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 $8 \times 8$ 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 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 maintenance 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 (e.g. 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 $8 \times 8$ 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.
\[
\text{rec}[m][n] = \begin{cases} 
2 \times x[m][n] \times Q(k) \times QT_1[m][n]/16, & \text{if INTRA} \\
(2 \times x[m][n] + \text{sign}(x[m][n])) \times Q(k) \times QT_2[m][n]/16, & \text{otherwise}
\end{cases}
\]

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 \(8 \times 8\) 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).

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

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 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.

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 \(j\)th MB in a slice. Also, let \(Q(j)\) associate the CODED_MQ flag and the MQ value of MB\((j)\) in the following manner:

\[
\begin{align*}
Q(j) = 0 & \iff \text{CODED_MQ} = \text{FALSE}; \\
Q(j) > 0 & \iff \text{CODED_MQ} = \text{TRUE}, \text{where the coded MQ value is as specified.}
\end{align*}
\]
that \( \beta = \alpha \times \beta \) to increase the payload. Here, 
\[
\mathbb{S}_1 := \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31\}. \tag{4}
\]

Figure 1

The embedding flow is summarized in Algorithm 1. Instead of a fixed MQ value, we consider \( \beta \in \mathbb{S}_1 \) and the corresponding \( \alpha \)'s such that \( Q = \alpha \times \beta \) to increase the payload. Here,
\[
\mathbb{S}_1 := \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31\}. \tag{4}
\]

Algorithm 1 Exciting an INTRA macroblock

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

Algorithm 2 Promoting a macroblock

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

Consider MB\((j_0)\) such that \( Q(j_0) = \alpha \times \beta \) for some \( \alpha, \beta \in \mathbb{N} \) and \( \beta \neq 1 \). Here, \( \mathbb{N} \) denotes the set of natural numbers. Obviously, \( \alpha \) and \( \beta \) are the factors of \( Q(j_0) \). If \( Q(j_0) \) is a prime, then the factors are unity and \( Q(j_0) \) itself. We excite MB\((j_0)\) using Algorithm 1. Here, \( \alpha \) is chosen as the new \( Q(j_0) \) value and \( \beta \) 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 \( \beta \) that is “divided out” from the original \( Q(j_0) \) is compensated by the multiplication of \( \beta \) 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 \( Q(j_0) \) 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. \( Q(j_0 + 1) \neq 0 \) 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 \( i \)th 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 \( \beta \in \mathbb{S}_1 \) and the corresponding \( \alpha \)'s such that \( Q = \alpha \times \beta \) to increase the payload. Here,
\[
\mathbb{S}_1 := \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31\}. \tag{4}
\]
Thus, any MQ value in the range of $[2, 31]$ could be factored into $\alpha$ and $\beta$ for at least one $\beta \in \mathcal{S}_1$. Note that each $\beta \in \mathcal{S}_1$ 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 $Q \neq 0$ 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

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

for each nonzero qDCTC $x \in \text{MB}$. Case (i) occurs if condition (5) is true for a specific $\beta \in \mathcal{S}_1$ for all $x \in \text{MB}$. We store the smallest value of $\beta \in \mathcal{S}_1$ that satisfies condition (5), update $Q \leftarrow Q \times \beta$, and yank an “1”. On the other hand, case (ii) occurs if condition (5) fails but $\text{mod}\ (Q, \beta) = 0$ for at least one $\beta \in \mathcal{S}_1$ (i.e., Cond.D holds true). In this case, “0” is yanked as
Fig. 2. Flowchart for data extraction and restoration. \(^4\text{Cond.C}:=\) Is \(\text{mod} (x, \beta) = 0\) for all qDCTCs \(x \in \text{MB}(j)\) for at least one \(\beta \in \mathbb{S}_1\)\(^2\), \(^5\text{Cond.D}:=\) Is \(\text{mod} (Q(j), \beta) = 0\) for at least one \(\beta \in \mathbb{S}_1\)\(^2\),

the output. Case (iii) occurs if \text{Cond.D} fails, i.e., when the following holds true, and nothing is output:

\[
Q = 1
\]  
(6)

