

Image Compression using Fourier Transformation with Genetic Algorithm

Mohammed Mustafa Siddeq - Software Engineering Depart.
Technical College/ Kirkuk –Iraq

Abstract:

This paper introduce proposed algorithm consist of: (1) using Discrete Fourier Transformation (DFT) which convert an image into frequency domain image, then compress frequency domain image with Run-Length-Encoding (RLE), and Arithmetic Coding. (2) Apply Inverse Fourier Transformation (IDFT) to obtains an approximately original image, and then compared with original image to get the difference stored in a new matrix called ($D_{(spatial)}$). The matrix $D_{(spatial)}$ transformed to frequency domain by Discrete Cosine Transform (DCT) ($D_{(frequency)}$). Finally applying weight vector ($W = [0.5, 0.3, 0.2]$) on the $D_{(frequency)}$, multiply "W" with each three coefficients from matrix $D_{(frequency)}$ to produce a new matrix (G), at last compress matrix G by arithmetic coding. (3) The decompression process start from arithmetic decoding to return frequency domain matrix (i.e. return DFT image), then apply Inverse DFT to get an image A, also from arithmetic decoding produced matrix G. The Genetic Algorithm used to produce minimized matrix $D_{(frequency)}$ by take each data from matrix G and using fitness function. Finally apply inverse DCT to generate matrix $D_{(spatial)}$, added with image A to produce a decompressed image. In this paper our approach, compared with JPEG technique, by using Peak Signal to Noise Ratio (PSNR).

الخلاصة:

نقدم في هذا البحث طريقة جديدة للكبس الصور وتتكون هذا البحث من 1- "Discrete Fourier Transformation (DFT)" والتي تحول الصورة الى مجال ترددي. وبعد ذلك نستخدم "Run-Length-Encoding (RLE)" و "Arithmetic Coding" لكبسها. 2- نطبق عملية عكسية للصورة المحولة عن طريق "Inverse DFT" للحصول على صورة مشابهة الى الصورة الأصلية وبعد ذلك نقوم بحساب الفرق بينها وبين الصورة الأصلية وهذا الفرق يخزن في مصفوفة جديدة تسمى ب " $D_{(spatial)}$ " ويحول هذه المصفوفة الى مجال ترددي عن طريق "Discrete Cosine Transform (DCT)" ويسمى المصفوفة المحولة الى " $D_{(frequency)}$ ". وأخيرا نقلص حجم المصفوفة بأستخدام متجه " $W = [0.5, 0.3, 0.2]$ " بواسطة هذه المتجه يتقلص حجم المصفوفة الى ثلث الحجم وتسمى هذه المصفوفة ب " G " ومن ثم نستخدم "Arithmetic Coding" لكبس المصفوفة الأخيرة. 3- في عملية فك التشفير نسترجع المصفوفات " G " و " $D_{(frequency)}$ ". نستخدم العملية العكسية للحصول على مصفوفة " A "، وتكتمل عملية فك التشفير بفك مصفوفة " G " عن طريق تقنية "Genetic Algorithm (GA)" وبأستخدام العملية العكسية نحصل على مصفوفة فرق " $D_{(spatial)}$ ". ثم ندمج " $A + D_{(spatial)}$ " للحصول على صورة. وفي هذا البحث قارنا طريقتنا مع تقنية "JPEG" بأستخدام "PSNR".

Keywords: Discrete Fourier Transformation, Discrete Cosine Transformation, Genetic Algorithm

1. Introduction

Since the mid-80s, members from both the International Telecommunication Union (ITU) and the International Organization for Standardization (ISO) have been working together to establish a joint international standard for the compression of grayscale and color still images. This effort has been known as JPEG, the Joint Photographic Experts Group the “joint” in JPEG refers to the collaboration between ITU and ISO[1,2,4,11]. Officially, JPEG corresponds to the ISO/IEC international standard 10928-1, digital compression and coding of continuous-tone (multilevel) still images or to the ITU-T Recommendation T.81[1,2,4]. The text in both these ISO and ITU-T documents is identical. The process was such that, after evaluating a number of coding schemes, the JPEG members selected a DCT1-based method in 1988. From 1988 to 1990, the JPEG group continued its work by simulating, testing and documenting the algorithm. JPEG became a Draft International Standard (DIS) in 1991 and an International Standard (IS) in 1992 [1-3]. Lossy compression is compression in which some of the information from the original message sequence is lost. This means the original sequences cannot be regenerated from the compressed sequence. Just because information is lost doesn’t mean the quality of the output is reduced[2,4,11]. For example, random noise has very high information content, but when present in an image or a sound file, we would typically be perfectly happy to drop it. Also certain losses in images or sound might be completely imperceptible to a human viewer (e.g. the loss of very high frequencies). For this reason, lossy compression algorithms on images can often get a factor of 2 better compressions than lossless algorithms with an imperceptible loss in quality. However, when quality does start degrading in a noticeable way, it is important to make sure it degrades in a way that is least objectionable to the viewer (e.g., dropping random pixels is probably more objectionable than dropping some color information). For these reasons, the ways most lossy compression techniques are used are highly dependent on the media that is being compressed[2]. Lossy compression for sound, for example, is very different than lossy compression for images.

2. Image Compression algorithm

In this paper we introduce an idea for image compression with Discrete Fourier transformation (DFT), and Discrete Cosine Transformation (DCT). The first step in image compression using DFT; consist from real and imaginary part, divide each part into by a value, then compress each part by using Run-Length-Encoding (RLE) and Arithmetic coding algorithm. The difference between Inverse DFT for a frequency domain and original image, is be used by DCT (i.e. Convert the difference into frequency domain matrix by DCT). The weight vector introduced in this paper used to minimize the frequency domain matrix to be compressed by Arithmetic coding algorithm, Figure -1 shown the proposed algorithm.

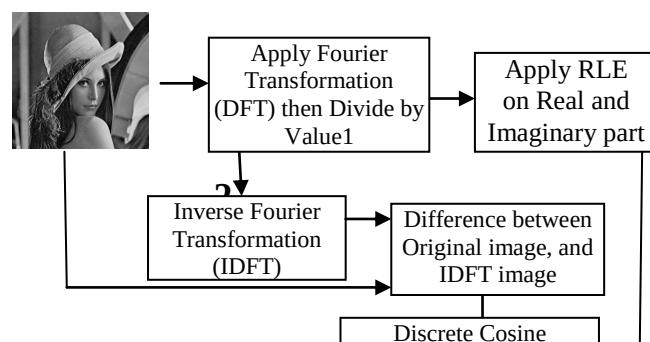


Figure – 1 Image Compression Algorithm

2.1 Using Discrete Fourier Transformation (DFT)

The following equations are represents two-dimensional DFT and Inverse DFT [1,2]:-

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) * [\cos(2\pi (ux/M + vy/N)) - j \sin(2\pi (ux/M + vy/N))] \dots\dots\dots(1)$$

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) * [\cos(2\pi (ux/M + vy/N)) + j \sin(2\pi (ux/M + vy/N))] \dots\dots\dots(2)$$

Where f(x,y) is represent image matrix at spatial domain, and F(u,v) is represent image at frequency domain consist from real, and imaginary parts[1,2]. Divide each part on a value, the idea for dividing on a value increasing number of zero's, this lead to increase compression ratio, as shown in Figure – 2. The RLE and Arithmetic Coding convert each part into stream of bits.

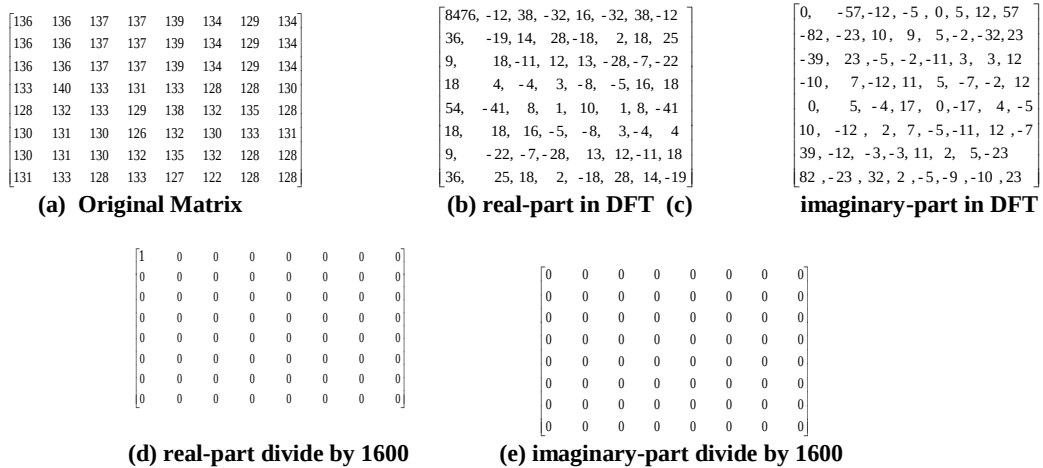


Figure – 2 (a-e) matrix 8x8 converted into DFT and divided on value=1600

Mohammed Mustafa Siddeq

In Figure – 2 (d,e) the matrices used by RLE, the basic idea is to identify numbers of adjacent data of equal value and replace them with a single occurrence along with a count. In above example, the numbers sequence in the real-part=(1,0,0,0....0), and imaginary-part=[0,0,0,...0] could be transformed to (1,1), (0,63), (0,64). Once transformed, a probability coder (*e.g.*, Arithmetic coding) can be used to code both the data and the counts[1,2,11,12].

2.2 Using Discrete Cosine Transformation (DCT)

After divide DFT on the value1=1600, we use the Inverse DFT to get approximately original image then the difference between original image and approximately original image (i.e. Difference = Original image - Inverse DFT) stored in the new matrix called Matrix $D_{(spatial)}$. This matrix converted into frequency domain (Matrix $D_{(frequency)}$) using DCT, the following equations shows DCT, and Inverse DCT[1,2,4,11]:-

$$C(u, v) = a(u) a(v) \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) \cos \left[\frac{(2x+1) u \pi}{2N} \right] \cos \left[\frac{(2y+1) v \pi}{2M} \right] \dots\dots\dots (3)$$

$$\text{Where } a(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u \neq 0 \end{cases}$$

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} a(u) a(v) C(u, v) \cos \left[\frac{(2x+1) u \pi}{2N} \right] \cos \left[\frac{(2y+1) v \pi}{2M} \right] \dots\dots\dots (4)$$

The following figure shown the difference between original matrix, and Inverse DFT for the image converted into DCT.

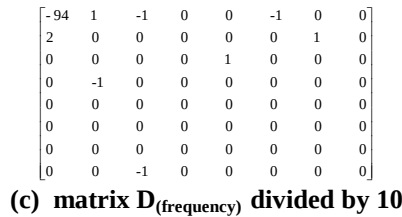
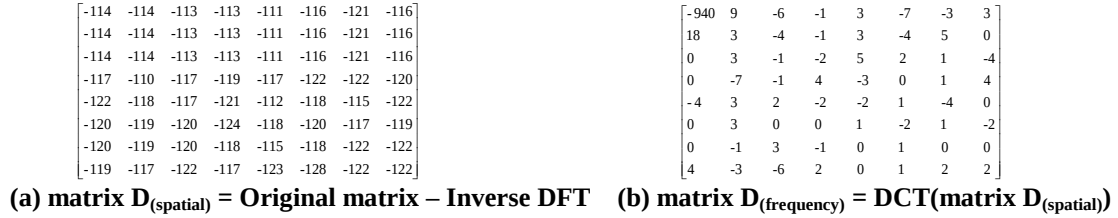


Figure – 3 (a-c) Converting the matrix $D_{(spatial)}$ into frequency domain by DCT

The DCT is similar to the DFT: it transforms an image from the spatial domain to the frequency domain. One of the advantages of DCT over DFT is the fact that it is a real transform, whereas DFT is complex[2,4,11]. This implies lower computational complexity, which is sometimes important for real-time applications [3,4,9,11,12]. The discrete cosine transformation is used to decorrelate the pixels of image or to pack as much information as possible into

the smallest number of transform coefficients (See Figure -3(b)). The coefficients for the matrix $D_{(\text{frequency})}$ divided by a value to convert most of coefficients to zero's (See Figure-3(c)). The main idea for using DCT to reduce the matrix size (i.e. eliminate part of the matrix coefficients from high frequency domains) by eliminate last two columns and last two rows (See Figure – 4(a,b)).

Finally we use weight vector multiply with each three coefficients from minimized matrix $D_{(\text{frequency})}$, to produce the new matrix G. The weights vector containing floating point numbers, and total of weights equivalent to one. Assume the weights values $W = [0.5, 0.3, 0.2]$, used for compression and decompression. The idea of weights values is similar to (Mask Filter 3x3) used in image enhancement and image restoration, and total of Mask Filter 3x3 equivalent to one[7-10]. The matrix G converted into stream of bits by Arithmetic coding,

List -1

$W = [0.5, 0.3, 0.2];$

For i=1 to Row

 J=1; v=1

 While (j<=Column)

$$G(I, v) = \sum_{k=0}^2 W[k] * \text{Minimized_Matrix_D}_{(\text{frequency})}(i, j + k)$$

 v=v+1;

 j=j+3;

 End; // while

End; // for

The Figure -4 illustrates matrix G generated from matrix $D_{(\text{frequency})}$, and weight vector.

$$\begin{bmatrix} -94 & 1 & -1 & 0 & 0 & -1 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(a) Matrix $D_{(\text{frequency})}$

$$\begin{bmatrix} -94 & 1 & -1 & 0 & 0 & -1 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(b) Delete part from matrix $D_{(\text{frequency})}$ (G)

$$\begin{bmatrix} -46.9 & -0.2 \\ 1 & 0 \\ 0 & 0.3 \\ -0.3 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

c) Multiply weight vector produce Matrix

$$\begin{bmatrix} -469 & -2 \\ 10 & 0 \\ 0 & 3 \\ -3 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

(d) multiply each coefficients with 10

Figure- 4 (a-d) Matrix G generated from Matrix $D_{(\text{frequency})}$ and Weight Vector.

3. Decompression by using Genetic Algorithm

The genetic algorithm is a parallel search algorithm by using a number of strings and computing fitness value for each string, these strings are shared with each others by crossover until reached to the result [13,15]. In this section we will explain how the genetic algorithm reached to the original coefficients in the matrix $D_{(frequency)}$. The genetic algorithm reverse the operation to find matrix $D_{(frequency)}$ by using weight vector $= (0.5, 0.3, 0.2)$ and matrix G, the genetic algorithm generate lost coefficients by using fitness function and crossover. The following equation represents fitness function:

$$\text{Fitness function}(i) = \begin{bmatrix} 0.5 \\ 0.3 \\ 0.2 \end{bmatrix} * [X1, X2, X3] \quad (5)$$

Where $i = 1, 2, 3, \dots$ matrix G size

The genetic algorithm generates strings, and each string consists from 3 elements, because the weight vector size is 3. The genetic algorithm string is integer numbers, these numbers ranges depend on the matrix $D_{(frequency)}$ content, for example in Figure-4(b) the matrix $D_{(frequency)}$ probability data: -94, 1, -1, 0, 2, and the genetic algorithm depend on this probability to generate strings, and genetic algorithm search for the string depending on the fitness function compared with matrix G.

These strings are generated randomly consist from integer numbers depend on the probability of data; this leads the genetic algorithm, to find matrix $D_{(frequency)}$ coefficients [13,14,16]. The crossover and fitness function play an main rule in genetic algorithm, the genetic algorithm search for X1, X2, and X3, depending on the fitness function is described at Equation (5) it is used for comparison with matrix G coefficients (i.e. fitness values matched with matrix G coefficients). While the crossover its meaning exchange between any two strings randomly, and this operation done by selecting an element randomly from the strings, then make exchange between them. For example assume we have the following two strings and then make exchange between them [13-16]:

The genetic algorithm generates 25 strings for the probability data: -94, 1, -1, 0, 2 used in Figure-4(b), the data are used distributed randomly in 25 strings. The following example illustrated genetic algorithm generates matrix $D_{(frequency)}$ coefficients:

Strings represents

Matrix $D_{(frequency)}$	Fitness Function
S1: 0 0 -1	$(-0.2*10) = -2$ matched
S2: 0 -1 2	$(0.1*10) = 1$
S3: 2 0 0	$(1*10) = 10$ matched
S4: -94 2 1	$(-46.2*10) = -462$
S5: 0 0 1	$(0.2*10) = 2$
.....	
S25: 0 -1 0	$(-0.3*10) = -3$ matched

-469	-2
10	0
0	3
-3	0
0	0
0	0

Matrix G

(a) Genetic algorithm generates random strings

with Matrix G

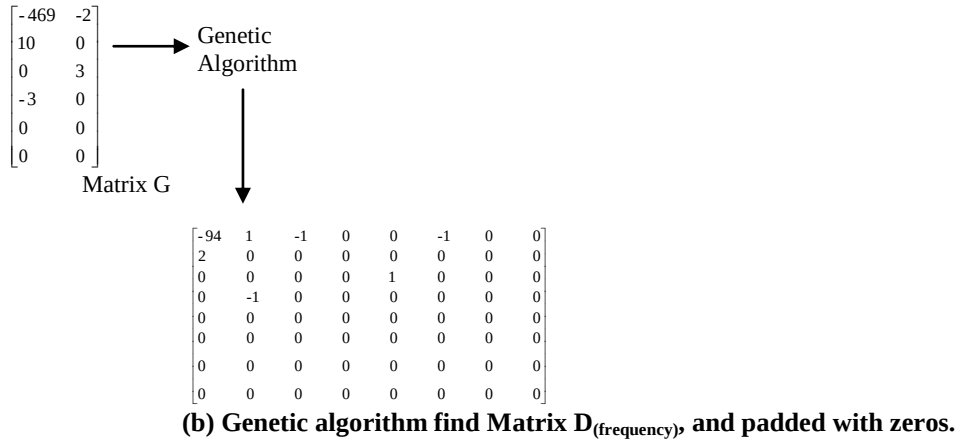
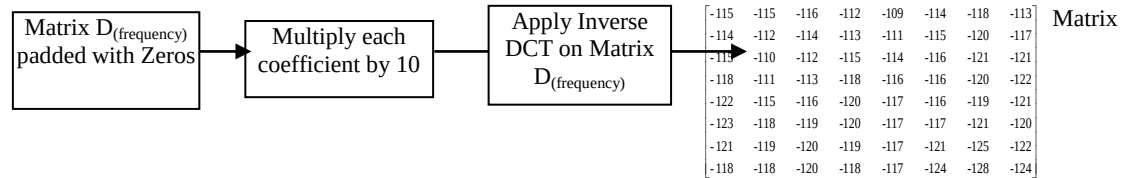


Figure -5 (a-b) Genetic algorithm searches for matrix D coefficients

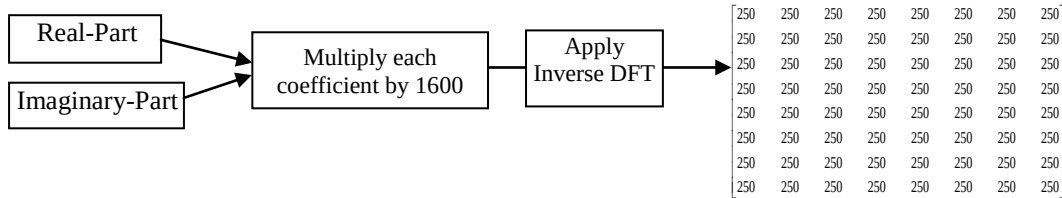
After the crossover between all 25 strings, we not need to ignore some string or make copy for some strings has minimum error at next generation. In this paper the crossover will be done between all 25 strings, and then they all transferred to the next generation, because these strings have all probability of data and one or more strings reach to the solution (See Figure -5). The matrix $D_{(\text{frequency})}$ padded with zeros transformed to spatial domain by inverse DCT, this matrix represents matrix $D_{(\text{spatial})}$. Finally using Inverse DFT to transform the matrices in Figure – 2 (d,e) into spatial domain, then add with matrix $D_{(\text{spatial})}$ to produce decompressed matrix approximately equal to original matrix (See Figure-2(a)), the Figure -6 shown decompression by using Inverse DCT, and Inverse DFT.



$D_{(\text{spatial})}$

(a) Convert Matrix $D_{(\text{frequency})}$ to Matrix $D_{(\text{spatial})}$

(Continue)



(b) Apply Inverse Fourier Transformation on Real, and imaginary matrices Matrix A

135	135	134	138	141	136	132	137
136	138	136	137	139	135	130	133
135	140	138	135	136	134	129	129
132	139	137	132	134	134	130	128
128	135	134	130	133	134	131	129
127	132	131	130	133	133	129	130
129	131	130	131	133	129	125	128
132	132	130	132	133	126	122	126

(c) Add matrix $D_{(\text{spatial})}$ with matrix A to get approximately original image

Figure – 6 (a-c) using Inverse DCT, and Inverse DFT produce decompressed image

4. Computer Simulation

Our approach applied on **Pentium4 – 1.7GHz**, with **RAM – 1GByte**, and using **MATLAB language**, the images are tested on our approach as shown in Figure – 7.

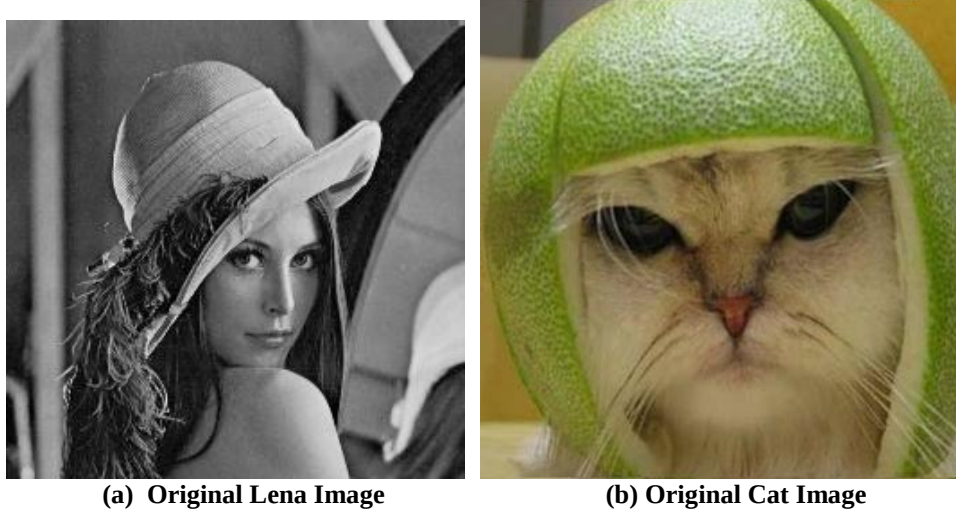


Figure – 7 (a) Gray level size 64KByte with dimension 256 x 256, (b) Color Image size 363KByte, with dimension 352 x 352.

The Lena Image consists from 8-bit gray level tested on our approach. At DFT part the frequency domain matrix divided by the Value1=16000, and also at the DCT the matrix $D_{(frequency)}$ is divided by the Value2=20, The compressed size for Lena image is 11.7KByte, after decompression the PSNR= 31.6 dB for Lena image as shown in Figure -9(a). Peak signal to noise ratio (PSNR) can be calculated very easily and is therefore a very popular quality measure [6,10,13]. The PSNR it is measured on a logarithmic scale and is based on the mean squared error (MSE) between an original image and decompressed image, relative to $(255)^2$ (i.e. the square of the highest possible signal value in the image).

The Cat Image consist from three layers (i.e. consist from 3-dimensional array; Red, Green, and Blue). This mean each layer is compressed independently using our approach. Red, Green and Blue are transformed to YC_bC_r , the goal of this transformation is to obtain compression efficiency [1,2,7,10], figure 8 shown the matrix transformation.

$$\begin{aligned}
 & \begin{bmatrix} \frac{R}{5} \\ \frac{G}{2} \\ \frac{B}{2} \end{bmatrix} \begin{bmatrix} 0.3 & 0.6 & 0.1 \\ -0.15 & -0.3 & 0.45 \\ 0.438 & -0.375 & -0.063 \end{bmatrix} = \begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} \quad (a) \\
 & \begin{bmatrix} Y * 5 \\ C_b * 2 \\ C_r * 2 \end{bmatrix} \begin{bmatrix} 1 & 0.0016 & 1.5987 \\ 1 & -0.3341 & -0.7994 \\ 1 & 2 & 0 \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (b)
 \end{aligned}$$

Figure – 8 (a) Transform matrix from RGB to YC_bC_r , (b) Transform matrix from YC_bC_r to RGB

At the compression part the matrix in Figure - 8(a) used to transform the Cat Image from Red, Green, and Blue to YCbCr these three forms reduce the image levels from 255 to less than 50, for each layer, this transformation increases compression performance. The transformed image compressed by using our approach, which the Value1=8000, and Value2=5. The compressed image size is 17.8Kbyte, and the PSNR=32.5 dB for Cat image. The Figure - 9(b) shown decompressed Cat Image.



(a) Decompressed Lena Image, PSNR = 31.6 dB (b) Decompressed Cat Image, PSNR = 32.5 dB

Figure – 9 (a,b)Decompressed Images by our approach

Our approach compared with JPEG and PNG, these method are popular used in image compression specially when transmits through internet [1,2,4,7,10,11]. The JPEG is a lossy data compression scheme for color and gray-scale images, based on DCT and Arithmetic coding. It works on full 24-bit color, and was designed to be used with photographic material and naturalistic artwork, also the JPEG can produce a smaller file than PNG for [photographic](#) images since it uses a [lossy encoding method](#) specifically designed for photographic image data [1,2,9]. The PNG use lossless data compression methods, which are used in compression of library. The most common general-purpose, lossless image compression algorithm used with LZW[1,2,9]. These algorithms compared with our approach by image size, and PSNR as shown in Table1.

Table 1 comparison our approach with JPEG

Algorithm	Image Name	Before Compression	After Compression	PSNR
Our Approach	Lena	64Kbytes	11.7Kbyte	31.6
	Cat	363Kbyte	17.8Kbyte	32.5
JPEG	Lena	64Kbytes	12.8Kbyte	33.5
	Cat	363Kbyte	18.4Kbyte	39.7

5. Conclusion

The advantage of our algorithm can be illustrated in the following steps:-

- 1- The main reason for using DFT most of data transform to zero, when divided by Value1 (See Figure – 2 (d,e)). Also the DCT used to transform most of data to zero, when divided on Value2 (See Figure – 3 (c)). This process leads for increasing compression performance.
- 2- Our algorithm gives a best performance for color image compression more than gray level images.
- 3- The Genetic Algorithm used in Decompression part, looking for Minimized Matrix $D_{(\text{frequency})}$, based on the probability of Matrix $D_{(\text{frequency})}$ and the Matrix G. The genetic Algorithm finds the result as fast as possible, because not needs for mutation just needs for crossover between strings, and not need to increase number of strings at next generation.

The disadvantage of our approach can be illustrated in the following steps:

- 1- Using more than one transformation (i.e. using DFT and DCT) these processes reduce image quality more than JPEG technique (See Table 1).
- 2- The arithmetic coding and decoding used two times by our approach, this led more computations and recurrence calculating may be led to increase time execution for compression and decompression, also the Genetic Algorithm increase decompression time. These reasons make our approach slower than JPEG.

REFERENCES

- [1] K. Sayood, "Introduction to Data Compression", Morgan Kaufmann Publishers, Inc., San Francisco, CA, 2nd dition, 2000.
- [2] Rafael C. Gonzalez, Richard E. Woods "Digital Image Processing", Addison Wesley publishing company – 2001.
- [3] S. G. Chang, B. Yu, and M. Vetterli, "Adaptive wavelet shareholding for image denoising and compression",. IEEE Trans. Image Process. vol. 9, no. 9, pp. 1532.1546, 2000.
- [4] K. R. Rao, P. Yip, Discrete cosine transform: Algorithms, advantages, applications, Academic Press, San Diego, CA, 1990.
- [5] S. S. Pradhan, K. Ramchandran, Enhancing analog image transmission systems using digital side information: A new wavelet-based image coding paradigm, in: Proc. IEEE Data Compression Conf. (DCC), Snowbird, UT, 2001, pp. 63–72.
- [6] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using wavelet transform," IEEE Trans. on Image Processing, vol. 1, no. 2, pp. 205–220, Apr. 1992.

- [7] C. Christopoulos, J. Askelof, and M. Larsson, "Efficient methods for encoding regions of interest in the upcoming JPEG 2000 still image coding standard," IEEE Signal Processing Letters, vol. 7, no. 9, pp. 247–249, Sept. 2000.
- [8] M.A.Figueiredo and R.D.Nowak, "Anew algorithm of wavelet-based image restoration," IEEE Trans. On Image Processing, vol. 12, no. 8, pp. 906–916, August 2003.
- [9] I. E. G. Richardson, Video Codec Design, John Wiley & Sons, 2002.
- [10] D. Marpe, H. Schwarz and T.Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard", IEEE Transactions on Circuits and Systems for Video Technology, to be published in 2003.
- [11] K. R. Rao and P. Yip, Discrete Cosine Transform, Academic Press, 1990.
- [12] 8. I. Witten, R. Neal and J. Cleary, Arithmetic coding for data compression, Communications of the ACM, 30 (6), June 1987
- [13] John R. Koza, "Genetic Programming II: Automatic Discovery of Reusable Programs", MIT Press, 1994.
- [14] Reiko Tanese, "Distributed Genetic Algorithms. In Proceedings of the Third International Conference on Genetic Algorithms", pp. 434-439, 1989.
- [15] Eun Yi Kim, Se Hyun Park, Sang Won Hwang, Hang Joon Kim, "Video sequence segmentation using genetic algorithms", Volume 23 , Issue 7 May, Elsevier Science Inc. New York, NY, USA, 2002.
- [16] Y. Hill, S. O'Keefe, and D. Thiel. "An investigation of wavelet design using genetic algorithms". In Microelectronic Engineering Research Conference, 2001.

Recived (20/12/2010)
Accepted (3/1 /2011)