Multidim Syst Sign Process (2018) 29:1801–1818
https://doi.org/10.1007/s11045-017-0530-8

An AMBTC compression based data hiding scheme using pixel value adjusting strategy

Aruna Malik[1] · **Geeta Sikka**[1] · **Harsh K. Verma**[1]

Received: 14 December 2015 / Revised: 23 November 2016 / Accepted: 11 November 2017 /
Published online: 18 November 2017

Abstract Image steganography is one of the most important research areas of information security where secret data is embedded in the images to conceal its existence while getting the minimum possible statistical detectability. To achieve a good tradeoff between the hiding capacity and image quality, more work needs to be further researched. In this paper, we propose high capacity data hiding scheme by employing the absolute moment block truncation coding (AMBTC) compression. It exploits the redundancy of blocks of the AMBTC-compressed image to embed the secret data. The pixel values of the AMBTC-compressed image are modified at most by one for hiding the secret data. Thus, it is able to maintain the stego-image quality after hiding the secret data. Experimental results validate the effectiveness of the proposed scheme and show that it outperforms various existing methods in terms of both hiding capacity and stego-image quality.

Keywords Data hiding · Pixel value adjusting · Secret data · Stego-image · Absolute moment block truncation coding

1 Introduction

Due to the emergence and proliferation of network technologies, Internet has become a very popular means of communication. However, data can be transmitted over the public networks such as the Internet, is not reliably safe because of counterfeiting, copyright violation, forgery and fraud (Artz 2001). Therefore, there is a need to develop some methods in order to protect digital data especially sensitive data.Several techniques have been designed to secure

✉ Aruna Malik
arunacsrke@gmail.com

Geeta Sikka
sikkag@gmail.com

Harsh K. Verma
vermah@nitj.ac.in

[1] Department of Computer Science and Engineering, National Institute of Technology, Jalandhar, Punjab 144011, India

digital data, the data hiding (also known as information hiding) is one of them. It embeds secret messages into a cover-media so that an unintended observer will not be aware of the existence of the hidden messages (Li et al. 2006). Cover media can be in the form of texts, images, digital sound files, and videos. There exist two main parameters that really affects data hiding scheme, these are the visual quality of stego images and embedding capacity (or payload) (Lin et al. 2008). A data hiding scheme having low image distortion is more secure than with high distortion because it does not raise any suspicions of adversaries. Therefore, image quality factor is the first important one. The next important factor is embedding capacity. Generally, an information hiding scheme with high payload is preferred so that more secret data can be communicated. However, embedding capacity is inversely proportional to visual quality i.e, if capacity is increased then visual quality is reduced and vice versa. The tradeoff between the two factors above is made by users depending on user's requirement and application fields. Thus, different techniques of data hiding are needed to span the range of possible applications. It has been found that data hiding techniques are divided into two groups namely steganography and watermarking. In case of first group i.e., steganography, it is a technique in which secret data is embedded into cover media. There is no relationship between message and the cover image in steganography. The cover image is treated as a decoy to mask the presence of communication. The basic purpose of steganography is the covert communication in which the very existence of the secret data is not possible to be detected by the attackers. There are three fundamental requirements namely perceptual transparency, robustness, and hiding capacity for designing steganographic systems (Lu et al. 2009). The security or perceptual transparency deals with the ability of an eavesdropper to figure, or suspect the hidden information easily. High security can be achieved by minimizing the distortion between the cover text and the stego text. The robustness refers to the resist possibility of modifying or destroying the unseen data. The capacity refers to the maximum no of bits that can be hidden into the cover media. It is given by the ratio of bits of secret message to bits of stego cover. As far as watermarking is concerned, it alters a cover object, either imperceptibility or perceptibility to embed a message about the cover objects. Watermarking finds application in copyright protection, transaction tracking, and broadcast monitoring. In watermarking, some relevant information is provided by the embedded message (like image caption, authentication code, and author's signature).

Data hiding techniques in image steganography can be broadly classified into three domains: Spatial domain, transform domain and compressed domain. In the spatial domain, secret data are embedded into pixels by directly modifying the image pixels (Lin et al. 2009). In the transform domain scheme, the host image must first be transformed into frequency coefficients by utilizing a frequency oriented mechanism such as discrete cosine transformation (DCT) (Yang et al. 2011) or discrete wavelet transformation (DWT) (Xuan et al. 2006). Subsequently, secret data are combined with the relative coefficients. Lastly, the modified coefficients are used to reconstruct the frequency-form image as a stego-image. Generally, the transform domain based schemes have the capability to resist against attack when compared with the spatial domain, still these suffers from computational complexity problem and uncompressed stego-image. To overcome this problem, compressed domain schemes are evolved. The compressed domain-based methods hide the secret data in the images compressed by different methods like JPEG (Wang et al. 2013), vector quantization (Lu et al. 2009), block truncation coding (Hu and Chang 1999) etc. In compressed domain, embedding and extraction process does not involve transforming to the frequency domain due to this, computation cost is relatively low as compared to transform domain. Moreover, compressed stego-image can enhance the efficiency of data transmission and reduce suspicion from the attackers. The AMBTC method is a variant of BTC, proposed by Lema and Mitchell (1984)

to further improve the compression performance of traditional BTC. AMBTC is computationally simpler than BTC method and also maintains the first absolute moment along with the mean. Here, each image block is encoded by using one-bit plane and two quantization levels. To get advantages of AMBTC compression, in this paper, we propose an absolute moment block truncation coding (AMBTC) compression based data hiding scheme using a pixel value adjusting strategy. This scheme considers the compressed image in the non-overlapping 4×4 pixel blocks for hiding the secret data. It embeds the secret data (converted into a base 3 representation) into the AMBTC compressed image by modifying the pixel value at most by one and some of the pixels might remain untouched. Thus, it is able to maintain the good quality stego-image even after hiding a large amount of the secret data. Experimental results show that the proposed scheme is superior to other existing schemes in terms of both hiding capacity and stego-image quality.

