Embedding Auxiliary Data in H.264/AVC Compression Domain

Sang Beom Kim
Dept. of Electronics Eng.
Dongguk University
Seoul, 100-715, Korea

Chee Sun Won*
Dept. of Electronics Eng.
Dongguk University
Seoul, 100-715, Korea
cswon@dongguk.edu

ABSTRACT

In this paper, some auxiliary data about the video are embedded into the H.264/AVC compressed bits. To extract the auxiliary data, we just need a simple syntax parsing of the compressed bit stream to identify the locations of the embedded bits. Embedding data bits in the H.264/AVC compressed domain, the main concerns are to maintain the same bit rate before and after the embedding and to ensure no error propagation due to the embedding. Our solution for the above two issues is to embed the data bit into the *sign of Trailing Ones*. Since it is a coded-bit of one-bit fixed length, there will be no change in data rate after the embedding. Also, by checking the context of intra prediction modes for the neighboring 4×4 blocks, we can selectively embed the data into a specific set of *sign of Trailing Ones*, which guarantee no error propagation. The simplicity of the proposed data embedding and extraction enables real-time applications.

Categories and Subject Descriptors

I.4 [**IMAGE PROCESSING AND COMPUTER VISION**]: [Compression(Coding)]

General Terms

Digital Video Processing

Keywords

Data Hiding, H.264/AVC

1. INDRODUCTION

H.264/AVC [1] achieves compression efficiency up to almost double of the existing MPEG-2 compression standard [2]. With this high compression efficiency and network friendliness, it is expected that H.264/AVC will be used for a variety of multimedia applications including IPTV, wireless video communications, and digital multimedia broadcasting (DMB) system [3][4]. The DMB is the system for sending multimedia data to mobile devices including mobile phones. Also, the next broadcasting systems such as MVC(Multiview Video Coding) [5] and SVC(Scalable Video Coding) [6], which are being standardized by JVT(Joint Video Team) of ISO/IEC MPEG and ITU-T VCEG, are based on H.264/AVC. However, this high compression performance of H.264/AVC was possible by adopting more complex and computationally demanding new functionalities such as intra predictions and 4×4 block treatment. This implies that we have to pay more for the compression and decompression costs. Unfortunately, this could be a major obstacle for real applications. For example, if we need to see key frames of the video or to find scene changes, we may have to execute the full decompression process to obtain the full size video.

In this paper, we propose a method to avoid the full decompression of the H.264/AVC compressed bit stream to obtain some information about the video. That is, we embed the auxiliary data into the compressed bit stream. Then, whenever we need the information about the video, we can parse the compressed bit syntax to identify the locations of the coded bits with embedded data and to extract them from the compressed bit stream.

Data embedding (or watermarking) is to insert message bits into the video data for the purposes of protecting intellectual property, authenticating the content of the original video, and carrying auxiliary data via host video bit stream. The data embedding can be done either in uncompressed spatial domain or in compressed domain. The watermarking in uncompressed domain has more flexibility and is more robust against various attacks [7][8][9]. However, since most of the video data exist in the form of the compressed bitstream, we need to fully decode the compressed bit-stream for embedding a watermark into the uncompressed spatial video data and to recompress the watermarked video. This obvious computational overhead motivates the development of the watermarking in compression domain [9]. In between, there is a watermarking scheme, which embeds a watermark during the compression (encoding) process. For example, in [10], a watermark is inserted during H.264/AVC encoding process, while the detection is performed during the decompression (decoding) process. However, since most of original videos to be watermarked are stored in a compressed form, it is a common practice to embed a watermark into a *already-compressed* video stream. In [11], a watermark embedding

*Corresponding author. This work was supported by Seoul R&BN Program (SFCC).

AMBI-SYS 2008, February 11-13, Quebec, Canada

DOI 10.4108/ICST.AMBISYS2008.2858

into H.264/AVC compressed video by making use of skipped macroblocks. This method, however, results in an increase of video data rate as well as video quality degradation. In [12], they embed the watermark in the quantized AC coefficients of I frames. Although it is claimed to be quite small for QCIF video sequences, the bit-rate change after the watermarking still occurs. Moreover, no explicit solution was proposed for preventing the error propagation. To keep the bit-rate unchanged after the watermarking, in [13], a compressed bit domain watermarking algorithm manipulating the *sign of trailing ones* was proposed. However, although the error caused by the change of the sign bit may be minimal within a 4×4 DCT block, it may propagate to other macroblocks to be noticeably accumulated.

