ERROR CONCEALMENT TECHNIQUES IN H.264/AVC, FOR VIDEO TRANSMISSION OVER WIRELESS NETWORKS

by

VINEETH SHETTY KOLKERI

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 ELETRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON
December 2009

ACKNOWLEDGEMENTS

Firstly, I would thank my advisor Prof. K.R.Rao for his invaluable guidance and support, and his tireless guidance, dedication to his students and maintaining new trend in the research areas has inspired me a lot without which this thesis would not have been possible.

I also like to thank the other members of my advisory committee Prof. W. Alan Davis and Prof. Kambiz Alavi for reviewing the thesis document and offering insightful comments.

I appreciate all members of Multimedia Processing Lab for their support during my research work. This includes Prof. Jung Ho Lee for introducing this topic and helped me in video simulation. I would also like to thank my friends Prasanna Alva, Shreyas Shashidhar, Archana, Thrishala Shetty, Rushikesh, Vinoj, Sanjeev and Siddu Wali and for their comments and suggestions at various stages of my research.

Finally, I am grateful to my family; my mother Ms. Jyothi. C. Shetty, my sister Ms. Vidya Shetty, my brother-in-law Santhosh. M. Shetty and my sweet little niece Snigdha Shetty for their support, patience, and encouragement during my graduate journey.

September 18, 2009

ABSTRACT

ERROR CONCEALMENT TECHNIQUES IN H.264/AVC,

FOR VIDEO TRANSMISSION OVER

WIRELESS NETWORKS

Vineeth Shetty Kolkeri, M.S.

The University of Texas at Arlington, 2009

Supervising Professor: Dr. K. R. Rao

Several error concealment methods are applied to H.264/AVC (Advanced Video Coding) baseline profile such that the decoded video is error free. The error concealment techniques are implemented both in the spatial and temporal domains. The original and error concealment video sequences are compared in terms of MSE (Mean Square Error), PSNR (Peak-to-peak Signal to Noise Ratio) and SSIM (Structural Similarity Index Metric). This comparision has demonstrated that the error concealment methods are very effective in improving the visual quality. Implementation complexity also has been considered as the video transmission in baseline profile is meant for wireless networks

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

LIST OF ACRONYMS

3G/4G	Third or Fourth Generation
3GPP	3rd Generation Partnership Project
AVC	Advanced Video Coding
AVI	Audio Video Interleave
CABAC	Context Adaptive Binary Coding
CAVLC	Context Adaptive Variable Length Coding
DCT	Discrete Cosine Transform
DP	Data Partition
DVD	Digital Versatile Disk
FMO	Flexible Macroblock Ordering
GOP	Group of Pictures
HVS	Human Visual System
IDR	Instantaneous Decoder Refresh
I-frame	Intra frame
IEC	International Electrotechnical Commission
IP	Internet Protocol
ISO	International Organization for Standardization
ITU	International Telecommunication Union
ITU-R	ITU – Radio communication Standardization Sector
ITU-T	ITU – Telecommunication Standardization Sector
JM	Joint Model
JVT	Joint Video Team

MAM	Macroblock Allocation Map
MB	Macroblock
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
NAL	Network Abstraction Layer
POC	Picture Order Count
PPS	Picture Parameter Set
PSNR	Peak-to-peak Signal to Noise Ratio
QCIF	Quarter Common Intermediate Format
QoS	Quality of Service
RGB	Red Green and Blue Components
RTP	Real-time Transfer Protocol
SPS	Sequence Parameter Set
SSIM	Structural Similarity Index Metric
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VCL	Video Coding Layer
VLC	Variable Length Code
YUV	Luminance and Chroma components

CHAPTER 1

INTRODUCTION

Due to the rapid growth of wireless communications, video over wireless networks has gained a lot of attention. Cellular telephony has had the most important development. At the beginning, cellular telephony was conceived for voice communication [20]; however, nowadays it is able to provide a diversity of services, such as data, audio and video transmission thanks to the apparition of third and fourth generation (3G/4G) developments of cellular telephony [23].

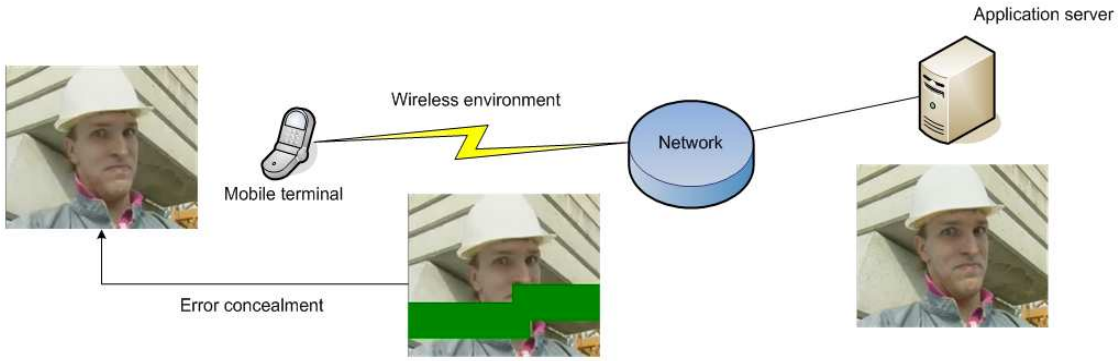


Figure 1.1: Typical situation on 3G/4G cellular telephony

Figure 1.1 illustrates a 3G/4G [11] cellular telephony system where a user, with his mobile terminal, demands a video streaming service. The video stream comes from the application server over the network. Then it is transmitted over the wireless environment to the user. During the transmission, the video signal is error prone. This system, because of the bandwidth limitation, works with low resolution (QCIF 176 x 144) videos so the loss of one packet means a big loss of information [12]. Since this process is a real time application it is not possible to perform retransmissions. The only way to fix the errors produced by packet losses is

by using error concealment methods in the mobile terminal. The focus of this thesis is on spatial and temporal correlations of the video sequence to conceal the errors [20].

The main task of error concealment is to replace missing parts of video content by previously decoded parts of the video sequence in order to eliminate or reduce the visual effects of bit stream error. The error concealment exploits the spatial and temporal correlations between the neighboring image parts (macroblocks) within the same frame or the past and future frames [6]. Techniques using these two kinds of correlation are categorized as spatial domain error concealment and temporal domain error concealment.

The spatial domain error concealment utilizes information from the spatial smoothness nature of the video image, and each missing pixel of the corrupted image part can be interpolated from the intact surroundings pixels [10]. The interpolation algorithm has been improved by the preservation of edge continuity using different edge detection methods.

The temporal domain error concealment utilizes from the temporal smoothness between the adjacent frames within the video sequence. The simplest implementation of this method is to replace the missing image part by spatially corresponding part within a previously decoded frame, which has the maximum correlation with the affected frame [9]. The copying algorithm has been improved by considering the dynamic nature of the video sequence. Different motion estimation algorithms have also been integrated to apply motion compensated copying [10].

There are still no standardized means for the performance evaluation of error concealment methods. To evaluate the quality of reconstruction, typically peak signal to noise ratio (PSNR) and structural similarity index metric (SSIM) [24] are used.

The focus of this thesis is the performance indicators for evaluating the error concealment methods. To test the performance evaluation methods, H.264 [3] video codec is used. H.264 [3] is the newest codec in video compression, which provides better quality with less bandwidth than the other video coding standards such as H.263 or MPEG-4 part-2 [7]. This

feature is very interesting for mobile networks due to the restricted bandwidth in these environments [20].

CHAPTER 2

H.264/AVC DESCRIPTION

H.264/MPEG-4 AVC [3] is the newest video compression standard, which promises a significant improvement over all previous video compression standards. In terms of coding efficiency, the new standard is expected to provide at least 2x compression improvement over the best previous standards and substantial perceptual quality improvements over both MPEG-2 and MPEG-4 part-2 visual.

Figure 2.1 shows the development of the video coding standards and the position of H.264 [3] standard which has highest compression gain among other standards. The ITU-T name for the standard is H.264 while the ISO/IEC [25, 30] name is MPEG-4 Advanced Video Coding (AVC), which is Part 10 of the MPEG-4 standard [3].

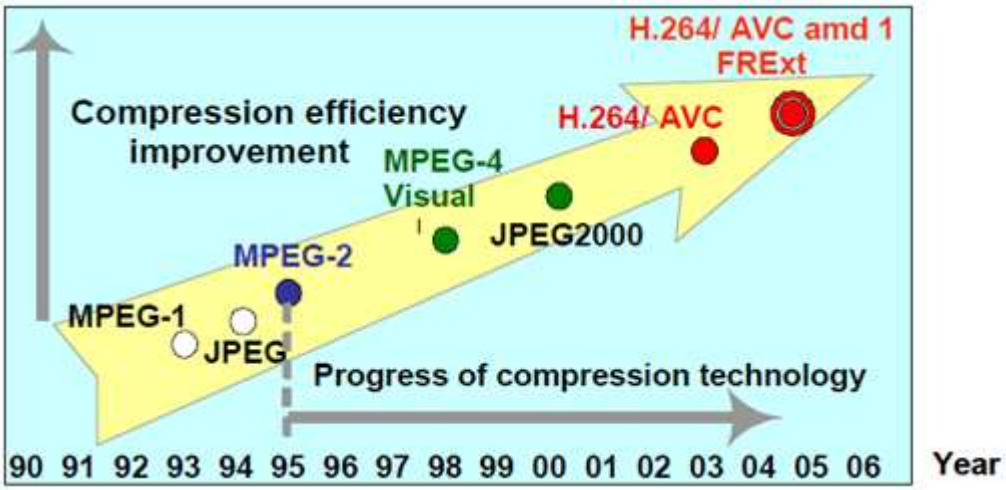


Figure 2.1: Position of H.264/MPEG-4 AVC standard [26]

The standard developed jointly by ITU-T and ISO/IEC supports video applications including low bit-rate wireless applications, standard-definition and high-definition broadcast television, video streaming over the internet, delivery of high-definition DVD content, and the highest quality video for digital cinema applications. Figure 2.2 shows the history of each video coding standard.

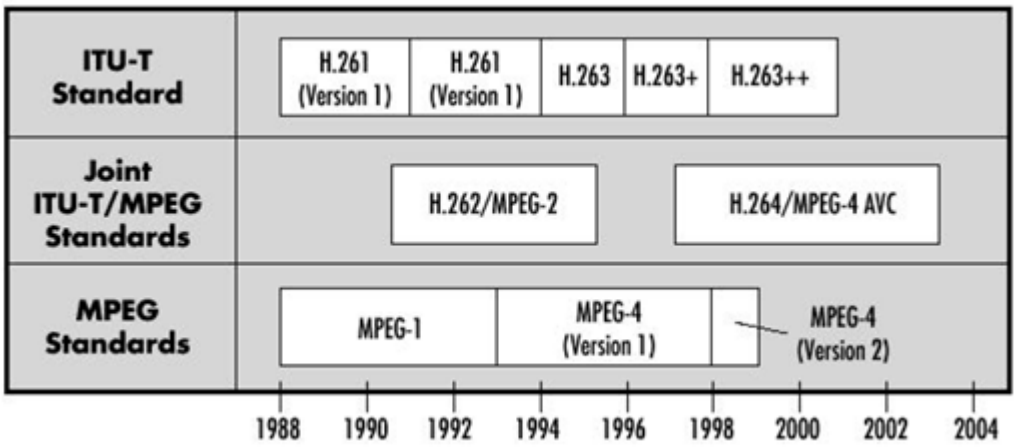


Figure 2.2: History of video standards [25]

Before becoming absorbed in deeper aspects of H.264/AVC like the encoding process or the new features that includes related to prior codecs, it will be better to explain some basics:

- Block

  A block is an 8 x 8 array of pixels.

- Macroblock

  A macroblock consists of a group of four blocks, forming a 16 x 16 array of pixels.

- Luminance

  In video signal transmission, luminance is the component that codes the information of luminosity (brightness) of the image.

- Chrominance

  Is the component that contains the information of color.

- YUV

  The YUV model defines a color space in terms of one luminance and two chrominance components. YUV models human perception of color more closely than the standard RGB model used in computer graphics hardware. Y stands for the luminance component (the brightness) and U and V are the chrominance (color) components. Concretely, U is blue-luminance difference and V is red-luminance difference.

- Chroma pixel structure

  A macroblock can be represented in several different manners when referring to the YUV color space. Figure 2.3 shows 3 formats known as 4:4:4, 4:2:2 and 4:2:0 video. 4:4:4 is full bandwidth YUV video, and each macroblock consists of 4 Y blocks, and 4 U/V blocks. Being full bandwidth, this format contains as much as data would if it were in the RGB color space. 4:2:2 contains half as much chrominance information as 4:4:4 and 4:2:0 contains one quarter of the chrominance information. The focus of this thesis is to use 4:2:0 format since it is the format typically used in video streaming applications.

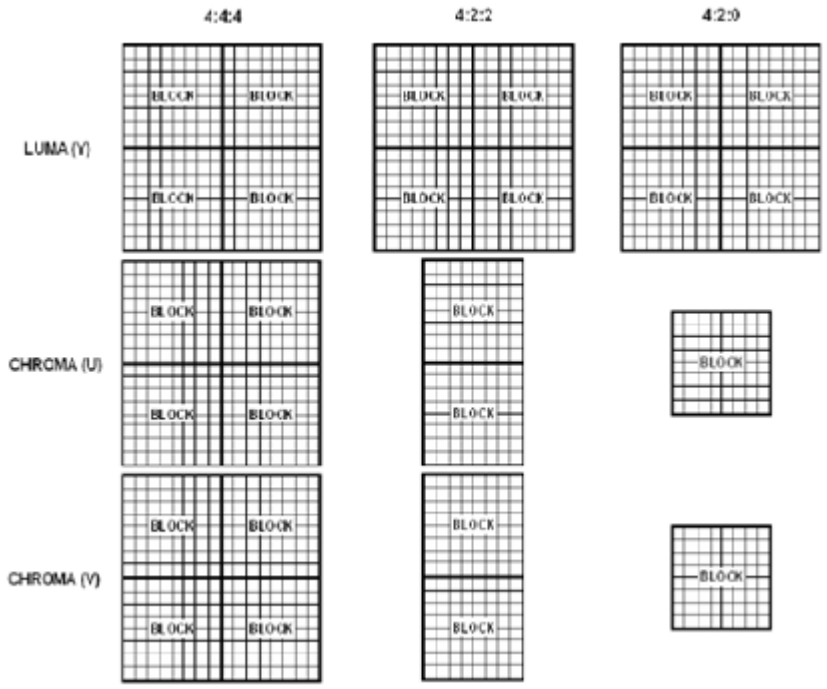


Figure 2.3: YUV different systems [17]

2.1 H.264/AVC coding process

The video coding layer (VCL) of H.264 consists of a hybrid of temporal and spatial predictions, in conjunction with transform coding [9]. Figures 2.4 and 2.5 shows the basic coding structure of H.264/AVC for a macroblock [3].

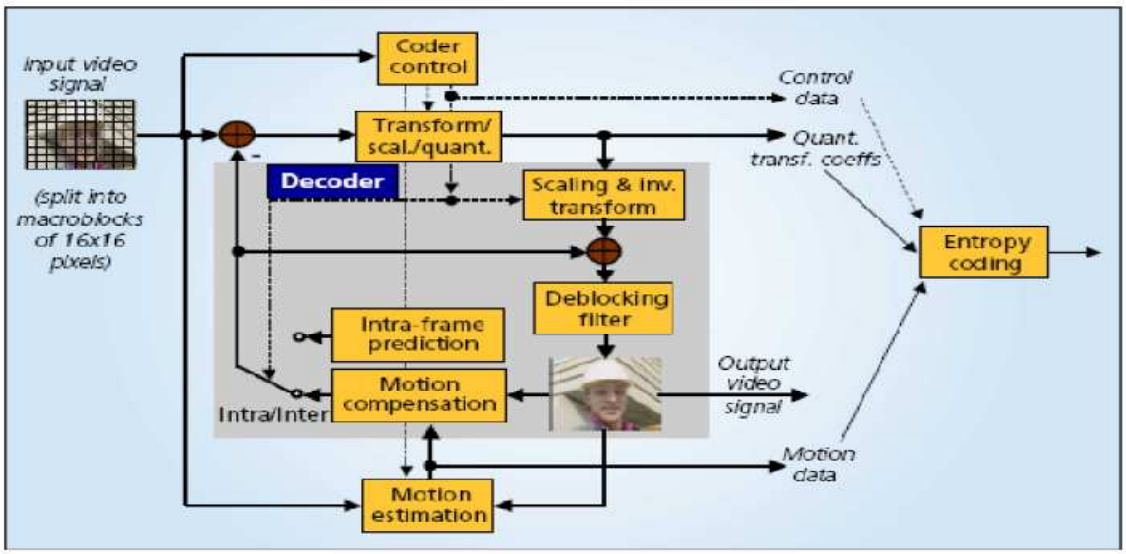


Figure 2.4: The basic coding structure of H.264/AVC for a macroblock [3, 18]

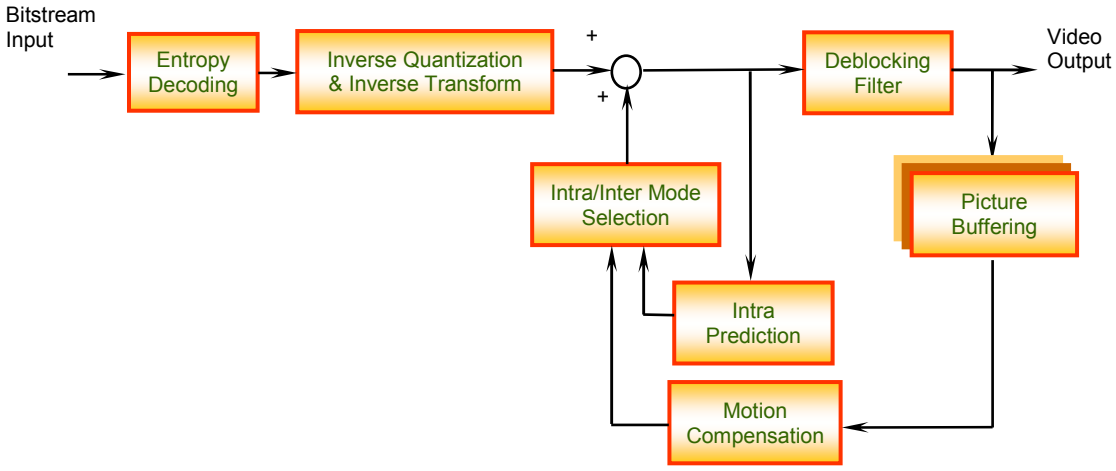


Figure 2.5: Block diagram of H.264 Decoder [3]

H.264 applies two types of slice coding, Intra-and Inter-slices. In case of Intra-slice, each sample of the macroblock within the slice is predicted using spatially neighboring samples of previously coded macroblocks. The coding process chooses which and how the neighboring samples are used for intra prediction, which is simultaneously conducted at the encoder and decoder using the transmitted Intra prediction side information [9]. In case of Inter slice the

encoder employs prediction (motion compensation) from other previously decoded pictures. The encoding process of Inter prediction consists of choosing motion data, comprising the reference picture, and a spatial displacement that is applied to all samples of the block. The motion data, which are transmitted as side information, are used by the encoder and decoder to simultaneously provide the Inter prediction signal.

In a series of frames, video data can be reduced by methods such as difference coding, which is used by most video compression standards including H.264. In difference coding, a frame is compared with a reference frame and only pixels that have changed with respect to the reference frame are coded. In this way, the number of pixel values that are coded and sent is reduced.

The residual of the prediction which is the difference of the original and the predicted blocks is transformed by the integer discrete cosine transform. The transform coefficients are scaled and quantized. The quantized transform coefficients are entropy coded by using CAVLC and transmitted together with the side information for either Inter frame or Intra frame prediction. The encoder contains decoder to conduct prediction for the next blocks or the next picture. Therefore, the quantized transform coefficients are inverse scaled and inverse transformed in the same way as at the decoder side, resulting in the decoded prediction residual. The decoded prediction residual is added to the prediction. The result of that addition is fed into a deblocking filter, which provides the decoded video as its output.

The functions of different blocks of the H.264 encoder are as follows:

Transform: A 4x4 multiplier-free integer transform is used and the transform coefficients are explicitly specified in AVC and allow it to be perfectly invertible. Its hierarchical structure is a 4 x 4 Integer DCT and Hadamard transform. The Hadamard transform is applied only when (16x16) intra prediction mode is used with (4x4) integer DCT. MB size for chroma depends on 4:2:0, 4:2:2 and 4:4:4 formats (see Figure 2.6).

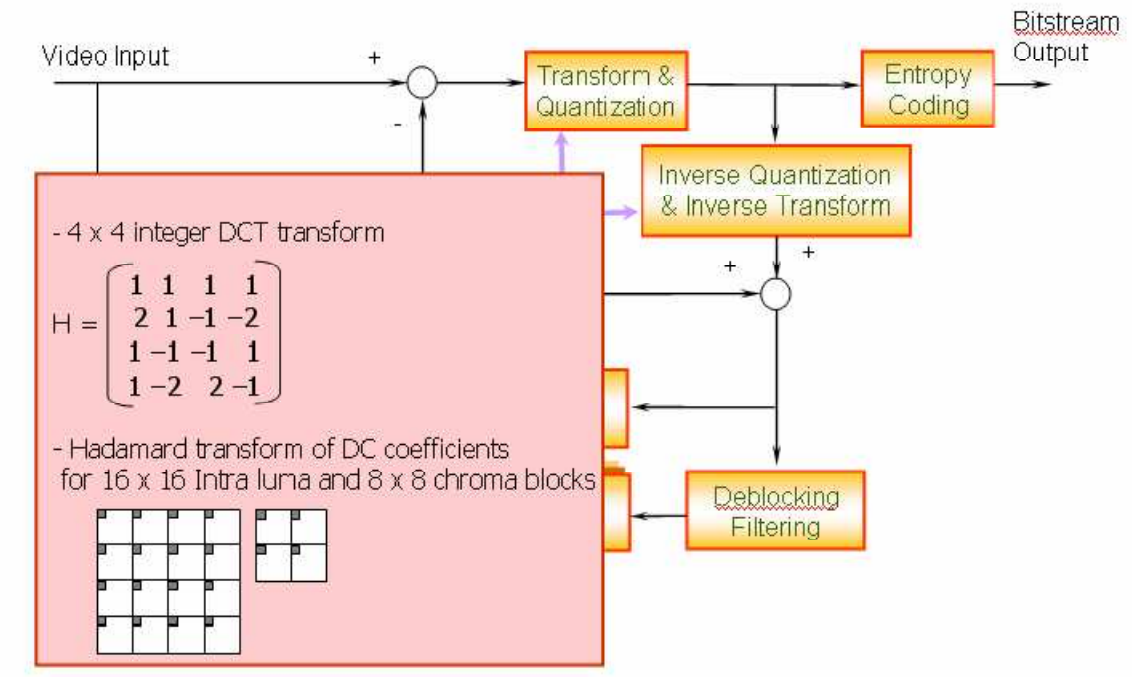


Figure 2.6: Block diagram emphasizing transform [3].

Figure 2.7 shows the assignment of the DC indices to the 4 x 4 luma block. The numbers 0, 1, …15 are the coding order for (4x4) integer DCT and (0,0), (0,1), (0,2), …, (3,3) are the DC coefficients of each 4x4 block.

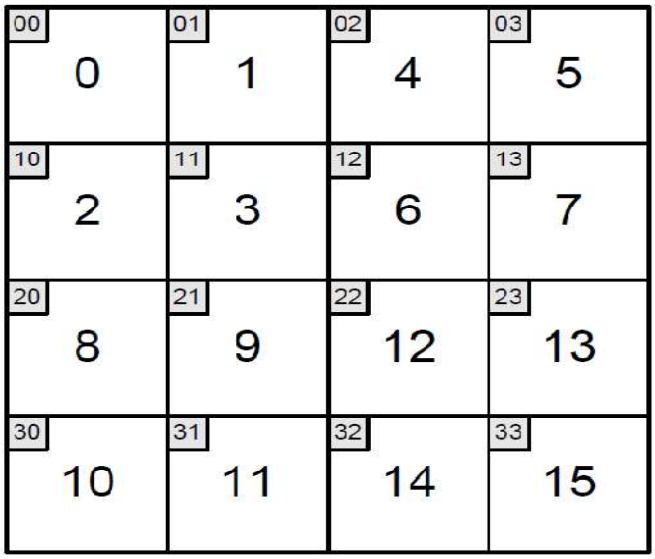


Figure 2.7: Assignment of indices of the DC (dark samples) to luma 4 x 4 blocks [3].

The 4x4 Integer DCT of X is given by:

$$Y = (C_f * C_f^T) \circledast E_f \tag{2.1}$$

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \otimes \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix}$$

$$a = \frac{1}{2}, b = \sqrt{\frac{2}{5}}, d = \frac{1}{2}$$

$$where \quad C_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}; \; E_f = \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix}$$

where X are the input pixels, Y are the output coefficients, $\circledast$ represents element by element multiplication.

The inverse 4x4 DCT can be represented by the following equation:

$$X' = C_i^T (Y \circledast E_i) C_i \quad (2.2)$$

$$where\ C_i = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}; E_i = \begin{bmatrix} a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \\ a^2 & \frac{ab}{2} & a^2 & \frac{ab}{2} \\ \frac{ab}{2} & \frac{b^2}{4} & \frac{ab}{2} & \frac{b^2}{4} \end{bmatrix}$$

The 16 DC coefficients of the 16 (4x4) blocks are transformed using the Walsh Hadamard transform and is given by:

$$Y_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_{D00} & x_{D01} & x_{D02} & x_{D03} \\ x_{D10} & x_{D11} & x_{D12} & x_{D13} \\ x_{D20} & x_{D21} & x_{D22} & x_{D23} \\ x_{D30} & x_{D31} & x_{D32} & x_{D33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) // 2 \quad (2.3)$$

where // represents rounding to the nearest integer.

The Walsh – Hadamard transform for 2x2 DC co-efficient for 4:2:0 chroma format can be represented as follows:

$$Y_D = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} DC_{00} & DC_{01} \\ DC_{10} & DC_{11} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (2.4)$$

DC_{ij} is the dc coefficient of the (4 x 4), $(i,j)^{th}$ block.

