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

Copyright © 2013 Jiann-Der Lee et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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 extra hiding 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 indicator 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 256 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 indicator 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 \( j \)th codeword in the codebook, the squared Euclidean distance is generally used as below:

\[
D(B, C_j) = \sum_{i=1}^{k} (b_i - c_{ji})^2. \tag{1}
\]

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 x 4. This work denotes the border vector and the side vector of block \( X \) as \( B(X) = (X_1, X_2, X_3, X_4, X_5, X_9, X_{13}) = (b_1, b_2, b_3, b_4, b_8, b_9) \) and \( S(X, U, L) = (U_{13} + L_4)/2, U_{14}, U_{15}, U_{16}, L_8, L_{12}, L_{16} = (s_1, s_2, s_3, s_4, s_5, s_6, s_7) \). 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:

\[
\text{SMD}(X, cw) = D(S(X, U, L), B(cw)) = \sum_{i=1}^{7} (s_i - b_i)^2. \tag{2}
\]

For each block, SMVQ sorts the codebook \( CB \) according to SMDs, and thus the first \( F_c \) codewords in the sorted
3. The Proposed Scheme

A grayscale cover image $C$ of size $N \times M$ is partitioned into nonoverlapped blocks $\{\text{BLK}_i\}_{i=1,...,L}$ of $w \times h$ pixels, in which $L = (N/w) \times (M/h)$. 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. 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.

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 $2^{N_z} - 2$, 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 $\land$ and $\parallel$ denote exponential and concatenation operators, respectively.

$$E'_i = E_i \parallel sd.$$  

(3)
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 \( (2^{N_c} - 2) + 0 \) and \( (2^{N_c} - 2) + 2^{N_c} - 1 \) 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:

\[
G_i' = \text{indicator 0} \oplus G_i - 2^{N_c} + 2.
\]  

(4)

The \( N_z \)-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 \( (2^{N_c} - 2) + 2^{N_c} - 1 \) 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:

\[
O_i' = \text{indicator 1} \oplus O_i.
\]  

(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 \( N_z + \log_2(cs) \) 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.

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

<table>
<thead>
<tr>
<th>Indices</th>
<th>Range</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tiny-value indices</td>
<td>( 0 - 2^{N_z} - 3 )</td>
<td>( E_i )</td>
</tr>
<tr>
<td>Middle-value indices</td>
<td>( 0 + 2^{N_z} - 2 - 2^{N_c} - 1 + 2^{N_z} - 2 )</td>
<td>( G_i )</td>
</tr>
<tr>
<td>Large-value indices</td>
<td>( 2^{N_z} + 2^{N_c} - 2 - cs - 1 )</td>
<td>( O_i )</td>
</tr>
</tbody>
</table>

The size of an embedded image can be measured with (6), where \( \|x\| \) denotes the size of \( x \) and \( I_z \), \( I_c \), and \( I_o \) denote the size of indices with tiny values, middle values, or large values, respectively:

\[
\| \text{embedded image} \| = (N_z + s) \ast I_z + (N_z + N_c) \ast I_c + (N_z + \log_2(cs)) \ast I_o.
\]  

(6)

In (6), \( (N_z + s) \) and \( (N_z + N_c) \) can be controlled in less or more than \( \log_2(cs) \) bits and only the third item always requires a space with more than \( \log_2(cs) \) 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.

**Input.** A grayscale cover image \( C \) of size \( N \ast M \), 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 \( (N/\nu) \ast (M/\mu) \).

**Step 2.** Perform SMVQ predictions with the VQ-compressed image \( I \) to create the SMVQ-compressed image \( T \) of size \( (N/\nu) \ast (M/\mu) \).

