

STUDY AND ANALYSIS OF ERROR RESILIENCE IN H.264/AVC VIDEO CODING STANDARD

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of

MASTER OF TECHNOLOGY

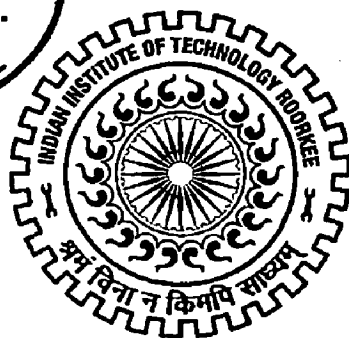
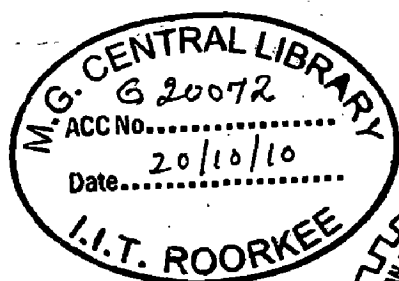
in

ELECTRONICS AND COMMUNICATION ENGINEERING

(With Specialization in Communication Systems)

By

RAM NARAYAN DUBEY



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2010**

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this dissertation report entitled, "Study and Analysis of Error Resilience in H.264/AVC Video Coding Standard" towards the partial fulfillment of the requirements for the award of degree of **Master of Technology in Electronics and Communication Engineering** with specialization in **Communication Systems**, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, is an authentic record of my own work carried out during the period from July 2009 to June 2010, under the guidance of **Dr. Debashis Ghosh, Associate Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee.**

The content of this dissertation has not been previously submitted for examination as part of any academic qualifications.

Date: 28-06-2010

Place: Roorkee



RAM NARAYAN DUBEY

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 28-06-2010

Place: Roorkee


Dr. DEBASHIS GHOSH
Associate Professor, E&C Department,
Indian Institute of Technology Roorkee,
Roorkee-247667 (India)

ACKNOWLEDGEMENT

There are people, who, simply by being what they are, influence and inspire you to do things you never thought yourself capable of doing. Among these are my teachers, family, friends and all my well wishers who made great contributions directly or indirectly by giving suggestions for improvement in one way or the other. During my master's studies at Indian Institute of Technology, Roorkee, I have had the opportunity to work with many excellent people whose support and encouragement have made this dissertation possible. I owe my deepest gratitude to all of them.

First, I would like to thank my supervisor, Dr. Debashis Ghosh, for his guidance and advice throughout the dissertation. His continuous support and encouragement have been instrumental in completing this work. It has been both a pleasure and an honour for me to work with him, an outstanding advisor – helpful, kind, insightful, and patient.

I am also thankful to the staff of E&C department, IIT Roorkee, for their cooperation and assistance and to my friends, especially Sandy, for sharing their knowledge and providing useful suggestions.

Fortunately, I belong to a family whose continuous support and inspiration led me to complete this work. I am grateful to my parents for their unflagging love and faith in my ability throughout my life.

Last but not the least, thanks be to God for my life through all tests in past two years. You have made my life more bountiful. May your name be exalted, honored and glorified.

ROORKEE, JUNE 2010

RAM NARAYAN DUBEY

LIST OF ACRONYMS

B	Bi-predicted (picture, slice, or macroblock)
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAVLC	Context-based Adaptive Variable Length Coding
CIF	Common Intermediate Format (352x288 luma samples)
DCT	Discrete Cosine Transform
DSL	Digital Subscriber Line
DVD	Digital Versatile Disc
FMO	Flexible Macroblock Ordering
H.264/AVC	Advanced Video Coding standard
HDTV	High Definition television
I	Intra-coded (picture, slice, or macroblock)
IDR	Instantaneous Decoding Refresh
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
ISDN	Integrated Service Digital Network
ISO	International Standardization Organization
ITU	International Telecommunication Union
ITU-T	Telecommunications Standardization Sector of ITU
JM	Joint Model, the reference software of H.264/AVC
JVT	Joint Video Team
Kbps	Kilobits per second

LAN	Local Area Network
MMS	Multimedia Messaging Service
MPEG	Moving Picture Experts Group
NAL	Network Abstraction Layer
P	Predicted (picture, slice, or macroblock)
PSNR	Peak Signal-to-Noise Ratio
QCIF	Quarter Common Intermediate Format (176x144 luma samples)
RBSP	Raw Byte Sequence Payload
RTP	Real-time Transport Protocol
SDTV	Standard Definition television
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VCEG	Video Coding Experts Group
VCL	Video Coding Layer

ABSTRACT

Real-time transmission of video data in network environments, such as wireless and Internet, is a challenging task, as it requires high compression efficiency and network-friendly design. To deal with these two challenges, not only the video needs to be compressed very efficiently but also the compression scheme needs to provide some error resilient features to deal with the high packet loss probability. The Advanced Video Coding standard (H.264/AVC) has become a widely deployed coding technique, which aims at achieving improved compression performance and network-friendly video representation for different types of applications, such as Blu-ray Disc, Adobe Flash, Video Conferencing, and Mobile Television.

H.264/AVC utilizes predictive coding to achieve high compression ratio. Predictive coding also makes H.264/AVC bitstreams vulnerable to transmission errors, as prediction incurs temporal and spatial propagation of the degradations caused by transmission errors. Due to the delay constraints of real-time video communication applications, transmission errors cannot usually be tackled by reliable communication protocols. Yet, most networks are susceptible to transmission errors. Consequently, error resilience techniques are needed to combat transmission errors in real-time H.264/AVC-based video communication.

The aim of this dissertation is to study in details the error resilient features of H.264/AVC in real-time video communication applications. A part of the work presented in this dissertation was targeted at specific features of the H.264/AVC standard, including slice structuring, intra placement and weighted prediction. Computer simulations were performed to investigate the performance of H.264/AVC standard incorporating the above mentioned features.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	iii
LIST OF ACRONYMS	iv
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	ix
CHAPTER 1: INTRODUCTION	1
1.1 Outline and Objectives of the Dissertation	4
CHAPTER 2: REVIEW OF VIDEO CODING	6
2.1 Video Compression- Basic Principles	7
2.1.1 Temporal Module	7
2.1.2 Spatial Module	10
2.1.3 Entropy Encoder	15
CHAPTER 3: THE H.264/AVC VIDEO CODING STANDARD	17
3.1 The H.264/AVC Codec-Video Coding Layer	19
3.1.1 Subdivision of Picture into Macroblocks	23
3.1.2 Intra Prediction	24
3.1.3 Motion Compensated Prediction	26
3.1.4 Transform Coding	26
3.1.5 Entropy Coding Schemes	27
3.1.6 In Loop Deblocking Filter	29
3.2 Profiles and Levels	29
3.3 Network Abstraction Layer	31
3.3.1 NAL Units	31
3.3.2 NAL Units in Byte-Stream Format Use	32

3.3.3 NAL Units in Packet-Transport System Use	32
3.3.4 VCL and Non-VCL NAL Units	32
3.3.5 Parameter Sets	33
3.4 Transport of H.264/AVC	33
CHAPTER 4: ERROR RESILIENCE TOOLS IN H.264/AVC.....	36
4.1 Error Resilience Features of H.264	36
4.1.1 Semantics, Syntax and Error Detection	37
4.1.2 Intra Placement	37
4.1.3 Data Partitioning	38
4.1.4 Slice Structuring: Flexible Macroblock Ordering	39
4.1.5 Redundant Slices	42
CHAPTER 5: RESULTS AND DISCUSSION	43
5.1 H.264/AVC Video Compression	43
5.2 Error Resilience Coding	45
5.2.1 Intra Placement	45
5.2.2 Slice Structuring	49
5.2.3 Weighted Prediction	53
CHAPTER 6: CONCLUSION.....	56
6.1 Future Scope	57
REFERENCES	58

LIST OF FIGURES

Figure No.	Title	Page No.
1.1	Functional block diagram for a video communications system	1
1.2	Example of temporal error propagation	2
2.1	General block diagram of a video encoder	7
2.2	Motion-compensated predictions	8
2.3	Macroblock (4:2:0)	8
2.4	Block matching	9
2.5	(a) 80 x 80 pixel image; (b) 2-D DCT	11
2.6	Quantization and rescaling	13
2.7	Zigzag scan	14
2.8	Functional diagram of video encoder	16
2.9	Functional diagram of video decoder	16
3.1	Luminance PSNR versus average bitrate for different video coding standard	18
3.2	Structure of H.264/AVC video encoder	18
3.3	Application areas of H.264/AVC	19
3.4	Typical encoder of H.264/AVC	20
3.5	Typical decoder of H.264/AVC	20
3.6	Multi-frame motion compensation	22
3.7	4 x 4 Luma prediction modes	25
3.8	Labeling of prediction sample (4 x 4)	25
3.9	Partitioning of Macro block (top) and sub-macro block (bottom) for motion-compensated prediction	25
3.10	Matrices H1, H2 and H3 of the three different transforms applied in H.264/AVC channel	27
3.11	CABAC Block diagram	28

3.12	The H.264/AVC Baseline, Main and Extended profiles	30
3.13	Sequence of RBSP	34
4.1	Slice Group: Interleaved map (three slice group)	40
4.2	Slice Group: Dispersed map (four slice group)	40
4.3	Slice Group: Box-out map	41
4.4	Slice Group: Raster map	41
4.5	Slice Group: Wipe map	41
5.1	Compression efficiency for H.264/AVC video coding standard	44
5.2	Average Y-PSNR over packet error rate under Intra frame placement	46
5.3	Average Y-PSNR over packet error rate in case of Bi-prediction	48
5.4	Division of an image into several slice groups using FMO	49
5.5	Average Y-PSNR over packet error rate under different slice structures	50
5.6	Dispersed FMO (4 slice groups per frame)	51
5.7	Performance comparison of 4 slice group versus 2 slice group per frame	52
5.8	Performance of Weighted-prediction	53
5.9	Performance of Weighted-prediction under slice structuring	54

Chapter 1

INTRODUCTION

Digital video communication systems, such as digital television and video streaming over the Internet, play important role in every-day life of many people. A simplified block diagram of a general video communication system is depicted in Figure 1.1 [1]. Due to the fact that uncompressed video requires a huge bandwidth, the input video is compressed by the source coder to a desired bitrate. The source coder may be divided into two components, namely the waveform coder and the entropy coder. The waveform coder performs lossy video signal compression, whereas the entropy coder losslessly converts the output of the waveform coder into a bit-stream. The transport coder following the source coder encapsulates the compressed video according to the communication protocols in use. The data is then transmitted to the receiver side via a transmission channel. The receiver performs inverse operations to obtain a reconstructed video signal for display.

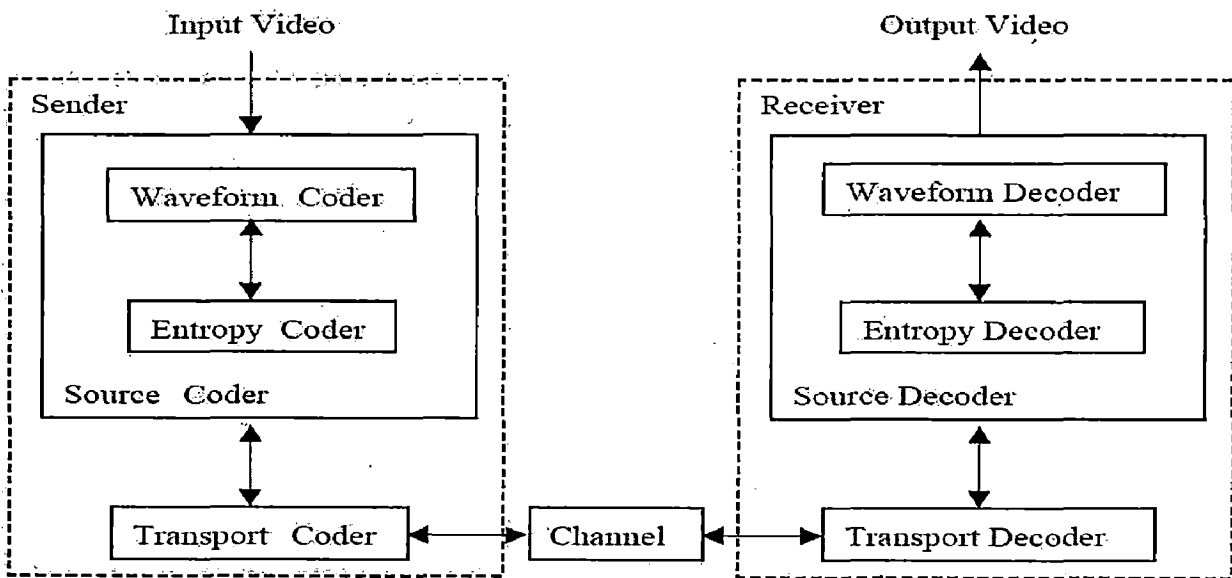


Figure 1.1: Functional block diagram for a video communications system.

Although the waveform coder can use any known video-coding method, in this dissertation, we will mainly focus on the type of hybrid coder that uses DCT and motion-compensated prediction.

Predictive coding is utilized in the waveform coder to achieve high compression efficiency. There are two basic types of prediction involved in video coding: intra and inter. Intra prediction refers to the estimation of a pixel block in a frame from other areas of the same frame. In inter prediction, a pixel block is estimated based on previous frames, usually by indicating the location of a similar pixel block in a previous frame as a motion vector.

As most real-world channels are susceptible to transmission errors, certain measures need to be considered in order to protect the video data from such errors. While error-free communication may be achieved by retransmitting data packets until they are correctly received, real time video communication cannot rely solely on retransmission due to delay constraints arising from user expectations and requirements. Therefore, it is important to devise video encoding/decoding schemes that are capable of making the compressed bit-stream resilient to transmission error.

Predictive coding makes video vulnerable to transmission errors [2]. In this, not only the regions that correspond to the lost or corrupted transmission packets are visibly damaged, but the damaged regions also propagate spatially and temporally. When a degraded region is used as a source for intra-prediction or inter-prediction, the damaged area becomes larger or spans in time, respectively. Figure 1.2 shows three consecutive coded frames illustrating how a degraded region propagate temporally and spatially if transmission error occurs in a previous frame.



Figure 1.2: Example of temporal error propagation.

To make the compressed bit-stream resilient to transmission error, one must add redundancy into the stream, so that it is possible to detect and correct errors. Such redundancy may be added either in the source coder or in the channel coder. The classical Shannon information theory states that one can separately design the source and channel coders to achieve error-free delivery of a compressed bit stream as long as the source is represented by a rate below the channel capacity. Therefore, the source coder should compress a source as much as possible (below the channel capacity) for a specified distortion, and then the channel coder may add redundancy through FEC to the compressed bit-stream to enable correction of transmission errors. However, such ideal error-free delivery can be achieved only with infinite delays in implementing FEC and hence is not acceptable in practice. Therefore, joint source and channel coding is often a more viable scheme, which allocates a total amount of redundancy between the source and channel coding. All the *error resilient encoding* methods essentially work under this premise, and intentionally make the source coder less efficient than it can be, so that the erroneous or missing bits in a compressed stream will not have a disastrous effect in the reconstructed video quality. This is usually accomplished by carefully designing both the predictive coding loop and the variable length coder so as to limit the extent of error propagation.

The channel, as referred to in Figure 1, usually consists of one or more networks including network elements and links connecting those network elements. Data communication over the channel is typically considered to comply with a stack of communication protocols usually organized in layers, including a physical layer, a link layer, a network layer, a transport layer, and an application layer. The physical layer provides physical means for connections between network elements, whereas the link layer manages data links between network elements. The network layer provides addressing of end-points and performs routing of transmitted data through the network. The transport layer provides a connection-oriented or connectionless end-to-end message transfer functionality between end-points. The application layer is the top-most layer in the protocol stack and serves the end-user directly. Source coding is considered to be included in the application layer. Error correction and concealment techniques can

operate in any layer of the protocol stack.

Standardization aims at creating specifications that enable development of interoperable implementations. Standards therefore have an essential role in open communication systems. The Advanced Video Coding standard [3][4], referred to as H.264/AVC, is one of the most recently specified video compression standards. As most other video coding standards, it specifies the bit-stream format and the decoding process for compliant bit-streams. H.264/AVC improves compression efficiency substantially compared to previous standards [5], such as MPEG-2 Video and H.263, and provides flexibility for a wide variety of applications and networks [6]. H.264/AVC is deployed extensively in products and services such as Blu-ray Disc, Adobe Flash, video conferencing, and mobile television.

1.1 Outline and Objectives of the Dissertation

This dissertation presents methods for reducing the quality degradation caused by transmission errors in video communication systems using H.264/AVC. The goal of the work is to study the error resilience of H.264/AVC video coding standard.

This dissertation primarily focuses on error resilience techniques that operate in the application layer and involve H.264/AVC encoders and/or decoders. The rest of this thesis is organized as follows:

Chapter 2 provides basic information on video coding, and mainly focuses on the need of compression, and the way to achieve the same. Basic working of video coding and decoding is also described in this chapter.

Chapter 3 demonstrates the H.264/AVC video coding standard in detail. H.264/AVC has adopted a two-layer structure design: a video coding layer (VCL) and a network abstraction layer (NAL), both of which are described in this chapter.

Chapter 4 describes in detail error resilience features such as flexible macroblock ordering, intra placement, and redundant slices etc., adopted in H.264/AVC.

Chapter 5 presents the results obtained by computer simulation in this dissertation work. The performance of various error resilience features adopted in H.264/AVC is discussed in this chapter. We also analyse the effect of these features in video coding layer.

Chapter 6 lists conclusions that we derive from the present work and outlines scope for further work in this area.

Chapter 2

REVIEW OF VIDEO CODING

During the last two decades, digital video compression methods have played a key role in virtually all applications that involve video. The main reason is that a raw digital video signal contains a huge amount of data, requiring a high capacity that the current transmission and storage media cannot practically provide. Due to the huge bandwidth requirements of raw video signals, a video application running on any networking platform can swamp the bandwidth resources of the communication medium if video frames are transmitted in the uncompressed format. For example, let us assume that a video frame is digitized in the form of discrete grids of pixels with a resolution of 176 pixels per line and 144 lines per frame. If the frame colour is represented by two chrominance frames, each one of which has half the resolution of the luminance frame, then each video frame will need approximately 38 kbytes to represent its content when each of luminance or chrominance component is represented with 8-bit precision. If the video frames are transmitted without compression at a rate of 25 frames per second, then the raw data rate for video sequence is about 7.6 Mbit/s and a 1-minute video clip will require 57 Mbytes of bandwidth. For a CIF (Common Intermediate Format) resolution of 352×288 , with 8-bit precision for each luminance or chrominance component and a half resolution for each colour component, each frame will then need 152 kbytes of memory for digital content representation. With a similar frame rate as above, the raw video data rate for the sequence is almost 30 Mbit/s, and a 1-minute video clip will then require over 225 Mbytes of bandwidth. This implies that digital video data must be compressed before transmission in order to optimize the required bandwidth for the provision of a multimedia service.

