Applied Mathematical Sciences, Vol. 8, 2014, no. 175, 8721 - 8736
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ams.2014.411945

Intra Prediction Header Bits Estimation Algorithm for RDO in H.265/HEVC

Maxim P. Sharabayko[1]

[1]Tomsk Polytechnic University, 634050 Tomsk, Russia, 30 Lenin Prospekt

Oleg G. Ponomarev[2,3]

[2]Tomsk State University of Control Systems and Radioelectronics
634050 Tomsk, Russia, 40 Lenin Prospekt
[3]Tomsk State University, 634050 Tomsk, Russia, 36 Lenin Prospekt

Abstract

The amount of video information to store and transmit is continuously increasing. The new video compression standard H.265/HEVC provides the ability to halve video bitrate at the cost of higher computational expenses. The most time-demanding part of a compression system is decision making, especially when based on rate-distortion optimization (RDO). There are a lot of papers that tend to decrease the complexity of decision making by either reducing the number of coding options to compare or by reducing the complexity of RDO estimation. The aim of the latter approach is to eliminate from rate estimation the binary arithmetic coder (BAC), as it is the most computationally expensive step. Most of works focus on proper estimation of the residual data, while proper estimation of prediction information is either neglected or is still performed with BAC.

In our work we elaborate the proper rate estimation of the prediction header without the use of BAC. The suggested algorithm counts the symbols to be coded with BAC and estimates the bit size of the intra prediction header based on the entropy of the symbols, obtained in the statistical experiments. The proposed algorithm provides the average

bitrate overhead of 0.09% on JCT-VC test video sequences, and can be used as a part of decision making algorithm in H.265/HEVC video compression systems.

Mathematics Subject Classification: 94A08

Keywords: intra prediction header, bit counting, RDO, H.265/HEVC

1 Introduction

Nowadays video compression systems have a lot of implementations in digital television, OTT, video conferencing, video sensing, etc. At the same time the amount of video information to store and transmit is continuously increasing. The efficiency of video compression systems determines the cost, the quality and the prevalence of the qualitative video itself. The compression efficiency of current industrial video compression standard H.264/AVC is already not enough to provide wide access to the ultra high definition (UltraHD) video content. To increase the compression efficiency the new video compression standard H.265/HEVC was developed.

The increase in compression efficiency of H.265/HEVC is achieved with larger coding blocks, more intra prediction directions, larger choice of transform sizes, etc. The larger choice of block coding options in H.265/HEVC leads to more complex and computationally more expensive compression algorithms. At the same time the technical application of the new standard strongly depends on the performance of HEVC-based systems.

One of the main computationally expensive steps in H.265 video compression is rate-distortion optimized decision making or, simply, RDO. The main computational difficulty of RDO is a conventional need to use adaptive binary arithmetic coder (BAC) to count the number of bits for each estimated block coding option. This bit counting procedure involves binary arithmetic coder represented by SBAC (Syntax-based Binary Arithmetic Coder) in HEVC. SBAC distinguishes input symbols (bins) by syntax groups. The least probable symbol (LPS) and the most probable symbol (MPS) values, as well as the probability of their occurrence within each syntax group, are determined by separate context probability model. The probability models determine the self-information for the value of each context-dependent bin. At the same time, some bins are coded with the fixed probability of 50% in the so-called 'bypass' mode and does not depend on the probability state of SBAC. The SBAC is an adaptive arithmetic coder, which means that each coded bin updates the state of context probability model and the probability interval of the arithmetic coder. The need to update and preserve SBAC probability

state to perform proper bit counting corresponds to the most computationally expensive part of RDO.

The most important simplification of the bit counting procedure is presented in [5] and was eventually included in the HM reference encoder. The authors suggest to replace BAC by tables with precalculated values of symbol self information for each probability state of adaptive BAC. The simplification does not impact rate-distortion performance while reducing the encoding time by 1% to 5%. However the main difficulty of storing and manipulating BAC probability states remains.

There are several works for H.265/HEVC, where authors suggest to replace bit counting procedure with empirical approximation of the number of bits for each coding option. For example, in [6] the authors perform approximation of the number of bits for residue information using logarithmic regression model. On the JCT-VC test video sequences set [3] with the HM v.6.1 reference encoder they get 8.7% bitrate overhead with execution time savings of 32%.

In [11] the authors partially replace BAC from bit counting procedure of HM v.8.0 with the empirical approach to bit size estimation of the residual information. The approach reduces RDO time (not the encoding time) by 46% providing the average compression bitrate overhead of 1.93%.

