PERFORMANCE ANALYSIS AND IMPLEMENTATION OF MODE DEPENDENT DCT/DST IN H.264/AVC

by

PRIYADARSHINI ANJANAPPA

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2012

Acknowledgements

I would like to offer my earnest gratitude to my advisor, Dr. K. R. Rao, who has been a constant support throughout my thesis with his persistent encouragement, motivation and patience. I am thankful to him for introducing me to the image and video compression domain.

I would like to thank Dr. William Dillon and Dr Ioannis Schizas for serving on my thesis advisory committee.

I would like to express my deepest gratitude to Dr Ankur Saxena for his tireless patience and invaluable guidance during every stage of my thesis. Without his persistent help this thesis would not have been possible. I am thankful to Shevach Riabtsev, Dr. Elena Alshina, Dr. Alexis Michael Tourapis and Madhu Krishnan for their suggestions and patience in answering all my questions which helped me in successfully completing my work.

I am thankful to every individual who has motivated and helped me during various phases of my graduate years.

Most importantly, I am indebted to my parents for their love, patience and encouragement. I am grateful to Abhijith and Rohan for their constant emotional support which kept me motivated and high-spirited during my graduate life.

November 26, 2012

Abstract

PERFORMANCE ANALYSIS AND IMPLEMENTATION

OF MODE DEPENDENT DCT/ DST

IN H.264/AVC

Priyadarshini Anjanappa, M.S.

The University of Texas at Arlington, 2012

Supervising Professor: K. R. Rao

Video communications require significant amounts of information transmission. Video compression involves the bitrate reduction of a digital video signal carrying visual information. Compression can be a lossless or a lossy operation. Because of the immense volume of video information, lossy operations are mainly used for video compression. H.264 is a video compression standard that defines a wide set of coding tools to achieve compression. Transform coding is a lossy compression technique that exploits statistical dependencies between image pixels by converting them into coefficients with little or no dependencies. Block transform coding is widely used in image and video compression systems, which makes use of the high degree of correlation between adjacent image pixels to provide energy compaction or coding gain in the transform domain. An integer approximation of DCT-II is used for transform coding in H.264 and also in HEVC, which is the latest upcoming video compression standard.

The conventional DCT is implemented through two 1D transforms, one along the vertical direction and another along the horizontal direction. The conventional DCT seems to be the best choice for image blocks in which vertical and/or horizontal edges are dominating. To exploit the redundancies along dominant directions in an image block

other than vertical and horizontal, many directional transform schemes were proposed. It is observed that after performing intra prediction, there is still significant directional information left in the prediction residual. To exploit the directionality in the intra prediction residuals, many mode dependent transforms were proposed. The mode dependent DCT/DST is one of them, which was considered during the standardization activities of HEVC. Categorizing the intra prediction directions depending upon whether prediction in an image block is made only from the top row of reference samples or the left column of reference samples or both, DST-VII and DCT-II are adaptively used for the 1D vertical and horizontal transforms. The purpose of this thesis is to revisit this scheme and analyze its performance in H.264 for 4x4 luma intra prediction residuals. HEVC uses DST for 4x4 intra prediction residuals only. The performance of the proposed mode dependent DCT/DST in H.264 is compared with the default H.264 transform implementation and also the latest default HEVC transform implementation.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

1.1 Transform Coding

Transform coding is one of the basic coding tools widely adopted in digital image and video compression to reduce the inherent spatial redundancy between adjacent pixels. The Karhunen-Loeve transform (KLT) posseses several optimality properties in terms of high resolution quantization and full decorrelation of transformed samples. However, practical use of KLT is limited due to its high computational complexity. The discrete cosine transform (DCT-II) [16] has excellent energy compaction and among the unitary transforms it is the best approximation to the optimal KLT.

A typical 2D DCT-II is separable into two 1D transforms performed first along one axis of the image or video frame, and then along the other axis of the resultant from the previous procedure. The resultant cosine values are difficult to approximate in fixed precision integers, thus producing rounding errors in practical applications. Rounding errors can introduce enough error into computations and alter the orthogonality property of the transform. Residual decoding contains the possibility of drift (mismatch between the decoded data in the encoder and decoder). The drift arises from the fact that the inverse transform is not fully specified in integer arithmetic; rather it must satisfy statistical tests of accuracy compared with a floating point implementation of the inverse transform. Unlike the DCT used in previous standards, the transforms in H.264 can be computed exactly in integer arithmetic, thus avoiding inverse transform mismatch problems [17].

1.2 Alternative Transforms

Conventionally, the 2D DCT is carried out as a separable transform by cascading two 1D transforms in the vertical and horizontal dimensions. This approach does not take advantage of the locally anisotropic features present in images because it favors

horizontal or vertical features over others [31]. The DCT is the most often used transform in block-based video codecs, but the DCT basis functions are sub-optimal for some types of residuals. Residual signals with strong directional components cannot be represented efficiently with DCT basis vectors. Typically, alternative transforms work efficiently for such intra residuals [28].

Many directional alternative transforms were proposed between April 2010 and July 2010 to the Ad Hoc Group which was a part of the HEVC standardization activities [18]. Some of them are as follows:

- MDDT (mode dependent directional transform) [19][20]
- Orthogonal MDDT [21]
- Adaptive DCT/DST [22]
- Rate-distortion optimized transform (RDOT) for the intra and inter prediction residual [23] [30]
- Rotational transform [24]
- Directional DCT [25] [26] [27]

In H.264, 8 directional intra-prediction modes are used for 4x4 and 8x8 blocks. In the MDDT, instead of using the DCT, a distinct separable transform is applied to the intra prediction residual in each of the 8 intra prediction modes. These 8 distinct transforms are obtained by applying the singular value decomposition (SVD) to prediction residuals from training video sequences. Thus the MDDT improves coding efficiency by approximating the optimal Karhunen Loeve Transform (KLT) in each mode by a separable transform obtained through training.

The adaptive DCT/DST achieves the same objective but incurs less computational complexity. A particular discrete sine transform (DST) was shown analytically to be the actual KLT along the prediction direction for H.264 intra prediction

modes [29]. It was also shown that if prediction is not performed along a particular direction, then the conventional DCT performs close to the KLT. Therefore, instead of using 9 distinct separable transforms, the adaptive DCT/DST applies either DST or the DCT along each row or column for all modes. The mapping from row/column/mode to DCT/DST is also derived analytically [28].

Unlike AVC in which the same DCT-like transform is applied at the intra prediction errors of all intra prediction modes of the same block size (4x4, 8x8 or 16x16), the directional DCT (DDCT) assigns a different transform and scanning pattern to each intra prediction mode. These transforms and scanning patterns are designed taking into picture each intra prediction direction. Since the intra prediction mode is known, the first 1D DCT is performed along the prediction direction in the first stage. By first applying the transform along the prediction direction, DDCT has the potential to minimize the artifact around the object boundaries. In the second stage of the two stage transform, another 1D DCT is performed perpendicular to the prediction direction. In cases of prediction modes that are neither horizontal nor vertical, the 1D DCTs used are of different sizes. To make the transform sizes more balanced, the DDCT groups pixels in the corners together in order to use DCT of longer size, hence more efficient in terms of compression [27]. Unlike MDDT, DDCT can be implemented as a fast transform and it does not require collecting scanning statistics in the decoder, computation can be reduced compared with MDDT.

In contrast to other alternative transforms, the rotational transform (ROT) does not replace the main transform core. It is applied as a secondary transform, after the primary DCT. The disadvantage of ROT is its high encoding complexity [24].

1.3 Outline

The purpose of the thesis is to implement and analyze mode dependent DCT/DST in H.264. At the time the thesis was in progress, mode dependent DCT/DST was being explored during HEVC standardization activities and experiments for different coding tools in the new standard, which was the motivation for this thesis. In order to implement the mode dependent DCT/DST in H.264, it is necessary to understand the structure and features of H.264/AVC and HEVC. So the second chapter will give an overview of H.264/AVC and its coding tools. The third chapter will give an overview of the structure of HEVC and its coding tools. The fourth chapter will contain the implementation and analysis details of mode dependent DCT/DST in H.264. Every detail regarding the implementation will be discussed in this chapter. The chapter ends with conclusions of the analysis and futurework.

Chapter 2

H.264/AVC

2.1 Introduction

H.264 Advanced Video Coding is an industry standard for video coding which was first jointly published in 2003 by two international standards bodies- International Telecommunication Union (ITU-T) and International Organization for Standardization / International Electrotechnical Commission (ISO / IEC). [1] The standard was developed by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC joint working group, the Moving Picture Experts Group (MPEG). The product of this partnership effort is known as the Joint Video Team (JVT) [3]. Recommendation H.264: Advanced Video Coding [2] is the standard document which defines a format or syntax for the compressed video and a method for decoding this syntax to produce a displayable sequence.

The application focus for the initial version of the standard document was broad – from video conferencing to entertainment (broadcasting over cable, satellite, terrestrial, cable modem, DSL etc.; storage on DVDs and hard disks; video on demand etc.) to streaming video, surveillance and military applications, and digital cinema [6]. Only the central decoder is standardized, by imposing restrictions on the bitstream and syntax, and defining the decoding process of the syntax elements such that every decoder conforming to the standard will produce similar output when given an encoded bitstream that conforms to the constraints of the standard [4].

Motivated by the rapidly growing demand for coding of higher-fidelity video material, especially in application areas like professional film production, video post-production and high definition TV/DVD, the JVT issued a Call for Proposal for the support of extended sample bit depth and chroma format in the H.264/MPEG4-AVC standard,

following which, in September 2004, the Fidelity Range Extensions (FREXt) of H.264/MPEG4-AVC was included in version 4 of the standard document [7] [2].

The increasing need for coding the same original content at different bandwidths and display resolutions led to the development of the Scalable Video Coding (SVC) extension to H.264, standardized as H.264 SVC. SVC supports efficient coding of video in such a way that multiple versions of the video signal can be decoded at a range of bitrates, spatial resolutions and/or temporal resolutions or frame rates [1]. Hence, SVC provides functionalities such as graceful degradation in lossy transmission environments as well as bit rate, format and power adaptation. These functionalities provide enhancements to transmission and storage applications. The term “scalability” refers to the removal of parts of the video bit stream in order to adapt it to the various needs or preferences of end users as well as to varying terminal capabilities or network conditions [8].

There is a trend towards creating and delivering multiple views of the same video scene. Stereoscopic video, with suitable display technology, gives the impression of a 3D image. Multiple views of a scene can give the user the option of choosing their viewpoint. Free viewpoint video (FVV) can deliver any view of a scene, by synthesizing intermediate views between actual camera positions. The multiview applications generally require coding of multiple, closely related video signals or views [1]. Multiview video coding (MVC) was standardized as an extension to H.264, which provides compact representation for multiple views of a video scene, such as multiple synchronized video cameras. It enables inter-view prediction to improve compression capability, as well as support of ordinary temporal and spatial prediction [9].

H.264 is based on hybrid video coding – video is compressed using a hybrid of motion compensation and transform coding. These video coding algorithms compress the

video data by reducing the redundancies inherent in video, which fall into four classes, namely, spatial, temporal, perceptual and statistical [5]. Various tools are used by video coding algorithms to reduce these redundancies:

- Chroma subsampling, quantization and prefiltering to remove perceptual redundancies
- DCT, intraprediction, integer transform and variable block size transform to remove spatial redundancies
- Block motion estimation, multiple reference frame motion estimation and variable block motion estimation to remove temporal redundancies

Huffman coding, adaptive VLC (variable length coding) and arithmetic coding to remove statistical redundancies.

These algorithms differ in which tools are used for reducing the redundancies and in the specific ways these tools are applied [5].

2.2 Profiles and Levels

H.264/AVC contains a rich set of video coding tools. Not all coding tools are required for all the applications. Therefore, subsets of coding tools are defined; these sunsets are called Profiles [7]. Profiles and levels specify conformance points that provide interoperability between encoder and decoder implementations within applications of the standard and between various applications that have similar functional requirements[11]. A profile defines a set of syntax features for use in generating conforming bitstreams, whereas a level places constraints on certain key parameters of the bitstream such as maximum bit rate and maximum picture size. Figure 2.1 shows an overview of the profiles in H.264/MPEG4-AVC.

By taking into account different requirements and limitations of typical target applications, three subsets of coding tools were defined in the initial, i.e., 2003 approved H.264/MPEG4-AVC standard [11]:

- Baseline profile: Targeted at low cost mobile applications and videoconferencing applications in which a minimum of computational complexity and a maximum of error robustness are required.
- Main profile: Targeted at standard-definition digital TV broadcast applications that require a maximum coding efficiency, with slightly less emphasis on error robustness.
- Extended profile: Intended for streaming video and designed to provide a compromise between the Baseline and Main profile capabilities with an additional focus on the specific needs of video streaming applications, and further added robustness to errors and packet losses.

The FRExt amendment, which was released in 2004, defines four new profiles in H.264 [6]:

- High (HP) for high definition broadcast and disc storage applications supporting 8-bit video with 4:2:0 sampling
- High 10 (Hi10P) with support for up to 10 bits of representation accuracy per sample of decoded picture precision
- High 4:2:2 (Hi422P) with support for 4:2:2 chroma subsampling and up to 10 bits per sample
- High 4:4:4 (Hi444P) supporting up to 4:4:4 chroma subsampling and up to 12 bits per sample and additionally supporting efficient lossless region coding and an integer color transform for coding RGB video while adding color-space transformation error.

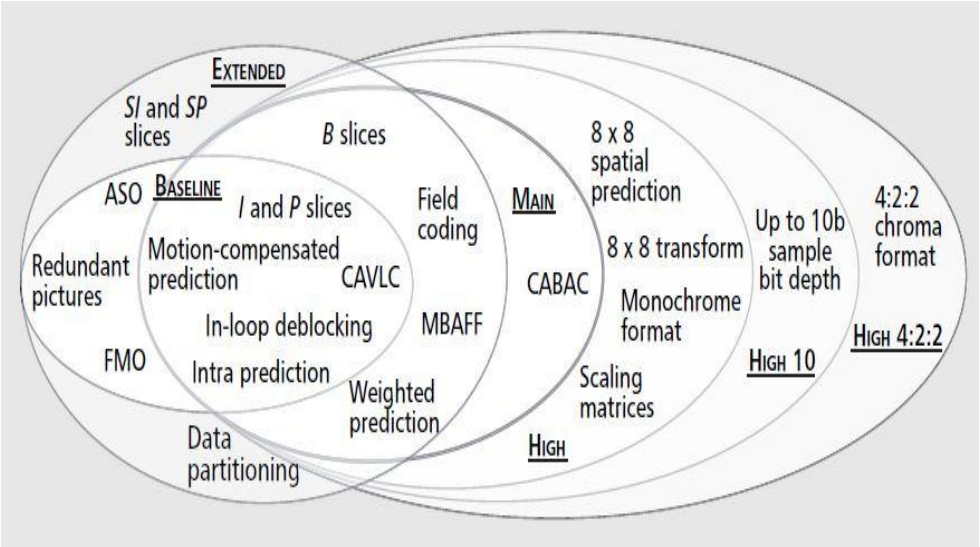


Figure 2.1 Illustration of profiles in H.264/AVC [11]

For real-time decoders or decoders with constrained memory size, it is important to specify the processing power and memory size needed for implementation. Picture size plays the main role in influencing those parameters. H.264/AVC defines 16 different levels, tied mainly to the picture size [7]. Levels also provide constraints on the number of reference pictures and the maximum compressed bit rate that can be used. In the standard, levels specify the maximum frame sizes in terms of only the total number of pixels/frame. Table 2.1 shows 16 different levels defined for H.264/AVC standard.

Table 2.1 Levels in H.264/AVC [7]

Level number	Typical picture size	Typical frame size	Maximum compression bit rate	Maximum number of reference frames
1	QCIF	15	64 kbps	4
1b	QCIF	15	128 kbps	4
1.1	CIF or QCIF	7.5 (CIF) / 30 (QCIF)	192 kbps	2 (CIF) / 9 (QCIF)
1.2	CIF	15	384 kbps	6
1.3	CIF	30	768 kbps	6
2	CIF	30	2 Mbps	6
2.1	HHR(480i or 576i)	30 / 25	4 Mbps	6

Table 2.1—*Continued*

2.2	SD	15	4 Mbps	5
3	SD	30 / 25	10 Mbps	5
3.1	1280x720p	30	14 Mbps	5
3.2	1280x720p	60	20 Mbps	4
4	HD Formats (720p or 1080i)	60p / 30i	20 Mbps	4
4.1	HD Formats (720p or 1080i)	60p / 30i	50 Mbps	4
4.2	1920x1080p	60p	50 Mbps	4
5	2kx1k	72	135 Mbps	5
5.1	2kx1k or 4kx2k	120 / 30	240Mbps	5

2.3 H.264/AVC Encoder

An H.264 video encoder carries out prediction, transforming and encoding processes to produce a compressed H.264 bitstream. Figure 2.2 shows the typical structure of an H.264 video encoder. A coded video sequence in H.264/AVC consists of a sequence of coded pictures. A coded picture can represent either an entire frame or a single field. A frame of video can be considered to contain two interleaved fields: a top field and a bottom field.