2.1 Video Compression- Basic Principles

The basic idea of compression is to remove redundancy. Standard video coding methods are based on three fundamental redundancy reduction principles: spatial and temporal redundancy reduction, and entropy coding [7]. A video encoder consists of three main functional units: a temporal module to reduce the redundancy between consecutive frames, a spatial module to reduce the redundancy in the same frame and an entropy encoder to reduce the statistical redundancy. Figure 2.1 shows a general video encoder consisting of these three modules.

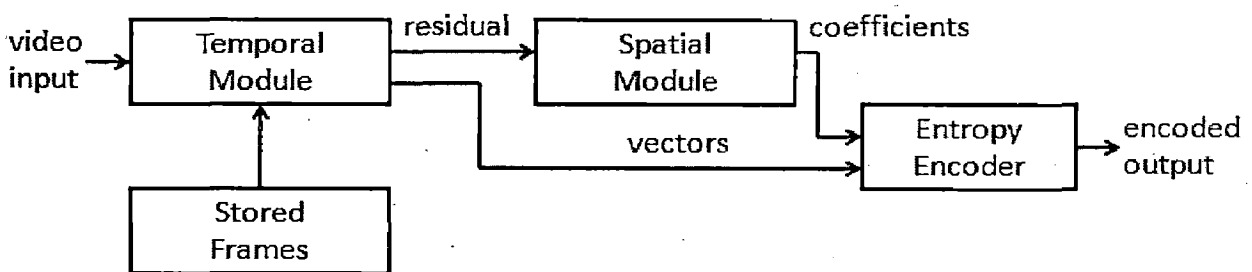


Figure 2.1: General block diagram of a video encoder

2.1.1 Temporal Module

The input to the temporal module is an uncompressed video sequence. The temporal module attempts to reduce temporal redundancy by exploiting the similarities between neighbouring video frames, usually by constructing a prediction of the current video frame. In H.264, the prediction is formed from one or more previous or future frames and is improved by compensating for differences between the frames (motion compensated prediction). The output of the temporal module is a residual frame (created by subtracting the predicted frame from the actual current frame) and a set of model parameters, typically a set of motion vectors describing how the motion was compensated in predicting the current frame from the previous/future reference frame. Figure 2.2 shows the basic operation in motion-compensated prediction.

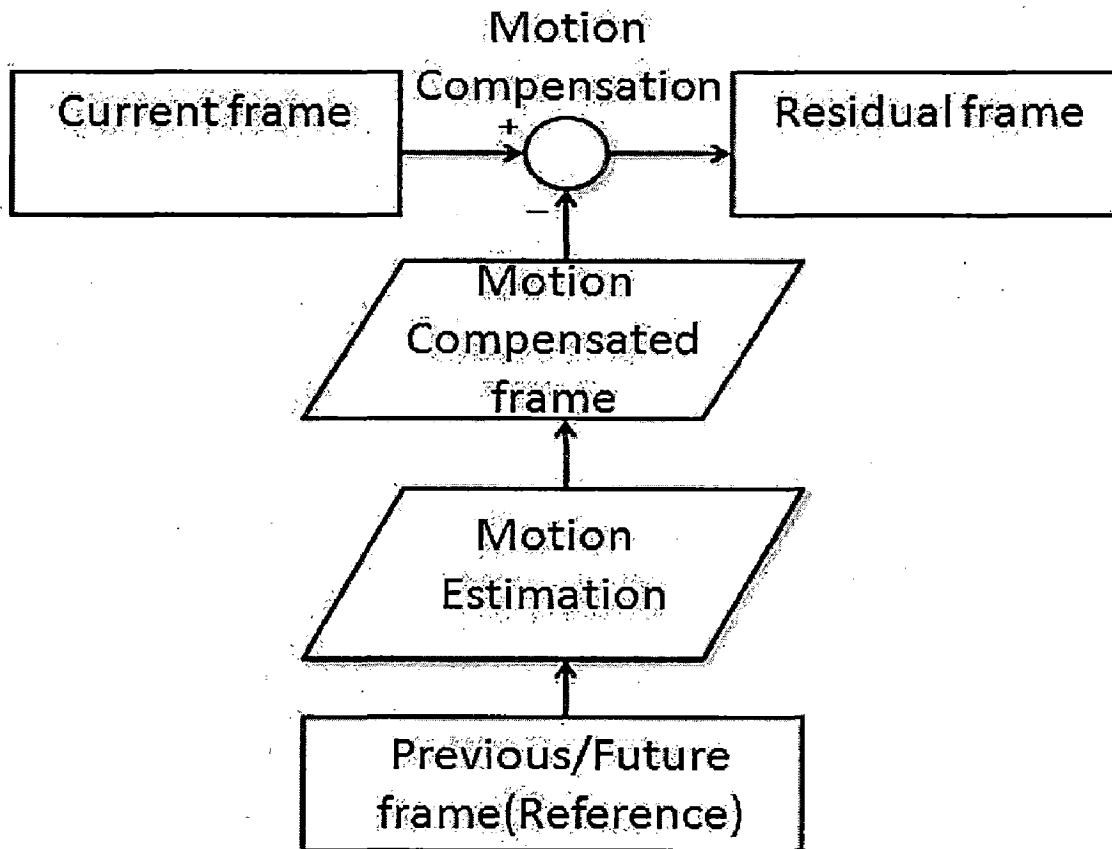


Figure 2.2: Motion-compensated predictions

The *macroblock*, typically corresponding to a 16×16 -pixel region of a frame, is the basic unit for motion compensated prediction. For source video material in 4:2:0 format, a macroblock is organized as shown in Figure 2.3. A 16×16 -pixel region of the source frame is represented by 256 luminance samples (arranged in four 8×8 -sample blocks), 64 blue chrominance samples (one 8×8 block) and 64 red chrominance samples (8×8), giving a total of six 8×8 blocks. An H.264/AVC CODEC processes each video frame in units of a macro block.

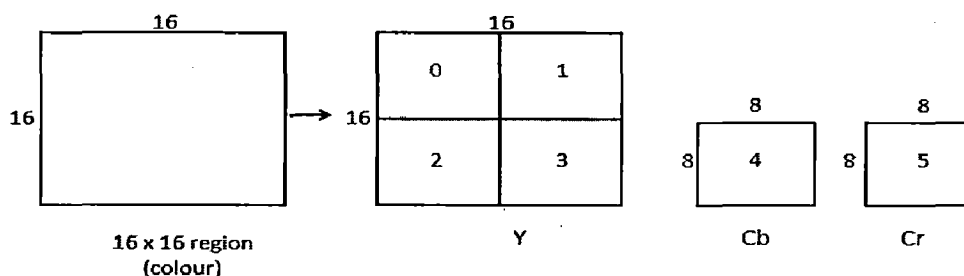


Figure 2.3: Macroblock (4:2:0)

Block matching as illustrated in Figure 2.4 is used to estimate the motion between the current frame and a reference frame. To start with, the current frame is divided into many small square blocks of equal size. (Blocks of size 16×16 are commonly used). Subsequently, following steps carried out by video encoder in motion-compensated prediction [8].

1. The energy of the difference between the current block and a set of neighbouring regions in the reference frame are calculated.
2. The region that gives the lowest error (the 'matching region') is selected.
3. The matching region from the current block is subtracted to produce a residual block
4. The residual block is encode and transmitted.
5. Corresponding 'motion vector' that indicates the position of the matching region relative to the current block position is encoded and transmitted.

Steps 1 and 2 above correspond to *motion estimation* and Step 3 correspond to *motion compensation*.

The video decoder reconstructs the block as follows:

1. The residual block and motion vector are decoded.
2. The residual block is added to the matching region in the reference frame (i.e. the region pointed by the motion vector).

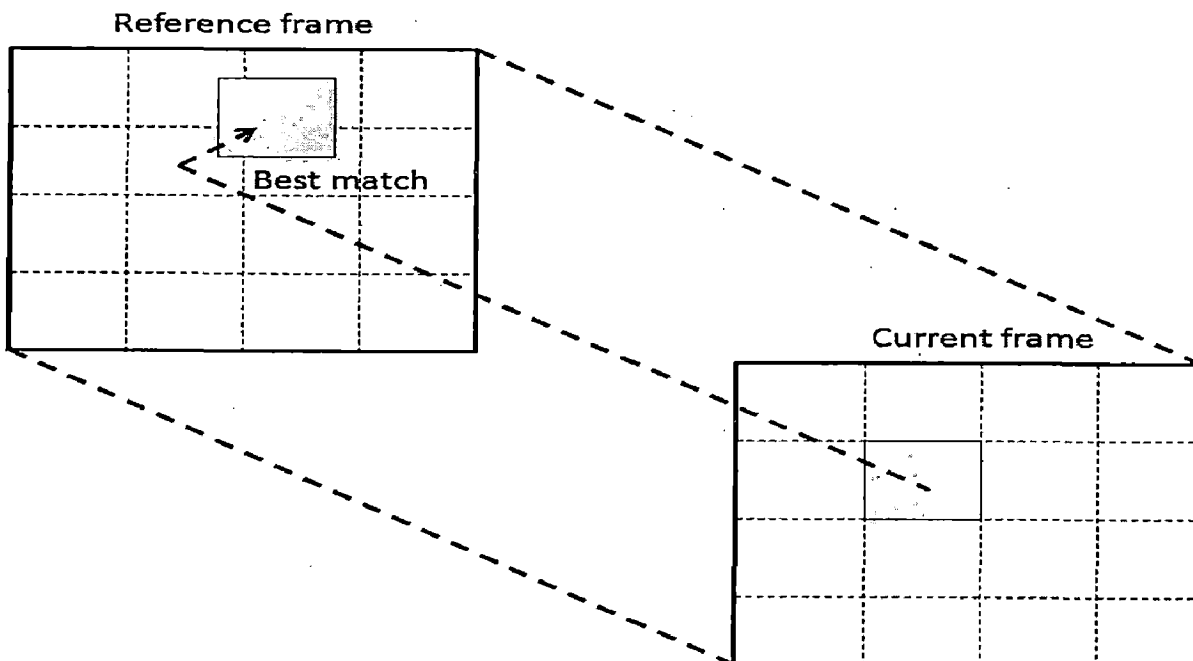


Figure 2.4: Block matching

2.1.2 Spatial Module

The residual frame (i.e. essentially the difference between the current frame and the corresponding motion compensated predicted frame) forms the input to the spatial module which makes use of similarities between neighbouring samples in the residual frame to reduce spatial redundancy. In H.264, this is achieved by applying a transform to the residual samples and quantizing the results. The transform converts the samples into another domain in which they are represented by transform coefficients. The coefficients are quantized to remove insignificant values, leaving only a small number of significant coefficients that provide a more compact representation of the residual frame. Thus, the output of the spatial module is a set of quantized transform coefficients.

The two most widely used image and video compression transforms are the discrete cosine transform (DCT) and the discrete wavelet transform (DWT). The DCT is usually applied to small, regular blocks of image samples (e.g. 8 x 8 squares) and the DWT is usually applied to larger image sections ('tiles') or to a complete image. In this dissertation, only the former is described in detail as it is exclusively employed in our work. There are two main reasons for its popularity: first, it is effective at transforming image data into a form that is easy to compress and second, it can be efficiently implemented in software and hardware [8].

The forward DCT (FDCT) transforms a set of image samples (the 'spatial domain') into a set of transform coefficients (the 'transform domain'). The transform is reversible; that is the inverse DCT (IDCT) transforms a set of coefficients into a set of image samples. The forward and inverse transforms are commonly used in 1-D or 2-D forms for image and video compression. The 1-D version transforms a 1-D array of samples into a 1-D array of coefficients, whereas the 2-D version transforms a 2-D array (block) of samples into a block of Coefficients.

The DCT has two useful properties for image and video compression, namely, energy compaction (concentrating the image energy into a small number of coefficients) and de-

-tization stage removes less important information (i.e. information that does not have a significant influence on the appearance of the reconstructed image), making it possible to compress remaining data.

The quantization process is split into two parts, an operation in the encoder that converts transform coefficients into levels (usually simply described as quantization) and an operation in the decoder that converts levels into reconstructed transform coefficients (usually described as rescaling or 'inverse quantization'). This is depicted in Figure 2.6 below.

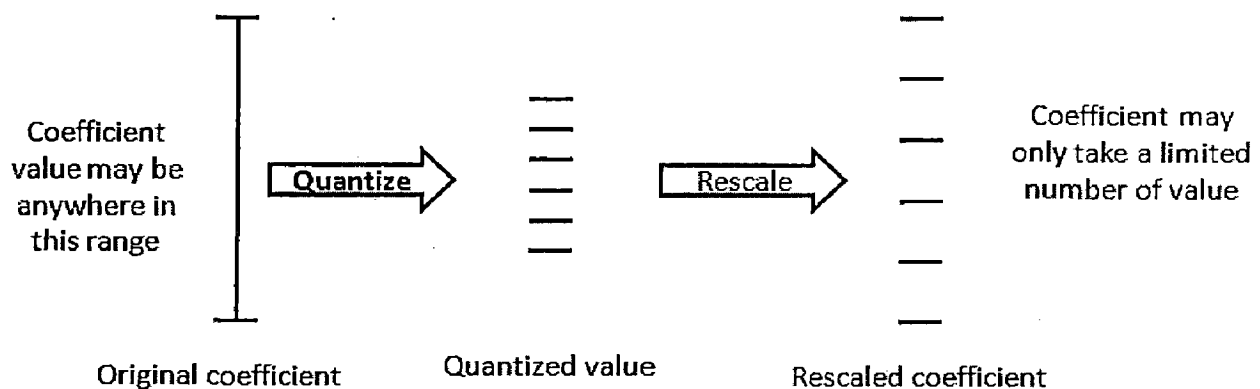


Figure 2.6: Quantization and rescaling

Quantized transform coefficients are required to be encoded as compactly as possible prior to storage and transmission. In a transform-based image or video encoder, the output of the quantizer is a sparse array containing a few nonzero coefficients and a large number of zero-valued coefficients. Hence, after quantization, the DCT coefficients for a block are reordered to group together nonzero coefficients, enabling efficient representation of the remaining zero-valued quantized coefficients. The optimum reordering path (scan order) depends on the distribution of nonzero DCT coefficients. For a typical frame block with a distribution similar to Figure 2.5 (b), a suitable scan order is a zigzag starting from the DC (top-left) coefficient. Starting with the DC coefficient, each quantized coefficient is copied into a one-dimensional array in the order shown in Figure 2.7. Nonzero coefficients tend to be grouped together at the start of the reordered array, followed by long sequences of zeros.

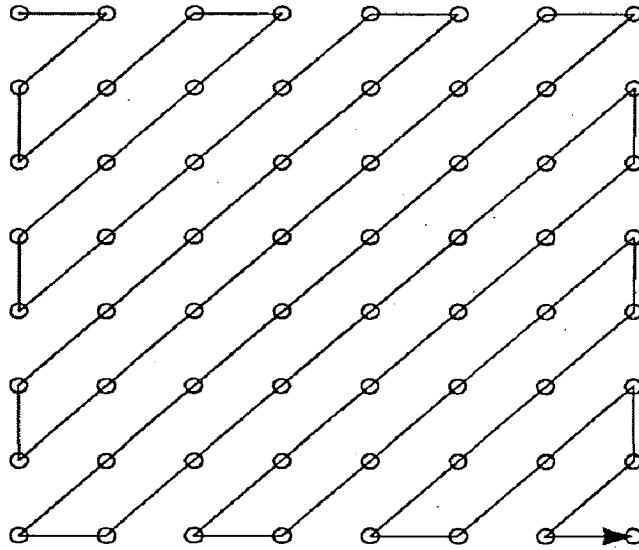


Figure 2.7: Zigzag scan

The output of the reordering process is an array that typically contains one or more clusters of nonzero coefficients near the start, followed by strings of zero coefficients. This large number of zero values may be encoded to represent them more compactly, for example by representing the array as a series of (run, level) pairs where *run* indicates the number of zeros preceding a nonzero coefficient and *level* indicates the magnitude of the nonzero coefficient. For example,

Input array: 16, 0, 0, -3, 5, 6, 0, 0, 0, 0, -7, ...

Output values: (0, 16), (2, -3), (0, 5), (0, 6), (4, -7), ...

A special case is required to indicate the final nonzero coefficient in a block. In so-called ‘Two-dimensional’ run-level encoding, each run-level pair is encoded as above and a separate code symbol, ‘last’, indicates the end of the nonzero values. If ‘Three-dimensional’ run-level encoding is used, each symbol encodes three quantities, run, level and last. In the example above, if -7 is the final nonzero coefficient, the 3D values are:

(0, 16, 0), (2, -3, 0), (0, 5, 0), (0, 6, 0), (4, -7, 1)

The 1 in the final code indicates that this is the last nonzero coefficient in the block.

2.1.3 Entropy Encoder

The parameters of the temporal module (typically motion vectors) and the spatial module (quantized transform coefficients) are compressed by the entropy encoder. An entropy encoder maps input symbols (for example, run-level coded coefficients) to a compressed data stream. It achieves compression by exploiting redundancy in the set of input symbols, representing frequently occurring symbols with a small number of bits and infrequently occurring symbols with a larger number of bits. A compressed sequence consists of coded motion vector parameters, coded residual coefficients and header information.

The two most popular entropy encoding methods used in video coding standards are Huffman coding and arithmetic coding. Huffman coding (or 'modified' Huffman coding) represents each input symbol by a variable-length codeword containing an integral number of bits. It is relatively straightforward to implement, but cannot achieve optimal compression because of the restriction that each codeword must contain an integral number of bits. On the other hand, Arithmetic coding maps an input symbol into a fractional number of bits, enabling greater compression efficiency at the expense of higher complexity (depending on the implementation) [9].

Based on the above discussion, the functional diagram of video encoder and decoder may be given by Figure 2.8 and Figure 2.9, respectively. A compressed sequence consists of coded motion vector parameters, coded residual coefficients and header information. The video decoder reconstructs a video frame from the compressed bit stream. The coefficients and motion vectors are decoded by an entropy decoder after which the spatial model is decoded to reconstruct a version of the residual frame. The decoder uses the motion vector parameters, together with one or more previously decoded frames, to create a prediction of the current frame and the frame itself is reconstructed by adding the residual frame to this prediction. In the next chapter, The H.264/AVC video coding standard is described in details, the focus being on those features that are relevant for this dissertation.

