High Capacity Lossless Data Hiding in JPEG Bitstream Based on General VLC Mapping

Yang Du, Zhaoxia Yin, *Member, IEEE*, and Xinpeng Zhang, *Member, IEEE*

***Abstract*—As a branch of reversible data hiding (RDH), lossless data hiding (LDH) technique is important especially. Because LDH can not only reconstruct the cover image losslessly but also keep the visual quality of the marked image no degradation. This paper proposes a new LDH scheme for JPEG images by general variable length code (VLC) mapping. In this scheme, the run size values are first reordered. Then a feasible solution space is generated. A simulated embedding model is established to find the optimal solution from the feasible solution space. The optimal mapping relationship is constructed according to the optimal solution. According to the optimal GVM relationship, the Huffman table in the file header is modified and then the additional data can be embedded by replacing the used VLC with the VLCs in the same mapping set. Experimental results demonstrate that most of the JPEG images using the proposed scheme cause less file size increments than previous RDH schemes while keeping the marked JPEG image with no distortion. Furthermore, the proposed scheme can obtain high embedding capacity.**

***Index Terms*—Reversible data hiding, lossless data hiding, JPEG, VLC, histogram modification.**

I. Introduction

REVERSIBLE data hiding (RDH) can embed additional data into cover media imperceptibly while the cover media can be reconstructed losslessly during the data extraction process. The reversibility feature is desirable for many applications, such as medical or military image processing and law forensics. Recently, RDH is extended to many other fields innovatively [1], such as reversible steganography [2] [3], reversible adversarial example [4] and reversible data hiding in encrypted domain [5] [6].

Many RDH schemes for uncompressed images have been proposed and can be classified into three categories based on different technologies: lossless compression [7] [8], difference expansion (DE) [9] [10], and histogram shifting (HS) [11]–[14]. However, JPEG is the most popular image format which is applied to digital cameras and the Internet widely. Therefore, developing the RDH scheme for JPEG images is more important.

The existing RDH schemes for JPEG images can be classified into two categories according to the modifying domain. The first category of schemes [15]–[24] is based on modifying the Discrete Cosine Transform (DCT) coefficients domain. We refer to this category of schemes as DCT-based schemes. The second category of schemes [25]–[27] is based on modifying the variable length codes (VLC) domain. We refer to this category of schemes as VLC-based schemes.

Some early works of DCT-based schemes [15] [16] modified the quantization table in the file header to embed data. In [15], Fridrich $et\ al.$ firstly proposed an RDH scheme that modifies the quantization table factors to embed one bit per DCT coefficient. The basic idea is to divide some quantized table factors by 2 and multiply the quantized DCT coefficients of their corresponding positions by 2 to embed the additional data into the least significant bits (LSB) of DCT coefficients. Then Wang $et\ al.$ [16] designed a new embedding order to embed information in a position preferentially where distortion is small. In [17], Fridrich $et\ al.$ proposed a scheme based on lossless compression of quantized DCT coefficients. This scheme creates a vacated room by compressing the generalized LSBs of non-zero alternating current (AC) coefficients. However, the LSB distribution of the non-zero AC is still balanced relatively. Therefore, the embedding capacity is rather limited.

Statistically, the quantized AC coefficient distribution is a zero-centered Laplacian-like distribution, so the literature [18] [19] migrated the HS technique from uncompressed images to JPEG images. [20] constructed a ternary optimal RDH code based on AC coefficients with values of 0, 1, and -1. But changing the AC coefficients with a value of 0 leads to file size increments inevitably. To further improve the performance of the above HS-based methods for JPEG images, Huang $et\ al.$ [21] proposed a novel HS-based scheme. It keeps the zero AC coefficient unchanged and sets the AC coefficients with values of 1 and -1 as the peak points of the histogram. Since the variation of all non-zero AC coefficients is at most 1, the visual quality is guaranteed. Besides, to further reduce the distortion, Huang $et\ al.$ designed a block selection strategy based on the number of zero coefficients in each 8×8 block. However, the number of zero coefficients in a block does not imply the degree of distortion of the block. In [22], Qian $et\ al.$ analyzed the features of JPEG and then proposed an ordered embedding strategy in several rounds to reduce the file size increments. In [23] [24], to optimize the block selection strategy, the influence of the quantization step in DCT coefficients are considered. These schemes can keep good visual quality and also achieve a smaller file size increments.

The VLC-based schemes embed additional data based on VLC mapping, which can preserve the visual quality of the marked image unchanged. Therefore, these schemes are also

This research work is partly supported by National Natural Science Foundation of China (61872003, U1636206,61860206004).

Yang Du and Zhaoxia Yin are with Key Lab of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230601, China, e-mail: yinzhaoxia@ahu.edu.cn.

Xinpeng Zhang was with school of Computer Science, Fudan University, China, e-mail: zhangxinpeng@fudan.edu.cn.

Corresponding author: Zhaoxia Yin.

referred to as lossless data hiding (LDH) schemes. In [25], Qian and Zhang proposed a scheme to embed data into the JPEG bitstream by replacing the used VLCs with unused VLCs. The visual quality and file size are both preserved. Hu $et\ al.$ [26] and Qiu $et\ al.$ [27] explored the redundancy of VLCs in JPEG bitstream, and further improved the embedding capacity by VLC frequency ordering and mapping optimizing. However, the embedding capacity is related to the frequency of used VLCs, which is rather limited. Unfortunately, at present, the developments of VLC-based RDH schemes are far from enough than DCT-based schemes.

For the most RDH schemes in JPEG images, the basic evaluation metrics are embedding capacity, visual quality, and file size preservation. Generally, there exists a trade-off between visual quality and embedding capacity. Similarly, there also exists a trade-off between file size preservation and embedding capacity. For the DCT-based RDH schemes, the optimization for all the three evaluation metrics is required. However, for the DCT-based RDH schemes, file size preservation and visual quality are difficult to be the optimal simultaneously. While for the VLC-based schemes, only the optimization for embedding capacity and file size preservation is required. The comparison of optimization objectives for the two kinds of schemes is depicted in Fig. 1.

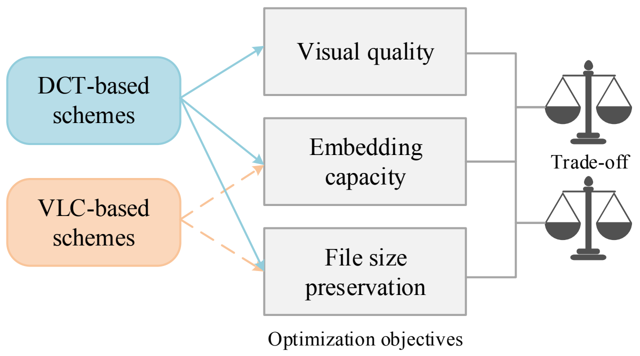


Fig. 1. The comparison of optimization objectives between DCT-based scheme and VLC-based scheme.

Although the VLC-based schemes maintain the file size and visual quality unchanged, the embedding capacity is low indeed. Compared to this case, the high embedding capacity with slight file size increments is more needed for most applications. To obtain the high embedding capacity, a new LDH scheme by adopting the general VLC mapping (GVM) strategy is proposed in this paper. The core of GVM strategy is to allow the lengths of used VLC and unused VLCs in one mapping set to be unequal, which is different from previous VLC-based schemes. Since the lengths of VLCs in one mapping set are unequal, the file size increments are inevitable during data embedding. The challenge is how to construct the optimal mapping relationship, which causes minimal file size increments. In this scheme, the run size values are first reordered in descending order. Then, a feasible solution space is generated according to the given payload and the statistics of VLCs. A simulated embedding model is established to obtain the optimal solution by exhaustive searching. According to the optimal solution, the optimal GVM relationship is constructed and the file header is modified. Finally, the additional data is embedded into the entropy-coded data. Experimental results demonstrate that using the proposed scheme causes smaller file size increments than state-of-the-art DCT-based schemes. Meanwhile, the image content is no distortion and the embedding capacity is higher.

The remainder of this paper is organized as follows. In Section II, the JPEG bitstream structure and a basic embedding instance of VLC-based schemes are briefly introduced. Then the proposed GVM-based LDH scheme is proposed in Section III. The experimental results and analysis are presented in Section IV to show the performance of the proposed scheme. Finally, this paper is concluded in Section V.

II. Preliminaries

In this section, the JPEG bitstream structure is first introduced briefly. Then, the basic embedding model of VLC-based schemes is described.

A. JPEG Bitstream Structure

The JPEG compression process consists of three phases: DCT, quantization and entropy encoding. In the entropy encoding phase, AC coefficients are pre-compressed by using the run-length encoding (RLE). The AC coefficients encoded by RLE are represented as intermediate symbols of the form of $(run/size, value)$. run represents the number of successive zero AC coefficients before the current non-zero AC coefficient. The non-zero AC coefficient is encoded by Variable-Length Integer (VLI) coding. $size$ represents the length of the VLI code. $value$ represents the value of the current nonzero AC coefficient. Each $run/size$, which called Run/Size Value (RSV), is encoded with one VLC from the Huffman table for further compression. $value$ is encoded with VLI coding and the generated codes are called appended bits.

After the JPEG compression process, the entropy-coded data and other information are stored in the form of bitstream. The JPEG bitstream consists of the file header and entropy-coded data. JPEG file header includes some marker segments, such as the define Huffman table (DHT) marker segment. In the DHT segment, the Huffman table records all the RSVs and the lengths of corresponding VLCs. To decode the entropy-coded data with no error, each RSV is corresponded to a unique VLC. Fig. 2 shows the structure of the DHT segment, which contains the DHT header and the information to construct the Huffman table. The DHT header records the length of the segment and other information. L_i is the number of VLCs whose length is i. All the VLCs can be generated according to the value from L_1 to L_{16}. $V_{i,j}$ is the RSV corresponding to the j-th VLC with the length of i. The entropy-coded data mainly consists of VLCs and appended bits. All the VLCs in the entropy-coded data can be found in the Huffman table.

