Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 476181, 11 pages
http://dx.doi.org/10.1155/2013/476181

Research Article

Reversible Data Hiding Scheme with High Embedding Capacity Using Semi-Indicator-Free Strategy

Jiann-Der Lee,[1] Yaw-Hwang Chiou,[1] and Jing-Ming Guo[2]

[1] *Department of Electrical Engineering, Chang Gung University, Tao-Yuan 333, Taiwan*
[2] *Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan*

Correspondence should be addressed to Jing-Ming Guo; jmguo@seed.net.tw

Received 12 June 2013; Accepted 5 September 2013

Academic Editor: Marco Perez-Cisneros

A novel reversible data-hiding scheme is proposed to embed secret data into a side-matched-vector-quantization- (SMVQ-) compressed image and achieve lossless reconstruction of a vector-quantization- (VQ-) compressed image. The rather random distributed histogram of a VQ-compressed image can be relocated to locations close to zero by SMVQ prediction. With this strategy, fewer bits can be utilized to encode SMVQ indices with very small values. Moreover, no indicator is required to encode these indices, which yields extrahiding space to hide secret data. Hence, high embedding capacity and low bit rate scenarios are deposited. More specifically, in terms of the embedding rate, the bit rate, and the embedding capacity, experimental results show that the performance of the proposed scheme is superior to those of the former data hiding schemes for VQ-based, VQ/SMVQ-based, and search-order-coding- (SOC-) based compressed images.

1. Introduction

With the explosive growth of communication through the Internet, information processing and management at any time have become a standard service for most people, subsequently inducing security problems such as interception, modification and montage. Thus, one critical issue is how to find effective ways to protect information transmission over the Internet with safety and security. To cope with this, data hiding techniques have attracted attentions for being able to embed secret data into a cover image with minimal perceptual degradation.

In general, data hiding techniques can be classified into two categories, namely, reversible data hiding schemes and irreversible data hiding schemes. For irreversible data hiding schemes, only secret data can be extracted, while restoration of cover images is unavailable. The irreversible data hiding scheme is not suited for certain applications that demand the original cover image to be unaltered by the data hiding process. Conversely, reversible data hiding schemes can extract the secret data and recover the original cover images in the decoder. Moreover, data hiding schemes can be performed in three possible formats, that is, in spatial domain [1–5], in frequency domain [6–8], and in compressed domain [9–15]. In spatial domain, Tian [1] proposed a scheme for reversible data hiding, called difference expansion (DE). Tian's work expands the value differences between two neighboring pixels to embed a bit. This scheme is easy to realize, yet the embedding capacity heavily depends on the smoothness of an image. Ni et al. [2] utilized the zero of the minimum points of the histogram in an image and slightly modified the pixel values to embed secret data into the peak points of the histogram of an image. The scheme is simple for implementation, and which supports high visual quality. However, the embedding capacity is restricted by the peak points of the histogram of an image. Moreover, a cover image can be transformed into the frequency domain with various possible transform kernels, for example, DCT, DFT, or DWT. Normally, frequency domain techniques embed message by modulating the transformed coefficients of subbands with just noticeable difference (JND) according to the sensitivity of the human visual system (HVS).

Data hiding techniques in compressed domain can relish the both advantages of data hiding and compression for a multimedia distribution. Jo and Kim [9] firstly proposed an irreversible VQ-based scheme which is easy to realize, yet

the hiding capacity is rather small. Later, many VQ/SMVQ-based schemes with indicators are proposed. For example, Chang et al. [10] developed a VQ-based data hiding with recovery capability. The scheme has low embedding capacity and high bit rate (BR) because of an indicator added in front of most encoded indices, and only two clusters are used to hide secret data. In [11, 12], a secret data hiding was proposed based on the search-order coding compression method of VQ indices to increase embedding capacity. The search-order coding selects the neighboring indices of the encoding index to form a state codebook, and then search-order code (SOC) and original index value (OIV) are employed to hide a secret bit 0 or 1. A 1-bit indicator is always employed to distinguish between SOC and OIV. For this scheme, the original cover image can be restored completely. Shie et al. [13] developed an adaptive data hiding based on VQ-compressed image, in which image blocks are classified into embeddable and unembeddable blocks based on the variances and side-match distortions (SMDs). The embeddable blocks are employed to hide secret data, while those unembeddable blocks remain unchanged. It is necessary that indicator be used for the block judgment. Although this scheme can yield a high embedding capacity, the quality of the reconstructed image reduces as the quantity of the secret data increases. In addition, this scheme is a case of irreversible information hiding. In [14], both VQ and SMVQ are used to hide secret data, in which a 1-bit indictor is always required for each index and only 1-bit secret data can be embedded in each index when the VQ technique is applied. Conversely, more than one bit secret data can be embedded in each index when the SMVQ technique is applied. This scheme can provide a higher hiding capacity, yet it is an irreversible data-hiding scheme. In [15], Yang and Lin extended the scheme proposed by Chang et al. [10] by dividing the VQ codebook into 2^{BS} clusters, and half of which are used to embed secret data, where BS denotes the size of the secret data embedded into each VQ index. In the scheme, both of the VQ and SMVQ are applied to hide secret data and a 1-bit indictor is required for the index identification. Moreover, it is a reversible data-hiding scheme. In these schemes, indicators are always used, which causes a higher bit rate.

By exploiting the special indices distribution of a SMVQ-compressed image that frequently used indices are encoded by short codes while rarely-used indices are encoded by long codes, this work proposed a novel reversible data hiding in the SMVQ-compressed domain. The scheme can increase the hiding capacity and completely restore the cover image from an embedded image after the hiding data is extracted. Moreover, the indices of a VQ-compressed image can be relocated close to zero by using SMVQ predications. SMVQ indices located around zero can be encoded using fewer bits. Compared with the previous schemes [10, 12, 15], in which the bit rates are increased because indicators are required in most of the indices, the proposed scheme employs the SMVQ predication, and thus most of the indices are encoded without indicator. As a result, high embedding capacity and low bit rate scenarios can be achieved. As documented in the experimental results, the performance of the proposed scheme is significantly superior to that of the previous works [10, 12, 15] in terms of the embedding rate, bit rate, and embedding capacity.

The rest of this paper is organized as follows. Section 2 presents the related works to briefly introduce the concepts of VQ and SMVQ. Section 3 describes the proposed scheme in detail. Section 4 gives the experimental results and performances comparisons of the proposed scheme with former approaches. Finally, Section 5 draws conclusions.

