Complete Video Quality-Preserving Data Hiding

KokSheik Wong, *Student Member, IEEE*, Kiyoshi Tanaka, *Member, IEEE*,
Koichi Takagi, and Yasuyuki Nakajima, *Member, IEEE*

***Abstract*— Although many data hiding methods are proposed in the literature, all of them distort the quality of the host content during data embedding. In this paper, we propose a novel data hiding method in the compressed video domain that completely preserves the image quality of the host video while embedding information into it. Information is embedded into a compressed video by simultaneously manipulating Mquant and quantized discrete cosine transform coefficients, which are the significant parts of MPEG and H.26x-based compression standards. To the best of our knowledge, this data hiding method is the first attempt of its kind. When fed into an ordinary video decoder, the modified video completely reconstructs the original video even compared at the bit-to-bit level. Our method is also reversible, where the embedded information could be removed to obtain the original video. A new data representation scheme called reverse zerorun length (RZL) is proposed to exploit the statistics of macroblock for achieving high embedding efficiency while trading off with payload. It is theoretically and experimentally verified that RZL outperforms matrix encoding in terms of payload and embedding efficiency for this particular data hiding method. The problem of video bitstream size increment caused by data embedding is also addressed, and two independent solutions are proposed to suppress this increment. Basic performance of this data hiding method is verified through experiments on various existing MPEG-1 encoded videos. In the best case scenario, an average increase of four bits in the video bitstream size is observed for every message bit embedded.**

***Index Terms*— Complete image quality-preserving, data hiding, DCT, H.261, H.263, MPEG, reverse zerorun length (RZL), reversible.**

I. Introduction

With THE advancement of computer and broadband internet technologies, it is now possible to stream high-quality video and to watch video online. Websites such as YouTube, DailyMotion, iFilm, etc. offer free viewing and download services, while portable devices such as iPod, PSP, Zen, and so on make it possible to playback a video anytime anywhere. As a result, digital video has become a popular multimedia content for both online and offline environments. While digital video is still gaining popularity, it is important to consider ways to protect the contents from malicious use, efficient ways to search the desired contents in the database, secure ways to provide extra features for upgraded viewers, reliable ways to recover from transmission error for uninterrupted viewing, etc. for the future of digital video. To accomplish such tasks, data hiding is one of the fields that provides the solutions [1], [2].

In general, there are two types of data hiding for video: one that hides the video content itself (video encryption or scrambling) so that nobody understands what is being transmitted [3]–[5]; the other that embeds external information into the video, hence utilizing video as the data host. We consider the latter in this paper. Under this category, one of the basic requirements for a data hiding method is the ability to produce video of high image quality. On top of that, additional properties are desired, depending on the application in question. In case of watermarking, the information embedded into a video should be able to withstand some common image processing attacks such as re-compression at a different bitrate, random video frame dropping, resizing, etc. [6]. In case of steganogrphy, the embedded information should stay undetectable with respected to steganalysis [7], which is a process for revealing the existence of hidden information in a suspicious video. In applications such as annotation or indexing, even though it is not a compulsory required property, it is usually preferable to achieve reversibility so that the embedded information could be removed to restore the original video. Other applications of data hiding could be found in [1], [8], [9].

Some representative data hiding methods in video domain could be found in [10]–[18]. For example, Kiya *et al*. embed information into an MPEG compressed video by modifying coefficients at selected location(s) in each 8×8 qDCTCs (quantized DCT coefficients) block [10]. The quantization table is further modified to suppress distortion. Nakajima *et al*. proposed a high carrier capacity data hiding method utilizing the idea of zerorun length coding in MPEG domain [16]. After zigzag scanning, a dummy nonzero value is inserted at a location that is x units away from the the original last nonzero qDCTC, where x depends on the data to be embedded. Zhang *et al*. utilize MV (motion vectors) in P and B-pictures as the data carriers for data embedding [11]. An MV is selected based on its magnitude, and its angle guides the modification operation. In these existing works, qDCTC or/and MV is/are usually utilized as the data carrier. Therefore, modifications done to the video during data embedding may alter the video bitrate. As a result, researches are carried out to maintain the original video bitrate for avoiding buffer overflow or underflow during video playback [19], [20]. For instance, Pranata *et al.* embed information into a frame by evaluating the combined bit lengths of a set of multiple watermarked variable length coding (VLC) codewords [19]. They successively replace the watermarked VLC codewords having the largest increase in

Manuscript received August 25, 2008; revised November 21, 2008. First version published May 12, 2009; current version published September 30, 2009. This paper was recommended by Associate Editor M. Barni.

K. Wong and K. Tanaka are with the Department of Electrical and Electronics Engineering, Faculty of Engineering, Shinshu University, 4-17-1 Wakasato, Nagano 380-8553, Japan (e-mail: koksheik@shinshu-u.ac.jp; ktanaka@shinshu-u.ac.jp).

K. Takagi is with the KDDI R&D Laboratories Inc., 2-1-15 Ohara, Fujimino-shi, Saitama 356-8502, Japan (e-mail: ko-takagi@kddilabs.jp).

Y. Nakajima is with the KDDI Corporation, Garden Air Tower, Chiyoda-ku, Iidabashi 3-10-10, Tokyo 102-8460, Japan (e-mail: ys-nakajima@kddi.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2009.2022781

bit length with their corresponding unmarked VLC codewords until a target bit length is achieved. Recently, a data hiding method using Mquant (i.e., the scaling factor in rate controller) as the data carrier is proposed, and this method always produces video with exactly the same bitrate as the original cover video even after data embedding [21]. Suboptimal histogram preserving modification scheme is also proposed to maintain the distribution of Mquant before and after data embedding.

In this paper, we focus on complete image quality-preservation, reversibility, and efficient data representation scheme as the fundamental research of data hiding in compressed video domain. Even though the aforementioned data hiding methods generally produce image/video of high quality regardless of their applications and data carrier in use, the image quality of the modified video is always lower than that of the original video. This is a critical drawback because these data hiding technologies cannot be utilized in applications where image quality degradation is not permitted. To solve this problem, we propose a novel data hiding method in the compressed video domain that completely preserves the image quality.

To the best of our knowledge, there is no data hiding method that completely preserves the image quality during data embedding, and this method is the first attempt of its kind. This method is also reversible, and it is applicable not only to the existing MPEG-1/2/4 or H.261/3 encoded videos but also to the encoding process of MPEG-1/2/4 or H261/3 video from a sequence of raw pictures. One of the possible applications of this method is video annotation where high image quality and reversibility are greatly desired because we can provide high-quality video as well as advanced special functions for searching, playback control, and/or hyper-linking with other media [8]. The RZL (reverse zerorun length) data representation scheme is proposed to exploit the statistics of macroblocks for achieving high embedding efficiency while trading off with payload. We theoretically show that RZL outperforms matrix encoding [22] in terms of payload and embedding efficiency for this particular data hiding method. The problem of video bitstream size increment as a result of data embedding is also addressed, and two independent solutions are presented to suppress this increment. Basic performance of this method is verified through experiments on various existing MPEG-1 encoded videos.

II. Review on Video Compression Standard

A. MPEG-1 and MPEG-2 Compression Standards

In MPEG-1/2 compression standard, a sequence of pictures is segmented into GOPs (group of pictures). Pictures in each GOP are labeled as I, P, or B, depending on the order in which they appear, and the interval between two consecutive P-pictures are determined by the M-factor parameter [23], [24]. Regardless of the picture type, each picture is divided into slices, and each slice is further divided into MBs (macroblocks). Each MB could be coded independently (INTRA mode), or motion-compensated (INTER mode) where motion vectors and differential signals are selectively coded. Finally, each MB contains blocks of 8×8 qDCTCs for luminance and chrominance information. During video playback, DCT coefficients are reconstructed from the qDCTCs using (1) (shown at the bottom of this page).

For a specified video bitrate, MPEG utilizes Mquant (hereinafter referred to as MQ) for distributing the available bits to code the pictures based on their spatial activities and similarity among index and reference frames. Each divisor in the default quantization table is scaled by MQ before the quantization operation is carried out. MQ assumes any integer in the range $[1, 31]$, and its value is recorded in the header of each slices. This value is utilized by all MBs in the slice for encoding/decoding, but each MB can have its own MQ value. The acquirement of a new MQ value is indicated by the CODED_MQ flag in the header of each MB, and this newly coded MQ value is utilized by all tailing MBs until a new value is coded.

B. H.261, H263, and MPEG-4 Compression Standards

H.261, H.263, and MPEG-4 are similar to MPEG-1/2 in the sense that they are all cosine transform-based compression standards and consist of many common entities such as MV, Mquant, qDCTCs, etc. On the other hand, these standards also differ in many aspects, such as MV of size 8×8 pixels is only available in MPEG-4, 3-D VLC in H.263 and MPEG-4, color schemes of 4:2:2 and 4:4:4 in MPEG-2, etc. Here, we present only the significant difference in these compression standards that is relevant to our data hiding method. In particular, during video playback, instead of using (1) to reconstruct the coefficients, (2) (shown at the bottom of this page) is utilized. The rest of the similarities and differences among MPEG-1/2/4 and H.261/3 can be found in [24], [25].

III. Methodology

We first present the basic idea using I-picture of MPEG-1 (i.e., all MBs are coded in INTRA mode, and no MB is skipped or purely motion-compensated), and extend our idea to handle INTER-MB that may occur in P and B-pictures. Modifications required to process MBs in MPEG-4 and H.261/3 are later justified.

