IMPLEMENTATION OF A FAST INTER-PREDICTION MODE DECISION IN H.264/AVC VIDEO ENCODER

by

AMRUTA KIRAN KULKARNI

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 2012

ACKNOWLEDGEMENTS

First and foremost, I would like to take this opportunity to offer my gratitude to my supervisor, Dr. K.R. Rao, who invested his precious time in me and has been a constant support throughout my thesis with his patience and profound knowledge. His motivation and enthusiasm helped me in all the time of research and writing of this thesis. His advising and mentoring have helped me complete my thesis.

Besides my advisor, I would like to thank the rest of my thesis committee. I am also very grateful to Dr. Dongil Han for his continuous technical advice and financial support.

I would like to acknowledge my research group partner, Santosh Kumar Muniyappa, for all the valuable discussions that we had together. It helped me in building confidence and motivated towards completing the thesis. Also, I thank all other lab mates and friends who helped me get through two years of graduate school.

Finally, my sincere gratitude and love go to my family. They have been my role model and have always showed me right way. Last but not the least; I would like to thank my husband Amey Mahajan for his emotional and moral support.

April 20, 2012

ABSTRACT

IMPLEMENTATION OF A FAST INTER-PREDICTION MODE DECISION IN H.264/AVC VIDEO ENCODER

Amruta Kulkarni, M.S

The University of Texas at Arlington, 2011

Supervising Professor: K.R. Rao

H.264/MPEG-4 Part 10 or AVC (advanced video coding) is currently one of the most widely used industry standards for video compression. There are several video codec solutions, both software and hardware, available in the market for H.264. This video compression technology is primarily used in applications such as video conferencing, mobile TV, blu-ray discs, digital television and internet video streaming.

This thesis uses the JM 17.2 reference software [15], which is available for all users and can be downloaded from http://iphome.hhi.de/suehring/tml. The software is mainly used for educational purposes; it also includes the reference software manual which has information about installation, compilation and usage.

In real time applications such as video streaming and video conferencing it is important that the video encoding/decoding is fast. It is known, that most of the complexity lies in the H.264 encoder, specifically the motion estimation (ME) and mode decision process introduces high computational complexity and takes a lot of CPU (central processing unit) usage. The mode decision process is complex because of variable block sizes (16X16 to 4x4) motion estimation and half and quarter pixel motion compensations.

Hence, the objective of this thesis is to reduce the encoding time while maintaining the same quality and efficiency of compression.

The Fast adaptive termination (FAT) [30] algorithm is used in the mode decision and motion estimation process. Based on the rate-distortion (RD) cost characteristics all the inter modes are classified as either skip modes or non-skip modes. In order to select the best mode for any macroblock, the minimum RD cost of these two modes is predicted. Further, for skip mode, an early-skip mode detection test is proposed; for non-skip mode a three-stage scheme is proposed to speed up the mode decision process. Experimental results demonstrate that the proposed technique has good robustness in coding efficiency with different quantization parameters (QP) and various video sequences. It is able to achieve encoding time saving by 47.6% and loss of only 0.01% decrease in structural similarity index matrix (SSIM) with negligible degradation in peak signal to noise ratio (PSNR) and acceptable increase in bit rate.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

Table Page

LIST OF ACRONYMS

AVC – Advanced Video Coding

ASO – Arbitrary Slice Order

B slice – Bi-directionally predictive slice

CABAC – Context-Based Adaptive Binary Arithmetic Coding

CAVLC – Context Adaptive Variable Length Coding

CD-ROM – Compact Disc- Read Only Memory

CIF – Common Intermediate Format

DCT – Discrete Cosine Transform

DVD – Digital Video Disk

EPZS – Enhanced Predictive Zonal Search

FBME – Full Search Block Matching Motion Estimation

FMBA – Flexible Macroblock Address Predictor

FMO – Flexible Macroblock Order

HD – High Definition

I slice – Intra slice

ISO – International Organization for standardization

ITU-T – International Telecommunication Union- Transmission standardization sector

JVT – Joint Video TeamJM - Joint Model

MPEG – Moving Picture Experts Group

MV- Motion Vector

NAL – Network Abstraction Layer

NTSC – National Television System Committee

P slice	Predictive slice
PSNR	Peak Signal to Noise Ratio
PCM	Pulse Code Modulation
QCIF	Quarter Common Intermediate Format
QP	Quantization Parameter
RAM	Random Access Memory
RDO	Rate Distortion Optimization
RS	Redundant Slices
SAD	Sum of Absolute Differences
SATD	Sum of Absolute Transformed Differences
SSIM	Structural Similarity Index Metric
TV	Television
VCEG	Video Coding Experts Group
VCL	Video Coding Layer
VCE	Video Codec Engine
VLE	Variable Length Encoding
FIR	Finite Impulse Response
EPZS	Enhanced predictive zonal search
UMhexagonS	Uneven multi grid hexagon search

CHAPTER 1

INTRODUCTION

1.1 Significance

Innovations in communication systems have been tremendous in the last decade. Technology in communication systems has transformed. In the early days,analog television would have very few channels. Mobile phones used to make voice calls or send SMS (short message service). Internet connections were slow, mostly connected through dial –up modem connected via telephone lines. Data was stored on floppy disks, magnetic tapes and bulky hard drives.

Today the world has transformed into the so called “digital age” or “electronic age”, where mobile phones are called smart phones because they can not only make phone calls but are also used for web browsing, sending emails, watching videos , transfering data, navigation purposes and as camera. Digital television sets have become more compact with availability of regional and international channels with HD (High Definition) quality. Data is stored on re-writable DVDs, Blu-ray discs and hard disks which are light weight, portable with huge space for storage. Internet connection is blazing fast with wireless routers and modems operating at faster speeds [4]. In this fast growing world of communications, data compression is still one of the most essential components in any multimedia system. Modern data compression techniques offer the possibility to store or transmit a vast amount of data necessary to represent digital videos and images in an efficient and robust way.

Compression is the process of removing redundant information and representing data with fewer bits than the original information would use. It is useful because it helps to reduce the consumption of expensive resources such as data storage on hard disks/servers and transmission bandwidths. Hence, still lot of research is being done on compression techniques to continuously improve real-time data transmission using less resources. Compression techniques are categorized as lossless or lossy.Lossless compression is possible because most of the real-world data has statistical redundancy. If the data has been compressed in a lossless way, the original data can be recovered with no loss. Lossless compression exploits statistical redundancy and represents data with more fidelity and less error[4]. It is beneficial in areas like text compression and audio compression. Lossy compression involves some information loss, so the data cannot be recovered exactly.It is applied in areas where data distorion is tolerable like video compression, image compression and some types of audio compression. Lossy image compression is used in digital cameras, to increase the storage capacity with less degradation of picture quality than original. Similarly lossy video compression is used on DVDs, Blu-ray disks [38],Internet telephony using MPEG-2 [39], H.264 video codec.

Video sequences conatin a significant amount of statistical and subjective redundancies within and between frames. The ultimate goal of a video source coding is bit rate reduction for storage and tranismission by exploring both statistical (spatial) and subjective (temporal) redundancies and to encode a “minimum set” of infromation using entropy coding tecniques[5].The volume of data in multimedia signals is very high, For example, to represent 2 minutes of CD-quality music (44,100 samples per second, 16 bits per sample) requires more than 84 million bits.For video signals to represent 1 second of video without compression (using the CCIR 601 format) [40], more than 20 Mbytes or 16Mbits is required.[6] This data indicates the importance of compression for multimedia signals.

Multimedia consumer applications have a very large market. The revenue involved in digital TV broadcasting and DVD, Blu-ray distrubtions are substantial. Thus standardization of video coding is essential. Standards simplify inter-operability between encoders and decoders from different manufacturers, they make it possible for different vendors to build platforms that incorporate video codecs, audieo codecs, security and rights management interact in well defined and consistent ways. There are numerous video compression standards both open source and proprietary depending on the applications and end-usage. The moving pictures experts group (MPEG) and video coding experts group (VCEG) joined together to form the joint video team (JVT) in 2001, which developed the ITU-T Rec. H.264 | ISO/IEC 14496-10,commonly referred as “H.264” / “MPEG-4 Part 10” / “Advanced Video Coding (AVC) ” published by the International Telecommunication Union (ITU) and the International Standards Organisation (ISO) [15].

1.2 Why is complexity reduction important in H.264/AVC ?

H.264/AVC has very efficient compression methods, which allow it to compress video much more efficiently than older standards and provide more flexibility for application to a wide variety of network enviornments. To achieve highly efficient compression, the computational cost associated with it is also very high. This is the reason why, these increased compression effeciencies cannot be exploited across all application domains. Resource constrained devices such as cell phones and other embedded systems use simple encoders or simpler profiles of the codec to tradeoff compression efficieny and quality for reduced complexity [6]. Video coding standards specify the decoding process and bitstream syntax of the compressed video. The encoding process or the process of producing a standard compliant video is not specified. This approach leaves room for innovation in the encoding algorithm development. The work in this thesis focuses on such a low complexity mode selection encoding algorithm .

1.3 Summary

The research presented here proposes a reduced complexity H.264 encoder by making use of JM 17.2 reference software [4]. A new technique is implemented for reducing encoding complexity in H.264. The results show reduction in complexity in terms of encoding time for different videos contexts, with acceptable deterioration in the PSNR (Peak Signal to Noise ratio) and bit-rates.

CHAPTER 2
H.264 VIDEO CODING STANDARD

2.1 Introduction

H.264/AVC standard was first published in 2003, with several revisions and updates published since then. It builds on the concepts of earlier standards such as MPEG-2 [40] and MPEG-4 [3] visual and offers the potential for better compression efficiency, i.e. better-quality compressed video, and greater flexibility in compressing, transmitting and storing video [4].The Joint Video Team (JVT) then developed extensions to the original standard that are known as the fidelity range extensions (FRExt) [7]. The new functionalities provided by the FRExt amendment include higher quality video coding by supporting increased sample bit depth precision and higher-resolution color information,adaptive switching between 4x4 and 8x8 integer transforms. The other important tools FRExt added are scaling matrices for perceptual tuned frequency –dependent quantization, lossless coding capability and residual color transform.[8]

H.264/AVC like any other motion-based codecs, uses the following basic principles of video compression[9]:

- 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.

Inherently there is a lot of redundancy in digital videos, video compression uses both spatial and temporal compresion to represent a digital video with less data. A video is basically a stack of frames attached one after the other, in most real world video data the difference between successive frames is minimal and thus data reduction is possible. The video frames are called picture types or frame types. The frame types are classified as I, P and B frames. I frames are intra coded frames, which do not use any other video frames for compression and thus are least compressible, P frames are predictive frames which use data from previous frames for compression and are more compressible than I-frames. B frames are bi-predictive frames ,which use both past and future frames for compression and thus are the most compressible frames. A frame is divided into multiple blocks known as macroblocks which are typically 16x16 pixels, on Y plane of the original image. A macroblock can be represented in several ways in YCbCr space. Figure 2.1 shows different formats for YCbCr color space.

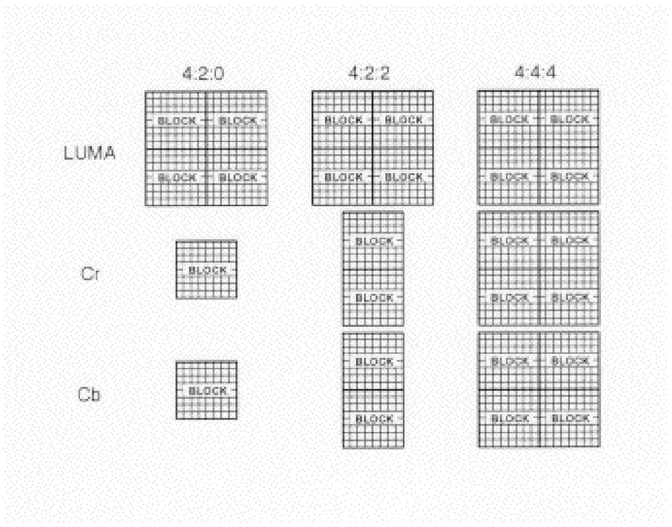


Figure 2.1 Formats (4:4:4, 4:2:2 and 4:2:0) for YCbCr color space [1]

The 4:4:4 represents 4 Y (luma) blocks, 4 Cb and 4 Cr blocks respectively , it represents a full bandwidth video and contains as much information as the data if it would be in RGB color space. The 4:2:2 contains half as much the 4:4:4 chrominance information and 4:2:0 contains one quarter of the chrominance information. H.264/AVC and MPEG-2 video codecs can support all the chrominance formats, but most of the consumer level products use 4:2:0 mode.

Intra (spatial) prediction compression uses only the current frame for prediction.It predicts, if there is any movement from the neighboring blocks. It reuses the mode information from adjacent blocks. Intra frame prediction is mostly used in uniform zones of the picture where there is not much movement.

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

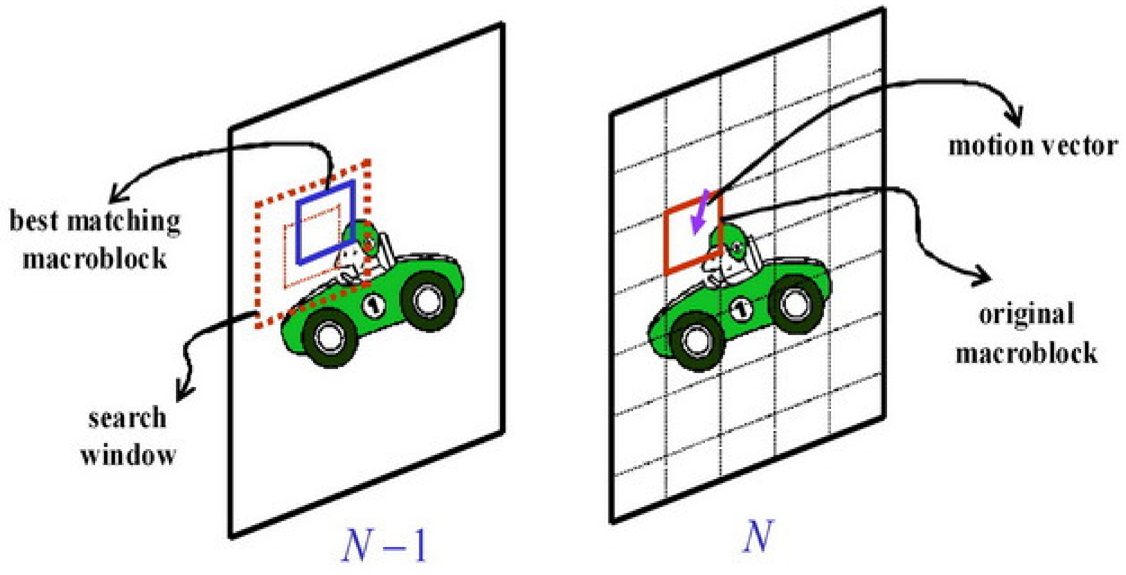


Figure 2.2 Block based motion compensation in H.264 [3]

A reference frame is used to remove redundancy, this frame is strictly coded with raw pixel values so has full information stored. A reference frame is also known as a I-frame, it can be a past 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 one it is encoding. Most likely, the encoder will find such a block, but it may not be the perfect match and can introduce prediction error, so the algorithm takes difference of the reference frame block and the block it is encoding and calculates the prediction error. If the prediction error is above certain threshold value, the algorithm searches for another reference frame block with matching characteristics and calculates prediction error. If a matching block with minimum prediction error is found, the encoder only transmits a vector, known as motion vector , which has co-ordinates that point to the block from the reference frame. Thus the encoder takes transform of the difference between reference and predicted frames, quantizes the transform coefficients followed by entropy coding which would be enough for reproducing the video frame at the decoder. It can so happen, that a similar block found from the reference frame introduces huge prediction error, which makes the overall size of motion vector and prediction error greater than raw encoded pixels of the block. In such cases, the algorithm makes an exception and sends raw encoded block.

The inter frame types,commonly known as P and B-frames and intra frame,I-frame together form a GOP (Group of Pictures). Fig. 2.3 shows a GOP structure for H.264/MPEG-4.

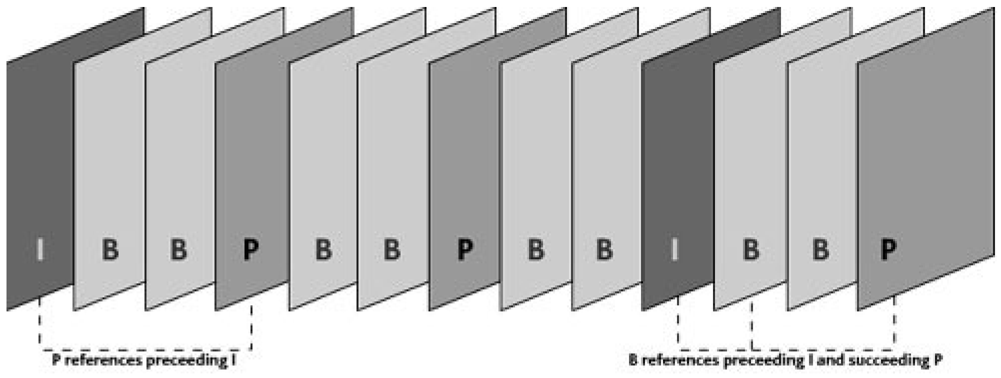

I
B
B
P
B
B
P
B
B
I
B
B
P
P references preceeding I
B references preceeding I and succeeding P

Figure 2.3 GOP structure for H.264/MPEG-4 [3]

A GOP (group of pictures) always begins with an I-frame. It contains full information therefore any errors within the GOP structure are corrected using the I-frame. B-frames are primarily used for compression efficiency but they also propogate errors in H.264. P-frames contain motion compensated difference from the previous I-frame or P-frame. The distance between two I-frames is known as GOP length.

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

- 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 macroblock coding
- An in-loop deblocking filter
- Loss resilence features like network abstraction layer (NAL), flexible macroblock 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.

2.2 Profiles and Levels of H.264/AVC

H.264/AVC standard defines 17 profiles , with different set of capabilities, targeting specific classes of applications. For example, error resilience tools are not important for networks with very little data loss or corruption. Forcing the decoder to implement all the tools makes its design unncessarily more complex. Segregation of the tools in different profiles helps the decoder to choose to implement only a subset of the tools. Figure 2.4 shows different profiles defined in H.264/AVC

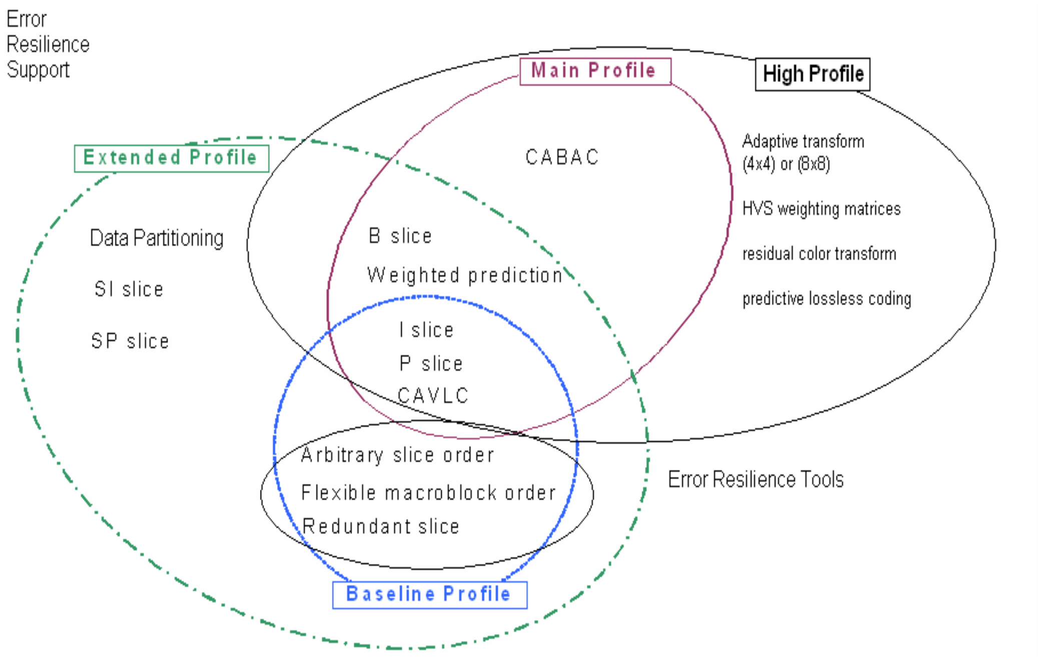


Figure 2.4 Profiles in H.264 with distribution of various coding tools [9]

2.2.1 Profiles

The following profiles were defined in the first standard and they still exist:

- Baseline Profile (BP)

  The baseline profile contains I and P-slices, CAVLC and some error resilience tools such as FMO, ASO and RS. It does not contain B –slices The importance of this profile is in applications which require some data loss robustness; it is mainly used in low cost applications like video-conferencing and mobile applications.

