Improving Capacity of (5,3) Hamming Code Scheme in Steganography Using Pixel Value Differencing

Shashidhar G Koolagudi*, Karthik M†, Aswin Manoj P‡, Pushpita Patil§ and C Aishwarya¶
Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Mangalore: 575025

*koolagudi@nitk.edu.in, †mkarthik2597@gmail.com, ‡aswin6197@gmail.com
§patilpushpita@gmail.com, ¶haygriaishu02@gmail.com

***Abstract*—In this paper, we propose a method to improve the capacity of the (5,3) Hamming Code steganographic scheme without significantly affecting the imperceptibility of the stego-image using techniques of Pixel Value Differencing and Optimal Pixel Adjustment Process.**

The (5,3) Hamming code scheme embeds 3 bits for every 5 pixels of the cover-image. While the method ensures that very less distortion is introduced into the cover-image, it suffers from its inability to handle a large payload. Pixel Value Differencing (PVD) is a technique designed to hide different amounts of data in different regions of the image contrary to simple LSB substitution which hides 1 bit in every pixel of the image. PVD partitions the cover-image into non-overlappable pixel blocks of two consecutive pixels and calculates a difference value from the values of the two pixels to cluster the blocks into a number of categories of smoothness and contrast properties. PVD tries to leverage human vision's sensitivity to gray value variations in the image by hiding more amount of data in the regions of high contrast and less amount of data in the smooth regions of the image.

Following the guiding principle of PVD, we try to enhance the capacity of (5,3) Hamming scheme by hiding more data in the edged regions or the high contrast regions and less data in the smooth regions of the cover-image. We also use the Optimal Pixel Adjustment Process (OPAP) to reduce the error between the cover-image and the stego-image. Moreover, data embedding is done by selecting pixel blocks in a pseudo-random mechanism to achieve secrecy protection and prevent tampering from illicit users.

1. Introduction

Steganography is a technique of encoding secret information that takes cryptography a step further by hiding an encrypted message in a cover medium so that no one suspects it exists. Ideally, anyone scanning the cover medium will fail to know that it contains encrypted data. Computer-image based steganography uses digital images as cover-media to embed and deliver secret messages. In this paper, 8-bit grayscale images are selected as cover-images. cover-images with the secret data embedded in them are called stego-images.

A good steganographic scheme should have a high embedding payload (or capacity) and high embedding efficiency (or least modifications to the cover medium). Information hiding schemes with high embedding efficiency are hard to be detected by steganalytical detectors. However, such schemes have a low embedding payload. On the contrary, information hiding schemes with high embedding payload sacrifice embedding efficiency and are weak against visual and statistical attacks. Thus, a trade-off between high embedding efficiency and high embedding payload is made by the users, depending upon their different desires.

Image files provide a limited steganographic capacity and in many cases, an embedded message does not require the full capacity. Therefore, a part of the image file remains unused and with continuous embedding, the secret data tends to concentrate at the start of the file and the unused rest resides at the end.

Ron Crandall [1] suggested the matrix encoding scheme for low rate steganography in images where the amount of covert information is below one bit per pixel. If most of the capacity is unused in a stego-image, matrix encoding decreases the number of necessary changes and equalises the embedding rate in the stego-image, thereby improving the embedding efficiency. Westfield is probably the first one who implemented the matrix encoding scheme by proposing a steganographic scheme known as F5 [2]. Then, Zhang et al. proposed the "Hamming+1" scheme [3] to improve the embedding efficiency and embedding payload. Kim et al. proposed the (5,3) Hamming code scheme [4] which has a better embedding rate and image quality compared to "Hamming+1" scheme.

Hiding data in the LSBs of the pixels of an image is a common information hiding method that utilises the characteristic of human vision's insensitivity to small changes in the image. This technique is computationally easy and a large amount of data can be embedded without great quality loss. However, this simple method is vulnerable to attackers who may gather information by concentrating on the statistics of the LSBs. Also, with a high embedding rate, the LSB substitution method can give rise to unwanted artifacts in the image which may arouse suspicion and defeat the purpose of steganography.

