IMPLEMENTATION AND ANALYSIS OF DIRECTIONAL DISCRETE COSINE TRANSFORM IN H.264 FOR BASELINE PROFILE

by

SHREYANKA SUBBARAYAPPA

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2012

ACKNOWLEDGEMENTS

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

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

I appreciate all members of Multimedia Processing Lab, Tejas Sathe, Priyadarshini and Darshan Alagud for their support during my research work. I would also like to thank my friends Adarsh Keshavamurthy, Babu Hemanth Kumar, Raksha, Kirthi, Spoorthi, Premalatha, Sadaf, Tanaya, Karthik, Pooja, and my Intel manager Sumeet Kaur who kept me going through the trying times of my Masters.

Finally, I am grateful to my family; my father Prof. H Subbarayappa, my mother Ms. Anasuya Subbarayappa, my sister Dr. Priyanka Subbarayappa, my brother-in-law Dr. Manju Jayram and my sweet little nephew Dishanth for their support, patience, and encouragement during my graduate journey.

April 16, 2012

ABSTRACT

IMPLEMENTATION AND ANALYSIS OF DIRECTIONAL DISCRETE COSINE TRANSFORM IN H.264 FOR BASELINE PROFILE

Shreyanka Subbarayappa, M.S

The University of Texas at Arlington, 2012

Supervising Professor: K.R.Rao

H.264/AVC [1] is a video coding standard that has a wide range of applications ranging from high-end professional camera and editing systems to low-end mobile applications. They strive to achieve maximum compression efficiency without compromising the quality of video. To this end many coding tools are defined in the coder. Transform coding is one among them. Transform Coding represents the signal/image (that is currently in time/spatial domain) in another domain (transform domain), where most of the energy of the signal/image is concentrated in a fewer number of coefficients. Thus the insignificant coefficients can be discarded after transform coding to achieve compression. In images/videos, the DCT-II [2][3] (which represents a signal/image as the weighted sum of cosine functions with different frequencies) is primarily used for transform coding [2].

Nearly all block-based transform schemes for image and video coding developed so far choose the 2-D discrete cosine transform (DCT) [2] of a square block shape. With almost no exception, this conventional DCT is implemented separately through two 1-D transforms [2], one along the vertical direction and another along the horizontal direction. Developing a new

block based DCT framework is one in which the first transform may choose to follow a direction other than the vertical or horizontal one (directional one) and the second transform chooses the horizontal one. The coefficients produced by all the directional transforms in the first step are arranged appropriately so that the second transform can be applied to the coefficients that are best aligned with one another. Compared with the conventional DCT, the resulting directional DCT [4] framework is able to provide a better coding performance for image blocks that contain directional edges—a popular scenario in many image signals. By choosing the best from all directional DCTs (including the conventional DCT as a special case) for each image block, the rate distortion coding performance can be improved remarkably.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

Table Page

LIST OF ACRONYMS

AVC – Advanced Video Coding

B slice – Bi-directional slice

CABAC – Context-Based Adaptive Binary Arithmetic Coding

CAVLC – Context Adaptive Variable Length Coding

CIF – Common Intermediate Format

DCT – Discrete Cosine Transform

DDCT – Directional Discrete Cosine Transform

DC coefficients – zero frequencies

DVD – Digital Versatile Disc

DFT – Discrete Fourier Transform

FRExt – Fedility Range Extensions

FMO – Flexible Macroblock Order

HD – High Definition

I frame – Intra frame

Int-DCT – Integer Discrete Cosine Transform

ISO – International Organization for standardization

ITU-T – International Telecommunication Union- Transmission standardization sector

IEC – International Electro-Technical Commission

IDCT – Inverse Discre Cosine Transform

IDDCT – Inverse Directional Discrete Cosine Transform

JPEG – Joint Photographic Experts group

JVT – Joint Video Team

JM - Joint Model

MPEG – Moving Picture Experts Group

MP3 – Moving Picture Experts Group Layer-3 Audio

MVC – Multiview Video Coding

MB – Macroblock

MSE – Mean Square Error

NTSC – National Television System Committee

PSNR – Peak Signal to Noise Ratio

PCM – Pulse Code Modulation

QCIF – Quarter Common Intermediate Format

QP – Quantization Parameter

RDO – Rate Distortion Optimization

SSIM – Structural Similarity Index Measure

SVC – Scalable Video Coding

SI – Switched intra coded

SP – Switched predictive coded

VLC – Variable Length Coding

YUV – Y-signal U-signal and V-signal

1-D – One Dimensional

2-D – Two Dimensional

CHAPTER 1

INTRODUCTION

1.1 Need for Compression

Uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia-based web applications have not only sustained the need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology. Image compression minimizes the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in the size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from Web pages.

For still image compression, the 'Joint Photographic Experts group' or JPEG standard has been established by ISO (International Standards Organization) and IEC (International Electro-Technical Commission). The performance of this standard generally degrades at low bit-rates mainly because of the underlying block-based discrete cosine transform (DCT) [2] scheme. More recently, the directional discrete cosine transform (DDCT) [4] has emerged as a cutting edge technology, within the field of image compression and video compression. Directional discrete cosine transform based coding provides substantial improvements in picture quality at higher compression ratios [5].

1.2 Objective of image compression

Images contain large amounts of information that requires much storage space, large transmission bandwidths and long transmission times. Advantageous to compress the image is to store only the essential information needed to reconstruct the image which will take less bandwidth to store on CDs, DVDs or Blu Ray Discs or transmit it through some medium. An image can be thought of as a matrix of pixel (or intensity) values. In order to compress the image, redundancies must be exploited, for example, areas where there is little or no change between pixel values. Therefore images having large areas of uniform color will have large redundancies, and conversely images that have frequent and large changes in color will be less redundant and harder to compress.

1.3 Types of data compression

There are two types of data compression

1. Lossless compression
2. Lossy compression

1.3.1 Lossless compression [9]

Lossless data compression is a class of data compression algorithms that allows the exact original data to be reconstructed from the compressed data. Lossless compression is used when it is important that the original and the decompressed data be identical, or when no assumption can be made on whether certain deviation is uncritical. Lossless compression is necessary for text and medical imaging, where every character is important. In other words each and every input symbol is very vital. Typical examples are executable programs and source code. Figure 1.1 shows the comparison between Lossless compression and Lossy compression.

1.3.2 Lossy Compression [9]

Lossy compression is a data encoding method that compresses data by discarding (losing) some of it. The procedure aims to minimize the amount of data that need to be held, handled, and/or transmitted. Lossy compression is most commonly used to compress multimedia data (audio, video and still images), especially in applications such as streaming media and internet telephony. It is possible to compress many types of digital data in a way that reduces the size of a computer file needed to store it, or the bandwidth needed to stream it, with no loss of the full information contained in the original file. A picture, for example, is coverted to a digital file by considering it to be an array of dots and specifying the color and brightness of each dot. If the picture contains an area of the same color, it can be compressed without loss by saying "200 red dots" instead of "red dot, red dot (197 more times)..., red dot."

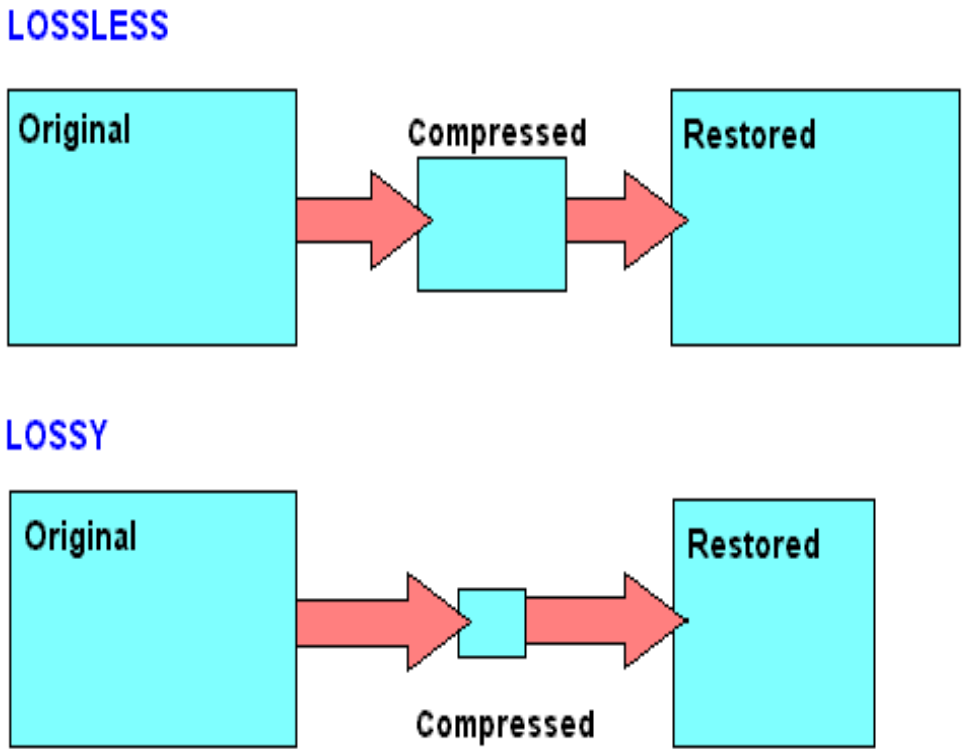


Figure 1.1 Comparison of lossless and lossy Image coding [9]

1.4 Transform coding [3]

Lossy compression can be thought of as an application of transform coding – in the case of multimedia data, perceptual coding is by transforming the raw data to a domain that more accurately reflects the information content. For example, rather than expressing a sound file as the amplitude levels over time, one may express it as the frequency spectrum over time, which corresponds more accurately to human audio perception. While data reduction (compression, be it lossy or lossless) is a main goal of transform coding, it also allows other goals: one may represent data more accurately for the original amount of space – for example, in principle, if one starts with an analog or high-resolution digital master, an MP3 file of a given size should provide a better representation than a raw uncompressed audio in WAV file of the same size. This is because uncompressed audio can only reduce file size by lowering bit rate or depth, whereas compressing audio can reduce size while maintaining bit rate and depth. This compression becomes a selective loss of the least significant data, rather than losing data across the board. Further, a transform coding may provide a better domain for manipulating or otherwise editing the data – for example, equalization of audio is most naturally expressed in the frequency domain (boost the bass, for instance) rather than in the raw time domain.

1.5 Significance of video

Pervasive, seamless, high-quality digital video has been the goal of companies, researchers and standards bodies over the last two decades. In some areas (for example broadcast television and consumer video storage), digital video has clearly captured the market while in others (videoconferencing, video mail, mobile video), market share is growing by the day. However, ther is no doubt that digital video is globally important industry which will continue to pervade business, networks and homes. The continuous evolution of the digital video industry is being driven by commercial and technical forces. The commercial drive comes

from the huge revenue potential of persuading consumers and business to replace analog technology and older digital technology with new, efficient, high-quality digital video products and to adopt new communication and entertainment products. The technical drive comes from continuing improvements in processing performance, the availability of higher-capacity storage and transmission mechanism and research and development of video and image processing technology. Figure 1.2 gives an example of a home media eco system and the importance of video.

Figure 1.2 Home media ecosystems [12]

1.6 Significance of video compression and its standardization

Getting digital video from its source (a camera or a stored clip) to its destination (a display) involves a chain of components or processes. Key to this chain are the processes of compression (encoding) and decompression (decoding), in which bandwidth-intensive ‘raw’ digital video is reduced to a manageable size for transmission or storage and then reconstructed for display. Getting the compression and decompression processes ‘right’ can give a significant technical and commercial edge to a product, by providing better image quality, greater reliability and/or more flexibility than competing solutions. There is therefore a keen

interest in the continuing development and improvement of video compression and decompression methods and systems.

Video compression also involves lossy compression. Video has both spatial and temporal redundancies. Getting rid of these redundancies and other lossy techniques facilitate very high compression. The volume of video data is usually very high. For example, in order to digitally represent 1 second of video without compression (using CCIR 601 [6] format more than 20 Mega bytes or 160 Mega bits is required. This amount of data indicates the importance of compression for video signals as bandwidth usage plays a crucial role. Table 1.1 gives an idea of the bit rates needed for raw video without any compression.

Table 1.1 Raw bit rates of uncompressed video [6]

Code	Width	Height	Description	Bit rate @ 30 fps, 8 bits/pixel, 4:2:0 format
QCIF	176	144	Quarter CIF	9 Mbps
QVGA	320	240	Quarter Video Graphics Array	27 Mbps
CIF	352	288	Common Intermediate Format	36 Mbps
HVGA	640	240	Half Video Graphics Array	55 Mbps
VGA	640	480	Video Graphics Array	110 Mbps
SD	720	486	Standard Definition	125 Mbps
SVGA	800	600	Super Video Graphics Array	172 Mbps
XGA	1024	768	Extended Graphics Array	283 Mbps
XGA+	1152	768	Extended Graphics Array plus	318 Mbps
	1152	864		358 Mbps
SXGA	1280	1024	Super Extended Graphics Array	471 Mbps
SXGA+	1400	1050	Super Extended Graphics Array plus	529 Mbps
UXGA	1600	1200	Ultra Extended Graphics Array	691 Mbps
HD	1920	1080	High Defintion	746 Mbps
QXGA	2048	1536	Quad Extended Graphics Array	1.1 Gbps

Multimedia applications are targeted for a wide range of applications such as video conferencing, video on demand, mobile video broadcast and even medical applications. Given such wide range of applications, standardization of video compression techniques is essential. Standardization ensures interoperability of implementation from different vendors thereby enabling end-users to access video from different services both software and hardware. There are numerous video compression standards [8] both open-source and proprietary depending on the application and end-usage.

1.7 Summary

Hence the need for compression in media communication was expressed in detail. Chapter 2 is a detailed study of the H.264 codec and its implementation.

CHAPTER 2

H.264 VIDEO CODING STANDARD

2.1 Introduction

H.264/MPEG4-Part 10 advanced video coding (AVC) introduced in 2003 became one of the latest and most efficient video coding standards [11]. The H.264 standard was developed by the joint video team (JVT), consisting of VCEG (video coding experts group) of ITU-T (International Telecommunication Union – Telecommunication standardization sector), and MPEG (moving picture experts group) of ISO/IEC [13].

H.264 can support various interactive (video telephony) and non-interactive applications (broadcast, streaming, storage, video on demand) as it facilitates a network friendly video representation. It leverages on the previous coding standards such as MPEG-1 [40], MPEG-2 [41], MPEG-4 part 2 [42], H.261 [43], H.262 [44] and H.263 [45] [47]and adds many other coding tools and techniques which give it superior quality and compression efficiency.

The standardization of the first version of H.264/AVC was completed in May 2003. The JVT then developed extensions to the original standard that are known as the fidelity range extensions (FRExt) [34]. These extensions enable higher quality video coding by supporting increased sample bit depth precision and higher-resolution color information, including sampling structures known as YUV 4:2:2 and YUV 4:4:4 as Figure 2.6. Several other features are also included in the fidelity range extensions, such as adaptive switching between 4×4 and 8×8 integer transforms, encoder-specified perceptual-based quantization weighting matrices, efficient inter-picture lossless coding, and support of additional color spaces. The design work on the fidelity range extensions was completed in July 2004, and the drafting work on them was completed in September 2004.

Scalable video coding (SVC) [17] allows the construction of bitstreams that contain sub-bitstreams that conform to H.264/AVC. For temporal bitstream scalability, i.e., the presence of a sub-bitstream with a smaller temporal sampling rate than the bitstream, complete access units are removed from the bitstream when deriving the sub-bitstream. In this case, high-level syntax and inter prediction reference pictures in the bitstream are constructed accordingly. For spatial and quality bitstream scalabilities, i.e. the presence of a sub-bitstream with lower spatial resolution or quality than the bitstream, network abstraction layer (NAL) units are removed from the bitstream when deriving the sub-bitstream. In this case, inter-layer prediction, i.e., the prediction of the higher spatial resolution or quality signal by data of the lower spatial resolution or quality signal, is typically used for efficient coding. The scalable video coding extension was completed in November 2007 [17].

The next major feature added to the standard was Multiview Video Coding (MVC). MVC enables the construction of bit streams that represent more than one view of a video scene. An important example of this functionality is stereoscopic 3D video coding. Two profiles were developed in the MVC work: one supporting an arbitrary number of views and designed specifically for two-view stereoscopic video. The Multiview Video Coding extensions were completed in November 2009 [13].

Like any other previous motion-based codecs, it uses the following basic principles of video compression [8]:

- Transform for reduction of spatial correlation
- Quantization for control of bit rate
- Motion compensated prediction for reduction of temporal correlation
- Entropy coding for reduction in statistical correlation.

The improved coding efficiency of H.264 can be attributed to the additional coding tools and the new features. Listed below are some of the new and improved techniques used in H.264 for the

first time [15]:

- Adaptive intra-picture prediction
- Small block size transform with integer precision
- Multiple reference pictures and generalized B-frames
- Quarter pixel precision for motion compensation
- Variable block size
- Content adaptive in-loop deblocking filter and
- Improved entropy coding by introduction of CABAC (context adaptive binary arithmetic coding) and CAVLC (context adaptive variable length coding).

The increase in the coding efficiency and increase in the compression ratio result in a greater complexity of the encoder and the decoder algorithms of H.264, as compared to previous coding standards. In order to develop error resilience for transmission of information over the network, H.264 supports the following techniques [19]:

- Flexible macroblock ordering (FMO)
- Switched slice
- Arbitrary slice order (ASO)
- Redundant slice (RS)
- Data partitioning (DP)
- Parameter setting

2.2 Profiles and levels of H.264

The H.264/AVC standard is composed of a wide range of coding tools. Also, the standard addresses a large range of bit rates, resolutions, qualities, applications and services. Not all the tools and all the bit rates are required for any given application at a given point of time. All the various tools of H.264 are grouped in profiles.

2.2.1 Profiles in H.264

Profiles are defined as a subset of coding tools. They help to maximize the interoperability while limiting the complexity [15]. Also, the various levels define the various parameters like size of decoded pictures, bit rate, etc.

The profiles defined for H.264 can be listed as follows [15]:

- Constrained baseline profile
- Baseline profile
- Main profile
- Extended profile
- High profile
- Progressive high profile
- Constrained high profile
- High 10 profile
- High 4:2:2 profile
- High 4:4:4 predictive profile
- High stereo profile
- High 10 intra profile
- High 4:2:2 intra profile
- High 4:4:4 intra profile
- CAVLC 4:4:4 intra profile
- Scalable baseline profile
- Scalable high profile
- Scalable high intra profile 18
- Scalable constrained high profile
- Stereo high profile

- Multiview high profile

Figure 2.1 illustrates the coding tools for the various profiles of H.264.

The standard defines 21 sets of capabilities, which are referred to as *profiles*, targeting specific classes of applications.

