Circuits Syst Signal Process (2012) 31:813–825
DOI 10.1007/s00034-011-9338-1

SHORT PAPER

Efficient Differential Pixel Value Coding in CABAC for H.264/AVC Lossless Video Compression

Jin Heo · Yo-Sung Ho

Received: 18 September 2010 / Revised: 28 June 2011 / Published online: 27 July 2011

Abstract Since context-based adaptive binary arithmetic coding (CABAC) as the entropy coding method in H.264/AVC was originally designed for lossy video compression, it is inappropriate for lossless video compression. Based on the fact that there are statistical differences of residual data between lossy and lossless video compression, we propose an efficient differential pixel value coding method in CABAC for H.264/AVC lossless video compression. Considering the observed statistical properties of the differential pixel value in lossless coding, we modified the CABAC encoding mechanism with the newly designed binarization table and the context-modeling method. Experimental results show that the proposed method achieves an approximately 12% bit saving, compared to the original CABAC method in the H.264/AVC standard.

Keywords H.264/AVC · Context-based adaptive binary arithmetic coding (CABAC) · Lossless video compression

1 Introduction

The latest international video coding standard, H.264/AVC, was developed by the Joint Video Team (JVT) of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG). For higher compression efficiency, H.264/AVC has adopted several powerful coding tools by mainly focusing on lossy video compression [12, 16].

J. Heo (✉) · Y.-S. Ho
School of Information and Communications, Gwangju Institute of Science and Technology (GIST), 261 Cheomdan-gwagiro (Oryong-dong), Buk-gu, Gwangju 500-712, Republic of Korea
e-mail: jinheo@gist.ac.kr

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

To date, since the H.264/AVC standard has been developed by mainly focusing on lossy coding, it does not provide good coding performance for lossless coding. Thus, in order to provide improved functionality for lossless coding, JVT developed a *pulse-code modulation* (PCM) macroblock coding mode and then a transform-bypass lossless mode in the fidelity range extensions (FRExt) [13].

Recently, instead of developing a block-based intra prediction, new intra prediction methods called sample-wise *differential pulse-code modulation* (DPCM) [6, 9] were introduced for lossless intra prediction, and they have been shown to provide better coding performance. As a result, one contribution [6] was adopted as a part of the new draft amendment for the H.264/AVC standard.

In lossy coding, residual data are quantized transform coefficients [7], and they are encoded by the entropy coder. However, in lossless coding, residual data do not represent quantized transform coefficients, but rather differential pixel values between the original and predicted pixel values, because neither transform nor quantization is used. In other words, prediction residuals are directly coded by the entropy coder. Therefore, there are significant statistical differences of residual data between lossy and lossless coding. This means that the current residual data coding method in *context-based adaptive binary arithmetic coding* (CABAC) [8] originally designed for lossy coding is inappropriate for lossless coding.

In our previous work, we introduced improved CAVLC for lossless intra coding [3, 4] based on the conventional CAVLC [10] in H.264/AVC. Although CAVLC in available for all profiles in H.264/AVC, the main scope of CAVLC is the baseline profile and the application areas of which include video telephony, video conferencing, and wireless communications. Thus, the use of the proposed CAVLC was limited in a small range of applications.

In this paper, we proposed a new differential pixel value coding method in CABAC for lossless intra coding by modifying the semantics and the decoding process without requiring any other syntax elements in the H.264/AVC standard. In order to efficiently encode differential pixel values, we modified the binarization table and context-modeling method based on their statistical characteristics. Experimental results show that the proposed method improves coding performance, compared to the conventional CABAC in H.264/AVC.

The rest of this paper is organized as follows. In Sect. 2, we briefly present an overview of CABAC encoder structure and then explain the structure of CABAC for residual data coding. In Sect. 3, the proposed differential pixel value coding method is explained. In Sect. 4, coding performance of the proposed method is compared to that of the conventional CABAC and well-known lossless coding method. Finally, conclusion is presented in Sect. 5.

