International Journal of Innovative
Computing, Information and Control
Volume **15**, Number **6**, December **2019**

ICIC International ©2019 ISSN 1349-4198
pp. **2211–2227**

A REVERSIBLE VISIBLE WATERMARKING TECHNIQUE FOR BTC-COMPRESSED IMAGES

Shu-Chien Huang

Department of Computer Science
National Pingtung University
No. 4-18, Minsheng Road, Pingtung City, Pingtung County 90003, Taiwan
schuang@mail.nptu.edu.tw

Received February 2019; revised June 2019

Abstract. *A reversible visible watermarking technique is proposed for BTC (Block Truncation Coding) -compressed images in this study. First, the watermark embedding region is determined automatically. Second, the watermark bits are embedded into the compressed codes of blocks which are inside the watermark embedding region in a visible manner. Third, the recovery information is embedded into the compressed codes of blocks which are outside the watermark embedding region in an invisible manner. After the data embedding procedure is executed, the embedded compressed codes still follow the standard format of BTC. The original BTC-compressed image can be recovered without distortion after the watermark is removed. Experimental results demonstrate the effectiveness and superiority of the proposed method.*

Keywords: Reversible watermarking, Visible watermarking, Lossless data embedding, Copyright protection, Block Truncation Coding (BTC)

1. **Introduction.** Image watermarking techniques [1,2] can be used to protect the ownership of valuable images. According to the visibility of the watermark embedded in the protected image, watermark mechanisms are classified as visible or invisible. In general, visible watermark techniques are utilized to indicate the copyright of digital media by embedding an inconspicuous but recognizable logo into media. Watermarks can also be reversible or irreversible, indicating that the original host image can be fully recovered or not, respectively. Among the various types of watermarks, reversible visible watermarks are popular because they are recognizable and the original host image can be recovered after the watermark is removed.

Many visible watermarking algorithms [3-11] have been proposed. Yang et al. [3] proposed a removable visible watermarking algorithm in the Discrete Cosine Transform (DCT) domain. First, the original image is divided into 16×16 blocks and the preprocessed watermark to be embedded is generated by performing element-by-element matrix multiplication. Second, the scaling and embedding factors are computed for each block of the host image and the preprocessed watermark. Third, the DCT coefficients of the preprocessed watermark are adaptively added to those of the host image to yield the watermarked image. In their experiments, legally recovered images can achieve superior visual effects. However, their algorithm needs the original watermark during original image recovery. Lin et al. [4] proposed a removable visible watermarking mechanism in the Discrete Wavelet Transform (DWT) domain. The subsampling technique was adopted to divide the host image into four sub-images. Then DWT is applied to the sub-images to obtain four corresponding decomposed wavelet domains. In the embedding steps, the perceptible weight

DOI: 10.24507/ijicic.15.06.2211

is estimated for the visible watermark. According to the weight, the visible watermark is embedded into the host image. Their method allows the resource provider to adjust the strength of the embedded watermark according to the contrast between the host image and the watermark. In their experiments, the recovered image is nearly identical to the original image. However, their algorithm also needs the original watermark during original image recovery. Chen et al. [5] proposed a reversible and visible watermarking scheme. In their method, the cover image is partitioned into non-overlapped $k \times k$ blocks with $2 \leq k \leq \min\left\{\left\lfloor \frac{M_c}{M_w} \right\rfloor, \left\lfloor \frac{N_c}{N_w} \right\rfloor\right\}$, where $M_c \times N_c$ and $M_w \times N_w$ denote the size of the host image and the size of the binary watermark image, respectively. Each block is then applied to two watermarking schemes: an invisible watermarking scheme and a visible watermarking scheme. Their proposed watermarking scheme embedded one watermark bit into one block of pixels using some parameters to create the visual effect. The invisible watermarking scheme used the difference-expansion method [12] to embed the reversible watermark bit. The exceeding numbers, larger than 255 or smaller than 0, generated from the difference-expansion method need to be recorded for a lossless recovery. Their experimental results showed that not recording any exceeding numbers still results in a high quality recovered host image and similar extracted watermark image.

Many watermarking algorithms based on BTC [13-17] have been proposed in the literature in recent years due to the simplicity of BTC methods. Yang and Yin [16] presented a removable visible watermarking scheme applicable for BTC-compressed images. The visible watermark strength is adaptive to host image content by exploiting image features. Their method can recover the original image nearly losslessly after the watermark bits are extracted. Mohammad et al. [17] proposed a reversible visible watermarking scheme based on the BTC domain. Their scheme used adaptive pixel circular shift operation that adapts to local properties of the image to embed the visible watermark into two quantization levels of BTC according to the parity of the bitplane of BTC triple. In the extraction procedure, the watermark bit can be extracted according to the parity of the number of 1s in the bitplane of the watermarked image. The original BTC-compressed image can be recovered without distortion after watermark removal.

In this paper, a new reversible visible watermarking method is proposed. The BTC-compressed image is used as the host image, and the binary image is used as the watermark. After the data embedding procedure is executed, the watermark can be clearly distinguished by the human eye. The original BTC-compressed image can be recovered without distortion after the binary watermark is removed. The proposed method can provide copyright protection for digital images in real-time environments. The ideas presented in this study can also be applied to many other problems [21]. The remainder of the paper is organized as follows. Some related works are reviewed in Section 2. The proposed method is described in Section 3. Section 4 gives some experimental results. Finally, conclusions are given in Section 5.

2. **Related Works.**

In this section, we review the basic process of BTC [18,19], Chen et al.'s [20] BTC data hiding scheme, and Mohammad et al.'s [17] lossless visible watermarking scheme.

2.1. **Block Truncation Coding (BTC).**

Block truncation coding [18,19] is an efficient lossy compression technique. In BTC, a grayscale image is divided into non-overlapping blocks of size $L \times L$. The value of L is set to 4 or 8 in most cases. For each block with size 4×4, its mean value mv is computed as follows:

$$mv = \frac{1}{4 \times 4} \sum_{i=1}^{4} \sum_{j=1}^{4} x(i,j), \tag{1}$$

where $x(i,j)$ indicates the pixel value in the position (i,j) of the block. Then, all pixels within the block are separated into two groups, smaller and greater than or equal to the mean value mv, denoted as G_0 and G_1, respectively. A binary bitmap BM with the same size as the image block is used to record the output bits of BTC compression. The bitmap is generated using the following rule:

$$BM(i,j) = \begin{cases} 1 & \text{if } x(i,j) \geq mv, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $BM(i,j)$ represents the bit in position (i,j) of BM.

Next, two mean values of G_1 and G_0, denoted as XH and XL, are set as the quantization levels used to reconstruct the image. Therefore, a grayscale image block is decomposed to one binary bitmap BM and two quantization levels, XH and XL.

In the decoding procedure, the approximate image block can be reconstructed according to the binary bitmap BM and two quantization levels XH and XL. The reconstructed rule is defined as follows:

$$\tilde{x}(i,j) = \begin{cases} XH & \text{if } BM(i,j) = 1, \\ XL & \text{otherwise,} \end{cases} \tag{3}$$

where $\tilde{x}(i,j)$ denotes the reconstructed pixel value in position (i,j) of the current decoded block. By collecting all the reconstructed image blocks, the decoded image can be obtained.