Most of such works tend to concentrate the efforts on the "BAC-free" estimation of residue information, while proper estimation of prediction information is either neglected [6] or is still performed with BAC [11].

In our work we elaborate the proper prediction header bits counting algorithm without the use of BAC. The remainder of this paper is organized as follows. In Section 2 we provide a brief overview of RDO coding decision to show the part of this process where our estimation of prediction header size is applicable. In Section 3 we get into detailed investigation of arithmetic coding of intra prediction header to eliminate BAC from this part of rate estimation. In Section 4 we describe the proposed algorithm to count bits of intra prediction header and provide experimental results on compression efficiency losses. In all our experiments we use JCT-VC test video sequence set [3, 4] and HM reference encoder v.13.0 [16]. For completeness we use the new Class E test sequences [4] and the old Class E* test sequences [3].

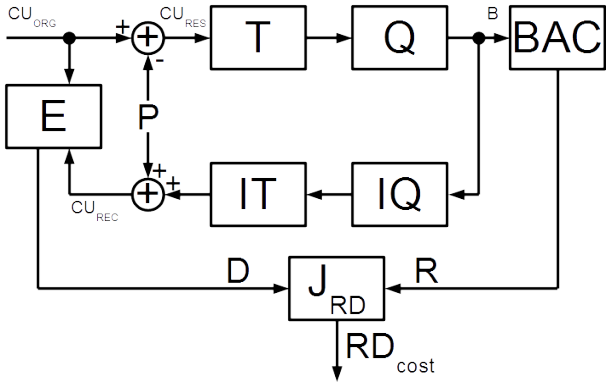
2 RDO Coding Decisions

The general compression dataflow of HEVC standard consists of partitioning of each frame of a video sequence on a series of coding tree units or CTUs. Each CTU covers a 64×64 pixels region of a video frame and is basically just a coding unit (CU). The adaptive quadtree partitioning of CTU, introduced in HEVC, makes it possible to optionally partition $2N \times 2N$ pixels CU on four $N \times N$ subCUs. The partitioning can continue till the minimum CU

size is reached thus forming a quadtree-like partitioning structure. Each CU, produced as a result of a partitioning, can be coded using a plenty of coding options. Basically, each coding option corresponds to the prediction used. In this paper we narrow down to investigate only intra prediction.

There are a total of 35 intra prediction modes available in HEVC. Only 33 of these modes are directional interpolation of the neighboring pixels values, while the rest are Planar and DC modes. A comprehensive overview of HEVC intra prediction can be found in [9].

Figure 1: Rate-distortion optimization (RDO) block estimation data-flow



For a video encoder to properly select CTU partitioning and coding option for each CU, rate-distortion optimization is involved. The technique is based on estimation of the compression rate (R) and the compression distortion (D). Block compression options are compared with each other through the Lagrangian RDO cost [1]:

$$J_{RD} = D + \lambda R, \tag{1}$$

where λ is the Lagrangian multiplier. Its value is usually determined empirically like [12, 13, 15]:

$$\lambda = 0.85 \cdot 2^{(QP-12)/3},$$

where QP is the quantization parameter used in compression data-flow. The RDO estimates the trade-off between compression rate and compression distortion thus providing a criterion of choosing the best coding option.

General RDO estimation dataflow is illustrated on Fig. 1. Initial block prediction P is subtracted from the original block CU_{ORG} being compressed. Residuals CU_{RES} are subject for transform (T) and quantization (Q). Encoder should compress data B with the help of binary arithmetic coder (BAC) in order to estimate compression bit rate R. Meanwhile, compression distortion D involves inverse quantization (IQ) and inverse transform (IT). Restored

residuals are summed up with the block prediction P to get the reconstructed block CU_{REC}. Distortion metric D is based on comparison of the initial block CU_{ORG} and the reconstructed block CU_{REC}. Estimated values R and D are used to get the rate-distortion cost RD_{cost} of a block coding decision.

Elimination of BAC from rate estimation requires proper handling of the prediction information and the residual data after transform (T) and quantization (Q). While rate estimation of the residual data is a subject of several works [6, 11, 10], in this paper we focus on the reliable estimation of the number of bits in intra prediction header of HEVC.

