REDUCING ENCODER COMPLEXITY OF INTRA-MODE DECISION USING CU EARLY TERMINATION ALGORITHM

by

NISHIT SAMIRBHAI SHAH

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 2015

Acknowledgements

Foremost, I would like to express my sincere gratitude to Dr. K. R. Rao from the bottom of my heart for being a guide, mentor and a constant source of encouragement throughout my thesis. He has been and will be a great source of inspiration for his zeal, resourcefulness and ideas. I am very grateful for the opportunity to work under his guidance.

I would also like to thank my MPL lab-mates: Harshdeep Jain, Kushal Shah, Pratik Mehta and Srikanth Vasireddy for providing valuable inputs throughout my research. Also I thank Shiba Kuanar for his support and inputs.

Finally and most importantly, I would like to acknowledge my parents, Bharati Shah and Samir Shah and my adorable sister Riddhi for their unwavering love and support throughout my entire life.

September 14, 2015

Abstract

REDUCING ENCODER COMPLEXITY OF INTRA-MODE DECISION USING CU EARLY TERMINATION ALGORITHM

Nishit Samirbhai Shah, M.S.

The University of Texas at Arlington, 2015

Supervising Professor: K.R. Rao

In this thesis an intra prediction algorithm is proposed that terminates complete full search prediction for the CU and is replaced by CU early termination algorithm which determines the complexity of the CU block and then decision is made to further split or non-split the CU. When the CU texture is complex the CU is split into smaller sub units to find the best size and when the CU texture is flat, the CU is not divided further into sub – units. Down sampling is done first after which complexity is calculated by which a threshold value is set. This threshold value dictates early termination of CU block. This is followed by a TU mode decision to find the optimal prediction mode from the 35 prediction modes. Proposed method will use tree split/tree merge algorithm. Experimental results based on several video test sequences suggest a decrease of about 12%-24% in encoding time is achieved with implementation of the proposed CU early termination algorithm and fast intra mode decision algorithm for intra predication with negligible degradation in peak signal to noise ratio (PSNR). Metrics such as BD-bitrate (Bjøntegaard Delta bitrate), BD-PSNR (Bjøntegaard Delta Peak Signal to Noise Ratio), RD graph (Rate Distortion) and computational complexity are also used.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

1.1 Significance

Innovations in the communication systems have been extraordinary in the last decade. Technology in communication systems has transformed tremendously from having only analog television via cable, satellite with availability of only a few channels or mobile phones that can only make voice calls or internet connections that are slow, mostly connected through a dial up modem connected via telephone lines.

Video traffic is dominating both the wireless and wireline networks. Globally, IP video is expected to be 79% of all IP traffic in 2018, up from 66% in 2013. On wireless networks, video is 70% of global mobile data traffic in 2013(Cisco VNI forecast). Movie studios, broadcasters, streaming video providers, TV and consumer electronics device manufacturers are working towards providing immersive "real life" "being there" video experience to consumers by using features such as increased resolution (Ultra HD 4K/8K), higher frame rate, higher dynamic range (HDR), wider color gamut (WCG), and 360 degrees video.

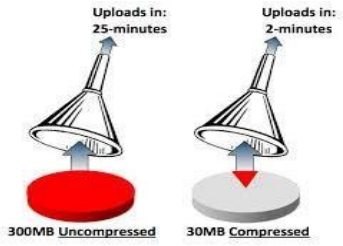

Figure 1-1 Basics of video compression [19]

Compression is the process of removing redundant information and representing data with fewer bits than the original information would use. It is useful because it helps to reduce the consumption of expensive resources such as data storage on hard disks and wider transmission bandwidths [2].

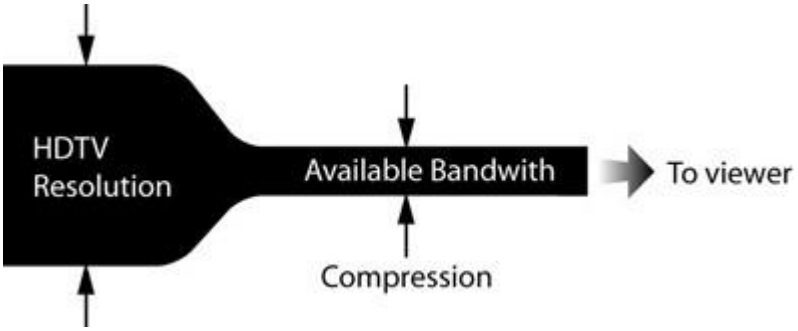

Figure 1-2 Bandwidth requirements [19]

Hence, research is still going on in the field of compression techniques to enable real-time data transmission using fewer resources. Compression techniques are categorized as lossless or lossy. Lossless compression is possible because most of the real-world data has statistical redundancy. If the data has been losslessly compressed, the original data can be recovered with no loss. Lossless compression exploits statistical redundancy and represents data with more fidelity and less error [2]. It is beneficial in areas like text compression and audio compression. Lossy compression involves some information loss, so the data cannot be recovered exactly. It is applied in areas where data distortion is tolerable like video compression, image compression and some types of audio compression. Lossy image compression is used in digital cameras, to increase the storage capacity with less degradation of picture quality. Similarly lossy video compression is used on DVDs, Blu-ray disks [5], Internet telephony using MPEG-2 [2], H.264 [2] and HEVC (High Efficiency Video Coding) [1].

Multimedia consumer applications have a very large market. The revenues involved in digital TV broadcasting and DVD, Blu-ray distributions are substantial. Thus standardization of video coding is essential. Standards simplify inter-operability between encoders and decoders from different manufacturers; they make it possible for different vendors to build platforms that incorporate video codecs, audio codecs, security and rights management and they all interact in well-defined and consistent ways. There are numerous video compression standards, both open source and proprietary, depending on the applications and end-usage. Figure 1-3 shows the evolution of the video codec standards from the 90's till today.

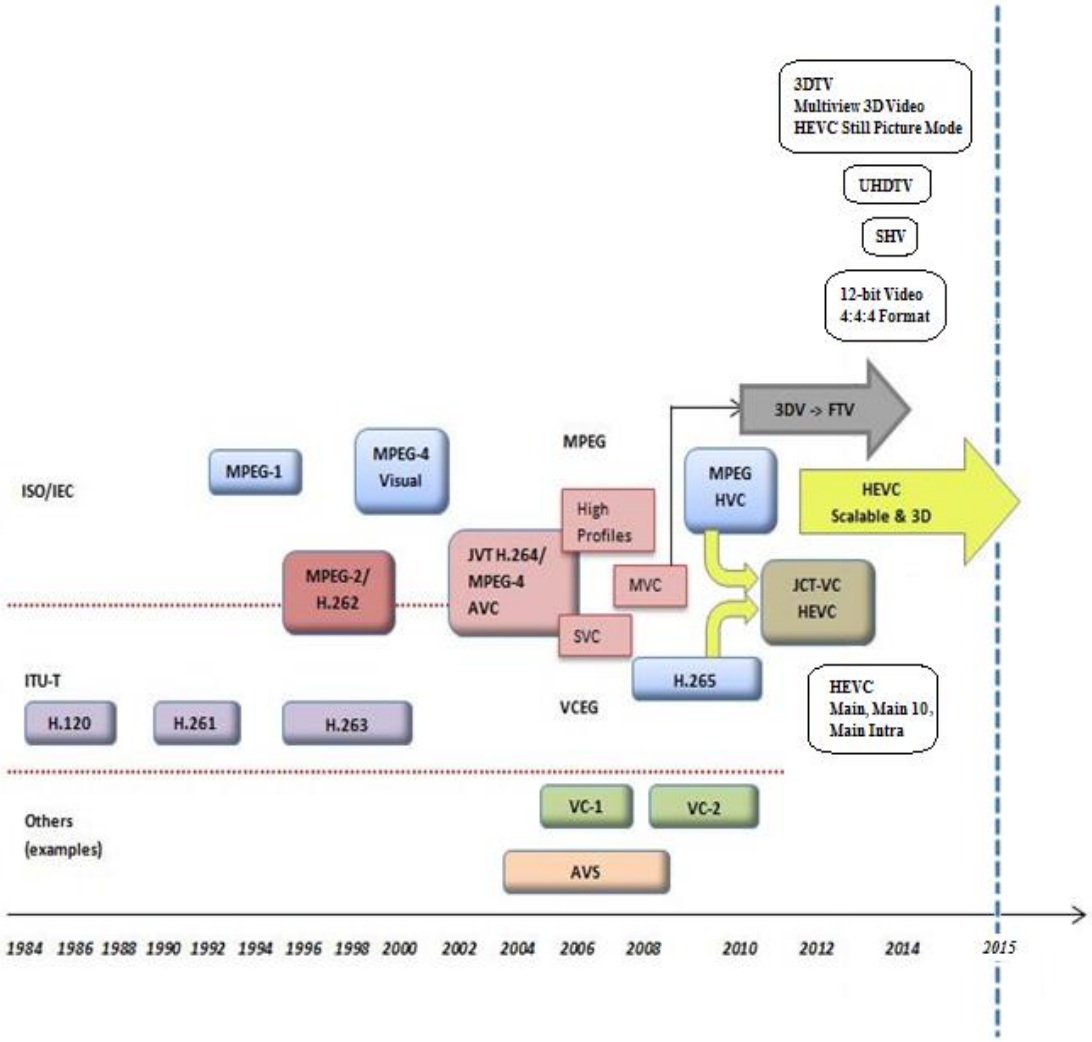

Figure. 1-3 Evolution of video coding standards[21]

The High Efficiency Video Coding (HEVC) standard is the most recent joint video project of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) standardization organizations, working together in a partnership known as the Joint Collaborative Team on Video Coding (JCT-VC) [1]. However, an increasing diversity of services, the growing popularity of HD video, and the emergence of beyond- HD formats (e.g., 4k×2k or 8k×4k resolution) [10] are creating even stronger needs for coding efficiency superior to H.264/MPEG-4 AVC's capabilities. The need is even stronger when higher resolution is accompanied by stereo or multiview capture and display. Moreover, the traffic caused by video applications targeting mobile devices and tablets PCs, as well as

the transmission needs for video-on-demand services, are imposing severe challenges on today's networks. An increased desire for higher quality and resolutions is also arising in mobile applications [1].

1.2 Why is complexity reduction important in HEVC/H.265?

HEVC/H.265 has very efficient compression methods, which allow it to compress video much more efficiently than older standards and provide more flexibility for application to a wide variety of network environments. To achieve highly efficient compression, the computational cost associated with it is also very high. This is the reason why, these increased compression efficiencies cannot be exploited across all application domains. Resource constrained devices such as cell phones and other embedded systems use simple encoders or simpler profiles of the codec to tradeoff compression efficiency and quality for reduced complexity [3]. Video coding standards specify the decoding process and bit stream syntax of the compressed video. The encoding process or the process of producing a standard compliant video is not specified. This approach leaves room for innovation in the encoding algorithm development. The work in this thesis focuses on coding unit early termination algorithm and tree split/merge algorithm for Transform Unit (TU) to decrease the encoder complexity for the intra prediction modes of HEVC.

1.3 Outline of the research

The research presented here proposes a reduced complexity HEVC encoder by making use of HM 16.0 reference software [38]. A new technique is implemented for reducing encoding complexity in HEVC. The results show reduction in complexity in terms of encoding time for different videos sequences, with acceptable loss in the PSNR and bit-rates.

1.4 Thesis Outline

Chapter 2 provides details of various blocks in HEVC encoder along with brief explanation of encoding process. Chapter 3 discusses present intra-prediction technique along with various encoder complexity reduction algorithms present for coding unit and prediction unit blocks along with proposed implementation method for reducing complexity using coding unit early termination with tree split/merge algorithm for transform unit. Chapter 4 discusses the simulations and the results for different formats of

test sequences. Chapter 5 outlines the conclusions and further research. The configuration files used by the HM 16.0 [38] software of HEVC encoder for the generation of the bitstreams are also provided.

Chapter 2

High Efficiency video coding

2.1 Introduction

HEVC defines a high-level syntax that supports network interfacing and other systems implementation aspects, and a video coding layer that carries the compressed picture data. Many of the high-level syntax features of HEVC have been retained or extended from the H.264/MPEG-4 Advanced Video Coding (AVC) standard [3]. Parameter sets contain information that can be shared for the decoding of several pictures or sequences of pictures in the video bitstream. The parameter set structure provides a robust mechanism for conveying data that are essential to the decoding process by separating out this top-level header information to enable it to be repeated or reliably conveyed “out of band” as appropriate for the application. Each syntax structure is placed into a logical data packet called a network abstraction layer (NAL) unit. Depending on the content of a two-byte NAL unit header, it is possible to readily identify the purpose of the associated payload data, e.g., parameter sets, data for decoding random-accessible pictures, etc. A total of 31 NAL unit types are defined in the first edition (although the number can be increased, as a 6-bit code is used for NAL unit type signaling).

The HEVC extension [21] also includes extended-range formats with increased bit depth and enhanced color component sampling, scalable coding, and 3-D/stereo/multi-view video coding (the latter including the encoding of depth maps for use with advanced 3-D displays) [1]. Three profiles, namely main, main intra and main 10 bit profiles have been finalized as final draft international standard (FDIS) by JCT-VC in January 2013. Apart from that various extensions such as 3D video, scalable video coding (SVC),screen content coding, higher bit depth etc. are under development. While the highest performance gain also comes with associated high complexity requirements, just marginally lowering performing solutions also brings high coding gains [9][4][6]. Coding gains in HEVC are due to both advanced inter and intra predictions.

2.2 HEVC coding design and Feature highlights

The video coding layer of HEVC employs essentially the same block-based "hybrid" approach (inter-/intra-picture pre- diction and 2D transform coding) used in all video compression standards since H.261. Figure. 2-1 depicts the block diagram of a hybrid video encoder that could create a bitstream that conforms to the HEVC standard. A block-wise prediction residual is computed from corresponding regions of previously decoded pictures (inter-picture motion compensated prediction) or neighboring previously decoded samples from the same picture (intra-picture spatial prediction). The residual is then processed by a block transform, and the transform coefficients are quantized and entropy coded. Side information data such as motion vectors and mode switching parameters are also encoded and transmitted. Some key elements that enable the enhanced compression capability of HEVC are discussed below. A more detailed description of the key technical features can be found in [2].

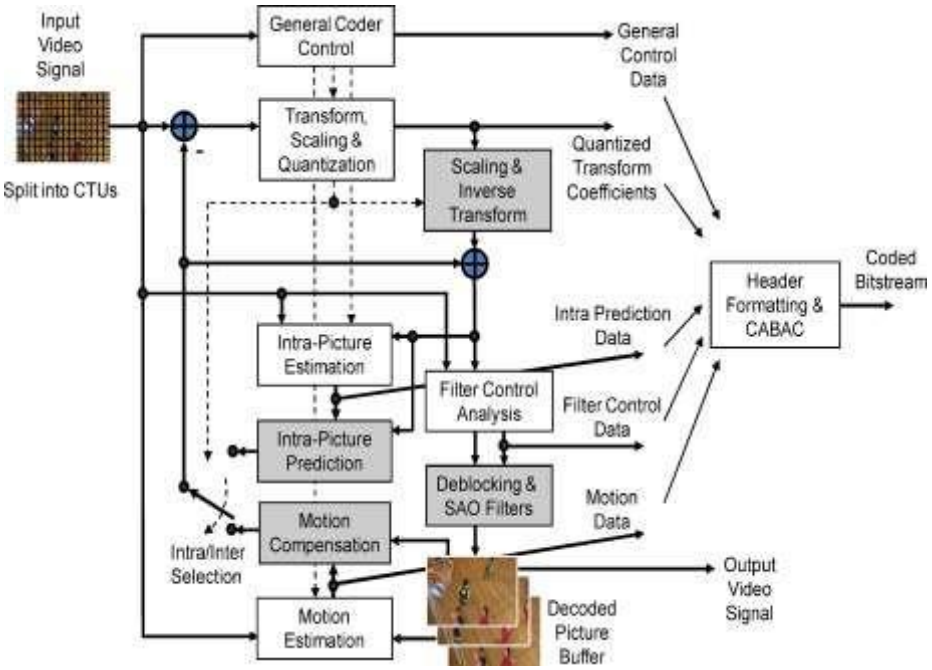

Figure 2-1 HEVC encoder block diagram [1]

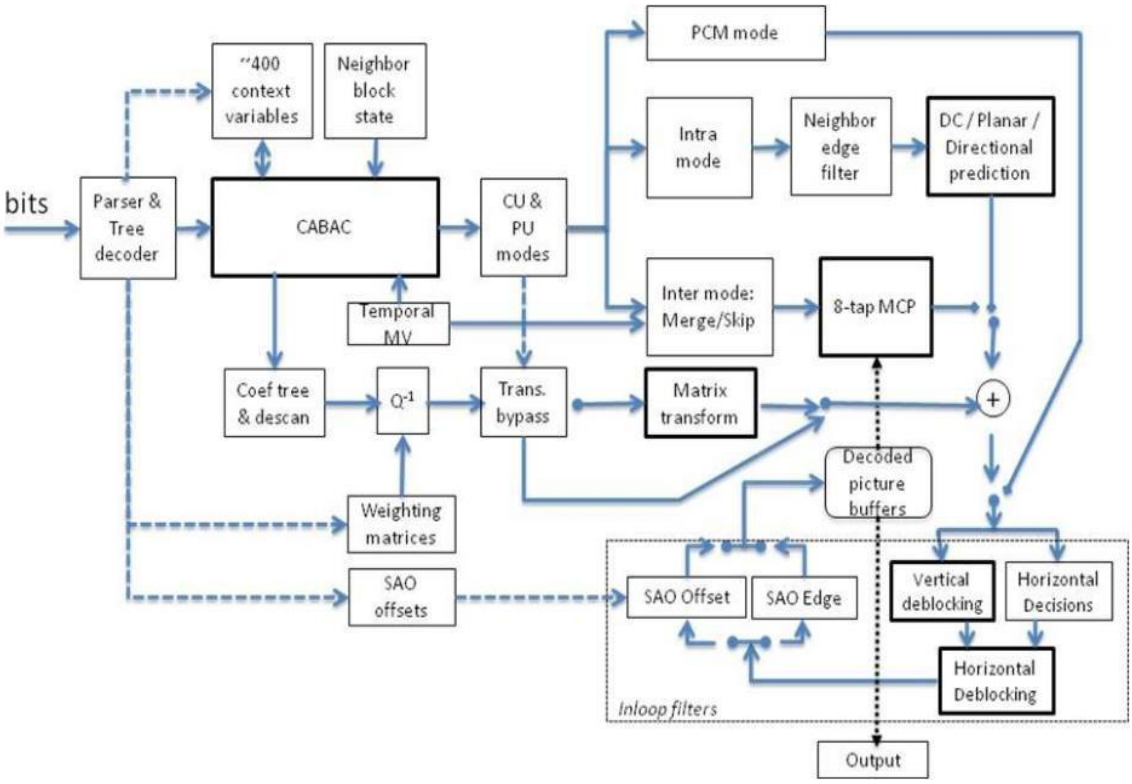

Figure 2-2 HEVC decoder block diagram [10]

2.2.1 Structure of Encoder, Video Format and Sampling

The quad-tree block partitioning is based on a coding tree unit (CTU) structure as shown in figure 2-4 which is analogous to macro block in previous standards. Video is a packet or sequence of frames and in the HEVC standard each coded video frame is partitioned into tiles, slices and CTUs. CTUs are subdivided into square regions called coding units (CU). CUs are predicted using intra or inter prediction where the first frame at each random access point of a video sequence is coded using only intra prediction so that it has no dependence on other pictures. The remaining frames are mostly coded by inter prediction, then residual is transformed using transform units and encoded using CABAC [11] [12] [13]. HEVC uses YCbCr color space with 4:2:0 color format with 8 bps (bits per color sample). Y is symbol for luma component, Cb is symbol for blue chroma component and Cr is symbol for red chroma component [11] as shown in figure 2.3 [15].