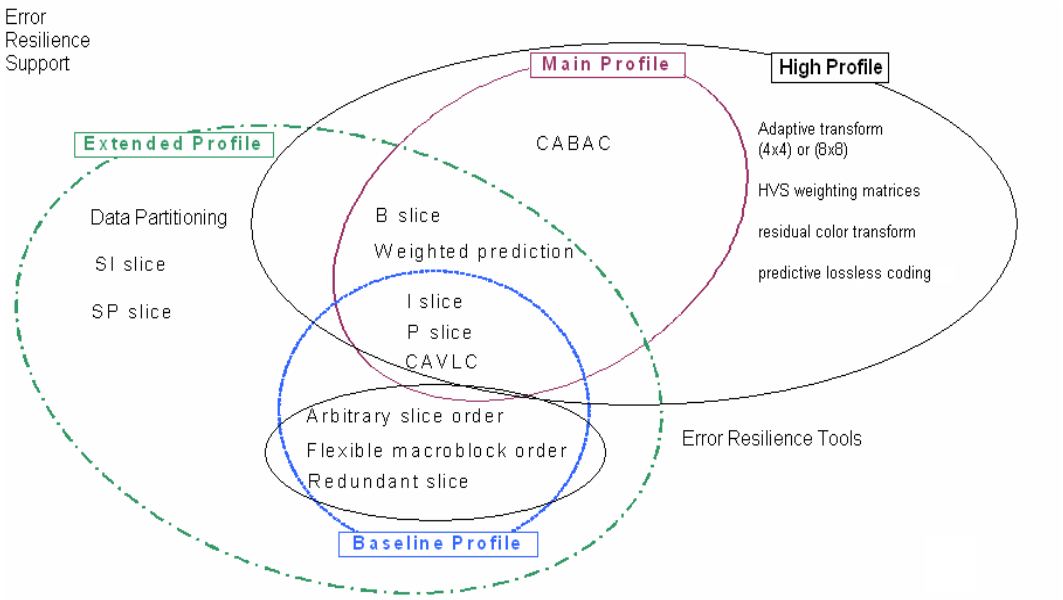


Figure 2.1 Different profiles in H.264 with distribution of various coding tools among the profiles [15]

2.2.1.1 Baseline Profile

The list of tools included in the baseline profile are I (intra coded) and P (predictive coded) slice coding, enhanced error resilience tools of flexible macroblock ordering, arbitrary slices and redundant slices. It also supports CAVLC (context-based adaptive variable length coding). The baseline profile is intended to be used in low delay applications, applications demanding low processing power, and in high packet loss environments. This profile has the least coding efficiency among all the three profiles.

2.2.1.2 Main Profile

The coding tools included in the main profile are I, P, and B (bi directionally predictive coded) slices, interlace coding, CAVLC and CABAC (context-based adaptive binary arithmetic coding). The tools not supported by main profile are error resilience tools, data partitioning and SI (switched intra coded) and SP (switched predictive coded) slices. This profile is aimed to achieve highest possible coding efficiency.

2.2.1.3 Extended Profile

This profile has all the tools included in the baseline profile. As illustrated in the Figure 2.1 this profile also includes B, SP and SI slices, data partitioning, interlaced frame and field coding, picture adaptive frame/field coding and macroblock adaptive frame/field coding. This profile provides better coding efficiency than baseline profile. The additional tools result in increased complexity.

2.2.1.4 High Profiles defined in the FRExts amendment

In September 2004 the first amendment of H.264/MPEG-4 AVC video coding standard was released [15]. A new set of coding tools were introduced as a part of this amendment. These are termed as “Fidelity Range Extensions” (FRExts). The aim of releasing FRExts is to be able to achieve significant improvement in coding efficiency for higher fidelity material. It also has lossless representation capability: I PCM raw sample value macroblocks and entropy coded transform by-pass lossless macroblocks (FRExts only). The application areas for the FRExts tools are professional film production, video production and high-definition TV/DVD.

The FRExts amendment defines four new profiles (refer Figure 2.2) [17]:

- High (HP)
- High 10 (Hi10P)

- High 4:2:2 (Hi422P)
- High 4:4:4 (Hi444P)

  The other profiles constrained to intra use in H.264 are

- High 10 intra profile
- High 4:2:2 intra profile
- High 4:4:4 intra profile

Figure 2.2 gives us the specification of High profile in H.264.

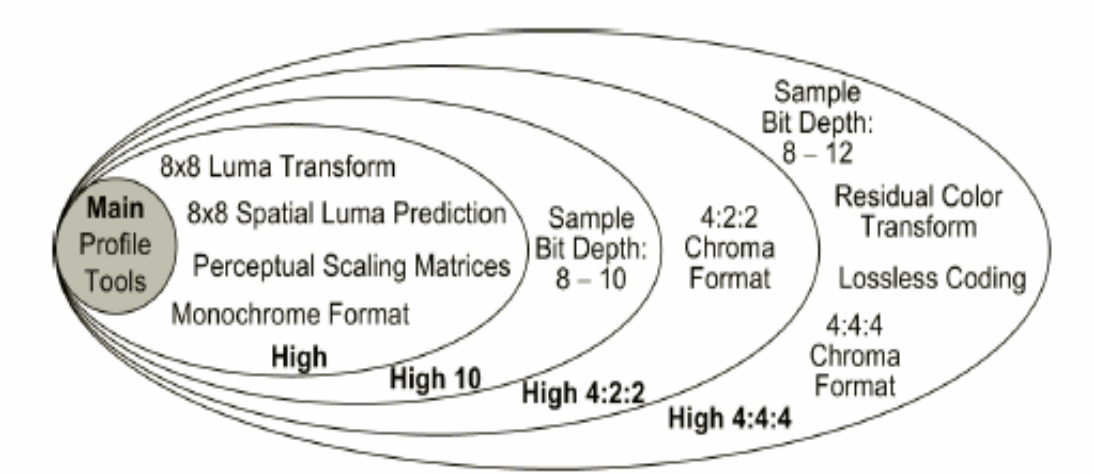


Figure 2.2 Tools introduced in FRExts and their classification under the new high profiles [28]

All four of these profiles build further upon the design of the prior main profile. Table 2.1 provides a comparison of the high profiles introduced in FRExts with a list of different coding tools and which of them are applied to which profile. All of the high profiles include the following three enhancements of coding efficiency performance [28]:

- Adaptive macroblock-level switching between 8x8 and 4x4 transform block sizes.

  The main aim behind introducing 8x8 transform in FRExts is that high fidelity video demands preservation of fine details and textures. To achieve this, larger basis functions are required. However, smaller size transform like 4x4 reduces ringing

artifacts and reduces computational complexity. The encoder adaptively chooses between 4x4 and 8x8 transforms block sizes.

The transform selection process is limited by the following conditions

- If an inter-coded MB has sub-partition smaller than 8x8 (i.e. 4x8, 8x4, 4x4), then 4x4 transform is used.
- If an intra-coded MB is predicted using 8x8 luma spatial prediction, only 8x8 transform is used.

- Encoder-specified perceptual-based quantization scaling matrices

  The encoder can specify a matrix for scaling factor according to the specific frequency associated with the transform coefficient for use in inverse quantization scaling by the decoder. This allows optimization of the subjective quality according to the sensitivity of the human visual system, less sensitive to the coded error in high frequency transform coefficients [37].

Table 2.1 Comparison of the high profiles and corresponding coding tools introduced in the FRExts [34]

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

2.3 H.264 Encoder

Figure 2.3 illustrates the schematic of the H.264 encoder. H.264 encoder works on macroblocks and motion-compensation like most other previous generation codecs. Video is formed by a series of picture frames. Each picture frame is an image which is split down into blocks. The block sizes can vary in H.264. The encoder may perform intra-coding or inter-coding for the macroblocks of a given frame. Intra coded frames are encoded and decoded independently. They do not need any reference frames. Hence they provide access points to the coded sequence where decoding can start. H.264 uses nine spatial prediction modes in intra-coding to reduce spatial redundancy in the source signal of the picture as shown in Figure 2.6. These prediction modes are explained in Figure 2.4. Inter-coding uses inter-prediction of a given block from some previously decoded pictures. The aim to use inter-coding is to reduce the temporal redundancy by making use of motion vectors. Motion vectors give the direction of motion of a particular block from the current frame to the next frame. The prediction residuals are obtained which then undergo transformation to remove spatial correlation in the block. The transformed coefficients, thus obtained, undergo quantization. The motion vectors (obtained from inter-prediction) or intra-prediction modes are combined with the quantized transform coefficient information. They are then encoded using entropy code such as context-based adaptive variable length coding (CAVLC) or context-based adaptive binary arithmetic coding (CABAC) [34].

There is a local decoder within the H.264 encoder. This local decoder performs the operations of inverse quantization and inverse transform to obtain the residual signal in the spatial domain. The prediction signal is added to the residual signal to reconstruct the input frame. This input frame is fed in the deblocking filter to remove blocking artifacts at the block boundaries. The output of the deblocking filter is then fed to inter/intra prediction blocks to generate prediction signals.

The various coding tools used in the H.264 encoder are explained in the sections 2.3.1 through 2.3.6.

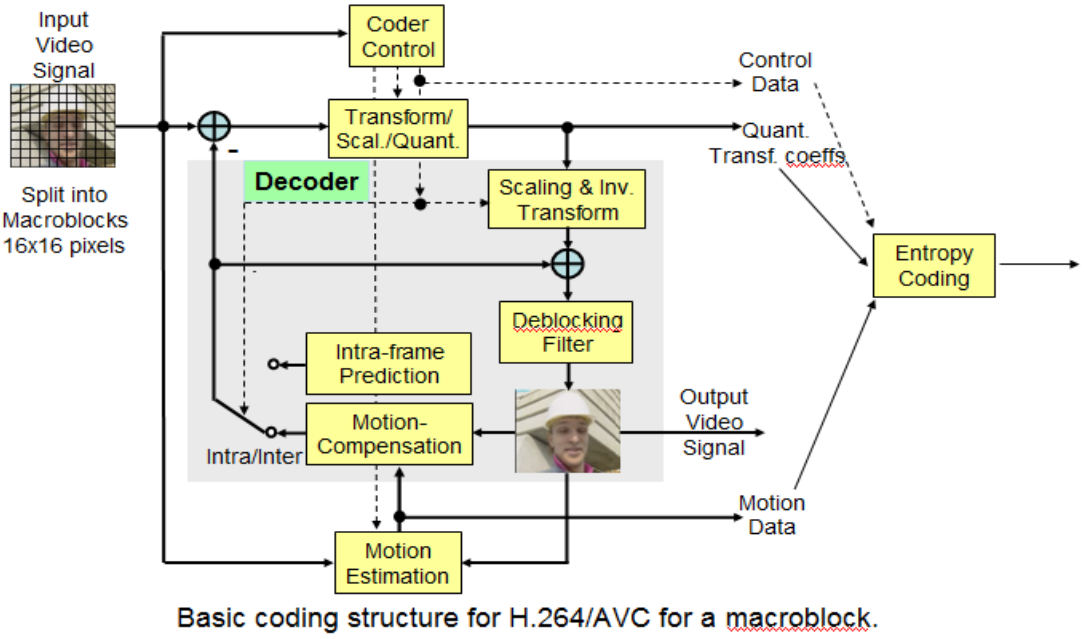


Figure 2.3 H.264 Encoder block diagram [1]

2.3.1 Intra-prediction

Intra-prediction uses the macroblocks from the same image for prediction. Two types of prediction schemes are used for the luminance component. These two schemes can be referred as INTRA_4x4 and INTRA_16x16 [38]. In INTRA_4x4, a macroblock of size 16x16 pixels are divided into 16 4x4 sub blocks. Intra prediction scheme is applied individually to these 4x4 sub blocks. There are nine different prediction modes supported as shown in Figure 2.4. In FRExts profiles alone, there is also 8x8 luma spatial prediction (similar to 4x4 spatial prediction) and with low-pass filtering of the prediction to improve prediction performance.

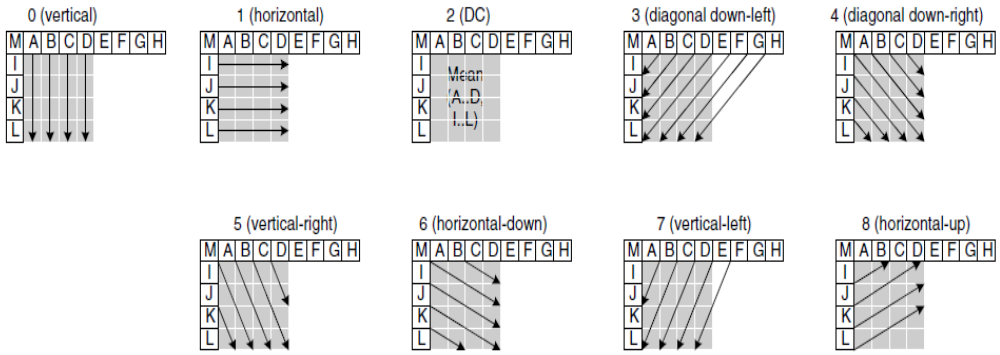


Figure 2.4 4x4 Luma prediction (intra-prediction) modes in H.264 [1]

In mode 0, the samples of the macroblock are predicted from the neighboring samples on the top. In mode 1, the samples of the macroblock are predicted from the neighboring samples from the left. In mode 2, the mean of all the neighboring samples is used for prediction. Mode 3 is in diagonally down-left direction. Mode 4 is in diagonal down-right direction. Mode 5 is in vertical-right direction. Mode 6 is in horizontal-down direction. Mode 7 is in vertical-left direction. Mode 8 is in horizontal up direction. The predicted samples are calculated from a weighted average of the previously predicted samples A to M.

For prediction of 16x16 intra prediction of luminance components, four modes are used as shown in Figure 2.7. The three modes of mode 0 (vertical), mode 1 (horizontal) and mode 2 (DC) are similar to the prediction modes for 4x4 block. In the fourth mode, the linear plane function is fitted in the neighboring samples.

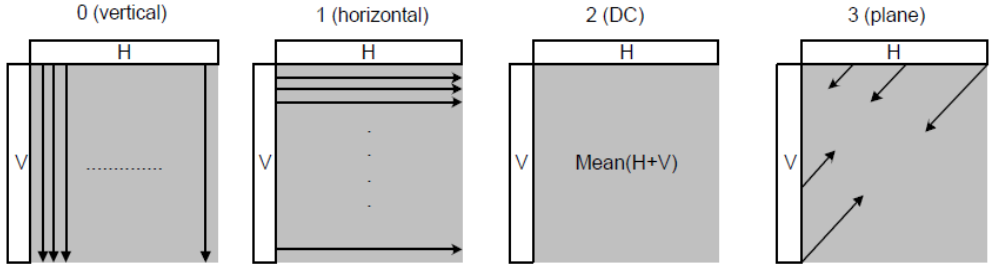


Figure 2.5 16X16 Luma prediction (intra-prediction) modes in H.264 [1]

The chroma macroblock is predicted from neighboring chroma samples. The four prediction modes used for the chroma blocks are similar to 16x16 luma prediction modes. The number in which the prediction modes are ordered is different for chroma macroblock: mode 0 is DC, mode 1 is horizontal, mode 2 is vertical and mode 3 is plane. The block sizes for the chroma prediction depend on the sampling format. For 4:2:0 format, 8x8 size of chroma block is selected. For 4:2:2 format, 8x16 size of chroma block is selected. For 4:4:4 format, 16x16 size of chroma block is selected [1]. Figure 2.6 illustrates chroma sub sampling.

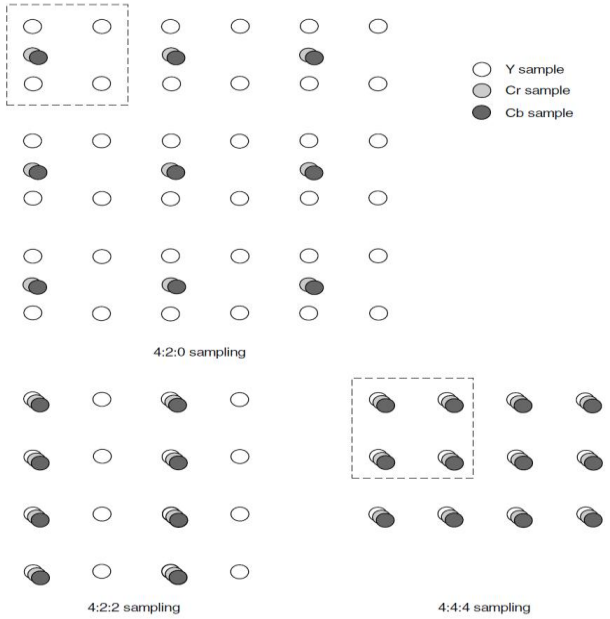


Figure 2.6 Chroma sub sampling [1]

2.3.2 Inter-prediction

Inter-prediction is used to capitalize on the temporal redundancy in a video sequence. The temporal correlation is reduced by inter prediction through the use of motion estimation and compensation algorithms [1]. An image is divided into macroblocks; each 16x16 macroblock is further partitioned into 16x16, 16x8, 8x16, 8x8 sized blocks. A 8x8 sub-macroblock can be further partitioned into 8x4, 4x8, 4x4 sized blocks. Figure 2.7 illustrates the partitioning of a macroblock and a sub-macroblock [1]. The input video characteristics govern the block size. A

smaller block size ensures less residual data; however smaller block sizes also mean more motion vectors and hence more number of bits required to encode these motion vectors [1] .

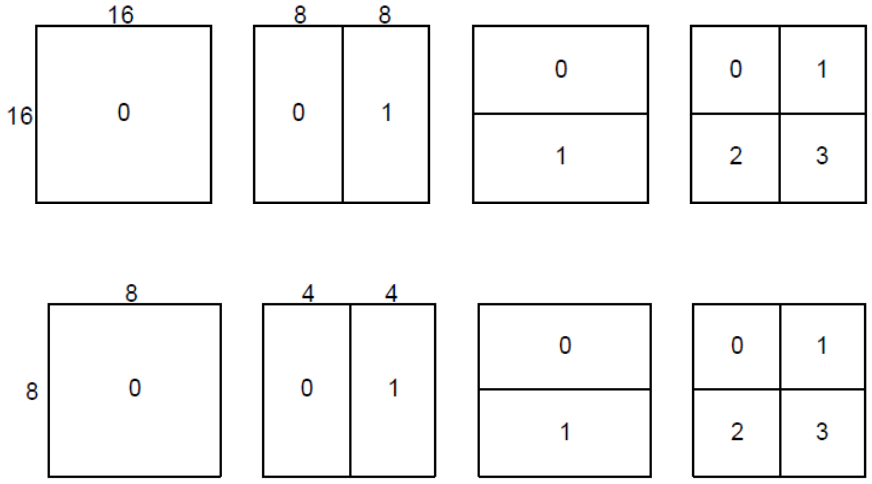


Figure 2.7 Macroblock portioning in H.264 for inter prediction [1] row1 (L-R) 16x16, 8x16, 16x8, 8x8 blocks and row2 (L-R) 8x8, 4x8, 8x4, 4x4 blocks

Each partition or sub-macroblock partition in an inter-coded macroblock is predicted from an area of the same size in a reference picture. The offset between the two areas (the motion vector) has quarter-sample resolution for the luma component and one-eighth-sample resolution for the chroma components. The luma and chroma samples at sub-sample positions do not exist in the reference picture and so it is necessary to create them using interpolation from nearby coded samples. Figures 2.8 and 2.11 illustrate half and quarter pixel interpolation used in luma pixel interpolation respectively. Six-tap filtering is used for derivation of half-pel luma sample predictions, for sharper sub pixel motion-compensation. Quarter-pixel motion is derived by linear interpolation of the half pel values, to save processing power.