- Extended Profile (XP)

  Extended profile is a super set of baseline, it adds B, SP and SI-slices and interlace coding tools and further support in error resilience tool set in the form of data partitioning (DP). This profile is useful for video streaming applications; it has relatively high compression capability and extra robustness for data losses and server stream switching.

- Main Profile (MP)

Main profile contains I, P and B slices, CAVLC and CABAC entropy coding, it does not include error resilience tools (FMO, ASO, RS and DP) or SI and SP slices. This profile is mainly popular in consumer applications like digital TV broadcasting for standard definition. However its importance faded when the high profile was developed as FRExt [11] amendment in 2004.

The FRExt [11] amendment defined four new profiles:

- High Profile (HP)
- High Profile 10 (Hi10P)
- High Profile 4:2:2 (Hi422P)
- High Profile 4:4:4 (Hi444P)

Figure 2.5 shows tools introduced in FRExt for high profiles. All of these profiles include three enhancements of coding efficiency:

- Adaptive macroblock 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.

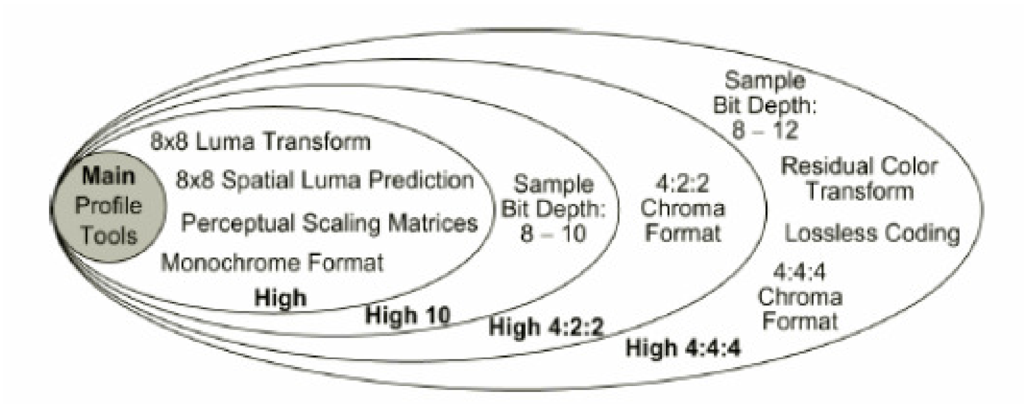


Figure 2.5 Tools introduced in FRExt and their classification under FRExt profiles [4]

All the high profiles also support monochrome coded video sequences, in addition to typical 4:2:0 video. The difference in capability among these profiles is primarily in terms of supported sample bit depths and chroma formats. However, the high 4:4:4 profiles also support residual color transform and predictive lossless coding features. [9]

2.2.2 Levels

Picture size and frame rate play main role in influencing the processing power and the memory size needed for implementation. Table 2.2.1 provides a comparison of the high profiles introduced in FRExts with a list of different coding tools utilized in the profile [11]. H.264/AVC defines 16 different levels, tied mainly to the picture size and frame rate. Levels also provide constraints on the number of reference frames and the maximum compressed bit rate that can be used. For primarily addressing the needs of 3G wireless environments the level “1B” was added in the FRExt amendment. The FRExt profiles are specified for more demanding high-fidelity applications, with increased bit-rate capabilities.

Table 2.1 Comparison of the high profiles and corresponding coding tools introduced in the FRExts [11]

Coding tools	High	High 10	High 4:2:2	High 4:4:4
Main Profile tools	x	x	X	X
4:2:0 Chroma format	x	x	X	X
8 bit sample bit depth	x	x	X	X
8x8 vs. 4x4 transform adaptively	x	x	X	X
Quantization scaling matrices	x	x	X	X
Separate Cb and Cr Quantization parameter (QP) control	x	x	X	X
Monochrome video format	x	x	X	x
9 and 10 bit sample depth		x	X	X
4:2:2 Chroma format			X	X
11 and 12 sample bit depth				X
4:4:4 Chroma format				X
Residual color transform				X
Predictive lossless coding				X

2.3 H.264 Encoder

H.264 encoder works on the same principles as that of any other codec. Figure 2.6 shows the basic building blocks of H.264 video codec.

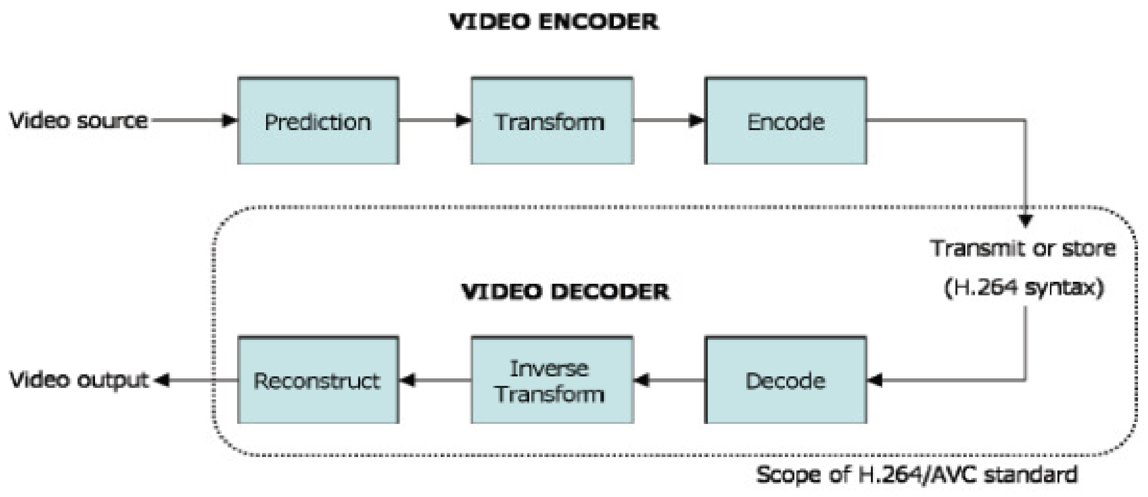


Figure 2.6 H.264 video coding and decoding process [4]

The input to the encoder is generally an intermediate format stream, which goes through the prediction block; the prediction block will perform intra and inter prediction (motion estimation and compensation) and exploit the redundancies that exist within the frame and between successive frames. The output of the prediction block is then transformed and quantized. An integer approximate of the discrete cosine transform is used (DCT) for transformation [12]. It uses 4x4 or 8x8 integer transform, and outputs a set of coefficients each of which is a weighted value for a standard basis pattern. The coefficients are then quantized i.e. each coefficient is divided by an integer value. Quantization reduces the precision of the transform coefficients according to the quantization parameter (QP). Typically, the result is a block in which most or all of the coefficients are zero, with a few non-zero coefficients. Next, the coefficients are encoded into a bit stream. The video coding process creates a number of parameters that must be encoded to form a compressed bit stream [13]. These values include:

- Quantized transform coefficients.
- Information to re-create prediction.
- Information about the structure of compressed data and the compression tools used under encoding.

These parameters are converted into binary codes using variable length coding or arithmetic coding. Each of these encoding methods produces an efficient, compact binary representation of the information. The encoded bit stream can now be transmitted or stored.

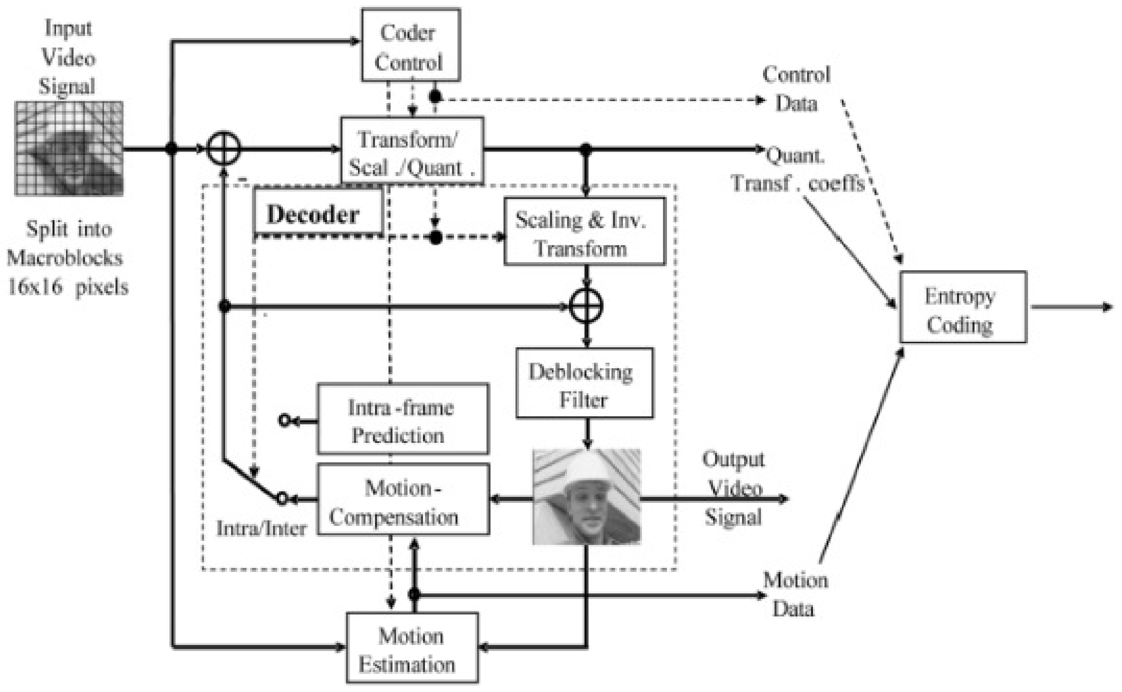


Figure 2.7 Basic coding structure of H.264/AVC for a macroblock [9]

2.3.1 Intra prediction

Intra prediction exploits the spatial correlation among pixels, there are three basic types defined:

- Full macroblock prediction for 16x16 luma or the corresponding chroma size
- 8x8 for luma prediction in FRExt [11] defined profiles

- 4x4 luma prediction

In full macroblock prediction, the edge pixels of the neighboring previously decoded macroblocks are used to predict the pixel values of an entire macroblock of luma or chroma. Full macroblock intra prediction is used for luma in a macroblock type called the intra 16x16 intra macroblock type. Because of the differences in sizes for chroma arrays, the macroblock in different chroma sizes are used i.e.8x8 chroma in 4:2:0 macroblocks, 8x16 chroma in 4:2:2 macroblocks and 16x16 chroma in 4:4:4 macroblocks. The prediction type for 16x16 macroblock is shown in Fig 2.8.

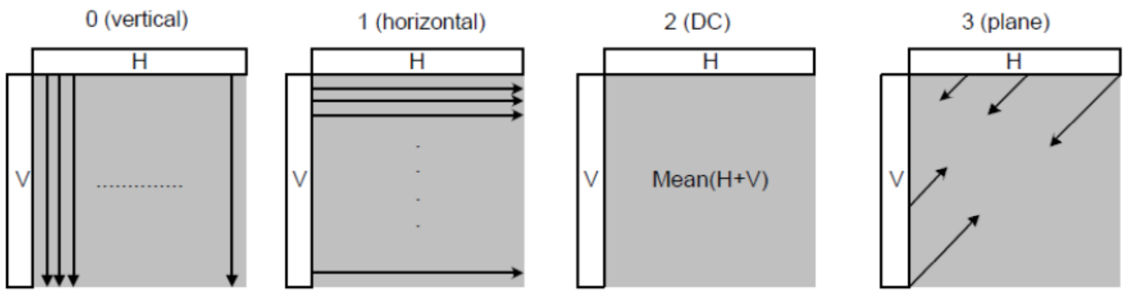


Figure 2.8 Intra prediction blocks for 16x16 luma macroblocks [4]

Full macroblock prediction can be performed in one of four different ways that can be used by the encoder to select the prediction of each macroblock:

(i) Vertical: For vertical prediction the pixel values of a macroblock are predicted from the pixels just above the macroblock. Vertical mode is commonly referred as the mode 0 for intra prediction.

(ii) Horizontal: In horizontal prediction the pixel values of a macroblock are predicted from the pixels just left to the macroblock. Horizontal mode is commonly referred as the mode 1.

(iii) DC: The luma values of the neighboring pixels are averaged and that average is used as predictor. DC is commonly referred to as the mode 2.

(iv) Planar: In planar prediction, a three curve fitting equation is used to form a prediction block having a brightness, slope in horizontal direction and slope in vertical direction that approximately matches the neighboring pixels.

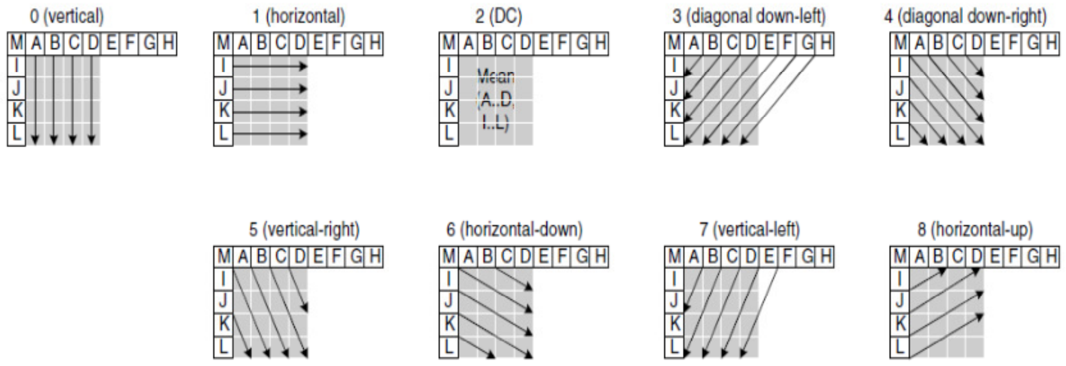


Figure 2.9 4x4 Luma intra prediction modes in H.264 [4]

In Figure 2.9, Pixels A to M are previously reconstructed and coded. In spatial 4x4 prediction mode, the values of each 4x4 block of luma samples are predicted from the neighboring pixels above or left of a 4x4 block. The encoder can select from the nine differential directional ways of performing the prediction for a 4x4 intra prediction block for luma as shown in Figure 2.9 [4]. Each prediction direction corresponds to a particular set of spatially dependent linear combinations of previously decoded samples for use as the prediction of each input sample. In mode 0, the samples of the macroblock are predicted from the neighboring samples on the top. In mode 1, the samples of the macroblock are predicted from the neighboring samples from the left. In mode 2, the mean of all the neighboring samples is used for prediction. Mode 3 is in diagonally down-left direction. Mode 4 is in diagonal down-right direction. Mode 5 is in vertical-right direction. Mode 6 is in horizontal-down direction. Mode 7 is in vertical-left direction. Mode 8 is in horizontal up direction. The predicted samples are calculated from a weighted average of the previously predicted samples A to M.

In FRExt profiles, 8x8 luma prediction can be selected, which uses basically the same concepts as 4x4 predictions. The 8x8 luma prediction has the block size as 8x8 and uses low-pass filtering of the predictor to improve prediction performance.

2.3.2 Inter Prediction

Inter prediction creates a prediction model from one or more previously encoded video frames. The temporal correlation between the neighboring frames along with motion estimation and compensation algorithms is used for encoding. An inter coded frame is divided into 16x16 size macroblocks; each macroblock is divided into 16x16, 16x8 8x16 and 8x8 sized blocks. The 8x8 sub-macroblock can further be partitioned into 8x4, 4x8, 4x4 sized blocks. Figure 2.10 illustrates the partitioning of a macroblock and sub-macroblocks [9]. A smaller block size ensures less residual data; however smaller block sizes also mean more motion vectors and hence more number of bits required to encode these motion vectors [1].

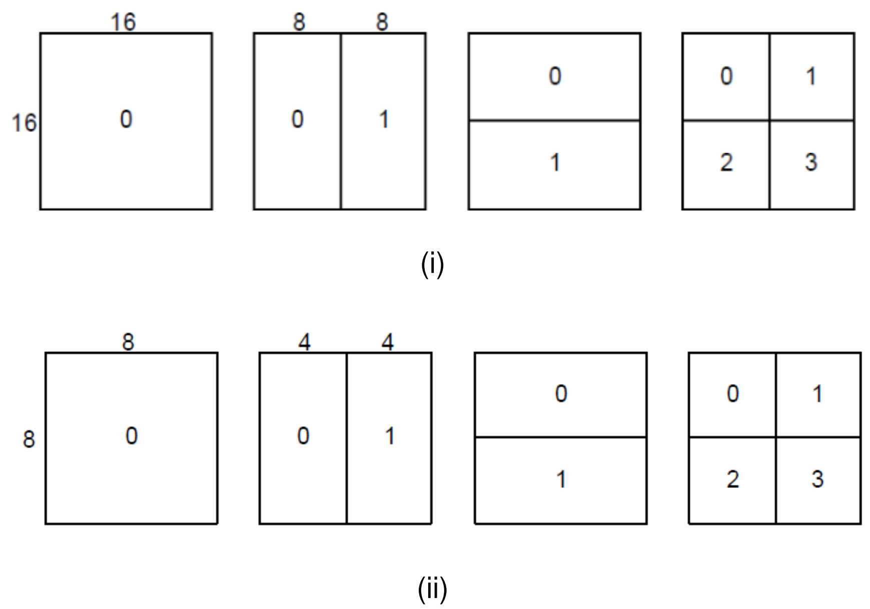


Figure 2.10 Macroblock portioning in H.264 for inter prediction [2] (i) (L-R) 16x16, 16x8, 8x16, 8x8 blocks; (ii)(L-R) 8x8,8x4,4x8 4x4 blocks

2.3.3 Inter prediction of macroblocks in P-slices

The H.264 video codec used block based motion compensation, the same principle adopted by every major video coding standard. H.264 provides some important differences like block sizes down to 4x4 and fine sub-pixel motion vectors up to quarter pixel in luma component [14].

These partitions and sub-partitions give rise to a large number of possible combinations within each macroblock. This method of partitioning the macroblock into motion compensated macroblocks of varying sizes is known as tree structure motion compensation. The choice of partition is important as it has a significant impact on compression performance. Generally for homogenous regions in a frame large partition size is appropriate, for non-homogenous areas small partition size may be beneficial.