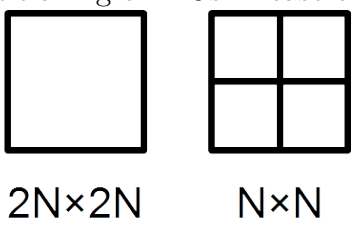
3 Arithmetic Coding of Intra Prediction Header

The information on CU intra prediction signals the partitioning on prediction units or PUs. Each PU is a region of pixels on a video frame to perform the single type of prediction. PU indicates intra prediction mode for luma and chroma samples separately. We further investigate the arithmetic coding of the mentioned data.

3.1 Partitioning Information

In case of intra prediction there are two possible CU partitioning on PUs: $2N \times 2N$ or $N \times N$ (Fig. 2). The $2N \times 2N$ partitioning forms one PU with the size of the CU, while the $N \times N$ partitioning forms four PUs with the half size of the CU being split. However, the $N \times N$ intra partitioning is only possible for the bottom-level CU of a minimal size within the given coding configuration.

Figure 2: CU partitioning on PUs in case of intra prediction



The minimum size the CU can have is 8×8 pixels. In the case of $2N \times 2N$ partitioning one PU covers the whole 8×8 CU, and in case of $N \times N$ partitioning four 4×4 PUs cover the 8×8 CU. It is also possible to restrict the minimum CU size to 16×16 or even 32×32, but this configuration does not find real applications.

The arithmetic coding of CU partitioning makes sense only for the CUs of the bottom level of the partitioning quadtree, i.e. 8×8 CUs. The SBAC

codes one context-dependent symbol to indicate the PU partition mode. This symbol is referred by the H.265/HEVC specification [8] as *part_mode*. The value of 1 indicates the $2N \times 2N$ partitioning, while the value of 0 indicates the $N \times N$ partitioning.

Table 1: The average number of bits to represent CU partitioning mode

Class		Resolution	$2N \times 2N$	$N \times N$
A	Traffic	2560×1600	0.63	2.07
	PeopleOnStreet		0.72	1.86
	Nebuta		0.29	3.12
	SteamLocomotive		0.75	1.51
B	Kimono	1920×1080	0.89	1.48
	ParkScene		0.78	1.58
	Cactus		1.58	0.74
	BQTerrace		0.51	2.22
	BasketballDrive		0.51	2.26
C	RaceHorses (C)	832×480	1.05	1.18
	BQMall		0.50	2.36
	PartyScene		0.40	2.44
	BasketballDrill		0.15	3.89
D	RaceHorses (D)	416×240	0.54	2.01
	BQSquare		0.35	2.52
	BlowingBubbles		0.54	2.15
	BasketballPass		0.50	2.22
E	FourPeople	1280×720	1.38	0.94
	Johnny		0.85	1.33
	KristenAndSara		0.66	1.69
E*	Vidyo1	1280×720	1.42	0.87
	Vidyo3		0.76	1.64
	Vidyo4		0.31	2.93
F	BaskeballDrillText	832×480	0.43	2.40
	ChinaSpeed	1024×768	0.36	2.74
	SlideEditing	1280×720	0.33	2.77
	SlideShow		0.34	2.72
Average			**0.65**	**2.06**

With the reference HM encoder v. 13.0 we carried out an experiment to calculate the average number of bits used to represent the *part_mode* values 0 and 1. This estimation, in fact, corresponds to the average entropy of the bin value in the coded video sequence. Table 1 provides the results of the experiments. Obviously $2N \times 2N$ is the most probable partitioning mode for the bottom level CU as soon as the self information of the corresponding bin

value is less then the self information of $N \times N$ partitioning bin value. On average indication of $2N \times 2N$ partitioning takes 0.65 bits in the coded sequence, while indication of $N \times N$ partitioning takes 2.06 bits. However, on several video sequences the situation is different. For the sequences **Cactus**, **Kimono** and **FourPeople** $N \times N$ partitioning produces less bits per *part_ mode* bin compared to $2N \times 2N$ partitioning. At the same time Class C video sequence **RaceHorses** produces almost identical number of bits for both partitioning modes. Those video sequences have more details and 4×4 pixels prediction is used more often than in other sequences. Therefore the obtained estimation of the *part_ mode* bin entropy when used in RDO will increase the RD_{cost} of 4×4 prediction units and they will be chosen less often. This influence of RD_{cost} might change the frequency of $N \times N$ partitioning selection and provide higher bitrate overhead for the mentioned video sequences.

3.2 Intra Prediction Mode for Luma Samples

Each PU specifies the intra mode used for prediction of luma samples within the PU. There are a total of 35 possible intra prediction modes. Without arithmetic coding the indication of the chosen intra prediction mode would take 6 bits in the bitstream. The bits of intra prediction mode form the most part of the coded intra prediction header and should be handled accurately.

