Multimedia Systems (2019) 25:551–563
https://doi.org/10.1007/s00530-017-0560-y

SPECIAL ISSUE PAPER

Cloud image watermarking: high quality data hiding and blind decoding scheme based on block truncation coding

Yung-Yao Chen[1] · **Kuan-Yu Chi**[1]

Published online: 8 July 2017

Abstract Cloud computing is an Internet-based computing model that shares computing resources such as computers, servers, and storage. In future smart cities, new applications can take advantage of cloud computing; for example, a cloud-based home health-care system can provide immediate disease management by analyzing medical images. However, if the size of cloud data is large (e.g., medical images), their transition through the Internet is time consuming. Cloud data authentication is another problem that should be addressed. To address the aforementioned problems, this study presents a novel data hiding method based on the block truncation coding (BTC) image compression technique. This method has two advantages: reduces image size and increases security. This study also proposes a block classification scheme for determining smooth blocks, complex_1 blocks, and complex_2 blocks in an image. Secret data are embedded in these three block types using different approaches. To improve the quality of images without damaging the secret data, we propose a progressive data hiding strategy and integrating an iteration-based halftoning method into BTC. When decoding, the secret data are extracted without recalling the original image. According to experimental results, the proposed method outperforms existing methods in both embedding capacity and image quality evaluations.

✉ Yung-Yao Chen
yungyaochen@ntut.edu.tw

[1] Graduate Institute of Automation Technology, National Taipei University of Technology, No.1, Sec. 3, Zhongxiao E. Rd., Daan Dist., Taipei 106, Taiwan, ROC

Keywords Cloud data security · Data hiding · Blind decoding · Block truncation coding (BTC) · Direct binary search (DBS)

1 Introduction

Cloud computing, a type of on-demand computing that shares computer processing resources over the Internet, plays a major role in smart cities. Through the use of cloud computing, services can be scaled to the dynamic usage needs and the cloud data are safely backed up constantly. Many cloud services have thus been developed to make life more convenient. For example, by analyzing a patient's medical records and prompt medical images, a cloud-based home health care system can provide disease management online immediately. However, because the cloud data are easily accessible from any web-connected device, cloud data protection is essential for the security concern that ensures authentication and trustworthiness of the cloud data. Therefore, numerous studies have been conducted for improving cloud computing security [1–3].

Although cloud computing substantially reduces the processing time, downloading (or uploading) data to cloud services is a time-consuming process. However, digital images have a high level of redundancy. Therefore, image compression is essential in cloud computing because by reducing irrelevancy and redundancy of images, we can transmit or store images more efficiently. For example, the size of medical images is typically large; therefore, they should be compressed and stored in a cloud storage system [4].

Hiding data in digital signals is commonly referred to as *digital watermarking* [5–7]. With the development of the Internet, digitized multimedia data are easily distributed. Digital images are one of the most common formats of

multimedia data. However, distributing sensitive or private images, such as private medical images or military documents, through a public network or an accessible cloud service increases the possibility of counterfeit or illegal copying. Moreover, as mentioned, compressing image size is crucial to promoting the popularity of cloud services.

Generally, digital images that are selected to be carrier signals are referred to as *cover images*, and images that have embedded secret data are referred to as *stego images*. Most of the existing data hiding methods can be classified into two categories: (1) frequency-domain [8–10] and (2) spatial-domain [11–13] methods.

For the frequency-domain data hiding methods, the cover images are transformed into the frequency domain. Therefore, data hiding is achieved by tuning or modifying the selected frequency components. Chen and Lin [8] proposed a steganography method in which secret data are embedded in high-frequency coefficients, and low-frequency coefficients are unchanged to maintain image quality. The discrete wavelet transform (DWT) is used to transform image data into the frequency domain. Shejul and Kulkarni [9] proposed a data hiding method that is based on biometrics and DWT. In the method in [9], secret data are embedded in the high-frequency DWT subband within the skin region of an image. In [10], a block-based DCT coefficient modification was proposed for blind watermarking. The methods in this category are robust to attacks; however, they typically involve very high computational complexity.

For the spatial-domain data hiding methods, the least significant bit (LSB) substitution approach might be the simplest method; in this method, secret data are embedded by slightly changing the pixel values of selected pixels in cover images (i.e., replacing the last significant k bits of a pixel value). However, if the size of the secret data is large, the original pixel values are considerably changed and thus the resulting image distortion is obvious. Chang et al. [11] proposed an LSB-based data hiding method using a bijective mapping function; in this method, a dynamic programming strategy is used to minimize the distortion. Balasubramanian et al. [12] proposed a data hiding method that is also performed in the spatial domain. To obtain large embedding capacity without damaging quality, secret data are embedded using a pixel-value differencing scheme in an Octonary manner. In the method in [13], secret data are embedded in intermediate significant bit planes.

For the aforementioned data hiding methods, secret data are embedded in an uncompressed format. In other words, the stego images are all uncompressed. Currently, most digital images that are distributed through the Internet are compressed for transmission efficiency. If the stego images are uncompressed, they might engender suspicions from invaders. Nevertheless, the new challenge for the conventional data hiding methods is that the hidden secret data scarcely survive after the compression process. Therefore, the goal of this study was to hide data in block truncation coding (BTC) compressed images while maintaining high image quality such that the compressed image has a close visual impression to the original image.

The remainder of this paper is organized as follows. In Sect. 2, BTC compression methods and studies related to hiding data in BTC-based compressed images are reviewed. Section 3 presents the proposed data hiding method based on BTC-compressed. In Sect. 4, the experimental results are provided and compared with existing methods. Finally, Sect. 5 provides the conclusion.

2 Preliminary

2.1 Concept of BTC compression

The concept of BTC compression [14] is introduced in this section. First, the original image is divided into nonoverlapping blocks of size $n \times n$, after which the pixel values in each block are quantized into two levels to save memory. The BTC method can be considered a spatial-domain compression method. The procedure of a simplified BTC, called absolute moment BTC (AMBTC) [15], is subsequently described. Compared with BTC, AMBTC considerably reduces the computational complexity. Therefore, we used AMBTC to compress images.

Unless intendedly stated, the default size of the block is $n = 4$. In AMBTC, the pixel values in each block are quantized into two levels by thresholding the block mean. Figure 1 presents an example of applying the AMBTC method.

For each block, the block mean ($\bar{x}$) and the two quantization levels (high mean b_i and low mean a_i) are calculated by

$$\bar{x} = \frac{1}{16}\sum_{i=1}^{16} x_i, \tag{1}$$

$$b_i = \frac{1}{16-q}\sum_{x_i \geq \bar{x}} x_i, \tag{2}$$

85	86	105	86
84	95	75	82
85	90	80	86
86	86	84	81

(a)

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

(b)

Fig. 1 Example of the AMBTC method. **a** The original image block. **b** The resulting bit map B with quantization levels ($a_i = 82, b_i = 90$)

and

$$a_i = \frac{1}{q} \sum_{x_i < \bar{x}} x_i, \tag{3}$$

where x_i denotes the ith pixel value within this block, $\bar{x}$ denotes the mean gray level of the block and q denotes the number of pixels whose values are less than the block mean. The threshold value is set to be the mean value to preserve the statistical moments of each block so that the quality of the compressed images remains acceptable. AMBTC preserves the absolute central moment of each original block.

