IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received August 22, 2019, accepted September 12, 2019, date of publication September 24, 2019, date of current version October 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2943505

HBF-DH: An Enhanced Payload Hybrid Data Hiding Method Based on a Hybrid Strategy and Block Features

ZHAN YU[1,2], CHIA-CHEN LIN[3], CHIN-CHEN CHANG[4], (Fellow, IEEE), AND GUO-DONG SU[1,4]

[1]School of Electronics and Information Engineering, Fuqing Branch of Fujian Normal University, Fuzhou 350300, China
[2]Engineering Research Center for ICH Digitalization and Multi-Source Information Fusion, Fuqing Branch of Fujian Normal University, Fuzhou 350300, China
[3]Computer Science and Information Management, Providence University, Taichung 43301, Taiwan
[4]Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan

Corresponding author: Chia-Chen Lin (ally.cclin@gmail.com)

This work was supported in part by the Education-Scientific Research Project for Middle-Aged and Young of Fujian Province under Grant JAT170683, and in part by the Project of Ministry of Science and Technology of Taiwan under Grant MOST 106-2221-E-035-013-MY3, Grant MOST 108-2410-H-126 -021, and Grant MOST 106-2221-E-035-018-MY3.

ABSTRACT This paper proposes an innovative and novel steganography method for an AMBTC compressed image by combining a turtle-shell reference matrix and (7, 4) Hamming code, called hybrid strategy and block feature-based data hiding method (HBF-DH). Firstly, the proposed method compresses an original image by an AMBTC compression technique, and then applies a user predefined threshold pair to classify all compressed blocks into three categories: a smooth block, less complex block, and high complex block. Next, the smooth and less complex blocks are smoothed out with the aim of vacating more space for data hiding. The embedding process is divided into two phases. In the first phase, each quantization level pair of the AMBTC compression codes is used to embed an 8-ary secret digit by using a restrictive turtle-shell reference matrix. The second phase embeds binary secret data into a bitmap according to block characteristics. For smooth blocks, the bitmap is directly replaced as binary secret data by using bit replacement. For a less complex block, the binary secret data is concealed into a bitmap by using our proposed (7, 4) Hamming code based data hiding strategy, which increases the data hiding capacity without significant impact on image quality. The experimental results and analyses prove that the proposed HBF-DH method is superior to other existing AMBTC based data hiding methods in terms of data hiding ability with satisfactory image visual quality.

INDEX TERMS Steganography, AMBTC, (7, 4) hamming code, turtle-shell reference matrix, high payload, block features.

I. INTRODUCTION

With the increasing popularity of the Internet, information security has become increasingly important, especially in regard to the security of data exchange in public channels [1]–[3]. Because of the invisible nature of steganographic technologies, which are not vulnerable to the attention of attackers, and they are increasingly turned to as possible solutions for securing the transmitted data over the insecure channels.

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

Currently, steganographic technologies are generally divided into the following three categories: spatial domain-based [35], frequency domain-based [36], and compression domain-based [37]. Spatial domain-based methods hide confidential data by directly modifying the original pixels of an image [28]–[31]. Among the spatial domain-based data hiding methods, the least-significant bits (LSB) substitution [31] is the most typical strategy and embeds data into the least significant bits of pixels in an image.

Following the LSB-substitution concept, many LSB-based data hiding variants have been proposed to increase hiding capacity while maintaining limited distortion [29]. In general, the hiding capacity of spatial domain-based methods is the largest compared with the remaining categories at the cost of easily suffering from attacks. The cover images used in spatial domain-based methods can be binary images [27], grayscale images [28], [29], [31] and color images [30].

In frequency domain-based methods [34], a cover image's pixels are first converted into coefficients by a pre-determined transformation function, i.e., a discrete wavelet transformation (DWT) [38], a discrete cosine transformation (DCT) [34], or discrete Fourier transformation (DFT) [39]. Confidential data is then hidden into these coefficients by modifying the coefficients. In general, the hiding capacity of frequency domain-based methods is less than that of the spatial domain-based methods. However, they offer the hidden data better protection and their stego images can withstand more attacks compared with those generated by the spatial domain-based data hiding methods.

As online communities such as Facebook, Line, WeChat, and Instagram continue to grow; users have become accustomed to recording their daily life via digital cameras and then sharing their photos online with their friends. However, the direct transmission of images or video consumes huge communication resources. Therefore, images or video are usually compressed to reduce the required bandwidth and speed up data transmission. As such, researchers also expanded the categories for data hiding from spatial domain-based and frequency domain-based to compression domain-based.

Compression domain-based methods first compress an image with an existing compression method, i.e., JPEG [6], vector quantization (VQ) [7], block truncation coding (BTC) [8]–[10], and then embed the confidential data into the compression codes [32], [33]. Finally, the generated compressions codes are modified to carry confidential data. Among modern compression techniques, BTC is a type of lossy image compression technique for greyscale images [8] and it has unique features that require a significantly smaller computational load and relative less memory than other data compression techniques. Later, absolute moment block truncation coding (AMBTC), a BTC variant, was proposed and uses the first absolute moment and mean to present the general features of a block [12].

AMBTC's compression speed is relatively efficient and effective, and the PNSR of the decompressed image is better than other BTC variants. Many researchers treated the compression codes of AMBTC as cover media and studied how to apply different block classification strategies to achieve good performance on hiding capacity no matter their designed data hiding schemes with or without reversibility [11]–[16], [41], [42]. Chuang et al. proposed a BTC-based data hiding method in 2006 [13] which classified BTC compression codes into smooth and edge categories by a pre-determined threshold, and embedded 16 bits secret data into each smooth block. Ou and Sun proposed an AMBTC-based adaptive data hiding method in 2015 [14]. They classified AMBTC compression codes into smooth and complex categories, which embedded an additional one bit in the complex category by changing the order of the quantization level pair and recalculated the quantization levels in the smooth category based on the replaced bitmap to reduce distortion after using a replacement strategy on a bitmap. In 2015, Lin *et al.* [41] proposed an AMBTC-based reversible data hiding method, which first used the redundancy of an AMBTC compressed block to identify if it is embeddable or un-embeddable, and then utilized four combinations of mean value and standard deviation to achieve a high payload while limiting distortion. In 2017, Hong *et al.* [18] proposed a method that utilized a quantization level modification technique and quantization perturbation technique to reduce the distortion caused by bitmap replacement and embedded additional 2 bits of secret data into the complex block by the parity of two quantization levels. Chen et al. proposed a high-quality data hiding method based on block truncation coding in 2017 [15] that classified BTC compressed code into smooth block, complex_1 block and complex_2 block based on two thresholds, and then three different hiding strategies were respectively applied in bitmap embedding for high visual quality. In 2019, Lin *et al.* [42] proposed a reversible data hiding method based on an edge-based quantization (ABTC-EQ) approach that first classified AMBTC compressed blocks into edge-block and non-edge block; and then used zero-point fixed histogram shifting to enhance hiding capacity while maintaining satisfactory visual quality in the stego image. In the same year, Kumar et al. utilized two thresholds to classify AMBTC compressed blocks into three types: smooth block, less smooth block and complex block, and then adopted Hamming distance and the pixel value difference to design a hiding strategy for prompting the performance in embedding capacity and visual quality of the stego image.

The above literature review indicates how block classification strategies have been designed based on different features or thresholds for either conventional data hiding methods or reversible data hiding methods. Among these various methods, reversibility usually limits the hiding capacity. Therefore, in this paper, we only propose a novel steganographic method for an AMBTC compressed image, called HBF-DH method. To enhance the hiding capacity while eliminating the image distortion, a new weight matrix was designed to serve as a reference matrix for a smooth filter before data hiding. Additionally, two thresholds are used to classify the compression codes into three categories: smooth blocks, less complex blocks and high complex blocks. Each category has its own data embedding strategy in our proposed HBF-DH method.

In general, the data embedding process of our proposed HBF-DH method is divided into two phases. In the first phase, each quantization level pair of AMBTC compression codes is used to embed an 8-ary secret digit by using a restrictive turtle-shell reference matrix. The second phase embeds binary secret data into a bitmap according to the characteristic of the block. For smooth blocks, the bitmap is directly replaced as binary secret data by using bit replacement. For a less complex block, 6 bits of binary secret data is concealed into a bitmap by using (7, 4) Hamming code based data hiding strategy. Our method is significantly superior to other methods in terms of embedding capacity while achieving acceptable performance in visual quality. The advantages and disadvantages of the proposed HBF-DH method are discussed in detail in Section IV. The main contributions of the proposed HBF-DH method include:

(1) Provides a real-time secret communication method under low bandwidth conditions.
(2) Addresses the problem of low data hiding capacity based on compressed domain.
(3) Provides an image optimization method that effectively improves the AMBTC compression quality of complex images and has no negative impact on general images.
(4) A predefined threshold combination improves the performance of a data hiding method based on block classification, expanding the data hiding capacity while ensuring image quality.
(5) An innovative use of the (7, 4) Hamming code and the turtle-shell reference matrix for data hiding of an AMBTC compressed image.

The rest of this paper is arranged as follows. Section II briefly introduces the AMBTC methodology and some related work. The proposed embedding and extraction algorithm are described in Section III. Experimental results and analysis are given in Section IV. Finally, Section V offers conclusions.

II. RELATED WORK

In this section, two primary techniques are introduced in Subsections II-A and II-B, respectively, to give readers enough background knowledge. One is AMBTC compression technique and the other is (7,4) Hamming coding technique.

A. AMBTC TECHNIQUE

BTC was first proposed by Mitchell et al. in 1979 [8] as an image compression technique to encode grayscale images. It is based on local binarization of non-overlapping blocks in a grayscale, and then a sample mean and variance are preserved to work with a bitmap so that an image can be recovered at the decoding phase. Their experimental results confirmed that BTC guarantees a 1 bpp data rate for coding a 32-level color image. Later, BTC was improved by Lema et al. in 1984 [9], in a method called absolute moment block truncation coding (AMBTC), which is easier and simpler to implement in the compression, while providing reasonable performance with a no less than a 2.13 bpp data rate [12].

