Lossless data hiding for VQ indices based on neighboring correlation

Jiann-Der Lee
Department of Electrical Engineering
Chang Gung University
Tao-Yuan, 333 Taiwan
Email: jdlue@mail.cgu.edu.tw

Yaw-Hwang Chiou
Department of Electrical Engineering
Chang Gung University
Tao-Yuan, 333 Taiwan
Email: chiou.yh@msa.hinet.net

I. INTRODUCTION

With the rapid development of Internet, it is often for users to communicate and exchange information with others every day. Hence, tons of digital information, such as digital images, video, audio and so on, is transmitted over the Internet and security problems such as interception, modification and others have become more and more serious. It is extremely important for taking precautions to ensure that information over the Internet remains safe and secure. Basically, data hiding is an important technique that embeds secret data into a cover image with minimal perceptible degradation. That is, in order to avoid causing the third party to pay attention to the cover image, the quality of the embedded image must be as similar as original image. In general, information hiding techniques can be classified into two categories, namely reversible information hiding schemes [1-4, 8] and irreversible information hiding schemes [5-7]. However, some applications such as military, legal literature, medical, fine art work, and so on, it is also desired that the original cover image can be recovered because of the required content integrity. In irreversible information hiding schemes, only secret data are extracted and no restoration of cover images is made. In contrast, in reversible information hiding schemes, secret data are extracted and cover images can be completely restored. In general, information hiding techniques can be performed in three domains, namely spatial domain [8-11], frequency domain [12-17], and compressed domain. In 2002, Jo and Kim [18] proposed an irreversible data hiding technique based on vector quantization. In 2006, Shie et al. proposed an adaptive data hiding scheme based on VQ-compressed code [5]. This scheme is also an irreversible information hiding scheme. During 2007 to 2009, Chang et al. have proposed various schemes [22, 1, 3-4] about reversible data hiding on VQ compressed domain. In reference [3], a VQ-based data hiding scheme with recovery capability is presented. In their 2-bit embedding method, first, codewords in the original VQ codebook is sorted in the descending order based on referred counts. Next, the sorted codebook is partitioned into six clusters. The clusters with the largest and second largest referred counts are used to hide secret data. The others are used only for image reconstruction. In reference [4], the secret data are hidden in compression image based on principle that the neighboring blocks in an image have a high correlation and frequently occurring indices are encoded by short codes and rarely occurring symbols are encoded by long codes. Further, reference [23] shows a general scheme of reversible data embedding and extraction. In this scheme, a transform is chosen in such a way that it creates a large free space, i.e., it has a strong decorrelating effect. Furthermore, filling the free space with random values should not lead to great distortions in the reconstructed image.

In order to restore the original image and increase hiding capacity, based on the neighboring blocks of an image with high correlation and inspired from the SMVQ, we propose a novel lossless data hiding scheme in the VQ-compressed domain. In the VQ compressed image, the indices with the area directly above and the area directly to the left of the current encoding index are used as a reference. The values of indices around the reference are selected to form sub-codebook. A great deal of compression codes can be encoded by using the corresponding sub-codebook. Since the size of sub-codebook is smaller than that of original codebook, the encoded size of each index by using sub-codebook can be reduced. The VQ-compressed cover image is encoded in such a way that it creates a large free space. Furthermore, filling the free space with secret data should not lead to great distortions in the reconstructed image.

The remainder of this paper is organized as following. The proposed scheme is described in detail in section II. The experimental results and performances comparison of our scheme and previous approaches are included in section III.
II. THE PROPOSED SCHEME

Although the neighboring blocks in an image have high correlation, the variation between two neighboring indices in VQ is usually not so small. Therefore, a mean sorted codebook can be applied to reduce the variation between two neighboring indices. Then we can compress the indices with a lossless compression method.

The grayscale cover image $C$ with the size $N \times M$ is partitioned into small non-overlapping blocks of $n \times m$ pixels. In this case, cover image is partitioned into $(N/n) \times (M/m)$ blocks. Each block of the image can be encoded by using VQ. Finally, a block of the image is only represented with an index $z$ directly above and the area directly to the left of the current image as follows. Read the next compression code from compressed image.