A. INTRA-MB: Excitement and Promotion

For any video decoder compliant to MPEG compression standard, the DCT coefficients (i.e., only ac components from

$$\mathrm{rec}[m][n] = \begin{cases} 2 \times \mathrm{qDCTC}[m][n] & \times \ \mathcal{Q}(k) \ \times \ \mathrm{QT}_{\mathrm{INTRA}}[m][n]/16, & \text{if INTRA} \\ (2 \times \mathrm{qDCTC}[m][n] + \mathrm{sign}(\mathrm{qDCTC}[m][n])) & \times \ \mathcal{Q}(k) \ \times \ \mathrm{QT}_{\mathrm{INTER}}[m][n]/16, & \text{otherwise} \end{cases} \tag{1}$$

$$\mathrm{rec}[m][n] = \begin{cases} 0, & \text{if } \mathrm{qDCTC}[m][n] = 0, \\ \mathrm{sign}(\mathrm{qDCTC[m][n]}) \times (2\mathcal{Q}(k) \times |\mathrm{qDCTC}[m][n]| + \mathcal{Q}(k)), & \text{if } \mathrm{qDCTC}[m][n] \neq 0, \text{ and } \mathcal{Q}(k) \text{ is odd} \\ \mathrm{sign(qDCTC[m][n])} \times (2\mathcal{Q}(k) \times |\mathrm{qDCTC}[m][n]| + \mathcal{Q}(k) - 1), & \text{if } \mathrm{qDCTC}[m][n] \neq 0, \text{ and } \mathcal{Q}(k) \text{ is even} \end{cases} \tag{2}$$

Algorithm 1: *Exciting* an INTRA macroblock

1: $\mathcal{Q}(j_0) \leftarrow \alpha$
2: **for** Y, Cb, and Cr channel **do**
3: **for** all nonzero qDCTC$[m][n]$ **do**
4: qDCTC$[m][n] \leftarrow \beta \times$ qDCTC$[m][n]$
5: **end for**
6: **end for**

Algorithm 2: *Promoting* a macroblock

1: $\mathcal{Q}(j_0 + 1) \leftarrow \alpha \times \beta$
2: CODED_MQ flag $\leftarrow$ TRUE

INTRA-MB, and both dc and ac components from INTER-MB) are reconstructed by using (1), where $1 \leq m, n \leq 8$, and $\mathrm{QT}_{\mathrm{INTRA}}$ and $\mathrm{QT}_{\mathrm{INTER}}$ are the default quantization table for INTRA-MB and INTER-MB, respectively. However, dc components of INTRA-MB are reconstructed by the multiplication of a constant value (i.e., 8, and is irrelevant to the MQ value). To ease the discussion, let MB(j) denote the jth MB in a slice. Also, let $\mathcal{Q}(j)$ associate the CODED_MQ flag and the MQ value of MB(j) in the following manner:

$$\begin{aligned} &\mathcal{Q}(j) = 0 \leftrightarrow \text{CODED\_MQ} = \text{FALSE};\\ &\mathcal{Q}(j) > 0 \leftrightarrow \text{CODED\_MQ} = \text{TRUE, where the}\\ &\qquad\qquad\qquad\qquad \text{coded MQ value is as specified.} \end{aligned} \tag{3}$$

Consider MB(j_0) such that $\mathcal{Q}(j_0) = \alpha \times \beta$ for some $\alpha, \beta \in \mathbb{N}$ and $\beta \neq 1$. Here, $\mathbb{N}$ denotes the set of natural numbers. Obviously, α and β are the factors of $\mathcal{Q}(j_0)$. If $\mathcal{Q}(j_0)$ is a prime, then the factors are unity and $\mathcal{Q}(j_0)$ itself. We *excite* MB(j_0) using Algorithm 1. Here, α is chosen as the new $\mathcal{Q}(j_0)$ value and β as the multiplying factor for all nonzero ac qDCTCs residing in MB(j_0). However, their roles could be interchanged as long as none of them is of value unity.

Referring to the INTRA case of (1), the value of the reconstructed ac coefficient $rec[m][n]$ is exactly the same before (original) and after MB *excitement* (modified). In particular, the factor β that is "divided out" from the original $\mathcal{Q}(j_0)$ is compensated by the multiplication of β to all nonzero ac qDCTCs in MB(j_0) as performed in line 4 of Algorithm 1. Therefore, the image quality of MB(j_0) is completely preserved even after MB *excitement*.

As mentioned in Section II, the last coded MQ value is utilized by the remaining MBs in the slice unless a new MQ value is coded. Since $\mathcal{Q}(j_0)$ is modified when MB(j_0) is *excited*, MB($j_0 + 1$) must be modified accordingly to achieve complete image quality-preservation in the slice level. Note that modification is not required for MB($j_0 + 1$) if:

1) MB(j_0) is the last MB in the slice, i.e., $j_0 = N$, where N is the number of MBs in a slice; or
2) $\mathcal{Q}(j_0 + 1) \neq 0$ in which case this value can never be $\alpha \times \beta$.

Otherwise MB(j_0+1) is *promoted* using Algorithm 2 to avoid the use of $Q(j_0) = \alpha$ as the MQ value. Thus, the operation of MB *excitement*, followed by MB *promotion* when necessary, have no effect on the image quality of a slice of MBs, and hence a frame in a video. Note that, although the image quality of the video is completely preserved, it is achieved at the expense of an increase in the video bitstream size, which is further discussed in Section V.

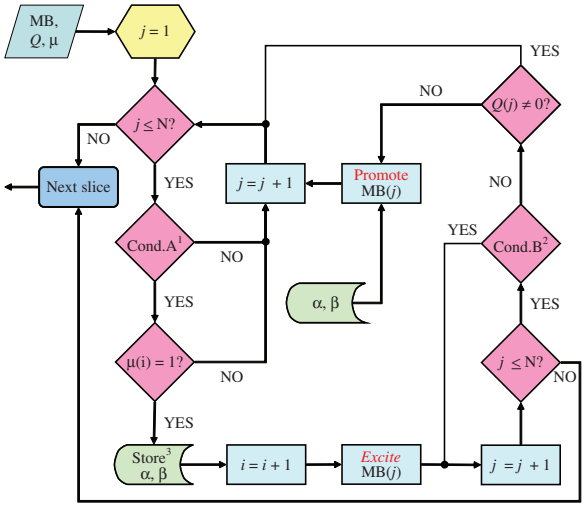

Fig. 1. Flowchart for data embedding. [1]Cond.A := Is $\mathcal{Q}(j) > 0$ and $\mathrm{mod}(\mathcal{Q}(j), \beta) = 0$ for at least one $\beta \in \S_1$?, [2]Cond.B := Is MB(j) completely motion-compensated without any qDCTC or skipped?, [3]This symbol denotes storage.

B. Complete Video Quality-Preserving Data Hiding

To embed information into a MPEG-1 compressed video utilizing MBs with MQ value of $\alpha \times \beta$, we propose an ordinary representation scheme (ORS), which is defined as follows:

1) If the ith message bit $\mu(i) =$ "1," *excite* the MB, and *promote* the affected MB when necessary, OR
2) If $\mu(i) =$ "0," do nothing to the MB.

The embedding flow is summarized in Fig. 1. Instead of a fixed MQ value, we consider $\beta \in \S_1$ and the corresponding α's such that $\mathcal{Q} = \alpha \times \beta$ to increase the payload where

$$\S_1 := \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31\}. \tag{4}$$

Thus, any MQ value in the range [2, 31] could be factored into α and β for at least one $\beta \in \S_1$. Note that each $\beta \in \S_1$ could also be utilized independently to realize multiple messages embedding [26]. In particular, a maximum of 11 unique messages could be embedded into a video using INTRA-MBs in I-pictures while completely preserving the original image quality.

C. Data Extraction and Reversibility

To extract the embedded data, the modified video is parsed in the exact same order as MPEG decoder does. The flowchart for data extraction is summarized in Fig. 2. When an MB with $\mathcal{Q} \neq 0$ is encountered, there are three interpretations:

1) *excited* MB that holds the information bit "1"; or
2) original MB that holds the information bit "0"; or
3) original MB that holds nothing.

To resolve this ambiguity, we consider the condition

$$\mathrm{mod}(x, \beta) = 0 \tag{5}$$

TABLE I
DISTINGUISHING PROMOTED MB AND UNMODIFIED MB

MB(j_0)	MB(j_0+1)	Interpretation
α	$\neq \alpha \times \beta$	MB(j_0) encodes "1," and MB(j_0+1) may encode "0," "1" or nothing.
α	$\alpha \times \beta$	MB(j_0) encodes "1," and MB(j_0+1) is a *promoted* MB, or it encodes "1." It can never encode "0" because $\mathcal{Q}(j_0)$ and $\mathcal{Q}(j_0+1)$ can never be the same in the original video.
$\alpha \times \beta$	$\neq \alpha \times \beta$	MB(j_0) encodes "0," and MB(j_0+1) may encode "0," "1" or nothing.
$\alpha \times \beta$	$\alpha \times \beta$	MB(j_0) encodes "0," and MB(j_0+1) must encode "1" because $\mathcal{Q}(j_0)$ and $\mathcal{Q}(j_0+1)$ can never be the same in the original video.

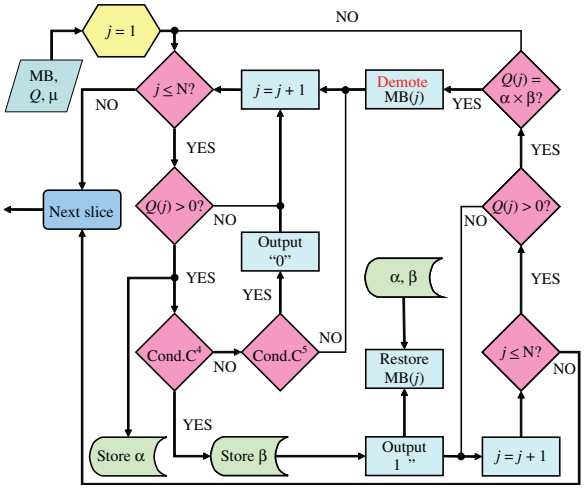

Fig. 2. Flowchart for data extraction and restoration. [4]Cond.C : = Is mod$(x, \beta) = 0$ for all qDCTCs $x \in$ MB(j) for at least one $\beta \in \S_1$?, [5]Cond.D : = Is mod$(\mathcal{Q}(j), \beta) = 0$ for at least one $\beta \in \S_1$?

for each nonzero qDCTC $x \in$ MB. Case 1) occurs if condition (5) is true for a specific $\beta \in \S_1$ for all $x \in$ MB. We store the smallest value of $\beta \in \S_1$ that satisfies condition (5), update $\mathcal{Q} \leftarrow \mathcal{Q} \times \beta$, and yank a "1." On the other hand, case 2) occurs if condition (5) fails but mod$(\mathcal{Q}, \beta) = 0$ for at least one $\beta \in \S_1$ (i.e., Cond.D holds true). In this case, "0" is yanked as the output. Case 3) occurs if Cond.D fails, i.e., when the following holds true, and nothing is output

$$\mathcal{Q} = 1. \tag{6}$$

Condition (5), or more generally Cond.C, is valid for identifying *excited* MBs because it is very unlikely that all nonzero qDCTCs in an MB (from the original video) are divisible by a specific factor $\beta \in \S_1$, and this claim is verified through extensive experiments. Nevertheless, when such a case occurs, we can restrict data embedding to a set of MQ values. Suppose that there is at least one MB with $\mathcal{Q} = \alpha$ in the original video and all of its nonzero ac qDCTCs are divisible by β. We make sure that no other MQ values get mapped to α during data embedding, i.e., MB(j)'s with $\mathcal{Q}(j) = 2 \times \alpha, 3 \times \alpha, \ldots$ are ignored for data embedding. However, MB(j)'s with $\mathcal{Q}(j) = \alpha$ themselves could be utilized for data embedding since the value α will be replaced by $\hat{\alpha}$ ($\alpha = \hat{\alpha} \times \hat{\beta}$, and $\hat{\alpha}, \hat{\beta} \leq \alpha$) if they happen to be *excited* during data embedding. When MB(j) with $\mathcal{Q}(j) = \alpha$ is encountered during data extraction, we know that it must hold the information "0" since nothing gets mapped to it.

