# IMPLEMENTATION OF COMPLEXITY REDUCTION ALGORITHM FOR INTRA MODE SELECTION IN H.264/AVC

by

SANTOSH KUMAR MUNIYAPPA

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

December 2011

## ACKNOWLEDGEMENTS

First and foremost I would like to offer my earnest gratitude to my supervisor, Dr. K. R. Rao, who has been a constant support throughout my thesis with his patience and profound knowledge. His persistent encouragement and motivation throughout the research process was of substantial value, without which this thesis would not have been complete.

Besides my advisor, I would like to thank the rest of my thesis advisory committee: Dr. W. Alan Davis and Dr. Kambiz Alavi for reviewing this thesis and providing their insightful comments. I am also grateful to Dr. Dongil Han for his practical advice and financial support.

I would like to acknowledge my research partner, Amruta Kulkarni for all the valuable discussions that we have had together. Special thanks to Thejaswini Purushotham for her inputs during the course of this research. Also, I thank all my friends who helped me get through two years of graduate school.

Most importantly, none of this would have been possible without the love and patience of my family. I would like to thank my parents and siblings for supporting and encouraging me to pursue this degree. Lastly, I would like to thank Bhavya for her emotional and moral support during my graduate years.

November 7, 2011

## ABSTRACT

IMPLEMENTATION OF COMPLEXITY REDUCTION ALGORITHM

FOR INTRA MODE SELECTION

IN H.264/AVC

Santosh Kumar Muniyappa, MS

The University of Texas at Arlington, 2011

Supervising Professor: K. R. Rao

For applications with low computational capabilities like handheld devices, it is necessary that the encoding complexity is minimal. But H.264, which is the most widely accepted video platform employs several powerful coding techniques that increase encoding complexity. Hence, the objective of this thesis is to implement an algorithm which reduces the encoding complexity by about 25%, but retains the quality of the existing intra prediction algorithm.

H.264 offers nine modes for intra prediction of 4x4 luminance blocks, which includes DC prediction and eight directional modes (N4). For regions with less spatial detail, H.264 supports 16x16 intra coding, where in one of the four prediction modes (DC, vertical, horizontal and planar) is chosen for the prediction of the entire luminance component of the macro-block (N16). In addition, H.264 supports intra prediction for the 8x8 chrominance blocks which also use the similar four prediction modes as 16x16 luminance blocks (N8). The existing intra prediction algorithm uses Rate Distortion Optimization(RDO) to examine all possible combinations of coding modes. Therefore the number of mode combinations for each macro-block would be N8x (16xN4 + N16) = 4 x (16 x 9 + 4), which sums up to 592.

Thus, to select the best mode for one macro-block in the intra prediction, the H.264/AVC encoder carries out 592 RDO calculations. As a result, the complexity of the encoder increases extremely. This thesis adopts a complexity reduction algorithm using simple directional masks and neighboring modes where in, the number of mode combinations are reduced to 132 at the most, with negligible loss of PSNR(peak signal to noise ratio) and bit-rate increase compared with the H.264 exhaustive search.

## TABLE OF CONTENTS

## LIST OF ILLUSTRATIONS

LIST OF TABLES

## LIST OF ACRONYMS

| | |
|---|---|
| 3D: | Three Dimensional |
| 3G: | Third Generation |
| ASO: | Arbitrary Slice Order |
| AVC: | Advanced Video Coding |
| B slice: | Bi-directionally predictive slice |
| CABAC: | Context-Based Adaptive Binary Arithmetic Coding |
| CAVLC: | Context Adaptive Variable-Length Coding |
| CCITT: | International Telegraph and Telephone Consultative Committee |
| CD-ROM: | Compact Disc – Read Only Memory |
| CIF: | Common Intermediate Format |
| DCT: | Discrete Cosine Transform |
| DVD: | Digital Video Disk |
| FMO: | Flexible Macro-block Order |
| I slice: | Intra slice |
| IEC: | International Electrotechnical Commission |
| ISDN: | Integrated Services Digital Network |
| ISO: | International Organization for Standardization |
| ITU-T: | International Telecommunication Union – Transmission standardization sector |
| JM: | Joint Model |
| JVT: | Joint Video Team |
| MC: | Motion Compensation |

MPEG: Moving Picture Experts Group

MV: Motion Vector

MVD: Motion Vector Difference

MVp: Prediction Motion Vector

NAL: Network Abstraction Layer

NTSC: National Television System Committee

P slice: Predictive slice

PAL: Phase Alternating Line

PCM: Pulse Code Modulation

PSNR: Peak signal to noise ratio

QP: Quantization Process

RDO: Rate Distortion Optimization

RGB: Red, Green and Blue

RS: Redundant slice

SAD: Sum of Absolute Differences

SATD: Sum of Absolute Transformed Differences

SI: Switching Intra

SIF: Source Input Format

SP: Switching Prediction

SSD: Sum of Squared Differences

SSIM: Structural Similarity Index Metric

TV: Television

UVLC: Universal Variable Length Coding

VCEG: Video Coding Experts Group

VCL: Video Coding Layer

VHS: Video Home System

# CHAPTER 1

# INTRODUCTION

## 1.1 Significance

The growing market for high bit-rate connections, large storage capacity of hard disks, flash memories and optical media has come a long way to satiate the demands of users. With the price per transmitted or stored bit continually falling, video compression is a necessity, and there has been a significant effort to make it better. Imagine a world without video compression; current Internet throughput rates would have been insufficient to handle uncompressed video in real time (even at low frame rates and/or small frame size), a digital versatile disk (DVD) could only store a few seconds of raw video at television-quality resolution and frame rate. Video compression enables an efficient use of transmission and storage resources. For example, if a high bit-rate transmission channel is available, then it is a more attractive proposition to send high-resolution compressed video or multiple compressed video channels than to send a single, low-resolution, uncompressed stream. Even with constant advances in storage and transmission capacity, compression is likely to be an essential component of multimedia services for many years to come [1].

By definition, compression is the process of removing redundancy from an information carrying signal. In a lossless compression system statistical redundancy is removed so that the original signal can be perfectly reconstructed at the receiver. Unfortunately, there is a trade off involved here; lossless methods only achieve a modest amount of compression of video signals. Most of the practical video compression algorithms are based on lossy compression, in which greater compression is achieved with the penalty that the decoded video signal is not identical to the original. The goal of a video compression algorithm is to achieve efficient compression while minimizing the distortion introduced by the compression process.

When it comes to video clips, it is possible to compress the data by combining the principles behind lossless and lossy encoding. The simplest ways of building a video clip is to tack together consecutive pictures and refer to them as frames. Inherently there is a lot of redundancy in a video clip; most of the information contained in a given frame is also in the previous frame. Only a small percentage of any particular frame is new information; by calculating where that percentage of information lies, and storing only that amount, it is possible to drastically cut down the data size of the frame. This compression process involves applying an algorithm to the source video to create a compressed file that is ready for transmission or storage. An inverse algorithm is applied to the compressed video to produce a video that shows nearly the same content as the original video. This pair of algorithms which work together is called a video codec (encoder / decoder).

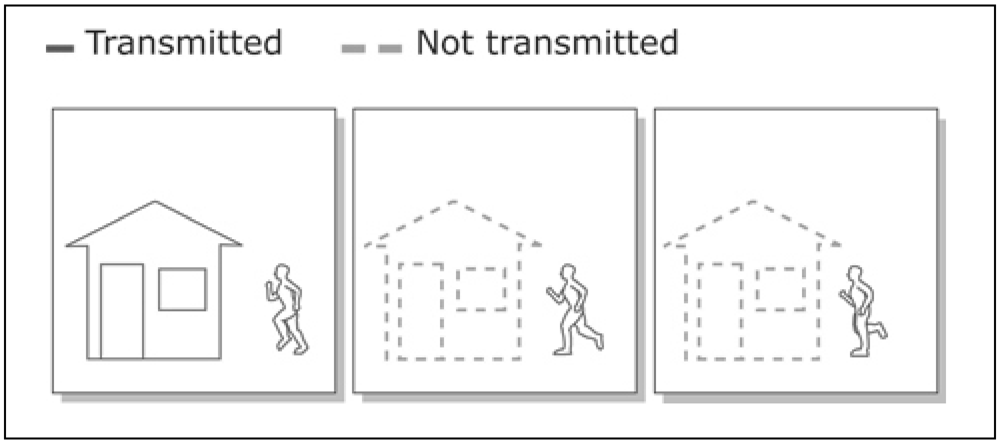


Figure 1.1 Interframe prediction in modern video compression algorithms [2]

Video compression algorithms such as MPEG-4 [1] and H.264 [1] [3] [11] are highly complex processes which include techniques such as difference coding, where in only the first image is coded in its entirety. Referring to Figure 1.1, in the two following images, references are made to the first picture for the static elements, i.e. the house. Only the moving parts, i.e. the running man, are coded using motion vectors, thus reducing the amount of information that is sent and stored. Also, techniques like block-based motion compensation are included to further reduce the data. Block-based motion compensation is based on the observation that a new frame in a video sequence can be found in an earlier frame, but perhaps in a different location. This technique divides a frame into a series of macro-blocks (blocks of pixels). Block by block, a new frame can be predicted by finding a matching block in a reference frame. If there is a match, the encoder codes only the position where the matching block is to be found in the reference frame. This technique takes a lot less bits than coding the actual content of a block itself.

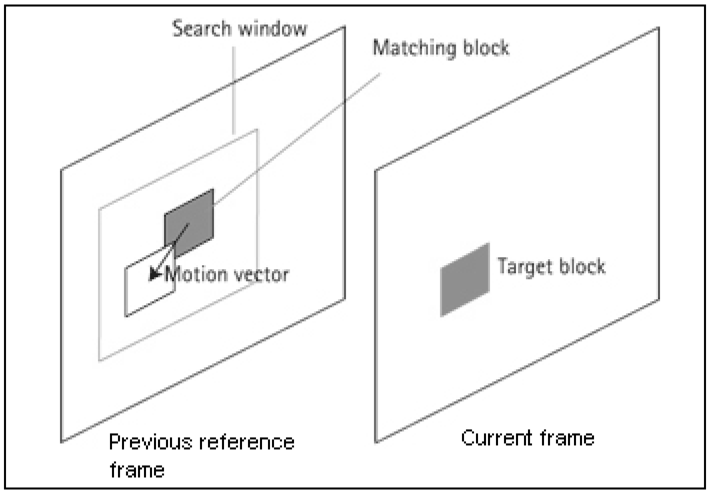


Figure 1.2 Illustration of block-based motion compensation [2]

H.264 video coding standard is the latest block-oriented motion-compensation-based codec standard developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG) [11]. The intent of the H.264 project was to create a video coding standard which could achieve quality equivalent to previous standards at substantially lower bit rates. H.264 provides significantly better compression than any previous standards, it contains a number of built in features to support reliable, robust transmission over a range of channels and networks. Unfortunately, this comes with a cost of increased encoder computational complexity when compared to previous standards. To achieve a practical implementation of H.264/AVC, a significant reduction in encoding complexity must be achieved while maintaining the coding efficiency [12].

The requirement of capturing and playing of high definition video applications on devices like smart phones and tablets has led to a challenging scenario of developing efficient video encoders with low complexity. Many techniques have been proposed by researchers around the globe to reduce the complexity in H.264. Different Intra mode complexity reduction approaches like in [12], [13], [14], [15], have been proposed, but very few approaches achieve efficient encoding. Some approaches reduce the encoding time but, fail to maintain the quality of that of original video clip. It is important to strike a balance between gain and quality. This thesis proposes an efficient intra mode complexity reduction algorithm; wherein the encoding time of a video clip is greatly reduced with negligible quality loss and increase in bit-rate [16].

## 1.2 Summary

The proposed thesis focuses on reducing encoding complexity of H.264 for intra mode selection by making use of JM 18.0 [17]. It is heavily based on the observation that adjacent macro-blocks tend to have similar properties. Thus, by simple use of directional masks and neighboring modes, the usually tasking RDO (Rate Distortion Optimization) which examines all possible combinations of coding modes process can be reduced significantly [16]. Results show reduction in complexity in terms of encoding time for different video formats and video context. Chapter 2 gives a brief insight about some of the video coding standards and video formats.

# CHAPTER 2

# VIDEO CODING STANDARDS AND VIDEO FORMATS

## 2.1 Introduction

From analog television to digital television, VHS video tapes to DVDs, cell phones used for only making calls and send text messages to cell phones functioning as cameras, web browsers, navigation systems, social networking devices and barely used to make calls, there has been quite a revolution over the past few years in the way users create, share and watch videos. The continuous evolution of digital video industry is driven by commercial factors and technological advances. The commercial drive comes from the huge revenue potential of persuading consumers and businesses. In the technology field, the factors include better communications infrastructure, inexpensive broadband networks, 3G mobile networks and the development of easy-to-use applications for recording, editing, sharing and viewing videos.

There are a series of processes involved in getting a video from a source (camera or stored clip) to its destination (a display). The key processes in this operation are compression (encoding) and decompression (decoding), which involve in reducing the “bandwidth intensive” raw video source to an optimal size suitable for transmission or storage, then reconstructed for display. For having that commercial and technical edge to a product, the compression and decompression processes should strike a proper balance between three parameters that are odds with one another: quality of video, encoding time and the size. There is therefore an acute interest in video compression and decompression techniques and systems.

## 2.2 Video Coding Standards

There are many video coding techniques proposed and many other researches still ongoing out there. Hundreds of research papers are published each year describing new and innovative compression techniques. However, the commercial video coding applications tend to use a limited number of standardized techniques for video compression. Standardized video coding formats have a number of benefits like [3]:

- Standards simplify inter-operability between encoders and decoders from different manufacturers.
- Standards make it possible to build platforms that incorporate video, in which many different applications such as video codecs, audio codecs, transport protocols, security and rights management, interact in well defined and consistent ways.
- Many video coding techniques are patented and therefore there is a risk that a particular video codec implementation may infringe patent(s). The techniques and algorithms required to implement a standard are well defined and the cost of licensing patents that cover these techniques, i.e., licensing the right to use the technology embodied in the patents, can be clearly defined.

## 2.3 MPEG and H.26x

The recommendations or international standards are prepared jointly by ITU-T SG16 Q.6 (the International Telecommunication Union), also known as VCEG (Video Coding Experts Group) and by ISO/IEC JTCI/SC29/WG11 (the International Organization for Standardization), also known as MPEG (Moving Picture Experts Group). VCEG was formed in 1997 [11] to maintain prior ITU-T video coding standards and develop new video coding standard(s) appropriate for a wide range of conversational and non-conversational services. MPEG was formed in 1988 [22] to establish standards for coding of moving pictures and associated audio for various applications such as digital storage media, distribution, and communication. Later on, the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) formed a Joint Video Team (JVT) in 2001 for development of a new

Recommendation or International Standard, H.264 Recommendation/MPEG-4 part 10 standard [4].

*2.3.1. H.120*

H.120 [5], the first digital video coding standard was developed in 1984 by ITU-T formerly CCITT (the International Telegraph and Telephone Consultative Committee). It evolved into different versions, Version1 developed in 1984 featured conditional replenishment, differential pulse code modulation, scalar quantization, variable length coding and a switch for quincunx sampling. Version2 developed in 1988 added motion compensation and background prediction. In 1993, a final edition was published as a result of the creation of the ITU-T to replace the prior CCITT standardization body. H.120 streams ran at 1544 kbps for NTSC (National Television System Committee) and 2048 kbps for PAL (Phase Alternating Line) [10]. H.120 video was not of good quality for practical use since the differential PCM (Pulse Code Modulation) in it worked on pixel by pixel basis, which is good for spatial resolution but the temporal quality was really poor. It was necessary to improve the quality of video without exceeding the target bitrates for the stream. Hence the researchers came up with the block-based codecs that followed H.120, such as H.261 [6].

*2.3.2 H.261*

H.261 [6] [46] was the first video codec with the widespread practical success (in terms of product support in significant quantities). The first design of this ITU-T video coding standard was in 1988 and was the first member of the H.26x family. H.261 was originally designed for transmission over ISDN (Integrated Services Digital Network) lines on which data rates are integer multiples of 64kbps. The coding algorithm uses a hybrid of motion compensated inter-picture prediction and spatial transform coding with 16x16 macro-block motion compensation, 8x8 DCT (discrete cosine transform) [31], scalar quantization, zigzag scan and variable-length coding. All the subsequent international video coding standards have been based closely on the H.261 design [10]. Figure 2.1 shows an outline block diagram of the H.261 codec.

