A Reversible Data Hiding Scheme for BTC-Compressed Images

Ching-Chiuan Lin
Department of Information Technology
Overseas Chinese University
Taichung, Taiwan

Kuo Feng Hwang
Department of Information Technology
Overseas Chinese University
Taichung, Taiwan

Shih-Chieh Chen
Department of Multimedia and Game Design
Overseas Chinese University
Taichung, Taiwan

Chi-Ming Yao
Department of Information Technology
Overseas Chinese University
Taichung, Taiwan

***Abstract*—This paper proposes a reversible data hiding scheme for BTC-compressed images. A block in the BTC-compressed image consists of a larger block-mean pixel and a smaller block-mean pixel. Two message bits are embedded into a pair of neighboring blocks. One is embedded by expanding the difference between the two larger block-mean pixels and the other is embedded by expanding the one between the two smaller block-mean pixels. Experimental results show that the embedding strategy may decrease the modification of images. The proposed scheme may obtain a stego-image with high visual quality and a payload capacity of one bit per block, approximately.**

Keywords—Block Truncation Coding; Reversible Data Hiding; Difference Expansion

I. INTRODUCTION

Transmitting secret data over the Internet is a popular application. To prevent secret data from malicious attack, users usually encrypt important data before transmission. Encrypting data is a complicated computation which coverts data into a meaningless format, which may attract hackers' attention and result in undesired attack. Hiding data in an image is an alternative way for secret communication. It embeds important data by slightly modifying the image. Hackers may not percept that important data are embedded in the normal image. Therefore, an undesired attack may be avoided.

To speed up transmission or decrease required storage, images are usually converted into smaller ones with compressed format. JPEG[1] is a popular image format which applying Discrete Cosine Transform (DCT) to compress an image. It needs complicated computation for image compression and decompression. Another technique for image compression is Vector Quantization (VQ) [2] which records image blocks in a code book and uses an index of code word to encode a block of image. A popular block-oriented image compression method is Block Truncation Coding (BTC) [3]. Compared to JPEG and VQ, BTC is a simple and efficient encoding method for image compression.

Many data hiding schemes for BTC-compressed images were proposed [4–9]. However, most of them were irreversible after secret data were extracted. Namely, the original image may not be completely recovered. This may degrade the image and decrease user's motivation for hiding data in the image. Therefore, a reversible data hiding scheme, which can completely recover the original image, is required.

Reversible data hiding schemes embed data in redundant space of an image [10–13]. Most of the schemes belong to the two families: shifting histogram and difference expansion. The former shifts histogram of pixels or differences to get redundant space located in the peak point of histogram for embedding a message. The latter expands the difference between a pair of pixels, i.e. doubles the difference. As a result, the expanded difference would be an even number and the least significant bit of each expanded difference value is equal to 0 which is the available embedding space.

TABLE I. AN EXAMPLE OF DIFFERENCE EXPANSION

(y_1, y_2)	Δy	ΔY	Embed a bit of 0		Embed a bit of 1	
			(y_1', y_2')	$\Delta Y'$	(y_1', y_2')	$\Delta Y'$
(95, 95)	0	0	(95, 95)	0	(95, 96)	1
(95, 96)	1	2	(94, 96)	2	(94, 97)	3
(94, 96)	2	4	(93, 97)	4	(93, 98)	5
(94, 97)	3	6	(92, 98)	6	(92, 99)	7

Table I is an example illustrating the embedding process of difference expansion, where (y_1, y_2) are pixel values, Δy and ΔY are original and expanded differences, respectively. After embedding a bit of 0 or 1 into the pair of pixels, their stego-pixel values and difference would become (y_1', y_2') and $\Delta Y'$, respectively, as shown in the table. If $\Delta y = 1$, expanding a difference may be implemented by either increasing the larger pixel or decreasing the smaller by one. If $\Delta y > 1$, increasing the larger one and decreasing the smaller one at the same time is a better option, since it may result in less modification of pixel values and get a benefit of smaller perception of image distortion by human vision.