Chan et al. [5] proposed the Optimal Pixel Adjustment Process (OPAP) as a mechanism to greatly improve the quality of the stego-image with low extra computational overhead. The idea of OPAP comes from the work by Wang et al. [6] on moderately significant bit (MSB) substitution which uses local pixel adjustment process to reduce image degradation. Wang et al. [7] further proposed a data hiding scheme by optimal LSB substitution (OLSB) and genetic algorithm. However, Wang's OLSB method requires huge computation for the genetic algorithm to find an optimal substitution matrix. The results of OPAP by Chan et al. confirm that their proposed method has much better performance than OLSB with respect to image quality as well as computational complexity.

Wu et al. [8] proposed an efficient, novel steganographic method called Pixel Value Differencing (PVD) to hide data in gray-valued images imperceptibly. The problem with simple LSB substitution is that it changes all pixel values by more or less equal amounts. But not all pixels in an image can tolerate equal amounts of changes without causing notice to an observer. The largest number of LSBs whose gray values can be changed without producing a perceptible artifact in each pixel is different. PVD overcomes this issue by hiding different amounts of data in different pixels based on its location in the image. More data is embedded in a pixel located in an edged area of an image than a pixel in the smooth area.

The rest of the paper is organised as follows. Section 2 describes the (5,3) Hamming Code scheme, the PVD and the OPAP scheme in some detail. In Section 3, we discuss the encoding and the decoding procedures for improving the capacity of the (5,3) Hamming Code scheme using PVD and OPAP. Experiment results are given in Section 4, and Section 5 concludes the paper.

2. Related Works

2.1. The (5,3) Hamming Code Scheme

The Hamming (5,3) steganographic technique [4] on grayscale images conceals 3 bits per 5 pixels of a grayscale image. The cover-image is divided into consecutive blocks of 5 pixels each. The LSBs of the 5 pixels are extracted and multiplied with a parity check matrix (see (1) below) to obtain a 3-bit vector called as a syndrome. This vector is then XORed bit-wise with 3 bits taken as a sub-stream of the secret message. With this 3-bit vector as a new syndrome, a corresponding 5-bit error vector is obtained by referring the error-syndrome table. The 5-bit vector error is then XORed bitwise with the original LSB vector to get a new 5-bit vector which is substituted back in the 5-pixel block of the cover-image to be transmitted. The encoded message is extracted by simply multiplying the transmitted 5-bit LSB vector with the parity check matrix. This technique has an embedding rate close to 0.6 and is highly steganographic since at most 2 LSBs change for every 5 pixels of the image. However, the capacity of the algorithm is very poor, even when compared to 1-bit LSB substitution in that only 3 bits are encoded for every 5 pixels of the image.

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

Error-Syndrome Table

Order	Error Vector	Syndrome
0	[0 0 0 0 0]	[0 0 0]
1	[0 0 0 0 1]	[0 0 1]
2	[0 0 0 1 0]	[0 1 0]
3	[0 0 1 0 0]	[0 1 1]
4	[0 1 0 0 0]	[1 0 0]
5	[1 0 0 0 0]	[1 0 1]
6	[1 1 0 0 0]	[1 1 0]
7	[0 1 1 0 0]	[1 1 1]

2.2. Pixel Value Differencing

PVD [8] is another steganographic method for embedding secret messages into gray-valued cover-images. The cover-image is divided into a number of non-overlapping two-pixel blocks. Each block is categorised according to the difference of the gray values of the two pixels. A large difference value indicates that the pixel block is in an edged area and small value indicates that the pixel block is in a smooth area. More data is embedded in edged areas than in smooth areas. Because the pixels in edged areas can tolerate larger changes of pixel values than those in the smooth areas, the changes in the resulting stego-image are unnoticeable.

All the possible difference values of the pixel blocks are classified into a number of ranges. The selection of the range intervals is based on the characteristics of human vision's sensitivity to gray value variations. One such set of range widths used is 8, 8, 16, 32, 64 and 128. These range widths partition the total range of possible pixel value difference [0,255] into [0,7], [8,15], [16,31], [32,63], [64,127], [128,255]. The number of bits which can be embedded in a pixel pair is decided by the particular range interval that the difference value belongs to. The method is designed in such a way that after modification of a pixel, the new difference value is never out of its range interval. PVD thus provides an easy way to produce a more imperceptible result than those

yielded by simple LSB replacement methods. Moreover, the embedded secret message can be extracted from the resulting stego-image without referencing the original cover-image.