As shown in Fig. 1b, a bit map B_i is generated by comparing the pixel value of the corresponding position with the block mean. If the pixel value is less than the block mean, it is set to be 0; otherwise, it is set to be 1. The storage to save the original block is 128 bits: 16 (pixels) × 8 (bits). By contrast, when AMBTC is performed, the storage to save the compressed block is 32 bits: the bit map (16 bits) and two quantization levels (16 bits). The compression code of each block consists of a trio (a_i, b_i, B_i). Therefore, the compression ratio of AMBTC is four.

2.2 Related BTC-based data hiding methods

In the network information-explosion era, combining data hiding algorithms with compression techniques has become a popular topic. Since the past decade, BTC-based data hiding methods have drawn considerable interest because of their low computational complexity [16–23]. Hiding data in compressed images not only reduces the transmission time over the Internet but also improves the security of the images. In addition, some BTC-based data hiding methods have been designed for image authentication [24–26].

Hong et al. [16] proposed a BTC-based data hiding method that entails embedding secret data by possibly exchanging the quantization levels (a_i and b_i), as well as reversing the bitmap (B_i) simultaneously. Therefore, this data hiding method is lossless. The secret data can be extracted by identifying whether the quantization levels are exchanged from the compression code. However, this method merely provides 1-bit embedding capacity in each block. Chen et al. [17] improved [16] by taking advantage of blocks with the same high and low mean values. In [17], if the high and low means of a block are same, the bitmap of that block is replaced by 16-bit secret data instead of 1-bit. Therefore, the embedding capacity is increased because of the use of those blocks.

Recently, Ou and Sun [19] extended [17] by predefining a threshold to classify smooth and complex blocks. In [17], the block is embedded with 16-bit secret data only if a_i equals to b_i. In contrast, in [19], as long as $|b_i - a_i|$ of a block is less than the threshold, that block is classified as a smooth block and can be embedded with 16-bit secret data. Therefore, the embedding capacity of [19] is larger than that of [17]. Furthermore, after the 16-bit data are embedded, the two quantization levels are recalculated to minimize image distortion. Compared with [16, 17], Ou and Sun's method substantially increases the embedding capacity with an acceptable compromise of image quality.

For BTC-based data hiding methods, achieving large embedding capacity and good image quality simultaneously is challenging. To maximize the preservation of image quality, in [21], the interpolation technique is combined with BTC to increase the embedding capacity; whereas in [22, 23], the relationship (e.g., the difference) between the two quantization levels is modified using embedding rules to increase the embedding capacity. In [24–26], an extra predefined reference table is used to adjust the compression code to avoid degrading the image quality. Some of the BTC-based data hiding methods exhibit the property of reversibility [27, 28], in which the original AMBTC-compressed image can be completely recovered from the compressed stego image.

Although hiding data in compressed stego images is desirable, of all the aforementioned methods, only those in [16, 27, 28] are lossless data hiding methods. Specifically, only the stego images generated from the methods in [16, 27, 28] have the same quality as that of the original BTC-compressed images. For the others, the quality of the stego images is lower (than that of the original BTC-compressed images) and the local average grayscale level is changed. However, compared with the quality of the original grayscale images, that of the compressed images is already degraded because of the AMBTC compression process.

2.3 Study motivation

Considering the aforementioned shortcomings of the existing methods, we present a novel BTC-based data hiding method. The advantages of this study are as follows.

- We propose a data hiding method that achieves high-quality and large embedding capacity simultaneously. The experimental results, presented in Sect. 4, demonstrate that the embedding capacity of the proposed method is the largest among all comparative methods (from [16, 17, 19, 28]); moreover, the image quality is the highest.
- We propose a block classification scheme that classifies blocks into three types. Therefore, the embed-

ding capacity of each type of block is more flexible. Moreover, we use a progressive data hiding strategy to maintain the local average grayscale level.

- When decoding, secret data can be extracted from the stego image itself without other information; that is, a blind decoding is achieved.
- In contrast to the aforementioned methods, the proposed method integrates data hiding with a quality improvement scheme to improve the quality of the final stego images. We overcome the challenge of maintaining the secret data while improving the image quality.

3 Proposed high quality data hiding method for AMBTC compressed images

Figure 2 illustrates a flowchart of the proposed BTC-based data hiding method, which involves three steps: (1) block classification, (2) progressive data hiding, and (3) image quality improvement. The following subsections detail these steps.

3.1 Block classification

A grayscale cover image is divided into nonoverlapping blocks of size 4×4 during the AMBTC compression process. To maximize the embedding capacity in a more flexible manner, blocks are classified into three types: smooth blocks (to be embedded with 16-bit data), complex_1 blocks (to be embedded with 1-bit data), and complex_2 blocks (to be embedded with 2-bit data).

First, each AMBTC-compressed block is identified as smooth or complex according to the absolute difference between its high and low means: If the difference is less than the threshold thr_1, it is classified as a smooth block; otherwise, it is a complex block:

$$\text{block type} = \begin{cases} \text{smooth}, & \text{if } |b_i - a_i| \le \text{thr}_1 \\ \text{complex}, & \text{otherwise} \end{cases}. \tag{4}$$

Second, each complex block is further classified as a complex_1 block or complex_2 block according to the number of its eight neighboring smooth blocks; that is, a complex block is further defined as follows:

$$\begin{cases} \text{complex\_1 block}, & \text{if the number of neighboring smooth blocks} \ge \text{thr}_2 \\ \text{complex\_2 block}, & \text{otherwise} \end{cases}. \tag{5}$$

Figure 4 illustrates an example of block classification. The proposed block classification scheme was designed based on two simple criteria; therefore, in a decoding process, the same block classification scheme can be used to classify the blocks of stego images, thus achieving blind decoding.

3.2 Proposed progressive data hiding strategy

To maintain the local average grayscale level and compatibility with the following quality improvement step, we propose a progressive data hiding strategy in which not all blocks are embedded with secret data simultaneously. Specifically, the data hiding for the complex_2 blocks is

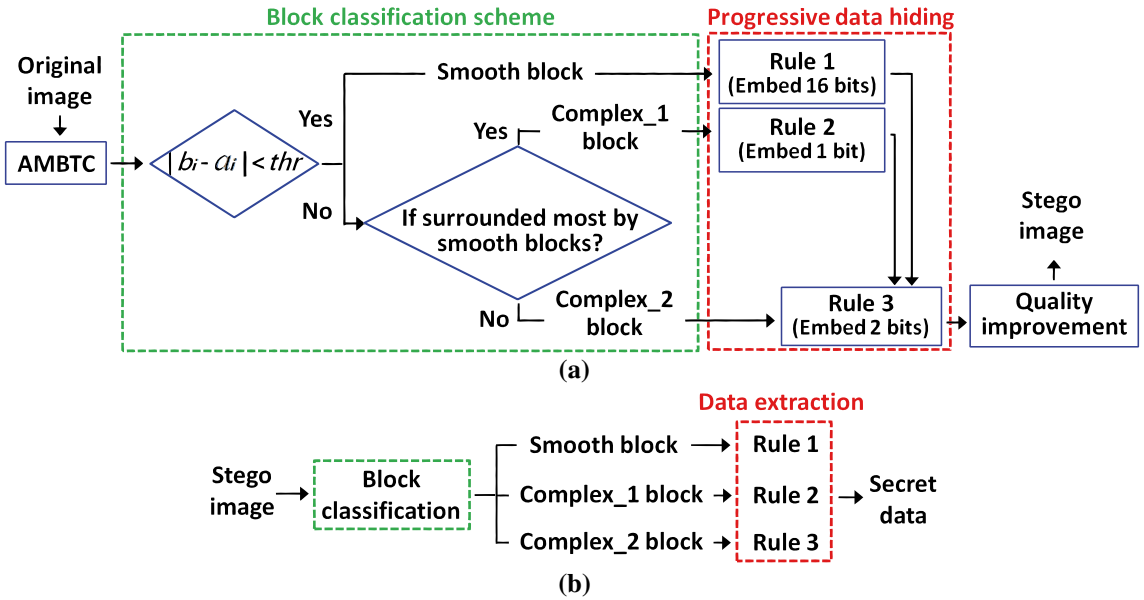

Fig. 2 Flowchart of the proposed BTC-based data hiding method: **a** data hiding phase; and **b** data extracting phase

