**Wasit Journal for Science & Medicine** **2011 4 (1): ( 1 - 11 )**

# Cosine Transform and Shift Number Coding uses for 24-Bit Bitmap image Coding

Ammar O. Hassan Al-barznji
Computer Science Deprat. -Science Education College -Salahhaden University

## Abstract:

This paper represents an approach for image coding which consists of three parts: The first part is DCT which is used one dimensional discrete cosine transform converts each (n-pixel) into frequency domain, then reduces number of frequency domain elements by deleting half number of elements. The second part applies SNC (Shift Number Coding) on the remained coefficient elements, these elements are converted to integer number. SNC does not need to compute the probability for an image file, just compute the values for each coefficient in frequency domain matrix. The value is computed by selecting maximum number in matrix, then it divides the interval {0…1} by maximum number. This value is shifted by shift function then sum with previous value, the total values represent Compression Value. Finally the arithmetic coding used to compress the matrix of compression value into stream of bits. This approach is tested with three types of images, and compared with JPEG, PNG, and TIFF by using Compression Ratio and PSNR.

**Keywords:** ***Discrete Cosine Transform, Shift Number Coding, Arithmetic Coding.***

## 1. Introduction

The transport of images across communication paths is an expensive process and image compression provides an option for reducing the number of bits in transmission[1,2]. This in turn helps increase the volume of data transferred in a space of time, along with reducing the cost required. It has become increasingly important to most computer networks, as the volume of data traffic has begun to exceed their capacity for transmission. Traditional techniques that have already been identified for data compression include: Predictive coding, Transform coding and Vector Quantization[3.4]. In brief, predictive coding refers to the decorrelation of similar neighbouring pixels within an image to remove redundancy. Following the removal of redundant data, a more compressed image or signal may be transmitted[5,6]. Transform-based compression techniques have also been commonly employed. These techniques execute transformations on images to produce a set of coefficients. A subset of coefficients is chosen that allows good

data representation(minimum distortion) while maintaining an adequate amount of compression for transmission. The results achieved with a transform-based technique is highly dependent on the choice of transformation used (Cosine, Wavelet, Karhunen-Loeve,….etc.)[7,8].

Finally, vector quantization techniques require the development of an appropriate codebook to compress data. Usage of codebooks does not guarantee convergence and hence do not necessarily deliver infallible decoding accuracy. Also the process may be very slow for large codebooks as the process requires extensive searches through the entire codebook[9]. Following the review of some of the traditional techniques for image compression, it is possible to discuss some of the more recent techniques that may be employed for data compression.

## 2. Compression Approach

This paper introduces an approach for (24-Bitmap) image coding by using DCT and SNC. The first part of this approach it is most popular uses in image compression, converting each n-pixel to frequency domain, then eliminates half data from the frequency domain to reduce number of coefficients. The second part it's new algorithm for image compression, introduce in this paper, which receive frequency domain coefficients to convert it into floating point numbers. Finally the floating point numbers are coded into binary code by arithmetic coding algorithm. In Figure - 1 shows the image compression for this paper.

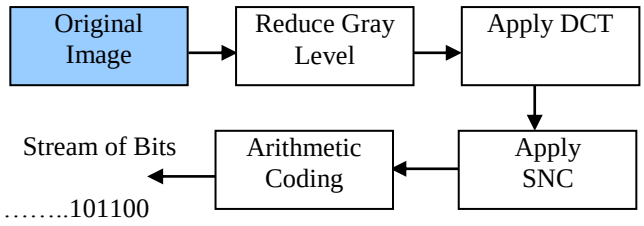



**Figure – 1, DCT and SNC for Image Compression**

## 2.1 Discrete Cosine Transform

The implementation of DCT begins with reducing pixels levels from 256-level into less than 256 gray levels by using the following equation [7]:-

$$Pixels_{(new)} = \frac{(Pixels_{(old)} * reduce)}{256} \qquad (1)$$

Reduction of the pixels levels leads to reduction the pixels size, and to simplify converting pixels into single floating point value. The discrete cosine transform (DCT) helps separate the image into parts. The DCT is similar to the discrete

Fourier transform: it transforms a signal or 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. This implies lower computational complexity, which sometimes important for real-time applications [9,12]. 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. The following equation represents one dimensional DCT:-

$$X(u) = \sqrt{2/n}\, C(u) \sum_{i=0}^{n-1} x(i) \cos \frac{(2i+1)u\pi}{2n} \qquad (2)$$

$$\text{Where } C(u) \begin{cases} = 2^{-1/2}, \text{ if } u = 0 \\ = 1 \text{ , if } u > 0 \end{cases} \quad \text{for } u=0,1,2,\ldots,n-1$$

In the above equation the *X(u)* represents the frequency domain of the pixels *x(i)*, and the *n* is number of pixels. Let's assume that the number of pixels eight (*n=8*) the pixels are :{ 11, 22, 33, 44, 55, 66, 77, 88}.Using equation (1) to calculate one-dimensional DCT, creates the eight coefficients {140,-71, 0,-7, 0,-2, 0, 0}. Deleting the last two nonzero coefficients to be the sequence: {140,-71,0,0,0,0,0,0}. This means we not needs for zeros and the eights pixels are reduces in two coefficients 140,-71. Then applying IDCT (Inverse Discrete Cosine Transform) on the two coefficients (140,-71) to return eight pixels, and the sequence of pixels are: {15 , 20 , 30 , 43 , 56 , 69,79, 84}. These pixels are approximately same as original pixels. The IDCT as shown below:

$$x(i) = \sqrt{2/n} \sum_{u=0}^{n-1} C(u)\, X(u) \cos \frac{(2i+1)u\pi}{2n} \qquad (3)$$

$$\text{Where } C(u) \begin{cases} = 2^{-1/2}, \text{ if } u = 0 \\ = 1 \text{ , if } u > 0 \end{cases} \quad \text{for } u=0,1,2,\ldots,n-1$$

## 2.2 Shift Number Coding

The implementation of (SNC) begins with selects maximum coefficient in DCT matrix, this coefficient is called "MAX", then divide the interval {0….1} into MAX, in this way the all coefficients are converted into floating point between {0..1}. This means the interval between any two sequential data is 1/MAX, and this interval constant for all types of images. For this reason this approach does not need to compute the probabilities for an image file each data has specific Value, and each Value computing from the List -1.

**List -1**

```
Set Value = 0.0;
MAX=Select_Maximum_Number(DCT);
For Symbol =0 to MAX Do
  Interval (Symbol) =Value;
   Value =Value + (1/MAX);
End// for
```

SNC converting number of bytes into floating point number, the following equation used by SNC approach:

$$Compression\ Value = \sum_{i=1}^{N} Shift(i) * Value(i) \qquad (4)$$

The final value of equation(4) it is floating number, function "Shift(i)" is used to shift the value to the left as shown in List -2, and compression algorithm by SNC is shown in List -3 respectively:

**List -2**

```
Shift(1)=1.0;
For i=2 to number of data
 Shift(i)=Shift(i-1)*0.01;
End //for
```

**List -3**

```
Set S  to 0.0;
Set index to 1;
While (index ≤N) Do
    Read (symbol)
    Value = interval (Symbol);
    S = S + Shift (index) * Value;
    Index = index + 1;
End //While
Compression_Value=S;
```

To illustrate the compression algorithm by DCT and SNC, assumes the following pixels reduced by equation (1).

$$\mathbf{T}=\begin{bmatrix}129, 134, 255, 189\\ 100, 99, 156, 50\end{bmatrix} \longrightarrow \text{Reduce matrix by equation (1)}$$

$$\mathbf{T}=\begin{bmatrix}30, 31, 60, 44\\ 23, 23, 37, 12\end{bmatrix} \longleftarrow$$

The first step applies DCT (i.e. equation (2) ) on the above data by take each two data as vector, which means the matrix "T" converted to frequency domain by DCT, the Figure-2 shows how the special domain matrix converts to frequency domain matrix.

$$[30\ 31] \xrightarrow{DCT} [43\ \ -1]$$
$$[60\ 44] \xrightarrow{DCT} [74\ \ 11]$$
$$[23\ 23] \xrightarrow{DCT} [33\ \ 0]$$
$$[\ 37\ 12] \xrightarrow{DCT} [35\ \ 18]$$

(a)

$$\mathbf{C}=\begin{bmatrix}43 & -1\\ 74 & 11\\ 33 & 0\\ 35 & 18\end{bmatrix}$$

(b)

**Figure – 2 (a) Convert each two pixels to frequency domain, (b) Final Cosine transform matrix**

At the end of this part we eliminates the data {-1,11,0,18} from matrix "C". This step reduce number of coefficient to half, and the remaining coefficients are {43,74,33,35}, is shown in the following figure:

$$\mathbf{C}=\begin{bmatrix}43 & 0\\ 74 & 0\\ 33 & 0\\ 35 & 0\end{bmatrix} \xrightarrow{delete\ last\ columns} \mathbf{C}=\begin{bmatrix}43\\ 74\\ 33\\ 35\end{bmatrix}$$

(a) (b)

**Figure – 3 (a) frequency domain matrix containing 8- coefficients (b) Matrix "C" containing 4- coefficients**

The 4-coefficients in matrix "C" are used by SNC (i.e. equation (4)) to convert the matrix "C" into integer number. We see in matrix "C" the maximum coefficient is "MAX=74" and increment MAX by "1" to be MAX=75, to generate the interval 1/75 = 0.013, from this interval we generate values for all data in matrix these values in range{0…1}. The Table 1 shown the SNC algorithm

Table 1 SNC Compression

| Index | Data | Value | Shift(index) | S |
|---|---|---|---|---|
| **1** | 43 | 0.559 | 1 | 0.559 |
| **2** | 74 | 0.962 | 0.01 | 0.56862 |
| **3** | 33 | 0.425 | 0.0001 | 0.5686625 |
| **4** | 35 | 0.455 | 0.000001 | 0.568662955 |
| **Compression Value =** | | | | 0.56866 |

In the above table the final value (Compression Value = 0.56866), we see from the result that the "Compression Value" consist of five digits, because the length of "Compression Value" affects on data compression size when convert it into binary code. Table 1 shows the final result stores as integer (56866) which its size 16-Bit, this means our approach reducing 8-Bytes into 16-Bits. In this example not use the "**Arithmetic Coding**", but we use it in Section 4.

To compute the efficient of this algorithm by using Compression Performance (C.P.) as it's shown in the following equation [4,7]:-

$$Compression\ Rate = \frac{Size\ after\ compression}{Size\ before\ compression} \quad (5)$$

$$C.P. = [100(1 - Compression\ Rate)]\% \quad (6)$$

The compression performance for above example is 75% this means this algorithm save 75% from data and this result for our approach is good. The decompression for this approach can be illustrates in the following figure:-

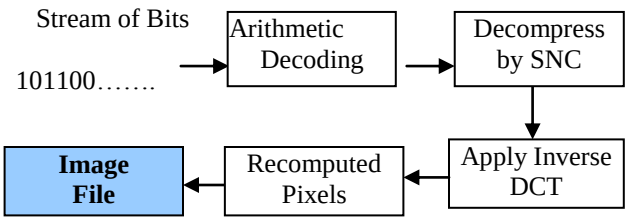



**Figure – 4 Decompress SNC and IDCT for image Decompression**

In the decompression algorithm returns the approximately original frequency domain coefficients, by using SNC. List -4 illustrates decompressions for SNC, and Table 4 shown decompression steps.

**List -4**

```
N=number of data – 1;
S=Compression_Value;
While ( N>0) Do
  For I =0 to MAX Do
     IF Value(I) ≤ S < Values(I+1) THEN
        Let  K=Value( I );
```

```
            N=N-1
            Pixel=Get_Pixel ( I );
        End //IF
    End //For
     S=S - K;
     S=S * 100;
End //While
```

SNC decompression algorithm start by search "Compression value" between any two sequential values, if the "Compression value" found between any two values, subtract minimum value from "Compression value". The minimum value represents the frequency domain coefficient. The algorithm repeats to search for "Compression value" between values until number of data reaches to zero or Compression Value less than interval (i.e. interval=1/MAX). The decompression algorithm for SNC can represent in Table 2.

Table 2 Decompression for SNC

| Compression Value | Compare "S" Between two sequence Values | Selected Data | N |
|---|---|---|---|
| 0.56866 | [0.559 – 0.572] | 43 | 3 |
| 0.966 | [0.962 – 1.0] | 74 | 2 |
| 0.40 | [0.39 – 0.403] | 30 | 1 |
| 0.01 | Stop | - | 0 |

After decompress "*Compression Value*" generating the pixels {43, 74, 30}. But its three frequency domain data, because the N = 0 and "Compression Value" less than interval, for this reason the last data (i.e. 3rd data) made copy at 4th location, and final coefficients are {43, 74, 30, 30}.

From the Table 4 we construct the frequency domain matrix "C", by padding the last column with zeros, then using Inverse DCT (See equation (3)) to get image pixels. The equation (3) applied on each vector from matrix "C". The Figure - 5 illustrates Inverse DCT.

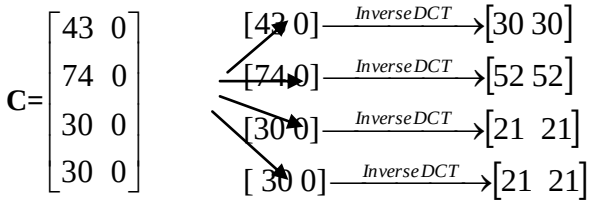



**Figure – 5 Constructs image pixels by Inverse DCT**

The constructed matrix by our approach as shown below:-

$$\mathbf{T}=\begin{bmatrix}30 & 30\\ 52 & 52\\ 21 & 21\\ 21 & 21\end{bmatrix} \longrightarrow \text{Inverse equation (1)}$$

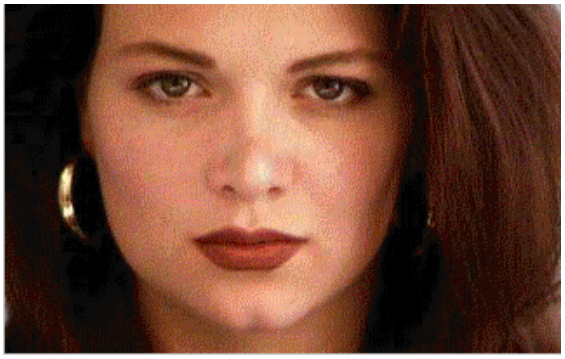
$$\mathbf{T}=\begin{bmatrix}128, 128, 222, 222\\ 90, 90, 90, 90\end{bmatrix}$$

The above matrix "T" represents final matrix, all data in this matrix not same in original matrix, this difference not effects on image brightness, for this reason this approach is very interactive for image compression.

## 3. Computer Simulation

This approach implemented on computer *Pentium4 -2.4GHz* with **Visual C++.NET**, and the images are used for testing as shown in the figure-6

(a) Original **Image1** (256 x 256)



(b) Original **Image2** (320 x 208)

(c) Original **Image3** (352 x 352)

**Figure – 6 (a) Gray level image 24-Bits, (b) Color image 24-Bits, (c) Color image 24-Bits.**

The images in Figure - 6 are compressed by our approach; at first pixels reduced by equation (1) using "reduce=20". The number of pixels used by equation(2) is (2 x 4 - pixels) to be compressed into four frequency domain coefficients. The SNC compressing each (4- coefficient) into integer number by equation(4), the performance for our approach shown in Table 3.

**Table 3 Compression Results by our appraoch**

| File name | Before Compression | After Compression | C.P. |
|---|---|---|---|
| **Image 1** | 192-KByte | 47.3-KByte | 75% |
| **Image 2** | 195-KByte | 41.5-KByte | 78% |
| **Image 3** | 363-KByte | 99-KByte | 73% |

## 3.1 Arithmetic Coding

We add Arithmetic coding to our approach, to compress again, by taking stream of data (i.e. compressed data from SNC) and convert it to stream of floating point value. These output values in range less than one and greater than zero[7], when decoded these values 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[10]. The final compressed matrix from SNC, are compressed again by arithmetic coding, as shown in the Table 4, and arithmetic coding algorithm is shown in List -5:-

**List – 5**

```
Set Low =0.0;
Set High = 1.0;
While (not reached end of data)
  data = read_data_from_vector ();
    Range = High – Low;
   High = Low + Range * High_Range(data);
   Low = Low + Range * Low_Range(data);
End;
```

**Table 4 Compression Results with Arithmetic coding**

| File name | Before Compression | After Compression | C.P. |
|---|---|---|---|
| **Image 1** | 192-KByte | 31-KByte | 84% |
| **Image 2** | 195-KByte | 26.5-KByte | 86% |
| **Image 3** | 363-KByte | 52.3-KByte | 85% |

From the Table 6 the performance for our approach increased by using arithmetic coding, and this is because the arithmetic coding able to convert group from data into single floating point[12]. The Decompression by arithmetic coding has shown in List - 6.

**List -6**

```
While (Low >=0 )
 For I = 0 to 10
 IF(Low>Low_Range(I)ANDLow<=High_Range (I)) THEN
      Range = High_Range(I) – Low_Range(I);
               Low = Low – Low_Range(I);
               Low = Low / Range;
        End // if
    End // for
  End // while
```

In the decompression part returns all data from stream of bits, by using arithmetic decoding. And SNC convert each integer number into four frequency domain coefficients (See List-4), then generates (2 x 4-pixels) by using inverse DCT (See equation(3) ) the Figure - 7 shown the decompressed images. Also the Table 5 shows the time execution for compression and decompression of images.

(a) Decompress **Image1** (256 x 256)

(c) Decompress **Image3** (352 x 352)

(b) Decompress **Image2** (320 x 208)

Figure – 7 (a– c) Decompressed images by our approach

**Table 5 Time execution for our approach**

| File name | Compression Time (min:sec: $\mu$ sec) | Decompression Time (min:sec: $\mu$ sec) |
|---|---|---|
| Image 1 | 00:24:90 | 00:35:11 |
| Image 2 | 00:27:69 | 00:35:47 |
| Image 3 | 00:52:23 | 01:00:51 |

## 3.2 Comparison methods

Our approach is compared with JPEG, PNG and TIFF. These method are popular use in image compression specially when transmits through internet [9]. The JPEG is lossy data compression scheme for color and gray-scale images. It works on full 24-bit color, and was designed to be used with photographic material and naturalistic artwork, also the JPEG can produce smaller file than PNG and TIFF for photographic images since it uses lossy encoding method specifically designed for photographic image data [10].

The TIFF and PNG use lossless data compression methods, which are used in compression of library. The most common general-purpose, lossless compression algorithm used with TIFF is LZW, which is inferior to PNG [11]. There is a TIFF variant that uses the same compression algorithm as PNG uses, but it is not supported by many proprietary programs. TIFF also offers special-purpose lossless compression algorithms like CCITT Group IV, which can compress bi-level images (e.g., faxes or black-and-white text) better than PNG compression algorithm [12]. The Table 6 has shown the comparison our approach with JPEG, PNG and TIFF using PSNR.

**Table 6 Comparison with author type of comprised image**

| Method | Image Name | Compression | | C.P. | PSNR |
|---|---|---|---|---|---|
| | | Before | After | | |
| JPEG | Image1 (256 x 256) | 192-KB | 12.8-KB | 93% | 33.5 |
| | Image 2 (320 x 208) | 195-KB | 10.8-KB | 94% | 31.3 |
| | Image 3 (352 x 352) | 363-KB | 19-KB | 95% | 39.7 |
| Our Approach | Image1 (256 x 256) | 192-KB | 36.4-KB | 81% | 36.8 |
| | Image 2 (320 x 208) | 195-KB | 26.5-KB | 86% | 37.1 |
| | Image 3 (352 x 352) | 363-KB | 52.3-KB | 85% | 37.0 |
| PNG | Image1 (256 x 256) | 192-KB | 77.8-KB | 59% | Inf |
| | Image 2 (320 x 208) | 195-KB | 132-KB | 32% | Inf |
| | Image 3 (352 x 352) | 363-KB | 280-KB | 22% | Inf |
| TIFF | Image1 (256 x 256) | 192-KB | 101-KB | 47% | Inf |
| | Image 2 (320 x 208) | 195-KB | 166-KB | 15% | Inf |
| | Image 3 (352 x 352) | 363-KB | 380-KB | NON | Inf |

Peak signal to noise ratio (PSNR) can be calculated very easily and is therefore a very popular quality measure it is shown in the following equation [13]:
For TIFF section Image 3 C.P was NON because the Image size was the same and the compression was very few .

$$\mathrm{PSNR} = 10 \log_{10} \frac{255^2}{\mathrm{MSE}} \qquad (7)$$

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).

## 4. Conclusions

This research represents new approach for image compression, by using three algorithms. The first algorithm is DCT it is most popular used in digital signal processing especially in image compression, the second part SNC new algorithm represented in this research, the idea of SNC converting n-bytes to an integer number. Finally the matrix of integer numbers compressed by Arithmetic Coding to get stream of bits, the advantage of this approach: (1) It does not needed to compute the probability for an image, (2) The DCT reduce the n-bytes to half (i.e. n/2 - bytes) and also SNC reduce the remainder bytes to half, and this process leads to compressing big part from image (See Table 4). (3) The compression value by arithmetic coding represented in binary code. the arithmetic coding playing main role for image compression in this research. (4) Our approach gives better compression performance than PNG and TIFF, also our approach gives better image quality than JPEG (See Table 6).The disadvantage for this approach takes more time for compression, because more computations by DCT, SNC and Arithmetic coding. Also the decoding by arithmetic and SNC may leads to increasing time execution for decompression (See Table 3). Our approach represents lossy data compression especially eliminates half coefficients from DCT matrix, this means for gain high-quality images the PNG and TIFF is stronger than our approach, because these methods use lossless data compression methods which are used in compression of library[10].

**Notes:**

1- The comparison for jpg images is Lossy and my comparison was lossy to, the comparison was Lossy with Lossy

2- The other ways was Lossless and it was added to the research in order to compare it with the lossless but in other hand the file size will be much bigger.

3- The TIFF file had become bigger therefore I wrote NON in the table because the C.P was zero.

4- The formula (7) shows MSE and how it affects the PSNR, therefore when MSE become Zero, then any number divided by Zero the result will be infinity ($\infty$) and I wrote it Like (**inf)** which is lossless

## References


**[1]** Castelli. V., Robinson j. and Turek j.j., Multiresolution lossless/lossy compression and storage of data for efficient processing thereof, U.S. Patent No. 6,141,445. Octobar, 2000.

**[2]** Dembo A. and Kontoyainnis I., "Source coding large deviations and approximate pattern matching", Invited paper in IEEE Trans. Theory special issue on Shannon theory, dedicated to Aaron D. Wyner, 48. pp. 1590 - 1615, June, 2002.

**[3]** Harrison M. and Kontoyiannis I., "Maximum likelihood estimation for lossy data compression", (Invited paper). 40th Allerton Conference on Communication, Control and Computing, Allenton, IL, October, 2002.

**[4]** James. E. F. and Beatrice Pesuet-Popescu, "An Overview on Wavelets in Source Coding, Communication, and Networks," EURSIP Journal on Image and Video Processing, Vol 2007.

**[5]** Kontoyainnis I. and Meyn S. P., "Large deviation asymptotic and the spectral theory of multiplicatively regular Markov processes.", Electronic Journal of probability,No.10, paper 3 pp. 61-72, February, 2005.

**[6]** Kontoyainnis I., "Pointwise redundancy in lossy data compression and universal lossy data compression", IEEE Trans. on Inform. Theory. 46. pp. 136-152, January, 2000.

**[7]** Nelson, M., The Data Compression Book, M & T Publishing Inc, 1991.

**[8]** Pennebaker W.B. and Mitchell J.L., JPEG still image data compression standard. Van Nostrandm Reinhold, 1992.

**[9]** Siddeq M. M., "SEQUNCE DYNAMIC CODE FOR STREAM TWO BYTE DATA (16-Bit) COMPRESSION", *Al-Taqani Journal*,Vol. 19,No.2,pp 85-99, 2006

**[10]** Sayood, Khalid, "Introduction to Data Compression", San Francisco, Morgan Kaufmann, 2000.

**[11]** Usevitch B. E., "A tutorial on modern lossy wavelet image compression: foundations of JPEG2000," IEEE Signal Processing Mag., vol. 18, no. 5, pp. 22–35, September 2001.

**[12]** Witten, H. I., Neal, Radford M., and Cleary, John G.," Arithmetic Coding for Data Compression", Communications of the ACM, pp 520-540, June, 1987.

**[13]** Xiong Z., Wu X., Cheng S., and Hua j., "Loss-to-lossless compression of medical volumetric data using three-dimensional integer wavelet transforms", IEEE Trans. on Medical Imaging, Vol. 22, No. 3, March 2003.



**Recived ……..……. (1/2 /2010 )**
**Accepted ……...……... (20/3/2010 )**