In this paper we solve this error propagation problem. That is, to avoid the error propagation due to the data embedding, we check the context of the intra prediction modes of the neighboring 4×4 blocks. Then, the watermark bit is inserted only when the intra prediction modes of the neighboring 4×4 blocks take no reference from the current block for the intra prediction.

2. THE PROPOSED METHOD

2.1 Selection of non-error-propagation-block (NEPB)

Our solution for the error propagation problem is to selectively embed digital message only into those blocks with no error propagation. For this, we have investigated the relationship between the current block and its four neighboring blocks in terms of their intra prediction modes (see Fig. 1). As shown in Fig. 2-(a), if, at least, one DCT coefficient of current block is modified by the data embedding, then its spatial boundary pixel values will be also changed. This subsequently causes the change of the gray levels in the neighboring blocks if the altered boundary values are referred by the neighboring blocks for the intra prediction. Specifically, as shown in Fig. 2-(a), there are four neighboring blocks (i.e., Block-(1), Block-(2), Block-(3), and Block-(4)) that can refer the boundary pixel values of the current block for the intra prediction. For example, any intra prediction mode for Block-(1) except modes numbered 0, 3, and 7 of Fig. 1 uses the boundary pixel values of the current block, causing an error propagation. This means that if the intra prediction mode of Block-1 happens to be one of the modes 0, 3, and 7, then the gray levels of the Block-(1) will not be affected by the data embedding for the current block. Similarly, we can identify the intra prediction modes for Block-(2), Block-(3), and Block-(4), which do not use the pixel values of the current block for the intra prediction. Note that in the case of Block-(4), as shown in Fig. 2-(b), we have an additional case for no error propagation. That is, if the current block is located at the position of A, then there will be no error propagation to block B because of the encoding order of 4×4 blocks in the 16×16 macroblock. In summary, if the four neighboring blocks of the current block have intra prediction modes of those listed in Fig. 2-(c) only, then we will have no error propagations due to the data embedding of the current block. Also, if the intra prediction mode happens to be 16×16 intra prediction mode, then nine of 4×4 blocks (i.e., the shaded blocks in Fig. 2-(d)) in the left upper corner of the 16×16 macro block will not be referred. We call the set of those 4×4 blocks listed in the table of Fig. 2-(c) and the shaded blocks in Fig. 2-(d) as non-error-propagating-block (NEPB). For those 4×4 blocks belonging to NEPB are the candidates for our data embedding blocks. Now, for each j^{th} 4×4 block in i^{th} macroblock, we indicate whether the block is in NEPB or not by C_{ij} as

$$C_{ij} = \begin{cases} 1, & \text{if the block at (i,j) is in NEPB} \\ 0, & \text{otherwise.} \end{cases}, \qquad (1)$$

2.2 Selection of coded DCT bits for data embedding

Next problem to be solved is the bit-rate change after the DCT coefficient modifications. As already mentioned, the bit-rate change after the compression domain watermarking may destroy the synchronization between audio and video. To solve this problem, we need to investigate CAVLC(Context Adapted Variable Length Coding) process, which is newly adopted in H.264/AVC standard. There are five elements for CAVLC in H.264/AVC. They are "Coefficient token", "Sign of T1", "Level", "Total zeros", and "Run before". "Coefficient token" represents the number of non-zero coefficients (TC: Total Coefficient) and trailing ones(T1). There are four look-up tables to be selected for encoding "Coefficient token" for all 4×4 blocks, three variable length code tables and a fixed one. "Level" means the value of the remaining nonzero coefficients except for T1s in the block, and the "Run before" means the number of zeros preceding each non-zero coefficient in the reverse zig-zag order. Note that four elements except "Level" (i.e., "Coefficient token", "Sign of T1", "Total zeros" and "Run before") are closely related with TC and T1. That is, any change of TC and T1 affects the whole bit-rates. Therefore, our strategy for preventing bit-rate change is not to modify the TC and T1 values.