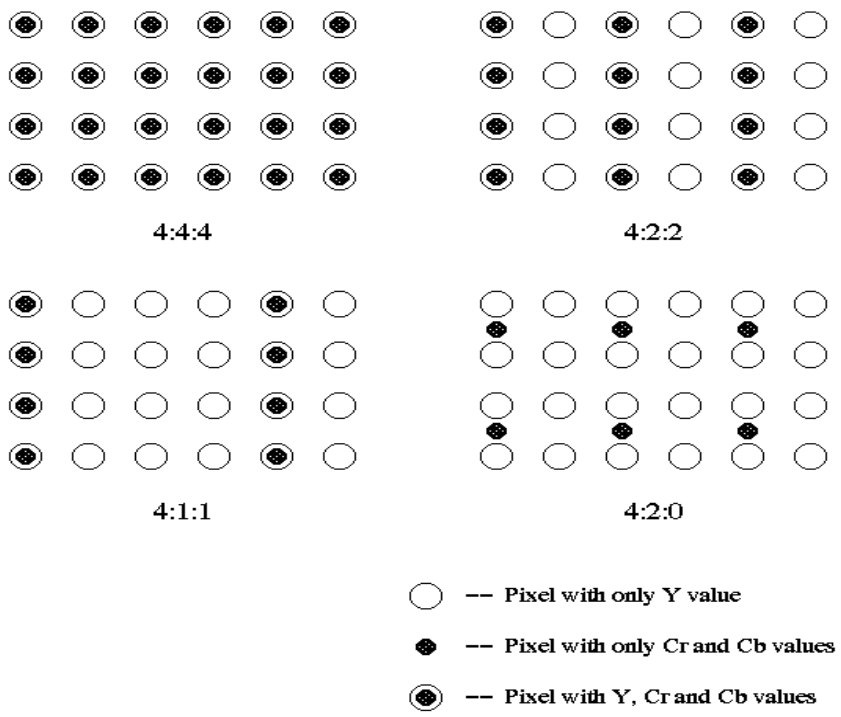

Figure 2-3 formats for YUV components [15]

2.2.2 Coding Tree Units and Coding Tree Block Structure

In contrast to the macroblock of previous standards (consisting of a 16 x 16 block of luma samples and two corresponding blocks of chroma samples), the analogous structure in HEVC is the coding tree unit (CTU). Each picture is split into CTUs of equal size. The CTU consists of a square coding tree block (CTB) for luma and corresponding CTBs for chroma. However, the specific size of a luma CTB can be chosen by the encoder using, 32, or 64, and the larger sizes tend to provide better compression. In version 1, only 4:2:0 color sampling is supported, such that the corresponding chroma structures always have half the luma array size both horizontally and vertically. Each picture is segmented into sequences of CTUs in raster scan order, and each such sequence of CTUs is referred to as a slice. Each slice has a header that enables it to be de- coded independently of all other slices in the picture. The CTBs of each CTU are partitioned into coding blocks (CBs), as indicated by a quadtree structure (Figure. 2). When a luma CTB is split by the quadtree, the luma and chroma CBs are split together, and a luma

CB can be as small as 8 x 8 (accompanied by two 4 x 4 chroma CBs). One luma CB together with the two corresponding chroma CBs and associated syntax elements is referred to as a coding unit (CU).

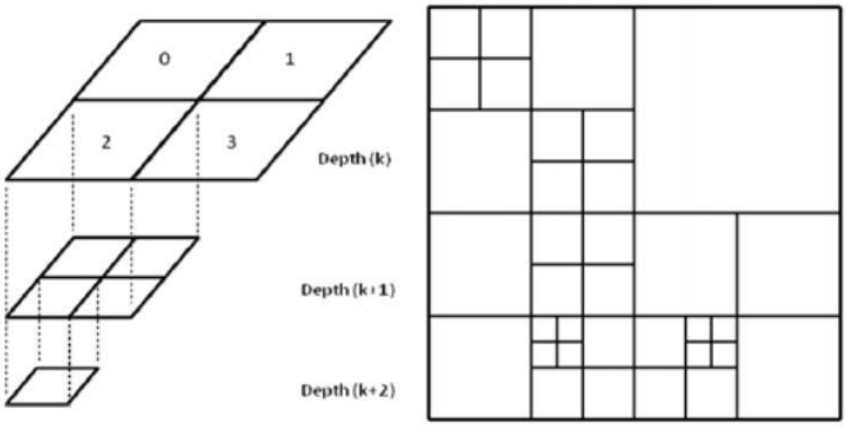

Figure 2-4 Quad tree CU structure in HEVC [16] [1]

Below the CU level, additional partitioning is performed into prediction units (PUs) and transform units (TUs). The decision whether to encode a picture area by inter-picture (motion compensated) or intra-picture (spatially extrapolated) prediction is made at the CU level. CBs have always square shapes. The luma and chroma prediction blocks (PBs) within a PU are also always square in the case of intra-picture prediction; for inter-picture prediction several non-square rectangular block shapes can also be chosen.

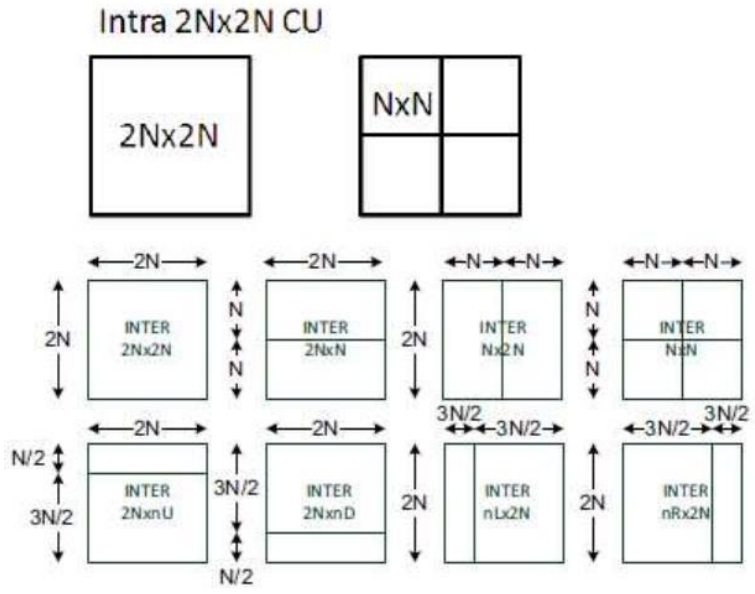

Figure 2-5 Intra and Inter frame prediction modes for HEVC [16]

2.2.3 Transform Units and Transform Blocks

The prediction residual difference signal is coded using block transforms. A transform unit (TU) tree structure has its root at the CU level, where the CBs may be further split into smaller transform blocks (TBs). Integer basis functions approximating the discrete cosine transform (DCT) are defined for dyadic TB sizes from 4 x 4 to 32 x 32. For the 4 x 4 transform of intra-picture prediction residuals, Luma only integer approximation of the discrete sine transform (DST) is used instead. The quantization of transform coefficients is controlled by a quantization parameter (QP) value which maps logarithmically to the quantizer step size (doubling each time the QP value increases by 6). Frequency-dependent quantization step size variation (based on transform coefficient position) is also supported. Coding and decoding of non-zero quantized coefficients is performed by grouping them into 4 x 4 coefficient sub-blocks and scanning the coefficients in each sub-block using a scanning order that is usually diagonal, but becomes horizontal or vertical for small TBs (8 x 8 and smaller) with particular directional modes of intra-picture prediction. The position of the last non-zero coefficient in the scanning order is encoded first, followed by a "significance map" to identify which other preceding coefficients have non-zero values, and then the signs and magnitudes of the significant coefficients are coded.

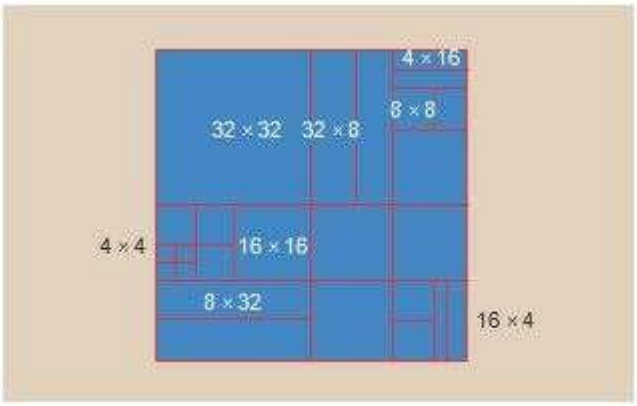

Figure 2-6 Arrangement of TUs in a CU [14]

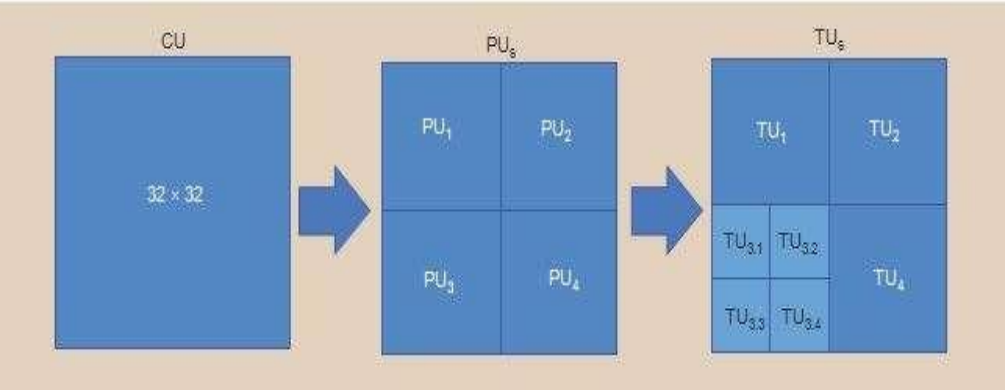

Figure 2-7 Partitioning of 32x32 CU into PUs and TUs [14]

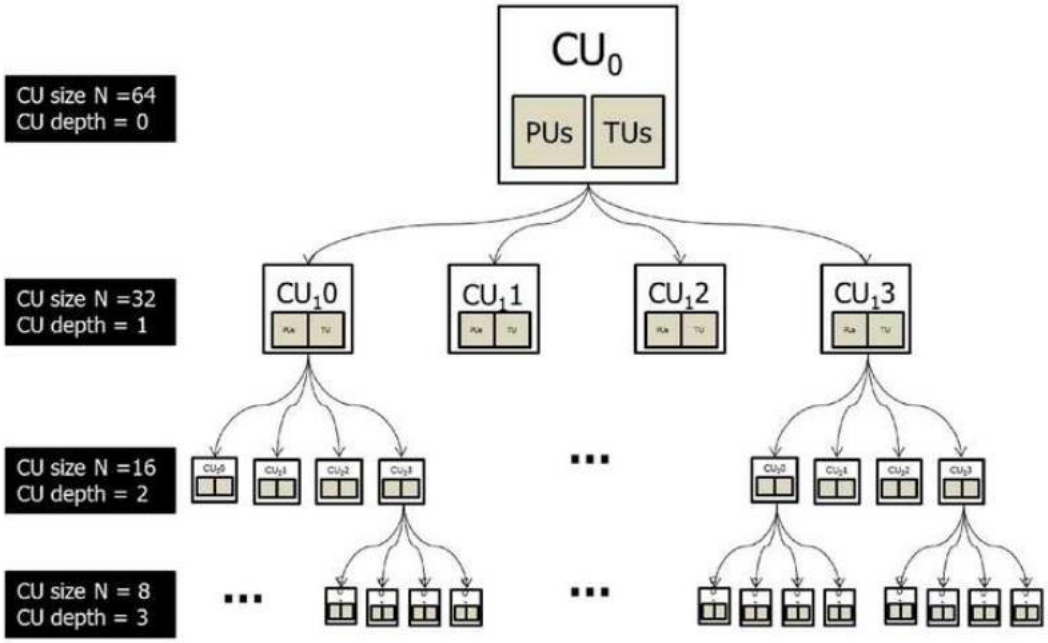

Figure 2-8 Splitting Coding unit into prediction units and transform units [23]

Similarly, starting at the level of a CU, a CB (Coding Block) can have one Transform Block (TB) of the same size as the CB or be split into smaller TBs [1] [21] [22] as shown in figures 2-9, 2-10 and 2-11.

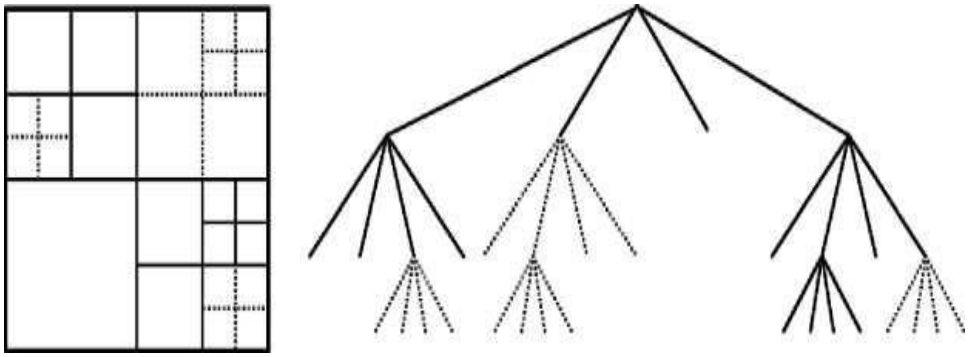

Figure 2-9 CTB with its partitioning and corresponding quad tree [1]

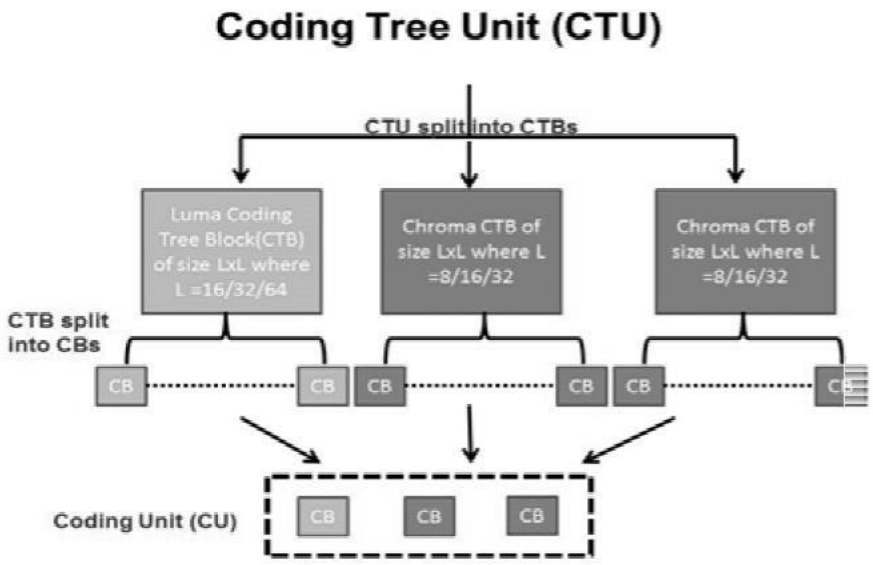

Figure 2-10 Splitting Coding tree unit into Coding Blocks [1]

2.2.4 Slice and Tiles

The HEVC standard introduced tiles as a means to support parallel processing, with more flexibility than the normal slices in the H.264/AVC standard [2] but considerably lower complexity than the Flexible Macroblock Ordering (FMO) standard. Tiles are specified by vertical and horizontal boundaries with intersections that partition a picture into rectangular regions. Figure 2-12 shows an example of tile

partitions that contain slices. The spacing of the row and column boundaries of tiles need not be uniform. This offers greater flexibility and can be useful for error resilience applications. In each tile, LCUs are processed in a raster scan order. Similarly, the tiles themselves are processed in a raster scan order within a picture.

The HEVC standard also supports slices, similar to slices found in the H.264/AVC standard, but without FMO. Slices and tiles may be used together within the same picture. To support parallel processing, each slice in HEVC can be subdivided into smaller slices called entropy slices. Each entropy slice can be independently entropy decoded without reference to other entropy slices. Therefore, each core of a CPU can handle an entropy-decoding process in parallel [14]. Figure 2-12 shows the tile partitions containing slices.

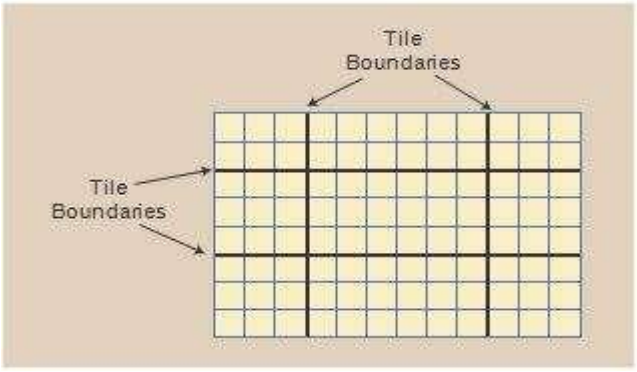

Figure 2-11 A picture partitioned into nine tiles [14]

The slices are processed in the order of a raster scan. A picture may be split into one or several slices as shown in figure 2-13 so that a picture is a collection of one or more slices. Slices are self-contained in the sense that, given the availability of the active sequence and picture parameter sets, their syntax elements can be parsed from the bit stream and the values of the samples in the area of the picture that the slice represents can be correctly decoded without the use of any data from other slices in the same picture.

Tiles are self-contained and independently decodable rectangular regions of the picture. The main purpose of tiles is to enable the use of parallel processing architectures for encoding and decoding. Multiple tiles may share header information by being contained in the same slice. Alternatively, a single

tile may contain multiple slices. A tile consists of a rectangular arranged group of CTUs as shown in figure 2-12.

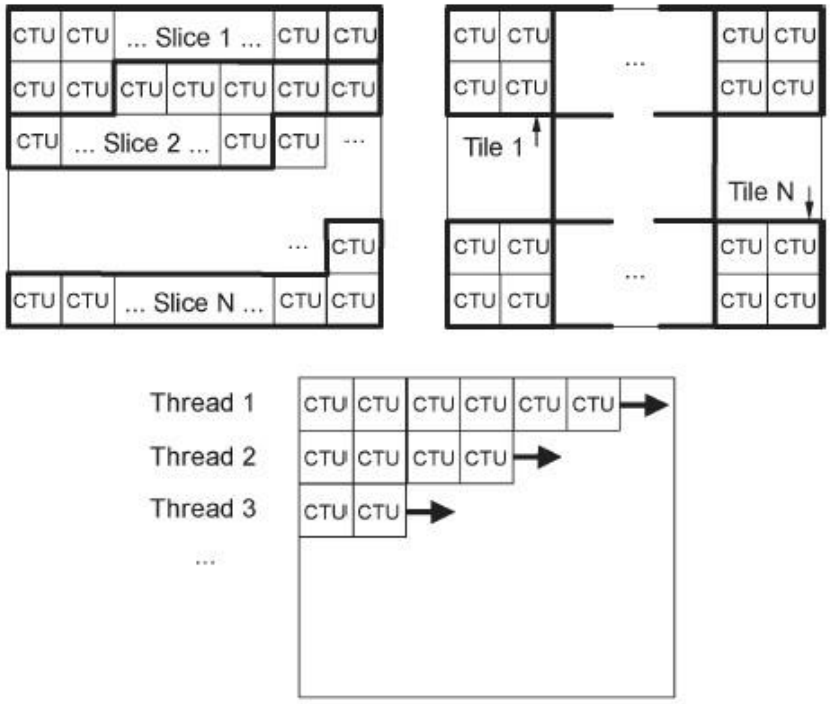

Figure 2-12 Subdivision of picture into slice and Tiles [1]

2.3 HEVC Encoder Description

Major parts of the HEVC encoder are discussed in the following section.

2.3.1 Intra-picture prediction

Intra-picture prediction operates according to the TB size, and previously decoded boundary samples from spatially neighboring TBs are used to form the prediction signal. Directional prediction with 33 different directional orientations is defined for (square) TB sizes from 4×4 up to 32×32. The possible prediction directions are shown in figure 2-9. Alternatively, planar prediction can also be used. The chroma component should be explicitly signed as horizontal, vertical, planar, or DC prediction modes if it is different from luma prediction modes.

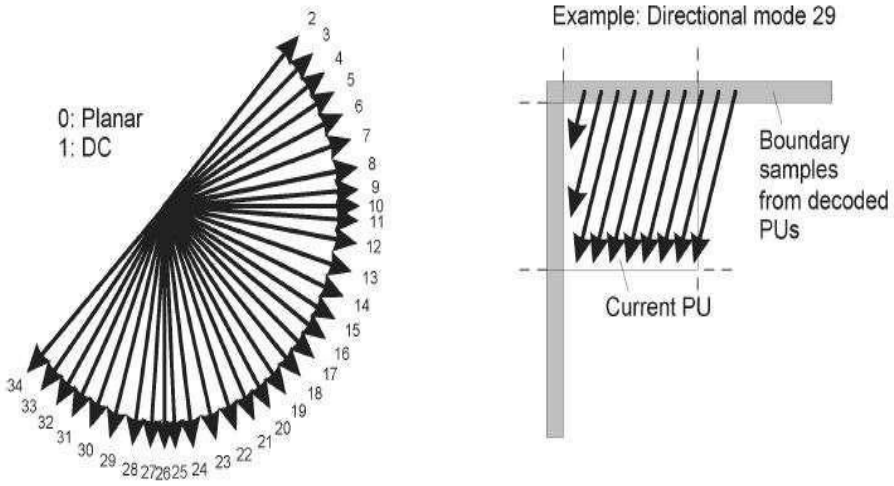

Figure 2-13 Mode decision for intra picture prediction [1]

The HEVC standard also includes a planar intra-prediction mode which is useful for predicting smooth picture regions. In planar mode, the prediction is generated from the average of two linear interpolations.

2.3.2 Inter-picture Prediction

Compared to intra-picture predicted CBs, the HEVC standard supports more PB partition shapes for inter-picture predicted CBs. The partitioning modes of PART_2N×2N, PART_2N×N and PART_N×2N as shown in Figure 2-15 indicate the cases when the CB is not split, split into two equal-size PBs horizontally, and split into two equal-size PBs vertically, respectively. PART−N×N specifies that the CB is split into four equal size PBs, but this mode is only supported when the CB size is equal to the smallest allowed CB size. In addition, there are four partitioning types that support splitting the CB into two PBs having different sizes: PART−2N×nU, PART−2N×nD, PART−nL×2N, and PART−nR×2N (U=up, D=down, L=left and R=right) as shown in figure 2-5. These types are known as asymmetric motion partitions [1].