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

**Step 4.** If \( Q \leq 2^{N_z} - 3 \) (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), \( cs.B = cs.B \oplus D2B(Q) \oplus sd \), where \( \oplus \) 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 \( (2^{N_z} - 2) + 2^{N_z} - 1 \) (a middle-value index), then \( Q \) is encoded using (4), \( cs.B = cs.B \oplus \text{indicator 0} \oplus D2B(Q - 2^{N_z} + 2) \).
Step 6. If \( Q \geq (2^{N_z} - 2) + 2^{N_z} \) (a large-value index), then
\[
Q \text{ is encoded using (5),} \nonumber
\]
\[
cs_b = cs_b \parallel \text{ indicator}_1 \parallel \text{D2B}(Q). \nonumber
\]

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 \text{ indicator } 0 \text{ and } I_1 \text{ indicator } 1, \text{ where } N_z = 2, N_c = 5, s = 5, \text{ and } cs = 128, \text{ and the size of the state codebook is identical to that of the VQ codebook.}

For each image (Figure 5), let \( R_1 \) to \( R_s \) be the indices located in the first row and \( R_s \) to \( R_t \) 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 \( 2^{N_z} - 2 \). The index is encoded using (3). Secret data are (00000). According to (3), the index with \( N_z \) bits followed by secret data (00000), is presented as the encoded result for \( R_1 \). \( R_2 \) has a value of 3, and it is in between \( (2^{N_z} - 2) + 0 \) and \( (2^{N_z} - 2) + 2^{N_z} - 1 \). 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 \( (2^{N_z} - 2) + 2^{N_z} - 1 \). 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 \((N_z + \log_2(cs))\) 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 \( 2^{N_z} - 2 \), the encoded index is a \( E \) index. Then, the original index is directly restored by \( d_{N_z} \). Next, read \( s \) bits from the code stream \( cs_b \) to reconstruct the secret data. In the second case, if \( d_{N_z} \) is \( 2^{N_z} - 1 \), the encoded index is an \( O_i \) index. Then, read the next \( \log_2(cs) \) 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 \( 2^{N_z} - 2 \), the encoded index is a \( G \) 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.

\textbf{Input.} The code stream in binary form \( cs_b \) and parameters \( N_z \) and \( N_c \).
\textbf{Output.} The reconstructed VQ-compressed image \( I' \) of size \( N \times M \) and the retrieved secret data stream.

Step 1. Set \( ptr = 1 \) and the restored \( sd = \text{Null} \).

Step 2. Read the next \( N_z \) bits from the code stream \( cs_b \) and convert the \( N_z \) bits to decimal value, denoted by \( d_{N_z} \). The \( N_z \) bits are processed.

Step 3. If \( d_{N_z} < 2^{N_z} - 2 \) (a tiny-value index), then
\[
\text{restored index } = d_{N_z} \nonumber
\]
\[
\text{read the next } s \text{ bits, denoted by } sd', \text{ from code stream } cs_b; \nonumber
\]
\[
\text{restored } sd = \text{restored } sd' \parallel sd', \nonumber
\]
\[
ptr = ptr + s \nonumber
\]

Step 4. If \( d_{N_z} = 2^{N_z} - 1 \) (\text{ indicator } 1), then
\[
\text{read the next } \log_2(cs) \text{ bits from the code stream } cs_b, \nonumber
\]
\[
d_{O_i} = \text{B2D}(cs_b(ptr : ptr + \log_2(cs) - 1), ptr = ptr + \log_2(cs), \nonumber
\]
\[
\text{restored index } = d_{O_i} \nonumber
\]

Step 5. If \( d_{N_z} = 2^{N_z} - 2 \) (\text{ indicator } 0), then
\[
\text{read the next } N_c \text{ bits from the code stream } cs_b, \nonumber
\]
\[
d_{G_i} = \text{B2D}(cs_b(ptr : ptr + N_c - 1), ptr = ptr + N_c, \nonumber
\]
\[
\text{restored index } = d_{G_i} + (2^{N_z} - 2) \nonumber
\]

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
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 $512 \times 512$. 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:

$$ BR = \frac{\| \text{embedded image} \|}{N \times M}, $$

$$ EC = s \times I_z, $$

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

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 $\log_2(cs)$ bits. A VQ index encoded as an $E_i$ or $G_i$ index yields an additional space with $\log_2(cs) - N_z$ 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 $N_z$ are set to $\log_2(cs) - N_z$. 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), \text{ and } (1024, 5, 5, 5), \text{ 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.
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 the 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 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

