16

Complete Video Quality Preserving Data Hiding for Multimedia Indexing

KokSheik Wong and Kiyoshi Tanaka
Shinshu University, Nagano
Japan

1. Introduction

Digital video has become a popular multimedia content in both online and offline environments thanks to the advancement of computer and portable devices, as well as broadband internet technologies. 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 (Johnson et al., 2003; Katzenbeisser & Petitcolas, 2000).

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 (Takayama et al., 2006; Wong et al., 2003; Zeng & Lei, 1999); the other that embeds external information into the video, hence utilizing video as the data host. We consider the latter in this chapter. 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 different bitrate, random video frame dropping, resizing, etc. (Cox et al., 2002). In case of steganography, the embedded information should stay undetectable with respect to steganalysis (Budhia et al., 2006), 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 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 at (Katzenbeisser & Petitcolas, 2000; Kurosaki & Kiya, 2002; Yanagihara et al., 2005).

Some representative data hiding methods in video domain could be found at (Bodo et al., 2004; Kiya et al., 1999; Liu et al., 2004; Nakajima et al., 2005; Ni et al., 2006; Qiu et al., 2004; Sarkar et al., 2007; Xu et al., 2006; Zhang et al., 2001). 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 (Kiya et al., 1999). 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 (Nakajima et al., 2005). After zigzag scanning, a dummy nonzero value is inserted at a location that is

z units away from the the original last nonzero qDCTC, where z 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 (Zhang et al., 2001). 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 (Hartung & Girod, 1996; Pranata et al., 2004). For instance, Pranata et al. embed information into a frame by evaluating the combined bit lengths of a set of multiple watermarked VLC (variable length coding) codewords (Pranata et al., 2004). They successively replace the watermarked VLC codewords having the largest increase in 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 (compressed) video even after data embedding (Wong & Tanaka, 2007). Suboptimal histogram preserving modification scheme is also proposed to maintain the distribution of Mquant before and after data embedding.

In this work, 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 applicable to the encoding process of MPEG-1/2/4 or H261/3 video from a sequence of raw pictures. 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 analyze that RZL outperforms matrix encoding (Crandall, 1998) 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 with various existing MPEG-1 encoded videos.

One of the possible applications of our data hiding method is video indexing where high image quality and reversibility are greatly desired because we can provide high quality video as well as additional specialized functions such as searching, playback control, hyper-linking with other media, etc. (Yanagihara et al., 2005). Here, we briefly introduce three practical scenarios using video indexing:

The first scenario is *educational use*. Instead of the traditional straight forward video (i.e., lecture) playback, we can provide interactive learning environment for the viewer. For example, the video could be played back at a specific speed suitable for the viewer, the parts of the video where important or complex ideas are presented could be easily accessed or repeated upon viewer's request, external resources such as presentation slides and related websites could

be hyper-linked to the frame of interest and accessed upon viewer's action, the area(s) in a frame where the viewer should pay attention to could be emphasized by marking or masking, frames in which the object of interest appear could be tagged for retrieval, etc. Note that the aforementioned features could be customized for each viewer based on his/her level of understanding on the material, and the embedded information could be removed when necessary. On top of that, the lecture to be viewed could be searched by using the information such as keyword that is embedded in it.
The second scenario is *servicing and maintainance business*. Instead of the traditional paper-based manual for an equipment, an interactive video manual could be authored using video indexing technology. Video based instruction along with audio description is extraordinarily informative especially when verbal or picture-based manual itself is incapable to provide detailed instructions. Suppose we author an interactive video manual for servicing computer printer. When a technician services the printer, he could watch and follow the interactive video instructions at his own pace. The technician could choose the area of the printer to be serviced (eg. printer cartridge, paper tray, firmware update, etc.) and the desired video will be played back. When there is a need for part replacement, the technician places the mouse cursor on the part to be replaced, and information such as URL to the order page, review and evaluation of the spare part manufactured by each company, current prices in the market, etc. could be displayed and accessed upon further action. This type of video could also be utilized for training new technicians.
The third scenario is *handling video database*. Instead of storing the classification information or unique tag of each video in a separate file, this information could be embedded into the video. By doing so, we could avoid managing two separate files (i.e., the record and the video itself) because the identifying information stays intact with the video and it could be decoded for purpose of indexing. Also, when two or more video databases are merged, ambiguity in video retrieval does not occur in our method. In case of using a pre-defined naming scheme for identifying the videos in place of data hiding technology, the ambiguity problem may occur because two videos, each from a different database, could have the exact same filename. From the ordinary user point of view, one could retrieve videos using query keywords, and browse these candidate videos having the same image qualities as the original unmodified videos. When the desired video is retrieved, the embedded indexing information could be removed (since our method is reversible) or it could be used for other purposes.

2. Review on video compression standard

2.1 MPEG-1 and MPEG-2 compression standards

In MPEG-1/2 compression standard, a sequence of pictures is segmented into GOP's (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 (Hanzo et al., 2007; Symes, 2004). 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 Eq. (1). Here, x denotes the qDCTC, QT_1 and QT_2 are the default quantization table for INTRA and INTER-MB, respectively.

$$\text{rec}[m][n] = \begin{cases} 2 \times x[m][n] & \times \quad \mathcal{Q}(k) \quad \times \quad \text{QT}_1[m][n]/16, & \text{if INTRA} \\ (2 \times x[m][n] + \text{sign}(x[m][n])) & \times \quad \mathcal{Q}(k) \quad \times \quad \text{QT}_2[m][n]/16, & \text{otherwise} \end{cases} \tag{1}$$

For a specified video bitrate, MPEG utilizes Mquant (hereinafter referred 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 of $[1, 31]$, and its value is recorded in the header of each slice. 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.