2. Backgrounds

2.1. Brief Concept of the VQ. VQ initially involves codebook construction from a set of training images; the training images are partitioned into nonoverlapping blocks and the most representative blocks are selected to form codebook, in which the elements in the codebook are called codewords. In general, the LBG algorithm [16] is employed to produce the desired codebook. With the generated codebook, each block in an image is encoded with the index of the nearest codeword, such that the total storage space for an image is minimized. To measure the similarity between a block, $B = (b_1, b_2, \ldots, b_k)$ and a codeword, $C_j = (c_{j1}, c_{j2}, \ldots, c_{jk})$, where C_j is the jth codeword in the codebook, the squared Euclidean distance is generally used as below:

$$D\left(B, C_j\right) = \sum_{i=1}^{k} \left(b_i - c_{ji}\right)^2. \quad (1)$$

When the encoding process is completed, block B is simply represented by the index min $\_j$, where $C_{\min\_j}$ is the nearest codeword of the block B. For decoding process, the table lookup is performed with the same codebook as that used by the encoder.

2.2. SMVQ. Kim initially proposed SMVQ for image coding [17]. SMVQ exploits the high correlation among neighboring blocks and side-match prediction to create a state codebook with a size smaller than that of the original VQ codebook. The side-match prediction assumes that the values of the adjacent pixels between the neighboring blocks are equal. In SMVQ, blocks located in the first row and first column of an image are encoded by traditional VQ and the remaining blocks are predicted using the corresponding state codebook. Figure 1 shows an example of the relationships among an encoding block X, its upper neighboring block U, and its left neighboring block L for blocks of size 4×4. This work denotes the border vector and the side vector of block X as $B(X) = (X1, X2, X3, X4, X5, X9, X13) = (b_1, b_2, b_3, b_4, b_5, b_6, b_7)$ and $S(X, U, L) = ((U13 + L4)/2, U14, U15, U16, L8, L12, L16) = (s_1, s_2 s_3, s_4, s_5, s_6, s_7)$. The SMD between a block X,predicted by blocks U and L, and a codeword cw in the codebook is measured by the squared Euclidean distance denoted as follows:

$$\mathrm{SMD}\,(X, \mathrm{cw}) = D\,(\mathrm{S}\,(\mathrm{X, U, L})\,, \mathrm{B}\,(\mathrm{cw})) = \sum_{i=1}^{7} \left(s_i - b_i\right)^2. \quad (2)$$

For each block, SMVQ sorts the codebook CB according to SMDs, and thus the first F_c codewords in the sorted

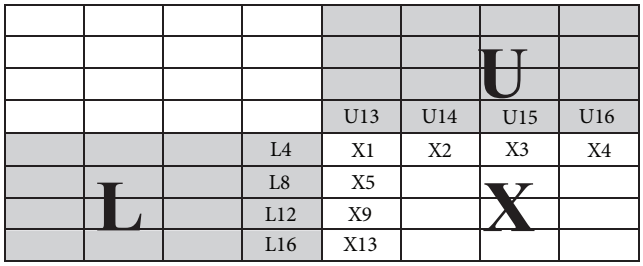


Figure 1: An example of SMVQ.

codebook are picked to form the state codebook SC, in which SMDs are the smallest for block X. Subsequently, the SMVQ searches the closest codeword from the state codebook for block X, and then the index of the closest codeword is used to encode block X. Because the size of the state codebook is smaller than that of the VQ codebook, the size of the SMVQ index is reduced to improve the coding gain. However, there is a distortion between the block, decoded by the SMVQ index, and the corresponding block, decoded by the VQ index.

3. The Proposed Scheme

A grayscale cover image C of size $N * M$ is partitioned into nonoverlapped blocks $\{\mathrm{BLK_i}\}_{i=1,\ldots,L}$ of $w * h$pixels, in which $L = (N/w) * (M/h)$. Normally, the variables $w = h = 4$. Each block of the image can be encoded using VQ, and thus each block is represented with an index of the nearest codeword in the codebook. A VQ-compressed image, denoted by I, consists of indices. The size of an index is proportional to the number of codewords in the codebook. A SMVQ-compressed image is generated by applying SMVQ predictions to a VQ-compressed image. To reconstruct a VQ-compressed image completely, the size of the state codebook and that of the VQ codebook are first set as identical. Next, the codebook is sorted according to the SMDs which denote the differences between an encoding block and the codewords in the VQ codebook. Finally, the state codebook is generated using all of the codewords in the sorted codebook. For a VQ index, the corresponding SMVQ index can be obtained using the state codebook and no distortion is presented between the two codewords decoded by the two indices.

Figures 2 and 3 show histograms of the VQ-compressed Peppers image and the corresponding SMVQ-compressed image, respectively, with a codebook of size 256. Based on the SMVQ predication, most indices of the SMVQ-compressed image are distributed close to zero. For a SMVQ-compressed image, a very small value, N_z binary digits (N_z bits) representing unsigned integers from 0 to $2^{Nz} - 1$, can cover most of the indices. In the proposed scheme, the 2^{N_z} numbers are employed as indicators or/and encoding codes. Table 1 shows the functions of each number of the N_z binary digits (N_z bits). A binary number among 0 and $2^{Nz} - 3$ represents an encoding code and indicator. Binary numbers

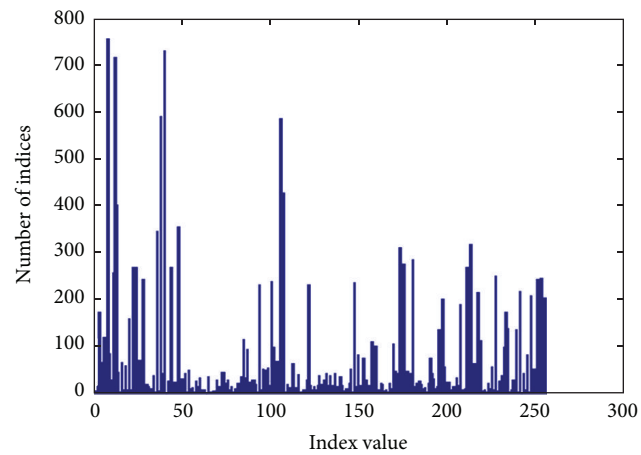


Figure 2: Histogram of the VQ-compressed Peppers image.

