Key-Byte Algorithm for Image Compression

Feras Sameer Alhaijawy
Computer Science and Engineering Department
University Politehnica of Bucharest
Bucharest, Romania
feras.sameer@gmail.com

Abstract—**Image size is increasing every day, according to their accuracy and resolution; where the memory needed to store these images become larger with time, Image compression algorithms are the utilities used to utilize the memory resources; this is done by reducing the space needed to store these images, natural images are the best images to use in testing the compression images; since the adjacent pixels are close together but not 100% equal.**

The paper presents a compression algorithm that constrained on the compression factor, where it tries to get the highest compression factor with an accepted visual contribution to the reconstructed image, the algorithm works in two stages the first one called the Averaging technique, which compresses the image by converting the image into a metric of averages, the second stage is called the Key-Byte technique, in this stage the output of the first stage divided into blocks and extract three new values for each block those are Average, MinMax, and Key-Byte values, this means three new metrics one holds the averages of the blocks, another one to holds the difference between the maximum and minimum values in each block, the third holds 0 or 1 according to the relation between the original average and the new averages .

Keywords— Lossy, Compression, Compression Factor, Structural Similarity, Mean Square Error, Compressed Image.

I. INTRODUCTION

A high-resolution image occupies a huge space of memory, in addition to, the high transmission time over a network [1,2] ; So for an efficient use of the storage in an imaging system, besides a faster transfer of images through the communication channels, computer scientists try to develop an image compression algorithms that ensure reducing the transmission speed and occupying smaller space, Image compression algorithms divided into two types Lossy algorithms and Lossless algorithms, classified according to the relation between the original image and the reconstructed one, where in Lossless algorithms; the reconstructed image and the original image are identical, in the contrast to Lossy algorithms that Suffer from details loss in the reconstructed image.

The paper proposes a lossy algorithm called Key-Byte gives good compression factors as a separate algorithm and works by summarizing the image details into three new related tables, in addition, to get higher compression factors the algorithm uses the averaging technique as a previous stage, which causes more distortion in the reconstructed image.

The paper reviews some compression algorithms from literature in (Section II), (Section III) presents the Key-Byte algorithm, in (Section IV) proposes how Averaging combined with Key-Byte algorithm, results shown in (section V), Discussion stated in (Section VI), while comparison presented in (Section VII), Related works mentioned in (Section VIII), and conclusion stated briefly in (Section IX).

II. BACKGROUND

Many of image compression algorithms were developed in the recent decades [3], some of those algorithms reconstruct the image to be identical to the original image, most of these algorithms depend on pixel's values redundancy to reduce the image size [4] and other algorithms depend on ignoring boring or unnoticeable background details [5].

image compression efficiency based on several factors, the most fundamental factor is Compression Ratio that shows the relation between the Original image and the compressed image as shown in (1), Lossy algorithms provide a larger compression ratio; due to losing some of the original image information [9].

$$CompressionRatio = \frac{C}{O} \quad (1)$$

The Compression Ratio is a percentage represents the change of size between the compressed image and the original image, where C is the size of the compressed image over O which is the size of the original image.

The compression factor is another factor to measure if a compression was done, the compression factor is the inverse of compression ratio [10], the decompressed image quality depends mainly on the compression ratio, where higher compression ratio and smoother original image means better-decompressed image [11]. Keeping in mind the compression factor, we have to get higher compression factor with a visually acceptable reconstructed image.

The compression algorithm must preserve the balancing relation between the decompressed image quality and the compression factor, in other words, the algorithm must keep the compression factor as high as possible with acceptable reconstructed image quality to adopt it.

Some measurements are used to judge the closeness between the original image and the reconstructed one, those measurements work like the visual verdict of the eye but in a mathematical way [12], the closeness between the original image and decompressed image Identifies the algorithm performance, the distortion is the measurement used with lossy Algorithms because in lossless algorithms the Original images and the decompressed images are identical with zero distortion.

The Mean Square Error (MSE) that shown in (2) is an average measure, which is used to measure the distortion in the reconstructed image, by using this measure the average of the square of the error is known and shows the difference between the original image and the decompressed image [10].

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (P_i - Q_i)^2 \quad (2)$$

Where P_i is the pixel values for the decompressed image, Q_i is the pixel values for the original image, i= 1,2,…., N and N is the number of pixels in the image, and the difference between P and Q is the error value.

Because MSE doesn't give a reliable measure of the decompressed image quality [13], which doesn't agree with the human eye perception, thus the best way to evaluate the quality of the image is the visual experiments, which is costly time to consume [14]. Equation 3 calculates the Structural Similarity (SSIM) which is a new measurement that uses structural information for both images (the original one and the decompressed), the structural information is "those attributes that represent the structure of objects in the scene" [15].

$$SSIM = \left(\frac{2\mu_o\mu_x + C1}{\mu^2_o + \mu^2_x + C1} \right) \left(\frac{2\sigma_o\sigma_x + C2}{\sigma^2_o + \sigma^2_x + C2} \right) \left(\frac{2\sigma_{ox} + C3}{\sigma^2_o \sigma^2_x + C3} \right) \quad (3)$$

Where μ_o is the mean of the original image, μ_x is the mean of the decompressed image, σ_o is the standard deviation of the original image, σ_x is the standard deviation of the decompressed image, C positive constants used to avoid null denominator.

The paper uses the Averaging procedure which was used in [17] as a previous stage to the Key-Byte algorithm. The averaging works by dividing the image into blocks each block size is n*n pixels, a procedure calculates the average for each block using (4), the averages are stored in a new metric that represents a compressed image that can be reconstructed by generating random values for each pixel in a block to be close to this block average, Fig.1 shows how the averaging technique works and how it reduces a block of pixels into one value which is the average of the pixels values in that block.

$$Avg = \frac{\sum P}{n^2} \quad (4)$$

Where Avg is the average for each 3*3 block, P is each pixel value in the block, after finding the summation of all pixels in the block the algorithm divides it over n^2 which is the block size, in our case the block size is 9.

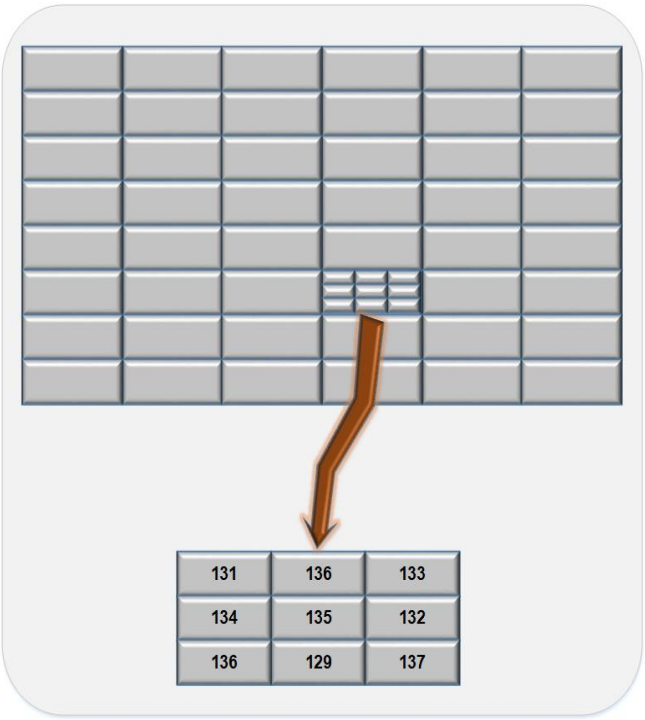

Fig.1. Averaging technique

III. KEY-BYTE TECHNIQUE

The paper represents the key-byte technique as a new compression algorithm. The algorithm abbreviates the original Image in three tables, the algorithm starts by dividing the image into 3*3 blocks, and then the algorithm calculates the average for each block and the difference between the largest pixel value and the smallest one, and records the relation between the original pixel value in the block and the average of the block. Each of the previous values is saved in a separate table. The produced tables represent the compressed image, and those tables are:

A. Average Table

After dividing the image into blocks, the average for each block is calculated and stored in this table to use it in creating the Key-Byte values.

B. MinMax table

This table stores the MinMax value which is the difference between the highest pixel value and the lowest pixel value in each block, the MinMax is used to reduce the gap between the original image and the reconstructed image due to the random way used in reconstructing the image.

C. Key-Byte table

Key-Byte table is derived from the averaging metric (3*3 blocks) and it needs to be passed to the reconstruction phase. Each Key-Byte cell consists of 8 bits, every bit shows the relation between the pixel value in the 3*3 block in the original

image and its average, it begins with the first pixel in the block with the leftmost bit that exist in the first field of the Key-Byte table, if the pixel value is equal or larger than the average, the algorithm sets the corresponding bit, and resets it if the pixel value smaller than the average. The algorithm applies this step till all bits in Byte-Key value is reconstructed, to save the size of 8 bits for each 3*3 block, the algorithm omits the center pixel value in the block, this will not effect on the reconstruction significantly, because the algorithm deals with natural images which mean convergent values for adjacent pixels. And this process is not like what used in LBP, which uses the center pixel's value and compare it with its neighbors.

Fig. 2 shows an example of how a block converted to its key-byte value known that its average is 134. And the min-max value for the block in this figure will be 8.

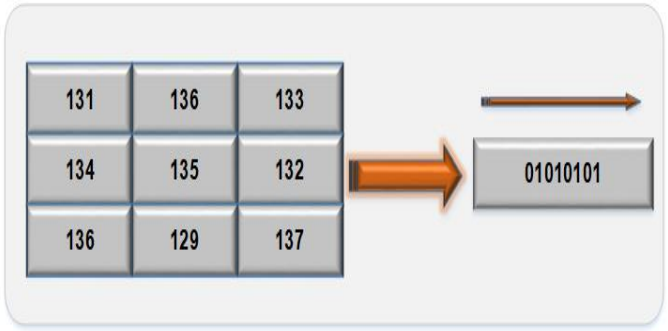

Fig.2. Block Key-Byte

All the three tables are important to reconstruct the image; because using them together will reduce the bias between the reconstructed pixel value and the original pixel value.

Decompression the image is done by taking one cell from each table (i.e. taking the first cell of each table the second cells and so on), each corresponding cells helps to reconstruct 3*3 block, reconstructing the pixels starts by taking the eight bits in the cell taken from the Key-Byte, begins from the left most bit if its value is one the value of the reconstructed pixel value will be the value taken from the average table and adding a random number between zero and the value taken from the MinMax table, but if the bit value is zero, then the pixel value calculated by subtracting the random number from the value in the average table, this procedure repeated each time for each cell in the Key-Byte table. Equation 5 shows how exactly the procedure for calculating the decompressed pixel works.

$$v = \begin{Bmatrix} 0 \rightarrow avg[i] - rnd(MinMax[i]) \\ 1 \rightarrow avg[i] + rnd(MinMax[i]) \end{Bmatrix} \quad (5)$$

Where v is the reconstructed pixel value, avg is the average value taken from the average table, and rnd is a random number between zero and the value taken from the MinMax table, i represents the index of the cells used to reconstruct the 3*3 block, and this index is the same of the three tables.

The previous procedure creates eight pixels values for each 3*3 block, while the value of the middle pixel takes the value of the average, this will not be noticed because the algorithm works on grayscale images, this means the pixels values are very close to each other. The number of blocks created is equal to the number of the cells in one of the tables.

IV. AVERAGING AND KEY-BYTE ALGORITHM

The Key-Byte algorithm will take the compressed image produced by the averaging algorithm instead of the original image, in this way the compression ratio will be increased. But the loss of the image information will occur in two stages, and the reconstructing algorithm can't reconstruct the exact pixel values for the original image.

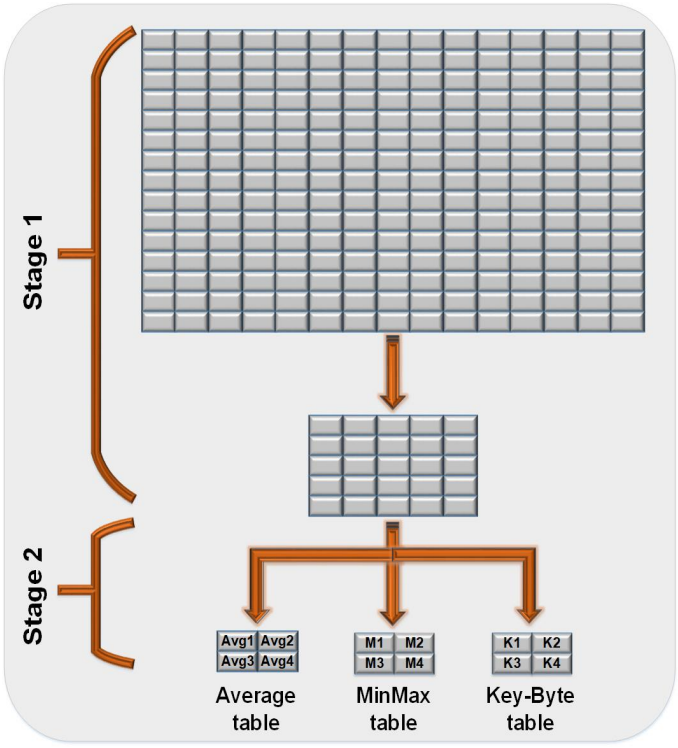

Fig.3. Averaging and Key-Byte algorithm

Fig. 3 shows that in the first stage the averaging procedure is applied and compresses the image producing a new array (table) by dividing the original image into blocks and calculates the average for each block, these averages are passed to the Key-Byte algorithm as a second stage, the process of applying the (averaging and Key-Byte) algorithm works according to the following steps:

- The $(N \times N)$ grayscale Image is divided into (3*3) blocks where each pixel value is in the range [0-255].
- Finding the averages of each block and saving them as (N/3 *N/3) metric, then pass the metric to the next stage.
- In this stage Average and MinMax tables are created first, to use the average values in Key-Byte table creation.
- Key-Byte table is now created according to the average table as shown in the previous section, every pixel in the block is compared with the average of that block and set

the leftmost bit of the first byte if the pixel value is equal or larger than the average, otherwise it will reset the bit and continue doing this till it finishes the pixel of the block and achieves the rightmost bit in the same cell.

V. RESULTS

The Key-Byte algorithm abbreviates the image in three tables the size those table cuts one-third of the original image size, moreover, to improve this compression the averaging procedure is used before applying the Key-Byte algorithm. The overall compression size calculated in the two stages, the averaging technique compression size after adopting 5*5 block size equals to 1/25 from the original image size this means every 25 pixels are represented by one pixel. In the Key-Byte algorithm, each 3*3 block represented by 1 value in each table of the three created tables (average, minmax, and key-byte), thus every 9 pixels represented in 3 values each one of them is a one-byte size, and the compression will be 1/3. If the two algorithms merged, then the overall compression equals to 1/25 in the first stage and 1/3 on the second stage to give a compression equal to 1/75.

The averaging algorithm applied to 3*3 block size, and on 7*7 block size, where the compression factor increases each time the block size increased; but when applying the Key byte algorithm as a second stage, the 5*5 block gave the best results according to MSE and the SSIM measures, Table. I shows a comparison between the three block sizes when using the averaging procedure as a first compression stage, and according to the result the 5*5 block size, gave the best compression factor, Fig. 4 shows an example of a reconstructed image compressed using the proposed algorithm, as shown in the figure, all the images present a considerable distortion, but at the same time the main detail of the image is clear and the image can be recognized.

TABLE I. VARIOUS BLOCK SIZE RESULTS

	3*3 block size		5*5 block size		7*7 block size	
Image	***SSIM***	***MSE***	***SSIM***	***MSE***	***SSIM***	***MSE***
Boat	0.3768	0.0931	0.5582	0.0554	0.4531	0.1052
Lena	0.4788	0.0802	0.5695	0.0558	0.4850	0.1044
Barbara	0.4019	0.0931	0.5033	0.0781	0.4135	0.1197
Cameraman	0.5510	0.1229	0.6426	0.0890	0.5800	0.1355
Mandrill	0.1311	0.1330	0.2536	0.0934	0.1879	0.1269

1. The result in the table calculated using MATLAB

VI. DISCUSSION

Combining averaging technique and Key-Byte technique, produced a compressed image with a very high compression factor, but on the other hand, it adds a clear distortion of the reconstructed image, the distortion size accepted or rejected according to the application going to use these reconstructed images, such that some application concentrate on the objective of the image, but other applications such as those used in medical industry needs to restore the original image [16].

In this paper, 5*5 block size is adopted for averaging technique, where the overall compression factor and visual aspects are the best when compared with other block sizes, the compression factor (R) calculated using (6):

$$R = R_{(A)} + R_{(K)} \tag{6}$$

Where R is the total compression factor which calculated by adding $R_{(A)}$ that is the Averaging compression factor to $R_{(k)}$ that represents the Key-Byte compression factor.

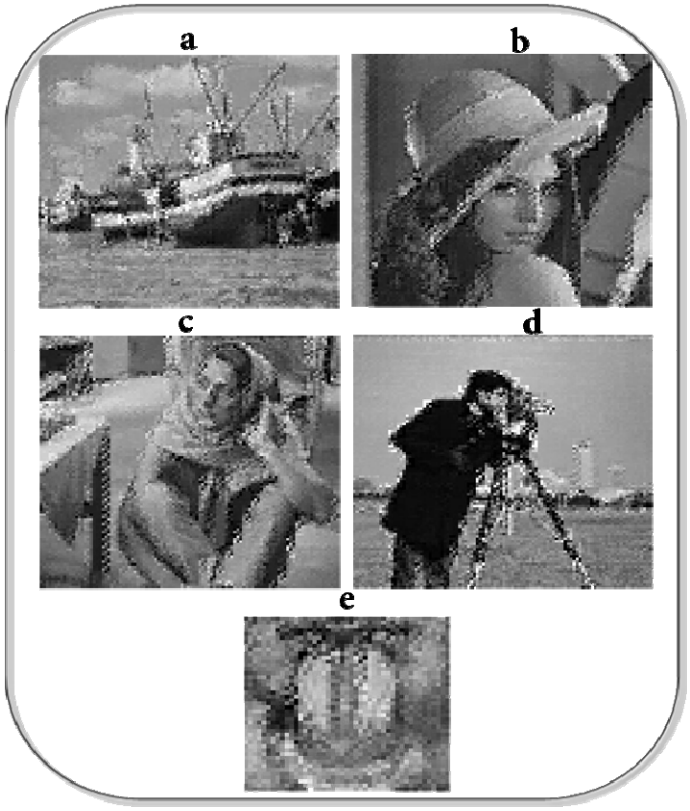

Fig. 4. Averaging with Key-Byte using 5*5 a) Boat, b) Lena, c) Barbara, d) Cameraman and e) Mandrill

Fig. 5 shows the huge differences between the original image size and the compressed image size after compressing those images using average and Key-Byte Algorithm, Fig. 6 shows the results using the MSE and SSIM for the reconstructed images which are mathematical representation that give an idea of how much the decompressed images are close to the original images.

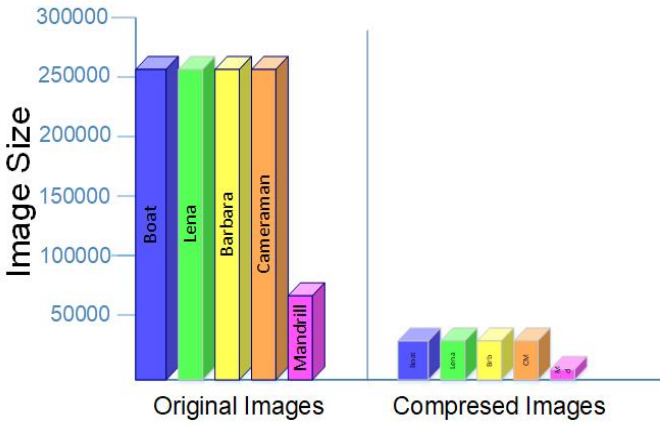

Fig. 5. averaging and Key-Byte compressed size

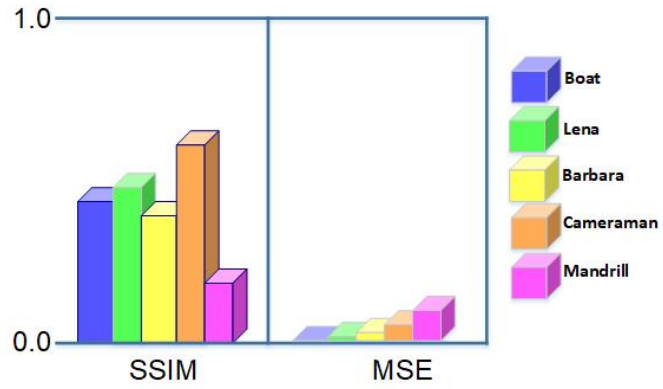

Fig.6. averaging and Key-Byte SSIM, and MSE

VII. COMPARISON

The averaging with Key-Byte algorithm gives a new way to compress the images, and it provides a new compression factor record, as shown in Fig. 7 the compression factor produced using this algorithm is 12 times larger than any compression factor produced using any well-known algorithms, the reason of this large difference refers to using the averaging procedure tow time one as a first stage before the Key-Byte algorithm and another time during the Key-Byte algorithm to create the average table.

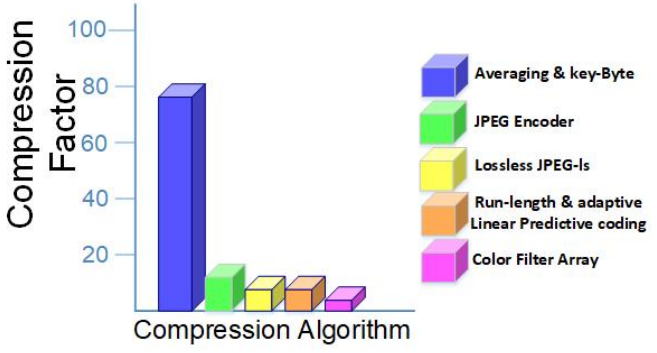

Fig.7. Comparison between different techniques.

All the compression algorithms shown in the Fig. 7 are applied in grayscale images and as shown the compression factor for all of them is less than 20%, while the averaging and Key-Byte algorithm exceeded the 75%, but this high compression factor is on the account of the decompressed image quality.

VIII. RELATED WORK

One of most related work is a compression algorithm passed on LBP (Local Binary Pattern) which explained in [18], and it works by dividing the image into blocks, then in each block the algorithm comparing each center pixel with its neighbors and moves in a circular pattern to cover all of its surrounding pixels if the center pixel is larger than all of its neighbors zero is recorded otherwise one is recorded, the main differences between LBP and Key-Byte are in Key-Byte the comparison is done between each pixel value and the average of the block, the second difference is the pixels taken from the block row by row not in a circular way.

IX. CONCLUSIONS

The paper presents a new Lossy compression algorithm with very high compression ratio, the main contribution of this algorithm is to increase the compression factor with getting an acceptable visual aspects of the reconstructed images – minimum distortion -, these two factors used to judge the algorithm, moreover, the SSIM gives an idea that how close is the reconstructed image to the original image.

The algorithm succeeds to reach a very high compression factor and decompress the image successfully which is the main advantage of it, even if there is some data loss, another advantage of it is that despite the noticeable distortion in the reconstructed image the user still can recognize the main details of it, while the main drawback of this algorithm is the large distortion which tied the ability to use it to certain fields like space exploring and studying the topography of a planet.

In future more work on the key-byte table will be done by increasing the number of bits in each cell to sixteen bits and use the extra bits to have more accurate reconstructed pixel, by constraining the random number that added to the average, this constraint will decide if the original pixel more than the average plus half of the minmax value or less, which will minimize the distortion of the reconstructed image.

REFERENCES

[1] M.Asif Ali, Aftab khan, M.Younus javed, Aasia Khanm. And S. Jones, "Lossless Image Compression Using Kernal Based Global Structure Transform (GST)". In Proc. 2010 IEEE ICET. Digit. Conf., pp.170-174.

[2] Võ, D.T., Yeong-Taeg Kim, "Low line memory visually lossless compression for color images using non-uniform quantizers,".2011 IEEE Trans. Consumer Electronics, vol. 57, pp. 187-195.

[3] J. L. Nunez and S. Jones, "Run-length coding extensions for high performance hardware data compression". In Proc. 2003 IEEE Comput. Digit. Tech Conf., pp. 387-395.

[4] J. F. Kennedy." Random projection and orthonormality for lossy image compression". Image and Vision Computing, vol.25, pp.754-766, 2007.

[5] Brewer, N. Lei Wang; Nianjun Liu; Li Cheng "User-Driven Lossy Compression for Images and Video". Proc. 2009 IVCNZ Conf., 346-351.

[6] Dung Trung Vo, Yeong-Taeg Kim" Visually lossless compression for color images with low line memory requirement using non-uniform quantizers" 2011 IEEE ,ICCE Conf. 639 – 640.

[7] Yong Zhang, Adjeroh, D.A.;"Prediction by Partial Approximate Matching for Lossless Image Compression". 2008 Trans. Image Processing; Vol.17 ,page.924-935.

[8] Hua Li; Yiming Zhu; "Lossless Image Compression Based on DPCM-IWPT". IEEE, ISECS .vol.1, page. 157 - 160.

[9] Ponomarenko, N., Lukin, V. ; Egiazarian, K. ; Delp, E. "Comparison of lossy compression performance on natural color images" . Proc.2009 PCS conf.,1-4.

[10] I. M. Pu. (2006). Fundamental data compression. 1st Ed.

[11] Xinpeng Zhang."Lossy Compression and Iterative Reconstruction for Encrypted Image", IEEE Trans. INFORMATION FORENSICS AND SECURITY, VOL. 6, NO. 1, MARCH 2011.

[12] Beghdadi, A.;" DESIGN OF AN IMAGE DISTORTION MEASURE USING SPATIAL/SPATIALFREQUENCY ANALYSIS," Communications and Signal Processing, 2004. Conf. pp 29-32.

[13] Almohammad, A.; Ghinea, G. ;" Stego image quality and the reliability of PSNR," IPTA 2010,Conf. pp 215-220.

[14] Dumic, E.; Bacic, I. ; Grgic, S. ;" Simplified structural similarity measure for image quality evaluation,". IWSSIP 2012,Conf. pp 442-447.

[15] Zhou Wang ; Bovik, A.C. ; Sheikh, H.R. ; Simoncelli, E.P. "Image quality assessment: from error visibility to structural similarity," IEEE Trans. Image Processing,Vol.13 , pp.600-612,April.2004.

[16] Lukin, V.V. ; Zriakhov, M.S. ; Ponomarenko, N.N. ; Krivenko, S.S. ; Miao Zhenjiang ; " Lossy compression of images without visible distortions and its application," , Proc., ISCP 2010 Conf. pp 698-701.

[17] P. J. Burt, E. H. Adelson, "The Laplacian pyramid as a compact image code", IEEE Trans. Commun., vol. COM-31, pp. 532-540, Apr. 1983.

[18] Szoke, Ildiko-Angelica, Diana Lungeanu, and Stefan Holban. "Image Compression Techniques Using Local Binary Pattern." Applied Machine Intelligence and Informatics (SAMI), 2015 IEEE 13th International Symposium on. IEEE, 2015.