JPEG Image Compression Systems

The lossy JPEG

JPEG is an acronym for Joint Photographic Experts Group and JFIF is an acronym JPEG File Interchange Format. Normally, the files with the .JPG extension are JFIF files. If a JPEG file is loaded into a text editor such as Window’s NotePad , the string JFIF is started in the first line starting at the 7th character. JPEG compression is mainly discards much of the original information so the exact original image cannot be reconstructed from a JPEG file. This is called lossy compression. While this sounds bad, a photograph actually contains considerable information that the human eye cannot detect so this can be safely discarded.

JPEG Compression Modes

The JPEG standard defined four compression modes: Hierarchical, Progressive, Sequential and lossless. Figure 0 shows the relationship of major JPEG compression modes and encoding processes.

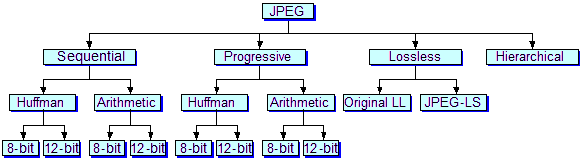


Figure 0. JPEG Operation Modes

1. Sequential: Sequential-mode images are encoded from top to bottom. Sequential mode supports sample data with 8 and 12 bits of precision. In the sequential JPEG, each color component is completely encoded in single scan. Within sequential mode, two alternate entropy encoding processes are defined by the JPEG standard: one uses Huffman encoding; the other uses arithmetic coding.
2. Progressive: In progressive JPEG images, components are encoded in multiple scans. The compressed data for each component is placed in a minimum of 2 and as many as 896 scans. The initial scans create a rough version of the image, while subsequent scans refine it.
3. Lossless: preserves exact, original image, small compression ration, less use
4. Hierarchical: JPEG is a super-progressive mode in which the image Is broken down into a number of subimages called frames. A frame is a collection of one or more scans. In hierarchical mode, the first frame creates a low-resolution version of image. The remaining frames refine the image by increasing the solution.

Baseline JPEG

Baseline JPEG decompressor supports a minimal set of features. It must be able to decompress image using sequential DCT-based mode. For baseline compression the bit depth must be *B*=8; however, it is convenient to describe a more general situation. The image samples are assumed to be unsigned quantities in the range [0, 2^B-1]. The level offset subtract 2^(B-1) from every sample value so as to produce signed quantities in the range [-2^(B-1), 2^(B-1)-1]. The purpose of this is to ensure that all the DCT coefficients will be signed quantities with a similar dynamic range. The image is partitioned into blocks of size 8×8. Each block is then independently transformed using the 8×8 DCT. If the image dimensions are not exact multiples of 8, the blocks on the lower and right hand boundaries may be only partially occupied. These boundary blocks must be padded to the full 8×8 block size and processed in an identical fashion to every other block. The compressor is free to select the value used to pad partial boundary blocks.

Color Space

The VGA card displays colors by setting the intensity of the three colors RED, BLUE and GREEN. The three colors form the axis of a cartesian coordinate system. Any color is then a point in this color space. A grayscale image is formed by only using points in color space where RED, BLUE and GREEN intensities are all equal. The alternate set of axis so called luminance and chrominance. The luminance axis is labelled Y, the blue chrominance axis is labelled U and the red chrominance axis is labelled V. The three new axes we have created form the three components used in JPEG image files. The following formulas will convert between the two coordinate systems.

Y = 0.299 R + 0.587 G + 0.114 B

U = -0.1687 R - 0.3313 G + 0.5 B + 128

V = 0.5 R - 0.4187 G - 0.0813 B + 128

R = Y + 1.402 (V-128)

G = Y - 0.34414(U-128) - 0.71414(V-128)

B = Y + 1.772 (U-128)

There is a reason for using the YUV color space. The human eye is more sensitive to luminance than to chrominance. Typically JPEGs throw out 3/4 of the chrominance information before any other compression takes place. This reduces the amount of information to be stored about the image by 1/2. With all three components fully stored, 4 pixels needs 3 x 4 = 12 component values. If  3/4 of two components are discarded we need 1 x 4 + 2 x 1 = 6 values.  
The largest horizontal and vertical sample factors determine the height and width of the Minimum Coded Unit (MCU) respectively. For the case of figure 4 the MCU would be two 8×8 blocks high and two 8×8 blocks wide for a total of four 8×8 blocks. The blocks are stored in the file all Ys first then all Us then all Vs. For this case one MCU would contain four Y 8×8 blocks followed by one U 8×8 block and one V 8×8

