DEVELOPMENT AND COMPARISON OF TRANSFORMS FOR PREDICTION RESIDUALS OF MARKOV-PROCESS-BASED INTRA PREDICTION

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

NİHAT TEKELİ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JANUARY 2014

Approval of the thesis:

DEVELOPMENT AND COMPARISON OF TRANSFORMS FOR PREDICTION RESIDUALS OF MARKOV-PROCESS-BASED INTRA PREDICTION

submitted by **NİHAT TEKELİ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. Gönül Turhan Sayan
Head of Department, **Electrical and Electronics Engineering** ________________

Assist. Prof. Dr. Fatih Kamışlı
Supervisor, **Electrical and Electronics Engineering Dept., METU** ________________

Examining Committee Members:

Prof. Dr. Uğur Halıcı
Electrical and Electronics Engineering Dept., METU ________________

Assist. Prof. Dr. Fatih Kamışlı
Electrical and Electronics Engineering Dept., METU ________________

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering Dept., METU ________________

Assoc. Prof. Dr. İlkay Ulusoy
Electrical and Electronics Engineering Dept., METU ________________

Halil Cüce, M.Sc.
Senior Researcher, İLTAREN BİLGEM TÜBİTAK ________________

Date: ________________

ABSTRACT

DEVELOPMENT AND COMPARISON OF TRANSFORMS FOR PREDICTION RESIDUALS OF MARKOV-PROCESS-BASED INTRA PREDICTION

TEKELİ, NİHAT
M.S., Department of Electrical and Electronics Engineering
Supervisor : Assist. Prof. Dr. Fatih Kamışlı

January 2014, 84 pages

Intra prediction is an important tool used in modern intra-frame coding. In intra prediction, a block of pixels are predicted from previously reconstructed neighbor pixels of the block by copying the previously reconstructed neighbor pixels of the block along an angular direction inside the block. The prediction residual block is then transformed with the conventional 2-D Discrete Cosine Transform (DCT). Recently, it has been shown that transforming the intra prediction residuals with an Asymmetric Discrete Sine Transform (ADST) along the prediction direction and the well-known DCT along the perpendicular direction improves the compression performance.

More recently, a recursive intra prediction algorithm, obtained by modeling image blocks with 2-D Markov processes, is proposed to improve the conventional copying-based intra prediction methods. In this thesis, we develop transforms for the intra prediction residuals obtained with these new recursive intra prediction algorithms. Using the 2-D Markov process correlations of each intra prediction mode, we obtain the correlations of the prediction residuals, and numerically compute Karhunen Loeve Transforms (KLT) for each intra prediction mode. We present compression results to compare the derived transforms with the conventional 2-D DCT and the hybrid ADST/DCT within the H.264 reference software.

Keywords: Markov Based Intra Prediction, Transform, Video Compression, Scanning Order, H.264

ÖZ

MARKOV-TABANLI RESİM İÇİ KESTİRİM ARTIKLARI İÇİN TRANSFORMLARIN GELİŞTİRİLMESİ VE KARŞILAŞTIRILMASI

TEKELİ, NİHAT
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Yrd. Doç. Dr. Fatih Kamışlı

Ocak 2014 , 84 sayfa

Modern resim içi imge kodlamada resim içi kestirim önemli bir araçtır. Resim içi kestirimde, bir piksel bloğu aynı bloktaki daha önce kodlanmış komşu piksellerden açısal bir yönde kopyalanarak kestirilir. Daha sonra, kestirim artıkları klasik Ayrık Kosinüs Transformu ile dönüştürülür. Son yıllarda, resim içi kestirim artıklarını kestirim yönünde Asimetrik Ayrık Sinüs Transformu ile kestirime dik yönde ise ayrık kosinüs dönüşümü ile kodlamanın sıkıştırma performansını artırdığı görülmüştür.

Çok yakın geçmişte, imge bloklarını 2 boyutlu Markov tabanlı süreçle modelleyen bir özyinelemeli resim içi kestirim algoritması klasik kopyalama tabanlı resim içi kestirim yerine sıkıştırma performansını artırmak amacıyla önerilmiştir. Bu tezde, yeni özyinelemeli resim içi kestirim yöntemi kullanıldığı zaman oluşan resim içi kestirim artıklarını dönüştürmek için transformlar geliştirilmiştir. 2 boyutlu Markov süreç korelasyonlarını her bir resim içi kestirim modu için kullanarak, resim içi kestirim artıkları arasındaki korelasyon elde edilmiştir ve her bir mod için nümerik olarak Karhunen Loeve Transformları hesaplanmıştır. Yeni türetilmiş transformlar ile 2 boyutlu Ayrık Kosinüs Transformu ve hibrid Asimmetrik Ayrık Sinüs Transform/Ayrık Kosinüs Transform performans karşılaştırılmaları sonuç olarak sunulmaktadır.

Anahtar Kelimeler: Markov Tabanlı Resim içi Kestirim, Transform, Video Sıkıştırma, Tarama Sırası, H.264

To My Mother

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere thanks to my supervisor Assist. Prof. Dr. Fatih KAMIŞLI for his supervision and guidance throughout this study.

I would like to thank my director Halil CÜCE for his understanding and his encouragement throughout my thesis study.

I would like to thank Tayfun AYTAÇ for his time on proofreading of my work and for his valuable comments.

Lastly, I am indebted to my mother for her continuous love, encouragement, and support during my academic studies.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

1D	One dimensional
2D	Two dimensional
ADST	Asymmetric discrete sine transform
BD	Bjontegaard-Delta
BR	Bitrate
CABAC	Context adaptive binary arithmetic coding
CAVLC	Context adaptive variable length coding
CD	Compact disk
C#	C sharp (a software language)
DCT	Discrete cosine transform
DFT	Discrete fourier transform
DST	Discrete sine transform
DVB	Digital video broadcast
DVD	Digital versatile disc
DWP	Distance based prediction
DWT	Discrete wavelet transform
fps	Frame per second
GOP	Group of pictures
HD	High definition
HDTV	High definition television
HEVC	High efficiency video coding
ISO	International Organization for Standardization
ITU	International Telecommunication Union
JVT	Joint Video Team
Kbit	Kilobit
KLT	Karhunen Loeve transform
Mbit	Megabit
MPEG	Moving Pictures Experts Group
NAL	Network abstraction layer
PSNR	Peak signal to noise ratio
QP	Quantization parameter
RDO	Rate distortion optimization

SNR	Signal to noise ratio
VCD	Video compact disk
VCEG	Video Coding Experts Group
VCL	Video coding layer
VLC	Variable length coding

CHAPTER 1

INTRODUCTION

Imaging and video playback applications have started to take more place in our lives with the advancement in technology. As a result of these, mobile phones, digital cameras, pictures, and videos have become a routine of daily life. Video playback websites, mobile phones, televisions, computers, DVD players, surveillance systems, and digital cameras are some of the examples of video applications in our daily life.

Raw video footage requires a very large storage capacity when it is not compressed. If video transfer is done via a data line, uncompressed video data would require very large transmission rates and would create a huge traffic. Therefore, almost all video applications compress video data or supports compressed video formats. Nowadays, a standard movie DVD, sold with high-quality image data, is compressed 100 times and still contains high quality image data. The videos, received from internet and from satellite receivers, are transmitted after compression and decoded on the user site.

Many international video coding standards have been developed to increase the efficiency of video compression by recognized international organizations. The most popular ones of those organizations are International Telecommunication Union (ITU) [6], the International Organization for Standardization (ISO) [5] and Moving Pictures Experts Group (MPEG) [7]. Video compression standards evolved in time. First video compression standard MPEG-1 is mostly used in CD-ROMs, VCD players, and camrecorders in 1980s. MPEG-2 added support for interlacing and improved compression efficiency and mostly used in DVDs and video broadcasting. H.263 was a popular video conferencing standard in previous years. MPEG-4 and H.264 are most commonly used standards today [18]. HEVC (High Efficiency Video Coding) is the most recent video coding standard and currently under development [46]. A brief history of popular video compression standards are given in Figure 1.1.

H.264 is one of the most widely used high compression ratio video coding standard developed by the ITU-T Video Coding Experts Group (VCEG). H.264 offers good video quality at lower bit rates compared to the previous video standards [4].

The general idea of video compression is to reduce data size as much as possible while

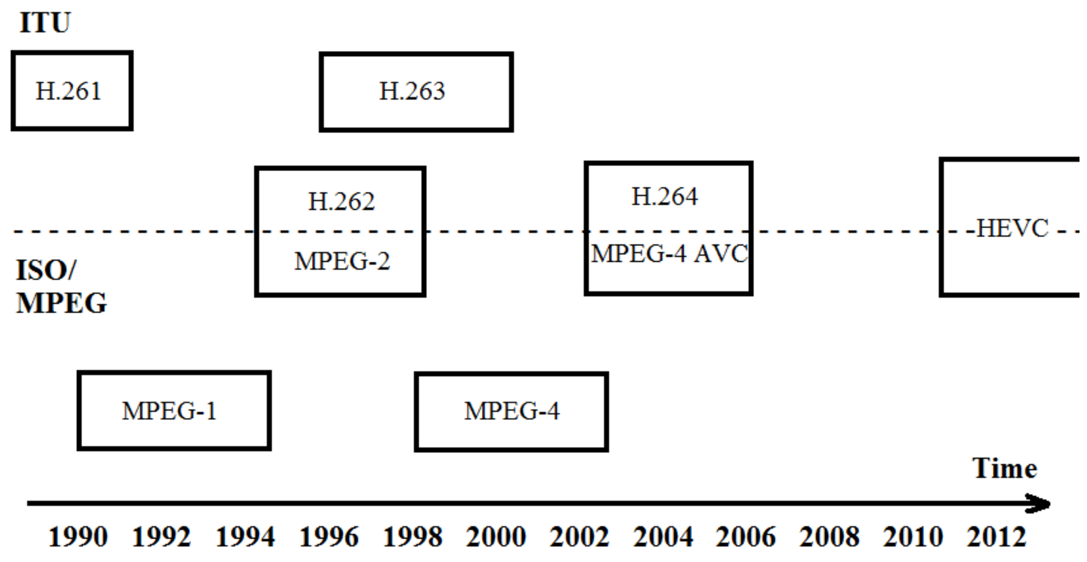


Figure 1.1: Brief history of popular video compression standards

maintaining the image quality at a reasonable level. The general structure of video compression and decompression is given in Figure 1.2. The video that is planned to be compressed is first fed into an encoder block. The encoder block processes video and outputs the compressed data that represents video in a smaller size. The size of compressed data may be two, ten, fifty or more times smaller than the size of the original video according to the chosen encoding algorithm. The compressed data is transmitted or stored via a medium (such as harddrive, internet, and satellite). The another purpose in video compression is to preserve the valuable resources of medium in order to reduce cost, store more data, or serve to much more people. Therefore, the video compression technology is very critical to internet, broadcasting, and multimedia industry.

There are two types of video compression which are lossless and lossy compression. Lossless video compression, as its name suggets, is the compression that no video data is lost. The original video and the corresponding decompressed video are exactly the same. Compressed data exactly denotes the original video within a smaller space; therefore offers the best quality image possible. The disadvantage of lossless compression is the larger data size compared to lossy compression. In lossy compression, a video is compressed with data loss. Generally, the lost data are the unimportant or unperceivable parts of the image. Due to this flexibility, lossy compression offers higher compression ratio in exchange of video quality. The lossy compression is optimized according the perception capabilities of human eyes. When the multimedia industry is investigated, it is seen that lossy video compression is much more widely used than lossless video compression. Lossy video compression is very common on internet and

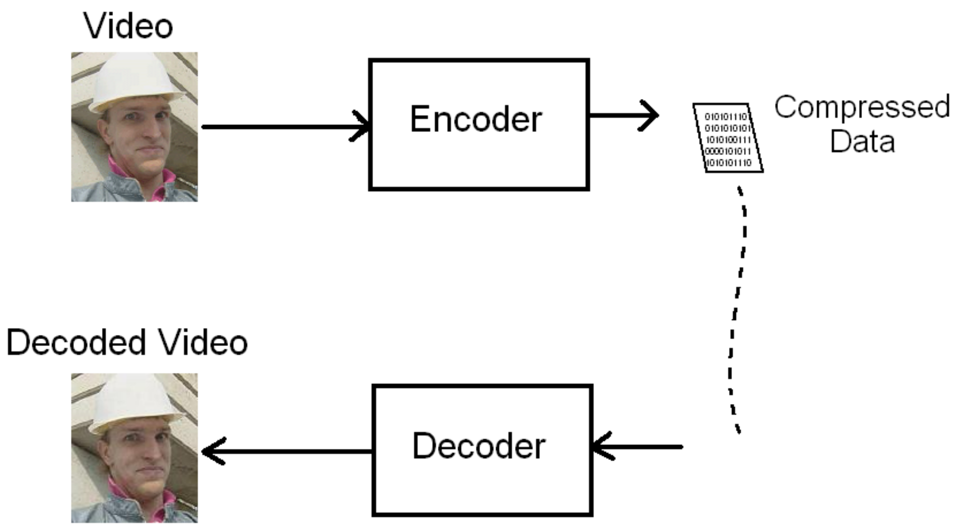


Figure 1.2: Basic idea of video compression

other multimedia areas; however, lossless video compression is limited to the video editing and biomedical area where data loss in unacceptable.

Video coding employs different algorithms to compress video. Some algorithms only work on single frames; others work on multiple frames and use the correlation among them. In video coding, pixel blocks can be predicted from neighbor pixels or they can be reconstructed from similar block either in the same or neighbor frame. Multiple algorithms can be switched according to their performance. Image blocks can be transformed and quantized into leves to reduce data size. Frames and pixel blocks are generally buffered and used for the use of next frames. In addition, lossless file compression methods can be employed to further reduce the size of the video. Different video coding standards make use of different algorithms and obtain different compression ratios and image quality. Also, computation requirements of different video coding methods differ according to the chosen algorithms.

The overall objective of video compression methods is to keep a reasonable level of image quality by using as little information as possible for representing video. For this purpose, video encoders compress by ignoring repetitive, unnecessary, and unimportant information. Among this information, there exists low amplitude frequency information, the pixels which have the same values with their neighbors and the images with similar characteristics in the previous frame. If prediction method uses information from previously coded frames, it is called inter prediction. If the prediction method uses information present only in the current frame, it is called intra prediction.

Intra prediction methods attempt to predict currently processed block by using previously predicted blocks in only current frame. If there exists a previously predicted similar block in the same frame, it is possible to predict current block by compensating block motion.

1.1 Scope and Outline of the Thesis

In this thesis, new intra coding methods based on Markov based intra prediction together with the new generation and widely used transforms are proposed and their performance are compared.

The rest of the thesis is organized as follows.

Chapter 2 gives an overview of the H.264 video coding standard.

Chapter 3 explains intra predictions methods and transforms used in the literature. First, it introduces intra prediction algorithm used in H.264, then it gives some intra prediction methods and some transforms used in the literature.

Chapter 4 explains the implementation details of proposed intra coding methods.

Chapter 5 gives the results and comparisons of the implemented intra coding methods.

Chapter 6 includes the conclusion.

CHAPTER 2

OVERVIEW OF H.264

2.1 History

ITU Video Coding Experts Group (VCEG) and ISO Moving Picture Experts Group (MPEG) played an important role in the standardization of many popular video coding standards. The video coding standards H.261, H.263, and H.26L are developed by ITU. The video coding standards MPEG-1, MPEG-2, and MPEG-4 are developed by ISO. In 2001, these two organizations decided to merge their video teams and a joint team is formed named as Joint Video Team (JVT) [49]

JVT started working on the new video coding standard based on ITU's video coding standard H.26L [16]. JVT group made several meetings until March 2003 [49] The standardization of H.264 is completed in March 2003 [54]. The standard is called H.264, H.264/AVC or H.264/MPEG-4 Part 10 [11]. In this thesis, we call the standard by its shortest name H.264.

Standardization includes only bit stream and syntax restrictions as well as a standard decoder but not the implementation details of encoder. Therefore, H.264 standard allows manufacturers to design their optimal encoder as long as they respect syntax and bit stream restrictions [54].

2.2 Expectations and Features of the Standard

To meet the expectations from the video compression community; H.264 standard supplied the following features.

- Variable block size: H.264 offers 4x4, 4x8, 8x4, 8x8, 8x16, 16x8, 16x16 block sizes [13].
- Unlimited length motion vector: Block can move within a frame freely without any vector size restriction [13].

- Network friendly interface: H.264 standard with its network abstraction layer offers easy transmission of network packets [53].

- Error resilience: H.264 contains tools for robust transmission in case of packet losses and bit errors. H.264 can also encode and decode redundant frames in case of data loss [13].

- Quarter motion vector accuracy: Previous standards supported upto half motion sample vector accuracy, however, H.264 supports up to quarter motion sample vector accuracy [54].

- Multiple reference picture motion compensation: Different than MPEG-2, H.264 supports choosing reference frame among multiple previous frames. Therefore, it offers the high compression capability for periodically repeating frames [42].

- Deblocking filter: The artifacts at the block edges are eliminated using a deblocking filter in H.264 standard. Deblocking filter simply filters the transitions between blocks and presents subjectively higher quality image [22].

- Weighted prediction: Previous standards offered only equal weights for the reference signals. However, some popular effects in movies such as fade require different weights for reference frames. H.264 standard offers weighted prediction capability [44].

- Scalar quantization: H.264 offers scalar quantization. Step sizes increase by 12.5% at each quantization step [44].

- Improved coding and compression: H.264 offers higher quality image at the same bit rate compared to the previous standards. It achieves this performance by using improved inter/intra prediction, context adaptive arithmetic coding and adaptive slicing [13].

- Support for different profiles: H.264 offers different profiles and levels for different purposes [13]. For example; mobile devices can use base stream and satellite receivers can use main profile and give higher quality image.

2.3 Architecture

H.264 basically consists of two different layers. Those two layers are Video Coding Layer (VCL) and Network Abstraction Layer (NAL). VCL layer handles coding and compression of video data information. VCL's main purpose is to code video data as efficient as possible. NAL is responsible for transmitting coded video data. NAL has the knowledge of the VCL format and it partitions coded data and adds header information to make it suitable for transmission or storage. NAL can abstract data for different mediums and formats [49]. Due to the two layer structure of H.264, it presents a flexible and customizable framework for all current and next generation transmission mediums and storage media [54]. The two layer structure of H.264 is given in Figure 2.1

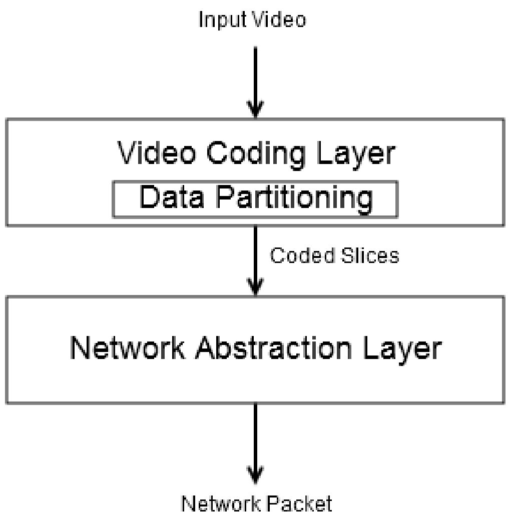


Figure 2.1: Two layer structure of H.264

2.3.1 Encoder Structure

Although H.264 standard does not define an encoder structure, an encoder structure can be implied from the standard as a block diagram. An example block diagram of H.264 encoder is given in Figure 2.2 [13]. In the encoder, input image is first divided in macroblocks of size 16x16. Luma and chroma parts of a macroblock are predicted using inter (temporal) or intra (spatial) prediction. The decision to choose intra or inter mode is given in encoder itself with included decoder and previous frame buffer [54]. Intra coding uses several prediction modes to predict current block using neighbor pixels and reduces spatial redundancy in the image [44].

2.3.2 Inter Coding

Inter prediction exploits the temporal redundancy in video. Since a video generally consists of fast subsequent frames with little change (such as movies, TV broadcasts), temporally adjacent frames have much in common. Inter prediction uses buffered reference frames to predict current block. Reference frames are the previously predicted frames (either past or future frames) buffered in encoder. Inter coding uses motion estimation and compensation to predict current block from reference frames [22]. A basic motion compensation scheme is given in Figure 2.3

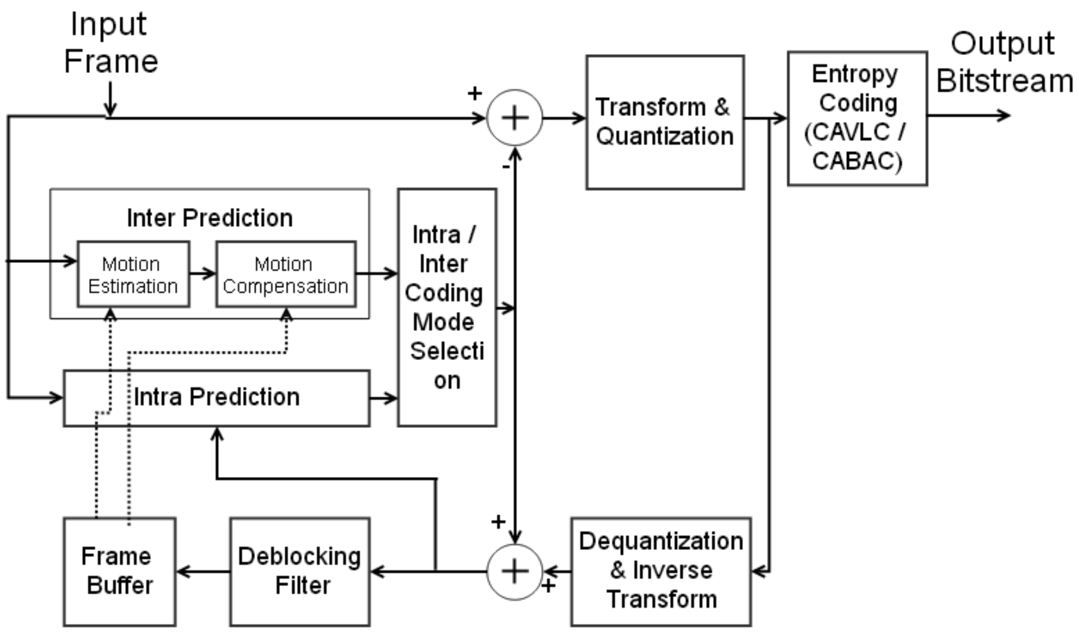


Figure 2.2: An example block diagram of H.264 encoder [13]

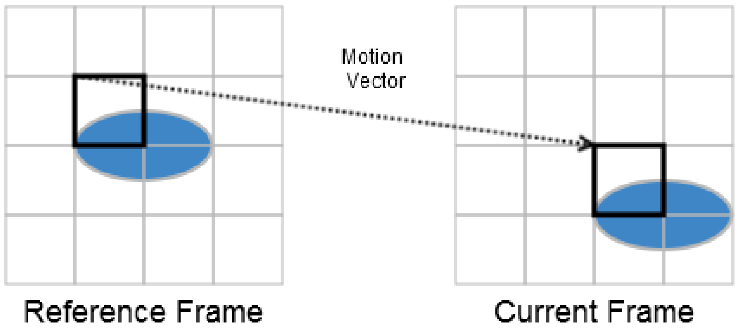


Figure 2.3: H.264 motion compensation

In inter coding, there are I frames, B frames, and P frames. I frames are the individual frames that are intra coded. P frames are the frames predicted using previous I frame in forward direction. B frames are the frames between I frames and P frames. B frames are the frames predicted using both I and P frames both in forward and reverse direction temporally. Since B frames have more source to originate prediction, it offers higher compression ratio [14]. A diagram of I, B, and P frames are given in Figure 2.4.