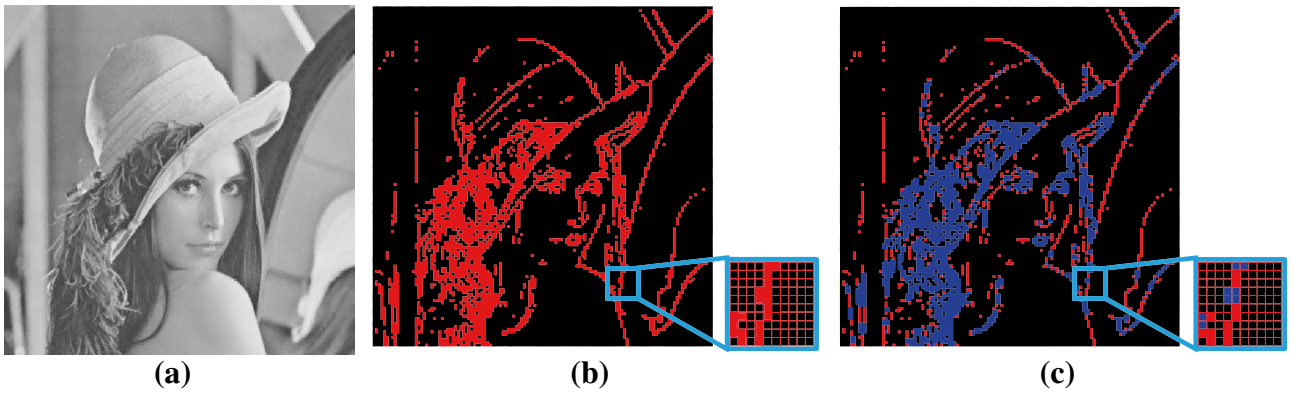

Fig. 3 Illustration of the proposed block classification scheme: **a** AMBTC-compressed image Lena; **b** after the first step of block classification ($\text{thr}_1 = 20$), where *black* and *red* units indicate the smooth and complex blocks, respectively; and **c** after the second step of block classification ($\text{thr}_2 = 5$), where the *blue* units are further classified as the complex_2 blocks (color figure online)

performed later to compensate for the gray level deviations that are engendered by the data hiding of the smooth blocks.

Given an image of size $W \times H$, we use $B_{m,n}$ to denote the bitmap of each AMBTC-compressed block, where $m = 1, \ldots, W/4$ and $n = 1, \ldots, H/4$ indicate the block location. According to the type of a block, one of the following three data hiding rules is implemented.

- *Rule 1 (for smooth blocks)* For the smooth blocks, because a_i is similar to b_i, the bitmap is relatively not significant. Therefore, the original bitmap is totally replaced by 16 bits secret data. For example, if the 16 bits secret data are $\{s_i, i = 1, \ldots, 16\}$, the first row of bitmap is replaced by $\{s_1, s_2, s_3, s_4\}$. As mentioned, through the use of AMBTC, a_i and b_i preserve the absolute central moment of the uncompressed block. However, the bitmap is changed due to the random secret data. Therefore, the gray level deviation (d) is calculated and must be compensated later:

$$d = a_i \times (q - q') + b_i \times (q' - q), \tag{6}$$

  where q denotes the number of "0" of the original bitmap, and q' denotes that of the new bitmap after the secret data are embedded.
- *Rule 2 (for complex_1 blocks)* In a complex_1 block, the difference between its high and low means is large, and the block is mostly surrounded by smooth blocks; hence, hiding data in complex_1 blocks is sensitive. Accordingly, only 1-bit secret data are embedded in complex_1 blocks in a lossless manner. As suggested in [16], the embedding of 1-bit information depends on whether the bitmap is reversed. If code 0 is embedded, the 32-bit compression code of this block is maintained as (a_i, b_i, B_i). Conversely, the compression code becomes $(b_i, a_i, \bar{B}_i)$, where $\bar{B}_i$ is obtained by executing the logical operation NOT on B_i. During the decoding process, the 1-bit secret data are extracted by judging whether the first mean (i.e., the first 8-bit grayscale) is greater than the second mean (i.e., the following 8-bit grayscale) of the compression code; this thus yields a blind decoding process.
- *Rule 3 (for complex_2 blocks)* We embed 2-bit secret data into each complex_2 block by not only possibly reversing the bitmap but also controlling the number of q' to be even or odd. Table 1 defines Rule 3 in detail. Once the secret data are embedded in smooth and complex_1 blocks, data hiding in complex_2 blocks is accomplished by performing Rule 3 in a raster order (see Fig. 4). According to Table 1, if the secret data are Code 00 and if q (of the original compressed block) is even, we can let q' to be odd by either adding or subtracting 1. However, it is not selected randomly. In addition, if the value of q must be changed for the even/odd property, the choice of plus one (or minus one) depends on the gray level deviations, as defined in (6), of previous blocks. A 2D error diffusion kernel $f[k, l]$, which is defined as the mirrored-type of the Floyd–Steinberg's kernel [29], can be expressed by

$$f[k, l] = \frac{1}{16} \times \begin{bmatrix} 1 & 5 & 3 \\ 7 & P & \end{bmatrix}. \tag{7}$$

Table 1 Rule 3 for embedding 2 bits secret data

Secret data	Embedded mode
Code 00	Bit map is maintained, and q' is odd
Code 01	Bit map is maintained, and q' is even
Code 10	Bit map is reversed, and q' is odd
Code 11	Bit map is reversed, and q' is even

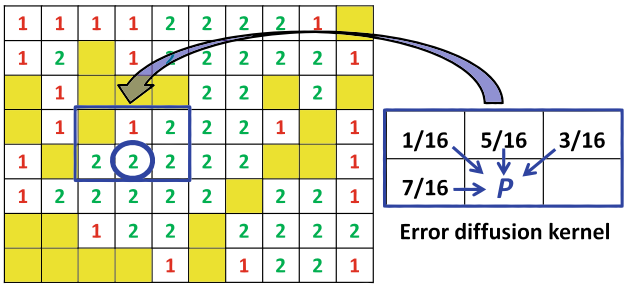

Fig. 4 Illustration of calculating the accumulated error (e), where the *blue circle* denotes current processing block, *1* complex_1 block, *2* complex_2 block, and the *yellow square* denotes a smooth block (color figure online)

where P indicates the current complex_2 block position, in which the weights are $f[-1,-1] = 1/16$, $f[-1,0] = 5/16$, $f[-1,1] = 3/16$, and $f[0,-1] = 7/16$. Let $d[m, n]$ denote the gray level deviation from data hiding in the block position $[m, n]$. The accumulated error (e) is defined as the sum of weighted deviations from previous blocks:

$$e[m,n] = \sum_{k,l} f[k,l] \times d[m+k, n+l]. \tag{8}$$

However, when applying Rule 3, how should we change the even/odd property of q'? We can address this concern by considering the accumulated error (e). Because the gray level deviations are diffused to the current block, if e is greater than zero, it implies that the local average grayscale level is increased due to the embedded secret data. Therefore, in the current block, the number of pixels with low mean is suggested to be more (i.e., making $q' = q + 1$). By contrast, if e is less than zero, we construct $q' = q - 1$. Through the consideration of e, data hiding in complex_2 blocks can compensate for gray level deviations.