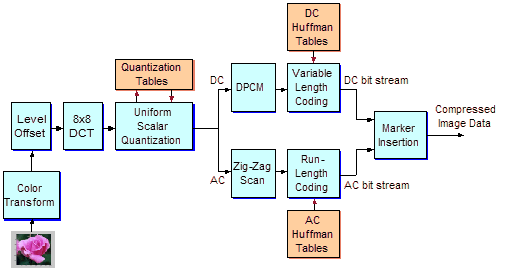


Figure 1. Three-component image with the chrominance subsampled

The basic data unit is an 8×8 block. Each data unit represents the information for a single component of pixel color information. The component could be either luminance(Y), blue chrominance(U) or red chrominance(V).

JPEG Image Compression System

Figure 2 describes the basic parts of a JPEG compression system. The color image (which is represented by three basic color images Red, Green, and Blue) are transformed into the equivalent luminance and chrominance images (Y, U, and V), using the transform formular as shown in the previous section. The level offset is included in the color transformation. The chrominance images U and V then are subsampled by a certain factor to reduce the number of samples for saving bit-rate.

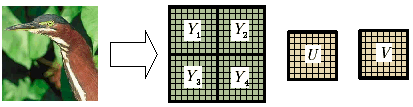


Figure 2. JPEG image compression system

The chrominance and luminance images output are the partitioned into 8×8 blocks. Each 8×8 data block is a subject of discrete cosine transform (DCT). The 8×8 original data block and its equivalent block of DCT coefficients are shown in Figure 3

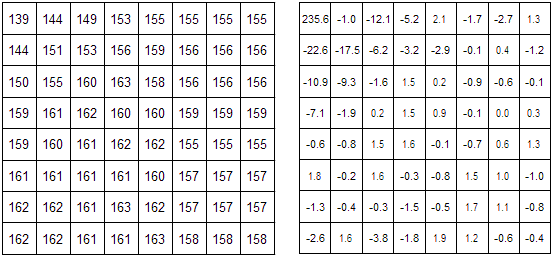


Figure 3. Original data block (left) and Equivalent DCT coefficients (right)

The DCT coefficients are then quantized by divided then for the quantization values defined in the luminance and chrominance quantization tables. The quantization values can be set individually for each DCT coefficient, using criteria based on visibility of the basis functions. The quantization generates error by its nature. Therefore, the fault tolerance systems need to distinguish between the quantization errors and the computer failure errors. This is one of the challenges in the research project.

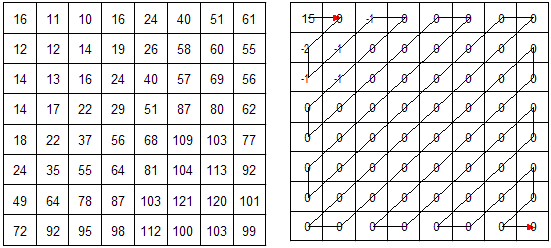


Figure 4. Luminance quantization table (left) and the Quantized DCT coefficients (right)

Figure 4 shows a particular luminance quantization table and the quantized coefficients of the DCT block as shown in Figure 3. Most of the AC coefficients are reduced to zero and leave a very small number of nonzeroes are concentrated at the low spatial frequencies (the neighborhood of the DC coefficient). To improve the compression ratio, the quantized block is rearranged into the zig-zag order, then applied the runlength coding method to convert the sequence into the intermediate symbols. The *Symbol-*1 contains the (RUNLENGTH, SIZE) information, and the *Symbol-*2 contains the (AMPLITUDE) information

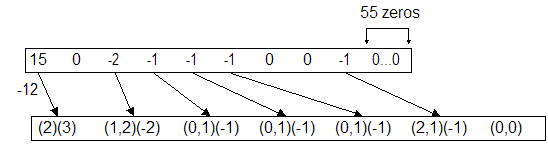
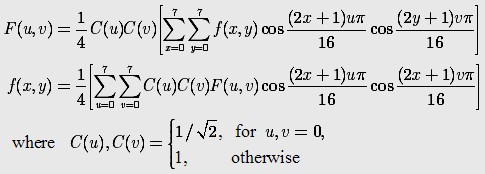


Figure 5. Quantized coefficients in zigzag order (top) and Intermediate symbols from DPCM and runlength codings (bottom)