Example 2.1. *As an example, a block of the image is shown in Figure 1(a). The mean value of this block is 162.813 and the corresponding bitmap for this block is shown in Figure 1(b). In this example, the value of XH is 177 and the value of XL is 148. The reconstructed image block is shown in Figure 1(c).*

153	171	179	188
148	161	171	183
141	151	164	182
135	139	157	182

(a) Original image block

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

(b) Bitmap

148	177	177	177
148	148	177	177
148	148	177	177
148	148	148	177

(c) Reconstructed image block

FIGURE 1. An example of BTC encoding and decoding

2.2. **Chen et al.'s BTC data hiding scheme.** Chen et al. [20] proposed a reversible data hiding scheme that does not distort the host image. In their scheme, one bit of secret data would be embedded in a block if the quantized values were not equal. In the event that the quantized values were equal, the bitmap would be replaced with the secret data. For the i-th block, its compressed code is (XH_i, XL_i, BM_i) with $XL_i \leq XH_i$. Suppose $XL_i \neq XH_i$, if the secret bit is 1, then the original compressed code (XH_i, XL_i, BM_i) becomes $(XL_i, XH_i, \overline{BM_i})$; otherwise, no changes are required. Suppose $XH_i = XL_i$, the bitmap BM_i would be replaced with the secret data.

To recover the secret data, the process is similar to the embedding process. Let the compressed code be (XH_i, XL_i, BM_i). If $XH_i > XL_i$, then the secret data is 0. If

$XH_i < XL_i$, then the secret data is 1. If $XH_i = XL_i$, then bitmap BM_i contains the secret data.

2.3. **Mohammad et al.'s lossless visible watermarking scheme.** Mohammad et al. [17] provided a lossless visible watermarking scheme for BTC-compressed images. This subsection only describes the steps for the watermark embedding scheme as follows.

Step 1: Let I be a gray-level image ($M \times N$) and divide it into non-overlapping blocks of size $L \times L$. The block size is set to 3×3 in their experiments. The number of blocks is $m \times n$. For each block, the compressed code is (XH, XL, BM).

Step 2: A binary watermark image $W = \{W(k)|W(k) \in \{0,1\}, k = 1, \ldots, m \times n\}$.

Step 3: The smoothness factor of each block can be calculated as follows:

$$Sf_k = \frac{|L \times L - 2 \times q_k|}{L \times L}, \tag{4}$$

where q_k is the number of 1-bits in the k-th image block, $k = 1, 2, \ldots, m \times n$. The bit depth Bd_k of the k-th image block is calculated by the following equation

$$Bd_k = \text{round}\left\{(r-1) \times \frac{\frac{1}{Sf_k} - \min\left(\frac{1}{Sf_k}\right)}{\max\left(\frac{1}{Sf_k}\right) - \min\left(\frac{1}{Sf_k}\right)} + 1\right\}, \tag{5}$$

where r is set to 4 in their experiments. Function round(z), min(z) and max(z) return the nearest integer, the minimum and maximum of the array z respectively.

Step 4: The corresponding watermark bit is embedded by the modification of the BTC compressed code, and it can be written by the following equation

$$xx = \begin{cases} LS(xx, Bd_k) \text{ and } QB(BM, 1) & \text{if } W(k) = 1 \text{ and } EB(BM) = 0, \\ LS(xx, Bd_k) & \text{if } W(k) = 1 \text{ and } EB(BM) = 1, \\ xx \text{ and } QB(BM, 0) & \text{if } W(k) = 0 \text{ and } EB(BM) = 1, \\ xx & \text{if } W(k) = 0 \text{ and } EB(BM) = 0, \end{cases} \tag{6}$$

where $xx \in \{XH, XL\}$, LS denotes circular left shift operation. Function $QB(BM, 1)$ lets the number of 1s in bitplane BM be even by the logical Not operation, while function $QB(BM, 0)$ lets the number of 1s in bitplane BM be odd by the logical Not operation. The values of XH and XL will be exchanged when the $QB(BM, 1)$ or $QB(BM, 0)$ operation is applied. Function $EB(BM)$ has value 1 when the number of 1s in bitplane BM is even, otherwise 0.

Step 5: Repeat Step 3 and Step 4 until all the watermark bits are embedded and finally the watermarked image is generated.

In the proposed method, the recovery information is embedded in an invisible manner according to the method described in Section 2.2. In the experiments, the performance of the proposed method is compared with the performance of Mohammad et al.'s method described in Section 2.3.

3. **The Proposed Method.** Let I be the gray-level image with size $M \times N$ and divide it into non-overlapping blocks of size $L \times L$. Each block is denoted by B_{ij}, $i = 0, 1, \ldots, m-1$, $j = 0, 1, \ldots, n-1$, where $m = \left\lfloor \frac{M}{L} \right\rfloor$ and $n = \left\lfloor \frac{N}{L} \right\rfloor$. For the block B_{ij}, the compressed code is (XH_{ij}, XL_{ij}, BM_{ij}). $W = \{W_{ij}|W_{ij} \in \{0,1\}, i = 0, 1, \ldots, m-1, j = 0, 1, \ldots, n-1\}$ is the binary watermark image with size $m \times n$, and $W_{ij} = 0$ denotes a white pixel as background information and $W_{ij} = 1$ denotes a black pixel as logo information. The flowchart of visible watermark embedding is illustrated in Figure 2(a), and the flowchart of watermark extraction and host image recovery is illustrated in Figure 2(b). The details

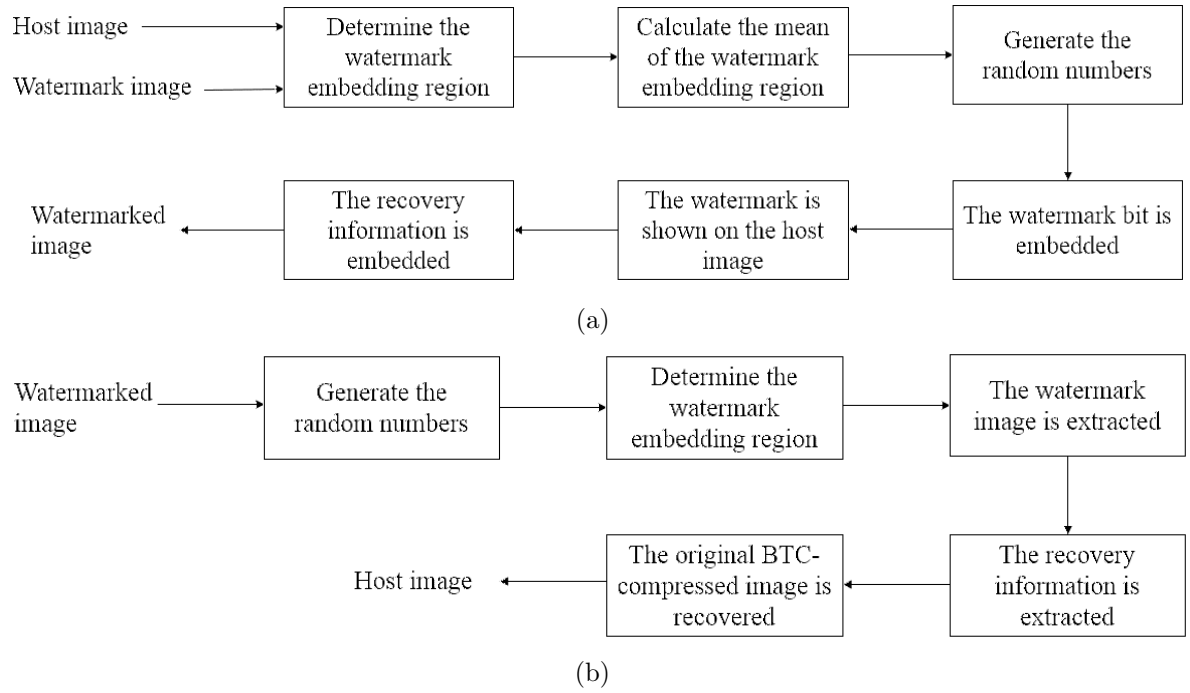


FIGURE 2. (a) The flowchart of visible watermark embedding, (b) the flowchart of watermark extraction and host image recovery

of visible watermark embedding, watermark extraction and original image recovery are described in the following subsections.