2.3. Optimal Pixel Adjustment Process

The Optimal Pixel Adjustment Process (OPAP) [5] is a simple and computationally inexpensive technique to reduce the error between the stego-image and the cover-image resulting from LSB substitution. All the possible error values between the original and the new pixel values are segmented into three intervals. If δ is the error value and k bits are substituted in every pixel using simple LSB substitution, then the three intervals are:

Interval 1: $2^{k-1} < \delta < 2^k$
Interval 2: $-2^{k-1} \leq \delta \leq 2^{k-1}$
Interval 3: $-2^k < \delta < -2^{k-1}$

Based on the interval where the error value belongs, OPAP modifies the stego-image pixel value such that in most of the cases, the maximum possible absolute error value reduces from 2^k to 2^{k-1}. The modification of the pixels is such that the k encoded LSBs in any pixel remain intact and the changes occur only in the higher order bits. This ensures that decoding of the hidden message does not require referencing the original image.

3. Proposed Algorithm

To improve the capacity of the (5,3) Hamming scheme, we follow the essential technique of PVD, i.e, we partition the cover-image into numerous blocks of 2 consecutive pixels and calculate their difference value. Based on the particular range interval that the pixel difference belongs to, we decide how many LSBs from each of the pixels can be chosen for data hiding. Naturally, the greater the pixel difference, the more number of LSBs are chosen for data encoding. The encoding algorithm used is the (5,3) Hamming scheme. We ensure that in most of the cases, the new difference value of the pixel pair is within the same range as it was before encoding and thus, the changes in the resulting stego-image are kept unnoticeable. The decoding algorithm is the same as the one used in the (5,3) Hamming scheme.

The algorithm uses 256 gray-valued images. The cover-image is partitioned into two-pixel blocks by traversing the image in a zig-zag fashion. The pixel blocks are then reordered according to a sequence generated by a pseudo random number generator. The data embedding in the pixel blocks is then done sequentially according to this order. After data embedding, the pixel blocks are replaced in the image after rearranging them back according to their original zig-zag sequence.

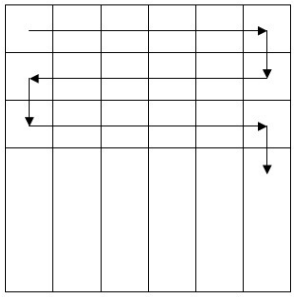


Figure 1: Zigzag Scanning Of Image Rows

Let p_i and p_{i+1} be two consecutive pixels of a pixel block and their gray values be g_i and g_{i+1}. The pixel value difference between them is calculated as $d = |\, g_{i+1} - g_i \,|$. A block with d close to 0 is considered to be an extremely smooth block, whereas a block with d close to 255 is considered as a sharply edged block. All the possible difference values are classified into a number of contiguous ranges, say R_i where $i = 1, 2, \ldots, n$. The lower and upper bound values of R_i are denoted by l_i and u_i respectively, where l_1 is 0 and u_n is 255. The width of the range interval R_i is $u_i - l_i + 1$.

As mentioned previously, one such set of range intervals is [0,7], [8,15], [16,31], [32,63], [64,127] and [128,255]. The selected range intervals are based on the human visual capability. The widths of the ranges which represent difference values of smooth blocks are chosen to be smaller while those which represent the difference values of edged blocks are chosen to be larger. If a difference value is replaced by another in the same range, the change presumably cannot be easily noticed by human eyes. Therefore, in the proposed algorithm, we ensure that in most of the cases, the new pixel value difference obtained after data embedding lies in the same range as it was before data embedding.

The following sections describe the data encoding and the decoding procedure.