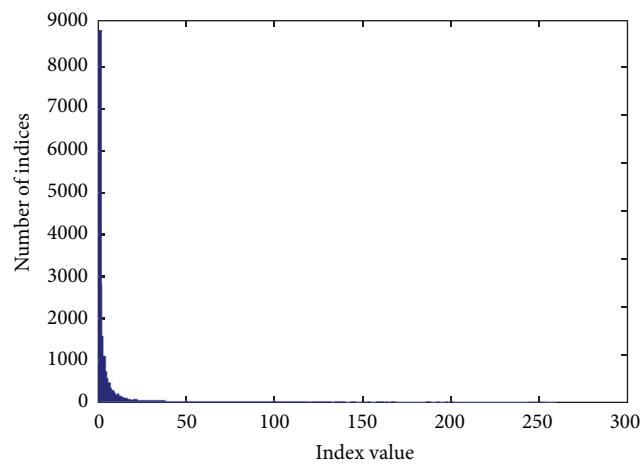


Figure 3: Histogram of the SMVQ-compressed Peppers image.

Table 1: The functions of each number of the N_z binary digits (N_z bits).

Number	Function
$0 \sim 2^{N_z} - 3$	Indicator and encoding code
$2^{N_z} - 2$	Indicator (*indicator 0*)
$2^{N_z} - 1$	Indicator (*indicator 1*)

$2^{Nz} - 2$ and $2^{Nz} - 1$ represent indicators, denoted by *indicator 0* and *indicator 1*, respectively.

3.1. The Encoding Process. Suppose the size of a secret data, sd, is s bits. During the encoding process, a SMVQ index with a value less than $2^{Nz} - 2$, which is defined as a tiny-value index, is denoted by E_i, and the corresponding encoded index is denoted by E_i'. Then, E_i' is expressed in (3), where $\wedge$ and $\|$ denote exponential and concatenation operators, respectively,

$$E_i' = E_i \,\|\, \mathrm{sd}\,. \tag{3}$$

Table 2: The ranges and symbols of tiny-value, middle-value, and large-value indices.

Indices	Range	Symbol
Tiny-value indices	$0 \sim 2^{N_z} - 3$	E_i
Middle-value indices	$0 + 2^{N_z} - 2 \sim 2^{N_c} - 1 + 2^{N_z} - 2$	G_i
Large-value indices	$2^{N_c} + 2^{N_z} - 2 \sim \text{cs} - 1$	O_i

Most of the SMVQ indices can be encoded using (3) including no indicator, and the rest are encoded with the help of *indicator 0* or *indicator 1*. However, the adoption of the indicator increases the size of the encoded result. A SMVQ index with a value in between $(2^{Nz} - 2) + 0$ and $(2^{Nz} - 2) + 2^{Nc} - 1$ is defined as a middle-value index. Indices with middle values are encoded using *indicator 0* concatenated with an encoding space with N_c bits. A middle-value index is denoted by G_i, and the corresponding encoded index is denoted by G_i' which can be expressed in the following:

$$G_i' = indicator\ 0 \parallel G_i - 2^{Nz} + 2. \tag{4}$$

The N_c-bit encoding space of (4) is not to hide data but to collect more indices. *Indicator 1* is employed to assist encoding a SMVQ index which cannot be encoded using (3) or (4). A SMVQ index with a value more than $(2^{Nz} - 2) + 2^{Nc} - 1$ is defined as a large-value index, denoted by O_i, and the corresponding encoded index is denoted by O_i' which can be expressed in the following:

$$O_i' = indicator\ 1 \parallel O_i. \tag{5}$$

Equation (5) not only hides no data but also increases an additional indicator with N_z bits. For a SMVQ-compressed image, most indices with tiny values are encoded using (3), which hides s bits and generates an encoded result with $N_z + s$ bits for each index; a few indices are encoded using (4), which generates an encoded result with $N_z + N_c$ bits for each index; few indices are encoded using (5), which generates an encoded result with $N_z + \log_2(\text{cs})$ bits for each index, where cs indicates the size of a codebook. Table 2 shows the ranges and symbols of tiny-value, middle-value, and large-value indices. The size of an embedded image can be measured with (6), where $\|x\|$ denotes the size of x andI_z, I_c, and I_o denote the number of indices with tiny values, middle values, or large values, respectively:

$$\begin{aligned} \|\text{embedded image}\| = {} & (N_z + s) * I_z \\ & + (N_z + N_c) * I_c \\ & + (N_z + \log_2(\text{cs})) * I_o. \end{aligned} \tag{6}$$

In (6), $(N_z + s)$ and $(N_z + N_c)$ can be controlled in less or more than $\log_2(\text{cs})$ bits and only the third item always requires a space with more than $\log_2(\text{cs})$ bits. It is remarkable that a larger s leads to a larger embedding capacity, and vice versa. Moreover, a larger N_c increases I_c, while decreasing I_o, and vice versa. Based on the special index distribution property of the SMVQ, most indices can be encoded using (3), in which no indicator is required and s-bit data are hidden in each index, and thus a high embedding capacity and a low embedded image size can be achieved in the encoding process. The algorithm of the embedding process is summarized as follows and an example of the index classification and encoding schemes is shown in Figure 4.

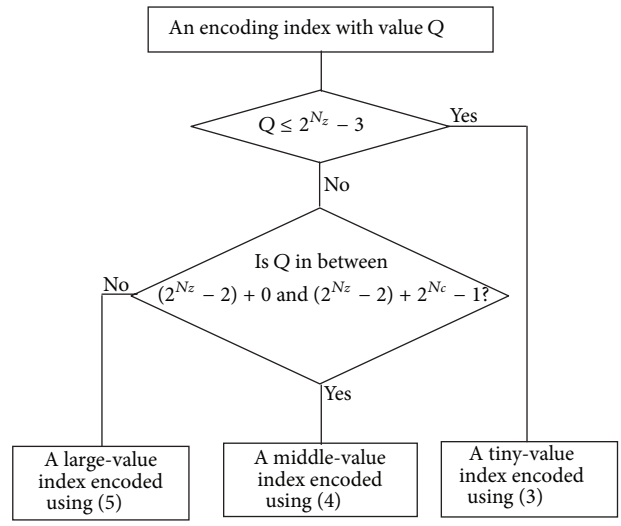


Figure 4: An example of the index classification and encoding schemes.

Input. A grayscale cover image C of size $N * M$, a codebook CB of size cs, a secret data stream, and parameters N_z and N_c.

Output. The code stream in binary form cs_b and parameters N_z and N_c.

Step 1. Compress the cover image C using VQ to obtain the VQ-compressed image I of size $(N/w) * (M/h)$.

Step 2. Perform SMVQ predictions with the VQ-compressed image I to create the SMVQ-compressed image T of size $(N/w) * (M/h)$.