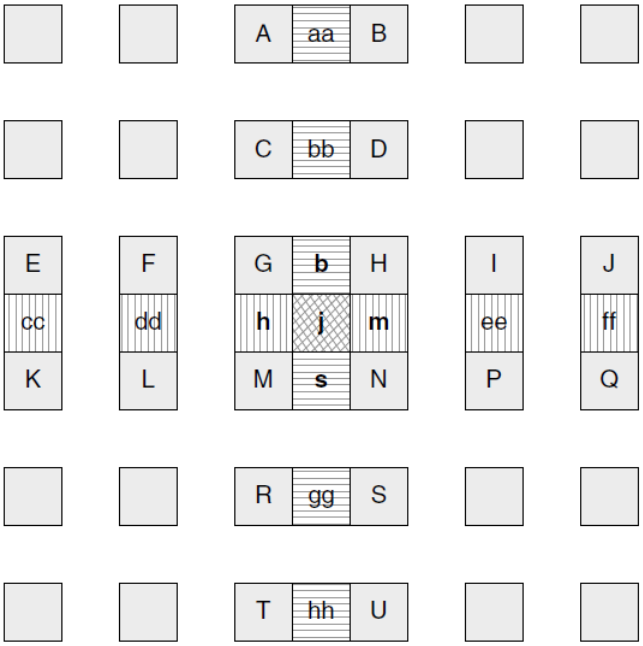


Figure 2.8 Interpolation of luma half-pel positions [1]

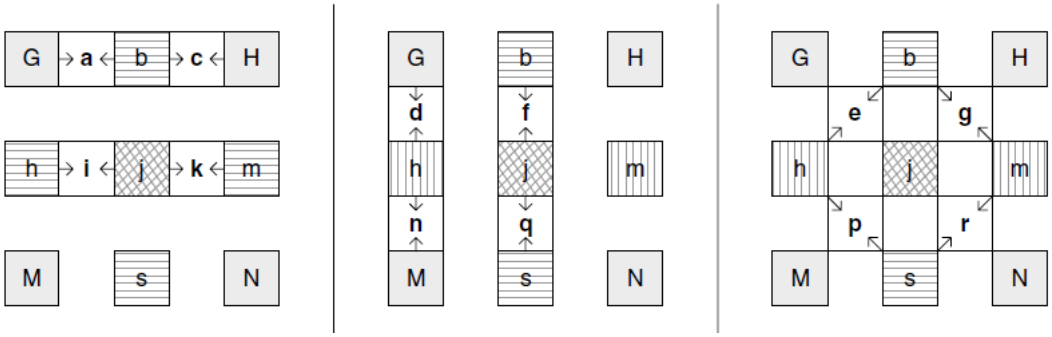


Figure 2.9 Interpolation of luma quarter-pel positions [1]

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

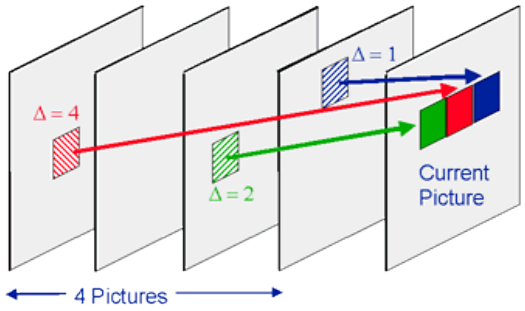
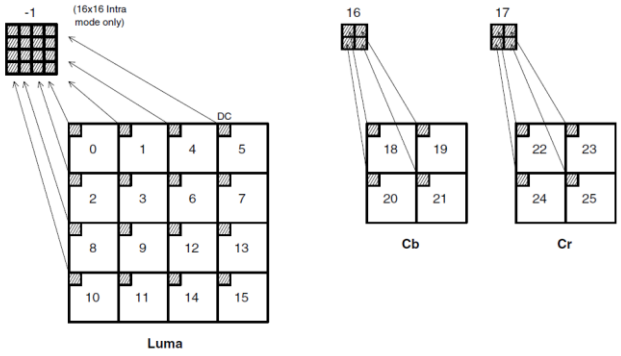


Figure 2.10 Motion compensated prediction with multiple reference frames [1]

2.3.3 Transform coding

There is high spatial redundancy among the prediction error signals. H.264 implements a block-based transform to reduce this spatial redundancy [1]. The former standards of MPEG-1 and MPEG-2 employed a two dimensional discrete cosine transform (DCT) for the purpose of transform coding of the size 8x8 [1]. H.264 uses integer transforms instead of the DCT. The size of these transforms is 4x4 [1]. The advantages of using a smaller block size in H.264 are stated as follows:

- The reduction in the transform size enables the encoder to better adapt the prediction error coding to the boundaries of the moving objects and to match the transform block size with the smallest block size of motion compensation.
- The smaller block size of the transform leads to a significant reduction in the ringing artifacts.
- The 4x4 integer transform has the benefit for removing the need for multiplications.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 2.11 (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)

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

(c)

$$\mathbf{W}_{QD} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \mathbf{W}_D \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

(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 2.11 H.264 Transformation
(a) DC coefficients of 16 4x4 luma blocks, 4 4x4 Cb and Cr blocks [1]
(b) Matrix H_1 (e) is applied to 4x4 block of luma/chroma coefficients X (a) [34]
(c) Matrix H_2 (e) (4x4 Hadamard transform) applied to luma DC coefficients W_D [34]
(d) Matrix H_3 (e) (2x2 Hadamard transform) applied to chroma DC coefficients W_D [34]
(e) Matrices H_1, H_2 and H_3 of the three transforms used in H.264 [34]

As shown in Figure 2.11 (b) the first transform (matrix H_1 in is applied to all samples of all prediction error blocks of the luminance component (Y) and for all blocks of chrominance components (Cb and Cr). For blocks with mostly flat pixel values, there is significant correlation among transform DC coefficients of neighboring blocks. Hence, the standard specifies the 4x4 Hadamard transform (matrix H_2 in figure 2.11 (c)) for luma DC coefficients (Figure 2.11 (c)) for 16x16 intra-mode only, and 2x2 Hadamard transform as shown in figure 2.11 (d) (matrix H_3 in figure 2.11 (e)) for chroma DC coefficients.

2.3.4 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 as shown in Figure 2.12. 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 2.12 illustrates a macroblock with sixteen 4x4 sub blocks along with their boundaries.

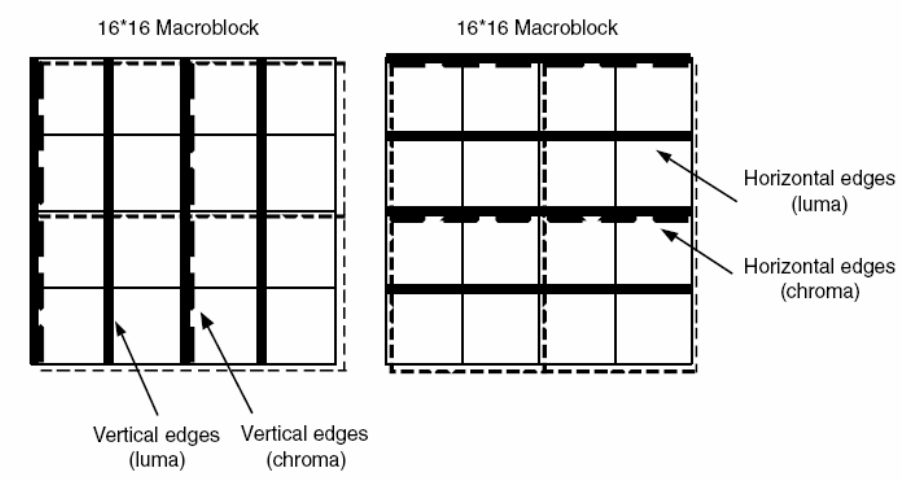


Figure 2.12 Boundaries in a macroblock to be filtered (luma boundaries shown with solid lines and chroma boundaries shown with dotted lines) [1]

As shown in the figure 2.12, 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:

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

2.3.5 Entropy coding

H.264 uses variable length coding to match a symbol to a code based on the context characteristics. All the syntax elements except for the residual data are encoded by the Exp-Golomb codes[1]. The residual data is encoded using CAVLC. The main and the high profiles of H.264 use CABAC.

- Context-based adaptive variable length coding (CAVLC):

  After undergoing transform and quantization the probability that the level of coefficients is zero or +1 is very high [1]. CAVLC handles these values differently. It codes the number of zeroes and +1. For other values, their values are coded.
- Context-based adaptive binary arithmetic coding (CABAC):

  This technique utilizes the arithmetic encoding to achieve good compression. The schematic for CABAC is shown in Figure 2.13.

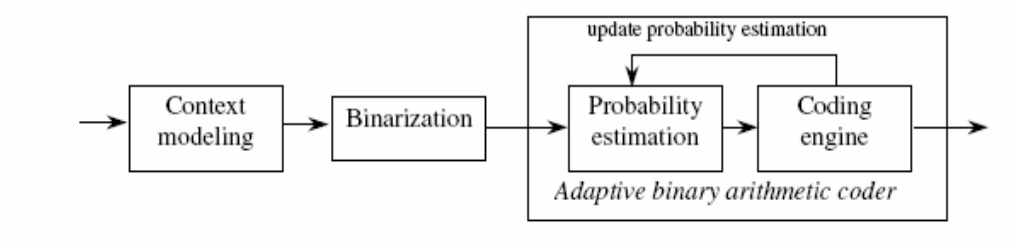


Figure 2.13 Schematic block diagram of CABAC [1]

CABAC consists of three steps:

- Step 1: Binarization: A non-binary value is uniquely mapped to a binary sequence
- Step 2: Context modeling: A context model is a probability model for one or more elements of binarized symbol. The probability model is selected such that corresponding choice may depend on previously encoded syntax elements.
- Step 3: Binary arithmetic coding: An arithmetic encoder encodes each element according to the selected probability model.

2.3.6 B-slices and adaptive weighted prediction

Bi-directional prediction which uses both past and future frames for reference can be very useful in improving the temporal prediction. Bi-directional prediction in H.264 uses multiple reference frames. Figure 2.14 shows bidirectional prediction from multiple reference frames. The video coding standards, before H.264, with B pictures use the bidirectional mode, with limitation that it allows the combination of a previous and subsequent prediction signals. In the previous standards, one prediction signal is derived from subsequent inter-picture, another from a previous picture, the other from a linear averaged signal of two motion compensated prediction signals.

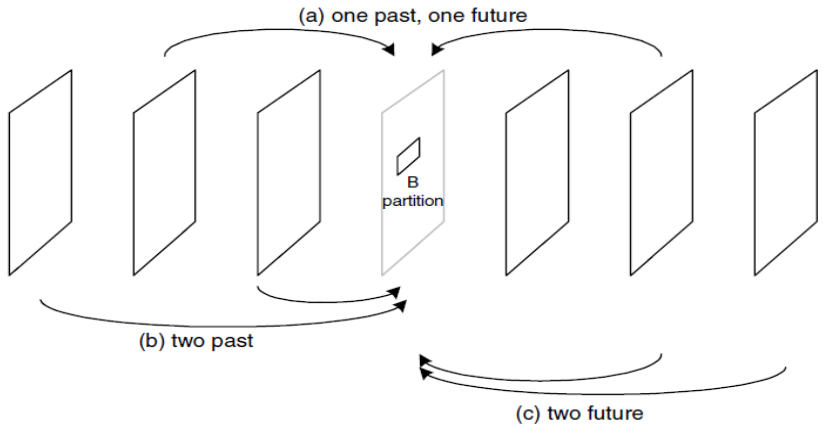


Figure 2.14 Partition prediction examples in a B macroblock type: (a) past/future, (b) past, (c) future [1]

H.264 supports forward/backward prediction pair and also supports forward/forward and backward/backward prediction pair [1]. Figure 2.14 (a) and Figure 2.14 (b) describe the scenario where bidirectional prediction and multiple reference frames respectively are applied and a macroblock is thereby predicted as a linear combination of multiple reference signals using weights as described in equation 2.1. Two forward references for prediction are beneficial for motion compensated prediction of a region just before scene change. Two backward reference frames are beneficial for frames just after scene change. H.264 also allows bi-directionally predictive-coded slice which may also be used as reference for inter-coding of other pictures. Except H.264, all the existing standards consider equal weights for reference pictures. Equal weights of reference signals are averaged and the prediction signal is obtained. H.264 also uses weighted prediction [1]. It can be used for a macroblock of P slice or B slice. Different weights can be assigned to the two different reference signals and the prediction signal is calculated as follows:

$$p = w_1 * r_1 + w_2 * r_2 \qquad (2.1)$$

In (2.1), p is the prediction signal, r_1 and r_2 are the reference signals and w_1 and w_2 are the prediction weights.

2.4 H.264 Decoder

The H.264 [34] decoder works similar in operation to the local decoder of H.264 encoder. An encoded bit stream is the input to the decoder. Entropy decoding (CABAC or CAVLC) takes place on the bit stream to obtain the transform coefficients. These coefficients are then inverse scanned and inverse quantized. This gives residual block data in the transform domain. Inverse transform is performed to obtain the data in the pixel domain. The resulting output is 4x4 blocks of residual signal. Depending on inter predicted or intra-predicted, an appropriate prediction signal is added to the residual signal. For an inter-coded block, a prediction block is constructed depending on the motion vectors, reference frames and previously decoded pictures. This prediction block is added to the residual block to reconstruct the video frames. These reconstructed frames then undergo deblocking before they are stored for future use for prediction or being displayed. Figure 2.15 illustrates the decoder.

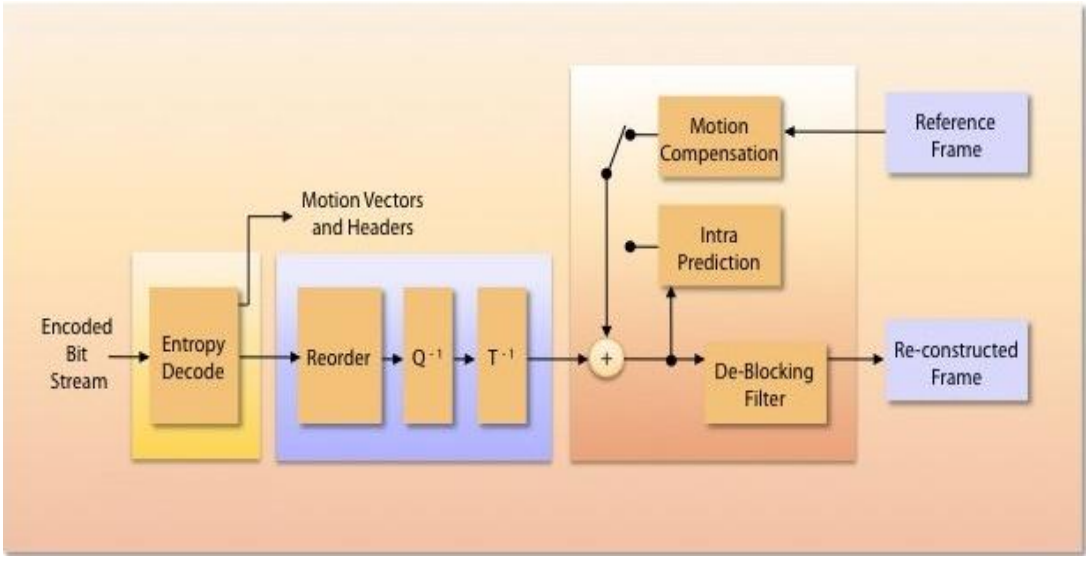


Figure 2.15 H.264 Decoder block diagram [1]

2.5 Summary

This chapter outlines the coding tools of H.264 codec. Having had a good understanding of H.264, next chapter describes the theoretical concept of directional discrete cosine transforms.

CHAPTER 3

DIRECTIONAL DISCRETE COSINE TRANSFORMS [25]

3.1 Introduction

Nearly all block-based transform techniques developed so far for image and video coding applications choose the 2-D discrete cosine transform (DCT) [2] of a square block shape. With no exception, this conventional DCT is always implemented separately through two 1-D transforms, along the vertical and horizontal directions, respectively. This discrete cosine transform frameworkis replaced by directional DCT framework in which the first transform may choose to follow a direction other than the vertical or horizontal one, while the second transform is arranged to be a horizontal one. Compared to the conventional DCT, directional DCT [25] framework has been demonstrated to provide a better coding performance for image blocks that contain directional edges – a popular scenario in many image and video signals.

It is known that each digital image or video frame contains a lot of directional edges/details, and such edge/detail orientation varies gently from one image block to another. When the conventional 2-D DCT is applied to an image block, the first DCT will be performed along the vertical or horizontal direction. If the edge/detail orientation within this image block does not follow the vertical or horizontal direction, many non-zero AC components are produced after the transform, thus making the coding rather inefficient. By recognizing this defect, directional DCT framework was developed in which the first transform can choose to follow the dominating direction (which can be different from the vertical or horizontal direction) within each image block, while the second transform can be implemented according to the arrangement of the coefficients after the first transform.

The directional framework has been demonstrated to provide a remarkable coding gain

for all image blocks that contain directional edges other than the vertical/horizontal ones [4]. However, there are a number of important issues that need to be solved before it can be applied to practical image and video coding applications. One of these issues is the theoretical analysis so as to understand why coding gain is obtained and how big it can be. Further on in this chapter, the concept is explained in detail.

3.1.1 Conventional DCT [2]

A discrete cosine transforms (DCT expresses a sequence of finite data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from lossy compression of audio (e.g. MP3) [40] [41] and images (e.g. JPEG) [36] (where small high-frequency components can be discarded), to spectral methods for the numerical solution of differential equations. The use of cosine rather than sine functions is critical in these applications: for compression it turns out that the cosine functions are much more efficient (as described below, fewer functions are needed to approximate a typical signal), whereas for differential equations the cosine functions express a particular choice of boundary conditions.

In particular, the DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT) [46] , but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output data are shifted by half a sample. There are eight standard DCT variants [3], of which four are common.

The most common variant of discrete cosine transform is the type-II DCT [3], which is often called simply “the DCT”; its inverse is called simply “the inverse DCT” or “IDCT” [3]. The

2D DCT of a square or a rectangular block is used for almost all block-based transform schemes for image and video coding. The conventional 2D DCT is implemented separately through two 1D transforms, one along the vertical direction and the other along the horizontal direction, as shown in Figure 3.1. The conventional DCT seems to be the best choice for image blocks in which vertical and/or horizontal edges are dominating.

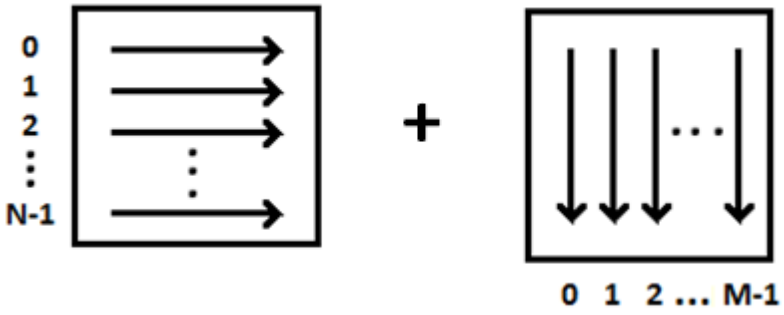


Figure 3.1 2D DCT implementation: A combination of 1D DCTs along horizontal and vertical directions [3]

3.1.1.1 Forward 2D DCT (NXM) [2]

$$X^{c2}(k,l) = \frac{4}{MN} C(k)\, C(l) \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x(n,m) \cos\left[\frac{2m+1\ l\pi}{2M}\right] \cos\left[\frac{2n+1\ k\pi}{2N}\right]$$

$k = 0,1, \ldots, N-1$ $\quad C(p) = \frac{1}{\sqrt{2}}, p=0$ $\quad p=k,l$

$l = 0,1, \ldots, M-1$ $\quad C(p) = 1, p \neq 0$

x(n,m) = samples in the 2D data domain

$X^{c2}(k,l)$ = coefficients in the 2D DCT domain

x(n,m) = Samples in the 2D data domain

$X^{c2}(k,l)$ = Coefficients in the 2D- DCT domain

3.1.1.2 Inverse 2D DCT (NXM)

$$x(n,m) = \sum_{k=0}^{N-1} \left(\sum_{l=0}^{M-1} C(l)\, X^{C2}(k,l) \cos\left[\frac{2m+1\ l\pi}{2M} \right] \right) \cos\left[\frac{2n+1\ k\pi}{2N} \right]$$

$n = 0,1, \ldots, N-1$

$m = 0,1, \ldots, M-1$

x(n,m) = Samples in the 2D data domain.

X^{C2} (k, l) = Coefficients in the 2D-DCT domain.

The transform used by H.264/AVC [3] to process both intra and inter prediction residuals is related to an integer 2D DCT, implemented using 1 D DCTs horizontally, followed by 1D DCTs vertically or vice versa.

3.2 Intra coding in H.264 [1]

The existing intra prediction algorithm in H.264 using rate distortion optimization (RDO) [36] 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 as shown in Figure 3.2, and 4 modes for a 16X16 luma block as shown in Figure 3.3 and a 8X8 chroma block respectively.

A- H → They are the previously coded pixels of the upper macroblock and are available both at encoder and decoder.

I-L → They are the previously coded pixels of the left macroblock and are available both at encoder and decoder.

M → it is the previously coded pixel of the upper left macroblock and available both at the encoder and decoder.