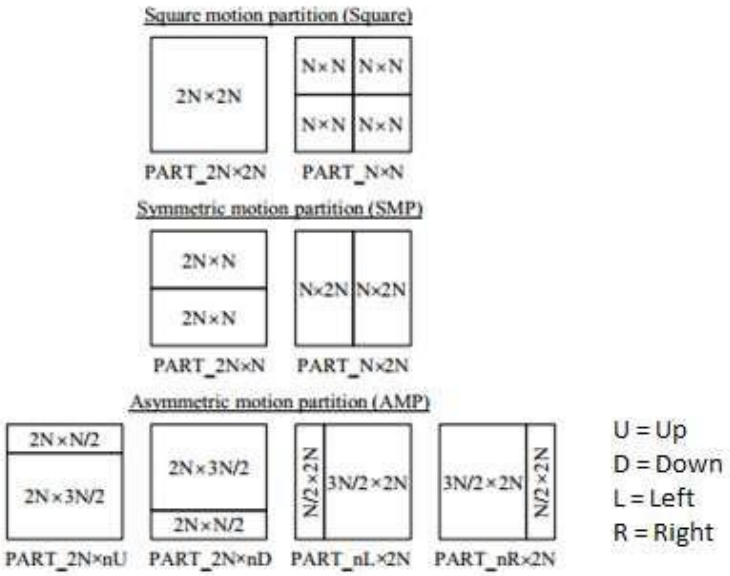

Figure 2-14 Partition modes in HEVC inter-prediction [22]

2.3.3 Transform, Scaling and Quantization

The HEVC standard uses transform coding of the prediction error residual in a similar manner as in prior standards. The residual block is partitioned into multiple square TBs. The supported transform block sizes are 4×4, 8×8, 16×16, and 32×32 [1]. Pre-scaling operation is not needed when using HEVC code since the rows of the transform matrix are a close approximations of values of uniformly scaled basis functions of the orthonormal DCT (Discrete Cosine Transform) [1]. Uniform reconstruction quantization (URQ) is used in the HEVC standard, with quantization scaling matrices supported for the various transform block sizes [1]. The range of the QP values is defined from 0 to 51, and an increase by 6 doubles the quantization step size such that the mapping of QP values to step sizes is approximately logarithmic.

2.3.4 Entropy Coding

Five generic binarization schemes are defined for symbol encoding, and it is specified which of these is applied to each type of syntax element. Context-adaptive binary arithmetic coding (CABAC) is then used for entropy coding. The basic method is similar to the CABAC scheme in AVC, but has undergone a number of improvements, especially in regard to reducing the number of adaptive coding contexts, increasing the use of fast “bypass” coding, and improving the ability for parallel processing to increase the throughput.

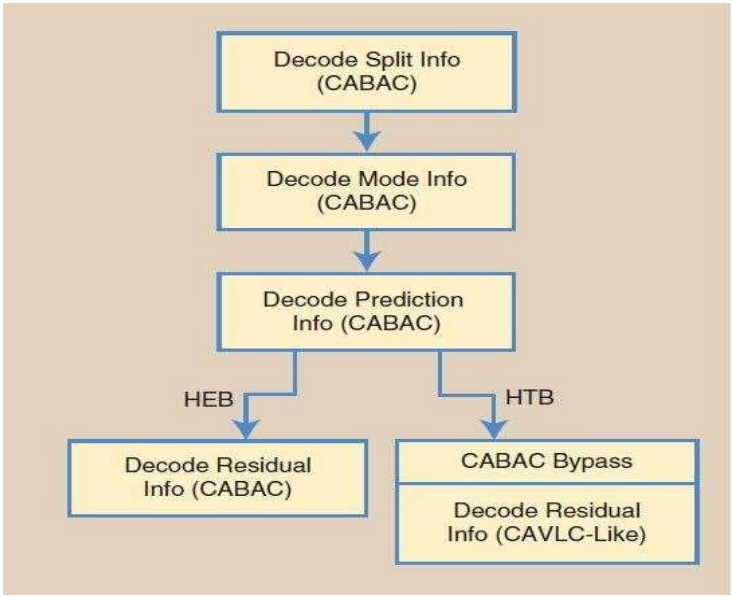

Figure 2-15 HEVC entropy coding [14]

Figure 2-16 Example of waveform processing [1]

2.3.5 In-loop Filtering

One or two filtering stages can be optionally applied (within the inter-picture prediction loop) before writing the reconstructed picture into the decoded picture buffer. A deblocking filter (DBF) is used that is similar to the one in AVC; however the DBF design has been simplified with regard to its decision-making and filtering processes and also has been made more friendly to parallel processing. The second stage, called the sample adaptive offset (SAO) filter, is a non-linear amplitude mapping. The goal of SAO is to improve the reconstruction of the signal amplitude by adding an offset based on a look-up table mapping that is controlled by the encoder. Two types of SAO operation can be selected for each CTB—the band offset and edge offset modes, where depending on additional criteria (amplitude or local directional amplitude constellation) an offset value is added to the reconstructed sample amplitude.

2.4 Summary

This chapter outlines the coding tools of the HEVC codec. The intent of the HEVC is to create a standard capable of providing good video quality at substantially lower bit rates than previous standards. Chapter 3 outlines the description of intra-prediction mode decision and the proposed complexity reduction using coding unit early termination with tree split/merge algorithm for transform unit.

Chapter 3

Intra-prediction and Early Termination Algorithm

3.1 Introduction to Intra Prediction

In H.264, intra prediction [22][24][25][26] is based on spatial extrapolation of samples from previously decoded image blocks, followed by integer discrete cosine transform (DCT) [23] based coding [E3]. HEVC uses the same principle, but further extends it to efficiently representing a wider range of textural and structural information in images. HEVC contains several elements for improving the efficiency of intra prediction over earlier approaches. The introduced methods can model accurately different structures as well as smooth regions with gradually changing sample values. The word “intra” indicates that the considered frame uses only pixels within itself for the prediction process.

3.2 Intra Prediction

Intra coding in HEVC is be considered as an extension of H.264/AVC [28], since both approaches are based on spatial sample prediction followed by transform coding. The basic elements in the HEVC intra coding design include: 1) quad tree-based coding structure following the HEVC block coding architecture; 2) angular prediction with 33 prediction directions; 3) planar prediction to generate smooth sample surfaces; 4) adaptive smoothing of the reference samples; 5) filtering of the prediction block boundary samples; 6) prediction mode-dependent residual transform and coefficient scanning; 7) intra mode coding based on contextual information. HEVC contains several elements in improving the efficiency of intra prediction. The introduced methods can model accurately different directional structures as well as smooth regions with gradually changing sample values. There is also emphasis on avoiding introduction of artificial edges with potential blocking effects. This is achieved by adaptive smoothing of the reference samples and smoothing the generated prediction boundary samples for DC and directly horizontal and vertical modes. All the prediction modes use the same basic set of reference samples from above and to the left of the image block to be predicted. In the following sections, the reference samples are denoted by with (x, y) having its origin one pixel above and to the left of the block’s top-left corner. [44]

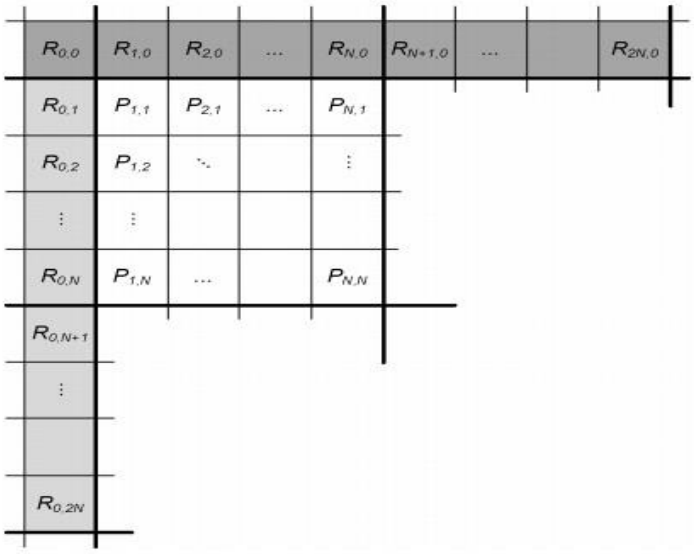

Figure 3-1 Reference samples Rx,y used in prediction to obtain predicted samples Px,y for a block of size N x N samples [3].

As shown in Figure 3-1 the reference samples located on top left and above of the image block to be predicted and denoted by Rx,y while the predicted block is denoted by Px,y where (x,y) denotes the position of the predicted sample value. In some cases neighboring reference samples may be unavailable for intra prediction. Hence in such cases missing reference samples on the top boundary are obtained by copying the closest available reference sample [3]. In Figure 3-1 the missing reference samples in the top boundary are obtained by copying the closest available reference sample from the left while the missing reference samples in the left boundary are generated by copying the reference samples below.

Intra picture prediction operates according to the TB size, and previously decoded boundary samples from spatially neighboring TBs are used to form the prediction signal. Directional prediction with 33 different directional orientations is defined for TB sizes from 4×4 up to 32×32. Alternatively, planar prediction and DC can also be used. For chroma, the horizontal, vertical, planar, and DC prediction modes can be explicitly signaled, or the chroma prediction mode can be indicated to be the same as the luma prediction mode. Each CB can be coded by one of several coding types, depending on the slice type. Similar to H.264/MPEG-4

AVC [2], intra picture predictive coding is supported in all slice types. HEVC supports various intra picture predictive coding methods referred to as Intra−Angular, Intra−Planar, and Intra−DC. Figure 3-2 shows the luma intra prediction modes of HEVC [1].

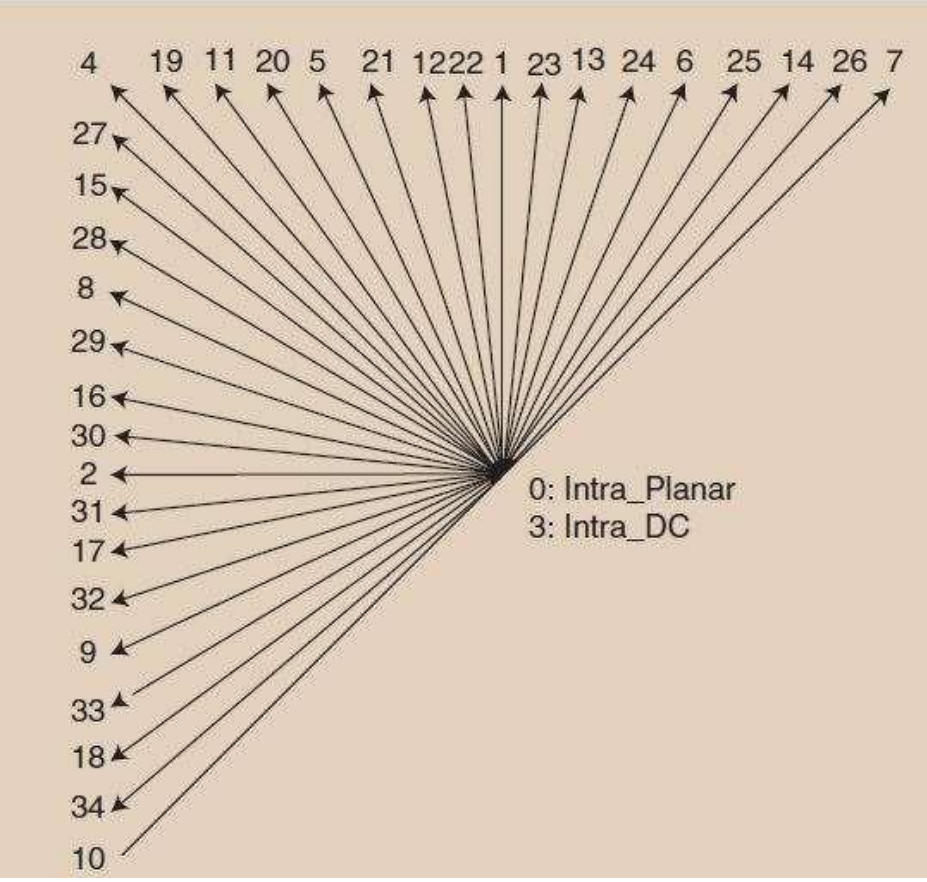

Figure 3-2 Luma intra prediction modes of HEVC [14]

3.3 Intra Angular Prediction

The intra picture prediction of the HEVC standard similarly operates in the spatial domain, but is extended significantly mainly due to the increased size of the TB and an increased number of selectable prediction directions. The HEVC standard supports a total of 33 prediction directions 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 to reflect the observed statistical prevalence of the angles and the effectiveness of the signal prediction processing. When using an Intra Angular mode, each TB is predicted directionally from spatially neighboring

samples that are reconstructed before being used for this prediction. For a TB of size N×N, a total of 4N+1 spatially neighboring samples may be used for the prediction, as shown in figure 3-1. When available from preceding decoding operations, samples from lower left TBs can be used for prediction in HEVC in addition to samples from TBs at the left, above, and above right of the current TB [1].

The prediction process of the Intra Angular modes can involve extrapolating samples from the projected reference sample location according to a given directionality. To remove the need for sample-by-sample switching between reference row and column buffers, for Intra−Angular[k] with k in the range of 2–17, the samples located in the above row are projected as additional samples located in the left column and with k in the range of 18–34, the samples located at the left column are projected as samples located in the above row. To improve the intrapicture prediction accuracy, the projected reference sample location is computed with 1/32 sample accuracy. Bilinear interpolation is used to obtain the value of the projected reference sample using two closest reference samples located at integer positions [1].

3.4 Intra−Planar and Intra−DC Prediction

In addition to Intra−Angular prediction that targets regions with strong directional edges, HEVC supports two alternative prediction methods, Intra−Planar and Intra−DC. Intra−DC prediction uses an average value of reference samples for the predictions average values of two linear predictions using four corner reference samples are used in Intra−Planar prediction to prevent discontinuities along the block boundaries [1].

3.5 Mode Coding

HEVC supports a total of 33 Intra−Angular prediction modes (Figure 3-2) and Intra−Planar and Intra−DC prediction modes for luma prediction for all block sizes. Due to the increased number of directions, HEVC considers three most probable modes (MPMs) when coding the luma intra picture prediction mode.

Among the three most probable modes, the first two are initialized by the luma intra picture prediction modes of the above and left PBs if those PBs are available and are coded using an intra picture prediction mode. Any unavailable prediction mode is considered to be intra−DC. The PB above the luma CTB is always considered to be unavailable in order to avoid the need to store a line buffer of neighboring

luma prediction modes. When the first two most probable modes are not equal, the third most probable mode is set equal to Intra−Planar, Intra−DC, or Intra−Angular, according to which of these modes, in this order, is not a duplicate of one of the first two modes. When the first two most probable modes are the same, if this first mode has the value Intra−Planar or Intra−DC, the second and third most probable modes are assigned as Intra−Planar, Intra−DC, or Intra−Angular, according to which of these modes, in this order, are not duplicates [1]. When the first two most probable modes are the same and the first mode has an Intra−Angular value, the second and third most probable modes are chosen as the two angular prediction modes that are closest to the angle (i.e., the value of k) of the first. In the case that the current luma prediction mode is one of three MPMs, only the MPM index is transmitted to the decoder. Otherwise, the index of the current luma prediction mode excluding the three MPMs is transmitted to the decoder by using a 5-b fixed length code. For chroma intra picture prediction, HEVC allows the encoder to select one of five modes: Intra−Planar, Intra−Angular, Intra−Angular, Intra−DC, and Intra−Derived. The intra derived mode specifies that the chroma prediction uses the same angular direction as the luma prediction. With this scheme, all angular modes specified for luma in HEVC can, in principle, also be used in the chroma prediction, and a good tradeoff is achieved between prediction accuracy and the signaling overhead. The selected chroma prediction mode is coded directly [1]. Table 3-2 shows the HEVC encoder complexity for CU and PB blocks.

Table 3-1 Luma intra prediction modes supported by different PU sizes [14]

PU Size	Intraprediction Modes
4 × 4	0–16, 34
8 × 8	0–34
16 × 16	0–34
32 × 32	0–34
64 × 64	0–2, 34

Table 3-2 Current Problem-Complexity for HEVC [33]

Size of PB	Number of PBs in a 64x64 CU	Number of Modes to be Tested in each PB	Total number of modes to be tested at this level
32x32	4	35	140
16x16	16	35	560
8x8	64	35	2240
4x4	256	35	8960
		Total	11900

3.6 Proposed Solution

A large number of researchers have proposed various techniques for making the intra prediction process faster. [20–28] A two step method is proposed as a solution. In CU splitting, decision is made whether to split the current CU further by analyzing the CU texture characteristics. This determines the complexity of the CU block and then decision is made to further split or non-split the CU. It is followed by a TU mode decision to find the optimal prediction mode from the 35 prediction modes. Proposed method will use tree split/tree merge algorithm [19].

3.6.1 CU early termination

It is shown in [12] that when texture of a CU is complex it is has higher computational complexity. So, when the CU texture is complex the CU is split into smaller units until the best-size with flat CU texture is found.

In the first stage, to decrease the computational complexity, the down-sampling method is exploited by applying a 2:1 down sampling filter by a simple averaging operator to the current CU and other CU have the similar operation as shown in figure 3-2.

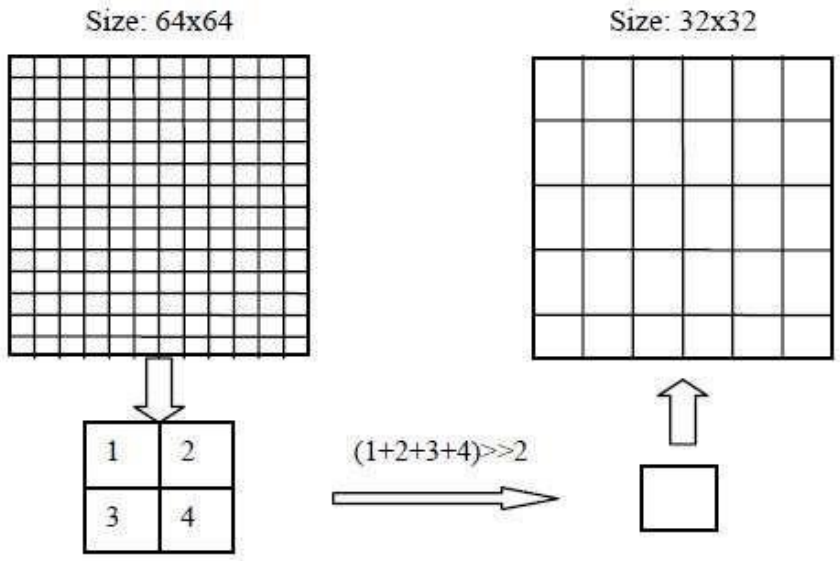

Figure 3-3 Illustration of simple averaging based down-sampling on 64x64 CU [27]

After down-sampling, the complexity of the original LCU can be executed as its texture complexity is defined as:

$$E_{com} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left[p(i,j) - \frac{1}{N}\frac{1}{N}\left(\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} p(i,j) \right) \right]^2 \quad (1)$$

Where Ecom represents the texture complexity, N is the size of the current CU, p(i, j) is the pixel and (i, j) is the coordinate in CU. Based on the texture calculation, we set the two threshold for tradeoff of coding quality and complexity reduction, they are Thre1 and Thre2. When the complexity is greater thanThre1, the CU is split; when the complexity is less than Thre2, the CU is optimal; When the complexity is between Thre1 and Thre2 , referring the HEVC reference software[38] is applied.

3.6.2 Priority of TU Size in Mode Decision

The HM 16.0 depth-first RQT decision process adopts the top-down search order. It always starts from the maximum admissible size TU in a CU leaf and evaluates the possibility of further partitions. But from the earlier discussions, we conclude that the 32×32 TU does not offer as much rate-distortion efficiency

as the smaller TUs. Thus, we force the 32x32 TU to be the last candidate in the mode search procedure if it appears. By doing this, we are able to save computing power by skipping unnecessary TU partition evaluation. The modified RQT mode decision algorithm becomes a TU split-and-merge process as illustrated in Figure 3-4.