In the decoding process, a bit of $s = 0$ or $s = 1$ is extracted if $\Delta Y'$ is equal to an even or odd number, respectively, and the original difference may be obtained by calculating $\Delta y = \lfloor \Delta Y'/2 \rfloor$. Then $y_1 = y_1' + \lfloor (\Delta y + 1)/2 \rfloor$ and $y_2 = y_2' -$

$\lfloor \Delta y/2 \rfloor - s$ are completely recovered. For a 256-level grayscale image, if blocks with $y_1' < 0$ or $y_2' > 255$, these blocks would not be candidates for embedding a message. These exceptions are recorded in the overhead information for identifying if a block embeds a message.

This paper proposes a reversible data hiding scheme for BTC-compressed images. Embedding space is gotten from expanding a difference between mean pixel values. A block in the BTC-compressed image consists of a larger block-mean pixel and a smaller block-mean pixel. Two message bits are embedded into a pair of neighboring blocks. One is embedded by expanding the difference between the two larger block-mean pixels and the other is embedded by expanding the one between the two smaller block-mean pixels. Experimental results show that the embedding strategy may decrease the modification of images. The proposed scheme may obtain a stego-image with high visual quality and a payload capacity of one bit per block, approximately.

The rest of this paper is organized as follows. The BTC algorithm is briefly reviewed in Section II. Section III introduces the proposed scheme including embedding and extraction processes. To help readers understand the proposed scheme, an embedding example is also given in this section. Section IV demonstrates our experimental results in terms of image visual quality and payload capacity. Finally, conclusions are given in Section V.

II. Block Truncation Coding

In the following, the BTC encoding algorithm is briefly reviewed. Notations are defined as follows:

- X_i is a block of image, with $n = k \times k$ pixels in the block,
- $x_i(p,q)$ is the pixel value of block i where p and q are indexes of pixels and $1 \le p,q \le k$,
- $\bar{x}_i = \frac{\sum_{p=1}^{k}\sum_{q=1}^{k} x_i(p,q)}{k \times k}$ is the average pixel value of block i,
- m_i is the number of pixels in block i with $x_i(p,q) \ge \bar{x}_i$,
- $\bar{x}_i^{MAX} = \left\lfloor \frac{\sum x_i}{m_i} |_{x_i(p,q) \ge \bar{x}_i} \right\rfloor$ is the larger block-mean pixel value, for $x_i(p,q) \ge \bar{x}_i$, in block i,
- $\bar{x}_i^{min} = \left\lfloor \frac{\sum x_i}{n - m_i} |_{x_i(p,q) < \bar{x}_i} \right\rfloor$ is the smaller block-mean pixel value, for $x_i(p,q) < \bar{x}_i$, in block i, and
- X_i' is the decoded block of BTC-compressed image.

The BTC encoding algorithm is introduced as follows:

1) Select an image and divide it into non-overlapping blocks each of them contains $k \times k$ pixels.

2) For each block i, calculate $\bar{x}_i$, and then m_i, $\bar{x}_i^{MAX}$, $\bar{x}_i^{min}$.

3) Encode block i denoted by $\hat{B}_i = (\bar{x}_i^{MAX}, \bar{x}_i^{min}, B_i)$, where $B_i = \{b_i(p,q) | b_i(p,q) = 0 \ or \ 1\}$ is a binary block with $b_i(p,q) = 0$ and $b_i(p,q) = 1$ if $x_i(p,q) < \bar{x}_i$ and $x_i(p,q) \ge \bar{x}_i$, respectively.

4) Repeat step 3 until all blocks are encoded.

The following example gives an illustration for the process of BTC. Given, for a 256-level grayscale image, a block $X_i = \{x_i(p,q) | 0 \le x_i(p,q) \le 255\}$ is as follows:

136	132	133	134
135	134	137	138
132	132	131	132
133	134	135	136

We have $\bar{x}_i = 134$, $m_i = 9$, $\bar{x}_i^{MAX} = 135$, $\bar{x}_i^{min} = 132$ and $B_i = \{b_i(p,q) | b_i(p,q) = 0 \text{ or } 1\}$ as shown below.