2 Overview of CABAC in H.264/AVC

The encoding process of CABAC consists of four coding steps: binarization, context modeling, binary arithmetic coding, and probability update. In the first step, a given nonbinary valued syntax element is uniquely mapped to a binary sequence (*bin string*). When the binary valued syntax element is given, the first step is bypassed. In

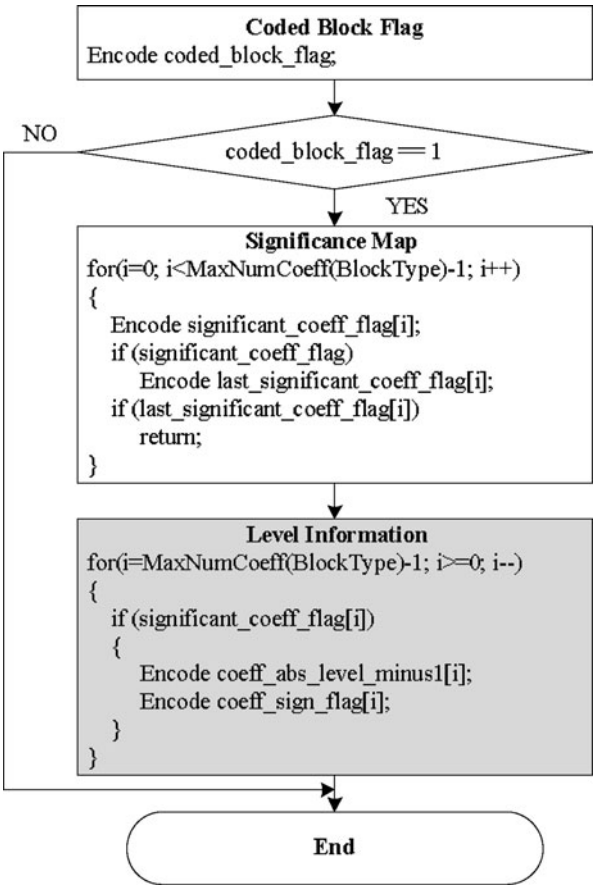

Fig. 1 CABAC encoding structure for residual data coding

the regular coding mode, each binary value (*bin*) of the binary sequence enters the context modeling stage, where a probability model is selected based on the previously encoded syntax elements. Then, the arithmetic coding engine encodes each binary value with its associated probability model. Finally, the selected context model is updated according to the actual coded binary value. Alternatively, in the bypass coding mode, each binary value is directly encoded via the bypass coding engine without using an explicitly assigned model.

Figure 1 illustrates the CABAC encoding structure for a 4×4 subblock of quantized transform coefficients. First, for each subblock, a 1-bit symbol *coded_block_flag* is transmitted to indicate that a subblock has significant coefficients. If *coded_block_flag* is zero, no further information is transmitted for the current subblock; otherwise, the significance map and level information coding processes are sequentially encoded.

If *coded_block_flag* indicates that a subblock has significant coefficients, a binary-valued significance map is encoded. For each coefficient, a 1-bit syntax element *significant_coeff_flag* is encoded in scanning order. If *significant_coeff_flag* is one, i.e., if a nonzero coefficient exists at this scanning position, a further 1-bit syntax element

last_significant_coeff_flag is encoded. This syntax element states whether the current significant coefficient is the last coefficient inside the subblock or not.

After the encoded significance map determines the locations of all significant coefficients inside a subblock, the values of the significant coefficients are encoded by using two syntax elements: *coeff_abs_level_minus1* and *coeff_sign_flag* in reverse scanning order. The syntax element *coeff_sign_flag* is encoded by a 1-bit symbol, whereas the *Unary/0th-order Exponential Golomb* (UEG0) binarization method is used to encode the values of *coeff_abs_level_minus1* representing the absolute value of the level minus 1.

3 Proposed Differential Pixel Value Coding Method

In this section, we describe a new differential pixel value coding method for lossless intra coding by reflecting the statistical properties of residual sample values. In Fig. 1, the gray shaded processes are modified in the proposed method.

3.1 Binarization for Differential Pixel Value Coding

For the absolute value of the quantized transform coefficient (*abs_level*) in lossy coding, the *Unary/kth-order Exp-Golomb* (UEGk) binarization method is applied. The design of the UEGk binarization method is motivated by the fact that the unary code is the simplest prefix-free code to implement, and it permits the fast adaptation of individual symbol probabilities in the subsequent context modeling stage. These observations are only accurate for small *abs_level*s; however, for larger *abs_level*s, adaptive modeling has limited functionality. Therefore, these observations lead to the concept of concatenating an adapted truncated unary (TU) code as a prefix and a static Exp-Golomb code [14] as a suffix.