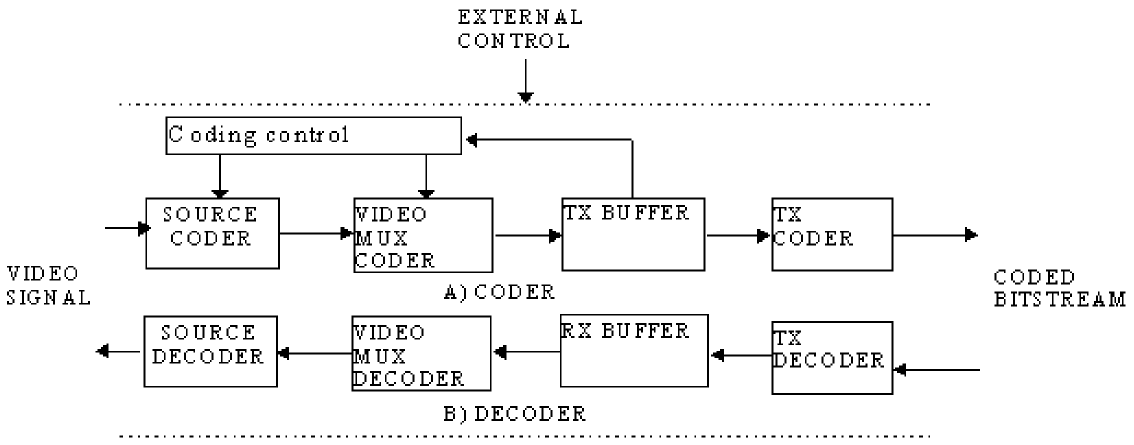


Figure 2.1 Outline block diagram of H.261 encoder and decoder [26].

*2.3.3 MPEG-1*

MPEG-1 (Moving Picture Experts Group) [7] was developed by ISO/IEC JTC1 SC29 WG11 (MPEG) in 1993[5]. MPEG-1 provides the resolution of 352x240 (source input format) for NTSC or 352x288 for PAL at 1.5 Mbps. MPEG-1 had a superior video quality compared to H.261 when operated at higher bitrates and was close to VHS quality. Its main applications were focused on video storage for multimedia (e.g., on CD-ROM).

*2.3.4 H.262 / MPEG-2*

H.262 / MPEG-2 [8] coding standard was jointly developed by ITU-T Video Coding Experts Group and ISO/IEC Moving Picture Experts Group in 1994 [5]. MPEG-2 video is similar to MPEG-1, but also provides support for interlaced video (the format used by analog broadcast TV systems). MPEG-2 video is not optimized for low bit-rates (less than 1 Mbps), but outperforms MPEG-1 at 3 Mbps and above. For the consistency of the standards, MPEG-2 is also compatible with MPEG-1, which means a MPEG-2 player can play back MPEG-1 video without any modification.

### *2.3.5 H.263, H.263+ and H.263++*

This next generation of video coding overtook H.261 as the most dominant videoconferencing codec. H.263 [9] has superior video quality compared to its prior standards at all bit rates, by a factor of two. H.263 Version1 was developed by ITU-T in 1995. Features which beat H.261 [10] are:

- 3-D variable length coding of DCT coefficients
- Median motion vector prediction
- Bi-directional prediction
- Arithmetic entropy coding

H.263+ or Version2 was developed in the late 1997 and early 1998 [23], which included lot of new features like error resilience, custom and flexible video formats, supplemental enhancement information and also there was a an improved compression efficiency over H.263v1. H.263++ or Version3 [24], developed in 2000 came with significant improvement in picture quality, packet loss and error resilience and additional supplemental enhancement information.

### *2.3.6 MPEG-4*

MPEG-4 [25] an ISO / IEC standard was developed by MPEG (Moving Picture Experts Group) in late 1998. The fully backward compatible extensions under the title of MPEG-4 Version 2 were frozen at the end of 1999, to acquire the formal International Standard Status early in 2000. To cater to variety of applications ranging from low-quality, low-resolution surveillance cameras to high definition TV broadcasting and DVDs, MPEG-4 Part2 has approximately 21 profiles. Some of the profiles are listed below [25]:

- Simple
- Simple Scalable
- Main
- Core
- N-Bit
- Hybrid
- Basic Animated Texture
- Scalable Texture
- Simple FA
- Core Scalable
- Advanced Scalable Texture
- Simple FBA
- Advanced Coding Efficiency
- Advanced Real Time Simple

### *2.3.7 H.264 / MPEG-4 Part 10 / AVC (Advanced Video Coding)*

In 1998, the ITU-T Video Coding Experts Group (VCEG) started work on a long term effort to draft “H.26L” standard, which would offer significantly better video compression efficiency than previous ITU-T standards. In 2001, the ISO Motion Picture Experts Group (MPEG) recognized the potential benefits of H.26L and The Joint Video Team (JVT) was formed, including experts from MPEG and VCEG with the charter to finalize the new video coding standard H.264/AVC. The “official” title of the new standard is Advanced Video Coding (AVC); however, it is widely known by its old working title, H.26L and by its ITU document number, H.264 [11] [27] [40] [41].

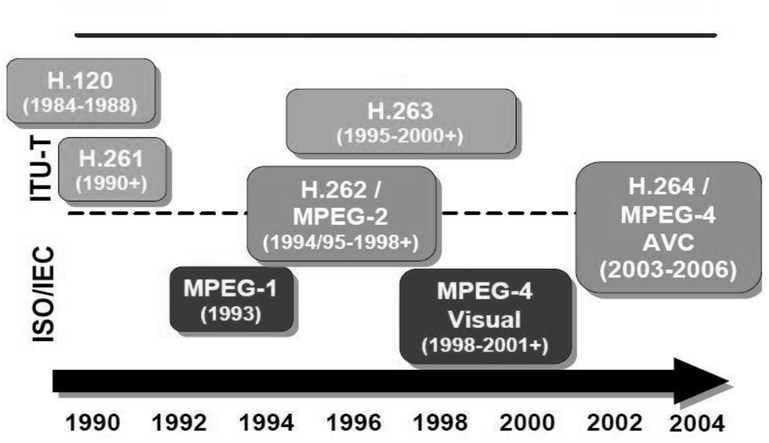


Figure 2.2 Chronology of international video coding standards [10].

H.264 [11] has brought in a significant increase in compression ratio and also saves up to 50% bit rate as compared to its prior video coding standards. The standard can increase resilience to errors by supporting flexibility in coding as well as organization of coded data. The increase in coding efficiency and coding flexibility comes at the expense of increase in complexity as compared to the other standards. These features will be discussed in much detail in chapter 3.

## 2.4 Video Formats and Quality

A typical real-world scene is composed of multiple objects with their own characteristic shape, depth, texture and illumination. The spatial characteristics like texture variation within scene, number and shape of objects, color, etc., and temporal characteristics like object motion, changes in illumination, movement of the camera or viewpoint, etc., of a typical natural video scene are relevant for video processing and compression.

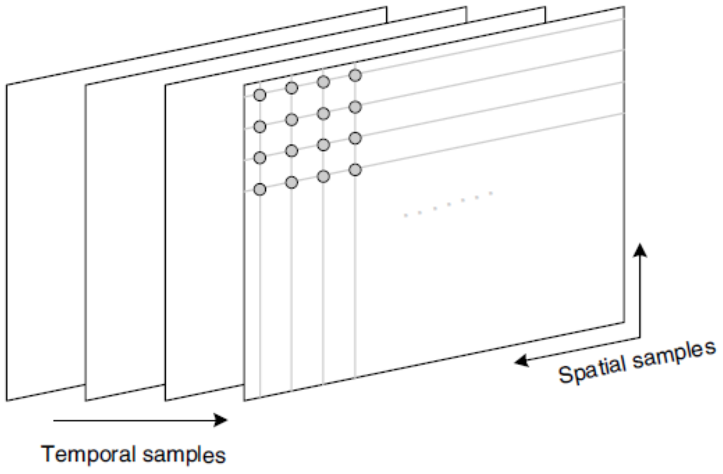


Figure 2.3 Spatial and temporal sampling of a video sequence [1].

A natural visual scene is spatially and temporally continuous. Representing a visual scene in digital form involves sampling the real scene spatially (usually on a rectangular grid in the video image plane) and temporally (as a series of still frames or components of frames sampled at regular intervals in time) (Figure 2.3). Digital video is the representation of a sampled video scene in digital form [1].

*2.4.1 Frames and Fields*

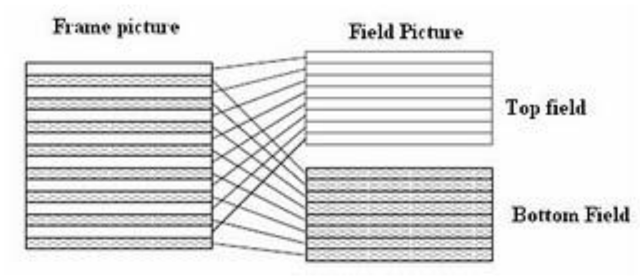


Figure 2.4 Interlaced video sequence [3].

A video signal can be progressively sampled (series of complete frames) or interlaced (sequence of interlaced fields). In an interlaced video sequence two fields comprise one video frame (Figure 2.4) and a field consists of either the odd-numbered or even-numbered lines within a complete video frame. The advantage of this sampling method is that it is possible to send twice as many fields per second as the number of frames in an equivalent progressive sequence with the same data rate, giving the appearance of smoother motion [1].

*2.4.2 Color Spaces*

Almost all digital video applications at present have color displays; hence it becomes a necessity to represent this color information. Color space method comes in handy in symbolizing brightness (luminance or luma) and color.

In the RGB color space, a color image sample is represented with three numbers that indicate the relative proportions of Red, Green and Blue (the three additive primary colors of light). Any color can be created by combining red, green and blue in varying proportions. In the RGB color space the three colors are equally important and so are usually all stored at the same resolution. However, the human visual system has lower acuity for color difference than for luminance. Therefore, the well known color space YUV is used, which represents a color image more efficiently by separating the luminance from the color information and representing luma with a higher resolution than color. Y is the luminance (luma) component and can be calculated as a weighted average of R, G and B.

$$Y = k_rR + k_gG + k_bB \qquad (2.1)$$

where, $k_r+k_g+k_b=1$

The color difference information (Chroma) can be derived as:

$$C_b = B - Y \qquad (2.2)$$

$$C_r = R - Y \qquad (2.3)$$

$$C_g = G - Y \qquad (2.4)$$

In reality, only three components (Y, $C_b$ and $C_r$) need to be transmitted for video coding because $C_g$ can be derived from Y, $C_b$ and $C_r$. As recommended by ITU-R [18], $k_r = 0.299$, $k_g = 0.587$ and $k_b = 0.114$. The equations (2.2) through (2.4) can be rewritten as:

$$Y = 0.299R + 0.587G + 0.114B \tag{2.5}$$

$$Cb = 0.564(B - Y) \tag{2.6}$$

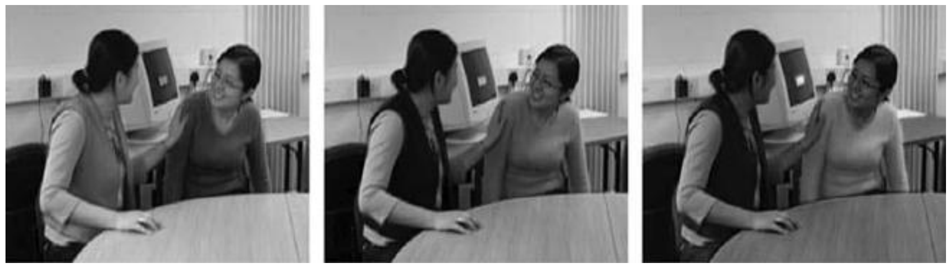
$$Cr = 0.713(R - Y) \tag{2.7}$$

$$R = Y + 1.402Cr \tag{2.8}$$

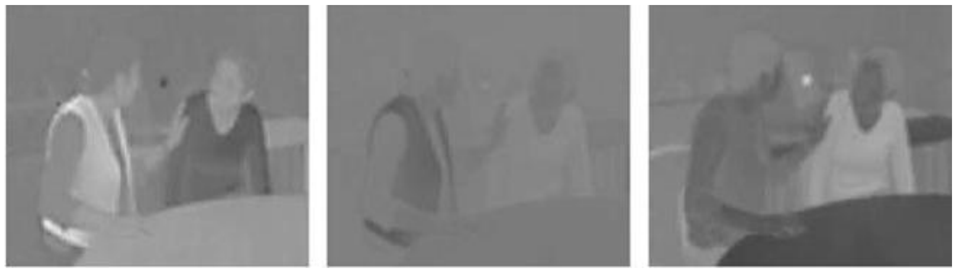
$$G = Y - 0.344Cb - 0.714Cr \tag{2.9}$$

$$B = Y + 1.772Cb \tag{2.10}$$

Figure 2.6 (a) shows the red, green and blue components of a color image in comparison to chroma components $C_r$, $C_g$ and $C_b$ of Figure 2.6 (b).



(a)



(b)

Figure 2.5 Color components of a color image. (a) Red, Green and Blue components (b) Cr, Cg and Cb components [1]

### 2.4.2.1 YCbCr Sampling Formats

Figure 2.7 shows three sampling patterns for Y, $C_b$ and $C_r$ that are supported by modern video coding standards like, MPEG-4 Visual and H.264. 4:4:4 sampling preserves the full fidelity of the chrominance components. The three components Y, Cb and Cr have same resolution and for every four luminance samples there are four $C_b$ and four $C_r$ samples. In 4:2:2 sampling also referred as YUY2, the chrominance components have the same vertical resolution as the luma but half the horizontal resolution. Meaning, for every four luminance samples in the horizontal direction there are two $C_b$ and two $C_r$ samples. 4:2:2 video is usually used for high-quality color reproduction.

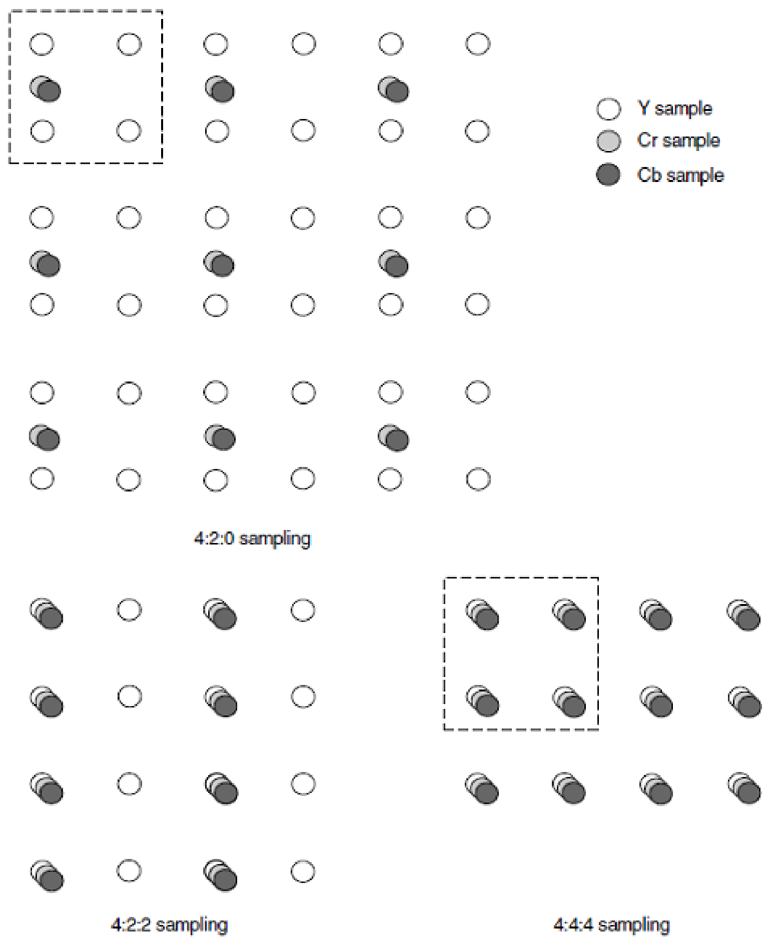


Figure 2.6 4:2:0, 4:2:2 and 4:4:4 sampling patterns (progressive) [1]

The most popular sampling pattern is 4:2:0 also referred as YV12. Here Cb and Cr have half the horizontal and vertical resolution of Y, each color difference component contains one quarter of the number of samples in the Y component. 4:2:0 YCbCr video requires exactly half as many samples as 4:4:4 or RGB vdeo, hence is widely used for consumer applications such as video conferencing, digital television and DVD [1].

### *2.4.3 Video Formats*

It is a very common practice to capture or convert to one of a set of "intermediate formats" prior to compression and transmission. Table 2.1 shows some of the popular set of formats.