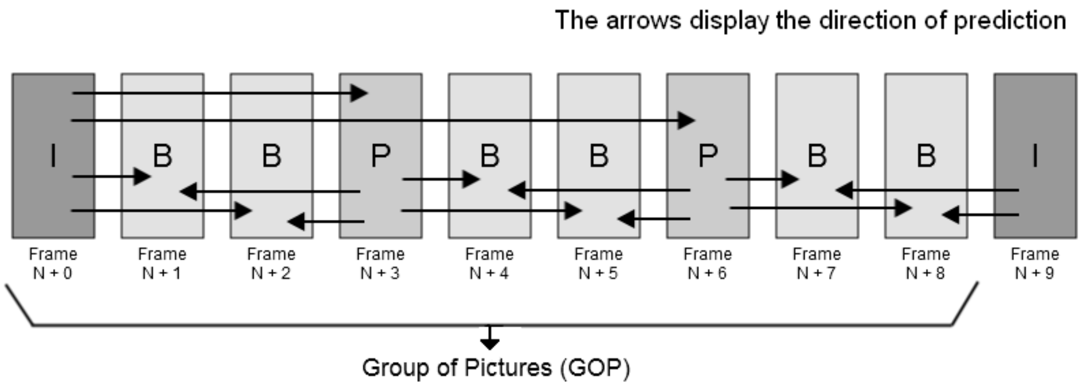


Figure 2.4: I, B, and P frames in a group of pictures in H.264 standard

In H.264, a macroblock can be split into multiple smaller blocks. Supported block sizes are 4x4, 4x8, 8x4, 8x8, 8x16, 16x8, and 16x16. When the block size is large, it is possible to represent a block with fewer coefficients. However, if the block is very detailed, it might be better to use smaller block sizes. Therefore, the selection of block size depends on the incoming video characteristics [44]. The possible block sizes are constructed according to incoming block size. The possible block sizes for H.264 are shown in Figure 2.5

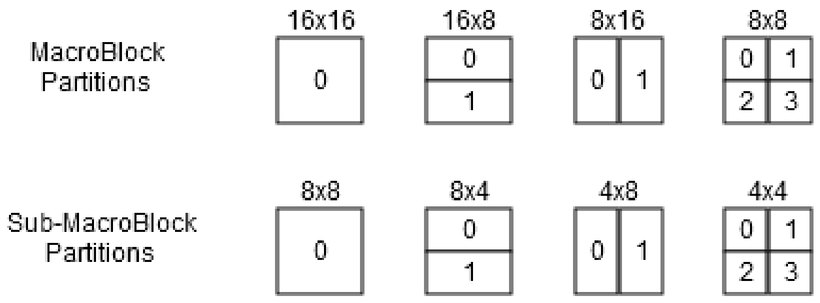


Figure 2.5: Variable block sizes in H.264

2.3.3 Integer Transform and Quantization

In H.264, block residuals remaining from prediction are converted into a different domain (frequency domain) to gather information into a few coefficients. Two dimensional discrete cosine transform is the most widely used transform for image coding due to its proximity to KLT transform. DCT allows higher efficiency coding [22]. Brahimi and Bouguezel propose an algorithm that merges quantization and transform together which results in fast DCT integer transform called integer transform [9]. In H.264 a similar approach is adapted. Integer transform for DCT approach that reduces the computation cost is chosen in the implementation of H.264 [32]. Integer transform also guarantees that its inverse transformation will give the same coefficients, on the contrary to DCT does not always give the same coefficients due to its fractional nature and limited precision in computer hardware [32]. 4x4 integer transform matrix and 8x8 integer transform matrix are given in Equation 2.1 and Equation 2.2, respectively.

$$T_{4x4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & 1 \end{bmatrix} \tag{2.1}$$

Integer transform matrix for 4x4 luma blocks (4x4 integer transform matrix just includes addition, subtraction and multiplication by 2)

$$T_{8x8} = \frac{1}{8} \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 & 8 & 8 \\ 12 & 10 & 6 & 3 & -3 & -6 & -10 & -12 \\ 8 & 4 & -4 & -8 & -8 & -4 & 4 & 8 \\ 10 & -3 & -12 & -6 & 6 & 12 & 3 & -10 \\ 8 & -8 & -8 & 8 & 8 & -8 & -8 & 8 \\ 6 & -12 & 3 & 10 & -10 & -3 & 12 & -6 \\ 4 & -8 & 8 & -4 & -4 & 8 & -8 & 4 \\ 3 & -6 & 10 & -12 & 12 & -10 & 6 & -3 \end{bmatrix} \tag{2.2}$$

Integer transform matrix for 8x8 luma blocks (Multiplication by $\frac{1}{8}$ in the transform matrix is just right shift by 3 in hardware implementation)

Although H.264 merges quantization and transformation, transformation coefficients can be extracted from the implementation and can be given as table. There are 52 quantization levels with an increase of about 12%. This increase corresponds to a factor of two at each 6 step. With this variety of quantization steps, it addresses to a wide range of values and gives opportunity to compress them efficiently [32]. Quantization levels and inverse quantization levels are given in Table 2.1 and in Table 2.2, respectively. The main principle in quantization is to multiply by a number and shift right, therefore, this corresponds to division by a fractional number. For inverse quantization, multiplying the number with corresponding inverse level is sufficient. Quantization level in H.264 is denoted by the quantization parameter (QP) which can take 52 different values [49]. For other quantization levels, the modulo 6 of quantization parameter is taken and a scale factor is added.

Table2.1: Scaled integer quantization coefficients

QP	Quantization Coefficients
0	410
1	365
2	325
3	290
4	258
5	230

Table2.2: Scaled integer dequantization coefficients

QP	Reverse Quantization Coefficients
0	40
1	45
2	50
3	57
4	63
5	71

2.3.4 Scan Order

Since a kind of DCT transformation is used in H.264, coefficients are generally higher at the left-top corner of the block and lower in the right-bottom corner of the block. Entropy coding algorithms require coefficients to be ordered in descending fashion so that zero coefficients are gathered at the end. Therefore, a zig-zag scanning pattern is mostly used in H.264 standard [44]. Scanning pattern for 4x4 blocks in frame mode and 8x8 blocks in frame mode are given in Figure 2.6 and in Figure 2.7, respectively.

2.3.5 Entropy Coding

There are two types of entropy encoding in H.264 standard. First one is Context Adaptive Variable Length Coding (CAVLC) which is used in simpler applications. Second one is Context Adaptive Binary Arithmetic Coding (CABAC) which provides higher compression ratio compared to CAVLC [22].

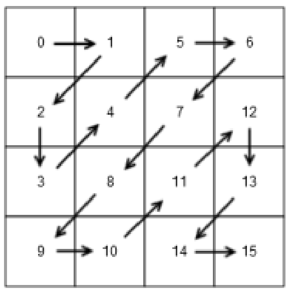


Figure 2.6: Default zig-zag pattern for 4x4 block in frame mode

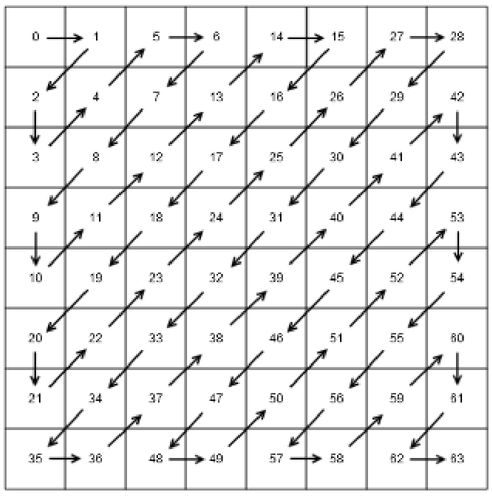


Figure 2.7: Default zig-zag pattern for 8x8 block in frame mode

2.3.5.1 Context Adaptive Variable Length Coding

CAVLC employs variable length coding to eliminates inter symbol redundancies. It uses previously transmitted coding symbols [33]. In this entropy coding mode, run-length coding is used to encode bit strings and eliminate zeros. The remaining non-zero coefficients are coded in inversed zig-zag order. By using prediction, appropriate coding table is selected and implemented on the bit string [36]. Due to the imposed limit of 1 bit/symbol in variable length coding, high probability symbols cannot be efficiently coded [33]. Thanks to the low complexity of CAVLC algorithm, CAVLC is exploited in low complexity applications.

2.3.5.2 Context Adaptive Binary Arithmetic Coding

The coefficients to be encoded are first converted into binary string. Then, binary string is sent to context modeling stage of CABAC where the probability model of the given bit string is determined. Using the chosen context model, the binary string is encoded using arithmetic coding. Then, the whole binary string is converted into a binary coded fractional number which is in the region [0,1]. This binary coded number is the encoded string which represents transform coefficients. Since the whole binary string is encoded at the same time, it offers higher compression compared to CAVLC, but introduces more complexity [33].

2.3.6 Decoder Structure

Decoder implements the reverse process of the encoder. Decoder is exactly defined in H.264 standard. First, entropy decoding is employed on incoming bit stream. Then, dequantization and inverse transform are applied on the decoded string. Coefficients are obtained at this stage and the prediction mode is determined (either intra or inter). According to chosen prediction mode, required reverse intra/inter coding processes are applied on the coefficients and a frame is reconstructed. Deblocking is applied on resulting frame and an approximation of original frame is obtained [44]. Before the output is given, frames are reordered in the correct order [49]. The block diagram of H.264 decoder is given in Figure 2.8

2.3.7 Deblocking Filter

Since video is processed in encoder in the form of macroblocks and blocks, artifacts occur at the block edges. To eliminate these artifacts, deblocking filters are used in H.264 standard. Deblocking filter is employed in H.264 as either loop filter or post filter. The performance of loop filter mostly exceeds post filter and mostly used in H.264 [30]. The difference in the output image with and without deblocking filter can be seen in Figure 2.9. Deblocking filter both improves objective and subjective quality of the decoded frames [22]. However the increase in subjective quality is very high as seen in Figure 2.9.

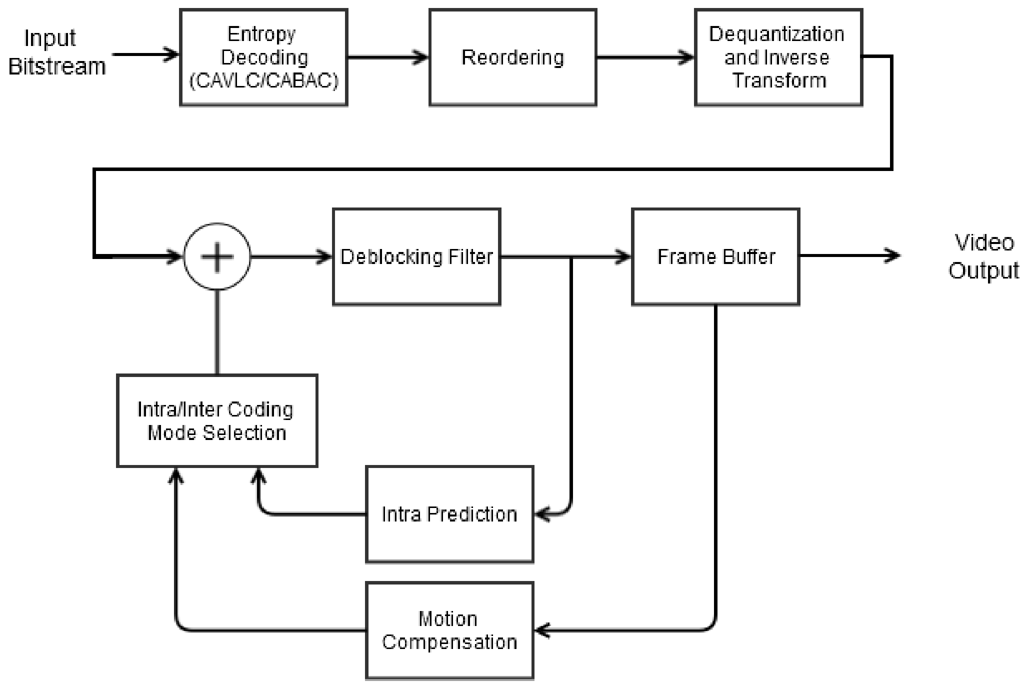


Figure 2.8: H.264 decoder block diagram

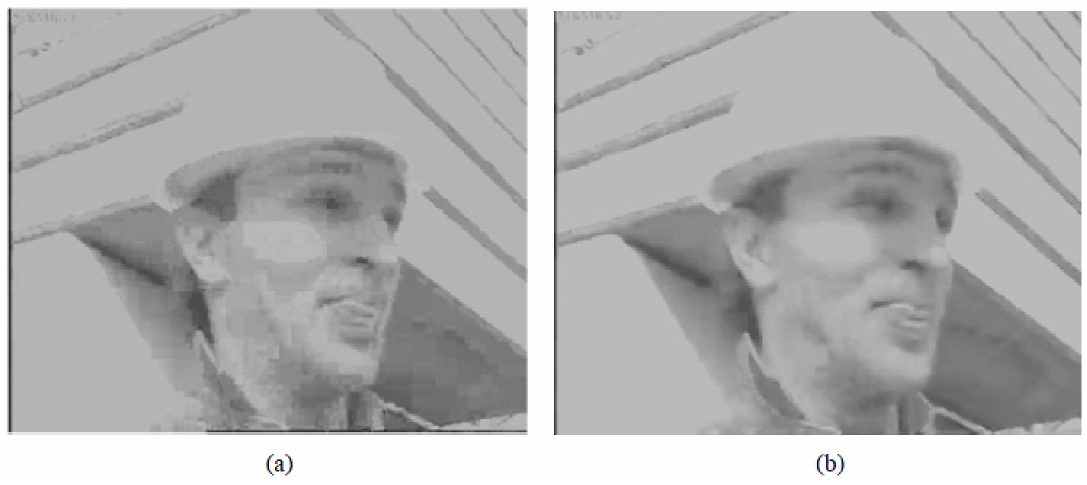


Figure 2.9: Deblocking filter effect a) Without deblocking filter b) With in-loop deblocking filter [54]

2.4 Profiles and Levels

Considering that all features of H.264 may not be needed by all users, profiles and levels have been created. Profiles and levels specify features and limits that must be supported by conforming decoders to a specific profile and levels [49]. Encoders are responsible only for generating conforming bit stream. A profile includes a collection of algorithms and coding tools that can be used to output corresponding bit stream. On the other hand, a level defines limits and requirements on some parameters of the bit stream. There are around 15 levels in H.264. Levels specify picture size, processing rate, buffer size, and bit rate [54]. In Table 2.3, levels and corresponding maximum picture sizes can be seen. There are profiles in H.264 standard which are Baseline Profile, Main Profile, and Extended Profile.

Table2.3: Levels in H.264 standard and corresponding popular image sizes and frame per second (This table does not give the exact supported resolutions, but given resolution and frame per second are the most popular and highest values in that level) [4] [49]

Level	Resolution @ fps
1	176x144@15.0
1.1	176x144@30.0
1.2	352x288@15
1.3	352x288@30
2	352x288@30
2.1	352x480@30
2.2	720x480@15
3	720x480@30
3.1	1280x720@30
3.2	1280x720@60
4	1920x1080@30
4.1	1920x1080@30
4.2	352x288@15
5	1920x1080@60
5.1	4096x2048@30

Baseline Profile is the lower end of the H.264 standard. This profile is mainly intended for video conferencing and video telephony applications. Main features of this profile are: I and P slice support, deblocking filter, zig-zag scan, quarter sample motion vector, CAVLC, 4:2:0 chroma format, and redundant slices [49].

Main profile is mainly designed for video broadcasting via satellite/internet, video storage, and entertainment consumer market. Main profile does not entirely cover baseline profile and lacks some features of baseline profile such as error resilience. Some of the features supported by main profile are I, B, and P slices, CAVLC, CABAC, and weighted prediction [44].

Extended profile is designed for streaming applications. It includes special features to increase robustness of video transmission [22]. Extended profile fully covers all features of baseline profile. Extended profile includes the following features: advanced error resilience over baseline profile, interlacing, I, B, P, SI, and SP slices, data partioning, and weighted prediction [44]

In Figure 2.4, Profiles and features supported by that profile are given.

Table2.4: Feature and profile matching table [16][4]

Feature	Baseline Profile	Main Profile	Extended Profile
I and P Slices	X	X	X
B Slices		X	X
SI and SP Slices			X
CAVLC	X	X	X
CABAC Slices		X	
Flexible Macroblock Ordering	X		X
Interlaced Coding		X	X
Error Resilience	X		X
Enhanced Error Resilience			X

2.5 Comparison with other standards

H.264 started to achieve significant performance improvements over the previous video coding standards. H.264 includes many new features to improve coding gain. Kamaci demostrated in [23] that H.264 is 50% superior to MPEG-2, 47% to H.263 baseline and 24% to H.263 high profile.

The novel features of H.264 and some drawbacks of the standard compared to the previous standards can be listed as following;

- With its two layer implementation, H.264 coding standard can be used in new technologies.
- In-loop deblocking filter presents higher subjective and objective image quality at lower bit rates.
- Advanced error resilience and redundant pictures are included to improve robustness.
- Unlimited motion vector movement with quarter accuracy is superior to previous standards.
- Variable block sizes allow higher compression ratio.
- Adaptive Binary Arithmetic Coding highly improves compression ratio.

- Weighted prediction with multiple reference frames improves coding.
- Profiles and levels make H.264 address different sectors.
- H.264 increases implementation complexity compared to previous standards.

In Table 2.5, comparison of H.264 to several other standards are given.

Kumar gives an objective comparison of compression efficiency in [13] which can be seen in Figure 2.10.

Wedi presents an initial subjective comparison of H.264 standard in [48].Oelbaum made a subjective comparison of image quality of H.264 standard. The results of this research can be seen in Figure 2.11 [47].

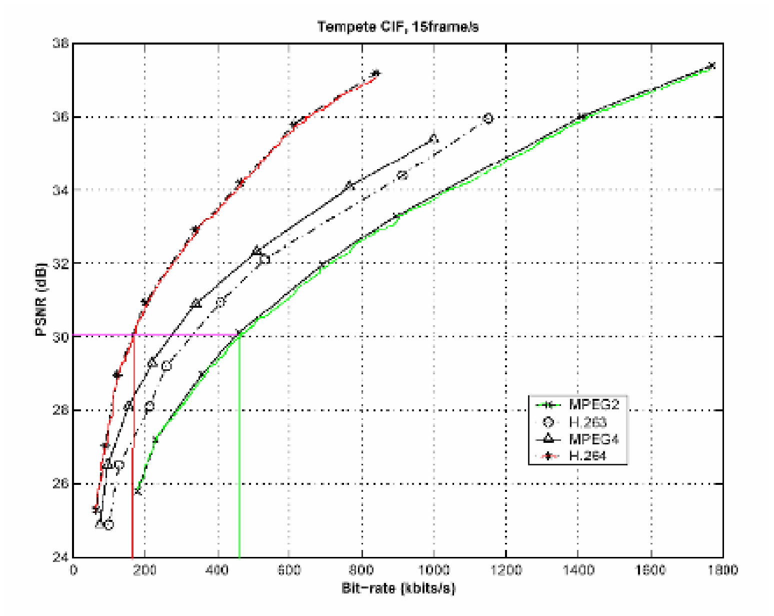


Figure 2.10: Objective comparison of H.264, MPEG-4, H.263 and MPEG-2 [13]

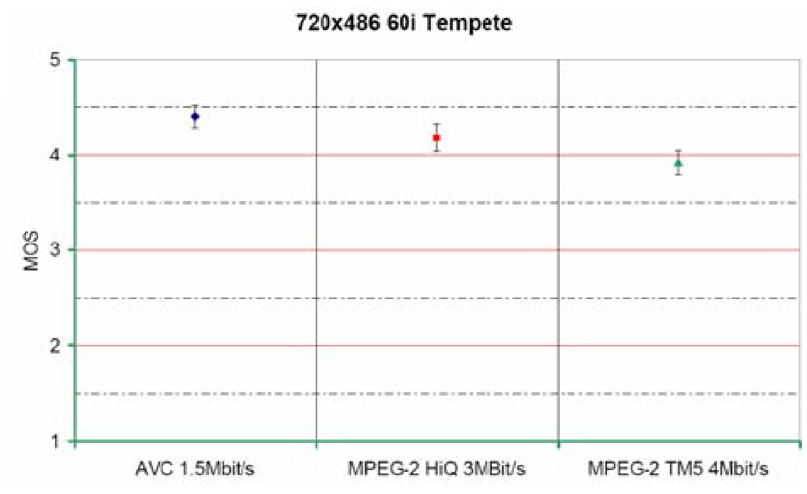
2.6 Applications

H.264 has very wide area of application since it allows a few Kbit to Several Mbit bit-rates, very small image sizes to huge pictures. H.264 is used both in very low bit-rate internet streaming applications and in HDTV broadcasts. In cinema sector, it used as a nearly lossless video compression standard. HD DVD and Blu-ray movies,

Table2.5: Feature comparison of MPEG-1, MPEG-2, MPEG-4 and H.264 [49]

Feature	MPEG-1	MPEG-2	MPEG-4	H.264
Macroblock Size	16x16	16x16 (frame mode) 16x8 (field mode)	16x16	16x16
Block Size	8x8	8x8	8x8,16x8,16x16	4x4,4x8,8x4 8x8,8x16,16x8 16x6
Transform	DCT	DCT	DCT/Wavelet	Integer
Transform Size	8x8	8x8	8x8	4x4
Quantization Step Size	Increases with constant increment	Increases with constant increment	Vector quantization used	Step sizes that increase at the rate of 12.5%
Entropy Coding	VLC	VLC	VLC	CAVLC/CABAC
Motion Estimation and Compensation	Yes	Yes	Yes	Yes (more flexible)
Pel Accuracy	Integer 1/2 pel	Integer 1/2 pel	Integer 1/2 pel 1/4 pel	Integer 1/2 pel 1/4 pel
Profiles	None	5 profiles + levels	8 profiles + levels	3 profiles + levels
Reference Frame	Single frame	Single frame	Single frame	Multiple frames
Picture Types	I,P,B,D	I,P,B	I,P,B	I,P,B,SI,SP
Transmission Rate	Up to 1.5Mbps	2-15Mbps	64kbps-2Mbps	64kbps-150Mbps
Encoder Complexity	Low	Medium	Medium	High
Backward compatibility	Yes	Yes	Yes	No

Digital Video Broadcast (DVB), Computers, Mobile Phones, Video Surveillance, Video Conferencing are some of the popular application areas of H.264 [4].