UEGk binarization of *abs_level* has a cutoff value $S = 14$ for the TU prefix and the order $k = 0$ for the Exp-Golomb (EG0) suffix. Previously, Golomb codes have been proven to be optimal prefix-free codes for geometrically distributed sources [2]. Moreover, EG0 used in lossy coding is the optimal code for a probability density function (pdf) as follows:

$$p(x) = 1/2 \cdot (x+1)^{-2} \quad \text{for } x \geq 0 \tag{1}$$

The statistical properties of the absolute value of the differential pixel value (*abs_diff_pixel*) in lossless coding are quite different from those of *abs_level* in lossy coding. In lossy coding, the statistical distribution of *abs_level* is highly skewed on small level values.

However, in lossless coding, the statistical distribution of *abs_diff_pixel* is quite wide; note the large variation and wide tails, as shown in Fig. 2. Moreover, from Fig. 2, we can also observe that the TU code is a fairly good model for the statistical distribution of *abs_level* in lossy coding, whereas, it is not appropriate for the statistical distribution of *abs_diff_pixel* in lossless coding. Therefore, as UEG0 binarization was originally designed for lossy coding, it is not appropriate for lossless coding.

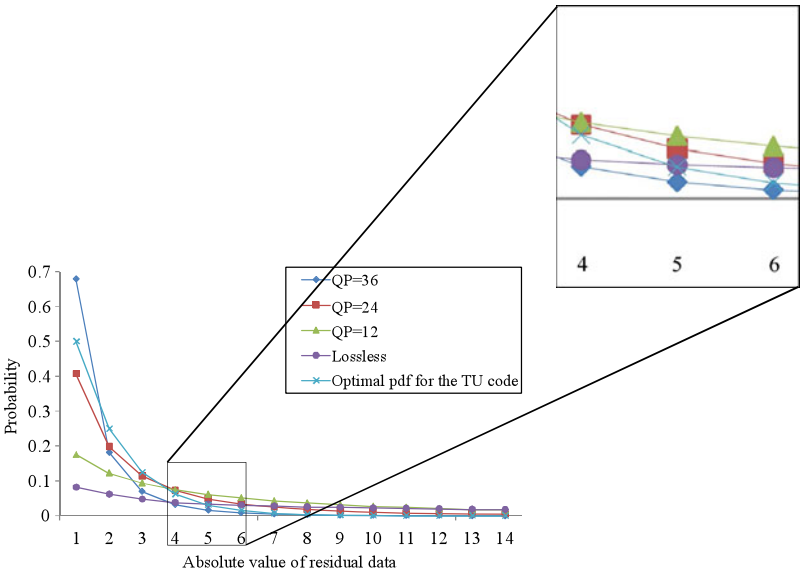

Fig. 2 Probability distribution of the absolute value of residual data and the optimal pdf of the TU code

In order to efficiently encode *abs_diff_pixel* in lossless coding, we adjusted the cutoff value S of the TU prefix in UEG0 binarization. In Fig. 2, the optimal pdf curve for the TU code and the statistical distribution curve for *abs_diff_pixel* in lossless coding intersect at an absolute value of 5. Moreover, as the absolute value increases, the statistical difference between the TU code and *abs_diff_pixel* in lossless coding becomes larger. Therefore, we determined a new cutoff value $S = 5$ for the TU prefix in the proposed binarization method.

In order to provide a good prefix-free code for lossless coding, we also determined an appropriate parameter k for the EGk code. The prefix of the EGk codeword consists of a unary code corresponding to the value $l(x) = \lfloor \log_2(x/2^k + 1) \rfloor$. The suffix is then computed as the binary representation of $x + 2^k(1 - 2^{l(x)})$ using $k + l(x)$ significant bits. Consequently, for EGk binarization, the number of symbols having the same code length of $k + 2l(x) + 1$ grows geometrically. Then, by inverting Shannon's relationship between the ideal code length and the symbol probability, we can find each pdf corresponding to an EGk having an optimal code according to a parameter k:

$$p_k(x) = 1/2^{k+1} \cdot \left(x/2^k + 1\right)^{-2} \quad \text{for } x \geq 0 \tag{2}$$

where $p_k(x)$ is the optimal pdf corresponding to the EGk code for a parameter k. This implies that for an appropriately chosen parameter k, the EGk code represents a fairly good prefix-free code for tails of typically observed pdfs.