The SBAC distinguishes intra prediction modes on the most probable (MPM) and the rest. To simplify the presentation of the arithmetic coding we will refer those "not MPM" as the least probable modes or LPM.

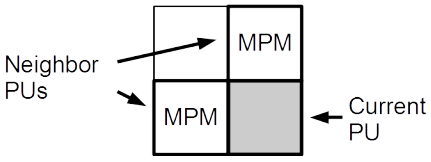


Figure 3: Illustration of the the neighbor PUs, whose intra prediction modes are MPM for the current PU

A list of MPM contains three elements. The algorithm of determining the most probable modes is given in section 8.4.2 "Derivation process for luma intra prediction mode" of the specification [8] and altered in Algorithm 1. Generally MPM list depends on whether the left and the above neighbor PUs (Fig. 3) are available for the current PU. Their intra prediction modes form two elements of the MPM list. Unavailable neighbor PUs or neighbor PUs with inter prediction are substituted with the DC predictor.

In case the two intra prediction modes are similar and refer to the angular prediction modes the MPM list is formed the following way. The first element is the predicted mode. The rest elements are formed as described in Algorithm 1.

Algorithm 1 Derivation process for luma intra prediction mode

procedure GETINTRADIRLUMAPREDICTOR($LeftPu$, $AbovePu$)
  $PLANAR \leftarrow 0$
  $DC \leftarrow 1$
  $VER \leftarrow 26$
  if $LeftPu.IsAvailable()$ AND $LeftPu.IsIntra()$ **then**
    $LeftIntraDir \leftarrow LeftPu.IntraDir$
  else
    $LeftIntraDir \leftarrow DC$
  end if
  if $AbovePu.IsAvailable()$ AND $AbovePu.IsIntra()$ **then**
    $AboveIntraDir \leftarrow AbovePu.IntraDir$
  else
    $AboveIntraDir \leftarrow DC$
  end if
  if $LeftIntraDir = AboveIntraDir$ **then**
    if $LeftIntraDir > DC$ **then**
      $MPM[0] \leftarrow LeftIntraDir$
      $MPM[1] \leftarrow ((LeftIntraDir + 29) >> 6) + 2$
      $MPM[2] \leftarrow ((LeftIntraDir + 29 - 1) >> 6) + 2$
    else
      $MPM[0] \leftarrow PLANAR$
      $MPM[1] \leftarrow DC$
      $MPM[2] \leftarrow VER$
    end if
  else
    $MPM[0] \leftarrow LeftIntraDir$
    $MPM[1] \leftarrow AboveIntraDir$
    if $LeftIntraDir \neq 0$ AND $AboveIntraDir \neq 0$ **then**
      $MPM[2] \leftarrow PLANAR$
    else if $LeftIntraDir + AboveIntraDir < 2$ **then**
      $MPM[2] \leftarrow VER$
    else
      $MPM[2] \leftarrow DC$
    end if
  end if
end procedure

The following corresponds to the case when predicted intra modes are not similar. If none of the neighbor modes are planar, the planar intra prediction mode forms the third element of the list. Otherwise if the second intra mode is angular, the DC intra mode forms the third element of the list, and if the

second intra mode is DC, then vertical intra prediction mode 26 is included in the list.

Table 2: The average number of bits to indicate the MPM or LPM prediction mode

Class	Sequence	Resolution	*prev_intra_luma_pred_flag*	
			1 (MPM)	0 (LPM)
A	Traffic	2560×1600	0.52	1.95
	PeopleOnStreet		0.64	1.63
	Nebuta		0.63	1.66
	SteamLocomotive		0.36	2.50
B	Kimono	1920×1080	0.41	2.23
	ParkScene		0.42	2.35
	Cactus		0.53	1.93
	BQTerrace		0.52	2.02
	BasketballDrive		0.48	2.15
C	RaceHorses (C)	832×480	0.67	1.59
	BQMall		0.61	1.79
	PartyScene		0.84	1.37
	BasketballDrill		0.62	1.74
D	RaceHorses (D)	416×240	0.84	1.29
	BQSquare		0.98	1.15
	BlowingBubbles		0.80	1.35
	BasketballPass		0.63	1.67
E	FourPeople	1280×720	0.53	1.87
	Johnny		0.48	2.06
	KristenAndSara		0.57	1.77
E*	Vidyo1	1280×720	0.55	1.88
	Vidyo3		0.50	2.03
	Vidyo4		0.49	2.01
F	BaskeballDrillText	832×480	0.61	1.79
	ChinaSpeed	1024×768	0.54	1.92
	SlideEditing	1280×720	0.43	2.24
	SlideShow		0.37	2.41
Average			**0.58**	**1.86**