Choosing a large partition size (16x16, 16x8, 8x16 and 8x8) means only few bits are required to signal the choice for motion vector (s) and types of partition; however, the motion compensated residual may contain a significant amount of energy in frame areas with high detail. Choosing a small partition size (8x4, 4x8 and 4x4) may give a lower-energy residual after motion compensation but requires a large number of bits to signal the motion vectors and choice of partition(s) [14].

The resolution of each chroma component in a macroblock (Cr and Cb) is half that of the luminance (luma) component. Each chroma block is partitioned in the same way as the luma component,

except that partition sizes have exactly half the horizontal and vertical resolutions. An 8x16 partition in luma corresponds to a 4x8 in chroma [14].

2.3.4 Sub-pixel motion vectors

Each partition in an inter-coded macroblock is predicted from an area of the same size in a reference picture. The offset between two areas, the motion vectors, have quarter pixel resolution for luma component. In sub-pixel motion vector prediction, the reference pictures do not have sub-pixel positions and so it is necessary to create them using interpolation from nearby image samples [14].

Sub-pixel motion compensation can provide significantly better compression performance than integer-pixel compensation, but adds to the complexity. Quarter pixel accuracy outperforms half-pixel accuracy [14]. In the luma component, the sub-pixel samples at half-pixels are generated first and interpolated from neighboring integer pixel samples using a six-tap finite impulse response filter. Figure 2.11 and 2.12 shows the six-tap filter weights [4] (1, –5, 20, 20, –5, 1)/32 is used to derive 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 save processing power.

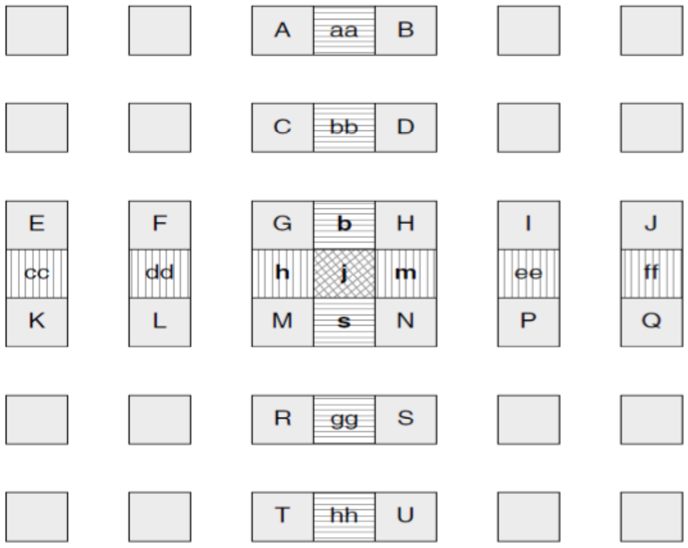


Figure 2.11 Interpolation of luma half-pel positions [4]

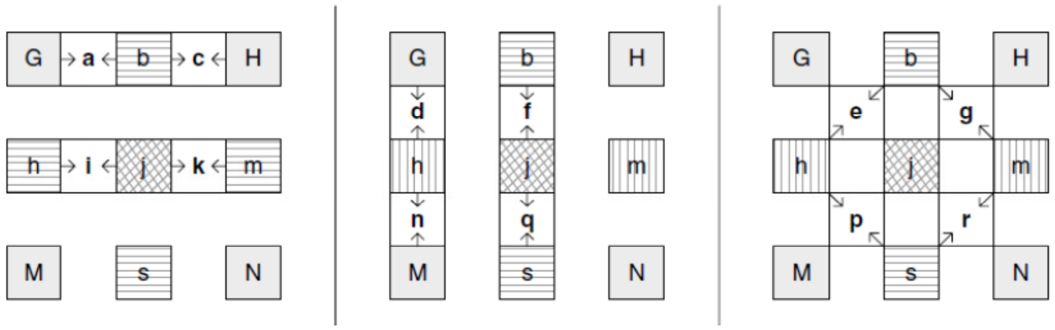
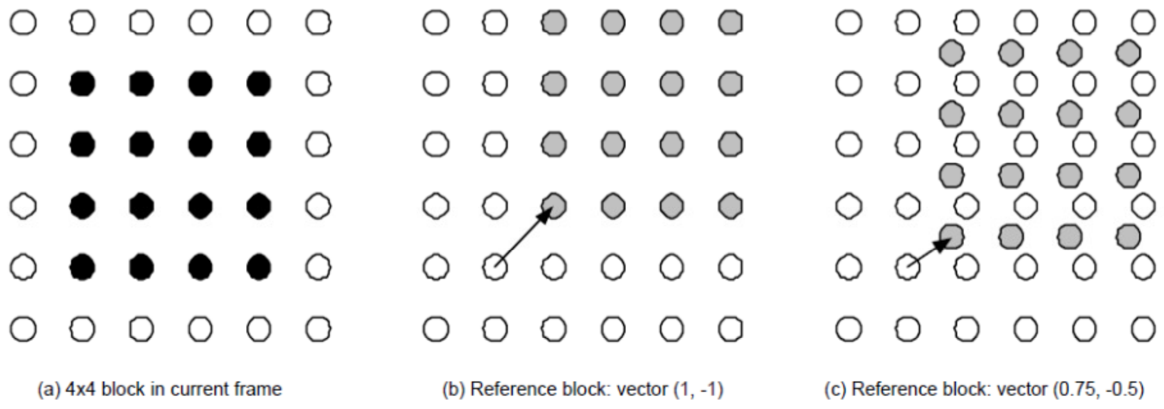


Figure 2.12 Interpolation of luma quarter-pel positions [4]

This means that each half-pixel sample is a weighed sum of six neighboring integer samples. Once all the half-pixel samples are available, each quarter pixel sample is produced using bilinear interpolation between neighboring half or-integer pixel samples [14]. Figure 2.13 shows an example of integer and sub-pixel prediction. A 4x4 sub-partition in the current frame can be predicted from neighboring region of the reference frame. If both the horizontal and vertical components of the motion vectors are integers, then the relevant samples in the reference block actually exist, as shown in Figure 2.13(b). If both or one of the vector components are fractional values, the prediction samples are to be interpolated from the adjacent samples in the reference frame.


(a) 4x4 block in current frame
(b) Reference block: vector (1, -1)
(c) Reference block: vector (0.75, -0.5)
Figure 2.13 Example of integer and sub-pixel predictions [14]

2.3.5 Transform and Quantization

A block of residual samples is transformed using a 4x4 or 8x8 integer transform, an approximate form of the discrete cosine transform (DCT) [4].The discrete cosine transform outputs a set of coefficients, each of which is a weighting value for a standard basis pattern. When combined, the weighted basis patterns re-create the block of residual samples.
The output coefficients of the transform are quantized. Quantization reduces the precision of the transform coefficients according to the quantization parameter (QP). Quantization parameter is the number by which each transformed coefficient is divided by an integer value.

2.3.6 In –Loop De-blocking Filter

H.264 employs an adaptive in-loop de-blocking filter after the inverse transform in the encoder and decoder respectively. The filter is applied to every decoded macroblock to reduce blocking distortion [4]. The de-blocking filter is applied after the inverse transform in the encoder before reconstructing and storing the macroblock for future predictions and in the decoder before reconstructing and displaying the macroblock. The filter smoothes block edges, improving the appearance of decoded frames. The filtered image is used for motion-compensation of future frames and this generally improves compression performance because the filtered image is a more faithful reproduction of the original frame than a blocky, unfiltered image.
The operation of de-blocking filter can be divided into three main steps i.e. filter strength computation, filter decision and filter implementation respectively.

2.3.6.1 Filter strength

The filter strength i.e. the amount of filtering is computed with the help of parameter boundary strength. The boundary strength of the filter depends on the current quantizer, macroblock type, motion vector, gradient of the image samples across the boundary and other parameters. The boundary strength is derived for each edge between 4x4 neighboring blocks and for each edge, boundary strength parameter is assigned an integer value 0 to 4. There are rules for selecting the boundary strength parameter. The filter is stronger at places where there is likely to be significant blocking distortion, such as the boundary of an intra coded macroblock or a boundary between blocks that contain coded coefficients [16].

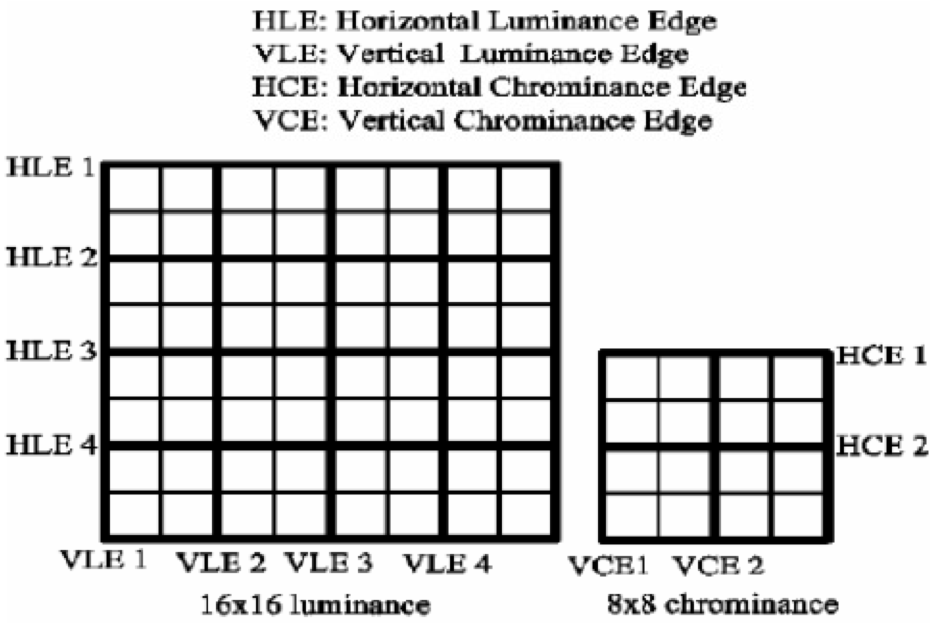

HLE: Horizontal Luminance Edge
VLE: Vertical Luminance Edge
HCE: Horizontal Chrominance Edge
VCE: Vertical Chrominance Edge
HLE 1
HLE 2
HLE 3
HLE 4
HCE 1
HCE 2
VLE 1
VLE 2
VLE 3
VLE 4
VCE1
VCE 2
16x16 luminance
8x8 chrominance

Figure 2.14 Edge filtering order in a macroblock [16]

Filtering is applied to the vertical or horizontal edges of blocks in a macroblock excluding edges on slice boundaries. Figure 2.14 shows the order of the filtering at a macroblock level is filtering the four vertical boundaries of the luma component in order VLE (vertical luminance edge) VLE1, VLE2, VLE3 and VLE4 are filtered. Then horizontal edges of the luminance component i.e. (horizontal luminance edge) HLE1, HLE2, HLE3 and HLE4 are filtered. Finally, vertical and horizontal edges (vertical chrominance edge) VCE1, VCE2 and horizontal component (horizontal chrominance edge) HCE1, HCE2 are filtered respectively. It is also possible for the filter to alter the filter strength or to disable the filter.

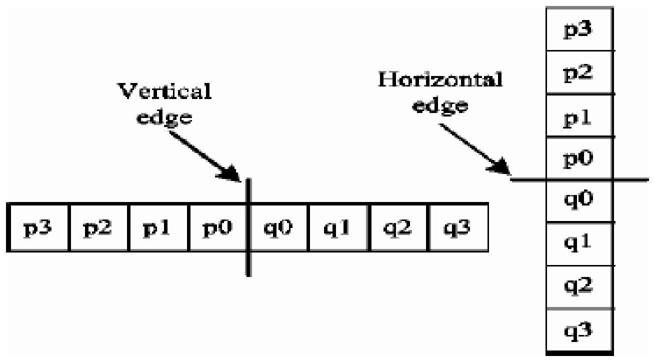


Figure 2.15 Pixels adjacent to horizontal and vertical boundaries [16]

Figure 2.15 shows the filtering operation affects three samples on either side of the boundary. The four samples on vertical edge or horizontal edge in adjacent blocks are p0, p1, p2, p3 and q0, q1, q2, q3 [16].

2.3.6.2 Filter implementation

H.264 employs deblocking process adaptively at the following three levels:

- At slice level – global filtering strength is adjusted to the individual characteristics of the video sequence.
- At block-edge level – deblocking filter decision is based on inter or intra prediction of the block, motion differences and presence of coded residuals in the two participating blocks.
- At sample level – it is important to distinguish between the blocking artifact and the true edges of the image. True edges should not be de blocked. Hence decision for deblocking at a sample level becomes important.

2.4 Entropy Coding

A coded H.264 stream or an H.264 file consists of a series of coded symbols. These symbols make up the syntax and include parameters, identifiers and delimiting codes, 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:

2.4.1 Fixed length code

A symbol is converted into a binary code with a specified length (n bits). Every word in the code has fixed length. In fixed length coding methods, 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.

2.4.2 Exponential-Golomb variable length code

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

2.4.3 Context adaptive variable length coding (CAVLC)

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.

After prediction, transformation and quantization, blocks are typically sparse, often containing mostly zeros. CAVLC uses run-level coding to compactly represent strings of zeros. The highest non-zero coefficients after block scan are often sequences of +/-1 and CAVLC signals the number of high frequency. The number of non-zero coefficients in neighboring blocks is correlated. The number of coefficients is encoded using a look-up table and the choice of look-up table depends on the number of non-zero coefficients in neighboring blocks. The level or magnitude of non-zero coefficients tends to be larger at the start of the scanned array, near the DC coefficients and smaller towards the higher frequencies.

2.4.4 Context adaptive binary arithmetic coding (CABAC)

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

a) Selecting probability models for each syntax element according to the element’s context.

b) Adapting probability estimates based on local statistics

c) Using arithmetic coding rather than variable-length coding.

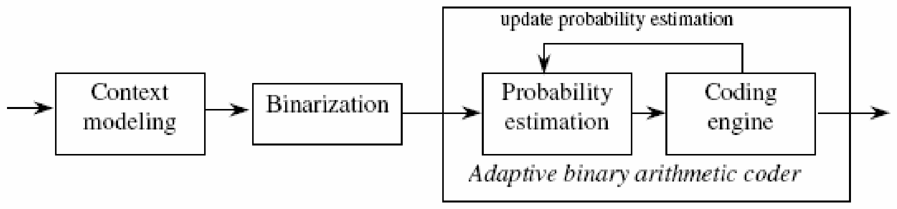


Figure 2.16 Block diagram for CABAC [4]

Figure 2.16 shows schematic for CABAC [4].Coding a data symbol involves the following stages:

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.

2.5 H.264 Decoder

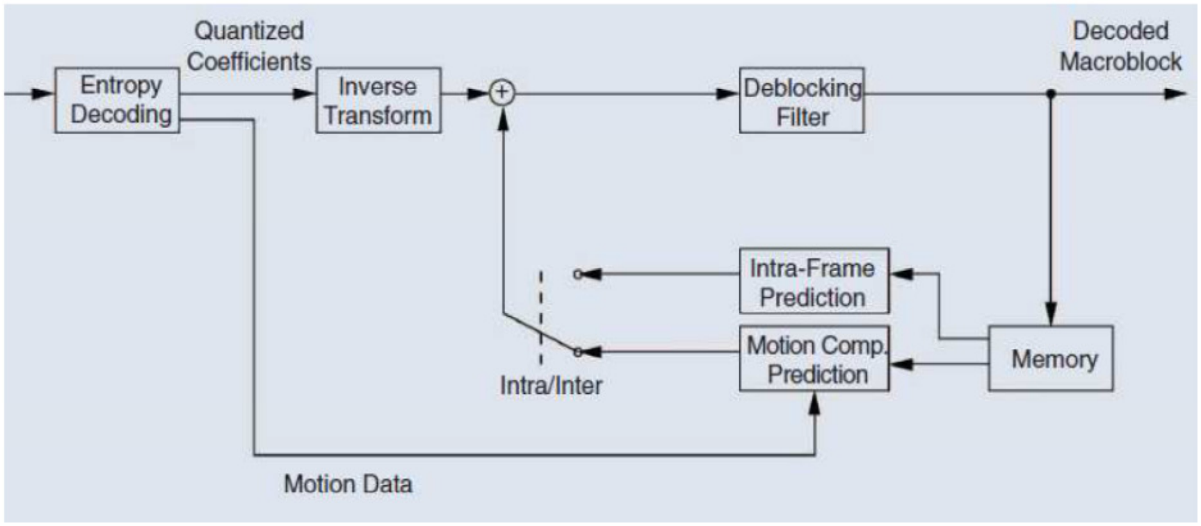


Figure 2.17 Basic coding structure H.264/AVC video decoder [4]

Decoding process is the exact opposite of the encoding process. A video decoder receives the compressed H.264 bit stream, decodes the syntax elements and extracts information such as quantized transform coefficients, prediction information etc. This data is used to recreate the video sequence. The quantized transform coefficients are multiplied by the quantization parameter. The quantization parameter is an integer value. After the transform coefficients are rescaled, the inverse transform combines the standard basis pattern, weighed by the rescaled coefficients, to re-create each block of residual data. These blocks are combined together to form the residual data macroblock. For each macroblock, the decoder performs prediction identical to the one created by the encoder. This is then added to the decoded residual data to reconstruct a decoded macroblock which can then be displayed as part of a video frame.

2.6 Summary

This chapter outlines the coding tools of H.264 codec. The intent of the H.264/AVC project was to create a standard capable of providing good video quality at substantially lower bit rates than previous standards (i.e., half or less the bit rate of MPEG-2, H.263, or MPEG-4 Part 2), without increasing the complexity of design so much that it would be impractical or excessively expensive to implement. The H.264 standard can be viewed as a "family of standards", the members of which are the profiles described in table 2.2.

CHAPTER 3

FAST MOTION ESTIMATION TECHNIQUES

3.1 Introduction

The main goals of the H.264/AVC standardization effort have been to enhance compression performance and provide a “network-friendly” video representation. The H.264/AVC standard uses variable block sizes and quarter-pixel motion compensation with multiple reference frames to achieve high coding efficiency [20]. It has motion compensation units in sizes of 16x16, 16x8, 8x16, 8x8 and sub 8x8. Such wide block choices improve coding efficiency at the cost of largely increased motion estimation time [21]. In H.264/AVC encoding, the most computationally critical part is motion estimation. The H.264 standard also has quarter-pixel motion vector accuracy as another of its important feature, which requires interpolation of pictures by a factor of four. This is done by a 2-tap bilinear filter and a 6-tap finite impulse response (FIR) filter as shown in figures 2.11 and 2.12 [21]. This increased accuracy of motion vectors and the subsequent coding gain is significant. On the other hand, the filtering process and the extra quarter-pixel motion estimation search demands substantial amount of computation. The computational complexity becomes even worse with larger search ranges or when multiple reference frames are used. Such high computational complexity is often a bottle-neck for real-time conversational applications. It also causes inconvenience for researchers during codec optimization or evaluation [22].

There are several techniques being researched for reducing the computational complexity in H.264 encoder. Complexity reduction algorithms propose improved and simplified fast motion estimation schemes to speed up the encoding process and enhance the rate-distortion performance. This chapter explains some of these algorithms in detail. Motion estimation is generally conducted in two steps the first is integer pixel motion estimation and the other is fractional pixel motion estimation around the position obtained by the integer pixel motion estimation [23]. To achieve more accurate motion description and higher compression efficiency, the H.264 [3] standard, the MPEG-1 standard, the MPEG-2 [39] standard and the MPEG-4 [2] standard use fractional pixel motion estimation while JVT uses ¼ and 1/8 pixel accuracies. Reference programs are primarily used for educational purposes. They are used by researchers as a benchmark because of their public availability. The Joint Model (JM) 17.2 for H.264/AVC [15] is one such publicly available program. JM reference software is optimized for coding efficiency rather than encoding speed. Rate distortion (R-D) performance is a paramount concern during the standardization process.

