Chapter Nine

Image Compression Standards the JPEG standards

algorithms for compression of still images, both monochrome and color. JPEG is a collaborative enterprise between ISO and CCITT. The standard proposed by the committee was published in 1991.

JPEG Compression Process

The Major Steps in JPEG Coding involve:

• Divide the image into blocks

• Color Space Transform from RGB to YCbCr or YUV and subsampling

• perform DCT (Discrete Cosine Transformation) on image blocks

• apply quantization

• Zigzag scan the quantized value

• DPCM on DC component and RLE on AC Components

• Entropy Coding — Huffman or Arithmetic

1. **Dividing the image into blocks**

An image is a large array of pixels, and because of the sheer number of pixels, not all of them are processed at the same time. First, the image is decomposed into 8X8 blocks of pixels. Then these blocks are submitted to the JPEG system to be compressed one by one.

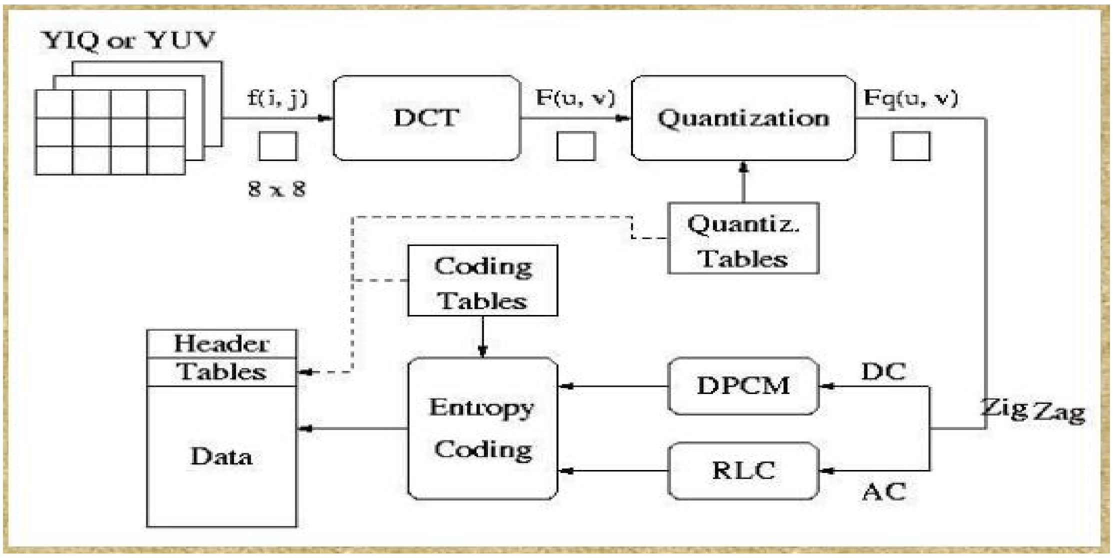

Fig JPEG compression

2. **Color Transform and subsampling**

The JPEG algorithm is capable of encoding images that use any type of color space. JPEG itself encodes each component in a color model separately, and it is completely independent of any color-space model, such as RGB, HSI, or CMY. However, the best compression ratios result if a luminance/chrominance color space, such as YUV or YCbCr, is used.

Most of the visual information to which human eyes are most sensitive is found in the high-frequency, gray-scale, luminance component (Y) of the YCbCr color space. The other two chrominance components, Cb and Cr, contain high-frequency color information to which the human eye is less sensitive. Most of this information can therefore be discarded.

Hence, the chrominance information can be downsampled. Downsampling is indicated as

4:4:4, 4:2:2, 4:1:1, 4:2:0, etc. The numbers indicate how many luminance (Y), and chrominance (Cb, and Cr) information is used in four pixels. For example, 4:2:2 indicates that all the four luminance information are used. However, only 2 Cb and 2 Cr color information is used out of the four pixel values. The remaining two will be discarded in each case. In 4:1:1, only 1 Cb, and 1 Cr color information is used discarding the remaining three.

3. **Discrete Cosine Transform (DCT)**

The DCT transforms the data from the spatial domain to the frequency domain. The spatial domain shows the amplitude of the color as you move through space. The frequency domain shows how quickly the amplitude of the color is changing from one pixel to the next in an image file.

The frequency domain is a better representation for the data because it makes it possible for you to separate out and throw away information that isn't very important to human perception. The human eye is not very sensitive to high frequency changes -specially in photographic images, so the high frequency data can, to some extent, be discarded.