Generally, the AMBTC encoding phase begins from dividing an original image I into several non-overlapping blocks C_i with a size of $m \times m$ by a raster scan. For each block, the i-th pixel is denoted as x_i where $1 \leq i \leq 16$, so the mean value of the block $\bar{x}$ can be calculated by Equation (1). And then, the bitmap can be constructed by a comparison of the result of x_i with $\bar{x}$ by Equation (2):

$$\bar{x} = \frac{\sum_{i=1}^{m} x_i}{m}. \tag{1}$$

$$BM_i = \begin{cases} 0 & x_i < \bar{x} \\ 1 & x_i \geq \bar{x}. \end{cases} \tag{2}$$

According to the value of bitmap BM_i, the block's pixel can be divided into two groups G_0 and G_1. Here, q is denoted as the number of G_1 corresponding to $BM_i = 1$, and $(m - q)$ is the number of G_0 corresponding to $BM_i = 0$. As a result, the lower quantitation value a and the high quantization value b can be calculated by Equations (3) and (4).

$$a = \frac{1}{m-q} \sum_{i=1}^{m-q} x_i, \tag{3}$$

$$b = \frac{1}{q} \sum_{i=1}^{q} x_i. \tag{4}$$

Therefore, the compression code, also called a trio of block C_i is represented as $\{a_i, b_i, BM_i\}$, and the original image can be represented by $\{a_i, b_i, BM_i\}_{i=1}^{N}$.

The following example demonstrates phases of encoding and decoding of AMBTC: assume that block $X = [83, 83, 76, 80; 75, 83, 83, 78; 79, 79, 79, 76; 80, 80, 81, 78]$ is a 4×4 block of an image, and its block mean value $\bar{x} = 79.56$ is computed. After comparing x_i, where $1 \leq i \leq 16$, with $\bar{x}$ the bitmap $BM_i = [1, 1, 0, 1; 0, 1, 1, 0; 0, 0, 0, 0; 1, 1, 1, 0]$ can be concluded. Correspondingly, the low quantization value a and the high quantization value b can be gained by rounding the average value of the pixel in groups G_0 and G_1, as $a = 73$, $b = 82$, respectively. Finally, the trio $\left\{73, 82, BM_i = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix}\right\}$ is generated for block X. During the decoding phase, the element of bitmap BM_i is sequentially substituted by either the low quantization value a or the high quantization value b according to whether its corresponding element is equal to 0 or 1. Therefore, the reconstructed block with AMBTC decoding is obtained as $X' = [82, 82, 73, 82; 73, 82, 82, 73; 73, 73, 73, 73; 82, 82, 82, 73]$.

B. (7, 4) HAMMING CODE

As a linear error correction code, the (7, 4) Hamming code [26] was first proposed by Richard Hamming in 1950. Since then, it has been widely used in data hiding as an efficient steganography method to achieve satisfactory image visual quality. It has a fascinating characteristic of only utilizing three parity check bits and a parity check matrix to work with four original bits to guarantee that one error bit can be successfully identified by the recipient.

Specifically, 4-bit original bits denoted by d_1, d_2, d_3, d_4 are used to yield 3-bit parity check bits, i.e., p_1, p_2, p_3, by multiplying with the code generator matrix G of the (7, 4) Hamming code. Accordingly, one code C with a size of

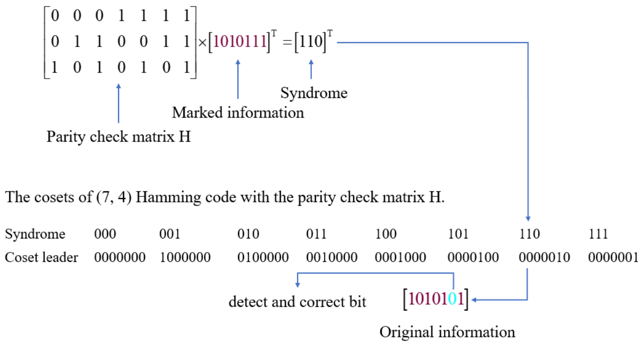


FIGURE 1. The error detection example based on (7, 4) Hamming code.

7 bits is formed by combining 4 original bits with 3 parity bits. The procedure can be represented by the following equations:

$$\begin{aligned} C &= d \times G \\ &= (d_1, d_2, d_3, d_4) \times \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}, \\ &= (p_1, p_2, d_1, p_3, d_2, d_3, d_4). \end{aligned} \tag{5}$$

The three parity check bits p_1, p_2, p_3 in Equation (5) can be computed by the following Equation (6), where $\oplus$ is the exclusive-or operation:

$$\begin{aligned} p_1 &= d_1 \oplus d_2 \oplus d_4 \\ p_2 &= d_1 \oplus d_3 \oplus d_4 \\ p_3 &= d_2 \oplus d_3 \oplus d_4. \end{aligned} \tag{6}$$

Fig. 1 presents an example demonstrating how to utilize the principle of (7, 4) Hamming coding to derive a codeword that carries 4 secret bits. Assume secret data is d = [1101], and its 3-bit parity code can be computed by Equations (6), i.e., p1 =1, p2 = 0, p3 = 0. Finally, a codeword carrying 4 secret bits can be derived from Equation (5), i.e., C = [1010111]. After the message is received, the recipient utilizes parity check matrix H to detect and correct one error bit by Equation (7), that is:

$$z = \left(H \times R^T\right)^T, \tag{7}$$

to verify whether the message has been tampered, where R represents the 7-bit binary representation of the tampered message, and z is called the syndrome vector. We can judge whether R is tampered or not by the value of z. Specifically, $z = 0$ indicates R is not tampered, otherwise, R is tampered. Assume $R = (0100001)_2$, then $z = (101)_2 = 5$, which implies that one error bit occurs in the fifth bit of R, and the original data can be revealed by flipping the fifth bit of R as shown in Fig. 1.

III. THE PROPOSED HBF-DH SCHEME

Inspired by Ou and Sun [14], we propose a new AMBTC-based data hiding method, called HBF-DH method, which enhances hiding capacity by classifying blocks into three types as an extension of the block classification-based method. In general, there is the smaller distortion caused by replacing the bitmap of smooth block with secret data, but it arises significant distortion in complex block. Therefore, we deal with them by different strategies according the block feature. The neighboring pixels could be used to predict the complexity of a block, if the predictive value large than a predefined threshold, the block is seen as complex. On the contrary, the block is smooth. In our proposed scheme, we utilize two build-in quantization levels to reflect the complexity of the block. These two quantization levels are derived from experimental results. To increase the hiding capacity, two predetermined thresholds are used to segment the absolute difference between the two quantization levels a and b of an AMBTC compressed block into three ranges; then each block can be categorized as: smooth, less complex or complex. The proposed data hiding consists of two phases: data embedding and data extraction, which are described in Subsections III-A and III-B, respectively. To give readers a better understanding, two examples are demonstrated at the end of these two subsections.

A. DATA EMBEDDING PHASE

Our proposed data embedding phase consists of three operations: preparation operation, TSM embedding operation, and bitmap operation. During the preparation operation, cover image I' is consecutively divided into a serial arrangement of 4×4 no-overlapping blocks. Those blocks are then transformed into a set of AMBTC compression codes (a, b, BM). To reduce the distortion of the stego image, a predefined threshold pair (T_1, T_2) is applied to classify the AMBTC compressed code into two categories according to two inequalities $|a_i - b_i| \leq T_1$ and $|a_i - b_i| \leq T_2$. If the inequality $|a_i - b_i| \leq T_1$ holds, the current trio is classified to smooth block, and if otherwise, it is regarded as a complex block. The complex block is further classified to less complex and high complex by the inequality $|a_i - b_i| \leq T_2$ for the higher payload based on the similar principle. To enhance hiding capacity while minimizing image distortion, in our proposed scheme, except for high complex blocks, all remaining blocks are further smoothed using a smooth filter that is defined by the weighted average value of each pixel and the eight pixels surrounding this pixel according to the weight matrix presented in Fig. 2.

After the preparation operation, three categories, smooth block, less complex block, and high complex block, of the AMBTC compression codes have been identified. A data embedding operation is conducted based on a turtle-shell reference matrix, called TSM, defined in [11] as shown in Fig. 3. Note that the scales of horizontal axes p_i and vertical p_j are from 0 to 255. Pixel pair $\left(p_i, p_j\right)$ presents a point on the coordinate plane and ranges from 0 to 7. The value of TSM (0, 0) is set to 0 and there are two important rules for TSM: (1) two consecutive adjacent elements in the horizontal direction should satisfy Equations (8) and (9), and two consecutive adjacent elements in the vertical direction should

$I_{i-1,j-1}$	$I_{i-1,j}$	$I_{i-1,j+1}$
$I_{i,j-1}$	$I_{i,j}$	$I_{i,j+1}$
$I_{i+1,j-1}$	$I_{i+1,j}$	$I_{i+1,j+1}$

1	4	1
4	16	4
1	4	1

FIGURE 2. Weight matrix.