B. Basic embedding instance of VLC-based schemes

In [25], Qian and Zhang proposed a file size preserving LDH scheme by VLC mapping. For the used and unused VLCs, Qian and Zhang proposed a VLC mapping method,

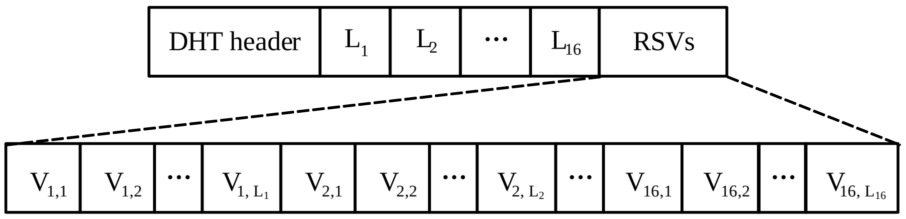


Fig. 2. The structure of the DHT segment.

which maps the unused VLCs to the used VLC. To preserve the file size unchanged, the used and the mapped unused VLCs are the same lengths. While the visual quality is also unchanged by modifying the DHT segment in the file header. Fig. 3 illustrates an basic embedding instance using one mapping set, which the mapping set is $\{VLC_1 \leftrightarrow \{VLC_2, VLC_3, VLC_4\}\}$. VLC_1 is a used VLC and the other VLCs are unused VLCs. VLC_1, VLC_2, VLC_3 and VLC_4 represent the binary data ”00”, ”01”, ”10” and ”11” respectively. As Fig. 3 shows, When data ”11” is to be embedded, VLC_1 is replaced with VLC_4. To preserve the visual quality, the RSVs corresponding to the VLCs in this mapping set are all modified to the RSV corresponding to VLC_1.

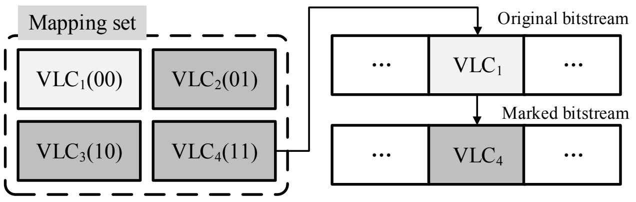


Fig. 3. The instance of embedding data ”11” by replacing the VLC.

III. Proposed Scheme

In this section, the proposed scheme is described in detail. The framework of the proposed scheme is illustrated in Fig. 4. In the preprocessing phase, the VLCs are first parsed and the corresponding RSVs are reordered. The construction phase consists of three parts, the generation of feasible solution space, the establishment of simulated embedding model and the construction of the optimal GVM relationship. According to the optimal GVM relationship, the file header is modified and the additional data is embedded in the JPEG bitstream.

A. Preprocessing

1) VLC Parsing: In order to obtain the statistical information of the RSVs, all the frequencies of VLCs are recorded by parsing the entropy-coded data firstly. Then the Huffman table is generated by parsing the DHT segment. Each RSV in the DHT segment is corresponding to one unique VLC, so the frequencies of VLCs are equivalent to the frequencies of RSVs. We denote the frequency of RSVs and VLCs as F, such that $F = \{f_1, f_2, \cdots, f_N\}$, where f_i is the frequency of the i-th RSV in the DHT segment and N is the number of RSVs.

2) RSV Reordering: The purpose of reordering the RSVs consists of two points. First, for most JPEG images, the default Huffman table is used to encode the DCT coefficients during the entropy-coding phase, which is Huffman coding. In Huffman coding phase, the RSVs are symbols, and the VLCs are the corresponding codes to represent these symbols. The mapping relationship between RSVs and VLCs constructed by the default Huffman table is fixed for all the JPEG images using the default Huffman table. So for a JPEG image using the default Huffman table, a long code might be assigned for a high-frequency RSV, which will cause coding redundancy. We reorder the RSVs according to their frequency to ensure the short codes can be assigned for high frequency RSVs. That is, a more suitable mapping relationship can be constructed by RSV reordering, which removes the coding redundancy the most. Second, RSV reordering is an initialized-liked process. After reordering the RSVs into the descending order, the performance of an arbitrary mapping relationship can be easily measured to obtain the optimal mapping relationships.

According to the Huffman coding, the short VLCs are assigned to the RSVs with high frequency. Therefore, the RSVs are reordered according to the frequencies in the descending order. The frequencies of the reordered RSVs are represented as $F' = \{f'_1, f'_2, \cdots, f'_N\}$. We use the difference of JPEG file size before and after RSV reordering to define coding redundancy C, which can be calculated by

$$C = \sum_{i=1}^{N} (f_i - f'_i) \cdot l_i, \qquad (1)$$

where l_i denotes the length of the i-th VLC, in the order of original RSVs. Noted that the file size change caused by byte alignment and zero-byte padding in the encoding phase is not included in Equation. (1) since the change is acceptable and cannot be predicted in advance. Fig. 5 shows the histograms of the first 30 RSVs before and after reordering for the image Baboon with quality factor (QF) of 70. X-axis stands for the index of RSVs in the DHT segment. It can be easily seen from Fig. 5 that the reordered RSV histogram is sharper and suitable for modification.

B. Selection of Embedding Manner

Before presenting our construction of the GVM relationship, we first describe two kinds of embedding manners. Then,

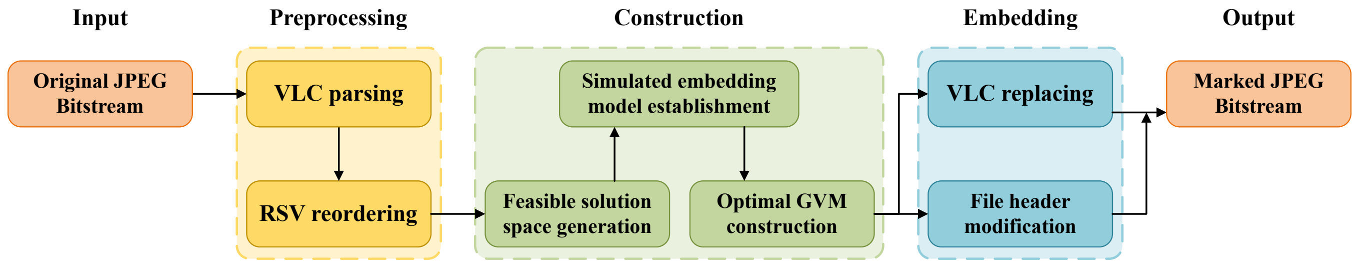


Fig. 4. The framework of the proposed scheme.

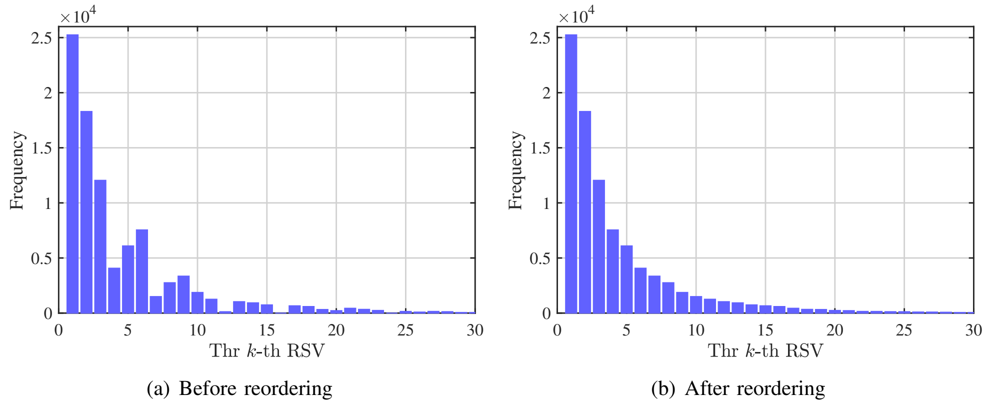


Fig. 5. The histograms of the first 30 RSVs before and after reordering for the image Baboon (QF = 70).

which manner is more suitable to construct the GVM relationship is discussed.

1) Direct Mapping (DM) Embedding Manner: In the DM embedding manner, several unused VLCs and one used VLC are selected to construct the mapping relationship directly. The positions in the file header of the RSVs corresponding to these unused VLCs are unchanged. An example of the DM embedding manner is illustrated in Fig. 6. In Fig. 6, all the frequencies of the RSVs after RSV_4 (0/0) are all zero, which is the VLCs corresponding to RSV_5 (0/4) and RSV_6 (1/1) are unused VLCs. Then they are mapped to RSV_1 (0/1) and RSV_2 (0/2) respectively. The other RSVs are not involved in the construction of the mapping relationship.

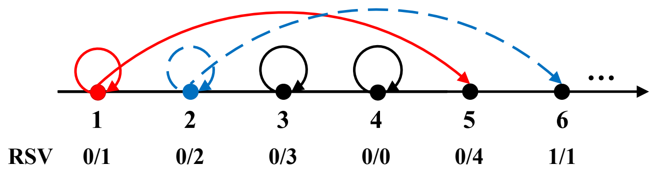


Fig. 6. Example of the direct mapping embedding manner.

2) Histogram Shifting (HS) Embedding Manner: Based on the histogram of reordered RSVs, the VLC mapping model can be converted into the HS model. For the histogram of RSVs, one RSV corresponding to the used VLC can be seen as one nonzero bin while one RSV corresponded to the unused VLC can be seen as one zero bin. In the embedding process using this manner, all the bins in the right of the selected nonzero bin are shifted one or more units. After shifting, the several nearest bins in the right of the selected nonzero bin are changed to zero bins. Then the additional data can be embedded by replacing the original VLCs with one of the VLCs in the same mapping set. The lengths of VLCs in a mapping set are close or even equal, which causes small increments in the file size. Multiple nonzero bins can be selected to obtain higher embedding capacity and better performance of the file size preservation. For one selected nonzero bin, the number of assigned zero bins can also be multiple. Fig. 7 illustrates the HS embedding manner. In Fig. 7, RSV_1 (0/1) is expanded to embed data, RSV_2 (0/2) is firstly shifted to the right by 1 unit and then expanded to embed data. The other RSVs are shifted to the right by 2 units to create new zero bins.

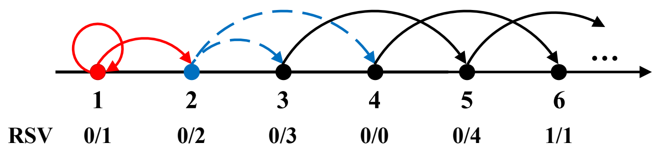


Fig. 7. Example of histogram shifting embedding manner.

3) Analysis of Two Embedding Manners: In all the previous VLC-based schemes [25]–[27], the DM embedding manner is utilized. Since the used VLC and unused VLCs in the same mapping set are with the same length, the length of the VLC in the JPEG bitstream is unchanged before and after replacement. However, if we use this manner to construct the GVM relationship, the file size of marked JPEG bitstream will be increased significantly than the original JPEG bitstream. After reordering the RSVs in descending order, these zero-frequency RSVs are at the end of the RSV sequence. In general, the lengths of unused VLCs are close or equal to 16 bits. While the lengths of used VLCs corresponding to the high-frequency RSVs are far shorter than 16 bits. The increased file size is unacceptable after replacement. Therefore, we select the HS embedding manner to construct the GVM relationship.