2.2 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, three dimensional VLC in H.263 and MPEG-4, color schemes of 4:2:2 and 4:4:4 in MPEG-2, etc. Here, we only present the significant difference in these compression standards that is relevant to our data hiding method. In particular, during video playback, instead of using Eq. (1) to reconstruct the coefficients, Eq. (2) is utilized. The rest of the similarities and differences among MPEG-1/2/4 and H.261/3 could be found at (Hanzo et al., 2007; ITU-T H.261, 1990).

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

3. 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 nor 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.

3.1 INTRA-MB: *Excitement* and *promotion*

For any video decoder compliant to MPEG compression standard, the DCT coefficients (i.e., only AC components from INTRA-MB, and both DC and AC components from INTER-MB) are reconstructed by using Eq. (1), where $1 \leq m, n \leq 8$, respectively. However, DC components of INTRA-MB are reconstructed by the multiplication of a constant value (i.e., eight, 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:

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

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

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 Eq. (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 2**, 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:

i. 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
ii. $\mathcal{Q}(j_0+1) \neq 0$ 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 5**.

3.2 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 ORS (ordinary representation scheme) 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 **Figure 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. Here,

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

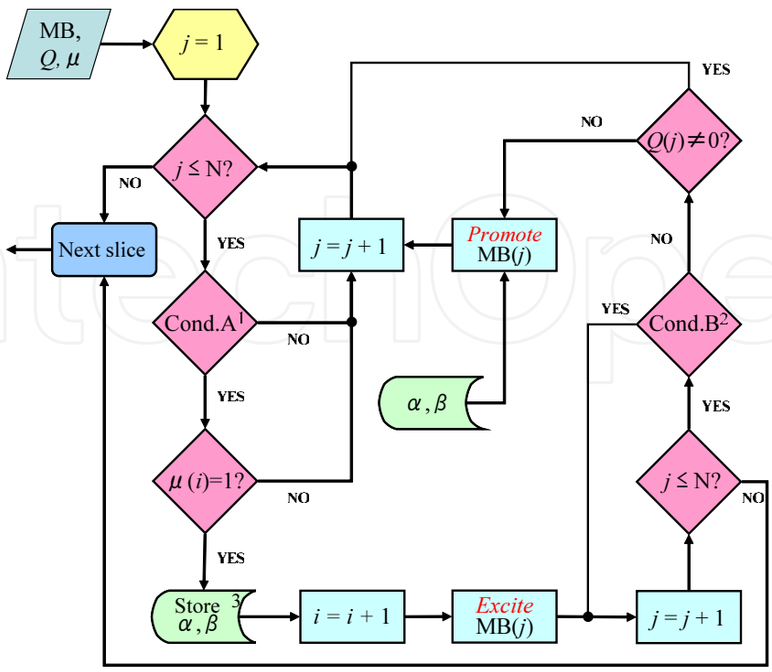


Fig. 1. Flowchart for data embedding. [1] **Cond.A**:= Is $\mathcal{Q}(j) > 0$ and $\text{mod}(\mathcal{Q}(j), \beta) = 0$ for at least one $\beta \in \S_1$?, [2]**Cond.B**:= Is MB(j) completely motion compensated without any qDCTC or skipped?, [3]This symbol denotes storage.

Thus, any MQ value in the range of $[2, 31]$ 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 (Wong et al., 2006). 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.

3.3 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 **Figure 2**. When an MB with $\mathcal{Q} \neq 0$ is encountered, there are three interpretations:

(i) *excited* MB that holds the information bit "1", or

(ii) original MB that holds the information bit "0", or

(iii) original MB that holds nothing.

To resolve this ambiguity, we consider the condition

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

for each nonzero qDCTC $x \in$ MB. Case (i) 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 an "1". On the other hand, case (ii) 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

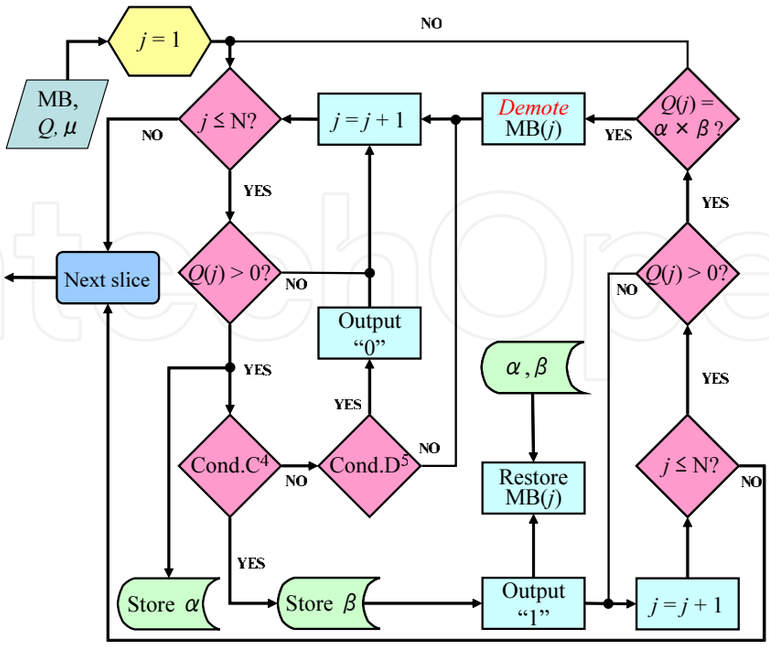


Fig. 2. Flowchart for data extraction and restoration. [4]**Cond.C**:= Is $\mod(x, \beta) = 0$ for all qDCTCs $x \in \text{MB}(j)$ for at least one $\beta \in \S_1$?, [5]**Cond.D**:= Is $\mod(\mathcal{Q}(j), \beta) = 0$ for at least one $\beta \in \S_1$?

the output. Case (iii) 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 $\mathcal{Q}(j) = 2 \times \alpha, 3 \times \alpha, \cdots$ 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 happened 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.