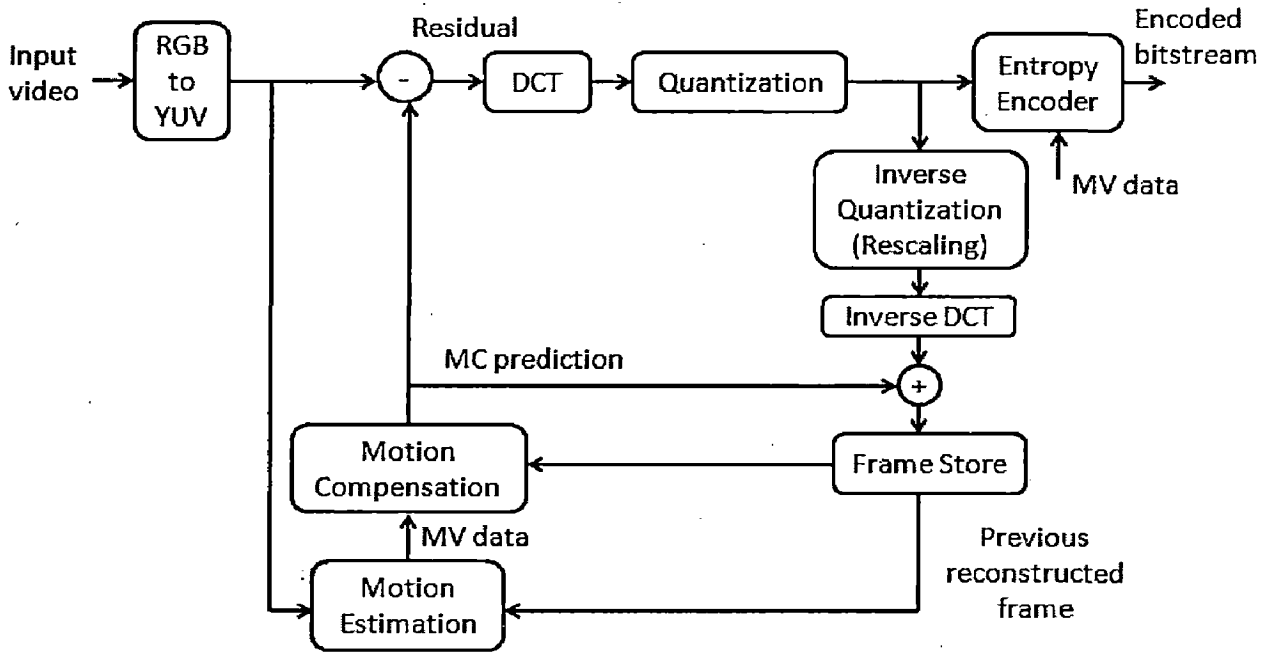


Figure 2.8: Functional diagram of video encoder

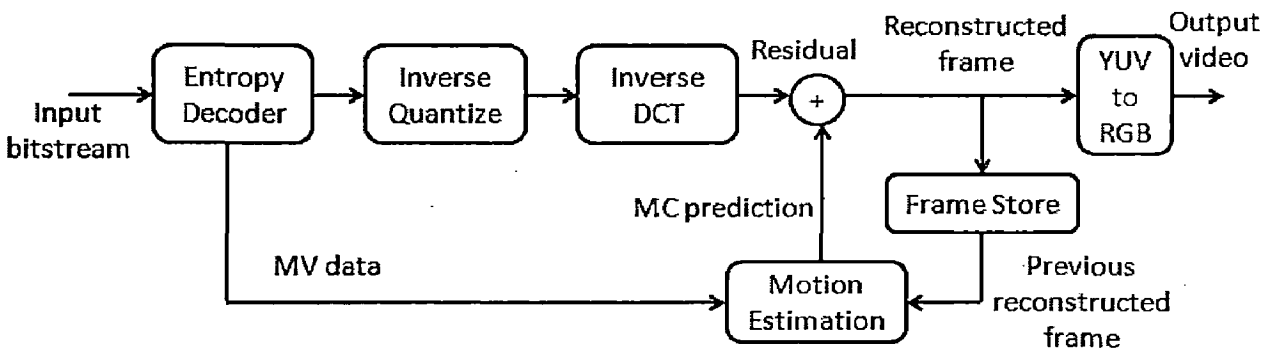


Figure 2.9: Functional diagram of video decoder

Chapter 3

THE H.264/AVC VIDEO CODING STANDARD

Video transmission has become a major application for the wireless systems, such as Wireless Local Area Networks (WLAN), Third Generation (3G) cellular networks etc. The wireless environment is very error prone due to the effect of fading, shadowing, interference etc. and also due to limitation in the radio resources such as transmission bandwidth and power are limited. Hence compression efficiency and error resilience are the main requirements for video coding standard. H.264/AVC was introduced with significant enhancement both in compression efficiency and error resilience compared to some other former video coding standards such as MPEG2 and MPEG4.

The H.264/AVC [4] standard was developed by the Joint Video Team (JVT) of the Video Coding Experts Group (VCEG) of the Telecommunications Standardization Sector of International Telecommunication Union (ITU-T) and the Moving Picture Experts Group (MPEG) of International Standardization Organization (ISO) / International Electro technical (IEC) Commission, which achieves better compression than all other previously existing video coding standards [9]. This is depicted in Figure 3.1.

Apart from better coding efficiency, the standard has also given strong emphasis to error resiliency and adaptability to various networks [4]. To give consideration to both coding efficiency and network friendliness, H.264/AVC has adopted a two-layer structure design: a video coding layer (VCL), which is designed to obtain highly compressed video data, and a network abstraction layer (NAL), which formats the VCL data and adds corresponding header information for adaptation to various transportation protocols or storage media. This is depicted in Figure 3.2.

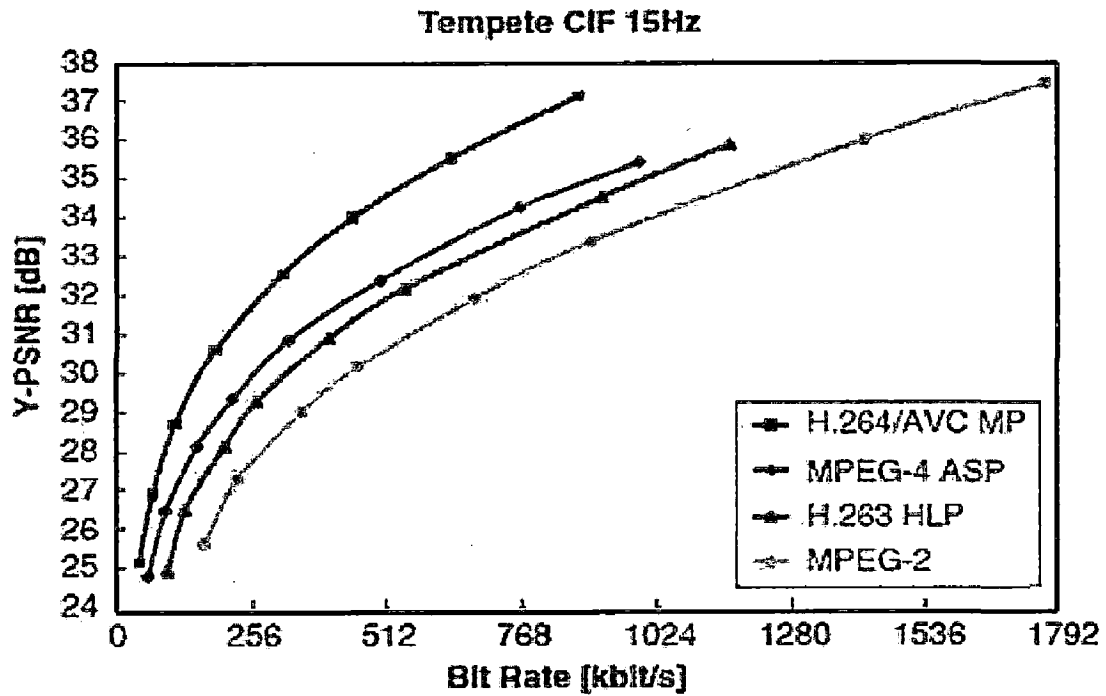


Figure 3.1: Luminance PSNR versus average bitrate for different video coding standard (Figure from [9])

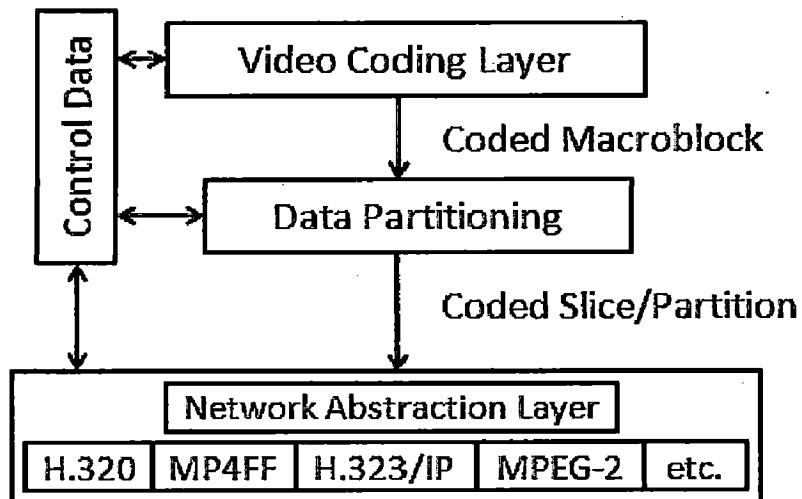


Figure 3.2: Structure of H.264/AVC video encoder [4]

The H.264/AVC standard is designed for technical solutions including at least the following application areas [4]:

- ❖ Broadcast over cable, satellite, cable modem, DSL, terrestrial, etc.
- ❖ Interactive or serial storage on optical and magnetic devices, DVD, etc.

- ❖ Conversational services over ISDN, Ethernet, LAN, DSL, wireless and mobile networks, modems, etc. or mixtures of these.
- ❖ Video-on-demand or multimedia streaming services over ISDN, cable modem, DSL, LAN, wireless networks, etc.
- ❖ Multimedia messaging services (MMS) over ISDN, DSL, Ethernet, LAN, wireless and mobile networks, etc.

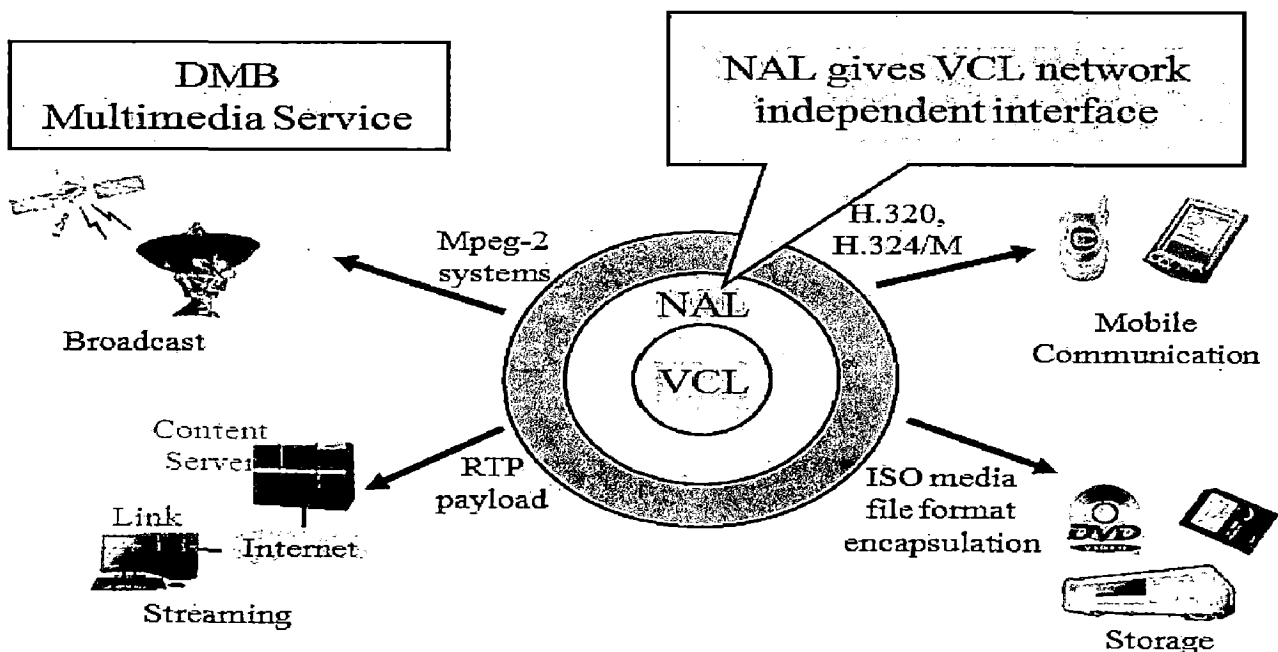


Figure 3.3: Application areas of H.264/AVC

3.1 The H.264/AVC Codec-Video Coding Layer

A coded video sequence in H.264/AVC consists of a sequence of coded frames [10]. A coded frame can represent either an entire frame or a single field, as was also the case for MPEG2 video. Generally, a frame of video is considered to contain two interleaved fields: a top field and a bottom field. If the two fields of a frame are captured at different time instants, the frame is referred to as an interlaced-scan frame; otherwise, it is referred to as a progressive-scan frame. The typical encoding operation for a frame begins with splitting the frame into blocks of samples as shown in Figure 3.4. The first frame of a sequence or a random access point is typically coded in Intra (intra-frame) mode (i.e., without using any other frames as prediction reference). Each sample of a block in such

an Intra-frame is predicted using spatially neighboring samples of previously coded blocks. The encoding process chooses which neighboring samples are to be used for Intra prediction and how these samples are to be combined to form a good prediction, and sends an indication of its selection to the decoder

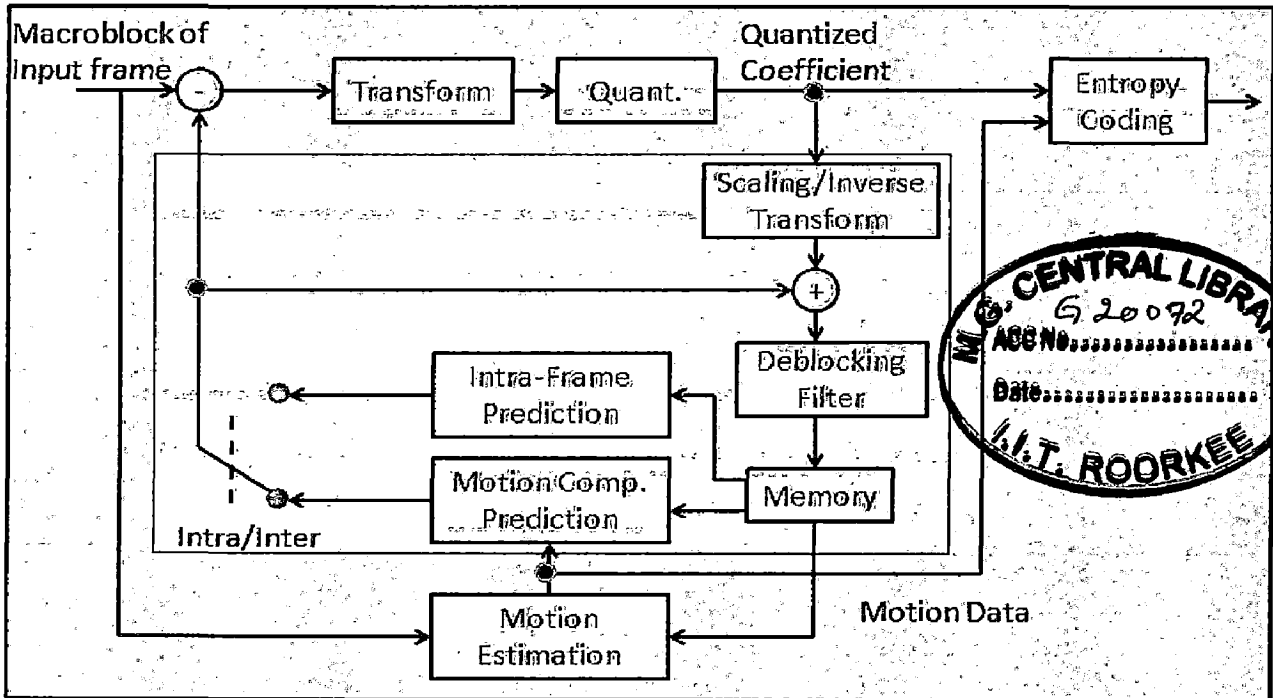


Figure 3.4: Typical encoder of H.264/AVC [9]

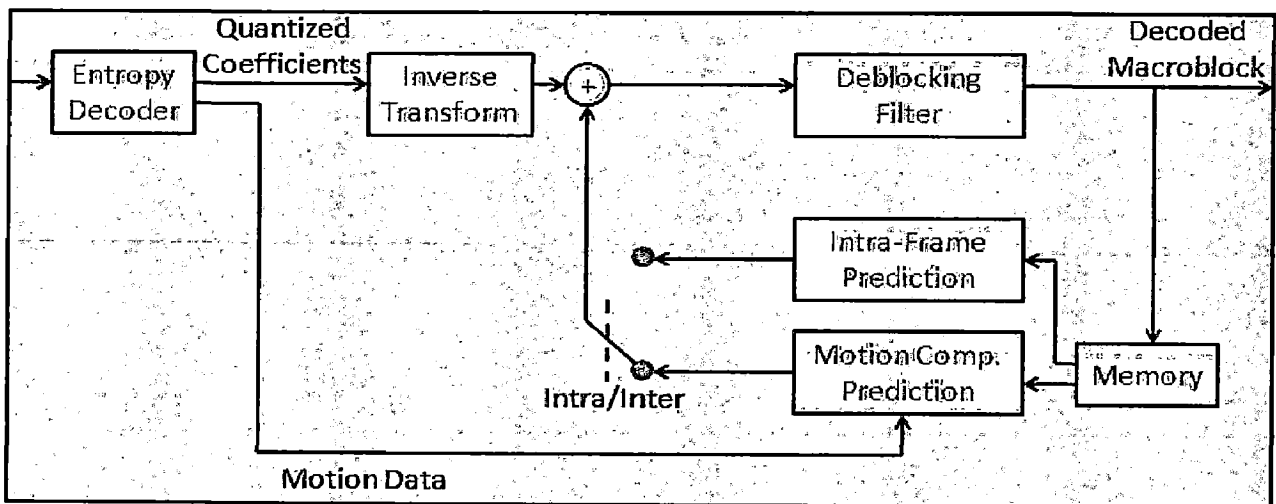


Figure 3.5: Typical decoder of H.264/AVC [9]

For all the remaining frames of a sequence or in between random access points, typically Inter (inter-frame) coding is utilized. Inter coding employs inter-frame temporal prediction (motion compensation) using other previously decoded frames. The encoding process for temporal prediction includes choosing motion data that identifies the reference frames and spatial displacement vectors that are applied to predict the samples of each block.

The residual of the prediction (either Intra or Inter), which is the difference between the original input samples and the predicted samples for the block, is transformed. The transform coefficients are then scaled and approximated using scalar quantization. The quantized transform coefficients are entropy coded and transmitted together with the entropy-coded prediction information for either Intra- or Inter-frame prediction. The encoder contains a model of the decoding process (shown as the shaded part of the block diagram in Figure 3.4) so that it can compute the same prediction values computed in the decoder for the prediction of subsequent blocks in the current frame or subsequent coded frames.