3.1. Encoding Procedure

We consider the secret message as a long bit stream. We want to embed every bit in the bit stream into the two-pixel blocks of the cover-image. The number of bits which can be embedded in each block varies and is decided by the width of the range to which the difference value of the two pixels in the block belongs. The algorithm is precisely explained below:

- Scan the cover-image in a zig-zag fashion, taking two pixels at a time to form a two-pixel block and add them to a list PIXEL BLOCK LIST
- Randomise the elements of PIXEL BLOCK LIST using a pseudo-random number generator.
- Traverse the PIXEL BLOCK LIST sequentially one block at a time. If P is a pixel block with the pixels

p_i and p_{i+1} and intensities g_i and g_{i+1}, calculate the pixel value difference as $d = \mid g_{i+1} - g_i \mid$. Add this difference value d to list PIXEL DIFFERENCE LIST. PIXEL DIFFERENCE LIST thus consists of a list of the pixel value differences of the pixel blocks in the image.

- Traverse through PIXEL DIFFERENCE LIST sequentially and for each element of the list, find the appropriate range interval R_k to which the difference belongs to. Let l_k and u_k be the lower and the upper limits of R_k
- Let $X = min\{d - l_k, u_k - d\}$ and $m = \lfloor log_2 X \rfloor$. If $m > 1$, select $m - 1$ bits from both the pixels p_1 and p_2, else select 1 bit from each of the two pixels for encoding your secret data. Add these bits to a list called LSB LIST
- The (5,3) Hamming Code algorithm requires that the number of bits selected for encoding must be a multiple of 5. So, the number of encode-able bits is equal to that multiple of 5 closest to the length of LSB LIST. We select these bits and copy it to another list called ENCODABLE BITS LIST
- The ENCODABLE BITS LIST is divided into units of 5 bits each. The message to be encoded is converted to binary format and 0s are added to make the number of bits a multiple of 3. The message is then divided into units of 3 bits each. If the number of units of 5 bit LSBs is greater than the number of units of 3-bit messages, sufficient zeroes are added to make the number of units equal. On the other hand, if the number of units of 3-bit messages is more than the number of units of 5 bit LSBs, then an error is generated.
- We traverse the message and the ENCODABLE BITS LIST and encode each 3-bit message unit into a 5-bit LSB unit using the (5,3) Hamming Code algorithm.
- All the units of the ENCODABLE BITS LIST are merged. This list now contains the encoded message. The bits left out from the LSB LIST (due to non-divisibility with 5) are appended to the ENCODABLE BITS LIST.
- Based on the pixel difference values in the PIXEL DIFFERENCE LIST, recalculate m, the number of encodable bits in the corresponding pixel block of the PIXEL BLOCK LIST. Replace m LSBs from each of the pixels in the pixel block with $2m$ bits from the ENCODABLE BITS LIST.
- Finally, replace the pixel blocks in the PIXEL BLOCK LIST back into the cover-image to generate the stego-image using the same pseudo-random sequence generated above.

Appendix A proves that if $m > 1$, by altering $m - 1$ bits, the new pixel difference values stay in the same range as before and if $m < 1$, the new pixel difference values can jump out of the bounds by at most 2 units. And hence, we show that the changes in the pixel values are imperceptible.

3.2. Decoding Procedure

The number of bits that can be encoded in a pixel block depends upon the closeness of the pixel difference value to the boundaries of the corresponding range width. The encoding procedure alters this difference value and in most of the cases stays in the same interval. Since the difference value has been altered, the distance of the new difference value to the boundaries of the interval also changes. Thus, we need to refer the original cover-image to figure out how many bits have been encoded into each pixel block.

Thus, after the data embedding phase has been done, the sender sends the cover-image, the stego-image and the seed used for the pseudo-random number generation across a secure channel.

- Similar to the encoding procedure, scan the cover-image in a zig-zag fashion and generate PIXEL BLOCK LIST using the same pseudo-random sequence used for encoding. From this list, generate the PIXEL DIFFERENCE LIST.
- Using the PIXEL DIFFERENCE LIST of the cover-image, obtain the LSB LIST and the ENCODABLE BITS LIST from the stego-image.
- Divide the ENCODABLE BITS LIST into units of 5-bits each. Treating each unit as a column vector, multiply it with the (5,3) Hamming parity check matrix to recover the message.

