Adaptive block-based pixel value differencing steganography

Article *in* Security and Communication Networks · November 2016

DOI: 10.1002/sec.1676

2 authors:

Osama Hosam
Taibah University
34 PUBLICATIONS **85** CITATIONS
SEE PROFILE

Nadhir Ben Halima
Taibah University
56 PUBLICATIONS **56** CITATIONS
SEE PROFILE

SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks (2016)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1676

RESEARCH ARTICLE

Adaptive block-based pixel value differencing steganography

Osama Hosam[1,2]* and Nadhir Ben Halima[1]

[1] The College of Computer Science and Engineering in Yanbu, Taibah University, Medina, Saudi Arabia
[2] Informatics Research Institute,The City for Scientific Research and Technology Applications, Alexandria, Egypt

ABSTRACT

Steganography is the science of hiding secure data in digital carriers such as images and videos. Pixel value differencing (PVD) steganography algorithms embed data into images depending on pixel neighborhood differences. We have proposed PVD scheme for embedding secure data into digital images. The image is divided into non-overlapping 3×3 blocks. The block's median pixel is used as a reference for calculating pixel differences. The distance between the minimum and maximum differences are fine tuned for spreading the secure data on a wide range of image regions with high-intensity fluctuations. The embedding procedure embeds secure data into the content regions with edges and intensity transitions. Texture images provide higher embedding size compared with regular images. The results showed that the proposed algorithm is successfully able to avoid smooth regions in the embedding process. In addition, the proposed algorithm shows better embedding quality compared with the state of the art PVD approaches especially with low-embedding rates. Copyright © 2016 John Wiley & Sons, Ltd.

KEYWORDS
steganography, pixel value differencing, least significant bit, content regions

***Correspondence**
Osama Hosam, Informatics Research Institute, The City of Scientific Research and Technology Applications, Borg Al Arab, 21934, Alexandria, Egypt.
E-mail: ohosameldeen@mucsat.sci.eg

1. INTRODUCTION

Steganography is often given a formulation using the prisoner's problem introduced by Simmons [1] where two prisoners want to exchange escape planes. They communicate such that the warden cannot discover the secret communication. The warden has the authority to examine all communication between the two prisoners. A passive warden will eavesdrop the communication to find if there is secret data, while active warden will try to remove the secret data from the communication channel [2].The main objective of image steganography is to hide a message into an image in a way that prevents any attacker from detecting the message. Images that are used as a carrier for the secret data are called stego-image, and the original image is called cover image. Steganalysis techniques used the distortion of statistical connectivity of the embedding pixels in the carrier image to detect the existence of secret data [3]. Inserting data into the cover image changes the statistical properties of the image. This change opens a gate for steganalysis tools to detect, extract, or remove the secret data [4]. The distortion of the cover image is proportional to volume of secret data. Higher-image distortion means higher rates of secret data detection and vice versa. There are two types of steganography: spatial domain steganography and frequency domain steganography. Frequency domain steganography depends on calculating the equivalent transform domain representation of the cover image. Basic transformations used are discrete cosine transform (DCT), Fourier transform, and discrete wavelet transform. The authors in [5] updated the transform domain DCT coefficients to comply with a predefined dither-modulation based formula. Li and Wang [6] proposed JPEG compression-based data hiding scheme, and their approach modifies the quantization table of JPEG compression and inserts the secure data into the middle coefficients of DCT transform. Spatial domain steganography techniques are straightforward and convenient implementation. Compared with transform domain steganography, they lack imperceptibility and robustness to attacks. One of the most common spatial domain steganography techniques is least significant bit (LSB) [7]. The concept of

LSB is to embed data into the LSB of the binary representation of pixel value. The least significant bit is either 1 or 0. Changing it will add or subtract 1 from the pixel intensity value. For example, the value 128 (1000 0000) is a gray color that can be changed to either 129 (1000 0001) or left unchanged. The change is carried out according to a single bit value of the secret message. To overcome staganalysis tools, the locations of the carrier pixels can be selected randomly. Pseudo random number generator can be used to randomize the image distortion [8]. Random LSB increases security but gives the same imperceptibility level as the regular LSB. To increase the level of imperceptibility, an edge-based LSB steganography is proposed in [5,9]. Edge-based LSB steganography is mainly intended to avoid smooth areas and embed only in high-intensity variation areas. Zhang [10] proposed a plus-minus based steganography technique for embedding –2, –1, 0, 1, 2 into LSB of image pixels. The cover image is divided into regions according to their intensity variations. Regions with high-intensity variations hold 2, while regions with lower-intensity variations hold 1 and regions without variations, that is, smooth areas are avoided in the embedding process. Their algorithm spans the embedding to the variant regions to avoid statistical attacks. However, simple histogram analysis can detect the perturbation of plus-minus based techniques [11]. As stated in [12], a well-known steganography approach that can avoid histogram analysis is LSB+ technique. LSB+ updates the histogram to look natural by adding bits that intentionally updated the histogram. However, LSB+ still affects the image by adding visual artifacts. Kazem [13] proposed better steganography technique called LSB++ that adjusts the histogram with little distortion. The perturbation in LSB++ is carried out with the objective to maintain imperceptibility and connectivity statistics of the cover image. LSB++ changes the unused bits inside a specific unit to keep the original frequency unchanged. This helped in preserving the original histogram and introduced less distortion to the cover image. Another spatial domain steganography trend is pixel value differencing (PVD) based steganography. Wu and Tsai [14] first proposed PVD by using predefined range table. The cover image is divided into two pixels blocks by raster scanning order. Each two consecutive pixels are used for embedding secure data. The length of the secure data is defined according to the difference between the two consecutive pixels. PVD is enhanced thereafter by adding the capability of LSB to the original PVD, PVD+LSB algorithm [15,16], and PVD with ranges is proposed in [7,17]. The security of PVD steganography depends on the range table. The attacker can extract the embedded data if and only if he or she obtained the original range table. However, the range table is weak in that it is only defined by a power of two values. The power of two weakness limits the range of search for the attacker. In addition, the raster scanning order allows embedding adaptively according to only the vertical edges [18–20]. In this paper, we proposed random block based PVD embedding steganography. The block-based PVD embeds secure data with adaptability to vertical, horizontal, and diagonal content edges. In addition, the block location is selected randomly, and the locations are saved into private key. The main objective of our approach is to embed secure data into the content edges and avoids smooth regions even with higher-embedding capacity. Embedding into the content edges will not be detected by histogram analysis. As shown in Figure 1(b), the edge-based PVD [17] steganography results in difference map with clues for steganalysis. The attacker can easily detect the existence of secure data by examining the difference histogram. However, in Luo and Huang[18] approach, Figure 1(c) the authors randomized the embedding process but still their algorithm embeds into smooth regions and enables region-based difference histogram analysis to detect the existence of secure data.

In our approach, Figure 1(c) we employed the randomness of Luo approach and totally avoided the smooth regions in the embedding process. The cover image is divided into non-overlapping 3×3 pixel blocks. Pseudo random number generator is used to locate a random block. The random block pixel values are sorted in ascending order. The median of the block pixels is found. The

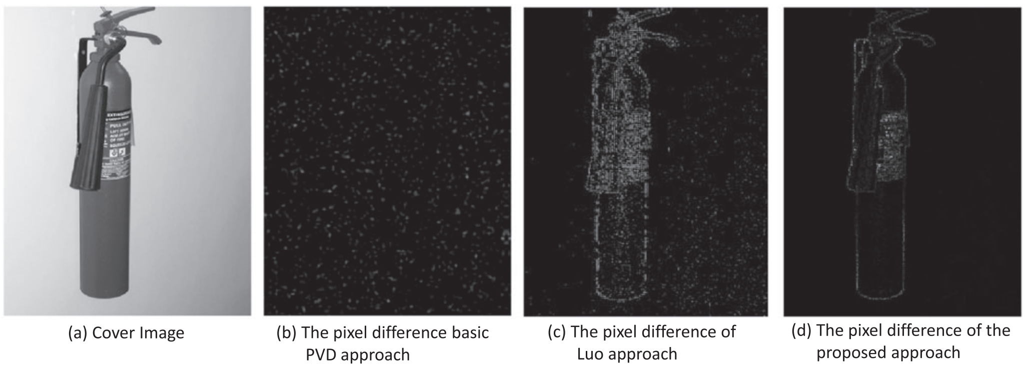

(a) Cover Image
(b) The pixel difference basic PVD approach
(c) The pixel difference of Luo approach
(d) The pixel difference of the proposed approach

Figure 1. Embedding secure data based on PVD steganography (a) the cover image, (b) the difference map of PVD approach, (c) the difference map of Luo and Huang approach, and (d) the difference map of the proposed approach. In the proposed approach, the content edges are used in embedding secure data, and the smooth regions are avoided even with higher capacity.

differences between each pixel value and the median value are obtained. The minimum difference is a measure of acceptable intensity fluctuation. The intensity fluctuation can increase up to the maximum difference without degrading the cover image quality. The pixel intensity can be perturbed by increasing its value to any value between the minimum difference and the maximum difference. This balance is selected according to the size of the secure data. In the extraction process, two important data are needed, the location of the block and the number of bits embedded in each block. Our approach provides higher imperceptibility to histogram analysis.