The rest of the paper is organized as follows. In Sect. 2, we briefly introduce the Absolute Moment Block Truncation Coding (AMBTC) compression technique and existing BTC and AMBTC based data hiding methods. The embedding and extraction process of the proposed scheme are introduced in Sect. 3. The experimental results and analysis are given in Sect. 4. Finally, in Sect. 5, the paper is concluded.

2 Related work

In this section, we will review the standard concepts of Absolute Moment Block Truncation Coding (AMBTC) and some recently developed and important BTC and AMBTC based data hiding methods which are presented in Sects. 2.1 and 2.2, respectively.

2.1 Absolute moment block truncation coding (AMBTC)

BTC is a simple and efficient block based spatial domain lossy image compression technique for grayscale images. This technique is mainly used for the real-time application and arm-based application due to its low complexity, low computational cost and easy to implement characteristics. The main idea of this technique is to quantize the pixels in each block into two levels while maintaining certain statistical moments of small blocks of the gray scale image. Later, the AMBTC method as a special kind of BTC is proposed by Lema and Mitchell (1984) to further improve the compression performance of traditional BTC by preserving the first absolute moment along with the mean instead of using the standard deviation. AMBTC is computationally simpler than the BTC method. The main difference between AMBTC and BTC is the calculation of two quantization level during the encoding procedure. However, the image decoding procedure of AMBTC technique is same as that of the traditional BTC technique. The details of the image encoding procedure and decoding procedure of the AMBTC method are described as follows.

During the encoding procedure, the input image is first decomposed into several non-overlapping $M \times M$ sized blocks. It is noted that M is set to be 4 during the conventional AMBTC encoding phase. For each image block, its mean pixel value AVG and its standard deviation var are then computed by using Eqs. (1) and (2), respectively.

$$AVG = \frac{\sum_{i=1}^{M \times M} \mu_i}{M \times M} \tag{1}$$

$$var = \frac{\sum_{i=1}^{M \times M} |\mu_i - AVG|}{M \times M} \tag{2}$$

Where μ_i represents the ith pixel value in each block of the image and M denotes the size of the image block. For each block, take the ith block, for example; each pixel value μ_i in the ith block is compared with its AVG in such a way that a bitmap that is composed of two groups is generated according to the following two rules:

(i) If the pixel value μ_i is less than its corresponding AVG, then the corresponding bit in the bitmap of the ith block is determined as group-0 and is denoted as'0' in the bitmap.
(ii) Otherwise, the bit belongs to group-1 and is denoted as'1' in the bitmap.

During the decoding phase, the low mean values L_m for group-0 and the high mean values H_m for group-1 can be derived from Eqs. (3) and (4), respectively.

$$L_m = AVG - \frac{M \times M \times var}{2\,(M \times M - \beta)} \tag{3}$$

$$H_m = AVG + \frac{M \times M \times var}{2\beta} \tag{4}$$

Where β is the number of pixels that are greater than or equal to AVG derived using Eq. (1) in the bitmap of the ith block, M denotes the size of the encoding block, and var denotes the standard deviation for the ith block. Once two mean values L_m and H_m are obtained, the corresponding reconstructed image block can be easily generated from the received bitmap by replacing each'0' of bitmap with a low mean value L_m and each'1'of bitmap with a high mean value H_m.

To compare the image quality of the reconstructed images after BTC and AMBTC compression, an experiment was implemented by us. The experiment was performed with twelve commonly used grayscale images, namely Airplane, Couple, Elaine, Boat, House, Lena, Man, Mandrill, Peppers, Sailboat, Splash, and Tiffany of size 512×512. The block size used in BTC and AMBTC compression is set to 4×4 pixels. After compression, the host image was reconstructed to compare the image quality. For example, a block of the image is shown in Table 1(a). The calculated mean value of the image block is $AVG = 139$ and $var = 28$ by using Eqs. (1) and (2), respectively. The mean value $AVG = 139$ is then taken as a threshold to generate the bitmap as shown in Table 1(b). Pixels with values greater than or equal to AVG are set to 1 in the bit plane; otherwise, the binary value 0 is set. Two quantization levels $L_m = 111$ and $H_m = 167$ are obtained by Eqs. (3) and (4), respectively. By using these two quantization levels L_m, H_m and a bitmap, the image block can be reconstructed as shown in Table 1(c). If the corresponding indicator is 0, then the receiver decodes a pixel with L_m Otherwise, the pixel is reconstructed by H_m.

Assume that each of two quantization levels L_m and H_m requires 8 bits, and the bit pattern of the block requires $M \times M$ bits. For a 4×4 block, the total number of bits required is $(8 + 8 + 4 \times 4) = 32$ bits. Thus, the compression rate of the AMBTC method is $(8 + 8 + 4 \times 4)/16 = 2$ bpp, which is the same as that of the traditional BTC method. As stated in Lema and Mitchell (1984), the AMBTC method preserves the first absolute moment along with the mean instead of using the standard deviation in the BTC method, so that the AMBTC method does not require complex computation as compared to the BTC method.

Table 2 shows the comparison of PSNR values derived by BTC and AMBTC for reconstructed images. Note that all of the reconstructed images derived from AMBTC keep the same image quality as that of BTC. In addition, for each encoding block, the two quantization levels, the mean pixel value AVG and its standard deviation var, are generated during the encoding phase rather than the two mean values that are required in BTC. Hence, hiding data using AMBTC degrades the hiding distortion while preserving the image quality as in BTC because the standard deviation var is slightly smaller than the mean values generated by

Table 1 Shows the AMBTC encoding and decoding

(a) Original image block			
130	92	161	165
133	97	90	170
147	175	140	183
124	194	106	116
(b) Bit pattern of the image block			
0	0	1	1
0	0	0	1
1	1	1	1
0	1	0	0
(c) Reconstructed image block			
111	111	167	167
111	111	111	167
167	167	167	167
111	167	111	111

BTC. In next sub-section, we discuss some related and important BTC and AMBTC based data hiding methods.