Step 3. Read the next SMVQ index, denoted as Q, from the SMVQ-compressed image T with the raster scan order.

Step 4. If $Q \le 2^{N_z} - 3$ (a tiny-value index), then we have the following.

Step 4.1. Read s-bit data, sd, from the secret data stream,

Step 4.2. Q is encoded using (3), cs_b = cs_b $\parallel$ D2B(Q) $\parallel$ sd, where $\parallel$ denotes a concatenation operation, and D2B denotes a decimal to binary operation.

Step 5. If Q is in between $(2^{N_z} - 2) + 0$ and $(2^{N_z} - 2) + 2^{N_c} - 1$ (a middle-value index), then Q is encoded using (4), cs_b = cs_b $\parallel$ *indicator* 0 $\parallel$ D2B($Q - 2^{N_z} + 2$).

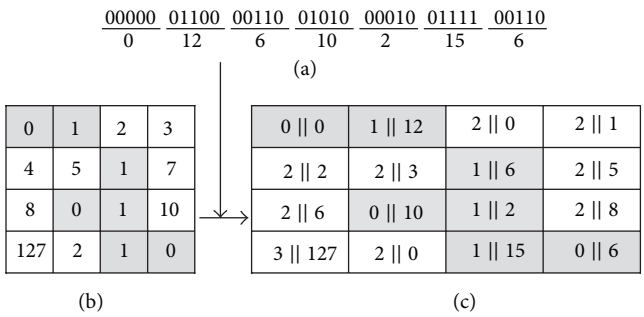


Figure 5: An example of the encoding process for hiding 5-bit secret data. (a) Secret data, (b) SMVQ-compressed image, and (c) encoded image.

Step 6.

If $Q \geq (2^{N_z} - 2) + 2^{N_c}$ (a large-value index), then

Q is encoded using (5),

$\text{cs\_}b = \text{cs\_}b \parallel \textit{indicator } 1 \parallel \text{D2B}(Q)$.

Step 7. Repeat Steps 3–7 until all of the indices in the SMVQ-compressed image are processed.

Step 8. Output the code stream cs_b and parameters N_z and N_c.

Figure 5 shows an example of the encoding process for hiding 5-bit secret data. Let I_0 be *indicator 0* and I_1 *indicator 1*, where $N_z = 2$, $N_c = 5$, $s = 5$, and $cs = 128$, and the size of the state codebook is identical to that of the VQ codebook. For each image (Figure 5), let R_1 to R_4 be the indices located in the first row and R_5 to R_8 the indices located in the second row, and so on. For the SMVQ-compressed image, R_1 has a value 0, and it is less than $2^{Nz} - 2$. The index is encoded using (3). Secret data are $(00000)_2$. According to (3), the index with N_z bits followed by secret data $(00000)_2$ is presented as the encoded result for R_1. R_4 has a value of 3, and it is in between $(2^{Nz} - 2) + 0$ and $(2^{Nz} - 2) + 2^{Nc} - 1$. An indicator I_0, 2, followed by the offset index, 1, is the encoded result for R_4. R_{13} has a value of 127 and it is more than $(2^{Nz} - 2) + 2^{Nc} - 1$. An indicator I_1, 3, followed by the index, 127, is the encoded result for R_{13}. All of the encoded results are in binary form, and the length of the encoded result of each index is $(N_z + s)$, $(N_z + N_c)$ or $(N_z + \log_2(\text{cs}))$ bits using (3), (4), or (5), respectively.

3.2. The Extraction and Reversion Process. The goal of the data extraction and reversion process is to extract the embedded secret data from an embedded image and then recover the original VQ-compressed image. The extraction and reversion process is introduced as below. For an encoded index, N_z bits are firstly read from code stream cs_b and the N_z-bit binary value is converted to the decimal value, denoted by d_N_z. In the first case, if d_N_z is less than $2^{Nz} - 2$, the encoded index is an E_i index. Then, the original index is directly restored by d_N_z. Next, read s bits from the code stream c_bs to reconstruct the secret data. In the second case, if d_N_z is $2^{Nz} - 1$, the encoded index is an O_i index. Then, read the next $\log_2(\text{cs})$ bits and convert them into decimal value, denoted by d_O_i; the original index is recovered by d_O_i. In the third case, if d_N_z is $2^{Nz} - 2$, the encoded index is a G_i index. Then, read the next N_c bits and convert them into decimal value d_G_i; the original index is recovered by $d_{G_i} + (2^{N_z} - 2)$. When the original SMVQ-compressed image has been reconstructed, SMVQ predictions can be employed to restore the original VQ-compressed image. In summary, the extraction and reversion process of the proposed scheme is organized as below.

Input. The code stream in binary form cs_b and parameters N_z and N_c.

Output. The reconstructed VQ-compressed image I' of size $N * M$ and the retrieved secret data stream.

Step 1. Set ptr = 1 and the restored sd = Null.

Step 2. Read the next N_z bits from the code stream c_bs and convert theN_z bits to decimal value, denoted by d_N_z, $d\_N_z = \text{B2D}(\text{cs\_}b(\text{ptr} : \text{ptr} + N_z - 1)$, $\text{ptr} = \text{ptr} + N_z$.

Step 3. If $d\_N_z < 2^{Nz} - 2$ (a tiny-value index), then

restored index = d_N_z;

read the next s bits, denoted by sd$'$, from code stream c_bs,

restored sd = restored sd $\parallel$ sd$'$,

ptr = ptr + s.

Step 4. If $d\_N_z = 2^{Nz} - 1$ (*indicator 1*), then

read the next $\log_2(\text{cs})$ bits from the code stream c_bs,

$d\_O_i = \text{B2D}(\text{cs\_}b(\text{ptr} : \text{ptr} + \log_2(\text{cs}) - 1), \text{ptr} = \text{ptr} + \log_2(\text{cs})$,

restored index = d_O_i.

Step 5. If $d\_N_z = 2^{Nz} - 2$ (*indicator 0*), then

read the next N_c bits from the code stream c_bs,

$d\_G_i = \text{B2D}(\text{cs\_}b(\text{ptr} : \text{ptr} + N_c - 1), \text{ptr} = \text{ptr} + N_c$,

restored index = $d\_G_i + (2^{Nz} - 2)$.

