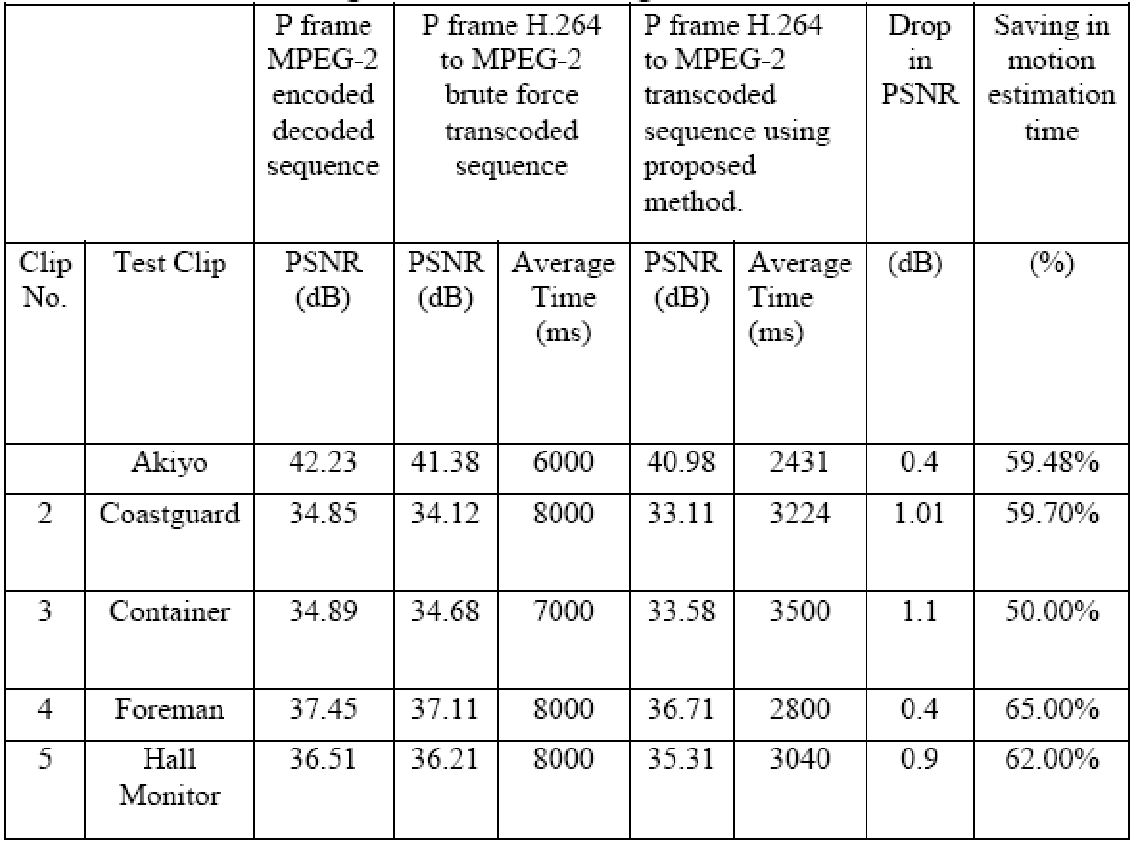


Figure 5.14 Block diagram depicting P frame transcoding of H.264 to MPEG-2 [2]

For P frame transcoding the test sequence is fed into the H.264 encoder and the obtained H.264 bit stream is then decoded. Motion vectors are extracted from the H.264 decoding stage and then summed together using the proposed algorithm. The obtained motion vectors are then refined over -2 pixels to +2 pixels in the horizontal and vertical direction to get the optimum motion vectors. These obtained motion vectors are then fed into the MPEG-2 encoding stage to get the final MPEG-2 transcoded bitstream.

Table 5.1 Results obtained by transcoding a P frame from H.264 to MPEG-2 using the proposed method at 1 Mbps compared with the complete H.264 decoding and MPEG-2 encoding of the H.264 bitstream.



		P frame MPEG-2 encoded decoded sequence	P frame H.264 to MPEG-2 brute force transcoded sequence		P frame H.264 to MPEG-2 transcoded sequence using proposed method.		Drop in PSNR	Saving in motion estimation time
Clip No.	Test Clip	PSNR (dB)	PSNR (dB)	Average Time (ms)	PSNR (dB)	Average Time (ms)	(dB)	(%)
	Akiyo	42.23	41.38	6000	40.98	2431	0.4	59.48%
2	Coastguard	34.85	34.12	8000	33.11	3224	1.01	59.70%
3	Container	34.89	34.68	7000	33.58	3500	1.1	50.00%
4	Foreman	37.45	37.11	8000	36.71	2800	0.4	65.00%
5	Hall Monitor	36.51	36.21	8000	35.31	3040	0.9	62.00%

It can be observed that the results obtained by the proposed method are very close to those obtained by H.264 decoding and MPEG-2 encoding and decoding of the clip. However there is significant saving in the encoding time due to motion vector reuse from the H.264 decoding stage. The transcoder complexity is measured as the total time spent for the MPEG-2 encoding process (as this was the only variable).

