SIViP (2015) 9:1067–1079
DOI 10.1007/s11760-013-0542-2

ORIGINAL PAPER

Improved entropy coding for quantized transform coefficients in HEVC screen content coding

Jung-Ah Choi · Yo-Sung Ho

Received: 7 December 2012 / Revised: 28 March 2013 / Accepted: 28 July 2013 / Published online: 5 September 2013

Abstract In the emerging high efficiency video coding (HEVC) standard, a Golomb-Rice code is adopted to binarize level information. The Golomb-Rice code is optimal for encoding symbols with the exponential probability distribution. In general, quantized transform coefficients of natural images show the exponential probability distribution. However, screen contents usually have a totally different probability distribution, and the Golomb-Rice code is not appropriate for screen content coding. In this paper, we focus on a new entropy coding scheme for screen content coding. In the proposed scheme, we clip inefficient high magnitude coefficient levels that do not fit the exponential probability distribution. Then, we apply a limited length Golomb-Rice code to binarize clipped levels. From experimental results, it is verified that the proposed method achieves on average 0.60 % BD-rate saving and up to 1.13 % BD-rate saving, compared to HEVC screen content coding. When the proposed method is combined with a well-known screen content coding technique, transform skipping, it shows up to 24.02 % BD-rate saving.

Keywords High efficiency video coding (HEVC) · Screen content coding · Context-based adaptive binary arithmetic coding (CABAC) · Level coding · Golomb-Rice code

J.-A. Choi (✉) · Y.-S. Ho
Gwangju Institute of Science and Technology (GIST),
123 Cheomdan-gwagiro, Buk-gu, Gwangju 500-712, Korea
e-mail: jachoi@gist.ac.kr

Y.-S. Ho
e-mail: hoyo@gist.ac.kr

1 Introduction

Screen contents represent images or videos rendered by electronic devices such as computers or mobile phones. Screen contents have a large number of target applications: desktop sharing, video conferencing, remote desktop, remote education, e-books, and computer games. As these applications are widely used, it also becomes important to record, store, and transmit the screen contents. Among these technologies, efficient solutions for screen contents compression would be an important contribution to the market needs.

High efficiency video coding (HEVC) [1] is a new video compression standard that is developed by joint collaborative team on video coding (JCT-VC) of ISO/IEC moving picture experts group (MPEG) and ITU-T video coding experts group (VCEG). HEVC is designed for various multimedia applications including the ultra-high-definition television (UHDTV), low-delay communications, mobile video services, streaming, and storage-based video applications [2].

In the 4th JCT-VC meeting, National Bodies (NBs) of Finland, China, and USA consider screen content coding within the scope of HEVC. After discussion, it was agreed that screen content is considered within the scope. As a result, ad hoc group was established to find out a standard solution for screen content coding. To date, HEVC experts have focused their efforts on the novel coding techniques for screen contents.

As in any video coding technologies, entropy coding is an essential part in HEVC. Context-based adaptive binary arithmetic coding (CABAC) [3] is a form of entropy coding used in H.264/AVC [4] and also in HEVC. CABAC typically provides better compression efficiency than variable length coding (VLC)-based entropy coders such as low complexity entropy coding (LCEC) [5] and context-based adaptive variable length coding (CAVLC) [6]. The usage of arithmetic

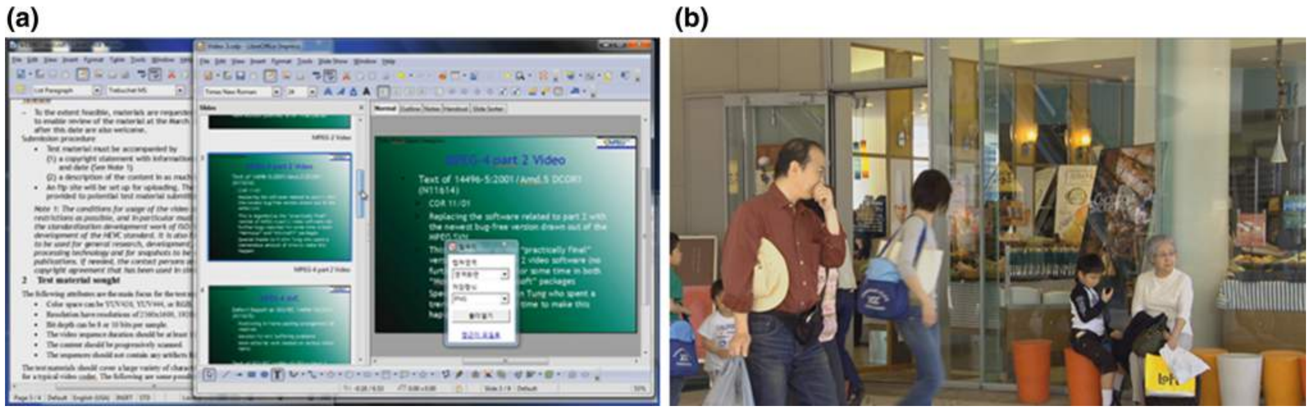

Fig. 1 Comparison of texture characteristics between the screen content and the natural image. **a** Screen content. **b** Natural image

Table 1 CABAC syntax elements for residual data coding

Syntax element	Description
$significant\_coeff\_flag$	indicates the significance of each coefficient
$coeff\_abs\_level\_greater1\_flag$	indicates whether the coefficient amplitude is larger than one for each nonzero coefficient (i.e., with $significant\_coeff\_flag$ as '1')
$coeff\_abs\_level\_greater2\_flag$	indicates whether the coefficient amplitude is larger than two for each coefficient with amplitude larger than one (i.e., with $coeff\_abs\_level\_greater1\_flag$ as '1')
$coeff\_sign\_flag$	indicates sign information of the nonzero coefficients
$coeff\_abs\_level\_remaining$	indicates remaining absolute value of a transform coefficient level that is coded with Golomb-Rice code

Fig. 2 The encoding process of CABAC for coefficient level coding in HEVC

18	6	-6	-1
-12	4	-4	0
7	4	2	1
2	4	-1	0

4×4 sub-block

16	14	11	7
15	12	8	4
13	9	5	2
10	6	3	1

Coding order

	0	1	-1	0	2	4	-1	-4	4	2	-6	4	7	6	-12	18
significant_coeff_flag	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
coeff_abs_level_greater1_flag		0	0		1	1	0	1	1	1						
coeff_abs_level_greater2_flag					0											
coeff_sign_flag		0	1		0	0	1	1	0	0	1	0	0	0	1	0
coeff_abs_level_remaining						2		2	2	0	5	3	6	5	11	17

coding allows the assignment of a noninteger number of bits to each symbol, which is extremely beneficial for symbol probabilities that are greater than 0.5 [7]. Thus, CABAC is adopted as the entropy coding tool in HEVC.