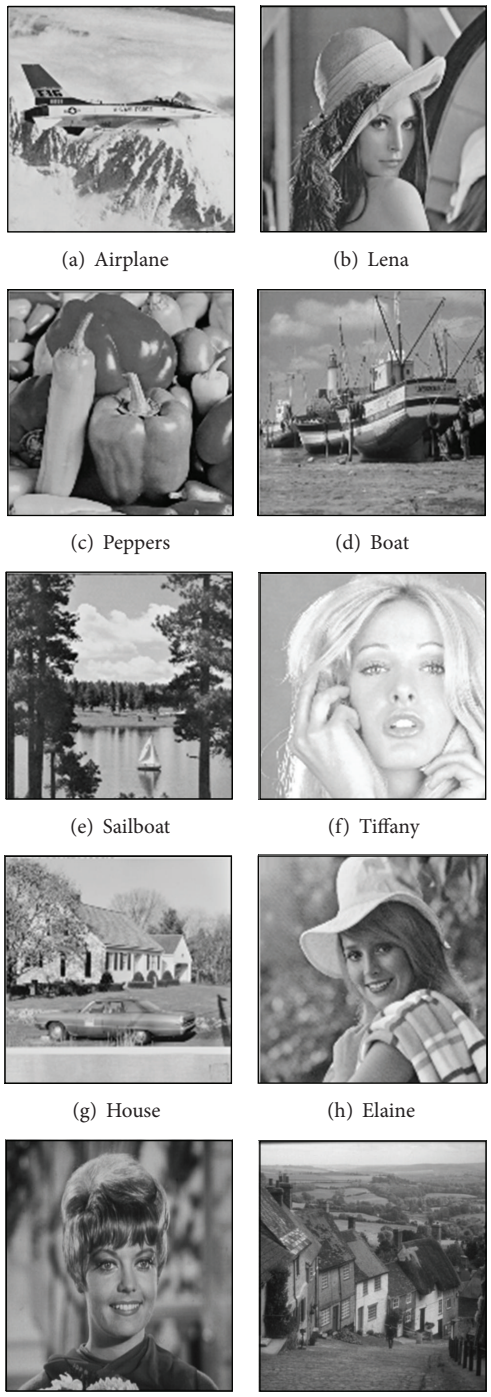
Step 6. Repeat Steps 2–6 until all of the bits in the code stream are processed.

Step 7. Apply the SMVQ prediction to the reconstructed SMVQ-compressed image to obtain the original VQ-compressed image.

Step 8. Output the reconstructed VQ-compressed image and the restored secret data stream.

4. Experimental Results

In this section, we present the experimental results for evaluating the performance of the proposed scheme in terms



(a) Airplane (b) Lena (c) Peppers (d) Boat (e) Sailboat (f) Tiffany (g) House (h) Elaine (i) Zelda (j) Gold hill

Figure 6: Test images.

of the embedding rate, bit rate, and embedding capacity. Various gray-level images as shown in Figure 6 are employed as the training images or cover images, each of which is of size $512 * 512$. The codebooks of sizes 128, 256, 512, and 1024 with codewords of 16 dimensions were trained by the LBG algorithm with "Airplane," "Boat," "Lena," "Peppers," and "Sailboat" as the training images.

The secret data in the experiment is in binary format, 0 and 1, and which are generated by a pseudorandom number generator. If a high-level security is required, secret data can be encrypted prior to the embedding using the well-known cryptographic methods such as DES or RSA. In this work, the embedding rate (ER), bit rate (BR), and embedding capacity (EC) as defined in (7)–(9), respectively, are employed to evaluate the performance of various schemes. In general, a data-hiding method with a small value in BR and large values in EC and ER endorses a good performance, and vice versa:

$$\mathrm{BR} = \frac{\|\text{embedded image}\|}{N * M}, \tag{7}$$

$$\mathrm{EC} = s * I_z, \tag{8}$$

$$\mathrm{ER} = \frac{\mathrm{EC}}{\|\text{embedded image}\|}. \tag{9}$$

4.1. Experiments on Selecting the Appropriate Parameter N_z. To obtain the best performance of the proposed scheme, the parameter N_z is needed to be properly determined. As mentioned, when N_z and s are set to a larger value, I_z, EC, and BR are increased. The encoding length of a VQ index is $\log_2(\mathrm{cs})$ bits. A VQ index encoded as an E_i or G_i index yields an additional space with $\log_2(\mathrm{cs}) - N_z$ bits, the space can be used to hide secret data or collect more indices. Experiments are performed to choose an appropriate N_z value which can achieve the highest embedding rate for different codebooks and different images. First of all, s and N_c are set to $\log_2(\mathrm{cs}) - N_z$. Figure 7 shows the performance of the proposed algorithm using different N_z, codebook sizes, and test images. For codebooks of sizes 128 and 256, SMVQ provides accurate prediction, and the most of the VQ indices are predicted by the codewords with smaller indices in the SMVQ state codebook. In these cases, the highest ERs can be obtained when N_z is set to a small value 3. For codebooks of sizes 512 and 1024, N_z must be set at greater values, 4 or 5, to obtain a high ER. For each image, larger ERs appear as N_z with values of 3, 3, 4, and 5 for codebooks of sizes 128, 256, 512, and 1024, respectively.

The experiments are performed based on the selected various N_z with the four codebooks of sizes 128, 256, 512, and 1024. Experimental results for the test images are shown in Table 3, in which different columns of the simulation results are obtained with (cs, N_z, N_c, s) = (128, 3, 4, 4), (256, 3, 5, 5), (512, 4, 5, 5), and (1024, 5, 5, 5), respectively. From Table 3, it can be seen that the BRs of the proposed scheme are only somewhat larger than that of the original VQ BRs and the embedding capacities of the proposed scheme are very high. Moreover, for most images the high embedding capacities are still obtained when the sizes of codebooks are increased. The smooth image, such as "Tiffany," has a high embedding capacity than that of the complicated image, such as "Boat," since the SMVQ prediction normally presents good performance for a smooth image, thus many indices are encoded with E_i indices. With the proposed method, when the codebooks are of sizes 128, 256, 512, and 1024, the average ECs are 3.246, 3.637, 3.848, and 3.872 bits/index, respectively.