3.3 Image quality improvement

To improve image quality, we adopted the direct-binary-search BTC (DBSBTC) method [30]. The original DBS method [31] is a halftoning approach that converts a continuous-tone image into a halftone image by optimally arranging discrete dots. The nature of the human visual system (HVS) is considered in the DBS method; therefore, this method generates halftone images with high quality.

Figure 5 displays a conceptual flowchart of the DBSBTC method; in this method, the traditional DBS strategy is conducted on AMBTC-compressed images through block-based processing (i.e., arranging the bitmap of each compressed block individually). With such a parallel property, DBSBTC achieves a high processing speed. It has been proved in [30] from extensive test images that because DBSBTC applies

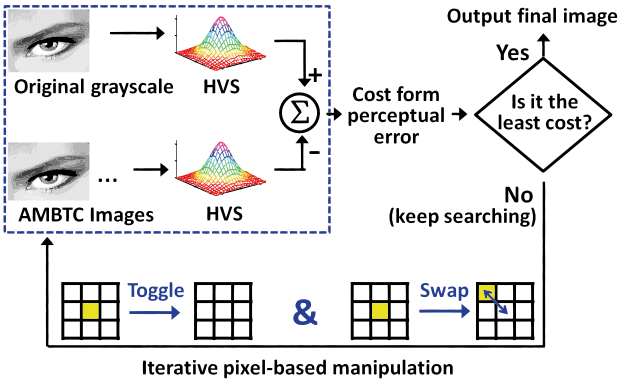

Fig. 5 Conceptual flowchart of the DBSBTC method, in which high mean (b_i) and low mean (a_i) of each block are continuously toggled and swapped, until the minimal cost is achieved

the DBS framework, the annoying contour artifacts of typical BTC-compressed images can be effectively alleviated.

Inherently, the DBSBTC method is an image compression approach, and not a data hiding approach; therefore, we must modify the DBSBTC method to improve the stego image quality without damaging the embedded secret data. We briefly introduce the concept of the DBSBTC method and describe the proposed modification in Step 4 of the stego construction procedure in Sect. 3.4.

Given an uncompressed continuous-tone image block $g[m, n]$ and the bitmap $B[m, n]$ of the corresponding AMBTC compressed block, the HVS-based perceived error $\tilde{E}$ can be expressed by

$$\tilde{E}(x,y) = \sum_{m,n} E[m,n] P_{HVS}(x - mX, y - nY), \tag{9}$$

where P_{HVS} denotes the HVS filter defined in the spatial domain, (X, Y) denotes the lattice basis of addressable dot (in units of inches/dot), and $E[m, n]$ denotes the error image between the continuous-tone image and the bitmap:

$$E[m,n] = g[m,n] - B[m,n]. \tag{10}$$

In DBSBTC, the perceptual-error-based cost is defined as

$$\phi = \int_x \int_y \left|\tilde{E}(x,y)\right|^2 \mathrm{d}x\mathrm{d}y. \tag{11}$$

The goal of the DBSBTC method is to manipulate the initial bitmap at each pixel continuously, until the minimal cost is achieved. Specifically, the optimal bitmap is found by minimizing ϕ:

$$B^*[m,n] = \arg\min_{B[m,n]} \phi. \tag{12}$$

To generate more trial patterns, the DBSBTC method adopts two operations, namely *Toggle* and *Swap*, at each pixel of $B[m, n]$. For the toggle operation, the value of

the current processing position is changed to an opposite value (i.e., 1 to 0, or vice versa). For the compressed image block, performing toggle in a pixel of $B[m, n]$ implies the exchange between the high and low means at the same pixel position. For the swap operation, the value of the current processing position is exchanged with that of its eight neighbors. At each pixel, all trial changes are tested, and only the change corresponding to the maximal reduction of cost is accepted. The trial changes of each pixel of $B[m, n]$ is tested iteratively, until the cost is converged to the desired local minimum.

Calculating the integral value in (11) is time consuming. Therefore, the DBSBTC method simplifies this by introducing two functions, (1) the autocorrelation function of HVS filter:

$$C_{\tilde{p}\tilde{p}}[m,n] = C_{\tilde{p}\tilde{p}}(mX, nY), \tag{13}$$

where

$$C_{\tilde{p}\tilde{p}}(mX, nY) = \int_s \int_t P_{HVS}(s,t) P_{HVS}(s+x, t+y) \mathrm{d}x\mathrm{d}y, \tag{14}$$

and (2) the cross-correlation function of HVS filter:

$$C_{\tilde{p}\tilde{e}}[m,n] = E[m,n] \ast\ast\ C_{\tilde{p}\tilde{p}}[m,n], \tag{15}$$

where $\ast\ast$ denotes the 2-D convolution. By utilizing the functions $C_{\tilde{p}\tilde{p}}[m,n]$ and $C_{\tilde{p}\tilde{e}}[m,n]$, the cost corresponding to a trial change can be estimated without calculating (11) directly. (Please refer to Ref. [30] for more details).

3.4 Summary of the proposed data hiding and blind decoding algorithm

- *Stego image construction end* The proposed BTC-based data hiding method is presented as follows. *STEP 1* The input grayscale image is compressed by the AMBTC method and is divided into 4×4 nonoverlapping blocks. *STEP 2* Each block is classified as one of three types: smooth block, complex_1 block, and complex_2 block. *STEP 3* Secret data are embedded in smooth blocks and complex_1 blocks according to Rules 1 and 2, respectively. Subsequently, the secret data are embedded in complex_2 blocks according to Rule 3. *STEP 4* The DBSBTC method is applied in each block except for the smooth blocks, because the bitmap of a smooth block is completely determined by the embedded 16-bit secret data. Nevertheless, the quality of the stego images is considerably improved by manipulating the bitmaps of the complex_1 and complex_2 blocks, which surround the smooth blocks. To avoid damaging the embedded secret data, the DBSBTC method is performed with a slight modification. In other words, for the execution of the DBSBTC method in complex_2 blocks, the toggle operation is not performed because it might change the odd/even property defined in Table 1.
- *Receiver end* The corresponding blind decoding method is presented as follows. *STEP 1* The stego image is redivided into 4 × 4 nonoverlapping blocks. *STEP 2* The blocks are arranged in a raster order. For each block, the difference $|b_i - a_i|$ is calculated and is classified into one of the three block types (smooth, complex_1, and complex_2) by the proposed block classification scheme. *STEP 3* According to the block types, the secret data are separately extracted using Rule 1 (for the smooth block), Rule 2 (for the complex_1 block), and Rule 3 (for the complex_2 block), respectively. *STEP 4* STEP 3 is repeated block by block in a raster order throughout the stego image. The secret data are completely extracted.

4 Experimental results

To evaluate the performance of the proposed method with various aspects, we conducted several experiments. For comparison, four existing BTC-based data hiding methods from [16, 17, 19, 28] were also executed. Figure 6 shows the thumbnail of all test images. A total of 20 grayscale test images were used: five standard images (size of 512 × 512), ten miscellaneous images (size of 504 × 756) from the McGill Image Database [32], and five medical images (size of 400 × 400) from the Osirix Database [33]. All the test images were selected randomly. In the experiments, the hidden messages were bitstreams generated by a pseudo-random number generator.

