International Journal of Electronic and Electrical Engineering.
ISSN 0974-2174 Volume 4, Number 3 (2011), pp. 291-312

http://www.irphouse.com

A New Joint Compression-Watermarking Scheme using Truncation Coding

D. Minola Davids

Research Scholar, Singhania University, Rajasthan, India
E-mail: dminaladavids@yahoo.co.in

Abstract

Block Truncation coding (BTC) is a technique for image compression. To divide the original image into many nonoverlapped blocks, each of which is represented by two distinct values. In traditional BTC, the two values preserve the first- and second-moment characteristics of the original block. Nowadays, most of the multimedia is compressed before it is stored. It is more appropriate to embed information such as a watermark during compression.BTC is a good solution for image or video compression with an extremely low complexity. To improve the bitmap arrangement, error diffusion is used which is called using Error-Diffused Block Truncation Coding (EDBTC), in which the energy-preserving property of EDF is exploited to improve image quality. In the proposed system, Majority-Parity-Guided Error-Diffused Block Truncation Coding (MPG-EDBTC) is applied which embeds the watermark simultaneously during compression, by evaluating the parity value in a predefined Parity-Check Region (PCR). It is found that the false contour and blocking effect that existed in the traditional BTC (Block Truncation Coding) can be removed by this proposed system. The Advanced Encryption Standard (AES) has been included in the proposed system so as to remove the attacks due to intruders.

Keywords: Block Truncation Coding, digital halftoning, digital watermarking, error diffusion

Introduction

Block truncation coding is a type of lossy compression which works by dividing the image into small sub images and then reducing the number of gray levels in each block was proposed by Delp and Mitchell in 1979 [1]. This reduction is performed by a quantizer that adapts to the local image statistics. The levels for quantizer are chosen

to minimize a specified error criterion, and then all the pixel values within each block are mapped to quantized levels.

A technique for image Compression is called Block Truncation coding (BTC) [2]. The numbers of bits required to preserve high dynamic range and resolution in a typical PCM Picture ranges from 106 to 108 .This technique uses a one bit nonparametric quantizer adaptive over local regions of the image. Sub blocks of 4x4 pixels allow compression of about 25% assuming 8-bit integer values are used during transmission or storage. Larger blocks allow greater compression ("a" and "b" values spread over more pixels) however quality also reduces with the increase in block size due to the nature of the algorithm.

For compression, a 256x256 image is divided into blocks of typically 4x4 pixels. For each block the Mean and Standard Deviation are calculated, these values change from block to block. These two values define what values the reconstructed or new block the blocks of the BTC compressed image will all have the same mean and standard deviation of the original image. A two level quantization on the block is where we gain the compression, it is performed as follows:

$$y(i,j)=\begin{cases}1, x(i,j) \succ x' \\ 0, x(i,j) \le x'\end{cases} \tag{1}$$

Where $x(i,j)$ are pixel elements of the original block and $y(i,j)$ are elements of the compressed block. If a pixel value is greater than the mean it is assigned the value "1", otherwise "0". Values equal to the mean can have either a "1" or a "0" depending on the preference of the person or organization implementing the algorithm.

This 16 bit block is stored or transmitted along with the values of Mean and Standard Deviation [15]. Reconstruction is made with two values "a" and "b" which preserve the mean and the standard deviation. The values of "a" and "b" can be computed as follows:

$$a = \bar{x} - \sigma\sqrt{\frac{q}{m-q}}$$
$$b = \bar{x} + \sigma\sqrt{\frac{m-q}{q}} \tag{2}$$

Where σ is the standard deviation, m is the total number of pixels in the block and q is the number of pixels greater than the mean

To reconstruct the image, elements assigned a 0 are replaced with the "a" value and elements assigned a 1 are replaced with the "b" value. This demonstrates that the algorithm is asymmetric in that the encoder has much more work to do than the decoder. This is because the decoder is simply replacing 1's and 0's with the estimated value whereas the encoder is also required to calculate the mean, standard deviation and the two values to use.

$$x(i,j)=\begin{cases}a, y(i,j)=0 \\ b, y(i,j)=1\end{cases} \tag{3}$$

Block truncation coding (BTC)[19] method first divides the image to be coded into small non-overlapping image blocks. The small blocks are coded one at a time. For each block, the original pixels within the block are coded using a binary bit-map the same upper mean color size as the original blocks and two mean pixel values. In the original implementation the block mean and the variance of the pixels and used to preserve the first and second moment of the blocks. The method first computes the mean pixel value of the whole block and then each pixel in that block is compared to the block mean. Binary images can be classified as either halftone or non-halftone. Halftone images are binary representations of grayscale images. Halftoning techniques simulate shades of gray by scattering proper amounts of black and white pixels [12]. Error diffusion is a popular halftoning algorithm that in its most widely used form is inherently serial [11].

The image is divided into blocks of m pixels and each block is processed separately. Pixels with values less than mean value are set to '0'and those with values greater than are equal to the mean value are set to '1'.The bit '0' is set to a, and the bit '1' is set to b. The pixel values greater than the quantization threshold are marked as '1's and others are marked as '0's

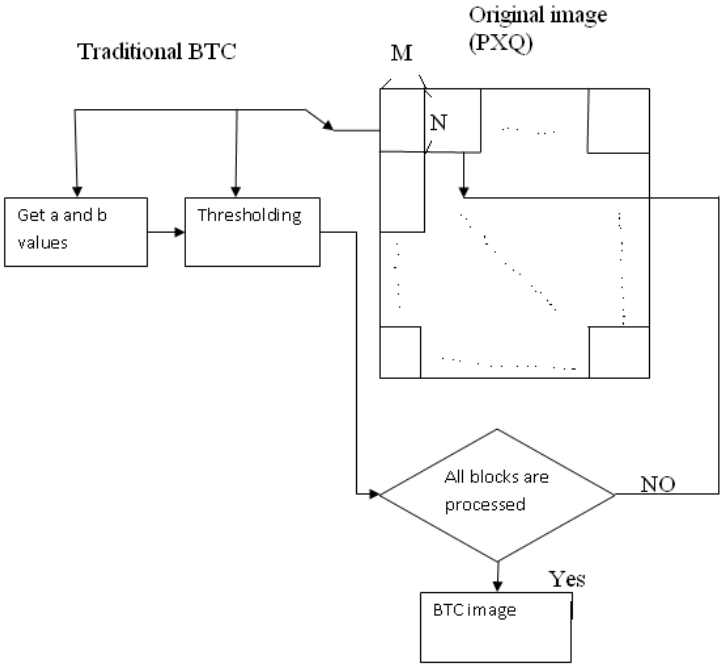


Figure 1: Algorithms of traditional BTC.

