Digital Image Processing Laboratory: Achromatic Baseline JPEG encoding Lab

December 2, 1998

1 Introduction

JPEG is an industry standard for digital compression and coding of continuous-tone still images. Since it is a simple transform coder with good performance, it became very popular. It was developed as a general-purpose standard for many continuous-tone image applications, jointly by ISO and CCITT as the JPEG stands for 'Joint Picture Expert Group'. There are 4 kinds of operation modes defined in JPEG standard.

sequential mode: block-by-block encoding in scan order

progressive mode: image is built-up from coarse to clear detail

lossless mode: instead of DCT, it uses predictive coding based on a neighborhood of 3 samples.

hierarchical mode:lower-resolution image is encoded first, upsampled and interpolated to predict the full-resolution image and the prediction error is encoded with one of above 3 operation modes.

In this lab, we will explore the baseline JPEG coder which is the simplest version of DCT-based sequential coder. Moreover, we will handle 8 bit luminance component only for simplicity.

2 Baseline JPEG Encoding

Figure 1 illustrates the main procedures of the DCT-based encoder. The source image is partitioned into 8x8 blocks. Then, each block is transformed through FDCT and quantized. After the final step of entropy coding, we can get the compressed JPEG data.

2.1 8x8 FDCT, IDCT

The source image component's samples are grouped into 8x8 blocks. Each block is transformed by the forward DCT into a set of 64 values. These are called DCT-coefficients and they are further categorized as one DC coefficient and 63 AC coefficients. The 8x8 blocks are partitioned as in Figure 2 and the sample orientation is also shown.

Questions or comments concerning this laboratory should be directed to Prof. Charles A. Bouman, School of Electrical and Computer Engineering, Purdue University, West Lafayette IN 47907; (765) 494-0340; bouman@ecn.purdue.edu

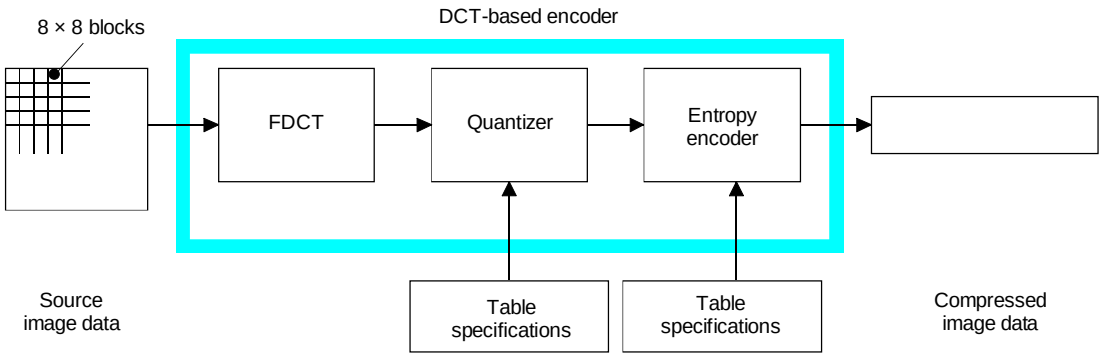

Figure 1: DCT-based JPEG encoder simplified diagram

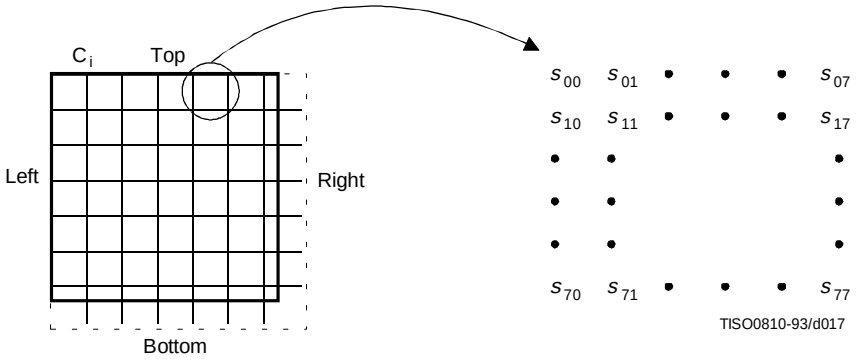

Figure 2: Partition and orientation of 8x8 blocks

FDCT and IDCT are defined as follows:

$$
\begin{aligned}
&FDCT: \\
&\quad S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} cos \frac{(2x+1)\,u\pi}{16} cos \frac{(2y+1)\,v\pi}{16} \\
&IDCT: \\
&\quad s_{yx} = \frac{1}{4} \sum_{x=0}^{7} \sum_{y=0}^{7} C_u C_v S_{vu} cos \frac{(2x+1)\,u\pi}{16} cos \frac{(2y+1)\,v\pi}{16}
\end{aligned}
$$

where ,

$C_u, C_v = \frac{1}{\sqrt{2}}$ for $u, v = 0$

$C_u, C_v = 1$ otherwise.

FDCT function in MATLAB conforms to the above definition.

Prior to FDCT operation, each 8x8 block samples shall be level shifted to a signed representation by subtracting 128. That is, the image sample values $[0, \ldots 255]$ are converted to the values $[-128, \ldots 127]$ in 2's complement form.

A numerical analysis of the 8x8 FDCT shows that, the nonfractional part of the DCT coefficients can grow by at most 8 times(3 bits) of N, if the 64-points contains N bit integer. So, the sample values $[-128, \ldots 127]$ may grow to $[-1024, \ldots 1023]$ 11 bit 2's complement

values.

2.2 Quantization

After the 8x8 FDCT, each of the 64 resulting DCT coefficients is quantized by a uniform quantizer. The quantizer step size is defined as a table of 8x8 block for each 8x8 DCT coefficients. Loss of image information is caused by this quantization with different step size operation. It means that we compress the image with some distortion. The uniform quantizer operates as following equation:

$$Sq_{vu} = round\left(\frac{S_{vu}}{Q_{vu}}\right) \tag{1}$$

where,

S_{vu} = input sample (v,u) in 8x8 block
Q_{vu} = quant step size at location(v,u) in 8x8 quant table
Sq_{vu} = quantized DCT coefficient, rounded after normalizing by Q_{vu}.

Through the average statistics of a large set of video image with 8 bit precision, some typical quatization tables which may prove to be useful for many application, are developed. Also you can use any quantization table of your own, as long as you specify it in the JPEG file parameter field. The quantized DCT coefficient values for 8 bit image are signed integer with 11 bit precision. The reason is 3 bit increase in precision due to FDCT process. One of typical quantization table for luminance component is shown in Figure 3.

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

Figure 3: Typical luminance uniform quantizer table

Now we will experiment the DCT and quantization.

1. Load the gray image *img03y.tif* from the lab home page. The image size is an integer multiple of 8x8 block.

2. Convert it to *double* type matrix. Then subtract 128 from all the element of the array.

3. Execute *jpgqz.m* M-script file. You will get *Quant* and *Zig* 8x8 matrix.

4. Perform 8x8 block FDCT and Quantization using MATLAB's builtin functions *dct2* and *blkproc*.
   Note: Use the command
   f= blkproc(img,[8,8],'round(dct2(x,[8,8])./P1)',Quant);
   where *Quant* is the parameter into **P1** in the inline function **'round(dct2(x,[8,8])./P1)'**.

5. Do the inverse operation in the order of dequantization, IDCT, level restoration, to get restored image.
   Note: As before, use *blkproc* and *idct2* command.

6. Get the difference image obtained by subtracting restored image from source image. The difference image may contain negative values. So you should shift the difference values by adding 128 to make them positive.

Section 2.2 Report:
Hand in the hard copy of the difference image.

2.3 DC coding and Zig-zag scan

Among 64 DCT coefficients, the DC coefficient is treated separately from the other 63 AC coefficients. It corresponds to the local 8x8 block average. Since the DC coefficient has the highest energy, a fine step quantization leads to a large information bits. Too much loss of information will distort the image with blocking effects, and therefore a proper fine quantization step is necessary. Also, there is still high correlation among the DC coefficients of each block, so that it is differentially encoded. What should be coded for DC coefficients, is the difference in the quantized DC coefficient between the current block and the previous block of the same component(in our case, luminance):

$$DIFF = Sq_{00} - PRED \tag{2}$$

PRED value is initially set to zero. The *DIFF* value is encoded as *Variable-Length Integer*.

The remaining 63 AC coefficients have high probability to be zero after quantization, since higher frequency coefficients have lower energy. It is highly probable for a high frequency AC coefficient to be zero, given that its predecessors are zeros. Therefore there will be runs of zeros in the AC coefficients. To exploit the advantage of runs of zeros further, so-called zig-zag scanning is used. Through the scan, we can group longer runs of zeros. The zig-zag scan sequence is shown in Figure 4.

2.4 VLC(Huffman) and VLI coding

DC and AC DCT coefficients in a 8x8 block, are coded differently. The DC difference values are structured as 2 symbols before encoding;

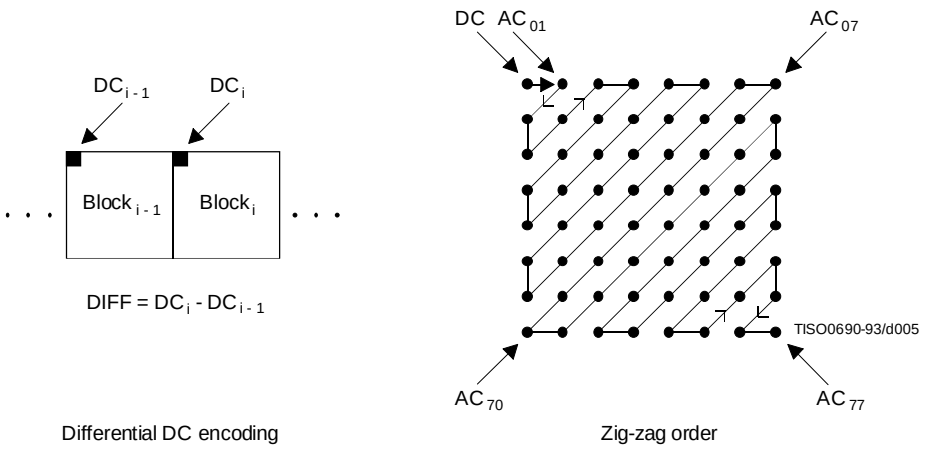

Figure 4: Zig-zag sequence of quantized DCT coefficients

symbol-1	symbol-2
(BIT SIZE)	(DIFF value)

Symbol-1 is encoded as one of Huffman (Variable Length) code words defined in the DC code table, which designates the bit size of the *symbol-2*. Then *symbol-2*(*DIFF*) is appended to LSB of the code word, most significant bit first. Let's say m bits are designated for *DIFF* by *symbol-1*. When *DIFF* is positive, low order m bits of DIFF is appended. When *DIFF* is negative, low order m bits of $DIFF - 1$ is appended. For example, if *DIFF* value is '-3', then it belongs to the set $\{-3, -2, 2, 3\}$ and m is 2. According to above scheme, each *DIFF* value whose $m = 2$ is coded as each of $\{00, 01, 10, 11\}$ 2 bit code. So, the *DIFF* value '-3' is encoded as '01100'. *Symbol-2* coding scheme is called 'Variable Length Integer'. Table 1 shows the *DIFF* magnitude category and the *Symbol-1* Huffman code word.

DIFF values	m	Code word for m
0	0	00
$-1, 1$	1	010
$-3, -2, 2, 3$	2	011
$-7 \ldots -4, 4 \ldots 7$	3	100
$-15 \ldots -8, 8 \ldots 15$	4	101
$-31 \ldots -16, 16 \ldots 31$	5	110
$-63 \ldots -32, 32 \ldots 63$	6	1110
$-127 \ldots -64, 64 \ldots 127$	7	11110
$-255 \ldots -128, 128 \ldots 255$	8	111110
$-511 \ldots -256, 256 \ldots 511$	9	1111110
$-1023 \ldots -512, 512 \ldots 1023$	10	11111110
$-2047 \ldots -1024, 1024 \ldots 2047$	11	111111110

Table 1: Luminance DC coefficient DIFF value and code word

Before encoding, the AC coefficient values are represented by a pair of symbols:

symbol-1	symbol-2
(RUN LENGTH, BIT SIZE)	(AC MAGNITUDE)

Symbol-1 represents 2 pieces of information. RUN LENGTH is the number of consecutive zero-valued AC coefficients preceding the nonzero AC coefficient in the zig-zag scaned sequence. BIT SIZE is the number of bits used to encode the VLI representing AC MAGNITUDE of the nonzero AC coefficient. The relationship between BIT SIZE and AC MAGNITUDE is the same as in the DC VLI's case. Table 2 shows the relationship.

Bit Size	AC coefficient value range
0	0
1	$-1, 1$
2	$-3, -2, 2, 3$
3	$-7 \ldots -4, 4 \ldots 7$
4	$-15 \ldots -8, 8 \ldots 15$
5	$-31 \ldots -16, 16 \ldots 31$
6	$-63 \ldots -32, 32 \ldots 63$
7	$-127 \ldots -64, 64 \ldots 127$
8	$-255 \ldots -128, 128 \ldots 255$
9	$-511 \ldots -256, 256 \ldots 511$
10	$-1023 \ldots -512, 512 \ldots 1023$

Table 2: AC coefficient magnitude category for bit size

However, BIT SIZE information, once combined with RUN LENGTH information, is mapped to different Huffman code from DC Huffman code. RUN LENGTH field can have values 0 to 15. Actual zero-runs in the zig-zag scanned sequence can be greater that 15, so *symbol-1* pair (15,0) meaning *ZRL* is interpreted as the extension symbol with 16 zero-runs. There can be up to 3 consequtive (15,0) extension and 1 terminating *symbol-1*. If the only nonzero AC coefficient occurs, for example, at the 51^{th} position in the zig-zag scan, then it is expressed as *symbol-1* pair string: '(15,0), (15,0), (15,0),(2,x)' where x is one of $[1, \ldots A]$ representing BIT SIZE information. Also, in that case, the last run of zeros starting from the 52^{th} position, includes the 63^{th} AC coefficient. This special zero-run is denoted as *symbol-1* pair (0,0) meaning *EOB(End of Block)*, and can be interpreted as an 'escape' symbol saying no more non-zero value in the 8x8 zig-zag scaned sequence. Actual Huffman code assignment for the *symbol-1* pair(i,j) is shown at the Appendix B.

Zero run	Bit Size of AC magnitude	
	0	1 2 . . . 9 A
0	EOB	Composite values
.	N/A	
.	N/A	
.	N/A	
F	ZRL	

Figure 5: Two dimensional value array for AC Huffman coding

1. Repeat the exersize upto step 4 in Section 2.2 .

2. Extract two quantized 8x8 block corresponding to the image location (1,1) and (1,9). Then, convert each 8x8 block into two 1x64 matrix named *A,B* according to the zig-zag scan order using *Zig*.
   Note: All matix or array index conform to MATLAB convention.

3. Assume DC_pred = 0. By referncing section 2.4, write down the corresponding BIT SIZE code and VLI value for the DC prediction at *A* matrix.

4. For AC coefficients, until you find the nonzero AC coefficient, count the consequtive zeros and determine the magnitude BIT SIZE of the coefficient. Form ('count','range') pair and convert the pair to VLC code as shown in Table at Appendix B. Write down the code value and concatenate it to the above DC codes. Then, write down the VLI value for the AC coefficient. Continue the AC encoding to the end of *A* matrix.

5. Also, encode *B* matrix for DC and AC coefficients and concatenate the result to previous one.

6. Convert the whole bitwise code for *A,B* matrix into byte unit. If you get a byte value '0xFF', then insert byte value '0' next to 0xFF. This procedure is called *byte stuffing*, to prevent '0xFF' from being accidentally taken as a part of markers in Appendix A.

Section 2.4 Report:
Hand in the coded byte string for *A, B* matrix.

3 Baseline JPEG Compressed data format

To insure proper representaion of the compressed data, it shall maintain an ordered format which consists of parameters, markers, entropy-coded data. All of them are represented as byte-aligned value and for each byte, MSB comes first and LSB comes last. (ref: [1] Annex B) Figure 6 shows the order of the high-level consistuent parts for DCT-based encoding process. Among them, only necessary parts for achromatic baseline JPEG encoding will be briefed in the Appendix A.

1. Build your own MATLAB routine performing DC and AC coding. Make it work on 1x64 zig-zag scanned array. All the Huffman tables are given in the *huff.m* M-script fie. The routine may output code values as string or byte value. **Note:** You can use **find** MATLAB command to count zero-run for AC coefficients.

2. Compare the routine's output for *A, B* matrix mentioned at the exercise of Section 2.4 to your previously written-down code value by hand.

3. With the provided JPEG header management M-function files *put_header.m* and *put_tail.m*, extend your routine to work as baseline JPEG encoder. It is easier to have your routine only work on achromatic image whose screen size is multiple of 8 in both horizontal and vertical direction. The arguments and return value of the subfunctions are as follows:

   function fid = put_header(v,h,qtable,fid) ;
   function fid = put_tail(fid) ;

   - **v** – vertical dimension of source image in pixel unit
   - **h** – horizontal dimension of source image
   - **qtable** – 8x8 quantization matrix used in encoding process
   - **fid** – output file pointer designated by user.

4. Output the *img03y.tiff* into *img03y.jpg*, as a result of your JPEG encoder routine. Check the validity of output using *xv* program.

5. Double the 'Quant' matrix value and run your baseline JPEG encoder routine. Then Output the *img03y.tiff* into *img03yq2.jpg*. Compare the quality and size of *img03yq2.jpg* to those of *img03y.jpg*.

Section 3 Report:

1. Hand in your JPEG encoder source code.
2. Hand in hard copy of *img03y.jpg* and *img03yq2.jpg*.

References

[1] ISO/IEC 10918-1,1993(E)

[2] G.K.Wallace. The JPEG still picture compression standared. *Communications of the ACM*, Vol. 34, No.4:30-44,April 1991.

A Basic JPEG Header Formats

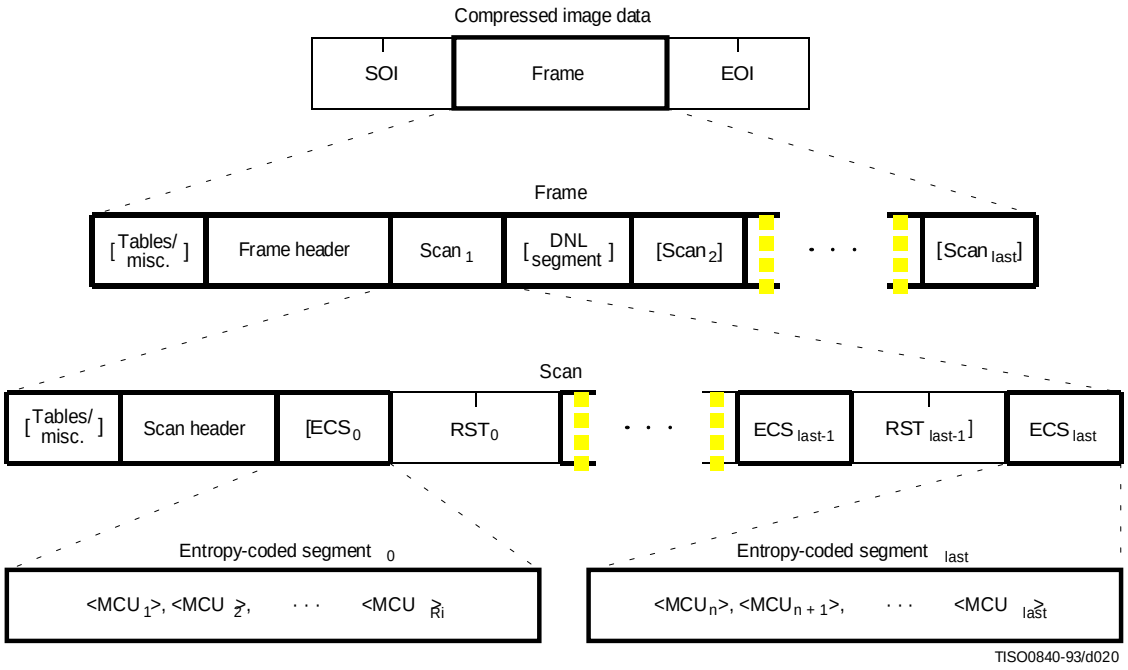

Figure 6: Syntax for sequential DCT-based operation

A.1 Frame

Frame is located between SOI(0xFFD8) and EOI(0xFFD9) mark. It contains some tables and headers and scan segments. In baseline encoder, we will use only one scan segment and one luminance quantization table and two Huffman table for each of DC and AC encoding as in sec A.2.
The header specifies the source image characteristics, and encoded components specific parameters. Baseline DCT JPEG is designated by frame start marker SOF_0(0xFFC0).

Lf: (16 bit) Frame header length in bytes.

P: (8 bit) Bits/Sample precision.

Y: (16 bit) Number of lines in the source image.

X: (16 bit) Number of samples in one line.

Nf: (8 bit) Number of image component in the frame.

C_1**:** (8 bit) Component identifier label.

H_1**:** (4 bit) Horizontal sampling factor.

V_1**:** (4 bit) Vertical sampling factor.

Tq_1**:** (8 bit) Quantization table destination selector.

For example, with 512x768 luminance only source image, the frame header content looks like following in hexadecimal numbers:

```
FF C0 0B 08 03 00 02 00 01 11 00
```

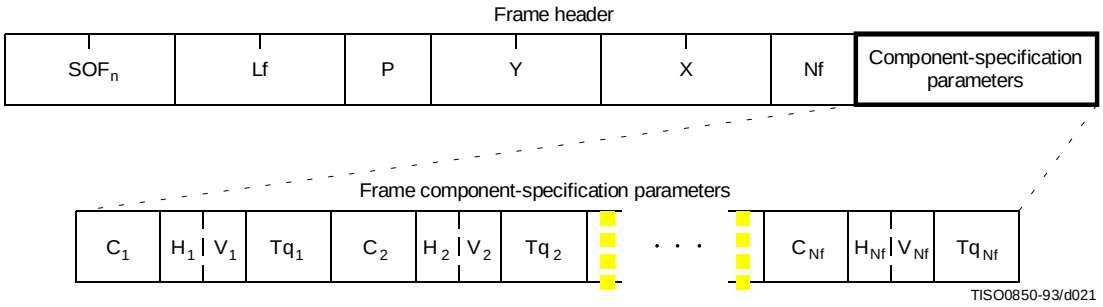

Figure 7: Frame Header syntax

A.2 Quantization and Huffman code Tables

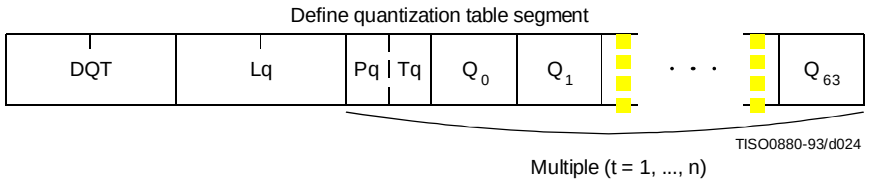

Figure 8: Quantization table syntax

1 Quantization table and 2 Huffman code tables are inserted between SOI(start of image) and SOF_0(start of frame). Each parameter in the DQT(Define Quantization table:0xFFDB) is defined below.

Lq: (16 bit) Quantization table length.

Pq: (4 bit) Table element Qk's precision. $'0' = 8$ bit, $'1' = 16$ bit.

Tq: (4 bit) Quantization table destination identifier.

Qk: (8 bit) Quantization table element in zig-zag scan order.

We will use a typical quantization table in Figure 3 for the luminance component. For baseline DCT JPEG, the quantizer table is denoted as follows:

```
FF DB 00 43 00
10 0B 0C 0E 0C 0A 10 0E 0D 0E 12 11 10 13 18 28
1A 18 16 16 18 31 23 25 1D 28 3A 33 3D 3C 39 33
38 37 40 48 5C 4E 40 44 57 45 37 38 50 6D 51 57
5F 62 67 68 67 3E 4D 71 79 70 64 78 5C 65 67 63
```

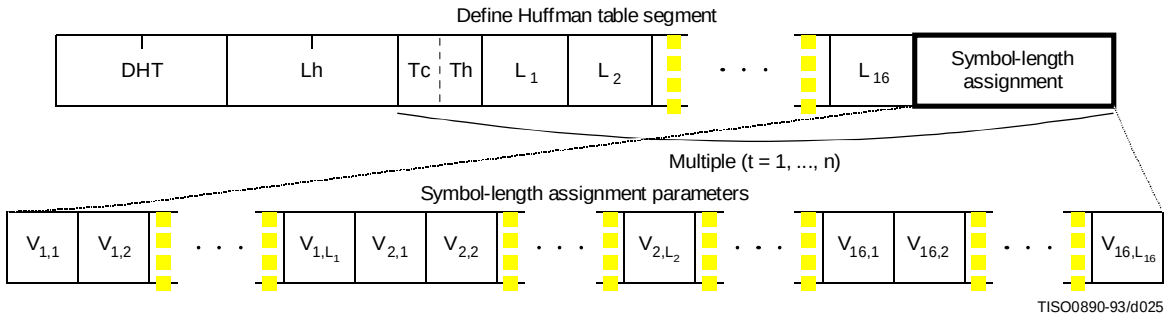

Figure 9: Huffman table syntax

The Huffman tables are located before SOF_0 marker. Its syntax is shown in Figure 9. Like Qantization table definition, it starts with DHT(Define Huffman Table:0xFFC4) marker. Following parameters are defined:

Lh: (16 bit) Huffman table definition length

Tc: (4 bit) Table class, 0 = DC table, 1 = AC table

Th: (4 bit) Huffman table destination identifier

L_i**:** (8 bit) Number of Huffman codes of length i. $(1 \leq i \leq 16)$

$V_{i,j}$**:** (8bit) Value associated with each Huffman code.

Table 1 content is converted into DHT segment for DC. DC Huffman table definition example for a typical Huffman code is shown below

```
FF C4 00 1F 00
00 01 05 01 01 01 01 01 01 00 00 00 00 00 00 00
00 01 02 03 04 05 06 07 08 09 0A 0B
```

First row shows DHT to Th field. Second row represents $L_1 \ldots L_{16}$. Since L_3 value is 05, it is interpreted as that there are 5 codes of length 3 bit in the DC Huffman code table, and they are {01 02 03 04 05} in the third row. They corresponds to Table 1. For AC Huffman table definition example, refer citeitu-t81 Annex K.3.3.2. Only difference is that they needs 162 $V_{i,j}$ entries. It represents the Figure 5's content.

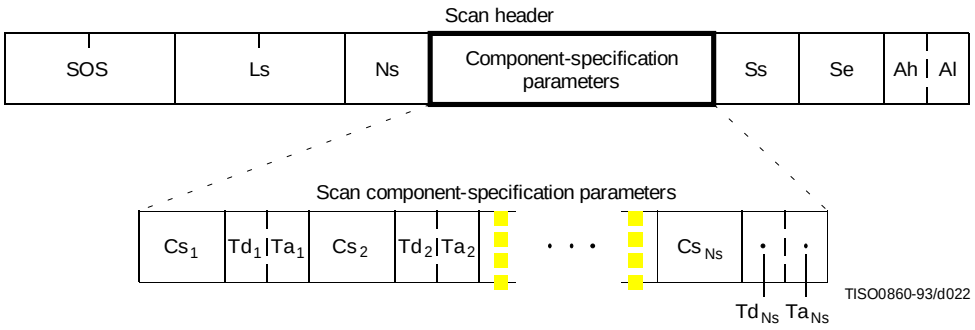

Figure 10: Scan Header Syntax

A.3 Scan Segment

This segment is the last parameter segment before the actual encoded data appear. It specifies which component was coded and which Huffman tables were used. Following parameters are to be defined.

Ls: (16 bit) Scan header length.

Ns: (8 bit) Number of image components in this scan segment.

Cs_j**:** (8 bit) scan component selector.

Td_j**:** (4 bit) DC Huffman table destination selector.

Ta_j**:** (4 bit) AC Huffman table destination selector.

Ss**:** (8 bit) Start of spectral selection. Specify the first DCT coefficient in zig-zag order,to be coded.

Se: (8 bit) End of spectral selection. Sepcify the last DCT coefficient in zig-zag order, to be coded.

Ah: (4 bit) Set to zero in Sequential DCT.

Al: (4 bit) Set to zero in Sequential DCT.

For achromatic the baseline DCT JPEG, scan header values and the encoded data stream upto EOI marker example is shown below. 'zz ww yy . . . ' represents the Huffman code for first 8x8 DCT block's coefficients.

```
FF DA 00 08        % Start of Scan marker
01 01 00 00 3f 00
zz ww yy . . .     % encoded data stream
. . .
. . .
FF D9              % EOI marker
```

B Typical AC Huffman Table

Run/Size	Code length	Code word
0/0 (EOB)	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	1111000
0/7	8	11111000
0/8	10	1111110110
0/9	16	1111111110000010
0/A	16	1111111110000011
1/1	4	1100
1/2	5	11011
1/3	7	1111001
1/4	9	111110110
1/5	11	11111110110
1/6	16	1111111110000100
1/7	16	1111111110000101
1/8	16	1111111110000110
1/9	16	1111111110000111
1/A	16	1111111110001000
2/1	5	11100
2/2	8	11111001
2/3	10	1111110111
2/4	12	111111110100
2/5	16	1111111110001001
2/6	16	1111111110001010
2/7	16	1111111110001011
2/8	16	1111111110001100
2/9	16	1111111110001101
2/A	16	1111111110001110
3/1	6	111010
3/2	9	111110111
3/3	12	111111110101
3/4	16	1111111110001111
3/5	16	1111111110010000
3/6	16	1111111110010001
3/7	16	1111111110010010
3/8	16	1111111110010011
3/9	16	1111111110010100
3/A	16	1111111110010101

Table 3: Typical Huffman table for AC coefficients(sheet 1 of 4)

Run/Size	Code length	Code word
4/1	6	111011
4/2	10	1111111000
4/3	16	1111111110010110
4/4	16	1111111110010111
4/5	16	1111111110011000
4/6	16	1111111110011001
4/7	16	1111111110011010
4/8	16	1111111110011011
4/9	16	1111111110011100
4/A	16	1111111110011101
5/1	7	1111010
5/2	11	11111110111
5/3	16	1111111110011110
5/4	16	1111111110011111
5/5	16	1111111110100000
5/6	16	1111111110100001
5/7	16	1111111110100010
5/8	16	1111111110100011
5/9	16	1111111110100100
5/A	16	1111111110100101
6/1	7	1111011
6/2	12	111111110110
6/3	16	1111111110100110
6/4	16	1111111110100111
6/5	16	1111111110101000
6/6	16	1111111110101001
6/7	16	1111111110101010
6/8	16	1111111110101011
6/9	16	1111111110101100
6/A	16	1111111110101101
7/1	8	11111010
7/2	12	111111110111
7/3	16	1111111110101110
7/4	16	1111111110101111
7/5	16	1111111110110000
7/6	16	1111111110110001
7/7	16	1111111110110010
7/8	16	1111111110110011
7/9	16	1111111110110100
7/A	16	1111111110110101

Table 4: Typical Huffman table for AC coefficients(sheet 2 of 4)

Run/Size	Code length	Code word
8/1	9	111111000
8/2	15	111111111000000
8/3	16	1111111110110110
8/4	16	1111111110110111
8/5	16	1111111110111000
8/6	16	1111111110111001
8/7	16	1111111110111010
8/8	16	1111111110111011
8/9	16	1111111110111100
8/A	16	1111111110111101
9/1	9	111111001
9/2	16	1111111110111110
9/3	16	1111111110111111
9/4	16	1111111111000000
9/5	16	1111111111000001
9/6	16	1111111111000010
9/7	16	1111111111000011
9/8	16	1111111111000100
9/9	16	1111111111000101
9/A	16	1111111111000110
A/1	9	111111010
A/2	16	1111111111000111
A/3	16	1111111111001000
A/4	16	1111111111001001
A/5	16	1111111111001010
A/6	16	1111111111001011
A/7	16	1111111111001100
A/8	16	1111111111001101
A/9	16	1111111111001110
A/A	16	1111111111001111
B/1	10	1111111001
B/2	16	1111111111010000
B/3	16	1111111111010001
B/4	16	1111111111010010
B/5	16	1111111111010011
B/6	16	1111111111010100
B/7	16	1111111111010101
B/8	16	1111111111010110
B/9	16	1111111111010111
B/A	16	1111111111011000

Table 5: Typical Huffman table for AC coefficients(sheet 3 of 4)

Run/Size	Code length	Code word
C/1	10	1111111010
C/2	16	1111111111011001
C/3	16	1111111111011010
C/4	16	1111111111011011
C/5	16	1111111111011100
C/6	16	1111111111011101
C/7	16	1111111111011110
C/8	10	1111111111011111
C/9	16	1111111111100000
C/A	16	1111111111100001
D/1	11	11111111000
D/2	16	1111111111100010
D/3	16	1111111111100011
D/4	16	1111111111100100
D/5	16	1111111111100101
D/6	16	1111111111100110
D/7	16	1111111111100111
D/8	16	1111111111101000
D/9	16	1111111111101001
D/A	16	1111111111101010
E/1	16	1111111111101011
E/2	16	1111111111101100
E/3	16	1111111111101101
E/4	16	1111111111101110
E/5	16	1111111111101111
E/6	16	1111111111110000
E/7	16	1111111111110001
E/8	16	1111111111110010
E/9	16	1111111111110011
E/A	16	1111111111110100
F/0 (ZRL)	11	11111111001
F/1	16	1111111111110101
F/2	16	1111111111110110
F/3	16	1111111111110111
F/4	16	1111111111111000
F/5	16	1111111111111001
F/6	16	1111111111111010
F/7	16	1111111111111011
F/8	16	1111111111111100
F/9	16	1111111111111101
F/A	16	1111111111111110

Table 6: Typical Huffman table for AC coefficients(sheet 4 of 4)