

OVERCOMING THE EFFECT OF JPEG COMPRESSION ON STEGANOGRAPHY

*Dr. Imad H. AL-Hussaini

**Dr. Hussain J. Abass

***Jamal M. A. Al-Towayjri

Received on: 6/5/2004

Accepted on: 18/7/2004

ABSTRACT

This work presents an effective approach using a novel coding scheme to overcome the errors caused by JPEG (Joint Photographic Expert Group) compression. The proposed novel coding technique (distance-to-origin) depends on viewing the pixel as a point in space with the three color channel values as the coordinates in the spatial domain. It is used as steganography to improve the quality of the image at the receiving end. In this work, data (text and various images) was hidden within three different covers using the Least Significant Bit (LSB) coding technique and the novel coding technique (distance-to-origin), and then a JPEG compression-decompression was applied. The error percentage for each steganography/compression was computed and then analyzed. Comparisons were made for different types of steganography/compression with and without using the proposed approach to show the correctness and consistency of this work.

الخلاصة

هذا البحث يقدم طريقة مثالية وعملية باستخدام مخطط (novel coding) للتغلب على الأخطاء الناتجة عن الضغط للمعلومات بطريقة JPEG. تعتمد الطريقة المقترحة على الأخذ بنظر الاعتبار بأن كل نقطة ضوئية (Pixel) هي نقطة في نظام إحداثي متكون من ثلاثة محاور، هي: الأحمر، والأخضر، والأزرق. في هذه الطريقة، تم إخفاء كتابة وصورة داخل ثلاث صور غطاء مختلفة من حيث الخواص (تدرج الألوان، وتردد الألوان)، وتم اختبار الفرق بين استخدام هذه الطريقة والطريقة العادية وهي LSB Replacement، ونسبة الخطأ التي يسببها الضغط بطريقة JPEG في الحالتين على المعلومات المخفية. وقد بينت النتائج نجاح الخوارزمية المقترحة لهذه الطريقة في هذا البحث في تقليل تأثير طريقة ضغط

Key Words

JPEG Compression - Decompression, Steganography, Hidden Information, LSB Insertion.

*Computer & Software Eng. Dept., University of Technology, Baghdad, Iraq.

Ministry of Sciences & Tech. *Laser and Optoelectronics Eng. Dept., University of Technology, Baghdad, Iraq.

1. Introduction

Steganography, which literally means "Covered writing" in Greek, is the art of hiding the very existence of a message. This digital steganography conceals even the evidence of encrypted messaging. It is the science of communicating in a hidden manner [Johnson N., 1998].

Steganography plays an increasingly important role in today's connected society, as the demand for covert communication continues to rise. It is a powerful idea, since an enemy cannot read the hidden information unless they can first detect it. Sometimes steganographic methods combine traditional method of cryptography with steganography that is the sender encrypts the secret message before embedding it into the selected cover. Therefore, this process increases the security of the communication process [Benjamin B., 2001].

1. Steganography Media

Steganography uses many media to embed data such as Image, Audio, and Text. It is very important to remember the following restrictions when embedding data:

1. The embedded data should be directly encoded into the media, rather than into a header or wrapper. So that the data remain intact across varying data file formats.
2. The embedded data should be as minimum as possible to modifications from intelligent attacks or anticipated manipulations such as channel noise, filtering, reassembling, cropping, encoding lossy compression, printing and scanning, digital-to-analog (D/A) conversion and analog-to-digital (A/D) conversion.
3. The cover data should not be significantly degraded by the embedded data, and the embedded data should be as imperceptible as possible.

4. A symmetrical coding of the embedded data is desirable, since the purpose of data hiding is to keep the data in the host signal, but not necessarily to make the data difficult to access.

5. Some distortion or degradation of the embedded data can be expected when the cover data is modified. To minimize this, error-correcting codes should be used.

i. Steganography in Texts.

Soft copy text is in many ways the most difficult place to hide data. While, hard-copy text can be treated as a highly structured image and is readily amenable to variety of techniques such as slight variations in letterforms, kerning, baseline. This is largely due to the relative lack of redundant information in a text file as compared with a picture or a sound. Data hiding in text is an exercise in the discovery of modifications that are not noticed by readers. Three major methods of encoding data (Line-shift coding, word-shift coding and Feature coding) with some alternative methods will be considered in the following subsections [Katzen B., 2000].

ii. Steganography in Images.

Image steganography has come quite far in recent years with the development of fast, powerful graphical computers, and steganographic software is now readily available over the Internet for everyday users. To computer, an image is an array of numbers that represent light intensities at various points, or pixels. These pixels make up the image's raster data.

An image size of 640 by 480 pixels, utilizing 256 colors (8 bits per pixel) is fairly common. Digital images are typically stored in either 24 bit or 8 bit pixel files. 24 bit images are sometimes known as true color images. Clearly, a 24 bit image provides more space

for hiding information; however, 24 bit images are generally large and not common. 24 bit image (of 1024 pixels wide by 768 pixels height) would have a size in excess of 2 megabytes as such large files would attract attention when they are to be transmitted across a network or the Internet. Therefore, image compression is desirable.

Alternatively, 8 bit color images can be used to hide information. In 8 bit color images, (such as GIF files), each pixel is represented as a single byte. Each pixel only points to a color index table or palette, with 256 possible colors. The pixel's value, then, is between 0 and 255. The image software merely needs to paint the indicated color on the screen at the selected pixel position [Petitcolas F., 1998].

If using 8 bit images as the cover-image, many steganography experts recommend using images featuring 256 shades of gray as the palette, for reasons that will become apparent. Gray-scale images are preferred because the shades change very gradually between palette entries. This increases the image's ability to hide information [Watt A., 1998].

The steganography will need to consider the image as well as the palette in 8 bit images. Obviously, an image with large areas of solid color is a poor choice, as variances created by embedded data might be noticeable. Once a suitable cover image has been selected, an image encoding technique needs to be chosen [Gonzalez R., 1995, Lala Z., 2000].

Least Significant Bit (LSB) insertion

The Least Significant Bit insertion method is probably the most well-known image steganography technique. It is a common, simple approach to embedding information in a graphical image file, but it is extremely vulnerable to attacks, such as image manipulation. A simple conversion from GIF or BMP format to a lossy compression format such as JPEG can

destroy the hidden information in the image [Schneier B., 1996].

When applying LSB technique to each byte of 24 bit image, three bits can be encoded into each pixel. A 1024x768 image has the potential to hide a total of 2,359,296 bits (294,912 bytes) of information.

Any change in the pixel bits will be indiscernible to the human eye, and if it is compressed prior to hiding, then a large amount of information can be embedded [Schroeder M., 2000].

2. JPEG Operation

In this section, the JPEG standard will be described, which has become the leading standard for still image compression applications and motion video applications. The JPEG algorithm provides lossy compression including 8x8 DCT, which is the mandatory part of the algorithm, uniform quantization, and Huffman coding. Fig. (1) shows JPEG encoder steps.

The Decoding Process

The compression procedures of the JPEG algorithm are illustrated. The image is processed from left to right and from top to bottom. (8x8) pixel blocks are typically input block by block, each block is transformed, quantized and coded.

Fig. (2) illustrates how the data is decompressed. The first step of decompression recovers the run length symbol pairs and the differential DC-encoded from the coded bit stream. By making a single pass through the inverse run-length, the zigzag sequence is recovered.

4. JPEG Compression Effects

As mentioned before, embedding data in the Least Significant Bit of image pixels is a simple steganographic technique, but it cannot survive the deleterious effects

of JPEG. To investigate the possibility of employing some kinds of encoding to ensure survivability of embedded data, it is necessary to identify what kind of loss/corruption JPEG causes in an image and where in the image it occurs.

At first glance, the solution seems to look at the compression algorithm trying to predict mathematically where changes to the original pixels will occur. This is impractical since the DCT converts the pixel values to coefficient values representing 64 basis signal amplitudes. This has the effect of spatially smearing the pixel bits so that the location of any particular bit is spread over all the coefficient values. Because of the complex relationship between the original pixel values and the output of DCT, it is not feasible to trace the bits through the compression algorithm and predict their location in the compressed data.

To study the effects of JPEG algorithm an empirical approach was done. 24-bit windows BMP format files were compressed, decompressed, and the resulting file saved under a new filename.

The BMP file format was chosen for its simplicity and widespread acceptance for image processing applications. For the experiments of three images, one of a Balloon, the second of Cats, and the third of Galileo, were chosen for their different amount of details and number of colors since JPEG is sensitive to these factors.

In order to study the effect of JPEG on LSB replacement steganography, the block diagram shown in Fig. (3) was implemented upon the three chosen images.

The obtained results for byte-by-byte and pixel-by-pixel compressions are shown in Table (1) and Table (2) respectively.

Table (1) shows that the numbers of errors in the lower four bits are increased with the increase in image frequency. While the

errors in the upper four bits highly depend on the color range of the image (Galileo image has the least error in the upper four bits).

Generally the JPEG highly scrambles most of the bits in the image.

Table (2) shows that number of bits changed in each pixel (hamming distance) after JPEG compression-decompression depends mostly on the color range of the image. While the higher frequency image has a higher short-hamming distance.

5. The Developed Algorithm

Given the information in Table (1) and Table (2), it is apparent that data embedded in any or all of the lower 5 bits would be corrupted beyond recognition. Attempts to embed data in these bits and recover it after JPEG processing showed that the recovered data was completely garbled by JPEG.

JPEG is indiscriminate in the amount of changes it makes to byte values and produces enough errors to make the hidden data unrecognizable.

The negative results of such attempts to embed data indicated that a more subtle approach to encoding was necessary. It was noticed that, in a JPEG processed image, the pixels, which were changed from their original appearance, were similar in color to the original. This indicates that the changes made by JPEG, to some extent, maintain the general color of the pixels. To attempt to take advantage of this, a new coding scheme was devised based on viewing the pixel as a point in space with the three colors channel values as the coordinates in Fig. (5).

The coding scheme as shown in Fig. (5) begins by computing the distance d from the pixel to the origin (0, 0, 0) by equation (1).

$$d = \sqrt{R^2 + G^2 + B^2} \quad \dots\dots (1)$$

Then the value of d is rounded to the nearest integer, and then d is divided by a number (modulus M) and the remainder ($r = d \bmod M$) is found. The pixel value is adjusted such that its remainder is changed to a number corresponding to the bit value being encoded. Qualitatively, this means that the length of the vector representing the pixel's position in three-dimensional RGB color space is modified to encode information. Because the vector's direction is unmodified, the relative sizes of the color channel values are preserved.

Suppose choosing an arbitrary modulus of 42. When the bit is decoded, the distance to origin will be computed and any value from 21 to 41 will be decoded as a 1 and any value from 0 to 20 will be decoded as a 0. So changing the pixel value to a middle value in one of these ranges is desired to allow for error introduced by JPEG. In this case, the vector representing the pixel would have its length modified so that the remainder is 10 to code a 0 or a 31 to code a 1. It was hoped that JPEG would not change the pixel's distance from the origin by more than 10 in either direction, thus allowing the hidden information to be correctly decoded.

For example, given a pixel (128, 65, 210) the distance to the origin would be computed:

$$d = \sqrt{128^2 + 65^2 + 210^2}$$

= 254.38. The value of d is rounded to the nearest integer. Next find $r = d \bmod 42$, which is 2. If a 0 is being coded in this pixel, the amplitude of the color vector will be increased by 8 units to an ideal remainder of 10 ($d = 262$) and moved down 13 ($d = 241$) units to code a 1. Note that the maximum displacement any pixel would suffer would be 21.

In other words, r value ranges from 0 to $M-1$ and this range is divided into two sections (0 to $M/2$) for bit 0 coding and ($M/2+1$ to $M-1$) for bit 1 coding. In order to avoid JPEG

errors the coding of bit 0 was made so that $r = M/4$ (in the middle of 0 part) and for bit 1 coding.

$r = 3M/4$ (in the middle of 1 part) as shown in Fig. (7).

Simple vector arithmetic permits the modified values of the red, green, and blue components to be computed. Simply squaring the change in d value, then divide it by three, after which taking the square root, and finally add the result to (or subtract the result from) each of the R, G, and B values, as shown in Fig. (8).

The value of the modulus was made to be adjustable in the developed program so that the effect of Modulus on error percentage caused by JPEG compression can be noticed. The range of Modulus was chosen to be from 22 to 82 and the corresponding results were taken.

To verify that this method hides (text, or images) in a way that data can survive after JPEG compression-decompression process, cover images are first opened in BMP format and the data is hidden according to this specific method and with a specific modulus value.

Then JPEG compression is carried on the image. The JPEG image is now transferred back to BMP format and the hidden data is recovered from it. By calculating the remainder of distance to the origin of each pixel according to the chosen modulus in hiding, the remainder exists in the lower half range values of the modulus, and the recovered bit is 0. While if the remainder is in the upper half range values of the modulus, then the recovered bit is 1, as shown in Fig. (9).

Then error percentages caused by JPEG compression were computed by (bit-by-bit) comparison of the recovered data with the originally hidden data in the cover image.

Two types of data hiding were implemented using the distance to origin method, text

under image, and image under image hiding. In text hiding method, a multiple redundancy was used to decrease the error caused by JPEG. Multiple redundancies mean each data bit was hidden three times in the image. Therefore, after retrieving the hidden data, the majority value of these three bits will be the recovered bit value. For example, if the three values retrieved for a specific hidden bit was (1, 1, 0), then the decision will be made so that the recovered bit value is (1) since two out of three was one.

It must be noticed that in both implemented cases (text in image, and image in image) a 100 % replacement was made. Although it is not recommended to make such a high replacement percentage, (usually not larger than 75% replacement) because high replacement percentages would highly affect the cover image quality.

The reason behind 100% replacement used was to clarify the effect of the chosen image characteristics (color range, and frequency) on the error percentages.

6. Results

6.1 Text under Image

The error percentages range from about 46% to 9% caused by JPEG are increased in images of high frequency and wider color range. And although error percentage was greatly enhanced after applying redundancy in hidden data, it is still a big noise to the letters. Since letters are so sensitive to errors, then a change in one bit changes the character greatly. The results are shown in Fig. (10).

6.2 Image under Image

Images are less sensitive to errors than letters, and to make use of this fact any secret image or secret message could

be written as an image that is to be hidden in a cover image. In image under image experiments, the same images, as in text under image, were used, but with (400X240) pixels resolution used to hide (120X100) pixels secret images one of the images is message image and the other is flower image. No data redundancy was used in hidden image under image for many reasons. First, because the error percentages caused by JPEG do not highly affect the readability of the message inside image. Second, taking the advantage of much more data that can be hidden, which is better than inserting multiple copies of data to be hidden. Third, the errors introduced by JPEG in this case are much less than text under image cases. This can be explained easily by recalling the fact that JPEG introduces more errors in the high frequency regions, and since the text data are more robust than the image data, the errors are much less for the image under image case as the results show.

The error percentages caused by JPEG range from about 45% to 9% and are increased in images of high frequency and wider color range (Cats image with flowers image), while decreased in images of low frequency and narrow color range (Galileo image with message image). The larger the value of modulus (M) used, the less the errors caused by JPEG. But increasing M dramatically affects the cover image quality. M=42 may also be the best choice for good error percentage and good cover image quality. The relationship between error percentage and M using three images for message image and flowers image are shown in Fig. (11) and Fig. (12) respectively.

The error percentage in most cases is considered to be acceptable since in most cases the text inside the recovered secret

message is still readable. For bigger messages, bigger cover images should be used. Fig. (13) shows the error percentage for message and flower images using three images with different M's.

Many methods could be used to enhance the visibility of the recovered image such as, character pattern recognition, or noise elimination filters.

7. Conclusion

Because of the discrete cosine transform and the quantization process available in the JPEG compression algorithm, low frequency images have less error under JPEG compression, while high frequency images undergo a large number of errors. By using wide color range, the JPEG compression causes errors in the higher order bits. Also, high frequency images with wide color range would have a large number of errors in both high and low order bits, and vice versa.

Increasing the modulus (M) would decrease the errors caused by JPEG, but at the same time would affect the image quality of the cover image.

In the text hiding method, whatever the error percentage is low, the readability of the recover text would be highly affected.

Low frequency secret image (message image) in image hiding method produces better results for all values of modulus (M). While high frequency secret images (flowers image), will produce larger error percentages for all values of modulus (M). Inserting multiple copies of data to be hidden in the cover image would help decreasing the final error percentage for all values of modulus (M), by taking into account the original error percentage to be not greater than (45 %) for triple redundancy.

References

- Benjamin B., Santiago G., Victor B., (2001), "Steganographic Watermarking for Documents", Proceedings of the 34th Hawaii International Conference on System Science.
- Gonzalez R., Woods R., (1995), "Digital Image Processing", Addison Wesley Publishing Company.
- Johnson N., Sushil J., (1998), "Steganalysis of Images Created Using Current Steganography Software", Information Hiding: 2nd International Workshop, Springer.
- Katzen B., Petitcolas A., (2000), "Information Hiding Techniques for Steganography and Digital Watermarking", Artech House, USA.
- Lala Z., (2000), "Image in Image Steganography", Ph.D. Thesis, University of Technology, Baghdad-Iraq.
- Petitcolas F., Anderson R. and Kuhn M., (1998), "Attacks on Copyright Marking Systems", Information Hiding: 2nd International Workshop, Proceeding, Springer.
- Schneier B., (1996), "Applied Cryptography", John Wiley and Sons Inc.
- Schroeder M., (2000), "JPEG Compression Algorithm and Associated Data Structures", University of North Dakota.
- Watt A., Policapro F., (1998), "The Computer Image", Addison Wesley, Acm press.

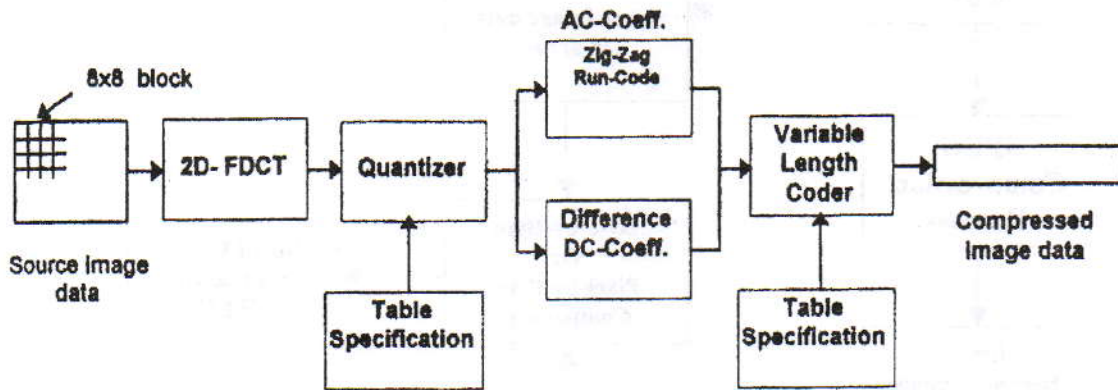


Fig. (1) JPEG encoder steps

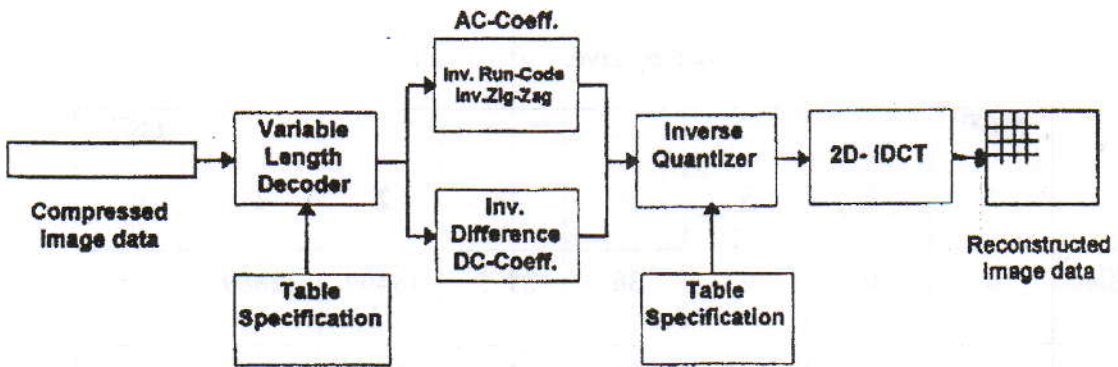


Fig. (2) JPEG decoder steps

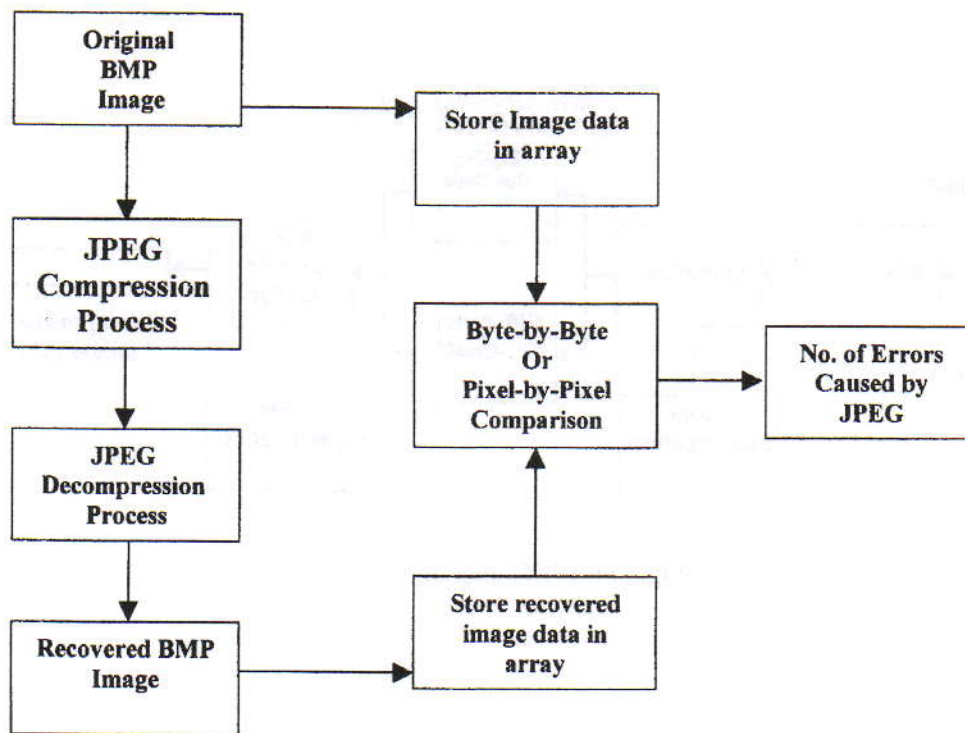


Fig. (3) Block diagram of comparison method

Table (1) Byte-by-Byte Comparison

	MSB							LSB
	7	6	5	4	3	2	1	0
Galileo	0	0	0	36	2977	18409	32889	38154
Balloon	0	36	607	2565	6716	12514	24782	37236
Cats	0	0	233	3710	14454	25792	34064	38216

Table (2) Pixel-by-Pixel Comparison

	1-3	4-6	7-9	10-12	13-15	16-18	19-21	22-24
	Bits	Bits	Bits	Bits	Bits	Bits	Bits	Bits
Galileo	12756	11021	922	8	0	0	0	0
Balloon	14806	7268	1919	136	4	0	0	0
Cats	7645	14202	3567	117	0	0	0	0




			
Error % after standard JPEG using LSB Replacement	Error =48.48%	Error =49.76%	Error =49.68%

Fig. (4) Error % after standard JPEG compression-decompression using LSB replacement for Balloon, Cats & Galileo images

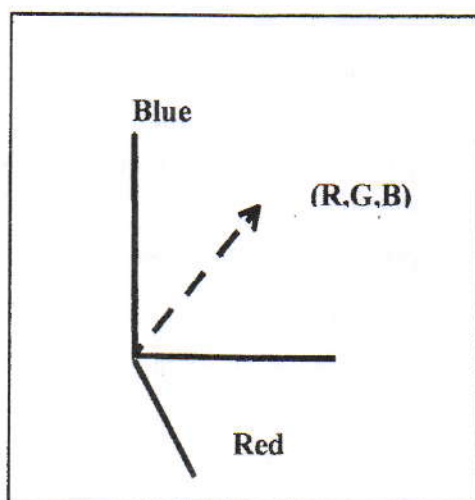


Fig. (5) R,G,B Color Coordinates

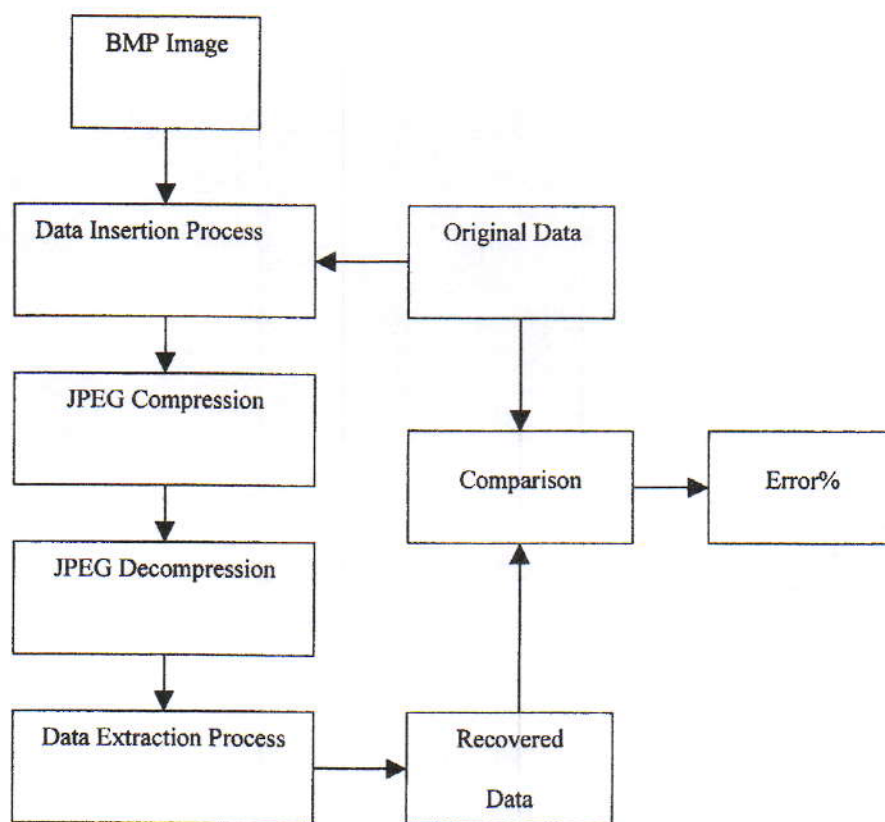
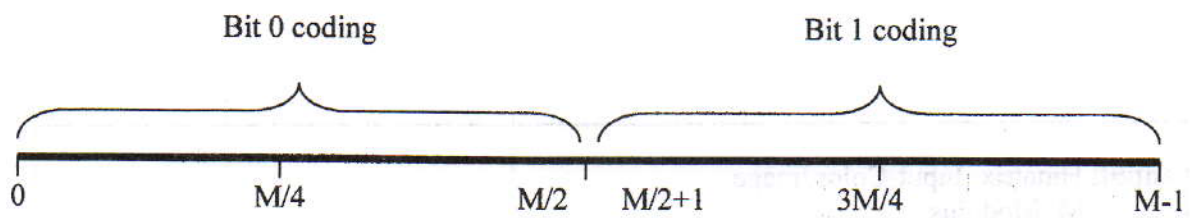


Fig. (6): Block diagram of embedding system

Fig. (7) r values range

Input: Hmatrix: Color-Image-Input

M: Modulus

Output: Modified values of RGBmatrix

Procedure:

Begin

Hmatrix \leftarrow Input image data

{* find RGB to each pixel *}

RGBmatrix \leftarrow Find RGB (Hmatrix)

For all pixels in image do

Begin

{* compute the distance into d *}

$d \leftarrow \text{sqrt}(\text{RGBmatrix}(R)^2 + \text{RGBmatrix}(G)^2 + \text{RGBmatrix}(B)^2)$

$d \leftarrow \text{Round-nearst-integer}(d)$

$r \leftarrow \text{Mod}(d, M)$

If $d = 0$ then

$r \leftarrow M/4$

else

$r \leftarrow (3*M)/4$

end else

{* compute the value that d must be moved *}

Update - Value (d)

Update - RGB - Values (RGBmatrix)

Re - Save the pixel RGB values into image

End For

End Procedure.

Fig. (8) Insertion Process Algorithm

Input: Hmatrix: Input-Color Image
M: Modulus

Output: r: remainder of distance
If $r = 0$ lower half range values of the modulus
r = 1 upper half range values of the modulus

Procedure:

Begin

Hmatrix \leftarrow Input image data

RGBmatrix \leftarrow Values – RGB (Hmatrix)

For all pixels in image do

Begin

d \leftarrow sqrt (RGBmatrix (R)² + RGBmatrix(G)² + RGBmatrix (B)²)

d \leftarrow round-nearest-integer (d)

r \leftarrow Mod (d, M)

If $0 < r < M/2$ then

Set the corresponding recovered data bit value to 0

else

Set the corresponding recovered data bit value to 1

end else

End For

End Procedure.

Fig. (9) Extraction Process Algorithm















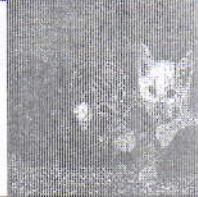
Galileo	M=22	M=42	M=62	M=82
				
Error %	44.488%	31.523%	22.109%	15.378%
After redundancy Error %	42.141%	25.128%	13.980%	8.758%
Balloon	M=22	M=42	M=62	M=82
				
Error %	35.429%	27.254%	20.480%	15.785%
After redundancy Error %	30.726%	21.360%	14.808%	9.731%
Cats	M=22	M=42	M=62	M=82
				
Error %	45.754%	35.703%	27.921%	22.140%
After redundancy Error %	43.847%	31.273%	22.273%	15.914%

Fig. (10) JPEG compression for text under image and error percentage using Galileo, Balloon and Cats images.

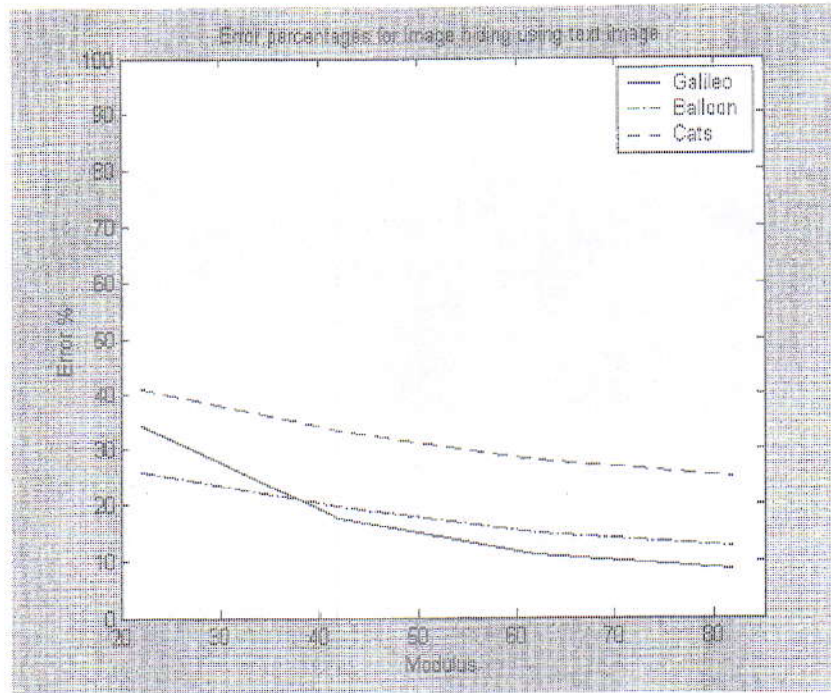


Fig. (11) Relationship between error percentage for message image & Modulus (M) using the three images

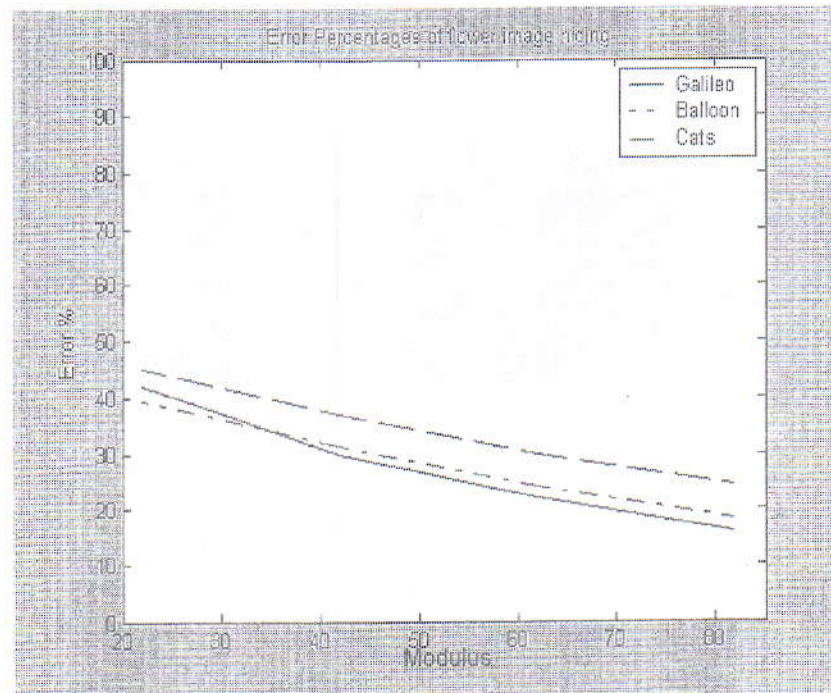


Fig. (12) Relationship between error percentage for flowers image & Modulus (M) using the three images







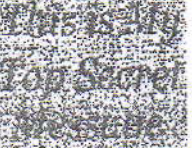
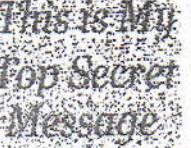
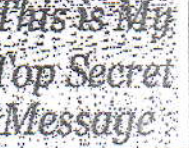











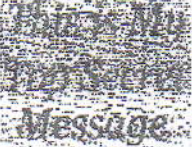
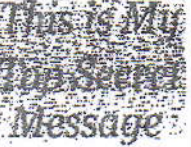













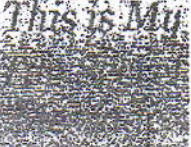
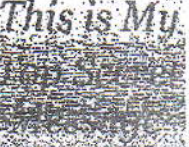





Galileo	M=22	M=42	M=62	M=82
				
<i>This is My Top Secret Message</i>				
Error %	33.934%	17.642%	11.136%	8.55%
				
Error %	41.922%	29.946%	21.932%	16.054%
Balloon	M=22	M=42	M=62	M=82
				
<i>This is My Top Secret Message</i>				
Error %	25.991%	19.808%	14.967%	12.6%
				
Error %	39.325%	31.353%	24.003%	18.304%
Cats	M=22	M=42	M=62	M=82
				
<i>This is My Top Secret Message</i>				
Error %	40.935%	33.201%	27.851%	25.028%
				
Error %	44.892%	37.032%	29.977%	24.293%

Fig. (13) Error % for message & flowers images using the three images with different M's