C. Construction

For the HS embedding manner, there are several parameters to represent how to modify the histogram. The parameters consist of the selected nonzero bins and the number of zero bins assigned for each selected nonzero bin. We want to assign more zero bins to the high-frequency RSVs, that is, assign the high-frequency used VLC to more unused VLCs. For example, if the number of VLCs in one mapping relationship is 8 (one used VLC and 7 unused VLCs), the VLC in this mapping relationship can represent 3 ($log_2(8)$) bits data. Clearly, the more assigned zero bins (unused VLCs), the more data the VLC in one mapping relationship can represent. Since the frequency of the reordered RSVs is in descending order, we only select adjacent nonzero bins for one parameter combination. Then we just need to determine the position of first selected nonzero bin p^s, the number of selected nonzero bins m and the number of zero bins assigned for each selected nonzero bin $\{a_1, a_2, \cdots, a_m\}$. These parameters can be expressed in one combination, which is formatted as $(p^s, m, \{a_1, a_2, \cdots, a_m\})$. Different parameter combinations correspond to different mapping relationships, which are also equivalent to different modifications for a histogram.

An appropriate parameter combination is called a feasible solution that the capacity obtained by using this combination can be greater than or equal to a given payload, which is what we need. For the histogram of RSVs, obviously there are many kinds of feasible solutions. They together constitute the entire feasible solution space. The key issue of our proposed scheme is how to obtain the optimal solution to construct the optimal GVM relationship from the feasible solution space. We want to compare the performance of each feasible solution to find the optimal solution. To reduce the timing consumed in comparison, we set some constraints to reduce the scale of solution space. Then we establish a simulated embedding model to compare the simulated file size increments for each feasible solution. Finally, we can construct the optimal GVM relationship according to the optimal solution.

1) Generation of feasible solution space: The scale of the solution space depends on the the ranges of p^s, m and $\{a_1, a_2, \cdots, a_m\}$. To reduce the timing cost in searching the optimal solution, some insights on reducing the scale of the solution space are proposed in this section.

Firstly, the histogram of the reordered RSVs h can be represented as:

$$h(i) = \{f'_i \mid 1 \leqslant i \leqslant N\}, \tag{2}$$

where $h(i)$ is the frequency of the i-th reordered RSV, which is equal to the f'_i in Equation. (1). In the entire solution space, there is an extreme solution that only selecting one nonzero bin and assigning only one zero bin for it while embedding, which equivalent to the single HS manner. We refer the only selected bin as the extreme bin. In the extreme case, to reduce the file size increments while embedding all the additional data, the frequency of the extreme bin should be greater than or equal to the given payload and better to the closest to the given payload. Denote the given payload by P, the position of the extreme bin by p^e, then the determination of p^e is equivalent to solve the following problem:

$$p^e = \underset{i \in N}{\arg\min}\{h(i) \mid h(i) \geqslant P\}. \tag{3}$$

To reduce the solution space, for each feasible solution, we only select the nonzero bins which are greater than or equal to p^e. That is, we only select the solution that the first selected nonzero bin is in the right of the p^e-th bin from solution space. Surely, the position of the first selected nonzero bin p^s can also be p^e. Thus, the candidate range of p^s can be represented as:

$$p^s = \{i \mid p^e \leqslant i \leqslant N_{nz}\}, \tag{4}$$

where N_{nz} is the total number of nonzero bins, which is also equivalent to the position of the last nonzero bin in the histogram. Fig. 8 describes the determination of p^e in the extreme solution and the corresponding candidate range for the entire solution space. In Fig. 8, the histogram has 6 nonzero bins. The frequency of the third bin is greater than and also the closest to the given payload P, so the value of p^e in this histogram is 3. As the number of nonzero bins in this histogram is 6, the value of N_{nz} is 6. Thus the candidate range of p^s is from 3 to 6.

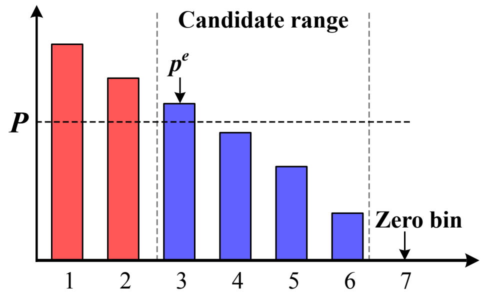


Fig. 8. The selection of the only nonzero bin in the extreme case and the corresponding candidate range.

For a feasible solution, after determining the position of the first selected nonzero bin p^s, the max number of selected nonzero bins m should be determined. We define a user-specified factor U to control the scale of solution space to some extent artificially:

$$m = \min\{(N_{nz} - p^s + 1), U\}. \tag{5}$$

The larger the value of U is, the larger the solution space will have. In general, the value of U is no need to be large. Then the position of the n-th selected nonzero bins in h can be expressed as

$$p_n = \{p^s + n - 1 \mid 1 \leqslant n \leqslant m\}, \tag{6}$$

where p_n is the position of the n-th selected nonzero bin in a solution.

In addition, we propose four constraints to limit the range of the number of zero bins assigned to each selected nonzero bin. For one feasible solution, these constraints are described as following:

In addition, we propose four constraints to limit the range of the number of zero bins assigned to each selected nonzero bin. For one feasible solution, these constraints are described as following:

Constraint 1: The sum of the numbers of assigned zero bins should be less than or equal to the total number of zero bins N_z.

This constraint can be expressed by

$$\sum_{k=1}^{m} a_k \leqslant N_z, \tag{7}$$

where a_k is the number of zero bins assigned for the k-th selected nonzero bin.

Constraint 2: The capacity obtained using this solution must be greater than or equal to the given payload P.

This constraint can be expressed by

$$\sum_{k=1}^{m} \log_2(a_k+1) \cdot h(p^k) \geqslant P, \tag{8}$$

where $h(p^k)$ is the frequency of the p^k-th bin in the histogram h. $\log_2(a_k+1)$ means that each VLC in the k-th mapping set can represent $\log_2(a_k+1)$ bits data.

Constraint 3: The number of zero bins assigned to each selected nonzero bin should be less than or equal to the one assigned to the previous nonzero bin.

This constraint can be expressed by

$$a_{k+1} \leqslant a_k, 1 \leqslant k < m. \tag{9}$$

As the RSVs are reordered in descending order according to the frequency, the solution space can be reduced effectively by setting this constraint.

Constraint 4: The number of zero bins assigned for selected nonzero bins should be equal to one of the $\{0, 1, 3, 7, 15, 31, 63\}$.

This constraint can be expressed by

$$a_k = 2^j - 1, j = 1, 2, \cdots, 6. \tag{10}$$

To embed the binary additional data conveniently, the length of each mapping set should be equal to j-th power of 2. Since the number of RSVs in the default Huffman table is 162, the value of j can be taken from 1 to 6.

2) Establishment of Simulated Embedding Model: After reducing the solution space, the performance of each feasible solution should be evaluated to determine the optimal one. Since the visual quality of the marked JPEG images is no degradation using our proposed scheme, only the capability of file size preservation need to be evaluated. Thus, we want to measure the file size increments for each feasible solution in advance to find the optimal solution, which the file size increments are minimal.

During the shifting process, the RSVs between the selected nonzero bin and zero bins are shifting to the right by multiple units. The frequency of RSVs in the right will increase, which means the frequency of VLC with long code will increase. Thus the file size will increase in the shifting process. In the embedding process, the original VLC can be replaced with the VLC in the same mapping set. For the HS embedding manner, the zero bins assigned to the selected bin are in the right of the selected nonzero bin, which means the lengths of VLCs in the same mapping set are greater than or equal to the original VLC, which will increase the file size. Therefore, we establish a simulated embedding model to simulate the shifting and embedding process for each feasible solution. The file size increments caused by shifting and embedding can be calculated approximately by using the simulated embedding model, which is referred to as simulated file size increments (SFI).

In the first place, we take a simple example to illustrate the shifting process and simulated embedding process, which are shown in Fig. 9 and Fig. 10 respectively. Fig. 9 illustrates a simple example of the shifting process of the feasible solution $(1, 2, \{1, 1\})$. That is, the position of the first selected nonzero bin is 1 and the number of selected nonzero bins is 2. In addition, the numbers of zero bins assigned to all selected nonzero bins are all equal to 1. As Fig. 9 shows, the shifting process in this example consists of two successively sub-processes, since the number of selected nonzero bins is 2. The first and second bins are all selected nonzero bins, which signified by red and yellow respectively. In the first shifting sub-process, all the bins are shifted to the right by one unit except the first bin. Then the second bin now becomes a new zero bin. In the second shifting sub-process, the remaining bins except the first three bins are shifted to the right by one unit again, and the fourth bin also becomes a new zero bin now. The number of shifted units in each sub-process is equal to one means that the number of zero bins assigned to each nonzero bin is equal to 1. Fig. 10 describes a simple example of the simulated embedding process of the feasible solution $(1, 2, \{1, 1\})$. In Fig. 10, the left is the histogram after shifting, bin "2" and "4" are all new zero bins which correspond to bin "1" and bin "3" respectively. Therefore, there are two mapping sets for histogram. The first mapping set consists of VLC_1 and VLC_2 and the second mapping set consists of VLC_3 and VLC_4. Then VLC_1 and VLC_3 represent data "0". VLC_3 and VLC_4 represent data "1". Assume that the probabilities of data "0" and data "1" are the same, the half of the bin "1" is then expanded to the bin "2" to embed data "1" and the half of the bin "3" is expanded to the bin "4" to embed data "1".

For the RSV histogram h of a JPEG image, assume that one feasible solution is the j-th solution in solution space, which is denoted by $(p_j^s, m_j, \{a_{j,1}, a_{j,2}, \cdots, a_{j,m_j}\})$. The position of the n-th selected nonzero bin $p_{j,n}$ can be expressed as

$$p_{j,n} = \{p_j^s + n - 1 \mid 1 \leqslant n \leqslant m_j\}. \tag{11}$$

In fact, the shifting process consists of multiple successively sub-processes. The number of sub-processes is equal to the number of selected nonzero bins. Correspondingly, the position of the n-th selected nonzero bin after shifting $p'_{j,n}$ can be expressed as:

$$p'_{j,n} = \begin{cases} p_{j,n}, & \text{if } n = 1 \\ p_{j,n} + \sum_{k=1}^{n-1} a_{j,k}, & \text{if } 2 \leqslant n \leqslant m_j \end{cases}. \tag{12}$$

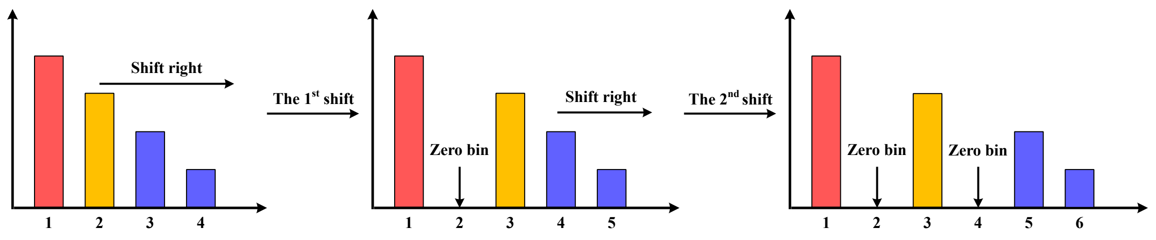


Fig. 9. An example of shifting process, which consists of two shifting sub-processes. The red bin and yellow bin are all selected nonzero bins for the histogram.