The arithmetic coding of the PU intra prediction mode starts with the context-dependent bin referred as *prev_intra_luma_pred_flag* [8]. When this flag is equal to 1, the intra prediction mode is included in the MPM list. When the flag is equal to 0, the intra prediction mode is LPM. It takes 1–2 bypass bins to code the index (*mpm_idx)* of the corresponding intra mode in the MPM list. In case of LPM it takes 5 bypass bins to code

the remainder (*rem_intra_luma_pred_mode*) of intra prediction mode: given that the selected prediction mode is not one of the three MPM modes, only $35 - 3 = 32 = 2^5$ possible modes are left.

It is not a problem to count the number of bypass-coded bins that directly corresponds to the number of bits in the coded sequence. Once we don't have BAC, the accurate bit counting for the context-dependent bin *prev_intra_luma_pred_flag* is of a high value.

With the reference HM encoder v. 13.0 we calculated the average number of bits used to represent the *prev_intra_luma_pred_flag* value 0 and 1 in the coded video sequences from the JCT-VC test set [3, 4]. This estimation also corresponds to the average entropy of the bin value in the coded video sequence. Table 1 provides the results of the experiments. Indication of the MPM is more often for most of the test video sequences, providing the average self information of *prev_intra_luma_pred_flag=1* equal to 0.58. There are no cases in the test video sequences set for which LPM is more probable. However, the sequences **PartyScene**, **RaceHorses (D)**, **BQSquare** and **BlowingBubbles** have almost equal probability of MPM and LPM. The replacement of BAC-based estimation with the precalculated entropy estimation should force the encoder to increase the selection of MPM in RDO process for those sequences.

3.3 Intra Prediction Mode for Chroma Samples

Intra prediction of chroma samples is performed separately to the prediction of luma samples. Intra prediction for chroma planes is indicated by the first PU in a CU in case of 4:2:0 subsampling. However, chroma intra prediction mode depends on the one chosen for luma prediction and is indicated in the syntax element *intra_chroma_pred_mode* of the H.265/HEVC video sequence [8]. The first bin of *intra_chroma_pred_mode* is context-dependent. The value of 0 unambiguously means that chroma intra prediction mode is identical to luma intra prediction mode of the first PU within a CU and no other bins follow. The value of 1 means that chroma intra prediction mode is different to luma intra prediction mode. In this case its value can be planar, DC, vertical, horizontal or vertical down-left (mode 34) with respect to the table "Specification of IntraPredModeC" of the H.265/HEVC standard [8]. In this case the context-dependent bin is followed by the two bypass-coded bins. The described algorithm of *intra_chroma_pred_mode* coding provides five possible intra modes for chroma prediction. To substitute the arithmetic coding of context-dependent bin in RDO we collected statistics on the average entropy of both *intra_chroma_pred_mode* values 0 and 1. The results are given in Table 3. The average self information of *intra_chroma_pred_mode=0* is 0.36, which means this value is the most probable and the most commonly used.

Table 3: The average number of bits to indicate the chroma prediction mode

Class	Sequence	Resolution	*intra_chroma_pred_mode*	
			0 (as luma)	1
A	Traffic	2560×1600	0.33	3.05
	PeopleOnStreet		0.22	3.68
	Nebuta		0.52	1.97
	SteamLocomotive		0.24	3.84
B	Kimono	1920×1080	0.53	2.12
	ParkScene		0.38	3.06
	Cactus		0.39	2.93
	BQTerrace		0.25	3.61
	BasketballDrive		0.31	3.03
C	RaceHorses (C)	832×480	0.62	1.94
	BQMall		0.46	2.61
	PartyScene		0.70	2.04
	BasketballDrill		0.25	3.31
D	RaceHorses (D)	416×240	0.76	1.54
	BQSquare		0.33	3.06
	BlowingBubbles		0.49	2.48
	BasketballPass		0.37	2.45
E	FourPeople	1280×720	0.23	3.59
	Johnny		0.22	3.49
	KristenAndSara		0.26	3.26
E*	Vidyo1	1280×720	0.17	3.93
	Vidyo3		0.14	4.22
	Vidyo4		0.18	3.88
F	BaskeballDrillText	832×480	0.33	2.99
	ChinaSpeed	1024×768	0.32	3.08
	SlideEditing	1280×720	0.50	2.87
	SlideShow		0.14	4.18
Average			**0.36**	**3.04**

4 Results and Discussion

Our investigation on the entropy of the context-dependent bins involved in arithmetic coding of intra prediction header results in elimination of SBAC-based counting of intra prediction header bits. The algorithm follows the steps of BAC coding except the coding itself does not happen. Bit counting after arithmetic coding of context-dependent bins of intra prediction header is replaced with the average entropy values for bit counts obtained in Section 3.