Compared to natural images, screen contents usually have totally different texture characteristics, as shown in Fig. 1. Accordingly, the prediction error in screen contents, i.e., residual data, has different statistical characteristics with that in natural videos. Current HEVC entropy coding is developed focusing on natural video coding, and it is not appropriate screen content coding. Hence, in this paper, we attempt to improve the coding performance of residual data coding for screen contents. By considering statistical properties of residual data in screen content coding, we propose an efficient and simple level coding technique using clipping and a new binarization method with the limited codeword length. Note that our research goal is to improve screen content coding performance of CABAC, which can then be easily applied

Table 2 The Golomb-Rice code

Symbol	Codeword				
	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$
0	0	0.0	0.00	0.000	0.0000
1	10	0.1	0.01	0.001	0.0001
2	110	10.0	0.10	0.010	0.0010
3	1110	10.1	0.11	0.011	0.0011
4	11110	110.0	10.00	0.100	0.0100
5	111110	110.1	10.01	0.101	0.0101
6	1111110	1110.0	10.10	0.110	0.0110
7	11111110	1110.1	10.11	0.111	0.0111
8	111111110	11110.0	110.00	1.000	1.1000
9	1111111110	11110.1	110.01	1.001	1.1001
10	11111111110	111110.0	110.10	1.010	1.1010
11	111111111110	111110.1	110.11	1.011	1.1011
12	1111111111110	1111110.0	1110.00	1.100	1.1100
13	11111111111110	1111110.1	1110.01	1.101	1.1101
14	111111111111110	11111110.0	1110.10	1.110	1.1110
15	1111111111111110	11111110.1	1110.11	1.111	1.1111
…	…	…	…	…	…

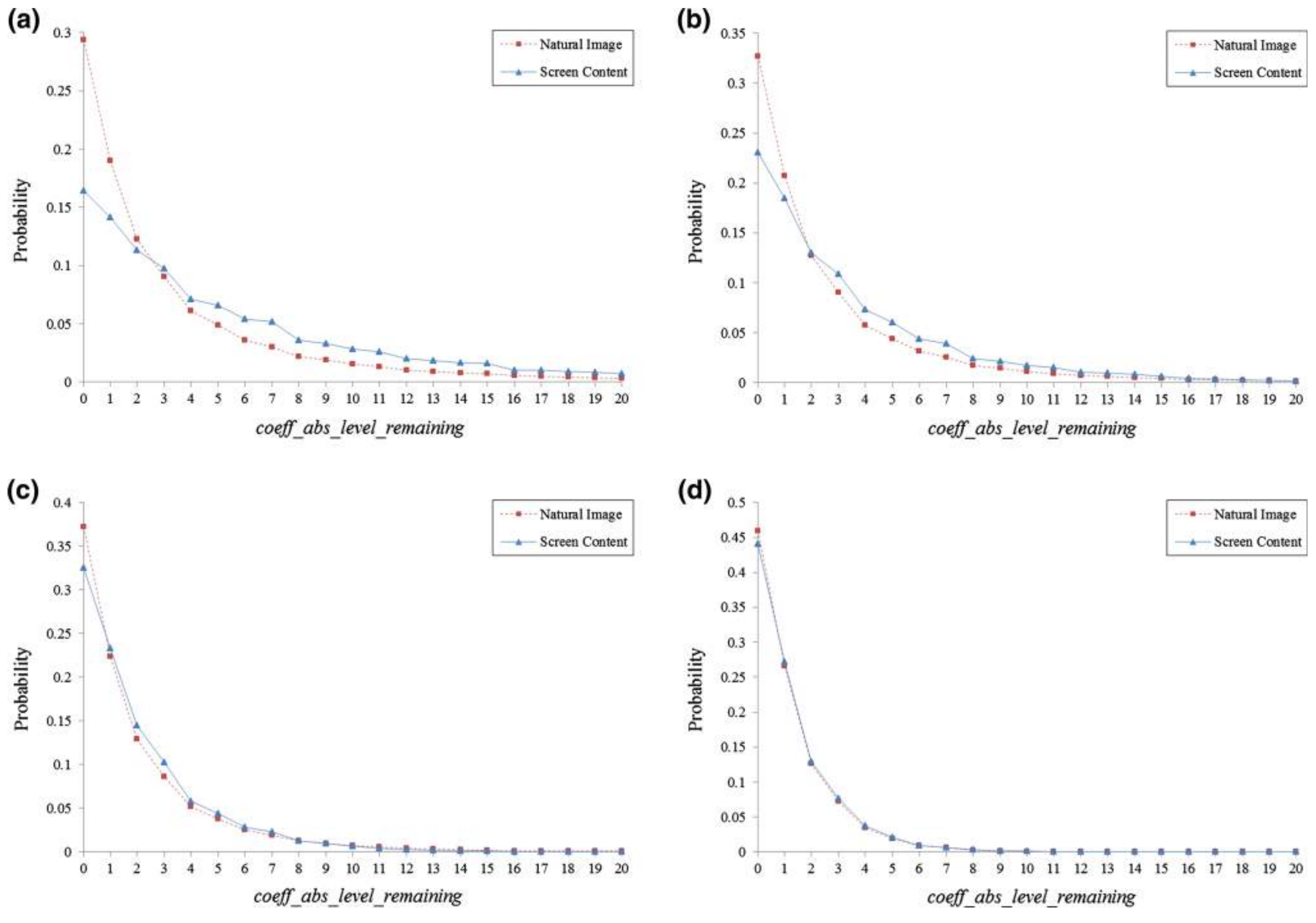

Fig. 3 Comparison of the statistical characteristics between natural video coding and screen content coding for different QP values. **a** QP = 22. **b** QP = 27. **c** QP = 32. **d** QP = 37