Table 2.1 Video frame formats [19]

| **Format** | **Video Resolution** |
|---|---|
| Sub-QCIF | 128 × 96 |
| Quarter CIF (QCIF) | 176 × 144 |
| SIF(525) | 352 x 240 |
| CIF/SIF(625) | 352 × 288 |
| 4SIF(525) | 704 x 480 |
| 4CIF/4SIF(625) | 704 × 576 |
| 16CIF | 1408 × 1152 |
| DCIF | 528 × 384 |

The choice of frame resolution depends on the application and available storage or transmission capacity. For example, 4CIF is appropriate for standard-definition television and DVD-video; CIF and QCIF are popular for videoconferencing applications; QCIF or SQCIF are appropriate for mobile multimedia applications where the display resolution and the bit-rate are limited. SIF (Source Input Format) is practically identical to CIF, but taken from MPEG-1 rather than ITU standards. SIF on 525-Line ("NTSC") based systems is 352 × 240, and on 625-line ("PAL") based systems, it is identical to CIF (352 × 288). SIF and 4SIF are commonly used in certain video conferencing systems [19].

### *2.4.4 Quality*

It is necessary to determine the quality of the video images displayed to the viewer in order to specify, evaluate and compare. Visual quality is inherently subjective and is influenced by many factors that make it difficult to obtain a completely accurate measure of quality. Measuring visual quality using objective criteria gives accurate, repeatable results but as yet there are no objective measurement systems that completely reproduce the subjective experience of a human observer watching a video display [3].

#### 2.4.4.1 PSNR

Peak signal to noise ratio (PSNR) is the most widely used objective quality measurement. PSNR (Equation 2.11) is measured on a logarithmic scale and depends on the mean squared error (MSE) of between an original and an impaired image or video frame, relative to $(2n - 1)^2$ (the square of the highest-possible signal value in the image, where n is the number of bits per image sample).

$$PSNR_{dB} = 10 \log_{10} ((2^n - 1)^2 / MSE) \qquad (2.11)$$

PSNR can be calculated easily and quickly and is therefore a very popular quality measure, widely used to compare the 'quality' of compressed and decompressed video images.

#### 2.4.4.2 SSIM

The structural similarity (SSIM) [20] index is a method for measuring the similarity between two images. The SSIM index can be viewed as a quality measure of one of the images being compared provided the other image is regarded as of perfect quality.

# CHAPTER 3

# OVERVIEW OF H.264 / AVC STANDARD

## 3.1 Introduction

H.264 / Advanced Video Coding (AVC) standard was first jointly published in 2003 by two international standards bodies International Telecommunication Union (ITU-T) and International Organization for Standardization / International Electro-technical Commission (ISO / IEC) called as Joint Video Team (JVT) [10] [11] [21] [27]. The main objective of H.264 is to provide a means to achieve substantially higher video compression and a network friendly video representation compared to what could be achieved using any of the prior video coding standards.

H.264 has number of advantages that distinguishes it from prior standards, but it is built on the concepts of earlier standards such as MPEG-2 [8] and MPEG-4 Visual [25]. Some of the key advantages being:

- 50% reduction in bit-rate: When compared to MPEG-2 and MPEG-4 Visual simple profile, it reduces the bit rate by almost a factor of two at most bit-rates for a similar degree of encoder optimization.
- High quality video: Consistent good video quality at higher and lower bit-rates.
- Error resilience: Provides robust tools necessary to deal with packet losses in packet networks and bit errors in error prone wireless networks.
- Network friendliness: H.264 encoded bit-stream can be easily transported over different networks through the Network Adaptation Layer [11].

## 3.2 H.264 Video Codec

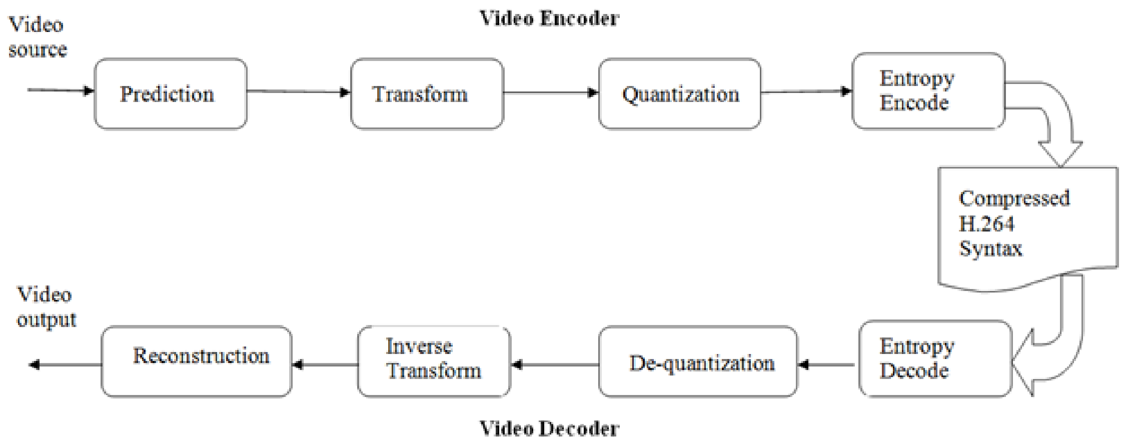


Figure 3.1 H.264 video coding and decoding process [3]

Figure 3.1 shows the basic block diagram of an H.264 codec. Like any other previous video codecs, the underlying approach is the same and consists of following four main stages:

- Spatial and temporal predictions (motion estimation and compensation) are used to exploit the redundancies that exist within the frame and between successive frames respectively.
- Transform the predicted data to reduce spatial correlation.
- Quantize the transformed co-efficients to check the bit-rate.
- Finally use entropy encoding to reduce the statistical correlation to produce compressed H.264 bit stream.

A H.264 video stream is organized in discrete packets, called “NAL units” (network abstraction layer units). NAL units are classified into VCL (video coding layer) and non-VCL NAL units. The VCL NAL units contain the data that represents the values of the samples in the video pictures, and the non-VCL NAL units contain any associated additional information such as parameter sets (important header data that can apply to a large number of VCL NAL units) and supplemental enhancement information (timing information and other supplemental data that may enhance usability of the decoded video signal but are not necessary for decoding the values of the samples in the video pictures).

Each picture of a video, which can either be a frame or a field, is partitioned into fixed-size macro-blocks that cover a rectangular picture area of 16×16 samples of the luma component and 8×8 samples of each of the two chroma components. The macro-blocks are organized in slices, which generally represent subsets of a given picture that can be decoded independently. The transmission order of macro-blocks in the bit-stream depends on the so-called Macro-block Allocation Map and is not necessarily in raster-scan order. H.264 defines five different slice types: I, P, B, SI and SP [11].

- I (intra) slice – Intra slice describes a full still image, containing only references to itself. In I slices, all macro-blocks are coded without referring to other pictures within the video sequence.
- P (predictive) slice – Predictive slices use one or more recently decoded slices as a reference (or prediction) for picture construction. The prediction is usually not exactly the same as the actual picture content, so a "residual" may be added.
- B (bi-predictive) slice – Bi-predictive slices work similar to P slices except that former and future I or P slices may be used as reference pictures. While decoding, B slices are always decoded after the following I or P slice.
- SI and SP or "switching" slices may be used for transitions between two different H.264 video streams.

Figure 3.2 shows a block diagram of H.264 encoder. The encoder includes two dataflow paths, a 'forward' path (left to right) and a 'reconstruction' path (right to left).

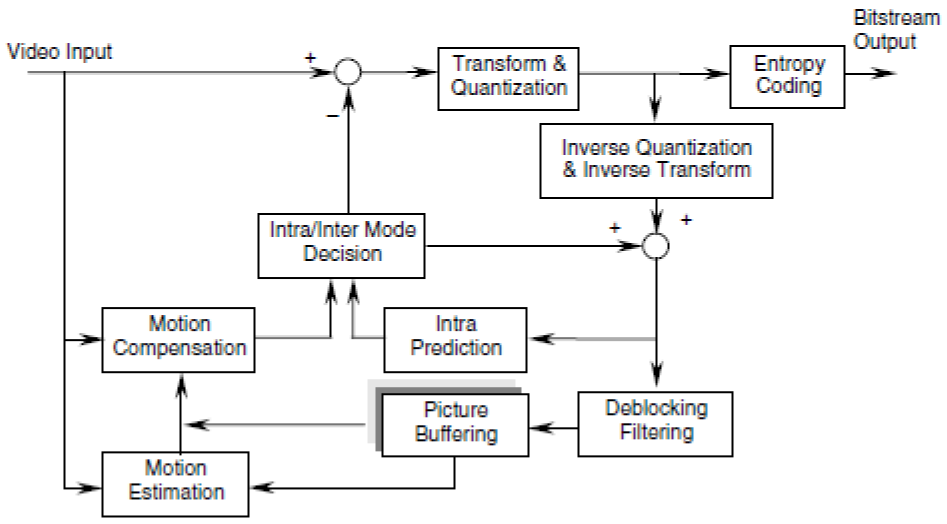


Figure 3.2 H.264 encoder [27]

The data flow in the decoder (left to right) (Figure 3.3) illustrates the similarities between the encoder and decoder.

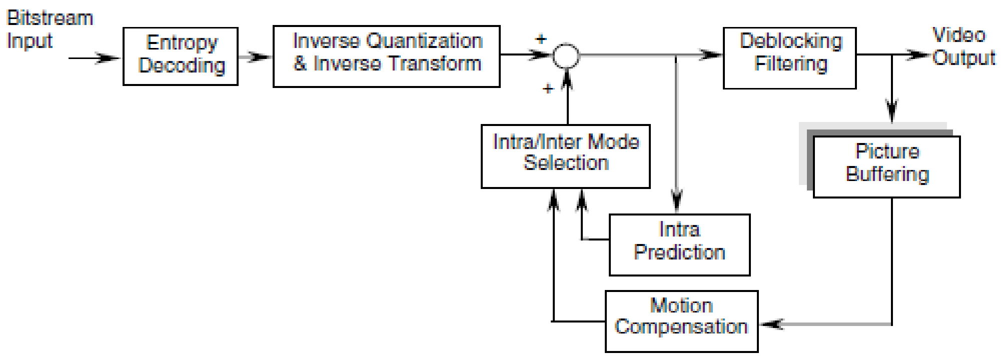


Figure 3.3 H.264 decoder [27]

### *3.2.1 Intra Prediction and Coding*

Intra coding refers to exploiting the spatial redundancies only within a video frame. The encoded I – pictures are usually large in size since a large amount of information is usually present in the frame, and no temporal information is used as part of the encoding process. To overcome this, in H.264, the spatial correlation between adjacent macro-blocks in a given frame is exploited. This was based on the fact that the adjacent macro-blocks tend to have similar properties. Therefore, as a first step in the encoding process for a given macro-block, one may predict the macro-block of interest from the surrounding macro-blocks (generally the ones located on top and to the left of the macro-block of interest, as they would have already been encoded). Now, the difference between the actual macro-block and its prediction is coded. This will result in a fewer bits to represent the macro-block of interest compared to when applying the transform directly to the macro-block itself.

To perform the intra prediction as mentioned above, H.264 offers nine modes for prediction of 4x4 luminance blocks, including DC prediction (Mode 2) and eight directional modes, labeled 0, 1, 3, 4, 5, 6, 7, and 8.

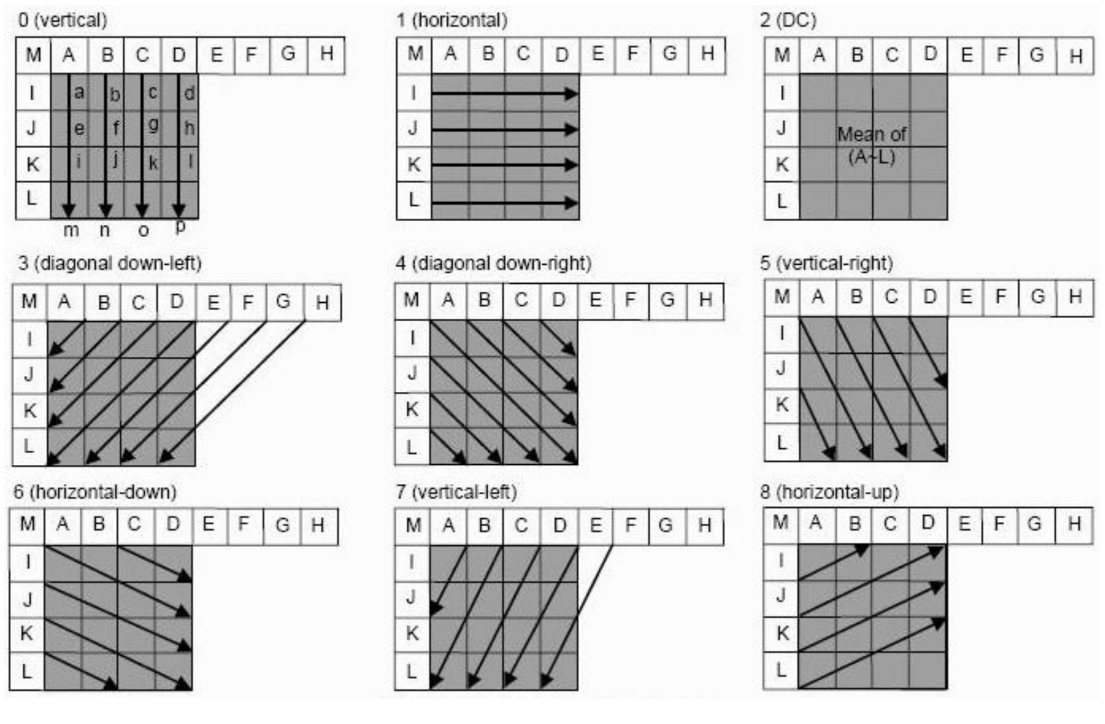


Figure 3.4 Intra 4x4 prediction mode directions (vertical, 0; horizontal, 1; DC, 2; diagonal down left, 3; diagonal down right, 4; vertical right, 5; horizontal down, 6; vertical left, 7; and horizontal up, 8) [27]

Figure 3.4 shows the eight directional modes for prediction of 4x4 luminance blocks. Pixels A to M are the pixels of the encoded neighboring blocks, which may be used for prediction. For example, if Mode 0 (vertical prediction) is selected, then the values of the pixels a to p are assigned as follows [11]:

- a, e, i and m are equal to A,
- b, f, j and n are equal to B,
- c, g, k and o are equal to C, and
- d, h, l and p are equal to D.

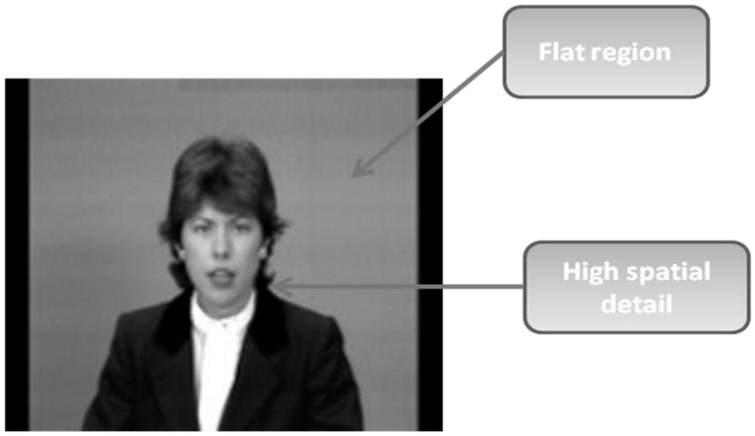


Figure 3.5 Frame with flat and highly detailed spatial regions [39]

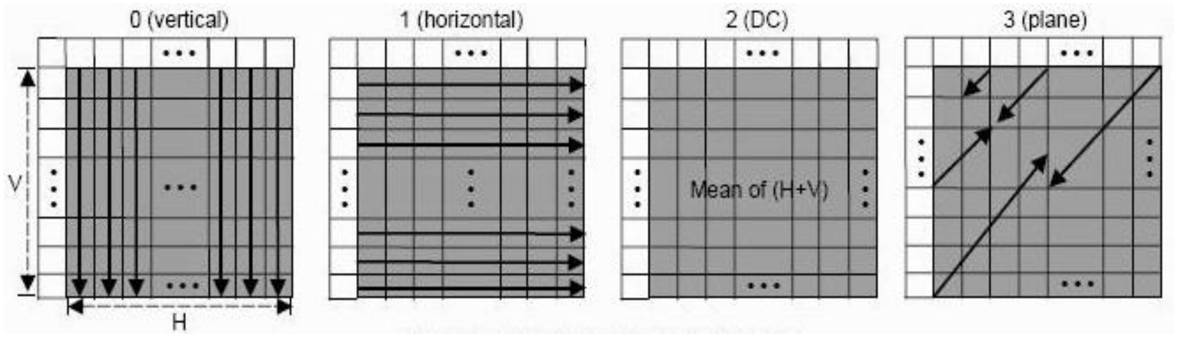


Figure 3.6 Prediction modes for 16x16 luma blocks (vertical, 0; horizontal, 1; DC, 2; Plane, 3) [27]

Frames of naturally occurring scenes are a fine mixture of detail, flat regions, shading, and texture. From Figure 3.5, we can observe that the background behind the lady is pretty much flat but, around the lady's neck region, there is a lot of detailing involved. Hence, small sized blocks are used in the highly detailed spatial region and in the case of flat regions (or less spatial detail), H.264 supports 16x16 sized luma blocks. H.264 offers four modes of intra prediction for 16x16 luminance block, including DC, vertical, horizontal and planar (Figure 3.6). Also, H.264 supports intra prediction for the 8x8 chrominance blocks, which uses four prediction modes similar to 16x16 luminance blocks. Finally, the prediction mode for each block is efficiently coded by assigning shorter symbols to more likely modes, where the probability of each mode is determined based on the modes used for coding the surrounding blocks.