The typical encoding operation for a picture begins with splitting the picture into blocks of samples. The first picture of a sequence or a random access point is typically coded in Intra mode. This is done without using any other pictures as prediction references. Each sample of a block in such an Intra picture is predicted using spatially neighboring samples of preciously coded blocks. For all remaining pictures of a sequence or between random access points, Inter (inter-picture) coding is used. Intercoding employs interpicture temporal prediction using other previously decoded pictures. The residual of the prediction (either Intra or Inter), which is the difference between the

original input samples and the predicted samples for the block, is transformed. The transform coefficients are then scaled and approximated using scalar quantization. The quantized transform coefficients are entropy coded and transmitted together with the entropy-coded prediction information. The encoder contains a model of the decoding process so that it can compute the same prediction values computed in the decoder for the prediction of subsequent blocks in the current picture or subsequent coded pictures.

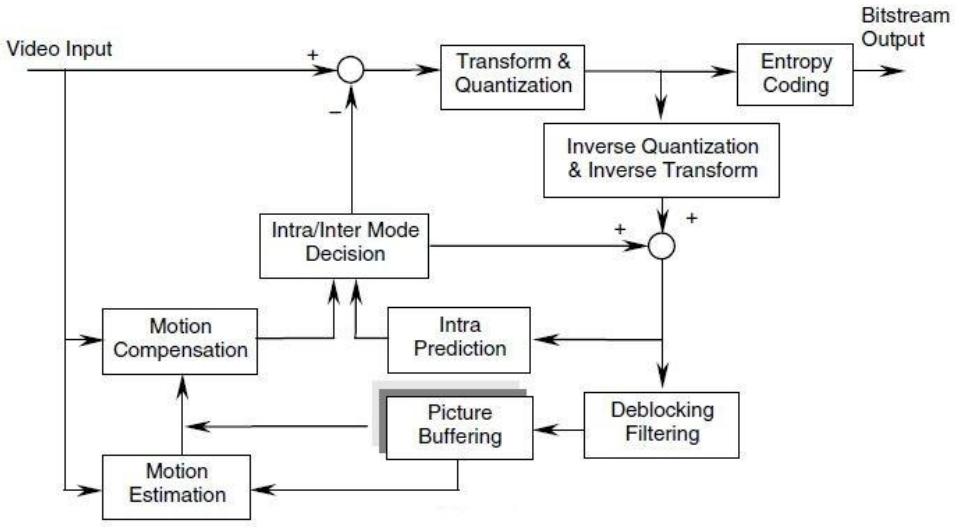


Figure 2.2 Block representation of an H.264/AVC video encoder [12]

2.3.1 Subdivision of a picture into macroblocks

Each picture of a video sequence is partitioned into fixed size macroblocks that each cover a rectangular picture area of 16x16 samples of the luma component and 8x8 samples of each of the two chroma components. The macroblocks are organized in slices which represent regions of a given picture that can be decoded independent of each other. H.264/AVC supports five slice-coding types:

- I (intra) slice : All macroblocks in this slice are coded without referring to any other pictures of the video sequence

- P (predictive) and B (bi-predictive) slices: Prior coded images can be used to form a prediction signal for macroblocks in these slices
- SP (switching P) and SI (switching I) slices: Specified for efficient switching between bitstreams coded at various bit rates.

2.3.2 Intra prediction

Spatial redundancy in a video frame is exploited using intra prediction. An intra macroblock is coded without referring to any data outside the current slice. A macroblock is coded as an intra macroblock when temporal prediction is impossible (for the first frame of video) or inefficient (at scene changes) [5].

Prediction for intra macroblocks is determined using a weighted average of pixels from the adjacent edges of neighbouring macroblocks that are decoded before the current macroblock. The weights of the prediction are determined using the directional prediction modes that are specified for each block size.

Figure 2.3 (a) shows the neighbouring pixels used to predict a 4x4 block. Figure 2.3 (b) shows the eight prediction directions allowed for 4x4 and 8x8 block sizes; the mode 2, DC mode, is the average of the neighbouring pixels and isn't shown in the figure.

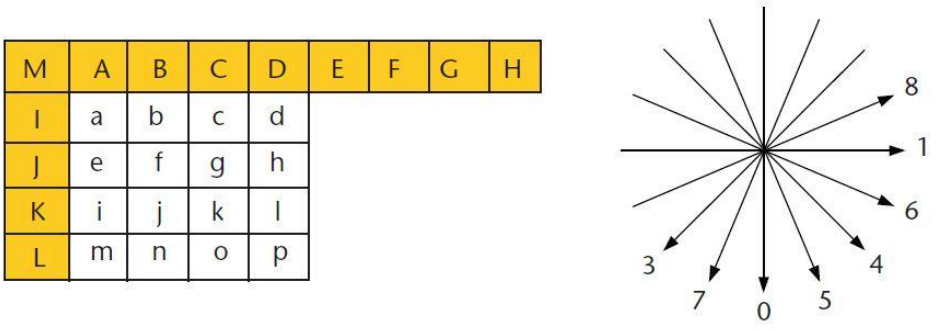


Figure 2.3 (a) Intra prediction samples for a 4x4 block; A through M are previously predicted pixels (b) Directions for prediction modes for 4x4 and 8x8 blocks [5]

For 4x4 and 8x8 (High profiles only) luma blocks, nine intra prediction modes are defined [1]:

- Mode 0 (Vertical) : The upper samples A, B, C, D are extrapolated vertically
- Mode 1 (Horizontal) : The left samples I,J,K,L are extrapolated horizontally
- Mode 2 (DC) : All samples in P are predicted by the mean of samples A to D and I to L
- Mode 3 (Diagonal Down-Left) : The samples are interpolated at a 45° angle between lower-left and upper right
- Mode 4 (Diagonal Down-Right) : The samples are extrapolated at a 45° angle down and to the right
- Mode 5 (Vertical-Left) : Extrapolation at an angle of approximately 26.6° to the left of vertical, i.e. width/height = ½
- Mode 6 (Horizontal-Down) : Extrapolation at an angle of approximately 26.6° below horizontal
- Mode 7 (Vertical-Right) : Extrapolation or interpolation at an angle of approximately 26.6° to the right of vertical
- Mode 8 (Horizontal-Up) : Interpolation at an angle of approximately 26.6° above horizontal

Figure 2.4 illustrates the nine 4x4 intra prediction modes. Figure 2.5 illustrates the four intra prediction modes for 16x16 luma blocks.

For 16x16 luma blocks, four intra prediction modes are defined [1]:

- Mode 0 (Vertical) : Extrapolation from upper samples (H)
- Mode 1 (Horizontal) : Extrapolation from left samples (V)
- Mode 3 (DC) : Mean of upper and left hand samples (H+V)

- Mode 4 (Plane): A linear ‘plane’ function is fitted to the upper and left-hand samples H and V. This works well in areas of smoothly-varying luminance.

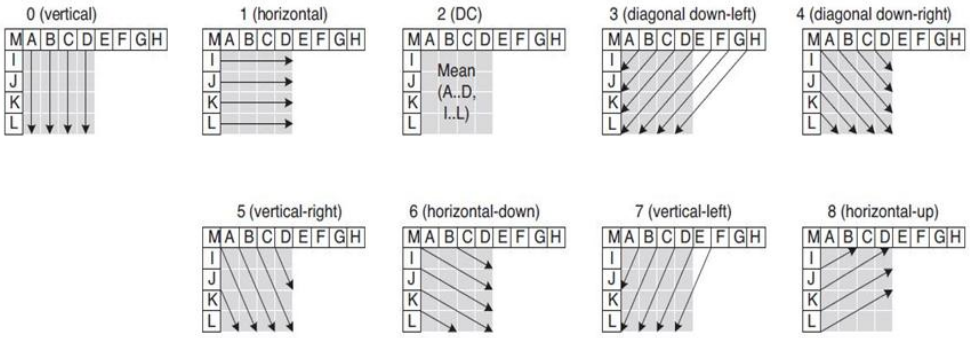


Figure 2.4 Intra prediction modes for 4x4 luma blocks [1]

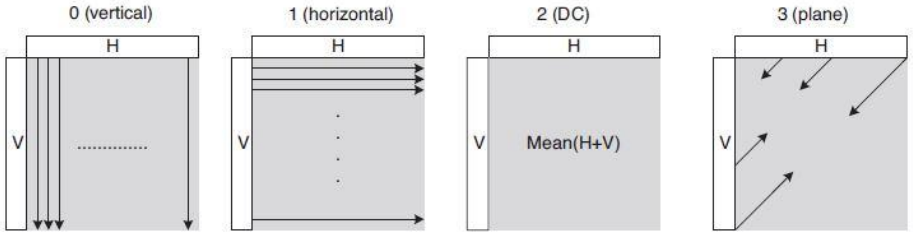


Figure 2.5 Intra prediction modes for 16x16 luma blocks [1]

For chroma blocks, four intra prediction modes are defined which are very similar to the 16x16 luma prediction modes, except that the numbering of the modes is different. The modes are DC (mode 0), horizontal (mode 1), vertical (mode 2) and plane (mode 3).

2.3.3 Inter prediction

Inter prediction is used to reduce temporal redundancies between video frames with the help of motion estimation and compensation. Each 16x16 macroblock may be predicted using a range of block sizes as shown in Figure 2.6. Each macroblock partition may be predicted from different reference frames. However, the sub-macroblock partitions within an 8x8 macroblock share the same reference frames.

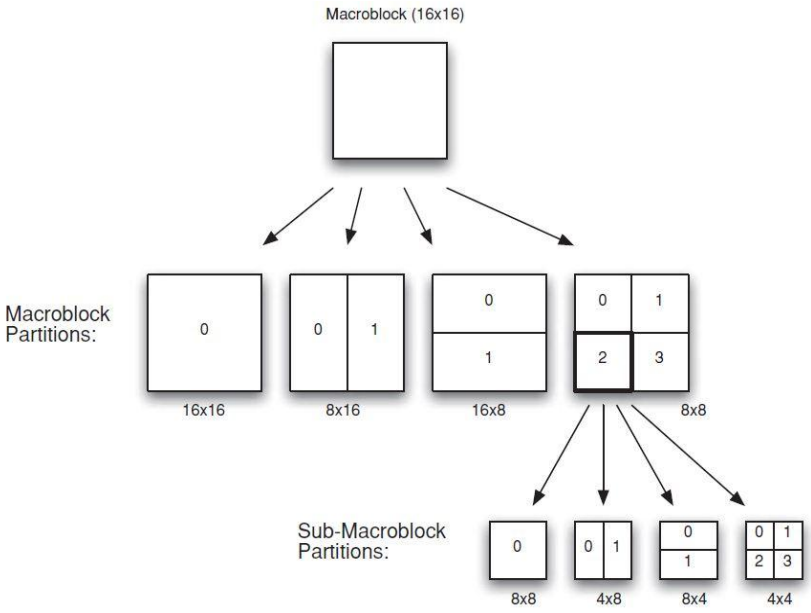


Figure 2.6 Macroblock partitions and sub-macroblock partitions [1]

The reference picture for inter prediction is chosen from a list of previously decoded pictures, stored in a DPB (decoded picture buffer), which may include pictures before and after the current picture in display order. The offset between the position of the current partition and the prediction region in the reference picture is a motion vector. Encoding a motion vector for each partition can cost a significant number of bits, especially if small partition sizes are chosen. Motion vectors for neighboring partitions are often highly correlated. So each motion vector is predicted from vectors of nearby, previously coded partitions. The motion vector may point to integer, half- or quarter-sample positions in the luma component of the reference picture. The accuracy of motion compensation is in units of one quarter of the distance between luma samples. In case the motion vector points to an integer-sample position, the prediction signal consists of the corresponding samples of the reference picture; otherwise the corresponding sample is obtained using interpolation of the samples of the reference picture to generate noninteger positions. The prediction values at half-sample positions are obtained by

applying a one-dimensional 6-tap FIR filter horizontally and vertically. Prediction values at quarter-sample positions are generated by averaging samples at integer- and half-sample positions. Figure 2.7 and Figure 2.8 show interpolation of luma half-sample and quarter-sample positions respectively. The prediction values for the chroma component are always obtained by bilinear interpolation.

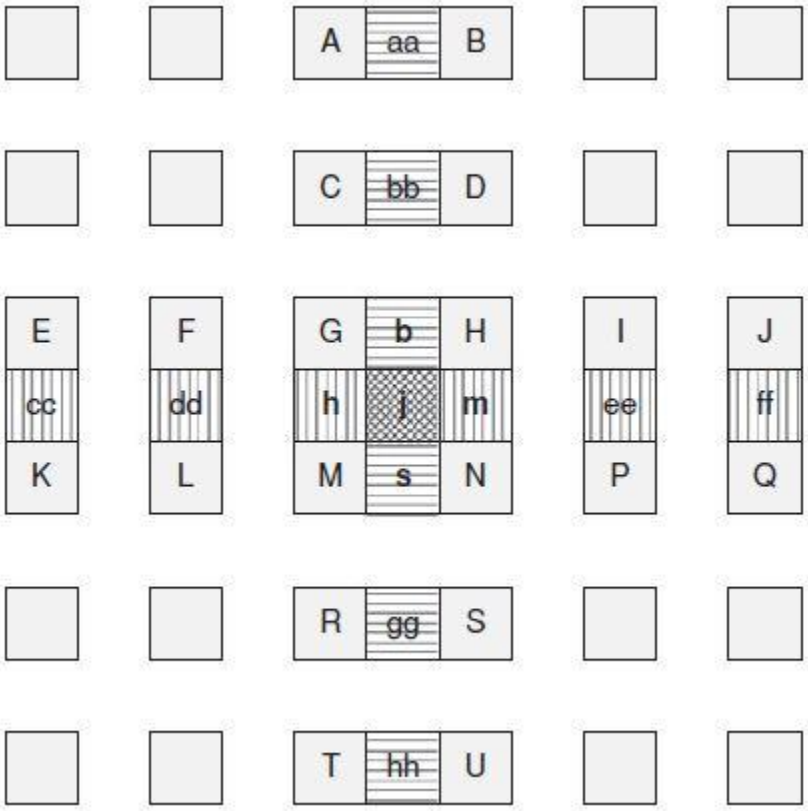


Figure 2.7 Interpolation of luma half-sample positions [1]

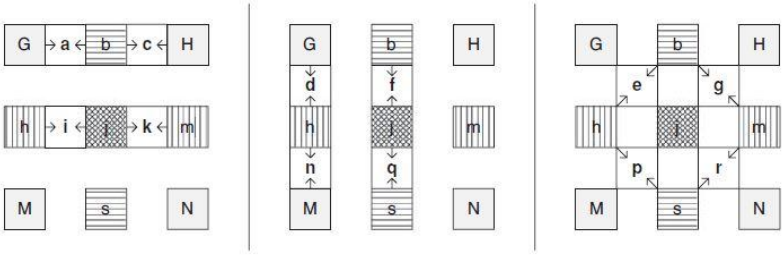


Figure 2.8 Interpolation of luma quarter-sample positions [1]

Each motion vector is differentially coded from the motion vectors of neighboring blocks. The prediction block may be generated from a single prediction region in a reference picture, for a P or B macroblock, or from two prediction regions in reference pictures, for a B macroblock. Optionally, the prediction block may be weighted according to the temporal distance between the current and reference pictures, known as weighted prediction. In a B macroblock, a block may be predicted in direct mode, in which case no residual samples or motion vectors are sent and the decoder infers the motion vector from previously received vectors.

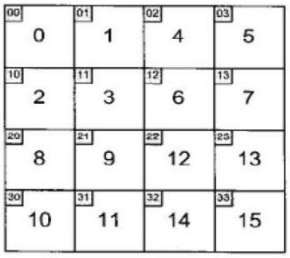
2.3.4 Transform coding

H.264/AVC applies transform coding to the prediction residuals. At the encoder the process includes a forward transform, zig-zag scanning, scaling and rounding as the quantization process is followed by entropy coding. The transform and quantization processes are designed to provide efficient coding of video data, to eliminate mismatch or ‘drift’ between encoders and decoders and to facilitate low complexity implementations. The core transform is a 4x4 or 8x8 integer transform, a scaled approximation to the discrete cosine transform (DCT), which is applied to the prediction residuals. If a macroblock is predicted using 16x16 intra prediction, which is intended for coding of smooth areas, a second transform, Hadamard transform, is applied to the lowest or DC frequency coefficients of the core transform. The DC values tend to be

highly correlated and this second transform improves the coding performance. The Hadamard transform is 4x4 for luma and 2x2 for chroma (due to the 4:2:0 sampling) as shown in Figure 2.9 (b) and (c) respectively. The 4x4 integer transform matrix is shown in Figure 2.9 (a).

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

(a) (b) (c)



(d)

Figure 2.9 (a) 4x4 integer transform matrix (b) 4x4 Hadamard transform matrix (c) 2x2 Hadamard transform matrix (d) The sixteen 4x4 blocks with DC coefficients drawn as smaller blocks on upper left corner

The forward and inverse transforms are implemented using only additions and bit-shifting operations of 16-bit integer values. Only 16-bit memory accesses are needed for a good implementation of the forward transform and quantization process in the encoder. For the quantization of transform coefficients, H.264 uses uniform-reconstruction quantizers [11]. One of 52 quantizer step size scaling factors is selected for each macroblock by a quantization parameter (QP). The scaling operations are arranged such that there is a doubling in quantization step size for each increment of six in the value of QP. The quantized transform coefficients are scanned in a zig-zag fashion as shown in Figure 2.10.