The bypass-coded bins take exactly one bit each in the coded sequence and their size is counted accordingly.

We implemented our algorithm in the HM reference encoder v. 13.0 and used it in RDO-based intra coding decisions. We compared the resulted compression to the compression efficiency of the original HM encoder. The comparisons were made with the Bjontegaard delta rate [2]. The "Intra Main" configuration was selected.

Table 4 shows the loss of the compression efficiency of the HM reference encoder with the proposed bit counting procedure for intra prediction header on JCT-VC test video sequences set [3, 4]. Class E* test sequences correspont to the older test set [3], but we included them in our experiments for completeness of the obtained results. BD-RATE column corresponds to the Bjontegaard delta rate with regard to average $PSNR$ value of luma and chroma. The combined $PSNR$ ($PSNR_{YUV}$) is calculated as the weighted sum of the $PSNR$ per picture of the individual components ($PSNR_Y$, $PSNR_U$, and $PSNR_V$) [7]:

$$PSNR_{YUV} = (6 \cdot PSNR_Y + PSNR_U + PSNR_V)/8,$$

where $PSNR_Y$, $PSNR_U$ and $PSNR_V$ are each computed as

$$PSNR = 10 \cdot log_{10}\left(\frac{2^B - 1}{MSE_{AVG}}\right),$$

with $B = 8$ and $B = 10$ is the number of bits per sample of the video signal to be coded for 10-bit (Nebuta and SteamLocomotive) and 8-bit video sequences, and MSE_{AVG} is the average MSE value for the video frames

$$MSE_{AVG} = \frac{1}{N} \cdot \sum_{i=1}^{N} MSE_i,$$

where N is the number of video frames in a sequence, and the MSE is the SSD divided by the number of samples in the signal [7].

We also evaluate BD-RATE (Y) with regard to $PSNR_Y$ value of luma color component. Furthermore we calculate BD-PSNR (Y) and BD-PSNR (UV) as an average delta PSNR for luma and chroma components respectively.

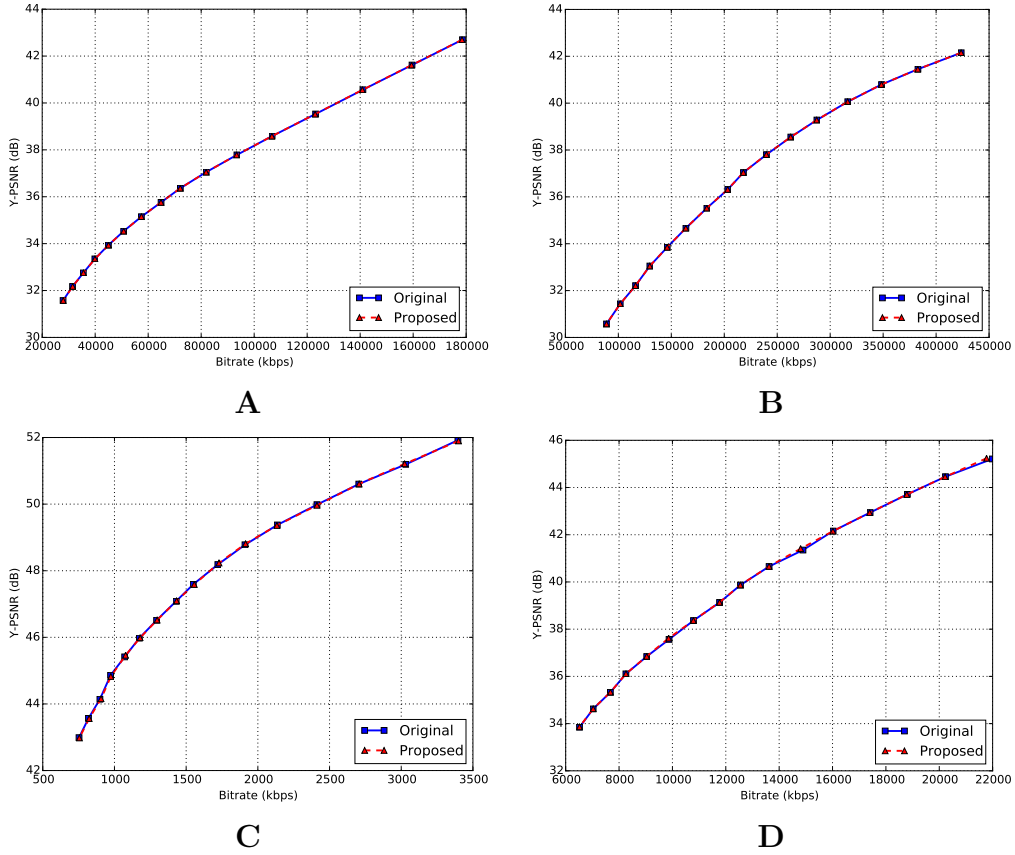
As far as the compression efficiency of the modified HM encoder is very close to the original, the Bjontegaard delta rate metric starts showing incorrect results [14] when only four quantization parameters are used as suggested in [4]. Therefore we obtained results for all quantization parameters in range $[22; 37]$ and performed rd-curve approximation using the 4^{th} order polynomial. This resulted in more reliable results given in Table 4. The highest bitrate overhead of 0.26% is on SlideShow sequence. The compression of sequences SteamLocomotive, ParkScene, BasketballDrive, Vidyo1, Vidyo3 and Vidyo4 provides bitrate overhead of 0.15%–0.18%.