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

Note that $\bar{x}_i$ may not be exactly equal to $(\bar{x}_i^{MAX} + \bar{x}_i^{min})/2$. For convenience, we use $\hat{B}_i = (\bar{x}_i^{MAX}, \bar{x}_i^{min}, B_i)$ to denote an encoded block. In the example, $\bar{x}_i^{MAX}$, $\bar{x}_i^{min}$ and B_i need a memory space of 8, 8 and 16 bits, respectively. Compared to its uncompressed block, the compression rate of BTC is $(8 + 8 + 16)/(8 * 16) = 0.25$. This may significantly decrease the required space for storing an image without complicated computation.

The encoded image is decoded by replacing $b_i(p,q)$ with $\bar{x}_i^{MAX}$ or $\bar{x}_i^{min}$ if $b_i(p,q) = 1$ or $b_i(p,q) = 0$, respectively. In the above example, X_i' is decoded as follows:

135	132	132	135
135	135	135	135
132	132	132	132
132	135	135	135

III. Proposed Scheme

The proposed scheme is designated to embed a message into a BTC-compressed image and extract the message from the stego-image. Therefore, the BTC encoding procedure in Section II must be applied to a grayscale cover image if it is not compressed by the BTC algorithm. We will introduce the proposed scheme, including the embedding and extraction procedures in the following sections. Note that the stego-image would be completely recovered in the proposed scheme.

A. Embedding procedure

The embedding procedure is used to embed a binary bit string into a BTC-compressed image *T*. Required embedding space is obtained from expanding the difference between block-mean values in blocks. Details are listed as follows:

1) Convert a message into a binary bit string $S = s_1 s_2 \dots s_i \dots$, where $s_i \in \{0,1\}$. For example, a message $(23)_{16}$ is converted into $(00100011)_2$ and $s_1 = 0, s_2 = 0, s_3 = 1$, etc.

2) Sequentially scan the BTC-compressed image T in an order which was negotiated with the decoder. For each pair of blocks $\hat{B}_{2i-1} = (\bar{x}_{2i-1}^{MAX}, \bar{x}_{2i-1}^{min}, B_{2i-1})$ and $\hat{B}_{2i} = (\bar{x}_{2i}^{MAX}, \bar{x}_{2i}^{min}, B_{2i})$, $i = 1,2,3,\dots$ in the image T, calculate

$$\Delta x_i^{MAX} = \left| \bar{x}_{2i-1}^{MAX} - \bar{x}_{2i}^{MAX} \right|,$$

$$\Delta x_i^{min} = \left|\bar{x}_{2i-1}^{min} - \bar{x}_{2i}^{min}\right|,$$

$$\begin{pmatrix}\bar{\bar{x}}_{2i-1}^{MAX}\\ \bar{\bar{x}}_{2i}^{MAX}\end{pmatrix} = \begin{cases}\begin{pmatrix}\bar{x}_{2i-1}^{MAX} + \lfloor\Delta x_i^{MAX}/2\rfloor + s_{2i-1}\\ \bar{x}_{2i}^{MAX} - \lfloor(\Delta x_i^{MAX}+1)/2\rfloor\end{pmatrix} & \text{if } \bar{x}_{2i-1}^{MAX} > \bar{x}_{2i}^{MAX},\\ \begin{pmatrix}\bar{x}_{2i-1}^{MAX} - \lfloor(\Delta x_i^{MAX}+1)/2\rfloor\\ \bar{x}_{2i}^{MAX} + \lfloor\Delta x_i^{MAX}/2\rfloor + s_{2i-1}\end{pmatrix} & \text{otherwise,}\end{cases}$$

$$\begin{pmatrix}\bar{\bar{x}}_{2i-1}^{min}\\ \bar{\bar{x}}_{2i}^{min}\end{pmatrix} = \begin{cases}\begin{pmatrix}\bar{x}_{2i-1}^{min} + \lfloor\Delta x_i^{min}/2\rfloor + s_{2i}\\ \bar{x}_{2i}^{min} - \lfloor(\Delta x_i^{min}+1)/2\rfloor\end{pmatrix} & \text{if } \bar{x}_{2i-1}^{min} > \bar{x}_{2i}^{min},\\ \begin{pmatrix}\bar{x}_{2i-1}^{min} - \lfloor(\Delta x_i^{min}+1)/2\rfloor\\ \bar{x}_{2i}^{min} + \lfloor\Delta x_i^{min}/2\rfloor + s_{2i}\end{pmatrix} & \text{otherwise.}\end{cases}$$