The decoder shown in Figure 3.5 inverts the entropy coding processes, performs the prediction process as indicated by the encoder using the prediction type information and motion data. It also inverse-scales and inverse-transforms the quantized transform coefficients to form the approximated residual and adds this to the prediction. The result of that addition is then fed into a deblocking filter, which provides the decoded video as its output.

The main features of H.264/AVC, which distinguish it from the previous video compression standards, are briefly discussed below [4] [9]

- ❖ Two context adaptive coding schemes, context-adaptive variable-length coding (CAVLC) and context-adaptive binary arithmetic coding (CABAC), improve coding efficiency by adjusting the code tables according to the surrounding information.

- ❖ B-frame may be used as reference frame.
- ❖ Multiple reference frame motion compensation as shown in Figure 3.6 is allowed, which also improves the prediction accuracy. The restriction that only the immediate previous frame can be used as reference frame is thus removed.

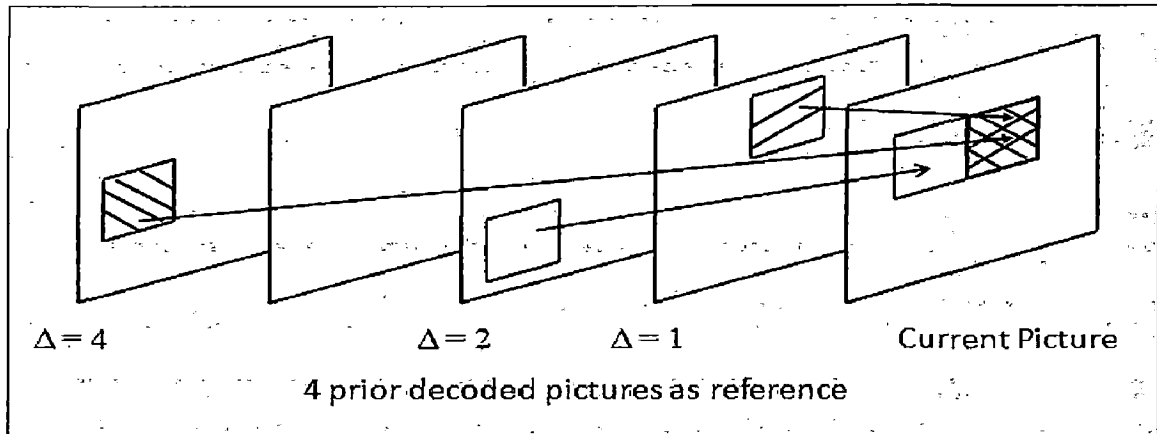


Figure 3.6: Multi-frame motion compensation. In addition to the motion vector, frame reference parameters (Δ) are also transmitted.

- ❖ More motion compensation block sizes and shapes, such as 8×4 , 4×8 , and 4×4 , are supported. The minimum luma motion compensation block size can be as small as 4×4 .
- ❖ Directional spatial prediction is applied in intra-coded macroblocks (MBs) of frames to reduce the amount of information before their block transform.
- ❖ In-loop deblocking filtering removes the blocking artifacts caused by transform and quantization.
- ❖ Small block-size transform of 4×4 is used rather than 8×8 used in earlier standards, which results in less ringing artifacts.
- ❖ The Discrete Cosine Transform (DCT) is replaced by integer transform that is exact-match inverse transform, thus avoiding drift during inverse transform.
- ❖ Parameter sets are used between the encoder and decoder to achieve synchronization in terms of syntax.
- ❖ Flexible macroblock ordering (FMO) partitions a frame into different slice groups.
- ❖ Data partitioning groups a slice in up to three packets by their importance.

- ❖ SP/SI-synchronization/switching frames reduce the penalty of switching between ongoing video bit-streams by avoiding transmission of an I-frame.
- ❖ Encoder can send redundant representation of some regions of a frame to enhance robustness to data loss

3.1.1. Subdivision of Picture into Macroblocks

Each frame of a video sequence is partitioned into fixed size macroblocks each of which cover a rectangular frame area of 16×16 samples of the luma component and, in the case of video with 4:2:0 chroma sampling format, 8×8 samples of each of the two chroma components. All luma and chroma samples of a macroblock are either spatially or temporally predicted, and the resulting prediction residual is represented using transform coding. Each color component of the prediction residual is subdivided into blocks. Each block is transformed using an integer transform, and the transform coefficients are quantized and entropy coded. The macroblocks are organized in slices, which generally represent subsets of a given frame that can be decoded independent of each other.

H.264/AVC supports five different slice-coding types. The simplest one is the I slice (where “I” stands for intra). In I slice, all macroblocks are coded without referring to other frames within the video sequence. On the other hand, prior-coded images can be used to form a prediction signal for macroblocks of the predictive-coded P and B slices (where “P” stands for predictive and “B” stands for bi-predictive).

The remaining two slice types are SP (switching P) and SI (switching I), which are specified for efficient switching between bit streams coded at various bit-rates. The Inter prediction signals of the bit streams for one selected SP frame are quantized in the transform domain, forcing them into a coarser range of amplitudes. This coarser range of amplitudes permits a low bit-rate coding of the difference signal between the bit streams. SI frames are specified to achieve a perfect match for SP frames in cases where Inter prediction cannot be used because of transmission errors.

3.1.2 Intra Prediction

Intra prediction means that the samples of a macroblock are predicted by using only information of already transmitted macroblocks of the same image. In H.264/AVC, two different types of intra prediction are possible for the prediction of the luminance component Y.

The first type is called INTRA_4×4 and the second one INTRA_16×16. Using the INTRA_4×4 type, the macroblock, which is of the size 16×16 frame elements (16×16), is divided into sixteen 4 ×4 sub-blocks and a prediction for each 4×4 sub-block of the luminance signal is applied individually. For the prediction purpose, nine different prediction modes are supported. One mode is DC-prediction mode, whereas all samples of the current 4×4 sub-block are predicted by the mean of all samples neighboring to the left and to the top of the current block and that have been already reconstructed at the encoder and at the decoder side (Figure 3.7, Mode 2). In addition to DC-prediction mode, eight prediction modes each for a specific prediction direction are supported. All possible modes are shown in Figure 3.7.

Using the type INTRA_16×16, only one prediction mode is applied for the whole macroblock. Four different prediction modes are supported for the type INTRA_16×16: Vertical prediction, horizontal prediction, DC-prediction and plane-prediction. Plane-prediction uses a linear function between the neighboring samples to the left and to the top in order to predict the current samples. This mode works very well in areas of a gently changing luminance. The mode of operation of these modes is the same as the one of the 4×4 prediction modes. The only difference is that they are applied for the whole macroblock instead of a 4×4 sub-block. The efficiency of these modes is high if the signal is very smooth within the macroblock. The intra prediction for the chrominance signals Cb and Cr of a macroblock is similar to the INTRA_16×16 type for the luminance signal because the chrominance signals are very smooth in most cases. It is performed always on 8×8 blocks using vertical prediction, horizontal prediction, DC-prediction or plane-prediction.

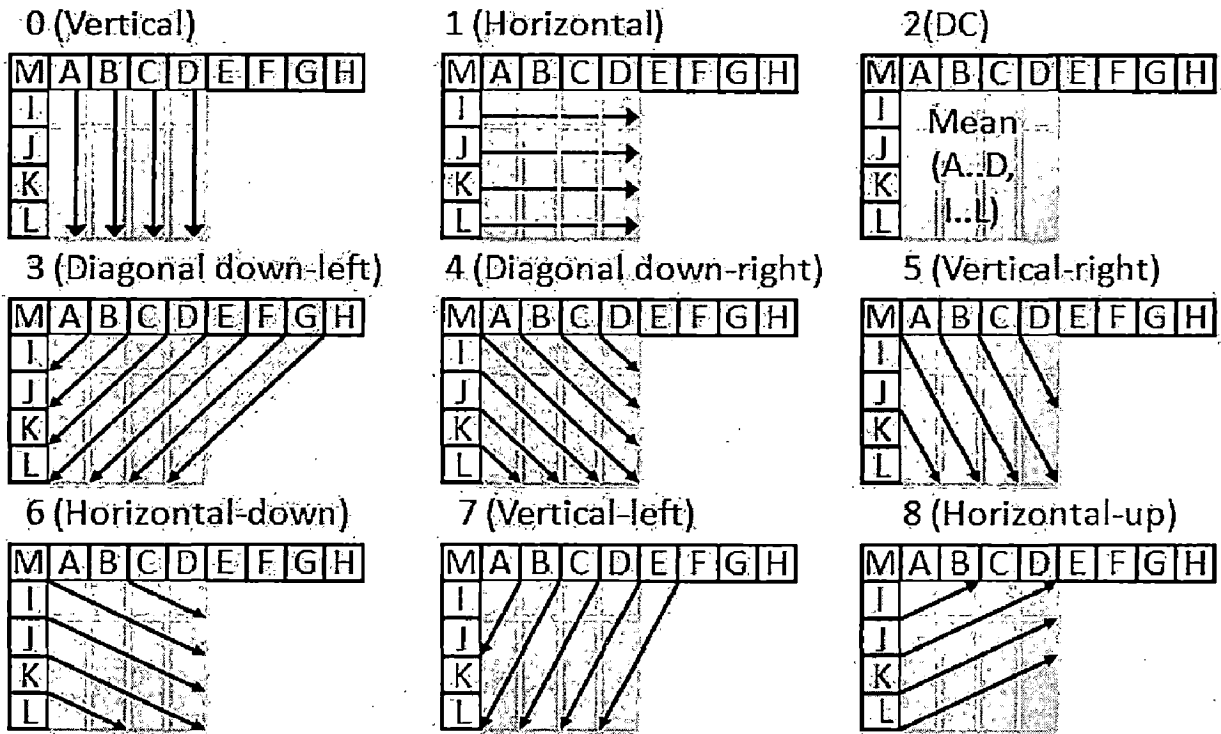


Figure 3.7: 4 x 4 Luma prediction modes

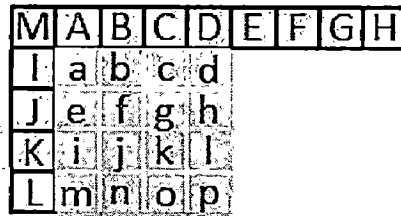


Figure 3.8: Labelling of prediction sample (4 x 4)

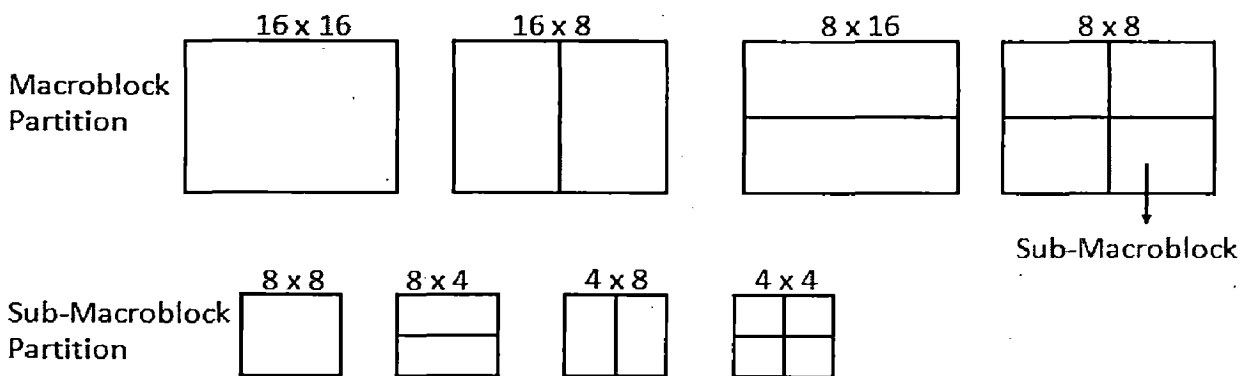


Figure 3.9: Partitioning of Macro block (top) and sub-macro block (bottom) for motion-compensated prediction

3.1.3 Motion Compensated Prediction

In case of motion compensated prediction macroblocks are predicted from the already transmitted reference images [refer section 2.2.1]. For this purpose, each macroblock may be divided into smaller partitions. Partitions with luminance block sizes of 16×16 , 16×8 , 8×16 , and 8×8 samples are supported. In case of an 8×8 sub-macroblock in a P-slice, one additional syntax element specifies if the corresponding 8×8 sub-macroblock is further divided into partitions with block size of 8×4 , 4×8 or 4×4 (Figure 3.9). In former standards as MPEG-4 or H.263 [8], only blocks of the size 16×16 and 8×8 are supported and the reference image is the most recent preceding image. In H.264/AVC it is possible to refer to several preceding images as shown in Figure 3.6. For this purpose, an additional frame reference parameter needs to be transmitted together with motion vector.

3.1.4 Transform Coding

The task of the transform is to reduce the spatial redundancy of the prediction error signal. For the purpose of transform coding, all former standards such as MPEG-1 and MPEG-2 apply a two dimensional Discrete Cosine Transform (DCT) of the size 8×8 [11]. Instead of the DCT, different integer transforms are applied in H.264/AVC. The size of these transforms is mainly 4×4 , and in special cases is 2×2 . This smaller block size of 4×4 enables the encoder to better adapt the prediction error coding to the boundaries of moving objects, to match the transform block size with the smallest block size of the motion compensation, and to generally better adapt the transform to the local prediction error signal.

Three different types of transforms are used. The first type, an Integer transform, is applied to all samples of all prediction error blocks of the luminance component Y and also for all blocks of both chrominance components Cb and Cr regardless of whether

motion compensated prediction or intra prediction was used. The size of this transform is 4×4 . Its transform matrix H_1 is given in Figure 3.10.

If the macroblock is predicted using the type INTRA_16x16, the second transform, a Hadamard transform with matrix H_2 (see Figure 3.10), is applied in addition to the first one. It transforms all 16 DC coefficients of the already transformed blocks of the luminance signal. The size of this transform is also 4×4 .

The third transform is also a Hadamard transform but of size 2×2 . It is used for the transform of the 4 DC coefficients of each chrominance component. Its matrix H_3 is also shown in Figure 3.10

$$H_1 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \quad H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \quad H_3 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Figure 3.10: Matrices H_1 , H_2 and H_3 of the three different transforms applied in H.264/AVC

3.1.5 Entropy Coding Schemes

H.264/AVC specifies two alternative methods of entropy coding: a low-complexity technique based on the usage of context-adaptive switched sets of variable length codes, (CAVLC), and the computationally more demanding algorithm of context-based adaptive binary arithmetic coding (CABAC). Both methods represent major improvements in terms of coding efficiency compared to the techniques of statistical coding traditionally used in prior video coding standards [4].

When using the first entropy-coding configuration, which is intended for lower complexi-

-ty implementations, the exponential-Golomb code is used for nearly all syntax elements except those of quantized transform coefficients, for which a more sophisticated method called context-adaptive variable length coding (CAVLC) is employed. When using CAVLC [4], the encoder switches between different VLC tables for various syntax elements, depending on the values of the previously transmitted Syntax elements in the same slice. Since the VLC tables are designed to match the conditional probabilities of the context, the entropy coding performance is improved from that of schemes that do not use context-based adaptivity.

The entropy coding performance is further improved if the second configuration is used, this is referred to as context-based adaptive binary arithmetic coding (CABAC) [10]. As shown in Figure 3.11, the CABAC design is based on three components: binarization, context modeling, and binary arithmetic coding. Binarization enables efficient binary arithmetic coding by mapping nonbinary syntax elements to sequences of bits referred to as bin-strings. Each bin of a bin-string can be processed in either an arithmetic coding mode or a bypass mode. The later is a simplified coding mode that is chosen for selected bins such as sign information or less-significance bins in order to speed up the overall decoding (and encoding) processes. The arithmetic coding mode provides the largest compression benefit, where a bin may be context-modeled and subsequently arithmetic encoded.

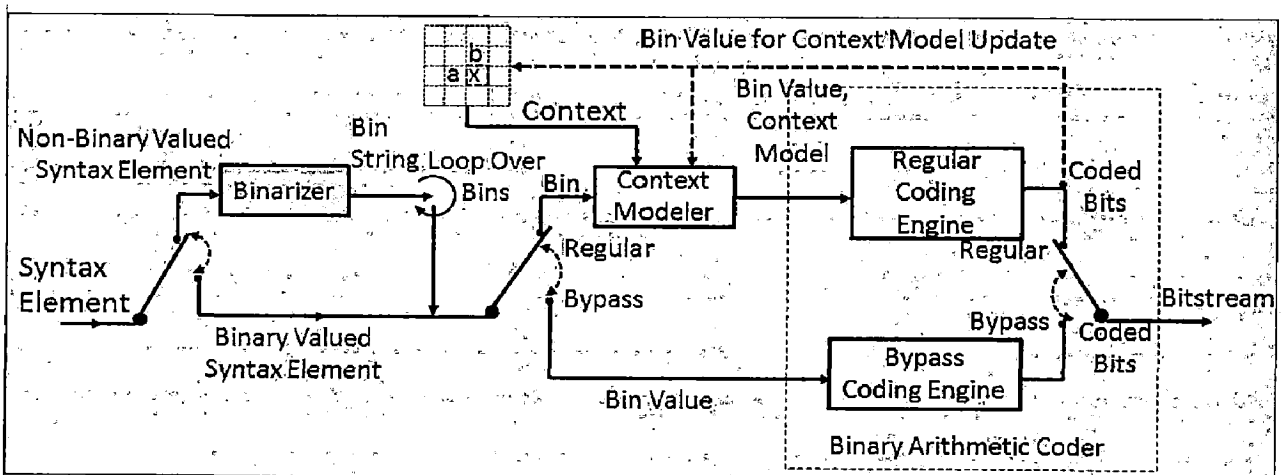


Figure 3.11: CABAC Block diagram

Compared to CAVLC, CABAC can typically provide reductions in bit rate of 10–20 percent for the same objective video quality when coding SDTV/HDTV signals.

3.1.6 In Loop Deblocking Filter

Due to coarse quantization at low bit rates, block-based coding typically results in visually noticeable discontinuities along the block boundaries. If no further provision is made to deal with this, these artificial discontinuities may also diffuse into the interior of blocks by means of the motion-compensated prediction process. The removal of such blocking artifacts provides a substantial improvement in perceptual quality. For that purpose, H.264/AVC defines a deblocking filter that operates within the predictive coding loop, and thus constitutes a required component of the decoding process [4]. The filtering process exhibits a high degree of content adaptivity on different levels, from the slice level along the edge level down to the level of individual samples. As a result, the blockiness is reduced without affecting the sharpness of the content much. Consequently, the subjective quality is significantly improved. At the same time, the filter reduces bit rate by typically 5–10 percent while producing the same objective quality as the non-filtered video.

