Preserve Robustness for Image Data Hiding

D. B. SATRE AND R. V. PAWAR
Department of Computer Engineering
Vishwakarma Institute of Technology, Pune University
INDIA
dbsatre@yahoo.com, rvspawar@rediffmail.com

***Abstract:* -** In this paper, we have explained an algorithm for preserving robustness for image data hiding. Robust data hiding is the process of extraction the correct data after compression or any other alteration applied on the embedded image. Embedded image is the image that obtains after hiding some information on the original image. Our technique compare with the existing technique called Y. Q. Shi method. We are comparing results of both techniques, i.e., PSNR of the embedded image and the number of error bits introduced in the extracted data from the embedded image. Both the algorithm is implemented on all kind of images including aerial, texture, and miscellaneous image. It also successfully implemented on the medical images.

Key-Words: - Data Hiding, Robust data hiding, embedded image, steganography.

1 Introduction

Data hiding is the technique that is basically used for the covert communication. But the scope of the data hiding increased and used for various applications such as digital watermarking, fingerprinting, copy control, image authentication, etc. Applications of the data hiding techniques could be divided into two groups depending on the relationship between the embedded message and the cover image. The first group is formed by steganographic applications in which the message has no relationship to the cover image and it can be used only for communication. The content of the cover image has no value to the sender or the decoder. Its main purpose is to mask the secret message. The technique can be use for covert communication. The second group of applications is frequently addressed as digital watermarking. In a typical watermarking application, the message has a close relationship to the cover image. The message supplies additional information about the image, such as image caption, author signature, image authentication code, etc. In most cases, the cover media will experience some permanent distortion and can not be inverted to original media. Therefore, the digital watermarking is further divided into fragile watermarking and robust (semi-fragile) watermarking ([1], [2] and [3]). Many lossless data hiding algorithms [5] have been proposed in last few years ([6], [7] and [8]). However, most of them are fragile in the sense that hidden data can not be recovered after compression or any other incidental alteration applied to the marked image. As suggested in paper [1], De Vleeschouwer [2] method is only existing robust lossless data hiding technique against high quality JPEG compression. This technique can be applied to semi-fragile image authentication. But the drawback of this technique is that, it is suffered from salt-and-pepper noise. The performance of the De Vleeschouwer method is shown in fig1. In fig1, we can easily observe salt-and-pepper noise on the medical image. This happens because the De Vleeschouwer method uses the modulo-256 addition method so, a very bright pixel with larger grayscale value close to 255 will be possibly change to dark pixel with small grayscale value 0 and vice versa. To overcome this problem, Z. Ni and Y. Q. Shi [1] have designed new technique that is robust against high quality JPEG/JPEG2000 compression.

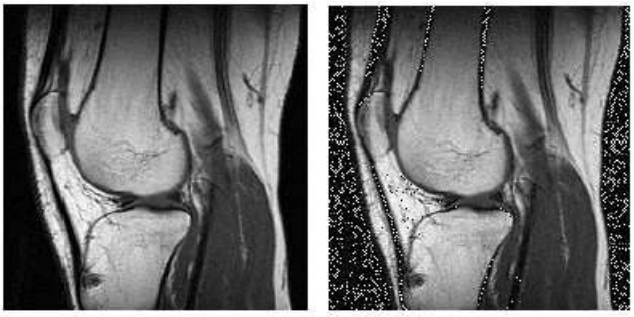

Fig1: Salt-and-Pepper noise on medical image

Here we use the same technique of data hiding presented in [1] and tried to improve the performance of the algorithm. Further we will see the algorithm and results and analysis of those results.

2 Proposed Method

As suggested in technique [1], image is divided into non-overlapping blocks, say 8x8, 10x10 etc. Each block is then further divided into two sets called set *A* and set *B* as shown in Fig 2.

+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+
+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+
+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+
+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+

Fig2: Pixel Marking Strategy

Set *A* consists of all pixels marked by "+" and set *B* consists of all pixels marked by "-". For each block robust statistical parameter is computed (difference value α). Then we use this value to embed a bit into each block. The difference value α is calculated as:

$$\alpha = \frac{1}{n}\sum_{i=1}^{n}(a_i - b_i)$$

where, a_i is the pixel value marked by the "+" sign and b_i is the pixel value marked by "-" sign and the n is the total number of pairs of pixels marked by "+" and "-" sign.

162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
164	164	158	155	161	159	159	160
160	160	163	158	160	162	159	156
159	159	155	157	158	159	156	157

Fig3: Sample 8x8 image block

In fig3, we have selected the 8x8 block size of image "lena", then we mark the pixels by "+" and "-" sign as shown in fig2. Hence the total number of pixels marked by "+" sign or "-" sign is 32, i.e, n=32 and the difference value α is 0.4375. Since the pixel grayscale values in a local block are often highly correlated and have spatial redundancy, the difference value α is expected to be very close to zero. The frequency distribution for the difference value α for image "lena" is shown if fig4. Graph in fig4 indicates that the difference value α is ranges from -3 to 3, that is, all these values gather around the zero.

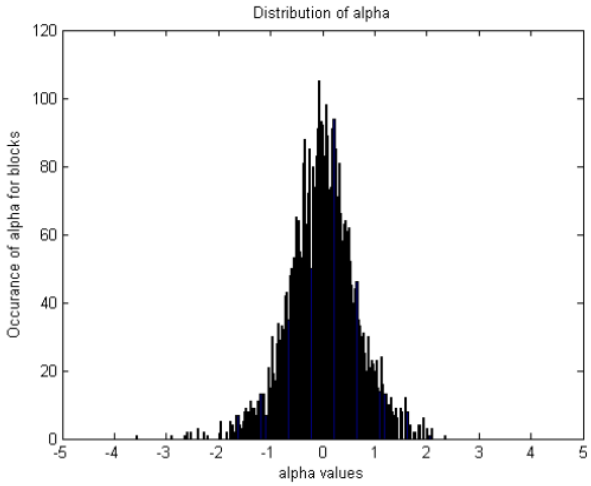

Fig4: Frequency Distribution of alpah

2.1 Bit Embedding Strategy

Bit embedding strategy here is slight different than technique [1]. Here, specified threshold for bit embedding is defined equal to a shift quantity β, but in technique [1] it was half to the shift quantity β.

The main idea for bit embedding is that the difference value α is kept within specified thresholds K and –K to embed bit "0" and the difference value α is shifted beyond the threshold K or –K to embed bit "1."

While doing this, one may encounter overflow-underflow problem, which mean that after data embedding, the grayscale values of the pixels in marked image may exceeds the upper bound (255 for gray level image) and/or lower bound (0 for gray level image). Hence, to avoid this problem, check every time while embedding bit that the shifting of pixels does not exceeds the limit of gray levels. If so, embed bit "0" in that block.

Bit embedding strategy is divided into two category based on the difference value α . First, if difference value α is zero or greater than zero (positive value of α). Second, if the difference value α is less than zero (negative value of α). Step by step algorithm is explained in fig5.

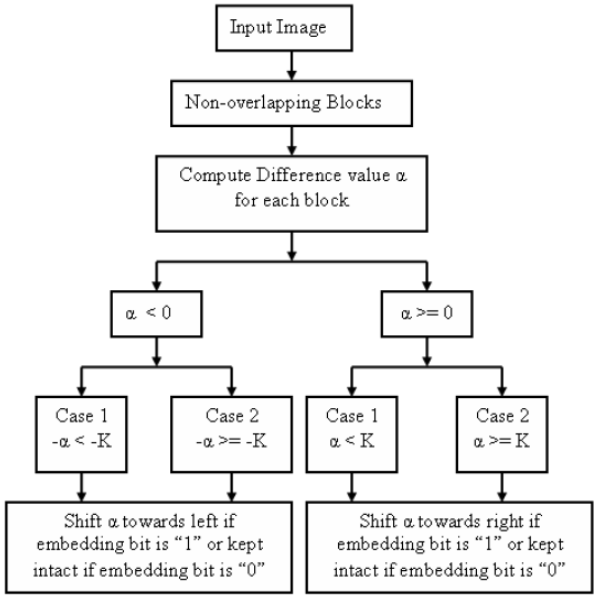

Fig5: Bit Embedding Strategy

Category1: In this category, the difference value α is always positive value, i.e, $\alpha \geq 0$. Threshold

is defined equal to β, hence following two cases are considered.

- **Case 1:** This case consider the difference value α between 0 and threshold K (0 <= α <= K). If embedding bit is "1" shifts pixels in set *A* towards right by a shift quantity β, or if the embedding bit is "0" keep block intact (fig6).

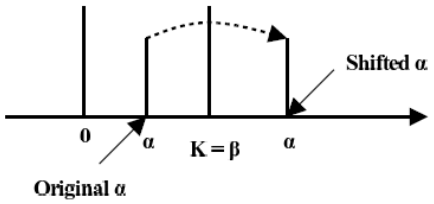

Fig6: Embedding bit "1"

- **Case 2:** In this case, the difference value α is beyond the threshold K, i.e, α >K. In this case, we always embed bit "1" whether embedding bit is "0" or "1"(fig7). Hence error bits may introduce in this case. To correct these error bits we use BCH (63,7,15) error correction code ([9] and [10]).

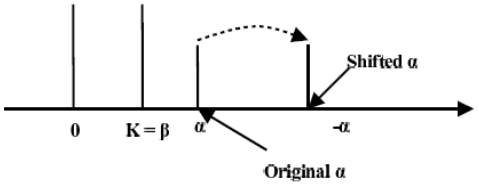

Fig7: Embedding bit "1"

Category2: Here, the difference value α is always negative, i.e, α < 0. This category also considers two cases.

- **Case 1:** In this case the difference value α is always between 0 and threshold –K. The operation of this case is same as case1 of category1 except that we shift the pixels in set *A* toward left by a shift quantity β, that is, if embedding bit is "1"then α is shifted towards left, otherwise block is kept intact (fig8).

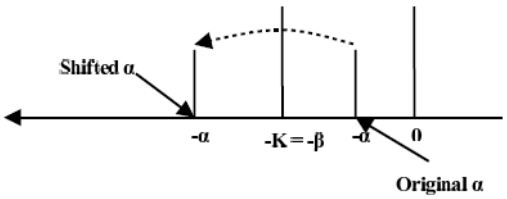

Fig8: Embedding bit "1"

- **Case 2:** In this case the difference value α is beyond the threshold –K. Again the operation of this case is same as the case2 of category1 except that the pixel values are shifted towards left. Hence the error bits may introduced in this case, because it always embed bit "1" whether embedding bit is "0" or "1" (Fig.9).

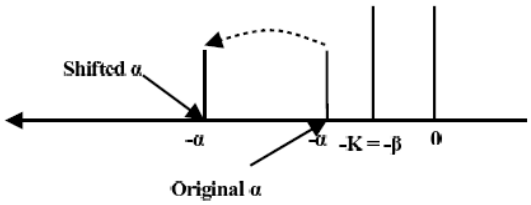

Fig9: Embedding bit "1".

2.2 Data Extraction

Data extraction process is exactly reverse process of data embedding. Like bit embedding process first we have to do blocking of an image (say 8x8, or 10x10), then find out the difference value α for each block. If difference value α is above the specified threshold (K or –K) extract bit "1" and shift the difference value α towards zero or it is between the specified threshold extract bit "0." In data extraction process error bits may introduce as mentioned in case2 of both category. To get the original data from the error bits we are using BCH error correction code [9][10]. According to error correction capability, there are different BCH code like BCH(15,11,1), BCH(15,7,2), BCH(15,5,3), BCH(31,6,7), BCH(63,7,15), etc. We are using the BCH(63,7,15) code because its error correcting capability is 15, that is, from the 63 bits message it can correct 15 error bits.

2.3 Preserve Robustness

To preserve the robustness of the algorithm we are using the difference value α. The difference value α is based on statistic of all pixels in the block, even though the pixel in the block has changed after JPEG/JPEG2000 compression, this statistic value α is not easy to change. Hence, this value intrinsically robust against JPEG/JPEG2000 compression and other incidental alteration. Therefore, the difference value α is selected as the robust quantity.

3 Experimental Results

In order to compare the performance of the method [1] and the proposed method, we have tested number of images such as standard images (Lena, Baboon, Boat, etc), medical images and test images of USC SIPI image database (Texture, Aerial, Miscellaneous, etc) .

TABLE I
TEST RESUSTS FOR MEDICAL IMAGES WITH BLOCK SIZE AS 8, EMBEDDING LEVEL AS 4 (Z. NI AND Y. Q. SHI TECHNIQUE)

Image Name	Image Size	Emb Cap	PSNR (dB)	Robustness (bpp)	Error Bits
Mpic1	256 x 256	112	41.64	0.45	13
Mpic2	256 x 256	112	41.49	0.49	14
Mpic3	256 x 256	112	41.68	0.48	12
Mpic4	256 x 256	112	41.41	0.47	12
Mpic5	256 x 256	112	41.59	0.47	4
Mpic6	480 x 640	532	42.12	0.42	0
Mpic7	480 x 640	532	42.11	0.44	0
Mpic8	480 x 640	532	42.12	0.37	0
Mpic9	480 x 640	532	42.11	0.37	0
Mpic10	480 x 640	532	42.11	0.38	0

Table I shows the test results for method [1] and Table II shows test results for proposed method on the medical images. In this experimentation, block size is selected as 8 and the embedding level is 4. Hence, the embedding capacity is 112. The PSNR of the proposed method is slightly increased than the technique [1]. There is no change in the robustness obtained by both techniques. Again the number of error bits introduced in propose method is decreased than the technique [1].

TABLE II
TEST RESUSTS FOR MEDICAL IMAGES WITH BLOCK SIZE AS 8, EMBEDDING LEVEL AS 4 (PROPOSED METHOD)

Image Name	Image Size	Emb Cap	PSNR (dB)	Robustness (bpp)	Error Bits
Mpic1	256 x 256	112	42.28	0.45	11
Mpic2	256 x 256	112	42.36	0.48	4
Mpic3	256 x 256	112	42.45	0.45	3
Mpic4	256 x 256	112	42.27	0.47	0
Mpic5	256 x 256	112	42.24	0.47	0
Mpic6	480 x 640	532	42.15	0.42	0
Mpic7	480 x 640	532	42.18	0.44	0
Mpic8	480 x 640	532	42.14	0.37	0
Mpic9	480 x 640	532	42.15	0.37	0
Mpic10	480 x 640	532	42.16	0.38	0

Table III to VI shows comparison of both technique on texture images. From table, it is seen that the number of error bits introduced in proposed method are decreased. Fig10 and fig11 show the graphical analysis of both techniques. Both the graphical results are taken for texture images

TABLE III
PSNRS WITH DIFFERENT EMBEDDING LEVELS, BLOCK SIZE IS SET AS 10 (Z. NI. AND Y. Q. SHI METHOD)

Image (512x512)	Embedding Levels 2 PSNR (dB)	4 PSNR (dB)	6 PSNR (dB)	8 PSNR (dB)	10 PSNR (dB)
Texture1	46.74	41.65	38.47	36.07	34.15
Texture2	47.03	41.92	38.58	36.1	34.16
Texture3	46.73	41.72	38.49	36.07	34.14
Texture4	46.91	41.86	38.57	36.1	34.16
Texture5	46.54	41.53	38.44	36.07	34.16
Texture6	46.3	41.24	38.27	36.03	34.15
Texture7	47.15	41.98	38.59	36.1	34.16
Texture8	48.05	42.12	38.59	36.1	34.16

TABLE IV
NUMBER OF ERROR BITS INTRODUCED WITH INCREASING EMBEDDING LEVEL, BLOCK SIZE SET AS 10 (Z. NI. AND Y. Q. SHI METHOD)

Image (512x512)	Embedding Levels 2 Error Bits	4 Error Bits	6 Error Bits	8 Error Bits	10 Error Bits
Texture1	45	3	0	0	0
Texture2	13	1	0	0	0
Texture3	32	1	0	0	0
Texture4	13	1	0	0	0
Texture5	44	4	0	0	0
Texture6	103	4	8	0	0
Texture7	5	1	0	0	0
Texture8	1	0	0	0	0

TABLE V
PSNRS WITH DIFFERENT EMBEDDING LEVELS, BLOCK SIZE AS 10 (PROPOSE METHOD)

Image (512x512)	Embedding Levels 2 PSNR (dB)	4 PSNR (dB)	6 PSNR (dB)	8 PSNR (dB)	10 PSNR (dB)
Texture1	47.67	42.01	38.61	36.14	34.23
Texture2	47.94	42.13	38.6	36.11	34.19
Texture3	47.74	42.09	38.6	36.1	34.14
Texture4	47.88	42.12	38.6	36.1	34.17
Texture5	47.54	42.1	38.6	36.1	34.17
Texture6	47.26	42.05	38.61	36.11	34.17
Texture7	48	42.13	38.61	36.11	34.18
Texture8	48.14	42.12	38.6	36.1	34.16

TABLE VI
NUMBER OF ERROR BITS INTRODUCED WITH INCREASING EMBEDDING LEVEL, BLOCK SIZE SET AS 10 (PROPOSED METHOD)

Image (512x512)	Embedding Levels				
	2	4	6	8	10
	Error Bits	Error Bits	Error Bits	Error Bits	Error Bits
Texture1	1	0	0	0	0
Texture2	0	0	0	0	0
Texture3	1	0	0	0	0
Texture4	1	0	0	0	0
Texture5	2	0	0	0	0
Texture6	4	0	8	0	0
Texture7	1	0	0	0	0
Texture8	0	0	0	0	0

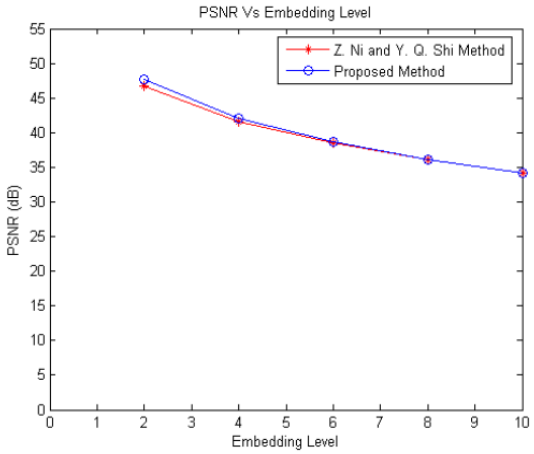

Fig10: PSNR Vs Embedding Level

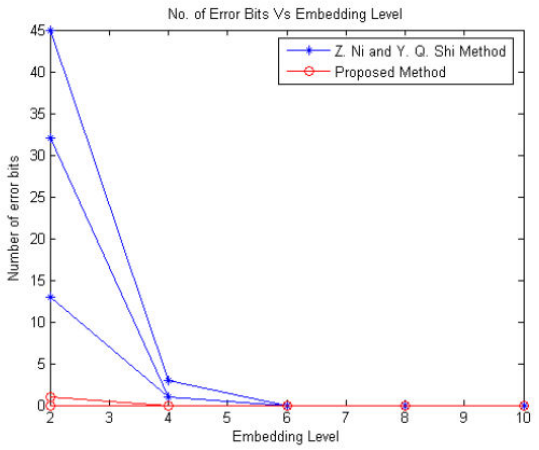

Fig11: Number of Error Bits Vs Embedding Levels

4 Conclusion

A data hiding algorithm proposed here is slightly better than technique [1] to preserve the robustness. Experimental results shows that the in proposed algorithm the PSNR of the embedded image is above 38dB and the number of error bits introduced are also reduced to zero as increasing the block size and embedding level. In this way we can conclude that the algorithm does not generate salt-and-pepper noise, applicable to all images, robust against JPEG/JPEG2000 compression and bit embedding capacity varies with block size selected to embed a bit.

References:

[1] Zhichen Ni, Yun Q. Shi, Nirwan Ansari, Wei Su, Qibin Sun and Xiao Lin, "Robust Lossless Image Data Hiding Designed for Semi-Fragile Image Authentication," *IEEE Trans,* Vol 18, 497-509 (2008).

[2] Christophe De Vleeschouwer, Jean-Francois Deloaigle, and Benoit Macq, "Circular Interpretation of Bijective Transformation in Lossless Watermarking for Media Asset Management," *IEEE Trans,* Vol 5, 97-104(2003).

[3] Yun Q. Shi, Zhicheng Ni, Dekun Zou, Changuin Liang and Guourong Xuan, "Data Hiding: Fundamentals, Algorithms and Applications," *IEEE,* 33-36(2004).

[4] Dekun Zou, Yun Q. Shi, Zhicheng Ni, and Wei Su, "A Semi-Fragile Lossless Digital Watermarking Scheme Based on Integer Wavelet Transform," *IEEE Trans,* Vol 16, 1294-1300(2006).

[5] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for Data Hiding," *IBM Syst. J*, Vol. 35, 313-336 (2003).

[6] Z. Ni, Y. Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *in Proc. IEEE Int. Symp. Circuits Syst, Bangkok, Thailand*, May 2003, pp. 912-915.

[7] M. Goljan, J. Fridrich, and R. Du, "Distortion-free data embedding," *in Proc. 4th Inf. Hiding Workshop, Pittsburgh, PA*, pp. 27-41, Apr. 2001.

[8] J. Fridrich, M. Goljan, and R. Du, "Invertible authentication," *in Proc. SPIE Photonics West, Security and Watermarking of Multimedia Contents III, San Jose, CA,* Vol. 397, pp. 197-208, Jan. 2001.

[9] H. W. Wallace, "Error Detection and Correction using the BCH Code."

[10] J. G. Proakis, "Digital Communication," 4th ed. New York: McGraw-Hill, 2000.