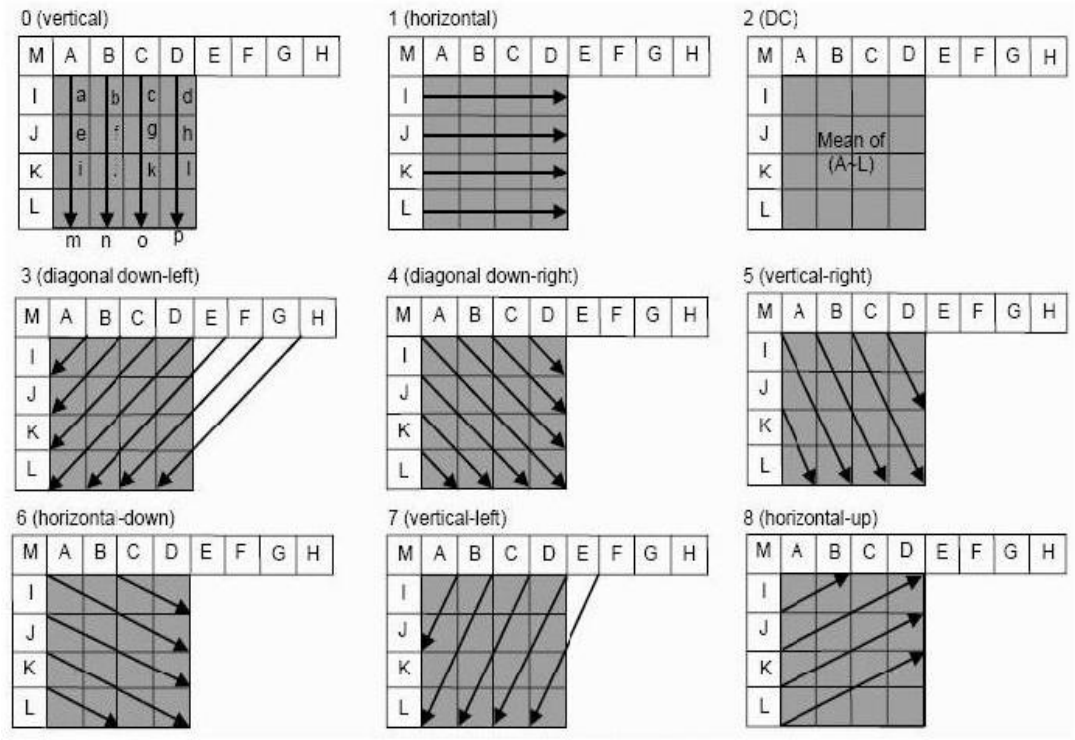


Figure 3.2 Intra 4X4 prediction mode directions (vertical, 0; horizontal, 1; DC, 2; diagonal down left, 3; diagonal down right, 4; vertical right, 5; horizontal down, 6; vertical left, 7; and horizontal up, 8) [4]

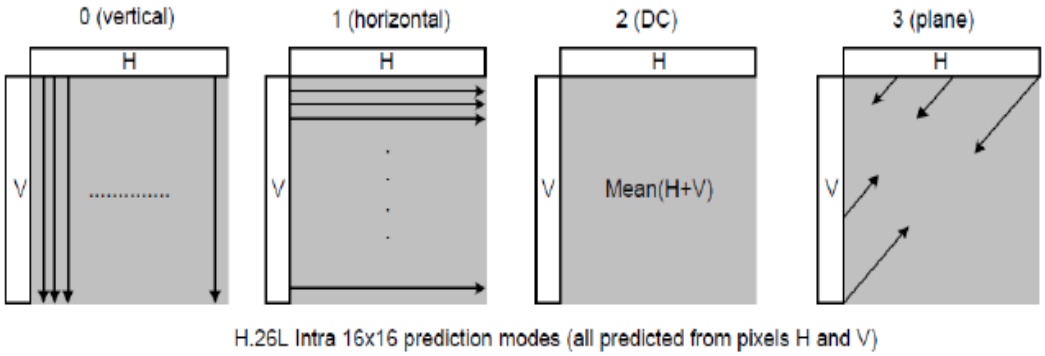


Figure 3.3 16X16 luma intra prediction modes [3]

i) 16X16 for Luma

- Mode 0(vertical): extrapolation from upper samples (H).
- Mode 1(horizontal): extrapolation from left samples (V)

- Mode 2 (DC): mean of upper and left-hand samples (H+V).
- Mode 3 (Plane): a linear “plane” function is fitted to the upper and left-hand samples H and V. This works well in areas of smoothly-varying luminance.

ii) 4X4 for Luma

Mode 0: Vertical

Mode 1: Horizontal

Mode 2: DC

Mode 3: Diagonal-down-left

Mode 4: Diagonal-down-right

Mode 5: Vertical-right

Mode 6: Horizontal-down

Mode 7: Vertical-left

Mode 8: Horizontal-up

Intra coding predicts the image content based on the value of previously decoded pixels. It has 9 prediction modes for 4x4 blocks, 9 prediction modes for 8x8 blocks, and 4 prediction modes for 16x16 blocks. For each intra prediction mode, an intra prediction algorithm is used to predict the image content in the current block based on decoded neighbors. The intra prediction errors are transformed using a DCT-like transform. The transform coefficients are then quantized, scanned into a 1D signal, and entropy coded using CAVLC [1] or CABAC [1].

In this framework, DDCT is to replace the AVC transforms by a set of transforms that taking into account the prediction mode of the current block. Hence, DDCT provides 9 transforms for 4x4, 9 transforms for 8x8, and 4 transforms for 16x16, although many of them are the same or can be simply inferred from a core transform. For each transform, the DDCT also

provides a fixed scanning pattern based on the QP and the intra prediction mode to replace the zigzag scanning pattern of DCT coefficients in AVC.

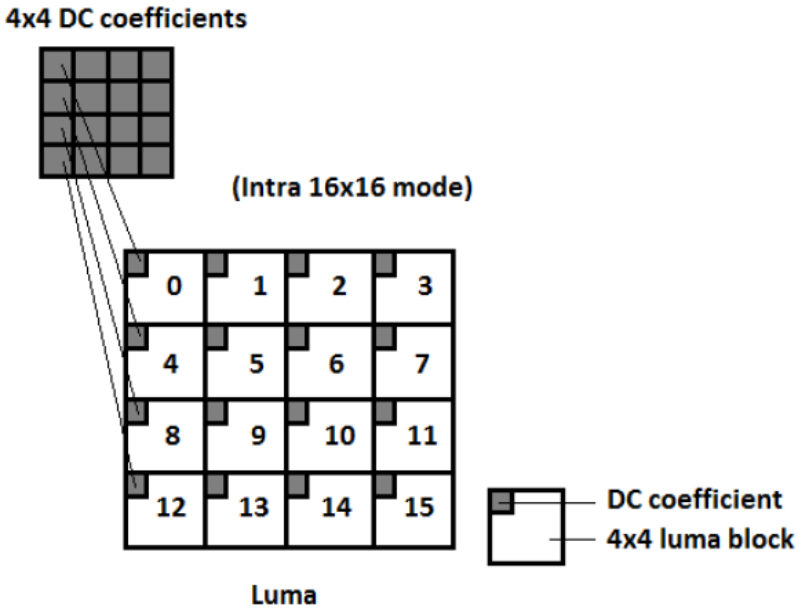


Figure 3.4 4X4 DC coefficients for intra 16X16 mode [3]

3.3 Modes of Directional DCT [25]

In H.264, there are total of eight directional prediction modes (the dc mode—Mode 2—is not counted) for blocks of size 4X 4. Among these modes, one is the vertical prediction (Mode 0), one is the horizontal prediction (Mode 1), and the remaining are named as diagonal down-left (Mode 3), diagonal down-right (Mode 4), vertical-right (Mode 5), horizontal-down (Mode 6), vertical-left (Mode 7), and horizontal-up (Mode 8), respectively [25].

This idea can be readily applied to any block size to define the same eight directional modes. For instance, Figure 3.5 shows six directional modes (Modes 3–8) [25]. It is easy to find that Mode 4 can be obtained by flipping Mode 3 either horizontally or vertically; Mode 6 can be obtained by transposing Mode 5, and Mode 7/8 can be obtained by flipping Mode 5/6 either

horizontally or vertically. To make our results general enough, we consider an arbitrary block size and will first develop a truly directional DCT for the diagonal down-left mode, and then discuss the extension to other modes and some further extensions.

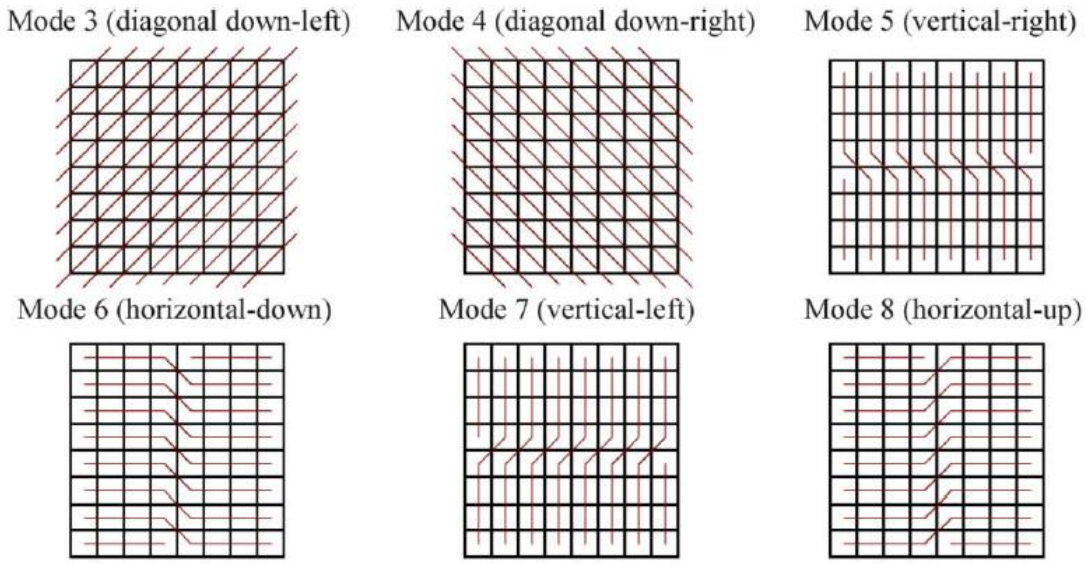


Figure 3.5 Six directional modes defined in a similar way as was used in H.264 for the block size 8X8. The vertical and horizontal modes are not included here [25].

3.3.1 MODE 3 - Directional DCT for Diagonal Down Left

As shown in Figure 3.6, the first 1-D DCT will be performed along the diagonal down-left direction, i.e., for the diagonal line with i + j = k, k = 0, 1, …, 2N – 2. There are in total 2N – 1 diagonal down-left DCTs to be done, whose lengths are $[N_k]$ = [1, 2, …., N-1, n, N-1, …., 2,1]. All of coefficients after these DCTs are expressed into group of column vectors.

$$\boldsymbol{A}_k = [A_{0,k}, A_{1,k}, \ldots, A_{N_k-1,k}]^T, \quad k = 0, 1, \ldots, 2N-2.$$

Notice that each column A_k has a different length N_k, with the dc component placed at top, followed by the first ac component and so on.

Next, the second 1-D DCT is applied to each row that can be expressed as $[\hat{A}_{u,v}]_{v=0:2N-2-2u}$ for u = 0, 1, ., N-1 . The coefficients after the second DCT are pushed horizontally to the left and denoted as $[\hat{A}_{u,v}]_{v=0:2N-2-2u}$ for u = 0, 1, …, N-1. The right part of Fig. 3.7 shows a modified zigzag scanning that will be used to convert the 2-D coefficient block into a 1-D sequence so as to facilitate the runlength-based VLC.

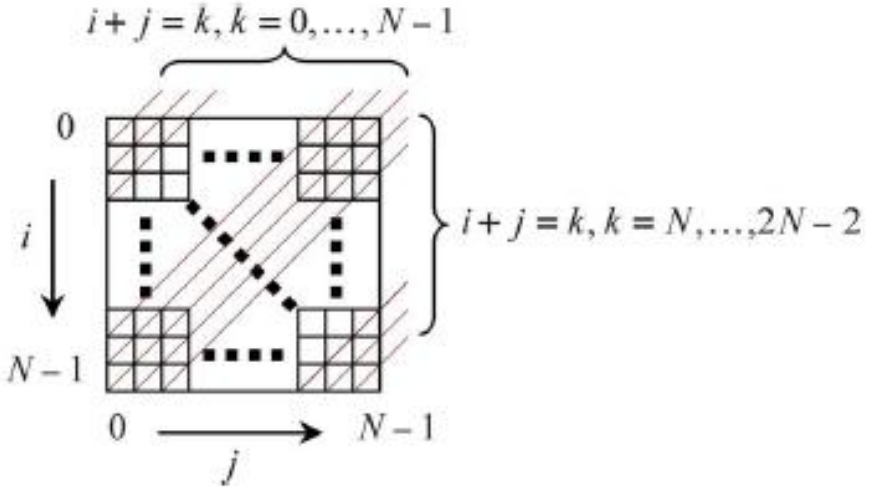


Figure 3.6 NXN image block in which the first 1-D DCT will be performed along the diagonal down-left direction [25]

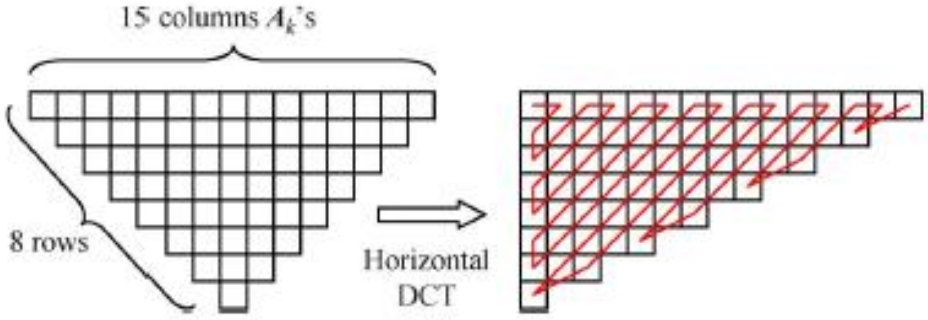


Figure 3.7 Example of N=8; arrangement of coefficients after the first DCT (left) and arrangement of coefficients after the second DCT as well as the modified zigzag scanning (right) [25]

Stepwise working of mode 3 DDCT in H.264 for 4X4 block:

- Step 1: As shown in Figure 3.8, X00, X01, ….., X33 are the pixels in the 2D spatial domain.
- Step 2: 1D DCT is performed for the 4X4 block in diagonal down-left position with lengths L= 1, 2, 3, 4, 3, 2 and 1 as shown in Figure 3.9.
- Step 3: The coefficients of step 2 after 1 D DCT are arranged vertically in the same pattern as shown in Figure 3.10. Then apply horizontal 1 D DCT for lengths L = 7, 5, 3 and 1 and arrange in the same pattern.
- Step 4: Apply horizontal 1 D DCT for lengths L= 7, 5, 3 and 1. The coefficients are arranged the same pattern as shown in the Figure 3.11.
- Step 5: After step 4, move all 2D (4X4) directional DCT coefficients to the left as shown in Figure 3.12. Implement quantization followed by 2D VLC for compression/coding zigzag scan. This scanning helps to increase the run-length of zero (transform) coefficients leading to reduce bit rate in 2D-VLC coding (similar to JPEG [8]).

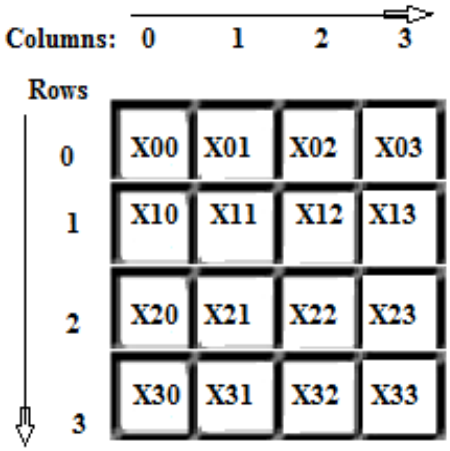


Figure 3.8 Pixels in the 2D spatial domain for a 4X4 block

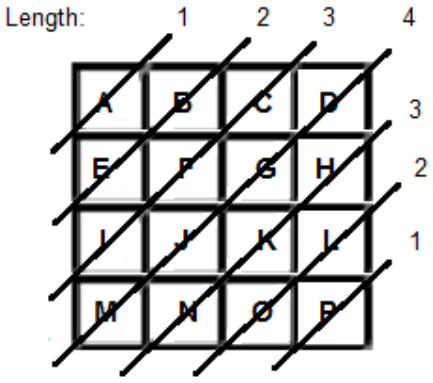


Figure 3.9 1D DCT performed for 4X4 block for a diagonal down left for lengths = 1, 2, 3, 4, 3, 2 and 1

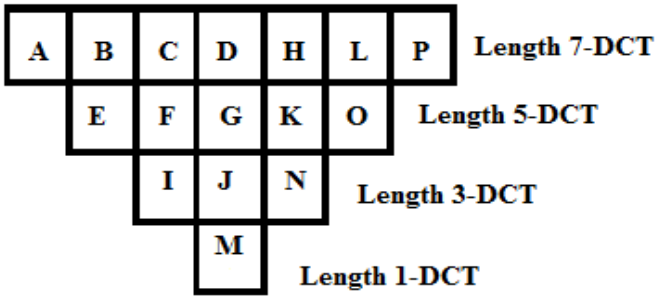


Figure 3.10 Coefficients of 1D DCT arranged vertically for step 4

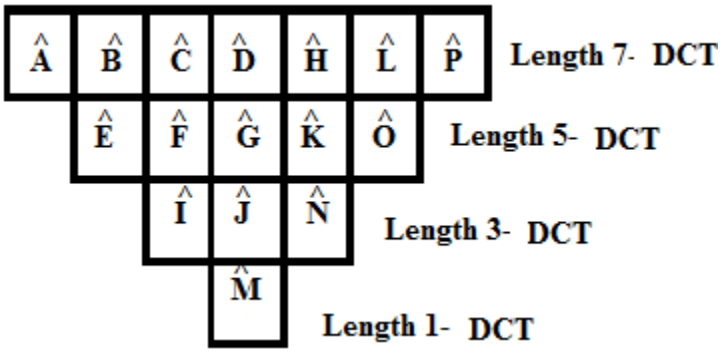


Figure 3.11 1D DCT applied horizontally for lengths = 7, 5, 3 and 1

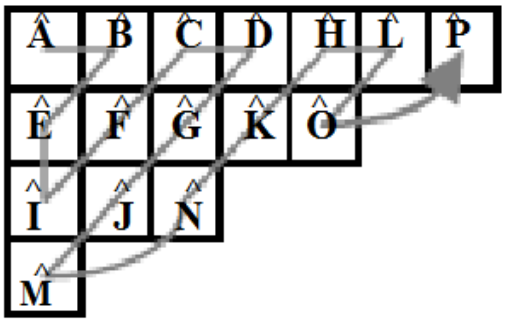


Figure 3.12 Move all 2D (4X4) Directional DCT coefficients to the left. Implement quantization followed by 2D VLC for compression/coding zigzag scan

3.3.2 MODE 4 - Directional DCT for Diagonal Down Right

Stepwise working of mode 4 DDCT in H.264 for 4X4 block:

- Step 1: As shown in Figure 3.13, X00, X01, ….., X33 are the pixels in the 2D spatial domain.
- Step 2: 1D DCT is performed for the 4X4 block in diagonal down-right position with lengths L= 1, 2, 3, 4, 3, 2 and 1 as shown in Figure 3.14
- Step 3: The coefficients of step 2 after 1 D DCT are arranged vertically in the same pattern as shown in Figure 3.15. Then apply horizontal 1 D DCT for lengths L = 7, 5, 3 and 1 and arrange in the same pattern.
- Step 4: Apply horizontal 1 D DCT for lengths L= 7, 5, 3 and 1. The coefficients are arranged the same pattern as shown in the Figure 3.16
- Step 5: After step 4, move all 2D (4X4) directional DCT coefficients to the left. Implement quantization followed by 2D VLC for compression/coding zigzag scan as shown in Figure 3.17.