Figure 3 presents the probability distribution of $p_k(x)$ for $k = 0$, 1, 2, and 3 and the probability distribution of *abs_diff_pixel*s from 6 to 20, where *abs_diff_pixel*s up to 5 are specified by the TU code. In the figure, the probability distribution of $p_k(x)$ for $k = 3$ is well matched to the probability distribution of *abs_diff_pixel*. This

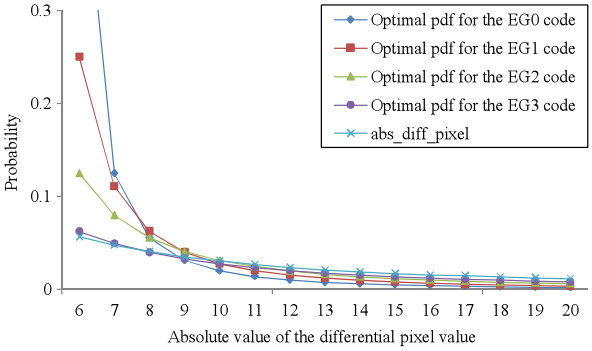

Fig. 3 Probability distribution of the optimal pdf corresponding to the EGk code for $k = 0, 1, 2,$ and 3 and the probability distribution of the absolute value of the differential pixel value

Table 1 Proposed UEG3 binarization for encoding the absolute value of the differential pixel value

abs_diff_pixel	Bin string											
	TU prefix					EG3 suffix						
1	0											
2	1	0										
3	1	1	0									
4	1	1	1	0								
5	1	1	1	1	0							
6	1	1	1	1	1	0	0	0	0			
7	1	1	1	1	1	0	0	0	1			
8	1	1	1	1	1	0	0	1	0			
9	1	1	1	1	1	0	0	1	1			
10	1	1	1	1	1	0	1	0	0			
11	1	1	1	1	1	0	1	0	1			
12	1	1	1	1	1	0	1	1	0			
13	1	1	1	1	1	0	1	1	1			
14	1	1	1	1	1	1	0	0	0	0	0	
15	1	1	1	1	1	1	0	0	0	0	1	
16	1	1	1	1	1	1	0	0	0	1	0	
17	1	1	1	1	1	1	0	0	0	1	1	
...	...					...						
bin	1	2	3	4	5	6	7	8	9	10	11	...

result implies that the EG3 code represents a fairly good approximation of the ideal prefix-free code for encoding *abs_diff_pixel* in lossless coding.

Based on these observations, we designed an efficient binarization method to encode *abs_diff_pixel* in lossless coding. In the proposed algorithm, UEGk binarization of *abs_diff_pixel* is specified by the cutoff value $S = 5$ for the TU prefix and the order $k = 3$ for the EGk suffix. Table 1 shows the proposed UEG3 binarization for *abs_diff_pixel*.

3.2 Context Modeling for Differential Pixel Value Coding

Each model can be identified by the unique context index, as the entirety of probability models used in CABAC can be arranged in a linear fashion. The context in-

Table 2 Block types with the number of coefficients and associated context category (*ctx_cat*)

Block type	MaxNumCoeff	Context category
Luma DC block for Intra16 × 16	16	0
Luma AC block for Intra16 × 16	15	1
Luma block for Intra4 × 4	16	2
Luma block for Inter		
U- and V-Chroma DC block	4	3
U- and V-Chroma AC block	15	4
Luma block for 8 × 8	64	5