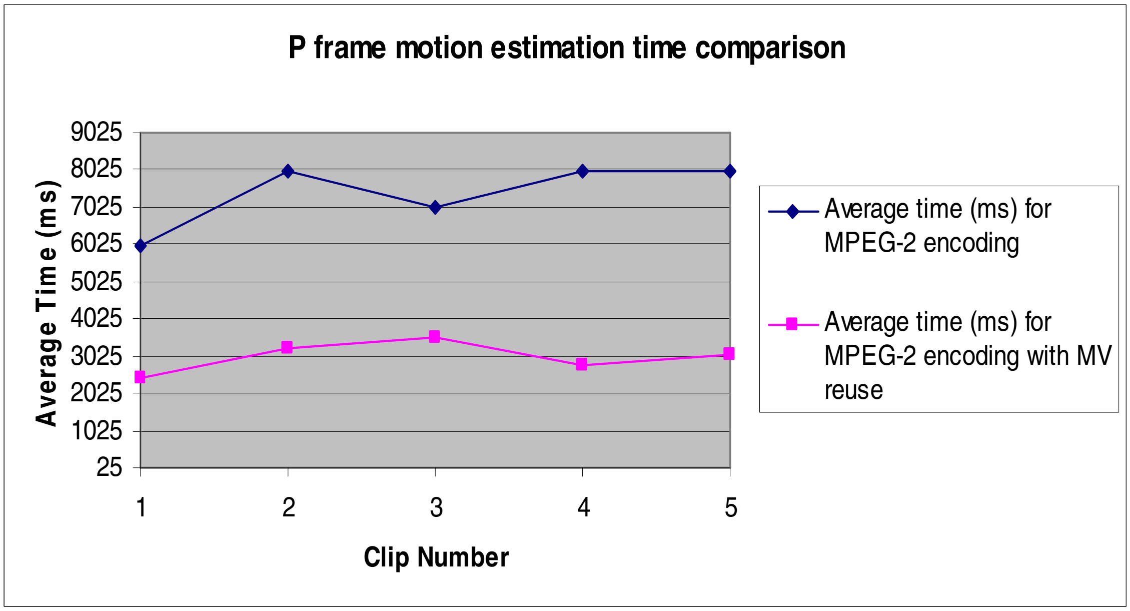


Figure 5.15 Comparison of average time (ms) for the different sequences obtained using the proposed method and MPEG-2 encoding and decoding.

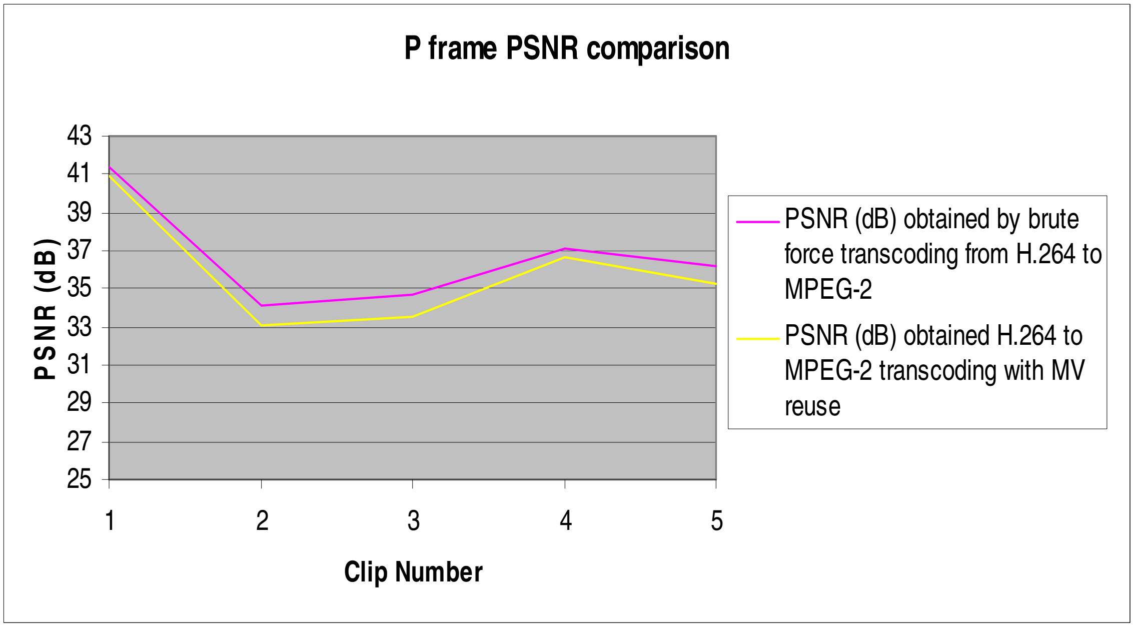


Figure 5.16 Comparison of PSNR (dB) values for the different sequences obtained using the proposed method and MPEG-2 encoding and decoding method.