Algorithm 3: Restoring an *excited* INTRA macroblock

```
1: Q ← α × β
2: for Y, Cb, and Cr channel do
3:   for all nonzero qDCTC[m][n] do
4:     qDCTC[m][n] ← qDCTC[m][n] / β
5:   end for
6: end for
```

Algorithm 4: *Demoting* a *promoted* macroblock

```
1: Q(j0 + 1) ← 0
2: CODED_MQ flag ← FALSE
```

After encountering an *excited* MB, the next MB with a coded MQ value might be a *promoted* MB. If its MQ value is $\alpha \times \beta$, then it is either a *promoted* MB or an *excited* MB, depending on Cond.C. Otherwise, it is an MB that may hold "0," "1," or nothing, and hence condition Cond.C and Cond.D are considered again to determine its state. Note that a *promoted* MB never occurs by itself without a preceding *excited* MB. Therefore, we only resolve the ambiguity of reaching a *promoted* MB or an MB encoding "0" when its previous MB is *excited*. Suppose MB(j_0) is identified as holding "1," i.e., MB(j_0) is *excited* with $\mathcal{Q}(j_0) = \alpha$, and the information held by MB(j_0+1) is to be determined. We list the interpretations of MB(j_0+1) based on $\mathcal{Q}(j_0+1)$ in Table I. For completeness of discussion, we also list the cases where MB(j_0) encodes "0," i.e., $\mathcal{Q}(j_0) = \alpha \times \beta$.

With the procedures presented above, the *excited* and *promoted* MBs could be identified. Hence, we could restore an *excited* INTRA-MB to its original state by using Algorithm 3. Similarly, a *promoted* INTRA-MB could be restored to its original state by using Algorithm 4.

D. Example: Encoding and Decoding Information

An example is shown in Fig. 3 using two arrays of MBs (original at the top, modified at the bottom) together with their coded MQ values extracted from an I-picture. Here, the value of $\mathcal{Q}(j)$ is recorded in the array if $\mathcal{Q}(j) > 0$, or "X" is marked otherwise. Suppose the message $\mu = 1010\ldots$. MB(j_0) is *excited* to hold the first bit of μ (i.e., "1"), thus $\mathcal{Q}(j_0)$ is updated to $\alpha = 3$ and each nonzero ac qDCTC is scaled by the factor $\beta = 2$ as shown. Then, we check if MB(j_0+1) needs to be *promoted*. Since $\mathcal{Q}(j_0+1) = 10 \neq 0$, MB$(j_0+1)$ is not *promoted*. Instead, MB(j_0+1) is considered to encode the second bit of μ. MB(j_0+1) is left as it is because a "0" is to be embedded. Note that $\mathcal{Q}(j_0+2) = 0$, but MB(j_0+2) is not

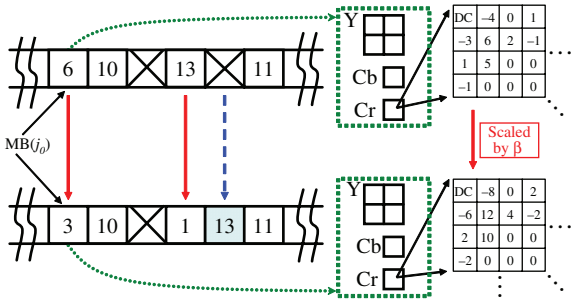

Fig. 3. Example: encoding and decoding information.

promoted since MB($j_0 + 1$) is not *excited*. Next, MB($j_0 + 3$) is *excited* to encode the third bit of μ (i.e., "1"). MB($j_0 + 4$) is *promoted* so that $\mathcal{Q}(j_0+4) = 13$. MB(j_0+5) is left as it is to encode the fourth bit of μ (i.e., "0"). The same process is repeated to embed the entire message.

To extract the embedded information, MBs are visited in the exact same order as in the encoding phase. DEC (message decoder) extracts "1" from MB(j_0) because it satisfies Cond.C with $\beta = 2$. Since $\mathcal{Q}(j_0 + 1) > 0$ and $\mathcal{Q}(j_0 + 1) \neq \mathcal{Q}(j_0) \times 2$, DEC declares that MB($j_0 + 1$) is not a *promoted* MB. DEC extracts "0" from MB($j_0 + 1$) because it fails Cond.C but satisfies Cond.D with $\beta = 2$. DEC does not check MB(j_0+2) for possible *promotion* performed because MB(j_0+1) was not *excited*. DEC extracts "1" from MB($j_0 + 3$) since it satisfies Cond.C with $\beta = 13$. DEC declares MB(j_0+4) as a *promoted* MB because $\mathcal{Q}(j_0 + 4) = \mathcal{Q}(j_0 + 3) \times 13$ and MB($j_0 + 4$) fails Cond.C. DEC further extracts "0" from MB($j_0 + 5$) because it fails Cond.C but satisfies Cond.D with $\beta = 11$. The decoding process continues until the entire message is extracted.

E. Handling INTER-MB

When dealing with INTER-MB which could occur in P and B-pictures, line 4 in Algorithm 1 is modified as follows during MB *excitement*:

$$x \leftarrow \beta x + \text{sign}(x) \times y \tag{7}$$

where x is a nonzero qDCTC in the MB (dc components are also included), and $y = (\beta - 1)/2$. The scaling equation given by (7) is derived in Appendix A. Since all operations are carried out in integer mode in MPEG coding standard, y must be an integer. Hence, the multiplying factor β has to be an odd number, and 2 must be removed from $\S_1$ when processing an INTER-MB. In particular, we use $\beta \in \S_2$ when dealing with INTER-MBs where

$$\S_2 := \S_1 \setminus \{2\} = \{3, 5, 7, 11, 13, 17, 19, 23, 29, 31\}. \tag{8}$$

Next, an INTER-MB is *promoted* using the same operations as applied to *promote* an INTRA-MB, i.e., Algorithm 2. However, since an INTER-MB may be skipped or completely motion-compensated, these MBs are ignored during the search for the next MB coded with qDCTCs (i.e., differential signal) for necessary MB *promotion*. Refer to [24] for more information on MB type.

When extracting the embedded information from an INTER-MB, (5) is replaced by the following condition:

$$\text{mod}(x - \text{sign}(x) \times y, \beta) = 0. \tag{9}$$

Cond.C is also updated accordingly using $\S_2$ in place of $\S_1$. An INTER-MB is declared as encoding nothing if it fails Cond.D, and this happens when

$$\mathcal{Q} \in \{1, 2, 4, 8, 16\}. \tag{10}$$

Algorithm 3 is utilized to restore an *excited* INTER-MB, except that the division operation in line 4 is replaced by

$$x \leftarrow (x - \text{sign}(x) \times y)/\beta \tag{11}$$

for all nonzero qDCTC $x \in$ MB. Finally, a *promoted* INTER-MB is restored to its original state by using Algorithm 4.

F. Application to H.261, H.263, and MPEG-4

Since H.261 and H.263 reconstruct the DCT coefficients of an INTRA-MB using a linear equation similar to (1), Algorithm 1 could be applied directly when *exciting* an INTRA-MB. Interestingly, similar to the case of MPEG-1/2, line 4 in Algorithm 1 is also deduced to be replaced by (7) when dealing with an INTER-MB. Note that the reconstruction equation given by (2) differs depending on the value of MQ (i.e., odd or even), but (7) handles both cases simultaneously. This claim is justified in Appendix B.

In case of MPEG-4, since the dc component of an INTRA-MB is reconstructed using MQ with a nonlinear equation, the proposed method is restricted to INTER-MBs in MPEG-4. Similar to MPEG-1/2 and H.261/3, INTER-MB of MPEG-4 compressed video is *excited* using Algorithm 1 and (7).

By *exciting* MB and *promoting* the affected MB, data embedding could be carried out while completely preserving the image quality of the original video. Procedure for decoding the embedded information, restoring the *excited* MB, and *demoting* the *promoted* MB could be easily derived from the aforementioned discussions and we omit the presentation here.

IV. Reverse Zerorun Length

In this section, we propose RZL data representation scheme for achieving high embedding efficiency. For the rest of the paper, embedding efficiency η refers to *the number of bits embedded per modification*. When embedding at a rate lower than the maximum carrier capacity (i.e., payload), more than k locations could be utilized to encode k bits of information while attempting to reduce the number of modifications. This idea is realized by exploiting the statistics of MB with respect to the proposed data hiding method.

A. Method

Suppose we treat each MB in its original state as "0" and the *excited* state as "1." The original/natural message induced by any video is thus an array of zeros. We exploit this statistic to encode a binary message μ while aiming to reduce the number of MB *excitements*.

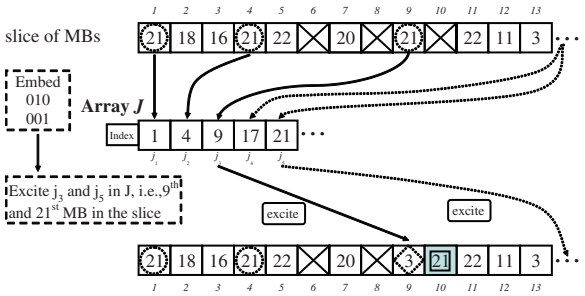

Fig. 4. Example of modification of macroblocks using RZL.

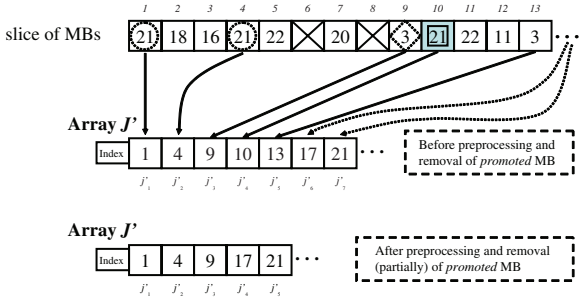

Fig. 5. Decoding example using RZL.