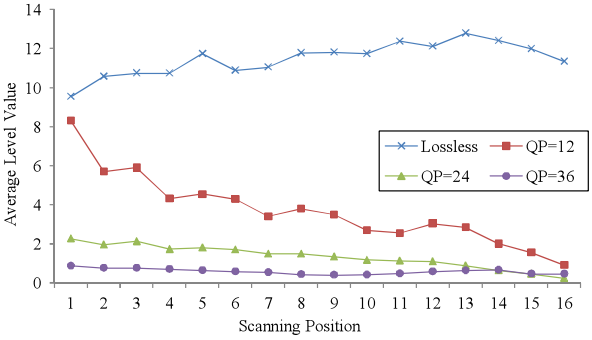

Fig. 4 Distribution of average absolute level value according to the scanning position

dex for all syntax elements of residual data (γ_S), except for the syntax element of *coded_block_pattern*, is specified by

$$\gamma_S = \Gamma_S + \Delta_S(ctx\_cat) + \chi_S \tag{3}$$

where Γ_S represents the *context index offset*, which is defined as the lower value of the range of a given syntax element S, and χ_S denotes the *context index increment* of a given syntax element S. In addition, $\Delta_S(ctx\_cat)$ is the *context category* (*ctx_cat*) dependent offset Δ_S and is determined based on the block type, as shown in Table 2.

In lossless coding, since neither transform nor quantization is performed, the differential pixel values within a subblock do not separate dc and ac coefficients. As a result, only block type corresponding to a context category of 2 is used in lossless intra coding. Therefore, we can fix the context category value in the context modeling stage for the absolute value of the differential pixel minus 1 (*abs_diff_pixel_minus1*).

In order to determine the context index increment for *coeff_abs_level_minus1* (χ_{Coeff}) in lossy coding, we used two adequately designed sets of context models; one for the first bin (bin index 1, $\chi_{\text{Coeff}}(i, bin\_index = 1)$) and the other for the bins (bin indices 2 to 14, $\chi_{\text{Coeff}}(i, bin\_index)$) remaining in the UEG0 prefix. In lossy coding, at the end of the scanning position, *abs_level* is likely to observe the occurrence of successive ±1, called trailing ones. In addition, *abs_level* is going to be larger as the scanning position decreases, as shown in Fig. 4. Based on these observations, χ_{Coeff} is determined according to the accumulated number of encoded trailing ones

Table 3 Encoding parameters

Parameter	Setting
ProfileIDC	244 (High 4:4:4)
IntraPeriod	1 (only intra coding)
QPISlice	0 (lossless)
SymbolMode	1 (CABAC)
QPPrimeYZeroTransformBypassFlag	1 (lossless)

(*NumT1*(*i*)) and the accumulated number of encoded levels with an absolute value greater than one (*NumLgt1*(*i*)), where i is the scanning position.

However, the figure shows that *abs_diff_pixel* in lossless coding is independent of the scanning position. Therefore, we designed an adaptive context modeling method for *abs_diff_pixel_minus1* in lossless coding.

In lossy coding, selection of the context model is based on the expectation that *abs_level* is likely to increase at low frequencies. Thus, $\chi_{\mathrm{Coeff}}(i, bin\_index)$ increases based on the assumption that the next *abs_level* to be coded is going to be larger. However, since the next *abs_diff_pixel* does not necessarily increase at lower frequencies in lossless coding, we cannot assume that the next *abs_diff_pixel* is larger than the current *abs_diff_pixel*. Therefore, in lossless coding, the context index increment for the remaining bins (bin indices 2 to 5, $\chi_{\mathrm{DiffPix}}(i, bin\_index)$) applied the new cutoff value $S = 5$ is selected by considering the previously encoded *abs_diff_pixel*, due to the fact that we cannot predict whether the next *abs_diff_pixel* will increase or not.

In order to efficiently encode the remaining bins with $2 \leq$ bin index ≤ 5 in lossless coding, as shown in the gray shaded columns in Table 1, the corresponding context index ($\gamma_{\mathrm{DiffPix}}(i, bin\_index)$) is determined by

$$\gamma_{\mathrm{DiffPix}}(i, bin\_index) = \Gamma_{\mathrm{DiffPix}} + \Delta_{\mathrm{DiffPix}}(ctx\_cat = 2) + \chi_{\mathrm{DiffPix}}(i, bin\_index) \quad (2 \leq bin\_index \leq 5) \tag{4}$$

$$\chi_{\mathrm{DiffPix}}(i, bin\_index) = 5 + ctx$$
$$ctx = \begin{cases} \min(4, NumLgt1(i-1)+1) & \text{if } AbsDiff \geq PrevAbsDiff \\ \min(4, NumLgt1(i-1)) & \text{otherwise} \end{cases} \tag{5}$$

where *AbsDiff* and *PrevAbsDiff* represent the absolute values of the current differential pixel value and the previous differential pixel value, respectively.