3.1. **The visible watermark embedding algorithm.** The steps of visible watermark embedding are described as follows.

Step 1: Determine the watermark embedding region.

For the binary watermark image, all black pixels can be contained in a rectangle. For the watermark W shown in Figure 3, the coordinate of the top-left point of the rectangle is $(x1, y1)$, and the coordinate of the bottom-right point of the rectangle is $(x2, y2)$. The watermark bit W_{ij} corresponds to the block B_{ij} of the image I. For the pixels inside this rectangle, the corresponding part in the image I is the watermark embedding region. The values of $x1$, $y1$, $x2$, and $y2$ are determined automatically in the proposed method.

Step 2: Calculate the mean of the watermark embedding region.

For the block B_{ij}, its mean value $meanB_{ij}$ can be computed by

$$meanB_{ij} = (XH_{ij} \times N1_{ij} + XL_{ij} \times N0_{ij})/(L \times L), \tag{7}$$

where $N1_{ij}$ denotes the number of 1s in bitmap BM_{ij} and $N0_{ij}$ denotes the number of 0s in bitmap BM_{ij}. Therefore, the mean of the watermark embedding region, denoted by $ERmean$, can be calculated according to Algorithm 1.

Algorithm 1

```
procedure ERmean_computation()
  ERmean = 0;
  for (i = x1; i ≤ x2; i++)
    for (j = y1; j ≤ y2; j++)
      {ERmean = ERmean + meanB_ij;}
  ERmean = ERmean/((x2 − x1 + 1) × (y2 − y1 + 1));
end procedure
```

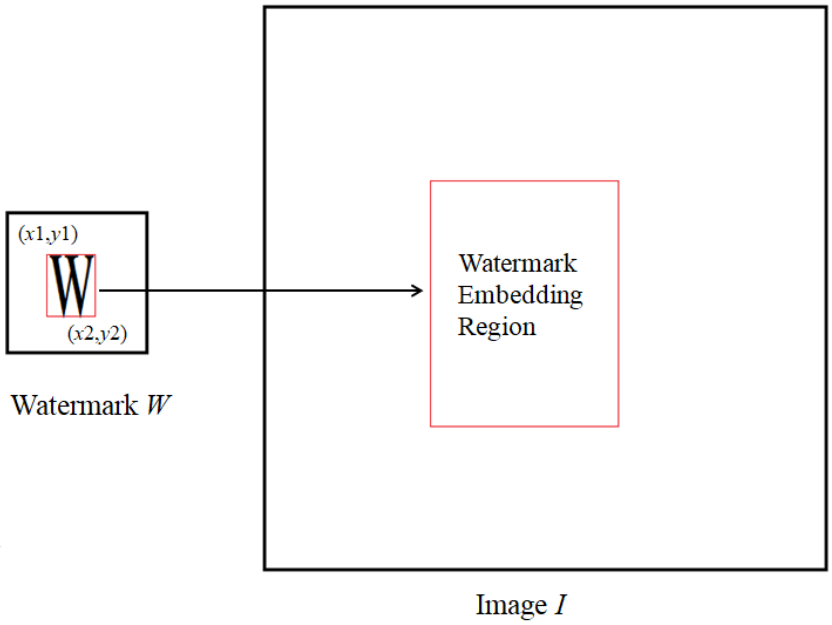


FIGURE 3. Watermark embedding region

Step 3: Use a secret key K to generate the random numbers $rnum_{ij}$, $i = 0, 1, \ldots, m-1$, $j = 0, 1, \ldots, n-1$.

Step 4: For the block B_{ij} which is inside the watermark embedding region, the corresponding watermark bit W_{ij} is embedded according to Algorithm 2.

Algorithm 2

```
procedure Watermark_embedding()
 for (i = x1; i ≤ x2; i++)
   for (j = y1; j ≤ y2; j++)
   {  p = rnum_ij%(L × L);
      row = ⌊p/L⌋; col = p%L;
      If (BM_ij[row][col] ≠ W_ij)
      {BM_ij = BM̄_ij;
         The values of XH_ij and XL_ij are exchanged;
      }
   }
end procedure
```

Example 3.1. *Suppose the quantized values* $XH_{ij} = 150$ *and* $XL_{ij} = 141$ *with bitmap* BM_{ij} *as shown in Figure 4. If the value of* $p = rnum_{ij}\%(L \times L)$ *is equal to 11, then the resulting values of row and col are 2 and 3, respectively. It is observed that the value of* $BM_{ij}[2][3]$ *is equal to 0. If the watermark bit* $W_{ij} = 1$*, then the logical NOT operation is applied to* BM_{ij} *and the values of* XH_{ij} *and* XL_{ij} *are exchanged because* $BM_{ij}[2][3] \neq W_{ij}$*. If the watermark bit* $W_{ij} = 0$*, then the* BM_{ij}*,* XH_{ij} *and* XL_{ij} *are kept unchanged because* $BM_{ij}[2][3] = W_{ij}$*.*

Step 5: In order to show the watermark on the host image, the values of XH_{ij} and XL_{ij} are modified according to Algorithm 3.

If the watermark bit $W_{ij} = 1$, it is obvious that the values of XH_{ij} and XL_{ij} will be modified. If the value of *ERmean* is greater than or equal to 128, the values of XH_{ij} and XL_{ij} will be modified to be less than 128. If the value of *ERmean* is less than 128, the values of XH_{ij} and XL_{ij} will be modified to be greater than or equal to 128. This scheme

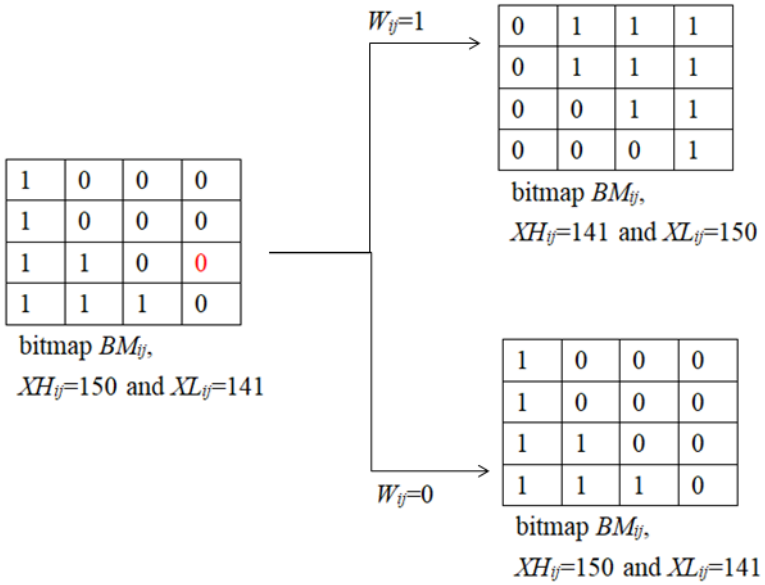


FIGURE 4. The watermark bit is embedded.