2.2 Related work of BTC and AMBTC based data hiding methods

Chuang and Chang (2006) discussed a BTC based data embedding scheme. It first compresses a gray scale cover image by dividing it into blocks using BTC technique to get two quantization values and one bitmap for each block. Then, a predefined threshold is used to classify each BTC-encoded block as smooth or complex. Subsequently, secret data is embedded into the bitmap of the selected BTC-encoded blocks. This scheme is limited to gray scale images only and the stego image quality is significantly degrades with the increase in threshold value thus increasing the suspicion from the attackers. Later, Chang et al. (2008) proposed a data hiding method for color images compressed by BTC. In this method, BTC technique is applied to each block of a cover image which results into three pairs of quantization levels and three bitmaps. In order to improve the compression rate further, a genetic algorithm is used to find an approximate optimal bitmap out of three bitmap. Next, Chang et al. (2009) suggested a data embedding method based on the genetic algorithm and absolute moment block truncation coding for color images. In this scheme, the bitmap generation procedure of GA-AMBTC has been modified. This method suffers from low embedding capacity and high distortion after pixel modifications. Chen et al. (2010) introduced a data hiding method for BTC compressed images. Here, the high mean and the low mean values are used to embed the secret data in the uniform blocks. The difference between the high mean and the low mean values of each block is used to determine whether only 1 bit of the secret data is to be hidden in the block or to toggle bits in the bitmap to hide more bits. However, it mimimize the distortion problem but also suffers from the low hiding capacity. Next, Li et al. (2011) discussed a data hiding method for BTC-compressed images based on bit plane flipping and histogram shifting of mean tables. In this technique, two strategies are used, one is based on the bitplane flipping technique that hides the secret bits by swapping the high mean and low mean, other is a histogram shifting of the resulting mean tables after swapping. However, it achieves very low image distortion after data embedding and enhances the security of embedded conven-

Table 2 Comparison of the PSNR values computed by BTC and AMBTC compression schemes for reconstructed images

Scheme	Airplane	Couple	Elaine	Boat	House	Lena	Man	Mandrill	Peppers	Sailboat	Splash	Tiffany
BTC	31.96	32.91	33.87	31.14	30.88	33.19	36.81	26.97	33.39	29.85	36.72	35.72
AMBTC	31.96	32.91	33.87	31.14	30.88	33.19	36.81	26.97	33.39	29.85	36.72	35.72

tional data, still this technique not able to provide sufficient data hiding capacity. Raja (2012) proposed a data hiding scheme that makes use of AMBTC and interpolative technique. In this scheme, the secret image is stored in the bit plane of the AMBTC compressed image and is recovered during decompression process. Later, Zhang et al. (2013) introduced a data hiding scheme for BTC-compressed images by further losslessly encoding the BTC-compressed data according to the secret bits. Initially, the BTC technique is applied to the original image to get the BTC-compressed data that consists of a high mean table, a low mean table, and a bitplane sequence which is used to hide the secret data. Li et al. (2015) discuss a bi-stretch data hiding algorithm for absolute moment block truncation coding compressed images. In this scheme, the AMBTC compressed image is divided into non overlapped blocks first, after that, four feasible cases are employed to embed secret data, which takes advantage of the characteristics of the coefficients of the AMBTC compressed image and leads to very small distortion of the AMBTC compressed image. Huang et al. (2016) discuss a hybrid secret hiding schemes based on absolute moment block truncation coding for embedding more secret data into the complex and smooth blocks. It utilizes two different hiding methods for embedding the secret data into the smooth and complex blocks. In case of smooth blocks, the small variation of the block is adopted to define the embedding rule to minimize the distortion after data embedding. As for the complex blocks, the large variation of the block is used to embed more secrets while maintaining good visual quality. Malik et al. (2016) modified the AMBTC technique by providing two bit plane with four quantization levels to improve image quality and payload. Although the aforementioned methods achieve good visual quality and payload, they cause permanent distortion to the original AMBTC codes. Sun et al. (2013) proposed high-performance data hiding technique for block truncation coding compressed image. It improves the Li et al. (2011) method in terms of higher embedding capacity. In this method, a joint neighbor coding technique is used for encoding the BTC-compressed data according to the secret bits. The secret data is embedded in both the high mean and low mean table. It embeds two bits of the secret data in each mean table. This method can obtain a capacity that is four times as large as the number of blocks in the cover image. However, it requires extra data to be included in the stego-code stream, and also the reconstructed images cannot be derived by a conventional BTC-decoding algorithm directory. Later, Lin et al. (2013) discussed a data hiding method based on the AMBTC compression technique. This method explores the redundancy in a block of AMBTC-compressed images to determine if a block is embeddable or non-embeddable. In this method, four disjoint sets are created for embeddable blocks to embed data using different combinations of the mean value and the standard deviation. It does not require extra data to be included in the stego-code stream like Sun et al. (2013) method. This method suffers from low hiding capacity. Pan et al. (2014) introduced a data hiding scheme for AMBTC compressed images using a reference matrix. The quantization levels of each AMBTC-compressed image block are used to embed the secret data by using a reference matrix. For each image block, original quantization levels are changed into another stego message which is combined with a bitmap. Later, Ou and Sun (2014) discussed an AMBTC based data hiding scheme having minimum distortion. In this technique, a threshold is defined to divide the blocks of the AMBTC-compressed codes as smooth and complex, in which secret data are embedded. In case of smooth blocks, the bit planes are directly used to embed the data by replacing the bits with the secret data bits. Next, the two quantization levels are recalculated to reduce the caused distortion. For complex blocks, a proportion of secret bits are hidden by exchanging the order of two quantization levels with together toggling the bit plane. As discussed above, the existing techniques either suffer from low embedding capacity or low quality of stego-image. Our method tries to keep a balance between hiding capacity and the quality of stego image, i.e., it tries to improve the hiding capacity while maintaining a good

