REDUCING THE COMPLEXITY OF INTER-PREDICTION MODE DECISION FOR HIGH EFFICIENCY VIDEO CODEC

by

Kushal 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

May 2014

Acknowledgements

I would like to thank Dr. K. R. Rao for being a supervisor, mentor and inspirational source of encouragement during my thesis. I would like to thank Dr. A. Davis and Dr. W. Dillon for serving on my committee.

I would also like to thank Dr. Budagavi, Senior Researcher, Texas Instruments for promptly replying to my queries.

I would also like to thank my MPL lab-mates: Tuan Ho, Abhishek Hassan Thungaraj, Abhijith Jagannath and Sapna Vasudevan for providing valuable inputs throughout my research.

Last but not least, I would like to thank my family and friends for supporting me in this undertaking

April 21, 2014

Abstract

REDUCING COMPLEXITY OF INTER-PREDICTION

MODE DECISION FOR HIGH EFFICIENCY VIDEO CODEC

Kushal Shah, M.S.

The University of Texas at Arlington, 2014

Supervising Professor: K.R. Rao

The High Efficiency Video Coding (HEVC) standard is the latest joint video project of the International Telecommunication Unit (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). While the HEVC is based on the same architecture of the widely used H.264/AVC (Advance Video Coding) standard [8], it includes many new coding tools, and almost all the encoder blocks are optimized with respect to their counterparts in the H.264/AVC standard. This allows the new standard to achieve up to 50% bitrate reduction compared to its predecessor with the same visual quality at the cost of increased complexity [1].

Like H.264/AVC, mode decisions with Motion Estimation (ME) remain among the most time-consuming computations in HEVC. In an inter-prediction mode decision, a full-search algorithm searches for every possible block size and refines the results from integer-pel to quarter-pel resolution. Thus, a full-search algorithm guarantees the highest level of compression performance. However, the considerable computational complexity for a mode decision decreases the encoding speed.

In this thesis a fast adaptive termination [20] algorithm is proposed that terminates early the mode decision in inter-prediction for HEVC. Based on Rate Distortion (RD) cost, all the inter prediction modes are classified as skip or non-skip modes, and to select the best mode minimum RD cost of these two modes are predicted. For skip mode, the mode decision is predicted in early stage while in non-skip mode different stages are proposed to speed-up the mode decision. Experimental results based on several video test sequences suggest a decrease of about 25%-40% in encoding time is achieved with implementation of the Fast Adaptive Termination algorithm for inter-prediction mode decision 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), SSIM (Structural Similarity) and computational complexity are also used.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

1.1 Video Compression Basics

Today the world has transformed into the so called “digital age” or “electronic age”, where mobile phones are called smart phones because they can not only make phone calls but are also used for web browsing, sending emails, watching / capturing videos, transfer data, navigation purposes to make parallel structures and as camera. Digital television sets have become more compact with availability of regional and international channels with HD (High Defination) quality. Data is stored on re-writable DVDs, Blu-ray discs and hard disks which are light weight, portable and have huge data storage. Internet connection is fast with wireless routers and modems operating at faster speeds. In this fast growing world of communications, data compression is still one of the most essential components in any multimedia system. Modern data compression techniques offer the possibility to store or transmit the vast amounts of data necessary to represent digital videos and images in an efficient and robust way [15].

Uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia-based web applications have not only sustained the need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology. Image compression minimizes the size in bytes of a graphics file without degrading the quality of the image to an unacceptable level. The reduction in the size allows more images to be stored in a given amount of disk or memory space. It also reduces the time required for images to be sent over the Internet or downloaded from web pages [15].

1.2 Need for Video Compression

Compression is useful because it helps reduce resource usage, such as data storage space or transmission capacity. Because compressed data must be decompressed to use, this extra processing imposes computational or other costs through decompression; this situation is far from being a free lunch. Data compression is subject to a space–time complexity trade-off. For instance, a compression scheme for video may require expensive hardware for the video to be decompressed fast enough to be viewed as it is being decompressed, and the option to decompress the video in full before watching it may be inconvenient or require additional storage. The design of data compression schemes involves trade-offs among various factors, including the degree of compression, the amount of distortion introduced (e.g., when using lossy data compression), and the computational resources required to compress and uncompress the data.

The high bit rates that result from the various types of digital video make their transmission through their intended channels very difficult. Even entertainment video with modest frame rates and dimensions would require bandwidth and storage space far in excess of that available from a CD-ROM. Thus delivering consumer quality video on a compact disk would be impossible. This is analagous to an envelope being too large to fit into a letterbox. Similarly the data transfer rate required by a video telephony system is far greater than the bandwidth available over the plain old telephone system (POTS). Even if high bandwidth technology (e.g. fibre-optic cable) was in place, the per-byte-cost of transmission would have to be very low before it would be feasible to use it for the staggering amounts of data required by HDTV and ultra HDTV. Finally, even if the storage and transportation problems of digital video were overcome, the processing power needed to manage such volumes of data would make the receiver hardware very expensive.

1.3 Video Compression Standards

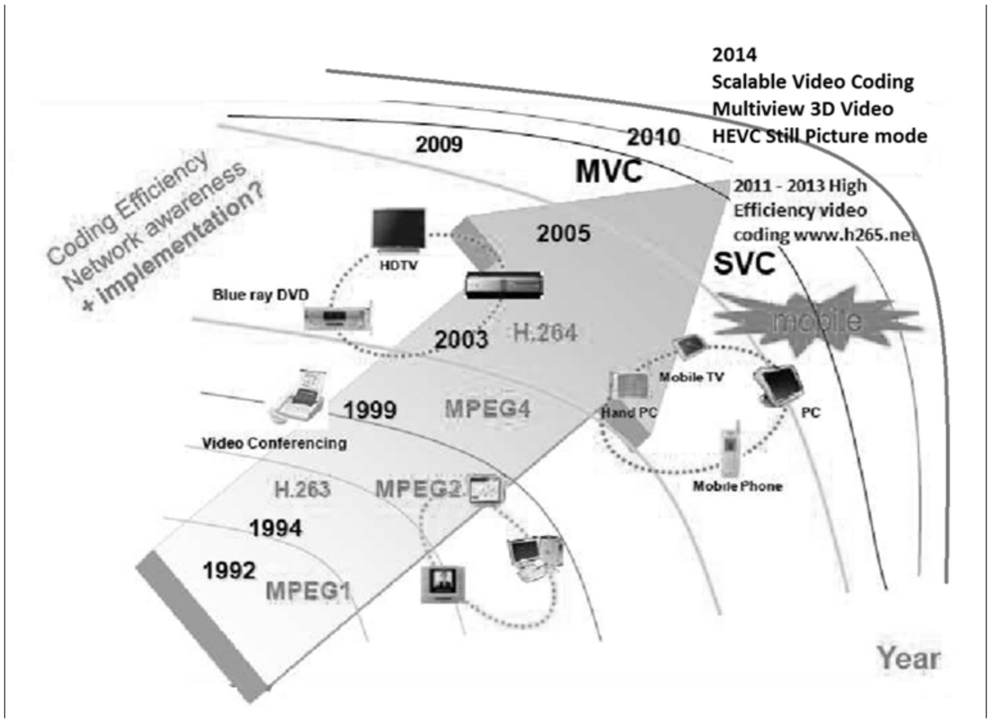


Figure 1-1 Evolution of video codec standards [9]

Video coding standards have evolved primarily through the development of the well-known ITU-T and ISO/IEC standards. The ITU-T produced H.261 [2] and H.263 [3], ISO/IEC produced MPEG-1 [4] and MPEG-4 Visual [5], and the two organizations jointly produced the H.262/MPEG-2 Video [6] and H.264/MPEG-4 advanced video coding (AVC) [7] standards as shown in figure 1-1 [9]. The two standards that were jointly produced have had a particularly strong impact and have found their way into a wide variety of products that are increasingly prevalent in our daily lives. Throughout this evolution, continued efforts have been made to maximize compression capability and improve other characteristics such as data loss robustness, while considering the computational resources that were practical for use in products at the time of anticipated deployment of each standard. The major video coding standard directly preceding the HEVC project was H.264/MPEG-4 AVC, which was initially developed in the period between 1999 and 2003, and then was extended in several important ways from 2003–2009. H.264/MPEG-4 AVC has been an enabling technology for digital video in almost every area that was not previously covered by H.262/MPEG-2 Video and has substantially displaced the older standard within its existing application domains. It is 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-ray disks, and real-time conversational applications such as video chat, video conferencing, and telepresence systems [11]. 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) 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.

1.4 Thesis Outline

Chapter 2 provides the introduction to video quality and formats along with a high level description of HEVC Standard. Key features in the encoding process are also described in brief. Chapter 3 discusses present inter-prediction technique and various complexity reduction algorithms already present for mode decision along with the proposed implementation method for reducing complexity using fast adaptive termination algorithm for HEVC. Chapter 4 discusses the experiments and the results. Here the analysis of the results and a comparative analysis between different algorithms are provided. Chapter 6 outlines the conclusions and further research. The configuration files used by the HM 13.0 [26] software of HEVC encoder for the generation of the bit streams are also provided.

Chapter 2

High Efficiency Video Coding

The HEVC is latest video standard introduced by the Joint Collaborative Team on Video Coding (JCT-VC) in January, 2013 [7-8] which contains three profiles main (8-bit), main10 (10-bit) and still frame. Here only the main (8-bit) profile is considered since it is most widely used profile. The HEVC standard is designed to achieve multiple goals, including coding efficiency, ease of transport system integration, data loss resilience and implementation using parallel processing architectures. The HEVC standard has been designed to address essentially all this existing applications of the H.264/MPEG-4 AVC standard [14] and to particularly focus on two key issues: increased video resolution and increased use of parallel processing architectures [1]. The major achievements of the HEVC standard in comparison with the H.264 [14] standard are flexible prediction modes, transform block sizes, better partitioning options, improved interpolation and deblocking filters, prediction, signaling of modes and motion vectors and support efficient parallel processing [1]. HEVC has been designed to address essentially all existing applications of H.264/MPEG-4 AVC and to particularly focus on two key issues: increased video resolution and increased use of parallel processing architectures. The HEVC syntax should be generally suited for other applications and not specifically to two applications mentioned above [1]. This is not the result of optimizing a single step in the encoding process, but a combined result of optimization of many processes together.

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]. As more and more emphasis is laid on video streaming and playback of HD and beyond HD quality, the HEVC standard is a great improvement with respect to the previous standards. The basic design of the HEVC standard remained the same as that of the H.264/AVC i.e., the block based hybrid coding approach which efficiently exploits the temporal statistical dependencies and the spatial statistical dependencies [1].

The block diagrams of the HEVC encoder and decoder are shown in figures 2-1 and 2-2 respectively.

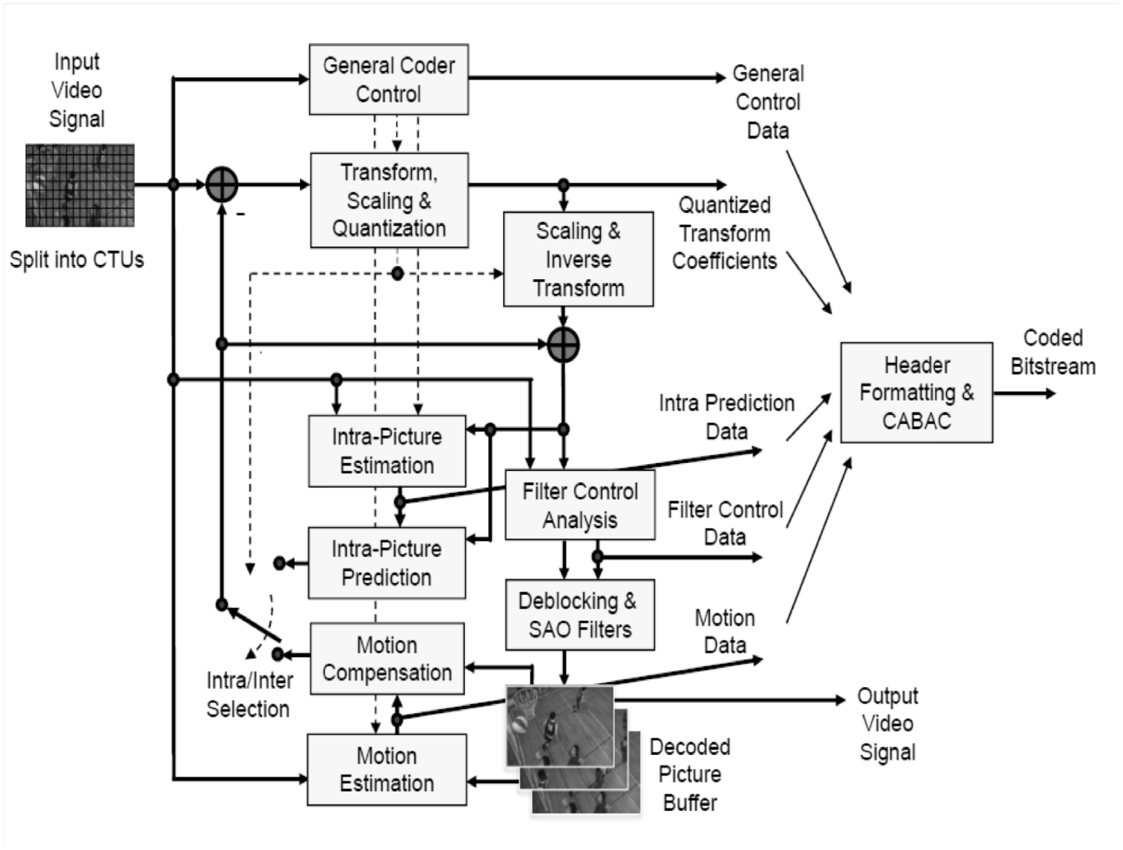


Figure 2-1 HEVC encoder block diagram [1]

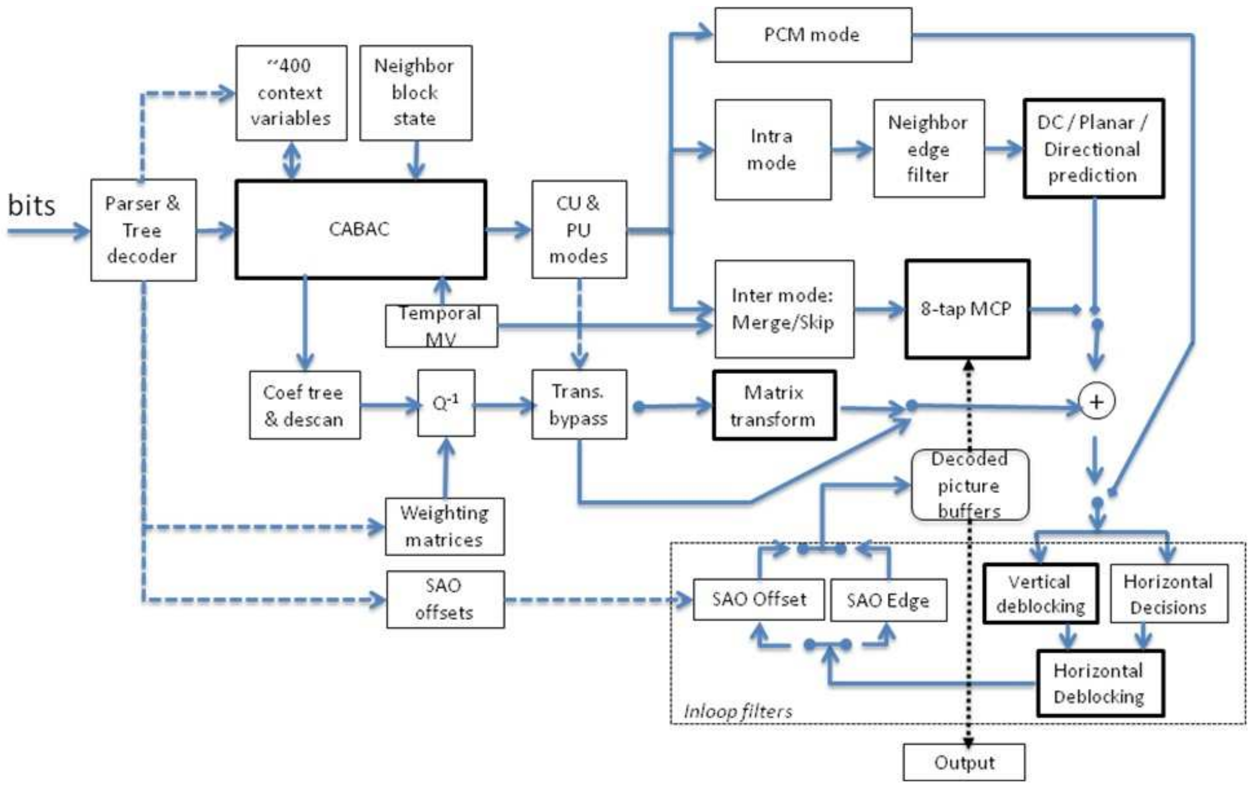


Figure 2-2 HEVC decoder block diagram [10]

2.1 Video Coding Techniques Description for HEVC

The video coding layer of the HEVC standard employs the same hybrid approach (inter/intrapicture prediction and 2-D transform coding) used in all video compression standards since H.261 [1]. The HEVC standard employs adaptive and flexible quad-tree coding block partitioning structure which enables efficient use of large multiple sizes of prediction, coding, transform block ,employs improved intra prediction, adaptive motion parameter prediction, new loop filter and an enhanced context-adaptive binary arithmetic coding (CABAC) as entropy coding method [11].

2.1.1 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].

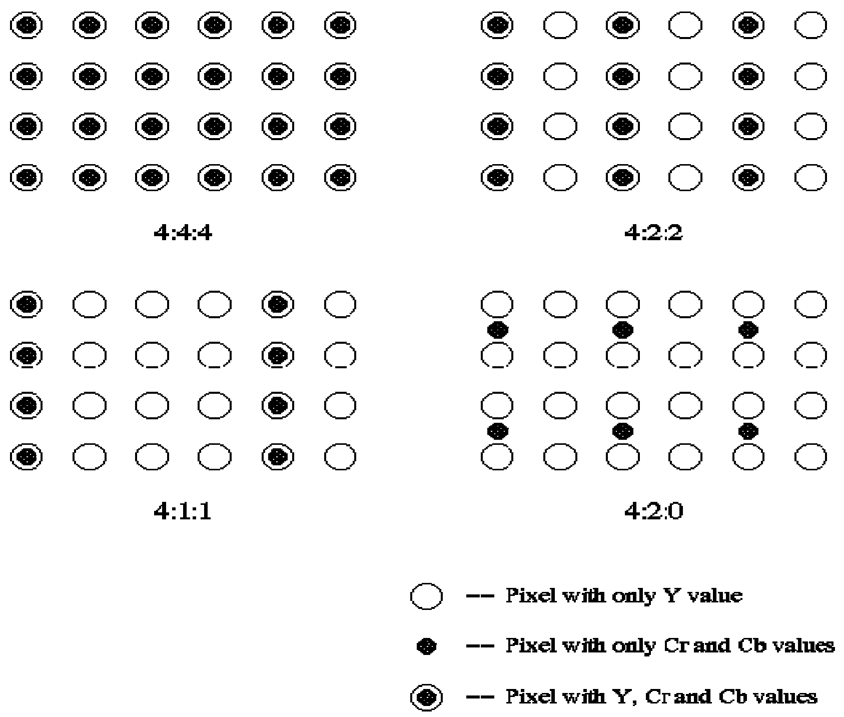

4:4:4
4:2:2
4:1:1
4:2:0
-- Pixel with only Y value
-- Pixel with only Cr and Cb values
-- Pixel with Y, Cr and Cb values

Figure 2-3 Format for YUV components [15]

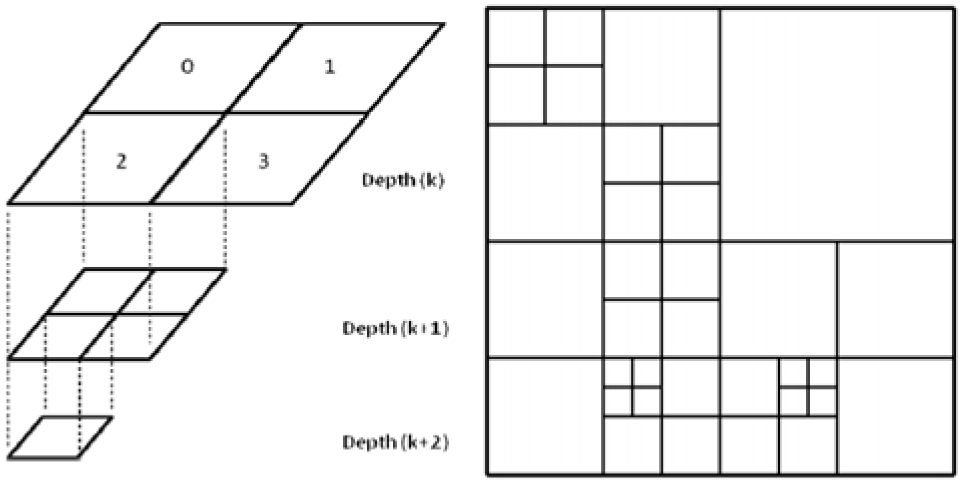


Figure 2-4 Quad tree CU structure in HEVC [16] [1]

The Coding Tree Unit (CTU) consists of a Luma Coding Tree Block (CTB) and the corresponding chroma CTBs and syntax elements. The CTU specifies the positions and the sizes of the luma Coding Block (CB) and chroma CB. One luma CB and generally two chroma CBs together with syntax form a Coding Unit (CU). A CTB can have one CU or be split into several CUs. The decision to code an area of image as intra or inter is taken at the CU level. A CU is the root for both the Prediction Unit (PU) and the Transform Unit (TU). A Prediction Block (PB) can be the size of a CB or be split further into smaller luma and chroma PBs. The supported sizes are 64x64, 32x32, 16x16, 8x8 and 4x4. For inter prediction modes, non-square modes are allowed as shown in figure 2-5. An inter frame PB cannot have a size of 4x4.

Similarly, starting at the level of a CU, a CB can have one Transform Block (TB) of the same size as the CB or be split into smaller TBs [1] [17] [18] as shown in figures 2-6 and 2-7.

