ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 4, No. 3, 2009, *pp*. 311-320

Using Shift Number Coding with Wavelet Transform for Image Compression

Mohammed Mustafa Siddeq [+]

Software Engineering Depart, Technical College –Kirkuk – Iraq

(*Received March 10, 2009, accepted September 24, 2009*)

Abstract: This paper introduces an idea for image compression; consist from two parts; the first part used single stage Discrete Wavelet Transform (DWT), which produces four sub matrices; LL, HL, LH, and HH. The two sub matrices HL and LH are merged to be one matrix "Merged - Matrix", and store non-zero data in array "Non-Zero-Array", then compress it by RLE and Arithmetic coding. In the second part the sub-matrix LL coefficients converted into data at range {0…80}, LL sub matrix used by Shift Number Coding (SNC), this algorithm converts each four data into single floating point number, which stored in one dimensional array "SNC-Array", finally apply arithmetic coding on it. Our approach compared with JPEG, and JPEG-2000, by using compression ratio and PSNR.

Keywords: Discrete Wavelet Transform, Shift Number Coding, Run-Length-Encoding.

1. Introduction

The JPEG and the related MPEG format make good real-world examples of compression because (a) they are used very widely in practice, and (b) they use many of the compression techniques we have been talking about, including Huffman codes, arithmetic codes, residual coding, run-length-coding, scalar quantization, and transform coding [1-3]. JPEG is used for still images and is the standard used on the web for photographic images. JPEG is designed so that the loss factor can be tuned by the user to tradeoff image size and image quality, and is designed so that the loss has the least effect on human perception [4-7,11]. It however does have some anomalies when the compression ratio gets high, such as odd effects across the boundaries of 8x8 blocks. For high compression ratios, other techniques such as wavelet compression appear to give more satisfactory results. The wavelet transform has emerged as a cutting edge technology, within the field of image compression [5-9]. Wavelet-based coding provides substantial improvements in picture quality at higher compression ratios. Over the past few years, a variety of powerful and sophisticated wavelet-based schemes for image compression, as discussed later, have been developed and implemented. Because of the many advantages, the top contenders in the upcoming JPEG-2000 standard are all wavelet-based compression algorithms [13].

This paper describes image compression consists form two parts: (1) using Discrete Wavelet Transform (DWT), and (2) using Shift Number Coding (SNC). The DWT decomposes an image into four-sub images; "LL", "HL, "LH", and "HH", and the second part in this paper using SNC applied on sub-image "LL", which convert each four data into single floating point number. Also in this paper we describe comparison with JPEG, and JPEG-2000.

2. 2. Compression Algorithm

In this research we introduce an idea for image compression, depending on SNC and DWT. The DWT decomposes an image into four-sub images; consist from low-frequency, and high-frequencies "LL", "HL, "LH", and "HH". The SNC used to encode the "LL" matrix into floating point number, by taking each four data from "LL" matrix to be converted into single floating point number. The matrices "HL" and "LH" are merged into a single matrix, and scan (Row-by-Row) for non-zero data, finally these data are coded by Run Length Encoding (RLE), and Arithmetic Coding. Fig -1 illustrated our algorithm.

[+] Author Tel.: +9647701256324
Email: - mamadmmx76@yahoo.com

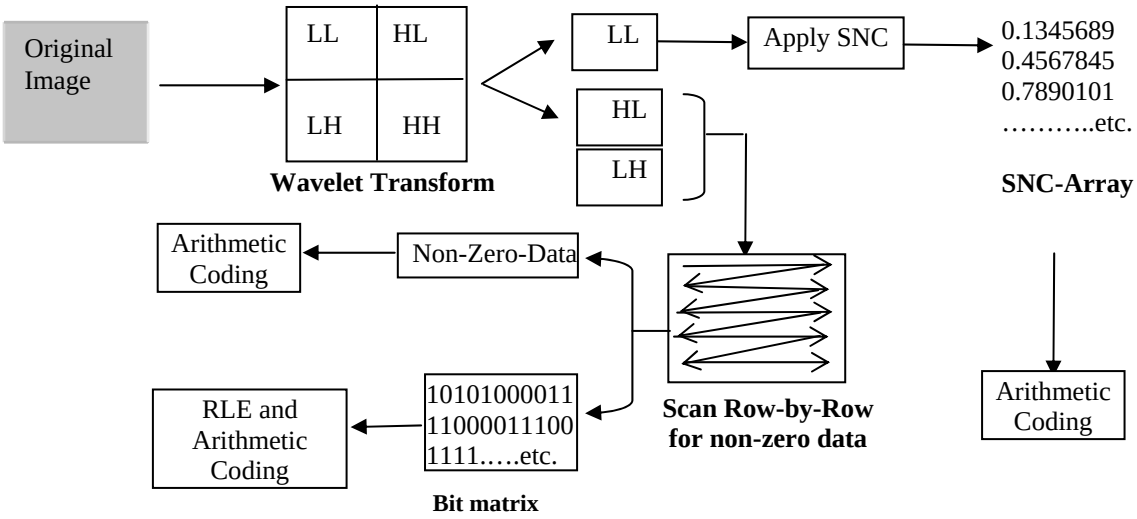


Fig -1 Compression Algorithm

2.1. Shift Number Coding (SNC)

Before applying SNC on matrix "LL", we reduce data of the matrix "LL" into range between {0...79} by using the following equation [1,2]:

$$Data_{(new)} = \frac{(Data_{(old)} * Threshold)}{MAX} \tag{1}$$

The reduction of data depend on Threshold=80, and MAX (i.e. maximum data in sub matrix "LL"). The new range of matrix "LL" becomes between {0 – 79}, the main reason for choosing this small range, to get maximum compression ratio.

SNC algorithm begin by assigning a value for all new data (i.e. new data for matrix "LL") where the interval of these values between {0 - 1}. Each data divided by 256, which means the interval between any two sequential values, is 0.0039. The SNC algorithm compresses each four data into single floating point by shifting each value, and total values called "*Compressed Value*". SNC do not need to compute the probability for an image and does not need to store information about compressed image. The output for SNC is a one dimensional array contains floating point numbers called "SNC-Array". The following equation is used by our algorithm:

$$Compression\ Value = \sum_{i}^{n} Shift(i) * Value(i) \qquad \text{For } i = 1,2\ldots4 \tag{2}$$

The value of "Compression Value" it's single floating value, and the "Shift(i)" it's used to shift the value according to index of data. The shift function can be illustrates in the Table 1.

Table 1 Shift function

Index	Shift(index)
1	1
2	0.01
3	0.0001
4	0.000001

The Compression algorithm for each 4-data is shown below:

Set Compression_Value to 0.0;
Set index to 1;

DOI https://doi.org/2024-JICS-22741 | Generated on 2025-04-09 07:03:22

While (index ≤ 4) **Do**
  Read (symbol)
  Value = **interval** (Symbol);
  Compression_Value = Compression_Value + Shift (index) * Value;
  Index = index + 1;
End *While*

2.2. Discrete Wavelet Transform (DWT)

In this section decompose an image into four sub-images; "LL", "HL", "LH", and "HH", which represents coefficient matrix, horizontal matrix, vertical matrix, and diagonal matrix respectively, these matrices generated from single-stage DWT [2,3,5]. The matrix "LL" encoded by SNC (explained at previous section), and the other matrices "HL", and "LH" are merged into one matrix is called "Merged Matrix".

The "HH" matrix ignored in this paper, for obtaining maximum compression ratio, the Merged Matrix scan (row-by-row), and for mapping it into one dimensional array is called "Non-Zero-Array".

Also, Transform the Merged Matrix, to "Bit Matrix", this matrix contains, bits {0, 1}, the transformation done by changing every non-zero data to "1", the idea of this transformation, save position for each non-zero data.

2.3. Coding Algorithm

In this paper we use two important coding algorithms Run-Length-Encoding (RLE), and Arithmetic Coding. These algorithms are useful in image compression, especially in JPEG-2000 technique [1]. The basic concept of RLE is to code each contiguous group of 0's or 1's from left to right by scan a row (i.e. computes number of 1's or 0's in a row) [11,12].

In our algorithm RLE applied on Bit Matrix to produce a one-dimensional array contains total of 0's and 1's in a row from left to right. The RLE stores start bit 0 or 1, for example: assume the stream of bits {10000111001111}, the RLE result: {1, 1, 4, 3, 2, 4}, the first number is start bit, and second number is refer to total of 1's, and third number refer to total of 0's, and so on, This process will continues until reached to the end of "Bit Matrix". Finally the arithmetic coding applied on Non-Zero-array, and SNC-Array to produce stream of bits (See Fig -1).

The Arithmetic coding takes a stream of input data and replaces it with a single floating point output number. The output from an arithmetic coding process is a single number less than 1 and greater than or equal to 0. This single number can be uniquely decoded to create the exact stream of data [1,2,12]. For test our algorithm, assume the matrix size 8x8 as shown below:

$$X=\begin{bmatrix} 204 & 214 & 206 & 218 & 193 & 121 & 124 & 171 \\ 223 & 222 & 216 & 196 & 135 & 112 & 153 & 211 \\ 217 & 200 & 201 & 166 & 133 & 128 & 190 & 210 \\ 211 & 205 & 190 & 138 & 106 & 168 & 218 & 202 \\ 200 & 201 & 171 & 120 & 121 & 195 & 218 & 196 \\ 210 & 188 & 159 & 90 & 153 & 196 & 218 & 161 \\ 207 & 181 & 125 & 96 & 198 & 210 & 174 & 161 \\ 196 & 160 & 79 & 181 & 226 & 166 & 163 & 169 \end{bmatrix}$$

1- Using DWT to generate four sub images by MATHLAB:

[LL, HL, LH, HH] =DWT2(X, 'Haar');

$$LL=\begin{bmatrix} 432, & 418, & 281, & 330 \\ 417, & 348, & 268, & 410 \\ 400, & 270, & 333, & 397 \\ 372, & 241, & 400, & 334 \end{bmatrix} \quad HL=\begin{bmatrix} -14, & 6, & 34, & -35 \\ 1, & 20, & -7, & -10 \\ 2, & 21, & -17, & 18 \\ 16, & -19, & 8, & 2 \end{bmatrix} \quad LH=\begin{bmatrix} -4, & 4, & 48, & 53 \\ 12, & 44, & -29, & -2 \\ 10, & 60, & -59, & 40 \\ 31, & -37, & 24, & 3 \end{bmatrix}$$

The sub-matrix "HH" not used in our algorithm, this led to reduce in computation time, and this led to reducing at image quality.

1- Apply Equation (1) on "LL", and divided each data at sub-matrices "HL", and "LH" by the Quality Factor =20, this value specify image quality:

$$LL = \begin{bmatrix} 79, 76, 51, 60 \\ 76, 64, 49, 75 \\ 73, 49, 61, 73 \\ 68, 44, 73, 61 \end{bmatrix} \qquad HL = \begin{bmatrix} 1, 0, 2, -2 \\ 0, 1, 0, -1 \\ 0, 1, -1, 1 \\ 1, -1, 0, 0 \end{bmatrix} \qquad LH = \begin{bmatrix} 0, 0, 2, -3 \\ 1, 2, -1, 0 \\ 1, 3, -3, 2 \\ 2, -2, 1, 0 \end{bmatrix}$$

2- Apply Equation (2) on "LL" to generate "SNC-Array", as shown in Table 2.

Table 2 SNC algorithm

Data	**Value**	**Shift(index)**	**Compression_Value**
79	79 / 256	1	0.30859375
76	76 / 256	0.01	0.3115625
51	51 / 256	0.0001	0.3115824
60	60 / 256	0.000001	0.31158265
Data	**Value**	**Shift(index)**	**Compression_Value**
76	76 / 256	1	0.296875
64	64/256	0.01	0.299375
49	49 / 256	0.0001	0.2993941
75	75 / 256	0.000001	0.29939443
Data	**Value**	**Shift(index)**	**Compression_Value**
73	73 / 256	1	0.28515625
49	49 / 256	0.01	0.2870703
61	61 / 256	0.0001	0.2870941
73	73/256	0.000001	0.28709442
Data	**Value**	**Shift(index)**	**Compression_Value**
68	68 / 256	1	0.265625
44	44 / 256	0.01	0.2673437
73	73 / 256	0.0001	0.2673722
61	61 / 256	0.000001	0.26737250

The final array; **SNC-Array** = [0.31158265, 0.29939443, 0.28709442, 0.2673725]

3- Compress SNC-Array, first separate floating point into 2-digits, then apply arithmetic coding:

SNC-Array=[31, 15, 82, 65, 29, 93, 94, 43, 28, 70, 94, 42, 26, 73, 72, 5]

The Arithmetic Coding convert the SNC-Array to stream of bits, and its size = **66-Bits.**

4- The HL, and LH merged into one matrix (Merged Matrix), to generate Non-Zero-Array from it, also transform it to Bit Matrix, as shown below:

$$\text{Merged Matrix} = \begin{bmatrix} 1, 0, 2, -2, 0, 0, 2, -3 \\ 0, 1, 0, -1, 1, 2, -1, 0 \\ 0, 1, -1, 1, 1, 3, -3, 2 \\ 1, -1, 0, 0, 2, -2, 1, 0 \end{bmatrix} \qquad \text{Bit Matrix} = \begin{bmatrix} 1, 0, 1, 1, 0, 0, 1, 1 \\ 0, 1, 0, 1, 1, 1, 1, 0 \\ 0, 1, 1, 1, 1, 1, 1, 1 \\ 1, 1, 0, 0, 1, 1, 1, 0 \end{bmatrix}$$

Transformation

RLE =[**1**,1,1,2,2,2,1,1,1,4,2,9,2,3,1]

$$\text{Non-Zero-Array} = \begin{bmatrix} 1, 2, -2, 2, -3, 1, -1, 1, 2, -1, 1, -1, 1, 1, 3, -3, 2, 1, -1, 2, -2, 1 \end{bmatrix}$$

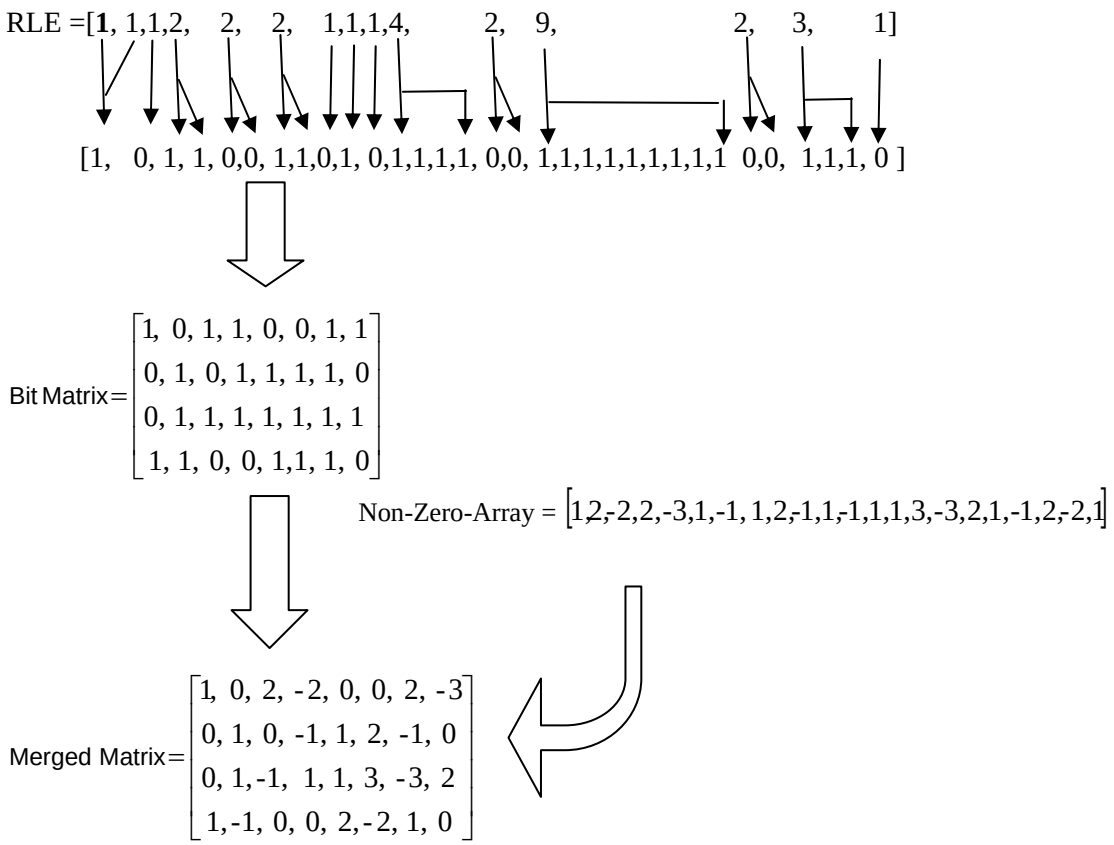
The Arithmetic coding applied on Non-Zero-Array, to produce stream of bits: 57 Bits, and also arithmetic coding applied on RLE to get: 33-Bits, the total size for compressed image by our algorithm (66

Bits +57 Bits +33 Bits) 156 Bits. We compute the compression performance for above example by the equation [1]:-

$$Compression\ Rate = \frac{Size\ after\ compression}{Size\ before\ compression} \tag{3}$$

$$Compression\ Performance = [100(1 - Compression\ Rate)]\% \tag{4}$$

The compression performance is 69.5% for the above example; this means the algorithm save 69.5% from the data. The decompression algorithm starts decoding bits to generate RLE, Non-Zero-Array, and SNC-Array. From the RLE generate Bit Matrix by taking first number, which is represents start bit "1", the following steps illustrates return Bit Matrix, and Merged Matrix:



The Bit Matrix transformed to Merged Matrix, by taking each "1-Bit" replaced with data at Non-Zero-Array, in order. Then separate Merged Matrix to get "HL" and "LH", and applying Inverse SNC on SNC-Array to return "LL". Table-3 shows Inverse SNC, and the following steps illustrated Inverse SNC:

```
For J=1 to Size (SNC-Array)
  Compression_Value=SNC-Array(J)
Number_Of_Data = 4;
While (Number_Of_Data > 0) Do
  For I =1 to Threshold Do
     IF Data(I) /256 ≤ Compression_Value< Data(I+1) /256 THEN
        Let  K=  Data(I );
        Store_Data_in_MatrixLL (Data, I);
     End //IF
  End //For
  Compression_Value=Compression_Value -K;
```

Compression_Value=**Compression_Value***100;
Number_Of_Data=Number_Of_Data - 1;
End *//While*
End *// For*

Table 3 Inverse SNC, for return LL matrix

Data Range	Value	Selected Data	Data Range	Value	Selected Data
[**79** /256 – **80** /256]	0.31158265	79	[**76** /256 – **77** /256]	0.29939443	76
[**76** /256 – **77** /256]	0.29889	76	[**64** /256 - **65** /256]	0.251943	64
[**51** /256 – **52** /256]	0.2015	51	[**49** /256 – **50** /256]	0.1943	49
[**58** /256 – **59** /256]	0.228125	58	[**74** /256 – **75** /256]	0.289375	74
Data Range	**Value**	**Selected Data**	**Data Range**	**Value**	**Selected Data**
[**73** /256 – **74** /256]	0.28709442	73	[**68** /256 – **69** /256]	0.26737250	68
[**49** /256 – **50** /256]	0.193817	49	[**44** /256 – **45** /256]	0.17475	44
[**61** /256 – **62** /256]	0.241075	61	[**73** /256 -**74** /256]	0.2875	73
[**71** /256 – **72** /256]	0.279375	71	[**61** /256 – **62** /256]	0.234375	61

After constructing the "LL", using inverse equation (1) for obtaining approximately original coefficients and multiply each data in "HL", and "LH" by Quality Factor, finally applying inverse DWT to obtain "matrix $\hat{X}$ " which is equivalent to "matrix X" , as shown below:

$$LL=\begin{bmatrix}79, 76, 51, 58\\ 76, 64, 49, 74\\ 73, 49, 61, 71\\ 68, 44, 73, 61\end{bmatrix} \quad HL=\begin{bmatrix}1, 0, 2, -2\\ 0, 1, 0, -1\\ 0, 1, -1, 1\\ 1, -1, 0, 0\end{bmatrix} \quad LH=\begin{bmatrix}0, 0, 2, -3\\ 1, 2, -1, 0\\ 1, 3, -3, 2\\ 2, -2, 1, 0\end{bmatrix}$$

$$LL=\begin{bmatrix}427, 410, 275, 313\\ 410, 346, 265, 400\\ 394, 265, 329, 383\\ 367, 238, 394, 329\end{bmatrix} \quad HL=\begin{bmatrix}20, 0, 40, -40\\ 0, 20, 0, -20\\ 0, 20, -20, 20\\ 20, -20, 0, 0\end{bmatrix} \quad LH=\begin{bmatrix}0, 0, 20, -60\\ 20, 40, -20, 0\\ 20, 60, -60, 40\\ 40, -40, 20, 0\end{bmatrix}$$

$\hat{X}$ =round (IDWT2 (LL, HL, LH, [0], 'Haar'))

$$\hat{X}=\begin{bmatrix}224 & 224 & 205 & 205 & 168 & 148 & 107 & 167\\ 204 & 204 & 205 & 205 & 128 & 108 & 147 & 207\\ 215 & 195 & 203 & 163 & 123 & 143 & 190 & 190\\ 215 & 195 & 183 & 143 & 123 & 143 & 210 & 210\\ 207 & 187 & 173 & 113 & 125 & 185 & 222 & 182\\ 207 & 187 & 153 & 93 & 145 & 205 & 202 & 162\\ 214 & 174 & 89 & 129 & 207 & 187 & 165 & 165\\ 194 & 154 & 109 & 149 & 207 & 187 & 165 & 165\end{bmatrix}$$

3. Computer Results

Our algorithm applied on (MATLAB Language), by using "Pentium4 - 1.73MHz, 1MB Cache Memory", with Windows XP professional. The "Lena" image two-dimensional array (256 x 256) tested by our algorithm, at first the gray level for "Lena" image reduced by equation (1) Threshold=90. Then apply one stage DWT, and apply SNC on "LL", and the two matrices "HL", and "LH" divided by Quality Factor =20, then applying coding algorithm to produce stream of bits, the original image and decompressed image shown

DOI https://doi.org/2024-JICS-22741 | Generated on 2025-04-09 07:03:22

in Fig.-2.

Peak signal to noise ratio (PSNR) can be calculated very easily and is therefore a very popular quality measure; it is widely used as a method of comparing the "Quality" of original and decompressed images [2,5,8].

(a) Original Image (b) Quality Factor = 20, PSNR = 31.5 dB

Fig – 2 (a) Original "Lena" image, (b) Decompressed image by our algorithm

Our approach compared with JPEG, and JPEG-2000, the JPEG based on DCT and Huffman coding, while JPEG-2000 based on two stages DWT, and Arithmetic Coding [7,11], in Fig.-3 shown results of JPEG, and JPEG2000 and Table 4 shown the comparison with our approach.

(a) (b)

Fig – 3 (a) Decompressed image by JPEG-2000, (b) Decompressed image by JPEG

In Fig -4 Lena's eyes zoom-in three times, to show the detail for our approach compared with JPEG, and JPEG-2000.

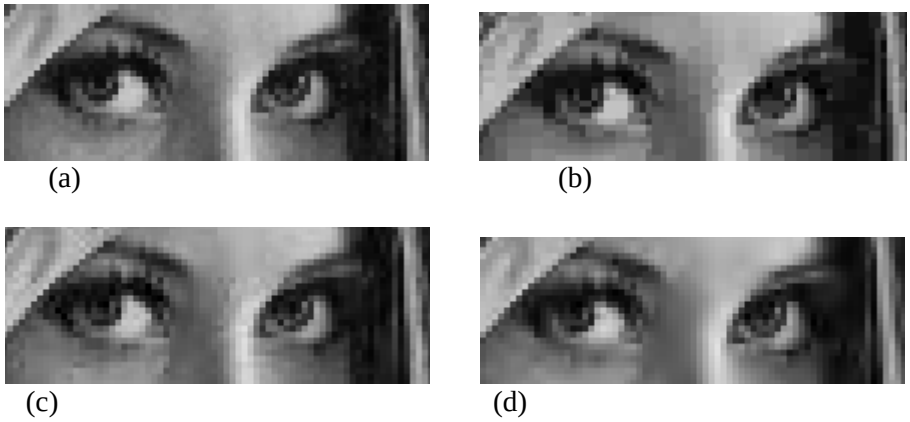


Fig – 4 (a) From Original image, (b) From our approach, (c) From JPEG, (d) From JPEG-2000

Table 4 Comparison with our approach.

Algorithm	Image size Before Compression	Image size After Compression	Compression Performance	PSNR
Our Approach	64 Kbytes	11.1 Kbytes	82.6%	31.5 dB
JPEG	64-Bytes	12.8 Kbytes	80%	33.5 dB
JPEG-2000	64 Kbytes	7 Kbytes	89%	33 dB

4. Conclusion

In this paper introduce an idea for image compression, depend on two parts: (1) SNC, and (2) Discrete Wavelet Transform. The advantage of our approach illustrated in the following steps:-

1- The SNC used for reduce the coefficient of sub-matrix "LL". This operation assist for simplifies the compression process.
2- The DWT separate an image into four sub-images, the two sub-images: "HL", and "LH" has zero's. These zeros could be eliminated, for increasing compression ratio.
3- The "HH" sub-matrix not used in our approach, this leads for increasing compression ratio.
4- The Arithmetic coding play main rule for image compression, it is more efficient than Huffman coding used in JPEG (See Table 4).
5- The RLE used in our algorithm operate on "Bit Matrix", to minimize number of 1's and 0's. While in JPEG technique, the RLE applied on real numbers, this makes our approach give good compression ratio than JPEG.

The disadvantage for our approach illustrated in the following steps:

1- For not using HH matrix, make our approach give results less quality, than JPEG, and JPEG-2000 (See Fig - 4(b)), and Table 4.
2- The Inverse SNC not return the data as original, this makes our approach give less quality than JPEG, and JPEG-2000 (See Fig - 4(b)).
3- Taking more time for compression, and decompression, because our approach produced three arrays; SNC-Array, RLE, and Non-Zero-Array, each array compressed by arithmetic coding independently. The process led to more computations and recurrence calculating and may be led to increase execution time for compression and decompression.

DOI https://doi.org/2024-JICS-22741 | Generated on 2025-04-09 07:03:22

5. REFRENCES

[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, “An em algorithm fo 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] I. Witten, R. Neal and J. Cleary, Arithmetic coding for data compression, *Communications of theACM*, 30 (6), June 1987

[13] W. Yang, F. Wu, Y. Lu, J. Cai, K. N. Ngan, and S. Li, .4-d wavelet-based multiview video coding,. *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 16, no. 11, pp. 1385.1396, Nov. 2006..

DOI https://doi.org/2024-JICS-22741 | Generated on 2025-04-09 07:03:22