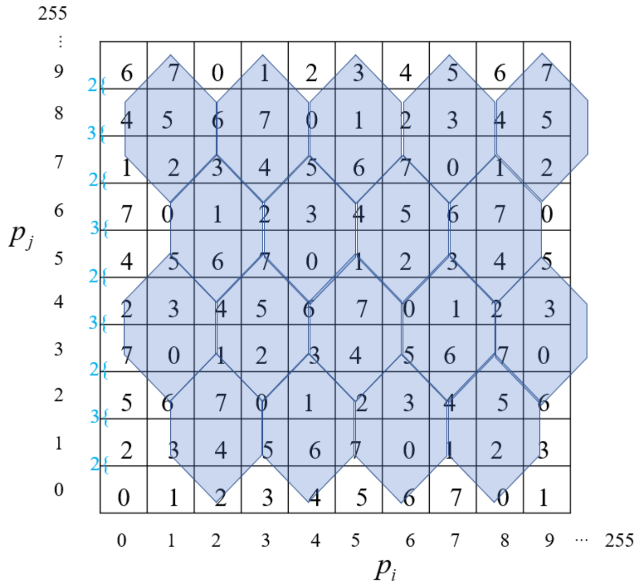


FIGURE 3. The turtle shell reference matrix, called TSM.

satisfy the TSM embedding procedure separately. TSM modifies the quantization level pair by a replacement strategy based on the turtle-shell reference matrix to provide a 3-bit payload, which increases the data hiding capacity without any significant impact on visual quality.

$$TSM(p_{i+1}, p_j) = (TSM(p_i, p_j) + 1) \bmod 8. \quad (8)$$

$$TSM(p_i, p_{j+1}) = \begin{cases} (TSM(p_i, p_j) + 2) \bmod 8 \text{ if } p_j \text{ is even,} \\ (TSM(p_i, p_j) + 3) \bmod 8 \text{ if } p_j \text{ is odd.} \end{cases} \quad (9)$$

Then, denote the high quantization level b and the low quantization level a as the horizontal p_i and vertical axes p_j. Finally, the coordinate (p_i, p_j) is represented as a point in a coordinate plane. The rules in our method are as follows:

Case 1: If $TSM\,(p_i, p_j)$ is equal to secret data s

R1. The modified low quantization level a' is equal to p_j and high quantization level b' is equal to p_i.

Case 2: If $TSM\,(p_i, p_j)$ is not equal to secret data s, we use (p_i', p_j') for an alternative target, which satisfies the following four conditions:

R2. The (p_i', p_j') is within a 8×8 size rectangular range, its left, right, upper and lower boundaries are equal to $p_i - 3$, $p_i + 4$, $p_j - 4$ and $p_j + 4$ respectively. And the coordinate value $TSM\,(p_i', p_j')$ is equal to secret data s.

R3. $p_j' \leq p_i'$, both $|p_j' - p_i'|$ and $|p_j - p_i|$ must be classified into the same category before and after data embedding.

R4. Collect all alternative points satisfying the conditions R2 and R3 into a candidate set, and then measure the Euclidean distance between (p_i, p_j) and each candidate, finally, the candidate with the shortest Euclidean distance is used to replace (p_i, p_j).

R5. If the former conditions are all true, we set the flag to 0; else the flag is set to 1.

We experimentally confirmed that the coordinate point that meets the conditions can always be found in an 8×8 rectangle. A detailed explanation and illustration is presented in Section IV. Through the previous process the 8-ary secret data are hidden into the quantization level pair of the compressed code.

In the bitmap embedding phase, the proposed method provides sixteen bits and a 6-bit embedding capacity to the two former categories respectively. In the smooth block, where the two quantization levels are very close to each other, provides a 16-bit payload by replacing the entire bitmap with secret data. For the less complex block, as the two quantization levels have a certain difference, a subtle strategy, namely HC embedding based on a (7, 4) Hamming code was adopted to achieve a 6-bit payload. For the high complex block, we can't do anything on the bitmap, because it is extremely sensitive to modification.

Steps for the embedding algorithm and the detailed algorithm are as follows:

Input: The original image I with the size of $M \times N$, the predefine threshold pair (T_1, T_2), the secret data S.

Output: The AMBTC compressed code (a', b', BM').

Step 1: Read the original image and the matrix of I is denoted as $\{I_{i,j} | 1 \leq i, j \leq M \times N\}$, copy I as the duplicate cover image I'.

Step 2: Use a pseudo-random generator to generate a binary array S, which includes 400000 indices.

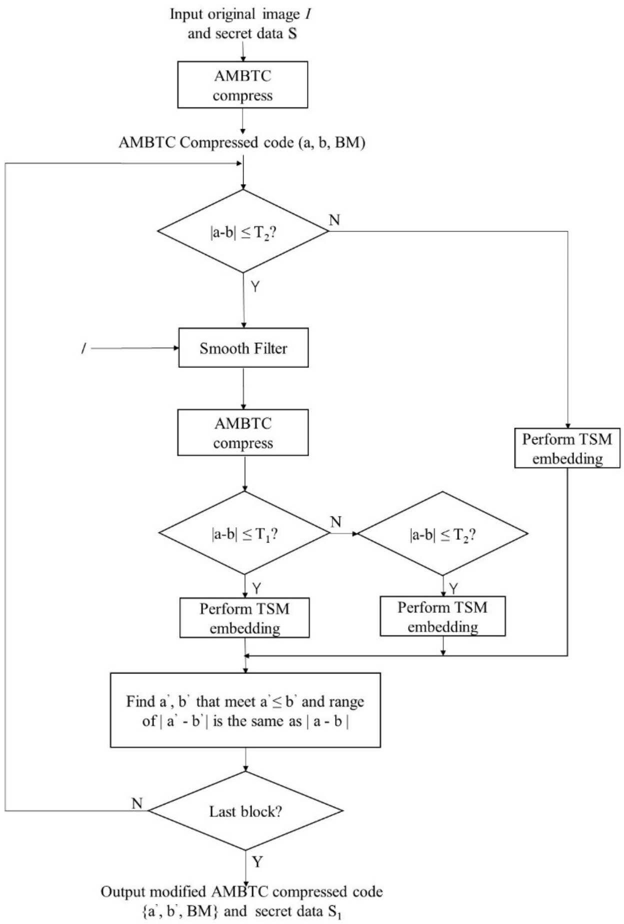
Step 3: Deal with I' using a smooth filter, the detailed process is implemented in the following equation, that is:

$$I'_{i,j} = \frac{4I_{i,j}}{9} + \frac{(I_{i-1,j} + I_{i+1,j} + I_{i,j-1} + I_{i,j+1})}{9} + \frac{I_{i-1,j-1} + I_{i-1,j+1} + I_{i+1,j-1} + I_{i+1,j+1}}{36}, \quad (10)$$

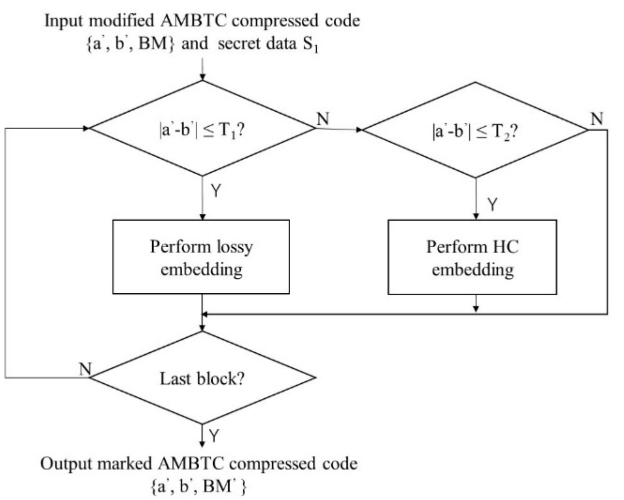
where $2 \leq i \leq W - 1$ and $2 \leq j \leq H - 1$.

Step 4: Divide the cover image I' into a serial of consequent and no-overlapping 4×4 blocks, which are denoted as $\{C_i | 1 \leq i \leq \frac{W \times H}{16}\}$.

Step 5: Compress the block of C_i by the AMBTC compression method, and the trio of the AMBTC compressed code is denoted as (a_i, b_i, BM_i) by Equations (2), (3) and (4).



(a) The preparation and TSM embedding phase



(b) The bitmap embedding

FIGURE 4. The embedding algorithm flowchart include (a) the preparation phase, the TSM embedding phase and (b) the bitmap embedding phase.

Step 6: Classify C_i by the result of $d_i = |a_i - b_i|$ and apply ALGORITHM 1 to the secret data hiding for the AMBTC compressed code. If $d_i \leq T_1$, means C_i is the smooth block,

---

Algorithm 1 The TSM Embedding

Input: The *i-th* AMBTC compressed code (a_i, b_i, BM_i), the predefined threshold pair (T_1, T_2), the turtle shell reference matrix M and the secret data S.

Output: The partial modified AMBTC compressed code (a'_i, b'_i, BM_i).

(1) $len = length(S)$;// Compute the length of the secret data
(2) $s = S(1:3)$;// Read 3bits of the secret data
(3) $S = (4 : len)$;// Update the secret data by subtracting the embedded data
(4) Locate the coordinate (a_i, b_i) in the turtle shell reference matrix M, compute the $y_i = f(a_i, b_i)$;
(5) If $y_i = s$// Indicate that it is the target point and copy them to a'_i and b'_i
(6) $a'_i = a_i$;
(7) $b'_i = b_i$;
(8) Else
(9) Find another coordinate $(a'_i, b'_i$ in the 8×8 rectangle around (a_i, b_i);// The modified coordinate satisfy: the first $f(a'_i, b'_i) == s$, and the second $a'_i \leq b'_i$, and the third both $|a_i - b_i|$ and $|a'_i = b'_i|$ have the same range.
(10) End
(11) Save (a'_i, b'_i, BM_i);// Save and quit the function

---

if $T_1 \leq d_i \leq T_2$, means C_i is the less complex block, else $d_i > T_2$, means C_i is the high complex block. The modified codes satisfy the following rules: (1) the relationship of a modified quantization level pair maintains $a'_i < b'_i$; (2) the classification of the embedding block remains unaltered before and after data embedding; (3) the 3 bits of secret data which is hidden into quantization level pair can be recovered.