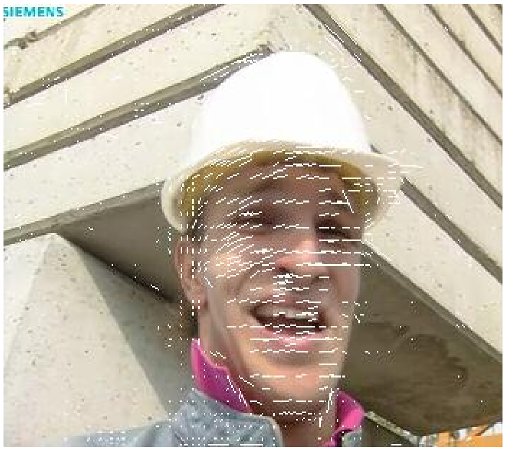


Fig. 5.17 P frame of 1Mbps clip foreman in H.264 stream with motion vectors marked

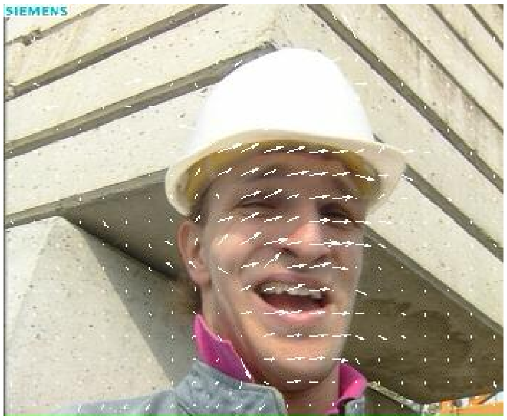


Fig. 5.18 P frame of 1Mbps clip foreman in MPEG-2 transcoded stream with motion vectors marked

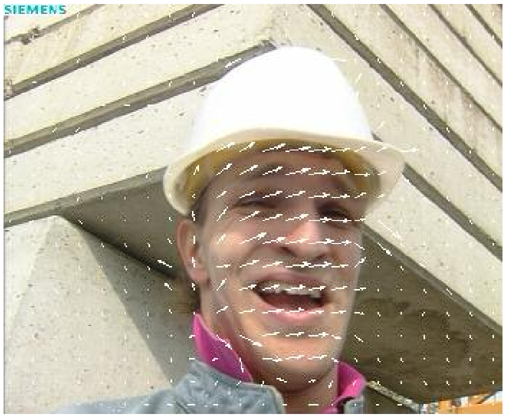


Fig. 5.19 P frame of 1Mbps clip foreman in MPEG-2 encoded stream with motion vectors marked

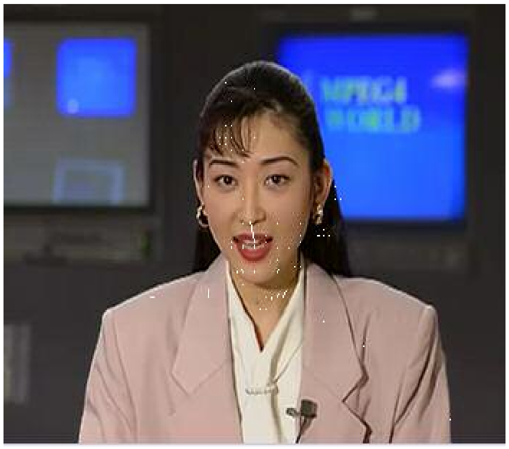


Fig 5.20 P frame of 1Mbps clip Akiyo in H.264 stream with MVs marked.

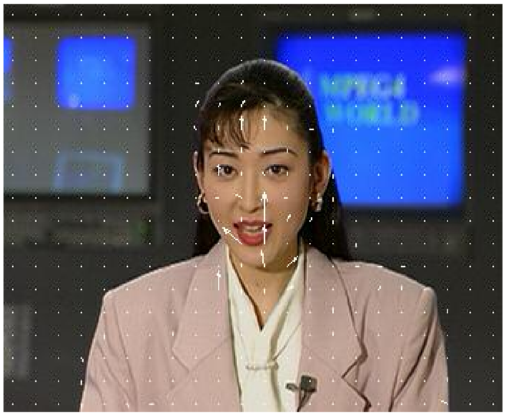


Fig 5.21 P frame of 1 Mbps clip Akiyo in MPEG-2 transcoded stream with motion vectors marked

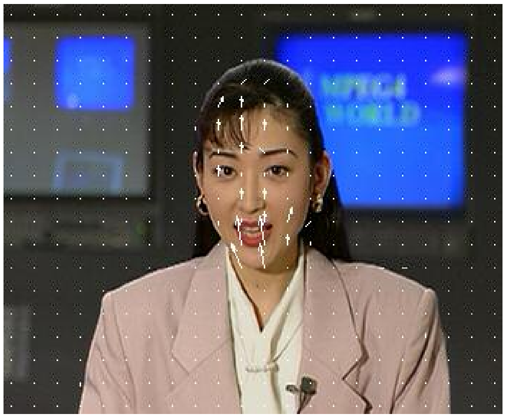


Fig. 5.22 P frame of 1Mbps clip Akiyo in MPEG-2 encoded stream with motion vectors marked.