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 $\mathcal{Q}(j_0 + 1)$

$MB(j_0)$	$MB(j_0+1)$	Interpretation
α	$\neq \alpha \times \beta$	$MB(j_0)$ encodes "1", and $MB(j_0+1)$ encodes "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 $MB(j_0+1)$ encodes "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.

Table 1. Distinguishing promoted MB and unmodified MB

Algorithm 3 Restoring an *excited* INTRA macroblock

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

Algorithm 4 *Demoting* a *promoted* macroblock

1: $\mathcal{Q}(j_0+1) \leftarrow 0$
2: CODED_MQ flag $\leftarrow$ FALSE

in **Table 1**. 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**.

3.4 Example: Encoding and decoding information

An example is shown in **Figure 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 $\mu = 1010\cdots$. $MB(j_0)$ 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 an "0" is to be embedded. Note that $\mathcal{Q}(j_0+2) = 0$, but $MB(j_0+2)$ is not *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 forth 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

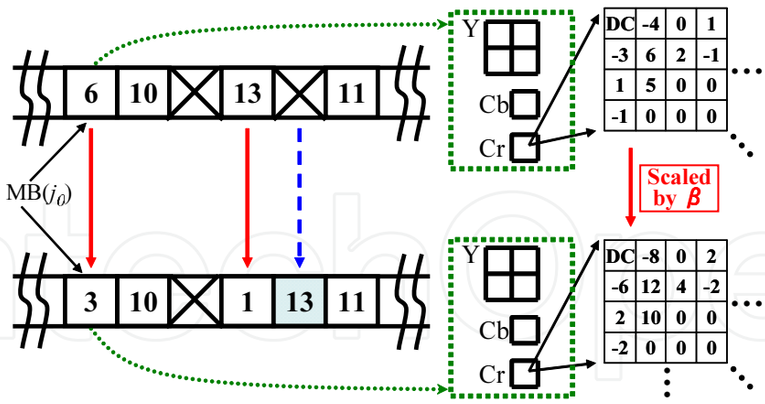


Fig. 3. Example: Encoding and decoding information

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.

3.5 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 Eq. (7) is derived in **Appendix**. 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., image signal in case of INTRA-MB or differential signal in case of INTER-MB) for necessary MB *promotion*. Refer to (Hanzo et al., 2007) for more information on MB type.

When extracting the embedded information from an INTER-MB, condition (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**.

3.6 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 that of Eq. (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 Eq. (7) when dealing with an INTER-MB. Note that the reconstruction equation given by Eq. (2) differs depending on the value of MQ (i.e., odd or even), but Eq. (7) handles both cases simultaneously. This claim is justified in **Appendix**.

In case of MPEG-4, since the DC component of an INTRA-MB is reconstructed using MQ with a non-linear 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 Eq. (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.

4. Reverse zerorun length

In this section, we propose RZL (reverse zerorun length) data representation scheme for achieving high embedding efficiency. For the rest of the chapter, 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.

4.1 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*.

Let μ be a binary message of length $|\mu|$, and let $k \in \mathbb{N}$ such that $k \geq 2$. μ is divided into segments of length k bits, and each segment $\mu(i)$ is processed one at a time for $i = 1, 2, \cdots, \lceil |\mu|/k \rceil$, where $\lceil z \rceil$ is the smallest integer greater than or equal to z. Unless specified otherwise, $|Z|$ denotes the cardinality of the set/array Z. Given $\mu(i)$ in ORS format as the input, the binary-to-RZL convertor (hereinafter referred as BIN2RZL) computes the decimal equivalent of $\mu(i)$ (denoted by $\mu_{10}(i)$), outputs $\mu_{10}(i)$ number of zeros, followed by an "1" to mark the end of the message segment. For example, suppose $\mu = 011101001$ and $k = 3$,

$$\begin{array}{lcll} \text{BIN2RZL}(011) & = & 0001 & \text{(i.e., three zeros followed by unity)} \\ \text{BIN2RZL}(101) & = & 000001 & \text{(i.e., five zeros followed by unity)} \\ \text{BIN2RZL}(001) & = & 01 & \text{(i.e., one zero followed by unity)} \end{array} \tag{12}$$

To embed the message μ using RZL, the output of BIN2RZL$(\mu(i))$ for all i are concatenated. The resulting sequence of zeros and ones are threated as a new message $\hat{\mu}$, and $\hat{\mu}$ is embedded