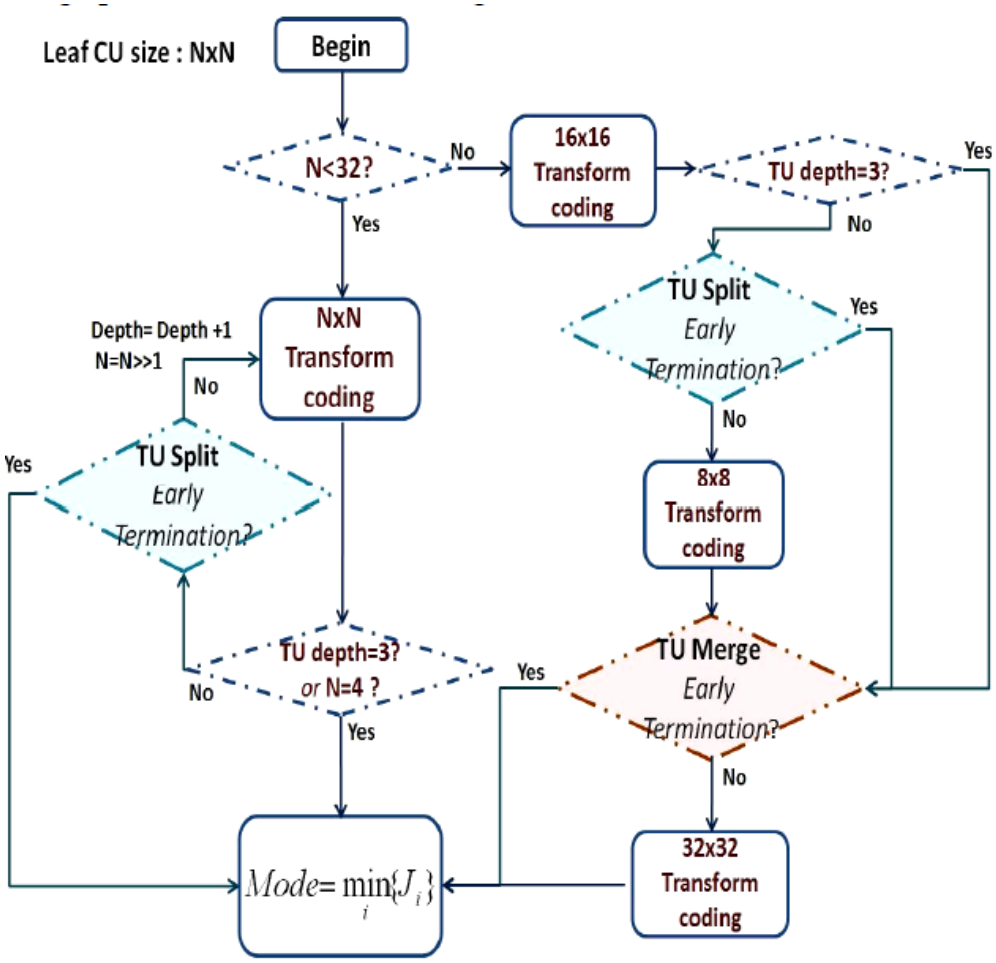

Figure. 3-4 TU Split-and-Merge RQT mode decision algorithm

3.7 Summary

This chapter has introduced different angular prediction modes, CU splitting algorithm and the TU mode decision. CU early termination with TU size mode decision algorithm is explained. Chapter 4 outlines simulations and results for different resolution test sequences (Appendix A) for unmodified and modified HM version code base [38].

Chapter 4

Results

4.1 Test Conditions

In order to evaluate the performance of the proposed intra prediction algorithm, the algorithm is implemented on the recent HEVC reference software (HM 16.0) [38]. The intra main profile is used for coding with the intra period set as 1 and frame rate set at 30 fps. The proposed algorithm is evaluated with 4 QPs of 22, 27, 32 and 37 using the following test sequences recommended by JCT-VC [35]. A frame of each test sequence is shown in Appendix A.

Table 4-1 Test sequences used [39]

No.	Sequence Name	Resolution	Type	No. of frames
1.	RaceHorses	416x240	WQVGA	30
2.	BasketballDrillText	832x480	WVGA	30
3.	KristenAndSara	1280x720	SD	30
4.	BasketballDrive	1920x1080	HD	30
5.	PeopleOnStreet	2560x1600	WQHD	30

4.2 Encoder Complexity Reduction

With the proposed CU early termination algorithm, encoder complexity in terms of encoding time for the test sequences is reduced by 12-24% as compared to the unmodified encoding HM16.0 [38]. The following test results (figures 4-1 to 4-5) show the difference in encoding time of the original HM16.0 and the proposed for different quantization parameter (QP) values as suggested by JCTVC [35].

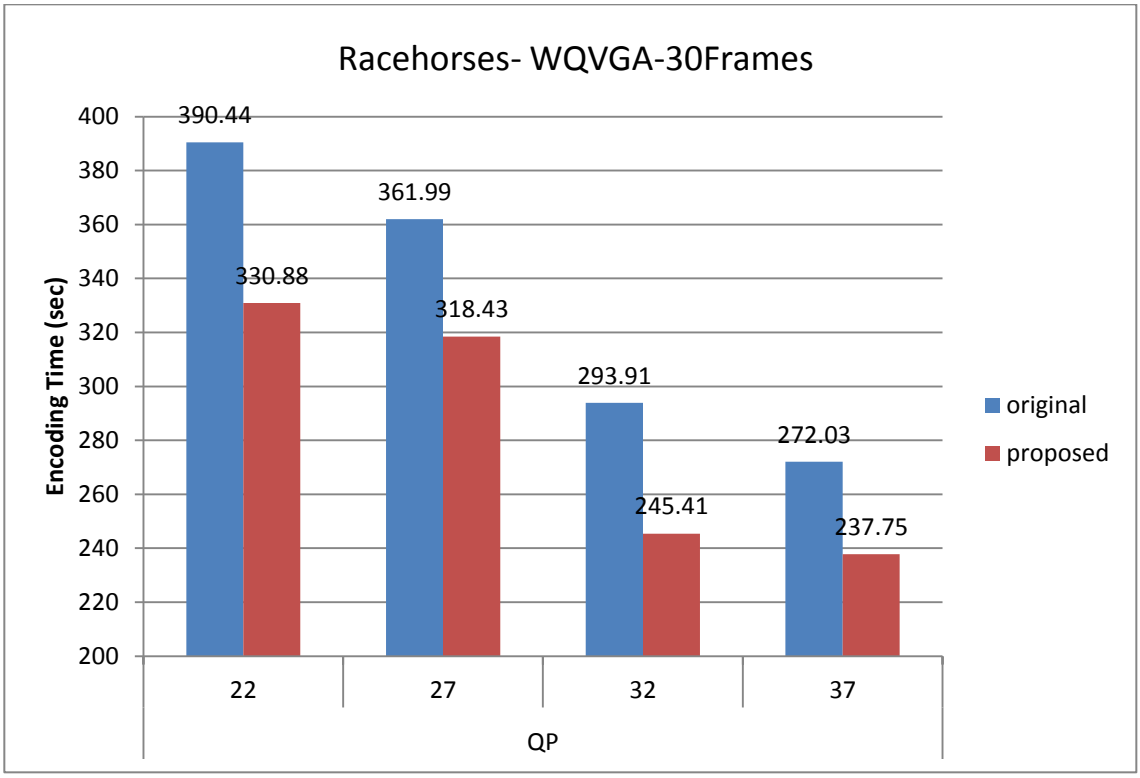

Figure 4-1 Encoding time vs. quantization parameter for Racehorses

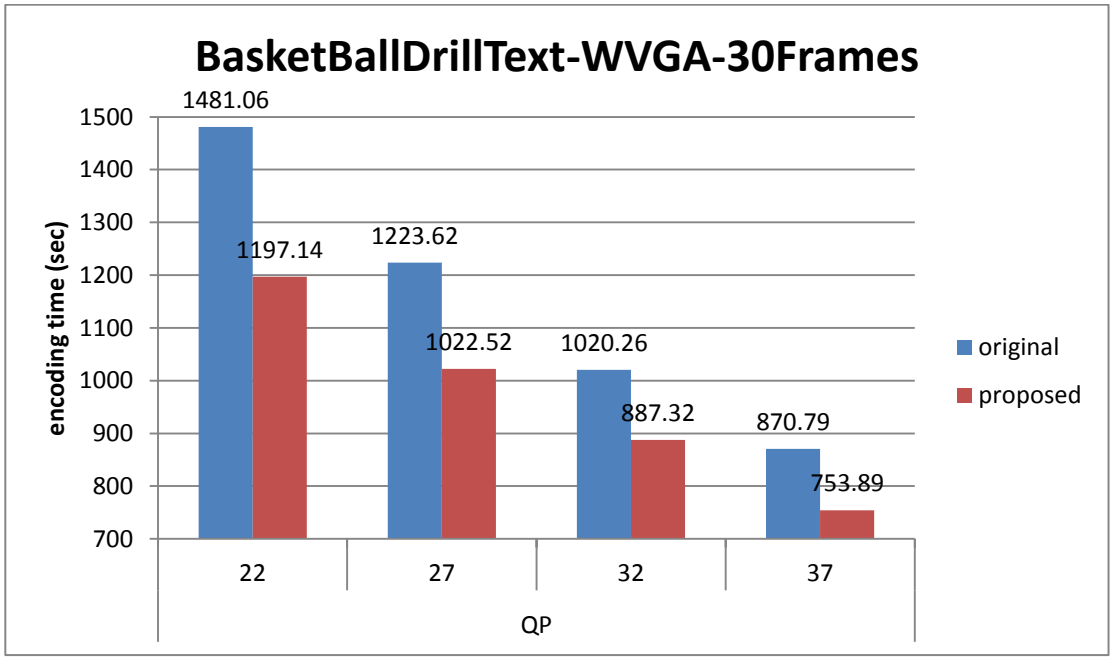

Figure 4-2 Encoding time vs. quantization parameter for BasketBallDrillText

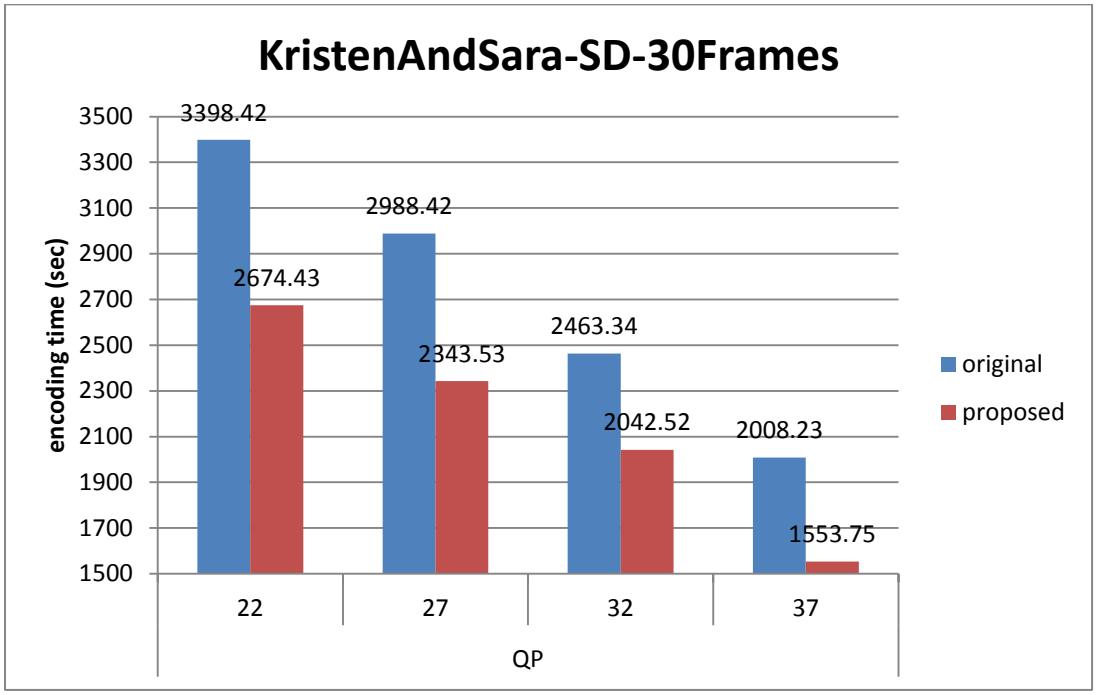

Figure 4-3 Encoding time vs. quantization parameter for KristenAndSara

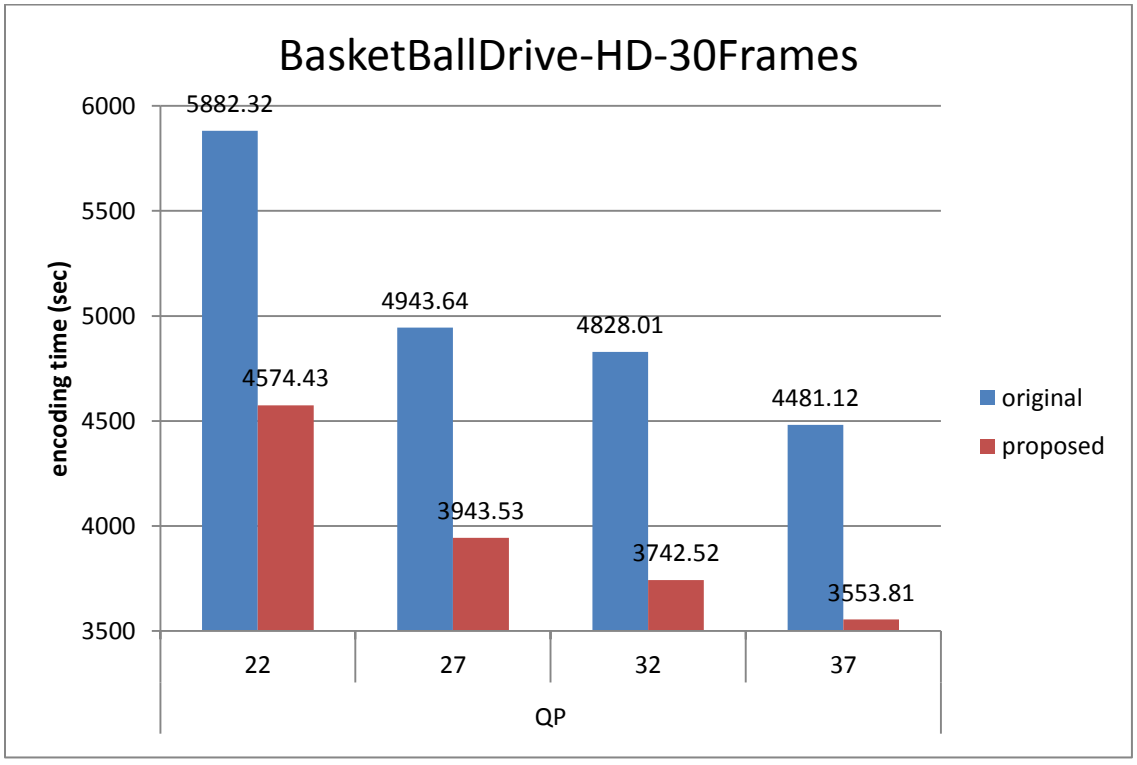

Figure 4-4 Encoding time vs. quantization parameter for BasketBallDrive

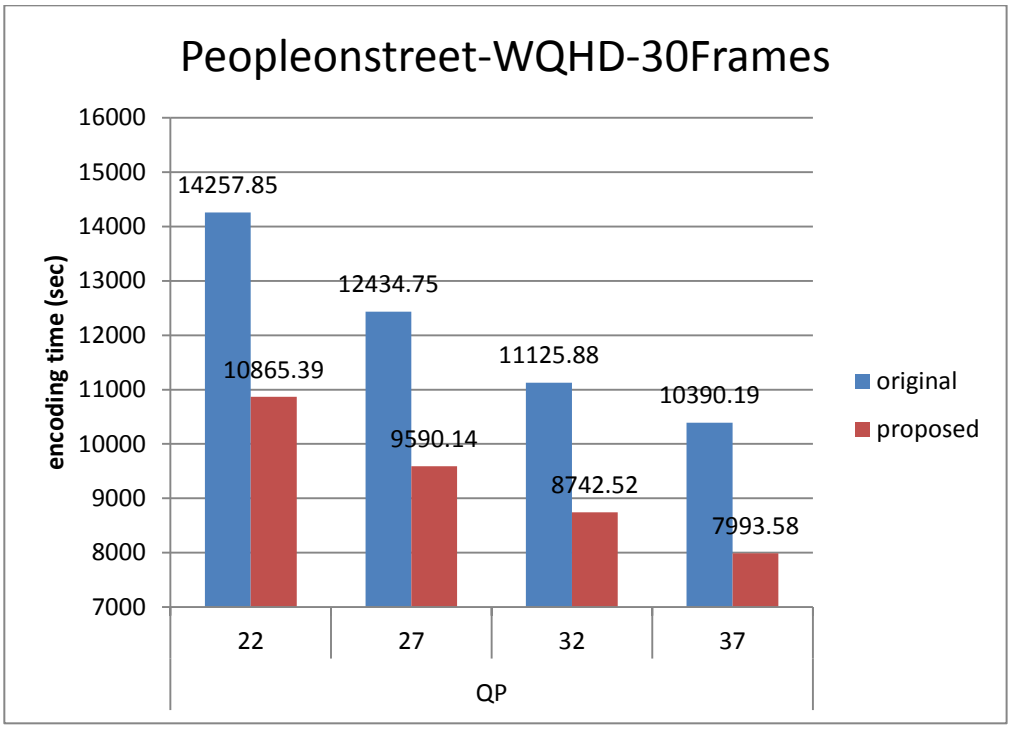

Figure 4-5 Encoding time vs. quantization parameter for Peopleonstreet

4.3 BD-PSNR

To objectively evaluate the coding efficiency of video codecs, Bjøntegaard Delta PSNR (BD-PSNR) was proposed [36]. Based on the rate-distortion (R-D) curve fitting, BD-PSNR is able to provide a good evaluation of the R-D performance [36]. BD-PSNR is a curve fitting metric based on rate and distortion of the video sequence. However, this does not take into account the complexity of the encoder, but the BD metric tells a lot about the quality of the video sequence [30] [31]. Ideally, BD-PSNR should increase and BD-bitrate should decrease. The following results show a plot of BD-PSNR versus the quantization parameter (QP). It can be observed from figures 4-6 to 4-10 that there is a slight drop in PSNR using BD metrics for the proposed algorithm for the HM16.0 in the range of 0.29 dB to 0.51 dB.