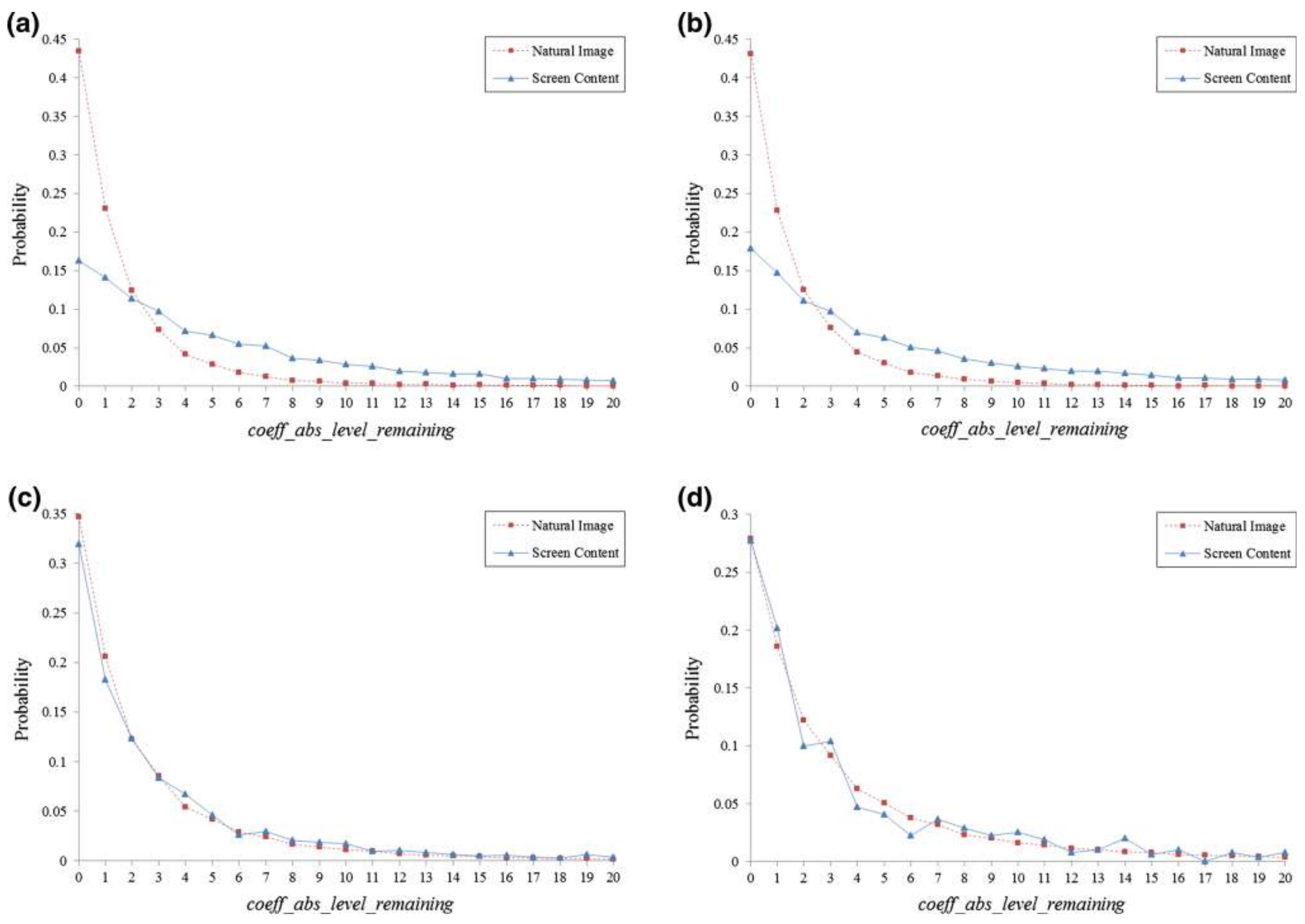

Fig. 4 Comparison of the statistical characteristics between natural video coding and screen content coding for different TU sizes. **a** 4 × 4 TU. **b** 8 × 8 TU. **c** 16 × 16 TU. **d** 32 × 32 TU

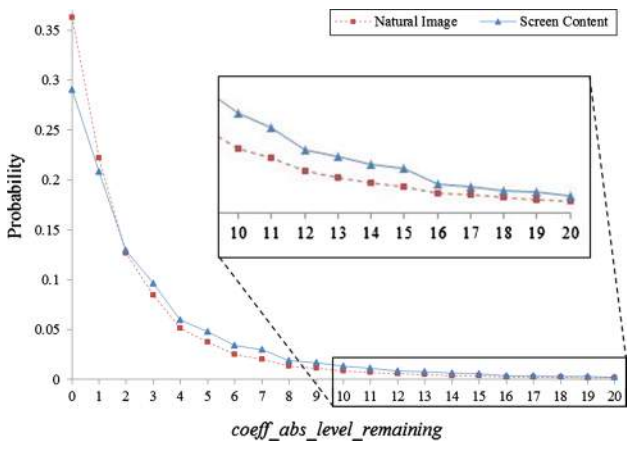

Fig. 5 Average probability distribution of *coeff_abs_level_remaining* in screen content coding and natural video coding

to HEVC by modifying the semantics and coding processes without requiring any additional syntax elements in the current HEVC standard.

The paper is organized as follows. In Sect. 2, coefficient level coding in HEVC is introduced in detail. In Sect. 3, our proposed algorithm is explained. In Sect. 4, experimental results are given, and finally, our conclusion is drawn in Sect. 5.

2 Overview of coefficient level coding in HEVC

Coefficient level coding in HEVC is similar to H.264/AVC. Several modifications are introduced for higher throughput and large transform units (TUs). In HEVC, 16 × 16 TU or 32 × 32 TU is divided into 4 × 4 subsets, and each subset corresponds to a sub-block. Coefficient level coding is conducted in each sub-block unit. A sub-block consists of 16 coefficients, and it is encoded in the inverse diagonal scan order. Five syntax elements described in Table 1 are signaled to represent the coefficients level information within a sub-block [1]. Table 1 also describes semantics of each syntax element.

The syntax elements, $coeff\_abs\_level\_greater1\_flag$ and $coeff\_abs\_level\_greater2\_flag$, do not process all the coefficients in a sub-block to improve the throughput performance [8]. At most eight $coeff_abs_level_greater1_flag$s and one $coeff\_abs\_level\_greater2\_flag$ are coded.

Table 3 The limited length Golomb-Rice code using the natural binary code

Symbol, s	Codeword				
	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
0	0	0.0	0.00	0.000	0.0000
1	10	0.1	0.01	0.001	0.0001
2	110	10.0	0.10	0.010	0.0010
3	1110	10.1	0.11	0.011	0.0011
4	1111.0000	110.0	10.00	0.100	0.0100
5	1111.0001	110.1	10.01	0.101	0.0101
6	1111.0010	1110.0	10.10	0.110	0.0110
7	1111.0011	1110.1	10.11	0.111	0.0111
8	1111.0100	1111.000	110.00	1.000	1.1000
9	1111.0101	1111.001	110.01	1.001	1.1001
10	1111.0110	1111.010	110.10	1.010	1.1010
11	1111.0111	1111.011	110.11	1.011	1.1011
12	1111.1000	1111.100	111.00	1.100	1.1100
13	1111.1001	1111.101	111.01	1.101	1.1101
14	1111.1010	1111.110	111.10	1.110	1.1110
15	1111.1011	1111.111	111.11	1.111	1.1111

Table 4 The example of the adjusted binary code

Symbol, s	Range			
	[0, 4]	[0, 5]	[0, 6]	[0, 7]
0	00	00	00	000
1	01	01	010	001
2	10	100	011	010
3	110	101	100	011
4	111	110	101	100
5	–	111	110	101
6	–	–	111	110
7	–	–	–	111

The values are left to be coded by $coeff\_abs\_level\_remaining$, defined as Eq. (1). Here, *absCoeffLevel* represents the absolute coefficient level. Figure 2 illustrates an example of HEVC level coding for a 4×4 sub-block.

$$coeff\_abs\_level\_remaining = absCoeffLevel - baseLevel \quad (1)$$

$$baseLevel = significant\_coeff\_flag + coeff\_abs\_level\_greater1\_flag + coeff\_abs\_level\_greater2\_flag \quad (2)$$

The syntax element $coeff\_abs\_level\_remaining$ is binarized by the Golomb-Rice code with the Rice parameter k [9]. The motivation of the Golomb-Rice code is to reduce the complexity of the unary/kth order Exp-Golomb code in the previous video coding standard, H.264/AVC. The complexity

Table 5 The limited length Golomb-Rice code using the adjusted binary code

Symbol, s	Codeword				
	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
0	0	0.0	0.00	0.000	0.0000
1	10	0.1	0.01	0.001	0.0001
2	110	10.0	0.10	0.010	0.0010
3	1110	10.1	0.11	0.011	0.0011
4	1111.000	110.0	10.00	0.100	0.0100
5	1111.001	110.1	10.01	0.101	0.0101
6	1111.010	1110.0	10.10	0.110	0.0110
7	1111.011	1110.1	10.11	0.111	0.0111
8	1111.1000	1111.000	110.00	1.000	1.1000
9	1111.1001	1111.001	110.01	1.001	1.1001
10	1111.1010	1111.010	110.10	1.010	1.1010
11	1111.1011	1111.011	110.11	1.011	1.1011
12	1111.1100	1111.100	111.00	1.100	1.1100
13	1111.1101	1111.101	111.01	1.101	1.1101
14	1111.1110	1111.110	111.10	1.110	1.1110
15	1111.1111	1111.111	111.11	1.111	1.1111

problem of the unary/kth order Exp-Golomb code is caused by the adaptive context modeling process. Since the Golomb-Rice code does not require any context modeling, it efficiently reduces the complexity of encoding and decoding processes.