quality of the stego-image. It exploits the redundancy of blocks of the AMBTC-compressed image to embed the secret data. The pixel values of the AMBTC-compressed image are modified at most by one to embed the secret data, which consequently gives high-quality stego-image. Our scheme has better performance than the recently developed methods like Ou and Sun (2014), Lin et al. (2013), Li et al. (2011) and Chang et al. (2008).

3 Proposed scheme

The proposed scheme is a compressed domain based scheme which hides the secret data into the compressed codes. The scheme compresses the cover image using AMBTC compression and then embeds the secret data into the AMBTC compressed image using embedding rules. After AMBTC compression, the cover image is divided into M × M sized blocks where each block contains only two different values i.e. Low mean (L_m) and High mean (H_m). The secret data bit stream is converted into a base 3 representation. For embedding the secret data digits into the compressed image, we, first of all, identify the embeddable and non-embeddable blocks of the compressed image. If the difference between the H_m and L_m is greater than ± 1 then this block is determined to be an embeddable one otherwise it is non-embeddable block. Now, we identify the non-embeddable pixels in each embeddable block. The first H_m and the first L_m pixel of the block are considered as non-embeddable pixels.

3.1 Embedding rule

Embed Secret data into the embeddable pixels p_{E_j} of each block b_i

```
For each block b_i in image I
    For each pixel p_{E_j} in block b_i
        If secret digit s_b in S equals 0, then
          Do nothing
        Else if secret digit s_b in S equals 1, then
            p_{E_j} = p_{E_j} + 1;
        Else                          // secret digit s_b in S equals 2
            p_{E_j} = p_{E_j} − 1;
        End
    End
End
```

3.2 Illustrations of embedding process

An example to explain the proposed embedding process for stego-image construction is shown in Fig. 1. The embedding process considers 4 × 4 pixel block size and scanning the image in a raster manner from coordinates (0, 0) to (3,3). We have P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}, P_{13}, P_{14}, P_{15}, and P_{16} pixels in 4 × 4 block which are 130, 92, 161, 165, 133, 97, 90, 170, 147, 175, 140, 183, 124, 194, 106, and 116, respectively. Firstly, this block is compressed using AMBTC compression and then Low mean (L_m) and High mean (H_m) values are calculated. The bit plane of the original image block which contains only two values i.e., 0 and 1 is generated and then the compressed block can be reconstructed using Low mean (L_m) and High mean (H_m) values. The pixel values of the compressed block

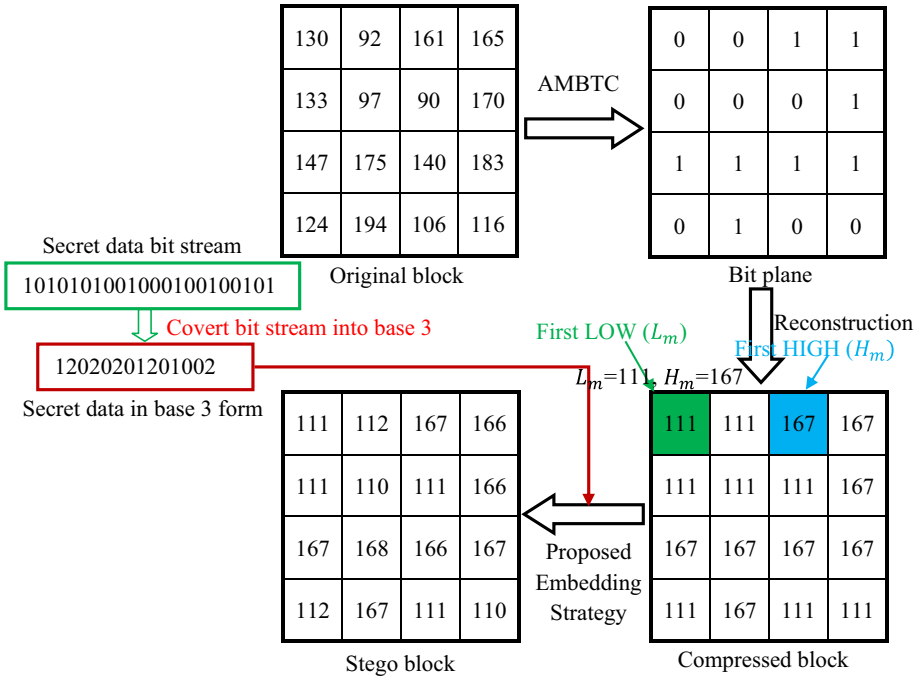


Fig. 1 An example to explain the proposed embedding process

are P_1 (= 111), P_2 (= 111), P_3 (= 167), P_4 (= 167), P_5 (= 111), P_6 (= 111), P_7 (= 111), P_8 (= 167), P_9 (= 167), P_{10} (= 167), P_{11} (= 167), P_{12} (= 167), P_{13} (= 111), P_{14} (= 167), P_{15} (= 111), and P_{16} (= 111). We identify the non-embeddable pixels from the compressed block which are the first L_m and the first H_m valued pixels. In our example, the first Low mean pixel (L_m) is made green colored i.e., P_1 (= 111) and the first High mean pixel (H_m) is made blue colored i.e., P_3 (= 167).