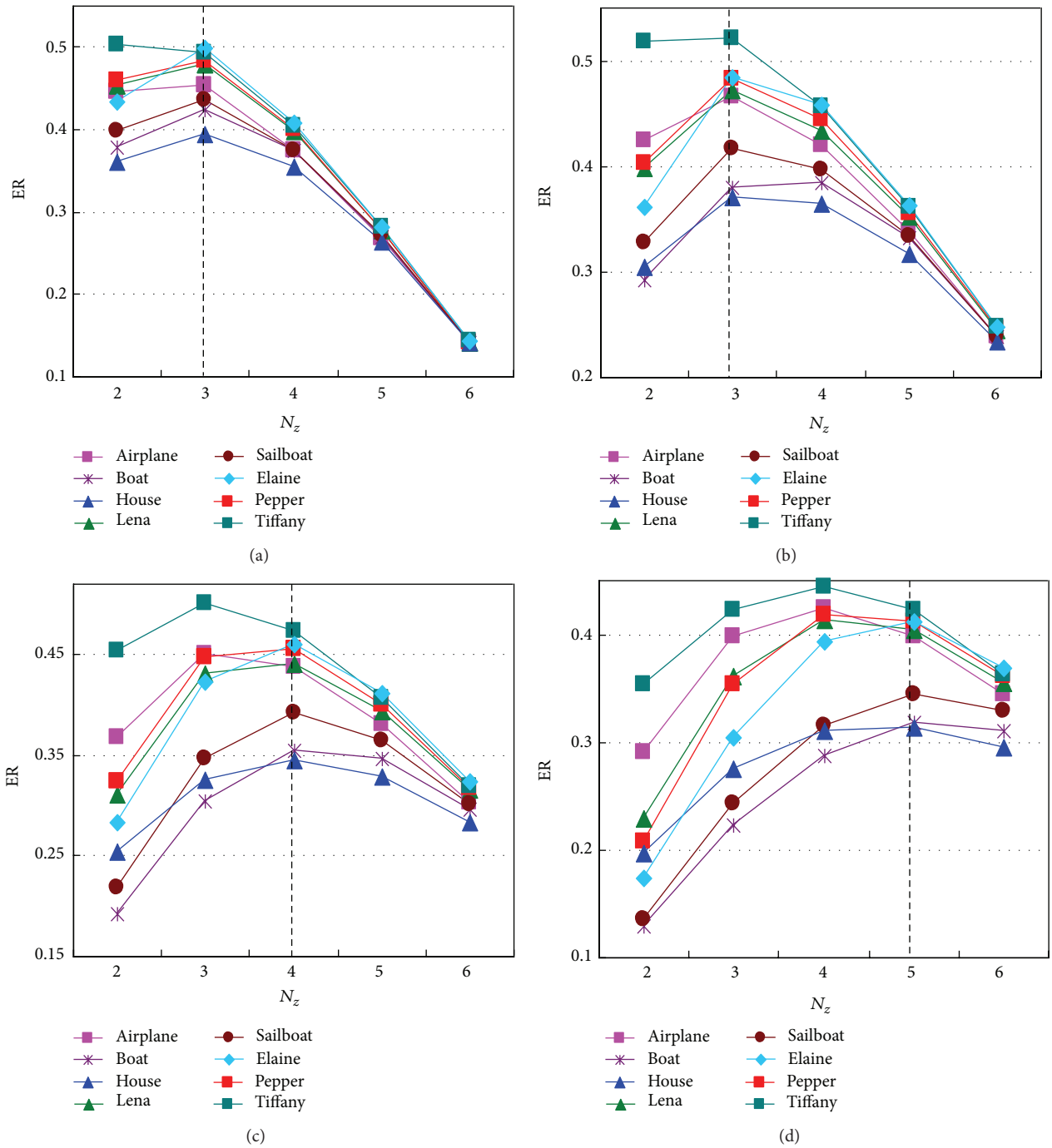


FIGURE 7: The performance of the proposed algorithm using different N_z, different test images, and different codebook sizes: (a) codebook size 128, (b) codebook size 256, (c) codebook size 512, and (d) codebook size 1024.

In addition, for applications of secret communication, s can be set to a value smaller (larger) than $\log_2(cs)$ to obtain a lower (higher) embedding capacity, and consequently the corresponding bit rate can be changed as well.

4.2. Experiments on Comparing the Proposed Method with Other Methods. To demonstrate the superiority of the proposed algorithm, it is compared with the VQ-based scheme proposed by Chang et al. [10], the VQ/SMVQ-based scheme proposed by Yang and Lin [15], and SOC-based scheme proposed by Shie and Lin [12] as detailed below. To ensure all of them can be fairly compared, the proposed scheme is performed with the N_z, N_c, and s values are set to obtain an embedding capacity close to that of the other schemes, and the codebook size for each comparison is set as identical.

The results on the left and the middle of Table 4 show the ERs, BRs, and ECs of the schemes developed by Chang et al. [10] and Yang and Lin [15], respectively. The results

Table 3: Simulation results for various test images and codebook sizes.

Images	Codebook sizes/VQ *BR* (in bpp)			
	128/0.438	256/0.50	512/0.563	1024/0.625
	N_z/s			
	3/4	3/5	4/5	5/5
	ER			
	BR (in bpp)			
	EC (in bits)			
Airplane	0.454	0.466	0.438	0.398
	0.446	0.509	0.574	0.640
	53144	62095	65905	66785
Lena	0.478	0.471	0.441	0.403
	0.442	0.505	0.570	0.637
	55396	62350	65955	67390
Pepper	0.483	0.483	0.456	0.412
	0.442	0.504	0.570	0.636
	55928	63810	68030	68685
Boat	0.423	0.380	0.354	0.318
	0.446	0.509	0.579	0.650
	49432	50775	53720	54155
Sailboat	0.435	0.416	0.391	0.344
	0.446	0.509	0.576	0.645
	50792	55515	59120	58110
Tiffany	0.492	0.521	0.473	0.422
	0.441	0.503	0.569	0.635
	56860	68625	70475	70220
House	0.394	0.370	0.345	0.314
	0.449	0.513	0.584	0.654
	46392	49780	52745	53840
Elaine	0.498	0.484	0.460	0.412
	0.440	0.503	0.568	0.634
	57492	63775	68445	68385