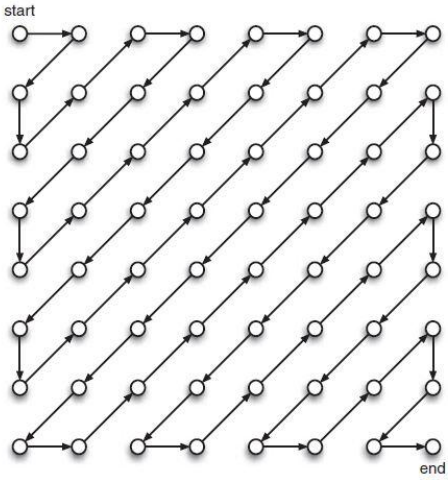


Figure 2.10 Zig-zag scan order illustrated for an 8x8 block [1]

2.3.5 Entropy coding

A coded H.264 stream or a file consists of a series of coded symbols. These make up the H.264 syntax and include parameters, identifiers and delimiting codes, differentially coded motion vectors and transform coefficients. Entropy coding converts each symbol into a binary pattern that is transmitted or stored. Symbols occurring in the H.264 syntax above slice data level are coded using Fixed Length Codes or Exp-Golomb Codes. Symbols at the slice data level and below in the H.264 syntax are coded in one of the two ways:

- Context Adaptive Variable Length Coding (CAVLC)
- Context Adaptive Binary Arithmetic Coding (CABAC)

CAVLC is used to encode residual, scan ordered blocks of transform coefficients. It is designed to take advantage of several characteristics of quantized coefficient blocks. A block of coefficients is scanned using zigzag scanning and converted into a series of variable length codes (VLCs). Certain VLC tables are chosen based on local statistics like the number of non-zero coefficients in neighboring blocks and the magnitude of recently coded coefficients. If CAVLC is used together with the 8x8 integer transform,

each 8x8 block of quantized transform coefficients is processed as four 4x4 blocks for the purpose of CAVLC encoding and decoding.

CABAC is an optional entropy coding mode available in Main and High profiles. CABAC achieves good performance through selecting probability models for each syntax element according to the element's context, adapting probability estimates based on local statistics and using arithmetic coding rather than variable-length coding. CABAC design is based on three components: binarization, context modeling and binary arithmetic coding. Compared to CAVLC, CABAC can typically provide reductions in bit rate of 10-20% for the same objective video quality when coding SDTV/HDTV signals [11].

2.3.6 Deblocking Filter

In H.264, there are two sources that can introduce blocking artifacts. The most significant one is the integer 4x4 transform in intra and inter frame prediction residual coding. Coarse quantization of the transform coefficients can cause visually disturbing discontinuities at the block boundaries. The second source of blocking artifacts is motion compensation prediction. Motion compensated blocks are generated by copying interpolated pixel data from different locations of possibly different reference frames. Since there is almost never a perfect fit for this data typically arise. Additionally, in the copying process, existing edge discontinuities in the reference frames are carried into the interior of the block to be compensated [6]. The deblocking loop filter typically improves both objective and subjective quality of video streams. Quality improvements are mainly due to the fact that filtered reference frames offer higher quality prediction for motion compensation. The deblocking filter process is invoked for the luma and chroma components separately. For each macroblock, vertical edges are filtered first, from left to right, and then horizontal edges are filtered from top to bottom. The luma deblocking filter process is performed on four 16-sample edges and the deblocking filter process for each

chroma components is performed on two 8-sample edges, for the vertical edges as shown in the left side of Figure 2.11 and for the horizontal edges as shown in the right side of Figure 2.11.

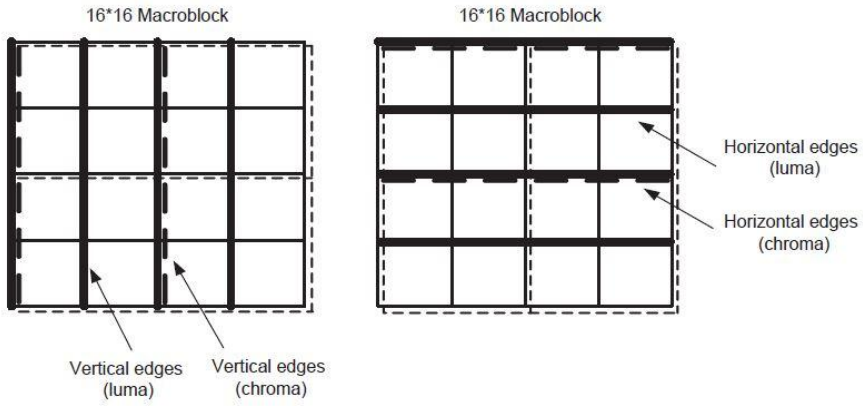


Figure 2.11 Boundaries in a macroblock to be filtered (luma boundaries shown with solid lines and chroma boundaries shown with dashed lines) [6]

2.4 H.264 decoder

The H.264 decoder is illustrated in Figure 2.12. The decoder works similar to the local decoder at the encoder. The decoder receives the compressed H.264 bitstream, decodes each of the syntax elements and extracts the following information:

- Quantized transform coefficients
- Prediction information
- Information about the structure of the compressed data and the compressed tools used during encoding
- Information about the complete video sequence

After entropy (CABAC or CAVLC) decoding, the transform coefficients are inverse scanned and inverse quantized prior to being inverse transformed. To the resulting blocks of the residual signal, an appropriate prediction signal (intra or inter) is added depending on the macroblock type and mode, the reference frame and the motion

vectors. The reconstructed video frames undergo deblocking filtering prior to being stored for future use for prediction. The frames at the output of the deblocking filter may need to undergo reordering prior to display.

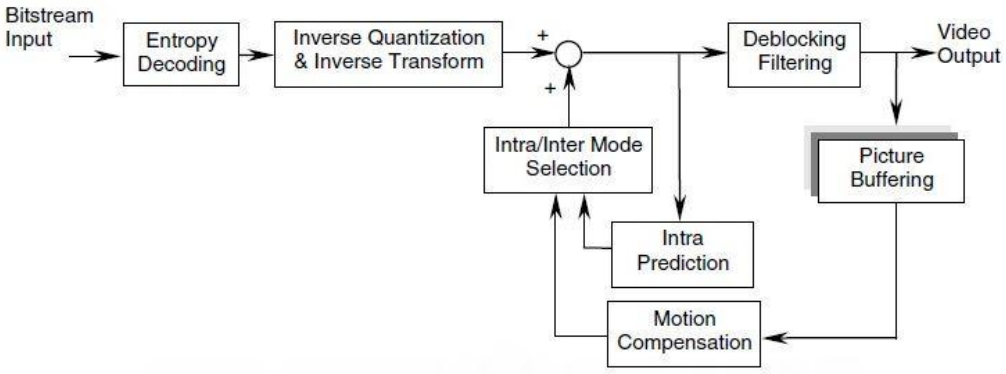


Figure 2.12 H.264/AVC decoder block diagram [12]

Chapter 3

HEVC

3.1 Introduction

High Efficiency Video Coding (HEVC) standard is the most recent joint video standardization project of ITU-T Video Coding Experts Group (VCEG) and ISO/IEC Moving Picture Experts Group (MPEG), currently under development in a collaboration known as the Joint Collaborative Team on Video Coding (JCT-VC). The first edition of the HEVC standard is expected to be finalized in January 2013, resulting in an aligned text that will be published by both ITU-T and ISO/IEC. In ISO/IEC, the HEVC standard will become MPEG-H Part 2 and in ITU-T it is likely to become ITU-T Recommendation H.265 [13].

The major video coding standard directly preceding the HEVC project was H.264/MPEG-4 AVC, which was initially developed during 1999-2003, and then was extended in several important ways during 2003-2009. It has been widely used for many applications, including broadcast of high definition (HD) TV signals over satellite, cable, and terrestrial transmission systems, video content acquisition and editing systems, camcorders, security applications, internet and mobile network video, blu-rays discs, and real-time conversational applications such as video chat, video conferencing and telepresence systems. However, an increasing diversity of services, the growing popularity of HD video, and the emergence of beyond-HD formats (4kx2k or 8kx4k resolution) are creating even stronger needs for coding efficiency superior to H.264's capabilities. The need is even stronger when higher resolution is accompanied by stereo or multi-view capture and display. The traffic caused by video appplications targeting mobile devices and tablet-PCs, as well as the transmission needs for video on demand services, are imposing severe challenges on today's networks. An increased need for

higher quality and resolutions is also arising in mobile applications. HEVC has been designed to address essentially all existing applications of H.264 and to particularly focus on two key issues: increased video resolution and increased use of parallel processing architectures.

3.2 Profiles, Levels and Tiers

Profiles, tiers and levels specify conformance points for implementing the standard in an interoperable way across various applications of the standard that have similar functional requirements. A profile defines a set of coding tools or algorithms that can be used in generating a conforming bitstream, whereas a level places constraints on certain key parameters of the bitstream, corresponding to decoder processing load and memory capabilities. Level restrictions are established in terms of maximum sample rate, maximum picture size, maximum bit rate, minimum compression ratio and capabilities of the DPB and the coded picture buffer (CPB) that holds compressed data prior to its decoding for data flow management purposes. In the design of HEVC, it was determined that some applications existed that had requirements that differed only in terms of maximum bit rate and CPB capabilities. To resolve this issue, two tiers were specified for some levels – a “Main” tier for most applications and a “High” tier for use in the most demanding applications.

Currently, only one profile, called the “Main” profile is foreseen to be defined in the first version of HEVC. Other profiles of the standard are expected to be specified in the first version or later. Currently, the definition of 13 levels is planned to be included in the first version of the standard as shown in Table 3.1. There are two tiers supported for eight of these levels (levels 4 and higher). The CPB capacity is equal to the maximum bit rate times 1 second for all levels except level 1, which has a higher CPB capacity of 350,000 bits. The specified maximum DPB capacity in each level is 6 pictures when

operating at the maximum picture size supported by the level. When operating with a smaller picture size than the maximum size supported by the level, the DPB picture storage capacity can increase to as many as 16 pictures (depending on the particular selected picture size).

Table 3.1 Level limits for the Main profile [13]

Level	Max luma picture size (samples)	Max luma sample rate (samples/sec)	Main Tier Max bit rate (1000 bits/s)	High Tier Max bit rate (1000 bits/s)	Min comp. ratio
1	36,864	552,960	128	-	2
2	122,880	3,686,400	1,500	-	2
2.1	245,760	7,372,800	3,000	-	2
3	552,960	16,588,800	6,000	-	2
3.1	983,040	33,177,600	10,000	-	2
4	2,228,224	66,846,720	12,000	30,000	4
4.1	2,228,224	133,693,440	20,000	50,000	4
5	8,912,896	267,386,880	25,000	100,000	6
5.1	8,912,896	534,773,760	40,000	160,000	8
5.2	8,912,896	1,069,547,520	60,000	240,000	8
6	33,423,360	1,069,547,520	60,000	240,000	8
6.1	33,423,360	2,005,401,600	120,000	480,000	8
6.2	33,423,360	4,010,803,200	240,000	800,000	6

3.3 HEVC Encoder

The video coding layer of HEVC employs the same “hybrid” approach (inter-/intra-picture prediction and 2D transform coding) used in all video compression standards since H.261. Figure 3.1 depicts the block diagram of a hybrid video encoder, which could create a bitstream conforming to the HEVC standard. An encoding algorithm producing an HEVC compliant bitstream would typically proceed as follows. Each picture is split into block-shaped regions, with the exact block partitioning being conveyed to the decoder. The first picture of a video sequence is coded using only intra-picture prediction. For all remaining pictures of a sequence or between random access points, inter-picture temporally-predictive coding modes are typically used for most blocks. The encoding process for inter-picture prediction consists of choosing motion data comprising the

selected reference picture and motion vector (MV) to be applied for predicting the samples of each block. The encoder and decoder generate identical inter prediction signals by applying motion compensation (MC) using the MV and mode decision data, which are transmitted as side information.

The residual signal of the intra or inter prediction, which is the difference between the original block and its prediction, is transformed by a linear spatial transform. The transform coefficients are then scaled, quantized, entropy coded and transmitted together with the prediction information. The encoder duplicates the decoder processing loop such that both will generate identical predictions for subsequent data. Therefore, the quantized transform coefficients are constructed by inverse scaling and are then inverse transformed to duplicate the decoded approximation of the residual signal. The residual is then added to the prediction, and the result of that addition may then be fed into one or two loop filters to smooth pout artifacts induced by the block-wise processing and quantization. The final picture representation (which is the duplicate of the output of the decoder) is stored in a decoded picture buffer (DPB) to be used for the prediction of subsequent pictures. In general, the order of the encoding or decoding processing of pictures often differs from the order in which they arrive from the source; necessitating a distinction between the decoding order (bitstream order) and the output order (display order) for a decoder.

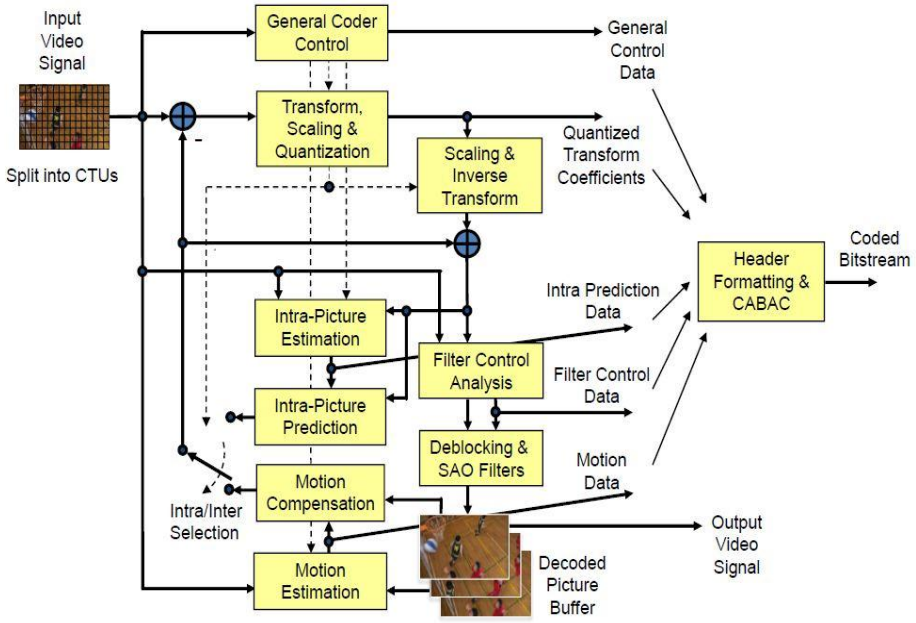


Figure 3.1 A typical HEVC video encoder [13]

3.3.1 Division of a picture into coding tree units

A picture is partitioned into coding tree units (CTUs), which each contain luma coding tree blocks (CTBs) and chroma CTBs. Lima CTB covers a rectangular picture area of LxL samples of the luma component and the corresponding chroma CTBs cover each L/2xL/2 samples of each of the two chroma components. The value of L may be equal to 16, 32 or 64. HEVC supports variable-size CTBs selected according to the needs of encoders in terms of memory and computational requirements. The support of larger CTBs than in previous standards is particularly beneficial when encoding high-resolution video content. The luma CTB and the two chroma CTBs together with the associated syntax form a coding tree unit (CTU). The CTU is the basic processing unit used in the standard to specify the decoding process.

The blocks specified as luma and chroma CTBs can be directly used as coding blocks (CBs) or can be further partitioned into multiple CBs. The partitioning is achieved

using tree structures. The boundaries of the picture are defined in units of the minimum allowed luma CB size. As a result, at the right and bottom edges of the picture, some CTUs may cover regions that are partly outside the boundaries of the picture. This condition is detected by the decoder, and the CTU quadtree is implicitly split as necessary to reduce the CB size to the point where the entire CB will fit into the picture.

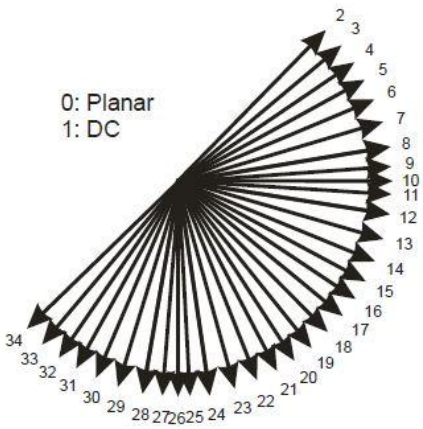
3.3.2 Intra prediction

The prediction mode for the CU is signaled as being intra or inter, according to whether it uses intra-picture (spatial) prediction or inter-picture (temporal) prediction. When the prediction mode is signaled as intra, the block size at which the intra prediction mode is established is the same as the CB size for all block sizes except for the smallest CB size that is allowed in the bitstream. For the latter case, a flag is present that indicates whether the CB is split into 4 quadrants that each has their own intra prediction mode. An intra-coded CB of size MxM may have one of the two types of prediction block (PB) partitions referred to as PART_2Nx2N, which indicates that CB is not split and PART_NxN, which indicates that the CB is split into four equal size PBs. HEVC design only allows the partitioning type PART_NxN to be used when the current CB size is equal to minimum CU size. This means that the PB size is always equal to the PB size when the CB is coded using an intra prediction mode and the CB size is not equal to the minimum CU size.