Condition (5) or more generally, \text{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 \(\beta \in \mathbb{S}_1\), 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 \(Q = a\) in the original video and all of its nonzero AC qDCTCs are divisible by \(\beta\). We make sure that no other MQ values get mapped to \(a\) during data embedding, i.e., \(\text{MB}(j)\)'s with \(Q(j) = 2 \times a, 3 \times a, \cdots\) are ignored for data embedding. However, \(\text{MB}(j)\)'s with \(Q(j) = a\) themselves could be utilized for data embedding since the value \(a\) will be replaced by \(\hat{a} (a = \hat{a} \times \hat{\beta}\), and \(\hat{a}, \hat{\beta} \leq a\) if they happened to be excited during data embedding. When \(\text{MB}(j)\) with \(Q(j) = a\) 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 \(a \times \beta\), then it is either a promoted MB or an excited MB, depending on \text{Cond.C}. Otherwise, it is an MB that may hold “0”, “1”, or nothing, and hence condition \text{Cond.C} and \text{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 \(\text{MB}(j_0)\) is identified as holding “1”, i.e., \(\text{MB}(j_0)\) is excited with \(Q(j_0) = a\), and the information held by \(\text{MB}(j_0 + 1)\) is to be determined. We list the interpretations of \(\text{MB}(j_0 + 1)\) based on \(Q(j_0 + 1)\)
<table>
<thead>
<tr>
<th>MB((j_0))</th>
<th>MB((j_0 + 1))</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\alpha)</td>
<td>(\neq \alpha \times \beta)</td>
<td>MB((j_0)) encodes “1”, and MB((j_0 + 1)) encodes “0”, “1” or nothing. MB((j_0)) encodes “1”, and MB((j_0 + 1)) is a promoted MB, or it encodes “1”. It can never encode “0” because (Q(j_0)) and (Q(j_0 + 1)) can never be the same in the original video.</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>(\alpha \times \beta)</td>
<td></td>
</tr>
<tr>
<td>(\alpha \times \beta)</td>
<td>(\neq \alpha \times \beta)</td>
<td>MB((j_0)) encodes “0”, and MB((j_0 + 1)) encodes “0”, “1” or nothing. MB((j_0)) encodes “0”, and MB((j_0 + 1)) must encode “1” because (Q(j_0)) and (Q(j_0 + 1)) can never be the same in the original video.</td>
</tr>
<tr>
<td>(\alpha \times \beta)</td>
<td>(\alpha \times \beta)</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Distinguishing promoted MB and unmodified MB

**Algorithm 3** Restoring an excited INTRA macroblock

```
1: \(Q \leftarrow \alpha \times \beta\)
2: for Y, Cb, and Cr channel do
3:   for all nonzero qDCTC\([m][n]\) do
4:     qDCTC\([m][n]\) \leftarrow qDCTC\([m][n]\) / \(\beta\)
5: end for
6: end for
```

**Algorithm 4** Demoting a promoted macroblock

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

In Table 1, for completeness of discussion, we also list the cases where MB\((j_0)\) encodes “0”, i.e., \(Q(j_0) = \alpha \times \beta\).

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 \(Q(j)\) is recorded in the array if \(Q(j) > 0\), or ‘X’ is marked otherwise. Suppose the message \(\mu = 1010 \cdots\). MB\((j_0)\) is excited to hold the first bit of \(\mu\) (i.e., “1”), thus \(Q(j_0)\) 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 \(Q(j_0 + 1) = 10 \neq 0\), MB\((j_0 + 1)\) is not promoted. Instead, MB\((j_0 + 1)\) is considered to encode the second bit of \(\mu\). MB\((j_0 + 1)\) is left as it is because an “0” is to be embedded. Note that \(Q(j_0 + 2) = 0\), 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 \(\mu\) (i.e., “1”). MB\((j_0 + 4)\) is promoted so that \(Q(j_0 + 4) = 13\). MB\((j_0 + 5)\) is left as it is to encode the fourth bit of \(\mu\) (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 \(Q(j_0 + 1) > 0\) and \(Q(j_0 + 1) \neq Q(j_0) \times 2\), 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 $Q(j_0 + 4) = Q(j_0 + 3) \times 13$ 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:

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

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 $\beta$ has to be an odd number, and 2 must be removed from $\mathcal{S}_1$ when processing an INTER-MB. In particular, we use $\beta \in \mathcal{S}_2$ when dealing with INTER-MBs where

$$\mathcal{S}_2 := \mathcal{S}_1 \setminus \{2\} = \{3, 5, 7, 11, 13, 17, 19, 23, 29, 31\}.$$  

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:

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

**Cond.C** is also updated accordingly using $\mathcal{S}_2$ in place of $\mathcal{S}_1$. An INTER-MB is declared as encoding nothing if it fails **Cond.D**, and this happens when

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

---

Fig. 3. Example: Encoding and decoding information

---

![Diagram](diagram.png)
Algorithm 3 is utilized to restore an excited INTER-MB, except that the division operation in line 4 is replaced by
\[
x \leftarrow (x - \text{sign}(x) \times y) / \beta
\]
for all nonzero qDCTC \( x \in \text{MB} \). 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 \( \eta \) 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 \( \mu \) while aiming to reduce the number of MB excitations.

Let \( \mu \) be a binary message of length \( |\mu| \), and let \( k \in \mathbb{N} \) such that \( k \geq 2 \). \( \mu \) is divided into segments of length \( k \) bits, and each segment \( \mu(i) \) is processed one at a time for \( i = 1, 2, \cdots, \lceil \frac{|\mu|}{k} \rceil \), 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 \( k = 3 \),

\[
\begin{align*}
\text{BIN2RZL}(0011) &= 0001 \quad \text{(i.e., three zeros followed by unity)} \\
\text{BIN2RZL}(1011) &= 000001 \quad \text{(i.e., five zeros followed by unity)} \\
\text{BIN2RZL}(001) &= 01 \quad \text{(i.e., one zero followed by unity)}
\end{align*}
\]

(12)

To embed the message \( \mu \) using RZL, the output of \( \text{BIN2RZL}(\mu(i)) \) for all \( i \) are concatenated. The resulting sequence of zeros and ones are treated as a new message \( \hat{\mu} \), and \( \hat{\mu} \) is embedded
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 \( \mu = 011010001 \) in case of ORS but only three MB excitations 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 excitations 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 discussion, let \( \hat{\mu}(j) \) denote the \( j \)th 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,

\[
\text{RZL2BIN}(\text{BIN2RZL}(\mu(i)), k) = \mu(i).
\]  

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 \( Q = a \times b \) for \( b \in \mathbb{S}_1 \) or \( b \in \mathbb{S}_2 \) (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., \( \mu(i) = 000 \cdots 0 \)), and occupies \( 2^k \) locations in the worst case scenario (i.e., \( \mu(i) = 111 \cdots 1 \)).

Since we may assume that the message \( \mu \) 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

\[
\frac{2^k \cdot (2^k + 1)}{2} \times \frac{1}{2^k} = \frac{2^k + 1}{2} = 2^{k-1} + 0.5
\]

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, \( \lceil |\mu|/k \rceil \times \)
Table 2. Expected number of locations required

<table>
<thead>
<tr>
<th>k</th>
<th>1 2 3 4 5 6 7 8 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME</td>
<td>1 3 7 15 31 63 127 255 511</td>
</tr>
<tr>
<td>RZL</td>
<td>1 2.5 4.5 8.5 16.5 32.5 64.5 128.5 256.5</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>k</th>
<th>1 2 3 4 5 6 7 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ME</td>
<td>1.000 0.667 0.429 0.267 0.161 0.095 0.055 0.031</td>
</tr>
<tr>
<td>RZL</td>
<td>1.000 0.800 0.667 0.471 0.303 0.185 0.109 0.062</td>
</tr>
</tbody>
</table>

$(2^{(k-1)} + 0.5)$ gives a coarse estimation for the number of locations required to encode $\mu$ 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 $\eta$ 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

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

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

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

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 $\beta$. 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 (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 minimal. In particular, all prime factors no larger than $\phi \in \mathbb{S}_1$ is utilized for INTRA-MB, and all prime factors no larger than $\rho \in \mathbb{S}_2$ is utilized for INTER-MB. The restriction using $\phi$ and $\rho$ reduces the payload since only a subset of $\beta$ is considered for data embedding.