A. Encoding procedure

The encoding phase of compression codes is explained as follows. Let $C$ be the cover image. Two nearby blocks, $z$ and $Y$, represent the area directly above the current encoding index $Y$, respectively. First, fine subcodebook is created by the two neighboring indices, $Y_U$ and $Y_L$. A search operation is performed on fine subcodebook using $Y$ as a search key. If there is a coincidence between $Y$ and one codeword in fine subcodebook, we obtain a hit state, i.e., $Y = Y_U$, or $Y = Y_L$. If $Y$ is equal to $Y_U$ ($Y_L$), then $Y$ has vertical (horizontal) extension. Next, $(Y_U - z \ldots Y_U - 1, Y_U + 1 \ldots Y_U + z) \neq (Y_U - z, Y_U + z)$, denoted as $sb11$, and $(Y_L - z \ldots Y_L - 1, Y_L + 1 \ldots Y_L + z) \neq (Y_L - z, Y_L + z)$, denoted as $sb12$, are used to form a coarse subcodebook. In similar way as done for fine subcodebook, a search operation is performed on coarse subcodebook using $Y$ as a search key. If there is a coincidence between $Y$ and one codeword in coarse subcodebook, we have a hit state. If $Y$ belongs to $sb11$ or $sb12$, then $Y$ has vertical (horizontal) extension. Finally, if there is no coincidence between $Y$ and one codeword in coarse subcodebook or fine subcodebook, we have a miss state. The above process is repeated until all indices of the compressed image are processed. Because three kinds of search states are used, i.e., hit state, coarse hit state and miss state, respectively and $Y$ has vertical extension or horizontal extension, some leading codes (indicators) are required to distinguish these. The leading code of each search state is shown in Table I. In fine hit state, the search result is represented by leading code, i.e., only 2 bits. In coarse hit state, the search result is represented by leading code followed by $y_i$, where $y_i$ is the position of $Y$ in the coarse subcodebook. In miss state, the search result is represented by leading code followed by the index $Y$. In encoding phase, compression code located in the first column and the first row of the compressed image is encoded by miss state. Compression codes located in the first row excluded the first one only use $Y_U$ to create fine subcodebook or coarse subcodebook for encoding. Compression codes locate in the first column excluded the first one only use $Y_L$ to create fine subcodebook or coarse subcodebook for encoding.

The encoding process of the proposed method is summarized as follows. Input: A grayscale cover image $C$ sized $N \times M$, a codebook $CB$ sized $cs$. Output: The code stream in binary form $bs$.

1. Rearrange codewords of codebook in the ascending order according to mean value of each codeword.
2. Compress cover image $C$ by using VQ to obtain the compressed image $I$ sized $(N/n) \times (M/m)$.
3. Read the next compression code, denoted as $Y$, from compressed image $I$ in raster scan order.
4. Use the upper neighboring index and left neighboring index of $Y$ to create fine subcodebook or coarse subcodebook.
5. If fine hit state occurs
   a) if it is horizontal extension, then $bs = bs || (002)$,
   where $||$ denotes the concatenation operation.
   b) if it is vertical extension, then $bs = bs || (012)$.
6. If coarse hit state occurs and the position of $Y$ in $sb11$ or $sb12$ is $y_i$, then $bs = bs || (102)$.
   a) if it is horizontal extension, then $bs = bs || (002) || (y_i2)$
   b) if it is vertical extension, then $bs = bs || (112) || (y_i2)$.
7. If miss state occurs, then $bs = bs || (112) || (Y2)$.
8. Repeat Step 3 to Step 8 until all compression codes in the compressed image are processed.
9. Output the code stream $bs$.

An example of the encoding process is shown in Fig. 1. Fig. 1a is the original compressed image. Let $C1$ to $C5$ be the indices located in the first row and $C6$ to $C10$ be the indices located in the second row, and so on. Suppose that the codebook size is 128 and $z$ is 2. $C1$ located in the first column and the first row is encoded by miss state. A leading code $(11)2$ followed by compression code $(00011002)$ are outputted as the encoding result of index $C1$. Because the left neighboring compression code of $C2$ is number 12, then the coarse subcodebook $\{10, 11, 13, 14\}$ is corresponding to the number 12 and the position of $C2$ in the coarse subcodebook...
is 2. A leading code (100)_2 followed by position value (10)_2 are outputted as the encoding result of index C2. Because the left neighboring compression code of C5 is number 25, then the coarse_subcodebook {23, 24, 26, 27} is corresponding to the number 25. C5 does not belong to coarse_subcodebook and is encoded by miss state. C7 possesses fine_hit state and horizontal extension. A leading code (00)_2 is outputted as the encoding result of index C7. C10 possesses fine_hit state and vertical extension. A leading code (01)_2 is outputted as the encoding result of index C10.

