REWRITABLE DATA EMBEDDING ON MPEG CODED DATA DOMAIN

Katsuhiro Nakajima, Kiyoshi Tanaka*, Tetsuya Matsuoka** and Yasuyuki Nakajima****

* Faculty of Engineering, Shinshu University, Japan
** Research Laboratory, EPSON AVASYS Corp, Japan
*** KDDI R&D Laboratories Inc, Japan

ABSTRACT

In this paper, we propose a rewritable data embedding scheme on MPEG coded data domain for content managements including DRM, content controlling and indexing. Data embedding is performed in a block by block basis, where the length of zero run and the value of dummy AC component of quantized DCT coefficients are used as a data carrier. In the detection process, we can reconstruct the MPEG coded data that is very close to the original one, which enables us not only to rewrite embedded data but also retain the original MPEG video quality. In the experiment, we show that up to a few kbits/frame data embedding without any large PSNR penalty and data recovery can be realized using typical MPEG-1 coded stream.

1. INTRODUCTION

Recently, various kinds of data hiding techniques for multimedia contents have been actively studied aiming for such purposes as secret communication, copyright protection, tamper detection, and attachment of attribute information[1][2][3][4]. Data carrier for these data hiding can be classified into the following four groups; (i) time domain samples such as pixels, (ii) frequency domain components after orthogonal transform such as FFT, (iii) statistics such as histogram, and (iv) coded data itself such as JPEG and MPEG.

Since MPEG and JPEG are widely used for compression of video and image data, coded data domain data hiding has been also studied recently. For example, Takagi et al. [5] proposed a data embedding scheme on MPEG coded data, which uses the last AC component (the 63rd one) of each block only in I-picture frames. Although this scheme can achieve rewritable data embedding and high image quality, the total payload (total amount of data embedded) is very limited, 11 bytes per frame typically, and total bit count of coded data is also increased significantly. On the other hand, Miyake et al. [6] proposed a data embedding scheme on JPEG codes for steganography purposes, which can increase the payload considerably while keeping thestatistics of the original content. Although this scheme keeps the distribution of frequency components apparently similar to the original one that is a suitable feature for steganograply, we can no longer reconstruct the original codes after embedding.

Since rewritable data embedding is useful when data updating is required for DRM, content controlling, and indexing purposes, practical data embedding and decoding scheme in terms of data hiding rate, coded data amount variation, and picture quality variation for rewriting data would be required. In this paper, we propose a novel rewritable data embedding scheme on MPEG coded data domain. This scheme can achieve quite fast data embedding and detection because only partial decoding and encoding of MPEG coded data are required. Basically, data hiding is achieved block by block by using the length of zero run and the value of dummy AC component based on the position of the last non-zero AC component as a data carrier. In the detection process, we can specify the correct position of final non-zero AC component in most cases, and therefore very close MPEG coded data to the original MPEG data can be reconstructed and picture quality can be maintained after the data embedding.

2. PROPOSED SCHEME

2.1. Data Embedding

As for MPEG compressed domain data embedding, several types of coding parameters can be used. Here, we use DCT coefficients as data carrier since more flexible data embedding can be expected in 8×8 block based DCT domain rather than in 16×16 block based motion compensation domain. We denote each 8×8 block pixels, its corresponding DCT components and their quantized values as $f_{ij}(i, j = 0, 1, \cdots, 7)$, $F_{uv}(u, v = 0, 1, \cdots, 7)$ and $R_{uv}(u, v = 0, 1, \cdots, 7)$, respectively. $R_k(k = 0, 1, \cdots, 63)$ are one-dimensional values of $R_{uv}(u, v = 0, 1, \cdots, 7)$ after ZigZag scanning. R_0 (DC component) is separately coded by the predictive coding scheme in case of I-picture encoding while it is coded together with other AC components (R_1 to R_{63}) in P-picture and B-picture encoding.

At first, we seek the position $p(0 \leq p \leq 63)$ of the last

0-7803-9332-5/05/$20.00 ©2005 IEEE

non-zero AC component in $R_k(k = 0, 1, \cdots, 63)$. In this work, we use the length of zero run from p to the position p' where a dummy non-zero AC component is attached after p, and the value $R_{p'}$ of the dummy component as a data carrier. Before embedding, we set a marking position P_m for data embedding as

$$P_m = 63 - 2^{\alpha}, \tag{1}$$

which ensures to embed α bits after p by using the length of zero run. If $p \geq P_m$, which means the last non-zero AC component exists after P_m, we set $R_k = 0(k = P_m, P_{m+1}, \cdots, 63)$, and again seek the position p of the last non-zero AC component within the range of $k < P_m$. Only in this case, we may lose the original information in MPEG coded data, which may not be the cases when α is small values such as 1$\sim$3.