The rest of the pixels are considered as embeddable pixels which can be used to hide the secret data. For embedding the secret data digits, if the secret data digits are '1' then the corresponding pixel value of the block is increased by one and if it is '2' then pixel value is decreased by one. In case the secret data digit is '0' then the corresponding pixel value remains unchanged. Consider, a secret data bitstream S = $(1010101001000100100101)_2$ which is converted into a base 3 representation as $(12020201201002)_3$. The first digit of the secret data is '1' which will be hidden into the first embeddable pixel of the block which is P_2 by increasing its value by 1. Thus, after embedding, the pixel value is 112. The second digit of the secret data is also '2' which will be embedded into the second embeddable pixel which is P_4 by decreasing its value by 1. Now, the pixel P_4 value is 166. The third digit of the secret data is '0' which will be embedded into the third embeddable pixel P_5 which does not cause any change in the pixel value. The fourth digit of the secret data is '2' which will be embedded into the fourth embeddable pixel P_6 of the block by decreasing the pixel value by 1. Now, the pixel value becomes 110. Thus proceeding, the entire secret data is embedded into the block and the obtained pixel values are as follows P_1 = 111, P_2 = 112, P_3 = 167, P_4 = 166, P_5 = 111, P_6 = 110, P_7 = 111, P_8 = 166, P_9 = 167, P_{10} = 168, P_{11} = 166, P_{12} = 167, P_{13} = 112, P_{14} = 167, P_{15} = 111, and P_{16} = 110.

3.3 Extracting phase

In the extracting phase, the secret data are extracted from the Stego AMBTC compressed image and the image can be recovered to the original AMBTC compressed image. For extracting the secret data, the extracting algorithm is given below:
Input: A Stego AMBTC-compressed image.
Output: A recovered AMBTC-compressed image and extracted secret data.
Step 1. Read the $M \times M$-sized block of the stego-image X.
Step 2. Count the number of different values in the current block.
Step 3. Check the number of different pixel values in the current block. If the number of different pixel values is 2, then calculate the difference of the values if it is <2 then go to step 4 otherwise go to Step 6.
Step 4. Because the number of different pixel values is 2 and the difference of the values is <2, means the block is a non-embeddable block which does not contain any secret data digit. The current block is the original block of the AMBTC compressed image.
Step 5. Go to *Step1* until all of the blocks of the stego-image have been processed.
Step 6. Because the number of different pixel values is >2, the block is embeddable block and contains the secret data.
Step7. Identify the non-embeddable pixels in each block i.e. two pixels. The very first pixel of the block is always considered as the non-embeddable pixel. The second non-embeddable pixel is the first pixel whose difference is greater than one of the first non-embeddable pixel of the block.
Step 8. Least valued non-embeddable pixel is denoted as L_m and another non-embeddable pixel is by H_m.
Step 9. Extract the secret data from the embeddable pixels p'_E of each block b'_i

```
For each block b'_i in image
  For each pixel p'_Ej in block b'_i
    If p'_Ej equals H_m OR L_m, then
      Secret data digit is 0
          Do nothing
    Else p'_Ej ± 1 equals H_m OR L_m, then
      If ((H_m OR L_m) +1= p'_Ej)then
        p'_Ej = p'_Ej − 1;
          Secret data digit is 1
      Else
        p'_Ej = p'_Ej + 1;
          Secret data digit is 2
      End
    End
  End
End
```

Step10. Go to *Step1* until all of the blocks of the stego-image have been processed.
Step11. Convert the obtained secret data digits into binary form to obtain the original secret data bit stream.

Thus, we get the secret data S and the recovered AMBTC compressed image I.

3.4 Illustrations of extracting process

Similarly, an example is taken for illustrating the process of data extraction. Recall to the previous example of Fig. 1, for extraction of the secret data digits from the stego block, we first identify the non-embeddable pixels of the block which are $P_1 = 111$ and $P_3 = 167$. The remaining pixels of the block contain the secret data. The minimum of $\{P_1, P_3\}$ is considered as Low mean (L_m) value and maximum of $\{P_1, P_3\}$ is considered as High mean (H_m) value. If the embeddable pixel value is equal to the $\{Low\ mean\ (L_m), High\ mean\ (H_m)\}$ then the embedded secret data digit is '0' and there had been no change in the pixel value; if the pixel value is greater than by at most 1 than $\{Lowmean\ (L_m), High\ mean\ (H_m)\}$ it means embedded digit of the secret data is '1' and recover the pixel value by decreasing by 1 otherwise the embedded digit of the secret data is '2'and recover the pixel value by increasing it by 1. The first embeddable pixel is $P_2 = 112$ which is equal to $(L_m + 1)$ means the embedded digit is '1'. For recovering the pixel value, we decrease its value by 1. The second embeddable pixel is $P_4 = 166$ which is equal to $(H_m - 1)$ means the embedded digit is '2'. For recovering, the pixel value, we also increase it by 1. The third embeddable pixel is $P_5 = 111$ whose value is equal to Low means the embedded digit is '0' and it also implies that the pixel had not been modified in embedding phase, so no need to recover it as it is in its original form. The fourth embeddable pixel $P_6 = 110$ whose value is equal to $(L_m - 1)$ means embedded digit is '2'. For recovering the pixel value, we increase it by 1. Proceeding, in the same way, we will obtain the original compressed image and the secret data digits $(12020201201002)_3$ which are converted into binary form to obtain original bit stream $S = (10101010010001001001 01)_2$. The obtained pixel values are P_1 (= 111), P_2 (= 111), P_3 (= 167), P_4 (= 167), P_5 (= 111), P_6 (= 111), P_7 (= 111), P_8 (= 167), P_9 (= 167), P_{10} (= 167), P_{11} (= 167), P_{12} (= 167), P_{13} (= 111), P_{14} (= 167), P_{15} (= 111), and P_{16} (= 111). Then, this block uncompressed by the AMBTC technique and we have P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}, P_{11}, P_{12}, P_{13}, P_{14}, P_{15}, and P_{16} original pixels of the cover image in 4×4 block which are 130, 92, 161, 165, 133, 97, 90, 170, 147, 175, 140, 183, 124, 194, 106, and 116, respectively.

