The JPEG Standard

Rashid Ansari
Department of EECS
University of Illinois at Chicago
Chicago IL, 60607

Nasir Memon
Department of Computer Science
Polytechnic University
Brooklyn, NY 11201

February 8, 1999

1 Introduction

2

JPEG is currently a widely used standard for compression of continuous-tone images. The standard is named after the group of experts that created it and continue to guide its evolution. This group, the Joint Photographic Experts Group (JPEG), consists of experts nominated by national standards bodies and by leading companies engaged in image-related work. The JPEG committee has an official title of ISO/IEC JTC1 SC29 Working Group 1, with a web-site at http://www.jpeg.org. The committee is charged with the responsibility of pooling efforts to pursue promising approaches to compression in order to produce an effective set of standards for still image compression. The lossy JPEG image compression procedure described in this chapter is part of the multi-part set of ISO standards IS 10918-1,2,3 (ITU-T T.81, T.83, T.84).

The JPEG standardization activity commenced in 1986, which generated twelve proposals for consideration by the committee in March 1987. The initial effort produced consensus that the compression should be based on the Discrete Cosine Transform (DCT). Subsequent refinement and enhancement led to the Committee Draft in 1990. Deliberations on the JPEG Draft International Standard (DIS) in 1991 culminated in the International Standard (IS) being approved in 1992.

The key elements of the standard are a baseline algorithm based on the use of DCT and Huffman coding, and extensions that permit the application of arithmetic coding and the use of a progressive and a hierarchical mode. Further extensions continue to be added as the standard evolves.

The main features of the lossy JPEG standard are as follows:

- Compression can be performed in either a lossy or a lossless mode. Lossless compression is a reversible mapping of the raw image into a more compact form, from which the original image can be recovered without loss of information. Lossy compression enatils an irreversible transformation with a controlled loss of information, but which produces a compressed form of the image that is usually much smaller than that produced with lossless compresssion.

- Both sequential and progressive modes of encoding are permitted. These modes refer to the manner in which quantized DCT coefficients are encoded. In sequential coding, the coefficients are encoded block by block basis in a single scan that proceeds from left to right and top to bottom. On the other hand, in progressive encoding only partial information on the coefficients is encoded in the first scan followed by encoding the residual information in successive scans.

- Low complexity implementations in both hardware and software are feasible.
- All types of images, regardless of source, content, resolution, color formats, etc., are permitted.
- A graceful tradeoff in bit rate and quality is offered
- A hierarchical mode with multiple levels of resolution is allowed.
- Bit resolution of 8 to 16 bits is permitted.

Lossy JPEG encoder structure

Image Data Format and Components:

The structure of a JPEG encoding system will now be considered. We examine the system building blocks used in the context of DCT-based compression algorithm. But first we briefly examine the format and components used in representing an image.

An aspect of the data to be compressed that one has to deal with is the number of components that constitute an image representation. In grayscale image there is a single "luminance" component. However a color image is represented with multiple components. JPEG allows a maximum of 255 color components, which consist of rectangular arrays of pixel values with 8-16 bit precision. The corresponding dimensions of the component arrays may be equal or unequal. If the horizontal (vertical) dimensions are unequal, then the larger horizontal (vertical) dimension of one component should be multiple of the smaller horizontal (vertical) dimension of the other component by a factor equal to 2, 3, or 4. The largest dimension supported is $2^{16} = 65,536$.

In storing the component, one has the option of using interleaved or non-interleaved formats. Processing and storage efficiency is aided, however, by interleaving the components. The interleaving is performed by defining a data unit for lossy coding as a single block of 8×8@ pixels. Assume that there are N color components, with the n-th component consisting of array of $H_n \times V_n$ data units.

Example: Consider a raw image consists of three matrices of integers: one luminance matrix (Y), and two chrominance matrices (Cr and Cb). The luminance is of size 512×512 and the Cr and Cb ****

A JPEG compression system is conveniently decomposed into the units shown in Figure 1. Most common techniques can be partitioned into a series of mappings performed by the four units shown in the figure. Note that the compression system shown in Figure 1 is applicable in open-loop/unbuffered environments where the system is not operating under a constraint of a prescribed bit rate/budget.

Signal transformation unit – DCT

In lossy image compression, the input image is usually first applied to a signal transformation unit to map the signal reversibly into a representation that is better suited for compression. The object of the transformation is to reconfigure the information in the signal to capture the redundancies, and present the information in a "machine-friendly" form, which is convenient for disregarding the perceptually least relevant content. In JPEG the transformation used is the Discrete Cosine Transform (DCT) which is applied to 8×8 rectangular blocks of the data. DCT captures the spatial redundancy and packs the signal energy into a few DCT coefficients.