After seeking the position p in R_k, we embed data by using $p' - p$ and $R_{p'}$ as follows. Here we prepare a sequence of binary data $B = b_r(r = 0, 1, \cdots) \in \{0, 1\}$. First, we extract α and β bits successively from B and transform them to decimal numbers N_α and N_β, respectively. Then we determine the position of p' and the value of $R_{p'}$, for example, as

$$p' = p + N_\alpha + 1, \tag{2}$$

and

$$\begin{cases} R_{p'} = -2^{\beta-1} + N_\beta \quad (N_\beta < 2^{\beta-1}) & (3) \\ R_{p'} = -2^{\beta-1} + N_\beta + 1 \quad (N_\beta \geq 2^{\beta-1}), & (4) \end{cases}$$

respectively. Note that $R_{p'}$ should become to a non-zero value in Eqs.(3)$\sim$(4). In this way, we embed $\alpha + \beta$ bits of binary data B in $\{R_{uv}\}$ for a current block $\{f_{ij}\}$. This procedure is repeated for all the blocks in every frame of the content, and the MPEG codes are partially re-encoded accordingly.

2.2. Data Detection

As for the detection of embedded data from MPEG compressed data, quantized DCT coefficients $R_k(k = 0, 1, \cdots, 63)$ are recovered for the current block, which can be done without dequentization and inverse DCT. Then, we find the positions of the last and the one before the last non-zero AC components, and set them to $p'(0 \leq p' \leq 63)$ and $p(0 \leq p \leq 63, p < p')$, respectively.

Then, we can first detect α bits of the embedded data from the length of zero-run between p' and p. That is, a decimal number N_α is obtained from Eq.(5),

$$N_\alpha = p' - p - 1, \tag{5}$$

and α bits of binary data can be recovered from N_α. Secondly, we can detect β bits of partial data embedded from the value of $R_{p'}$. That is, a decimal number N_β can be obtained in the following equations,

$$\begin{cases} N_\beta = R_{p'} + 2^{\beta-1} \quad (R_{p'} < 0) & (6) \\ N_\beta = R_{p'} + 2^{\beta-1} - 1 \quad (R_{p'} > 0), & (7) \end{cases}$$

and β bits of binary data can be recovered. In this way, we can detect $\alpha + \beta$ bits of data from partially decoded $\{R_k\}$ for the current block. This detection procedure is repeated for all the blocks in every frame of the content, and then the entire binary data $B = b_r(r = 0, 1, \cdots)$ can be reconstructed.

2.3. Reconstruction of Original Codes

As mentioned in section **2.1**, the last non-zero AC component $R_{p'}$ in $R_k(k = 0, 1, \cdots, 63)$ is a dummy component for data embedding and the previous one R_p can be considered as the "true" last non-zero component in most cases. Thus, we can reconstruct MPEG coded data that is almost the same as the original ones by just discarding the attached dummy component $R_{p'}$ in $\{R_k\}$. Therefore, a video decoder which includes this code recovery function can playback the content with almost the same quality as that of original MPEG data. On the other hand, although normal MPEG decoder can also playback the data embedded content, playback quality may be degraded when compared with the original MPEG data.

This model is slightly different from other approaches where the playback quality of data embedded contents using normal decoder is basically the same as that of the original contents. However, these models are for write-once type of data embedding and therefore the reconstruction of the original MPEG coded data is difficult. On the other hand, the proposed method can realize the reconstruction of the original MPEG data, and therefore it can provide rewriting of data embedding. Although the original MPEG picture quality is not maintained in the data embedded stream when the normal MPEG decoder is used, its quality can be recovered when the normal MPEG decoder is used after MPEG data is reconstructed, or when the MPEG decoder with data reconstruction function is used.

2.4. Embedding Parameters

In **2.1** we use two parameters α and β to control the payload. Obviously, we can increase the payload as we increase $\alpha + \beta$. However, we should be also careful about the degradation of picture quality as well as the increase of coded data due to data embedding. Here, we prepare two parameters to control picture quality and coded data amount. The distribution of frequency components $\{R_{uv}\}$ may sometimes be extremely concentrated in low frequency band in flat region, for example. In this case R_p will be placed in low frequency

band as well, which may cause serious degradation of image quality by data embedding. Thus, we set a parameter P_Q, and use the length of zero-run $p' - P_Q$ instead of $p' - p$ as a data carrier if $p < P_Q$. Also, in case of $p < P_Q$, the zero-run between p and p' may become excessively long by embedding (attaching a dummy AC component), which may cause serious increase of codes. To avoid this, we prepare another parameter P_C, and skip the data embedding if $p - P_Q > P_C$.

3. PERFORMANCE EVALUATION

In our computer simulation, we used MPEG-1 encoder coding at 1.5[Mbit/s] with "Flower Garden" (SIF size: 352×240 pixels, 150 frames, I:B:P=1:10:4 frames) as a test sequence. **Table 1** shows the results of payload for each picture type when varying $\alpha + \beta$ values at $P_Q = 10$ and $P_C = 5$. It can be seen that the payload increases almost linearly as we increase $\alpha+\beta$. Although it depends on bit allocation of rate control mechanism, the size of payload in I-picture and P-picture is much larger than that of B-picture as expected. **Table 2** shows PSNR difference when the embedded MPEG data and the original MPEG data are compared varying α and β values. The other conditions are the same as that of **Table 1**. In the table, the upper columns indicate the results without code recovery function (i.e when normal MPEG decoder is used.), whereas the lower columns show the results with code recovery function as mentioned in **2.3**. From these results, PSNR difference is kept small with code recovery function even when α and β are increased except $\alpha = 4$. **Table 2** also shows the increase of MPEG coded data amount. We can see that the data embedding results in about 5$\sim$8% increase in the coded data amount while the code recovery results in almost the same or little bit less size than the original size.

Table 1. Payload per frame for $\alpha + \beta$ [bits/f]

$\alpha+\beta$	I-picture	P-picture	B-picture
1	1,532	1,268	476
2	3,064	2,536	952
3	4,600	3,800	1.432
4	6,136	5,072	1,880
5	7,664	6,312	2,264

($P_Q = 10$, $P_C = 5$, C_R=1.5[Mbps])

Next we show some results on the effect of the performance by other parameters, P_Q and P_C. **Fig.1** shows the amount of payload when P_Q value is changed at α=β=1. From the figure, it can be seen that the payload decreases in all kinds of pictures as we increase P_Q, for example, the amount of payload becomes about half at $P_Q = 40$ when compared with $P_Q = 10$. **Fig.2** shows the PSNR results for the same condition as **Fig.1**. It is noted that PSNR variation is small when P_Q is changed from 10 to 40. This means that PSNR penalty can be kept small even when the amount of payload becomes large by decreasing P_Q value. **Fig.3** shows the variation of coded data amount when changing P_C values. It can be seen that the coded data amount increases as we increase P_C because the number of skipping will be decreased.

Table 2. PSNR difference and the increase of coded data amount when varying α and β values

β	α	code recov.	PSNR difference [dB] I-picture	P-picture	B-picture	codes [%]
1	1	OFF	-1.433	-1.499	-1.158	+5.4
		ON	-0.028	-0.021	-0.012	$\pm$ 0
	2	OFF	-1.444	-1.522	-1.151	+6.2
		ON	-0.042	-0.032	-0.019	-0.2
	3	OFF	-1.611	-1.678	-1.247	+5.4
		ON	-0.284	-0.248	-0.150	-2.5
	4	OFF	-2.241	-2.342	-1.692	+2.5
		ON	-1.067	-1.042	-0.677	-7.8
2	1	OFF	-2.523	-2.787	-2.115	+6.8
		ON	-0.028	-0.021	-0.012	$\pm$ 0
	2	OFF	-2.434	-2.712	-2.025	+8.3
		ON	-0.042	-0.032	-0.015	-0.2

($P_Q = 10$, $P_C = 5$, C_R=1.5[Mbps])

Furthermore, we show the results on the payload as we change the coding rate C_R in **Fig.4**, where α=β=1, $P_Q = 30$, and $P_C = 5$. As we can see in the figure, the payload increases linearly in all kinds of pictures because the number of coded DCT coefficients increases. Therefore, higher C_R can realize larger data payload in our scheme. **Fig.5** shows the PSNR results under the same condition as **Fig.4**. From these figures, it can be seen that both the PSNR and the amount of embedded data can be scalably increased as we increase coding bit rate.

4. APPLICATION TO INDEXING

We have developed MPEG video player application software with indexing capability using the proposed data embedding scheme. **Fig.6** shows one of the examples where spotlighting information is embedded in the MPEG stream. When normal MPEG player playbacks this content, video can be viewed without any indexing including a spotlight. On the other hand, when the MPEG player with the embedded data detection function is used, users can playback video with specific indexing information such as a spotlight, where the position and spotlight mask information are embedded for each frame as an example. By including several indexing information in the embedded data, users can interactively select indexing information and use it for video playback. For example, secret text, figure data, and/or any