3) If $0 \le \bar{\bar{x}}_{2i-1}^{min}, \bar{\bar{x}}_{2i-1}^{MAX}, \bar{\bar{x}}_{2i}^{min}, \bar{\bar{x}}_{2i}^{MAX} \le 255$, *encode blocks* $2i-1$ *and* $2i$ *as* $\ddot{B}_{2i-1} = (\bar{\bar{x}}_{2i-1}^{MAX}, \bar{\bar{x}}_{2i-1}^{min}, B_{2i-1})$ *and* $\ddot{B}_{2i} = (\bar{\bar{x}}_{2i}^{MAX}, \bar{\bar{x}}_{2i}^{min}, B_{2i})$.

4) If $\bar{\bar{x}}_{2i-1}^{min} < 0$, $\bar{\bar{x}}_{2i}^{min} < 0$, $\bar{\bar{x}}_{2i-1}^{MAX} > 255$, or $\bar{\bar{x}}_{2i}^{MAX} > 255$, let the pair of blocks be unchanged, i.e. $\ddot{B}_{2i-1} = (\bar{x}_{2i-1}^{MAX}, \bar{x}_{2i-1}^{min}, B_{2i-1})$ and $\ddot{B}_{2i} = (\bar{x}_{2i}^{MAX}, \bar{x}_{2i}^{min}, B_{2i})$, and record the pair of blocks index $2i-1$ as an overhead information. In other words, this pair of blocks embeds nothing, and s_{2i-1} and s_{2i} are embedded into the next pair of blocks.

5) Obtain the stego-image $T' = \{\ddot{B}_i | i = 1,2,3,\ldots\}$.

Note that if $\Delta x_i^{MAX} = 0$, $\Delta x_i^{min} = 0$, $s_{2i-1} = 0$, and $s_{2i} = 0$, the pair of blocks are also unchanged and they embed two bits of 0. This means an unchanged pair of blocks is not equivalent to embedding nothing.

The proposed scheme embeds two bits in a pair of blocks, instead of embedding one bit in a block. Our embedding strategy is to decrease the modification of an image. According to our experiments, for most blocks, Δx_i^{MAX} or Δx_i^{min} is usually less than $\left|\bar{x}_{2i-1}^{MAX} - \bar{x}_{2i-1}^{min}\right|$ or $\left|\bar{x}_{2i}^{MAX} - \bar{x}_{2i}^{min}\right|$. Namely, expanding a smaller difference can make slighter image modification than expanding a larger one.

B. Extraction procedure

Whenever a decoder would like to extract the embedded message from a stego-image and recover it to its original BTC-compressed image *T*, the extraction procedure would be applied. Details of the procedure are listed as follows:

1) Block by block scan the stego-image T' as the order in the embedding procedure.

2) For each pair of blocks $\ddot{B}_{2i-1} = (\bar{\bar{x}}_{2i-1}^{MAX}, \bar{\bar{x}}_{2i-1}^{min}, B_{2i-1})$ and $\ddot{B}_{2i} = (\bar{\bar{x}}_{2i}^{MAX}, \bar{\bar{x}}_{2i}^{min}, B_{2i})$ $i = 1,2,3,\ldots$, if index $2i-1$ is not recorded in the overhead information, do step 3, otherwise skip blocks $\ddot{B}_{2i-1}$ and $\ddot{B}_{2i}$ since they embed nothing and are not needed to be recovered.