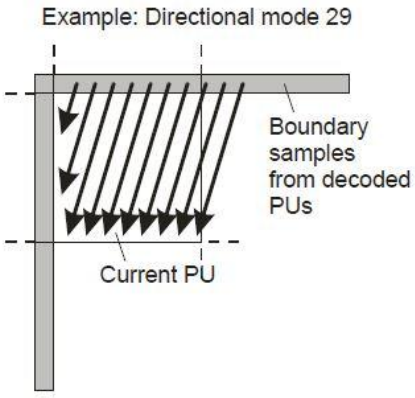
HEVC supports a total of 33 directional intra predictions denoted as Intra_Angular [k] where k is a mode number from 2 to 34. The angles are intentionally designed to provide denser coverage for near-horizontal and near-vertical angles and coarser coverage for near-diagonal angles. Figure 3.2 (a) shows modes and directional orientations for intra-picture prediction. Figure 3.2 (b) shows an example of a directional mode for an 8x8 block. Figure 3.2 (c) illustrates the intra predictions mapped to the

angular directions. When using an Intra_Angular mode each PB is predicted directionally from spatially neighboring samples which are reconstructed before being used for this prediction. For a PB of size NxN, a total of 4N+1spatially neighboring samples may be used for prediction. The prediction process of the Intra_Angular modes is consistent across all block sizes and prediction directions. In addition to intra angular predictions, HEVC supports two alternate prediction methods:

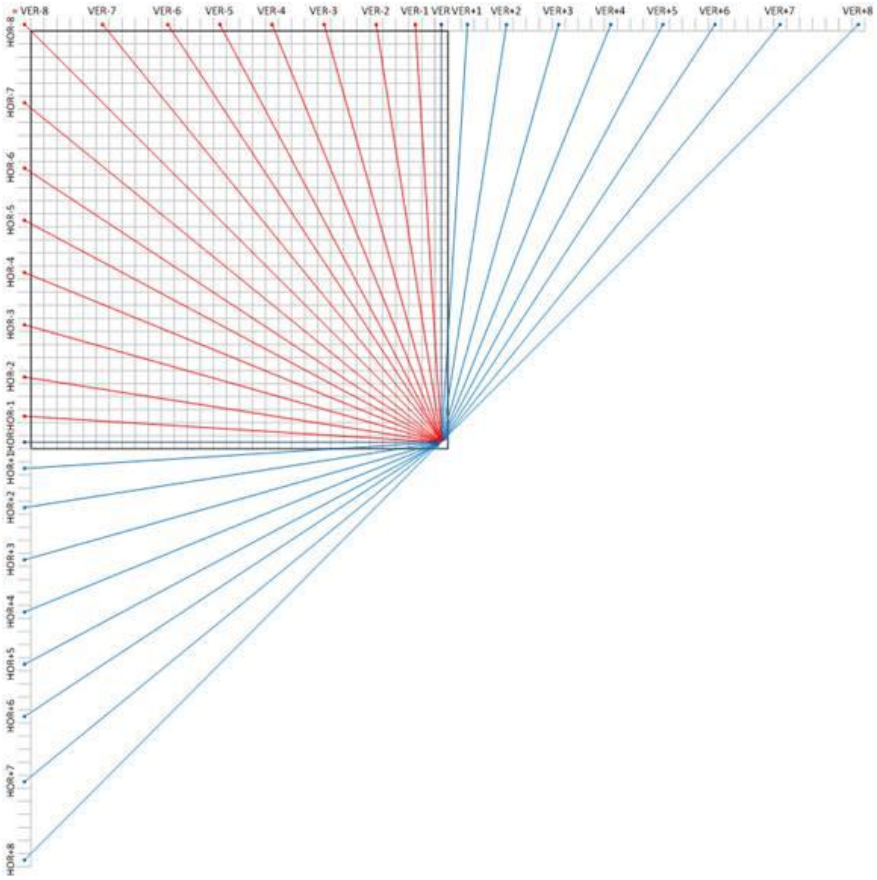
- Intra_DC prediction: Uses an average value of reference samples for the prediction.
- Intra_Planar prediction: Averages values of two linear predictions using four cornered reference samples to prevent discontinuities along the block boundaries.



(a)



(b)



(c)

Figure 3.2 (a) Modes and directional orientations for intra-picture prediction [13] (b) Directional mode for an 8x8 block [13] (c) Illustration of Intra prediction directions with mode names mapped to the angular directions [40]

3.3.2 Inter prediction

Compared with intra-coded CBs, HEVC supports more PB partition shapes for inter-coded CBs. Figure 3.3 shows modes for splitting a CB into PBs in case of inter prediction. The partitioning modes of PART_2Nx2N, PART_2NxN and PART_Nx2N indicate the cases when the CB is not split, split into two equal-size PBs horizontally and split into two equal-sized PBs vertically, respectively. PART_NxN specifies that the CB is split into four equal-sized PBs, but this mode is only supported when the CB size is equal to the smallest allowed CB size. Additionally, there are four partitioning types that support

splitting the CB into two PBs having different sizes: PART_2NxnU, PART_2NxnD, PART_nLx2N and PART_nRx2N. These types are known as “asymmetric motion partitions”.

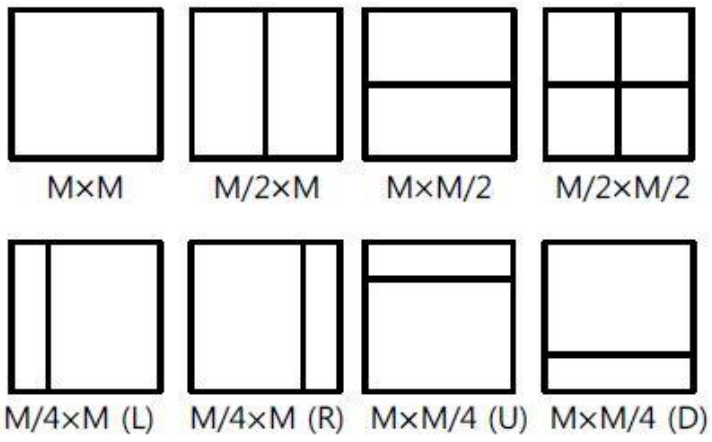


Figure 3.3 Modes for splitting CB into PBs in case of inter prediction. In the case of intra prediction, only MxM and M/2xM/2 (when M/2=4) are supported [13]

Each inter-coded PB is assigned one or two motion vectors and reference picture indices. To minimize worst-case memory bandwidth, PBs of size 4x4 are not allowed for inter-prediction, and PBs of size 4x8 and 8x4 are restricted to uni-predictive coding.

The samples of the PB for an inter-coded CB are obtained from those of a corresponding block region in the reference picture identified by a reference picture index, which is at a position displaced by the horizontal and vertical components of the motion vector. Except for the case when the motion vector has an integer value, fractional sample interpolation is used to generate prediction samples for non-integer sampling positions. HEVC supports motion vectors with units of one quarter of the distance between luma samples. For chroma samples, the motion vector accuracy is determined according to the chroma sampling format, which for 4:2:0 sampling results in units of one-eighth of the distance between chroma samples. The fractional sample

interpolation for luma samples in HEVC uses separable application of an 8-tap filter for the half-sample positions and a 7-tap filter for the quarter-sample positions.

3.3.4 Transform coding

For residual block coding, a CB can be recursively partitioned into transform blocks (TBs). The largest possible TB size is equal to the CB size. The partitioning itself is signaled by a residual quadtree. Only square partitioning is specified, where a block can be recursively split into quadrants as illustrated in Figure 3.4. For a given luma CB of size MxM, flag signals whether it is split into four blocks of size M/2xM/2. Since the minimum TB size is 4x4, splitting that would result in a transform size smaller than this is not supported. In the case of intra-predicted CUs, the decoded samples of the nearest TBs (within or outside the CB) are used as reference data for intra prediction. In contrast to previous standards, the HEVC design allows a TB to span across multiple PBs for inter-predicted CUs to maximize the potential coding efficiency benefits of quadtree structured TB partitioning. The possible TB sizes in HEVC are 4x4, 8x8, 16x6 and 32x32.

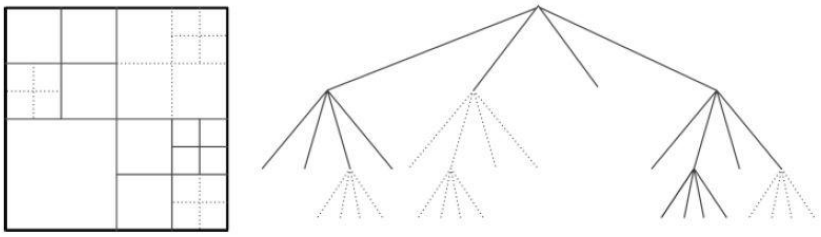


Figure 3.4 Subdivision of a CTB into CBs and TBs. Solid lines indicate CB boundaries and dotted lines indicate TB boundaries. Left: the CTB with its partitioning, right: the corresponding quadtree [13]

Two dimensional transforms are computed by applying one-dimensional transforms in both horizontal and vertical directions. The elements of the core transform matrices were derived by approximating scaled discrete cosine transform (DCT) basis functions, under considerations such as limiting the necessary dynamic range for

transform computation and maximizing the precision and closeness to orthogonality when the matrix entries are specified as integer values. For simplicity, only one integer matrix for the length of 32 points is specified, and sub-sampled versions are used for other sizes. For example, the matrix for the length-16 transform is as shown in Figure 3.5.

$$H = \begin{bmatrix}
64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 & 64 \\
90 & 87 & 80 & 70 & 57 & 43 & 25 & 9 & -9 & -25 & -43 & -57 & -70 & -80 & -87 & 90 \\
89 & 75 & 50 & 18 & -18 & -50 & -75 & -89 & -89 & -75 & -50 & -18 & 18 & 50 & 75 & 89 \\
87 & 57 & 9 & -43 & -80 & -90 & -70 & -25 & 25 & 70 & 90 & 80 & 43 & -9 & -57 & -87 \\
83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 & 83 & 36 & -36 & -83 & -83 & -36 & 36 & 83 \\
80 & 9 & -70 & -87 & -25 & 57 & 90 & 43 & -43 & -90 & -57 & 25 & 87 & 70 & -9 & -80 \\
75 & -18 & -89 & -50 & 50 & 89 & 18 & -75 & -75 & 18 & 89 & 50 & -50 & -89 & -18 & 75 \\
70 & -43 & -87 & 9 & 90 & 25 & -80 & -57 & 57 & 80 & -25 & -90 & -9 & 87 & 43 & -70 \\
64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 & 64 & -64 & -64 & 64 \\
57 & -80 & -25 & 90 & -9 & -87 & 43 & 70 & -70 & -43 & 87 & 9 & -90 & 25 & 80 & -57 \\
50 & -89 & 18 & 75 & -75 & -18 & 89 & -50 & -50 & 89 & -18 & -75 & 75 & 18 & -89 & 50 \\
43 & -90 & 57 & 25 & -87 & 70 & 9 & -80 & 80 & -9 & -70 & 87 & -25 & -57 & 90 & -43 \\
36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 & 36 & -83 & 83 & -36 & -36 & 83 & -83 & 36 \\
25 & -70 & 90 & -80 & 43 & 9 & -57 & 87 & -87 & 57 & -9 & -43 & 80 & -90 & 70 & -25 \\
18 & -50 & 75 & -89 & 89 & -75 & 50 & -18 & -18 & 50 & -75 & 89 & -89 & 75 & -50 & 18 \\
9 & -25 & 43 & -57 & 70 & -80 & 87 & -90 & 90 & -87 & 80 & -70 & 57 & -43 & 25 & -9
\end{bmatrix}.$$

Figure 3.5 The core transform matrix used in HEVC for the length-16 transform [13]

The matrices for the length-8 and length-4 transforms can be derived by using the first 8 entries of rows 0,2,4,…, and using the first 4 entries of rows 0,4,8… respectively. Although the standard specifies the transform simply in terms of the value of a matrix, the values of the entries in the matrix were selected to have key symmetry properties that enable fast “partially-factored” implementations with far fewer mathematical operations than an ordinary matrix multiplication, and the larger transforms can be constructed by using the smaller transforms as building blocks.

Due to the increased size of the supported transforms, limiting the dynamic range of the intermediate results from the first stage of the transformation is quite important. HEVC explicitly inserts a 7-bit right shift and 16-bit clipping operation after the first one-dimensional inverse transform stage of the transform (the vertical inverse transform stage), to ensure that all intermediate values can be stored in a 16 bit memory (for 8-bit video decoding).

For the transform block size of 4x4, an alternate integer transform derived from a discrete sine transform (DST) is applied to the luma residual blocks for intra prediction modes, with the transform matrix shown in Figure 3.6.

$$H = \begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix}$$

Figure 3.6 4x4 DST matrix in HEVC [13]

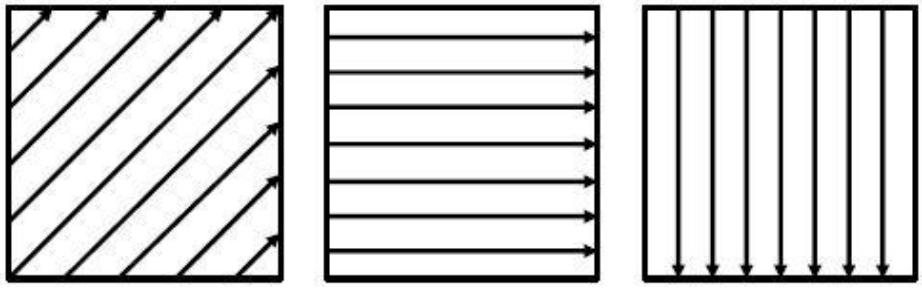
With the basis functions of the DST, the property is better modeled that the residual amplitudes tend to increase as the distance from the boundary samples which are used for prediction becomes larger. In terms of complexity, the 4x4 DST-style transform is not much more computationally demanding than the 4x4 DCT-style transform, and it provides approximately 1% bit rate reduction in intra-predictive coding. The usage of the DST type of transform is restricted to only 4x4 luma transform blocks, since for other cases, the additional coding efficiency improvement for including the additional transform type was found to be marginal.

Since the rows of the transform matrix are close approximations of values of uniformly-scaled basis functions of the orthonormal DCT, the pre-scaling operation that is incorporated in the dequantization of H.264 is not needed in HEVC.

3.3.5 Entropy coding

HEVC specifies only one entropy coding method, context adaptive binary arithmetioc coding (CABAC). Appropriate selection of context modeling is known to be a key factor to improve the efficiency of CABAC coding. In HEVC, the splitting depth of the coding tree or transform tree is exploited to derive the context model indices of various syntax elements in addition to the spatially neighboring ones used in H.264/AVC. Although the number of contexts used in HEVC is substantially less than in H.264, the entropy coding design actually provides better compression. More extensive use is made in HEVC of the bypass-mode of CABAC operation, to increase throughput by reducing the amount of data that needs to be coded using CABAC contexts.

Coefficient scanning is performed in 4x4 sub-blocks for all TB sizes. Three coefficient scanning methods, diagonal up-right, horizontal and vertical scans as shown in Figure 3.7, are selected implicitly for coding the intra-coded transform coefficients of 4x4 and 8x8 TB sizes. The selection of the coefficient scanning order depends on the directionalities of the intra prediction. When the direction is close to horizontal direction, the vertical scan is used and the horizontal scan is used when the direction is close to the vertical direction. For other prediction directions, the diagonal up-right scan is used. For the transform coefficients in inter prediction modes of all block sizes and for the transform coefficients of 16x16 or 32x32 intra prediction, the 4x4 diagonal up-right scan is exclusively applied to sub-blocks of transform coefficients.



(a) (b) (c)

Figure 3.7 Three coefficient scanning methods in HEVC (a) diagonal up-right scan (b) horizontal scan and (c) vertical scan [13]

3.3.6 In-loop filtering

In HEVC, two processing steps, namely, a deblocking filter (DBF) followed by a sample adaptive offset (SAO) operation are applied to the reconstructed samples before writing them into the DPB in the decode loop. The DBF is intended to reduce the blocking artifacts due to block-based coding. The DBF is similar to that of H.264, whereas SAO is newly introduced in HEVC. While the DBF is only applied to the samples located at block boundaries, the SAO operation is applied adaptively to all samples satisfying certain conditions, e.g. based on gradient. The deblocking filter is applied to all samples adjacent to a PU or TU boundary except the case when the boundary is also a picture boundary, or when deblocking is disabled across slice or tile boundaries. HEVC only applies the deblocking filter to the edges which are aligned on an 8x8 simple grid, for both the luma and chroma samples. The processing order of the deblocking filter is defined as horizontal filtering for vertical edges for the entire picture first, followed by vertical filtering for horizontal edges. SAO is a process which modifies the samples after the deblocking flter through a look-up table. It is a non-linear operation which allows additional minimization of the reconstruction error in a way that cannot be achieved by linear filters, and particularly is able to enhance edge sharpness. In addition, SAO is very efficient to

suppress pseudo-edges referred to as “banding artifacts” as well as the “ringing artifacts” coming from the quantization errors of high frequency components in the transform domain.

3.4 HEVC Decoder

The HEVC decoder is shown in Figure 3.8. The decoder operation is the same as in H.264. The overall complexity of the HEVC decoder does not appear to be significantly different from that of the H.264/AVC decoder, even with addition of an in-loop filter such as SAO, making HEVC decoding in software very practical on current hardware.