3.2 Profiles and Levels

A profile consists of a subset of the algorithmic features or coding tools of the standard and a set of constraints on those features [4]. A profile is typically targeted for a family of applications sharing similar trade-offs between memory, processing, latency, and error resilience requirements. A level corresponds to a set of limits mainly on memory requirements and computational performance. Decoders conforming to a profile must support all the features of a profile, whereas encoders have the freedom to select which features of the profile are used to produce compliant bit-streams. A decoder conforming to a level must be capable of decoding any bit-stream that conforms to the level. In other words, levels give minimum requirements for decoders and constraints for

bit-streams and encoders. Figure 3.12 illustrates the three profile of H.264/AVC and their corresponding main features, as further discussed below.

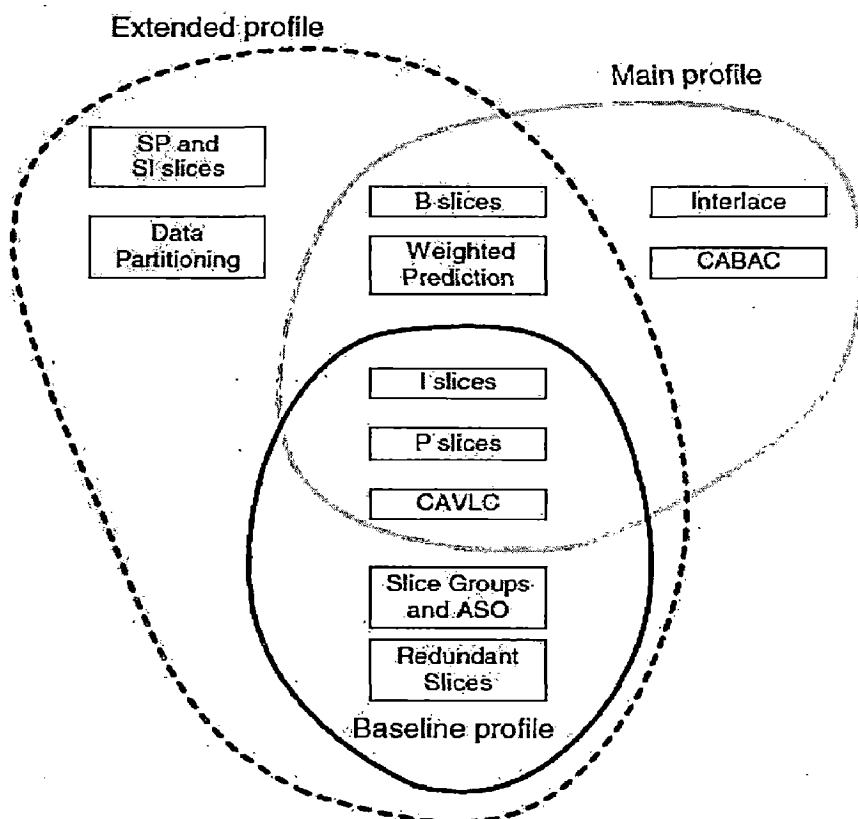


Figure 3.12: The H.264/AVC Baseline, Main and Extended profiles [9]

The Baseline Profile supports coded sequences containing I- and P-slices. In the Baseline Profile, transform coefficients are entropy coded using a context-adaptive variable length coding scheme (CAVLC) and all other syntax elements are coded using fixed-length or Exponential-Golomb Variable Length Codes [7]. Potential applications of the Baseline Profile include video-telephony, video-conferencing and wireless communications.

Suitable application for the Main Profile includes (but are not limited to) broadcast media applications such as digital television and stored digital video. The Main Profile is almost a superset of the Baseline Profile, except that multiple slice groups, ASO and redundant slices (all included in the Baseline Profile) are not supported. The additional tools provided by Main Profile are B slices (bi-predicted slices for greater coding efficiency),

weighted prediction (providing increased flexibility in creating a motion-compensated prediction block), support for interlaced video (coding of fields as well as frames) and CABAC (an alternative entropy coding method based on Arithmetic Coding).

The Extended Profile may be particularly useful for applications such as video streaming. It includes all of the features of the Baseline Profile (i.e. it is a superset of the Baseline Profile, unlike Main Profile), together with B-slices, Weighted Prediction and additional features to support efficient streaming over networks such as the Internet. SP and SI slices facilitate switching between different coded streams and Data Partitioned slices can provide improved performance in error-prone transmission environments.

3.3 Network Abstraction Layer

The NAL is designed in order to provide “network friendliness” to enable simple and effective customization of the use of the VCL for a broad variety of systems. The NAL facilitates the ability to map H.264/AVC VCL data to transport layers such as [4]:

- RTP/IP for any kind of real-time wire-line and wireless Internet services (conversational and streaming);
- File formats, e.g., ISO MP4 for storage and MMS;
- H.32X for wireline and wireless conversational services;
- MPEG-2 systems for broadcasting services, etc.

A short description of some of key concepts of NAL is given below [11]

3.3.1 NAL Units

The coded video data is organized into NAL units, each of which is effectively a packet that contains an integer number of bytes. The first byte of each NAL unit is a header byte that contains an indication of the type of data in the NAL unit, and the remaining bytes contain payload data of the type indicated by the header. The payload data in the NAL unit is interleaved as necessary with emulation prevention bytes, which are bytes inserted

with a specific value to prevent a particular pattern of data called a start code prefix from being accidentally generated inside the payload.

3.3.2 NAL Units in Byte-Stream Format Use

Some systems require delivery of the entire or partial NAL unit stream as an ordered stream of bytes or bits within which the locations of NAL unit boundaries need to be identifiable from patterns within the coded data itself. For use in such systems, the H.264/AVC specification defines a byte stream format. In the byte stream format, each NAL unit is prefixed by a specific pattern of three bytes called a start code prefix. The boundaries of the NAL unit can then be identified by searching the coded data for the unique start code prefix pattern. The use of emulation prevention bytes guarantees that start code prefixes are unique identifiers of the start of a new NAL unit.

3.3.3 NAL Units in Packet-Transport System Use

In Internet protocol/RTP systems [12], the coded data is carried in packets that are framed by the system transport protocol, and identification of the boundaries of NAL units within the packets can be established without use of start code prefix patterns. In such systems, the inclusion of start code prefixes in the data would be a waste of data carrying capacity. So the NAL units can be carried in data packets without start code prefixes.

3.3.4 VCL and Non-VCL NAL Units

NAL units are classified into VCL and non-VCL NAL units. The VCL NAL units contain the data that represents the values of the samples in the video frames, and the non-VCL NAL units contain any associated additional information such as parameter sets (important header data that can apply to a large number of VCL NAL units) and supplemental enhancement information (timing information and other supplemental data

that may enhance usability of the decoded video signal but are not necessary for decoding the values of the samples in the video frames).

3.3.5 Parameter Sets

A parameter set is supposed to contain information that is expected to rarely change and offers the decoding of a large number of VCL NAL units. There are two types of parameter sets:

- sequence parameter sets, which apply to a series of consecutive coded video frames called a coded video sequence;
- frame parameter sets, which apply to the decoding of one or more individual frames within a coded video sequence

The sequence and frame parameter-set mechanism decouples the transmission of infrequently changing information from the transmission of coded representations of the values of the samples in the video frames. Each VCL NAL unit contains an identifier that refers to the content of the relevant frame parameter set and each frame parameter set contains an identifier that refers to the content of the relevant sequence parameter set. In this manner, a small amount of data (the identifier) may be used to refer to a larger amount of information (the parameter set) without repeating that information within each VCL NAL unit.

3.4 Transport of H.264/AVC

A coded H.264 video sequence consists of a series of NAL units, each containing a Raw Byte Sequence Payload. Coded slices (including Data Partitioned slices and IDR slices) and the End of Sequence RBSP are defined as VCL NAL units while all other elements are just NAL units. A typical sequence of RBSP units is shown in Figure 3.13. Each of these units is transmitted in a separate NAL unit. The header of the NAL unit (one byte) signals the type of RBSP unit and the RBSP data makes up the rest of the NAL unit.

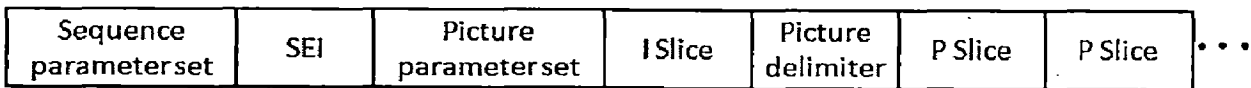


Figure 3.13: Sequence of RBSP

The method of transmitting NAL units is not specified in the standard but some distinction is made between transmission over packet-based transport mechanisms (e.g. packet networks) and transmission in a continuous data stream (e.g. circuit-switched channels). In a packet based network, each NAL unit may be carried in a separate packet and is organized into the correct sequence prior to decoding. In a circuit-switched transport environment, a start code prefix (a uniquely-identifiable delimiter code) is placed before each NAL unit to make a byte stream prior to transmission. This enables a decoder to search the stream to find a start code prefix identifying the start of a NAL unit.

On the transport layer, IP networks commonly employ two protocols [12]: the TCP and the UDP. Both include features common to transport protocols, such as application addressing through the port number, and error control for the payload. TCP offers a byte-oriented, guaranteed transport service, which is based on re-transmission and timeout mechanisms for error control. Due to its unpredictable delay characteristics it is not suitable for real-time communication. UDP, on the other hand, offers a simple, unreliable datagram transport service. The UDP header contains a checksum, which can be used to detect and remove packets containing bit errors—the mode of operation generally used. UDP header is 8 bytes in size. In addition, since UDP doesn't guarantee packet delivery, the receiver needs to rely on upper layer (i.e., RTP) to detect loss.

RTP is an Internet standard protocol designed to provide end-to-end transport functions for supporting real time application. RTP is typically used above IP/UDP. It is session oriented, and a session is associated with a transport address—the combination of the IP address and the UDP port number. Each RTP packet consists of an RTP header, optional payload headers, and the payload itself. The RTP header contains the following:

- The sequence number, which is incremented by one for each packet sent in a session and used for packet-loss detection.
- The timestamp that contains timing information relative to the establishment of the session. Timestamps are normally used to determine the precise moment for media reproduction, but also for purposes such as the synchronization of media streams carried in more than one session.
- The type of payload contained in an RTP packet is indicated by an RTP-header field called payload type identifier. The receiver interprets the content of the packet based on this field.
- The marker bit, which is normally set for the very last packet of a group of packets that have the same time stamp, e.g., belonging to the same video frame. It is employed to quickly detect the end of a group of related packets without having to wait for the next packet to arrive.

When transmitted over a mobile channel, video may be affected by a number of loss mechanisms, like multipath fading, shadowing, and co-channel interference. The effects of such errors are magnified due to the fact that the video bit-stream is highly compressed to meet the stringent bandwidth limitations. The higher the compression, the more sensitive the bit-stream is to errors, since in this case each bit represents a large amount of decoded video. The effects of errors are also magnified by the use of predictive and VLC coding, which can lead to temporal and spatial error propagation. It is therefore not difficult to realize that when transmitted over a mobile channel, compressed video can suffer severe degradation, and hence making the use of error-resilience techniques vital.

In the next chapter, we will focus on the error resiliency schemes adopted in H.264/AVC assuming a typical networking environment. The H.264/AVC error resiliency schemes includes flexible macroblock ordering, intra Placement, redundant slices etc., all will be described in details in next chapter.

Chapter 4

ERROR RESILIENCE TOOLS IN H.264/AVC

In multimedia communication systems, channel bandwidth and probability of error are the two main limitations that affect the quality of service. Therefore, in applications such as video over mobile networks, a video codec should cope with the erroneous situation of the channel as well as the bandwidth limitation. There is a number of tools in H.264/AVC video coding standard which allow robust transmission of compressed video data over error prone channels. In error resilient coding, the encoder operates in a way that transmission errors on the coded bit-stream will not adversely affect the decoder operation that leads to unacceptable distortion in the reconstructed video quality.

The error resiliency schemes in H.264/AVC are mainly contained in the VCL [13][14]. Some of these schemes, such as use of slices, data partitioning, and placement of intra-MBs have also been used in some previous video coding standards whereas H.264 introduces few additional new error resilience tools like parameter sets, flexible macroblock ordering and redundant slices. In this chapter, we focus on the error resilience techniques for video communication applications using H.264.

4.1 Error Resilience Features of H.264

Real-time transmission of video data in network environments, such as wireless and Internet, is a challenging task as it requires high compression efficiency and network-friendly design. Achieving a high compression efficiency and network friendliness have become the major goals for the H.264 standard. Apart from this, considering the lossy nature of the wireless environment, the standard should provide good error resilience

features. A brief description of the error resilience features of H.264 is presented in the section to follow:

4.1.1 Semantics, Syntax and Error Detection

The H.264/AVC video coding standard explicitly defines all the syntax elements, such as motion vectors, block coefficients, frame numbers, and the order they appear in the video bit-stream [13]. Syntax is the most important tool for ensuring compliance and error detection. Like other video coding standards, H.264/AVC only defines the syntax of the decoder to allow flexibility in specific implementation at the encoder. However, it provides no guarantee to end-to-end reproduction quality, as it allows even crude encoding techniques to be considered. Basically, a video bit stream corrupted by error(s) will incur syntax/semantics error(s). Due to the use of VLC, errors often propagate in the bit-stream until they are detected. The syntax/semantics errors may include

- Illegal value of syntax elements.
- Illegal sync header.
- More than 16 coefficients decoded in a 4×4 block.
- An incorrect number of stuffing bits found. This could also occur when extra bits remain after decoding all expected coefficients of the last coded block in a video packet.
- Some of the coded blocks in a video packet cannot be decoded.

4.1.2 Intra Placement

Intra placement on the macroblock, slice, or frame level, is used primarily to combat drifting effects. Its main features are [12]

- H.264 allows Intra macroblock prediction even from predictively coded (Inter) macroblocks. This feature is helpful for coding efficiency, but is counterproductive to the re-synchronization property of Intra coding. The Constrained Intra-Prediction Flag on the sequence level, when set, prevents this

the encoder and the decoder. In order to be used, they require the availability of the type A partition, but not the type B partition.

Because of the small amount of data in DP A, it can be transmitted through out-of-band channel. Compared to MPEG-4 in which all kinds of data are carried by one packet, the DPs in H.264 are more flexible in terms of unequal data protection.

When data partitioning is used, the source coder puts symbols of different types to three different bit buffers. Furthermore, the slice size must be adjusted in such a way that the biggest partition does not lead to a packet bigger than the maximum transportation unit size. For this reason, it is the source coder and not the NAL that has to implement data partitioning [12].

Table 4.1
Recommended actions when partition loss is detected [13]

Available partition(s)	Concealment method
A and B	Conceal using MVs from Partition A and texture from Partition B; intra concealment is optional.
A and C	Conceal using MVs from Partition A and inter info from Partition C; inter texture concealment is optional.
A	Conceal using MVs from Partition A
B and/or C	Drop Partitions B and C. Use motion vectors of the spatially above MB row for each lost MB

4.1.4 Slice Structuring: Flexible Macroblock Ordering

A slice group is a subset of the macroblocks in a coded frame and may contain one or more slices. Within each slice in a slice group, MBs are coded in raster order. If only one slice group is used per frame, then all macroblocks in the frame are coded in raster order. Multiple slice group (Flexible Macroblock Ordering) makes it possible to map the

sequence of coded MBs to the decoded frame in a number of flexible ways. The allocation of macroblocks is determined by the macroblock to slice group map that indicates which slice group each MB belongs to.

The H.264 encoder intelligently groups MBs into a slice whose size is less than (or equal to) the size of the maximum transportation unit (MTU). Here, the MTU represents the largest size of a packet that may be transported through networks without being split. Prediction beyond the slice boundaries is forbidden to prevent error propagation from intra-frame predictions. The slice structuring strategy thus aims at avoiding error propagation from a corrupted packet to subsequent packets. Figure 4.1 to figure 4.5 illus-

0
1
2
0
1
2
0
1
2

Figure 4.1: Slice Group: Interleaved map (three slice group)

0	1	2	3	0	1	2	3	0	1	2
2	3	0	1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3	0	1	2
2	3	0	1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3	0	1	2
2	3	0	1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3	0	1	2
2	3	0	1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3	0	1	2

Figure 4.2: Slice Group: Dispersed map (four slice group)

-trates the different types of macroblock to slice group maps supported by H.264/AVC. Slice groups are present in the Baseline and Extended profile and not in the Main Profile.

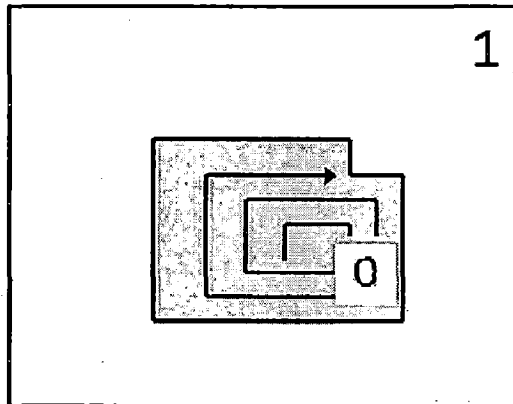


Figure 4.3: Slice Group: Box-out map

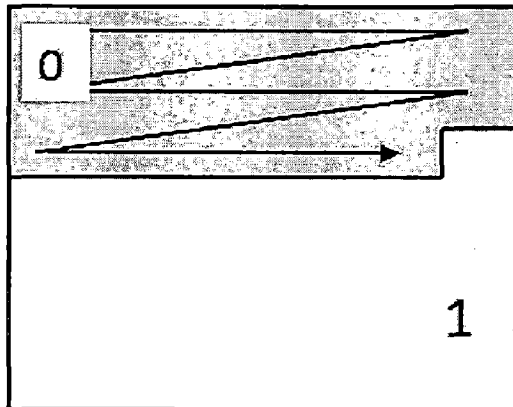


Figure 4.4: Slice Group: Raster map

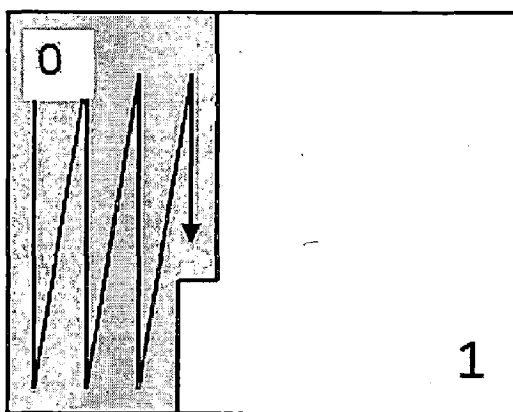


Figure 4.5: Slice Group: Wipe map

4.1.5 Redundant Slices

In order to enhance robustness to data loss, H.264/AVC design contains a new provision to allow an encoder to send redundant representations of regions of frames, enabling a (typically somewhat degraded) representation of regions of frames for which the primary representation has been lost during data transmission. The syntax and semantics of a redundant coded frame are identical to those of a primary coded frame. However, a redundant coded frame may contain a subset of the macroblocks of an entire frame, and different values of coding parameters, such as macroblock modes and reference frames, which can be used when compared to the primary coded frame. Decoders do not decode redundant coded frames when the corresponding primary coded frame is correctly received and can be correctly decoded. However, when a primary coded frame is lost or cannot be correctly decoded, a redundant frame is utilized to improve the decoded video quality.