B. Decoding procedure

In the decoding process, the receiver can obtain the reconstructed compression codes by analyzing code stream bs. The decoding procedure of reconstructed compression codes is explained as follows. Initially, read the next leading code from code stream bs. In the first case, if leading code equals to (11)_2, the encoding state is a miss state. Then, read next \( \log_2(cs) \) bits and convert them into decimal value \( h \) and the original compression code is recovered by \( h \). In the second case, if leading code equals to (00)_2, the encoding state is a fine hit state with horizontal extension. Then, read the left neighboring compression code \( Lr \) and the original compression code is recovered by \( Lr \). In the third case, if leading code equals to (01)_2, the encoding state is a fine hit state with vertical extension. Then, read the upper neighboring compression code \( Ur \) and the original compressed code is recovered by \( Ur \). In the fourth case, if leading code equals to (10)_2, the encoding state is a coarse hit state. Next, read the upper neighboring compression code and left neighboring compression code to create coarse_subcodebook. Then, read next one bit in code stream bs to distinguish between horizontal extension and vertical extension. Next, read next \( \log_2(2z) \) bits and convert them into decimal value \( Lr \). If vertical (horizontal) extension is hold, the \( i \)-th element in \( sb11 \) (sb12) is retrieved to recover the original compression code. Fig. 2 shows an example of the decoding process. For the convenience to understand, the encoded code stream is partitioned into blocks according to the compression code as a unit. The two neighboring indices are separated by a symbol ‘→’. Fig. 2a shows the partitioned code stream and the original code stream can be restored by using the symbols ‘→’ to concatenate two separated items.

In summary, the decoding process of reconstructed compression codes is presented as below.

Input: The code stream in binary form \( bs \).
Output: The reconstructed cover image \( C' \) sized \( N \times M \).
1) Set \( ptr=1 \).
2) Read leading code(2 bits) from code stream \( bs \).
3) If leading code=(11)_2, then \( ptr = ptr + 2 \),
   read next \( \log_2(cs) \) bits from code stream \( bs \), reconstructed compression code=\( B2D(bs(ptr : ptr + \log_2(cs) - 1)) \), where \( B2D(x) \) denotes binary to decimal operation, \( ptr = ptr + \log_2(cs) \).
4) If leading code=(00)_2, then \( ptr = ptr + 2 \),
   read left neighboring compression code \( Lr \), reconstructed compression code=\( Lr \).
5) If leading code=(01)_2, then \( ptr = ptr + 2 \),
   read upper neighboring compression code \( Ur \), reconstructed compression code=\( Ur \).
6) If leading code=(10)_2, then \( ptr = ptr + 2 \),
   a) read upper neighboring compression code and left neighboring compression code to create coarse_subcodebook,
   b) read next 1 bit from code stream \( bs \) to distinguish between horizontal extension and vertical extension, \( ptr = ptr + 1 \),
   c) read next \( \log_2(2z) \) bits from code stream \( bs \), \( y_i = B2D(bs(ptr : ptr + \log_2(2z) - 1)) \), \( ptr = ptr + \log_2(2z) \),
   d) if vertical extension, reconstructed compression code=\( sb11(y_i) \).
   e) if horizontal extension, reconstructed compression code=...
thereby increased, and embedding capacity and method with a larger value of \( EC \) space of a compression code is denoted as \( 0.4375, 0.5, 0.5625, 0.625 \text{ bpp} \), respectively. The space of \( 256*16, 512*16, \) and \( 1024*16 \) respectively, then the \( BR \) value of \( K \) denotes the size of \( x \). A compression method with a smaller value of \( BR \) means that the method has a better compression performance and vice versa.

\[
BR = \| \text{code stream} \| / \| \text{image} \| \tag{1}
\]

For example, a cover image sized \( 512*512 \) pixels is compressed by using VQ with the codebooks sized \( 128*16, 256*16, 512*16, \) and \( 1024*16 \) respectively, then the \( BR \) are \( 0.4375, 0.5, 0.5625, 0.625 \text{ bpp} \), respectively. The space of compression codes by using the proposed scheme would be stored for less one. In the proposed scheme, supposed, the space of a compression code is denoted as \( K \) bits. Embedding capacity (EC) may be denoted as Eq. (2). A compression method with a larger value of \( EC \) means that the method has more space for secret data. It is clearly shown that the \( K \) is thereby increased, and embedding capacity and \( BR \) becomes larger as well.

\[
EC = \| \text{miss state} \| \times (K - 2 - \log_2(cs)) + \| \text{fine hit state} \| \times (K - 2) + \| \text{coarse hit state} \| \times (K - 3 - \log_2(2z)) \tag{2}
\]

\[
Z_{cs} = -EC \mid_{K=0} \tag{3}
\]

\[
\| \text{code stream} \| = EC + Z_{cs} = (N/n) \times (M/m) \times K \tag{4}
\]