The remaining of the paper is organized as follows. Firstly, the PVD-based algorithms are explored with a detailed explanation of the basic PVD proposed by Wu and Tsai[14]. Secondly, the proposed approach is introduced. Lastly, the experimental results and discussions are introduced with a comparison with the latest PVD steganography approaches.

2. Basic PVD Algorithm

The PVD techniques such as the algorithm proposed by Wu and Tsai [14] divide the image into non-overlapping two consecutive pixels block. The difference between each pair of pixels in each block is calculated, and the secure data are embedded into each block according to that difference. Larger difference reflects larger modification. The algorithm is explained in the following steps:

PVD – Step1: The cover image is arranged into a single vector by using raster scanning technique. The vector is then divided into non-overlapping two pixels block. A difference is calculated for each two consecutive pixels $p_{i,j}$ and $p_{i,j+1}$, so $d_i = p_{i,j+1} - p_{i,j}$. Given that $p_{i,j}$ is in the range [0, 255].

PVD – Step2: The absolute difference is classified into k ranges $R_i, R_i \in [L_i, U_i]$, where L_i is the lower bound of the range and U_i is the upper bound of the range. The number of bits embedded in each range is decided according to the range width w_i. An example of the range table is [0, 7], [8, 15], [16, 31], [32, 63], [64, 127], and [128, 255].

PVD – Step3: Extract n bits of the secret data to be hidden according to the range width. The number of bits to be embedded is decided according to the following formula:

$$n = \lfloor log_2 (w_k) \rfloor \qquad (1)$$

Convert the extracted bits into decimal value b. For example, if $n = 3$, then 3 bits are extracted from the secure data. Suppose the extracted bits are $(110)_2$ then they are converted into decimal value $b = 6$.

PVD – Step 4: Calculate the new difference $d_i^{'}$ by using the following formula:

$$d_i^{'} = \begin{cases} l_k + b & for\ d \geq 0 \\ -(l_k + b) & for\ d < 0 \end{cases} \qquad (2)$$

PVD – Step 5: p_i and p_{i+1} are modified to hold the secret bits b by using the following formula:

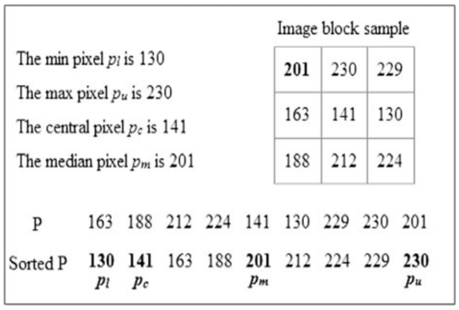
$$\left(p_i^{'}, p_{i+1}^{'}\right) = \begin{cases} p_i - \lceil \frac{(d_i^{'} - d)}{2} \rceil, p_{i+1} + \lfloor \frac{(d_i^{'} - d)}{2} \rfloor, & d \in odd \\ p_i - \lfloor \frac{(d_i^{'} - d)}{2} \rfloor, p_{i+1} + \lceil \frac{(d_i^{'} - d)}{2} \rceil, & d \in even \end{cases} \qquad (3)$$

The range table is very important in the extraction process because the new difference $d^{'}$ in the stego-image will always fall into the same embedding region R_k as the original embedding difference d. So, the embedded bits can be extracted simply by $b = |d^{'}| - d$. For example, suppose the two consecutive pixel values 113, 122 are to be used for embedding. The pair of pixels are used for embedding part of the secret message $(110111)_2$. The range table is [0 7], [8 15], [16 31], [32 63], [64 127], and [128 255]. The difference d is $113 - 122 = 9$, 9 is found in the second range with width $w = 8$. The number of bits to be extracted n is $log_2(8) = 3$. Extract 3 bits from the secret message and convert them to decimal $(110)_2 = 6$. Now, $d^{'} = l + b = 8 + 6 = 14$. The new stego-values of both pixels are $(113 - \lceil(14-9)/2\rceil, 122 + \lfloor(14-9)/2\rfloor)$ or 110, 124. In the extraction process, the difference between 110, 124 is obtained, which is 14. The embedded secret data in decimal are $b = |14| - l$ or $b = 14 - 8 = 6$. The binary equivalent is $(110)_2$, which is the original embedded secret value. Notice that each two consecutive pixels embed single bit string. In our approach, each pixel carries different secure bit string but with the same string length in each block. As explained by Luo and Huang [18], PVD-based steganography scans the image using raster scanning, and the vertical edges are employed into the embedding process. To overcome this limitation, we proposed an algorithm that divides the image into square areas. The algorithm considers horizontal, vertical, and diagonal connectivity of the pixels in the embedding process. Therefore, more imperceptibility and quality of embedding is introduced. The basic PVD algorithm considers all the image regions with variable length even the smooth regions are employed in embedding, which introduces good clues for steganalysis tools, for example, simple histogram analysis can reveal the existence of hidden data [21]. In contrary, in our proposed algorithm, we forced the embedding into only the regions with abrupt changes. Regions with regular or smooth transition are not considered in the embedding process.

3. THE PROPOSED ALGORITHM

3.1. The embedding procedure

The algorithm takes a standard $m \times n$ grayscale image and divides it to non-overlapping 3×3 pixels blocks. $p_{i,j}$ is the center pixel of the block, and the remaining 8 pixels are $p_{i,j-1}, p_{i-1,j-1}, p_{i-1,j}, p_{i-1,j+1}, p_{i,j+1}, p_{i+1,j+1}, p_{i+1,j}$, and $p_{i+1,j-1}$.



The min pixel p_l is 130
The max pixel p_u is 230
The central pixel p_c is 141
The median pixel p_m is 201

Image block sample

201	230	229
163	141	130
188	212	224

P	163	188	212	224	141	130	229	230	201
Sorted P	**130** p_l	**141** p_c	163	188	**201** p_m	212	224	229	**230** p_u

Figure 2. A sample pixels block with the corresponding median, min, max, and central pixels.

Step 1: Select random block, and save the random location of the block into the symmetric privately shared key *K*.
Step 2: Get block pixels median. First, the pixels $p_{i,j-1}$, $p_{i-1,j-1}$, $p_{i-1,j}$, $p_{i-1,j+1}$, $p_{i,j}$, $p_{i,j+1}$, $p_{i+1,j+1}$, $p_{i+1,j}$, and $p_{i+1,j-1}$ are represented by 1*D* vector *P* as shown in Figure 2. *P* is then sorted in ascending order. The first item in the sorted *P* is p_l or the lowest value pixel. The last item in the sorted *P* is p_u or the highest value pixel.

The median pixel p_m of the vector *P* of the block pixels can be found by using the following formula

$$p_m = \begin{cases} P(\lceil N/2 \rceil) & N \in odd \\ \frac{P(\frac{N}{2})+P(\frac{N}{2}+1)}{2} & N \in even \end{cases} \tag{4}$$

In the sorted *P*, $p_l = P(1)$, and $p_u = P(N)$, where *N* is the number of block pixels, which in our case is 9. 9 is an odd number and the location of the median pixel is always the 5^{th} location.
Step 3: Select the pivot pixel, and the pivot pixel is p_m and can be changed to p_l or p_u according to the user selection. Calculate the difference between each pixel in the block B_i and p_m and put the result into nine elements vector *D*.