In this chapter we introduce some error resilience tools used by H.264/AVC video coding standards. Usage of them is very useful if we transmit compressed bit-stream over error prone environments like wireless channels. In these cases, these error resilience tools can significantly improve decoded video quality.

Since the goal of this dissertation is to study the error resilience of H.264/AVC video coding standard. We present the simulation result (performance) of various error resilient tools used by H.264/AVC in the next chapter. We also analyze the effect of these tools, if any, on the compression performance.

Chapter 5

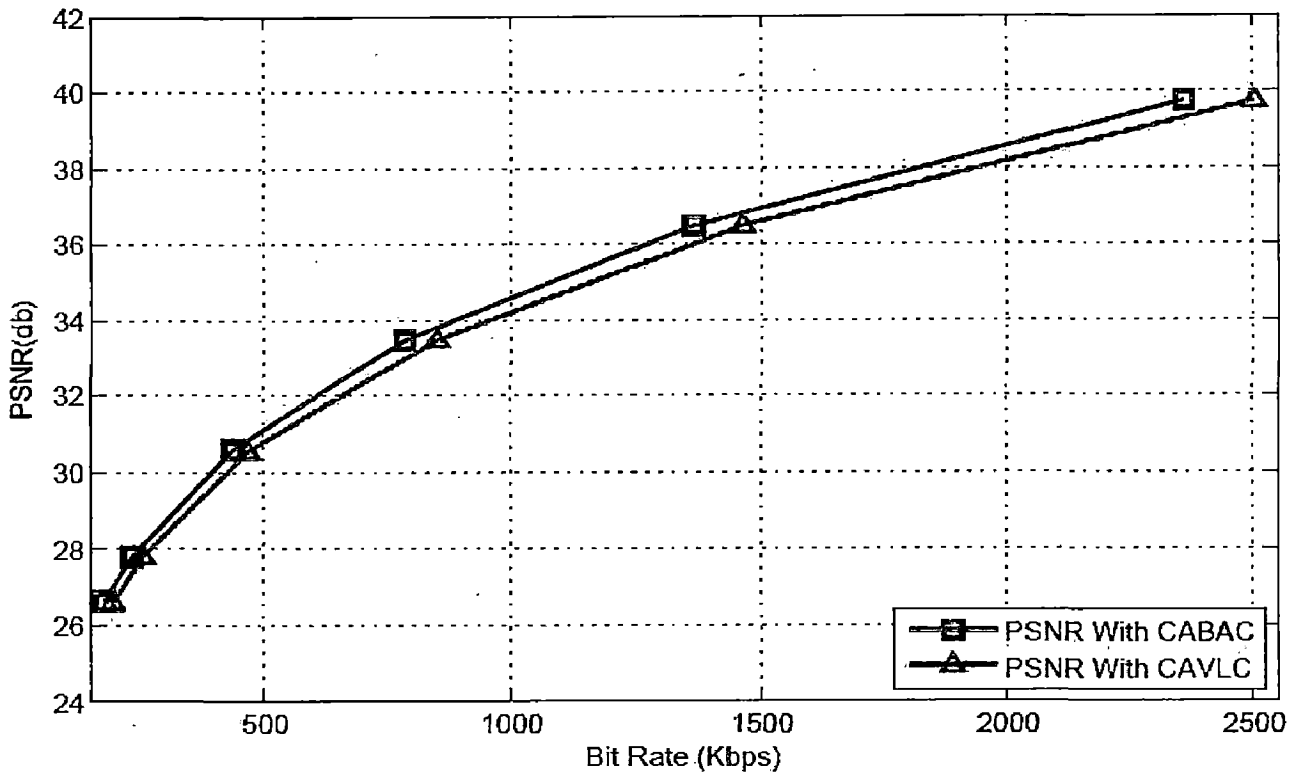
RESULTS AND DISCUSSION

This chapter presents the results obtained through computer simulation using H.264/AVC reference software version JM 10.2 [15], for various error resilient video coding features employed by the H.264/AVC, with some necessary computations performed using MATLAB 7.6. We begin with the results obtained for the H.264/AVC video compression.

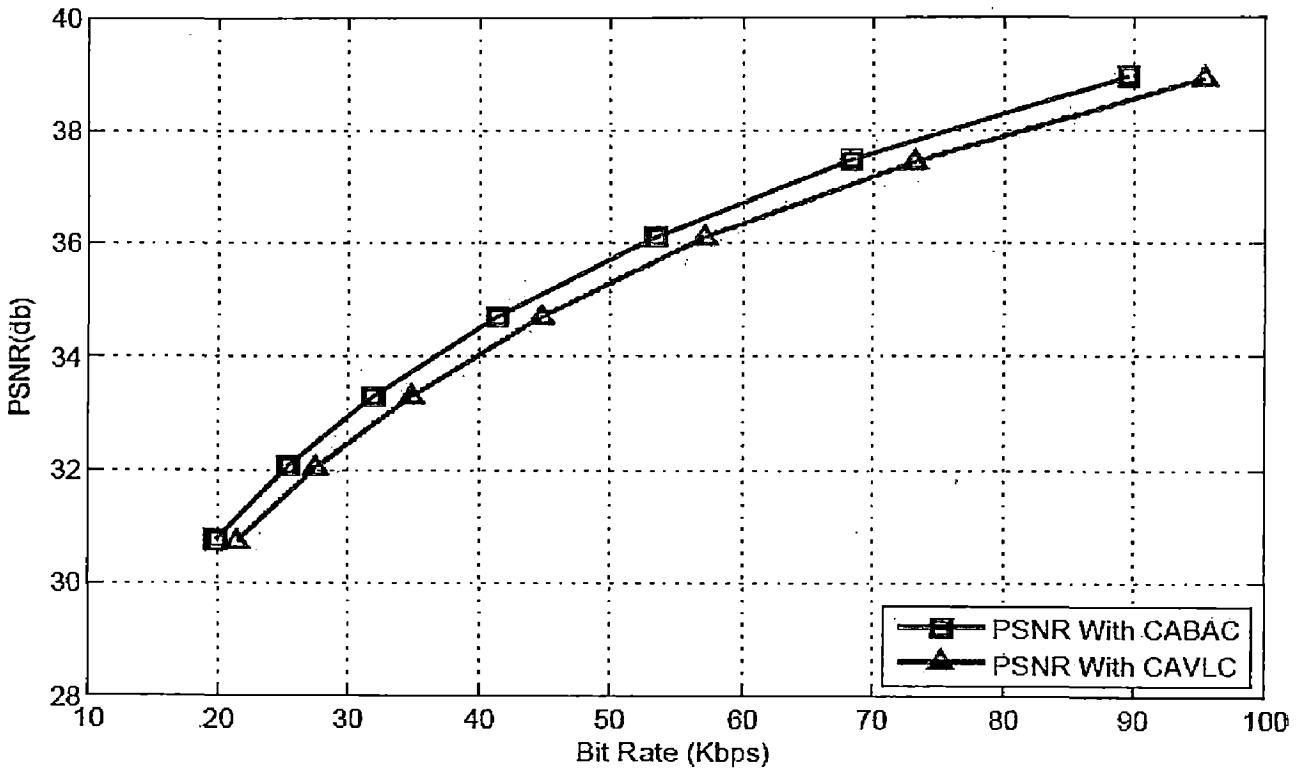
5.1 H.264/AVC Video Compression

In video coding, the system performance is measured by the average luminance PSNR. We begin our simulations with coding efficiency of H.264/AVC, which achieves better compression [refer Figure 3.1] than all other previously existing video coding standards [9].

In our simulation, we used two test video sequences, Bus and Foreman of length 150 frames each. The sequence Bus is in the Common Intermediate Format (CIF, 352×288 frame elements, progressive) and Foreman is in the Quarter Common Intermediate Format (QCIF, 176×144 frame elements, progressive). Only the first frame was intra-coded and the rest were coded as inter-frame (IPPPP....). CABAC and CAVLC both were used as entropy coding method. The coding efficiency was measured in term of average bit rate savings for a constant peak signal to noise ratio (PSNR). Figure 5.1 shows the PSNR of the luminance component versus the average bit rate for the two test sequences, out of which the former was encoded at 30 Hz, and the later at 10 Hz. We observe from Figure 5.1 that the compression performance is further improved if the CABAC entropy coding configuration is used. Compared to CAVLC, CABAC can typically provide reductions in bit-rate of the order of 10–20 percent for the same objective video quality when coding SDTV/HDTV signals.



(a) Bus CIF @30 Hz



(b) Foreman QCIF @10 Hz

Figure 5.1: Compression efficiency for H.264/AVC video coding standard.

Due to its two-layer structure design [refer Figure 3.2] (a video coding layer (VCL), which is designed to obtain highly compressed video data, and a network abstraction layer (NAL), which formats the VCL data and adds corresponding header information for adaptation to various transportation protocols or storage media, the standard gives strong emphasis to error resiliency and the adaptability to various networks.

5.2 Error Resilience Coding

When transmitted over a mobile channel, compressed video may suffer severe degradation, making the use of error-resilience techniques vital. We have simulated various error resilience features employed by H.264/AVC video coding standard. Simulation results of the error resilience features of H.264/AVC are presented below.

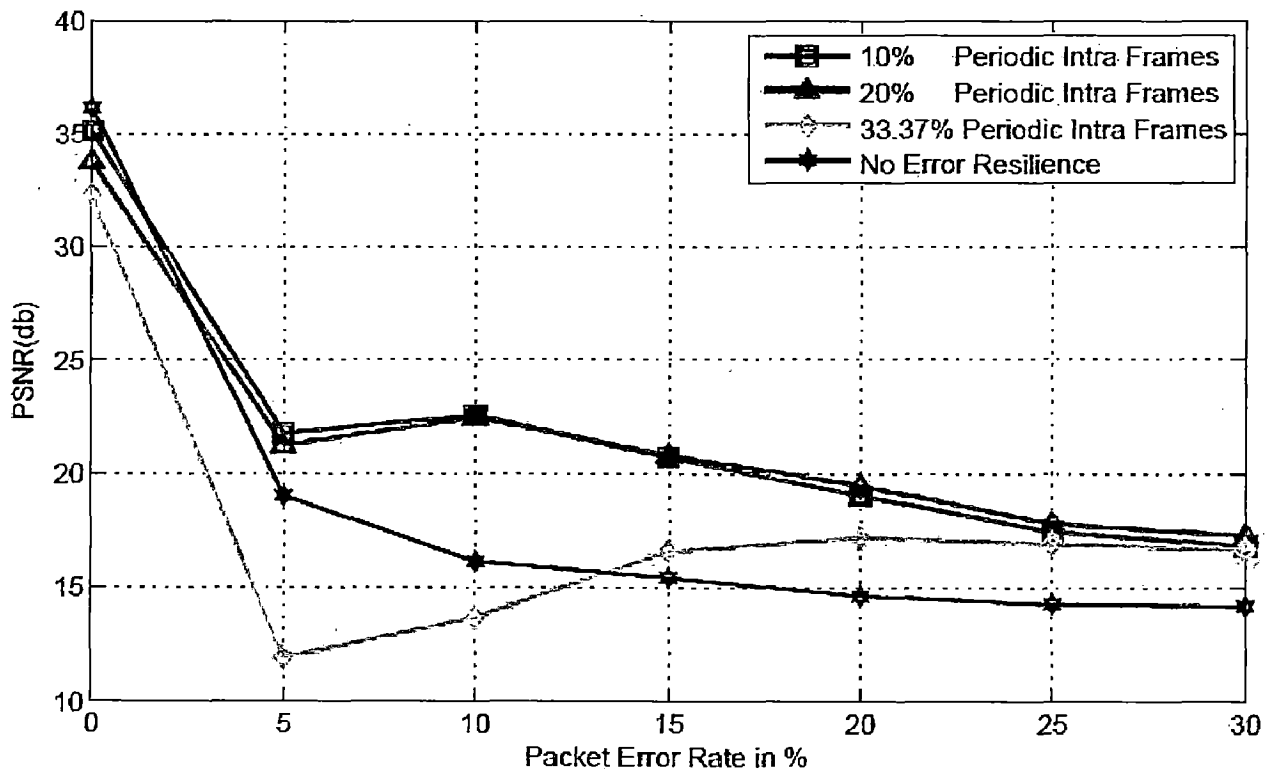
5.2.1 Intra Placement

Intra placement on the macroblock, slice, or frame level, was used primarily to combat drifting effects. Simulation was carried out in Baseline profile using two QCIF test video sequence Foreman and City, with a target bit-rate of 56 kbps.

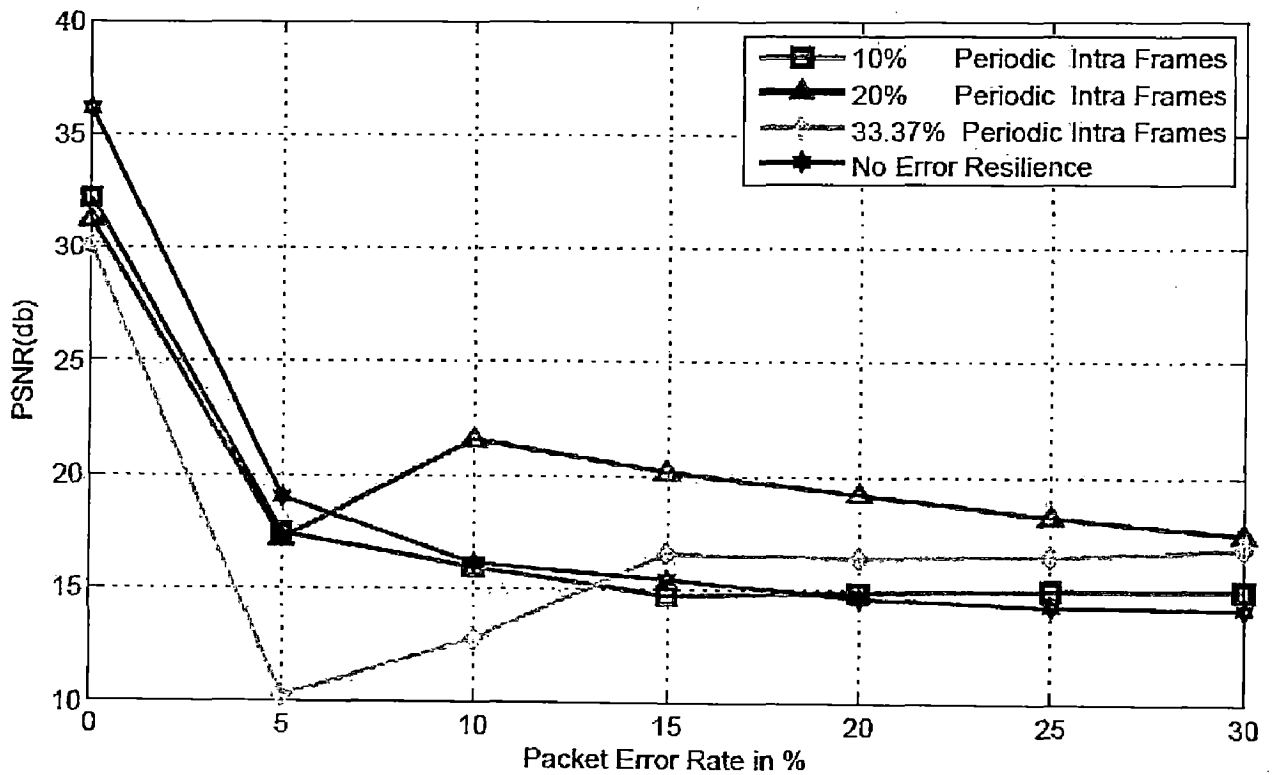
The simulation parameters were set as follows:

- period of additional Intra frame was chosen as 10, 5 and 3 ;
- QP was kept constant at a value to match the bit rate requirements;
- 10 previous frames were used for inter motion search;
- no B-slices was used;
- CAVLC entropy coding was employed;
- frame-copy was used as error concealment method.

Figure 5.2 shows the PSNR of the luminance component versus the packet error rate for the two test sequences, under lossy condition, both of which were encoded at 10 Hz.



(a) Foreman QCIF @10 Hz



(b) City QCIF @10 Hz

Figure 5.2: Average Y-PSNR over packet error rate under Intra frame placement.

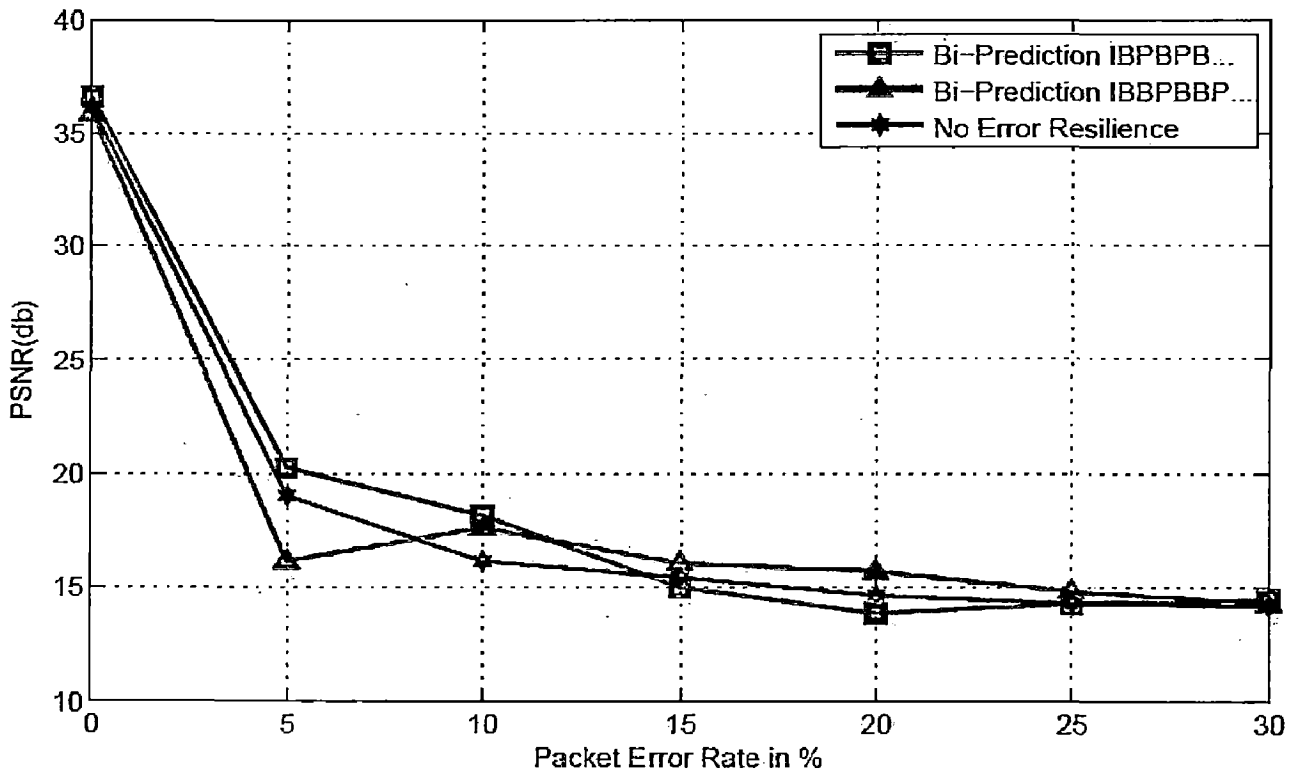
It is observed from Figure 5.2 that placing additional intra-frames can reduce the error propagation. However, intra coded frames require a large number of bits and so there is a limitation on the number of intra coded frames per video frame. We cannot place too much intra frames since that reduces compression efficiency. On the other hand, it is also not necessary that the performance will be better in case of less intra. It is clear from Figure 5.2 that using 20 percent additional intra provides the best performance and hence their application should always be considered very carefully.