Algorithm 3

```
procedure Show_watermark()
 rb = 0;
 for (i = x1; i ≤ x2; i++)
    for (j = y1; j ≤ y2; j++)
    {   If (W_ij == 1)
        { Rbit[rb] = XH_ij%2;
          Rbit[rb + 1] = XL_ij%2;
          rb = rb + 2;
          if (ERmean ≥ 128)
          {XH_ij = ⌊XH_ij/2⌋; XL_ij = ⌊XL_ij/2⌋; }
          else
          {XH_ij = ⌊XH_ij/2⌋ + 128; XL_ij = ⌊XL_ij/2⌋ + 128; }
        }
    }
end procedure
```

is designed to form a bright watermark on the dark areas in the host image, and a dark watermark on the bright areas in the host image.

The array *Rbit* is used to store data for the recovery of the values of XH_{ij} and XL_{ij}. It is noted that each element of the array *Rbit* is 0 or 1. The size of the array *Rbit* is 2× (the number of black pixels in the binary watermark image W), where $W_{ij} = 1$ denotes a black pixel.

Example 3.2. *Suppose the quantized values* $XH_{ij} = 150$ *and* $XL_{ij} = 141$ *with the watermark bit* $W_{ij} = 1$ *as shown in Figure 5. It can be computed that the values of* $Rbit[rb]$ *and* $Rbit[rb + 1]$ *are 0 and 1, respectively. If the value of ERmean is equal to 160, then the values of* XH_{ij} *and* XL_{ij} *will be modified to be 75 and 70, respectively. If the value of ERmean is equal to 90, then the values of* XH_{ij} *and* XL_{ij} *will be modified to be 203 and 198, respectively.*

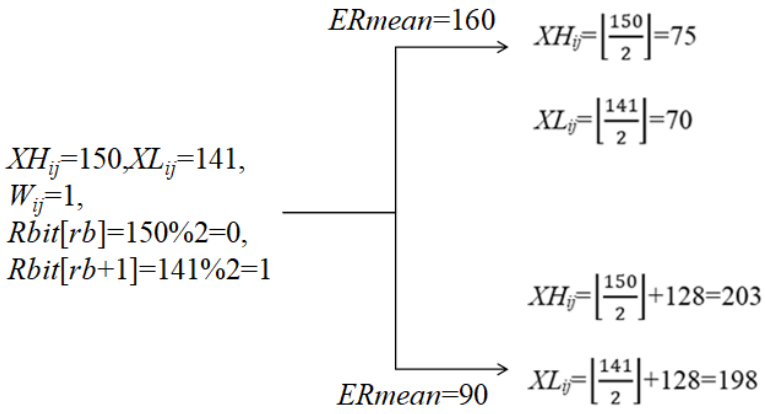


FIGURE 5. The values of XH_{ij} and XL_{ij} are modified

Step 6: The values of $x1$, $y1$, $x2$, and $y2$ are transformed to a binary string, where each value of $x1$, $y1$, $x2$, and $y2$ is represented by 8 bits in the proposed method. Therefore, the length of the binary string is 32. These 32 bits are embedded into the compressed codes of the top blocks according to the method described in Section 2.2. Certainly, the values of W_{0j} are set to 0 for $j = 0, 1, \ldots, n-1$ in the proposed method.

Step 7: Each element of the array *Rbit* is embedded into the compressed codes of the blocks which are not top and are outside the watermark embedding region according to the method described in Section 2.2. Finally the watermarked image is generated.

3.2. **Watermark extraction and original image recovery.** Given the watermarked image, divide it into non-overlapping blocks of size $L \times L$. For the block B_{ij}, the compressed code is $(XH_{ij}, XL_{ij}, BM_{ij})$. The steps for the watermark extraction and the original image recovery are described as follows.

Step 1: Use a secret key K' to generate the random numbers $rnum'_{ij}$, $i = 0, 1, \ldots, m-1$, $j = 0, 1, \ldots, n-1$.

Step 2: Extract 32 bits from the compressed codes of the top blocks to determine the values of $x1$, $y1$, $x2$, and $y2$.

Step 3: The binary watermark image W' is extracted according to Algorithm 4.

Algorithm 4

```
procedure Watermark_extraction()
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
    {  W'_ij = 0;
       if  ((i ≥ x1) and (i ≤ x2) and (j ≥ y1) and (j ≤ y2))
       { p = rnum'_ij %(L × L);
         row = ⌊p/L⌋; col = p%L;
         W'_ij = BM_ij[row][col];
       }
    }
end procedure
```

Example 3.3. *If the block B_{ij} is outside the watermark embedding region, then the extracted watermark bit W'_{ij} is 0. If the block B_{ij} is inside the watermark embedding region, suppose the quantized values $XH_{ij} = 150$ and $XL_{ij} = 141$ with bitmap BM_{ij} as shown in Figure 6. If the value of $p = rnum'_{ij}\%(L \times L)$ is equal to 11, then the resulting values of*

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

bitmap BM_{ij},
XH_{ij}=150 and XL_{ij}=141

FIGURE 6. An example of the watermark bit extraction

row and col are 2 and 3, respectively. It is observed that the value of $BM_{ij}[2][3]$ *is equal to 0. That is, the extracted watermark bit* W'_{ij} *is 0.*

Step 4: Extract each element of the array *Rbit* from the compressed codes of the blocks which are not top and are outside the watermark embedding region according to the method described in Section 2.2. Each element of the array *Rbit* is 0 or 1. The size of the array *Rbit* is 2× (the number of black pixels in the binary watermark image W'), where $W'_{ij} = 1$ denotes a black pixel.

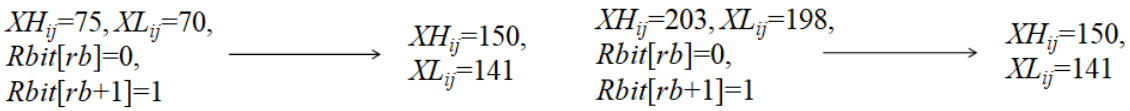
Step 5: The original BTC-compressed image can be recovered according to Algorithm 5.