Error-Diffused Block Truncation Coding

Block Truncation Coding (BTC) is a technique for grayscale image compression. To divide the original image into many non-overlapped blocks, each of which is represented by two distinct values. When a BTC image is transmitted, each pair of

values (2×8 bits/block) and the bitmap which stores the arrangement of the two values in each block (1 bit/pixel) are required.

To remove the false contour and blocking effect that traditional BTC (Block Truncation Coding).In proposed system, to improve the bitmap arrangement, error diffusion is used which is called using Error-Diffused Block Truncation Coding (EDBTC)[4] and ordered dither Block Truncation Coding (ODBTC)[9],in which the energy-preserving property of EDF is exploited to improve image quality. Binary images can be classified as either halftone or non-halftone. Halftone images are binary representations of grayscale images. Halftoning techniques simulate shades of gray by scattering proper amounts of black and white pixels.

Digital watermarking is a value-added technique for providing copyright protection .Nowadays, it is impossible to store or transmit an image or a video sequence without prior compression. BTC is a good solution for image/video compression with an extremely low complexity. In this proposed system, a watermarking, namely majority-parity-guided error-diffused block truncation coding (MPG-EDBTC), is proposed, in which the energy-preserving property of EDF is exploited to improve image quality.

In this error-diffused block truncation coding proposed system error diffusion (EDF) maintains the local grey level after converting the pixel value into a two-tone result. The original image of size P× Q is divided into nonoverlapped blocks, each of which is size M× N and processed independently [13]. For traditional BTC, the first-moment, second-moment, and the corresponding variance are obtained as

$$\bar{x} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} x_{i,j} \tag{4}$$

$$\bar{x}^2 = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} x^2{}_{i,j} \tag{5}$$

$$\bar{\sigma}^2 = \bar{x}^2 - \left(\bar{x}^2\right) \tag{6}$$

Where $x_{i,j}$ denotes the grayscale value of the original image. The concept of traditional BTC is to preserve the first-and second-moments of a block when original value is substituted by its high- or low-means.

$$m\bar{x} = (m-q)a + qb \tag{7}$$

$$m\bar{x}^2 = (m-q)a^2 + qb^2 \tag{8}$$

Here $m = M \times N$ and q is the number of pixels greater than $\bar{x}$ .The high-and low-means can be evaluated as

$$a = \bar{x} - \bar{\sigma}\sqrt{\frac{q}{m-q}} \tag{9}$$

$$b = \bar{x} + \bar{\sigma}\sqrt{\frac{m-q}{q}} \tag{10}$$

Where a and b denote low-mean and high-mean, respectively.

Only the first-order statistical information is preserved, namely the mean of the pixels less than a threshold $\bar{x}_L$,and $\bar{x}_H$ is the pixels that are greater than or equal to the threshold…The new output levels are defined as,

$$a = \bar{x}_L = \frac{1}{m-q}\sum_{i=1}^{M} x(i) where x(i) \le x_{th} \tag{11}$$

$$b = \bar{x}_H = \frac{1}{q}\sum_{i=1}^{M} x(i) where x(i) \ge x_{th} \tag{12}$$

Where M is n×n, is the number of pixels which are greater than the threshold value. Threshold values of quantizer and two reconstructed levels will change as statistical character of a block changes. Furthermore, after the quantization, block will be representing by a n×n mapping matrix. This matrix consists of pixel classifications (bitmap), and representative intensity for each class. Finally, the receiver reconstructed the image block by calculating 'a' and 'b', the put these values in accordance with the code in the bitmap.BTC is a one-bit quantizer, the mean $\bar{x}$ is employed to threshold the block. The binarized result is called bitmap, the arrangement of the two represented values, low-mean and high-mean

$$y_{i,j} = \begin{cases} b, if h_{i,j} = 1, \\ a, if h_{i,j} = 0, \end{cases} where h_{i,j} = \begin{cases} 1, if x_{i,j} \ge \bar{x} \\ 0, if x_{i,j} \prec \bar{x} \end{cases} \tag{13}$$

Where $h_{i,j}$ denotes bitmap, and $y_{i,j}$ denotes the resulted image.

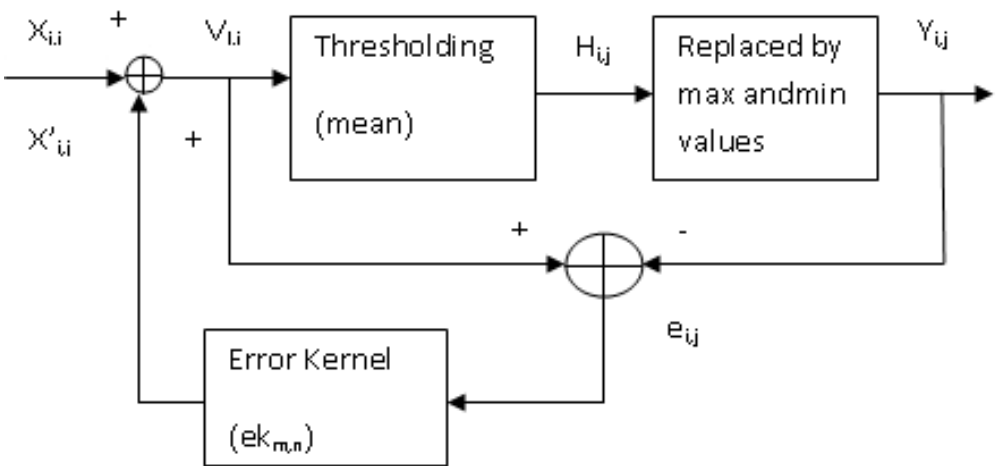


Figure 2: Error-diffuse block truncation coding.

The corresponding variables are defined as

$$v_{i,j} = x_{i,j} + x'_{i,j}$$

$$where x'_{i,j} = \sum_{m=0}^{2}\sum_{n=-2}^{2} e_{i+m,j+n} \times ek_{m,n} \tag{14}$$

$$e_{i,j} = v_{i,j} - y_{i,j}$$

$$where y_{i,j} = \begin{cases} x_{\max}, if h_{i,j} = 1 \\ x_{\min}, if h_{i,j} = 0 \end{cases} \tag{15}$$

The variable $x'_{i,j}$ denotes the diffused error sum added up from neighboring processed pixels. The binary result $h_{i,j}$ is replaced by either maximum $(x_{\max})$ or minimum $(x_{\min})$ value of a block. The variable $ek_{m,n}$ denotes the employed error kernel to diffuse the quantized error to its neighboring pixels [8], [18].

The main advantage of EDBTC is to solve the annoying false contour and blocking effect of traditional BTC [5]. Fig. 1 shows the flowchart of BTC and EDBTC. The image is divided into blocks of m pixels and each block is processed separately. Pixels with values less than mean value are set to ‘0’and those with values greater than are equal to the mean value are set to ‘1’.The bit ‘0’ is set to a, and the bit ‘1’ is set to b. The pixel values greater than the quantization threshold are marked as ‘1’s and others are marked as ‘0’s.Suppose the original image of size $P \times Q$ is divided into nonoverlapped blocks, each of which is of size $M \times N$ and processed independently.