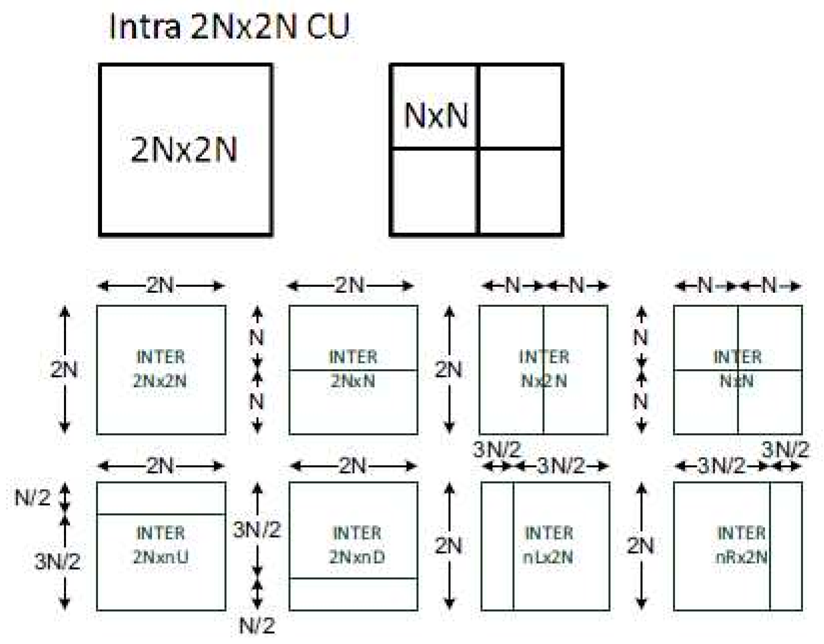


Figure 2-5 Intra and Inter frame prediction modes for HEVC[16]

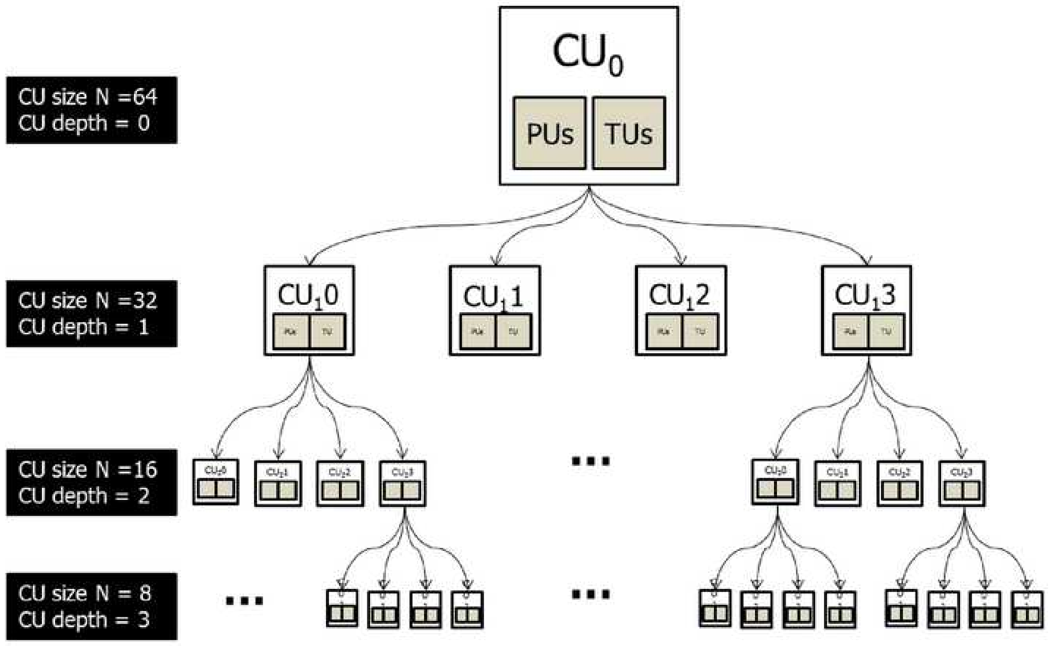


Figure 2-6 Splitting Coding unit into prediction units and transform units [19]

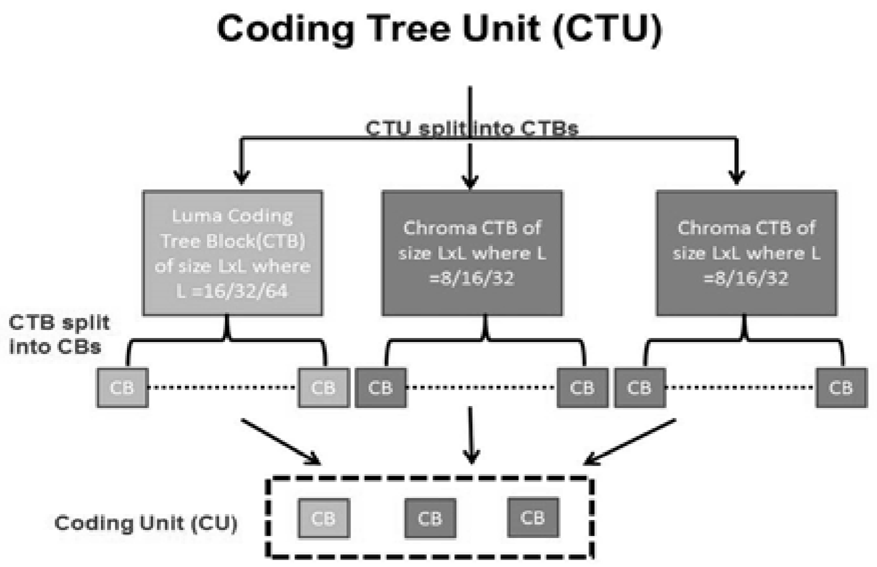

Coding Tree Unit (CTU)
CTU split into CTBs
Luma Coding Tree Block(CTB) of size LxL where L =16/32/64
Chroma CTB of size LxL where L =8/16/32
Chroma CTB of size LxL where L =8/16/32
CTB split into CBs
CB
Coding Unit (CU)

Figure 2-7 Splitting Coding tree unit into Coding Blocks [1]

2.1.2 Slice and Tiles

Slices are processed in the order of a raster scan. A picture may be split into one or several slices as shown in figure 2-8 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 bitstream 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 2-8.

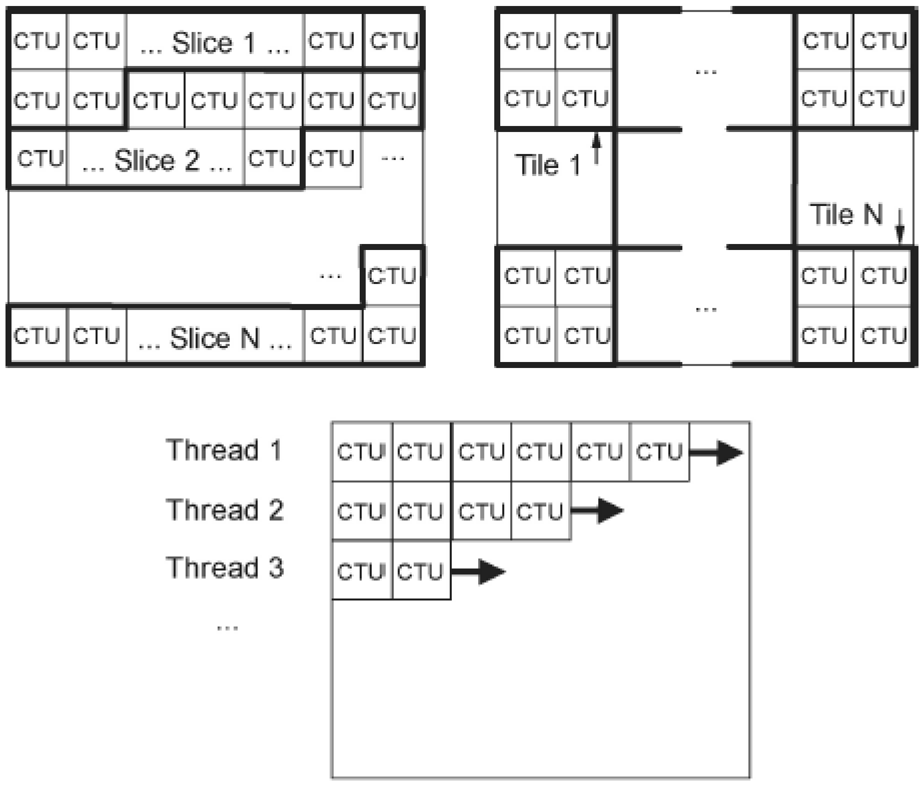


Figure 2-8 Subdivision of picture into slice and Tiles[1]

2.2 HEVC Encoder Description

There are five major parts of the HEVC encoder which are discussed in the following section.

2.2.1 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.

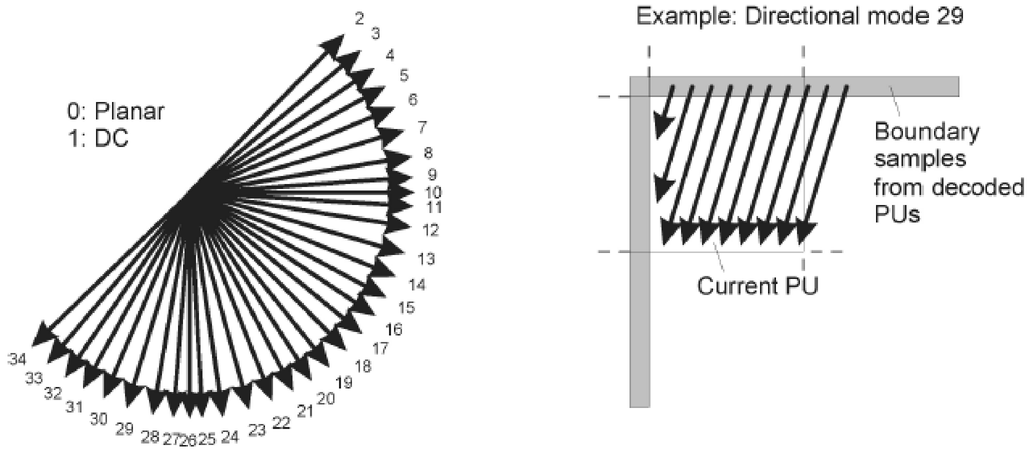


Figure 2-9 Mode decision for intra picture prediction [1]

2.2.2 Inter-picture Prediction

Compared to intrapicture-predicted CBs, the HEVC standard supports more PB partition shapes for interpicture-predicted CBs. The partitioning modes of PART_2N×2N, PART_2N×N, and PART_N×2N as shown in figure 3-5 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.

2.2.3 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 since the rows of the transform matrix are close approximations of values of uniformly scaled basis functions of orthonormal DCT [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.

2.2.4 Entropy Coding

A new and improved CABAC (Context Adaptive Binary Arithmetic Coding) is used for the entropy coding of the bitstreams. This coding has improved speed, compression and requires less memory then entropy coding used in the H.264/AVC standard. Instead of doing the normal CABAC re-initialization for every CTB row, the context state from the second CTU in the previous row is used to start the processing of a brand new CTB row (figure 2-10), and thus taking huge advantage of parallel processing.

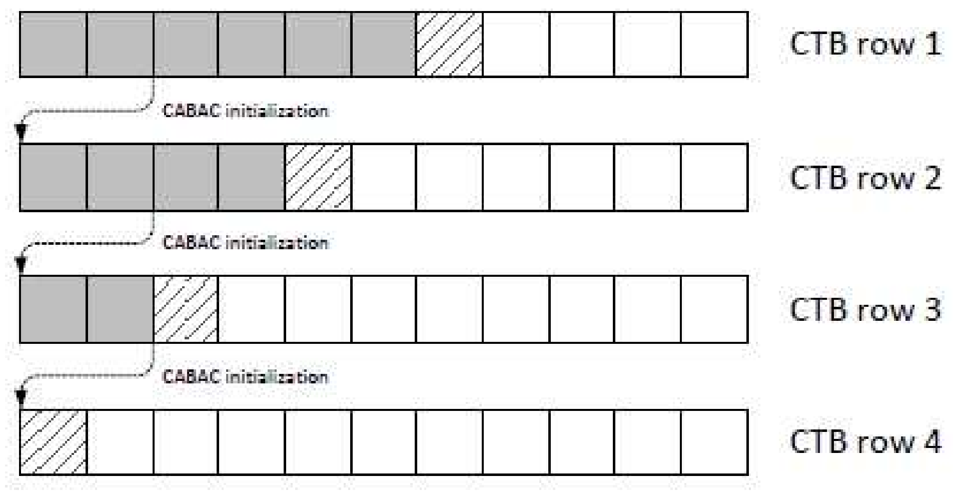


Figure 2-10 Example of waveform processing [13]

2.2.5 In-loop Deblocking Filtering

A deblocking filter is applied to the adjacent side of the transform unit and the prediction unit separately since the prediction unit boundaries are not always aligned with the transform unit for HEVC. The main function of a deblocking filter is to remove the high frequency artifacts from the edge of PU and TU. The design of deblocking filter is simplified with regard to its decision-making and filtering processes, and it is made for parallel processing. However, it relies on the same principles as the H.264/AVC deblocking filter [14]. Number of filter samples is reduced by half in the HEVC standard since filter is applied to 8x8 blocks as compared to 4x4 blocks in H.264 [1].

2.3 Summary

This chapter outlines the coding tools of the HEVC codec. The intent of the HEVC project 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 inter-prediction mode decision and fast adaptive early termination algorithm.

Chapter 3

Inter-prediction

3.1 Inter-prediction Introduction

The HEVC standard defines the coding unit (CU) as the basic processing unit instead of the macroblock (MB). Unlike a MB whose size is fixed at 16×16 pixels, the size of a CU is not fixed, varying from 8×8 to 64×64. A large CU reduces the motion information data. Thus, the compression efficiency is improved in a lossless manner, especially for high-resolution videos. A CU can be partitioned into smaller CUs and the structure among different CUs is represented by a quad-tree. The depth of this tree can be as large as four. The largest CU in depth 0 is denoted as LCU. For a CU whose size is denoted by 2N×2N, predictions are performed for various block sizes of 2N×2N, 2N×N, N×2N and N×N as shown in figure 2-5 from chapter 2. The processing unit for prediction is called the prediction unit (PU) [27].

3.1.1 Motion Vector Prediction

Like the AVC, the HEVC standard has two reference lists: L0 and L1. They can hold 16 references each, but the maximum total number of unique pictures is 8. This means that to find a maximum output, the same picture has to be added more than once. The encoder may choose to do this to be able to predict the same picture with different weights (weighted prediction). The HEVC standard uses more complex motion prediction than AVC. The HEVC standard uses candidate list indexing. There are two MV prediction modes: merge and AMVP (advanced motion vector prediction). The encoder decides between these two modes for each PU and signals it in the bitstream with a flag. Only the AMVP process can result in any desired MV, since it is the only one that codes an MV delta. Each mode builds a list of candidate MVs, and then selects one of them using an index coded in the bitstream.

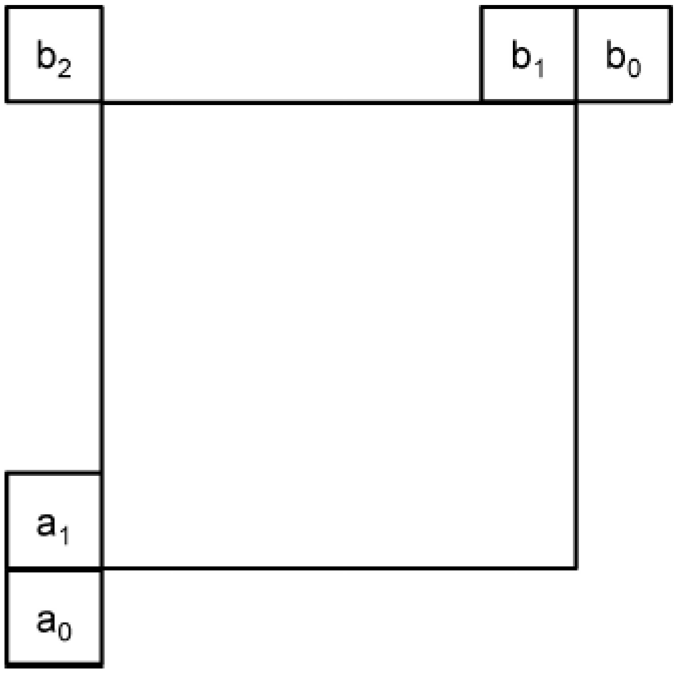


Figure 3-1 Position of spatial candidates of motion information [1]

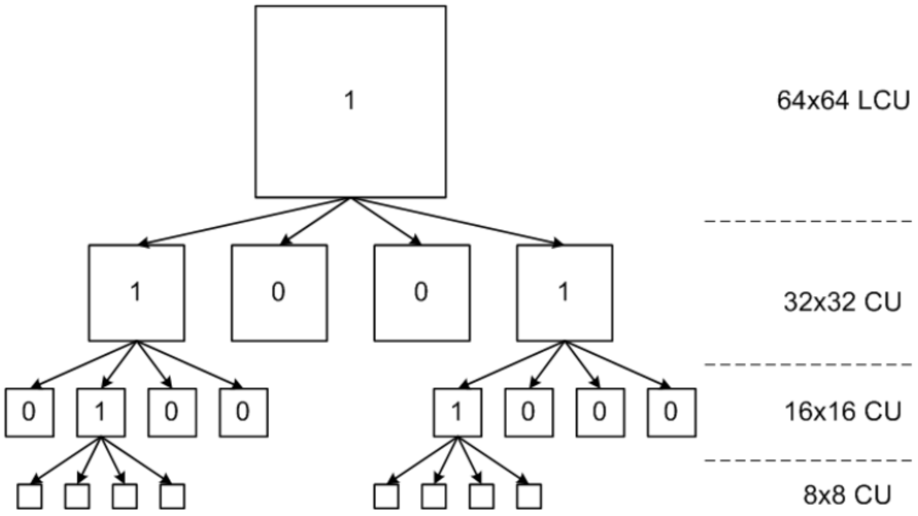


Figure 3-2 Quad-tree splitting flag. 1- Level 1 (L1), 0 – Level (L0) [42]

3.1.2 Advanced Motion Vector Prediction Process

AMVP process is performed once for each MV; so once per L0 or L1 PU for unidirectional PU, or twice for a bidirectional PU as shown in figure 3-2. The bitstream specifies the reference picture to use for each MV. A two-deep candidate list is formed:

First, a left predictor is obtained. a_0 is preferred over a_1, same list is preferred over the opposite list, and neighbor is preferred that points to the same picture over one that does not. If no neighbor points to the same picture, the motion vector is scaled to match the picture distance (similar process as the AVC temporal direct mode). If all this results are in a valid candidate, a motion vector is added to the candidate list.

Second, upper predictor is obtained. b_0 is preferred over b_1, b_1 is preferred over b_2 and the neighbor MV that points to the same picture is preferred for motion vector prediction over the one that are not in same picture. Neighbor scaling for the upper predictor is only done if it was not done for the left neighbor, ensuring no more than one scaling operation per PU. If the candidate is found, it is added to the list. If the list still contains less than two candidates, a temporal candidate (scaled MV according to picture distance) is obtained, which is co-located with the right bottom of the PU. If the candidate lies outside the CTB row, or outside the picture, or if the co-located PU is intra, center position is tried again. If temporal candidate is found, it is added to the list. If the candidate list is still empty, a (0,0) motion vector is added until full. Finally, with the transmitted index, right candidate is selected and is added in the transmitted MV delta.

3.1.3 Merge Mode

The merge process results in a candidate list of up to five entries deep, configured in the slice header. Each entry ends up being L0, L1 or bidirectional. Four spatial candidates are added in this order: a_1, b_1, b_0, a_0, b_2. A candidate is not added to the list if it is the same as one of the earlier candidates. Then, if the list still has room, a temporal candidate is added, which is found by the same process as in AMVP (Advance Motion Vector Prediction) process. Then, if the list still has room, bidirectional candidates are added and formed by making combinations of the L0 and L1 vectors of the other candidates already in the list. Then finally if the list still is not full, (0,0) MVs are added with increasing reference indices. The final motion vector is obtained by picking one of the up-to-five candidates as signaled in the bitstream.

The HEVC sub-samples the temporal motion vectors on a 16x16 grid. Thus, decoder only needs to make room for two motion vectors (L0 and L1) per 16x16 regions in the picture when it allocates the temporal motion vector buffer. When the decoder calculates the co-located position, lower 4 bits are zeroed out of the x/y position, snapping the location to a multiple of 16 and picture is considered to be co-located, that is signaled in the slice header.

3.1.4 Motion Compensation

Like MPEG-4/AVC [1], HEVC specifies motion vectors in 1/4-pel, but uses an 8-tap filter for luma (all positions), and a 4-tap 1/8-pel filter for chroma as shown in figure 3-2. Because of the 8-tap filter, any given NxM sized block requires extra pixels on all sides (3 left/above, 4 right and below) to provide the filter with the data it needs. With small blocks like an 8x4, (8+7)x(4+7) = 15x11 pixels are needed. The HEVC standard limits the smallest block to be uni-directional and 4x4 is not supported since more small blocks require more memory read, thus increasing more memory access, more time and more power.

The HEVC standard also supports weighted prediction for both uni- and bi-directional PUs. However, the weights are always explicitly transmitted in the slice header, there is no implicit weighted prediction like in MPEG-4/AVC [1]. Quarter-sample precision is used for the motion vectors. 7-tap (weights: -1, 4, -10, 58, 17, -5, 1) or 8-tap (weights: -1, 4, -11, 40, 40, -11, 4, 1) filters are used for interpolation of fractional-sample positions as shown in figure 3-3. Similar to H.264/MPEG-4 AVC [1], multiple reference pictures are used as shown in figure 3-4. For each PB, either one or two motion vectors can be transmitted, resulting either in unipredictive or bipredictive coding, respectively. A scaling and offset operation can be applied to the prediction signal/signals in a manner known as weighted prediction.

