IMPLEMENTATION of FAST RESIDUAL QUADTREE CODING and FAST INTRA PREDICTION in HIGH EFFICIENCY VIDEO CODING

by

SAPNA VASUDEVAN

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 2013

Acknowledgements

First of all, I would like to thank Dr. K. R. Rao from the bottom of my heart for being a guide, mentor and a constant source of encouragement throughout my thesis. I would like to thank Dr. Alan Davis and Dr. Ioannis Schizas for serving on my committee.

I would also like to thank Dr. Zhan Ma, Senior Standards Researcher, Samsung Research America Dallas and Dr. Chuo Hao Yeo, Lab Head, Signal Processing Department, Institute of Infocomm Research, Singapore for promptly replying to my queries.

I would be remiss if I do not acknowledge and give full credit to the author of Bjontegaard metric calculation program, Giuseppe Valenzsie of Politecnico di Milano, which helped me compare the quality of my proposed method to the existing codebase. This metric is now used for calculation of video metrics.

I would also like to thank my MPL lab-mates: Dilip Prasanna Kumar, Nayana Parashar, Abhijith Jagannath and Kushal Shah for providing valuable inputs throughout my research.

Last, but not the least, I would like to acknowledge my family especially Ashwin, my husband without whose support and encouragement I would not be here.

November 22, 2013

Abstract

IMPLEMENTATION of FAST RESIDUAL QUADTREE CODING and FAST INTRA PREDICTION in HIGH EFFICIENCY VIDEO CODING

Sapna Vasudevan, M.S.

The University of Texas at Arlington, 2013

Supervising Professor: K.R. Rao

High Efficiency Video Coding, one of the latest video compression schemes proposed by Joint Collaborative Team on Video Coding (JCT-VC) which is a partnership between International Telecommunication Union Video Coding Experts Group (ITU T-VCEG) and International Organization for Standardization Moving Picture Experts Group (ISO/IEC MPEG), has been promoted to Final Draft International Standard (FDIS) on January 25, 2013. With three profiles: main, main10 and still-frame, it achieves far better compression ratio than already existing schemes, while maintaining visual quality.

However, this superior bitrate savings is achieved at the expense of huge computational complexity which in turn means higher encoding time. One of the aspects of encoding is intra prediction. A fast intra prediction method has been explored here. Residual quadtree (RQT) coding is another computation intensive step before intra prediction. Here, a fast RQT method has been combined with the fast intra prediction method to achieve better encoding time.

In this thesis, a decrease of about 50% in encoding time can be seen with combining the fast intra prediction method and fast RQT into a single codebase for encoding the raw YUV videos while maintaining the visual quality.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

1.1 Basics of Video Compression

A video is simply a collection of images. An image is a collection of pixels or pels. Every image is characterized by its height i.e. the number of pixels along the vertical, and its width, i.e. the number of pixels along the horizontal of the image. In addition to this, each pixel is characterized by its color and brightness.

When a collection of images stacked together are displayed quickly in succession, the user gets the feeling of movement of images which is called video. Essentially, a time factor is introduced by displaying the images in succession.

The meat of video compression lies in being able to transmit information from a collection of images by taking advantage of redundancies between the various images and within particular images. There are two types of redundancies exploited: spatial redundancies and temporal redundancies. Spatial redundancies exploit the fact that in a single image, neighboring pixels more or less tend to have the same information. Temporal redundancies exploit the fact that in a collection of images, there is very little change between images. This may not always be the case but it tends to be this way most of the time, a fact exploited by any compression scheme. [1]

A frame compressed using its own pixels as reference is calledan intra frame or I-frame. An I-frame is usually used as a reference frame for the compression of a collection of upcoming frames called inter frames or P-frames. A third type of frame called a bi-directional frame or B-frame is compressed using information from both P-frames and I-frames. This is shown in Fig 1-1

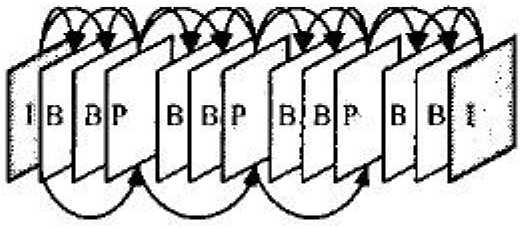


Figure 1-1 I, P and B frames [2]

1.2 Need for Video Compression

Uncompressed video occupies a huge amount of memory and thus will require enormous bandwidth to transmit the video from point to point. Therefore, there is a need for the development of more and more compression schemes that will reduce the memory footprint and bandwidth requirements of video transmission making everyday applications like online video streaming, video chatting etc. easier and economical in terms of memory.

High efficiency video coding (HEVC) is a fairly new video compression scheme (January, 2013). It has the advantage of about 50% bitrate savings when compared to its predecessor H.264/AVC [3] while maintaining the same visual quality.

1.3 Video Compression Standards

Fig 1-2 shows the evolution of the video coding standards [4] from the 80’s till today.

1.4 Thesis Outline

Chapter 2 details the HEVC compression scheme and gives an overview of the HEVC encoder. Key features in the encoding process are also described briefly. Chapter 3 describes the current intra prediction frame coding process and faster method to do this based on [6]. Residual quadtree coding is also discussed and a fast approach for the same is seen when it is combined with fast intra prediction. [7] Chapter 4 shows the results and conclusions of the proposed combined fast intra prediction with the fast residual quadtree coding scheme when compared to the unmodified HM9.2 [8] Chapter 5 describes further work in encoding that could give time improvement. In the Appendix sections, details of the test sequences used and the test environments are provided.

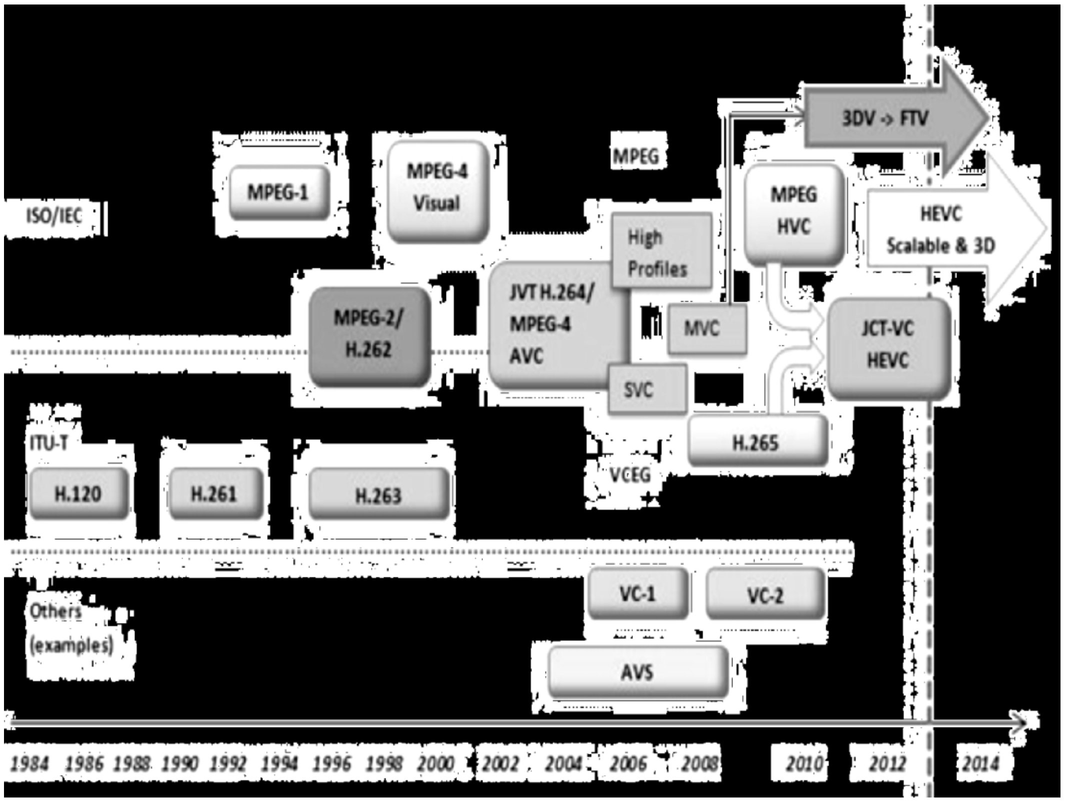


Figure 1-2 Evolution of compression standards [5]

Chapter 2

High Efficiency Video Coding

HEVC is the newest video compression standard in which 3 profiles – main (8 bit), main10 (10 bit) and still frame profile were standardized by the Joint Collaborative Team on Video Coding (JCT-VC) in January, 2013. [9-10] It has many advantages over previous standards like H.264/AVC. Some of the features under development for HEVC are 3-D video, and scalable video coding. As mentioned earlier, it has almost 50% more bit-rate savings over the previous H.264/AVC standard. This is not the result of optimizing a single step in the encoding process but a combined result of optimization of many processes together. As more and more emphasis is laid on video streaming and playback of HD and beyond HD quality, HEVC is a great improvement with respect to the previous standards. There are many new features in HEVC like wave front processing, tiles, dependent slices. which are introduced into the new standard. This advantage of HEVC comes at the price of high encoder complexity. The decoder is similar to the H.264/AVC decoder in complexity [10], and a diagram of the decoder is provided in Fig 2-1.

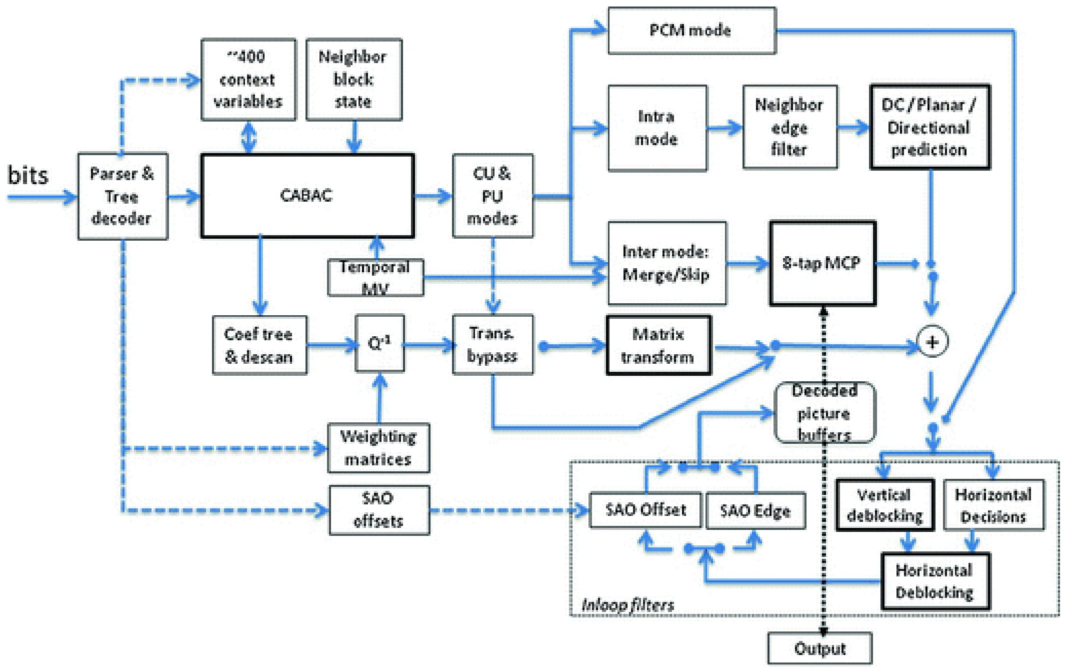


Figure 2-1 HEVC decoder block diagram [11]

2.1 Encoder description

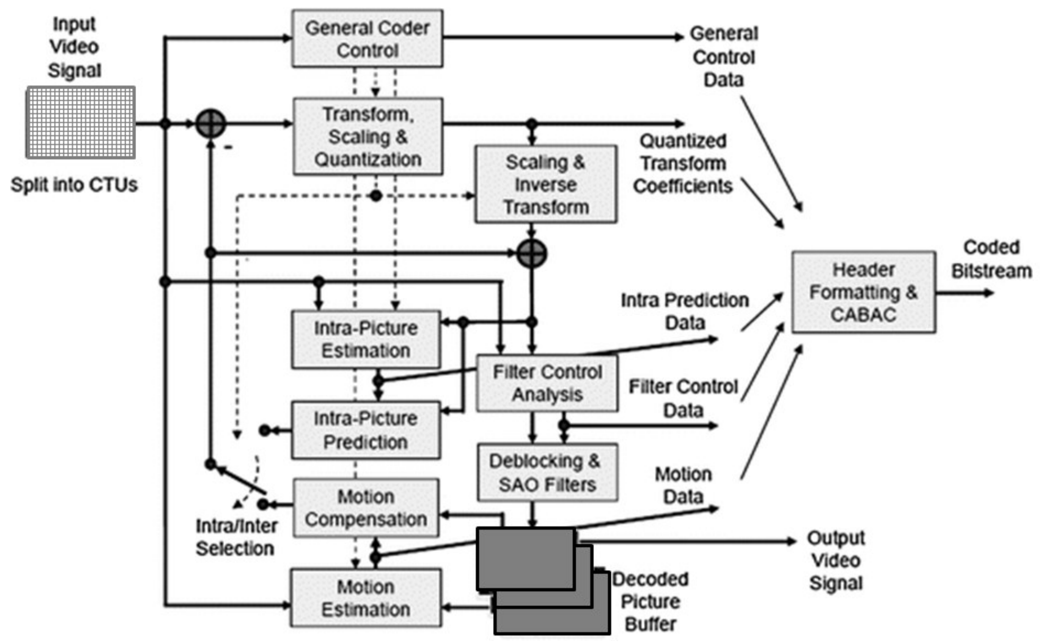


Figure 2-2 HEVC encoder block diagram [10]

Fig 2-2 shows the encoder block diagram. The raw input picture in the YC_bC_r with 4:2:0 sampling as shown in in Fig 2-3 at 8 bits per sample will be divided into coding tree units (CTUs). [10] Each CTU is divided into coding tree blocks (CTBs). The luma CTB can be of size 16x16 or 32x32 or 64x64. The chroma CTB can be of size 8x8 or 16x16 or 32x32. Each CTB is further split into coding blocks (CBs). One luma CB along with two chroma CBs make a coding unit (CU). This CU will be further split into prediction unit (PU) which will be further split into transform unit (TU). This process of splitting the CU to the TU is called quad tree syntaxing. [7,10] This is depicted in Figs 2-4 and 2-5 respectively.

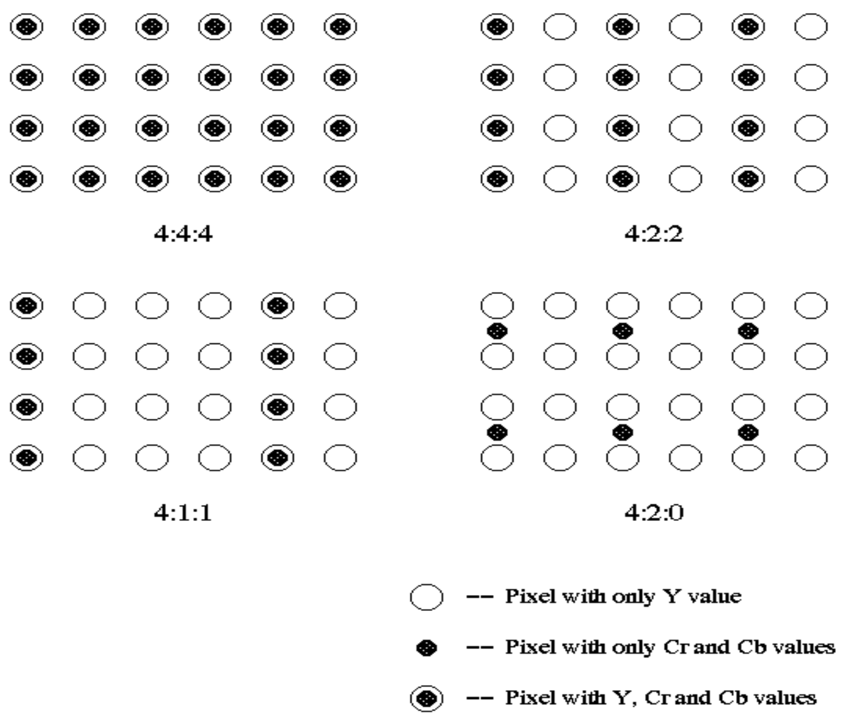

4:4:4
4:2:2
4:1:1
4:2:0
-- Pixel with only Y value
-- Pixel with only Cr and Cb values
-- Pixel with Y, Cr and Cb values

Figure 2-3 Chroma subsampling [12]