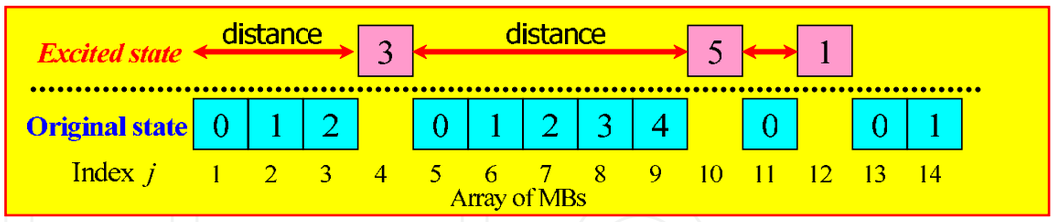


Fig. 4. Encoding and decoding example using RZL. Each MB is assumed to be usable for data embedding.

by *exciting* and *promoting* MBs as described earlier. In other words, RZL utilizes the distance between two consecutive "1"s output by BIN2RZL to encode information. Using the same example, $\hat{\mu} = 000100000101$ and **Figure 4** illustrates the resulting macroblocks after encoding $\hat{\mu}$ with RZL. Note that in this particular example, five MB *excitements* are required to encode $\mu = 011101001$ in case of ORS but only three MB *excitements* are required in case of RZL. Also note that given any binary sequence of length k, the output of BIN2RZL may be longer or shorter than k. Discussions on the number of *excitements* and locations required to encode $\mu(i)$ are presented in Section 4.2. For completion of discussion, RZL switches to ORS when $k = 1$.
To decode the message embedded in RZL format, $\hat{\mu}$ is first decoded by considering the state of each MB as described earlier. To ease the dicussion, let $\hat{\mu}(j)$ denote the jth bit in $\hat{\mu}$. The indices j_i where $\mu(j_i) = 1$ are stored in the array J in the order in which they appear. Note that $|J| \times k = |\mu|$. To decode $\mu(i)$, we set $j_0 = 0$ and consider the RZL-to-binary convertor (hereinafter referred as RZL2BIN) that converts $j_i - j_{i-1} - 1$ into binary representation of k digits, with zeros stuffing as the prefix when necessary. In particular,

$$\text{RZL2BIN}(\text{BIN2RZL}(\mu(i)), k) = \mu(i). \tag{13}$$

Note that the value k is required in RZL2BIN.

4.2 Performance analysis and discussion

The performance of RZL is compared to that of matrix encoding (ME) (Crandall, 1998). 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., $\mu(i) = 000\cdots0$), and occupies 2^k locations in the worst case scenario (i.e., $\mu(i) = 111\cdots1$).
Since we may assume that the message μ is randomly distributed, i.e., $P(0) = P(1) = 0.5$, hence the expected number of locations required for encoding a k bit message segment in case of RZL is

$$\frac{2^k \cdot (2^k + 1)}{2} \times \frac{1}{2^k} = \frac{2^k + 1}{2} = 2^{k-1} + 0.5 \tag{14}$$

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, $\lceil |\mu|/k \rceil \times$

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 2. Expected number of locations required

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

Table 3. Expected payload for ME and RZL for various k [$\times 10^{-4}$ bits]

$(2^{(k-1)} + 0.5)$ gives a coarse estimation for the number of locations required to encode μ with segment size k when using RZL. **Table 2** records the expected number of locations required to encode a k bit message segment for $k = 1, 2, \cdots, 9$. For comparison purposes, the number of locations required for ME is also recorded in the same table. Obviously, RZL requires, in general, less 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 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 2**, RZL requires less 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 an example, suppose $|J| = 10000$, and we compute the payload of ME and RZL using various values of k. **Table 3** records the expected payload in fraction, i.e., actual payload divided by $|J|$. The results suggest that

$$\Omega[\text{RZL}(k+1)] \geq \Omega[\text{ME}(k)], \tag{15}$$

where $\Omega[Z(k)]$ denotes the payload for data representation scheme $Z \in \{\text{ME}, \text{RZL}\}$ with parameter k. Eq. (15) 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

$$k/(1 - 1/2^k) \leq k + 1, \forall k \geq 1. \tag{16}$$

The aforementioned expectations are verified in **Section 6**. 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.

5. 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 (Symes, 2004), 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:

(i) Only small multiplying factors (eg. $\beta = 2$ or 3) are utilized so that the increase in code length (in VLC table) for each nonzero qDCTC is kept to the minimal. In particular, all prime factors no larger than $\phi \in \S_1$ is utilized for INTRA-MB, and all prime factors no larger than $\rho \in \S_2$ is utilized for INTER-MB. The restriction using ϕ and ρ reduces the payload since only a subset of β is considered for data embedding.

(ii) 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 less nonzero qDCTCs implies less multiplications by the factor β.

To ease the discussion, we define

$$\delta(V) = \text{filesize}(V') - \text{filesize}(V). \tag{17}$$

Note that $\delta(V) > 0$ since the modified video V' has larger filesize than the original video V. We use “filesize" and “video bitstream size" interchangeably for the rest of the chapter. Also, unless specified otherwise, “increase of video bitstream size" 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 6**. Last but not least, because higher embedding efficiency implies less modifications, which in turn leads to smaller $\delta(V)$, RZL proposed in **Section 4** could be utilized to further suppress $\delta(V)$ while trading off with payload.

6. Experimental results

Eight test videos are utilized to verify the performance of the proposed method. Each video is encoded by MPEG-1 compression standard using ISO/IEC TR 11172-5 (1998) at 1.5Mbps with 15 pictures in a GOP and picture type ratio of I:P:B = 1:4:10. More information on the test videos could be found in **Table 4**. 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 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.