3) Calculate

$$\Delta\ddot{x}_i^{MAX} = \left|\bar{\bar{x}}_{2i-1}^{MAX} - \bar{\bar{x}}_{2i}^{MAX}\right|,$$

$$\Delta\ddot{x}_i^{min} = \left|\bar{\bar{x}}_{2i-1}^{min} - \bar{\bar{x}}_{2i}^{min}\right|,$$

$$\Delta x_i^{MAX} = \lfloor\Delta\ddot{x}_i^{MAX}/2\rfloor,$$

$$\Delta x_i^{min} = \lfloor\Delta\ddot{x}_i^{min}/2\rfloor$$

and extract

$$s_{2i-1} = \Delta\ddot{x}_i^{MAX} \bmod 2,$$

$$s_{2i} = \Delta\ddot{x}_i^{min} \bmod 2.$$

Then calculate

$$\begin{pmatrix}\bar{x}_{2i-1}^{MAX}\\ \bar{x}_{2i}^{MAX}\end{pmatrix} = \begin{cases}\begin{pmatrix}\bar{\bar{x}}_{2i-1}^{MAX} - \lfloor\Delta x_i^{MAX}/2\rfloor - s_{2i-1}\\ \bar{\bar{x}}_{2i}^{MAX} + \lfloor(\Delta x_i^{MAX}+1)/2\rfloor\end{pmatrix} & \text{if } \bar{\bar{x}}_{2i-1}^{MAX} > \bar{\bar{x}}_{2i}^{MAX},\\ \begin{pmatrix}\bar{\bar{x}}_{2i-1}^{MAX} + \lfloor(\Delta x_i^{MAX}+1)/2\rfloor\\ \bar{\bar{x}}_{2i}^{MAX} - \lfloor\Delta x_i^{MAX}/2\rfloor - s_{2i-1}\end{pmatrix} & \text{otherwise,}\end{cases}$$

$$\begin{pmatrix}\bar{x}_{2i-1}^{min}\\ \bar{x}_{2i}^{min}\end{pmatrix} = \begin{cases}\begin{pmatrix}\bar{\bar{x}}_{2i-1}^{min} - \lfloor\Delta x_i^{min}/2\rfloor - s_{2i}\\ \bar{\bar{x}}_{2i}^{min} + \lfloor(\Delta x_i^{min}+1)/2\rfloor\end{pmatrix} & \text{if } \bar{\bar{x}}_{2i-1}^{min} > \bar{\bar{x}}_{2i}^{min},\\ \begin{pmatrix}\bar{\bar{x}}_{2i-1}^{min} + \lfloor(\Delta x_i^{min}+1)/2\rfloor\\ \bar{\bar{x}}_{2i}^{min} - \lfloor\Delta x_i^{min}/2\rfloor - s_{2i}\end{pmatrix} & \text{otherwise,}\end{cases}$$

and recover the pair of blocks to

$$\hat{B}_{2i-1} = (\bar{x}_{2i-1}^{MAX}, \bar{x}_{2i-1}^{min}, B_{2i-1}) \text{ and}$$

$$\hat{B}_{2i} = (\bar{x}_{2i}^{MAX}, \bar{x}_{2i}^{min}, B_{2i}).$$

4) Obtain the original BTC-compressed image T.

C. An example illustrating the proposed scheme

This section gives an example to illustrate the proposed scheme. Figure 1(a) is a BTC-compressed cover image with 8 blocks. Since B_{2i-1} and B_{2i} are binary arrays and they remain unchanged during the embedding procedure, their contents would not be shown in the example for simplicity. Let the message to be embedded be a character "A" and it's ASCII code is $S = (41)_{16} = (00100001)_2$.

The encoder calculates $\Delta x_1^{MAX} = |\bar{x}_1^{MAX} - \bar{x}_2^{MAX}| = |135 - 134| = 1$, $\Delta x_1^{min} = \left|\bar{x}_1^{min} - \bar{x}_2^{min}\right| = |130 - 132| = 2$, and

$$\begin{aligned}\begin{pmatrix}\bar{\bar{x}}_1^{MAX}\\ \bar{\bar{x}}_2^{MAX}\end{pmatrix} &= \begin{pmatrix}\bar{x}_1^{MAX} + \lfloor\Delta x_1^{MAX}/2\rfloor + s_1\\ \bar{x}_2^{MAX} - \lfloor(\Delta x_2^{MAX}+1)/2\rfloor\end{pmatrix}\\ &= \begin{pmatrix}135 + \lfloor 1/2\rfloor + 0\\ 134 - \lfloor(2+1)/2\rfloor\end{pmatrix}\\ &= \begin{pmatrix}135\\ 133\end{pmatrix},\end{aligned}$$