### *3.2.2 Inter-Prediction*

In order to provide a very efficient coding, a technique called as inter-prediction is used. It uses motion estimation and compensation which takes advantage of the temporal redundancies that exist among successive frames. Temporal prediction involves predicting a frame from one or more past or future frames known as reference frames. The accuracy of the prediction can usually be improved by compensating for motion between the reference frame(s) and the current frame.

A practical and widely used method of motion compensation is to compensate for movement of rectangular sections or blocks of the current frame. The following procedure is carried out for each block of MxN samples in the current frame [3]:

- Search an area in the reference frame (past or future frame, previously coded and transmitted) to find a 'matching' M × N-sample region. This is carried out by comparing the *M* × *N* block in the current frame with some or all of the possible M × N regions in the search area (usually a region centered on the current block position) and finding the region that gives the ‘best’ match. A popular matching criterion is using the SAD (sum of absolute differences) which works by taking the absolute difference between each pixel in the original block and the corresponding pixel in the block being used for comparison, so that the candidate region with the least SAD is chosen as the best match. This process of finding the best match is known as motion estimation.

- The chosen candidate region becomes the predictor for the current M × N block and is subtracted from the current block to form a residual M × N block (motion compensation, MC).
- The residual block is transformed, encoded and transmitted and the offset between the current block and the position of the candidate region (motion vector, MV) is also transmitted.

The decoder uses the received motion vector to re-create the predictor region and decodes the residual block, adds it to the predictor and reconstructs a version of the original block.

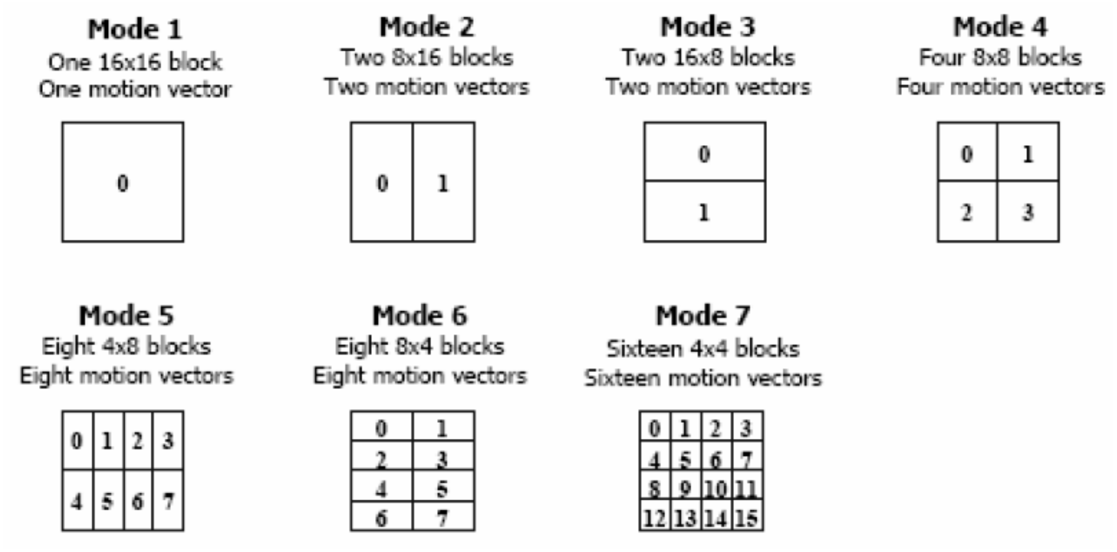


Figure 3.7 Different modes of dividing a macro-block for motion estimation in H.264 [11]

The macro-block, corresponding to 16x16-pixel region of a frame, is the basic unit for motion compensated prediction in a number of important visual coding standards including

MPEG-1, MPEG-2, MPEG-4 Visual, H.261, H.263 and H.264 [3]. In case of H.264, motion compensation for each 16x16 macro-block is performed using a number of different block sizes and shapes. These are illustrated in Figure 3.7. Individual motion vectors can be transmitted for blocks as small as 4x4, so up to 16 motion vectors may be transmitted for a single macro-block. The availability of smaller motion compensation blocks improves prediction in general, and in particular, the small blocks improve the ability of the model to handle fine motion detail and result in better subjective viewing quality because they do not produce large blocking artifacts. Figure 3.8 shows a configuration for a 16x16 macro-block.

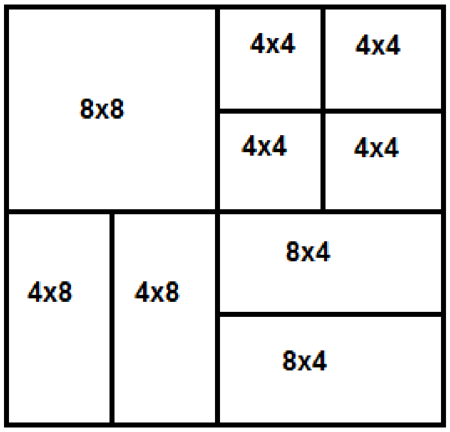


Figure 3.8 Example of 16x16 macro-block [11]

If your work contains more than four chapters, add additional template chapters to your template now before continuing.

3.2.2.1 Sub-Pixel Prediction

The prediction capability of the motion compensation algorithm in H.264 is further improved by allowing motion vectors to be determined with higher levels of spatial accuracy than in prior standards. Predicting from interpolated sample positions in the reference frame gives a better motion compensated prediction, producing a smaller residual at the expense of increased complexity.

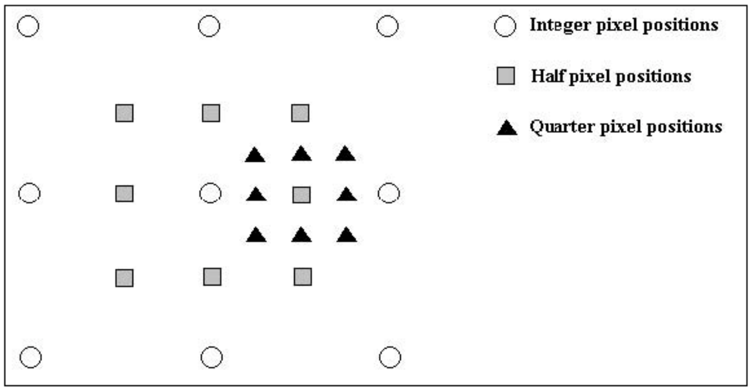


Figure 3.9 Example of integer and sub-pixel prediction [3]

Quarter-pixel accurate motion compensation is the lowest-accuracy form of motion compensation in H.264 (in contrast with prior standards based primarily on half-pixel accuracy, with quarter-pixel accuracy only available in the newest version of MPEG-4). Figure 3.9 shows an example of integer and sub-pixel prediction.

3.2.2.2 Skipped Mode

In addition to the macro-block modes described above, a P-slice macro-block can also be coded in the so-called skip mode. If a macro-block has motion characteristics that allow its motion to be effectively predicted from the motion of neighboring macro-blocks, and it contains no non-zero quantized transform coefficients, then it is flagged as skipped [30] [40]. For this mode, neither a quantized prediction error signal nor a motion vector or reference index parameter is transmitted. The reconstructed signal is computed in a manner similar to the prediction of a macro-block with partition size $16 \times 16$ and fixed reference picture index equal to 0. In contrast to previous video coding standards, the motion vector used for reconstructing a skipped macro-block is inferred from motion properties of neighboring macro-blocks rather than being inferred as zero (i.e., no motion).

#### 3.2.2.3 Weighted Prediction

In addition to the use of motion compensation and reference picture selection for prediction of the current picture content, weighted prediction can be used in P slices. When weighted prediction is used, customized weights can be applied as a scaling and offset to the motion-compensated prediction value prior to its use as a predictor for the current picture samples. Weighted prediction can be especially effective for such phenomena as "fade-in" and "fade-out" scenes.

#### 3.2.2.4 Motion Vector Prediction

After the temporal prediction, the steps of transform, quantization, scanning, and entropy coding are conceptually the same as those for I-slices for the coding of residual data (the original minus the predicted pixel values). The motion vectors and reference picture indexes representing the estimated motion are also compressed. Because, encoding a motion vector for each partition can take a significant number of bits, especially if small partition sizes are chosen. Motion vectors for neighboring partitions are often highly correlated and so each motion vector is predicted from vectors of nearby, previously coded partitions. A predicted motion vector, MVp, is formed based on previously calculated motion vectors. MVD, the difference between the current motion vector and the predicted motion vector, is encoded and transmitted. The method of forming the prediction MVp depends on the motion compensation partition size and on the availability of nearby vectors. The "basic" predictor is the median of the motion vectors of the macro-block partitions or sub-partitions immediately above, diagonally above and to the right, and immediately left of the current partition or sub-partition. The predictor is modified if (a) 16x8 or 8x16 partitions are chosen and/or (b) if some of the neighboring partitions are not available as predictors. If the current macro-block is skipped (not transmitted), a predicted vector is generated as if the MB was coded in 16x16 partition mode.

At the decoder, the predicted vector MVp is formed in the same way and added to the decoded vector difference MVD. In the case of a skipped macro-block, there is no decoded vector and so a motion compensated macro-block is produced according to the magnitude of MVp.

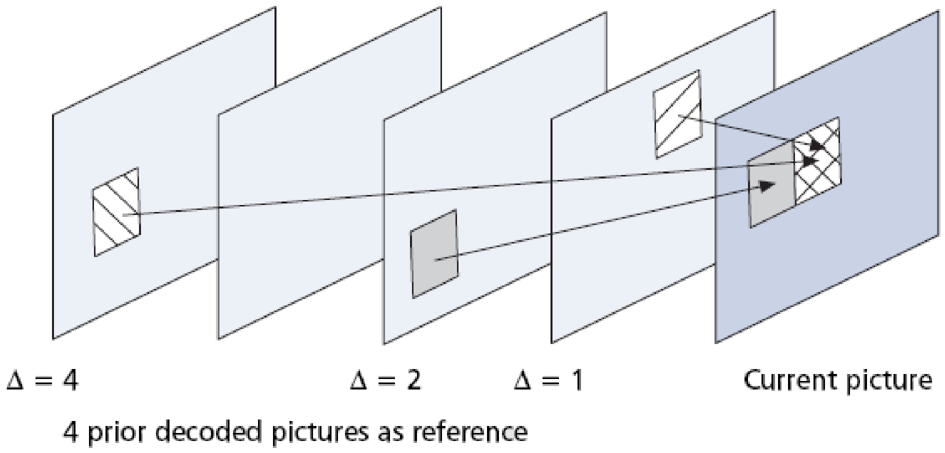
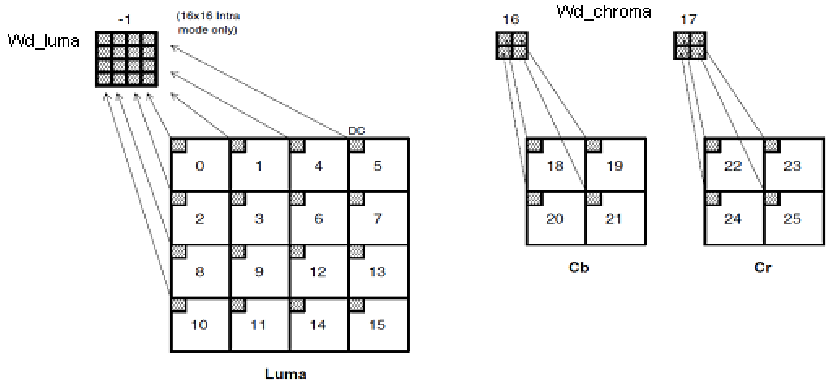


Figure 3.10 Motion compensated prediction with multiple reference frames [1]

3.2.2.5 Integer Transform

The information contained in a prediction error block resulting from either intra prediction or inter prediction is then re-expressed in the form of transform coefficients. H.264 employs a purely integer spatial transform (a rough approximation of the DCT [31]) which is primarily 4x4 in shape, as opposed to the usual floating-pint 8x8 DCT specified with rounding-error tolerances as used in earlier standards. The small shape helps reduce blocking and ringing artifacts, while the precise integer specification eliminates any mismatch issues between the encoder and decoder in the inverse transform. H.264 employs a hierarchical transform structure, in which the DC coefficients of neighboring 4x4 transforms for luma and chroma signals are grouped into 4x4 blocks (blocks -1, 16 and 17) and transformed again by the Hadamard transform as shown in Figure 3.7 (a).



(a)

$$\left(\begin{bmatrix}1 & 1 & 1 & 1\\ 2 & 1 & -1 & -2\\ 1 & -1 & -1 & 1\\ 1 & -2 & 2 & -1\end{bmatrix}\begin{bmatrix} & & & \\ & \mathbf{X} & & \\ & & & \\ & & & \end{bmatrix}\begin{bmatrix}1 & 2 & 1 & 1\\ 1 & 1 & -1 & -2\\ 1 & -1 & -1 & 2\\ 1 & -2 & 1 & -1\end{bmatrix}\right)$$

(b)

$$\mathbf{Y}_D = \left(\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}\begin{bmatrix} & & & \\ & \text{Wd\_luma} & & \\ & & & \\ & & & \end{bmatrix}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & 1 & -1 & -1\\ 1 & -1 & -1 & 1\\ 1 & -1 & 1 & -1\end{bmatrix}\right)/2$$

(c)

$$\mathbf{W}_{QD} = \begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}\text{Wd\_chroma}\end{bmatrix}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

(d)

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

(e)

Figure 3.11 H.264 Transformation (a)DC coefficients of 16 4x4 luma blocks for 4 4x4 $C_b$ and $C_r$ blocks [1] (b) Matrix H1 is applied to 4x4 block of luma/chroma coefficients X [27] (c) Matrix $H_2$ (4x4 Hadamard transform) applied to luma DC coefficients Wd_luma [27] (d) Matrix $H_3$ (2x2 Hadamard transform) applied to chroma DC coefficients Wd_chroma [27] (e) Matrices $H_1$, $H_2$ and $H_3$ of the three transforms used in H.264 [27]

As shown in Figure 3.7(b) the first transform (matrix $H_1$ in Figure 3.7(e)) is applied to all samples of all prediction error blocks of the luminance component (Y) and for all blocks of chrominance components ($C_b$ and $C_r$). For blocks with mostly flat pixel values, there are significant correlations among transform DC coefficients of neighboring blocks. Hence, the standard specifies the 4x4 Hadamard transform (matrix $H_2$ in Figure 3.7 (e)) for luma DC coefficients (Figure 3.7 (c)) for 16x16 intra-mode only, and 2x2 Hadamard transform as shown in Figure 3.7 (d) (matrix $H_3$ in Figure 3.7 (e)) for chroma DC coefficients [27] [28]

#### 3.2.2.6 Quantization and Transform Coefficient Scanning

The quantization step is where a significant portion of data compression takes place. In H.264, the transform coefficients are quantized using scalar quantization with no wide dead-zone. Unlike prior standards, fifty-two different quantization step sizes can be chosen on a macro-block basis. Moreover, in H.264 the step sizes are increased at a compounding rate of approximately 12.5% rather than increasing it by a constant increment [11]. The fidelity of chrominance components is improved by using finer quantization step sizes compared to those used for the luminance coefficients, particularly when the luminance coefficients are coarsely quantized.

The quantized transform coefficients correspond to different frequencies, with the coefficient at the top left hand corner in Figure 3.8 representing the DC value, and the rest of the coefficients corresponding to different non-zero frequency values. The next step in the encoding process is to arrange the quantized coefficients in an array, starting with the DC coefficient. A single coefficient-scanning pattern is available in H.264 (Figure 3.8) for frame coding, and another one is being added for field coding.