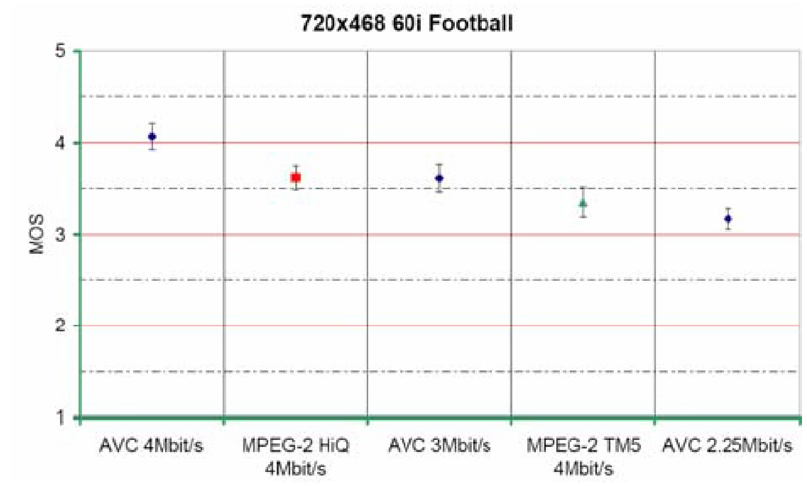


Figure 2.11: Subjective comparison of H.264/AVC and MPEG-2 at different bit rates for two different videos [47]

CHAPTER 3

LITERATURE SURVEY

3.1 Intra Coding in H.264

Intra Coding in H.264 is used to encode I frames. Intra prediction uses only spatial information (neighbor pixels) in the same frame to encode entire frame. Intra coding efficiency has more importance in compression than the inter frame coding since I frames holds much more space compared to B and P frames.

Intra coding consists of four steps. The image macroblock is first predicted from its neighbors. Then, intra prediction residuals are obtained by subtracting prediction block from original current macroblock. Residuals are quantized and transformed at the same time in quantization and transform block. After transformation, scanning order of coefficients are changed to improve entropy coding efficiency. Lastly, entropy coding is implemented on rescanned transform coefficients and bit string is obtained. Figure 3.1 shows the block diagram of intra coding in H.264.

In H.264, a new approach of intra prediction is used. Different than previous codecs, H.264 implements intra prediction in spatial domain rather than the frequency domain. H.264 uses neighbor pixel values that are in previously coded macroblocks and predict the current macroblock from its neighbors. This is called directional prediction, and prediction is done in the specified direction. Advantage of spatial domain prediction is that it gives much more efficient results compared to frequency domain and it is more intuitive for humans [13].

H.264 intra prediction have nine prediction modes for 4x4 and 8x8 luma blocks, four prediction modes for 16x16 luma blocks [35]. The difference in the count of prediction modes is due to the fact that 16x16 blocks are more smoother and 4x4 blocks are more detailed and require more options to be predicted correctly. Four intra prediction modes for 16x16 luma blocks are: vertical prediction, horizontal prediction, DC prediction, and plane prediction. Those four intra prediction modes are the first four modes of 4x4 luma blocks. 4x4 luma blocks include five more intra prediction modes. Figure 3.2 shows nine intra prediction modes for 4x4 luma blocks. The name of nine intra prediction modes are shown at the above of each prediction shape in Figure 3.2.

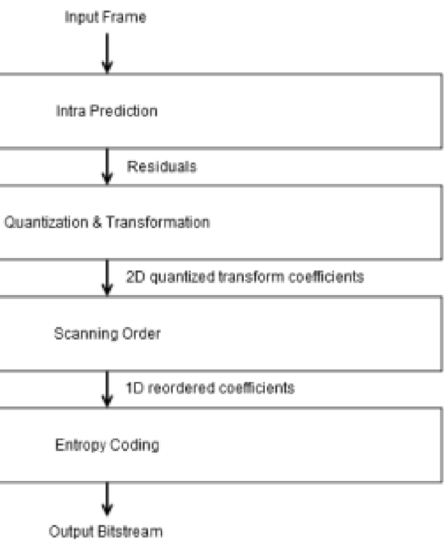

Input Frame
Intra Prediction
Residuals
Quantization & Transformation
2D quantized transform coefficients
Scanning Order
1D reordered coefficients
Entropy Coding
Output Bitstream

Figure 3.1: Intra coding block diagram

In 4x4 luma intra prediction at most 13 neighbor samples are used.

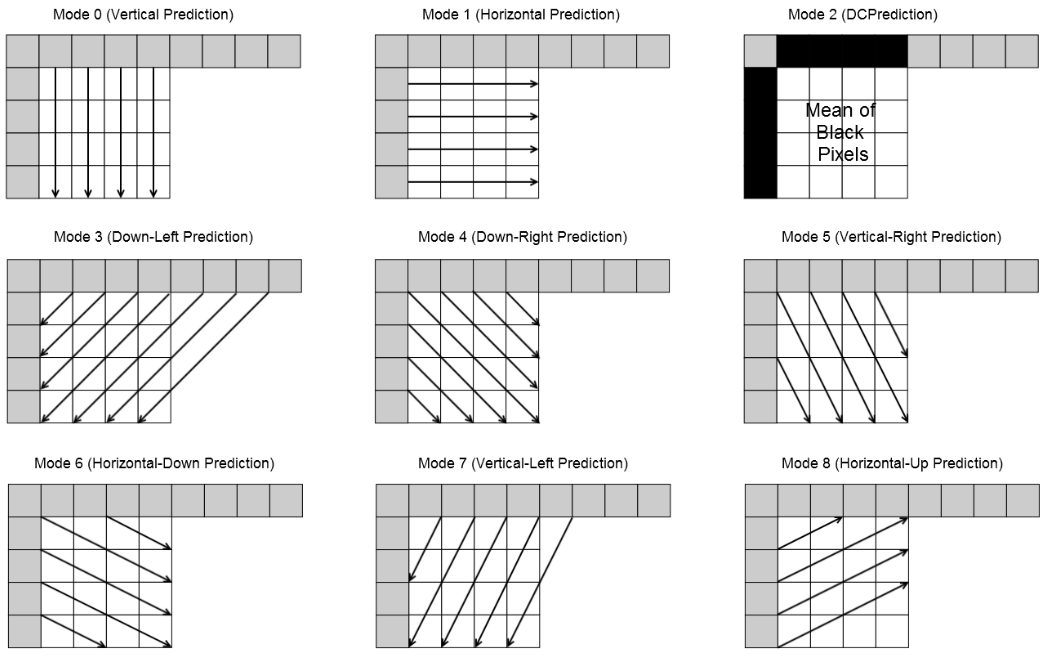

Mode 0 (Vertical Prediction)
Mode 1 (Horizontal Prediction)
Mode 2 (DCPrediction)
Mean of Black Pixels
Mode 3 (Down-Left Prediction)
Mode 4 (Down-Right Prediction)
Mode 5 (Vertical-Right Prediction)
Mode 6 (Horizontal-Down Prediction)
Mode 7 (Vertical-Left Prediction)
Mode 8 (Horizontal-Up Prediction)

Figure 3.2: Default intra prediction modes

H.264 encoder tries all intra prediction modes and decides on the best matching mode at run time in order to achieve coding efficiency when Rate Distortion Optimization (RDO) is enabled. Applying all prediction modes increases the complexity of encoder

[15].

Spatial redundancies in intra prediction residuals should be eliminated. In H.264, integer transform (an approximation to DCT) is used to transform blocks. Also, quantization is employed to reduce required number of bits to represent each pixel. Quantization step size can be chosen in H.264 according to application requirements.

Derivation of integer transform and quantization in H.264 is given below.

Let X be the input image block, Y be the output transform coefficients, Q_{step} be the quantization step . Then, a standard DCT transform and quantization is defined like this:

$$Y = Round\left(DCT\{X\}\frac{1}{Q_{step}}\right) \tag{3.1}$$

Let's divide DCT block in two parts; one is a core transform C_f, one is a scaling matrix S_f. Then Equation 3.1 becomes:

$$Y = Round\left(S_f\{C_f\{X\}\}\frac{1}{Q_{step}}\right) \tag{3.2}$$

Let's divide quantization part in two parts by introducing a constant 2^{15}.

$$Y = \left(\frac{1}{2^{15}}\right)\left(\frac{2^{15}}{Q_{step}}\right)(S_f\{C_f\{X\}\}) \tag{3.3}$$

Lets combine S_f and $\left(\frac{2^{15}}{Q_{step}}\right)$ parts and name it M_f such that $M_f \approx \frac{S_f 2^{15}}{Q_{step}}$

$$Y = \left(\frac{1}{2^{15}}\right)(M_f\{S_f\{C_f\{X\}\}\}) \tag{3.4}$$

Equation 3.4 is the method used in H.264 standard. Its reverse process is exactly defined for decoder.

Richardson [38] derives C_f and M_f as in the following:

$$C_f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & 1 \end{bmatrix}$$

$$M_f = \begin{bmatrix} \frac{m(QP\%6,0)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,0)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} \\ \frac{m(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,1)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,1)}{2^{\lfloor QP/6 \rfloor}} \\ \frac{m(QP\%6,0)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,0)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} \\ \frac{m(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,1)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{m(QP\%6,1)}{2^{\lfloor QP/6 \rfloor}} \end{bmatrix}$$

$$where$$

$$m_{i,j} = \begin{bmatrix} 13107 & 5243 & 8066 \\ 11916 & 4660 & 7490 \\ 10082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \\ 7282 & 2893 & 4559 \end{bmatrix}$$

In the similar way to forward transform and quantization, Richardson [38] derives corresponding dequantization and inverse transform as:

$$Y = \left(\frac{1}{2^6}\right)(C_i\{V_i\{X\}\}) \tag{3.5}$$

Richardson [38] also derives inverse transformations for 4x4 blocks C_i and S_i as in the following:

$$C_i = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & \frac{1}{2} & -\frac{1}{2} & -1 \\ 1 & -1 & -1 & 1 \\ \frac{1}{2} & -1 & 1 & -\frac{1}{2} \end{bmatrix}$$

$$V_i = \begin{bmatrix} \frac{v(QP\%6,0)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,0)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} \\ \frac{v(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,1)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,1)}{2^{\lfloor QP/6 \rfloor}} \\ \frac{v(QP\%6,0)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,0)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} \\ \frac{v(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,1)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,2)}{2^{\lfloor QP/6 \rfloor}} & \frac{v(QP\%6,1)}{2^{\lfloor QP/6 \rfloor}} \end{bmatrix}$$

$$where$$

$$v_{i,j} = \begin{bmatrix} 10 & 16 & 13 \\ 11 & 18 & 14 \\ 13 & 20 & 16 \\ 14 & 23 & 18 \\ 16 & 25 & 20 \\ 18 & 29 & 23 \end{bmatrix}$$

With Equation 3.4, Equation 3.5 and the matrices in hand, transform and quantization (and their inverse) can be easily calculated.

In frame coding, scanning is used to order coefficients in decreasing order and gather zero-values coefficients together. After scanning, 2D block is reordered in a 1D vector with all the coefficients in 2D block. Since DCT gathers highest coefficients at the top-left corner and smallest coefficients at the right-bottom corner for typical images and typical prediction residuals, zig-zag scanning order is employed in H.264 standard as seen in Figure 2.6

For different transforms, scanning pattern of H.264 should be changed in order to benefit from new transforms advantages. Otherwise, a mismatching scanning pattern will give disastrous results in terms of compression efficiency.

As explained in Chapter 2, CAVLC and CABAC algorithms are employed in order to entropy-code the rescanned transform coefficients and obtain compressed bit string.

Decoding process of intra coded frames is the reverse process of encoding. Bit string is first entropy decoded. Scanning order of coefficients are transformed to original order. Then, inverse transform and dequantization are applied. Finally, macroblock is reconstructed from its prediction. Intra decoding block diagram is given in Figure 3.3

3.2 Related Work for Intra Coding

In H.264 standard, directional (angular) spatial prediction is employed to predict current block from its neighbors. This approach has had many successes over previous methods. However, several different methods of intra prediction are proposed to improve intra coding efficient of H.264 standard. Some [29] [34] suggest that angular prediction used in H.264 should be improved and they offer new angular prediction methods. [59] proposes that distance should also be used in determination of used weights during prediction process, since distance from neighbor pixels affects current pixel value. There are many intra prediction methods that employs adaptive linear intra prediction in the literature such as [31] and [10]. In the literature there are many different approaches to predict intra coded blocks. In [50] and [63], template matching is used to predict blocks. Others [61] [62] employs linear filtering to intra-predict the current block. In the following sections, the mentioned intra prediction methods are

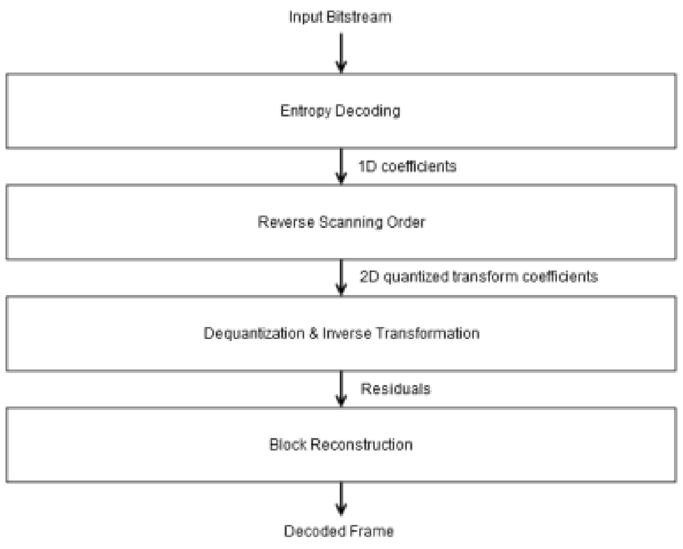


Figure 3.3: Intra decoding block diagram

explained. Then, Markov based intra prediction method, which is used in this research, is discussed in detail.

3.2.1 Angular Prediction

Angular intra prediction assumes that there is a high spatial correlation between the predicted pixel and their neighbors. Since this approach gives good results and imposes low implementation complexity, angular prediction is used in H.264 and it will be used in new High Efficiency Video Coding (HEVC) standard.

The paper [29], adopted by HEVC as its intra prediction method, claims that their angular intra prediction method gives higher compression ratio by upto 10% compared to H.264 and it is suitable for high resolution videos due to effective implementation in hardware.

Lainema and Ugur propose an angular prediction method to increase the number of angular directions of H.264 intra prediction and obtain more accurate results. Their method uses displacement vector and reference pixels to predict a value, then final predicted value is obtained by interpolation [29]. Their algorithm is adaptable to varying block sizes. They use different number of directions for different block sizes. The number of directions used in [29] and inherently in HEVC standard are given in Table 3.1.

The algorithm to intra-predict current pixel for 32 directions used in HEVC standard

Table3.1: The number of prediction directions in HEVC according to given block size

Prediction Block Size	Number of directions
4x4	16
8x8	33
16x16	33
32x32	33
64x64	2

[29] is given in Equation 3.6.

$$P_{x,y} = ((32 - w_y)R_i + w_y R_{i+1} + 16) >> 5 \tag{3.6}$$

where

$P_{x,y}$ is the predicted pixel value

$x \ and \ y$ are the predicted pixel coordinates assuming top-left corner as origin

$R_i \ and \ R_{i+1}$ are the reference samples

w_y is the weight. Calculation of weight is done via a simple AND operation at run time

Matsuo and Takamura propose a method to improve coding gain in HEVC intra prediction. Their proposal is to increase number of taps in the interpolation tap filter of intra prediction from 2 to 4 or 6. They obtain 2.2% increase in coding gain in the luma component for 4x4 and 8x8 blocks [34].

3.2.2 Distance Based Weighted Prediction

Distance based weighted prediction is a special form of linear prediction. It uses neighbor pixels and distance information to predict current pixel value. The DWP (distance based weighted prediction) algorithm described in [59] is defined as in the following Equation 3.7:

$$P_{x,y} = C_y Rrow_x + C_x Rcol_y + \Delta \tag{3.7}$$

where

$P_{x,y}$ is the predicted pixel value

$x \ and \ y$ are the predicted pixel coordinates assuming top-left corner as origin

$Rrow_x$ is the reference sample at the same row

$Rcol_y$ is the reference sample at the same column

C_x is the multiplication coefficient which is proportional to $\frac{1}{x+1}$

C_y is the multiplication coefficient which is proportional to $\frac{1}{y+1}$

Δ is the adjusting factor

DWP algorithm requires 2 multiplications, 1 division, and 6 additions; therefore, it is not suitable for hardware implementation. Yu proposes integral DWP algorithm which has lower computational complexity in the same paper [59]. He substitutes division by shift operation and rounds some numbers and get the integral DWP as in Equation 3.8 [59].

$$P_{x,y} = (5(Rrow_x - Rcol_y) + (Rcol_y << 3) + 4) >> 3 \tag{3.8}$$

where

$P_{x,y}$ is the predicted pixel value

$x \ and \ y$ are the predicted pixel coordinates assuming top-left corner as origin

$Rrow_x$ is the reference sample at the same row

$Rcol_y$ is the reference sample at the same column

Yu shows with experimental results that by using integral DWP method, an average of 0.1 dB peak signal to noise ratio (PSNR) enhancement is obtained in H.264 [59].

3.2.3 Linear Prediction

Linear prediction is the general form of both angular and distance based prediction. The current pixel value is predicted using all neighbor pixels that are previously coded. When predicting current pixel value, a weight is associated with each neighbor. The difficult part of linear prediction is the determination of weights. There are many adaptive and non-adaptive approaches in the literature.

In non-adaptive linear filtering, weight coefficients are determined offline. The advantage of non-adaptive filtering is that linear prediction only uses static coefficients and overhead is low compared to adaptive linear prediction. However, lack of adaptability gives results lower than adaptive linear prediction methods. The basic algorithm for linear prediction is given in Equation 3.9.

$$P_i = W_i' R \tag{3.9}$$

where

P_i is the predicted pixel value

i is the index of current pixel in the block

W_i is the weight vector

R is the reference values vector (neighbor pixels vector)

There are nine prediction modes in H.264. We can parameterize the linear prediction with mode selection parameter k. Let's introduce intra mode selection parameter and rewrite Equation 3.9.

$$P_i = W'_{k,i} R \tag{3.10}$$

where

P_i is the predicted pixel value

k is the prediction mode

i is the index of current pixel in the block

W_i is the weight vector

R is the reference values vector (neighbor pixels vector)

Equation 3.10 is the general linear prediction algorithm for H.264 kind intra prediction applications. The size of weight vector and reference value vector can change according to the number of used neighbors. As seen in 3.10, linear filtering is just a vector multiplication. However, it includes a lot of multiplications in it, and hardware implementation can be complex.

In [61], the left and top neighbors are used for linear prediction and linear prediction coefficients are trained offline. Linear prediction with offline training gives good results over H.264 default intra prediction mode, however, correlation of training and test dataset plays an important role. Result can vary widely according to different test datasets because of the offline specific training.

Zhang [62] proposes a similar approach to [61]. For each prediction mode and each index, a different weight vector is introduced. The weights are determined offline using least square error technique. To reduce the complexity, Zhang eliminated some neighbors and corresponding coefficient [62].

Liu's method [31] depends on determining the weight coefficients by using least square prediction and the author claims that local texture characteristics and intra prediction mode is adaptively selected by this approach.

In least square prediction, neighbors are filtered using spatial and temporal filters. Then, weights are determined within a training window from the reconstructed data [31]. In this way, neighbors are used to both predict the current pixel and train/determine weights. Therefore, weights change over time and adapt to current block.

A block based and line based linear prediction method is proposed by Chen and Han [10] instead of pixel based prediction method. They extract a set of weights for each block and each line according to current mode [10]. Since weights are updated for each block and each line, the algorithm is adaptive. It is the improved version of Liu's work [31].

3.2.4 Prediction via Template Matching

In intra prediction, the current block is generally predicted from neighbor pixels and further blocks are not mostly used. This approach generally gives good results for relatively smoother blocks. However, it might be possible to match the current block with some neighbor blocks. Template matching is to match an image/block with another image/block by their similarities. Tan [50] modifies H.264 JM software to include its template matching intra prediction algorithm. Tan, in his research, divides each 4x4 block into 2x2 blocks, and searches each 2x2 block within already predicted blocks. The search block diagram is shown in Figure 3.4 The template matching algorithm dramatically increases encoder and decoder complexity, however, it results in average bit-rate savings of 11.3%.

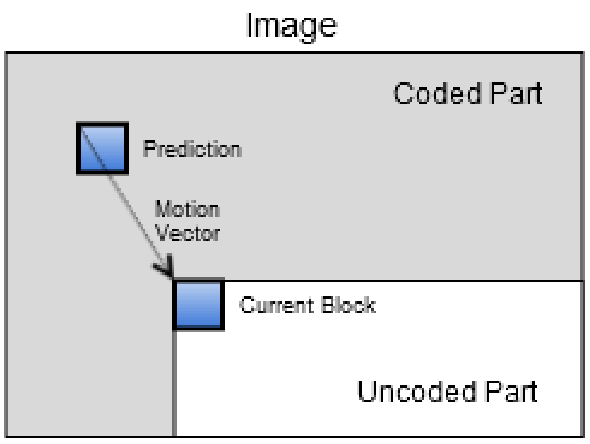


Figure 3.4: Template matching for intra frame coding

In another work, template matching is employed and then illumination compensation procedure is applied in order to further improve coding efficiency [63]. Zheng compensates illumination variations locally by introducing scaling and offset parameters to adjust contrast and brightness [63]. His work slightly increases coding gain over standard template matching algorithms.