4 Experimental results and discussion

In this section, we will investigate the performance of the proposed work by conducting some experiments. The main objective of this work is to demonstrate that the proposed scheme achieves high capacity and good image quality at the same time. Twelve grayscale cover images of size 512×512 pixels as shown in Fig. 2a–l are used in our experiment. The block size used in the AMBTC compression is 4×4 pixels. The proposed scheme is implemented in MATLAB running on Intel (R) Core (TM) i5 processor 3.20-GHz with the 4-GB RAM hardware platform. The secret data used in the experiments is a stream of random bits, generated by a pseudo-random number generator (PRNG). For comparison, two parameters, namely hiding capacity and visual quality of the stego-image are used to evaluate the power of the previously existing schemes and the proposed scheme.

The hiding capacity is used to measure the amount of secret data bits that can be hidden in the cover images. The bit per pixel is a measure for hiding capacity. Here it is symbolized by bpp as given by Eq. (5). It is calculated by dividing the number of secret data bits embedded in an image with the number of pixels used in the embedding.

$$\text{bpp} = \frac{\text{number of secret data bits}}{\text{number of pixels}} \tag{5}$$

Fig. 2 Cover images, each of size 512×512

With regard to the second parameter, the visual quality of the image is measured by the peak-signal-to-noise ratio (PSNR) defined as:

$$\mathrm{PSNR} = 10 \log_{10} \left[\frac{255 * 255}{MSE} \right] \tag{6}$$

Where, MSE is the Mean Square Error and is defined as

$$MSE = \frac{1}{M \times M} \sum_{i=1}^{M} \sum_{j=1}^{M} \left(I_{ij} - X_{ij} \right)^2 \tag{7}$$

MSE represents the difference between the original cover image I and the stego-image X sized $M \times M$, and I_{ij} and X_{ij} are the pixels located at the ith row and jth column of image I and X, respectively. In general, the larger the PSNR value is, the better the image quality of the image would be. Figure 3a–l shows the visual quality of the AMBTC-compressed images with their corresponding PSNR values. Figure 4a–l shows the stego-image along with computed the data hiding capacity and PSNR between AMBTC compressed image and stego-image. It is clearly evident from Fig. 4 that it is too hard to distinguish the stego-images from the cover image. Since, embedding the secret data in the AMBTC-compressed image slightly decreases the PSNR value. It is also observed that after extracting the secret data

Fig. 3 AMBTC compressed images

from stego images the PSNR values of all stego-images are equal to the original AMBTC-compressed images, for this reason, our proposed scheme works in a lossless manner.

4.1 Performance comparison of proposed scheme with another scheme

We now discuss the experimental results of our proposed scheme by comparing with some AMBTC and BTC based data hiding schemes such as Chang et al. (2008) method, Li et al. (2011) method, Lin et al. (2013) method, and Ou and Sun (2014) method for twelve AMBTC compressed images. Table 3 demonstrates the comparison results in terms of hiding capacity measured in bpp, and image quality (measured in PSNR) for proposed scheme and related schemes for all the images. However, Chang et al. (2008) method can't reconstruct the original image by the BTC-decoding compression as the additional information is incorporated in the stego-image while hiding the secret data bits. It is not good for the security during the image transmission because the unusual stego code that are generated by Chang et al. (2008) scheme can attract the suspicion of malicious users and incur attacks on this basis. With respect to hiding capacity, it is found from Table 3 that proposed scheme has higher hiding capacity