Majority-Parity-Guided Error-Diffused Block Truncation Coding

MPG-EDBTC Encoder

The proposed watermarking encoder is developed based upon the framework of EDBTC. Notably, the only different part is that the functional block is replaced by “MPG error diffusion,” which controls the watermark embedding [3]. In addition, one watermark bit associates to one block of the original image. In MPG error diffusion algorithm, watermark is embedded by modifying the bitmap of BTC image [6].

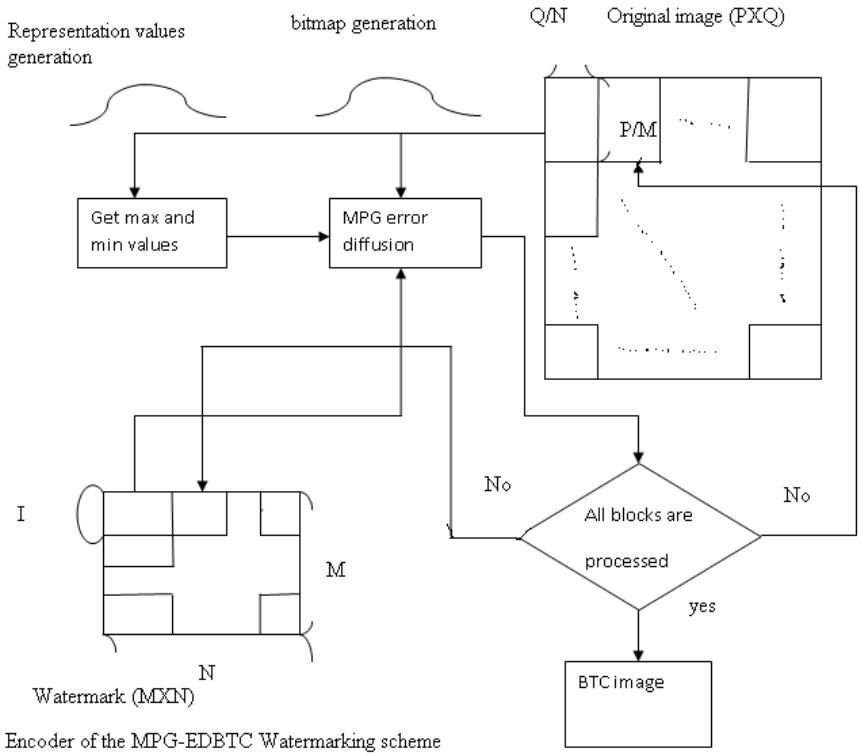


Figure 3: Encoder Of The MPG-EDBTC Watermarking.

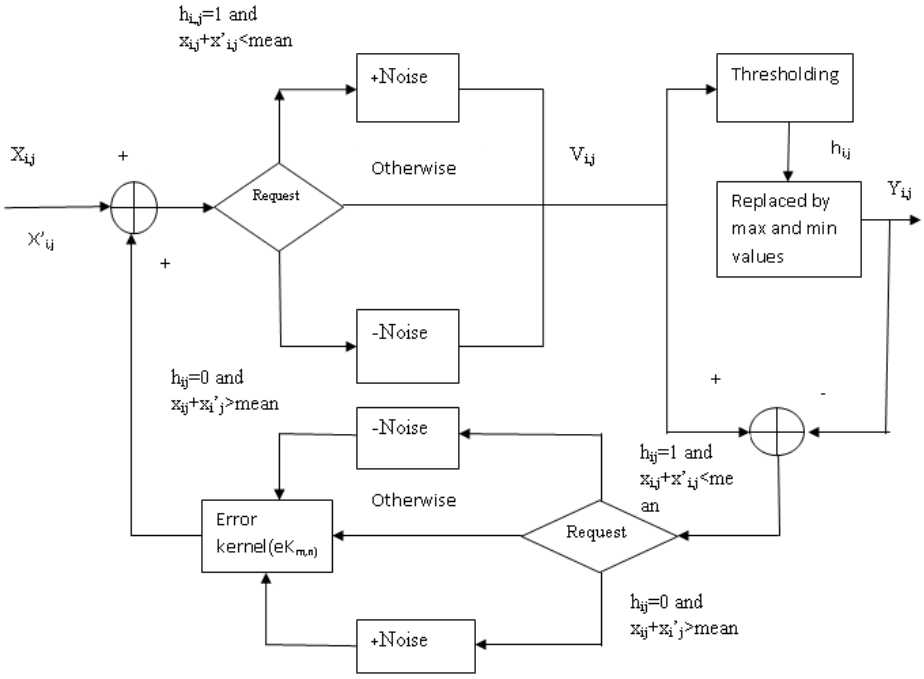


Figure 4: Majority-parity-guided (MPG) error diffusion algorithm.

The corresponding equations are

$$v_{i,j} = x_{i,j} + x'_{i,j} + \varepsilon$$

$$where x'_{i,j} = \sum_{m=0}^{2}\sum_{n=-2}^{2} e_{i+m,j+n} \times ek_{m,n} \tag{16}$$

$$e_{i,j} = v_{i,j} - y_{i,j} - \varepsilon$$

$$where y_{i,j} = \begin{cases} x_{\max}, if h_{i,j} = 1 \\ x_{\min}, if h_{i,j} = 0 \end{cases} \tag{17}$$

$$\varepsilon = \begin{cases} N_i, if the request of h_{i,j} is 1 and x_{i,j} + x'_{i,j} \prec \bar{x} \\ 0, otherwise \\ -N_i, if the request of h_{i,j} is 0 and x_{i,j} + x'_{i,j} \succ \bar{x} \end{cases} \tag{18}$$

Where N_i denotes noise increment.A positive ε increases the probability of $y_{i,j} = x_{\max}$, and conversely,a negative ε Increase the probability of $y_{i,j} = x_{\min}$. In fact; the request of (16) depends upon the watermark and the diffused grayscale image.

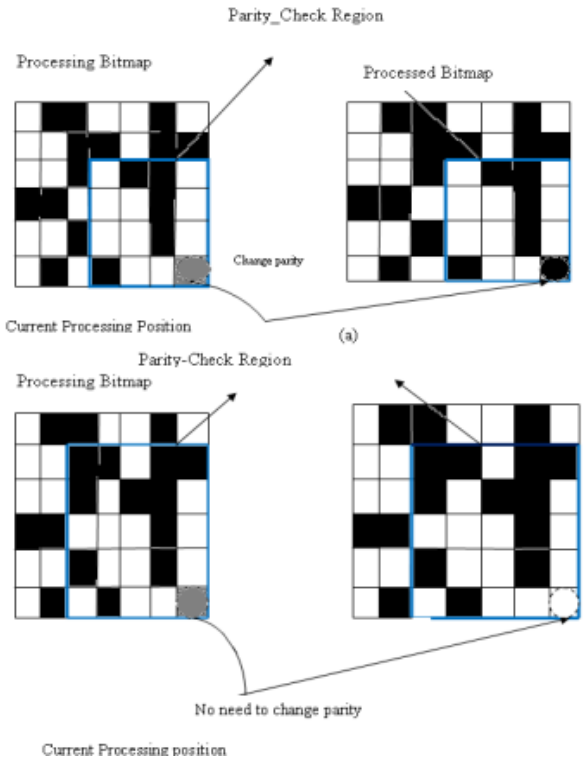


Figure 5: Two distinct watermark bits embedding examples.(a)Black watermark embedding.(with PCR size =4)(b) White watermark embedding.(with PCR size=5).

The binary patterns represent bitmaps, and black and white associate to values 0 and 1, respectively. A black watermark bit is attempted to be embedded to the current processing position, which has not been binarized yet. First, a parity-check region (PCR) with a predefine region of size 4×4 is employed for calculating the parity, which is used for representing the embedded watermark bit. The PCR always covers the top-left processed region relative to the currently position (i,j) . The parity calculation is defined as

$$P_{i,j}=\left(\sum_{m,n\in PCR}\sum\begin{cases}1, if h_{i+m,j+n}=0\\0, otherwise\end{cases}\right)\bmod 2 \tag{19}$$

Where $h_{i,j}$ denotes the bitmap. The PCR excludes the position (i,j).The $P_{i,j}=5\bmod 2=1$ is derived. Thus the calculated parity has to be modified to 0 for representing correct watermark information. If the diffused grayscale value $\left(x_{i,j}+x'_{i,j}\right)$ is greater than mean, an additive noise "-Noise" is added on the current processing position to make the bitmap has higher probability becoming 0.Conversely,if the diffused grayscale value $\left(x_{i,j}+x'_{i,j}\right)$ is less than mean, the current bitmap will become 0 without any noise adding.

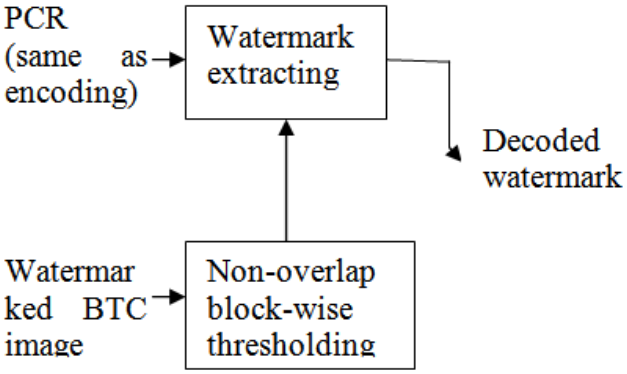


Figure 6: MPG-EDBTC decoding procedure.

First, the bitmap of the received watermarked EDBTC [14] image is extracted finding the positions of the local minimum and maximum directly, a simple thresholding method is employed for producing a robust temporary bitmap as

$$\bar{x}=\frac{1}{\left(\frac{P}{M}\right)\times\left(\frac{Q}{N}\right)}\sum_{m=1}^{P/M}\sum_{n=1}^{Q/N}y_{i+m,j+n} \tag{20}$$

$$h_{i+m,j+n} = \begin{cases} 0, if y_{i+m,j+n} \prec \bar{y} \\ 1, if y_{i+m,j+n} \geq \bar{y} \end{cases} \tag{21}$$

Where $y_{i,j}$ and $h_{i,j}$ denote the watermarked BTC image and the extracted bitmap. $\left(P/M\right)\times\left(Q/N\right)$ denotes the block size[4]. When the parity of the current processing position of the bitmap on the right hand side of Black watermark embedding is calculated, the correct watermark can be yielded since $P_{i,j} = 6 \bmod 2 = 0$.The PCR includes the current processed position (i, j) for watermark bit with a different PCR of size 5.The calculated parity $P_{i,j} = 11 \bmod 2 = 1$ is equal to the white watermark bit1.If the diffused grayscale value is lower than mean, an additive noise "+Noise" is added on the current processing position to make the bitmap has higher probability becoming 1.Conversely,if the diffused grayscale value is higher than mean, the current bitmap will become 1 without any noise adding.

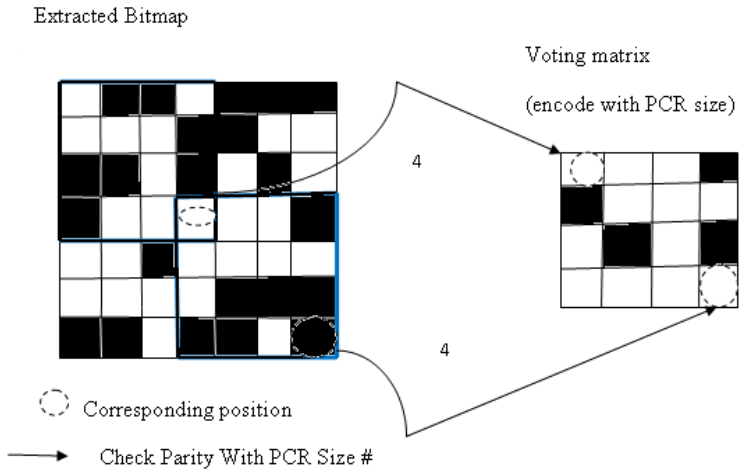


Figure 7: Example of watermark extracting.

The extracted bitmap on the left-hand side as is employed to yield a voting matrix on the right-hand side by parity determination with PCR of size 4 as

$$v_{i,j} = \left(\sum_{m,n \in PCR} \sum \begin{cases} 1, if h_{i+m,j+n} = 0 \\ 0, if h_{i+m,j+n} = 1 \end{cases} \right) \bmod 2 \tag{22}$$

Where $v_{i,j}$ denotes the pixels in the voting matrix. Since a watermark bit is embedded into a 4×4 block, the majority voting scheme is employed to produce a robust decoded watermark [17].

$$w_{i,j}\begin{cases}0, if \sum_{m=1}^{P/M}\sum_{n=1}^{Q/N} v_{i+m,j+n} \prec \dfrac{\left(P/M\right)\times\left(Q/N\right)}{2}\\ 1, otherwise\end{cases} \tag{23}$$

Where the $w_{i,j}$ denotes the decoded watermark, in which 0 and 1 represent white and black pixels.

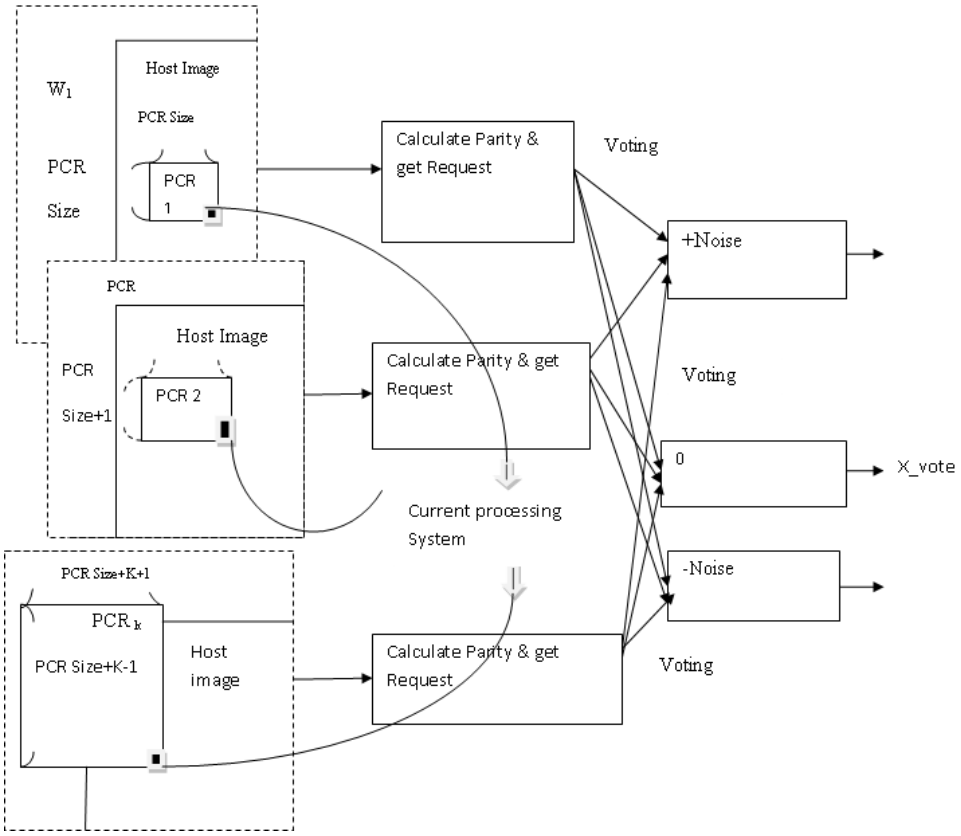


Figure 8: Multiple Watermarks embedding Mechanism.

The proposed MPG-EDBTC can be extended for multiple watermarks embedding [7]. It embedding mechanism where k watermark bits are embedded to the current processing position simultaneously. If the PCR exceeds the boundary of the host image, such as the case of w_k watermark embedding in the bottom of all the exceeding areas are set as 0 for parity calculating. In addition, each embedding has identical request for bitmap output. All the requests from the $k\left(= p_{vote} + x_{vote} + n_{vote}\right)$ watermarks are collected, and the two values, p_{vote} and n_{vote} ,are employed to make the final decision by selecting "+Noise," "-Noise," as

$$\varepsilon = \begin{cases} N_i, if p_{vote} \succ n_{vote} and x_{vote} \\ 0, otherwise \\ -N_i, if n_{vote} \succ p_{vote} and x_{vote} \end{cases} \quad (24)$$

Where $g_{m,n}$ denotes a Gaussian filter (GF) for simulating the low pass characteristic of HVS, and R denotes the support region. The support region size is affected by the viewing distance

$$N_v = r \times dpi \times \frac{cm}{inch}, where r = 2 \times d \times \tan\left(\frac{\theta}{2}\right) \quad (25)$$

Where $\theta = 1^0$ denotes one viewing degree; d denotes the viewing distance (cm); r denotes the viewed width.

Performance Evaluation

The performance evaluation is the BER, which determines the difference between the original watermark $w_{i,j}$ and the corresponding decoded watermark $\hat{w}_{i,j}$.Suppose a watermark is of size $M \times N$ and in binary fashion, the BER is defined as

$$BER = \frac{1}{M \times N}\left(\sum_{i=1}^{M}\sum_{j=1}^{N} w_{i,j} \oplus \hat{w}_{i,j}\right) \quad (26)$$

Where $\oplus$ denotes exclusive OR (XOR) operation.

For multiple watermarks decoding, each watermark decoding shares the same method, MPG-EDBTC decoding procedure, The PCR size of each watermark extracting has to be synchronized in encoder and decoder and, the PCR size is carried.

In this system, two performance evaluation approaches, which include human visual system peak signal-to noise ratio (HVS-PSNR) and bit error rate (BER). Let's denote the original image as $x_{i,j}$ and its corresponding altered image as $y_{i,j}$. Suppose an image is of size $P \times Q$, the quality evaluation is defined as

$$HVS\text{–}PSNR = 10\log_0 \times \frac{P \times Q \times 255^2}{\sum_{i=1}^{P}\sum_{j=1}^{Q}\left[\sum_{m,n \in R} g_{m,n}\left(x_{i+m,j+n} - y_{i+m,j+n}\right)\right]^2} \quad (27)$$

Advanced Encryption Standard

The advanced encryption standard (AES), also known as the Rijndael algorithm, is a block cipher that can encrypt data blocks of 128,192 or 256 bits using symmetric keys

of 128,192 or 256 bits [10]. Due to its performance, security, efficiency, ease of implementation and flexibility Rijndael was chosen for AES standard. Our initial investigation will only focus on the 128 bit implementation; however, the same techniques are applicable also for bigger key sizes.In AES security the design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use.AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys. By 2006, the best known attacks were on 7 rounds for 128-bit keys, 8 rounds for 192-bit keys, and 9 rounds for 256-bit keys.

AES encryption uses a single key as a part of the encryption process. The key can be 128 bits (16bytes), 192 bits (24 bytes), or 256 bits (32 bytes) in length. The term 128-bit encryption refers to the use of a 128-bit encryption key. With AES both the encryption and the decryption are performed using the same key. This is called a symmetric encryption algorithm. Encryption algorithms that use two different keys, a public and a private key, are called asymmetric encryption algorithms. An encryption key is simply a binary string of data used in the encryption process. Because the same encryption key is used to encrypt and decrypt data, it is important to keep the encryption key a secret and to se keys that are hard to guess. Some keys are generated by software used for this specific task. Another method is to derive a key from a pass phrase. Good encryption systems never use a pass phrase alone as an encryption key. Among the top survivors is the Advanced Encryption Standard (AES). Unlike many encryption algorithms which came before AES and which were intended to be executed by single core microprocessors, AES was invented with hardware offload in mind. Like other encryption algorithms, AES consists of rounds of cyclic processing. However, each round in AES may be executed independently of the previous round without a feedback path, lending itself to hardware parallelization and deep pipelining. Pipeline depth is chosen based on the throughput required and the achievable clock frequency. Also of consideration is the silicon area consumed. The deeper the pipeline, the more encryption table instances required to support parallel accesses. Hardware AES embodiments surpass the throughput performance requirements of modern disk technology with ample headroom, thus such tradeoffs are generally quite painless.

BTC is a simple and fast algorithm which achieves constant bitrates of 2 bits per pixel. The method is however suboptimal. It divides the original image into small-sub images and then using a quantizer, which adapts itself according to the image statistics to reduce the number of gray levels in the image.

Step 1: Open image

Basic on the grey level image is a two-dimensional image, and color is a three-dimensional image. When a image is read, then it will judge it is a gray level or color

image. If read a ‘lena.bmp’ image

Step 2: Determine the block size
An image is divided into non-overlapping blocks. The size of a block could be (4×4) or (8×8) etc.

Step 3: Calculate the average of each block
The first 4×4 block as

$$m = \frac{1}{16}\sum\sum x(i, j) \tag{28}$$

Where $x(i, j)$ represent pixels in a block

Pixels in the image block are then classified into two kinds according to whether their gray level is greater than block. The average gray level of these two kinds of pixel are calculated as

$$m_u = \frac{1}{k}\sum_{x(i,j)<m} x(i, j) \tag{29}$$

$$m_l = \frac{1}{16-k}\sum_{x(i,j)\geq m} x(i, j) \tag{30}$$

Where k is the number of pixels whose graylevel is greater than m

Step 4: Building a bitmap
A binary block, denoted b is also needed to classify the pixels. We can use “1” to represent a pixel whose graylevel is less than or equal to m

The encoder writes m_u , m_l and b to a file. Assume than we use 8 bit to represent $m_u and m_l$ represent. Then the total number of bits required for a block is 8+8+16=32 bits .Thus the bitrates for the very basic BTC algorithm is 2 bits/pixel.

Step 5: Work out two reconstructed level
In the decoder, an image block is reconstructed by replacing the “1”s with m_u and the “0”s by m_l

High-level description of the algorithm

1. KeyExpansion—round keys are derived from the cipher key using Rijndael's key schedule
2. Initial Round
    a. AddRoundKey—each byte of the state is combined with the round key using bitwise xor

3. Rounds

a. SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
b. ShiftRows—a transposition step where each row of the state is shifted cyclically a certain number of steps.
c. MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
d. AddRoundKey

4. Final Round (no MixColumns)
   a. SubBytes
   b. ShiftRows
   c. AddRoundKey

The SubBytes step

In the SubBytes step, each byte in the state is replaced with its entry in a fixed 8-bit $b_{i,j} = s(a_{i,j})$

In the SubBytes step, each byte in the matrix is updated using an 8-bit substitution box, the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF , known to have good non-linearity properties.

The ShiftRows step

In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row.

The Shift Rows step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For the block of size 128 bits and 192 bits the shifting pattern is the same. In the case of the 256-bit block, the first row is unchanged and the shifting for second, third and fourth row is 1 byte, 3 bytes and 4 bytes respectively as AES does not use 256-bit blocks. Here $A_{i,j}$ is from cipher text and $B_{i,j}$ is from key.

The MixColumns step

In the MixColumns step, each column of the state is multiplied with a fixed polynomial $c(x)$ In the MixColumns step, the four bytes of each column of the state are combined using an invertible linear transformation. The MixColumns function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with ShiftRows, MixColumns provides diffusion in the cipher.

The AddRoundKey step

In the AddRoundKey step, each byte of the state is combined with a byte of the round subkey using the XOR operation $(\oplus)$.In the AddRoundKey step, the subkey is

combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise XOR.

Analysis with Noise

Three adjustable parameters, noise increment, PCR size, and watermark size, are involved in the proposed scheme. Extensive experimental results with various parameter configurations. In host images of size 512 × 512 and watermarks of size 64 × 64 in the simulation. The average performance with PCR size image quality, decoded watermark quality, and processing efficiency by adjusting the PCR size. In Average HVS-PSNR of the watermarked images. When, the parity value cannot be changed via the encoding algorithm (because the current processing position is excluded from the PCR). Hence, the whole block is directly affected by the embedded watermark bit (each pixel in a block is added with the same additive noise), which significantly decrease the image quality. As indicated from average HVS-PSNR of the watermarked images a greater PCR size accompanies a better performance in image quality and decoded accuracy, yet which impedes the image size versus processing time(sec) $PCRsize = 4$ is a good choice for the proposed watermarking. In addition, the PCR size also is a secret key for watermark extracting. We suggest that the selected possibilities of PCR sizes are greater or equal to four. Average performance with computational cost indicates that the image quality is proportional to the watermark size, since the block size is $\left(P/M\right)\times\left(Q/N\right)$ the influence of blocking effect is increased with increasing in block size. For BER, the bigger block means that the higher likelihood in obtaining the required parity value is available by a small noise, which is a positive benefit for system performance. The block size also affects the bit rate, which is formulated as follows:

$$Bitrate = \frac{M \times N + 2 \times 8}{M \times N} \tag{31}$$

Where $M \times N$ in denominator denotes block size; $M \times N$ and 2×8 in numerator denote the bitmap and the two represented maximum and minimum values, respectively. In practical application, when data capacity is adjusted, the bit rate issue has to be taken into consideration. Practical embedded images and the corresponding decoded watermarks with different sizes of watermarks shows the practical watermarked images and the corresponding decoded watermarks with different watermark sizes.

According to the experiments of average performance with different PCR size and practical embedded images and the corresponding decoded watermarks with different sizes of watermarks the noise increment is inversely proportional to image quality and BER. The optimal noise increment can be determined from these results to yield acceptable image quality (around 48 dB) and mean2error (around 0.99). Under this

condition, $PCRsize = 4$, watermark size=64×64,and noise increment=20.So far, few former approaches address the issue of embedding watermarks in BTC images. In the watermark embedding is independent of BTC image compression. Hence, the image quality of the obtained watermarked image is identical to that of the original BTC image, while an overhead of the same size as the watermark should be transmitted.

Experimental Results*:* In this section, the proposed MPG-EDBTC is compared with encoding and decoding watermark different embedding schemes with various types of attacks, including lightening, darkening, pulse noising, Gaussian noising, Gaussian smoothing, Two common attacks, JPEG and JPEG2000, are not taken into consideration in our simulation, since BTC is a compression scheme for efficient coding application, and, thus, it is rarely employed to concatenate with another high complexity compression scheme.