3.2.5 Markov Based Intra Prediction

Kamisli [24] [25] uses a linear prediction model to predict the current block. The research differs from other linear prediction methods in weight determination part such that Kamisli models image blocks statistically rather than using least squares approach to determine weights [24].

Kamisli modifies the H.264 standard such that only three prediction modes (vertical, horizontal and DC) are modified. The used number of neighbors are three for each pixel in the current block [24]. The neighbors that are used in the prediction process can be seen in Figure 3.6. The numbering style for neighbors and block pixels are shown in Figure 3.5.

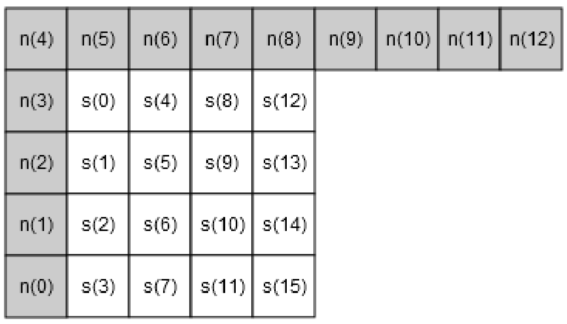

n(4) n(5) n(6) n(7) n(8) n(9) n(10) n(11) n(12)
n(3) s(0) s(4) s(8) s(12)
n(2) s(1) s(5) s(9) s(13)
n(1) s(2) s(6) s(10) s(14)
n(0) s(3) s(7) s(11) s(15)

Figure 3.5: The numbering of block and neighbor pixels in Kamisli's work [24]

The parameters used in Figure 3.6 are defined as;

n_l is the neighbor pixel l=0..12,

v is the vertical distance of current pixel to upper neighbors,

h is the horizontal distance of current pixel to left neighbors,

$s(k)$ is current pixel value to be predicted,

k is the pixel index of currently predicted pixel.

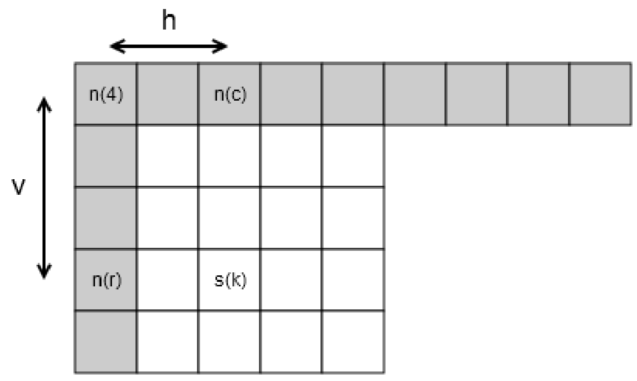


Figure 3.6: Kamisli's prediction method from 3 neighbors [24]

He uses the following formula to predict current pixel:

$$s_d(k) = w_d' \begin{bmatrix} n(r) \\ n(4) \\ n(c) \end{bmatrix} \tag{3.11}$$

where

d is the current prediction mode d=0,1,2 (first three prediction modes of H.264)

w_d is the weight vector consisting of three elements
item[$s_d(k)$] is current pixel value to be predicted for each prediction mode d

The novel part of the algorithm is in determining the weights. He models the relationship between block pixels s(k) k=0..15 and neighbors pixels n(l) l=0..12 using a first order Markov model as seen in Equation 3.12 [24]

$$R(v,h) = \rho_y^v \rho_x^h \tag{3.12}$$

where

v is the vertical distance between two pixels that correlation is sought

h is the horizontal distance between two pixels that correlation is sought

ρ_x is a parameter between -1 and 1. It models the horizontal correlation of two adjacent pixels

ρ_y is a parameter between -1 and 1. It models the vertical correlation of two adjacent pixels

Using the assumed first order Markov model, Kamisli derives R_n $R_{snd,k}$ and R_n^{-1}. Finally, he derives the weight vector as; [24]

$$w_d(k) = \begin{bmatrix} \rho_x^h \\ -\rho_y^v \rho_x^h \\ \rho_y^v \end{bmatrix} \tag{3.13}$$

To derive the weight vector and parameters ρ_x, ρ_y, Kamisli uses a training video dataset and intra codes every 10th frame of video sequences with four different quantization parameters in action [25]. The result of this work (the parameters ρ_x and ρ_y) are shown in Table 3.2.

Table3.2: ρ_x and ρ_y parameters for 4x4 blocks and prediction modes 0,1,2 estimated from the training set in Kamisli's work [25]

Prediction Mode	ρ_x	ρ_y
0 (vertical)	0.21	0.96
1 (horizontal)	0.97	0.30
2 (DC)	0.38	0.42

Kamisli compares his algorithm with two algorithms: default H.264 intra prediction and linear estimation with least square error optimization [25]. The results for 4x4 blocks are given in Figure 3.7

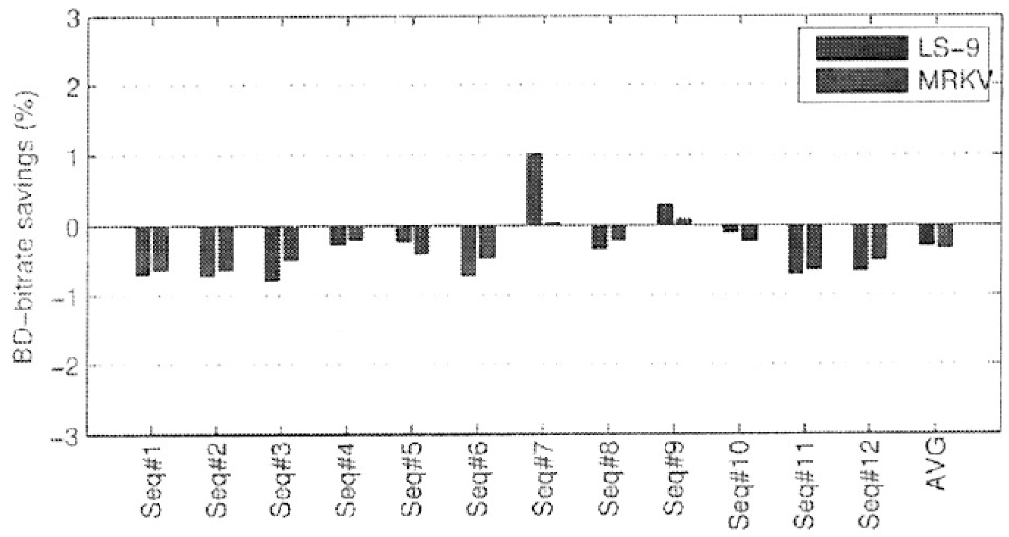


Figure 3.7: Comparison of Kamisli's Markov algorithm and LS-9 algorithm [25]

The proposed algorithm offers reduced storage space and lower computation complexity. The Markov based intra prediction can be applied offline or online. Also, it presents an analytical expression to derive optimal weights [25]

[26] extends Markov based intra prediction to any arbitrary angular direction. The way to achieve any angular direction is to use an interpolated image for given angular direction. Image block pixels and neighbor pixels are renamed as in Figure 3.8. Then, new formula for intra pixel prediction becomes 3.14

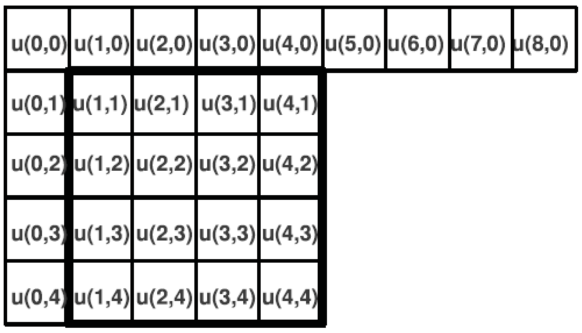


Figure 3.8: 4x4 pixel block and its neighbors [26]

$$\tilde{u}_d(i,j) = \rho_x \tilde{u}_{x,d}(i,j) + \rho_y \tilde{u}_{y,d}(i,j) - \rho_x \rho_y \tilde{u}_{x,y,d}(i,j) + e(i,j) \tag{3.14}$$

where

$\tilde{u}_d(i,j)$ is the predicted pixel value

$\tilde{u}_{x,d}(i,j), \tilde{u}_{y,d}(i,j) and \tilde{u}_{x,y,d}(i,j)$ are the interpolated pixel values according to the given direction d. Interpolated pixel values are generally chosen as the half pixel along direction d.

$e(i,j)$ is the prediction error

Kamisli changes the number of independent parameters used in Markov-based intra prediction. He conducts experiments with the number of independent parameters 1,2, and 4. As the number of independent parameters changes, the compression performance and complexity also change. One can use different number of independent parameters according to compression requirements. In table 3.3, neighbor pixels corresponding to the independent parameter are given where the number of independent parameters is equal to 4.

The experiment results for all angular directions in H.264 and for different number of independent parameters are given in Figure 3.9. The overall gain varies from around 0.5% to around 1%. As the number of independent parameters increase, compression performance increases.

Table3.3: Neighbor pixels that are multiplied by independent parameters when number of independent parameters are equal to 4 [26]

Prediction Mode	ρ_1	ρ_2	ρ_3	ρ_4
0	$\tilde{u}(i-1,j)$	$\tilde{u}(i-1,j-1)$	$\tilde{u}(i,j-1)$	
1	$\tilde{u}(i-1,j)$	$\tilde{u}(i-1,j-1)$	$\tilde{u}(i,j-1)$	
2	$\tilde{u}(i-1,j)$	$\tilde{u}(i-1,j-1)$	$\tilde{u}(i,j-1)$	
3	$\tilde{u}(i-1,j)$	$\tilde{u}(i,j-1)$	$\tilde{u}(i+1,j-1)$	$\tilde{u}(i+2,j-1)$
4	$\tilde{u}(i-1,j)$	$\tilde{u}(i-1,j-1)$	$\tilde{u}(i,j-1)$	$\tilde{u}(i-2,j-1)$
5	$\tilde{u}(i-1,j)$	$\tilde{u}(i-1,j-1)$	$\tilde{u}(i,j-1)$	$\tilde{u}(i-2,j-1)$
6	$\tilde{u}(i-1,j)$	$\tilde{u}(i-1,j-1)$	$\tilde{u}(i,j-1)$	$\tilde{u}(i-1,j-2)$
7	$\tilde{u}(i-1,j)$	$\tilde{u}(i-1,j-1)$	$\tilde{u}(i,j-1)$	$\tilde{u}(i+1,j-1)$
8	$\tilde{u}(i,j-1)$	$\tilde{u}(i-1,j-1)$	$\tilde{u}(i-1,j)$	$\tilde{u}(i-1,j+1)$

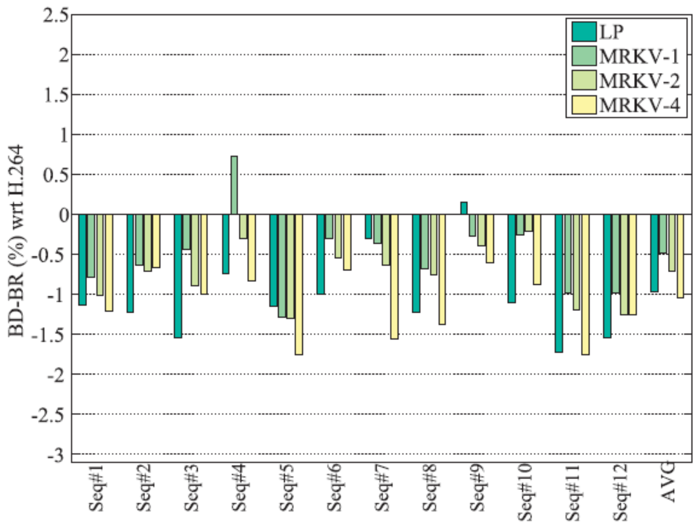


Figure 3.9: Linear Prediction, Markov-1, Markov-2 and Markov-4 Compression Performance Results for 4x4 blocks only [26]

3.3 Related Work for Transforms

Transform coding is a widely used technique in image processing, image compression, and video compression applications. The purpose in transforming the image in video coding is generally to concentrate energy into as few coefficients as possible. Therefore, the image can be represented with less number of bits. Transforms are used in different ways. Transform operation can be directly applied on image blocks, in addition, transform operation can be applied the residuals remaining after prediction which is the technique used in H.264.

There are many types of transforms used for images and residuals. They serve for different purposes and offer different characteristics. Karhunen-Loeve Transform is the optimal transform; however, it lacks a fast implementation. DCT is one of the most widely used transform in image processing and video coding since it gives good results, it is widespread, and it has fast computation algorithms. There are other transforms that are not as popular as KLT or DCT, but can achieve better coding gains.

The transform used in video coding highly vary. Some transforms are only 1D directional such as [27]. Some transforms make use of directional transforms. They use different transforms for different directions and modes such as [58]. Some transforms used in literature combine or alternate two 1D transforms to code intra frames [40] [41]. Some transforms optimize prediction and transformation together and offers different transforms for different predictions [17]. Some uses unitary transforms such as Odd Sine Transform of type 3 [20]. Some use the transforms that are popular in different areas such as discrete sine [52]. More complex algorithms such as two dimensional discrete wavelet transform (DWT) are used in some specific applications [21].

In this thesis, we will mention about discrete cosine transform (DCT), Karhunen-Loeve transform (KLT), discrete sine transform (DST) and asymmetric discrete sine transform (ADST) in depth. The mentioned transforms below are implemented and compared in the following sections.

3.3.1 Discrete Cosine Transform

High ratio of block based image coding algorithms choose to implement 2D-DCT transform. 2D-DCT transform is generally applied as two 1D transforms. First, vertical 1D-DCT is applied, then horizontal 1D-DCT is applied on the already 1D transformed image or vice versa. The popularity of DCT comes from its low complexity implementation, high energy compaction, and its convergence to KLT.

NxN image will give NxN 2D-DCT coefficients after transformation. This same-size feature enables easy observation of coefficients and frequency components. Therefore, 2D-DCT is intuitive in terms of frequency components of image.

As mentioned above, 2D-DCT is implemented as two 1D-DCT transforms. In Equation 3.15, 1D-DCT transformation is given.

$$X_k = \sum_{n=0}^{N-1} x_n cos\left[\left(n+\frac{1}{2}\right)\frac{\pi k}{N}\right] \tag{3.15}$$

where

X is the 1D-DCT transform

x is the input vector

k is the vector index

N is the length of transform

2D image is first applied 1D-DCT in one direction, then resulting image is transformed again in the other direction one more time with 1D-DCT. The transformation function for 2D-DCT is given in Equation 3.16

$$X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1,n_2} cos \left[\left(n_1 + \frac{1}{2} \right) \frac{\pi k_1}{N_1} \right] cos \left[\left(n_2 + \frac{1}{2} \right) \frac{\pi k_2}{N_2} \right] \tag{3.16}$$

where

X is the 2D-DCT transform of input image

x is two 2D input image

k_1 *and* k_2 are the image indeces

N_1 *and* N_2 are the dimensions of 2D-DCT transform

In addition, there exists many fast computation algorithms for 1D-DCT and 2D-DCT such as [12] [43] [51] [28] [39] and [37]. Some of these algorithms are using DFT, some algorithms are standalone. Most requires that the size of DCT is $N = 2^m$, m is a non-negative integer.

Another way of computing 2D-DCT transform is matrix multiplication. Matrix multiplication method requires high number of multiplications, but its analytical expression is very simple. 2D-DCT transform as matrix multiplication is given in Equation 3.17

$$X = TxT' \tag{3.17}$$

where

X is the 2D-DCT transform of input image

x is 2D input image

T is the 2D-DCT transform matrix

The DCT transform matrix used in matrix multiplication for 4x4 blocks is given as following.

$$\begin{bmatrix} 0.5000 & 0.5000 & 0.5000 & 0.5000 \\ 0.6533 & 0.2706 & -0.2706 & -0.6533 \\ 0.5000 & -0.5000 & -0.5000 & 0.5000 \\ 0.2706 & -0.6533 & 0.6533 & -0.2706 \end{bmatrix} : \text{DCT Transform Matrix for 4x4 Blocks}$$

DCT has been used in many applications, there exists a lot of papers on the subject. In [20] and [45], it is mentioned about four different types of DCT, their features, and formulas. Each DCT transform type has different characteristics and each is suitable for different tasks.

2D-DCT transform is generally thought as combination of vertical and horizontal transforms. Therefore, conventional 2D-DCT gives good results when image contains vertical and horizontal edges. Zeng proposes a method to widen this concept of 2D-DCT. He offers to implement two directional transforms in different directions other than vertical and horizontal according to characteristics of the input image block [60]. This directional DCT offers higher bit rate savings at the same time it increases the complexity.

The disadvantage of DCT is that it gives maximum amplitude at the block edges and causes artifact at the block transitions. Due to this fallback of DCT, a deblocking filter becomes essential as used in the H.264 coding standard.

3.3.2 Karhunen-Loeve Transform

Karhunen-Loeve Transform is the optimal transform such that it minimizes mean square error for a given input vector or image. The coefficients in KLT are derived from covariance matrix of the input images, therefore, it has different orthogonal basis vectors for different image characteristics, thus it is an adaptive transform. Although the adaptivity feature of KLT offers best results, it has different transform coefficients for different images and this makes implementation harder. Generally, KLT does not have a fast implementation due to changing coefficients of transform even though some fast algorithms have been offered in the literature such as [57].

3.3.2.1 2-D Separable Karhunen-Loeve Transform

The coefficients in KLT transform are real valued. Therefore, transform is generally computed with the following equation;

$$X = TxT'; \qquad (3.18)$$

where

X is the transform of input image,

x is 2D input image,

T is the transform matrix.

The tricky part in KLT is determination of transform matrix. Since transform matrix changes over changing input, transform matrix can be either obtained offline by training or online with respect to incoming images. However, online computation approach is generally not preferred due to increased complexity. And, offline training is not effective as online training due to non-adaptivity.

The eigenvectors of covariance matrix of input image sequence are the basis vectors of KLT. However, it is not always possible to find covariance matrix since input images are not exactly known. Biswas offers a technique to predict covariance matrix from the errors between current block and the same block in the previous frame [8]. KLT basis functions derived in encoder are derived in decoder too, using the similar technique of error analysis. He obtains high bit-rate savings compared to H.264 default intra coding [8]. Due to the high computational complexity of his KLT method, the implementation is difficult in hardware.

Yeo proposes low complexity mode dependent KLT transforms for intra coding. He proposes a low complexity approach and separable KLT transform. Also, he uses fixed mode dependent scan orders to increase coding efficiency [55] [57]. However, his implementation converges to DCT transform because of the fixed KLT transform.

3.3.2.2 2-D Non-Separable Karhunen-Loeve Transform

2-D Non-Separable Karhunen-Loeve Transform is applied like a 1-D transform. Input image is first converted to a 1-D vector by scanning the pixels in the block (generally left to right or top to bottom). Then, the obtained 1-D input vector is multiplied by transformation matrix T and 1-D transformed vector is obtained. By reordering transformed vector into 2-D image matrix, a Non-Separable Karhunen-Loeve Transform is implemented. A simple formula for 2-D Non-Separable Karhunen-Loeve Transform is given in 3.19.

$$\tilde{X} = T\tilde{x}; \tag{3.19}$$

where

$\tilde{X}$ is the transform of input vector (reoerdering $\tilde{X}$ into 2-D array gives X),

$\tilde{x}$; is ordered 1D input vector (ordering x into 1-D vector gives $\tilde{x}$),

T is the transform matrix.

To summarize, KLT transform is an adaptive transform and offers best compression when it is used in this manner. However, lack of fast computational algorithms and changing basis vectors make KLT a less preferred choice.

3.3.3 Discrete Sine Transform

Discrete sine transform is a close relative of DCT. DST has real valued transform matrix and is equal to imaginary parts of the DFT. DST offers different boundary conditions than the DCT, therefore, DST is preferred in some circumstances.

Like the DCT, DST has a short-hand formula as given in Equation 3.20.

$$X_k = \sum_{n=0}^{N-1} x_n sin\left[(n+1)(k+1)\frac{\pi}{N+1}\right] \tag{3.20}$$

where

X is the 1D-DST transform of input vector,

x is the input vector,

k is the vector index,

N is the length of DST transform.

DST basis vectors are first defined by Jain as in Equation 3.21 [19]. Jain also showed that KLT approaches to DST when certain boundary conditions are met by first order Markov sequence [19].

$$\phi_{i,j} = \sqrt{\frac{2}{N+1}} sin\left(\frac{\pi ij}{N+1}\right) \qquad \text{i,j = 1,2,...,N.} \tag{3.21}$$

where

$\phi_{i,j}$ is the basis vector of DST

$i\ and\ j$ are the basis vector indices

N is the length of transform

DST transformation can be done using matrix multiplication as done for DCT and KLT. The DST transform matrix for 4x4 blocks is given as following;

$$\begin{bmatrix} 0.3717 & 0.6015 & 0.6015 & 0.3717 \\ 0.6015 & 0.3717 & -0.3717 & -0.6015 \\ 0.6015 & -0.3717 & -0.3717 & 0.6015 \\ 0.3717 & -0.6015 & 0.6015 & -0.3717 \end{bmatrix} : \text{DST Transform Matrix for 4x4 Blocks}$$

Rose offers a mode dependent DST approach. He first introduces KLT/DST alternate transform, then introduces DCT/DST alternate transform [40]. Saxena claims that DST is the optimal transform under some conditions for intra prediction residuals [41]. Saxena [41] uses separable transforms and transforms each direction either using DCT or DST depending on prediction mode. Saxena achieves up to 4% higher compression ratio compared to DCT approach. [41].

Due to fixed transform matrix, DST has fast computation algorithms. Wang offers a fast computation algorithm for DST that uses existing fast algorithms for DCT [52]. He just adds two steps in front of DCT computation and obtains Discrete Sine Transform [52].

3.3.4 Asymmetric Discrete Sine Transform

Asymmetric discrete sine transform is a transform which is a close relative of DST. If the image is assumed as first order Gauss-Markov model, ADST codes image block in a way that is very close to optimal coding [17].

Han shows that ADST/DCT combination performs better than default H.264 algorithm both analytically and experimentally and reduces blocking artifacts [17].