Algorithm 5

```
procedure Image_recovery()
  rb = 0;
  for (i = 0; i < m; i++)
   for (j = 0; j < n; j++)
   {    if (W'_ij == 1)
        { if (XH_ij ≥ 128)   {XH_ij = (XH_ij − 128) × 2 + Rbit[rb]; }
          else   {XH_ij = XH_ij × 2 + Rbit[rb]; }
          if (XL_ij ≥ 128)   {XL_ij = (XL_ij − 128) × 2 + Rbit[rb + 1]; }
          else   {XL_ij = XL_ij × 2 + Rbit[rb + 1]; }
          rb = rb + 2;
        }
   }
end procedure
```

Example 3.4. *If the extracted watermark bit* $W'_{ij} = 1$*, the values of* XH_{ij} *and* XL_{ij} *will be modified. An example is shown in Figure 7(a). Suppose that the values of* XH_{ij}*,* XL_{ij}*,* $Rbit[rb]$*, and* $Rbit[rb + 1]$ *are 75, 70, 0, and 1, respectively. The value of* XH_{ij} *will be modified to be* $XH_{ij} \times 2 + Rbit[rb] = 75 \times 2 + 0 = 150$ *and the value of* XL_{ij} *will be modified to be* $XL_{ij} \times 2 + Rbit[rb + 1] = 70 \times 2 + 1 = 141$*. Figure 7(b) shows another example. Suppose that the values of* XH_{ij}*,* XL_{ij}*,* $Rbit[rb]$*, and* $Rbit[rb + 1]$ *are 203, 198, 0, and 1, respectively. The value of* XH_{ij} *will be modified to be* $(XH_{ij} - 128) \times 2 + Rbit[rb] = (203 - 128) \times 2 + 0 = 150$ *and the value of* XL_{ij} *will be modified to be* $(XL_{ij} - 128) \times 2 + Rbit[rb + 1] = (198 - 128) \times 2 + 1 = 141$*.*

4. **Experimental Results.** The purpose of this section is to evaluate the performance of the proposed method by experiments and to compare it with the performance of Mohammad et al.'s method [17]. The program is coded in C language and run on a PC with a Pentium 4 CPU. Six gray-level images "Lena", "Baboon", "Airplane", "Lake", "Elaine" and "Truck" shown in Figure 8 are used as the test images, each of which is 512 × 512



(a) (b)

FIGURE 7. Examples of original image recovery

(a) Lena (b) Baboon

(c) Airplane (d) Lake

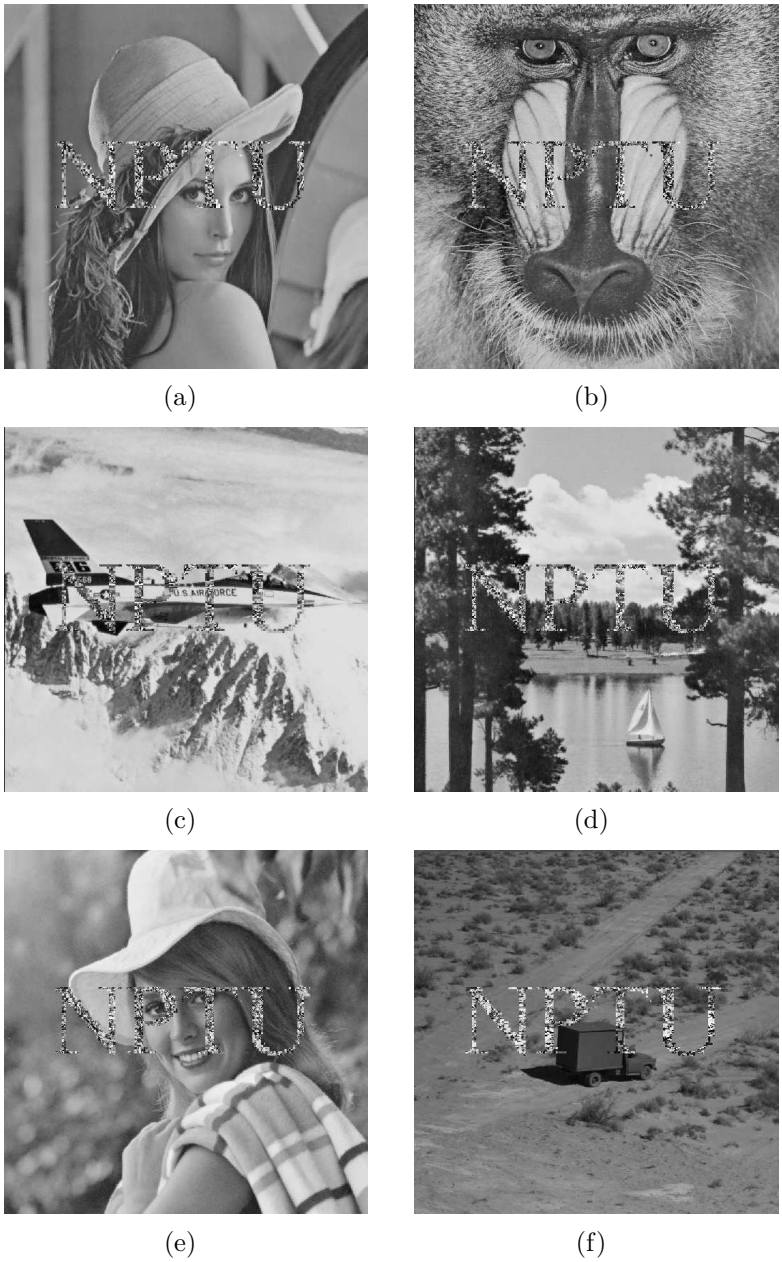
(e) Elaine (f) Truck

FIGURE 8. Six gray-level images

in size. In order to compare with Mohammad et al.'s method, each test image is divided into non-overlapping blocks of size 3×3. The number of blocks is 170×170. Figure 9 shows the binary watermark image W, which is also 170×170. The binary watermark image $W = \{W_{ij} | W_{ij} \in \{0, 1\}, 0 \leq i \leq 169, 0 \leq j \leq 169\}$, where $W_{ij} = 1$ indicates that it

