An Approach to Improve JPEG for Lossy Still Image Compression

Mohammad Ali Akber Dewan, Rashedul Islam, Mohammad Amir Sharif, and Md. Aminul Islam
Computer Science & Engineering Discipline, Khulna University, Khulna 9208, Bangladesh
Emails: sumon_430@yahoo.com , rashed_406@hotmail.com , amirsharif618@yahoo.com , cseku@khulna.bangla.net

Abstract: *JPEG (Joint Photographic Experts Group) is one of the most popular compression standard in the field of still image compression. The compression ratio of lossless methods (e.g., Huffman, Arithmetic, LZW) is not high enough for image and video compression, especially when the distribution of pixel values is relatively flat. The lossless encoding schemes can be used as the final step for the lossy compression. In JPEG technique an input image is decomposed with the DCT, quantized and end of block coded to give input symbol sequence. Then JPEG uses entropy coding to compress the image data. In this paper, the proposed method performs a modification at this stage. When the whole image is encoded after applying the entropy (JPEG-like Huffman) encoding, the bitstream of the image is created. Then we split the bit stream into 8 bits blocks and the generation of blocks is done until the whole bits are accommodated with the blocks. After completing the block creation of the whole stream, we apply another compression encoding technique on these blocks, so the average bit rate of each block will be reduced and as a result a better compression ratio can be achieved.*

Keywords: *JPEG, Lossy Compression, Entropy Coding, Quantization, DCT.*

1. INTRODUCTION

Image usually contains so much data that they need to be compressed prior to storage or transmission. Without the use of compression, an image of size 1024 pixel x 1024 pixel x 24bit would require 3 MB of storage and 7 minutes for transmission if utilizing a high speed, 64 Kbits/s, ISDN line. If the image is compressed at a 10:1 compression ratio, the storage requirement would be reduced to 300 KB and the transmission time will drop to under 6 seconds [1].

At present there are many international standards to compress and decompress images and one of the most successful families of still image compression standards have been resulted from the ongoing work of the Joint Photographic Expert Group (JPEG). In JPEG the image is first subdivided into pixel blocks of size 8 x 8 and then each block is encoded in mainly three steps: DCT computation, quantization, and variable-length code assignment. In the last step entropy (modified Huffman) coding is used to compress data [2]. These compressed data is nothing but a sequence of bits. In our proposed method we use this bitstream and prepare blocks. Each block contains a fixed amount of bits. If the average bit rate of blocks can be minimized, then a considerable compression performance can be found and this is the basic idea behind our proposed method.

In this proposed method to reduce the average bit rate of blocks, Huffman encoding technique is used. Because Huffman coding creates variable-length codes, each represented by an integer number of bits. Symbols with higher probabilities get shorter codewords [2]. Huffman coding is the best coding scheme possible when codewords are restricted to integer length, and it is not too complicated to implement [3]. It is therefore the popular coding scheme of choice in many applications. In our proposed method Huffman coding is effective, because integer codeword lengths are suitable for the symbol sequence. As Huffman is a lossless encoding scheme, so there is no possibility of any distortion of the images at this stage. In case of decoding, reverse process can be followed. The important feature of this modification is that it keeps the quality of the image as JPEG does but requires less storage space to store and less time for transmission in the network.

2. AN OVERVIEW OF JPEG

The JPEG lossy compression algorithm operates in several successive stages and these are shown in Figure 1.

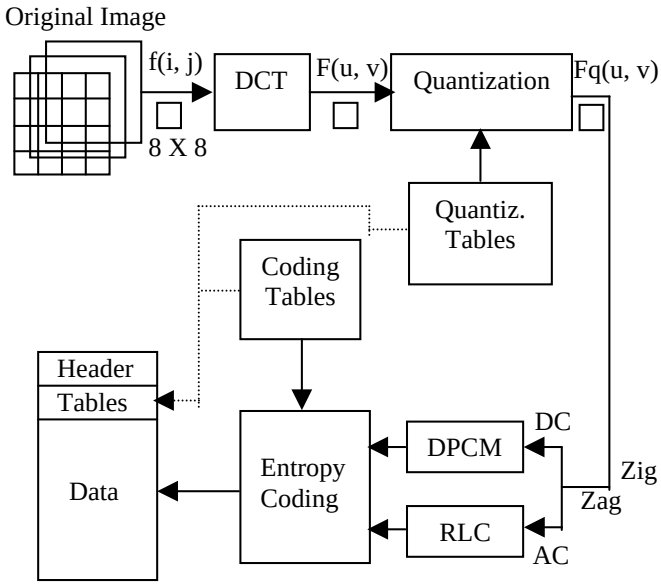

Figure 1. Block diagram of JPEG still image compression technique

Decoding is done by reversing the order of the entire process [3].

The major steps that are used sequentially to compress images are as follows:

2.1 Discrete Cosine Transform (DCT)

The image is first divided into blocks of pixels of size 8 x 8, which are processed left to right and top to bottom. As 8 x 8 block or sub image is encountered, its 64 pixels are level shifted by subtracting the quality 2^{n-1}, where 2^n is the maximum number of the gray levels. The 2-D discrete cosine transform of the block is then computed. Figure 2 and Figure 3 shows the process of dividing the image into sub images and the process of the discrete cosine transformation respectively.

The mathematical functions that are used for Discrete Cosine Transformation (DCT) and Inverse Discrete Cosine Transformation (IDCT) are as follows.

i) Discrete Cosine Transform (DCT)

$$F(u,v) = \frac{?(u)?(v)}{4} \sum_{i=0}^{7}\sum_{j=0}^{7} \cos\frac{(2i+1).up}{16} \cos\frac{(2j+1).vp}{16}.f(i,j)$$

$$?(?) = \begin{cases} 1/\sqrt{2} & \text{for } ? = 0 \\ 1 & \text{otherwise} \end{cases}$$

ii) Inverse Discrete Cosine Transform (IDCT)

$$\hat{f}(i,j) = \frac{1}{4} \sum_{u=0}^{7}\sum_{v=0}^{7} ?(u)?(v)\cos\frac{(2i+1).up}{16} \cos\frac{(2j+1).vp}{16}.F(u,v)$$

$$?(?) = \begin{cases} 1/\sqrt{2} & \text{for } ? = 0 \\ 1 & \text{otherwise} \end{cases}$$

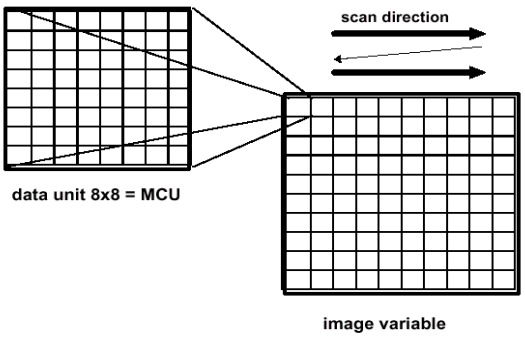

Figure 2. The image is subdivided into pixel blocks of size 8 x 8

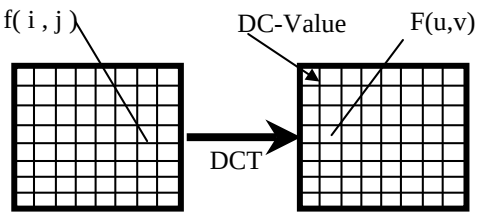

Figure 3. Discrete Cosine Transformation (DCT) of a subimage

2.2 Quantization

DCT is a lossless transformation that does not actually perform compression. The “drastic” action used to reduce the number of bits required for storage of the DCT matrix is referred to as “Quantization”. Quantization [4] error is the main source of the Lossy Compression. Quantization is simply the process of reducing the number of bits needed to store an integer value by reducing the precision of the integer. The JPEG algorithm implements Quantization using a Quantization matrix. For every element position in the DCT matrix gives a Quantum value. The function that is used for quantization is as follows

$$F'[u,v] = \text{round}\,(F[u,v] / q[u,v]).$$

Here q [u, v] is the quantum value, which can be found from the Luminance Quantization Table q (u, v) [5] shown in Table 1. The numbers in the quantization table can be scaled up (or down) to adjust by changing the value of quality factor [5].

Table 1. Luminance Quantization Table

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

2.3 Zigzag Scan

One way to increase the length of runs is to reorder the coefficients in the zigzag sequence. Instead of compressing the coefficient in row-major order, JPEG algorithm moves through the block along diagonal paths, which is here called the zigzag sequence [2]. The Quantized values are reordered using the zigzag pattern and then formed a 1-D sequence of Quantized coefficients [3]. As a result it maps 8 x 8 matrix to a 1 x 64 vector.

Table 2. JPEG Coefficient Coding Categories

Symbol	DPCM Difference	Additional bits
0	0	0
1	-1, 1	1
2	-3, -2, 2, 3	2
3	-7, -4, 4, 7	3
4	-15, -8, 8, 15	4
.	.	.
.	.	.
.	.	.

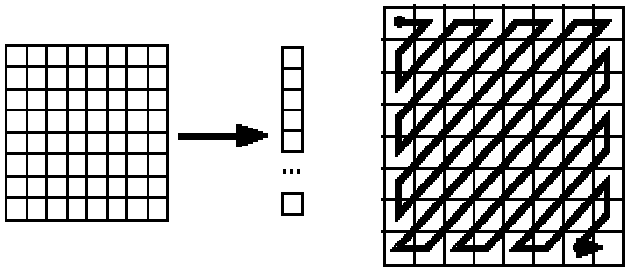

Figure 4. The path of the Zigzag sequence

2.4 Differential Pulse Code Modulation (DPCM) on DC component

The DC component and the AC components are coded separately. The DC component is DPCM coded. DC component is large and varied, but often close to previous value. Calculating the difference of DC values from previous 8 x 8 matrix does the DPCM encoding.

2.5 Run Length Encode (RLE) on AC components

The vector 1 X 64 found from the Zigzag sequence [2] has lots of zeros in it. For the AC component the zeros are run length coded. It keeps *skip* and *value*, where *skip* is the number of zeros and *value* is the next non-zero component. Symbol (0) is used as EOB.

2.6 Entropy Coding / JPEG-like Huffman coding

Entropy coding is also known as JPEG-like Huffman coding. In this step, The DC component and the AC components are coded separately. The DC component is DPCM coded and the symbols are defined in table 2. For the AC component, the zeros are run length coded. Each symbol consists of two parts, the first part is the run that tells how many zeros that precede the value (R), and the second part is the value symbol (S). The value symbols are the same as the symbols used for the DPCM differences. To completely specify the value each symbol is succeeded by additional bits the same way as for the DPCM differences. The combined symbol (represented as one integer) is 16R+S. Symbol (0) is EOB. As a result a completely coded bitstream is produced at this step, which is stored in the storage.

3. PROPOSED MODIFICATION

In this paper, the proposed modification deals with the bitstream that is produced after entropy coding. The bit stream is nothing but a sequential arrangement of bits. The creation of block starts, when the whole image is completely coded by entropy encoding and bitstream is generated. Blocks are generated in fixed length and in this method each block consists of 8 bits. In this way, the block creation will continue until the whole bitstream is encoded as blocks. Now each block will be treated as a symbol. The probability of occurrence of each symbol is different. Now the symbols are arranged as the tree structure shown in Figure 5 according to their probability and bits are assigned to each symbol. Lowest number of bits is assigned to the symbol, which has the highest probability, and in this way the highest number of bits is assigned to the symbol, which has the lowest probability. As a result the average bit rate of the sequence of symbols will be reduced. In the proposed method, assignment of bits to the symbols is done by following the Huffman encoding technique. So the Huffman code tables need to be included in the compressed file as side information. The block diagram of our proposed algorithm is given in Figure 5.

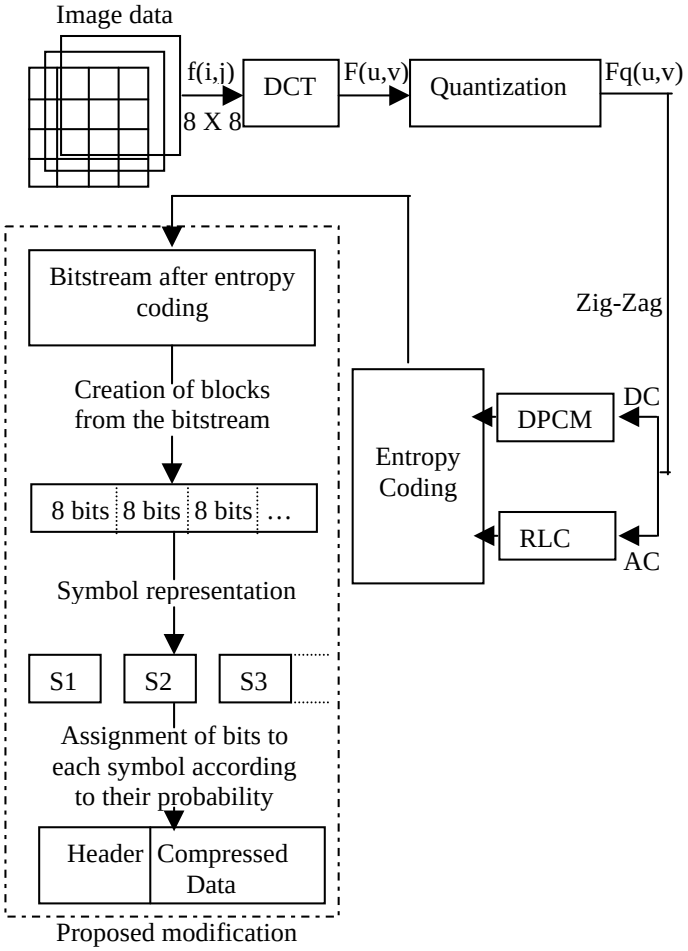

Figure 5. Block diagram of JPEG algorithm including our proposed modification

For example, we consider the following 8 x 8 sub image, whose pixel values are given in Table 3.

Table 3. Input Pixel matrix

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

As the subimage has been taken from such an image, which consists of 256 or 2^8 possible gray levels, so the coding process begins by level shifting the pixels of the original sub images by -2^7 or –128 gray levels. The resulting shifted array is given in Table 4.

Table 4. Level shifted matrix

-76	-73	-67	-62	-58	-67	-64	-55
-66	-69	-62	-38	19	-43	-59	-56
-66	-69	-60	- 15	16	-24	-62	-55
-65	-70	-57	- 6	26	-22	-58	-59
-61	-67	-60	24	-2	-40	-60	-58
-49	-63	-68	-58	-51	-65	-70	-53
-43	-57	-64	-69	-73	-67	-63	-45
-41	-49	-59	60	-63	-52	-50	-34

After transforming the values of Table 4 in accordance with the forward DCT, the transformed matrix becomes a DCT matrix and it is shown in Table 5.

Table 5. Output DCT matrix

-415	-29	-62	25	55	-20	-1	3
7	-21	-6	29	11	-7	-6	6
-46	8	77	-25	-30	10	7	-5
-50	13	35	-15	-9	6	0	3
11	-8	-13	-2	-1	1	-4	1
-10	1	3	-3	-1	0	2	-1
-4	-1	2	-1	2	-3	1	-2
-1	-1	-1	-2	-1	-1	0	-1

If the Quantam value is used from the Luminance Quantization Table (Table 1) to quantize the transformed array, the transformed truncated coefficients are given in Table 6.

Table 6. Quantized Matrix

-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-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

The transformation and normalization process produces a large number of zero-valued co-efficient. When the coefficients are reordered in accordance with the zigzag ordering, the resulting 1-D coefficient sequence becomes as follows

[-26 -3 1 -3 -2 -6 2 -4 1 -4
1 1 5 0 2 0 0 -1 2 0 0
0 0 0 -1 -1 EOB]

Then after DPCM on DC component, RLE on AC components and after entropy coding, the completely coded bitstream becomes as follows

1010110 0100 001 0100 0101 100001 0110
100011 001 100011 001 001 100101 11100110
110110 0110 11110100 000 1010

Here the spaces have been inserted solely for readability. Practically these are not required. Thus the number of bits required to represent the entire 8 x 8, 8-bit/pixel subimage is 92. Now by taking this 92 bits, preparing of 8 bits block starts and as the last block contains 1010, so four 0 will be added to the last four bits positions to complete the last block. This impacts the image quality, a little bit which is negligible. Then all the blocks are treated as symbols and depending on the probability of their occurrence, the symbols are arranged as a tree structure shown in Figure 6.

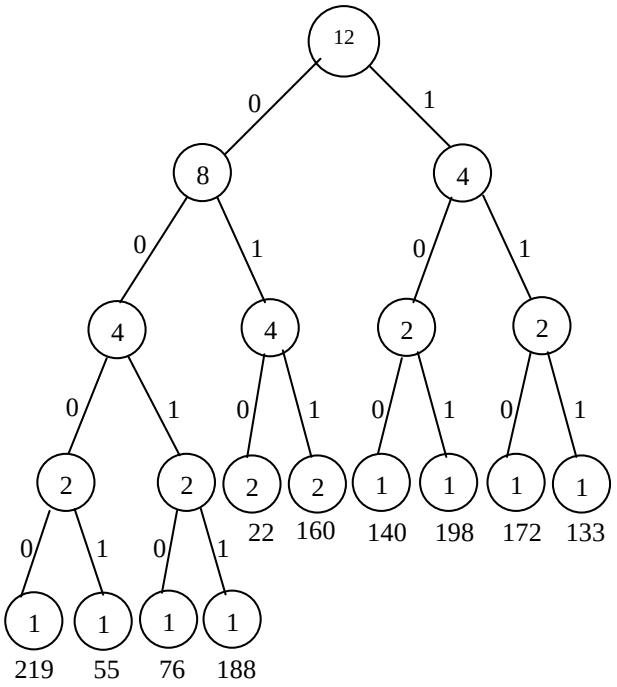

Figure 6. Tree structure to represent bits to symbols

Table 7. Blocks of bit stream

Block of bits	ASCII value	Newly assigned bits
10101100	172	110
10000101	133	111
00010110	22	010
00010110	22	010
10001100	140	100
11000110	198	101
01001100	76	0010
10111100	188	0011
11011011	219	0000
00110111	55	0001
10100000	160	011
10100000	160	011

The tree shown in Figure 6 is constructed by following the Huffman encoding mechanism. Table 7 shows the blocks structure, ASCII values of symbol that are created from blocks and the new bits representation to each symbol by using the tree shown in figure 5. After encoding all the blocks in accordance to the proposed method, the number of bits required to represent all the symbols is 40. i.e. a considerable compression ratio is found.

In case of decompression, the reverse process needs to apply, i.e. at first applying the decoding technique of Huffman on the compressed data, a bit stream will be found. Then by following the reverse process of JPEG algorithm, step by step, the pixel values of the image can be found.

This observation is done only on a 8 x 8 sub image of the picture. When this method is applied on the whole image the compression ratio may differ than that of individual blocks, which is shown in the example.

4. SIMULATIONS AND COMPARISON

Experiment is done on five images. Table 8 shows the experimental results. It shows the original image size, compressed image size by using JPEG and the compressed image size by using our proposed modification. The experiment is performed for the quality factor of 1, 2 and 3. In the above experiment it is found that the proposed method contains a better compression ratio with compare to the JPEG technique for the same quality factor. As a result better compression is found from the proposed modification.

Table 8. Original and Compressed file size

Number	Original File Size / kb	Compressed File Size / kb		
	BMP	JPEG	Proposed Method	Quality factor
1. Scene	133.18	48.30	47.77	1
2. Village	127.49	51.52	50.98	
3. Fishing	163.07	49.81	49.29	
4. Nest	153.30	50.37	49.84	
5. Forest	129.67	51.24	50.73	
1. Scene	133.18	39.60	38.62	2
2. Village	127.49	43.06	42.22	
3. Fishing	163.07	41.28	40.35	
4. Nest	153.30	41.91	40.97	
5. Forest	129.67	42.77	41.98	
1. Scene	133.18	33.94	32.63	3
2. Village	127.49	37.61	36.38	
3. Fishing	163.07	35.72	34.43	
4. Nest	153.30	36.44	35.11	
5. Forest	129.67	37.36	36.17	

5. CONCLUSION

For the last few years JPEG committee has been working towards the establishment of a new standard known as JPEG 2000(i.e. ISO/IEC 15444)[6]. The fruits of these labors are now coming to bear, as JPEG-2000 part 1(i.e. ISO/IEC 15444-1) has recently been approved as a new international standard [6]. From a functionality point of view JPEG 2000 is a true improvement, providing lossy and loss-less compression, progressive and parse able bit streams, error resilience, random access, region of interest and other features in one integrated algorithm. But practically it does not provide any truly substantial improvement in compression efficiency and are significantly more complex [7]. In the cases where lossy compression is of interest and low complexity is of high priority, JPEG still provides a good solution [7]. So in our proposed method we have chosen JPEG as our compression technique and made some modification in order to improve the compression efficiency. The above result given in Table 8 shows that the proposed modification has achieved a significant improvement in terms of compression efficiency. As in the proposed method, no additional loss occurs in the image quality, so it keeps the image quality as JPEG does. Thus it is clear that the proposed method shows better performance than JPEG in terms of storage space and image quality.

REFERENCES

[1] Ho Weng Leong, Umi Kalthum Ngah, P.A.Venkatachalam, Noor Eliza Abd.Khalid, “Image Compression Software Analysis and Design”, *Proc. ROVPIA 1999.* pp. 385-392, Malaysia, 16-18 July 1999.

[2] Mark Nelson, Jean-Loup Gailly, *The Data Compression Book,* M&T Books, New York, USA 1996.

[3] William B. Pennebaker, Joan L. Mitchell, “JPEG: Still Image Data Compression Standard “. Van Nostrand Reinhold, New York, USA, 1992.

[4] Allen Gersho and Robert M. Gray, *Vector Quantization and Signal Compression,* Kluwer Academic Publishers, Boston, 1992.

[5] Rafael C.Gonzalez, Richard E. Woods, *Digital Image Processing*, Addison Wesley Longman.

[6] Diego Santa-Cruz and Touradj Ebrahimi, “A Study of JPEG 2000 still image coding versus other Standards”, LTS/DE/EPFL, CH-1015 Lausanne, Switzerland.

[7] Michel D. Adams, “The JPEG-2000 Still Image Compression Standard”, Vancouverver, BC, Canada.