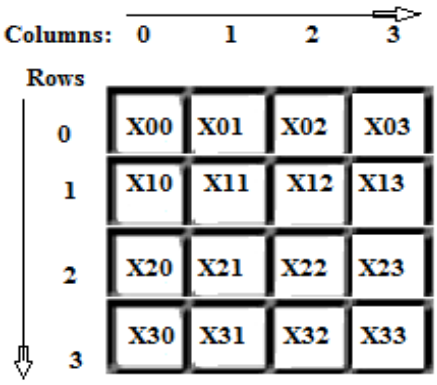


Figure 3.13 Pixels in the 2D spatial domain for a 4X4 block

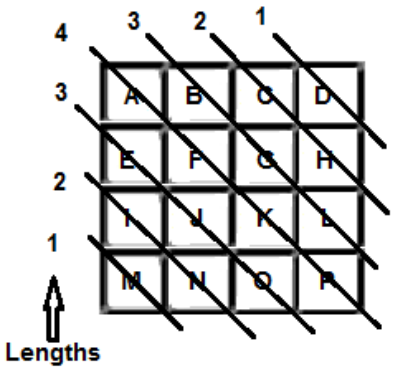


Figure 3.14 DCT performed for 4X4 block for a diagonal down right for lengths = 1, 2, 3, 4, 3, 2 and 1

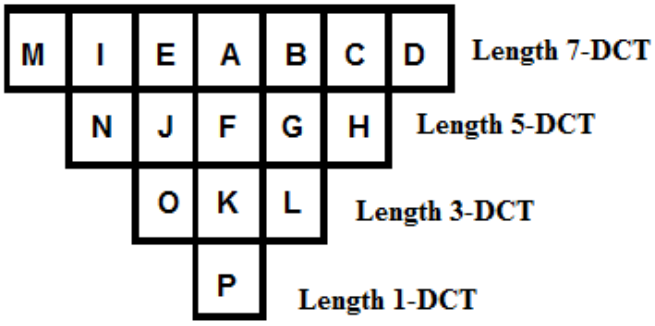


Figure 3.15 Coefficients of 1D DCT arranged vertically for step 4

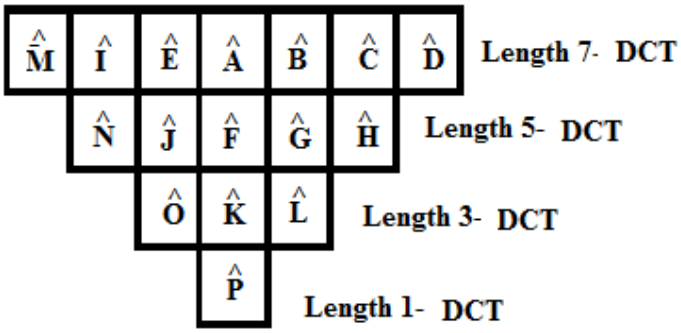


Figure 3.16 1D DCT applied horizontally for lengths = 7, 5, 3 and 1

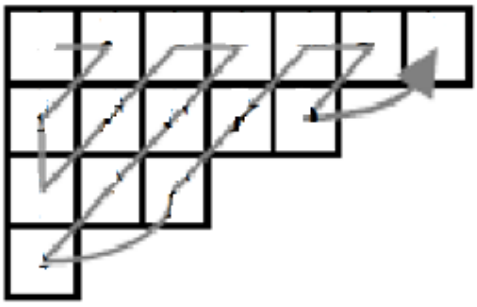


Figure 3.17 Move all 2D (4X4) directional DCT coefficients to the left. Implement quantization followed by 2D VLC for compression/coding zigzag scan

3.3.3 MODE 5 - Directional DCT for Vertical Right

Stepwise working of mode 5 DDCT in H.264 for 4X4 block:

- Step 1: As shown in Figure 3.18, X00, X01, ….., X33 are the pixels in the 2D spatial domain.
- Step 2: 1D DCT is performed for the 4X4 block in vertical-right position with lengths L= 2,4,4,4,2 as shown in Figure 3.19.
- Step 3: The coefficients of step 2 after 1 D DCT are arranged vertically in the same pattern as shown in Figure 3.20. Then apply horizontal 1 D DCT for lengths L = 5, 5, 3 and 3and arrange in the same pattern.

- Step 4: Apply horizontal 1 D DCT for lengths L= 5, 5, 3 and 3. The coefficients are arranged the same pattern as shown in the Figure 3.21.
- Step 5: After step 4, move all 2D (4X4) Directional DCT coefficients to the left. Implement quantization followed by 2D VLC for compression/coding zigzag scan as shown in Figure 3.22. This scanning helps to increase the run-length of zero (transform) coefficients leading to reduce bit rate in 2D-VLC coding (similar to JPEG).

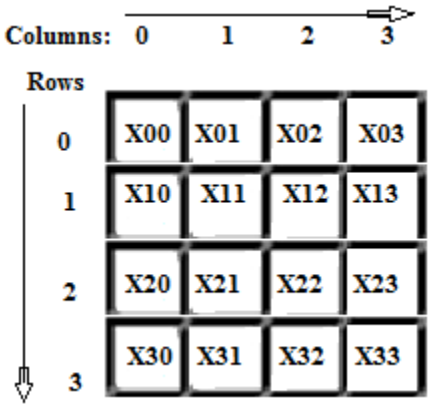


Figure 3.18 Pixels in the 2D spatial domain for a 4X4 block

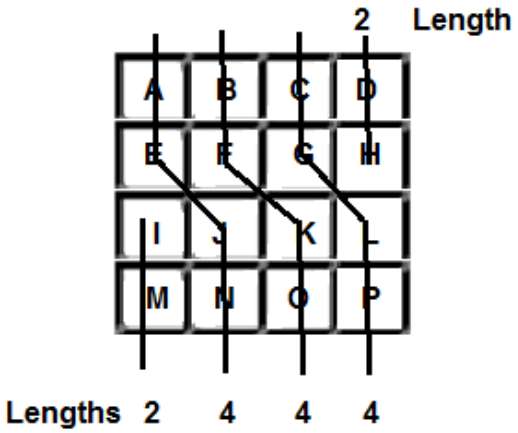


Figure 3.19 DCT performed for 4X4 block for vertical right, for lengths = 2, 4, 4, 4 and 2

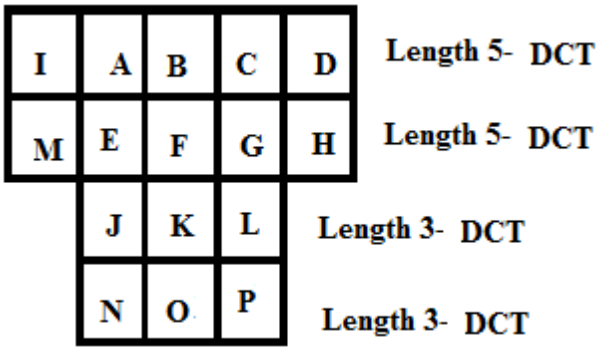


Figure 3.20 Coefficients of 1D DCT arranged vertically for step 4

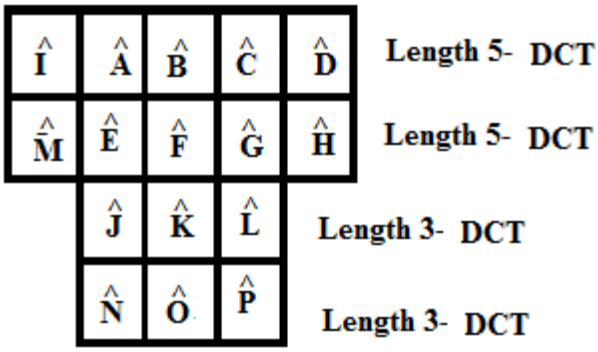


Figure 3.21 1D DCT applied horizontally for lengths = 5, 5, 3 and 3

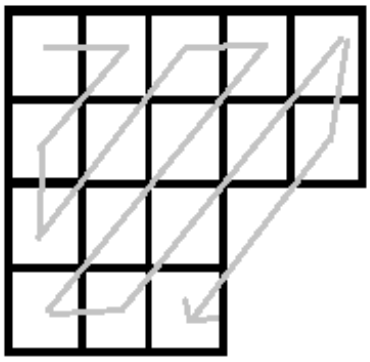


Figure 3.22 Move all 2D (4X4) Directional DCT coefficients to the left. Implement quantization followed by 2D VLC for compression/coding zigzag scan

3.3.4 MODE 6 - Directional DCT for Horizontal down

Stepwise working of mode 6 DDCT in H.264 for 4X4 block:

- Step 1: As shown in Figure 3.23, X00, X01, ….., X33 are the pixels in the 2D spatial domain.
- Step 2: 1D DCT is performed for the 4X4 block in Horizontal down position with lengths L= 2, 4, 4, 4 and 2 as shown in Figure 3.24.
- Step 3: The coefficients of step 2 after 1 D DCT are arranged vertically in the same pattern as shown in Figure 3.25. Then apply horizontal 1 D DCT for lengths L = 5, 5, 3 and 3 and arrange in the same pattern.
- Step 4: Apply horizontal 1 D DCT for lengths L= 5, 5, 3 and 3. The coefficients are arranged the same pattern as shown in the Figure 3.26
- Step 5: After step 4, move all 2D (4X4) directional DCT coefficients to the left. Implement quantization followed by 2D VLC for compression/coding zigzag scan as shown in Figure 3.27. This scanning helps to increase the run-length of zero (transform) coefficients leading to reduce bit rate in 2D-VLC coding (similar to JPEG).

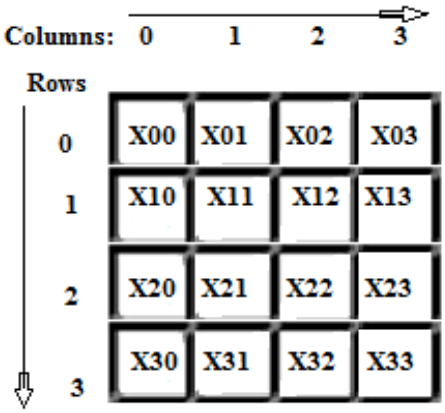


Figure 3.23 Pixels in the 2D spatial domain for a 4X4 block

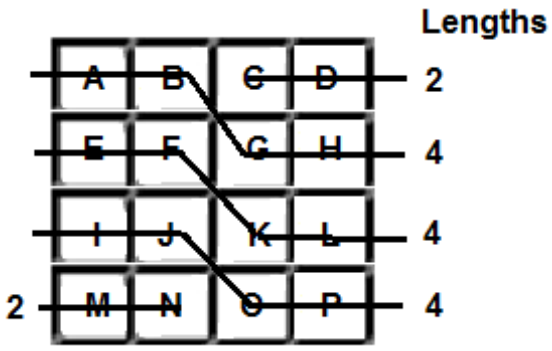


Figure 3.24 DCT performed for 4X4 block for horizontal down, lengths = 2, 4, 4, 4 and 2

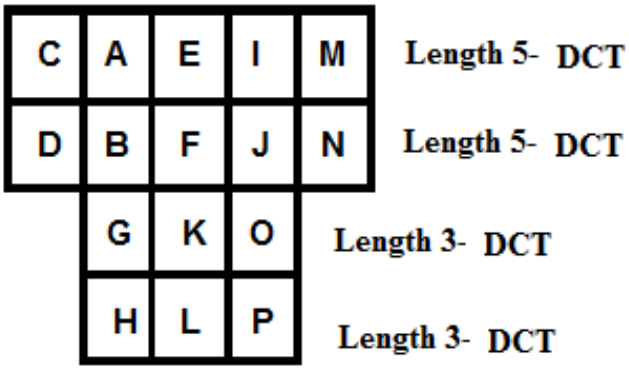


Figure 3.25 Coefficients of 1D DCT arranged vertically for step 4

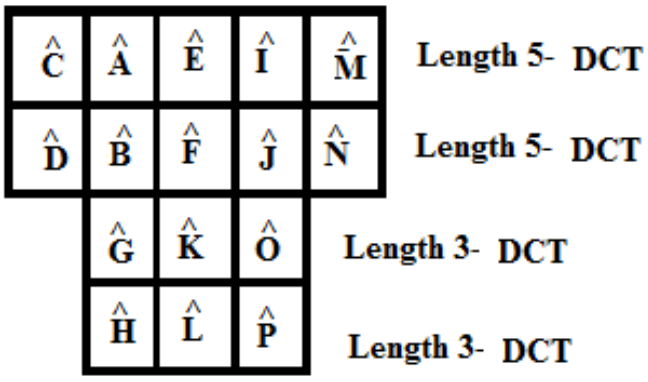


Figure 3.26 1D DCT applied horizontally for lengths = 5, 5, 3 and 3

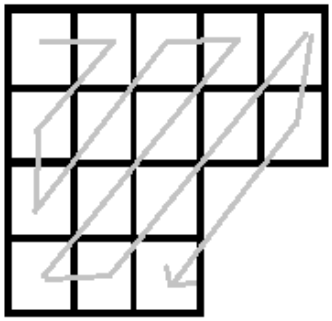


Figure 3.27 Move all 2D (4X4) directional DCT coefficients to the left. Implement quantization followed by 2D VLC for compression/coding zigzag scan

3.3.5 MODE 7 - Directional DCT for vertical left

Stepwise working of mode 7 DDCT in H.264 for 4X4 block:

- Step 1: As shown in Figure 3.28, X00, X01, ….., X33 are the pixels in the 2D spatial domain.
- Step 2: 1D DCT is performed for the 4X4 block in Vertical left position with lengths L= 2,4,4,4 and 2 as shown in Figure 3.29.
- Step 3: The coefficients of step 2 after 1 D DCT are arranged vertically in the same pattern as shown in Figure 3.30. Then apply horizontal 1 D DCT for lengths L = 5, 5, 3 and 3 and arrange in the same pattern.
- Step 4: Apply horizontal 1 D DCT for lengths L= 5, 5, 3 and 3. The coefficients are arranged the same pattern as shown in the Figure 3.31.
- Step 5: After step 4, move all 2D (4X4) Directional DCT coefficients to the left. Implement quantization followed by 2D VLC for compression/coding zigzag scan as shown in Figure 3.32.

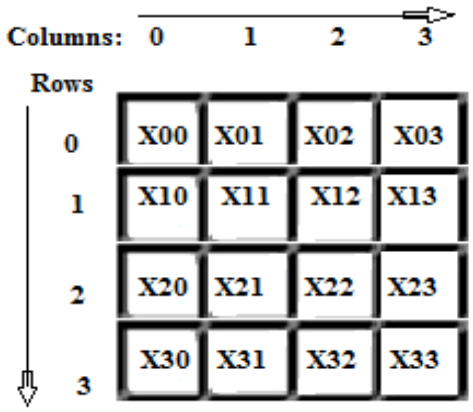


Figure 3.28 Pixels in the 2D spatial domain for a 4X4 block

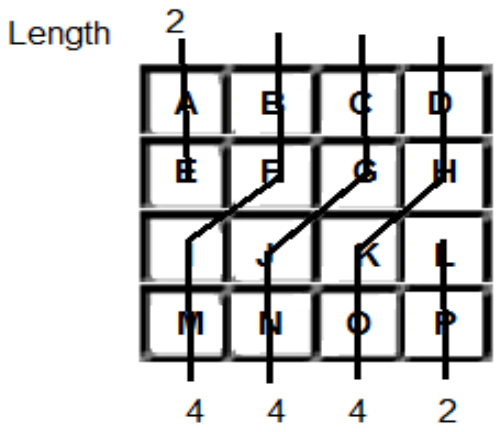


Figure 3.29 DCT performed for 4X4 block for a vertical left for lengths = 2, 4, 4, 4 and 2

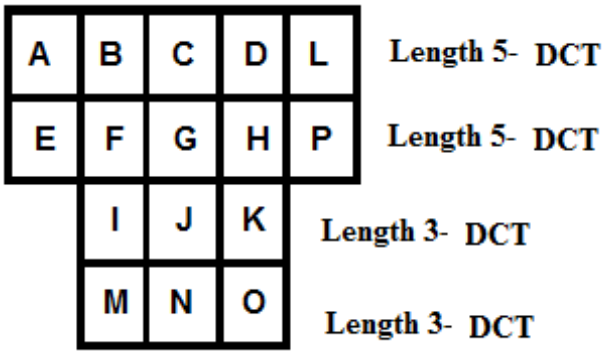


Figure 3.30 Coefficients of 1D DCT arranged vertically for step 4

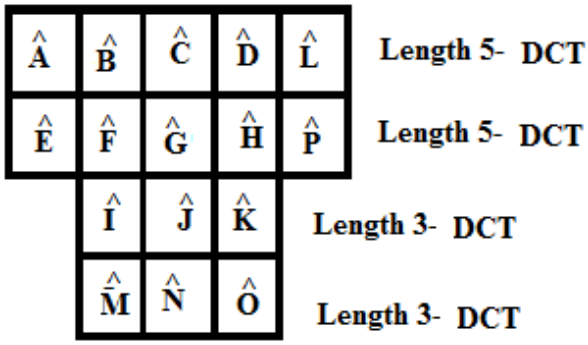


Figure 3.31 1D DCT applied horizontally for lengths = 5, 5, 3 and 3

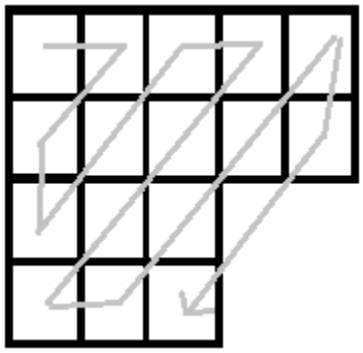


Figure 3.32 Move all 2D (4X4) directional DCT coefficients to the left. Implement quantization followed by 2D VLC for compression/coding zigzag scan

3.3.6 MODE 8 - Directional DCT for Horizontal Up

Stepwise working of mode 8 DDCT in H.264 for 4X4 block:

- Step 1: As shown in Figure 3.33, X00, X01, ….., X33 are the pixels in the 2D spatial domain.
- Step 2: 1D DCT is performed for the 4X4 block in Horizontal Up position with lengths L= 2, 4, 4, 4 and 2 as shown in Figure 3.34.

- Step 3: The coefficients of step 2 after 1 D DCT are arranged vertically in the same pattern as shown in Figure 3.35. Then apply horizontal 1 D DCT for lengths L = 5, 5, 3 and 3 and arrange in the same pattern.
- Step 4: Apply horizontal 1 D DCT for lengths L= 5, 5, 3 and 3. The coefficients are arranged the same pattern as shown in the Figure 3.36.
- Step 5: After step 4, move all 2D (4X4) directional DCT coefficients to the left. Implement quantization followed by 2D VLC for compression/coding zigzag scan as shown in figure 3.37. This scanning helps to increase the run-length of zero (transform) coefficients leading to reduce bit rate in 2D-VLC coding (similar to JPEG [4]).