The implementation of the H.264/AVC in JM reference software has implemented the following motion estimation algorithms:

- Full search
- Uneven multi grid hexagon search (UMhexagonS)
- Enhanced predictive zonal search (EPZS)

3.2 Full search algorithm

Full search is based on the block matching algorithm for motion estimation. It is a way of locating matching blocks in a sequence of digital video frames for the purpose of motion estimation [24]. To remove inter frame redundancy and achieve high data compression, in a full search block matching algorithm (FBMA) , the current frame of a video sequence is divided into non overlapping square blocks of pixels of size N x N [24]. For each reference block in the current frame, full search searches for the best matched block within a search window size of (2W+N) x (2W+N) in the previous frame where, W stands for maximum allowed displacement and N represents pixels in row or column. A non-negative matching error function *Dp (i, j)* is defined over all the positions to be searched. The equation is as shown below:

$$D_P(i,j) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} |f_t(l + x, k + y) - f_{t-1}(l + i + x, k + j + y)|^p$$

$$p = 1 \text{ or } 2 \quad \text{and} - W \leq i; j \leq W$$

where, f_t (l, k) is the reference block of its upper left pixel at the coordinate (l, k) in the current frame, and f_{t-1} (l+I, k+j) is a candidate block of its upper left pixel at the coordinate (l+i,k+j) in the previous frame. The computations calculated in one complete measurement of D_p (i, j) are N^2 absolute values and $2N^2 – 1$ additions. The full search block matching algorithm which requires computation of the D_p (i, j)'s for all $(2W + 1)^2$ positions of candidate blocks in the search window. The full search is computationally intensive process, it needs $(2W + 1)^2$ N^2 absolute values or squaring, $(2W + 1)^2$ $(2N^2 - 1)$ additions, and $(2W + 1)^2$ comparisons for each reference block [24]. Full search, being computationally complex, has limited practical applications. It gives the best results when compared to other algorithms [24].

3.3 UMHEXAGONS search algorithm

JM (joint model) [15] adopted a fast motion estimation method including unsymmetrical multi-hexagonal search (UMHexagon). UMHexagonS algorithm has fast integer-pixel search and fast fractional pixel search for sub-pixel search. It reduces the motion estimation time by about 55% on average when compared with Full search. In addition, the method yields bit rate reduction up to 18 % when compared to full search in low complexity mode [22]. H.264 is the new video coding standard targeted for a wide range of applications from QCIF to HD. Designing robust fast motion estimation (FME) to meet such wide applications can be a challenging task. Unsymmetrical-cross multi hexagon-grid search (UMhexagonS) algorithm is also a kind of hierarchical motion search uses strategy. UMhexagonS is drawn from the basic idea that the movement in the horizontal direction is much heavier than that in the vertical direction and the distribution of motion vectors is zero centered. Figure 3.4 shows a typical search procedure in a search window with search range equals 16 (it is assumed the initial search point is (0, 0) vector here).

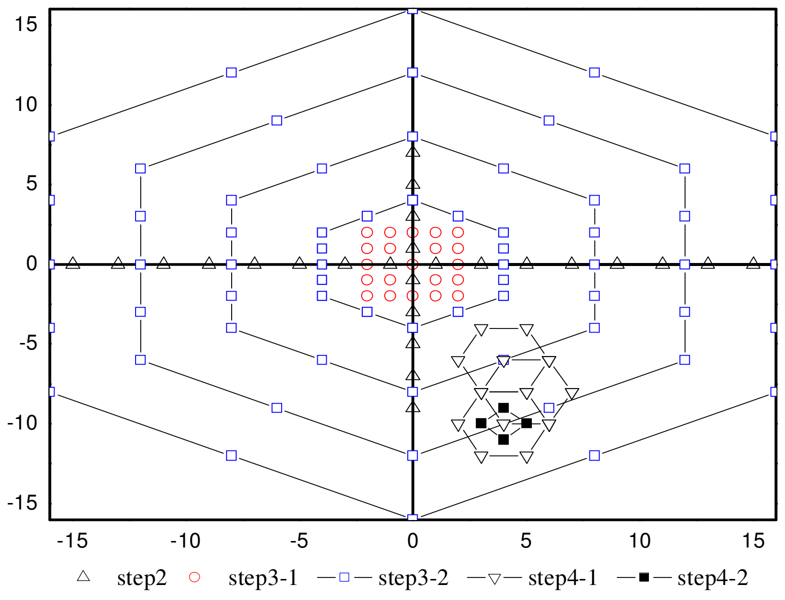


Figure 3.1 Search process of UMhexagonS algorithm [22]

It is called a hybrid method because it has four steps with different search patterns: 1) Initial search point prediction; 2) Unsymmetrical-cross search; 3) Uneven multi-hexagon grid search; 4) Extended hexagon based search.

The following four steps give a brief review of the fast integer sample search algorithm:

3.3.1 Initial search point prediction

Spatial median prediction, upper layer prediction, neighboring reference frame prediction, and temporal prediction are used to predict a current block's motion vector (MV).

Figure 3.5 shows a median predictor that is calculated using the adjacent blocks on the top, left and top-left (or top-right) of the current block.

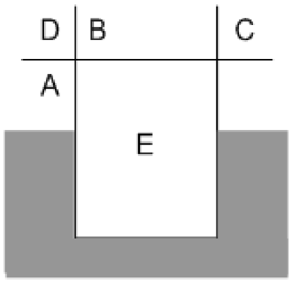


Figure 3.2 Reference block location for prediction of motion vectors [22]

The median predictor is used in median prediction of motion vectors and it is calculated as:

pred_mv = median (mv_A, mv_B, mv_C)

pred_mv = median predictor motion vector,

mv_A = motion vector of left block A

mv_B = motion vector of top block B

mv_C = motion vector of top-right block C

The predicted motion vector value has been defined in the editors proposed draft text modifications of the joint video specification [20]. If block A lies outside the group of blocks boundary, it is replaced by the (0, 0) motion vector; if block C lies outside the group of blocks boundary it is replaced by the motion vector of block D; when two blocks B and C lie outside, however, they are replaced by the motion vector of the third block. The prediction with the minimum cost among these candidates will be chosen as the initial search position of the next step search.

3.3.2 Unsymmetrical cross search

The unsymmetrical cross search across the x-y direction is based on the phenomenon that movement in the horizontal direction is much heavier than that in the vertical direction for natural picture sequences. As Figure 3.4 step 2 indicates an unsymmetrical-cross search with the horizontal search range equals W and vertical search range equals W/2. The unsymmetrical-cross search can be seen as a simple, but efficient prediction method to give an accurate starting search point for the next step [22]. In some special sequences with heavier vertical motion, the vertical search range can be expanded to W. The motion vector with the minimum cost will be chosen as the search center, i.e. the starting search point, of next search step.

3.3.3 Uneven multi-hexagon-grid search

Two sub-steps include a square search, with search range equal to two, which is carried out around the search center as shown in Figure 3.4. Step 3-1 shows square search around the search center. Then multi-hexagon grid search strategy is applied as shown in Figure 3.4 step 3-2. The uneven multi-hexagon-grid is used to handle the large and irregular motion cases. A sixteen point hexagon search pattern is used as basic search pattern in this thesis, just as shown in Figure 3.6; there are more search points in the horizontal direction than in the vertical direction.

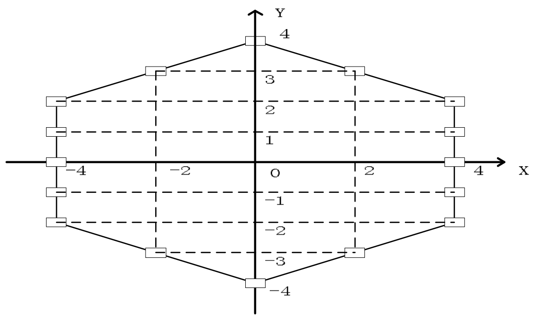


Figure 3.3 Sixteen points hexagon pattern (16-HP) used in UMhexagonS [22]

An uneven multi hexagon grid is constructed with scale factors ranging from 1 to W/4. The search process starts from the inner hexagon to the outer hexagon [22]. The best motion vector derived in this step will be chosen as the search center of the next search step.

3.3.4 Extended hexagon-based search

The two sub-steps include a hexagon search pattern and a diamond search pattern. The multi-grid search may obtain optimum accuracy of motion vectors according to the distance between the search window center and the search point. If the optimum motion vector in the previous steps locates in the outer concentric area, the search result has relatively low accuracy. Thus UMhexagonS uses the extended hexagon based search (EHS) algorithm as the center biased search algorithm. Thus, when switching from a larger to a smaller size of hexagon, as shown in Figure 3.4 step-4, the search continues until the minimum block distortion point is the center of the newly formed hexagon.

Center Biased Fractional Pixel Search algorithm for fractional pixel search: The CBFPS algorithm is much faster and more accurate especially in case of 1/8 pixel search. Figure 3.6 shows the implementation of the CBFPS algorithm, the following steps describe the whole algorithm:

Step 1) Predict the motion vector of the current block by using equations (1) and (2). The predicted motion vector is (pred_x,pred_y)

$$frac\_pred\_mv = (pred\_mv - mv)\% \beta \qquad (2)$$

where, mv is the integer motion vector of a block; $pred\_mv$ is defined as the fractional pixel unit; If pixel case is 1/8, then scale factor β is equal to 8. If pixel case is ¼, scale factor β is equal to 4.

Step 2) Cost of the original search center (0, 0) and (pred_x,pred_y) are compared. The point with the minimum matching error is chosen as the search center.

Step 3) If the MBD (Minimum Block Distortion) point is located at the center, go to step 4; otherwise choose the MBD point in this step as the center of next search, then iterate this search step.

Step 4) Choose the MBD point as the motion vector.

Early termination scheme is also applied during the search process. It is based on detection of zero blocks.

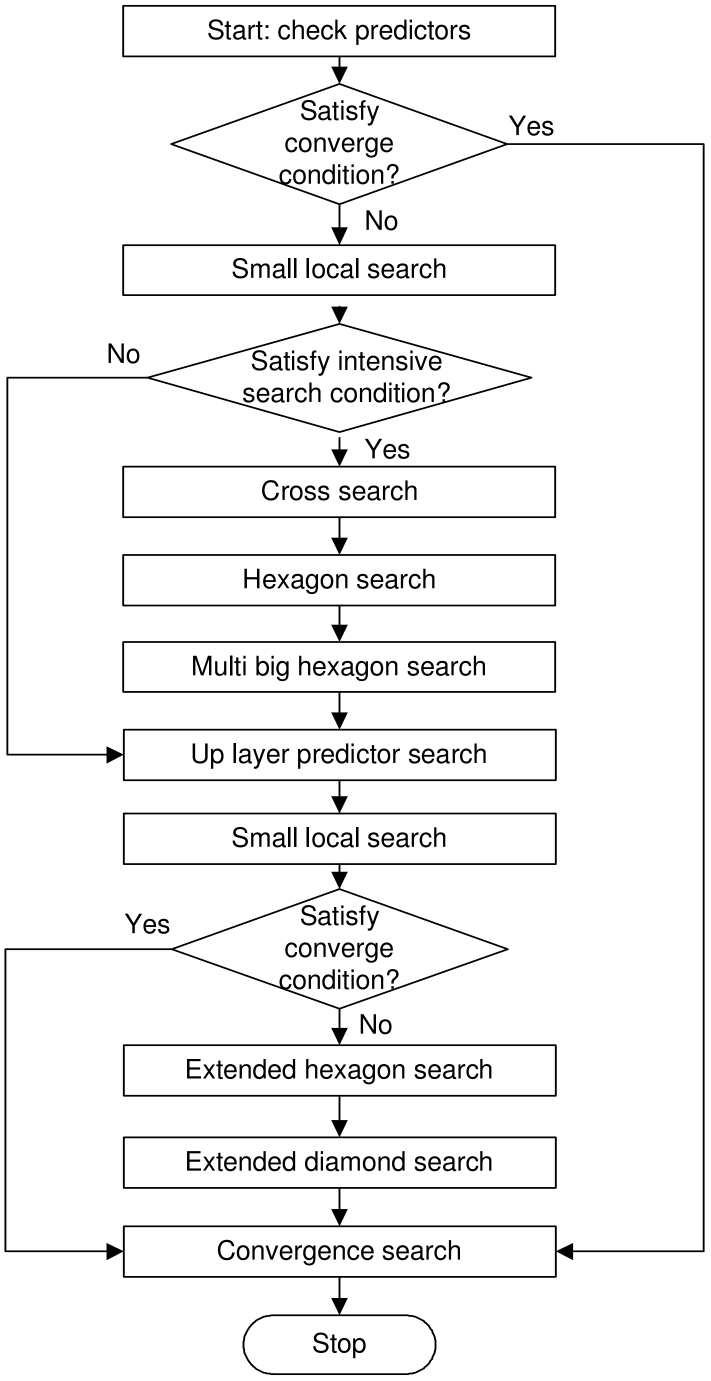


Figure 3.4 Flow chart illustrating example for UMhexagonS [22]

The early termination scheme is based on three cost prediction modes, namely, median prediction, upper layer prediction, and neighboring reference frame prediction. Following the integer-pixel search, a full fractional pixel search is performed for the 16×16, 16×8 and 8×16 blocks. A fast sub-pixel search is also performed only for other sub-partition blocks.

3.4 Enhanced predictive zonal search (EPZS)

Motion estimation (ME) techniques for video encoding took an entirely different direction with the emergence of zonal algorithms [23]. These algorithms helped reduce the tremendous computational overhead of motion estimation that has occurred in a video coding system, with little loss of video quality. Enhanced predictive zonal search (EPZS) further improves upon these algorithms by considering additional sets of predictors, improved early termination thresholds and simplifying the search pattern. It achieves better output quality and also reduces the complexity of motion estimation process [23].

The three important steps in EPZS are:

3.4.1 Predictor selection

The most important and key feature for zonal algorithms performance is prediction selection. The predictors are defined and examined within a set. Afterwards it is necessary to select the best candidate among the set and perform a local search with a predefined pattern for refining the prediction. It is observed that the motion vectors are highly correlated with the motion vectors of temporally and spatially adjacent blocks that have been previously calculated. These motion vectors can be considered as initial predictors [23]. The median predictor, calculated from the median value of the motion vectors of the three adjacent blocks, the blocks on the top, left and top-right from the current position, appears as the optimal predictor candidate. Thus, the median predictor candidate is considered as a part of Subset A. Subset B consists of the (0, 0) motion vector and all the other predictor candidates. EPZS technique also uses subset C, which includes other possible candidates on such candidate is the *accelerator* *motion vector* in Figure 3.2 [23]. It is the differentially increased/decreased motion vector that is used in the motion vector of the previous frame and also the previous frame before that. The concept behind such a motion vector is that a block may not be following constant velocity but accelerating [23].

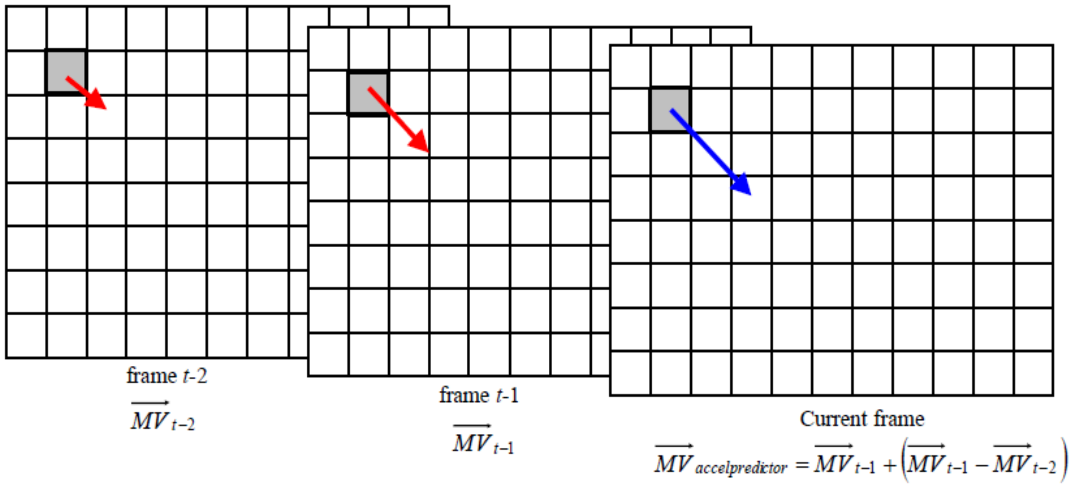

frame t-2
$\overrightarrow{MV}_{t-2}$
frame t-1
$\overrightarrow{MV}_{t-1}$
Current frame
$\overrightarrow{MV}_{accelpredictor} = \overrightarrow{MV}_{t-1} + \left(\overrightarrow{MV}_{t-1} - \overrightarrow{MV}_{t-2}\right)$

Figure 3.5 Acceleration information as a motion vector predictor [23]

It is also possible that the current block may not only be correlated spatially within the current frame or temporally correlated from a co-located block in the previous frame but can also be highly correlated with adjacent blocks to the co-located block in the previous frame. In particular, it is quite possible that it has very fast motion, which may have caused it to become highly correlated with the co-located blocks from a previous frame and possibly be a better candidate for prediction phase. These additional predictors constitute a final third subset, called subset C.

3.4.2 Improved adaptive threshold

EPZS uses an improved adaptive threshold technique, in which it calculates the thresholding parameters by considering minimum SAD (sum of absolute differences) of the spatially located three adjacent blocks and the co-located block in the previous frame.

Calculating threshold parameter by considering the minimum of all these candidates significantly reduces the possibility of error and of erroneous inadequate early termination. The thresholding parameter could in general be seen as:

$$T_k = a_k + \min(MSAD_1, MSAD2, \ldots, MSADn) + b_k$$

where,

T_k = threshold

a_k = fixed parameter; b_k = fixed parameter

MSAD = minimum SAD

3.4.3 Simplified search pattern

EPZS uses the small diamond or square search pattern, which simplifies the algorithmic design and implementation complexity. Figure 3.3 shows search patterns used in EPZS [23]. These methods can significantly improve the accuracy of the prediction and be quite beneficial for certain applications and especially hardware designs. The possible patterns selected for search can be a first order diamond pattern.

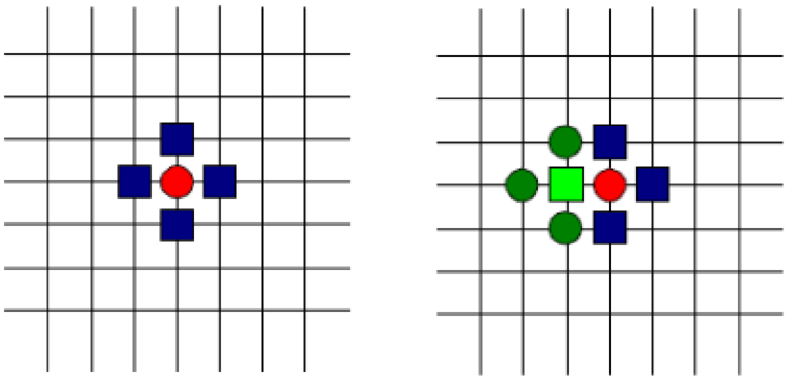


Figure 3.6 Small diamond pattern used in EPZS [23]

Another possible pattern can be the 8-point square pattern around the current minimum. Both patterns are very simple and easy to implement in software and hardware. The EPZS implementation using square search pattern is called as EPZS^2 (EPZS square).