4. Experimental Results

We compare the performance of our proposed algorithm (we call it Hamming PVD for convenience) with (5,3) Hamming code using different metrics namely PSNR, Embedding Rate, Change Density and Capacity. The metrics are defined below:

PSNR: Measured in dB, the Peak Signal to Noise Ratio is a standard metric used to measure the extent of similarity between the cover-image and the stego-image. It is defined as

$$PSNR = 10 \log_{10} \left(\frac{I_{max}^2}{MSE} \right)$$

where I_{max} is the maximum possible intensity or pixel value of the image. Since we have used 8-bit images, I_{max} is 255. MSE stands for Mean Square Error and is defined as

$$MSE = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\mid I_{i,j} - I_{i,j}^{'} \mid \right)^2$$

where M is the width of the image and N is the height of the image in pixels. $I_{i,j}$ and $I_{i,j}^{'}$ are the $(i,j)^{th}$ pixel values of the cover-image and the stego-image respectively.

High PSNR means the stego-image is similar to the cover-image and low PSNR means no resemblance between them. In general, PSNR over 30dB means that stego-images cannot be detected visually.

Embedding Rate: It is the average number of bits of the message hidden in each pixel. It is calculated as

$$\text{Embedding Rate} = \frac{\text{length of message in bits}}{MN}$$

Change Density: Expressed as a percentage, it is the ratio of the number of altered pixels in the stego-image to the total number of pixels in the cover-image.

$$\text{Change Density} = \frac{\text{Total number of altered pixels}}{MN} * 100$$

Capacity: Capacity is the total number of bits which can be hidden in a cover-image.

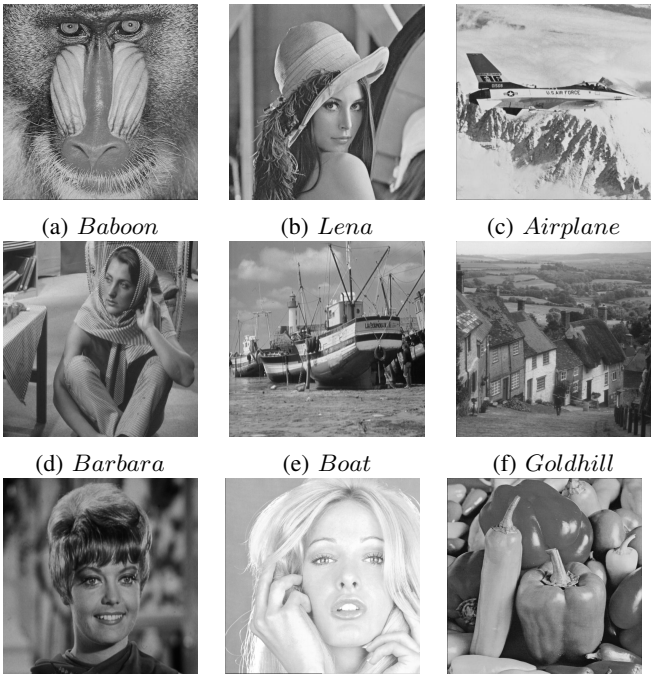
We perform two sets of experiments to evaluate the performance of Hamming PVD. We use two different sets of range widths for testing as used in [8]. The first set of range widths was described previously [8, 8, 16, 32, 64, 128]. The second set of range widths is [2, 2, 4, 4, 4, 8, 8, 16, 16, 32, 32, 64, 64]. On each of the images, the same message (stored in a text file) is used and the results of the (5,3) Hamming code and Hamming PVD are recorded.

In the first experiment, we use some of the standard images used in image processing for various types of testing. (see Figure 2). The test images include smooth content images (e.g. Zelda, Tiffany) and complex content images (e.g. Baboon).
Tables 1 to 9 show the results obtained by testing the (5,3) Hamming Code and the Hamming PVD algorithms on corresponding images.

For the second testing a set of 500 realistic background images provided by [9] is used. Each algorithm is run over the entire text file and the average values for each of the metrics is calculated.
Table 10 contains the results of the testing done on a set of random images provided by [9].