Step 7: Record d'_i by a blank matrix and go back to Step 6 until all the blocks are dealt with, and the partial modified AMBTC compressed code $\left(a', b', BM\right)$ is achieved.

Step 8: Choose the corresponding ALGORITHM to implement the second phase of secret data hiding for the modified compressed code. Firstly, hide data into the smooth block in order of priority by d'_i, and use ALGORITHM 2 to deal with the data hiding process. And then, hide into the less complex block in the same way, and use ALGORITHM 3 to deal with the data hiding process. Finally, if it is a high complex block, there is nothing to do, thus skip to the next block.

Step 9: Go back to Step 8 until all blocks have completed the embedding process, and the marked AMBTC compressed code $\left(a', b', BM'\right)$ is achieved.

B. DEMONSTRATION OF EMBEDDING DATA

The first example is illustrated in Fig 5. Assume that the AMBTC compressed code is $(4, 5, BM_i)$ and the secret data is $m = 3$, so the quantization level pair conforms to coordinate point $\left(p_i, p_j\right) = (5, 4)$. We can find the coordinate value in the turtle shell reference by $TSM\,(5, 4) = 7$ which is marked in a green circle. Because the value is not equal to the secret data, find another point for embedding. Draw a

Algorithm 2 The Smooth Block Embedding

Input: The *i-th* AMBTC compressed code (a_i, b_i, BM_i), and the secret data S.

Output: The marked AMBTC compressed code (a_i', b_i', BM_i').

(1) $len = length(S)$;// Compute the length of the secret data
(2) $s = S(1:16)$;// read 16bits secret data
(3) $S = (17:len)$;// Update the secret data by subtracting the embedded data
(4) $BM_i' = reshape(s, [4, 4])$;// Reshape the s_1 by 4×4 matrix
(5) Save (a_i', b_i', BM_i');// Save and quite the function

Algorithm 3 The Less Complex Block Embedding

Input: The *i-th* modified compressed code (a_i, b_i, BM_i), the parity check matrix H and the secret data S.

Output: The marked AMBTC compressed code (a_i', b_i', BM_i').

(1) $len = length(S)$;// Compute the length of the secret data
(2) $s_1 = S(1:3)$; $s_2 = S(4:6)$;// Read the secret data
(3) $S = (7:len)$;// Update the secret data by subtracting the embedded data
(4) $BM_i' = BM_i$;// Coppy B_i to B_i'
(5) $z_1 = s_1 \oplus (H \times BM_i(1:7)^T \%2)$;// Embed s_{11} based on (7,4) Hamming code
(6) $BM_i'(1:7)$ is equal to flip one bit in z_1 position of $BM_i(1:7)$;
(7) $z_2 = s_2 \oplus (H \times BM_i(9:15)^T \%2)$;
(8) $BM_i'(9:15)$ is equal to flip one bit in z_2 position of $BM_i(9:15)$;
(9) Save (a_i', b_i', BM_i);// Save and quit the function

red rectangle with a size of 8×8 based on R2 and find that there are eight candidates. Draw a yellow diagonal in the matrix as a reference line, and all candidates marked in yellow rectangle are eliminated based on R3. The remaining candidates are marked in a brown pentagon, according to R4, and the point $TSM\,(4, 3)$ marked in red circle finally stands out. Finally, the marked AMBTC compressed code is modified to $(3, 4, BM_i)$.

The second example is illustrated in Fig. 6, where the AMBTC compressed code is $(11, 16, BM_1)$, and the absolute difference value of low and high quantization $d_1 = 5$, which is less than the first predefined threshold $T_1 = 8$, so ALGORITHM 2 can be adopted for data hiding. In ALGORITHM 2, the bitmap BM_1 is embedding $s = 1010010100101110$ by lossy embedding. Firstly, the 16 bits of secret data s is intercepted from the binary secret data stream S. And then, the s is transformed into a 4×4 matrix. Finally, the entire bitmap is directly replaced to the matrix and 16 bits of the secret data are embedded into the cover image, and the marked AMBTC compressed code $\left(11, 16, BM_1'\right)$ is achieved.

The third example is illustrated in Fig. 7, where the AMBTC compressed code is $(31, 43, BM_3)$, and the absolute

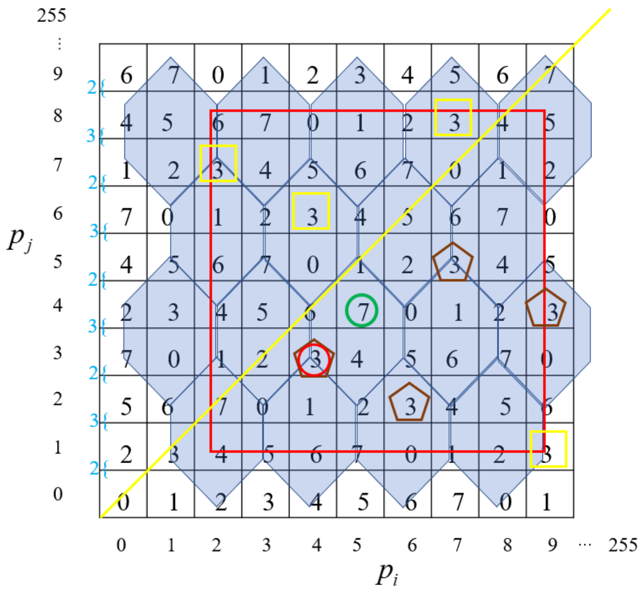


FIGURE 5. The TSM embedding.

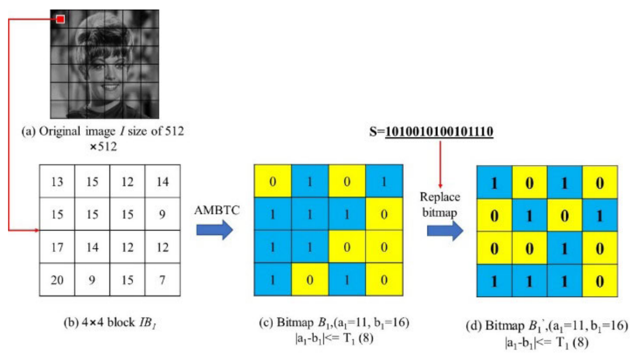


FIGURE 6. The data hiding of smooth block.

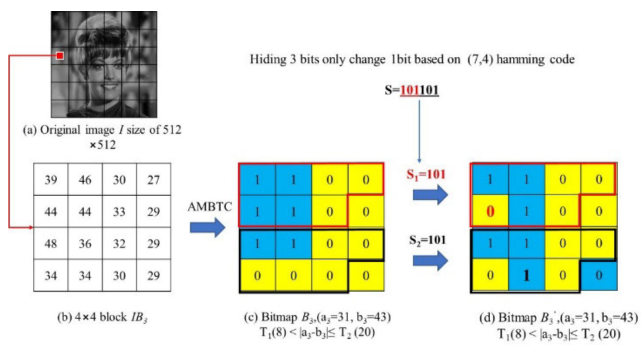


FIGURE 7. The data hiding of less complex block.

difference value of low and high quantization $d_3 = 12$, which is greater than or equal to $T_1 = 8$, but is less than or equal to the second predefined threshold $T_2 = 20$, so ALGORITHM 3 can be adopted for data hiding. In ALGORITHM 3, the bitmap BM_3 is carrying $s = 101101$ by two methods. Firstly, the s is divided into two parts of $s_1 = 101$ and $s_2 = 101$. And then, block B_3 also can be divided into

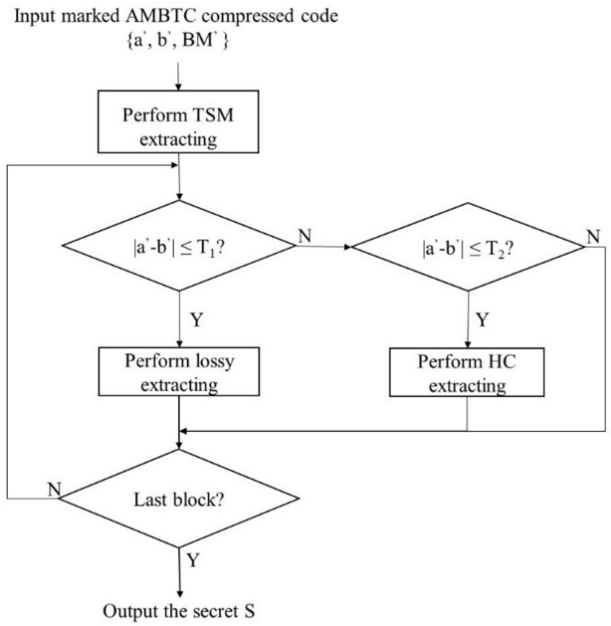


FIGURE 8. The extracting algorithm flowchart.

four parts: $BM_{31} = 1100110$, $BM_{32} = 0$, $BM_{33} = 1100000$ and $BM_{34} = 0$, where BM_{31} and BM_{33} are marked with an irregular shape. They are used to hide s_1 and s_2 based on the (7, 4) Hamming code. BM_{32} or BM_{34} are unchanged. Finally, the new bitmap is computed as $BM'_3 = 1100010011000101$.

C. EXTRACTION ALGORITHM

The extracting phase is even simpler, and can be divided into two phases: extraction from quantization and extraction from bitmap. Firstly, the receiver obtains the marked AMBTC compressed code (a', b', BM') from the Internet. Then the turtle-shell reference matrix can be constructed by public rules. Finally, the 8-ary secret data can be recovered from the coordinate plane by $TSM\ (a', b')$, which can be transformed into binary.