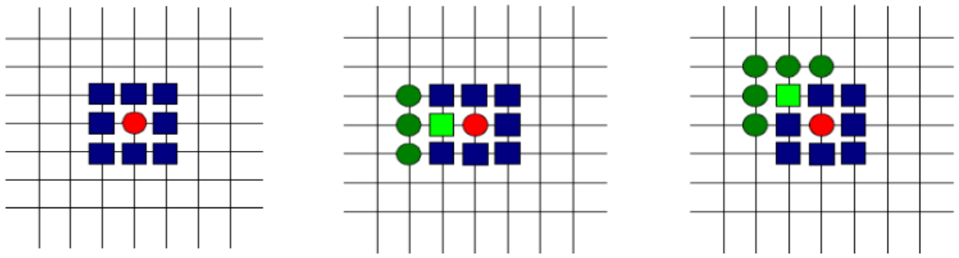


Figure 3.7 EPZS using the circular/square pattern [23]

3.5 Summary

This chapter briefly describes various motion estimation algorithms implemented in JM software for H.264/AVC. It describes full search, EPZS and UMhexagonS algorithms. The next chapter discusses the complexity reduction algorithm used in this thesis.

CHAPTER 4

COMPLEXITY REDUCING FAST ADAPTIVE TERMINATION ALGORITHM

4.1 Introduction

As discussed in previously, the H.264/AVC [4] standard achieves better coding efficiency than the previous coding standards. It has more advanced coding features such as quarter pixel motion compensation, adaptive deblocking filter and variable block size mode selection. In order to provide coding accuracy, one macroblock is partitioned into several block sized including 16x16,16x8,8x16,8x8,8x4,4x8 and 4x4, where small size blocks correspond to detailed regions or large motion areas while large size blocks correspond to homogeneous regions or relatively stationary motion areas [30]. In implementation of the H.264/AVC codec that is part of JM17.2 [15], to find mode for one macroblock in a P frame, a P16x16 is first checked followed by P16x8, P8x16, P8x8 and so on. For each mode, a rate distortion-based mode selection is carried out to obtain the best candidate mode which has the minimum rate distortion cost. The rate distortion cost is calculated for all the modes and then the best one with the least cost is selected, this process for finding the best mode increases the computational complexity [30]. The increase in computational complexity poses implementation issues on portable devices with limited battery-life. This thesis adopts a fast adaptive early termination mode decision algorithm [30]. It combines three different useful techniques for fast motion estimation as fast mode prediction, adaptive rate-distortion thresholds and homogeneity detection.

4.2 Fast adaptive early termination mode selection

4.2.1 Fast mode prediction

H.264/AVC video coding is performed on macro blocks from up-left to the right-bottom direction. The fast adaptive early termination mode selection approach is quick and correct mode prediction and avoid the large amount of computation associated with all checking modes. Macro blocks adjacent to each other in same frame may have same characteristics like similar motion or similar detailed regions. Figure 4.1 illustrates that the current macroblock X may have similar characteristics with its neighboring macro blocks from A through H.

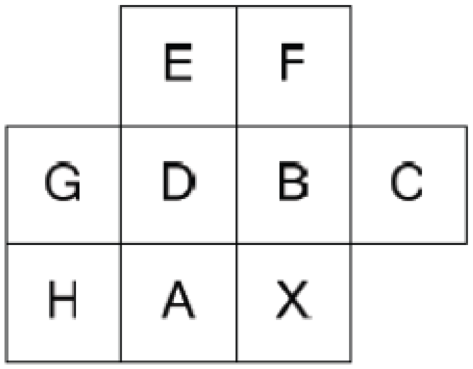


Figure 4.1 Spatial neighboring macroblocks [30]

If most neighboring macro blocks have the skip mode, it means that the current macroblock also has more chance to have skip mode. There also exists temporal similarity between a current macroblock and the collocated macroblock PX in the previous frame and its neighbors as shown in Figure 4.2. A mode histogram from spatial and temporal macroblocks is obtained. After obtaining the mode histogram, the index corresponding to the maximum value in the mode histogram is selected as the best mode. The average rate distortion cost of each neighboring macroblock corresponding to the best predicted mode is selected as the prediction cost for the current macroblock. For example, in Figure 4.1, if macroblocks A,B,C,D,E and P1,PX, P3 have P8x8 modes, which would also be the maximum value in the mode histogram, then P8x8 mode is the best predicted mode and the average rate distortion cost among A,B,C,D,E and P1,PX,P3 is used as the predicted cost for the current macroblock.

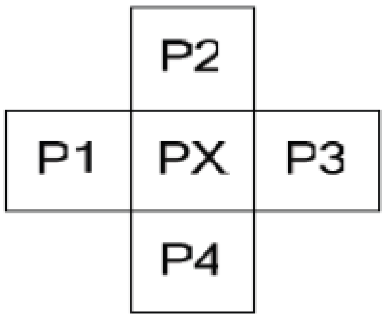


Figure 4.2 Temporal neighboring macroblocks [30]

4.2.2 Adaptive rate distortion threshold

Texture information obtained from current macroblock can be used to decide whether this macroblock is homogeneous or not. If a macroblock is homogeneous, only large block sizes can be used and small block sizes can be skipped, which helps to reduce the computation complexity. It is helpful because the motion estimation computation amount is unequal for different block sizes smaller block sizes like 8x8 block requires four times the motion estimation computation as compared to that of 16x16 block. Due to the importance of homogeneous regions they can be checked before considering small block sizes. This thesis adopts a method to identify homogeneous regions by calculating adaptive threshold levels which use the information of neighboring blocks around a current macro block. The adaptive threshold uses a modulator β, for reliable comparison based on the predicted cost to establish trade-off between complexity and accuracy. Assuming the minimum rate distortion cost for a current macroblock is

RD_{best}, and the predicted rate distortion cost according to the spatial and temporal correlations is *RD_{pred}*, the rate distortion threshold *RD_{thresh}* for early termination is defined according to the formula given by $RD_{thresh} = (1 + \beta) X RD_{pred}$ where,

β = modulation coefficient,

RD_{thresh} = the rate distortion threshold,

RD_{pred} = the rate distortion prediction

If the rate-distortion cost of the current macroblock for a specific mode is less than *RD_{thresh}*, it will stop checking the other modes and exit the current macro block. The Modulation coefficient *β* value should be appropriately assigned,following explains,how the value is assigned. Assuming the minimum SAD for a current macroblock is *SAD_{best}* and the SAD for exiting is *SAD_{exit}*, the threshold *SAD_{thresh}* is defined to meet the requirement *$SAD_{best} \leq SAD_{exit} \leq SAD_{thresh}$*. After motion estimation, the residual information is used for the discrete cosine transform (DCT) transformation using the following formula:

$$F(u,v) = [\frac{2}{N}]^{\frac{1}{2}} [\frac{2}{M}]^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} diff(i,j) C(i) C(j) \cos[\frac{\pi u}{2N} (2i+1)] \cos[\frac{\pi v}{2M} (2j+1)]$$

where

$$C(i) = \begin{cases} \frac{1}{\sqrt{2}}; i = 0 \\ 1; i \neq 0 \end{cases}$$

diff(I,j) denotes the pixel difference between the value in the current macroblock at (i, j) position and corresponding pixel value in the best matched block in the reference frame.

Next, to quantize the DCT coefficients, *Q_{step}* is used. If the following condition $| F(u, v) / Q_{step} | < 1$ is satisfied. Thus,

$$|F_1(u,v) - F_2(u,v)| \leq Q_{step}$$

$$\rightarrow |F_1(u,v) - F_2(u,v)| = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} |\mathrm{Diff}_1^B(i,j) - \mathrm{Diff}_2^B(i,j)| C(i)C(j) \cos\left[\frac{\pi u}{2N}(2i+1)\right] \cos\left[\frac{\pi v}{2M}(2j+1)\right] \leq Q_{step}$$

$$= \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} |\mathrm{Diff}_1^B(i,j)| - \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left|\mathrm{Diff}_2^B(i,j)\right| \leq Q_{step} * \left(\frac{N}{2}\right)^{\frac{1}{2}} \left(\frac{M}{2}\right)^{\frac{1}{2}}$$

where, F_1 and F_2 are predicted and best outcomes, respectively. Denoting,

$$SAD_{thresh} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left|\mathrm{Diff}_1^B(i,j)\right|$$

$$SAD_{best} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \left|\mathrm{Diff}_2^B(i,j)\right|$$

By assuming $SAD_{pred} \approx SAD_{best,}$ the modulator *β* can be written as follows:

$$SAD_{thresh} - SAD_{best} \leq Q_{step} * \left(\frac{N}{2}\right)^{\frac{1}{2}} \left(\frac{M}{2}\right)^{\frac{1}{2}}$$

$$\Rightarrow SAD_{pred}(1+\beta) - SAD_{best} \leq Q_{step} * \left(\frac{N}{2}\right)^{\frac{1}{2}} \left(\frac{M}{2}\right)^{\frac{1}{2}}$$

$$\Rightarrow \beta \leq \frac{Q_{step} * (N/2)^{\frac{1}{2}} (M/2)^{\frac{1}{2}}}{SAD_{pred}}$$

The modulation coefficient *β* depends on two factors; the quantization step and the block size. Furthermore, the SAD threshold for a current macroblock can be obtained via $SAD_{thresh} = SAD_{pred} \times (1 + \beta)$. Similarly, replace SAD with RD_{cost}, the following formula is obtained $RD_{thresh} = RD_{pred} \times (1 + \beta)$.The threshold values are adaptive as they depend on the predicted rate distortion cost derived from spatial and temporal correlations.

4.2.3 Homogeneity detection

Smaller block sizes require more computation as compared with larger size blocks since such blocks often have more detailed regions or non-homogenous regions. Before checking for the smaller block sizes P4x4, P4x8 and P8x4, it is necessary to check whether large block sizes like P8x8, P8x16, P16x8 and P16x16 are homogeneous or not. Texture information of different blocks can be used to find if it reflects similar spatial properties. There are many techniques [25, 26] to detect texture homogeneity in an image. In [25], it is shown that the statistics consisting of standard deviation, variation and skew are effective for detecting homogenous regions. In [26], textures are modeled by Gaussian Markov random fields. These techniques mentioned in [25, 26] are effective, but computationally intensive, and thus are not suitable for real-time implementation. The proposed fast adaptive early termination algorithm uses edge detection based on homogeneous regions. This technique has been used for detecting homogeneous regions for fast intra mode prediction in [27] and this information obtained from intra mode detection is used as a guideline for selecting inter mode in [28]. However, the problem with these approaches lies in the fixed thresholds. Here adapting a method for adaptive threshold obtained from spatial neighboring blocks used to detect homogeneous textures. The proposed algorithm uses edge detection mentioned in [27]. First an edge map is created for each frame using Sobel operator [27]. For each pixel $p_{m,n}$, an edge vector is obtained $D_{m,n}(dx_{m,n}, dy_{m,n})$

$$dx_{m,n} = p_{m-1,n+1} + 2 * p_{m,n+1} + p_{m+1,n+1} - p_{m-1,n-1} - 2 * p_{m,n-1} - p_{m+1,n-1} \qquad 4.1$$

$$dy_{m,n} = p_{m+1,n-1} + 2 * p_{m+1,n} + p_{m+1,n+1} - p_{m-1,n-1} - 2 * p_{m-1,n} - p_{m-1,n+1} \qquad 4.2$$

Here $dx_{m,n}$ and $dy_{m,n}$ represent the differences in the vertical and horizontal directions respectively. The amplitude Amp (D (m, n)) of the edge vector is given by,

$$\text{Amp}(D(m,n)) = |dx_{m,n}| + |dy_{m,n}| \qquad 4.3$$

A homogeneous region is detected by comparing the summation of the amplitudes of edge vectors over one region with predefined threshold values [30]. In the proposed algorithm, such thresholds are made adaptive depending on the amplitude of left block, up blocks and mode information. The adaptive threshold is determined as per the following four cases:

Case 1: If the left block and the up block are both P8x8

$$Threshold = \min \begin{cases} \sum_{(m,n)\in\, leftBlock} Amp(D(m,n)) \\ \sum_{(m,n)\in\, upBlock} Amp(D(m,n)) \end{cases}$$

Case 2: If the left block is P8x8 and up block is not P8x8, then

$$\text{Threshold} = \textstyle\sum_{(m,n)\in\, left\ block} Amp\ (D\ (m,n))$$

Case 3: If the left block is not P8x8 and up block is P8x8, then

$$\text{Threshold} = \textstyle\sum_{(m,n)\in\, up\ block} Amp\ (D\ (m,n))$$

Case 4: If the left block is not P8x8 and up block is not P8x8, then

$$Threshold = \frac{\begin{pmatrix} \sum_{(m,n)\in\, leftBlock} Amp(D(m,n)) & + \\ \sum_{(m,n)\in\, leftBlock} Amp(D(m,n)) & \end{pmatrix}}{10}$$

4.3 Proposed fast adaptive mode selection algorithm

The proposed fast adaptive mode selection algorithm is explained here. Figure 4.3 shows the flow chart of the proposed fast adpative early termination algorithm.

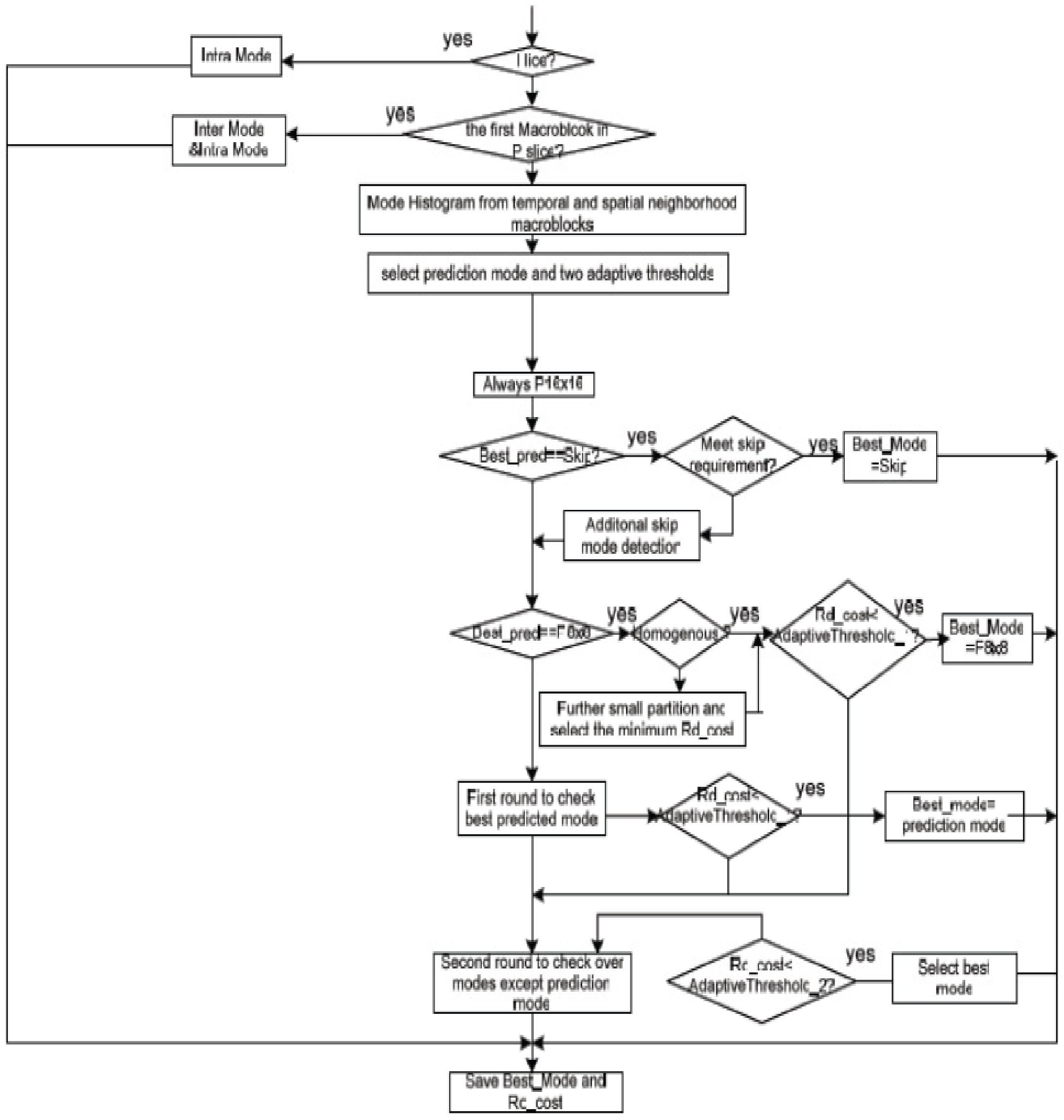

yes
Intra Mode
I lice?
yes
Inter Mode &Intra Mode
the first Macroblcok in P slice?
Mode Histogram from temporal and spatial neighborhood macroblocks
select prediction mode and two adaptive thresholds
Always P16x16
Best_pred==Skip?
yes
Meet skip requirement?
yes
Best_Mode =Skip
Additonal skip mode detection
Best_pred==P8x8
yes
Homogenous?
yes
Rd_cost< AdaptiveThreshold_1?
yes
Best_Mode =P8x8
Further small partition and select the minimum Rd_cost
First round to check best predicted mode
Rd_cost< AdaptiveThreshold_1?
yes
Best_mode= prediction mode
Second round to check over modes except prediction mode
Rd_cost< AdaptiveThreshold_2?
yes
Select best mode
Save Best_Mode and Rd_cost

Figure 4.3 Fast adaptive early termination (FAT) algorithm flowchart [30]

4.3.1 Fast adaptive early termination for mode decision algorithm

The Fast adaptive early termination for mode decision algorithm is explained as follows:

Step 1) If a current macroblock belongs to I slice, check I4x4 and I16x16 mode, go to step 10; otherwise, go to step 2.

Step 2) If a current macroblock belongs to the first macroblock in the P slice, check all inter modes and intra modes, go to Step 10; otherwise go to step 4.

Step 3) Compute the mode histogram from neighboring spatial and temporal macroblocks; go to step 4.

Step 4) Select the prediction mode as the index corresponding to the maximum value in the mode histogram and obtain two available adaptive thresholds*: Adaptive Threshold I* and *Adaptive Threshold II.* Go to step 5.

Step 5) Always check over the P16x16 mode and check the conditions of skip mode, if the conditions of a skip mode are satisfied. Go to step 10; otherwise go to step 6.

Step 6) If all the left, up, up-left and up-right macroblocks have skip modes, then check the skip mode against *Adaptive Threshold I*. If the rate distortion is less than *Adaptive Threshold I* the current macroblock is labeled as skip mode and go to step 10; otherwise, go to step 7.

Step 7) First round check over the predicted mode; if the predicted mode is P8x8, go to step 8; otherwise, check the rate distortion cost of the predicted mode against *Adaptive Threshold I*. If the rate distortion of P8x8 is less than Adaptive Threshold I, go to step 10; otherwise, go to step 8.

Step 8) If a current 8x8 block is homogeneous, no further partition is required. Otherwise, further partitioning into smaller blocks 8x4, 4x8 and 4x4 is performed. If the rate distortion of P8x8 is less than *Adaptive Threshold I*; go to step 10; otherwise, go to step 9.

Step 9) Second round check over the remaining modes against *Adaptive Threshold II* : If the rate distortion is less than *Adaptive Threshold II* , go to step 10; otherwise, continue checking all the remaining modes. Go to step 10.