Given a particular Rice parameter k, the Golomb-Rice code for a symbol s consists of two parts: a unary representation of p and a binary representation of r. The relation of p and r is shown in Eq. (3). The unary representation is formed by the p '1's, followed by a '0'. Then, the codeword for r is constructed by appending the k least significant bits of r to the binary representation. The length of the Golomb-Rice code is $k + 1 + p$.

$$p = \left\lfloor \frac{s}{2^k} \right\rfloor \quad \text{where } r = s - p \cdot 2^k \quad (3)$$

The range of the Rice parameter k is from 0 to 4. Overall Golomb-Rice codewords according to each parameter are stated in Table 2. The initial value of k is 0, and it monotonically increases according to the magnitude of the coefficient level.

3 Proposed entropy coding for quantized transform coefficients

3.1 Analysis of probability distribution of coefficient levels

In order to compare the statistical characteristics, we encoded several natural videos (*Traffic*, *Kimono*, and *BQSquare*) and screen contents (*HKUST1*, *BJUT-web*, and *SlideEditing*). Here, HEVC test model (HM) 8.0 [10] was used, and we

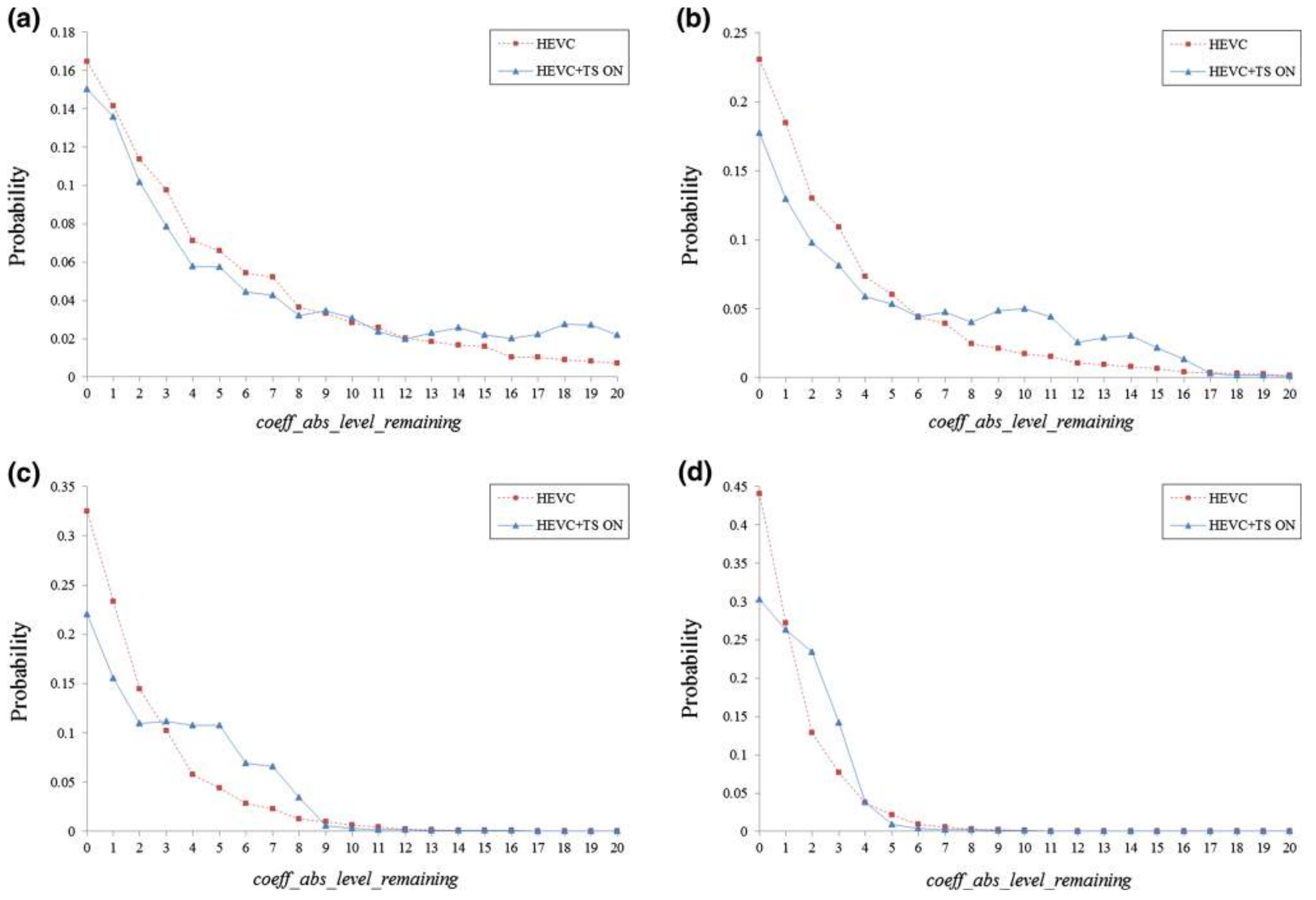

Fig. 6 The probability distribution of having the transform skipping disabled versus enabled for different QPs. **a** QP = 22. **b** QP = 27. **c** QP = 32. **d** QP = 37

Table 6 Encoder parameters

Parameter	Value	Description
MaxCUWidth	64	CU: 8 × 8 ∼ 64 × 64
MaxCUHeight	64	
MaxPartitionDepth	4	
QuadtreeTUMaxDepthIntra	3	TU: 4 × 4 ∼ 32 × 32
IntraPeriod	1	All-intra coding
QP	22, 27, 32, 37	Recommended QP set
SAO	1	SAO on
RDOQ	1	RDOQ on

Table 7 The detail information of test sequences

Sequence	Resolution (pixels)	Frame rate (fps)	Total frame (frames)
HKUST1	1,280 × 720	30	450
HKUST2		30	450
HKUST3		30	450
BJUT-doc		10	400
BJUT-slide		20	500
BJUT-web		10	500
SlideEditing		30	300
ChinaSpeed	1,024 × 768	30	500

tested four different quantization parameters (QPs): 22, 27, 32, and 37. The coding structure was all-intra coding. For other configurations, we followed the recommended test conditions by JCT-VC [11].

In Fig. 3, probability distributions of $coeff\_abs\_level\_remaining$ for natural videos and screen contents are shown. The statistics of $coeff\_abs\_level\_remaining$ in screen content coding are different from those of $coeff\_abs\_level\_remaining$ in natural video coding. Obviously, the probability distribution of natural video coding takes on the character of exponentially decaying distributions. However, probability distribution of screen content coding shows wider distributions than that of natural video coding, especially in lower QPs (QP = 22 and QP = 27).