4 Experimental Results

The proposed algorithm was implemented on JM13.2 [5]. We performed experiments on several test sequences of YUV 4:2:0 and 8 bits per pixel (bpp) with QCIF, CIF, and HD resolutions. Table 3 shows the encoding parameters for the reference software.

Table 4 Comparison of bit savings for the conventional CABAC and the proposed methods with QCIF (176 × 144) resolution sequences

Sequence	Image size (bits)	Method	Encoding bits (bits)	Stuffing Bytes (bits)	Occurrence frequency of stuffing bytes	Saving bits (%)
Foreman	91238400	CABAC	42062024	4217672	300	0
		Method I	38599072	1145688	300	8.2330
		Method II	38580912	1082824	300	**8.2761**
News	91238400	CABAC	41683584	4957960	300	0
		Method I	36464336	906120	300	12.5211
		Method II	36420544	821776	300	**12.6262**
Container	91238400	CABAC	43565952	6384336	300	0
		Method I	38447512	2226112	300	11.7487
		Method II	38439072	2186368	300	**11.7681**
Stefan	91238400	CABAC	67131824	19222240	300	0
		Method I	53908696	9101360	300	19.6973
		Method II	53888896	9051488	300	**19.7268**
Salesman	91238400	CABAC	48875168	7833048	300	0
		Method I	44032176	3374928	300	9.9089
		Method II	44015016	3301656	300	**9.9440**
Average		CABAC				0
		Method I				12.4218
		Method II				**12.4682**

For applications such as content-contribution, content-distribution, and studio editing, JVT developed extensions to the original standard—known as the fidelity range extensions (FREext) [13]. The FREext supports a suite of four new profiles collectively called the *High* profiles: High profile, High 10 profile, High 4:2:2 profile, and High 4:4:4 profile. The High 4:4:4 profile supports up to 4:4:4 chroma sampling format, up to 12 bpp, efficient lossless coding, and integer residual color transform for coding RGB video sequence. Therefore, although we used the test sequences of YUV 4:2:0 and 8 bpp for experiments, we selected the High 4:4:4 profile for lossless coding.

In order to evaluate coding performance of each proposed method, we experimented with two parts, based on the following settings:

(1) *Method I*: use modified binarization table.
(2) *Method II*: *Method I* + use new context modeling method.

In order to verify coding efficiency, we performed two kinds of experiments. In the first experiment, we have compared coding performance of the proposed methods to that of the conventional CABAC, as shown in Tables 4, 5, and 6. In the experiment, we

Table 5 Comparison of bit savings for the conventional CABAC and the proposed methods with CIF (352 × 288) resolution sequences

Sequence	Image size (bits)	Method	Encoding bits (bits)	Stuffing Bytes (bits)	Occurrence frequency of stuffing bytes	Saving bits (%)
Flowergarden	182476800	CABAC	136699464	41942256	150	0
		Method I	108328768	20586112	150	20.7541
		Method II	108297232	20507576	150	**20.7771**
Mobile	182476800	CABAC	139668256	38233120	150	0
		Method I	113776160	18320456	150	18.5383
		Method II	113723104	18142776	150	**18.5763**
Tempete	182476800	CABAC	117918224	28731408	150	0
		Method I	100282024	14026776	150	14.9563
		Method II	100240728	13908752	150	**14.9913**
Paris	182476800	CABAC	97911392	15504928	150	0
		Method I	85348872	5691776	150	12.8305
		Method II	85312184	5525176	150	**12.8680**
Football	182476800	CABAC	121390696	29013992	150	0
		Method I	107143952	16146032	150	11.7363
		Method II	107125568	16010928	150	**11.7514**
Average		CABAC				0
		Method I				15.7631
		Method II				**15.7928**

encoded 300 frames, 150 frames, and 100 frames in QCIF, CIF, and HD sequences, respectively. In the second experiment, we have compared coding performance of the proposed method (*Method II*) to that of lossless joint photographic experts group (JPEG-LS) [11, 15] with only one frame (first frame) encoding, as shown in Table 7.