DCT can be computed with the formula:

$$F(u,v) = \frac{1}{4}C(u)C(v) * \left[\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) * \cos\frac{(2x+1)u\pi}{16} * \cos\frac{(2y+1)v\pi}{16}\right]$$

where

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } u, v = 0 \\ 1 & \text{otherwise} \end{cases}$$

4. **Quantization**

JPEG utilizes a quantization table in order to quantize the results of the DCT. Quantization allows us to define which elements should receive fine quantization and which elements should receive coarse quantization. Those elements that receive a finer quantization will be reconstructed close or exactly the same as the original image, while those elements that receive coarse quantization will not be reconstructed as accurately, if at all.

A quantization table is an 8x8 matrix of integers that correspond to the results of the DCT. To quantize the data, one merely divides the result of the DCT by the quantization value and keeps the integer portion of the result. Therefore, the higher the integer in the quantization table, the coarser and more compressed the result becomes. This quantization table is defined by the compression application, not by the JPEG standard. A great deal of work goes into creating "good" quantization tables that achieve both good image quality and good compression. Also, because it is not defined by the standard, this quantization table must be stored with the compressed image in order to decompress it

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

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Fig a)luminance quantization table b)chrominance quantization table

The quantization can be done as follows:

$$F^Q(u,v) = Integer\ Round\ \left(\frac{F(u.v)}{Q(u.v)}\right)$$

5. **Zigzag Scan**

The zigzag sequence order of the quantized coefficients results in long sequences, or runs, of zeros in the array. This is because the high-frequency coefficients are almost always zero and these runs lend themselves conveniently to run-length encoding.

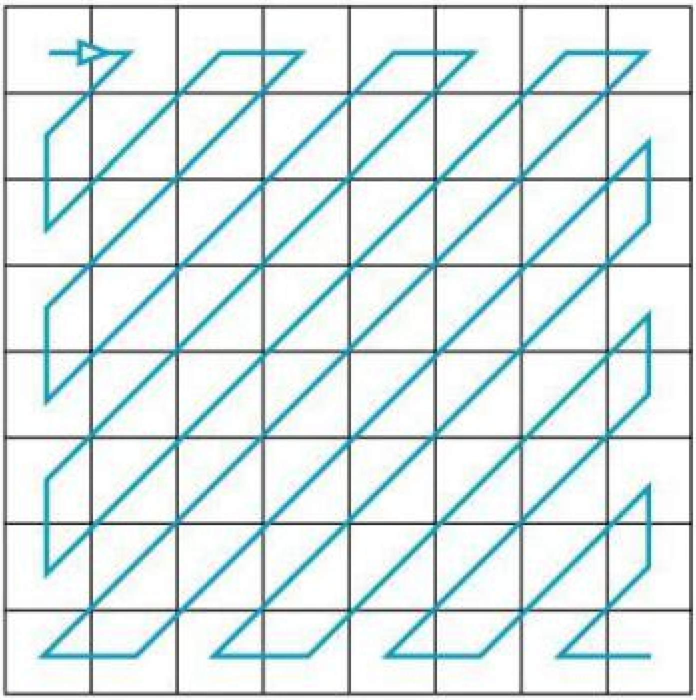

Fig Zigzag scan

6. **Differential Pulse Code Modulation(DPCM) on DC Coefficient**

After quantization, the DC coefficient is treated separately from the 63 AC coefficients. Each 8x8 image block has only one DC coefficient. The DC coefficient is a measure of the average value of the 64 image samples of a block. Because there is usually strong correlation between the DC coefficients of adjacent 8x8 blocks, the quantized DC coefficient is encoded as the difference from the DC value of the previous block in the encoding order.

DC coefficients are unlikely to change drastically within short distance in blocks. This makes DPCM an ideal scheme for coding the DC coefficients.

Example: given the DC coefficients

150, 155, 149, 152, 144

If the predictor is $d_i = DC_{i+1} - DC_i$, the DPCM will be 150, 5, -6, 3, -8

Unlike Run Length Coding of AC coefficients, which is performed on each individual block, DPCM for the DC coefficients is carried out on the entire image at once.

Run Length Encoding(RLE) on AC Coefficients

Zigzag scan produces vector of 64 values. This vector has lots of zeros in it. The run length step replaces values by a pair (runlength, value) for each run of zeros in the AC coefficient. Where, runlength is the number of zeros in the run and value is the next nonzero coefficient. A special pair (0,0) indicates the end-of-blocks after the last nonzero AC coefficient is reached.