In the experiments, we apply two objective image quality measurements: the HVS-based Peak

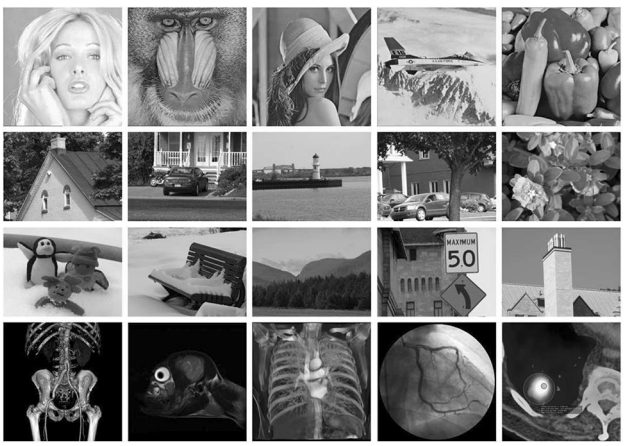

Fig. 6 The twenty test images used in this study, arranged in a raster order

Signal-to-Noise-Ratio (HPSNR) as suggested in Ref. [30] and the mean Structural Similarly Index Measure (MSSIM) as suggested in Ref. [19]. The first quality measurement, HPSNR, is defined as

$$10 \times \log_{10}\left(\frac{W \times H \times 255^2}{\sum_{W,H}\left[\sum_{m,n} q_{m,n}(g_{i+m,j+n} - h_{i+m,j+n})\right]^2}\right), \tag{16}$$

where the variable $g_{i,j}$ and $h_{i,j}$ denote the pixel values at position (i, j) of original grayscale cover image and the corresponding compressed stego image, respectively, and (H, W) denotes the image size. The variable $q_{m,n}$ denotes the coefficient of a 2-D Gaussian filter that model HVS. The HPSNR measurement can be viewed as integrating the lowpass characteristics of HVS into the conventional PSNR. A larger HPSNR value indicates higher degree of similar visual impression between $g_{i,j}$ and $h_{i,j}$, i.e., a higher quality.

The second quality measurement, MSSIM, is defined in [34] as

$$\text{MSSIM} = \frac{1}{M}\sum_{j=1}^{M} \text{SSIM}(g_j, h_j), \tag{17}$$

where M is the number of local windows (M set as 3 in this work, with window sizes 8, 12, and 16). The SSIM index formula is also defined in [34], in which the variables g_j and h_j denote the image contents at the jth local window, from the cover image and its corresponding stego image, respectively. For MSSIM, it ranges from zero to one, and a larger MSSIM value stands for more visually pleasing image. For payload size (or embedding capacity) comparison, the Payload indicator is defined as

$$\text{Payload} = \text{bit number of the secret data.} \tag{18}$$

4.1 Comparison of the proposed method with other methods

To demonstrate the superiority of the proposed method, we compare it with the methods of Hong [16], Chen et al. [17], Ou and Sun [19], and Chang [28]. The goal of this study was to demonstrate that the proposed method achieves a higher payload and superior image quality simultaneously, compared with the existing methods.

For the comparisons, the block size was set as 4×4 for each method. Because in all the comparative methods and our method, the AMBTC compression and the data hiding procedure is performed on individual blocks, achieving a fair comparison entails comparing the image quality and the payload using the same block size setting. Moreover, for Ou and Sun's method and our method, the parameter thr_1, which determines whether the block is complex or smooth, was set as 5 uniformly. The effects of different block sizes or different thr_1 values are discussed in Sect. 4.2. However, we chose the same parameter settings as much as possible to ensure fair comparisons of each data hiding scheme.

Figure 7 illustrates the results of the performance comparison in three aspects: Payload, HPSNR, and MSSIM. Table 2 summarizes the corresponding average (Avg.), standard deviation (Std.), required memory, and the processing time. The results were yielded with Windows 7 operating system, Intel core i5 with 2.7 GHz CPU and 4GB RAM. Figure 7a presents the Payload comparison results, and Fig. 7b, c present the results of comparing the corresponding image quality. The proposed method outperformed the comparative methods in terms of both payload size and image quality (Fig. 7). The experimental results obtained using the 20 test images fully validate the effectiveness of the proposed method.

We discuss in detail the experimental results presented in Fig. 7 as follows. With respect to Payload, because the block size setting was the same, it was determined by the payload of individual blocks. For the method of Chang [28], the payload of each block depends on its histogram. However, to maintain the reversibility of data hiding, the payload of [28] is typically low. For the method of Hong [16], each block is embedded with 1-bit information. In addition, for the method of Chen et al. [17], each block is embedded with 1- or 16-bit information, depending on whether it is a complex or smooth block. The method of Ou and Sun [19] basically has the same block classification concept as that of Chen et al.; however, the threshold in the method of Chen et al. is considerably rigid: Only when a_i equals to b_i, the block is classified as a smooth block and is embedded with 16-bit secret data. Whereas the method of Ou and Sun merely requests that if $|b_i - a_i|$ is less than a predefined parameter thr_1, the block is classified as a smooth block. Therefore, the comparison of the payload results among the three methods yielded the following ranking: the method of Ou and Sun (the highest of the three), the method of Chen et al., and the method of Hong. The proposed method improves the method of Ou and Sun by further defining the complex_2 blocks, which are embedded with 2-bit information. Therefore, the proposed method achieves the highest payload.

With respect to image quality, the proposed method achieved the highest quality indicator values among all methods. As shown in Fig. 7b, c, the line charts of [16, 17, 28] are totally overlapped because they have the same image quality explained as follows. For the methods of both Hong [16] and Chang [28], the quality of the produced stego images was equal to that of the original AMBTC-compressed images due to the reversibility property. For