To ease the discussion, consider all indices j such that $\mathcal{Q}(j) = \alpha \times \beta$, and store them in the array J in the order in which they appear, i.e., $J = \{j_1, j_2, \ldots\}$. Unless specified otherwise, $|X|$ denotes the cardinality of the set/array X. Let μ be the binary message of length $|\mu|$, and let $k \in \mathbb{N}, k \geq 2$. μ is divided into segments of length k bits, and each segment $\mu(i)$ is processed one at a time for $i = 1, 2, \ldots, \lceil |\mu|/k \rceil$, where $\lceil x \rceil$ is the smallest integer greater than or equal to x. $\mu(i)$ is embedded by *exciting* the MB which is $\mu(i) + 1$ (in decimal) units away from the previously *excited* MB. In particular, the decimal equivalent value of $\mu(i)$ is computed, and unity is added to this value, i.e., $\mu(i)_{10}+1$. To embed the first message segment $\mu(1)$, we *excite* the $j(i = 1) = (\mu(1)_{10} + 1)$th MB in J. For $i \geq 2$, $\mu(i)$ is embedded by *exciting* the $j(i)$th MB in J where $j(i)$ is computed as follows:

$$j(i) = j(i-1) + \mu(i)_{10} + 1. \tag{12}$$

Note that $j(i) \neq j_i$ for almost all i. To further illustrate the modification on MBs using RZL, Fig. 4 shows an example with $\alpha = 7, \beta = 3, k = 3, \mu(1) = 010$, and $\mu(2) = 001$. Since $j(1) = 2+1 = 3$ and $j(2) = 3+1+1 = 5$, the third and fifth MBs in J, or equivalently, $\text{MB}(j_3 = 9)$ and $\text{MB}(j_5 = 21)$, are *excited* as shown. Also, MB(10) is *promoted* to preserve the image quality. For completeness of discussion, when $k = 1$, we utilize ORS where an *excited* MB represents a "1" and vice versa.

To decode the embedded message, we need to reconstruct the array J' from the modified video. J' contains MBs with $\mathcal{Q} = \alpha \times \beta$ and MBs with $\mathcal{Q} = \alpha$ which includes both the *excited* MBs and the MBs that originally assume the value of α. The indices associated with MBs that originally assume $\mathcal{Q} = \alpha$ are removed from J' by considering Cond.C. Note that still $|J'| \geq |J|$ because J' includes the *promoted* MBs; hence extra operations are carried out to discard them while decoding the embedded message. To decode the first message segment $\mu'(1)$, the index of the first MB (accessed by the indices in J') with $\mathcal{Q} = \alpha$, i.e., $j'(1)$, is searched and unity is subtracted from it. The resulting value is converted into binary representation of k digits, with zeros stuffing as the prefix when necessary. For $i \geq 2$, we need to check if the $[j'(i-1)+1]$th MB in J' is a *promoted* MB. If it is so, the $[j'(i-1)+1]$th index in J' is discarded. The index of the next *excited* MB $j'(i)$ is searched, and $\mu(i)$ is obtained by converting the value $j'(i) - j'(i-1) - 1$ into k digits binary representation. The entire message is obtained by concatenating all $\mu(i)$'s.

Fig. 5 shows the array of indices J' associated with the encoding example shown in Fig. 4. $\text{MB}(j'_3)$ is found to be the first *excited* MB, hence $\mu(1) = 3 - 1 = 2 = 010$ since $k = 3$. j'_4 is discarded from J' since $\text{MB}(j'_4 = 10)$ is identified as a *promoted* MB, and j'_5 is also discarded (by preprocessing) since it is not an *excited* MB. Proceeding in this manner, J' is updated while decoding the message segments, and $J' = J$ at the end of the decoding process as shown in Fig. 5.

To restore the modified video to its original form, $\text{MB}(j'(i))$ is restored to its original state for all i, and the *promoted* MBs, if any, are *demoted* as discussed in Section III-C. Therefore, the proposed data hiding method is still completely reversible even when the new data representation scheme RZL is used.

B. Performance Analysis and Discussion

The performance of RZL is compared to that of matrix encoding (ME) [22]. ME(k) is a data representation scheme that encodes k bits of information by utilizing $2^k - 1$ locations with at most one modification. In the context of the proposed data hiding method, location refers to MB with $\mathcal{Q} = \alpha \times \beta$ for $\beta \in \S_1$ or $\beta \in \S_2$ (depending on the MB type), and modification refers to MB *excitement*. ME is widely utilized because its payload could be traded for higher embedding efficiency. However, in order to store k bits of information, ME constantly requires $2^k - 1$ locations. On the other hand, the number of locations required by RZL varies depending on the message segment to be embedded. In particular, RZL occupies one location in the best case scenario (i.e., $\mu(i) = 000\ldots0$), and occupies 2^k locations in the worst case scenario (i.e., $\mu(i) = 111\ldots1$).

Since we may assume that the message μ is randomly distributed, i.e., $P(0) = P(1) = 0.5$, the expected number of locations required for encoding a k bit message segment in case of RZL is

$$\frac{2^k \cdot (2^k + 1)}{2} \times \frac{1}{2^k} = \frac{2^k + 1}{2} = 2^{k-1} + 0.5 \tag{13}$$

using the fact that $\sum_{i=1}^{n} i = n(n+1)/2$ for $n \in \mathbb{N}$. For embedding $\lceil |\mu|/k \rceil$ segments, it is difficult to formulate the estimated number of locations required. Nevertheless, $\lceil |\mu|/k \rceil \times (2^{(k-1)} + 0.5)$ gives a coarse estimation for the number of locations required to encode μ with segment size k when using RZL. Table II records the expected number of locations required to encode a k bit message segment for $k = 1, 2, \ldots, 9$. For comparison purposes, the number of

TABLE II
EXPECTED NUMBER OF LOCATIONS REQUIRED

k	1	2	3	4	5	6	7	8	9
ME	1	3	7	15	31	63	127	255	511
RZL	1	2.5	4.5	8.5	16.5	32.5	64.5	128.5	256.5

TABLE III
EXPECTED PAYLOAD FOR ME AND RZL FOR VARIOUS k [$\times 10^{-4}$ BITS]

k	1	2	3	4	5	6	7	8
ME	1.000	0.667	0.429	0.267	0.161	0.095	0.055	0.031
RZL	1.000	0.800	0.667	0.471	0.303	0.185	0.109	0.062

locations required for ME is also recorded in the same table. Obviously, RZL requires, in general, fewer locations (almost half) when compared to ME for encoding the same amount of information (i.e., k bits).

Next, we analyze the embedding efficiency η of ME and RZL (hereinafter referred as $\eta[\text{ME}(k)]$ and $\eta[\text{RZL}(k)]$, respectively). When ME is applied, an MB is *excited* with the probability of $1 - 1/2^k$, and hence $\eta[\text{ME}(k)] = k/(1 - 1/2^k)$. The subtraction of $1/2^k$ is due to the fact that an array of zeros (message segment) with length k occurs with probability $1/2^k$ in which case no modification is required. In other words, we always start with an array of zeros (be it of length $2^k - 1$ locations), and the only time we need not *excite* any MB is when the message segment to be embedded (k bits) is an array of zeros (which occurs with probability $1/2^k$). In the case of RZL, we always need to *excite* an MB to mark the message, hence $\eta[\text{RZL}(k)] = k$, which is lower than $\eta[\text{ME}(k)]$. However, as recorded in Table II, RZL requires fewer locations to encode the same amount of information when compared to ME, and hence a larger k could be utilized in case of RZL. As example, suppose $|J| = 10\,000$, and we compute the payload of ME and RZL using various values of k. Table III records the expected payload in fraction, i.e., actual payload divided by $|J|$. The results suggest that

$$\Omega[\text{RZL}(k+1)] \geq \Omega[\text{ME}(k)] \tag{14}$$

where $\Omega[X(k)]$ denotes the payload for data representation scheme $X \in \{\text{ME, RZL}\}$ with parameter k. Equation (14) becomes more obvious for $k \geq 2$. Therefore, we may use a larger k for RZL to encode the same amount of information held by ME. Specifically, for a fixed message length $|\mu|$ and a fixed number of usable MBs, we expect $k_{\text{RZL}} \geq k_{\text{ME}}$. For the same reason, we expect higher embedding efficiency in RZL since

$$k/(1 - 1/2^k) \leq k + 1 \quad \forall k \geq 1. \tag{15}$$

The aforementioned expectations are verified in Section VI. Thus, when embedding at any rate lower than the maximum carrier capacity, RZL could be utilized for achieving higher embedding efficiency, which in turn leads to smaller video bitstream size increment during data embedding.

V. SUPPRESSING VIDEO BITSTREAM SIZE INCREMENT

When an MB is *excited* during data embedding, the magnitude of all nonzero ac components (including dc components in case of INTER-MB) are increased by a factor of at least β. Referring to the default VLC table specified in MPEG coding standard [23], it is observed that when the magnitude of a qDCTC increases, the length of its VLC increases even if its zerorun length remains the same. As a result, whenever an MB is *excited* (modified), the video bitstream size always increases. The simplest way to suppress video bitstream size increment is to code a new VLC table and utilize it in place of the original VLC table. However, the original VLC table in a compressed video could itself be a user-defined table or the default VLC table as specified by MPEG coding standard. Therefore, to achieve complete reversibility after data embedding, we must not code a new VLC table. Instead, we propose the following two independent solutions.

1) Only small multiplying factors (e.g., $\beta = 2$ or 3) are utilized so that the increase in code length (in VLC table) for each nonzero qDCTC is kept to the minimum. In particular, all prime factors no larger than $\phi \in \S_1$ are utilized for INTRA-MB, and all prime factors no larger than $\rho \in \S_2$ are utilized for INTER-MB. The restriction using ϕ and ρ reduces the payload since only a subset of β is considered for data embedding.
2) Instead of using all MBs with $\mathcal{Q} = \alpha \times \beta$ ($\beta \in \S_1$ and $\beta \in \S_2$ in case of INTRA and INTER-MB, respectively) for data embedding, only MBs that are made up of at most τ number of nonzero qDCTCs are utilized. This is a natural way to limit the bitstream size increment since fewer nonzero qDCTCs implies fewer multiplications by the factor β.

To ease the discussion, we define

$$\delta(V) = \text{filesize}(V') - \text{filesize}(V). \tag{16}$$

Note that $\delta(V) > 0$ since the modified video V' has larger file size than the original video V. We use "filesize" and "video bitstream size" interchangeably for the rest of the paper. Also, unless specified otherwise, "increase of video bitstream" or $\delta(V_i)$ refers to the change of filesize for the entire video instead of "change of bitstream size per second" or such. The effect of each solution on payload and $\delta(V)$ are verified in Section VI. Last but not least, because higher embedding efficiency implies less modifications, which in turn leads to smaller $\delta(V)$, RZL proposed in Section IV could be utilized to further suppress $\delta(V)$ while trading off with payload.