Figure 5 shows the intermediated symbols that are need to represent for the data block given in Figure 3. The DC coefficient is substituted by the difference between the DC coefficient of the current block and that of the previous block. In the example presents in FigureThe DCT operation is completed by the computer base with the fix point data type. The fast algorithm for the DCT includes the addition, subtraction and shifting operations. It requires a temporary memory space to store 8×8 input data block and intermediate results. All of these are the subject of computer errors. Therefore, fault tolerance design for the DCT stage is the important work in the research project. The quantized DCT coefficients are the subjects of DPCM and runlength coding. The DPCM is used to compute the difference between DC value of the current block and that of the previous block. The runlength coding is used to the rearrange the AC coefficients following the zig-zag path in the 8×8 block.

Discrete Cosine Transform

The ideas behind JPEG compression come from an engineering background. Electrical and sound waves can be represented as a series of amplitudes over time. Discrete cosine transform (DCT) is one of the basic building blocks for JPEG. The discrete cosine transform was first applied to image compression in Ahmed, Natarajan and Rao’s pioneering work, in which they showed that this particular transform was very close to the KLH transform, a transform that produces uncorrelated coefficients. Another important aspect of the DCT is the ability to quantize the DCT coefficients using usually weighted quantization values. To detect a fault in DCT networks, concurrent error detection (CED) design have been proposed.  
In image transformation, a continuous tone image can be represented by a series of amplitudes, for each color component, over 2 dimensional space. Similar to the Fourier transform, a discrete cosine transform is used to discard higher frequency information that has little visual effect on the image. For the still image representation, the frequencies here are referring to spatial frequencies rather than time frequencies.  
The DCT operation in a JPEG image compression system starts with 8×8 image data block, f(x,y). This block can be transformed to a new 8×8 block, F(x,y), by the forward discrete cosine transform (FDCT). The original block f(x,y) can be obtained by the inverse discrete cosine transform (IDCT). The equations for the discrete cosine transforms are:

  
Because the 2-D DCT is separable, the summations can be done as eight 1-D DCTs on all rows, and then eight 1-D DCTs on the eight columns formed by the coefficients for the rows. The fast algorithm of the 8-point DCT can be realized in Figure 6. The arrow path in the graph represents for the minus sign at the node. The node itself represent for an addition or a subtraction.

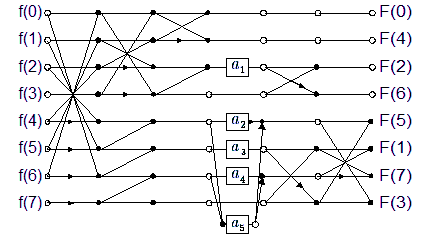


Figure 6. Flowgraph for 8-point DCT adapted from Arai, Agui, and Nakajima.  
a1 = 0:707; a2 = 0:541; a3 = 0:707; a4 = 1:307; and a5 = 0:383

This version of the DCT has 13 multiplications and 29 additions, which is competitive with the best that has been achieved by other techniques. Among 13 multiplications, eight of them are to scale the final output to the correct range. If the output is to be quantized, the output can be left in this scaled form and the scaling factors can be absorbed into the divisions needed to quantized the outputs. Finally, only 5 multiplications are actually needed before the quantization, making the most efficient 1-D quantized DCT known.

Huffman Coding

Huffman codes are widely used and very effective technique for compression data; saving of 20% to 90% are typical, depending on the characteristics of the data being compressed. The data to be considered here is the sequence of characters. Huffman’s greedy algorithm uses a table of the frequencies of occurrence of characters to build up an optimal way of representing each character as a binary string. Given a 100,000-character data file that needs to be stored compactly. Assume the characters in the file occur with frequencies given by Table 1

Table 1. Frequency of characters in a file

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| – | a | b | c | d | e | f |
| Frequency (in thousands) | 45 | 13 | 12 | 16 | 9 | 5 |
| Fixed-length Codeword | 000 | 001 | 010 | 011 | 100 | 101 |
| Variable-length Codeword | 0 | 101 | 100 | 111 | 1101 | 1100 |

