2008

An information hiding scheme using a pattern-based compression algorithm

Farhad Keissarian
University of Wollongong in Dubai, farhadk@uow.edu.au

An Information Hiding Scheme using a Pattern-based Compression Algorithm

Farhad Keissarian
University of Wollongong in Dubai
farhadkeissarian@uowdubai.ac.ae

Abstract

This paper presents a data hiding scheme that hides the data in the compression domain of a digital image. The pattern-based compression algorithm used for hiding the information, compresses the image according to the visual activity of individual image blocks. The key point of the proposed scheme is to first identify the smooth area of the host image and then embed the data in these areas. Experimental results confirm that the proposed scheme can embed a large amount of data in the compressed file while maintaining satisfactory image quality.

1. Introduction

Nowadays, digital data of all kinds play a very important part in information communication. Most mass media data, such as videos, audios, images, magazines, newspapers and so on, are usually used to convey and share information. On the Internet or other local computer networks, some pieces of information are made openly accessible to all the public. However, other pieces of information may be considered confidential data and thus need to be kept or transmitted securely. As a result, we are faced with the great challenge of transmitting secret data safely from the sender to the right receiver through an open channel without anything leaking out. One of the possible ways to accomplish this is to embed secret data inside a cover carrier that appears meaningful but not important. Through this way, the existence of the secret data can be concealed and the attention of attackers will be avoided. Data hiding techniques that follow this strategy are called 'methods of steganography'[1].

The word steganography, which is composed of the two Greek words *steganos* and *graphia*, means "hidden writing" or "covered writing." Among the wide variety of cover carriers, digital images are the most commonly used because they are the most ubiquitous and readily available on the Internet. In addition, the higher degree of distortion tolerance that digital images have over other kinds of plaintext data provides them with a larger hiding capacity. Therefore many data hiding schemes have been proposed especially for digital images as the cover media. In general, an image before any secret data gets hidden in is called a cover image, and the term stego-image is for an image with the secret message already embedded in [2].

All the data hiding techniques should satisfy at least two fundamental requirements [3]. One of the two requirements is imperceptibility; that is to say, the embedded image, also known as the stego-image, must not be degraded so much that any perceivable difference occurs between the stego-image and the original cover image. A less distorted image possesses good image quality, looks like a normal picture transmitted on the Internet, and should not attract the attention of hostile interceptors. The other fundamental requirement is a high-embedding capacity; in other words, the quantity of secret data that can be embedded in a cover image should be as large as possible. However, there exists a natural trade-off relationship between the stego-image quality and the embedding capacity: the more secret data embedded in the cover image, the more severely degraded the stego-image would be and vice versa. Therefore, the challenge is to develop a data hiding scheme that can offer both a high stego-image quality and a high-embedding capacity at the same time.

Digital data hiding techniques can be roughly classified into three kinds: spatial domain methods, frequency domain methods and compression domain methods. The difference here is the domain where the embedding happens. Methods in the spatial domain work by directly replacing the raw data of the digital image with secret data. One simple and well-known

978-0-7695-3122-9/08 $25.00 © 2008 IEEE
DOI 10.1109/SITIS.2007.105

example of such schemes is the least-significant-bit (LSB) hiding [4]. As the name implies, it replaces the least significant bits of each pixel value in the original image with secret data bit by bit. Because of its high capacity and simple implementation, a number of variants of the LSB method has been developed so far. Different from spatial domain methods, frequency domain methods first transform an image from the spatial domain to the frequency domain by using discrete cosine transform, discrete Fourier transform or discrete wavelet transform. And then, the secret data is hidden inside the transformed coefficients [5, 6].

In recent years, some researchers have concentrated on a third possibility: embedding secret data into the compression domain . Image compression used in some of these studies includes vector quantization (VQ) [7], and block truncation coding (BTC) [8]. Owing to the rapidly growing number of digital images stored in computer memory space or transmitted on the Internet, more and more digital images are stored or transmitted in a compressed form so as to minimize memory space consumption or to deal with the limited-bandwidth problem. For this reason, if secret data can be directly embedded into the compressed codes of the image, then we can spare all those decompression and recompression processes. Furthermore, the compressed codes transmitted on the Net attract less attention than the raw data itself .

In this paper, we develop an image hiding scheme that can hide the secrete data into compression codes of the host image, generated by the compression technique that we reported earlier in [9]. The key point of the proposed scheme is to embed the data in the smooth area of the host image. The rest of the paper is organized into four sections. The concept of the proposed compression algorithm is introduced in Section 2. In section 3, the proposed hiding scheme is presented. Experimental results are given in Section 4, and finally some conclusions are made in Section 5.

2. Pattern-based Compression Algorithm

In the proposed compression algorithm in [9], an image is block coded according to the type of individual blocks. A novel classifier, which is designed based on the histogram analysis of blocks, is employed to classify the image blocks according to their level of visual activity. Each block is then represented by a set of parameters associated with the pattern appearing inside the block. The use of these parameters at the receiver reduces the cost of reconstruction significantly and exploits the efficiency of the proposed technique.

2.1. Block Classification

A novel histogram-based classification scheme has been developed for classifying the image blocks. The method operates based on the distribution of the block residuals and classifies block either as a low-detail (uniform) or as a high-detail (edge) block. The classifier employs the residual values of a block and classifies the block according to the shape of the histogram of these values. The classification is carried out through a peak detection method on the block histogram. A brief description of the classifier is as follows.

Each block of *4x 4* pixels is converted into a residual block by subtracting the sample mean from the original pixels. The residual samples are less correlated than the original samples within a block. Here, two of the most important local characteristics of the image block are considered*: central tendency*, represented by the mean value and the *dispersion* of the block samples about the mean, which is represented by the residual values. The challenge here is to analyze the dispersion of the residual values about the mean. One way of achieving this is to sort the histogram of the block residual samples. As the neighboring pixels in the original block are highly correlated, the residual samples will tend to concentrate around zero. One can then quantize the residual samples prior to forming the histogram. The histogram of the quantized residuals may then be formed and analyzed by simply detecting its peaks.

A peak on the histogram indicates a high score of residual values; therefore it is fair to conclude that there is a considerable number of pixels that have the same dispersion about the block mean. This, in turn will lead us to conclude that the gray level values of these pixels are very close to one another. Hence, this group of pixels can be represented by a single intensity value. According to the number of detected peaks on the histogram image blocks can be placed into two major categories of uniform and edge blocks. A histogram with a unique peak at its centre (uni-modal histogram) identifies a uniform block. The existence of two peaks or more implies that the processed block is an edge (high detailed) block.

A minimum score, $Score_{\min}$ can be defined below which a peak is not detected. Setting the $Score_{\min}$ to different values will result in different numbers of detected peaks on the block histogram. Obviously, a

lower (higher) $Score_{\min}$ results in more (less) number of detected peaks. Fig. 1 depicts an example of 4 x 4 uniform block and its histogram.

2.2. Coding of the Image Blocks

Once the image blocks have been classified, the coder, switches between a one-level and a bi-level representation depending on whether the block is uniform or edge. A peak on the histogram of an image block demonstrates a high score of residual values indicating that the gray level values of a group of pixels are very close to one another. Therefore, they all can be represented by a single representative intensity. This argument is valid for both peaks of a bi-modal histogram, hence resulting in two representative intensities for an edge block. These are the block low and high intensities, denoted by I_{low} and I_{high} that correspond to the peaks on the bi-modal histogram of the edge block.

By forcibly clustering all pixels in an edge block into two groups, a bi-level approximation of the block may be obtained. Only two representative intensities and certain binary bits, forming a bit-map are necessary to specify the bi-level representation.

Once the representative intensities of an edge block have been determined, a bit- map may be constructed to specify the correspondence between the pixels and the representative intensities. In such a bit-map, each pixel is represented as a '1' or a '0'. The detailed description is given simply as:

$$B_{i,j} = \begin{cases} 1 & x_{i,j} \le I_{mid} \\ 0 & x_{i,j} > I_{mid} \end{cases} \qquad (1)$$

Where, $I_{mid} = \dfrac{I_{low} + I_{high}}{2}$

Once the bit-map of an edge block has been formed, the block can be coded by finding the best match for its bit-map from a set of patterns in a look-up table. A set of 30 patterns shown in Fig. 2, which preserve the location and polarity of edges in four major directions is used for the pattern matching stage. This process determines the index of the matched pattern selected from the set. The matching score P_{score} of each pattern is calculated as:

$$P_{score} = \sum_{i=0}^{3}\sum_{j=0}^{3} \begin{cases} 1 & if \quad P_{i,j} = B_{i,j} \\ 0 & if \quad P_{i,j} \neq B_{i,j} \end{cases} \qquad (2)$$

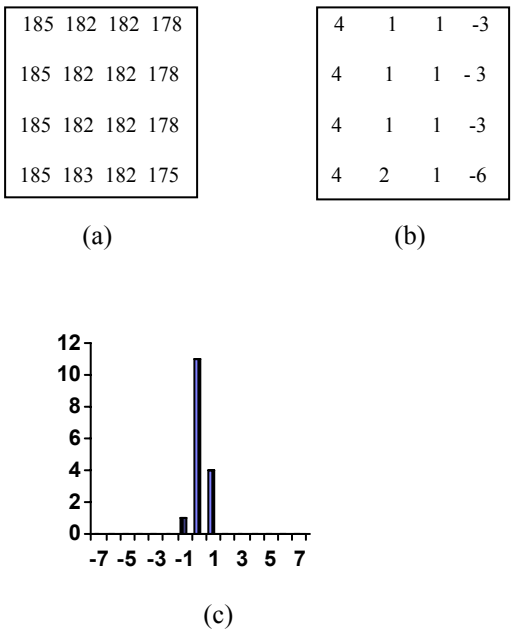

Fig. 1 : (a) a uniform block (b) the block residuals (c) the block histogram

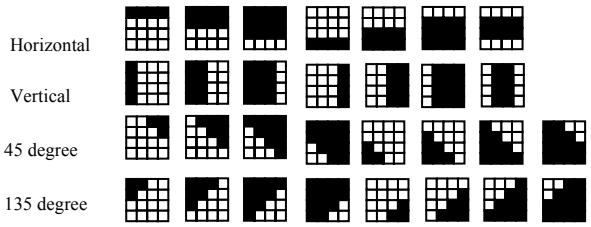

Fig. 2: Set of patterns for the classification

The pattern *p*, which has the maximum score, P_{score} is selected as the pattern with the best match. The transmitted information includes the index P_{best} of the selected pattern in the look-up table as well as the block representative intensities I_{low} and I_{high} for the areas indicated in black and white of the selected pattern, respectively.

During the decoding process, the decoder replaces the pixels of a uniform block by the block mean. Whereas,

in decoding an edge block, the decoder uses the index of the selected pattern as well as the transmitted intensities to reconstruct the block. Reconstruction of an edge block is carried out by replacing the 1's and 0's of the selected pattern by I_{high} and I_{low}, respectively. The number of bits required to code a uniform block, denoted by $B_{uniform}$ is: $B_{mean}+1$ (3)

where B_{mean} is the number of bits required to code the mean intensity of a uniform block and the "1" is the overhead to inform the decoder the block is a uniform block. The number of bits to code an edge block *edge B* is computed as:

$$B_{edge} = 2B_{rep} + \log_2^P + 1 \quad (4)$$

where, B_{rep} denotes the number of bits required to code one of the representative intensity, *P* is the number of patterns used, $\log_2^p$ is the number of bits required to transmit the index of P_{best}, and the 1 is the overhead to inform the decoder the block is an edge block. The overhead information for an *MxN* image size is $\frac{MxN}{4x4}$ bits. The overhead bits form a binary file which will be used later as an ownership file during the extraction of the hidden data.

Assuming that $N_{uniform}$ and N_{edge} are the number of the uniform and edge blocks in an image, respectively then the compression ratio (CR) achievable for an 8-bit grey level image may be determined according to:

$$CR = \frac{(N_{uniform} x B_{uniform}) + (N_{edge} x B_{edge})}{MxN} x \frac{1}{8} \quad (5)$$

3. The Proposed Data Hiding Scheme

This section demonstrates how to embed the secrete bits into a gray level host image and how to extract the data. The whole process can be partitioned into two phased: one is the data embedding phase; the other is the data extraction phase.

3.1. Data Embedding Phase

In the data embedding phase, the host image is compressed using the pattern-based compression algorithm, described in Section 2. An ownership file is constructed according to the type of image blocks. The ownership file is a bit stream where each bit indicates whether the processed block is uniform (bit 1) or an edge (bit 0). For each encoded edge block, the selected pattern or bitmap records the counter information of the block. If any bit in the bitmap is changed, then the image quality of the reconstructed edge block may become poor [8]. To lower the distortion, we embed the secrete data into the pattern of each uniform block rather than edge block. Because the pixel intensities in the uniform block are close to their neighboring pixels, even though the bits in the bitmap are changed, the reconstructed pixel value is still close to its original one.

In the coding algorithm described in the previous section, the pixels of a uniform block are represented by a single value that is the block mean. Therefore the block pattern will not have 1s and 0s. However, in order to embed secrete binary data into uniform blocks , the block mean which is the block reprehensive intensity is tuned to produce two intensities as follows : $I_{mean} \pm \delta$, where δ is a small tuning value. The bitmap of a uniform block is then given by

$$B_{uniform} = \begin{cases} 0 & x_{i,j} \le I_{mean} \\ 1 & x_{i,j} > I_{mean} \end{cases} \quad (6)$$

The sequence of embedded positions are from left to right and then up to down, which is in the row-major order. For the sake of data security, secret data should be encrypted before hiding.

In our scheme, the population of uniform blocks in an encoded image is an important parameter to balance between the embedding capacity and the image quality of the stego-image. A larger population of the uniform blocks provides high embedding capacity but lower image quality of the stego-image, and vice versa. The number of uniform blocks depends on the $Score_{\min}$ of the detected peaks Thus, as we use the block size in *4×4*, the hiding capacity *C* of our scheme is:

$$C = (N_{uniform} \text{ x } 16) \text{ bits} \quad (7)$$

Here, the secret length *L* should be smaller than or equal to *C*.

Data Embedding Algorithm:

Input : A grey-level host image *H* of *N x M* pixels
Total number of secrete bits L.

Output : A compressed image file H' , the ownership file *O* of size $\frac{MxN}{4x4}$.

Step 1. Apply the block classifier to the blocks of on the host image *H*.

Step 2. Construct the ownership file *O* by assigning 1 to a uniform block and 0 to an edge blocks.

Step 3. Count the number of uniform blocks of the host image by counting the 1s in the ownership file *O* to calculate the embedding capacity *C*. If $C < L$, then decrement the $Score_{\min}$ and go back to step 1.

Step 4. For each uniform block, set a block pixel $I_{i,j}$ to '1' if its value is grater than the block mean I_{mean}; otherwise, set it to '0'.

Step 5. Transmit the block mean and also the tuning value δ.

Step 6. Embed the secret bits into the bit map of each uniform block.

Step 7. Repeat the above steps until *L* secret bits are totally embedded.

An example of the embedding procedure is shown in Fig. 3

3.2. Data Extraction Phase

The extraction of data is relatively simpler than the embedding phase. For each encoded block, the corresponding bit in the ownership file is examined. If the bit is a '1' the block is identified as a uniform block containing the secrete bits. The representative intensities of such a block are $I_{mean} + \delta$ and $I_{mean} - \delta$. The uniform block is then transformed into a bit plane. All the secret bits can then be retrieved from the bit plane. The data extracting order is by row-major policy and is and the process is repeatedly executed for all uniform blocks until *L* secret bits are retrieved. The extracted information need to be decrypted to gain the secret data.

Data Extracting Algorithm:

Input : The compressed image file H' and the ownership file *O*

Output : *L* secret bits

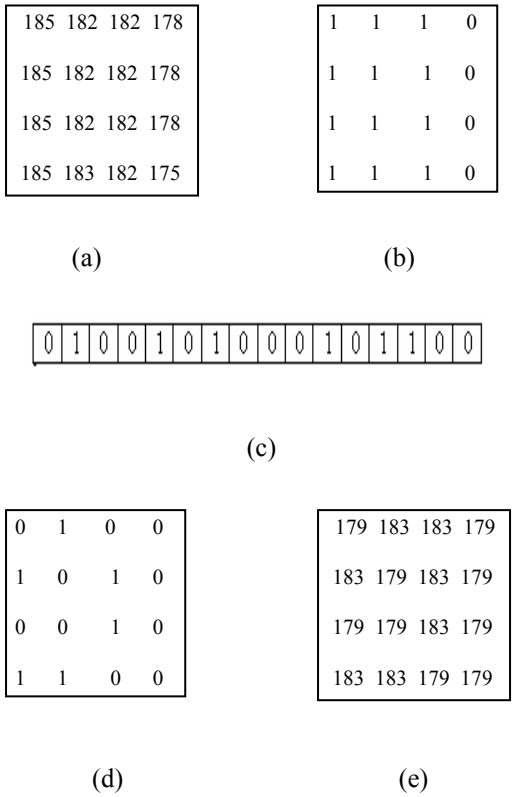

Fig. 3 : (a) original uniform block, mean value :181

(b) block bit map,

(c) the first 16 secret bits

(d) the modified bitmap,

(e) the reconstructed block after embedding the secret bits (mean =181 and $\delta = 2$)

Step 1. Use the ownership file to identify a uniform block

Step 2. Form the bit plane of the processed block by replacing the pixels by 1 , if the pixel value is greater then the block mean or 0 otherwise.

Step 3. If *L* secrete bits are not retrieved yet , go to step 1.

4. Simulation Results

We have evaluated the performance of the proposed coding scheme through a computer simulation. The computer simulation has been carried out, using a set of *256 x 256*, 8-bit intensity, monochromatic standard

images including the images of 'Lena' (Fig.4a) and 'Baboon' (Fig.5a). In the implementation used here, the $Score_{\min}$ was set to 10 and 12, the index P_{best} was given by 5 bits (indicating the use of 30 patterns), and 8 bits were used to transmit each of the intensity values, I_{mean}, I_{low}, and I_{high}.

The stego-image of 'Lena' and its ownership file for $Score_{\min}$, set to 10 are shown in Fig. 5b. The ownership file has a size of *64 x 64* binary bits and demonstrates the effectiveness of the classification scheme, which extracts the uniform and edge blocks.

A steganographic technique is usually evaluated in two aspects ; *Imperceptibility* and *Hiding capacity*. To evaluate the performance of the proposed scheme in terms of quality, we can judge whether the stego-image quality is acceptable to the human eye by using the peak signal-to-noise ratio(PSNR) in order to measure the distortion between the host image and the processed image embedded with secret bits. It is defined as follows :

$$PSNR = 10x\log_{10}\frac{255^2}{MSE}dB,$$

where MSE is the mean-square error defined as :

$$MSE = \frac{1}{MxN}\sum_{i=1}^{M}\sum_{j=1}^{N}(X_{ij} - \overline{X}_{ij})^2$$

Here X_{ij} and $\overline{X}_{ij}$ denote the original and processed pixel values, respectively.

The hiding capacity indicating the maximum number of bits that can be hidden with acceptable resultant stego-image quality depends highly on the identification of the uniform block in the host image, which in turn is the important factor for the compression ratio. Tables 1 and 2 show the compression results, the embedding capacity (EC) and the image quality (PSNR) for the cover images 'Lena' and 'Baboon' for two different $Score_{\min}$ values.

By setting the $Score_{\min}$ to 10, nearly 70% of the blocks of the 'Lena' image where identified as uniform blocks, resulting in a compression ratio of 9.2. For this compression ratio, the compressed-stego image has a size of nearly 7 Kbytes. The total capacity for the stego image was found to be more than 2/3 of the compressed image. The embedded capacity takes into account the transmission of the ownership file, as the ownership file is transmitted by the overheads given in Equations 3 and 4.

Table1

Compression and Embedding Capacity result for Image size of 64 KB at $Score_{\min} = 10$

Image	P_u	CR	C I	EC	PSNR
Lena	69%	9.2 : 1	6.9 KB	5.5 KB	31.74
Baboon	52%	7.8 : 1	8.2 KB	4.2 KB	31.15

Table2

Compression and Embedding Capacity result for Image size of 64 KB (256 x 256) at $Score_{\min} = 12$

Image	P_u	CR	C I	EC	PSNR
Lena	58%	8.3 : 1	7.7KB	4.6 KB	32.24
Baboon	40%	7.8 : 1	8.2 KB	3.2 KB	31.85

P_u : Population of uniform blocks
CR : Compression Ratio
CI : Compressed Image
EC : Embedding Capacity

The results show that, setting a higher value for $Score_{\min}$ will result in less number of uniform blocks being identifies. This in turn leads to lower compression ratio and smaller embedding capacity, but a better image quality. The visual quality of each stego- image at different $Score_{\min}$ is illustrated in Figs. 4b-4c and 5b-5c respectively.

5. Conclusions

In this paper, we have proposed an information hiding scheme to hide secrete data into compression domain of the host image, generated by a pattern-based compression algorithm. The secrete data are embedded in the bitmaps of the uniform blocks of the host image and can be extracted directly without decompressing the stego compressed file. The ownership file which is constructed during the compression phase is the key to extract the information at the receiver end.

6. References

[1] Petitcolas, F.A.P., Anderson, R.J., and Kuhn, M.G.: 'Information hiding – a survey', *Proc. IEEE*, 1999, 87, (7), pp. 1062–1078

Fig. 4(a) : Original Image of Lena (256x256)

Fig. 4(b) : Compressed Stego image at $Score_{min}$ = 10 and its Ownership File (64x64 bits)

Fig 4(c): Compressed stego image at $Score_{min}$=12

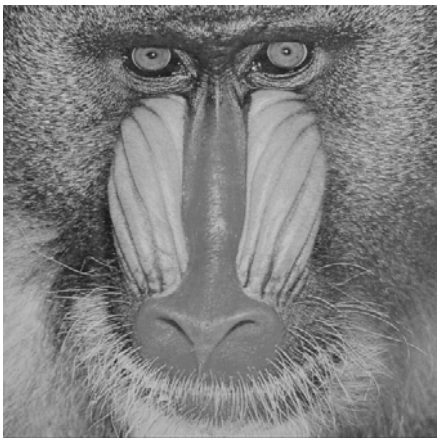

Fig. 5(a) : Original Image of Baboon (256x256)

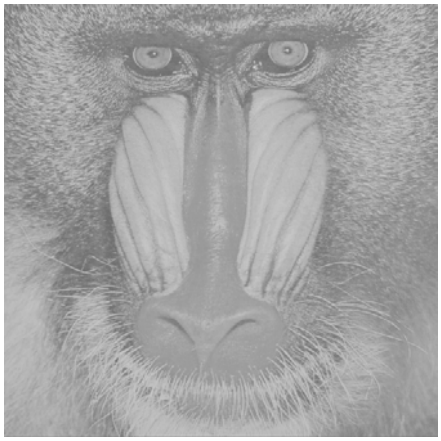

Fig. 5(b) : Compressed Stego image at $Score_{min}$ = 10

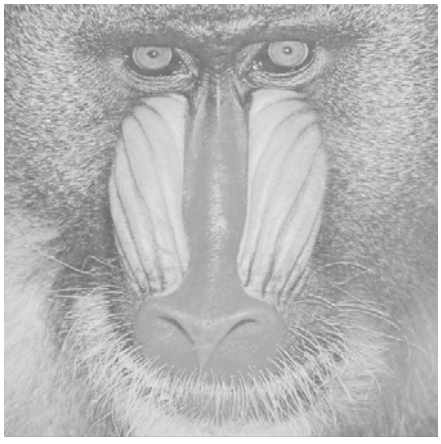

Fig. 5(c) : Compressed Stego image at $Score_{min}$ = 12

[2] Nan-I Wu and Min-Shaing Hwang, "Data Hiding: Current Status and Key Issues", *International Journal of Network Security*, Vol. 4, No.1, Jan. 2007, pp. 1-9.

[3] Bender, W., Gruhl, D., Morimoto, N., and Lu, A.: 'Techniques for data hiding', *IBM Syst. J.*, 1996, 35, (3 and 4), pp. 313–336

[4] Chan, C.K., and Cheng, L.M.: 'Hiding data in images by simple LSB substitution', *Pattern Recognit.*, 2004, 37, (3), pp. 469–474

[5] Bao, P., and Ma, X.: 'Image adaptive watermarking using wavelet domain singular value decomposition', *IEEE Trans. Circuits Syst. Video Technol.*, 2005, 15, (1), pp. 96–102

[6] Chu, W.C.: 'DCT-based image watermarking using sub-sampling', *IEEE Trans. Multimedia*, 2003, 5, (1), pp. 34–38

[7] Chang, C.C., and Wu, W.C.: 'A steganographic method for hiding secret data using side match vector quantization', *IEICE Trans. Inf. Syst.*, 2005, E88-D, (9), pp. 2159–2167

[8] J-C Chuang and C-C Chang, "Using a simple and fast image compression algorithm to hide secret information", *International Journal of Computers and Applications*, Vol. 28, No. 4, 2006, pp. 329-333.

[9] F Keissarian, "Novel quad-tree predictive image coding technique using pattern-based classification, *Proc. SPIE, Visual Communications and Image Processing (VCIP-2003*), vol. 5150, pp. 1481-1490, June 2003, Lugano, Switzerland.