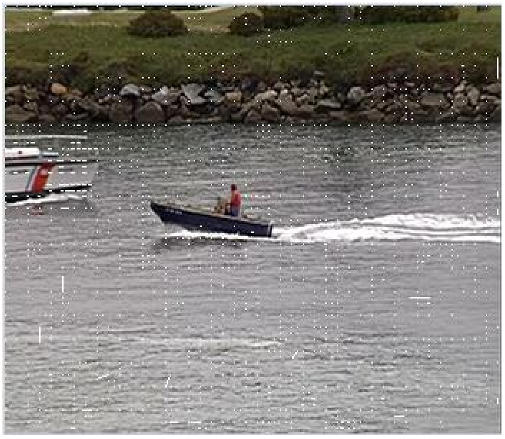


Fig 5.23 P frame of 1Mbps clip coastguard in H.264 stream with motion vectors marked.

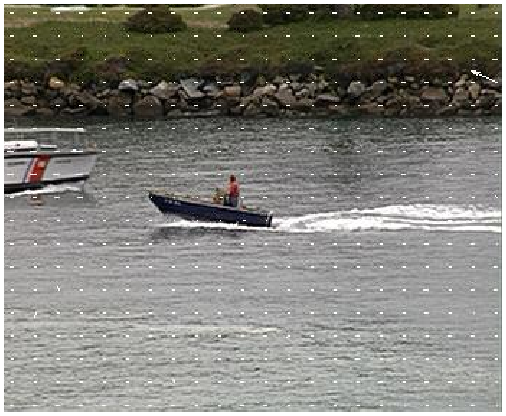


Fig.5.24 P frame of 1Mbps clip coastguard in MPEG-2 transcoded stream with motion vectors marked.

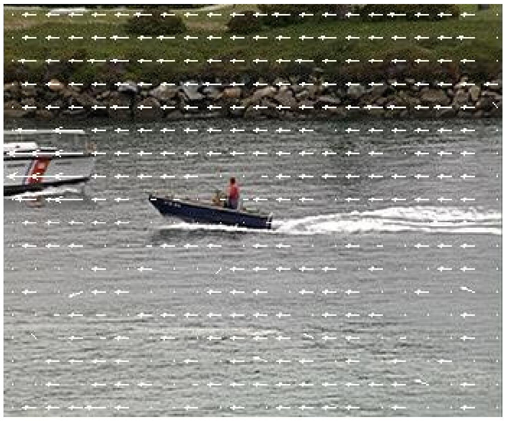


Fig.5.25 P frame of 1Mbps clip coastguard in MPEG-2 encoded stream with motion vectors marked.

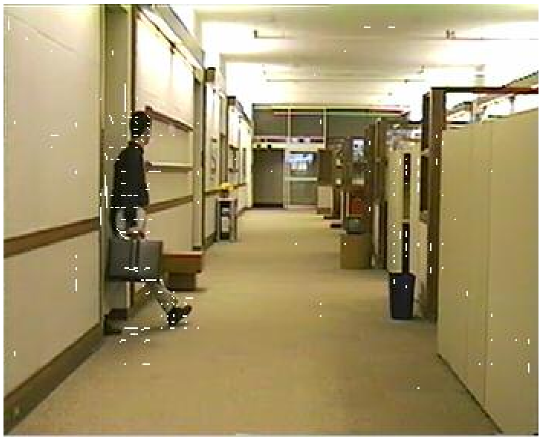


Fig.5.26 P frame of 1Mbps clip hall monitor in H.264 stream with motion vectors marked.

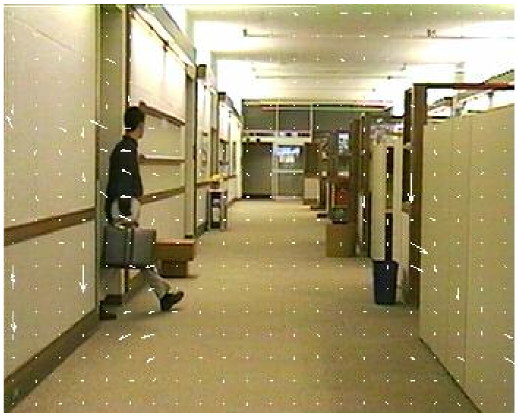


Fig.5.27 P frame of 1Mbps clip hall monitor in MPEG-2 transcoded stream with motion vectors marked.