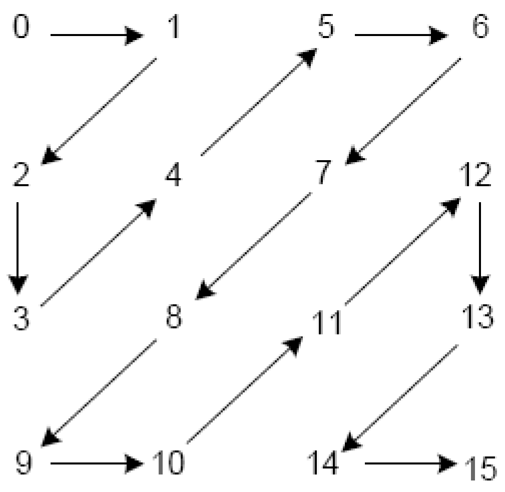


Figure 3.12 Scan pattern for frame coding in H.264

The zigzag scan illustrated in Figure 3.8 is used in all frame-coding cases, and it is identical to the conventional scan used in earlier video coding standards. The zigzag scan arranges the coefficient in an ascending order of the corresponding frequencies.

3.2.2.7 Deblocking Filter

The deblocking filter is used to remove the blocking artifacts due to the block based encoding pattern. The transform applied after intra-prediction or inter-prediction is on blocks; the transform coefficients then undergo quantization. These block based operations are responsible for blocking artifacts which are removed by the in-loop deblocking filter. It reduces the artifacts at the block boundaries and prevents the propagation of accumulated noise. The presence of the filter however adds to the complexity of the system [1]. Figure 3.9 illustrates a macro-block with sixteen 4x4 sub blocks along with their boundaries.

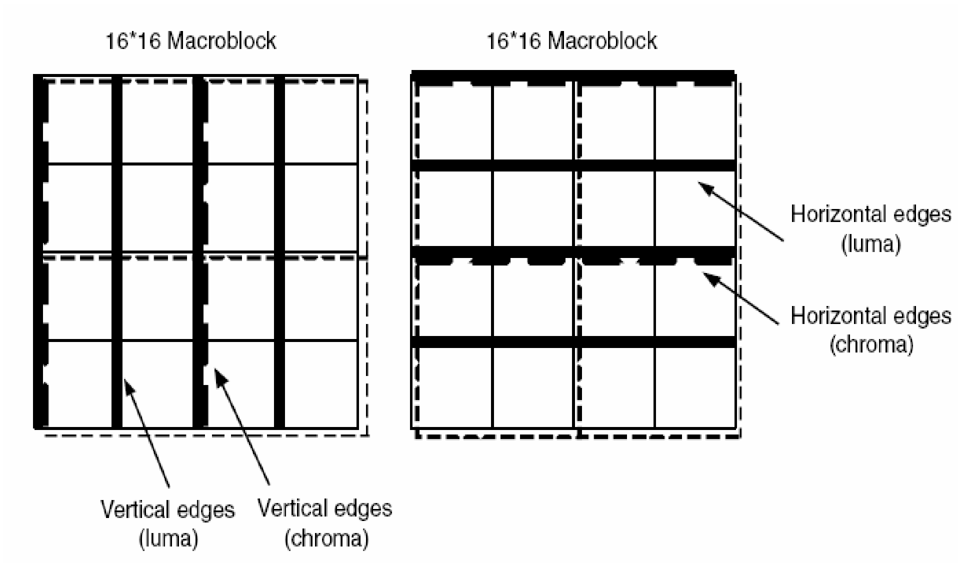


Figure 3.13 Boundaries in a macro-block to be filtered (luma boundaries shown with solid lines and chroma boundaries shown with dotted lines) [27]

As shown in the Figure 3.9, the luma deblocking filter process is performed on the 16 sample edges – shown by solid lines. The chroma deblocking filter process is performed on 8 sample edges – shown in dotted lines.

H.264 employs deblocking process adaptively at the following three levels [27]:

- 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.

3.2.2.8 Entropy Coding

The last step in the video coding process is entropy coding. Entropy coding is based on assigning shorter code-words to symbols with higher probabilities of occurrence, and longer code-words to symbols with less frequent occurrences. Some of the parameters to be entropy coded include transform coefficients for the residual data, motion vectors and the other encoder information. Two types of entropy coding have been adopted. The first method represents a combination of universal variable length coding (UVLC) and context adaptive variable-length coding (CAVLC). The second method is represented by context-based adaptive binary arithmetic coding (CABAC).

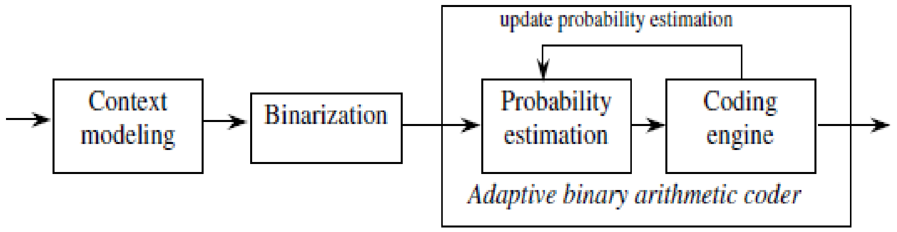


Figure 3.14 Schematic block diagram for CABAC [27]

If your work has more than five chapters, repeat the previous chapter text selection and copy operations and chapter paste and rename processes until you have created a template that has as many chapters as your work has.

The characteristics of each coding method are explained below [29]:

1. Context Adaptive Variable Length Coding

   i. No end-of block, but number of coefficients is decoded.
   ii. Coefficients are scanned backwards and contexts are built depending on transform coefficients.
   iii. Transform coefficients are coded with the following elements: number of non-zero coefficients, levels and signs for all non-zero coefficients,

total number of zeros before the last non-zero coefficient, and number of successive zeros preceding the last non-zero coefficient.

iv. The VLC table to use is adaptively chosen based on the number of coefficients in the neighboring blocks.

2. Context Adaptive Binary Arithmetic Coding

i. Overview of CABAC is shown in Fig 3.10
ii. Usage of adaptive probability models for most symbols.
iii. Exploiting symbol correlations by using contexts.
iv. Discriminate between binary decisions by their positions in the binary sequence.
v. Probability estimation is realized via the look up table.

## 3.3 H.264 Profiles

H.264 describes three profiles in the first version: Baseline, Main and Extended. Baseline profile is to be applicable to real-time conversational services such as video conferencing and videophone. Main profile is designed for digital storage media and television broadcasting. Extended profile targets multimedia services over the internet. Additionally, there are four high profiles (Figure 3.11) defined in the fidelity range extensions [30] for applications such as content-contribution, content-distribution, and studio editing and post-processing: High, High 10, High 4:2:2, and High 4:4:4.

- High profile is to support the 8-bit video with 4:2:0 sampling for applications using high resolution.
- High 10 profile is to support the 4:2:0 sampling up to 10 bits of representation accuracy per sample.

- High 4:2:2 profile is to support up to 4:2:2 chroma sampling and up to 10 bits per sample.
- High 4:4:4 profile is to support up to 4:4:4 chroma sampling, up to 12 bits per sample and integer residual color transform for coding RGB signal.

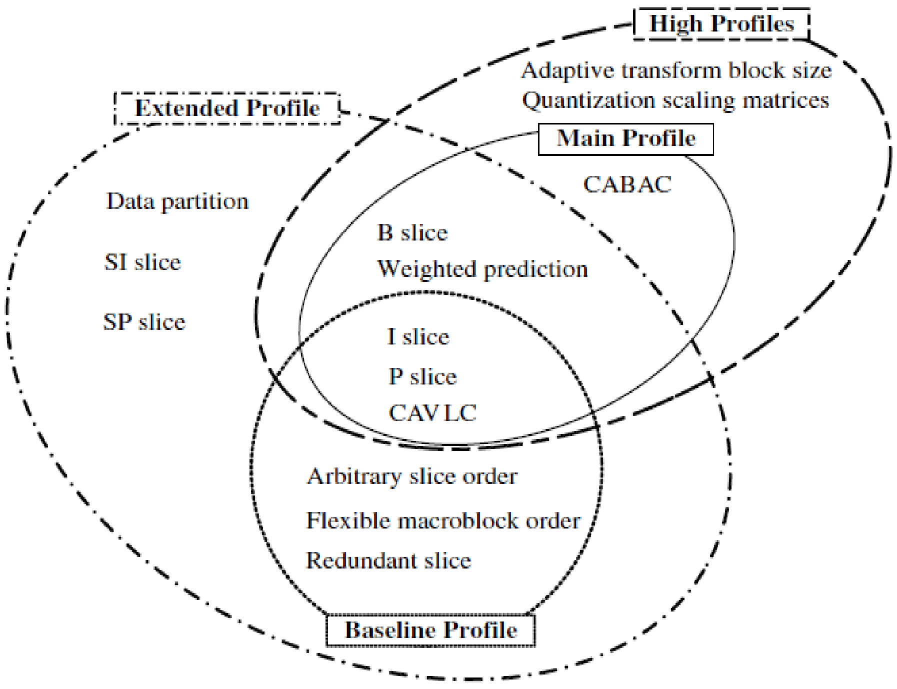


Figure 3.15 Specific coding parts of the profiles in H.264 [27]

*3.3.1 Coding Parts of H.264 [27]*

Figure 3.11 shows that the profiles have both common coding parts and specific coding parts. Some of the common parts of all profiles are:

- I slice (Intra-coded slice): the coded slice by using prediction only from decoded samples within the same slice.

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

Table 3.1 lists the H.264 profiles and important requirements for each application.

Table 3.1 H.264 profiles and important requirements for each application [1] [27]

| **Application** | **Requirements** | **H.264 Profiles** |
|---|---|---|
| Broadcast television | Coding efficiency, reliability (over a controlled distribution channel), interlace, low-complexity decoder | Main |
| Streaming video | Coding efficiency, reliability (over a uncontrolled packet-based network channel), scalability | Extended |
| Video storage and Playback | Coding efficiency, interlace, low-complexity encoder and decoder | Main |
| Videoconferencing | Coding efficiency, reliability, low latency, low-complexity encoder and decoder | Baseline |
| Mobile video | Coding efficiency, reliability, low latency, low-complexity encoder and decoder, low power consumption | Baseline |
| Studio distribution | Lossless or near-lossless, interlace, efficient transcoding | Main, High Profiles |

#### 3.3.1.1 Baseline Profile

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

#### 3.3.1.2 Main Profile

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

#### 3.3.1.3 Extended Profile

- Includes all parts of Baseline Profile: flexible macroblock order, arbitrary slice order, redundant slice
- SP slice: the specially coded slice for efficient switching between video streams, similar to coding of a P slice.
- SI slice: the switched slice, similar to coding of an I slice.
- Data partition: the coded data is placed in separate data partitions, each partition can be placed in a different layer unit.

- B slice
- Weighted prediction

### 3.3.1.4 High Profiles

- Includes all parts of Main Profile: B slice, weighted prediction, CABAC
- Adaptive transform block size: 4x4 or 8x8 integer transform for luma samples.
- Quantization scaling matrices: different scaling according to specific frequency associated with the transform coefficients in the quantization process to optimize quality.

# CHAPTER 4

# COMPLEXITY REDUCTION ALGORITHM FOR INTRA MODE SELECTION

## 4.1 Introduction

The existing intra prediction algorithm in H.264 using Rate Distortion Optimization (RDO) examines all possible combinations of coding modes, which depend on spatial directional correlation with adjacent blocks. There are 9 modes for a 4x4 luma block (Fig. 3.4), and 4 modes for a 16x16 luma block and an 8x8 chroma block (Fig. 3.6), respectively. Therefore the number of mode combinations for each MB sums up to 592. This thesis adopts a complexity reduction algorithm using simple directional masks and neighboring modes. This algorithm reduces the number of mode combinations into 132 at the most, with negligible loss of PSNR, SSIM and bit-rate increase compared with the H.264/AVC exhaustive search [16].

## 4.2 Selection of the Best Mode using Rate Distortion Optimization

The H.264/AVC intra-prediction is conducted for all types of blocks such as 4x4 luma blocks, 16x16 luma blocks, and 8x8 chroma blocks. The residual between the current block and its prediction is then transformed, quantized, and entropy coded.

To obtain the best mode among these modes, the H.264/AVC encoder performs the rate-distortion optimization (RDO) technique for each macro block [38] [41].

Set macro-block parameters: QP (quantization parameter) and Lagrangian multiplier $\lambda$MODE

Calculate: $\lambda MODE = 0.85x\ 2^{(QP-12)/3}$ [45] (4.1)

Then calculate the cost, which determines the best mode

$$\text{Cost} = D + \lambda\ MODE \times R, \tag{4.2}$$

D = Distortion

R = Bit rate with given QP

Distortion (D) is obtained by SSD (sum of squared differences) between the original macro block and its reconstructed block.

Bit-rate(R) includes the number of bits for the mode information and transform coefficients for macro block.

Considering the RDO procedure for intra mode selection in H.264/AVC, the number of mode combinations in one macro block is N8x (16xN4 + N16)

N8 – number of modes of an 8x8 chroma block

N4 – number of modes of a 4x4 luma block

N16 – number of modes of a 16x16 luma block

Computing the best mode for one macro block:

$$N8 \times (16 \times N4 + N16) = 4 \times (16 \times 9 + 4) = 592$$

Thus, to select the best mode for one macro-block in the intra prediction, the H.264/AVC encoder carries out 592 RDO calculations. As a result, the complexity of the encoder increases extremely [16].

## 4.3 Proposed Intra Mode Selection Algorithm for a 4x4 Luma Block

In Figure 4.1, black dots indicate positions of the pixels to be computed for computing directional correlation in the 4x4 luma block, and arrows represent the directions of correlation associated with the corresponding mask.

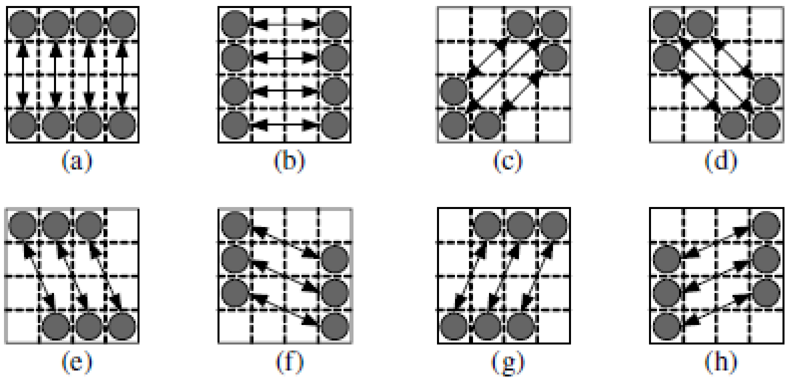


Figure 4.1 Proposed directional masks for a 4x4 luma block. (a) vertical, (b) horizontal, (c)diagonal down left, (d) diagonal down right, (e) vertical right, (f) horizontal down, (g) vertical left, (h) horizontal up mask [16].

Since directions of the H.264/AVC intra-prediction are limited to 8 directions except DC mode, this thesis proposes 8 directional masks instead of a precise edge detector such as Sobel operator. One candidate mode with the minimum Diff is selected using [16]:

$$\text{Diff} = |a - m| + |b - n| + |c - o| + |d - p|, \quad \text{for vertical direction} \qquad (4.3)$$

$$\text{Diff} = |a - d| + |e - h| + |i - l| + |m - p|, \quad \text{for horizontal direction} \qquad (4.4)$$

$$\text{Diff} = |c - i| + 2\cdot|d - m| + |h - n|, \quad \text{for diagonal down left direction} \qquad (4.5)$$

$$\text{Diff} = |b - l| + 2\cdot|a - p| + |e - o|, \quad \text{for diagonal down right direction} \qquad (4.6)$$

$$\text{Diff} = |a - n| + 2\cdot|b - o| + |c - p|, \quad \text{for vertical right direction} \qquad (4.7)$$

$$\text{Diff} = |a - h| + 2\cdot|e - l| + |i - p|, \quad \text{for horizontal down direction} \qquad (4.8)$$

$$\text{Diff} = |b - m| + 2\cdot|c - n| + |d - o|, \quad \text{for vertical left direction} \qquad (4.9)$$

$$\text{Diff} = |e - d| + 2\cdot|i - h| + |m - l|, \quad \text{for horizontal up direction} \qquad (4.10)$$

where a to p denote the pixels for computing directional correlation associated with the corresponding mask of Figure 4.1. Figure 4.2(a) shows the indices for pixel positions used in (4.3) to (4.10). Diff is used as a criterion for correlation, i.e., the direction with smaller Diff is more correlated one. From the second observation, additional candidate modes using mode information of adjacent blocks can also be obtained, where one is the upper block with the corresponding mode of modeA and the other is the left block with the corresponding mode of modeB, as shown in Figure 4.2(b). These additional modes, i.e., modeA and modeB, are included as candidate modes for RDO procedure, since the directions in the H.264/AVC intra-prediction are defined with the directional relation between current block and boundary pixels of adjacent blocks, instead of direction within the current block only. In this case, one mode when modeA and modeB are same, or two modes when modeA and modeB are different from each other, are included in RDO procedure.