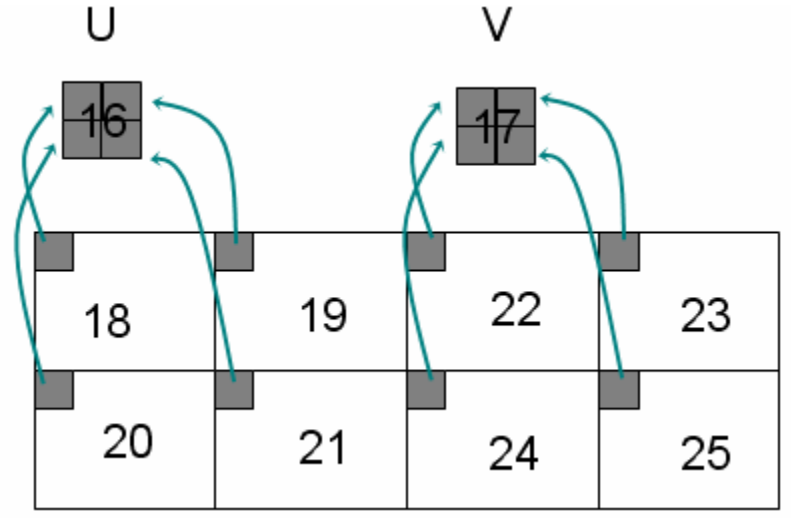


Figure 2.8: Chroma DC coefficients for 4x4 IntDCT for 4:2:2 and 4:4:4 chroma formats [3].

Scaling and Quantization: For the quantization of transform coefficients, H.264/AVC uses scalar quantization. One of 52 quantizers is selected for each macroblock by the Quantization Parameter (QP). The quantizers are arranged so that there is an increase of approximately 12.5% in the quantization step size when incrementing the QP by one. The quantized transform coefficients of a block are generally scanned in a zigzag fashion and transmitted using entropy coding methods. For blocks that are part of a macroblock coded in field mode, an alternative scanning pattern is used. The 2×2 DC coefficients of the chroma component are scanned in raster-scan order. All transforms in H.264/AVC can be implemented using only additions to, and bit-shifting operations on, the 16-bit integer values. Figure 2.9 shows the transform, scaling and quantization blocks at the encoder part of H.264 / AVC.

Quantization and scaling at the encoder can be represented by the following equation:

$$B_{ij} = A_{ij}\, round\left(\frac{SF_{ij}}{Qstep}\right) \tag{2.5}$$

where *A* is the quantizer input, *B* refers to the quantizer output, *Qstep* is the quantization parameter and *SF* is the scaling term.

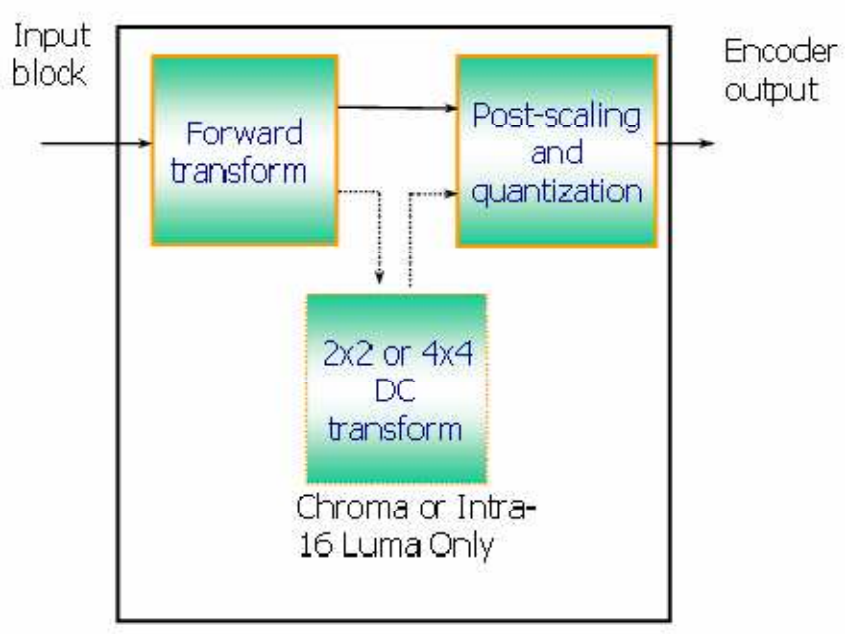


Figure 2.9: Transform, scaling and quantization at H.264 encoder [3].

Entropy coding: The H.264 AVC includes two different entropy coding methods for coding quantized transform coefficients, namely, CAVLC (Context-based Adaptive Variable Length Coding) and CABAC (Context-based Adaptive Binary Arithmetic Coding).

CAVLC handles the zero and +/- 1 coefficient based on the levels of the coefficients. The total numbers of zeros and +/-1 are coded. For the other coefficients, their levels are coded. Context adaptive VLC of residual coefficients make use of run-length encoding.

CABAC on the other hand, uses arithmetic coding. Also, in order to achieve good compression, the probability model for each symbol element is updated. Both motion vector and residual transform coefficients are coded by CABAC. CABAC increases compression efficiency by 10% over CAVLC, but it is computationally more intensive.

There are two types of scan orders used to read the residual data (quantized transform coefficients) namely, zig-zag and alternate scan (Figure 2.10).

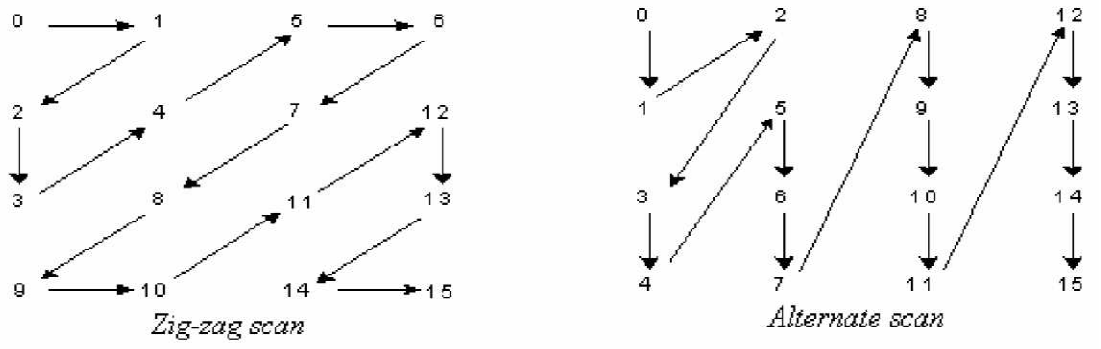


Figure 2.10: H.264 scan orders to read residual data [3].

Deblocking filter: Coarse quantization of the block-based image transform produces disturbing blocking artifacts at the block boundaries of the image. Motion compensation of the macroblock by interpolation of data from previous reference frames might never give a perfect match and discontinuities appear at the edges of the copied blocks. When the P-frames (Predicted) reference these images having blocky edges, the blocking artifacts further propagate to the interiors of the current block worsening the situation further.

The best way to deal with these artifacts is to filter the blocky edges to have a smoothed edge. This filtering process is known as the “deblock” filtering. The In-Loop deblock filter not only smoothens the blocky edges but also helps to increase the rate-distortion performance. After this, the frame decode process is carried out which ensures that all the top/left neighbors have been fully reconstructed and available as inputs for de-blocking the current macroblock. This is applied to all 4x4 blocks except at the boundaries of the picture. Filtering for block edges of any slice can be selectively disabled by means of flags [34]. Vertical edges are filtered first (left to right) followed by the horizontal edges (top to bottom) as shown in Figure. 2.11.

This filter operates on a macro-block after motion compensation and residual coding, or on a macro-block after intra-prediction and residual coding, depending whether the macroblock

is inter-coded or intra-coded. The results of the loop filtering operation are stored as a reference picture.

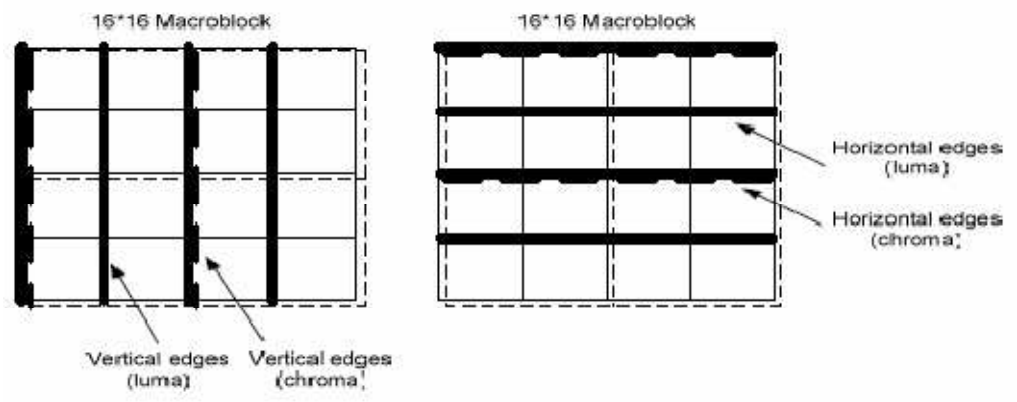


Figure 2.11: De-blocking filter process [34].

The decoder performs inverse quantization and pre-scaling as represented in the following equation:

$$A_{ij}^{'} = B_{ij} \cdot Qstep \cdot SF_{ij} \tag{2.6}$$

where *B* is the inverse quantizer input, *A'* is the inverse quantizer output and *Qstep* is the quantization parameter and *SF* is the scaling term.

Figure: 2.12 shows the inverse transform, scaling and quantization blocks at the decoder part of H.264 / MPEG-4 Part 10.

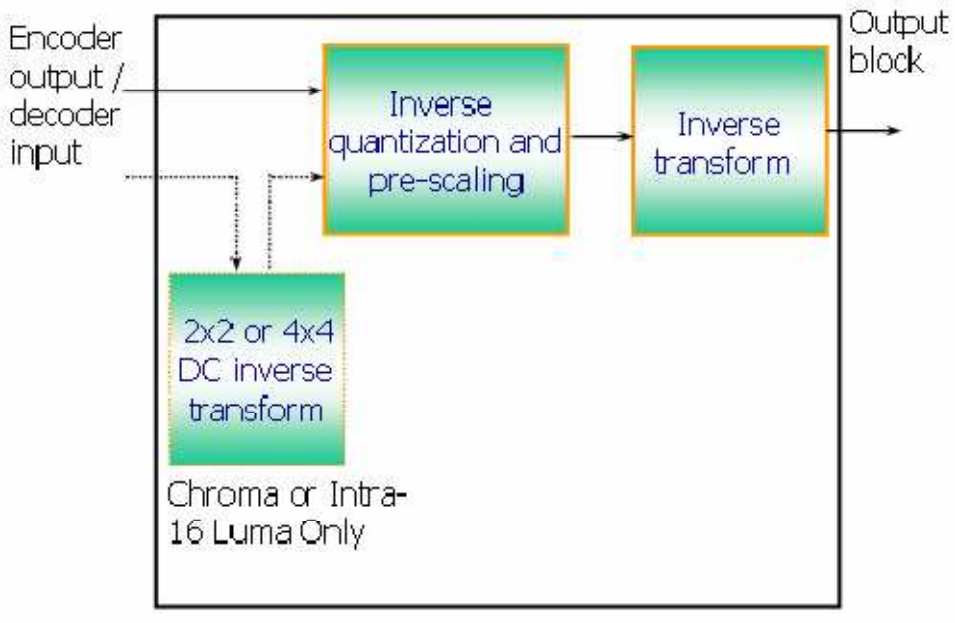


Figure 2.12: Inverse transform, scaling and quantization at H.264 decoder [3].

2.2 Video stream structure

The H.264/AVC video stream has a hierarchical structure shown in Figure 2.13. The different layers are explained next:

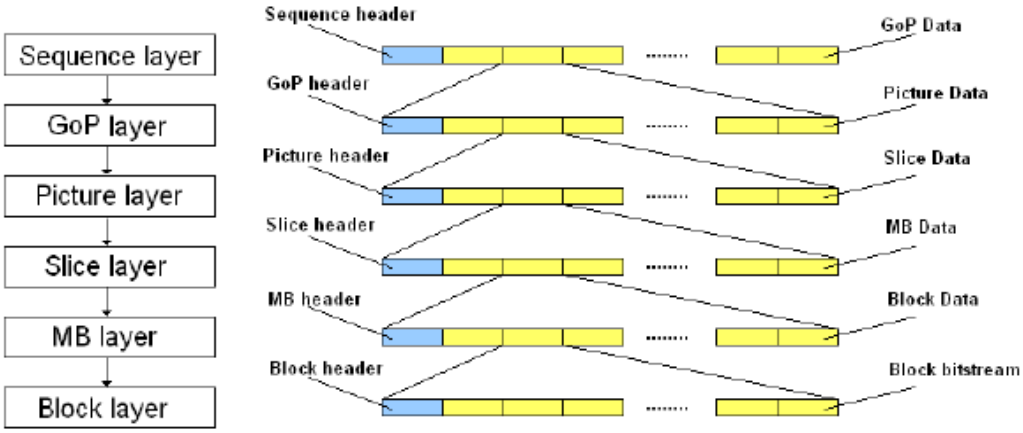


Figure 2.13: Structure of H.264/AVC video stream

H.264 provides a clearly-defined format or syntax for representing compressed video and related information. At the top level, an H.264 sequence consists of a series of “packets” or Network Adaptation Layer Units (NAL Units or NALUs). These can include parameter sets (containing key parameters that are used by the decoder to correctly decode the video data) and slices (coded video frames or parts of video frames). At the next level, a **slice** represents all or part of a coded video frame and consists of a number of coded macroblocks, each containing compressed data corresponding to a 16x16 block of displayed pixels in a video frame. Detailed explanation of different layer header information is as follows:

- Block layer: A block is an 8 x 8 array of pixels.
- Macroblock layer: Contains single MB. A MB consists of a number of blocks that depend upon the chroma pixel structure. In this thesis work 4:2:0 profile is been used.
- Slice layer: Slice is a sequence of MBs which are processed in the order of a raster scan when not using FMO. A picture may be split into one or several slices. Slices are self decodable, i.e. if an error occurs, it only propagates spatially within the slice. At the start of each slice the CAVLC is resynchronized.
- Picture layer: Pictures are main coding units of a video sequence. There are three types of frames:
    - Intra coded frame: coded without any reference to any other frames.
    - Predictive coded frame: coded as the difference from a motion compensated prediction frame, generated from an earlier I or P frame in the GOP.
    - Bi directional coded frame: coded as the difference from a bi-directionally interpolated frame, generated from earlier and later I or P frames in the sequence.
- Group of Pictures layer: Sequence of an I frame and temporally predicted frames until the next I frame. Allows random access to the sequence and provides refresh of the

picture after errors. If an error occurs, it will propagate only until the start of the next GOP.

- Sequence layer: This layer starts with the sequence header and ends with an end of sequence code. The header carries information about picture size, aspect ratio, number of frames and bit rate of the images contained within the encoded sequence.

2.3 Slice structure

The macroblocks are organized into slices. A picture is a collection of one or more slices in the H.264/AVC standard [8]. Each picture may be split into one or several slices as shown in Figure 2.14. The transmission order of macroblocks in the bit stream depends on the so called Macroblock Allocation Map (MAM), and it is not necessarily in raster scan order.

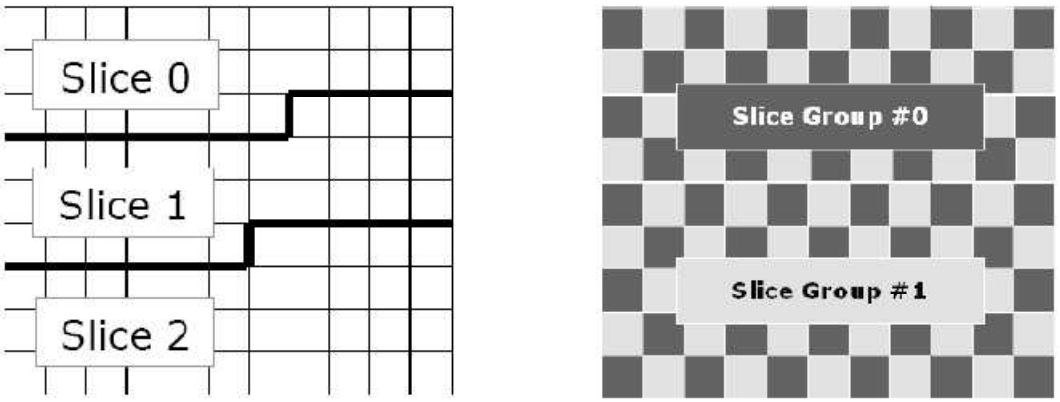


Figure 2.14: Subdivision of video frames [12].

Encoded video introduces slice units to make transmission packets smaller (compared to transmitting a whole frame as a packet). The probability of a bit error hitting a short packet is generally lower than for large packets [11], [12] and [15]. Moreover, short packets reduce the amount of lost information thereby limiting the error. Thus the error concealment methods can be applied in a more efficient way. Figure 2.15 illustrates the advantages of using slicing when an error occurs. Instead of concealing the whole frame, it just has to conceal the slice.

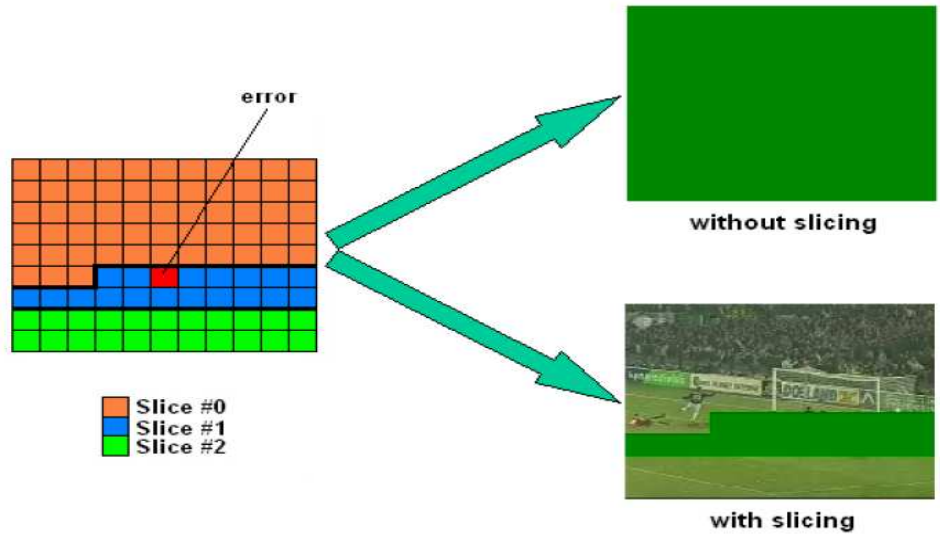


Figure 2.15: Error detection without and with slicing [12].

H.264/AVC supports five different slice-coding types. The simplest one is the I slice. In I slice, all macroblocks are coded without referring to other pictures 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 slices.

The transmission order of MBs in the bitstream depends on the so-called Macroblock Allocation Map and is not necessarily in raster-scan order if we use Flexible Macroblock Ordering (FMO). FMO modifies the way how pictures are partitioned into slices and MBs by utilizing the concept of slice group map, which is specified by the content of the picture parameter set and some information from slice headers. The macroblock to slice group map consists of a slice group identification number for each MB in the picture, specifying which slice group the associated MB belongs to. Each slice group can be partitioned into one or more slices, such that a slice is a sequence of MBs within the same slice group that is processed in the order of a raster scan within the set of MBs of a particular slice group.

Using FMO (shown in Figure 2.16), a picture can be split into many MB scanning patterns. Figure 2.16 illustrates the advantages of using different slicing techniques.

- One slice per frame: Is the simplest method, but it misses the advantages of slicing. This method also leads to huge packets that have to be segmented at the IP layer.
- Fixed number of MB per slice: The frame is divided into slices with the same number of MB. This results in packets with different lengths in bytes.
- Fixed number of bytes per slice: The frame is divided in slices with the same byte length. This results in packets with different number of MBs.
- Scattered slice: Every P MB (P is the number of different slices) belongs to one slice. The advantage is that a MB has always neighbors of different slice groups, so if one slice is lost, there are always possible interpolation errors with the neighbors. The disadvantages are loss of efficiency of spatial prediction, complexity and time delay.
- Rectangular slice structure: It consists of one or more “foreground” slice groups and a “leftover” slice group. It allows for coding of a region of interest to improve coding loss.

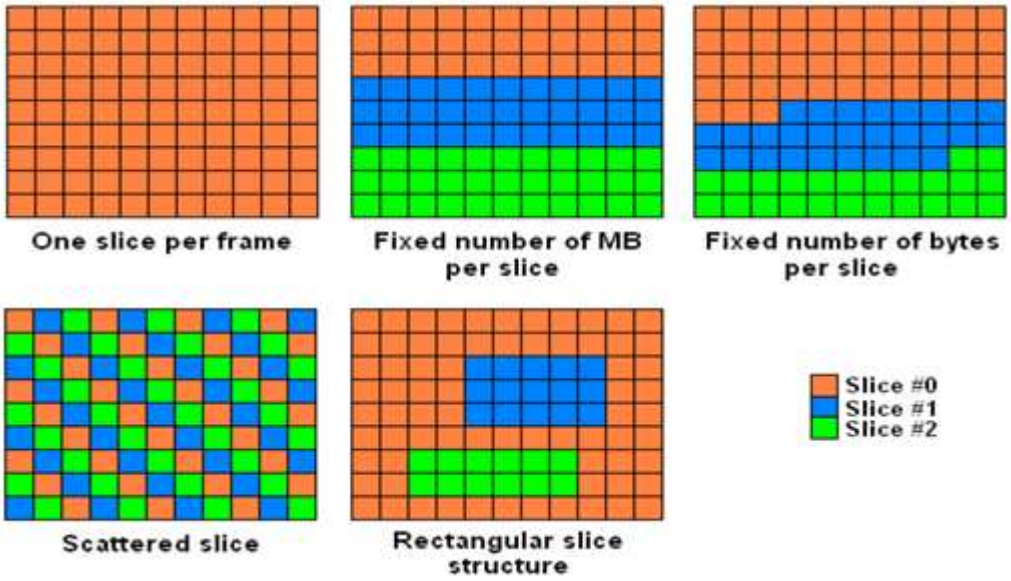


Figure 2.16: Slicing types in H.264/AVC [12].

2.4 H.264/AVC profiles

The H.264/AVC standard includes the following sets of capabilities, which are referred to as profiles. They target specific classes of applications [3]:

- Constrained Baseline Profile (CBP): Primarily for low-cost applications this profile is used widely in videoconferencing and mobile applications. It corresponds to the subset of features that are common between the Baseline, Main, and High Profiles
- Baseline Profile (BP): Primarily for low-cost applications that require additional error robustness, this profile is used rarely in videoconferencing and mobile applications, and it adds additional error resilience tools to the Constrained Baseline Profile. The importance of this profile is fading after the Constrained Baseline Profile has been defined.
- Main Profile (MP): This was originally intended as the mainstream consumer profile for broadcast and storage applications. The importance of this profile faded when the High profile was developed for these applications.
- Extended Profile (XP): This was intended as the streaming video profile. This profile has relatively high compression capability. It has some extra tricks for robustness to data losses and server stream switching.
- High Profile (HiP): This is the primary profile for broadcast and disc storage applications, particularly for high-definition television applications. This is the profile adopted into HD DVD and Blu-ray Disc.
- High 10 Profile (Hi10P): Going beyond today's mainstream consumer product capabilities, this profile builds on top of the High Profile, adding support for up to 10 bits per sample of decoded picture precision.
- High 4:2:2 Profile (Hi422P): This profile primarily targets professional applications that use interlaced video. It builds on top of the High 10 Profile, adding support for the 4:2:2

chroma subsampling format while using up to 10 bits per sample of decoded picture precision.

- High 4:4:4 Predictive Profile (Hi444PP): This profile builds on top of the High 4:2:2 Profile, supporting up to 4:4:4 chroma sampling, up to 14 bits per sample, and additionally supporting efficient lossless region coding and the coding of each picture as three separate color planes.

In addition, the standard contains four additional *all-Intra profiles*, which are defined as simple subsets of other corresponding profiles. These are mostly for professional (e.g., camera and editing system) applications:

- High 10 Intra Profile: The High 10 Profile constrained to all-Intra use.
- High 4:2:2 Intra Profile: The High 4:2:2 Profile constrained to all-Intra use.
- High 4:4:4 Intra Profile: The High 4:4:4 Profile constrained to all-Intra use.
- CAVLC 4:4:4 Intra Profile: The High 4:4:4 Profile constrained to all-Intra use and to CAVLC entropy coding (i.e., not supporting CABAC).

The common coding parts for the profiles are listed below [3]:

- I slice (Intra-coded slice): coded by using prediction only from decoded samples within the same slice.
- P slice (Predictive-coded slice) : coded by using inter prediction from previously decoded reference pictures, using at most one motion vector and reference index to predict the sample values of each block.
- CAVLC (Context-based Adaptive Variable Length Coding) for entropy coding.

The common coding parts for the baseline profile are listed below:

- Common parts: I slice, P slice, CAVLC.

- FMO Flexible macro block order: macro-blocks may not necessarily be in the raster scan order. The map assigns macro-blocks to a slice group.
- ASO Arbitrary slice order: the macro-block address of the first macro-block of a slice of a picture may be smaller than the macro-block address of the first macro-block of some other preceding slice of the same coded picture.
- RS Redundant slice: This slice belongs to the redundant coded data obtained by same or different coding rate, in comparison with previous coded data of same slice.

The common coding parts for the main profile are listed below:

- Common parts: I slice, P slice, CAVLC.
- B slice (Bi-directionally predictive-coded slice) : the coded slice by using inter prediction from previously-decoded reference pictures, using at most two motion vectors and reference indices to predict the sample values of each block.
- Weighted prediction: scaling operation by applying a weighting factor to the samples of motion-compensated prediction data in P or B slice.
- CABAC (Context-based Adaptive Binary Arithmetic Coding) for entropy coding.