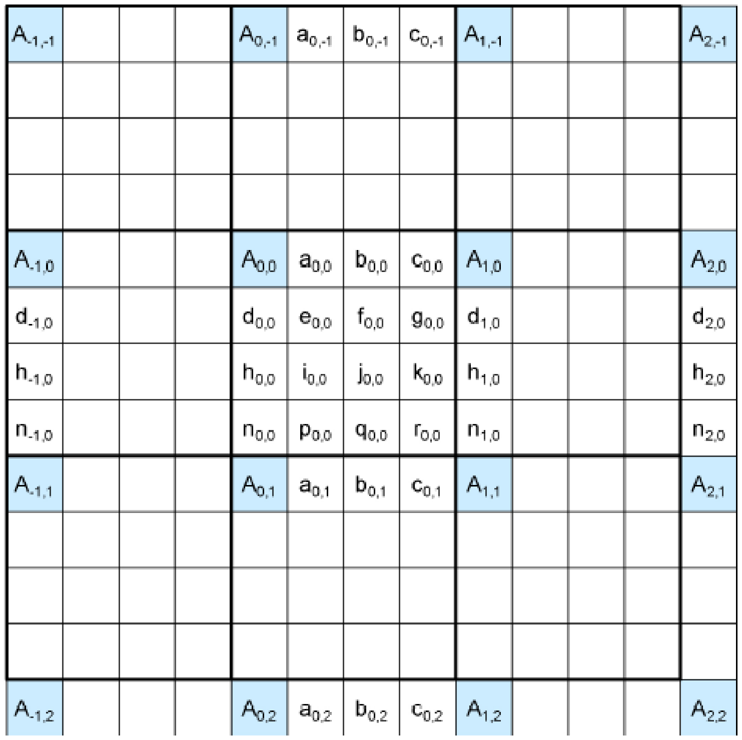


Figure 3-3 Integer and fractional sample position for luma interpolation [1]

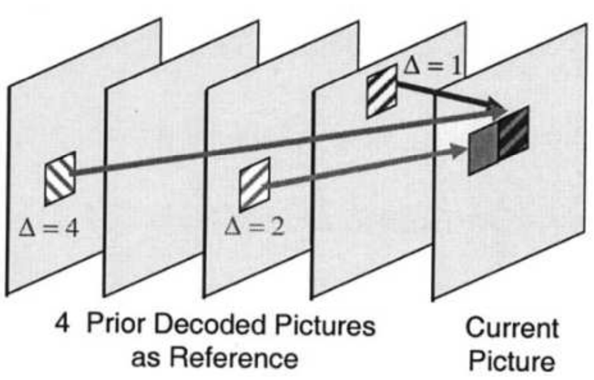


Figure 3-4 Multiple pictures used as reference for the current picture for motion compensation [8]

Figure 3-3 shows the positions labeled with upper-case letters, $A_{i,j}$, representing the available luma samples at integer sample locations, whereas the other positions labeled with lower-case letters represent samples at non-integer sample locations, which need to be generated by interpolation. The samples labeled $a_{0,j}$, $b_{0,j}$, $c_{0,j}$, $d_{0,0}$, $h_{0,0}$, and $n_{0,0}$ are derived from the samples $A_{i,j}$ by applying the eight-tap filter for half-sample positions and the seven-tap filter for the quarter-sample positions as follows [1]:

$$\begin{aligned}
a_{0,j} &= (\textstyle\sum_{i=-3..3} A_{i,j}\,\text{qfilter}[i]) >> (B-8)\\
b_{0,j} &= (\textstyle\sum_{i=-3..4} A_{i,j}\,\text{hfilter}[i]) >> (B-8)\\
c_{0,j} &= (\textstyle\sum_{i=-2..4} A_{i,j}\,\text{qfilter}[1-i]) >> (B-8)\\
d_{0,0} &= (\textstyle\sum_{i=-3..3} A_{0,j}\,\text{qfilter}[j]) >> (B-8)\\
h_{0,0} &= (\textstyle\sum_{i=-3..4} A_{0,j}\,\text{hfilter}[j]) >> (B-8)\\
n_{0,0} &= (\textstyle\sum_{j=-2..4} A_{0,j}\,\text{qfilter}[1-j]) >> (B-8)
\end{aligned}$$

where the constant B ≥ 8 is the bit depth of the reference samples (typically B = 8 for most applications). In these formulas, the symbol, >>, denotes an arithmetic right shift operation. The samples labeled $e_{0,0}$, $f_{0,0}$, $g_{0,0}$, $i_{0,0}$, $j_{0,0}$, $k_{0,0}$, $p_{0,0}$, $q_{0,0}$, and $r_{0,0}$ can be derived by applying the corresponding filters to samples located at vertically adjacent $a_{0,j}$, $b_{0,j}$ and $c_{0,j}$ positions as follows [1]:

$$e_{0,0} = (\textstyle\sum_{v=-3..3} a_{0,v}\ \text{qfilter}[v]) >> 6$$
$$f_{0,0} = (\textstyle\sum_{v=-3..3} b_{0,v}\ \text{qfilter}[v]) >> 6$$
$$g_{0,0} = (\textstyle\sum_{v=-3..3} c_{0,v}\ \text{qfilter}[v]) >> 6$$
$$i_{0,0} = (\textstyle\sum_{v=-3..4} a_{0,v}\ \text{hfilter}[v]) >> 6$$
$$j_{0,0} = (\textstyle\sum_{v=-3..4} b_{0,v}\ \text{hfilter}[v]) >> 6$$
$$k_{0,0} = (\textstyle\sum_{v=-3..4} c_{0,v}\ \text{hfilter}[v]) >> 6$$
$$p_{0,0} = (\textstyle\sum_{v=-2..4} a_{0,v}\ \text{qfilter}[1-v]) >> 6$$
$$q_{0,0} = (\textstyle\sum_{v=-2..4} b_{0,v}\ \text{qfilter}[1-v]) >> 6$$
$$r_{0,0} = (\textstyle\sum_{v=-2..4} c_{0,v}\ \text{qfilter}[1-v]) >> 6.$$

3.2 Inter-prediction Mode Decision

Figure 3-3 depicts partition modes in HEVC inter-prediction and figures 3-4 and 3-5 depict mode decision process of the HEVC encoder. This process is recursively implemented for each inter-coded CU at every quadtree depth h after which a final assignment of coding modes is accomplished at CTU level.

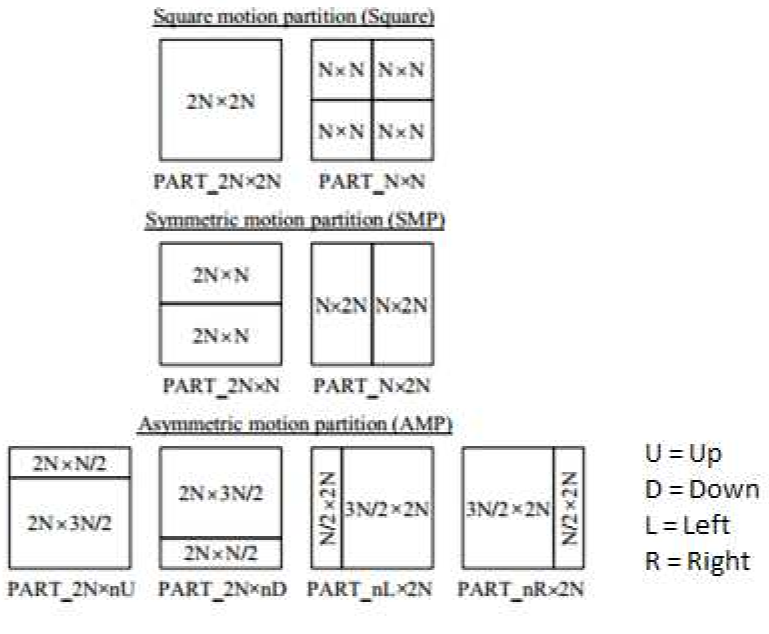


Figure 3-5 Partition mode in HEVC inter-prediction [24]

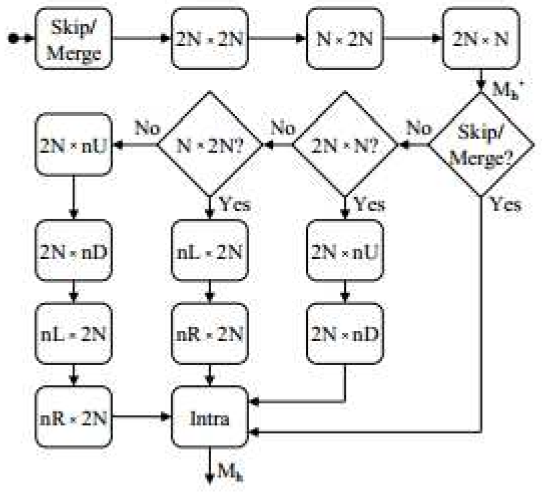


Figure 3-6 Mode decision in HM software for N ∈ {8, 16} [27]

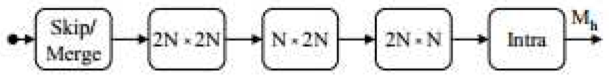


Figure 3-7 Mode decision in HM software for N ∈ {4, 32} [27]

Consider an inter-coded CU at depth h, then the HEVC encoder evaluates the Skip, Merge, Square, and SMP modes for CU of size 2N × 2N and selects the best interim mode (M_h') among them. If N ∈ {8, 16}, M_h' is used to assign a mode set for the subsequent AMP mode decision. The mode set contains none of the AMP modes if M_h' ∈ {Skip, Merge}, half of the AMP modes if M_h' ∈ {PART_2N×N, PART_N×2N}, and all AMP modes if M_h' ∈ PART_2N×2N. After the AMP modes, the encoder evaluates the Intra mode and resolves the best mode (M_h).

The HEVC encoder takes advantage of a low-complexity merge mode only testing in the AMP mode evaluation if M_h' and the coding mode of the parent CU (M_h-1) meets the conditions [23]. With N = 32, the encoder restricts the assessment of the AMP modes to the merge mode, so its flowchart for N = 32 converges to that for N = 4 as shown in figure 3-4. The HEVC encoder also offers three optional early termination mechanisms to speed-up its mode decision: early CU (ECU) [29], early skip detection (ESD) [30], and CBF fast mode (CFM) [31], where CBF denotes a coded block flag. The ECU monitors M_h at the end of each quadtree depth h and terminates the mode decision process for the further depths ($h + 1 \dots h_{max}$) if M_h = Skip. When the ESD is enabled for early mode decision, the evaluation of PART_2N×2N is conducted before the Skip/Merge modes in order to discover its motion information and the CBFs at the earliest possible stage of each quadtree depth h. If PART_2N×2N at depth h contains only luma/chroma PBs whose CBF = 0 and no supplementary motion information, ESD sets M_h = Skip and terminates the remaining mode evaluations at depths h … hmax. CFM monitors CBFs of the Square, SMP, and AMP modes at each quadtree depth h. If an evaluated mode at depth h contains only PBs having CBF = 0, it is selected as M_h without evaluating the remaining modes at that depth and the mode decision proceeds to depth h + 1.

3.3 Inter-prediction PU Mode Decision Complexity

The inherent data structures defined in the HEVC standard can be highlighted as one of the main causes of its high complexity [33]. Frames are now divided into treeblocks, which can be subdivided into Coding Units (CU). Furthermore, CUs can be recursively partitioned, forming a quad-tree structure, which is illustrated in figure 3-6. The CU quad-tree decision is originally performed using the Rate- Distortion Optimization (RDO) decision [34], which evaluates the bitrate and the objective quality (generally expressed by the PSNR, in dB) produced by every possible configuration. In other words, the prediction, residual coding and entropy coding stages are performed for every possible CU partition. During the prediction stage, each CU is once again divided into Prediction Units (PU), introducing a PU decision inside each node of the CU decision tree. Each PU decision also considers the RDO as decision strategy for mode selection [33].

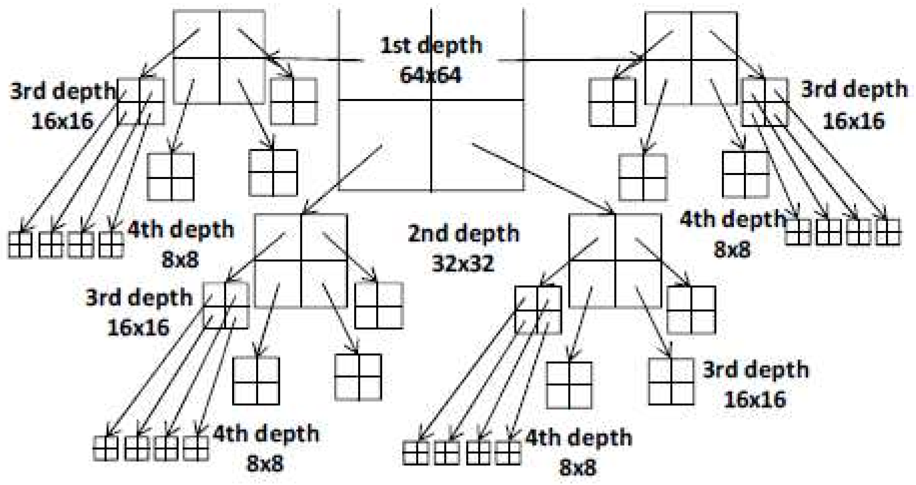


Figure 3-8 CU partitioning [32]

In the proposed method encoding time is heuristically accelerated by acting on a different level in the CU quadtree decision: the PU level, more specifically in the inter-prediction step. HEVC defines that each PU must be predicted for the three different modes available: skip, intra and inter-prediction. Analyses point that the inter-frame prediction consumes from 60% up to 90% of the total encoding time [32]. Such complexity is derived from motion estimation, the main tool in the inter-frame prediction, which performs exhaustive searches in order to find the best match between the current block of pixels that is being encoded and an area of pixels from one or more reference frames as shown in figures 3.7 and 3.8. These results indicate immense degrees of optimizations, since each CU node in the quad-tree must call ME for each possible PU partition.

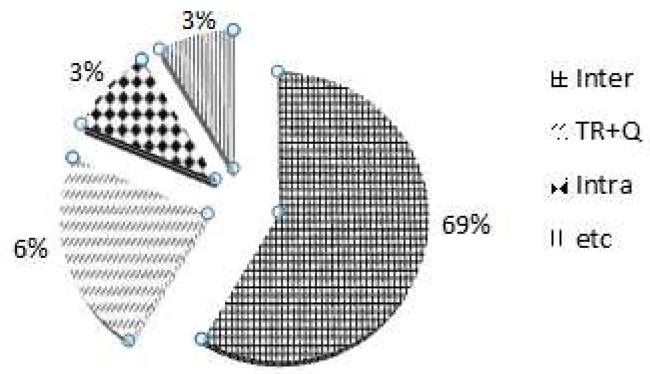


Figure 3-9 Encoder complexity of HEVC [35]

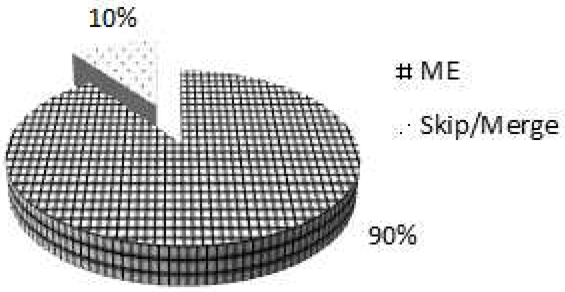


Figure 3-10 Complexity distribution between ME and skip/merge mode [35]

3.4 FAT Algorithm for Inter-prediction Mode Decision

3.4.1 Skip Mode

SKIP mode is the dominant mode at low bitrates (high QPs) in the HEVC, and the distribution is similar to that in the previous video coding standard, H.264/AVC. Once SKIP mode is pre-defined, the variable-sized ME computation of a CU can be entirely skipped. Usually the decision to use SKIP mode is delayed until the RD costs of all other modes (inter modes and intra modes) have been calculated and SKIP mode is found to have the minimum RD cost. Based on this consideration, early SKIP mode decision strategy is proposed for HEVC to avoid the entire variable-sized ME process as well as evaluations on Intra modes.

3.4.2 Prediction Size Based Mode Decision

At each depth level, various prediction mode sizes are used in the prediction procedures. Large sizes are always chosen for CUs in the homogeneous region, and small sizes are chosen for CUs with active motion or rich texture [36].

A mode complexity (MC) parameter of the current CU can be defined according to the mode context of the available predictors in Ω (mode complexity- MC) as follows [38]:

$$\mathbf{MC} = \frac{\sum_{i=1}^{N} w_i \cdot k_i \cdot \alpha_i}{\sum_{i=1}^{N} k_i \cdot \alpha_i}$$

where N is the number of CUs equal to 8 and w_i is the mode-weight factor. Only the prediction modes of those available adjacent CUs in Ω (mode complexity) will be used. Hence, k_i is set to 1, when the CU_i is available; otherwise, k_j is set to "0". The value α_i is the CU-weight factor, which is assigned to the adjacent CUs according to their correlation from the current CU. The stronger correlation between the neighboring CU and the current CU, the larger the weight should be assigned. Experiments are conducted to compute the correlation degree of neighboring CUs and the current CU [38].

Generally, the more complexity the CU has, the larger the value of the mode complexity will be. Based on the mode complexity, each CU is classified as simple mode, normal mode, or complex mode as follows [38]:

$$\begin{cases} \mathbf{MC} < T_{r0} & \mathrm{CU} \in \textit{simple mode} \\ T_{r0} \le \mathbf{MC} < T_{r1} & \mathrm{CU} \in \textit{normal mode} \\ \mathbf{MC} \ge T_{r1} & \mathrm{CU} \in \textit{complex mode} \end{cases}$$

where, Tr_0 is equal to the mode-weight factor of Inter 2Nx2N or Intra 2Nx2N, i.e., 1, and Tr_1 is equal to the mode-weight factor of Inter NxN, i.e., 4.

For a CU with simple mode, the area covered by the current CU and its adjacent CUs usually contain slow-motion content or homogeneous texture, and the optimal prediction modes of its adjacent CUs are usually within the Inter 2N×2N, Intra 2N×2N, merge mode and the SKIP mode. Therefore, a CU with a simple mode only an needs ME of size 2N×2N that performs Intra 2N×2N prediction. For a CU with complex mode, most of its adjacent CUs choose small-size inter modes or Intra N×N, and thus it is not necessary to perform ME on sizes of 2N×2N, 2N×N and N×2N. Thus, inter-mode decision for HEVC jointly utilizes inter-level and spatio-temporal correlations [38].

3.4.3 RD Cost Based Mode Decision

In HM, the mode decision process exhaustively searches the best mode from candidate modes according to the RD optimization scheme. It is analyzed in [38] [39] that the majority of best prediction modes after mode decision in inter frames are usually with large size modes such as SKIP mode, Merge mode and Inter 2Nx2N, which implies that small-sized ME and intra prediction are unnecessary in most cases. So, it is better to have a proper early termination (ET) strategy in the midway of fast mode decision algorithm. Basically, ET strategies are all threshold-based, and most of ET threshold determination algorithms are based on the RD cost. Meanwhile, a mass amount of inter-level, spatial and temporal correlations exists in the coding procedure of the HEVC. Thus, the resultant RD cost information of the current CU is closely related to that of its adjacent CUs. Thus, the RD cost of the current CU ($RDcost_{pre}$) can be predicted using the minimal RD cost values from the available predictors in Ω as follows [38]:

$$Rdcost_{pre} = \frac{\sum_{i=1}^{6} \alpha_i \cdot k_i \cdot Rdcost_i + \alpha_7 \cdot k_7 \cdot Rdcost_7 / 4 + \alpha_8 \cdot k_8 \cdot Rdcost_8 / 16}{\sum_{i=1}^{8} k_i \cdot \alpha_i}$$

where $Rdcost_i$ are the RD cost of CU_i, and $Rdcost_7$ and $Rdcost_8$ are the RD costs of parent CUs in the two upper depth levels. α_i and k_i are defined as the same as those motion complexity in section 3.4.2. The threshold (Thr) is determined based on the minimum value among $RDcost_{pre}$ and RD costs of spatial neighboring predictors in Ω [38]:

$$\mathbf{Thr} = \lambda \cdot \min\left\{RDcost_1, RDcost_2, RDcost_3, RDcost_4, RDcost_{pre}\right\}$$

where λ is the adjustment parameter. To set the value of λ, the signaling information CBF in the HEVC syntax should be considered, which specifies the non-zero transform coefficient levels. The CBF will be generated after checking each prediction mode. If the CBF of a test mode equals to zero, the current CU is an all-zero block. In this case, the adjustment parameter is reset with 1.25 x λ. When the minimal RD cost value is smaller than threshold (Thr), the mode decision procedure is terminated in the proposed method, and small-sized ME and intra-prediction are skipped.

3.4.3 Algorithm for Mode Decision

Based on the strategies of early SKIP mode decision, prediction size based mode decision and RD cost based mode decision, the proposed adaptive inter-mode decision algorithm for HEVC is determined as below [38]:

Step 1: Start mode decision for CUs in inter frames

Step 2: Derive coding information of parent CUs in the upper depth levels and spatially/temporally adjacent CUs.

Step 3: Test SKIP mode and Merge mode. If the current CU meets one of the three early SKIP mode decision conditions in section 3.4.1, go to Step 7.

Step 4: Compute MC from section 3.4.2. When the current CU is with simple mode, the candidate mode is Inter 2Nx2N and Intra 2Nx2N; when the current CU is with complex mode, the candidate modes include Inter NxN, Inter 2NxnU, Inter 2NxnD, Inter nLx2N, Inter nRx2N, and Intra NxN (available for the smallest CUs); otherwise, the candidate modes include all prediction modes.

Step 5: Compute Threshold (Thr) for early termination from section 3.4.2.

Step 6: Loop each candidate inter mode and intra mode:

> Step 6.1: Perform ME and get RD cost and CBF of the current mode;
>
> Step 6.2: If the RD cost value is smaller than threshold (Thr) , terminate the mode decision procedure and go to Step 7.
>
> End Loop

Step 7: Determine the best prediction mode for the current CU.

3.5 Summary

This chapter outlines the description of inter-prediction and existing mode decision performed in HM software along with a description of the fast adaptive termination of the mode decision algorithm for inter-prediction in the HEVC standard. Chapter 4 outlines the experimental setup, results and conclusions based on the algorithm developed in this chapter.

Chapter 4

Results

4.1 Test Conditions

In order to evaluate the performance of the proposed mode decision algorithm, the algorithm is implemented on the recent HEVC reference software (HM 13.0) [23]. The random access profile is used for coding with GOP (Group of pictures) as 8. Coding tree block size is fixed at 64x64 pixels for luma with maximum depth of 4 resulting in minimum CU size of 8x8 pixels for luma. The proposed algorithm is evaluated with 4 QPs of 22, 27, 32 and 37 using following test sequences recommended by JCT-VC. A frame of each test sequence is shown in Appendix A.

Table 1 Test Sequences Used

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

4.2 Encoding Time Gain

With the proposed fast adaptive early termination algorithm, encoding time for the test sequences is reduced by 29-38% as compared to the unmodified encoding HM13.0. The following test results (figures 4-1 to 4-5) show the difference in encoding time of original HM13.0 and the improved HM13.0 for different quantization parameter (QP) values as suggested by JCTVC [43].