After the first phase of extraction, the next phase can be conducted. Firstly, the receiver computes the absolute difference value of low and high quantization $d_i = |a'_i - b'_i|$ and compares it to the user predefine threshold pair (T_1, T_2). If d_i is less than or equal to T_1, secret data s can be extracted from the bitmap by converting BM'_i to a one-dimensional binary array. Define an empty binary array S to hold the extracted data, concatenate s into S. If d_i is greater than T_1 and less than or equal to T_2, then secret data s_1 and s_2 can be extracted respectively from the bitmap based on the (7, 4) Hamming code. And then, concatenate s_1 and s_2 into S. Repeat the process of the previous until the last block has been extracted. Finally, concatenate the secret data extracted in the former phase and this phase to achieve complete recovery of secret data S.

Input: Marked AMBTC compressed code (a', b', BM'), threshold pair (T_1, T_2)

Output: Secret data S

Step 1: Construct the turtle-shell reference matrix TSM and a blank array S by public rules.

Step 2: Read the AMBTC compressed code (a', b', BM') with a linear scan.

Step 3: Find the 8-ary secret data m in the matrix M by $TSM\ (a', b')$, transform it to binary s and concatenate to array S.

Step 4: Go to Step 2 until the last block has been treated.

Step 5: Calculate each absolute difference of a'_i and b'_i by $d_i = |a'_i - b'_i|$, and compare d_i with (T_1, T_2).

Step 6: If $d_i < T_1$, the secret data hiding in BM'_i is computed by

1) $s = (BM'_i)^T\ (1 : 16)$;
2) $S = S + s$;

Step 7: If $T_1 \leq d \leq T_2$, the secret data hiding in BM'_i is computed by

1) $s_1 = H \times (BM'_i)^T\ (1 : 7)'\ \%2$;
2) $s_2 = H \times (BM'_i)^T\ (9 : 15)'\%2$;
3) $S = S + s_1 + s_2$;

Step 8: Else $d > T_2$, skip and go to Step 5 until the last block has been treated.

Step 9: End

D. DEMONSTRATION OF DATA EXTRACTION

Example 1: Assume

$$C'_1 = \left\{ 11, 18, BM'_1 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix} \right\}$$

is a trio of the AMBTC compressed code, and the predefined threshold pair is $(T_1, T_2) = (8, 20)$. Thus, according to the extraction algorithm, $d_1 = |11 - 8|$ is less than T_1, and hidden data s can be computed by Step 6 of the extraction algorithm. $s = [1010010100101110110]$ can be concatenated into the array S.

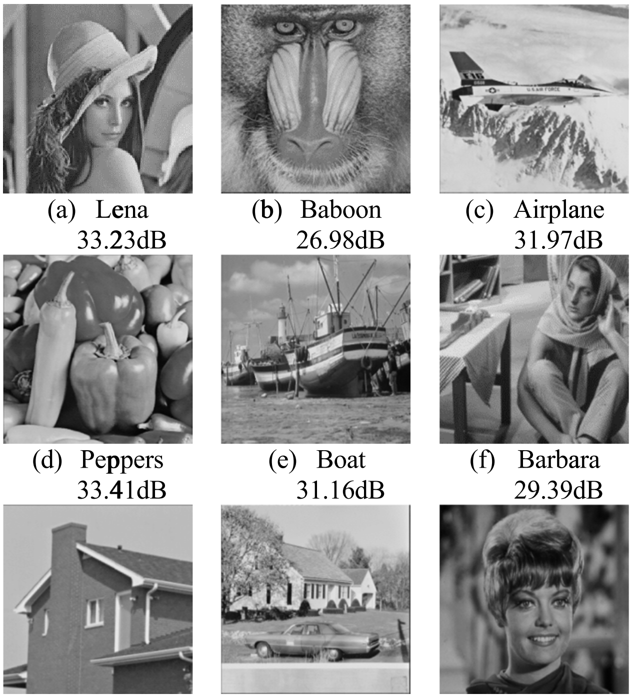
Example 2: Assume that

$$C'_3 = \left\{ 31, 42, BM'_3 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} \right\}$$

is another trio of the AMBTC compressed code, thus according to the extraction algorithm, $d_3 = |31 - 42|$ is greater than T_1, but less than T_2, so the hidden data can be computed by Step 7 of the extraction algorithm. Firstly, compute $s_1 = [101]$ and $s_2 = [101]$ by Steps (1) and (2). And then, concatenate them together to $s = [101101]$. Finally, secret data $s = [101101]$ can be concatenated into array S.

IV. EXPERIMENTAL RESULTS

The section discusses the experimental results of the proposed HBF-DH method and provides a comparison with two representative works, i.e., Ou and Sun [14] and Kumar *et al.* [20]. For a fair and credibly comparison, we used the same nine grayscale images sized 512 × 512 as shown in Fig. 9.



(a) Lena 33.23dB
(b) Baboon 26.98dB
(c) Airplane 31.97dB
(d) Peppers 33.41dB
(e) Boat 31.16dB
(f) Barbara 29.39dB
(g) House 36.59dB
(h) Houses 30.89dB
(i) Zelda 36.72dB

FIGURE 9. The AMBTC decompressed images and their PSNRs.

The experiment platform was MATLAB R2017a running on an Intel® CORE i7 8th Gen and 16 G RAM workstation. Both hiding capacity for secret data and the visual quality of a stego image were the major evaluation metrics. The first metric can be measured by the amount of data embedded into the cover image. The second metric can be computed by the following Equation:

$$MSE = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} \left(I_{ij} - I_{ij}^{*} \right)^{2}, \tag{11}$$

$$PSNR = 10 \log \frac{255^{2}}{MSE} (dB), \tag{12}$$

where H and W represent the height and width of the image, I_{ij} represents the i-th row and j-th column pixel of the original image, I_{ij}^{*} represents the i-th row and j-th column pixel of the stego image and MSE represents the degree of difference between the original image and stego image. The peak signal-to-noise ratio, namely PSNR, can be computed by Equation (12), which indicates the visual quality of the stego image.

The user predefined threshold pair (T_1, T_2), which we discussed in Section III, helps to classify the AMBTC compressed code into three categories: smooth block, less complex block, and high complex block. The experimental results indicate that the data hiding capacity and the image visual quality can be efficiently affected by thresholds. This occurs because with increasing thresholds, the number of smooth blocks also increases, which provides more data hiding capacity, while in contrast, the visual quality of the image decreases. We presented PSNRs of nine images in Fig. 9. Then, performance on embedding capacity and visual quality at various thresholds are presented in Table 1 to confirm our conjecture. From another perspective, we can utilize this characteristic to expand the application scenario of our method by adjusting the value of a threshold pair.

Table 1 shows the performance comparison for our method along with two popular data hiding methods based on an AMBTC compressed code and block classification in terms of embedding capacity Cap and PSNR. From Table 1, we can see that our HBF-DH method achieves a higher performance than methods of Ou and Sun [14] and Kumar *et al.* [20] under different values for the threshold pair (T_1, T_2). Taking "Lena" for example, when the threshold pair is set as $(T_1, T_2) = (8, 20)$, our method achieves higher hiding capacity and PSNRs compared with the methods of Kumar et al. and Ou and Sun. In detail, our gain over the Ou and Sun's method is 38,253 bits under the roughly same PSNR. Compared with the Kumar et al.'s method, our method achieves a larger capacity and higher PSNR.

For the highly-textured image such as "Baboon," our HBF-DH method also achieves a higher embedding performance than the two compared methods. From Table 1, at almost the same PSNR, the Cap achieved by the Ou and Sun's method is only 108,499 bits, while our Cap is 120,558 bits. In addition, our method is comparable to the Kumar et al.'s method.

Since the embedding capacity and visual quanlity of image is a trade-off, to compare the performance in terms of visual quality between the proposed HBF-DH method and two representative works, we choose six typical test grayscale images: (a) Lena, (b) Baboon, (c) Peppers, (d) Boat, (e) Airplane and (f) Barbara and conducted an experiment under the same embedding capacity. The experimental results are plotted on the coordinate plane, and the graph is shown in Fig. 10, in which the x-axis represents embedding capability, the y-axis represents image quality, and the points on the coordinate plane represent a set of experimental data. Our results are represented by a histogram, and others are represented by a line chart with data marks. Ou and Sun's [14] results are represented by a green curve with triangle marks and Kumar et al.'s [20] results are represented by an orange curve with rectangle marks.

Our overall performance is superior among all the methods as shown in Figs. 10 (a)-(f). Firstly, the PSNR of ours are the highest of all under the same embedding capacities. Moreover, the decline is relatively flat, while the other two methods have a clear turning point at 240000. Finally, our HBF-DH method does well with smooth images, as the PSNR is more than 30 dB as shown in Figs. 10 (a) and (c), even though the capacity reaches the maximum of the other method. Note also that the Ou and Sun's method [14] is always better than the Kumar et al.'s method [20] on smooth images, but is the reverse on the image of "Baboon" when the capacity is 190000. This means that both the turtle-shell reference matrix

TABLE 1. Comparisons of Cap and PSNR in different threshold combination between ours and existing two related works [14], [20].