on the right of Table 4 show the ERs, BRs, and ECs of the proposed scheme. In Table 4, Chang et al.'s scheme is VQ-based, Yang and Lin's scheme is VQ/SMVQ-based, and the proposed scheme is SMVQ-based. First of all, N_z, N_c, and s of the proposed scheme are set to obtain an embedding capacity close to that of the other schemes. Thus, when the codebooks are of sizes 128, 256, 512, and 1024, (N_z, s, N_c) are set at (2, 3, $\log_2(\text{cs}) - N_z$), (3,3, $\log_2(\text{cs}) - N_z$), (3,3, $\log_2(\text{cs}) - N_z$), and (4,3, $\log_2(\text{cs}) - N_z$), respectively. SMVQ can relocate the histogram of the VQ indices to locations near zero. In the proposed scheme, the indices with values close to zero are encoded using short codes without any indicator, while the indices of values away from zero are encoded using the original index with an indicator. Based on the characteristic of the SMVQ, most of the indices are with small values, and only a few indices have large values. Thus, this property leads to low bit rate and high embedding capacity with the proposed scheme. In Chang et al.'s [10] scheme, when the number of embeddable indices is increased, the embedding capacity of each index is decreased, and vice versa. The satisfactory EC cannot yield. Moreover, most indices are encoded by an indicator concatenated with the encoding code. The length of an indicator or an encoding code is identical to that of the original VQ index. The bit rate of an embedded image is thus increased. In Yang and Lin's [15] scheme, a 1-bit indicator is always used to distinguish a SMVQ encoding from a VQ encoding and other indicators are required for VQ encoding, leading to a high bit rate result. Compared with the other schemes, the proposed scheme has low bit rate yet high embedding capacity and embedding rate. In particular, when the embedding capacities of the proposed scheme are controlled to values close to that of the other schemes, the proposed scheme presents much lower BRs, even smaller than that of the original VQ BRs, for all of the test images.

Table 4: Performance comparisons among Chang et al.'s [10] scheme, Yang and Lin's [15] scheme, and the proposed scheme with various test images and codebook sizes.

Images	Chang et al.'s [10] scheme				Yang and Lin's [15] scheme		The proposed scheme			
	Codebook sizes						Codebook sizes/VQ *BR*			
	128	256	512	1024	256	512	128/0.438	256/0.500	512/0.563	1024/0.625
							N_z/s			
							2/3	3/3	3/3	4/3
							ER			
							BR (in bpp)			
							EC (in bits)			
Airplane	0.182	0.149	0.124	0.119	0.236	0.208	0.326	0.343	0.291	0.269
	0.580	0.670	0.760	0.850	0.500	0.550	0.364	0.414	0.442	0.503
	27684	26170	24608	26432	30894	29954	31077	37257	33747	35448
Lena	0.179	0.136	0.115	0.094	0.234	0.208	0.332	0.348	0.275	0.261
	0.580	0.670	0.750	0.830	0.500	0.550	0.360	0.410	0.445	0.504
	27164	23884	22626	20502	30736	29958	31308	37410	32067	34428
Peppers	0.187	0.150	0.116	0.104	0.244	0.216	0.338	0.359	0.287	0.264
	0.580	0.670	0.740	0.830	0.490	0.540	0.359	0.407	0.440	0.501
	28372	26284	22592	22676	31346	30576	31743	38286	33180	34677
Toys	0.195	0.160	0.136	0.119	0.254	0.234	0.397	0.373	0.332	0.302
	0.590	0.680	0.770	0.860	0.480	0.510	0.348	0.405	0.426	0.487
	30202	28606	27460	26760	31912	31226	36300	39594	37107	38529
Gold hill	0.142	0.109	0.084	0.074	0.203	0.158	0.306	0.321	0.248	0.232
	0.590	0.680	0.750	0.830	0.540	0.640	0.364	0.415	0.454	0.514
	21928	19420	16548	16116	28792	26502	29211	34866	29451	31287
Zelda	0.179	0.135	0.111	0.088	0.250	0.220	0.338	0.375	0.308	0.298
	0.580	0.670	0.740	0.820	0.480	0.530	0.357	0.401	0.431	0.485
	27258	23754	21506	18842	31444	30634	31704	39462	34809	37842

The results on the left of Table 5 show the ERs, BRs, and ECs of the SOC-based scheme with codebooks of sizes 256 and 512, respectively, developed by Shie and Lin [12]. The results on the right of the Table 5 show the ERs, BRs, and ECs of the proposed scheme with codebooks of sizes 256 and 512, respectively. In Shie and Lin's [12] scheme, a 1-bit indicator is always used to distinguish a SOC index from an OIV index, in which a SOC is encoded using $1 + d$ bits, while an OIV index is encoded using $1 + \log_2(\text{cs})$ bits. The variable d indicates the length of a search path, and which is set to 2 to yield a better performance. The proposed scheme encodes E_i and G_i indices using N_z and $\log_2(\text{cs}) - N_z$ bits, where N_z is set to 3 to obtain embedding capacities close to that of Shie and Lin's method. The length of a VQ index is $\log_2(\text{cs})$, in which N_z or $1 + d$ bits are used to encode an E_i index or a SOC index, and the remaining bits can be used to hide secret data. In Table 5, first of all, the encoded result for each E_i or SOC index of the two schemes is set to $\log_2(\text{cs})$ bits, which yields a hiding space of $\log_2(\text{cs}) - 1 - 2$ bits for a SOC index and $\log_2(\text{cs}) - N_z$ bits for a E_i index. For each test image with a specific codebook size, the bold-number columns in Table 5 show that the EC values of the proposed scheme are significantly greater than that of Shie and Lin's scheme, while lower BR values and higher ER values can be achieved with the proposed scheme.

Next, the hiding space of Shie and Lin's [12] scheme is set to $\log_2(\text{cs})$ bits, which can achieve a high embedding space, and the length of the encoded result for a SOC index is $1 + 2 + \log_2(\text{cs})$ bits. Because the proposed scheme has good performance in terms of ER, BR, and EC, N_z, N_c, and the hiding space of the proposed scheme is set to 3, $\log_2(\text{cs}) - N_z$ and $\log_2(\text{cs}) - 1$ bits, the set can offer an EC close to that of Shie and Lin's scheme and decrease the length of the encoded result of an E_i index to $3 + \log_2(\text{cs}) - 1$ bits. For each codebook size and test image, the unmarked columns in Table 5 show that the EC values of the proposed scheme are still much greater than that of Shie and Lin's scheme under the high hiding space, while the proposed scheme can preserve low BR values and high ER values. Notably, the "Zelda" and "Gold hill" images are not the training members for proper N_z selection. Yet it till can achieve superior performances.

As mentioned, the proposed scheme outperforms the VQ-based scheme proposed by Chang et al. [10], the VQ/SMVQ-based scheme proposed by Yang and Lin [15],

Table 5: Performance comparisons between Shie and Lin's [12] scheme and the proposed scheme with various test images and codebook sizes.

