Journal of Theoretical and Applied Information Technology
10th April 2013. Vol. 50 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

THE FAST ANALYTICAL RESEARCH ON JPEG BITSTREAMS

[1]JING GUO, [2] SHENGBING CHE, [2] XIAOLI LI

[1]Postgraduate, Department of Computer & Information Engineering, Central South University of Forestry and Technology, Changsha 410004 China

[2] Prof., Department of Computer & Information Engineering, Central South University of Forestry and Technology, Changsha 410004 China

[2] Postgraduate, Department of Computer & Information Engineering, Central South University of Forestry and Technology, Changsha 410004 China

E-mail: [1]675011625@qq.com, [2]cheshengbing727@163.com, [2]anssa88@gmail.com

ABSTRACT

In this paper, JPEG bitstreams has been analysized through its rapid analytical algorithm. Its optimized algorithm which was based on optimized huffman table has been studied. The experimental research shows that optimized bitstreams can save 2.4% - 27.8% size of file. In order to obtain the specific decoding information of each MCU block, the fast positioning method has been proposed. Calculated with some correlation coefficients and the sampling parameters of image, the fast positioning method can get the position information of each MCU block. It can obtain MCU Block decoding information and present each of the MCU chrominance information of image respectively. Through the grid function showed on picture, more specific analysis of each MCU Block decoding information can be presented when the MCU block number has been given.

Keywords: *JPEG Bitstreams, Optimized Huffman Table, MCU Block Decoding Information, Fast Positioning Method*

1. INTRODUCTION

JPEG file format, namely JFIF, it allows image compression quality and file compression size striking an average, so that it can produce high compression ratio at the same time without losing too much information. However, when using software to process JPEG images, which has belongs to secondary compressed files, usually abate picture quality greatly and also cause color distortion to the picture. So it is very important and necessary to obtain quantitative information, important documents parameters in order to study on image optimization [1].

A JPEG image consists of a sequence of segments, each beginning with a marker, each of which begins with a 0xFF byte followed by a byte indicating what kind of marker it is. Some markers consist of just those two bytes; others are followed by two bytes indicating the length of marker-specific payload data that follows. (The length includes the two bytes for the length, but not the two bytes for the marker.) Some markers are followed by entropy-coded data; the length of such a marker does not include the entropy-coded data. Note that consecutive 0xFF bytes are used as fill bytes for padding purposes, although this fill byte padding should only ever take place for markers immediately following entropy-coded scan data [1]-[2].

Within the entropy-coded data, after any 0xFF byte, a 0x00 byte is inserted by the encoder before the next byte, so that there does not appear to be a marker where none is intended, preventing framing errors. Decoders must skip this 0x00 byte. This technique, called byte stuffing, is only applied to the entropy-coded data, not to marker payload data. Note however that entropy-coded data has a few markers of its own; specifically the Reset markers (0xD0 through 0xD7), which are used to isolate independent chunks of entropy-coded data to allow parallel decoding, and encoders are free to insert these Reset markers at regular intervals (although not all encoders do this) [1]-[6]. Based on the above analysis, its standard and algorithm are always used in steganography method [3] , image transmission in WSNS [4]. Considering the wide spread of JPEG standard, the fast analytical research on JPEG bitstreams becomes has been proposed.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

In this paper, JPEG bitstreams has been rapidly analysized according to JPEG JIFIF markers parsing. Its optimized algorithm is based on optimized huffman table has been studied [7]-[9]. The research shows that optimized bitstreams format can save 2.4% - 27.8% of the file size more than the original one. At the same time, this paper put forward to the rapid positioning method according to the JPEG decoding algorithm. The sampling coefficient parameters and pictures from the calculation of the correlation coefficient make it possible. Through the these pathways of grid function and inputting MCU block number function, we can see the piece of MCU more clearly and know the decoding process steps more deeply as well.