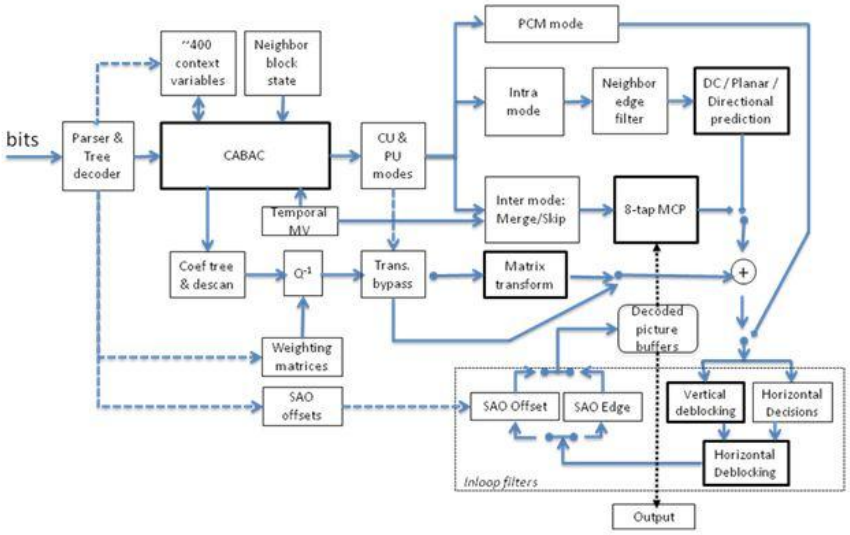


Figure 3.8 HEVC decoder block diagram [16]

Chapter 4

Implementation of Mode Dependent DCT/DST in H.264

4.1 Introduction

Mode dependent DCT/DST is a hybrid transform coding scheme which incorporates switching between sine and cosine transforms into the intra prediction mode, thus exploiting inter-block correlations. When prediction is performed from one side, the energy in prediction error residuals increases as we go away from the boundary. A sine transform is better adaptable to these prediction residual statistics [30]. It was shown that following intra prediction, the optimal transform for vertical (respectively horizontal) modes along the vertical (respectively horizontal) direction of prediction is the KLT of the autocorrelation matrix in [29] [32] calculated as the following sine transform which is DST-VII :

$$[T_S]_{i,j} = \frac{2}{\sqrt{2N+1}} \sin \frac{(2i-1)j\pi}{2N+1}$$

The ongoing HEVC standardization uses unified intra prediction in which up to 34 different directional intra prediction modes can be divided into 3 categories as follows:

- Category 1 oblique modes (as shown in Figure 4.1): Here prediction is performed from the decoded pixels from either the top row or the left column. The vertical mode and the horizontal mode are special cases of this oblique mode when prediction direction is vertical or horizontal respectively.
- Category 2 oblique modes (as shown in Figure 4.2): Here the prediction is performed from both the top row and the left column pixels.
- DC mode: Here prediction is performed from an average of all available decoded pixels.

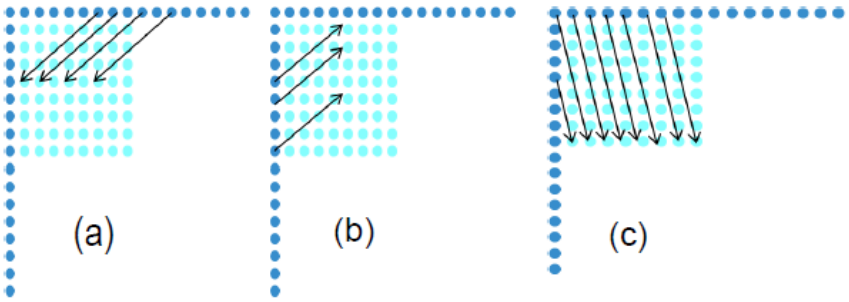


Figure 4.1 Category 1 oblique modes (a) Prediction from top row only (b) Prediction from left-column only (c) Category 2 oblique modes [31]

The mapping from unified intra prediction modes to DCT/DST used in earlier stages for HEVC standardization [30] [31] is shown in Table 4.1.

Table 4.1 Mapping from intra prediction modes to DCT/DST used in HM 2.0 [31]

Mode	Unified intra prediction direction	Vertical Transform	Horizontal Transform
0, 5, 6, 12, 13, 22, 23, 24, 25	VER to VER+8	DST	DCT
1, 8, 9, 16, 17, 30, 31, 32, 33	HOR to HOR+8	DCT	DCT
2	DC	DCT	DST
3, 4, 10, 11, 18, 19, 20, 21	VER-8 to VER-1	DST	DCT
7, 14, 15, 26, 27, 28, 29	HOR-8 to HOR-1	DST	DST

4.2 Transform implementation in the reference software

JM software [34] is the reference software for H.264/AVC standard. Fig. 4.2 shows flowgraphs for the fast implementation and forward and inverse DCT, which are applied to rows and columns of each 4x4 block in JM software [33]. No multiplications are needed, only additions and shifts. The DCT used in H.264, which is an integer approximation of DCT-II, maps a length-N vector x into a new vector X of transform coefficients by a linear transformation X = H x, where the element in the kth row and nth column of H is defined by

$$H(k,n) = c_k \sqrt{\frac{2}{N}} \cos\left[\left(n + \frac{1}{2} \right) \frac{k\pi}{N} \right]$$

for the frequency index k = 0, 1,…, N-1, and sample index n = 0, 1,…, N-1 with c0 = $\sqrt{2}$ and ck = 1 for k>0.

HM software [35] is the reference software for HEVC. Fast implementation of DST-VII is used in HM software, involving additions, multiplications and shifts.

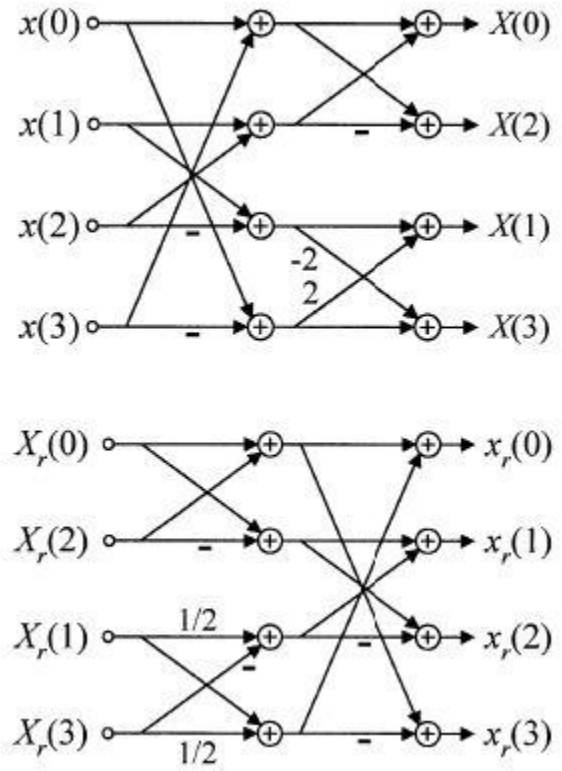


Figure 4.2 Fast implementation of H.264 forward transform (top) and inverse transform (bottom) [33]

4.3 Proposed Scheme

4.3.1 Mapping from intra prediction modes to DCT/DST

Similar to the mapping in HEVC, the nine intra prediction modes for 4x4 luma in H.264 can be classified into category 1 oblique modes, category 2 oblique modes and the

DC mode. The proposed mapping of intra prediction modes in H.264/AVC is shown in Table 4.2.

Table 4.2 Proposed mapping from intra prediction modes to DCT/DST in H.264/AVC

Mode	Intra prediction direction	Vertical Transform	Horizontal Transform
0,3,7	Vertical, Diagonal Down-Left, Vertical Right	DST	DCT
1,8	Horizontal, Horizontal-Up	DCT	DST
2	DC	DCT	DCT
4,5,6	Diagonal Down-Right, Vertical-Left, Horizontal-Down	DST	DST

4.3.2 Obtaining DST matrices for H.264

The forward DCT matrix used in H.264/AVC is:

$$\text{Forward DCT} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

Norms of basis vectors along the rows for the forward DCT matrix:

Row 1: 2

Row 2: $\sqrt{10}$ (3.1623)

Row 3: 2

Row 4: $\sqrt{10}$ (3.1623)

The inverse DCT matrix used in H.264/AVC is:

$$\text{Inverse DCT} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1/2 & -1/2 & -1 \\ 1 & -1 & -1 & 1 \\ 1/2 & -1 & 1 & -1/2 \end{bmatrix}$$

Norms of basis vectors along the rows for the inverse DCT matrix:

Row 1: 2

Row 2: $\sqrt{5/2}$ (1.5811)

Row 3: 2

Row 4: $\sqrt{5/2}$ (1.5811)

The DST matrix used in HEVC is:

$$\text{DST matrix} = \begin{bmatrix} 29 & 55 & 74 & 84 \\ 74 & 74 & 0 & -74 \\ 84 & -29 & -74 & 55 \\ 55 & -84 & 74 & -29 \end{bmatrix}$$

Norms of basis vectors along the rows for the DST matrix used in HEVC:

Row 1: $\sqrt{16398}$ (128.0547)

Row 2: $\sqrt{16428}$ (128.1718)

Row 3: $\sqrt{16398}$ (128.0547)

Row 4: $\sqrt{16398}$ (128.0547)

To implement the same DST matrix in H.264, the norm along each row of the DST matrix is made equal to the norm of the corresponding row of the DCT matrices.

To obtain the forward DST matrix for H.264, divide the rows of the DST used in HEVC by the factor

$$\frac{\text{Norm of row 'x' of DST matrix}}{\text{Norm of row 'x' of forward DCT matrix}}$$

The modified forward DST matrix obtained for use in H.264 is:

$$\text{Forward DST} = \begin{bmatrix} 0.4529 & 0.8590 & 1.1558 & 1.3119 \\ 1.8257 & 1.8257 & 0 & -1.8257 \\ 1.3119 & -0.4529 & -1.1558 & 0.8590 \\ 1.3582 & -2.0744 & 1.8274 & -0.7161 \end{bmatrix}$$

Norms of basis vectors along the rows for the modified forward DST matrix:

Row 1: 2

Row 2: 3.1622

Row 3: 2

Row 4: 3.1623

It is seen that the norms of the rows of the modified forward DST are the same as that of the forward DCT matrix.

To obtain the inverse DST matrix for H.264, divide the rows of the DST used in HEVC by the factor

$$\frac{\text{Norm of row 'x' of DST matrix}}{\text{Norm of row 'x' of inverse DCT matrix}}$$

The modified inverse DST matrix obtained for use in H.264 is:

$$\text{Inverse DST} = \begin{bmatrix} 0.4529 & 0.8590 & 1.1558 & 1.3119 \\ 0.9129 & 0.9129 & 0 & -0.9129 \\ 0.4529 & -0.8590 & -1.1558 & 1.3119 \\ 0.6791 & -1.0372 & 0.9137 & -0.3581 \end{bmatrix}$$

Norms of basis vectors along the rows for the modified inverse DST matrix:

Row 1: 2

Row 2: 1.5812

Row 3: 2

Row 4: 1.5812

It is seen that the norms of the rows of the modified forward DST are the same as that of the forward DCT matrix.

4.2.3 Implementation of DCT/DST in the reference software for H.264/AVC

The modified DST matrices are non-integer matrices and hence matrix multiplication has been implemented in the H.264 reference software JM 18.4 instead of fast implementation. Matrix multiplication is used for the proposed DCT/DST scheme. It is verified that matrix multiplication gives exactly the results as that of fast implementation of Integer DCT in H.264. Main profile is used for encoding the video sequences. The video

sequences can be downloaded from [36]. Encoding is performed using a system having Intel i7 Quad 4, 2.0 GHz processor, 8 GB RAM. The operating system used is Windows 7. Performance of default fast implementation in HEVC is also analyzed using HM 8.0, with video sequences encoded using the Intra Main profile. Video sequences belonging to HD (1920 x 1080, 1280 x 720), WVGA (832 x 480) and WQVGA (416 x 240) are used for evaluating the performance [39]. Only the first frame (I frame) of the video sequences are encoded. The results, bitrates versus PSNRs, percentage bit rate savings (BD-Bitrate) and absolute PSNR gain (BD-PSNR), for each sequence are tabulated. The RD curves for one sequence from each of the different resolutions are also shown. BD-PSNR and BD-Bitrate [37][38] are used to analyze the performance of the RD (rate distortion) curves.

The following parameter settings were changed in the encoder configuration file in JM18.4 for the analysis:

- FramesToBeEncoded = 1
- SourceWidth = 416, 832, 1280, 1920
- SourceHeight = 240, 480, 720, 1080
- OutputWidth = 416, 832, 1280, 1920
- OutputHeight = 240, 480, 720, 1080
- FrameRate = 30
- Profile IDC = 77
- LevelIDC = 50
- IntraPeriod = 0
- IDRPeriod = 0
- QPISlice = 16, 20, 24, 28
- RDOptimization = 1

- Transform8x8Mode = 0

The following parameter settings were changed in the encoder configuration file in HM 8.0 for the analysis:

- IntraPeriod = -1
- QP = 16, 20, 24, 28
- RDOQ = 1

The following parameter settings were changed in the input sequence configuration file in HM 8.0 for the analysis:

- FrameRate = 30
- FramesToBeEncoded = 1
- SourceWidth = 416, 832, 1280, 1920
- SourceHeight = 240, 480, 720, 1080

4.3 Calculation of BD-PSNR and BD-Bitrate

BD-PSNR and BD-Bitrate represent the average PSNR and bitrate differences respectively between two RD curves [37].

- Fit a curve through four data points corresponding to QP = 16, 20, 24, 28
- The data points are Y-PSNR values for BD PSNR and bitrate values for BD Rate
- A good interpolation curve through the four data points can be obtained by a third order polynomial of the form:

  $SNR = a + b*bit + c*bit^2 + d*bit^3$

where a, b, c, d are determined such that the curve passes through all the data points

- The difference between the RD curves is dominated by high bitrates. Hence logarithmic value of bitrate is considered for calculating BD Rate
- In the same way, we can do interpolation as a function of SNR as follows:

  $SNR = a + b*SNR + c*SNR^2 + d*SNR^3$

- Based on the interpolation , find an expression for the integral of the curve
- The average difference is the difference between the integrals divided by the integration interval

The MATLAB implementation for the calculation of BD-PSNR and BD-Bitrate from [38] is used for the analysis.

4.4 Performance analysis

4.4.1 Results for WQVGA (416x240) sequences

Table 4.3 Comparison of bitrates and PSNRs for three 416x240 sequences (H.264/AVC with DCT/DST)

Sequence Name	QP	Default H.264/AVC				H.264/AVC with DCT/DST			
		Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)	Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)
RaceHorses	16	10105.68	47.744	47.201	47.377	10306.32	47.155	47.201	47.377
	20	7556.16	44.084	44.030	44.497	7662.72	43.617	44.030	44.497
	24	5429.76	40.496	41.313	41.726	5476.08	40.198	41.313	41.726
	28	3792.24	37.097	38.894	39.324	3801.12	36.936	38.894	39.324
BlowingBubbles	16	11159.52	47.202	46.292	47.172	11401.2	46.494	46.292	47.172
	20	8124.00	43.127	43.187	44.521	8249.04	42.667	43.187	44.521
	24	5575.92	39.536	40.615	41.987	5617.68	39.253	40.615	41.987
	28	3744.00	36.385	38.323	39.879	3730.80	36.234	38.323	39.879
BQSquare	16	11502.72	47.337	47.540	47.689	11832.48	46.704	47.541	47.689
	20	8756.64	43.550	44.761	45.304	8965.44	43.090	44.761	45.304
	24	6485.76	39.948	42.499	43.097	6625.20	39.483	42.499	43.097
	28	4740.72	36.517	40.761	41.378	4814.16	36.227	40.761	41.378

Table 4.4 BD-PSNR and BD-Bitrate (H.264/AVC with DCT/DST)

Sequence Name	BD-PSNR (dB)	BD-Bitrate (%)
RaceHorses	-0.4913	4.775
BlowingBubbles	-0.4852	5.1894
BQSquare	-0.7315	6.2691

Table 4.4 Comparison of bitrates and PSNRs for three 416x240 sequences (H.264/AVC with HEVC)

Sequence Name	QP	Default H.264/AVC			Default HEVC		
		Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR V-PSNR	Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR V-PSNR
RaceHorses	16	10105.68	47.744	47.201 47.377	8411.52	47.3017	47.2436 47.3423
	20	7556.16	44.084	44.030 44.497	6175.20	43.8264	44.3700 44.7900
	24	5429.76	40.496	41.313 41.726	4414.56	40.5403	41.5526 42.2303
	28	3792.24	37.097	38.894 39.324	2962.80	37.1592	39.0778 39.5112
BlowingBubbles	16	11159.52	47.202	46.292 47.172	9757.92	46.8707	46.1636 47.3226
	20	8124.00	43.127	43.187 44.521	6872.88	42.8407	43.4216 44.8203
	24	5575.92	39.536	40.615 41.987	4662.96	39.4538	40.8938 42.4269
	28	3744.00	36.385	38.323 39.879	2977.44	36.3220	38.3301 39.8608
BQSquare	16	11502.72	47.337	47.540 47.689	9799.92	46.9003	47.6912 47.8703
	20	8756.64	43.550	44.761 45.304	7368.48	43.2459	44.8453 45.4624
	24	6485.76	39.948	42.499 43.097	5380.56	39.7239	42.4497 43.1106
	28	4740.72	36.517	40.761 41.378	3788.88	36.2686	40.4929 41.1079

Table 4.5 BD-PSNR and BD-Bitrate for three 416x240 sequences (H.264/AVC with HEVC)

Sequence Name	BD-PSNR (dB)	BD-Bitrate (%)
RaceHorses	2.0212	-17.735
BlowingBubbles	1.4722	-14.2433
BQSquare	1.8785	-14.6427