Han's derivation of ADST is as follows. He first considers a zero mean, unit variance first order Gauss-Markov sequence [17];

$$x_k = \rho x_{k-1} + e_k \tag{3.22}$$

where

ρ is the correlation coefficient

e_k is white Gaussian noise with variance $1 - \rho^2$

x_k is the element of random vector x with index k

Here, x_0 denotes boundary pixel and other elements of vector x denotes the correlated random vector. Then, Han defines each element of x as [17];

$$\begin{aligned} x_1 &= \rho x_0 + e_1 \\ x_2 - \rho x_1 &= e_2 \\ &. \\ &. \\ &. \\ x_N - \rho x_{N-1} &= e_N \end{aligned} \tag{3.23}$$

where

ρ is the correlation coefficient

e_k is white Gaussian noise with variance $1 - \rho^2$

x_k is the element of random vector x with index k

N is the size of random vector x

He then write Equation 3.23 as matrix multiplication as shown in Equation 3.24 [17].

$$Q\overline{x} = \overline{b} + \overline{e} \tag{3.24}$$

where

Q is the matrix given in Equation 3.25 [17].

$\overline{b}$ is the vector given in Equation 3.26 [17].

$\overline{e}$ is the error vector of length N

$\overline{x}$ is the random vector of length N

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & ... \\ -\rho & 1 & 0 & 0 & ... \\ 0 & -\rho & 1 & 0 & ... \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & . & 0 & -\rho & 1 \end{bmatrix} \tag{3.25}$$

$$\overline{b} = \begin{bmatrix} \rho x_0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \tag{3.26}$$

Using the above equations, Han defines the prediction residual as given in Equation 3.27 [17].

$$\overline{y} = Q^{-1}\overline{e} \tag{3.27}$$

where

y is prediction residual

The autocorrelation matrix of prediction residual is given in Equation 3.28 [17].

$$R_{\overline{yy}} = E\{\overline{y}\overline{y'}\} = (1-\rho^2)Q^{-1}(Q')^{-1} \tag{3.28}$$

He asserts that KLT for $\overline{y}$ is the unitary matrix that diagonilizes P_1 in Equation 3.29.

$$P_1 = Q'Q = \begin{bmatrix} 1+\rho^2 & -\rho & 0 & 0 & ... \\ -\rho & 1+\rho^2 & -\rho & 0 & ... \\ 0 & -\rho & 1+\rho^2 & -\rho & ... \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & ... & -\rho & 1+\rho^2 & -\rho \\ 0 & ... & 0 & -\rho & 1 \end{bmatrix} \tag{3.29}$$

Assuming that $p \to 1$, Han uses $\hat{P_1}$ instead of P_1 [17]. $\hat{P_1}$ is given in Equation 3.30

$$P_1 = Q'Q = \begin{bmatrix} 1+\rho^2 & -\rho & 0 & 0 & ... \\ -\rho & 1+\rho^2 & -\rho & 0 & ... \\ 0 & -\rho & 1+\rho^2 & -\rho & ... \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ 0 & ... & -\rho & 1+\rho^2 & -\rho \\ 0 & ... & 0 & -\rho & 1+\rho^2-\rho \end{bmatrix} \tag{3.30}$$

Then, he finds eigenvalues of $\hat{P_1}$ and obtains basis vectors of a new transform which is now called ADST given in Equation 3.31

$$\phi_{i,j} = \frac{2}{\sqrt{2N+1}} sin\left(\frac{\pi i(2j-1)}{2N+1}\right) \qquad \text{i,j = 1,2,...,N.} \tag{3.31}$$

where

$\phi_{i,j}$ is the basis vector of ADST,

i *and* j are the basis vector indices,

N is the length of transform.

Basis vector definition of ADST, DST, and DCT are very close. All of them are unitary transforms.

In addition, it is possible to use matrix multiplication for ADST transformation. The ADST transform matrix for 4x4 block is given as following:

$$\begin{bmatrix} 0.2280 & 0.4285 & 0.5774 & 0.6565 \\ 0.5774 & 0.5774 & 0.0000 & -0.5775 \\ 0.6565 & -0.2280 & -0.5774 & 0.4285 \\ 0.4285 & -0.6565 & 0.5774 & -0.2280 \end{bmatrix} : \text{ADST Transform Matrix for 4x4 Blocks}$$

DCT transform and ADST transform are used alternately in Han's work for intra coding. He obtains higher compression ratio than the integer transform implementation of H.264 [17]

A similar result is obtained by Yeo [58] who derives Asymmetric Discrete Sine Transform by following a different path than Han's [17]. Yeo also achieves same formulas and similar coding gains in his research [58].

CHAPTER 4

IMPLEMENTATION

In this thesis, different transform algorithms are implemented for Markov based intra prediction [24] [25]. Implemented transform algorithms are compared with each other and default H.264 intra coding. In the implementation JM reference software with Markov based intra coding implementation is used. Transform algorithms and scanning orders are built on this software. Some restrictions and assumptions made are given in the following chapter.

4.1 Decisions, Restrictions, and Assumptions

- JM reference software 11.4 [3] is used in the implementation. The project is coded in C++ by modifying JM reference software.
- JM reference software is in intra coding mode, no inter frame coding is done in the thesis.
- Due to complexity of H.264 implementation, new features are not included and intra coding for 4x4 blocks are modified.
- Since luma and chroma component display different characteristics, chroma implementation is not modified. Modifications and new transforms are applied only on luma component.

4.2 Scanning Order

Scanning order plays an important role in intra coding. The transform coefficients should be listed in descending order before entropy coding for the highest compression ratio. In the following sections, we derive scanning orders for used transforms and give them as tables.

4.2.1 Derivation of Scanning Order

The autocorrelation matrix of residual vector was given in the previous pages, we rewrite Equation 3.28. 4.1 gives the autocorrelation matrix of 1D transform (vertical transform in our case).

$$R_{\overline{yy}} = E\{\overline{y}\overline{y'}\} = (1-\rho^2)Q^{-1}(Q')^{-1} \tag{4.1}$$

where

$R_{\overline{yy}}$ is the autocorrelation matrix of residual vector,

$\overline{y}$ is the residual vector,

ρ is the correlation coefficient,

Q is the matrix given in Equation 3.25,

T is the transform matrix.

If we transform this image block with transformation matrix T, the autocorrelation matrix of transformed residual vector becomes:

$$R_{\overline{cc}} = E\{\overline{c}\overline{c'}\} = E\{T\overline{c}\overline{c'}T'\} = TR_{\overline{yy}}T' \tag{4.2}$$

where

$R_{\overline{cc}}$ is the autocorrelation matrix of transformed residual vector,

$\bar{c}$ is the transformed residual vector,

T is the transform matrix.

The diagonal elements of $R_{\overline{cc}}]$ gives the variance of each transform coefficient. Let's name them σ_1, σ_2 ... σ_N for a vector of length M as seen in Equation 4.3

$$R_{\overline{cc}} = \begin{bmatrix} \sigma_1 & . & . & . & & \\ . & \sigma_2 & . & . & & \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ . & . & . & \cdots & . \\ . & . & . & \cdots & \sigma_N \end{bmatrix} \tag{4.3}$$

where

$R_{\overline{cc}}$ is the autocorrelation matrix of transformed residual vector

σ_1, σ_2 ... σ_N are the variances

As a result of vertical transformation, we obtain variances and use them to derive the variances of each pixel after 2D transformation. Let's introduces σ coefficients to Equation 4.2.

$$R_{\overline{cc}} = E\{\overline{c}\overline{c'}\} = E\{T\overline{c}\overline{c'}T'\} = TR_{\overline{yy}}T'\sigma_n \tag{4.4}$$

where

$R_{\overline{cc}}$ is the autocorrelation matrix of transformed residual vector

$\overline{c}$ is the transformed residual vector

T is the transform matrix

n is the row index in the image block, n=1,2..N

By introducing σ coefficients, the multiplication coefficient of each row is defined. See the new multiplication coefficients of each row in Equation 4.4

$$\begin{bmatrix} \sigma_1 & \sigma_1 & ... & \sigma_1 \\ \sigma_2 & \sigma_2 & ... & \sigma_2 \\ . & . & ... & . \\ . & . & ... & . \\ . & . & ... & . \\ \sigma_N & \sigma_N & ... & \sigma_N \end{bmatrix} \tag{4.5}$$

Each row of image block is transformed with the same or different transformation matrix T using the formula 4.4. σ multiplication coefficients in Equation 4.5 which is obtained from 4.3 should be substituted in Equation 4.4. As a result, a number for each pixel in the image block is obtained. When these numbers are listed in descending order, the scanning order of the corresponding transform is obtained. An example scanning order and corresponding table for 4x4 block is given in Equation 4.6 and Table 4.1, respectively.

$$\begin{aligned} \textit{Example Scanning Order} = \{&(1,1),(1,2),(1,3),(1,4),(2,1),(2,2),\\ &(2,3),(2,4),(3,1),(3,2),(3,3),(3,4),\\ &(4,1),(4,2),(4,3),(4,4)\} \end{aligned} \tag{4.6}$$

Table4.1: Example scanning order in image form for 4x4 Block corresponding to scan order in equation 4.6

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

4.3 Derivation of Transform

4.3.1 Obtaining ρ's from Correlations

According to Markov model, the correlation between two neighbor pixels in the same direction are the same. The correlation between two pixels can be horizontal correlation, vertical correlation or cross correlation. In Figure 4.1, the correlations C_x, C_y, C_m, and C_n are shown.

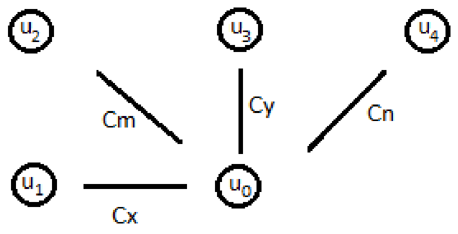


Figure 4.1: Correlations C_x, C_y, C_m, and C_n between any two neighbors in an image block (circles denote pixels, lines denote correlations between the neighbor pixels)

By using the correlations C_x, C_y, C_m, and C_n, we can easily determine Markov-based intra prediction parameters ρ_1, ρ_2, ρ_3, and ρ_4. ρ values are found by multiplying the inverse of autocorrelation matrix of neighbors with the correlation between current pixel u_0 and its neighbors u_1, u_2, u_3, and u_4. The equuation 4.7 shows the operations to determine ρ values. For example, if we write the expectations for mode 0, in terms of correlations (since all pixels are zero mean), 4.8 is obtained.

$$\begin{bmatrix}\rho_1\\ \rho_2\\ \rho_3\\ \rho_4\end{bmatrix} = E\left[\begin{bmatrix}u_1\\ u_2\\ u_3\\ u_4\end{bmatrix}\begin{bmatrix}u_1 & u_2 & u_3 & u_4\end{bmatrix}\right]^{-1}\begin{bmatrix}E[u_0u_1]\\ E[u_0u_2]\\ E[u_0u_3]\\ E[u_0u_4]\end{bmatrix} \tag{4.7}$$

$$\begin{bmatrix}\rho_1\\ \rho_2\\ \rho_3\\ \rho_4\end{bmatrix} = \begin{bmatrix}1 & C_y & C_n & C_xC_n\\ C_y & 1 & C_x & C_x^2\\ C_n & C_x & 1 & C_x\\ C_xC_n & C_x^2 & C_x & 1\end{bmatrix}^{-1} \begin{bmatrix}C_x\\ C_y\\ C_m\\ C_n\end{bmatrix} \tag{4.8}$$

The correlation values that are used to obtain prediction parameters and transforms are given in Table 4.2. Those values are substituted in equation 4.8 to obtain prediction parameters.

Table4.2: C_x, C_y, C_m, and C_n parameters for 4x4 blocks used to obtain Markov-based intra predictions and transforms [26]

Prediction Mode	C_x	C_y	C_m	C_n
0	0.06	0.97	0.06	0.05
1	0.98	0.20	0.20	0.20
2	0.58	0.67	0.37	0.41
3	0.25	0.27	0.00	0.40
4	0.47	0.43	0.68	0.09
5	0.22	0.64	0.48	0.02
6	0.50	0.26	0.38	0.00
7	0.34	0.61	0.01	0.43
8	0.55	0.40	0.01	0.55

4.3.2 Derivation of Karhunen-Loeve Transform Matrices From Correlations and Markov-Based Intra Prediction Parameters

The basis vectors of autocorrelation matrix of residual error will give the Karhunen-Loeve Transform for each mode. In order to obtain KLT, we need to find signal characteristics of residual error. The residual error simply is the difference between actual pixel value and its prediction as seen in Equation 4.9.

$$e(i,j) = x(i,j) - \hat{x}(i,j) \tag{4.9}$$

where

$e(i,j)$ is the residual error

$x(i,j)$ is the original pixel value

$\hat{x}(i,j)$ is predicted pixel value

By substituting predicted pixel value with multiplication of neighbors and intra prediction parameters, we can write the residual error as in Equation 4.10

$$e(i,j) = \begin{bmatrix} 1 & -\bar{w}(i,j) \end{bmatrix} \begin{bmatrix} x(i,j) \\ \bar{n} \end{bmatrix} \tag{4.10}$$

where

$\bar{w}(i,j)$ is the weight vector for the current pixel (i,j)

$\bar{n}$ is the vector consisting of neighbor pixels

The autocorrelation matrix of residual errors is of size 16x16 for 4x4 blocks and is given in Equation 4.11

$$R_e = E \begin{bmatrix} e(1,1)e(1,1) & e(1,1)e(1,2) & .. & e(1,1)e(2,1) & .. & e(1,1)e(4,4) \\ e(1,2)e(1,1) & e(1,2)e(1,2) & .. & e(1,2)e(2,1) & .. & e(1,2)e(4,4) \\ : & .. & : & ... & ... & : \\ e(2,1)e(1,1) & e(2,1)e(1,2) & .. & e(2,1)e(2,1) & .. & e(2,1)e(4,4) \\ : & .. & : & ... & ... & : \\ e(4,4)e(1,1) & e(4,4)e(1,2) & .. & e(1,1)e(2,1) & .. & e(4,4)e(4,4) \end{bmatrix} \tag{4.11}$$

Each element of autocorrelation matrix is calculated using the formula given in Equation 4.12. Here, we found the expected value of correlation between residual errors at any location (i,j) and (k,l).

$$E\begin{bmatrix} e(i,j)e(k,l) \end{bmatrix} = E\begin{bmatrix} \begin{bmatrix} 1 & -\bar{w}(i,j) \end{bmatrix} \begin{bmatrix} x(i,j) \\ \bar{n} \end{bmatrix} \begin{bmatrix} x(k,l) & \bar{n} \end{bmatrix}^T \begin{bmatrix} 1 & -\bar{w}(k,l) \end{bmatrix}^T \end{bmatrix} \tag{4.12}$$

Correlations are obtained in terms of correlations of original pixels $x(i,j)$ and $x(k,l)$ when calculating 4.12. Those correlation values are listed in Table 4.2.

Non-Separable Karhunen-Loeve Transform is obtained by finding and ordering eigenvectors (basis vectors) of autocorrelation of residual error. Eigenvalues of the autocorrelation function give the scan order of transform coefficients. The highest eigenvalue will be the one first scanned and lowest eigenvalue will be the one last scanned.

4.4 Transform Algorithms and Scan Order

In this thesis, 2D transformation is obtained via transforming residuals with two 1D transformation in vertical and horizontal directions. 2D separable transform is also the preferred method in default H.264 implementation. By separating 2D transform into two 1D transforms, the freedom of applying different transforms in different directions is attained. The transforms used in the thesis have different characteristics. 5 different hybrid transform schemes are implemented named option0, option1, option2, option3 and option4. Their implementation details, relevant scanning orders and purposes are given in the following sub-sections.

4.4.1 Option 0 - Default Intra Prediction and Default DCT Transform

Option 0 includes algorithms that are used in default H.264 standard. In this option, default directional intra prediction consisting of nine modes are applied. Two 1D directional DCT, resulting in 2D separable DCT, are applied to intra prediction residuals both in vertical and horizontal directions. Then zig-zag scanning is employed to transforms coefficients.

The transform pairs used in this option are given in Table 4.3. As seen in the table, all transforms are DCT as in H.264 standard.

Table4.3: Transforms and Scan Orders used for option 0

Prediction Mode	Vertical Transform	Horizontal Transform	Scan Order
0, 1, 2, 3, 4, 5, 6, 7, 8	DCT	DCT	Zig-Zag

For the scanning in frame mode, zig-zag scanning order is used in H.264. Therefore, option 0 uses the same scanning order as seen in Table 4.4.

Table4.4: Option 0 scan order - This is the zig-zag scanning order used in default H.264 implementation

1	2	6	7
3	5	8	13
4	9	12	14
10	11	15	16

The purpose of option 0 is to build a reference prediction, transform, and scanning implementation for comparison with other options. This option will be used to determine advantage and disadvantage of newly introduced transformations.

4.4.2 Option 1 - Default Intra Prediction and ADST/DCT Transform

Option 1 implements novel ADST/DCT transform pair over default H.264 intra prediction. According to prediction mode, a combination of ADST and DCT one dimensional transforms are used in this option. The scanning order also changes with respect to prediction mode.

[56] explains ADST/DCT implementation and scan orders. In this option, either ADST or DCT is used in a given direction according to the prediction mode. Scan orders are chosen to minimize bit rate of the compressed transform coefficients according to the statistics of the residuals resulting from prediction and transformation.

The 1D transformation pairs that are used in option 1 are given in Table 4.5.

Table4.5: Transforms and Scan Orders used for option 1

Prediction Mode	Vertical Transform	Horizontal Transform	Scan Order
0	ADST	DCT	Horizontal
1	DCT	ADST	Vertical
2	DCT	DCT	Down-Left
3	ADST	DCT	Down-Left
4	ADST	ADST	Up-Right
5	ADST	ADST	Horizontal
6	ADST	ADST	Vertical
7	ADST	DCT	Horizontal
8	DCT	ADST	Vertical

The scan orders used for option 1 (horizontal, vertical, down-left, and up-right) are given in Table 4.6, 4.7, 4.8, and 4.9

Table4.6: Horizontal Scan Order

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

There are four different combinations of ADST/DCT transform pair. The basis functions of 2D transforms used for ADST/DCT transform are given in Figure 4.2

Table4.7: Vertical Scan Order

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Table4.8: Down-Left Scan Order

1	2	4	7
3	5	8	11
6	9	12	14
10	13	15	16

Table4.9: Up-Right Scan Order

1	3	6	10
2	5	9	13
4	8	12	15
7	11	14	16

4.4.3 Option 2 - Markov-Based Intra Prediction and Default DCT Transform

Option 2 is the modified version of default H.264 implementation with Markov-based intra prediction. Transforms and scan orders of H.264 left unmodified.

In Markov-based intra prediction implementation, four independent prediction parameters are used. The values for those intra prediction parameters are given in Table 4.10. Nine intra prediction directions of H.264 are modified to use those independent prediction parameters in a Markov-based prediction model.

The transforms used in option 2 are given in Table 4.11. The zig-zag scan given in Table 4.4 is used in option 2 as well.

4.4.4 Option 3 - Markov-Based Intra Prediction and ADST/DCT Transform

Option 3 combines Markov-based intra prediction and ADST/DCT transform implementations. Since Markov-based intra prediction and ADST/DCT transformations

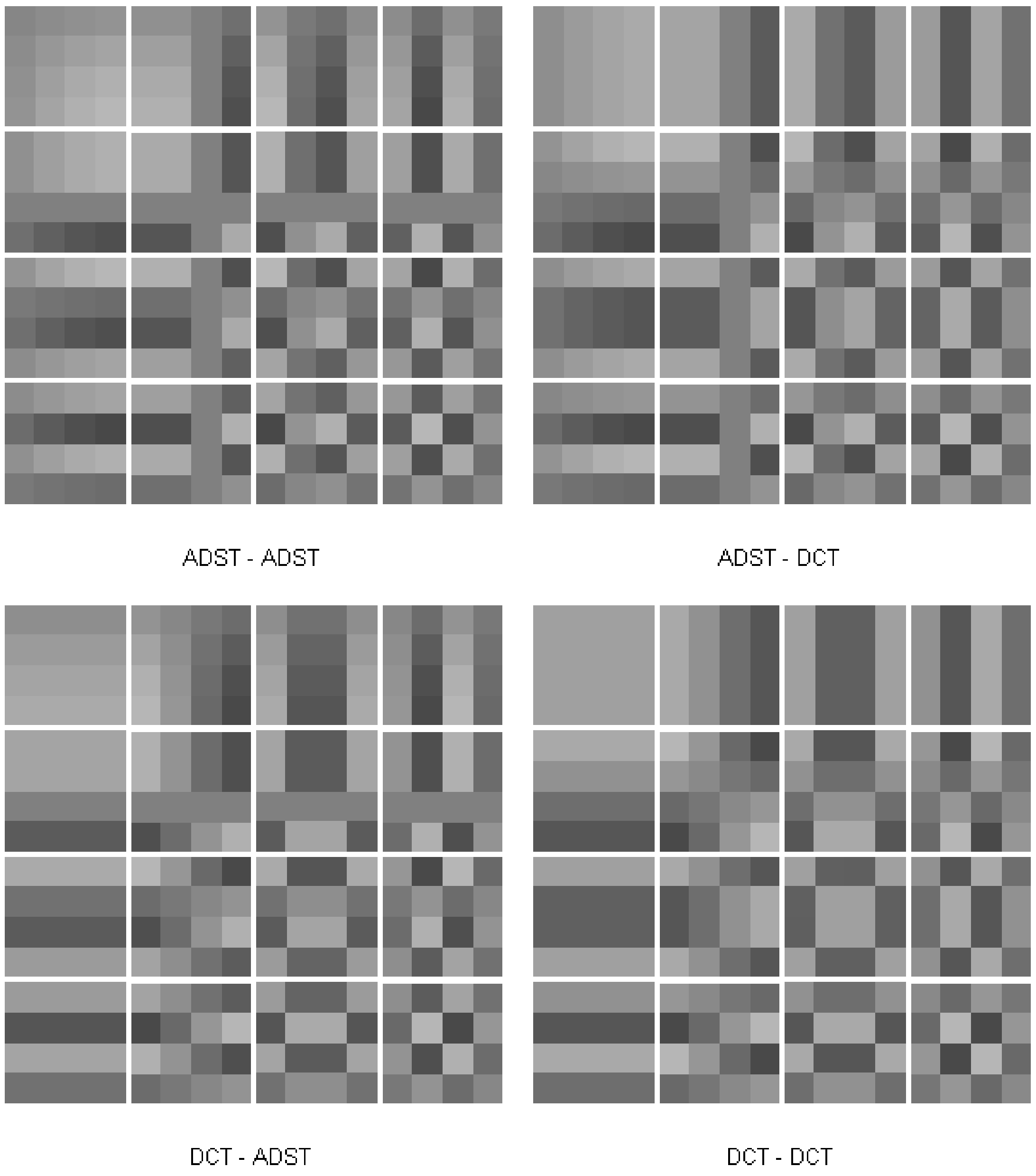


Figure 4.2: ADST-DCT implementation basis functions.

make use of directionality of prediction modes, their combination is expected to increase compression performance.