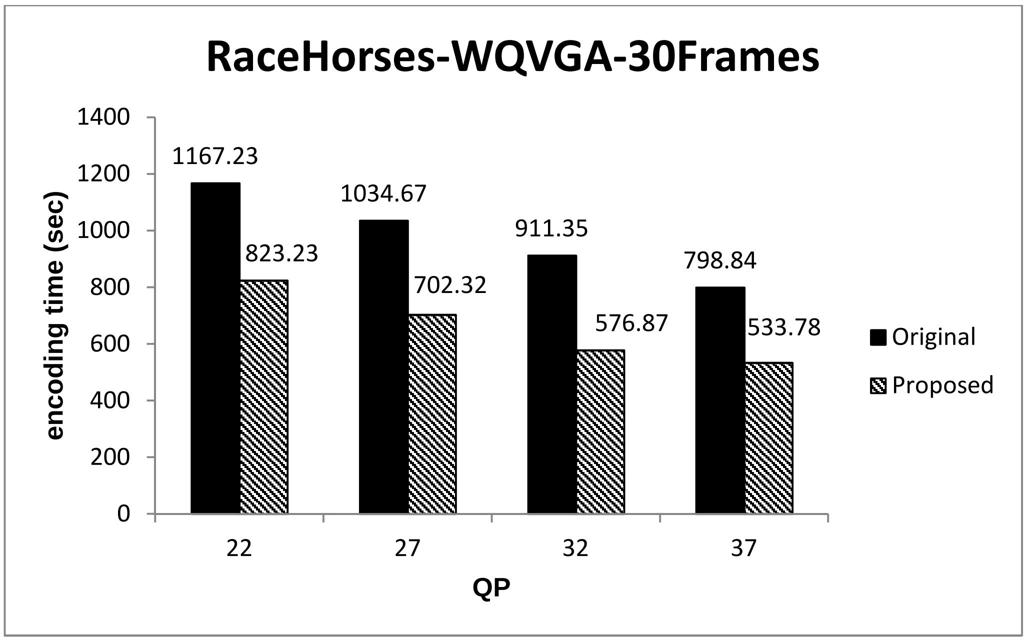


Figure 4-1 Encoding time vs. quantization parameter for Racehorses

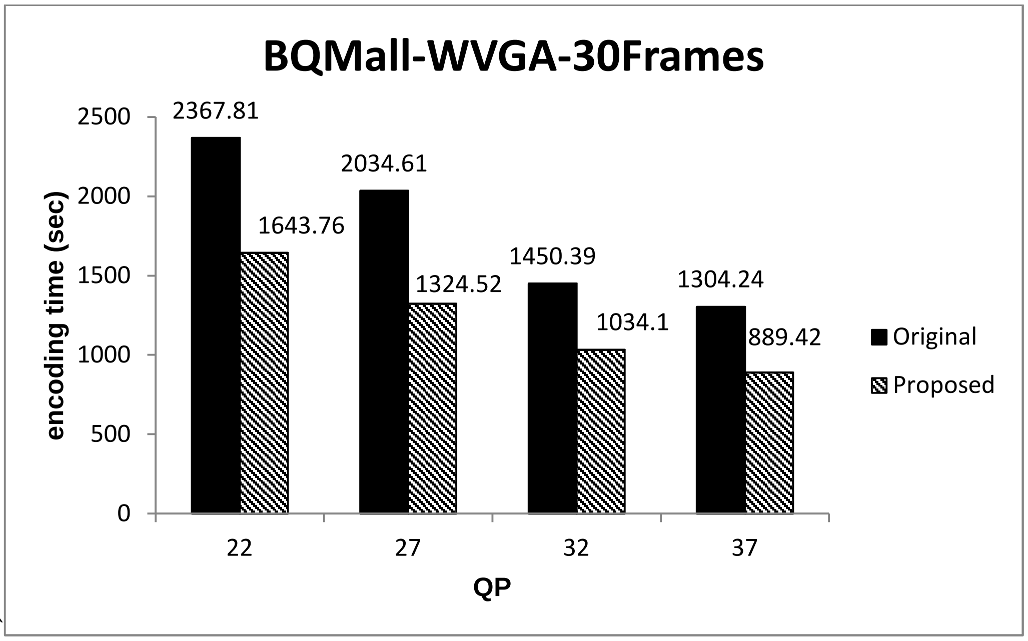


Figure 4-2 Encoding time vs. quantization parameter for BQMall

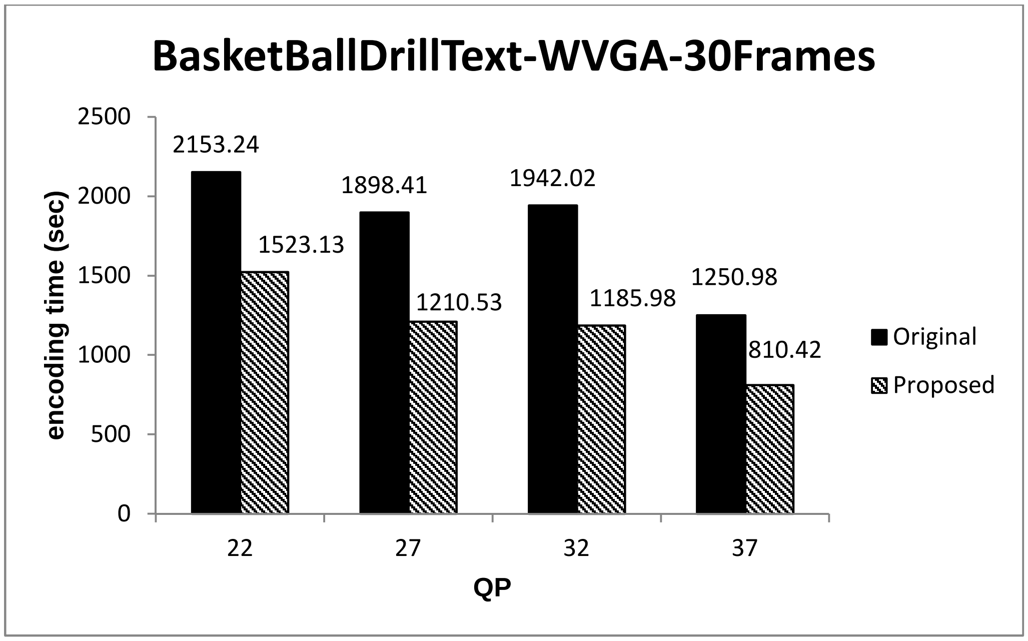


Figure 4-3 Encoding time vs. quantization parameter for BasketBallDrillText

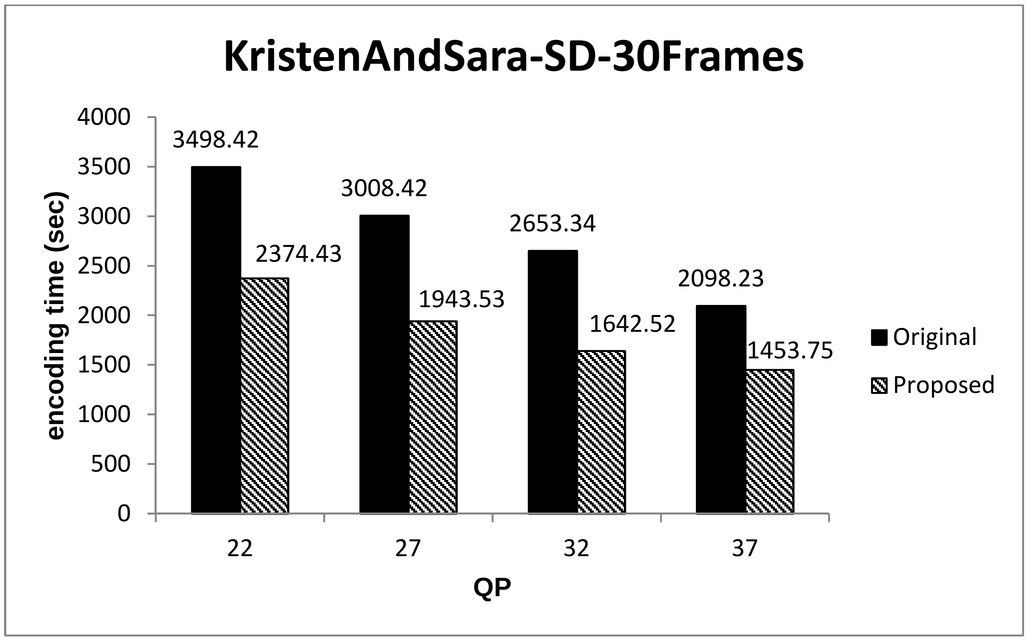


Figure 4-4 Encoding time vs. quantization parameter for KristenAndSara

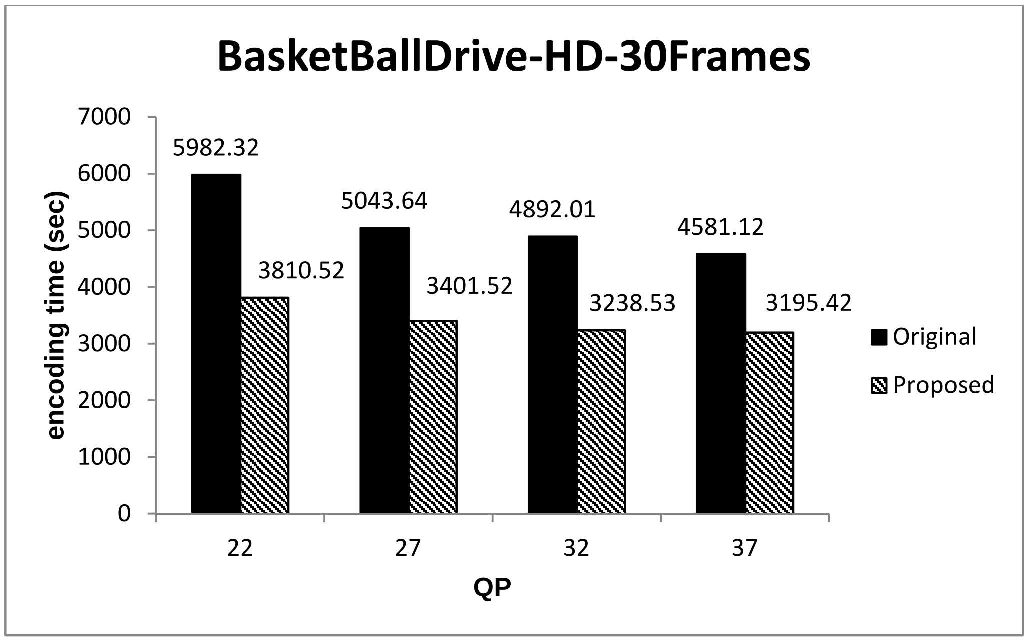


Figure 4-5 Encoding time vs. quantization parameter for BasketBallDrive

4.3 BD-PSNR

To objectively evaluate the coding efficiency of video codecs, Bjøntegaard Delta PSNR (BD-PSNR) was proposed [46]. Based on the rate-distortion (R-D) curve fitting, BD-PSNR is able to provide a good evaluation of the R-D performance [46]. 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 video sequence [23] [24]. 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 drop in PSNR using BD metrics for the proposed algorithm for the HM13.0 in the range of 0.31 dB to 0.51 dB.

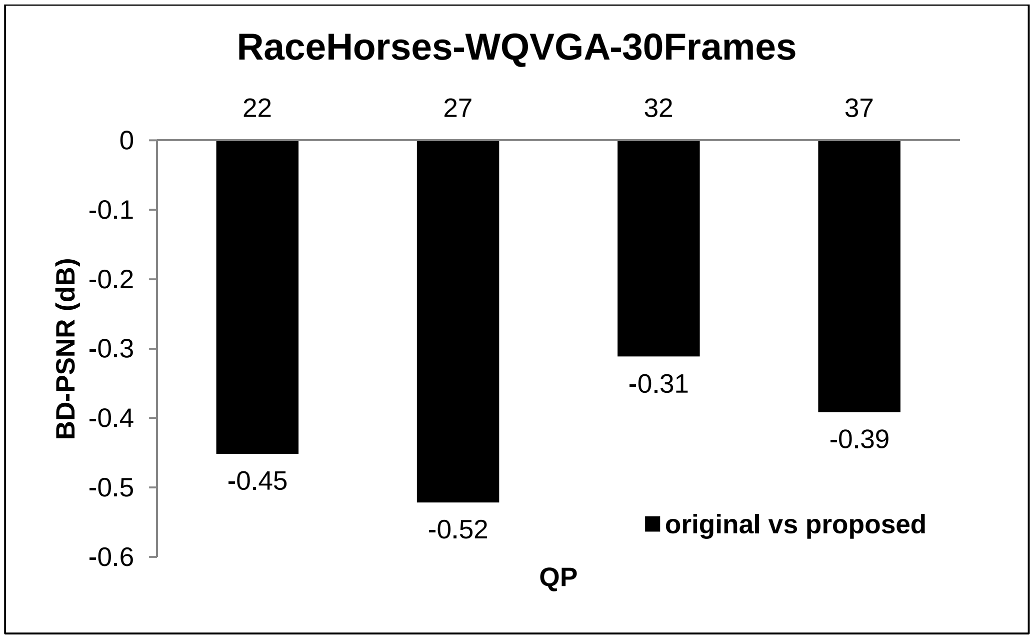


Figure 4-6 BD-PSNR vs. quantization parameter for RaceHorses

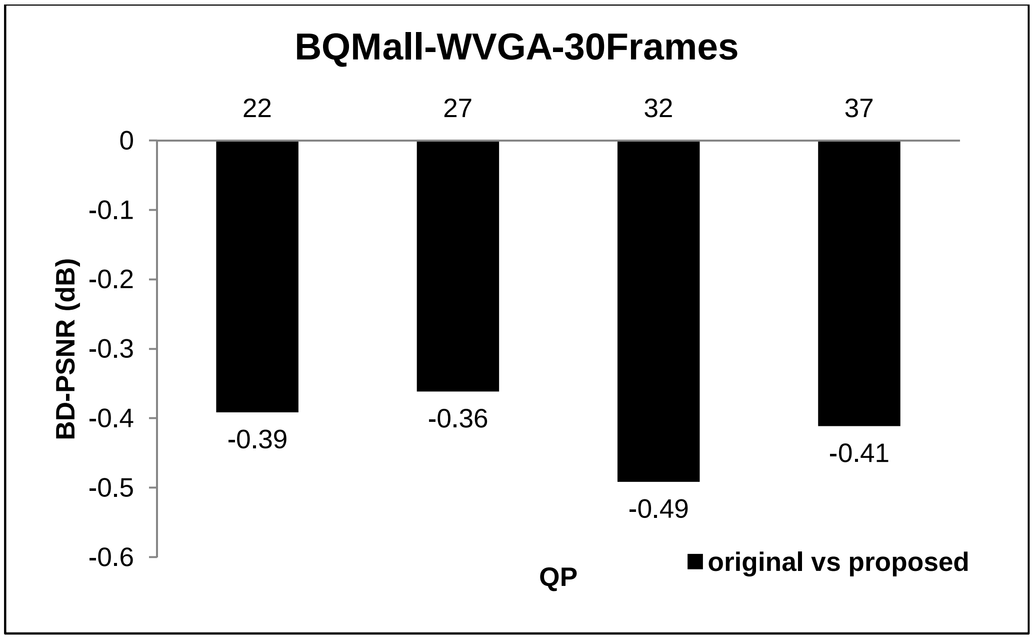


Figure 4-7 BD-PSNR vs. quantization parameter for BQMall

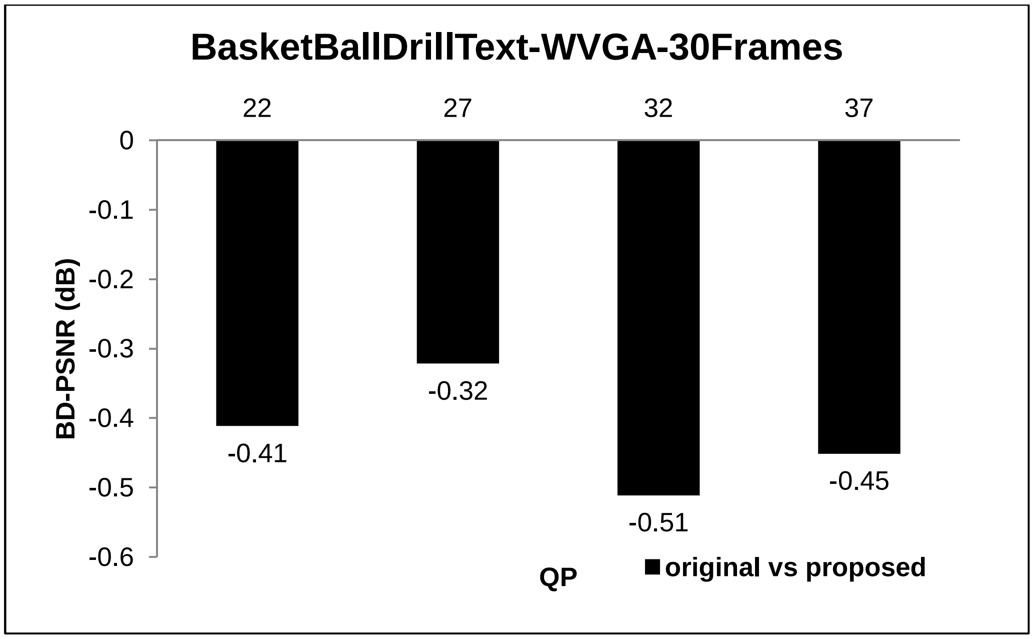


Figure 4-8 BD-PSNR vs. quantization parameter for BasketBallDrillText

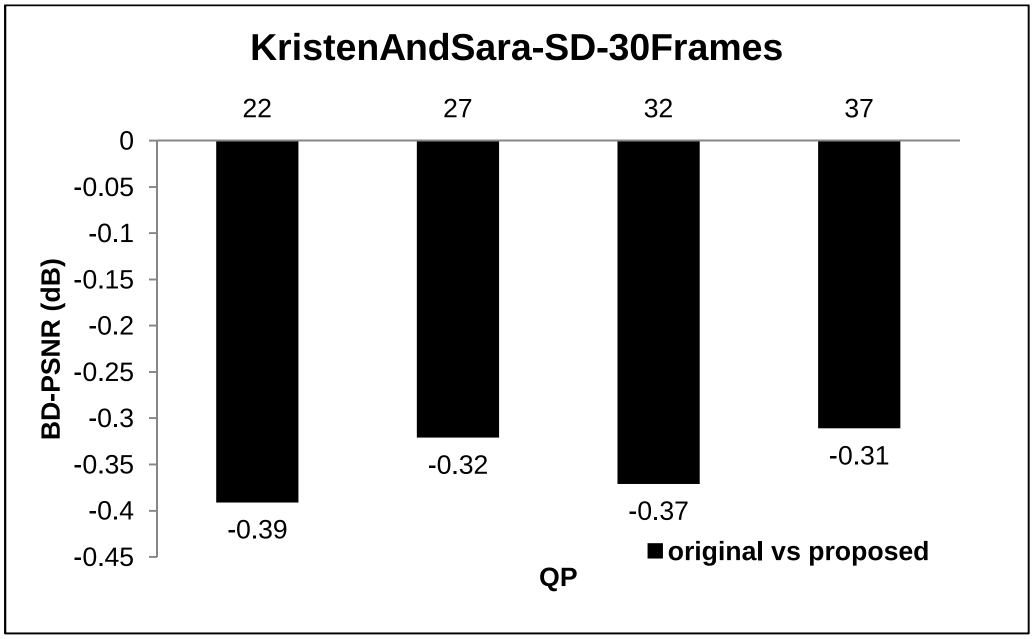


Figure 4-9 BD-PSNR vs. quantization parameter for KristenAndSara

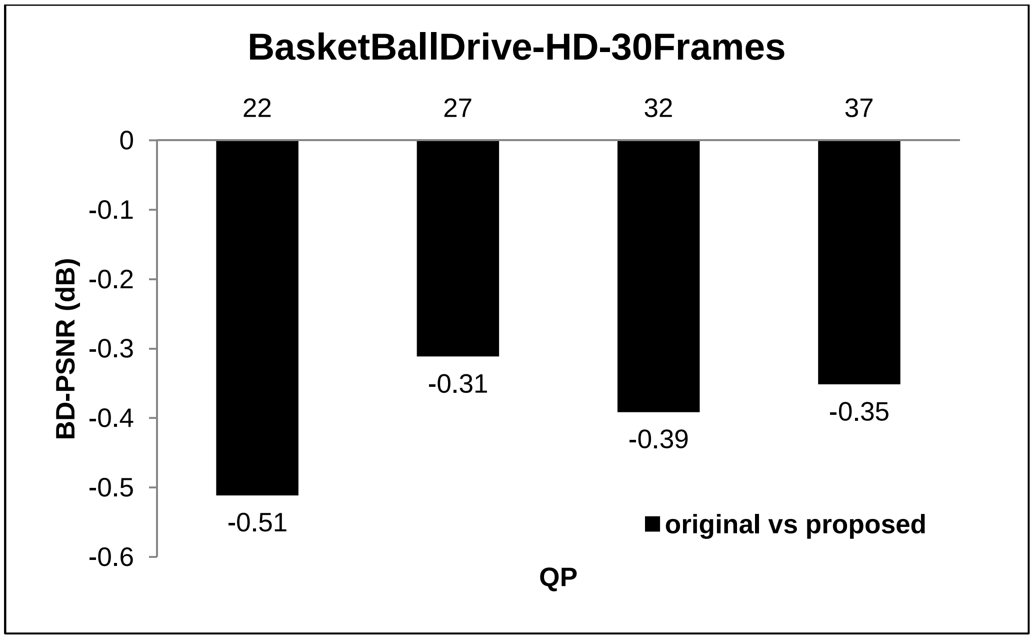


Figure 4-10 BD-PSNR vs. quantization parameter for BasketBallDrive

4.4 BD-bitrate

BD-bitrate is a metric similar to the BD-PSNR metric which determines the quality of encoded video sequence along with measure the output bitstream of encoded video sequence. It can be observed from figures 4-11 to 4-15 that there is an increase in bitrate using BD metrics for the proposed algorithm for HM13.0 in the range of 7 kbps to 12 kbps (b-Bit).