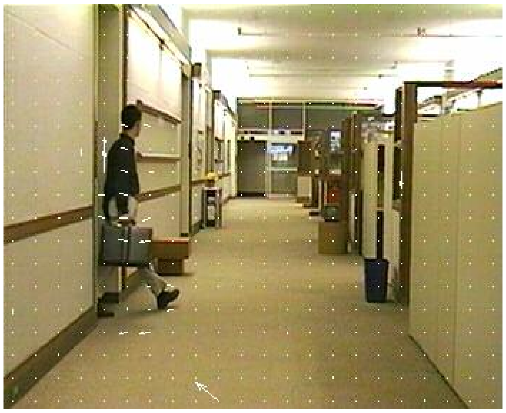
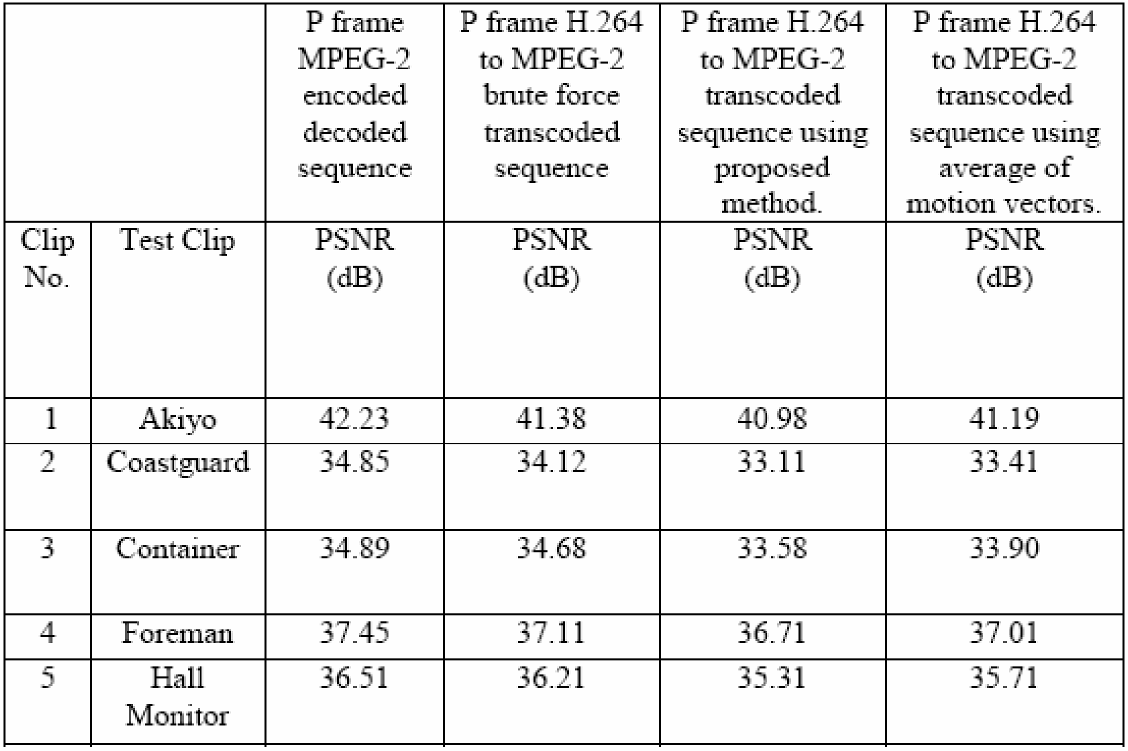


Fig.5.28 P frame of 1Mbps clip hall monitor in MPEG-2 encoded stream with motion vectors marked.

The figures 5.17 to 5.28 depict the results based on my proposed method. However if we take the average of all the motion vectors instead of scaling them according to their area then the motion vector plot is as shown below in figures 5.29 to 5.34.

Table 5.2 Comparison of P frame results obtained by different methods.



		P frame MPEG-2 encoded decoded sequence	P frame H.264 to MPEG-2 brute force transcoded sequence	P frame H.264 to MPEG-2 transcoded sequence using proposed method.	P frame H.264 to MPEG-2 transcoded sequence using average of motion vectors.
Clip No.	Test Clip	PSNR (dB)	PSNR (dB)	PSNR (dB)	PSNR (dB)
1	Akiyo	42.23	41.38	40.98	41.19
2	Coastguard	34.85	34.12	33.11	33.41
3	Container	34.89	34.68	33.58	33.90
4	Foreman	37.45	37.11	36.71	37.01
5	Hall Monitor	36.51	36.21	35.31	35.71

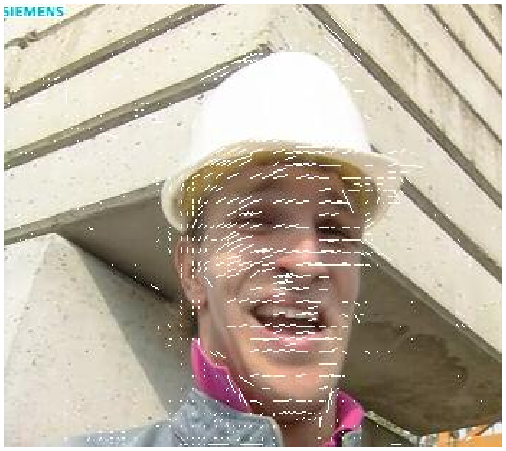


Fig. 5.29 P frame of 1Mbps clip foreman in H.264 stream with motion vectors marked

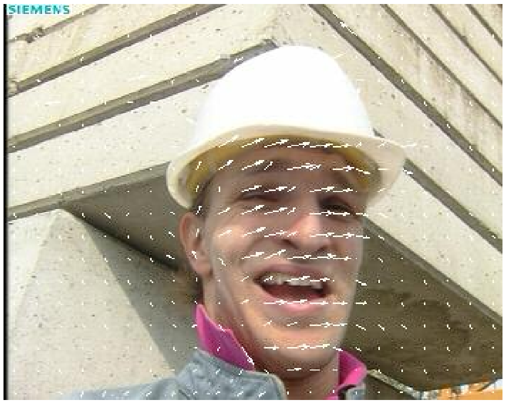


Fig. 5.30 P frame of 1Mbps clip foreman in MPEG-2 transcoded stream with average motion vectors marked