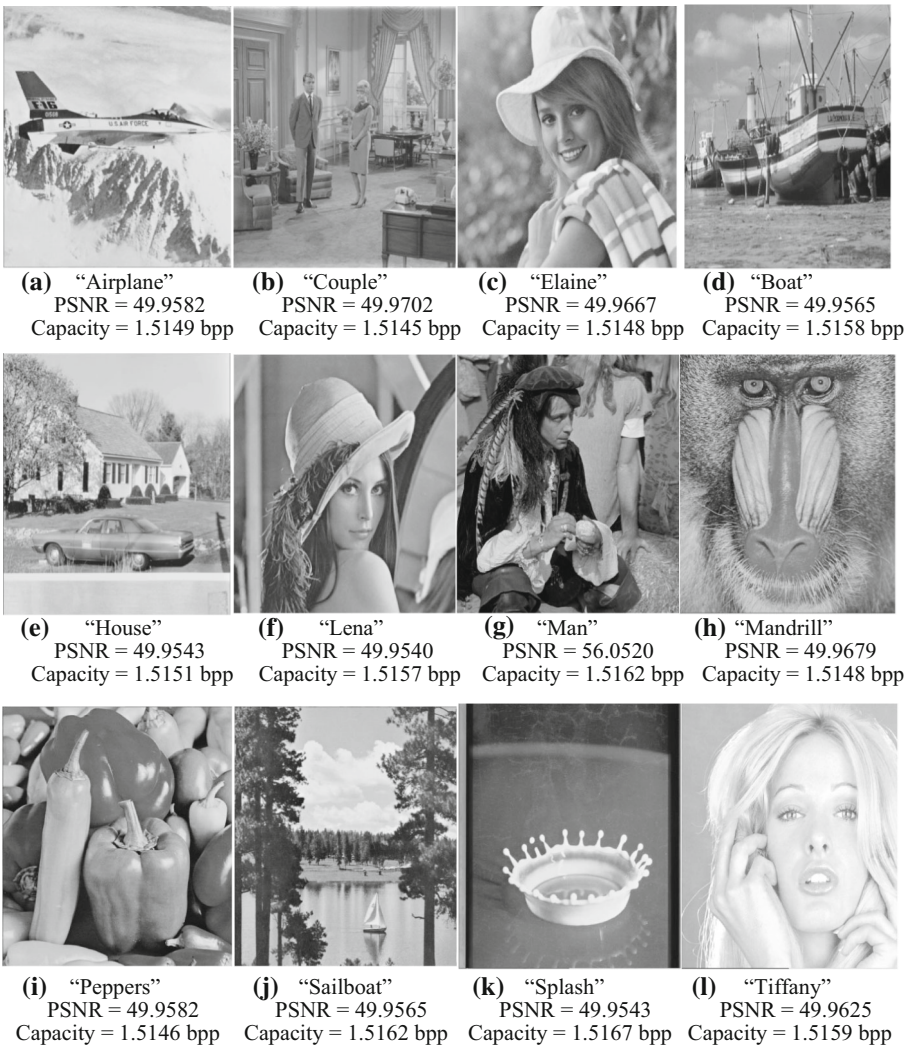


Fig. 4 Stego-images of the proposed scheme

than all the other schemes. The percentage increments for the hiding capacity of our scheme is in the range of 51.48–51.78%, respectively, as compared to Lin et al. (2013) method which has the highest hiding capacity in the existing data hiding schemes.Our scheme embeds 24 bits on average in each 4×4 pixel block. However, Lin et al. (2013) method embeds only 16 bits in each 4×4 pixel block. The capacity of Ou and Sun (2014) method, Li et al. (2011) and Chang et al. (2008) methods is approximately 1, 1 and 2 bit(s) per pixel, respectively, in case of 4×4 pixel block. Our proposed scheme is able to achieve approximately 1.5 bpp data hiding capacity and also allows the recoverability of the original image at the same time. Thus, our scheme outperforms the existing AMBTC and BTC based data hiding schemes in terms of embedding capacity.

The other very important aspect of analyzing the performance is PSNR. In Table 3, we have also shown the performance of the proposed scheme and other related schemes in

Table 3 Comparison of hiding capacity and PSNR for different images between the proposed scheme and other AMBTC and BTC based schemes

Method	Factor	Airplane	Couple	Elaine	Boat	House	Lena	Man	Mandrill	Peppers	Sailboat	Splash	Tiffany
Proposed method (B)	Hiding Capacity	397147	397018	397098	397380	397187	397348	397463	397105	397057	397466	397600	397387
	PSNR	31.9018	32.8311	33.774	31.0926	30.8515	33.102	36.7629	26.9474	33.304	29.8081	36.5267	35.5633
Lin et al. (2013) method (A)	Hiding Capacity	261984	262048	261984	262096	262160	262112	261984	262144	261984	262064	261952	261968
	PSNR	31.64	31.8469	32.2367	30.32	29.9547	33.05	34.9874	26.0047	32.2021	28.8049	34.8729	34.5631
Ou and Sun (2014) method	Hiding Capacity	223039	204552	226549	217264	214609	234004	212334	141919	238969	219169	225454	234659
	PSNR	30.71	31.34	31.84	29.54	29.21	30.87	33.98	26.02	31.59	28.61	34.54	33.98
Li et al. (2011) method	Hiding Capacity	17659	17054	16580	17082	16820	16789	17090	16880	17264	16990	17761	16755
	PSNR	30.18	31.54	31.83	31.02	30.34	31.05	33.56	27	32.35	28.43	34.23	34.12
Chang et al. (2008) method	Hiding Capacity	30518	30254	29558	28766	30543	31011	30638	27870	31537	30547	30962	30151
	PSNR	Can not be reconstructed by the stego code stream											
% increment (B − A)/A	Hiding Capacity	51.6199	51.5058	51.5734	51.6162	51.5056	51.6132	51.7127	51.4835	51.5577	51.6675	51.7835	51.693
	PSNR	5.3497	3.0904	4.7688	3.0976	2.9939	2.2803	5.0747	3.6251	3.4218	3.4827	4.7424	2.8938

terms of PSNR. It is clearly evident from Table 3 that proposed scheme has an increment in PSNR percentage in the range of 2.28–5.35% in comparison to Lin et al. (2013) method. The proposed scheme achieves very good quality because the pixel values are modified at most by 1 (i.e., ± 1) and even some of the pixels might remain unchanged depending on the secret data bits. Thus, we find that the proposed scheme is superior as compared to the other schemes in terms of the hiding capacity and image quality for all the test images.

5 Conclusions

In this paper, we have proposed an AMBTC compression based data hiding scheme using pixel value adjusting strategy. Our scheme achieves very good quality as it modifies the pixel value of the compressed image maximum by 1 either by increasing or decreasing the pixel value and even some pixel values remain untouched depending on the secret data. Additionally, it is also able to manage approximately 1.5 bpp data hiding capacity as it hides 1 digit of base 3 representation of the secret data stream into every embeddable pixel of the compressed image while the contemporary schemes provide only 1 bpp hiding capacity. Thus, it has the higher hiding capacity and better image quality. Our scheme can also be applied to color images by operating on individual color components and hence, it is applicable to both the grayscale and color images.

References

Artz, D. (2001). Digital steganography: Hiding data within data. *IEEE Internet Computing*, *5*(3), 75–80.

Chang, C. C., Chen, Y. H., & Lin, C. C. (2009). A data embedding scheme for color images based on genetic algorithm and absolute moment block truncation coding. *Soft Computing*, *13*, 321–331.

Chang, C. C., Lin, C. Y., & Fan, Y. H. (2008). Lossless data hiding for color images based on block truncation coding. *Pattern Recognitation*, *41*(7), 2347–2357.