Prediction can be further improved by the use of bi-prediction at the cost of computational complexity. We have simulated the performance of bi-prediction under lossy condition.

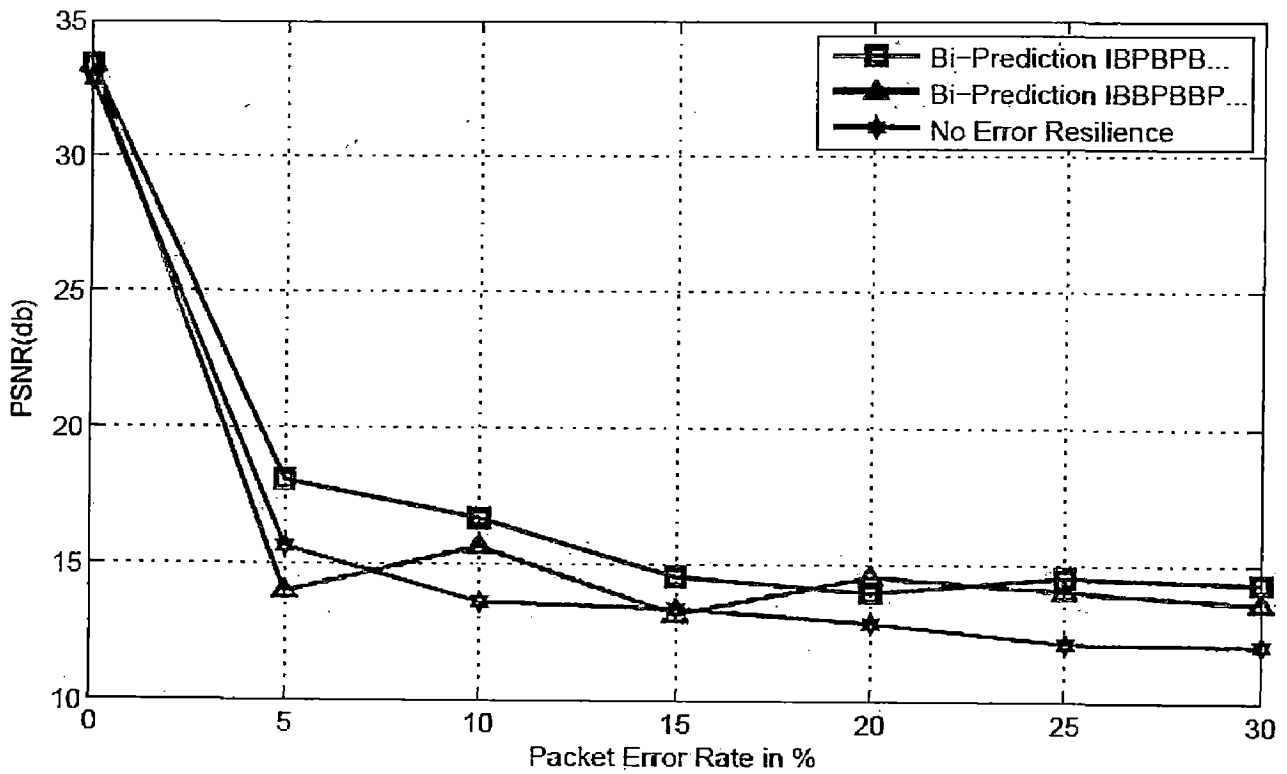
The simulation parameters were set as follows:

- only first frame was intra coded;
- QP was kept constant at a value to match the bit rate requirements;
- 10 previous frames were used for inter motion search;
- IBPBPBPB.... and IBBPBBPBBP... patterns of sequence were used;
- CAVLC entropy coding was used;
- frame-copy was used as error concealment method.

Figure 5.3 shows the PSNR of the luminance component versus the packet error rate for the two test sequences, City and Foreman, under lossy condition, in case of Bi-prediction, both of which were encoded at 10 Hz with a target bit rate of 56 kbps. It is observed that pattern IBPBPBPB... perform better than IBBPBBPBBP... in case of City, while opposite is the case in Foreman. However, H.264/AVC video coding standard does not specify bi-prediction as error resilient feature [12][13], but as we see from Figure 5.3, this method may be used as an error resilient feature, if we can neglect the complexity, which is high in this case.



(a) Foreman QCIF @10 Hz



(b) City QCIF @10 Hz

Figure 5.3: Average Y-PSNR over packet error rate in case of Bi-prediction.

5.2.2 Slice Structuring

Flexible macroblock ordering, available in the Baseline and Extended profile, but not in the Main profile, allows assigning MBs to slices in an order other than the scan order. To do so, each MB is statically assigned to a slice group using a macroblock allocation map (MBAmap). The slice structuring strategy aims at avoiding error propagation from a corrupted packet to subsequent packets. We have simulated different slice structures supported by H.264/AVC video coding standard [refer section 4.2.4]. Firstly, the video sequence was encoded without any slice structure i.e. one packet per frame, then different slice structures were used to encode the test sequence, and the performance was analyzed under lossy condition. Simulation was carried out in Baseline profile using two QCIF test video sequences Foreman and City, with a target bit-rate of 56 kbps.

The video error concealment schemes perform very well when the lost blocks are arranged in the checker board/scattered blocks fashion or as interleaving of rows. This is depicted in Figure 5.4. This arrangement is helpful in concealing the lost blocks by their surrounding blocks. The objective behind the flexible macroblock ordering (FMO) is to scatter possible errors to the whole frame as equally as possible to avoid error accumulation in a limited region.

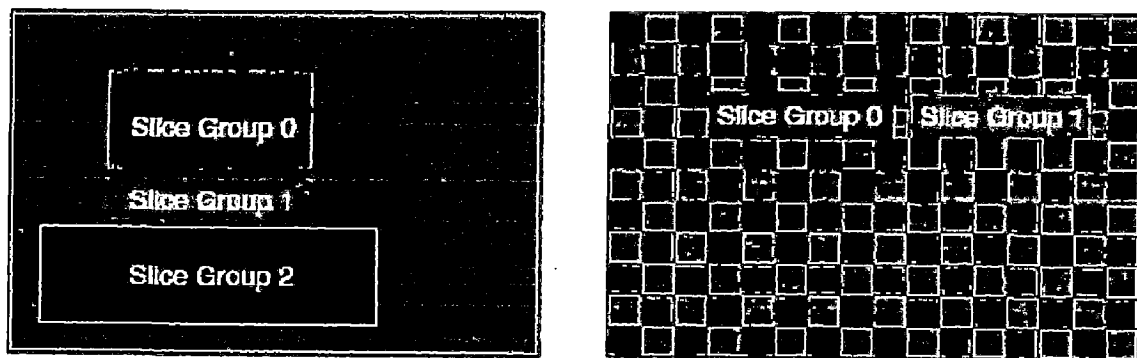
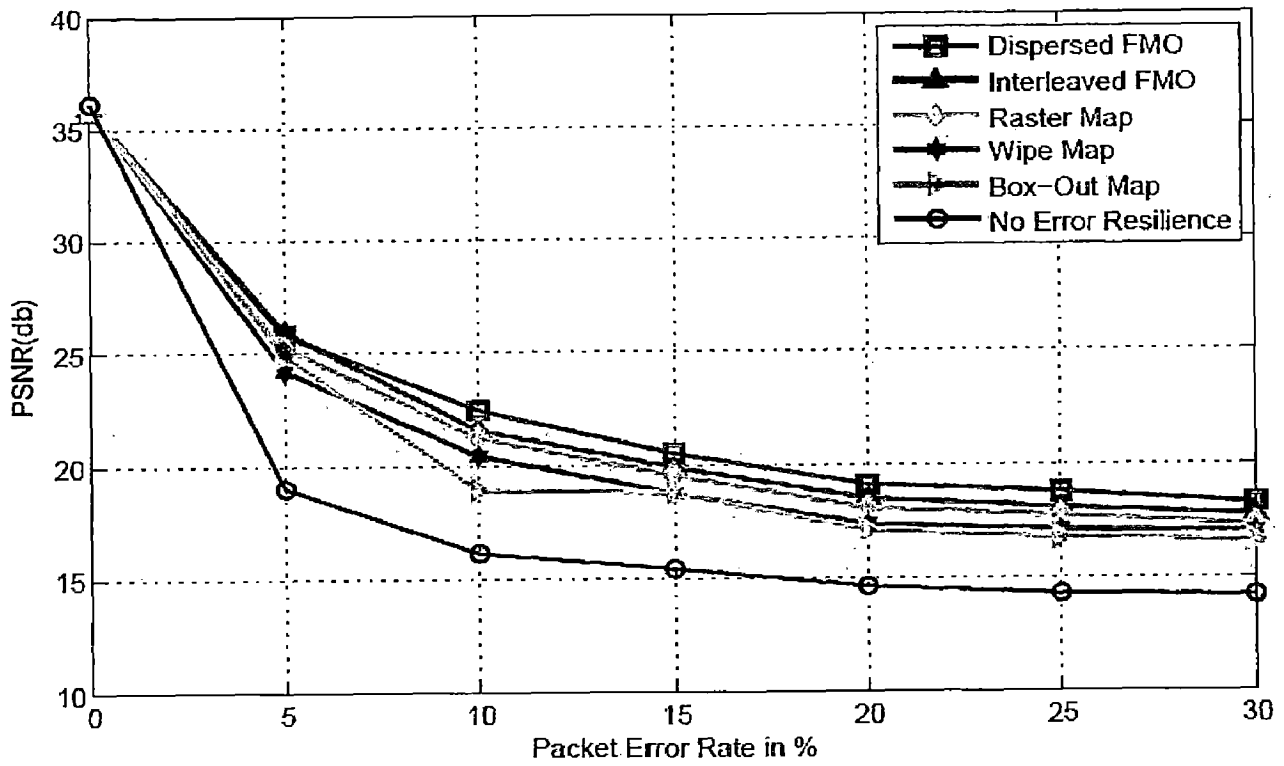
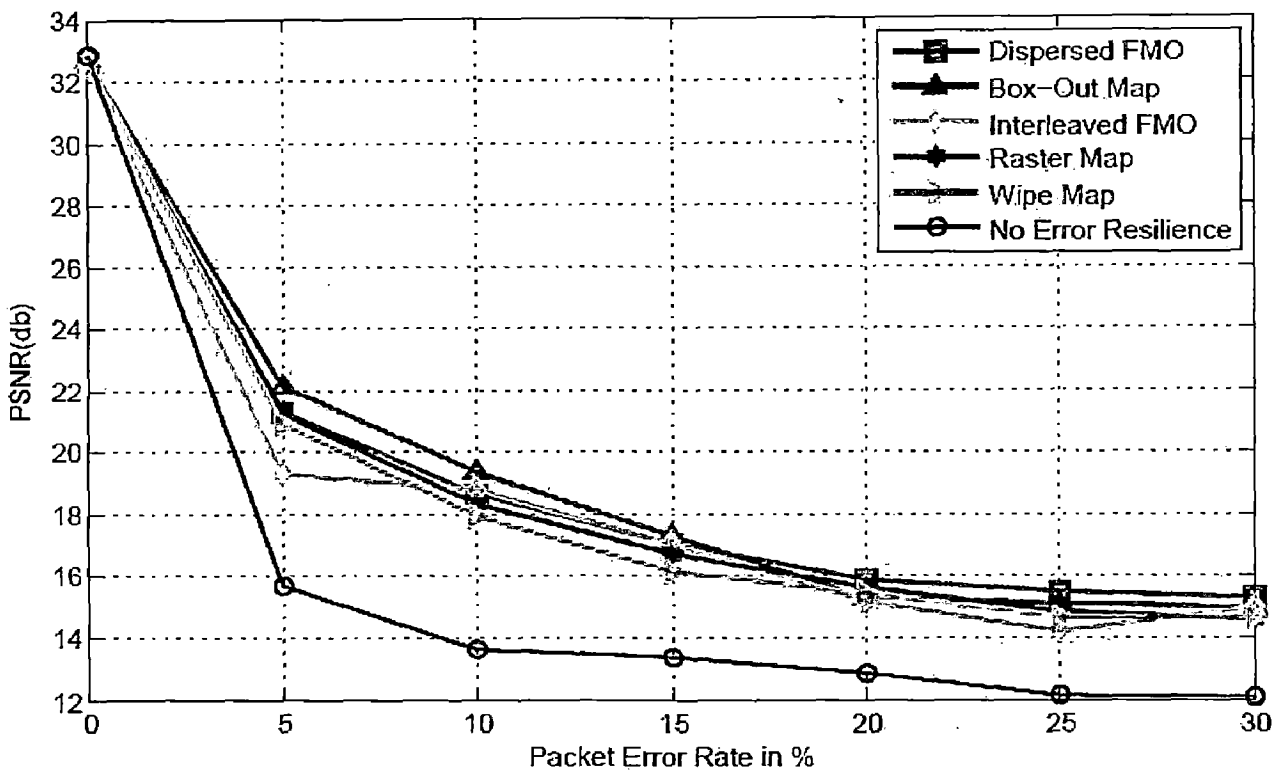


Figure 5.4: Division of an image into several slice groups using FMO [11].

Figure 5.5 shows the PSNR of the luminance component versus the packet error rate for the two test sequences, City and Foreman, under lossy condition; in case of different slice structures where 2 slice groups per frame were taken for simulation. Both sequences were encoded at 10 Hz with a target bit rate of 56 kbps.



(a) Foreman QCIF @10 Hz



(b) City QCIF @10 Hz

Figure 5.5: Average Y-PSNR over packet error rate under different slice structures.

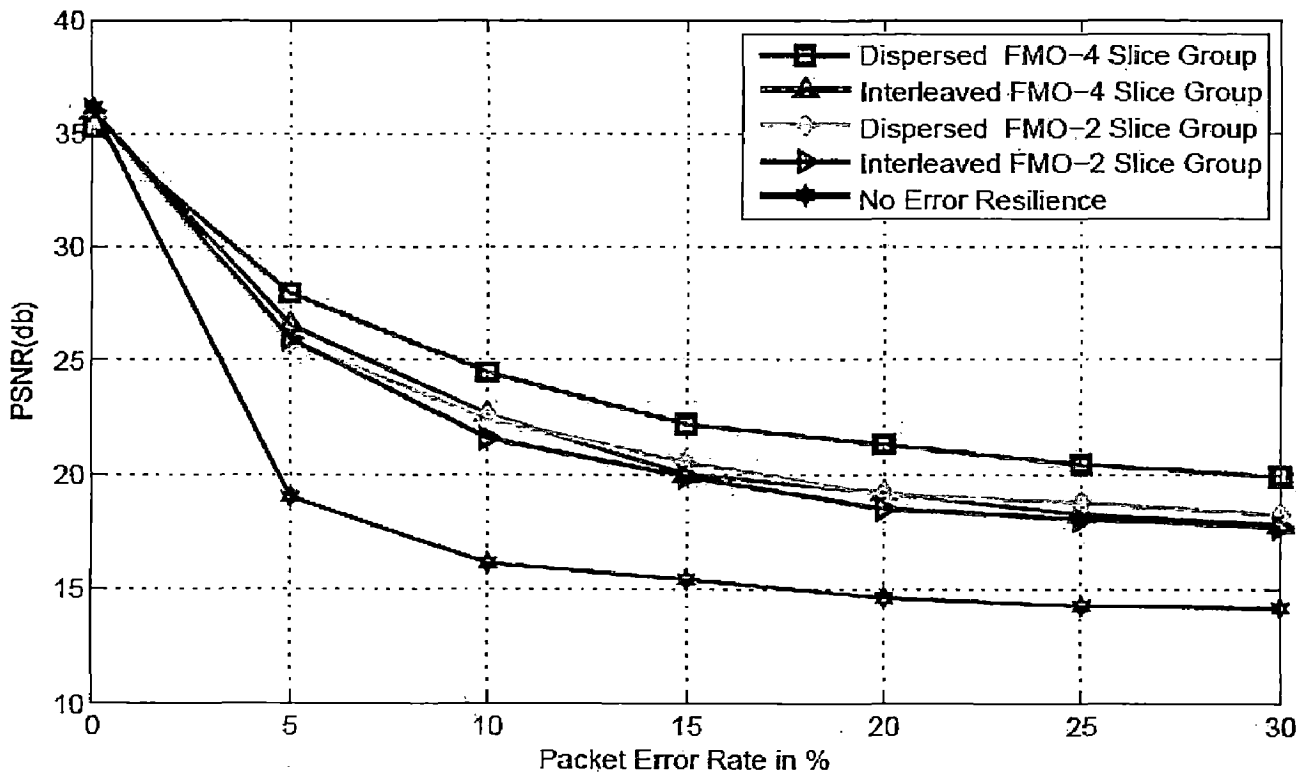
In case of slice structuring, every lost MB has several spatial neighbors that belong to the other slice. Hence, an error-concealment mechanism has sufficient information for efficient concealment. It is observed from the Figure 5.5 that dispersed or checker board structure (2 slice group per frame i.e., macroblocks with odd addresses in slice group 1, with even addresses in slice group 2) performs better than other slice structures with increase in loss rate.

Performance may be further improved, if 4 slice groups per frame are used, because, in this case, the samples of a missing slice are surrounded by many samples of correctly decoded slices. Figure 5.6 illustrates the slice structuring in case of dispersed FMO, when 4 slice groups per frame were adopted in simulation.