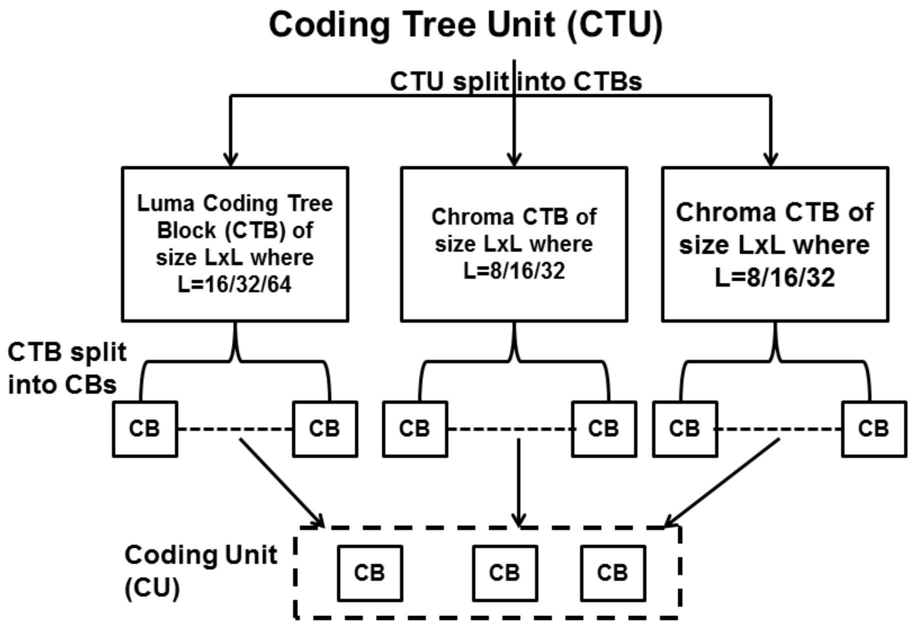

Coding Tree Unit (CTU)
CTU split into CTBs
Luma Coding Tree Block (CTB) of size LxL where L=16/32/64
Chroma CTB of size LxL where L=8/16/32
Chroma CTB of size LxL where L=8/16/32
CTB split into CBs
CB
CB
CB
CB
CB
CB
Coding Unit (CU)
CB
CB
CB

Figure 2-4 Formation of a CU [10]

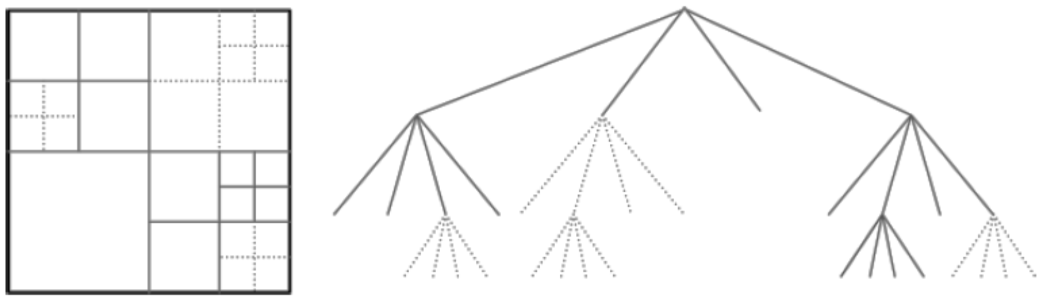


Figure 2-5 Left: CTB to CBs and TBs. Solid lines indicate CB boundaries and dotted lines indicate TB boundaries. Right: Corresponding quadtree [10]

The difference between the values in the original pixel and its prediction is called prediction residual. [10] This residual is transformed using integer like discrete cosine transform (DCT) for all block sizes except 4x4 intra block where an alternate transform related to discrete sine transform (DST) is used. [13] The control data from the residual needs to be shared between the encoder and decoder and this is placed into the header of the final bit-stream that will be sent. The residual itself will be transformed as mentioned above and then quantized and added to the final bit-stream. The decoder loop consists of inverse transform and scaling followed by filtering done by deblocking filter in the encoder block diagram shown in Fig 2-2. This reduces the error or drift between what the encoder predicts and what the decoder actually has. [10] From this loop, the encoder can get information regarding the motion compensation and intra prediction. The final bit-stream is encoded using context adaptive binary arithmetic coding (CABAC). [14]

2.2 Key Topics in the Encoding Process of the Video Coding Layer [10]

2.2.1 Motion vector signaling

Advanced motion vector prediction (AMVP) is used. This includes derivation of several most probable candidates based on data from adjacent prediction blocks (PBs) and the reference picture. A merge mode for motion vector coding is also used, allowing the inheritance of motion vectors from temporally or spatially neighboring PBs. Moreover, compared to AVC there is improved skipped and direct motion inference.

2.2.2 Motion compensation

Quarter-sample precision is used for the motion vectors. A filter with 7 taps (weights: -1, 4, -10, 58, 17, -5, 1) or 8 taps (weights: -1, 4, -11, 40, 40, -11, 4, 1) are used for interpolation of fractional-sample positions as shown in Fig 2-6. Similar to H.264/MPEG-4 AVC, [3] multiple reference pictures are used as shown in Fig 2-7. 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.

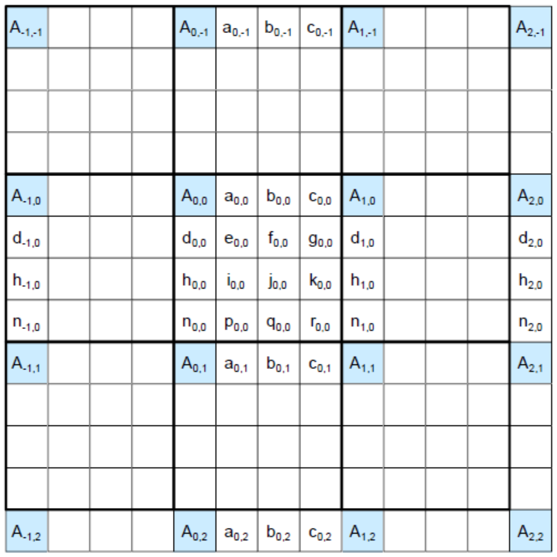


Figure 2-6 Integer and fractional sample positions for luma interpolation [10]

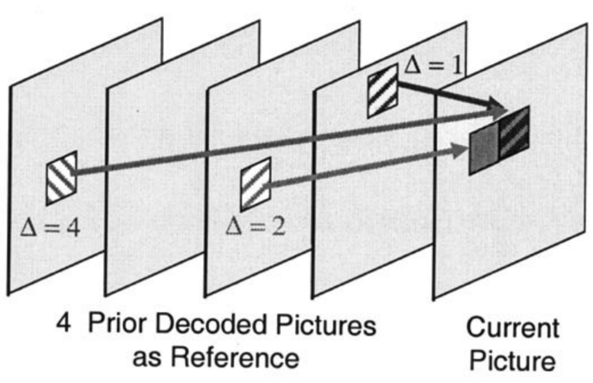


Figure 2-7 Multiple pictures used as reference for the current picture for motion compensation [3]

2.2.3 Quantization control

Uniform reconstruction quantization (URQ) is used in HEVC, with quantization scaling matrices supported for the various transform block sizes.

2.2.4 Entropy coding

A new and improved context adaptive binary arithmetic coding (CABAC) is used for the entropy coding of the bitstreams. This coding has improved speed, compression and requires less memory. This is heavily used in wavefront processing which is a new feature in HEVC. Instead of doing the normal CABAC reinitialization 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 as seen in Fig 2-8 and thus taking huge advantage of parallel processing.

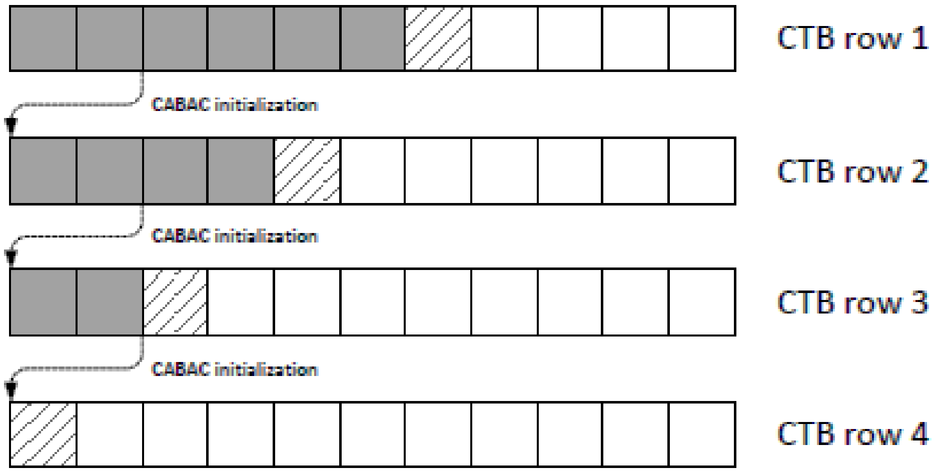


Figure 2-8 Example of waveform processing [15]

2.2.5 In-loop deblocking filtering

A deblocking filter is operated within the inter-picture prediction loop. However, the design is simplified with regard to its decision-making and filtering processes, and it is made more friendly to parallel processing. However, it relies on the same principles as the H.264/AVC [3] deblocking filter. It also differs from H.264/AVC in aspects that affect the complexity of the filtering process. Unlike H.264 which relies on the edge of a 4x4 block, HEVC deblocking filter relies on 8x8 edge block thus immediately reducing the number of filters samples by half. [15]

2.3 Key topics in high level syntax [10]

2.3.1 Parameter set structure

Information that needs to be shared between the encoder and decoder needs to be sent as overhead through a parameter structure. This information is useful for decoding several regions of the decoded video. The already existing concepts of sequence and picture parameters in H.264/AVC [3] are now augmented by video parameter set (VPS).

2.3.2 Network abstraction layer (NAL) syntax structure

Every syntax structure is placed in a NAL unit, which is based on its associated header. The purpose of the payload can be determined by the system.

2.3.3 Slices

A slice is a data structure that can be decoded independently from other slices in the same picture in terms of entropy coding, signal prediction and residual signal reconstruction. This is mainly used for recovery in the event of a data loss when it happens during transmission.

2.3.4 Tiles

The main purpose of this data structure is to take advantage of parallel processing rather than error resilience. They are independently decodable regions of a picture that have been encoded with some shared header information. They provide a coarse granularity level of parallelism and thus require no sophisticated thread level synchronization mechanism.

2.3.5 Wavefront Parallel Processing

Wavefront parallel processing is a key feature to take advantage of parallel processing. Here, each slice is divided into rows of CTUs as shown in Fig 2-8. Key decisions from previous rows aid the start of processing of the next row. They avoid the visual artifacts introduced by tiles and also are a better choice for higher compression ratio.

2.3.6 Dependent slices

A dependent slice is a data structure that associates data with a particular wavefront entry point or a tile in a separate NAL unit. This facilitates fragmented packetization with low latency and is a feature that is best used when low delay compression is required.

The focus of this thesis is mainly on intra prediction and residual quadtree coding. Both these topics are covered in Chapter 3.

Chapter 3

Intra Prediction and Residual Quadtree Coding

3.1 Intra Prediction Introduction

In order to generate prediction for a current block, the encoder will have information of decoded pixels in the row above this block and a column to the left of this block. Using this information, the encoder can predict the value of the current block and subsequently quantize and transform the residual for transmission. This is the basic idea of intra prediction. The word “intra” indicates that the considered frame uses only pixels within itself for the prediction process.

3.2 Intra Prediction in Detail

At the CU level, a decision will be made whether the prediction is going to be intra or inter. Intra prediction exploits the spatial redundancy of a frame. Frames predicted through intra prediction are called I-frames. I-frames use the boundary samples of decoded neighbor blocks above and to the left side. [10] Subsequent samples within the frame can be predicted from these values. There are a total of 33 directions in which intra prediction mode can be selected for square PU sizes ranging from 4x4 to 32x32. [6] In addition to these modes, there are 2 additional modes: DC and planar modes for the prediction. DC mode takes the average of the boundary value pixels and considers the surface to be flat whereas the planar mode considers the surface to be an amplitude surface and vertical and horizontal slope values are derived from the boundary values. These are shown in Fig 3-1. [10]

Intra_angular[k] represents the mode chosen. When k=0, the DC mode is chosen and k=1 implies the planar mode is chosen. The value of k can go from 2 to 34 for the remaining 33 modes. The angles are more dense near the horizontal and vertical to reflect the observed statistical prevalence of the angles. [6] If the chosen mode has a k between 2 and 17, then the pixels on the left column are used as the reference and when k is from 18 to 24, the pixels on the top are used as the reference. Due to the availability of a large number of modes, three most probable modes (MPM) are used. [8,10]

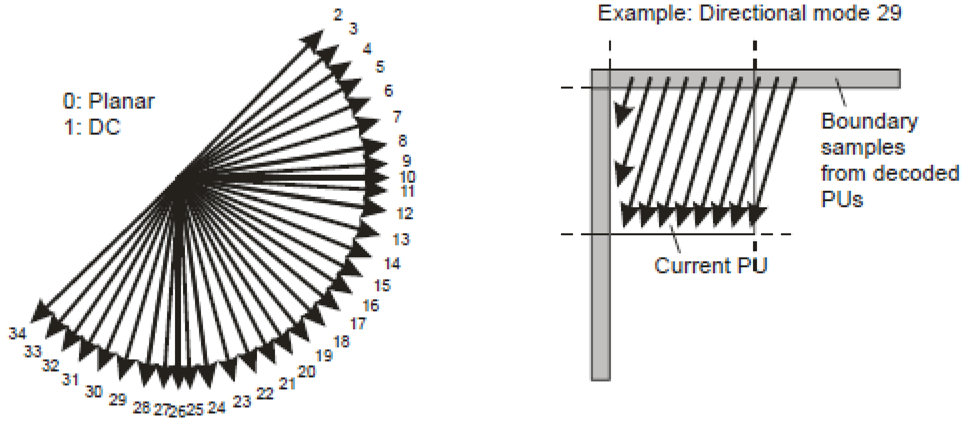


Figure 3-1 Left: Adaptive angular intra prediction (33 modes); Right: Example of mode 29 intra prediction [10]

The first two modes are initialized by the luma intra prediction of the above and left PBs, if available. Intra_DC is the mode used if any PB is unavailable. The PB above the luma CTB is always considered to be unavailable in order to avoid the need of a buffer. When the first two MPMs are not equal, the third MPM is set equal to Intra_Planar, Intra_DC or Intra_Angular [26] (vertical), according to which of these modes, in this order, is not a duplicate of one of the first two modes. When the first two MPMs are the same, and the first mode has the value Intra_Planar or Intra_DC, the second and third MPMs are assigned to be Intra_Planar, Intra_DC, or Intra_Angular [26], according to which of these modes, in this order, are not duplicates. When the first two MPMs are the same and the first mode has an Intra_Angular value, the second and third MPMs are chosen as the two angular prediction modes that are closest to the angle (i.e. the value of *k*) of the first mode. When the current luma intra prediction mode is one of three MPMs, only the MPM index is transmitted to the decoder to save on bits transmitted. Otherwise, the index of the current luma intra prediction mode excluding the three MPMs is transmitted to the decoder.

3.3 Residual Quadtree Coding

RQT coding is used to encode prediction residuals in both intra and inter CUs. As discussed earlier, blocks of various sizes have to be split into smaller blocks to proceed with the encoding process. The block with the least cost in terms of bits represented and least distortion needs to be chosen. Consider in a 64x64 CU, various possibilities of splitting this block exists as shown in Table 3-1. [16]

Table 3-1 Current problem-complexity and encoding time [16]

Size of a PB	Number of PBs in 64x64 CU	Number of modes to be tested in one 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			19000

An example, with 35 intra prediction modes and 11900 possibilities of splitting a single CU, it is no small feat for the encoder to perform this computation and come up with the best mode i.e. one that costs least, has minimum distortion and bitrate [16]. An important factor considered in this thesis is encoding time which can be saved if one is able to skip certain computations of residual quadtree coding (RQT) and intra prediction mode selection and arrive at the same decision with little or no loss in peak signal to noise ratio (PSNR). Keeping this goal in mind, certain modifications regarding the intra mode selection process [6] and RQT [7] a combined in the existing HM9.2 [8] to achieve the results described in Chapter 4.

3.5 Proposed solution-Fast RQT followed by Fast Intra Prediction

There are a large number of papers available on making the intra prediction process faster. [17-27] A three step method proposed by Zhang and Ma [6] was implemented in an older version of HM. In order to compare results between the unmodified HM9.2 and the proposed method, first, code related to fast intra prediction was ported from HM7.0 [28] to HM9.2. Subsequently, modifications were made to this ported code in order to reduce the RQT steps.

3.5.1 Details of three step fast intra prediction [6]

3.5.1.1 Hadamard Transform of 2:1 downsampled prediction residual

First, the residual block is 2:1 downsampled. Then a 4x4 or 2x2 Hadamard transform is applied on the downsampled block to derive the sum of the absolute transform difference (SATD) as shown in Fig 3-2. The SATD will be different for the different modes and a few modes with minimum SATD can be selected at this stage.

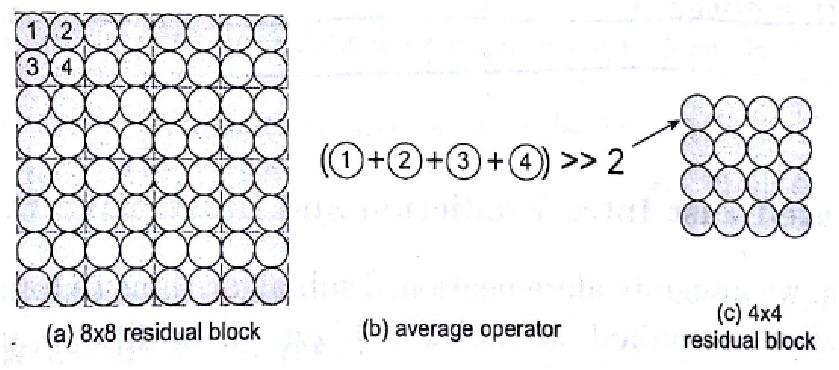


Figure 3-2 Downsampling by averaging operator [6]

3.5.1.2 Progressive Mode Search

Besides the Hadamard transform, a progressive mode search can be done to find a rough mode decision (RMD). The search range gets smaller and finally converges to the best location when the previously found best location becomes the new search area. The difference between two intra mode indices, m_i and m_j is $d=|m_i-m_j|$. The algorithm checks 8 equally spaced modes (adjacent modes are of distance 4). For the best 6 modes (least SATD), adjacent modes that are of distance 2 are checked. For the best 2 modes, adjacent modes of distance 1 are checked. This mode is combined with the modes selected from the upper and left CUs.

3.5.1.3 Early RDOQ Termination

In the third stage, there are M modes selected from the result of the second step. These are put into a group Ψ that go through the RDOQ process to get the best mode, m_{opt}. An early RDOQ termination is proposed for further encoder time reduction. For each intra mode $m \in \Psi$, its overall cost $J(m)$ as the combination of SATD cost and associated mode index bits consumption is calculated. Within Ψ there is a mode with minimal overall cost, J_{min}, defined as the rough best mode $m_{opt\ rough}$. If $m_{opt\ rough}$ is a Planar or a DC mode, all other modes in Ψ are skipped. If $m_{opt\ rough}$!= 0 or 1 , and $|m-m_{opt\ rough}| > 3$, such mode m is not used; Meanwhile, if $J(m) > \alpha J_{min}$, mode m will not be checked, $\alpha = 1.2$ is used. After such early termination procedure, all the remaining modes are checked by RDOQ.

3.5.2 Fast RQT step

As suggested in the literature, further gains in encoding time can be achieved by combining two or more fast methods together [17-27] This was the driving force behind exploring fast methods in areas besides intra prediction. RQT coding is used to encode prediction residuals. It is a great way to code a number of sub-blocks with the same intra prediction mode. With the number of intra modes so high, making a good joint decision on intra prediction mode is essential to save in encoding time.

After running multiple simulations on the unmodified code and the code on which fast intra prediction was ported, it was noticed that the unmodified version of HM9.2 extensively performed RQT on a number of modes and then subsequently, after determining the least costly mode, would again repeat the cost calculation on the best mode. In the proposed fast RQT, only the best is used for an extensive RQT. The best mode is the least costly of three modes along the MPM, if it is not in the least costly modes. Once the least costly mode was found, the process of recalculating the cost is completely avoided bringing further reduction in encoding time. In addition to this, the maximum depth of intra prediction was reduced from three to two as suggested by Tan et al., [7]

Chapter 4

Results and Conclusions

4.1 Test Sequences Used

During the duration of the thesis, five sequences of thirty frames each listed in Table 4-1 were used for comparing the performance of unmodified HM9.2 versus the HM9.2 with fast RQT and fast intra prediction added in, i.e the proposed HM. Appendix A shows one frame of each sequence. HM software manual is available at [29].

Table 4-1 Test sequences used

No.	**Sequence Name**	**Resolution**	**Type**
1.	RaceHorses	416x240	WQVGA
2.	BQMall	832x480	WVGA
3.	BasketballDrillText	832x480	WVGA
4.	KristenAndSara	1280x720	SD
5.	Johnny	1280x720	SD

4.2 Encoding time gain

The gain in encoding time on the proposed combined fast intra prediction and fast RQT HM9.2 is in the range of 55%-61% as seen in Figs 4-1 to 4-5 when compared to the unmodified HM9.2.

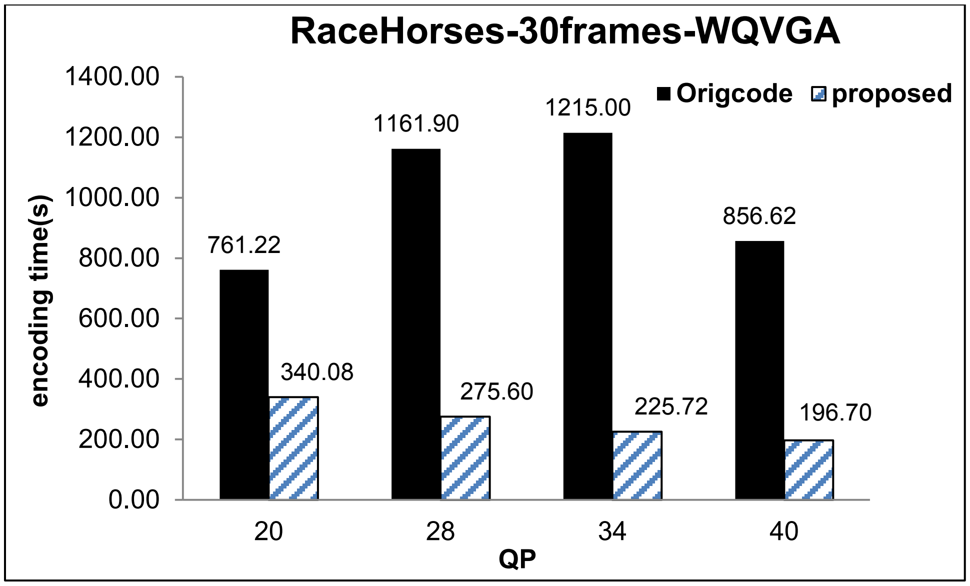


Figure 4-1 Encoding time vs. quantization parameter for Racehorses

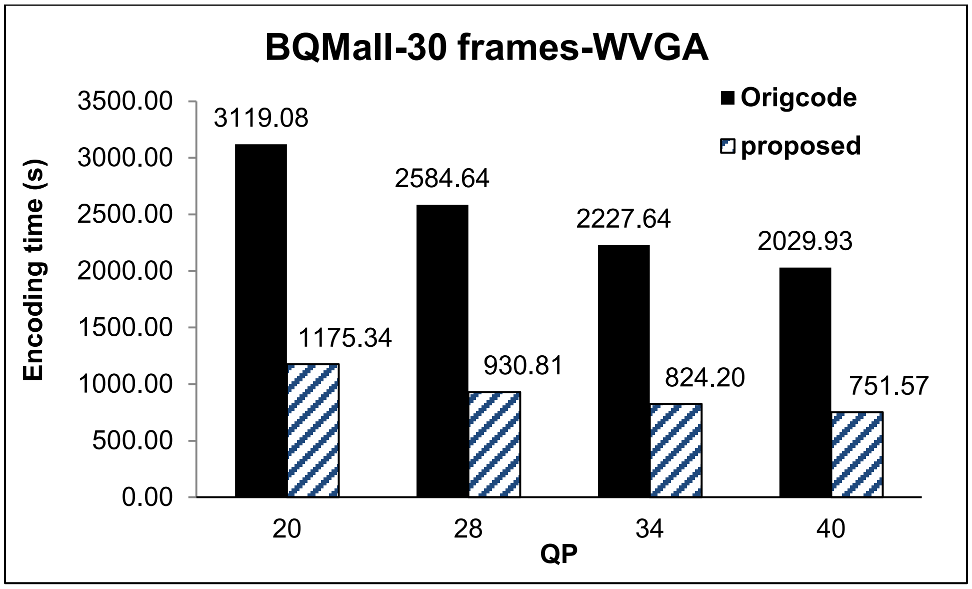


Figure 4-2 Encoding time vs. quantization parameter for BQMall

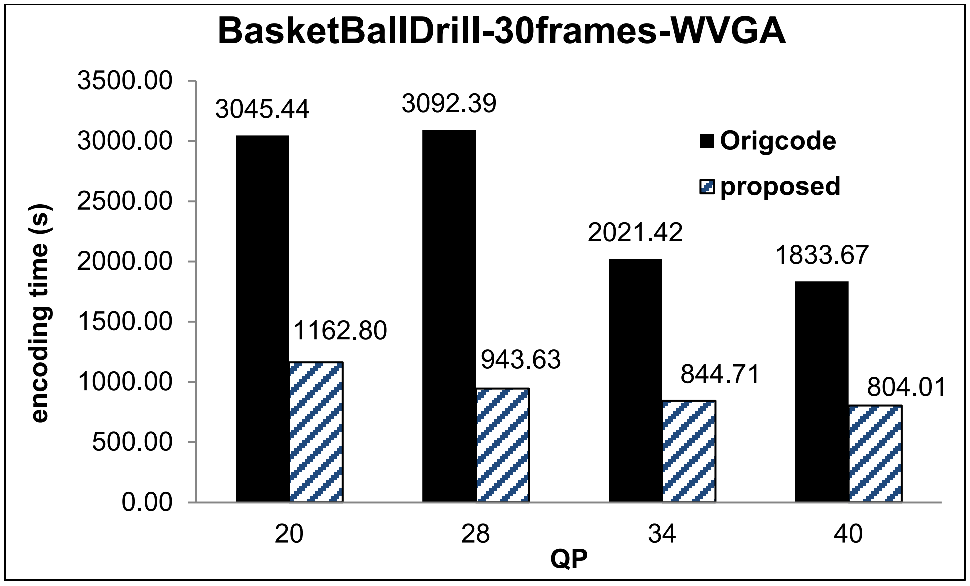


Figure 4-3 Encoding time vs. quantization parameter for BasketBallDrillText

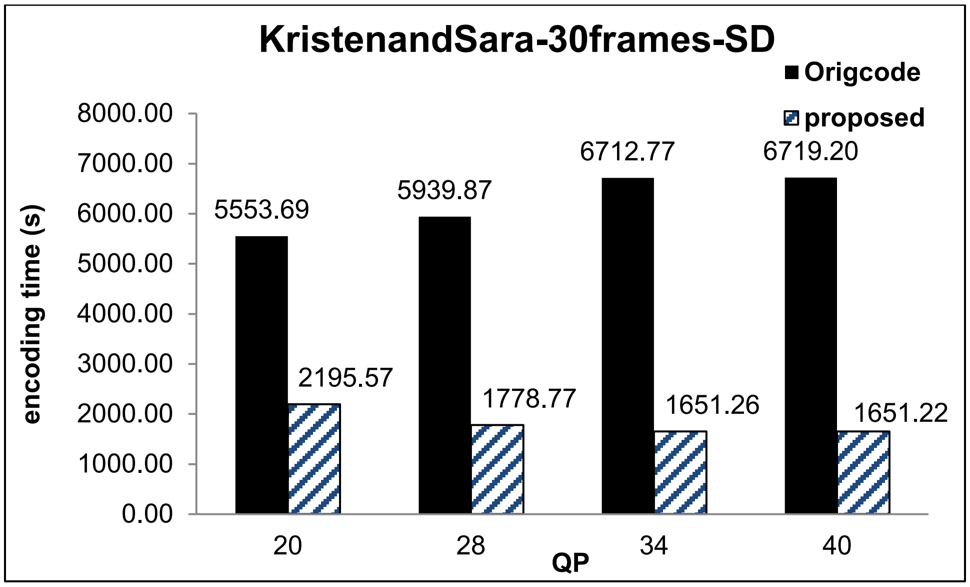


Figure 4-4 Encoding time vs. quantization parameter for KristenAndSara

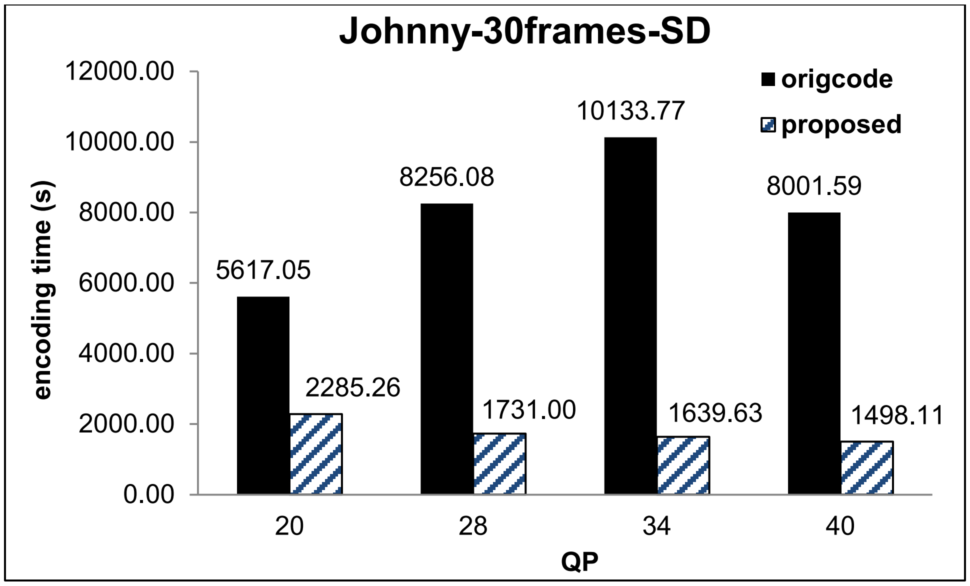


Figure 4-5 Encoding time vs. quantization parameter for Johnny

4.3 BD-PSNR loss

As seen in Figs 4-6 to 4-10, there is a drop in the PSNR (in the range of 0.1dB to 0.5db) using BD metrics for the proposed combined fast intra prediction and fast RQT HM9.2. The BD-PSNR is a curve fitting procedure based metric that works on rate and distortion of the video sequence. Though this metric does not take into account the complexity of the encoder, BD metrics tell a lot about the quality of the video sequence.[30-31] Ideally, BD-PSNR should increase and BD-bitrate should decrease.

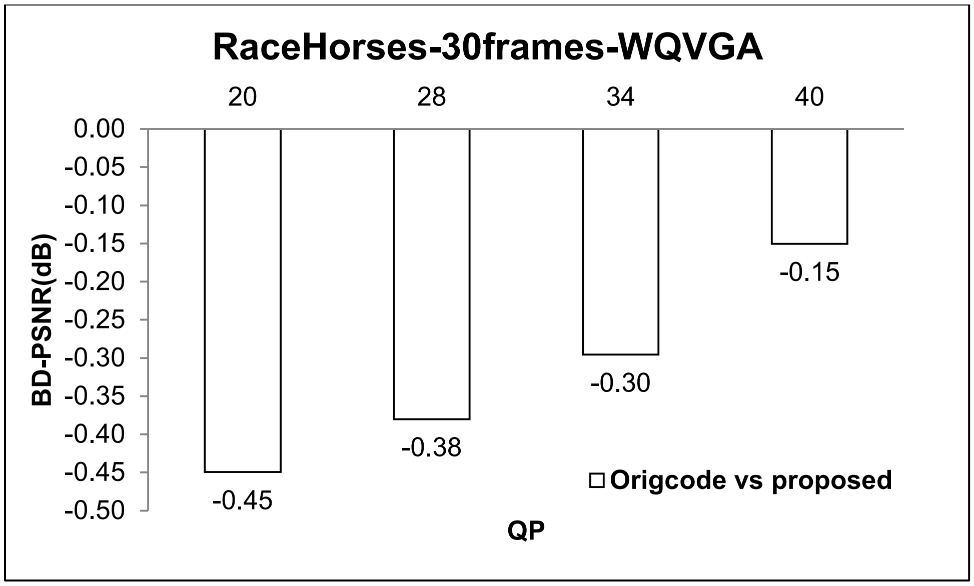


Figure 4-6 BD-PSNR vs. quantization parameter for RaceHorses

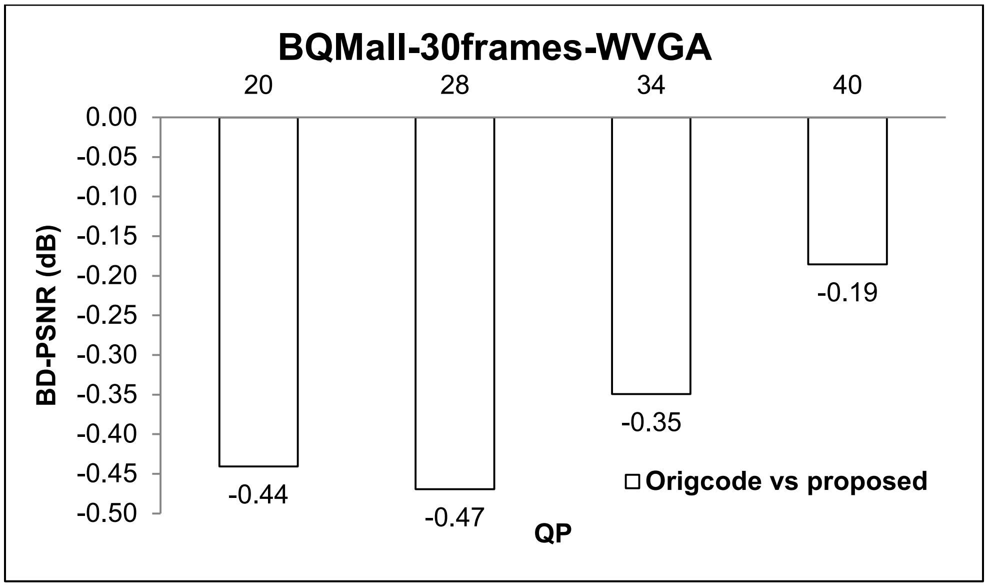


Figure 4-7 BD-PSNR vs. quantization parameter for BQMall

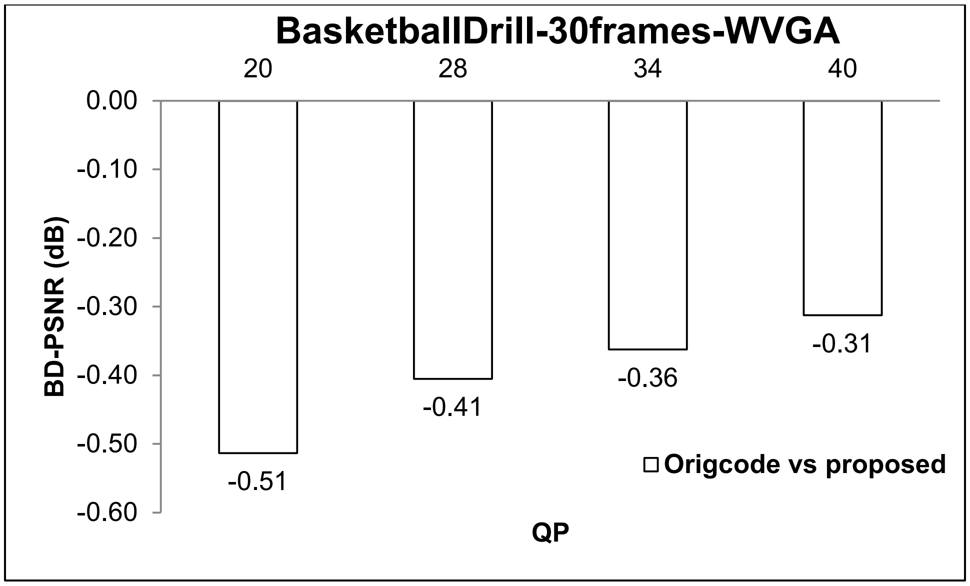


Figure 4-8 BD-PSNR vs. quantization parameter for BasketballDrillText

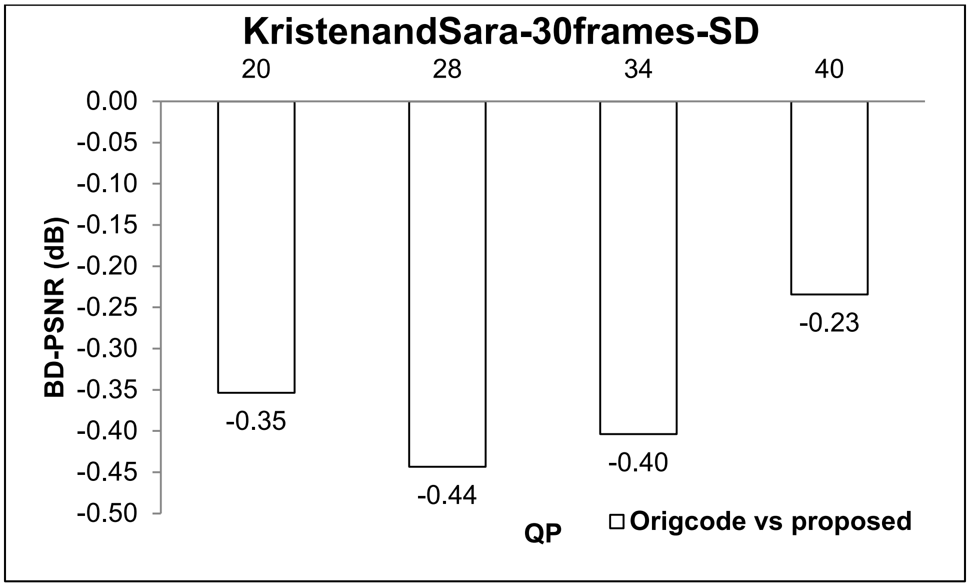


Figure 4-9 BD-PSNR vs. quantization parameter for KristenandSara

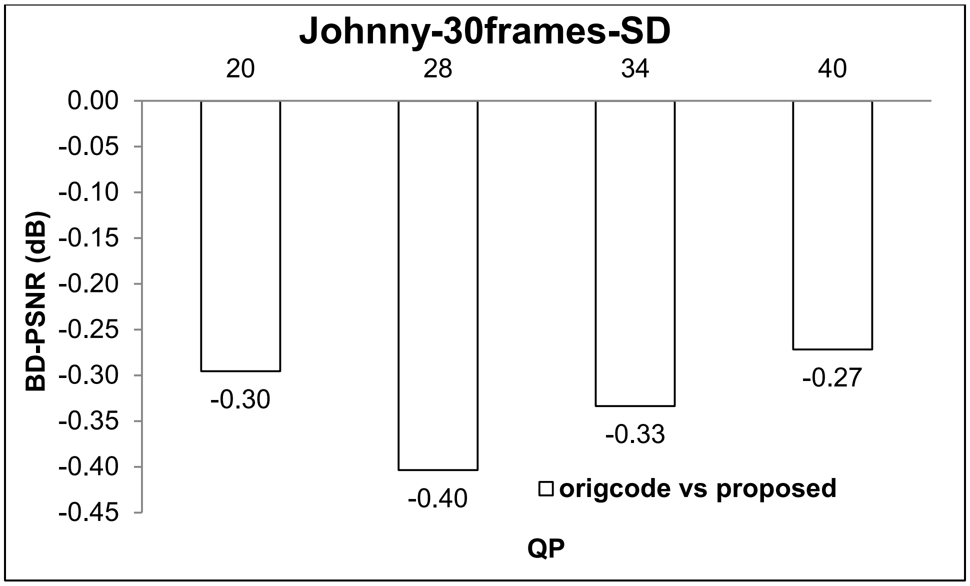


Figure 4-10 BD-PSNR vs. quantization parameter for Johnny

4.4 BD-bitrate gain

As expected from the observed BD-PSNR loss, there is a bitrate gain in the range of 5 kbps to15 kbps amongst the sequences tested at various quantization parameters (QPs) as seen in Figs 4-11 to 4-15. BD-bitrate is another curve fitting procedure based metric similar to BD-PSNR.[30-31]

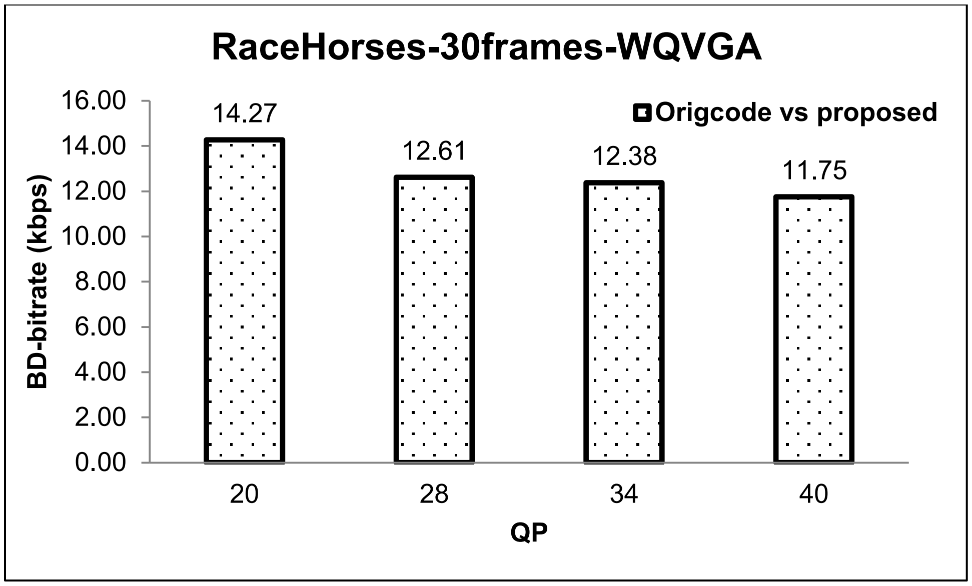


Figure 4-11 BD-bitrate vs. quantization parameter for Racehorses

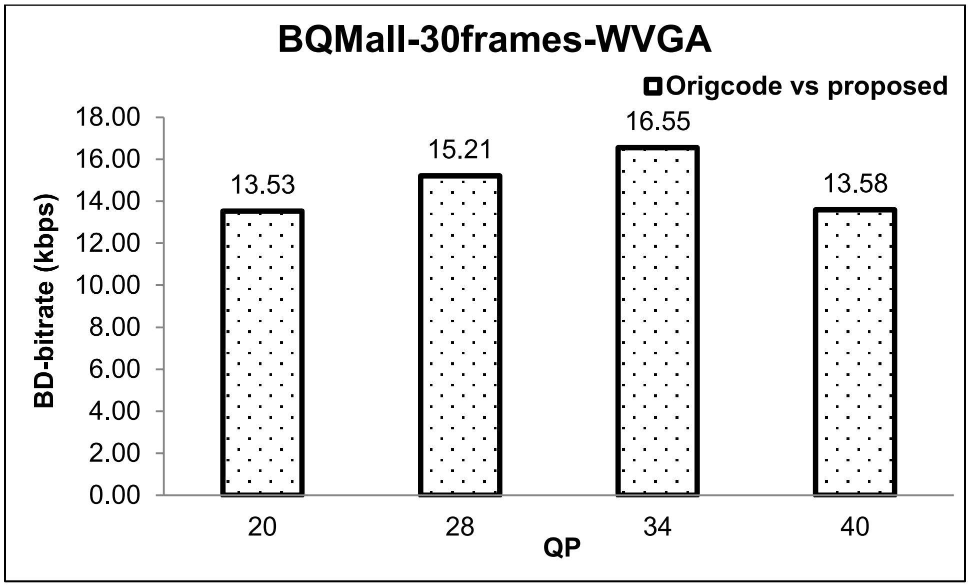


Figure 4-12 BD-bitrate vs. quantization parameter for BQMall

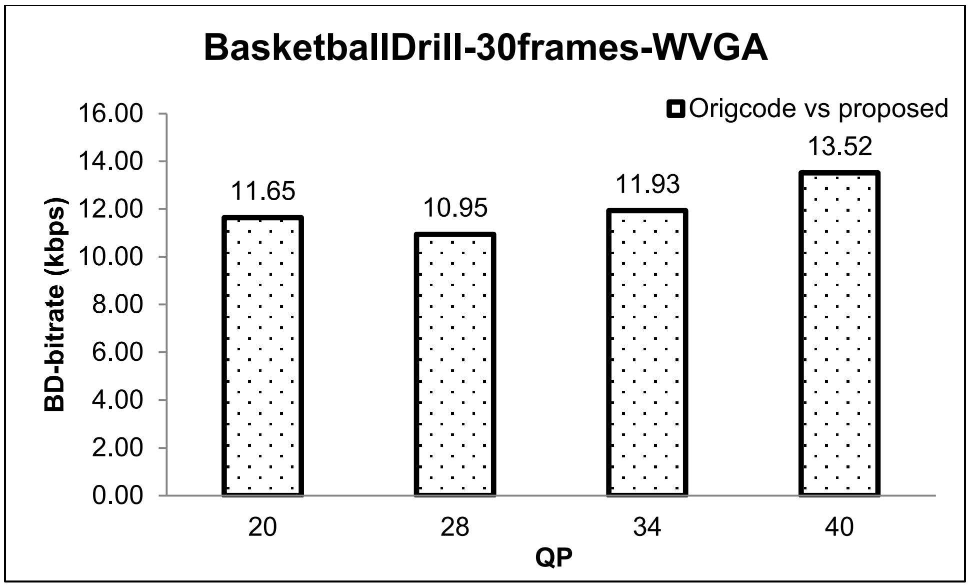


Figure 4-13 BD-bitrate vs. quantization parameter for BasketballDrillText

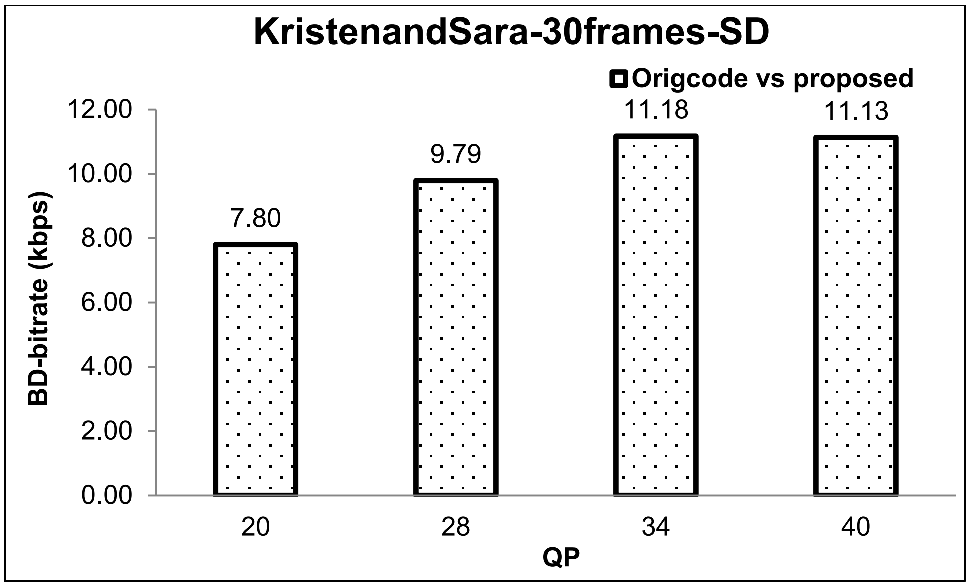


Figure 4-14 BD-bitrate vs. quantization parameter for KristenAndSara

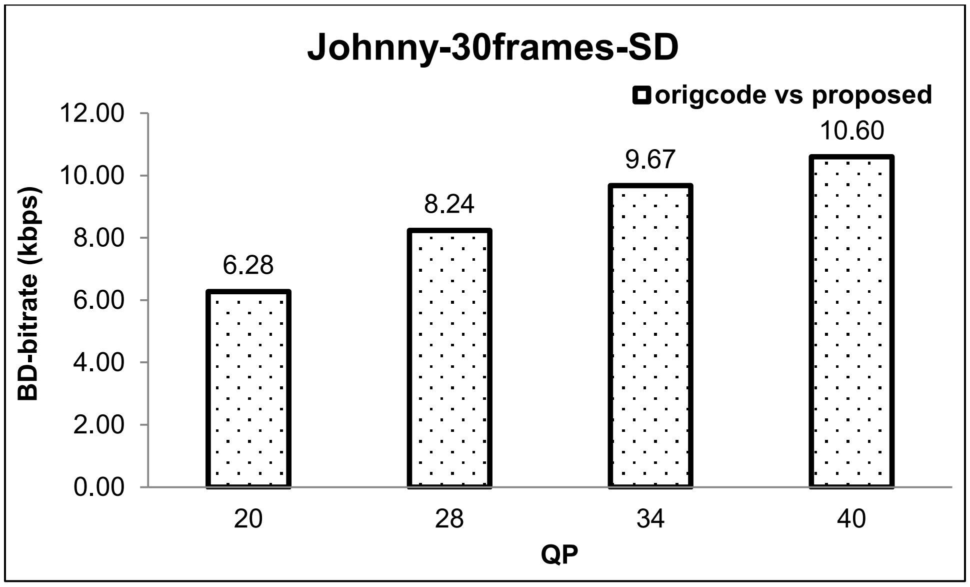


Figure 4-15 BD-bitrate vs. quantization parameter for Johnny

4.5 PSNRavg versus bitrate

As seen from Figs 4-16 to 4-20 that both the unmodified HM9.2 and the proposed combined fast intra prediction with fast RQT HM9.2 maintain similar PSNRavg-bitrate plots. The average PSNR for 4:2:0 video format is computed using the following formula:

$$PSNRavg = 0.125 \times ((6 \times PSNRy) + PSNRu + PSNRv) \quad (4.1)$$

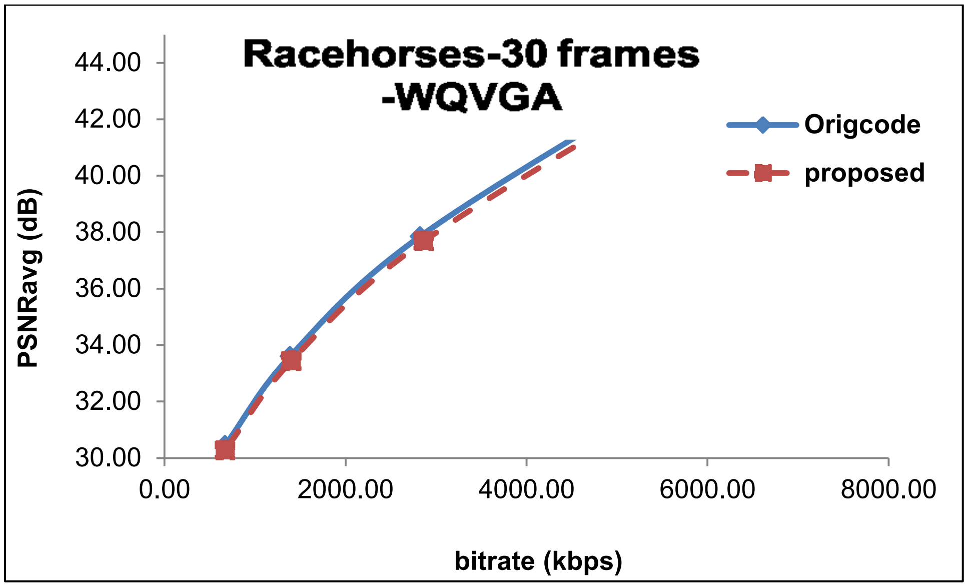


Figure 4-16 PSNRavg vs. bitrate for Racehorses

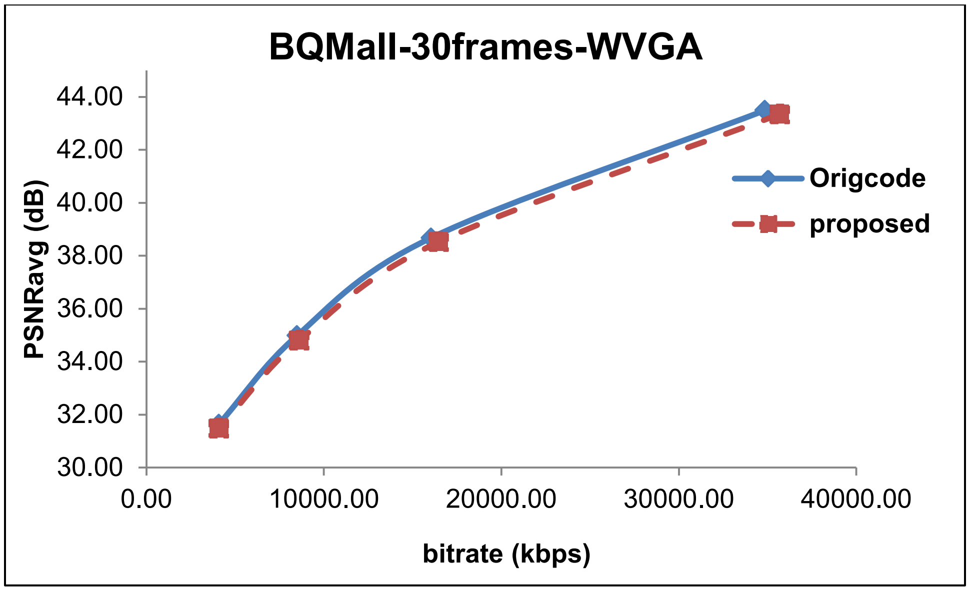


Figure 4-17 PSNRavg vs. bitrate for BQMall

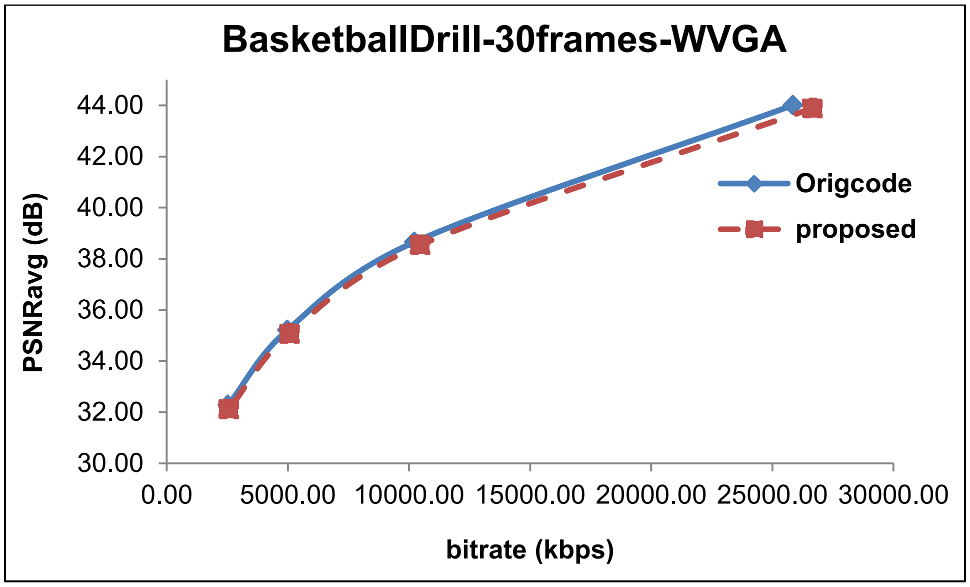


Figure 4-18 PSNRavg vs. bitrate for BasketballDrillText

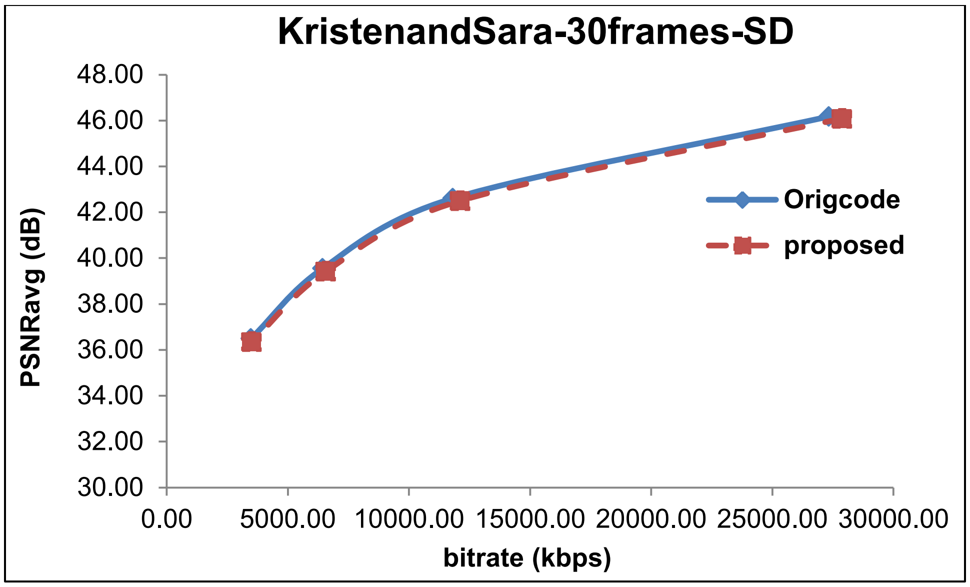


Figure 4-19 PSNRavg vs. bitrate for KristenAndSara

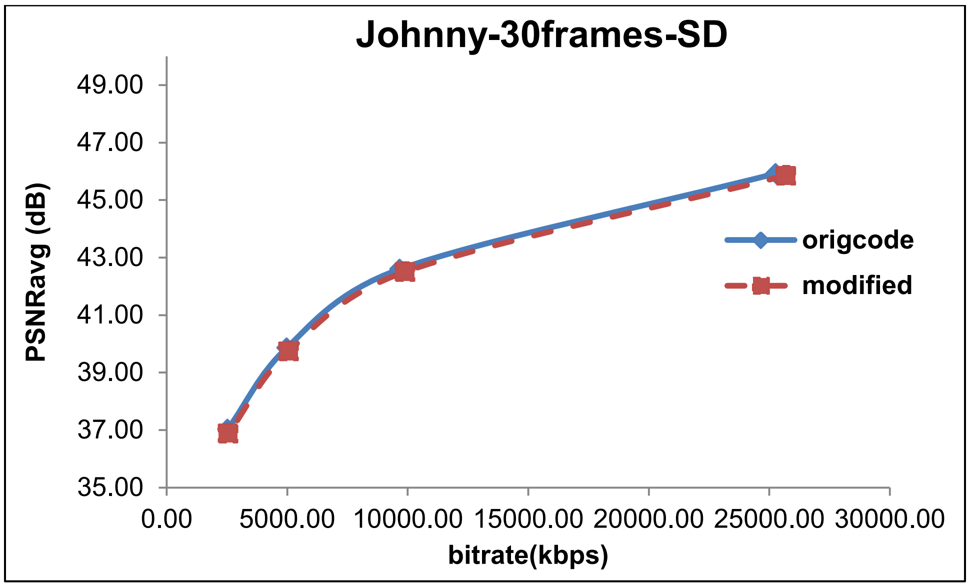


Figure 4-20 PSNRavg vs. bitrate for Johnny

4.6 Encoded bitstream size gain

As seen in Figs 4-21 to 4-25, there is almost a 1% to 3% increase in the final encoded bitstream size.

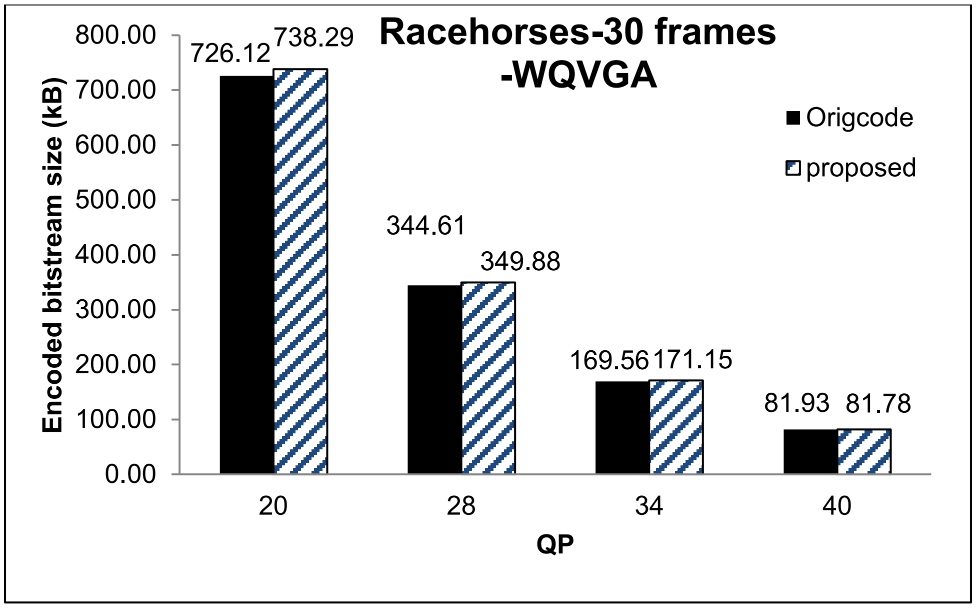


Figure 4-21 Encoded bitstream size vs. quantization parameter for Racehorses

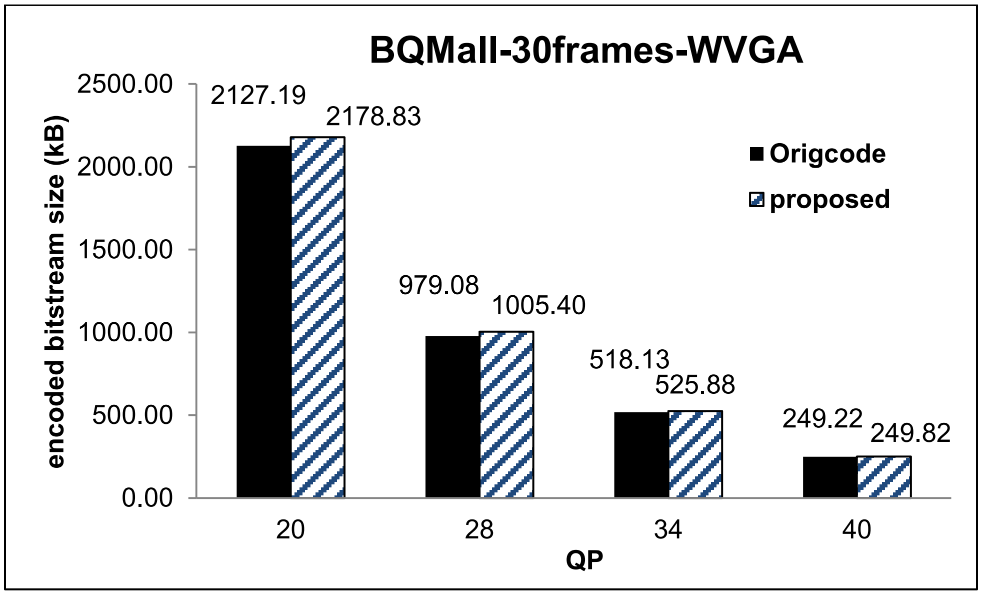


Figure 4-22 Encoded bitstream size vs. quantization parameter for BQMall

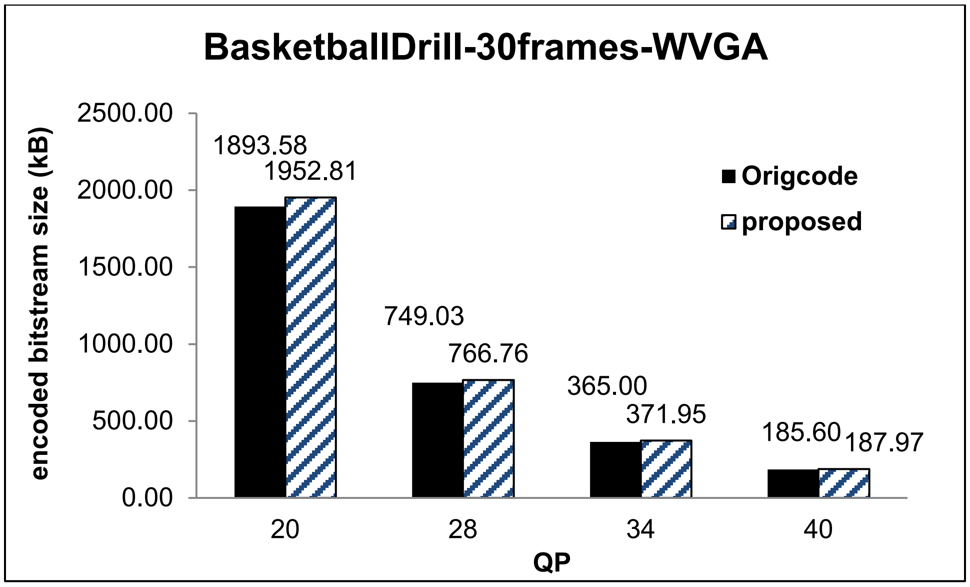


Figure 4-23 Encoded bitstream size vs. quantization parameter for BasketballDrillText

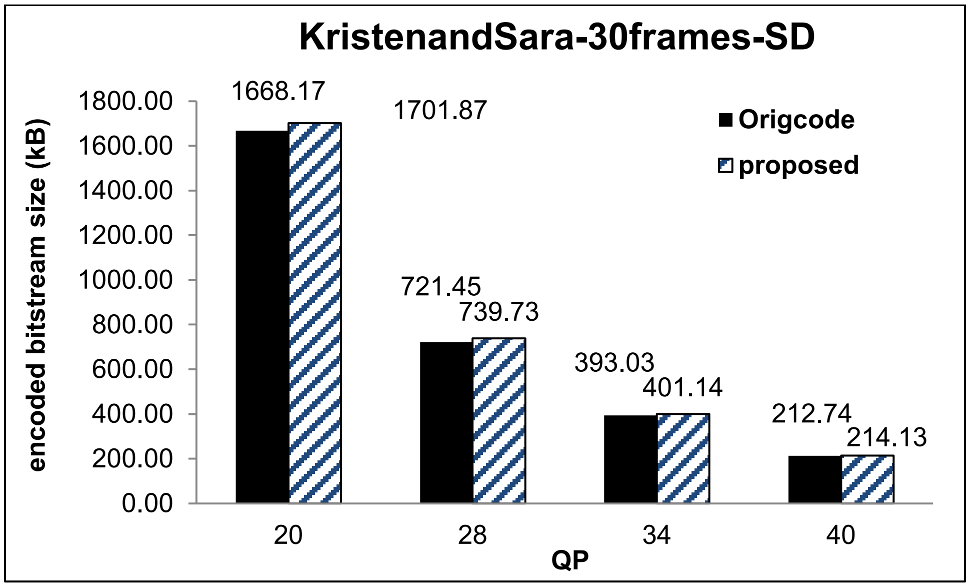


Figure 4-24 Encoded bitstream size vs. quantization parameter for KristenAndSara

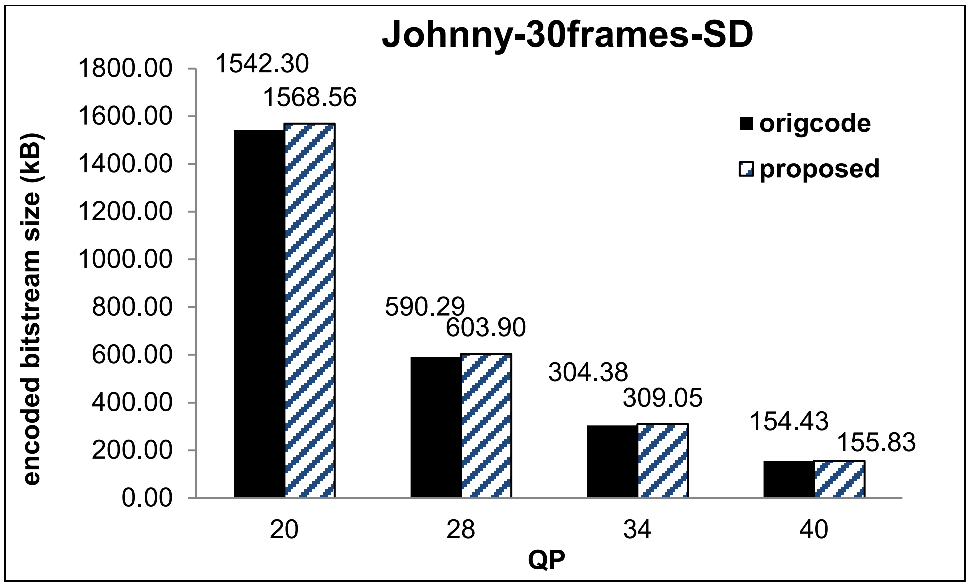


Figure 4-25 Encoded bitstream size vs. quantization parameter for Johnny

4.7 Encoding time of fast intra prediction and RQT combined versus only RQT

Simulations were also run to compare the original unmodified codebase against the codebase with only fast RQT implemented. Figs 4-26 to 4-30 show the improvement in encoding time of proposed fast intra prediction combined with fast RQT HM9.2 versus only the fast RQT implemented HM9.2. The proposed method has an additional 20% decrease in encoding time. It can be seen that as the sequences range from WQVGA to SD, at higher QPs of 38 and 40, both the methods seem to be equal, sometimes the RQT implementation HM9.2 performs better than the proposed HM9.2. However, at lower QPs, the proposed HM9.2 is clearly faring better.

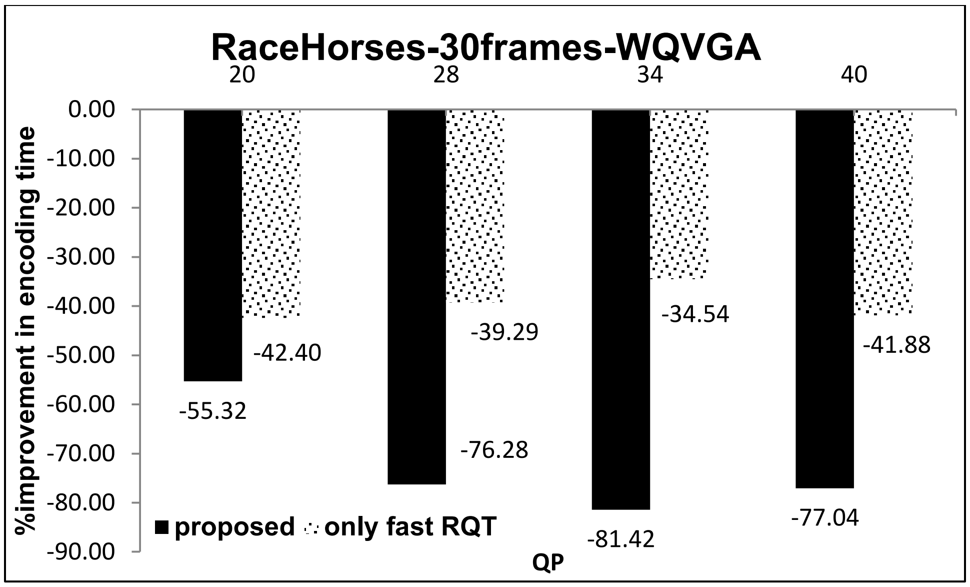


Figure 4-26 Encoding time improvement of fast intra prediction combined with fast RQT vs. only fast RQT for RaceHorses

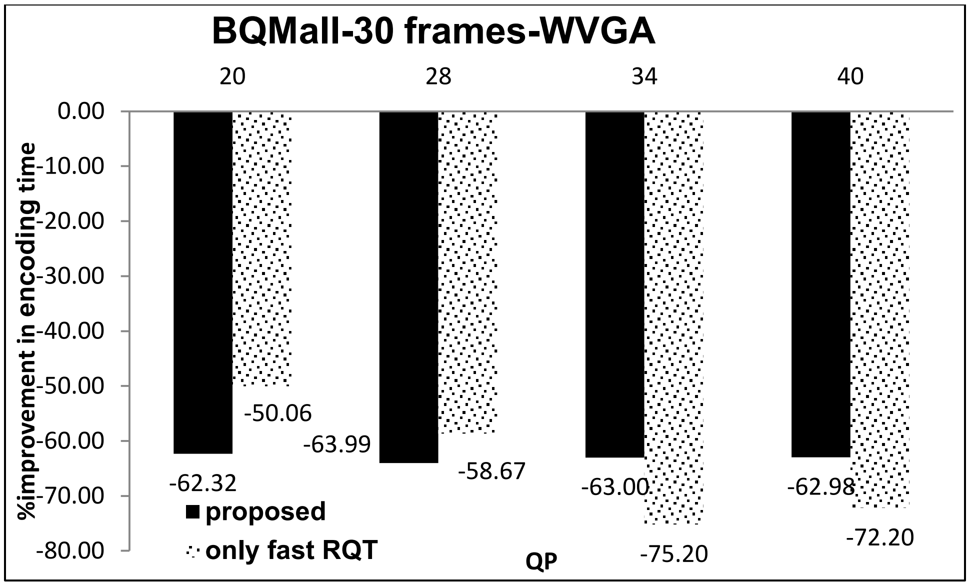


Figure 4-27 Encoding time improvement of fast intra prediction combined with fast RQT vs. only fast RQT for BQMall

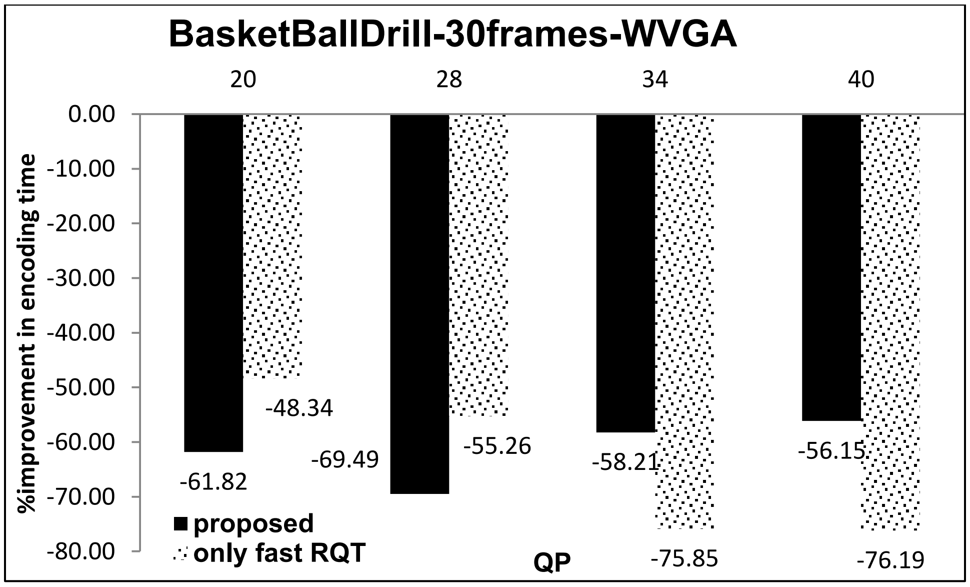


Figure 4-28 Encoding time improvement of fast intra prediction combined with fast RQT vs. only fast RQT for BasketballDrillText

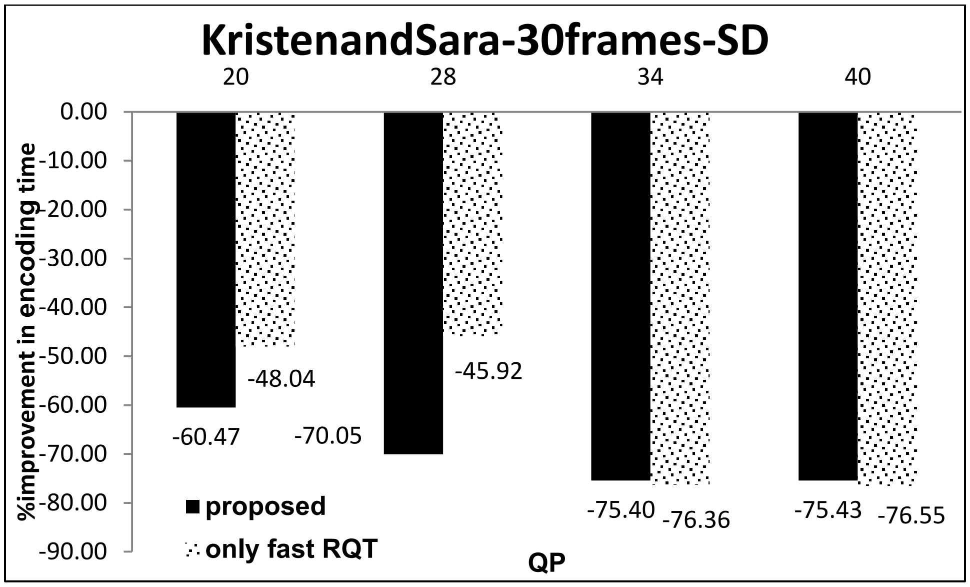


Figure 4-29 Encoding time improvement of fast intra prediction combined with fast RQT vs. only fast RQT for KristenandSara

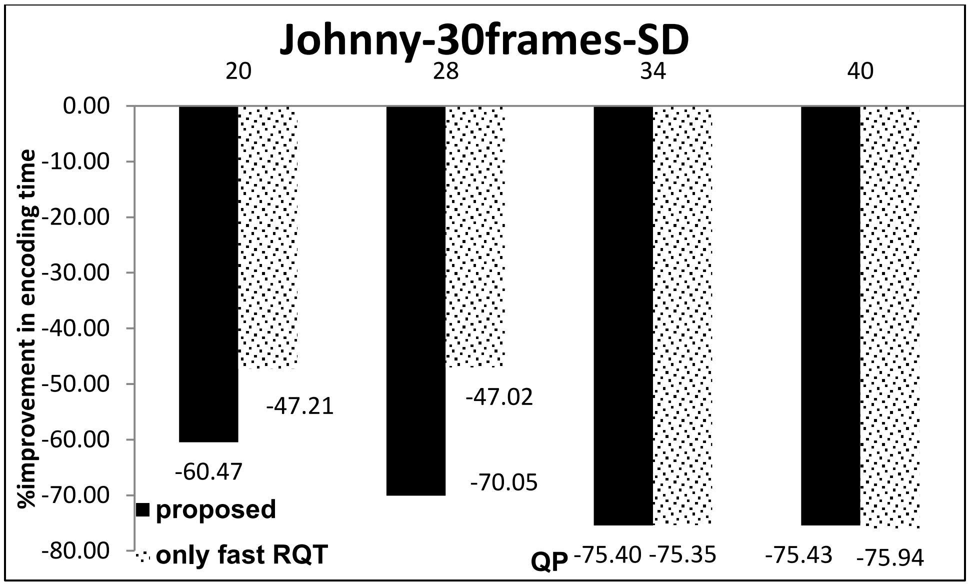


Figure 4-30 Encoding time improvement of fast intra prediction combined with fast RQT vs. only fast RQT for Johnny

4.8 PSNRavg comparison

Figs 4-31 to 4-35 show the comparison of loss in PSNRavg of proposed fast intra prediction combined with fast RQT HM9.2 against only fast RQT implemented HM9.2. The PSNRavg loss in the proposed method is two times more than only RQT implemented HM9.2.

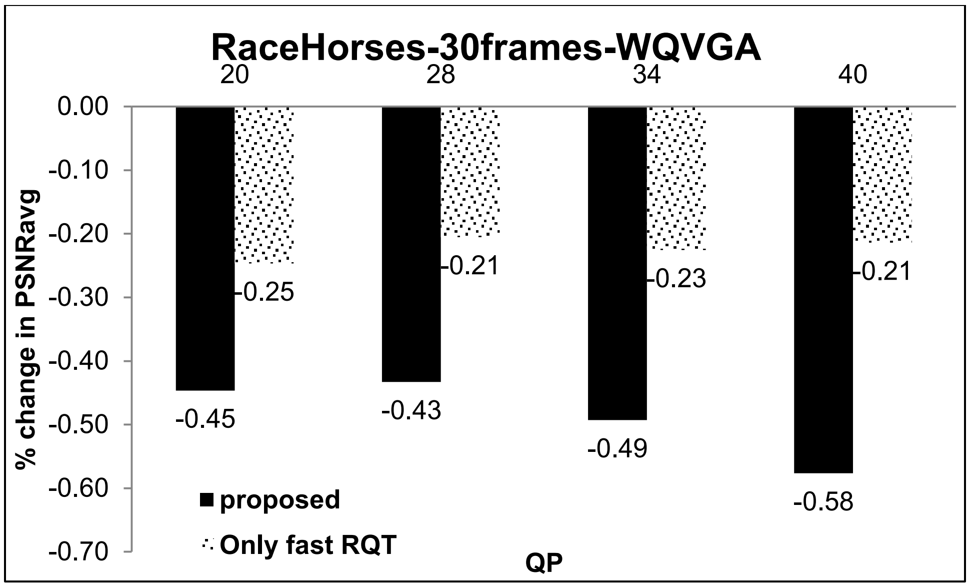


Figure 4-31 PSNRavg loss of fast intra prediction combined with fast RQT vs. only fast RQT for RaceHorses

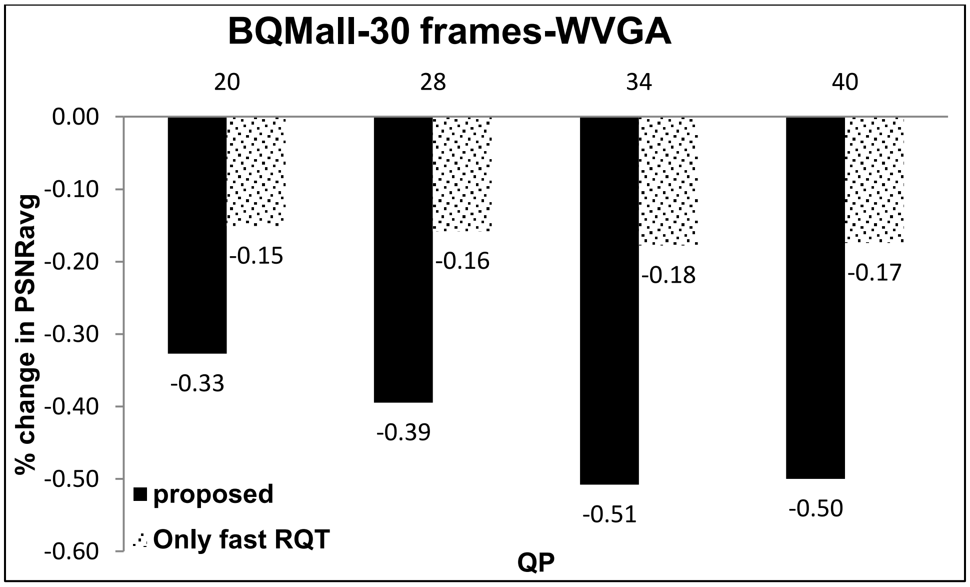


Figure 4-32 PSNRavg loss of fast intra prediction combined with fast RQT vs. only fast RQT for BQMall

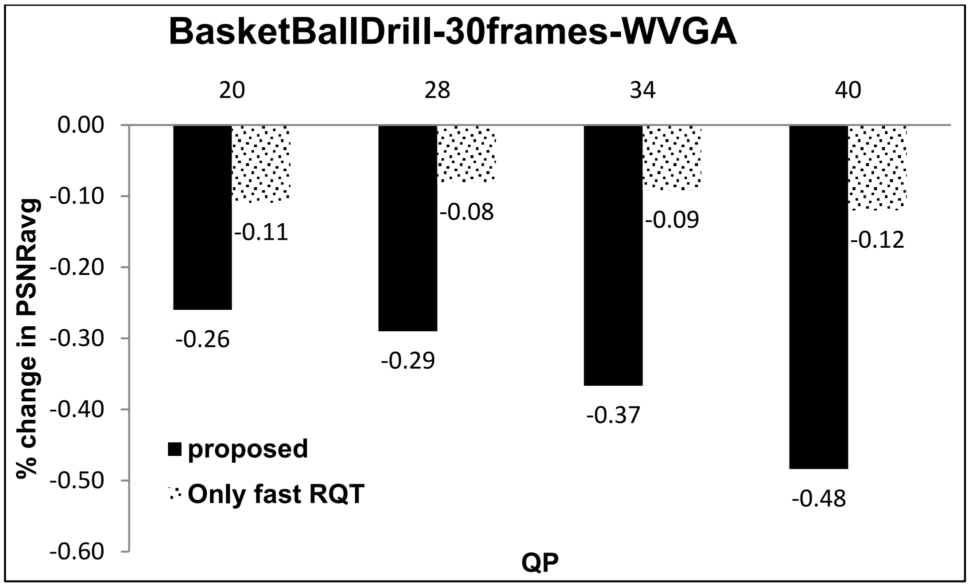


Figure 4-33 PSNRavg loss of fast intra prediction combined with fast RQT vs. only fast RQT for BasketBallDrillText

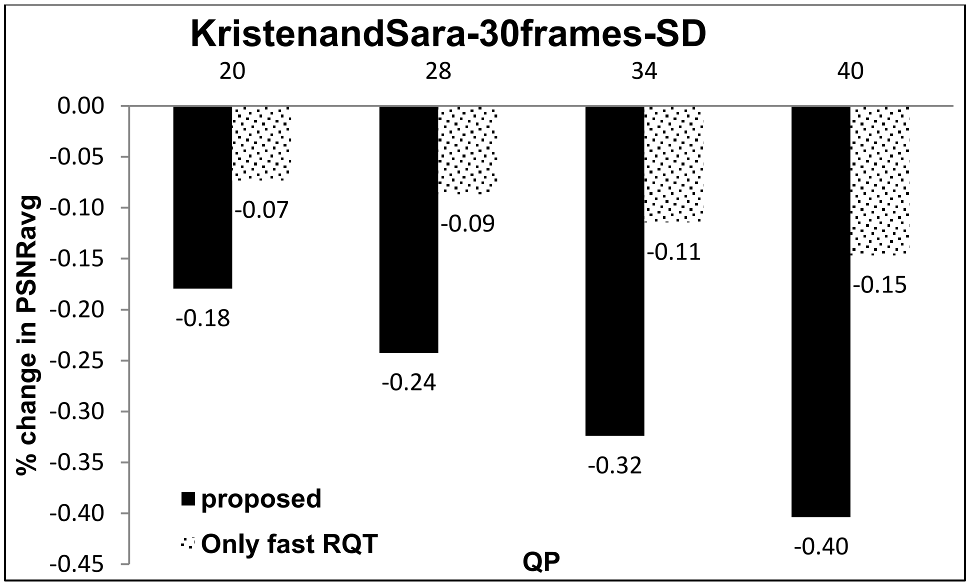


Figure 4-34 PSNRavg loss of fast intra prediction combined with fast RQT vs. only fast RQT for KristenandSara

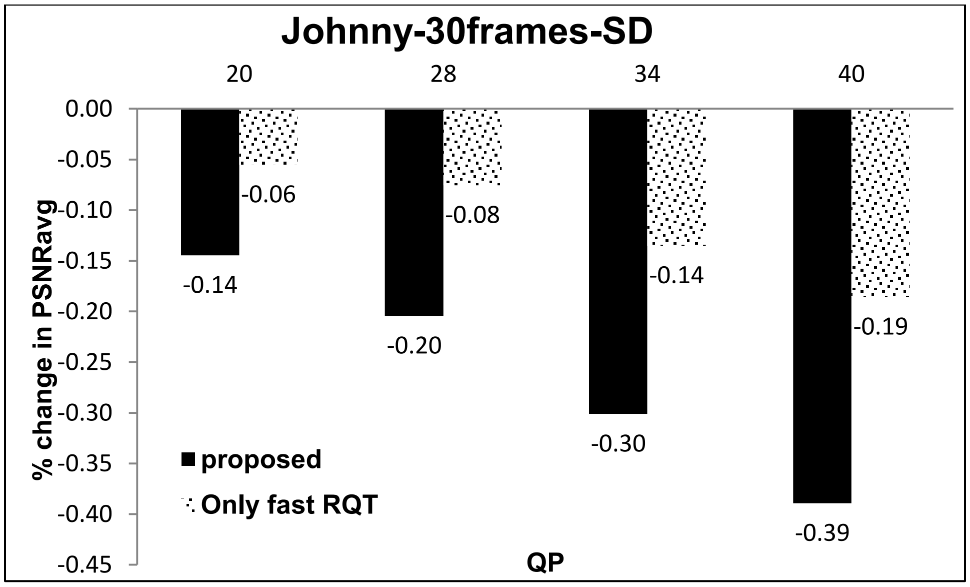


Figure 4-35 PSNRavg loss of fast intra prediction combined with fast RQT vs. only fast RQT for Johnny

4.9 Bitrate/size of encoded bitstream size gain of proposed method versus only RQT

Figs 4-36 to 4-40 show the gain in bitrate of fast intra prediction combined with fast RQT against only fast RQT in the codebase. As expected from the higher loss in PSNR for the proposed method, it also has a higher gain in bitrate when compared to the HM9.2 with only fast RQT implemented.

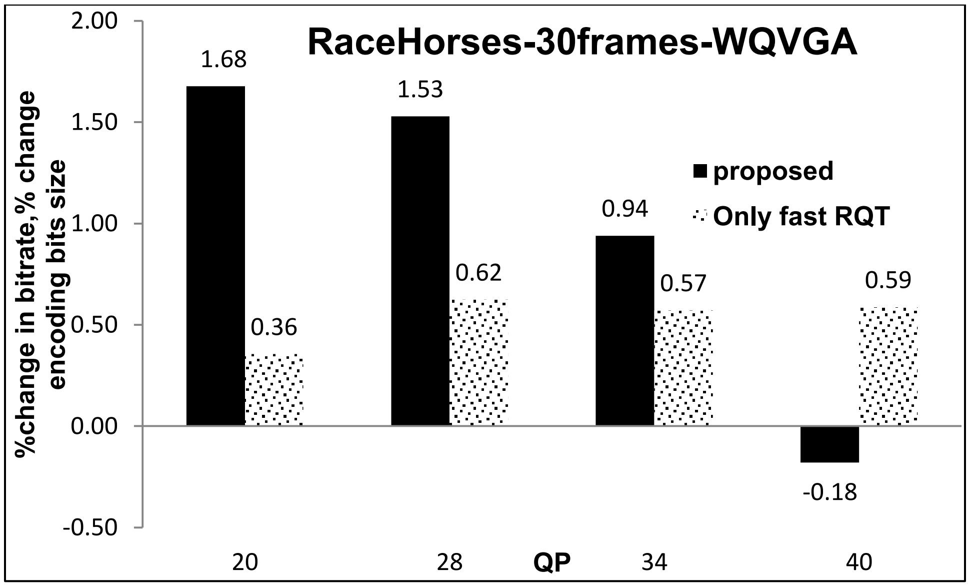


Figure 4-36 Bitrate gain of fast intra prediction combined with fast RQT vs. only fast RQT for RaceHorses

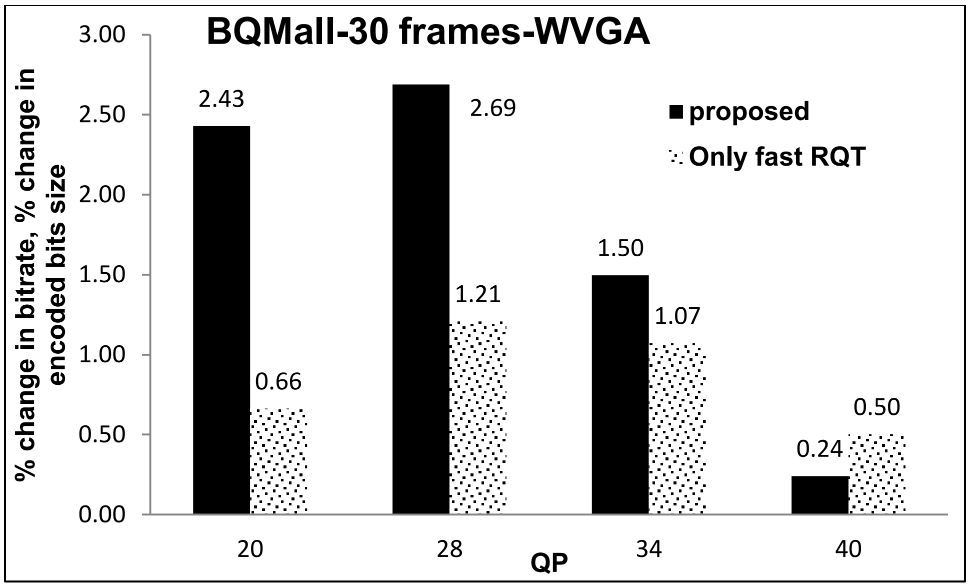


Figure 4-37 Bitrate gain of fast intra prediction combined with fast RQT vs. only fast RQT for BQMall

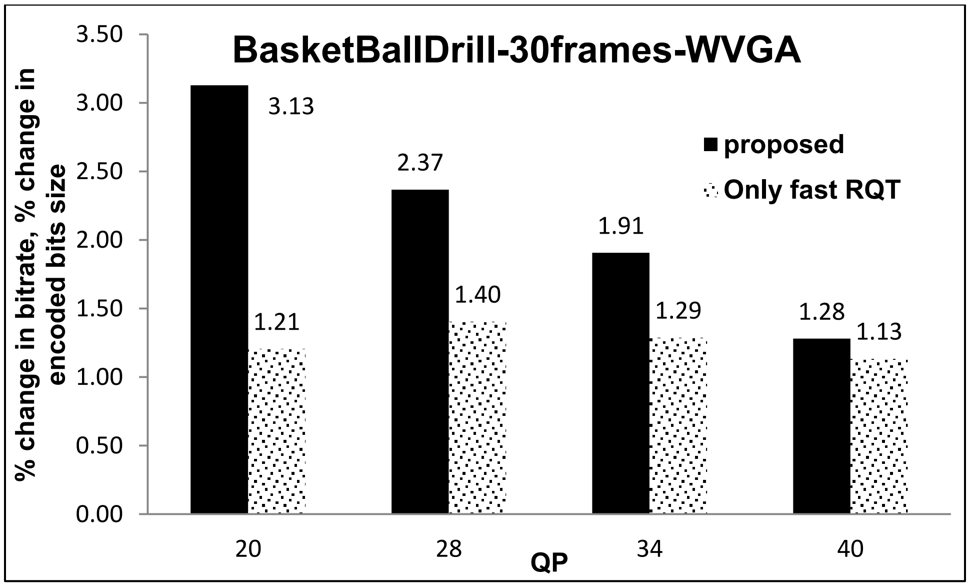


Figure 4-38 Bitrate gain of fast intra prediction combined with fast RQT vs. only fast RQT for BasketballDrillText

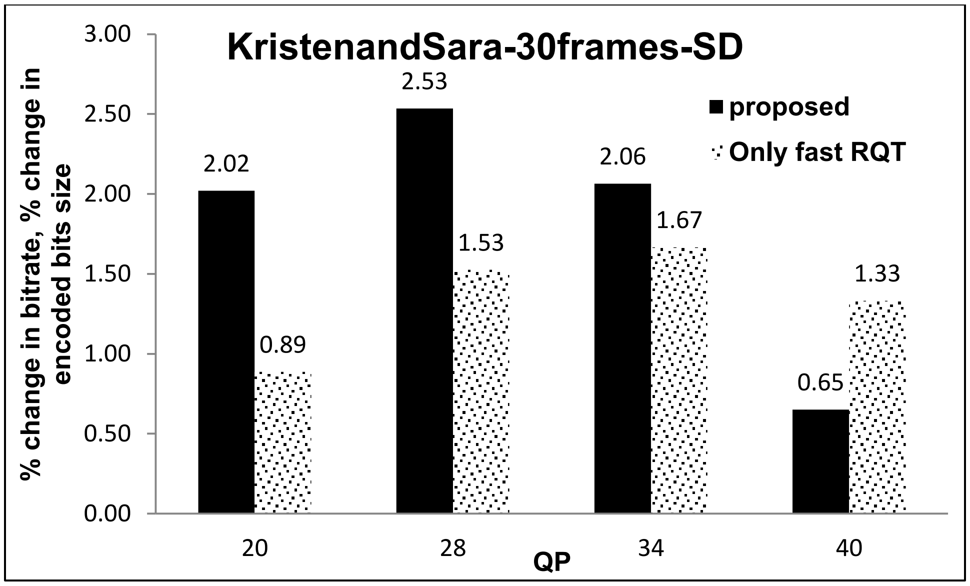


Figure 4-39 Bitrate gain of fast intra prediction combined with fast RQT vs. only fast RQT for KristenandSara

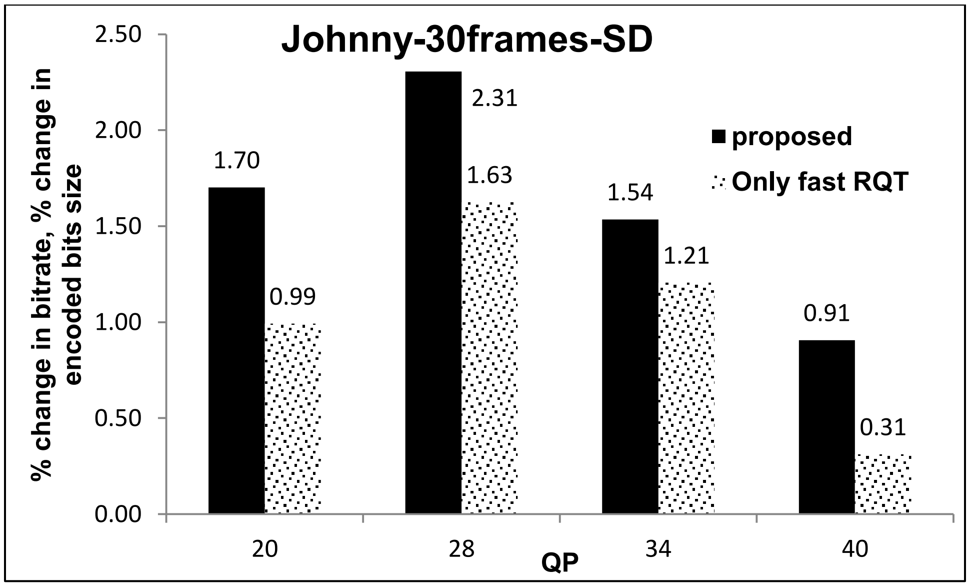


Figure 4-40 Bitrate gain of fast intra prediction combined with fast RQT vs. only fast RQT for Johnny

4.9 Conclusions

It can be seen that though there is a slight drop in the PSNR and a negligible increase in the bitrate and encoding bitstream size, the reduction in encoding time for sequences of different resolutions at different QPs is in the range of 55%-61% for the proposed fast intra prediction combined with fast RQT HM9.2. It can also been seen that with only fast RQT implemented, the encoding time was reduced, bitrate was increased and PSNRavg was decreased. However, these effects are more pronounced when two methods are combined into one codebase. Overall the proposed method has shown significant reduction in HEVC encoding time compared to unmodified HM9.2 with negligible loss in the visual quality based on BD-PSNR and BD-bitrate.

Chapter 5

Future Work

There are many other ways to be explored in the fast intra prediction and fast RQT area as suggested by the author in [6]. 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.

Teng et [22] al talk about a merge-and-split process that can replace the original depth first RQT coding scheme. It describes an early termination of the splitting when TU size is zero.

Zhao et al [21] take advantage of the fact that with respect to the current block, the surrounding blocks will most likely have similar textures. This piece of information can be used to predict the current block and thus skip the RDOQ steps during intra mode selection. This can be combined with the proposed method.

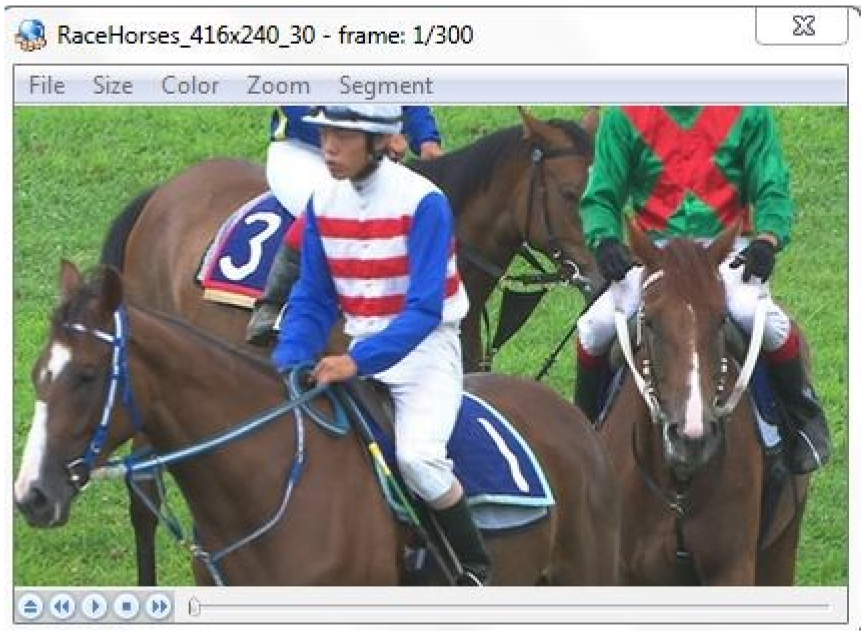
Another fact of encoding is CU size decisions which is the leaf node of the encoding process in the quadtree. Bayesian decision rule can be applied to calculate the CU size and then this information can be combined with the proposed method to achieve further encoding time gains. [24]

Reduction in encoding time can also be achieved through hardware optimization. This is a vast path and can be explored in this regard.

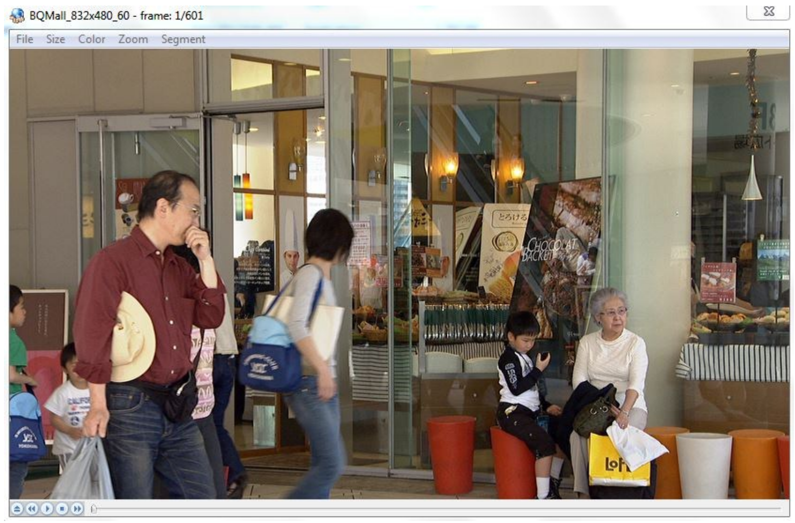
Appendix A.

Test Sequences used [32]

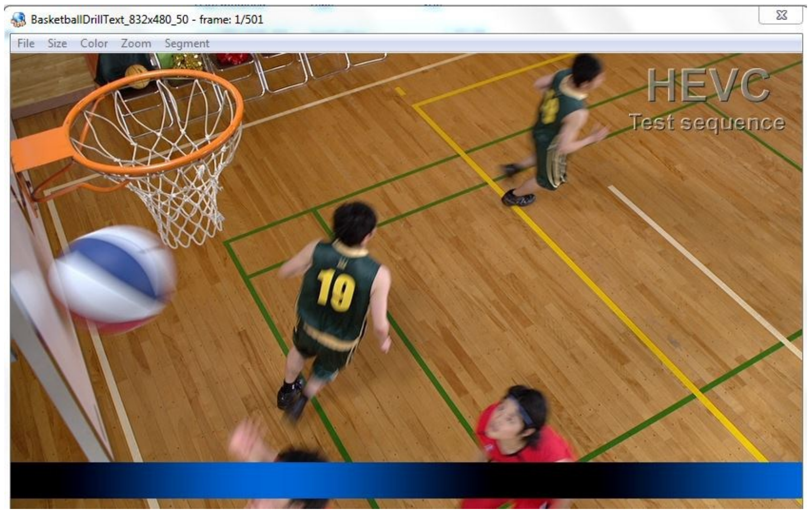
A.1 Racehorses



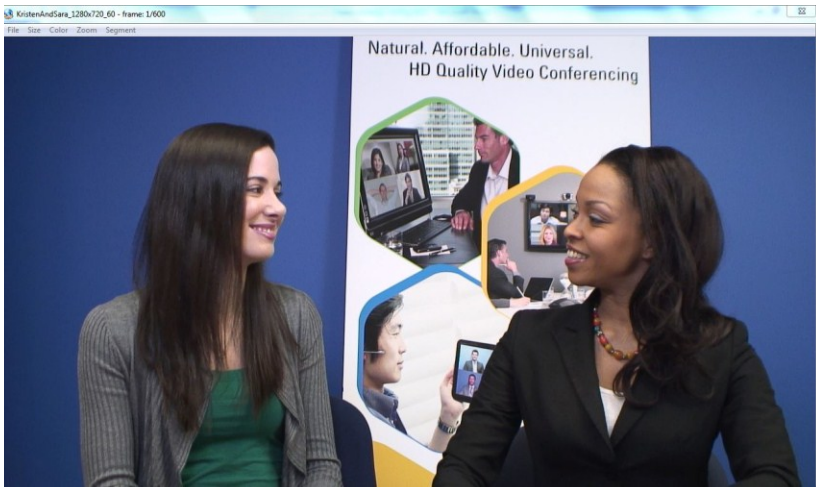
A.2 BQMall


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

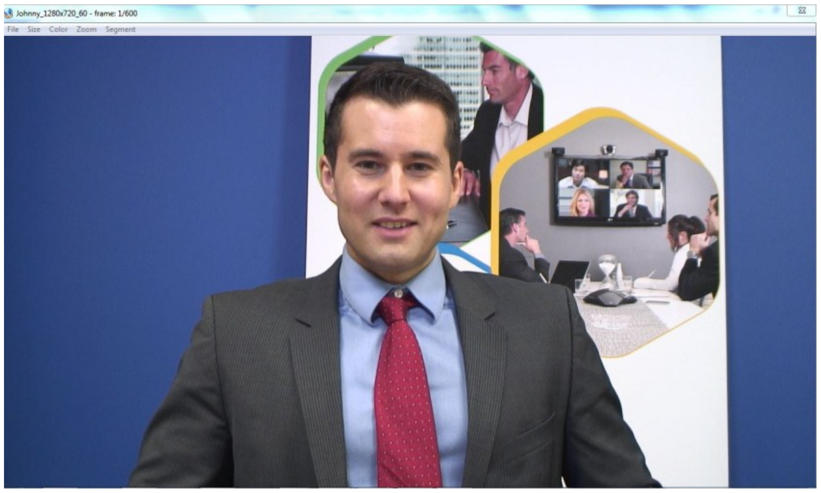
A.3 BasketBallDrillText


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 Johnny


Johnny_1280x720_60 - frame: 1/600
File Size Color Zoom Segment

Appendix B.

Test Conditions

B.1 Description

The code revision used for this work is revision 3374 termed HM9.2 [8]. The research was carried out using an Intel Core 2 Duo machine with Microsoft Windows 7 32bit version running with 3 GB RAM at a speed of 2GHz.

Appendix C.

Bjontegaard Metrics

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*bit^2)/(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: $10^{0.05} = 1.122$ (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*bit^2 + d*bit^3

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*SNR^2 + d*SNR^3

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 ∆SNR and ∆bitrate 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?

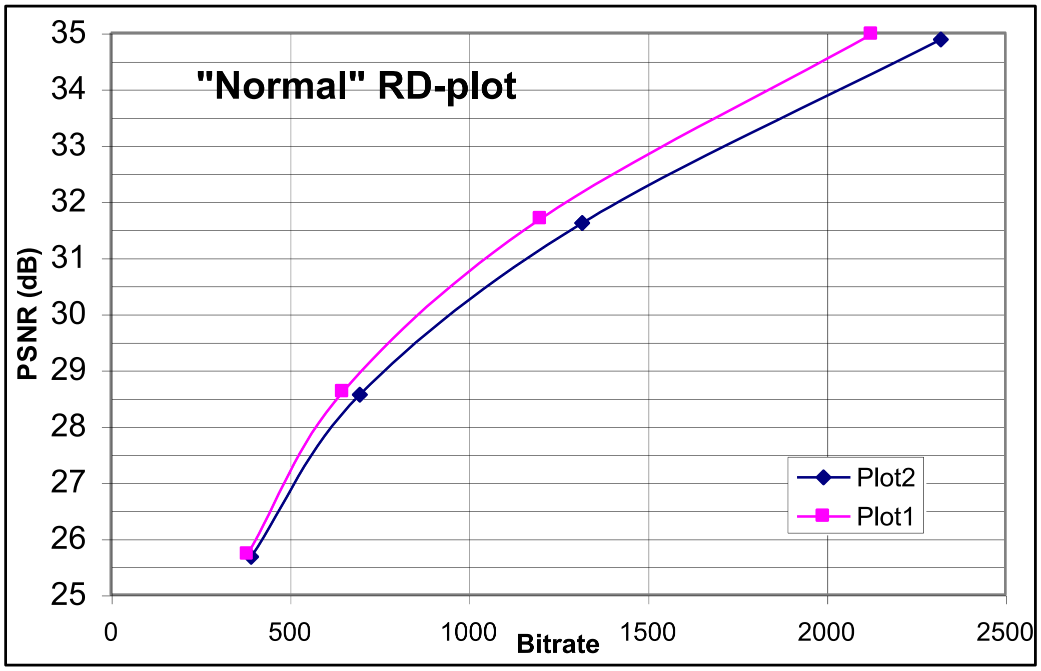

"Normal" RD-plot
PSNR (dB)
35
34
33
32
31
30
29
28
27
26
25
0
500
1000
1500
2000
2500
Bitrate
Plot2
Plot1

Figure 1

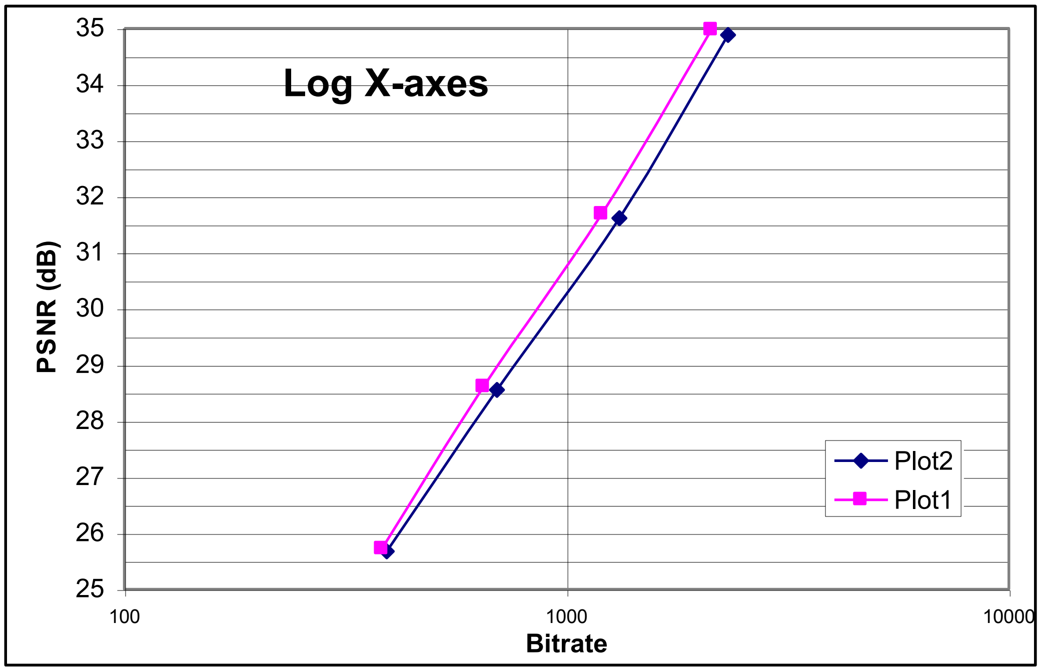

Log X-axes
PSNR (dB)
35
34
33
32
31
30
29
28
27
26
25
100
1000
10000
Bitrate
Plot2
Plot1

Figure 2

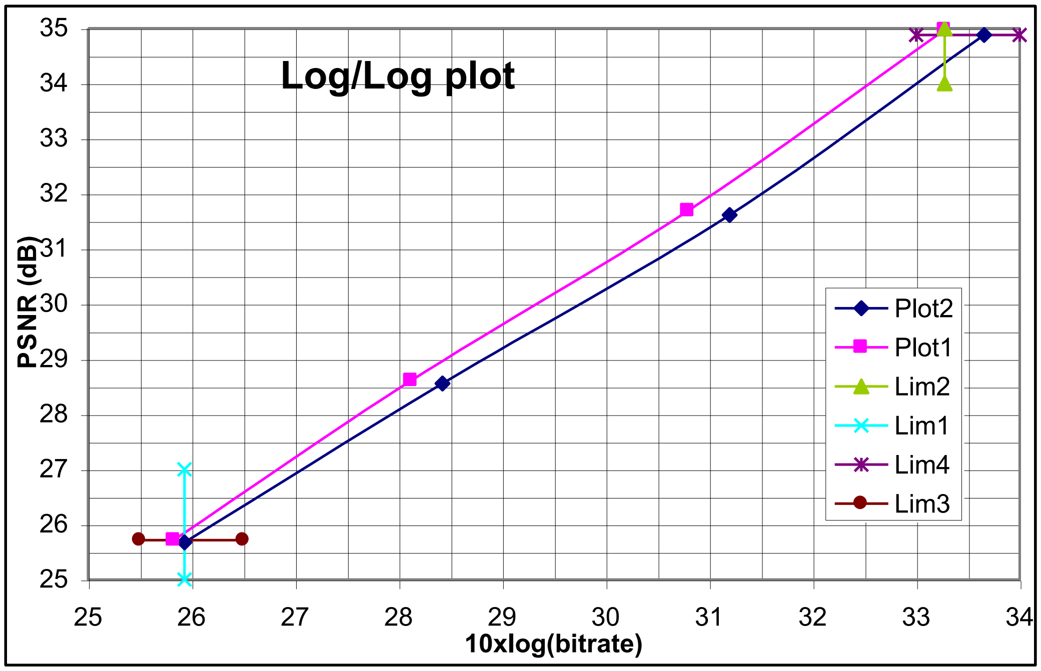


Figure 3

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 = bjontegaard(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.
%
%   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
```

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.

List of acronyms

AVC – Advanced Video Coding

AMVP – Advanced Motion Vector Prediction

BD - Bjontegaard Delta

CABAC – Context Adaptive Binary Arithmetic Coding

CB – Coding Block

CTU – Coding Tree Unit

CTB – Coding Tree Block

CU – Coding Unit

DCT – Discrete Cosine Transform

DST – Discrete Sine Transform

FDIS – Final Draft Internation 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

MPEG – Moving Picture Experts Group

MPM – Most Probable Modes

NAL – Network Abstraction Layer

PB – Prediction Block

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
VCEG	– Video Coding Experts Group
VPS	– Video Parameter Set
WQVGA	– Wide Quarter Video Graphics Array
WVGA	– Wide Video Graphics Array
TU	– Transform Unit
URQ	– Uniform Reconstruction Quantization

References

[1] K. Sayood, "Introduction to data compression", 4th edition, Morgan Kaufmann, 2010

[2] C.M. Huang et al, "Error resilience supporting bi-directional frame recovery for video streaming", Proceedings, ICIP, v1, pp 537-540, 2004

[3] A. Luthra and P. Topiwala, "Overview of the H.264/AVC video coding standard", Proceedings of SPIE - The International Society for Optical Engineering, vol 5203, pp 417-431, Applications of Digital Image Processing XXVI, 5-8 Aug. 2003

[4] K.R. Rao et al, "Video Coding Standards", 1st edition, Springer, 2014

[5] N. Ling," High efficiency video coding and its 3d extension: A research perspective", 7th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 2150-2155, Jul 2012.

[6] H. Zhang and Z. Ma, "Fast intra prediction for high efficiency video coding", Advances in Multimedia Information Processing, 13th Pacific-Rim Conference on Multimedia, Proceedings, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol 7674 LNCS, pp 568-577, 4-6 Dec. 2012

[7] Y.H. Tan et al, "On residual quad-tree coding in HEVC", MMSP 2011 - IEEE International Workshop on Multimedia Signal Processing, 17-19 Nov. 2011

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

[9] B. Bross et al, "High efficiency video coding (HEVC) text specification draft 8", Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 10th Meeting: Stockholm, SE, 11–20 July 2012

[10] G. Sullivan et al, "Overview of the high efficiency video coding (HEVC) standard", IEEE Transactions on Circuits and Systems for Video Technology, vol 22, n 12, pp 1649-1668, Dec. 2012

[11] 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

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

[13] A. Saxena and F.C. Fernandes, “ Mode dependent DCT/DST for intra prediction in block-based image/video coding”, Proceedings - International Conference on Image Processing, IEEE ICIP, pp 1685-1688, 11-14 Sept. 2011

[14] V. Sze and M. Budagavi, “High throughput CABAC entropy coding in HEVC”, IEEE Transactions on Circuits and Systems for Video Technology, vol 22, n 12, pp 1778-1791, Dec. 2012

[15] F. Bossen et al, "HEVC complexity and Implementation analysis", IEEE Transactions on Circuits and Systems for Video Technology, vol 22 , pp 1685-1696, Dec. 2012

[16] 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

[17] A.S. Motra et al, “Fast intra mode decision for HEVC video encoder”, 2012 20th International Conference on Software, Telecommunications and Computer Networks, SoftCOM, 11-13 Sept. 2012

[18] J. Kim et al, “Fast intra mode decision of HEVC based on hierarchical structure”, ICICS 2011 - 8th International Conference on Information, Communications and Signal Processing, 13-16 Dec. 2011

[19] K. Choi and E.S. Jang, “Early TU decision method for fast video encoding in high efficiency video coding”, IET, Electronics Letters, vol. 48, No. 12, pp 689-691, 7th June 2012

[20] W. Jiang, H. Ma and Y. Chen, “Gradient based fast mode decision algorithm for intra prediction in HEVC”, 2nd International Conference on Consumer Electronics, Communications and Networks, pp 1836-1840, 21-23 April 2012

[21] X. Shen, L. Yu and J. Chen, “Fast coding unit size selection for HEVC based on Bayesian decision rule”, Picture Coding Symposium, 2012, Proceedings, pp 453-456, 7-9 May 2012

[22] S.W. Teng, H.M. Hang and Y.F. Chen, “Fast mode decision algorithm for residual quadtree coding in HEVC”, Conference Article no.6116062, IEEE Visual Communications and Image Processing, 6-9 Nov. 2011.

[23] J. Leng et al, “Content based hierarchical fast coding unit decision algorithm for HEVC”, International Conference on Multimedia and Signal Processing, vol 1, pp 56-59, 14-15 May 2011.

[24] L. Zhao et al, “Fast mode decision algorithm for intra prediction in HEVC”, Conference Article no.6115979, IEEE Visual Communications and Image Processing, 6-9 Nov. 2011

[25] H. Zhao et al, “Fast intra mode decision for high efficiency video coding (HEVC)”, IEEE Transactions on Circuits and Systems for Video Technology (under review)

[26] T.L. Da Silva et al, “HEVC intra coding acceleration based on tree-inter level mode Correlation”, IEEE and SPA, Poznam, Poland, 26-28 Sept. 2013

[27] L.L.Wang et al,”Novel adaptive algorithm for intra prediction with compromised modes skipping processes in HEVC”, IEEE Trans. CSVT, vol. 23,pp 1686-1694, Oct. 2013

[28] Open source for fast intra prediction on HM7.0 http://vision.poly.edu/~zma03/opensrc/

[29] HEVC software manual (available as a README inside the codebase)

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

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

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

Biographical Information

Sapna Vasudevan was born in Taliparamba, Kerala, India. After completing her schooling at Indian School, Bahrain in 2003, she went on to obtain her B.Tech in Electronics and Communication from Model Engineering College in 2007. From 2008 to 2011, she worked as an ASIC Verification Engineer with Wipro Technologies, Bangalore.

She joined University of Texas at Arlington to pursue her M.S in Electrical Engineering in 2012. This was around the time she joined the Multimedia Processing Lab. She is presently working as a Summer/Fall ASIC Verification intern at NVidia, Austin and subsequently will join AMD, Austin after she graduates.