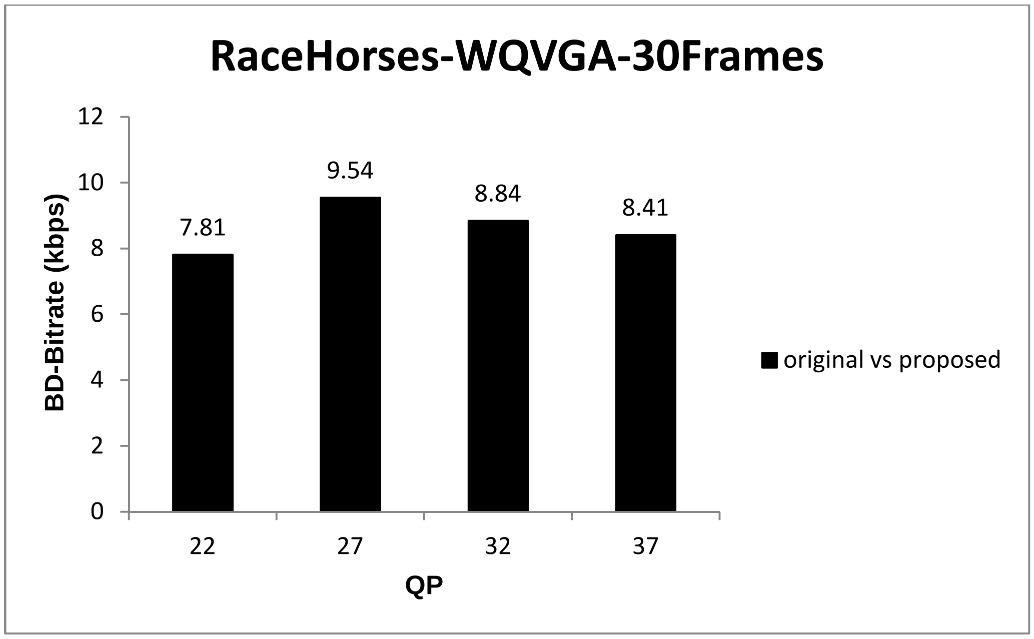


Figure 4-11 BD-bitrate vs. quantization parameter for RaceHorses

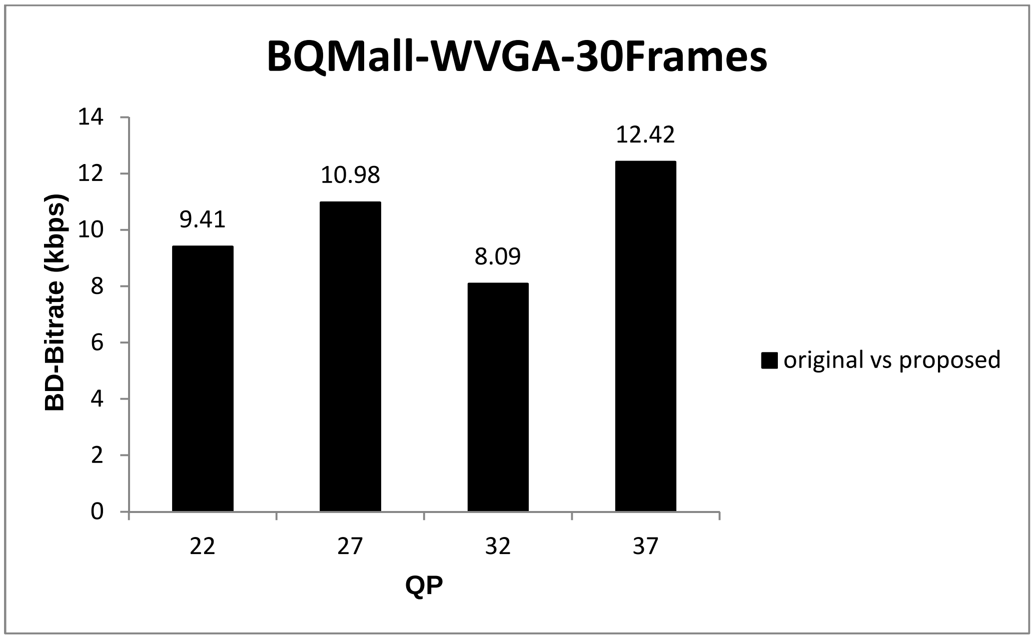


Figure 4-12 BD-bitrate vs. quantization parameter for BQMall

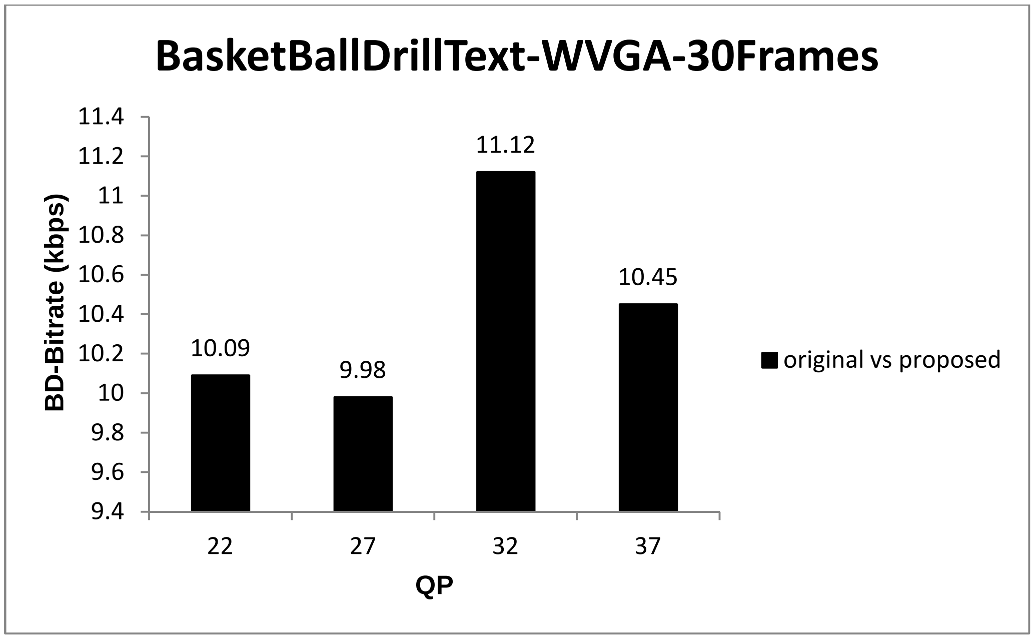


Figure 4-13 BD-bitrate vs. quantization parameter for BasketBallDrillText

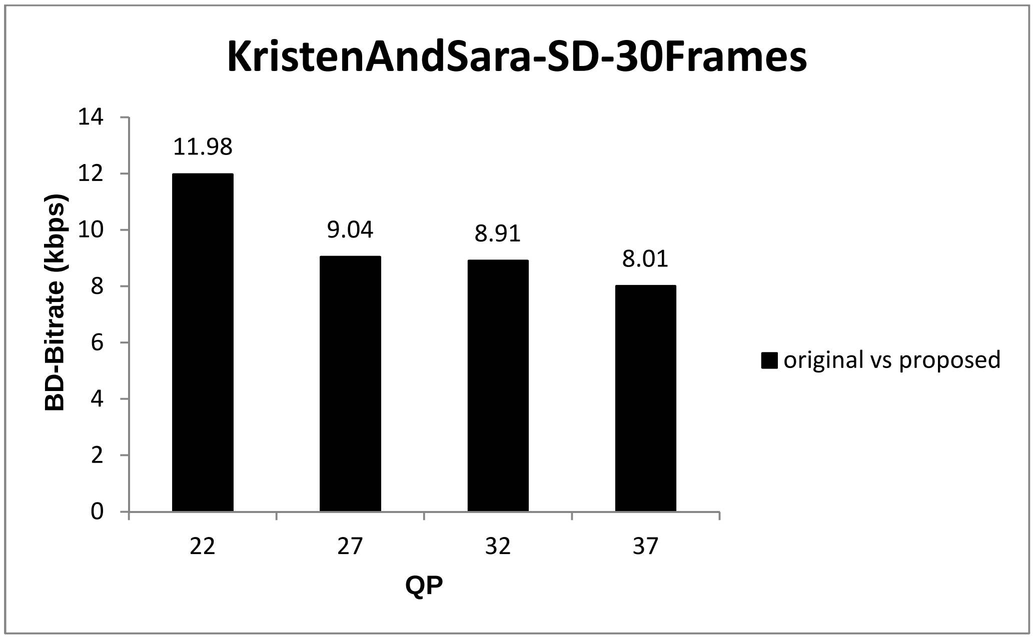


Figure 4-14 BD-bitrate vs. quantization parameter for KristenAndSara

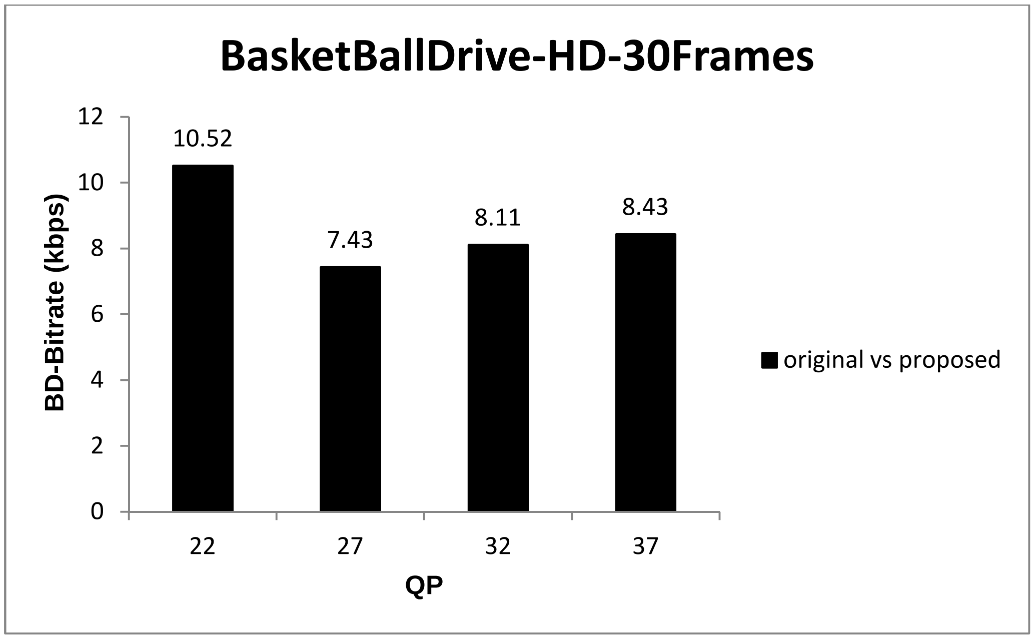


Figure 4-15 BD-bitrate vs. quantization parameter for BasketBallDrive

4.5 Rate Distortion Plot (RD Plot)

The proposed algorithm has negligible PSNR loss and bitrate increase. Figure 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 HM13.0 encoder.

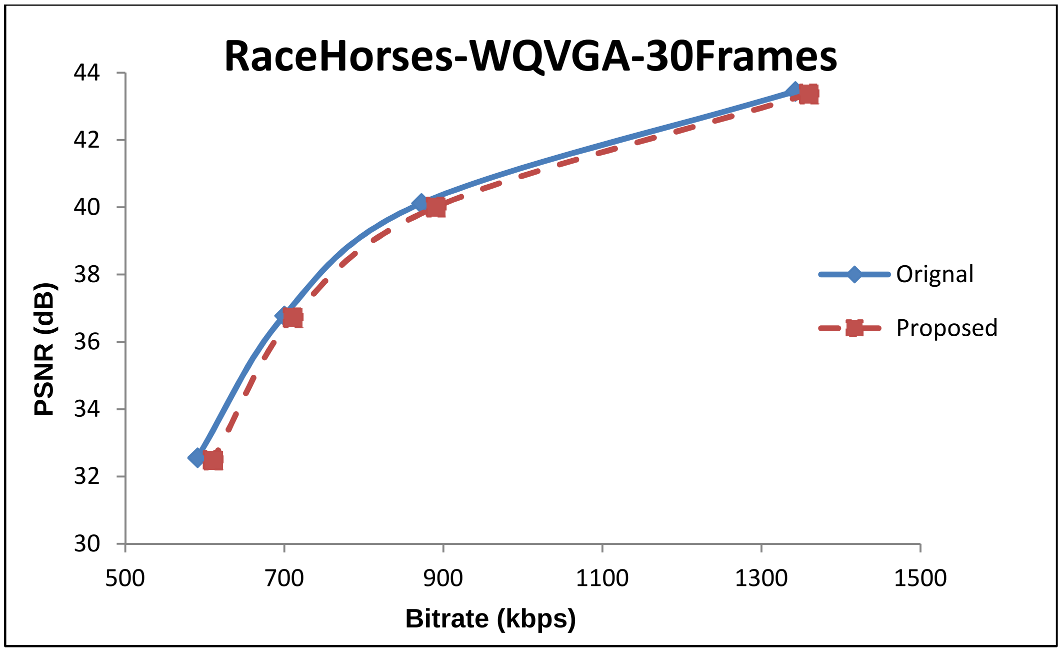


Figure 4-16 PSNR vs. bitrate for RaceHorses

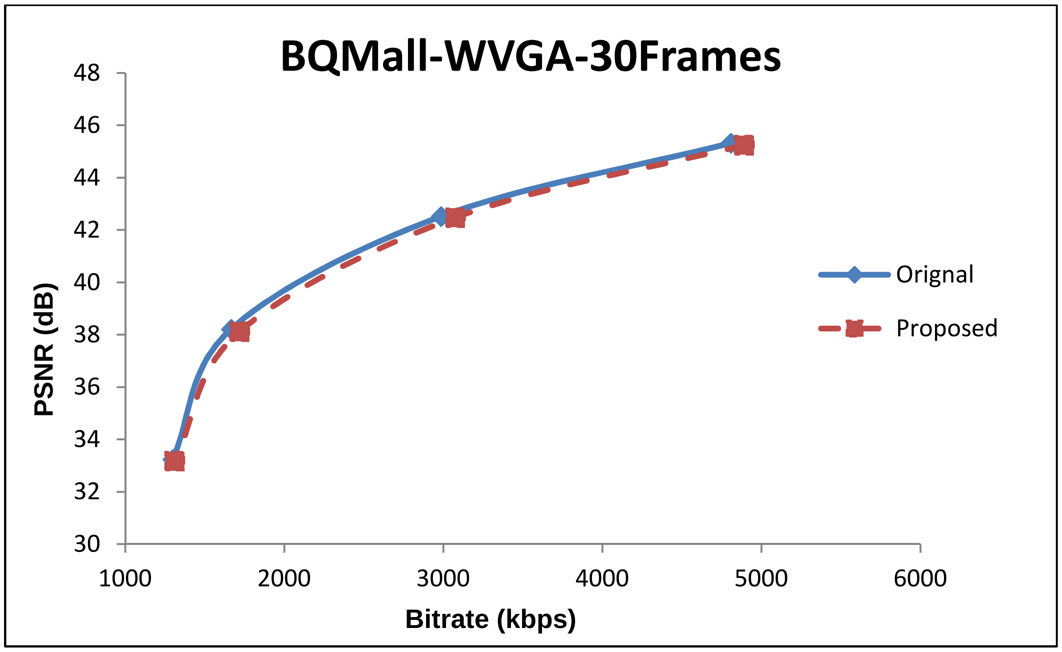


Figure 4-17 PSNR vs. bitrate for BQMall

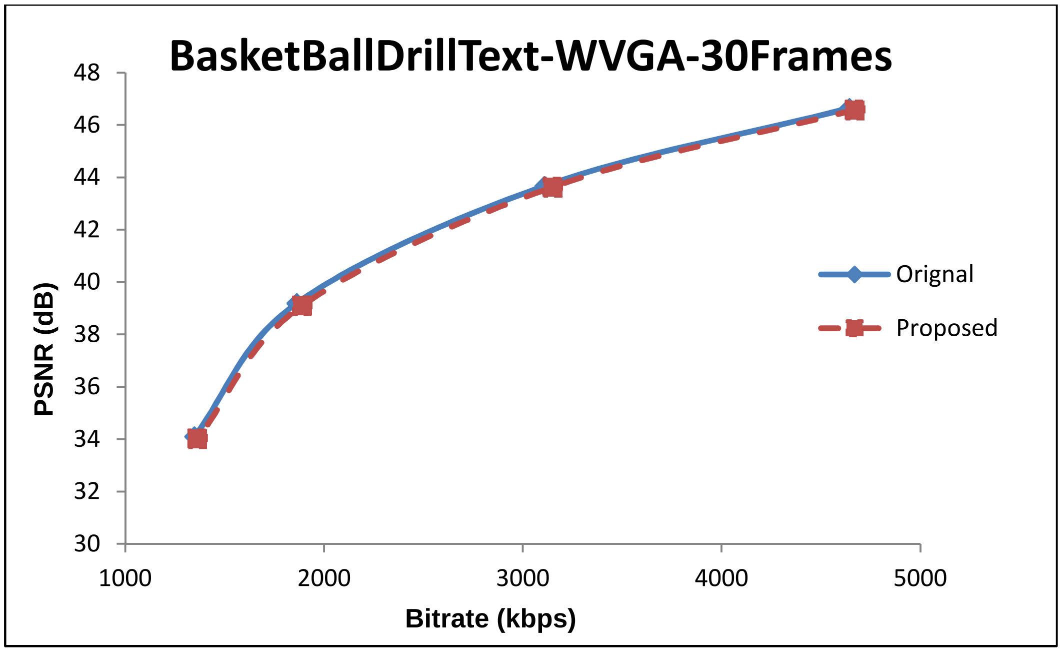


Figure 4-18 PSNR vs. bitrate for BasketBallDrillText

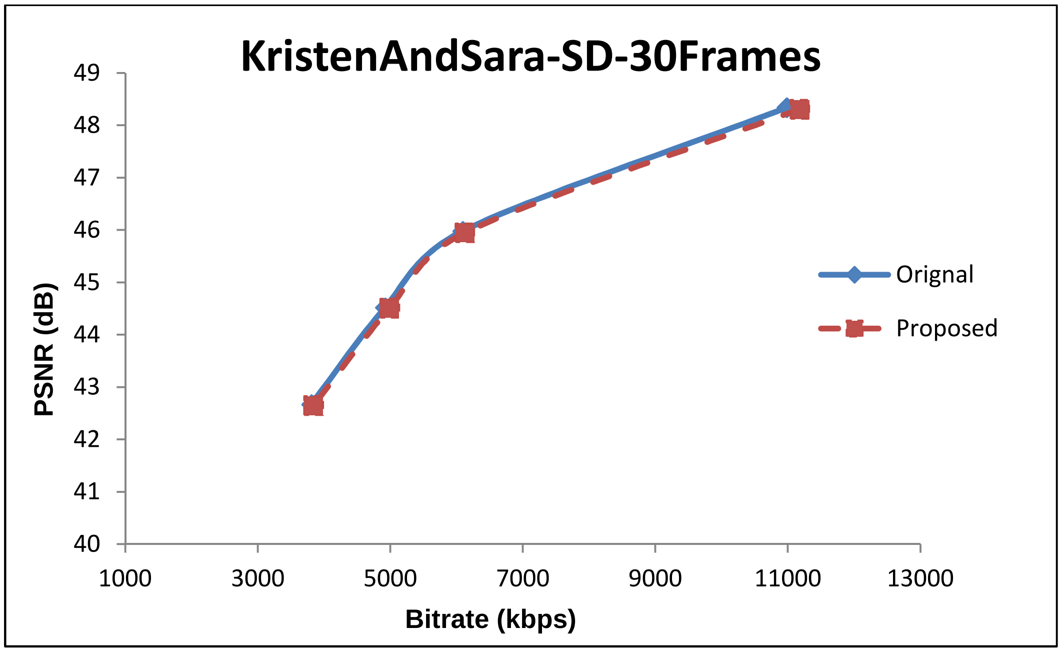


Figure 4-19 PSNR vs. bitrate for KristenAndSara

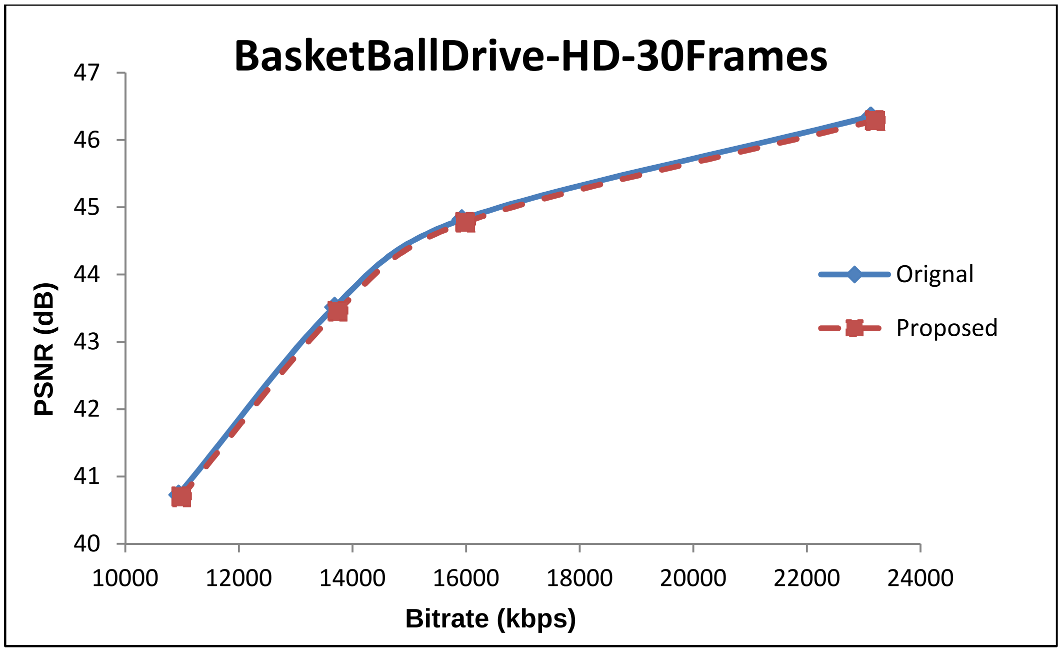


Figure 4-20 PSNR vs. bitrate for BasketBallDrive

4.6 Bitstream Size Gain

Figures 4-21 to 4-25 show the encoded bitstream size for the original HM13.0 and the proposed HM13.0 encoded for different quantization parameter values. It can be observed that there is 1% to 5% increase in bitstream size.