NPTU

FIGURE 9. Watermark

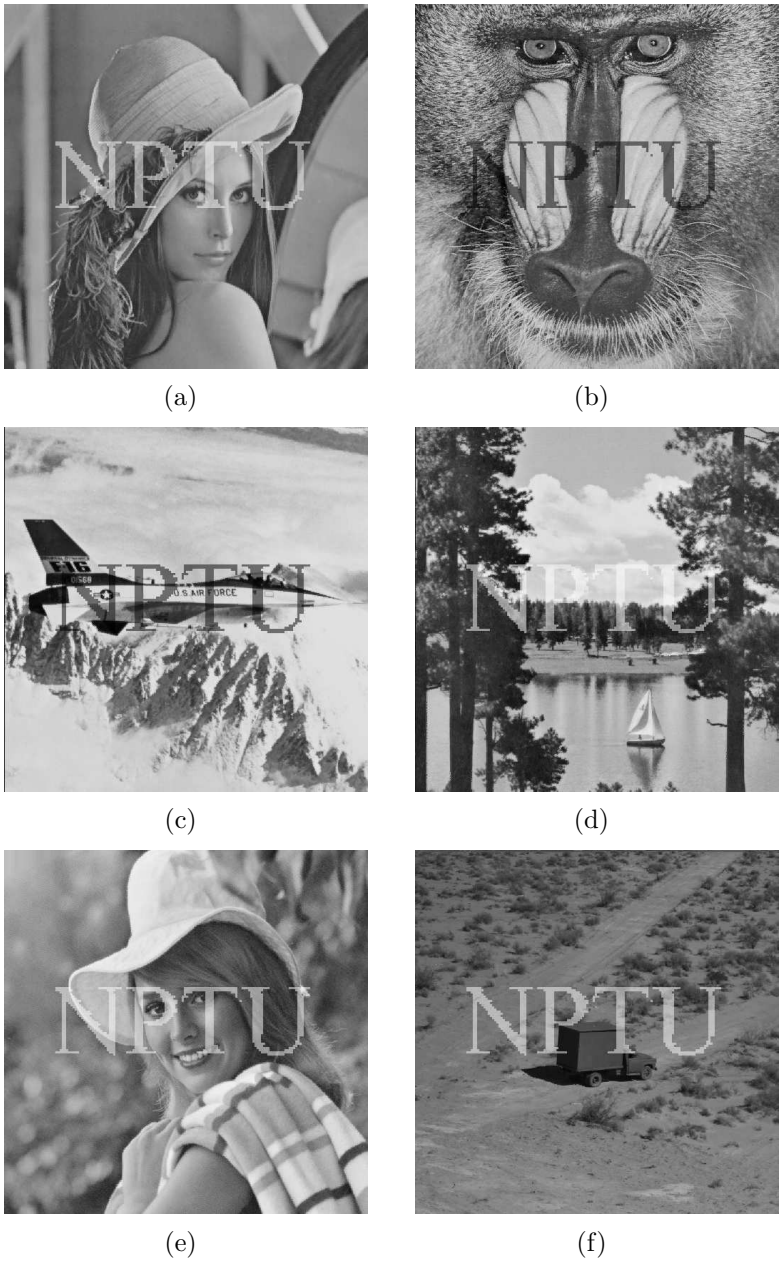


(a) (b) (c) (d) (e) (f)

FIGURE 10. The watermarked images produced by Mohammad et al.'s method

is a black pixel, $W_{ij} = 0$ indicates that it is a white pixel. For this watermark, the values of $x1$, $y1$, $x2$, and $y2$ are 64, 24, 95, and 142, respectively.

Figure 10 shows the watermarked images produced by Mohammad et al.'s method. Figure 11 shows the watermarked images produced by the proposed method. The Peak Signal to Noise Ratio (PSNR) is used to measure the distortion between the original image



(a) (b) (c) (d) (e) (f)

FIGURE 11. The watermarked images produced by the proposed method

and the watermarked image, and is defined as

$$\text{PSNR} = 10 \times \log_{10} \frac{255^2}{\text{MSE}}, \quad \text{MSE} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(I_{ij} - I'_{ij}\right)^2, \tag{8}$$

where I_{ij} denotes the original pixel value and I'_{ij} denotes the modified pixel value. The PSNR comparison with Mohammad et al.'s method is listed in Table 1. From Table 1, it is found that the watermarked image of the proposed method introduces less distortion after watermark insertion compared to Mohammad et al.'s method.

Figure 8(a) is used as the host image and Figure 9 is used as the watermark image; the computation time is illustrated in Table 2. The computation time includes visible watermark embedding and image recovery times. Table 2 shows that the proposed visible watermark embedding algorithm and image recovery algorithm performed efficiently. The

TABLE 1. PSNR comparison result

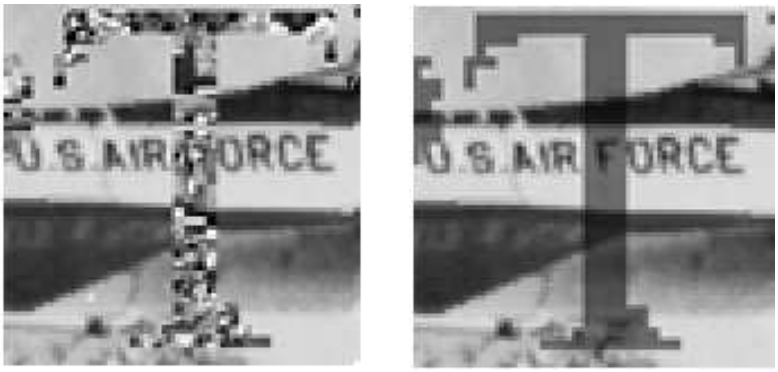
Image	Mohammad et al.'s method [17]	The proposed method
Lena	23.34	24.222
Baboon	21.553	22.718
Airplane	22.25	23.208
Lake	21.597	24.643
Elaine	21.131	24.824
Truck	23.662	24.666

TABLE 2. Computation time

Method	Visible watermark embedding time (sec.)	Image recovery time (sec.)
Mohammad et al.'s method [17]	0.016	0.014
The proposed method	0.01	0.009

computation time of Mohammad et al.'s method is slightly greater than that of the proposed method.

Next, the watermarked images are enlarged to examine the performance of the two methods in terms of watermark transparency. Figure 12(a) and Figure 12(b) present the enlarged partial areas of Figure 10(c) and Figure 11(c), respectively. In Figure 12(b), the character 'F' can be recognized. However, the character 'F' in Figure 12(a) has been seriously obscured. This shows that the proposed method is superior to Mohammad et al.'s method in terms of watermark transparency.

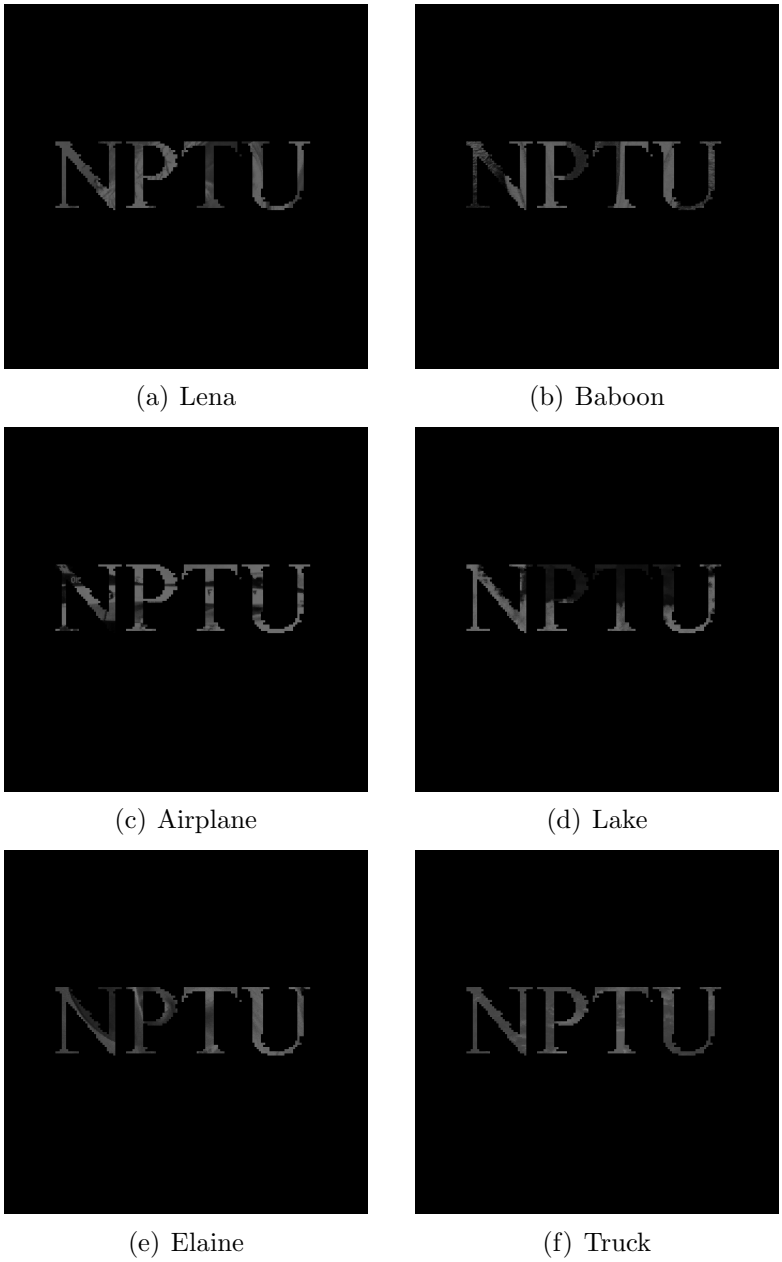


(a) (b)