Video	Video description	Total Frames			Dimension	MB per frame
		I	P	B		
V_1	Coastguard	20	80	198	352×288	396
V_2	Container	20	80	198	352×288	396
V_3	Driving	20	80	198	352×240	330
V_4	Flower garden	10	40	98	352×240	330
V_5	Foreman	20	80	198	352×288	396
V_6	Hall monitor	20	80	198	352×288	396
V_7	Mobile calendar	20	80	198	352×288	396
V_8	A walk in the sqaure	30	120	298	352×240	330

Table 4. Information of standard test videos

6.1 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 5**. As expected, the payload is relatively low if we utilize only a specific pair of α and β for data embedding (eg. $\alpha = 5$ 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 chapter, 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 are recorded in **Table 6**. 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 (eg. V_4 and V_7) offers high payload, and vice versa (eg. 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 (i) which is proposed in **Section 5** for suppressing filesize 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 of 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 $(13, 13)$

$\mathcal{Q}$	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.

Table 5. Distribution of usable MQ values for V_1 : Coastguard

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 ~ 104, ~ 38, and ~ 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 (ii) in **Section 5**). The results for applying this restriction to the MBs extracted with $(\phi,\rho) = (31,31)$ and $(2,5)$ are recorded in **Table 7** 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

Video	$(\phi,\rho)=(31,31)$			$(\phi,\rho)=(13,13)$			$(\phi,\rho)=(7,7)$			$(\phi,\rho)=(2,5)$		
	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
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
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
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
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
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
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
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

Table 6. Average payload per frame for each picture type with parameter $(\phi,\rho,\tau=384)$ and ORS $[bpf]$

Video	$(\phi,\rho)=(31,31)$								
	$\tau=10$			$\tau=4$			$\tau=1$		
	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
V_2	1.05	5.18	8.59	0.60	2.46	5.69	0.00	0.51	2.85
V_3	1.85	8.04	57.20	0.30	1.99	32.04	0.10	0.33	10.98
V_4	5.80	5.55	100.58	2.00	2.15	67.42	0.70	0.88	24.67
V_5	2.10	7.48	57.40	0.80	2.48	34.32	0.10	0.59	12.30
V_6	0.00	8.61	32.81	0.00	2.69	21.89	0.00	0.40	8.35
V_7	2.20	33.11	120.64	1.55	9.61	81.99	0.55	1.95	33.24
V_8	0.03	4.08	33.20	0.00	1.18	21.64	0.00	0.25	9.01

Video	$(\phi,\rho)=(2,5)$								
	$\tau=10$			$\tau=4$			$\tau=1$		
	I	P	B	I	P	B	I	P	B
V_1	0.00	4.28	27.66	0.00	1.19	15.71	0.00	0.28	5.29
V_2	1.00	4.01	4.76	0.55	1.85	3.22	0.00	0.38	1.72
V_3	1.10	4.93	19.93	0.25	1.11	11.40	0.10	0.19	3.74
V_4	3.20	1.28	29.53	1.00	0.55	20.26	0.20	0.23	7.50
V_5	1.25	4.53	33.34	0.40	1.54	20.31	0.05	0.36	7.22
V_6	0.00	4.09	20.20	0.00	1.55	11.87	0.00	0.33	3.61
V_7	0.55	9.46	36.63	0.35	3.01	24.74	0.20	0.59	10.00
V_8	0.00	2.56	14.08	0.00	0.86	9.23	0.00	0.21	4.03

Table 7. Average payload influenced by τ with respect to ORS $[bpf]$

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 ~ 5.4*bpf*.

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

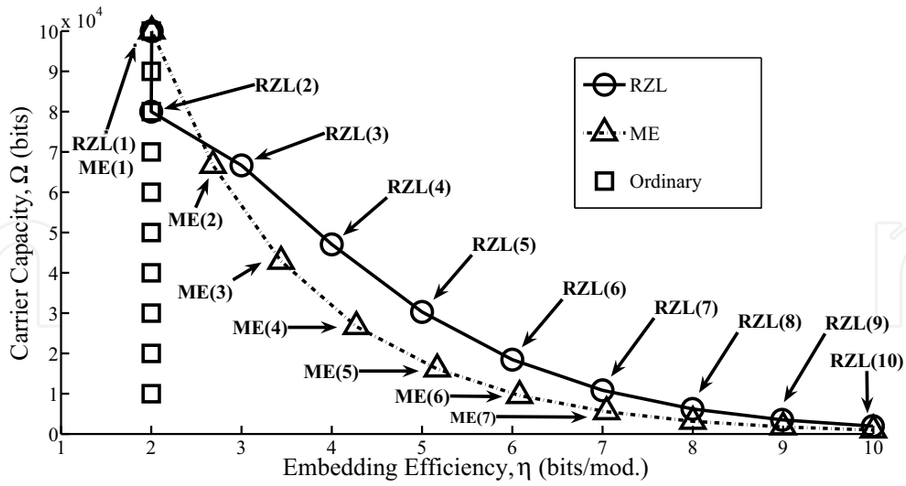


Fig. 5. Embedding efficiency vs. payload for ORS, ME, and RZL

parameter k of RZL, and it is recorded in **Table 3**. 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 (i), (ii), or RZL is applied, but their roles in suppressing $\delta(V_i)$ become obvious in **Section 6.3**.