The data file contains only the characters a-f, with the frequencies indicated. If each character is assigned as three bit codeword, the file can be encoded in 300,000 bits. Using the variable length code shown, the file can be encoded in 224,000 bits, which saves approximately 25%. In fact, this is an optimal character code for this file. Once alphabet symbols have been defined, the compression efficiency can be improved by using shorter code words for more probable symbols and longer codewords for the less probable symbols. This variable-length codewords belong to entropy coding scheme.  
Huffman coding is one of the entropy coding techniques that JPEG uses in its compression standard. We see here only codes in which no codeword is also a prefix of some other codeword. Such codes are called prefix codes. It is possible to show that the optimal data compression achievable by a character code can always be achieved with a prefix code. Encoding is always simple for any binary character code; The codewords representing each character of the file are just concatenated. For example, with a variable length prefix code of Table 1, the 3 characters abc are coded as 0.101.100=0101100, where “.” is used to denote concatenation. The decoding process needs a convenient representation for the prefix code so that the initial codeword can be easily picked off.

A binary tree, whose leaves are the given characters provides one such presentation. We interpret the binary codeword for a character as the path from the root to that character, where 0 means “go to the left child”, and 1 means “go to the right child”. Figure 7 shows the tree for the fixed length codes of the above example. Each leaf is labelled with a character and its frequency of occurrence. Each internal node is labelled with a sum of the frequencies of the leaves in its subtree. An optimal code for a file is always represented by a full binary tree, in which every none-leaf node has two children. The fixed length code as shown in Figure 7, is not a full binary tree: there are codewords beginning 10…., but none beginning 11….

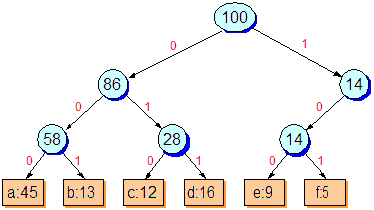


Figure 7. Tree corresponding to fixed-length code scheme  
a=000, b=001, c=010, d=011, e=100, f=101

If the tree is restricted to the full binary tree, then if *C* is the alphabet from which the characters are drawn, then the tree for the optimal prefix code has exactly |*C*| leaves, one for each letter of the alphabet, and exactly |*C*|-1 internal nodes. Figure 8 demonstrates a full binary tree, which realizes the optimal prefix code for the above example.

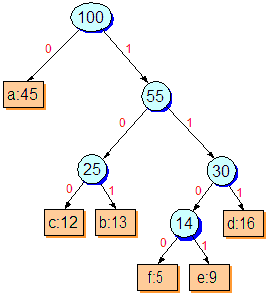
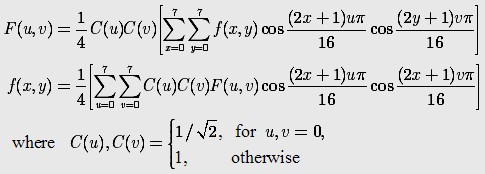


Figure 8. Tree corresponding to the optimal prefix code for the given data file  
a=0, b=101, c=100, d=111, e=1101, f=1100

Given a tree *T* corresponding to a prefix code, it is simple to compute the number of bits required to code a file. For each character *c* in *C*, let *f*(*c*) denote the frequency of *c* in the file and let *d*(*c*) denote the depth of *c*‘s leaf in the tree. *d*(*c*) is also the length of the codeword for character *c* . The number of bits required to encode a file is

The problem with tree structures is that they are very sensitive to memory errors, lacking exploitable redundancy. Trees are constructed by using pointers to link nodes together and each node carries information about itself, its parent and children. Therefore, if one internal node disappears as a result of a memory error, then all its child nodes are lost. Coding by using Huffman code tables is applied in the JPEG image compression standard. The row and column indices indicate the code size as well as the zero runlength of the nonzero DCT coefficients in a block. The using of code table is described more in the fault tolerance design for Huffman coding in JPEG compression systems.

JPEG Compressed Data Format

Figure 8 specifies the order of the high constituent parts of the interchange format, which begins by an SOI marker, a frame, and ends with EOI marker. The structure of frame begins with the frame header and shall contains one or more scans. A frame header may be preceded by one or more table-specification or simultaneous marker segments.