Methods	Metrics	Lena	Baboon	Airplane	Peppers	Boats	Barbara	House	Houses	Zelda
(T_1,T_2)=(8,20)										
Ours	PSNR	31.71	26.59	31.06	31.56	29.92	28.80	34.30	30.09	33.49
	Cap	236,956	120,558	234,058	244,516	204,814	179,100	268,180	191,956	257,482
Ou et al. [14]	PSNR	31.45	26.56	30.86	31.53	29.86	28.80	34.24	30.01	33.48
	Cap	220,174	108,499	209,524	228,559	196,039	160,354	241,834	116,719	242,854
Kumar et al. [20]	PSNR	30.48	25.47	29.53	30.36	28.60	27.55	33.25	28.07	32.77
	Cap	206,070	125,862	206,022	211,665	182,025	168,584	229,130	175,750	219,501
(T_1,T_2)=(16,28)										
Ours	PSNR	30.50	25.89	30.22	30.53	28.72	27.93	33.15	28.92	31.85
	Cap	267,706	166,232	257,380	272,050	248,844	215,256	284,764	230,638	289,636
Ou et al. [14]	PSNR	30.46	25.83	30.15	30.38	28.67	27.89	33.15	28.92	31.83
	Cap	238,879	150,004	228,394	243,379	228,109	193,159	250,834	208,594	259,219
Kumar et al. [20]	PSNR	30.01	24.85	29.19	30.01	28.15	26.90	32.83	27.54	32.27
	Cap	230,534	162,367	224,446	234,240	216,705	195,198	243,112	204,956	245,090
(T_1,T_2)=(24,36)										
Ours	PSNR	29.32	24.89	29.29	29.63	27.69	26.89	31.93	27.52	30.69
	Cap	283,636	200,416	270,372	284,066	268,902	239,312	294,452	257,090	301,506
Ou et al. [14]	PSNR	29.18	24.81	28.91	29.61	27.63	26.76	31.83	27.50	30.99
	Cap	251,284	185,104	237,229	250,219	242,449	215,509	256,894	233,644	262,144
Kumar et al. [20]	PSNR	29.37	24.01	28.76	29.68	27.70	26.04	32.43	26.89	31.78
	Cap	242,177	189,479	234,194	243,361	232,144	215,180	249,949	223,941	254,398
(T_1,T_2)=(32,40)										
Ours	PSNR	28.21	23.74	28.26	28.83	26.80	25.76	30.92	26.28	29.89
	Cap	294,040	229,968	280,066	291,210	280,790	258,092	300,036	275,110	306,886
Ou et al. [14]	PSNR	28.20	23.66	28.24	28.80	26.78	25.63	30.92	26.25	30.99
	Cap	259,159	215,674	246,319	254,914	250,564	232,804	259,534	249,139	262,144
Kumar et al. [20]	PSNR	28.86	23.07	28.28	29.32	27.04	25.16	31.94	26.32	31.52
	Cap	249,745	213,528	240,899	248,516	241,223	230,581	254,160	236,958	258,676
(T_1,T_2)=(40,52)										
Ours	PSNR	27.41	22.63	27.36	28.08	25.98	24.81	30.14	25.30	29.48
	Cap	300,082	255,244	286,896	296,110	289,542	271,960	303,358	286,954	309,086
Ou et al. [14]	PSNR	27.41	22.56	27.35	28.03	25.94	24.78	29.64	25.27	30.99
	Cap	261,829	239,899	251,644	257,929	257,419	245,059	260,449	256,189	262,144
Kumar et al. [20]	PSNR	28.51	22.38	27.69	28.84	26.63	24.27	31.38	25.88	31.32
	Cap	254,209	232,590	245,752	252,089	247,601	243,059	256,640	245,663	260,428

and the pixel value difference PVD algorithm have certain advantages for a complex image.

To test the impact of our designed smooth filter and Kumar et al.'s smooth filter, we also disabled two smooth filters while leaving the other conditions unchanged for both our HBF-DH method and Kumar et al.'s method. The comparative experimental data are shown in Table 2. Take average PSNR and Cap of ours under threshold pair (8, 20) for example, value (−0.07, 12841) indicates the average PSNR of our HBF-DH method is less 0.07 dB with smooth filter than that without smooth filter, but Cap is more than 12841 bits with smooth filter than that without smooth filter. The smooth filter improves the performance both in terms of Cap, especially when the value of a threshold pair is small. With a growth of thresholds, the influence declines.

To further demonstrate the performance of our HBF-DH method, finally, we compared the PSNR and Cap between the proposed method and other four existing schemes with different combinations of the thresholds as shown in Table 3. The experimental results further confirm the upper bound of embedding capacity offered by our HBF-DH method is significantly higher than those of other four existing schemes. In other words, the embedding capacity of our proposed HBF-DH method is adaptive and it is suitable to support various applications.

Consider TSM is our core technique to achieve high embedding capacity, two cases and four rules must be workable for various images and remain the characteristics of the AMBTC compressed code after data embedding. Finally, we tested the TSM embedding algorithm on 10,000 images selected from the BOSSBase database [43]. We use the amunt of accident point to count how many the point of a coordinate plane not satisfy our rules when TSM embedding algorithm is applied. The results of statistics depicted in Fig. 11 indicate that therei s no error for all images. As such, the TSM method is suitable for hiding data in a quantization level pair of the AMBTC compression code, which increases the embedding capacity with a slight distortion, while

TABLE 2. Comparing between enable and disable the smooth filter in terms of the average data hiding capacity and image quality.

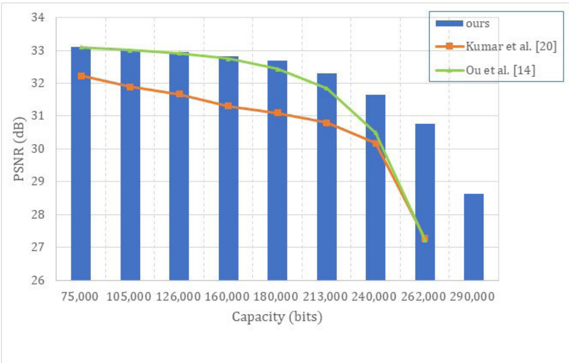
THRESHOLD		thr1=8 thr2=20	thr1=16 thr2=28	thr1=24 thr2=32	thr1=32 thr2=44	thr1=40 thr2=52
Ours	PSNR	-0.07	-0.06	-0.11	-0.14	-0.07
	CAP	12841	8148	7359	7164	6258
Kumar et al. [20]	PSNR	0.45	1.06	0.02	-0.1	-0.2
	CAP	8539	5503	5702	6271	6150

TABLE 3. The comparison of Cap and PSNR between the proposed scheme and other works [13]–[15], [20] with different combinations of thresholds.

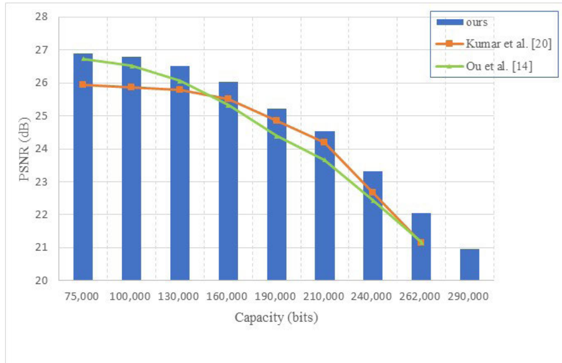
Methods	Metrics	Lena	Baboon	Airplane	Peppers	Boats	Barbara	House	Houses	Zelda
(T1,T2)=(10,25)										
ours	PSNR	31.28	26.37	30.77	31.18	29.50	28.49	33.92	29.67	32.87
	Cap	249,356	138,652	243,502	256,462	222,554	193,564	274,808	207,484	271,950
Chuang et al. [13]	PSNR	32.03	26.85	31.54	32.37	31.04	29.01	33.11	28.78	32.98
	Cap	166,608	36,256	172,496	178,288	141,040	112,400	175,436	128,416	167,849
Ou et al. [14]	PSNR	32.63	26.91	31.12	32.58	30.75	29.20	35.57	30.52	35.07
	Cap	176,659	54,214	179,179	180,829	132,664	124,324	215,329	131,944	200,599
Chen et al. [15]	PSNR	32.70	26.90	31.70	32.50	30.10	33.20	34.10	30.70	36.30
	Cap	177,890	90,568	182,106	178,526	98,574	159,671	154,260	145,230	215,288
kumar et al. [20]	PSNR	30.65	26.33	29.54	30.58	28.91	28.27	32.94	27.96	33.05
	Cap	216,042	140,786	213,334	221,273	195,009	178,880	235,082	187,238	231,149
(T1,T2)=(20,35)										
ours	PSNR	29.76	25.23	29.65	29.99	28.07	27.24	32.41	28.05	31.11
	Cap	278,416	190,262	265,876	279,628	262,344	232,024	291,192	248,032	297,646
Chuang et al. [13]	PSNR	30.43	26.11	30.39	30.78	29.63	28.19	32.45	28.07	31.22
	Cap	213,488	96,160	206,304	222,512	187,232	151,904	209,864	190,254	217,009
Ou et al. [14]	PSNR	31.75	26.56	30.86	31.73	29.86	28.80	34.62	30.01	33.73
	Cap	217,594	108,499	209,524	223,444	196,039	160,354	237,829	176,719	238,984
Chen et al. [15]	PSNR	31.75	26.52	31.12	31.62	28.41	31.12	34.12	30.10	34.86
	Cap	225,640	137,654	238,456	226,592	178,957	207,456	174,554	162,548	248,455
kumar et al. [20]	PSNR	30.62	26.20	29.47	30.66	28.97	28.12	32.85	27.93	33.04
	Cap	230,534	162,367	224,446	234,240	216,705	195,198	243,112	204,956	245,090
(T1,T2)=(30,45)										
ours	PSNR	28.37	23.72	28.46	28.94	26.92	25.81	31.08	26.42	29.99
	Cap	292,854	231,116	278,770	290,386	279,692	257,728	299,266	273,416	306,324
Chuang et al. [13]	PSNR	29.06	25.04	29.34	29.58	28.05	26.83	31.04	27.45	30.83
	Cap	232,128	133,904	220,432	237,424	214,272	182,928	227,896	216,504	231,040
Ou et al. [14]	PSNR	30.78	25.98	30.51	31.07	29.09	27.98	33.67	28.94	32.67
	Cap	234,604	143,419	223,024	236,524	219,634	188,929	247,249	206,254	252,724
Chen et al. [15]	PSNR	30.90	25.96	30.53	30.99	27.51	30.61	33.30	28.99	33.76
	Cap	236,421	152,834	218,954	238,429	217,419	232,421	191,744	190,824	257,940
kumar et al. [20]	PSNR	30.10	25.68	29.12	30.34	28.64	27.61	32.50	27.48	32.63
	Cap	242,177	189,479	234,194	243,361	232,144	215,180	249,949	223,941	254,398
(T1,T2)=(40,55)										
ours	PSNR	27.30	22.30	27.26	28.02	25.84	24.52	30.08	25.17	29.43
	Cap	300,906	262,842	287,870	296,666	290,980	276,370	303,634	288,514	309,268
Chuang et al. [13]	PSNR	27.78	23.58	28.06	28.50	26.76	25.47	30.16	26.78	29.46
	Cap	244,464	168,096	231,408	246,224	229,504	204,288	240,765	234,859	239,429
Ou et al. [14]	PSNR	29.94	25.08	29.80	30.27	28.37	27.19	32.71	27.98	31.99
	Cap	245,959	175,429	232,864	244,069	232,264	208,684	252,694	226,279	258,154
Chen et al. [15]	PSNR	29.99	25.12	29.81	30.32	26.95	29.51	32.39	27.99	33.12
	Cap	246,802	184,972	229,892	246,204	231,624	246,589	189,810	210,329	260,699
kumar et al. [20]	PSNR	29.61	24.89	28.73	30.02	28.03	26.96	32.07	26.98	32.39
	Cap	249,745	213,528	240,899	248,516	241,223	230,581	254,160	236,958	258,676