6.2 Performance of RZL

In this section, we simulate the embedding process using ME (Crandall, 1998) and RZL as the data representation scheme. For comparison purposes, we also consider ORS proposed in **Section 3.2**. As for the experiment parameters, $|J| = 100000$, and $k = 1, 2, \cdots, 7$. For each k, a 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 **Figure 5**.

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, $\eta[\text{RZL}(k)]$ stays the same, i.e., 2, as indicted by the vertical (solid) line in **Figure 5**. 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., $\eta[\text{RZL}(3)] > \eta[\text{ME}(2)]$.

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

Table 8. Increase of video bitstream for ORS, ME(k), and RZL$(k+1)$ [Kbytes]

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

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

and

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

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).

6.3 Suppression of video bitstream size increment

First, we compare the performance of ME (Crandall, 1998) and RZL in terms of $\delta(V_i)$ when embedding the same amount of information. Since solution (i) and (ii) are independent, it

Video	$(\phi,\rho)=(31,31)$				$(\phi,\rho)=(2,5)$			
τ	384	10	4	1	384	10	4	1
V_1	414.18	47.87	13.62	1.99	166.78	18.33	5.20	0.82
V_2	120.94	5.27	1.56	0.35	60.28	3.12	0.88	0.21
V_3	522.67	34.89	8.47	1.22	198.74	12.67	3.13	0.46
V_4	364.40	20.83	6.59	0.92	98.60	5.98	1.93	0.28
V_5	300.97	32.88	9.04	1.40	141.87	18.24	5.41	0.86
V_6	158.86	19.92	6.90	1.17	85.22	12.78	3.93	0.50
V_7	676.18	67.76	21.17	3.43	179.60	19.90	6.29	1.00
V_8	416.32	24.57	7.08	1.24	185.12	10.15	2.94	0.58

Table 9. Increase of video bitstream size with respect to (ϕ,ρ,τ) [kbytes]

is sufficient to consider a particular set of parameters and infer the results for other settings. In partcular, we consider $(\tau,\phi,\rho)=(384,31,31)$ which causes the largest $\delta(V_i)$ during data embedding. Since

$$\Omega[\mathrm{ME}(k)] \leq \Omega[\mathrm{RZL}(k+1)] \leq \Omega[\mathrm{ORS}], \tag{20}$$

we embed a message of length $\Omega[\mathrm{ME}(k)]$ bits into each video by using ORS, $\mathrm{ME}(k)$, and $\mathrm{RZL}(k+1)$. 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 $k=2,3,\cdots,8$, and the resulting filesizes are recorded in **Table 8**. As expected, when either ME or RZL is applied, $\delta(V_i)$ is significantly suppressed. When embedding the same amount of information, $\mathrm{RZL}(k+1)$ consistently yields the smallest $\delta(V_i)$ when compared to ORS and $\mathrm{ME}(k)$ regardless of the value k. For example, using the filesize increased in case of ORS as the reference, the average reduction of $\delta(V_i)$ are 25% and 34% for $\mathrm{ME}(2)$ and $\mathrm{RZL}(3)$, respectively. However, when we consider $\mathrm{ME}(7)$ and $\mathrm{RZL}(8)$, the average reduction 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 **Figure 5**. 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 (i) and (ii) proposed in **Section 5** 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 9**. Using V_1 and the parameter setting of $(\phi,\rho)=(31,31)$ as the representative example, we elaborate the results. When $\tau=384$, $\delta(V_1)=414.18$ kbytes, which is the upper bound of $\delta(V_1)$ for the proposed method. Once τ is reduced to 10, $\delta(V_1)$ reduces to 47.87 kbytes, or equivalently $\sim 1/9$ of $\delta(V_1)$ for $\tau=384$. When τ is further reduced from 10 to 4, $\delta(V_1)$ is suppressed to 13.62 kbytes. A negligible $\delta(V_1)$ of ~ 2 kbytes is observed when $\tau=1$, but the payload is also significantly reduced (refer to **Table 7**). The result 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 as *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 10**. 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

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

Table 10. Coding efficiency for various (ϕ,ρ,τ) at 1.5Mbps $(\times 10^{-3})$

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 solution (i) and (ii) are capable in increasing the coding efficiency, and solution (ii) has greater impact in suppressing $\delta(V_i)$ than solution (i).
In conclusion, the results suggest that RZL, solution (i), and (ii) are capable in suppressing $\delta(V_i)$ while trading off with payload $\Omega(V_i)$.

6.4 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 Mbps for $i = 1,2,\cdots,8$. 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 11**. 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 other words, the rate controller is frequently in 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 $i = 1,2,\cdots,8$. These values are selected (i.e., payload of V_2 at 3.0 Mbps) because these are the smallest payloads for all videos and for all bitrates considered. $\delta(V_i)$ after data embedding is recorded in **Table 12** 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

Video	1.0Mbps			2.0Mbps		
	I	P	B	I	P	B
V_1	251.7	127.1	105.5	196	65.8	98.8
V_2	142.9	31.9	22.8	95.2	11.9	14.2
V_3	233.6	130	106.7	194.8	73.5	102.1
V_4	246.8	172.6	67.7	227.1	130.6	133.9
V_5	241.3	113.5	93.6	195	47.4	64.3
V_6	181.9	50.9	77.2	162.7	28.8	31.5
V_7	324.3	235.6	96	300.4	166.4	148.3
V_8	237.1	112	60.2	184.4	48.8	58.4