(ii) Instead of using all MBs with $Q = \alpha \times \beta$ ($\beta \in \mathbb{S}_1$ and $\beta \in \mathbb{S}_2$ in case of INTRA and INTER-MB, respectively) for data embedding, only MBs that are made up of at most $\tau$ 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 $\beta$.

To ease the discussion, we define

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

(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 $\delta(V_j)$ 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.
### Table 4. Information of standard test videos

<table>
<thead>
<tr>
<th>Video</th>
<th>Video description</th>
<th>Total Frames</th>
<th>Dimension</th>
<th>MB per frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_1$</td>
<td>Coastguard</td>
<td>20 I 80 P 198</td>
<td>352 x 288</td>
<td>396</td>
</tr>
<tr>
<td>$V_2$</td>
<td>Container</td>
<td>20 I 80 P 198</td>
<td>352 x 288</td>
<td>396</td>
</tr>
<tr>
<td>$V_3$</td>
<td>Driving</td>
<td>20 I 80 P 198</td>
<td>352 x 240</td>
<td>330</td>
</tr>
<tr>
<td>$V_4$</td>
<td>Flower garden</td>
<td>10 I 40 P 98</td>
<td>352 x 240</td>
<td>330</td>
</tr>
<tr>
<td>$V_5$</td>
<td>Foreman</td>
<td>20 I 80 P 198</td>
<td>352 x 288</td>
<td>396</td>
</tr>
<tr>
<td>$V_6$</td>
<td>Hall monitor</td>
<td>20 I 80 P 198</td>
<td>352 x 288</td>
<td>396</td>
</tr>
<tr>
<td>$V_7$</td>
<td>Mobile calendar</td>
<td>20 I 80 P 198</td>
<td>352 x 288</td>
<td>396</td>
</tr>
<tr>
<td>$V_8$</td>
<td>A walk in the square</td>
<td>30 I 120 P 298</td>
<td>352 x 240</td>
<td>330</td>
</tr>
</tbody>
</table>

6.1 Payload  

First, we consider the payload offered by each $Q = \alpha \times \beta$ for $\beta \in S_1$ and $\beta \in S_2$ in case of INTRA and INTER-MB, 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 $\alpha$ and $\beta$ for data embedding (eg. $\alpha = 5$ and $\beta = 3$ that provide merely 20-bits in INTRA-I). However, more than a pair of $\alpha$ and $\beta$ could be utilized simultaneously to provide higher total payload. Note that the payload offered by $Q = 1$ is always zero because $1 \notin S_1$ and $1 \notin S_2$. Also, in case of INTER-MB, the payload offered by $Q \in \{2, 4, 8, 16\}$ 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 $\alpha$ and $\beta$ 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 $\phi$ or $\rho$ 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 $(\phi, \rho)$ 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 $(\phi, \rho)$ from $(13,13)$

www.intechopen.com
Table 5. Distribution of usable MQ values for $V_1$: Coastguard

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.

To $(7, 7)$. When $(\phi, \rho)$ 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 $\sim 33\text{bpf}$ for I, P and B-pictures, respectively.