Among all elements for the CAVLC, it turns out that only "sign of T1" is encoded with an one-bit fixed length. Specifically, as shown in Table 1, the "sign of T1" satisfies the following requirements for the best data embedding:

- no bit-rate change: fixed length coding parameters are preferable
- minimization of visual degradation: high frequency components are preferable

That is, each sign bit of the trailing ones is encoded by a fixed single bit such that "+1" is coded to a binary number "0" and "-1" to "1". This is a fixed length (one-bit) coding. Also, the sign bits belong to the trailing ones and they are the QDCT coefficients at the highest frequency components of the 4×4 QDCT. Thus, if we change the sign bit of the "trailing ones" for the bit-embedding, then the two requirements mentioned above are satisfied. Thus, the sign bit of the last Trailing One can be a good candidate for data embedding and we basically exploit the "sign of T1" for our data embedding.

2.3 Data embedding

As explained in the previous section, we will modify the last "trailing-ones-sign-flag(T1-sign-flag)" to embed the binary data. The flow of the data embedding algorithm is illustrated in Fig 3. After parsing the H.264/AVC bitstream, we check the slice type. If slice type is an inter slice, then, since there is no intra prediction propagation,

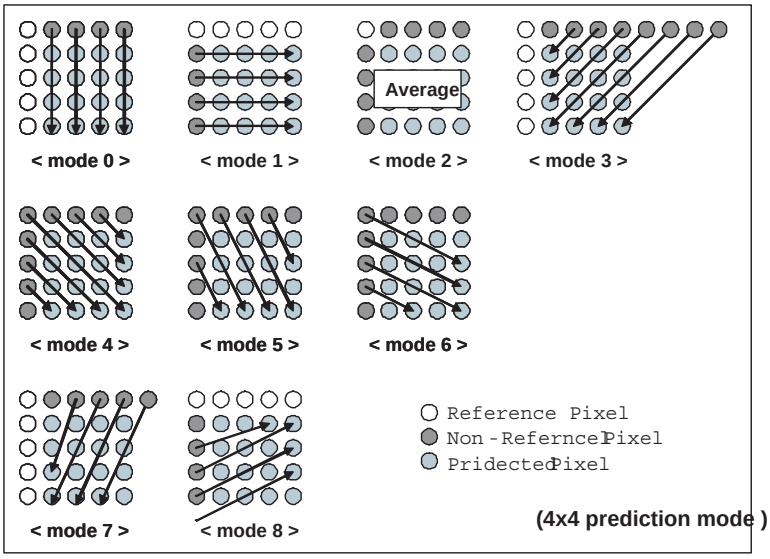

Figure 1: Intra prediction modes and referred boundary pixels for neighboring blocks.