VI. EXPERIMENTAL RESULTS

Eight test videos, namely, V_1:*Coastguard*, V_2:*Container*, V_3:*Driving*, V_4:*Flower Garden*, V_5:*Foreman*, V_6:*Hall Monitor*, V_7:*Mobile Calendar*, and V_8:*A walk in the square*, are utilized to verify the performance of the proposed method. Each video is encoded by MPEG-1 compression standard using [27] at 1.5 Mb/s with 15 pictures in a GOP and picture type ratio of I:P:B = 1:4:10. Note that we need not evaluate the image quality of the modified video because, when the modified video is fed into an ordinary decoder, it completely reconstructs the original video even compared at the bit-to-bit level. Nevertheless, we verified that each modified video has exactly the same PSNR value as its original counterpart. It is

TABLE IV
DISTRIBUTION OF USABLE MQ VALUES FOR V_1: *Coastguard*

$\mathcal{Q}$	INTRA-I	INTRA-P	INTER-P	INTRA-B	INTER-B
1	0	0	0	0	0
2	24	0	0	0	0
3[6]	282	1	0	0	0
4	601	6	0	0	0
5	763	18	3430	0	2672
6	727	22	0	0	0
7	568	17	2451	0	6503
8	411	15	0	0	0
9	330	9	0	0	0
10	211	5	664	0	3549
11	147	3	558	0	2618
12	116	3	0	0	0
13	105	3	340	0	1396
14	89	1	164	0	1088
15	20	1	46	0	931
16	1	1	0	0	0
17	0	0	7	0	750
18	0	0	0	0	0
19	0	0	0	0	260
20	0	0	0	0	145
21	0	0	0	0	62
22	0	0	0	0	18
23	0	0	0	0	13
24	0	0	0	0	0
25	0	0	0	0	0
26	0	0	0	0	0
27	0	0	0	0	0
28	0	0	0	0	0
29	0	0	0	0	0
30	0	0	0	0	0
31	0	0	0	0	0
Sum	4395	105	7660	0	20005

[6]Note that $\beta = 3$ is removed from $\S_2$ because there are INTER-MBs in some original videos, in which case all of their qDCTCs are of the form $3z + \text{sign}(z)$ for integer z. This causes false detection of *excited* MB. Thus, in our experiments, $\S_2 = \{5, 7, 11, 13, 17, 19, 23, 29, 31\}$ is considered.

verified that the embedded information could be decoded and removed to restore the original video. Also, using Microsoft Windows Media Player (version 6.4.09.1130), it is verified that all modified videos could be played-back properly.

A. Payload

First, we consider the payload offered by each $\mathcal{Q} = \alpha \times \beta$ for $\beta \in \S_1$ and $\beta \in \S_2$ in case of INTRA and INTER-MB[6], respectively. We record the payloads of V_1:*Coastguard* as the representative example in Table IV. As expected, the payload is relatively low if we utilize only a specific pair of α and β for data embedding (e.g., $\alpha = 5$ and $\beta = 3$ that provide merely 20-bits in INTRA-I). However, more than a pair of α and β could be utilized simultaneously to provide higher total payload. Note that the payload offered by $\mathcal{Q} = 1$ is always zero because $1 \notin \S_1$ and $1 \notin \S_2$. Also, in case of INTER-MB, the payload offered by $\mathcal{Q} \in \{2, 4, 8, 16\}$ is always zero since the only factor for these values is 2, which is not in the set $\S_2$. Similar results are observed for other test videos. For the rest of the paper, payload refers to the sum of the payloads offered by each pair of α and β for $\beta \in \S_1$ or $\beta \in \S_2$ depending on the MB type.

Secondly, for each video, the average payload (bits per frame, hereinafter bpf) for I, P, and B-picture is recorded in Table V. When operating at full capacity, i.e., $(\phi, \rho, \tau) = (31, 31, 384)$, the payload of a video reaches its upper bound with respect to the proposed method. Using I-pictures of V_1 as the representative example, we elaborate the results. The value 4395-bits/20 frames = 219.8 bpf implies that, on average, out of 396 MBs in an I-picture, 219.8 MBs are equipped with coded MQ values that are each divisible by at least one $\beta \in \S_1$. Hence, on average, 219.8 bits could be embedded into each I-picture of V_1. The rest of the results are interpreted similarly.

In the full capacity mode, it is observed that the payload is influenced by the variation of spatial complexity (activity) in the video and similarity among frames. Regardless of the picture type, video of high variation in spatial complexity and low similarity among frames (e.g., V_4 and V_7) offers high payload, and vice versa (e.g., V_2 and V_6). On average, approximately 55.9, 24.0, and 23.6% of all MBs are usable for data embedding in I, P, and B-pictures, respectively. Also, for the same parameter setting, it is observed that regardless of the video, I-picture consistently yields the highest payload. This is because all MBs in I-picture are coded, while only selected MBs in P and B-pictures are coded due to motion compensation.

Next, we investigate how the payload is influenced by solution 1), which is proposed in Section V for suppressing file size increase. When ϕ or ρ is reduced from 31, the payload reduces regardless of the picture type or video. Nevertheless, for each video, I-picture still yields the highest payload. When (ϕ, ρ) is reduced from $(31, 31)$ to $(13, 13)$, the reduction in payload is insignificant, i.e., an average drop of $\sim$2.6, $\sim$4.1, and $\sim$7.9% are observed for I, P, and B-pictures, respectively. Similar results are observed for the reduction of (ϕ, ρ) from $(13, 13)$ to $(7, 7)$. When (ϕ, ρ) is further reduced to $(2, 5)$, the average payload of each video drops to approximately half of its original payload regardless of the picture type. With this relatively strict condition, i.e., $(\phi, \rho, \tau) = (2, 5, 384)$, we still achieve an average payload of $\sim$104, $\sim$38, and $\sim$33 bpf for I, P and B-pictures, respectively.

Now, suppose that data embedding is restricted to MBs with at most τ number of nonzero qDCTCs (i.e., solution 2) in Section V). The results for applying this restriction to the MBs extracted with $(\phi, \rho) = (31, 31)$ and $(2, 5)$ are recorded in Table VI for $\tau = 10, 4$, and 1. Payload for each type of picture generally decreases when τ is decreased. The results indicate that the reduction in payload is most severe in case of I-picture, followed by P and B-pictures. As an example, consider the payload for $(\phi, \rho) = (31, 31)$. When τ is reduced from 384 to 10, the payload is $\sim$1, $\sim$14, and $\sim$69% of the original payload (i.e., when $\tau = 384$) of I, P, and B-pictures, respectively. When τ is reduced to 4 and 1, the payload further decreases. Similar results are observed for $(\rho, \tau) = (2, 5)$. Nevertheless, in case of $(\phi, \rho, \tau) = (2, 5, 1)$, B-pictures could still be utilized to embed information at the rate of $\sim$5.4 bpf.

TABLE V
AVERAGE PAYLOAD PER FRAME FOR EACH PICTURE TYPE WITH PARAMETER ($\phi, \rho, \tau = 384$) AND ORS [bpf]

Video	$(\phi,\rho)=(31,31)$			$(\phi,\rho)=(13,13)$			$(\phi,\rho)=(7,7)$			$(\phi,\rho)=(2,5)$			Total frames			Dimension	MB per frame
	I	P	B	I	P	B	I	P	B	I	P	B	I	P	B		
V_1	219.8	97.1	101.0	219.8	97.0	95.9	207.2	85.7	75.5	109.0	52.4	36.9	20	80	198	352 × 288	396
V_2	118.1	20.5	12.8	118.1	20.5	12.8	117.6	20.4	12.8	66.6	15.9	6.8	20	80	198	352 × 288	396
V_3	215.1	103.6	106.5	214.8	103.4	99.6	200.7	88.9	70.3	107.5	54.5	35.9	20	80	198	352 × 240	330
V_4	239.0	144.2	132.2	212.3	118.1	96.8	184.3	84.5	61.9	118.9	44.0	39.3	10	40	98	352 × 240	330
V_5	216.5	64.9	88.5	216.2	64.7	87.4	210.6	61.5	72.6	106.4	37.6	50.1	20	80	198	352 × 288	396
V_6	165.2	29.3	45.3	165.2	29.3	45.3	164.0	29.2	37.1	87.5	14.4	29.2	20	80	198	352 × 288	396
V_7	312.0	190.4	154.3	282.1	163.4	119.6	242.6	118.9	70.7	151.9	55.4	47.3	20	80	198	352 × 288	396
V_8	162.4	49.5	48.0	162.4	49.5	47.5	159.0	46.7	34.5	86.0	28.0	20.2	30	120	298	352 × 240	330

TABLE VI
AVERAGE PAYLOAD INFLUENCED BY τ WITH RESPECT TO ORS [bpf]

Video	$(\phi,\rho)=(31,31)$									$(\phi,\rho)=(2,5)$								
	$\tau=10$			$\tau=4$			$\tau=1$			$\tau=10$			$\tau=4$			$\tau=1$		
	I	P	B	I	P	B	I	P	B	I	P	B	I	P	B	I	P	B
V_1	0.00	7.53	72.25	0.00	2.20	40.49	0.00	0.45	13.37	0.00	4.28	27.66	0.00	1.19	15.71	0.00	0.28	5.29
V_2	1.05	5.18	8.59	0.60	2.46	5.69	0.00	0.51	2.85	1.00	4.01	4.76	0.55	1.85	3.22	0.00	0.38	1.72
V_3	1.85	8.04	57.20	0.30	1.99	32.04	0.10	0.33	10.98	1.10	4.93	19.93	0.25	1.11	11.40	0.10	0.19	3.74
V_4	5.80	5.55	100.58	2.00	2.15	67.42	0.70	0.88	24.67	3.20	1.28	29.53	1.00	0.55	20.26	0.20	0.23	7.50
V_5	2.10	7.48	57.40	0.80	2.48	34.32	0.10	0.59	12.30	1.25	4.53	33.34	0.40	1.54	20.31	0.05	0.36	7.22
V_6	0.00	8.61	32.81	0.00	2.69	21.89	0.00	0.40	8.35	0.00	4.09	20.20	0.00	1.55	11.87	0.00	0.33	3.61
V_7	2.20	33.11	120.64	1.55	9.61	81.99	0.55	1.95	33.24	0.55	9.46	36.63	0.35	3.01	24.74	0.20	0.59	10.00
V_8	0.03	4.08	33.20	0.00	1.18	21.64	0.00	0.25	9.01	0.00	2.56	14.08	0.00	0.86	9.23	0.00	0.21	4.03