Now, suppose that data embedding is restricted to MBs with at most $\tau$ 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 $\tau$ 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 $\tau$ is reduced from 384 to 10, the
Table 6. Average payload per frame for each picture type with parameter $(\phi, \rho, \tau = 384)$ and ORS [bpf]

<table>
<thead>
<tr>
<th>Video</th>
<th>$(\phi, \rho) = (31, 31)$</th>
<th>$(\phi, \rho) = (13, 13)$</th>
<th>$(\phi, \rho) = (7, 7)$</th>
<th>$(\phi, \rho) = (2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I</td>
<td>P</td>
<td>B</td>
<td>I</td>
</tr>
<tr>
<td>$V_1$</td>
<td>219.8</td>
<td>97.1</td>
<td>101.0</td>
<td>219.8</td>
</tr>
<tr>
<td>$V_2$</td>
<td>118.1</td>
<td>20.5</td>
<td>12.8</td>
<td>118.1</td>
</tr>
<tr>
<td>$V_3$</td>
<td>215.1</td>
<td>103.6</td>
<td>106.5</td>
<td>214.8</td>
</tr>
<tr>
<td>$V_4$</td>
<td>239.0</td>
<td>144.2</td>
<td>132.2</td>
<td>212.3</td>
</tr>
<tr>
<td>$V_5$</td>
<td>216.5</td>
<td>64.9</td>
<td>88.5</td>
<td>216.2</td>
</tr>
<tr>
<td>$V_6$</td>
<td>165.2</td>
<td>29.3</td>
<td>45.3</td>
<td>165.2</td>
</tr>
<tr>
<td>$V_7$</td>
<td>312.0</td>
<td>190.4</td>
<td>154.3</td>
<td>282.1</td>
</tr>
<tr>
<td>$V_8$</td>
<td>162.4</td>
<td>49.5</td>
<td>48.0</td>
<td>162.4</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Video</th>
<th>$(\phi, \rho) = (31, 31)$</th>
<th>$(\phi, \rho) = (2, 5)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\tau = 10$</td>
<td>$\tau = 4$</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>P</td>
</tr>
<tr>
<td>$V_1$</td>
<td>0.00</td>
<td>7.53</td>
</tr>
<tr>
<td>$V_2$</td>
<td>1.05</td>
<td>5.18</td>
</tr>
<tr>
<td>$V_3$</td>
<td>1.85</td>
<td>8.04</td>
</tr>
<tr>
<td>$V_4$</td>
<td>5.80</td>
<td>5.55</td>
</tr>
<tr>
<td>$V_5$</td>
<td>2.10</td>
<td>7.48</td>
</tr>
<tr>
<td>$V_6$</td>
<td>0.00</td>
<td>8.61</td>
</tr>
<tr>
<td>$V_7$</td>
<td>2.20</td>
<td>33.11</td>
</tr>
<tr>
<td>$V_8$</td>
<td>0.03</td>
<td>4.08</td>
</tr>
</tbody>
</table>

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 $\tau$ 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 $\sim 5.4 \text{bpf}$. Finally, when RZL is applied to any combination of $(\phi, \rho, \tau)$, we expect the resulting payload to be a fraction of the original payload achieved by $(\phi, \rho, \tau)$. The fraction depends on the
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 \( \phi, \rho \), and \( \tau \), 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 \( k = 1, 2, \ldots, 7 \). 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 (\( \omega \)) and the number of modifications (\( \nu \)) are counted. This process is repeated for 1000 times, and the average of \( \omega \) and \( \nu \) 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, \( \eta[RZL(k)] \) 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, \( \eta[RZL(2)] = 2/1 \).

For each \( k \), it is observed that RZL(2) 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., \( \Omega[ME(2)] \sim \Omega[RZL(3)] \), but RZL(3) achieves a higher embedding efficiency when compared to ME(2), i.e., \( \eta[RZL(3)] > \eta[ME(2)] \).
<table>
<thead>
<tr>
<th>Ordinary representation scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_1$</td>
</tr>
<tr>
<td>$V_2$</td>
</tr>
<tr>
<td>$V_3$</td>
</tr>
<tr>
<td>$V_4$</td>
</tr>
<tr>
<td>$V_5$</td>
</tr>
<tr>
<td>$V_6$</td>
</tr>
<tr>
<td>$V_7$</td>
</tr>
<tr>
<td>$V_8$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Matrix encoding with parameter $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video</td>
</tr>
<tr>
<td>$V_1$</td>
</tr>
<tr>
<td>$V_2$</td>
</tr>
<tr>
<td>$V_3$</td>
</tr>
<tr>
<td>$V_4$</td>
</tr>
<tr>
<td>$V_5$</td>
</tr>
<tr>
<td>$V_6$</td>
</tr>
<tr>
<td>$V_7$</td>
</tr>
<tr>
<td>$V_8$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reverse zerorun length with parameter $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video</td>
</tr>
<tr>
<td>$V_1$</td>
</tr>
<tr>
<td>$V_2$</td>
</tr>
<tr>
<td>$V_3$</td>
</tr>
<tr>
<td>$V_4$</td>
</tr>
<tr>
<td>$V_5$</td>
</tr>
<tr>
<td>$V_6$</td>
</tr>
<tr>
<td>$V_7$</td>
</tr>
<tr>
<td>$V_8$</td>
</tr>
</tbody>
</table>