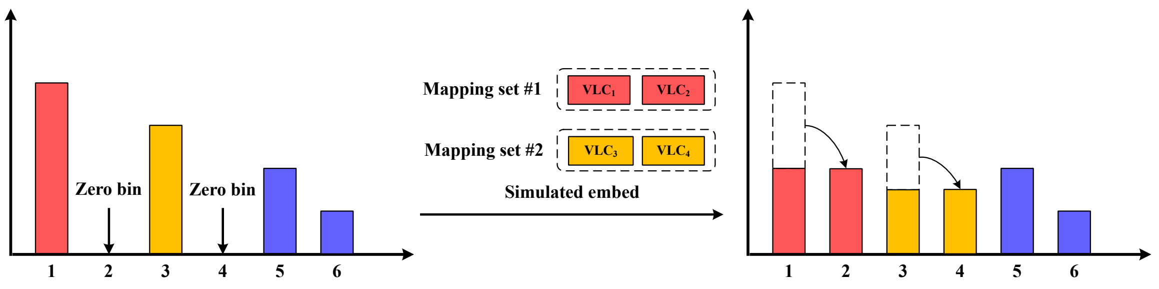


Fig. 10. An example of simulated embedding process. The red bin and yellow bin are all selected nonzero bins for the histogram.

In Equation. (12), the n-th selected nonzero bins (except the first selected nonzero bin) has been shifted to the right by $n-1$ sub-processes during the shifting process, so its position is the sum of numbers of zero bins assigned to the first $n-1$ selected nonzero bins. According to the Equation. (12), the frequency of i-th bin in the RSV histogram after shifting, $h_j^s(i)$, can be represented as:

$$h_j^s(i) = \begin{cases} h(i), & \text{if } 1 \leqslant i < p_s \\ h(p_{j,n}), & \text{if } i = p'_{j,n} \\ h(i - \sum_{k=1}^{m_j} a_{j,k}), & \text{if } p'_{j,m_j} + a_{j,m_j} + 1 \leqslant i \\ & \leqslant N_{nz} + \sum_{k=1}^{m_j} a_{j,k} \\ 0, & \text{others} \end{cases} . \quad (13)$$

As Equation. (13) shows, the frequencies of the bins in the left of the first selected nonzero bin keep unchanged during the shifting process. The frequency of the $p'_{j,n}$-th bin in the histogram after shifting is equal to the frequency of the $p_{j,n}$-th bin in h. The third line in Equation. (13) shows that the remaining nonzero bins in the right of all the selected nonzero bins are all shifted to the right by $\sum_{k=1}^{m_j} a_{j,k}$ units. Then the frequencies of the other bins are all equal to zero. The simulated file size increments SFI_j^s caused by shifting can be calculated directly as

$$SFI_j^s = \sum_{i=1}^{N} (h_j^s(i) - h(i)) \cdot l_i, \quad (14)$$

where l_i is the length of VLC corresponding the i-th RSV and N is the number of all the bins in h.

Generally, the additional data before embedding will be encrypted to improve the security. The additional data after encrypting satisfies independently and identically distributed (i.i.d.), so the frequency of each VLC in a mapping set used to embed can be seen as the same approximately. Therefore, the RSV histogram after simulated embedding h_j^e can be represented as

$$h_j^e(i) = \begin{cases} \frac{h_j^s(p'_{j,n})}{a_{j,n}+1}, & \text{if } p'_{j,n} \leqslant i \leqslant p'_{j,n} + a_{j,n} \\ h_j^s(i), & \text{others} \end{cases} . \quad (15)$$

In Equation. (15), there are $a_{j,n}+1$ VLCs in the n-th mapping set so the frequency of each VLC in this mapping set is equal to $\frac{h_j^s(p'_{j,n})}{a_{j,n}+1}$ approximately after simulated embedding. Then the simulated file size increments SFI_j^e caused by embedding can be calculated as

$$SFI_j^e = \sum_{i=1}^{N} (h_j^e(i) - h_j^s(i)) \cdot l_i. \quad (16)$$

Finally, the SFI of the j-th solution SFI_j can be calculated as

$$\begin{aligned} SFI_j &= SFI_j^s + SFI_j^e \\ &= \sum_{i=1}^{N} (h_i^s(i) - h(i)) \cdot l_i + \sum_{i=1}^{N} (h_j^e(i) - h_j^s(i)) \cdot l_i \\ &= \sum_{i=1}^{N} (h_i^e(i) - h(i)) \cdot l_i. \end{aligned} \quad (17)$$

According to the simulated embedding model, we can obtain the optimal solution by exhaustive searching from the solution space. Fig. 11 illustrates how to obtain the optimal solution in the solution space. In Fig. 11, the given payload P is 10000 so the position of extreme bin is 3. Then the solution space can be generated. For the whole solution space, we calculate the SFI of each feasible solution and find the solution, (3,1,{1}), with minimal SFI, which is the optimal solution.

3) Construction of Optimal GVM Relationship: After obtaining the optimal solution, the optimal GVM relationship G_{opt} can be constructed. Denote the optimal solution by $(p^s, m, \{a_1, a_2, \cdots, a_m\})$, the n-th mapping set in G_{opt} can

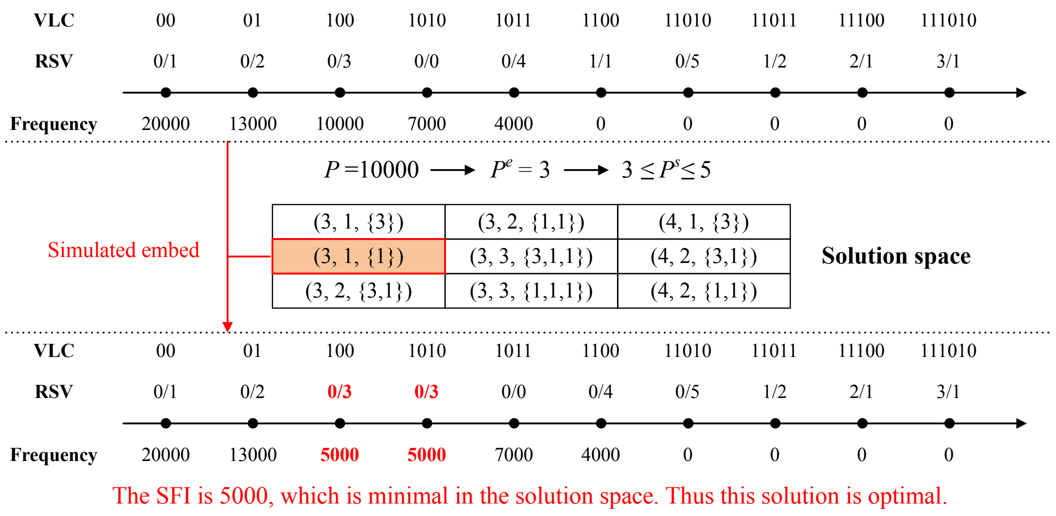


Fig. 11. The optimal solution obtained by calculating the SFI of each solution in the feasible solution space.

be represented as:

$$M_n = \{VLC_{p^s+\sum_{k=1}^{n-1}(a_k+1)} \leftrightarrow \{VLC_{p^s+\sum_{k=1}^{n-1}(a_k+1)+1}, \cdots, VLC_{p^s+\sum_{k=1}^{n} a_k}\}\}, \tag{18}$$

where M_n is the n-th mapping set. In the n-th mapping set, a_n unused VLCs are assigned to one used VLC, which is equivalent that a_n zero bins are assigned to one selected nonzero bin. Then the GVM relationship can be expressed as:

$$G_{opt} = \{M_1, M_2, \cdots, M_m\}. \tag{19}$$

D. Data Embedding & Extraction

After the optimal GVM relationship G_{opt} constructed, the additional data can be embedded by replacing the original VLC with the mapped VLC. In order to make the marked JPEG image have good format compatibility and no distortion, we modify the RSVs sequence in the DHT segment according to G_{opt}. As mentioned in section II, each RSV corresponds to one unique VLC. While in a mapping set, the frequency of the unused VLCs is zero, so the corresponding RSVs can be changed to the RSV corresponding to the used VLC. Then the RSVs corresponding to all the VLCs in one mapping set is the same, which means that the RSV is unchanged before and after embedding. Therefore, the visual quality is preserved and no error will occur during the decoding process.

When embedding the additional data, each VLC is read in the entropy-coded data successively. For the current VLC, we firstly judge whether the corresponding RSV appears multiple times in the modified DHT segment or not. Appearing multiple times means that the VLC belongs to one mapping set. If yes, we replace the VLC with another VLC according to the data to be embedded. In general, the Huffman table adopted by the original JPEG image is the default Huffman table, which can be obtained on the Internet. Thus, there is no auxiliary information to be embedded.

Based on the above discussion, the detailed data embedding steps are summarized as:

Step 1. Decode the original JPEG bitstream J and parse all the VLCs according to the Huffman table in the file header. Then reorder the corresponding RSVs in descending order.

Step 2. Generate the solution space according to III-C.

Step 3. According to the simulated embedding model, search the optimal solution by calculating the simulated file size increments for each feasible solution.

Step 4. Construct the optimal GVM relationship G_{opt} by modifying the corresponding RSVs in the file header.

Step 5. Replacing the original VLCs with the VLCs in the same mapping set according to the current data to be embedded.

Step 6. Repeat Step 5 until all the data is embedded and the marked JPEG bitstream J_M is generated.

The detailed data extraction steps are summarized as follows:

Step 1. Decode the marked JPEG image J_M and parse all the VLCs from the entropy-coded data.

Step 2. Restore the adopted GVM relationship G_{opt} according to the modified DHT segment in the file header.

Step 3. Extract the embedded data and restore the entropy-coded data according to the GVM relationship G_{opt}.

Step 4. Restore the original RSVs in the file header, then the original JPEG bitstream J is restored.

An example of the coding process for a DCT block before and after embedding is shown in Fig. 12. In Fig. 12, all the 63 AC coefficients (except the DC coefficient in the left upper corner) in a DCT block are formatted as several RSV pairs by Zigzag scanning. In the entropy coding phase, these RSV pairs are coded with the corresponding VLCs. In the original JPEG image, the RSV ”1/1” corresponds to the VLC ”1100” and ”0/0” corresponds to ”1010”. While in the modified JPEG image, there are two different VLCs corresponds to the RSV ”1/1”. To embed additional data ”0”, the VLC ”1010” is used in the coding process. Similarly, the VLC ”11010” is used to embed data ”1”. Thus, the additional data ”01” is embedded in this DCT block and the file size increases one bit.

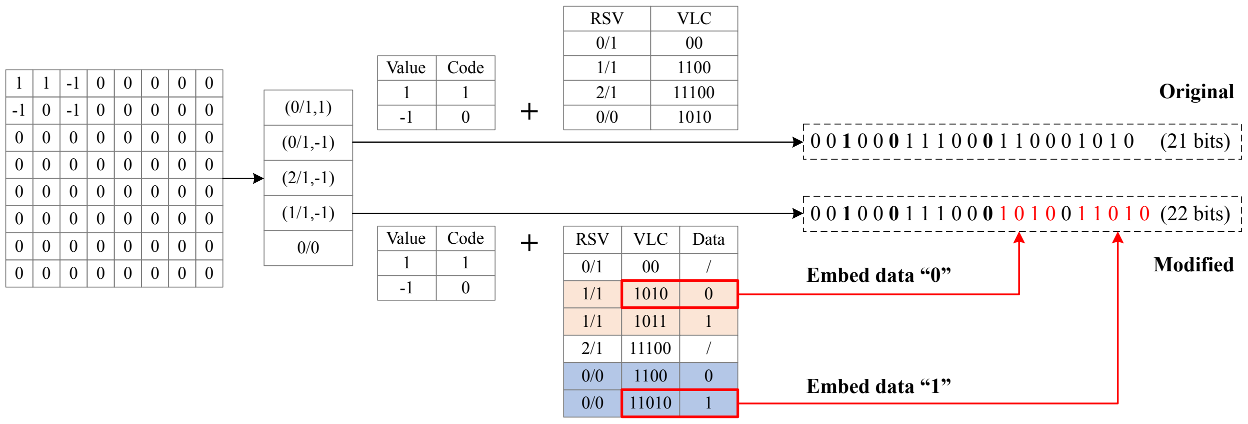


Fig. 12. Illustration of the coding process for a DCT block before and after embedding. The codes underlined are VLCs and the codes bolded are appended bits.

(a) Baboon (b) Boat (c) Elaine (d) Lena (e) Peppers (f) Tiffany

Fig. 13. The test images with QF = 70.

IV. Experimental Results

In our experiments, six 512×512 grayscale test images, including Baboon, Lena, Tiffany, Peppers, Elaine and Airplane, from the USC-SIPI database [28] are converted into grayscale JPEG image with various QFs to test. Fig. 13 illustrates these test images after converted with QF =70. Moreover, to investigate the suitability of our proposed scheme, 200 images are randomly selected from the image database UCID [29]. The selected 200 images are converted into grayscale JPEG images with four QFs including 30,50,70, and 90. In our proposed scheme, the default Huffman table is modified to construct GVM relationship. Therefore, all the test JPEG images are compressed with the default Huffman table by using the IJG toolbox [30].

To evaluate the performance of our proposed scheme, several previous VLC-based schemes are selected for comparison, which are proposed by including Qian and Zhang [25], Hu *et al.* [26], and Qiu *et al.* [27]. Several state-of-the-art DCT-based RDH schemes are also selected for comparison, which are proposed by including Huang *et al.* [21], Qian *et al.* [22], Wedaj *et al.* [23], and Hou *et al.* [24] respectively. Several experiments are conducted to demonstrate the effectiveness of the proposed scheme. Given a test JPEG image, we investigate two aspects, which are visual quality and file size preservation. The performance of visual quality is measured using the peak signal-to-noise ratio (PSNR) value between the cover JPEG image and the marked JPEG image. The performance of file size preservation is measured by the file size increments between the marked JPEG bitstream and the cover JPEG bitstream.

Noted that the program codes for the schemes [21] [22] [23] [25] [26] [27] are implemented by ourselves using MATLAB, and the program code for [24] is downloaded from http://home.ustc.edu.cn/~houdd. A Dell PC running Windows 10 with 1.6 GHz Core i5 CPU and 6 GB memory is used in our experiments. The source code of our proposed scheme is uploaded into http://github.com/odinaris/gvm.

A. User-specified Factor U

As shown in Equation. (5), the max number of selected nonzero bins m is determined by user-specified factor U to some extent. In general, if the value of U is large, the scale of solution space is large and the optimal solution is more likely to be included in. Conversely, a large solution space also brings more time costs. Therefore, there exists a trade-off between file size increments and runtime in the selection of U for our proposed scheme. To find the appropriate user-specified factor, different values of U, i.e., U=1, 3, 5, 7, 9 are used for demonstration in our experiments. 4000:1000:18000 bits of additional data are embedded into 50 JPEG images converted from the UCID image data base with QF=70. The average file size increments and average runtime are shown in Fig. 14 and Fig. 15.

It can be observed from Fig. 14 that the file size increments in marked JPEG images generated by our proposed scheme decrease as U increases from 1 to 9 regardless of the payload. Naturally, the runtime of our proposed scheme increase as U increases from 1 to 9 as shown in Fig. 15. In addition, the decrease in file size increments is not quite significant when U from 5 to 9. However, the increase in runtime is obvious when U from 5 to 9. Therefore, we choose the value of U to be 5 for our following experiments.

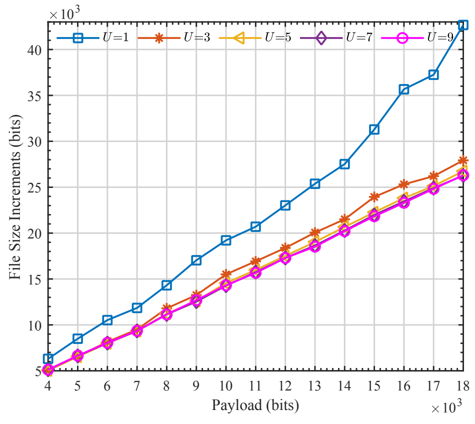


Fig. 14. Average file size increments of 50 images from the UCID image database.

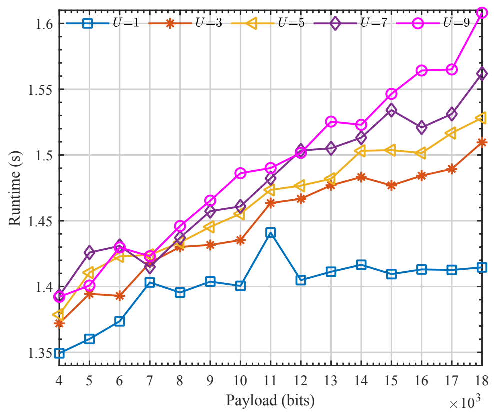


Fig. 15. Average runtime of 50 images from the UCID image database.

B. Coding Redundancy

As mentioned in Section III-A, for a JPEG image compressed by using the default Huffman table, the coding redundancy exists since the original RSVs are not in a descending order strictly. The coding redundancy for the six test images with various QFs are described in Fig. 16. For one test image, the coding redundancy at different QFs are different and high, which demonstrates the default Huffman table is not the most suitable no matter which the QFs. Furthermore, the lowest redundancy is often obtained with the higher QFs, such as 70, 80, and 90, which indicates that at the high QFs, the frequency order of the original RSVs in JPEG bitstream is more close to the order defined by the default Huffman table.

In the following experiments, the actual file size increments of our proposed scheme are called pure file size increments (PFI). For a fair comparison, the file size increments caused in the shifting and embedding processes are also listed, which called grass file size increments (GFI). The GFI can be calculated by subtracting the coding redundancy from the PFI approximately.

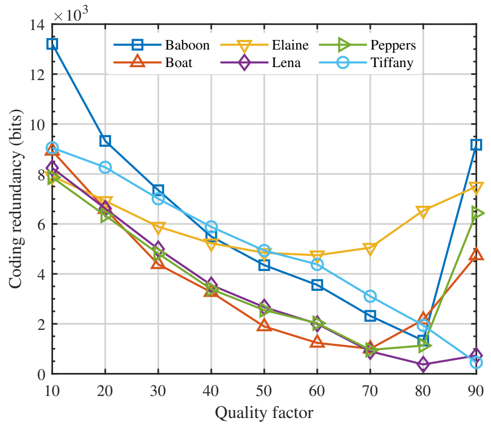
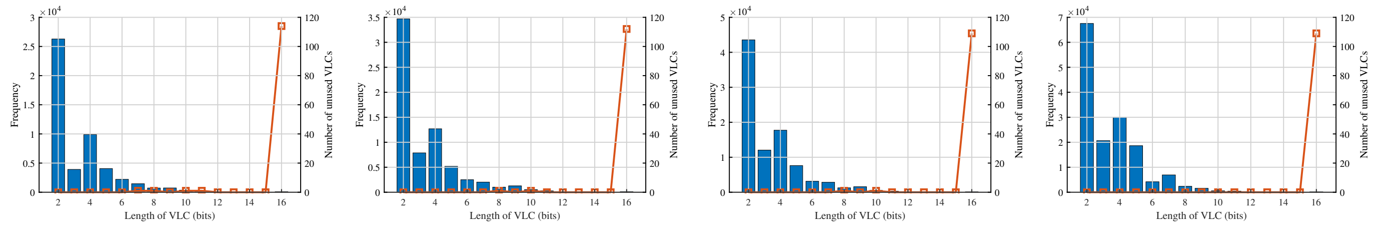


Fig. 16. Coding redundancy (bits) with various QFs of different test images.

C. Comparison with VLC-based schemes

For the previous VLC-based schemes, the lengths of used VLC and the mapped unused VLCs should be the same. The VLCs of different lengths can be classified to 16 categories. While for the most categories, the number of unused VLCs is quite low and even zero. Fig. 17 describes the distribution of VLCs for *Baboon* with different QFs. The blue histogram shows the frequency of VLCs in different lengths. The orange line chart is the number of unused VLCs in different lengths. As Fig. 17 illustrated, for the categories with a high frequency which are also the VLCs of short length, the number of unused VLCs is quite low and close to zero. While for the categories with low frequency, which is the VLCs with a length of 16, the number of unused VLCs is much high. This is consistent with the characteristics of Huffman coding. Most VLCs with short lengths are used so the number of unused VLCs of short length is close or equal to zero. Therefore, for the previous VLC-based schemes, the embedding capacity is difficult to be improved significantly.