2. THE FAST ANALYTICAL RESEARCH ON JPEG BITSTREAMS

The fast analytical research on JPEG bitstreams is mainly based on the optimized Huffman table [5]-[9]. The optimization bitstreams analysis can carry the example of the blue_01.jpg image. As is shown in figure 1, figure 1(a) shows the original JPEG picture image, while figure1(b) is to show the optimized the JPEG bitstreams image.

A. Original Image(130×100pixels)

B. Optimized Bitstreams Image(130×100pixels)

Figure 1. Blue_01.Jpg Original Image And Its Optimization Bitstreams Image

2.1. Standard Huffman Tables

The following tables are provided in the JPEG standard/specification as examples of "typical" huffman tables. The advantage of using these tables is that no compute-intensive second-pass analysis would then be required prior to encoding into the JPEG file format.

You will note that the standard huffman tables provide lookups for all possible code word bytes. In the DC tables, the first nybble is always 0, and the size field can only be from 0 to 11 (0x0 to 0xB). Therefore, code words will run from x00 to x0B. Code word x00 is a special value that basically indicates the end of block. For the DC entry this simply means that there is no change in value from the previous block (MCU). So, the standard huffman tables for luminance and chrominance both include all 12 possible DC code words [1]-[2].

(1) Standard DC Luminance Huffman Table

Table 1. Standard Dc Luminance Huffman Table

Code Length	Code Words
2	00
3	01 02 03 04 05
4	06
5	07
6	08
7	09
8	0A
9	0B

Total number of code words in table: 12 code words.

(2) Standard DC Chrominance Huffman Table

Table 2. Standard Dc Chrominance Huffman Table

Code Length	Code Words
2	00 01 02
3	03
4	04
5	05
6	06
7	07
8	08
9	09
10	0A
11	0B

Total number of code words in table: 12 code words.

For the AC tables, the first nybble encodes the run-length of zeros that precede the non-zero quantized DCT coefficient. This can be from 0 to 15 (0x0 to 0xF). The second nybble in the code word indicate the size in bits of the non-zero coefficient that followed the run of zeros. This can be from 1 to 10 (0x1 to 0xA). Together, this would give 16 x 10 = 160 possible code words. However, there are two additional code words that are used in describing the AC scan entries: 0x00 and 0xF0. x00

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

represents an End of Block (EOB), which indicates that there are no more non-zero AC coefficients in this component, and that the decoder/encoder will move on to the next component. 0xF0 represents a Zero Run Length (ZRL) which indicates that there was a run of > 15 zeros. This code word represents a run of 15 zeros, and will be followed by another code word that indicates another ZRL or a normal run + size code word. So, there are in total 162 possible AC code words. The standard huffman AC tables include all 162.

(3) Standard AC Luminance Huffman Table

Table 3. Standard Ac Luminance Huffman Table

Code Length	Code Words
2	01 02
3	03
4	00 04 11
5	05 12 21
6	31 41
7	06 13 51 61
8	07 22 71
9	14 32 81 91 A1
10	08 23 42 B1 C1
11	15 52 D1 F0
12	24 33 62 72
15	82
16	09 0A 16 17 18 19 1A 25 26 27 28 29 2A 34 35 36 37 38 39 3A 43 44 45 46 47 48 49 4A 53 54 55 56 57 58 59 5A 63 64 65 66 67 68 69 6A 73 74 75 76 77 78 79 7A 83 84 85 86 87 88 89 8A 92 93 94 95 96 97 98 99 9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5 B6 B7 B8 B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3 D4 D5 D6 D7 D8 D9 DA E1 E2 E3 E4 E5 E6 E7 E8 E9 EA F1 F2 F3 F4 F5 F6 F7 F8 F9 FA

Total number of code words in table: 12 code words.

(4) Standard AC Chrominance Huffman Table

Table 4. Standard Ac Chrominance Huffman Table

Code Length	Code Words
2	00 01
3	02
4	03 11
5	04 05 21 31
6	06 12 41 51
7	07 61 71
8	13 22 32 81
9	08 14 42 91 A1 B1 C1
10	09 23 33 52 F0
11	15 62 72 D1
12	0A 16 24 34
14	E1
15	25 F1
16	00 01 02 03 11 04 05 21 31 06 12 41 51 07 61 71 13 22 32 81 08 14 42 91 A1-B1 C1 09 23 33 52 F0 15 62 72 D1 0A 16 24 34 E1-25 F1 17 18 19 1A 26 27 28 29 2A 35 36 37 38 39-3A 43 44 45 46 47 48 49 4A 53 54 55 56 57 58 59-5A 63 64 65 66 67 68 69 6A 73 74 75 76 77 78 79-7A 82 83 84 85 86 87 88 89 8A 92 93 94 95 96 97-98 99 9A A2 A3 A4 A5 A6 A7 A8 A9 AA B2 B3 B4 B5-B6 B7 B8 B9 BA C2 C3 C4 C5 C6 C7 C8 C9 CA D2 D3-D4 D5 D6 D7 D8 D9 DA E2 E3 E4 E5 E6 E7 E8 E9 EA F2 F3 F4 F5 F6 F7 F8 F9 FA

Total number of code words in table: 12 code words.

2.2. The fast JPEG bitstreams analytical research based on the optimized Huffman table

In contrast to the above tables that are provided in the ITU-T standard, the following is an example of the huffman tables when optimization is enabled for an image from the figure 1(b).

When comparing this to the non-optimized tables, we can find that fewer code words are available. Not all combinations of runs and AC values are used.

A few observations might be noticed:

(1) For the DC Chrominance table: there are no 0A or 0B code words. This indicates that the image did not have any two adjacent MCUs (block) with extreme 10 or 11-bit swings in average color value.

(2) For the DC Luminance table, the standard table places the 00 code word (marking the End of Block) with a 2-bit value. While this may be optimum for encoding images with wide areas of constant luminance. Instead, the EOB code is assigned to a 4 bit string.

(3) Not all the code words are possibly included in each table. This is because the figure 1(b) didn't need to use these run+size combinations and so they could be eliminated. By eliminating

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

these codes, other code words could occupy shorter variable-length codes and lead to decreased file size.

So the fewer entries in the table, on average the fewer consumed entries bits will be encoded. Therefore, the end result is a more efficient representation of the code words in the final JPEG file (meaning a smaller file size). This method will preserve the original image content and quantization tables, allowing us to isolate the effects of huffman table optimization.

The optimized Huffman tables are shown as follows:

(1) Optimized example of DC Luminance Huffman Table of figure 1(b)

Table 5. Optimized Example Of DC Luminance Huffman Table

Code Length	Code Words
2	03 04 05
4	01 02 06
5	07
6	00

Total number of code words in table: 8 code words. Compared with the 12 number of code words in the standard DC Luminance Huffman Table, it saves 4 number of code words.

(2) Optimized example of DC Chrominance Huffman Table of figure 1(b)

Table 6. Optimized Example Of DC Chrominance Huffman Table

Code Length	Code Words
2	02 03 04
3	01
4	00
5	05
6	06

Total number of code words in table: 7 code words. Compared with the 12 number of code words in the standard DC Chrominance Huffman Table, it saves 5 number of code words.

(3) Optimized example of AC Luminance Huffman Table of figure 1(b).

Table 7. Optimized Example Of AC Luminance Huffman Table

Code Length	Code Words
2	01
3	11 02 03
4	00 04
5	21 31 12 05
6	41 51 61 13
7	22 32 14
8	71 81 A1 15 06
9	91 B1 C1 D1 42 62
10	E1 F1 52 23 16
11	F0 82 33 53 25
12	55

Total number of code words in table: 39 code words.

The total code byte of this table are 39 bytes. Compared with the 162 number of code words in the standard AC Luminance Huffman Table, it saves 123 number of code words.

(4) Optimized example of AC Chrominance Huffman Table of figure 1(b)

Table 8. Optimized Example Of AC Chrominance Huffman Table

Code Length	Code Words
2	00 01
3	11 02
4	03
5	21 31 12 04
6	41 13
7	51 22
8	61
9	81 A1 32
10	71 23 14 24
11	C1 D1 05

Total number of code words in table: 24 code words. Compared with the 162 number of code words in the standard AC Chrominance Huffman Table, it saves number of code words. Results show that: figure1(b) image can save 270 bytes more than figure1(a) image and the optimized code flow can save 17.8% of the original code flow.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2.3. Analytical experimental results of saving length of the optimization pictures

JPEG bitstreams analytical research based on optimization huffman coding has been studied on the optimized bitstreams of image. This paper has operate its algorithm on the 32 bit system. Through the repeated experiments, optimized code words has been saving 2.4% - 27.8% of the file size more than the original one. Table 6 is the part of the test results.

Table 9. Part Of Experimental Results

Serial number	Original file length	Optimization file length
1	8345 Bytes	6859 Bytes
2	7948 Bytes	6337 Bytes
3	9657 Bytes	6973 Bytes
4	10815 Bytes	8561 Bytes
5	10276 Bytes	8296 Bytes
6	11373 Bytes	8252 Bytes
7	10155 Bytes	9482 Bytes

Table 10. Percentage Of The Saved File Length

Serial number	Percentage of the saved length
1	17.8%
2	20.3%
3	27.8%
4	20.8%
5	19.3%
6	25.1%
7	6.7%

3. FAST POSITIONING BASED ON THE MCU BLOCK QUERY PARSING

At present, in most commonly used image compression algorithm, a significant feature is that the algorithm always have uniform image segmentation first and then come into a series of transformation and coding processing.

After subsampling, each channel must be split into 8×8 blocks. Depending on chrominance subsampling, this yields (Minimum Coded Unit) MCU blocks of size 8×8 (4:4:4 – no subsampling), 16×8 (4:2:2), or most commonly 16×16 (4:2:0). In video compression MCUs are called macroblocks.

If the data for a channel does not represent an integer number of blocks then the encoder must fill the remaining area of the incomplete blocks with some form of dummy data. Filling the edges with a fixed color (for example, black) can create ringing artifacts along the visible part of the border; repeating the edge pixels is a common technique that reduces (but does not necessarily completely eliminate) such artifacts, and more sophisticated border filling techniques can also be applied.

JPEG algorithm is a typical algorithm that have the static images into 8*8 block division and every 8*8 block will be transformed into entropy coding separately. Decoding program first read sample coefficient from JPEG file, when it conclude the size of MCU, it will figure out the whole MCU numbers of image. Decoding program will recycle every MCU block decoding until it check the EOI mark [10]-[11].

3.1. MCU block analysis

This article has grab the information of MCU block through the decoding process and present each chrominance and luminance information of MCU block, which is convenient for us to come to the unified cognition of the decoding system and provides favorable information and help for the recovery process and the optimized work. Through the image shown in the program as well as the grid function and input MCU block number function, we can thoroughly see the coefficients of every MCU block and know the decoding process steps process more deeply.

The steps it intend to perform are:

(1) Huffman decoding of an MCU.

(2) Fetch the DC & AC coefficients for each component.

(3) Calculate the position of an MCU.

(4) Perform DPCM on DC and RLE on AC coefficients.

(5) Perform Huffman encoding per MCU.

3.2. Fast positioning based on the MCU block query parsing

The proposed fast positioning method is based on the calculation of the sampling coefficient parameters of JPEG fast decoding algorithm and correlation coefficient from pictures. The algorithm is mainly divided into two steps:

Journal of Theoretical and Applied Information Technology
10th April 2013. Vol. 50 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Step 1: calculate the physical relative displacement of the MCU block from the picture, namely MCU [Mcu_x, Mcu_y]. Then calculate the MCU block length according to the sampling coefficients.

First of all, To find the MCU block width from the sampling coefficient of the code flow. Name sampling factor of X direction vfactor and sampling factor of Y direction hfactor. So the length of MCU block and the width of MCU block can be shown as follow:

mcuHeight=8×hfactor, mcuWidth=8×vfactor.

Secondly, the total MCU block numbers of picture which are named mcuNum, is differentiated according to the minimum 8x8 unit block. If the length of picture and width of picture can't be exact division by 8, then the algorithm will automatically added length and width when decoding. So its length and width can adjust to a multiple of eight. Therefore, we must get MCU block numbers from length direction and the width direction before we get mcuNum, and then do an adjustment judgment of its length and width, if they cannot be divided by 8, the algorithm will automatically added length and width spreading to a multiple of 8. We call num_x_line that MCU block numbers of X direction and the num_y_line that MCU block numbers of Y direction after adjustment judgment. So we can get num_x_line and num_y_line through the program.

Calculate the total numbers of MCU blocks: mcuNum=num_x_line×num_y_line.

Finally, calculate the physical relative displacement of MCU block from the picture according to the need of inquired input number of MCU block. Name the inquired input number for findMcuBlockNum, X direction physical displacement of MCU block for Mcu_x, Y direction physical displacement of MCU block for Mcu_y.

So it comes to the following formula:

Mcu_x=findMcuBlockNum/num_x_line;

Mcu_y=findMcuBlockNum%num_y_line.

Step 2: calculate the MCU block coordinate from the picture. Specific MCU block coordinate calculation has been shown in figure 2 below:

Figure 2 is a specific coordinate calculation diagram of one MCU block:

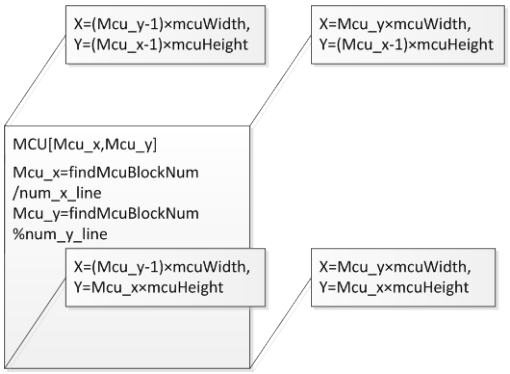

Figure 2. MCU Block Coordinate Setting Schematic Diagram

3.3. The experimental results

This paper tries to operate the optimized JPEG decoding algorithm in 32 bit system. Through the c + + language to develop the corresponding procedures, this paper analyze the decoding information of MCU block and add the grid drawing function, MCU block quick location function. This paper use Microsoft Visual C++ 6.0 for programming environment, use 32 bit Win7 system, 2g memory of hardware, dual-core CPU of 1.73 GHz.

3.3.1. Display of MCU grid function

JPEG algorithm is a typical algorithm that have the static images into 8*8 block division and every 8*8 block will be transformed into entropy coding separately. The image is split into blocks of 8×8 pixels, and for each block, each of the Y, CB, and CR data undergoes the Discrete Cosine Transform (DCT). Depending on chrominance subsampling, this yields (Minimum Coded Unit) MCU blocks of size may be 8×8 (4:4:4 – no subsampling), 16×8 (4:2:2), or most commonly 16×16 (4:2:0). This paper firstly read the sampling coefficients, calculate the total numbers of MCU block, finally display it in the form of grid.

Figure 3 and figure 4 are showing the grid function of JPEG block splitting.

Figure 3 shows the 8×8 MCU block size decoding while figure 4 shows the 16×16 MCU block size decoding.

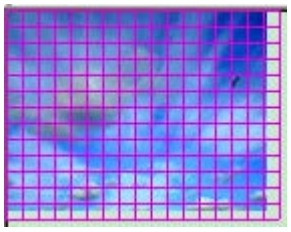

Figure 3. Grid Display Of 8×8 MCU Block Size

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Figure 4. Grid Display Of 16×16 MCU Block Size

3.3.2. Results of fast positioning

Using blue_01.jpg image for example, its basic situation are analyze by program which are shown below:

Table 11. Image Basic Information Sheet Of Blue_01.Jpg

Image：blue_01.jpg	blue_01.jpg
Image scale coefficient：	1.000000
Sampling factor of Y direction：	1
Sampling factor of X direction：	1
Width of MCU block：	8
Length of MCU block：	8
Number of MCU block：	221

According to the table, the fast positioning of the first MCU block are shown as below:

Figure 5. Fast Positioning Display Of The First MCU Block

Using the opening of the case of its first MCU block of blue_01.jpg as an example, the specific component coding coefficient information are shown in the list below:

Table 12. Y Component Coding Coefficient Information

-33	-11	1	0	1	0	0	0
-6	-2	1	-1	0	0	0	0
2	-1	0	0	0	0	0	0
-1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Table 13. Cb Component Coding Coefficient Information

47	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	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

Table 14. Cr Component Coding Coefficient Information

-32	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	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 corresponding experimental results are shown as below:

```
------------------------------------------------------------
The first MCU block coding coefficient information
------------------------------------------------------------
Y component coding coefficient information
(in accordance with the zig-zag scanning sequence arrangement)
------------------------------------------------------------
 -33  -11    1    0    1    0    0    0
  -6   -2    1   -1    0    0    0    0
   2   -1    0    0    0    0    0    0
  -1    1    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0

------------------------------------------------------------
Cb component coding coefficient information
(in accordance with the zig-zag scanning sequence arrangement)
------------------------------------------------------------
  47    1    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    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

------------------------------------------------------------
Cr component coding coefficient information
(in accordance with the zig-zag sequence arrangement)
------------------------------------------------------------
 -32    0    0    0    0    0    0    0
   1    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
   0    0    0    0    0    0    0    0
```

Figure 6. The First MCU Block Coding Coefficient Information

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.3.3. Display of detailed decoding information of MCU block

At the same time, the corresponding detailed decoding can also be presented:

(1) Y component coding coefficient information:

```
Val=[ -33] Coef=[00=DC] Data=[0x E7 AD 21 73
=10b(11100111 10------ -------- --------)]
Val=[ -11] Coef=[01..01] Data=[0x AD 21 73 4A
= 8b(--101101 00------ -------- --------)]
Val=[ -6] Coef=[02..02] Data=[0x 21 73 4A 80
= 6b(--100001 -------- -------- --------)]
Val=[ 2] Coef=[03..03] Data=[0x 73 4A 80 A8 =
5b(01110--- -------- -------- --------)]
Val=[ -2] Coef=[04..04] Data=[0x 4A 80 A8 6B =
5b(-----011 01------ -------- --------)]
Val=[ 1] Coef=[05..05] Data=[0x 80 A8 6B EB
= 3b(--001--- -------- -------- --------)]
Val=[ 1] Coef=[06..06] Data=[0x 80 A8 6B EB
= 4b(-----010 1------- -------- --------)]
Val=[ -1] Coef=[07..07] Data=[0x A8 6B EB D9 =
8b(-1011010 1------- -------- --------)]
Val=[ -1] Coef=[08..08] Data=[0x 6B EB D9 F3
= 3b(-000---- -------- -------- --------)]
Val=[ 1] Coef=[09..09] Data=[0x 6B EB D9 F3 =
4b(----0101 -------- -------- --------)]
Val=[ -1] Coef=[10..10] Data=[0x EB D9 F3 F2 =
4b(0100---- -------- -------- --------)]
Val=[ 1] Coef=[11..11] Data=[0x EB D9 F3 F2
= 3b(----001- -------- -------- --------)]
EOB Data=[0x EB D9 F3 F2
= 4b (-------1 010----- -------- --------)]
```

(2) Cb component coding coefficient information:

```
Val=[ 47] Coef=[00= DC] Data=[0x D9 F3 F2 3C
=12b(---11111 0101111- -------- --------)]
Val=[ 1] Coef=[01..01] Data=[0x F3 F2 3C F8
= 3b(-------0 11------ -------- --------)]
EOB Data=[0x F2 3C F8 C0
= 2b(--00---- -------- -------- --------)]
```

(3) Cr component coding coefficient information:

```
Val=[ -32] Coef=[00= DC] Data=[0x F2 3C F8 C0
=12b(----1111 10011111 -------- --------)]
Val=[ 1] Coef=[01..01] Data=[0x F8 C0 E0 98
= 4b(1001---- -------- -------- --------)]
EOB Data=[0x F8 C0 E0 98
= 2b(----00-- -------- -------- --------)]
```

4. CONCLUSION

The fast analytical research on JPEG bitstreams has an important significance to optimized study of JPEG image. In this paper, JPEG bitstreams analytical research based on optimization huffman coding has been studied on the optimized bitstreams of image. Through the repeated experiments, optimized code words has been saving 2.4% - 27.8% of the file size more than the original one. At the same time, this paper put forward to the rapid positioning method according to the JPEG decoding algorithm. Information of MCU block has been grabbed through the decoding process and each chrominance and luminance information of MCU block have been presented thoroughly, which is convenient for us to come to the unified cognition of the decoding system and provides favorable information and help for the recovery process and the optimized work. Through the image shown in the program as well as the grid function and input MCU block number function, we can thoroughly see the coefficients of every MCU block and know the decoding process steps process more deeply.

5. ACKNOWLEDGMENT

This paper was supported by the Graduate's Scientific Research Foundation of Central South University of Forestry and Technology.

REFRENCES:

[1] Eric Hamilton, "JPEG File Interchange Format", www3.org/Graphics/JPEG/jfif.txt, 2005.

[2] Gregory K. Wallace, et al, "The JPEG Still Picture Compression Standard", IEEE Transactions on Consumer Electronics, Vol. 38, No. 1, 1992, pp. xviii-xxxiv.

[3] YE XUEYI, LU GUOPENG, WANG YUNLU, ZHANG YAN, "A JPEG steganographic method based on syndrome-trellis codes", Journal of Theoretical and Applied Information Technology, Vol. 47, No. 1, January 2013, pp. 194-200.

[4] A.KARTHIKEYAN, T.SHANKAR, V.SRIVIDHYA, SURYALOK.SARKAR, AKANKSHAGUPTE, "Energy efficient distributied image compression using JPEG2000 in wireless sensor networks(WSNS)", Journal of Theoretical and Applied Information Technology, Vol. 47, No. 3, January 2013, pp. 875-883.

[5] Pevny, T, "Estimation of primary quantization matrix in double compressed JPEG images", Security, Forensics, Steganography, and Watermarking of Multimedia Contents X, Binghamton, Vol. 6819, 2008, pp. 247-258.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[6] Sanjit K, Mitra, "Digital signal processing: A computer-based approach", the third Edition, New York : McGraw-Hill, 2006.

[7] Junfeng He, Zhouchen Lin, Lifeng Wang, Xiaoou Tang, "Detecting doctored JPEG images via DCT coefficient analysis", 9th European Conference on Computer Vision, Graz, Austria, Vol. 3953, 2006, pp. 423-435.

[8] Weihai, Li Yuan Yuan, Nenghai Yu, "Passive detection of doctored JPEG image via block artifact grid extraction", Signal Processing, Vol. 89, No. 9, 2009, pp. 1821–1829.

[9] Matthias Stimer, Gerhard Seelmann, "Improved Redundancy Reduction for JPEG files", Picture Coding Symposium, Portugal, 2007.

[10] Goyal, V. K, "Theoretical Foundations of Transform Coding", IEEE SIGNAL PROCESSING MAGAZINE, Vol. 18, No. 5, 2001, pp. 9-21.

[11] Thomas M. Cover, Joy A. Thomas. Elements of information Theory. The Second Edition, New York: John Wiley & Sons, 2006.