$$D = |P - p_m| \tag{5}$$

Remove the fifth element from *D*. The fifth element is the median difference that will always be zero. The first and the last elements are removed from *D* in case p_l and p_u are selected as pivot pixel, respectively. Using the difference vector *D*, calculate d_{min} and d_{max} as the minimum and maximum differences, respectively.
Step 4: Calculate the fluctuation range *r*. The fluctuation range is the range in which the perturbation is allowed for block pixels. For example, if there are block pixels with $d_{min} = 10$ and $d_{max} = 25$, this gives a fluctuation range of 15, which is $d_{max} - d_{min}$, so the pixel with d_{min} is allowed to increase any value from 0 to 15. The fluctuation range is controlled by using parameter α: when α is 0, the fluctuation range is d_{min}, when it is 1 the fluctuation range is d_{max}. Formally, the block's fluctuation range *r* is defined as follows:

$$r = \alpha(d_{max} - d_{min}) + d_{min} \tag{6}$$

By experiment, *r* is found higher in blocks with abrupt changes. In blocks with smooth areas, *r* is lower, and in single background color regions, *r* will be zero. In our algorithm, blocks with $r > 1$ will be considered in embedding. Blocks with $r \leq 1$ will be ignored. The corresponding random location of the block with $r \leq 1$ is not saved in the extraction key. The condition of $r \leq 1$ forces embedding into only the regions with sufficient intensity fluctuations. The same range table proposed by Wu and Tsai [14] is adopted. The fluctuation range *r* is assigned to one of the following ranges [0 7], [8 15], [16 31], [32 63], [64 127], and [128 255].
Step 5: According to *r* value, calculate how many bits (*n* bits) can be embedded into each block pixel. Higher *r* values allow more bits to be embedded into the block pixels. For example, if $r = 16$, $log_2(16) = 4$ bits can be embedded into each block pixel. Formally, the number of bits *n* that can be embedded in each block pixel is decided by the following mathematical formula:

$$n = \lfloor log_2(r) \rfloor \tag{7}$$

The formula specifies that *n* is logarithmically proportional to *r*. With $r = 2$, the number of bits *n* is 1. With $r = 4$, the number of bits *n* is 2, and so on.
Step 6: Convert the secure data into binary vector. Extract *n* bits from the binary vector. For better memory performance, it is recommended to convert only part of the secure data into binary vector and then embed it into sufficient number of blocks. Then extract another part of the secure data and embed it into sufficient number of blocks. This process is repeated until the entire secure data are finished.
Step 7: Convert the extracted *n* bits into decimal value *h*. For example, if the secure data are $(1001010010101.)_2$ and $n = 3$, then the algorithm extracts the first three bits to the left $(100)_2$ then converts them to the corresponding decimal value that is $h = 4$ in this example.
Step 8: For each pixel in the block B_i including the median pixel, change their value to the corresponding stego value according to the following formula:

$$p_i' = p_i - (mod(p_i, 2^n)) + h \tag{8}$$

Repeat the aforementioned steps for new randomly selected block until the secure data are totally embedded into the image.

3.2. The extraction procedure

The extraction process is blind, that is, the cover image is not needed in the data extraction process. The secure key *K* holds the block locations, which are randomly selected in the embedding process. Steps 1 to 4 are the same as those for the embedding procedure.

Step 1: Use the block locations index from K to obtain the first block's location in the stego-image.
Step 2: Extract the pixel values p_i from the block, sort them and find p_m using Equation 4.
Step 3: Calculate the corresponding differences between p_m and all pixels in the block.
Step 4: Calculate r and find the corresponding range's lower bound l_i in the range table, where i is the index of the range in the range table. The number of pixels n is calculated by

$$n = log_2(l_i) \tag{9}$$

Step 5: Extract the embedded decimal value from each block pixel. The decimal value is obtained for stego pixel $p_i{}'$ using the following formula:

$$h = mod\left(p_i^{\prime}, 2^n\right) \tag{10}$$

This process is repeated for all the nine block pixels.
Step 6: Convert the decimal value h to the corresponding binary value. The binary value must be consistent with the number of bits n. As an example: if the decimal value is $h = 0$ and the number of bits $n = 3$, then the resulting binary value is $(000)_2$. Concatenate the binary value to the extracted data stream.
Step 7: Using the security key K, move to the next random block location and repeat Steps 1 to 6 until the entire secure data are extracted from the stego-image.

For clarification of embedding and extraction procedures, we give an example to show a complete step-by-step process.

Suppose the selected random block has the following pixel values: 211 228 230 164 131 140 198 231 and 225. The secure binary data are (1110 1001 1111 0000 0101 1011 1111 0010 1110). The median value of the block pixels is $p_m = 211$. The distances between 211 and the remaining values are 17 19 47 80 71 13 20 and 14. The minimum distance is d_{min}=13, and the max distance is d_{max}=80. Suppose $\alpha = 0.20$, then the fluctuation range is $r = 0.20(80 - 13) + 13 = 26.4$. Then r represents how many bits can be embedded in each block pixel. $n = max(\lfloor log_2(26.4) \rfloor, 1) = 4$, so 4 bits can be embedded into the nine block pixels. A total of $4 \times 9 = 36$ bits can be embedded into this block. The first 4 bits from the secure data will be embedded into the first block pixel. The first 4 bits of the secure data are $(1110)_2$ considering MSB is 0 and LSB is 1 then the corresponding decimal value is $h = 7$. The block pixel value is 211. Using Equation 8, the new block pixel value is 215. The next pixel value 228 holds the secure binary value $(1001)_2$, $h = 9$. Using Equation 8, the new value of 228 is 233. This process is repeated for pixels 230 164 131 140 198 231 and 225. The new stego-block pixel values are 215, 233, 239, 160, 138, 141, 207, 228, and 231. In the extraction process, n together with the block location are given. The location of the random block gives the aforementioned stego-block with the stego-values 215, 233, 239, 160, 138, 141, 207, 228, and 231. The first pixel 215 holds 4 bits of data that can be obtained using Equation 10: $24 = 16, mod(215, 16) = 7$, so $h = 7$ is converted to binary with binary width $= 4(1110)_2$. LSB is leftmost bit. The same procedure is applied on the remaining stego-pixels to extract the entire secure data.

4. RESULTS AND DISCUSSION

For experimental results of our algorithm, we considered different types of images. For example, the algorithm is tested against crowded scenes, natural scenes, and background images. We have collected images from the following online locations:

- USC-SIPI image database [22] is used in our experiments. USC-SIPI image database is a collection of various volumes of image with different sizes and color schemes. All images are TIFF formats. Sizes of the volumes vary such as 256×256 pixels, 512×512 pixels, or 1024×1024 pixels. All experiments are carried out using grayscale images. Therefore, we standardized the color scheme of the images to be grayscale.
- UCID [23] Uncompressed colored image database, version 2 of the database is downloaded in colored uncompressed format. Version 2 has 1300 images of size 384×512 or 512×384.

The cover image distortion can be measured using two error measure metrics, peak signal-to-noise ratio (PSNR) and mean square error (MSE). These two measurements can be defined mathematically as

$$MSE = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{i=1}^{n} (I_{ij} - S_{ij})^2 \tag{11}$$

$$PSNR = 10 log_{10} \left(\frac{255^2}{MSE} \right) dB \tag{12}$$

where I_{ij} and S_{ij} are the grayscale intensity values of the pixels in the cover and stego-image, respectively. m and n are the numbers of rows and columns in both cover and stego-images. The accepted PSNR should be higher than 40 dBs. Higher PSNR means higher embedding quality, while lower PSNR means the cover image is severely distorted, and the distortion can be noticed by HVS and easily detected by steganalysis tools. wPSNR [24] is currently the most commonly used metric for embedding quality. It is well known that PSNR does not calculate the stego-image quality well. wPSNR is a more accurate metric, because it considers the texture masking effect. wPSNR is more accurate in spatial domain and is not a feasible metric in frequency domain embedding. Texture masking effect or noise visibility function (NVF) is the increasing of imperceptibility of noise-like secure data because of the existence of a similar noisy-effect in the cover image. In general, the secure data embedding is difficult to be noticed in the texture regions. wPSNR is calculated by

$$wPSNR = 10log_{10}\left(\frac{max(I)^2}{\|NVF(S-I)\|^2}\right) \tag{13}$$

where I and S are the cover and stego-images, respectively. A universal quality index of the stego-images is proposed in [25]. The quality index Q is calculated with respect to the average and the variance of the cover and the stego-images. Formally, Q is mathematically represented by

$$Q = \frac{4\sigma_{IS}\bar{I}\bar{S}}{\left(\sigma_I^2\sigma_S^2\right)[(\bar{I})^2+(\bar{S})^2]} \tag{14}$$

where $\bar{I}$ and $\bar{S}$ are the averages of the cover and stego-images, respectively. $\bar{I}$ and $\bar{S}$ are calculated by

$$\bar{I} = \frac{1}{m\times n}\sum_{i=1}^{m\times n} I_i \tag{15}$$

$$\bar{S} = \frac{1}{m\times n}\sum_{i=1}^{m\times n} S_i \tag{16}$$

σ_I^2 and σ_S^2 are the variance of both cover and stego-images are respectively calculated by

$$\sigma_I^2 = \frac{1}{(m\times n)-1}\sum_{i=1}^{m\times n}(I_i-\bar{I})^2 \tag{17}$$

$$\sigma_S^2 = \frac{1}{(m\times n)-1}\sum_{i=1}^{m\times n}(S_i-\bar{S})^2 \tag{18}$$

For calculating the embedding capacity, we calculated the bits per pixel as the bit rate (BR) of embedding secure data in a specific cover image. BR is mathematically represented as

$$BR = \frac{msg}{m\times n}\; bpp \tag{19}$$

where msg is the size of the secure message and the size of the cover image is $m \times n$.