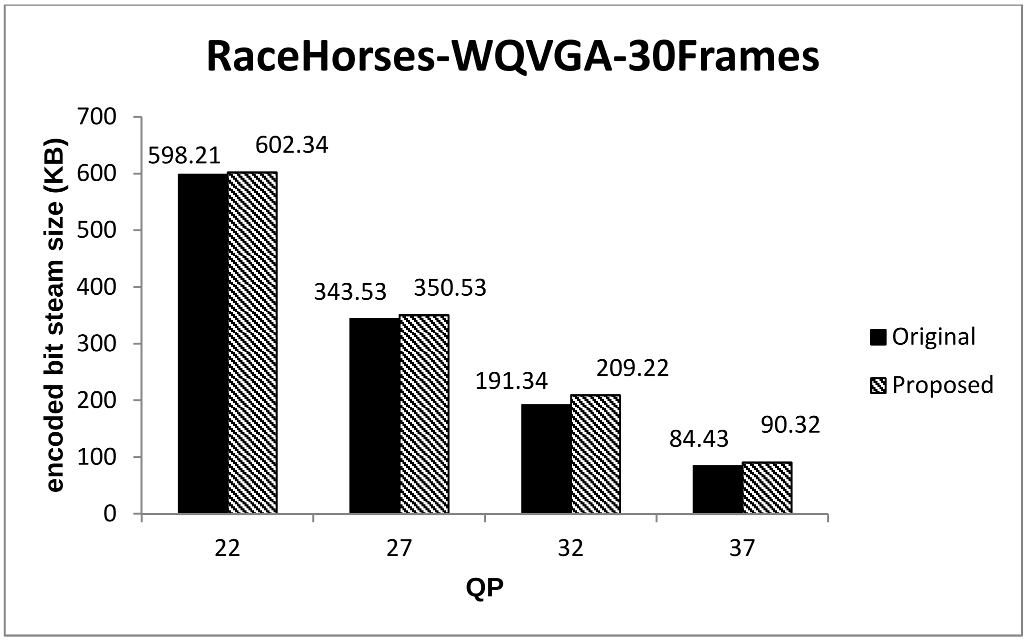


Figure 4-21 Encoded bitstream size vs. quantization parameter for RaceHorses

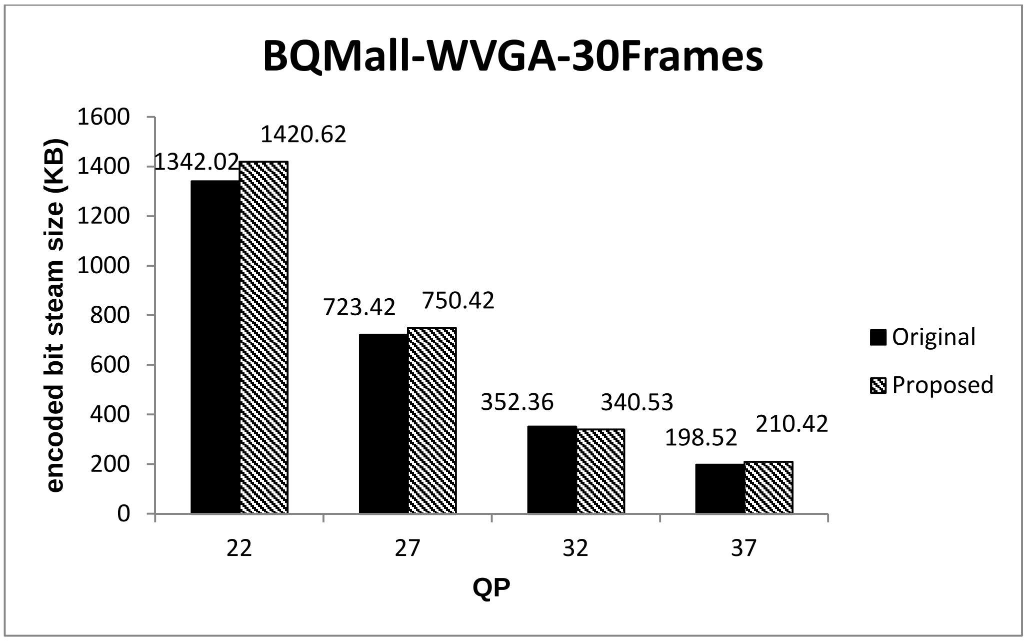


Figure 4-22 Encoded bitstream size vs. quantization parameter for BQMall

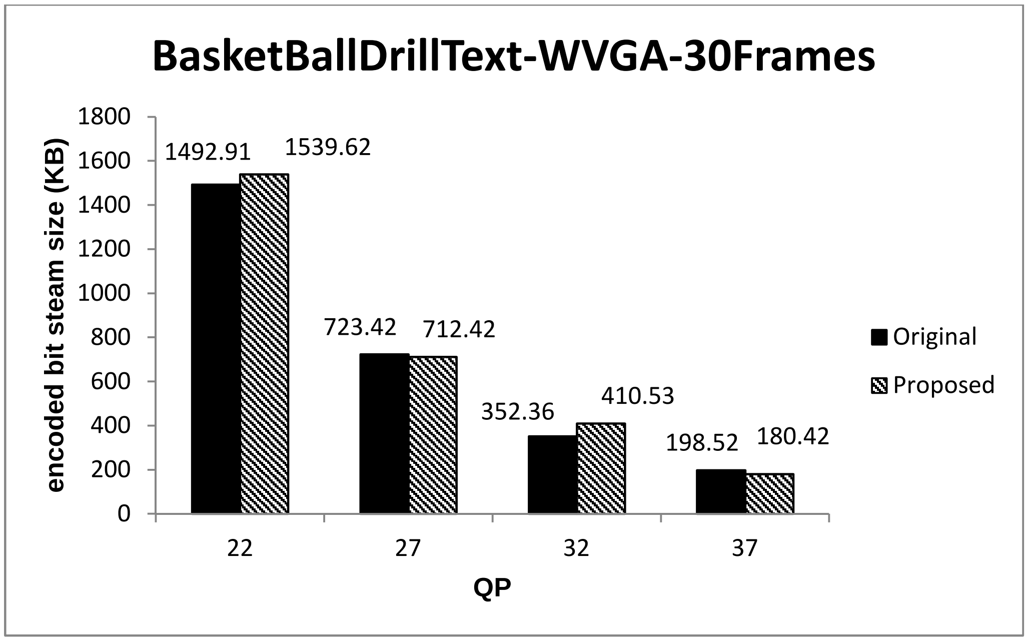


Figure 4-23 Encoded bitstream size vs. quantization parameter for BasketBallDrillText

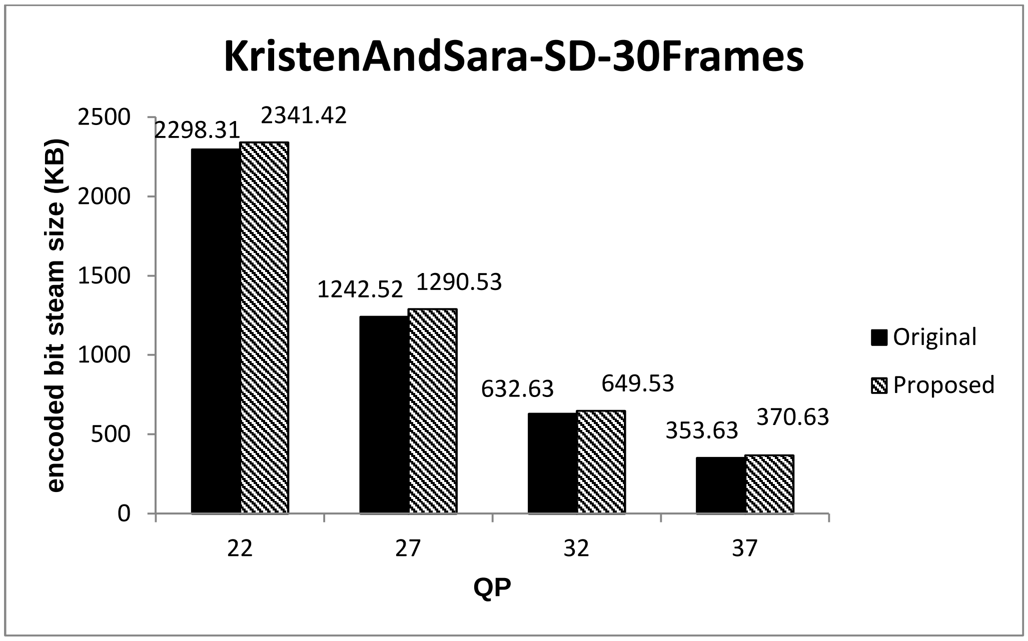


Figure 4-24 Encoded bitstream size vs. quantization parameter for KristenAndSara

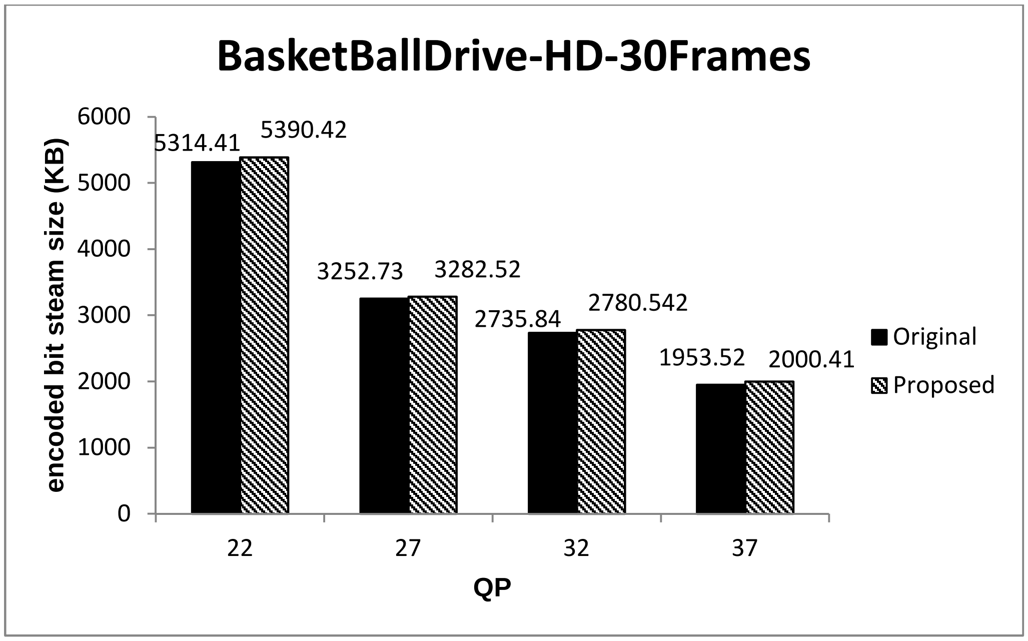


Figure 4-25 Encoded bitstream size vs. quantization parameter for BasketBallDrive

4.7 Structural Similarity (SSIM)

The structural similarity [44] (SSIM) index is a method for measuring the similarity between two images. The SSIM index is a full reference metric; in other words, the measurement of image quality based on an initial uncompressed or distortion-free image as reference. SSIM is designed to improve on traditional methods like peak signal-to-noise ratio (PSNR) and mean squared error (MSE), which have proven to be inconsistent with human eye perception. Figures 4-26 to 4-30 show the SSIM for the original HM13.0 and the improved HM13.0 encoded for different quantization parameter values. It can be observed that there is negligible decrease in SSIM (0.003dB to 0.008dB).

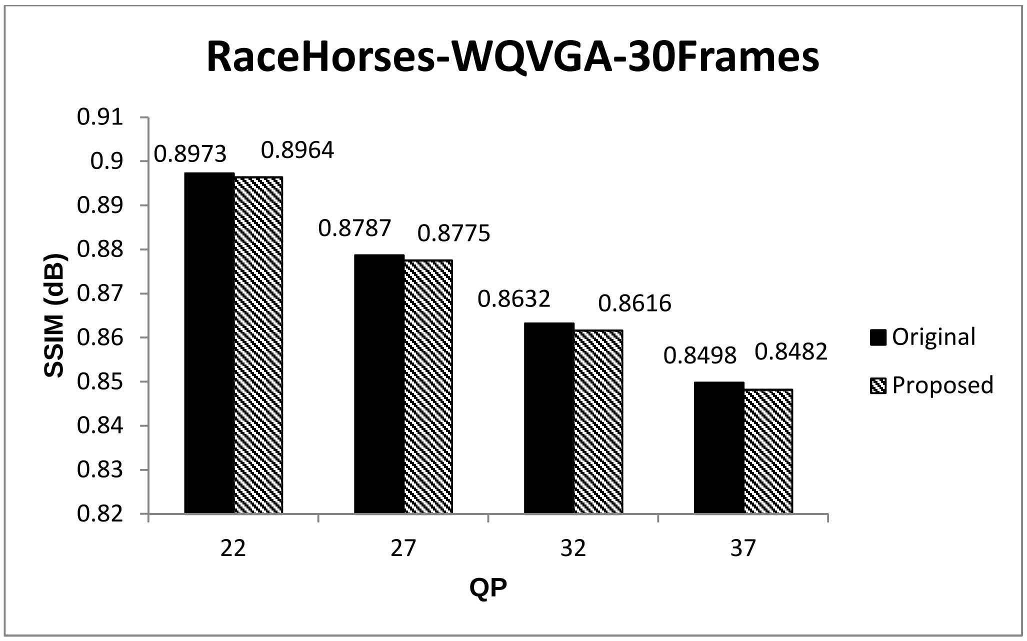


Figure 4-26 SSIM size vs. quantization parameter for RaceHorses

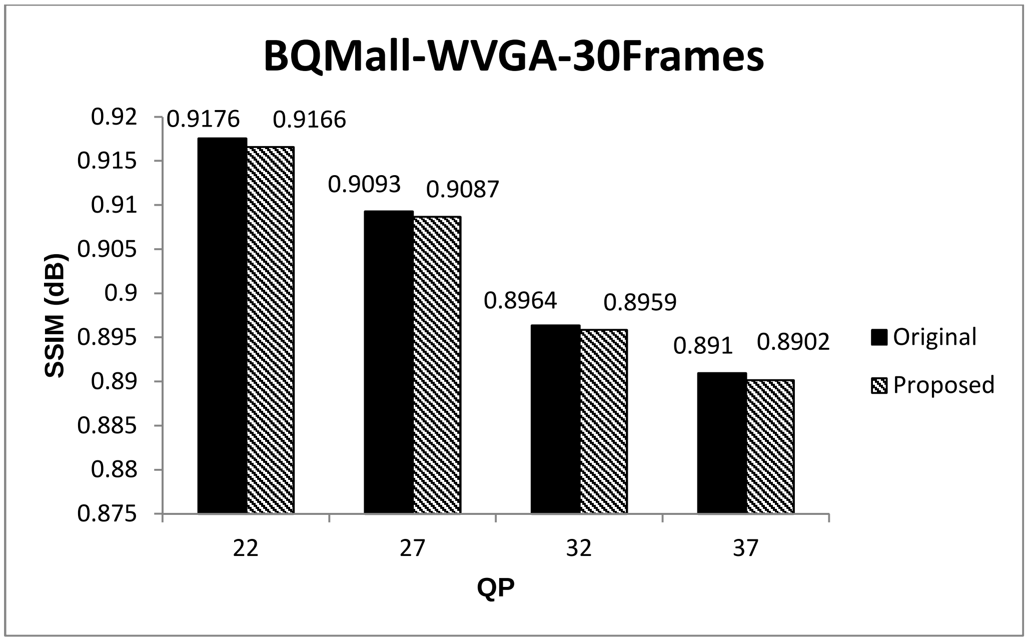


Figure 4-27 SSIM vs. quantization parameter for BQMall

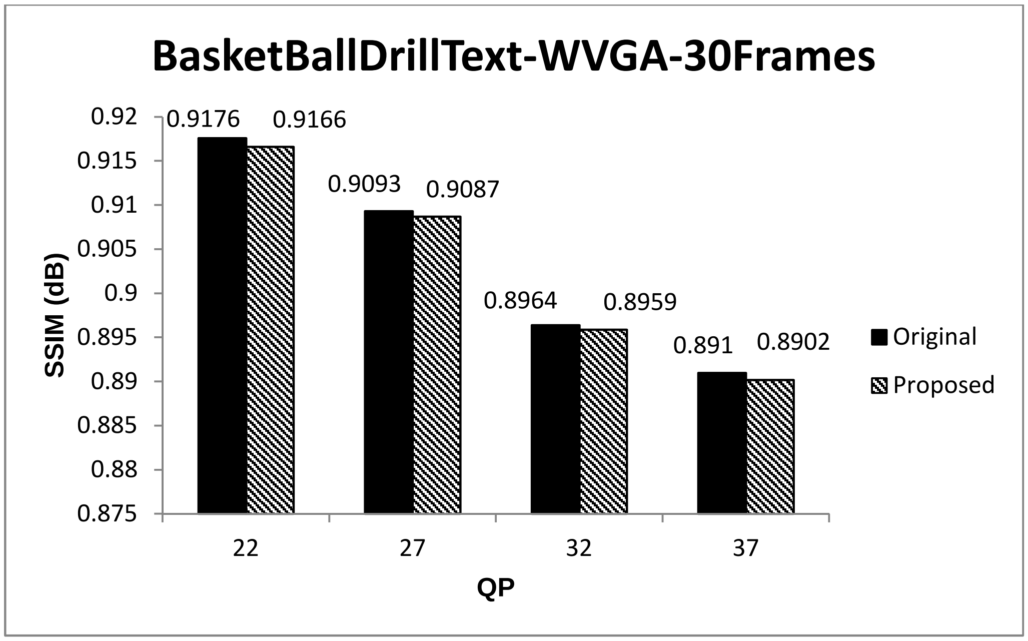


Figure 4-28 SSIM vs. quantization parameter for BasketBallDrillText

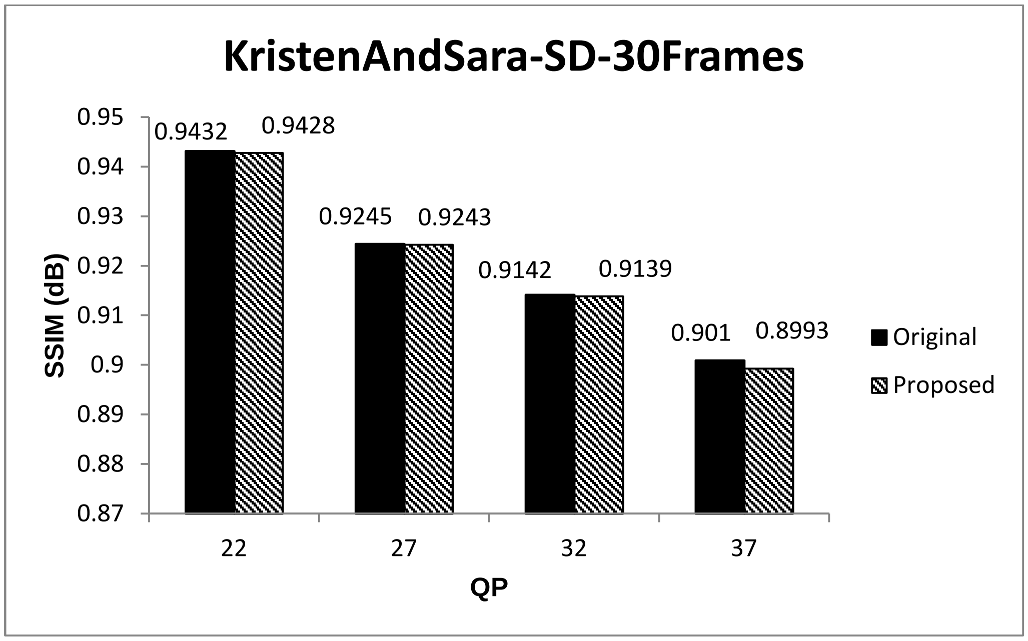


Figure 4-29 SSIM vs. quantization parameter for KristenAndSara

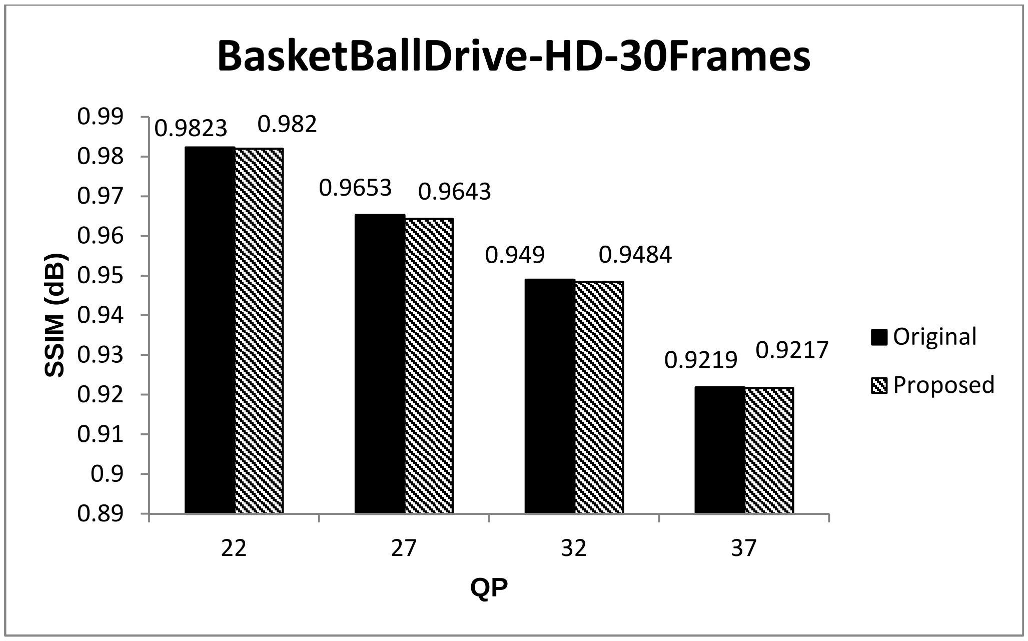


Figure 4-30 SSIM vs. quantization parameter for BasketBallDrive

4.8 SSIM-bitrate Plot

SSIM-bitrate plot determines the quality of encoded video similar to PSNR-bitrate plot. SSIM-bitrate plot also determine the loss in perceptual quality of encoded video [47]. Figures 4-31 to 4-35 show the SSIM-bitrate plot for original HM13.0 and the improved HM13.0 encoded. It can be observed that there is negligible loss in perceptual quality as indicated by SSIM.