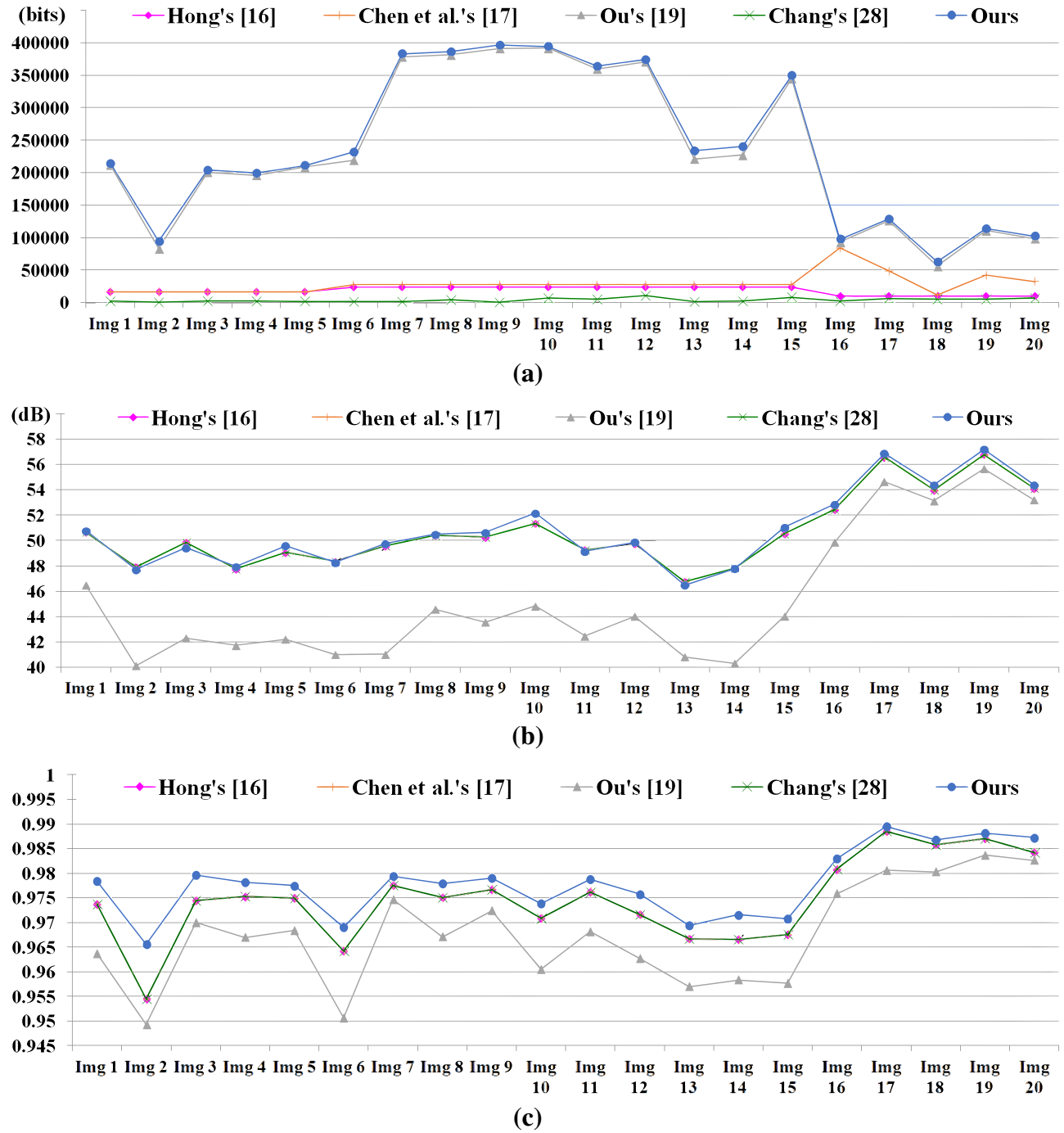

Fig. 7 Comparisons among methods using twenty images in various aspects: **a** payload; **b** HPSNR; and **c** MSSIM. Note that in **b**, **c**, the line charts of [16, 17, 28] are totally overlapped because they have the same image quality

the method of Ou and Sun [19], the quality of the produced stego images was poorer than that of the original AMBTC-compressed images due to the input of random secret data which are not related to the image content itself. For the proposed method, although the payload was the highest, we propose a progressive data hiding scheme to ensure that the images have less distortion and maintain local average grayscale during data hiding. According to [30], the DBS-BTC technique achieves superior image quality to that of the AMBTC technique; this work further shows that integrating the DBSBTC method with the proposed data hiding scheme achieved superior quality to that of the AMBTC method, which does not embed secret data. Figure 8 presents an example for visual comparison, in which the

Table 2 Comparisons among various BTC-based data hiding methods (using the test images shown in Fig. 6)

Method	Payload[a]	HPSNR (dB)		MSSIM		Required	Processing	Blind
	Avg.	Avg.	Std.	Avg.	Std.	memory[b]	time (ms)[c]	decoding
Hong's [16]	18503 bits	50.67	2.82	0.975	0.008	32×10^4 bits	4.28	Yes
Chen et al.'s [17]	20442.5 bits	50.67	2.82	0.975	0.008	32×10^4 bits	5.04	Yes
Ou and Sun's [19]	225080 bits	45.30	5.11	0.968	0.010	32×10^4 bits	5.62	Yes
Chang's [28]	3950.8 bits	50.67	2.82	0.975	0.008	36×10^4 bits	7.76	No
Proposed	231063.8 bits	50.84	2.99	0.978	0.007	32×10^4 bits	7.05	Yes

[a] The Std. of payload is not compared because payload varies significantly from image sizes

[b] The required memory for storing a stego image of size 400×400

[c] The average processing time for producing a stego image of size 400×400

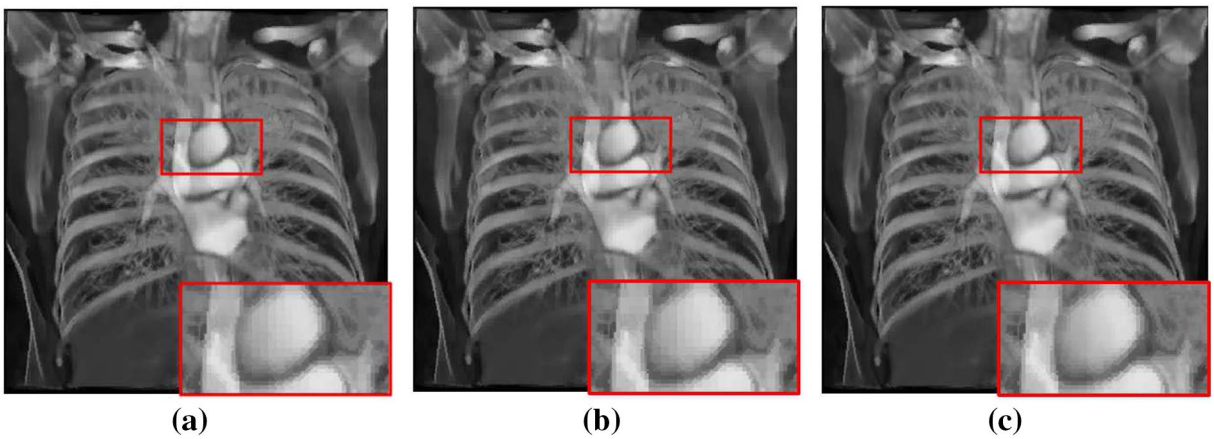

Fig. 8 Resulting stego images of the test image Carcinomix using various methods (block size $n = 4$ and $\text{thr}_1 = 5$). **a** Using methods of [16, 17, 28] (HPSNR = 53.97 dB, MSSIM = 0.985). **b** Using method of [19] (HPSNR = 53.14 dB, MSSIM = 0.980). **c** Using the proposed method (HPSNR = 54.38 dB, MSSIM = 0.987)

bottom right corner is an enlarged version used to demonstrate the image quality perceptually.

4.2 Discussion of the performance of the proposed algorithm regarding the influence of n and thr_1

For the proposed method, the block size (n) and the threshold value (thr_1) play an essential role for two reasons. First, BTC itself is a block-based compression scheme, and the number of blocks directly influences the payload. Second, the parameter thr_1, which classifies the smooth block and complex block, determines the payload allowed in each block. Observing the results of using different block sizes and the threshold values provides a more complete evaluation of proposed method's performance.

In the following, we only compare the performances among [17, 19], and the proposed method. The Hong's method [16] is not discussed because it does not use thr_1, and all block are embedded into 1-bit identically. The Chang's method [28] is not discussed because it is not based on block classification scheme.

With respect to the influence of block size (n), Fig. 9a–c illustrate the averages of Payload, HPSNR, and MSSIM derived from using 20 images, respectively. When the block size decreases, the number of blocks is higher and the payload increases; yet the image quality reduces. This trend is observed for all methods. For the proposed method, using a larger block size implies not only embedding less unnatural hidden data, but also allowing DBSBTC to adjust the image content freely within a larger area. As shown in Fig. 9, the proposed method outperforms comparative methods at each block size when the average results of Payload, HPSNR, and MSSIM are compared.