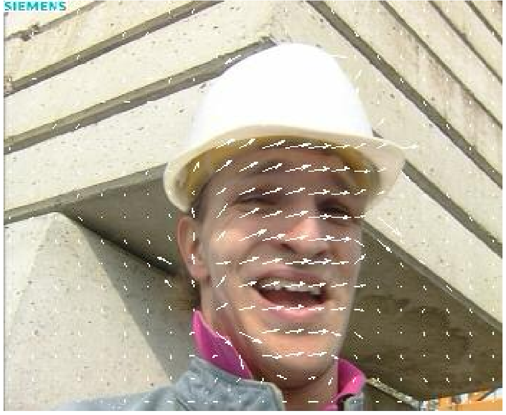


Fig. 5.31 P frame of 1Mbps clip foreman in MPEG-2 encoded stream with motion vectors marked

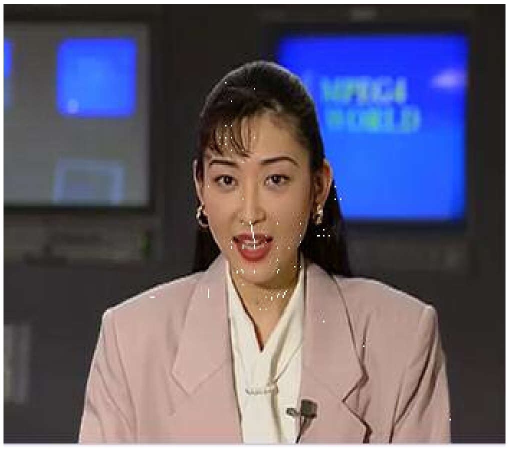


Fig 5.32 P frame of 1Mbps clip Akiyo in H.264 stream with MVs marked.

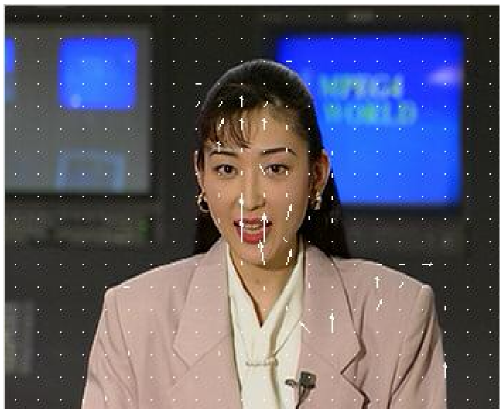


Fig 5.33 P frame of 1 Mbps clip Akiyo in MPEG-2 transcoded stream with average motion vectors marked

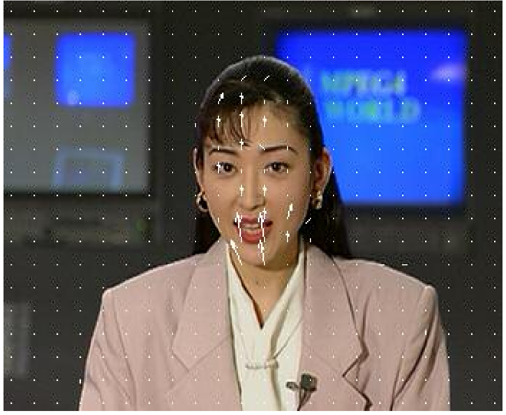


Fig. 5.34 P frame of 1Mbps clip Akiyo in MPEG-2 encoded stream with motion vectors marked.

The proposed method produced comparable results to the one obtained by [14] and much better results than simply taking the average of the motion vectors without any refinement as in [15] which produced up to 3db degradation in PSNR. The results obtained by using [14] are as shown in Figure 5.20.

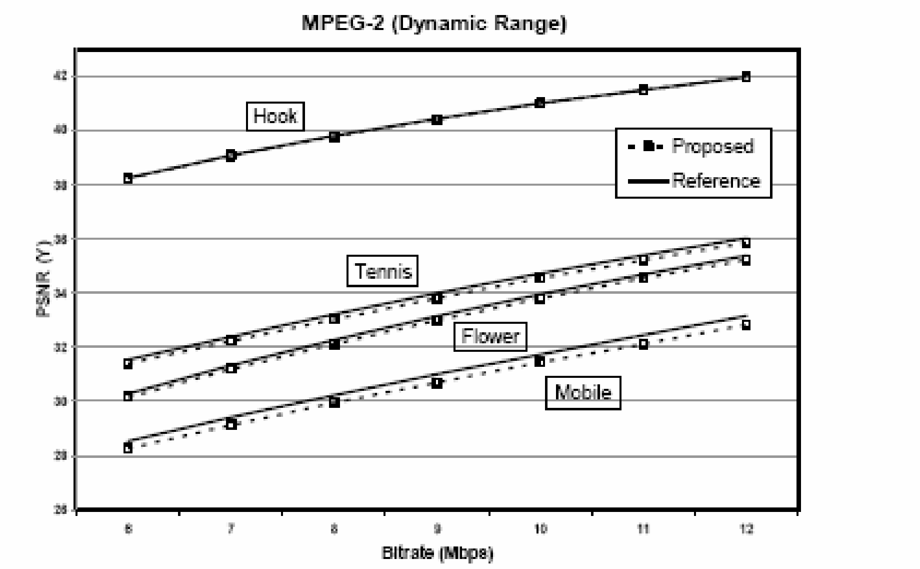


Figure 5.35 PSNR vs bitrate comparison for [14]

The method described in number [14] first takes the average of the incoming motion vectors and then refines them over a search window of the size of the largest incoming motion vectors.