4.1. Perturbing the Block's Pivot Pixel

Three volumes with 20 images, each is used in this experiment. The volumes used are regular images, texture images, and aerials images. The proposed algorithm is applied on the three volumes. The images are divided into non-overlapping 3×3 pixel blocks. Eight-neighborhood pixels scheme is used. The block is rearranged into a vector for finding the central pixel (p_c), the median pixel (p_m), the min pixel (p_l), and the max pixel(p_u). The central pixel p_c is the fifth pixel in the block vector. In each case, the resulting pixel is perturbed and used for embedding variable-length secret bits. The secret bits length is determined by using the difference between the pivot pixel – which is one of the aforementioned pixels and eight neighbor pixels. Lower difference reflects lower embedding capacity, and higher difference reflects higher embedding capacity. In [26], the authors used p_c as a pivot pixel but embedded in all the image blocks. In our approach, we embedded only in the selected random blocks with $r > 1$. The PSNR is calculated when p_m pixel is used as a pivot pixel and compared with PSNR of the same algorithm when using p_c, p_l and p_u pixels as a pivot pixel. The results are shown in Table I.

From Table I, using the median pixel p_m as a pivot pixel increases the quality of the stego-image, PSNR for p_m based embedding is 63.09 which is higher compared with 62.52 for p_c based embedding. The main reason is that the suitable difference is found in regions with more fluctuations and intensity variations. It spreads the secret data regularly on more pixels; this appears in Table I because p_c based embedding uses 1775 pixels on average, while p_m based embedding uses 1844 pixels on average for embedding the 3000 secret bits. Those pixels are the location of the secret data, and they are referred to as location map. The location map is a map representing the locations of the pixels used for embedding. In Figure 3, the location map is presented for each type of embedding. The p_c based embedding technique results in a blocky effect as shown in Figure 3(a). This weakness point can be used as a clue for steganalysis tools. For p_l and p_u based embedding, the algorithm has a bulky embedding effect, that is, it uses large differences. The bulky effect is apparent in Figure 3(b) and (c) for p_l and p_u based embedding, respectively. The locations of the secret data are bulked into upper areas of the image. Embedding in the lower and higher pixel intensities decreases the quality of the stego-image. It can be shown from Table I that p_u and p_l based embedding produces stego-image with an average quality of 58 dB, while p_m based embedding produces stego-images with an average quality of 63 dB. In Figure 3(a) and (b), the lower part of the image is background area. It has no abrupt changes in pixel intensities, and hence, the pixels in this area are not suitable for embedding. Any change in smooth areas is noticeable by human visual system (HVS). Embedding data using p_m based algorithm results in lower differences, hence lower probability of using the pixel for embedding, as embedding is carried out only in higher frequencies blocks. As shown in the figure, the lower image areas are used mostly for embedding when using p_c based embedding, while this area is relatively avoided when using p_m based embedding. As shown in Table I, embedding in F16 image does not depend on the median pixel. Using p_l or p_u pixels will increase the possibility of embedding because

Table I. Inserting 3000 bits of secret data into cover images using p_c, p_l, p_u, and p_m pivot pixels.

		p_c		p_l		p_u		p_m	
Image type	**Image name**	**PSNR**	**No. of pixels used**	**PSNR**	**No. of pixels used**	**PSNR**	**No. of pixels used**	**PSNR**	**No. of pixels used**
	Car	62.93	1857	56.28	1248	59.51	1401	63.56	1966
	Baboon	58.7	1437	54.29	1075	54.58	1083	60.38	1537
Regular images	**F16**	60.93	2080	65.91	2277	65.79	2376	60.14	1886
	Pepper	62.63	2154	60.66	1792	60.64	1808	63.57	2099
	Sailboat	62.27	1895	57.81	1550	56.97	1469	62.78	1955
	Grass	58.58	1449	52.46	1081	53.36	1064	59.39	1523
	Straw	56.64	1307	50.55	912	51.84	1028	57.14	1361
Textures	**Bubbles**	58.76	1572	54.23	1173	52.83	1132	59.18	1554
	Wood	59.67	1388	55.51	1177	55.41	1124	60.01	1375
	Bark	58.85	1410	52.57	1037	53.92	1081	59.43	1511
	Pentagon	70.69	2188	67.45	1772	66.74	1634	71.35	2317
	Miramar	62.15	1815	58.04	1449	56.61	1260	63.38	2064
Aerials	**San Fran.**	70.46	2016	64.91	1399	65.26	1644	70.51	2169
	Hills	64.17	2048	58.47	1448	60.31	1543	65.13	2182
	San Fran2	70.46	2016	64.91	1399	65.26	1644	70.51	2169
Average		**62.52**	**1775**	**58.27**	**1385**	**58.6**	**1419**	**63.09**	**1844**

PSNR, peak signal-to-noise ratio.

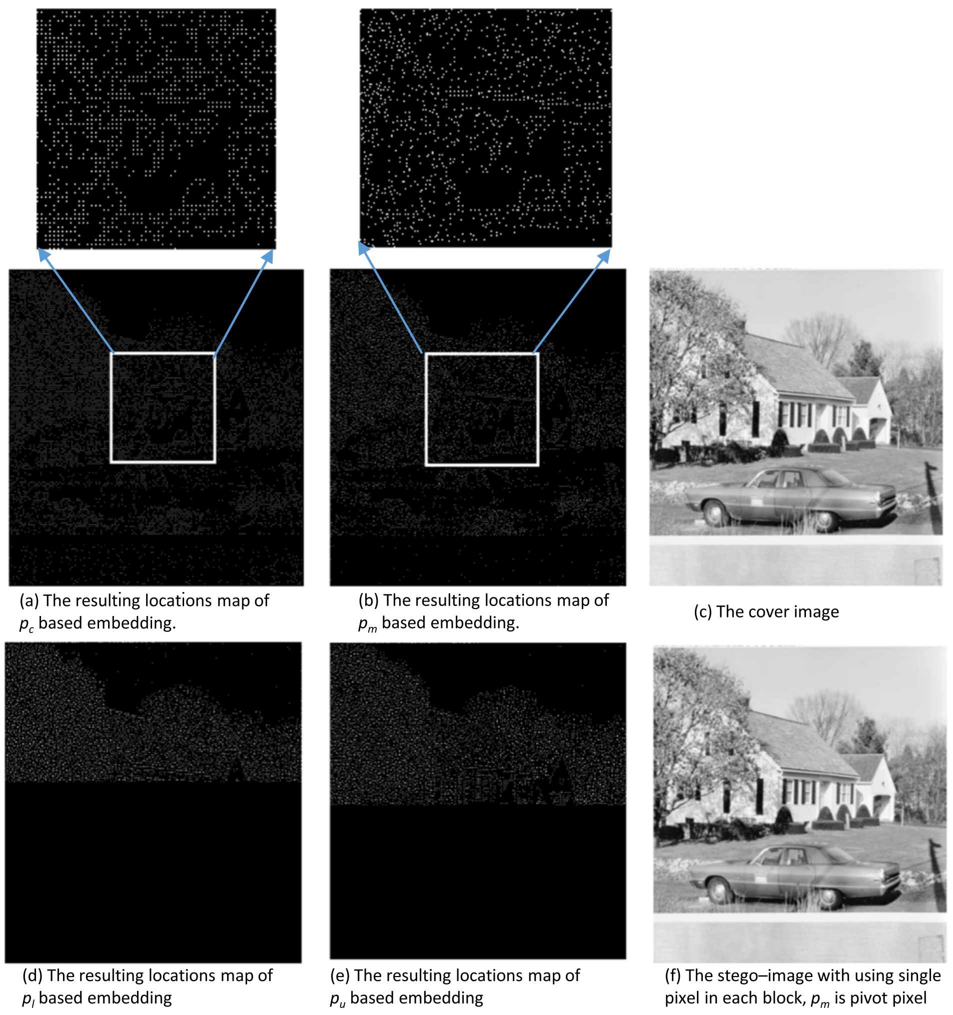

(a) The resulting locations map of p_c based embedding.
(b) The resulting locations map of p_m based embedding.
(c) The cover image
(d) The resulting locations map of p_l based embedding
(e) The resulting locations map of p_u based embedding
(f) The stego–image with using single pixel in each block, p_m is pivot pixel

Figure 3. The changed pixels map for embedding 3000 bits into the car image using (a) central pixel, (b) median pixel, (d) min pixel, and (e) max pixel. (c) and (f) are respectively the cover and stego-images for car image.

the pixel values are located in higher-level intensity due to the large area of white background.