With respect to the influence of the threshold thr_1, because it is used to choose the smooth blocks, a larger thr_1 leads to more smooth blocks and higher payload. To

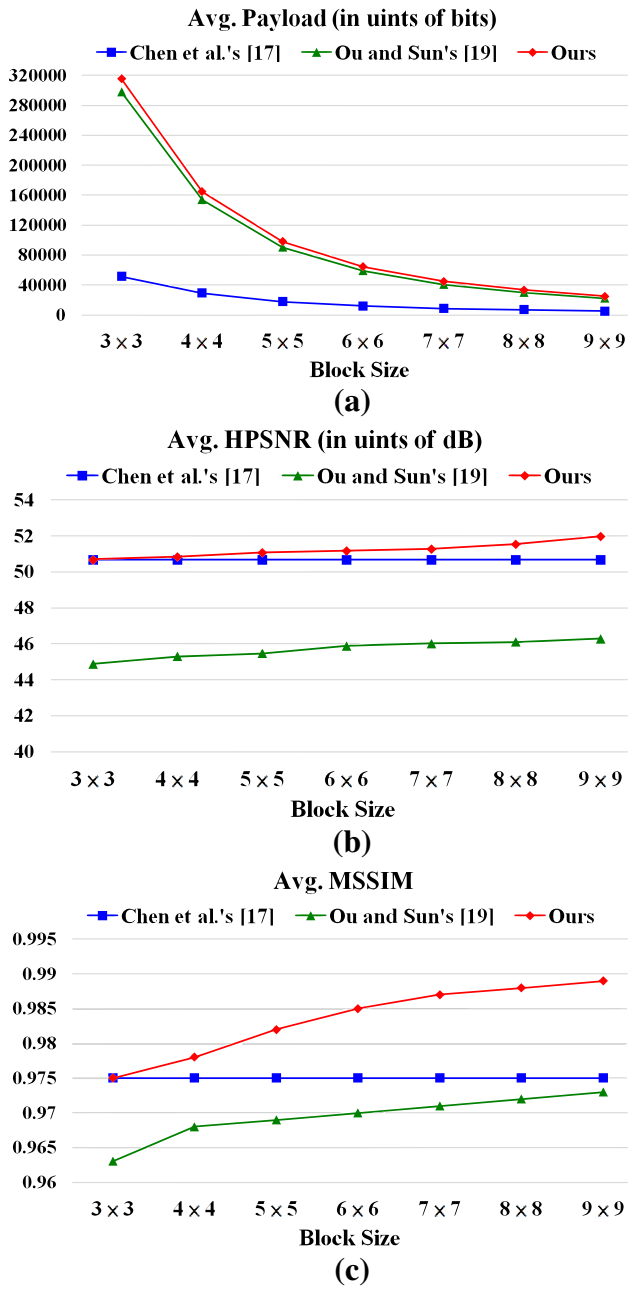

Fig. 9 When the block size (n) is changed, comparisons among methods using twenty images. **a** Comparison of average Payload. **b** Comparison of average HPSNR. **c** Comparison of average MSSIM

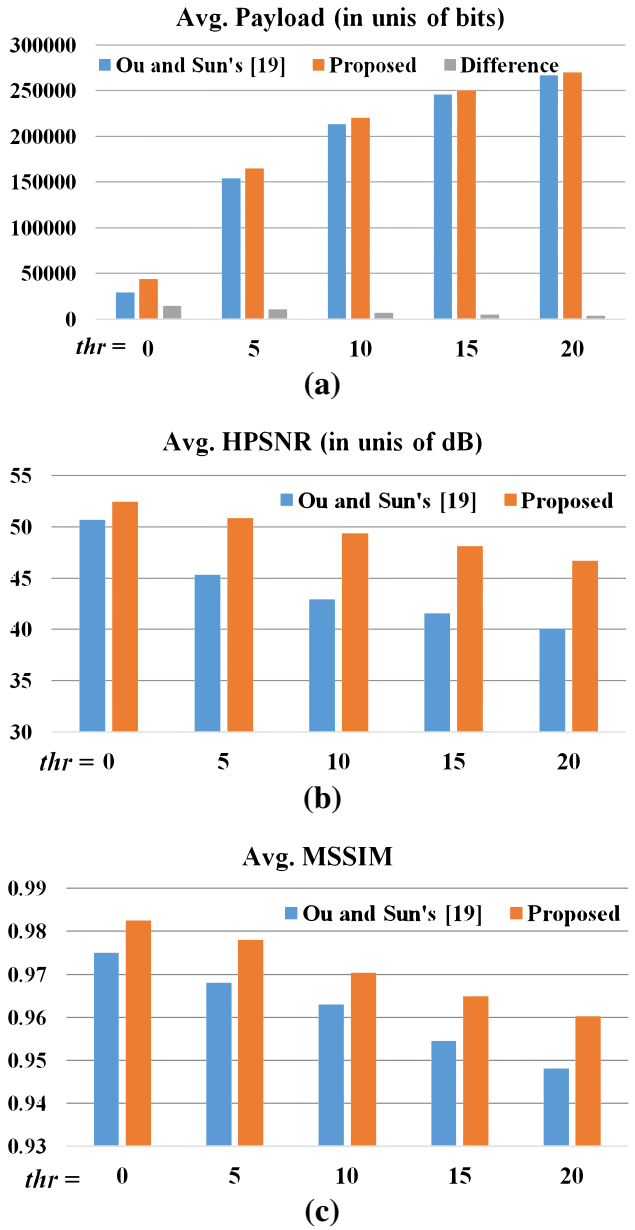

Fig. 10 When the threshold value (thr_1) is changed, comparisons between methods using twenty images. **a** Comparison of average Payload. **b** Comparison of average HPSNR. **c** Comparison of average MSSIM

compare the use of different threshold values, we only compared the proposed method with the method of Ou and Sun [19] because it is more representative than Chen et al.'s method [17]. Figure 10a–c present the averages of Payload, HPSNR, and MSSIM obtained from using twenty images, respectively. In Fig. 10, the threshold value thr_2 was set as 5. The proposed method again achieves both the highest payload and the best image quality at different threshold values.

For Chen et al.'s method [17] and Ou and Sun's method [19], data hiding is performed under a binary classification: smooth block (to be embedded with 16 bits) and complex block (to be embedded with 1 bit). However, the proposed method applies local neighborhood information for further classifying the complex blocks into complex_1 (to be embedded into 1 bit) and complex_2 (to be embedded into 2 bits) blocks. Therefore, the proposed method gains more payloads at different threshold values. For better understanding, Table 3

Table 3 Comparison between Ou and Sun's method [19] and the proposed method using the test image Lena*

Threhold	Ou and Sun's method [19]		Proposed method			Gain (bits)
	# of smooth	# of complex	# of smooth	# of complex_1	# of complex_2	in Payload
$thr_1 = 0^{\dagger}$	0	16384	0	4	**16380**	**16380**
$thr_1 = 5$	5640	10744	5640	786	**9958**	**9958**
$thr_1 = 10$	10407	5977	10407	677	**5300**	**5300**
$thr_1 = 15$	12290	4094	12290	909	**3185**	**3185**
$thr_1 = 20$	13333	3051	13333	947	**2104**	**2104**

∗ We highlight the values in the right two columns as boldface to show their direct relationship.