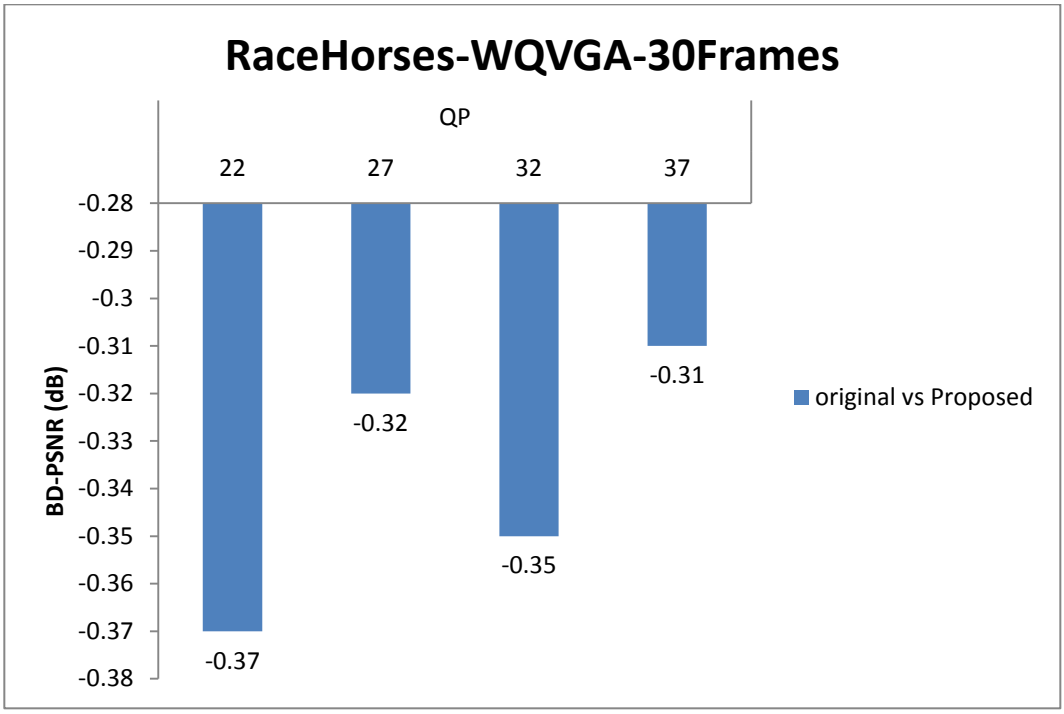

Figure 4-6 BD-PSNR vs. quantization parameter for RaceHorses

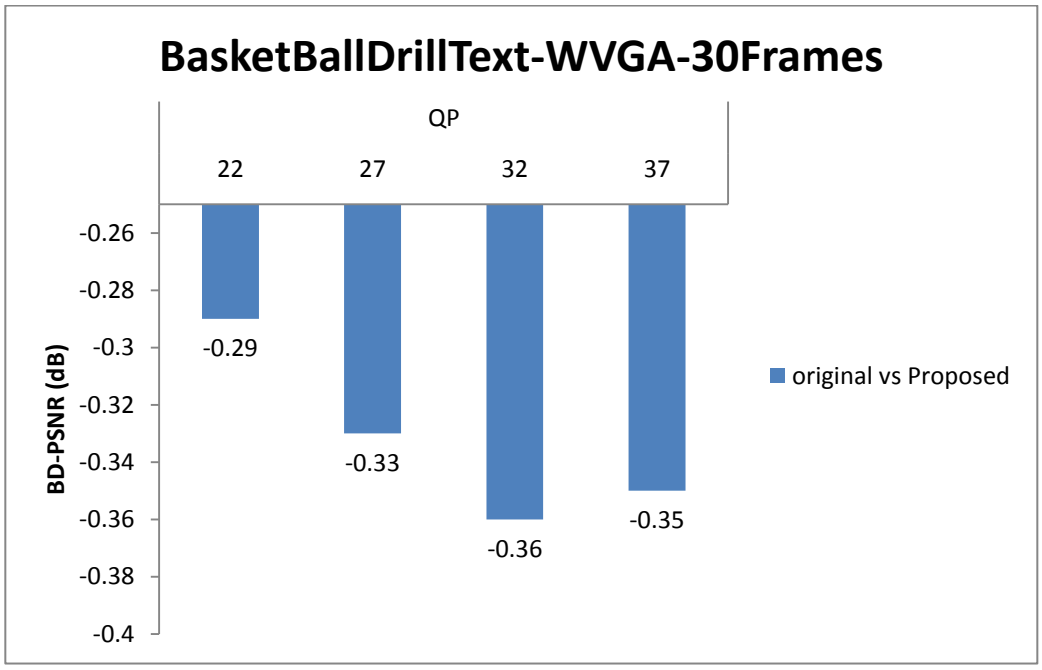

Figure 4-7 BD-PSNR vs. quantization parameter for BasketBallDrillText

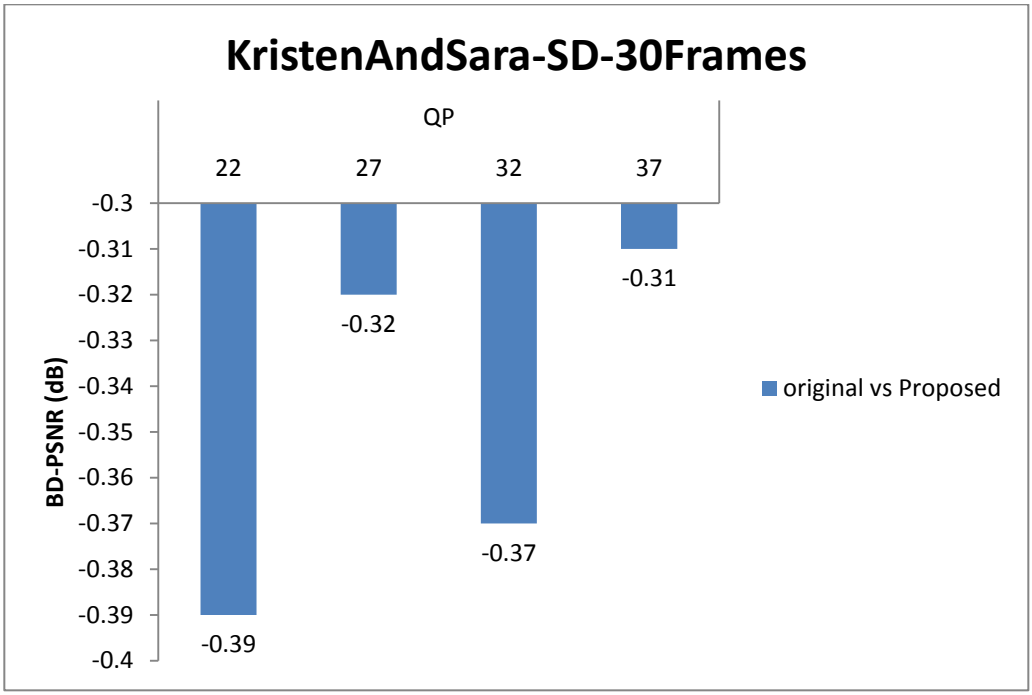

Figure 4-8 BD-PSNR vs. quantization parameter for KristenAndSara

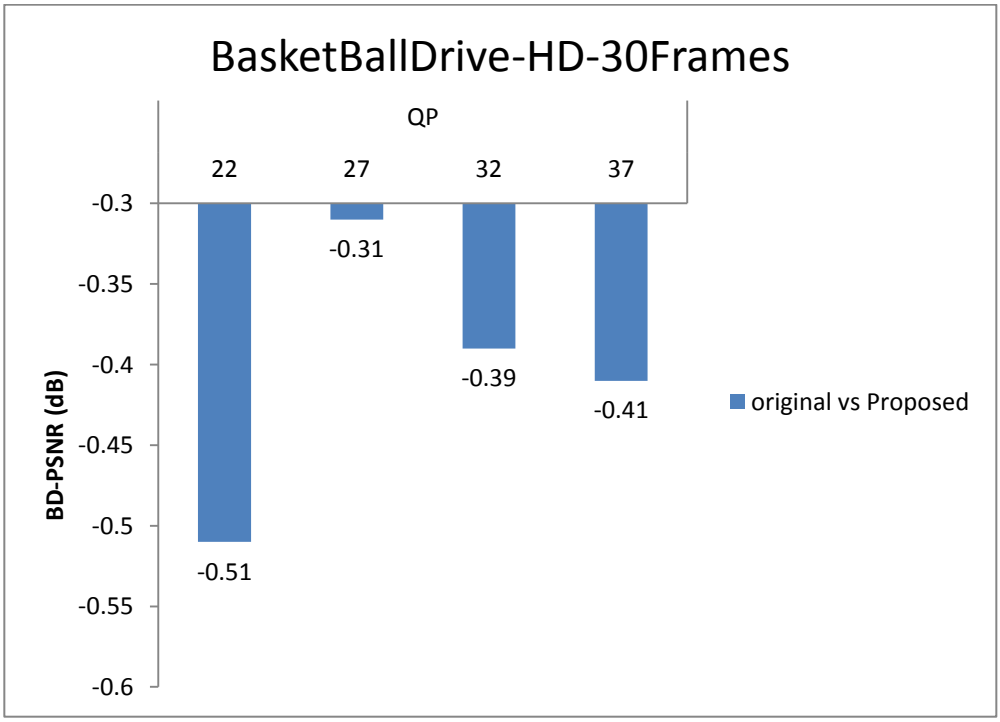

Figure 4-9 BD-PSNR vs. quantization parameter for BasketBallDrive

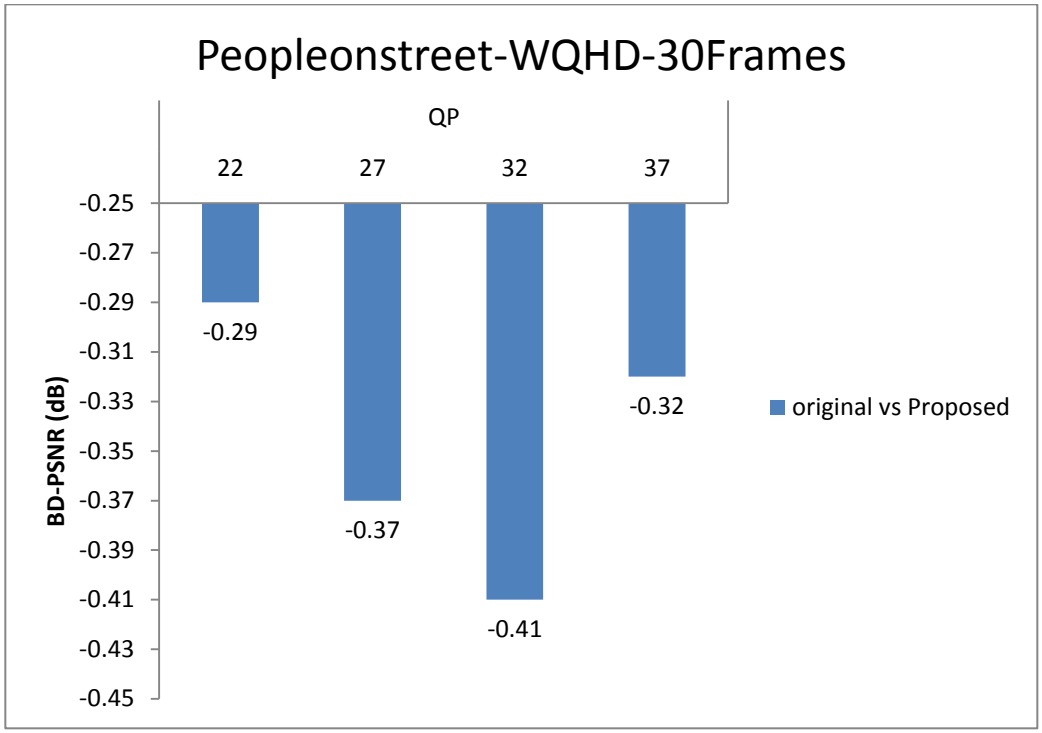

Figure 4-10 BD-PSNR vs. quantization parameter for Peopleonstreet

4.4 BD-bitrate

BD-bitrate [Appendix C] is a metric similar to the BD-PSNR metric which determines the quality of encoded video sequence along with the of measure the output bitstream of encoded video sequence. It can be observed from figures 4-11 to 4-15 that there is a slight increase in bitrate using the BD metrics [Appendix C] for the proposed algorithm for HM16.0 in the range of 8 kbps to 13 kbps (b-Bit).

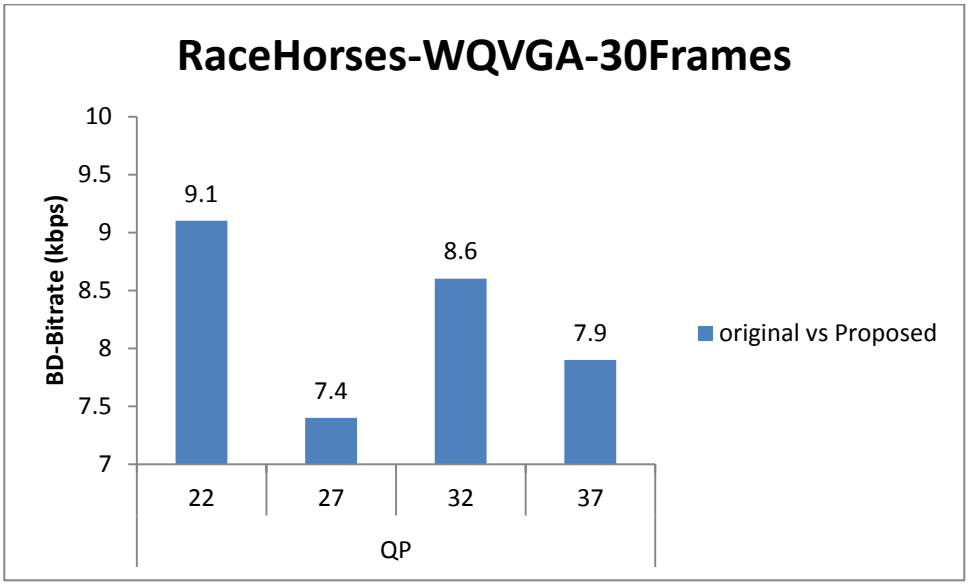

Figure 4-11 BD-bitrate vs. quantization parameter for RaceHorses

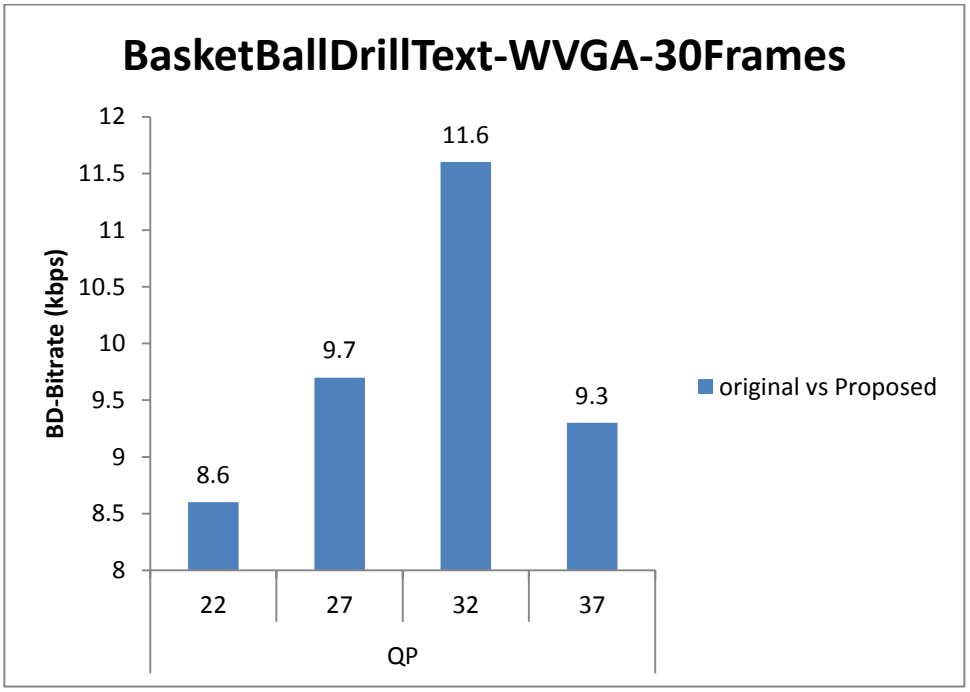

Figure 4-12 BD-bitrate vs. quantization parameter for BasketballDrillText

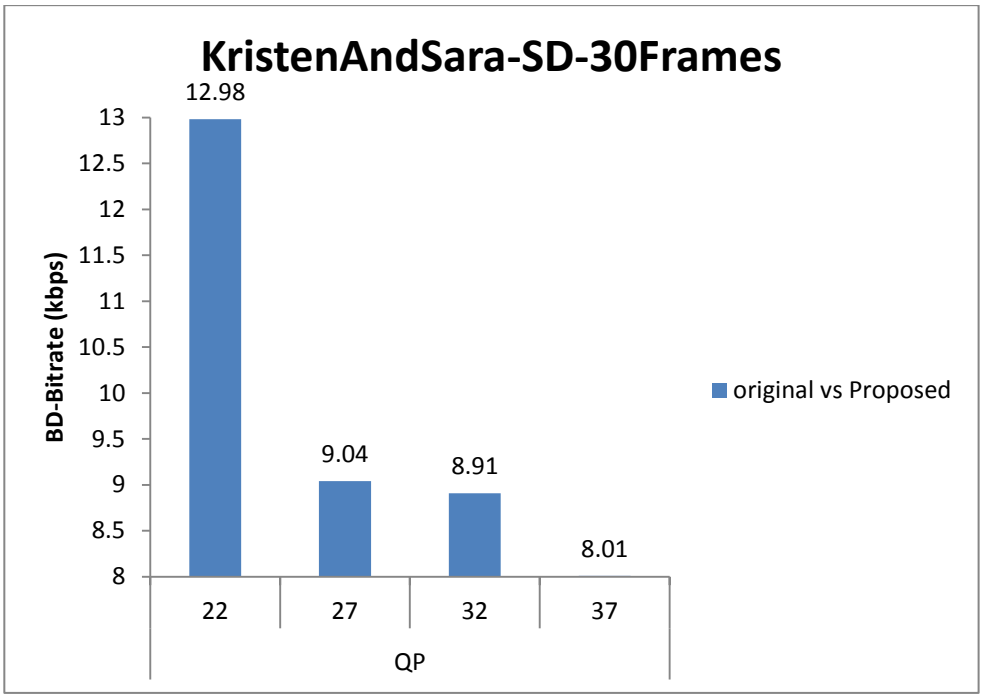

Figure 4-13 BD-bitrate vs. quantization parameter for KristenAndSara

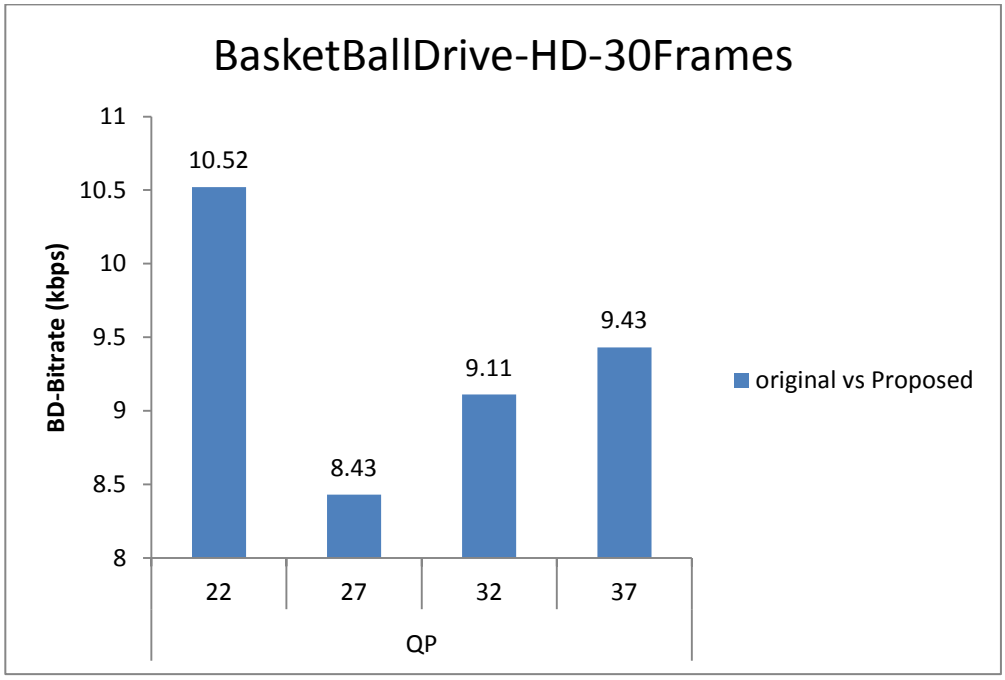

Figure 4-14 BD-bitrate vs. quantization parameter for BasketballDrive

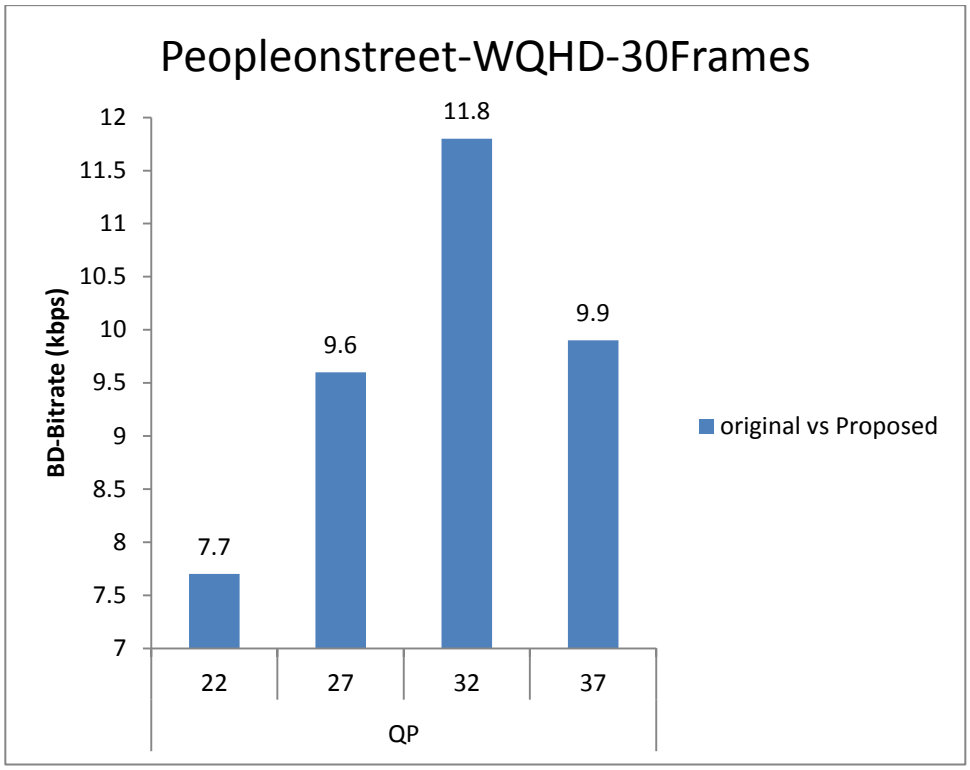

Figure 4-15 BD-bitrate vs. quantization parameter for Peopleonstreet

4.5 Rate Distortion Plot (RD Plot)

The proposed algorithm has negligible PSNR loss and bitrate increase. Figures 4-16 to 4-20 show bitrate-PSNR graphs for the various test sequences. It can be seen that performance is similar to the original HM16.0 encoder.

