IMPLEMENTATION OF AN ADAPTIVE BLOCK FILTER ON SUB-BLOCKS OF A MACROBLOCK IN H.264/AVC

by

BHAVANA PRABHAKAR

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2013

Acknowledgements

Firstly, I would thank my advisor Prof. K. R. Rao for his guidance, his tireless assistance and dedication. His enthusiasm in maintaining new trend in the research areas has inspired me. I am very grateful to him for having given me an opportunity to work under his guidance. This thesis would not have been possible without his constant support and encouragement.

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

I appreciate all members of Multimedia Processing Lab, Shreyanka Subbarayappa, Dilip Prasanna Kumar for their support during my research work. I would also like to thank my friends Shashank Vijay, Rohit Sridharan, Karthik Suresh, Radhika Vasani, my Intel manager Sunita Joshi, my mentors Yasin Mohammad and Rohan Adyanthaya who helped me in many ways leading to my thesis completion.

Finally, I am grateful to my family; my father V. N. Prabhakar Rao, my mother M. L. Sharada Rao for their support, patience, and immense encouragement during my graduate journey.

April 16, 2012

Abstract

IMPLEMENTATION OF AN ADAPTIVE BLOCK FILTER ON SUB-BLOCKS OF A MACROBLOCK

Bhavana Prabhakar

The University of Texas at Arlington, 2013

Supervising Professor: K.R.Rao

Plenty of coding tools are being developed to boost the efficiency of H.264/AVC. Some of these tools are implemented in the key technical area (KTA). The adaptive interpolation filter (AIF) [3], the high precision interpolation filter [5], and the adaptive loop filter (ALF) [4] have already been introduced in order to provide a more precise reference picture. Other efficient coding methods, such as the motion vector competition [6] and the extended macro-block (MB) [7], are also implemented in the KTA [16].

In order to improve the coding efficiency of H.264/AVC, an adaptive prediction block filter (APBF) based on Wiener filter is implemented on every Sub-Block (SB) of a macro-block (MB) where each MB is decomposed into 4x4 SBs. For each SB using the prediction and reconstruction results of the neighboring SBs the filter coefficients are calculated. The proposed filter is applied to the prediction block of the current SB [16], and the filtered block is selectively used depending on the rate-distortion (RD) cost [9]. For each SB, if the APBF is used, the residual signal between the prediction and original signal of the SB by reducing the number of bits required for encoding, the coding efficiency is improved. Additionally, since the same filter coefficients can be obtained in

the decoder, they do not need to be encoded into the bit-stream. The proposed method achieves a 5.04% bitrate saving on an average when compared to H.264/AVC.

Table of Contents

List of Illustrations

List of Tables

List of Acronyms

- AIF: Adaptive interpolation filter
- ALF: Adaptive loop filter
- APBF: Adaptive Prediction Block Filtering
- ASO: Arbitrary slice ordering
- AVC: Advanced video coding
- BALF: Block-based adaptive loop filter
- BD – ROM: Blue ray disc – read only memory
- CABAC: Context-adaptive binary arithmetic coding
- CAVLC: Context-adaptive variable-length coding
- CBP: Coded Block Pattern
- CIF: Common intermediate format
- DCT: Discrete cosine transform
- DP: Data partitioning
- FMO: Flexible macroblock ordering
- HD – DVD: High definition - digital video disc
- ITU: International telecommunication union
- KTA: Key technical areas
- MB: Macro-block
- MCIF: Motion compensated interpolation filter
- MPEG: Moving picture experts group
- PSNR: Peak signal to noise ratio
- QALF: Quad-tree adaptive loop filter
- RD: Rate distortion

- RDO: Rate distortion optimization
- RDOQ: Rate-Distortion Optimized Quantization
- RS: Redundant slices
- SB: Sub-block
- SI: Side information
- SP: Simple profile
- VCEG: Video coding experts group
- WVGA: Wide video graphics array
- WQVGA: Wide quarter video graphics array

Chapter 1

Introduction

Multimedia

Over the last couple of years multimedia computing has emerged as a major area of research. Multimedia computer systems have a variety of applications by combining a wide range of information sources like graphics, text, images, audio, and full-motion video. Multimedia today is basically a combination of three industries:- computer, communication, and broadcasting.

The fundamental characteristic of multimedia systems is that they integrate continuous media, such as voice, video, and animated graphics. This emphasizes the need for multimedia systems to handle data with strict timing requirements and a at high rate. The requirement of continuous media in distributed systems implies the need for continuous data transfer over relatively long periods of time [21]. Additional important primary issues are:

1. Media synchronization
2. Very large storage required,
3. Need for special indexing and retrieval techniques, tuned to multimedia data types [20]

Figure 1.1 elucidates the basic principles of dealing with continuous media examples of operations on these media. Audio and video information can be either stored, used in an application, such as training, or can be used interactively, such as in multimedia conferencing, or non-interactively, in TV broadcast applications [21].

Correspondingly, still images, can be used in an interactive mode such as browsing and retrieval, or in non-interactive slide show mode.

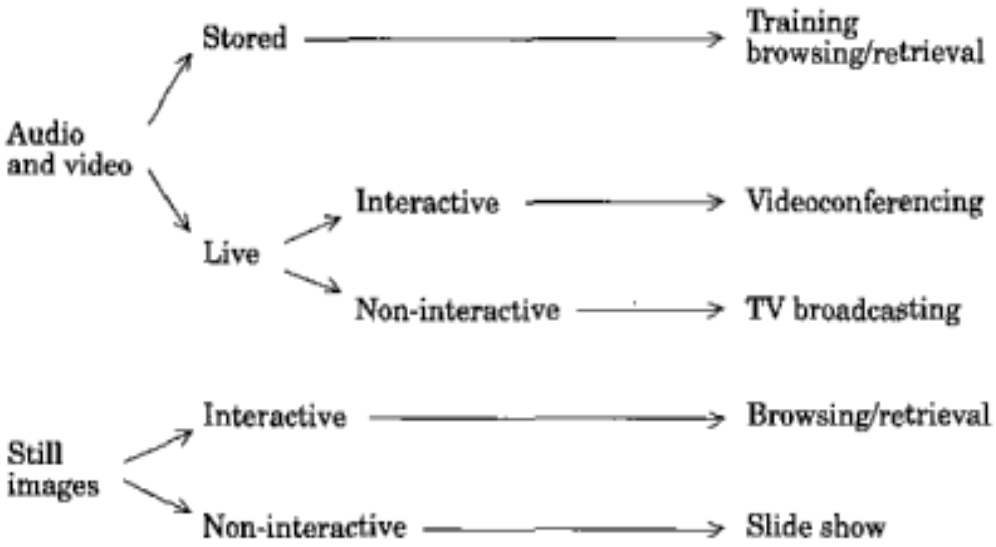


Figure 1.1 Basic principles in dealing with continuous media [21]

Multimedia research areas include many areas in the computer field such as fast processors, high-speed networks, large-capacity storage devices, new algorithms and data structures, parallel processing methods, video and audio compression algorithms, object-oriented programming, graphics systems, real-time operating systems, hypertext and hypermedia, languages for scripting, human-computer interaction, and complex architectures for distributed systems.

Storage Requirements for Multimedia Applications

Audio, image, and video signals require a vast amount of data for their representation. Table 1.1 illustrates the mass storage requirements for various media types, namely text, image, audio, and video.

Table 1.1 Mass storage requirements by various media types [21] (B=byte)

	Text	Image	Audio	Video
Object type	-ASCII -EBCDIS	-Bitmapped graphics -Still photos -Faxes	Non coded stream of digitized audio or voice	TV analog or digital image with synched streams at 24-30 frames/s
Size and bandwidth	2KB per page	-Simple 64KB/image -Detailed(color) 7.5MB/image	Voice/Phone 8 KHz/8 bits (mono) 6-44 KB/s AUDIO CD 44.1 KHz/ 16 bit/stereo 176 KB/s	27.7 MB/s for 640 × 480 × 24 pixels per frame (24-bit color) 30 frames/s

There are three main reasons why present multimedia systems require data to be compressed. They are:

(a) Large storage requirements of multimedia data

(b) Relatively slow storage devices which do not allow playing multimedia data (specifically video) in real-time

(c) The present network's bandwidth, which does not allow real-time video data transmission

Classification of Compression Techniques

Compression of digital data is based on various computational algorithms, which can be implemented either in software or in hardware [21]. Compression techniques are classified into two categories: (a) *Lossless* (b) *Lossy approaches [22]*

Lossless techniques are capable of recovering the original representation perfectly. Lossy techniques involve algorithms which recover the presentation to be

analogous to the original one. The lossy techniques deliver higher compression ratios, and therefore they are more often applied in image and video compression than lossless techniques. The classification schemes for lossless and lossy compression are presented in Figures 1.2 and 1.3 respectively.

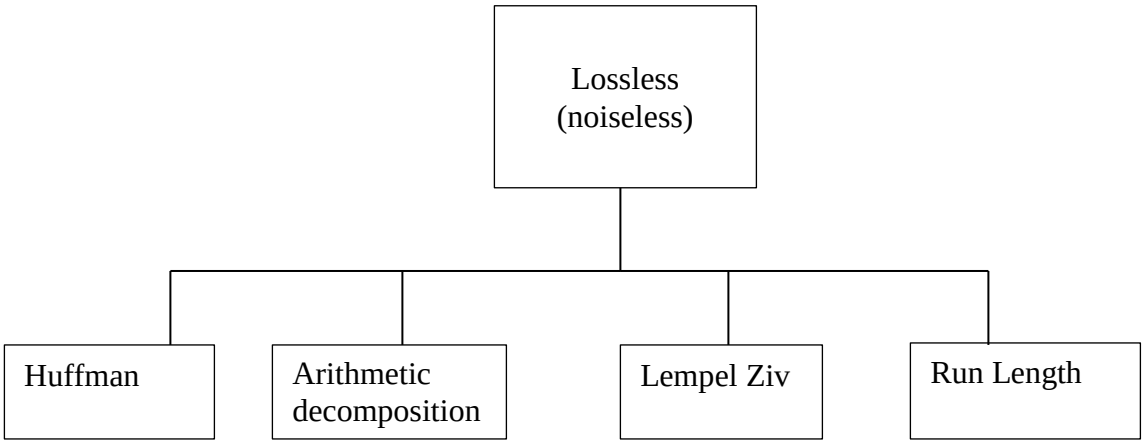


Figure 1.2 Lossless compression schemes [21]

The lossy techniques are classified as: (i) Prediction-based techniques (ii) Frequency-oriented techniques (iii) Importance-oriented techniques

Prediction-based techniques predict the subsequent values by observing previous values. Frequency-oriented techniques apply the discrete cosine transform (DCT). Importance-oriented techniques use the characteristics of the image such as color as the base for the compression [21].

The hybrid compression techniques are those where MPEG, H.264, HEVC etc. are used. These techniques combine several approaches like DCT, vector quantization, differential pulse code modulation, motion compensation, entropy coding, etc.

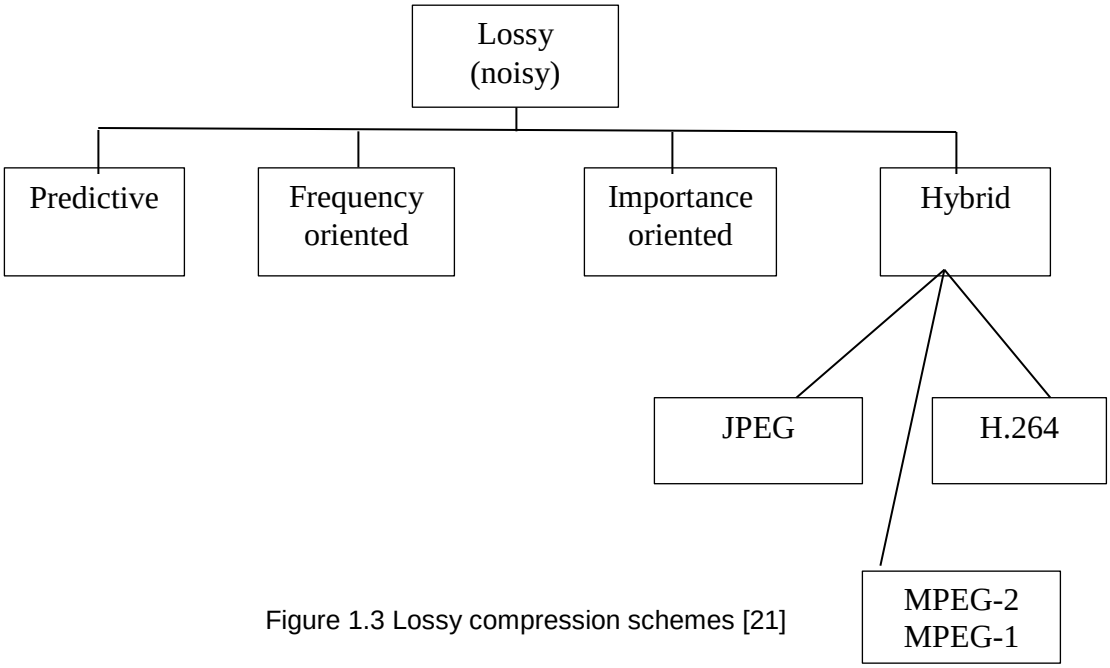


Figure 1.3 Lossy compression schemes [21]

Summary

The importance of multimedia, the requirement for compression and the techniques of compression are described. In Chapter 2, video compression mainly H.264/AVC standard and its implementation are discussed.

Chapter 2

Video compression

The entertainment videos with reasonable dimensions and frame rates would require storage space and bandwidth far in excess of that available on a CD-ROM or DVD [23]. Therefore providing consumer quality video on compact disc would not be possible. If high bandwidth technology such as fibre-optic cable was in place, even then per-byte cost of transmission needs to be very low before it would be feasible to use it for the tremendous amounts of data required by HDTV [23]. Even if the transportation and storage problems of digital video were eliminated, the processing power required to handle such volumes of data would make the receiver hardware extremely expensive.

Although significant gains in storage, processor technology and transmission have been achieved in the past years, it is mainly the decrease in the volume of data that needs to be transmitted, stored, and processed that has made extensive use of digital video a prospect. The reduction of bandwidth has been made possible by the progress in compression technology. Compression reduces the bandwidth required to store and transmit digital video [23].

H.264/AVC

The two main standards bodies that are doing equivalent development of video compression standards are: The Moving Picture Experts Group (MPEG) by the ISO/IEC and the International Telecommunication Union (ITU). H.264/AVC is one of the newer video coding standards of the ITU-T video coding experts group (VCEG) and the MPEG. H.264 is a video compression scheme that has become the worldwide digital video standard for consumer electronics and personal computers. In particular, H.264 has already been selected as a key compression scheme (codec) for the next generation of

optical disc formats, HD-DVD and Blu-ray disc (sometimes referred to as BD or BD-ROM) [13].

H.264 has been adopted by MPEG to be a key video compression scheme in the MPEG-4 format for digital media exchange. H.264 is sometimes referred to as “MPEG-4 Part 10” (part of the MPEG-4 specification), or as “AVC” and has achieved a significant increase in the rate-distortion efficiency providing, a factor of two in bit-rate savings when compared with existing standards such as MPEG-2 Video [13, 14].

H.264/AVC like any other motion-based codecs, uses the following basic principles of video compression[24]:

- Transform for reduction of spatial correlation.
- Quantization for controlling the bitrate.
- Motion compensated prediction for reduction of temporal correlation.
- Entropy coding for reduction in statistical correlation.

A video is a pile of frames attached one after another, in most video data the difference between consecutive frames is insignificant and hence data cutback is possible. The video frames are known as picture types or frame types. The frame types are classified as I, P and B frames. I frames are intra coded frames, they do not use other video frames for compression and therefore are least compressible, P frames are predictive frames, they use data from previous frames for compression and are more compressible compared to I-frames. B frames are bi-predictive frames ,they use both past and/or future frames for compression and hence are the most compressible frames. A frame is divided into several blocks known as macroblocks (MBs) which are usually 16x16 pixels, on Y the plane of the original image. A MB can be represented in several ways in YC_bC_r space.Figure 2.1 shows different formats for YC_bC_r color space.

The 4:4:4 represents 4 Y blocks, 4 C_b and 4 C_r blocks respectively , it represents a complete bandwidth video and contains information as the data if it would be in RGB color space. The 4:2:2 contains half the 4:4:4 chrominance information and 4:2:0 contains one quarter of the 4.4.4 chrominance information. H.264/AVC video codecs can support all the formats, but most consumer level products use 4:2:0 mode.