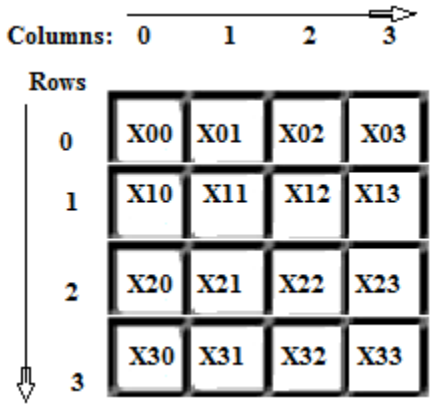


Figure 3.33 Pixels in the 2D spatial domain for a 4X4 block

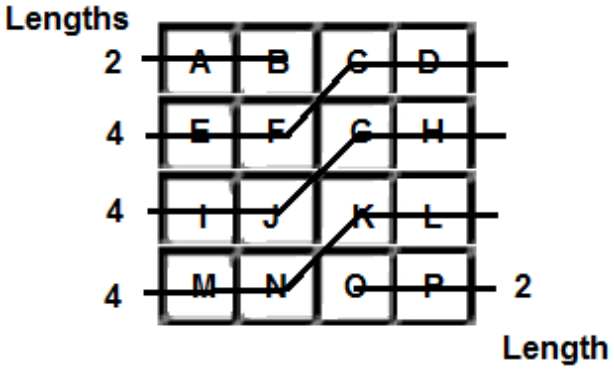


Figure 3.34 DCT performed for 4X4 block for horizontal up for lengths = 2, 4, 4, 4 and 2

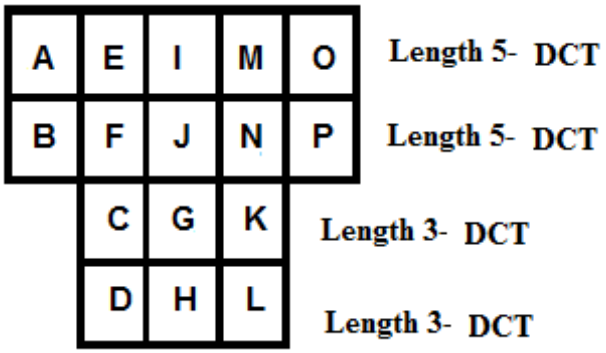


Figure 3.35 Coefficients of 1D DCT arranged vertically for step 4

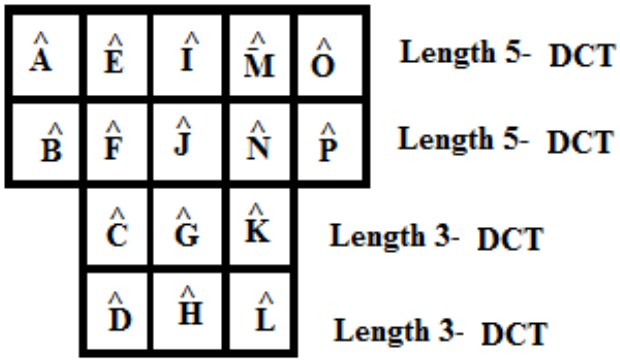


Figure 3.36 1D DCT applied horizontally for lengths = 5, 5, 3 and 3

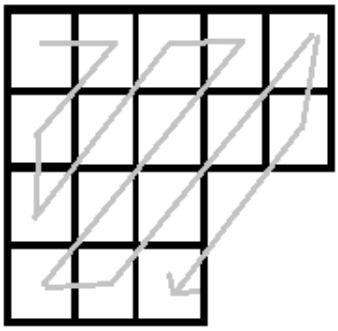


Figure 3.37 Move all 2D (4X4) directional DCT coefficients to the left. Implement quantization followed by 2D VLC for compression/coding zigzag scan

3.4 How to obtain a mode from other modes [31]:

Although there are 22 DDCTs for 22 intra prediction modes (9 modes for 4x4, 9 modes for 8x8, and 4 modes for 16x16), these transforms can be derived, using simple operators such as rotation and/ or reflection, from only 7 different core modes:

8x8 and 4x4:

- Modes 0, 1: The same transform similar to AVC, DCT is used, first horizontally, then vertically.
- Modes 3 and 4: The DDCT for mode 3 can be obtained from the transform for mode 4 using a reflection on the vertical line at the center of the block, as shown in Figure 3.38
- Modes 5 to 8: The DDCT for modes 6-8 can be derived from mode 5 using reflection and rotation this is shown in Figures 3.39, 3.40 and 3.41.

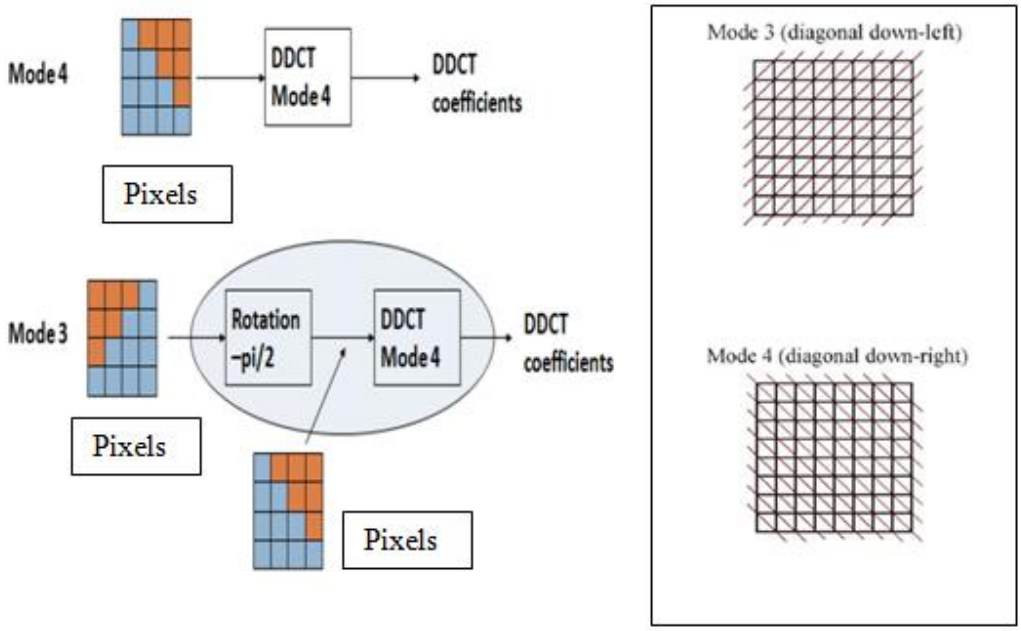


Figure 3.38 Obtaining mode 3 by rotation – π/2 and DDCT of Mode 4 [31]

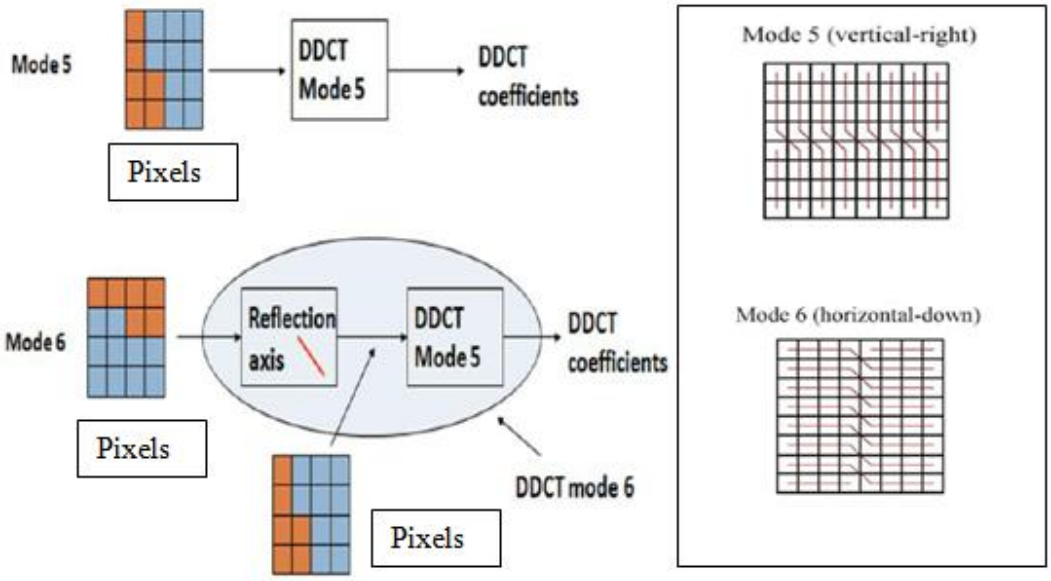


Figure 3.39 Obtaining mode 6 by reflection across axis and DDCT of mode5 [31]

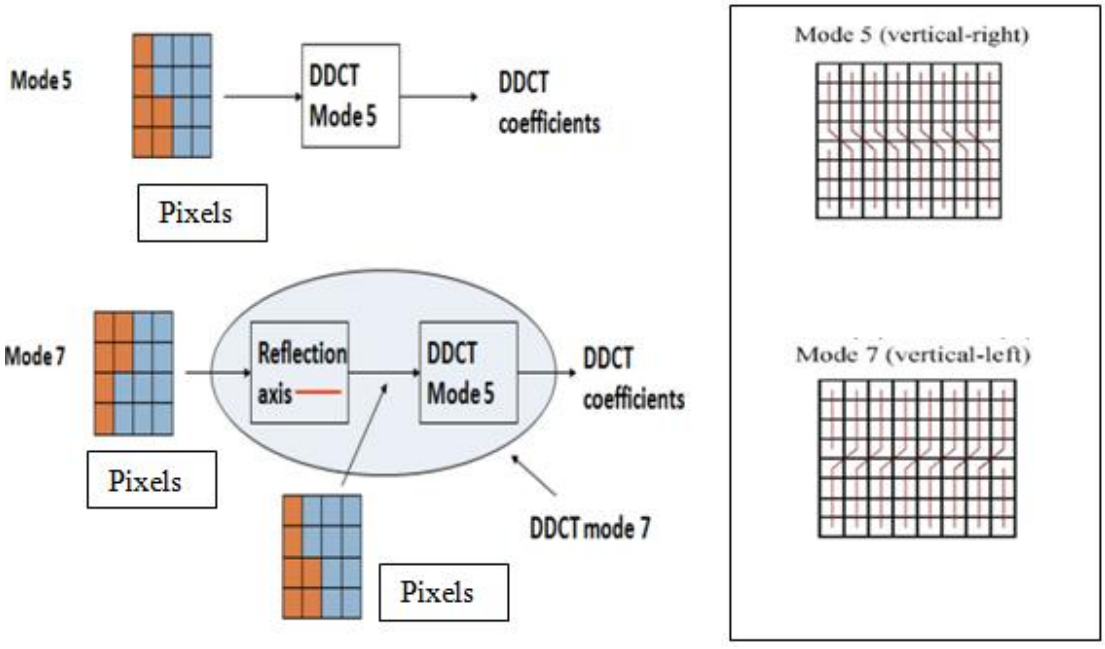


Figure 3.40 Obtaining mode 7 by reflection across horizontal axis and DDCT of mode 5 [31]

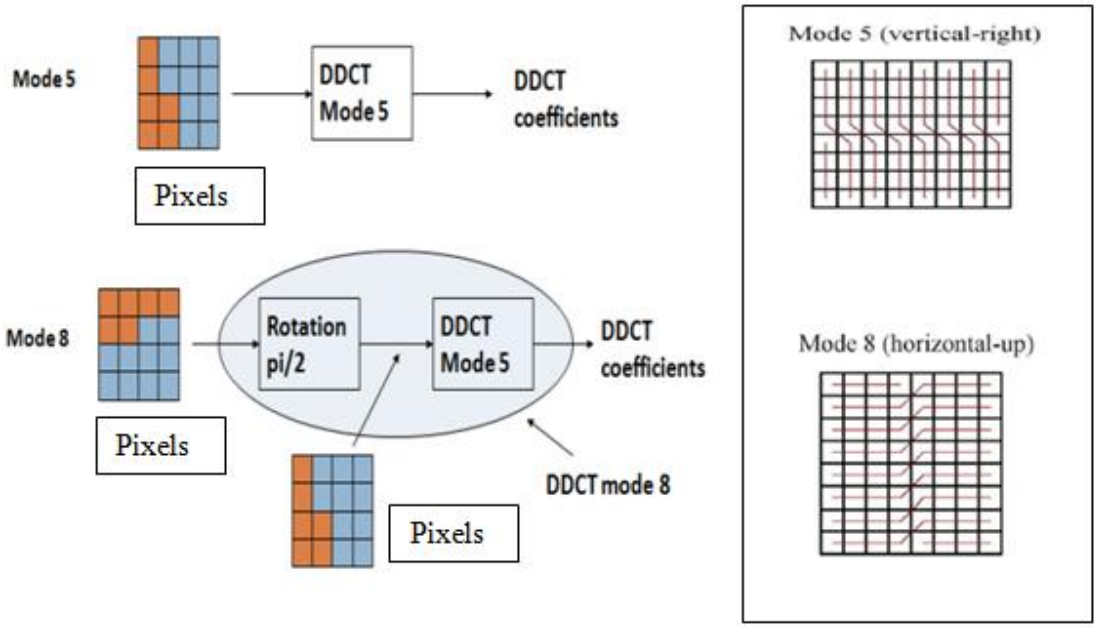


Figure 3.41 Obtaining mode 8 by rotation π/2 of pixels and DDCT of mode 8 [31]

3.5 Summary

The various modes in DDCT and its implementation in H.264 are shown above. Next chapter deals with the results obtained by implementing DDCT on an image as well as in H.264 JM 18.0 [24] for intra frame.

CHAPTER 4

IMPLEMENTATION AND ANALYSIS OF DDCT IN H.264

4.1 Introduction

The directional discrete cosine transforms (DDCT) is a set of transforms for applying to the intra prediction errors in the video compression framework AVC [28]. In this section, description and properties of DDCT are described.

4.2 Directional DCT of Image coding

Human eyes are highly sensitive to vertical and horizontal edges within each image. Meanwhile a lot of blocks in an image do contain a vertical and/or horizontal edge(s). Thus, the conventional DCT seems to be the best choice for image blocks in which vertical and/or horizontal edges are dominating. On the other hand, however there may also exist other directions in one image block that are perhaps as equally important as the vertical/horizontal directions, e.g., two diagonal directions. The conventional DCT would be unlikely to be the best choice for image blocks in which some directional edges other than the vertical/horizontal ones dominate. Such a belief motivates to attempt to develop a “directional” DCT framework so that the best “directional” DCT can be chosen according to the dominating edge(s) within each individual image block. The results later on demonstrate that this framework can indeed improve the coding performance remarkable.

Figure 4.1 shows the block wise steps taken to perform DDCT on an image and get back the reconstructed image back. The steps taken for this coding are:

Step 1: Dividing the entire image into 8X8 blocks along the raster scan

Step 2: For each 8X8 block, apply 8 DDCT modes to check which gives the best coding performance for that particular block and image.

Step 3: After the 2 D DDCT, move all the coefficients of 8X8 block to the left as shown in Figure 3.37 for the zig-zag scanning. Scanning is performed to group low frequency coefficients in top of vector.

Step 4: Performs quantization to round off most of the coefficients to zero or to the nearest level. This completes the encoded part. Send these encoded bit streams for transmission over the medium.

Step 5: At the decoder side, inverse quantization takes place with the same quantization level that is used at the encoder side.

Step 6: Move coefficients to the position as in step 2.

Step 7: Apply inverse DDCT (IDDCT) for each 8X8 block of the image to get the pixel values.

Step 8: Regroup the 8X8 block to get the whole image or frame.

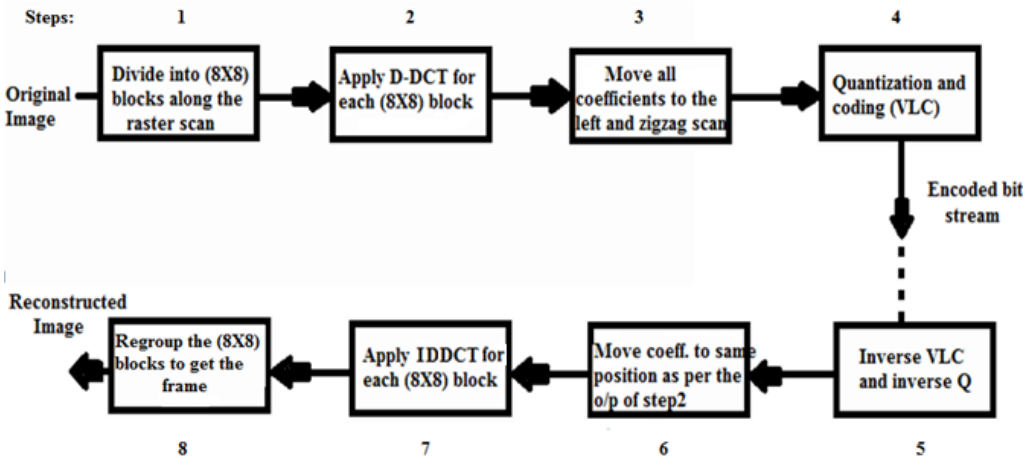


Figure 4.1 Stepwise computation of DDCT on an Image

4.3 Eigen or Basis images

Mapping of a 2D data array into a 2D DCT domain implies decomposing the 2D data array into the basis images of the DCT. Equation 4.1 shows the 2D DCT for a 4X4 matrix and Equation 4.2 shows the DCT matrix for the same 4X4 block.

4X4 2D DCT is:

$$X_{u,v}^{c2} = \frac{1}{4} c_u c_v \sum_{n=0}^{3} \sum_{m=0}^{3} x_{n,m} \cos\left[\frac{(2n+1)u\pi}{8}\right] \cos\left[\frac{(2m+1)v\pi}{8}\right], \qquad (4.1)$$

$$u, v = 0, 1, 2, 3, \qquad c_k = \begin{cases} 1/\sqrt{2}, & k = 0 \\ 1, & k \neq 0 \end{cases}$$

The (4X4) DCT matrix is:

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & (1 & 1 & 1 & 1) \\ & C^1 & C^3 & C^5 & C^7 \\ & C^2 & C^6 & C^{10} & C^{14} \\ & C^3 & C^9 & C^{15} & C^{21} \end{bmatrix}, \qquad C^k = \cos\left(\frac{k\pi}{8}\right) \qquad (4.2)$$

The basis images are obtained by the outer (vector) product of each basis vector with all the basis vectors. Two dimensional (4X4) DCT implies decomposing 2D (4X4) data array into 16 basis images, (4X4) image array. Equation 4.3 gives the lowest frequency, top left basis image. This calculation, when applied to the entire 4X4 block, 16 basis images from basis image (0, 0) to (3, 3) are obtained.

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} & (1\ 1\ 1\ 1) & \frac{1}{\sqrt{2}} \\ & (4 \times 1) & (1 \times 4) & \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \qquad (4.3)$$

$$= \text{basis image } (0, 0)$$

The highest frequency (bottom right) basis image is given by Equation 4.4.

$$\left[\begin{pmatrix} C^3 \\ C^9 \\ C^{15} \\ C^{21} \end{pmatrix} (C^3\ C^9\ C^{15}\ C^{21})\right] = \begin{bmatrix} C^3 & (C^3\ C^9\ C^{15}\ C^{21}) \\ C^9 & (C^3\ C^9\ C^{15}\ C^{21}) \\ C^{15} & (C^3\ C^9\ C^{15}\ C^{21}) \\ C^{21} & (C^3\ C^9\ C^{15}\ C^{21}) \end{bmatrix} \tag{4.4}$$

$$= \text{basis image } (3,3),$$
$$C^k = \cos(k\pi/8)$$

4.3.1 Basis Images for different Modes

Figures 4.2 and 4.7 gives the diagrammatic representation and computation for a mode 3 4X4 basis images. As shown in Figure 4.2 (b), the 1-D DCT is computed for each row for lengths 7, 5, 3 and 1 with the 1st coefficient value as 1 and rest others zero. When all pixels are zero of any length, then the corresponding length DCT yields only zeros.