TABLE 1: Baboon

	Hamming	**Hamming PVD**	
		Range 1	Range 2
PSNR	60.466	59.36	60.35
Embedding Rate	0.6	0.636	0.603
Change density	22.45	23.52	22.57
Capacity	150000	158970	150984
Capacity Change		8970	984



(a) $Baboon$ (b) $Lena$ (c) $Airplane$

(d) $Barbara$ (e) $Boat$ (f) $Goldhill$

(g) $Zelda$ (h) $Tiffany$ (i) $Peppers$

Figure 2: Testing Images

TABLE 2: Lena

	Hamming	**Hamming PVD**	
		Range 1	Range 2
PSNR	60.49	60.169	60.434
Embedding Rate	0.6	0.608	0.601
Change density	22.31	22.71	22.48
Capacity	150000	152163	150273
Capacity Change		2163	273

TABLE 3: Airplane

	Hamming	**Hamming PVD**	
		Range 1	Range 2
PSNR	60.464	60.04	60.40
Embedding Rate	0.6	0.612	0.602
Change density	22.47	22.83	22.51
Capacity	150000	153030	150543
Capacity Change		3030	543

TABLE 4: Barbara

	Hamming	**Hamming PVD**	
		Range 1	Range 2
PSNR	60.466	58.54	60.14
Embedding Rate	0.6	0.542	0.611
Change density	22.45	24.00	22.83
Capacity	150000	163563	152787
Capacity Change		13563	2787

TABLE 5: Boat

	Hamming	**Hamming PVD**	
		Range 1	Range 2
PSNR	60.465	59.998	60.414
Embedding Rate	0.6	0.613	0.601
Change density	22.4592	22.882	22.504
Capacity	150000	153288	150447
Capacity Change		3288	447

TABLE 6: Goldhill

	Hamming	**Hamming PVD**	
		Range 1	Range 2
PSNR	60.50	60.288	60.464
Embedding Rate	0.6	0.604	0.600
Change density	22.267	22.679	22.442
Capacity	150000	151200	150060
Capacity Change		1200	60

TABLE 7: Zelda

	Hamming	**Hamming PVD**	
		Range 1	Range 2
PSNR	60.465	60.442	60.450
Embedding Rate	0.6	0.601	0.6
Change density	22.459	22.532	22.437
Capacity	150000	150315	150000
Capacity Change		315	0

TABLE 8: Tiffany

	Hamming	**Hamming PVD**	
		Range 1	Range 2
PSNR	60.456	60.201	60.436
Embedding Rate	0.6	0.606	0.6008
Change density	22.506	22.701	22.495
Capacity	150000	151551	150216
Capacity Change		1551	216

TABLE 9: Peppers

	Hamming	**Hamming PVD**	
		Range 1	Range 2
PSNR	60.46	59.877	60.342
Embedding Rate	0.60	0.609	0.602
Change density	22.49	22.79	22.53
Capacity	150000	152355	150678
Capacity Change		2355	678

TABLE 10: Random Image Results

	Hamming	**Hamming PVD**	
		Range 1	Range 2
PSNR	60.831	59.866	60.402
Embedding Rate	0.6	0.615	0.604
Change density	20.77	22.521	22.216
Capacity	150000	153808	150958
Capacity Change		3808	958

The rationale for the results tabulated above are as follows:

Capacity The capacity increase is more for the first set of range intervals compared to the second. The explanation is as follows: The increase in capacity of Hamming PVD compared to (5,3) Hamming is primarily because of selection of more than 1 bit from a pixel for the encoding procedure by Hamming PVD whereas (5,3) Hamming selects exactly 1 bit from every pixel. The number of bits that Hamming PVD chooses for encoding depends upon whether the pixel block, of which it is a part, lies in a smooth region or an edged region of the image. If the pixel block lies in an edged region, then the pixel difference value of the block is high and consequently the range width to which the pixel block belongs to, is also high. Precisely, the number of additional bits that can be chosen for encoding depends upon the closeness of the actual pixel difference value to the boundaries of the range width interval. The larger the range width interval, more is the probability of higher selection of bits for encoding.

The first set of range intervals have higher values of range widths compared to the second set of range widths. The range widths in the second set are narrow and hence Hamming PVD ends up choosing 1 bit from every pixel most of the times, giving a capacity increase barely more than what the (5,3) Hamming algorithm gives.

The extent of capacity increase for an image depends upon the number of smooth and edged regions in the image. Fig. 2a, the Baboon has a lot of edges and hence the potential for increase in the capacity is high. The capacity increase for Fig. 2b, Lena and for Fig. 2g, Zelda is very less since they are smooth images.

PSNR The PSNR for Hamming PVD in both the sets of range widths is lesser than that of (5,3) Hamming is because of the higher embedding payload of Hamming PVD. The PSNR for second set of range widths is higher than the first set for the same reason.

Embedding Rate Since the dimensions of the image are the same, by the definition of embedding rate, an increase in capacity should correspond to a proportional increase in the embedding rate.

Change Density The probability of a pixel value changing depends upon how many LSBs it contributes for data embedding. If it contributes only one LSB, then there is a 50% chance that the pixel value will change. If it contributes 2 LSBs, there is a 75% chance and so on. A higher capacity points to usage of more LSBs for data embedding and thereby higher change density.

The results for the random images show an increase in the capacity of (5,3) Hamming Code algorithm by about 3800 bits for the first set of range widths and an increase of about 960 bits for the second set of range widths. As explained previously, the disparity here is due to closeness of the interval boundaries in the second set of range widths compared to the first set.

5. Conclusion and Future Work

In this paper, a method was proposed to improve the capacity of the (5,3) Hamming Code steganographic scheme without significantly affecting the imperceptibility of the stego-image using Pixel Value Differencing and Optimal Pixel Adjustment Process.
The potential increase in the embedding payload of an image depends upon the number of smooth and sharp edged regions in the image. More the number of sharp edged regions in the image, more is the capacity increase. Also, the choice of the set of range intervals also affects the capacity increase. A set whose range intervals with narrow range widths will give a very less capacity increase compared to a set of range intervals which have broad range widths.

A drawback of the proposed method is that extraction of the encoded message from the stego-image requires reference to the original cover-image. A workaround to overcome this demerit will be beneficial.

References

[1] R. Crandall. Some Notes on Steganography. Posted on steganography mailing list, http://os.inf.tu-dresden.de/ westfeld/crandall.pdf, 1998.

[2] A. Westfeld. F5-a steganographic algorithm. In Proceedings of the 4th International Workshop on Information Hiding, pages 289âĂŞ302. Pittsburgh, PA, USA, April 2001.

[3] W. Zhang, S.Wang, and X. Zhang. Improving embedding efficiency of covering codes for applications in steganography. IEEE Communications Letters, 11(8):680âĂŞ682, August 2007.

[4] Kim C., Yang CN. (2015) Steganography Based on Grayscale Images Using (5, 3) Hamming Code. In: Shi YQ., Kim H., PÃl'rez-GonzÃąlez F., Yang CN. (eds) Digital-Forensics and Watermarking. IWDW 2014. Lecture Notes in Computer Science, vol 9023. Springer, Cham

[5] Chi-Kwong Chan, L.M. Cheng. Hiding data in images by simple LSB substitution. In Pattern Recognition, Volume 37, Issue 3, March 2004, Pages 469-474

[6] Ran-Zan Wang, Chi-Fang Lin, Ja-Chen Lin, Hiding data in images by optimal moderately signi7cant-bit replacement, IEE Electron. Lett. 36 (25) (2000) 2069âĂŞ2070.

[7] Ran-Zan Wang, Chi-Fang Lin, Ja-Chen Lin, Image hiding by optimal LSB substitution and genetic algorithm, Pattern Recognition 34 (3) (2001) 671âĂŞ683.

[8] Wu D.-C., Tsai W.-H. A steganographic method for images by pixel-value differencing. Pattern Recognition. Lett., 24 (2003), pp. 1613-1626

[9] L. Fei-Fei, R. Fergus and P. Perona. Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. IEEE. CVPR 2004, Workshop on Generative-Model Based Vision. 2004. Link: www.vision.caltech.edu/Image_Datasets/Caltech101

Appendix A

Let us suppose that we change z LSBs from both p_i and p_{i+1} of the pixel block P.

Let a and A be the decimal values of the $(8-z)$ MSBs of p_i and p_{i+1} respectively. Similarly, let b and B be the decimal values of the z LSBs of p_i and p_{i+1} respectively. Thus, we have:

$$g_i = a * 2^z + b$$
$$g_{i+1} = A * 2^z + B$$

Therefore, the difference between these two pixel values is computed as:

$$d = |\, g_{i+1} - g_i \,| = |\, (A * 2^z + B) - (a * 2^z + b) \,|$$

After changing the z LSBs in both the pixels, let the new values corresponding to the z LSBs be b' and B' respectively. Thus, we now have new pixel intensity values g_i^* and g_{i+1}^* given by:

$$g_i^* = a * 2^z + b'$$
$$g_{i+1}^* = A * 2^z + B'$$

The new difference d' is given as:

$$d' = |\, g_{i+1}^* - g_i^* \,| = |\, (A * 2^z + B') - (a * 2^z + b') \,|$$

We now wish to determine the maximum possible change in the difference when z LSBs are altered.

From the law of triangular inequality, we know that for every real quantities a and b,

$$|\, a \,| - |\, b \,| \leq |\, a - b \,|$$

$$\Rightarrow \quad |\, b \,| - |\, a \,| \leq (\, |\, b - a \,| = |\, a - b \,|\,)$$
$$\Rightarrow \quad -(\, |\, b \,| - |\, a \,|\,) \geq -|\, a - b \,|$$
$$\Rightarrow \quad |\, a \,| - |\, b \,| \geq -|\, a - b \,|$$

The first equation together with this equation gives

$$
\begin{aligned}
& -|a-b| \le |a| - |b| \le |a-b| \\
\Rightarrow\ & ||a| - |b|| \le |a-b|
\end{aligned}
$$

The above result helps us to arrive at the following inequality:

$$
\begin{aligned}
& |d' - d| \\
= \ & ||g^*_{i+1} - g^*_i| - |g_{i+1} - g_i|| \\
\le \ & |(g^*_{i+1} - g^*_i) - (g_{i+1} - g_i)| \\
= \ & |((A * 2_z + B') - (a * 2^z + b')) \\
& -((A * 2^z + B) - (a * 2^z + b))| \\
= \ & |(B' - B) - (b' - b)| \\
\le \ & |(2^z - 1) + (2^z - 1)| \\
= \ & |2^{z+1} - 2| \\
< \ & |2^{z+1}| \\
= \ & 2^{z+1}
\end{aligned}
$$

Thus, a change of z LSBs can change the original pixel difference by a value of less than 2^{z+1}

If we choose $z = m - 1$, a change in $m - 1$ LSBs can change the original pixel difference by a value of less than 2^m

$$
\begin{aligned}
& |d - d'| < 2^m \\
\Rightarrow\ & d - 2^m < d' < d + 2^m
\end{aligned}
$$

Case 1: $m \ge 2$

As defined in the data embedding section,

$$X = min\{d - l_k, u_k - d\}$$
$$m = \lfloor log_2 X \rfloor$$

Since $m \ge 2$,

$$
\begin{aligned}
& \lfloor \log_2 X \rfloor \ge 2 \\
\Rightarrow\ & \log_2 X \ge \lfloor \log_2 X \rfloor \ge 2 \\
\Rightarrow\ & X \ge 4
\end{aligned}
$$

We now show that the new difference value d' never goes out of bounds of the interval $R_k : [l_k, u_k]$

For all real y, we have:

$$
\begin{aligned}
& \lfloor y \rfloor \le y \\
\Rightarrow\ & \lfloor log_2 X \rfloor \le log_2 X \\
\Rightarrow\ & 2^{\lfloor log_2 X \rfloor} \le X \\
\Rightarrow\ & 2^m \le X
\end{aligned}
$$

Thus, we get

$$d - X < d' < d + X$$

If $X = d - l_k$, we get $l_k < d'$ or if $X = u_k - d$, we get $d' < u_k$
This proves that the new difference does not go out of bounds

Case 2: $m \le 1$

When $m \le 1$, we can have X equal to 3, 2, 1 or 0. In all these cases, we change only one bit for both the pixels in the pixel block.

From the previous equations, we get

$$
\begin{aligned}
|d - d'| &\le |2^{1+1} - 2| \\
&= |2| \\
&= 2
\end{aligned}
$$

Rearranging the equations, we get

$$d - 2 \le d' \le d + 2$$

In the worst case, if $d = l_k$, we get $d' = l_k - 2 = u_{k-1} - 1$ and if $d = u_k$, we get $d' = u_k + 2 = l_{k+1} + 1$.

This shows that every pixel can contribute atleast one LSB for data encoding without affecting the visual perceptibility of the cover-image.