The common coding parts for the extended profile are listed below:

- Common parts : I slice, P slice, CAVLC.
- SP slice : specially coded for efficient switching between video streams, similar to coding of a P slice.
- SI slice: switched, similar to coding of an I slice.
- Data partition: the coded data is placed in separate data partitions, each partition can be placed in different layer unit.
- Flexible macro-block order (FMO), arbitrary slice order (ASO).
- Redundant slices (RS), B slice.

- Weighted prediction.

Table 2.1: H.264 / MPEG-4 Part 10 profile specifications [3].

	Baseline	Main	Extended	High
I & P Slices	X	X	X	X
Deblocking Filter	X	X	X	X
¼ Pel Motion Compensation	X	X	X	X
Variable Block Size (16x16 to 4x4)	X	X	X	X
CAVLC/UVLC	X	X	X	X
Error Resilience Tools – Flexible MB Order, ASO, Red. Slices	X		X	
SP/SI Slices			X	X
B Slice		X	X	X
Interlaced Coding		X	X	X
CABAC		X		X
Data Partitioning			X	

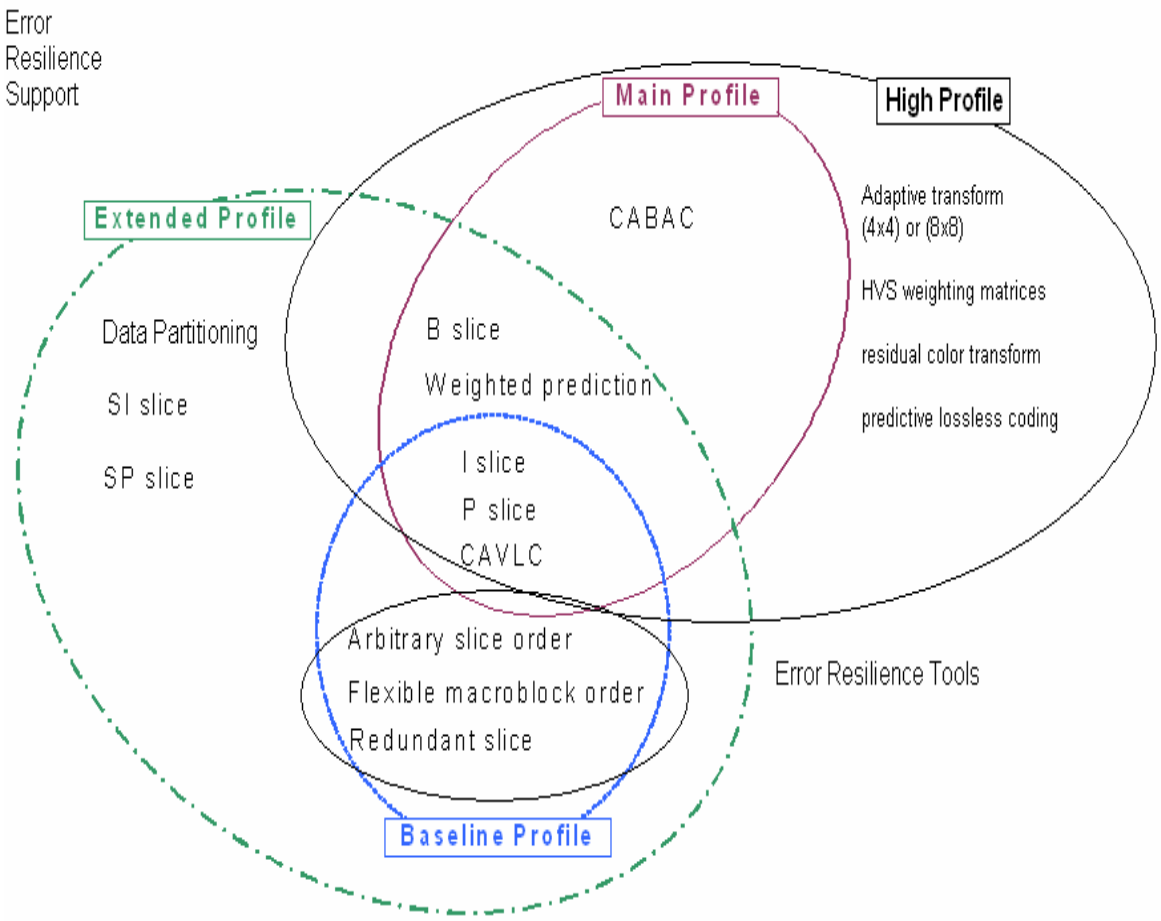


Figure 2.17: Specific coding parts for H.264 profiles [3].

CHAPTER 3

ENCAPSULATION OF VIDEO DATA THROUGH NETWORK LAYERS

The H.264/AVC standard consists of two layers, the video coding layer (VCL) and the network abstraction layer (NAL) as shown in Figure 3.1. The VCL specifies an efficient representation for the coded video data. It is designed to be as network independent as possible. The coded video data is organized into NAL units, each of which is 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 [5, 12]. The payload data in the NAL unit is interleaved if necessary with emulation prevention bytes, which are bytes with a specific value inserted to prevent a particular pattern of data called a start code prefix from being accidentally generated inside the payload. The NAL unit structure definition specifies a generic format for use in both packet oriented and bits stream oriented transport systems, and a series of NAL units generated by an encoder referred to as a NAL unit stream. The NAL adapts the bit strings generated by the VCL to various network and multiplex environments and covers all syntactical levels above the slice level. In particular, it includes mechanisms for:

- The representation of the data that is required to decide individual slices.
- The start code emulation prevention
- The framing of the bit strings that represent coded slices for the use over byte oriented networks.

As a result of this effort, it has been shown that NAL design specified in the recommendation is appropriated for the adaptation of H.264 over RTP/UDP/IP [12].

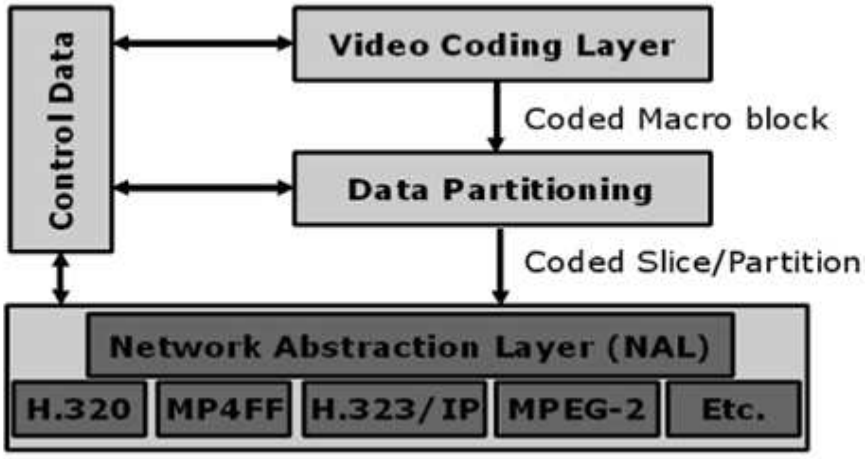


Figure 3.1: Layer structure of H.264/AVC encoder [14]

The number and the order of macroblocks, which can be sent in one NAL unit is defined by the slice mode parameter: It is possible to set all macroblocks in the frame to one slice, or to choose a constant number of macroblocks per slice or constant number of bytes per slice.

A slice can also be divided according to its video content into three partitions: Data partition A (DPA), which includes header information, sub block format and Intra prediction modes in case of I-slices or motion vectors in case of P and B-slices. Data partition B (DPB), which includes the Intra residuals. Data partition C (DPC), which includes the Inter residuals.

The H.264 specifications define several NAL unit types according to the type of information included as shown in Figure 3.2.

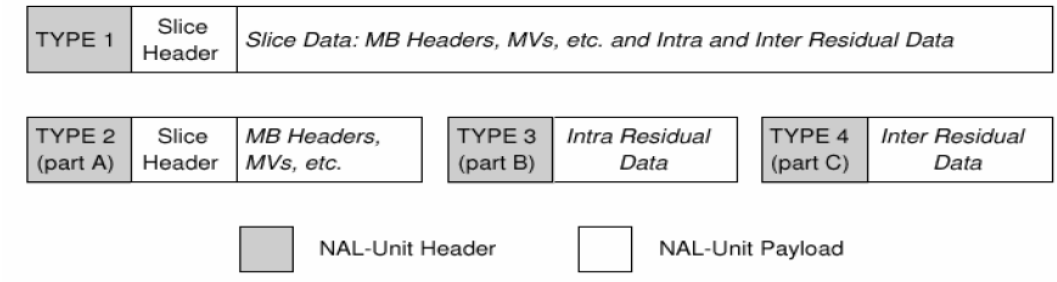


Figure 3.2: Data partitioning types of slices [19]

In the video transmission, the order in which the NAL units have to be sent is fixed. The first NAL unit to be sent is the sequence parameter set (SPS) followed by the picture parameter set (PPS). Both SPS and PPS include some parameters which have been set in the encoder configuration for all pictures in the video sequence, for example: entropy coding mode flag, number of reference index, weighted prediction flag, picture width in MB, picture height in MB and number of reference frames.

The next NAL unit is the Instantaneous Decoder Refresh (IDR). After receiving a NAL unit of this type all the buffers have to be deleted. An IDR frame may only contain I slice without data portioning. IDR frames are usually sent at the start of the video sequence. All NAL units following the IDR have NAL type slice or one of DPA/DPB/DPC. Figure 3.3 shows the NAL units when no data portioning is used.

Figure 3.3: NAL units order.

For the streaming video services over the mobile technologies, the IP packet switched communication is of major interest, which uses real time transport protocol (RTP). Each NAL unit regardless of its type is encapsulated in the RTP/UDP/IP packet by adding header information of each protocol to the NAL unit as shown in Figure 3.4. IP header is 20 or 40 bytes long, depending on the protocol version and contains the information about the source and destination IP address. The UDP header is 8 bytes long and contains the CRC and length of the encapsulated packet. RTP header is 12 bytes long and contains sequence number and time stamps. Figure 3.5 illustrates the encapsulation of the video data starting at Network Adaptation Layer (NAL) down to the Physical Layer [12].

IP	UDP RTP	NAL unit

Figure 3.4: Encapsulation of NAL unit in RTP/UDP/IP.

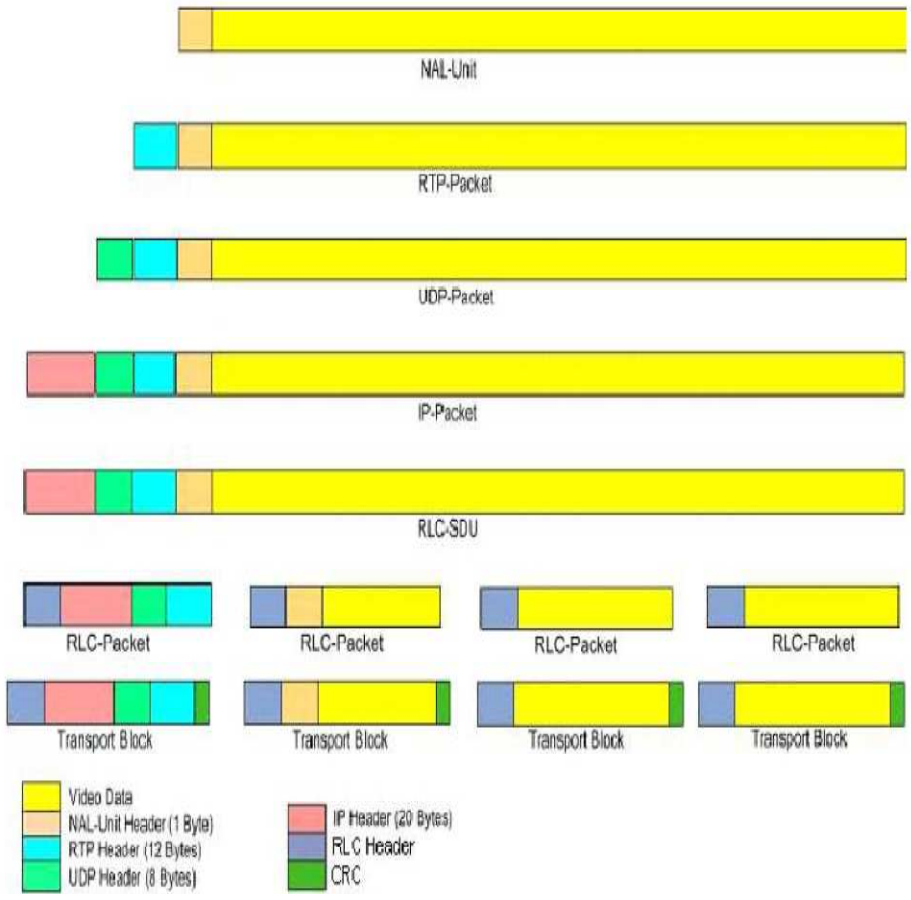


Figure 3.5: Encapsulation of video data through protocol stack.

CHAPTER 4

ERROR PROPAGATION

The visual artifact caused by the bit stream error has different shapes and ranges depending on which part of video data stream is affected by the transmission error. Therefore these artifacts can be described in 2 levels: Slice level and GOP level

4.1 Slice level

In the slice level these artifacts are caused by either desynchronization of the variable length code or the loss of the reference in a spatial prediction.

4.1.1 Variable length code

The quantized transform coefficients are entropy coded using a variable length code (VLC) which means that the codewords have variable lengths [16]. The advantages of this kind of code consist in the fact that they are more efficient in the sense of representing the same information using fewer bits on average, reducing therefore the bit rate. That is possible if some symbols are more probable than others. The most frequent symbols will correspond to the shorter codewords, and the rare symbols will correspond to the longer codewords. However, variable length codes between the codewords may be determined in a wrong way, and the decoding process may desynchronize. Figure 4.1 describes how just one erroneous bit shown in red can desynchronize the whole sequence.

Figure 4.1: Example of a VLC desynchronization

4.1.2 Spatial prediction

The H.264/AVC performs intra prediction in the spatial domain. Even for an intra picture, every block of data is predicted from its neighbors before being transformed and coefficients generated for inclusion in the bit stream. As a first step in coding of a macroblock in intra mode, spatial prediction is performed on either 4x4 or 16x16 luminance blocks. Although, in principle, 4x4 block prediction will offer more efficient prediction compared to a 16x16 block, in reality, taking into account the mode decision overhead, sometimes the 16x16 block based prediction may offer overall better coding efficiency. Figure 4.2 shows two types of luminance intra coding.

Num	Intra 4x4 Pred Mode
0	vertical
1	horizontal
2	dc
3	diagonal_down_left
4	diagonal_down_right
5	vertical_right
6	horizontal_down
7	vertical_left
8	horizontal_up

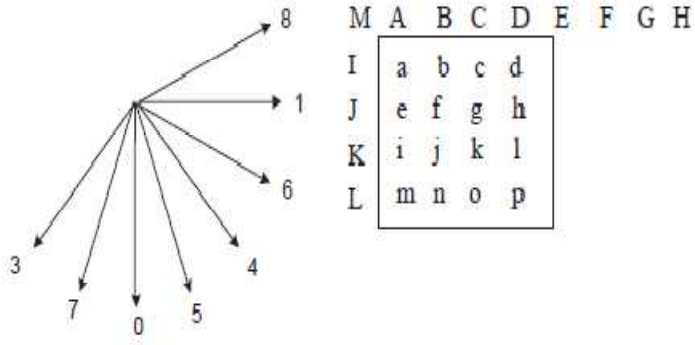


Figure 4.2: Left: Intra 4x4 predictions are conducted for samples a-p of a block by 9 different modes. Right: 8 prediction directions for Intra 4 x 4 prediction. [17].

Num	Intra 16x16 Pred Mode
0	vertical
1	horizontal
2	dc
3	plane

Figure 4.3: Intra 16x16 prediction modes. [17]

There are two 8x8 blocks of chroma in a macroblock one corresponding to each of the components, Cb and Cr. Each 8x8 block of chroma is subdivided into 4, 4x4 blocks such that

each 4x4 block depending on its location uses a pre-fixed prediction using decoded pixels of corresponding chroma component. Figure 4.4 illustrate variable size of macroblocks.

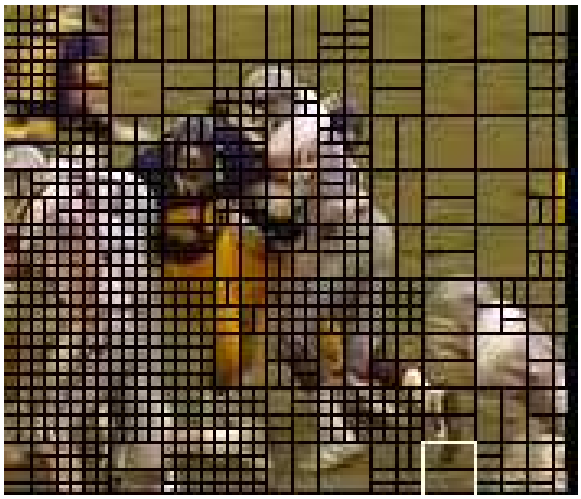


Figure 4.4: Frame divided into multiple macroblocks of 16 x 16, 8 x 8, 8 x 4, 4 x 8 and 4 x 4 variable sizes to represent different coding profiles.

Inter prediction: The inter prediction block includes both motion estimation (ME) and motion compensation (MC). It generates a predicted version of a rectangular array of pixels, by choosing similarly sized rectangular arrays of pixels from previously decoded reference pictures and translating the reference arrays to the positions of the current rectangular array. Figure. 4.5 depicts inter-prediction.

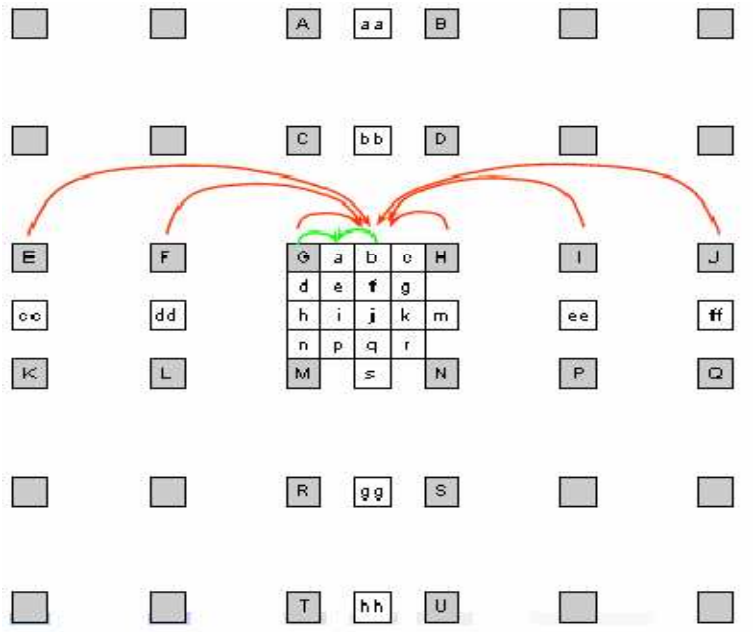


Figure 4.5: Inter prediction in H.264 [3].

In Figure 4.5, a half-pel is interpolated from neighboring integer-pel samples using a 6-tap Finite Impulse Response filter with weights (1, -5, 20, 20, -5, 1) / 32, quarter-pel is produced using bilinear interpolation between neighboring half- or integer-pel samples.

In the AVC, the rectangular arrays of pixels that are predicted using MC can have the following sizes: 4x4, 4x8, 8x4, 8x8, 16x8, 8x16, and 16x16pixels. The translation from other positions of the array in the reference picture is specified with quarter pixel precision. In case of 4:2:0 format, the chroma MVs have a resolution of 1/8 of a pixel. They are derived from transmitted luma MVs of 1/4 pixel resolution, and simpler filters are used for chroma as compared to luma. Figure. 4.6 illustrates the partitioning of the macroblock for motion compensation and Figure 4.7 depicts sub-pel motion compensation block of the H.264/AVC encoder.

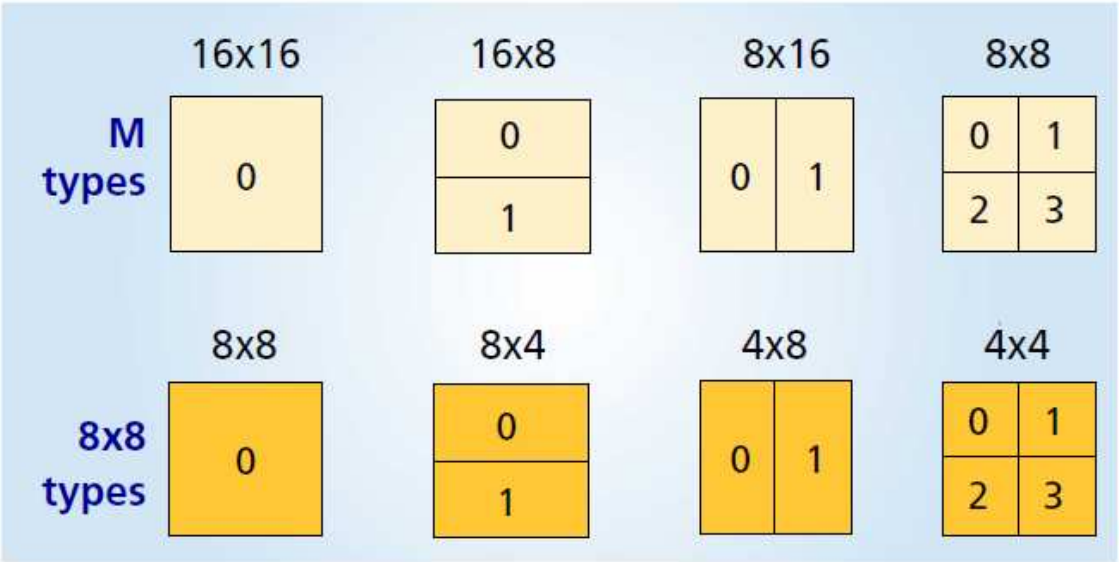


Figure 4.6: Segmentations of the macro-block for motion compensation [3].

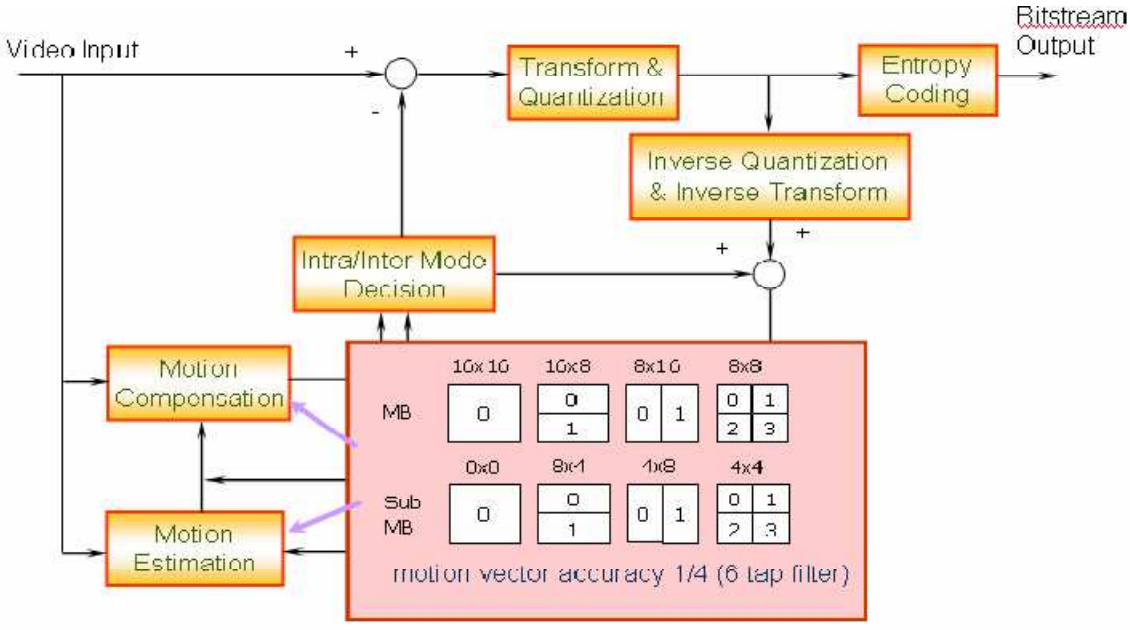


Figure 4.7: Block diagram emphasizing sub-pel motion compensation [3].

H.264/AVC standard supports multi-picture motion-compensated prediction. That is, more than one prior-coded picture can be used as a reference for motion-compensated prediction as shown in Figure 4.8. In addition to the motion vector, the picture reference parameters (Δ) are also transmitted. Both the encoder and decoder have to store the reference pictures used for Inter-picture prediction in a multi-picture buffer. The decoder replicates the multi-picture buffer of the encoder, according to the reference picture buffering type and any memory management control operations that are specified in the bit stream [35].

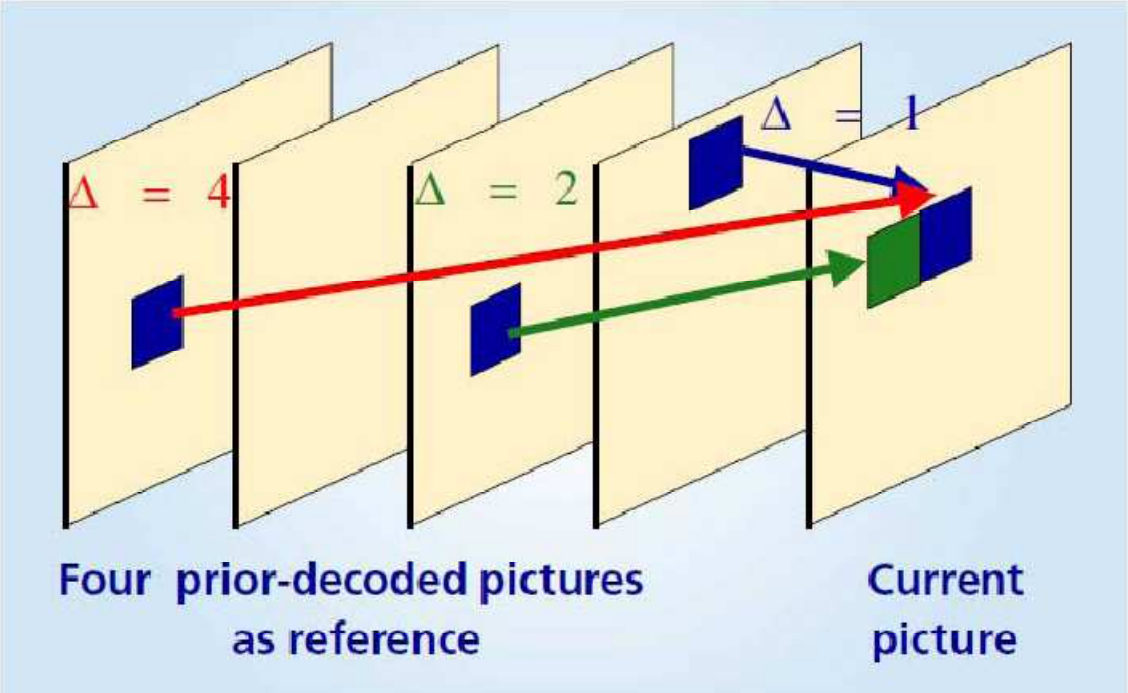


Figure 4.8: Multi-frame motion compensation in H.264 [35].