$$\begin{aligned}\begin{pmatrix}\bar{\bar{x}}_1^{min}\\ \bar{\bar{x}}_2^{min}\end{pmatrix} &= \begin{pmatrix}\bar{x}_1^{min} - \lfloor(\Delta x_1^{min}+1)/2\rfloor\\ \bar{x}_2^{min} + \lfloor\Delta x_2^{min}/2\rfloor + s_2\end{pmatrix}\\ &= \begin{pmatrix}130 - \lfloor(2+1)/2\rfloor\\ 132 + \lfloor 2/2\rfloor + 0\end{pmatrix}\\ &= \begin{pmatrix}129\\ 133\end{pmatrix}.\end{aligned}$$

Since $0 \le \bar{\bar{x}}_1^{min}, \bar{\bar{x}}_1^{MAX}, \bar{\bar{x}}_2^{min}, \bar{\bar{x}}_2^{MAX} \le 255$, we have stego-blocks $\ddot{B}_1 = (135, 129, B_1)$ and $\ddot{B}_2 = (133, 133, B_2)$ as shown in Figure 1(b). Blocks 3 and 4 embed a bit of 1 and 0 in the larger and smaller block-mean pixels, respectively. Their embedding results would be $\ddot{B}_3 = (137, 130, B_3)$ and $\ddot{B}_4 = (134, 134, B_4)$. The embedding results of remaining four blocks are shown in Figure 1(b).

i	$\hat{B}_{2i-1}$	$\hat{B}_{2i}$
1	(135,130, B_1)	(134,132, B_2)
2	(136,131, B_3)	(135,133, B_4)
3	(134,132, B_5)	(135,131, B_6)
4	(133,130, B_7)	(135,130, B_8)

(a) Cover image

i	$\ddot{B}_{2i-1}$	$\ddot{B}_{2i}$
1	(135,129, B_1)	(133,133, B_2)
2	(137,130, B_3)	(134,134, B_4)
3	(133,132, B_5)	(135,130, B_6)
4	(132,130, B_7)	(136,131, B_8)

(b) Stego-image

Fig. 1. An embedding example

To extract the embedded message from the stego-image in Figure 1(b), the decoder scans the image as the order in the embedding procedure and calculates

$$\Delta\ddot{x}_1^{MAX} = |\bar{\bar{x}}_1^{MAX} - \bar{\bar{x}}_2^{MAX}| = |135 - 133| = 2,$$

$$\Delta\ddot{x}_1^{min} = |\bar{\bar{x}}_1^{min} - \bar{\bar{x}}_2^{min}| = |129 - 133| = 4,$$

$$\Delta x_1^{MAX} = \lfloor \Delta\ddot{x}_1^{MAX}/2 \rfloor = 1,$$

$$\Delta x_1^{min} = \lfloor \Delta\ddot{x}_1^{min}/2 \rfloor = 2 \text{ and}$$

extracts

$$s_1 = \Delta\ddot{x}_1^{MAX} \bmod 2 = 2 \bmod 2 = 0,$$

$$s_2 = \Delta\ddot{x}_1^{min} \bmod 2 = 4 \bmod 2 = 0.$$

Then calculate

$$\begin{pmatrix} \bar{x}_1^{MAX} \\ \bar{x}_2^{MAX} \end{pmatrix} = \begin{pmatrix} \bar{\bar{x}}_1^{MAX} - \lfloor \Delta x_1^{MAX}/2 \rfloor - s_1 \\ \bar{\bar{x}}_2^{MAX} + \lfloor (\Delta x_1^{MAX} + 1)/2 \rfloor \end{pmatrix} = \begin{pmatrix} 135 - \lfloor 1/2 \rfloor - 0 \\ 133 + \lfloor (1+1)/2 \rfloor \end{pmatrix} = \begin{pmatrix} 135 \\ 134 \end{pmatrix}, \text{ and}$$

$$\begin{pmatrix} \bar{x}_1^{min} \\ \bar{x}_2^{min} \end{pmatrix} = \begin{pmatrix} \bar{\bar{x}}_1^{min} + \lfloor (\Delta x_1^{min} + 1)/2 \rfloor \\ \bar{\bar{x}}_2^{min} - \lfloor \Delta x_1^{min}/2 \rfloor - s_2 \end{pmatrix} = \begin{pmatrix} 129 + \lfloor (2+1)/2 \rfloor \\ 133 - \lfloor 2/2 \rfloor - 0 \end{pmatrix} = \begin{pmatrix} 130 \\ 132 \end{pmatrix}.$$