Chen, J., Hong, W., Chen, T. S., & Shiu, C. W. (2010). Steganography for BTC compressed images using no distortion technique. *The Imaging Science Journal*, *58*(4), 177–185.

Chuang, J., & Chang, C. (2006). Using a simple and fast image compression algorithm to hide secret information. *International Journal of Computer and Application*, *28*(4), 329–333.

Huang, Y. H., Chang, C. C., & Chen, Y. H. (2016). Hybrid secret hiding schemes based on absolute moment block truncation coding. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-015-3208-y.

Hu, Y. C., & Chang, C. C. (1999). Quadtree-segmented image coding schemes using vector quantization and block truncation coding. *Optical Engineering*, *39*(2), 464–471.

Lema, M., & Mitchell, O. (1984). Absolute moment block truncation coding and its application to color images. *IEEE Transaction on Communcations*, *32*, 1148–1157.

Li, F., Bharanitharan, K., Chang, C. C., & Mao, Q. (2015). Bi-stretch reversible data hiding algorithm for absolute moment block truncation coding compressed images. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-015-2924-7.

Li, S., Leung, K., Cheng, L. M., & Chan, C. K. (2006). A novel image-hiding scheme based on block difference. *Pattern Recognition*, *39*(6), 1168–1176.

Li, C. H., Lu, Z. M., & Su, Y. X. (2011). Reversible data hiding for BTC-compressed images based on bitplane flipping and histogram shifting of mean tables. *Information Technology Journal*, *10*(7), 1421–1426.

Lin, C. C., Liu, X. L., Tai, W. L., & Yuan, S. M. (2013). A novel reversible data hiding scheme based on AMBTC compression technique. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-013-1801-5.

Lin, I. C., Lin, Y. B., & Wang, C. M. (2009). Hiding data in spatial domain images with distortion tolerance. *Computer Standards & Interfaces*, *31*(2), 458–464.

Lin, C. C., Tai, W. L., & Chang, C. C. (2008). Multilevel reversible data hiding based on histogram modification of difference images. *Pattern Recognitation*, *41*(12), 3582–3591.

Lu, Z. M., Wang, J. X., & Liu, B. B. (2009). An improved lossless data hiding scheme based on image VQ-index residual value coding. *Journal of Systems and Software*, *82*(6), 1016–1024.

Malik, A., Sikka, G., & Verma, H. K. (2016). A high payload data hiding scheme based on modified AMBTC technique. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-016-3815-2.

Ou, D., & Sun, W. (2014). High payload image steganography with minimum distortion based on absolute moment block truncation coding. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-014-2059-2.

Pan, J., Li, W., & Lin, C. (2014). Novel reversible data hiding scheme for AMBTC-compressed images by reference matrix. *Communications in Computer and Information Science*, *473*, 427–436.

Raj, I. (2012). Image data hiding in images based on interpolative absolute moment block truncation coding. *Communications in Computer and Information Science*, *283*, 456–463.

Sun, W., Lu, Z. M., & Wen, Y. C. (2013). High performance reversible data hiding for block truncation coding compressed images. *Signal, Image and Video Processing*, *7*(2), 297–306.

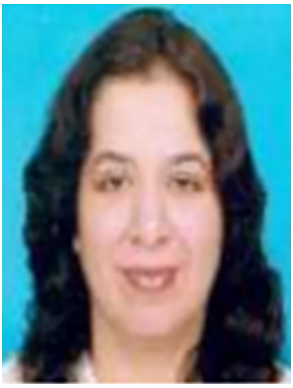
Wang, K., Lu, Z. M., & Hu, Y. J. (2013). A high capacity lossless data hiding scheme for JPEG images. *Journal of Systems and Software*, *86*, 1965–1975.

Xuan, G., Shi, Y. Q., Yao, Q., Ni, Z., Yang, C., & Gao, J. (2006). Lossless data hiding using histogram shifting method based on integer wavelets. In 2006 international workshop on digital watermarking. Lecture notes in computer science (Vol. 4823, pp. 323–332).

Yang, B., Schmucker, M., Funk, W., Brush, C., & Sun, S. (2011). Integer DCT-based reversible watermarking for images using companding technique. *International Journal of Electronics and Communications*, *65*, 814–826.

Zhang, Y., Shi-Ze, G., Zhe-Ming, L., & Hao, L. (2013). Reversible data hiding for btc-compressed images based on lossless coding of mean tables. *IEICE Transactions on Communications*, *96*(2), 624–631.

Aruna Malik received her B.Tech. in Computer Science and Engineering from Uttar Pradesh Technical University, Lucknow, India and M.Tech. in Computer Science and Engineering from National Institute of Technology, Jalandhar, Punjab, India. She is pursuing her doctoral degree in Computer Science and Engineering, at National Institute of Technology, Jalandhar, Punjab, India. Her research areas lie in the area of Data hiding and Image processing.



Geeta Sikka is presently working as an Associate Professor in the Department of Computer Science and Engineering at National Institute of Technology, Jalandhar. She received her Ph.D. in Computer Science and Engineering, form National Institute of Technology, Jalandhar, India. She did her Master's degree in Computer Science from Punjab Agricultural University, Ludhiana. Her research interests are Software Engineering, Databases, Data hiding and Data mining.

Harsh K. Verma is working as a Professor and Head of Computer Centre at National Institute of Technology, Jalandhar (India). He has done his Bachelor's degree in computer science and engineering in 1993 and Master's degree in Software Systems from Birla Institute of Technology, Pilani, in 1998. He received his Ph.D. degree from Punjab Technical University, Jalandhar (India) in 2006. He has many publications of international and national level to his credit. His research interests include Information security, Data hiding, Computer networks, Image processing and Scientific computing.