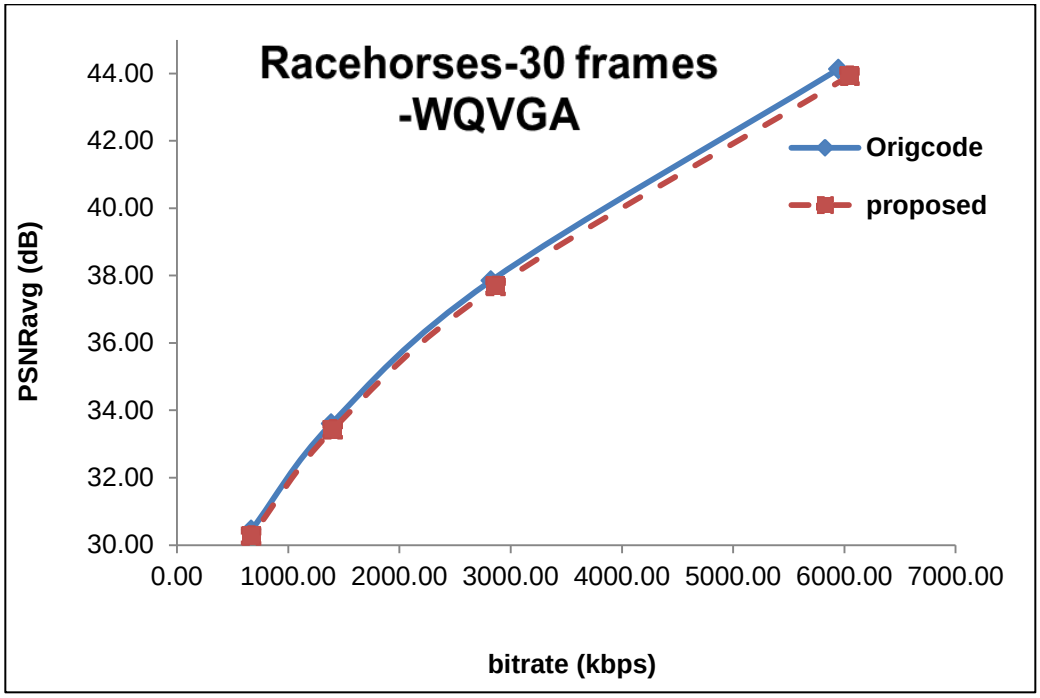

Figure 4-16 PSNRavg vs. bitrate for Racehorses

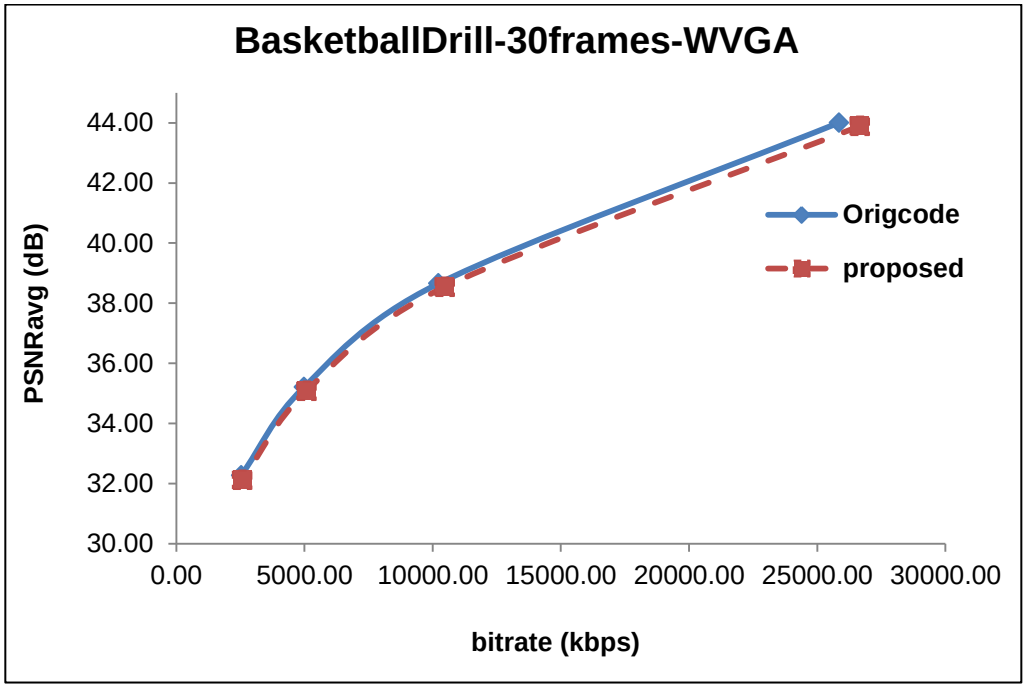

Figure 4-17 PSNRavg vs. bitrate for BasketballDrillText

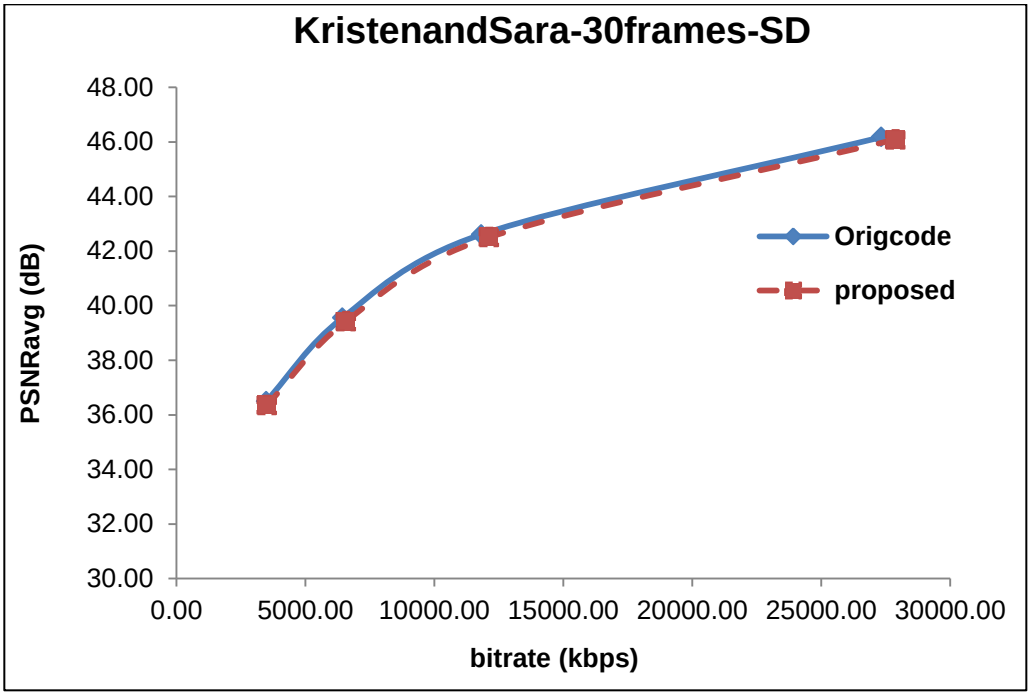

Figure 4-18 PSNRavg vs. bitrate for KristenAndSara

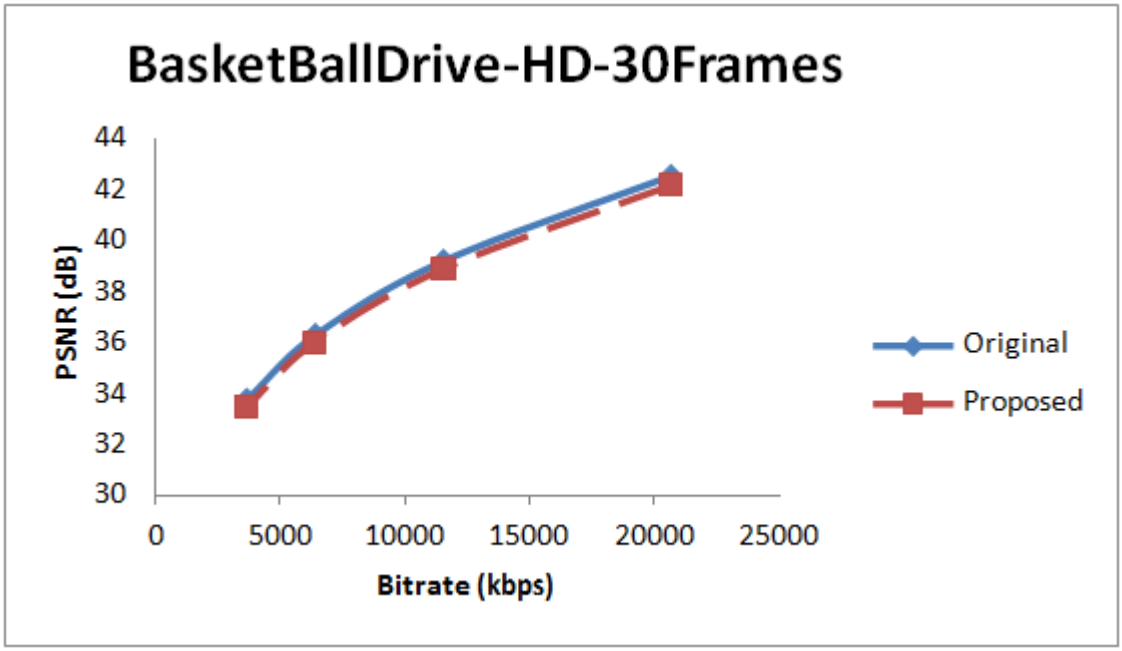

Figure 4-19 PSNR vs. bitrate for BasketBallDrive

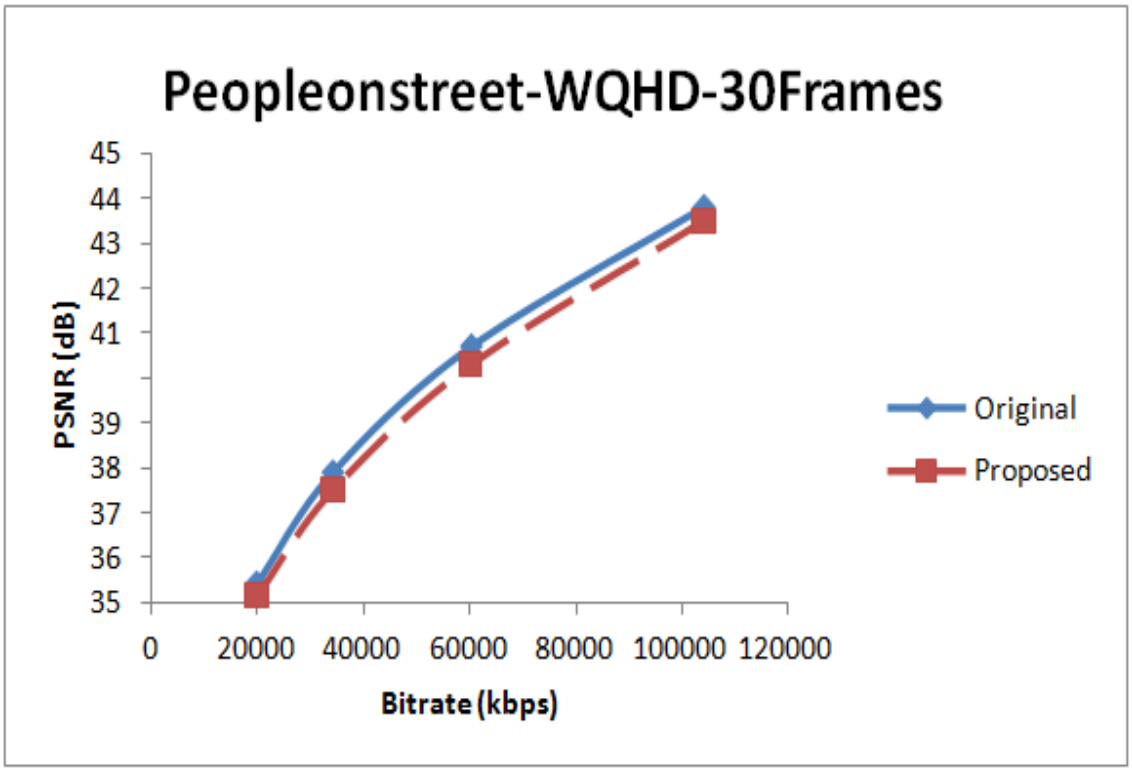

Figure 4-20 PSNR vs. bitrate for Peopleonstreet

4.6 Bitstream Size Gain

Figures 4-21 to 4-25 show the encoded bitstream size for the original HM16.0 and the proposed HM 16.0 encoded for different quantization parameter values. It can be observed that there is only 1% to 5% increase in bitstream size.

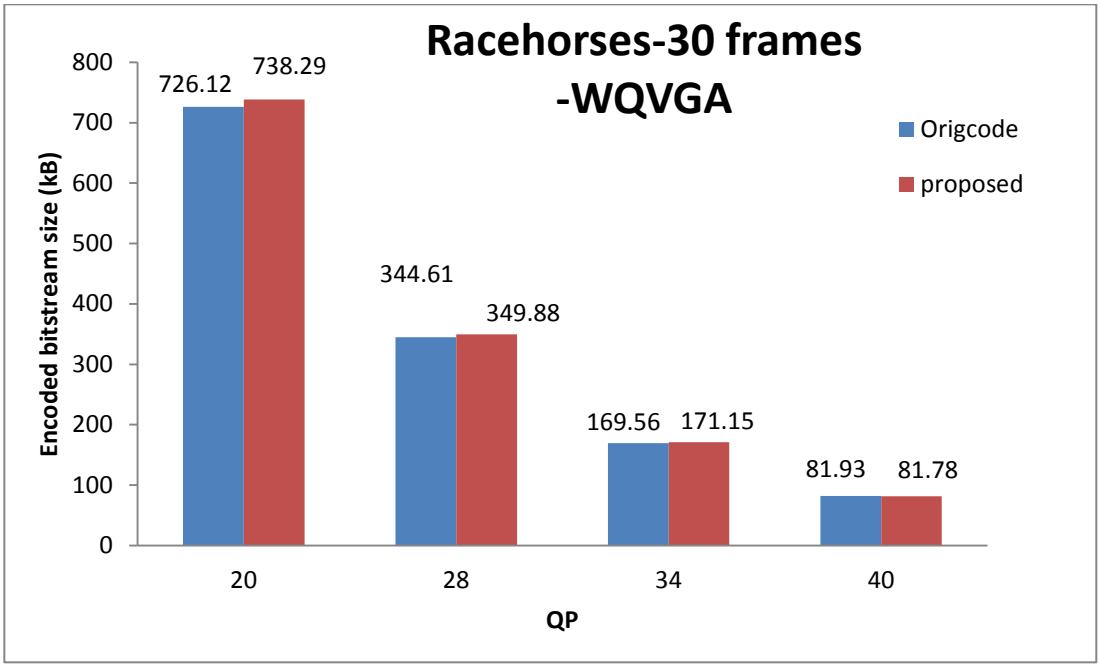

Figure 4-21 Encoded bitstream size vs. quantization parameter for Racehorses

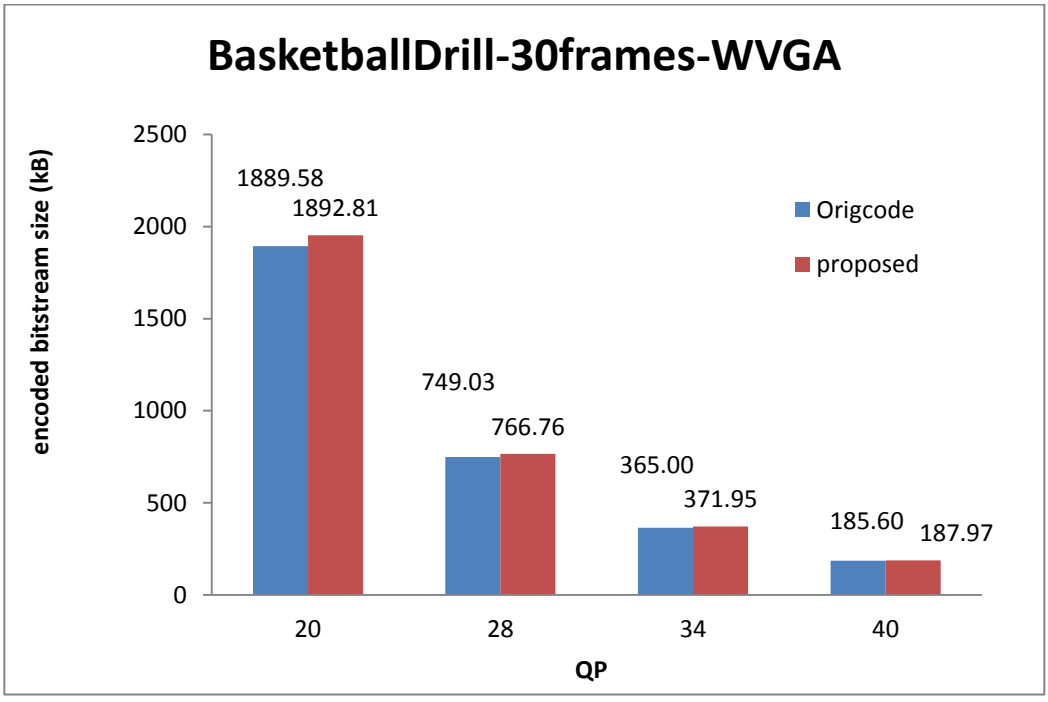

Figure 4-22 Encoded bitstream size vs. quantization parameter for BasketballDrillText

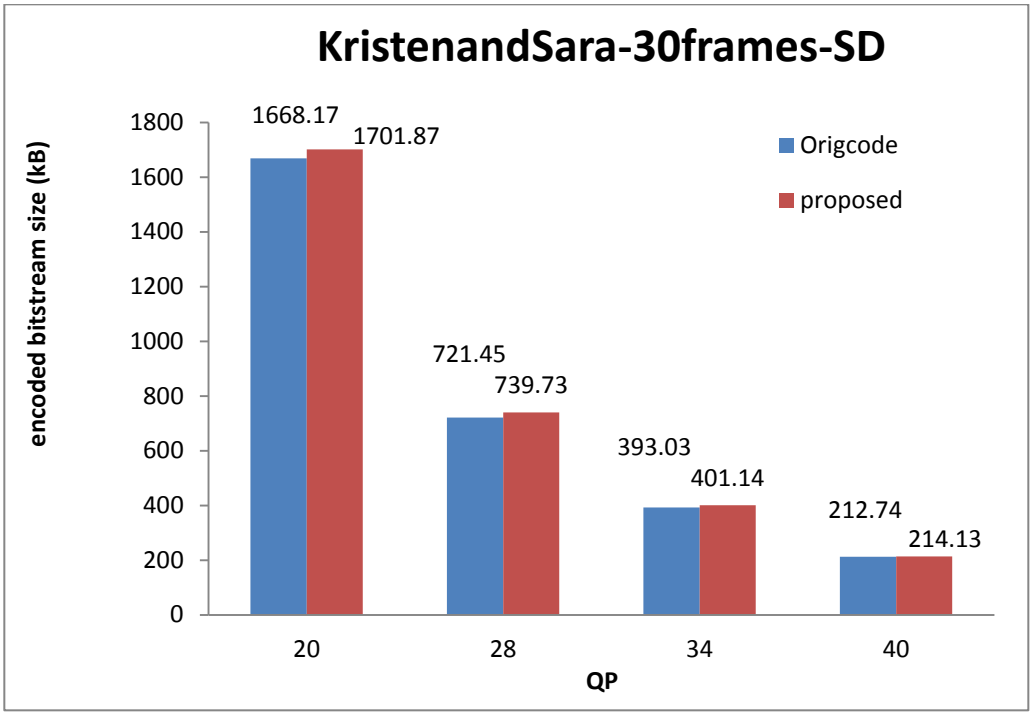

Figure 4-23 Encoded bitstream size vs. quantization parameter for KristenAndSara

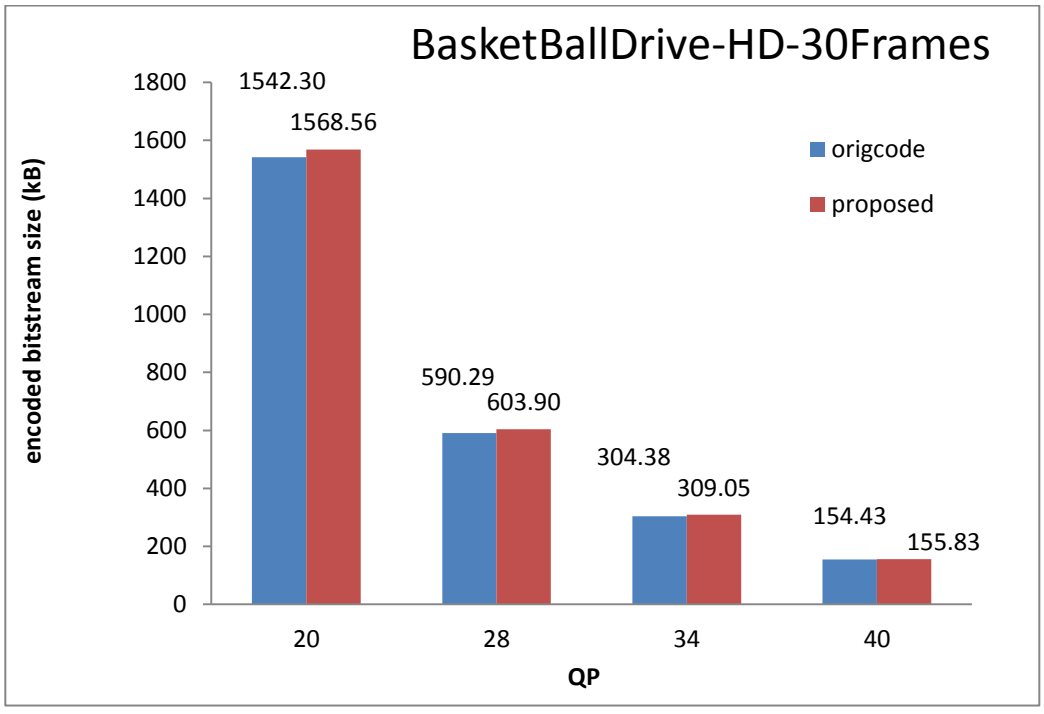

Figure 4-24 Encoded bitstream size vs. quantization parameter for BasketBallDrive