The H.264/AVC decoder takes in the encoded bit stream as input and gives raw YUV (Y-Luminance, (U, V)-Chrominance) video frames as output. The header or syntax information and slice data with motion vectors is extracted by the entropy decoder block through which the bit stream is passed. Next the residual block data is extracted by means of inverse scan and inverse quantizer. An inverse transform is carried out on all the blocks in order to map them

from the transform domain to the pixel domain. A predicted block is formed using motion vectors, and previously decoded reference frames if the block is found to be inter coded. Then the predicted block and residual block are combined to reconstruct the complete frame. This decoded frame is then presented to the user after it is passed through a de-blocking filter.

4.2 GOP level

Due to the temporal and spatial predictions of the images, the image distortion caused by a erroneous MB is not restricted to that MB. Since MBs are spatially and/or temporally dependent on neighboring MBs, the errors can also propagate in time (in following frames) and in space (the same frame). Error propagation represents a problem for error concealment because if the error concealed picture differs from the original picture, the error will propagate until the next I frame occurs or until the beginning of the next GOP. If more frames per GOP are used to improve compression, there will be degradation in video quality since the error can propagate over more frames.

CHAPTER 5

QUALTIY METRICES

Digital videos are subject to a wide variety of distortions during transmission, compression, processing and reproduction, any of which may result in a degradation of visual quality. For applications in which videos are ultimately to be viewed by human beings, the only correct method of quantifying visual video quality is through subjective evaluation. However, subjective evaluation is usually too inconvenient, time-consuming and expensive. That explains why there is a increasing popularity to develop objective quality measurement techniques that can predict perceived image and video quality automatically.

An objective image quality metric can play a variety of roles in image processing applications. First, it can be used to dynamically monitor and adjust image quality. For example, a network digital video server can examine the quality of video being transmitted in order to control and allocate streaming resources. Second, it can be used to optimize algorithms and parameter settings of image processing systems. For instance, in a visual communication system, a quality metric can assist in the optimal design of pre-filtering and bit assignment algorithms at the encoder and of optimal reconstruction, error concealment, and post-filtering algorithms at the decoder. Third, it can be used to benchmark image processing systems and algorithms.

Most widely used quality metric is the mean squared error (MSE), computed by averaging the squared intensity differences of distorted and reference image pixels, along with the related quantity of peak signal to noise ratio (PSNR). MSE and PSNR are widely used because they are simple to calculate and have clear physical meanings, and are mathematically easy to deal with for optimization purposes. However they have been widely criticized as well

for not correlating well with perceived quality measurement. Therefore, a distortion measure that is based on human perception is more appropriate for picture quality estimation. A great deal of effort has gone into the development of quality assessment methods that take advantage of known characteristics of the human visual system (HVS) like blockiness and blurriness or a measure of structural similarity (SSIM) [21].

5.1 Peak signal to noise ratio (PSNR)

In scientific literature it is common to evaluate the quality of reconstruction of a frame by analyzing its peak signal to noise ratio (PSNR). There are different ways of representing PSNR, One of the effective way of calculating PSNR is by dividing the frame in a graph with luminance and the two chrominance. The unambiguous way is to take only the luminance component of the YUV color space (Y-PSNR) Y-PSNR is the PSNR based on luminance only. This is also sufficient for the error concealment methods that handle chrominance in the same way as luminance, since chrominance is smoother and thus, in general easier to conceal.

Joint Model 13.2 (JM 13.2) [27] Reference Software outputs PSNR for every component of the YUV color space (Y-PSNR, U-PSNR and V-PSNR, corresponding to the luminance, chrominance B, chrominance R respectively) for every frame k

$$PSNR_k{}^{(c)} = 10.log_{10}\frac{255^2}{MSE_k{}^{(c)}}[dB] \qquad (5.1)$$

MSE is the mean square error for the component for which PSNR is calculated. It is defined as

$$MSE_k^{(c)} = \frac{1}{M.N}\sum_{i=1}^{N}\sum_{j=1}^{M}[F\,(i,j) - F_o(i,j)]^2 \qquad (5.2)$$

Where MxN is the size of the frame, $F_o(i,j)$ is the reconstructed frame and $F\,(i,j)$ is the original frame (uncompressed and without losses) of the color component (c).

5.2 Structural similarity (SSIM)

The main function of the human visual system (HVS) is to extract structural information from the viewing field, for which it is highly adapted for this purpose. Therefore, a measurement of structural information loss can provide a good approximation to perceived image distortion. SSIM compares local patterns of pixel intensities that have been normalized for luminance and contrast. The luminance of the surface of an object being observed is the product of the illumination and the reflectance, but the structures of the objects in the scene are independent of the illumination. Consequently, the structural information in an image can be determined by separating the influence of the illumination. The structural information in an image can be defined as those attributes that represent the structure of objects in the scene, independent of the average luminance and contrast. The system diagram of the proposed quality assessment system is shown in Figure 5.1

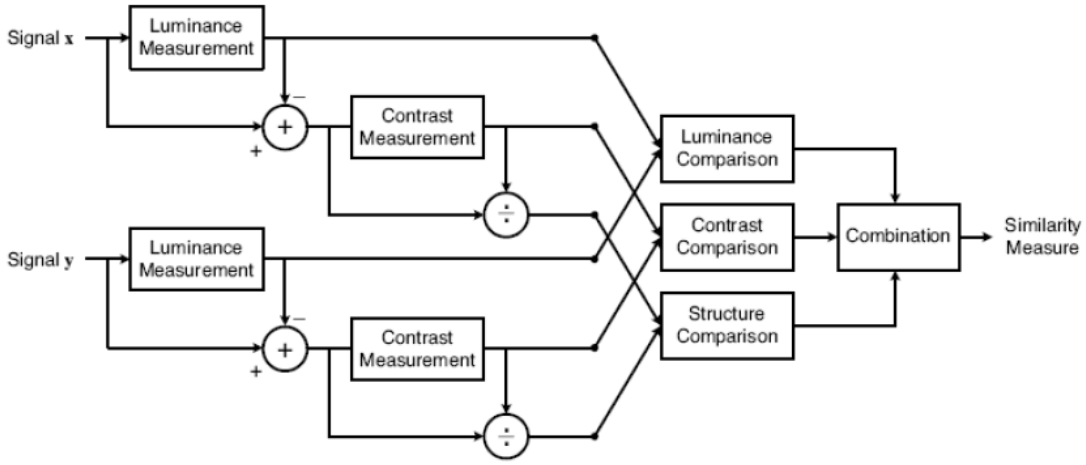


Figure 5.1: Diagram of the structural similarity (SSIM) measurement system [24].

Let x and y be two non-negative signals that have been aligned with each other (e.g., two image patches extracted from the same spatial location from two images being compared, respectively), and let $\mu_x, \mu_y, \sigma_x^2, \sigma_y^2 \; and \; \sigma_{xy}$ be the mean of x, the mean of y, the variance of

x, the variance of y and the variance of x and y respectively. Approximately, $\mu_x$and $\sigma_x$can be viewed as estimates of the luminance and contrast of x and σ_{xy} measures the tendency of x and y to vary together (An indication of structural similarity). The luminance, contrast and structure comparison measures are given as follows:

$$I(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{5.3}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{5.4}$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \tag{5.5}$$

where C_1, C_2 and C_3 are small constants given by $\mathbf{C_1} = (\mathbf{K_1}\,\mathbf{L})^2$,$\mathbf{C_2} = (\mathbf{K_2}\,\mathbf{L})^2$ and $\mathbf{C_3} = {}^{\mathbf{C_2}}/_{\mathbf{2}}$ respectively. L is the dynamic range of the pixel values (L = 255 for 8 bits/pixel gray scale images), and $\mathbf{K_1}$ <<1 and $\mathbf{K_2}$ <<1 are two scalar constants. The general form of the SSIM index between signals x and y is defined as:

$$SSIM\ (x,y) = [I(x,y)]^{\alpha}\,.\,[c(x,y)]^{\beta}\,.\,[s(x,y)]^{\gamma} \tag{5.6}$$

where α, β and γ are parameters to define the relative importance of the three components. Specifically, setting $\alpha = \beta = \gamma = 1$, the resulting SSIM index is given by:

$$SSIM\ (x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{5.7}$$

which satisfies the following conditions:

1. Symmetry: $SSIM\ (x,y) = SSIM\ (y,x)$

2. $SSIM\ (x, y) \leq 1$

3. $SSIM\ (x, y) = 1$ If and only if $x = y$

Here one of the image signals being compared to have perfect quality, then the SSIM index provides a quantitative measurement of the quality of the other image signal.

CHAPTER 6

ERROR CONCEALMENT

The loss of transmitted data packets influences the quality of the received video. This problem is caused by the band limited channel used by the mobile communication networks. Since the real time transmission of video stream limits the channel delay, it is not possible to retransmit all erroneous or lost packets. Therefore there is a need for post processing methods, which try to reduce the visual artifacts caused by bit stream error after locating the missing or defective parts of video data [11]. Error concealment methods which will be implemented on the receiver side restore the missing and corrupt video content by using the previously decoded video data. The error concealment benefits from the spatial and temporal correlations between the video blocks within one frame or more than one frame within the video sequence. Therefore the error concealment methods are implemented both in the spatial domain and time domain. The spatial domain based error concealment uses the video information from the neighboring blocks to restore the missing pixels within a specified area. The time domain based error concealment uses the video information from the blocks lying in the previous and next frames to restore the missing pixels within a specified area [15] and [16].

There are some assumptions adopted in this thesis to concentrate and limit the efforts on the presentation of the error concealment methods:

- The missing part of a video content is limited to one macroblock
- The location of the missing macroblocks is known
- Features like data partitioning belonging to one macroblock such as motion vectors, prediction mode and residuals are lost.

6.1 Joint Model (JM) Reference Software

There was a compilation error that was been encountered while using error concealment methods built in JM 13.2 [27] reference software. It was giving us the following assertion error:

Assertion failed: numofPredblocks !=0

The problem of encoding the frames was fixed as follows:

By working on this routine by changing the input profile into baseline method by making these changes in the configuration file present in the encoder, change the subroutine.

Some of the error concealment algorithms implemented in the decoder of the JM 13.2 [27] are explained briefly:

6.2 Error concealment in spatial domain

The spatial redundancy in image and video signals is always present. Here the interpixel difference between adjacent pixels for an image is determined. The interpixel difference is defined as the average of the absolute difference between a pixel and its four surrounding pixels. This property has been exploited to perform error concealment. All error concealment methods in the spatial domain are based on the same idea which says that the pixel values within the damaged macroblocks can be recovered by a specified combination of the pixels surrounding the damaged macroblocks.

6.2.1 Weighted averaging

The first step done to implement spatial based error concealment was to interpolate the pixel values within the damaged macroblock from the four next pixels in its four 1-pixel wide boundaries. This method is known as ‘weighted averaging’ [28], because the missing pixel values can be recovered by calculating the average pixel values from the four pixels in the four 1-pixel wide boundaries of the damaged macroblock weighted by the distance between the

missing pixel and the four macroblocks boundaries (upper, down, left and right boundaries) as shown in Figure 6.1.

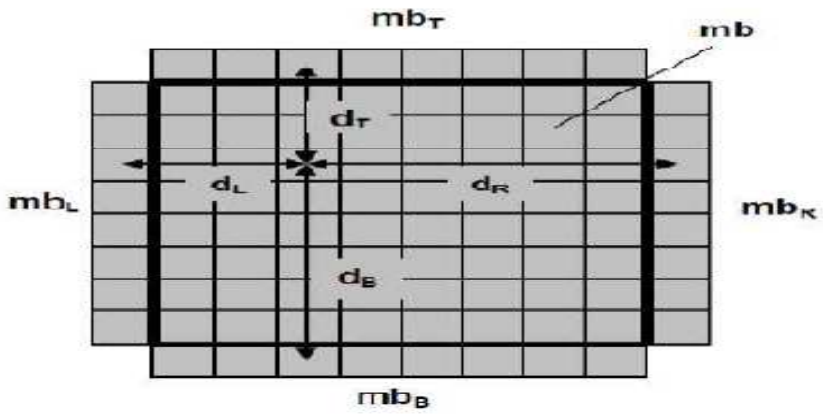


Figure 6.1: Weighted Averaging

Using a macroblocks with NxN pixel size, the weighted averaging (macroblock based) can be described as follows:

$$mb(i,k) = \frac{1}{d_L + d_R + d_T + d_B}[d_R mb_L(i,2N) + d_L mb_R(i,l) + d_B mb_T(2N,k) + d_T mb_B(l,k) \quad (6.1)$$

where $i,k = 1,2,3,\ldots\ldots\ldots\ldots\ldots\ldots\ldots N$.

d_L : distance between the interpolated pixel and the nearest pixel $mb_L = mb(i,0)$ in the left boundary.

d_R : distance between the interpolated pixel and the nearest pixel $mb_R = mb(i,\ N+1)$ in right boundary.

d_T : distance between the interpolated pixel and the nearest pixel $mb_T = mb(0,j)$ in top boundary.

d_B : distance between the interpolated pixel and the nearest pixel $mb_B = mb(N+1,j)$ in bottom boundary.

N: Size of the block.

mb : macroblock

Used symbols are seen in Figure 6.1.

Another way to implement weighted averaging is called block based weighted averaging [28]. The damaged macroblock is split into four independent blocks; each pixel within a block is interpolated from two pixels in its two nearest boundaries. When using a macroblock with 2Nx2N pixels the weighted averaging (block based) can be described as follows:

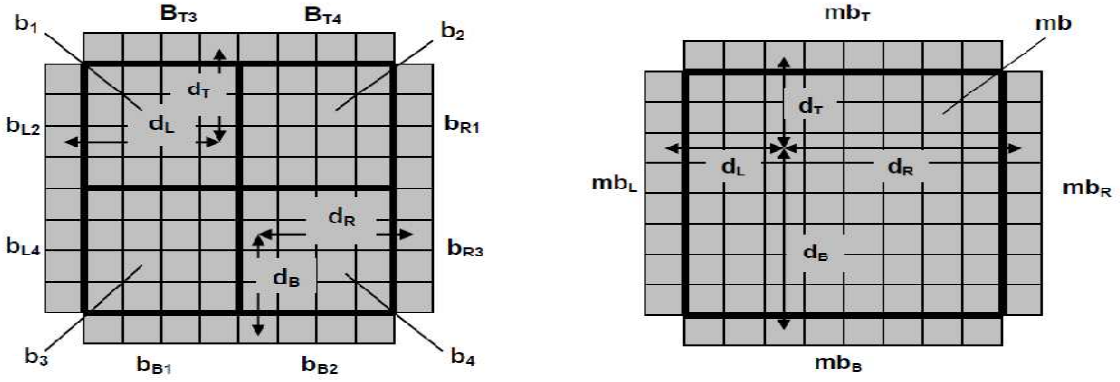
$$b_1(i,k) = \frac{d_T b_{L2}(i,N) + d_L b_{T3}(N,k)}{d_L + d_T} \tag{6.2}$$

$$b_2(i,k) = \frac{d_T b_{R1}(i,1) + d_R b_{T4}(N,k)}{d_R + d_T} \tag{6.3}$$

$$b_3(i,k) = \frac{d_B b_{L4}(i,N) + d_L b_{B1}(1,k)}{d_L + d_T} \tag{6.4}$$

$$b_4(i,k) = \frac{d_B b_{R3}(i,1) + d_R b_{B2}(1,k)}{d_R + d_B} \tag{6.5}$$

Where $i,k = 1,2,3,\ldots\ldots\ldots\ldots\ldots\ldots N$.



(a) (b)

Figure 6.2: Weighted Averaging: a) block based, b) macroblock based

The weighted averaging method - based on macroblock approach showed good results in cases where the missing macroblock lies within a smooth area. An example is when a picture with a sky view or plain background is considered (see Figs. 6.2 and 6.4). On the other hand the block based weighted averaging method does not guarantee the smoothness of the recovered macroblock and shows a slight blocking effect (see figure 6.3 and 6.5).

Otherwise this method is more efficient than the macroblock based method if the missing macroblock consists of two or more parts, where each part belongs to a different smooth area. Figure 6.2 shows an example of this case, where the missing macroblock lies between the black smooth area and blue smooth area. Using macroblock based weighted averaging does not guarantee the smoothness property of video signal along the boundaries of the missing macroblock.

a b c

Figure 6.3: Recovery of the damaged macroblock in Akiyo video sequence (a) distorted image lying within a smooth area b) macroblock based weighted averaging applied on a blue smooth area; c) block based weighted averaging applied on a blue smooth area.

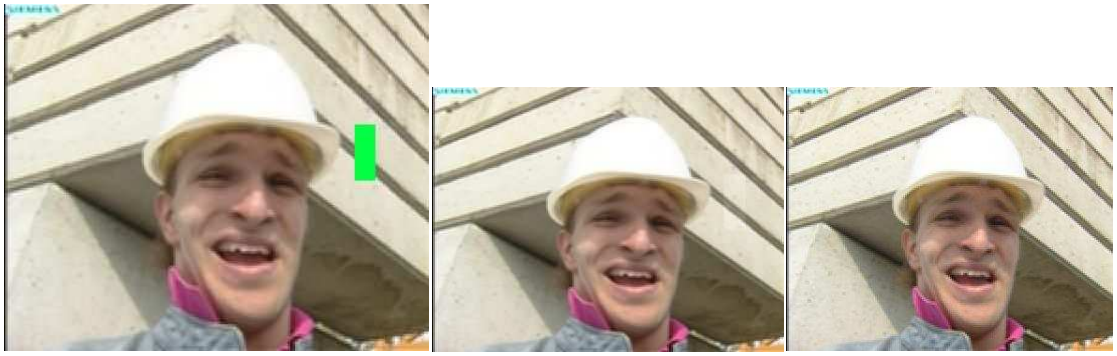
a b c

Figure 6.4: Recovery of the damaged macroblock in Akiyo video sequence (a) distorted image lying between black and blue smooth area b) macroblock based weighted averaging applied on

a missing macroblock lying between black and blue smooth areas; c) block based weighted averaging applied on a missing block lying between black and blue smooth areas.

a b c

Figure 6.5: Recovery of the damaged macroblock in Foreman video sequence (a) distorted image lying within a smooth area; b) macroblock based weighted averaging applied on a white smooth area; c) block based weighted averaging applied on a white smooth area.



a b c

Figure 6.6: Recovery of the damaged macroblock in Foreman video sequence (a) distorted image lying between white and black smooth area b) macroblock based weighted averaging applied on a missing macroblock lying between black and white smooth areas; c) block based weighted averaging applied on a missing block lying between black and white smooth areas.

6.3 Error concealment in temporal domain

Among the error concealment methods in the spatial domain the weighted averaging methods exploit the spatial smoothness property of natural video signal. In addition to the spatial correlation within each video frame in the spatial domain a video signal has also a

significant nature which is represented by the existence of correlation between the adjacent video frames in the time domain. This redundancy can also be exploited in the error concealment. In this section two different error concealment techniques based on the temporal domain are presented.

- Movement characteristics

  It is easier to conceal linear movements in one direction because pictures can be predicted from previous frames (the scene is almost the same). If there is movements in many directions or scene cuts, finding a part of previous frame that is similar is going to be more difficult, or even impossible.

- Speed characteristic

  Slower camera movement makes it easier to conceal an error.

This kind of error concealment seizes on temporal correlation of the video sequence to conceal the error. Motion estimation using previous frames is performed to reconstruct the missing data.

6.3.1 Copy-Paste algorithm

Copy-Paste is the simplest temporal error concealment method. Here the missing blocks of one frame, F_n , are replaced by the spatially corresponding blocks of the previous frame, F_{n-1}.

$$F_n(i,j) = F_{n-1}(i,j) \qquad (6.6)$$

This method only performs well for a low motion sequence, but the advantages lie in its low complexity (see figure 6.6). Better performance is provided by the motion compensated interpolation methods (see figure 6.7).

Figure 6.7: Frames# 5, 6 and 7 are the output of H.264 encoded frames after it is transmitted in the error prone wireless medium.

Figure 6.8: Frame# 5 is the decoded frame. Here Frame# 6 successfully copied lost information from Frame 5 by copy algorithm; Frame #7 is degraded (Because Frame#7 is reconstructed by collecting the information from previous reference frames).

6.3.2 Recovery of inter prediction side information

The H.264/AVC decoder needs the inter prediction side information and the DCT coefficients of the residuals. The Inter prediction side information includes the motion vectors and the corresponding reference frame number. The loss of motion vectors degrades the decoded image. This degradation propagates to the subsequent inter frames until an intra frame is decoded. The decoding of the n-th inter frame is given by:

$$x_n\,(i,j) = x_{n-1}\left(i + V_x\,, j + V_y\right) + r_n\,(i,j)\,) \qquad (6.7)$$

Where (V_x, V_y) represent the x and y-component of the motion vector for the $(i,j)^{th}$ pixel and $r_n\ (i,j)$ denotes the residual value. Note that in opposition to luma and chroma values of a pixel, a motion vector is assigned to block at least 4x4=16 pixels. Therefore all pixels belonging to the same 4x4 block have the same motion vector.

As mentioned before the H.264/AVC encoder applies the compression to the motion vectors information by taking the difference between the current motion vector and the motion vector of an already encoded neighboring macroblock. The information, to which the neighboring macroblock this differential value is related to, is added to the other inter prediction side information. Therefore, the loss of a macroblock motion vector propagates the following macroblocks in the frame or in the slice, which depend on the motion vector prediction from the affected macroblock.

By dealing with a video sequence containing the slow motion scenes, then motion vectors of the macroblock are near to zero. Considering this scenario, when a video bitstream is been received in the decoder after it is traversed in the error prone wireless medium. During the reconstruction of video frames a misinterpreted motion vector, which may have a different displacement in motion from the original position within the frame in all the following inter frames. This displacement distorts the smoothness around the affected macroblock, which degrades the perceptual video quality. The simplest way to recover the lost motion vectors of a damaged macroblock is to set its value to zero. The visual artifacts that might be produced by this method depend on the maximal detected motion. For a maximal value of 1 pixel per frame those artifacts can be held in small ranges and cannot be recognized.

In some video scenes a homogenous movement of all objects within the video frame can be recognized, such a scene is created by a moving camera shot. The difference between the motion vectors of adjacent macroblocks is near zero. This difference value is extracted by the video compressor; a misinterpretation of this value on the receiver side means automatically a misinterpretation of the actual motion vector and leads to a global displacement of a group of

macroblocks within the actual frame. Similar behavior can be recognized in case of homogenous movement of a group of macroblocks within a moving object in the video scene; all macroblocks belonging to this object have the same motion vector, a transmission failure of the motion vector belonging to the first decoded macroblock of this group could cause a local displacement of the object. In these two cases the simplest way to recover motion vectors is to use the motion vector, to which the corrupted motion vector is related to. This can be implemented by setting the differential motion vector value which has been affected by the bit stream error to zero. With that the resulting motion vector value is the same as the reference motion vector value. This method can also be applied to the video frames of low motion video scenes.

6.3.3 Motion Estimation: Motion vectors interpolation

The efficiency of the two methods presented in the previous section is still limited by special types of the video scene. Generally a video sequence is a mixture of slow motion and fast motion. Also a video scene could include objects with different dynamic behaviors. For this reason there is a need of motion estimation methods which use the smoothness in the space and time domains. A motion vector of a 4x4 block can be estimated by interpolating this value from the motion vectors in the surrounding macroblocks. Distance between the 4x4-block and the surrounding 4x4-block can be used as a weighting factor (See Figure 6.8)

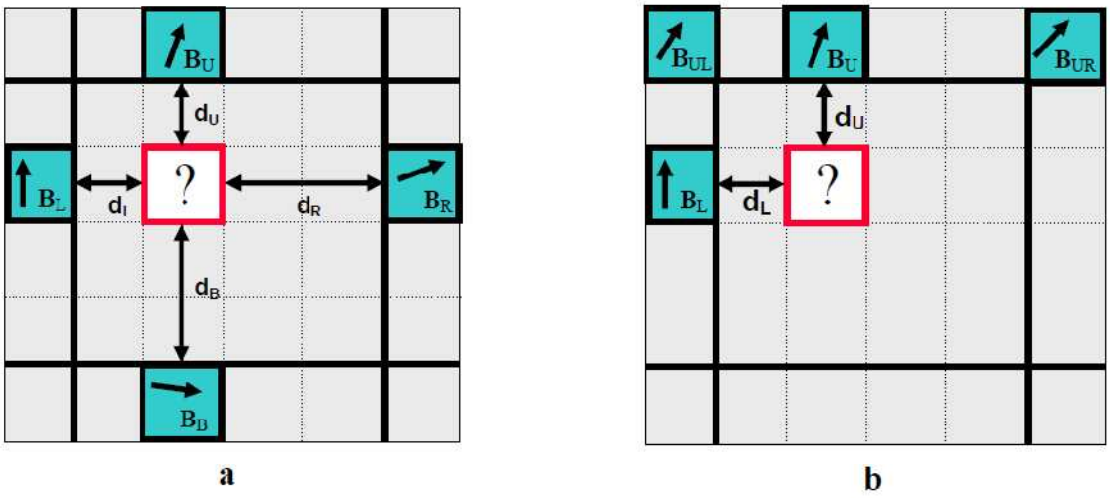


Figure 6.9: Motion vector recovery by **a)** Using the motion vectors from the surrounding macroblocks after frame decoding **b)** Using the motion vectors from the surrounding macroblocks during macroblock decoding [28].