Comparisons were made in terms of bit-rate percentage differences in Tables 4, 5, and 6 and compression ratio differences in Table 7 with respect to H.264/AVC CABAC and JPEG-LS, respectively. These changes were calculated as follows:

$$\text{Saving Bits}(\%) = \frac{\text{Bitrate}_{\text{H.264/AVC}} - \text{Bitrate}_{\text{Method}}}{\text{Bitrate}_{\text{H.264/AVC}}} \times 100 \tag{6}$$

$$\text{Compression Ratio} = \frac{\text{Image size}}{\text{Bitrate}_{\text{Method}}} \tag{7}$$

Tables 4, 5, and 6 show that the proposed method provides an approximately 12%, 15%, and 8% bit savings with QCIF, CIF, and HD resolution sequences, compared to the conventional CABAC in H.264/AVC. In Table 7, we show that the proposed method provides better coding performance than JPEG-LS in lossless coding.

Table 6 Comparison of bit savings for the conventional CABAC and the proposed methods with HD (1280 × 720 and 1920 × 1080) resolution sequences

Sequence	Image size (bits)	Method	Encoding bits (bits)	Stuffing Bytes (bits)	Occurrence frequency of stuffing bytes	Saving bits (%)
City_corr (1280 × 720)	1105920000	CABAC	565080864	91459600	100	0
		Method I	521989336	48017288	100	7.6257
		Method II	521736264	47234600	100	**7.6705**
Night (1280 × 720)	1105920000	CABAC	455951136	40631984	100	0
		Method I	424775144	11798360	100	6.8376
		Method II	424531640	11100448	100	**6.8910**
Crowdrun (1920 × 1080)	2488320000	CABAC	1250235376	192383640	100	0
		Method I	1152830408	101425120	100	7.7909
		Method II	1152640992	100236808	100	**7.8061**
Parkjoy (1920 × 1080)	2488320000	CABAC	1283550664	227556528	100	0
		Method I	1163176784	119618112	100	9.3782
		Method II	1162938240	118485224	100	**9.3968**
Average		CABAC				0
		Method I				7.9081
		Method II				**7.9411**

There is an important issue, the so-called stuffing mechanism in H.264/AVC when we are tinkering with CABAC for lossless coding. This mechanism was incorporated into the standard to guarantee a certain upper limit on the bin-to-bit ratio per picture with the intention to limit the worst-case bin processing rate in CABAC. To fulfill this requirement, the standard encoder must insert a sufficient number of *cabac_zero_word*s, a byte-aligned sequence of two bytes equal to $0x0000$, into the last part of each slice whenever the bin-to-bit ratio exceeds the given limit. More precisely, each *cabac_zero_word* is represented by a 3-byte sequence $0x000003$ to avoid an emulation of the start code ($0x000001$).

The stuffing mechanism occurs frequently in lossless or near-lossless coding (quantization parameter (QP) < 10). In Tables 4, 5, and 6, we can observe that although the stuffing bytes occur in all frames of all test sequences, the proposed method reduces the bit rate used for generating the stuffing bytes. As a result, the proposed method enhances coding efficiency. Since the stuffing bytes are used to guarantee the upper limit on the bin-to-bit ratio per picture which is very important for applications, bit saving calculation method excluding the stuffing bytes is not suitable. Therefore, we give bit-saving results by taking stuffing bytes into consideration. Note that more details pertaining to this stuffing mechanism are explained in subclause 7.4.2.10 of the H.264/AVC recommendation [1].

Table 7 Comparison of compression ratio for JPEG-LS and the proposed method

Resolution	Sequence	Method	Compression ratio
QCIF	Foreman	JPEG-LS	1.81785
		Method II	**2.36486**
	News	JPEG-LS	2.08716
		Method II	**2.50514**
	Container	JPEG-LS	1.90297
		Method II	**2.37358**
	Stefan	JPEG-LS	1.55746
		Method II	**1.69308**
	Salesman	JPEG-LS	1.68685
		Method II	**2.07289**
CIF	Flowergarden	JPEG-LS	1.62005
		Method II	**1.68496**
	Mobile	JPEG-LS	1.48648
		Method II	**1.60457**
	Tempete	JPEG-LS	1.58259
		Method II	**1.82039**
	Paris	JPEG-LS	1.72034
		Method II	**2.13893**
	Football	JPEG-LS	1.60613
		Method II	**1.70339**
HD	City_corr	JPEG-LS	1.90791
		Method II	**2.11969**
	Night	JPEG-LS	2.25829
		Method II	**2.60504**
	Crowdrun	JPEG-LS	1.68024
		Method II	**2.15880**
	Parkjoy	JPEG-LS	1.86636
		Method II	**2.13968**
Average		JPEG-LS	1.77005
		Method II	**2.07036**