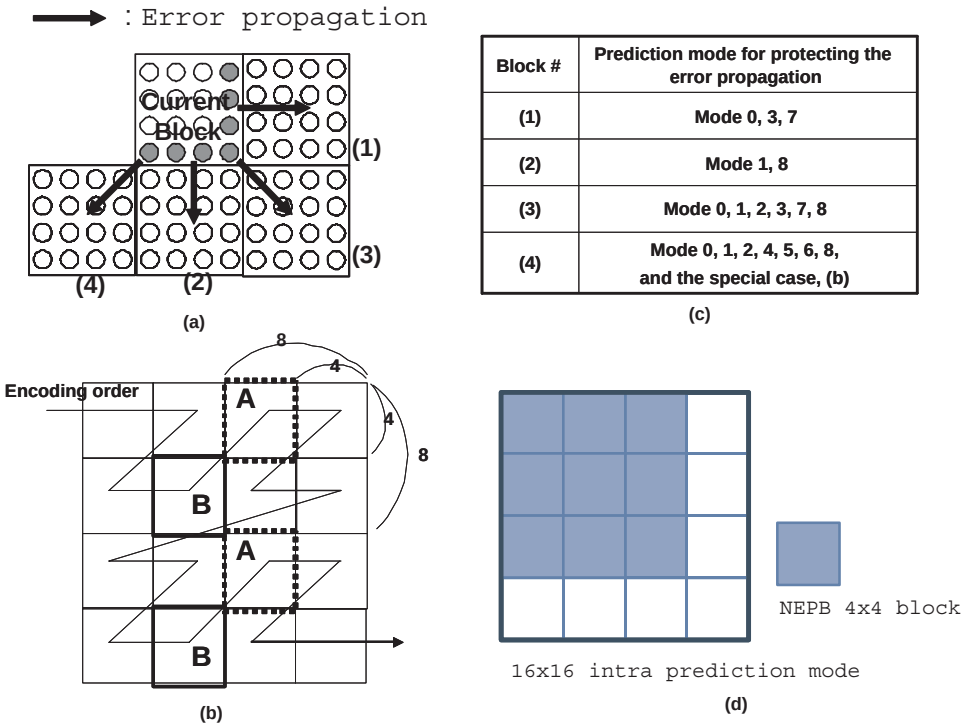

Figure 2: (a) Neighboring 4×4 prediction blocks from the current one, (b) special case for Block-(4), (c) intra prediction modes for NEPB, (d) NEPB for 16×16 intra prediction mode.

Table 1: Characteristics of coded data.

Coding Parameter	Values to be coded	Bit-length	Frequency components
Coefficient token	TC & T1	Variable	All
Trailing ones sign flag	Signs of T1	Fixed	Highest
Level prefix	Non-zero coefficients excluding T1	Variable	Medium to low
Level suffix	Non-zero coefficients excluding T1	Variable	Medium to low
Total zeros	Total number of zeros occurring after the first non-zero coefficients	Variable	NA
Run before	Number of zeros preceding each non-zero coefficient	Variable	NA

all 4×4 DCT blocks can be used for data embedding. But if slice type turns out to be an intra slice, then we extract the T1-sign-flag from the H.264/AVC bit-stream. And then, the relationship between the 4×4 block which includes the extracted T1-sign-flag and its prediction mode neighboring blocks are examined to check whether the block is in NEPB or not. Here, let us denote the first T1-sign-flag in the encoding order of the j^{th} 4×4 block in i^{th} macroblock as S_{ij} which is the first sign bits for the trailing ones in the reverser order of zig-zag scan. This means that S_{ij} is the highest frequency component among the non-zero coefficient values. Now, denoting the binary bit to be embedded at the j^{th} 4×4 DCT block in i^{th} macroblock as b_{ij}, we can embed the watermark bit b_{ij} to the sign bit S_{ij} as follows. If the block is an intra slice, then we embed the data bit as

$$S'_{ij} = \begin{cases} b_{ij} & \text{if } C_{ij} = 1 \\ S_{ij} & \text{otherwise .} \end{cases}, \tag{2}$$

Otherwise, for an inter slice, we have

$$S'_{ij} = b_{ij} \tag{3}$$

where S'_{ij} denotes the modified sign bit. As a result of the embedding in (2) and (3), only the last (and highest) frequency component of the zig-zag scanned QDCT will be altered due to the bit-embedding, if there is at least one T1.

The extraction is just the reverse of the embedding. First, parsing the watermarked H.264/AVC bit-stream, the slice type and the intra prediction modes of neighboring blocks are extracted. The extraction method depends on the slice type. If the slice type is an intra slice, then watermark bits are extracted only when it belongs to NEPB. That is, if C_{ij} is 0, then the block is not used for the data embedding and we just skip it. Otherwise, for the intra slice with $C_{ij} = 1$ and for the inter slice, we extract the embedded data bit as

$$\hat{b}_{ij} = \hat{S}_{ij}, \tag{4}$$

where $\hat{b}_{ij}$ represents the extracted watermark bit at the j^{th} 4×4 DCT block in i^{th} macroblock and $\hat{S}_{ij}$ is the received, possibly altered from the original S'_{ij}, first sign bit of the trailing ones.