4.2. Perturbing all the block pixels

To view the quality of the proposed approach, we used three image volumes, namely, regular images, texture images, and aerials images. The images are of different sizes such as 512 × 512, 420 × 420 and 400 × 400. The quality Q, PSNR, and BR are calculated for each volume. The results are listed in Table II. From Table II, we find that smaller α has the effect of spreading the embedding of secure data on more edge contents. While larger α has the effect of spreading, the embedding of the secure data deeper into the pixels, that is, starting from the LSB and deeper to the MSB of each pixel. For the Baboon image, with higher α ($\alpha = 0.20$) the embedding BR is 2.15 bpp. With lower α ($\alpha = 0.10$), the embedding BR is lowered to 1.60 bpp. Higher α means embedding more bits in each block, and lower α means searching the blocks with sufficient fluctuation range and then limits the embedding to only the blocks with higher fluctuation range. In the regular images volume, the highest capacity is achieved with Baboon image. The difference between embedding data into baboon image and other images in the regular images volume is that raising α means providing more space for embedding secure data ($BR = 3.13$ with $\alpha = 0.50$). However, raising α with other images does not significantly increase the embedding rate ($BR = 1.91$ with $\alpha = 0.50$ for F16 image). That is mainly because baboon image has more fluctuations in the block's intensity compared with other images in the same volume. The embedding capacity is increased with the increase of textures in the image. Texture images introduced the highest embedding capacity ($BR = 3.83$ bpp for $\alpha = 0.50$ compared with 2.06 bpp for regular images and 2.92 bpp for aerials).

wPeak signal-to-noise ratio is a more suitable metric to the proposed embedding algorithm. The algorithm

Table II. The embedding quality of the proposed approach.

		$\alpha = 0.10$			$\alpha = 0.20$			$\alpha = 0.50$		
Image volume	**Image name**	**BR**	**PSNR**	**Q**	**BR**	**PSNR**	**Q**	**BR**	**PSNR**	**Q**
Regular images	**Lena**	0.86	44.31	0.996	1.19	41.09	0.992	1.94	34.81	0.971
	Car	1.1	43.8	0.996	1.5	40.03	0.992	2.28	33.64	0.969
	Baboon	1.6	42.54	0.996	2.15	39.03	0.991	3.13	32.95	0.966
	F16	0.88	43.33	0.995	1.2	39.75	0.991	1.91	33.37	0.966
	Pepper	0.87	44.06	0.995	1.22	40.89	0.991	1.98	34.43	0.969
	Couple	0.65	48.21	0.994	0.96	44.4	0.989	1.63	38.08	0.966
	Tree	1.25	41.8	0.995	1.68	38.19	0.989	2.52	31.87	0.958
	House	0.7	46.14	0.992	1.02	42.62	0.985	1.67	35.98	0.955
	Tiffany	0.66	48.55	0.996	0.95	45.06	0.992	1.66	38.73	0.969
	Splash	0.54	47.61	0.994	0.79	43.65	0.989	1.48	37.22	0.963
	Sailboat	1.2	42.19	0.995	1.64	38.48	0.989	2.45	32.14	0.96
Average		**0.94**	**44.78**	**0.994**	**1.3**	**41.2**	**0.99**	**2.06**	**34.84**	**0.964**
Textures	**Grass**	2.68	36.18	0.996	3.39	32.41	0.991	4.5	26.68	0.965
	Straw	3.29	32.71	0.995	3.94	29.49	0.989	4.95	23.48	0.961
	Bubbles	2.87	34.73	0.996	3.52	31.3	0.992	4.56	24.99	0.969
	Wood	2.78	35.81	0.997	3.47	31.8	0.992	4.56	24.98	0.968
	JellyBeans	0.48	46.33	0.997	0.66	42.99	0.994	1.01	36.92	0.98
	Bark	3	34.28	0.996	3.67	30.78	0.992	4.72	24.61	0.969
	Brick	1.3	45.14	0.996	1.8	41.32	0.991	2.81	34.32	0.96
	Gravel	1.29	43.19	0.997	1.79	39.43	0.993	2.7	33.22	0.974
	Leather	2.41	38.81	0.995	3.07	35.29	0.99	4.1	29.05	0.964
	Sand	1.76	42.27	0.996	2.36	38.74	0.991	3.41	32.3	0.965
	Water	3.13	33.89	0.995	3.8	30.31	0.99	4.83	24.02	0.963
Average		**2.27**	**38.48**	**0.996**	**2.861**	**34.89**	**0.991**	**3.83**	**28.59**	**0.967**
Aerials	**Pentagon**	1.25	44.51	0.996	1.74	41.16	0.992	2.7	34.61	0.967
	Miramar	1.63	42.16	0.996	2.2	38.7	0.992	3.22	32.57	0.967
	San Fran.	1.74	42.75	0.995	2.35	39.3	0.991	3.37	33.03	0.963
	Hills	1.57	42.78	0.996	2.12	39.34	0.991	3.14	33.16	0.965
	San Fran2	1.74	42.75	0.995	2.35	39.3	0.991	3.37	33.03	0.963
	Earth	1.16	44.51	0.996	1.6	40.79	0.992	2.47	34.66	0.967
	Oakland	1.49	44.33	0.995	2.05	40.7	0.991	3.09	34.34	0.961
	Richmond	1.15	46.22	0.995	1.61	42.63	0.991	2.5	36.27	0.962
	SanDiego	1.26	43.5	0.996	1.7	39.72	0.992	2.55	33.43	0.968
	Wash-ir	1.71	41.88	0.996	2.31	38.4	0.991	3.34	31.98	0.964
	Stockton	1.03	47.01	0.996	1.47	43.49	0.992	2.39	36.92	0.966
Average		**1.43**	**43.85**	**0.996**	**1.95**	**40.32**	**0.991**	**2.92**	**34**	**0.965**
Total average		**1.54**	**42.37**	**0.996**	**2.03**	**38.8**	**0.991**	**2.93**	**32.47**	**0.966**

The embedding procedure is applied on regular, textures, and aerials image volumes.
PSNR, peak signal-to-noise ratio; BR, bit rate.

efficiently locates the texture regions and embeds data adaptively into those variable intensity regions. Therefore, the algorithm adapts with NVF. Texture regions have more noise that is suitable for embedding noisy data. The results of wPSNR compared with PSNR are shown in Figure 4. Higher BR represents texture images; lower BR represents regular and smooth images. For example, Grass is a texture image, which is shown to the right side of the figure, and Splash is regular image because it is located to the left side of the figure. wPSNR with texture images is higher than PSNR. However, with regular images, the accuracy of wPSNR is lowered, some wPSNR measurements are lower than PSNR, and some others are higher than PSNR. With texture images, (right-hand side of the figure) wPSNR metric is always higher than PSNR metric. As a result, wPSNR is a more suitable metric in our approach only in the case of embedding into texture images.

The algorithm efficiently locates texture regions and embeds secure data into them. Figure 5 shows stego-images of car and sailboat and the corresponding difference maps. The secure data are mainly located in texture regions such as trees. Trees represent an extensive texture

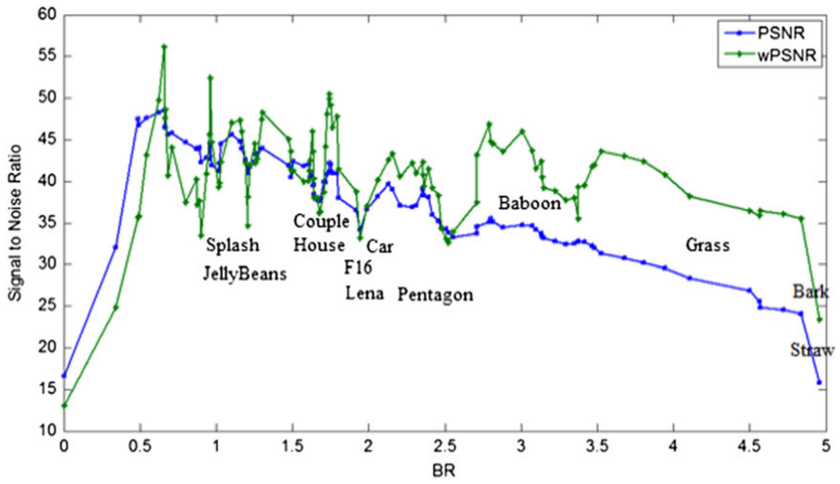


Figure 4. Peak signal-to-noise ratio (PSNR) compared with wPSNR with regard to the embedding bit rate. When BR increases, wPSNR depicts more embedding quality than PSNR. The right-hand side of the image describes images with intensive textures, while the left-hand side of the figure represents regular and smooth images.