Step 10) Save the best mode and rate distortion cost.

To summarize the proposed fast adaptive mode selection algorithm includes the following:

1) Skip mode detection: The algorithm has additional skip mode detection. If all neighboring macro blocks have skip mode and the rate distortion of skip modes for a current macroblock is less than the adaptive threshold then skip mode is the best mode for that block.
2) Adaptive rate distortion threshold: An adaptive rate distortion cost based on neighboring macro blocks through the parameter *β,* reflecting the quantization parameter.
3) Two round checks for early termination during mode selection, where the first round check is done over predicted modes in comparison with *Adaptive Threshold I*, while the second round check is done based on other modes in comparison with *Adaptive Threshold II*.
4) An adaptive threshold to check homogenity of video content.

4.4 Experimental results

In order to evaluate the proposed research, JM 17.2 [15] provided by JVT (Joint Video Team) under H.264/AVC baseline profile, using the fast motion estimation search called hybrid unsymmetrical cross multi-hexagon-grid [22]. In the original JM17.2 implementation, all modes including intra modes are checked and then the best mode is selected with the minimum rate distortion cost. The following encoding specifications are used in experiments listed below:

1) Motion estimation search range was set to 32 pixels for both CIF and QCIF.

2) The Hadamard transform was used.

3) The reference frame number was 5.

4) The CABAC was enabled.

5) The GOP structure is IPPP (No B frames)

6) The number of frames in the sequence is 30.

The proposed FAT mode selection algorithm was implemented based on JM17.2, and the platform used was a 2.1GHz Intel Core 2 Duo with a 4GB RAM. The mode histogram was computed based on spatial/temporal neighbors and two adaptive thresholds were computed based by $RD_{thresh} = RD_{pred} \times (1-8 \times \beta)$ and $RD_{thresh} = RD_{pred} \times (1+10x\ \beta)$. The proposed algorithm was extensively tested using a wide range of QCIF and CIF format video sequences ranging from spatial content to motion content variations. The experiments were carried out on these sequences with different quantization parameters of QP = 22, 27, 32, 37.

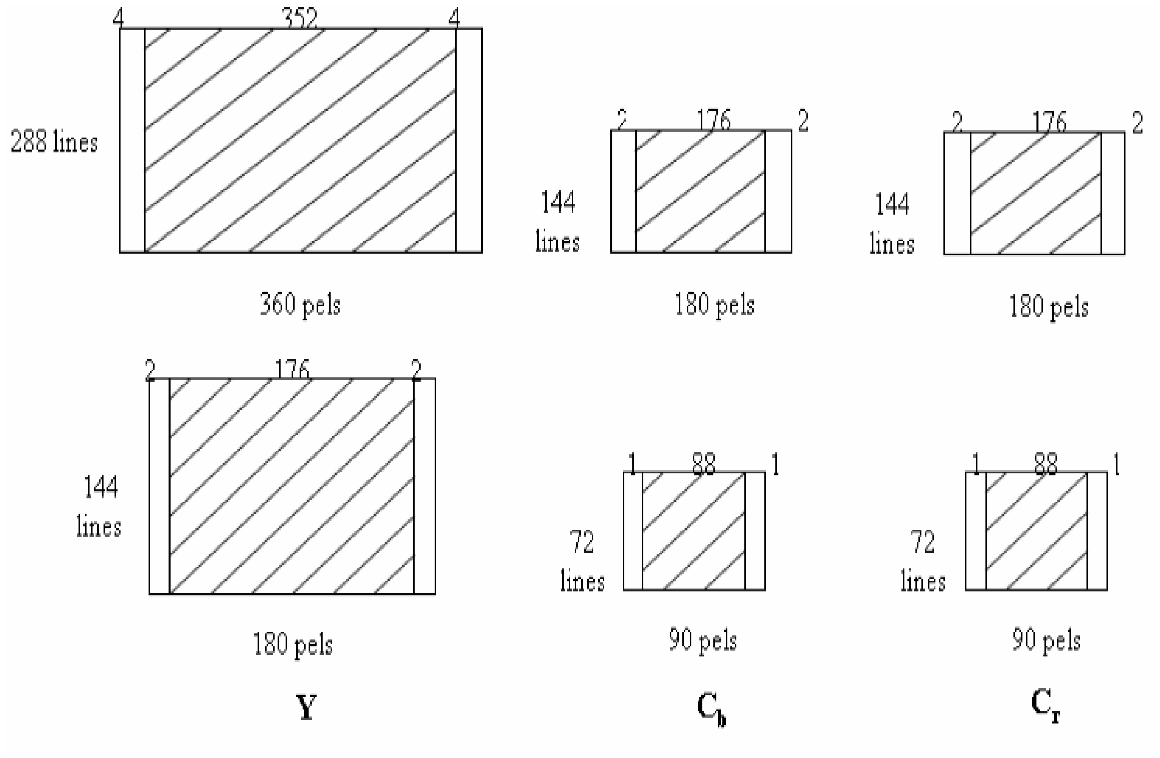


Figure 4.4 4:2:0 format for CIF and QCIF [35]

The simulations for test sequences of bridge-close, bridge-far, akiyo, news, hall monitor, silent, coastguard, foreman, container and bus with QCIF (176x144) and CIF (352x288) resolutions was carried out.

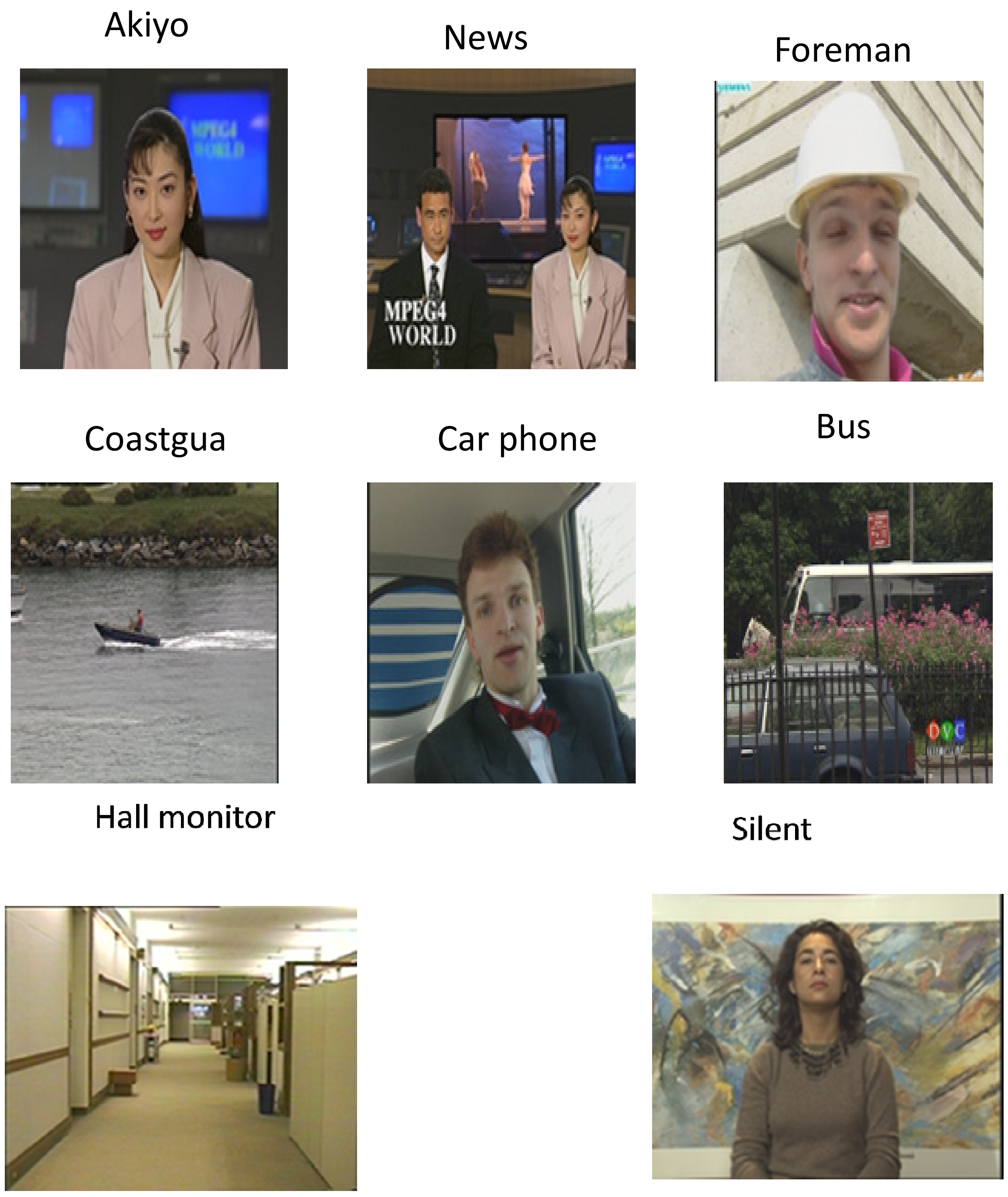


Figure 4.5 Test sequences for CIF and QCIF [29]

Tables 4.1 through 4.4 show the simulation results of QCIF and CIF video sequences at various QP. From this table it can be observed that a maximum of 67.8 % of encoding time was saved in one of the cases with negligible change in PSNR, SSIM and bit-rate.

Table 4.1 Simulation results for QCIF video sequences at QP = 22, 27

Sequence (QCIF)	QP = 22				QP = 27			
	ΔT (%)	ΔPSNR (dB)	Δ Bit-rate (%)	Δ SSIM (%)	ΔT (%)	ΔPSNR (dB)	Δ Bit-rate (%)	Δ SSIM (%)
coastguard	-42.113	-0.043	0.756	0.0309	-65.244	-0.118	1.035	0.151
foreman	-38.693	-0.005	3.933	0.1426	-40.779	-0.414	5.041	0.1347
news	-30.309	-0.002	-0.369	0.0507	-40.275	-0.243	1.311	0.0702
silent	-42.223	-0.176	2.999	0.0407	-39.686	0.032	5.407	0.0416
akiyo	-42.222	-0.223	-0.491	0.0303	-43.159	-0.051	4.49	0.0307
container	-47.233	-0.315	6.33	0.0517	-35.767	-0.28	6.804	0.0528
carphone	-37.888	-0.12	5.023	0.0812	-35.542	-0.122	5.703	0.1029
hall	-40.454	-0.043	-0.705	0.0305	-45.245	0.01	0.936	0

Table 4.2 Simulation results for QCIF video sequences at QP = 32, 37

Sequence (QCIF)	QP = 32				QP = 37			
	ΔT (%)	ΔPSNR (dB)	Δ Bit-rate (%)	Δ SSIM (%)	ΔT (%)	ΔPSNR (dB)	Δ Bit-rate (%)	Δ SSIM (%)
coastguard	-66.548	-0.118	-2.689	0.440	-39.1	-0.56	-5.58	1.22
foreman	-67.602	-0.414	13.924	0.330	-40.785	-0.69	13.924	0.547
news	-67.884	-0.243	0.21	0.148	-39.419	-0.3	0.403	0.33
silent	-64.98	0.032	6.111	0.108	-34.95	-0.29	9.039	0.106
akiyo	-67.228	-0.051	-0.039	0.0740	-37.816	-0.21	1.694	0.0890
container	-65.772	-0.28	6.176	0.1512	-38.48	-0.52	6.351	0.2560
carphone	-67.128	-0.122	15.28	0.379	-38.617	-0.74	14.775	0.515
hall	-69.554	0.01	0.205	0.0312	-44.847	-0.09	1.306	0.064

Tables 4.3 through 4.5 show the simulation results of CIF videos at various QP values. From these tables it can be observed that, a maximum of 68.6% of encoding time was saved in one of the cases with negligible PSNR, SSIM and bit rate.

Table 4.3 Simulation results for CIF sequences for QP = 22, 27

Sequence (CIF)	QP = 32				QP = 37			
	ΔT (%)	ΔPSNR (dB)	Δ Bit-rate (%)	Δ SSIM (%)	ΔT (%)	ΔPSNR (dB)	Δ Bit-rate (%)	Δ SSIM (%)
bridge-close	-38.3	-0.134	-0.134	0.073	-38.225	-0.19	0.302	0.043
bridge-far	-37.954	-0.08	-0.08	-0.63	-32.7	0	0	-0.851
coastguard	-56.69	0.185	23.36	0.071	-54.221	-0.93	12.923	1.461
foreman	-45.41	0.899	36.68	0.462	-38.67	0.75	8.425	0.82
highway	-40.52	-0.726	13.069	0.531	-33.35	-0.77	13.291	0.89
container	-68.6	-0.423	5.5	-0.180	-67.123	0.635	-6.209	-0.187

Table 4.4 Simulation results for CIF videos at QP = 32, 37

Sequence (CIF)	QP = 32				QP = 37			
	ΔT (%)	ΔPSNR (dB)	Δ Bit-rate (%)	Δ SSIM (%)	ΔT (%)	ΔPSNR (dB)	Δ Bit-rate (%)	Δ SSIM (%)
bridge-close	-38.3	-0.134	-0.134	0.073	-38.225	-0.19	0.302	0.043
bridge-far	-37.954	-0.08	-0.08	-0.63	-32.7	0	0	-0.851
coastguard	-56.69	0.185	23.36	0.071	-54.221	-0.93	12.923	1.461
foreman	-45.41	0.899	36.68	0.462	-38.67	0.75	8.425	0.82
highway	-40.52	-0.726	13.069	0.531	-33.35	-0.77	13.291	0.89
container	-68.6	-0.423	5.5	-0.180	-67.123	0.635	-6.209	-0.187

The results are compared with the case of exhaustive search in terms of change of PSNR (ΔPSNR), bit-rate (Δ bit rate), SSIM (ΔSSIM) and encoding time (Δ Time). Computational efficiency is measured by the amount of time reduction, which is computed as follows:

$$\Delta Time = \frac{Time_{JM17.2} - Time_{fat}}{Time_{17.2}} \times 100\%$$

Delta Bit rate is measured by the amount of reduction which is computed by,

$$\Delta Bit\ rate = \frac{Bit\ rate_{17.2} - Bit\ rate_{fat}}{Bit\ rate_{17.2}} \times 100\%$$

Delta PSNR (Peak Signal to Noise Ratio) is measured by the amount of reduction which is computed by,

$$\Delta PSNR = \frac{PSNR_{17.2} - PSNR_{fat}}{PSNR_{17.2}} \times 100\%$$

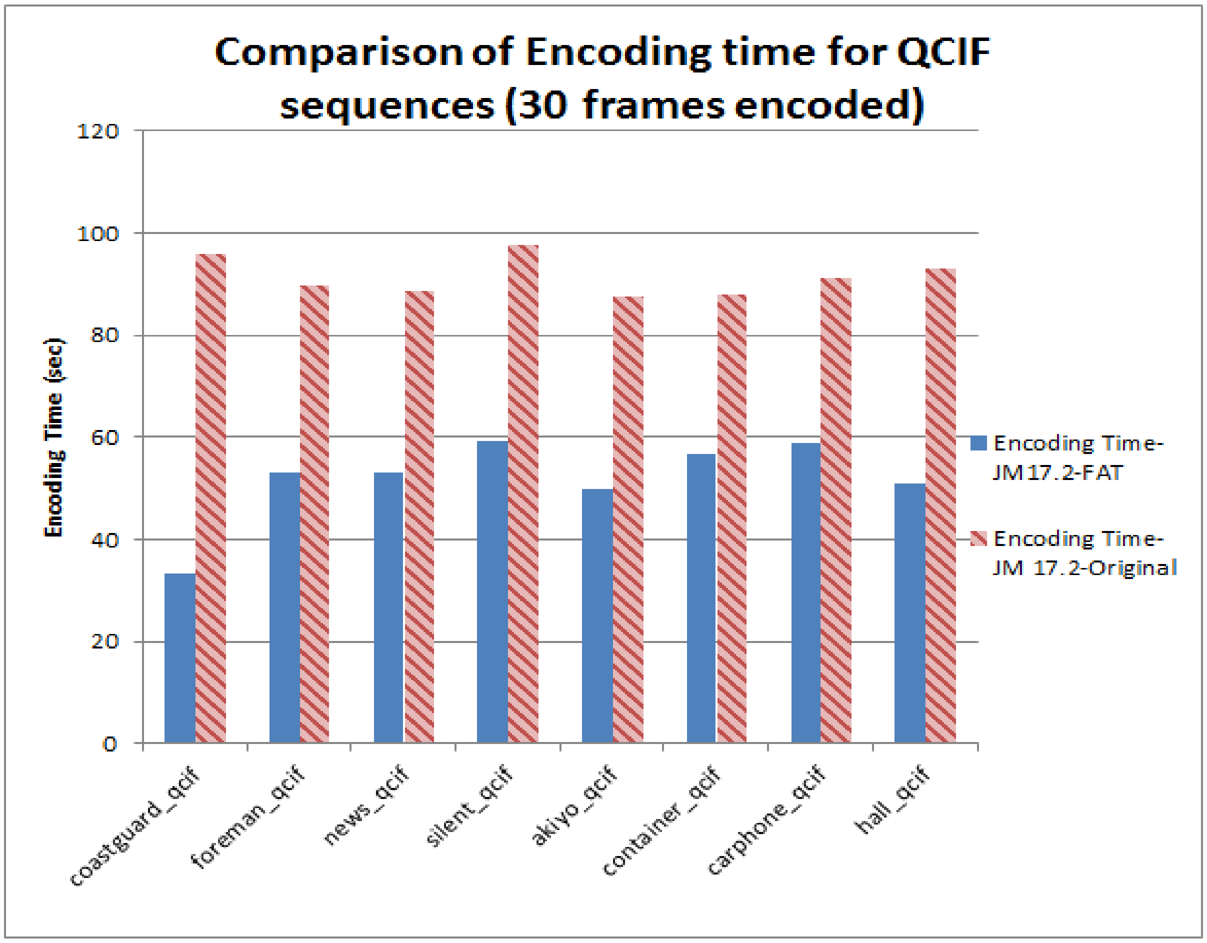


Figure 4.6 Comparison of encoding time for QCIF sequences

Figures 4.6 and 4.7 show the comparison of encoding time taken by the JM17.2 original and fast adaptive termination encoder. The results are taken at QP = 27, which is the default configuration value. It can be observed that fast adaptive termination takes less time than the JM reference software.

Figure 4.8 shows the comparison of PSNR values of the reference software JM and fast adaptive termination encoder. The results are taken at QP = 27, which is default configuration value. It can be observed that there is not much of decrease in PSNR value in fast adaptive termination algorithm encoder's results when compared to JM reference software results.