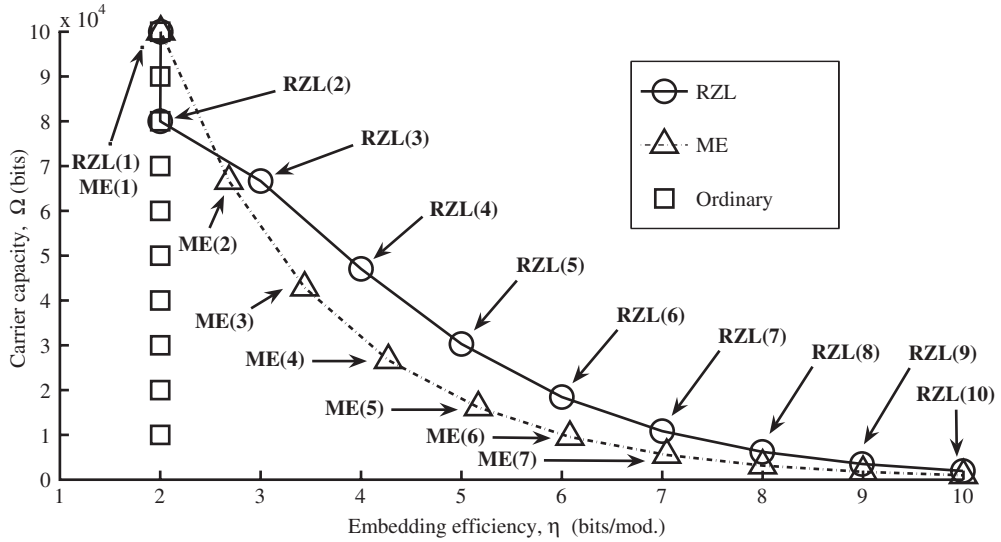

Fig. 6. Embedding efficiency versus payload for ORS, ME, and RZL.

Finally, when RZL is applied to any combination of (ϕ, ρ, τ), we expect the resulting payload to be a fraction of the original payload achieved by (ϕ, ρ, τ). The fraction depends on the parameter k of RZL, and it is recorded in Table III. Nevertheless, when k increases, the payload decreases, and vice versa.

We conclude that the payload (bpf) for each type of picture is influenced by the parameters ϕ, ρ, and τ, as well as the application of RZL. Payload decreases whenever solution 1), 2), or RZL is applied, but their roles in suppressing $\delta(V_i)$ become obvious in Section VI-C.

B. Performance of RZL

In this section, we simulate the embedding process using ME [22] and RZL as the data representation scheme. For comparison purposes, we also consider ORS proposed in Section III-B. As for the experiment parameters, $|J| = 100\,000$, and $k = 1, 2, \ldots, 7$. For each k, a

TABLE VII
INCREASE OF VIDEO BITSTREAM FOR ORS, ME(k), AND RZL($k+1$) [Kbytes]

Ordinary representation scheme						
V_1	266.17	172.43	107.99	66.40	39.08	22.10
V_2	82.04	53.83	33.36	22.24	12.00	6.70
V_3	340.06	224.02	136.78	81.02	50.07	26.65
V_4	246.62	154.82	95.44	60.80	32.60	18.47
V_5	205.22	125.50	77.17	46.46	25.78	15.96
V_6	102.09	64.73	39.41	22.66	13.31	6.94
V_7	441.21	280.47	173.76	104.13	65.43	33.78
V_8	282.98	184.63	118.76	70.28	38.69	23.17

Matrix encoding with parameter k						
Video	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
V_1	203.74	104.05	51.46	24.66	12.81	6.60
V_2	57.65	29.34	15.39	7.37	3.20	2.20
V_3	258.22	129.78	64.47	32.44	16.06	9.08
V_4	180.97	94.27	46.16	24.15	11.56	4.99
V_5	151.58	73.92	37.72	19.00	8.26	4.86
V_6	76.63	40.68	19.54	10.12	4.86	2.48
V_7	343.39	165.56	86.15	42.59	21.37	10.67
V_8	210.62	103.03	53.39	25.81	12.60	5.76

Reverse zerorun length with parameter k						
Video	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$
V_1	179.42	86.81	43.17	21.18	10.60	5.47
V_2	53.44	26.80	13.07	6.37	2.96	1.93
V_3	227.95	109.30	52.12	29.24	13.21	6.95
V_4	160.21	79.36	37.60	19.61	10.18	5.19
V_5	127.76	61.95	30.68	16.28	9.09	4.61
V_6	69.30	34.17	17.35	8.13	3.94	2.24
V_7	300.39	145.76	72.35	35.59	18.35	9.52
V_8	186.70	90.90	45.93	22.57	11.80	6.25

message segment of length k bits is randomly generated and embedded. This process is repeated until all available locations ($|J|$ of them in total) are occupied. Next, the number of message segments embedded (ω) and the number of modifications (ν) are counted. This process is repeated for 1000 times, and the average of ω and ν are computed. The average embedding efficiency is computed as $\bar{\eta} = k \times \bar{\omega}/\bar{\nu}$. The aforementioned procedures are carried out using ORS, ME, and RZL. The results are shown in Fig. 6.

As expected, the embedding efficiency for ORS is always 2 because an MB is *excited* with the probability of 0.5 when embedding a random binary message regardless of its length. Next, the results suggest that when we increase the parameter k from 1 to 2, η[RZL(k)] stays the same, i.e., 2, as indicted by the vertical (solid) line in Fig. 6. This is because every time we embed a message segment, we have to *excite* an MB regardless of its length k. Thus, $\eta[\text{RZL}(2)] = 2/1$.

For each k, it is observed that RZL(k) is inferior to ME(k) in terms of embedding efficiency. Rather, the results should be considered from the payload point of view. For instance, when we consider ME(2) and RZL(3), the carrier capacities are about the same, i.e., $\Omega[\text{ME}(2)] \sim \Omega[\text{RZL}(3)]$, but RZL(3) achieves a higher embedding efficiency when compared to ME(2), i.e., $\eta[\text{RZL}(3)] > \eta[\text{ME}(2)]$.

TABLE VIII
INCREASE OF VIDEO BITSTREAM SIZE WITH RESPECT TO (ϕ, ρ, τ) [Kbytes]

Video	$(\phi,\rho)=(31,31)$			
τ	384	10	4	1
V_1	414.18	47.87	13.62	1.99
V_2	120.94	5.27	1.56	0.35
V_3	522.67	34.89	8.47	1.22
V_4	364.40	20.83	6.59	0.92
V_5	300.97	32.88	9.04	1.40
V_6	158.86	19.92	6.90	1.17
V_7	676.18	67.76	21.17	3.43
V_8	416.32	24.57	7.08	1.24

Video	$(\phi,\rho)=(2,5)$			
τ	384	10	4	1
V_1	166.78	18.33	5.20	0.82
V_2	60.28	3.12	0.88	0.21
V_3	198.74	12.67	3.13	0.46
V_4	98.60	5.98	1.93	0.28
V_5	141.87	18.24	5.41	0.86
V_6	85.22	12.78	3.93	0.50
V_7	179.60	19.90	6.29	1.00
V_8	185.12	10.15	2.94	0.58

TABLE IX
CODING EFFICIENCY FOR VARIOUS (ϕ, ρ, τ) AT 1.5 MB/S ($\times 10^{-3}$)

Video	$(\phi,\rho)=(31,31)$				$(\phi,\rho)=(2,5)$			
τ	384	10	4	1	384	10	4	1
V_1	9.5	38.0	73.5	164.5	10.0	38.7	75.3	160.2
V_2	6.6	49.4	104.6	208.7	8.0	50.3	110.1	210.8
V_3	7.9	42.0	93.9	221.2	8.4	42.0	91.6	203.1
V_4	7.1	59.4	124.3	327.5	8.4	60.8	127.5	327.2
V_5	11.0	44.6	94.6	216.2	13.0	46.8	93.7	207.2
V_6	11.2	44.0	80.5	175.3	12.5	41.3	76.8	178.9
V_7	9.4	47.9	98.2	240.2	11.5	49.2	99.9	249.1
V_8	7.4	51.6	113.7	266.3	7.9	54.2	118.3	260.2

In fact, the embedding efficiency of RZL($k+1$) is always higher than that of ME(k) for $k \geq 2$. Moreover

$$\Omega[\text{ME}(k)] \leq \Omega[\text{RZL}(k+1)] \tag{17}$$

and

$$\eta[\text{ME}(k)] \leq \eta[\text{RZL}(k+1)] \tag{18}$$

hold true simultaneously for $k \geq 3$. Since higher embedding efficiency implies less modifications (i.e., MB *excitements*), $\delta(V_i)$ is expected to be smaller when RZL($k+1$) is utilized as the data representation scheme as oppose to ORS or ME(k).

C. Suppression of Video Bitstream Size Increment

First, we compare the performance of ME [22] and RZL in terms of $\delta(V_i)$ when embedding the same amount of information. Since solution 1) and 2) are independent, it is sufficient to consider a particular set of parameters and infer the results for other settings. In partcular, we consider

TABLE X
PAYLOAD FOR VIDEO BITRATE 1.0, 1.5, 2.0, 2.5, AND 3.0 MB/S [bpf]

Video	1.0 Mb/s			1.5 Mb/s			2.0 Mb/s			2.5 Mb/s			3.0 Mb/s		
	I	P	B	I	P	B	I	P	B	I	P	B	I	P	B
V_1	251.7	127.1	105.5	219.8	97.1	101.0	196.0	65.8	98.8	176.5	41.2	80.5	161.7	27.4	58.2
V_2	142.9	31.9	22.8	118.1	20.5	12.8	95.2	11.9	14.2	74.3	2.1	12.3	55.8	1.5	10.1
V_3	233.6	130.0	106.7	215.1	103.6	106.5	194.8	73.5	102.1	180.5	52.4	83.3	167.8	37.4	66.7
V_4	246.8	172.6	67.7	239.0	144.2	132.2	227.1	130.6	133.9	217.7	107.2	125.6	205.9	85.2	120.7
V_5	241.3	113.5	93.6	216.5	64.9	88.5	195.0	47.4	64.3	173.9	32.8	51.3	152.5	22.4	41.4
V_6	181.9	50.9	77.2	165.2	29.3	45.3	162.7	28.8	31.5	138.7	22.9	29.2	118.9	19.4	31.4
V_7	324.3	235.6	96.0	312.0	190.4	154.3	300.4	166.4	148.3	289.2	141.8	148.3	275.8	109.0	154.9
V_8	237.1	112.0	60.2	162.4	49.5	48.0	184.4	48.8	58.4	125.6	24.3	33.9	107.9	14.6	31.0

TABLE XI
INCREASE OF VIDEO BITSTREAM SIZE FOR VARIOUS BITRATES [Kbytes]

