symmetry

MDPI

Article

A Hybrid Data Hiding Method for Strict AMBTC Format Images with High-Fidelity

Chin-Chen Chang [1,2], Xu Wang [1,*] and Ji-Hwei Horng [3]

[1] Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan; alan3c@gmail.com

[2] School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China

[3] Department of Electronic Engineering, National Quemoy University, Kinmen 89250, Taiwan; horng@email.nqu.edu.tw

* Correspondence: wx1990555@gmail.com; Tel.: +886-4-2706-6495

Received: 11 September 2019; Accepted: 18 October 2019; Published: 19 October 2019

Abstract: With the rapid development of smartphones, cloud storage, and wireless communications, protecting the security of compressed images through data transmission on the Internet has become a critical contemporary issue. A series of data hiding methods for AMBTC compressed images has been proposed to solve this problem. However, most of these methods either change the file size of the final compressed code or exchange the order of two quantization values in some blocks. To reverse this situation, this paper proposes a data hiding method for strict AMBTC format images using a hybrid strategy: replacement, matrix encoding, and symmetric quantization value embedding for three block types i.e., smooth blocks, less complex blocks and highly complex blocks. According to the hybrid strategy, an efficient data hiding order is designed to achieve higher-fidelity. Experimental results show that our proposed method provides an excellent balance between image quality and hiding capacity and has no error blocks in the final stego-compressed code.

Keywords: data hiding; strict AMBTC format; high-fidelity; matrix coding

1. Introduction

People all around the world are increasingly using the Internet and a tremendous amount of information is transferred every day. When sensitive data is transmitted over the Internet, malicious actors may use various techniques to intercept messages and steal information. Therefore, information security has become a critical issue and many methods have been proposed to protect it. In recent years, multimedia files become prevalent in network transmissions, and some efficient data hiding techniques have been proposed for such applications. Data hiding techniques embed secret data into various multimedia files, such as image, audio, video, and in other forms for transmission over the network, so that others are unaware of the existing embedded data [1]. Because of the practicability and convenience of images, adopting a data hiding method to embed secret data in a digital image is of keen research interest [2].

Current data hiding methods are mainly divided into four types: data hiding in the spatial domain, the transform domain, the encryption domain and the compressed domain. In the spatial domain, the difference expansion technique and the histogram shifting technique are two of the most famous methods. The difference expansion technique uses differences between neighboring pixel values to explore redundancy and to find the embedding area [3]. The histogram shifting technique treats pixels as a histogram and vacates room for data embedding by shifting the points between the peak and zero points toward the zero point of the histogram [4]. Due to higher operability, data hiding in the spatial domain has attracted wide attention, and a series of related methods have been proposed to improve the performance of hiding capacity and image quality [5–8]. In the transform domain, images are first converted into frequency coefficients using different techniques such as discrete wavelet transform (DWT) [9], discrete Fourier transform (DFT) [10], or discrete cosine transform (DCT) [11]. Subsequently, redundant coefficients can be used for data hiding. In the encryption domain, images are encrypted before transmission over the Internet and no useful content is accessible to others [12]. Then, the data hider will embed data into the encrypted image.

With the rapid development of smartphones, cloud storage, and wireless communications, compressed images are widely used for such applications because of fast transmission and reduced storage requirements. Therefore, data hiding methods in the compressed domain have received more attention. Various data hiding methods in different compression techniques have been proposed, such as vector quantization (VQ) [13], side match vector quantization (SMVQ) [14], joint photographic experts group (JPEG) [15], block truncation coding (BTC), and absolute moment block truncation coding (AMBTC). The BTC compression method divides an image into non-overlapping blocks and stores two quantization values and a bitmap of each block as one compressed block code [16]. AMBTC is an improved version of BTC [17], and has advantages both in computing speed and image quality. A detailed description is provided in Section 2.1. Because the AMBTC compression technique is easy to implement, many data hiding methods have been proposed based on AMBTC compressed images.

To date, data hiding methods in AMBTC compressed images are mainly divided into two categories: embedding data into AMBTC compressed images while changing the final file size, and embedding data into AMBTC compressed images while preserving the final file size. For the first category, Lin et al. [18] proposed a novel data hiding method by adjusting the two quantization values of each block according to the combination of the secret bit and the bitmap. After embedding, one 4×4 block contains four quantization values and one double sized bitmap. Kim et al. [19] applied histogram modification to use ′1s′ of the bitmap to embed almost 8 bits of secret data and to add one new quantization value for one block to reduce the compression ratio while increasing image quality compared with [18]. Later, Chen et al. [20] adopted four disjointed embedding strategies according to the four corner values of the bitmap. This method can achieve the same hiding capacity and compression ratio as [18] and offer almost the same image quality as [19]. Malik et al. [21] converted secret data into a ternary system which can embed, at most, 23 bits into one block by adding or subtracting the ternary secret data; as a result, four new quantization values are added and the size of the bitmap changes to 32 bits. Huynh et al. [22] provided a minima–maxima preserving algorithm to replace 1 LSB or 2 LSBs of the reconstructed image block to achieve an almost 2 bpp hiding capacity, but the final stego-code has the same size as the original image. All the above methods will increase the final size of the compressed image code. Sun et al. [23] proposed a JNC predictor-based RDH method to embed four bits into one modified compressed block code and reduced the size of the final compressed image codes at the same time. Later, Hong et al. proposed two methods [24,25] to improve the performance of Sun et al.'s method by achieving a higher hiding capacity and lower bitrate, i.e., a smaller file size. A MED predictor was used in these two methods and a series of improvements were proposed to achieve better performance.

All of the above methods will increase or decrease the size of the final stego-compressed image code, and these significant changes in file size will no doubt become a clue for malicious actors. For the second category, Chuang and Chang [26] first proposed a bitmap replacement method that can embed 16 bits of data into a block when the block is smooth while preserving edge blocks. Hong et al. [27] proposed a method where a compressed block code (L, H, BM) is stored when the embedded secret bit is '0' and changes to (H, L, $\overline{BM}$) when the embedded secret bit is '1', and cannot embed data when L is equal to H. Combining advantages of these two methods, Ou and Sun [28] divided blocks into smooth blocks and complex blocks, then replaced the bitmap with secret data and recalculated two quantization values in a smooth block, and used Hong et al.'s method to embed one bit into a complex block. Later, Chen and Chi [29] sub-divided complex blocks into less complex blocks and highly complex blocks, then used the method of [28] to process the smooth blocks and less complex blocks while embedding 2 bits into a highly complex block by changing the number of '1's in the bitmap to even or odd. Kumar et al. [30] further improved the above methods by using the Hamming distance and pixel value differencing to embed 8 bits into a less complex block and 5 bits into a highly complex block, respectively. In the AMBTC compressed image, each block is strictly represented by a code (*L*, *H*, *BM*). However, except for [26], all of the other methods will exchange the order of two quantization values in some blocks of the final stego-compressed code, and thus, the reversed two quantization values will undoubtedly draw attacker attention. In this paper, we treat these blocks as the error blocks and try to avoid them.

After reviewing the previous AMBTC-based data hiding methods, some methods will change the size of the final compressed image code, while other methods will exchange the order of two quantization values. These two obvious changes are no doubt violating the file format. Thus, in this paper, we propose a data hiding method for strict AMBTC format images. Three different strategies are adopted in three block types: smooth blocks, less complex blocks and highly complex blocks. The first two strategies preserve the difference and the third strategy will not reduce the difference of each block, so we can define the hiding order of the difference from low to high when the block belongs to the smooth or less complex blocks, and from left to right and top to bottom of the image when the block belongs to the highly complex blocks. Therefore, our proposed method provides high-fidelity when the hiding amount is lower.