Video	2.5Mbps			3.0Mbps		
	I	P	B	I	P	B
V_1	176.5	41.2	80.5	161.7	27.4	58.2
V_2	74.3	2.1	12.3	55.8	1.5	10.1
V_3	180.5	52.4	83.3	167.8	37.4	66.7
V_4	217.7	107.2	125.6	205.9	85.2	120.7
V_5	173.9	32.8	51.3	152.5	22.4	41.4
V_6	138.7	22.9	29.2	118.9	19.4	31.4
V_7	289.2	141.8	148.3	275.8	109	154.9
V_8	125.6	24.3	33.9	107.9	14.6	31.0

[7]Refer to **Table 6** for payload at 1.5Mbps

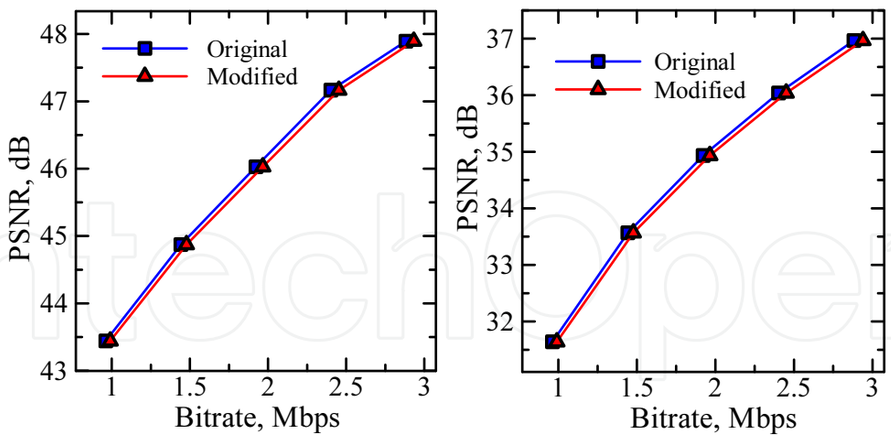
Table 11. Payload for video bitrate 1.0, 2.0, 2.5,and 3.0Mbps $[bpf]$

Bitrate [Mbps]	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

Table 12. Increase of video bitstream size for various bitrates [Kbytes]

$\delta(V_i)$ could be suppressed by tuning the parameters (ϕ, ρ, τ), and/or utilizing ME (Crandall, 1998) or RZL as the data representation scheme.

Last but not least, we utilize the results from **Table 12** and PSNR values to plot the graph of bitrate vs. PSNR in **Figure 6** 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 compare 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



(a) V_2: Container (b) V_7 : Mobile calendar

Fig. 6. Graphs of bitrate vs. PSNR

excitements and *promotions*. In other words, to achieve a specific PSNR value, MPEG-1 requires 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.

7. Conclusions

A novel data hiding method that completely preserves the image quality of the host video was 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 (reserve zerorun length) data representation scheme was proposed to improve the embedding efficiency by trading off with payload. It was theoretically and experimentally verified that RZL outperforms matrix encoding in terms of payload and embedding efficiency. Basic performance of this method was 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, which is sufficient for applications including indexing and annotation. The video bitstream size increases up to 40% when operating at full-capacity and this 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 works, 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.

8. References

Bodo, Y., Laurent, N. & Dugelay, J.-L. (2004). Watermarking video, hierachical embedding in motion vectors, *IEEE Proc. of ICIP*, pp. 739–742.

Budhia, U., Kundur, D. & Zourntos, T. (2006). Digital video steganalysis exploiting statistical visibility in the temporal domain, *IEEE Trans. on Information Forensics and Security* **1**(4): 502–516.

Cox, I., Miller, M. L. & Bloom, J. A. (2002). *Digital Watermarking*, Morgan Kaufmann Publishers.

Crandall, R. (1998). Some notes on steganography.
URL: *http://os.inf.tu-dresden.de/westfeld/crandall.pdf*

Hanzo, L., Cherriman, P. & Streit, J. (2007). *Video Compression and Communications*, IEEE PRESS.

Hartung, F. & Girod, B. (1996). Digital watermarking of raw and compressed video, *Proc. SPIE Digital Compression Technologies and Systems for Video Communication*, Vol. 2952, Berlin, Germany.

ISO/IEC TR 11172-5 (1998). 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, *Technical report*, ISO/IEC, Switzerland.

ITU-T H.261 (1990). Video codec for audiovisual service at p×64 kbit/s, *Technical report*, International Telecommunication Union, Geneva.

Johnson, N. F., Duric, Z. & Jajodia, S. (2003). *Information Hiding: Steganography and Watermarking - Attacks and Countermeasures*, Kluwer Academic Publishers.

Katzenbeisser, S. & Petitcolas, F. (2000). *Information Hiding Techniques for Steganography and Digital Watermarking*, Artech House Publishers.

Kiya, H., Noguchi, Y., Takagi, A. & Kobayashi, H. (1999). A method of inserting binary data into MPEG video in the compressed domain, *IEICE Trans. on fundamentals of electronics, communications and computer sciences* **82**(8): 1485–1492.

Kurosaki, M. & Kiya, H. (2002). Error concealment using a data hiding technique for MPEG video, *IEICE Trans. on fundamentals of electronics, communications and computer sciences* **E85-A**(4): 790–796.

Liu, Z., Liang, H., Niu, X. & Yang, Y. (2004). A robust video watermarking in motion vectors, *IEEE Proc. of International Conference of Signal Processing*, pp. 2358 – 2361.