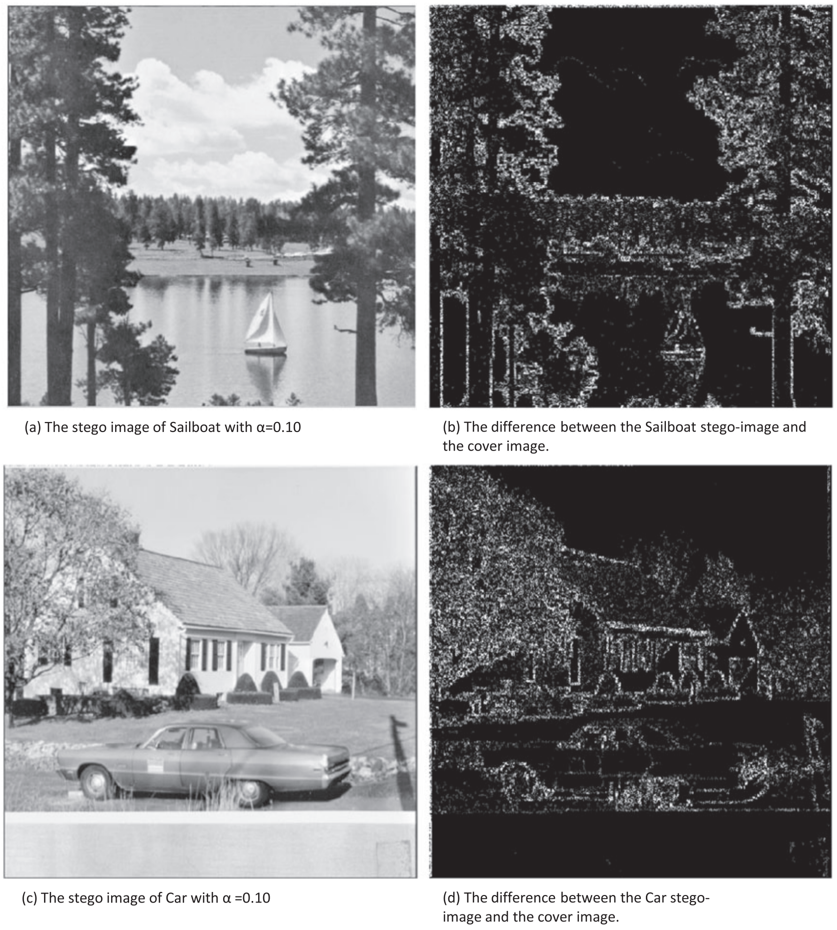

(a) The stego image of Sailboat with α=0.10
(b) The difference between the Sailboat stego-image and the cover image.
(c) The stego image of Car with α =0.10
(d) The difference between the Car stego-image and the cover image.

Figure 5. The difference maps of the proposed algorithm showing the locations of the secure data in the stego-images (a) stego-image of sailboat image, (b) the difference map sailboat image, (c) the stego-image of car image, and (d) the difference map of car image.

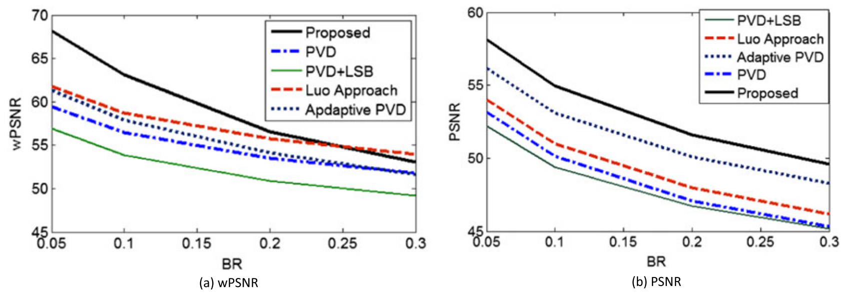


Figure 6. The quality of embedding of the proposed approach with respect to the embedding capacity together with the state of the art approaches (a) wPSNR and (b) PSNR. PSNR, peak signal to noise ratio.

with intensive fluctuations in pixel intensities. Smooth regions such as the sky in both images and ground in car image are avoided in the embedding process.

4.3. Comparison with previous approaches

The quality of embedding is measured with different BRs. The average values of wPSNR and PSNR are measured for the proposed approach and previous approaches with BRs varying from 0.05 to 0.30. The average wPSNR and PSNR is taken at each BR. For example, the algorithm is applied on UCID database with fixing BR to 0.05 and α to 0.10; the resulting average values of wPSNR and PSNR are stored. This procedure is repeated for all values of BR starting with 0.05 and ending with 0.30. A plot of the resulting wPSNR and PSNR with respect to the embedding BR is depicted in Figure 6. Figure 6(a) shows that the embedding quality of the proposed approach outperforms other approaches. The adaptive PVD approach has similar trend to the proposed approach. Adaptive PVD approach and the proposed approach embed secure data into the regions with intensity fluctuations such as edges and gradually reduce the embedding rate when the intensity fluctuations are low. In Figure 6(a), the algorithm shows higher quality with low-embedding rate, but the quality is reduced with higher embedding rates. The results of PSNR, shown in Figure 6(b), confirm that the main competitor of the proposed approach is the adaptive PVD approach.

The problem with the previous approaches such as PVD [14], PVD-LSB [15] Luo's approach, [18] and adaptive PVD [17] is that they did not consider neighbor pixels of all directions in defining the suitable difference for embedding secret data. Therefore, background areas in the image are used in the embedding process. HVS and steganalysis tools easily discover changes in the image smooth regions. For solving the problem of embedding into the background areas, the proposed algorithm embeds data into the block pixels if the fluctuation range r is greater than 1. Therefore, the algorithm embeds data into the content edges and leaves smooth areas as shown in Figure 7. From Figure 7(b), smooth areas are not considered in the embedding process, and hence, the quality of the stego-images will be increased. The algorithms are applied on Lena image with embedding 274023 bits of data, and the resulting PSNR is 43.65. When the size of the secret data is increased to 317007 bits, the quality is reduced to 41.71 dB. Even if the embedding size is increased, but still the smooth areas are avoided in the embedding process as shown in Figure 7(c). In each embedding block, the distance between the minimum median difference (d_{min}) and maximum median difference (d_{max}) is controlled with the parameter α. If $\alpha = 1$, the maximum difference is used. If α=0, the minimum difference is used. The increase in the capacity is controlled with α parameter. A median difference of zero is not considered, because it represents absolute smooth regions.

We compared the proposed approach with Luo's approach [18]. A similar test for embedding into smooth images is used. The ucid01162 image with fire extinguisher on a plain background is shown in Figure 7(d). Luo *et al.* embed the watermark into the content edges. With lower capacity embedding, shown in Figure 7(e), the algorithm embeds into regions with abrupt changes such as edge regions. However, with higher capacity, shown in Figure 7(f), the algorithm spreads embedding to the remaining smooth parts of the image, as shown in the right side of Figure 7(f). This opens a way for steganalysis for easier detection of pixel connectivity changes such as steganalysis tool proposed in [3]. We solved the problem of embedding into the smooth areas by spreading vertically inside each pixel. As shown in Figure 7(h), with lower embedding capacity ($BR = 0.05$), the proposed approach embeds data into the regions with abrupt changes, such as the written instructions region on the distinguisher label. When the embedding capacity increases ($BR = 0.10$), the algorithm embeds into the same regions with more bits assigned to each pixel. The algorithm is not spreading the watermark into smooth regions. The algorithm in this way increases imperceptibility of the embedded secure data. Figure 7(g) shows the stego-image of the fire distinguisher with embedding $BR = 0.05$.