The rest of this paper is organized as follows. Section 2 introduces the AMBTC compression technique and matrix coding technique. Section 3 explains block classification and the three different strategies of our proposed method and offers some simple examples to illustrate the embedding and extraction phases. Section 4 provides the experimental results and discussions, including visual comparisons, PSNR under different amounts of the hiding data, and performance under the various thresholds. Finally, Section 5 gives some conclusions.

2. Related Work

Two important techniques are used in our proposed method: absolute moment block truncation coding (AMBTC) compression and matrix coding. These techniques are briefly introduced in Sections 2.1 and 2.2, respectively.

2.1. Absolute Moment Block Truncation Coding (AMBTC)

In 1979, an efficient block-based lossy image compression technique named block truncation coding (BTC) was proposed by Delp and Mitchell [16]. In order to achieve compression, an image is divided into several non-overlapping blocks and stores only one bitmap and two corresponding quantization values of each block. Because the algorithm utilized in the BTC technique has significant computational complexity, Lema and Mitchell [17] proposed an improved version named absolute moment block truncation coding (AMBTC). In AMBTC, the image is also divided into several non-overlapping blocks with a size of $k \times k$ pixels, for each block, the mean value *AVG* can be calculated by Equation (1).

$$AVG = \frac{\sum_{j=1}^{k \times k} p_j}{k \times k}, \tag{1}$$

where p_j denotes the *j*-th pixel value in the block. The bitmap can then be constructed by comparing each pixel value with *AVG*: Set the bit value of bitmap to '0' if the pixel value p_j is less than *AVG* and set to '1' for others.

Next, Equations (2) and (3) can be applied to derive two quantization values, i.e., the low mean value *L* and the high mean value *H*, respectively.

$$L = \frac{\sum_{p_j \in B_0} p_j}{k \times k - q}, \tag{2}$$

$$H = \frac{\sum_{p_j \in B_1} p_j}{q} \tag{3}$$

where q represents the number of '1' that exist in the bitmap BM. Meanwhile, B_0 and B_1 denote two original pixel sets which the corresponding bits are '0' and '1', respectively in bitmap BM. Therefore, the final compression code of the block can be obtained by concatenating L, H, and BM, i.e., (L, H, BM). The corresponding restored block can be constructed by replacing each '0' and '1' of the bitmap with low mean value L and high mean value H, respectively.

2.2. Matrix Coding

Matrix coding method is a modified version of the (7, 4) Hamming code, and is widely used in LSB-based data hiding methods [31]. By utilizing the matrix operation, for a seven-bit sequence (also called a code-word), at most one bit is changed when embedding three secret bits. In the matrix coding method, H is the check matrix and is defined as $\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$. Equation (4) is used to obtain which bit needs to be changed.

$$z = \left(H \times CW^T\right)^T \otimes S, \tag{4}$$

where CW denotes the seven-bit sequence, and S represents the three-bit vector to be hidden. Notice that Equation (4) is calculated under the modulo-2 operation. The location of z is listed in Table 1.

Table 1. Changed bit location of corresponding z.

z	**001**	**010**	**011**	**100**	**101**	**110**	**111**
location	1	2	3	4	5	6	7

In the data extraction phase, Equation (5) can be used to extract three bits of secret data from each seven-bit sequence.

$$S = (H \times RCW^T)^T \tag{5}$$

where RCW denotes the seven-bit sequence that has been embedded. Equation (5) is also calculated under the modulo-2 operation.

3. The Proposed Method

In this section, we propose a data hiding method for AMBTC compressed images using a hybrid strategy to process different types of AMBTC compressed blocks that can maintain the strict AMBTC format. The proposed method also uses two thresholds to define three block types as done in [30]: smooth blocks, less complex blocks and highly complex blocks. Three strategies named bitmap replacement, matrix encoding, and symmetric quantization value embedding are then applied to embed data into the blocks of different types and no error blocks are generated after embedding. We also designed an efficient data hiding order to achieve higher-fidelity. In Section 3.1, we first define our block classification rule and their corresponding strategies. Detailed processes regarding the data embedding and extraction phases are described in Sections 3.2 and 3.3, respectively. Subsequently in Section 3.4, some examples are provided to illustrate the proposed method.

3.1. Block Classification and Hybrid Strategy

After AMBTC compression, the original $M \times N$ sized image is divided into i ($i = 1, 2, \ldots, \frac{M \times N}{k \times k}$) non-overlapping $k \times k$ sized blocks, and the block size is set to 4×4, i.e., $k = 4$ for the experiments. Each block has only two quantization values H_i, L_i, and one bitmap B_i, the carriers for data embedding are chosen from them. Equation (6) is first used to calculate the difference D_i. Then, according to D_i, we set two thresholds *thr1* and *thr2* to classify each block into three block types: smooth blocks, less complex blocks and highly complex blocks.

$$D_i = H_i - L_i \tag{6}$$

$$\text{Block type} = \begin{cases} \text{smooth}, & \text{if } D_i \le thr1 \\ \text{less complex}, & \text{if } thr1 \le D_i \le thr2 \\ \text{highly complex}, & \text{otherwise} \end{cases}$$

Figure 1 shows an example of block classification under different thresholds. The complex blocks are the edges of the image and more data are embedded into edges will lead to a significant decline in image quality. To maintain a balance between embedding capacity and image quality, a hybrid strategy contains three different hiding methods is used to process different block types for different sized data embedding.

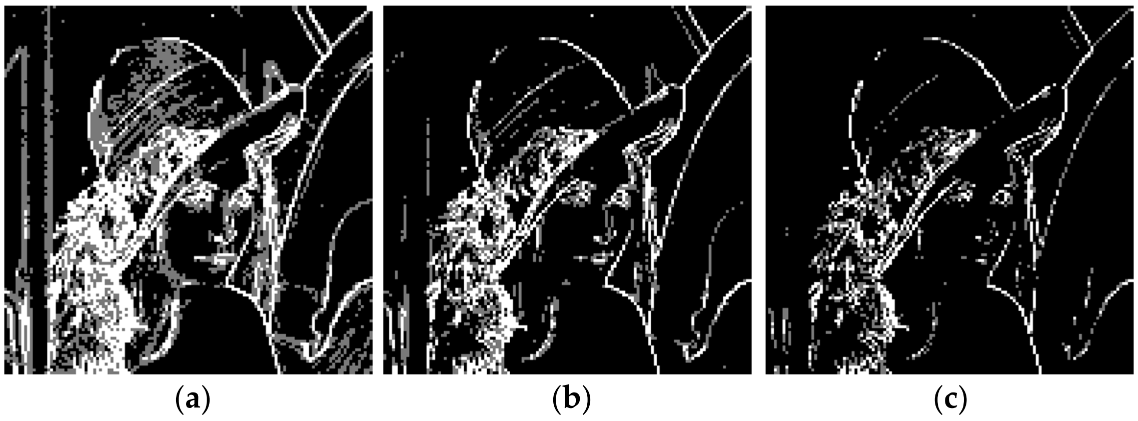


Figure 1. Examples of block classification for Lena with different thresholds: (**a**) *thr1* = 10 and *thr2* = 25; (**b**) *thr1* = 20 and *thr2* = 35; (**c**) *thr1* = 30 and *thr2* = 45. The black blocks are smooth blocks, the gray blocks are less complex blocks, and the white blocks are highly complex blocks.

The changes in the smoother blocks will reduce the image quality less. Therefore, data are embedded following the order of ascending difference value in smooth blocks and less complex blocks, while from left to right and top to bottom of the image in highly complex blocks in our method. On this account, in our hybrid hiding strategy, first two strategies will preserve blocks' difference, and guarantee blocks' difference will not decrease in the rest blocks.