Example: given this vector

(32, 6, -1, -1, 0, -1, 0, 0, 0, -1, 0, 0, 1, 0, 0, ..., 0)

It can be coded in RLC as:

(0,6)(0,-1)(0,-1)(1,-1)(3,-1)(2,1)(0,0)

7. **Entropy Coding**

The DC and AC coefficients finally undergo entropy coding. This could be Huffman coding or arithmetic coding. This step achieves additional compression losslessly by encoding the quantized DCT coefficients more compactly based on their statistical characteristics.

The JPEG proposal specifies two entropy coding methods - Huffman coding and arithmetic coding. The baseline sequential codec uses Huffman coding, but codecs with both methods are specified for all modes of operation.

Huffman coding requires that one or more sets of Huffman code tables be specified by the application. The same tables used to compress an image are needed to decompress it.

Huffman tables may be predefined and used within an application as defaults, or computed specifically for a given image in an initial statistics-gathering pass prior to compression. Such choices are the business of the applications which use JPEG; the JPEG proposal specifies no required Huffman tables

Compression Performance

JPEG reduces the file size significantly. Depending on the desired image quality, this could be increased. Some rates are:

• 10:1 to 20:1 compression without visible loss (effective storage requirement drops to 1-2 bits/pixel)

• 30:1 to 50:1 compression with small to moderate visual

• deterioration

• 100:1 compression for low quality applications

JPEG Bitstream

The figure shows the hierarchical view of the organization of the bitstream of JPEG images. Here, a frame is a picture, a scan is a pass through the pixels (e.g. the red component), a segment is a group of blocks, and a block consists of 8x8 pixels.

Some header information are:

⭑ Frame header

- Bits per pixel
- (width, height) of image
- number of components
- Unique ID (for each component)
- horizontal/vertical sampling factors (for each component)
- quantization table to use (for each component)

⭑ Scan header

- number of components in scan
- component ID (for each component)
- Huffman/Arithmetic coding table(for each component)

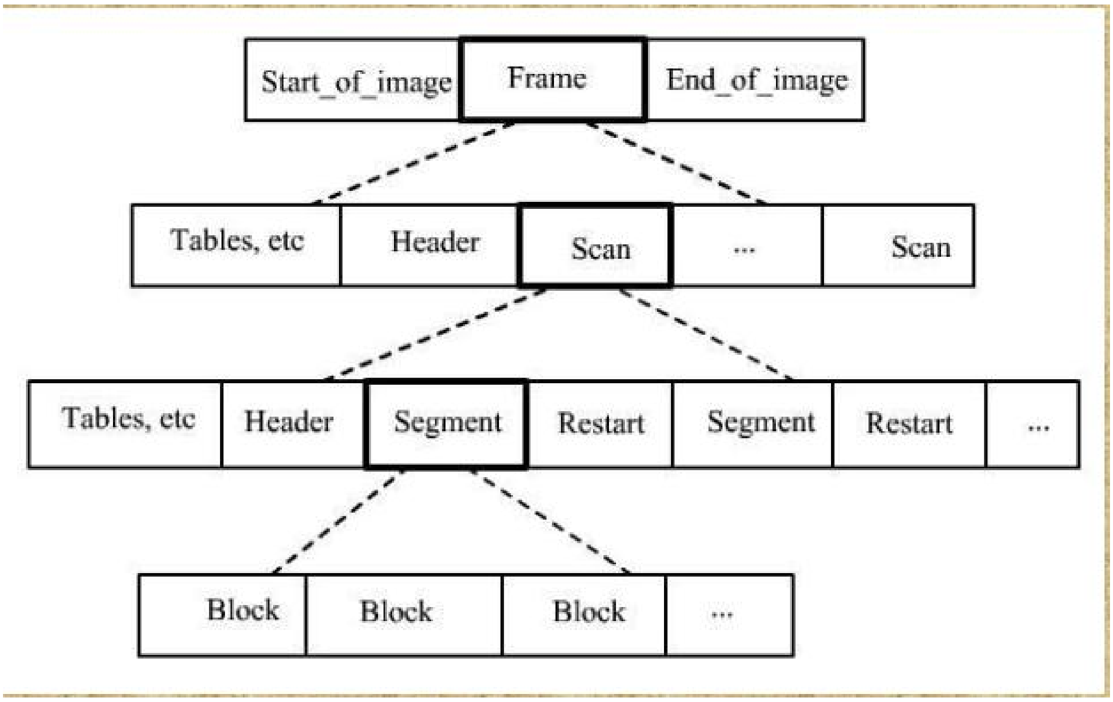

Fig JPEG bitstream

In JPEG bitstream, start_of_image is indicated by a value 0xFF 0xd8, and end_of_image by 0xFF 0xd9.