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,

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

and

$$\eta[ME(k)] \leq \eta[RZL(k + 1)]$$  \hspace{1cm} (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
is sufficient to consider a particular set of parameters and infer the results for other settings. In particular, we consider \((\tau, \phi, \rho) = (384, 31, 31)\) which causes the largest \(\delta(V_i)\) during data embedding. Since

\[
\Omega[\text{ME}(k)] \leq \Omega[\text{RZL}(k + 1)] \leq \Omega[\text{ORS}],
\]

we embed a message of length \(\Omega[\text{ME}(k)]\) 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 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, 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 filesize increased in case of ORS as the reference, the average reduction 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 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 \((\phi, \rho, \tau)\). 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 \(\tau\) 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 \(\tau\) is further reduced from 10 to 4, \(\delta(V_1)\) is suppressed to 13.62 kbytes. A negligible \(\delta(V_1)\) of \(\sim 2\) 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 \(\varepsilon\) that is defined as the number of message bits embedded per video bit increased. Higher \(\varepsilon\) implies smaller \(\delta(V_i)\), and vice versa. The results for \(\varepsilon\) 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

<table>
<thead>
<tr>
<th>Video</th>
<th>((\phi, \rho) = (31, 31))</th>
<th>((\phi, \rho) = (2, 5))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau)</td>
<td>384</td>
<td>10</td>
</tr>
<tr>
<td>(V_1)</td>
<td>414.18</td>
<td>47.87</td>
</tr>
<tr>
<td>(V_2)</td>
<td>120.94</td>
<td>5.27</td>
</tr>
<tr>
<td>(V_3)</td>
<td>522.67</td>
<td>34.89</td>
</tr>
<tr>
<td>(V_4)</td>
<td>364.40</td>
<td>20.83</td>
</tr>
<tr>
<td>(V_5)</td>
<td>300.97</td>
<td>32.88</td>
</tr>
<tr>
<td>(V_6)</td>
<td>158.86</td>
<td>19.92</td>
</tr>
<tr>
<td>(V_7)</td>
<td>676.18</td>
<td>67.76</td>
</tr>
<tr>
<td>(V_8)</td>
<td>416.32</td>
<td>24.57</td>
</tr>
</tbody>
</table>

Table 9. Increase of video bitstream size with respect to \((\phi, \rho, \tau)\) [kbytes]
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 \( \varepsilon \) increases when \( \tau \) 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 \( \tau \) 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

<table>
<thead>
<tr>
<th>Video</th>
<th>((\phi, \rho) = (31, 31))</th>
<th>((\phi, \rho) = (2, 5))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau )</td>
<td>384</td>
<td>10</td>
</tr>
<tr>
<td>( V_1 )</td>
<td>9.5</td>
<td>38.0</td>
</tr>
<tr>
<td>( V_2 )</td>
<td>6.6</td>
<td>49.4</td>
</tr>
<tr>
<td>( V_3 )</td>
<td>7.9</td>
<td>42.0</td>
</tr>
<tr>
<td>( V_4 )</td>
<td>7.1</td>
<td>59.4</td>
</tr>
<tr>
<td>( V_5 )</td>
<td>11.0</td>
<td>44.6</td>
</tr>
<tr>
<td>( V_6 )</td>
<td>11.2</td>
<td>44.0</td>
</tr>
<tr>
<td>( V_7 )</td>
<td>9.4</td>
<td>47.9</td>
</tr>
<tr>
<td>( V_8 )</td>
<td>7.4</td>
<td>51.6</td>
</tr>
</tbody>
</table>