In our proposed scheme, the embedding capacity is different using the different GVM relationships. The GVM relationship is determined by multiple parameters, such as the first selected nonzero bin p_s, the number of selected nonzero bins m, and the numbers of zero bins assigned to each selected bin $\{a_1, a_2, \cdots, a_m\}$. To compare the embedding capacity with the previous VLC-based schemes, we designed two combinations of the three parameters. The first combination of the three parameters is $(1, 1, \{1\})$. That is only one nonzero bin is selected, which is on the far left, and the number of zero bins assigned to it is 1. The second combination of the three parameters are $(1, 2, \{1, 1\})$. Two nonzero bins are selected, which are on the far left, and the number of zero bins assigned to each bin is all 1. Table I lists the comparison of embedding capacity and file size increments with previous VLC-based schemes. As Table I illustrates, the embedding capacity of previous VLC-based schemes is rather low. By slightly increasing the file size, the embedding capacity is improved significantly. It can be easily seen from Table I that the PFI is smaller than the capacity in some cases, which is because the high coding redundancy.



(a) QF=30 (b) QF=50 (c) QF=70 (d) QF=90

Fig. 17. The distribution of VLCs for Baboon with different QFs.

TABLE I
EMBEDDING CAPACITY AND FILE SIZE INCREMENTS COMPARISON WITH PREVIOUS VLC-BASED SCHEMES FOR DIFFERENT TEST IMAGES AT DIFFERENT QFS.'-' MEANS NO FILE SIZE INCREMENTS FOR THIS SCHEME.

Image	Scheme	QF=30			QF=50			QF=70			QF=90		
		Capacity(bits)	GFI(bits)	PFI(bits)	Capacity(bits)	GFI(bits)	PFI(bits)	Capacity(bits)	GFI(bits)	PFI(bits)	Capacity(bits)	GFI(bits)	PFI(bits)
Baboon	Qian and Zhang [25]	1592	–	–	1117	–	–	1089	–	–	342	–	–
	Hu *et al.* [26]	1739	–	–	1254	–	–	1348	–	–	662	–	–
	Qiu *et al.* [27]	1966	–	–	1473	–	–	1620	–	–	926	–	–
	Proposed (1,1,{1})	16335	19443	12093	20588	27935	23586	25251	38934	36616	38776	66977	57807
	Proposed (1,2,{1,1})	26279	31062	23712	34687	43411	39062	43548	60045	57727	67540	103363	94193
Boat	Qian and Zhang [25]	649	-	–	552	–	–	428	–	–	978	–	–
	Hu *et al.* [26]	685	–	–	681	–	–	667	–	–	1935	–	–
	Qiu *et al.* [27]	1242	–	–	1079	–	–	951	–	–	2165	–	–
	Proposed (1,1,{1})	8966	11319	6936	11686	14721	12829	14809	21501	20499	25553	39755	35019
	Proposed (1,2,{1,1})	13671	17270	12887	18835	24136	22244	24758	34163	33161	42438	64028	59292
Elaine	Qian and Zhang [25]	197	–	–	483	–	–	326	–	–	1002	–	–
	Hu *et al.* [26]	198	–	–	547	–	–	401	–	–	1718	–	–
	Qiu *et al.* [27]	842	–	–	974	–	–	781	–	–	1900	–	–
	Proposed (1,1,{1})	5910	8939	3046	8730	12068	7225	13978	17376	12327	31273	41092	33596
	Proposed (1,2,{1,1})	10006	13171	7278	13345	18496	13653	20740	27820	22771	49272	64073	56577
Lena	Qian and Zhang [25]	253	–	–	365	–	–	214	–	–	298	–	–
	Hu *et al.* [26]	260	–	–	367	–	–	280	–	–	576	–	–
	Qiu *et al.* [27]	916	–	–	996	–	–	759	–	–	841	–	–
	Proposed (1,1,{1})	6087	9035	4039	8284	11447	8780	11370	15031	14134	21938	30641	29911
	Proposed (1,2,{1,1})	10183	13816	8820	12923	17970	15303	17952	24765	23868	35282	48944	48214
Peppers	Qian and Zhang [25]	622	–	–	433	–	–	323	–	–	762	–	–
	Hu *et al.* [26]	707	–	–	548	–	–	449	–	–	1576	–	–
	Qiu *et al.* [27]	1077	–	–	924	–	–	768	–	–	1675	–	–
	Proposed (1,1,{1})	5893	8937	4125	8409	11449	8900	12238	14995	14039	24068	32914	26481
	Proposed (1,2,{1,1})	9989	13715	8903	12981	18005	15456	18711	24862	23906	38818	52723	46290
Tiffany	Qian and Zhang [25]	613	–	–	573	–	–	816	–	–	258	–	–
	Hu *et al.* [26]	674	–	–	775	–	–	1003	–	–	765	–	–
	Qiu *et al.* [27]	1125	–	–	1090	–	–	1284	–	–	1004	–	–
	Proposed (1,1,{1})	5499	8411	1403	7815	11083	6149	11580	14355	11252	22906	30974	30523
	Proposed (1,2,{1,1})	9595	12313	5305	12098	16791	11857	17823	23407	20304	37065	48647	48196

D. Comparison with DCT-based schemes

The comparison with the DCT-based RDH schemes mainly consists of two aspects, visual quality and file size preservation, which are measured by PSNR values and file size increments respectively. Limited to the embedding capacity, we set different payloads for the test images with different QFs. In our experiments, 3000, 6000, 9000, and 12000 bits of additional data are embedded into six test JPEG images with QF = 30. 4000, 8000, 12000, and 16000 bits of additional data are embedded into six test images with QF = 50. 5000, 10000, 15000, and 20000 bits of additional data are embedded into six test images with QF = 70. 6000, 12000, 18000, and 24000 bits of additional data are embedded into six test images with QF = 50. For the six test images, the PSNR comparison at different payloads are listed in Table II. Table II shows that the PSNR values of our scheme are always $+\infty$. That means our proposed scheme can keep the visual quality unchanged, which is a truly lossless data hiding scheme.

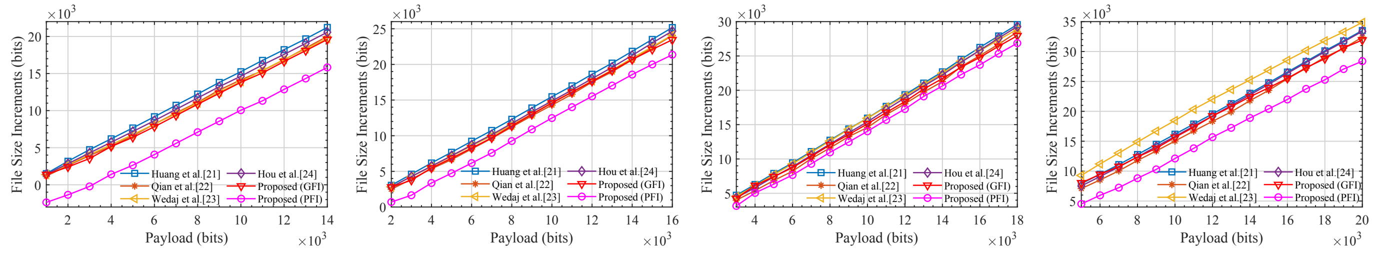
The file size increments of our proposed scheme and the four related DCT-based schemes are provided in Table III. It can be seen from Table III that the PFIs of most of the marked JPEG images generated by our scheme are much smaller than previous schemes. The lowest file size increments for different test images with different QFs are displayed in bold. Some file size increments are even negative, which means that the coding redundancy is larger than the file size increments caused by shifting and embedding. Additionally, the GFIs obtained by our scheme are close to the previous method, even smaller, which are displayed in underlined.

To further compare the performance of file size preservation, 200 images from the UCID image database are embedded into different payloads with different QFs. The average file size increments are shown in Fig. 18. It is observed from Fig. 18 that the average PFIs obtained by the proposed method are smaller than the previous schemes at different payloads and QFs, which demonstrates that our proposed scheme is highly effective for file size preservation.

For the DCT-based RDH schemes, the additional data is

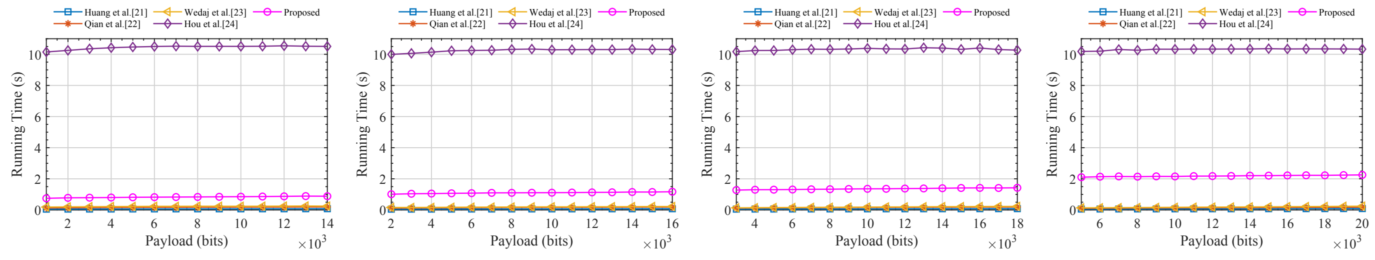
TABLE II
THE PSNR VALUES (DB) OF THE MARKED JPEG IMAGES WITH DIFFERENT QFS AND EMBEDDED WITH DIFFERENT LENGTHS OF DATA USING OUR SCHEME AND FOUR PREVIOUS SCHEMES. '-' MEANS THE CAPACITY IS NOT ENOUGH FOR THE PAYLOAD.