retaining the characteristics of the AMBTC compressed code. In other words, the modified AMBTC-compression code by our proposed HBF-DH still can be correctly decoded using the conventional AMBTC decoder. It means the security of hidden data can be guaranteed by our proposed method.

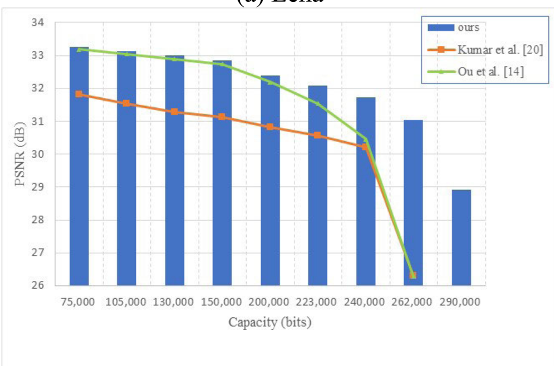
To veritify the security of our method the statistical RS-steganalysis method [44] is adopted, which uses a discrimination function (DF) with two matrix [0,1;1,0] and $[0, -1; -1, 0]$ as parameters M and $-$M respectively. Futhermore, four results as R_M, R_{-M}, S_M and S_{-M} are calculated using DF function to find the magnitude of steganographic.



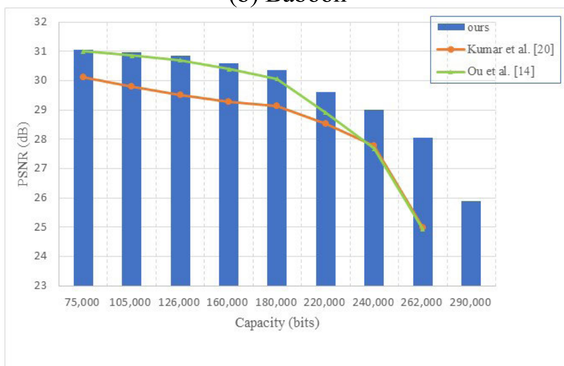
(a) Lena



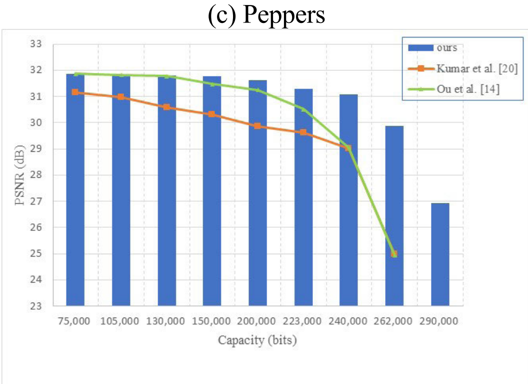
(b) Baboon



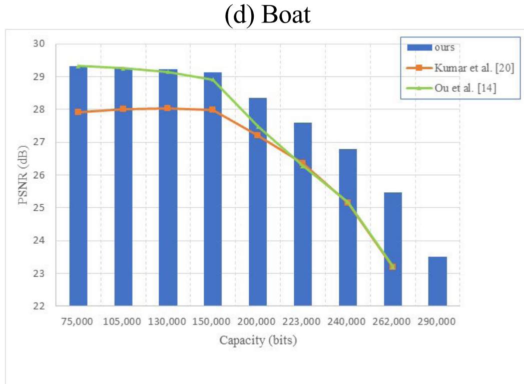
(c) Peppers



(d) Boat



(e) Airplane



(f) Barbara

FIGURE 10. **The comparison of PSNR under the same Cap between ours and the existing related works for vary images. (a) Lena, (b) Baboon, (c) Peppers, (d) Boat, (e)Airplane and (f) Barbara.**

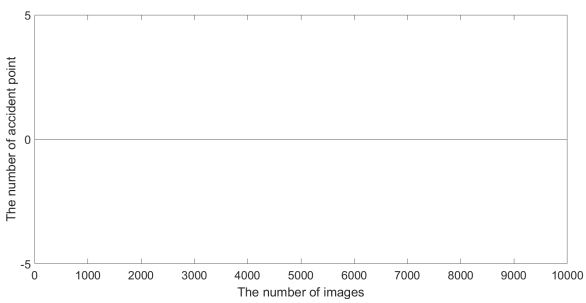


FIGURE 11. **The distributing of accident point of 10000 images.**

If they satisfy $R_M \approx R_{-M} > S_M \approx S_{-M}$, there are no hidden data in the detected image. Vice vas, when an image has hidden data in its least significant bits, the results are sinificant different and is exposed by RS-steganalysis.

TABLE 4. **The RS-steganalysis detection of nine marked images.**

Images	Proposed HBF-DH method	
Lena	0.0329	0.0018
Baboon	0.0262	0.0005
Airplane	0.0229	0.0014
Boat	0.0006	0.0059
Barbara	0.0248	0.0035
House	0.0377	0.0039
Houses	0.0215	0.0041
Zelda	0.0039	0.0035
Average	0.0213	0.003

Firstly, the HBF-DH method is applied under maximum payload to the nine image shown in Fig. 9. And then, the compressed stego image is decoded by AMBTC technique. Finally, the RS-steganalysis method is used to those stego images. The experimental results indicate that the RS-steganalysis detection difference results of the our method are extremely close to each other between R_M and R_{-M} and between S_M and S_{-M}. The average value of $|R_M - R_{-M}|$ and $|S_M - S_{-M}|$ is equal to 0.0213 and 0.003 respectivly. In other word, our method can resist against the RS-steganalysis detection, the detailed results shown in Table 4.

V. CONCLUSION

In this paper, we proposed a novel HBF-DH method based on classification of the AMBTC compressed code. By comparing the absolute difference of two quantization levels to a user predefined threshold pair, a trio of AMBTC compressed image can be divided into three categories: smooth block, less complex block and high complex block. First, we embedded three bits of secret data into two quantization levels with a turtle-shell reference matrix. And then based on the characteristics of each category, we designed different data hiding strategies to deal with the processes for data hiding and extraction. For a smooth block, the bitmap is directly replaced by binary secret data. For a less complex block, the modification is more detailed to reduce distortion. Thus, data hiding strategy based on the (7, 4) Hamming code was designed to provide more capacity while maintaining visual image quality image. Finally, the proposed HBF-DH method maintains a comparable compression ratio and characteristics of the AMBTC compressed code in an up-to-date data hiding method based on an AMBTC compressed image. The experimental results and analysis indicate that our proposed scheme is superior to four existing representative DH methods in terms of data hiding ability and also for image visual quality.

REFERENCES

[1] C.-C. Chang, T. D. Kieu, and Y.-C. Chou, "Reversible data hiding scheme using two steganographic images," in *Proc. IEEE Region Conf.*, Taipei, Taiwan, Oct./Nov. 2007, pp. 1–4.

[2] S. Arjun, A. Negi, C. Kranthi, and D. Keerthi, "An approach to adaptive steganography based on matrix embedding," in *Proc. IEEE Region Conf.*, Taipei, Taiwan, Oct. /Nov. 2007, pp. 1–4.

[3] C.-C. Chang, T. D. Kieu, and Y.-C. Chou, "Using nearest covering codes to embed secret information in grayscale images," in *Proc. 2nd Int. Conf. Ubiquitous Inf. Manage. Commun.*, Suwon, Korea, 2008, pp. 315–320.

[4] C.-C. Chang, T. D. Kieu, and Y.-C. Chou, "A high payload steganographic scheme based on (7, 4) Hamming code for digital images," in *Proc. Int. Symp. Electron. Commerce Secur.*, Guangzhou, China, Aug. 2008, pp. 16–21.

[5] J. Bai and C.-C. Chang, "A high payload steganographic scheme for compressed images with Hamming code," *Int. J. Netw. Secur.*, vol. 18, no. 6, pp. 1122–1129, Jan. 2016.

[6] C.-C. Chang, T.-S. Chen, and L.-Z. Chung, "A steganographic method based upon JPEG and quantization table modification," *Inf. Sci.*, vol. 141, nos. 1–2, pp. 123–138, Mar. 2002.

[7] C.-C. Chang, T. S. Nguyen, and C.-C. Lin, "A novel VQ-based reversible data hiding scheme by using hybrid encoding strategies," *J. Syst. Softw.*, vol. 86, no. 2, pp. 389–402, Feb. 2013.