3. EXPERIMENTS

In our experiments, four CIF(352×288) standard videos (Mobile, City, Foreman, and Tempete) and three D1(720×480) standard videos are used. All videos are compressed by the standard reference JM-software [14] with various QP(24, 26, 28, and 30). The experimental results with the previous method are compared in Table 2. The PSNR's are slightly lower than the original bit-stream, but they are much higher than those of the previous method [13]. For all videos, the average PSNR decreases are 1.02dB and 0.92dB at CIF and D1, respectively. On the other hand, the average PSNR decreases of the previous method [13] are 6.9dB and 7.85dB at CIF and D1 movies, respectively. The average payloads of the embedded data bits per frame range from 143 to 1074 bits/frame, which are sufficient to store at least 17 characters per frame. The subjective visual quality in Fig. 4 shows that there is no noticeable degradations for our method, while errors are noticeable in the previous method in the directions of intra prediction modes (see Fig. 4-(c)). Finally, in our method, there is no flickering artifact in the temporal direction of the video sequence which occurred in the previous method [13]. Of course, there is no bit-rate change after the message bits embedding.

4. CONCLUSION

In this paper, a novel data embedding method is proposed for H.264/AVC bit-stream. We have solved two problems of the compression domain (more specifically, bit-domain) data embedding, the spatial error propagation and the bit-rate increase. For data embedding, we embed the message bit into the NEPB blocks only, yielding no error propagation due to the intra prediction. And, to make the bit-rate unchanged after the watermarking, we have selected the sign of trailing ones as the container for data embedding. In conclusion, our method satisfies two requirements: no bit-rate change and minimization of visual degradation. Experimental results show that the errors are not propagated and the bit-rate is the same as the original one.

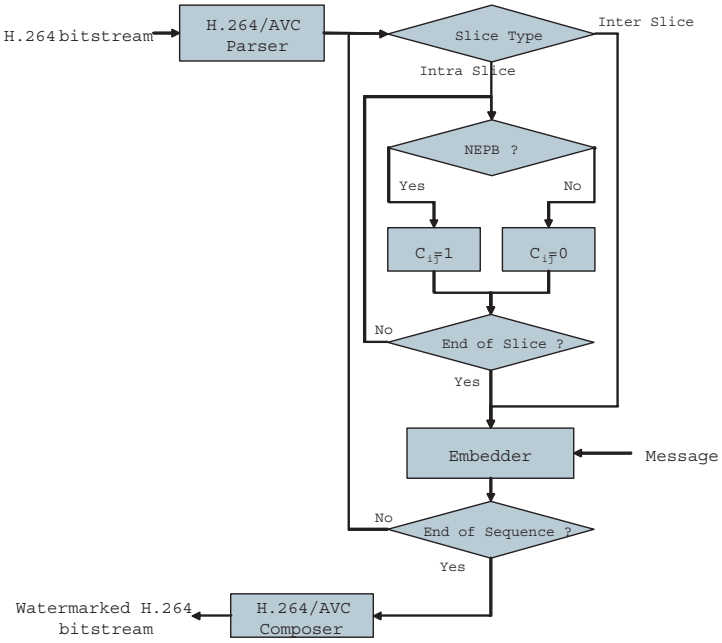

Figure 3: Flowchart for the proposed data embedding method.

Table 2: Comparisons with the previous method.

Sequence		Proposed Method		Method of *Kim* [11]	
		Payload (bits / frame)	PSNR decrease	Payload (bits / frame)	PSNR decrease
CIF (352x288)	Mobile	327	0.78	298	4.69
	City	143	0.70	112	7.46
	Foreman	183	1.88	157	8.42
	Tempete	247	0.74	246	6.93
D1 (720x480)	Football	707	0.67	484	7.76
	Mobile	1074	1.27	1118	8.10
	Garden	693	0.81	691	7.69