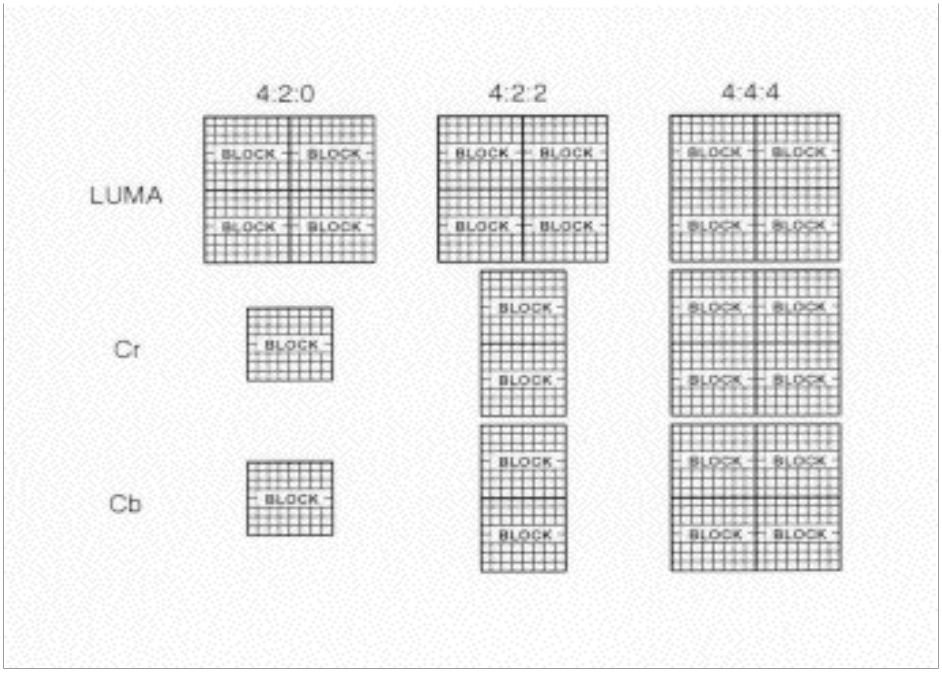


Figure 2.1 Formats (4:4:4, 4:2:2 and 4:2:0) for YC_bC_r color space [25].

Intra prediction compression uses only the present frame for prediction. It predicts the movements from the neighboring blocks. It uses the mode information from adjoining blocks. Intra frame prediction is frequently used in uniform zones of the picture where there is no too much movement.

While in inter prediction the encoder divides one frame into MBs and attempts to find a block similar to the frame it is encoding from a reference frame. Figure 2.2 shows block-based motion compensation in H.264 encoder.

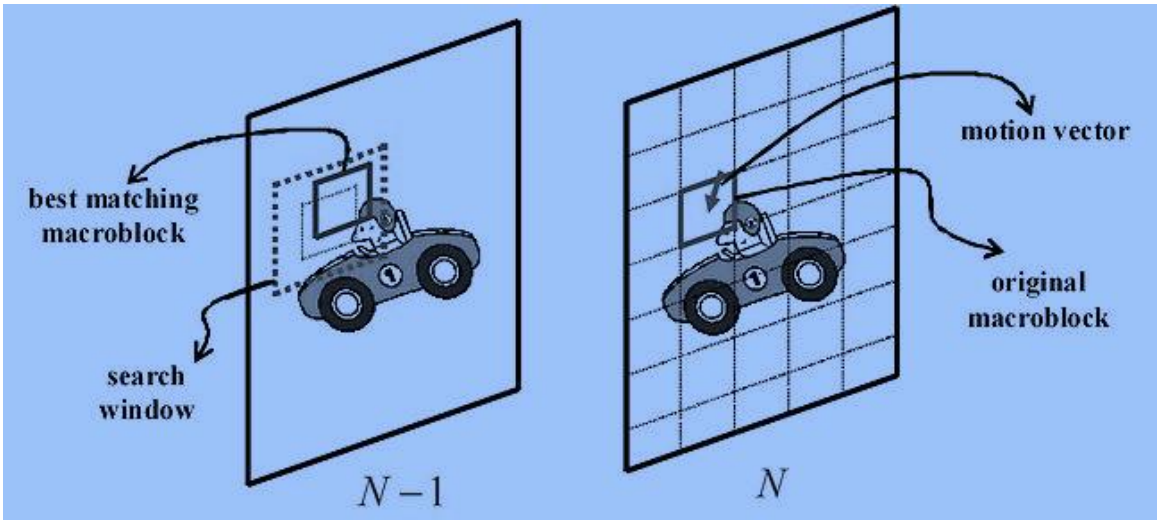


Figure 2.2 Block based motion compensation in H.264 [26]

A reference frame is used to eliminate redundancy, this frame is precisely coded with raw pixel values so it has the complete data stored. A reference frame, I-frame/P-frame can be a previous or a future frame from the video sequence. A block matching algorithm is used to find a similar block from the reference frame to the frame it is encoding. The probability of finding such a block is very high, but it may not be an accurate match and can introduce prediction error. The algorithm calculates the prediction error by taking the difference of the reference frame block and the block it is encoding. If the prediction error is higher than a threshold value, the algorithm looks for a different reference frame block with identical characteristics and calculates the prediction error. In case a matching block with least prediction error is found, the encoder transmits a vector - motion vector. It has co-ordinates that direct to the block from the reference frame. Therefore the encoder takes transform of the difference between reference and predicted frames, quantizes the transform coefficients, entropy coding is implemented to reduce the bitrate

There is a possibility that a similar block found from the reference frame presents large prediction error, which makes the complete size of motion vector and prediction error higher than raw encoded pixels of the block. The algorithm then sends data on the raw encoded block in these cases.

The H.264 standard defines 21 sets of capabilities; these are known as profiles which target specific classes of applications.

Profiles for non-scalable 2D video applications include the following:

Constrained Baseline Profile (CBP)

This profile is mainly used for low-cost applications, mobile applications and video conferencing. It relates to the subset of features that are common between the Baseline, Main, and High Profiles [26].

Baseline Profile (BP)

The baseline profile comprises of I and P-slices, CAVLC and some error resilience tools like arbitrary slice ordering (ASO), flexible macroblock ordering (FMO) and redundant slices (RS). This profile is primarily used for low-cost applications that require other data loss robustness, some mobile applications and video conferencing. The significance of this profile has diminished since the definition of the Constrained Baseline Profile in 2009. All Constrained Baseline Profile bitstreams are considered to be Baseline Profile bitstreams, since these profiles share the same profile identifier code value [26].

Main Profile (MP)

The main profile contains of I, P and B slices, context-adaptive variable-length coding (CAVLC) and context-adaptive binary arithmetic coding (CABAC) entropy coding. It does not include error resilience tools like FMO, ASO, RS and data partitioning (DP) or switching – I (SI) and switching – P (SP) slices. This profile is used for standard-definition digital TV broadcasts which use the MPEG-4 format defined in the digital video broadcasting standard. It is not used for high-definition television broadcasts, since the importance of this profile diminished when the High Profile was developed in 2004 for the same application [26].

Extended Profile (XP)

Extended profile is a superset of baseline profile. It adds B, SP and SI-slices and interlaces coding tools and further support error resilience tool set in the form of data partitioning (DP). Intended as the streaming video profile, this profile has relatively high compression capability and some extra tricks for robustness to data losses and server stream switching [26].

High Profile (HiP)

This is the primary profile for disc storage applications and broadcast, mainly for high-definition television applications [26].

Progressive High Profile (PHiP)

This profile is similar to the high profile, but without support of field coding features [26].

Constrained High Profile

This profile is similar to the Progressive High profile, but without support of B (bi-predictive) slices [26].

Some other profiles are [26]:

- High 10 Profile (Hi10P)
- High 4:2:2 Profile (Hi422P)
- High 4:4:4 Predictive Profile (Hi444PP)
- High 10 Intra Profile
- High 4:2:2 Intra Profile
- High 4:4:4 Intra Profile
- CAVLC 4:4:4 Intra Profile
- Scalable Baseline Profile
- Scalable Constrained Baseline Profile
- Scalable High Profile
- Scalable Constrained High Profile
- Scalable High Intra Profile
- Stereo High Profile
- Multiview High Profile

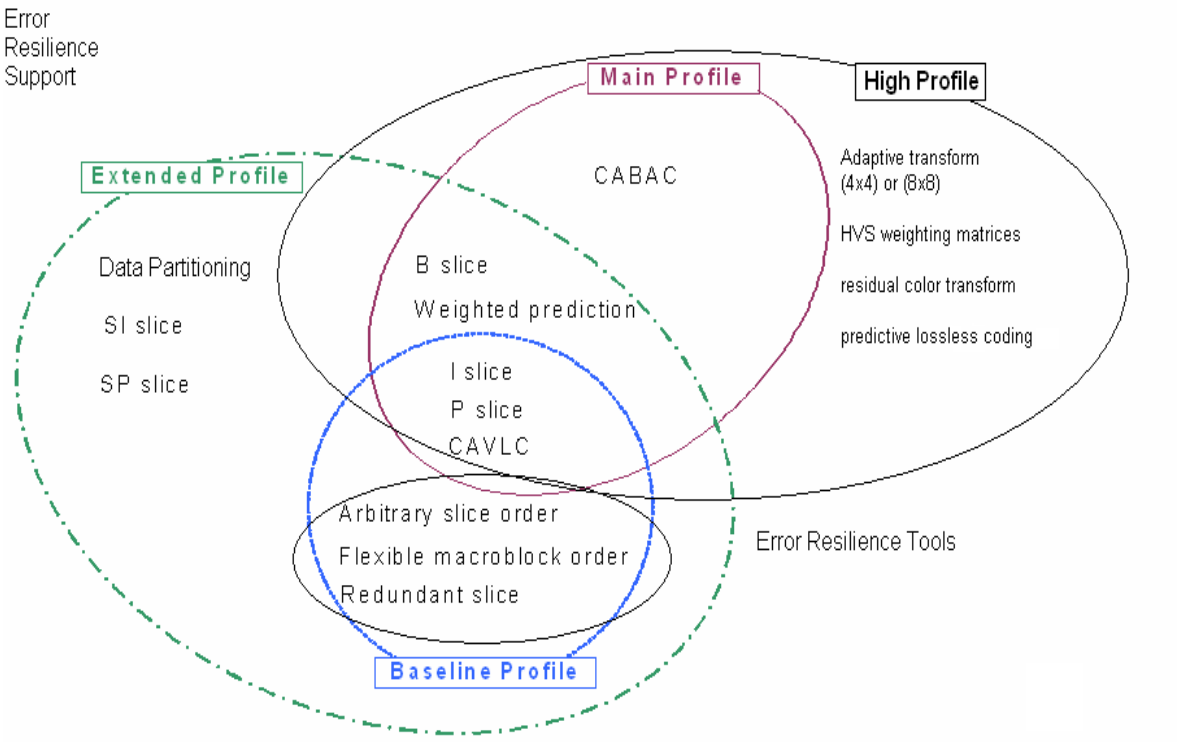


Figure 2.3 Profiles in H.264 with distribution of various coding tools [24].

These profiles include three enhancements of coding efficiency [27]:

- Adaptive MB level switching between 8x8 and 4x4 transform block sizes.
- Encoder specific perceptual based quantization scaling matrices.
- Encoder specified separate control of the quantization parameter for each chroma component.

All the high profiles support monochrome coded video sequences, in addition to typical 4:2:0 video as shown in figure 2.1. The difference in capability among these profiles is chiefly in terms of chroma formats and supported sample bit depths. Nevertheless, the high 4:4:4 profile also supports predictive lossless coding and residual color-transform features. [24]

H.264 Encoder:

H.264 encoder works on the same principles as that of any other codec. Figure 2.6 shows the block diagram of the H.264 encoder video codec.

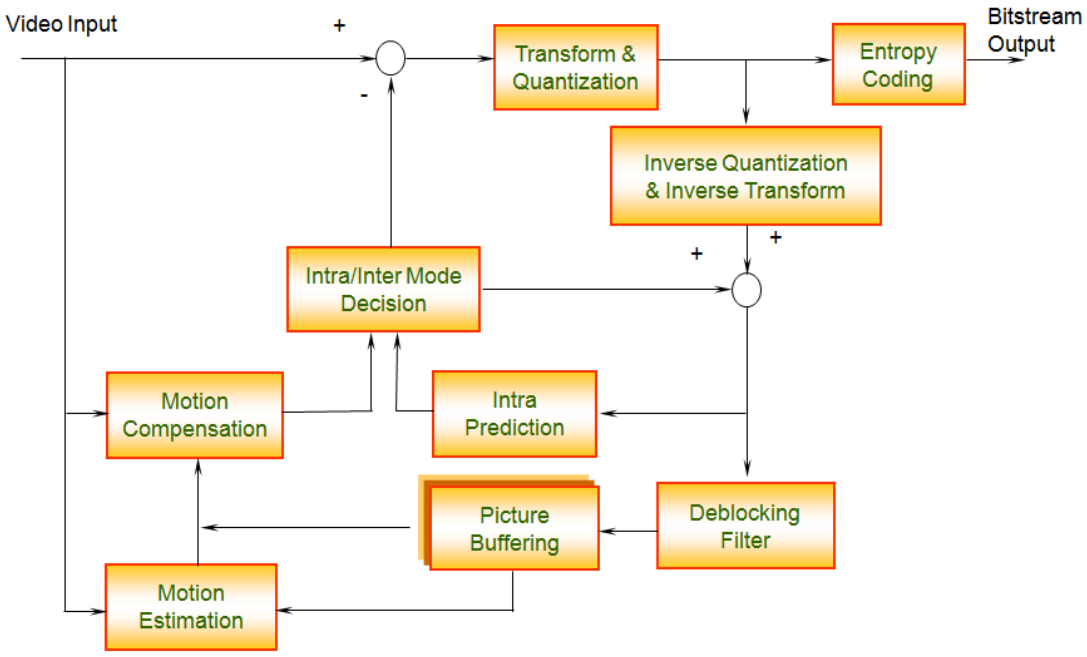


Figure 2.4 Block diagram of H.264 Encoder [4].

The H.264 encoder includes two dataflow paths: 1. Forward path. 2. Reconstruction path.

Encoder (Forward Path) [28]

As shown in figure 2.4 the input frame is processed in units of a MB. Each MB is encoded in either intra or inter mode. In any of the cases, a predicted MB 'P' is formed based on a reconstructed frame. It is formed from samples in the current frame that have been formerly encoded, decoded and reconstructed in intra mode. The samples which are unfiltered are used to form P. Whereas P is formed by motion-compensated prediction from one or more reference frame(s) in inter mode. The prediction for each MB may be formed from one or more future or past frames that have already been encoded and reconstructed [29]. A residual or difference MB is produced by taking the difference

of prediction P and current MB. This is transformed and quantized to produce a set of quantized transform coefficients. These coefficients are reorganized and entropy encoded. The entropy encoded coefficients along with side information required to decode the MB form the compressed bitstream is passed to a network abstraction layer (NAL) for either transmission or storage [28].

Decoder (Reconstruction path) [28]

As shown in Figure 2.4, the quantized MB coefficients are decoded in order to reconstruct a frame for encoding of future MBs. The coefficients are rescaled and inverse transformed to obtain a difference MB. This is not the same as the original difference MB. The quantization process introduces losses. The predicted MB P is added to the difference MB to form a reconstructed MB which is a distorted version of the original MB. A deblocking filter is applied to minimize the effects of blocking distortion and reconstructed reference frame is formed from a series of MBs [15].

Intra prediction:

Intra prediction uses the spatial correlation among pixels, there are three basic types defined as full MB prediction for 16x16 luma or the corresponding chroma size, 8x8 for luma prediction in FRExt [11] defined profiles, 4x4 luma prediction

In full MB prediction, the edge pixels of the previously decoded neighboring MBs are used to predict the pixel values of an entire MB of luma or chroma. Full MB intra prediction is used for luma in a MB type called the intra 16x16 intra MB type. Due to the differences in sizes for chroma arrays, the MB in different chroma sizes are used which are 8x8 chroma in 4:2:0 MBs, 8x16 chroma in 4:2:2 MBs and 16x16 chroma in 4:4:4 MBs. The prediction type for 16x16 MB is shown in Fig 2.5.