The four independent prediction parameters used in option 3 are the same as given in Table 4.10.

The transformations used in option 3 are given in Table 4.12. Horizontal, vertical, down-left, and up-right scan orders are given in Table 4.6, Table 4.7, Table 4.8, and Table 4.9.

Table4.10: ρ_1, ρ_2, ρ_3, and ρ_4 parameters for 4x4 blocks estimated from the training set in Kamisli's work [26]

Prediction Mode	ρ_1	ρ_2	ρ_3	ρ_4
0	0.12	-0.09	0.98	0.00
1	0.98	-0.20	0.23	0.00
2	0.55	-0.44	0.61	0.00
3	0.17	0.11	0.36	0.39
4	0.31	0.56	0.31	-0.08
5	0.05	0.40	0.66	-0.05
6	0.61	0.33	0.21	-0.10
7	0.38	-0.31	0.59	0.33
8	0.44	-0.30	0.39	0.47

Table4.11: Transforms used for option 2

Prediction Mode	Vertical Transform	Horizontal Transform	Scan Order
0, 1, 2, 3, 4, 5, 6, 7, 8	DCT	DCT	Zig-Zag

Table4.12: Transforms and Scan Orders used for option 3

Prediction Mode	Vertical Transform	Horizontal Transform	Scan Order
0	ADST	DCT	Horizontal
1	DCT	ADST	Vertical
2	DCT	DCT	Down-Left
3	ADST	DCT	Down-Left
4	ADST	ADST	Up-Right
5	ADST	ADST	Horizontal
6	ADST	ADST	Vertical
7	ADST	DCT	Horizontal
8	DCT	ADST	Vertical

4.4.5 Option 4 - Markov-Based Intra Prediction and KLT Transform

Option 4 implements 2D non-separable KLT transform for all directions over Markov-based intra prediction. The KLT transform matrix for each mode is different and calculated according to Kamisli's Markov prediction parameters at the given directions which is given in Table 4.10. It is expected that KLT transform should give the best results since it is the optimal transform.

Figure 4.3 displays the basis functions of KLT transforms for each intra prediction mode. Basis function patterns (especially first few basis functions) are compatible with the intra prediction directions, and shows a directional pattern.

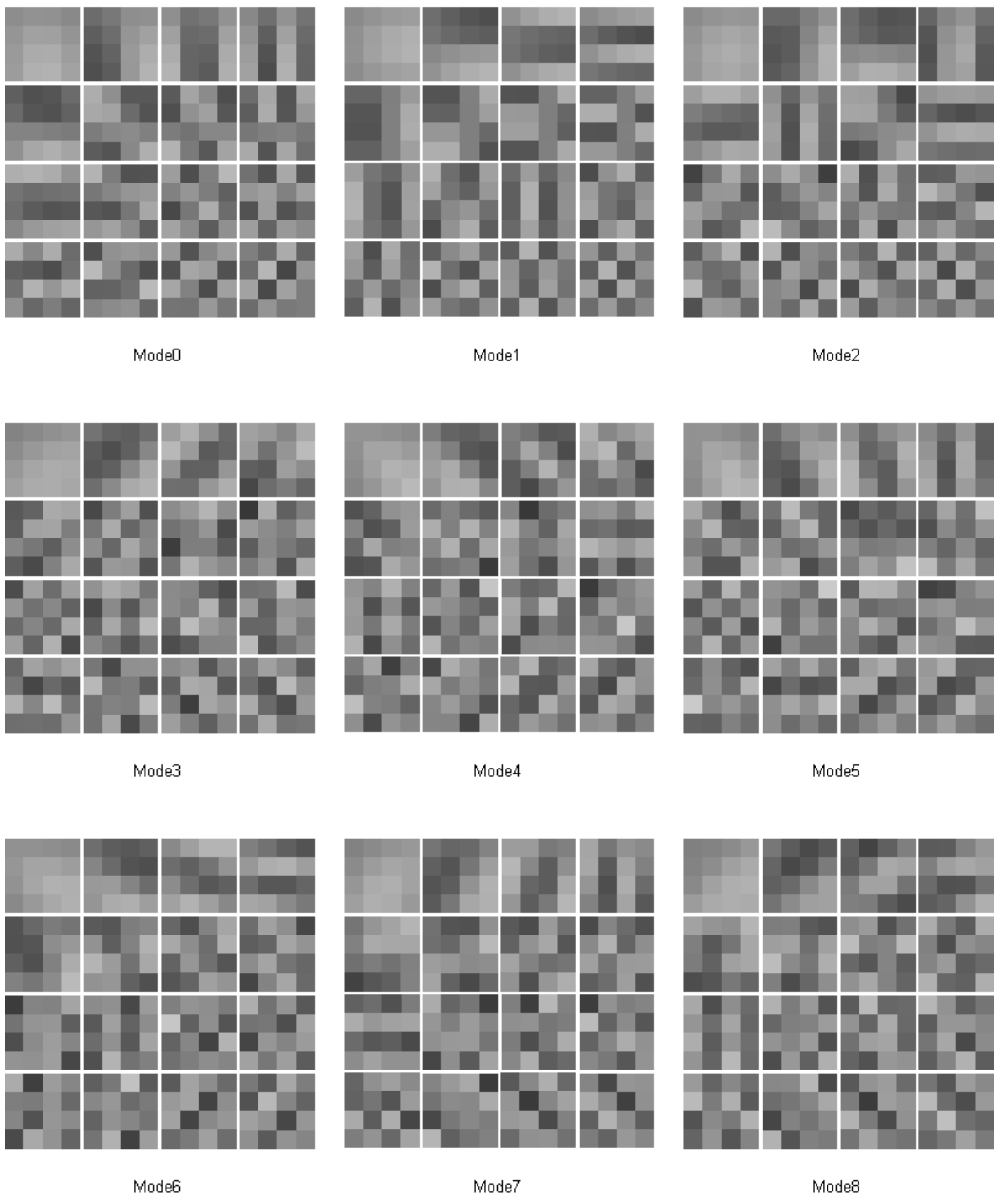


Figure 4.3: KLT basis functions are shown according to the intra prediction mode.

The transforms used in Option 4 are given in Table 4.13.

Basis vectors are ordered according to eigenvalues in descending manner, therefore all scan orders are horizontal (the same as given in Table 4.6).

Table4.13: Transforms and Scan Orders used for option 4

Prediction Mode	2D Non-Separable Transform	Scan Order
0, 1, 2, 3, 4, 5, 6, 7, 8	2D Non-Separable KLT	Horizontal

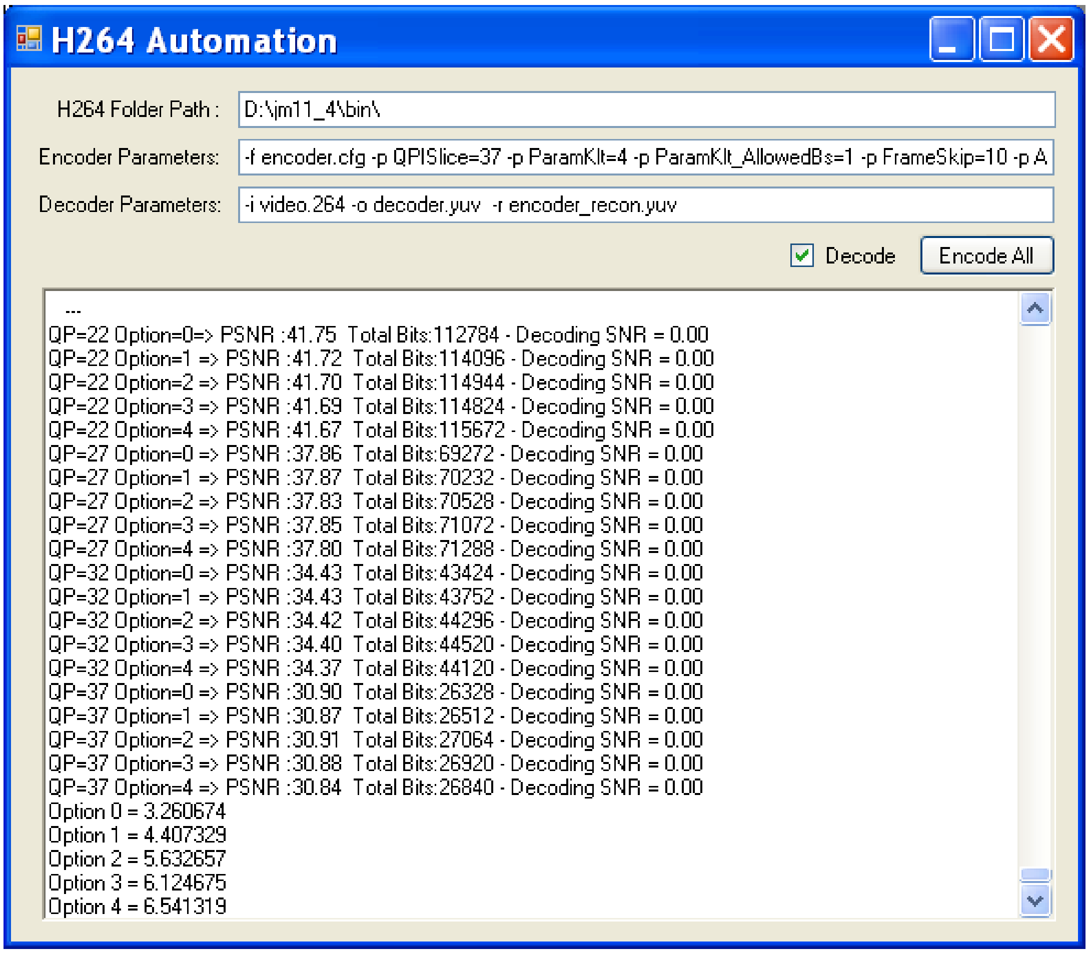
4.5 Software Implementation Details

JM reference software is updated without harming the integrity of its original implementation. New compile-time keywords are defined and new .c files are created for the brand new implementation. Transformation and scanning order parts of JM reference software are branched with those new defined compile-time keywords and redirected to newly created .c file that is special to transform algorithms and scanning orders in this thesis. The implementation is done in C++ which is JM reference software's implementation language.

Encoder software is run by parameterizing its console line input. In the encoder console input, new parameters that specify the use of new algorithm and option selection are defined. At each run of encoder, these new parameters and input video are supplied to the encoder. The console output of encoder gives the success status of encoding operation, the bit rate ratio and PSNR of the newly encoded video. Those console line outputs are the main results and performance metrics of this thesis' algorithms. Those performance metrics are supplied as a standard by JM reference software and gives opportunity to different researchers to compare their results.

The decoder is also updated corresponding to the algorithm implemented in encoder. The encoder and decoder should be in exact harmony in order to conclude that the asserted algorithms working correctly and gives satisfying results. Within the scope of this work, the encoded bit stream file is submitted to decoder, and decoded video is obtained and viewed to verify the correct operation. The verification is done for each option separately and correct operation is guaranteed.

In this thesis, there are many transformation options and several test videos. In each video, there are lots of frames to be encoded. The combination of these several options hardens the result collection part by hand. For this purpose, a C# program is coded. The C# program asks for encoder and decoder path. It supplies all the options needed by encoder and decoder to console line one at a time for each video and option. After each encoding for one specific option and video, the decoder is run for the same bit stream, and the correctness of operation is verified automatically, too. The c# program writes the result both on the screen and to a log file. At the end of this automated process, bit rate savings of each option are compared with each other with an algorithm that compensates bit rate-PSNR difference and bring all to an equal scale. The details of comparison algorithm are given later in this thesis. A screenshot of the C# program is given in Figure 4.4



H264 Folder Path : D:\jm11_4\bin\

Encoder Parameters: -f encoder.cfg -p QPISlice=37 -p ParamKlt=4 -p ParamKlt_AllowedBs=1 -p FrameSkip=10 -p A

Decoder Parameters: -i video.264 -o decoder.yuv -r encoder_recon.yuv

Decode [✓]

```
 ...
QP=22 Option=0=> PSNR :41.75  Total Bits:112784 - Decoding SNR = 0.00
QP=22 Option=1 => PSNR :41.72  Total Bits:114096 - Decoding SNR = 0.00
QP=22 Option=2 => PSNR :41.70  Total Bits:114944 - Decoding SNR = 0.00
QP=22 Option=3 => PSNR :41.69  Total Bits:114824 - Decoding SNR = 0.00
QP=22 Option=4 => PSNR :41.67  Total Bits:115672 - Decoding SNR = 0.00
QP=27 Option=0 => PSNR :37.86  Total Bits:69272 - Decoding SNR = 0.00
QP=27 Option=1 => PSNR :37.87  Total Bits:70232 - Decoding SNR = 0.00
QP=27 Option=2 => PSNR :37.83  Total Bits:70528 - Decoding SNR = 0.00
QP=27 Option=3 => PSNR :37.85  Total Bits:71072 - Decoding SNR = 0.00
QP=27 Option=4 => PSNR :37.80  Total Bits:71288 - Decoding SNR = 0.00
QP=32 Option=0 => PSNR :34.43  Total Bits:43424 - Decoding SNR = 0.00
QP=32 Option=1 => PSNR :34.43  Total Bits:43752 - Decoding SNR = 0.00
QP=32 Option=2 => PSNR :34.42  Total Bits:44296 - Decoding SNR = 0.00
QP=32 Option=3 => PSNR :34.40  Total Bits:44520 - Decoding SNR = 0.00
QP=32 Option=4 => PSNR :34.37  Total Bits:44120 - Decoding SNR = 0.00
QP=37 Option=0 => PSNR :30.90  Total Bits:26328 - Decoding SNR = 0.00
QP=37 Option=1 => PSNR :30.87  Total Bits:26512 - Decoding SNR = 0.00
QP=37 Option=2 => PSNR :30.91  Total Bits:27064 - Decoding SNR = 0.00
QP=37 Option=3 => PSNR :30.88  Total Bits:26920 - Decoding SNR = 0.00
QP=37 Option=4 => PSNR :30.84  Total Bits:26840 - Decoding SNR = 0.00
Option 0 = 3.260674
Option 1 = 4.407329
Option 2 = 5.632657
Option 3 = 6.124675
Option 4 = 6.541319
```

Figure 4.4: The C# program that automates the process of encoding and decoding many options, videos and frames with one click. It makes the result collection very easier.

Although it is possible to specify quantization parameter, it is not possible to obtain the same bit count for all encoded streams. In order to compare all transformations with each other, we have to come up with a solution that makes us possible to compare results. Luckily, average SNR difference calculator is implemented by Telenor [1] and the source code of this program is given in Appendix A. This program compares two video coding algorithms at four different points, in our specific case; the algorithms are compared at quantization parameters 22, 27, 32 and 37.

The test videos used in this thesis are given in Table 4.14. Every 10th frame of sequences are coded in this work, and the average bit rate savings values are given in Chapter 5.

Table4.14: Test Videos used to evaluate transforms and scanning orders

Video Number	Video Name-Resolution-Hz
1	BasketballPass-416x240-50
2	BlowingBubbles-416x240-50
3	RaceHorses-416x240-30
4	BasketballDrill-832x480-50
5	BQMall-832x480-60
6	RaceHorses-832x480-30
7	vidyo3-720p-60
8	vidyo4-720p-60
9	BasketballDrive-1920x1080-50
10	BQTerrace-1920x1080-60
11	Cactus-1920x1080-50
12	ParkScene-1920x1080-24

In JM software implementation, encoder and decoder takes several parameters to use different options/tools for encoding/decoding. The used options are given as following:

- Rate distortion optimization is enabled for intra prediction. It requires much computation, however it gives the best results.
- In-loop deblocking filter is activated during encoding.
- Due to higher compression ratio, CABAC entropy coding is used in the implementation.
- Only 4x4 blocks are coded.
- Luma component is modified. Chroma components of frames left unmodified at their original status in JM reference implementation.
- Ten frames of each video sequence is coded and their average result is obtained.
- Four different quantization parameters (22, 27, 32 and 37) are used in the implementation.
- All new tools and features of H.264 are deactivated in this study to prevent complication.

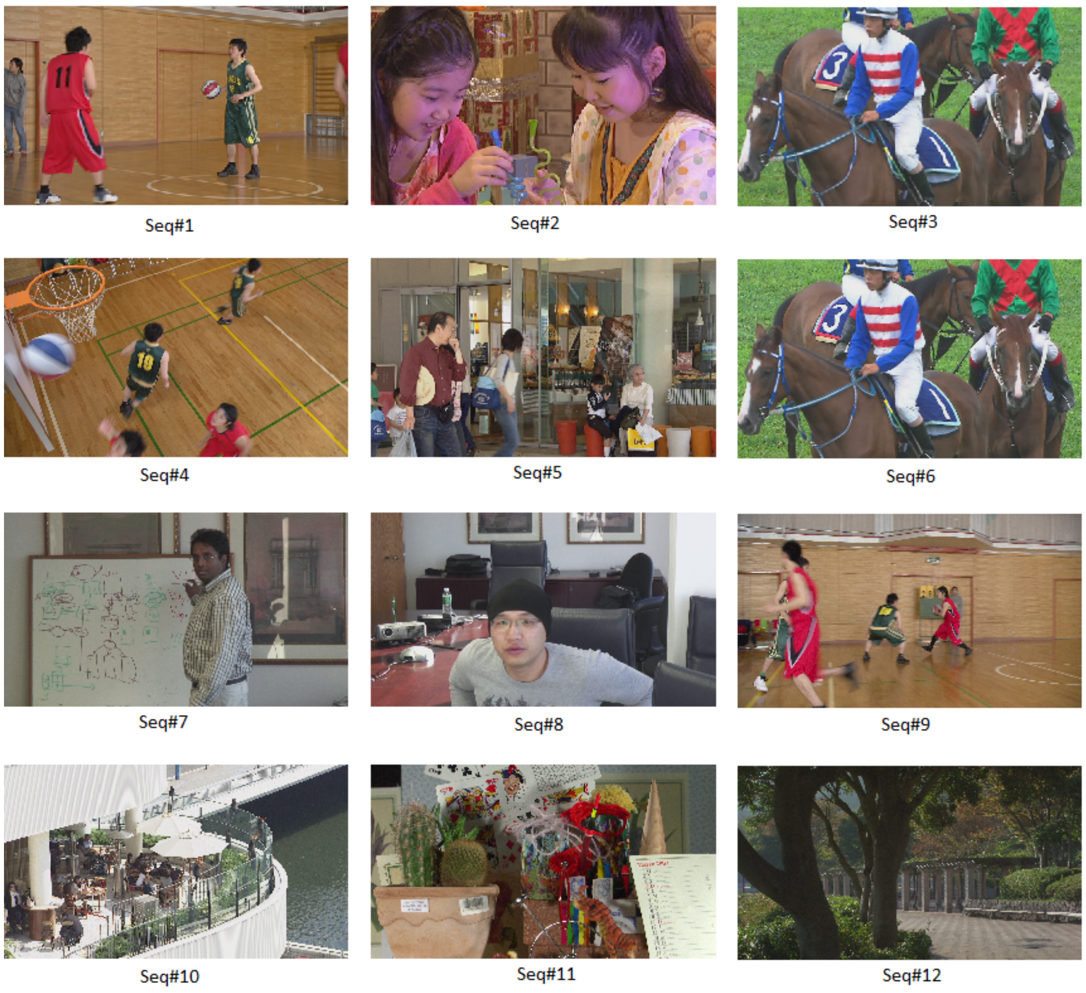


Figure 4.5: Sample images from the video sequences used in the experiments

CHAPTER 5

RESULTS

Results of this study are presented in following sections with corresponding figures. "Option 0 - Default Intra Prediction and Default DCT Transform" is used as reference to other transforms. As mentioned in the previous chapters, option 0 is the default H.264 intra prediction, DCT transform and zig-zag scan. Option 0 is assumed as reference result in all comparisons and the results of other options are given as percentage over option 0.

The bit rate savings are given using Bjontegaard-Delta bit rate metric [2] for the luma component. BD-BR metric is a widely used technique for average percentage bit rate saving comparison in video compression. In addition, PSNR-Bitrate curves are given for a few video samples.

As mentioned in the previous chapters, we have only modified 4x4 blocks, the remaining intra prediction modes, transforms and scan orders left unmodified at their default status in H.264 JM Reference Software Implementation. In addition, the modifications are only applied to luma components only. Bit rate savings are given for 4x4 luma blocks in following figures.

In the following figures, 12 test video sequences given in Table 4.14 are used.

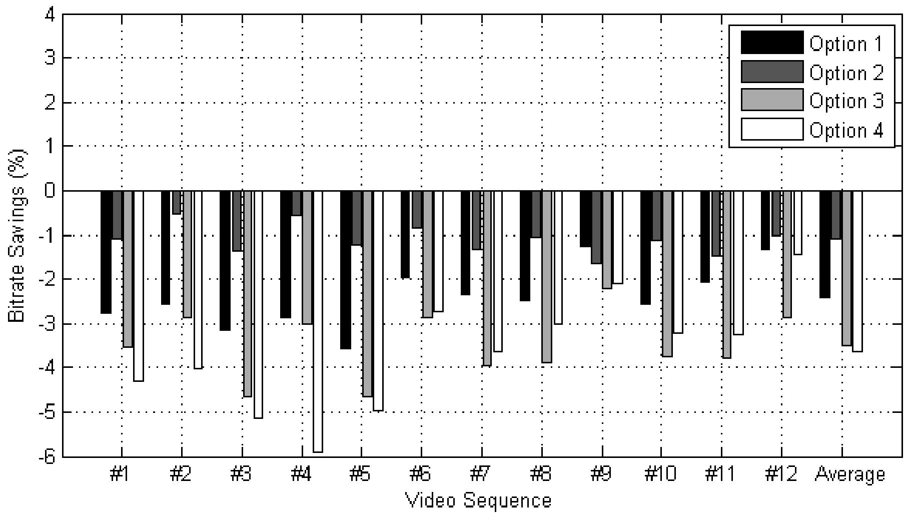


Figure 5.1: Percentage bit rate savings for options 1, 2, 3 and 4 over option 0 (default H.264 implementation)

Table5.1: BD-BR percentage (%) bitrate savings of options 1, 2, 3 and 4 with respect to option 0 (default H.264 implementation). Negative number indicates bitrate reduction with respect to H.264

Seq#	Sequence Name	Option 1	Option 2	Option 3	Option 4
1	BasketballPass-416x240-50	-2.78	-1.09	-3.52	-4.31
2	BlowingBubbles-416x240-50	-2.56	-0.52	-2.88	-4.01
3	RaceHorses-416x240-30	-3.16	-1.38	-4.64	-5.15
4	BasketballDrill-832x480-50	-2.88	-0.55	-3.01	-5.92
5	BQMall-832x480-60	-3.57	-1.22	-4.64	-4.97
6	RaceHorses-832x480-30	-1.98	-0.85	-2.87	-2.74
7	vidyo3-720p-60	-2.36	-1.32	-3.95	-3.65
8	vidyo4-720p-60	-2.48	-1.06	-3.87	-2.99
9	BasketballDrive-1920x1080-50	-1.27	-1.63	-2.21	-2.11
10	BQTerrace-1920x1080-60	-2.57	-1.13	-3.75	-3.20
11	Cactus-1920x1080-50	-2.07	-1.46	-3.77	-3.25
12	ParkScene-1920x1080-24	-1.35	-1.46	-2.87	-1.42
	AVERAGE	-2.42	-1.10	-3.50	-3.65

5.1 Compression Performance Comparison of Options

Option 1 is the implementation of ADST/DCT transform over default H.264 intra prediction. As seen in the results, option 1 increases the performance of compression. This option shows that ADST transform is the suitable transform when correlation in the given direction is high. Also, the default H.264 transform which is DCT transform gives good results if the correlation between pixels are average or unknown. In this option, choice of ADST and DCT changes depending on the prediction mode.