Table 4: Compression efficiency change of the proposed prediction header size estimation algorithm

Class	Sequence	Resolution	HM INTRA MAIN			
			BD-RATE, %	BD-RATE (Y), %	BD-PSNR (Y), dB	BD-PSNR (UV), dB
A	Traffic	2560×1600	0.10	0.07	0.00	-0.01
	PeopleOnStreet		0.07	0.11	-0.01	0.00
	Nebuta		0.00	-0.02	0.00	-0.01
	SteamLocomotive		0.18	0.18	-0.01	0.00
B	Kimono	1920×1080	0.10	-0.01	0.00	-0.02
	ParkScene		0.18	0.16	-0.01	-0.01
	Cactus		0.10	0.10	0.01	0.00
	BQTerrace		0.03	0.01	0.00	0.00
	BasketballDrive		0.18	0.09	0.00	-0.01
C	RaceHorses (C)	832×480	0.05	-0.06	0.00	-0.02
	BQMall		0.07	0.00	0.00	-0.02
	PartyScene		0.02	-0.06	0.0	-0.02
	BasketballDrill		0.03	0.00	0.00	-0.01
D	RaceHorses (D)	416×240	0.07	-0.07	0.00	-0.03
	BQSquare		0.06	0.00	0.00	-0.02
	BlowingBubbles		0.04	-0.08	0.01	-0.02
	BasketballPass		0.11	0.01	0.00	-0.02
E	FourPeople	1280×720	0.08	0.09	-0.01	0.00
	Johnny		0.10	0.10	0.00	0.00
	KristenAndSara		0.07	0.03	0.00	-0.01
E*	Vidyo1	1280×720	0.17	-0.01	0.01	0.01
	Vidyo3		0.15	-0.01	0.01	-0.02
	Vidyo4		0.15	-0.01	0.01	-0.03
F	BaskeballDrillText	832×480	0.06	-0.07	0.00	-0.02
	ChinaSpeed	1024×768	-0.14	-0.23	0.02	-0.01
	SlideEditing	1280×720	0.10	0.00	0.00	-0.03
	SlideShow		0.26	0.18	-0.01	-0.02
Average			**0.09**	**0.02**	**0.00**	**-0.01**

Fig. 4 provides sample RD-plots for several test sequences. Illustration of RD-plots for the sequences BQTerrace (Fig. 4A) and Nebuta (Fig. 4B) is recommended in [14]. The bitrate overhead for this sequences is almost absent. The highest compression efficiency loss of the HM encoder is 0.26% on SlideShow test sequence (Fig. 4C). This value of BD-Rate is due to the points of RD curve have a small upward offset along the curve. The BD-Rate estimation of the compression efficiency of ChinaSpeed (Fig. 4D) video sequence increased, but in fact only two RD points can be found above the RD curve for the original HM encoder. Those points correspond to 45.2 dB and 41.4 dB for luma plane.