In addition, we compared the probability distribution between natural videos and screen contents for different transform unit (TU) sizes. As illustrated in Fig. 4, natural videos follow exponential distributions, but screen contents

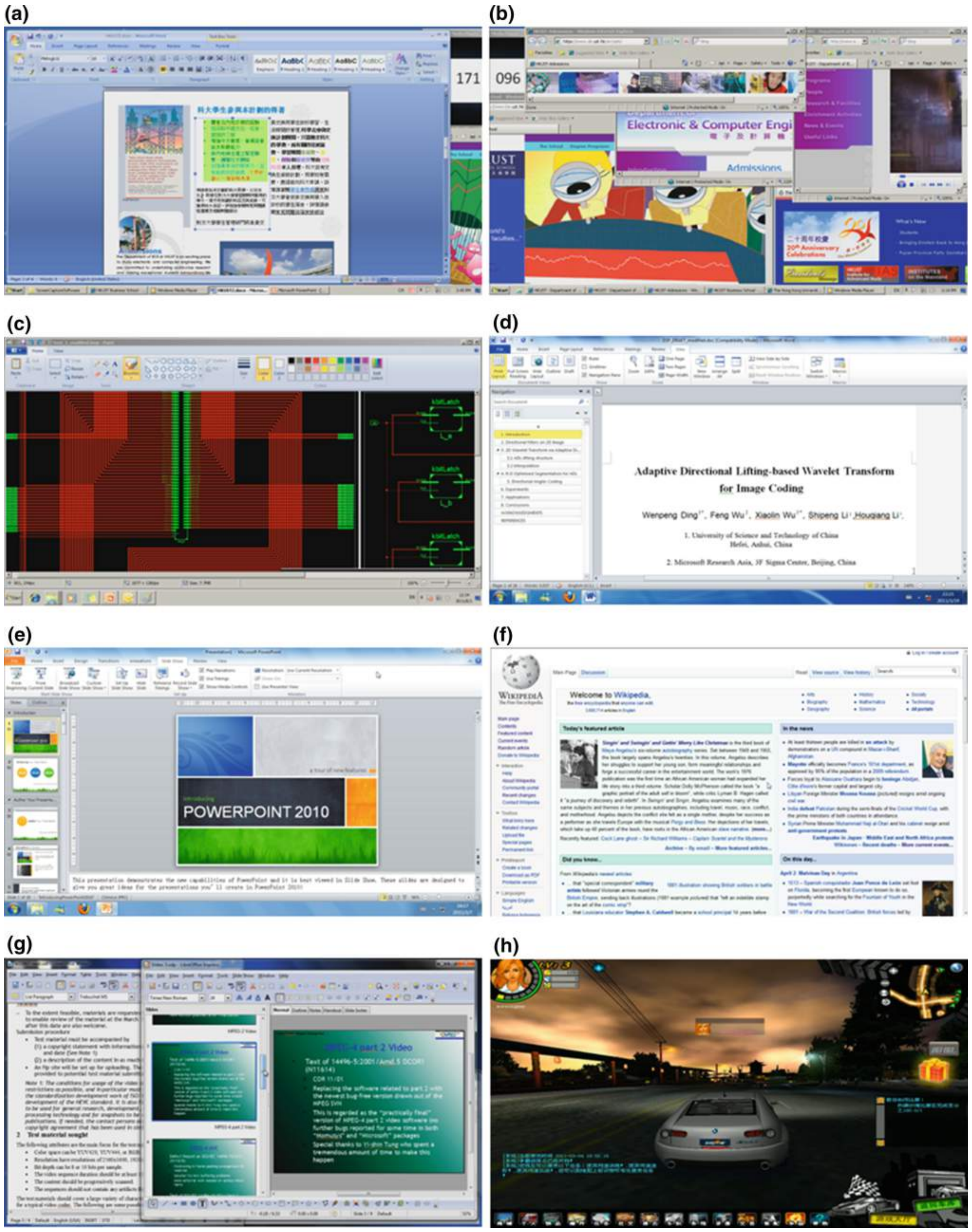
(a)
171
(b)
096
Electronic & Computer Engi
Admissions
(c)
(d)
Adaptive Directional Lifting-based Wavelet Transform for Image Coding
(e)
POWERPOINT 2010
(f)
Welcome to Wikipedia,
(g)
MPEG-4 part 2 Video
(h)

Fig. 7 Test sequences. **a** HKUST1. **b** HKUST2. **c** HKUST3. **d** BJUT-doc. **e** BJUT-slide. **f** BJUT-web. **g** SlideEditing. **h** ChinaSpeed

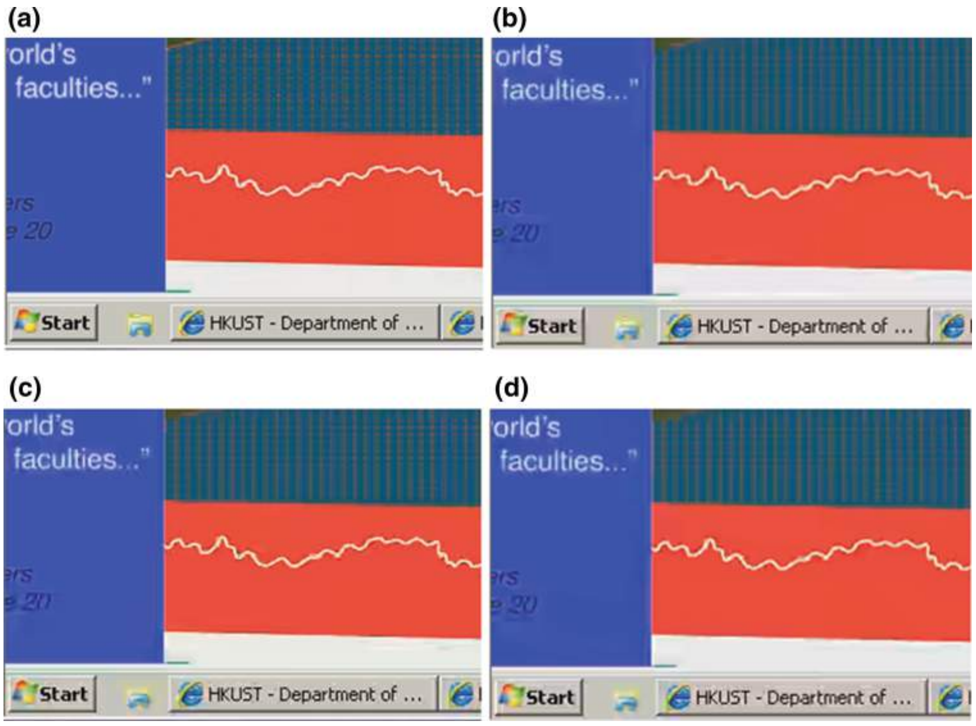

Fig. 8 Decoded image quality comparison. **a** Original image. **b** HM 8.0. **c** Proposed method. **d** Proposed method + transform skipping

do not, especially in smaller TUs (4×4 TU and 8×8 TU). From Figs. 3 and 4, we can confirm that screen contents do not show exponential probability distribution and contain much more high magnitude coefficient levels than natural videos.

It is known that the Golomb-Rice code is optimal for exponential probability distributions [12]. In other words, the Golomb-Rice code is less robust to signals with other statistical characteristics. Therefore, Golomb-Rice binarization in HEVC which is designed for natural image coding is not appropriate for screen content coding. Alternative entropy coding techniques are desirable to improve the performance of HEVC screen content coding.