CHAPTER 6

RESULTS AND CONCLUSION

The basic purpose of this thesis is to resolve the incompatibility between H.264 and MPEG-2 standards. Since most of the existing equipment is capable of playing MPEG-2 streams and due to the anticipated shift to H.264 for transport due to large savings in bandwidth, the transcoder will be able leverage the low cost of H.264 video and does not require significant investment in additional video coding hardware.

The main purpose of every transcoder is to produce comparable quality output with low complexity as would be obtained by simply encoding and decoding the stream in the required output format. Keeping this in mind, the results presented so far compare the output of the proposed transcoder with encoding and decoding of the MPEG-2 bitstream.

6.1 Experimental setup

The test clips used are standard CIF (352x288) resolution test clips. For each clip 30 frames are transcoded. For I frame transcoding these clips are transcoded at different bit rates from H.264 to MPEG-2. The GOP size is set equal to 30 with IIIIIIIIII…..GOP structure. The PSNR of the transcoded sequence is then compared with the PSNR of the MPEG-2 encoded decoded sequence. For P frame transcoding, the clips are encoded into H.264 streams at a bit rate of 1Mbps and with a GOP size of

15 and the IPPPPP….GOP. These streams are then transcoded to 1Mbps MPEG-2 with the same GOP structure IPPPPP….. The PSNR of the transcoded sequence is then compared with the PSNR of the MPEG-2 encoded decoded sequence and the brute force method of H.264 to MPEG-2 transcoding. The PSNR values presented are the average PSNR values for all the frames in the test sequence used.

6.2 Results

The subjective quality of the transcoded bitstream is compared for each frame type as shown in figures 6.1-6.2. It can be noted that perceptually the frames are very close in quality.

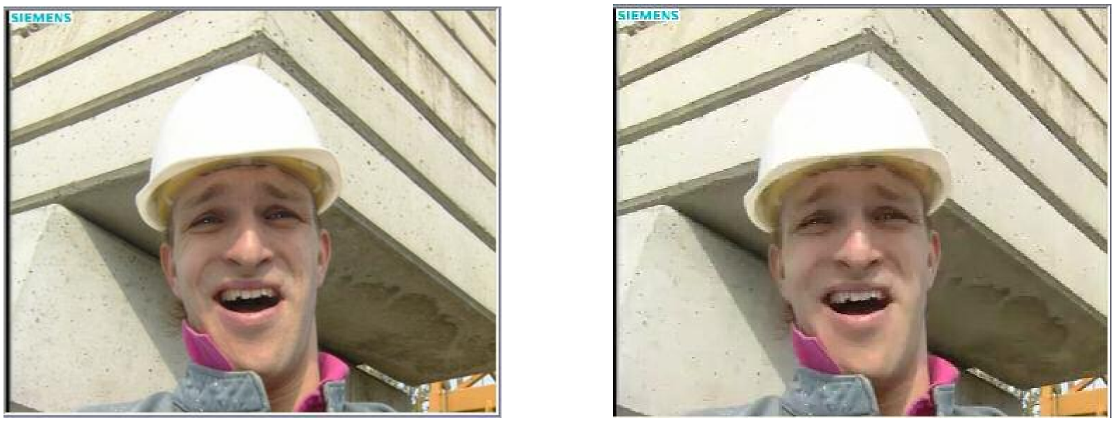


Figure 6.1 Comparison of the subjective quality of the input I frame (left) and the transcoded output I frame for the test sequence Foreman transcoded at 1.4Mbps

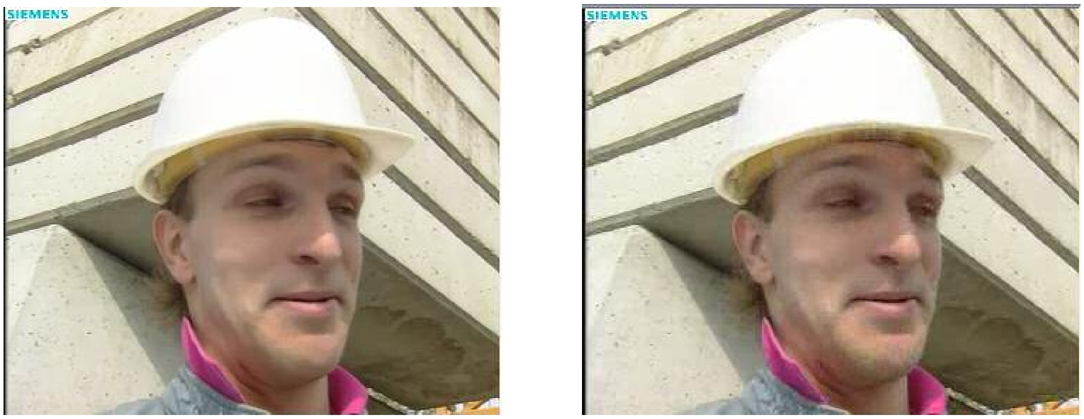


Figure 6.2. Comparison of the subjective quality of the input P frame (left) and the transcoded output P frame (right) for the test sequence Foreman transcoded at 1Mbps