To determine whether the DC mode is included in the RDO procedure or not, a sum of difference, denoted by S in (4.11), between an average of current block, denoted by the avg in (4.11), and each pixel, denoted by $p_i$ in (4.11) [16].

$$S = \sum_{i=0}^{15} | \mathrm{avg} - p_i | \tag{4.11}$$

where $\mathrm{avg} = \left(\sum_{i=0}^{15} p_i + 8)\right) >> 4$, and $p_i$ is each pixel of current block.

If S is smaller than a threshold, T1 (set to 32), RDO for at most 4 candidate modes is carried out, i.e., one mode from the proposed masks, at most two modes from adjacent blocks, and DC mode. If S is larger than a threshold, T1, RDO for at most 4 candidate modes is carried out, i.e., two modes from the proposed masks (with minimum and second minimum Diff) and at most two modes from adjacent blocks. The proposed intra mode selection algorithm for a 4x4 luma block is summarized as follows [16]:

- Step 1 - For a 4x4 luma block, obtain avg and S by (4.11).
- Step 2a - If S is larger than a threshold, T1, carry out RDO procedure for at most 4

candidate modes: two modes with minimum and second minimum Diff by (4.3) to (4.10), and at most two modes from adjacent blocks. In this case, DC mode of adjacent blocks is excluded from RDO procedure.

- Step 2b - If S is smaller than a threshold, T1, carry out RDO procedure for at most 4 candidate modes: one mode with minimum Diff by (4.3) to (4.10), at most two modes from adjacent blocks, and DC mode.

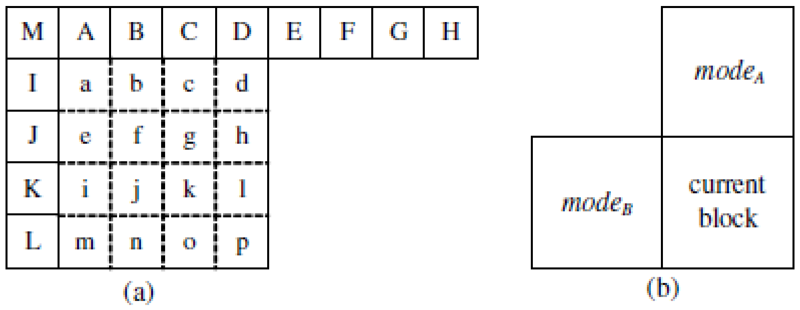


Figure 4.2 Pixel indices and modes of adjacent blocks used in the proposed intra mode selection algorithm. (a) indices used in (3) to (10) for a 4x4 luma block, (b) modes of upper and left blocks for additional candidate modes [16].

## 4.4 Proposed Intra Mode Selection Algorithm for a 16x16 Luma Block

- Step 1 - Examine sizes of adjacent blocks: if both blocks (upper block and left block) are 16x16, go to Step 2, otherwise go to Step 4.
- Step 2 - Examine modes of adjacent blocks: if both modes are the same, go to Step 3, otherwise select the best mode for a 16x16 luma block, which results in the minimum SATD (sum of absolute transformed differences) between two adjacent modes of modeA and modeB.
- Step 3 - If both adjacent modes are the DC mode, go to Step 4, otherwise select the best mode for a 16x16 luma block, which results in the minimum SATD between the adjacent mode and DC mode.

- Step 4 - Let ΔV be a vertical difference between upper boundary pixels of the current block and boundary pixels of the upper block, and ΔH be a horizontal difference between left boundary pixels of the current block and boundary pixels of the left block as follows.

$$\Delta \mathrm{V} = \sum_{i=0}^{15} | \mathrm{u_i} - q_i |, \ \Delta \mathrm{H} = \sum_{i=0}^{15} | l_\mathrm{i} - \mathrm{r_i} | \qquad (4.12)$$

where $u_i$ - upper block boundary pixels

$q_i$ - upper boundary pixels of current block

$l_i$ - boundary pixels of the left block

$r_i$ - left boundary pixels of the current block

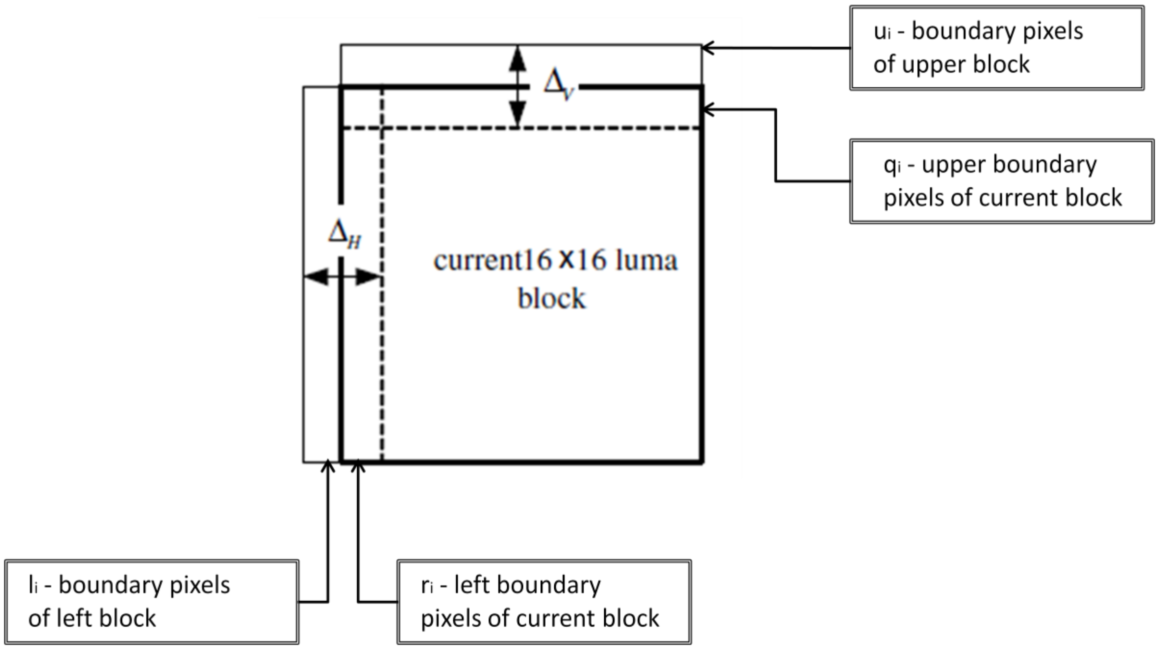

$u_i$ - boundary pixels of upper block
$q_i$ - upper boundary pixels of current block
$\Delta_V$
$\Delta_H$
current16 X16 luma block
$l_i$ - boundary pixels of left block
$r_i$ - left boundary pixels of current block

Figure 4.3 definition of ΔV and ΔH in 16x16 luma block. [16]

- Step 5 - Obtain candidate modes by using two difference values, ΔV and ΔH: if |ΔV − ΔH | is smaller than 2xT2, candidate modes are the DC mode and the plane mode; if

(ΔV − ΔH) is larger than T2, the candidate modes are the DC mode and the horizontal mode; if (ΔV − ΔH) is smaller than T2, candidate modes are the DC and the vertical mode, where T2 is a positive value.

The threshold T2 is set equal to 8. Finally, select the best mode among all candidate modes by choosing the mode with minimum SATD.

## 4.5 Experimental Results

In order to evaluate the proposed thesis, JM 18.0 [17] provided by JVT (joint video team) under H.264 / AVC baseline profile, with RD optimization and Hadamard transform [27] turned on is used. The simulations for test sequences of bridge-close, bridge-far, coastguard, container, and mobile with QCIF (176x144) and CIF (352x288) resolution was carried out.

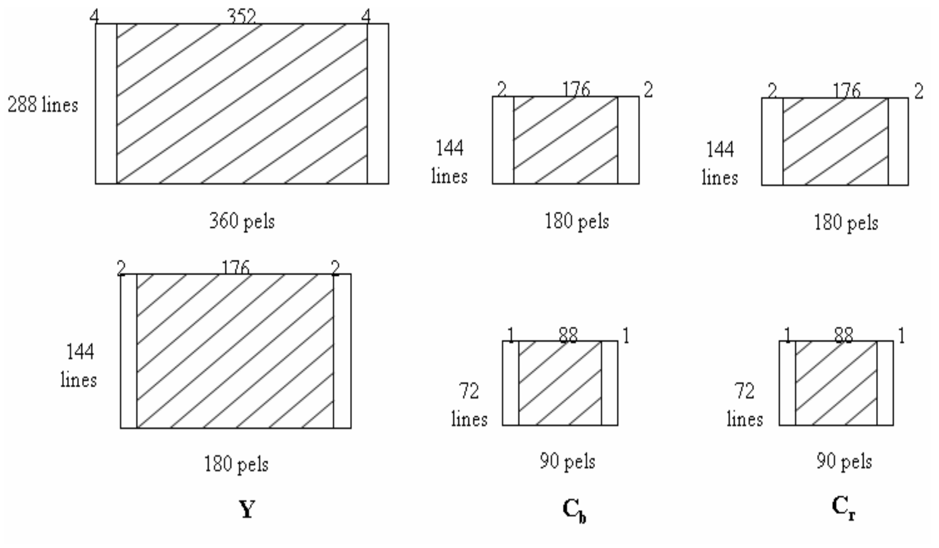


Figure 4.4 4:2:0 format of cif and qcif

Various QP (quantization parameter) of 10, 20, 28, 34, and 40 with I-only type, wherein for QCIF (176x144) resolution, 100 frames and for CIF (352x288) resolution, 30 frames were used. The results were compared with the case of exhaustive search in terms of the change of PSNR(ΔPSNR), bit-rate(ΔBit-rate), SSIM(ΔSSIM), compression ratio(ΔCompression-ratio), and encoding time (ΔT), with the machine of Intel Pentium Dual Core processor of 2GHz and 2GB memory

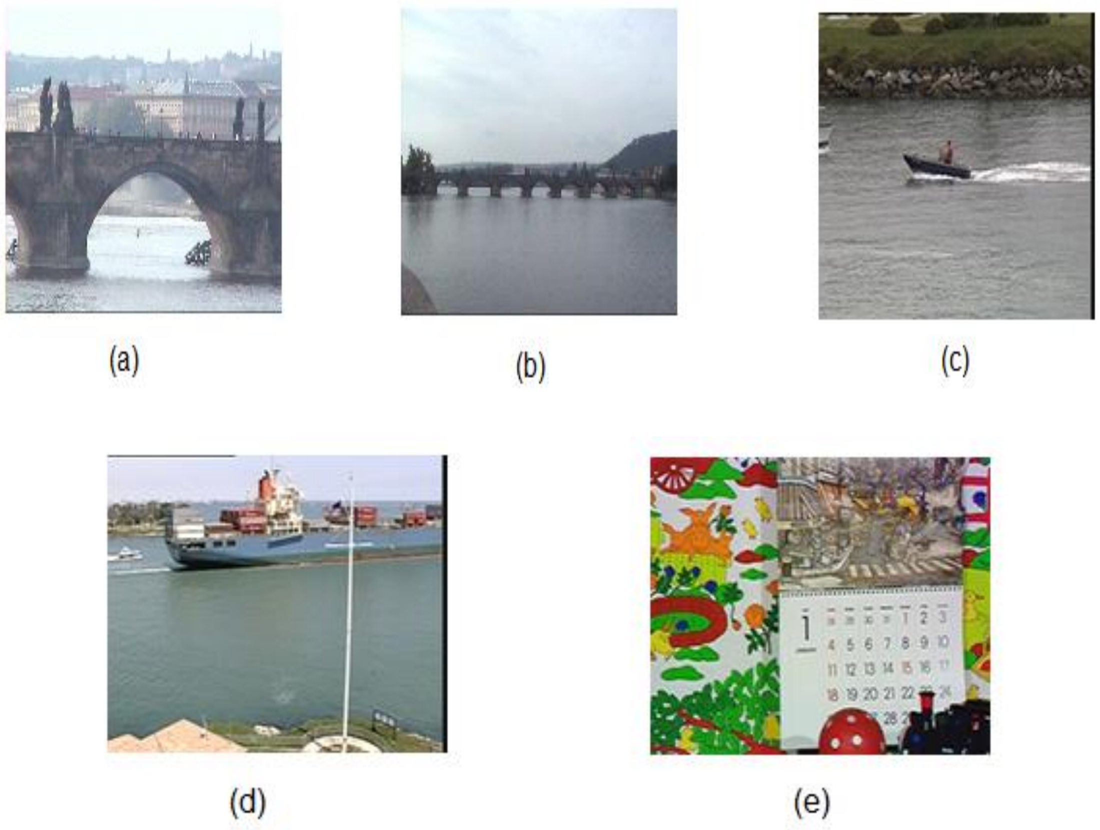


Figure 4.5 CIF and QCIF video sequences (a) bridge-close (b) bridge-far (c) coastguard (d) container (e) mobile [39]

Table 4.1 through 4.3 shows the simulation results of QCIF videos at various QP values. From this table it can be observed that, a maximum of 35.81% of encoding time was saved in one of the cases with negligible PSNR, SSIM, and bit-rate.

Table 4.1 Simulation results for QCIF videos at QP=10, 20

| Sequence (QCIF) | QP=10 | | | | QP=20 | | | |
|---|---|---|---|---|---|---|---|---|
| | ΔT (%) | ΔPSNR (dB) | ΔBit-rate (%) | ΔSSIM (%) | ΔT (%) | ΔPSNR (dB) | ΔBit-rate (%) | ΔSSIM (%) |
| bridge-close | -22.64 | -0.008 | 0 | -0.8125 | -23.56 | -0.014 | 0 | -1.4134 |
| bridge-far | -23.76 | -0.013 | 0 | -0.6622 | -26.07 | -0.047 | 0.02054 | -1.9981 |
| coastguard | -22.26 | -0.005 | 0 | -1.2691 | -17.7 | -0.006 | 0 | -2.1731 |
| container | -27.21 | -0.002 | 0 | -2.3724 | -35.81 | -0.001 | 0.505919 | -3.3028 |
| mobile | -23.67 | -0.004 | 0 | -1.1081 | -20.17 | -0.001 | 0 | -1.2988 |

Table 4.2 Simulation results for QCIF videos at QP=28, 34

| Sequence (QCIF) | QP=28 | | | | QP=34 | | | |
|---|---|---|---|---|---|---|---|---|
| | ΔT (%) | ΔPSNR (dB) | ΔBit-rate (%) | ΔSSIM (%) | ΔT (%) | ΔPSNR (dB) | ΔBit-rate (%) | ΔSSIM (%) |
| bridge-close | -23.26 | -0.004 | 0.010516 | -3.6515 | -24.28 | -0.041 | 0.169837 | -3.7720 |
| bridge-far | -23.04 | -0.051 | 0.021229 | -9.9030 | -23.88 | -0.111 | 0.163079 | -16.382 |
| coastguard | -21.84 | -0.02 | 0 | -3.7128 | -21.41 | -0.002 | 0.057399 | -5.0203 |
| container | -33.93 | -0.011 | 0.02096 | -5.4592 | -31.93 | -0.011 | 0.05408 | -7.3856 |
| mobile | -27.31 | -0.004 | 0 | -1.9034 | -28.08 | -0.006 | 0.021166 | -3.7720 |

Table 4.3 Simulation results for QCIF videos at QP=40

| Sequence (QCIF) | QP=40 | | | |
|---|---|---|---|---|
| | ΔT (%) | ΔPSNR (dB) | ΔBit-rate (%) | ΔSSIM (%) |
| bridge-close | -25.1496 | -0.001 | 0.180134 | -9.46485 |
| bridge-far | -22.4558 | -0.473 | 0.901104 | -26.1406 |
| coastguard | -22.6402 | -0.017 | 0.306151 | -7.47781 |
| container | -28.722 | -0.061 | 0.057293 | -7.86475 |
| mobile | -27.1755 | -0.006 | 0.012061 | -4.15416 |

Table 4.4 through 4.6 shows the simulation results of CIF videos at various QP values. From this table it can be observed that, a maximum of 31.29% of encoding time was saved in one of the cases with negligible PSNR, SSIM, and bit-rate