Images	Shie and Lin's [12] scheme				The proposed scheme			
	Codebook sizes/VQ *BR*							
	256/0.500	512/0.563	256/0.500	512/0.563	256/0.500	512/0.563	256/0.500	512/0.563
	d/*s*/encoded bits				N_z/*s*/encoded bits			
	2/5/8	2/6/9	2/8/11	2/9/12	3/5/8	3/6/9	3/7/10	3/8/11
	ER							
	BR (in bpp)							
	EC (in bits)							
Airplane	**0.383**	**0.342**	0.499	0.438	**0.466**	**0.451**	0.550	0.522
	0.522	**0.591**	0.643	0.692	**0.509**	**0.571**	0.603	0.657
	52500	**52998**	84000	79497	**62095**	**67494**	86933	89992
Lena	**0.371**	**0.308**	0.485	0.400	**0.471**	**0.431**	0.555	0.503
	0.524	**0.594**	0.640	0.686	**0.505**	**0.567**	0.600	0.649
	50915	**48000**	81464	72000	**62350**	**64134**	87290	85512
Peppers	**0.369**	**0.298**	0.484	0.389	**0.483**	**0.447**	0.567	0.518
	0.524	**0.595**	0.640	0.684	**0.504**	**0.567**	0.601	0.651
	50735	**46506**	81176	69759	**63810**	**66360**	89334	88480
Boat	**0.272**	**0.194**	0.374	0.265	**0.380**	**0.305**	0.462	0.369
	0.533	**0.605**	0.621	0.664	**0.509**	**0.573**	0.587	0.631
	38015	**30744**	60824	46116	**50775**	**45756**	71085	61008
Sailboat	**0.296**	**0.214**	0.402	0.289	**0.416**	**0.347**	0.499	0.414
	0.531	**0.604**	0.625	0.668	**0.509**	**0.571**	0.594	0.638
	41235	**33792**	65976	50688	**55515**	**51942**	77721	69256
Gold hill	**0.338**	**0.290**	0.450	0.380	**0.440**	**0.397**	0.524	0.467
	0.527	**0.596**	0.634	0.683	**0.503**	**0.566**	0.592	0.641
	46695	**45378**	74712	68067	**58110**	**58902**	81354	78536
Zelda	**0.361**	**0.304**	0.475	0.395	**0.500**	**0.471**	0.584	0.543
	0.525	**0.595**	0.638	0.685	**0.502**	**0.564**	0.602	0.653
	49665	**47340**	79464	71010	**65770**	**69618**	92078	92824

and SOC-based scheme proposed by Shie and Lin [12] in three aspects, that is, embedding rate, bit rate, and embedding capacity. Thus, the proposed method can be considered as effective coding candidate for reversible data hiding application.

5. Conclusions

In this paper, a novel reversible data-hiding scheme for SMVQ-compressed images is proposed. In most VQ/SMVQ hiding schemes, the bit rate is increased when an indicator is required to encode an index. The proposed scheme applies the SMVQ prediction to relocate most of the indices of a VQ-compressed image to locations close to zero. Consequently, these small-value frequently used indices are encoded using fewer bits, and no indicator is required, which thus increases the embedding capacity and decreases the bit rate. The experimental results demonstrate that the proposed scheme can completely extract the secret data and recover the original VQ-compressed image. Moreover, comparing to other schemes, the proposed scheme can achieve better efficiency under each codebook size for various test images.

Acknowledgment

The work was supported by National Science Council, Republic of China, Taiwan, under Grant NSC100-2221-E-182-047-MY3.

References

[1] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 890–896, 2003.

[2] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 354–361, 2006.

[3] Y. Hu, H.-K. Lee, and J. Li, "DE-based reversible data hiding with improved overflow location map," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 2, pp. 250–260, 2009.

[4] C. H. Lin and C. Y. Yang, "Multipurpose watermarking based on blind vector quantization (BVQ)," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 2, no. 2, pp. 239–246, 2011.

[5] S. Weng, S. C. Chu, N. Cai, and R. Zhan, "Invariability of mean value based reversible watermarking," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 4, no. 2, pp. 90–98, 2013.

[6] Y. Wang, J. F. Doherty, and R. E. Van Dyck, "A wavelet-based watermarking algorithm for ownership verification of digital images," *IEEE Transactions on Image Processing*, vol. 11, no. 2, pp. 77–88, 2002.

[7] J.-D. Lee, Y.-H. Chiou, and J.-M. Guo, "Reversible data hiding based on histogram modification of SMVQ indices," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 638–648, 2010.

[8] C. Y. Yang, C. H. Lin, and W. C. Hu, "Reversible data hiding for high-quality images based on integer wavelet transform," *Journal of Information Hiding and Multimedia Signal Processing*, vol. 3, no. 2, pp. 142–150, 2012.

[9] M. Jo and H. Kim, "A digital image watermarking scheme based on vector quantisation," *IEICE Transactions on Information and Systems*, vol. E85-D, no. 6, pp. 1054–1056, 2002.

[10] C.-C. Chang, W.-C. Wu, and Y.-C. Hu, "Lossless recovery of a VQ index table with embedded secret data," *Journal of Visual Communication and Image Representation*, vol. 18, no. 3, pp. 207–216, 2007.

[11] C.-C. Chang, G.-M. Chen, and M.-H. Lin, "Information hiding based on search-order coding for VQ indices," *Pattern Recognition Letters*, vol. 25, no. 11, pp. 1253–1261, 2004.

[12] S.-C. Shie and S. D. Lin, "Data hiding based on compressed VQ indices of images," *Computer Standards and Interfaces*, vol. 31, no. 6, pp. 1143–1149, 2009.

[13] S.-C. Shie, S. D. Lin, and C.-M. Fang, "Adaptive data hiding based on SMVQ prediction," *IEICE Transactions on Information and Systems*, vol. E89-D, no. 1, pp. 358–362, 2006.

[14] C.-C. Chen and C.-C. Chang, "High capacity SMVQ-based hiding scheme using adaptive index," *Signal Processing*, vol. 90, no. 7, pp. 2141–2149, 2010.

[15] C.-H. Yang and Y.-C. Lin, "Reversible data hiding of a VQ index table based on referred counts," *Journal of Visual Communication and Image Representation*, vol. 20, no. 6, pp. 399–407, 2009.

[16] Y. Linde, A. Buzo, and R. M. Gray, "Algorithm for vector quantizer design," *IEEE Transactions on Communications Systems*, vol. 28, no. 1, pp. 84–95, 1980.

[17] T. Kim, "Side match and overlap match vector quantizers for images," *IEEE Transactions of Image Processing*, vol. 1, no. 2, pp. 170–185, 1992.