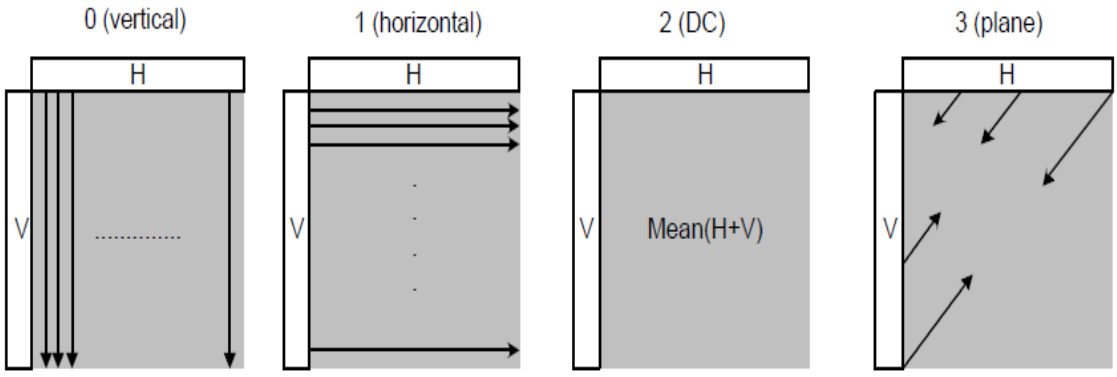
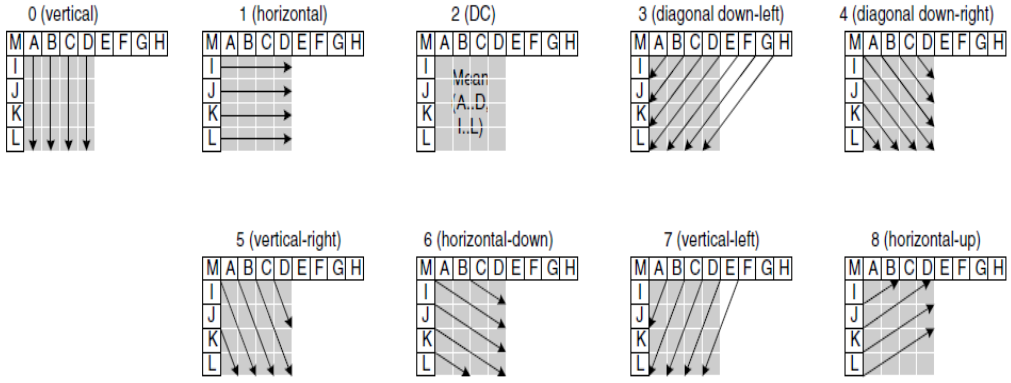


Figure 2.5 Intra prediction blocks for 16x16 luma MBs [15].

A full MB prediction can be achieved in one of four ways which could be used by the encoder to select the type of prediction [28]:

- Vertical: In vertical prediction the pixel values of a MB are predicted from the pixels just above the MB. Vertical mode is commonly known as the mode 0 for intra prediction.
- Horizontal: For horizontal prediction the pixel values of a MB are predicted from the pixels left to the MB. Horizontal mode is commonly known as the mode 1.
- DC: The luma values of the neighboring pixels are averaged and that average is used as predictor. DC is commonly known as the mode 2.
- Planar: In planar prediction, a three curve fitting equation is used to form a prediction block having a brightness, slope in the horizontal direction and slope in the vertical direction that approximately matches the neighboring pixels.



Pixels A to M are previously reconstructed and coded.

Figure 2.6 4x4 Luma intra prediction modes in H.264 [15].

The values of each 4x4 block of luma samples are predicted from the neighboring pixels above or left of a 4x4 block in spatial 4x4 prediction mode. The encoder has the choice to select from the nine differential directional ways of predicting for a 4x4 intra prediction block for luma as shown in Figure 2.6 [15]. The prediction direction relates to a certain set of spatially dependent linear combinations of formerly decoded samples for use as the prediction of each input sample.

1. The samples of the macroblock are predicted from the neighboring samples on the top in mode 0.

2. The samples of the macroblock are predicted from the neighboring samples from the left in mode 1.

3. The mean of all the neighboring samples is used for prediction in mode 2.

4. Mode 3 is in diagonally down-left direction.

5. Mode 4 is in diagonal down-right direction.

6. Mode 5 is in vertical-right direction.

7. Mode 6 is in horizontal-down direction.

8. Mode 7 is in vertical-left direction.

9. Mode 8 is in horizontal up direction.

The predicted samples are obtained from weighted average of the formerly predicted samples A to M. For 8x8 luma prediction the process is similar to that of 4x4 luma. The 8x8 luma prediction has the block size as 8x8 and uses a low-pass filter to enhance the prediction performance.

Inter Prediction [31]:

Inter prediction creates a prediction model from one or more previously encoded video frames. Inter-prediction is used to exploit the temporal redundancy in video data. The temporal correlation is reduced by inter prediction through the use of motion estimation and compensation algorithms [30]. An image is divided into MBs; each 16x16 MB is further partitioned into 16x16, 16x8, 8x16, 8x8 sized blocks. An 8x8 sub-MB can be further partitioned into 8x4, 4x8, 4x4 sized blocks. Figure 2.7 illustrates the partitioning of a MB and a sub-MB [30]. The input video characteristics administer the block size. A smaller block size ensures less residual data; however smaller block sizes also mean more motion vectors, and hence more bits are required to encode these motion vectors [30]

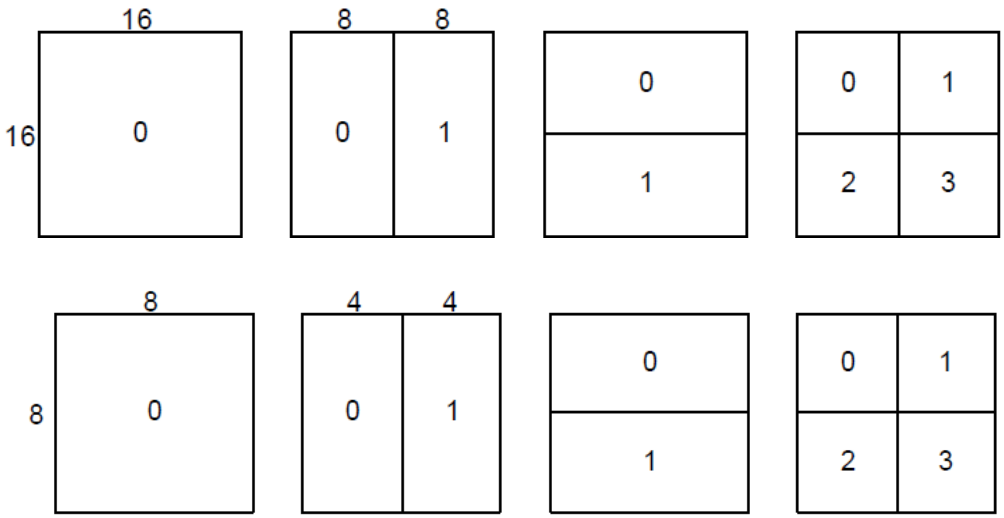


Figure 2.7 Macroblock partitioning in H.264 for inter prediction [30] row 1 (L-R) 16x16, 8x16, 16x8, 8x8 blocks and row 2 (L-R) 8x8, 4x8, 8x4, 4x4 blocks

Each partition in an inter-coded MB is predicted from an area of equal size in a reference picture. The offset between the two areas (the motion vector) has quarter-sample resolution for the luma component and one-eighth-sample resolution for the chroma components. The luma and chroma samples at sub-sample positions do not exist in the reference picture and so it is necessary to create them using interpolation from nearby coded samples [31]. Figure 2.8 shows half and quarter pixel interpolations used in luma pixel interpolation respectively. Six-tap filtering is used for derivation of half-pel luma sample predictions, for sharper sub pixel motion-compensation. Quarter-pixel motion is derived by linear interpolation of the half pel values, to conserve processing power.

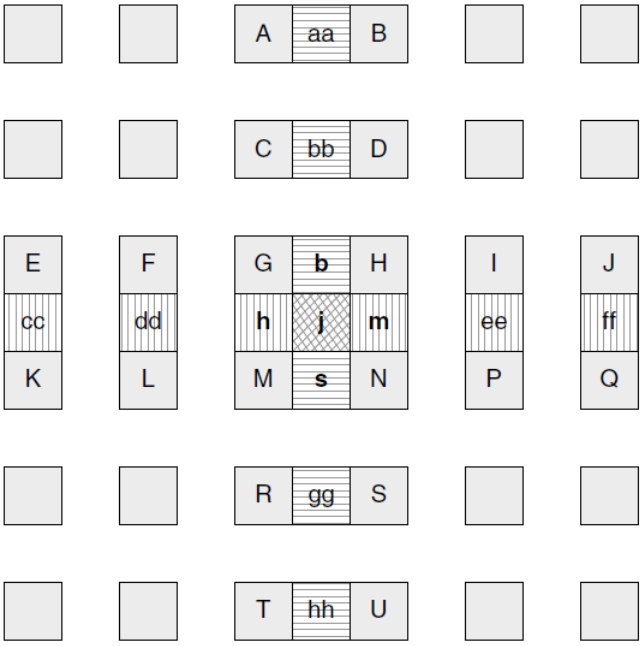


Figure 2.8 Interpolation of luma half-pel positions [30]

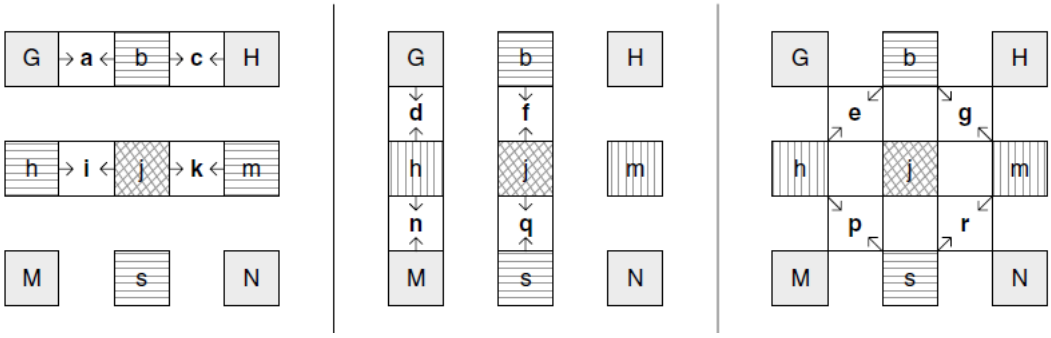


Figure 2.9 Interpolation of luma quarter-pel positions [30]

The reference pictures used for inter prediction are formerly decoded frames and are stored in the picture buffer. H.264 supports the use of multiple frames as reference frames. This is implemented by the use of an additional picture reference parameter which is transmitted along with the motion vector. Figure 2.10 illustrates an example with 4 reference pictures.

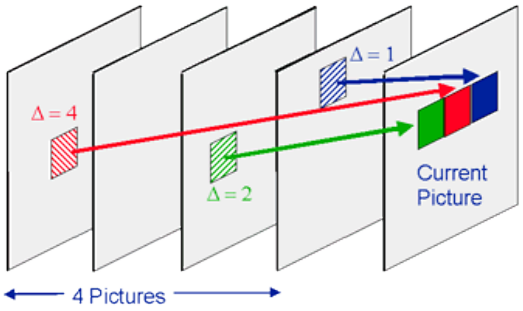


Figure 2.10 Motion compensated prediction with multiple reference frames [30]

Entropy Coding:

A coded stream or a file of H.264 consists of a series of coded symbols which make up the identifiers and delimiting codes, syntax and include parameters, prediction types, differentially coded motion vectors and transform coefficients. The H.264/AVC standard specifies several methods for coding the symbols i.e. converting each symbol into a binary pattern that is transmitted or stored as part of the bitstream. These methods are as follows [27]:

Fixed length code: It is a code in which a fixed number of source symbols are encoded into a fixed number of output symbols. A symbol is converted into a binary code with a specified length (n bits). In this method, particularly, data compression is only possible for large blocks of data, and any compression beyond the logarithm of the total number of possibilities comes with a finite probability of failure.

Exponential-Golomb variable length code [15]: The symbol is represented as an Exp-Golomb [4] codeword with a varying number of bits. In general, shorter Exp-Golomb codewords are assigned to symbols that occur more frequently.

CAVLC (Context adaptive variable length coding): Context adaptive variable length coding, a specifically designed method of coding transform coefficients in which different

sets of variable length codes are chosen depending on the statistics of recently-coded coefficients, using context adaptation [27].

After the processes such as prediction, transformation and quantization blocks are typically scarce, often containing only zeros. CAVLC uses run-level coding to efficiently represent strings of zeros. The number of non-zero coefficients in neighboring blocks is interrelated. The number of coefficients is encoded using a look-up table and choosing a look-up table depends on the number of non-zero coefficients in neighboring blocks.

CABAC (Context adaptive binary arithmetic coding): Context adaptive binary arithmetic coding [4] is a technique of arithmetic coding in which the probability models are updated based on previous coding statistics. CABAC is an optional entropy coding mode accessible in Main and High profiles. CABAC achieves good compression performance by [27]:

1. Selecting probability models for each syntax element according to the element's context.
2. Adapting probability estimates based on local statistics
3. Using arithmetic coding rather than variable-length coding.

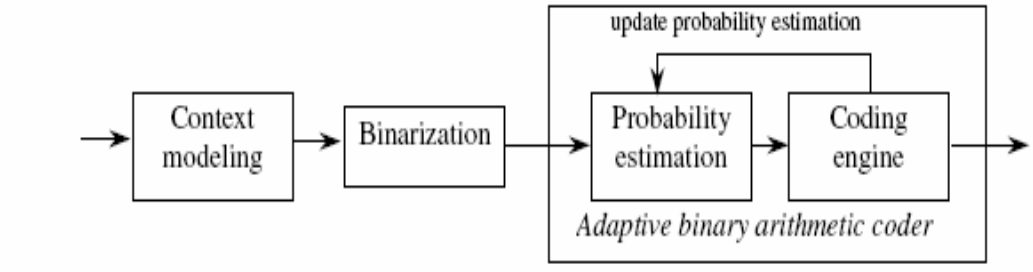


Figure 2.11 Block diagram for CABAC [15]

Figure 2.11 shows schematic for CABAC [15].Coding a data symbol involves the following stages [27]:

Binarization: CABAC uses binary arithmetic coding which means that only binary decisions (1 or 0) are encoded. A non-binary valued symbol is converted to a binary code prior to arithmetic coding. Context model selection: A “context model” is a probability model for one or more bits of the binarized symbol and is chosen from a selection of available models depending on the statistics of recently-coded data symbols.

Arithmetic encoding: An arithmetic coder encodes each bin according to the selected probability model. Note that there are just two sub-ranges values 1 or 0.

Probability update: The selected context model is updated based on the actual coded value.

H.264 Decoder:

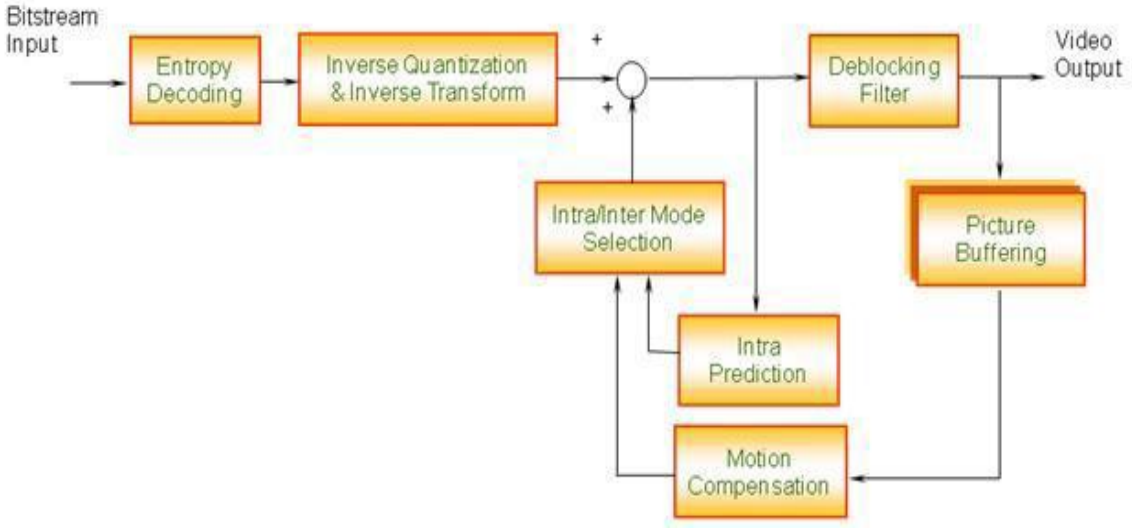


Figure 2.12 Block diagram of H.264/AVC video decoder [4].

The H.264/AVC decoder receives a compressed bitstream from the NAL as shown in Figure 2.12. The data elements are entropy decoded and reordered to produce a set of quantized coeffients. These are rescaled and inverse transformed to give a difference macroblock. Using the header information decoded from the bit stream, the decoder creates a prediction macroblock P, identical to the original prediction P formed in the encoder. P is added to the difference macroblock and this result is given to the deblocking filter to create the decoded macroblock.

The purpose of the reconstruction path in the encoder is to ensure that both encoder and decoder use identical reference frames to create the prediction P. If this is not the case, then the predictions P in encoder and decoder will not be identical, leading to an increasing error or drift between the encoder and decoder [29].

The key features that make H.264/AVC a highly efficient codec are [27] :

- Variable block size motion compensation with block sizes from 16x16 to 4x4, enabling precise segementation of moving regions.
- Six tap filtering for sharper subpixel motion compensation. Quarter-pixel motion is derived from linear interpolation.
- Weighted prediction , allowing encoder to specify the scaling and offset.
- Lossless MB coding
- An in-loop deblocking filter
- Loss resilence features like network abstraction layer (NAL), flexible MB ordering (FMO) , redundant slices (RS) and data partitioning (DP)
- An entropy coding design including context adaptive binary arithmetic coding (CABAC) , context adaptive variable length coding (CAVLC) and variable length coding (VLC)

- Switching slices like SI and SP slices.

Summary:

This chapter has covered the importance of video compression, H.264 video coding standard, encoder, decoder, and the coding tools in the standard. The next chapter covers the theoretical concept of prediction block filtering.

Chapter 3

Adaptive prediction block filtering

Many new coding tools have been developed in order to enhance the efficiency of H.264/AVC. Some of these tools are adopted in the Key Technical Areas (KTA) developed for preparing for the next generation video coding standard [2].The adaptive interpolation filter (AIF) [3], the adaptive loop filter (ALF) [4], and the high precision interpolation filter [5] have been introduced to provide a more precise reference picture [16]. Other efficient coding methods, such as the motion vector competition [6] and the extended macro-block (MB) [7], are also adopted in the KTA.

Both the AIF and ALF are designed based on the Wiener filter, which is an optimal filter to handle the degradation of image quality caused by additional noise and/or blurring [8]. The AIF is obtained by calculating the filter coefficients that make the reference picture closer to the original picture. Similarly, in the ALF, the loop filter coefficients are determined by improving the coding noise in the de-blocked picture [16].

Wiener Filter:

The goal of the Wiener filter is to filter out noise that has corrupted a signal. It is used to produce an estimate of a desired or target random process by filtering another random process through the filter. The Wiener filter minimizes the mean square error between the estimated random process and the desired process [37].

Wiener theory, formulated by Norbert Wiener [37], forms the foundation of data-dependent linear least square error filters. Wiener filters play a central role in a wide range of applications such as linear prediction, echo cancellation, signal restoration, channel equalization and system identification.

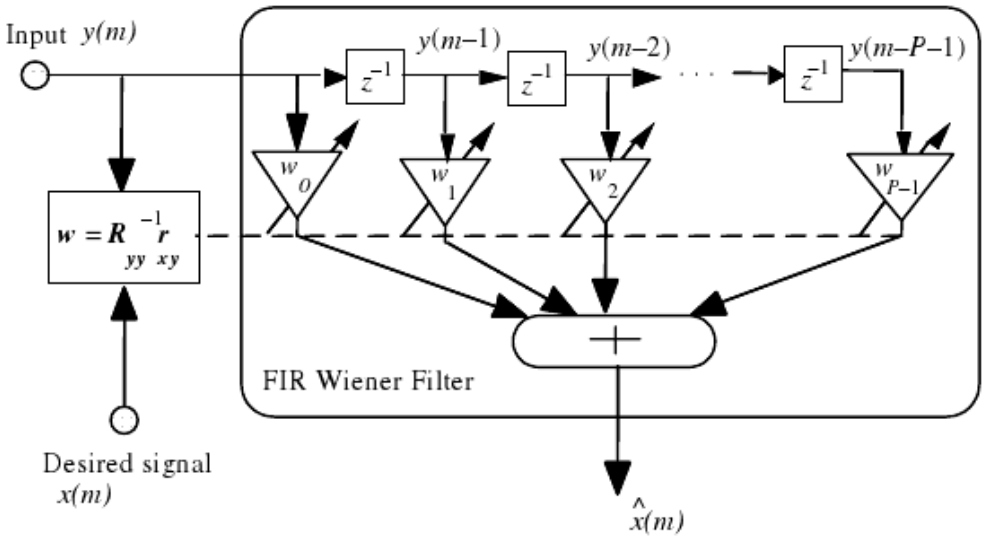


Figure 3.1 Illustration of a Wiener filter structure [38]

The coefficients of a Wiener filter are calculated to minimize the average squared difference between the filter output and a desired signal. In its basic form, the Wiener theory assumes that the signals are stationary processes. However, if the filter coefficients are periodically recalculated for every block of N signal samples then the filter adapts itself to the average characteristics of the signals within the blocks and becomes block-adaptive as in the case of current implementation of adaptive prediction block filter [38].

Adaptive Interpolation Filter [3], [18]:

In H.264/AVC, the resolution of motion vector is quarter-pixel, the reference frame is interpolated to be 16 times the size for MCP, 4 times both sides. As shown in Fig. 3.2(a), the interpolation defined in H.264 includes two stages, interpolating the half-pixel and quarter-pixel sub-positions, respectively. The interpolation in the first stage is separable, which means the sampling rate in one direction is doubled by inserting zero-

valued samples followed by filtering using a 1-D filter h1, [1, 0, -5, 0, 20, 32, 20, 0, -5, 0, 1]/32 [15], and then the process repeats in the other direction. The second stage, which is non-separable, uses bilinear filtering supported by the integer pixels and the interpolated half-pixel values.

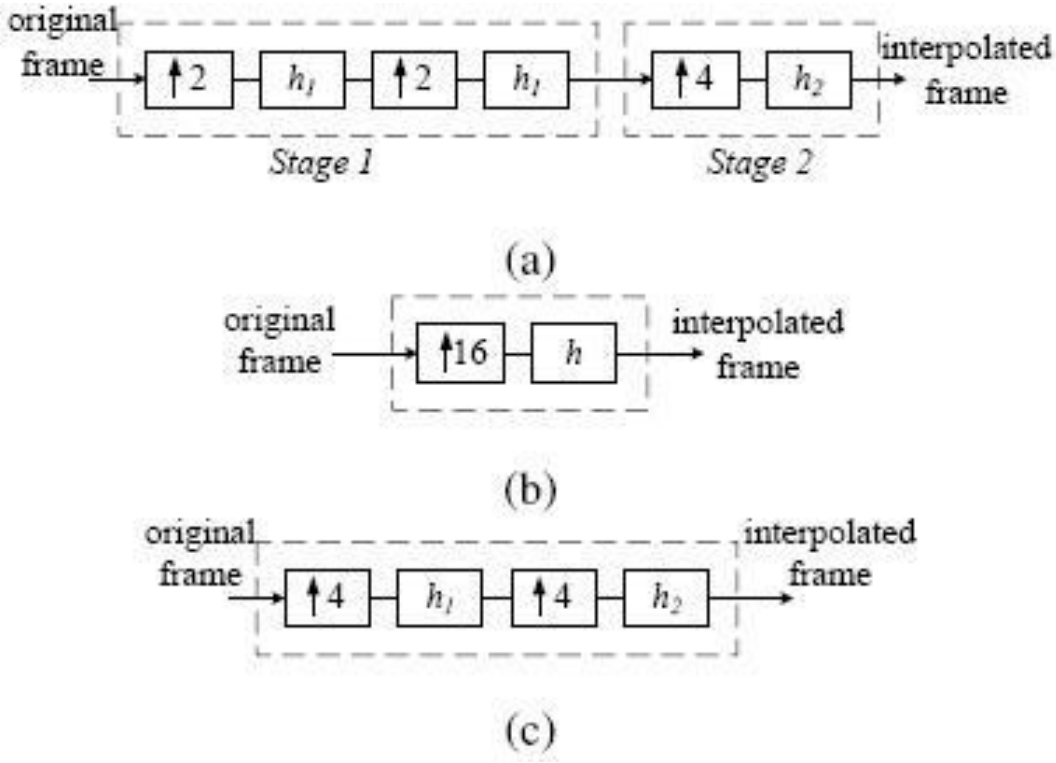


Figure. 3.2 Interpolation process of (a) the filter in H.264/AVC, (b) the optimal AIF, and (c) the separable AIF [18]

where $\boxed{\uparrow m}$ = m times the sampling rate due to interpolation.

To reduce the bit-rate of video signals, the international telecommunication union (ITU) coding standards [32] apply hybrid video coding with motion-compensated

prediction combined with transform coding of the prediction error. In the first step the motion- compensated prediction is performed. The temporal redundancy, i.e., the correlation between consecutive images is exploited for the prediction of the current image from already transmitted images. In a second step, the residual error is transform coded, thus the spatial redundancy is reduced.

For performing motion-compensated prediction, the current image of a sequence is split into blocks. For each block a displacement vector $\vec{d}_i$ is estimated and transmitted that refers to the corresponding position of its image signal in an already transmitted reference image. The displacement vectors have fractional-pel resolution. The H.264/AVC [1] is based on ¼ pel displacement resolution [33]. Displacement vectors with fractional resolution may refer to positions in the reference image, which are located between the sampled positions. In order to estimate and compensate the fractional-pel displacements, the reference image has to be interpolated on the fractional-pel positions.

H.264/AVC [1] uses a 6-tap Wiener interpolation filter with filter coefficients $(1, -5, 20, 20, -5, 1)/32$. The interpolation process is depicted in Figure 3.3 and can be subdivided into two steps. At first, the half-pel positions aa, bb, b, hh, ii, jj and cc, dd, h, ee, ff, gg are calculated, using a horizontal or vertical 6-tap Wiener filter, respectively. Using the same Wiener filter applied at fractional-pel positions aa, bb, b, hh, ii, jj the fractional-pel position j is computed. In the second step, the remaining quarter-pel positions are obtained, using a bilinear filter, applied at already calculated half-pel positions and existing full-pel positions.

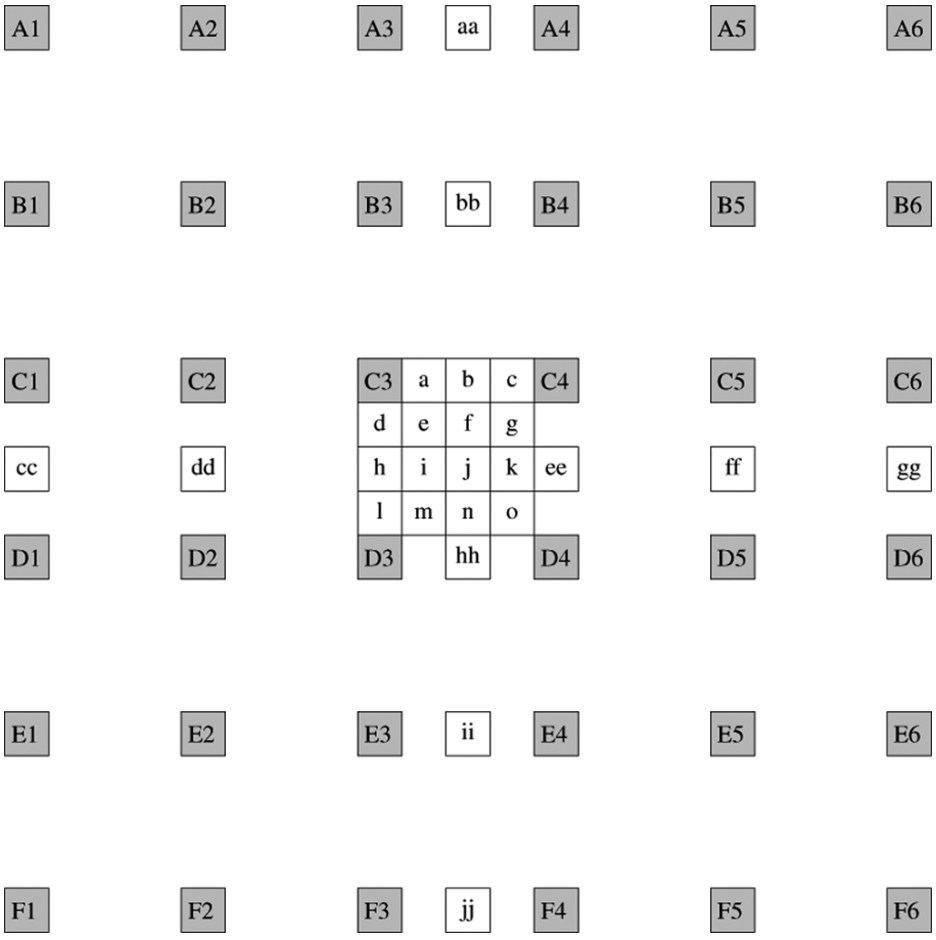


Figure 3.3 Integer pixels (shaded blocks with upper-case letters) and fractional pixel positions (non-shaded blocks with lower-case letters). Example for filter size 6 x 6. [3]

An adaptive interpolation filter as proposed in [3] is independently estimated for every image. This approach enables to take into account the alteration of image signal properties as aliasing on the basis of minimization of the prediction error energy. Analytical calculation of optimal filter coefficients is not possible due to nonlinearity, which is caused by subsequent application of 1-D filters. In [34] a 3-D filter is proposed. In this

proposal two techniques are combined: a 2-D spatial filter with a Motion Compensated Interpolation Filter (MCIF).

The main disadvantage of MCIF is the sensitivity concerning displacement vector estimation errors. Besides aliasing, there are further distorting factors, which impair the efficiency of motion compensated prediction. The main disadvantage of using a 2-D spatial filter with a motion compensated interpolation filter (MCIF) proposed in [34] is its numerical approach to determine the coefficients of a separable 2-D filter. Due to an iterative procedure, this method is nondeterministic in terms of time and requires a significantly higher encoder complexity.

In order to guarantee a limited increase of encoder complexity compared to the standard H.264/ AVC [1] on the one hand and to reach the theoretical bound for the coding gain obtained by means of a 2-D filter on the other hand, a non-separable filter scheme is proposed. An individual filter will be used for the interpolation of each fractional-pel position.

Adaptive Loop Filter [19]

There are three types of ALF: frame-based, block-based and quadtree-based ALFs. All of them are based on wiener filter, but with different filtering control basis. [19].

Wiener filter is capable of restoring the reconstructed picture to the original picture globally but some pixels are degraded locally. As the degraded area reduces the filtering efficiency, the means of picture restoration and loop filtering are enhanced if these areas are not filtered. Hence, the block-based ALF uses explicit flags for filtering on-off on block by block basis, whereas quadtree-based ALF introduces a quadtree data structure to carry out the variable-size block filtering [19].

Block-based Adaptive Loop Filter [19]

Figure 3.4 shows the block diagram of an encoder with block-based ALF. It applies a filter to luminance blocks, and signals a flag for each luminance block to indicate whether the block is filtered or not. Chrominance pixels are also filtered by filter coefficients designed individually for luminance.

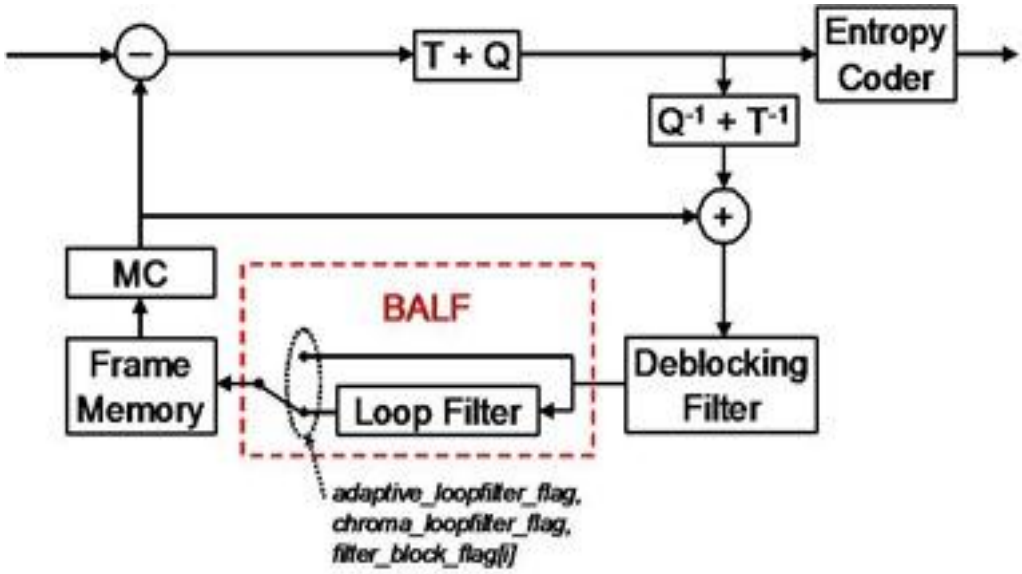


Figure 3.4 Block diagram of encoder with BALF [19]

In frame-based ALF, the tap length of Wiener filter is fixed. While, the optimal tap length depends on the characteristics of picture. Adaptive selection of tap length is selected slice by slice from 5×5, 7×7 or 9×9 taps for luminance of referenced pictures. However, only 5×5 tap filter is applied to reduce the overhead and complexity due to the inadequate enhancement of ALF in luminance of non-referenced pictures or chrominance of all pictures. The filter coefficients are point symmetric, as shown in figure 3.5

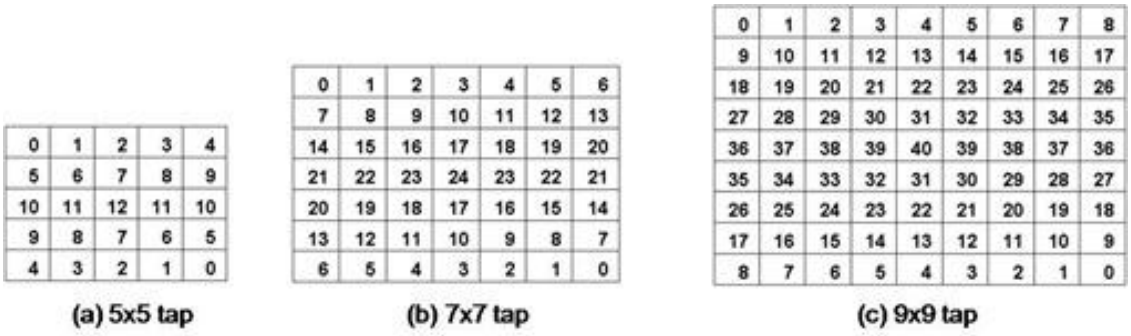


Figure 3.5 Point symmetric filters in raster scan order [19]

A filter designed for luminance can be applied to luminance of the reconstructed picture. A flag to indicate whether the luminance block is filtered or not is signaled for each luminance block. The luminance block size, which can be 8×8, 16×16, 24×24, 32×32, 48×48, 64×64, 96×96, or 128×128, is signaled for each frame [19].

A filter designed for chrominance can be applied to chrominance of the decoded picture only if a luminance filter is applicable. It is signaled for each frame whether the chrominance filter is applied to only C_b, only C_r, or both C_b and C_r.

Quadtree-based Adaptive Loop Filter [19]

Quadtree-based ALF further improved the filtering efficiency by using more flexible filtering control scheme – quadtree data structure. In this structure, as shown in Figure 3.7, each leaf indicates a block and each node has four branches. The information represented in the quadtree data structure are: the block partition flag indicated by circle, and block filtering flag, indicated by the diamond. For a block partition, each leaf is coded as “0”, and each node is coded as “1”. Only a leaf has a filter block flag to indicate whether the block is filtered or not. In order to reduce the redundancy, no block partition flag is coded at the bottom layer. Figure 3.5 depicts the block diagram of codec with QALF, which is similar to the BALF.

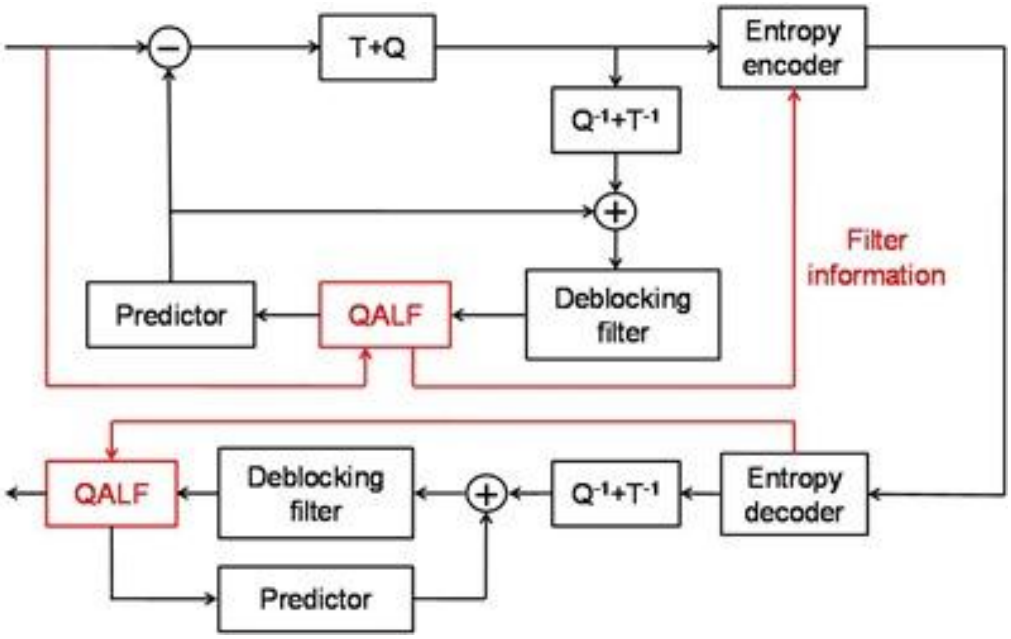


Figure 3.6. Block diagram of codec with QALF [19]

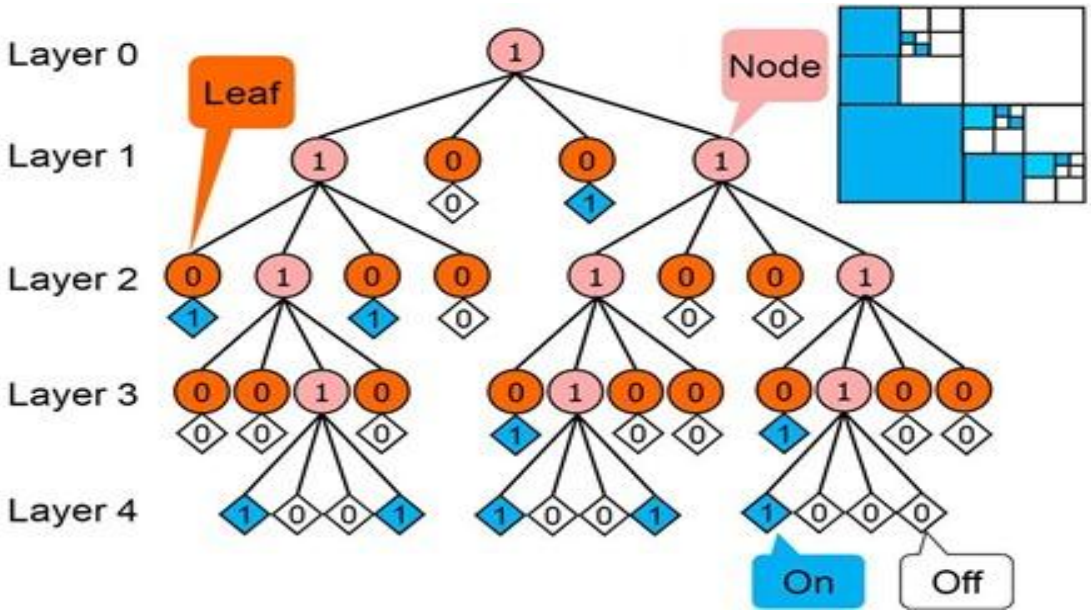


Figure 3.7. Quadtree representation in QALF [19]

In order to find the optimal quadtree data structure, a conventional bottom-up recursive algorithm is used in QALF. Suppose that we have already known the optimal quadtree data structures for layer l+1, the four branches should be combined into a leaf when J(l)<J(l+1), as shown in Figure 3.8. Using this algorithm, the quadtree data structure is decided [19].

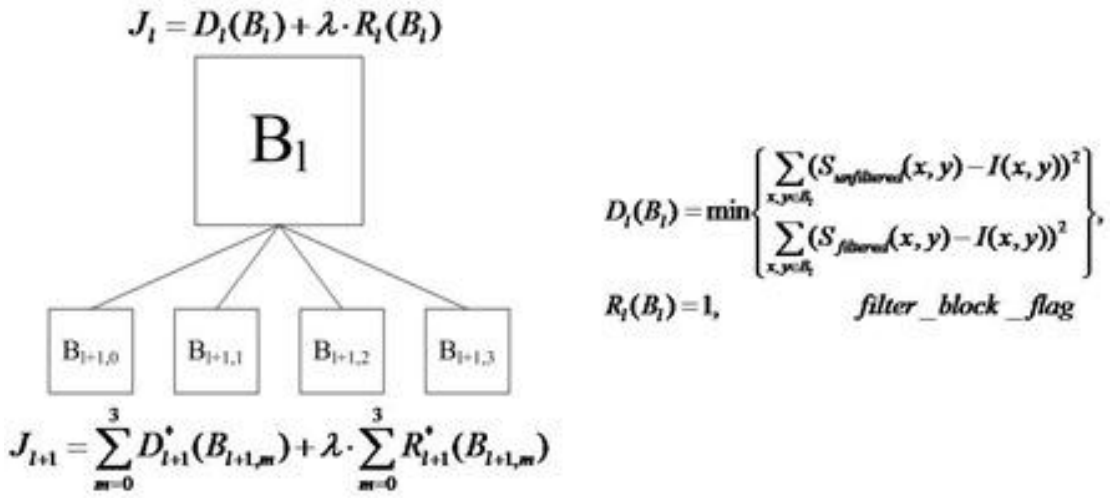


Figure 3.8 Bottom-up recursive algorithm [19]

Algorithm Description of Block/Quadtree-based ALF [19]

Here, the algorithms of Block/Quadtree-based ALF are summarized as follows:

Step 1: Filter tap size is set to 5×5, and the Wiener-Hopf [19] equation is used to find the optimal 5×5 filter coefficients for the whole picture

Step 2: The decided 5×5 filter in step 1 is used in the block/quadtree structure optimization procedure:

BALF: Block Size (8, 16, 24, 32, 48, 64, 96 and 128) and filter block flag are selected by rate-distortion optimization algorithm

QALF: Quadtree data structure (e.g. minimum block size and number of layers) and filter block flag are optimized by using bottom-up recursive algorithm

Step 3: Filter tap size (5×5, 7×7 or 9×9) and corresponding filter coefficients are further optimized by using decided block/quadtree data structure

Adaptive prediction block filter [16]

In order to improve the coding efficiency of H.264/AVC, an adaptive prediction block filter (APBF) based on Wiener filter is implemented on every Sub Block (SB) of a macro-block (MB) where each MB is decomposed into 4x4 SBs. For each SB the filter coefficients are calculated using the prediction and reconstruction results of the neighboring SBs. The proposed filter is applied to the prediction block of the current SB [16], and the filtered block is selectively used depending on the Rate Distortion (RD) cost [9]. For each SB, if the APBF is used, by reducing the number of bits required for encoding the residual signal between the prediction and original signal of the SB, the coding efficiency is improved. Additionally, as the same filter coefficients can be obtained in the decoder, they do not need to be encoded into the bit-stream [16]. The proposed method achieves a 5.04% bitrate savings on an average when compared to H.264/AVC.

The proposed method adopts the Wiener filter [16] to transform the prediction signal to almost match the original signal. By exploiting the original and predicted pixel values of the current SB, the Wiener filter coefficients can be calculated. But in this case, encoding the filter coefficients is required for each SB, since the original pixel values are not available in the decoder. Information available at both the encoder and the decoder should be utilized to avoid encoding the filter coefficients. Therefore, the formerly reconstructed neighboring SBs of the current SB are used to calculate the APBF coefficients. If the Wiener filter obtained by using the neighboring SBs is applied to the

current prediction signal, the filtered signal can be very similar to the original signal producing less prediction errors [16].

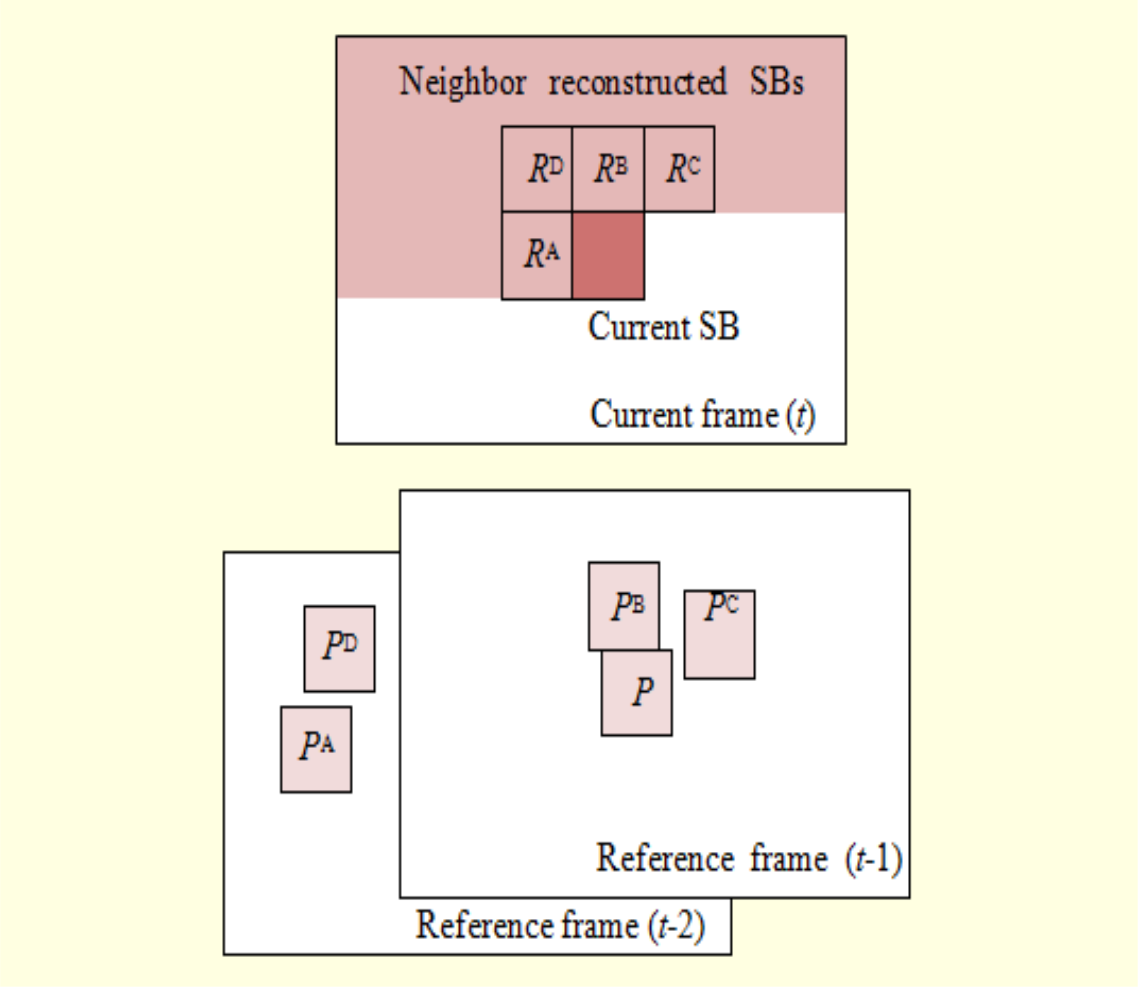


Figure 3.9. Neighbor reconstructed SBs $\{R^A, R^B, R^C, R^D\}$ and their corresponding prediction SBs $\{P^A, P^B, P^C, P^D\}$ [16]

The filter coefficients for the current SB are calculated by using four pairs composed of a prediction SB from $\{P^A, P^B, P^C, P^D\}$ and a reconstructed SB from $\{R^A, R^B, R^C, R^D\}$ in the neighborhood as shown in Figure 3.9 or by using each one of them. The Wiener filter coefficients are calculated by minimizing the mean square error

between the prediction and reconstruction signal of the neighboring SBs. For example, a set of filter coefficients using P^{A} and R^{A} is calculated as

$$C_{\mathrm{A}} = \underset{\{c_{i,j}^{\mathrm{A}} | -N \le i,j \le N\}}{\arg\min} E\left[\left(R_{x,y}^{\mathrm{A}} - \sum_{i,j=-N}^{N} c_{i,j}^{\mathrm{A}} \cdot P_{x+i,y+j}^{\mathrm{A}}\right)^2\right]$$

Where C_{A}, is a (2N+1)×(2N+1) symmetric filter consisting of filter coefficients $\mathrm{C}_{\mathrm{i,j}}^{\mathrm{A}}$, $\mathrm{P}_{\mathrm{x,y}}^{\mathrm{A}}$ and $\mathrm{R}_{\mathrm{x,y}}^{\mathrm{A}}$ are the predicted and reconstructed pixel values at the (x, y) position in the neighboring SB, respectively. LMS scheme is used to obtain C_{A}. N pixels are padded at the boundary of the SB for calculation. In this APBF scheme, the center symmetric filter [10] is employed. Thus, the positions located symmetrically from the center point have the same coefficient. Same filter coefficients can be derived at the decoder. Figure 3.8 shows the flowchart of the APBF scheme at the encoder [16].

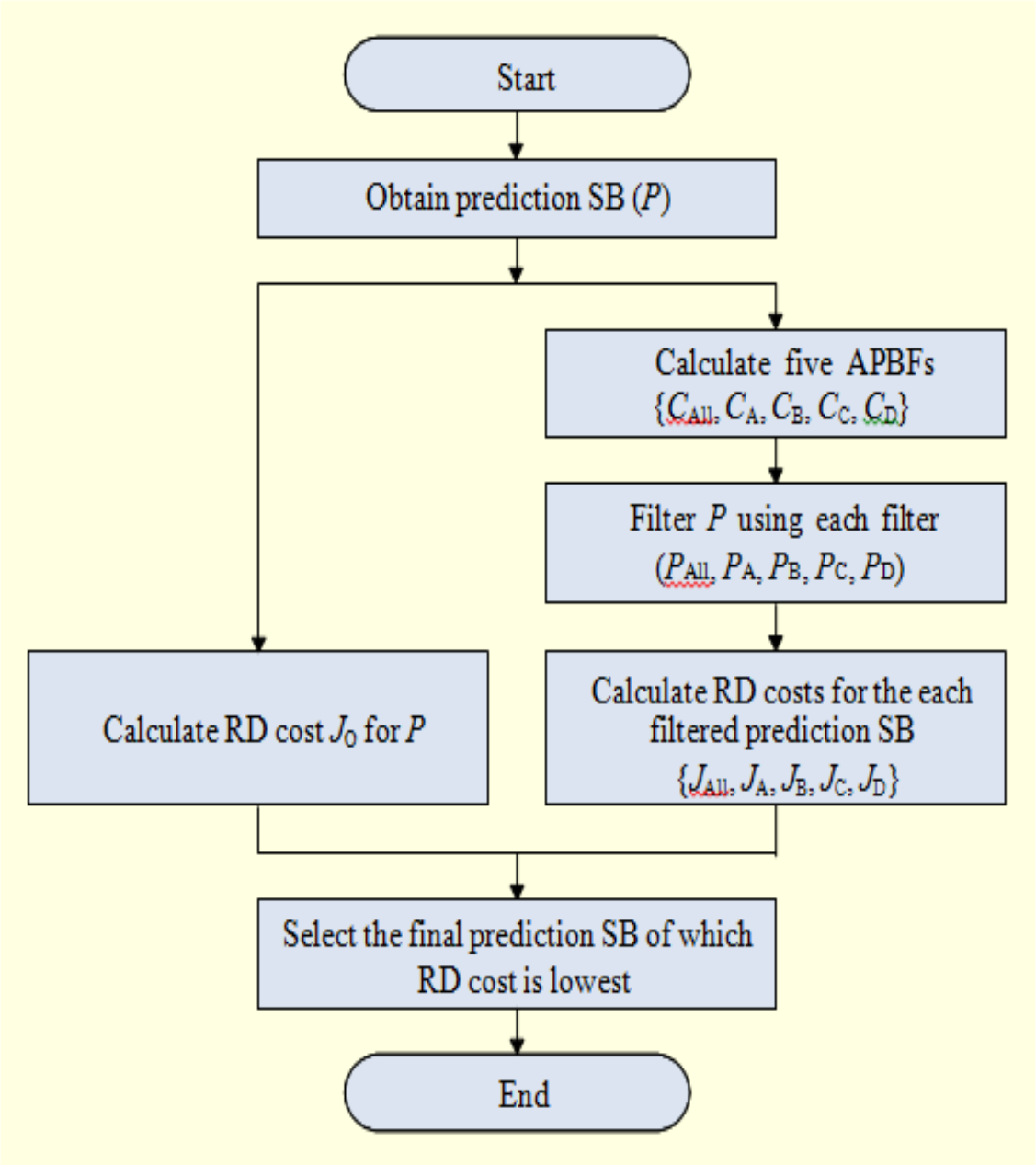


Figure 3.10. Flowchart of ABPF scheme [16]

- Step 1. Obtain the prediction SB, P, of the current SB [16].
- Step 2. Compute five sets of the Wiener filter coefficients, for P using the predicted and reconstructed pixel values of the neighboring SBs [16].
- Step 3. Process P by utilizing the filters in step 2 to obtain {P'_{All},P'_{A} , P'_{B}, P'_{C}, P'_{D}} [16].
- Step 4. Calculate the RD costs [9][18] {J_{All}, J_{A} , J_{B}, J_{C}, J_{D} , J_{O} } for P'_{All}, P'_{A} , P'_{B}, P'_{C}, P'_{D} and P, respectively [16].
- Step 5. Select the final prediction SB that yields the minimum RD cost. Then, the residue between the final prediction SB and the original SB are coded [16].

According to the selected prediction SB, an index $\mathrm{I_{ABPF}}$ is transmitted to the decoder to notify the usage of APBF. The best mode with the lowest RD [9] cost is selected after this process is tested for all prediction modes. At the decoder side, the same prediction signal used at the encoder side can be derived. The Wiener filter is constructed, and then the prediction SB is filtered concurring to the decoded value of $\mathrm{I_{ABPF}}$[16].

Summary

The theoretical aspects of adaptive prediction block filtering implemented on sub-blocks are explained along with the details of other filters such as adaptive interpolation filter and adaptive loop filter in this chapter. The next chapter will be a discussion on the analysis of implementing the APBF.

Chapter 4

Results of adaptive prediction block filtering on sub-blocks

The adaptive prediction block filter scheme is implemented using JM11KTA2.3 software [35]. For each sequence, 100 frames are encoded with IPPP prediction structure based on the conditions in [11] except for RDOQ. The test platform is an Intel Core 2 Quad Q8400 2.66-GHz CPU and 8-GB RAM with Windows 7 64-bit operating system. Although the intra-modes are also utilized for inter-frames, the probability that the intra-mode is selected as a best mode is quite low as shown in table 4.1.

Table 4.1 Selection ratio of intra-modes in P frame. Frames for 1 second are coded by the original H.264/AVC standard [16].

Sequence	Size	Frame rate (fps)	Selection ratio (%)
Kimono	1920 × 1080 (1080p)	24	9
ParkScene		24	3.36
Cactus		50	7.06
RaceHorses	832 × 480 (WVGA)	30	12.33
BasketballDrill		50	6.88
BQMall		60	2.91
RaceHorses	416 × 240 (WQVGA)	30	5.07
BasketballPass		50	0.44
BlowingBubbles		50	8.76

Hence, applying APBF to the intra-block does not considerably affect the compression performance, but rather increases the computational complexity. Therefore, the APBF scheme is performed only for the inter-prediction modes.

Quality Assessment Metrics

In lossy compression techniques, two aspects need evaluation – the type and amount of degradation induced in the reconstructed image. The objective of image quality evaluation is to measure the difference between the original and reconstructed images with great precision. The result obtained is used to design the finest video codecs. The objective quality measure like PSNR, measures the difference between the individual image pixels of original and reconstructed images.

$$MSE = \frac{1}{M * N} \sum_{m=1}^{M} \sum_{n=1}^{N} [x(m,n) - y(m,n)]^2$$

$$PSNR = 10 \log_{10} \frac{L^2}{MSE}$$

PSNR is calculated as shown in equation. x is the original image and y is the reconstructed image. M and N are the width and height of the image. L is the maximum pixel value in NxM pixel image. PSNR values generally are in 20dB to 40 dB range. Bjøntegaard Delta (BD) PSNR is calculated to compare the compression performance, and increasing percentages of the elapsed time at the encoder and decoder, ΔT_{Enc} and ΔT_{Dec}, are calculated for measuring the computational complexity.

(BD) PSNR = PSNR (dB) with ABPF in H.264 – PSNR (dB) without ABPF in H.264

T = CfunctionFprocessor

where T is the actual time required to execute each function, measured in seconds. Cfunction is the number of CPU cycles needed for each function, and Fprocessor is processor's speed measured in MHz.

$$\Delta T_{Enc} = \frac{(\text{T at encoder with APBF in H.264}) - (\text{T at encoder without APBF in H.264})}{(\text{T at encoder without APBF in H.264})} \times 100$$

$$\Delta T_{Dec} = \frac{(\text{T at decoder with APBF in H.264}) - (\text{T at decoder without APBF in H.264})}{(\text{T at decoder without APBF in H.264})} \times 100$$

Bitrate can also indicate the quality of a video file. A video file that is compressed at 3000 Kbps will look better than the same file compressed at 1000 Kbps. Just like the quality of an image is measured in resolution, the quality of a video file is measured by the bitrate. Bitrate often refers to the number of bits used per unit of playback time to represent a continuous medium data compression. The encoding bit rate of a multimedia file is the size of a multimedia file in bytes divided by the playback time of the recording (in seconds), multiplied by eight [36].

$$\text{Bitrate} = \left(\frac{\text{File size (bytes)}}{\text{Playback time(secs)}} \times 8 \right) \text{ bits/sec}$$

$$\text{BD-bitrate (\%)} = \frac{(\text{bitrate with APBF in H.264}) - (\text{bitrate without APBF in H.264})}{(\text{bitrate without APBF in H.264})} \times 100$$

3x3 adaptive prediction block filtering:

Table 4.2 shows the compression performance and the computational complexity of the algorithm for 3×3 APBFs. Figure 4.1. represents the difference in PSNR (dB) between H.264 without APBF and with 3x3 APBF. Figure 4.2. represents the difference in bitrate (%) between H.264 without APBF and with 3x3 APBF.

Table 4.2 Experimental results of 3x3 APBF scheme compared to H.264/AVC

Sequence	Size	H.264/AVC + 3 × 3 APBF			
		BD-PSNR (dB)	BD- bitrate (%)	ΔT_{Enc} (%)	ΔT_{Dec} (%)
Kimono	1920 × 1080 (1080p)	0.21	-4.75	65.78	84.65
ParkScene		0.16	-3.58	66.84	60.12
Cactus		0.19	-6.64	65.53	111.64
RaceHorses	832 × 480 (WVGA)	0.15	-3.12	63.51	97.23
BasketballDrill		0.38	-9.85	65.9	115.96
BQMall		0.29	-6.44	63.26	94.57
RaceHorses	416 × 240 (WQVGA)	0.13	-2.37	62.43	84.96
BasketballPass		0.22	-4.68	69.75	72.53
BlowingBubbles		0.18	-3.96	63.81	94.94
Average on 1080p seq.		0.187	-4.99	66.05	85.47
Average on WVGA seq.		0.273	-6.47	64.22	102.587
Average on WQVGA seq.		0.177	-3.67	65.33	84.143
Average on overall		0.212	-5.043	65.201	90.73

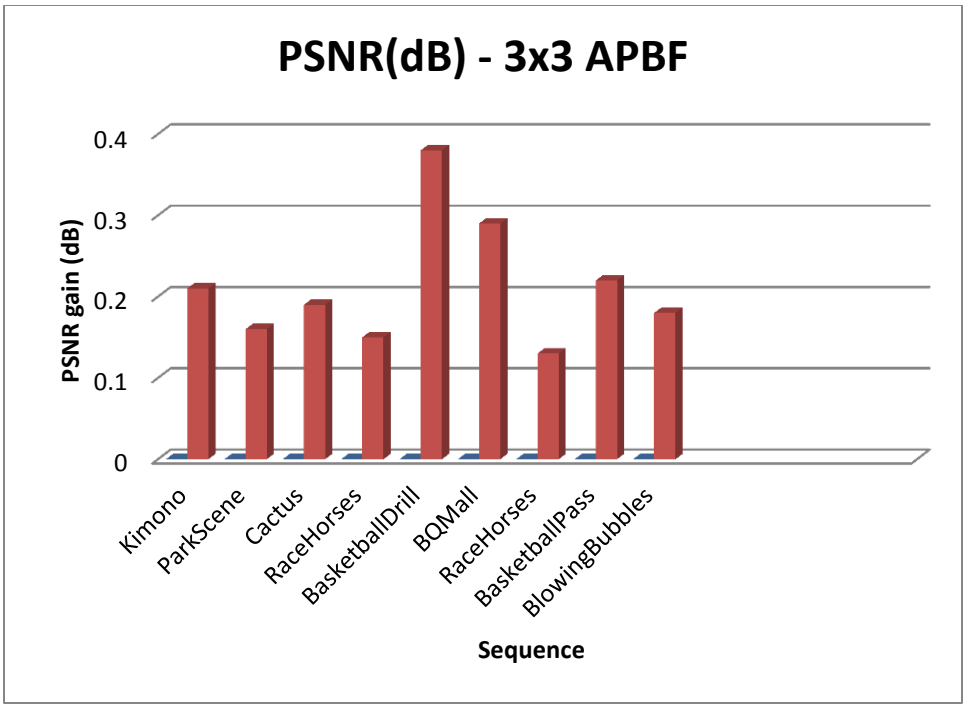


Figure 4.1 This plot shows the PSNR difference values for 3x3 APBF

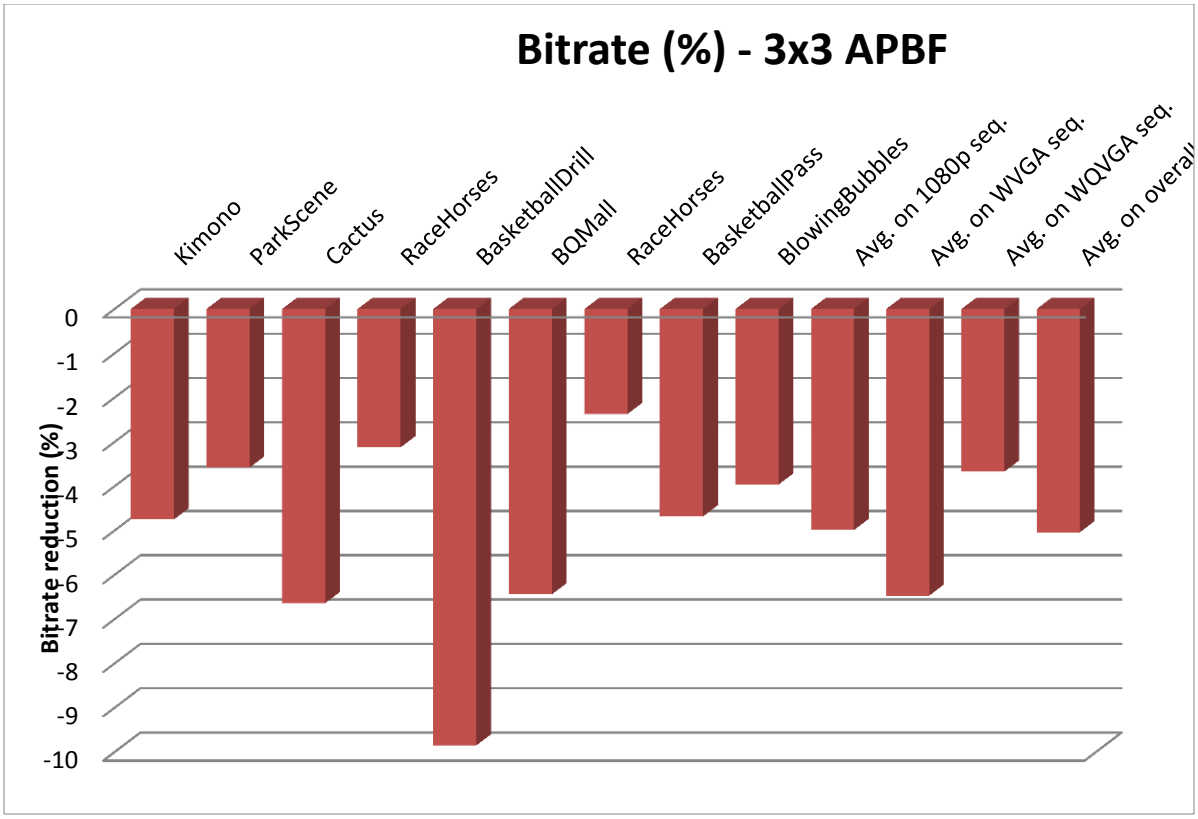


Figure 4.2 This plot shows bitrate difference (%) values for 3x3 APBF

5x5 adaptive prediction block filtering:

Table 4.3 shows the compression performance and the computational complexity of the algorithm for 5×5 APBFs. Figure 4.3. represents the difference in PSNR (dB) between H.264 without APBF and with 5x5 APBF. Figure 4.4. represents the difference in bitrate (%) between H.264 without APBF and with 5x5 APBF.

Table 4.3 Experimental results of 5x5 APBF scheme compared to H.264/AVC

Sequence	Size	H.264/AVC + 5 × 5 APBF			
		BD-PSNR (dB)	BD- bitrate (%)	ΔT_{Enc} (%)	ΔT_{Dec} (%)
Kimono	1920 × 1080 (1080p)	0.18	-4.56	95.48	197.47
ParkScene		0.15	-3.71	96.06	166.19
Cactus		0.17	-5.72	95.3	278.44
RaceHorses	832 × 480 (WVGA)	0.13	-2.67	89.29	242.35
BasketballDrill		0.38	-8.79	95.58	299.5
BQMall		0.28	-5.92	93.48	241.4
RaceHorses	416 × 240 (WQVGA)	0.1	-1.73	84.79	208.78
BasketballPass		0.21	-3.91	94.94	188.08
BlowingBubbles		0.17	-3.87	86.32	249.7
Average on 1080p seq.		0.167	-4.663	95.613	214.033
Average on WVGA seq.		0.263	-5.793	92.783	261.083
Average on WQVGA seq.		0.16	-3.17	88.683	215.52
Average on overall		0.197	-4.542	92.36	230.212

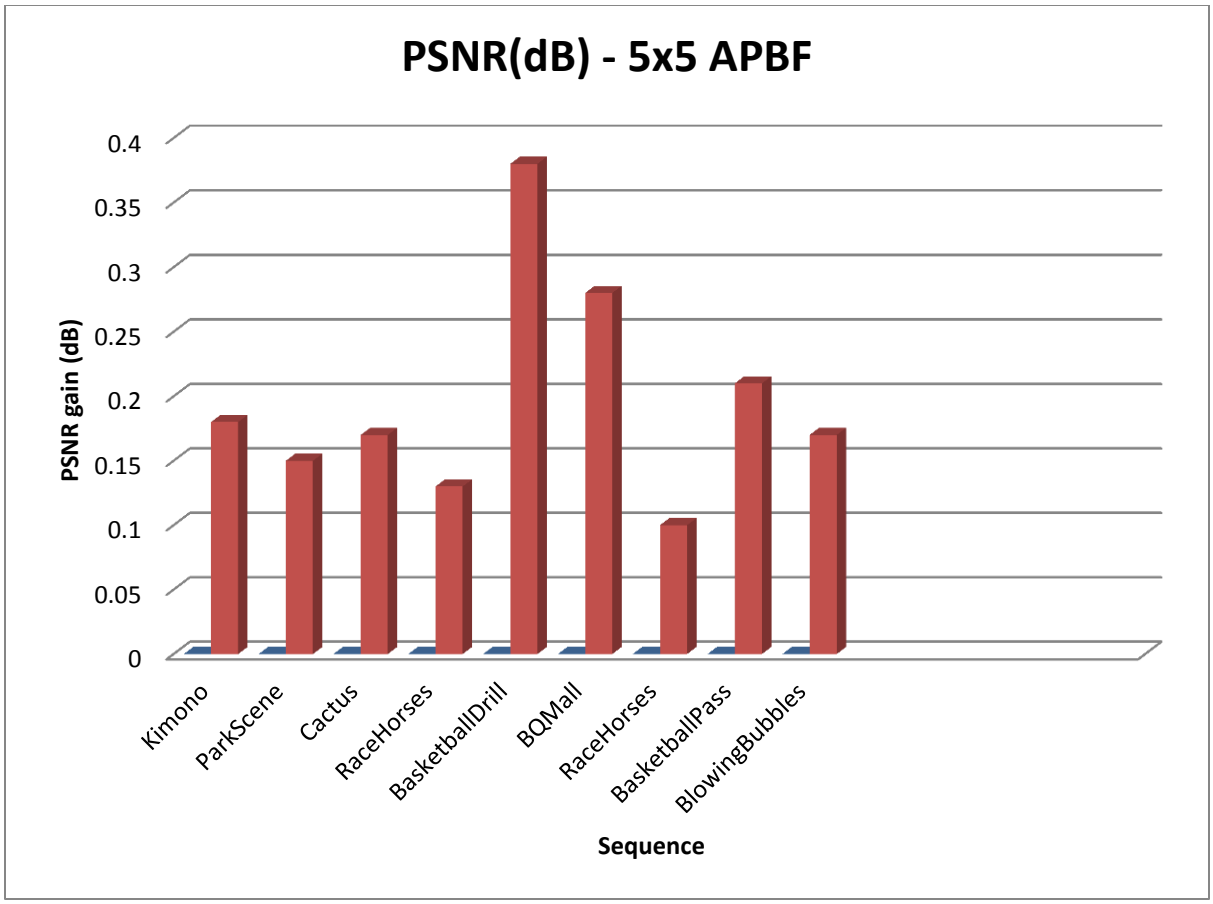


Figure 4.3 This plot shows the PSNR difference values for 5x5 APBF

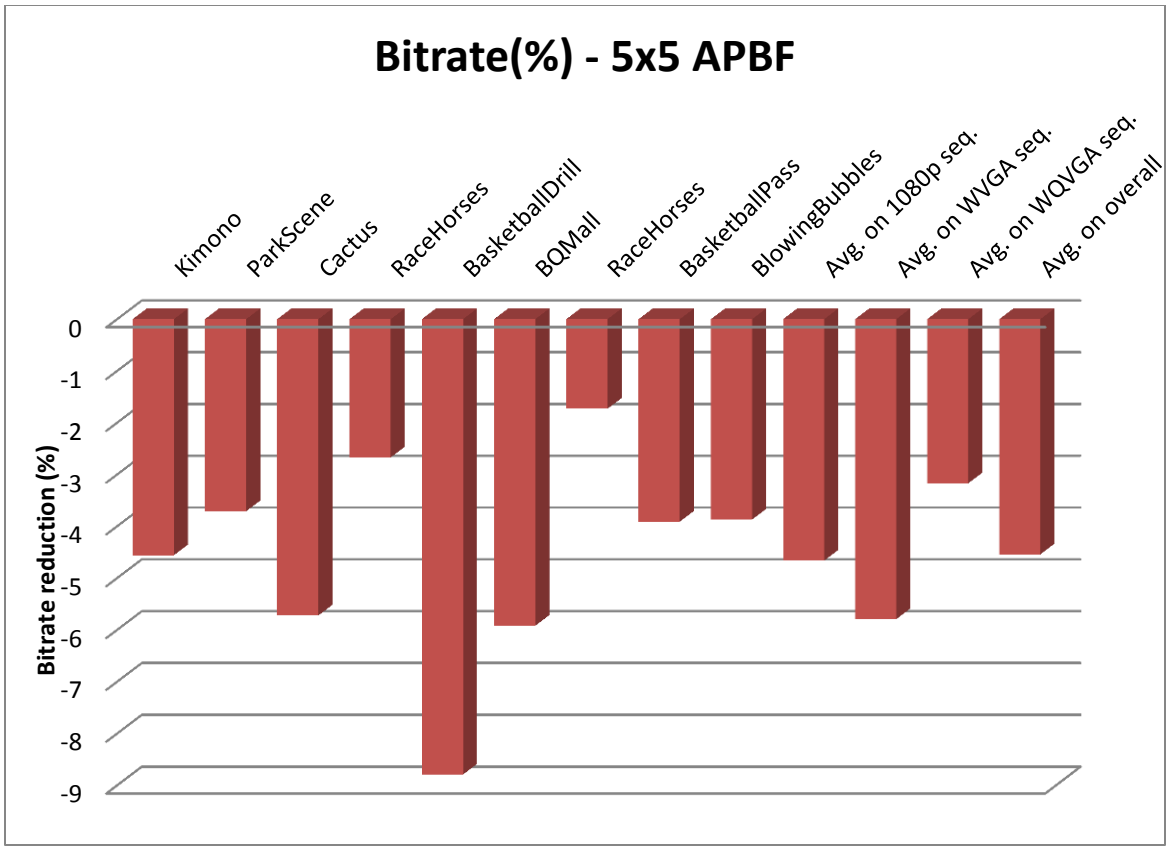


Figure 4.4 This plot shows the bitrate difference (%) values for 5x5 APBF

7x7 adaptive prediction block filtering:

Table 4.4 shows the compression performance and the computational complexity of the algorithm for 7×7 APBFs. Figure 4.5. represents the difference in PSNR (dB) between H.264 without APBF and with 7x7 APBF. Figure 4.6. represents the difference in bitrate (%) between H.264 without APBF and with 7x7 APBF.

Table 4.4 Experimental results of 7x7 APBF scheme compared to H.264/AVC

Sequence	Size	H.264/AVC + 7 × 7 APBF			
		BD-PSNR (dB)	BD- bitrate (%)	ΔT_{Enc} (%)	ΔT_{Dec} (%)
Kimono	1920 × 1080 (1080p)	0.16	-3.47	157.06	413.67
ParkScene		0.13	-2.83	153.44	406.93
Cactus		0.16	-5.85	155.91	556.88
RaceHorses	832 × 480 (WVGA)	0.09	-1.87	143.21	484.7
BasketballDrill		0.33	-7.96	154.04	599
BQMall		0.25	-5.66	147	482.8
RaceHorses	416 × 240 (WQVGA)	0.09	-1.42	142.42	417.56
BasketballPass		0.17	-3.83	154.51	376.16
BlowingBubbles		0.16	-3.48	138.66	499.4
Average on 1080p seq.		0.15	-4.05	155.47	459.16
Average on WVGA seq.		0.223	-5.163	148.083	522.167
Average on WQVGA seq.		0.14	-2.91	145.197	431.04
Average on overall		0.171	-4.041	149.583	470.789

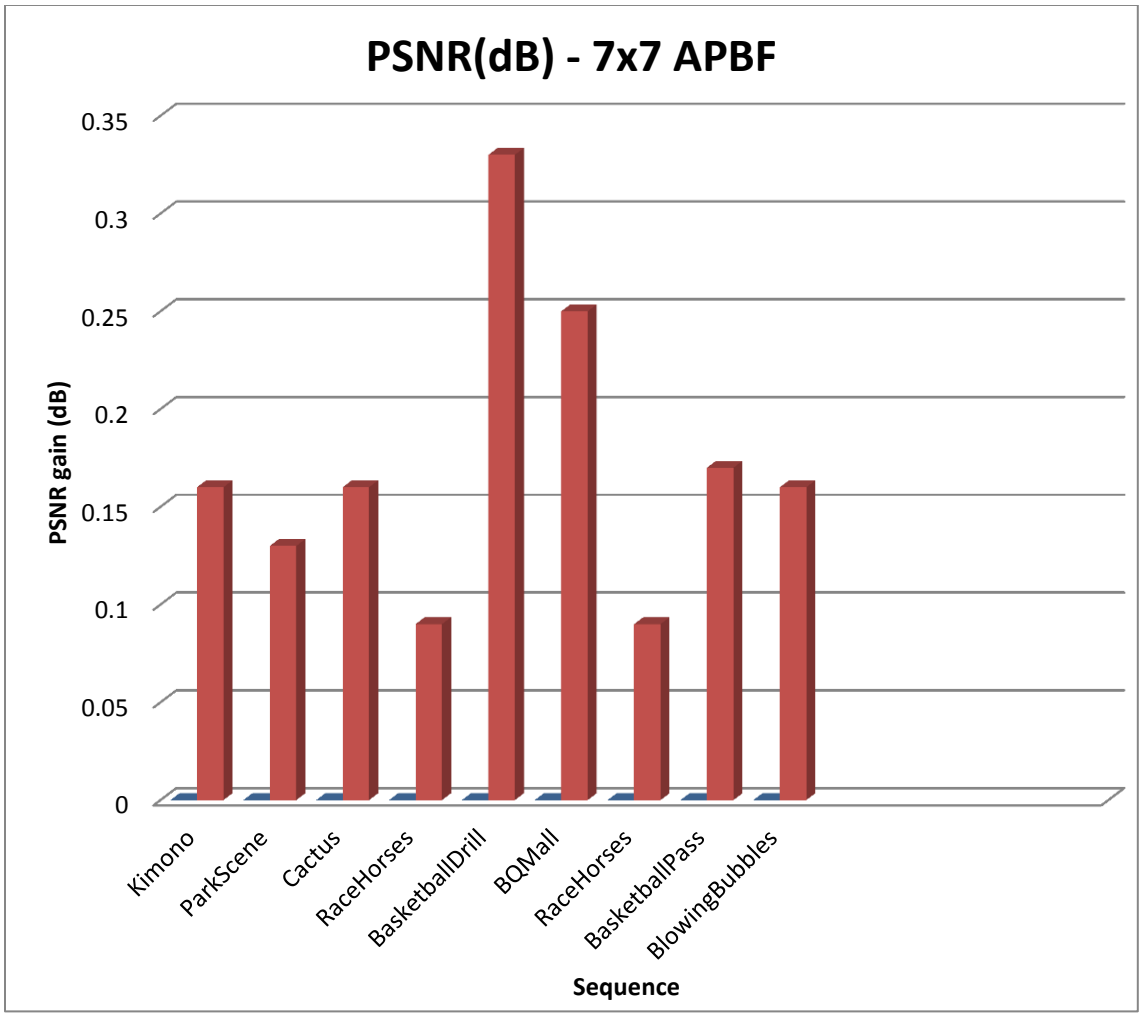


Figure 4.5 This plot shows the PSNR difference values for 7x7 APBF

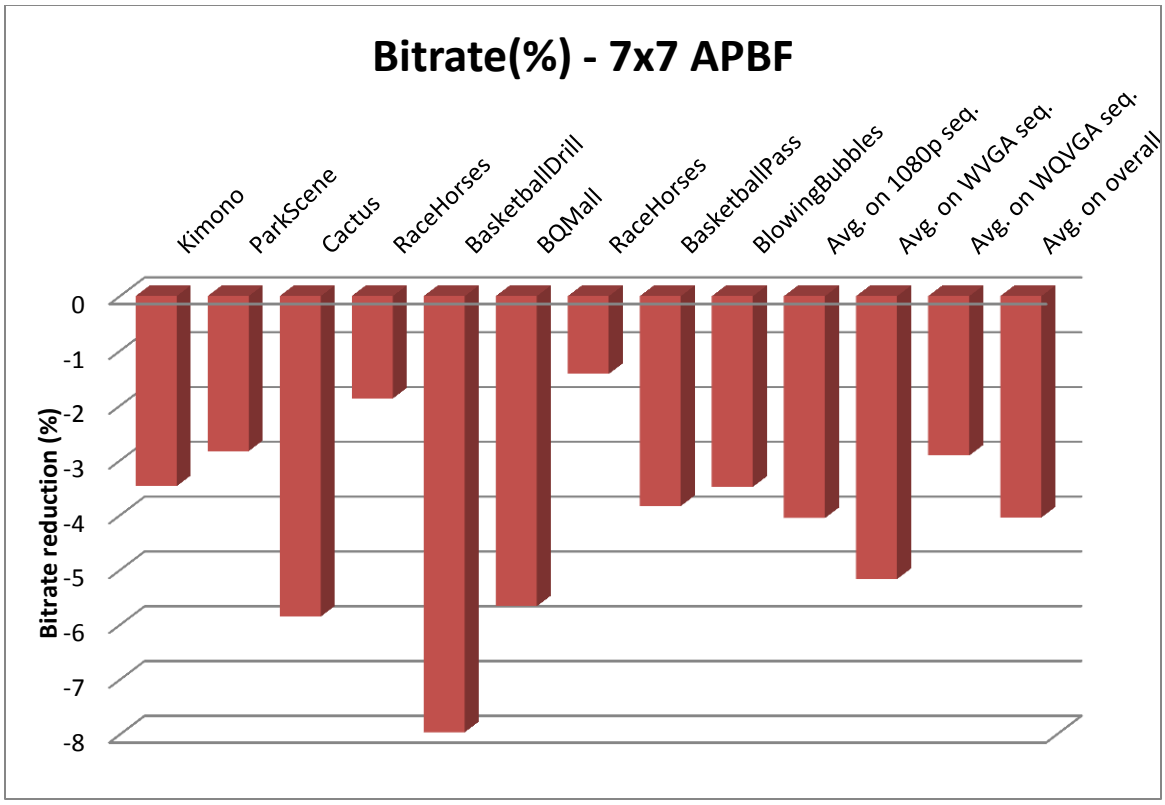


Figure 4.6 This plot shows the bitrate difference (%) values for 7x7 APBF

Conclusions

The result obtained demonstrates that when the smaller the filter tap used, the compression performance is compared with H.264/AVC is much better as shown in figure 4.7. Generally, using bigger tap filters produce better image quality when the image needs to be sharpened but in the application of compression since the coefficients of the Wiener filter are obtained from a reference frame, more number of coefficients implies more error. Therefore, the best compression performance is obtained when the 3×3 APBF is utilized, and the average BD-bitrate reduction is about 5.04% for overall sequences as seen in figure 4.8. The complexities of the encoder and the decoder are unavoidably increased because of the filter coefficient computation and filtering process. It should be also noted

that the increase in computational complexity is very low in 3×3 APBF. The increase rate of the encoder complexity is about 65.2% and decoder complexity is about 90.7% on an average when 3×3 APBF is used. As a result, it is verified that the APBF algorithm with a 3×3 filter achieves the highest coding performance in terms of the bitrate reduction as well as time consumption.

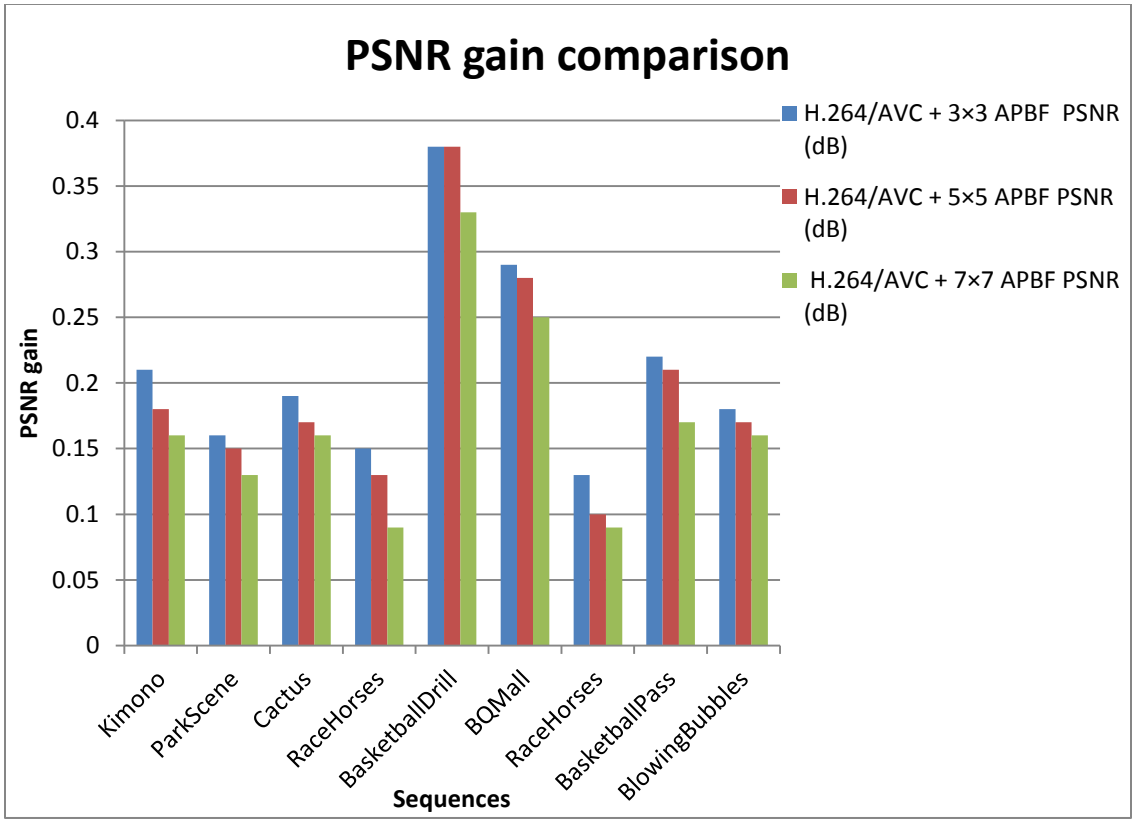


Figure 4.7 Comparison of PSNR difference (dB) values of 3x3, 5x5 and 7x7 APBF

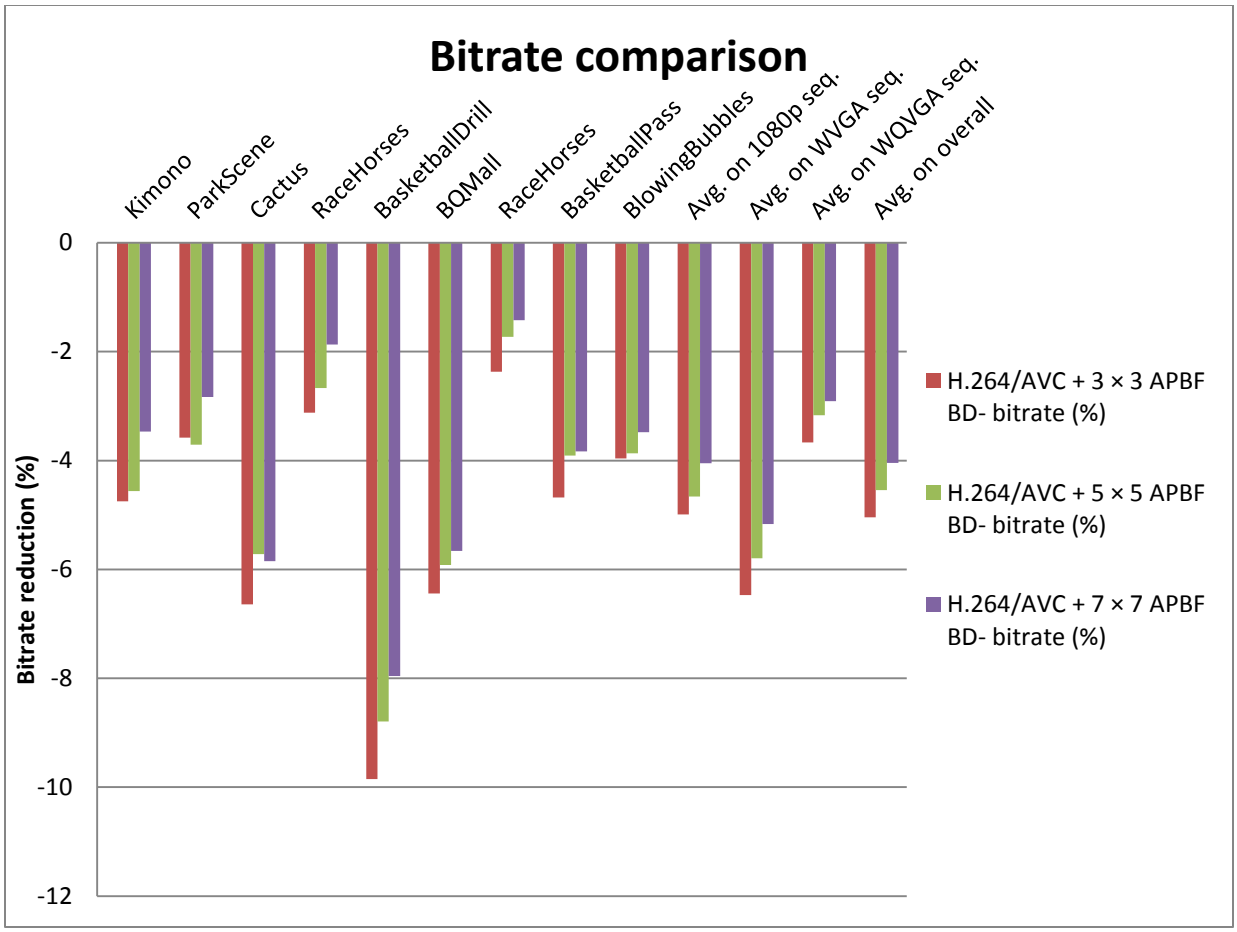


Figure 4.8 Comparison of bitrate difference (%) values of 3x3, 5x5 and 7x7 APBF

Summary:

The results are discussed explicitly and it is seen that the best APBF to choose would be 3x3 tap APBF. The next chapter includes further developments that can be explored to implement APBF to obtain better results.

Chapter 5

Future work

In order to reduce the encoder complexity of the APBF scheme, filtering can be applied only to the best inter-prediction mode. But to further improve the coding efficiency, the following are suggested:

1. In the implementation of APBF on sub-blocks, $(2N+1) \times (2N+1)$ a symmetric filter has been used. It is expected that both coding efficiency and computational complexity can be improved if other filter shapes are exploited for example, a cross-shaped filter [15]. Therefore, the APBF scheme needs to be adaptively adjusted depending on the target applications.
2. If the adaptive filter is applied to both decoded and prediction signals, additional coding gain is achieved at the expense of increase in the computational complexity.
3. It is seen that computational complexity is reduced to a large extent if QALF is implemented along with 3x3 APBF on a macroblock in [16]. The same logic can be used to implement QALF along with 3x3 APBF on SBs to reduce the computational complexity.
4. Instead of LMS scheme, RLS (Recursive least squares), LRLS (Lattice recursive least squares) or NLRLS (Normalized lattice recursive least squares) schemes can be used to obtain better APBFs.
5. APBF on SBs can also be implemented in HEVC standard keeping in view of the increase in computational complexity at the encoder and decoder.

Appendix A

Frames of sequences used [39]

Each of the video sequences chosen has a very distinctive character. Some reasons are: In the race horse video sequence, the horse racers and horses consists of the foreground while the grass is the background, both foreground and background are moving. It's a dynamic, motion-filled video. The basketball pass sequence contains pictures of high motion activity and high contrast. The random movements of the basketball players make the prediction much more difficult. Blowing bubbles has a comparably static background. The bubbles are growing and moving in random directions. The camera zooms out generally from the beginning to the end. BQ square video clip has a low motion background. Some people are moving in predicable directions with low speed. This sequence has lower motion activities.

1. Kimono – 1920 x 1080

2. Park scene – 1920 x 1080

3. Cactus – 1920 x 1080

4. BQ mall – 832 x 432

5. Basketball drill – 832 x 480

6. Race horses – 832 x 480

7. Blowing bubbles – 416 x 240

8. Basketball pass – 416 x 240

9. Race horses – 416 x 240

References

[1] T. Wiegand et al., “Overview of the H.264/AVC Video Coding Standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 7, pp. 560-576, July 2003.

[2] ITU-T VCEG KTA Reference software http://iphome.hhi.de/suehring/tml/download/KTA/

[3] Y. Vatis and J. Ostermann, “Adaptive Interpolation Filter for H.264/AVC,” IEEE Trans. Circuits Syst. Video Technol., vol. 19, no. 2, pp. 179-192, Feb. 2009.

[4] T. Chujoh, N. Wada, and G. Yasuda, “Quadtree-Based Adaptive Loop Filter,” ITU-T SG16/Q.6 Doc. COM16-C181-E, Geneva, Switzerland, Jan. 2009.

[5] Y. Ye, P. Chen, and M. Karczewicz, “High Precision Interpolation and Prediction,” ITU-T SG16/Q.6 Doc. VCEGAI33, Berlin, Germany, July 2008.

[6] J. Jung and G. Laroche, “Competition-Based Scheme for Motion Vector Selection and Coding,” ITU-T SG16/Q.6 Doc. VCEGAC06, Klagenfurt, Austria, July 2006.

[7] P. Chen, Y. Ye, and M. Karczewicz, “Video Coding Using Extended Block Sizes,” ITU-T SG16/Q.6 Doc. VCEG-AJ23, San Diego, USA, Oct. 2008.

[8] Y. Liu, “Unified Loop Filter for Video Compression,” IEEE Trans. Circuits Syst. Video Technol., vol. 20, no. 10, pp. 1378-1382, Oct. 2010.

[9] G.J. Sullivan and T. Wiegand, “Rate-Distortion Optimization for Video Compression,” IEEE Signal Process. Mag., vol. 15, no. 6, pp. 74-90, Nov. 1998.

[10] H. Lee et al., “Enhanced Block-Based Adaptive Loop Filter with Multiple Symmetric Structures for Video Coding,” ETRI J., vol. 32, no. 4, pp. 626-629, Aug. 2010.

[11] ISO/IEC JTC1/SC29/WG11 and ITU-T SG16/Q.6, “Joint Call for Proposals on Video Compression Technology,” WG11 Doc.N11113 and ITU-T SG16/Q.6 Doc. VCEG-AM91, Kyoto, Japan, Jan. 2010.

[12] G. Bjøntegaard, “Calculation of Average PSNR Differences between RD-Curves,” ITU-T SG16/Q.6 Doc. VCEG-M33, Austin, USA, Apr. 2001.

[13] D. Marpe, T. Wiegand and G. J. Sullivan, “The H.264/MPEG-4 AVC standard and its applications”, IEEE Communications Magazine, vol. 44, pp. 134-143, Aug. 2006.

[14] T. Wiegand and G. J. Sullivan, “The picturephone is here: Really”, IEEE Spectrum, vol.48, pp. 50-54, Sep. 2011.

[15] I. E. Richardson, “The H.264 Advanced Video Compression Standard”, 2nd Edition, Wiley 2010.

[16] Yeo-Jin Yoon et al., “Adaptive Prediction Block Filter for Video Coding”, ETRI J., vol.34, no. 1, pp 106-109, Feb. 2012.

[17] Ke-Ying Liao et al., “Rate-Distortion Cost Estimation for H.264/AVC”, IEEE transactions on circuits and systems for video technology, vol. 20, no. 1, pp. 38-49, Jan 2010.

[18] http://www.h265.net/2010/07/adaptive-interpolation-filter-for-video-coding.html

[19] http://www.h265.net/2009/08/adaptive-post-loop-filters-in-jmkta-part-2.html

[20] E. A. Fox, “Advances in interactive digital multimedia systems”, IEEE Computer, vol. 24, pp. 9-21.

[21] B. Furht, “Survey of multimedia compression techniques and standards. Part 1: JPEG standard”, Real time imaging, vol. 1, pp.49-67, 1995.

[22] B. Furht, “Multimedia systems : an overview”, IEEE Multimedia, vol. 1, pp. 47-59, 1994.

[23] C. E. Manning “Why do we need compression?”, http://www.newmediarepublic.com/dvideo/compression/adv03.html, 1996.

[24] S. 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.

[25] Open source article, “Intra frame coding” : http://www.cs.cf.ac.uk/Dave/Multimedia/node248.html

[26] Open source article, “H.264/MPEG-4 AVC,” Wikipedia Foundation, http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC

[27] A. K. Kulkarni,” Implementation of fast inter-prediction mode decision in H.264/AVC video encoder”, M.S. Thesis, E.E Dept, UTA, 2012. http://www-ee.uta.edu/Dip/Courses/EE5359/index.html

[28] S. S. Vaidyanath, “ Low complexity H.264 encoder using machine learning for streaming applications” M.S Thesis, E.E Dept, UTA, 2011. http://www-ee.uta.edu/Dip/Courses/EE5359/index.html

[29] “http://www.vcodex.com/_les/ ," working of H.264 codec.

[30] I. E.G. Richardson, “H.264 and MPEG-4 video compression: video coding for next-generation multimedia”, Wiley, 2003.

[31] S. Subbarayappa, “ Implementation and analysis of directional discrete cosine transform in H.264 for baseline profile ” M.S. Thesis, E.E Dept, UTA, 2012. http://www-ee.uta.edu/Dip/Courses/EE5359/index.html

[32] Y. Vatis and J. Ostermann, ITU-T SG16/Q [15] (VCEG) VCEG-AE16, Marrakech, Morocco, Jan. 2007.

[33] JVT of ISO/IEC & ITU-T, Draft ITU-T Recommendation H.264 and Draft ISO/IEC 14496-10 AVC, Doc JVT-Go50. Pattaya, Thailand, 2003.

[34] T. Wedi, “Adaptive interpolation filter for motion and aliasing compensated prediction”, in Proc VCIP, San Jose, CA, USA, pp. 415–422, Jan. 2002.

[35] JM11 reference software JM11KTA2.3 http://www.h265.net/2009/04/kta-software-jm11kta23.html

[36] Open source article, “Bit rate”, http://en.wikipedia.org/wiki/Bit_rate

[37] Open source article, “Wiener filter” http://en.wikipedia.org/wiki/Wiener_filter

[38] S. V. Vaseghi, “Advanced digital signal processing and noise reduction”, 2nd edition, 2000.

[39] Link for video sequences: ftp.tnt.uni-hannover.de

Biographical Information

Bhavana Prabhakar was born on June 21st, 1989 in KGF, Karnataka, India. She is the only daughter of V. N. Prabhakara Rao and M. L. Sharada Rao. She received her Bachelor's Degree in Electronics and Communication Engineering from K S Institute of Technology, Bangalore in 2011. She was offered a position at Robert Bosch, Koramangala, Bangalore but she decided to pursue her Master's Degree in Electrical Engineering at University of Texas at Arlington to fulfill her desire for studying further. During her studying period at Arlington she was very enthused to study the field of Multimedia and joined the Multimedia Group at UTA in January 2012 under the guidance of Dr. K. R. Rao. She got an opportunity to intern at INTEL Corporation between May 2012 to May 2013 at Santa Clara, California. After her graduation, she intends to find a job in multimedia field where she can utilize her knowledge and experience.