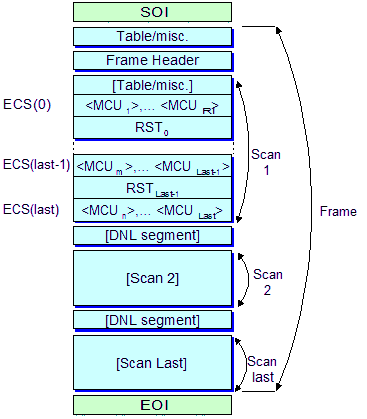


Figure 9. A simple compressed image data format

If a DLN (Define Number of Lines) segment is present, it shall immediately follow the first scan. For DCT-based process, each scan shall contain from one to to four image components. If 2 to 4 components are contained within the scan, they shall be interleaved within the scan. The scan structure begins with a scan header and contains one or more entropy coded data segments. Each scan header may be preceded by one or more table specification or miscellaneous marker segments. If restarted is not enabled, there will be only one entropy coded segment. If restart is enabled, the number of entropy-coded segments is defined by the size of the image and the defined restart interval. In this case the restart marker shall follow each entropy coded segment except the last one. To detect the error for the entire compressed data output, we break it into segments

A JPEG file consists of the eight following parts:

1. A Start of Image SOI
2. An APP0 Marker
   1. APP0 length
   2. Identifier
   3. Version
   4. Units for X & Y densities
   5. X density
   6. Y density
   7. Thumbnail horizontal pixels
   8. Thumbnail vertical pixels
   9. Thumbnail RGB bitmap
3. APPn Markers where n can be form 1 to 15 (Optional)
   1. APPn length
   2. Application specific information
4. One or more quantization tables DQT
   1. Quantization table length
   2. Quantization table number
   3. Quantization table
5. A Start of Frame SOF0
   1. Start of Frame length
   2. Precision (Bits per pixel per color component)
   3. Image height
   4. Image width
   5. Number of color components
   6. For each component
      1. An ID
      2. A vertical sample factor
      3. A horizontal sample factor
      4. A quantization table#
6. One or more huffman tables DHT
   1. Huffman table length
   2. Type, AC or DC
   3. Index
   4. A Bits table
   5. A Value table
7. A Start of Scan SOS
   1. Start of Scan length
   2. Number of color components
   3. For each component
      1. An ID
      2. An AC table #
      3. An DC table #
   4. Compressed image data (Not included in Start of Scan length)
8. An End of Image EOI

JPEG Image Decompression System

The JPEG decompression system is inverse of the JPEG compression system.

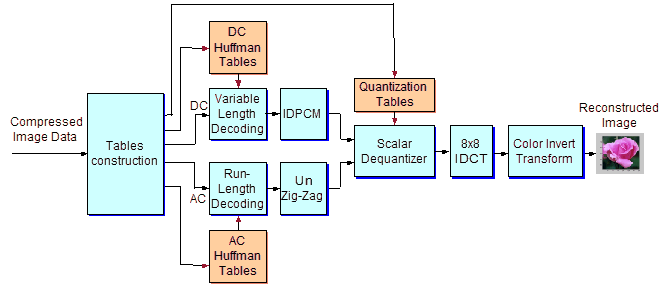


Figure 10. The JPEG decompression structure

As soon the code streams entered the decompression system, the all the received quantization and Huffman tables are reconstructed. The frame headers are also decoded to determine the size and precision of the image. The compressed stream for each 8×8 block is split into two parts. The DC code is decoded using the DC Huffman tables. The value output from DC decoder is, indeed, the difference between the DC value of the current and the previous 8×8 blocks. The IDPCM restores back the true DC value by adding the value obtained from the DC decoder with the DC value decoded from the previous block. The AC part is decoded using the AC Huffman tables to get the AC coefficients, which are organized in zig-zag order. Therefore, the unzigzag stage reorganizes the coefficients into 8×8 block. The dequantization stage performs the multiplications between the coefficients with the The IDCT performs the invert discrete cosine transform for each 8×8 block. Since the quantization generates quantization errors, the reconstructed block data is no longer identical to that of original image. The data obtained at the IDCT output form the chrominance and luminance images, adding with the level offset and finally are converted into the RGB image before displaying on the screen.

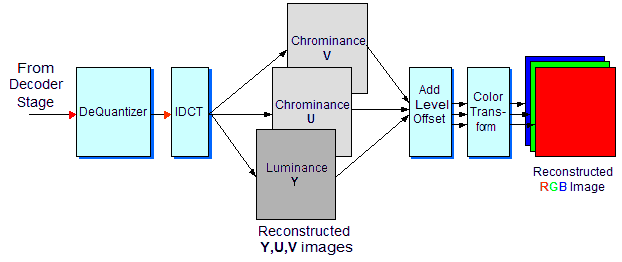


Figure 11. Color invert transformation at the decoder