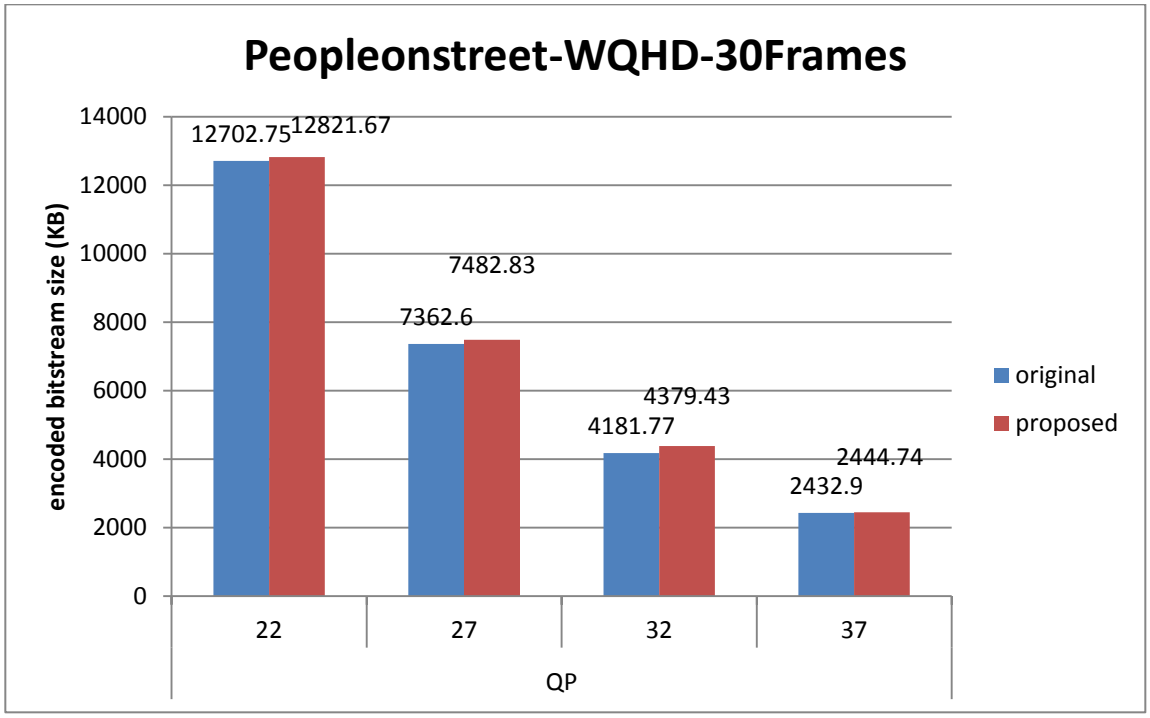

Figure 4-25 Encoded bitstream size vs. quantization parameter for Peopleonstreet

4.7 Percentage Decrease in Encoding Time

Figures 4-26 thru 4-30 show 12-24% decrease in encoding time which shows the decrease in the complexity of the encoder with proposed CU splitting algorithm and the TU mode decision as compared to the original HM16.0 algorithm [38].

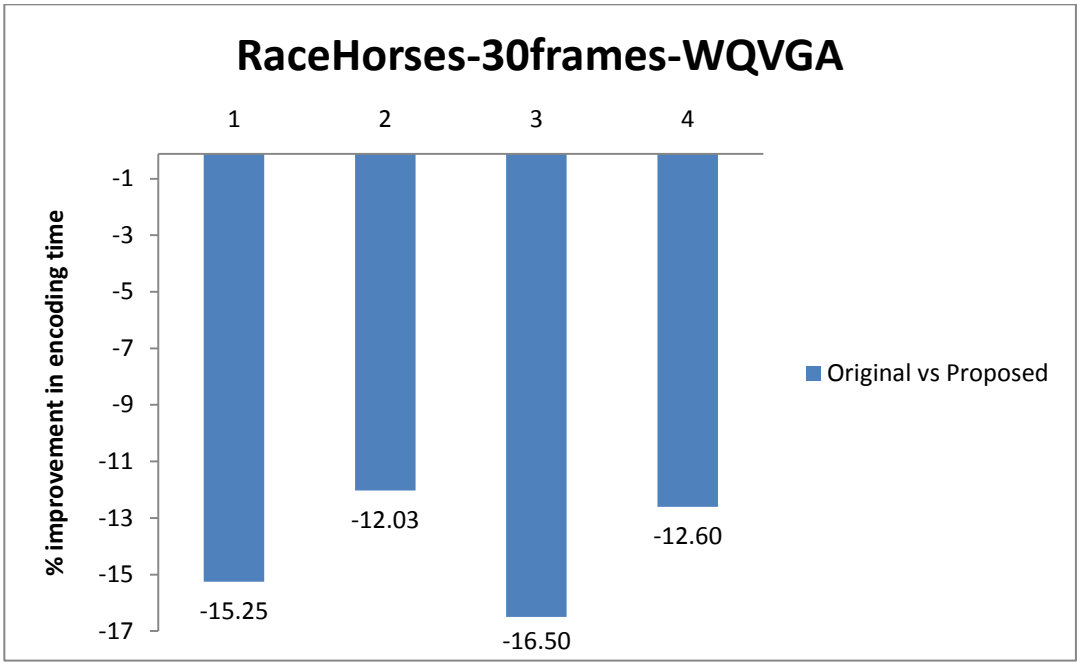

Figure 4-26 % improvement in encoding time vs. quantization parameter for RaceHorses

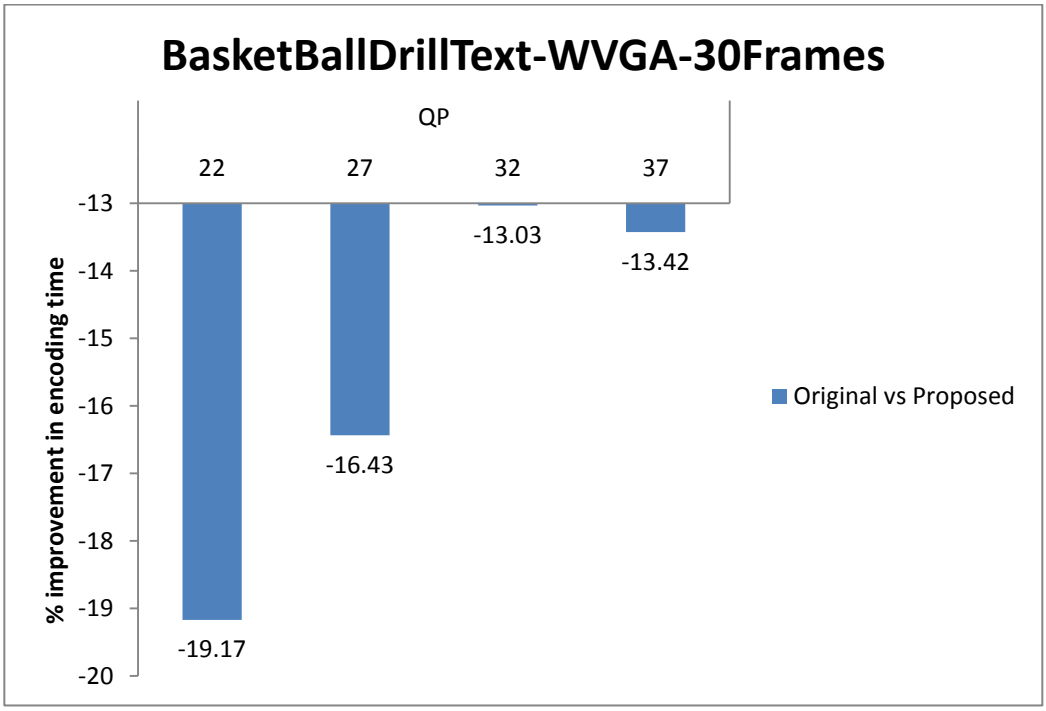

Figure 4-27 % improvement in encoding time vs. quantization parameter for BasketBalldrillText

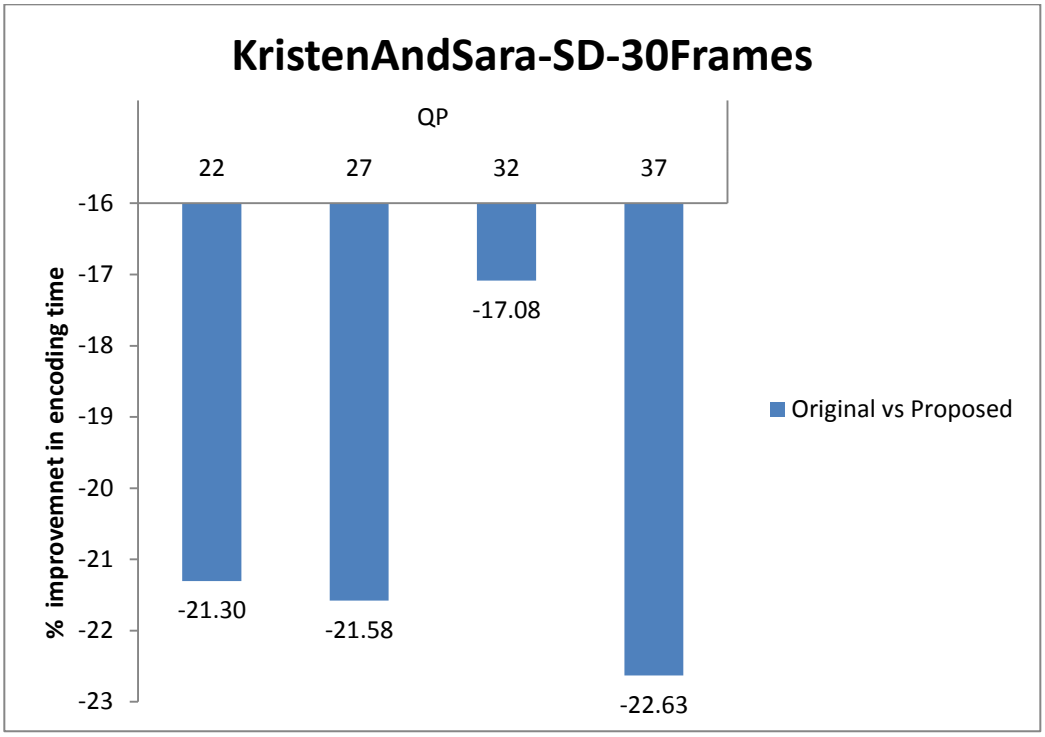

Figure 4-28 % improvement in encoding time vs. quantization parameter for KristenAndSara

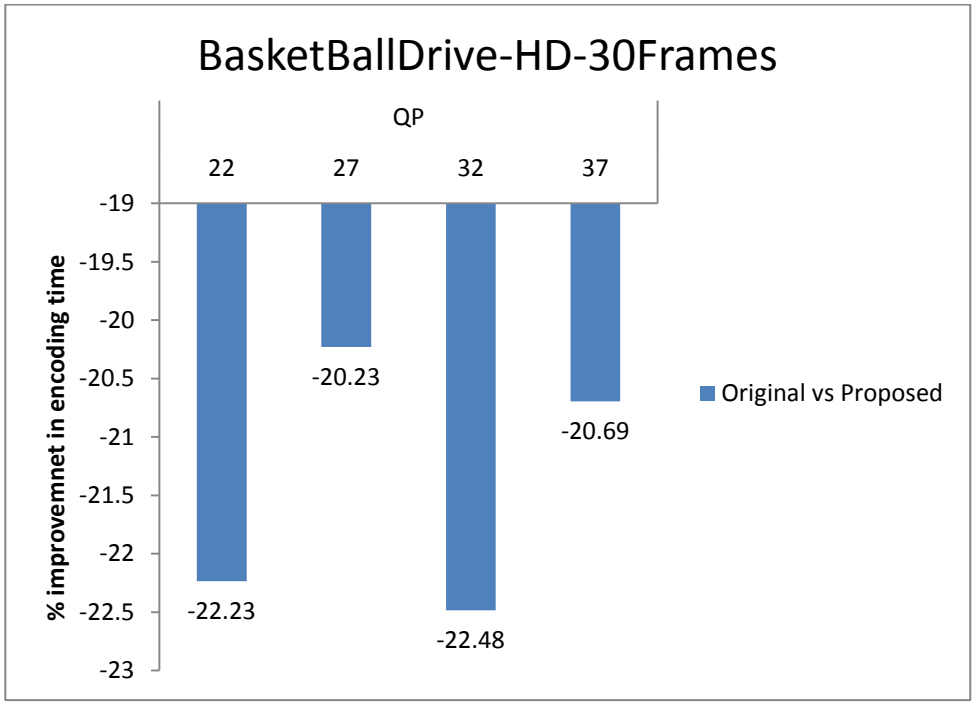

Figure 4-29 % improvement in encoding time vs. quantization parameter for BasketballDrive

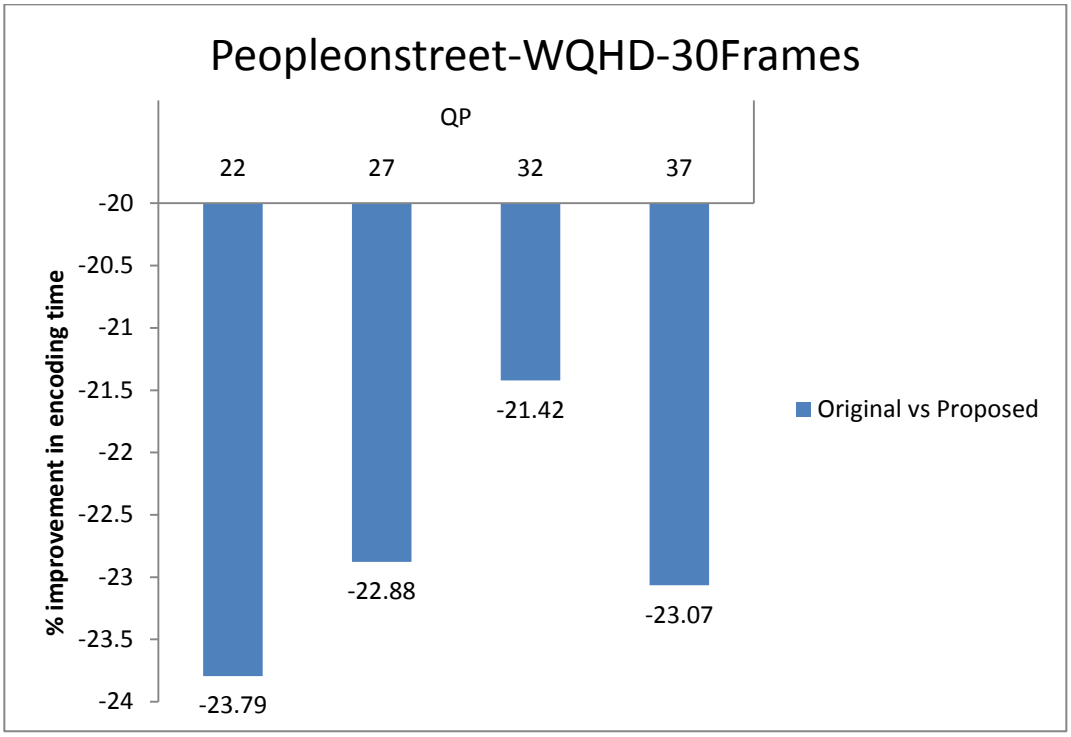

Figure 4-30 % improvement in encoding time vs. quantization parameter for Peopleonstreet

4.8 Summary

In this chapter, various results with graphs are described with and without implementation of the CU splitting algorithm and the TU mode decision using various metrics such as encoding time, BD-PSNR, BD-bitrate, and bitstream size. In chapter 5, conclusions and future work are discussed.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis a CU splitting algorithm and the TU mode decision algorithm are proposed to reduce the computational complexity of the HEVC encoder, which includes two strategies, i.e. CU splitting algorithm and the TU mode decision. The results of comparative experiments demonstrate that the proposed algorithm can effectively reduce the computational complexity (encoding time) by 12-24% on average as compared to the HM 16.0 encoder [38], while only incurring a slight drop in the PSNR and a negligible increase in the bitrate and encoding bitstream size for different values of the quantization parameter based on various standard test sequences [29]. The results of simulation also demonstrate negligible decrease in BD-PSNR [30] i.e. 0.29 dB to 0.51 dB as compared to the original HM16.0 software and negligible increase in the BD-bitrate [31].

5.2 Future Work

There are many other ways to explore in the CU splitting algorithm and the TU mode decision in the intra prediction area as suggested in [25] [33]. Many of these methods can be combined with this method, or if needed, one method may be replaced by a new method and encoding time gains can be explored.

Similar algorithms can be developed for fast inter-prediction in which the RD cost of the different modes in inter-prediction are explored, and depending upon the adaptive threshold [34], mode decision can be terminated resulting in less encoding time and reduced complexity combining with the above proposed algorithm.

Tan et al [37] proposed a fast RQT algorithm for both intra and inter mode coding in order to reduce the encoder complexity. In [37], for all intra case, 13% encoding time can be saved, However, BD-Rate just increases by 0.1%. For random access and low delay constraints it reduces by up to 9% encoding time with 0.3% BD-Rate performance degradation. This method can be integrated with the proposed algorithm to decrease the encoding time.

Another fact of encoding is CU size decisions which are the leaf nodes of the encoding process in the quadtree. Bayesian decision rule can be applied to calculate the CU size and then this information can be combined with the proposed method to achieve further encoding time gains. [24]

Complexity reduction can also be achieved through hardware implementation of a specific algorithm which requires much computation. The FPGA implementation can be useful to evaluate the performance of the system on hardware in terms of power consumption and encoding time.

Appendix A

Test Sequences [29]

Video test sequences used for the implementation of the proposed algorithm vs reference HM 16.0 [38]. Test sequences are arranged in ascending order of their resolution.

A.1 Racehorses

A.2 BasketBallDrillText

A.3 KristenAndSara

A.4 BasketBallDrive


BasketballDrive_1920x1080_50.yuv - YUV Player Deluxe
File Playing Help

A.5 PeopleOnStreet

Appendix B

Test Conditions

The code revision used for this work is revision HM16.0 [38]. The work was done using an Intel Core i5 processor running at 2.50 GHz, with Microsoft Windows 7 64 bit version running with 8 GB of RAM.

Appendix C

BD- PSNR and BD-bitrate [30] [31]

ITU-T standardization sector question regarding BD metrics, Group16,Question 16, Austin, TX, 2-4 April, 2001. Verbatim[31]

Introduction

VCEG-L38 defines "Recommended Simulation Conditions for H.26L". One of the outcomes is supposed to be RD-plots where PSNR and bitrate differences between two simulation conditions may be read. The present document describes a method for calculating the average difference between two such curves. The basic elements are:

Fit a curve through 4 data points (PSNR/bitrate are assumed to be obtained for QP = 16,20,24,28). Based on this, find an expression for the integral of the curve. The average difference is the difference between the integrals divided by the integration interval IPR

“The contributor(s) are not aware of any issued, pending, or planned patents associated with the technical content of this proposal.”

Fitting a curve

A good interpolation curve through 4 data points of a "normal" RD-curve (see figure 1) can be obtained by:

SNR = (a + b*bit + c*bit2)/(bit + d)

where a,b,c,d are determined such that the curve passes through all 4 data points.

This type of curve is well suited to make interpolation in "normal" luma curves. However, the division may cause problems. For certain data (Jani pointed out some typical chroma data) the obtained function may have a singular point in the range of integration - and it fails.

Use of logarithmic scale of bitrate

When we look at figure 1, the difference between the curves is dominated by the high bitrates. The range (1500-2000) gets 4 times the weight of the range (375-500) even if they both represent a bitrate variation of 33%

Hence it was considered to be more appropriate to do the integration based on logarithmic scale of bitrate. Figure 2 shows a plot where "Logarithmic x-axes" is used in the graph function of Excel. However, this function has no flexibility and only allows factors of 10 as units.

In figure 3 I first took the logarithm of bitrates and the plot has units of "dB" along both axes. The factor between two vertical gridlines in the plot is: 100.05 = 1.122 (or 12.2%). Could this be an alternative way of presenting RD-plots?

Interpolation with logarithmic bitrate scale

With logarithmic bitrate scale the interpolation can also be made more straight forward with a third order polynomial of the form:

SNR = a + b*bit + c*bit2 + d*bit3

This result in good fit and there is no problems with singular points. This is therefore the function I have used for the calculations in VCEG-M34. However, for integration of luma curves the results are practically the same as with the first integration method which was used for the software distributed by Michael regarding the complexity experiment.