FIGURE 12. (a) The enlarged partial area of Figure 10(c), (b) the enlarged partial area of Figure 11(c)

In Mohammad et al.'s method, the watermark bit is embedded and the watermark is shown on the host image by using Equation (6). In the proposed method, the watermark is embedded and the watermark is shown on the host image by using Algorithm 2 and Algorithm 3. This is the main difference between the two methods. The reversible visible watermarking technique can provide copyright protection for digital images. In order to obtain a high-quality watermarked image, an efficient method needs to be developed. It is the motivation of this study.

For the host image I and the watermarked image I', the absolute difference images $|I - I'|$ shown in Figures 13(a)-13(f), serve to show the absolute difference between the



(a) Lena (b) Baboon

(c) Airplane (d) Lake

(e) Elaine (f) Truck

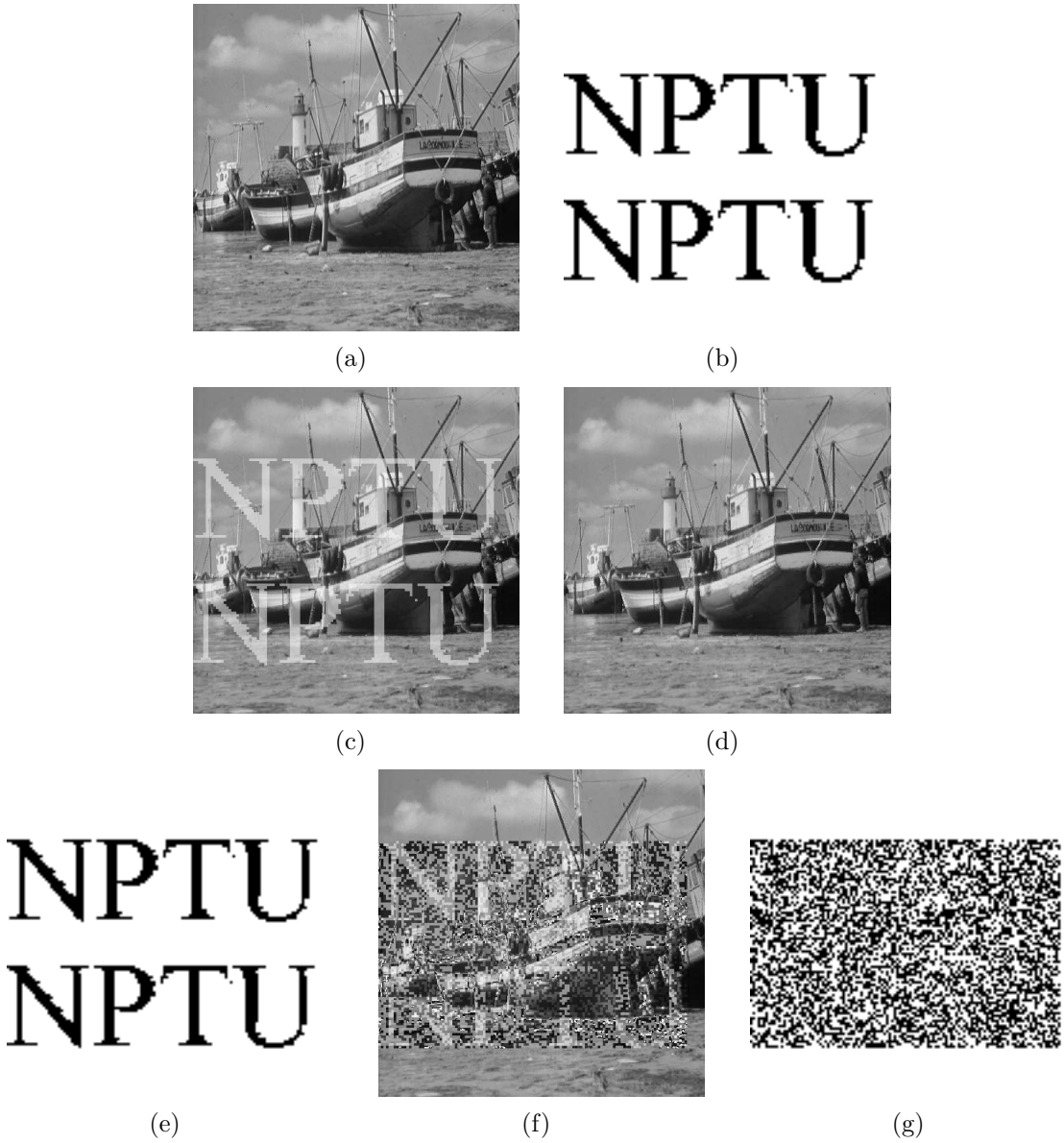
FIGURE 13. The absolute difference images between the host image and the watermarked image

six images before and after the processing in the proposed method. The watermark visibility can be measured by the visible watermark content in the absolute difference image. From Figure 13, it can be seen that the watermarks are apparently recognizable in different types of watermarked images.

The complexity of the computing of Algorithms 1-5 was analyzed through instances of addition/subtraction, multiplication/division, floor functions, and mod functions. Algorithm 1 was computed using $(x2-x1+1)\times(y2-y1+1)+4$ additions/subtractions, and 2 multiplications/divisions. Algorithm 2 was computed using $(x2-x1+1)\times(y2-y1+1)$ floor functions, and $2\times(x2-x1+1)\times(y2-y1+1)$ mod functions. In this study, the number of black pixels in the binary watermark image is denoted by *Nbp*. If the value of *ERmean* is greater than or equal to 128, Algorithm 3 was computed using $2\times$ *Nbp*

additions/subtractions, $2 \times Nbp$ floor functions, and $2 \times Nbp$ mod functions. If the value of *ERmean* is less than 128, Algorithm 3 was computed using $4 \times Nbp$ additions/subtractions, $2 \times Nbp$ floor functions, and $2 \times Nbp$ mod functions. Algorithm 4 was computed using $(x2 - x1 + 1) \times (y2 - y1 + 1)$ floor functions, and $2 \times (x2 - x1 + 1) \times (y2 - y1 + 1)$ mod functions. On average, Algorithm 5 was computed using $5 \times Nbp$ additions/subtractions, and $2 \times Nbp$ multiplications/divisions.