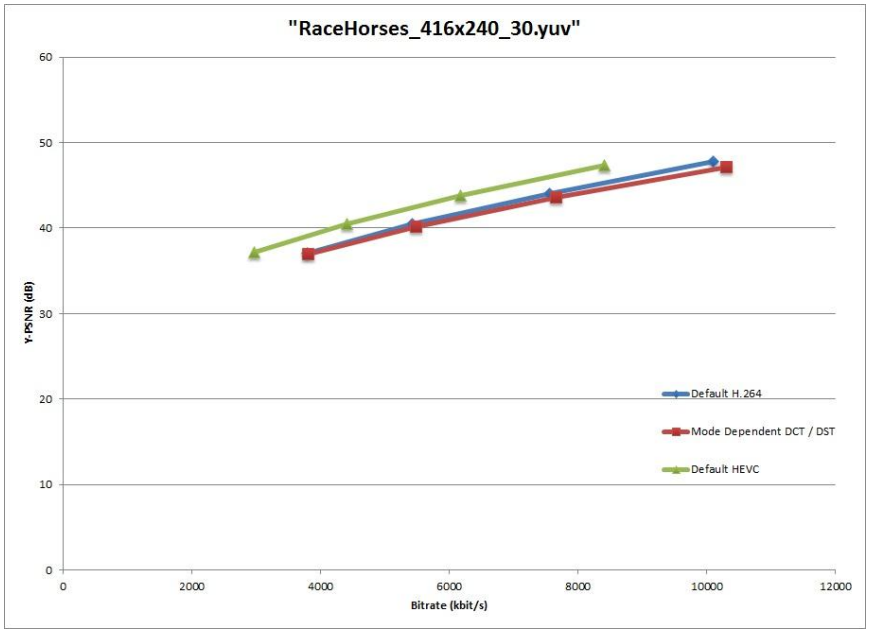


Figure 4.3 Y-PSNR variations with bitrate for RaceHorses sequence

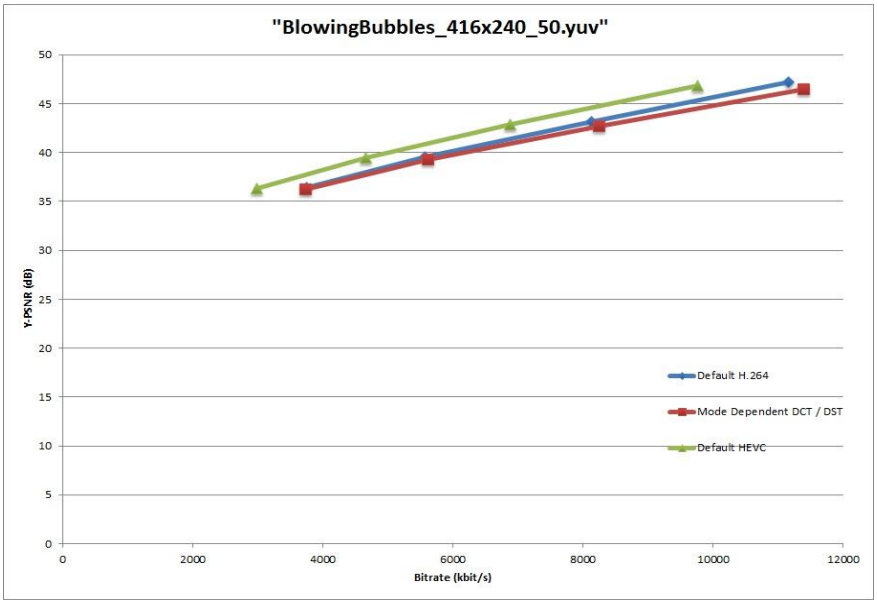


Figure 4.4 Y-PSNR variations with bitrate for BlowingBubbles sequence

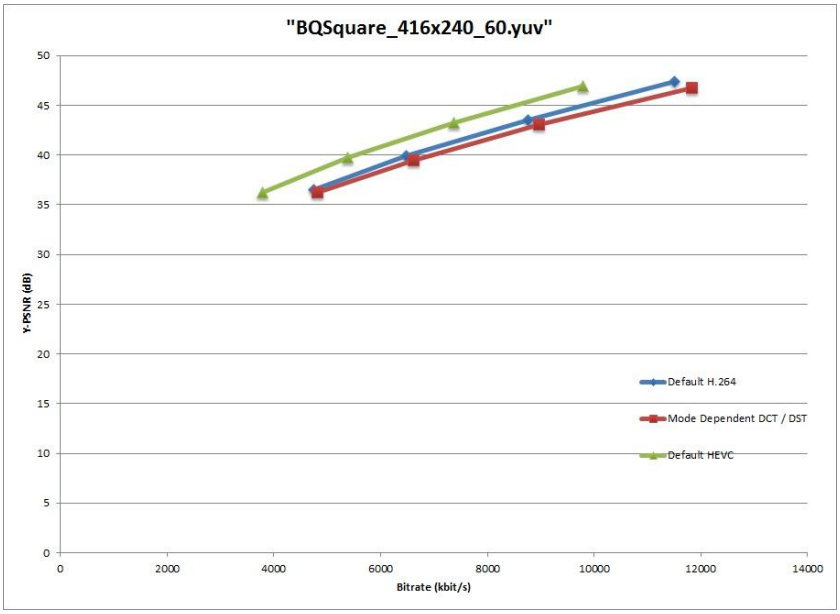


Figure 4.5 Y-PSNR variations with bitrate for BQSquare sequence

4.4.2 Results for WVGA (832x480) sequences

Table 4.6 Comparison of bitrates and PSNRs for three 832x480 sequences (H.264/AVC with DCT/DST)

Sequence Name	QP	Default H.264/AVC				H.264/AVC with DCT/DST			
		Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)	Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)
BQMall	16	33409.68	46.858	46.79	47.763	34025.28	46.343	46.790	47.763
	20	22413.84	43.270	44.406	45.660	22548.48	42.967	44.406	45.660
	24	14934.96	40.543	42.352	43.492	14905.20	40.278	42.352	43.492
	28	10274.40	37.901	40.496	41.472	10238.16	37.641	40.496	41.472
Keiba	16	23370.24	47.054	48.831	49.288	22886.88	46.626	48.831	49.288
	20	15057.36	43.808	46.659	47.422	14381.04	43.611	46.659	47.422
	24	9540.24	41.217	44.528	45.296	9150.24	41.136	44.528	45.296
	28	6162.48	38.683	42.594	43.321	5952.24	38.594	42.594	43.321
PartyScene	16	52926.96	47.397	46.547	47.006	54041.28	46.569	46.547	47.006
	20	40601.28	43.345	43.249	43.794	41196.00	42.705	43.249	43.794
	24	29612.16	39.492	40.323	40.846	29968.08	39.060	40.323	40.846
	28	20975.76	36.073	37.935	38.350	21224.40	35.760	37.935	38.350

Table 4.7 BD-PSNR and BD-Bitrate for three 832x480 sequences (H.264/AVC with DCT/DST)

Sequence Name	BD-PSNR (dB)	BD-Bitrate (%)
BQMall	-0.3462	4.6475
Keiba	0.0665	-1.1473
PartyScene	-0.6996	6.0669

Table 4.8 Comparison of bitrates and PSNRs for three 832x480 sequences (H.264/AVC with HEVC)

Sequence Name	QP	Default H.264/AVC				Default HEVC			
		Bitrate (kbit/s)	Y-PSNR	U-PSNR	V-PSNR (dB)	Bitrate (kbit/s)	Y-PSNR	U-PSNR	V-PSNR (dB)
BQMall	16	33409.68	46.858	46.79	47.763	28161.84	46.5165	46.6184	47.8058
	20	22413.84	43.270	44.406	45.660	17480.40	42.9236	44.4668	45.7898
	24	14934.96	40.543	42.352	43.492	11940.24	40.4678	42.4175	43.6628
	28	10274.40	37.901	40.496	41.472	8022.48	37.8871	40.4575	41.4867
Keiba	16	23370.24	47.054	48.831	49.288	19457.28	46.7680	49.0376	49.5021
	20	15057.36	43.808	46.659	47.422	11888.64	43.7236	46.8391	47.4610
	24	9540.24	41.217	44.528	45.296	7589.76	41.3272	44.6684	45.4265
	28	6162.48	38.683	42.594	43.321	4845.36	38.9652	42.5052	43.2928
PartyScene	16	52926.96	47.397	46.547	47.006	47571.60	47.0839	46.2439	46.7408
	20	40601.28	43.345	43.249	43.794	35791.44	42.9796	43.3825	43.9579
	24	29612.16	39.492	40.323	40.846	25742.40	39.2256	40.6084	41.0488
	28	20975.76	36.073	37.935	38.350	17608.56	35.7772	37.9938	38.4236

Table 4.9 BD-PSNR and BD-Bitrate three 832x480 sequences (H.264/AVC with HEVC)

Sequence Name	BD-PSNR (dB)	BD-Bitrate (%)
BQMall	1.4754	-17.985
Keiba	1.3704	-20.3635
PartyScene	1.2773	-10.2026

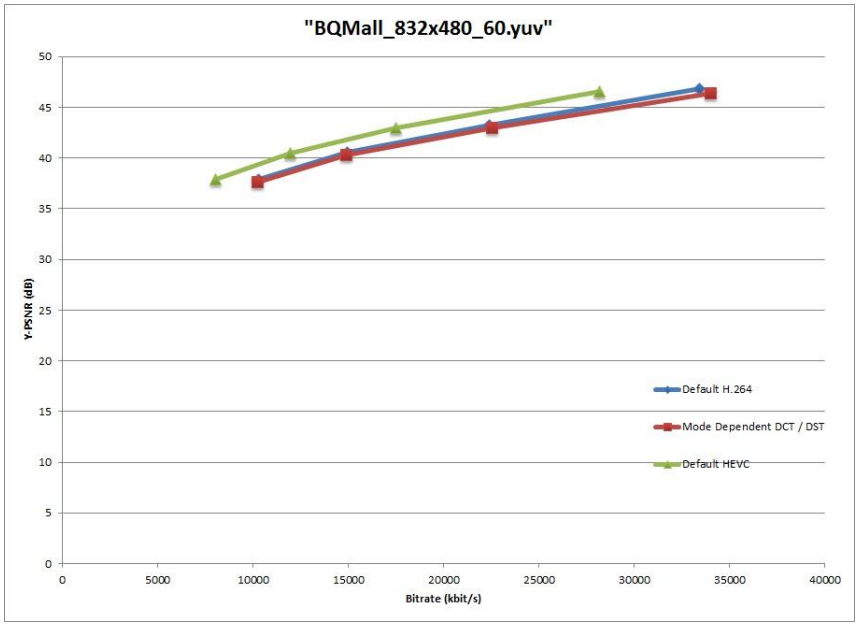


Figure 4.6 Y-PSNR variations with bitrate for BQMall sequence

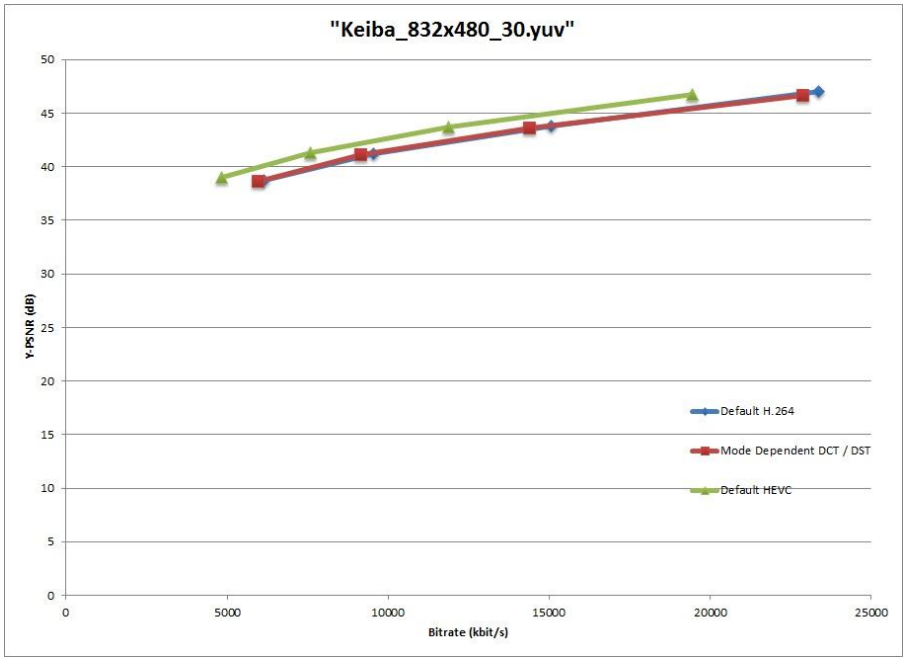


Figure 4.7 Y-PSNR variations with bitrate for Keiba sequence

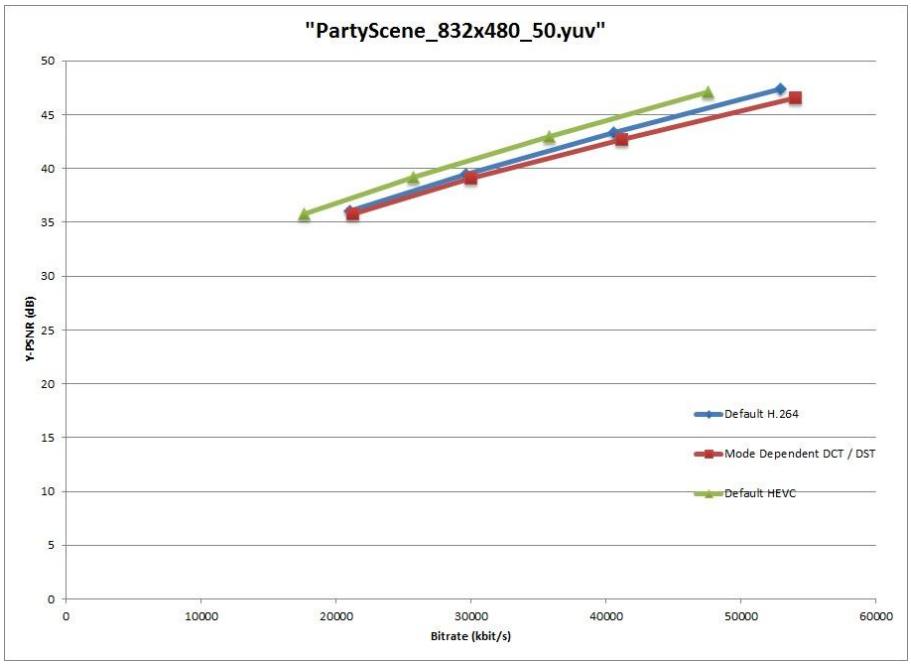


Figure 4.8 Y-PSNR variations with bitrate for PartyScene sequence

4.4.3 Results for HD (1920x1080) sequences

Table 4.10 Comparison of bitrates and PSNRs for three 1920x1080 sequences (H.264/AVC with DCT/DST)

Sequence Name	QP	Default H.264/AVC				H.264/AVC with DCT/DST			
		Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)	Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)
BQTerrace	16	180787.68	47.918	46.751	47.157	186144.72	47.188	46.751	47.157
	20	129416.88	43.886	43.906	45.222	132517.44	43.322	43.906	45.222
	24	86247.36	40.016	41.802	43.543	87560.64	39.665	41.802	43.543
	28	55845.60	36.873	40.042	42.078	56470.80	36.610	40.042	42.078
Cactus	16	181654.08	47.066	45.912	46.730	183171.12	46.419	45.912	46.730
	20	115491.36	42.578	42.644	44.690	115657.20	42.303	42.644	44.690
	24	65324.88	39.476	40.414	42.913	64902.24	39.390	40.414	42.913
	28	39115.92	37.389	39.039	41.249	38875.68	37.360	39.039	41.249
Tennis	16	104600.16	46.513	46.915	48.134	103009.20	46.183	46.915	48.134
	20	55239.84	42.994	45.378	46.868	54156.48	42.919	45.378	46.868
	24	28356.96	41.275	44.124	45.431	27876.96	41.243	44.124	45.431
	28	17067.12	39.592	42.906	43.879	16907.28	39.534	42.906	43.879

Table 4.11 BD-PSNR and BD-Bitrate for three 1920x1080 sequences (H.264/AVC with DCT/DST)

Sequence Name	BD-PSNR (dB)	BD-Bitrate (%)
BQTerrace	-0.6369	7.2692
Cactus	-0.2102	3.5532
Tennis	-0.0290	-0.4242

Table 4.12 Comparison of bitrates and PSNRs for three 1920x1080 sequences (H.264/AVC with HEVC)

Sequence Name	QP	Default H.264/AVC				Default HEVC			
		Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)	Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)
BQTerrace	16	180787.68	47.918	46.751	47.157	149503.44	47.9929	46.7115	47.3451
	20	129416.88	43.886	43.906	45.222	108491.76	44.6199	44.0397	45.3542
	24	86247.36	40.016	41.802	43.543	72256.56	40.6303	42.0087	43.7107
	28	55845.60	36.873	40.042	42.078	42464.64	37.0648	40.1334	42.1350
Cactus	16	181654.08	47.066	45.912	46.730	159421.68	46.8147	45.7602	46.8177
	20	115491.36	42.578	42.644	44.690	95585.76	42.5403	42.8707	44.8581
	24	65324.88	39.476	40.414	42.913	46632.96	39.3158	40.4718	43.1484
	28	39115.92	37.389	39.039	41.249	26866.32	37.4310	39.0336	41.4139
Tennis	16	104600.16	46.513	46.915	48.134	87396.24	46.4929	47.2047	48.5307
	20	55239.84	42.994	45.378	46.868	34297.68	42.8673	45.4749	47.0398
	24	28356.96	41.275	44.124	45.431	18642.48	41.5340	44.3207	45.6447
	28	17067.12	39.592	42.906	43.879	11315.28	40.2274	43.0237	44.0927