3.2 Proposed level coding

The Golomb-Rice code assigns shorter codewords to low magnitude coefficient levels and longer codewords to high magnitude coefficient levels. For example, the level '16' corresponds to the binary codeword '11111111111111110' in the Golomb-Rice code, when the Rice parameter is 0. If these longer codewords frequently occurred, the coding performance will be reduced. Figure 5 illustrates the average probability distribution of the Fig. 3. As shown, the probability that high magnitude coefficient levels occurred in screen content coding is higher than that in natural video coding. Fortunately, since high magnitude levels are not dominant, it does not affect decoded image quality much even though we remove high magnitude levels.

Therefore, in the proposed method, we clip inefficient high magnitude coefficient levels to a predetermined threshold value. The clipping equation is shown in Eq. (4).

As N of Eq. (4) is set to 4 determined empirically through simulations, the clipping threshold 2^N is also determined. Here, $c(i,j)$ and $\hat{c}(i,j)$ represent original and clipped $coeff\_abs\_level\_remaining$ values, respectively.

$$\hat{c}(i,j) = \begin{cases} c(i,j) & \text{if } c(i,j) < 2^N \\ 2^N - 1 & \text{otherwise} \end{cases} \tag{4}$$

After clipping, levels of quantized transform coefficients are within the range $[0, 2^N - 1]$. In order to encode the finite range of levels efficiently, we use a limited length Golomb-Rice code, designed by *Starosolski* and *Skarbek* [12]. Their scheme shows better coding efficiency, when the set of symbols to be encoded is finite and its probability distribution is not distributed geometrically.

Given the Rice parameter $0 \le k < N$, a threshold π_k is defined by Eq. (5).

$$\pi_k = \min((l_{\max} - N) \cdot 2^k, 2^N - 2^k) \tag{5}$$

Here, $l_{\max}$ is the maximum codeword length. In the proposed method, we set $l_{\max}$ to 8 and the designed code contains codewords that do not exceed 8 bits. The limited length Golomb-Rice code only covers the Rice parameter k in the range of $[0, N-1]$. Thus, in case of $k = 4$, we use the conventional Golomb-Rice code. For $k < 4$, codewords are constructed by follows. If the symbol s is smaller than the threshold π_k, we use the conventional Golomb-Rice code to binarize s where $0 \le s < 2^N$. Otherwise, if $s \ge \pi_k$, the limited length Golomb-Rice code is used as follows.

Table 8 Coding performance of the proposed method for screen contents

Sequence	QP	HM 8.0		Proposed Method		BD-rate (%)
		Bitrate (kbps)	PSNR (dB)	Bitrate (kbps)	PSNR (dB)	
HKUST1	22	28,494.83	46.65	27,467.86	46.63	−0.75
	27	21,523.94	41.94	21,312.54	41.93	
	32	15,904.06	37.21	15,892.88	37.19	
	37	11,119.81	32.60	11,121.16	32.59	
HKUST2	22	29,598.16	45.21	29,116.92	45.22	−0.32
	27	20,691.12	40.79	20,619.63	40.80	
	32	14,108.66	36.50	14,134.94	36.51	
	37	9,383.46	32.35	9,391.08	32.34	
HKUST3	22	19,251.09	48.06	18,976.97	48.05	−0.21
	27	14,293.57	43.25	14,251.69	43.24	
	32	10,212.34	38.40	10,220.95	38.39	
	37	6,795.21	33.76	6,778.54	33.74	
BJUT-doc	22	6,723.05	49.18	6,421.81	49.19	−1.00
	27	5,179.52	44.52	5,136.62	44.59	
	32	3,957.00	39.75	3,961.14	39.79	
	37	2,899.38	34.99	2,908.35	34.99	
BJUT-slide	22	4,016.34	51.32	3,983.59	51.31	−0.32
	27	2,489.39	47.66	2,483.86	47.66	
	32	1,588.52	44.15	1,587.41	44.17	
	37	1,026.16	40.66	1,024.06	40.65	
BJUT-web	22	13,637.60	46.85	12,966.61	46.89	−1.13
	27	10,371.04	42.12	10,238.79	42.13	
	32	7,828.59	37.48	7,829.81	37.49	
	37	5,761.57	32.82	5,774.26	32.84	
SlideEditing	22	38,347.50	45.81	37,027.43	45.82	−0.63
	27	28,711.14	41.12	28,470.95	41.11	
	32	21,374.70	36.25	21,399.40	36.24	
	37	15,197.20	31.48	15,226.96	31.48	
ChinaSpeed	22	23,928.57	44.90	23,464.41	44.90	−0.46
	27	16,497.79	40.82	16,138.84	40.81	
	32	10,796.62	37.06	10,728.64	37.06	
	37	6,941.92	33.45	6,935.63	33.45	
Average						−0.60

$$\overbrace{1\cdots1}^{\frac{\pi_k}{2^k}}\overbrace{x_{M-1}\cdots x_0}^{M} \tag{6}$$

$$M = \left\lceil \log_2(2^N - \pi_k) \right\rceil \tag{7}$$

As shown in Eq. (6), the codeword consists of prefix and suffix parts. $\pi_k/2^k$ ‘1’s can be regarded as prefix, and the last M-bit suffix part is the binary representation of $s - \pi_k$. The length of the suffix part M is calculated by Eq. (7). Table 3 shows the codeword table.

In Table 3, suffix parts of codewords are the N-bit natural binary code. The natural binary code is complete, when the source alphabet size is an integer power of 2. From $k = 1$ to $k = 3$, the number of codewords for $s \geq \pi_k$ is equal to an integer power of 2. However, for k=0, the number of codewords for $s \geq \pi_k$ is not equal to integer power of 2. In this case, the adjusted binary code [13] is used. To integers in range [0, $j - 1$], where j represents the number of codewords for $s \geq \pi_k$, it assigns codewords that are sequences of $\lfloor \log_2 j \rfloor$ or $\lceil \log_2 j \rceil$ bits, as shown in Table 4. Note that when the j is equal to an integer power of 2, the adjusted binary code becomes the fixed length code. In Table 3, the number of codewords for $s \geq \pi_k$ in k $= 0$ is 12. Table 5 shows the limited length Golomb-Rice code using the adjusted binary

Table 9 Coding performance of the proposed method for natural videos

Sequence	Resolution (pixels)	Frame rate (fps)	BD-rate (%)
Traffic	2,560 × 1,600	30	−0.04
PeopleOnStreet	2,560 × 1,600	30	−0.17
Kimono	1,920 × 1,080	24	−0.34
ParkScene	1,920 × 1,080	24	+0.03
Cactus	1,920 × 1,080	50	+0.02
BasketballDrive	1,920 × 1,080	50	−0.02
BQTerrace	1,920 × 1,080	60	−0.02
BasketballDrill	832 × 480	50	−0.02
BQMall	832 × 480	60	−0.02
PartyScene	832 × 480	50	+0.07
RaceHorsesC	832 × 480	30	+0.04
BasketballPass	416 × 240	50	+0.06
BQSquare	416 × 240	60	−0.08
BlowingBubbles	416 × 240	50	+0.01
RaceHorses	416 × 240	30	+0.09
FourPeople	1,280 × 720	60	−0.02
Johnny	1,280 × 720	60	−0.11
KristenAndSara	1,280 × 720	60	−0.13
Average			−0.04

code that we used in the proposed method. The selecting method of k is same with the conventional HEVC standard.