Finally, the pair of blocks are recovered to $\hat{B}_1 = (135, 130, B_1)$ and $\hat{B}_2 = (134, 132, B_2)$. A bit of 1 is extracted from $s_3 = \Delta\ddot{x}_3^{MAX} \bmod 2 = |\bar{\bar{x}}_3^{MAX} - \bar{\bar{x}}_4^{MAX}| \bmod 2 = |137 - 134| \bmod 2 = 1$. Similarly, a bit of 0 is extracted from $s_4 = |130 - 134| \bmod 2 = 0$. The process continues until the remaining messages are extracted and stego-blocks are recovered. Note that, in the example, none of block index is recorded in the overhead information.

IV. Experimental Results

To show the feasibility and performance of the proposed scheme, we implemented the proposed scheme on a personal computer with Java. The implementation included compressing a grayscale image into a BTC-format image as that in Section II. Test images are shown in Figure 2 and their dimension is 512×512. The block size of BTC-format image is 4×4 pixels. First, a randomly generated message, a binary bit string, was generated and embedded into a test cover image, i.e. the BTC-compressed image. Then we extracted the embedded message from the stego-image and recovered the stego-image to its cover image. The experimental results show that the extracted message is exactly the same as the embedded message and the cover image can be completely recovered. This means our proposed scheme can reversibly embed a message into a BTC-compressed image.

The peak signal noise ratio (PSNR) was used to evaluate the performance of the proposed scheme. It was defined as follows,

$$\text{PSNR} = 10\log_{10}\frac{255^2}{MSE}\text{dB},$$

where MSE is the mean square error. For an image with N pixels, MSE is computed as

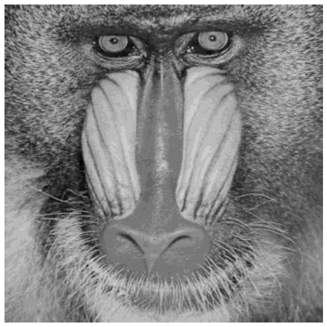
$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_i')^2,$$

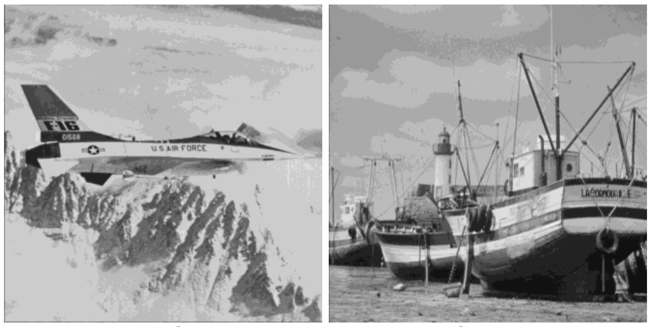
where x_i and x_i' are cover and stego-pixels, respectively.

A larger PSNR implies that a stego-image is more similar to its cover image than a smaller one. It also implies that the visual quality of a stego-image with a higher PSNR is better than that with a smaller one. Researchers usually would like to get an embedding scheme which can obtain a higher PSNR.

Table II shows the visual quality, in terms of PSNR, of a stego-image applying the proposed scheme. When a message is embedded into an image, the image may be distorted by the modification of pixels. The more the messages are embedded, the more the image will be distorted. A good embedding scheme may provide enough embedding space and keep image quality as high as possible. Table II shows that payload capacity is approximately equal to the number of blocks, which means most blocks may embed a message. In addition, the image quality, i.e. PSNR, is more than 28 dB. This shows that a stego-image is similar to its cover image and they may not be distinguishable by human vision.