**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.
<table>
<thead>
<tr>
<th>Images</th>
<th>Codebooks sizes/VQ</th>
<th>BR (in bpp)</th>
<th>ER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>128/0.438</td>
<td>256/0.50</td>
<td>512/0.563</td>
</tr>
<tr>
<td></td>
<td>3/4</td>
<td>3/5</td>
<td>4/5</td>
</tr>
<tr>
<td></td>
<td>N_1/s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Airplane</td>
<td>0.454</td>
<td>0.466</td>
<td>0.438</td>
</tr>
<tr>
<td></td>
<td>0.446</td>
<td>0.509</td>
<td>0.574</td>
</tr>
<tr>
<td></td>
<td>53144</td>
<td>62095</td>
<td>65905</td>
</tr>
<tr>
<td>Lena</td>
<td>0.478</td>
<td>0.471</td>
<td>0.441</td>
</tr>
<tr>
<td></td>
<td>0.442</td>
<td>0.505</td>
<td>0.570</td>
</tr>
<tr>
<td></td>
<td>55396</td>
<td>62350</td>
<td>65955</td>
</tr>
<tr>
<td>Pepper</td>
<td>0.483</td>
<td>0.483</td>
<td>0.456</td>
</tr>
<tr>
<td></td>
<td>0.442</td>
<td>0.504</td>
<td>0.570</td>
</tr>
<tr>
<td></td>
<td>55928</td>
<td>63810</td>
<td>68030</td>
</tr>
<tr>
<td>Boat</td>
<td>0.423</td>
<td>0.380</td>
<td>0.354</td>
</tr>
<tr>
<td></td>
<td>0.446</td>
<td>0.509</td>
<td>0.579</td>
</tr>
<tr>
<td></td>
<td>49432</td>
<td>50775</td>
<td>53720</td>
</tr>
<tr>
<td>Sailboat</td>
<td>0.435</td>
<td>0.416</td>
<td>0.391</td>
</tr>
<tr>
<td></td>
<td>0.446</td>
<td>0.509</td>
<td>0.576</td>
</tr>
<tr>
<td></td>
<td>50792</td>
<td>55515</td>
<td>59120</td>
</tr>
<tr>
<td>Tiffany</td>
<td>0.492</td>
<td>0.521</td>
<td>0.473</td>
</tr>
<tr>
<td></td>
<td>0.441</td>
<td>0.503</td>
<td>0.569</td>
</tr>
<tr>
<td></td>
<td>56860</td>
<td>66825</td>
<td>70475</td>
</tr>
<tr>
<td>House</td>
<td>0.394</td>
<td>0.370</td>
<td>0.345</td>
</tr>
<tr>
<td></td>
<td>0.449</td>
<td>0.513</td>
<td>0.584</td>
</tr>
<tr>
<td></td>
<td>46392</td>
<td>49780</td>
<td>52745</td>
</tr>
<tr>
<td>Elaine</td>
<td>0.498</td>
<td>0.484</td>
<td>0.460</td>
</tr>
<tr>
<td></td>
<td>0.440</td>
<td>0.503</td>
<td>0.568</td>
</tr>
<tr>
<td></td>
<td>57492</td>
<td>63775</td>
<td>68445</td>
</tr>
</tbody>
</table>

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_1, 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_1/s, N_c) are set at (2, 3, log_2(cs) – N_1), (3, 3, log_2(cs) – N_1), (3, 3, log_2(cs) – N_1), and (4, 3, log_2(cs) – N_1), 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>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Codebook sizes</td>
<td>Codebooksizes/VQ</td>
<td>BR</td>
</tr>
<tr>
<td></td>
<td>128/0.438</td>
<td>256/0.500</td>
<td>512/0.563</td>
</tr>
<tr>
<td></td>
<td>BR (in bpp)</td>
<td>EC (in bits)</td>
<td></td>
</tr>
<tr>
<td>Airplane</td>
<td>0.182 0.149 0.124 0.119 0.236</td>
<td>0.208 0.326 0.343 0.291 0.269</td>
<td></td>
</tr>
<tr>
<td>Lena</td>
<td>0.179 0.136 0.115 0.094 0.234</td>
<td>0.208 0.332 0.348 0.275 0.261</td>
<td></td>
</tr>
<tr>
<td>Peppers</td>
<td>0.187 0.150 0.116 0.104 0.244</td>
<td>0.216 0.338 0.359 0.287 0.264</td>
<td></td>
</tr>
<tr>
<td>Toys</td>
<td>0.195 0.160 0.136 0.119 0.254</td>
<td>0.234 0.397 0.373 0.332 0.302</td>
<td></td>
</tr>
<tr>
<td>Gold hill</td>
<td>0.142 0.109 0.084 0.074 0.203</td>
<td>0.158 0.306 0.321 0.248 0.232</td>
<td></td>
</tr>
<tr>
<td>Zelda</td>
<td>0.179 0.135 0.111 0.088 0.250</td>
<td>0.220 0.338 0.375 0.308 0.298</td>
<td></td>
</tr>
</tbody>
</table>