It has been shown that transform skipping can significantly improve the coding efficiency for screen contents [14,15]. In order to check the probability distribution change caused by transform skipping, we compared transform skipping on/off cases. Figure 6 shows the probability distribution of having the transform skipping disabled versus enabled for different QP values. Since transform skipping in HEVC is only applied to 4 × 4 TUs in intra- and inter-coding, we do not check the probability distribution of having transform skipping disabled versus enabled for different TU sizes. When we use transform skipping, the probability distribution is changed, especially in lower QP values (QP = 22 and QP = 27). From Fig. 6, we can know that transform skipping results in increases in large coefficients. However, the fact that we noticed is only a small percentage of increases take place. Thus, the clipping process of the proposed method may not cause significant video quality degradation, while the amount of required bit-rate is reduced. That is, if we incorporate transform skipping to the proposed method, it can provide better coding efficiency.

4 Experimental results and analysis

In order to verify the performance of the proposed method, we implemented the proposed method in HM 8.0 [10]. In experiments, we used JCT-VC common test conditions [11]. Detail encoder parameters are summarized in Table 6.

Figure 8 describes the test sequences used in our experiments. The experiments are performed using eight screen contents suggested in HEVC standardization activities [16]. The detail information of test sequences is shown in Table 7. All sequences are YUV 4:2:0 format and 8 bits per pixel. Fig. 7a–g are computer screen pictures in editing slides and a slide presentation, respectively. Figure 7h is generated by computer graphic technology for a two-dimensional (2D) racing game.

Table 8 shows the experimental results. To evaluate the coding efficiency, the Bjøntegaard delta peak signal-to-noise ratio [17], which is recommended by video coding standard organizations, was used. As shown in Table 8, the average coding gain of the proposed method is 0.60 % in terms of BD-rate saving. In particular, for “*BJUT-web*” having a lot of texts, we found that the coding gain is 1.13 %, compared with the conventional HEVC standard.

The proposed method shows better coding performance in lower QPs. In lower QPs, high magnitude coefficient levels exist even after quantization and it results in undesirable increases in bit-rate. Using clipping, we can remove these levels in disregard of distortions. In addition, we tried to obtain additional bit-rate savings by the limited length Golomb-Rice code. Since the amount of bit-rate savings is bigger than occurred distortion, the proposed method could finally achieve the coding gain.

Table 10 Coding performance of the combination of the proposed method and transform skipping

Sequence	QP	Transform skipping		Proposed method + transform skipping		BD-rate(%)
		Bitrate (kbps)	PSNR (dB)	Bitrate (kbps)	PSNR (dB)	
HKUST1	22	23,789.97	47.33	21,825.14	47.38	−1.92
	27	17,588.65	42.69	17,076.23	42.67	
	32	12,807.60	38.05	12,874.74	38.02	
	37	9,363.49	33.83	9,368.15	33.79	
HKUST2	22	27,063.51	45.62	26,217.71	45.60	−0.49
	27	18,828.34	41.25	18,650.88	41.25	
	32	12,943.51	37.04	12,997.97	37.02	
	37	8,890.66	32.92	8,888.99	32.91	
HKUST3	22	15,904.25	49.21	15,379.18	49.21	−0.62
	27	11,995.36	44.45	11,969.22	44.46	
	32	9,152.10	39.70	9,125.51	39.69	
	37	6,554.33	34.62	6,553.56	34.63	
BJUT-doc	22	5,897.19	49.60	5,372.78	49.61	−2.27
	27	4,434.71	44.80	4,310.58	44.85	
	32	3,411.24	40.21	3,425.14	40.26	
	37	2,631.63	35.84	2,632.13	35.74	
BJUT-slide	22	3,865.75	51.51	3,815.93	51.52	−0.26
	27	2,417.23	47.91	2,410.78	47.91	
	32	1,572.24	44.42	1,574.82	44.43	
	37	1,026.87	40.83	1,025.80	40.84	
BJUT-web	22	12,112.97	47.86	10,857.19	47.86	−2.68
	27	9,061.50	42.74	8,721.99	42.74	
	32	6,786.76	38.14	6,824.73	38.16	
	37	5,082.82	33.38	5,086.28	33.33	
SlideEditing	22	34,612.02	46.27	32,134.06	46.28	−1.41
	27	25,489.12	41.86	24,992.72	41.86	
	32	19,193.40	37.30	19,276.97	37.29	
	37	14,336.43	32.58	14,342.78	32.59	
ChinaSpeed	22	22,023.64	45.09	21,171.58	45.09	−1.02
	27	14,798.24	41.19	14,543.66	41.20	
	32	9,721.96	37.36	9,765.05	37.37	
	37	6,426.73	33.67	6,428.50	33.67	
Average						−1.33

We also tested the proposed method for natural video sequences. Table 9 shows the experimental results for natural videos using the proposed method. The proposed method only provides an average BD-rate savings of 0.04 %. Moreover, for some sequences, slight BD-rate increases are observed. As shown in Fig. 5, natural videos have the exponential probability distribution. Thus, most coefficients are placed in the range of [0, 15] and a few coefficients are clipped. In addition, the limited length Golomb-Rice code that we used in the proposed method cannot guarantee remarkable coding efficiency improvement because HEVC already employs appropriate codewords for the exponential probability distribution, the probability distribution of natural videos. This is the reason that the proposed method achieves better coding gains in screen content coding than in natural video coding. Therefore, we can assure that the target application of the proposed method is screen content coding, not natural video coding.

In Sect. 3, we introduced the transform skipping algorithm that was adopted to HEVC. Then, from the probability distribution, we expected the proposed method is well incorporated with transform skipping. We checked the effect of transform skipping on the performance of the proposed method. Table 10 shows the experimental results. As shown

Table 11 Coding performance comparison with respect to the HM 8.0 anchor

Sequence	BD-rate (%)		
	Proposed method	Transform skipping	Proposed method + transform skipping
HKUST1	−0.75	−22.80	−24.02
HKUST2	−0.32	−12.25	−12.61
HKUST3	−0.21	−19.30	−19.66
BJUT-doc	−1.00	−15.57	−17.29
BJUT-slide	−0.32	−4.93	−5.17
BJUT-web	−1.13	−16.11	−17.89
SlideEditing	−0.63	−14.97	−16.00
ChinaSpeed	−0.46	−12.60	−13.46
Average	−0.60	−14.82	−15.76

in Table 10, when we combine transform skipping and the proposed method, we can achieve additional 1.33 % BD-rate saving, compared to transform skipping. Table 11 shows the coding efficiency of the proposed method, transform skipping, and the combination of the proposed method and transform skipping. From the experimental results, we can confirm that the proposed method does not collide with transform skipping. On average, 15.76 % BD-rate saving is obtained by using the proposed method and transform skipping together.

In Fig. 8, decoded images of the "*HKUST2*" test sequence at QP = 27 are shown. Some blurring artifacts are observed around the letters, but these are not significant. Therefore, we are convinced that the clipping operation does not introduce significant visual artifacts. In addition, we used video quality metric (VQM) [18] to measure the subjective quality of decoded images. VQM is closely correlated with characteristics of the human visual system (HVS). It allows more accurate measurement of the quality of the 2D video. We checked VQM using HM 8.0 as an anchor. The measured results are given in Table 12. In VQM, the smaller value means the better quality. Overall VQM results for four schemes in Table 12 are very similar. From the results, we can verify that the quality degradation caused by the clipping operation is not significant.