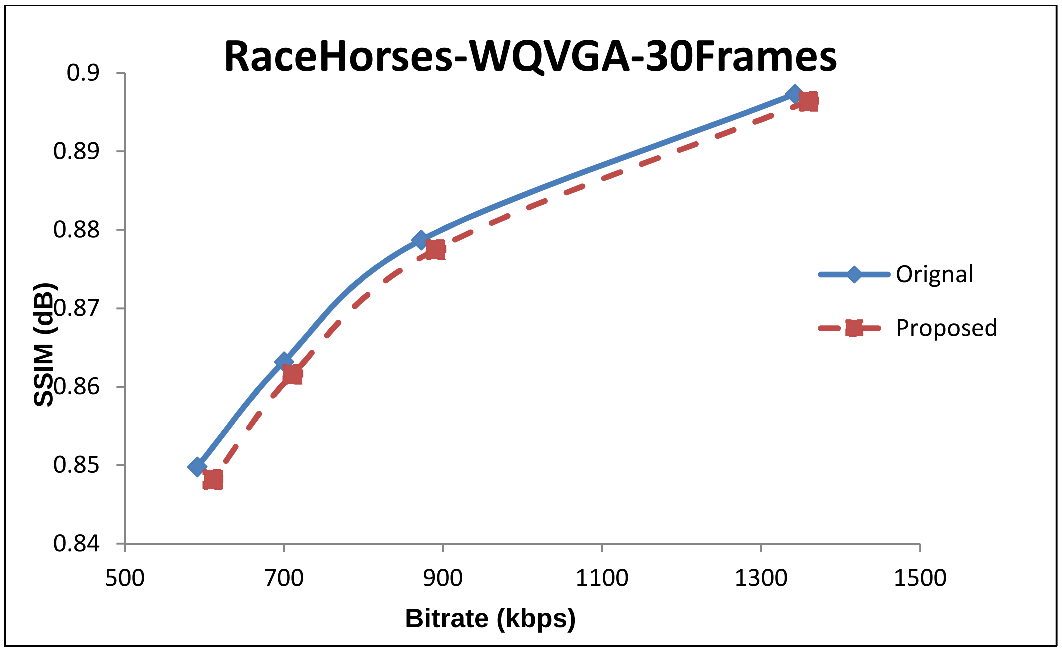


Figure 4-31 SSIM vs. bitrate for RaceHorses

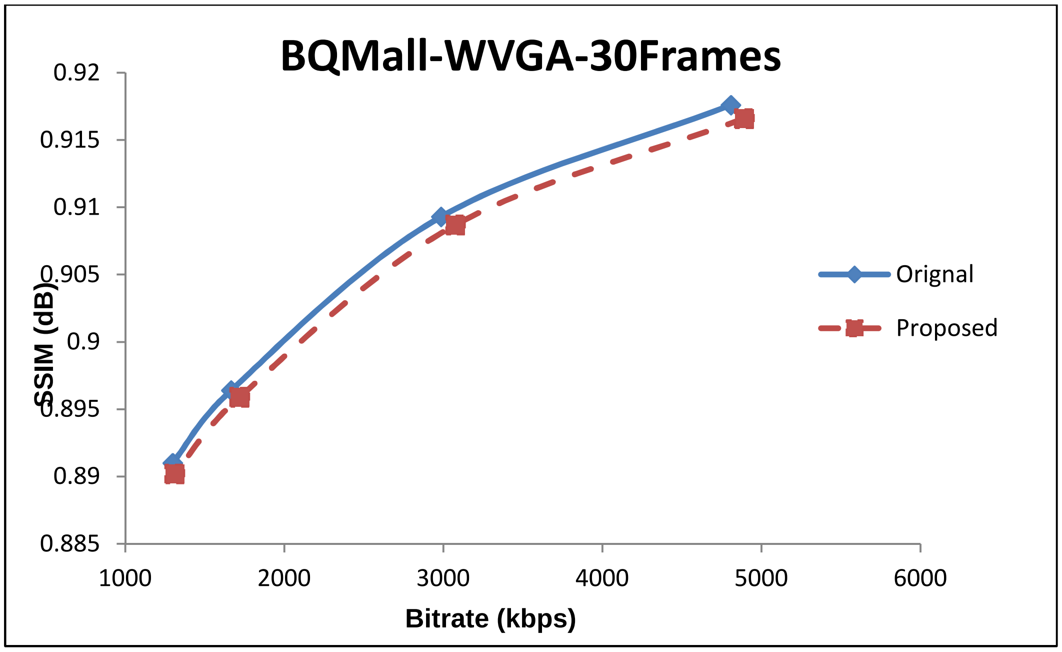


Figure 4-32 SSIM vs. bitrate for BQMall

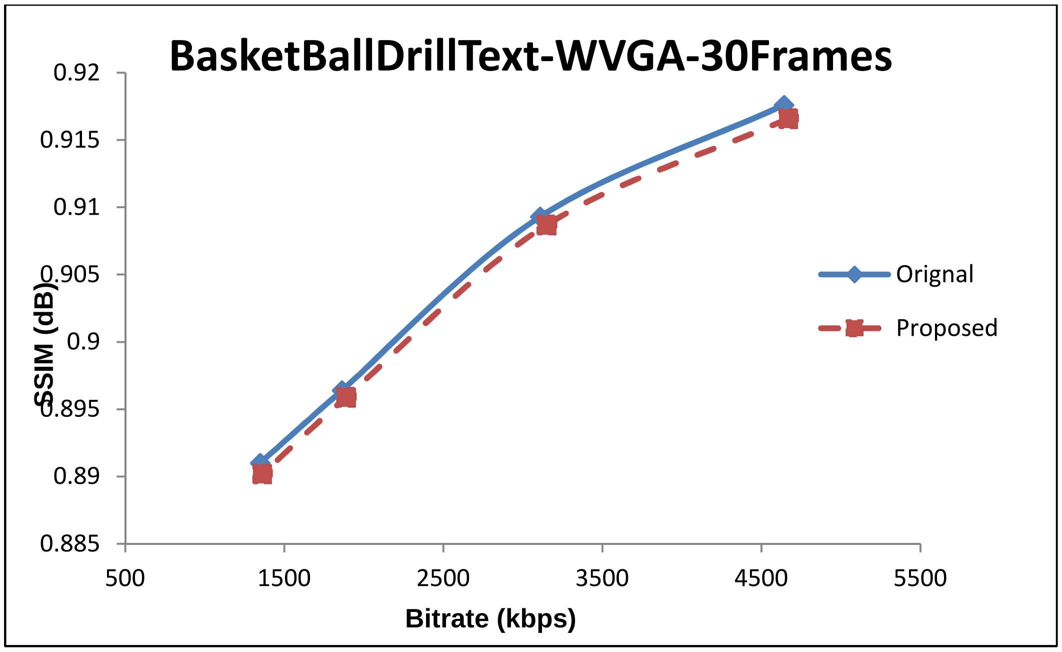


Figure 4-33 SSIM vs. bitrate for BasketBallDrillText

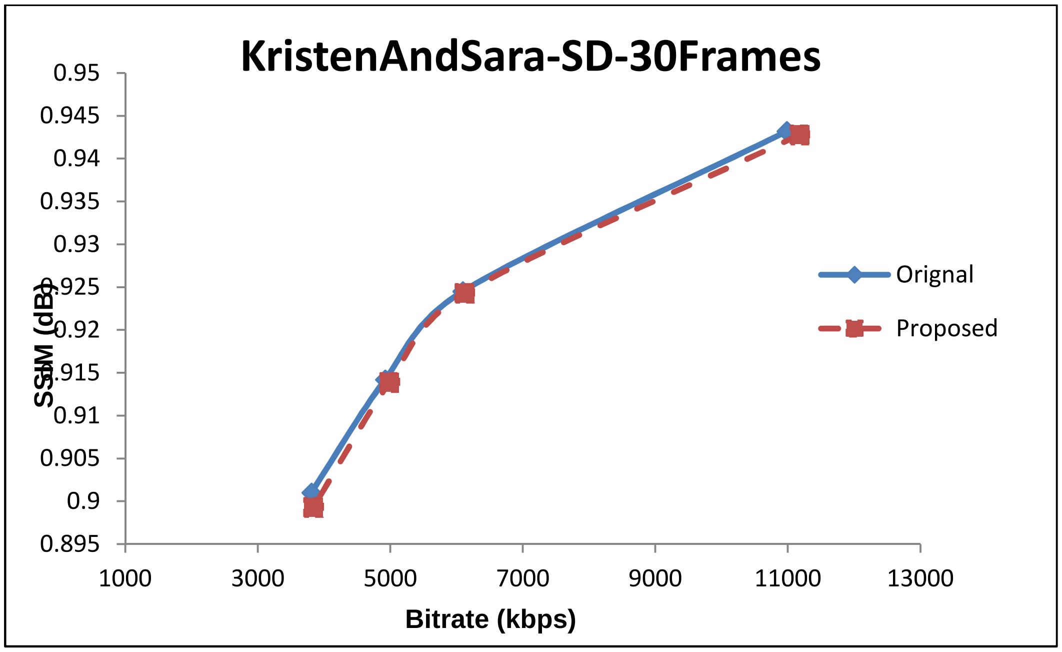


Figure 4-34 SSIM vs. bitrate for KristenAndSara

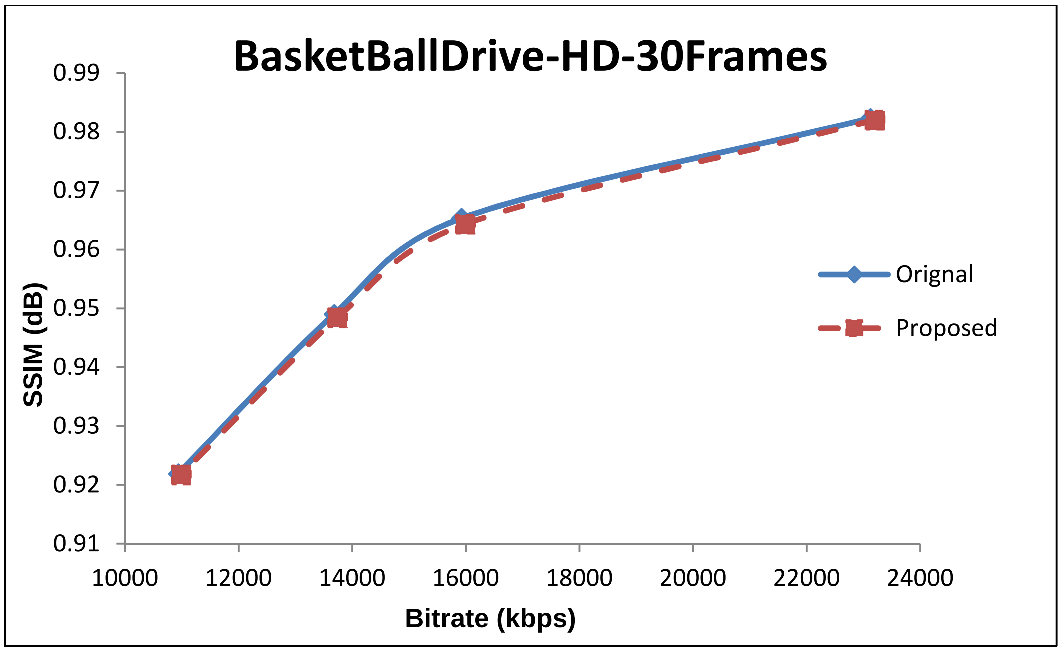


Figure 4-35 SSIM vs. bitrate for BasketBallDrive

4.9 Percentage Decrease in Encoding Time

Figures 4-36 to 4-40 show 25-39% decrease in encoding time with the proposed fast adaptive early termination algorithm as compared to the original HM13.0 algorithm.

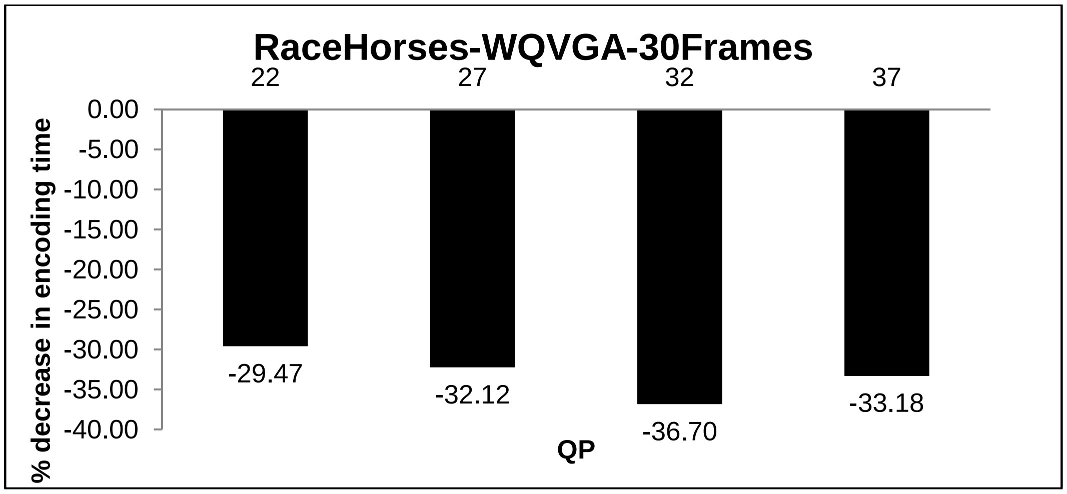


Figure 4-36 % decrease in encoding time vs. quantization parameter for RaceHorses

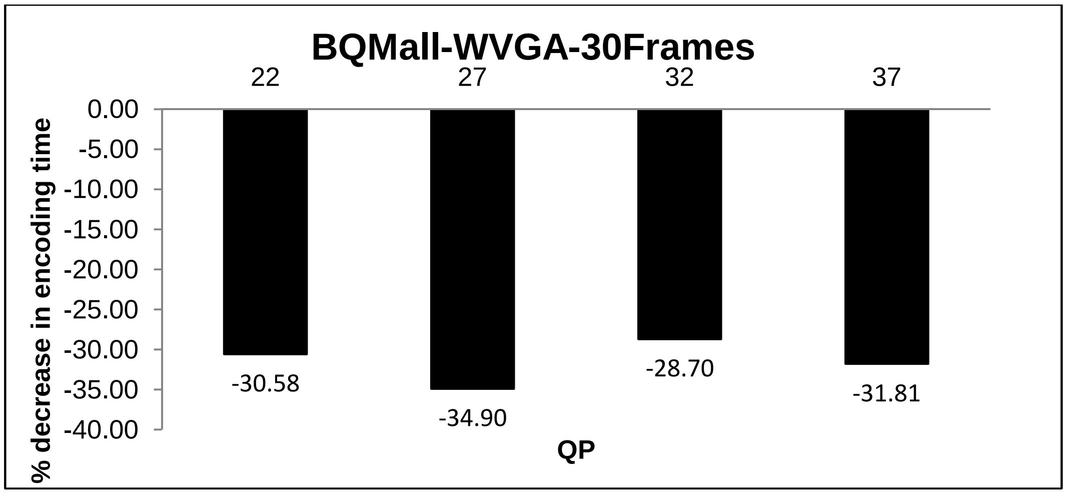


Figure 4-37 % decrease in encoding time vs. quantization parameter for BQMall

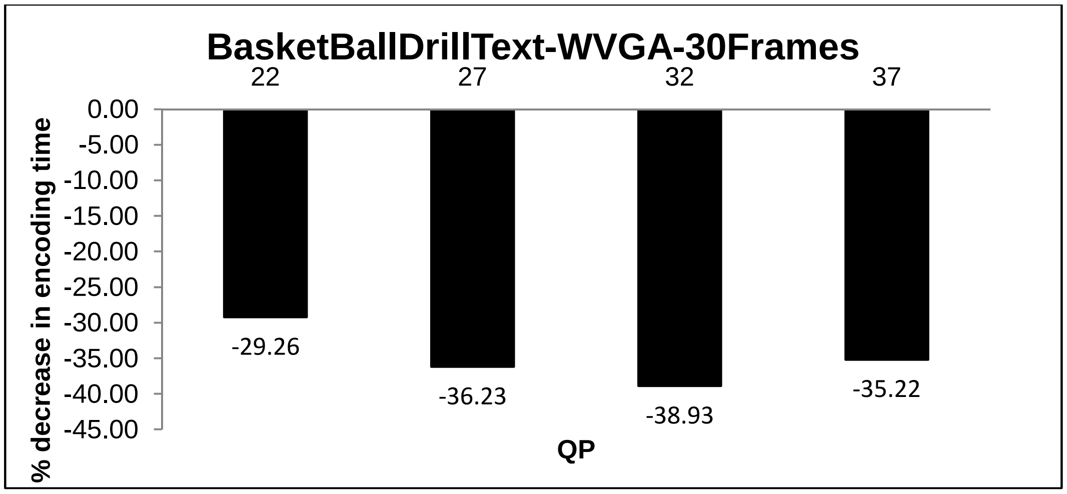


Figure 4-38 % decrease in encoding time vs. quantization parameter for BasketBallDrillText

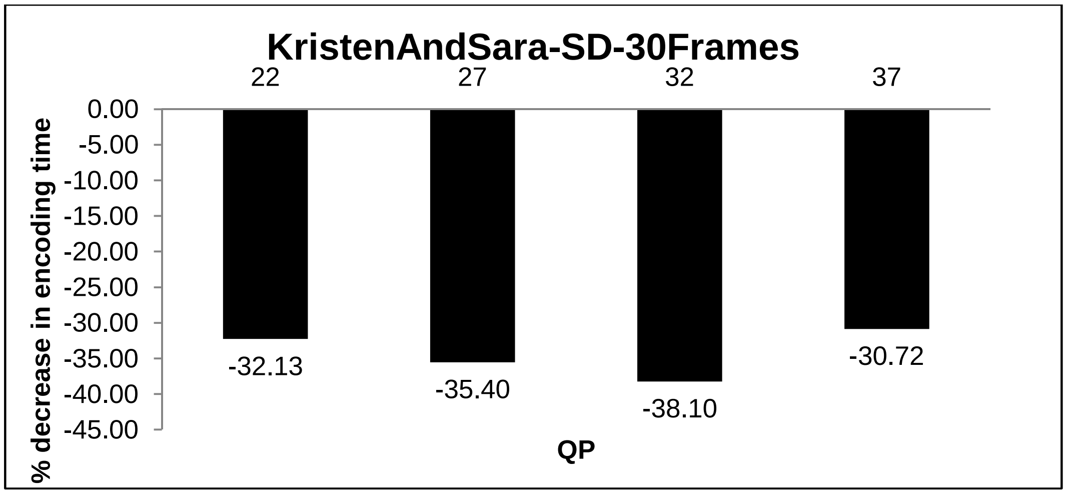


Figure 4-39 % decrease in encoding time vs. quantization parameter for KristenAndSara

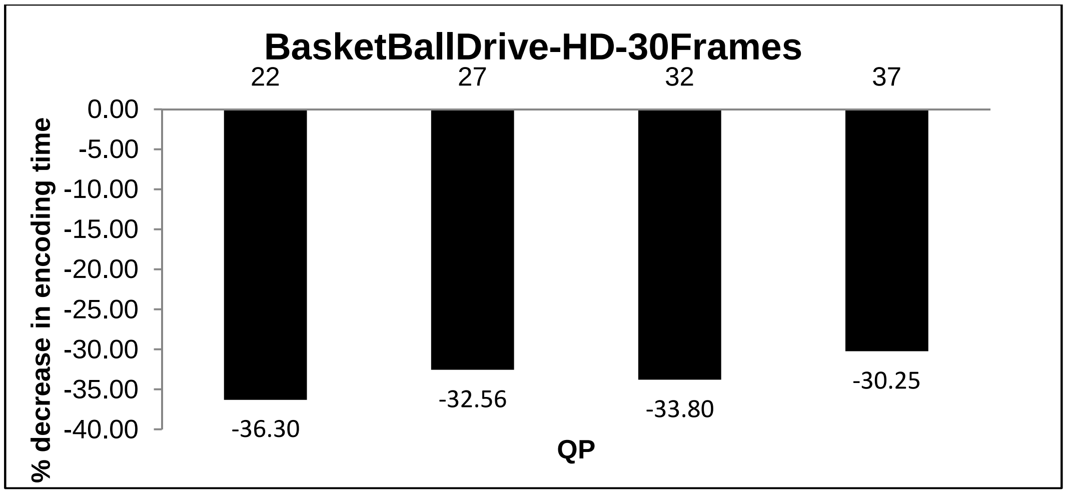


Figure 4-40 % decrease in encoding time vs. quantization parameter for BasketBallDrive

4.10 Summary

In this chapter, various results and graphs are explored with and without implementation of fast adaptive early termination algorithm using various metrics such as encoding time, BD-PSNR, BD-bitrate, bitstream size and SSIM. In chapter 5, conclusions and future work are discussed.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis a fast adaptive termination inter-mode decision algorithm is proposed to reduce the computational complexity of the HEVC encoder, which includes three strategies, i.e., Early SKIP mode decision, prediction size correlation based mode decision and RD cost correlation based mode decision. The results of comparative experiments demonstrate that the proposed algorithm can effectively reduce the computational complexity (encoding time) by 25-40% on average as compared to the HM 13.0 encoder [23], while only incurring a negligible loss of coding efficiency i.e. 0.1%-0.3% decrease in PSNR and 0.03%-0.15% decrease in SSIM for different values of quantization parameter based on various standard test sequences. The results of simulation also demonstrates negligible increase in bitrate i.e. 1%-3% as compared to the original HM13.0 software and 0.31 dB-0.51 dB decrease in BD-PSNR and 7 kbps-12 kbps increase in BD-bitrate.

5.2 Future Work

There are many other ways to explore fast inter-prediction, one of these is the CU decision early termination algorithm in which CU size decision is terminated depending upon adaptive RD cost calculation. Integrating for the PU mode-decision and CU size decision algorithm can result in higher complexity reduction.

Neural networks [47] can also be used for mode decision for fast inter-prediction. Networks can be trained with a number of test sequences and subsequently these networks can be used for mode decision in which the necessity of calculating RD cost for different modes can be eliminated.

Similar algorithms can be developed for fast intra-prediction in which RD cost of the different modes in intra-prediction are explored, and depending upon adaptive threshold, mode decision can be terminated resulting in less encoding time and reduced complexity.

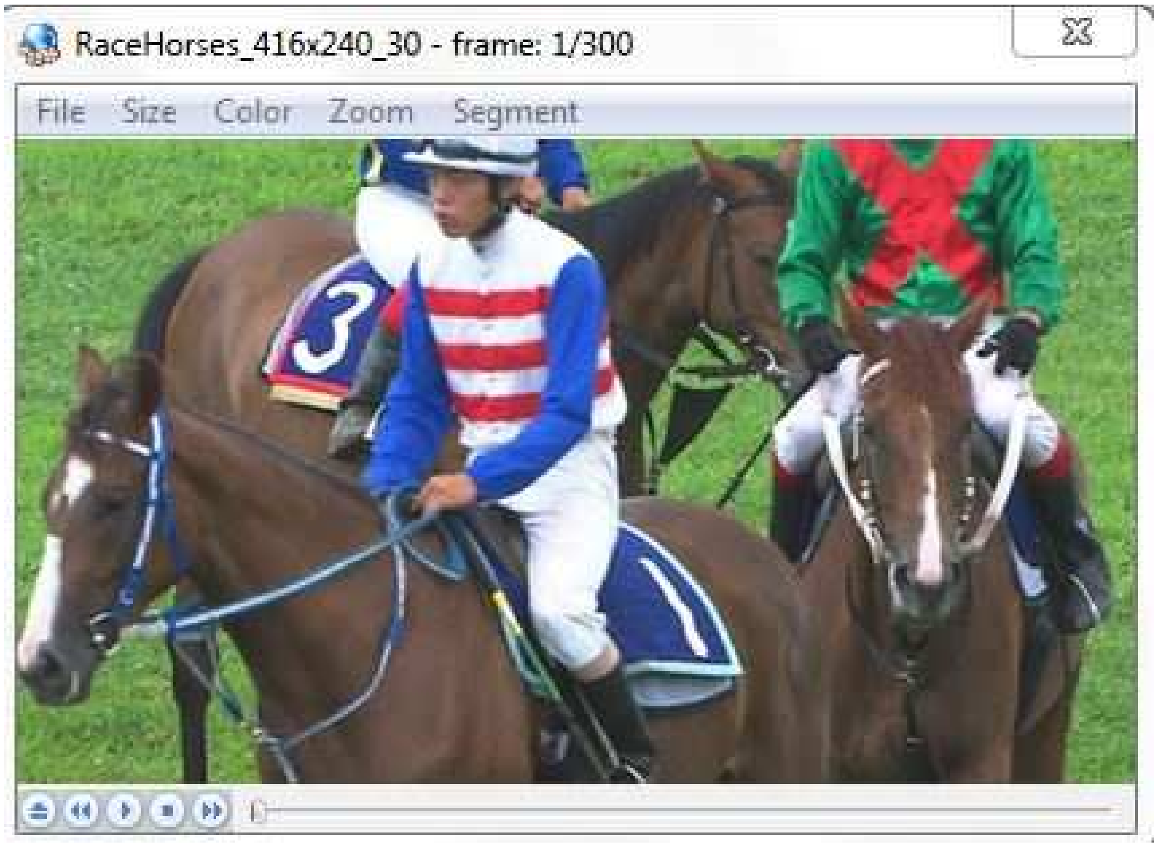
Bayesian decision [48] rule can be applied to calculate the CU size and then this information can be combined with the proposed method to achieve further encoding time gains.

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

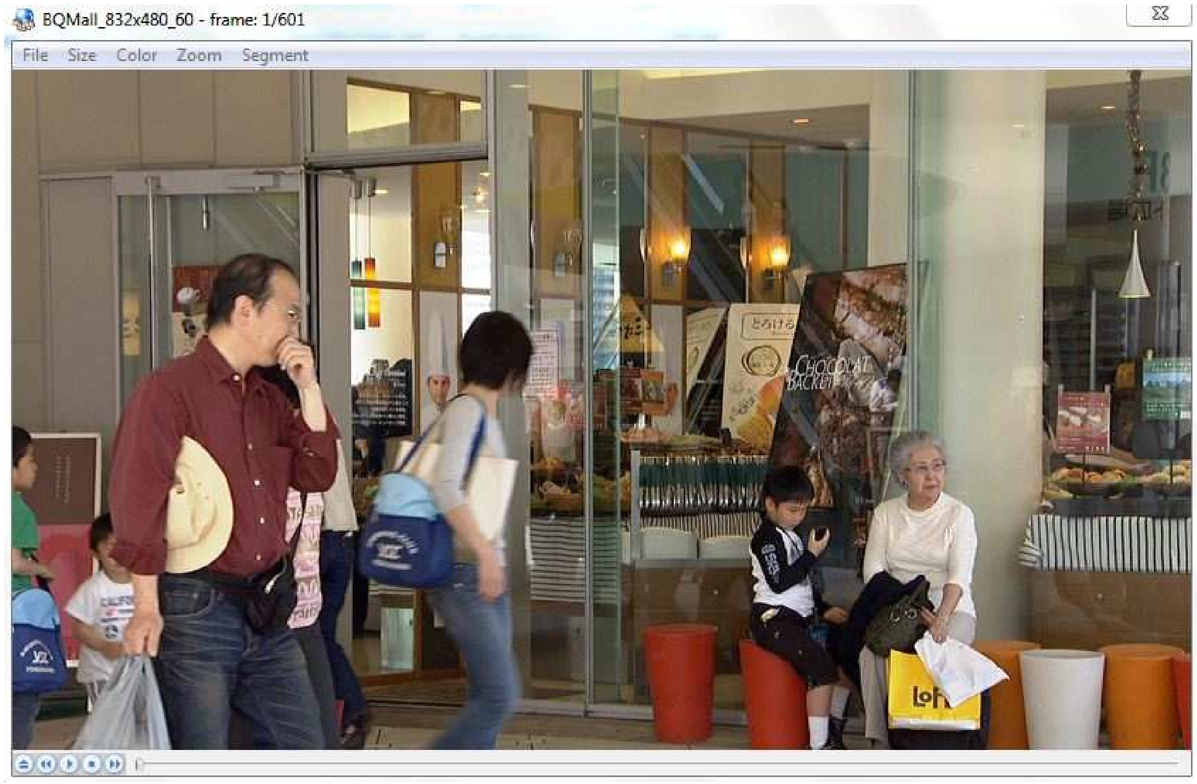
Appendix A

Test Sequences [22]

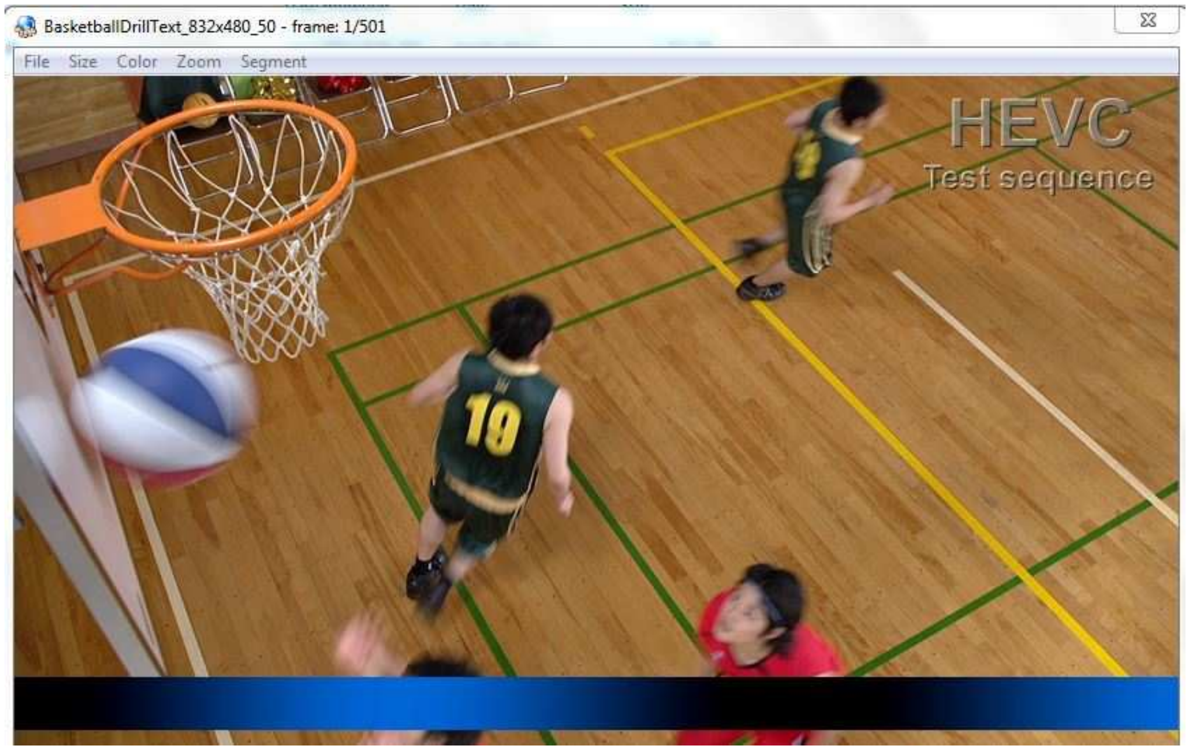
A.1 Racehorses



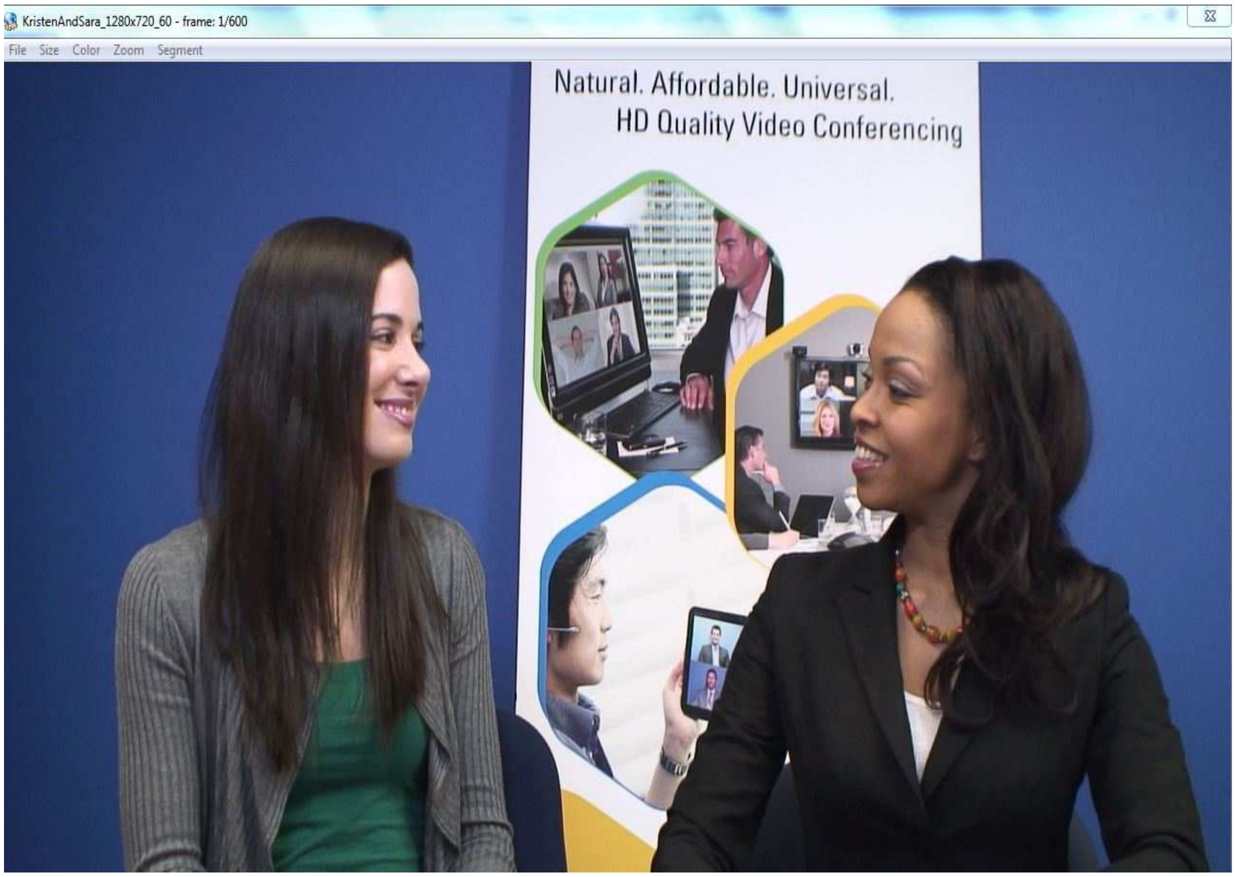
A.2 BQMall


BQMall_832x480_60 - frame: 1/601
File Size Color Zoom Segment

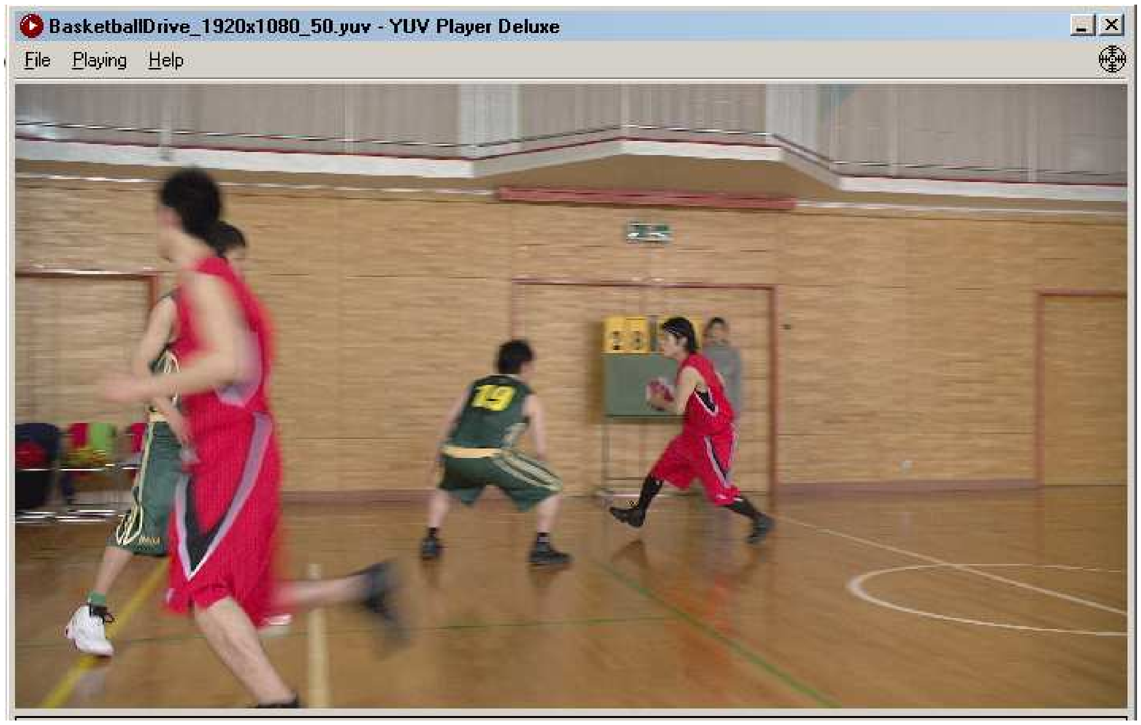
A.3 BasketBallDrillText


BasketballDrillText_832x480_50 - frame: 1/501
File Size Color Zoom Segment
HEVC
Test sequence

A.4 KristenAndSara


KristenAndSara_1280x720_60 - frame: 1/600
File Size Color Zoom Segment
Natural. Affordable. Universal.
HD Quality Video Conferencing

A.5 BasketBallDrive



Appendix B

Test Conditions

The code revision used for this work is revision HM13.0 [23]. The work was done using an Intel Core 5 with Microsoft Windows 8 64bit version running with 8 GB RAM at a speed of 2.3GHz.

Appendix C

BD- PSNR and BD-bitrate [24] [25]

BD-PSNR (Bjontegaard – PSNR) and BD-bit rate (Bjontegaard – bit rate) metrics are used to compute the average gain in PSNR and the average per cent saving in bit rate between two rate-distortion graphs respectively and is an ITU-T approved metric [24]. This method was developed by Bjontegaard and is used to gauge compression algorithms from a visual aspect in media industry and referenced by many multimedia engineers. The MATLAB code is available online [25].

```
function avg_diff = bjontegaard(R1,PSNR1,R2,PSNR2,mode)

%BJONTEGAARD    Bjontegaard metric calculation
%   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
%
%   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

% 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
        min_int = min([lR1; lR2]);
        max_int = max([lR1; 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
        min_int = min([PSNR1; PSNR2]);
        max_int = max([PSNR1; 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);

        % find avg diff
        avg_exp_diff = (int2-int1)/(max_int-min_int);
        avg_diff = (exp(avg_exp_diff)-1)*100;
end
```

Appendix D

Acronyms

AVC	Advanced Video Coding
AMVP	Advanced Motion Vector Prediction
BD	Bjontegaard Delta
CABAC	Context Adaptive Binary Arithmetic Coding
CB	Coding Block
CBF	Coding Block Flag
CFM	CBF Fast Mode
CTU	Coding Tree Unit
CTB	Coding Tree Block
CU	Coding Unit
DCT	Discrete Cosine Transform
DST	Discrete Sine Transform
ECU	Early Coding Unit
ESD	Early Skip Detection
ET	Early Termination
FDIS	Final Draft International Standard
HEVC	High Efficiency Video Coding
HM	HEVC Test Model
ISO	International Standards Organization
ITU	International Telecommunications Union
JCT-VC	Joint Collaborative Team on Video Coding
MC	Motion Compensation
ME	Motion Estimation
MPEG	Moving Picture Experts Group
MPM	Most Probable Modes
NAL	Network Abstraction Layer

PB – Prediction Block

POTS – Plain old telephone service

PSNR – Peak Signal to Noise Ratio

PU – Prediction Unit

QP – Quantization Parameter

RDOQ – Rate Distortion Optimization Quantization

RMD –Rough Mode Decision

SATD –Sum of Absolute Transform Differences

SD – Standard Definition

SSIM – Structural Similarity

TU – Transform Unit

URQ – Uniform Reconstruction Quantization

VCEG – Video Coding Experts Group

VPS – Video Parameter Set

WQVGA – Wide Quarter Video Graphics Array

WVGA – Wide Video Graphics Array

References

[1] G. Sullivan et al, “Overview of the high efficiency video coding (HEVC) standard”, IEEE Transactions on Circuits and Systems for Video Technology, vol 22, no. 12, pp 1649-1668, Dec. 2012.

[2] Video Codec for Audiovisual Services at px64 kbit/s, ITU-T Rec. H.261, version 1: Nov. 1990, version 2: Mar. 1993.

[3] Video Coding for Low Bit Rate Communication, ITU-T Rec. H.263, Nov. 1995 (and subsequent editions).

[4] Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to About 1.5 Mbit/s—Part 2: Video, ISO/IEC 11172-2 (MPEG-1), ISO/IEC JTC 1, 1993.

[5] Coding of Audio-Visual Objects—Part 2: Visual, ISO/IEC 14496-2 (MPEG-4 Visual version 1), ISO/IEC JTC 1, Apr. 1999 (and subsequent editions).

[6] Generic Coding of Moving Pictures and Associated Audio Information- Part 2: Video, ITU-T Rec. H.262 and ISO/IEC 13818-2 (MPEG 2 Video), ITU-T and ISO/IEC JTC 1, Nov. 1994.

[7] Advanced Video Coding for Generic Audio-Visual Services, ITU-T Rec. H.264 and ISO/IEC 14496-10 (AVC), ITU-T and ISO/IEC JTC 1, May 2003 (and subsequent editions).

[8] T. Wiegand et al, “Overview of the H.264/AVC video coding standard”, IEEE Transactions on Circuits and Systems for Video Technology, vol.13, no.7, pp. 560-576, March 2003.

[9] MPL Website: http://www-ee.uta.edu/dip/

[10] C. Fogg, “Suggested figures for the HEVC specification”, ITU-T/ISO/IEC Joint Collaborative Team on Video Coding (JCT-VC) document JCTVC- J0292r1, July 2012.

[11] B. Bross et al, “HM9: High Efficiency Video Coding (HEVC) Test Model 9 Encoder Description”, JCTVC-K1002v2, October 2012. [Online]. Available: http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=6807

[12] B. Bross et al, “High Efficiency Video Coding (HEVC) text specification draft 9 (SoDIS)”,

JCTVC-K1003v13, October 2012. [Online]. Available: http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=6803

[13] F. Bossen, B. Bross, K. Sühring, and D. Flynn, "HEVC Complexity and Implementation Analysis," IEEE Trans. on CSVT, vol.22, no.12, pp.1685-1696, Dec. 2012.

[14] I. Richardson, “The H.264 Advanced Video Compression Standard", Wiley, 2010.

[15] Basics of video: http://lea.hamradio.si/~s51kq/V-BAS.HTM

[16] S. Lui et al, “Video Prediction Block Structure and the Emerging High Efficiency Video Coding Standard”, IEEE proceedings on Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific, pp. 1-4, 2012.

[17] G. Sullivan et al, “Efficient quadtree coding of images and video", IEEE Transactions on Image Processing, vol. 3, pp. 327-331, May 1994.

[18] P. Helle et al, “Block merging for quadtree-based partitioning in HEVC”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, pp. 1720-1731, May 2012.

[19] K. Choi et al, “Fast coding unit decision method based on coding tree pruning for high efficiency video coding”, Proc. SPIE Optical Engineering, vol. 51, 030502 , March 2012.

[20] “Editor's Proposed Draft Text Modifications for Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), Draft 2”, JVT-E022d2, Geneva, Switzerland, 9-17 October, 2002.

[21] G J. Sullivan et al,” Standardized Extensions of HEVC”, IEEE Journal of Selected topics in Signal Processing, Vol.7, No.6, pp.1001-1016, December 2013.

[22] HEVC test sequences: ftp://ftp.tnt.uni-hannover.de/testsequences

[23] HMX.X, HEVC code: http://hevc.kw.bbc.co.uk/svn/jctvc-a124/branches/

[24] G. Bjontegaard, “Calculation of average PSNR differences between RD-curves”, Q6/SG16, Video Coding Experts Group (VCEG), 2-4 April. 2001.

[25] BD metrics code - http://www.mathworks.com/matlabcentral/fileexchange/27798-bjontegaardmetric/content/bjontegaard.m

[26] HEVC software manual:

https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/branches/HM-9.2-dev/doc/software-manual.pdf

[27] C. E. Rhee et al, “A Survey of Fast Mode Decision Algorithms for Inter-Prediction and Their Applications to High Efficiency Video Coding”, IEEE Transactions on Consumer Electronics, vol 58, no. 4, pp 1375-1383, Dec. 2012.

[28] J. Vanne et al, “Efficient Mode Decision Schemes for HEVC Inter Prediction”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, Jan. 2014. (Accepted for publication in a future issue of this journal)

[29] K. Choi et al, “Coding tree pruning based CU early termination,” document JCTVC-F092, Torino, Italy, Jul. 2011.

[30] J. Yang et al, “Early SKIP detection for HEVC,” document JCTVC-G543, Geneva, Switzerland, Nov. 2011.

[31] R. H. Gweon et al, “Early termination of CU encoding to reduce HEVC complexity”, document JCTVC-F045, Torino, Italy, July 2011.

[32] F. Sampaio et al, “Motion Vectors Merging: Low Complexity Prediction Unit Decision Heuristic for the Inter-prediction of HEVC Encoders”, IEEE International Conference on Multimedia and Expo (ICME), pp. 657 – 662, Nov. 2012.

[33] JCT. Working Draft 3 of High-Efficiency Video Coding. JCTVCE603, 2011.

[34] G. Sullivan et al, "Rate-Distortion Optimization for Video Compression", IEEE Signal Processing Magazine, vol. 15, pp.74-90, Nov. 1998.

[35] A. Lee et al ,"An efficient inter prediction mode decision method for fast motion estimation in HEVC", International Conference on ICT Convergence (ICTC), pp. 502- 505, 2013.

[36] D. T. Hoang et al, "Rate-Distortion Optimizations for Motion Estimation in Low-Bitrate Video Coding", IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, pp. 147-148, Aug. 1998.

[37] S. Kim at el, “ Fast mode decision algorithm for inter-layer coding in scalable video coding”, IEEE Trans. Consumer Electronics, vol. 55, no. 3, pp. 1572-1580, Aug. 2009.

[38] L. Sen et al, “ Adaptive inter-mode decision for HEVC jointly utilizing inter-level and spatio-temporal correlations”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, Mar. 2014. (Accepted for publication in a future issue of this journal)

[39] L. F. Chen et al, “Model-based early termination scheme for H.264/AVC inter prediction” Proceedings IEEE ICASSP, pp. 597-600, Apr. 2009.

[40] L. Shen et al, “Low-complexity mode decision for MVC”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 21, pp. 837-843, June 2011.

[41] K. R. Rao, D. N. Kim and J. J. Hwang, “Video coding standards”, Springer 2014.

[42] HEVC: http://www.apsipa2013.org/wp-content/uploads/2013/09/Tutorial_8_NextGenerationVideoCoding_Part_2.pdf

[43] JCT-VC documents: http://www.itu.int/en/ITU-T/studygroups/2013-2016/16/Pages/video/jctvc.aspx

[44] Z. Wang et al, “Image quality assessment: From error visibility to structural similarity”, IEEE Transactions on Image Processing, vol. 13, pp. 600-612, April 2004.

[45] SSIM code: https://ece.uwaterloo.ca/~z70wang/research/iwssim/

[46] X. Li et al, “Rate-Complexity-Distortion evaluation for hybrid video coding”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 21, pp. 957 - 970, July 2011.

[47] D.P. Kumar, “Intra Frame Luma Prediction using Neural Networks in HEVC”, website: http://www-ee.uta.edu/Dip/Courses/EE5359/Dilip_Thesis_Document.pdf, Thesis, University of Texas at Arlington, UMI Dissertation Publishing, May, 2013.

[48] X. Shen et al, “Fast coding unit size selection for HEVC based on Bayesian decision rule”, IEEE Picture Coding Symposium (PCS), pp. 453-456, May 2012.

[49] Special issue on emerging research and standards in next generation video coding, IEEE Transactions on Circuits and Systems for Video Technology (CSVT), vol. 22, pp. 1646-1909, Dec. 2012.

[50] Special issue on emerging research and standards in next generation video coding, IEEE Transactions on Circuits and Systems for Video Technology (CSVT), vol. 23, pp. 2009-2142, Dec. 2013.

[51] IEEE Journal of Selected Topics in Signal Processing, vol.7, pp. 931-1151, Dec. 2013.

[52] H. Zhang et al, “Fast Intra Mode Decision for High Efficiency Video Coding (HEVC)”, IEEE Transactions on Circuits and Systems for Video Technology (CSVT), vol. 24, pp. 660-668, Apr. 2014.

Biographical Information

Kushal Shah was born in Vapi, Gujarat, India in 1988. After completing his schooling at Mother of Hope School, Vapi in 2003, he went on to obtain his Diploma in Electronics and Communication from K.B.P polytechnic from 2003 to 2006. From 2006 to 2009, he pursued his Bachelors of Engineering in Electronics and Telecommunication from University of Pune, India. After that he worked for 2 years and 6 months in IBM India Private Limited and L&T InfoTech as Software Engineer in India.

He joined University of Texas at Arlington to pursue his M.S in Electrical Engineering in fall 2012. This was around the time he joined the Multimedia Processing Lab. He worked as Software Intern in NVidia Corporation for summer and fall 2013 and subsequently will join Apple Inc., Cupertino after he graduates.