By using a macroblock of size 4Nx4N pixels, the macroblock includes NxN 4x4-blocks, the x- and y-components of the corrupted motion vector are estimated by:

$$V_x(mb(i,k)) = \frac{1}{d_L + d_R + d_T + d_B}[d_R V_x(mb_L(N, j)) + d_L V_x(mb_R(1, j)) + d_B V_x(mb(i, N)) + d_T V_x(mb_B(i, 1)) \quad (6.8)$$

$$V_y(mb(i,k)) = \frac{1}{d_L + d_R + d_T + d_B}[d_R V_y(mb_L(N, j)) + d_L V_y(mb_R(1, j)) + d_B V_y(mb_U(i, N)) + d_T V_y(mb_B(i, 1)) \quad (6.9)$$

Where $i, k = 1, 2, 3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots N$.

d_L : distance between the interpolated pixel and the nearest pixel $mb_L = mb(i, 0)$ in left boundary.

d_R : distance between the interpolated pixel and the nearest pixel $mb_R = mb(i,\ N + 1)$ in right boundary.

d_T : distance between the interpolated pixel and the nearest pixel $mb_T = mb(0, j)$ in top boundary.

d_B : distance between the interpolated pixel and the nearest pixel $mb_B = mb(N + 1, j)$ in bottom boundary.

N: Size of the block

V_x: motion vector in x-direction.

V_y: motion vector in y-direction.

mb: macroblock

The use of motion vectors from the macroblocks on the right and the bottom of the affected macroblocks is only possible if the corresponding macroblocks are already decoded and if the differential motion vector of these macroblocks is not related to the motion vector of the affected macroblock. In many cases these two requirements cannot be fulfilled, and therefore, an estimation process has to be performed during the decoding of the affected macroblocks using motion vectors of the previously decoded macroblocks (left and upper macroblocks in figure 5.3 b). The x- and y-components of the corrupted motion vector are estimated by:

$$V_x(mb(i\ , j)) = \frac{1}{d_L + d_T}[\ d_L V_x(mb_T(i\ , N)) + \ d_T V_x(mb_L(N\ , j)) \tag{6.10}$$

$$V_y(mb(i\ , j)) = \frac{1}{d_L + d_T}[\ d_L V_y(mb_T(i\ , N)) + \ d_T V_y(mb_L(N\ , j)) \tag{6.11}$$

By using the motion vector in the upper right and upper left corner. The x- and y-components of the corrupted motion vector are estimated by:

$$V_x(mb(i\ ,j)) = \frac{1}{d_L + d_T}[\ d_L V_{T,\ x}(mb(i\ ,j)) + \ d_T V_x(mb_L(N\ ,j)) \tag{6.12}$$

$$V_{T,x}(mb(i,j) = \frac{1}{2N}[(N-i)V_x\big(mb_{TL}(N,N)\big) + N\ V_x\big(mb_T(i,N)\big) + iV_x\big(mb(1,N)\big)] \tag{6.13}$$

$$V_y(mb(i\ ,j)) = \frac{1}{d_L + d_T}[\ d_L V_{T,\ y}(mb(i\ ,j)) + \ d_T V_y(mb(N\ ,j)) \tag{6.14}$$

$$V_{T,y}\ (mb(i,j) = \frac{1}{2N}[(N-i)V_y\big(mb_{TL}(N,N)\big) + \ N\ V_y\big(mb_T(i,N)\big) + \ iV_y\big(mb_{TR}\ (1,N)\big)] \tag{6.15}$$

where $i, k = 1, 2, 3, \ldots\ldots\ldots\ldots\ldots\ldots\ldots N$.

d_L : distance between the interpolated pixel and the nearest pixel $mb_L = mb(i, 0)$ in left boundary.

d_R : distance between the interpolated pixel and the nearest pixel $mb_R = mb(i,\ N+1)$ in right boundary.

d_T : distance between the interpolated pixel and the nearest pixel $mb_T = mb(0, j)$ in top boundary.

d_B : distance between the interpolated pixel and the nearest pixel $mb_B = mb(N+1, j)$ in bottom boundary.

N: Size of the block

V_x: motion vector component in x-direction.

V_y: motion vector component in y-direction.

mb: macroblock

As mentioned before a misinterpreted motion vector leads to a displacement of the macroblock. The smoothness at the boundary of the affected macroblock is not fulfilled any more. The smoothness can be recovered by searching the best macroblock position within the reference frame, where the difference of the pixels values between the outer boundary of the affected macroblock and the boundary of the concealed macroblock is minimal. The difference between the concealed position and the raster position of the affected macroblock is the missing motion vector; this method is called boundary matching.

Another significant inter prediction side information is the reference frame number. The loss of this information can degrade the quality of decoded image by replacing the original block by a different one which does not fit in the actual frame. If the motion vector has been received correctly, the lost reference frame number can be recovered by scanning all available reference frame at the position indicated by the motion vector till it finds the block which match in the actual frame. This method can be complicated if a big reference frame buffer is used. The recovery process can be made easier by using the most used reference frame number in the neighboring blocks. Figure 6.9, 6.10 and 6.11 illustrate the encoding; decoding and how lost information is recovered with an I-frame using a motion estimation algorithm. Figure 6.12 illustrates the size of each frames that is being encoded.

Figure 6.10: Frame#1 to frame#20 of original encoded output from H.264 encoder.

Figure 6.11: Frame#1 to frame#20 of distorted video sequence due to the packet loss during transmission of bit stream in an error prone wireless medium.

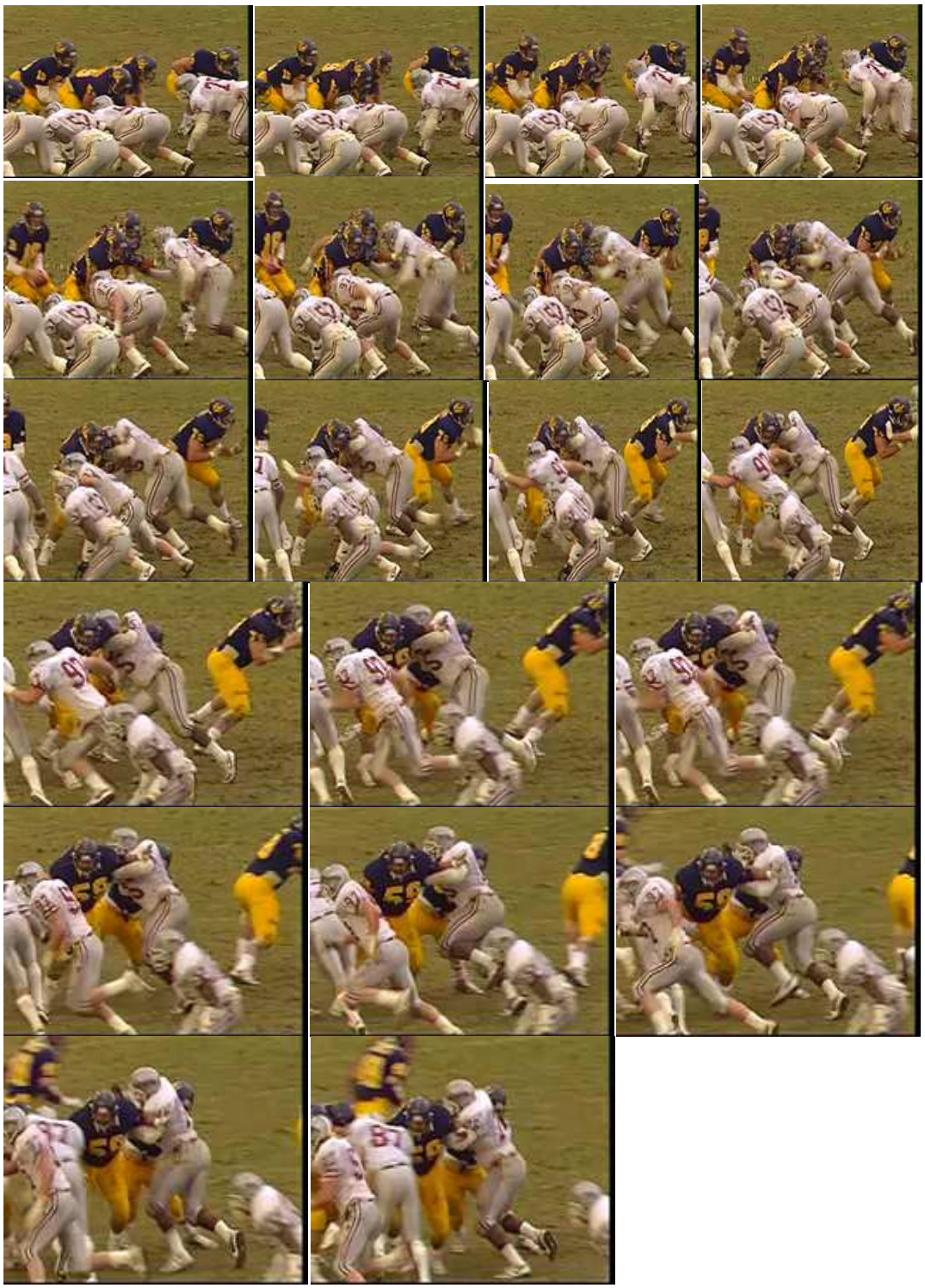


Figure 6.12: Frame#1 to frame#20 of motion estimation algorithm (motion vector interpolation) output.

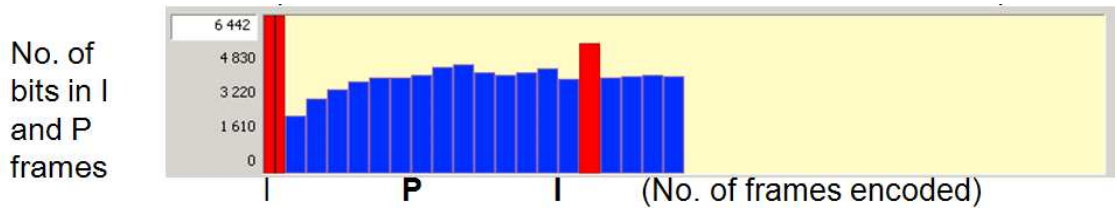


Figure 6.13: Graph shows the size (number of bits) of the different I and P frames obtained after encoding 20 frames of the Football QCIF video sequence. Green line shows the average values of the bits lost when it is passed through the lossy wireless medium.

Table 6.1 represents the number of frames encoded, with type of frame, offset value which is the distance between successive frames and time required for coding the bitstream and number of bits the particular frame represents. Figures 6.13 and 6.14 show the SSIM image output and graph of original and concealed video sequences.

Table 6.1: Representation of coded video sequence.

num	type	offset	Time (Sec)	Size(bits)
0	I	0x0000000000000000	00:00.0	6442
1	P	0x000000000000192a	00:00.0	1244
2	P	0x0000000000001e06	00:00.1	3074
3	P	0x0000000000002a08	00:00.1	3460
4	P	0x000000000000378c	00:00.1	3765
5	P	0x0000000000004641	00:00.2	3931
6	P	0x000000000000559c	00:00.2	3964
7	P	0x0000000000006518	00:00.2	4078
8	P	0x0000000000007506	00:00.3	4393
9	P	0x000000000000862f	00:00.3	4482
10	P	0x00000000000097b1	00:00.3	4144
11	P	0x000000000000a7e1	00:00.4	4067
12	P	0x000000000000b7c4	00:00.4	4145
13	P	0x000000000000c7f5	00:00.4	4306
14	P	0x000000000000d8c7	00:00.5	3902
15	I	0x000000000000e805	00:00.5	5357
16	P	0x000000000000fcf2	00:00.5	3940
17	P	0x0000000000010c56	00:00.6	4019
18	P	0x0000000000011c09	00:00.6	4060
19	P	0x0000000000012be5	00:00.6	4015

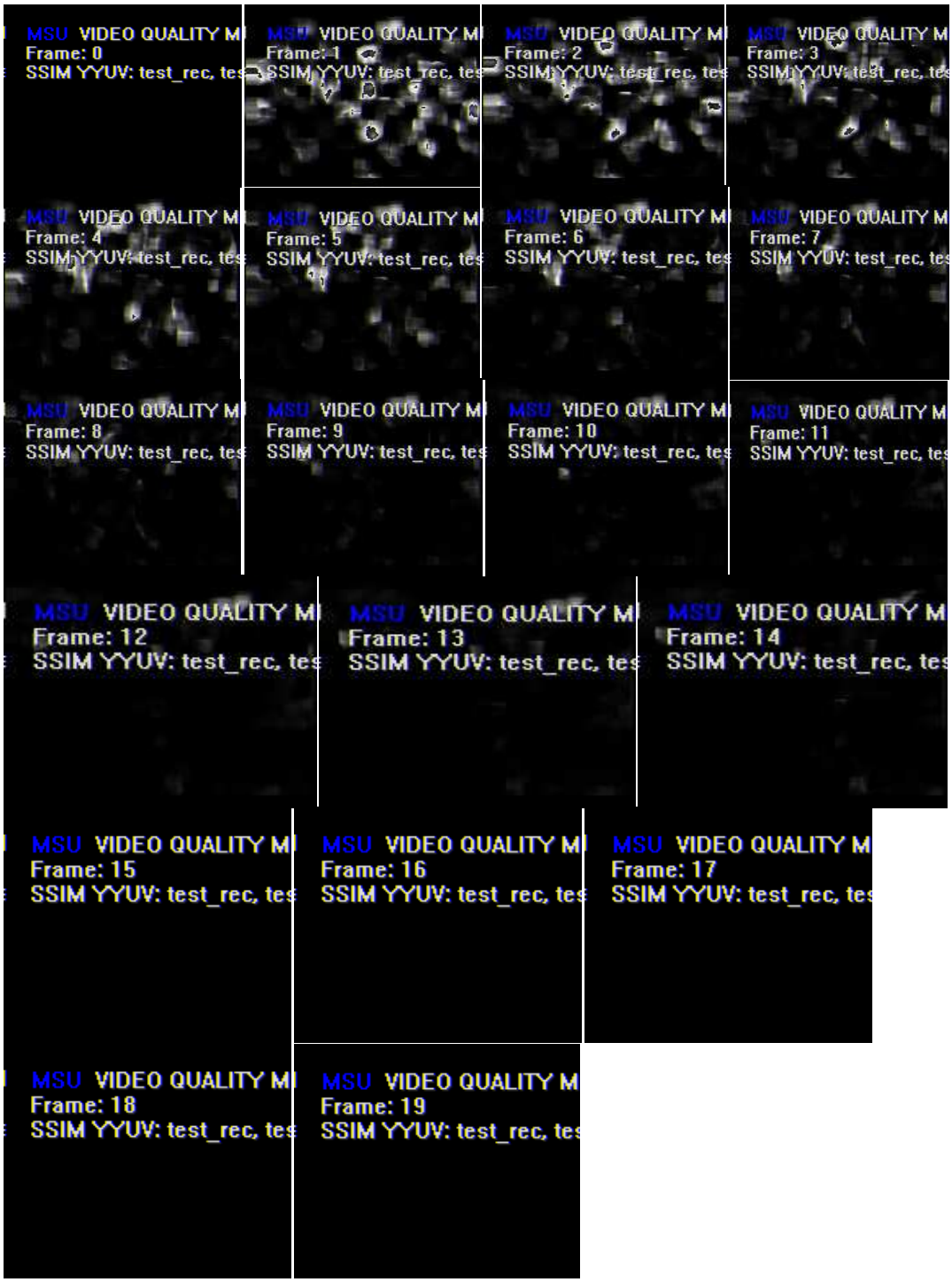


Figure 6.14: Representation of images from the SSIM metric where it gives the visual differentiation between original and concealed video sequence (Completely black image in this figure represents that both the images are having same pixel representation).

Table 6.2 shows the value of SSIM output between original and concealed video sequences.

Table 6.2: Representation of SSIM output (1->two images are alike, 0->two images have completely different pixel values).

num	type	SSIM
0	I	1
1	P	0.87019
2	P	0.89112
3	P	0.91215
4	P	0.92346
5	P	0.93983
6	P	0.95667
7	P	0.96764
8	P	0.97571
9	P	0.9813
10	P	0.98323
11	P	0.98439
12	P	0.98775
13	P	0.98858
14	P	0.99051
15	I	1
16	P	1
17	P	1
18	P	1
19	P	1

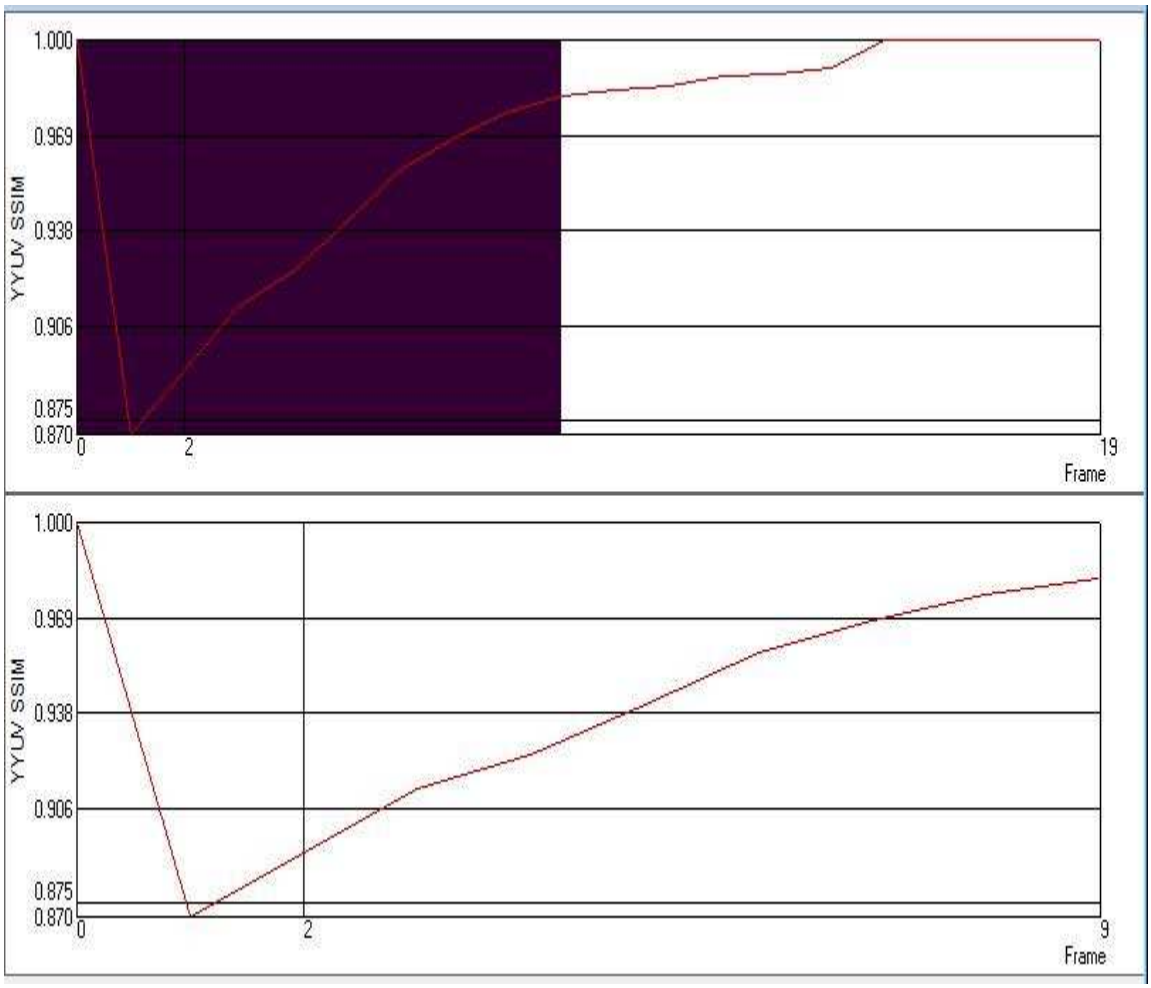


Figure 6.15: Comparison of the recovered frame with original sequence by motion estimation using SSIM index.

Table 6.3 shows PSNR values between encoded and concealed video sequences for Y, U and V separately.

Table 6.3: Performance comparison between concealed and original video sequences using PSNR representation.

Frame	SNR_Y (dB)	SNR_U (dB)	SNR_V (dB)
1	100	100	100
2	24.6474	35.3199	41.4868
3	25.6231	36.6582	42.7562
4	26.7219	38.7013	44.6911
5	27.6355	38.9417	45.2911
6	28.9347	40.5029	46.9383
7	30.9396	42.9496	48.9062
8	32.9711	46.4796	51.8757
9	34.3972	48.2814	53.3843
10	35.9502	49.8022	55.7311
11	37.2341	51.7953	57.9142
12	37.4774	52.0513	57.8239
13	39.2709	54.0244	59.3005
14	39.7164	54.6568	59.4931
15	40.1675	55.6336	60.0851
16	100	100	100
17	100	100	100
18	100	100	100
19	100	100	100

Figure 6.15 and 6.16 illustrates the concealed results for motion estimation algorithm. Figures 6.17 and 6.18 illustrate the SSIM graph. Figure 6.19 and 6.20 show PSNR graph for concealed video sequence for Foreman video sequence.

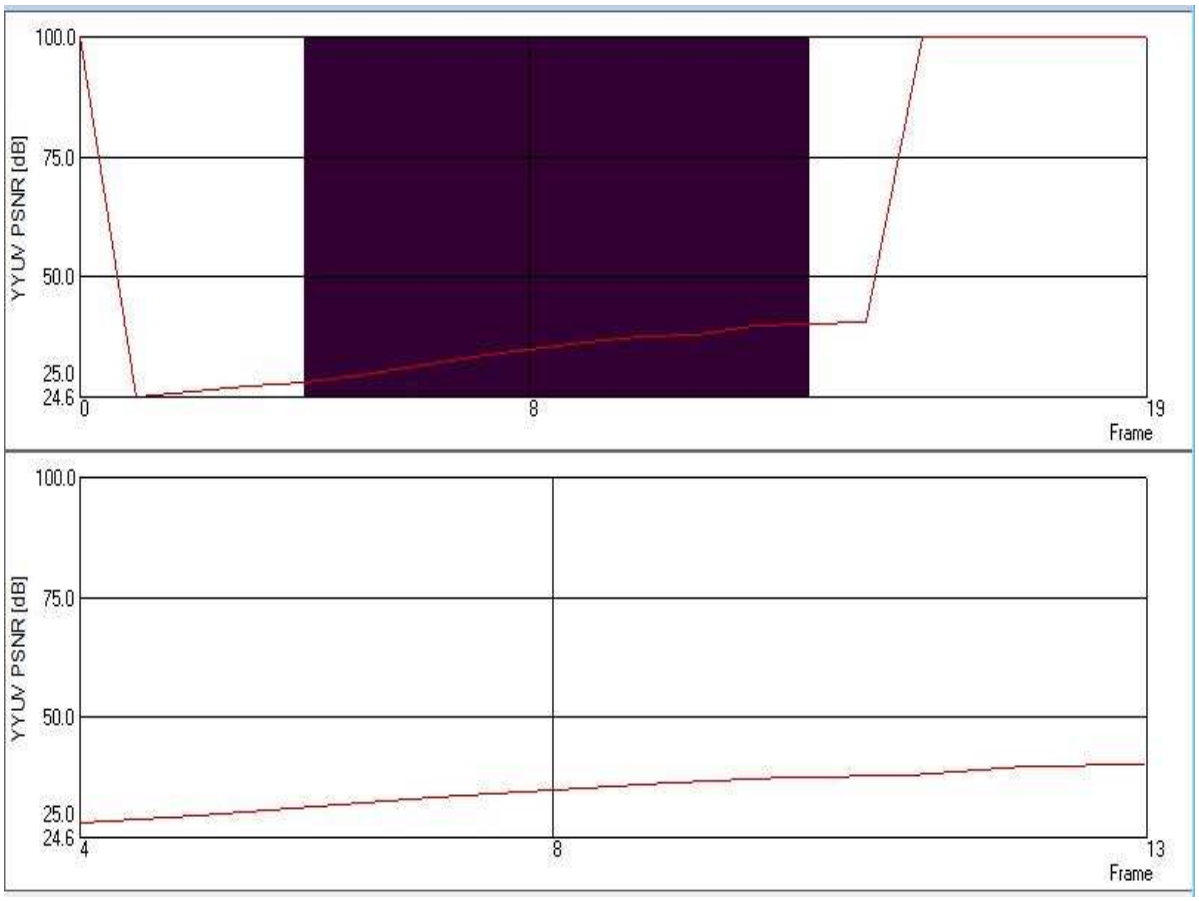


Figure 6.16: Comparison between original and recovered frames by motion estimation using PSNR metric.

a b c

Figure 6.17: Recovery of the damaged macroblock in Foreman video sequence (a) original sequence b) Distorted Sequence c) Concealed output using motion estimation.

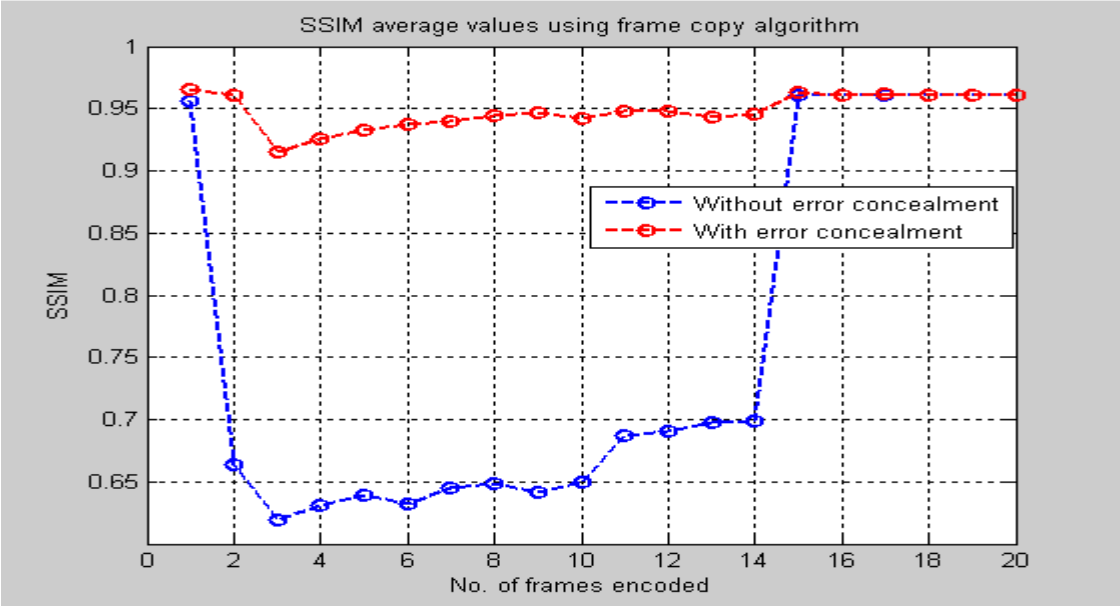


Figure 6.18: SSIM average values using frame copy algorithm (Foreman video sequence).

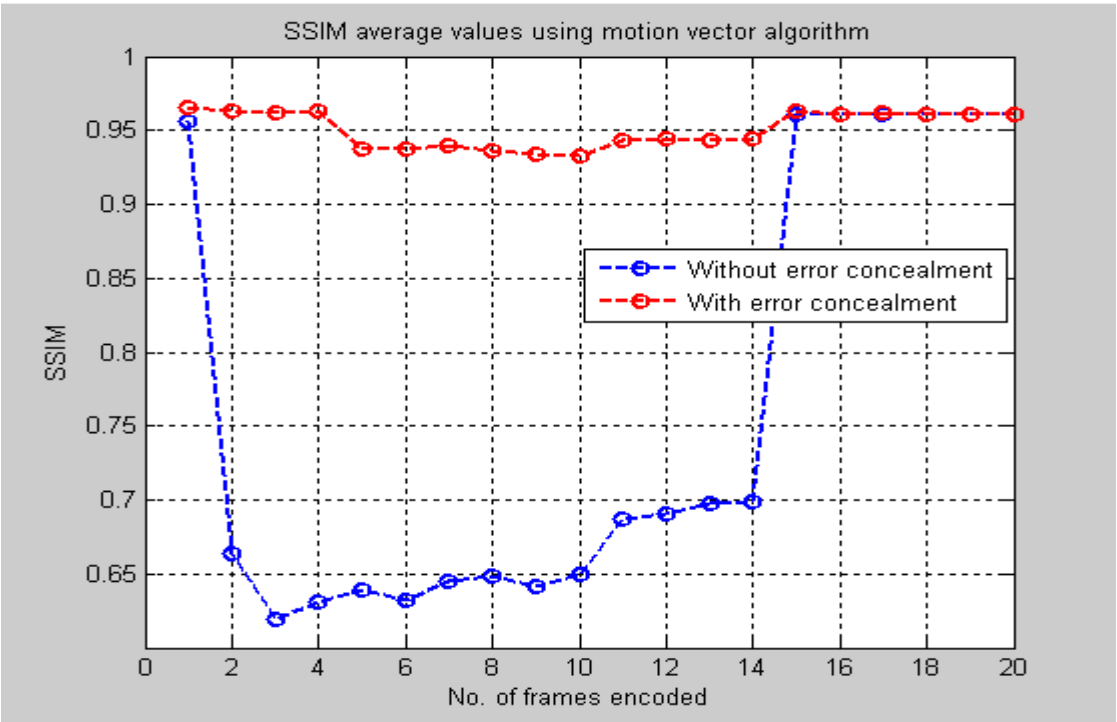


Figure 6.19: SSIM average values using motion estimation algorithm (Foreman video sequence).

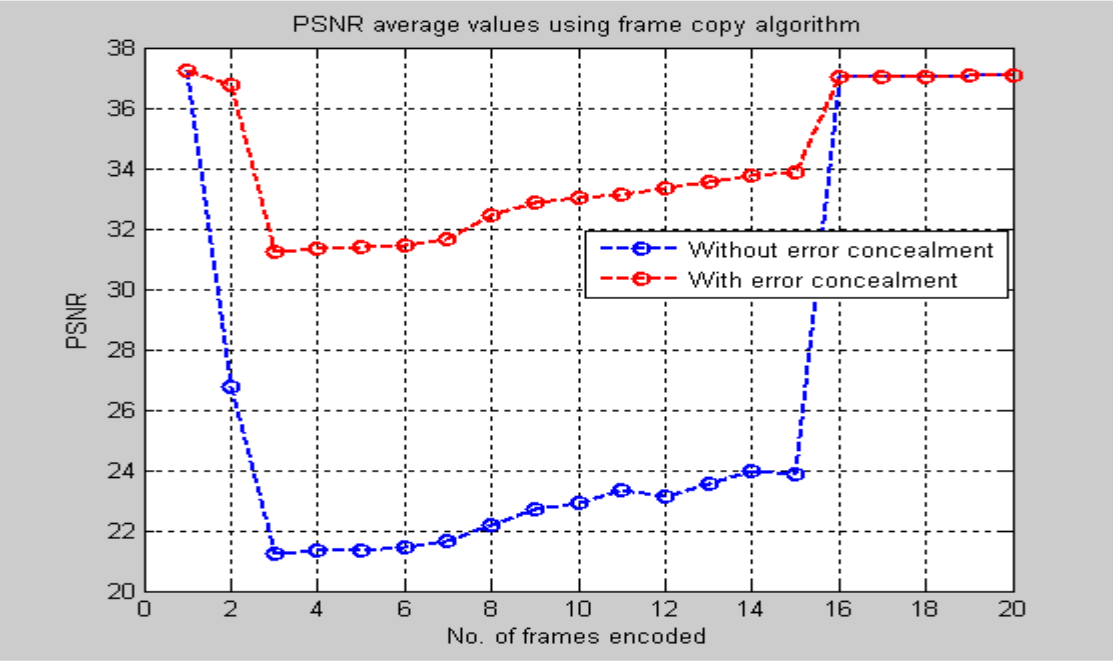


Figure 6.20: PSNR average values using frame copy algorithm (Foreman video sequence.

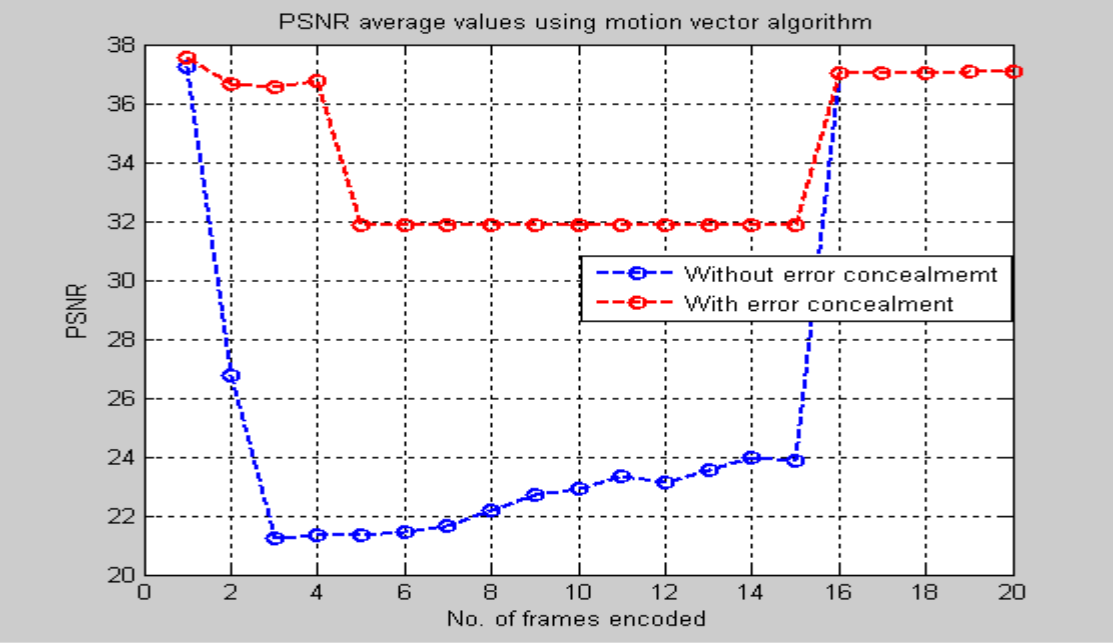


Figure 6.21: PSNR average values using motion estimation algorithm (Foreman video sequence).

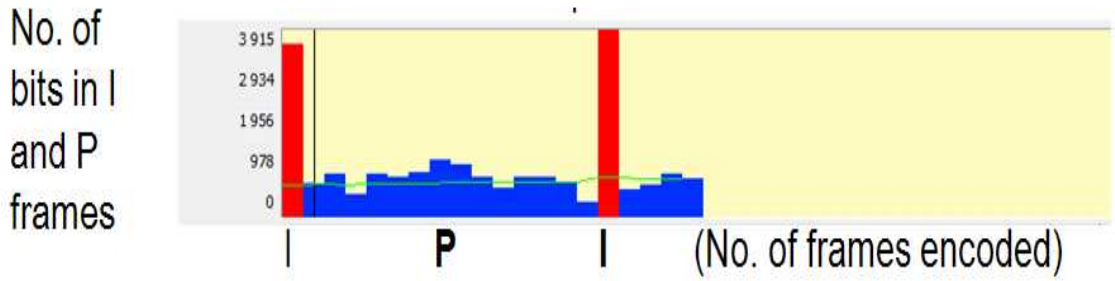


Figure 6.22: Size of I (red color bar) and P (blue color bar) frames obtained after encoding 19 frames of the foreman QCIF (176 x 144) video sequence. Green line shows the average values of the bit lost when it is passed through the lossy wireless medium (Foreman Video Sequence)

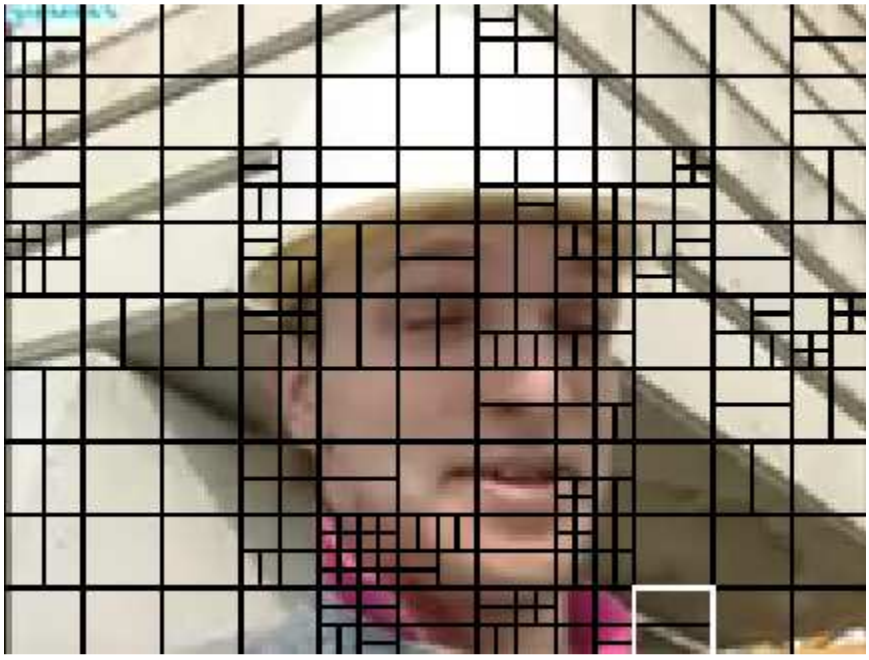


Figure 6.23: Representation of different block sizes used for decoding in the motion estimation algorithm

Algorithmic analysis based on GOP (Group of Pictures), QP (Quantization Parameters) for QCIF and CIF video sequences is shown in Tables 6.4 and 6.5.

Table 6.4: Simulation results of different error concealment algorithms for Foreman QCIF 20 frame video sequence, (frame rate =30fps).

Video Sequence	QP	Bitrate (kbps)	Original PSNR (dB) (No Errors)	Error Concealment method	PSNR of error concealment methods (dB)
Foreman (GOP=15, No. of Frames Encoded=20)	28	271.52	36.84	Weighted Averaging	33.02
				Copy Paste	32.43
				Motion Vector Interpolation	33.46
	24	430.36	39.65	Weighted Averaging	34.60
				Copy Paste	33.21
				Motion Vector Interpolation	35.42
	20	692.38	42.88	Weighted Averaging	36.43
				Copy Paste	35.80
				Motion Vector Interpolation	37.20
Foreman (GOP=20, No. of Frames Encoded=20)	28	231.73	36.79	Weighted Averaging	33.98
				Copy Paste	32.23
				Motion Vector Interpolation	34.09
	24	379.34	39.62	Weighted Averaging	33.01
				Copy Paste	32.92
				Motion Vector Interpolation	34.76
	20	625.21	42.87	Weighted Averaging	34.21
				Copy Paste	33.57
				MV Interpolation	35.05

Table 6.5: Simulation results of different error concealment algorithms for Stefan CIF 20 frame video sequence (frame rate = 30fps).

Video Sequence	QP	Bitrate (kbps)	Original PSNR (dB) (No Errors)	Error Concealment methods	PSNR of different error concealment methods (dB)
Stefan (GOP=15, No. of Frames Encoded=20)	28	1903.90	36.88	Weighted Averaging	33.46
				Copy Paste	30.49
				Motion Vector Interpolation	33.02
	24	3074.17	40.13	Weighted Averaging	34.01
				Copy Paste	31.25
				Motion Vector Interpolation	33.46
	20	4777.70	43.51	Weighted Averaging	35.61
				Copy Paste	32.59
				Motion Vector Interpolation	36.02
Stefan (GOP=20, No. of Frames Encoded=20)	28	1725.25	36.86	Weighted Averaging	33.03
				Copy Paste	30.08
				Motion Vector Interpolation	32.46
	24	2868.44	40.10	Weighted Averaging	33.52
				Copy Paste	31.10
				Motion Vector Interpolation	33.21
	20	4563.82	43.46	Weighted Averaging	33.99
				Copy Paste	32.01
				Motion Vector Interpolation	33.24

CHAPTER 7

COMPUTATIONAL COMPLEXITY

The computational complexity is crucial especially for the wireless video due to the size and power limited terminals. At present there is no standard criteria used to compare the complexity of error concealment methods. The obtained result is based on the amount of data access for each method.

7.1 Decoding time

One way of calculating complexity is analyzing decoding time. JM 13.2 Reference Software [27] outputs time needed for decoding every frame of the video sequence and the time for entire sequence. With this measurement it is difficult to realize the grade of complexity of the different concealment methods. It would be a better option to know decoding time per macroblock (MB) concealed. The problem is that this time is very short to be calculated in C code with functions like time or ftime (file time) in Windows.

7.2 Number of operations

By the error concealment algorithm analyzed in section 6.1, here spatial domain method has low complexity due to the fact that it is implemented by gathering the information from the neighboring macroblock of a current frame there by reducing the computational complexity in which it can be implemented in a small handheld devices there by limiting battery power for processing in mobile terminals [36, 37]. Here complexity is measured by counting the number of operations.

7.2.1 Weighted averaging

Weighted averaging is given by:

$$mb(i,k) = \frac{1}{d_L + d_R + d_T + d_B}[d_R mb_L(i,2N) + d_L mb_R(i,1) + d_B mb_T(2N,k) + d_T mb_B(1,k) \quad (7.1)$$

where $i,k = 1,2,3,\ldots\ldots\ldots\ldots\ldots\ldots\ldots N.$

d_L : distance between the interpolated pixel and the nearest pixel $mb_L = mb(i,0)$ in left boundary.

d_R : distance between the interpolated pixel and the nearest pixel $mb_R = mb(i,\ N+1)$ in right boundary.

d_T : distance between the interpolated pixel and the nearest pixel $mb_T = mb(0,j)$ in top boundary.

d_B : distance between the interpolated pixel and the nearest pixel $mb_B = mb(N+1,j)$ in bottom boundary.

N: Size of the block

V_x: motion vector in x-direction.

V_y: motion vector in y-direction.

mb: macroblock

By looking at the equation (7.1) it is very simple to find out the number of operations:

Number of additions = $(N-1).(Q^2+1)$ (7.2)

Number of multiplications = $(N+1).Q^2$ (7.3)

where:

Q: block size.

N: Whole frame size NxN.

7.2.2 Inverse Transform

To be able to compare the number of operations of the different error concealment methods there must be a reference. By choosing reference as the number of operations of the inverse transforms for the luma component, neglecting other operations as i.e. dequantization or run-length calculation.

All major prior video coding standards [26] used a transform block size of 8×8, while the new H.264/AVC design is based primarily on a 4×4 transform block size. This allows the encoder to represent signals in a more locally adaptive fashion, which reduces artifacts known colloquially as “ringing”. All the examples of the way the transform is calculated are given for the case of a 4×4 transform. However, here the number of operations considered is both 4×4 and 8×8 block size.

The integer transform is based in the discrete cosine transform (DCT). It works in this way: maps a length-N vector x into a new vector X of transform coefficients by a linear transformation X = H x. The DCT is not used because the matrix H that defines the transformation has irrational numbers. Thus, by computing the forward and inverse transforms in cascade, the resultant may not be exactly the same as the original data. It is desirable to replace H by an orthogonal matrix with integer entries [29]. It is represented as:

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} \tag{7.4}$$

The rows of H in equation 7.4 are orthogonal, but they do not have the same norm (Sum of absolute values in any row of H). However, that can be easily compensated by the quantization process [29]. The decoder uses just the transpose of H with appropriate scaling and reconstructed transform co-efficients.

The inverse transform matrix is defined by

$$\grave{\mathbf{H}}_{inv} = \begin{pmatrix} 1 & 1 & 1 & 1/2 \\ 1 & 1/2 & -1 & -1 \\ 1 & -1/2 & -1 & 1 \\ 1 & -1 & 1 & -1/2 \end{pmatrix} \tag{7.5}$$

where `Hinv in equation 7.5, is a scaled inverse of H. The multiplications by 1/2 can be implemented by 1-bit right shifts, so that all decoders produce identical results. To count the number of operations there must be a reference on how it is mathematically programmed in H.264/AVC [30]. The transform process shall convert the block of scaled transform coefficients w

$$\mathbf{w} = \begin{pmatrix} \mathbf{w}_{00} & \mathbf{w}_{01} & \mathbf{w}_{02} & \mathbf{w}_{03} \\ \mathbf{w}_{10} & \mathbf{w}_{11} & \mathbf{w}_{12} & \mathbf{w}_{13} \\ \mathbf{w}_{20} & \mathbf{w}_{21} & \mathbf{w}_{22} & \mathbf{w}_{23} \\ \mathbf{w}_{30} & \mathbf{w}_{31} & \mathbf{w}_{32} & \mathbf{w}_{33} \end{pmatrix} \tag{7.6}$$

to a block of output samples in a manner mathematically equivalent to the following process:

1. First, each (vertical) column of scaled transform coefficients is transformed using a one-dimensional inverse transform, and

2. Then, each (horizontal) row of the resulting matrix is transformed using the same one-dimensional inverse transform.

The one-dimensional inverse transform is specified as follows for four input samples w_0 ,w_1,w_2, w_3, where the subscript indicates the one-dimensional frequency index.

1. A set of intermediate values is computed:

$$Z_0 = w_0 + w_2 \tag{7.7}$$

$$Z_1 = w_0 - w_2 \tag{7.8}$$

$$Z_2 = (w_1 \gg 1) - w_3 \tag{7.9}$$

$$Z_3 = w_1 + (w_3 \gg 1) \tag{7.10}$$

2. The transformed result is computed from these intermediate values:

$$x_0 = z_0 + z_3 \tag{7.11}$$

$$x_1 = z_1 + z_2 \tag{7.12}$$

$$x_2 = z_1 - z_2 \tag{7.13}$$

$$x_3 = z_0 - z_3 \tag{7.14}$$

Figure 7.1 shows a flow graph of the inverse transform, which is applied to rows and columns in the case of 4×4 transform block size.

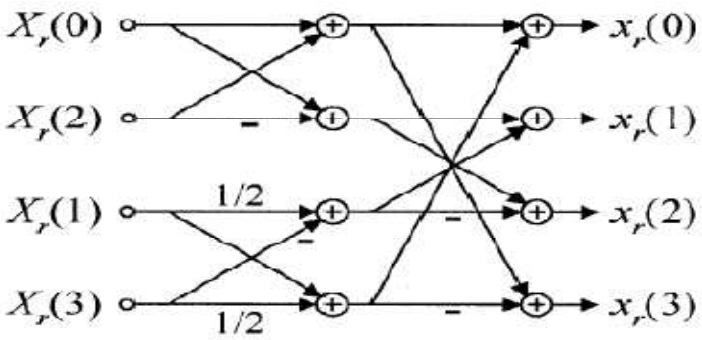


Figure 7.1 Fast implementation of the H.264/AVC inverse transform. No multiplications are needed, only additions and shifts.

The number of operations of the inverse transform for 4×4 block size (B) is:

$$\text{Number of additions} = 32 \cdot 16 = 512,$$

$$\text{Number of shifts} = 4 \cdot 16 = 64,$$

The transform used in H.264/AVC is always 4×4. By assuming that the 8×8 transform is calculated as 4 transforms of a 4×4 block and the 16×16 transform as 16 transforms of a 4×4 block, the number of operations can be calculated.

7.3 Decoding time

Here in this section mainly concentrating on time that an error concealment method spends concealing. The value is calculated by considering 2 cases one decoding without error concealment algorithm and other with decoding with error concealment algorithm. Below are the test results:

Table 7.1: Decoding time values (ms) under windows vista platform

Sequence	Decode without error concealment algorithm	Decode with error concealment algorithm
Foreman	5.023	5.145
Akiyo	4.002	4.171
Football	5.372	5.620
Videoclip	10.261	10.533

CHAPTER 8

H.264/AVC VIDEO CODEC IMPLEMENTATION

8.1 Assumptions

To evaluate the performance of the different error concealment methods, representative video sequences under different error rates are selected. These images are called Akiyo, Foreman and Football.

To perform the simulations in real-time, the video content must be sent over the UMTS (Universal Mobile Telecommunications System) network to be reproduced in a mobile telephone display. For this there is a limitation of the display screen size; the usual format used for the mobile terminals is QCIF (Quarter Common Intermediate Format) resolution (176x144 pixels). The H.264/AVC standard is well known for its very good compression rate thereby reducing the number of bits needed for videos. 30fps used for lab testing purpose. For streaming in wireless networks like UMTS, the frame rate is reduced to 7.5fps. For implementation of frames, only I-frame (intra) and P-frame (predicted) are used with following format IPPPP...PIPP... structure. This is to reflect the baseline profile of the H.264/AVC standard [7], which does not necessarily support B-frame (Bi-directinal).

8.2 Changes to the Joint Model Source Code

For implementation of the proposed algorithm, the Joint Model H.264/AVC version 13.2 is used. [27]. This software is free to the user without any license fee or royalty. Generated by the JVT this software consists of both video encoder and decoder. All the source code is included in the package. This source code is written in C programming language, and Microsoft Visual Studio is used as the tool for working with it. In the implementation encoder code is not modified but some changes in the configuration files is introduced since it is required in the decoder for efficient reconstruction of lost frames during the transmission of bitstreams in the wireless medium. Appendix A, describes the parameters of the configuration file of the encoder (”encoder.cfg”) which is changed in order for fruitful recovery of a degraded video sequence.

The decoder has another configuration file (”decoder.cfg”), but it is less complex than the encoder. In this file there is a need to indicate which video stream is used for decoding and error concealment. This is explained in detail in Appendix A. Both configuration files, encoder and decoder, are shown in the Appendicies B and C.

8.3 Generation of Errors in the coded Bit stream

The main aim is to identify the performance of the different error concealment methods. First the algorithm is written and implemented in the decoder and then carried out in different real time applications. There must be a test bench to execute this algorithm in a practical implementation that includes random addition of errors to the encoded bitstream. An error concealment algorithm is applied to an artificially developed wireless medium which will introduce errors in an abrupt manner. A new source code is written which defines the different characteristics of a wireless medium. Here the actual process of introducing the errors is a slight tricky method. Once a video

sequence is encoded, a continuous stream of bits is developed. Then the bits are multiplexed with the overhead information such as motion vectors, control bits and NAL bits and are transmitted.

In real-time applications the multiplexed bits are transmitted in the wireless medium and are received in the decoder. The same scenario was created in the form of a block where the multiplexed bit stream is passed to an artificial wireless medium. Errors are randomly introduced into a single frame in this medium. Then this error propagates until the end of GOP (Group of Pictures) i.e., before the start of another I frame.

In the decoder the degraded video is received and the error concealment algorithm is applied to conceal the lost frames. Finally the PSNR of the decoded video sequence is calculated and is compared with that of the original sequence.

8.4 Simulation steps, commands and output results

The following steps give a brief description of how encoding, error introduction and decoding are to be implemented using the JM 13.2 standard [27].

Steps for compilation of the codec:

- Open the project in Microsoft Visual Studio compiler.
- Build encoder (lencod) file and compile it for errors.
- Open Command Prompt and point to the path where the encoder configuration file is present.
- Type in command: - "**lencod.exe -f encoder.cfg"** and it will get the output stream file in the form of a **.264** file.
- The **.264** file is copied to a routine where a lossy error block is created. The file is then built and compiled. The output of this block will be a combination of error and original bitstream information.

- Copy the output of lossy block into the decoder block and modify the configuration file to select which error concealment technique is to be used, what name the output and input file for calculating the PSNR. Here type in the command along with the input file name: -**"ldecod.exe decoder.cfg".**

The changes which were made in the encoder and decoder block with the output are shown in the Appendix A.

CHAPTER 9

CONCLUSIONS

In this thesis different error concealment methods in the spatial and the temporal domains have been implemented as functions written in the C language. These functions have been added to the decoder C source code provided in [27]. Each method is more or less efficient than the other according to the structure of the video image and the dynamic character of the video sequence.