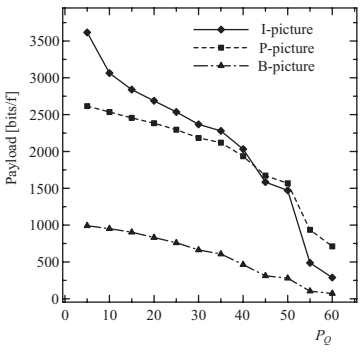

Fig. 1. Payload per frame for P_Q [bits/f] ($\alpha = \beta = 1$, $P_C = 5$, $C_R = 1.5$[Mbps])

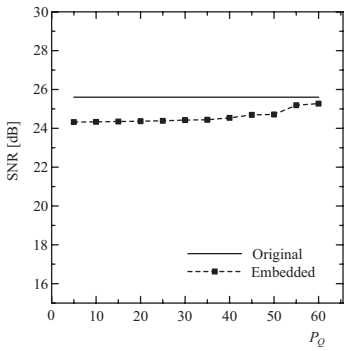

Fig. 2. Image quality measured by PSNR for P_Q [dB] ($\alpha = \beta = 1$, $P_C = 5$, $C_R = 1.5$[Mbps])

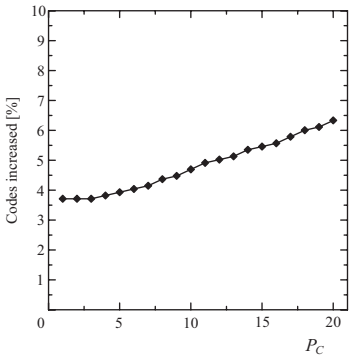

Fig. 3. Increase of coded data amount for P_C [%] ($\alpha = \beta = 1$, $P_Q = 30$, $C_R = 1.5$[Mbps])

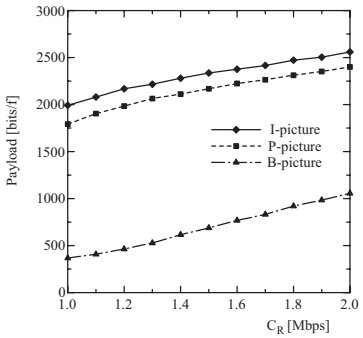

Fig. 4. Payload per frame for C_R [bits/f]

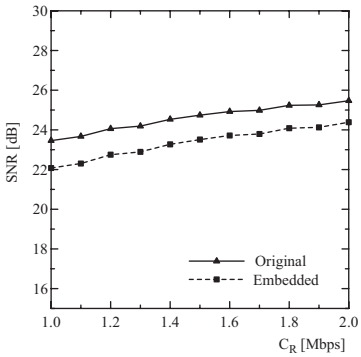

Fig. 5. Image quality measured by PSNR for C_R [dB]

Fig. 6. An example of the way of using index information (partially spotlighted an important object)

object data can be inserted in a specified frame. Although for these applications, meta data file can be separately prepared and used with video playback, content management would be much convenient if video content itself contains all the necessary information for indexing, for example. The simplest way for this is the use of user data area in MPEG data. However, when a content provider want to keep these information from any manipulation by a third party, data hiding with rewriting capability mentioned above would be considered as an appropriate approach.

5. CONCLUSIONS

We proposed a rewritable data embedding scheme on MPEG coded data domain. This scheme can realize very fast data embedding and detection due to the use of quantized DCT coefficient as a data carrier. It is shown in the experiments that users can control the amount of payload, picture quality, the amount of coded data by adjusting a few parameters in this scheme. It is also shown that almost the same MPEG coded data as the original one can be successfully reconstructed, which enables embedded data rewriting. We also addressed on the video playback software with video indexing capability using the embedded data.

6. REFERENCES

[1] K. Matsui, *Fundamentals of Digital Watermark*, Morikita Publishing, 1998 (in Japanese).

[2] N. Morimoto, W. Bnder, D. Gruhl and A. Lu, "Techniques for Data Hiding," *IBM Systems Journal*, 1996, Vol. 35, pp. 313-336.

[3] I. Setyawan, G. C. Langelaar and R. L. Lagendijk, "Watermarking Digital Image and Video Data,"*IEEE Signal Processing Magazine*, 1998, Vol. 77, pp. 20-46.

[4] F. A. P. Petitcolas and S. Katzenbeisser, *Information Hiding: Techniques for Steganography and Digital Watermarking*, Artech House, 2000.

[5] A. Takagi, H. Kiya, Y. Noguchi and H. Kobayashi, "A Method of Inserting Binary Data into MPEG Video Compressed Domain", *IEICE Trans.*, 1999, Vol.E82-A, pp. 1485-1492.

[6] K. Miyake, M. Iwata and A. Shiozaki, "Digital Steganography Utilizing Features of jpeg images", *IEICE Trans.*, 2004, Vol.E87-A, pp. 929-936.