Figure 4: RD-plots for (**A**) BQTerrace, (**B**) Nebuta, (**C**) SlideShow and (**D**) ChinaSpeed test sequences



5 Conclusion

We proposed intra prediction header bits counting algorithm for HEVC without the use of BAC. The provided bitrate overhead of 0.09% can be considered insignificant. This algorithm may underlie a more complex algorithm

of BAC-free RDO estimation to reduce the computational complexity of HEVC compression system. Furthermore, the similar approach can be used to elaborate on bit counting procedure of inter prediction header.

Acknowledgements. The results were obtained at Tomsk State University of Control Systems and Radioelectronics as part of the complex project 'Provision of multimedia broadcasting services in Internet public networks, based on peer-to-peer network technology and adaptive data streaming' with the financial support of the Ministry of Education and Science of the Russian Federation.

References

[1] T. Berger Rate Distortion Theory: Mathematical Basis for Data Compression (Prentice-Hall series in information and system sciences); Prentice-Hall: Endlewood Cliffs, NJ, 1971; p. 352.

[2] G. Bjontegaard. Improvements of the BD-PSNR model. ITU-T SC16/Q6, 35th VCEG Meeting, Berlin, Germany, 16-18 July, 2008, Doc. VCEG-AI11.

[3] F. Bossen. Common test conditions and software reference configurations. ITU-T SC16/Q6, 2nd JCT-VC Meeting, Geneva, Switzerland, 21-28 July, 2010, Doc. JCTVC-B300.

[4] F. Bossen. Common test conditions and software reference configurations. ITU-T SC16/Q6, 11th JCT-VC Meeting, Shanghai, China, 10-19 October, 2012, Doc. JCTVC-K1100.

[5] F. Bossen Table-based bit estimation for CABAC. In *Document of ITU-T Q.6/SG16 JCTVC-G763*. ITU-T: Geneva, CH, 2011.

[6] S. Johar and M. Alwani. Method for fast bits estimation in rate distortion for intra coding units in hevc. *In 2013 IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 721–724, Jan 2013. http://dx.doi.org/10.1109/ccnc.2013.6488534

[7] J. Ohm; G. Sullivan; H. Schwarz; T.K. Tan; T. Wiegand. Comparison of the Coding Efficiency of Video Coding Standards - Including High Efficiency Video Coding (HEVC). In *IEEE Transactions on Circuits and Systems for Video Technology*, **2012**, Vol. *22*, No. 12, pp. 1669–1684. http://dx.doi.org/10.1109/tcsvt.2012.2221192

[8] Recommendation ITU-T H.265: High Efficiency Video coding, 2013.

[9] M.P. Sharabayko; O.G. Ponomarev; R.I. Chernyak. Intra Compression Efficiency in VP9 and HEVC. *Applied Mathematical Sciences* **2013**, *7*, 6803-6824. http://dx.doi.org/10.12988/ams.2013.311644

[10] M.P. Sharabayko and N.G. Markov. Entropy-based intra-coding RDO estimation for HEVC. In *9th International Forum on Strategic Technology (IFOST)*. University of Engineering and Technology: Chittagong, Bangladesh, 21-23 October, 2014.

[11] Z. Sheng; D. Zhou; H. Sun; S. Goto. Low-Complexity Rate-Distortion Optimization Algorithms for HEVC Intra Prediction. In *MultiMedia Modeling*; Springer International Publishing, 2014; Vol. 8325, *Lecture Notes in Computer Science*, pp. 541-552. http://dx.doi.org/10.1007/978-3-319-04114-8_46

[12] T. Stockhammer; D. Kontopodis; T. Wiegand, T. Rate-distortion optimization for JVT/H.26L video coding in packet loss environment. In *International Packet Video Workshop on Packet Loss Environment*; Munich University of Technology: Munich, Germany, 2002; pp. 1-12.

[13] G.J. Sullivan and T. Wiegand. Rate-distortion optimization for video compression. *Signal Processing Magazine, IEEE* **1998**, *15*, 74–90. http://dx.doi.org/10.1109/79.733497

[14] J. Wang, X. Yu, and D. He (RIM). On BD-Rate calculation. ITU-T SC16/Q6, 6th JCT-VC Meeting, Torino, Italy, 14-22 July, 2011, Doc. JCTVC-F270.

[15] T. Wiegand, B. Girod. Lagrange multiplier selection in hybrid video coder control. In *International Conference on Image Processing*. IEEE: Thessaloniki, Greece, 2001, Vol. 3, pp. 542-545. http://dx.doi.org/10.1109/icip.2001.958171

[16] `https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/`. HEVC software repository.

Received: November 25, 2014; Published: December 12, 2014