The first implementation started with the spatial domain error concealment. The first step done in this field is based on the interpolation of the pixel values within the damaged macroblock from the pixels within the surrounding area, the distance between the concealed pixel and its neighboring pixels is used as the weighting factor. Spatial error concealment works in a video sequence where the motion between frames is negligible (Eg: Akiyo video sequence) and where the background is constant. Here one advantage of the spatial domain error concealment is the low complexity compared with enhanced error concealment in the temporal domain.

The error concealment in the temporal domain is based on the copying algorithm. In this error concealment technique the frames stored in the decoded frame buffer are used to conceal the missing part within the affected frame. This algorithm just replaces damaged macroblock by the spatially corresponding macroblock in this frame. For this purpose copy-paste function has been integrated to get the index of the frame within the decoded picture buffer, which has the maximal correlation with the affected frame. The basic copying algorithm can only be used for

error concealment in a slow motion video sequence which is characterized by high correlation in the temporal domain between the adjacent frames. The efficiency of the basic copying algorithm is limited by the dynamics of the video material. In the presence of gross motion it can produce adverse visual artifacts. For this reason motion vector interpolation is implemented to conceal the damaged image area. By using information of motion vectors for applying motion compensation to the copied macroblock, smoothness along the boundary of concealed macroblock is guaranteed.

For performing the comparison of the original and concealed video sequence PSNR, MSE and SSIM metric are used. Although PSNR has been criticized precisely for not correlating well with perceived quality measurement (i.e. [30], [31], [32], [33]). It is the one which obtains the higher correlation. The PSNR can be derived easily. Minimizing MSE is very well understood from the mathematical point of view and it can be concluded that for low resolution videos (QCIF) the most suitable metric for analyzing visual quality is PSNR.

With regard to implementation complexity, a method is proposed by analyzing the number of operations for every error concealment method. The number of operations gives useful information for deciding which error concealment method is appropriate in terms of complexity. Most of the discussed methods can also be implemented in another way, depending on the required ratio between memory and computational power.

Finally how exactly error concealment is implemented is described. Some of the commands required for running encoder and decoder configuration files, are explained.

The psuedo code structure describing the error concealment method decision tree for low resolution videos (0: Off, 1: Frame-Copy, 2: weighted-averaging, 3: motion-vector) is listed below.

if (motion vectors ==0)

method = Weigted Averaging()

else

if scene cuts and fast movements

method = Motion Vector()

else

method = copy-paste()

By applying the error concealment techniques mentioned in my thesis, there is a 10dB improvement in the PSNR over the scheme which does not have error concealment.

APPENDIX A

CONFIGURATION SETTING OF THE REFERENCE

Parameters changed in the encoder configuration file of the encoder:

While encoding a video sequence in the encoder configuration file input file is specified along with the number of frames that is to be encoded, frame rate and the video resolution of the input file. The details of the configuration input parameters is as follows:

```
###########################################################
# Files
###########################################################
InputFile            = "FOOTBALL_176x144_15_orig_01.yuv"
InputHeaderLength     = 0
StartFrame           = 0      # Start frame for encoding.
FramesToBeEncoded     = 20     # Number of frames to be coded
FrameRate            = 30.0   # Frame Rate per second (0.1-100.0)
SourceWidth          = 176    # Frame width
SourceHeight         = 144    # Frame height
TraceFile            = "trace_enc.txt"  # Trace file
ReconFile            = "test_rec.yuv"   # Recontruction YUV file
OutputFile           = "test1.264"      # Bitstream
ProfileIDC           = 66 # Profile IDC 66=Baseline
IntraProfile         = 0
IntraPeriod          = 15   # Period of I-pictures   (0=only first)
QPISlice             = 28  # Quant. param for I Slices (0-51)
QPPSlice             = 28  # Quant. param for P Slices (0-51)
NumberReferenceFrames = 1    # Number of previous frames used for inter motion
NumberBFrames         = 0  # Number of B coded frames inserted (0=not used)
SymbolMode                =   0   # Symbol mode (Entropy coding method: 0=UVLC,
1=CABAC)
```

OutFileMode = 0 # Output file mode, 0:Annex B, 1:RTP

PartitionMode = 0 # Partition Mode, 0: no DP, 1: 3 Partitions per Slice

UseWeightedReferenceME = 1 # Use weighted reference for ME (0=off, 1=on)

Parameters changed in the decoder configuration file of the decoder:

##

Files

##

E:\Thesis\software\jm13.2_\JM\bin\ggtest1.264

E:\Thesis\software\jm13.2_\JM\bin\test_dec.yuv

E:\Thesis\software\jm13.2_\JM\bin\FOOTBALL_176x144_15_orig_01.yuv

1 .....Write 4:2:0 chroma components for monochrome streams

2 .....Poc Scale (1 or 2)

leakybucketparam.cfg .....LeakyBucket Params

1.....Err-Concealment (0: Off, 1: Frame-Copy, 2: weighted-averaging, 3: motion-vector)

2 .....Reference POC gap (2: IPP (Default), 4: IbP / IpP)

2 .....POC gap (2: IPP /IbP/IpP (Default), 4: IPP with frame skip = 1 etc.)

Encoder Output:

```
------------------------------ JM 13.2 (FREXt) ------------------------------
 Input YUV file                 : FOOTBALL_176x144_15_orig_01.yuv
 Output H.264 bitstream         : test1.264
 Output YUV file                : test_rec.yuv
 YUV Format                     : YUV 4:2:0
 Frames to be encoded I-P/B     : 20/0
 Freq. for encoded bitstream    : 30
 PicInterlace / MbInterlace     : 0/0
 Transform8x8Mode               : 0
 ME Metric for Refinement Level 0  : SAD
 ME Metric for Refinement Level 1  : Hadamard SAD
 ME Metric for Refinement Level 2  : Hadamard SAD
 Mode Decision Metric           : Hadamard SAD
 Motion Estimation for components  : Y
 Image format                   : 176x144
 Error robustness               : Off
 Search range                   : 32
 Total number of references     : 1
 References for P slices        : 1
 Sequence type                  : IPPP (QP: I 28, P 28)
 Entropy coding method          : CAVLC
 Profile/Level IDC              : (66,20)
 Motion Estimation Scheme       : Fast Full Search
 Search range restrictions      : none
 RD-optimized mode decision     : used
```

Data Partitioning Mode : 1 partition

Output File Format : H.264 Bit Stream File Format

Frame	Bit/pic	QP	SnrY	SnrU	SnrV	Time(ms)	MET(ms)	Frm/Fld	Ref
0000(NVB)	160								
0000(IDR)	51376	28	35.012	38.336	39.637	307	0	FRM	1
0001(P)	19232	28	34.092	37.600	39.017	1779	1337	FRM	1
0002(P)	24592	28	34.197	37.293	38.945	1773	1307	FRM	1
0003(P)	27680	28	34.343	37.151	38.701	1818	1344	FRM	1
0004(P)	30120	28	34.504	37.131	38.867	1822	1329	FRM	1
0005(P)	31448	28	34.657	37.297	38.841	1783	1303	FRM	1
0006(P)	31712	28	34.656	37.100	38.997	1839	1350	FRM	1
0007(P)	32624	28	34.922	37.306	38.801	1821	1323	FRM	1
0008(P)	35144	28	35.296	37.503	39.156	1828	1328	FRM	1
0009(P)	35856	28	34.867	37.366	39.333	1851	1345	FRM	1
0010(P)	33152	28	34.978	37.386	39.382	1826	1332	FRM	1
0011(P)	32536	28	35.324	37.385	39.803	1808	1330	FRM	1
0012(P)	33160	28	35.336	37.444	39.523	1837	1347	FRM	1
0013(P)	34448	28	35.689	37.258	38.996	1801	1317	FRM	1
0014(P)	31216	28	35.961	37.232	39.256	1847	1339	FRM	1
0015(I)	42856	28	36.255	38.623	39.597	269	0	FRM	1
0016(P)	31520	28	35.859	37.524	39.456	1866	1352	FRM	1
0017(P)	32152	28	36.049	37.400	38.954	1834	1356	FRM	1
0018(P)	32480	28	36.247	37.450	38.993	1865	1395	FRM	1
0019(P)	32120	28	36.255	37.361	38.993	1843	1360	FRM	1

Total Frames: 20 (20)

Leaky BucketRateFile does not have valid entries.

Using rate calculated from avg. rate

Number Leaky Buckets: 8

Rmin	Bmin	Fmin
983130	51376	51376
1228890	51376	51376
1474650	51376	51376
1720410	51376	51376
1966170	51376	51376
2211930	51376	51376
2457690	51376	51376
2703450	51376	51376

---------- Average data all frames ----------

Total encoding time for the seq. : 33.417 sec (0.60 fps)

Total ME time for sequence : 24.094 sec

PSNR Y(dB) : 35.22

PSNR U(dB) : 37.46

PSNR V(dB) : 39.16

cSNR Y(dB) : 35.17 (19.77)

cSNR U(dB) : 37.44 (11.72)

cSNR V(dB) : 39.15 (7.91)

Total bits : 655584 (I 94232, P 561192, NVB 160)

Bit rate (kbit/s) @ 30.00 Hz   : 983.38

Bits to avoid Startcode Emulation : 8

Bits for parameter sets   : 160

Exit JM 13 (FRExt) encoder ver 13.2

Decoder Output:

```
------------------------------ JM 13.2 (FREXt) ------------------------------
 Decoder config file                    : decoder.cfg
-------------------------------------------------------------------------------
 Input H.264 bitstream      : E:\Thesis\software\jm13.2_\JM\bin\ggtest1.264
 Output decoded YUV         : E:\Thesis\software\jm13.2_\JM\bin\test_dec.yuv
 Output status file         : log.dec
 Input reference file       : E:\Thesis\software\jm13.2_\JM\bin\FOOT
BALL_176x144_15_orig_01.yuv
-------------------------------------------------------------------------------
POC must = frame# or field# for SNRs to be correct
-------------------------------------------------------------------------------
  Frame        POC  Pic#   QP    SnrY     SnrU     SnrV   Y:U:V Time(ms)
-------------------------------------------------------------------------------
00000(IDR)       0     0    28  35.0122  38.3359  39.6369  4:2:0     127
00001( P )       2     1    28  23.8236  33.4617  37.0038  4:2:0     117
00002( P )       4     2    28  24.6357  34.4494  37.7906  4:2:0     121
00003( P )       6     3    28  25.8626  35.1454  38.0278  4:2:0     125
00004( P )       8     4    28  26.9735  35.9484  38.3886  4:2:0     127
00005( P )      10     5    28  28.1940  36.4707  38.6069  4:2:0     123
00006( P )      12     6    28  28.9430  36.6224  38.9282  4:2:0     137
00007( P )      14     7    28  29.6336  37.0487  38.7475  4:2:0     127
00008( P )      16     8    28  31.1590  37.4340  39.1586  4:2:0     131
00009( P )      18     9    28  32.0543  37.3335  39.3018  4:2:0     129
00010( P )      20    10    28  32.6137  37.3043  39.3878  4:2:0     132
00011( P )      22    11    28  32.8080  37.3469  39.7984  4:2:0     135
```

00012(P) 24 12 28 33.7489 37.4025 39.5302 4:2:0 132

00013(P) 26 13 28 34.0266 37.2344 39.0095 4:2:0 133

00014(P) 28 14 28 34.1013 37.1822 39.2788 4:2:0 134

00015(I) 30 15 28 36.2548 38.6228 39.5968 4:2:0 128

00016(P) 32 0 28 35.8589 37.5237 39.4560 4:2:0 130

00017(P) 34 1 28 36.0492 37.4003 38.9537 4:2:0 130

00018(P) 36 2 28 36.2468 37.4496 38.9930 4:2:0 127

00019(P) 38 3 28 36.2547 37.3613 38.9934 4:2:0 137

---------- Average SNR all frames ----------

SNR Y(dB) : 31.71

SNR U(dB) : 36.85

SNR V(dB) : 38.93

Total decoding time : 2.582 sec (7.746 fps)

Exit JM 13 (FRExt) decoder, ver 13.2

APPENDIX B

ENCODER CONFIGURATION FILE

Encoder Configuration file : encoder.cfg :

```
# New Input File Format is as follows
# <ParameterName> = <ParameterValue> # Comment
# See configfile.h for a list of supported ParameterNames
# For bug reporting and known issues see:
# https://ipbt.hhi.de
###########################################################################
#Files
###########################################################################
InputFile            = "FOOTBALL_176x144_15_orig_01.yuv"       # Input sequence
InputHeaderLength     = 0      # If the inputfile has a header, state it's length in byte here
StartFrame            = 0      # Start frame for encoding. (0-N)
FramesToBeEncoded     = 20      # Number of frames to be coded
FrameRate             = 30.0   # Frame Rate per second (0.1-100.0)
SourceWidth           = 176    # Frame width
SourceHeight          = 144    # Frame height
TraceFile            = "trace_enc.txt"      # Trace file
ReconFile             = "test_rec.yuv"       # Recontruction YUV file
OutputFile           = "test1.264"          # Bitstream
###########################################################################
# Encoder Control
###########################################################################
ProfileIDC      = 66 # Profile IDC (66=baseline, 77=main, 88=extended; FREXT #Profiles:
100=High, 110=High 10, 122=High 4:2:2, 244=High 4:4:4, 44=CAVLC 4:4:4 Intra)
IntraProfile          = 0   # Activate Intra Profile for FRExt (0: false, 1: true)                    # (e.g.
ProfileIDC=110, IntraProfile=1  =>  High 10 Intra Profile)
```

LevelIDC = 20 # Level IDC (e.g. 20 = level 2.0)

IntraPeriod = 15 # Period of I-pictures (0=only first)

IDRPeriod = 0 # Period of IDR pictures (0=only first)

AdaptiveIntraPeriod = 0 # Adaptive intra period

AdaptiveIDRPeriod = 0 # Adaptive IDR period

IntraDelay = 0 # Intra (IDR) picture delay (i.e. coding structure of PPIPPP...)

EnableIDRGOP = 0 # Support for IDR closed GOPs (0: disabled, 1: enabled)

EnableOpenGOP = 0 # Support for open GOPs (0: disabled, 1: enabled)

QPISlice = 28 # Quant. param for I Slices (0-51)

QPPSlice = 28 # Quant. param for P Slices (0-51)

FrameSkip = 0 # Number of frames to be skipped in input (e.g 2 will code every #third frame)

ChromaQPOffset = 0 # Chroma QP offset (-51..51)

DisableSubpelME = 0 # Disable Subpixel Motion Estimation (0=off/default, 1=on)

SearchRange = 32 # Max search range

MEDistortionFPel = 0 # Select error metric for Full-Pel ME (0: SAD, 1: SSE, 2: #Hadamard SAD)

MEDistortionHPel = 2 # Select error metric for Half-Pel ME (0: SAD, 1: SSE, 2: #Hadamard SAD)

MEDistortionQPel = 2 # Select error metric for Quarter-Pel ME (0: SAD, 1: SSE, 2: #Hadamard SAD)

MDDistortion = 2 # Select error metric for Mode Decision (0: SAD, 1: SSE, 2: #Hadamard SAD)

ChromaMCBuffer = 1 # Calculate Color component interpolated values in advance #and store them. Provides a trade-off between memory and computational complexity

(0: disabled/default, 1: enabled)

ChromaMEEnable = 0 # Take into account Color component information during ME
# (0: only first component/default, 1: All Color components)

NumberReferenceFrames = 1 # Number of previous frames used for inter motion #search (0-16)

PList0References = 0 # P slice List 0 reference override (0 disable, N <= #NumberReferenceFrames)

Log2MaxFNumMinus4 = 0 # Sets log2_max_frame_num_minus4 (-1 : based on #FramesToBeEncoded/Auto, >=0 : Log2MaxFNumMinus4)

Log2MaxPOCLsbMinus4 = -1 # Sets log2_max_pic_order_cnt_lsb_minus4 (-1 : Auto, #>=0 : Log2MaxPOCLsbMinus4)

GenerateMultiplePPS = 0 # Transmit multiple parameter sets. Currently parameters #basically enable all WP modes (0: diabled, 1: enabled)

ResendPPS = 0 # Resend PPS (with pic_parameter_set_id 0) for every coded #Frame/Field pair (0: disabled, 1: enabled)

MbLineIntraUpdate = 0 # Error robustness(extra intra macroblock updates)(0=off, #N: One GOB every N frames are intra coded)

RandomIntraMBRefresh = 0 # Forced intra MBs per picture

PSliceSkip = 1 # P-Slice Skip mode consideration (0=disable, 1=enable)

PSliceSearch16x16 = 1 # P-Slice Inter block search 16x16 (0=disable, 1=enable)

PSliceSearch16x8 = 1 # P-Slice Inter block search 16x8 (0=disable, 1=enable)

PSliceSearch8x16 = 1 # P-Slice Inter block search 8x16 (0=disable, 1=enable)

PSliceSearch8x8 = 1 # P-Slice Inter block search 8x8 (0=disable, 1=enable)

PSliceSearch8x4 = 1 # P-Slice Inter block search 8x4 (0=disable, 1=enable)

PSliceSearch4x8 = 1 # P-Slice Inter block search 4x8 (0=disable, 1=enable)

PSliceSearch4x4 = 1 # P-Slice Inter block search 4x4 (0=disable, 1=enable)

BSliceSkip = 1 # B-Slice Skip mode consideration (0=disable, 1=enable)

BSliceSearch16x16 = 1 # B-Slice Inter block search 16x16 (0=disable, 1=enable)

BSliceSearch16x8 = 1 # B-Slice Inter block search 16x8 (0=disable, 1=enable)

BSliceSearch8x16 = 1 # B-Slice Inter block search 8x16 (0=disable, 1=enable)

BSliceSearch8x8 = 1 # B-Slice Inter block search 8x8 (0=disable, 1=enable)

BSliceSearch8x4 = 1 # B-Slice Inter block search 8x4 (0=disable, 1=enable)

BSliceSearch4x8 = 1 # B-Slice Inter block search 4x8 (0=disable, 1=enable)

BSliceSearch4x4 = 1 # B-Slice Inter block search 4x4 (0=disable, 1=enable)

DisableIntraInInter = 0 # Disable Intra modes for inter slices

IntraDisableInterOnly = 0 # Apply Disabling Intra conditions only to Inter Slices #(0:disable/default,1: enable)

Intra4x4ParDisable = 0 # Disable Vertical & Horizontal 4x4

Intra4x4DiagDisable = 0 # Disable Diagonal 45degree 4x4

Intra4x4DirDisable = 0 # Disable Other Diagonal 4x4

Intra16x16ParDisable = 0 # Disable Vertical & Horizontal 16x16

Intra16x16PlaneDisable = 0 # Disable Planar 16x16

ChromaIntraDisable = 0 # Disable Intra Chroma modes other than DC

EnableIPCM = 0 # Enable IPCM macroblock mode

DisposableP = 0 # Enable Disposable P slices in the primary layer (0: #disable/default, 1: enable)

DispPQPOffset = 0 # Quantizer offset for disposable P slices (0: default)

###

B Slices

###

NumberBFrames = 0 # Number of B coded frames inserted (0=not used)

QPBSlice = 30 # Quant. param for B slices (0-51)

BRefPicQPOffset = -1 # Quantization offset for reference B coded pictures (-51..51)

DirectModeType = 1 # Direct Mode Type (0:Temporal 1:Spatial)

DirectInferenceFlag = 1 # Direct Inference Flag (0: Disable 1: Enable)

BList0References = 0 # B slice List 0 reference override (0 disable, N <= #NumberReferenceFrames)

BList1References = 1 # B slice List 1 reference override (0 disable, N <= #NumberReferenceFrames) # 1 List1 reference is usually recommended for normal GOP Structures. # A larger value is usually more appropriate if a more flexible

structure is used (i.e. using HierarchicalCoding)

BReferencePictures = 0 # Referenced B coded pictures (0=off, 1=B references for #secondary layer, 2=B references for primary layer)

HierarchicalCoding = 0 # B hierarchical coding (0= off, 1= 2 layers, 2= 2 full #hierarchy, 3 = explicit)

HierarchyLevelQPEnable = 1 # Adjust QP based on hierarchy level (in increments of #1). Overrides BRefPicQPOffset behavior.(0=off, 1=on)

ExplicitHierarchyFormat = "b1r0b3r0b2e2b0e2b4r2" # Explicit Enhancement GOP. #Format is {FrameDisplay_orderReferenceQP}. # Valid values for reference type is r:reference, e:non reference.

ReferenceReorder = 1 # Reorder References according to Poc distance for #HierarchicalCoding (0=off, 1=enable)

PocMemoryManagement = 1 # Memory management based on Poc Distances for #HierarchicalCoding (0=off, 1=on)

BiPredMotionEstimation = 1 # Enable Bipredictive based Motion Estimation #(0:disabled, 1:enabled)

BiPredMERefinements = 3 # Bipredictive ME extra refinements (0: single, N: N extra #refinements (1 default)

BiPredMESearchRange = 16 # Bipredictive ME Search range (8 default). Note that #range is halved for every extra refinement.

BiPredMESubPel = 2 # Bipredictive ME Subpixel Consideration (0: disabled, 1: #single level, 2: dual level)

##

SP Frames

##

SPPicturePeriodicity = 0 # SP-Picture Periodicity (0=not used)

QPSPSlice = 36 # Quant. param of SP-Slices for Prediction Error (0-51)

QPSP2Slice = 35 # Quant. param of SP-Slices for Predicted Blocks (0-51)

SI_FRAMES = 0 # SI frame encoding flag (0=not used, 1=used)

SP_output = 0 # Controls whether coefficients will be output to #encode switching SP frames (0=no, 1=yes)

SP_output_name = "low_quality.dat" # Filename for SP output coefficients

SP2_FRAMES = 0 # switching SP frame encoding flag (0=not used, #1=used)

SP2_input_name1 = "high_quality.dat" # Filename for the first swithed bitstream #coefficients

SP2_input_name2 = "low_quality.dat" # Filename for the second switched #bitstream coefficients

##

Output Control, NALs

##

SymbolMode = 0 # Symbol mode (Entropy coding method: 0=UVLC, 1=CABAC)

OutFileMode = 0 # Output file mode, 0:Annex B, 1:RTP

```
PartitionMode          =  0  # Partition Mode, 0: no DP, 1: 3 Partitions per Slice
##############################################################################
# CABAC context initialization
##############################################################################
ContextInitMethod      =  0    # Context init (0: fixed, 1: adaptive)
FixedModelNumber       =  0    # model number for fixed decision for inter slices ( 0, 1, #or 2 )
##############################################################################
# Interlace Handling
##############################################################################
PicInterlace           =  0    # Picture AFF    (0: frame coding, 1: field coding, 2:adaptive
#frame/field coding)
MbInterlace            =  0    # Macroblock AFF (0: frame coding, 1: field coding, #2:adaptive
frame/field coding, 3: frame MB-only AFF)
IntraBottom            =  0    # Force Intra Bottom at GOP Period
##############################################################################
# Weighted Prediction
##############################################################################
WeightedPrediction     =  0    # P picture Weighted Prediction (0=off, 1=explicit mode)
WeightedBiprediction   =  0    # B picture Weighted Prediciton (0=off, 1=explicit #mode,
2=implicit mode)
UseWeightedReferenceME =  1    # Use weighted reference for ME (0=off, 1=on)
##############################################################################
# Picture based Multi-pass encoding
##############################################################################
```

RDPictureDecision=0 # Perform RD optimal decision between different coded #picture versions. # If GenerateMultiplePPS is enabled then this will test different WP met # Otherwise it will test QP +-1 (0: disabled, 1: enabled)

RDPictureIntra = 0 # Perform RD optimal decision also for intra coded pictures #(0: disabled (default), 1: enabled).