† When setting $thr_1 = 0$, the methods in [17] and [19] are the same

presents the results of different threshold values using the image Lena.

5 Conclusion and future work

The cloud technology plays a major role in future smart cities. Nowadays, many images are distributed worldwide daily, and most of such images are in a compressed format. Hiding data in compressed images improves the security level of transiting them over public networks. The hidden secret data can be used for purposes such as authentication, copyright protection, and source tracing. The BTC method is one of the most commonly used image compression techniques because of its low computational complexity. Therefore, this study presents a data hiding and blind decoding method for BTC-compressed images. The experimental results demonstrate superior performance for the proposed method in terms of embedding capacity and image quality, compared with existing methods.

Currently, the proposed method is designed for converting a grayscale image into a BTC-based stego image. In the future, we plan to extend this work to color images. In addition, when facing attacks, such as tampering and cropping, the robustness of the proposed method has yet to be discussed. A robust watermarking, in which the embedded data can be fully (or almost) extracted under different attacks, will be studied.

Acknowledgements This work was supported in part by the Ministry of Science and Technology (104-2221-E-027-032).

References

1. Deng, M., Petkovic, M., Nalin, M., Baroni, I.: A home healthcare system in the cloud-addressing security and privacy challenges. In: Proc. IEEE Int. Conf. Cloud, Computing, pp. 549–556 (2011)
2. Zissis, D., Lekkas, D.: Addressing cloud computing security issues. Future Gener. Comput. Syst. **28**(3), 583–592 (2012)
3. Jiang, R.: Advanced secure user authentication framework for cloud computing. Int. J. Smart Sens. Intell. Syst. **6**(4), 1700–1724 (2013)
4. Arka, I., Chellappan, K.: Collaborative compressed i-cloud medical image storage with decompress viewer. Procedia Comput. Sci. **42**, 114–121 (2014)
5. Podilchuk, C., Delp, E.: Digital watermarking: algorithms and applications. IEEE Signal Process. Mag. **18**, 33–46 (2001)
6. Chan, P.W., Lyu, M.R., Chin, R.T.: A novel scheme for hybrid digital video watermarking: approach, evaluation and experimentation. IEEE Trans. Circuits Syst. Video Technol. **15**, 1638–1649 (2005)
7. Panah, A.S., Schyndel, R.V., Sellis, T., Bertino, E.: On the properties of non-media digital watermarking: a review of state of the art techniques. In: IEEE, Access, pp. 2670–2704 (2016)
8. Chen, P., Lin, H.: A dwt based approach for image steganography. Int. J. Appl. Sci. Eng. **4**, 275–290 (2006)
9. Shejul, A.A., Kulkarni, U.L.: A dwt based approach for steganography using biometrics. In: Proc. IEEE Int. Conf. Data Storage and Data, Engineering, pp. 39–43 (2010)
10. Parah, S.A., Sheikh, J.A., Loan, N.A., Bhat, G.M.: Robust and blind watermarking technique in dct domain using inter-block coefficient differencing. Digit. Signal Process. **53**, 11–24 (2016)
11. Chang, C., Hsiao, J., Chan, C.: Finding optimal lsb substitution in image hiding by dynamic programming strategy. Pattern Recognit. **36**(7), 1583–1595 (2003)
12. Balasubramanian, C., Selvakumar, S., Geetha, S.: High payload image steganography with reduced distortion using octonary pixel pairing scheme. Multimed. Tools Appl. **73**(3), 2223–2245 (2014)
13. Parah, S.A., Sheikh, J.A., Assad, U.I., Bhat, G.M.: Hiding in encrypted images: a three tier security data hiding system. In: Multidimensional systems and Signal Processing, pp. 1–24 (2015)
14. Delp, E., Mitchell, O.: Image compression using block truncation coding. IEEE Trans. Commun. **27**(9), 1335–1342 (1979)
15. Lema, M., Mitchell, O.: Absolute moment block truncation coding and its application to color images. IEEE Trans. Commun. **32**(10), 1148–1157 (1984)
16. Hong, W., Chen, T., Shiu, C.: Lossless steganography for ambtc-compressed images. In: Proc. IEEE Congress on Image and Signal Processing, pp. 13–17 (2008)
17. Chen, J., Hong, W., Shiu, C.: Steganography for btc compressed images using no distortion technique. Imaging Sci J **58**(4), 177–185 (2010)
18. Yang, H., Yin, J.: A secure removable visible watermarking for btc compressed images. Multimed. Tools Appl. **74**(6), 1725–1739 (2015)
19. Ou, D., Sun, W.: High payload image steganography with minimum distortion based on absolute moment block truncation coding. Multimed. Tools Appl. **74**(21), 9117–9139 (2015)
20. Chang, C., Liu, Y., Nguyen, S.T.: A novel data hiding scheme for block truncation coding compressed images using dynamic

programming strategy. In: SPIE Int. Conf. Graphic and Image Processing (ICGIP), pp. 1–10 (2015)
21. Tang, M., Zeng, S., Chen, X., Hu, J., Du, Y.: An adaptive image steganography using ambtc compression and interpolation technique. Optik (Int. J. Light Electron Opt.) **127**, 471–477 (2016)
22. Huang, Y., Chang, C., Chen, Y.: Hybrid secret hiding schemes based on absolute moment block truncation coding. In: Multimedia tools and applications, pp. 1–16 (2016)
23. Hong, W., Chen, T., Yin, Z., Luo, B., Ma, Y.: Data hiding in ambtc images using quantization level modification and perturbation technique. In: Multimedia tools and applications, pp. 1–22 (2016)
24. Lin, C., Huang, Y., Tai, W.: Novel image authentication scheme for ambtc-compressed images. In: Int. Conf. Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), pp. 134–137 (2014)
25. Li, W., Lin, C., Pan, J.: Novel image authentication scheme with fine image quality for btc-based compressed images. In: Multimedia Tools and Applications, pp. 4771–4793 (2015)
26. Lin, C., Huang, Y., Tai, W.: A novel hybrid image authentication scheme based on absolute moment block truncation coding. In: Multimedia tools and applications, pp. 1–26 (2015)
27. Zhang, Y., Guo, S., Lu, Z., Luo, H.: Reversible data hiding for btc-compressed images based on lossless coding of mean tables. IEICE Trans. Commun. **E96-B**, 624–631 (2013)
28. Chang, I., Hu, Y., Chen, W., Lo, C.: High capacity reversible data hiding scheme based on residual histogram shifting for block truncation coding. Signal Process. **108**, 376–388 (2015)
29. Floyd, R., Steinberg, L.: An adaptive algorithm for spatial grayscale. In: Proc. Int. Conf. SID Dig. Soc. Inform. Display, pp. 75–77 (1976)
30. Guo, J., Su, C.: Improved block truncation coding using extreme mean value scaling and block-based high speed direct binary search. IEEE Signal Process. Lett. **18**, 694–697 (2011)
31. Analoui, M., Allebach, J.P.: Model-based halftoning using direct binary search. In: Proc. SPIE, Human Vision, Visual Processing, and Digital Display III, vol. 1666, pp. 96–108 (1992)
32. Olmos, A., Kingdom, F.A.: A biologically inspired algorithm for the recovery of shading and reflectance images. Perception **33**, 1463–1473 (2004)
33. Rosset, A., Spadola, L., Ratib, O.: Osirix: An open-source software for navigating in multidimensional dicom images. J. Digit. Imaging **17**(3), 205–216 (2004)
34. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**, 600–612 (2004)