Theoretical Analyses

In the theoretical analyses, a grayscale EDBTC image is assumed as uniform distribution without losing generality. To begin with, the MPG-EDBTC is analyzed first. In encoder, the probability that a pixel in a bitmap can correctly embed watermark is formulated

$$p_c(N_i) = \sum_{d=0}^{255} p_r(d) \times \left[H\left(N_i - \frac{d}{2}\right) + \frac{H\left(\frac{d}{2} - N_i\right)}{2} \right] \tag{32}$$

$$p_r(d) = \begin{cases} \dfrac{1}{256}, if d = 0 \\ \dfrac{2 \times (256 - d)}{256^2}, if\ 0 \prec d \le 255 \\ 0, otherwise \end{cases} \tag{33}$$

Where N_i denotes the noise increment; $H(\cdot)$ denotes a unit step function; d denotes the difference between the maximum and minimum values of one BTC block, and $p_r(d)$ denotes the probability of d occurrence. $p_c(N_i)$ is proportional to the noise increment. In a BTC image, each block is represented its maximum and minimum values. If an additional noise increment is higher than the half difference between the two represented values, the noise has sufficient strength to make the diffused grayscale value ($x_{i,j} + x'_{i,j}$) match the expected output $\left(H\left(N_i - \left(d/2\right)\right)\right)$. Otherwise, the correct embedding is determined by whether the diffused grayscale value higher than the mean of its block $\left(\left(H\left(N_i - \left(d/2\right)\right)\right)/2\right)$.

When receive an EDBTC image at decoder, the bitmap is extracted to detect the embedded watermark. This bitmap can be modeled as a Binomial distribution since which only has two types of values, black and white. To calculate the parity with the received PCR, the voting matrix can be obtained. This voting matrix also can be

represented by a Binomial distribution, since it has two possibilities, correct and incorrect embedding. Finally, the extracted watermark is determined by majority voting from the voting matrix. The error probability (BER) of the extracted watermark is formulated as

$$
\begin{aligned}
&p\left(N_c < \frac{M\times N}{2}\right) = \sum_{n=0}^{\frac{M\times N}{2}} \binom{M\times N}{2} \times p_{ext}(PCRS)^n \times \left[1-p_{ext}(PCRS)\right]^{M\times N-n} \\
&p_{ext}(PCRS) \\
&= p_c(N_i)\times\left[(1-p_{atk}\times p_{even}(PCRS)+p_{atk}\times p_{odd}(PCRS)\right]+\left[p_c(N_i)\right] \\
&\times\left[p_{atk}\times p_{even}(PCRS)+(1-p_{atk})\times p_{odd}(PCRS)\right] \\
&p_{even}(PCRS) = \sum_{i=0}^{\frac{PCRS^2-1}{2}} \binom{PCRS^2-1}{i\times 2} \times p_{atk}^{i\times 2} \times \left[1-p_{atk}\right]^{PCRS^2-1-i\times 2} \\
&p_{odd}(PCRS) = \sum_{i=0}^{\frac{PCRS^2-1}{2}} \binom{PCRS^2-1}{i\times 2+1} \times p_{atk}^{i\times 2+1} \times \left[1-p_{atk}\right]^{PCRS^2-1-(i\times 2+1)}
\end{aligned}
\tag{34}
$$

Where $M\times N$ and N_c denote the block size of a BTC image and the number of correct extracting results from the voting matrix, respectively. The $p_{ext}(PCRS)$ denotes the correct extracting probability of each result from voting matrix using the PCR size(PCRS).The $p_{ext}(PCRS)$ is constructed by four different combinations which can obtain one watermark result from the voting matrix. Among these, p_{atk} denotes the probability that a bit in bitmap is different from the original transmitted bit caused by attack. $p_{even}(PCRS)$ and $p_{odd}(PCRS)$ denote the number of error bit is even or odd, respectively, which are also caused by attack in PCR excepting the currently processing position. Two components, $p_c(N_i)\times(1-p_{atk})\times p_{even}(PCRS)$ even term and, are employed for constructing the equation $p_{ext}(PCRS)$.The first component $p_c(N_i)\times(1-p_{atk})\times p_{even}(PCRS)$ indicates that when the current position is not suffered by attack and other bits of bitmap in PCR have even number error, the calculated parity will be the same as the correct parity and, thus, the bitmap is considered as without being attack. The second component $\left[1-p_c(N_i)\right]\times(1-p_{atk})\times p_{odd}(PCRS)$ indicates that when the current bit in bitmap is wrong caused by attack, and odd number of others bits in PCR are incorrect, this condition will make the calculated parity in this PCR as the original parity without attack as well. According to this analysis, if a received EDBTC image is not suffered any attack, then $p_{atk}=0$ and $p_{even}(PCRS)=1$,in which $p_{ext}(PCRS)=p_c(N_i)$.

These attacks can be separated into two groups according to their attack types.

The first type demonstrates that each pixel in an EDBTC image has identical attack, in which p_{atk} can be modeled directly for evaluating the BER. This type of attacks includes lighting, darkening, pulse noising, Gaussian noising, and Gaussian smoothing. The second type of attacks associates that different area has different attack extent, in which the whole mathematical analysis of decoder has to be taken into account. The analyzed results demonstrate that the proposed method can widely be employed on different kinds of images and still achieves satisfactory robustness under different distortions.

The variable p_{atk} is affected by the strength of the added value (from lighting or darkening), and the decoded error occurs when the mean of a block cannot correctly separate the maximum and minimum values of a block. Since the dynamic range of a digital image is from 0 to 255, the error occurs when the added value causes the maximum and minimum values to be an identical value. This type of attack can be formulated as

$$p_{atk} = \sum_{D=0}^{255} p_r(d) \times p_s(d;s), where\ 0 \le s \le 255 \tag{35}$$

$$p_s(d;s) = \begin{cases} 0, if d \ne 0 and s \prec d \\ \dfrac{1-d+s}{256-d}, if d \ne 0 and d \le s \prec 255 \\ 1, otherwise \end{cases} \tag{36}$$

Where s denote the strength of the added value; $p_s(d;s)$ denotes the error probability when conditions d and s occur.

The well-known salt-and-pepper noise is considered in this type of attack. When a pixel in a bitmap is undergone this type of attack, there are 50% possibility that a pixel will change its polarity.

Thus, p_{atk} can be formulated as

$$p_{atk} = \frac{nr}{2}, where\ 0 \le nr \le 1 \tag{37}$$

Where nr denotes the ratio of noisy area.

In this type of attack, the decoded error occurs when the distortion caused by the Gaussian noise is higher than a half of the difference between maximum and minimum values in a block. Thus, this attack can be modeled as

$$p_{atk} = 2\sum_{d=0}^{255}\left[p_r(d) \times \sum_{i=\frac{d}{2}}^{\infty} N(i;\sigma) \right] \tag{38}$$

Where $p_r(d)$ is defined in equation (4),and $N(\bullet)$ denotes a 1-D Gaussian distribution with standard deviation σ.

The bit error caused by Gaussian smoothing attack occurs when the altered value

of an EDBTC image is higher than a half of the difference between maximum and minimum values.

In multiple watermarks embedding, noise increment can be further optimized for different numbers of watermark embedding. Average performance with different number of image size and noise increment under different number of watermarks and noise increments. The objective of this experiment is to determine a recommended N_i for the corresponding number of watermarks to achieve acceptable image quality (around HVS-PSNR=48db) and mean error.