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 \(1 + \log_2(cs)\) 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 \(\log_2(cs) - N_z\) 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 \(\log_2(cs)\), 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 \(\log_2(cs)\) bits, which yields a hiding space of \(\log_2(cs) - 1 - 2\) bits for a SOC index and \(\log_2(cs) - N_z\) bits for an \(E_i\) 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 \(\log_2(cs)\) bits, which can achieve a high embedding space, and the length of the encoded result for a SOC index is \(1 + 2 + \log_2(cs)\) bits. Because the proposed scheme has good performance in terms of ER, BR, and EC, \(N_z\), and the hiding space of the proposed scheme is set to \(3, \log_2(cs) - N_z\) and \(\log_2(cs) - 1\) 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 \(3 + \log_2(cs) - 1\) 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 still 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.

<table>
<thead>
<tr>
<th>Images</th>
<th>Shie and Lin’s [12] scheme</th>
<th>The proposed scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Codebooksizes/VQ</td>
<td>BR</td>
</tr>
<tr>
<td></td>
<td>256/0.500</td>
<td>512/0.563</td>
</tr>
<tr>
<td>Airplane</td>
<td>2/5/8</td>
<td>0.383</td>
</tr>
<tr>
<td>Lena</td>
<td>2/6/9</td>
<td>0.522</td>
</tr>
<tr>
<td>Peppers</td>
<td>2/8/11</td>
<td>25600</td>
</tr>
<tr>
<td>Boat</td>
<td>2/9/12</td>
<td>0.371</td>
</tr>
<tr>
<td>Sailboat</td>
<td>3/5/8</td>
<td>0.524</td>
</tr>
<tr>
<td>Gold hill</td>
<td>3/6/9</td>
<td>50915</td>
</tr>
<tr>
<td>Zelda</td>
<td>3/7/10</td>
<td>0.369</td>
</tr>
<tr>
<td>Zelda</td>
<td>3/8/11</td>
<td>0.524</td>
</tr>
<tr>
<td>Zelda</td>
<td>4/6/10</td>
<td>50735</td>
</tr>
<tr>
<td>Zelda</td>
<td>5/7/11</td>
<td>0.372</td>
</tr>
<tr>
<td>Zelda</td>
<td>6/8/12</td>
<td>0.533</td>
</tr>
<tr>
<td>Zelda</td>
<td>7/9/13</td>
<td>38015</td>
</tr>
<tr>
<td>Zelda</td>
<td>8/10/14</td>
<td>41235</td>
</tr>
<tr>
<td>Zelda</td>
<td>9/11/15</td>
<td>0.338</td>
</tr>
<tr>
<td>Zelda</td>
<td>10/12/16</td>
<td>0.527</td>
</tr>
<tr>
<td>Zelda</td>
<td>11/13/17</td>
<td>46695</td>
</tr>
<tr>
<td>Zelda</td>
<td>12/14/18</td>
<td>0.361</td>
</tr>
<tr>
<td>Zelda</td>
<td>13/15/19</td>
<td>0.525</td>
</tr>
<tr>
<td>Zelda</td>
<td>14/16/20</td>
<td>49665</td>
</tr>
</tbody>
</table>

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


Submit your manuscripts at http://www.hindawi.com