3.1.1. Strategy 1 for Smooth Blocks: Bitmap Replacement

For smooth blocks, the two quantization values are very close, and thus, changes of the bitmap have a slight effect on image quality. Therefore, the bitmap can be totally replaced by 16 bits of secret data. Then, refer to the original image blocks and original pixels p_j, Equations (7) to (10) are used to calculate two new quantization values H_i' and L_i' to obtain the best image quality that is closer to the original image, i.e., minimizing the square error of the final stego-compressed block while preserving the same difference D_i. A detailed proof for the optimization is provided in the Appendix A.

$$a_i = \frac{\sum_{p_j \in B_0'} p_j}{k \times k - q'}, \tag{7}$$

$$b_i = \frac{\sum_{p_j \in B_1'} p_j}{q'}, \tag{8}$$

$$L_i' = a_i - \frac{q'(D_i - b_i + a_i)}{k \times k}, \tag{9}$$

$$H_i' = b_i + \frac{(k \times k - q')(D_i - b_i + a_i)}{k \times k}, \tag{10}$$

where q' represents the number of '1' in the new bitmap, B_0' and B_1' denote two original pixel sets which the corresponding bits are '0' and '1', respectively, in a new bitmap.

3.1.2. Strategy 2 for Less Complex Blocks: Matrix Encoding

For less complex blocks, the difference between H_i' and L_i' is large, and the image quality is going to decline dramatically if we change all the values of the bitmap like the smooth blocks. Therefore, the matrix coding that can embed three bits by at most changing one of seven bits is used in less complex blocks. First, the bitmap is divided into two non-overlapping sub-blocks, i.e., the upper sub-block and the lower sub-block. The first seven bits of each sub-block are treated as code-words for data embedding. For details of the method, please refer to Section 2.2. This strategy embeds data by modifying the bitmap and thus, does not change D_i.

3.1.3. Strategy 3 for Highly Complex Blocks: Symmetric Quantization Value Embedding

For highly complex blocks, D_i is too large, such that even a one bit change of the bitmap is intolerable. Therefore, in consideration of the image quality, we turn to hiding data by modifying two quantization values to embed 2 bits into one of them, i.e., 4 bits can be embedded into one block. To guarantee that D_i will not reduce, instead of the LSB replacement method, the symmetric quantization value embedding method is used to obtain two new nearest expanding quantization values which is given by Equations (11) to (14). After embedding, L_i' will decrease and H_i' will increase, i.e., the difference D_i will not reduce, thus, misjudgments will not happen in the data extraction phase.

$$m_L = \mathrm{mod}(L_i,\ 4), \tag{11}$$

$$m_H = \mathrm{mod}(H_i,\ 4), \tag{12}$$

$$L_i' = L_i - \mathrm{mod}[(m_L - S_{i,L}'),\ 4], \tag{13}$$

$$H_i' = H_i + \mathrm{mod}[(S_{i,H}' - m_H),\ 4], \tag{14}$$

where $S_{i,L}'$ and $S_{i,H}'$ denote two quaternary system secret data where each one is converted by 2 bit binary system secret data. When extracting data, we only apply modulo-4 operation on two quantization values and convert two results into the binary to get the embedded secret data.

By utilizing the proposed hybrid strategy, we can embed 16 bits, 6 bits, and 4 bits into one smooth block, one less complex block and one highly complex block, respectively. Additionally, in smooth blocks and less complex blocks, our method can preserve the same value of D_i, and can guarantee D_i will not reduce in highly complex blocks. Therefore, the hiding order can be used to embed data into blocks in order of D_i from low to high when $D_i \le thr2$, and embed secret data into blocks from left to right and top to bottom of the image when $D_i > thr2$, and the extraction process will not violate the order of embedding.

3.2. Data Embedding Phase

According to the block classification, in the data embedding phase, the blocks are first classified into smooth blocks, less complex blocks, and highly complex blocks based on two thresholds, *thr1* and *thr2*. Three corresponding data embedding strategies are used to embed secret data into the three block types. As mentioned before, the order of data embedding is followed in D_i from low to high when $D_i \le thr2$, and from left to right and top to bottom of the image when $D_i > thr2$. Based on the above discussions, the detailed data embedding algorithm is given below:

Input: Original $M \times N$ sized image O, secret data bit stream S, and two thresholds $thr1$, $thr2$.

Output: Stego-AMBTC compressed code C'.

Step 1. Divide the original image into $\frac{M \times N}{4 \times 4}$ non-overlapping 4×4 sized blocks. Next, apply the AMBTC compression method to compress each block and obtain the compressed block code C_i with two quantization values and one bitmap (L_i, H_i, BM_i).

Step 2. Calculate D_i by Equation (6), sort the compressed blocks according to the ascending order of D_i when $D_i \leq thr2$ and the raster scan order of image for the rest of the blocks.

Step 3. Extract the compressed block code C_i of the i-th block according to the ascending sort order. If $D_i \leq thr1$, the block belongs to the smooth blocks and go to Step 4; or else if $thr1 < D_i \leq thr2$, the block belongs to the less complex blocks and go to Step 5; or else the block belongs to the highly complex blocks and go to Step 6.

Step 4. Extract the 16 bits S_i from S, and totally replace the bitmap BM_i with S_i to get the new bitmap BM'_i. Then, use Equations (7) to (10) to calculate two new quantization values L'_i and H'_i. Therefore, the final stego-compressed block code C'_i can be obtained by concatenating L'_i, H'_i, and BM'_i, i.e., (L'_i, H'_i, BM'_i).

Step 5. Extract the 6 bits S_i from S, and divide into $S_{i,1}$ and $S_{i,2}$ both with 3 bits of secret data. Then, divide BM_i into non-overlapping upper sub-block UBM_i and lower sub-block LBM_i, and extract the first seven bits of UBM_i and LBM_i to form two code-words UCW_i and LCW_i. Use Equation (4) to calculate which bits of UCW_i and LCW_i need to be flipped, respectively. Flip two corresponding bits of BM_i to obtain the new bitmap BM'_i, and the final stego-compressed block code C'_i can be obtained by concatenating L_i, H_i, and BM'_i, i.e., (L_i, H_i, BM'_i).

Step 6. Extract 4 bits of S_i from S, then divide into $S_{i,1}$ and $S_{i,2}$, both with 2 bits of secret data. Then, convert $S_{i,1}$ and $S_{i,2}$ into a quaternary system and use Equations (11) to (14) to calculate two new quantization values, L'_i and H'_i. Finally, concatenate L'_i, H'_i, and BM_i to obtain the final stego-compressed block code C'_i: (L'_i, H'_i, BM_i).

Step 7. Finish the algorithm when all of the final stego-compressed block codes are obtained, and concatenate each C'_i according its location in the original image to form the final stego-AMBTC compressed code C'. The stego-image

with secret data can also be obtained by AMBTC decoding.

3.3. Data Extraction Phase

In the data extraction phase, the secret data can be extracted follow the order of D_i from low to high when $D_i \leq thr2$, and from left to right and top to bottom of the image when $D_i > thr2$ without any loss. Details on the extraction algorithm are given below:

Input: Stego-AMBTC compressed code C', and two thresholds $thr1$, $thr2$.

Output: Secret data bit stream S.

Step 1. Collect all stego-compressed block code C'_i.

Step 2. Calculate D'_i by Equation (6), sort each stego-compressed block code according to the order for D'_i from low to high when $D'_i \leq thr2$ and the raster scan order of image for the rest of the blocks.

Step 3. Extract the stego-compressed block code C'_i of the i-th block according to the ascending sort order. If $D'_i \leq thr1$, the block belongs to the smooth blocks and go to Step 4; or else if $thr1 < D'_i \leq thr2$, the block belongs to the less complex blocks and go to Step 5; or else, go to Step 6.

Step 4. Extract the 16 bits of bitmap from the stego-compressed block code C'_i, and add these 16 bits directly into S.

Step 5. Divide BM'_i into non-overlapping upper sub-block UBM'_i and lower sub-block LBM'_i, and extract the first seven bits of UBM'_i and LBM'_i to form two code-words UCW'_i and LCW'_i, respectively. Use Equation (5) to calculate two parts of secret data $S_{i,1}$ and $S_{i,2}$ both with 3 bits. Then, add these 6 bits into S.

Step 6. Apply modulo-4 operation on its two quantization values and convert the results into the binary system, respectively. By doing so, 4 bits secret data can be obtained, then, add these 4 bits into S.

Step 7. Finish the algorithm when all of the stego-compressed block codes are processed, and the final secret data bit stream S can be obtained.

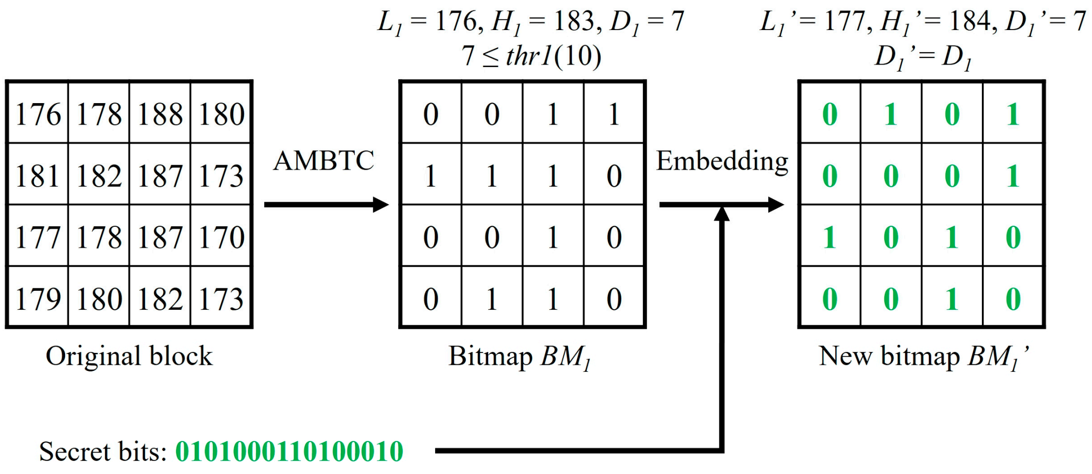
3.4. Example of Our Proposed Method

In this section, we will use three simple examples to illustrate the data embedding and data extraction phases of our proposed method. Three 4×4 sized blocks of the original image are assumed in the Figure 2a–c, i.e., (176, 178, 188, 180; 181, 182, 187, 173; 177, 178, 187, 170; 179, 180, 182, 173), (152, 143, 142, 127; 147, 140, 132, 126; 143, 141, 129, 123; 137, 135, 121, 114), and (111, 70, 44, 37; 108, 86, 52, 39; 116, 90, 55, 41; 120, 87, 50, 44). After being compressed by AMBTC, three compressed codes (176, 183, BM_1), (125, 142, BM_2), and (48, 103, BM_3) are obtained. We set $thr1$ = 10 and $thr2$ = 25.

For the first compressed code (176, 183, BM_1), the difference D_1 = 7 < $thr1$, so the block belongs to the smooth blocks, and BM_1 can be totally replaced by 16 bits of secret data, as shown in Figure 2a. Then, use Equations (7) to (10) to calculate two new quantization values L_1'= 177 and H_1'= 184. Even when the two quantization values are changed, the value of the difference between them is maintained.

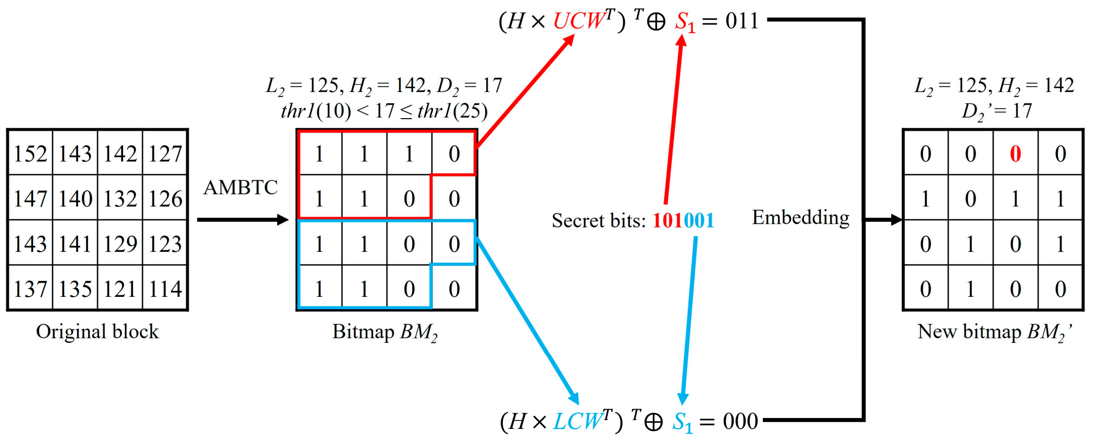
For the second compressed code (125, 142, BM_2), difference D_2 = 17, which is between $thr1$ and $thr2$, so the block belongs to the less complex blocks. The matrix coding method is used to embed 6 bits of secret data in BM_2. First, the secret bits are divided into $S_1 = 101$ and $S_2 = 001$. Then, use Equation (4) to separately calculate which bit needs to be flipped in the upper code-word and the lower code-word. As shown in Figure 2b, the third bit needs to be flipped in the upper code-word and the lower code-word is unchanged. Therefore, the secret bits '101001' can be embedded by just changing the third bit of BM_2 from '1' to '0', and the two quantization values are preserved.

For the third compressed code (48, 103, BM_3), difference D_3 = 55 > $thr2$, so the block belongs to the highly complex blocks. First, two secret data sequences '11' and '00' are converted into the quaternary system '3' and '0', respectively. Then, use Equations (11) to (14) to calculate two new quantization values $L_1' = 47$ and $H_1' = 104$. By doing so, 4 bits secret data can be embedded into this block. The difference between 47 and 104 is 59, which is higher than $thr2$, as expected. Therefore, misjudgments will not happen in the data extraction phase.



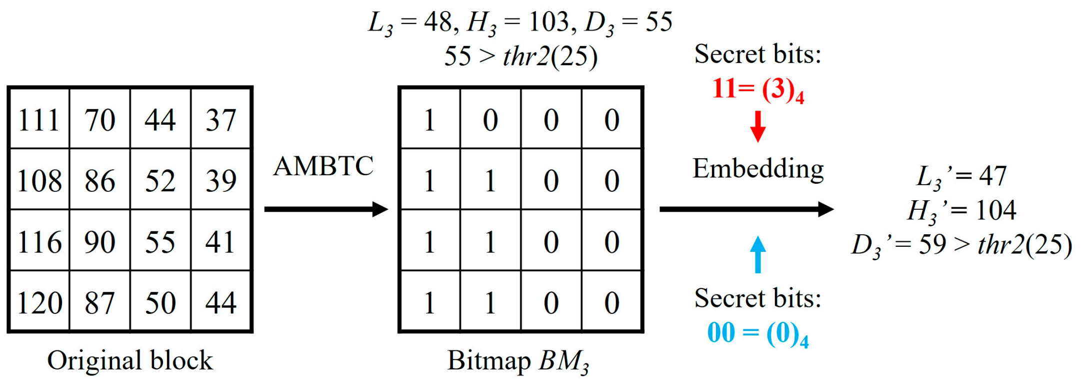
(**a**)

Figure 2. *Cont.*



$(H \times UCW^T)^T \oplus S_1 = 011$
$L_2 = 125, H_2 = 142, D_2 = 17$
$thr1(10) < 17 \le thr1(25)$
$L_2 = 125, H_2 = 142$
$D_2' = 17$
AMBTC
Secret bits: 101001
Embedding
Original block
Bitmap BM_2
New bitmap BM_2'
$(H \times LCW^T)^T \oplus S_1 = 000$

(**b**)



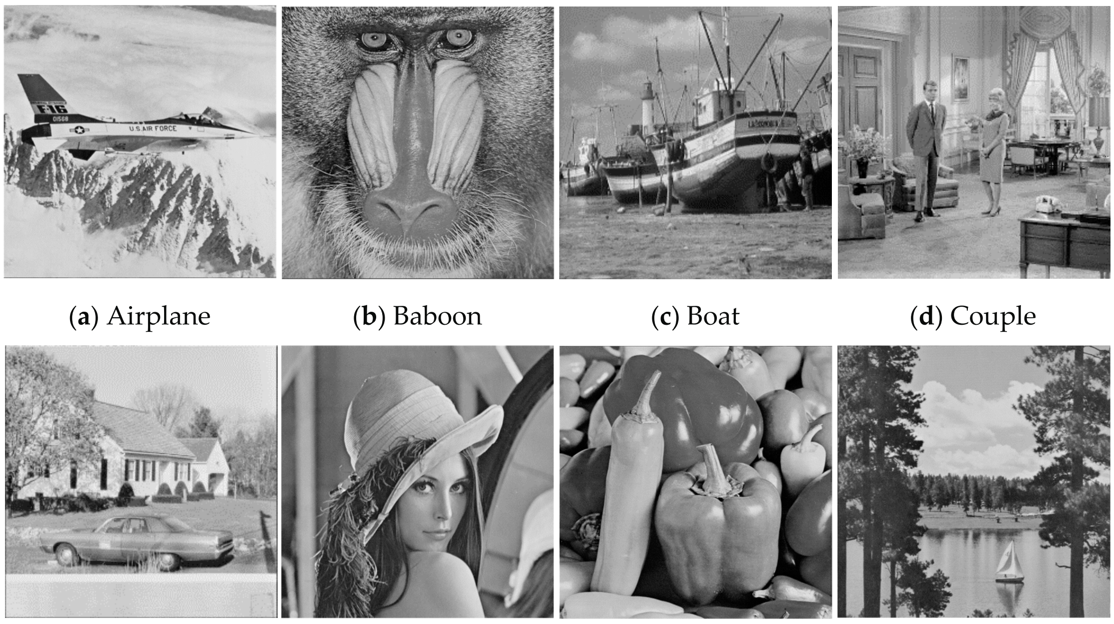
(**c**)

Figure 2. Examples of the data embedding phase of three type blocks: (**a**) a smooth block, (**b**) a less complex block, and (**c**) a highly complex block.

When extracting data, we also need to follow the order as mentioned above. For the first stego-compressed code (177, 184, BM_1'), $D_1' = 7 < thr1$, we just extract the 16 bits of bitmap '0101000110100010' to obtain the secret data. For the second stego-compressed code (125, 142, BM_2'), $D_2' = 17$, which is between *thr1* and *thr2*. The upper code-word and the lower code-word are used to calculate the secret data by Equation (5). After calculation, two parts of the secret data '101' and '001' can be obtained. For the third stego-compressed code (47, 104, BM_3), $D_3' = 59 > thr2$, we just apply modulo-4 operation on two quantization values and convert the results into a binary system to obtain two parts of secret data '11' and '00'.

4. Experimental Results and Discussion

This section presents the experimental data and compares the results of our proposed method with three relevant methods. Eight common 512 × 512 sized grayscale images, "Airplane", "Baboon", "Boat", "Couple", "House", "Lena", "Peppers", and "Sailboat" are used as test images as shown in Figure 3. In our experiments, the block size used for AMBTC compression of each method is set to 4 × 4. All the experiments were implemented by MATLAB R2017a, and run on a platform with an Intel (R) Core i5-8500 3.00 GHz processor, 8 GB RAM and Windows 10 operating system. The embedded secret data used in our experiments were generated by a pseudo-random number generator.



(**a**) Airplane (**b**) Baboon (**c**) Boat (**d**) Couple

(**e**) House (**f**) Lena (**g**) Peppers (**h**) Sailboat

Figure 3. Eight test images.

Hiding capacity and PSNR are used to quantitatively compare and evaluate our proposed method with other methods. Hiding capacity refers to the number of secret bits that can be embedded into the test images. Peak signal-to-noise ratio (PSNR) is used to estimate the visual quality of the stego-images compared to the original images. The definitions of mean square error (MSE) and PSNR are shown in Equations (15) and (16), respectively.

$$\mathrm{MSE} = \frac{1}{M \times N} \sum_{m=1}^{M} \sum_{n=1}^{N} (O_{m,n} - O'_{m,n})^2, \tag{15}$$

$$\mathrm{PSNR} = 10 \log_{10} \frac{(255)^2}{\mathrm{MSE}} \mathrm{dB}, \tag{16}$$

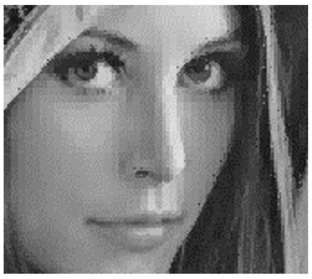
where (m, n) denotes the coordinate of each pixel.

Three relevant data hiding methods which embed data into AMBTC compressed images while preserving the final file size that were proposed by Ou and Sun [28], Chen and Chi [29], and Kumar et al. [30] are used to compare with our proposed method. For a fair comparison, two thresholds are set to the same values in the four methods. The visual comparison is shown in Figure 4. We can see that in (c), noticeable distortions appear on some edge regions such as the bridge of the nose and the eyebrows. In (d), there are some black points in the white blocks or some white points in the black blocks on the edge. The stego-images of our proposed method and Ou and Sun's method are closer to the original AMBTC compressed image.

(**a**) AMBTC

(**b**) Proposed

(**c**) Kumar et al. [30]


(**d**) Chen and Chi [29]

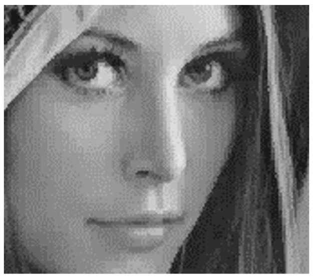
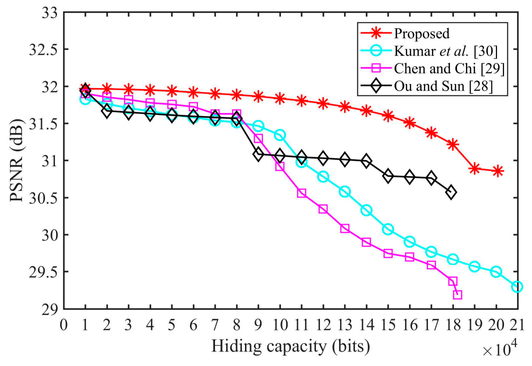
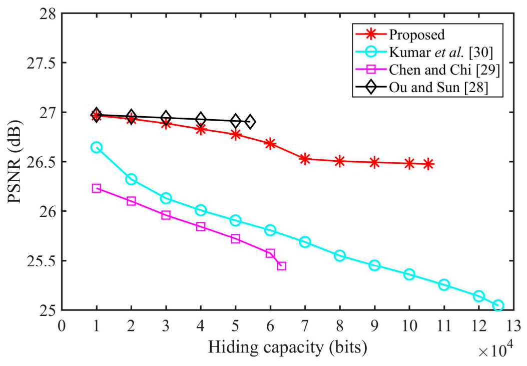

(**e**) Ou and Sun [28]

Figure 4. Visual comparison among different methods for the enlarged "Lena" image (block size $k = 4$, *thr1* = 10, *thr2* = 25, and the hiding capacity is 100,000).

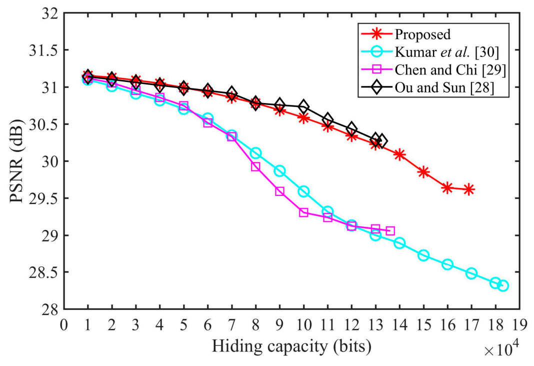
Then, we use PSNR to evaluate each method accurately. Figure 5 shows the PSNR for different amounts of the hiding data (from 1000 bits to the highest hiding capacity) of eight test images when *thr1* = 10 and *thr2* = 25.



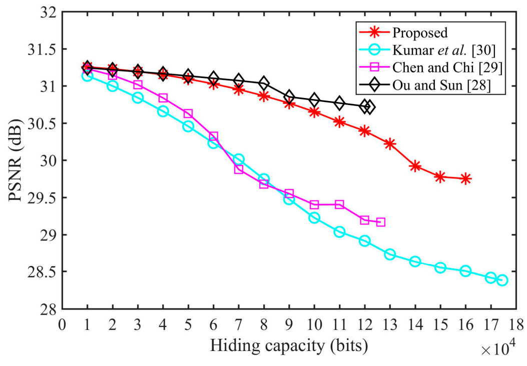
(**a**)



(**b**)



(**c**)



(**d**)

Figure 5. *Cont.*

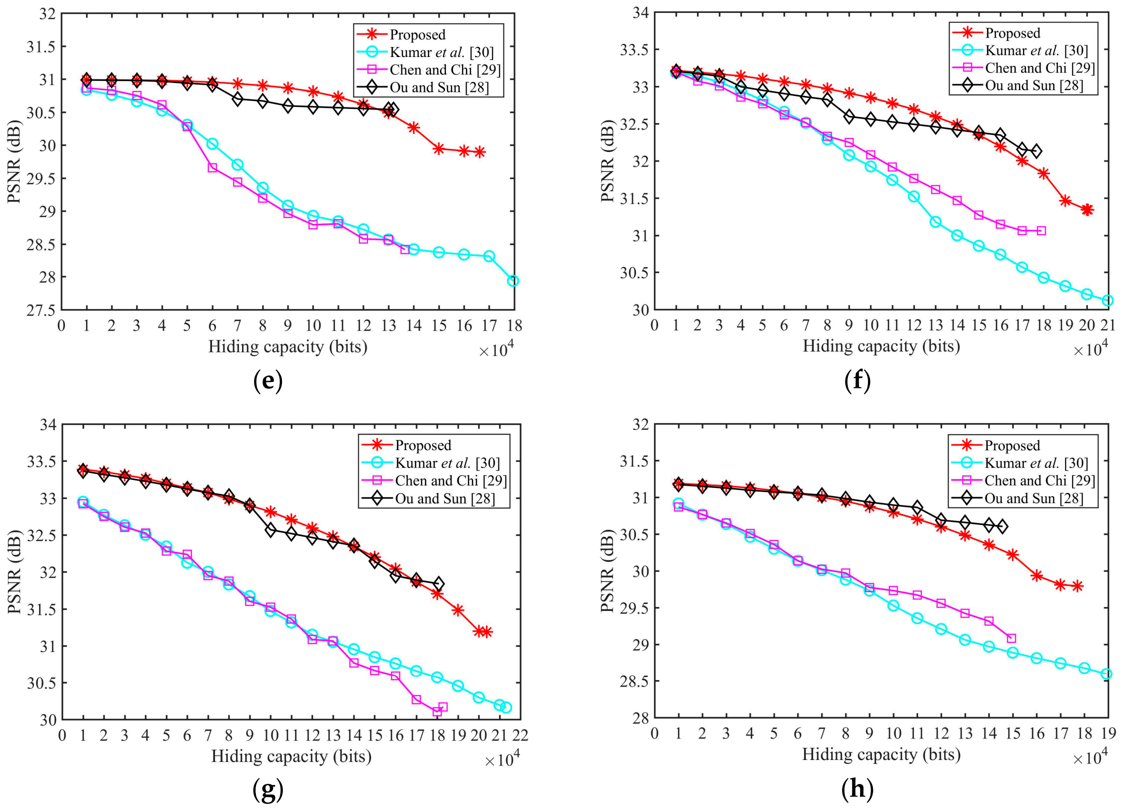


Figure 5. Comparison of peak signal-to-noise ratio (PSNR) among different methods for eight test images: (**a**) Airplane; (**b**) Baboon; (**c**) Boat; (**d**) Couple; (**e**) House; (**f**) Lena; (**g**) Peppers, and (**h**) Sailboat.

Kumar et al.'s method achieves the highest hiding capacity; however, the image quality is the worst. In the Ou and Sun method, the PSNR will not change when embedding bits into complex blocks. Therefore, the image quality of [28] is the best, but the hiding capacity is the lowest in most images. Our proposed method can achieve the second highest hiding capacity, i.e., only lower than [30]. In addition, our proposed method can achieve a relatively higher image quality. This is especially the case in smooth images such as Airplane, where the PSNR of our proposed method is the highest, and in Lena, when hiding capacity is lower, the PSNR of our proposed is also the highest. In Figure 5a, the PSNR of the other methods are rapidly declining when hiding capacity is higher than 8000 because the image becomes complex in the rear areas. Due to the hiding order of our proposed method, the PSNR of our proposed method will decline slowly with the increasing amount of hiding data. Therefore, our proposed method can achieve an excellent balance between image quality and hiding capacity, and can obtain high-fidelity images when the hiding amount is lower.

Table 2 shows the performance under different thresholds in eight test images. HC and ER denote hiding capacity and error ratio, respectively. The error ratio is the ratio of the error blocks to all blocks, and the error blocks are blocks where two quantization values are exchanged. Our proposed method can embed 6 bits and 4 bits into one less complex block and one highly complex block, with 2 bits and 1 bit less than [30], respectively. Thus, the gap in hiding capacity between our proposed method and [30] decreases when thresholds are enlarged, and the hiding capacity of our proposed method is always higher than the remaining two methods. The most important benefit of our proposed method is that there are no error blocks. The ER of the other three methods are all greater than 30%, and the ER increases with the decrease of thresholds, because the hiding strategy for smooth blocks in these three methods exchange the order of two quantization values. Method [28] has the highest ER in the most cases because of the embedding strategy in the complex block, thus, its security is the worst. Our proposed method can achieve a strict AMBTC format while the other methods cannot, and as such, security can be perfectly guaranteed in data transmission.

Table 2. Comparison of PSNR, hiding capacity (HC) and error ratio (ER) of the four methods under different thresholds in eight test images.