Table 1: recommented noise strengths and the corresponding performance for each number of watermarks (pcr size=4).

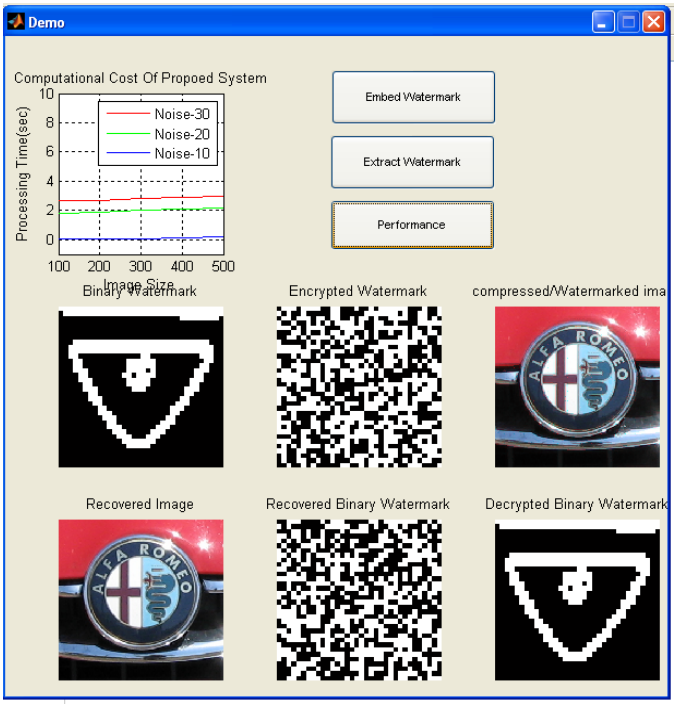
# of watermark	Noise strength	Corresponding HVS-PSNR	Corresponding Mean2error
1	10	45.095db	0.031
2	20	48.1078db	0.9974
3	30	39.95db	0.37

Recommended noise strengths and the corresponding performance for each number of watermarks (PCR size=4) which also includes the optimized $N_i(20)$ discussed. For the extremely high-capacity case of embedding watermarks, it still achieves mean2error=0.9974.

The parameters addressed in recommended noise strengths and the corresponding performance for each number of watermarks (PCR size=4) are employed for the proposed method, which is the reason why the HVS-PSNRs of the proposed MPGEDBTC are always around 48db.

Conclusion

In this paper a high-capacity watermarking technique for block truncation coding (BTC) images is proposed. This technique improves image quality of traditional BTC for configurations of high coding gain; where the energy-preserving property of EDF is exploited to effectively remove the false contour and blocking effect inherently exist in BTC images. The efficiency can also be improved by replacing the high and low means with the maximum and minimum values in a block. When watermarks are embedded, the proposed majority-parity-guided error-diffused BTC (MPG-EDBTC) can achieve good image quality and decoded rates under a huge embedded capacity. Two parameters, parity-check region (PCR) size and noise increment are employed for controlling the performance. The PCR size controls the quality of an embedded image, decoded watermark, and processing efficiency; the amount of noise increment provides the tradeoff between embedded image quality and decoded watermark quality.



References

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

[2] D. R. Halverson, N. C. Griswold, and G. L. Wise, "A generalized block truncation coding algorithm for image compression," IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP–32, no. 3, pp. 664–668, Jun. 1984.

[3] V. Udpikar and J.P. Raina, "Modified algorithm for block truncation coding of monochrome images," Electron. Lett, vol. 21, no. 20, pp.900–902, Sep. 1985.

[4] Q. Kanafani, A. Beghdadi, and C. Fookes, "Segmentation-based image compression using BTC-VQ technique," in Proc. IEEE Int. Conf. Information Science Signal Processing and their Applications, Paris, Jul. 1–4, 2003, vol. 1, pp. 113–116.

[5] S. Horbelt and J. Crowcroft, "A hybrid BTC/ADCT video codec simulation bench," presented at the Proc. 7th Int. Workshop on PacketVideo, Mar. 18–19, 1996.

[6] M. Kamel, C. T. Sun, and G. Lian, "Image compression by variable block truncation coding with optimal threshold," IEEE Trans. Signal Process, vol. 39, no. 1, pp. 208–212, Jan. 1991.

[7] L. G. Chen and Y. C. Liu, "A high quality MC-BTC codec for video signal processing," IEEE Trans. Circuits Syst. Video Technol., vol. 4, no. 1, pp. 92–98, Feb. 1994.

[8] J. F. Jarvis, C. N. Judice, and W. H. Ninke, "A survey of techniques for the display of continuous-tone pictures on bilevel displays," Comput. Graph. Image Process. vol. 5, pp. 13–40, 1976.
[9] V. Ostromoukhov, "A simple and efficient error-diffusion algorithm," in Proc. SIGGRAPH, 2001, pp. 567–572.
[10] P. Li and J. P. Allebach, "Block interlaced pinwheel error diffusion," J. Electron. Image. vol. 14, no. 2, Apr.–Jun. 2005.
[11] M. Mese and P. P. Vaidyanathan, "Optimized halftoning using dot diffusion and methods for inverse halftoning," IEEE Trans. Image Process., vol. 9, no. 4, pp. 691–709, Apr. 2000.
[12] A. U. Agar and J. P. Allebach, "Model-based color halftoning using direct binary search," IEEE Trans. Image Process., vol. 14, no. 12, pp. 1945–1959, Dec. 2005.
[13] J. M. Guo, "Improved block truncation coding using modified error diffusion," IET Electron. Lett. vol. 44, no. 7, pp. 462–464, Mar. 2008.
[14] J. M. Guo and M. F. Wu, "Improved block truncation coding based on the void-and-cluster dithering approach," IEEE Trans. Image Process., vol. 18, no. 1, pp. 211–213, Jan. 2009.
[15] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," IEEE Trans. Image Process., vol. 6, no. 12, pp. 1673–1687, Dec.
[16] S.A. Kassam, Quantisation based on mean-absolute erre criterion,'' IEEE Trans on communications, vol. com -26,pp.267-270, Feb 1978
[17] R.W. Floyd, L. Steinberg, An adaptive algorithm for spatial grey scale. Proceedings of the Society of Information Display 17, 75–77 (1976).
[18] P. Stucki, "Mecca-a multiple-error correcting computation algorithm for bilevel image hardcopy reproduction," Tech. Rep. RZ1060, IBM Research Laboratory, ...Zurich, Switzerland, Res. Rep. RZ
[19] Jing-MingGuo Yun-FuLiu Joint Compression/Watermarking Scheme Using Majority-Parity Guidance and Halftoning-Based Block Truncation Coding IEEE Trans. Image Process., vol. 19, issue 8,2010