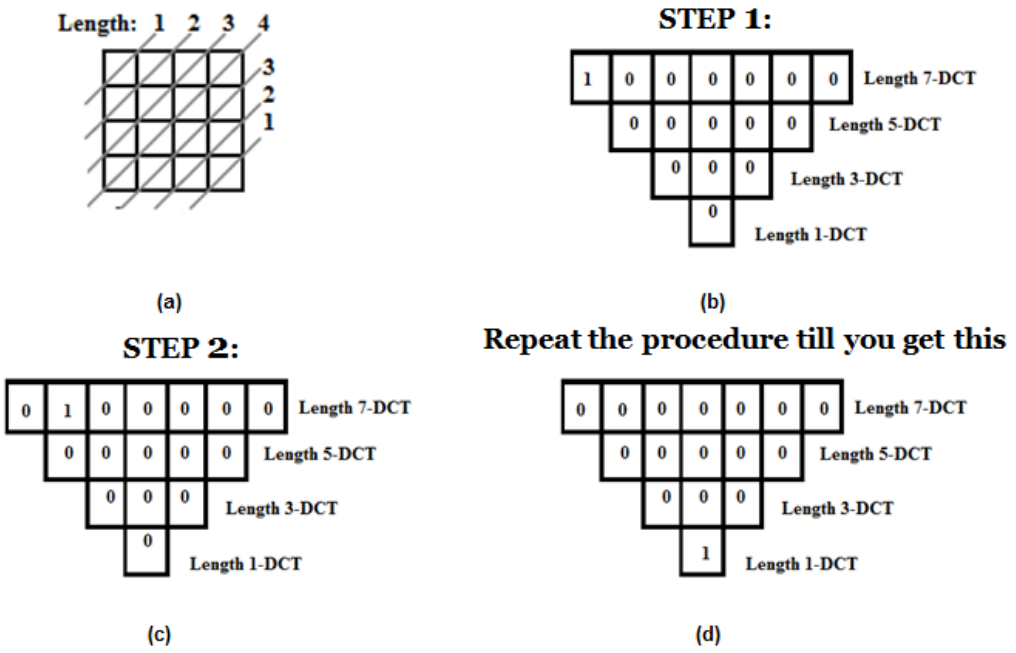


Figure 4.2 Computation of basis images for diagonal down left (a) The original 4X4 block with diagonal down left computation (b) The 1 D DCT of coefficients for lengths 7, 5, 3 and 1 for basis image (0, 0) (c) The 1 D DCT of coefficients for lengths 7, 5, 3 and 1 for basis image (0,1) (d) The 1 D DCT of coefficients for lengths 7, 5, 3 and 1 for basis image (3,3)

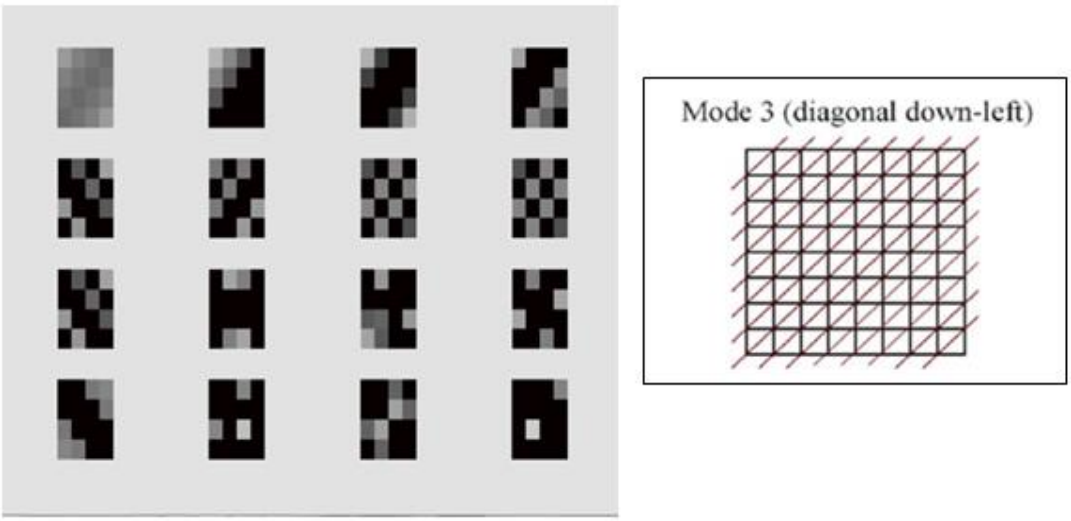


Figure 4.3 Basis images for Mode 3 – diagonal down left for a 4X4 block

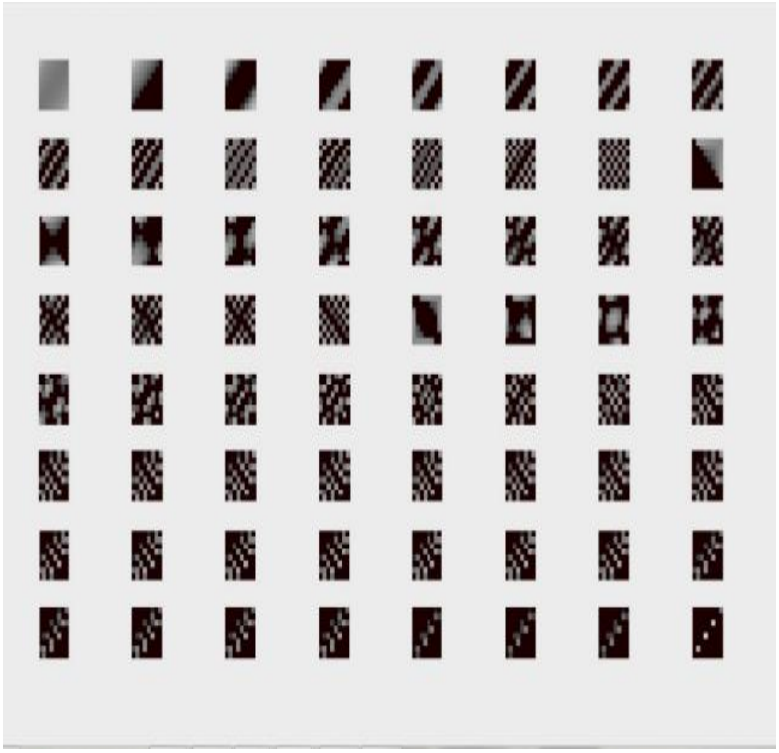


Figure 4.4 Basis images for Mode 3 – diagonal down left for a 8X8 block

Hence basis images are applied to other modes the same way and the following basis images are obtained as shown in Figures 4.5 and 4.6.

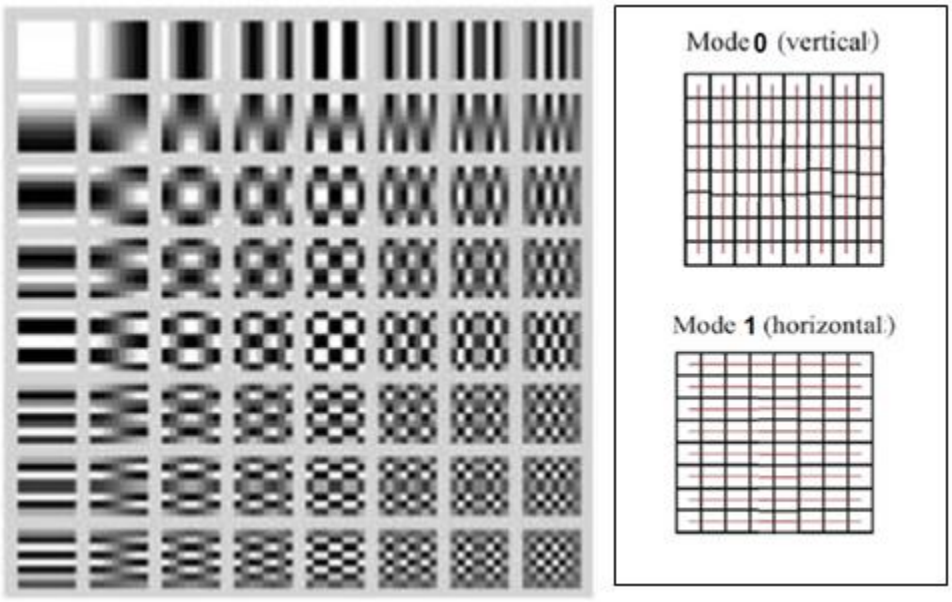


Figure 4.5 Mode 0 or 1 – Vertical or horizontal basis images for 8X8 block

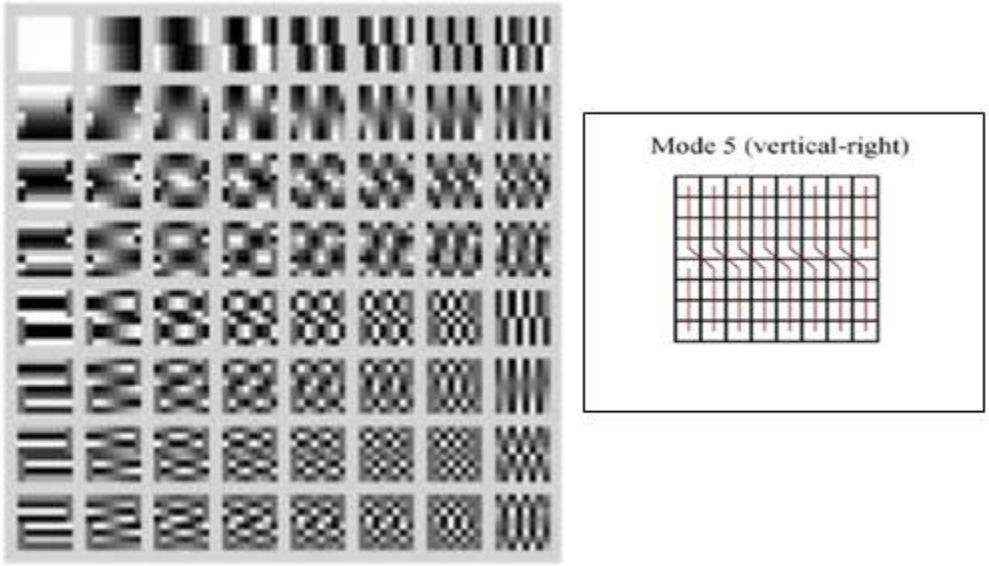


Figure 4.6 Mode 5 – Vertical right basis images for 8X8 block

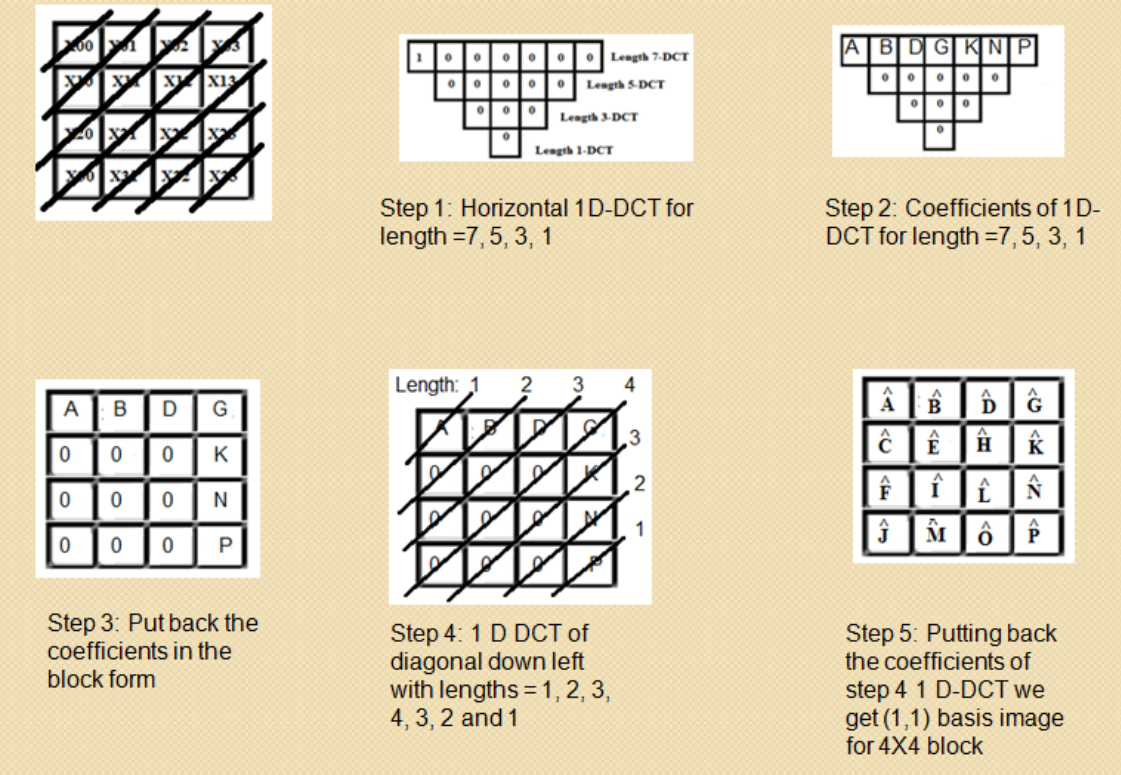


Figure 4.7 Step by step computation of the 1st basis image (1, 1) for 4X4 block of mode 3, diagonal down left.

4.4 Experimental Results

The objective of this thesis is to implement DDCT in place of Integer DCT in the transform block of the encoder in the H.264 reference software 18.0 [24]. Consider only the baseline profile of the H.264 implementation. A single intra prediction frame is considered for the DDCT results. Coding simulations are performed on various sets of test images and also on formats like QCIF and CIF. The coding performances are analyzed using different quality assessment metrics like MSE, PSNR and SSIM. Encoding time is also observed. These results are compared with respect to conventional DCT in existing H.264.

4.4.1 Quality Assessment Metrics

Lossless and lossy compressions use different methods to evaluate compression quality. Standard criteria like compression ratio, execution time, etc are used to evaluate the compression in lossless case, which is a simple task whereas in lossy compression, it is complex in the sense, it should evaluate both the type and amount of degradation induced in the reconstructed image [24] .The goal of image quality assessment is to accurately measure the difference between the original and reconstructed images, the result thus obtained is used to design optimal image codecs. The objective quality measure like PSNR, measures the difference between the individual image pixels of original and reconstructed images. The SSIM [36] is designed to improve on traditional metrics like PSNR and MSE (which have proved to be inconsistent with human visual perception) and is highly adapted for extracting structural information. The SSIM index is a full reference metric, in other words, the measure of image quality is based on an initial uncompressed or distortion free image as reference. The SSIM measurement system is shown in equation 4.4.

$$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}, \qquad c(\mathbf{x},\mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}, \qquad s(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3},$$

where $\mathbf{x}$ and $\mathbf{y}$ correspond to two different signals that would like to match, i.e. two different blocks in two separate images, μ_x , σ_x^2 , and σ_{xy} the mean of $\mathbf{x}$, the variance of $\mathbf{x}$, and the covariance of $\mathbf{x}$ and $\mathbf{y}$ respectively, while C_1, C_2, and C_3 are constants given by $C_1 = (K_1 L)^2$, $C_2 = (K_2 L)^2$, and $C_3 = C_2 / 2$. L is the dynamic range for the sample data, i.e. L=255 for 8 bit content and K_1<<1 and K_2<<1 are two scalar constants. Given the above measures the structural similarity can be computed as shown in equation 4.4

$$SSIM(\mathbf{x},\mathbf{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha} \cdot [c(\mathbf{x},\mathbf{y})]^{\beta} \cdot [s(\mathbf{x},\mathbf{y})]^{\gamma} \qquad (4.4)$$

where α , β and γ define the different importance given to each measure.

$$MSE = \frac{1}{M * N} \sum_{m=1}^{M} \sum_{n=1}^{N} [x(m,n) - y(m,n)]^2 \qquad (4.5)$$

$$PSNR = 10 \log_{10} \frac{L^2}{MSE} \qquad (4.6)$$

MSE and PSNR are calculated as shown in equation 4.5 and 4.6. Here the x is the original image and y is the reconstructed image. M and N are the width and height of the image. L is the maximum pixel value in NXM pixel image.

4.4.2 Encoder Configuration in JM 18.0

FramesToBeEncoded = 1 #Number of Frames to be coded

ProfileIDC = 66 # Profile IDC (66 = baseline, 77 = main, 88 = extended; FREXT Profiles: 100 = High, 110= High 10, 122= High 4:2:2, 244 = High 4:4:4, 44= CAVLC 4:4:4 Intra, 118 = Multiview High Profile, 128 = Stereo High Profile)

IntraProfile = 0 # Activate Intra Profile for FRExt (0: false, 1: true) # (e.g. ProfileIDC = 110, IntraProfile = 1 => High 10 Intra Profile)

Transform8X8Mode = 0 # (0: only 4X4 transform, 1: allow using 8X8 transform additionally, 2: only 8X8 transform

Transform16X16Mode =0 # (0: no 16X16 transform, 1: allow using 16X16 transform)

Input YUV file: foreman_qcif.yuv

Output H.264 bitstream: test.264

Output YUV file: test_rec.yuv

YUV format: YUV 4:2:0

Frames to be encoded: 1

Frequency used for encoded bitstream: 30.00 fps

DistortionSSIM = 1 # Compute SSIM distortion (0: disable/default, 1: enabled)

Tables 4.1 and 4.2 give the values of the Image metrics of an I-Frame sequence of Foreman which is a QCIF sequence. These results are taken with DDCT implemented in H.264 and the conventional DCT of the original H.264 and compared. The graphs are shown in figure 4.7 to 4.11.

Table 4.1 Image metrics for Foreman QCIF sequence in integer DCT implementation in H.264

Bit Rate (kbit/s/frame)	QP (I frame)	PSNR in dB	MSE	SSIM
5590.56	1	79.159	0.00079	1
5232.48	4	68.825	0.00852	1
3891.84	8	57.204	0.12378	0.9995
2168.16	16	48.86	0.84553	0.9965
1088.88	24	41.803	4.29364	0.9846
745.68	28	38.892	8.39157	0.9735
331.92	36	33.173	31.32035	0.9342
152.64	44	27.981	103.5018	0.8388
72	51	23.335	301.693	0.6703

Table 4.2 Image metrics for Foreman QCIF sequence in DDCT implementation in H.264

Bit Rate(kbit/s/frame)	PSNR in dB	MSE	SSIM
5486.96	90.357	0.00006	1
5201.96	82.436	0.00068	1
3882.53	69.689	0.0014	1
2264.63	60.147	0.00842	0.9996
1153.84	52.976	0.55385	0.9972
686.54	41.876	2.43788	0.9925
302.53	38.653	10.2537	0.9801
142.53	34.642	50.4376	0.9208
67	30.764	110.268	0.8674

Table 4.3 Encoding Time of I frame for Foreman QCIF sequence in DDCT and Int-DCT implementation in H.264

QP (I frame)	Encoding Time of Int-DCT (sec)	Encoding Time of DDCT (sec)
0	10.876	18.96
4	10.032	18.096
8	9.183	17.264
16	7.292	15.367
24	5.666	12.5437
28	4.968	11.0642
36	4.067	10.853
44	3.484	9.638
51	3.073	8.428

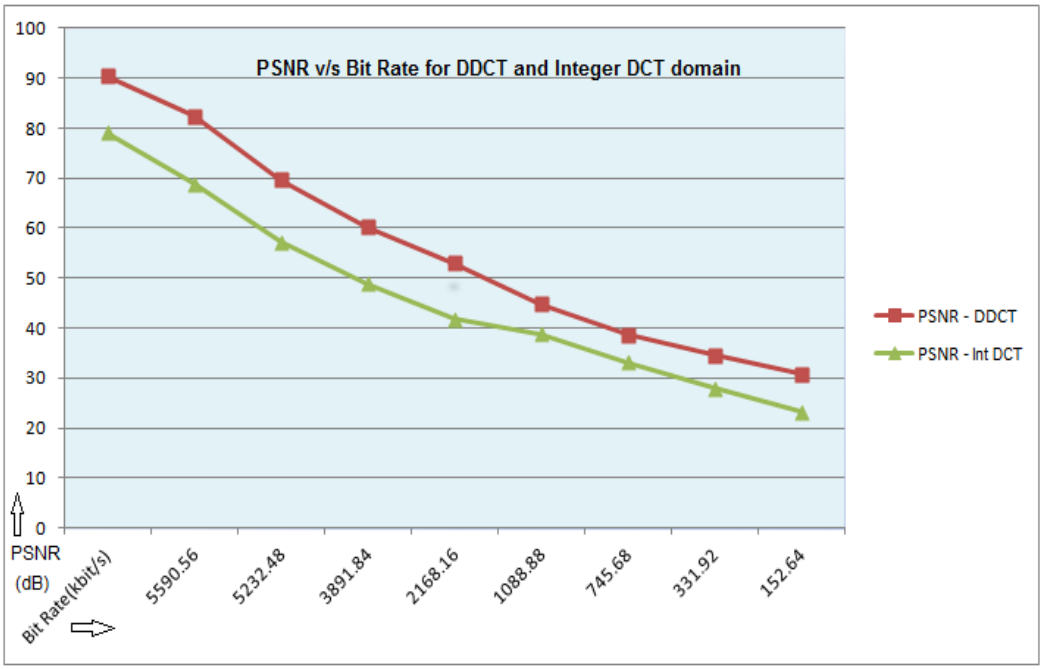


Figure 4.8 PSNR v/s bit rate for DDCT and integer DCT for foreman QCIF sequence

Figure 4.8 shows that the output obtained with directional DCT Transformation in H.264, gives a better PSNR value than the integer DCT. That is the reconstructed image has a better picture quality when compared to integer DCT. This also tells that the coding gain for DDCT is more than that for integer DCT. This graph is developed for almost similar bit rates of the frame measured in kbits/s/frame.