[8] E. Delp and O. Mitchell, "Image compression using block truncation coding," *IEEE Trans. Commun.*, vol. COM-27, no. 9, pp. 1335–1342, Sep. 1979.

[9] O. R. Mitchell and E. J. Delp, "Multilevel graphics representation using block truncation coding," *Proc. IEEE*, vol. 68, no. 7, pp. 868–873, Jul. 1980.

[10] C.-C. Chang, C.-Y. Lin, and Y.-H. Fan, "Lossless data hiding for color images based on block truncation coding," *Pattern Recognit.*, vol. 41, no. 7, pp. 2347–2357, 2008.

[11] X.-Z. Xie, C.-C. Chang, C.-C. Lin, and J.-L. Lin, "A Turtle Shell based RDH scheme with two-dimensional histogram shifting," *Multimedia Tools Appl.*, vol. 78, no. 14, pp. 19413–19436, Jul. 2019.

[12] M. Lema and O. Mitchell, "Absolute moment block truncation coding and its application to color images," *IEEE Trans. Commun.*, vol. COM-32, no. 10, pp. 1148–1157, Oct. 1984.

[13] J.-C. Chuang and C.-C. Chang, "Using a simple and fast image compression algorithm to hide secret information," *Int. J. Comput. Appl.*, vol. 28, no. 4, pp. 329–333, 2006.

[14] D. Ou and W. Sun, "High payload image steganography with minimum distortion based on absolute moment block truncation coding," *Multimedia Tools Appl.*, vol. 74, no. 21, pp. 9117–9139, Nov. 2015.

[15] Y.-Y. Chen and K.-Y. Chi, "Cloud image watermarking: High quality data hiding and blind decoding scheme based on block truncation coding," *Multimedia Syst.*, pp. 1–13, Jul. 2017.

[16] Y.-H. Huang, C.-C. Chun, and Y.-H. Chen, "Hybrid secret hiding schemes based on absolute moment block truncation coding," *Multimedia Tools Appl.*, vol. 76, no. 5, pp. 6159–6174, Mar. 2017.

[17] A. Malik, G. Sikka, and H. K. Verma, "A high payload data hiding scheme based on modified AMBTC technique," *Multimedia Tools Appl.*, vol. 76, no. 12, pp. 14151–14167, Jun. 2017.

[18] W. Hong, T. S. Chen, Z. Yin, B. Luo, and Y. Ma, "Data hiding in AMBTC images using quantization level modification and perturbation technique," *Multimedia Tools Appl.*, vol. 76, no. 3, pp. 3761–3782, Feb. 2017.

[19] W. Hong, "Efficient data hiding based on block truncation coding using pixel pair matching technique," *Symmetry*, vol. 10, no. 2, p. 36, Feb. 2018.

[20] R. Kumar, D.-S. Kim, and K.-H. Jung, "Enhanced AMBTC based data hiding method using Hamming distance and pixel value differencing," *J. Inf. Secur. Appl.*, vol. 47, pp. 94–103, Aug. 2019.

[21] A. Malik, S. Geeta, and H. K. Verma, "A high capacity data hiding scheme using modified AMBTC compression technique," *Int. Arab J. Inf. Technol.*, vol. 16, no. 1, pp. 148–155, Jan. 2019.

[22] Y.-Y. Chen, C.-H. Hsia, K.-Y. Chi, and B.-Y. Chen, "High-quality and high-capacity data hiding based on absolute moment block truncation coding," *J. Internet Technol.*, vol. 20, no. 2, pp. 379–387, 2019.

[23] C. Kim, D. Shin, and C.-N. Yang, "High capacity data hiding based on AMBTC and interpolation," in *Proc. Int. Conf. Secur. With Intell. Comput. Big-Data Service*, Springer, Cham, Switzerland, 2018, pp. 858–867.

[24] M. Tang, S. Zeng, X. Chen, J. Hu, and Y. Du, "An adaptive image steganography using AMBTC compression and interpolation technique," *Optik*, vol. 127, no. 1, pp. 471–477, Jan. 2016.

[25] A. Malik, G. Sikka, and H. K. Verma, "An AMBTC compression based data hiding scheme using pixel value adjusting strategy," *Multidimensional Syst. Signal Process.*, vol. 29, no. 4, pp. 1801–1818, Oct. 2018.

[26] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J.*, vol. 29, no. 2, pp. 147–160, Apr. 1950.

[27] C.-C. Wang, Y.-F. Chang, C.-C. Chang, J.-K. Jan, and C.-C. Lin, "A high capacity data hiding scheme for binary images based on block patterns," *J. Syst. Softw.*, vol. 93, pp. 152–162, Jul. 2014.

[28] M.-N. Wu, C.-C. Lin, and C.-C. Chang, "An embedding technique based upon block prediction," *J. Syst. Softw.*, vol. 81, no. 9, pp. 1505–1516, Sep. 2008.

[29] M. N. Wu, M.-H. Lin, and C.-C. Chang, "A LSB substitution oriented image hiding strategy using genetic algorithms," in *Proc. Adv. Workshop Content Comput.*, ZhenJiang, China, 2004, pp. 219–229.

[30] Y.-C. Hu, M.-H. Lin, and J.-H. Jiang, "A novel color image hiding scheme using block truncation coding," *Fundamenta Informaticae*, vol. 70, no. 4, pp. 317–331, May 2006.

[31] C.-K. Chan and L. M. Cheng, "Hiding data in images by simple LSB substitution," *Pattern Recognit.*, vol. 37, no. 3, pp. 469–474, Mar. 2004.

[32] C.-C. Chang, T. S. Nguyen, and C.-C. Lin, "Reversible data embedding for indices based on histogram analysis," *J. Vis. Commun. Image Represent.*, vol. 25, no. 7, pp. 1704–1716, Oct. 2014.

[33] C.-C. Chang, T.-S. Nguyen, M.-C. Lin, and C.-C. Lin, "A novel data-hiding and compression scheme based on block classification of SMVQ indices," *Digit. Signal Process.*, vol. 51, pp. 142–155, Apr. 2016.

[34] C.-C. Chang, C.-C. Lin, C.-S. Tseng, and W.-L. Tai, "Reversible hiding in DCT-based compressed images," *Inf. Sci.*, vol. 177, no. 13, pp. 2768–2786, 2007.

[35] C.-C. Chang and T.-C. Lu, "A difference expansion oriented data hiding scheme for restoring the original host images," *J. Syst. Softw.*, vol. 79, no. 12, pp. 1754–1766, Dec. 2006.
[36] T.-H. Lan and A. H. Tewfik, "A novel high-capacity data-embedding system," *IEEE Trans. Image Process.*, vol. 15, no. 8, pp. 2431–2440, Aug. 2006.
[37] M. Jo and H. D. Kim, "A digital image watermarking scheme based on vector quantisation," *IEICE Trans. Inf. Syst.*, vol. 85, no. 6, pp. 1054–1056, Jun. 2002.
[38] E. Ganic and A. M. Eskicioglu, "Robust DWT-SVD domain image watermarking: Embedding data in all frequencies," in *Proc. Workshop Multimedia Secur.*, Magdeburg, Germany, 2004, pp. 166–174.
[39] M. Ramkumar, A. N. Akansu, and A. A. Alatan, "A robust data hiding scheme for images using DFT," in *Proc. Int. Conf. Image Process.*, Kobe, Japan, Oct. 1999, pp. 211–215.
[40] J. Abraham and V. Paul, "An imperceptible spatial domain color image watermarking scheme," *J. King Saud Univ.—Comput. Inf. Sci.*, vol. 31, no. 1, pp. 125–133, Jan. 2019.
[41] C.-C. Lin, X.-L. Liu, W.-L. Tai, and S.-M. Yuan, "A novel reversible data hiding scheme based on AMBTC compression technique," *Multimedia Tools Appl.*, vol. 74, no. 11, pp. 3823–3842, 2015.
[42] C.-C. Lin, C.-C. Chang, and Z.-M. Wang, "Reversible data hiding scheme using adaptive block truncation coding based on an edge-based quantization approach," *Symmetry*, vol. 11, no. 6, p. 765, Jun. 2019.
[43] P. Bas, T. Filler, and T. Pevný, "'Break our steganographic system': The Ins and outs of organizing BOSS," in *Proc. Int. Workshop Inf. Hiding*, Prague, Czech Republic, 2011, pp. 59–70.
[44] J. Fridrich, M. Goljan, and R. Du, "Reliable detection of LSB steganography in color and grayscale images," in *Proc. Workshop Multimedia Secur. New Challenge*, Ottawa, ON, Canada, 2001, pp. 27–30.

ZHAN YU received the B.S. degree from the Lanzhou University of Technology, in 2004, and the M.S. degree in computer and technology from Fuzhou University, in 2017. He has chaired and participated in the research of several provincial projects. He is currently a Lecturer with the School of Electronics and Information Engineering, Fuqing Branch of Fujian Normal University. His research interests include the Internet of Things, image and signal processing, and computer network technology.

CHIA-CHEN LIN (MIN-HUI LIN) received the Ph.D. degree in information management from National Chiao Tung University, in 1998. From 2009 to 2012, she served as the Vice Chairman for the Tainan Chapter IEEE Signal Processing Society. Since 2018, she has been the School Counselor for Providence University, where she is currently a Professor with the Department of Computer Science and Information Management. Her research interests include image and signal processing, information hiding, mobile agent, and electronic commerce. Since 2018, she has been a Fellow of IET. She serves as an Associate Editor and an Editor for several representative EI and SCIE journals.

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

GUO-DONG SU received the B.S. degree from Quanzhou Normal University, in 2011, and the M.S. degree in communication and information system from Fujian Normal University, in 2014. He is currently pursuing the Ph.D. degree in information engineering and computer science with Feng Chia University, Taichung, Taiwan. His research interests include multimedia security, image processing, and digital forensics.