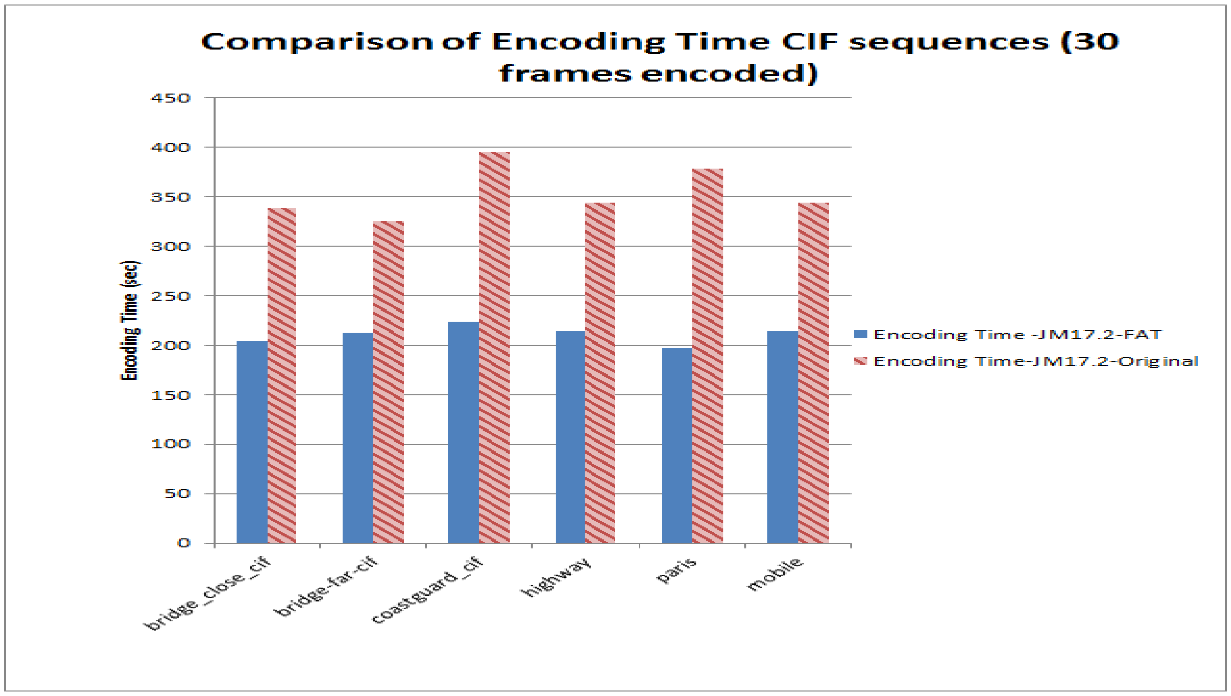


Figure 4.7 Comparison of encoding time for CIF sequences

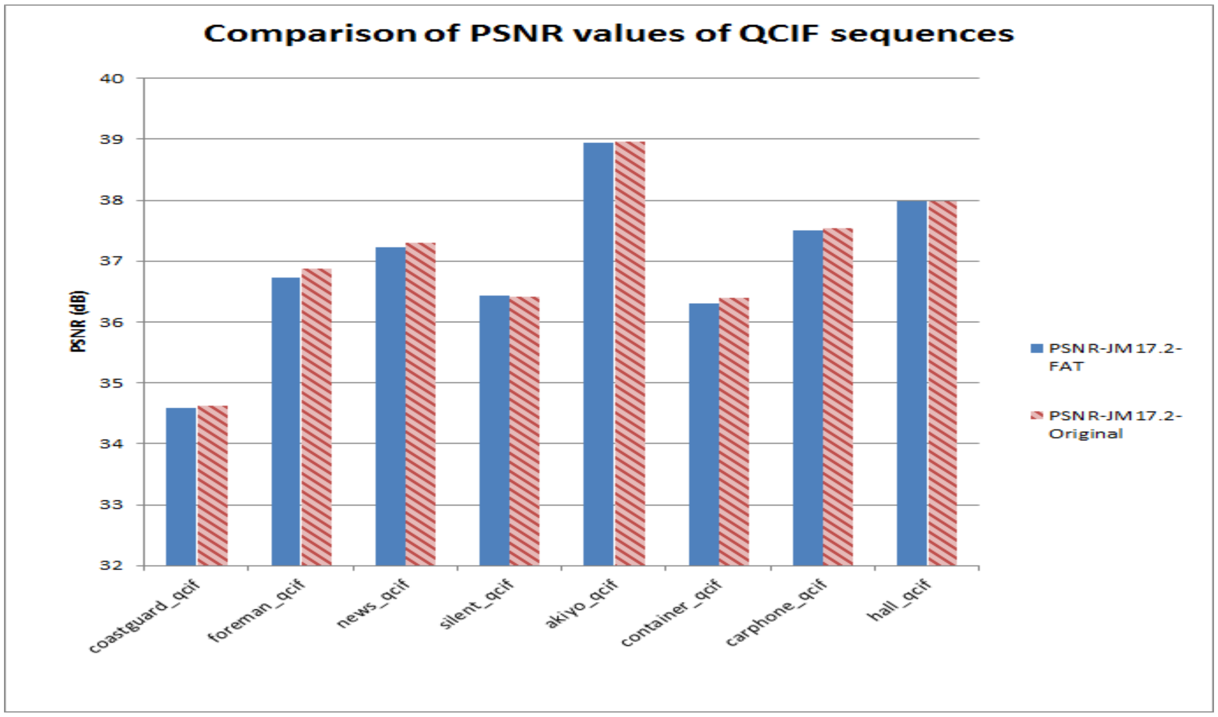


Figure 4.8 Comparison of PSNR for QCIF sequences

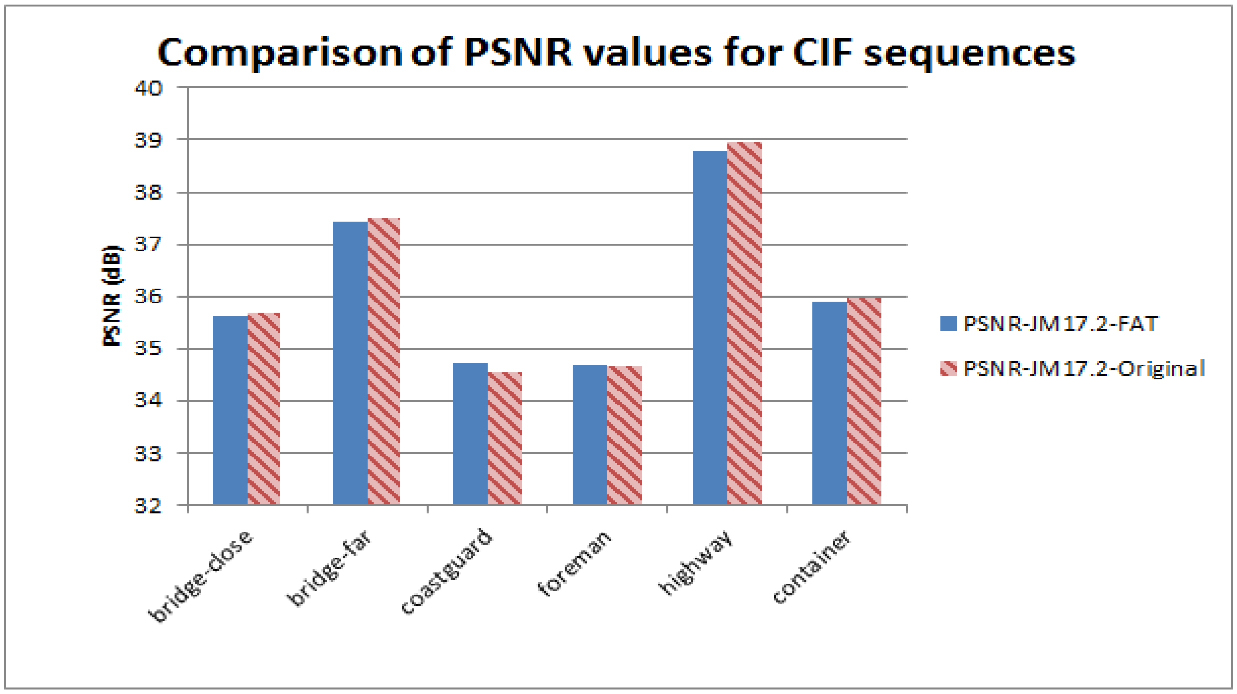


Figure 4.9 Comparison of PSNR for CIF sequences

Figure 4.10 shows the comparison of bit-rate values of the reference software JM and fast adaptive termination algorithm JM 17.2 encoder. The results are taken at QP = 27, which is the default configuration value. It can be observed that there is not much of increase in bit-rate in JM17.2-FAT encoder's results when compared to JM reference software results.

Figures 4.11 and 4.12 show the comparison of SSIM values of the reference software JM and fast adaptive termination JM encoder. The results were taken at QP = 27, which is the default configuration value. It can be observed that there is not much of decrease in SSIM in complexity reduced encoder's results when compared to JM reference software results.

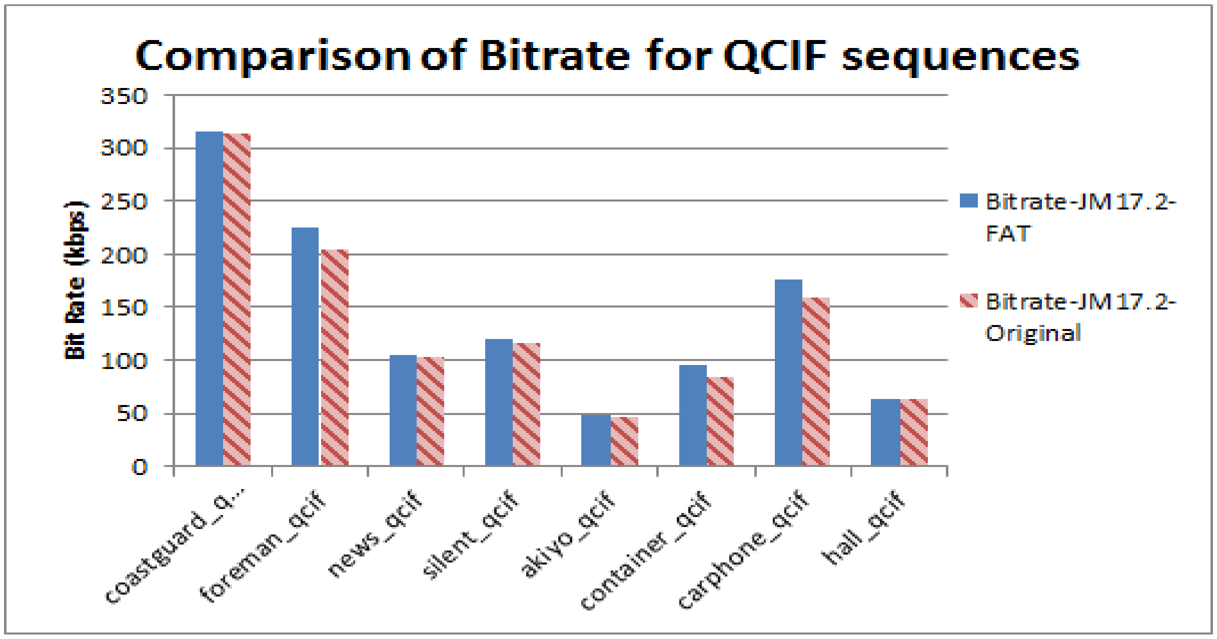


Figure 4.10 Comparison of bit-rate for QCIF sequences

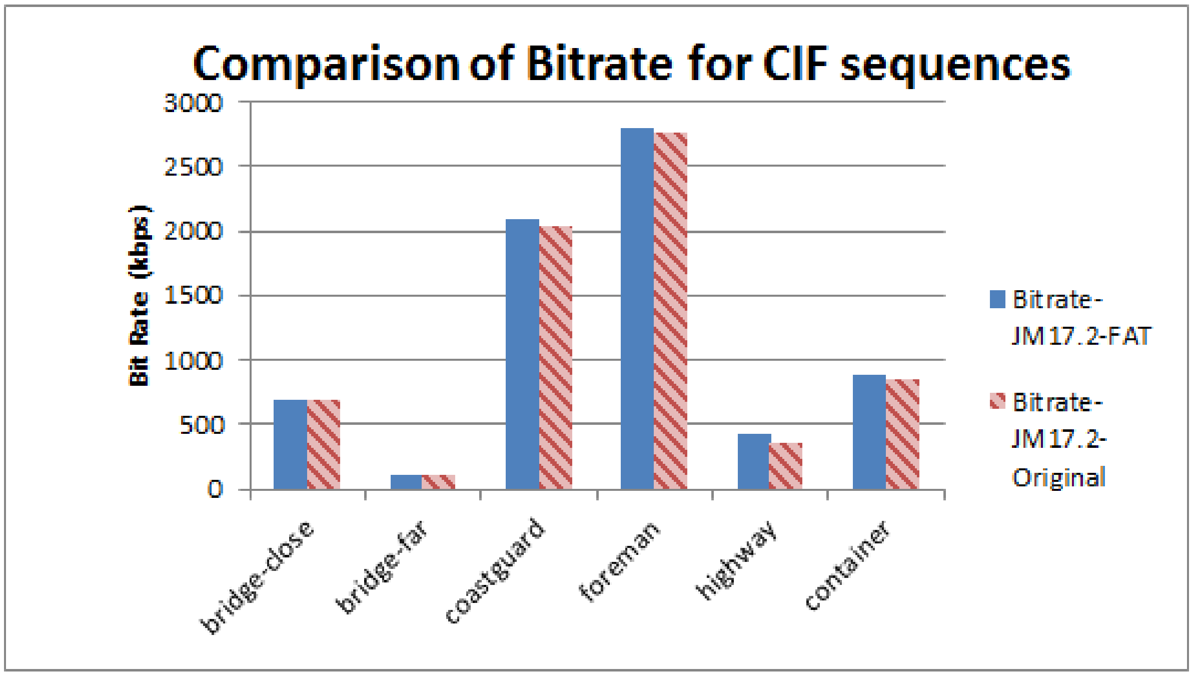


Figure 4.11 Comparison of bit-rate for CIF sequences

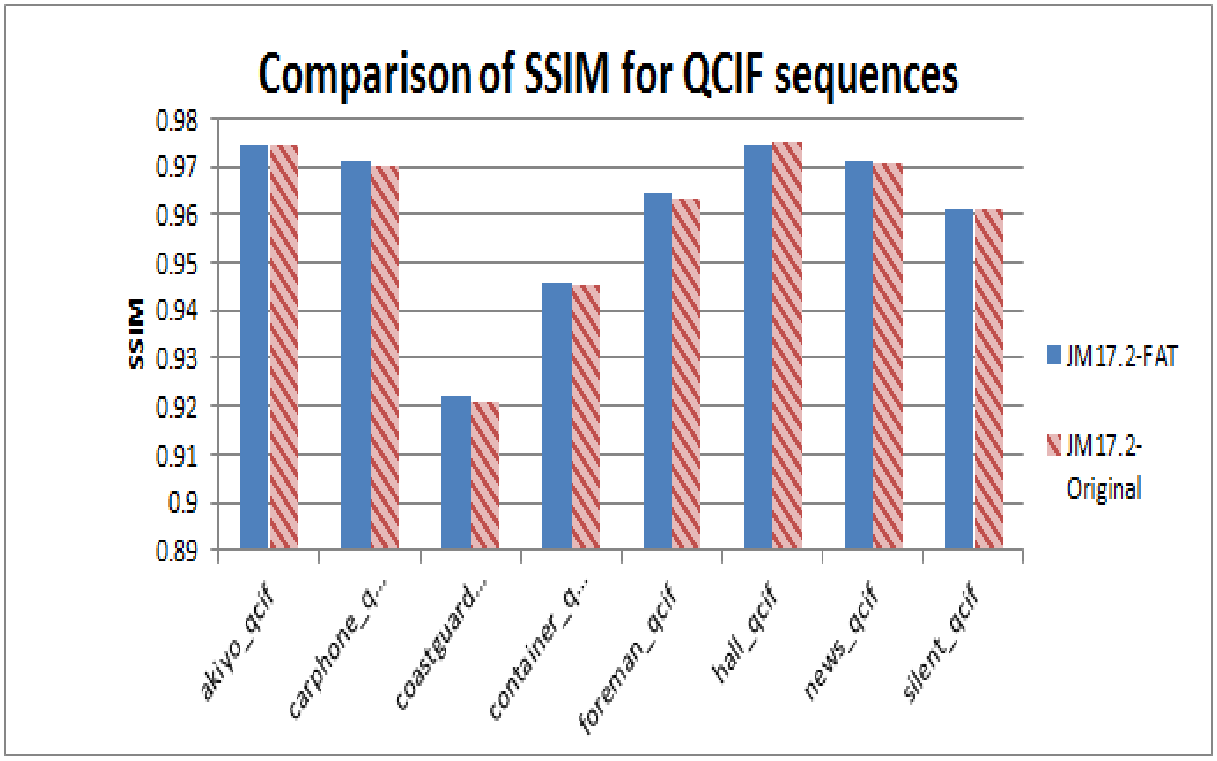


Figure 4.12 Comparison of SSIM for QCIF sequences

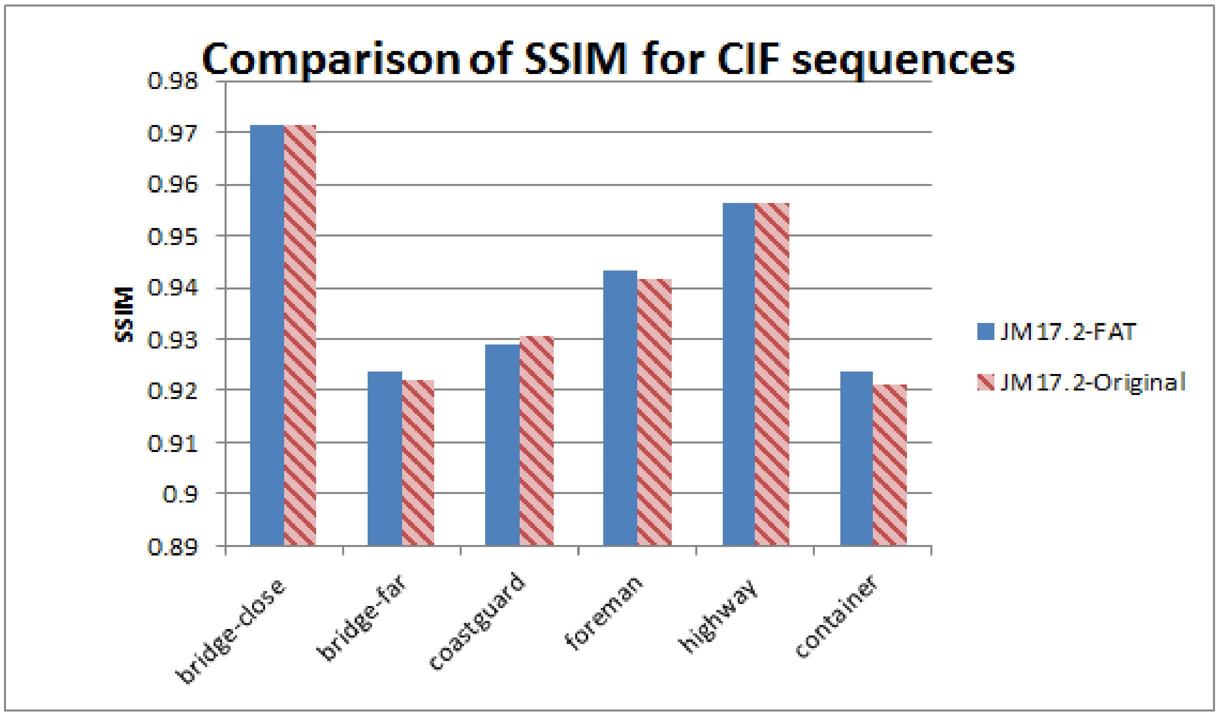


Figure 4.13 Comparison of SSIM for CIF sequences

Figure 4.13 shows a plot of PSNR Vs bit-rate, comparing PSNR and bit-rate for QCIF format, coastguard_qcif video sequence. It can be observed from the curve that on an average the complexity reduced encoder has less impact on performance when compared to reference JM software.

Figure 4.14 shows plot of encoding time Vs QP, comparing encoding time of the reference JM software with fast adaptive termination based JM17.2 encoder against different values of QP for QCIF, coastguard_qcif video sequence.