5 Conclusions

In this paper, we proposed an efficient differential pixel value coding method in CABAC for H.264/AVC lossless video compression. Considering the observed statistical properties of the differential pixel values in lossless coding, we changed the binarization table based on the *Unary/kth-order Exp-Golomb* (UEGk) binarization and then modified the context-modeling method. Experimental results show that the proposed method provides an approximately 12% bit saving, compared to the conventional CABAC in the H.264/AVC standard.

Acknowledgements This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2011-(C1090-1111-0003))

References

1. Editors' draft revision to ITU-T Rec. H.264|ISO/IEC 14496-10 Advanced Video Coding—in preparation for ITU-T SG17 AAP Consent, Document JVT-AD205.doc, Joint Video Team of ISO/IEC 14496-10 AVC, ISO/IEC JTC1/SC29/WG11 and ITU-T Q.6/SG16, 2009
2. R.G. Gallager, D.C. Van Voorhis, Optimal source codes for geometrically distributed integer alphabets. IEEE Trans. Inf. Theory **21**(2), 228–230 (1975)
3. J. Heo, Y.-S. Ho, Efficient level and zero coding methods for H.264/AVC lossless intra coding. IEEE Signal Process. Lett. **17**(1), 87–90 (2010)
4. J. Heo, S.-H. Kim, Y.-S. Ho, Improved CAVLC for H.264/AVC lossless intra coding. IEEE Trans. Circuits Syst. Video Technol. **20**(2), 213–222 (2010)
5. Joint Video Team, Reference Software Version 13.2 [Online]. Available: http://iphome/hhi.de/shehring/tml/download/old_jm/jm13.2.zip
6. Y.-L. Lee, K.-H. Han, G.J. Sullivan, Improved lossless intra coding for H.264/MPEG-4 AVC. IEEE Trans. Image Process. **15**(9), 2610–2615 (2006)
7. H. Malvar, A. Hallapuro, M. Karczewicz, L. Kerofsky, Low-complexity transform and quantization in H.264/AVC. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 598–603 (2003)
8. D. Marpe, H. Schwarz, T. Wiegand, Context-based adaptive binary arithmetic coding in the H.264/AVC video compression. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 620–636 (2003)
9. J.-H. Nam, D. Sim, Lossless video coding based on pixel-wise prediction. Multimed. Syst. **14**(5), 291–298 (2008)
10. I.E.G. Richardson, H.264/MPEG-4, part 10, in *H.264 and MPEG-4 Video Compression* (Wiley, New York, 2003), pp. 201–207
11. K. Sayood, Lossless image compression, in *Introduction to Data Compression* (Morgan Kaufmann, San Mateo, 2006), pp. 170–172
12. G.J. Sullivan, T. Wiegand, Video compression—from concepts to the H.264/AVC standard. Proc. IEEE **93**(1), 18–31 (2005)
13. G.J. Sullivan, P. Topiwala, A. Luthra, The H.264/AVC advanced video coding standard: Overview and introduction to the fidelity range extensions, in *Proc. SPIE Conf., Special Session Adv. New Emerg. Standard: H.264/AVC* (2004), pp. 454–474
14. J. Teuhola, A compression method for clustered bit-vectors. Inf. Process. Lett. **7**, 308–311 (1978)
15. M.J. Weinberger, G. Seroussi, G. Sapiro, The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. IEEE Trans. Image Process. **9**(8), 1309–1324 (2000)
16. T. Wiegand, G.J. Sullivan, G. Bjøntegaard, A. Luthra, Overview of the H.264/AVC video coding standard. IEEE Trans. Circuits Syst. Video Technol. **13**(7), 560–576 (2003)