6.3 Conclusions

This thesis is based on transcoding of H.264 bitstream to MPEG-2 bitstream. It employs two parts for transcoding firstly for I frame transcoding the brute force method of complete encoding and decoding is used because H.264 employs intra prediction but in MPEG-2 there is no intra prediction, MPEG-2 simply divides each macroblock into 8x8 blocks and applies a two dimensional DCT across it. In the case of P frame transcoding, in order to reduce the transcoder complexity the motion information from the H.264 decoding stage is reused in the MPEG-2 encoding stage thus leading to a fast transcoding technique with fairly acceptable loss in PSNR. Thus all the expectations of a good transcoder are met.

6.4 Future research

The research presented in this thesis is directed at low complexity, speed and comparable quality. Now that these targets have been achieved, the transcoder can be optimized for use in specific applications.

This transcoder can be applied on PVR or multimedia server to play the delivered H.264 content in the existing MPEG-2 equipment. Moreover this thesis is based on transcoding of H.264 from baseline profile to MPEG-2 simple profile with no B frames. The same process could be extended to transcoding from the main profile of H.264 to main profile of MPEG-2 with the presence of B frames.

REFERENCES

[1] J.Xin, C.W.Lin, M.T.Sun , "Digital Video Transcoding" , Proceedings of the IEEE, Vol. 93, Issue 1,pp 84-97, January 2005.

[2] J.Wang et. al.,"An AVS to MPEG-2 Transcoding System", Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, October 20-22, 2004 Hong Kong.

[3] A.Vetros, C.Christopoulos and H.Sun, "Video transcoding architectures and techniques: an overview", IEEE Signal Processing magazine, Vol. 20, Issue 2, pp 18-29,March 2003.

[4] S.K.Kwon, A. Tamhankar and K.R. Rao, "Overview of H.264 / MPEG-4 Part 10 (pp.186-216)", Special issue on " Emerging H.264/AVC video coding standard", J. Visual Communication and Image Representation, vol. 17, pp.183-552, April 2006.

[5] J. Youn and M.T.Sun , "Motion Vector Refinement for high-performance transcoding", in IEEE Int. Conf. Consumer Electronics, Los Angeles, C.A., Vol. 1, Issue 1, pp 30-40,March 1999.

[6] H.Kalva, "Issues in H.264/MPEG-2 Video Transcoding", Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL.

[7] B. Haskell, A. Puri and A. Netravali, "Digital Video: an introduction to MPEG-2", Chapman and Hall, 1997.

[8] MPEG-2 software (version 12) from MPEG software simulation group http://www.mpeg.org/MPEG/MSSG/#source.

[9] H.264 software JM (10.2) from:
http://iphome.hhi.de/suehring/tml/download/

[10] T.Wiegand et. al., “Overview of the H.264/AVC Video Coding Standard”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, Issue 7, July 2003.

[11] P.N.Tudor, “ Tutorial on MPEG-2 Video Compression”, IEE J Langham Thomson Prize, Electronics and Communication Engineering Journal, December 1995.

[13] I.E.G.Richardson, “H.264 and MPEG-4 Video Coding for Next Generation Multimedia”, Wiley 2003

[14] P.Kunzelman and H.Kalva, “Reduced complexity H.264 to MPEG-2 transcoder”, ICEE paper 10/02/2006.

[15] J.Chu et al., “H.264/MPEG-2 Transcoding based on Personal Video Recorder Platform”

[16] Test streams obtained from:
http://www.cipr.rpi.edu/resource/sequences/sif.html

[17] Commercially available transcoders, PSP Video 9,
http://www.pspvideo9.com

[18] J.McVeigh et. al., “A software based real time MPRG-2 video encoder”, IEEE Trans. CSVT, Vol 10, pp 1178-1184, Oct.2000.

[19] Information Technology – Generic coding of moving pictures and associated audio information: Video, ITU-T Rec H.262 (2000 E).

[20] T.Shanableh and M.Ghanbari, “Transcoding architectures for DCT domain heterogenous video transcoding”, Proc.IEEE ICIP, Vol.1 pp 433-436, Thessaloniki, Greece, Sept 2001.

[21] R.Periera, “Efficient transcoding of MPEG-2 to H.264”, Master’s thesis DEC 2005, EE Department, U.T.A.

[22] ITU-T, “Advanced video coding for generic audio visual services”, H.264, March 2005

[23] K.R.Rao and J.J.Hwang, “Techniques and standards for Image, Video and Audio Coding”, Prentice Hall, 1996

[24] I.E.G.Richardson, “Video Codec Design: developing image and video compression systems”, Chichester: Wiley, 2002.

[25] A.Puri et.al., “Video coding using the H.264/MPEG-4 AVC standard”, Signal Processing: Image Communication 19, 793–849, 2004.

BIOGRAPHICAL INFORMATION

Sreejana is currently pursuing a master's degree in the electrical engineering department of the University of Texas at Arlington. Sreejana has been an intern in the consumer electronics department of the Digital Home Group of Intel Corporation since June 2005. Sreejana is also a student member of the IEEE.