Option 2 implements Markov-process-based intra prediction with default H.264 DCT transform. Markov-process-based intra prediction carries simple copying of elements one step ahead, and makes use of other neighbors. This approach increases compression performance since it also uses the low correlation between the current pixel and neighbors.

Option 3 is the combination of Markov-process-based intra prediction and ADST/DCT hybrid transform. Since both algorithms take advantage of directionality, their combination gives better results than the option that use only Markov-based intra prediction or ADST/DCT hybrid transform. This option is the first to combine two novel algorithms in one implementation.

Option 4 is the best performing option. It makes use of optimal KLT transform specific to the used intra prediction parameters. Since, KLT is non-separable, it covers the directions that are not horizontal or vertical (such as diagonal) better than separable transforms. However, the difference between ADST/DCT and KLT is very low. The similar result can be observed in [56].

Options generally give better bitrate savings at lower resolutions. An example of this scenario is that sequence 3 and sequence 6 are the different resolutions of the same scene. The results of sequence 3 which is lower resolution is better than the results of sequence 6. As resolution increases, bitrate savings decrease. The reason is that only 4x4 block are modified in this thesis, 8x8 blocks are left unchanged. Since higher resolution videos have higher number of 8x8 blocks which are intra coded by default algorithms of H.264, bitrate savings are close to default H.264 implementation.

The interpretation of the results shows that different transforms perform better for different intra prediction modes since image residuals have different characteristics for different modes. For high correlation coefficients, ADST is the best transform of choice in the given direction. DCT is the best transform when either correlation is not known or is not very high.

Transformation types and correlation coefficients have huge impact on scan orders. Different transformations gather coefficients in different orders. To obtain the best bit rate savings, transform coefficients should be ordered in descending fashion. In this thesis, scan orders are derived and implemented in JM reference software.

Asymmetric discrete sine transform is used when the correlation between boundary pixels and block pixels are close to one. Since boundary and block pixels are highly correlated, the artifacts at the block edges are very small. The decrease in the artifacts is another advantage of ADST over DCT.

5.2 PSNR - Bitrate Curves

PSNR - Bitrate curves are given for video sequences 1, 4, 8, and 12 in Figure 5.2, 5.3, 5.4, and 5.5 respectively. The values for the curves are calculated at QP = 22, 27, 32, and 37. Higher quantization parameter means lower bitrate and lower quantization parameter means higher bitrate.

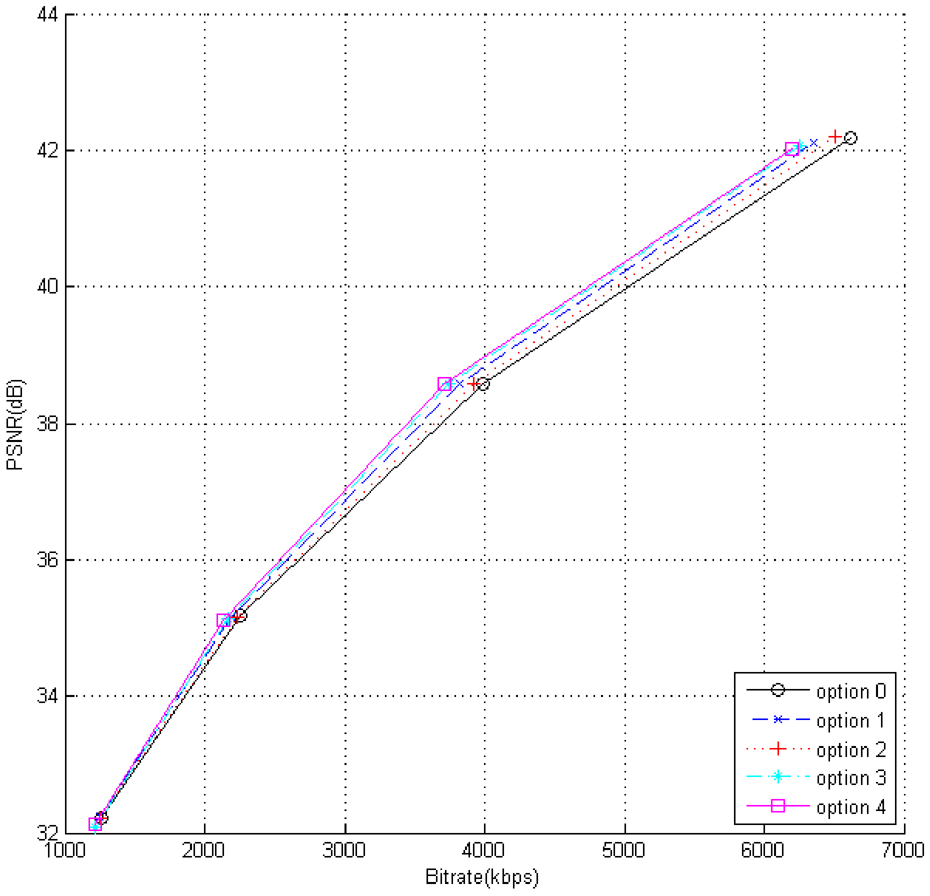


Figure 5.2: PSNR (dB) - Bitrate (kbps) curve for sequence 1

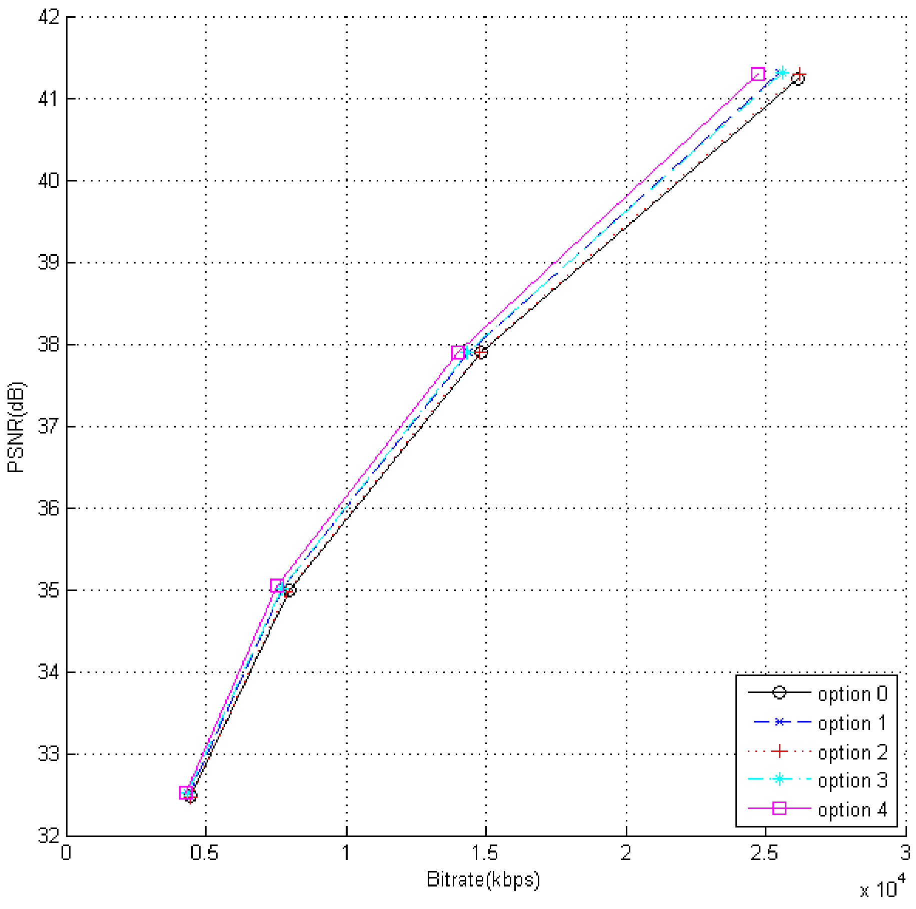


Figure 5.3: PSNR (dB) - Bitrate (kbps) curve for sequence 4

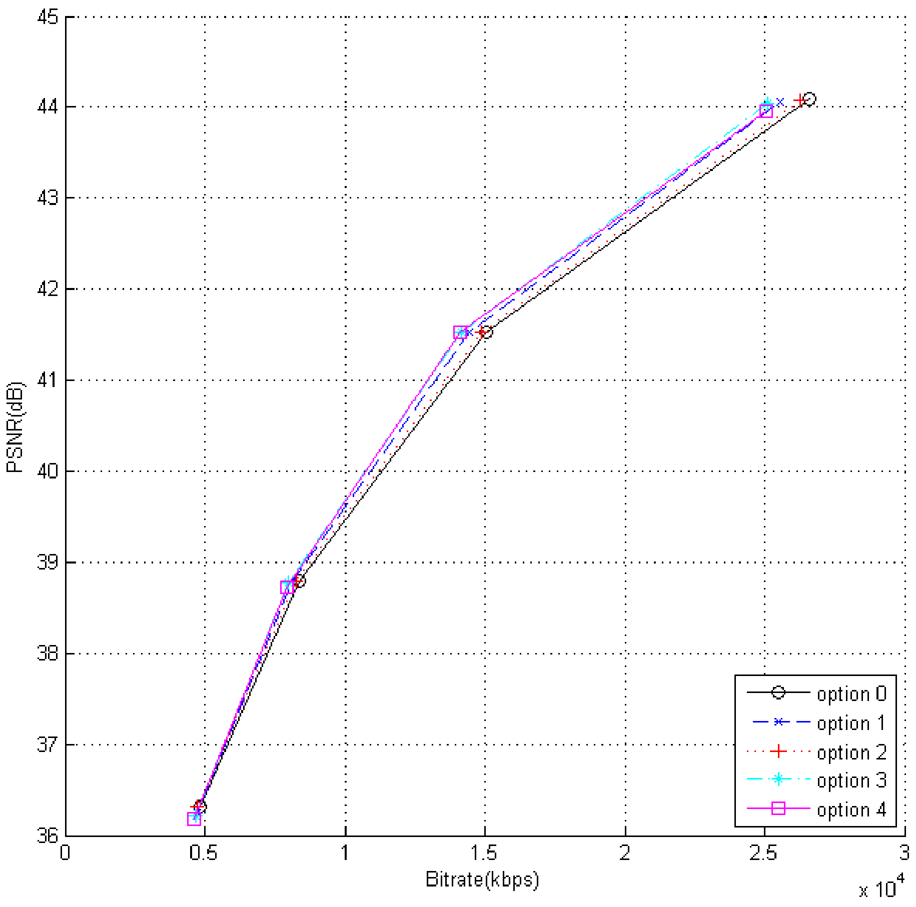


Figure 5.4: PSNR (dB) - Bitrate (kbps) curve for sequence 8

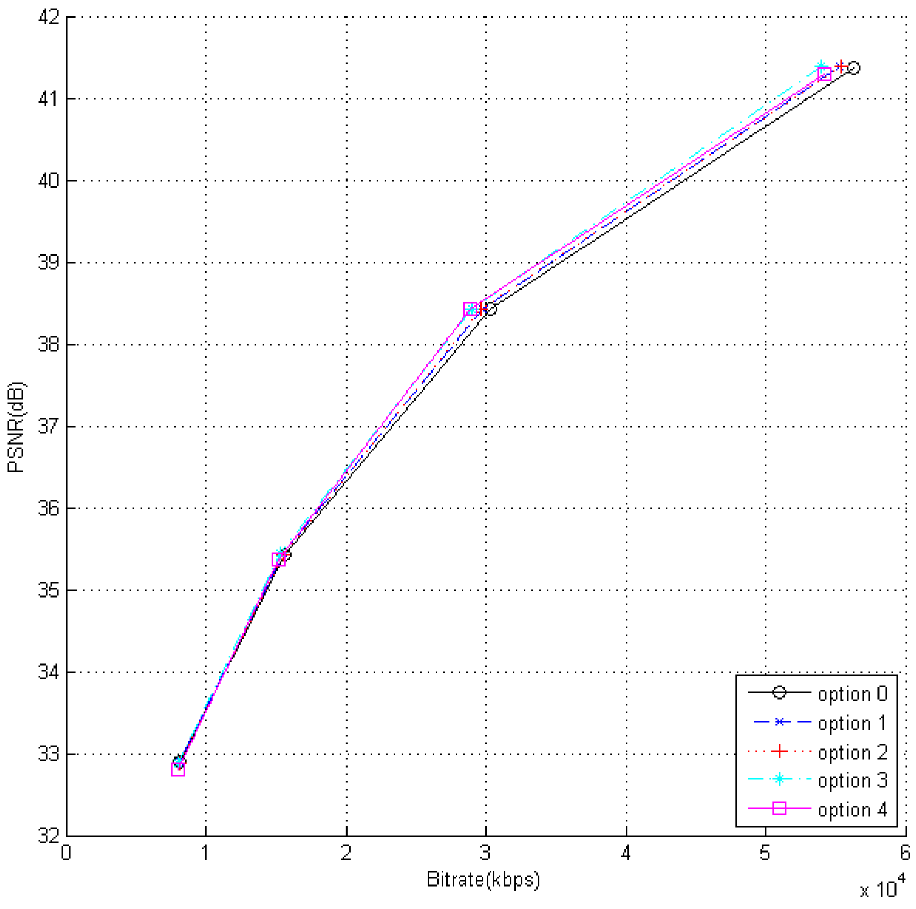


Figure 5.5: PSNR (dB) - Bitrate (kbps) curve for sequence 12

As seen in the PSNR - Bitrate figures, the effect of video sequence on curves are minimal. Options generally displays consistent and similar characteristics at different bitrates. If an option performs better for one QP value, it also generally performs better for other qp values. At higher bitrates, the difference between options are slightly clearer. The only exception is that, for video sequence 8 and 12, option 3 performs better at higher bitrates whereas it performs close to option 4 at lower bitrates.

5.3 Complexity Comparison of Algorithms

Various transform algorithms are used in this thesis. Some transforms have integer transform counterparts which reduces the computation times. Separable transforms are generally faster than non-separable transforms.

In our case, DCT has integer transform equivalent and it is mode independent, therefore it has the fastest implementation. ADST/DCT pair is mode dependent, switching between transforms require comparison which increases computation time. ADST has also equivalent integer transform, and ADST/DCT pair is expected to perform close to default DCT implementation. Non-separable KLT is the slowest transform, since it requires matrix multiplication which includes a lot of multiplications and additions.

Markov-based intra prediction encoder is 8.4% slower than default H.264 intra prediction [26]. Therefore, it is expected that options that include Markov-based intra prediction should be slower than those that does not include Markov-based intra prediction.

Based on the above comments, expected complexity of options is (C indicates complexity): $C_{option4} > C_{option3} > C_{option2} > C_{option1} > C_{option0}$

All transforms (including separable ones) are implemented using matrix multiplication in non-separable form. Therefore, encoding times of transforms for all modified options are the same. Therefore, only computation times of option 0 (default H.264 implementation) and option 4 are given in Table 5.2.

Since KLT and ADST/DCT hybrid transform display close performance, ADST/DCT can be the preferred transform due to its low complexity. Also ADST is currently used by newly introduced video coding standards. Due to lower complexity of both algorithms, Markov-based intra prediction and ADST/DCT hybrid transform could find popular use in the future.

Table5.2: Comparison of average encoding and decoding times

Method	H.264 (base)	Option 4 (maximum)
Encoder	100.0%	122.2%
Decoder	100.0%	120.8%

5.4 Evaluation of Results for Best Performing and Worst Performing Video Sequences

Markov-based intra prediction and KLT are formed by correlations that are obtained by the four training sequences which are given in [26]. Therefore, the prediction parameters and transforms can change depending on training sequences. If the statistics of test sequences are not compatible with the statistics of training sequences, the compression performance may degrade due to the incompabilities in correlations. Hence, it is expected that Markov-based intra prediction and KLT perform better in some video sequences than the others.

Sequences 11 and 12 have very complex textures, and there are almost no directionality in the frames. Generic transforms like DCT and ADST are the best performers in such conditions. Therefore, KLT performance is lower in those two sequences as seen in Figure 5.6 and 5.7.

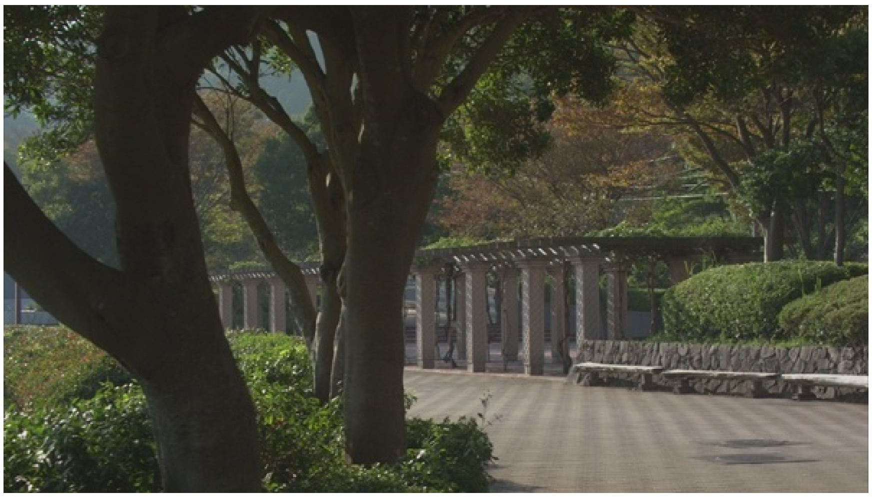


Figure 5.6: The Worst Performing Sequence (First image of seq#12)

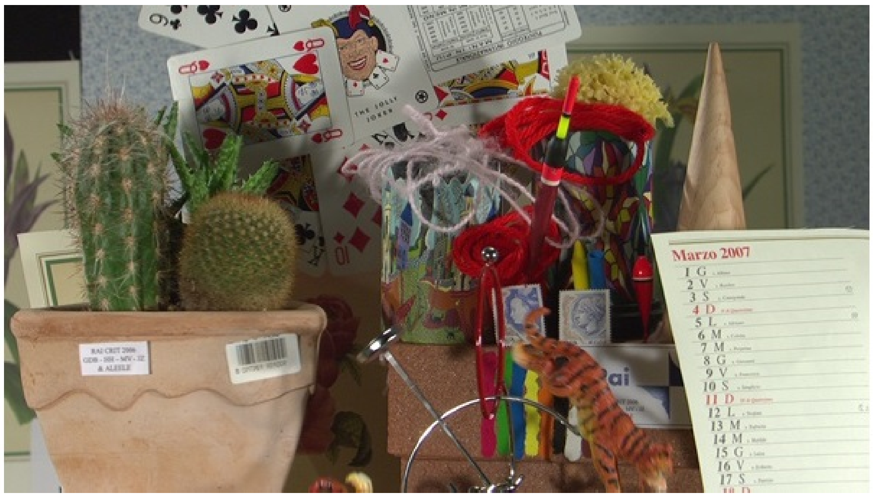


Figure 5.7: Another Bad Performing Sequence (First image of seq#11)

There are many horizontal and vertical lines in sequence 9. KLT transform is separable in horizontal and vertical mode and very close to ADST/DCT pair. Also, pure Markov-based intra prediction performs better in horizontal and vertical directions. Therefore, video sequence 9 gives very close results for all options as seen in Figure 5.8.

Figure 5.8: Close Performance by All Transforms (First image of seq#9)

Sequence 4 contains many diagonal lines. Sequence 4 is the best performer for option 4 - Markov-based intra prediction and KLT transform. Since KLT is non-separable, it captures the diagonal directionality information better than any other transform. Since video sequence 9 contains lots of intra prediction mode 4, the performance of KLT is much higher than the default H.264 implementation as seen in Figure 5.9

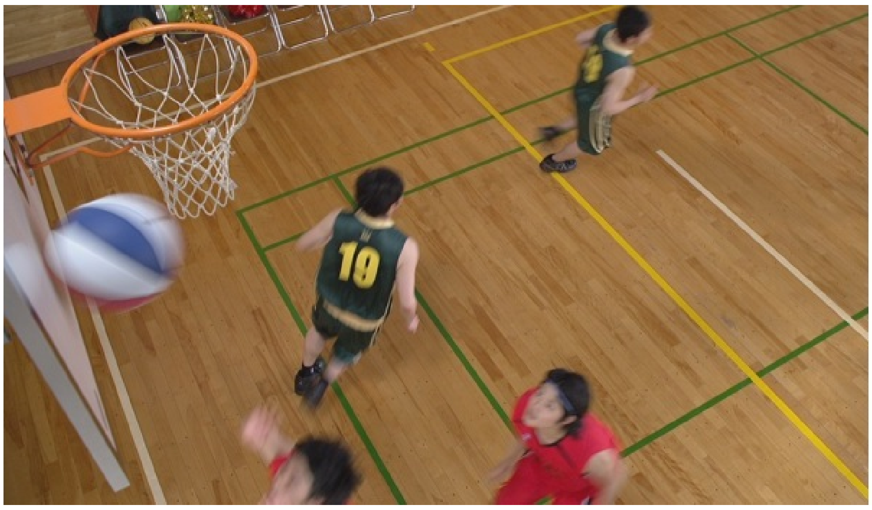


Figure 5.9: The Best Performing Sequence (First image of seq#4)

CHAPTER 6

CONCLUSION

In this thesis, transformation and scan order pairs for Markov based intra prediction in H.264 JM reference software are compared. The novel ADST transform and the novel Markov based intra prediction methods are combined together. The performance improvements are given in Chapter 5. The performance of Markov based intra prediction with ADST and optimal KLT scan is compared with default H.264 implementation. It has been shown that ADST transform is suitable for Markov based intra prediction. In addition, it has been shown that Markov based intra prediction with ADST transform provides high bit rate savings over standard H.264 implementation.