Image	Scheme	Payload (bits) with QF=30				Payload (bits) with QF=50				Payload (bits) with QF=70				Payload (bits) with QF=90			
		3000	6000	9000	12000	4000	8000	12000	16000	5000	10000	15000	20000	6000	12000	18000	24000
Baboon	Huang *et al.* [21]	41.96	37.78	35.11	33.22	43.65	39.41	36.43	34.07	45.33	40.74	37.75	35.16	48.25	44.17	41.39	39.33
	Qian *et al.* [22]	41.63	37.81	35.16	33.23	43.69	39.37	36.69	34.21	45.42	40.94	37.82	35.18	48.71	44.24	41.50	39.33
	Wedaj *et al.* [23]	40.55	36.97	34.78	33.19	41.97	38.39	36.11	34.44	42.86	39.48	37.54	35.71	46.25	43.23	40.83	39.33
	Hou *et al.* [24]	42.23	38.12	35.50	33.55	43.94	39.84	36.92	34.50	45.73	41.23	37.99	35.42	48.52	44.34	41.44	39.35
	Proposed	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
Boat	Huang *et al.* [21]	41.66	37.82	35.46	33.44	43.78	39.74	37.27	35.05	46.37	42.15	39.47	37.08	50.38	46.41	44.11	42.45
	Qian *et al.* [22]	41.71	37.94	35.42	33.49	44.04	39.82	37.31	35.19	46.63	42.03	39.49	37.11	51.01	46.95	44.48	42.78
	Wedaj *et al.* [23]	41.25	37.70	35.71	33.91	42.98	39.49	37.41	35.57	45.87	41.90	39.57	37.82	50.63	47.13	45.44	43.52
	Hou *et al.* [24]	42.28	38.40	35.89	33.83	44.51	40.46	37.73	35.30	47.13	42.76	39.91	37.27	50.82	46.64	44.17	42.45
	Proposed	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
Elaine	Huang *et al.* [21]	42.54	39.08	36.67	–	44.55	41.15	39.11	37.03	46.56	43.22	41.13	39.44	50.15	45.92	43.36	41.53
	Qian *et al.* [22]	42.55	38.89	36.64	–	44.69	41.40	39.14	37.09	46.98	43.58	41.43	39.75	51.13	46.79	43.87	42.00
	Wedaj *et al.* [23]	42.69	38.84	36.61	–	45.71	41.37	39.35	37.19	47.94	44.02	42.14	40.26	50.68	48.00	46.01	44.57
	Hou *et al.* [24]	43.39	39.21	36.65	–	45.77	41.83	39.26	37.11	47.72	44.08	41.64	39.72	51.07	46.50	43.65	41.84
	Proposed	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
Lena	Huang *et al.* [21]	42.45	38.13	35.16	–	45.18	40.56	37.33	–	48.02	43.83	40.51	37.26	53.48	49.84	47.50	45.20
	Qian *et al.* [22]	42.47	38.14	35.00	–	45.20	40.59	37.37	–	48.06	43.77	40.52	37.26	53.75	50.20	47.77	45.24
	Wedaj *et al.* [23]	40.99	37.92	35.17	–	44.40	40.27	37.49	–	46.75	42.83	40.45	37.32	52.82	48.99	47.15	45.27
	Hou *et al.* [24]	42.57	38.35	35.21	–	45.41	40.86	37.47	–	48.32	43.97	40.70	37.26	53.99	50.17	47.56	45.24
	Proposed	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
Peppers	Huang *et al.* [21]	42.56	37.72	34.92	–	45.20	40.96	37.65	–	47.98	43.99	41.24	38.39	51.94	48.11	45.64	43.57
	Qian *et al.* [22]	42.25	37.70	35.00	–	45.07	41.08	37.68	–	47.82	43.94	41.31	38.48	52.95	49.39	46.79	44.46
	Wedaj *et al.* [23]	41.75	37.68	35.27	–	44.35	40.85	38.00	–	47.63	43.87	41.05	39.01	51.80	49.23	47.20	45.47
	Hou *et al.* [24]	42.64	38.49	35.26	–	45.91	41.48	38.22	–	48.63	44.65	41.72	38.56	53.14	48.84	46.00	43.84
	Proposed	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$
Tiffany	Huang *et al.* [21]	42.59	37.99	35.46	–	45.09	40.31	37.48	–	48.24	43.29	40.38	37.62	53.71	50.09	47.39	44.91
	Qian *et al.* [22]	42.65	38.00	35.56	–	45.23	40.62	37.75	–	48.27	43.46	40.64	37.78	53.73	50.28	47.63	44.99
	Wedaj *et al.* [23]	41.81	38.34	35.98	–	44.82	40.69	38.25	–	47.12	43.53	41.20	38.13	51.81	49.10	47.57	45.51
	Hou *et al.* [24]	42.75	38.80	35.68	–	45.49	41.16	37.78	–	48.73	44.18	40.84	37.69	54.30	50.61	47.70	45.01
	Proposed	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$	$+\infty$



(a) UCID (QF=30) (b) UCID (QF=50) (c) UCID (QF=70) (d) UCID (QF=90)

Fig. 18. Average file size increments of 200 images from the UCID image database.



(a) QF=30 (b) QF=50 (c) QF=70 (d) QF=90

Fig. 19. Average running time of 200 images from the UCID image database.

TABLE III
THE FILE SIZE INCREMENTS (BITS) OF THE MARKED JPEG IMAGES WITH DIFFERENT QFS AND EMBEDDED WITH DIFFERENT LENGTHS OF DATA USING OUR SCHEME AND FOUR PREVIOUS SCHEMES. '-' MEANS THE CAPACITY IS NOT ENOUGH FOR THE PAYLOAD

Image	**Scheme**	**Payload(bits) with QF=30**				**Payload(bits) with QF=50**				**Payload(bits) with QF=70**				**Payload(bits) with QF=90**			
		3000	6000	9000	12000	4000	8000	12000	16000	5000	10000	15000	20000	6000	12000	18000	24000
Baboon	Huang*et al.* [21]	4200	8352	12920	17024	5712	10992	16904	23992	7408	15168	23312	31920	10264	20264	30832	41256
	Qian*et al.* [22]	2992	7296	11624	14880	5040	9104	16568	20920	6824	13776	22808	28304	8456	20088	28512	37408
	Wedaj*et al.* [23]	3400	7560	12096	15760	5280	10688	16680	22800	8512	16504	24816	33640	14904	27624	38104	47176
	Hou*et al.* [24]	3840	8312	12432	16416	5344	10680	16400	22936	6784	14464	22504	31224	9760	19600	30968	40776
	Proposed (GFI)	4063	7032	11223	14903	5959	11208	17664	23355	8307	13585	23028	29780	6885	17576	26576	32435
	Proposed (PFI)	**-3287**	**-318**	**3873**	**7553**	**1610**	**6859**	**13315**	**19006**	**5989**	**11267**	**20710**	**27462**	**-2285**	**8406**	**17406**	**23265**
Boat	Huang*et al.* [21]	4384	8664	12496	16752	5832	11720	17848	24192	8096	15512	23272	31472	9496	20096	30944	40992
	Qian*et al.* [22]	3680	8088	10488	14784	5040	10008	15304	23048	**6176**	**12408**	**20448**	**29440**	9152	18824	29728	37136
	Wedaj*et al.* [23]	3880	8224	11352	15968	5168	10904	16704	23592	6992	14592	22208	30568	9544	19824	28352	38952
	Hou*et al.* [24]	3920	7784	12168	16192	5408	10920	16840	23848	7544	14608	22336	30728	8944	19680	30584	41024
	Proposed (GFI)	4018	8161	11548	16431	5476	10750	17061	22715	8176	14681	23883	30685	11267	21661	31800	39753
	Proposed (PFI)	**-365**	**3778**	**7165**	**12048**	**3584**	**8858**	**15169**	**20823**	7174	13679	22881	29683	**6531**	**16925**	**27064**	**35017**
Elaine	Huang*et al.* [21]	4144	8288	12112	–	6688	13024	18216	24952	8744	16200	23680	30000	9440	19160	28792	38680
	Qian*et al.* [22]	3624	6608	12112	–	5176	10992	15936	23608	6144	12416	18632	27544	7840	17736	27624	36952
	Wedaj*et al.* [23]	4376	8840	11696	–	5568	11200	15928	23408	6024	11600	17576	25872	8416	17984	26416	34136
	Hou*et al.* [24]	4720	8328	12160	–	6120	12408	17944	24056	7536	14680	21672	28232	9088	18592	28496	38088
	Proposed (GFI)	3384	6765	11265	16372	5357	10182	15467	24076	7019	14233	22491	27450	10020	21566	31678	41090
	Proposed (PFI)	**-2509**	**872**	**5372**	**10479**	**514**	**5339**	**10624**	**19233**	**1970**	**9184**	**17442**	**22401**	**2524**	**14070**	**24182**	**33594**
Lena	Huang*et al.* [21]	3784	8000	12568	–	5568	11616	17680	–	7080	14392	21648	30688	9336	17560	25464	35256
	Qian*et al.* [22]	3528	8136	11952	–	5232	10304	16832	–	6952	**12560**	**19952**	30352	7136	**14136**	**22936**	34896
	Wedaj*et al.* [23]	4128	7672	12392	–	5464	10808	17144	–	**6064**	13712	21160	30192	**6800**	15520	24160	**33352**
	Hou*et al.* [24]	4056	7936	12624	–	5488	11240	17232	–	7440	14128	21368	30784	7624	16248	25192	34584
	Proposed (GFI)	3282	7551	13223	16942	4712	10422	17505	22843	7176	14757	21666	31066	9114	18181	30641	39488
	Proposed (PFI)	**-1714**	**2555**	**8227**	**11946**	**2045**	**7755**	**14838**	**20176**	6279	13860	20769	**30169**	8384	17451	29911	38758
Peppers	Huang*et al.* [21]	3640	8232	12576	–	5792	11104	17472	–	7328	14440	21728	29432	9128	17808	26352	36744
	Qian*et al.* [22]	3088	7264	12288	–	4536	10992	16256	–	**5280**	**11664**	20768	29536	7208	14080	23824	35160
	Wedaj*et al.* [23]	4712	8168	12312	–	6160	10648	16696	–	6376	12848	**20352**	**28880**	8352	16256	22864	31992
	Hou*et al.* [24]	3520	7640	12568	–	5808	10832	16712	–	7104	13656	21184	28992	8288	16624	**26192**	36744
	Proposed (GFI)	3135	7344	13113	16783	4779	10409	17513	22984	6917	13011	21949	31873	9868	19569	32914	32914
	Proposed (PFI)	**-1677**	**2532**	**8301**	**11971**	**2230**	**7860**	**14964**	**20435**	5961	12055	20993	30917	**3435**	**13136**	**26481**	**26481**
Tiffany	Huang*et al.* [21]	3760	8536	12976	–	5640	12144	18608	–	6824	14816	21672	30600	8720	16224	25304	35400
	Qian*et al.* [22]	3168	6840	12232	–	5160	11168	17280	–	6784	14656	20080	29768	**6432**	**15608**	**22824**	**34848**
	Wedaj*et al.* [23]	4088	7920	12232	–	5272	11368	17200	–	6048	13856	20232	29864	6864	16232	23448	33896
	Hou*et al.* [24]	3544	7576	12696	–	5736	11216	18168	–	7032	14096	21536	30448	7760	16080	24840	34960
	Proposed (GFI)	4211	7130	11806	15179	4702	10210	16737	21123	6590	13809	21992	30480	9119	22815	29846	41118
	Proposed (PFI)	**-2797**	**122**	**4798**	**8171**	**-232**	**5276**	**11803**	**16189**	**3487**	**10706**	**18889**	**27377**	8668	22364	29395	40667