We additionally checked the performance of the proposed method on random access and low-delay B configurations. The experimental results are shown in Table 13. In other configurations, the coding efficiency of the proposed method is lower than that in the all-intra coding configuration. When we determine the threshold for clipping in this paper, we do not consider other configurations, but the all-intra coding configuration. That is, the threshold value used in this paper is optimized for the all-intra coding configuration. As shown in Fig. 9, the probability distribution of other configurations is

Table 12 Decoded image quality degradation measurements using video quality metric (VQM)

Sequence	QP	HM 8.0	Transform skipping	Proposed method	Proposed method + transform skipping
HKUST1	22	0.239	0.258	0.243	0.245
	27	0.409	0.409	0.426	0.414
	32	0.690	0.714	0.683	0.715
	37	1.136	1.116	1.142	1.105
HKUST2	22	0.343	0.331	0.343	0.333
	27	0.557	0.565	0.563	0.571
	32	0.877	0.883	0.873	0.882
	37	1.381	1.349	1.381	1.349
HKUST3	22	0.269	0.247	0.271	0.253
	27	0.427	0.435	0.466	0.451
	32	0.876	0.755	0.902	0.766
	37	1.553	1.432	1.549	1.427
BJUT-doc	22	0.155	0.16	0.153	0.161
	27	0.257	0.293	0.259	0.274
	32	0.444	0.45	0.441	0.452
	37	0.767	0.773	0.776	0.782
BJUT-slide	22	0.183	0.186	0.181	1.957
	27	0.271	0.282	0.266	0.275
	32	0.384	0.386	0.386	0.386
	37	0.563	0.557	0.570	0.554
BJUT-web	22	0.227	0.199	0.221	0.203
	27	0.404	0.379	0.392	0.380
	32	0.694	0.653	0.681	0.639
	37	1.145	1.122	1.150	1.121
SlideEditing	22	0.316	0.325	0.315	0.324
	27	0.523	0.533	0.523	0.533
	32	0.896	0.867	0.889	0.868
	37	1.556	1.441	1.552	1.435
ChinaSpeed	22	0.787	0.788	0.787	0.790
	27	1.131	1.111	1.133	1.119
	32	1.652	1.617	1.651	1.610
	37	2.322	2.281	2.335	2.283
Average		0.732	0.716	0.734	0.771

quite different from that of the all-intra coding configuration. Thus, for other configurations, specifically for inter-coded frames, a new threshold value is required and we leave it as a future work.

5 Conclusions

In this paper, we have proposed the efficient level coding technique for screen content coding. We clip inefficient high

Table 13 Coding performance of the proposed method for other configurations

Sequence	Random access	Low-delay B
HKUST1	−1.12	−0.27
HKUST2	−0.20	+0.40
HKUST3	−0.18	+0.26
BJUT-doc	−0.62	+0.89
BJUT-slide	−0.30	−1.27
BJUT-web	+1.02	−0.81
SlideEditing	−0.26	−0.82
ChinaSpeed	−0.19	−0.08
Average	−0.23	−0.21

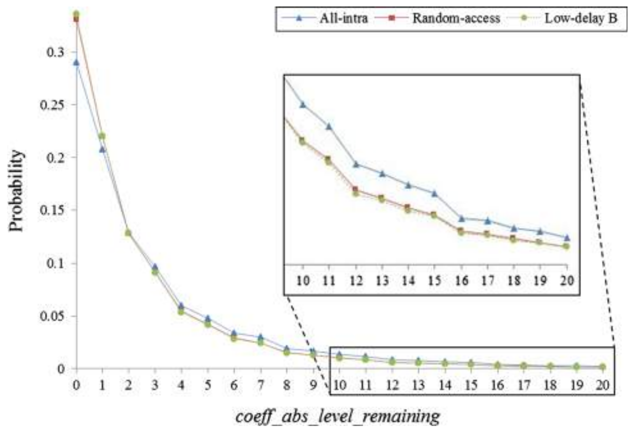

Fig. 9 The probability distribution comparison on various coding configurations

magnitude levels of quantized transform coefficients causing overhead bits. Then, we only encode levels that are smaller than the threshold, and we binarize the clipped levels using the limited length Golomb-Rice code. Simulation results demonstrate that the proposed method improves the compression performance of HEVC for a variety of screen content videos.

Acknowledgments This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. 2012-0009228).

References

1. ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11: High Efficiency Video Coding (HEVC) Text Specification Working Draft 5, JCTVC-G1103, Geneva, CH (Nov. 2011)
2. ISO/IEC JTC1/SC29/WG11: Vision, Application, and Requirements for High Performance Video Coding (HVC), MPEG Document, N11096, Kyoto, JP (Jan. 2010)
3. Marpe, D., Schwarz, H., Wiegand, T.: Context-based adaptive binary arithmetic coding in the H.264/AVC. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 598–603 (July 2003)
4. Sullivan, G., Topiwala, P., Luthra, A.: The H.264/AVC advanced video coding standard: overview and introduction to the fidelity range extensions. In: Proceedings of the SPIE Conference, Special Session on Advances in the New Emerging Standard: H.264/AVC, pp. 454–474 (2004)
5. ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11: Description of Video Coding Technology Proposal by Tandberg, Nokia, Ericsson. JCTVC-A119, Dresden, DE (April 2010)
6. ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6: Context-Adaptive VLC (CVLC) Coding of Coefficients. JVT Document, C028, Seattle, WA (May 2002)
7. Wiegand, T., Sullivan, G., Bjøntegaard, G., Luthra, A.: Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 560–576 (2003)
8. Sze, V., Budagavi, M.: High throughput CABAC entropy coding in HEVC. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1778–1791 (2012)
9. Nguyen, T., Marpe, D., Schwarz, H., Wiegand, T.: Reduced-complexity entropy coding of transform coefficient levels using truncated Golomb-Rice codes in video compression. In: Proceedings of the IEEE International Conference on Image Processing. Brussels, BE (Sept. 2011)
10. HM 8.0 Software: http://hevc.kw.bbc.co.uk/trac/browser/tags/HM-8.0
11. ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11: HM 8 Common Test Conditions and Software Reference Configurations, JCTVC-J1100, 10th JCT-VC Meeting, Stockholm, SE (July 2012)
12. Gallager, R., Voorhis, D.: Optimal source codes for geometrically distributed integer alphabets. IEEE Trans. Inf. Theory **21**(2), 228–230 (1975)
13. Starosolski, R., Skarbek, W.: Modified Golomb-Rice codes for lossless compression of medical images. In: Proceedings of the International Conference on E-health in Common Europe, Cracow, PL (2003)
14. ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11: Intra Transform Skipping, JCTVC-I0408, 9th JCT-VC Meeting, Geneva, CH (May 2012)
15. ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11: Inter Transform Skipping, JCTVC-J0237, 10th JCT-VC Meeting, Stockholm, SE (July 2012)
16. ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11: JCT-VC AHG Report: Screen Content Coding, JCTVC-F013, Torino, IT (July 2011)
17. ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11: On BD-Rate Calculation, JCTVC-F270, Torino, IT (July 2011)
18. Watson, A., Hu, J., McGowan, J.: DVQ: a digital video quality metric based on human vision. J. Electron. Imaging **10**(1), 20–29 (2001)