4.4. Security evaluation

Two important techniques are used for evaluating the security of the proposed approach, namely, histogram anal-

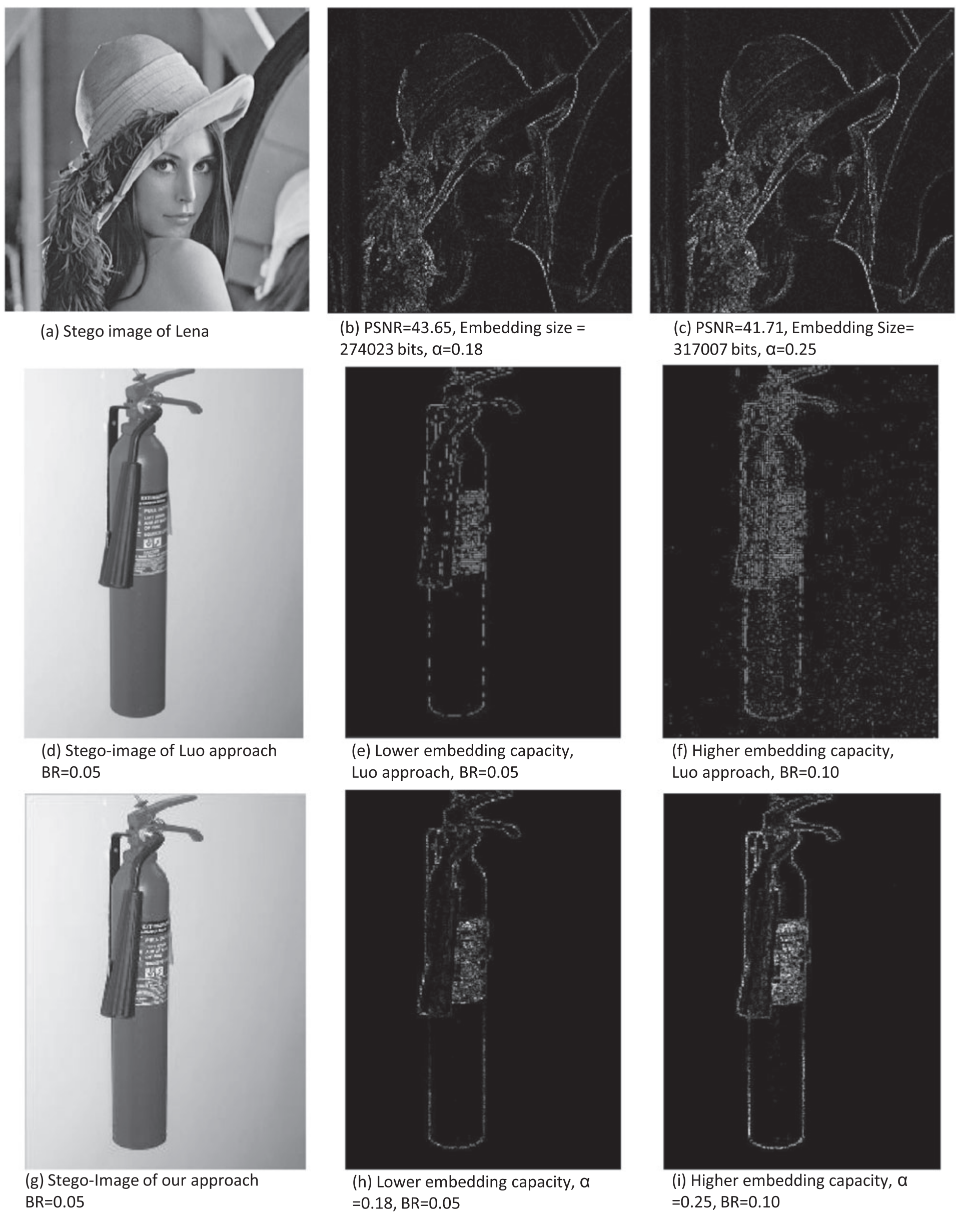

(a) Stego image of Lena
(b) PSNR=43.65, Embedding size = 274023 bits, α=0.18
(c) PSNR=41.71, Embedding Size= 317007 bits, α=0.25
(d) Stego-image of Luo approach BR=0.05
(e) Lower embedding capacity, Luo approach, BR=0.05
(f) Higher embedding capacity, Luo approach, BR=0.10
(g) Stego-Image of our approach BR=0.05
(h) Lower embedding capacity, α =0.18, BR=0.05
(i) Higher embedding capacity, α =0.25, BR=0.10

Figure 7. Comparison of the proposed approach with Luo approach [18]: (a) stego-image of Lena, (b) difference map of the proposed approach after embedding 274023 bits into Lena image, (c) difference map of the proposed approach after embedding 317007 bits, (d) stego-image of Luo embedding approach, (e) difference map of Luo's embedding approach with $BR = 0.05$ bpp, (f) difference map of Luo's embedding approach with $BR = 0.10$, (g) stego-image of the proposed approach, (h) difference map of the proposed approach with $BR = 0.05$ bpp, and (i) difference map of the proposed approach with $BR = 0.10$ bpp.

ysis and RS-attack. For histogram analysis, the following images are selected for inspecting their histograms before and after data embedding. The images are car, baboon, grass, F16, leather, lena, pentagon, and pepper. The histogram for each cover image is calculated by dividing the image into non-overlapping 3×3 blocks. The median pixel p_m is found, and the differences between each pixel in the block and the median pixel p_m are calculated. The resulting differences are collected into a vector of differences. The same procedure is applied on all the blocks in the cover image. The process is repeated for the stego-image. The corresponding histograms are depicted in Figure 8. Zhang and Wang [21] in their results showed that the basic PVD approach [14] has some visualized steps in the histogram of differences. The reader can refer to [19] for inspecting basic PVD difference histogram. The fixed quantization step in both PVD and PVD+LSB introduce histogram artifacts, which can be used to estimate the size

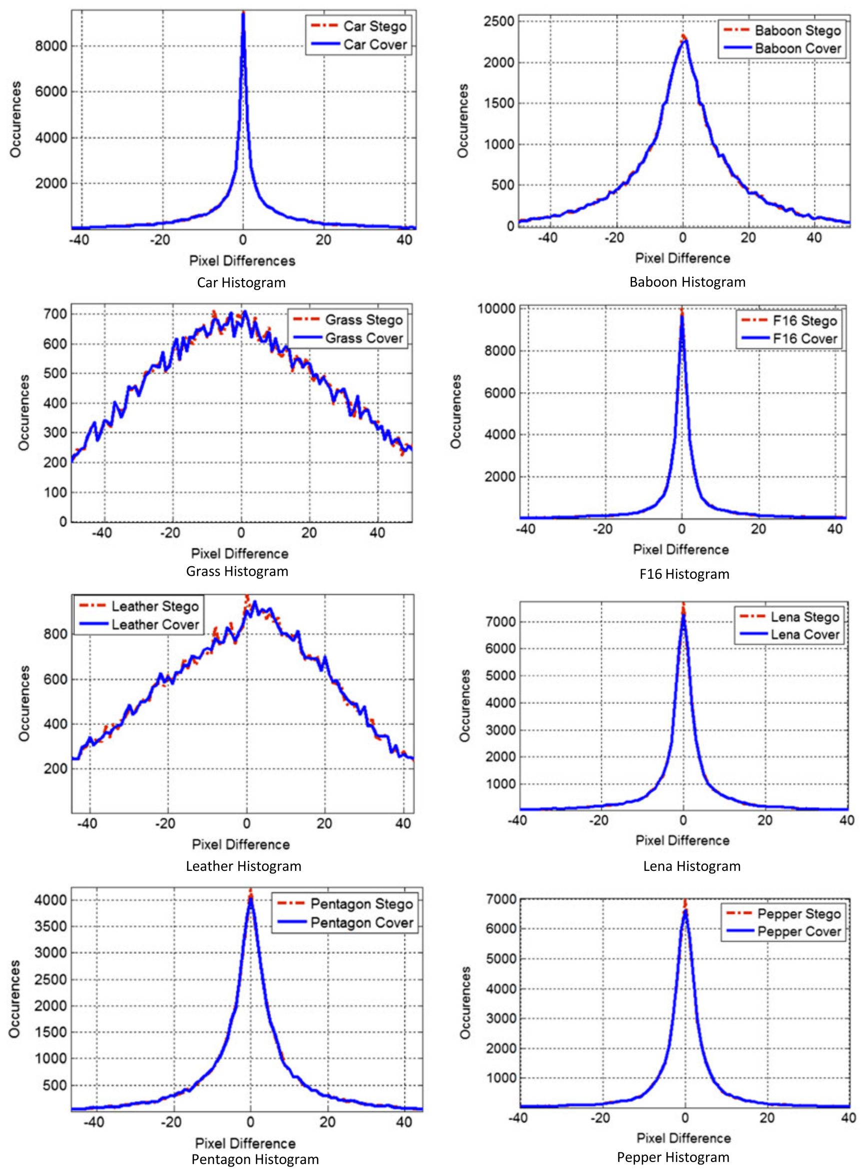


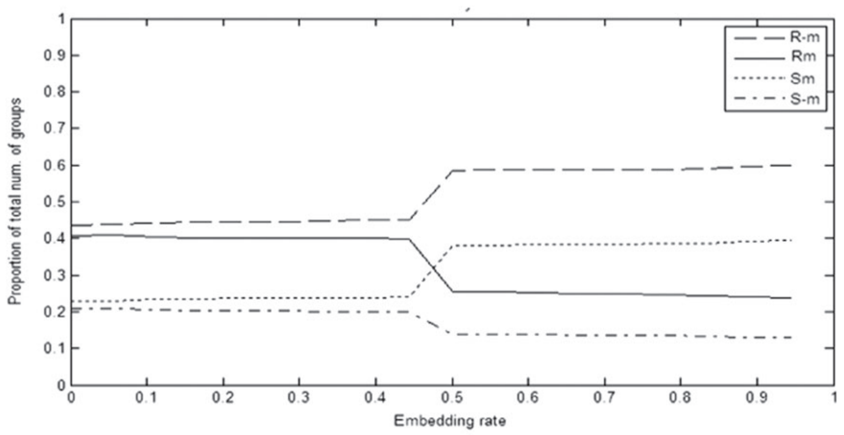
Figure 8. The resulting difference histograms for car, baboon, grass, F16, leather, lena, pentagon, and pepper images before and after data embedding, $BR = 0.50$ bpp.

of the embedded data especially of the embedding rate is very high.