Table 4.13 BD-PSNR and BD-Bitrate for three 1920x1080 sequences (H.264/AVC with HEVC)

Sequence Name	BD-PSNR (dB)	BD-Bitrate (%)
BQTerrace	2.1567	-21.9681
Cactus	1.3025	-19.1702
Tennis	1.4856	-33.1268

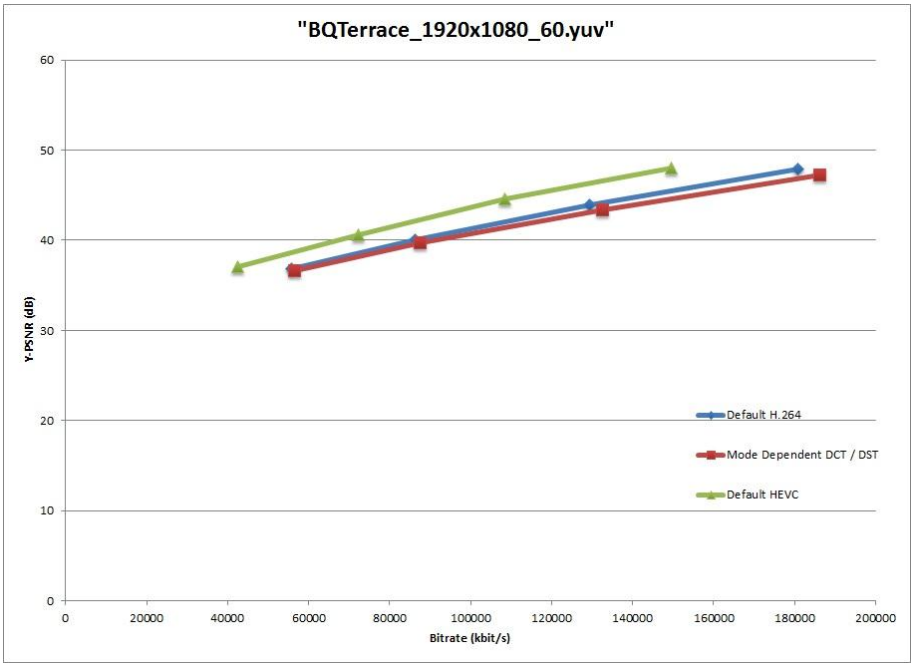


Figure 4.9 Y-PSNR variations with bitrate for BQTerrace sequence

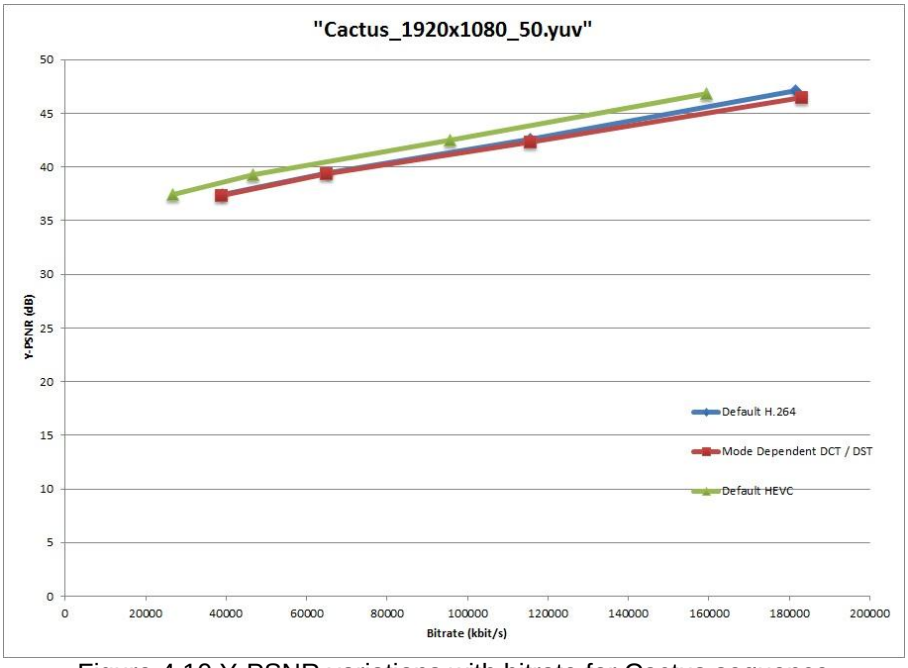


Figure 4.10 Y-PSNR variations with bitrate for Cactus sequence

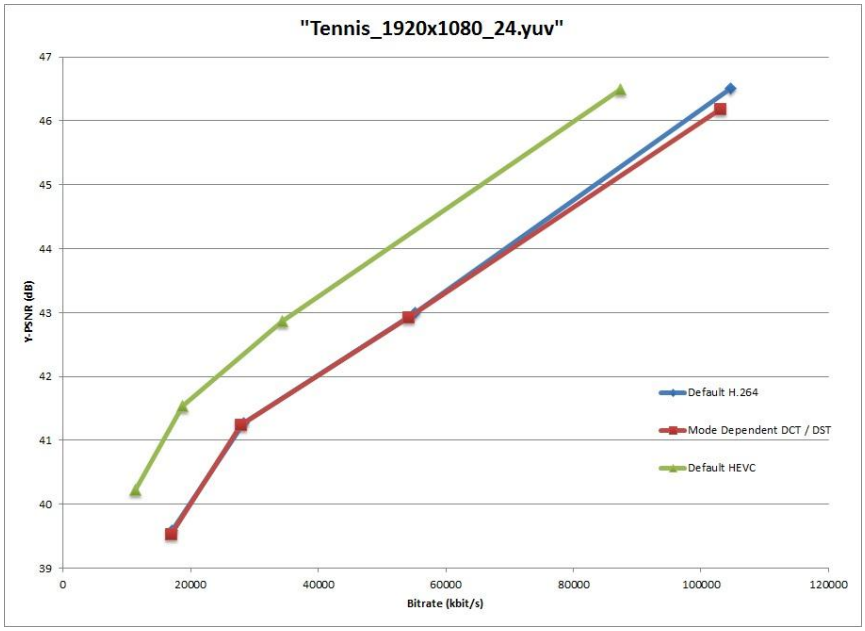


Figure 4.11 Y-PSNR variations with bitrate for Tennis sequence

4.4.4 Results for HD (1080x720) sequences

Table 4.14 Comparison of bitrates and PSNRs for three 1080x720 sequences (H.264/AVC with DCT/DST)

Sequence Name	QP	Default H.264/AVC				H.264/AVC with DCT/DST			
		Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)	Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)
Vidyo1	16	35621.04	47.790	48.745	49.461	35020.80	47.615	48.745	49.461
	20	21832.08	44.884	47.697	48.595	21250.08	44.779	47.697	48.595
	24	13267.68	42.711	46.404	47.183	12959.04	42.640	46.404	47.183
	28	8747.28	40.599	44.965	45.540	8509.44	40.501	44.965	45.540
Vidyo3	16	39600.00	47.536	49.564	49.662	38542.32	47.289	49.564	49.662
	20	24599.28	44.252	48.889	48.689	23705.28	44.156	48.889	48.689
	24	14207.52	42.023	47.630	47.027	13705.20	42.004	47.630	47.027
	28	8987.52	40.048	46.406	45.260	8718.24	39.969	46.406	45.260
Vidyo4	16	31649.28	48.215	49.821	50.004	30944.16	48.022	49.821	50.004
	20	19866.00	45.402	49.184	49.379	19135.20	45.270	49.184	49.379
	24	12456.24	43.057	47.783	47.964	12089.28	42.980	47.783	47.964
	28	8162.88	40.829	46.105	46.256	7959.36	40.657	46.105	46.256

Table 4.15 BD-PSNR and BD-Bitrate for three 1080x720 sequences (H.264/AVC with DCT/DST)

Sequence Name	BD-PSNR (dB)	BD-Bitrate (%)
Vidyo1	0.0216	-0.5157
Vidyo3	0.0851	-1.8388
Vidyo4	0.0422	-0.8327

Table 4.16 Comparison of bitrates and PSNRs for three 1080x720 sequences (H.264/AVC with HEVC)

Sequence Name	QP	Default H.264/AVC				H.264/AVC with HEVC			
		Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)	Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)
Vidyo1	16	35621.04	47.790	48.745	49.461	26735.76	47.5799	49.2065	50.0380
	20	21832.08	44.884	47.697	48.595	14940.96	44.8938	47.8111	48.8625
	24	13267.68	42.711	46.404	47.183	8738.88	42.9241	46.5683	47.4347
	28	8747.28	40.599	44.965	45.540	5621.52	41.1359	45.1231	45.7768
Vidyo3	16	39600.00	47.536	49.564	49.662	30262.08	47.3135	50.3305	50.3281
	20	24599.28	44.252	48.889	48.689	16935.36	44.2926	49.0662	48.9256
	24	14207.52	42.023	47.630	47.027	9007.44	42.1087	47.6992	47.2448
	28	8987.52	40.048	46.406	45.260	5832.96	40.5547	46.4103	45.2869
Vidyo4	16	31649.28	48.215	49.821	50.004	23430.24	47.8886	50.4548	50.6957
	20	19866.00	45.402	49.184	49.379	13652.64	45.3897	49.3501	49.5508
	24	12456.24	43.057	47.783	47.964	8428.08	43.3731	47.8472	48.1619
	28	8162.88	40.829	46.105	46.256	5282.88	41.3587	46.1969	46.3486

Table 4.17 BD-PSNR and BD-Bitrate for three 1080x720 sequences (H.264/AVC with HEVC)

Sequence Name	BD-PSNR (dB)	BD-Bitrate (%)
Vidyo1	2.0530	-33.5926
Vidyo3	1.9721	-33.2696
Vidyo4	2.0659	-33.2327

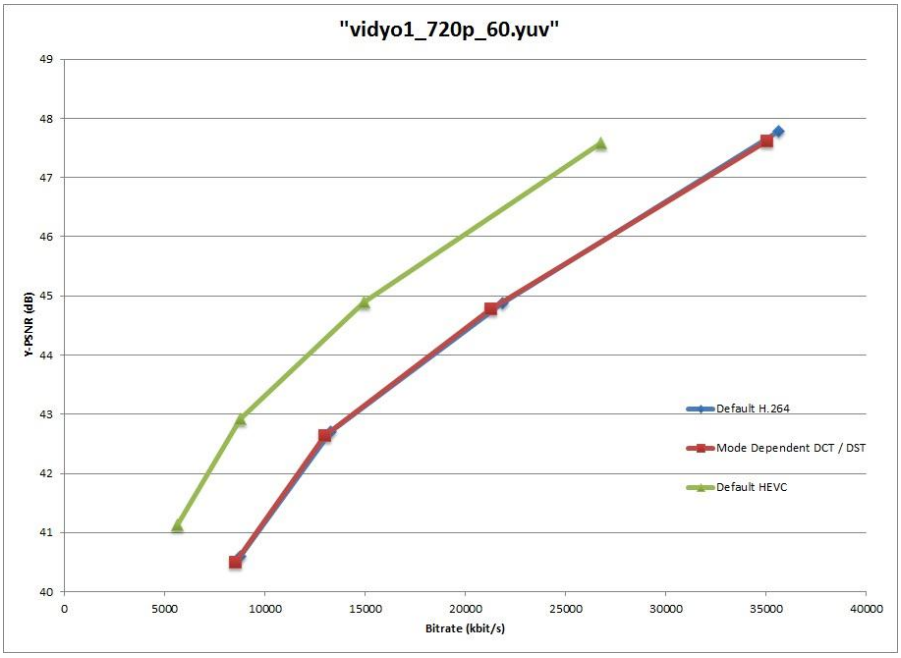


Figure 4.12 Y-PSNR variations with bitrate for Vidyo1 sequence

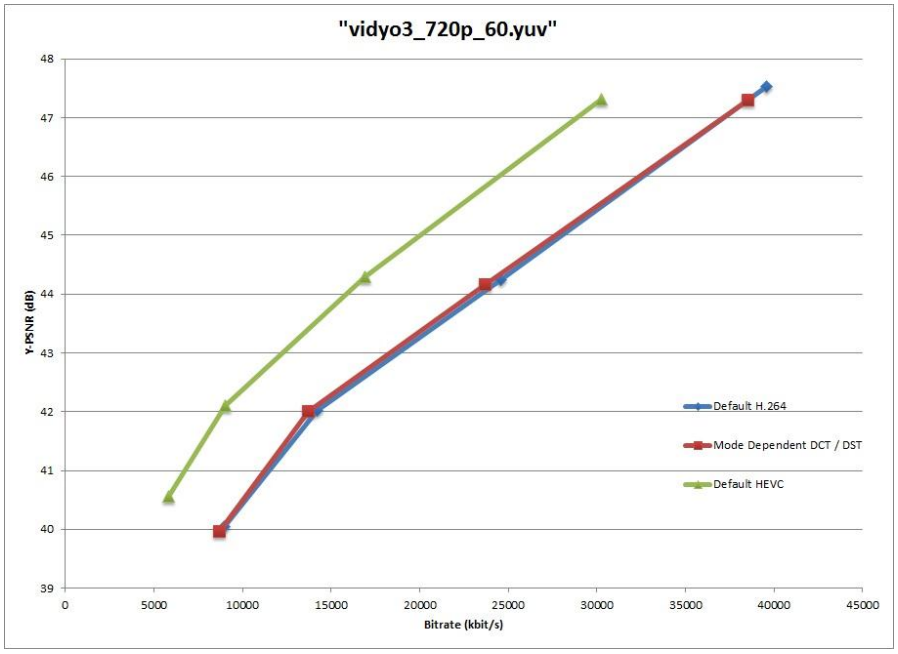


Figure 4.13 Y-PSNR variations with bitrate for Vidyo3 sequence

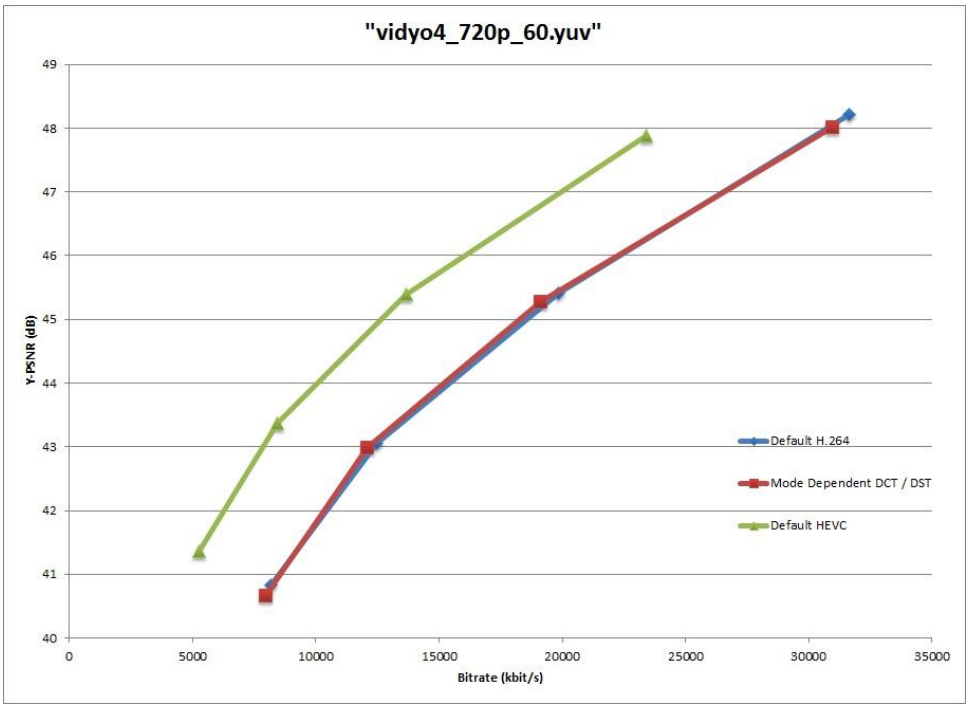


Figure 4.14 Y-PSNR variations with bitrate for Vidyo4 sequence

4.4.4 Results for different combinations of DCT/DST applied to RaceHorses sequence

To observe the performance of the proposed scheme , the DCT/DST combination was applied for 9 cases, starting with horizontal mode only and vertical mode only and then extending to category 1, category 2, combination of category 1 and 2 , and further extending to combination of all categories as described in the proposed scheme and lastly, DST for all modes.

The results are tabulated in Table 4.18 and Table 4.19. The corresponding RD curves are shown in Figure 4.15.