Table 4.4 Simulation results for CIF videos at QP=10, 20

| Sequence (CIF) | QP=10 | | | | QP=20 | | | |
|---|---|---|---|---|---|---|---|---|
| | ΔT (%) | ΔPSNR (dB) | ΔBit-rate (%) | ΔSSIM (%) | ΔT (%) | ΔPSNR (dB) | ΔBit-rate (%) | ΔSSIM (%) |
| bridge-close | -21.69 | -0.006 | 0 | -0.5680 | -22.13 | 0 | 0 | -1.2631 |
| bridge-far | -21.82 | -0.005 | 0 | -0.3171 | -20.84 | -0.001 | 0 | -2.1913 |
| coastguard | -24.12 | -0.001 | 0 | -0.991 | -26.73 | -0.02 | 0 | -2.0910 |
| container | -26.3 | -0.012 | 0 | -2.2394 | -25.1 | -0.01 | 0 | -3.7458 |
| mobile | -29.4 | -0.005 | 0 | -1.4101 | -28.52 | -0.004 | 0.010057 | -1.8272 |

Table 4.5 Simulation results for CIF videos at QP=28, 34

| Sequence (CIF) | QP=28 | | | | QP=34 | | | |
|---|---|---|---|---|---|---|---|---|
| | ΔT (%) | ΔPSNR (dB) | ΔBit-rate (%) | ΔSSIM (%) | ΔT (%) | ΔPSNR (dB) | ΔBit-rate (%) | ΔSSIM (%) |
| bridge-close | -22.12 | -0.006 | 0.031807 | -3.2028 | -21.26 | -0.007 | 0.034294 | -4.1608 |
| bridge-far | -23.26 | -0.004 | 0.010516 | -3.6515 | -23.32 | -0.047 | 0.034037 | -19.814 |
| coastguard | -23.04 | -0.051 | 0.021229 | -9.9030 | -24.39 | -0.003 | 0.129002 | -5.1468 |
| container | -21.84 | -0.02 | 0 | -3.7128 | -25.55 | -0.012 | 0 | -7.7552 |
| mobile | -31.11 | 0 | 0.01022 | -2.6653 | -31.29 | -0.002 | 0.05305 | -3.8336 |

Table 4.6 Simulation results for CIF videos at QP=40

| Sequence (CIF) | QP=40 | | | |
|---|---|---|---|---|
| | ΔT (%) | ΔPSNR (dB) | ΔBit-rate (%) | ΔSSIM (%) |
| bridge-close | -22.0166 | -0.073 | 0.417591 | -7.78436 |
| bridge-far | -23.2049 | -0.287 | 0.234797 | -19.8696 |
| coastguard | -25.2698 | -0.025 | 0.659275 | -8.57259 |
| container | -27.2056 | -0.047 | 0.143455 | -10.7092 |
| mobile | -27.2989 | -0.02 | 0.247146 | -5.07932 |

Figure 4.5 shows the comparison of encoding time taken by the reference software JM and complexity reduced encoder. The results were taken at QP = 28, which is the default configuration value. It can be observed that on an average the complexity reduced encoder takes 24.93% lesser time than the JM reference software.

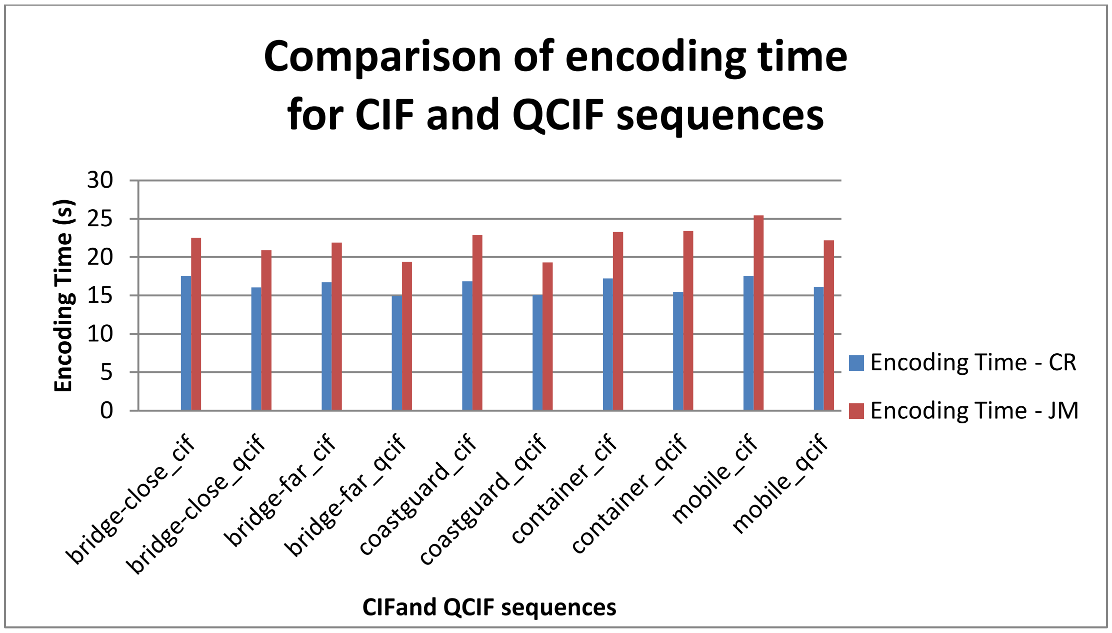


Figure 4.6 Comparison of encoding time for CIF and QCIF sequences

Figure 4.6 shows the comparison of PSNR values of the reference software JM and complexity reduced encoder. The results were taken at QP = 28, which is the default configuration value. It can be observed that there is not much of decrease in PSNR value in complexity reduced encoder's results when compared to JM reference software results.

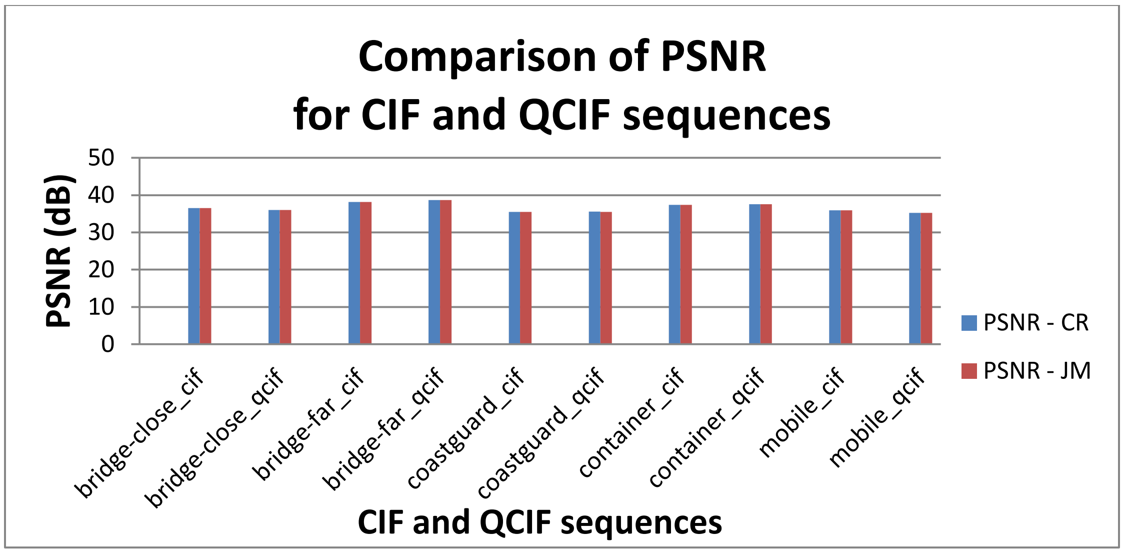


Figure 4.7 Comparison of PSNR values for CIF and QCIF sequences

Figure 4.7 shows the comparison of bit-rate values of the reference software JM and complexity reduced encoder. The results were taken at QP = 28, which is the default configuration value. It can be observed that there is not much of increase in bit-rate in complexity reduced encoder's results when compared to JM reference software results.

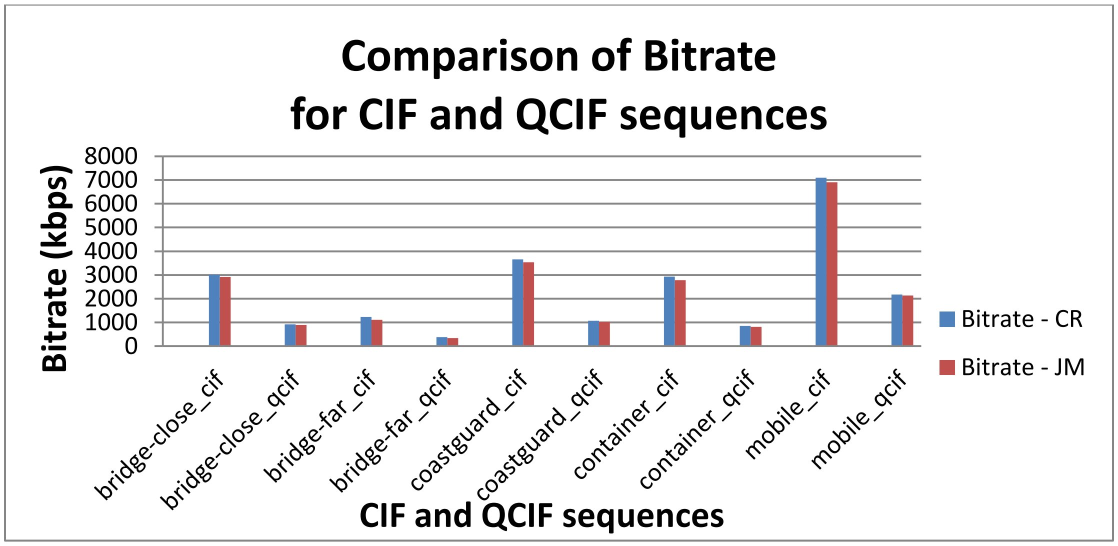


Figure 4.8 Comparison of bit-rates for CIF and QCIF sequences

Figure 4.8 shows the comparison of SSIM values of the reference software JM and complexity reduced encoder. The results were taken at QP = 28, 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.

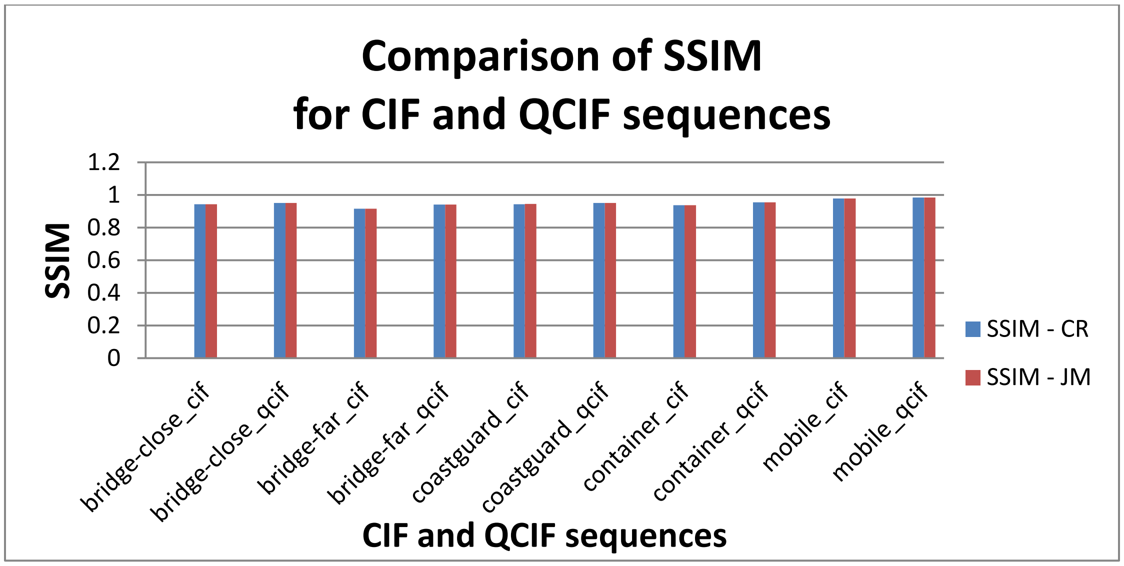


Figure 4.9 Comparison of SSIM for CIF and QCIF sequences

Figure 4.9 shows the comparison of the compression ratio of the reference software JM and complexity reduced encoder. The results were taken at QP = 28, which is the default configuration value. It can be observed that there is not much of decrease in the compression ratio in complexity reduced results of the encoder when compared to the JM reference software results.

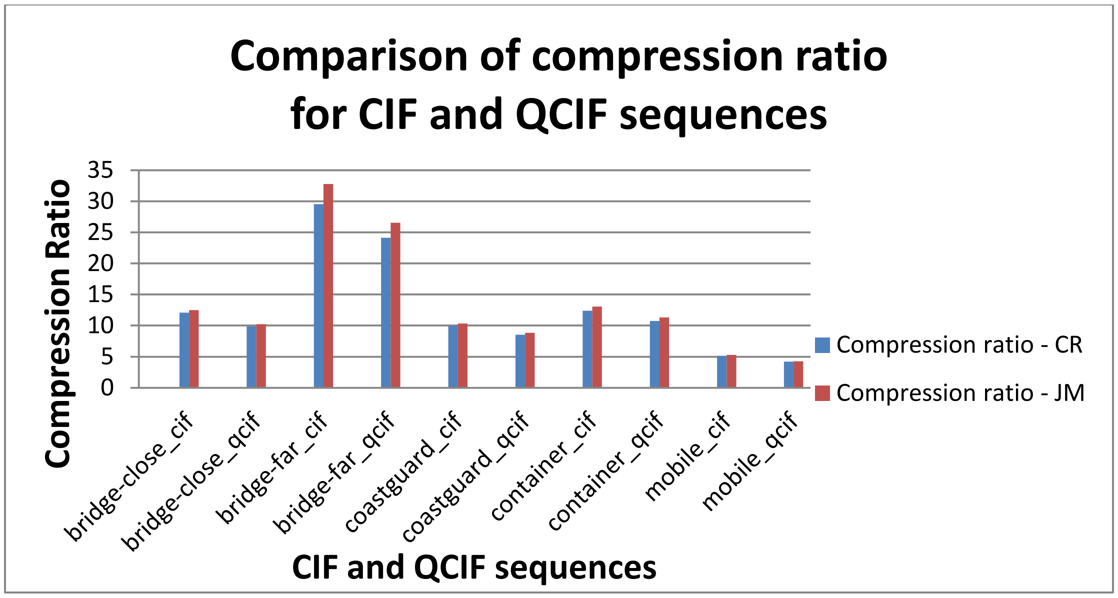


Figure 4.10 Comparison of compression ratio for CIF and QCIF sequences

Figure 4.10 shows a plot of encoding time Vs QP, comparing encoding time of the reference software JM and complexity reduced encoder against different values of QP for QCIF format, container_qcif video sequence. It can be observed from the curve that on an average the complexity reduced encoder takes around 24.93% lesser time than the JM reference software.

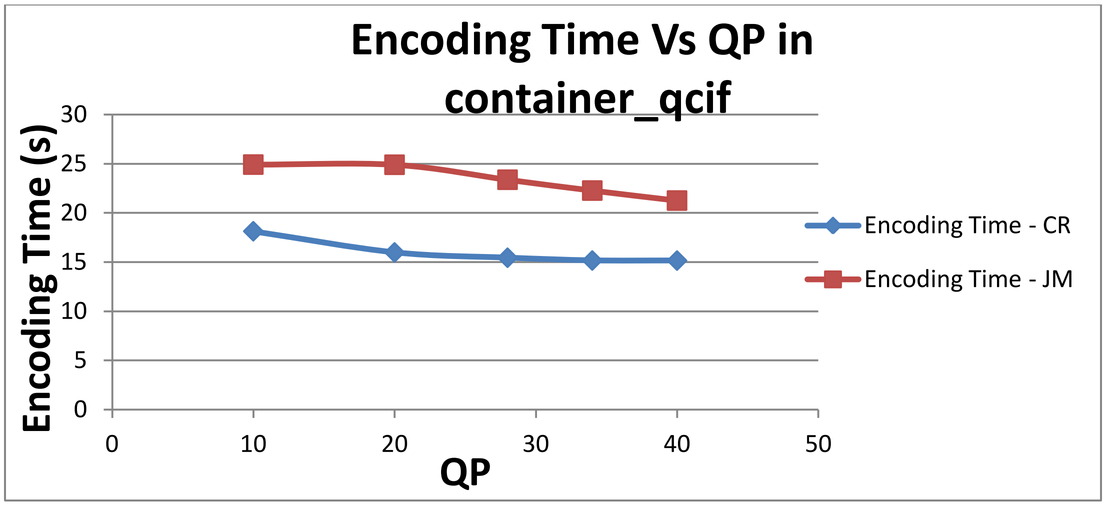


Figure 4.11 Plot of encoding time Vs QP for container_qcif video sequence

Figure 4.11 shows a plot of PSNR Vs QP, comparing PSNR values of the reference software JM and complexity reduced encoder against different values of QP for QCIF format, container_qcif video sequence. It can be observed from the curve that the PSNR values of complexity reduced encoder do not decrease significantly when compared to JM reference software's PSNR values.