Nakajima, K., Tanaka, K., Matsuoka, T. & Nakajima, Y. (2005). Rewritable data embedding on MPEG coded data domain, *IEEE Proc. of ICME*, pp. 682–685.

Ni, Z., Shi, Y.-Q., Ansari, N. & Su, W. (2006). Reversible data hiding, *IEEE Trans. on Circuits and Systems for Video Technology* **16**(3): 354–362.

Pranata, S., Wahadaniah, V., Guan, Y. L. & Chua, H. C. (2004). Improved bit rate control for real-time MPEG watermarking, *EURASIP Journal on Applied Signal Processing* **2004**(14): 2132–2141.

Qiu, G., Marziliano, P., Ho, A. T., He, D. & Sun, Q. (2004). A hybrid watermarking scheme for H.264/AVC video, *IEEE Proc. of ICPR*, pp. 865–868.

Sarkar, A., Madhow, U., Chandrasekaran, S. & Manjunath, B. S. (2007). Adaptive MPEG-2 video data hiding scheme, *SPIE Proc. of Security, Steganography, and Watermarking of Multimedia Contents IX*, pp. 489–497.

Symes, P. (2004). *Digital Video Compression*, McGraw-Hill.

Takayama, M., Tanaka, K., Yoneyama, A. & Nakajima, Y. (2006). A video scrambling scheme applicable to local region without data expansion, *IEEE Proc. of ICME*, pp. 1349–1352.

Wong, C., Yu, H. & Zheng, M. (2003). A DCT-based MPEG-2 transparent scrambling algorithm, *IEEE Trans. Consumer Electronics* **49**(4): 1208–1213.

Wong, K. & Tanaka, K. (2007). Mquant-based data hiding method in MPEG domain, *IIEEJ Proc. of Image Electronics and Visual Computing Workshop*.

Wong, K., Tanaka, K. & Qi, X. (2006). Multiple messages embedding using DCT-based Mod4 steganographic method, *LNCS Proc. of International Workshop on Multimedia Content Representation, Classification, and Security*, pp. 57–65.

Xu, C., Ping, X. & Zhang, T. (2006). Steganography in compressed video stream, *IEEE Proc. of International Conference on Innovative Computing, Information and Control*, pp. 269–272.

Yanagihara, H., Nakajima, Y., Matsuoka, T. & Tanaka, K. (2005). Advancement of streaming contents using watermark indexing (part iii) - development of software video player with watermark detection, *IIEEJ Proc. of 33rd Annual Conference*, pp. 77–78.

Zeng, W. & Lei, S. (1999). Efficient frequency domain video scrambling for content access control, *ACM Proc. of 7th International Multimedia Conference*, pp. 285–294.

Zhang, J., Li, J. & Zhang, L. (2001). Video watermark technique in motion vector, *IEEE Proc. of Brazilian Symposium on Computer Graphics and Image Processing*, pp. 179 – 182.

Appendix

[A]. Scaling equation for MPEG-1/2

We derive the scaling equation for INTER-MB. Assume that $x > 0$ (thus 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 Eq. (1), we want to update x using $x \leftarrow \beta \times x + y$ and hence, we need to find y such that Eq. (21) holds true.

$$\begin{aligned} rec &= \frac{[2 \times x + \text{sign}(x)] \times \mathcal{Q} \times \text{QT}_2}{16} \\ &= \frac{[2 \times (\beta \times x + y) + \text{sign}(\beta \times x + y)] \times \mathcal{Q}/\beta) \times \text{QT}_2}{16}. \end{aligned} \tag{21}$$

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 Eq. (21), 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 Eq. (2), we want to find y so that the following equation holds true:

$$\text{sign}(x) \cdot [2\alpha\beta|x| + \alpha\beta] = \text{sign}(\beta x + y) \cdot [2\alpha|\beta x + y| + \alpha] \tag{22}$$

Again, we have the freedom for setting the sign of y, and we force y to have the same sign as x. Hence, Eq (22) could be simplified to

$$2\beta|x| + \beta = 2|\beta x + y| + 1. \tag{23}$$

Assume that $x > 0$, then we obtain

$$2\beta x + \beta = 2\beta x + 2y + 1. \tag{24}$$

Simplifying Eq. (24) 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:

$$\text{sign}(x) \cdot [2\alpha\beta|x| + \alpha\beta - 1] = \text{sign}(\beta x + y) \cdot [2\alpha|\beta x + y| + \alpha - 1] \tag{25}$$

When we assume that x and $\beta x + y$ are both of the same sign, Eq. (25) simplifies to Eq. (23), 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 $\mathcal{Q}$.

IntechOpen

Multimedia

Edited by Kazuki Nishi

ISBN 978-953-7619-87-9

Hard cover, 452 pages

Publisher InTech

Published online 01, February, 2010

Published in print edition February, 2010

Multimedia technology will play a dominant role during the 21st century and beyond, continuously changing the world. It has been embedded in every electronic system: PC, TV, audio, mobile phone, internet application, medical electronics, traffic control, building management, financial trading, plant monitoring and other various man-machine interfaces. It improves the user satisfaction and the operational safety. It can be said that no electronic systems will be possible without multimedia technology. The aim of the book is to present the state-of-the-art research, development, and implementations of multimedia systems, technologies, and applications. All chapters represent contributions from the top researchers in this field and will serve as a valuable tool for professionals in this interdisciplinary field.

INTECH

open science | open minds

InTech Europe

University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China

Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen