


Article

# Reversible Steganographic Scheme for AMBTC-Compressed Image Based on (7,4) Hamming Code

Juan Lin <sup>1,2</sup>, Chia-Chen Lin <sup>3,\*</sup>  and Chin-Chen Chang <sup>4,\*</sup>

<sup>1</sup> School of Electronic and Information Engineering, Fuqing Branch of Fujian Normal University, Fuzhou 350300, China; lj2020229@gmail.com

<sup>2</sup> Engineering Research Center for ICH Digitalization and Multi-source Information Fusion, (Fuqing Branch of Fujian Normal University), Fujian Province University, Fuzhou 350300, China

<sup>3</sup> Department of Computer Science and Information Management, Providence University, Taichung 43301, Taiwan

<sup>4</sup> Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan

\* Correspondence: mhlin3@pu.edu.tw (C.-C.L.); alan3c@gmail.com (C.-C.C.)

Received: 22 August 2019; Accepted: 20 September 2019; Published: 3 October 2019



**Abstract:** In recent years, compression steganography technology has attracted the attention of many scholars. Among all image compression method, absolute moment block truncation coding (AMBTC) is a simple and effective compression method. Most AMBTC-based reversible data hiding (RDH) schemes do not guarantee that the stego AMBTC compression codes can be translated by the conventional AMBTC decoder. In other words, they do not belong to Type I AMBTC-based RDH scheme and easily attract malicious users' attention. To solve this problem and enhance the hiding capacity, we used (7,4) hamming code to design a Type I AMBTC-based RDH scheme in this paper. To provide the reversibility feature, we designed a prediction method and judgement mechanism to successfully select the embeddable blocks during the data embedding phase and data extraction and recovery phase. In comparing our approach with other BTC-based schemes, it is confirmed that our hiding capacity is increased while maintaining the limited size of the compression codes and acceptable image quality of the stego AMBTC-compressed images.

**Keywords:** reversible data hiding; AMBTC; (7,4) hamming code

## 1. Introduction

Data hiding, also called steganography [1], is the study of embedding secret message into innocuous cover media [2], such as images, audio signals, and videos files, to protect the confidentiality of the hidden data. Digital images are often used as the cover media because they can be accessed easily and can provide abundant redundancies for concealing data. According to the recoverability of the cover image after the extraction of hidden data, data hiding can be classified into two categories, i.e., conventional data hiding and reversible data hiding (RDH). RDH is crucial for some specific applications, such as medical imaging, remote sensing, and military communications. The first RDH was proposed by Barton in 1997 [3]. In Barton's approach, the bits to be overlaid were compressed in advance and then added to the secret bit string, after which the generated secret bit string was embedded into the data blocks of the images. Since then, researchers inspired by Barton's idea have proposed various RDH approaches, including difference expansion (DE) [4,5], histogram shifting [6–8], and prediction-error [9–11]. In 2003, DE was introduced by Tian et al. by embedding the secret message into the differences between two adjacent pixels [4]. In 2006, Ni et al. developed a different RDH

approach based on histogram modification. Their approach used the zero or the minimum pixel value of the histogram of an image as the peak value, and then they slightly modified peak value and shifted the related pixel values to embed secret data [6]. In 2008, Lin et al. transformed the original image into a difference image before using the histogram shifting technique to make sure the frequency of peak value is as largest as possible and ascertained that the hiding capacity was increased significantly [7]. In 2011, Li et al. designed a new hybrid RDH method based on prediction-error expansion (PEE), adaptive embedding, and pixels' selection [9]. Subsequently, many scholars used Li et al.'s approach and designed various improved RDH schemes based on PEE. However, these schemes were designed for images in the spatial domain rather than images in the compression domain, but the latter demand of compression techniques has increased gradually as people have become familiar with the capability of transmitting or sharing image files over the Internet. Therefore, it is essential to explore and design RDHs for compression-domain images.

Compression techniques are designed to solve the problem of limited bandwidth since multimedia files, such as image, audio, and video files, are quite large. The general compression techniques include discrete cosine transform (DCT) [12], discrete wavelet transform (DWT) [13], vector quantization (VQ) [14], and side match vector quantization (SMVQ) [15]. The latter two famous block-based image compression techniques, i.e., VQ and SMVQ, attracted the attention of researchers when they tried to design RDHs for images in the compression domain. In 2005, Yang et al. [16] applied modified, fast-correlation vector quantization (MFCVQ) to design a reversible watermarking scheme for VQ-compressed images. However, the weakness of their scheme is their low hiding capacity. To overcome this disadvantage, Lu et al. [17] proposed an RDH method based on VQ-index residual value coding technique. In 2006, Chang et al. [18] used a real codeword that could be mapped to an index in the SMVQ's codebook and a derived codeword that did not exist in the SMVQ's codebook to design their RDH strategy for SMVQ-compressed images. Inspired by Chang et al.'s idea, in 2011, Chiou et al. [19] proposed an RDH scheme with enhanced hiding capacity that was also based on SMVQ to improve the hiding capacity and visual quality of stego-image. In Chiou et al.'s scheme, one secret bit is concealed into each pixel of the residual blocks.

In addition to VQ and SMVQ, block truncation coding (BTC) is the other efficient, lossy, block-based image compression technique. BTC technique was first proposed by Delp and Mitchell in 1979 [20] and has been used for compressing Mars Pathfinder's rover images [21]. Later, Lema and Mitchell [22] proposed the absolute mean block truncation coding (AMBTC) method to prompt the compression performance. After learning about the features of BTC and AMBTC, many scholars designed BTC-based or AMBTC-based RDHs. In general, these RDHs can be classified into two types, i.e., (1) Type I in which the stego codes can be decoded correctly via a conventional BTC or AMBTC decoder [23–25] and a recovered image always can be constructed and (2) Type II in which the stego codes have specific coding structures [26–28] and cannot be translated by standard BTC or AMBTC decoders. In other words, as long as malicious attackers use BTC or AMBTC decoders to decode the stego codes, those that cannot be decoded will be identified easily because they carry some valuable information. To avoid drawing the attention of attackers, in this paper, we tried to design a Type I-RDH scheme for AMBTC-compressed images.

Inspired by Chang et al.'s high payload data hiding scheme for compressed images with (7,4) hamming code [29], several data hiding schemes based on (7,4) hamming code were proposed sequentially [30,31], but they were not reversible. In 2017, Biswapati et al. utilized (7,4) hamming code to design a partial RDH which allows the restored image to be very similar to its original image [32]. In 2018, the same research team designed a dual image-based RDH scheme using (7,4) hamming code [33]. According to the lectures we have collected to date, no one has used (7,4) hamming code to design RDH for a single, AMBTC-compressed image. However, (7,4) hamming code has a unique feature in that it can hide three secret bits into a seven-bit stream only at the cost of modifying a single bit. To take advantage of (7,4) hamming code, in this paper we introduce an RDH scheme using (7,4) hamming code that embeds secret data based on the relationship of the high-low mean tables and also

into the bitmaps. In other words, the original AMBTC compression codes can be completely restore once the hidden data has been extracted with our proposed scheme. There is no distortion between our restored compression codes and the original AMBTC compression codes.

The rest of the paper is organized as follows. Section 2 introduces the AMBTC scheme and (7,4) hamming code. Section 3 illustrates the use of the proposed scheme for embedding and extracting data using the high-low mean tables and bitmap. In Section 4, we describe the experiments we conducted and compare our proposed RDH scheme with other Type I AMBTC-based RDH schemes. Concluding remarks are given in Section 5.

## 2. Related Works

In this section, we briefly describe the concepts of AMBTC and (7,4) hamming code and matrix embedding.

### 2.1. Absolute Moment Block Truncation Coding (AMBTC)

In 1984, Lema and Mitchell [22] proposed a new compression technique for reconstructing good images that preserved the local characteristics of the spatial blocks of the image. After dividing an image into  $4 \times 4$  non-overlapping blocks, for each block, their method computes the mean and the standard deviation of the sample, i.e.,

$$\eta = \frac{1}{m} \sum_{i=1}^m x_i \text{ and } \sigma = \frac{1}{m} \sum_{i=1}^m |x_i - \eta| \quad (1)$$

where  $m$  is the total number of pixels in the block, and  $x_i$  is the gray value of each pixel. Both values are transmitted along with a bit plane that contains ones in those positions where  $x_i > \eta$  and zeros otherwise. At the recipient, a reconstructed block can be obtained with two quantizers: the low mean value  $L$  for group-0 and the high mean value  $H$  for group-1 that preserves the sample mean and variance [22] according to Equation (2),

$$L = \eta - \frac{m \times \sigma}{2(m - q)} \text{ and } H = \eta + \frac{m \times \sigma}{2q} \quad (2)$$

where  $q$  is the number of pixels that are greater than or equal to  $\eta$ .

Figure 1 shows an example of the results of encoding and decoding using the AMBTC compression method. Note that, in the original AMBTC encoding phase, an image is generally divided into non-overlapping blocks, and the size of each block is  $4 \times 7$  pixels. However, Figure 1a shows that the original image block was  $4 \times 7$ . This is because a block defined in our proposed RDH scheme is  $4 \times 7$  pixels. To give readers consistent representation and a better understanding, a block with the same size is demonstrated in Figure 1a. Even though the size of the block is different from the conventional AMBTC, the other operations in the encoding and decoding phases are the same as they are in the conventional AMBTC. The mean value was obtained, i.e.,  $\eta = 161.28$ . We computed the value of  $L = 159$  by averaging the pixel values that were less than 161.28, and the result is rounded to the nearest integer. Similarly, we computed the value. Subsequently, if the pixel values were less than 161.28, their corresponding bits were set as "0" in the bitmap; otherwise, their corresponding bits were set as "1." Figure 1b shows the corresponding AMBTC bitmap that was derived from a given block. As shown in Figure 1a, its AMBTC compressed trio is denoted as  $(L = 159, H = 162, \text{bitmap} = 1110101; 1110101; 1110101; 1110101)$ . To decode the compressed trio, the "0" and "1" in the bitmap were replaced by  $L = 159$  and  $H = 162$ , respectively. Finally, a reconstructed image block was generated, as shown in Figure 1c.

(L = 159, H = 162)

162	162	162	161	162	157	163
162	162	162	161	162	157	163
162	162	162	161	162	157	163
162	162	162	161	162	157	163

1	1	1	0	1	0	1
1	1	1	0	1	0	1
1	1	1	0	1	0	1
1	1	1	0	1	0	1

162	162	162	159	162	159	162
162	162	162	159	162	159	162
162	162	162	159	162	159	162
162	162	162	159	162	159	162

(a) Original image block    (b) AMBTC bitmap    (c) Reconstructed image block

**Figure 1.** Example of the absolute moment block truncation coding (AMBTC) compression method: (a) Original image block; (b) AMBTC bitmap; (c) Reconstructed image block.

2.2. (7,4) Hamming Code

(7,4) hamming code is a kind of linear code for the correction of block errors. It has been used extensively by researchers because it can identify a single bit error in a block of 7-bits and then correct it. In this paper, (7,4) hamming coding was used to operate the modification of bits. Four data bits, i.e.,  $d = (d_1, d_2, d_3, d_4)$ ,  $d$  were encoded into seven bits by adding three parity bits, i.e.,  $c = (c_1, c_2, c_3)$ . The encoding rule must satisfy Equation (3).

$$\begin{cases} c_1 = d_1 \oplus d_2 \oplus d_3, \\ c_2 = d_1 \oplus d_3 \oplus d_4, \\ c_3 = d_2 \oplus d_3 \oplus d_4, \end{cases} \tag{3}$$

where  $\oplus$  is the exclusive-OR operation. By transforming Equation (3) into a matrix, the result is transposed, i.e.,

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} \Leftrightarrow \begin{bmatrix} d_1 & d_2 & d_3 & d_4 \end{bmatrix} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix}. \tag{4}$$

when the unit identity is added, the form is changed as shown below:

$$\begin{bmatrix} d_1 & d_2 & d_3 & d_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} d_1 & d_2 & d_3 & d_4 & d_5 & d_6 & d_7 \end{bmatrix}, \tag{5}$$

$$G = \begin{bmatrix} I_k & Q_{k \times r} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}, \tag{6}$$

For decoding, a parity check matrix,  $H_0$ , is required. This matrix can be obtained from Equation (7) by changing Equation (3) to Equation (8).

$$\begin{cases} c_1 \oplus d_1 \oplus d_2 \oplus d_3 = 0, \\ c_2 \oplus d_1 \oplus d_3 \oplus d_4 = 0, \\ c_3 \oplus d_2 \oplus d_3 \oplus d_4 = 0. \end{cases} \tag{7}$$

Then, convert it into a matrix.

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \quad (8)$$

Finally, a parity check matrix  $H_0 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}_{3 \times 7}$  is generated.

The four codes and three parity bits are sent to receiver. At the recipient, the received 7-bits codeword,  $R = (1000110)$ , is multiplied by the parity check matrix,  $H_0$ , and, then, modulo 2 is conducted to determine whether an error occurred. The computed result is called a syndrome. If the syndrome is "000," it means there is no error. If a single bit error occurs, the syndrome will not be equal to "000." Assume the received codeword  $R$ , has one error in the first row of the  $G$  matrix, (ex.  $R = (1100110)$ ), the calculated syndrome is "101," which is identical to the second column of  $H_0$ , and  $R$  is corrected by  $R = (1100110) \oplus e_2 = (1000110)$  where  $e_i$  is the  $i^{\text{th}}$  unit vector of length seven ( $e_2$  is a zero vector of length seven with a 1 located at the second position,  $e_2 = (0100000)$ ). Then, we determine the correct original data bits by ignoring the last three bits, i.e.,  $d = (1000)$ .

### 3. Proposed Scheme

In this paper, we propose a (7,4) hamming code-based RDH for AMBTC-compressed images. Our proposed RDH scheme consists of two phases, i.e., 1) the data embedding phase and 2) the data extraction and recovery phase. In the data embedding phase, after AMBTC encoding, the cover image is divided into non-overlapping  $4 \times 7$ -sized blocks, where each block contains only different values of high mean (H), low mean (L), and  $4 \times 7$ -bits bitmap. The secret bits can be concealed by changing the order between  $H$  and  $L$ , and they also can be embedded into the  $4 \times 7$ -bits bitmap with (7,4) hamming code. In the former, we can extract a secret bit and losslessly reconstruct the stego image to the AMBTC-compressed image. However, for embedding data on a bitmap with (7,4) hamming code, three secret bits can be hidden into each  $1 \times 7$ -sized bitmap. To ensure that the modified bit of  $1 \times 7$ -bits bitmap can be restored, a prediction method and a judgment mechanism which is workable during both the data embedding phase and data extraction/recovery phase are designed in our proposed RDH scheme. The proposed prediction method and judgment mechanism are our core concept and they are introduced initially in Section 3.1, and the data embedding phase and data extraction/recovery phases are presented in detail in Sections 3.2 and 3.3, respectively. To give clear explanations for both phases, an example is provided to demonstrate our proposed data embedding and data extraction/recovery operations.

#### 3.1. Prediction Method and Selections of Embeddable Blocks

To embed secret data into two quantizers, the prediction method must be used, and it must be derived from the original neighboring pixels. Here, a simple prediction is conducted, and, first, a  $3 \times 3$ -pixels window is constructed, as shown in Figure 2.

$X_1$	$X_2$	$X_3$
	$Y_i$	
$X_4$	$X_5$	$X_6$

**Figure 2.** Reconstructed  $Y_i$  and its neighboring six reconstructed pixels.

To predict  $Y_i$ , we define its prediction value,  $Y_i'$ , can be derived from its neighboring six pixels as

$$Y_i' = \lambda_1 X_1 + \lambda_2 X_2 + \lambda_3 X_3 + \lambda_4 X_4 + \lambda_5 X_5 + \lambda_6 X_6. \quad (9)$$

where  $Y_i'$  are six neighboring pixels of  $Y_i$  in the image, and  $\lambda_1, \lambda_2, \dots, \lambda_6$  are the coefficients of  $X_i (i = 1, 2, \dots, 6)$ , respectively. The expression  $\lambda_1 + \lambda_2 + \dots + \lambda_6 = 1$  must be satisfied. If the distance between  $Y_i$  and each pixel,  $X_i (i = 1, 2, \dots, 6)$ , is not considered,  $\lambda_i (i = 1, 2, \dots, 6)$  is always  $1/6$ . However, if the Euclidean distance between  $Y_i$  and  $X_i (i = 1, 2, \dots, 6)$  is considered, their corresponding coefficients can be defined as:  $\lambda_1 : \lambda_2 = 1 : \sqrt{2}$ ,  $\lambda_2 = \lambda_5$ ,  $\lambda_1 = \lambda_3 = \lambda_4 = \lambda_6$ , and  $\lambda_1 + \lambda_2 + \dots + \lambda_6 = 1$ . Thus, six coefficients are obtained as  $\lambda_1 = \lambda_3 = \lambda_4 = \lambda_6 = 0.1465$ ,  $\lambda_2 = \lambda_5 = 0.207$ , as shown in Figure 3.

$X_1$ (0.1465)	$X_2$ (0.207)	$X_3$ (0.1465)
	$Y_i$	
$X_4$ (0.1465)	$X_5$ (0.207)	$X_6$ (0.1465)

**Figure 3.** Coefficients of the six neighboring pixels of  $Y_i$ .

For the pixels that are located in the left border of an image, we define that the prediction value,  $Y_i'$ , only requires its four neighboring pixels, i.e.,

$$Y_i' = \lambda_1 X_1 + \lambda_2 X_2 + \lambda_4 X_4 + \lambda_5 X_5. \quad (10)$$

where  $\lambda_1 = \lambda_4$ ,  $\lambda_2 = \lambda_5$ ,  $\lambda_1 : \lambda_2 = 1 : \sqrt{2}$  and  $\lambda_1 + \lambda_2 + \lambda_4 + \lambda_5 = 1$ . Thus, the four coefficients can be computed as  $\lambda_1 = \lambda_4 = 0.20$ ,  $\lambda_2 = \lambda_5 = 0.29$ . In a similar manner, the prediction pixels  $Y_i'$  located at the right border of an image can be derived from

$$Y_i' = \lambda_2 X_2 + \lambda_3 X_3 + \lambda_5 X_5 + \lambda_6 X_6. \quad (11)$$

where the coefficients are defined as  $\lambda_2 = \lambda_5 = 0.29$ ,  $\lambda_3 = \lambda_6 = 0.20$ .

Subsequently, given a constructed image, the embeddable blocks can be determined by Equation (12) to maintain the reversible feature based on the difference between the reconstructed pixels and their prediction values

$$|Y_i' - Y_i| < 0.5 \times (H - L) \quad (H \neq L). \quad (12)$$

Note that the block is defined as a  $1 \times 7$ -sized block instead of a  $4 \times 7$ -sized block when determining whether or not it is embeddable. Since it is a reconstructed image, each  $1 \times 7$ -sized block contains, at most, only two different values, i.e., either  $L$  or  $H$ . It must be determined whether the blocks located in the even rows are embeddable or not. Blocks located in the odd rows are treated as reference blocks to

ensure that all of the embedded blocks can be completely restored to the original AMBTC reconstructed blocks after the hidden data are extracted. For a  $1 \times 7$ -sized block located in an even row, Equation (12) is used to check seven reconstructed pixels. If Equation (12) is satisfied for all seven reconstructed pixels, the currently processing block is embeddable. If any pixel in a  $1 \times 7$ -sized block does not satisfy Equation (12), the currently processing block is un-embeddable.

Our proposed prediction method can be used to assist the selection of embeddable blocks irrespective of whether the process is in the data embedding phase or the data extraction and recovery phase. Note that the original neighboring pixels are considered to compute the prediction values during the data embedding phase. By contrast, the reconstructed pixels are considered during the data extraction and recovery phase. However, even in the data extraction phase, Equation (12) always holds because we assume that, if  $Y_i$  is  $H$ , then  $Y_i^T$  is  $L$ , and vice versa.

$$H - L = |Y_i^T - Y_i| = |Y_i^T - Y_i - Y_i' + Y_i'|.$$

Therefore, from

$$160 \times 0.146 + 160 \times 0.207 + 155 \times 0.146 + 157 \times 0.146 + 157 \times 0.207 + 153 \times 0.146 = 157.18$$

and according to Equation (12), Equation (13) can be derived as

$$|Y_i^T - Y_i'| > 0.5 \times (H - L) \quad (H \neq L) . \quad (13)$$

Since the  $1 \times 7$ -sized blocks located in the odd rows are treated as reference blocks and since not all  $1 \times 7$ -sized blocks located in the even rows are embeddable for a given AMBTC reconstructed image, a location map  $LP$  recording is required for the blocks located in the even rows. In the location map  $LP$ , "1" indicates that the corresponding block is embeddable; and "0" indicates that the corresponding block is unembeddable. Then, the location map  $LP$  is treated as secret data, and it is embedded into the bitmap in front of the secret message.

#### Example of the Selection of Embeddable Blocks

Let us assume that the size of a grayscale image is  $512 \times 512$  pixels and that it is divided into non-overlapping  $4 \times 7$ -sized blocks. Here, we only take the  $5 \times 7$  reconstructed pixels for example. It is noted the first four rows map to a  $4 \times 7$ -sized block. Therefore, the first four rows only contain two different values, which map to the same quantizer pair, which is  $L (= 155)$  and  $H (= 160)$ . However, the last row maps to the other  $4 \times 7$ -sized block, and the two values are different, i.e.,  $L (= 153)$  and  $H (= 157)$ .

Equation (12) is satisfied for all pixels located in the second row. Therefore, it is noted as "0" to indicate that the second row is embeddable in the location map. In contrast, the sixth prediction value located in the fourth row is derived by Equation (9) as

$$160 \times 0.146 + 160 \times 0.207 + 155 \times 0.146 + 157 \times 0.146 + 157 \times 0.207 + 153 \times 0.146 = 157.18$$

The sixth predicted value located in the fourth row does not satisfy Equation (12).

$$|157.18 - 160| = 2.82 > 1/2 \times |H - L| = 1/2 \times (160 - 155) = 2.5.$$

Therefore, it is marked in red as shown in Figure 4b. Since the fourth row has one pixel that does not satisfy Equation (12), it is noted as "0" in the location map to indicate that the fourth row is unembeddable.

160	155	155	160	160	160	155
160	155	155	155	160	160	155
160	155	155	155	160	160	155
160	155	155	155	160	160	155
157	153	157	157	157	157	153

(a) Example of 5 × 7 reconstructed pixels of Figure 1(a)

	160	155	155	160	160	160	155	
Embeddable block →	159.41	156.46	155	156.46	158.53	158.53	156.75	→ '1'
	160	155	155	155	160	160	155	
Unembeddable block →	158.20	155.90	155.41	156.73	157.76	157.18	155.61	→ '0'
	157	153	157	157	157	157	153	

$$160 \times 0.146 + 160 \times 0.207 + 155 \times 0.146 + 157 \times 0.146 + 157 \times 0.207 + 153 \times 0.146 = 157.18$$

$$|157.18 - 160| = 2.82 > 1/2 \times |H - L| = 1/2 \times (160 - 155) = 2.5$$

(b) The sixth prediction value located in the fourth row is derived as by Equation (9). The sixth pixel located in the fourth row does not satisfy with Equation (12).

**Figure 4.** Example of the selection of embeddable blocks: (a) Example of 5 × 7 reconstructed pixels of Figure 1a; (b) The sixth prediction value located in the fourth row is derived as by Equation (9). The sixth pixel located in the fourth row does not satisfy with Equation (12).

### 3.2. Data Embedding Phase

All 1 × 7-sized blocks are determined to be embeddable or unembeddable by using the location map. The data embedding phase is described in detail in this subsection. Figure 5 shows the flowchart of the embedding phase for embeddable blocks.

Input: Original grayscale image.

Output: AMBTC-compressed stego bitstream.

- Step 1. Partition the original image into  $n \times 7$  pixel blocks and conduct AMBTC encoding; a set of compressed trios, i.e.,  $(H, L, Bm_{n \times 7})_{i,j}$ , is obtained, where  $H$  is the high mean table,  $L$  is the low mean table,  $Bm_{n \times 7}$  is the bitmap,  $n = 4$ , and  $(i, j)$  is the coordinate of the  $n \times 7$  pixel block where  $i = 1, 2, \dots, 128$  and  $j = 1, 2, \dots, 73$ .
- Step 2. Based on their positions, use the pixels located in the odd rows to predict the pixels located in the even rows with Equations (9)–(11), respectively. If all of the pixels in a 1 × 7-sized block satisfy Equation (13), determine the block to be embeddable and denote it as “1” in the location map,  $LP$ . Otherwise, denote it as “0” in the location map,  $LP$ .
- Step 3. After all blocks have been evaluated, concatenate  $LP$  and secret data  $SD$  as the final secret message  $S$ , where  $S = LP || SD$ , and “||” denotes the concatenation of  $LP$  and  $SD$ .
- Step 4. Scan all AMBTC-compressed blocks in a zig-zag direction to embed the final secret message,  $S$ , into bitmap  $Bm$ . If  $H = L$ , 4 × 7 bits of  $S$  are selected and used to replace the original  $Bm$ . If not,  $H$  is not equal to  $L$ , select one bit,  $s$ , of  $S$  and embed it into the  $H$  and  $L$  pair by swapping the order when  $s = 1$ . Note that, if  $H \neq L$  and  $s = 0$ , then the order of  $H$  and  $L$  is not changed.
- Step 5. After all  $H$  and  $L$  pairs have been checked, take three bits of the remaining  $S$  as  $m$  and embed them into block  $x$ , which is marked with “1” in the location map  $LP$  by using Equation (14). The detailed description can be referred to Section 2.2.

$$syndrome = m \oplus H_0x \text{ and } y = Emd(x, m) = x \oplus F(syndrome), \tag{14}$$

where  $y$  is the received stego vector, and  $F(\cdot)$  is the value of the  $i^{\text{th}}$  position of block  $x$  that must be changed.



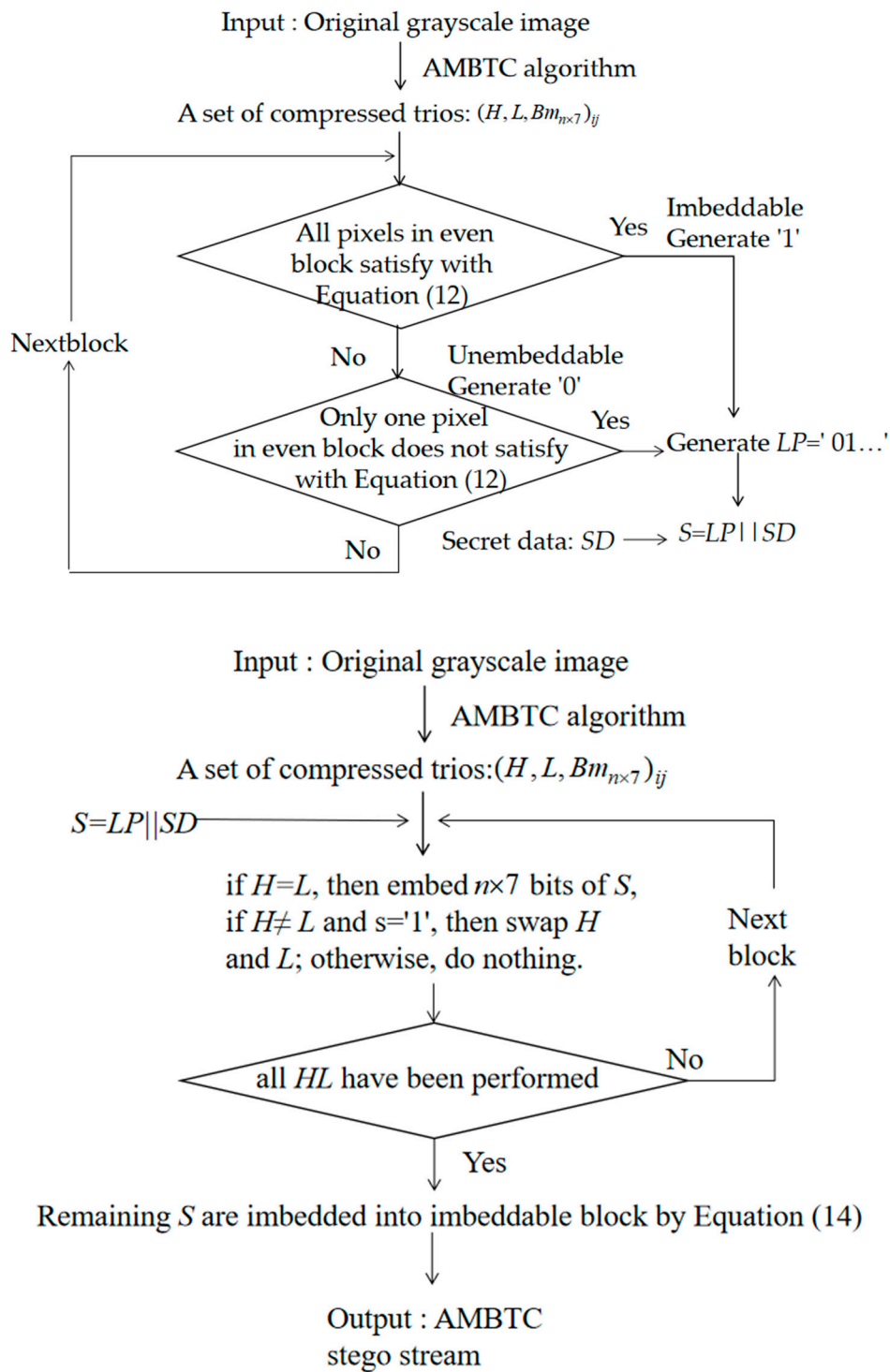


Figure 5. Flowchart of the embedding phase.

Example of Data Embedding

Below, we have provided an example to show our hybrid hiding strategy. First, assume that there is a secret bitstream as  $S = 101\ 010\ 101\ 100\ 011$  and an AMBTC-compressed trio ( $L = 155, H = 160, Bm = 10001101000110100011\ 010001001011110$ ), as shown in Figure 6. The first two bits, i.e., “10”, listed in  $S$  indicate that the  $1 \times 7$ -sized block located in the second row is embeddable and that the  $1 \times 7$ -sized block located in the fourth row is unembeddable. The third bit “1” of  $S$  indicates that the order of  $L$  and  $H$  should be swapped according to hiding strategy depicted in Step 2. The following

3-bits “010” is then embedded into the  $1 \times 7$  block located in the second row according to Equation (14). Since the block located in the second row is determined as embeddable and  $H \neq L$ , seven bits of the corresponding bitmap are extracted as  $x = (1\ 0\ 0\ 0\ 1\ 1\ 0)$ . Then, treat the 3-bit secret, “010”, as  $m$  and use parity check matrix,  $H_0$ . Finally,  $x$  must be changed as  $y = (1\ 0\ 0\ 0\ 1\ 0\ 0)$  to carry the secret bits “010”. Finally, the modified AMBTC-compressed trios are obtained and then transmitted to the receiver.

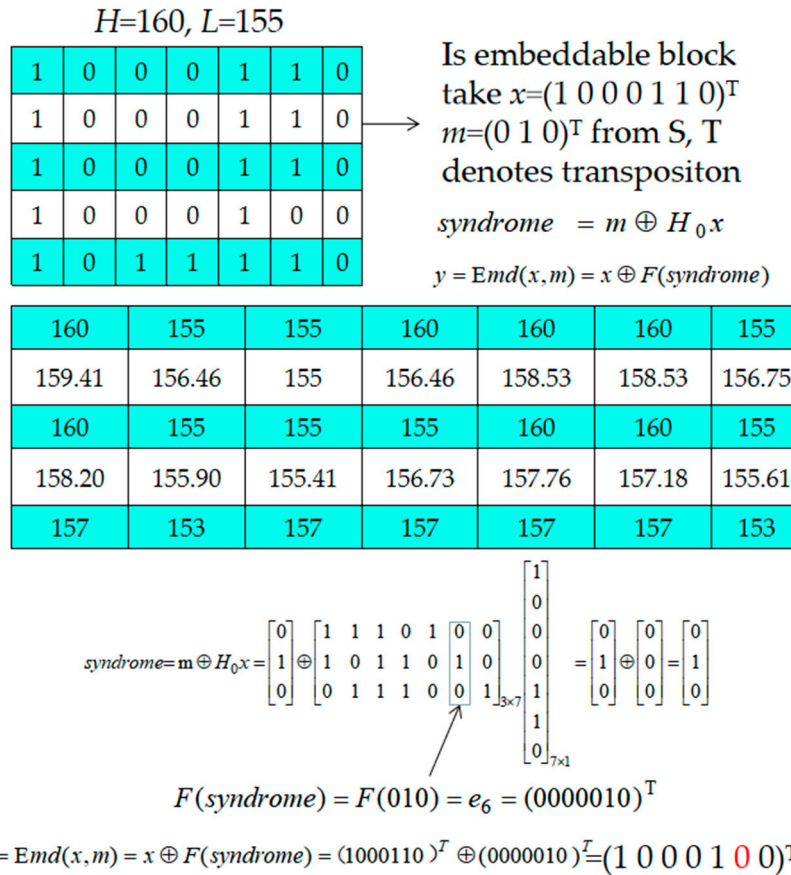


Figure 6. Example of data embedding phase.

### 3.3. Data Extraction and Recovery Phase

In data extraction phase, the receiver can perform data extraction with the received AMBTC-compressed trios,  $(H, L, Bm_{n \times 7})_{i,j}$ , where  $(i, j)$  is the coordinate of the  $n \times 7$  pixel block,  $i = 1, 2, \dots, 128$ , and  $j = 1, 2, \dots, 73$ . The parity check matrix,  $H_0$ , which is like a password, can be used several times as long as it is transmitted via a secure channel. Generally, the receiver is aware of the hidden data based on the order of  $L$  and  $H$ . From the extracted data from the  $L$  and  $H$ , location map  $LP$  can be derived to indicate which  $1 \times 7$  blocks are embeddable. Finally, not only the secret data are extracted but also the original AMBTC-compressed trios can be obtained. The detailed data extraction procedure is shown below.

Input: The stego bitstream with  $(H, L, Bm_{n \times 7})_{i,j}$  and the parity check matrix  $H_0$ .

Output: The original secret message and the reconstructed AMBTC compressed image.

- a. Step 1. Scan stego bitstream  $(H, L, Bm_{n \times 7})_{i,j}$ , and if  $H = L$ , extract  $1 \times 7$  secret bits from the  $Bm$ . If  $H > L$ , extract secret bit “0”; otherwise, extract secret bit “1”, and swap  $H$  and  $L$ .

- b. Step 2. Decide which  $1 \times 7$ -sized blocks located in even rows are embeddable or not according to the extracted location map,  $LP$ . If the current block is an embeddable block, 3 bits are extracted from the corresponding  $1 \times 7$  bits of  $Bm$  according to Equation (15) [34].

$$m' = H_0y \tag{15}$$

If the current block is unembeddable, go to Step 4.

- c. Step 3. Use Equation (12) to check each pixel of the embeddable block to see it is satisfied. If one pixel is not satisfied, modify its bit value to its complementary bit value, i.e., if the current bit value is "0", change it to "1" and vice versa.
- d. Step 4. Check the next embeddable block until all blocks are preceded. Output all extracted secret data and replace the corrected bitmaps with the corresponding  $H_s$  and  $L_s$  to obtain the reconstructed AMBTC-image.

Example of the Data Extraction and Recovery Phase

As shown in Figure 7, we demonstrate our proposed data extraction and recovery operation in this subsection. Here, we skip the data extraction and recovery operations for extracting the hidden data of the two quantizers and restoring the order of  $L$  and  $H$ . In this example, we focus only on how to extract the hidden data from the  $1 \times 7$ -sized block of  $Bm$  and restore the modified bit value. Based on the extracted location map,  $LP$ , the blocks of which are located in the even blocks can be easily to be identified. Therefore, the receiver knows which  $1 \times 7$ -sized block located in the second row is embeddable and which  $1 \times 7$ -sized block located in the second row should be treated as unembeddable. However, for the sixth pixel located in the second row, its prediction pixel value is 158.53 according to Equation (9), and the following equation does not hold. Such a situation is not consistent with the rule for the selection of an embeddable block as defined in Equation (12).

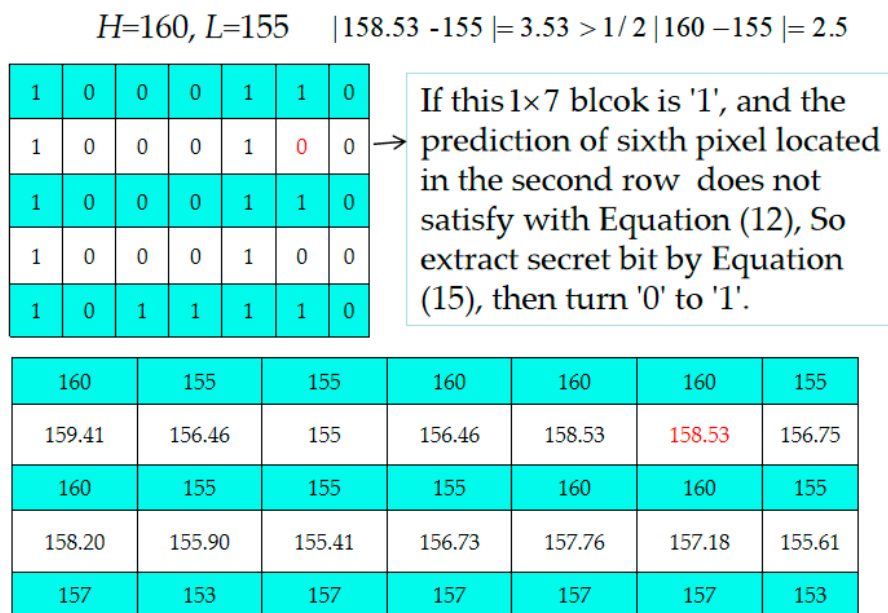


Figure 7. Example of the data extraction and recovery phase.

$$|158.53 - 155| > 1/2 |160 - 155|.$$

Therefore, we can conclude that the sixth pixel marked in red should be changed from "0" to "1." With our designed justified rule, the original  $1 \times 7$ -sized bitmap is guaranteed to be restored.

#### 4. Experimental Results

In this section, we describe the series of experiments and analyses that were performed to demonstrate the performance of the proposed scheme. All of the experiments were implemented in MATLAB R2014b on a PC with Intel® Core (TM) i7-8750H CPU @2.20 GHz, 16 GB RAM. The eight classic grayscale images, shown in Figure 8, with sizes of  $512 \times 512$ , i.e., Lena, Airplane, Barbara, Goldhill, Wine, Bird, Zelda, and Boat, were selected from the USC-SIPI data [35] and served as the test images. All of the test images were compressed using the AMBTC compression technique with the size of  $2 \times 7$  and  $4 \times 7$  pixels, respectively.

$$PSNR = 10 \log_{10} \left( \frac{255^2}{MSE} \right) (\text{dB}), \quad (16)$$

$$MSE = \sum_{i=1}^{HT} \sum_{j=1}^{WH} (x_{i,j} - x'_{i,j})^2 / (HT \times WH), \quad (17)$$



Figure 8. Eight test images

We defined Type I as the stego codes that can be correctly decoded via a standard BTC or AMBTC decoder [27]. In this section, we performed  $2 \times 7$  and  $4 \times 7$  experiments to demonstrate the performance of the proposed scheme and to compare the results with other related type I works, including the work of Chen et al. [23]. In Chen et al.'s scheme, the relationship of  $H$  and  $L$  was used to embed the secret bits.

In the first experiment, the peak signal-to-noise ratio ( $PSNR$ ) was used to estimate the visual quality of the AMBTC-compressed image, which is defined in Equation (16), where the mean-square-error ( $MSE$ ) is shown in Equation (17) and where  $x_{i,j}$  and  $x'_{i,j}$  indicate the pixel values for the position  $(i, j)$  of the original image and of the AMBTC-compressed image or steganography image, respectively.  $HT$  and  $WH$  represent the height and width of an image, respectively.

To examine the occurrence of secret bits in the stego bitstream codes, we used the embedding efficient rate ( $ER$ ), which was calculated by Equation (18). It is noted the stego bitstream codes mentioned in this section is the stego AMBTC compression codes generated by our proposed scheme.

$$ER = \frac{HC}{CS} \quad (18)$$

Both  $HC$  and  $CS$  are the total secret bits embedded and the size of the stego bitstream codes, respectively. An embedding method with higher efficiency indicated that our proposed RDH scheme offers a larger payload for the same size of stego bitstream codes.

In our proposed RDH scheme, when  $L = H$ , no secret bits can be embedded into two quantizers  $L$  and  $H$  by swapping the order of  $L$  and  $H$ . In addition, no secret bits can be embedded into the  $1 \times 7$  blocks located in the even rows. Table 1 indicates that there is no such scenario for “Lena,” “Barbara,” “Zelda,” and “Boat”, irrespective of whether the partition strategy was set as  $2 \times 7$  or  $4 \times 7$ . For “Airplane” and “Bird,” there was a single block the two quantizers of which were the same when the partition strategy was set as  $2 \times 7$ . By contrast, there are 398 blocks and 89 blocks which can be used for swapping the order of  $L$  and  $H$  to carry secret data in the “Wine” image.

**Table 1.** Amount blocks in which the high mean was equal to the low mean for  $2 \times 7$  and  $4 \times 7$  partition strategies.

Image	Lana	Airplane	Barbara	Goldhill	Wine	Bird	Zelda	Boat
$2 \times 7$	0	1	0	6	398	8	0	0
$4 \times 7$	0	0	0	1	89	0	0	0

Table 2 shows the embeddable blocks in eight test images for  $2 \times 7$  partition. The total number of blocks is  $512 \times 511/2/7 = 18,688$ . The average number of embeddable blocks is 3,869, and the average of ratio is 0.208. The number of blocks with one bit changed and the number of blocks without bits changed are also shown in Table 2. In other words, our designed prediction mechanism which is defined in data embedding phase and justify mechanism defined in data extraction and recovery phase is required.

**Table 2.** Embeddable blocks in eight test images for  $2 \times 7$  partition.

Image	Lana	Airplane	Barbara	Goldhill	Wine	Bird	Zelda	Boat
Number of embeddable blocks	3980	3482	3924	3037	5314	3676	4386	3155
Number of blocks with one bit changed	3614	3134	3556	2686	5060	3332	4006	2821
Number of blocks without bits changed	366	348	368	351	254	344	380	334
Ratio <sup>1</sup>	0.21	0.19	0.21	0.16	0.29	0.20	0.24	0.17

<sup>1</sup> Where Ratio = number of embeddable blocks/total number of odd blocks.

When Tables 3 and 4 were compared, it was obvious that the total hiding capacity with the  $2 \times 7$  partition was larger than that with the  $4 \times 7$  partitions in our proposed scheme. The comparison of the different partition indicated that the  $PSNR$  and  $HLC$  and  $THC$ ,  $IC$  are reduced when the partition size becomes larger. It is noted that the  $PSNR$  the computed from the stego compression codes generated by our proposed scheme and the original image. Because our proposed scheme is reversible data hiding scheme, the original AMBTC compression codes can be completely restored. In other words, the  $PSNR$  of our proposed scheme after extraction the hidden data will be the same as those listed in the first column in Table 4. Basically, there is no fixed rule between  $HC$  and  $EHC$ . In general, the more complex the image is, the less its embedding capacity is. As Table 1 indicates that only “Wine” has the largest hiding capacity in the  $4 \times 7$  and  $2 \times 7$  partitions when  $H = L$ . Although the effective hiding capacity ( $EHC$ ) in the  $2 \times 7$  case is less than that in the  $4 \times 7$  partition for bitmap embedding, it offers more quantizer pairs to carry secret data. Therefore, in general, the total hiding capacity is increased.

**Table 3.** Peak signal-to-noise ratios (*PSNRs*) and hiding capacity (*HC*) of our proposed reversible data hiding (RDH) scheme for the  $2 \times 7$  partition.

$2 \times 7$ (18,688) Criteria	AMBTC <i>PSNR</i> (dB)	<i>PSNR</i> (dB)	$H \neq L$ <i>HC</i> (bits)	Proposed			Total of <i>HC</i> <sup>1</sup> (bits)
				Scheme $H \neq L$ <i>IC</i> (bits)	$H \neq L$ <i>EHC</i> (bits)	$H = L$ <i>HLC</i> (bits)	
Lena	32.05	29.88	11,940	8612	3328	0	22,016
Airplane	31.9	29.17	10,446	7953	2493	14	21,194
Barbara	28.9	27.66	11,772	8266	3506	0	22,194
Goldhill	32.52	31.07	9111	7020	2091	84	20,857
Wine	32.02	29.9	15,942	10,456	5486	5572	29,348
Bird	29.57	28.3	11,028	8821	2207	112	20,999
Zelda	35.67	33.96	13,158	9794	3364	0	22,052
Boat	30.72	28.73	9465	7096	2369	0	21,057
Average	31.66	29.83	11,607	8502	3105	720	22,464

<sup>1</sup> Here *HC* is the data embedding capacity in the bitmap with the (7,4) hamming code. *IC* is the number of remarks concerning which the block is "1" or "0." *EHC* is the effective hiding capacity, where  $EHC = HC - IC$ . *HLC* is the data embedding capacity, when  $H = L$ , and *THC* is the total data hiding capacity, where  $THC = EHC$ ,  $THC = EHC + HLC + (18688 - HLC/14)$ .

**Table 4.** *PSNRs* and hiding capacity (*HC*) of our proposed RDH scheme for the  $4 \times 7$  partition.

$2 \times 7$ (9344) Criteria	AMBTC <i>PSNR</i> (dB)	<i>PSNR</i> (dB)	$H \neq L$ <i>HC</i> (bits)	Proposed			Total of <i>HC</i> <sup>1</sup> (bits)
				Scheme $H \neq L$ <i>IC</i> (bits)	$H \neq L$ <i>EHC</i> (bits)	$H = L$ <i>HLC</i> (bits)	
Lena	31.29	29.45	11,907	8902	3005	0	12,349
Airplane	30.49	28.21	12,159	8873	3286	0	12,630
Barbara	28.25	27.17	12,102	8553	3549	0	12,893
Goldhill	31.35	30.03	11,286	8063	3223	28	12,594
Wine	30.84	29.02	17,511	11,280	6231	2492	17,978
Bird	28.46	27.43	11,685	9352	2333	0	11,677
Zelda	34.66	33.25	13,440	10,223	3217	0	12,561
Boat	29.60	27.99	10,983	8037	2946	0	12,290
Average	30.61	29.06	12,634	9160	3474	315	13,121

<sup>1</sup> Here *HC* is the data embedding capacity in the bitmap with the (7,4) hamming code. *IC* is the number of remarks concerning which the block is "1" or "0." *EHC* is the effective hiding capacity, where  $EHC = HC - IC$ . *HLC* is the data embedding capacity, when  $H = L$ , and *THC* is the total data hiding capacity, where  $THC = EHC + HLC + (9344 - HLC/28)$ .

To further demonstrate the performance of our proposed RDH scheme, we compared it with the conventional AMBTC and with Chen et al.'s scheme [23], which is presented in Table 5. It is noted that the *PSNR* listed in Chen et al.'s scheme is demonstrated the *PSNR*.

**Table 5.** Comparisons of the performances of the proposed scheme and Chen et al.'s scheme.

Performance	Scheme	Lena	Airplane	Barbara	Goldhill	Wine	Bird	Zelda	Boat
<i>PSNR</i>	AMBTC	32.05	31.9	28.9	32.52	32.02	29.57	35.67	30.72
	Chen [23]	32.05	31.9	28.9	32.52	32.02	29.57	35.67	30.72
	Proposed	29.88	29.17	27.66	31.07	29.9	28.3	33.96	28.73
<i>HC</i>	Chen [23]	18,688	18,701	18,688	18,766	23,864	18,688	18,688	18,792
	Proposed	22,016	21,194	22,194	20,857	29,348	20,999	22,052	21,057
<i>CS</i> <sup>1</sup>	Chen [23]	564,736	564,736	564,736	564,736	564,736	564,736	564,736	564,736
	Proposed	564,736	564,736	564,736	564,736	564,736	564,736	564,736	564,736
<i>ER</i>	Chen [23]	0.033	0.033	0.033	0.033	0.042	0.033	0.033	0.033
	Proposed	0.039	0.038	0.039	0.037	0.052	0.037	0.039	0.037

<sup>1</sup> Note: *CS* is the size of the,  $ER = HC/CS$ .

Compared with Chen et al.'s scheme, which only embeds one-bit of secret data into a pair of quantization values by swapping them when two quantizers are not the same. Therefore, the sizes of the compression codes of the two schemes were the same. Therefore, the ER of our proposed RDH scheme for the  $2 \times 7$  partition was higher than that of Chen et al.'s scheme. In general, the hiding capacity of our proposed RDH scheme for the  $2 \times 7$  partition was higher than that of Chen et al.'s scheme at the cost of slightly less quality of the images than conventional AMBTC and Chen et al.'s scheme. This is because our proposed RDH scheme embeds both secret data and bitmaps into the quantizer pairs. Even the image quality of the stego compression codes with our proposed scheme is slightly less than that of Chen et al.'s scheme [23]. It is noted that both of our schemes are reversible data hiding. In other words, the original AMBTC compression codes are always completely restored after the extraction of the hidden data. The size of the stego compression codes is always the same as that of Chen et al.'s scheme [23] and the original AMBTC compression codes. Therefore, no matter from the size and structure of stego compression codes, our stego compression codes would not attract attackers' attention even the size of hidden data is larger than that of Chen et al.'s scheme [23].

To further prove the performance of our proposed RDH scheme on visual quality and hiding capacity outperforms other schemes, the comparisons of our proposed RDH scheme and five representative BTC/AMBTC-based RDH schemes are presented in Table 6.

**Table 6.** Comparisons of the performances of the proposed scheme and five representative block truncation coding (BTC)/AMBTC-based RDH schemes.

Schemes	Types	Average PSNRs	max PSNR/ min PSNR	Average HC	max HC/ min HC	Average ER	max ER/ min ER
Proposed	I(code)	29.83	33.96/27.66	22,464	29,348/20,999	0.04	0.052/0.037
Chen et al. [23]	I(code)	32.28	35.67/28.90	19,359	23,864/18,688	0.037	0.042/0.033
Lo et al. [24]	I(code)	33.1	33.23/32.97	3615	4570/2660	0.006	0.008/0.005
Chang et al. [25]	I(code)	31.74	32.89/30.59	16,381	12,683/20,080	0.031	0.024/0.038
Sun et al. [26]	II(code)	29.7	33.40/26.0	64,008	64,008/64,008	0.122	0.122/0.122
Hong et al. [27]	II(code)	30.19	33.39/26.91	64,516	64,516/64,516	0.12	0.116/0.125
Lin et al. [28]	II(code)	33.36	37.23/30.91	90489	114,533/70,889	0.240	0.241/0.217

From Table 6, we can see BTC/AMBTC-based RDH schemes [26–28] usually offer high hiding capacity at the cost of their stego-compression codes cannot be correctly decoded by the conventional BTC/AMBTC decoders. In other words, attackers can guess there is valuable information has been hidden out there. It could make the hidden conventional data insecure. By contrast, our proposed RDH scheme belongs to Type I AMBTC-based RDH scheme, and the stego-compression codes can be always decoded by the conventional AMBTC decoder. That means the stgo-compression codes could not attract malicious attackers' attention and the security of hidden data is better than that of RDH schemes which belong to Type II BTC/AMBTC-based RDH schemes [26–28]. To make sure the stego-compression codes is decodable by conventional AMBTC decoder; the hiding capacity is relatively limited. However, from Table 6, we can see the hiding capacity of our proposed RDH scheme is only less than that of schemes of Sun et al. [26] and Hong et al. [27]. As for Lin et al.'s scheme [28], even their hiding capacity is significantly higher than ours, their scheme is Type II and they hid secret data into spatial domain of image by referring the features derived from AMBTC. In other words, they did not conceal secret data into structure of AMBTC compression codes but into the pixels of images. However, their scheme still points out the other possible direction which we shall explore and set as our research topic in the future.

## 5. Discussion and Conclusions

In this paper, we proposed RDH based on AMBTC with (7,4) hamming code. We embedded secret bits into the relationship of *HL* and into the bitmap. Before embedding the secret bits, we determined which blocks were embeddable, and we embedded the secret bits into this embeddable block with (7,4)

hamming code. Compared with Type I AMBTC-based RDH scheme, experimental results confirm that our proposed RDH scheme has its merit in the capacity of the bitmap. Moreover, the hiding capacity offered by our proposed RDH scheme is significantly higher than that of schemes of Chen et al. [23], Lo et al. [24], and Chang et al. [25]. However, we also found even our proposed RDH scheme has enhanced the hiding capacity than other existing Type I BTC/AMBTC-based RDH schemes, but it is still less than that of Type II BTC/AMBTC-based RDH schemes. It will be our next research topic to allow our proposed RDH scheme to be supportive for various applications while maintaining security of the hidden data. Moreover, the data hiding method based on neural networks [36] and other methods [37–41] will be included to improve the hiding capacity in the future.

**Author Contributions:** C.-C.C. conceived the overall idea of the article. J.L. carried out the experiments and analyzed the data. C.-C.L. wrote the paper.

**Funding:** This work was supported in part by the Natural Science Foundation of Fujian Province of China under Grant 2018J01788, 2015J05146.

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

## References

- Hong, W.; Chen, T.S.; Chen, J. Reversible data hiding using delaunay triangulation and selective embedment. *Inf. Sci.* **2015**, *308*, 140–154. [[CrossRef](#)]
- Zielinska, E.; Mazurczyk, W.; Szczpiorski, K. Trends in steganography. *Commun. ACM* **2014**, *57*, 86–95. [[CrossRef](#)]
- Barton, J.M. Method and Apparatus for Embedding Authentication Information with Digital Data. U.S. Patent 5646997, 8 July 1997.
- Tian, J. Reversible data embedding using a difference expansion. *IEEE Trans. Circuits Syst. Video Technol.* **2003**, *13*, 890–896. [[CrossRef](#)]
- Li, X.; Zhang, W.; Gui, X.; Yang, B. A novel reversible data hiding scheme based on two-dimensional difference-histogram modification. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 1091–1100.
- Ni, Z.; Shi, Y.Q.; Ansari, N.; Su, W. Reversible data hiding. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 354–362.
- Lin, C.C.; Tai, W.L.; Chang, C.C. Multilevel reversible data hiding based on histogram modification of difference images. *Pattern Recognit.* **2008**, *41*, 3582–3591. [[CrossRef](#)]
- Wang, W.; Ye, J.; Wang, T. A high capacity reversible data hiding scheme based on right-left shift. *Signal Process.* **2018**, *150*, 1129–1143. [[CrossRef](#)]
- Li, X.; Yang, B.; Zeng, T. Efficient reversible watermarking based on adaptive predirection-error expansion and pixel selection. *IEEE Trans. Image Process.* **2011**, *20*, 1061–1070.
- Li, X.; Zhang, W.; Gui, X.; Yang, B. Efficient reversible data based on multiple histograms modification. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 2016–2027.
- Xiao, M.; Li, X.L.; Wang, Y.Y.; Zhao, Y.; Ni, R.R. Reversible data hiding based on pairwise embedding and optimal expansion path. *Signal Process.* **2019**, *158*, 210–218. [[CrossRef](#)]
- Ahmed, N.; Natarajan, T.; Rao, K. Discrete cosine transform. *IEEE Trans. Comput.* **1974**, *C-23*, 90–93. [[CrossRef](#)]
- Haar, A. Discrete cosine transform. *Math. Ann.* **1911**, *71*, 38–53. [[CrossRef](#)]
- Gray, R. Vector quantization. *IEEE ASSP Mag.* **1984**, *1*, 4–29. [[CrossRef](#)]
- Kim, T. Side match and overlap match vector quantizers for images. *IEEE Trans. Image Process.* **1992**, *1*, 170–185. [[CrossRef](#)]
- Yang, B.; Lu, Z.; Sun, S. Reversible watermarking in the VQ-compressed domain. In Proceedings of the 5th IASTED International Conference on Visualization, Imaging, and Image Processing, Benidorm, Spain, 7–9 September 2005; pp. 273–275.
- Lu, Z.; Wang, J.; Liu, B. An improved lossless data hiding scheme based on image VQ-index residual value coding. *J. Syst. Softw.* **2009**, *82*, 1016–1024. [[CrossRef](#)]
- Chang, C.C.; Tai, W.L.; Lin, C.C. A reversible data hiding scheme based on side match vector quantization. *IEEE Trans. Circuits Syst. Video Technol.* **2006**, *16*, 1301–1308. [[CrossRef](#)]



19. Chiou, S.F.; Liao, I.E.; Hwang, M.S. A capacity-enhanced reversible data hiding scheme based on SMVQ. *Imaging Sci. J.* **2011**, *59*, 17–24. [[CrossRef](#)]
20. Delp, E.J.; Mitchell, O.R. Image compression using block truncation coding. *IEEE Trans. Commun.* **1979**, *27*, 1335–1342. [[CrossRef](#)]
21. Rover Camera Instrument Description. Available online: [https://pdsimg.jpl.nasa.gov/data/mpfr-m-rvrcam-2-edr-v1.0/mprv\\_0001/document/rcinst.htm](https://pdsimg.jpl.nasa.gov/data/mpfr-m-rvrcam-2-edr-v1.0/mprv_0001/document/rcinst.htm) (accessed on 9 November 2011).
22. Lema, M.; Mitchell, O.R. Absolute moment block truncation coding and its application to color images. *IEEE Trans. Commun.* **1984**, *19*, 1148–1157. [[CrossRef](#)]
23. Chen, J.; Hong, W.; Chen, T.S.; Shiu, C.W. Steganography for BTC compressed image using no distortion technique. *Imaging Sci. J.* **2010**, *58*, 177–185. [[CrossRef](#)]
24. Lo, C.C.; Hu, Y.C.; Chen, W.L.; Wu, C.M. Reversible data hiding scheme for BTC-compressed images based on histogram shifting. *Int. J. Secur. Appl.* **2014**, *8*, 301–314. [[CrossRef](#)]
25. Chang, I.C.; Hu, Y.C.; Chen, W.L.; Lo, C.C. High capacity reversible data hiding scheme based on residual histogram shifting for block truncation coding. *Signal Process.* **2015**, *108*, 376–388. [[CrossRef](#)]
26. Sun, W.; Lu, Z.M.; Wen, Y.C.; Yu, F.X.; Shen, R.J. High performance reversible data hiding for block truncation coding compressed image. *SIViP* **2013**, *7*, 297–306. [[CrossRef](#)]
27. 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](#)]
28. Lin, C.C.; Chang, C.C.; Wang, Z.M. Reversible Data hiding scheme using adaptive block truncation coding based on an edge-based quantization approach. *Symmetry* **2019**, *11*, 765. [[CrossRef](#)]
29. Chang, C.C.; Kieu, T.D.; Chou, Y.A. high payload steganographic scheme based on (7,4) hamming code for digital images. In Proceedings of the 2008 International Symposium on Electronic Commerce and Security, Guangzhou, China, 3–5 August 2008; pp. 16–21. [[CrossRef](#)]
30. Cao, Z.; Yin, Z.; Hu, H.; Gao, X.; Wang, L. High capacity data hiding scheme based on (7,4) hamming code. *Springer Plus* **2016**, *5*, 1–13. [[CrossRef](#)]
31. Bai, J.; Chang, C.C. A high payload steganographic scheme for compressed images with hamming code. *Int. J. Netw. Secur.* **2016**, *18*, 1122–1129.
32. Biswapati, J.; Giri, D.; Mondal, S.K. Partial reversible data hiding scheme using (7,4) hamming code. *Multimed. Tool Appl.* **2017**, *76*, 21691–21706.
33. Biswapati, J.; Giri, D.; Mondal, S.K. Dual image based reversible data hiding scheme using (7,4) hamming code. *Multimed. Tool Appl.* **2018**, *77*, 763–785.
34. Mao, Q. A fast algorithm for matrix embedding steganography. *Digit. Signal Process.* **2014**, *25*, 248–254. [[CrossRef](#)]
35. The USC-SIPI Image Database. Available online: <http://sipi.usc.edu/database> (accessed on 9 November 1977).
36. Luigi, F.; Paolo, A.; David, B.; Ákos, Z. Cellular neural networks: A paradigm for nonlinear spatio-temporal processing. *IEEE Circuits Syst. Mag.* **2001**, *1*, 6–21.
37. Zhong, H.; Chen, X.; Tian, Q. An Improved Reversible Image Transformation Using K-Means Clustering and Block Patching. *Information* **2019**, *10*, 17. [[CrossRef](#)]
38. Hu, Y.C.; Lin, Y.H.; Lo, Y.H.; Lo, C.C.; Wu, C.M. Implementation of Block-Based Hierarchical Prediction for Developing an Error-Propagation-Free Reversible Data Hiding Scheme. *Symmetry* **2019**, *11*, 1146. [[CrossRef](#)]
39. Leng, H.S. Generalized Scheme Based on Octagon-Shaped Shell for Data Hiding in Steganographic Applications. *Symmetry* **2019**, *11*, 760. [[CrossRef](#)]
40. Chen, K.M.; Chang, C.C. Real-Time Error-Free Reversible Data Hiding in Encrypted Images Using (7,4) Hamming Code and Most Significant Bit Prediction. *Symmetry* **2019**, *11*, 51. [[CrossRef](#)]
41. Hou, X.; Min, L.Q.; Yang, H. A Reversible Watermarking Scheme for Vector Maps Based on Multilevel Histogram Modification. *Symmetry* **2018**, *10*, 397. [[CrossRef](#)]