RDPSliceWeightOnly = 1 # Only consider Weighted Prediction for P slices in #Picture RD decision. (0: disabled, 1: enabled (default))

RDBSliceWeightOnly = 0 # Only consider Weighted Prediction for B slices in #Picture RD decision. (0: disabled (default), 1: enabled)

##

Loop filter parameters

##

LoopFilterParametersFlag = 0 # Configure loop filter (0=parameter below ingored, #1=parameters sent)

LoopFilterDisable = 0 # Disable loop filter in slice header (0=Filter, 1=No Filter)

LoopFilterAlphaC0Offset = 0 # Alpha & C0 offset div. 2, {-6, -5, ... 0, +1, .. +6}

LoopFilterBetaOffset = 0 # Beta offset div. 2, {-6, -5, ... 0, +1, .. +6}

##

Error Resilience / Slices

##

SliceMode = 0 # Slice mode (0=off 1=fixed #mb in slice 2=fixed #bytes in slice #3=use callback)

SliceArgument = 50 # Slice argument (Arguments to modes 1 and 2 above)

num_slice_groups_minus1 = 1 # Number of Slice Groups Minus 1, 0 == no FMO, 1 == #two slice groups, etc.

slice_group_map_type = 1 # 0: Interleave, 1: Dispersed, 2: Foreground with left-#over, # 3: Box-out, 4: Raster Scan 5: Wipe 6: Explicit, slice_group_id read from #SliceGroupConfigFileName

slice_group_change_direction_flag = 0 # 0: box-out clockwise, raster scan or wipe #right, # 1: box-out counter clockwise, reverse raster scan or wipe left

slice_group_change_rate_minus1 = 85 #

SliceGroupConfigFileName = "sg0conf.cfg" # Used for slice_group_map_type 0, 2, 6

UseRedundantPicture = 0 # 0: not used, 1: enabled

NumRedundantHierarchy = 1 # 0-4

PrimaryGOPLength = 5 # GOP length for redundant allocation (1-16) # NumberReferenceFrames must be no less than PrimaryGOPLength when redundant #slice enabled

NumRefPrimary = 1 # Actually used number of references for primary slices (1-16)

##

Search Range Restriction / RD Optimization

##

RestrictSearchRange = 2 # restriction for (0: blocks and ref, 1: ref, 2: no restrictions)

RDOptimization = 1 # rd-optimized mode decision # 0: RD-off (Low complexity mode) # 1: RD-on (High complexity mode) # 2: RD-on (Fast high complexity mode - not work in FREX Profiles) # 3: with losses

CtxAdptLagrangeMult = 0 # Context Adaptive Lagrange Multiplier

0: disabled (default)

1: enabled (works best when RDOptimization=0)

FastCrIntraDecision = 1 # Fast Chroma intra mode decision (0:off, 1:on)

DisableThresholding = 0 # Disable Thresholding of Transform Coefficients (0:off, #1:on)

DisableBSkipRDO = 0 # Disable B Skip Mode consideration from RDO Mode #decision (0:off, 1:on)

SkipIntraInInterSlices = 0 # Skips Intra mode checking in inter slices if certain mode #decisions are satisfied (0: off, 1: on)

WeightY = 1 # Luma weight for RDO

WeightCb = 1 # Cb weight for RDO

WeightCr = 1 # Cr weight for RDO

##

Explicit Lambda Usage

##

UseExplicitLambdaParams = 0 # Use explicit lambda scaling parameters (0:disabled, #1:enable lambda weight, 2: use explicit lambda value)

FixedLambdaIslice = 0.1 # Fixed Lambda value for I slices

FixedLambdaPslice = 0.1 # Fixed Lambda value for P slices

FixedLambdaBslice = 0.1 # Fixed Lambda value for B slices

FixedLambdaRefBslice = 0.1 # Fixed Lambda value for Referenced B slices

FixedLambdaSPslice = 0.1 # Fixed Lambda value for SP slices

FixedLambdaSIslice = 0.1 # Fixed Lambda value for SI slices

LambdaWeightIslice = 0.65 # scaling param for I slices. This will be used as a #multiplier i.e. lambda=LambdaWeightISlice * 2^((QP-12)/3)

LambdaWeightPslice = 0.68 # scaling param for P slices. This will be used as a #multiplier i.e. lambda=LambdaWeightPSlice * 2^((QP-12)/3)

LambdaWeightBslice = 2.00 # scaling param for B slices. This will be used as a #multiplier i.e. lambda=LambdaWeightBSlice * 2^((QP-12)/3)

LambdaWeightRefBslice = 1.50 # scaling param for Referenced B slices. This will be #used as a multiplier i.e. lambda=LambdaWeightRefBSlice * 2^((QP-12)/3)

LambdaWeightSPslice = 1.50 # scaling param for SP slices. This will be used as a #multiplier i.e. lambda=LambdaWeightSPSlice * 2^((QP-12)/3)

LambdaWeightSIslice = 0.65 # scaling param for SI slices. This will be used as a #multiplier i.e. lambda=LambdaWeightSISlice * 2^((QP-12)/3)

LossRateA = 5 # expected packet loss rate of the channel for the first #partition, only valid if RDOptimization = 3

LossRateB = 0 # expected packet loss rate of the channel for the second #partition, only valid if RDOptimization = 3

LossRateC = 0 # expected packet loss rate of the channel for the third #partition, only valid if RDOptimization = 3

NumberOfDecoders = 30 # Numbers of decoders used to simulate the channel, #only valid if RDOptimization = 3

RestrictRefFrames = 0 # Doesnt allow reference to areas that have been intra #updated in a later frame.

###

Additional Stuff

###

UseConstrainedIntraPred = 0 # If 1, Inter pixels are not used for Intra macroblock #prediction.

LastFrameNumber = 0 # Last frame number that have to be coded (0: no effect)

ChangeQPI = 16 # QP (I-slices) for second part of sequence (0-51)

ChangeQPP = 16 # QP (P-slices) for second part of sequence (0-51)

ChangeQPB = 18 # QP (B-slices) for second part of sequence (0-51)

ChangeQPBSRefOffset = 2 # QP offset (stored B-slices) for second part of sequence #(-51..51)

ChangeQPStart = 0 # Frame no. for second part of sequence (0: no second part)

NumberofLeakyBuckets = 8 # Number of Leaky Bucket values

LeakyBucketRateFile = "leakybucketrate.cfg" # File from which encoder derives #rate values

LeakyBucketParamFile = "leakybucketparam.cfg" # File where encoder stores #leakybucketparams

NumberFramesInEnhancementLayerSubSequence = 0 # number of frames in the #Enhanced Scalability Layer(0: no Enhanced Layer)

SparePictureOption = 0 # (0: no spare picture info, 1: spare picture available)

SparePictureDetectionThr = 6 # Threshold for spare reference pictures detection

SparePicturePercentageThr = 92 # Threshold for the spare macroblock percentage

PicOrderCntType = 0 # (0: POC mode 0, 1: POC mode 1, 2: POC mode 2)

##

#Rate control

##

RateControlEnable = 0 # 0 Disable, 1 Enable

Bitrate = 45020 # Bitrate(bps)

InitialQP = 0 # Initial Quantization Parameter for the first I frame

InitialQp depends on two values: Bits Per Picture,

and the GOP length

BasicUnit = 11 # Number of MBs in the basic unit

should be a fractor of the total number

of MBs in a frame

ChannelType = 0 # type of channel(1=time varying channel; 0=Constant #channel)

RCUpdateMode = 0 # Rate Control type. Modes supported :

0 = original JM rate control

1 = rate control that is applied to all frames regardless of the slice type, # 2 =

original plus intelligent QP selection for I and B slices (including Hierarchical), # 3 = original + hybrid quadratic rate control for I and B slice using bit rate statistics

RCISliceBitRatio = 1.0 # target ratio of bits for I-coded pictures compared to P-#coded Pictures (for RCUpdateMode=3)

RCBSliceBitRatio0 = 0.5 # target ratio of bits for B-coded pictures compared to P-#coded Pictures - temporal level 0 (for RCUpdateMode=3)

RCBSliceBitRatio1 = 0.25 # target ratio of bits for B-coded pictures compared to P-#coded Pictures - temporal level 1 (for RCUpdateMode=3)

RCBSliceBitRatio2 = 0.25 # target ratio of bits for B-coded pictures compared to P-#coded Pictures - temporal level 2 (for RCUpdateMode=3)

RCBSliceBitRatio3 = 0.25 # target ratio of bits for B-coded pictures compared to P-#coded Pictures - temporal level 3 (for RCUpdateMode=3)

RCBSliceBitRatio4 = 0.25 # target ratio of bits for B-coded pictures compared to P-#coded Pictures - temporal level 4 (for RCUpdateMode=3)

RCBoverPRatio = 0.45 # ratio of bit rate usage of a B-coded picture over a P-#coded picture for the SAME QP (for RCUpdateMode=3)

RCIoverPRatio = 3.80 # ratio of bit rate usage of an I-coded picture over a P-#coded picture for the SAME QP (for RCUpdateMode=3)

RCMinQPPSlice = 8 # minimum P Slice QP value for rate control

RCMaxQPPSlice = 40 # maximum P Slice QP value for rate control

RCMinQPBSlice = 8 # minimum B Slice QP value for rate control

RCMaxQPBSlice = 46 # maximum B Slice QP value for rate control

RCMinQPISlice = 8 # minimum I Slice QP value for rate control

RCMaxQPISlice = 36 # maximum I Slice QP value for rate control

RCMinQPSPSlice = 8 # minimum SP Slice QP value for rate control

RCMaxQPSPSlice = 40 # maximum SP Slice QP value for rate control

```
RCMinQPSISlice          =  8    # minimum SI Slice QP value for rate control
RCMaxQPSISlice          = 36    # maximum SI Slice QP value for rate control
###########################################################################
#Fast Mode Decision
###########################################################################
EarlySkipEnable         = 0     # Early skip detection (0: Disable 1: Enable)
SelectiveIntraEnable    = 0     # Selective Intra mode decision (0: Disable 1: Enable)
###########################################################################
#FREXT stuff
###########################################################################
YUVFormat               = 1     # YUV format (0=4:0:0, 1=4:2:0, 2=4:2:2, 3=4:4:4)
RGBInput                = 0     # 1=RGB input, 0=GBR or YUV input
SeparateColourPlane     = 0     # 4:4:4 coding: 0=Common mode, 1=Independent mode
BitDepthLuma            = 8     # Bit Depth for Luminance (8...12 bits)
BitDepthChroma          = 8     # Bit Depth for Chrominance (8...12 bits)
CbQPOffset              = 0     # Chroma QP offset for Cb-part (-51..51)
CrQPOffset              = 0     # Chroma QP offset for Cr-part (-51..51)
Transform8x8Mode            =  0        # (0: only 4x4 transform, 1: allow using 8x8 transform
additionally, 2: only 8x8 transform)
ReportFrameStats        = 0     # (0:Disable Frame Statistics 1: Enable)
DisplayEncParams        = 0     # (0:Disable Display of Encoder Params 1: Enable)
Verbose                 = 1     # level of display verboseness (0:short, 1:normal, 2:detailed)
###########################################################################
#Q-Matrix (FREXT)
###########################################################################
QmatrixFile             = "q_matrix.cfg"
```

ScalingMatrixPresentFlag = 0 # Enable Q_Matrix (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag0 = 3 # Intra4x4_Luma (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag1 = 3 # Intra4x4_ChromaU (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag2 = 3 # Intra4x4_chromaV (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag3 = 3 # Inter4x4_Luma (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag4 = 3 # Inter4x4_ChromaU (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag5 = 3 # Inter4x4_ChromaV (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag6 = 3 # Intra8x8_Luma (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag7 = 3 # Inter8x8_Luma (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag8 = 1 # Intra8x8_ChromaU for 4:4:4 (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag9 = 3 # Inter8x8_ChromaU for 4:4:4 (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag10 = 2 # Intra8x8_ChromaV for 4:4:4 (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS)

ScalingListPresentFlag11 = 3 # Inter8x8_ChromaV for 4:4:4 (0 Not present, 1 Present in SPS, 2 Present in PPS, 3 Present in both SPS & PPS

```
###########################################################################
#Rounding Offset control
###########################################################################
OffsetMatrixPresentFlag  = 0     # Enable Explicit Offset Quantization Matrices  (0: disable 1: enable)
QOffsetMatrixFile        = "q_offset.cfg" # Explicit Quantization Matrices file
AdaptiveRounding         = 1    # Enable Adaptive Rounding based on JVT-N011 (0: disable, 1: enable)
AdaptRoundingFixed       = 1     # Enable Global Adaptive rounding for all qps (0: disable, 1: enable - default/old)
AdaptRndPeriod           = 1      # Period in terms of MBs for updating rounding offsets. # 0 performs update at the picture level. Default is 16. 1 is as in JVT-N011.
AdaptRndChroma           = 1    # Enables coefficient rounding adaptation for chroma
AdaptRndWFactorIRef      = 4    # Adaptive Rounding Weight for I/SI slices in reference pictures /4096
AdaptRndWFactorPRef      = 4     # Adaptive Rounding Weight for P/SP slices in reference pictures /4096
AdaptRndWFactorBRef      = 4    # Adaptive Rounding Weight for B slices in reference pictures /4096
AdaptRndWFactorINRef     = 4     # Adaptive Rounding Weight for I/SI slices in non reference pictures /4096
AdaptRndWFactorPNRef     = 4    # Adaptive Rounding Weight for P/SP slices in non reference pictures /4096
AdaptRndWFactorBNRef     = 4     # Adaptive Rounding Weight for B slices in non reference pictures /409
```

AdaptRndCrWFactorIRef = 4 # Chroma Adaptive Rounding Weight for I/SI slices in reference pictures /4096

AdaptRndCrWFactorPRef = 4 # Chroma Adaptive Rounding Weight for P/SP slices in reference pictures /4096

AdaptRndCrWFactorBRef = 4 # Chroma Adaptive Rounding Weight for B slices in reference pictures /4096

AdaptRndCrWFactorINRef = 4 # Chroma Adaptive Rounding Weight for I/SI slices in non reference pictures /4096

AdaptRndCrWFactorPNRef = 4 # Chroma Adaptive Rounding Weight for P/SP slices in non reference pictures /4096

AdaptRndCrWFactorBNRef = 4 # Chroma Adaptive Rounding Weight for B slices in non reference pictures /4096

###

#Lossless Coding (FREXT)

###

QPPrimeYZeroTransformBypassFlag = 0 # Enable lossless coding when qpprime_y is zero (0 Disabled, 1 Enabled)

###

#Fast Motion Estimation Control Parameters

###

SearchMode = 0 # Use fast motion estimation (0=disable/default, 1=UMHexagonS,

2=Simplified UMHexagonS, 3=EPZS patterns)

UMHexDSR = 1 # Use Search Range Prediction. Only for UMHexagonS method

(0:disable, 1:enabled/default)

UMHexScale = 3 # Use Scale_factor for different image sizes. Only for UMHexagonS method

```
                    # (0:disable, 3:/default)
                    # Increasing value can speed up Motion Search.
EPZSPattern           = 2    # Select EPZS primary refinement pattern.
                    # (0: small diamond, 1: square, 2: extended diamond/default,
                    # 3: large diamond, 4: SBP Large Diamond,
                    # 5: PMVFAST )
EPZSDualRefinement    = 3    # Enables secondary refinement pattern.
                    # (0:disabled, 1: small diamond, 2: square,
                    # 3: extended diamond/default, 4: large diamond,
                    # 5: SBP Large Diamond, 6: PMVFAST )
EPZSFixedPredictors   = 2    # Enables Window based predictors
                        # (0:disabled, 1: P only, 2: P and B/default)
EPZSTemporal          = 1    # Enables temporal predictors
                        # (0: disabled, 1: enabled/default)
EPZSSpatialMem        = 1    # Enables spatial memory predictors
                    # (0: disabled, 1: enabled/default)
EPZSMinThresScale     = 0    # Scaler for EPZS minimum threshold (0 default).
                    # Increasing value can speed up encoding.
EPZSMedThresScale     = 1    # Scaler for EPZS median threshold (1 default).
                    # Increasing value can speed up encoding.
EPZSMaxThresScale     = 2    # Scaler for EPZS maximum threshold (1 default).
                    # Increasing value can speed up encoding.
EPZSSubPelME          = 1    # EPZS Subpel ME consideration
EPZSSubPelMEBiPred    = 1    # EPZS Subpel ME consideration for BiPred partitions
EPZSSubPelThresScale  = 2    # EPZS Subpel ME Threshold scaler
EPZSSubPelGrid        = 0    # Perform EPZS using a subpixel grid
```

```
###########################################################################
# SEI Parameters
###########################################################################ToneM
appingSEIPresentFlag = 0    # Enable Tone mapping SEI  (0 Not present, 1 Present)
ToneMappingFile          = "ToneMapping.cfg"
GenerateSEIMessage       = 0                # Generate an SEI Text Message
SEIMessageText           = "H.264/AVC Encoder"  # Text SEI Message
###########################################################################
# VUI Parameters
###########################################################################
# the variables below do not affect encoding and decoding
# (many are dummy variables but others can be useful when supported by the decoder)
EnableVUISupport                 = 0     # Enable VUI Parameters
VUI_aspect_ratio_info_present_flag          = 0
VUI_aspect_ratio_idc                = 1
VUI_sar_width                    = 0
VUI_sar_height                   = 0
VUI_overscan_info_present_flag           = 0
VUI_overscan_appropriate_flag            = 0
VUI_video_signal_type_present_flag          = 0
VUI_video_format                 = 5
VUI_video_full_range_flag              = 0
VUI_colour_description_present_flag         = 0
VUI_colour_primaries                = 2
VUI_transfer_characteristics             = 2
VUI_matrix_coefficients             = 2
```

VUI_chroma_location_info_present_flag = 0

VUI_chroma_sample_loc_type_top_field = 0

VUI_chroma_sample_loc_type_bottom_field = 0

VUI_timing_info_present_flag = 0

VUI_num_units_in_tick = 1000

VUI_time_scale = 60000

VUI_fixed_frame_rate_flag = 0

nal hrd parameters

VUI_nal_hrd_parameters_present_flag = 0

VUI_nal_cpb_cnt_minus1 = 0

VUI_nal_bit_rate_scale = 0

VUI_nal_cpb_size_scale = 0

VUI_nal_bit_rate_value_minus1 = 0

VUI_nal_cpb_size_value_minus1 = 0

VUI_nal_vbr_cbr_flag = 0

VUI_nal_initial_cpb_removal_delay_length_minus1 = 23

VUI_nal_cpb_removal_delay_length_minus1 = 23

VUI_nal_dpb_output_delay_length_minus1 = 23

VUI_nal_time_offset_length = 24

vlc hrd parameters

VUI_vcl_hrd_parameters_present_flag = 0

VUI_vcl_cpb_cnt_minus1 = 0

VUI_vcl_bit_rate_scale = 0

VUI_vcl_cpb_size_scale = 0

VUI_vcl_bit_rate_value_minus1 = 0

VUI_vcl_cpb_size_value_minus1 = 0

VUI_vcl_vbr_cbr_flag = 0

VUI_vcl_initial_cpb_removal_delay_length_minus1 = 23

VUI_vcl_cpb_removal_delay_length_minus1 = 23

VUI_vcl_dpb_output_delay_length_minus1 = 23

VUI_vcl_time_offset_length = 24

VUI_low_delay_hrd_flag = 0

other params (i.e. bitsream restrictions)

VUI_pic_struct_present_flag = 0

VUI_bitstream_restriction_flag = 0

VUI_motion_vectors_over_pic_boundaries_flag = 1

VUI_max_bytes_per_pic_denom = 0

VUI_max_bits_per_mb_denom = 0

VUI_log2_max_mv_length_vertical = 16

VUI_log2_max_mv_length_horizontal = 16

VUI_num_reorder_frames = 16

VUI_max_dec_frame_buffering = 16

APPENDIX C

DECODER CONFIGURATION FILE

Decoder Configuration file: decoder.cfg:

E:\Thesis\software\jm13.2_\JM\bin\ggtest1.264 .....H.264/AVC coded bitstream

E:\Thesis\software\jm13.2_\JM\bin\test_dec.yuv .....Output file, YUV/RGB

E:\Thesis\software\jm13.2_\JM\bin\FOOTBALL_176x144_15_orig_01.yuv #.....Ref sequence (for SNR)

1 .....Write 4:2:0 chroma components for monochrome streams

0 .....NAL mode (0=Annex B, 1: RTP packets)

0 .....SNR computation offset

2 .....Poc Scale (1 or 2)

500000 .....Rate_Decoder

104000 .....B_decoder

73000 .....F_decoder

leakybucketparam.cfg .....LeakyBucket Params

2 .....Err Concealment (0:Off,1:Frame Copy,2:weighted averaging,3:motion vector #interpolation)

2 .....Reference POC gap (2: IPP (Default), 4: IbP / IpP)

2 .....POC gap (2: IPP /IbP/IpP (Default), 4: IPP with frame skip = 1 etc.)

0 .....Silent decode

This is a file containing input parameters to the JVT H.264/AVC decoder.

REFERENCES

[1] T. Stockhammer, M. M. Hannuksela and T. Wiegand, “H.264/AVC in Wireless Environments”, IEEE Trans. Circuits and Systems for Video Technology, Vol. 13, pp. 657- 673, July 2003.

[2] S. K. Bandyopadhyay, et al, “An Error Concealment Scheme for Entire Frame Losses for H.264/AVC”, Proc. IEEE Sarnoff Symposium, Mar. 2006.

[3] Soon-kak Kwon, A. Tamhankar and K.R. Rao, ”Overview of H.264 / MPEG-4 Part 10”, J. Visual Communication and Image Representation, vol. 17, pp.186-216, April 2006.

[4] J. Konrad and E. Dubois, “Bayesian Estimation of Motion Vector Field”, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 14, pp. 910-926, Sept. 1992.

[5] M. Ghanbari and V. Seferidis, “Cell-Loss Concealment in ATM Video Codecs”, IEEE Trans. Circuits and Systems for Video Technology, vol. 3, pp. 238-247, June 1993.

[6] M. Wada, “Selective Recovery of Video Packet Loss using Error Concealment,” IEEE Journal on Selected Areas in Communication, vol. 7, pp. 807-814, June 1989.

[7] Video Coding Standards 6. MPEG-1. ISO/IEC 11172-2 (’93).

[8] P.Salama, N. Shroff and E. J. Delp, “Error Concealment in Encoded Video Streams”, Proc. IEEE ICIP, vol. 1, pp. 9-12, 1995.

[9] H. Ha, C. Yim and Y.Y.Kim, “Packet Loss Resilience using Unequal Forward Error Correction Assignment for Video Transmission over Communication Networks,” ACM digital library on Computer Communications, vol. 30, pp. 3676-3689, Dec. 2007.

[10] Y. Chen, et al, “An Error Concealment Algorithm for Entire Frame Loss in Video Transmission”, Microsoft Research Asia, Picture Coding Symposium, Dec. 2004.

[11] L. Liu, S. Zhang, X. Ye and Y. Zhang, “Error Resilience Schemes of H.264/AVC for 3G Conversational Video”, Proc. IEEE Conf. Computer and Information Technology, pp. 657- 661, Sept. 2005.

[12] S. Wenger, “H.264/AVC over IP” IEEE Trans. Circuits and Systems for Video Technology, vol. 13, pp. 645-656, July 2003.

[13] T. Aladrovic, M. Matic, and M. Kos, “An Error Resilience Scheme for Layered Video Coding” IEEE Int. symposium of Industrial Electronics, vol. 3, pp. 1285-1290, June 2005.

[14] T. Wiegand, et al, “Overview of the H.264/AVC Video Coding Standard” IEEE Trans. Circuits and Systems for Video Technology, vol. 13, pp. 560-576, June 2003.

[15] S.Kumar, et al,” Error resiliency schemes in H.264/AVC standard,” J. Visual Communication and Image Representation, vol. 17, pp. 425-450, April 2006.

[16] T. Thaipanich, P.H. Wu, and C.C. J Kuo, “Low-Complexity Mobile Video Error Concealment Using OBMA”, IEEE Int. Conf. on Consumer Electronics, pp. 753-761, Jan 2008.

[17] JVT”Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T rec. H.264– ISO/IEC 14496-10 AVC),” March 2003, JVT-G050 available on http://ip.hhi.de/imagecom_G1/assets/pdfs/JVT- G050.pdf.

[18] R. Schafer, T. Wiegand, and H. Schwarz,”The emerging H.264/AVC standard,” EBU Technical Review, Special Issue on Best of 2003, Dec. 2003.

[19] M. M. Ghandi and M. Ghanbari, “Layered H.264 video transmission with hierarchical QAM” Electronic Systems Engineering Department, University of Essex, UK available in: http://privatewww.essex.ac.uk/.

[20] V. S. Kolkeri, J. H. Lee and K. R. Rao,” Error concealment techniques in H.264/AVC for wireless video transmission in mobile networks” submitted to International Conf. in Sinhgad Technical Education Society, Image Processing-2009.

[21] Z. Wang, L. Lu, and A.C. Bovik, ”Video quality assessment based on structural distortion measurement,” Signal Processing: Image Communication, vol. 19, no. 2, pp. 121-132, Feb. 2004.

[22] F. Chiaraluce, et al, "Performance Evaluation of Error Concealment Techniques in H.264 Video Coding," *Proc. PCS04 Picture Coding Symposium*, Dec. 15--17 2004, Nob Hill Masonic Center, San Francisco (CA), USA..

[23] K. R. Rao, Z. S. Bojkovic, and D. A. Milovanovic, Wireless Multimedia Communications. Boca Raton, FL: CRC press, 2009.

[24] Z. Wang, et al, “Image Quality Assessment: From Error Visibility to Structural Similarity”, IEEE Trans. Image Processing, vol. 13, pp.600-612, April 2004.

[25] LSI Logic Corporation: H.264/MPEG-4 AVC Video Compression Tutorial, available in: http://www.cs.ucla.edu/classes/fall03/cs218/paper/H.264_MPEG4_Tutorial.pdf

[26] Panasonic Corporation: AVC-Intra (H.264 Intra) Compression Tutorial, available in: ftp://ftp.panasonic.com/pub/Panasonic/Drivers/PBTS/papers/WP_AVC-Intra.pdf

[27] H.264/AVC Reference Software Download: http://iphome.hhi.de/suehring/tml/download/

[28] M.T. Sun, and A.R. Reibman, Compressed Video over Networks, Marcel Dekker, New York, 2001.

[29] H.S. Malvar, et al, ”Low-complexity transform and quantization in H.264/AVC,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no.7, pp. 598-603, July 2003.

[30] JVT ”Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T rec. H.264– ISO/IEC 14496-10 AVC),” March 2003, JVT-G050 available on http://ip.hhi.de/imagecom_G1/assets/pdfs/JVT-G050.pdf.

[31] Z. Wang, L. Lu, and A.C. Bovik, ”Video quality assessment based on structural distortion measurement,” Signal Processing: Image Communication, vol. 19, no. 2, pp. 121-132, Feb. 2004.

[32] Z. Wang, E.P. Simoncelli, and A.C. Bovik, ”Multi-scale structural similarity for image quality assessment, ” Proc. IEEE Asilomar Conf. on Signals, Systems and Computers, (Asilomar), vol. 2, pp. 1398-1402, Nov. 2003.

[33] S. Winkler, A. Sharma, and D. McNally, ”Perceptual video quality and blockiness metrics for multimedia streaming applications,” Proc. of the International Symposium on Wireless Personal Multimedia Communications, pp. 547–552, Aalborg, Denmark, Sept. 2001.

[34] D. Kumar, P. Shastry and A. Basu, “Overview of the H.264 / AVC”, 8th Texas Instruments Developer Conference India, 30 Nov – 1 Dec 2005, Bangalore.

[35] R. Schäfer, T. Wiegand and H. Schwarz, “The emerging H.264/AVC standard”, EBU Technical Review, Jan. 2003.

[36] A. A. Moghrabi, ”Error concealment for video transmission over wireless networks”, Diploma research.

[37] J. C. Ikuno, “Performance of an error detection mechanism for damaged H.264/AVC sequences”, Master’s thesis, 2007.

BIOGRAPHICAL INFORMATION

Vineeth Shetty Kolkeri was born in Kundapur, India, in 1985. He received the Bachelor of Engineering degree in Electronics and Communication Engineering from Vivesvaraya Technological University, India, in June 2006. His current research interests include video coding, embedded, digital signal processing and wireless telecommunications. He is currently pursuing his Master's degree in Electrical Engineering at The University of Texas at Arlington. He is a member of the multimedia processing research group, guided by Dr. K. R. Rao. He worked as an intern in Qualcomm Inc from Aug. 2008-May 2009.