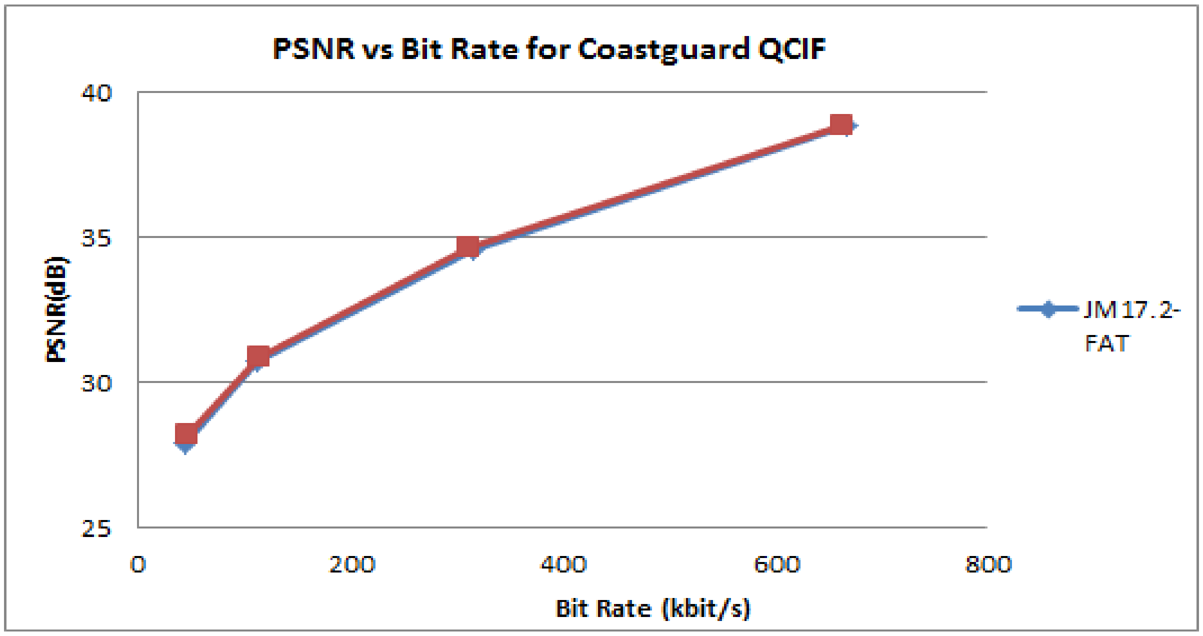


Figure 4.14 PSNR Vs bit-rate for coastguard_qcif sequence

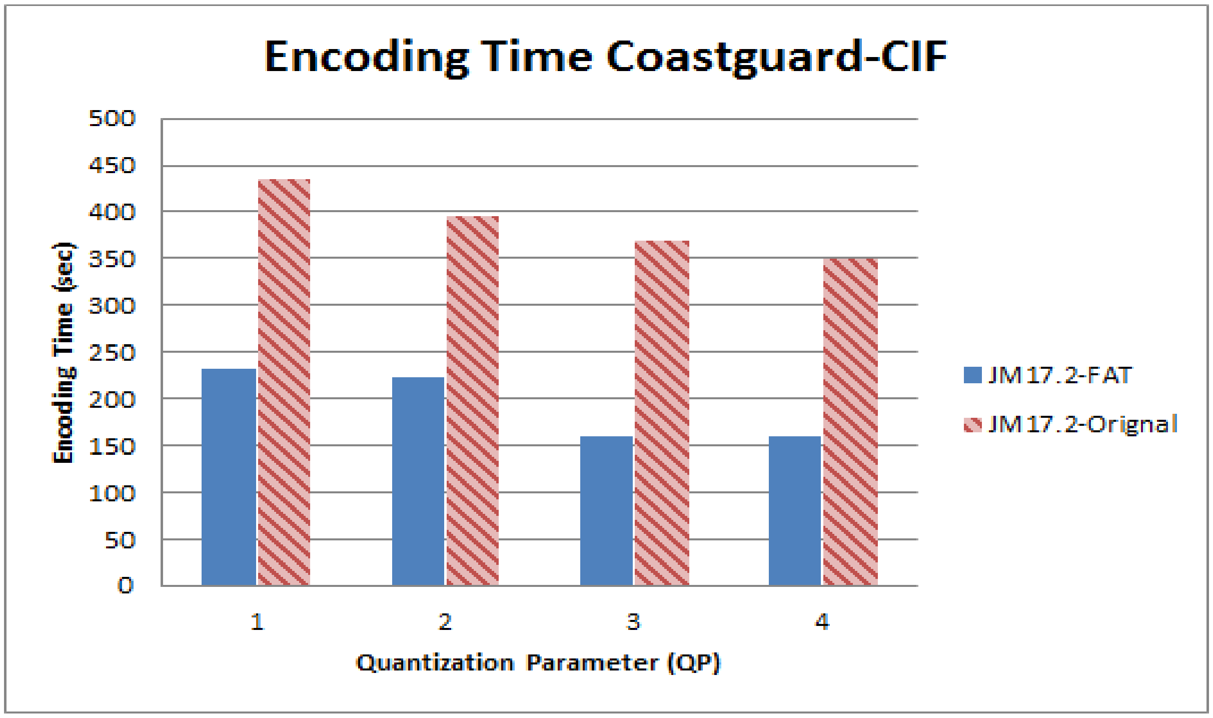


Figure 4.15 Encoding time Vs quantization parameter (QP) for coastguard_qcif sequence

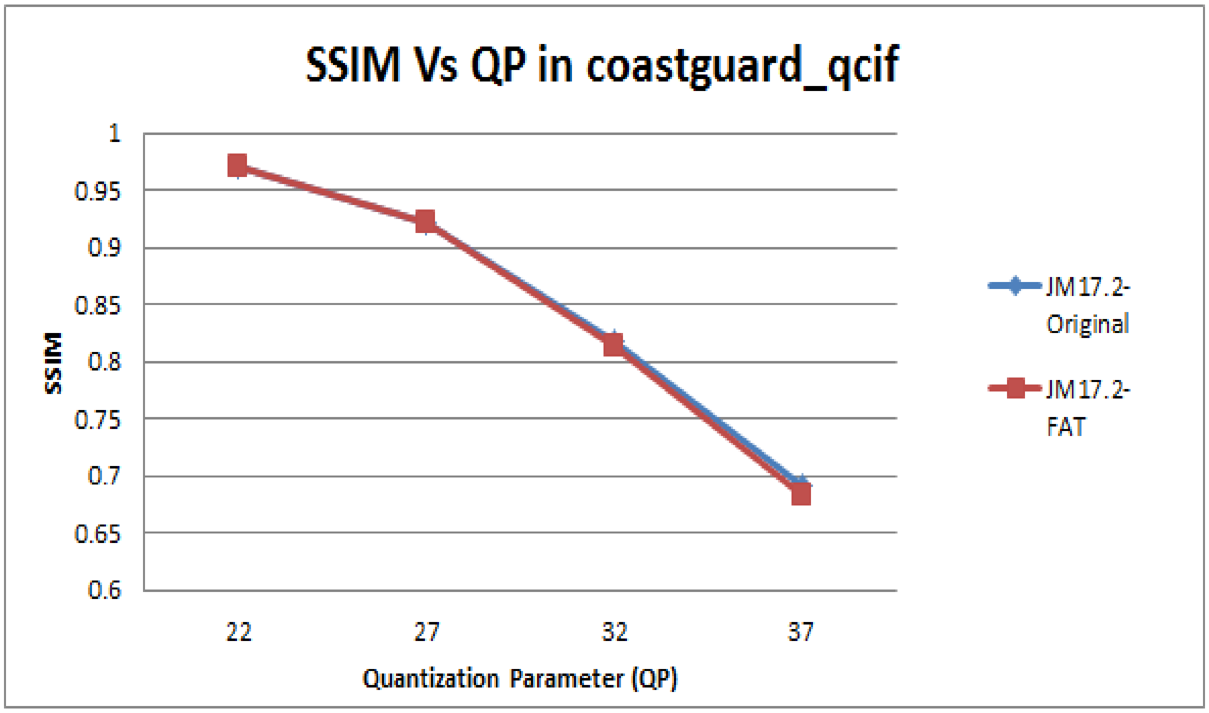


Figure 4.16 Comparison of SSIM Vs quantization parameter (QP) for coastguard_qcif sequence

Figure 4.16 shows a plot of SSIM Vs QP. This compares SSIM values of the reference software JM and fast adaptive termination based JM 17.2 encoder against different values of QP for the QCIF format, coastguard_qcif video sequence. It can be observed from the graph that the SSIM values of fast adaptive termination based JM17.2 encoder [15] do not decrease significantly when compared to the JM reference software's SSIM values.

4.5 Observations

From these simulation results it can be concluded that, the encoder with complexity reduction algorithm takes significantly less encoding time (around 43% reduction for QCIF format and around 40% reduction for CIF format) when compared to the JM reference software. It does not sacrifice the quality of the video (around 0.15% reduction in PSNR for QCIF format and around 0.26% reduction in PSNR for CIF format) nor does it increase the bit-rate significantly (around 6% increase in bitrate for QCIF format and around 9.5% increases in case of CIF format). Hence, this approach of reducing the number of mode combinations in temporal domain using fast adaptive termination can find its application in low complexity devices like mobile or any handheld devices.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

From the simulation results described in Chapter 4, it can be concluded that the proposed fast adaptive termination algorithm based JM encoder is faster than the JM reference software. This is because the JM reference software uses rate distortion optimization (RDO) in which it examines all possible combinations of coding modes i.e. brute force. The complexity reduction algorithm makes use of early skip mode detection, adaptive thresholds and homogeneity detection.

From Tables 4.1 through 4.4, it can be observed that there is an average of 43% reduction in encoding time when using the fast adaptive termination based encoder with negligible loss of PSNR and SSIM. The bite rate increases only slightly over JM reference software

Figures 4.5 through 4.11, show plots of comparison of encoding time, PSNR, bit-rate and SSIM between the JM reference software and fast adaptive termination based JM encoder. The simulation was performed on CIF and QCIF sequences. Figures 4.12 through 4.15 show plots of comparison of PSNR, bit-rate, encoding time and SSIM between JM reference software and fast adaptive termination based JM encoder, using coastguard_qcif sequence at QP = 27. The simulation was performed for various values of QP = 22, 27, 32, 37. These simulation results again concur that, significant encoding time can be reduced by using the proposed complexity reduction algorithm and at the same time, fidelity of the input video stream is maintained.

5.2 Future work

The fast adaptive termination based JM encoder was implemented for CIF and QCIF format video sequences. This idea can be extended further to other video formats like 4SIF and HD. Multi-view coding (MVC) an amendment to the H.264/MPEG-4 AVC video compression standard which enables efficient encoding of sequences captured simultaneously from multiple cameras using a single video stream also has high computing complexity [33]. Complexity reduction can be very useful in MVC as it contains large amounts of inter-view statistical dependencies. Also, the complexity reducing fast adaptive termination algorithm was integrated in to JM17.2 [15] reference software; it can also be integrated into other open source H.264 softwares like X264 and Intel IPP [41]. Since the aim is to reduce the overall complexity suitable for handheld devices with limited computing resources, algorithms which reduce the mode combinations in intra prediction can also be integrated with this complexity reducing fast adaptive termination based encoder.

REFERENCES

1. Open source article, “Intra frame coding” : http://www.cs.cf.ac.uk/Dave/Multimedia/node248.html
2. Open source article, “MPEG 4 new compression techniques” : http://www.meabi.com/wp-content/uploads/2010/11/21.jpg
3. Open source article, “H.264/MPEG-4 AVC,” Wikipedia Foundation, http://en.wikipedia.org/wiki/H.264/MPEG-4_AVC
4. I.E.Richardson,”The H.264 advanced video compression standard”,2nd Edition ,Wiley 2010.
5. R. Schafer and T.Sikora,”Digital video coding standards and their role in video communications,”Proceddings of the IEEE Vol 83,pp. 907-923,Jan 1995.
6. G.Escribano et.al,”Video encoding and transcoding using machine learning,” MDM/KDD’08,August 24,2008,Las Vegas,NV,USA.
7. D. Marpe, T. Wiegand and S. Gordon, “H.264/MPEG4-AVC Fidelity Range Extensions: Tools, Profiles, Performance, and Application Areas”, Proceedings of the IEEE International Conference on Image Processing 2005, vol. 1, pp. 593 - 596, Sept. 2005.
8. ITU-T Recommendation H.264-Advanced Video Coding for Generic Audio-Visual services.
9. 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.
10. A. Puri et al, “Video coding using the H.264/ MPEG-4 AVC compression standard”, Signal Processing: Image Communication, vol. 19, pp: 793 – 849, Oct. 2004.

11. G. Sullivan, P. Topiwala and A. Luthra, “The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions”, SPIE conference on Applications of Digital Image Processing XXVII, vol. 5558, pp. 53-74, Aug. 2004.
12. K. R. Rao and P. C. Yip, “The transform and data compression handbook”, Boca Raton, FL: CRC press, 2001.
13. T. Wiegand and G. J. Sullivan, “The H.264 video coding standard”, IEEE Signal Processing Magazine, vol. 24, pp. 148-153, March 2007.
14. I.E.Richardson “H.264/MPEG-4 Part 10 White Paper : Inter Prediction”, www.vcodex.com, March 2003.
15. JM reference software http://iphome.hhi.de/suehring/tml/
16. G. Raja and M.Mirza, “In-loop de-blocking filter for H.264/AVC Video”, Proceedings of the IEEE International Conference on Communication and Signal Processing 2006, Marrakech, Morroco, Mar. 2006.
17. M. Wien, “Variable block size transforms for H.264/AVC”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, pp. 604–613, July 2003.
18. A. Luthra, G. Sullivan and T. Wiegand, “Introduction to the special issue on the H.264/AVC video coding standard”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, issue 7, pp. 557-559, July 2003.
19. H.Kim and Y.Altunhasak, “Low-Complexity macroblock mode selection for H.264-AVC encoders”, IEEE International Conference on Image Processing, vol.2, pp .765-768, Oct 2004.
20. “Editor's Proposed Draft Text Modifications for Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), Draft 2”, JVT-E022d2, Geneva, Switzerland, 9-17 October, 2002

21. A. Tourapis, O. C. Au, and M. L. Liou, “Highly efficient predictive zonal algorithm for fast block-matching motion estimation,” IEEE Trans. Circuits Syst. Video Technol., vol. 12, pp. 934-947, Oct. 2002

22. Z. Chen, P. Zhou and Y He, “Fast integer pel and fractional pel motion estimation for JVT”, JVT-F017r1.doc, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, 6th meeting, Awaji, Island, JP, 5-13 December, 2002.

23. A. M. Tourapis, " Enhanced predictive zonal search for single and multiple frame motion estimation,” in proceedings of Visual Communications and Image Processing 2002 (VCIP-2002), pp. 1069-1079, San Jose, CA, January 2002.

24. Y. Lin and S. Tai, “Fast full search block matching algorithm for motion compensated video compression”, IEEE Transactions on Communications, vol. 45, pp. 527-531, May 1997.

25. T. Uchiyama, N. Mukawa, and H.Kaneko,”Estimation of homogeneous regions for segmentation of textured images,” Proceedings of IEEE ICPR, pp. 1072-1075, 2002.

26. X. Liu, D. Liang and A. Srivastava, “Image segmentation using local special histograms,” Proceedings of IEEE ICIP, pp. 70-73, 2001.

27. F. Pan, X. Lin, R. Susanto, K. Lim, Z. Li, G. Feng, D. Wu and S. Wu, ”Fast mode decision for intra prediction,“ Doc. JVT-G013,Mar.2003.

28. D. Wu et al.,“Fast intermode decision in H.264/AVC video coding,” IEEE Transactions on Circuits and System for Video Technology, vol. 15, no. 7, pp. 953-958,July 2005.

29. YUV test video sequences : http://trace.eas.asu.edu/yuv/

30. J. Ren et al., “Computationally efficient mode selection in H.264/AVC video coding”, IEEE Transactions on Consumer Electronics, Vol. 54, No.2, pp. 877-886, May 2008.

31. Z. Wang et al,”Image quality assessment: From error visibility to structural similarity,” IEEE Transactions on Image Processing, vol.13, no.4, pp.600-612, Apr. 2004.

32. A.Puri, et al, “Video coding using the H.264/MPEG-4 AVC compression standard”, Signal Processing: Image Communication, vol.19, pp. 793-849, Oct. 2004.

33. Multi-view Coding H.264/MPEG-4 AVC : http://mpeg.chiariglione.org/technologies/mpeg-4/mp04-mvc/index.htm

34. CIF and QCIF formats : http://en.wikipedia.org/wiki/Common_Intermediate_Format

35. T.Wiegand et al,”Rate-constrained coder control and comparison of video coding standards,” IEEE Trans. Circuits Systems Video Technology, vol. 13, no.7, pp.688-703, July 2003.

36. T.Stockhammer, D.Kontopodis, and T.Wiegand,” Rate-distortion optimization for H.26L video coding in packet loss environment,” in Proc. Packet Video Workshop 2002, Pittsburgh, PA, April 2002.

37. K.R.Rao and J.J.Hwang,”Techniques and standards for digital image/video/audio coding,” Englewood Cliffs, NJ: Prentice Hall, 1996.

38. Open source article,”Blu-ray discs”, http://www.blu-ray.com/info/

39. Open source article,” Coding of moving pictures and audio” http://mpeg.chiariglione.org/standards/mpeg-2/mpeg-2.htm

40. Open source article,” Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios”, http://www.itu.int/rec/R-REC-BT.601/

41. Integrated Performance Primitives from Intel Website: http://software.intel.com/en-us/articles/intel-ipp/#support, 2009.

42. T.Purushottam,” Low complexity H.264 encoder using machine learning,” M.S. Thesis, E.E Dept, UTA, 2010.

43. S.Muniyappa,” Implementation of complexity reduction algorithm for intra mode selection in H.264/AVC,” M.S. Thesis, E.E Dept, UTA, 2011.

44. R.Su, G.Liu and T.Zhang,” Fast mode decision algorithm for intra prediction in H.264/AVC with integer transform and adaptive threshold,” Signal, Image and Video Processing, vol.1, no.1, pp. 11-27, Apr. 2007.

45. D.Kim, K.Han and Y.Lee,” Adaptive single-multiple prediction for H.264/AVC intra coding,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 20, no. 4, pp. 610-615, April 2010.

46. G.J.Sullivan,” The H.264/ MPEG-4 AVC video coding standard and its deployment status,” Proc. SPIE Conf. Visual Communications and Image Processing (VCIP), Beijing, China, July 2005.

47. D.Marpe, T.Wiegand and G.Sullivan,” The H.264/MPEG-4 advanced video coding standard and its applications,” IEEE, Communications Magazine, vol. 44, no.8, pp. 134-143, Aug. 2006.

48. T.Wiegand and G.Sullivan,”The picturephone is here: Really”, IEEE Spectrum, vol.48, pp.50-54, Sept. 2011.

BIOGRAPHICAL STATEMENT

Amruta Kiran Kulkarni was born in Pune, India in 1985. She received the Bachelor's degree in Electronics and Communication Engineering from Pune University, India in 2006. She worked in Sasken Communication Technologies from Dec. 2006 to Feb. 2009 as a Software Engineer in Pune. She decided to pursue the Master's degree in University of Texas at Arlington in Fall 2009. She worked as a Graduate Research Assistant under Dr. Rao in the Multimedia Processing Lab from Summer 2010 to Spring 2011. She did an internship in Research in Motion in Irving, Texas as an OS Software Developer from Summer 2011 till December 2011. Her interests lie in the field of video coding and embedded systems.