In the same way we can do the interpolation to find Bit as a function of SNR:

SNR = a + b*SNR + c*SNR2 + d*SNR3

In this way we can find both:

Average PSNR difference in dB over the whole range of bitrates

Average bitrate difference in % over the whole range of PSNR

On request from Michael average differences are found over the whole simulation range (see integration limits in figure 3) as well as in the middle section - called mid range.

As a result VCEG-M34 shows 4 separate data tables.

Conclusions

It is proposed to include this method of finding numerical averages between RD-curves as part of the presentation of results. This is a more compact and in some sense more accurate way to present the data and comes in addition to the RD-plots.

The distinction between "total range" and "mid range" does not seem to add much and it is therefore proposed to use "total range" only.

From the data it is seen that relation between ΔSNR and Δbitrate is well represented by 0.5 dB = 10% or 0.05 dB = 1% It is therefore proposed to calculate either change in bitrate or change in PSNR.

Should it be considered to present RD-plots as indicated in figure 3?

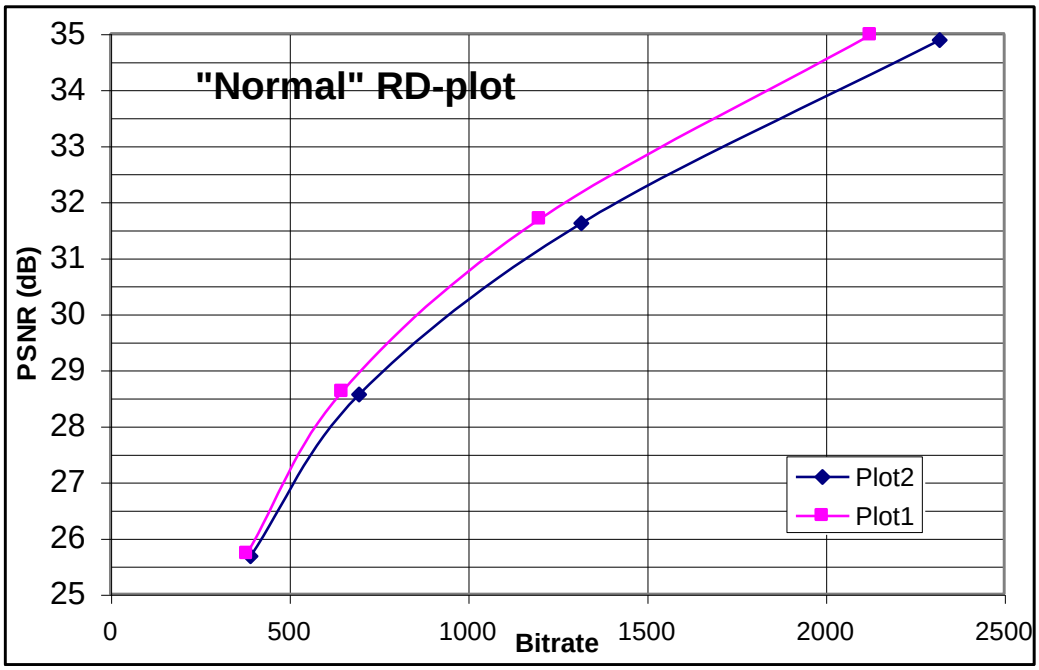

Figure 1

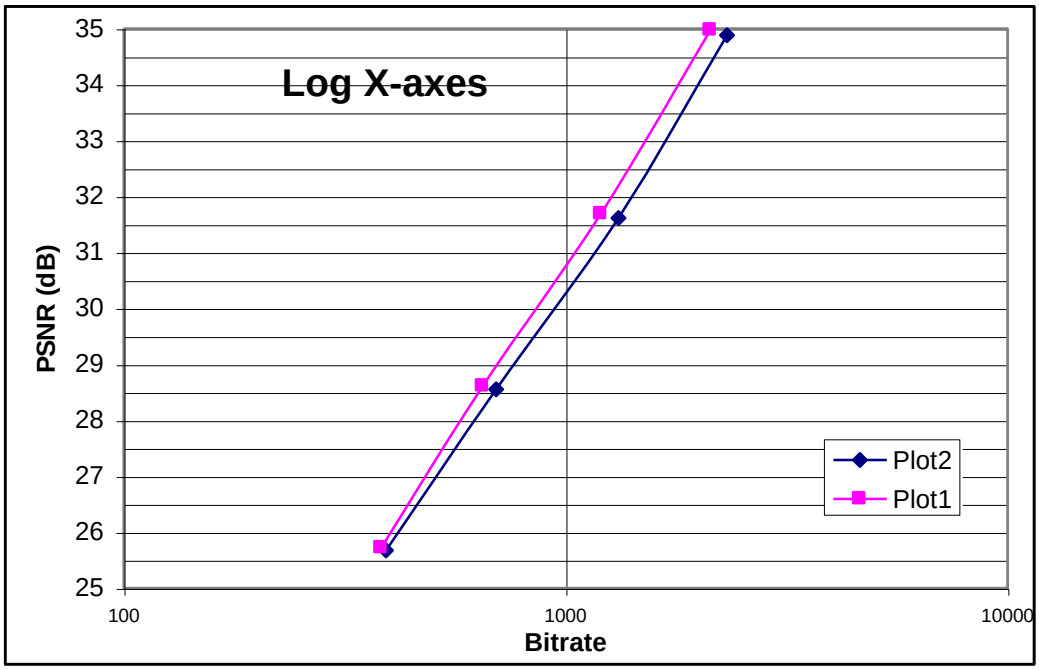

Figure 2

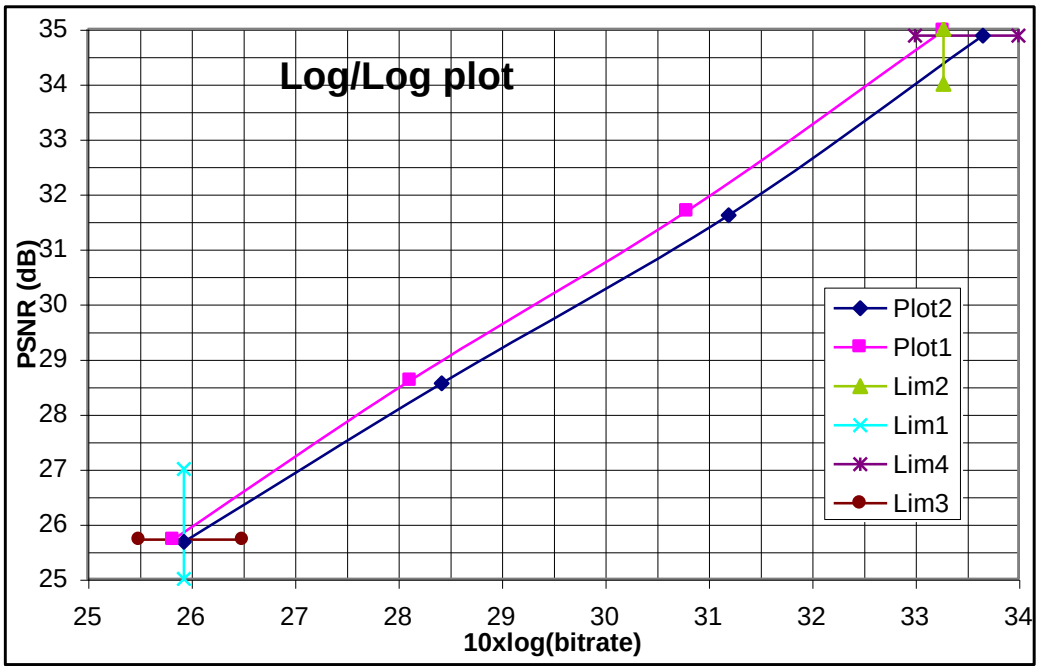

Figure 3

Here is a document about BD-PSNR which has been referenced by many Video Engineers. You can download it at http://wftp3.itu.int/av-arch/video-site/

The matlab code for computing BD-Bitrate and BD-PSNR is found in this link:

http://www.mathworks.com/matlabcentral/fileexchange/27798-bjontegaardmetric/content/bjontegaard.m

function avg_diff = bjontegaard2(R1,PSNR1,R2,PSNR2,mode)

% BJONTEGAARD Bjontegaard metric calculation

% Bjontegaard's metric allows to compute the average gain in PSNR or the

% average per cent saving in bitrate between two rate-distortion

% curves [1].

% Differently from the avsnr software package or VCEG Excel [2] plugin this

% tool enables Bjontegaard's metric computation also with more than 4 RD

% points.

% Fixed integration interval in version 2.

```
%
%  R1,PSNR1 - RD points for curve 1
%  R2,PSNR2 - RD points for curve 2
%  mode -
% 'dsnr' - average PSNR difference
% rate' - percentage of bitrate saving between data set 1 and
% data set 2
%
%  avg_diff - the calculated Bjontegaard metric ('dsnr' or 'rate')
%
%  (c) 2010 Giuseppe Valenzise
%
%% Bugfix 20130515
%  Original script contained error in calculation of integration interval.
%  It was fixed according to description and figure 3 in original
%  publication [1]. Script was verifyed using data presented in [3].
%  Fixed lines labeled as "(fixed 20130515)"
%
%  (c) 2013 Serge Matyunin
%%
%
%  References:
%
%  [1] G. Bjontegaard, Calculation of average PSNR differences between
%      RD-curves (VCEG-M33)
%  [2] S. Pateux, J. Jung, An excel add-in for computing Bjontegaard metric and
%      its evolution
%  [3] VCEG-M34. http://wftp3.itu.int/av-arch/video-site/0104_Aus/VCEG-
```

```
M34.xls

%

% convert rates in logarithmic units


lR1 = log(R1);

lR2 = log(R2);


switch lower(mode)

case 'dsnr'

% PSNR method

p1 = polyfit(lR1,PSNR1,3);

p2 = polyfit(lR2,PSNR2,3);


% integration interval (fixed 20130515) min_int = max([ min(lR1); min(lR2) ]); max_int = min([ max(lR1);
max(lR2) ]);




% find integral

p_int1 = polyint(p1);

p_int2 = polyint(p2);


int1 = polyval(p_int1, max_int) - polyval(p_int1, min_int);

int2 = polyval(p_int2, max_int) - polyval(p_int2, min_int);


% find avg diff

avg_diff = (int2-int1)/(max_int-min_int);


case 'rate'
```

```
% rate method

p1 = polyfit(PSNR1,lR1,3);

p2 = polyfit(PSNR2,lR2,3);


% integration interval (fixed 20130515) min_int = max([ min(PSNR1); min(PSNR2) ]); max_int = min([
max(PSNR1); max(PSNR2) ]);


% find integral

p_int1 = polyint(p1);

p_int2 = polyint(p2);


int1 = polyval(p_int1, max_int) - polyval(p_int1, min_int);

int2 = polyval(p_int2, max_int) - polyval(p_int2, min_int);

end

% find avg diff

avg_exp_diff = (int2-int1)/(max_int-min_int);

avg_diff = (exp(avg_exp_diff)-1)*100;
```

The above proposal is accepted by ITU-T. This work is completely owned by G. Bjontegaard. This metric is used industry wide to gauge compression algorithms from a visual aspect.

Appendix D

Acronyms

API: Application Programming Interface

AVC: Advanced Video Coding

CABAC: Context Adaptive Binary Arithmetic Coding

CB: Coding Block

CPU: Central Processing Unit

CTB: Coding Tree Block

CTU: Coding Tree Unit

CU: Coding Unit

CUDA: Compute Unified Device Architecture

DCC: Data Compression Conference

DCT: Discrete Cosine Transform

DST: Discrete Sine Transform

FDIS: Final Draft International Standard

HEVC: High Efficiency Video Coding

ISO: International Organization for Standardization

ICIP: International Conference on Image Processing

ITU-T: International Telecommunication Union – Telecommunication Standardization Sector

JCT-VC: Joint Collaborative Team on Video Coding

MC: Motion Compensation

MCP: Motion Compensated Prediction

OPENMP: Open Multiprocessing

PB: Prediction Block

PCM: Pulse Code Modulation

PU: Prediction Unit

SAO: Sample Adaptive Offset

SIMD: Single Instruction Multiple Data

SPIE: Society of Photo-Optical and Instrumentation Engineers

TB: Transform Block

VCIP: Visual Communication and Image Processing.

References

[1] G.J. Sullivan et al, “Overview of the high efficiency video coding (HEVC) standard”, IEEE Trans. CSVT, vol. 22, pp.1649-1668, Dec.2012.

[2] C.C.Chi et al, “Parallel scalability and efficiency of HEVC parallelization approaches”, IEEE Trans. CSVT, vol. 22, pp.1827-1838, Dec.2012.

[3] J. Lainema et al,”Intra coding of the HEVC standard”, IEEE Trans. CSVT, vol.22, PP.1792-1801, Dec.2012.

[4] F. Bossen et al, “HEVC complexity and implementation analysis”, IEEE Trans. CSVT, vol. 22, pp.1685-1696, Dec.2012.

[5] P. Hanhart et al, “Subjective quality evaluation of the upcoming HEVC video compression standard” SPIE Applications of digital image processing XXXV, vol.8499, pp.8499-30, Aug.2012.

[6] J.-R Ohm, et al, “Comparison of the coding efficiency of video coding standards-Including high efficiency video coding (HEVC)” , IEEE Trans. CSVT , vol.22, pp.1669-1684, Dec.2012.

[7] X. Zhang, S. Liu and S. Lei, ”Intra mode coding in HEVC standard”, Visual Communications and Image Processing, VCIP 2012, pp. 1-6, San Diego, CA, Nov.2012.

[8] Y.Duan, “An optimized real time multi-thread HEVC decoder”, Visual communications and image processing, VCIP 2012, San Diego, CA, Nov.2012.

[9] G. Correa et al, “Performance and computational complexity assessment of high efficiency video encoders”, IEEE Trans. CSVT, vol.22, pp.1899-1909, Dec.2012.

[10] A.Saxena, F. Fernandes and Y. Reznik, ”Fast transforms for intra-prediction-based image and video coding,” in Proc. IEEE Data Compression Conference (DCC’13), pp.13-22, Snowbird, UT, March 2013.

[11] T.L Silva et al, ”HEVC intra coding acceleration based on tree inter-level mode correlation”, SPA 2013 Poznan, Poland. Sept. 2013.

[12] A. Saxena and F. Fernanades, “Mode dependent DCT/DST for intra prediction in block based image/video coding”, IEEE ICIP, pp. 1685-1688, Sept. 2011

[13] H. Zhang and Z. Ma, ”Fast intra prediction for high efficiency video coding ”, 13th Pacific Rim Conf. on Multimedia, PCM2012, Singapore, vol.7674 LNCS, PP. 568-577, 4-6 Dec. 2012.

[14] M. Zhang, C. Zhao and J. Xu, ”An adaptive fast intra mode decision in HEVC ”, IEEE ICIP 2012, pp.221-224, Orlando, FL, Sept.- Oct.2012.

[15] K. Chen et al, ”Efficient SIMD optimization of HEVC encoder over X86 processors”, APSIPA, pp. 1732-1745, Los Angeles, CA, Dec. 2012.

[16] Y. Kim et al, “A fast intra-prediction method in HEVC using rate-distortion estimation based on Hadamard transform”, ETRI Journal, vol.35, #2, pp.270-280, Apr.2013.

[17] T. Wiegand et al., ”Overview of the H.264/AVC Video Coding Standard”, IEEE Trans.CSVT., vol. 13, no. 7, pp. 560-576, July 2003.

[18] M. Khan et al, “An adaptive complexity reduction scheme with fast prediction unit decision for HEVC Intra encoding”, IEEE ICIP, pp. 1578-1582, Sept. 2013.

[19] S.-W. Teng, H.-M. Hang and Y.-F. Chen. "Fast mode decision algorithm for residual quadtree coding in HEVC." In Visual Communications and Image Processing (VCIP), 2011 IEEE, pp. 1-4, 2011.

[20] S. Vasudevan and K. R. Rao “Combination method of fast HEVC encoding” IEEE ECTICON 2014, Korat, Thailand, May 2014.

[21] H. Li, B. Li and J. Xu, “Rate distortion optimized reference picture management for high efficiency video coding”, IEEE Trans. CSVT, vol. 22, pp.1844-1857, Dec. 2012.

[22] G. Bjontegaard, “Calculation of average PSNR differences between RD-Curves”, ITU-T SG16, Doc. VCEG-M33, 13th VCEG meeting, Austin, TX, April 2001.

[23] G. Bjontegaard, “Improvements of the BD-PSNR model”, ITU-T SG16 Q.6, Doc. VCEG-AI11, Berlin, Germany, July 2008.

[24] Y. Yuan et al, “Quadtree based non-square block structure for interframe coding in high efficiency video coding”, IEEE Trans. CSVT, vol. 22, pp.1707-1719, Dec. 2012.

[25] P. Helle et al, “Block merging for quadtree-based partitioning in HEVC”, IEEE Trans. CSVT, vol. 22, pp.1720-1731, Dec. 2012.

[26] Y. Qin, X. Zhang, S. Wang, and S. Ma. " Early termination of coding unit splitting for HEVC." In Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific, pp. 1-4. IEEE, 2012.

[27] H. Zhang, Q. Liu and Z. Ma, “Priority classification based intra mode decision for high efficiency video coding”, IEEE PCS 2013, pp.285-288, San Jose, CA, Dec. 2013.

[28] I. E. Richardson, “The H.264 advance video compression standard”, 2nd Edition, Wiley, 2010.

[29] D. Marpe, T. Wiegand and G. J. Sullivan, “The H.264/MPEG-4 AVC standard and its applications”, IEEE Communications Magazine, vol. 44, pp. 134-143, Aug. 2006.

[30] HEVC open source software (encoder/decoder) https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.0

[31] Information about quad tree structure of HEVC http://codesequoia.wordpress.com/2012/10/28/hevc-ctu-cu-ctb-cb-pb-and-tb/

[32] Website for downloading test sequence for research purposes http://media.xiph.org/video/derf/

[33] Information on developments in HEVC NGVC- Next generation video coding http://bisqwit.iki.fi/story/howto/openmp/

[34] F. Bossen, D. Flynn and K. Suhring (July 2012), “HEVC reference software manual” online available: http://phenix.int-evry.fr/jct/doc_end_user/documents/6_Torino/wg11/JCTVC-F634-v2.zip

[35] JCT-VC documents are publicly available at http://ftp3.itu.ch/av-arch/jctvc-site and http://phenix.it-sudparis.eu/jct/

[36] Detailed Overview of HEVC/H.265 by Shevach Riabtsev https://app.box.com/s/rxxxzr1a1lnh7709yvih

[37] Access the website http://www.uta.edu/faculty/krrao/dip/Courses/EE5359/

[38] Access to HM 16.0 Software Manual:http://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.0/doc/software-manual.pdf

[39] Video test sequences - http://forum.doom9.org/archive/index.php/t-135034.html or http://media.xiph.org/video/derf/

[40] G. J. Sullivan et al “Standardized Extensions of High Efficiency Video Coding (HEVC)”, IEEE Journal of selected topics in Signal Processing, vol. 7, pp.1001-1016, Dec. 2013.

[41] M. Wien, “High efficiency video coding: Tools and specification”, Springer, 2015.

[42] I.E. Richardson, “Coding video: A practical guide to HEVC and beyond”, Wiley, April 2016.

[43] V.Sze, M.Budagavi and G.J.Sullivan “High Efficiency Video Coding (HEVC) –Algorithms and Architectures”, Springer, 2014.

[44] HEVC tutorial by I.E.G. Richardson: http://www.vcodex.com/h265.html

[45] H.264 tutorial by I.E.G. Richardson: https://www.vcodex.com/h264.html

[46] F. Bossen, ”Excel template for BD-rate calculation based on piece-wise cubic interpolation”, JCT-VC Reflector.

[47] HEVC tutorial by I.E.G. Richardson: http://www.vcodex.com/h265.html

[48] V. Sze and M. Budagavi, “Design and implementation of next generation video coding systems”, Sunday 1 June 2014 (half day tutorial), IEEE ISCAS 2014, Melbourne, Australia, 1-5 June 2014.

[49] J. Chen et al, “Coding tools investigation for next generation video coding based on HEVC”, [9599 – 47], SPIE. Optics + photonics, San Diego, California, USA, 9 – 13, Aug. 2015.

[50] J. Chen et al, “Coding tools investigation for next generation video coding based on HEVC”, [9599 – 47], SPIE. Optics + photonics, San Diego, California, USA, 9 – 13, Aug. 2015.

[51] A. Alshin et al, “Coding efficiency improvements beyond HEVC with known tools,” [9599-48], SPIE. Optics + photonics, San Diego, California, USA, 9 – 13, Aug. 2015

Biographical Information

Nishit Samirbhai Shah was born in Bhavnagar, Gujarat, India in 1991. After completing his schooling at Reliance English Medium School, Vadodara in 2009, he went on to obtain his Bachelors of Technology in Electrical Engineering from Charotar University of Science And Technology, India from 2009-2013.

He joined the University of Texas at Arlington, USA to pursue his M.S in Electrical Engineering in Fall 2013. This was around the time he joined the Multimedia Processing Lab. He worked as LTE Test Engineer(full time – intern) at AT&T Labs, Redmond, WA and continues to work as Sr. Test Engineer since September 2015.