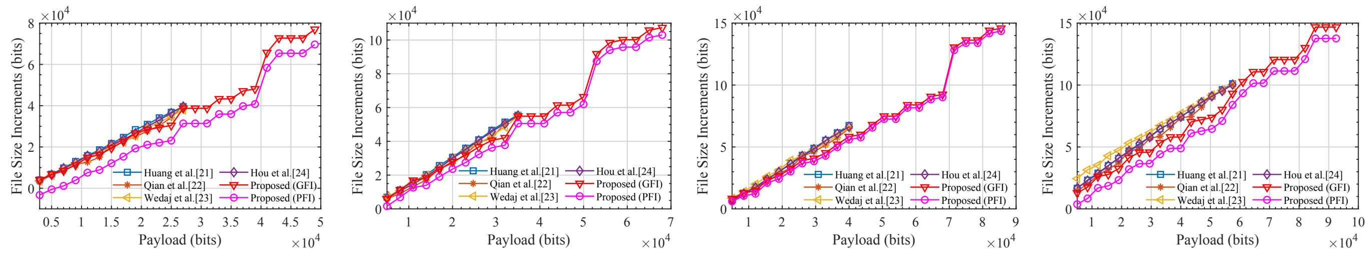
embedded by modifying the DCT coefficients, whereas the data is embedded by replacing the VLCs in JPEG bitstream for our proposed scheme. Since the embedding mechanism is different, the computational complexity cannot be compared and we only compare the running time between our proposed schemes and the DCT-based RDH schemes. The average running time of 200 images from the UCID images database is is illustrated in Fig. 19. As Fig. 19 shows, the running time of our proposed scheme is larger than the three schemes [21] [22] [23] but less than Hou *et al.*'s scheme [24]. In addition, the running time of our proposed scheme increases as the QF increases, the reason is that the frequencies of VLCs in entropy-coded data increase as the QF increases so the time cost in encoding increase with it.

The comparison of the performance in high embedding capacity is also considered. We compare the embedding capacity with previous DCT-based schemes for the image Baboon at different QFs, which is illustrated in Fig. 20. It can be observed from Fig. 20 that the upper bound of embedding capacity of the previous DCT-based schemes is limited, while the file size increments are also higher than the proposed scheme. Moreover, the upper bound of the embedding capacity of the proposed scheme can be higher if need. Note that there exists a sharp increment at the high payload for our proposed scheme, which means the performance of the proposed scheme after this sharp increment may not be an acceptable level. For the image Baboon with QF of 70, the PFI is close to GFI, which is because the coding redundancy is quite small.

V. Conclusion

In this paper, we proposed a new LDH scheme for JPEG images by constructing the optimal GVM relationship. The experimental results demonstrated that, at most of the time, the file size increments caused by the proposed scheme is smaller than the previous RDH schemes under the identical payload. Moreover, the visual quality of the marked JPEG image is preserved. The upper bound of the embedding capacity of our proposed scheme is also much larger than state-of-the-arts works. The main contributions of this paper are summarized as follows:

1) A general VLC mapping strategy is proposed, which allow the length of VLCs in one mapping set can be unequal.
2) The VLC mapping model is converted into the histogram shifting model. Then the feasible solution space is gener-



(a) QF=30 (b) QF=50 (c) QF=70 (d) QF=90

Fig. 20. High embedding capacity comparison with previous DCT-based schemes for image Baboon with different QFs.

ated and a simulated embedding model is established to find the optimal solution by exhaustive searching.

However, the limitation of our scheme cannot be ignored. Our proposed scheme is only implemented on the JPEG images compressed by the default Huffman table. Therefore, the proposed scheme cannot be applied to the JPEG images using the optimized Huffman table directly. In future work, we will utilize a more effective algorithm to solve the optimization problem of the proposed scheme more fastly. Besides, we will apply the proposed scheme to the JPEG images compressed by using the optimized Huffman table.

Acknowledgment

This research work is partly supported by National Natural Science Foundation of China (61872003, U1636206).

References

[1] D. Hou, W. Zhang, J. Liu, S. Zhou, D. Chen, and N. Yu, "Emerging applications of reversible data hiding," in *Proceedings of the 2nd International Conference on Image and Graphics Processing*. ACM, 2019, pp. 105–109.

[2] W. Hong, T.-S. Chen, and J. Chen, "Reversible data hiding using delaunay triangulation and selective embedment," *Information Sciences*, vol. 308, pp. 140–154, 2015.

[3] Z. Zhang and W. Zhang, "Reversible steganography: Data hiding for covert storage," in *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE, 2015, pp. 753–756.

[4] J. Liu, D. Hou, W. Zhang, and N. Yu, "Reversible adversarial examples," *arXiv preprint arXiv:1811.00189*, 2018.

[5] Z. Qian, H. Zhou, X. Zhang, and W. Zhang, "Separable reversible data hiding in encrypted jpeg bitstreams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 6, pp. 1055–1067, 2016.

[6] S. Zheng, Y. Wang, and D. Hu, "Lossless data hiding based on homomorphic cryptosystem," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[7] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding—new paradigm in digital watermarking," *EURASIP Journal on Advances in Signal Processing*, vol. 2002, no. 2, p. 986842, 2002.

[8] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless generalized-lsb data embedding," *IEEE transactions on image processing*, vol. 14, no. 2, pp. 253–266, 2005.

[9] J. Tian, "Reversible data embedding using a difference expansion," *IEEE transactions on circuits and systems for video technology*, vol. 13, no. 8, pp. 890–896, 2003.

[10] I.-C. Dragoi and D. Coltuc, "Local-prediction-based difference expansion reversible watermarking," *IEEE Transactions on image processing*, vol. 23, no. 4, pp. 1779–1790, 2014.

[11] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 354–362, 2006.

[12] X. Zhang, "Reversible data hiding with optimal value transfer," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 316–325, 2012.

[13] W. Zhang, X. Hu, X. Li, and N. Yu, "Recursive histogram modification: establishing equivalency between reversible data hiding and lossless data compression," *IEEE transactions on image processing*, vol. 22, no. 7, pp. 2775–2785, 2013.

[14] Y. Jia, Z. Yin, X. Zhang, and Y. Luo, "Reversible data hiding based on reducing invalid shifting of pixels in histogram shifting," *Signal Processing*, vol. 163, pp. 238–246, 2019.

[15] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding for all image formats," in *Security and Watermarking of Multimedia Contents IV*, vol. 4675. International Society for Optics and Photonics, 2002, pp. 572–584.

[16] K. Wang, Z.-M. Lu, and Y.-J. Hu, "A high capacity lossless data hiding scheme for jpeg images," *Journal of systems and software*, vol. 86, no. 7, pp. 1965–1975, 2013.

[17] J. Fridrich, M. Goljan, Q. Chen, and V. Pathak, "Lossless data embedding with file size preservation," in *Security, Steganography, and Watermarking of Multimedia Contents VI*, vol. 5306. International Society for Optics and Photonics, 2004, pp. 354–366.

[18] G. Xuan, Y. Q. Shi, Z. Ni, P. Chai, X. Cui, and X. Tong, "Reversible data hiding for jpeg images based on histogram pairs," in *International Conference Image Analysis and Recognition*. Springer, 2007, pp. 715–727.

[19] Q. Li, Y. Wu, and F. Bao, "A reversible data hiding scheme for jpeg images," in *Pacific-Rim Conference on Multimedia*. Springer, 2010, pp. 653–664.

[20] B. Chen, W. Zhang, K. Ma, and N. Yu, "Recursive code construction for reversible data hiding in dct domain," *Multimedia tools and applications*, vol. 72, no. 2, pp. 1985–2009, 2014.

[21] F. Huang, X. Qu, H. J. Kim, and J. Huang, "Reversible data hiding in jpeg images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 9, pp. 1610–1621, 2016.

[22] Z. Qian, S. Dai, and B. Chen, "Reversible data hiding in jpeg images using ordered embedding," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 11, no. 2, pp. 945–958, 2017.

[23] F. T. Wedaj, S. Kim, H. J. Kim, and F. Huang, "Improved reversible data hiding in jpeg images based on new coefficient selection strategy," *EURASIP Journal on Image and Video Processing*, vol. 2017, no. 1, p. 63, 2017.

[24] D. Hou, H. Wang, W. Zhang, and N. Yu, "Reversible data hiding in jpeg image based on dct frequency and block selection," *Signal Processing*, vol. 148, pp. 41–47, 2018.

[25] Z. Qian and X. Zhang, "Lossless data hiding in jpeg bitstream," *Journal of Systems and Software*, vol. 85, no. 2, pp. 309–313, 2012.

[26] Y. Hu, K. Wang, and Z.-M. Lu, "An improved vlc-based lossless data hiding scheme for jpeg images," *Journal of Systems and Software*, vol. 86, no. 8, pp. 2166–2173, 2013.

[27] Y. Qiu, H. He, Z. Qian, S. Li, and X. Zhang, "Lossless data hiding in jpeg bitstream using alternative embedding," *Journal of Visual Communication and Image Representation*, vol. 52, pp. 86–91, 2018.

[28] "The usc-sipi image database," http://sipi.usc.edu/database.

[29] G. Schaefer and M. Stich, "Ucid: An uncompressed color image database," in *Storage and Retrieval Methods and Applications for Multimedia 2004*, vol. 5307. International Society for Optics and Photonics, 2003, pp. 472–481.

[30] "Independent jpeg group," http://www.ijg.org.

Yang Du received his bachelor degree in software engineering in 2018 and now is a master student in the School of Computer Science and Technology, Anhui University. His current research interests include reversible data hiding for JPEG images and steganography.

Zhaoxia Yin received her B.Sc., M.E. & Ph.D. from Anhui University in 2005, 2010 and 2014 respectively. She is an IEEE/ACM/CCF member, CSIG senior member and the Associate Chair of the academic committee of CCF YOCSEF Hefei 2016–2017. She is currently working as an Associate Professor and a Ph.D advisor in School of Computer Science and Technology at Anhui University. She is also the Principal Investigator of two NSFC Projects. Her primary research focus including Information Hiding, Multimedia Security and she has published many SCI/EI indexed papers in journals, edited books and refereed conferences.

Xinpeng Zhang received the B.S. degree in computational mathematics from Jilin University, China, in 1995, and the M.E. and Ph.D. degrees in communication and information system from Shanghai University, China, in 2001 and 2004, respectively. Since 2004, he had been with the faculty of the School of Communication and Information Engineering, Shanghai University, and he is currently a Professor of School of Computer Science, Fudan University. He was with the State University of New York at Binghamton as a visiting scholar from January 2010 to January 2011, and Konstanz University as an experienced researcher sponsored by the Alexander von Humboldt Foundation from March 2011 to May 2012. He served IEEE Transactions on Information Forensics and Security as an Associate Editor from 2014 to 2017. His research interests include multimedia security, image processing, and digital forensics. He has published more than 200 papers in these areas.