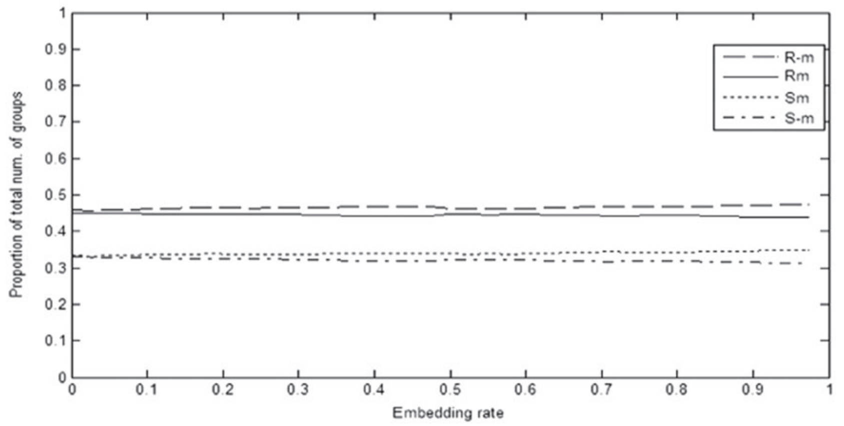
RS-attack proposed by Fridrich *et al*. [27] divides the image pixels into two groups, regular group and singular group. In natural images taken with digital camera, Fridrich found by experiment that

$$R_m \cong R_{-m} > S_m \cong S_{-m} \tag{20}$$

where R_m and S_m are respectively the proportions of regular and singular groups using the mask m =[0 1 1 0], R_{-m} and S_{-m} are respectively the proportions of regular and singular groups using the mask $-m$ = [0 – 1 – 10]. However, Equation 20 is violated with embedding secure data into the image. Especially, when embedding random data (encrypted or compressed data) in the LSB of the image pixels. When embedding more than 50% of data, the proportion of regular group R_m and the proportion of singular group S_m crosses each other as shown in Figure 9(a), which is an evidence of the existence of hidden data. Figure 9(b) shows the resistance of the proposed approach to RS-attack. The figure shows that R_m and R_{-m} are almost equal, and S_m and R = S_{-m} are almost equal. In addition, there is no intersection between R_m and S_m that comply with Equation 20 for natural clear images.



(a) RS-Attack applied on LSB based approaches



(b) RS-Attack applied on the proposed approach

Figure 9. Resistance of the proposed approach to RS-attack: (a) RS-attack applied on least significant bit-based approach, RS-attack successfully detects the existence of hidden data because R_m and S_m crosses each other. (b) RS-attack applied on the proposed approach, it is clear that $R_m \cong R_{-m} > S_m \cong S_{-m}$, which occurs in clear images without embedded data. This proves that the proposed approach embeds secure data while keeping the statistical properties of the image.

5. CONCLUSION

An adaptive PVD steganography technique has been proposed. The embedding process is directed to the content edges and regions with intensity fluctuations. In PVD-based embedding, we proved by experiment that selecting median pixel as pivot pixel is better than selecting minimum, maximum, or central pixels. The algorithm introduced higher embedding quality specially in low-embedding BR. The embedding spreads vertically in the pixels and horizontally in regions with abrupt changes. The proposed approach is robust to histogram analysis and RS-attacks. The algorithm outperformed the state of the art pixel value differencing schemes.

REFERENCES

1. Simmons G. The prisoners problem and the subliminal channel. *CRYPTO* 1983: 51–67.
2. Anderson RJ, Petitcolas FAP. On the limits of steganography. *IEEE Journal on Selected Areas in Communications* 1998; **16**(4): 474–481.
3. Farhat F, Ghaemmaghami S. Towards blind detection of low-rate spatial embedding in image steganalysis. *Image Processing, IET Year:* 2015; **9**(1): 31–42.
4. Kirchner M, Gloe T. On resampling detection in re-compressed images. *In Proceeding of the First IEEE International Workshop on Information Forensics and Security*, London UK, 2009; 21–25.
5. Osama H. Side-informed image watermarking scheme based on dither modulation in the frequency domain. *The Open Signal Processing Journal* 2013; **5**(1): 1–6.
6. Li X, Wang J. A steganographic method based upon JPEG and particle swarm optimization algorithm. *Information Sciences* 2007; **177**(15): 3099–31091.
7. Khodaei M, Faez K. New adaptive steganographic method using least significant-bit substitution and

pixel-value differencing. *Image Processing, IET Year:* 2012; **6**(6): 677–686.

8. Sutaone MS, Khandare MV. Image based steganographyusing LSB insertion technique. *Wireless, Mobile and Multimedia Networks, 2008. IET International Conference on*, IET, 2008; 146–151.
9. Hosam O, Malki Z. Steganography technique for embedding secure data into the image regions with abrupt changes. *Life Science Journal* 2014; **11**(9): 126–130.
10. Zhang X. Efficient data hiding with plus-minus one or two. *IEEE Signal Processing Letters* 2010; **17**(7): 635–638.
11. Mankun X, Tianyun L, Xijian P. Steganalysis of LSB Matching Based on Histogram Features in Grayscale Image. ICCT 11th, 2008.
12. Ashwin S, Ramesh J, Kumar S, Gunavathi K. Novel and secure encoding and hiding techniques using image steganography: a survey. *ICETEEEM'12*, 2012; 171–177.
13. Ghazanfari K, Ghaemmaghami S, Khosravi SR. LSB++: an improvement to LSB+ steganography. *TENCON'11 Conference*, Indonesia, 2011; 364–368.
14. Wu DC, Tsai WH. A steganographic method for images by pixel-value differencing. *Pattern Recognition Letters* 2003; **24**: 1613–1626.
15. Wu HC, Wu NI, Tsai CS, Hwang MS. Image steganographic scheme based on pixel-value differencing and LSB replacement methods. *Proceeding of IEEE Institute of Electrical and Electronics Engineers vis.* Images signal process 2005; **152**(5): 611–615.
16. Yang CH, Wang SJ, Weng CY. Analyses of pixel-value-differencing schemes with LSB replacement in stegonagraphy. *Proceedings of the 3rd International Conference Intelligent Information Hiding and Multimedia Signal Processing*, Taiwan, 2007; 445–448.
17. Yang CH, Weng CY, Wang SJ, Sun HM. Adaptive data hiding in edge areas of images with spatial LSB domain systems. *IEEE Trans. on Information Forensics and Security* 2008; **3**(3): 488–497.
18. Luo W, Huang F. A more secure steganography based on adaptive pixel-value differencing scheme. In *Network Security and Cryptography*. Springer, 2010.
19. Gandharba S. Adaptive pixel value differencing steganography using both vertical and horizontal edges. *Multimed Tools Appl* 2015, DOI:10.1007/s11042-015-2937-2.
20. Hempstalk K. Hiding behind corners: using edges in images for better steganography. *Proceedings of Computing Women's Congress*, Hamilton, New Zealand, 2006.
21. Zhang X, Wang S. Vulnerability of pixel-value differencing steganography to histogram analysis and modification for enhanced security. *Pattern Recognition Letters* 2004; **25**: 331–339.
22. *The USC-SIPI image database*, 2015. USC University of South California, Signal and Image processing Institute. Online http://sipi.usc.edu/database.
23. Schaefer G, Stich M. UCID – an uncompressed colour image database. *Technical Report*, School of Computing and Mathematics, Nottingham Trent University, U.K, 2003.
24. Voloshynovskiy S *et al.* Generalized watermarking attack based on watermark estimation and perceptual remodulation. *in Proceeding of Electronic Imaging. International Society for Optics and Photonics*, 2000; 358-370.
25. Wang Z, Bovik AC. A universal image quality index. *IEEE Signal Processing Letters* 2002; **9**(3): 81–84.
26. Tsai YY, Chen JT, Chan CS. Exploring LSB substitution and pixel-value differencing for block-based adaptive data hiding. *IJ Network Security* 2014: 363–368.
27. Fridrich J, Goljan M, Du R. Reliable detection of LSB stegnography in grayscale and color images. *Proceedings of the ACM Workshop on Multimedia and Security*, Ottawa, CANADA, 2001; 2730.