Table 10. Coding efficiency for various \((\phi, \rho, \tau)\) at 1.5Mbps \((\times 10^{-3})\)
Table 11. Payload for video bitrate 1.0, 2.0, 2.5, and 3.0Mbps [bpf]

<table>
<thead>
<tr>
<th>Video</th>
<th>1.0Mbps</th>
<th></th>
<th>2.0Mbps</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>V₁</td>
<td>251.7</td>
<td>127.1</td>
<td>105.5</td>
<td>196</td>
</tr>
<tr>
<td>V₂</td>
<td>142.9</td>
<td>31.9</td>
<td>22.8</td>
<td>95.2</td>
</tr>
<tr>
<td>V₃</td>
<td>233.6</td>
<td>130.7</td>
<td>106.7</td>
<td>194.8</td>
</tr>
<tr>
<td>V₄</td>
<td>246.8</td>
<td>172.6</td>
<td>67.7</td>
<td>227.1</td>
</tr>
<tr>
<td>V₅</td>
<td>241.3</td>
<td>113.5</td>
<td>93.6</td>
<td>195</td>
</tr>
<tr>
<td>V₆</td>
<td>181.9</td>
<td>50.9</td>
<td>77.2</td>
<td>162.7</td>
</tr>
<tr>
<td>V₇</td>
<td>324.3</td>
<td>253.6</td>
<td>96</td>
<td>300.4</td>
</tr>
<tr>
<td>V₈</td>
<td>237.1</td>
<td>112</td>
<td>60.2</td>
<td>184.4</td>
</tr>
</tbody>
</table>

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

| δ(Vᵢ) | 1.5Mbps | | 2.0Mbps | | 2.5Mbps | | 3.0Mbps | | 
|--------|--------|--------|--------|--------|--------|--------| 
| V₁    | 176.5  | 41.2   | 80.5   | 161.7  | 27.4   | 58.2   | 
| V₂    | 74.3   | 2.1    | 12.3   | 55.8   | 1.5    | 10.1   | 
| V₃    | 180.5  | 52.4   | 83.3   | 167.8  | 37.4   | 66.7   | 
| V₄    | 217.7  | 107.2  | 125.6  | 205.9  | 85.2   | 120.7  | 
| V₅    | 173.9  | 32.8   | 51.3   | 152.5  | 22.4   | 41.4   | 
| V₆    | 138.7  | 22.9   | 29.2   | 118.9  | 19.4   | 31.4   | 
| V₇    | 289.2  | 141.8  | 148.3  | 275.8  | 109    | 154.9  | 
| V₈    | 125.6  | 24.3   | 33.9   | 107.9  | 14.6   | 31.0   | 

δ(Vᵢ) 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. V₂ and V₇ 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...
excitement 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


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 \( Q \) is updated from \( \alpha \times \beta \) to \( \alpha \), which is the same as \( Q/\beta \). 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.

\[
\text{rec} = \frac{[2 \times x + \text{sign}(x)] \times Q \times QT_2}{16}
\]

\[
= \frac{[2 \times (\beta \times x + y) + \text{sign}(\beta \times x + y)] \times Q/\beta \times QT_2}{16}
\]

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 \( Q = \alpha \times \beta \). First, we consider the case when \( Q \) is odd. Referring to Eq. (2), we want to find \( y \) so that the following equation holds true:

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

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

\[
2\beta|x| + \beta = 2|\beta x + y| + 1.
\]
Assume that $x > 0$, then we obtain

$$2\beta x + \beta = 2\beta x + 2y + 1. \quad (24)$$

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 $Q$ is even. We want to find $y$ so that the following equation holds true:

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

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 $Q$. 
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.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:
