IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 10, 2020, accepted January 31, 2020, date of publication February 11, 2020, date of current version February 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2973179

High Capacity Reversible Data Hiding for VQ-Compressed Images Based on Difference Transformation and Mapping Technique

YUEHUI LI[1], CHIN-CHEN CHANG[2,3], (Fellow, IEEE), AND MINGXING HE[1]

[1]School of Computer and Software Engineering, Xihua University, Chengdu 610039, China
[2]Department of Information Engineering and Computer Science, Feng Chia University, Taichung 407, Taiwan
[3]School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China

Corresponding author: Chin-Chen Chang (alan3c@gmail.com)

This work was supported in part by the Science and Technology Department of Sichuan Province under Grant 2017HH0083 and Grant 2017RZ0009, in part by the Ministry of Education of the People's Republic of China under Grant Z2012030, in part by the Education Department of Sichuan Province under Grant 16ZA0157, in part by Xihua University under Grant Z1222624 and szjj2017-010, in part by the Chengdu Science and Technology Bureau under Grant 2016-XT00-00015-GX, in part by Yibin Kuntuoshen Electronic Technology Co., Ltd., and in part by the Innovation Fund of Postgraduate of Xihua University under Grant YCJJ2019022.

ABSTRACT In this paper, two reversible data hiding (RDH) methods for vector quantization (VQ)-compressed images are presented. In the first method, the difference-index table instead of the standard-index table is designed, and all difference indices are then classified into two groups, i.e., the to-be-mapped group and the candidate group, according to the occurrence frequencies of them. Afterwards, the mapping table between those two groups is adaptively constructed and used to serve for data embedding. Furthermore, another joint method is conducted by combining the standard-index based method and the difference-index based method, resulting in a better embedding performance. The experimental results confirm that both proposed methods, especially the joint method, are significantly superior to the most recent schemes in terms of the embedding capacity, bitrate and embedding rate.

INDEX TERMS Reversible data hiding, vector quantization, difference index, mapping.

I. INTRODUCTION

Data hiding (also called data embedding or information hiding) embeds secret data imperceptibly into a cover media, such as a video, a piece of audio, an image, etc. Since 1990s, a variety of data hiding techniques have been performing a growingly important role in privacy protection, copyright, and so on [1]–[12], [49], [50]. Traditional data hiding is usually irreversible, and the embedding process brings permanent distortion to the original host image, which is unacceptable in circumstances where a complete reconstruction of the original host image and verification of data are required, such as encrypted data annotation in the cloud environment, remote medical diagnostic and judicial forensics, and so on. To achieve data hiding and a non-distortional reconstruction of original cover media, reversible (also called lossless) data hiding (RDH) technique was proposed [8]. The benefit is that the original host image can be recovered completely after the removal of secret data. RDH schemes include the spatial domain, compressed domain, frequency domain, and so on. Because the image is usually stored or transmitted in the form of compressed file, such a reversible data hiding technique for compressed images has become a research hotspot.

Vector quantization (VQ) [13] is a lossy data compression technique widely used in the various fields of image processing. In recent years, some VQ-based reversible data hiding methods have been developed. In all VQ-based reversible data hiding techniques, a VQ standard-index table is used as the cover image, and the image embedded with the secret message is called the "stego image," hence the reversibility means the capability of completely recovering the original VQ standard-index table after removing the secret data. For simplicity in description, we denote the VQ standard-index table as the standard-index table in this paper.

The associate editor coordinating the review of this manuscript and approving it for publication was Michele Nappi.

Data hiding techniques try to replace the redundant data within the image with some secret data in an undetectable manner. As such, an exploration of the correlations among the indices in the standard-index table is needed. It can be used to acquire redundancy, which is then replaced by secret data, thus achieving data hiding. The more redundancy that is explored, the more secret data that can be hidden. Hence, the goal is to achieve high embedding capacity. Irreversible data hiding schemes can achieve a high capacity relatively easily. However, it is hard for reversible data hiding schemes to achieve this goal because it has another task of reconstructing the original carrier without a loss; further, there is also a trade-off between the embedding capacity and the bitrate. The biggest challenge is to enhance the embedding capacity while maintaining a satisfactory bitrate.

Recently, research [14]–[47] related to VQ-based reversible data hiding schemes have been published. Most of them employ such techniques, e.g., SMVQ [47], [15], SOC [28], LAS [26] and JNC [43], to exploit various types of indirect correlations to achieve confidential communication. Scheme [47] embeds secret data into a compressed image by selecting different codewords in a sub-codebook that is generated based on SMVQ technique, and scheme [15] transforms each standard index into the transformed index, making the histogram of the transformed indices that is normally distributed around zero. In what follows, the secret data is embedded into the smallest index value using a transformed-index mapping strategy. Reference [28] first reduces the size of the standard index by combining the algorithms of SOC and SCM (state-codebook mapping) and then embeds the secret data into the vacated redundancy room. Moreover, the locally adaptive coding (LAS)-based RDH scheme [26] embeds the secret data into a standard-index table throughout the whole LAS encoding phase. In [43], the secret data is embedded by using the difference values between the current standard-index and its left or upper neighboring indices. In particular, the histogram-based index mapping mechanism proposed by [3] achieved a much higher embedding capacity than that of past methods. Obviously, those schemes exploit various types of indirect correlations to achieve a good performance in embedding capacity and just only explore the redundancy room in the standard-index table. To achieve a higher embedding capacity, other forms of the index table can be constructed to explore further more redundancy.

In this paper, another form of the correlation between the standard indices is explored. First, a difference-index table that consists of the differences between adjacent index values is constructed, and this table could contain more redundancy than that of the original standard-index table. Second, a corresponding mapping table can be constructed based on the occurrence frequency of indices in the difference-index table. Third, the corresponding amount of candidate indices are assigned to each selected index with a high occurrence frequency in an adaptive approach. As a result, an embedding process based on this mapping table can achieve a much higher embedding capacity. To further improve performance, a joint method is also proposed in this paper, i.e. a method of jointing the standard-index based method and the difference-index based method. As such, two embedding stages are performed. It can be expected that the embedding capacity can be enhanced dramatically. Moreover, for a better compression effect, some lossless compressing algorithm, such as Huffman coding and locally adaptive scheme (LAS) [27], [35], can be employed to compress the stego index table to ensure a satisfactory bitrate.

The rest of this paper is organized as follows. Section II reviews related works related to the VQ technique. The proposed schemes, including the difference-index based method and the joint method, are presented in Section III. Section IV gives the experimental results and discussions. Finally, Section V concludes this paper.

II. RELATED WORKS

In this section, the VQ and adaptive index rearrangement (AIR) technique are introduced in Subsections II-A and II-B, respectively. Then, a brief literature review is given in Subsection II-C.

A. VECTOR QUANTIZATION

VQ compression technique was proposed by Linde *et al.* [36], in 1980 and became an important data compression technique since it is widely used in various fields of multimedia compression [37]–[42]. VQ compression method includes three main steps: codebook generation, VQ encoding and VQ decoding. A codebook is trained by a clustering algorithm, such as the LBG algorithm [36] with some training images. It is constructed with a number of representative image blocks called codewords. Let $Y = \{Y_1, Y_2, \ldots, Y_k, \ldots, Y_{\mathrm{cz}}\}$ denotes the representative codebook, where the codebook size is CZ and $Y_k = \{y_{k1}, y_{k2}, \ldots, y_{kn}\}$ is the k^{th} codeword of $\beta \times \beta$ dimensions in Y. In the encoding process, the VQ encoder divides the original image into non-overlapping blocks with a size of $n = \beta \times \beta$ pixels, $X = (x_1, x_2, \ldots, x_n)$ denotes the current block. For each image block, a most similar codeword is searched for from the codebook Y, and the Euclidean distance defined as follows is used to evaluate the similarity.

$$ED(X, Y_k) = \sum_{i=1}^{n} (x_i - y_{ki})^2, \tag{1}$$

where y_{ki} denotes the i^{th} pixel of the codeword Y_k, x_i denotes the i^{th} pixel of block X. The most similar codeword is that with the minimum value of ED. Then the index value of the most similar codeword in the codebook is used to encode the current block X. As a result, using the VQ compression technique, the standard-index table of the image is then generated.

During the decoding process, the VQ decoder performs the reconstructing of each image block by the table-lookup operation based on an identical codebook Y. The codeword corresponding to the index in the codebook is obtained to reconstruct the compressed image.

B. ADAPTIVE INDEX REARRANGEMENT TECHNIQUE

The AIR was proposed by Hong *et al.* [10] in 2018. The concept of the AIR technique comes from the palette reordering [45], [46], and it rearranges the standard indices to make their spatial correlations stronger. Here, we give an example to show how to arrange the index using the AIR technique.

Step 1: Construct the occurrence matrix D, where D(i, j) represents the occurrence frequency of adjacent indices i and j in both the horizontal and vertical directions.

Step 2: Select two indices l_{k-1} and l_k that are most frequently found adjacent to each other, and place them in an index list L_k, where k represents the number of elements in L. For example, the initial index list should be $L_2 = (l_1, l_2)$. Let l_i represent the indices already in L and u_i is the indices that not yet been assigned. In each iteration, an index u^* in the unassigned index list that has the maximal occurrence of adjacencies to all other already assigned indices should be selected. Therefore, the index u * in the unassigned index list that maximizes $\sum_{j=1}^{k} \mathrm{D}(u_i, l_j)^t$ is selected and placed in the index list, where t is a constant.

Step 3: Decide which side to place u* into L, calculate $h_L = \sum_{j=1}^{m} \mathrm{D}(u^*, l_j)$ and $h_R = \sum_{j=m+1}^{k} \mathrm{D}(u^*, l_j)$, where $m = \lfloor k/2 \rfloor$. If $h_L \leq h_R$, u* is then placed to the right of L, or otherwise, u* is placed to the left of L.

Step 4: Process each index in the unassigned list in the similar manner until all indices in the unassigned list are assigned. Finally, replace those indices l_i in original index table by i and rearrange the codewords of codebook accordingly. The rearranged index table is then constructed.

Here is an example to illustrate the process of rearranging a standard-index table using AIR. Let us assume the standard-index table is as shown in Fig. 1(a) and its sorted occurrence frequency of indices is shown in TABLE 1. Next, we count the occurrence frequency of adjacent indices i and j in both the horizontal and vertical directions. For example, the adjacent indices ($i = 4, j = 1$) appears 5 times in the horizontal direction and 4 times in vertical direction, thus, the D(i, j) is equal to 9. By the same way, the occurrence matrix D can be derived as shown in Fig. 1(b). Fig. 1(c) is the rearranged standard-index table obtained by using AIR method based on Fig. 1(a), and its sorted occurrence frequency of indices is shown in TABLE 2. By comparing Fig. 1(a) and Fig. 1(c), we can see that the original indices 4, 1, 0, and 2 are mapped to 1, 2, 0, and 3, respectively. Meanwhile it is apparent from TABLE 1 and TABLE 2 that the occurrence frequency of indices in Fig. 1(a) and Fig. 1(c) are the same.

TABLE 1. Sorted index of Figure 1(a) and its occurrence frequency.

Index	Frequency (number)
4	5
1	4
0	4
2	3

TABLE 2. Sorted index of Figure 1(c) and its occurrence frequency.

Index	Frequency (number)
1	5
2	4
0	4
3	3

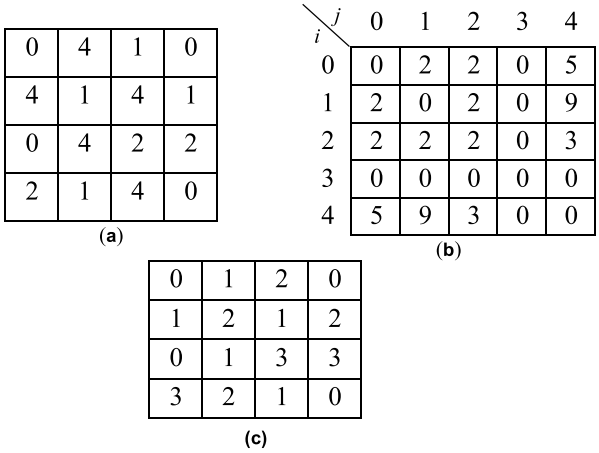


FIGURE 1. An example of AIR. (a) Original standard-index table. (b) Occurrence matrix D. (c) Rearranged standard-index table.

0	4	-3	-1
4	-3	3	-3
-1	4	-2	0
0	-1	3	-4

FIGURE 2. Difference-index table derived from Fig. 1(a).

On the other hand, we also construct a difference-index table corresponds to Fig. 1(a) and shown in Fig. 2, which can be derived by the following rules: the first index at the upper left corner, i.e., 0, remains unchanged; the differences are calculated by the operation of subtracting between two adjacent indices in a raster scanning manner. Similarly, Fig. 3 can be obtained from Fig. 1(c). The sorted occurrence frequency of difference indices in Fig. 2 and Fig. 3 are shown in TABLE 3 and TABLE 4, respectively. It can be seen by comparing TABLE 3 and TABLE 4 that, the occurrence frequency of the difference indices in Fig. 3 has a more concentrated distribution in comparison with that of the difference indices in Fig. 2. This means that, the larger amounts of redundancies within Fig. 3 can be vacated compared to that in Fig. 2. This characteristic of the AIR technique is beneficial for improving data embedding performance.

C. LITERATURE REVIEW

The index mapping scheme was proposed by Chang and Lin [29] in 2007 . It is based on a declustering strategy and utilizes the local spatial characteristics of the VQ-compressed

0	1	1	-2
1	1	-1	1
-2	1	2	0
0	-1	-1	-1

FIGURE 3. Difference-index table derived from Fig. 1(c).

TABLE 3. Sorted difference index of Figure 2 and its occurrence frequency.

Difference-index	Frequency (number)
4	3
-3	3
-1	3
0	3
3	2
-2	1
-4	1

TABLE 4. Sorted difference index of Figure 3 and its occurrence frequency.

Difference-index	Frequency (number)
1	6
-1	4
0	3
-2	2
2	1

images. Later, some other researchers also developed [3], [19], [22], [30]–[34]. In [22], [31], [32], index mapping schemes were based on the referred counts of the indices, and the embedding capacities are low. In [19], [29], [30], [34], the index mapping schemes were based on the side-match prediction, but they failed to use it in an efficient way. Scheme [3] also used the SMVQ and LAS schemes to explore the redundancy, followed by constructing an efficient index mapping mechanism based on the theory of Huffman coding, and finally they obtained a much higher embedding capacity and better bitrate than the previous methods.

Most of these index mapping methods embedded secret data based on the redundancy generated from the standard indices. However, if we construct another form of index table in an appropriate way, for example, a difference-index table which made up of the differences between adjacent indices, more redundancy might generate from it. There are some existing schemes that utilize the difference values between adjacent indices to hide secret data in a standard-index table, such as [10], [23], [24], [43], [44]. In schemes [23], [24], [43], [44], the difference values are first generated by employing their distinct strategies. Then, several cases are designed according to the amplitudes of the difference values and a preset threshold (or parameter). For each case, the variable-length of secret bits is embedded by encoding the index as a code stream, which is generated by concatenating the indicator and secret data or the binary representation of difference value. In [10], the AIR technique is employed to find a transformed standard-index table, thus making the neighboring indices have similar values. Meanwhile, a weight-controlled least square estimator is employed for prediction, and the indicators are better assigned according to the prediction errors. In summary, those schemes [10], [23], [24], [43], [44] utilize the difference-index coding to carry the secret data fully with the guidance of the amplitudes of the difference values. Besides, the embedding of these schemes is not based on an associate mapping table.

For the aim of high embedding performance, the novel difference-index based RDH methods for VQ compressed image are presented in this paper, which are quite different from the previous methods [10], [23], [24], [43], [44] with respect to the following aspects: first, the proposed methods effectively combine the mapping mechanism and the difference-index based scheme to generate more redundancy; second, a new difference-index table made up of the differences between adjacent standard indices is constructed, and the secret data will be embedded into it instead of a standard-index table. Third, an adaptive mapping mechanism for data embedding is constructed based on the occurrence frequency of the difference indices in the difference-index table, instead of the amplitudes of the difference values. Last, the redundancy of both the standard-index table and the difference-index table can even be easily combined by using the proposed joint method so that a higher embedding capacity can be obtained.

III. PROPOSED SCHEME

In this section, we describe the proposed difference-index based reversible data hiding scheme for the aim of achieving a high embedding performance. Before we start, let us first look at a simple example to illustrate that how to explore and vacate redundancy room.

Assume that an index sequence A = {10, 11, 12, 14, 17, 18, 19}. For A, it is considered that there is poor redundancy since there is no repeated index. But if we perform a simple transformation, i.e., let each index (except the first index, "10," which retains the original value) minus its adjacent left index to form a difference index. As a result, a new sequence, namely difference-index sequence, B = {10, 1, 1, 2, 3, 1, 1}, is generated, where it contains 4 repeated difference indices of 1. This means that more redundancies can be obtained in B than in A, hence a better embedding performance can be obtained. However, most of previous researches explored data embedding method only based on the standard index table (like A), while ignoring the better potential in their difference indices.

Therefore, inspired by the idea of above example, we present a novel RDH method based on difference-index table and a mapping mechanism, namely the difference-index

179	178	177	176	…	138
218	179	210	211	…	202
188	146	225	57	…	153
⋮	⋮	⋮	⋮	…	⋮
87	137	87	137	…	169

(a)

179	-1	-1	-1	…	-63
80	-39	31	1	…	56
-14	-42	79	-168	…	-42
⋮	⋮	⋮	⋮	…	⋮
-50	50	-50	50		0

(b)

FIGURE 4. Example of index tables for Baboon (codebook size = 256). (a) Standard-index table. (b) Difference-index table.

based method. Moreover, a joint method is conducted by combining the standard-index based method and the aforementioned difference-index based method, resulting in a better embedding performance. Next, we introduce how to construct the difference-index table in Subsection III-A. Subsection III-B describes the construction of mapping tables for the standard indices and the difference indices. Finally, the detailed process of data embedding and data extraction are given in Subsection III-C and Subsection III-D, respectively.

A. CONSTRUCTION OF THE DIFFERENCE-INDEX TABLE

In the pre-processing phase, the standard-index table is rearranged by using the AIR technique, and accordingly the codewords of codebook is also rearranged. Then the difference-index table can be derived from this rearranged standard-index table by using difference transformation. In this way, a higher redundancy can be achieved. Figure 4(a) gives the standard-index table with a size of 128 × 128 for image Baboon, where the codebook size is set to 256. Fig. 4(b) is the corresponding difference-index table, which can be derived from Fig. 4(a) by the following rules: 1) the first index at the upper left corner, i.e., 179, remains unchanged; 2) the differences are calculated by the operation of subtracting between two adjacent indices in a raster scanning manner.

Sequentially, the occurrence frequency of all indices or difference indices are counted and sorted by a descending order. The results are depicted in TABLE 5 and TABLE 6, respectively. Through comparing TABLE 5 with TABLE 6, it can be concluded that the indices with high occurrence frequency in the difference-index table are more condensed than those in the standard-index table.

For a better illustration, Fig. 5 shows the histogram of indices and difference-indices of images Lena and Baboon. It can be observed that the high occurrence indices in Fig. 5(a) and Fig. 5(c) are at a higher proportion than that of Fig. 5(b) and Fig. 5(d). These high occurrence indices can provide more redundancy for embedding secret data. Not only that, it is interesting that the first k difference indices have a high probability of occurrence in all difference indices. Fig. 6 and Fig. 7 show the sum of probability of occurrence of the first k difference indices or indices for images Lena and Baboon, respectively. This characteristic will be exploited to design

TABLE 5. Sorted index and its occurrence frequency.

Index: I	Frequency (number) of I: f_{I}
112	684
100	581
113	499
111	496
101	419
97	378
110	356
⋮	⋮
252	1

TABLE 6. Sorted difference index and its occurrence frequency.

Difference Index: DI	Frequency (number) of DI: f_{DI}
0	1981
1	870
-1	821
2	570
3	496
-3	478
-2	478
⋮	⋮
229	1

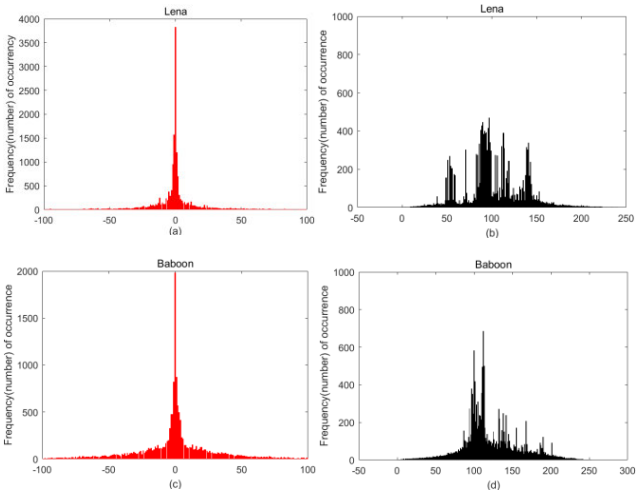


FIGURE 5. Comparison of histograms for (a) Difference indices (Lena). (b) Standard indices (Lena). (c) Difference indices (Baboon). (d) Standard indices (Baboon).

the idea of difference-index based RHD scheme and details will be stated in the following.

B. CONSTRUCTION OF THE MAPPING TABLE

In this section, the detailed discussion in construction of mapping tables for standard indices and difference indices is introduced respectively.

1) CONSTRUCTION OF MAPPING TABLE BASED ON STANDARD INDEX

For an image, we first sort its indices according to their frequency, and assume the sorted result $T = \{T_1, T_2, \ldots, T_s\}$,

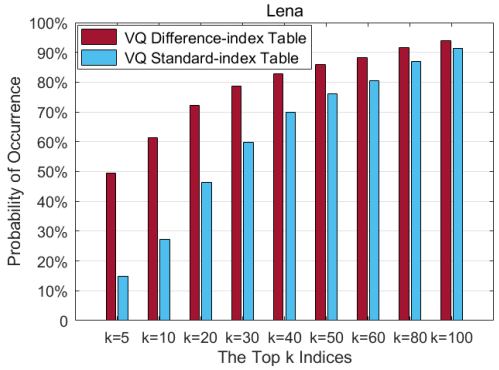


FIGURE 6. Comparison of the sum of probability of occurrence of the first k difference indices or standard indices for Lena.

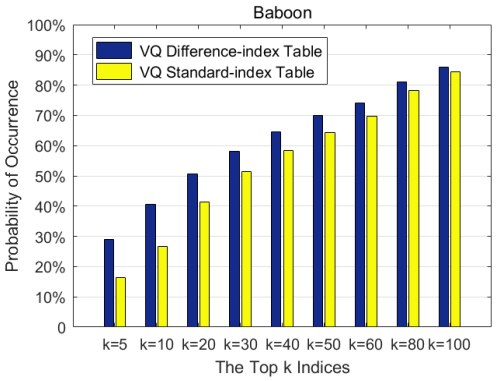


FIGURE 7. Comparison of the sum of probability of occurrence of the first k difference indices or standard indices for Baboon.

where s is the size of T. Generally, only one part of codewords are used during VQ compression. In other words, only some of indices will appear in the standard-index table for an image. For simplicity, we separate these appear indices into a group, represented by T_{appear}, and separate the remaining unused indices into other group, represented by $T_{nonappear}$. Moreover, in T_{appear}, we further separate the first N indices with highest frequency into the to-be-mapped group and denote it as TH $= \{T_1, T_2, \ldots, T_{\mathrm{N}}\}$. Accordingly, other remaining indices in T_{appear} are denoted as TL $= \{T_{\mathrm{N}+1}, T_{\mathrm{N}+2}, \ldots, T_m\}$, where m is the length of T_{appear}. Obviously, we can employ the indices in $T_{nonappear}$ and TL to represent additional bits to be embedded. Before providing the detailed embedding processes, we introduce how to adaptively assign indices between two groups and construct a mapping table based on standard index, as shown below.

Step 1: Derive the to-be-mapped group TH (first N indices with highest frequency).

Step 2: Construct the candidate group CI, which includes the indices in TL and $T_{nonappear}$, and denoted as CI $= \{T_{\mathrm{N}+1}, T_{\mathrm{N}+2}, \ldots, T_m, T_{m+1}, \ldots, T_s\}$.

Step 3: Map and assign the indices in CI to the indices in TH. Concretely, for the i^{th} index in TH, represented by TH(i) $= T_i$, l_i candidate indices in CI are selected and assigned to it, where i is also called as the serial number corresponds to T_i and ranged in [1, N]. It is obvious that the candidate indices can be assigned with an equal approach, i.e., $l_1 = l_2 = \ldots = l_{\mathrm{N}}$, to achieve a considerable embedding capacity. However, more hiding capacity can be provided by an adaptive assignment approach. That is, for the aim of achieving the high embedding capacity, a greater group length l_i should be assigned to a to-be-mapped index TH(i) $= T_i$ with higher frequency and vice versa. Based on this principle, the value of l_i can be determined as

$$l_i = \arg\max_{v=1}^{u_i} \left\{ 2^v - 1 \,\middle|\, (2^v - 1) < \mathrm{T_{SH}} \,\&\&\, (\mathrm{N} - i) < \mathrm{T_{SH}} \right\}, \quad (2)$$

where

$$u_i = \left\lceil \log_2 \left((s-2) \cdot \frac{f_i}{\sum_{i=1}^{\mathrm{N}} f_i} \right) \right\rceil, \quad (3)$$

$$\mathrm{T_{SH}} = |\mathrm{CI}| - \sum_{j=1}^{i-1} l_j. \quad (4)$$

Here $|\mathrm{CI}|$ is the number of indices in CI, and T_{SH} represents the remaining un-assigned indices in TH, and f_i is the occurrence frequency of index TH(i), $[\cdot]$ represents the operation of round, and u_i is an integer number. Besides, the following constraint condition in Eq. (5) should be satisfied.

$$\sum_{i=1}^{\mathrm{N}} l_i \le (s - \mathrm{N} - 2). \quad (5)$$

Step 4: After l_i is determined, a mapping table (See TABLE 7) is constructed among the corresponding indices in TH and CI. The main principle used for assigning which candidate indices to each T_i is based on the ascending order of the occurrence frequency of indices in CI; the lower frequency indices are priority to be assigned to the T_i with higher frequency and vice versa. Here, CI $= \{\mathrm{CI}(1), \mathrm{CI}(2), \ldots, \mathrm{CI}(\mathrm{N})\}$, and $\mathrm{CI}(1) \cap \mathrm{CI}(2) \cap \ldots \cap \mathrm{CI}(\mathrm{N}) = \emptyset$. It should be noted that both the standard-index based mapping table and the difference-index based mapping table have the same construction strategy.

To explain the process of constructing mapping table better, an example is given and shown in Fig. 8, where the standard-index table is sized of 4 × 4, and codebook size $s = 16$.

TABLE 7. Mapping table construction.

Serial number: i	Group length: l_i	To-be-mapped indices: TH(i)=T_i	Candidate indices group: CI(i) ⊂ CI
1	l_1	T_1	$\mathrm{CI}(1) = \{\mathrm{CI}_{(1,1)}, \mathrm{CI}_{(1,2)}, \ldots, \mathrm{CI}_{(1,l_1)}\}$
2	l_2	T_2	$\mathrm{CI}(2) = \{\mathrm{CI}_{(2,1)}, \mathrm{CI}_{(2,2)}, \ldots, \mathrm{CI}_{(2,l_2)}\}$
	⋮	⋮	⋮
N	l_{N}	T_{N}	$\mathrm{CI}(\mathrm{N}) = \{\mathrm{CI}_{(\mathrm{N},1)}, \mathrm{CI}_{(\mathrm{N},2)}, \ldots, \mathrm{CI}_{(\mathrm{N},l_{\mathrm{N}})}\}$

2	3	4	8
9	7	5	4
3	3	4	3
2	3	4	2

FIGURE 8. An example of standard-index table.

TABLE 8. An example of mapping table based on standard-index table.

Serial number: i	Group length: l_i	To-be-mapped indices: TH(i)	Candidate indices group: CI(i) $\subset$ CI
1	7	3	15, 14, 13, 12, 11, 10, 6
2	3	4	9, 8, 5
3	1	2	7

From Fig. 8, we first can derive the sorted codebook T = {3, 4, 2, 7, 5, 8, 9, 6, 10, 11, 12, 13, 14, 15, 0, 1} according to their frequencies {5, 4, 3, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0}. Here, assume N = 3, thus, the to-be-mapped group TH consists of the first 3 indices in T, i.e., TH = {3, 4, 2}. Then, the remaining indices make up the candidate group CI = {7, 5, 8, 9, 6, 10, 11, 12, 13, 14, 15, 0, 1}. It is worth note that TL = {7, 5, 8, 9}. According to the aforementioned assignment strategy, we can derive the $l_1 = 7$, $l_2 = 3$, and $l_3 = 1$. TABLE 8 gives the illustration of mapping table. As can be seen, for the 1st index 3 in TH, it is assigned $l_1 = 7$ indices from CI, i.e., CI(1) = {15, 14, 13, 12, 11, 10, 6}. As a result, for each index 3 in standard-index table, it can be used to carry $\log_2(1+l_1) = 3$ secret bits. Similarly, the indices 4 and 2 can represent $\log_2(1+l_2) = 2$ and $\log_2(1+l_3) = 1$ secret bits, respectively. Besides, it also can be observed that indices 0 and 1 are not used in this example. That is because they are reserved as the flag indices during the embedding phase, and used to indicate whether an encoded index is the original index itself or not. Note that, in fact, flag indices do not have to be 0 or 1; any two indices that do not appear in the index table can be selected as flag indices, and these flag indices cannot be included in CI set.

2) CONSTRUCTION OF MAPPING TABLE BASED ON DIFFERENCE INDEX

As Subsection III-A mentioned, the difference index of an image has significantly high peaks of the distribution of difference indices. This means that more elements will be classified into the to-be-mapped group under the same conditions, so its embedding capacity is higher than that of standard-index based method. Details on constructing a mapping table based on difference-index table are described as follows.

Step 1: Construct the difference-index table as mentioned in Subsection III-A. Then the frequency of each difference index is counted and all of difference indices are also sorted by a descending order according to their frequencies. For convenience, we assume the sorted difference indices DT = {$D_1, D_2, \ldots, D_{s'}$}.

Step 2: Separate difference indices into two groups, i.e., to-be-mapped group and the candidate group. Here, the to-be-mapped group includes the first N difference indices with highest frequency and denoted as DTH = {$D_1, D_2, \ldots, D_N$}. The candidate group makes up of the difference indices with low frequencies DTL = {$D_{N+1}, D_{N+2}, \ldots, D_{m'}$} and the indices with frequency of zero $DT_{nonappear}$ = {$DT_{m'+1}, \ldots, DT_{s'}$}, where m' is the number of indices without zero in frequency.

Step 3: Assign the difference indices in the candidate group into the difference indices in the to-be-mapped group, resulting in a mapping relation between those two groups.

Fig. 9(a) gives an example of difference-index table deriving from Fig. 8. Considering that it is customary to use a non-negative integer to represent the value in the index table, we add an offset to the value of each difference index except the first index 2. The result is depicted in Fig. 9(b). Here, we also set N = 3. Thus, we can derive DTH = {16, 14, 13}, and the remaining difference indices are assigned to those three elements by the same way as TABLE 7. TABLE 9 gives one of mapping table, where 0 and 1 are also used as flag indices and do not appear in the mapping table.

It should be noted that the assignment of indices of the candidate group in TABLE 8 and TABLE 9 can be selected with flexibility. Concretely, we can select all indices (difference indices) in CI as the elements in candidate group, or only select the indices (difference indices) in $T_{nonapaper}$ as elements in candidate group, or select the indices (difference indices) in $T_{nonapaper}$ along with part of indices (difference indices) in TL (DTL) as elements in candidate group. Generally, the first case has the highest embedding capacity, and

2	1	1	4
1	-2	-2	-1
-1	0	1	-1
-1	1	1	-2

(a)

2	16	16	19
16	13	13	14
14	15	16	14
14	16	16	13

(b)

FIGURE 9. An example of difference-index table (a) without offset. (b) with offset = 15.

TABLE 9. An example of mapping table based on difference-index table (with offset).

Serial number: i	Group length: l_i	To-be-mapped indices: DTH(i)	Candidate indices group: CI(i) $\subset$ CI
1	15	16	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,17,18, 20, 21
2	7	14	22, 23, 24, 25, 26, 27, 28
3	3	13	29, 30, 19

for the last two cases, there is a little degree of decrement in the embedding capacity, but the bitrate can be reduced so that the compression rate can be kept at a satisfactory level.

C. DATA EMBEDDING

In this section, the embedding procedures are described in detail, including three aspects: the standard-index based method, the difference-index based method, and the joint method.

1) STANDARD-INDEX BASED METHOD

After constructing the mapping table, for i-th index in the to-be-mapped group, it is assigned l_i candidate indices. Hence, this index can be used to represent a $\lfloor \log_2(1+l_i) \rfloor$-bit secret data. The algorithm of data embedding using standard-index based method can be described as **Algorithm 1**, where '||' represents the operation of cascade and CZ is the codebook size.

Note that, the notation $(0)_{cz}$ and $(1)_{cz}$ denote the binary representation of value 0 or 1 with the length of $\log_2 CZ$ bits. Let us focus on the example in Fig. 8. Based on this example, the detailed process of embedding secret data into this standard-index table is demonstrated in Fig. 10, where we suppose the random secret bitstream $S = (01111110110000011001110011110 1\ldots)_2$. For the first index TH(1) = 2, it is obvious from TABLE 8 that 2 belongs to the to-be-mapped group TH = {3, 4, 2}, and it is assigned one candidate index 7, i.e., CI(3) = {7}, where the serial number $i = 3$. Thus, it can be used to carry $\log_2(1+1) = 1$ bit secret data. Due to the current secret bit is $(0)_2$, and its decimal value is 0, so this index 2 remains unchanged. On the contrary, if the current secret bit is $(1)_2$, since its decimal value is 1, this index 2 should be changed to the 1st index of candidate index group CI(3), i.e. index 7, for the aim to carrying secret data $(1)_2$. Let us move on to the next index 3. Obviously, it belongs to TH and its corresponding candidate index group is CI(1) = {15, 14, 13, 12, 11, 10, 6}, so index 3 can be used to carry $\log_2(1+7) = 3$ bits secret data. Thus, we fetch the following 3-bit secret bits $(111)_2$ from S. Since its decimal value is 7, this index 3 is then mapped to the 7th candidate index in CI(1) (i.e., 6) to carry secret bits $(111)_2$. By the same way, the third index 4 is mapped to the 3th candidate index in CI(2) (i.e., 5) to represent the 2-bit secret bits $(11)_2$.

Take another specific case for example. Certainly, the fourth index 8 is not in TH. Moreover, it not only acts as an index in standard-index table but also as a candidate index in CI(2). To avoid conflict between those two cases, we will cascade a flag index with this standard index 8 as the stego standard index, so that we can identify the index is an original standard index during the decoding phase. As shown in TABLE 8, we have reserved digits 0 and 1 as flag indices.

Algorithm 1 Data Embedding Using Standard-Index Based Method

Input: Secret bitstream S = $\{s_k\}$, Standard-index table IT, group number N
Output: Stego standard-index table SIT
1. Initialization: SIT = empty, $loc = 1$
2. Construct a mapping table MT, including the following items:
  TH(i): the i-th index in TH, i is its serial number and $1 \le i \le$ N;
  l_i: the number of candidate indices assigned to TH(i);
  CI(i): the candidate indices group corresponds to TH(i);
  CI(i, j): the j-th candidate index of CI(i).
3. For $k = 1$ to size of IT
  if IT(k) is not in TH // IT(k) is the k-th index of IT
    Get 1 bit (s_{loc}) from S;
    if $s_{loc} = (0)_2$
      SIT(k) = $(0)_{cz}$|| IT(k); // $(0)_{cz}$ is flag index
    else
      SIT(k) = $(1)_{cz}$|| IT(k); // $(1)_{cz}$ is flag index
    end
    $loc = loc + 1$;
  else if IT(k) is in TH
    Get the serial number i of IT(k) in TH;
    Get $\log_2(l_i + 1)$ bits p from S and convert it into decimal form p_d;
    if $p_d = 0$
      SIT(k) = IT(k);
    else
      SIT(k) = CI(i, p_d);
    end
    $loc = loc + \log_2(l_i + 1)$;
  end
end

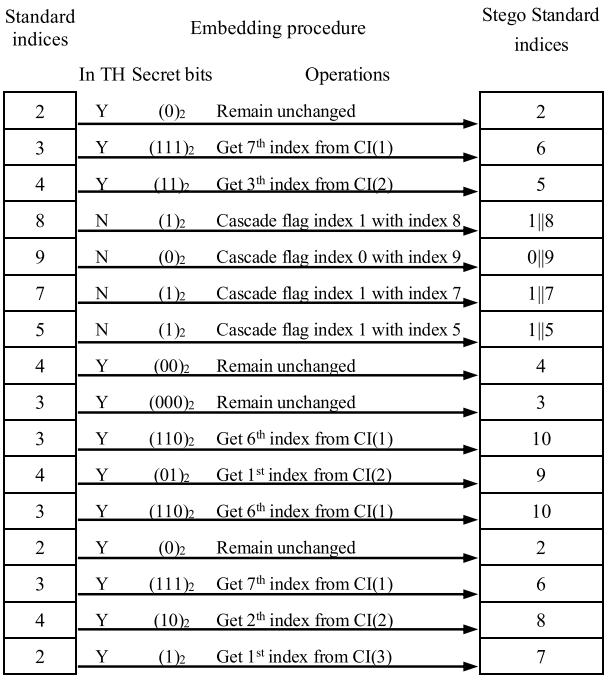


FIGURE 10. An example of data embedding using standard-index based method.

2	6	5	1||8
0||9	1||7	1||5	4
3	10	9	10
2	6	8	7

FIGURE 11. An example of stego standard-index table corresponds to Fig. 8.

As a result, one secret bit can be carried by combining the flag index and index 8. That is, if the to-be-embedded secret bit is $(1)_2$, we use 1||8 as the stego standard index; otherwise, it is replaced as 0||8.

After completely embedding, the corresponding stego standard-index table is shown in Fig. 11, and the sequence of stego standard indices SIT = {2, 6, 5, 1, 8, 0, 9, 1, 7, 1, 5, 4, 3, 10, 9, 10, 2, 6, 8, 7}. Also, 30 bits secret data, i.e., '01111110110000011001110011101', were embedded. Besides, in order to obtain better bitrate, the LAS compression will be employed to reduce the length of the stego standard indices.

2) DIFFERENCE-INDEX BASED METHOD

Similarly, the algorithm of data embedding using difference-index based method can be described as **Algorithm 2**.

Here, the notation $(0)_{cz'}$ and $(1)_{cz'}$ denote the binary representation of value 0 or 1 with the length of $\log_2 CZ + 1$ bits. Let us consider again the example of the previous Subsection III-B-2 and assume S = (01111110110000011001 1100111101011001011100 00...)$_2$. Fig. 12 gives the process

Difference indices	In TH	Secret bits	Operations	Stego Difference indices
2	-	-	Basic index, remain unchanged	2
16	Y	$(0111)_2$	Get 7th index from CI(1)	8
16	Y	$(1110)_2$	Get 14th index from CI(1)	20
19	N	$(1)_2$	Cascade flag index 1 with index 19	1||19
16	Y	$(1000)_2$	Get 8th index from CI(1)	9
13	Y	$(00)_2$	Remain unchanged	13
13	Y	$(11)_2$	Get 3th index from CI(3)	19
14	Y	$(001)_2$	Get 1th index from CI(2)	22
14	Y	$(110)_2$	Get 6th index from CI(2)	27
15	N	$(0)_2$	Cascade flag index 0 with index 15	0||15
16	Y	$(1111)_2$	Get 15th index from CI(1)	21
14	Y	$(010)_2$	Get 2th index from CI(2)	23
14	Y	$(110)_2$	Get 6th index from CI(2)	27
16	Y	$(0101)_2$	Get 5th index from CI(1)	6
16	Y	$(1100)_2$	Get 12th index from CI(1)	17
13	Y	$(00)_2$	Remain unchanged	13

FIGURE 12. Embedding process based on Algorithm 2.

Algorithm 2 Data Embedding Using Difference-Index Based Method

Input: Secret bitstream S = $\{s_k\}$, Difference-index table DT, group number N
Output: Stego difference-index table SDT
1. Initialization: SDT= empty, $loc = 1$
2. Construct a mapping table MT, including the following items:
   DTH(i): the i-th index in DTH, i is its serial number and $1 \le i \le$ N;
   l_i: the number of candidate indices assigned to DTH(i);
   CI(i): the candidate indices group corresponds to DTH(i);
   CI(i, j): the j-th candidate index of CI(i).
3. For $k = 1$ to size of DT
   if DT(k) is not in DTH // DT(k) is the k-th index of DT
      Get 1 bit (s_{loc}) from S;
      if $s_{loc} = (0)_2$
         SDT(k) = $(0)_{cz'}$|| DT(k); // $(0)_{cz'}$ is flag index
      else
         SDT(k) = $(1)_{cz'}$|| DT(k); // $(1)_{cz'}$ is flag index
      end
      $loc = loc + 1$;
   else
      Get the serial number i of DT(k) in DTH;
      Get $\log_2(l_i + 1)$ bits p from S and convert it into decimal form p_d;
      if $p_d = 0$
         SDT(k) = DT(k);
      else
         SDT(k) = CI(i, p_d);
      end
      $loc = loc + \log_2(l_i + 1)$;
   end
end

of embedding S into the difference-index table. It can' be observed from Fig. 12 that, the process of data embedding is similar to the previous example in Fig. 10. The difference is just the first index 2 is only used as the basic index hence remain unchanged and does not carry any secret bits.

After completely embedding, the corresponding stego difference-index table is shown in Fig. 13, and the sequence of stego difference indices SDT = {2, 8, 20, 1,19, 9, 13, 19, 22, 27, 0, 15, 21, 23, 27, 6, 17, 13}, which carries 44 bits secret data. compared to standard-index based method, we can see that, the embedding capacity provided by the difference-index based method is 14 bits more than that of the previous standard-index based method. It is concluded that the embedding capacity performance of the system has been improved.

3) JOINT METHOD

To exploit further the advantages of the above methods fully, a joint method that combines the standard-index based

2	8	20	1||19
9	13	19	22
27	0||15	21	23
27	6	17	13

FIGURE 13. An example of stego difference-index table corresponds to Fig. 9(b).

2	4	-1	1||8
0||9	1||7	1||5	-1
-1	7	-1	1
-8	4	2	-1

(a)

2	19	14	1||8
0||9	1||7	1||5	14
14	22	14	16
7	19	17	14

(b)

FIGURE 14. Examples of Difference-index table for Stage 2. (a) Without offset; (b) With offset.

method and the difference-index based method is presented, where the two-layer embedding process is performed, resulting in a significant improvement of the embedding capacity. The algorithm of data embedding using joint method includes two stages and can be described as **Algorithm 3**. Herein, we must point out that all indices used as flag indices (i.e., 0 or 1) in **Stage 1** will be skipped during the embedding processes of **Stage 2**.

Here, the notation $(2)_{cz'}$ and $(3)_{cz'}$ denote the binary representation of value 2 or 3 with the length of $\log_2 CZ+1$ bits. We still use the previous example in Subsection III-C-1 and assume the secret data S = $(01111110110000011001$ $1100111101011001011100000101010110100011\ldots)_2$. The process of **Stage 1** is the same as Subsection III-C-1 and the stego standard-index table SIT is shown in Fig. 11. Based on SIT, we reconstruct the difference-index table as shown in Fig. 14 and the mapping table as shown in TABLE 10. It should be noted that indices 0 and 1 are used for flag indices in **Stage 1**, and indices 2 and 3 are used as the flag indices in **Stage 2**.

After the execution of **Stage 1**, the first 30 bits of secret bitstream S have been embedded. Sequentially, the remaining secret bits in S continue to be processed in the following **Stage 2**. The detailed process of embedding is shown in Fig. 15. For the first index 2, it is treated as the basic index and remains unchanged. For the second index 19, according to TABLE 10, it can be found that index 19 belongs to the to-be-mapped group DTH = {14, 19, 17}, thus, the $\log_2(1+7) =$ 3-bit secret bits $(011)_2$ can be concealed by changing index from 19 to 27. Take another specific case for example. The 4^{th} index is a combined index 1||i8, where the index 1 is the flag index in **Stage 1**, so this combined index does not used to carry any secret bit in **Stage 2**. For the 10^{th} index 22, it is not in DTH, and it not only acts as an index in

Algorithm 3 Data Embedding Using Joint Method

Input: Secret bitstream S = $\{s_k\}$, Standard-index table IT, group number N
Output: Stego joint-index table SJT
Stage 1:
Perform **Algorithm 1**, and get the stego standard-index table SIT.
Stage 2:
1. Initialization: SJT = empty, $loc = 1$.
2. Construct difference-index table DT based on SIT, and then construct a mapping table MT, including the following items:
   DTH(i): the i-th index in DTH, i is its serial number and $1 \le i \le$ N.
   l_i: the number of candidate indices assigned to DTH(i);
   CI(i): the candidate indices group corresponds to DTH(i).
   CI(i, j): the j-th candidate index of CI(i).
3. For $k = 1$ to size of DT
   if DT(k) = 0 or 1 // 0, 1 are the flag indices in **Stage 1**
      $k = k+2$; continue;
   end
   if DT(k) is not in DTH // DT(k) is the k-th index of DT
      Get 1 bit (s_{loc}) from S;
      if $s_{loc} = (0)_2$
         SJT(k) = $(2)_{cz'}$|| DT(k); // $(2)_{cz'}$ is flag index
      else
         SJT(k) = $(3)_{cz'}$|| DT(k); // $(2)_{cz'}$ is flag index
      end
      $loc = loc + 1$;
   else
      Get the serial number i of DT(k) in DTH;
      Get $\log_2(l_i + 1)$ bits p from S and convert it into decimal value p_d;
      if $p_d = 0$
         SJT(k) = DT(k);
      else
         SJT(k) = CI(i, p_d);
      end
      $loc = loc + \log_2(l_i + 1)$;
   end
end

TABLE 10. Mapping table construction for stage 2 of joint method.

Serial number: i	Group length: l_i	To-be-mapped indices: DTH(i)	Candidate indices group: $CI_i \subset CI$
1	15	14	4, 5, 6, 8, 9, 10, 11, 12, 13, 15, 18, 20, 21, 23, 24
2	7	19	25, 26, 27, 28, 29, 30, 22
3	1	17	16

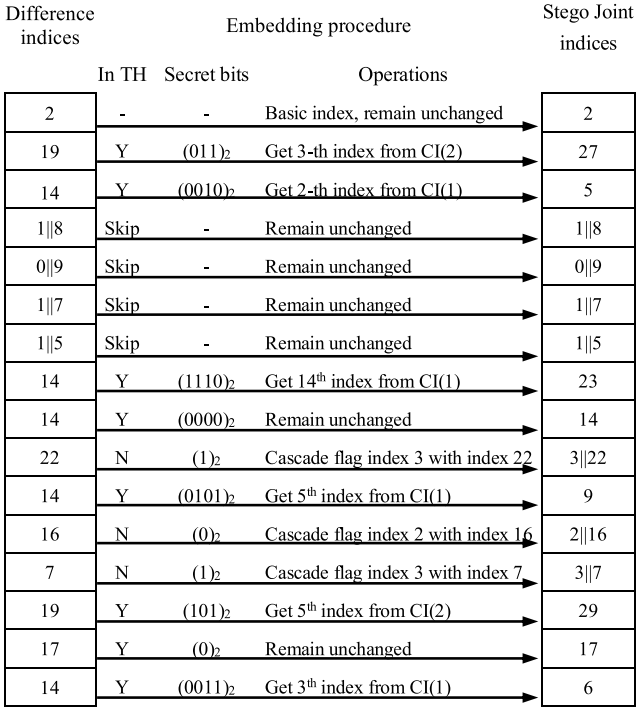


FIGURE 15. An example of embedding process for Stage 2 of joint method.

2	27	5	1||8
0||9	1||7	1||5	23
14	3||22	9	2||16
3||7	29	17	6

FIGURE 16. An example of stego joint-index table SJT.

difference-index table but also as a candidate index in CI(2). Hence, we cascade a flag index with index 22 as the stego difference index. Since the current secret bit is $(1)_2$, we use 1||22 as the stego difference index. Finally, in this stage, there are 30 bits secret data, i.e., "0110010111000001 01010110100011," are carried.

Fig. 16 shows the stego joint-index table SJT after combining the results of **Stage 1** and **Stage 2**. In summary, the proposed joint method embeds 60 bits of secret data in total, which is around 2.0 times than those of both the standard-index based method and the difference-index based method. In other words, the joint method has a superior performance in embedding capacity among those three methods.

Moreover, it is interesting to note that, the choice of candidate indices in the joint method can also be flexible. For example, it can be divided into the following cases according to the different choices of indices to constructing the candidate indices group CI in **Stages 1** and **2** of Algorithm 3.

Case 1: In **Stages 1** and **2**, both TL and $T_{nonappear}$ group indices are chosen.

Case 2: Only the $T_{nonappear}$ indices are chosen in **Stage 1**, and both TL and $T_{nonappear}$ group indices are chosen in **Stage 2**.

Case 3: Only the $T_{nonappear}$ group indices are chosen in both **Stage 1** and **Stage 2**.

Case 1 usually has the highest data embedding capacity, while Case 2 and Case 3, a little lower, but can reduce the bitrate so that can achieve better compression ratio. The related discussions will be introduced in Subsection IV-B.

D. DATA EXTRACTION AND IMAGE RECOVERY

When the recipient receives the VQ compression codes, s/he can decode the corresponding index table. Then, the data extraction and image recovery corresponding to above three methods can be performed, and details are described as follows.

1) STANDARD-INDEX BASED METHOD

The algorithm of data extraction corresponding to **Algorithm 1** can be described as **Algorithm 4**.

Take a specific case for example. Here, let us assume that the decoded sequence of stego standard indices SIT = {2, 6, 5, 1, 8, 0, 9, 1,7, 1, 5, 4, 3, 10, 9, 10, 2, 6, 8, 7} and the mapping

Algorithm 4 Data Extraction for Standard-Index Based Method

Input: Received VQ compression codes, Mapping table MT

Output: Standard-index table IT, Secret bitstream S

1. Decode the sequence of stego standard indices SIT from received VQ compression codes. Get the group number N.
2. Initialization: IT = empty, S = empty, $t = 1$, $k = 1$;
3. While $k \leq$ size of SIT
   if SIT(k) is 0
      S = S || $(0)_2$; IT(t) = SIT($k+1$);$t = t+1$; $k = k+2$;
   else if SIT(k) is 1
      S = S || $(1)_2$; IT(t) = SIT($k+1$); $t = t+1$; $k = k+2$;
   else
      for $i = 1$ to N
         if SIT(k) belongs to CI(i)
            S = S || b(j), where j makes SIT(k) == CI(i, j);
            IT(t) = TH(i); $t = t+1$; $k = k+1$;
         else if SIT(k) == TH(i)
            S = S || b($j = 0$), where $j = 0$ makes SIT (k) == TH(i);
            IT(t) = TH(i); $t = t+1$; $k = k+1$;
         End
         end
      end
   End
4. Output IT and S.

Notice: b(j) is the binary representation of j with length of $\log_2(l_i + 1)$ bits.

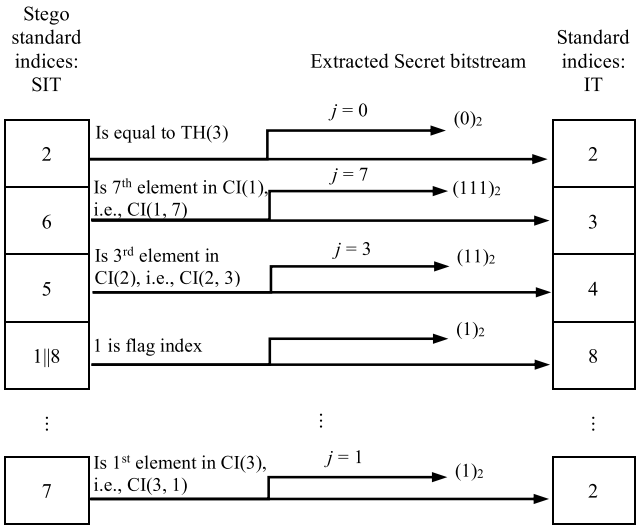


FIGURE 17. Data extraction and standard indices recovery for standard-index based method.

table is the same as TABLE 8, where TH = {3, 4, 2}. The process of data extraction and standard indices recovery are shown in Fig. 17. As can be seen, the first index in SIT is 2, which is just equal to TH(3). Thus, one binary secret bit $(0)_2$ is extracted and the original standard index 2 is also recovered. Similarly, the second index 6 is the 7th element in CI(1), so we can extract a 3-bit secret data $(111)_2$ and recover the original standard index as TH(1) = 3. By the same way, a 2-bit secret data $(11)_2$ and the corresponding index 4 can be derived from the third stego standard index. Next, the stego standard index 1 is encountered, which is a flag index and indicates that the following index 8 is an original standard index. As a result, the 4th original standard index is set to 8. Meanwhile, one bit secret data $(1)_2$ can be extracted because of the flag index is 1. After all stego standard indices have been processed, the secret bitstream can be extracted correctly and the original standard-index table can be recovered losslessly.

2) DIFFERENCE-INDEX BASED METHOD

The algorithm of data extraction corresponding to **Algorithm 2** can be described as **Algorithm 5**.

Similarly, the data extraction and standard-index table recovery corresponding to example in Subsection III-C-2 is shown in Fig. 18, where the received sequence of stego difference indices SDT = {2, 8, 20, 1,19, 9, 13, 19, 22, 27, 0, 15, 21, 23, 27, 6, 17, 13} and the mapping table is the same as TABLE 9, where TH = {16, 14, 13}.

From Fig. 18, we can observe that the first index in SDT is 2, which is skipped and remains unchanged due to it is the basic index. The second index 8 is the 7th element in CI(1), so we can extract a 4-bit secret data $(0111)_2$ and recover the original difference index as TH(1) = 16. By the same way, a 4-bit secret data $(1110)_2$ and the corresponding index 16 can be derived from the third stego difference index. Next, the stego difference index 1 is encountered, which is a flag

Algorithm 5 Data Extraction for Difference-Index Based Method

Input: Received VQ compression codes, Mapping table MT,
Output: Standard-index table IT, Secret bitstream S
1. Decode the sequence of stego difference indices SDT from received VQ compression codes. Get the group number N.
2. Initialization: DT = empty, S = empty, DT(1) = SDT(1), $t = 2$, $k = 2$.
3. While $k \leq$ size of SDT
  if SDT(k) is 0
    S = S || $(0)_2$; DT(t) = SDT(k+1);$t = t+1$;
  $k = k + 2$;
  else if SDT(k) is 1
    S = S || $(1)_2$; DT(t) = SDT(k+1);$t = t+1$;
  $k = k + 2$;
  else
    for $i = 1$ to N
      if SDT(k) belongs to CI(i)
        S = S || b(j), where j makes
        SDT(k) == CI(i, j);
        DT(t) = TH(i);$t = t+1$; $k = k + 1$;
      else if SDT(k) == TH(i)
        S = S || b($j = 0$), where j makes
      SDT(k) == TH(i);
      DT(t) = TH(i);$t = t+1$;$k = k + 1$;
      end
    end
  end
4. Recover the standard-index table IT from DT.
5. Output IT and S.

Notice: b(j) is the binary representation of j with length of $\log_2(l_i + 1)$ bits.

index and indicates that the following index 19 is an original difference index. Meanwhile, one bit of secret data $(1)_2$ can be extracted because of the flag index is 1.

After all stego difference indices have been processed, the embedded secret data bitstream "011111…1000" can be extracted correctly, and the difference-indices sequence {2, 16, 16, 19, …, 13} is recovered. Sequentially, it can be further recovered to the original standard-index table (see Fig. 8) losslessly using the inverse difference transformation.

3) JOINT METHOD

The algorithm of data extraction corresponding to **Algorithm 3** can be described as **Algorithm 6**.

By the same way, we assume that the decoded sequence of stego joint indices SJT = {2, 27, 5, 1, 8, 0, 9, 1, 7, 1, 5, 23, 14, 3, 22, 9, 2, 16, 3, 7, 29, 17, 6} and the mapping table for **Stages 1** and **2** are the same as TABLE 8 and TABLE 10, respectively. The process of **Stage 1** is shown in Fig. 19.

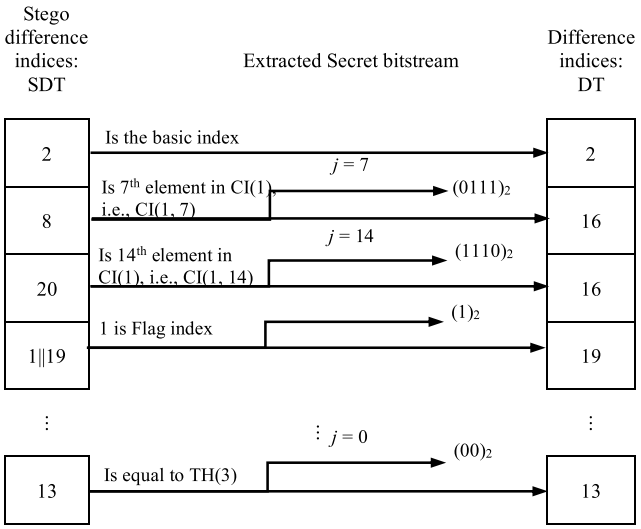


FIGURE 18. Data extraction and recovery for difference-indexbased method.

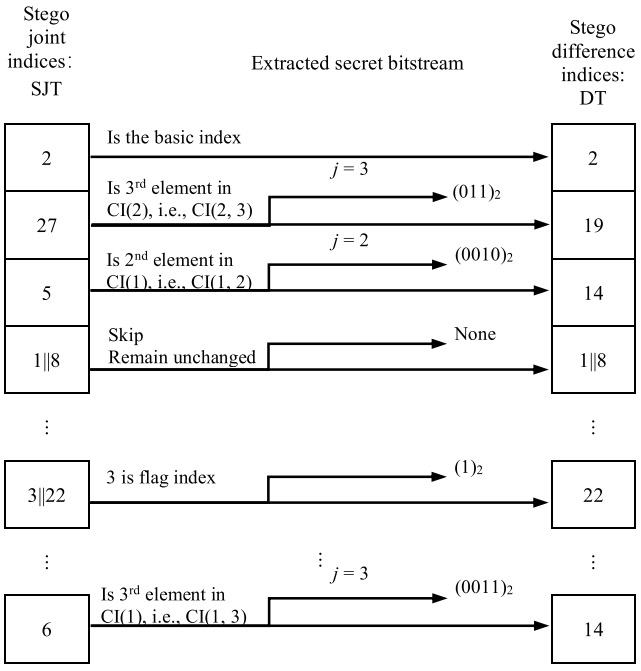


FIGURE 19. Data extraction and recovery for Stage 1 of joint method.

First of all, the 1st index in SJT is 2, it remains unchanged since it is the basic index, and there is no secret bit extracted. The second index 27 is the 3th element in CI(2), so a 3-bit secret data $(011)_2$ is extracted and recovers the original difference index as TH(2) = 19. In the same way, a 4-bit secret data $(0010)_2$ and the corresponding index 14 can be derived from the third stego joint index. Next, a flag index is encountered, i.e. 1, indicating this combined index 1||8 should be skipped in this stage. In the following, let us focus on the fifth row in Fig. 19, where a combined index 3||22 is encountered. Thus, 1-bit secret data $(1)_2$ is extracted and the following index 22 is set as its original difference index. Finally, after **Stage 1** completed, the recovered stego difference indices DT is recovered as '2, 19, 14, 1, 8, 0, 9, 1, 7, 1, 5, 14, 14, 22, 14, 16, 7, 19, 17, 14', and the extracted secret bitstream S is '0110010111000000101 010110100011'.

Sequentially, the process of **Stage 2** is performed and details are shown in Fig. 20. First, In **Stage 2**, the valid index in DT minus the offset (i.e. 15), then transform it into the stego standard indices sequence SIT. Then, we can further perform the decoding as mentioned in Algorithm 4, and the original standard-index table IT is recovered error-free, and the secret bitsream S is extracted successfully.

Algorithm 6 Data Extraction for Joint Method

Input: Received VQ compression codes, Mapping table MTD for difference-index based embedding method, Mapping table MTS for standard-index based embedding method.

Output: Standard-index table IT, Secret bitstream S

1. Decode the sequence of stego difference indices SJT from received VQ compression codes. Get the group number N.
2. Initialization: IT = empty, SIT= empty, S = empty, $t = 1$, DT = empty;

Stage 1:

3. DT (1) = SDT(1); $k = 2$;
   While $k \leq$ size of SJT
   if SJT(k) is 0 or 1
   DT (t) = SJT(k); DT ($t + 1$) = SJT($k + 1$);
   $t = t + 2$; $k = k + 2$;
   else if SJT(k) is 2
   S = S || $(0)_2$; DT (t) = SJT(k+1);$t = t + 1$;
   $k = k + 2$;
   else if SDT(k) is 3
   S = S || $(1)_2$; DT (t) = SJT(k+1);$t = t + 1$;
   $k = k + 2$;
   else
   for $i = 1$ to N
   if SJT(k) belongs to CI(i) in MTD
   S = S || b(j), where j makes SDT (k) == CI(i, j);
   DT(t) = DTH(i);$t = t$+ 1;
   else if SDT(k) == DTH(i)
   S = S || b($j = 0$), where j makes SDT (k) == DTH(i);
   DT(t) = DTH(i);$t = t + 1$;
   end
   end
   end
   end
4. Recover the Stego standard-index table SIT from DT

Stage 2:

5. Perform Algorithm 4, and get the original standard-index table IT from SIT, and recover the following secret data.
6. Output IT and S.

Notice: b(j) is the binary representation of j with length of $\log_2(l_i + 1)$ bits.

Output indices of Algorithm 2.3-**Stage 1**: DT (with offset)	-offset (15)	Stego difference indices: DT (without offset)	Stego indices: SIT	Algo-rithm 2.1	Output of Algorithm 2.3-**Stage 2**	
					Standard index: IT	Secret bits of **Stage 2**
2		2	2		2	$(0)_2$
19		4	6		3	$(111)_2$
14		-1	5		4	$(11)_2$
1‖8		1‖8	1‖8		8	$(1)_2$
⋮		⋮	⋮		⋮	⋮
14		-1	7		2	$(1)_2$

FIGURE 20. Data extraction and recovery for Stage 2 of joint method.

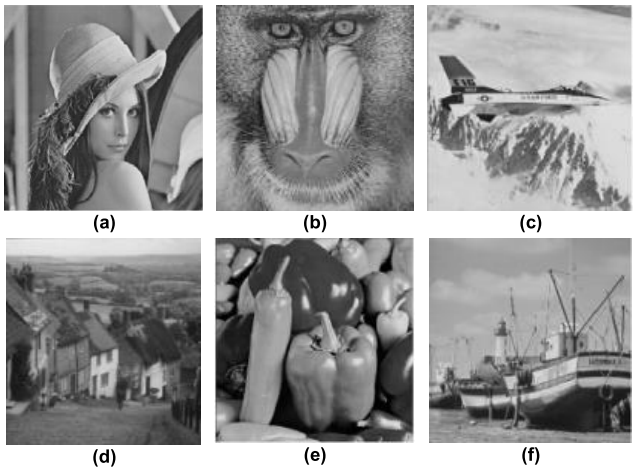


FIGURE 21. Six gray-scale test images. (a) Lena. (b) Baboon. (c) Airplane. (d) Goldhill. (e) Peppers. (f) Boat.

It should be pointed out that the mapping table should be sent to the receiver as the auxiliary information. Inspired by reference [3], in this paper, we used the scheme that is to embed the mapping table before the secret data through using a generic mapping table, followed by embedding the secret data by using the specific mapping table. Here, the specific mapping table means it constructed based on the Standard-index based histogram or Difference-index based histogram of each image, and the generic mapping table means a predefined and pre-shared mapping table.

IV. EXPERIMENTAL RESULTS

In this section, several experiments are implemented to demonstrate the effectiveness of the proposed difference-index based method and the joint method. Six gray-scale images with a size of 512 × 512, shown in Fig. 21, are served as test images. Meanwhile, the block size used in VQ compression is set to 4 × 4 pixels, and the size of codebook CZ is set to 256 and 512, respectively. Furthermore, the standard-index table is rearranged and the codebook is sorted accordingly by the AIR technique before embedding.

Besides, in our experiments, two widely used statistical metrics are employed to evaluate the performance, i.e. the embedding capacity (in bits) and the bitrate (in bpp, i.e., bits per pixel). Moreover, to evaluate the overall embedding efficiency of those VQ-based reversible data hiding schemes, we also use the embedding rate to further measure the ability of carrying secret bits of the developed data hiding scheme. The bitrate and embedding rate are defined as Eq. (6) and (7), respectively.

$$\text{Bitrate} = \frac{\text{Total number bits of VQ compression codes}}{\text{Total number bits of image}}, \quad (6)$$

$$\text{Embedding rate} = \frac{\text{Total number bits of embedded secret bitstream}}{\text{Total number bits of VQ compression codes}}. \quad (7)$$

Generally, the higher the embedding rate is, the better the embedding efficiency is, and the lower bitrate is, the higher compression efficiency has.

A. EXPERIMENTAL RESULTS OF PROPOSED DIFFERENCE-INDEX BASED METHOD

In this section, the experimental results provided by the standard-index based method and the difference-index based method are given and discussed. First of all, it is apparent that the different value of N will result in a different embedding capacity and efficacy. Moreover, the different assignment strategies between the to-be-mapped group and the candidate group make a significant effect in values of l_i s (the length of CI(i)), resulting in the different experimental results. Thus, two approaches of constructing a mapping table are evaluated. i.e., the equal approach, where the length of each candidate index group (i.e., l_i) is equal, and the adaptive approach, where the value of each l_i is determined by Eq. (2). TABLE 11 shows the results provided by standard-index based method and the difference-index based method for images Lena and Baboon, with respects to embedding capacity, bitrate and embedding rate, where codebook size CZ = 256 and N is set to 5, 15, and 30.

Firstly, compared to the standard-index based method, the proposed difference-index based method has significantly. improved in terms of embedding capacity, bitrate and embedding rate. That is because the former has a higher proportion of high-frequency indices and a more concentrated distribution in histogram of indices, which indicates a larger redundancy can be vacated for hiding more secret data. It also achieves a better satisfactory bitrate and embedding rate than that of standard-index based method. As an increment of N, the superior performance of the difference-index based method reveals significantly.

TABLE 11. Performance of standard-index based method and difference-index based method.

Image	Method		Group number N=5			Group number N=15			Group number N=30		
			Bitrate	Capacity	Embedding rate	Bitrate	Capacity	Embedding rate	Bitrate	Capacity	Embedding rate
Lena	Standard-index	Equal	0.574	29828	19%	0.561	36094	24%	0.549	36560	26%
		Adaptive		29849	19%		37254	24%		39896	26%
	Difference-index	Equal	**0.557**	65232	41%	**0.548**	61599	40%	**0.545**	55347	36%
		Adaptive		**67975**	**46%**		**73524**	**51%**		**73034**	**51%**
Baboon	Standard-index	Equal	0.575	30697	20%	0.564	34265	22%	0.554	33605	21%
		Adaptive		30809	20%		35714	23%		36690	24%
	Difference-index	Equal	**0.570**	42986	27%	**0.561**	45982	29%	**0.554**	45288	29%
		Adaptive		**46746**	**30%**		**51587**	**34%**		**53455**	**35%**

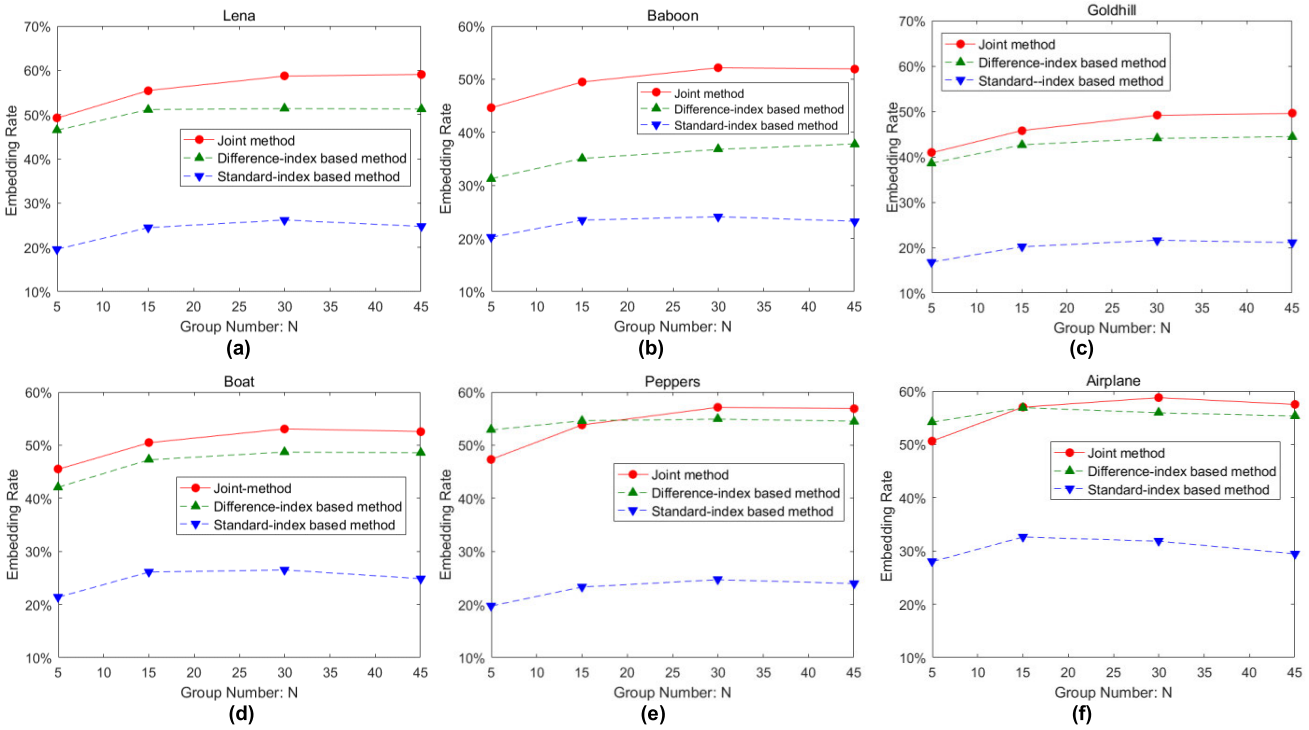


FIGURE 22. Embedding rate of standard-index based method, difference-index based method and the proposed joint method for images (a) Lena. (b) Baboon. (c) Gold hill. (d) Boat. (e) Peppers. (f) Airplane.

Secondly, we can observe that for those two methods, the adaptive approach always achieves a better performance than the equal approach. The reasons are two-fold: one is that more candidate indices are assigned to the to-be-mapped index with higher frequency in the adaptive approach; another one is that a larger number of indices are exploited to carry secret bits under the same condition (N). Therefore, the higher embedding capacity and embedding rate can be achieved at the same bitrate, especially in the difference-index based method.

B. EXPERIMENTAL RESULTS OF PROPSED JOINT METHOD

In this experiment, we demonstrate the performance of the proposed joint scheme and compare the results provided by the proposed joint scheme with the results provided by standard-index based method and the difference-index based method. Both **Stage 1** and **Stage 2** in joint method use the same N. TABLE 12 lists the performance of those three methods when the group number N is set to 5, 15, 30, and 45, respectively.

From TABLE 12, we can see that the embedding capacity provided by joint method are always higher than that of other two methods. When N = 30, the average embedding capacity reaches the maximum value of 96585 bits. Meanwhile, it is interesting to note that as the increasing of N, the embedding capacity is not always growing and vice versa it is a little decrement. That is mainly because that when N is large, the relative fewer candidate indices are assigned to the to-be-mapped index.

TABLE 12. Performance of the proposed joint method.

Image	Scheme	Group number N=5			Group number N =15			Group number N =30			Group number N=45		
		Standard -index	Differenc e-index	Joint	Standard-index	Differenc e-index	Joint	Standard -index	Differenc e-index	Joint	Standard -index	Differenc e-index	Joint
Lena	Equal	29828	65232	**83546**	36094	61599	**92828**	36560	55347	**95039**	37675	54646	**97673**
	Adaptive	29849	67975	**85689**	37254	73524	**93375**	39896	73034	**97927**	37856	72459	**99254**
Baboon	Equal	30697	42986	**80158**	34265	45982	**87710**	33605	45288	**87553**	34464	46995	**90772**
	Adaptive	30809	46746	**83644**	35714	51587	**88164**	36690	53455	**91972**	35374	54402	**91689**
Airplane	Equal	42348	75625	**88456**	45919	65215	**98769**	39722	56666	**100820**	39050	55135	**97777**
	Adaptive	42675	78484	**104924**	49350	81371	**105089**	48151	79404	**111135**	44563	78207	**107855**
Goldhill	Equal	25631	52672	**73567**	30140	53404	**81719**	31397	50035	**83729**	32210	50252	**86762**
	Adaptive	25659	57285	**77043**	30822	62126	**81806**	33498	63396	**86822**	32977	63487	**88763**
Peppers	Equal	30040	70413	**74794**	34234	63180	**84370**	34830	56358	**88720**	36476	55105	**87615**
	Adaptive	30060	76940	**82245**	35476	78318	**88494**	37569	77958	**95399**	36831	76985	**90431**
Boat	Equal	32507	57089	**81597**	37946	58504	**89947**	36927	54591	**92837**	37555	54208	**92095**
	Adaptive	32546	62118	**87291**	39722	68292	**92681**	40351	69376	**96255**	37852	68723	**93762**
Average	Equal	31842	60670	**80353**	36433	57981	**89224**	35507	53048	**91450**	36238	52724	**92116**
	Adaptive	31933	64925	**86806**	38056	69203	**91602**	39359	69437	**96585**	37576	69044	**95292**

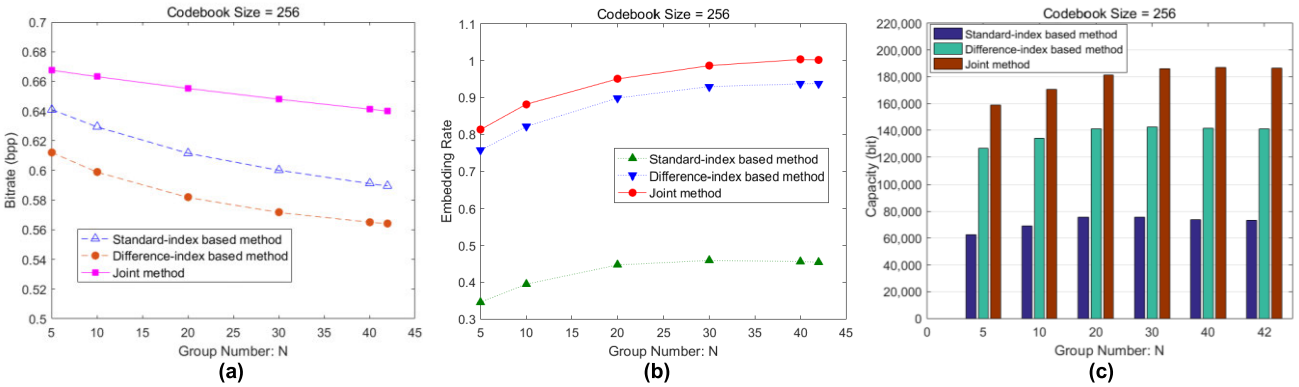


FIGURE 23. Performance of the index based method, the proposed difference-index based method, and the proposed joint method, on average, for 100 test images (codebook size = 256). (a) Bitrate. (b) Embedding Rate. (c) Embedding capacity.

Moreover, Fig. 22 shows the embedding rate provided by above three methods for images Lena, Baboon, Airplane, Goldhill, Peppers, and Boat. As can be seen from Fig. 22, the embedding rate of the proposed joint method is significantly higher than that of other two methods. This is because the proposed joint method has a greatly increasing in embedding capacity while keeps a relative fewer increment of VQ compression codes, so that a higher embedding rate can be obtained.

As can be seen from the above experimental results, the embedding capacity of the proposed joint method is not equal to the simple sum of the embedding capacity of the standard-index based method and the difference-index based method; it is usually a little less than that. That is because after the first embedding stage, the correlations between the neighboring indices in the mapped index table is decrement to some extent. However, in our experiments, the index values that belong to candidate groups are also sorted according to their occurrence frequency during the sort codebook phase, making the index values whose occurrence frequencies are close have as similar values as possible. Moreover, in the first embedding stage, for each index in the to-be-mapped group, the consecutive index values in sorted candidate groups are assigned to it. Thus, it has more opportunities to reserve some correlations between the neighboring indices in the mapped index table. Thus, the joint method can still maintain considerable redundancy in the second stage. As a result, the sum of the embedding capacity of the two stages in joint method can be greater than that obtained by either standard-index based method alone or difference-index based method alone.

Furthermore, we also experimented on 100 images from the widely used database UCID to verify the effectiveness of the proposed methods. The results are plotted in Fig. 23 and Fig. 24, which correspond to the codebook size CZ of 256 and 512, respectively. From Figs. 23 and 24, it can be apparent

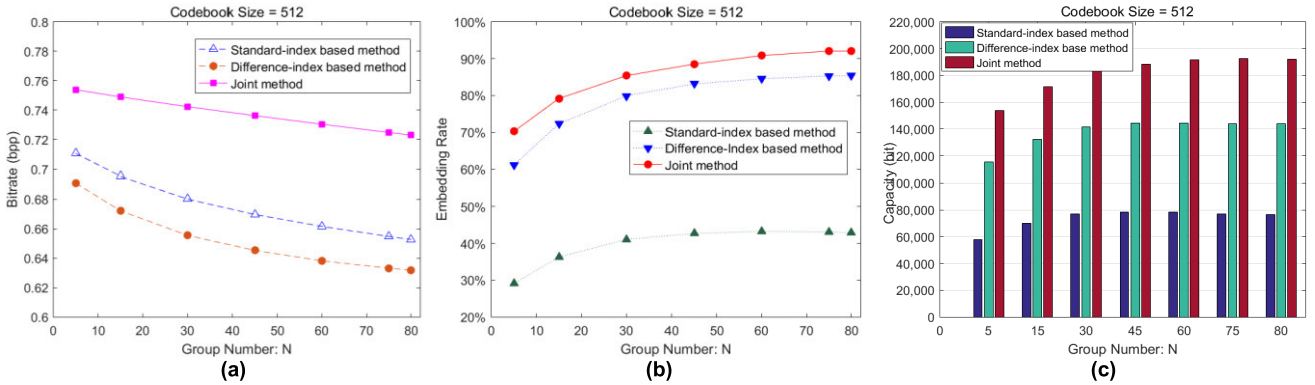


FIGURE 24. Performance of the index based method, the proposed difference-index based method, and the proposed joint method, on average, for 100 test images (codebook size = 512). (a) Bitrate. (b) Embedding Rate. (c) embedding capacity.

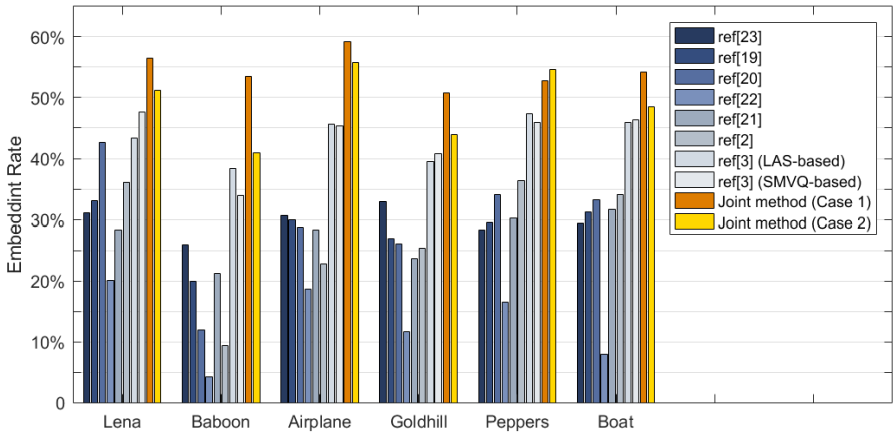


FIGURE 25. Comparison on embedding rate of the proposed joint method and the other methods.

that, no matter that codebook size is 512 or 256, both embedding capacity and embedding rate of the proposed joint method are the best and the bitrate can only be slightly increased compared to the other two methods. When CZ = 512, the embedding capacity is a little higher than that of CZ = 256. However, because the bitrate of the former is larger, the embedding rate is slightly lower than that of the latter. Also, as shown in Fig. 23(c), the peak of the embedding capacity is 187070 bits when the group number N is 40, and the peak of the embedding capacity in Fig. 24(c) reaches 192570 bits when the group number N is 75. This indicates that the larger the codebook size is, the more the peak of the embedding capacity will appear in the case of a larger group number N. For the problem of how the optimal number of groups (i.e., parameter N) and the group length for each to-be-mapped index are determined for a given hiding capacity can be resolved by traversing all possible N like [48], or by some optimization techniques that find the optimal solution.

To further demonstrate the effectiveness and superiority of our proposed joint method, we also compared the embedding capacity, bitrate and embedding rate between our joint method and other existing methods [2], [3], [19]–[23]. It should be noted that, in these experiments, two strategies of the proposed joint method based on Case 1 and Case 2 are performed, respectively. The difference between those two cases is that only the indices with zero frequency were employed as candidate indices in **Stage 1** in the former strategy and the others are the same. The comparisons of the embedding capacity and bitrate are shown in TABLE 13, and the comparisons of the embedding rate is depicted in Fig. 25. Here, codebook size is set to 256.

From TABLE 13, it is easy to find that the proposed joint methods, including strategies of Case 1 and Case 2, outperform all the previous methods [2], [3], [19]–[23]. Obviously, most of those schemes [2], [19]–[23] have a little weakness in embedding capacity (no more than 50000 bits), which is significantly lower than that of scheme [3] and our method. In scheme [3], an efficient index mapping mechanism based on the concepts of Huffman coding was designed to obtain a higher embedding capacity. This benefits from their idea in fully exploring redundancy room from the peaks of the distribution of the indices in the standard-index table. On the contrary, we further observed that the more redundancy can

TABLE 13. Comparison on capacity and bitrate of the proposed joint method and the other methods.

Image	Lena		Baboon		Airplane		Goldhill		Peppers		Boat	
Scheme	Capacity	Bitrate	Capacity	Bitrate	Capacity	Bitrate	Capacity	Bitrate	Capacity	Bitrate	Capacity	Bitrate
Rahmani *et al.* 's (LAS-based) [3]	78888	0.635	75645	0.751	74488	0.655	71406	0.689	80277	0.667	82556	0.690
Joint method (Case 1)	**93290**	**0.630**	**93583**	**0.667**	**101077**	**0.652**	**89223**	**0.670**	**90533**	**0.654**	**94904**	**0.667**
Rahmani *et al.* 's (SMVQ-based) [3]	66107	0.549	55661	0.626	70453	0.564	58071	0.544	66867	0.551	68236	0.574
Joint method (Case 1)	**73524**	**0.548**	**63463**	**0.598**	**81544**	**0.557**	**61796**	**0.537**	**78360**	**0.547**	**70685**	**0.556**
Wang *et al.* 's [23]	49149	0.661	49149	0.723	49149	0.604	49149	0.569	49149	0.637	49149	0.618
Chang *et al.* 's [19]	41684	0.539	34318	0.656	47473	0.548	37377	0.529	43918	0.537	43920	0.559
Lee *et al.* 's [20]	46362	0.519	16991	0.541	57927	0.518	35763	0.523	45477	0.522	39252	0.521
Qin *et al.* 's [22]	21632	0.500	5481	0.500	26321	0.500	15269	0.500	10421	0.500	24363	0.500
Lee *et al.* 's [21]	38491	0.485	29980	0.539	32171	0.432	31851	0.513	39731	0.479	36753	0.494
Lin *et al.* 's [2]	47618	0.500	12327	0.500	47326	0.500	33207	0.500	44758	0.500	29821	0.500

be exploited in the difference-index table rather than in its corresponding standard-index table. Moreover, the joint method is designed to achieve a better embedding performance. As expected, the Case 1-based joint method is not only far greater than other methods in embedding capacity, but also can maintain a satisfactory bitrate (i.e., lower than the LAS-based method in [3]). Moreover, the Case 1-based joint method also obtains the better embedding capacity than the SMVQ-based method in [3] while maintaining a relative lower bitrate.

Fig. 25 plots the comparisons of performance in embedding rate of those schemes [2], [3], [19]–[23] and our joint method. It can be seen that the embedding rates of the proposed joint method based on strategies Case 1 and Case 2 outperforms the other methods [2], [3], [19]–[23], especially strategies Case 1-based method achieves the highest embedding rate because of its great increment of embedding capacity. In other words, it is concluded that the proposed joint method has a superior performance than other methods, in respects to embedding capacity, bit rate and embedding capacity.

V. CONCLUSIONS

Most of previous reversible data hiding schemes are restricted to the generation of redundancy vacated from the conventional standard-index table directly, while neglecting the intrinsic relationship in differences of those indices. In this paper, we present two novel RDH methods for VQ- compressed images. The first one is difference-index based method, which aims on taking advantage of the sharp and compact distribution of difference indices to achieve high embedding performance. Based on the frequency of those indices, a mapping table is constructed and then sued to serve for hiding secret bits. The second one is the joint method by combining the standard-index based method and difference-index based method. This joint strategy draws on standard- and difference-index based methods' strengths, resulting in the improvement in embedding capacity and embedding rate while keeping the bitrate at a satisfactory level. Experimental results and analysis demonstrate the superior performance of the proposed scheme.

REFERENCES

[1] R. G. V. Schyndel, A. Z. Tirkel, and C. F. Osborne, "A digital watermark," in *Proc. IEEE Int. Conf. Image Process.*, Austin, TX, USA, vol. 2, Dec. 1994, pp. 86–90.

[2] C.-C. Lin, X.-L. Liu, and S.-M. Yuan, "Reversible data hiding for VQ-compressed images based on search-order coding and state-codebook mapping," *Inf. Sci.*, vol. 293, pp. 314–326, Feb. 2015.

[3] P. Rahmani and G. Dastghaibyfard, "An efficient histogram-based index mapping mechanism for reversible data hiding in VQ-compressed images," *Inf. Sci.*, vol. 435, pp. 224–239, Apr. 2018.

[4] G.-D. Su, C.-C. Chang, and C.-C. Lin, "A high capacity reversible data hiding in encrypted AMBTC-compressed images," *IEEE Access*, to be published.

[5] G.-D. Su, Y. Liu, and C.-C. Chang, "A square lattice oriented reversible information hiding scheme with reversibility and adaptivity for dual images," *J. Vis. Commun. Image Represent.*, vol. 64, Oct. 2019, Art. no. 102618.

[6] P. Singh and B. Raman, "Reversible data hiding based on Shamir's secret sharing for color images over cloud," *Inf. Sci.*, vol. 422, pp. 77–97, Jan. 2018.

[7] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Trans. Image Process.*, vol. 13, no. 8, pp. 1147–1156, Aug. 2004.

[8] J. M. Barton, "Method and apparatus for embedding authentication information within digital data," U.S. Patent 5 646 997, Jul. 8, 1997.

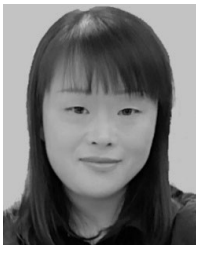
[9] K.-L. Chung, W.-J. Yang, and W.-N. Yang, "Reversible data hiding for depth maps using the depth no-synthesis-error model," *Inf. Sci.*, vol. 269, pp. 159–175, Jun. 2014.

[10] W. Hong, X. Zhou, D.-C. Lou, T. S. Chen, and Y. Li, "Joint image coding and lossless data hiding in VQ indices using adaptive coding techniques," *Inf. Sci.*, vols. 463–464, pp. 245–260, Oct. 2018.

[11] S. Weng and J.-S. Pan, "Integer transform based reversible watermarking incorporating block selection," *J. Vis. Commun. Image Represent.*, vol. 35, pp. 25–35, Feb. 2016.

[12] S. W. Weng, S. C. Chu, N. Cai, and R. X. Zhan, "Invariability of mean value based reversible watermarking," *J. Inf. Hiding Multimedia Signal. Process.*, vol. 4, no. 1, pp. 90–98, Apr. 2013.

[13] R. M. Gray, "Vector quantization," *IEEE Acoust. Speech Signal Process. Mag.*, vol. 1, no. 2, pp. 4–29, Apr. 1984.
[14] Y. T. Chang, C.-T. Huang, C.-L. Huang, and S.-J. Wang, "Data hiding of high compression ratio in VQ indices with neighboring correlations," *Multimedia Tools Appl.*, vol. 74, no. 5, pp. 1645–1666, Mar. 2015.
[15] J.-D. Lee, Y.-H. Chiou, and J.-M. Guo, "Reversible data hiding based on histogram modification of SMVQ indices," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 4, pp. 638–648, Dec. 2010.
[16] S.-C. Shie and J.-H. Jiang, "Reversible and high-payload image steganographic scheme based on side-match vector quantization," *Signal Process.*, vol. 92, no. 9, pp. 2332–2338, Sep. 2012.
[17] L. Wang, Z. Pan, X. Ma, and S. Hu, "A novel high-performance reversible data hiding scheme using SMVQ and improved locally adaptive coding method," *J. Vis. Commun. Image Represent.*, vol. 25, no. 2, pp. 454–465, Feb. 2014.
[18] W.-J. Wang, C.-T. Huang, C.-M. Liu, P.-C. Su, and S.-J. Wang, "Data embedding for vector quantization image processing on the basis of adjoining state-codebook mapping," *Inf. Sci.*, vol. 246, pp. 69–82, Oct. 2013.
[19] C.-C. Chang, C.-Y. Lin, and Y.-P. Hsieh, "Data hiding for vector quantization images using mixed-base notation and dissimilar patterns without loss of fidelity," *Inf. Sci.*, vol. 201, pp. 70–79, Oct. 2012.
[20] J.-D. Lee, Y.-H. Chiou, and J.-M. Guo, "Lossless data hiding for VQ indices based on neighboring correlation," *Inf. Sci.*, vol. 221, pp. 419–438, Feb. 2013.
[21] J.-D. Lee, Y.-H. Chiou, and J.-M. Guo, "Information hiding based on block match coding for vector quantization-compressed images," *IEEE Syst. J.*, vol. 8, no. 3, pp. 737–748, Sep. 2014.
[22] C. Qin, C.-C. Chang, and Y.-C. Chen, "Efficient reversible data hiding for VQ-compressed images based on index mapping mechanism," *Signal Process.*, vol. 93, no. 9, pp. 2687–2695, Sep. 2013.
[23] Z.-H. Wang, C.-C. Chang, K.-N. Chen, and M.-C. Li, "An encoding method for both image compression and data lossless information hiding," *J. Syst. Softw.*, vol. 83, no. 11, pp. 2073–2082, Nov. 2010.
[24] J.-X. Wang and Z.-M. Lu, "A path optional lossless data hiding scheme based on VQ joint neighboring coding," *Inf. Sci.*, vol. 179, no. 19, pp. 3332–3348, Sep. 2009.
[25] C.-C. Chang, T. D. Kieu, and Y.-C. Chou, "Reversible information hiding for VQ indices based on locally adaptive coding," *J. Vis. Commun. Image Represent.*, vol. 20, no. 1, pp. 57–64, Jan. 2009.
[26] C.-C. Chang, T. S. Nguyen, and C.-C. Lin, "A reversible data hiding scheme for VQ indices using locally adaptive coding," *J. Vis. Commun. Image Represent.*, vol. 22, no. 7, pp. 664–672, Oct. 2011.
[27] J. L. Bentley, D. D. Sleator, R. E. Tarjan, and V. K. Wei, "A locally adaptive data compression scheme," *Commun. ACM*, vol. 29, no. 4, pp. 320–330, Mar. 1986.
[28] C.-C. Chang, T. S. Nguyen, and C.-C. Lin, "A reversible compression code hiding using SOC and SMVQ indices," *Inf. Sci.*, vol. 300, pp. 85–99, Apr. 2015.
[29] C.-C. Chang and C.-Y. Lin, "Reversible steganographic method using SMVQ approach based on declustering," *Inf. Sci.*, vol. 177, no. 8, pp. 1796–1805, Apr. 2007.
[30] C.-C. Chang, Y.-P. Hsieh, and C.-Y. Lin, "Lossless data embedding with high embedding capacity based on declustering for VQ-compressed codes," *IEEE Trans. Inf. Forensics Security*, vol. 2, no. 3, pp. 341–349, Sep. 2007.
[31] C.-C. Chang, T. S. Nguyen, and C.-C. Lin, "Reversible data embedding for indices based on histogram analysis," *J. Vis. Commun. Image Represent.*, vol. 25, no. 7, pp. 1704–1716, Oct. 2014.
[32] T.-Y. Tu and C.-H. Wang, "Reversible data hiding with high payload based on referred frequency for VQ compressed codes index," *Signal Process.*, vol. 108, pp. 278–287, Mar. 2015.
[33] C.-H. Yang and Y.-C. Lin, "Reversible data hiding of a VQ index table based on referred counts," *J. Vis. Commun. Image Represent.*, vol. 20, no. 6, pp. 399–407, Aug. 2009.
[34] P. Rahmani and G. Dastghaibyfard, "Reversible data hiding for VQ-compressed images based on an index replacement technique," *Int. J. Comput. Electr. Eng.*, vol. 4, no. 2, pp. 359–362, Jun. 2012.
[35] C. C. Chang, C. H. Sung, and T. S. Chen, "A locally adaptive scheme for image index compression," in *Proc. Conf. Comput. Vis., Graph., Image Process.*, Taichung, Taiwan, 1997, pp. 93–99.
[36] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Commun.*, vol. 28, no. 1, pp. 84–95, Jan. 1980.
[37] C. H. Lee and L. H. Chen, "Fast closest codeword search algorithm for vector quantization," *IEE Proc. Image Sig. Process.*, vol. 141, no. 3, pp. 143–148, 1994.
[38] Z. N. Li and M. S. Drew, *Fundamentals of Multimedia*. Cham, Switzerland: Springer, 2014, pp. 227–233.
[39] N. M. Nasrabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Trans. Commun.*, vol. 36, no. 8, pp. 957–971, Aug. 1988.
[40] C.-C. Chang, W.-C. Wu, and Y.-C. Hu, "Lossless recovery of a VQ index table with embedded secret data," *J. Vis. Commun. Image Represent.*, vol. 18, no. 3, pp. 207–216, Jun. 2007.
[41] P. Rahmani and G. Dastghaibyfard, "A low distortion reversible data hiding scheme for search order coding of VQ indices," *Multimedia Tools Appl.*, vol. 74, no. 23, pp. 10713–10734, Dec. 2015.
[42] P. Rahmani, M. S. Norouzzadeh, and G. Dastghaibyfard, "A novel legitimacy preserving data hiding scheme based on LAS compressed code of VQ index tables," *Multidimensional Syst. Signal Process.*, vol. 27, no. 2, pp. 433–452, Apr. 2016.
[43] C.-C. Chang, T. D. Kieu, and W.-C. Wu, "A lossless data embedding technique by joint neighboring coding," *Pattern Recognit.*, vol. 42, no. 7, pp. 1597–1603, Jul. 2009.
[44] C. Qin and Y.-C. Hu, "Reversible data hiding in VQ index table with lossless coding and adaptive switching mechanism," *Signal Process.*, vol. 129, pp. 48–55, Dec. 2016.
[45] A. J. Pinho and A. J. R. Neves, "A survey on palette reordering methods for improving the compression of color-indexed images," *IEEE Trans. Image Process.*, vol. 13, no. 11, pp. 1411–1418, Nov. 2004.
[46] W. Zeng, J. Li, and S. Lei, "An efficient color re-indexing scheme for palette-based compression," in *Proc. IEEE Int. Conf. Image Process.*, vol. 3, Feb. 2000, pp. 476–479.
[47] C.-C. Chang, W.-L. Tai, and C.-C. Lin, "A reversible data hiding scheme based on side match vector quantization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 10, pp. 1301–1308, Oct. 2006.
[48] Y. Qiu, H. He, Z. Qian, S. Li, and X. Zhang, "Lossless data hiding in JPEG bitstream using alternative embedding," *J. Vis. Commun. Image Represent.*, vol. 52, pp. 86–91, Apr. 2018.
[49] H. Ge, Y. Chen, Z. Qian, and J. Wang, "A high capacity multilevel approach for reversible data hiding in encrypted images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 8, pp. 2285–2295, Aug. 2019.
[50] Z.-L. Liu and C.-M. Pun, "Reversible data-hiding in encrypted images by redundant space transfer," *Inf. Sci.*, vols. 433–434, pp. 188–203, Apr. 2018.



YUEHUI LI received the M.S. and Ph.D. degrees from the University of Electronic Science and Technology of China, in 2004 and 2011, respectively. She is currently an Associate Professor with the School of Computer and Software Engineering, Xihua University, Chengdu, China. She has published more than 20 articles in refereed professional journals and international conferences. Her current research interests include computer cryptography, information security, and numerical computation. She is a member of the ACM.

CHIN-CHEN CHANG (Fellow, IEEE) received the B.Sc. degree in applied mathematics and the M.Sc. degree in computer and decision sciences from National Tsing Hua University and the Ph.D. degree in computer engineering from National Chiao Tung University. He has served for National Chung Cheng University, from 1989 to 2005. Since February 2005, he holds a chair professor position at the Department of Information Engineering and Computer Science, Feng Chia University. He is also a Guest Professor with Hangzhou Dianzi University. Prior to joining Feng Chia University, he was an Associate Professor with Chiao Tung University, a Professor with National Chung Hsing University, and a Chair Professor with National Chung Cheng University. He had also been a Visiting Researcher and a Visiting Scientist with Tokyo University and Kyoto University, Japan. During his service in Chung Cheng, he has served as the Chairman of the Institute of Computer Science and Information Engineering, the Dean of College of Engineering, a Provost and then Acting President of Chung Cheng University, and the Director of Advisory Office in Ministry of Education, Taiwan. On numerous occasions, he was invited to serve as a visiting professor, a chair professor, an honorary professor, the honorary director, the honorary chairman, a distinguished alumnus, a distinguished researcher, a research fellow by universities and research institutes. His current research interests include database design, computer cryptography, image compression, and data structures. He is currently a Fellow of the IEE, U.K. He has won many research awards and honorary positions by and in prestigious organizations both nationally and internationally. Since his early years of career development, he consecutively won Outstanding Talent in Information Sciences of China, the AceR Dragon Award of the Ten Most Outstanding Talents, the Outstanding Scholar Award of China, the Outstanding Engineering Professor Award of China, Distinguished Research Awards of the National Science Council of China, and Top 15 Scholars in Systems and Software Engineering of the *Journal of Systems and Software*, and so on.

MINGXING HE received the M.S. degree from Chongqing University and the Ph.D. degree from Southwest Jiaotong University, in 1990 and 2003, respectively. He is currently a Full Professor with the School of Computer and Software Engineering, Xihua University, Chengdu, China. He has coauthored five books and has published more than 100 articles in refereed professional journals and international conferences. His current research interests include cryptography and information security. He is a member of the ACM and a Senior Member of the CACR. He received the DAAD Scholarship Reward of Germany, in 2002, the Excellent Ph.D. Dissertation Award from Southwest Jiaotong University, in 2003, and the Grant of National Science Foundation of China (NSFC), in 2004, 2007, and 2015.