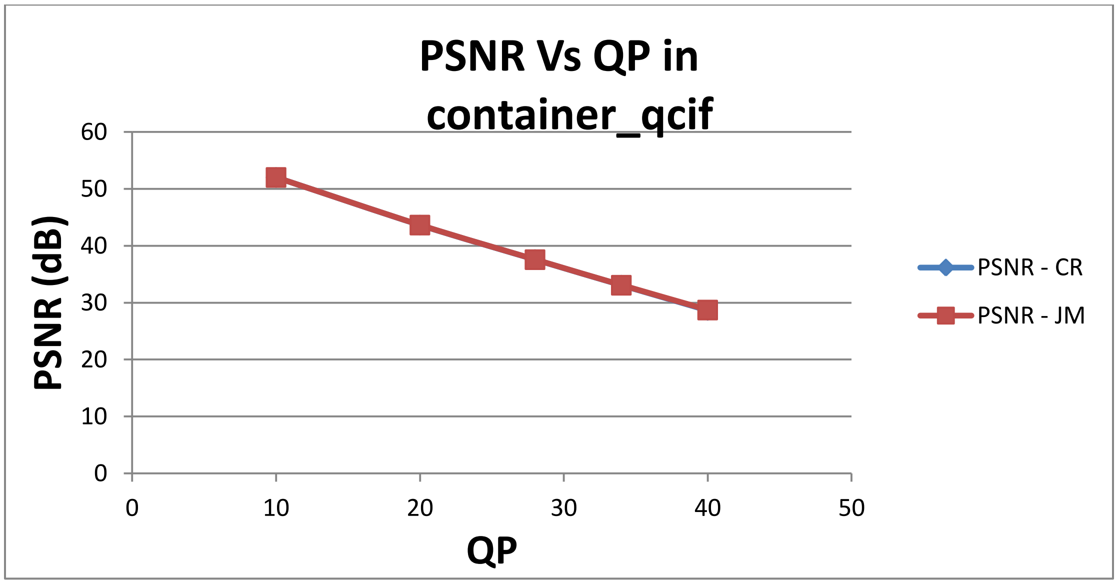


Figure 4.12 Plot of PSNR Vs QP for container_qcif video sequence

Figure 4.12 shows a plot of SSIM Vs QP, comparing SSIM values of the reference software JM and complexity reduced encoder against different values of QP for QCIF format, container_qcif video sequence. It can be observed from the curve that the SSIM values of complexity reduced encoder do not decrease significantly when compared to the JM reference software's SSIM values.

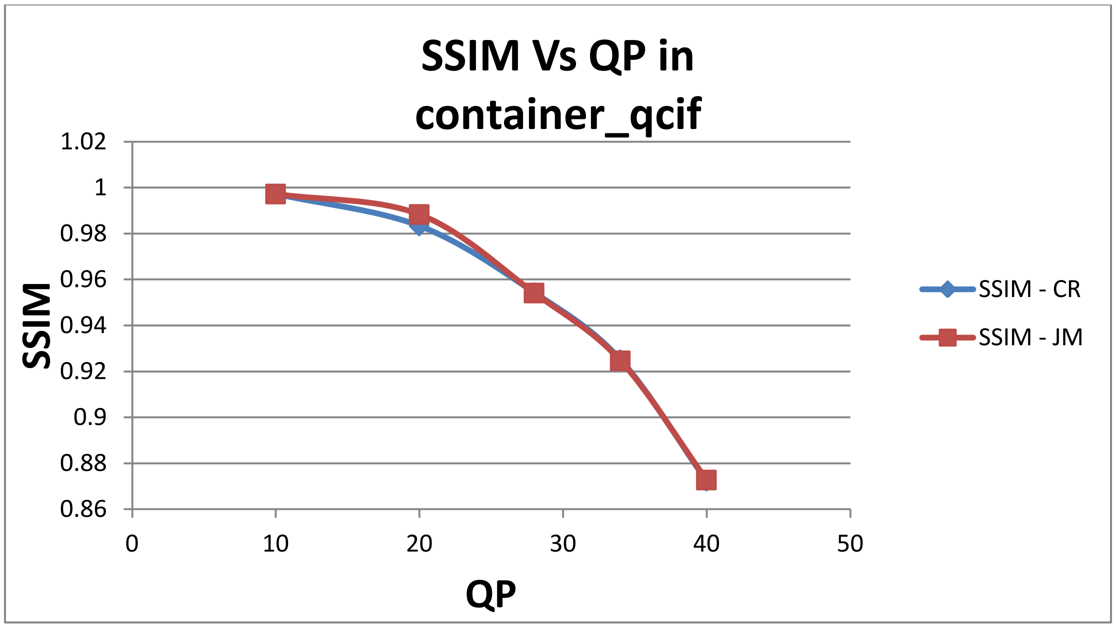


Figure 4.13 Plot of PSNR Vs QP for container_qcif video sequence

Figure 4.13 shows a plot of bit-rate Vs QP, comparing bit-rate values of the reference software JM and complexity reduced encoder against different values of QP for QCIF format, container_qcif video sequence. It can be observed from the curve that the bit-rates of complexity reduced encoder do not increase significantly when compared to JM reference software's bit-rates.

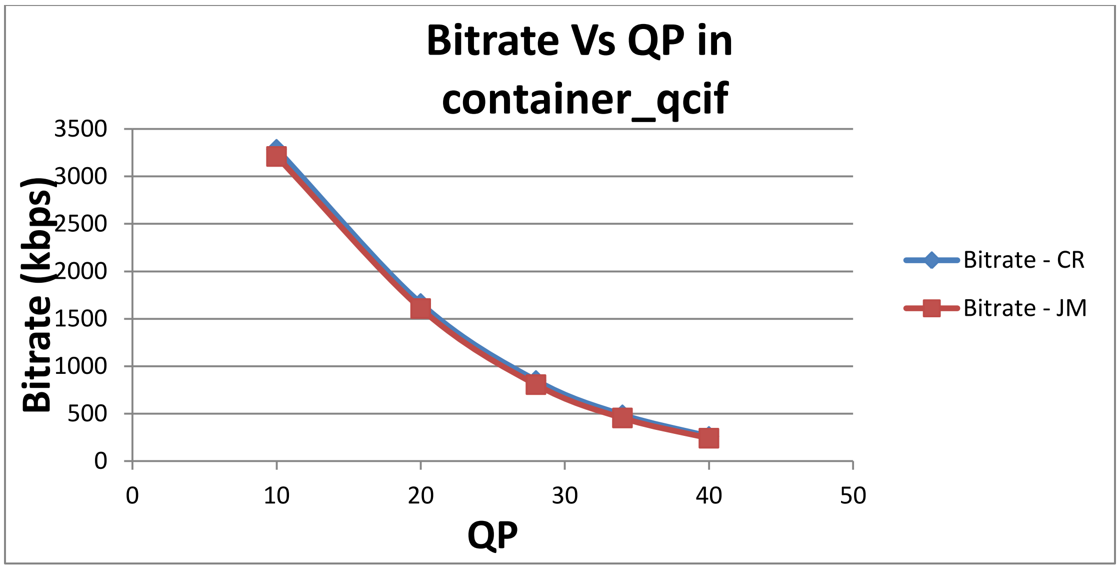


Figure 4.14 Plot of bit-rate Vs QP for container_qcif video sequence

## 4.6 Observation

From the above simulation results it can be deduced that, the encoder with complexity reduced algorithm takes significantly less encoding time when compared to the JM reference software and in the meantime, does not sacrifice the quality of the video nor does it increase the bit-rate significantly. Hence, this approach of reducing the number of mode combinations in spatial domain using simple directional masks can find its application in low complexity devices like mobile or any handheld device.

# CHAPTER 5

# CONCLUSIONS AND FUTURE WORK

## 5.1 Conclusions

From the simulation results in Chapter 4, it can be concluded that the proposed complexity reduction algorithm is faster than the JM reference software. This is because the JM reference software uses rate distortion optimization (RDO) which examines all possible combinations of coding modes, unlike the complexity reduction algorithm which makes use of simple directional masks and mode information of adjacent blocks.

From tables 4.1 through 4.6, it can be observed that there is an average of 24.93% reduction in encoding time in case of complexity reduced algorithm with negligible loss of PSNR and SSIM and a paltry increase in bit-rate and compression ratio.

Figures 4.5 through 4.9 shows the plots of comparison of encoding time, PSNR, bit-rate, SSIM and compression between the JM reference software and complexity reduced encoder at constant QP, 28. The simulation was performed on both CIF and QCIF sequences. Figures 4.10 through 4.14 show plots of comparison of encoding time, PSNR, bit-rate, SSIM and compression between the JM reference software and complexity reduced encoder for the QCIF format video sequence, container_qcif. This simulation was performed for various values of QP, 10, 20, 28, 34 and 40. These simulation results again concur that, significant encoding time can be reduced by using the proposed complexity reduction algorithm and at the same time, the fidelity of the input video stream is maintained.

## 5.2 Future work

The complexity reduction algorithm was implemented for CIF and QCIF format video sequences. The idea can be extended further to other video formats like 4SIF and HD. Also, the complexity reduction algorithm was integrated in to JM 18.0 reference software, it can be integrated to other open source H.264 softwares like X264 and performance analysis can be done. Since the aim is to reduce the overall complexity suitable for handheld devices with limited computing resources, algorithms which reduce the mode combinations in inter-prediction can also be integrated with this complexity reduced intra-prediction algorithm.

## REFERENCES

1. I. E.G. Richardson, “H.264 and MPEG-4 video compression: video coding for next-generation multimedia”, Wiley, 2003.
2. Interframe coding pictures, Axis Communications, http://www.axis.com/products/video/about_networkvideo/compression.htm
3. I. E. Richardson, “The H.264 advanced video compression standard”, 2nd Edition, Wiley, 2010.
4. Draft ITU-T Recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC), Mar. 2003.
5. ITU-T Recommendation H.120. Codecs for Videoconferencing using primary digital group transmission. March 1993.
6. ITU-T Recommendation H.261. Video codec for audiovisual services at px64 kbits. December 1990. March 1993 (revised).
7. ITU-T Recommendation H.262. Information technology – generic coding of moving pictures and associated audio information: video. July 1995
8. ISO/IEC 11172-5. Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbps. November 1998.
9. ITU-T Recommendation H.263. Video coding for low bit rate communication. February 1998.
10. Overview of International Video Coding Standards (preceding H.264/AVC), Gary J. Sullivan, ITU, 2005.
11. H.264 / MPEG-4 Part 10 White Paper, I. E G Richardson. www.vcodex.com.

12. M. Jafari and S. Kasaei, “Fast Intra- and Inter-Prediction Mode Decision in H.264 Advanced Video Coding”, International Journal of Computer Science and Network Security, VOL.8 No.5, pp. 1-6, May 2008.

13. F. Pan et al, “Fast intra mode decision algorithm for H.264/AVC video coding”, in Proc.IEEE Int. Conf. Image Process., pp. 781–784, Singapore, Oct. 2004.

14. F. Fu et al, “Fast intra prediction algorithm in H.264/AVC”, in Proc. 7th Int. Conf. Signal Process., pp. 1191–1194, Beijing, China, Sep. 2004.

15. Y. Zhang et al, “Fast 4x4 intra-prediction mode selection for H.264”, in Proc. Int. Conf. Multimedia Expo, pp. 1151–1154, Taipei, Taiwan, Jun. 2004.

16. J. Kim et al, “Complexity reduction algorithm for intra mode selection in H.264/AVC video coding” J. Blanc-Talon et al. (Eds.): pp. 454 – 465, ACIVS 2006, LNCS 4179, Springer-Verlag Berlin Heidelberg, 2006.

17. JM reference software, Fraunhofer Institute for Telecommunications Heinrich Hertz Institute. http://iphome.hhi.de/suehring/tml/

18. Recommendation ITU-R BT.601-7, Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios, BT Series, March 2011

19. Open source article on video formats, Wikipedia foundation. http://en.wikipedia.org/wiki/Common_Intermediate_Format

20. 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.

21. ITU-T Recommendation H.264. Advanced video coding for generic audiovisual services. November 2007.

22. ISO/IEC 11172-4. Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbps – Part 4. March 1998.

23. ITU Telecom. Standardization Sector of ITU, “Video coding for low bitrate communication,” Draft ITU-T Recommendation H.263 Version 2, Sept. 1997.

24. ITU-T "Video Coding for low bit rate communication," ITU-T Recommendation H.263; version 1, Nov 1995; version 2, Jan. 1998; version 3, Nov. 2000.

25. ISO/IEC 14496-2. Information technology - Coding of audio-visual objects – Part 2. December 2001.

26. Monash University, Multimedia webpage, http://www.ctie.monash.edu.au/EMERGE/multimedia/H261_263/H03.HTM

27. S. Kwon, A. Tamhankar, and K. Rao, “Overview of H. 264/MPEG-4 part 10, " Journal of Visual Communication and Image Representation, vol. 17, no. 2, pp.186-216, April 2006.

28. T. Purushotham, “Low complexity H.264 encoder using machine learning," M.S. Thesis, E.E Dept, UTA, 2010.

29. H. Yadav, “Optimization of the deblocking filter in H.264 codec for real time implementation,” M.S. Thesis, E.E Dept, UTA, 2006

30. G. Sullivan, P. Topiwala, and A. Luthra, The H.264/AVC advanced video coding standard: overview and introduction to the fidelity range extensions, in: SPIE Conference on Applications of Digital Image Processing XXVII, vol. 5558, pp. 53-74, 2004.

31. K.R. Rao and P. Yip, Discrete Cosine Transform, Academic Press, 1990.

32. Y. Huh, K. Panusopone and K.R. Rao, “Variable block size coding of images with hybrid quantization”, IEEE Trans. Circuits and Systems for Video Technology vol. 6, pp. 679–685, Dec. 1996.

33. J. Ribas-Corbera and D.L. Neuhoff, “Optimizing Block Size in Motion Compensation”, Journal of Electronic Imaging, vol. 7, pp.155-165, Jan. 1998

34. M. Wien, “Variable block size transforms for H.264/AVC”, IEEE Trans. For Circuits and Systems for Video Technology, vol. 13, pp. 604–613, July 2003.

35. M. Ravassi, M. Mattavelli and C. Clerc, “JVT/H.26L decoder complexity analysis”, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, doc. JVT-D153, Klagenfurt, Austria, 22–26 July, 2002 (available via anonymous ftp from ftp://ftp.imtc-files.org/jvt-experts/).

36. M.-T. Sun, T.-D. Wu and J.-N. Hwang, “Dynamic bit allocation in video combining for multipoint conferencing,” IEEE Trans. Circuits and Syst. II, vol. 45, no. 5, pp. 644-648, May 1998.

37. O. Werner, “Re-quantization for transcoding of MPEG-2 intra frames,” IEEE Trans. Image Processing, vol. 8, no. 2, pp. 179-191, Feb. 1999.

38. 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.

39. Repository for freely-redistributable test sequences, media.xiph.org.

40. D. Marpe, T. Wiegand and G. Sullivan, "The H.264/MPEG4 advanced video coding standard and its applications," IEEE, Communications Magazine, vol.44, no.8, pp.134-143, Aug. 2006.

41. ITU-T and ISO / IEC JTC 1, “Advanced Video Coding for Generic Audiovisual Services,” ITU-T Rec. H.264 & ISO / IEC 14496-10, Version 1, May 2003; Version 2, Jan. 2004; Version 3 (with High family of profiles), Sept 2004; Version 4, July 2005 [Online]. Available: http://www.itu.int/rec/T-REC-H.264.

42. G. J. Sullivan, “The H.264 / MPEG4-AVC video coding standard and its deployment status,” Proc. SPIE Conf. Visual Communications and Image Processing (VCIP), Beijing, China, July 2005.

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

44. A. Puri, X. Chen and A. Luthra, “Video coding using the H.264/MPEG-4 AVC compression standard”, Signal processing: image Communication, vol. 19, pp. 793-849, Oct. 2004.

45. 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, PY, April 2002.

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

# BIOGRAPHICAL STATEMENT

Santosh Kumar Muniyappa was born in Bangalore, India in 1985. He received the Bachelor’s degree in Electronics and Communication Engineering under Visvesvaraya Technological University in 2007. He worked for General Motors Technical Center India Pvt. Ltd. from July 2007 to July 2009 as a Validation Engineer in Bangalore. He decided to pursue his Master’s in University of Texas at Arlington in Fall 2009. He worked as a Graduate Research Assistant under Dr. Rao in the Multimedia Processing Lab from Summer 2010 to Fall 2010. Currently he is interning in Research In Motion at Irving, Texas as a Software Test Developer since Spring 2011. His interests lie in the field of video coding and embedded test automation. After graduation, he intends to join Intel Corp. (Austin).