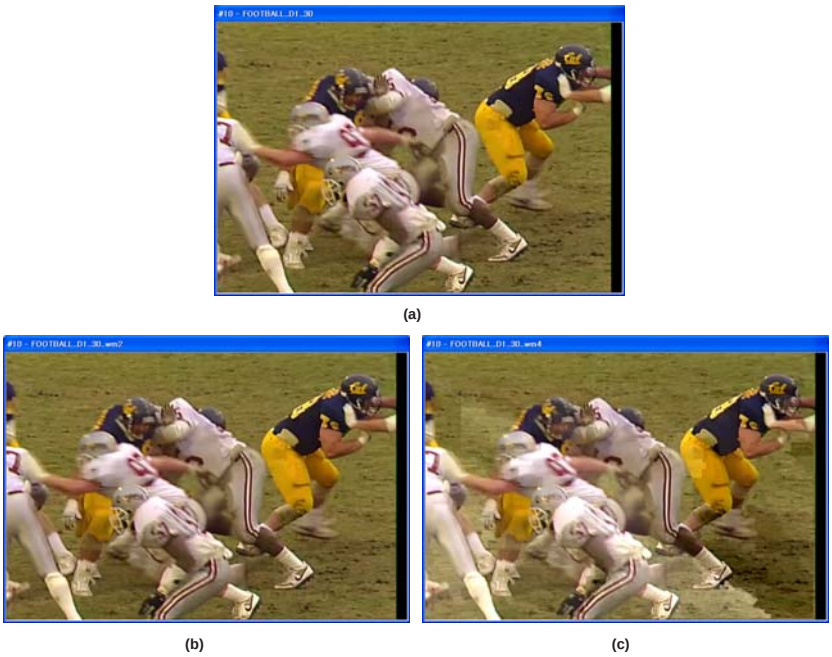

Figure 4: The visual artifact comparisons: (a)original image, (b)the proposed method, and (c)the previous method[13].

5. REFERENCES

[1] E.G. Ricardson, H.264/AVC and MPEG-4 Video Compression, Wiley, 2003.

[2] J. Ostermann, et al., "Video coding with H.264/AVC:Tools, performance, and complexity," IEEE Circuits and Systems Magazine, First Quarter, pp.7-28, 2004.

[3] D. Marpe, T. Wiegand, and G.J. Sullivan, "The H.264/MPEG-4 advanced video coding standard and its application," IEEE Communication Magazine, vol 44, Issue 8, PP.134-143, 2006.

[4] F.O. Gauthier, "What is DMB?", CBC Technology Review, Issue 2, 2003.

[5] A. Vetro, Y. Su, H. Kimata, and A. Smolic, Joint Multiview Video Model(JMVM) 2.0, Output Document of JVT Meeting at Hangzhou, N8459, Oct. 2006.

[6] T. Wiegand, G. Sullivan, J. Reichel, H. Schwarz, and M. Wien, Joint Draft 8 of SVC Amendment, Output Document of JVT Meeting at Hangzhou, JVT-U201, Oct. 2006.

[7] I. Setyawan, R. L. Lagendijk, "Low Bit-rate Video Watermarking using Temporally Extended Differenctial Energy Watermarking (DEW) algorithm," Proc. Of SPIE, Vol.4314, 2001.

[8] Y. Wang, and A. Pearmain, "Blind MPEG-2 Video Watermarking Robust against Geometric Attack: A Set of Approaches in DCT Domain," IEEE Trans. On Image Processing, Vol.15, No.6, June, 2006.

[9] A. Hanjalic, G. C. Langelaar, P. M. B. van Roosmalen, J. Biemond, and R. L. lagendijk, Advances in Image Communication Vol.8 Image and Video Databases: Restoration, Watermarking and Retreival, pp. 171-310, Elsevier, 2000.

[10] G. Qiu, P. Marziliano, A. Ho, D. He, and Q. Sun, "A hybrid watermarking scheme for H.264/AVC video," International Conf. on Pattern Recognition (ICPR), 2004.

[11] D. Profrock, H. Richter, M. Schlauweg, and E. Muller, "H.264/AVC video authentication using skipped macroblocks for an erasable watermark," Proc. of SPIE, vol. 5960, 2005.

[12] M. Noorkami and R.M. Mersereau, "Compressed-domain video watermarking for H.264," IEEE International Conference on Image Processing, vol.2, pp. II-890 3, 2005.

[13] S. M. Kim, S. B. Kim, Y. P Hong, and C. S. Won, "Data Hiding on H.264/AVC Compression Domain," International Conference on Imge Analysis and Recognition," Aug. 2007.

[14] http://iphome.hhi.de/suehring/tml/