In the proposed scheme, suppose that \( K = \log_2(cs) \). In a miss state, a compression code will sacrifice 2 bits from the hiding capacity. In a coarse hit state, a compression code will contribute \( \log_2(cs) - 3 - \log_2(2z) \) bits to the hiding capacity. In a fine hit state, a compression code will contribute \( \log_2(cs) - 2 \) bits to the hiding capacity. The compressed image can obtain larger (smaller) \( EC \) when more indices of the compressed image are encoded in fine hit state or coarse hit state (miss state). The value of \( z \) is increased in order to increase the number of coarse hit states. Accordingly, the number of miss states is decreased.

An appropriate value is selected for \( K \) such that \( EC \) is to be a positive number. The encoded code stream of compressed image contains encoded information, including the information about code, and embedded information, including the secret data, when \( K \) is constant. The size of encoded information \( (Z_{cs}) \) is defined in Eq. (3). The size of embedded information is denoted as \( EC \) and defined in Eq. (2). Finally, the size of code stream can be denoted as Eq. (4). After the compression codes of the compressed image are processed by the proposed scheme, it can provide a hiding space with \( EC \)-bit. In short, for the proposed scheme, the secret data can be embedded in compressed image by using two methods, namely batch data hiding method and distributed data hiding method. In batch data hiding method, the whole encoded code stream pattern is shown as (encoded information) || (total hiding secret data). In distributed data hiding method, a compression code provides some hiding space and partial secret data with the length as hiding space are taken and hidden. The encoded pattern of a compression code is shown as (original bit stream) || (partial hiding secret data). In the receiver, in order to extract the hidden secret data, the decoding procedure is modified and explained as follows. For the batch data hiding method, according to the decoding procedure is described in section II.B. \((N/n) \times (M/m)\) reconstructed compression codes can be decode from the preceding part of code stream. Then, the hidden secret data resident in the residuary part of code stream can be extracted. For the distributed data hiding method, a compression code possesses not only encoded information but also embedded information. The hiding space supplied by a compression code is variable, i.e., \( K – 2 \) bits for fine hit state, \( K – 3 – \log_2(2z) \) bits for coarse hit state and \( K – 2 – \log_2(cs) \) bits for miss state (none data is hidden if the value of the hiding space is negative). The extraction process is described as follows. Firstly, a reconstructed compression code can be decoded from code stream according to the decoding procedure which is described in section II.B. Secondly, the hidden secret data, named as partial secret data, are extracted from code stream according to the hidden space supplied by the compressed code. Finally, \((N/n) \times (M/m)\) reconstructed compression codes can be decoded from code stream and all the partial secret data are concatenated to form secret data.

III. EXPERIMENTAL RESULTS

In this section, the experimental results of applying the proposed technique are presented. Various test images with the size \( 512 \times 512 \) [21] are selected as the training images or cover images. The codebooks with 128, 256, 512 and 1024 16-dimensional codewords were trained by the LBG[19] algorithm with "airplane", "boat", "lena", "pepper", and "sailboat" as the training images. These codebooks are used to evaluate the compression performance between the proposed scheme and the others. There are 16384 compression codes (indices) are created when an image sized \( 512 \times 512 \) is encoded by using VQ of which codebook with 16-dimensional codewords.

Under the same experimental environment, the proposed method, Chang et al.'s [3] method, and Chang et al.'s [4] method are realized. The experimental results of the proposed method are also compared to other works and show a significant improvement. The secret data in the experiment is composed of bit value of 0 and 1 which is generated by pseudo random number generator. For more secure, the secret data
can be encrypted prior to embedding by using cryptographic methods such as DES or RSA.

The EC and BR of the proposed method with different codebook size and value of z are shown in Table II. Second row in the table shows BR of the VQ. In addition, the BR and EC of each image is obtained when K is set to be \( \log_2(cs) \). The proposed method has a smaller value of EC when a larger codebook is used. For a codebook sized 128*16, there is a larger EC when z is set to be 8. For a codebook sized 256*16, EC, z valued 8 or 16, shows slight difference from EC, z valued 16. For a codebook sized 512*16, there is a larger EC when z is set to be 16. For a codebook sized 1024*16, there is a larger EC when z is set to be 32. A larger value of z means that there are more coarse-hit states. The number of miss states is decreased when the number of coarse-hit states is increased. A new value of EC is formed by adjusting z.

Under the same experimental circumstance, the Chang et al.’s[3] scheme is realized for performance comparison. Third row in Table III shows BR of the VQ. The EC and BR of the Chang et al.’s[3] scheme, embedded two-bit secret data into each index value, are shown in left side of Table III, where these results are obtained in different codebook size. The EC and BR of the proposed scheme, z valued 8 or 16, are shown in right side of Table III, where these results are obtained in different codebook size. To remain the compressed effect of VQ the EC and BR of all the test images exclude “sailboat”, “boat” and “elaine” are obtained under the case that K is less than or equal to \( \log_2(cs) \). For certain of K, the EC of our proposed scheme is better than that of Chang et al.’s method.

The two-bit embedding algorithm of Chang et al. may supply more hiding space for secret data. Nevertheless, to get more hiding space makes the BR increased rapidly.

Accordingly, the BR of the embedded result is larger than that of the original compressed image. In this method, only secret data are hidden and no compression of compressed image is made. Compared to Chang et al.’s method [3], our scheme has larger embedding capacity and smaller BR for each cover image as shown in Table III.

Under the same experimental circumstance, the Chang et al.’s [4] scheme is realized. It has better performance when block size is set to be 16. Third row in Table IV shows BR of the VQ. The EC and BR of the Chang et al.’s [4] scheme, block size valued 16, are shown in left side of Table IV and that of the proposed scheme, z valued 8 or 16, are shown in right side of Table IV. To remain the compressed effect of VQ the EC and BR of all the test images are obtained under the case that K is less than or equal to \( \log_2(cs) \). For certain of k, the EC of our proposed scheme are better than that of Chang et al.’s method. Compared to Chang et al.’s method [4], our scheme has larger embedding capacity and smaller BR for each cover image as shown in Table IV. As illustrated in Tables II, III and IV, the compression rate and embedded capacity of the cover images produced by the proposed scheme are better than that of those produced by Chang et al.’s [3] method and Chang et al.’s [4] method. More specifically, The BR of all test images exclude “sailboat”, “boat” and “elaine” processed by using the proposed scheme are show on the same or smaller than that processed by using VQ.

For the Chang et al.’s [3] scheme, an index only located in cluster 1 or cluster 2 is able to embed two-bit secret data in that. In this case, the amount of indices used to embed secret data is significantly reduced. Moreover, an indicator with the length of \( \log_2(cs) \)-bit is added in front of an index when the
In this paper, a novel lossless recovery data hiding scheme based on neighboring correlation is proposed to embed secret data into a VQ compressed image. The proposed scheme is not only hiding secret data into a VQ compressed image but also able to restore the original image after secret data extraction. In addition to this, our proposed scheme has the best efficiency compared with previous schemes in various codebook sizes for eight test images.

### References


### IV. CONCLUSIONS

In this paper, a novel lossless recovery data hiding scheme based on neighboring correlation is proposed to embed secret data into a VQ compressed image. The proposed scheme is not only hiding secret data into a VQ compressed image but also able to restore the original image after secret data extraction. In addition to this, our proposed scheme has the best efficiency compared with previous schemes in various codebook sizes for eight test images.

### TABLE IV

<table>
<thead>
<tr>
<th>Proposed scheme</th>
<th>Chang et al.'s scheme[4]</th>
</tr>
</thead>
<tbody>
<tr>
<td>codebook size/block size</td>
<td>codebook size/g codec size</td>
</tr>
<tr>
<td>BR</td>
<td>R</td>
</tr>
<tr>
<td>128/16</td>
<td>256/16</td>
</tr>
<tr>
<td>0.375</td>
<td>0.5</td>
</tr>
<tr>
<td>0.4375</td>
<td>0.5</td>
</tr>
<tr>
<td>Image</td>
<td></td>
</tr>
<tr>
<td>airplane</td>
<td>13137</td>
</tr>
<tr>
<td>lena</td>
<td>14444</td>
</tr>
<tr>
<td>peppers</td>
<td>13992</td>
</tr>
<tr>
<td>boat</td>
<td>13911</td>
</tr>
<tr>
<td>sailboat</td>
<td>13617</td>
</tr>
<tr>
<td>tiffany</td>
<td>14860</td>
</tr>
<tr>
<td>house</td>
<td>13419</td>
</tr>
<tr>
<td>elaine</td>
<td>14336</td>
</tr>
</tbody>
</table>

### Note:

The proposed scheme embeds secret data into a VQ compressed image, while the Chang et al.’s scheme[4] embeds secret data into a VQ compressed image. The proposed scheme is not based on neighboring correlation and is proposed to embed secret data into a VQ compressed image. Based on the reasons, this scheme can not acquire larger hiding capacity and smaller BR.