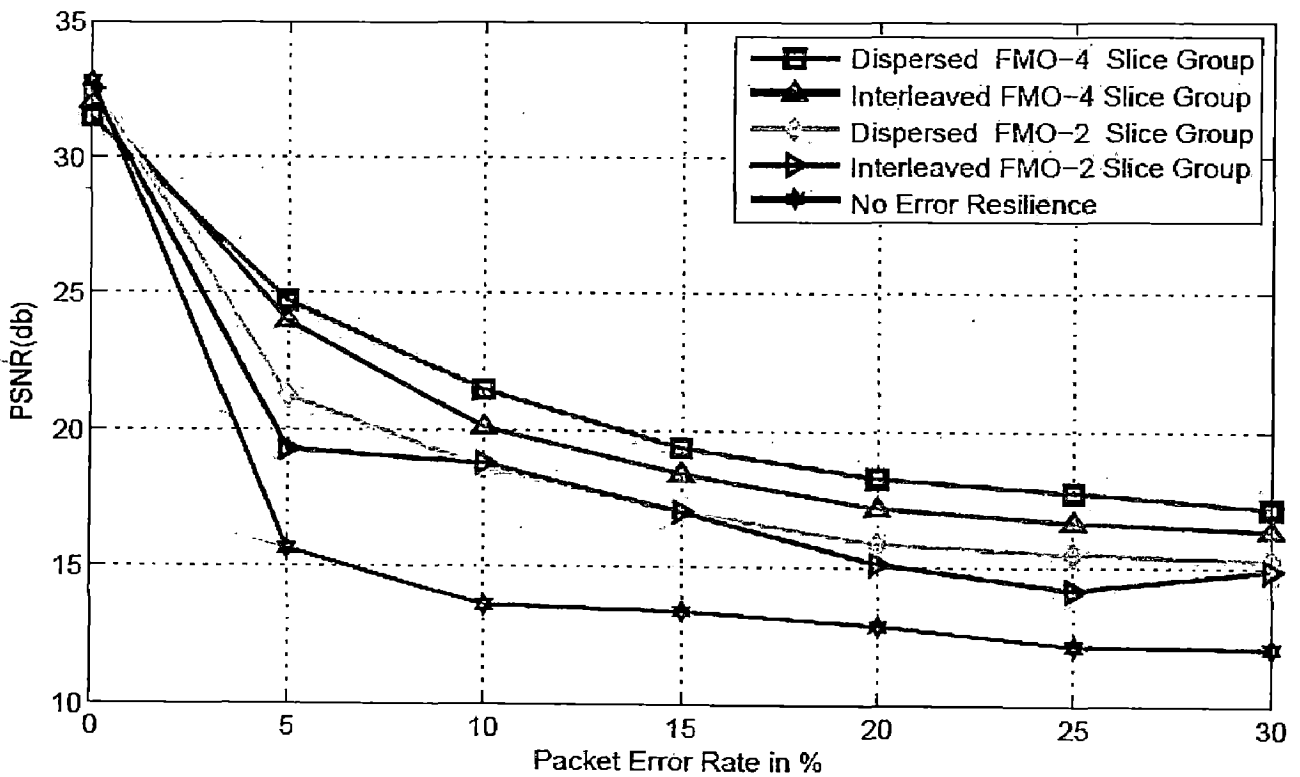
0	1	2	3	0	1	2	3	0	1	2
2	3	0	1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3	0	1	2
2	3	0	1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3	0	1	2
2	3	0	1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3	0	1	2
2	3	0	1	2	3	0	1	2	3	0
0	1	2	3	0	1	2	3	0	1	2

Figure 5.6: Dispersed FMO (4 slice groups per frame).

Figure 5.7 shows the PSNR of the luminance component versus the packet error rate for the two test sequences, City and Foreman, under lossy condition, in case of dispersed and interleaved slice structures where 4 slice groups per frame were taken for simulation. Both sequences were encoded at 10 Hz with a target bit rate of 56 kbps. It is observed from Figure 5.7 that the structure with 4 slice groups outperforms the structure with 2 slice groups.



(a) Foreman QCIF @10 Hz



(b) City QCIF @10 Hz

Figure 5.7: Performance comparison of 4 slice group versus 2 slice group per frame.

5.2.3 Weighted Prediction

A new innovation in H.264/AVC allows the motion-compensated prediction signal to be weighted and offset by amount specified by the encoder. This can dramatically improve coding efficiency for scenes containing fades [6], and can be used flexibly for other purposes as well.

Weighted prediction is available in the Main and Extended profiles, but not in the Baseline profile. We have simulated the weighted prediction in Main profile using CAVLC. Figure 5.8 shows the performance of weighted prediction. It is clear from the Figure 5.8 that the weighted prediction enhances the PSNR.

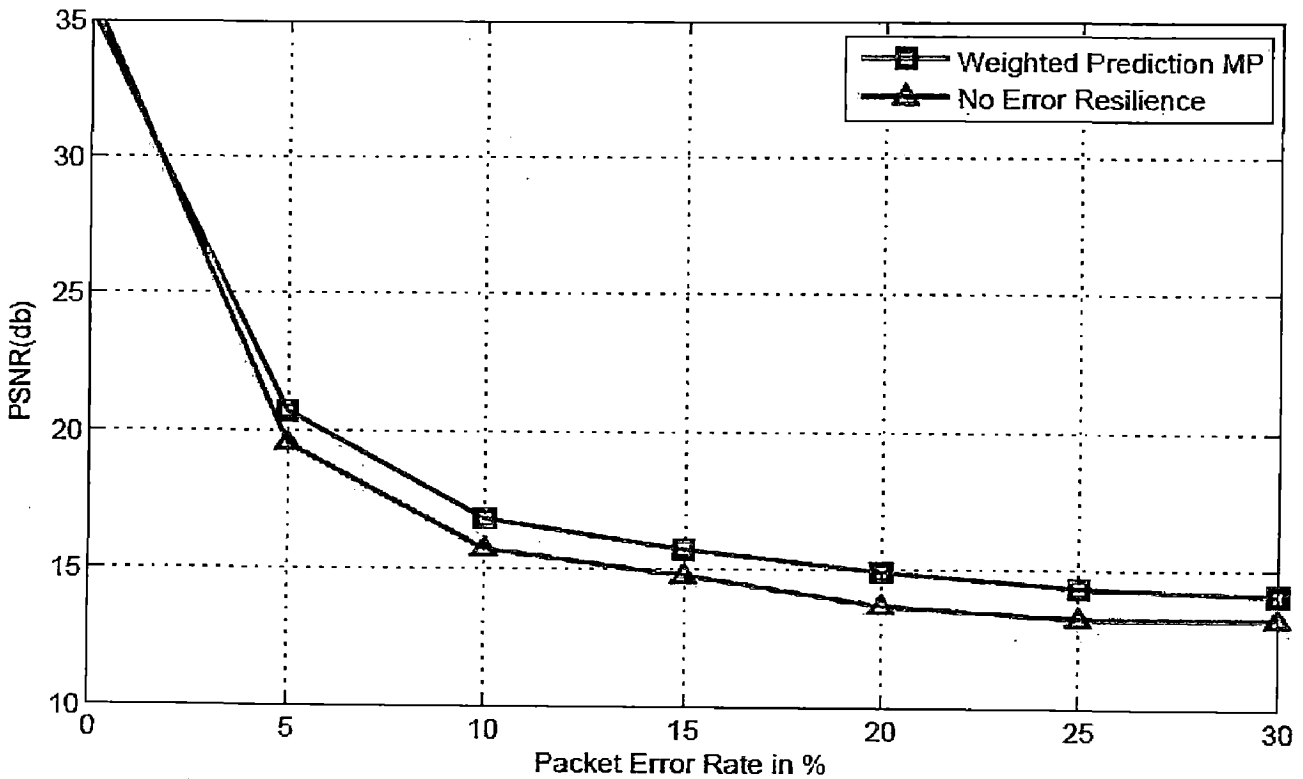
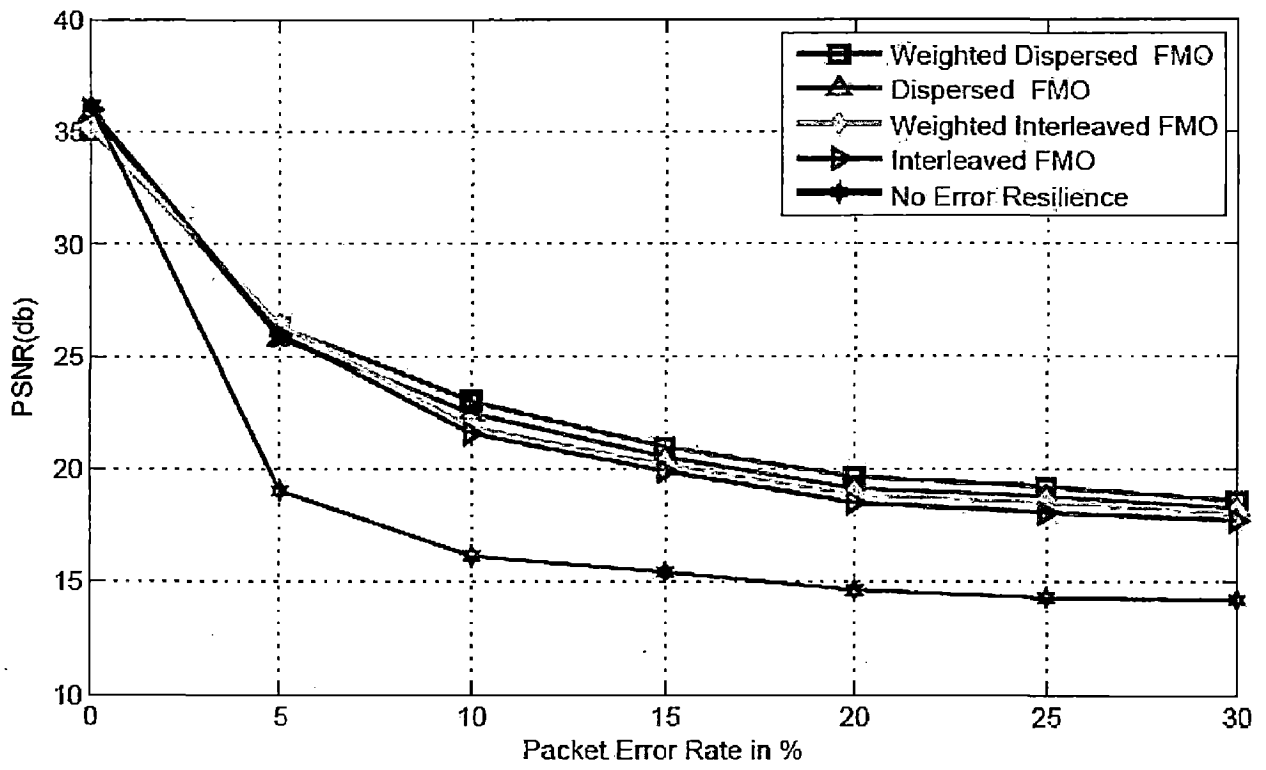
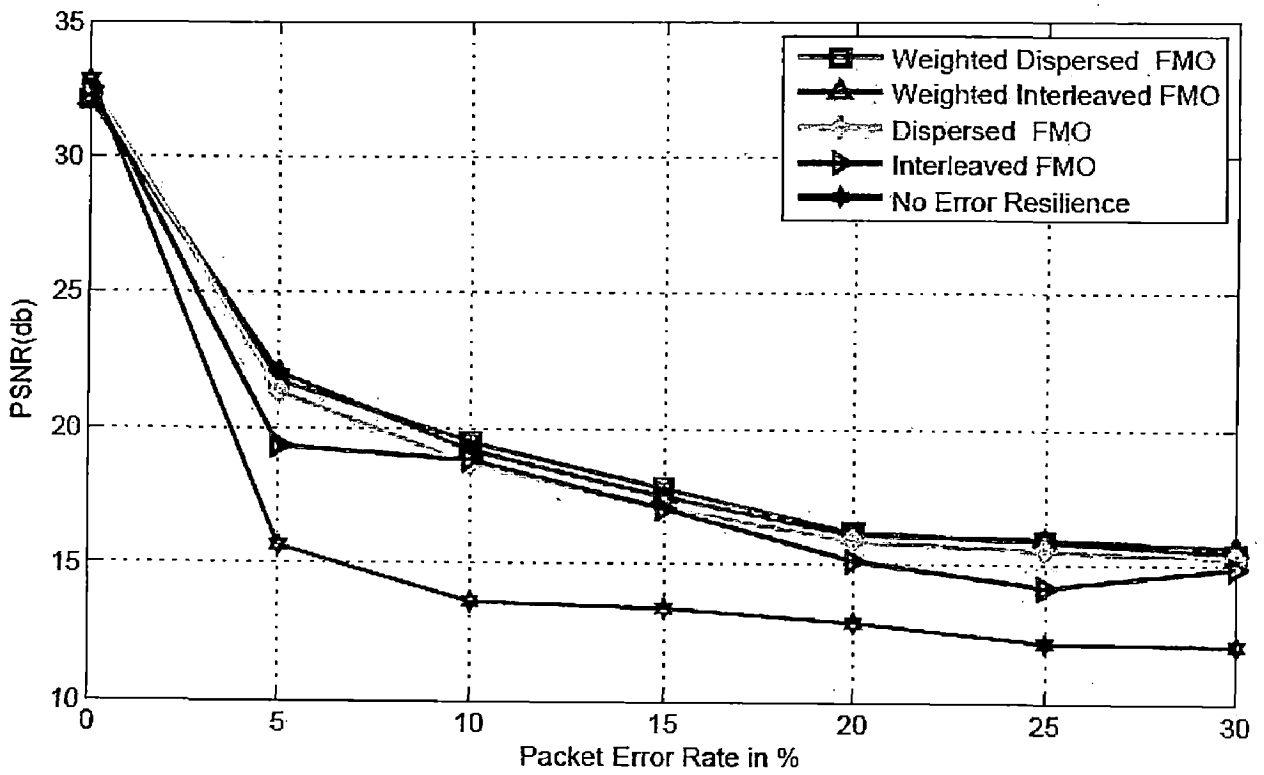


Figure 5.8: Performance of Weighted-prediction

We have combined the weighted-prediction in Extended-profile as an error resilient feature with slice structuring. In normal slice structure, the lost blocks are concealed by their surrounding blocks; while in this case, concealment is done by taking the weights of



(a) Foreman@10 Hz



(b) City@10 Hz

Figure 5.9: Performance of Weighted-prediction under slice structuring.

surrounding blocks. Our main aim was to check the performance of weighted-prediction within different slice structures. It is observed from the Figure 5.9 that the weighted prediction concealment scheme gives some improvement in PSNR.

Results obtained from simulation are very promising and authentic. Compression efficiency is higher in case of CABAC, as shown in Figure 5.1, this is same as discussed in [10]. Error resiliency results presented in this dissertation are conforming to the results discussed in [13].

Chapter 6

CONCLUSIONS

In this dissertation, various error resilient video coding schemes suitable for video transmission have been discussed in detail. Based on the results obtained, following conclusions are drawn.

The coding tools of H.264/AVC when used in an optimized mode allow for bit saving of about 50% compared to previous video coding standards like MPEG-4 and MPEG-2 for a wide range of bit rates and resolutions. However, this savings comes at the price of an increased computational cost.

The use of I-frame is undoubtedly the most powerful tool to stop error propagation and recover corrupted frames. Since the I-frames are independently coded without using temporal prediction, they reset the prediction process. The coding efficiency is therefore lower than the prediction frame. Moreover, transmitting an I-frame requires a large bandwidth and incurs large bit rate variation, which may not be acceptable in many low-delay applications.

Prediction may be improved by the use of bi-prediction at the cost of increased computational complexity. However, due to its higher complexity it is not suitable for low delay applications.

Slice structured coding reduces packet loss probability and the visual degradation from packet losses, especially in combination with decoder error concealment methods. The

price of the use of slice structuring offers somewhat lower coding efficiency (because of the broken in-frame prediction mechanisms between non-neighboring MBs) and, in highly optimized environments, a somewhat higher delay.

The use of weighted prediction with slice structuring allows the motion-compensated prediction signal to be weighted which in turn improves coding efficiency.

Finally, the choice of one or more of the above-mentioned error resiliency schemes should consider the practical applications as well as network environment.

6.1 Future Scope

The works presented in this thesis is limited to equal error protection. However, in video coding, some syntax elements in the bit-stream are more important than others. So, video bit-streams can be partitioned to segments of different priorities according to their impact on subjective quality. In new of this, the work may be extended to priority partitioning, which enables unequal error protection according to the importance of syntax elements.

REFERENCES

- [1] Y. Wang and Q. Zhu, "Error control and concealment for video communication: a review," *Proceedings of the IEEE*, vol. 86, no. 5, pp. 974-997, 1998.
- [2] Yao Wang, Stephan Wenger, Jiangtao Wen, and Aggelos K. Katsaggelos, "Error resilient video coding techniques: real-time video communication over unreliable networks", *IEEE Signal Processing Magazine*, vol. 17, no. 4, pp. 61-82, 2000
- [3] ITU-T Recommendation H.264 (05/2003), "Advanced video coding for generic audio visual services."
- [4] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra," Overview of the H.264/AVC video coding standard" *IEEE Transaction on Circuit and Systems for Video Technology*, vol. 13, no. 7, pp. 560-576, 2003
- [5] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-constrained coder control and comparison of video coding standards," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 688-703, 2003
- [6] T. Stockhammer and M. M. Hannuksela, "H.264/AVC video for wireless transmission," *IEEE Wireless Communications*, vol. 12, no. 4, pp. 6-13, 2005.
- [7] Iain E. G. Richardson, *H.264 AND MPEG-4 video compression: video coding for next-generation multimedia*, John Wiley & Sons Ltd, England, UK, 2003.
- [8] Iain E. G. Richardson, *Video codec design: developing image and video compression system*, John Wiley & Sons Ltd, England, UK, 2002.

- [9] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, T. Wedi, "Video coding with H.264/AVC: tools performance and complexity," *IEEE Circuits and Systems Magazine*, vol. 4, no. 1, pp. 7–28, 2004
- [10] D. Marpe, T. Wiegand, G. J. Sullivan, "The H.264/MPEG-4 advanced video coding standard and its applications," *IEEE Communications Magazine*, vol. 94, no. 8, pp. 134-143, 2006.
- [11] T. Stockhammer, M. M. Hannuksela, and T. Wiegand, "H.264/AVC in wireless environments," *IEEE Transaction on Circuit and Systems for Video Technology*, vol. 13, pp. 657–673, 2003.
- [12] S. Wenger, "H.264/AVC over IP," *IEEE Transaction on Circuit and Systems for Video Technology*, vol. 13, pp. 645–656, 2003.
- [13] Sunil Kumar, Liyang Xu, Mrinal K. Mandal, and Sethuraman Panchanathan, "Error resiliency schemes in H.264/AVC standard," *Journal of Visual Communication & Image Representation*, vol. 17, no. 2 pp. 425-450, 2006.
- [14] J Mochnac, S. Marchevsky, "Error resilience tools in the MPEG-4 and H.264 video coding standards," *Proceeding of the 18th International Conference on Radioelektronika*, Prague, pp. 1-4, 2008
- [15] H.264/AVC reference software available online at <http://iphome.hhi.de/suehring/>