Figure 4.9 shows that the output obtained with directional DCT in H.264, has less MSE value than the integer DCT. This tells that the reconstructed image of DDCT has a better picture quality when compared to integer DCT. Hence PSNR and MSE graphs give us the detailed picture quality of both DDCT and integer DCT. We examine the SSIM value for both to determine which output picture quality is better in the next graph. If SSIM value is almost equal to 1, it means the output picture quality is almost equal to the input picture frame given at the encoder block.

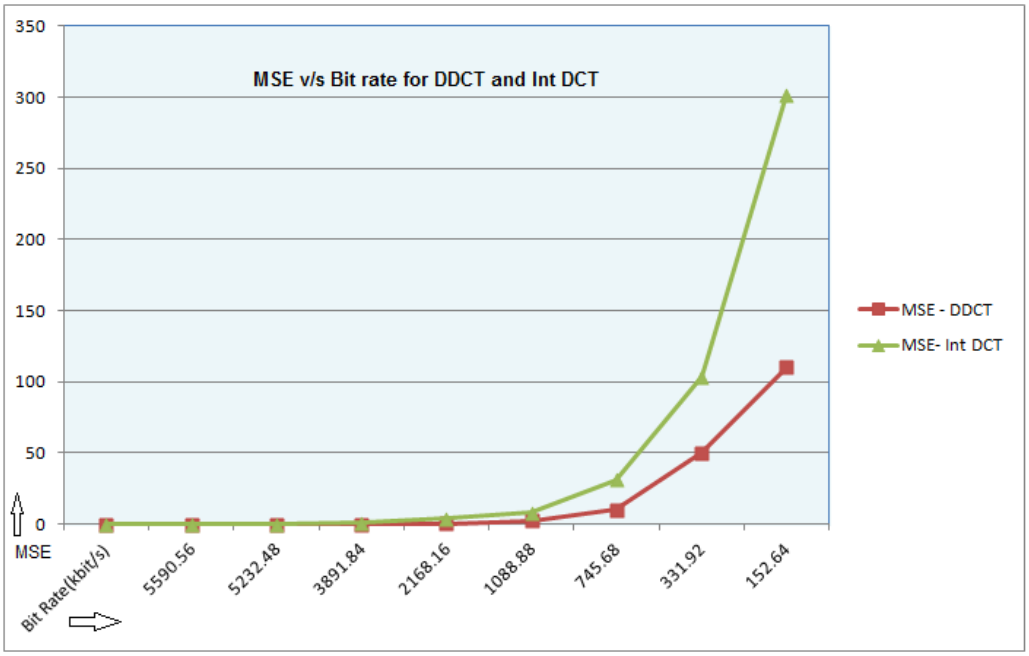


Figure 4.9 MSE v/s bit rate for DDCT and integer DCT for foreman QCIF sequence

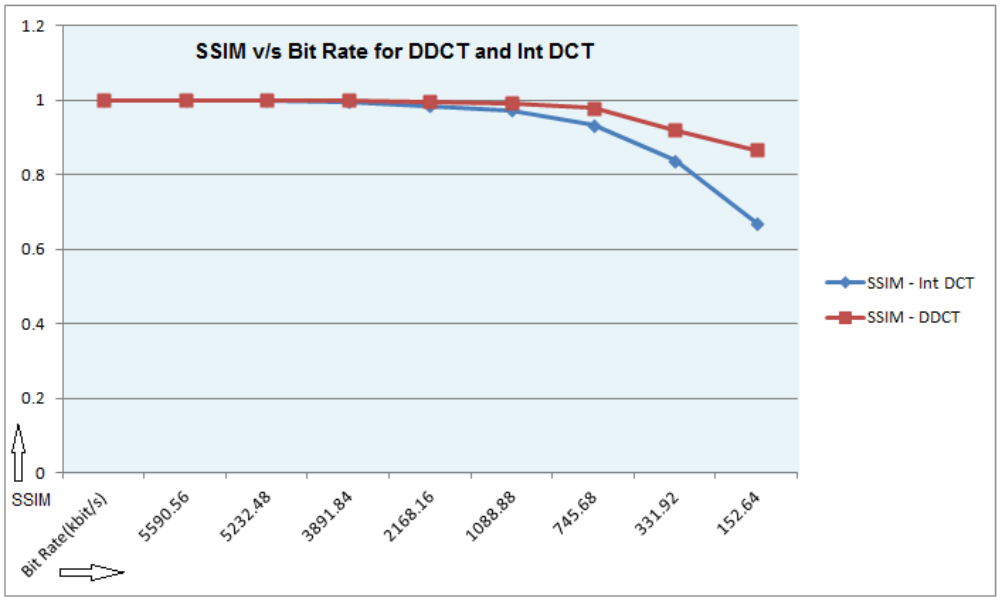


Figure 4.10 SSIM v/s bit rate for DDCT and integer DCT for foreman QCIF sequence

From the Figure 4.10, it is clear that the SSIM for the directional DCT Implementation in H.264 shows a better value that is maximum of 1 for a higher quantization parameter level and bit rate. Hence we can conclude that the reconstructed image of DDCT has better quality than that for integer DCT.

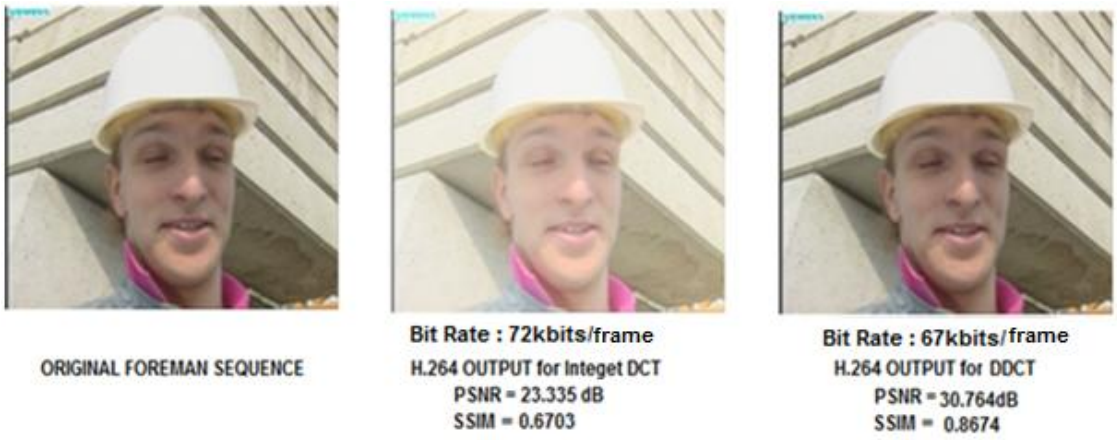


Figure 4.11 Test sequence used for simulation and their respective outputs

Figure 4.12 gives the encoding times taken for DDCT and integer DCT of H.264. The time taken to encode for DDCT is almost twice the time taken for integer DCT. This is because the Integer DCT checks only for 2 modes (horizontal and vertical) where as computational time for DDCT is more because it checks for 8 modes and the one which has the highest SSIM or PSNR is considered. Hence for computational time efficiency, DDCT implementation is not considered. This DDCT can be used for application for DVD storage and playing a movie but not for online streaming. This is because the encoding time taken for DDCT is slightly more when compared to integer DCT. Hence applications where the time taken is tolerable can use this to get better quality picture.

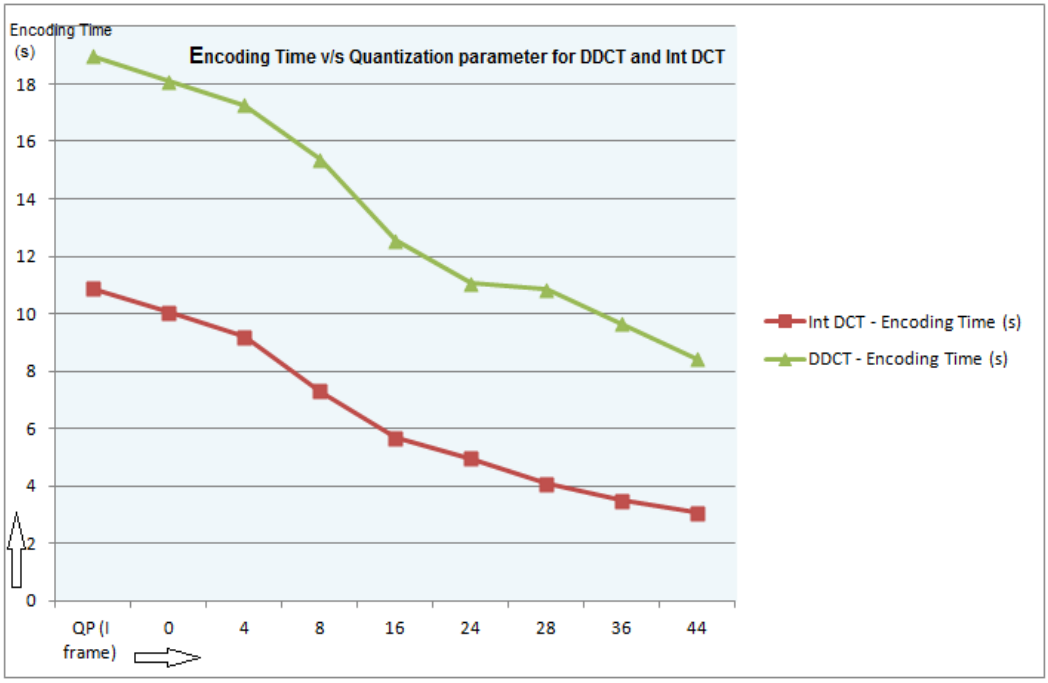


Figure 4.12 Encoding time v/s quantization parameter for DDCT and integer DCT for Foreman QCIF sequence

4.5 Properties of DDCT [31] [33]

The DDCT is a transform set to apply to the intra prediction errors in the video compression framework of AVC. The DDCT possesses the following properties: adaptivity, directionality, symmetry, and complexity (to the order of separable transforms).

- Adaptivity: Unlike AVC in which the same DCT-like transform is applied to the intra prediction errors for all intra prediction modes of the same block size (4x4, 8x8, or 16x16), DDCT assigns a different transform and scanning pattern to each intra prediction mode. These transforms and scanning patterns are designed taking into account the intra prediction direction.
- Directionality: Since the intra prediction mode is known, the DDCT is designed with the knowledge of the intra prediction direction. By first applying the transform along the prediction direction, DDCT has the potential to minimize the artifact around the object

boundaries.

- Symmetry: Although there are 22 DDCTs for 22 intra prediction modes (9 modes for 4x4, 9 modes for 8x8, and 4 modes for 16x16), these transforms can be derived, using simple operators such as rotation and/ or reflection, from only 7 different core modes.

4.6 Observation

From the above simulation results it can be concluded that, the encoder with DDCT algorithm takes significantly more encoding time when compared to the JM reference software [24] with integer DCT algorithm and in the meantime, does not sacrifice the quality of the image nor does it increase the bit-rate significantly. Hence, this approach using the directional modes in the transform domain gives a better picture quality when compared to integer DCT keeping in mind the more encoding time it would take to encode the sequence.

CHAPTER 5

CONCUSION AND FUTURE WORK

5.1 Conclusions

Looking at Tables 4.1 and 4.2, the conclusion is that the directional DCT has a better coding gain when compared to Conventianal DCT. Figures give a better understanding of that as the PSNR value for DDCT output is more when compared to Integer DCT. The SSIM graph as shown in Figure 4.9 tells us that the reconstructed image has a better quality when compared to conventional DCT. This can be proved again in Figure 4.10 for a Foreman frame of QCIF format, where the output obtained from DDCT has a better quality of image with respect to the output obtained from Integer DCT. The only drawback found in DDCT is that we should compromise on the Encoding time when compared to conventional DCT. Hence it can be used in applications like Bluray, DVD but not in broadcast communication.

5.2 Future work

Directional DCTs can be extended to many existing international standards (for image and video coding) that employ the conventional DCT. This thesis is for 1 I-frame, it can be extended for the entire video sequence which can give a better coding gain. This research can also be extended to other profiles of H.264 like high profile and extended profile. Only 8 modes as described in Figure 3.2 are implemented. This can be extended to more directional modes and compared with the integer DDCT for higher coding gain as specified in the results.

REFERENCES

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

[2] N. Ahmed, T. Natarajan, and K. R. Rao, “Discrete cosine transform,” IEEE Trans. Comput., vol. C-23, pp. 90-93, Jan. 1974.

[3] K. R. Rao and P. Yip, “Discrete cosine transform: Algorithms, advantages, applications,” Boca Raton FL: Academic Press, 1990.

[4] B. Zeng and J. Fu, “Directional discrete cosine transforms - A new framework for image coding”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 18, no. 3, pp. 305-313, Mar. 2008.

[5] B. Zeng and J. Fu, “A compensation techniques in directional DCT’s”, IEEE International Symposium on Circuits and Systems, pp. 521- 524, June-2007.

[6] E. Lallana and M. Uy, “The Information Age,” UNDP-APDIP, 2003.

[7] Open source article, “Digital Revolution,” Wikipedia Foundation, http://en.wikipedia.org/wiki/Digital_Revolution

[8] K. Sayood, "Introduction to data compression,” 3rd Edition, Morgan Kaufmann Publisher Inc., 2006.

[9] R. Schafer and T. Sikora, "Digital video coding standards and their role in video communications," Proceedings of the IEEE, Vol. 83, pp. 907-923, Jan. 1995.

[10] Information technology-generic coding of moving pictures and associated audio information: ISO/IEC 13818-2 (MPEG-2) Std.

[11] Advanced video coding for generic audiovisual services, ITU-T Rec. H.264 / ISO / IEC 14496-10, Nov. 2009.

[12] I. Ahmad et al, “Video transcoding: An overview of various techniques and research issues”, IEEE Trans. on Multimedia, vol. 7, pp. 793-804, Oct. 2005.

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

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

[15] T. Wiegand and G. J. Sullivan, “The H.264 video coding standard”, IEEE Signal Processing Magazine, vol. 24, pp. 148-153, March 2007.

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

[17] G. Sullivan, P. Topiwala and A. Luthra, “The H.264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions”, SPIE conference on Applications of Digital Image Processing XXVII, vol. 5558, pp. 53-74, Aug. 2004.

[18] K. R. Rao and P. C. Yip, “The transform and data compression handbook”, Boca Raton,FL: CRC press, 2001.

[19] T. Wiegand et al, “Overview of the H.264/AVC video coding standard”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, pp. 560-576, Jul. 2003.

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

[21] I. Richardson, “The H.264 advanced video compression standard”, Wiley, 2nd edition, 2010.

[22] Intra prediction modes in H.264. Website: http://www.vcodex.com/files/h264_intrapred.pdf

[23] F. Kamisli and J. S. Lim, “Video compression with 1-d directional transforms in

H.264/AVC", IEEE ICASSP, pp. 738-741, Mar. 2010.

[24] H.264/AVC reference software. Website: http://iphome.hhi.de/suehring/tml/download

[25] Intra coding with directional DCT and directional DWT, Document: JCTVC-B107_r1

[26] Directional Discrete Transform JCTV Website: http://wftp3.itu.int/av-arch/jctvc-site/2010_07_B_Geneva/JCTVC-B107.zip

[27] B.Chen, H.Wang and L.Cheng, "Fast directional discrete cosine transform for image compression", Opt. Eng. vol. 49, issue 2, article 020101, Feb. 2010.

[28] C. Deng et al, "Performance analysis, parameter selection and extensions to H.264/AVC FREXt for high resolution video coding", J. Vis. Commun. Image R., vol. 22 (In Press), Available on line, Feb. 2011.

[29] Z.Wang et al, "Image quality assessment: From error visibility to structural similarity", IEEE Trans. on Image Processing, vol. 13, no. 4, pp. 600-612, Apr. 2004.

[30] W.Zhao, J.Fan and A.Davari, "H.264-based wireless surveillance sensors in application to target identification and tracking", i-manager's Journal on Software Engineering, vol.4, no. 2, Oct. 2009.

[31] Website: http://web.eng.fiu.edu/fanj/pdf/J5_i-manager09h264_camera.pdf

[32] W.Zhao et al, "H.264-based architecture of digital surveillance network in application to computer visualization", i-manager's Journal on Software Engineering, vol.4, no. 4, Apr. 2010.

[33] Directional Discrete Cosine Transform theory Website: http://web.eng.fiu.edu/fanj/pdf/J8_i-mgr10architecture_camera.pdf

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

Aug. 2006.

[35] Website: http://iphome.hhi.de/wiegand/assets/pdfs/h264-AVC-Standard.pdf

[36] F. Kamisli and J. S. Lim, “Transforms for motion compensation residual”, IEEE ICASSP, pp.789-792, Apr. 2009.

[37] Z.Wang, E.P.Simoncelli and A.C.Bovik, “Multi-scale structural similarity for image quality assessment”, Thirty-Seventh Asilomar Conference on Signals, Systems and Computers, vol. 2, Nov. 2003.

[38] C.L.Chang and B.Girod, “Direction-adaptive partitioned block transform for image coding”, 15th IEEE International Conference on Image Processing, pp. 145-148, Oct. 2008.

[39] H.Xu, J.Xu and F.Xu, “Lifting-based directional DCT-like transform for image coding”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 17, issue 10, pp. 1325-1335, Oct. 2007.

[40] J.Xu, B.Zeng and F.Wu, “An overview of directional transforms in image coding”, Proceedings of 2010 IEEE International Symposium on Circuits and Systems, pp. 3036-3039, Aug. 2010.

[40] MPEG-1 basics Website: http://en.wikipedia.org/wiki/Mpeg-1

[41] MPEG-2 basics Website: http://en.wikipedia.org/wiki/Mpeg-2

[42] MPEG-4 basics Website: http://en.wikipedia.org/wiki/Mpeg-4

[43] H.261 basics Website: http://en.wikipedia.org/wiki/H.261

[44] H.262 basics Website: http://en.wikipedia.org/wiki/H.262

[45] H.263 basics Website: http://en.wikipedia.org/wiki/H.263

[46] DFT basics Website: http://en.wikipedia.org/wiki/Discrete_Fourier_transform

[47] K. R. Rao and J. J. Hwang, “Techniques and standards for image/video/audio coding”, Prentice hall, 1996.

BIOGRAPHICAL INFORMATION

Shreyanka Subbarayappa was born on April 25th, 1987 in Bangalore, Karnataka, India. She was the second daughter of Prof. H Subbarayappa and Anasuya Subbarayappa. She received her Bachelor's Degree in Tele-Communication Engineering from M S Ramaiah Institute of Technology, Bangalore in 2009. She taught Electronics at Mount Carmel College for 6 months where she loved the profession and the course, she decided to pursue her Masters Degree in Electrical Engineering at University of Texas at Arlington. During her studying period at Arlington she was more interested in Multimedia and joined the Multimedia Group at UTA in June 2009 under the guidance of Dr. K. R. Rao. She got an opportunity to intern at INTEL Corp. June 2011 to May 2012 at Chandler, Arizona in Embedded and Multimedia areas which broadened her understanding in these fields in depth. After her graduation, she intends to find a job in multimedia field where she can use her knowledge and experience practically.