Bitrate [Mb/s]	1.0	1.5	2.0	2.5	3.0
V_1	25.7	37.3	49.9	59.0	70.0
V_2	36.1	45.9	56.4	65.0	64.1
V_3	33.3	48.1	63.4	76.5	90.1
V_4	19.8	27.0	32.7	37.1	40.5
V_5	28.4	36.9	42.6	47.3	51.9
V_6	28.8	35.1	39.3	43.0	43.8
V_7	35.8	44.9	53.4	61.7	73.1
V_8	77.2	86.9	96.2	102.1	110.6

$(\tau, \phi, \rho) = (384, 31, 31)$ which causes the largest $\delta(V_i)$ during data embedding. Since

$$\Omega[\mathrm{ME}(k)] \leq \Omega[\mathrm{RZL}(k+1)] \leq \Omega(\mathrm{ORS}) \tag{19}$$

we embed a message of length $\Omega[\mathrm{ME}(k)]$ bits into each video by using ORS, ME(k), and RZL($k+1$). For fair evaluation purposes, we ensured that each type of picture (i.e., I, P, and B) holds the exact same amount of information when using different representation scheme. The experiment is repeated for $k = 2, 3, \ldots, 8$, and the resulting file sizes are recorded in Table VII. As expected, when either ME or RZL is applied, $\delta(V_i)$ is significantly suppressed. When embedding the same amount of information, RZL($k+1$) consistently yields the smallest $\delta(V_i)$ when compared to ORS and ME(k) regardless of the value k. For example, using the file size increased in case of ORS as the reference, the average reductions of $\delta(V_i)$ are 25 and 34% for ME(2) and RZL(3), respectively. However, when we consider ME(7) and RZL(8), the average reductions of $\delta(V_i)$ are 69 and 72%, respectively. In other words, the superiority of RZL over ME in suppressing $\delta(V_i)$ is more obvious for smaller k and become less obvious as k increases, which agree with the simulation results shown in Fig. 6. We conclude that both ME and RZL are capable in suppressing $\delta(V_i)$, and RZL outperforms ORS and ME when embedding the same amount of information.

Next, we verify that solution 1) and 2) proposed in Section V are also capable in suppressing $\delta(V_i)$ caused by data embedding. For each test video, we consider $\delta(V_i)$ after embedding at the maximum rate with respect to parameter (ϕ, ρ, τ). For simplicity, we utilize ORS as the data representation scheme. The results are recorded in Table VIII. Using V_1 and the parameter setting of $(\phi, \rho) = (31, 31)$ as the representative example, we elaborate the results. When $\tau = 384$, $\delta(V_i) = 414.18$ kbytes, which is the upper bound of $\delta(V_i)$ for the proposed method. Once τ is reduced to 10, $\delta(V_i)$ reduces to 47.87 kbytes, or equivalently $\sim 1/9$ of $\delta(V_i)$ for $\tau = 384$. When τ is further reduced from 10 to 4, $\delta(V_i)$ is suppressed to 13.62 kbytes. A negligible $\delta(V_i)$ of ~ 2 kbytes is observed when $\tau = 1$, but the payload is also significantly reduced (refer to Table VI). The results for other videos and the setting of $(\phi, \rho) = (2, 5)$ are interpreted similarly.

To better visualize the suppression of $\delta(V_i)$, we consider coding efficiency ε that is defined to be the *number of message bits embedded per video bit increased*. Higher ε implies smaller $\delta(V_i)$, and vice versa. The results for ε are recorded in Table IX. On average, 0.0087 bits (from the message) are encoded per increased video bit in case of $(\phi, \rho, \tau) = (31, 31, 384)$. In other words, embedding a single message bit causes an increment of 115 bits in the video bitstream. The rest of the results are interpreted similarly. The results show that ϵ increases when τ is reduced. In the best case scenario, an average increase of four bits in the video bitstream size is observed for every message bit embedded. Interestingly, $\varepsilon(31, 31, \tau) \sim \varepsilon(2, 5, \tau)$ for all τ considered. We conclude that both solutions 1) and 2) are capable of increasing the coding efficiency, and solution 2) has greater impact in suppressing $\delta(V_i)$ than solution 1).

In conclusion, the results suggest that RZL, solutions 1) and 2) are capable in suppressing $\delta(V_i)$ while trading off with payload $\Omega(V_i)$.

D. Influence of Video Bitrate

In this section, we investigate the influence of video bitrate on the performance of our method by re-compressing V_i at 1.0, 2.0, 2.5 and 3.0 Mb/s for $i = 1, 2, \ldots, 8$. Here, we only consider $(\phi, \rho, \tau) = (31, 31, 384)$ using ORS. The payload of V_i compressed at these bitrates are recorded in Table X. The results indicate that, in general, the payload of I and P-pictures decrease as the video bitrate increases. A possible explanation to these observations is that I and P-pictures are finely quantized (with the same MQ value) when the bitrate is high since there are enough (available) bits to code them before reaching the bitrate restriction. In

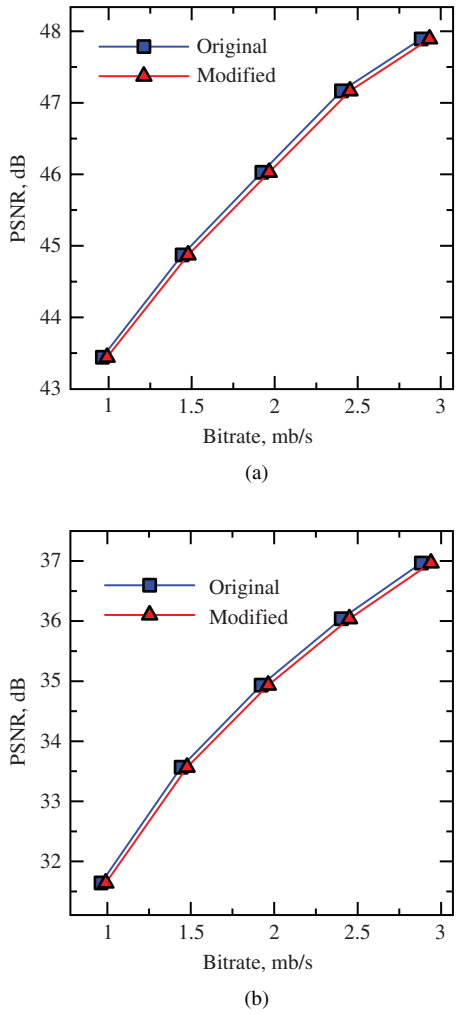

Fig. 7. Graphs of bitrate versus PSNR. (a) V_2: *Container* and (b) V_7: *Mobile calendar*.

other words, the rate controller is frequently in the idle state when coding at a higher bitrate, resulting in less coded MQ values when coding I and P-pictures. The remaining bits are utilized to code the B-pictures, and different MQ values are coded for achieving the specified bitrate. For that, the payload of B-picture shows no obvious trend as the video bitrate varies.

Next, we consider the influence of bitrate on $\delta(V_i)$. Since the payload and video length (i.e., number of frames) vary for each video, embedding at the maximum payload of each video does not lead to a conclusive result. Instead, to fairly evaluate our method, we embed information at a specific rate into each type of picture for all video bitrates. In particular, for each video V_i, we embed 55.8, 1.5, and 10.1 bpf into each I, P, and B-picture, respectively, for $i = 1, 2, \ldots, 8$. These values are selected (i.e., payload of V_2 at 3.0 Mb/s) because these are the smallest payloads for all videos and for all bitrates considered. $\delta(V_i)$ after data embedding is recorded in Table XI in unit of Kbytes. As expected, the results suggest that data embedding always leads to video bitstream size increment in the modified video. $\delta(V_i)$ generally increases as the (compression) bitrate increases because there are more nonzero qDCTCs in video compressed at higher bitrate, and vice versa. Again, we stress that the video bitstream increment $\delta(V_i)$ could be suppressed by tuning the parameters (ϕ, ρ, τ), and/or utilizing ME [22] or RZL as the data representation scheme.

Last but not least, we utilize the results from Table XI and PSNR values to plot the graph of bitrate versus PSNR in Fig. 7 for both the original and modified videos. V_2 and V_7 are shown here as the representative results. Instead of looking at a fixed bitrate and comparing the PSNR values, we should consider a particular PSNR value and compare the bitrates because our method does not distort the image quality of the video during data embedding. For the same PSNR value, the modified video is of higher bitrate than the original video due to MB *excitements* and *promotions*. In other words, to achieve a specific PSNR value, MPEG-1 requires a lower bitrate than our method (i.e., MPEG-1 coupled with the proposed data hiding method). In particular, when embedding 55.8, 1.5, and 10.5 bpf into each I, P, and B-picture, respectively, the video bitrate is increased by 2.3 and 4.2% for V_2 and V_7, respectively, which are very small increments. Similar results are observed for other videos. Thus, we stress again that our data hiding method preserves the image quality of the video at the expense of video bitstream size increment.

VII. Conclusion

A novel data hiding method that completely preserves the image quality of the host video has been proposed in the compressed video domain. During video playback, the modified video completely reconstructs the original video even compared at the bit-to-bit level. This method is reversible, where the original video could be restored from the modified video. RZL data representation scheme has been proposed to improve the embedding efficiency by trading off with payload. It is theoretically and experimentally verified that RZL outperforms matrix encoding in terms of payload and embedding efficiency. Basic performance of this method is evaluated using various MPEG-1 compressed videos. Results suggest that, approximately 55.9, 24.0, and 23.6% of all MBs are usable for data embedding in I, P and B-pictures, respectively, while the video bitstream increases up to 40% when operating at full capacity. The problem of video bitstream size increment can be suppressed by RZL or any of the two independent solutions proposed. In the best case scenario, an average increase of four bits in the video bitstream size is observed for every message bit embedded.

As future work, we seek for possible extensions of our data hiding method to withstand hostile environment so that the embedded message could still be extracted after common image processing attacks. We should explore complete quality preserving data hiding in other domains such as still picture and audio. At the same time, we should succeed in suppressing video bitstream size increment due to data embedding without sacrificing payload. We also seek for the applications of RZL in other data hiding domains.