Five different options are compared, taking default H.264 implementation as reference. The results show that Markov-process-based intra prediction with KLT is the best performer. The performance of Markov-process-based intra prediction with ADST/DCT pair is very close to KLT implementation, and it can be the transform of choice because of its low complexity.

In Chapter 4, the derivation of scanning order is given. According to the results of this derivation, zig-zag scanning order is very close to optimal scanning order for DCT transformation. However, the scanning order of basis vectors for ADST/DCT and KLT transform should be changed since they present different coefficient characteristics. Ordering transform coefficients has high impact on bit rate savings.

The contribution of this study is that ADST, DCT and KLT transforms are first implemented for the novel Markov based intra prediction in H.264. This study suggest that ADST-DCT transform improves bit rate savings in H.264 in the case that Markov based intra prediction is used. When we consider that Markov based intra prediction alone improves compression over standard H.264, the combination of ADST/DCT and Markov based intra prediction further increases bit rate savings.

This study shows that ADST/DCT pair for Markov based intra prediction with offline correlation coefficients improves coding efficiency about 3.5%. It is possible to further increases coding efficiency by introducing adaptive correlation coefficient at run-time. Also, different scan order tables for different correlation coefficients and transforms can be constructed to further increase coding efficiency. However, these two methods also increase the complexity of H.264 encoder and decoder.

Options generally perform better at lower resolutions than higher resolutions in the test video sequences. The reason is that all blocks are enabled in H.264 implementation, however only 4x4 blocks are modified. Since encoder chooses more 8x8 blocks for high resolution images, the effect of modifications on videos decreases.

In this study, the pixels are modeled according to first order Markov model. However, real time scenarios include random correlations between pixels, sharp transitions, amplitude decays and rises. It is not possible to cover all these cases with low number of modes and transforms. Therefore, DCT is the low complexity average solution of all these cases in H.264 and in the literature.

The computation times of Markov-process-based intra prediction with KLT and default H.264 are also compared in this study. Since KLT transform have no fast implementation, it should be performed by matrix multiplication. KLT drastically increases computation times since it requires lots of multiplications and additions. ADST and DST transforms have integer implementations like DCT integer transform. Their counterpart integer transforms should perform similarly to DCT integer transform in terms of computation times. Therefore, KLT transform will increase computation time and computation complexity drastically. Due to this handicap of KLT, it is not employed in video standards. However, ADST/DCT pair can substitute default DCT implementation in video coding standards.

As future work, 8x8 blocks for Markov-based intra prediction and transforms will be included in comparisons. Separable KLT transforms for arbitrary angular directions are planned to be included in the comparisons since they present a low complexity solution compared to non-separable ones. The combined optimization of Markov-based intra prediction and transforms is planned to be implemented in order to increase compression performance further.

REFERENCES

[1] avsnr.c, average snr difference calculator, telenor broadband services, keysers gate 13, n-0130 oslo, norway. `http://www.telenor.com`.

[2] Calculation of average psnr differences between rd curves. *ITU-T Q.6/SG16, VCEG-M33, 2001.*

[3] H.264/avc jm reference software. `http://iphome.hhi.de/suehring/tml/`.

[4] H.264/mpeg-4 avc. `http://en.wikipedia.org/wiki/H.264`.

[5] Iso - international organization for standardization website. `http://www.iso.org`.

[6] Itu - international telecommunication union website. `http://www.itu.int`.

[7] Moving picture experts group website. `http://www.mpeg.org/`.

[8] M. Biswas, M. Pickering, and M. Frater. Improved h.264-based video coding using an adaptive transform. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 165–168, 2010.

[9] N. Brahimi and S. Bouguezel. An efficient fast integer dct transform for images compression with 16 additions only. In *Systems, Signal Processing and their Applications (WOSSPA), 2011 7th International Workshop on*, pages 71–74, 2011.

[10] J. Chen and W.-J. Han. Adaptive linear prediction for block-based lossy image coding. In *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pages 2833–2836, 2009.

[11] J.-W. Chen, C.-Y. Kao, and Y.-L. Lin. Introduction to h.264 advanced video coding. In *Design Automation, 2006. Asia and South Pacific Conference on*, pages 6 pp.–, 2006.

[12] W.-H. Chen, C. Smith, and S. Fralick. A fast computational algorithm for the discrete cosine transform. *Communications, IEEE Transactions on*, 25(9):1004–1009, 1977.

[13] A. B. Dinesh Kumar, Pavan Shastry. Overview of the h.264/avc. *8th Texas Instruments Developer Conference India*, 2005.

[14] M. Flierl and B. Girod. Generalized b pictures and the draft h.264/avc video-compression standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):587–597, 2003.

[15] F. Fu, X. Lin, and L. Xu. Fast intra prediction algorithm in h.264-avc. In *Signal Processing, 2004. Proceedings. ICSP '04. 2004 7th International Conference on*, volume 2, pages 1191–1194 vol.2, 2004.

[16] A. L. Gary J. Sullivan, Pankaj Topiwala. The h.264/avc advanced video coding standard: Overview and introduction to the fidelity range extensions. pages 186–216, 2006.

[17] J. Han, A. Saxena, V. Melkote, and K. Rose. Jointly optimized spatial prediction and block transform for video and image coding. *Image Processing, IEEE Transactions on*, 21(4):1874–1884, 2012.

[18] M. Isnardi. Historical overview of video compression in consumer electronic devices. pages 1–2, 2007.

[19] A. Jain. A fast karhunen-loeve transform for a class of random processes. *Communications, IEEE Transactions on*, 24(9):1023–1029, 1976.

[20] A. K. Jain. A sinusoidal family of unitary transforms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-1(4):356–365, 1979.

[21] L. Jianqiu, N. Rongrong, and R. Qiuqi. A wavelet based intra-frame coding algorithm focussing on face region. In *Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference on*, volume 2, pages 1101–1104 Vol.2, 2003.

[22] B. Juurlink. Understanding the application: An overview of the h.264 standard. In *Scalable Parallel Programming Applied to H.264/AVC Decoding, SpringerBriefs in Computer Science*, 2012.

[23] N. Kamaci and Y. Altunbasak. Performance comparison of the emerging h.264 video coding standard with the existing standards. In *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, volume 1, pages I–345–8 vol.1, 2003.

[24] F. Kamisli. Intra prediction based on statistical modeling of images. *Visual Communications and Image Processing (VCIP), 2012 IEEE*, pages 1–6, 2012.

[25] F. Kamisli. Intra prediction based on markov process modeling of images. *accepted to IEEE ICIP*, 2013.

[26] F. Kamisli. Intra prediction based on markov process modeling of images. *Image Processing, IEEE Transactions on*, 22(10):3916–3925, 2013.

[27] F. Kamisli and J. Lim. 1-d transforms for the motion compensation residual. *Image Processing, IEEE Transactions on*, 20(4):1036–1046, 2011.

[28] Y. Kim, M. Kang, C. Kim, and J.-M. Kim. Optimal multi-core processors for fast discrete cosine transform. In *Strategic Technology (IFOST), 2011 6th International Forum on*, volume 2, pages 660–663, 2011.

[29] J. Lainema and K. Ugur. Angular intra prediction in high efficiency video coding (hevc). In *Multimedia Signal Processing (MMSP), 2011 IEEE 13th International Workshop on*, pages 1–5, 2011.

[30] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz. Adaptive deblocking filter. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):614–619, 2003.

[31] L. Liu, Y. (zoe Liu, and E. J. Delp. Enhanced intra prediction using context-adaptive linear prediction. *Proceedings of Picture Coding Symposium*, 2007.

[32] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky. Low-complexity transform and quantization in h.264/avc. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):598–603, 2003.

[33] D. Marpe, H. Schwarz, and T. Wiegand. Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):620–636, 2003.

[34] S. Matsuo, S. Takamura, and A. Shimizu. Modification of intra angular prediction in hevc. In *Signal Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific*, pages 1–4, 2012.

[35] Q. M. J. W. Mohammed Golam Sarwer. Improved intra prediction of h.264/avc. *InTech, Available from: http://www.intechopen.com/b ooks/effective-video-coding-for-multimedia- applications/improved-intra-prediction-of-h-264-avc.*

[36] N.-M. Nguyen, X.-T. Tran, P. Vivet, and S. Lesecq. An efficient context adaptive variable length coding architecture for h.264/avc video encoders. In *Advanced Technologies for Communications (ATC), 2012 International Conference on*, pages 158–164, 2012.

[37] V. Rajaravivarma and V. Rajaravivarma. Fast discrete cosine transform for 1-d and 2-d signals with and without symmetry. In *Communications, Computers, and Signal Processing, 1995. Proceedings., IEEE Pacific Rim Conference on*, pages 193–196, 1995.

[38] I. Richardson. 4x4 transform and quantization in h.264/avc. In *VCodex Limited White Paper, http://www.vcodex.com/*, volume 12, pages 1–13, 2010.

[39] M. Rizkalla, M. Ei-Sharkawy, P. Salama, and B. Dukel. Implementation of floating point fast discrete cosine transform. In *Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on*, volume 2, pages II–17–II–20 vol.2, 2002.

[40] K. Rose, A. Heiman, and I. Dinstein. Dct/dst alternate-transform image coding. *Communications, IEEE Transactions on*, 38(1):94–101, 1990.

[41] A. Saxena and F. Fernandes. Mode dependent dct/dst for intra prediction in block-based image/video coding. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 1685–1688, 2011.

[42] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the h.264/avc standard. In *ieee transactions on circuits and systems for video technology in circuits and systems for video technology*, pages 1103–1120, 2007.

[43] A. Skodras. Fast discrete cosine transform pruning. *Signal Processing, IEEE Transactions on*, 42(7):1833–1837, 1994.

[44] K. R. Soon-kak Kwon, A. Tamhankar. Overview of h.264/mpeg-4 part 10. In *SPIE conference on Applications of Digital Image Processing XXVII*, pages 454–474, 2004.

[45] G. Strang. The discrete cosine transform. *SIAM Review*, 41:135–147, 1999.

[46] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(12):1649–1668, 2012.

[47] T. K. T. C. F. T. Oelbaum, V. Baroncini. Subjective quality assessment of the emerging avc/h.264 video coding standard. *International Broadcasting Conference (IBC), Sept., 2004.*

[48] Y. K. T. Wedi. Subjective quality evaluation of h.264/avc frext for hd movie content. *Joint Video Team document JVT-L033, July, 2004*, 2004.

[49] A. Tamhankar and K. Rao. An overview of h.264/mpeg-4 part 10. volume 1, pages 1–51, 2003.

[50] T. Tan, C. Boon, and Y. Suzuki. Intra prediction by template matching. In *Image Processing, 2006 IEEE International Conference on*, pages 1693–1696, 2006.

[51] Z. Wang. Pruning the fast discrete cosine transform. *Communications, IEEE Transactions on*, 39(5):640–643, 1991.

[52] Z.-d. Wang. A fast algorithm for the discrete sine transform implemented by the fast cosine transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 30(5):814–815, 1982.

[53] S. Wenger. H.264/avc over ip. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):645–656, 2003.

[54] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h.264/avc video coding standard. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(7):560–576, 2003.

[55] C. Yeo, Y. H. Tan, and Z. Li. Low-complexity mode-dependent klt for block-based intra coding. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 3685–3688, 2011.

[56] C. Yeo, Y. H. Tan, and Z. Li. Low-complexity mode-dependent klt for block-based intra coding. In *Image Processing (ICIP), 2011 18th IEEE International Conference on*, pages 3685–3688, 2011.

[57] C. Yeo, Y. H. Tan, Z. Li, and S. Rahardja. Mode-dependent fast separable klt for block-based intra coding. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 621–624, 2011.

[58] C. Yeo, Y. H. Tan, Z. Li, and S. Rahardja. Mode-dependent transforms for coding directional intra prediction residuals. *Circuits and Systems for Video Technology, IEEE Transactions on*, 22(4):545–554, 2012.

[59] S. Yu, Y. Gao, J. Chen, and J. Zhou. Distance-based weighted prediction for h.264 intra coding. In *Audio, Language and Image Processing, 2008. ICALIP 2008. International Conference on*, pages 1477–1480, 2008.

[60] B. Zeng and J. Fu. Directional discrete cosine transforms;a new framework for image coding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(3):305–313, 2008.

[61] L. Zhang, S. Ma, W. Gao, and X. Zhao. Enhanced line-based intra prediction with fixed interpolation filtering. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 613–616, 2011.

[62] L. Zhang, X. Zhao, S. Ma, Q. Wang, and W. Gao. Novel intra prediction via position-dependent filtering. *J. Vis. Comun. Image Represent.*, 22(8):687–696, Nov. 2011.

[63] Y. Zheng, P. Yin, O. Escoda, X. Li, and C. Gomila. Intra prediction using template matching with adaptive illumination compensation. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 125–128, 2008.

APPENDIX A

AVERAGE SNR COMPARISON SOURCE CODE

```
/*********************************************************************
*
*  avsnr.c
*
*  Telenor Broadband Services
*  Keysers gate 13
*  N-0130 Oslo
*  Norway
*
********************************************************************/
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>


void main(int argc,char **argv)
{
  FILE *fd;
  FILE *p_log;
  char snr_filename[100];
  float tmp;

  double xl,xh;// end points
  double diff;
  int mode;    // dsnr or percentage mode


  int J0=0;//SNR index
  int J1=1;//bitrate index

  double X[2][4];

  double Y[2][4];

  double E[4],F[4],G[4],H[4];
  double SUM[2];

  int ind[2];
int i,j;

double  DET0,DET1,DET2,DET3,DET;
double  D0,D1,D2,D3;
double  A,B,C,D;

if (argc != 2)
{
  printf("Usage: %s <snr.txt> \n",argv[0]);
  printf("<snr.txt> gives snr values and bitrates\n");
  printf("Format:\n");
  printf("snrA0  snrA1  snrA2  snrA3 \n");
  printf("birtA0 bitrA1 bitrA2 bitrA3 \n");
  printf("snrB0  snrB1  snrB2  snrB3 \n");
  printf("birtB0 bitrB1 bitrB2 bitrB3 \n");


  exit(-1);
}
strcpy(snr_filename,argv[1]);

if((fd=fopen(snr_filename,"r")) == NULL){
    printf("Error: Input file %s not found\n",snr_filename);
    exit(0);
}
else{
  printf("-------------------------------------------------------\n");
  printf(" SNR input file              : %s \n",snr_filename);
  printf("-------------------------------------------------------\n");
  printf(" Computing average differance of 2 datasets \n");

}

fscanf(fd,"%d",&mode);      // read mode 0=DSNR 1=Percentage
fscanf(fd,"%*[^\n]");         // new line


for(i=0;i<4;i++){
  fscanf(fd,"%f",&tmp);      // first 4 SNR values
  if(mode==0)
    Y[0][i]=tmp;
  else
    X[0][i]=tmp;
  }
fscanf(fd,"%*[^\n]");         // new line

for(i=0;i<4;i++){             // first 4 bitrate values
  fscanf(fd,"%f,",&tmp);
  if(mode==0)
    X[0][i]=log(tmp);
  else
    Y[0][i]=log(tmp);
}
```

```
fscanf(fd,"%*[^\n]");

for(i=0;i<4;i++){
  fscanf(fd,"%f,",&tmp);
  if(mode==0)
    Y[1][i]=tmp;
  else
    X[1][i]=tmp;

}
fscanf(fd,"%*[^\n]");

for(i=0;i<4;i++){
  fscanf(fd,"%f,",&tmp);
  if(mode==0)
    X[1][i]=log(tmp);
  else
    Y[1][i]=log(tmp);
}
fscanf(fd,"%*[^\n]");


xl=max(X[J0][0],X[J1][0]);
xh=min(X[J0][3],X[J1][3]);
ind[0]=J0;
ind[1]=J1;


for (j=0;j<2;j++){
  for (i=0;i<4;i++){
    E[i]=X[ind[j]][i];
    F[i]=E[i]*E[i];

    G[i]=E[i]*E[i]*E[i];
    H[i]=Y[ind[j]][i];
  }

  DET0= E[1]*(F[2]*G[3]-F[3]*G[2])-E[2]*(F[1]*G[3]-F[3]*G[1])
         +E[3]*(F[1]*G[2]-F[2]*G[1]);
  DET1=-E[0]*(F[2]*G[3]-F[3]*G[2])+E[2]*(F[0]*G[3]-F[3]*G[0])
         -E[3]*(F[0]*G[2]-F[2]*G[0]);
  DET2= E[0]*(F[1]*G[3]-F[3]*G[1])-E[1]*(F[0]*G[3]-F[3]*G[0])
         +E[3]*(F[0]*G[1]-F[1]*G[0]);
  DET3=-E[0]*(F[1]*G[2]-F[2]*G[1])+E[1]*(F[0]*G[2]-F[2]*G[0])
         -E[2]*(F[0]*G[1]-F[1]*G[0]);
  DET=DET0+DET1+DET2+DET3;


  D0=H[0]*DET0+H[1]*DET1+H[2]*DET2+H[3]*DET3;



  D1=
    H[1]*(F[2]*G[3]-F[3]*G[2])-H[2]*(F[1]*G[3]-F[3]*G[1])+H[3]
```

```
            *(F[1]*G[2]-F[2]*G[1])-
    H[0]*(F[2]*G[3]-F[3]*G[2])+H[2]*(F[0]*G[3]-F[3]*G[0])-H[3]
            *(F[0]*G[2]-F[2]*G[0])+
    H[0]*(F[1]*G[3]-F[3]*G[1])-H[1]*(F[0]*G[3]-F[3]*G[0])+H[3]
            *(F[0]*G[1]-F[1]*G[0])-
    H[0]*(F[1]*G[2]-F[2]*G[1])+H[1]*(F[0]*G[2]-F[2]*G[0])-H[2]
            *(F[0]*G[1]-F[1]*G[0]);

  D2=
    E[1]*(H[2]*G[3]-H[3]*G[2])-E[2]*(H[1]*G[3]-H[3]*G[1])+E[3]
            *(H[1]*G[2]-H[2]*G[1])-
    E[0]*(H[2]*G[3]-H[3]*G[2])+E[2]*(H[0]*G[3]-H[3]*G[0])-E[3]
            *(H[0]*G[2]-H[2]*G[0])+
    E[0]*(H[1]*G[3]-H[3]*G[1])-E[1]*(H[0]*G[3]-H[3]*G[0])+E[3]
            *(H[0]*G[1]-H[1]*G[0])-
    E[0]*(H[1]*G[2]-H[2]*G[1])+E[1]*(H[0]*G[2]-H[2]*G[0])-E[2]
            *(H[0]*G[1]-H[1]*G[0]);

  D3=
    E[1]*(F[2]*H[3]-F[3]*H[2])-E[2]*(F[1]*H[3]-F[3]*H[1])+E[3]
            *(F[1]*H[2]-F[2]*H[1])-
    E[0]*(F[2]*H[3]-F[3]*H[2])+E[2]*(F[0]*H[3]-F[3]*H[0])-E[3]
            *(F[0]*H[2]-F[2]*H[0])+
    E[0]*(F[1]*H[3]-F[3]*H[1])-E[1]*(F[0]*H[3]-F[3]*H[0])+E[3]*(F[0]
            *H[1]-F[1]*H[0])-
    E[0]*(F[1]*H[2]-F[2]*H[1])+E[1]*(F[0]*H[2]-F[2]*H[0])-E[2]*(F[0]
            *H[1]-F[1]*H[0]);


  A=D0/DET;
  B=D1/DET;
  C=D2/DET;
  D=D3/DET;


  SUM[j]=A*(xh-xl)+B*(xh*xh-xl*xl)/2+C*(xh*xh*xh-xl*xl*xl)/3
            +D*(xh*xh*xh*xh-xl*xl*xl*xl)/4;

}

diff=(SUM[1]-SUM[0])/(xh-xl);

if(mode==1)  //Percentage
   diff=(exp(diff)-1)*100;

if (mode==0)
{
  printf(" Logaritmic mode \n");
  printf(" Areal of integrated area()  = %f\n",diff);
}
else
{
  printf(" Percentage mode \n");
```

```
    printf(" Percentage difference between the courves are = %f\n",diff);
  }

  /*
  write to log file
  */
  printf(" Write to 'log.dat' \n");
  if (fopen("log.dat","r")==0)          /* check if file exist */
  {
    if ((p_log=fopen("log.dat","a"))==0)/* append new statistic at the end */
    {
      printf("Error open file %s  \n",p_log);
      exit(0);
    }
    else                                /* Create header for new log file */
    {
      fprintf(p_log," ----------------------------------------------------
----------------------------------------------------------------- \n");
      fprintf(p_log,"|   Log file for  'avsnr'. This file is generated during
 first encoding session, new sessions will be appended |\n");
      fprintf(p_log," ----------------------------------------------------
----------------------------------------------------------------- \n");
      fprintf(p_log,"|SNR a0|SNR a1|SNR a2|SNR a3|Bit a0|Bit a1|Bit a2|Bit a3|
SNR b0|SNR b1|SNR b2|SNR b3|Bit b0|Bit b1|Bit b2|Bit b3| Mode | Output  |\n");
      fprintf(p_log," ----------------------------------------------------
----------------------------------------------------------------- \n");
    }
  }
  else
    p_log=fopen("log.dat","a");       /* File exist,just open for appending */

  fprintf(p_log,"|");
  if (mode==0){
    for (i=0;i<4;i++)
      fprintf(p_log,"%6.2f|",Y[0][i]);
    for (i=0;i<4;i++)
      fprintf(p_log,"%6.2f|",exp(X[0][i]));
    for (i=0;i<4;i++)
      fprintf(p_log,"%6.2f|",Y[1][i]);
    for (i=0;i<4;i++)
      fprintf(p_log,"%6.2f|",exp(X[1][i]));

     fprintf(p_log,"  DSNR mode  |",mode);
     fprintf(p_log,"%6.3f dB|\n",diff);
  }
  else{
    for (i=0;i<4;i++)
      fprintf(p_log,"%6.2f|",X[0][i]);
    for (i=0;i<4;i++)
      fprintf(p_log,"%6.2f|",exp(Y[0][i]));
    for (i=0;i<4;i++)
      fprintf(p_log,"%6.2f|",X[1][i]);
    for (i=0;i<4;i++)
```

```
       fprintf(p_log,"%6.2f|",exp(Y[1][i]));


      fprintf(p_log," Percent mode|");
      fprintf(p_log,"%8.2f% %|\n",diff);
   }


   printf(" AvSNR version 4 \n");

}
```

[1]