(a) Lena


(b) Baboon


(c) Airplane (d) Boat

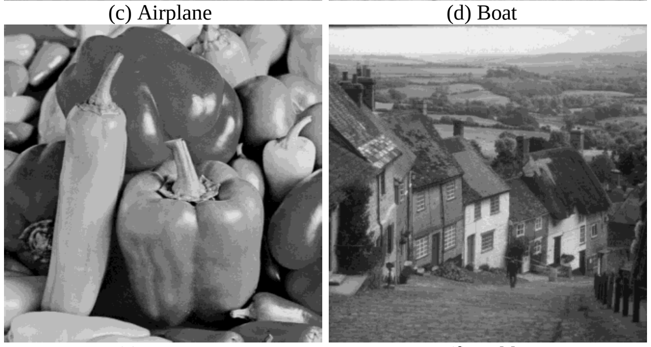

(e) Pepper (f) Gold

Fig. 2. Test images

TABLE II. EMBEDDING PERFORMANCE OF THE PROPOSED SCHEME

Images	PSNR(dB)	Payload(bits)
Lena	28.25	16,272
Baboon	28.03	16,366
Airplane	28.38	16,356
Boat	28.67	16,308
Pepper	29.42	16,016
Gold	29.72	16,368

V. CONCLUSIONS

A reversible data hiding scheme for BTC-compressed image has been proposed. In the proposed scheme, an image is divided into non-overlapping blocks and the BTC algorithm is applied to compress the image. Then a message with two bits is embedded into two blocks by expanding the difference between larger block-mean pair and the one between the smaller block-mean pair. The original BTC-format image may be completely reconstructed after the embedded message is extracted. Experimental results show that the proposed scheme may obtain a stego-image with high visual quality and a payload capacity of one bit per block, approximately. The proposed scheme is a good encoder for applications which need a reversible embedding scheme for BTC-compressed images without complicated computations.

REFERENCES

[1] G. K. Wallace, “The JPEG still picture compression standard,” IEEE Transactions on Consumer Electronics, 38(1), pp. xviii–xxxiv, 1992.

[2] R. Gray, “Vector quantization,” IEEE ASSP Magazine, 1(2), pp.4–29, 1984.

[3] E. Delp and O. Mitchell, “Image compression using block truncation coding,” IEEE Transactions on Communications, 27(9), pp. 1335–1342, 1979.

[4] J. Chen, W. Hong, T.-S. Chen, and C.-W. Shiu, “Steganography for BTC compressed images using no distortion technique,” The Imaging Science Journal, 58(4), pp. 177–185, 2010.

[5] J.-M. Guo and Y.-F. Liu, “High capacity data hiding for error-diffused block truncation coding,” IEEE Transactions on Image Processing, 21(12), pp. 4808–4818, 2012.

[6] D. Ou and W. Sun, “High payload image steganography with minimum distortion based on absolute moment block truncation coding,” Multimedia Tools and Applications, 74(21), pp 9117–9139, 2015.

[7] Y.-C. Chou and H.-H. Chang, “A high payload data hiding scheme for color image based on BTC compression technique,” 2010 Fourth International Conference on Genetic and Evolutionary Computing (ICGEC), pp. 626–629, 2010.

[8] H. Luo, Z. Zhao, and Z.-M. Lu, “Joint secret sharing and data hiding for block truncation coding compressed image transmission,” Information Technology Journal, 10(3), pp.681–685, 2011.

[9] C.-C. Chang, Y.-H. Chen, and C.-C. Lin, “A data embedding scheme for color images based on genetic algorithm and absolute moment block truncation coding,” Soft Computing, 13(4), pp 321–331, 2009.

[10] J. Tian, “Reversible data embedding using a difference expansion,” IEEE Transactions on Circuits Systems for Video Technolgy, 13 (8), pp. 890–896, 2003.

[11] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, “Reversible data hiding,” IEEE Transactions on Circuits and Systems for Video Technology, 16(3), pp. 354–362, 2006.

[12] C.-C. Lin and N.-L. Hsueh, “A lossless data hiding scheme based on three-pixel block differences,” Pattern Recognition, 41(4), pp. 1415–1425, 2008.

[13] C.-C. Chang, C.-C. Lin, C.-S. Tseng, and W.-L. Tai, “Reversible hiding in DCT-based compressed images,” Information Sciences, 177(13), pp. 2768–2786, 2007.