ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 7, No. 1, 2012, *pp*. 037-053

Applied two Stages Minimize-Matrix-Size Algorithm with DCT on DWT for Image Compression

Mohammed Mustafa Siddeq[+]

Software Engineering Dept. Technical College –Kirkuk - Iraq

(*Received September 25, 2011, accepted December 2, 2011*)

Abstract. As the use of digital imaging is on the rise, compression of acquired digital image data is becoming more and more important to cope with the storage requirements. One of the challenges of compression is to compress images with high efficiently while preserving critical data from getting permanently lost in reconstructed images. In this research we introduce a proposed algorithm for image compression, based on the *Minimize-Matrix-Size Algorithm* for coding and *Limited Sequential Search-Algorithm (LSS-Algorithm)* for decoding. The proposed algorithm starts by using single stage Discrete Wavelet Transform, to decompose an image into four subbands; low frequency and high frequencies. Each "$n \times n$" of the low-frequency subband (LL) are transformed by using two dimensional DCT, store all DC coefficients in different matrix called *DC-Matrix*, and the remain AC coefficients are stored in different matrix called *AC-Matrix*, then applying Minimize-Matrix-size algorithm for the AC-Matrix, to convert each group of AC coefficients into single floating point value. The DC-Matrix transformed again by DWT, and apply Minimize-Matrix-Size algorithm on it. LSS-Algorithm which is represents decoding; DC-Matrix and AC-Matrix, this algorithm is used to estimate original values by using iterative method, which is depends on the probability of the data of the AC-Matrix and DC-Matrix. Our compression algorithm proved good compression ratio, and compared with JPEG and JPEG2000 depending on the PSNR and HVS.

Keywords: Single stage DWT, DCT, Minimize-Matrix-Size, LSS-Algorithm.

1. Introduction

The researches in compression techniques has stemmed from the ever increasing need for efficient data transmission, storage and utilization of hardware resources. Uncompressed image data require considerable storage capacity and transmission bandwidth. Despite rapid progresses in mass storage density, processor speeds and digital communication system performance demand for data storage capacity and data transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia based applications have not only sustained the need for more efficient ways to encode signals and images but have made compression of such signals central to signal storage and digital communication technology[1-4].

Compressing an image is significantly different from compressing raw binary data. Of course, general purpose compression programs can be used to compress images, but the result is less than optimal. This is because images have certain statistical properties which can be exploited by encoders specifically designed for them. Also, some of the finer details in the image can be sacrificed for the sake of saving a little more bandwidth or storage space.

Lossless compression involves with compressing data which, when decompressed, will be an exact replica of the original data. This is the case when binary data such as executables documents etc. are compressed. They need to be exactly reproduced when decompressed. On the other hand, images need not be reproduced 'exactly'. An approximation of the original image is enough for most purposes, as long as the error between the original and the compressed image is tolerable.

The neighboring pixels of most of the images are highly correlated and therefore hold redundant information from certain perspective of view [4]. The foremost task then is to find out less correlated representation of the image. Image compression is actually the reduction of the amount of this redundant data (bits) without degrading the quality of the image to an unacceptable level [4-7]. There are mainly two basic

[+] *Email address*: mamadmmx76@yahoo.com

components of image compression – redundancy reduction and irrelevancy reduction. The redundancy reduction aims at removing duplication from the signal source image while the irrelevancy reduction omits parts of the signal that is not noticed by the signal receiver i.e., the Human Visual System (HVS) [7] which presents some tolerance to distortion, depending on the image content and viewing conditions. Consequently, pixels must not always be regenerated exactly as originated and the HVS will not detect the difference between original and reproduced images.

The current standards for compression of still image (e.g., JPEG) use Discrete Cosine Transform (DCT), which represents an image as a superposition of cosine functions with different discrete frequencies [8]. The DCT can be regarded as a discrete time version of the Fourier Cosine series. It is a close relative of Discrete Fourier Transform (DFT), a technique for converting a signal into elementary frequency components. Thus, DCT can be computed with a Fast Fourier Transform (FFT) like algorithm of complexity $O(nlog2\ n)$. More recently, the wavelet transform has emerged as a cutting edge technology within the field of image analysis. The wavelet transformations have a wide variety of different applications in computer graphics including radiosity [9], multiresolution painting [10], curve design [11], mesh optimization [12], volume visualization [13], image searching [14,15] and one of the first applications in computer graphics, image compression. The Discrete Wavelet Transformation (DWT) provides adaptive spatial frequency resolution (better spatial resolution at high frequencies and better frequency resolution at low frequencies) that is well matched to the properties of a Human Visual System (HVS).

This paper introduces a proposed image compression depends on the single stage DWT which is decomposes an image into approximation coefficients (LL) and high frequency domains (LH, HL and HH). Each *"$n \times n$"* group of data from the approximation coefficients are transformed by DCT to produce; DC coefficient and AC coefficients. The AC coefficients are coded by Minimize-Matrix-Size algorithm; while the each DC coefficient is stored in a new matrix called *DC-Matrix*. Also the DC-Matrix transformed again by single stage DWT. The high frequencies subbands are coded by using Minimize-Matrix-Size algorithm. This paper also describes LSS-Algorithm used for decode: DC-Matrix, AC coefficients and other high frequencies subbands. Finally these subbands are recomposed decoded low frequency and high frequency by using inverse DWT, Figure-1 shows our compression flowchart.

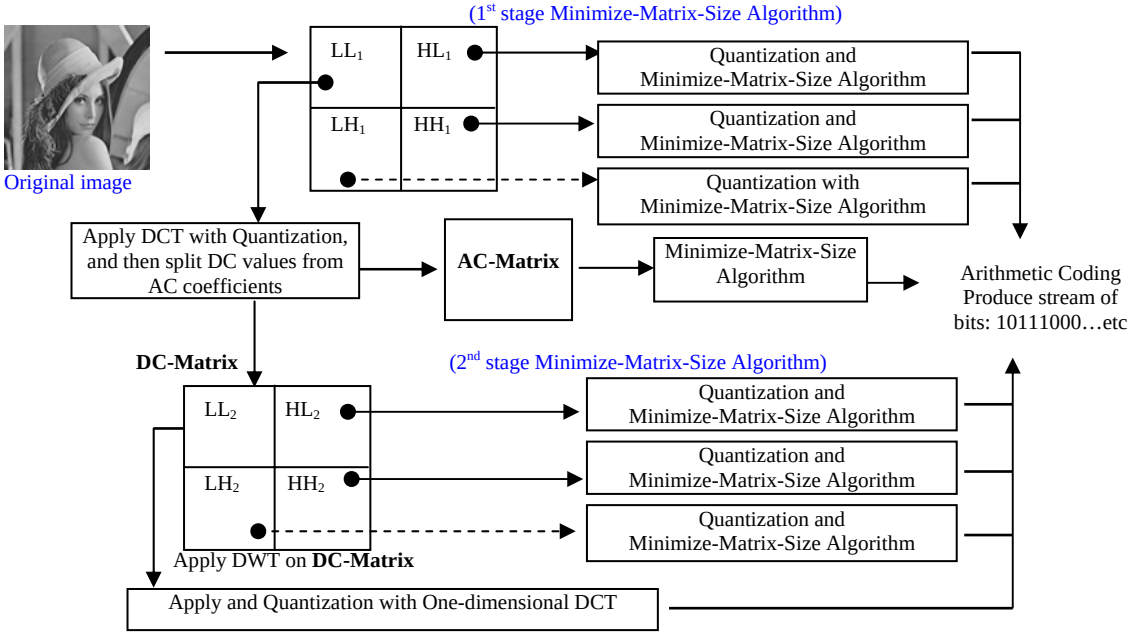


Figure-1, Our Compression algorithm flowchart

2. Discrete Wavelet Transform (DWT)

Wavelet transform exploits both the spatial and frequency correlation of data by dilations (or contractions) and translations of mother wavelet on the input data. It supports the multiresolution analysis of data (i.e. it can be applied to different scales according to the details required, which allows progressive transmission and zooming of the image without the need of extra storage)[7]. Another encouraging feature of wavelet transform is its symmetric nature that is both the forward and the inverse transform has the same

complexity, building fast compression and decompression routines. Its characteristics well suited for image compression include the ability to take into account of Human Visual System's (HVS) characteristics, very good energy compaction capabilities, robustness under transmission, high compression ratio etc[12].

The implementation of wavelet compression scheme is very similar to that of subband coding scheme: the signal is decomposed using filter banks. The output of the filter banks is down-sampled, quantized, and encoded. The decoder decodes the coded representation, up-samples and recomposes the signal. Wavelet transform divides the information of an image into approximation (i.e. LL) and detail subband[15]. The approximation subband shows the general trend of pixel values and other three detail subband show the vertical, horizontal and diagonal details in the images. If these details are very small (threshold) then they can be set to zero without significantly changing the image[13], for this reason the high frequencies subbands compressed to few bytes. In this research the DWT used twice, this is because the DWT assemble all low frequency coefficients in one region, which is represents quarter of the image size. This reduction of the size helped our approach to gets high compression ratio.

3. Discrete Cosine Transform (DCT)

This is the second transformation used in this research, which is applies on each "n × n" block from "LL1" subband as show in Figure-2.

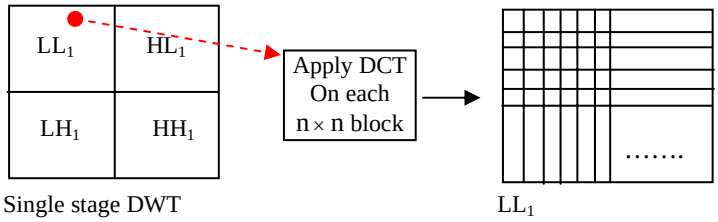


Figure-2, LL1 subband transformed by DCT for each n × n block

The energy in the transformed coefficients is concentrated about the top-left corner of the matrix of coefficients. The top-left coefficients correspond to low frequencies: there is a 'peak' in energy in this area and the coefficients values rapidly decrease to the bottom right of the matrix, which means the higher-frequency coefficients [2,3]. The DCT coefficients are de-correlated, which means that many of the coefficients with small values can be discarded without significantly affecting image quality. A compact matrix of de-correlated coefficients can be compress much more efficiently than a matrix of highly correlated image pixels. The following equations illustrated DCT and Inverse DCT function for two-dimensional matrices [7,8]:

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

Where a(u)= $\sqrt{\frac{1}{N}}$ for u = 0 and a(u) = $\sqrt{\frac{2}{N}}$ for u ≠ 0

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

One of the key difference between the applications of the Discrete Wavelet transformation (DWT) and Discrete Cosine transformation (DCT) is; the DWT typically apply to an image as a one block or a large rectangular region of the image, while for the small block size using DCT [6,8,9]. The DCT becomes increasingly complicated to calculate for the larger block sizes, for this reason in this paper used " n × n " block to be transformed by DCT, whereas a DWT will be more efficiently applied to the complete images and produce good compression ratio [10].

Each "n × n" coefficients from LL_1 divided by "Q. using matrix-dot-division then truncate it. This process called Quantization, which removes insignificant coefficients and increasing the zeros in LL_1. The Q can be computed as follows:

$$L = Quality \times \max(\mathrm{LL}_1) \tag{3}$$

$$Q(i,j) = \begin{cases} 10, & for\ \ i,j = 1 \\ L + i + j, & for\ \ i > 1 \end{cases} \tag{4}$$

$i, j=1,2,3,...n \times n$

Parameter "L" in the above Eq(3) is computed from the maximum value in the LL_1 subband, and "Quality" value $\geq$ 0.01. The quality value is represents as a ratio for maximum value, if this ratio increased; this leads to obtains low image quality. Each DC value from "n × n" block, stored in different matrix called DC-Matrix, and other AC coefficients (n × n -1) are stored in AC-Matrix. Figure-3 shows the LL_1 subband separated into AC-Matrix and DC-Matrix.

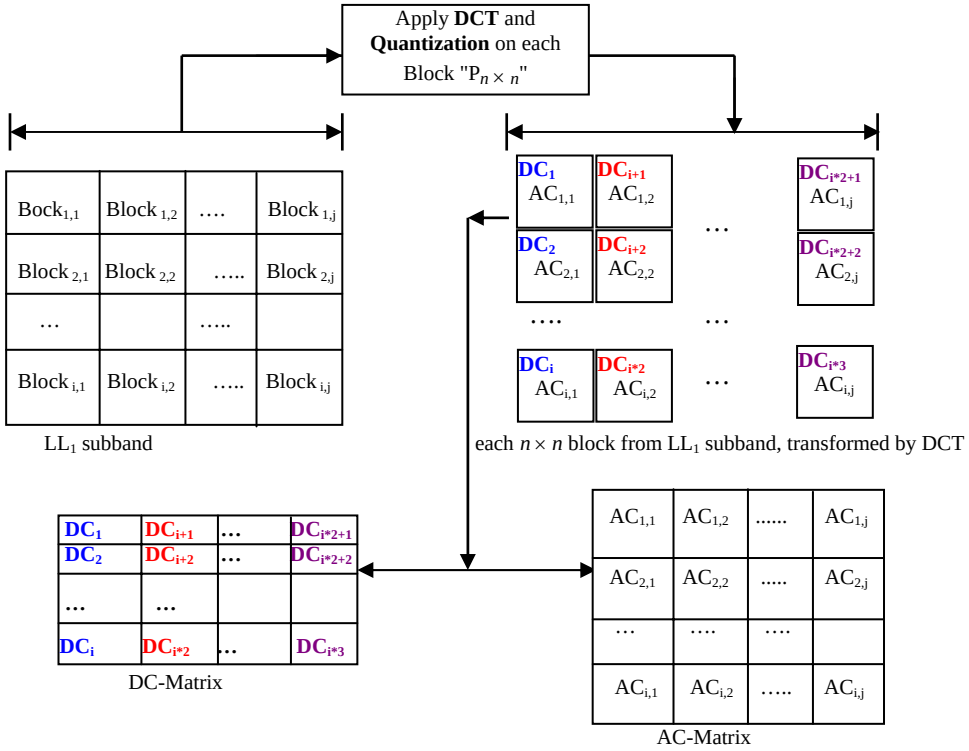


Figure -3, LL_1 subband separated into DC-Matrix and AC-Matrix

The other high frequency subband (HL_1, LH_1 and HH_1) produced from DWT (See Figure-1), quantized by using just Eq.(3) multiplied by "2". This is because for removes insignificant details coefficients. The following Eq(5) represents quantization for high frequencies:

$$Q_{High} = L \times 2 \tag{5}$$

The DC-Matrix transformed again by DWT to produce; LL_2, LH_2, HL_2 and HH_2. The approximation LL_2 divided by "2. while other high frequency subbands divided by "4". This is because for obtains more zeros in high frequency domains, and reduce size of values in LL_2 to be ready for transformations.

The LL_2 transformed by using one-dimensional DCT for each *n* data (i.e. assume *u=0,v=0* to converting two dimensional DCT into one dimensional DCT) in the same way that LL_1 subband transformed, but without splitting DC value from AC coefficients. The quantization for LL_2 is computed through:

$$Q_2(i) = i \times 2 \tag{6}$$

$i=1, 2, 3,..., n$

The next step, takes the difference between two neighbours values for each column in LL_2. This process is called DBV (Difference Between two Values), which is based on the well-known fact that neighbouring coefficients in the LL_2 are correlated. Correlated values are generally similar, so their differences are small and more data will be repeated, this will be very easy for compression. The following equation represents DBV:

$$D(i) = D(i) - D(i+1) \tag{7}$$

Where \ *i=1,2,3,…, m-1* *m* : column size for "LL_2"

Figure-4 illustrates the Eq.(7) applied on a matrix.

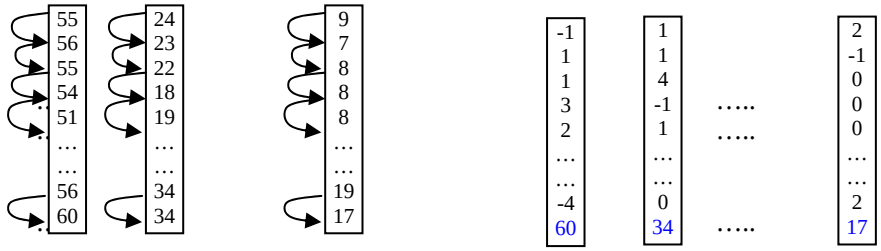


Figure-4, (a) A matrix before apply DBV, (b) Apply DBV between two neighbours in each column

4. Compress Data by Minimize-Matrix-Size Algorithm

This algorithm is used to reduce size of the AC-Matrix and all other high frequency subbands (LH, HL and HH). This algorithm depends on the *Random-Weights-Values*, by taking each *three* coefficients to convert it into single floating point values stored in an array. The following List-1 shows Minimize-Matrix-Size-Algorithm:

List-1 *Minimize-Matrix-Size Algorithm*

```
Let K=3;      %% take each three coefficients from a matrix
W=Generate-Ransom-Weights (K);   %% generate three random weights values, according to number of
                                 %% coefficients
Let p=1;
For I=1 to column size
  For  J=1 to row size
   Intermediate [p]=Matrix[i,j]; %% Scan row-by-row
  p++
   End;
End;

Let  J=1; p=1;
   While (J<row size*column size)
    Arr=Read_K_coefficnets (Intermediate [j]);
      M(p) = Σ_{i=1}^{K} W(i) * Arr(i)
     J=J+K;
    p++;
  End;   %% end while
```

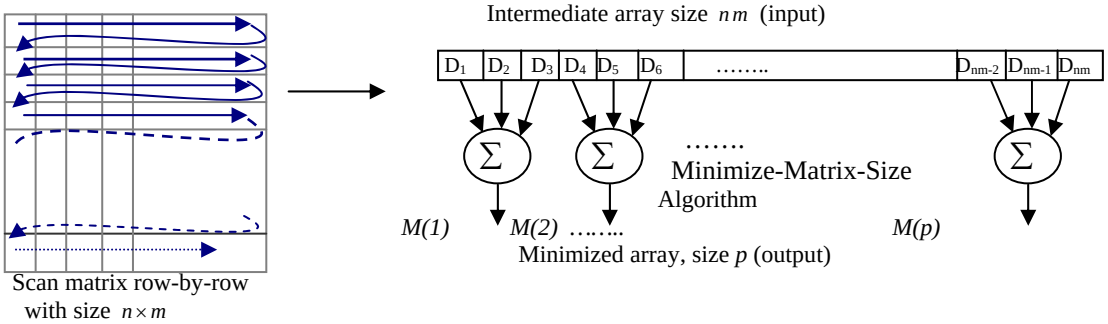


Figure-5, Minimized matrix $n \times m$ into minimized array "*M*"

In the above List-1 the random weights values are generated randomly (i.e. random numbers in the range ={0…1}) multiplied with Arr(i) (i.e. represents three coefficients from a matrix) to produce minimized array M(p). List-1 applied on each subband independently, this means each minimized subband, compressed independently by arithmetic coding. Figure-5 illustrates Minimize-Matrix-Size algorithm applied on a matrix.

Before apply the Minimize-Matrix-Size algorithm, our compression algorithm computes the probability of the data for each high frequency matrix. These probabilities are called Limited-Data, which is used later in decompression algorithm, the Limited-Data stored as a header for compressed file. Because these data uncompressible. Figure-6 illustrates probability of data for a matrix:

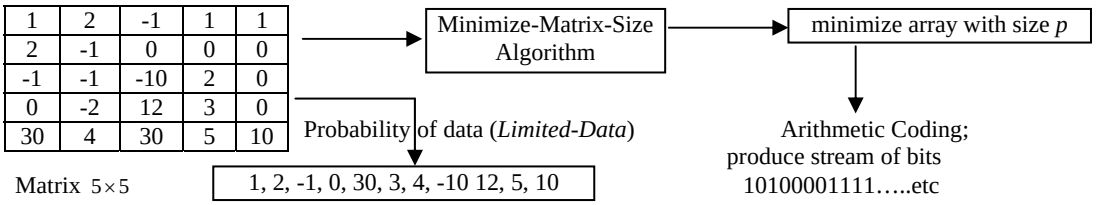
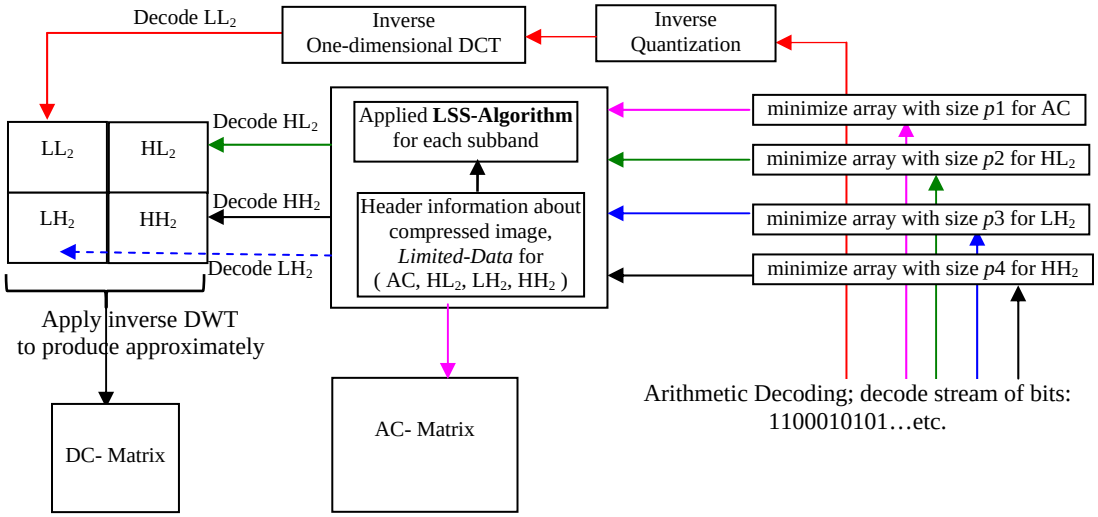


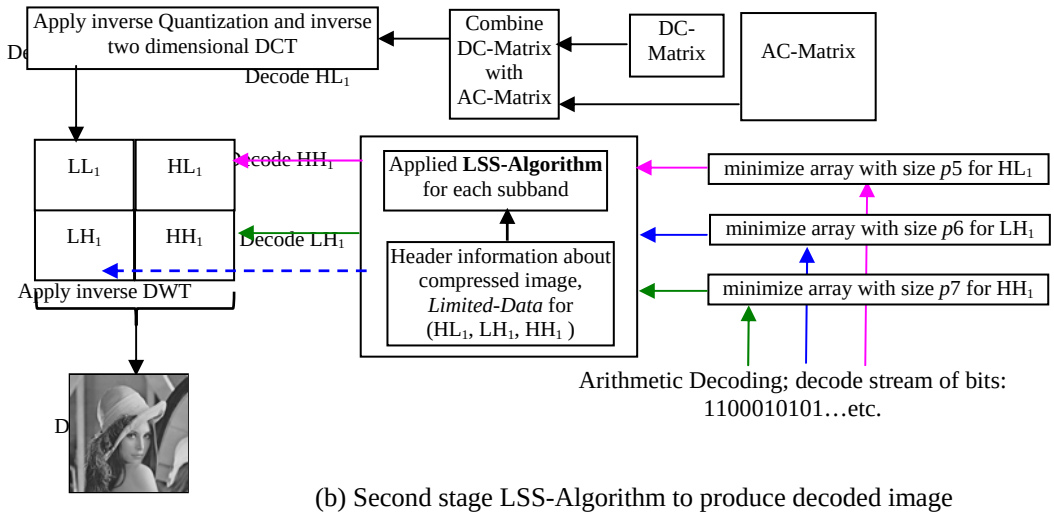
Figure-6, illustrates probability of data (*Limited-Data*) for the matrix 5×5

The final step in our compression algorithm is; *Arithmetic coding*, which takes stream of data and convert it into single floating point value. These output values in the range less than one and greater than zero [2], when decoded this value getting exact stream of data. The arithmetic coding need to compute the probability of all data and assign range for each data, the range value consist from Low and High value [7].

5. Decompression by LSS-Algorithm



(a) First stage LSS-Algorithm to produce DC-Matrix and AC-Matrix



(b) Second stage LSS-Algorithm to produce decoded image

Figure -7, (a,b) Two stages Limited Sequential Search algorithm for image decoding

our decompression algorithm it is reverse of our compression algorithm, the first step for our decompression algorithm starts to decode minimized array by using arithmetic decoding, then the LSS-Algorithm is used for decode each subband independently, LSS-Algorithm depends on the Limited-Data for each minimized array. If the limited-data is missed or destroyed, the image couldn't return back. The following figure shows the decompression flowchart:

List-2 *LSS-Algorithm*

```
Let Limited [1…m];        %% represents Limited Data
Let M [1…p];              % % represents minimized array with size p
Let K=3;                  %% number of estimated coefficients
 For i=1 to p
  S1=1; S2=1; S3=1;       %% initial location
  Iterations=1;
    Est=W(1)*Limited[S1] + W(2)*Limited[S2]+W(3)*Limited[S];

While ( (M(i) – Est)  ≠  0)   % check if Error =0 or not
      S3++;       % % increment pointer represents "Seconds"
      IF (S3>m)      S2++; S3=1; end;    %% check if S3 is over the limit, return back to "1. and increment S2
      IF (S2>m)      S1++; S2=1; end;
      IF (S1>m)      S1=1;          end;
    Est=W(1)*Limited[S1] + W(2)*Limited[S2]+W(3)*Limited[S3];   %% compute Est after increments
    Iterations++;       %% compute number of iterations
 End;      %% end of while
End;
```

Iteration=1			M(1) – Est
Locations			Error = -31.54
S1	S2	S3	[30,19,1,0]
1	1	1	Est =**35.19**

Iteration=2			M(1) – Est
Locations			Error = -26.887
S1	S2	S3	[30,19,1,0]
1	1	2	Est =**30.537**

Iteration=3			M(1) – Est
Locations			Error = -19.273
S1	S2	S3	[30,19,1,0]
1	1	3	Est =**22.923**

Iteration=4			M(1) – Est
Locations			Error = -18.85
S1	S2	S3	[30,19,1,0]
1	1	4	Est =**22.5**

Iteration=5			M(1) – Est
Locations			Error = -46.09
S1	S2	S3	[30,19,1,0]
1	2	1	Est =**49.74**

Iteration=6			M(1) – Est
Locations			Error = -19.737
S1	S2	S3	[30,19,1,0]
1	2	2	Est =**23.387**

Iteration=7			M(1) – Est
Locations			Error = -12.123
S1	S2	S3	[30,19,1,0]
1	2	3	Est =**15.773**

Iteration=8			M(1) – Est
Locations			Error = -11.7
S1	S2	S3	[30,19,1,0]
1	2	4	Est =**15.35**

Iteration=9			M(1) – Est
Locations			Error = -12.69
S1	S2	S3	[30,19,1,0]
1	3	1	Est =**16.34**

Iteration=10			M(1) – Est
Locations			Error = -8.037
S1	S2	S3	[30,19,1,0]
1	3	2	Est =**11.687**

Iteration=11			M(1) – Est
Locations			Error = -0.423
S1	S2	S3	[30,19,1,0]
1	3	3	Est =**4.073**

Iteration=12			M(1) – Est
Locations			Error = 0
S1	S2	S3	[30,19,1,0]
1	3	4	Est =**3.65**

Stop iteration

(a) LSS-Algorithm finds the estimated values for *M(1)*

Iteration=1			M(2) – Est
Locations			Error = -32.217
S1	S2	S3	[30,19,1,0]
1	1	1	Est = **35.19**

Iteration=2			M(2) – Est
Locations			Error = -27.564
S1	S2	S3	[30,19,1,0]
1	1	2	Est = **30.537**

Iteration=3			M(2) – Est
Locations			Error = -19.95
S1	S2	S3	[30,19,1,0]
1	1	3	Est = **22.923**

…
… After 27 iterations
…

Iteration=28			M(2) – Est
Locations			Error = 0
S1	S2	S3	[30,19,1,0]
2	3	3	Est =**2.973**

Stop iteration

(b) LSS-Algorithm finds the estimated values for *M(2)*

Figure-8, (a,b) LSS-Algorithm iterations to estimate the matrix values

The LSS-Algorithm is designed for finding missing data inside a Limited-Data, by using three pointers. These pointers are refers to index in the Limited-Data, the initial value for these pointers are "1" (i.e. first location in a Limited-Data). These three pointers are called *S1*, *S2* and *S3*; these pointers increment by one at each iteration (i.e. these three pointers work as a digital clock, where *S1*, *S2* and *S3* represent hour, minutes

and seconds respectively). To illustrate LSS-Algorithm assume we have the following matrix 2*3:

30	1	0
19	1	1

The above matrix will compress by using Minimize-Matrix-Size algorithm to produce minimized array *M(1)=3.65* and *M(2)=2.973*, the Limited-Data ={30,19,1,0}. Now the LSS-Algorithm will return original values for the above matrix 2*3, by using the Limited-Data and the random-weight-values= {0.1, 0.65, and 0.423}. The first step in our decompression algorithm, assign S1=S2=S3=1, then compute the result by the following equation:

$$Est = W(1) \times Limited(S1) + W(2) \times Limited(S2) + W(3) \times Limited(S3) \quad (8)$$

W:- generated weights. Limited:- Limited Data

LSS-Algorithm computes "*Est*" at each iteration, and compare it with *M(i)*. If it is zero, the estimated values in limited-data at locations= {S1, S2 and S3}. If not equal to zero, the algorithm will continue to find the original values inside the limited-data. This process will continue until end of minimized array "M(i)". The List-2 illustrates LSS-Algorithm, and Figure-8 illustrates LSS-Algorithm estimates matrix values.

5.1. First stage LSS-Algorithm

After the high frequency subbands (i.e. HL_2, LH_2, HH_2 and AC-Matrix) are decoded by LSS-Algorithm (See Figure-7(a)), the high frequency subbands (i.e. HL_2, LH_2 and HH_2) are multiplied by "4" which is inverse quantization. At this stage all high frequencies subbands are decoded.

The next step in our decompression algorithm applied ABV on the decoded LL_2, for return previous values. ABV (Addition Between two Values), represents inverse equation for DBV (See Figure-4). ABV applied on each column, which takes the last value at position "*m*" , and add it to the value at position "*m-1*. then the total adds with the value at position "*m-2*" and so on, this process continue until position "1. as shown in Figure-9. The following equation shows ABV decoder:-

$$D(i-1) = D(i-1) + D(i) \quad (9)$$

Where \ i= m, (m-1), (m-2), (m-3),…,2

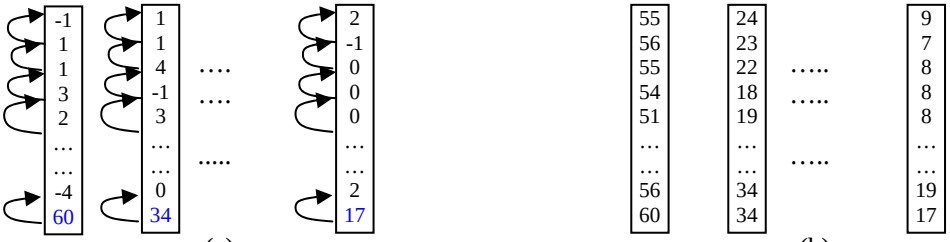


Figure-9, (a) A matrix before apply ABV, (b) Apply ABV between two neighbours in each column

Applied inverse quantization on the LL_2 with Q_2 for each *n* data, and then applied inverse one-dimensional DCT on each *n* data from LL_2. Finally multiply LL_2 by "2. and then apply inverse DWT to produce approximately DC-Matrix.

5.2. Second stage LLS-algorithm

Combine each DC value from DC-Matrix with each ($n \times n$-*1*) block from AC-Matrix (See Figure-3) to generate combined matrix called LL_1, and then applied inverse quantization (See Eq(4)), and followed by the inverse two dimensional DCT on each $n \times n$ block from the LL_1, the result is reconstructed LL_1 subband. Also in this stage the other high frequencies subbands (i.e. HL_1, LH_1 and HH_1) are decoded by LSS-Algorithm, and applied inverse quantization with Q_{High} (See Eq(5)). Now the image is decoded by applying inverse DWT to recomposes the LL_1, HL_1, LH_1 and HH_1 subbands.

6. Results

In this section, our algorithm tested on the three gray level images "Lena. "X-ray" and "Iris. as shown in Figure-10. These images are tested by Microprocessor AMD Athalon - 2.1GHz with 3G RAM and using "*MATLAB R2008a* ver. 7.7" programming language, linked with the programming language "*Visual C++.NET*". The operating system used is Windows 7(32bit).

(a) Lena, image (b) X-ray image (c) Iris image

Figure – 10, (a) Lena image with dimensional (500×500), (b) X-ray image with dimensional (500×522), (c) Iris image with dimensional (512×512)

Tables 1, show our compression algorithm result, with the following parameters:

1- Using decompose the images by single stage Daubechies DWT (db3)
2- Applies two dimensional DCT on each 3×3 coefficients from LL_1 subband (i.e. *Apply Eq(1) with N=3*)
3- Applying Minimize-Matrix-Size algorithm on each subbands, with W={0.1, 6.5, 851}
4- Applies one-dimensional DCT with *n=3* on LL_2 subband

Table-1, our compression algorithm results

Image Name	Parameter "Quality" used in Eq(3)	Compressed Image size
Lena dimension (500x500) original Image size:- 245 Kbytes	0.02	25.3 Kbytes
	0.04	15.4 Kbytes
	0.06	11.6 Kbytes
	0.08	9.5 Kbytes
X-ray dimension (500x522) original Image size:- 255 Kbytes	0.02	21 Kbytes
	0.04	12.1 Kbytes
	0.06	9 Kbytes
	0.08	7.6 Kbytes
Iris dimension (512x512) original Image size:- 257 Kbytes	0.02	18.6 Kbytes
	0.04	11.2 Kbytes
	0.06	8.4 Kbytes
	0.08	6.6 Kbytes

The decompressed images by our approach shown in Figure-11, Figure-12 and Figure-13, for the images; "Lena. "X-ray" and "Iris" respectively, and Table-2 shows time execution for LSS-Algorithm.

(a) Decoded with Quality=0.02, PSNR=36.2 dB (b) Decoded with Quality=0.04, PSNR=34.4 dB

(c) Decoded with Quality=0.06, PSNR=33.1 dB (d) Decoded with Quality=0.08, PSNR=31.7 dB

Figure-11, (a – d) Decompressed Lena image by our approach, according to "Quality"

(a) Decoded with Quality=0.02, PSNR=36.8 dB (b) Decoded with Quality=0.04, PSNR=34.7 dB

(c) Decoded with Quality=0.06, PSNR=34 dB (d) Decoded with Quality=0.08, PSNR=33.2 dB

Figure-12, (a – d) Decompressed X-ray image by our approach, according to "Quality"

(a) Decoded with Quality=0.02, PSNR=38.2 dB

(b) Decoded with Quality=0.04, PSNR=36 dB

(c) Decoded with Quality=0.06, PSNR=35 dB

(d) Decoded with Quality=0.08, PSNR=34.3 dB

Figure-13, (a – d) Decompressed Iris image by our approach, according to "Quality"

Table-2, execution time for LSS-Algorithm

Decoded Image Name	**Parameter "Quality"used in Eq(3)**	**Execution time for LSS-Algorithm to find the decoded image**	**PSNR**
Lena	0.02	7 (Sec.)	36.2 dB
	0.04	6.5 (Sec.)	34.4 dB
	0.06	7.4 (Sec.)	33.1 dB
	0.08	6.1 (Sec.)	31.7 dB
X-ray	0.02	6.2 (Sec.)	36.8 dB
	0.04	8 (Sec.)	34.7 dB
	0.06	7.9 (Sec.)	34 dB
	0.08	6.5 (Sec.)	33.2 dB
Iris	0.02	8.5 (Sec.)	38.2 dB
	0.04	9.2 (Sec.)	36 dB
	0.06	8.9 (Sec.)	35 dB
	0.08	9.0 (Sec.)	34.3 dB

Peak Signal to Noise Ratio (PSNR) its used in Table-2 refers to image quality mathematically, PSNR it is a very popular quality measure, can be calculates very easily between The decompressed image and the original image, as shown in the following equation[1-4]:

$$\text{PSNR} = 10\log_{10}\frac{255^2}{\text{MSE}} \tag{10}$$

$$MSE = \frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|Decode(i,j) - Original(i,j)\right|^2 \tag{11}$$

Table -3, Comparison between JPEG, JPEG2000 and our approach for "Lena" image

JPEG			**JPEG2000**			**Our approach**		
Quality	Compressed Image size	PSNR	Quality	Compressed Image size	PSNR	Quality	Compressed Image size	PSNR
53%	25.3 Kbytes	35.2 dB	84%	25.8 Kbytes	37.6 dB	0.02	25.3 Kbytes	36.2 dB
21 %	15.7 Kbytes	32.4 dB	75%	15.6 Kbytes	35.4 dB	0.04	15.4 Kbytes	34.4 dB
11%	11.9 Kbytes	30.1 dB	68%	11.5 Kbytes	33.9 dB	0.06	11.6 Kbytes	33.1 dB
6%	11.9 Kbytes	27.8 dB	63%	9.5 Kbytes	33.1 dB	0.08	9.5 Kbytes	31.7 dB

Our approach is compared with JPEG and JPEG2000; these two techniques are used widely in the digital image compression, especially for image transmission and video compression. The JPEG technique based on the DCT applied on the partitioned image, then each partition are encoded by RLE and Huffman encoding [4,7,11]. The JPEG2000 based on the multi decomposition for DWT applied on the also partitioned image and then each partition quantized, and encoded by Arithmetic encoding [12,13]. The new ACDSee application has many options to save the images as JPEG and JPEG2000 formats. The most option is used; the *Quality*. If the image quality is increased the compression ratio is decreased and vice versa. Table-3, Table-4 and Table-5 shows the comparison of the two techniques our approach.

Table -4, Comparison between JPEG, JPEG2000 and our approach for "X-ray" image

JPEG			**JPEG2000**			**Our approach**		
Quality	Compressed Image size	PSNR	Quality	Compressed Image size	PSNR	Quality	Compressed Image size	PSNR
51%	21.3 Kbytes	33.9 dB	80%	21.2 Kbytes	37.4 dB	0.02	21 Kbytes	36.8 dB
16 %	12 Kbytes	30.3 dB	69%	12.5 Kbytes	34.8 dB	0.04	12.1 Kbytes	34.7 dB
8%	9.3 Kbytes	28.2 dB	61%	9.1 Kbytes	33.6 dB	0.06	9 Kbytes	34 dB
4%	7.9 Kbytes	25.5 dB	56%	7.8 Kbytes	33.1 dB	0.08	7.6 Kbytes	33.2 dB

Table -5, Comparison between JPEG, JPEG2000 and our approach for "Iris" image

JPEG			**JPEG2000**			**Our approach**		
Quality	Compressed Image size	PSNR	Quality	Compressed Image size	PSNR	Quality	Compressed Image size	PSNR
40 %	18.2 Kbytes	37.1 dB	77%	18.2 Kbytes	39.4 dB	0.02	18.6 Kbytes	38.2 dB
17%	11.15 Kbytes	33.7 dB	67%	11.6 Kbytes	37.3 dB	0.04	11.2 Kbytes	36 dB
10%	8.52 Kbytes	31.6 dB	59%	8.49 Kbytes	36 dB	0.06	8.4 Kbytes	35 dB
5%	6.88 Kbytes	28.5 dB	51%	6.76 Kbytes	35.1 dB	0.08	6.6 Kbytes	34.3 dB

The second measurement is Human Visual System used for the comparison between decompressed images by JPEG, JPEG2000 and our approach. The human visual system it's important to show some details for the decoded images [4,7,12]. The decoded images are zoomed-in one time, to show the features for each method. Figure-14, Figure-15 and Figure-16 has shows the comparison for the decoded images; "Lena. "X-ray. and "Iris" respectively.

(a) Decoded by **Our Approach**, Quality=0.02, PSNR=36.2 dB

(b) Decoded by **JPEG2000** Quality=84%, PSNR=37.6 dB

(c) Decoded by **JPEG** Quality=53%, PSNR=35.2 dB

(d) Decoded by **Our Approach**, Quality=0.04, PSNR=34.4 dB

(e) Decoded by **JPEG2000** Quality=75%, PSNR=35.4 dB

(f) Decoded by **JPEG** Quality=21%, PSNR=32.4 dB

(g) Decoded by **Our Approach** Quality=0.06, PSNR=33.1 dB

(h) Decoded by **JPEG2000** Quality=68%, PSNR=33.9 dB

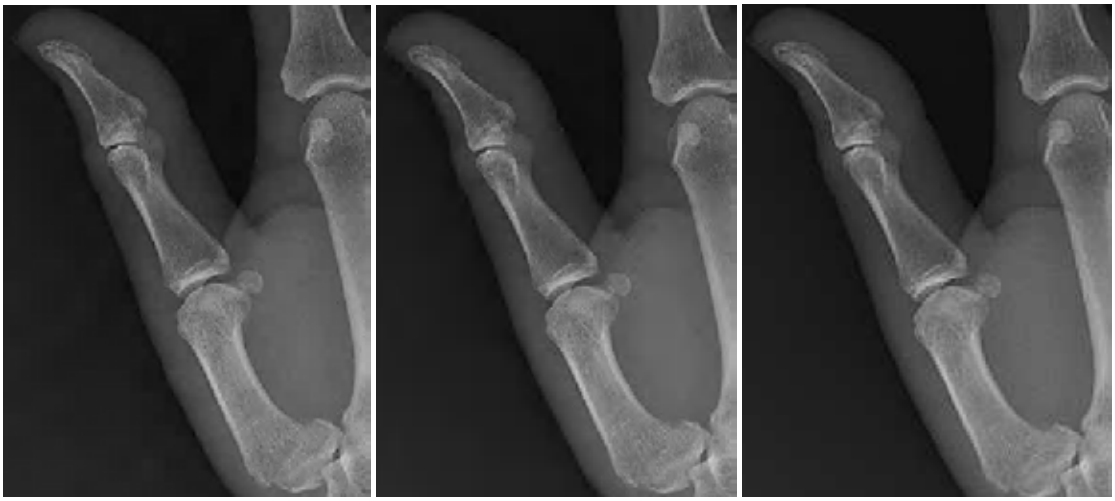
(i) Decoded by **JPEG** Quality=11%, PSNR=30.1 dB

(j) Decoded by **Our Approach** Quality=0.08, PSNR=31.7 dB

(k) Decoded by **JPEG2000** Quality=63%, PSNR=33.1 dB

(l) Decoded by **JPEG** Quality=6%, PSNR=27.8 dB

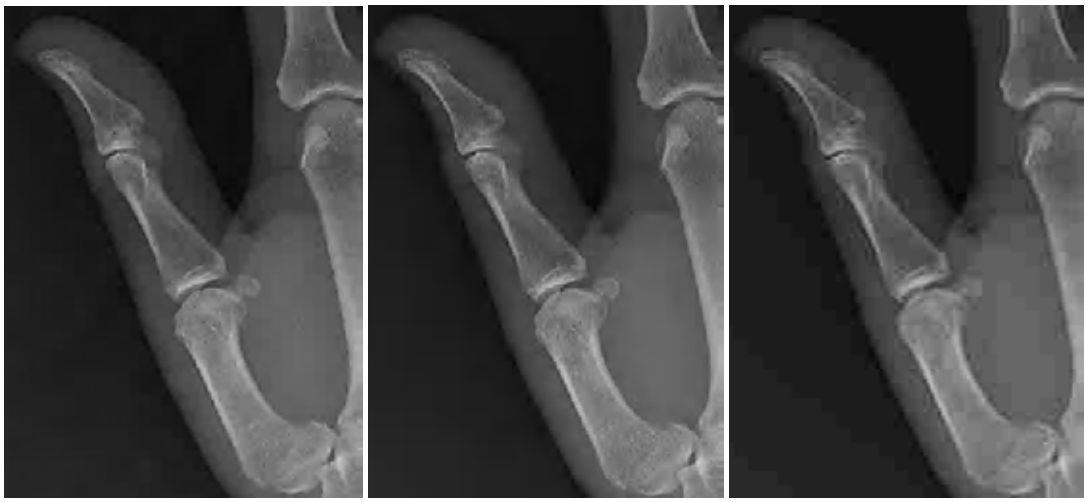
Figure-14, (a – l) Comparison the decompressed Lena image by our Approach with JPEG2000 and JPEG



(a) Decoded by **Our Approach** Quality=0.02, PSNR=36.8 dB

(b) Decoded by **JPEG2000** Quality=80%, PSNR=37.4 dB

(c) Decoded by **JPEG** Quality=51%, PSNR=33.9 dB



(d) Decoded by **Our Approach** Quality=0.04, PSNR=34.7 dB

(e) Decoded by **JPEG2000** Quality=69%, PSNR=34.8 dB

(f) Decoded by **JPEG** Quality=16%, PSNR=30.3 dB

(g) Decoded by **Our Approach** Quality=0.06, PSNR=34 dB

(h) Decoded by **JPEG2000** Quality=61%, PSNR=33.6 dB

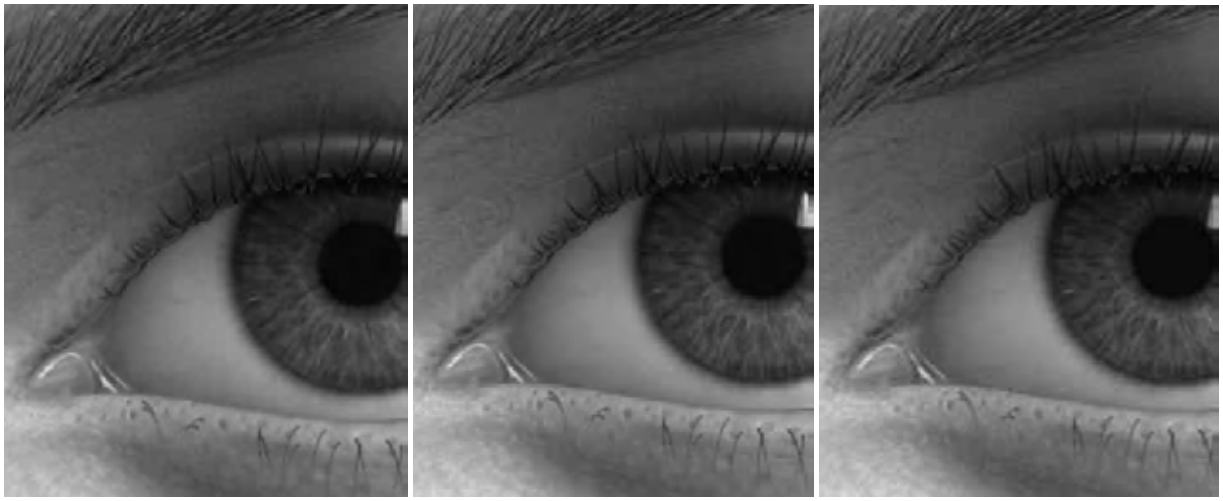
(i) Decoded by **JPEG** Quality=8% PSNR=28.2 dB

(j) Decoded by **Our Approach** Quality=0.08, PSNR=33.2 dB

(k) Decoded by **JPEG2000** Quality=56%, PSNR=33.1 dB

(l) Decoded by **JPEG** Quality=4%, PSNR=25.5 dB

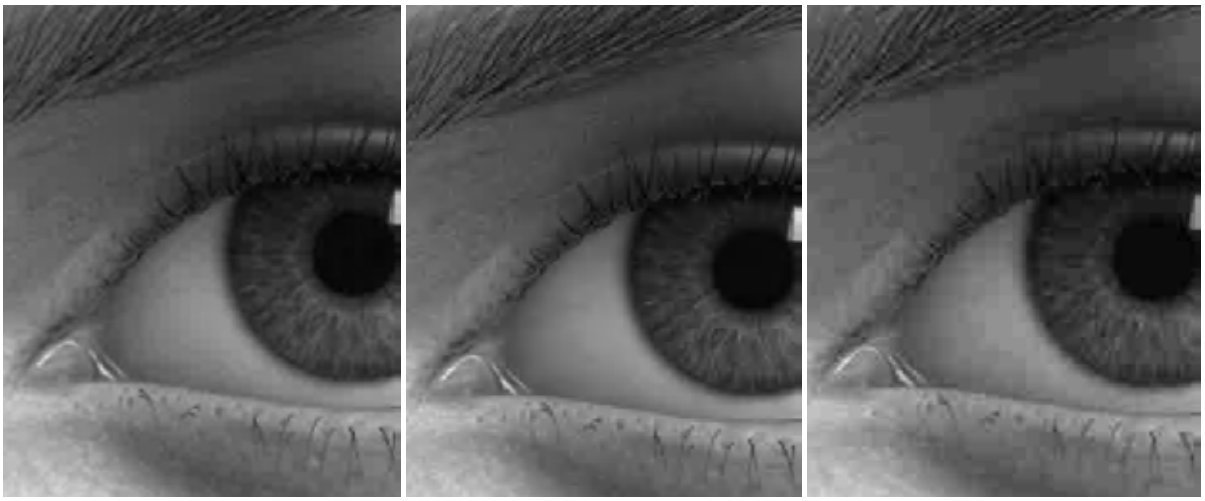
Figure-15, (a – l) Comparison the decompressed X-ray image by our Approach with JPEG2000 and JPEG



(a) Decoded by **Our Approach** Quality=0.02, PSNR=38.2 dB

(b) Decoded by **JPEG2000** Quality=77%, PSNR=39.4 dB

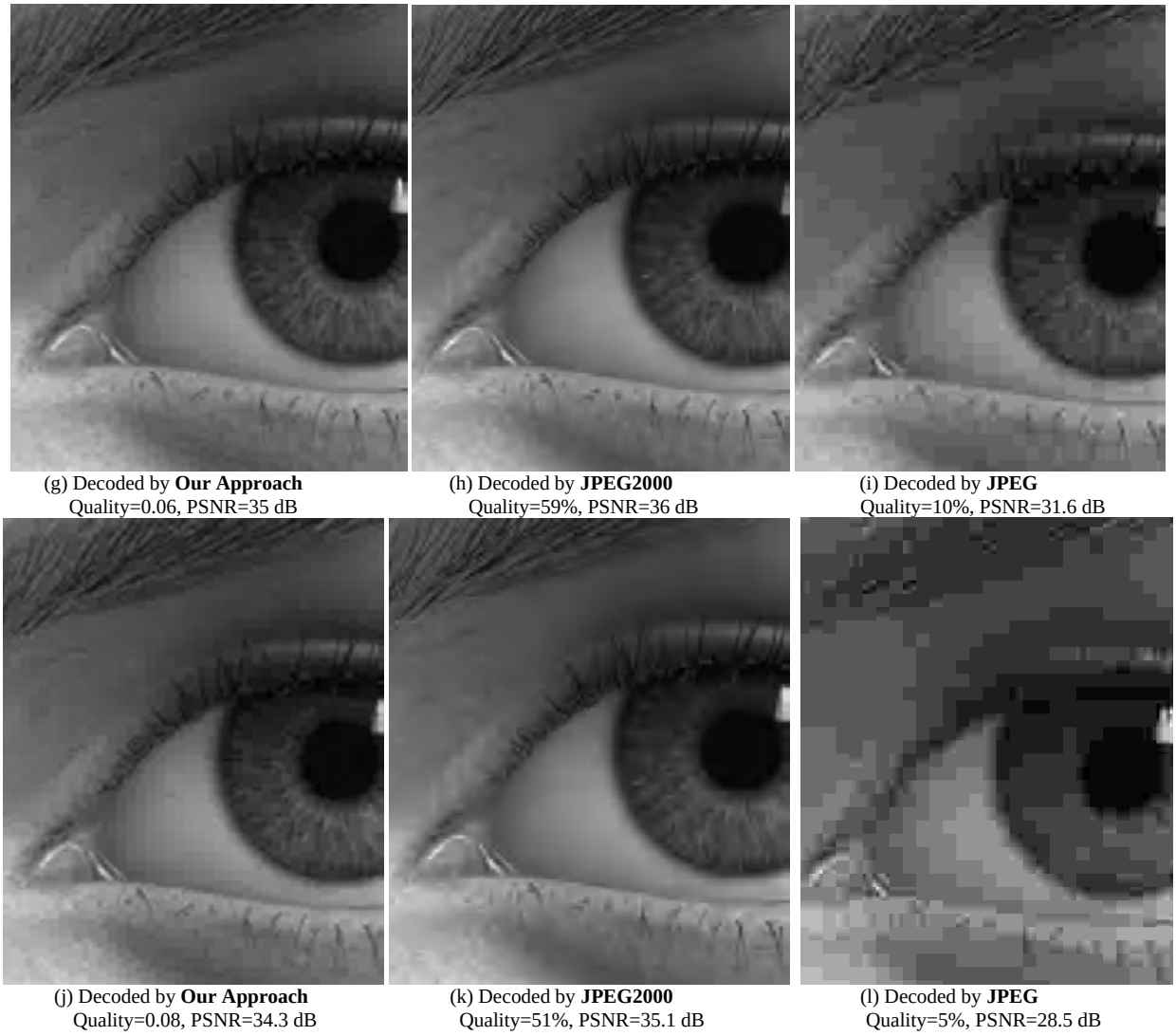
(c) Decoded by **JPEG** Quality=40%, PSNR=37.1 dB



(d) Decoded by **Our Approach** Quality=0.04, PSNR=36 dB

(e) Decoded by **JPEG2000** Quality=67%, PSNR=37.3 dB

(f) Decoded by **JPEG** Quality=17%, PSNR=33.7 dB



(g) Decoded by **Our Approach** Quality=0.06, PSNR=35 dB

(h) Decoded by **JPEG2000** Quality=59%, PSNR=36 dB

(i) Decoded by **JPEG** Quality=10%, PSNR=31.6 dB

(j) Decoded by **Our Approach** Quality=0.08, PSNR=34.3 dB

(k) Decoded by **JPEG2000** Quality=51%, PSNR=35.1 dB

(l) Decoded by **JPEG** Quality=5%, PSNR=28.5 dB

Figure-16, (a – l) Comparison the decompressed Iris image by our Approach with JPEG2000 and JPEG

7. Conclusions

This research introduces a proposed algorithm for image compression, based on; the transformations (DWT and DCT) and Minimize-Matrix-Size algorithm. This research has some advantages illustrated in the following steps:

- Using two transformations, this helped our compression algorithm to increases number of high-frequency coefficients, and leads to increases compression ratio.
- Minimize Matrix Size algorithm it is used to collect each three coefficients from the high-frequency matrices, to be single floating point value. This process converts a matrix into array, and this leads to increases compression ratio, and in another hand keep the quality of the high-frequency coefficients.
- The properties of the Daubechies DWT (db3) help our approach to obtain higher compression ratio. This is because the Daubechies DWT family has the ability to zooming-in, and not necessarily needs for the high-frequencies sub-bands [7,12].
- LSS-Algorithm is represents the core of our decompression algorithm, which convert a one-dimensional array into a matrix, depending on the Random-Weights-Values.
- The Random-Weights-Values and Limited-Data are represents as a key for coding and decoding an image, without it, the images cannot be reconstructed.
- The DWT divide an image into 4 subbands, each subband compressed independently. This makes the transmission easy, and takes less execution time for coding and decoding.
- LSS-Algorithm it's represents lossless decompression, depending on the Limited-Data (i.e. probability of the data). To find exact original data.

- Our approach gives good visual image quality at higher compression ratio, more than JPEG. This is because our approach removes most of the degradation that caused by 8x8 two-dimensional DCT at JPEG technique. Also our approach removes some blurring that caused by JPEG2000.
- One-dimensional DCT with size *n=3*, more efficient, this helped our decompression approach, to obtains good quality image more than JPEG.

Also this research has some disadvantages illustrated in the following steps:

- The differences between JPEG2000, JPEG and our approach are; JPEG2000 and JPEG faster than our approach, when obtains high image quality. This is because LSS-Algorithm iterations make our approach slower than JPEG and JPEG2000.
- The compressed header data contains floating point arrays, and this increasing header size for the compressed image size.

8. References

[1] G.Sadashivappa and K.V.S.Ananda Babu. PERFORMANCE ANALYSIS OF IMAGE CODING USING WAVELETS. *IJCSNS International Journal of Computer Science and Network Security.* 2008, **8**(10).

[2] Ali Al-Haj. Combined DWT-DCT Digital Image Watermarking. *Science Publications, Journal of Computer Science.* 2007, **3** (9): 740-746.

[3] Grigorios D. , N. D. Zervas, N. Sklavos and Costas E. Goutis. Design Techniques and Implementation of Low Power High-Throughput Discrete Wavelet Transform Tilters for JPEG 2000 Standard",*WASET* . *International Journal of Signal Processing.* 2008, **4**(1).

[4] K.Sayood. *Introduction to Data Compression*. 2nd edition. Academic Press, Morgan Kaufman Publishers, 2000.

[5] N. Ahmed, T. Natarajan and K. R. Rao. Discrete cosine transforms. *IEEE Transactions Computer*. 1974, **C-23**: 90-93.

[6] Tsai, M. and H. Hung. DCT and DWT based Image Watermarking Using Sub sampling. *in Proc. Of the 2005 IEEE Fourth Int. Conf. on Machine Learning and Cybernetics.* 2005, pp: 5308-5313.

[7] Rafael C. Gonzalez, Richard E. Woods. *Digital Image Processing.* Addison Wesley publishing company, 2001.

[8] K. R. Rao, P. Yip. Discrete cosine transform: Algorithms, advantages, applications. *Academic Press, San Diego, CA.* 1990.

[9] S. Esakkirajan, T. Veerakumar, V. Senthil Murugan, and P. Navaneethan. Image Compression Using Multiwavelet and Multi-stage Vector Quantization. *International Journal of Signal Processing.* 2008, **4**(4).

[10] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Trans. on Image Processing*. 1992, **1**(2): 205–220.

[11] 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*. 2000, **7**(9).

[12] I. E. G. Richardson. *Video Codec Design*. John Wiley & Sons, 2002.

[13] T. Acharya and P. S. Tsai. *JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures.* New York: John Wiley & Sons, 2005.

[14] Sana Ktata, Kaïs Ouni, and Noureddine Ellouze. A Novel Compression Algorithm for Electrocardiogram Signals based on Wavelet Transform and SPIHT. WASET. *International Journal of Signal Processing*. 2009, **5**(4).

[15] M.S. Joshi, R.R. Manthalkar and Y.V. Joshi. Image Compression Using Curvelet, Ridgelet and Wavelet Transform, A Comparative Study. *ICGST International Journal on Graphics, Vision and Image Processing (GVIP)*. 2008, **8**(III).