In the experiments, the number of black pixels in the binary watermark image W is 1,046. The size of the array *Rbit* is $2 \times 1{,}046 = 2{,}092$ bits. Each value of $x1$, $y1$, $x2$, and $y2$ is represented by 8 bits in the proposed method. It requires 32 bits to store the values of $x1$, $y1$, $x2$, and $y2$. Therefore, it takes 2,124 bits to store the values of $x1$, $y1$, $x2$, $y2$ and each element of the array *Rbit*. The number of blocks which are inside the



(a) (b) (c) (d) (e) (f) (g)

FIGURE 14. (a) Boat, (b) watermark, (c) watermarked image, (d) recovered image using the correct secret key, (e) extracted watermark image using the correct secret key, (f) recovered image using the wrong secret key, (g) extracted watermark image using the wrong secret key

watermark embedding region is $(95 - 64 + 1) \times (142 - 24 + 1) = 3{,}808$. The number of blocks which are outside the watermark embedding region is $170 \times 170 - 3{,}808 = 25{,}092$. The embedding capacity of each block is greater than or equal to 1 bit according to the method described in Section 2.2. It is obvious that there is enough embedding space to store the values of $x1$, $y1$, $x2$, $y2$ and each element of the array *Rbit* in these experiments.

Figure 14(a) shows the gray-level image "Boat" with size 512×512. This image is divided into non-overlapping blocks of size 4×4. The number of blocks is 128×128. Figure 14(b) shows the binary watermark image W which is also 128×128. Figure 14(c) shows the watermarked image produced by the proposed method. Figure 14(d) and Figure 14(e) show the recovered image and extracted watermark image using the correct secret key, respectively. In contrast, Figure 14(f) and Figure 14(g) show the recovered image and extracted watermark image using the wrong secret key, respectively. These experimental results show that the proposed method is efficient for preventing recovery using the wrong secret key.

5. **Conclusions.** A reversible visible watermarking technique is proposed for BTC-compressed images in this paper. The watermark bits are embedded in a visible manner. The recovery information is embedded in an invisible manner. The proposed scheme is designed to form a bright watermark on the dark areas in the host image, and a dark watermark on the bright areas in the host image. The original BTC-compressed image can be recovered without distortion after the watermark is removed.

The experimental results show that the proposed method not only reveals the visible watermark pattern in a visible manner, but also better preserves the visual quality of the watermarked image. In addition, the recovery process does not require the availability of the original watermark pattern, and the proposed method only allows authorized users with the correct secret key to recover the original image. In the experiments, if the size of the watermark embedding region is nearly equal to the size of the host image, there will not be enough space to store the values of $x1$, $y1$, $x2$, $y2$, and each element of the array *Rbit*. It is the limitation of the proposed method.

Moreover, the computational complexity of BTC coding is much less than DCT and DWT. The reversible visible watermarking technique based on BTC can provide copyright protection for digital images in real-time environments. A future study will consider the development of a reversible visible watermarking scheme for color images.

REFERENCES

[1] N. M. Makbol, B. E. Khoo, T. H. Rassem and K. Loukhaoukha, A new reliable optimized image watermarking scheme based on the integer wavelet transform and singular value decomposition for copyright protection, *Information Sciences*, vol.417, pp.381-400, 2017.

[2] X. L. Liu, C. C. Lin and S. M. Yuan, Blind dual watermarking for color images' authentication and copyright protection, *IEEE Trans. Circuits and Systems for Video Technology*, vol.28, no.5, pp.1047-1055, 2018.

[3] Y. Yang, X. Sun, H. Yang and C. T. Li, Removable visible image watermarking algorithm in the discrete cosine transform domain, *Journal Electronic Imaging*, vol.17, no.3, pp.033008-1-033008-11, 2008.

[4] P. Y. Lin, Y. H. Chen, C. C. Chang and J. S. Lee, Contrast-adaptive removable visible watermarking (CARVW) mechanism, *Image and Vision Computing*, vol.31, no.4, pp.311-321, 2013.

[5] C. C. Chen, Y. H. Tsai and H. C. Yeh, Difference-expansion based reversible and visible image watermarking scheme, *Multimedia Tools and Applications*, vol.76, no.6, pp.8497-8516, 2017.

[6] C. Qin, Z. He, H. Yao, F. Cao and L. Gao, Visible watermark removal scheme based on reversible data hiding and image inpainting, *Signal Processing: Image Communication*, vol.60, pp.160-172, 2018.

[7] Y. Hu and B. Jeon, Reversible visible watermarking and lossless recovery of original images, *IEEE Trans. Circuits and Systems for Video Technology*, vol.16, no.11, pp.1423-1429, 2006.
[8] H. M. Tsai and L. W. Chang, Secure reversible visible image watermarking with authentication, *Signal Processing: Image Communication*, vol.25, no.1, pp.10-17, 2010.
[9] M. J. Tsai, A visible watermarking algorithm based on the content and contrast aware (COCOA) technique, *Journal of Visual Communication and Image Representation*, vol.20, no.5, pp.323-338, 2009.
[10] Y. Yang, X. Sun, H. Yang, C. T. Li and R. Xiao, A contrast-sensitive reversible visible image watermarking technique, *IEEE Trans. Circuits and Systems for Video Technology*, vol.19, no.5, pp.656-667, 2009.
[11] T. Y. Liu and W. H. Tsai, Generic lossless visible watermarking – A new approach, *IEEE Trans. Image Processing*, vol.19, no.5, pp.1224-1235, 2010.
[12] J. Tian, Reversible data embedding using a difference expansion, *IEEE Trans. Circuits and Systems for Video Technology*, vol.13, no.8, pp.890-896, 2003.
[13] I. C. Chang, Y. C. Hu, W. L. Chen and C. C. Lo, High capacity reversible data hiding scheme based on residual histogram shifting for block truncation coding, *Signal Processing*, vol.108, pp.376-388, 2015.
[14] 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 and Applications*, vol.76, no.3, pp.3761-3782, 2017.
[15] W. Hong, S. Zheng, T. S. Chen and C. C. Huang, Reversible data hiding in block truncation coding compressed images using quantization level swapping and shifting, *KSII Trans. Internet and Information Systems*, vol.10, no.6, pp.2817-2834, 2016.
[16] H. Yang and J. Yin, A secure removable visible watermarking for BTC compressed images, *Multimedia Tools and Applications*, vol.74, no.6, pp.1725-1739, 2015.
[17] N. Mohammad, X. Sun, H. Yang, J. Yin, G. Yang and M. Jiang, Lossless visible watermarking based on adaptive circular shift operation for BTC-compressed images, *Multimedia Tools and Applications*, vol.76, no.11, pp.13301-13313, 2017.
[18] E. Delp and O. Mitchell, Image compressions using block truncation coding, *IEEE Trans. Communications*, vol.27, no.9, pp.1335-1342, 1979.
[19] M. Lema and O. Mitchell, Absolute moment block truncation coding and its application to color images, *IEEE Trans. Communications*, vol.32, no.10, pp.1148-1157, 1984.
[20] J. Chen, W. Hong, T. S. Chen and C. W. Shiu, Steganography for BTC compressed images using no distortion technique, *The Imaging Science Journal*, vol.58, no.4, pp.177-185, 2010.
[21] T. Hiraoka, H. Nonaka and Y. Tsurunari, A method for controlling patterns of stripe-patchwork images, *ICIC Express Letters*, vol.13, no.3, pp.175-180, 2019.