Table 4.18 Comparison of bitrates and PSNRs for DCT/DST combination applied to different intra prediction modes (H.264/AVC Default with DCT/DST for different modes

DST for intra prediction modes	QP	Default H.264/AVC			H.264/AVC with DCT/DST		
		Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	Bitrate (kbit/s)	Y-PSNR (dB)	U-PSNR (dB)	V-PSNR (dB)
DST for mode 1 only	16	10105.68	47.744	47.201	47.377	10100.40	47.748	47.201	47.377
	20	7556.16	44.084	44.030	44.497	7527.36	44.052	44.030	44.497
	24	5429.76	40.496	41.313	41.726	5435.52	40.524	41.313	41.726
	28	3792.24	37.097	38.894	39.324	3804.48	37.173	38.894	39.324
DST for modes 1 and 8 only	16	10105.68	47.744	47.201	47.377	10151.52	47.710	47.201	47.377
	20	7556.16	44.084	44.030	44.497	7559.52	44.021	44.030	44.497
	24	5429.76	40.496	41.313	41.726	5439.60	40.510	41.313	41.726
	28	3792.24	37.097	38.894	39.324	3788.88	37.143	38.894	39.324
DST for mode 0 only	16	10105.68	47.744	47.201	47.377	10119.84	47.746	47.201	47.377
	20	7556.16	44.084	44.030	44.497	7539.84	44.098	44.030	44.497
	24	5429.76	40.496	41.313	41.726	5423.52	40.497	41.313	41.726
	28	3792.24	37.097	38.894	39.324	3787.68	37.147	38.894	39.324
DST for modes 0, 3 and 7 only	16	10105.68	47.744	47.201	47.377	10193.28	47.664	47.201	47.377
	20	7556.16	44.084	44.030	44.497	7583.28	44.497	44.030	44.497
	24	5429.76	40.496	41.313	41.726	5422.80	40.413	41.313	41.726
	28	3792.24	37.097	38.894	39.324	3801.60	37.118	38.894	39.324

Table 4.18—*Continued*

DST for modes 0,1, 3 and 7 only	16 20 24 28	10105.68 47.744 47.201 47.377 7556.16 44.084 44.030 44.497 5429.76 40.496 41.313 41.726 3792.24 37.097 38.894 39.324	10175.52 47.642 47.201 47.377 7582.32 43.990 44.030 44.497 5423.28 40.438 41.313 41.726 3798.00 37.124 38.894 39.324
DST for modes 0, 1, 3, 7 and 8 only	16 20 24 28	10105.68 47.744 47.201 47.377 7556.16 44.084 44.030 44.497 5429.76 40.496 41.313 41.726 3792.24 37.097 38.894 39.324	10238.64 47.571 47.201 47.377 7602.00 43.942 44.030 44.497 5453.76 40.420 41.313 41.726 3795.84 37.121 38.894 39.324
DST for modes 4, 5 and 6 only	16 20 24 28	10105.68 47.744 47.201 47.377 7556.16 44.084 44.030 44.497 5429.76 40.496 41.313 41.726 3792.24 37.097 38.894 39.324	10291.68 47.663 47.201 47.377 7666.80 43.949 44.030 44.497 5508.48 40.393 41.313 41.726 3822.96 37.051 38.894 39.324
Mode dependent DST	16 20 24 28	10105.68 47.744 47.201 47.377 7556.16 44.084 44.030 44.497 5429.76 40.496 41.313 41.726 3792.24 37.097 38.894 39.324	10306.32 47.155 47.201 47.377 7662.72 43.617 44.030 44.497 5476.08 40.198 41.313 41.726 3801.12 36.936 38.894 39.324
DST for all modes	16 20 24 28	10105.68 47.744 47.201 47.377 7556.16 44.084 44.030 44.497 5429.76 40.496 41.313 41.726 3792.24 37.097 38.894 39.324	18232.32 44.351 49.698 49.648 9331.68 39.727 44.471 44.959 5510.88 37.405 41.330 41.779 3755.52 35.510 38.894 39.324

Table 4.19 BD-PSNR and BD-Bitrate (H.264/AVC Default with DCT/DST applied to different intra prediction modes

DST for intra prediction modes	BD-PSNR (dB)	BD-Bitrate (%)
DST for mode 1 only	0.0192	-0.1817
DST for modes 1 and 8 only	-0.0277	0.2627
DST for mode 0 only	0.0239	-0.2468
DST for modes 0, 3 and 7 only	0.0678	-0.7543
DST for modes 0,1, 3 and 7 only	-0.0844	0.7942
DST for modes 0, 1, 3, 7 and 8 only	-0.1595	1.4838
DST for modes 4, 5 and 6 only	-0.0236	0.1937
Mode dependent DST	-0.4913	4.775
DST for all modes	-6.3851	82.9101

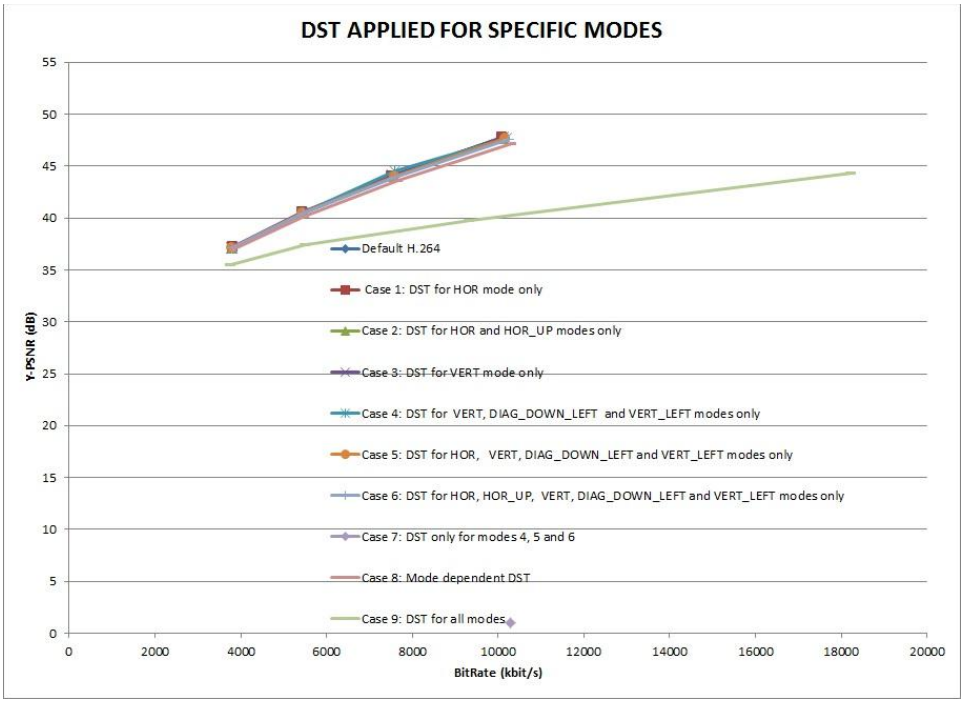


Figure 4.15 Y-PSNR variations with bitrate for DCT/DST applied to different intra prediction modes for RaceHorses sequence

4.5 Conclusions and Futurework

A negative BD-Bitrate or a positive BD-PSNR denotes better coding efficiency. HEVC shows almost 1.5 - 2dB performance improvement than the default H.264. Four out of twelve input video sequences that were analyzed showed some performance improvement for mode dependent DCT/DST in H.264. All the 3 HD sequences (1080x720) showed a slight performance improvement with less than 0.05dB PSNR gain and less than 1% bitrate savings. When mode dependent DCT/DST is applied to different categories of intra prediction modes, it is observed that there is a slight improvement seen only for horizontal and vertical modes.

The performance drop in most cases could be due to many reasons:

- Use of non-integer transform coefficients which results in the decrease in accuracy of reconstructed output
- The number of prediction modes is just 9 in case of H.264 and 33 in case of HEVC. The DCT/DST combinations are used for less number of prediction directions in the former case
- Selection of the best prediction mode and the corresponding mode dependent DCT/DST is not considered here

In order to achieve considerable performance improvements, many other factors may be considered.

- The scan order of the transformed coefficients can be modified to obtain better performance depending upon the intra prediction direction for each block, instead of using the conventional zig-zag scan in H.264.

- Rate distortion optimization (RDO) could be used to apply mode dependent DCT/DST only to the best performing modes. Higher resolution video sequences can be used for analysis by using High profiles in H.264.
- Analysis can also be extended to block sizes other than 4x4 luma and also for chroma intra prediction residuals.

Appendix

Selected Frames From Video Sequences

Figure A.1 First frame from RaceHorses (416x240)


Figure A.2 First frame from BlowingBubbles (416x240)

Figure A.3 First frame from BQSquare (416x240)

Figure A.4 First frame from BQMall (832x480)

Figure A.5 First frame from Keiba (832x480)

Figure A.6 First frame from PartyScene (832x480)

Figure A.7 First frame from BQTerrace (1920x1080)

Figure A.8 First frame from Cactus (1920x1080)

Figure A.9 First frame from Tennis (1920x1080)

Figure A.10 First frame from Vidyo1 (1080x720)

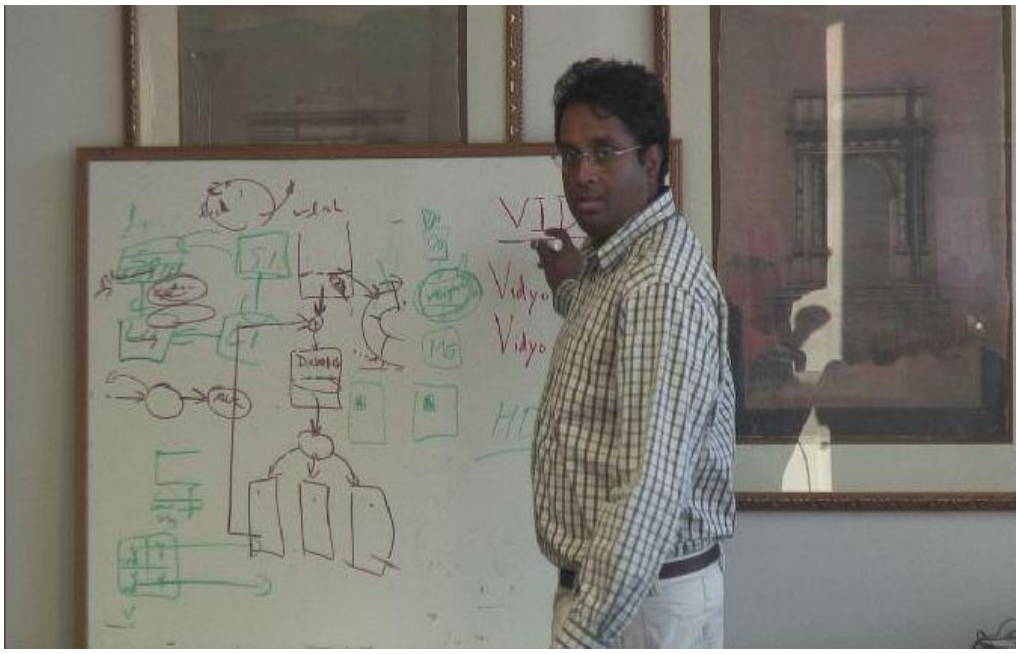


Figure A.11 First frame from Vidyo3 (1080x720)


Figure A.12 First frame from Vidyo4 (1080x720)

References

[1] I.E.Richardson, “The H.264 advanced video compression standard”, Wiley, Second edition, 2010

[2] Advanced video coding for generic audiovisual services, ITU-T Rec. H.264 / ISO / IEC 14496-10, Jan. 2012

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

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

[5] H.Kalva, “The H.264 video coding standard,” IEEE Multimedia, vol. 13, no. 4, pp.86-90, Oct. 2006

[6] G.J.Sullivan et al, “The H.264/AVC advanced video coding standard: overview and introduction to the fidelity range extensions,” SPIE conference on Applications of Digital Image Processing XXVII, vol. 5558, pp. 53-74, Nov. 2004

[7] T. Wiegand et al, “H.264/MPEG4-AVC fidelity range extensions: tools, profiles, performance, and application areas,” IEEE ICIP 2005, vol. 1, pp. 593-596, Sep. 2005

[8] H. Schwarz et al, “Overview of the scalable video coding extension of the H.264/AVC standard,” IEEE Trans. on Circuits and Systems for Video Technology, vol. 17, no. 9, pp. 1103-1120, Sep. 2007

[9] A. Vetro et al, “Overview of the stereo and multiview video coding extensions of the H.264/MPEG-4 AVC standard,” Proceedings of the IEEE, vol. 99, no. 4, pp. 626-642, Apr. 2011

[10] Whitepaper on MVC , Website: http://research.nokia.com/files/3D_Video.pdf

[11] D. Marpe et al, “The H.264/MPEG-4 AVC standard and its applications”, IEEE Communications Magazine, vol. 44, pp. 134-143, Aug. 2006

[12] S. Kwon et al, “Overview of H.264/MPEG-4 part 10”, Journal of Visual Communication and Image Representation, vol. 17, no. 2, pp. 186-216, April 2006

[13] G. J. Sullivan et al, “Overview of high efficiency video coding (HEVC) standard”, Pre-publication draft, to appear in IEEE Trans. Circuits and Systems for Video Technology, Dec. 2012

[14] J. –R. Ohm et al, “Comparison of the coding efficiency of video coding standards – including high efficiency video coding (HEVC)”, Pre-publication draft, to appear in IEEE Trans. Circuits and Systems for Video Technology, Dec. 2012

[15] F. Bossen et al, “HEVC complexity and implementation analysis”, Pre-publication draft, to appear in IEEE Trans. Circuits and Systems for Video Technology, Dec. 2012

[16] C. Fogg, “Suggested figures for the HEVC specification”, ITU-T / ISO-IEC Document: JCTVC-J0292r1, July 2012

[17] H. S. Malvar et al, “Low-complexity transform and quantization in H.264 and/AVC”, IEEE Trans. Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 598-603, July 2003

[18] R. Joshi and R. Cohen, "AHG report - alternative transforms", ITU-T / ISO-IEC Document: JCTVC-B005_r1, July 2010

[19] Y.Ye and M. Karczewicz, "Improved H.264 intra coding based on bi-directional intra prediction, directional transform, and adaptive coefficient scanning", IEEE ICIP 2008, pp. 2116-2119, 2008

[20] Open source article on MDDT in HEVC: http://www.h265.net/2009/09/mode-dependent-directional-transform-mddt-in-jmkta.html

[21] M. Budagavi et al, "Description of video coding technology proposal by Texas Instruments Inc.", ITU-T / ISO-IEC Document: JCTVC-A101, April 2010

[22] A. Ichigaya et al, "Description of video coding technology proposal by NHK and Mitsubishi", ITU-T / ISO-IEC Document: JCTVC-A122, April 2010

[23] H. Yang et al, "Description of video coding technology proposal by Huawei Technologies & Hisilcon Technologies", ITU-T / ISO-IEC Document: JCTVC-A111, April 2010

[24] K. McCann et al, "Samsung's response to the call for proposals on video compression technology", ITU-T / ISO-IEC Document: JCTVC-A124_r2, April 2010

[25] T. Suzuki and A. Tabatabai, "Description of video coding technology proposal by Sony", ITU-T / ISO-IEC Document: JCTVC-A103, April 2010

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

[27] C. Auyeung and A. Tabatabai, "Intra coding with directional DCT and directional DWT", ITU-T / ISO-IEC Document: JCTVC-B107_r1, July 2010

[28] E. Alshina et al, "Rotational transform for image and video compression", IEEE ICIP

[29] J. Han et al, "Towards jointly optimal spatial prediction and adaptive transform in video/image coding", International Conference on Acoustics, Speech and Signal Processing, pp. 726-729, Mar. 2010

[30] A. Saxena and F. Fernanades, "CE7: Mode-dependent DCT/DST for intra prediction in video coding", ITU-T / ISO-IEC Document: JCTVC-D033, Jan. 2011

[31] A. Saxena and F. Fernandes, "Mode dependent DCT/DST for intra prediction in block-based image/video coding", IEEE ICIP, pp. 1685-1688, Sep. 2011

[32] R. Chivukula and Y. Reznik, “Fast computing of discrete cosine and sine transforms of types VI and VII”, Proc. SPIE, Applications of Digital Image Processing XXXIV, vol. 8135, Sep. 2011

[33] H. Malvar et al, “Low-complexity transform and quantization in H.264/AVC”, IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 598-603, July 2003

[34] H.264 Reference Software JM 18.4: http://iphome.hhi.de/suehring/tml/download/

[35] HEVC Reference Software HM 8.0: http://hevc.kw.bbc.co.uk/trac/browser/tags/HM-8.0

[36] Test video sequences: ftp://ftp.tnt.uni-hannover.de/testsequences/

[37] G. Bjontegaard, "Calculation of average PSNR differences between RD curves", VCEG-M33, April 2001

[38] MATLAB source code for BD PSNR and BD Rate: http://www.mathworks.com/matlabcentral/fileexchange/27798-bjontegaard-metric/content/bjontegaard.m

[39] Graphics Display Resolution: http://en.wikipedia.org/wiki/Graphics_display_resolution

[40] Analysis of coding tools in HEVC Test model (HM 1.0) - Intra Prediction: http://www.h265.net/2010/12/analysis-of-coding-tools-in-hevc-test-model-hm-intra-prediction.html

Biographical Information

Priyadarshini Anjanappa completed her Bachelor of Engineering in Electronics and Communications from Visvesvaraya Technological University, Karnataka, India in 2009. She then joined the University of Texas at Arlington as a graduate student of Electrical Engineering department in Fall 2010. During the course of her graduate studies she worked under the guidance of Dr. K. R. Rao in the Multimedia Processing Lab. She interned at Research In Motion at Sunrise, Florida during Fall 2011 and Spring 2012.