Methods	Metrics	Airplane	Baboon	Boat	Couple	House	Lena	Peppers	Sailboat
AMBTC	PSNR	31.9832	26.9748	31.1592	31.2793	30.9910	33.2053	33.4056	31.1794
thr1 = 10 and *thr2* = 25									
	PSNR	30.8573	26.4748	29.6169	29.7528	29.8977	31.3441	31.1888	29.7954
Proposed	HC	200837	105437	168901	159993	166187	200531	203817	177087
	ER	0	0	0	0	0	0	0	0
	PSNR	29.2947	25.0439	28.3124	28.3805	27.9374	30.1206	30.1617	28.5961
Kumar et al. [30]	HC	209719	125563	183149	174552	179473	209672	213103	189226
	ER	0.3622	0.3599	0.5022	0.4705	0.3884	0.4506	0.4697	0.4482
	PSNR	29.1864	25.4410	29.0581	29.1671	28.4183	31.0622	30.1701	29.0832
Chen and Chi [29]	HC	182178	63258	136126	126380	136503	178991	182888	149453
	ER	0.3859	0.4885	0.4503	0.4597	0.3994	0.4195	0.4352	0.4360
	PSNR	30.5722	26.9025	30.2695	30.7148	30.5373	32.1325	31.8410	30.6045
Ou and Sun [28]	HC	179179	54214	132664	122014	131929	176674	180829	145729
	ER	0.3820	0.4885	0.4515	0.4613	0.4042	0.4156	0.4345	0.4364
thr1 = 15 and *thr2* = 30									
	PSNR	30.3581	26.0870	28.8905	29.0280	29.2570	30.5953	30.4882	29.1793
Proposed	HC	213679	129215	196907	185321	184375	218979	224239	197479
	ER	0	0	0	0	0	0	0	0
	PSNR	28.9107	24.5719	27.9008	27.8522	27.3197	29.8154	29.87642	28.0991
Kumar et al. [30]	HC	220792	146730	206789	196410	195790	225366	230105	206875
	ER	0.3495	0.3568	0.4569	0.4464	0.3875	0.4202	0.4268	0.4235
	PSNR	29.2814	25.4026	28.9909	29.0897	28.6809	30.7737	30.5654	29.2090
Chen and Chi [29]	HC	199877	94409	175623	161156	160229	204727	212117	177217
	ER	0.3834	0.4888	0.4414	0.4460	0.3918	0.4119	0.4189	0.4269
	PSNR	31.4106	26.7542	30.3014	30.4907	30.4016	32.1786	31.8813	30.4846
Ou and Sun [28]	HC	197269	86494	172789	157594	156544	202894	210409	174199
	ER	0.3781	0.4836	0.4418	0.4460	0.3966	0.4132	0.4250	0.4335
thr1 = 20 and *thr2* = 35									
	PSNR	29.7978	25.6147	28.2041	28.2754	28.4987	29.8847	29.9105	28.5282
Proposed	HC	222565	146017	213361	201759	199551	229697	233449	210615
	ER	0	0	0	0	0	0	0	0
	PSNR	28.7812	24.4117	27.7682	27.6246	27.0495	29.5163	29.6846	27.8663
Kumar et al. [30]	HC	228449	161804	220686	210580	209023	234623	237888	218300
	ER	0.3455	0.3741	0.4269	0.4388	0.3848	0.4166	0.4125	0.4224
	PSNR	29.2806	25.3028	28.6718	28.8222	28.4855	30.7335	30.0696	28.9940
Chen and Chi [29]	HC	211774	115348	198396	183117	180081	219048	224892	194609
	ER	0.3755	0.4794	0.4340	0.4485	0.3933	0.4113	0.4244	0.4331
	PSNR	30.8572	26.5590	29.8558	30.0382	30.0897	31.5808	31.7296	30.1260
Ou and Sun [28]	HC	209524	108499	196039	180229	177124	217654	223444	192169
	ER	0.3767	0.4779	0.4461	0.4490	0.3959	0.4080	0.4337	0.4279

5. Conclusions

This paper presented a high-fidelity data hiding method for strict AMBTC format images by using a hybrid strategy for three block types: smooth blocks, less complex blocks and highly complex blocks. Our proposed method can embed 16 bits and 6 bits into a smooth block and a less complex block while maintaining the same difference between two quantization values, and 4 bits into a highly complex block, which guarantees that the difference will not decrease. Due to the designed hiding order, our method can obtain a high-fidelity image in the lower hiding amount. Experimental results show that our proposed method achieves an excellent balance between image quality and hiding capacity. In addition, our proposed method is able to strictly maintain the AMBTC compressed code format and as such, there are no error blocks in the final stego-compressed code.

Author Contributions: conceptualization, C.-C.C.; methodology, X.W. and J.-H.H.; software, X.W.; validation, C.-C.C., X.W. and J.-H.H.; formal analysis, C.-C.C. and J.-H.H.; investigation, X.W.; resources, X.W.; data curation, X.W.; writing—original draft preparation, X.W.; writing—review and editing, X.W. and J.-H.H.; visualization, X.W.; supervision, C.-C.C.; project administration, X.W. and J.-H.H.

Funding: This research received no external funding.

Acknowledgments: The authors would like to thank the reviewers for their valuable suggestions on improving the quality of this paper to this professional international journal.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Details of the proofs for Equations (9) and (10) are provided below. We first list the symbols that are used:

Table A1. The list of symbols.

Symbol	Meaning
k	Block size
B_0	Pixels belong to '0' in new bitmap
B_1	Pixels belong to '1' in new bitmap
D_i	The difference between two original quantization values
q'	The number '1' in a new bitmap
L_i' and H_i'	Two new quantization values
e	The square error of the block

The following processes are used to prove Equations (9) and (10), to obtain the best image quality when maintaining the same D_i.

Get

$$a_i = \frac{\sum_{p_j \in B_0'} p_j}{k \times k - q'},$$

and

$$b_i = \frac{\sum_{p_j \in B_1'} p_j}{q'}.$$

We have

$$\begin{aligned}
e_i &= \sum_{p_j \in B_0'} (p_j - L_i')^2 + \sum_{p_j \in B_1'} (p_j - H_i')^2 \\
&= \sum_{p_j \in B_0'} \left[\left(p_j - a_i\right) + (a_i - L_i')\right]^2 + \sum_{p_j \in B_1'} \left[\left(p_j - b_i\right) + (b_i - H_i')\right]^2 \\
&= \sum_{p_j \in B_0'} \left(p_j - a_i\right)^2 + 2\left(p_j - a_i\right)(a_i - L_i') + (a_i - L_i')^2 + \\
&\sum_{p_j \in B_1'} \left(p_j - b_i\right)^2 + 2\left(p_j - b_i\right)(b_i - H_i') + (b_i - H_i')^2 \\
&= \sum_{p_j \in B_0'} \left(p_j - a_i\right)^2 + \sum_{p_j \in B_0'} 2\left(p_j - a_i\right)\left(a_i - L_i'\right) + \sum_{p_j \in B_0'} (a_i - L_i')^2 + \\
&\sum_{p_j \in B_1'} \left(p_j - b_i\right)^2 + \sum_{p_j \in B_1'} 2\left(p_j - b_i\right)\left(b_i - H_i'\right) + \sum_{p_j \in B_1'} (b_i - H_i')^2 \\
&= \sum_{p_j \in B_0'} \left(p_j - a_i\right)^2 + \sum_{p_j \in B_0'} (a_i - L_i')^2 + \sum_{p_j \in B_1'} \left(p_j - b_i\right)^2 + \sum_{p_j \in B_1'} (b_i - H_i')^2 \\
&= \sum_{p_j \in B_0'} \left(p_j - a_i\right)^2 + \sum_{p_j \in B_0'} \left[a_i - \left(H_i' - D_i\right)\right]^2 + \sum_{p_j \in B_1'} \left(p_j - b_i\right)^2 + \sum_{p_j \in B_1'} (b_i - H_i')^2 \\
&= \sum_{p_j \in B_0'} \left(p_j - a_i\right)^2 + (k \times k - q')\left[a_i - \left(H_i' - D_i\right)\right]^2 + \sum_{p_j \in B_1'} \left(p_j - b_i\right)^2 + q'(b_i - H_i')^2
\end{aligned}$$

Since

$$\frac{de_i}{dH_i'} = -2(k \times k - q')[a_i - (H_i' - D_i)] - 2q'(b_i - H_i') = 0$$

then

$$2q'b_i + 2a_i(k \times k - q') + 2D_i(k \times k - q') = 2q'H_i' + 2H_i'(k \times k - q')$$

so

$$\begin{aligned} H_i' &= \frac{q'b_i + a_i(k \times k - q') + D_i(k \times k - q')}{k \times k} \\ &= b_i - \frac{b_i(k \times k - q') + a_i(k \times k - q') + D_i(k \times k - q')}{k \times k} \\ &= b_i + \frac{(k \times k - q')(D_i - b_i + a_i)}{k \times k} \end{aligned}$$

and

$$L_i' = a_i - \frac{q'(D_i - b_i + a_i)}{k \times k}$$

References

1. Bender, W.; Gruhl, D.; Morimoto, N.; Lu, A. Techniques for data hiding. *IBM Syst. J.* **1996**, *35*, 313–336. [CrossRef]
2. Wu, M.; Liu, B. Data hiding in binary image for authentication and annotation. *IEEE Trans. Multimed.* **2004**, *6*, 528–538. [CrossRef]
3. Tian, J. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 890–896. [CrossRef]
4. Tsai, P.; Hu, Y.C.; Yeh, H.-L. Reversible image hiding scheme using predictive coding and histogram shifting. *Signal Process.* **2009**, *89*, 1129–1143. [CrossRef]
5. Chang, C.-C.; Lu, T.C. A difference expansion oriented data hiding scheme for restoring the original host images. *J. Syst. Softw.* **2006**, *9*, 1754–1766. [CrossRef]
6. Lee, C.F.; Chen, H.L.; Tso, H.K. Embedding capacity raising in reversible data hiding based on prediction of difference expansion. *J. Syst. Softw.* **2010**, *83*, 1864–1872. [CrossRef]
7. Hu, Y.; Lee, H.K.; Li, J. DE-based reversible data hiding with improved overflow location map. *IEEE Trans. Circuits Syst. Video Technol.* **2008**, *19*, 250–260.
8. Tai, W.L.; Yeh, C.M.; Chang, C.C. Reversible data hiding based on histogram modification of pixel differences. *IEEE Trans. Circuits Syst. Video Technol.* **2009**, *19*, 906–910.
9. Xuan, G.; Yao, Q.; Yang, C.; Gao, J.; Chai, P.; Shi, Y.Q.; Ni, Z. Lossless data hiding using histogram shifting method based on integer wavelets. In *International Workshop on Digital Watermarking 2006*; Lecture Notes in Computer Science; Jeon, B., Ed.; Springer: Berlin, Germany, 2006; pp. 323–332.
10. Chen, W.Y. Color image steganography scheme using DFT, SPIHT codec, and modified differential phase-shift keying techniques. *Appl. Math. Comput.* **2008**, *196*, 40–54. [CrossRef]
11. Chang, C.C.; Lin, C.C.; Tseng, C.S.; Tai, W.-L. Reversible hiding in DCT-based compressed images. *Inf. Sci.* **2007**, *177*, 2768–2786. [CrossRef]
12. Zhang, X. Reversible data hiding in encrypted image. *IEEE Signal Process. Lett.* **2011**, *18*, 255–258. [CrossRef]
13. Wang, J.X.; Lu, Z.M. A path optional lossless data hiding scheme based on VQ joint neighboring coding. *Inf. Sci.* **2009**, *179*, 3332–3348. [CrossRef]
14. Qin, C.; Chang, C.C.; Chiu, Y.P. A novel joint data-hiding and compression scheme based on SMVQ and image inpainting. *IEEE Trans. Image Process.* **2013**, *23*, 969–978.
15. Tseng, H.W.; Chang, C.C. High capacity data hiding in JPEG-compressed images. *Informatica* **2004**, *15*, 127–142.
16. Delp, E.; Mitchell, O. Image compression using block truncation coding. *IEEE Trans. Commun.* **1979**, *27*, 1335–1342. [CrossRef]
17. Lema, M.; Mitchell, O. Absolute moment block truncation coding and its application to color images. *IEEE Trans. Commun.* **1984**, *32*, 1148–1157. [CrossRef]

18. Lin, C.C.; Liu, X.L.; Tai, W.L.; Yuan, S.M. A novel reversible data hiding scheme based on AMBTC compression technique. *Multimed. Tools Appl.* **2015**, *74*, 3823–3842. [CrossRef]
19. Kim, C.; Shin, D.; Leng, L.; Yang, C.N. Lossless data hiding for absolute moment block truncation coding using histogram modification. *J. Real-Time Image Process.* **2018**, *14*, 101–114. [CrossRef]
20. Chen, Y.Y.; Hsia, C.H.; Jhong, S.Y.; Lin, H.J. Data hiding method for AMBTC compressed images. *J. Ambient Intell. Hum. Comput.* **2018**, 1–9. [CrossRef]
21. Malik, A.; Sikka, G.; Verma, H.K. An AMBTC compression based data hiding scheme using pixel value adjusting strategy. *Multidimens. Syst. Signal Process.* **2018**, *29*, 1801–1818. [CrossRef]
22. Huynh, N.-T.; Bharanitharan, K.; Chang, C.C.; Liu, Y. Minima-maxima preserving data hiding algorithm for absolute moment block truncation coding compressed images. *Multimed. Tools Appl.* **2018**, *77*, 5767–5783. [CrossRef]
23. Sun, W.; Lu, Z.M.; Wen, Y.C.; Yu, F.X.; Shen, R.J. High performance reversible data hiding for block truncation coding compressed images. *Signal Image Video Process.* **2013**, *7*, 297–306. [CrossRef]
24. Hong, W.; Ma, Y.B.; Wu, H.C.; Chen, T.S. An efficient reversible data hiding method for AMBTC compressed images. *Multimed. Tools Appl.* **2017**, *76*, 5441–5460. [CrossRef]
25. Hong, W.; Zhou, X.; Weng, S. Joint adaptive coding and reversible data hiding for AMBTC compressed images. *Symmetry* **2018**, *10*, 254. [CrossRef]
26. Chuang, J.C.; Chang, C.C. Using A Simple and Fast Image Compression Algorithm To Hide Secret Information. *Int. J. Comput. Appl.* **2006**, *28*, 329–333.
27. Hong, W.; Chen, T.S.; Shiu, C.W. Lossless steganography for AMBTC-compressed images. *2008 Congr. Image Signal Process.* **2018**, *2*, 3–17.
28. Ou, D.; Sun, W. High payload image steganography with minimum distortion based on absolute moment block truncation coding. *Multimed. Tools Appl.* **2015**, *74*, 9117–9139. [CrossRef]
29. Chen, Y.Y.; Chi, K.Y. Cloud image watermarking: High quality data hiding and blind decoding scheme based on block truncation coding. *Multimed. Syst.* **2017**, 1–13. [CrossRef]
30. Kumar, R.; Kim, D.S.; Jung, K.H. Enhanced AMBTC based data hiding method using hamming distance and pixel value differencing. *J. Inf. Secur. Appl.* **2019**, *47*, 94–103. [CrossRef]
31. Westfeld, A. F5—A steganographic algorithm. *Inf. Hiding* **2001**, *2137*, 289–302.