APPENDIX

A. Scaling Equation for MPEG-1/2

We derive the scaling equation for INTER-MB. Assume that $x > 0$ [thus $\text{sign}(x) = 1$] and assume that the coded Mquant value $\mathcal{Q}$ is updated from $\alpha \times \beta$ to α, which is the same as $\mathcal{Q}/\beta$. Based on (1), we want to update x using $x \leftarrow \beta \times x + y$ and hence, we need to find y such that (20) holds true

$$rec = \frac{[2 \times x + \text{sign}(x)] \times \text{MQ} \times \text{QT}_{\text{INTER}}}{16} = \frac{[2 \times (\beta \times x + y) + \text{sign}(\beta \times x + y)] \times (\text{MQ}/\beta) \times \text{QT}_{\text{INTER}}}{16}. \quad (20)$$

For simplicity, we also assume that $\text{sign}(x) = \text{sign}(\beta \times x + y)$ since there is no restriction on the sign of y. After simplification and trimming of (20), we obtain

$$[2 \times x + \text{sign}(x)] = [2 \times (\beta \times x + y) + \text{sign}(\beta \times x + y)]/\beta.$$

Since $\text{sign}(x) = \text{sign}(\beta \times x + y) = 1$, we have the following:

$$2 \times x + 1 = [2 \times (\beta \times x + y) + 1]/\beta.$$

Simplifying the equation gives us $\beta = 2y + 1$, and thus $y = (\beta - 1)/2$ for $x > 0$. Similarly, we can derive that $y = (1 - \beta)/2$ for $x < 0$.

B. Scaling Equation for H.261, H.263, and MPEG-4

Similar to MPEG-1/2, we assume that the Mquant value is $\mathcal{Q} = \alpha \times \beta$. First, we consider the case when $\mathcal{Q}$ is odd. Referring to (2), we want to find y so that the following equation holds true:

$$\text{sign}(x) \cdot [2\alpha\beta|x| + \alpha\beta] = \text{sign}(\beta x + y) \cdot [2\alpha|\beta x + y| + \alpha]. \quad (21)$$

Again, we have the freedom for setting the sign of y, and we force y to have the same sign as x. Hence, (21) could be simplified to

$$2\beta|x| + \beta = 2|\beta x + y| + 1. \quad (22)$$

Assuming that $x > 0$, we obtain

$$2\beta x + \beta = 2\beta x + 2y + 1. \quad (23)$$

Simplifying (23) leads us to $y = (\beta - 1)/2$ for $x > 0$. Similarly, we can derive that $y = (1 - \beta)/2$ for $x < 0$.

Now suppose $\mathcal{Q}$ is even. We want to find y so that the following equation holds true:

$$\text{sign}(x) \cdot [2\alpha\beta|x| + \alpha\beta - 1] = \text{sign}(\beta x + y) \cdot [2\alpha|\beta x + y| + \alpha - 1]. \quad (24)$$

When we assume that x and $\beta x + y$ are both of the same sign, (24) simplifies to (22), and the aforementioned discussion could be applied directly. Therefore, $y = (\beta - 1)/2$ when $x > 0$ and $y = (1 - \beta)/2$ when $x < 0$ for both odd and even $\mathcal{Q}$.

REFERENCES

[1] S. Katzenbeisser and F. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*. Norwood, MA: Artech House Publishers, 2000.

[2] N. F. Johnson, Z. Duric, and S. Jajodia, *Information Hiding: Steganography and Watermarking Attacks and Countermeasures*. Norwell, MA: Kluwer, 2003.

[3] M. Takayama, K. Tanaka, A. Yoneyama, and Y. Nakajima, "A video scrambling scheme applicable to local region without data expansion," in *Proc. IEEE ICME*, Toronto, ON, Jul. 2006, pp. 1349–1352.

[4] W. Zeng and S. Lei, "Efficient frequency domain video scrambling for content access control," in *Proc. ACM 7th Int. Multimedia Conf.*, Oct. 1999, pp. 285–294.

[5] C. Wong, H. Yu, and M. Zheng, "A DCT-based MPEG-2 transparent scrambling algorithm," *IEEE Trans. Consumer Electron.*, vol. 49, no. 4, pp. 1208–1213, Nov. 2003.

[6] I. Cox, M. L. Miller, and J. A. Bloom, *Digital Watermarking*. San Mateo, CA: Morgan Kaufmann, 2002.

[7] U. Budhia, D. Kundur, and T. Zourntos, "Digital video steganalysis exploiting statistical visibility in the temporal domain," *IEEE Trans. Inform. Forensics Security*, vol. 1, no. 4, pp. 502–516, Dec. 2006.

[8] H. Yanagihara, Y. Nakajima, T. Matsuoka, and K. Tanaka, "Advancement of streaming contents using watermark indexing (Part III) development of software video player with watermark detection," in *Proc. IIEEJ 33rd Annu. Conf.*, Jun. 2005, pp. 77–78.

[9] M. Kurosaki and H. Kiya, "Error concealment using a data hiding technique for MPEG video," *IEICE Trans. Fundamentals Electron., Commun. Comput. Sci.*, vol. E85-A, no. 4, pp. 790–796, Apr. 2002.

[10] H. Kiya, Y. Noguchi, A. Takagi, and H. Kobayashi, "A method of inserting binary data into MPEG video in the compressed domain," *IEICE Trans. Fundamentals Electron., Commun. Comput. Sci.*, vol. 82, no. 8, pp. 1485–1492, 1999.

[11] J. Zhang, J. Li, and L. Zhang, "Video watermark technique in motion vector," in *Proc. IEEE Brazilian Symp. Comput. Graph. Image Process.*, Florianópolis-SC, Brazil, Oct. 2001, pp. 179–182.

[12] Z. Liu, H. Liang, X. Niu, and Y. Yang, "A robust video watermarking in motion vectors," in *Proc. IEEE Int. Conf. Signal Process.*, vol. 3. Sep. 2004, pp. 2358–2361.

[13] C. Xu, X. Ping, and T. Zhang, "Steganography in compressed video stream," in *Proc. IEEE Int. Conf. Innovative Comput., Inform. Control*, vol. 1. Beijing, China, Aug. 2006, pp. 269–272.

[14] Y. Bodo, N. Laurent, and J.-L. Dugelay, "Watermarking video, hierachical embedding in motion vectors," in *Proc. IEEE ICIP*, Sep. 2004, pp. 739–742.

[15] A. Sarkar, U. Madhow, S. Chandrasekaran, and B. S. Manjunath, "Adaptive MPEG-2 video data hiding scheme," in *Proc. SPIE Security, Steganography, Watermarking Multimedia Contents IX*, Jan. 2007, pp. 489–497.

[16] K. Nakajima, K. Tanaka, T. Matsuoka, and Y. Nakajima, "Rewritable data embedding on MPEG coded data domain," in *Proc. IEEE ICME*, Jul. 2005, pp. 682–685.

[17] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 3, pp. 354–362, Mar. 2006.

[18] G. Qiu, P. Marziliano, A. T. Ho, D. He, and Q. Sun, "A hybrid watermarking scheme for H.264/AVC video," in *Proc. IEEE ICPR*, vol. 4. Aug. 2004, pp. 865–868.

[19] S. Pranata, V. Wahadaniah, Y. L. Guan, and H. C. Chua, "Improved bit rate control for real-time MPEG watermarking," *EURASIP J. Appl. Signal Process.*, vol. 2004, no. 14, pp. 2132–2141, Aug. 2004.

[20] F. Hartung and B. Girod, "Digital watermarking of raw and compressed video," in *Proc. SPIE Digital Compression Technologies Syst. Video Commun.*, vol. 2952. Oct. 1996, pp. 205–213.

[21] K. Wong and K. Tanaka, "Mquant-based data hiding method in MPEG domain," in *Proc. IIEEJ Image Electron. Visual Comput. Workshop* [CD-ROM], Nov. 2007.

[22] R. Crandall. (1998). *Some Notes on Steganography* [Online]. Available: http://os.inf.tu-dresden.de/westfeld/crandall.pdf

[23] P. Symes, *Digital Video Compression*. New York: McGraw-Hill, 2004.

[24] L. Hanzo, P. Cherriman, and J. Streit, *Video Compression and Communications*. Piscataway, NJ: IEEE Press, 2007.

[25] "H.261: Video codec for audiovisual services at p × 64 kbit/s," Tech. Rep., 1993. [Online]. Available: http://www.itu.int/rec/T-REC-H.261-199303-I/en

[26] K. Wong, K. Tanaka, and X. Qi, "Multiple messages embedding using DCT-based Mod4 steganographic method," in *Proc. LNCS Int. Workshop Multimedia Content Representation, Classification, Security*, Sep. 2006, pp. 57–65.

[27] "Information technology coding of moving pictures and associated audio for digital storage media at up to about 1, 5 MBIT/S-Part 5: Software simulation," Geneva, Switzerland, Tech. Rep. ISO/IEC TR 11172-5, 1998.

KokSheik Wong (S'06–M'09) received the B.S. and M.S. degrees in computer science and mathematics from Utah State University, Logan, in 2002 and 2005, respectively.

He is currently a doctoral student at the Department of Electrical and Electronics Engineering, Faculty of Engineering, Shinshu University, Japan. His research interests include steganography, digital watermarking, information hiding, and their applications.

Mr. Wong is a student member of IEICE, and IIEEJ.

Kiyoshi Tanaka (M'95) received the B.S and M.S. degrees in electrical engineering and operations research from the National Defense Academy, Yokosuka, Japan, in 1984 and 1989, respectively. In 1992, he received the Doctor of Engineering degree from Keio University, Tokyo, Japan.

In 1995, he joined the Department of Electrical and Electronics Engineering, Faculty of Engineering, Shinshu University, Japan, where he is currently a Professor. His research interests include image and video processing, information hiding, evolutionary computation, chaos and fractals, and their applications.

Prof. Tanaka is a member of IEICE, IPSJ, and IIEEJ. He is the Vice-Chairman of the Editing Committee of IIEEJ.

Koichi Takagi received the M.E. degree from the Tokyo Institute of Technology, Tokyo, Japan, in 1998.

He has been at Kokusai Denshin Denwa, Japan, since 1998 and now is a Research Engineer in the Multimedia Communications Laboratory at KDDI Research and Development Laboratories, Japan. He has been engaged in research on video coding and his current research interest is mobile multimedia processing.

Yasuyuki Nakajima (M'87) received the B.E. degree in electronics and communications engineering and the M.E. degree in electrical engineering from Waseda University, Japan, in 1980 and 1982, respectively. He received the Doctor of Engineering degree from Tokyo Institute of Technology in 2005.

In 1982, he joined KDD Corporation Limited, Japan. From 1985 to 1986, he was a Visiting Researcher at the Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge. In 1987, he joined KDD Research and Development Labs and was engaged in video coding, video communications over IP, compressed domain MPEG processing, content-based indexing, and multimedia database systems for intranet and the Internet. Since 2006, he has been Vice President of KDDI Research and Development Laboratories Incorporation, Japan. Currently, he is with the Content and Media Business Division of KDDI Corporation, where he is General Manager of the Media Service Planning Department.