Quantization unit

This block implements a many-to-one mapping of the DCT coefficients, so that the possible outputs are limited in number. This is what makes the encoding "lossy" in that there is a loss of information in the quantization mapping which is not reversible. The quantization of the DCT coefficients allows the information to be compactly represented while permitting reconstruction with

Figure 1: JPEG[3] encoding system

a very gradual loss in quality with increasing quantization. A key feature of the quantized DCT coefficients is that many of them are zero, making them suitable for efficient run-length coding.

Symbol definition unit

The quantized DCT coefficients are mapped to new symbols or events to facilitate a compact representation in the symbol coding unit that follows. The symbol definition unit can also be viewed as part of the symbol coding unit. However, it is shown here as a separate unit to emphasize the fact that the definition of symbols to be coded is an important task. As explained later, an effective definition of symbols in JPEG is the "runs" of zero coefficients followed by a nonzero terminating coefficient.

Symbol coding unit

This unit assigns a codeword to the symbols that appear at its input, and generates the bit-stream that is to be transmitted or stored. A variable-length coder is used, in which the object is to minimize the average length of the codeword, so that the bits/symbol is close to the entropy of the symbol source. Huffman coding is commonly used for variable-length coding in JPEG, with arithmetic coding allowed as an option

2.1 Discrete Cosine Transform coding

Lossy JPEG compression is based on the use of transform coding using discrete cosine transform [**?**] In DCT coding, the image is subdivided into blocks of 8×8 pixels. A two-dimensional DCT is applied to each block of data to obtain an 8×8 array of coefficients. The coefficients are then quantized and entropy coded. The relative precision of the quantized coefficients can be varied by adjusting the quantization intervals, so that transform coefficients judged most important are allocated more bits than those deemed less important. This operation can be implemented with the use of a quantization table that is tailored to preserve the image features that are perceptually most important.

The bulk of the compression achieved in transform coding occurs in the quantization step. The compression ratio is controlled by changing the total number of bits available to encode the 8×8 block of DCT coefficients. As the compression ratio increases, the DCT coefficients are quantized more coarsely. Those coefficients that are deemed less important become subject to a coarser quantization.

The DCT coefficient magnitudes exhibit a pattern in their occurrence in the coefficient array. Also their contribution to the perception of the information is not uniform across the array. These features are exploited in developing methods of quantization and symbol coding that are tailored to DCT. In DCT coding, the coefficients corresponding to the lowest frequency basis functions are usually large in magnitude, and are also deemed most significant for perception. Therefore, bit allocations are biased towards these coefficients. In early use of DCT coding, fixed bit allocation schemes, or adaptive selection from a small number of allocation tables, were used. However, it was found effective to not predetermine the bit allocation. Instead the approach employed in JPEG is to divide the transform coefficients in each block by a "quantization table", and round the result to an integer. This is the irreversible step in which information is lost.

After quantization the DCT coefficients are mapped to a vector using a scan procedure in which it is desired that on the average the coefficient magnitudes appear in decreasing order of magnitude in the vector [**?**]. This would cause most of the quantized coefficients that are zero to appear together in the last entries of vector, making them amenable to efficient run-length coding. A

typical scan is shown in Figure **??**, where the numbers indicate the order in which coefficients are read. This vector contains many zero-valued coefficients that often appear together, making the vector entries suitable for run-length coding. The symbols for entropy coding are typically defined as runs of zeros, terminated by a non-zero coefficient. These procedures are further examined in the discussion of an example later in this chapter. The symbols defined in terms of run-lengths are mapped to a bit-stream by using entropy coding techniques such as Huffman or arithmetic coding.

2.1.1 Quantization

In scalar quantization, each output of the transformation unit in the general compression system is quantized into a finite number of levels. The set of possible input values is typically finite, but this set is conveniently considered to be the set of real numbers $\mathbf{R}$. Let X be the input random variable, with continuous probability density $f_X(x)$. Scalar quantization is a mapping Q defined by a partition of $\mathbf{R}$ into disjoint subsets denoted by R_k, $1 \leq k \leq K$. If the quantizer input value belongs to R_k then it is assigned to, or *quantized* to, a value $\hat{x}_k$, $1 \leq k \leq K$. The mapping Q is defined by

$$Q(x) = \hat{x}_k, \quad x \in R_k \quad , 1 \leq k \leq K$$

An example of the quantizer is shown in Figure 2, where the quantization sets R_k, are given by the intervals (b_{k-1}, b_k), with $b_0 = -\infty$ and $b_K = \infty$.

The most commonly used measure of quantization distortion D_Q is the mean squared error given by

$$D_Q = E[X - Q(X)]^2 = \int_{-\infty}^{\infty} (x - Q(x))^2 f_X(x) dx$$

With the interval boundaries b_k as shown in Figure 2, the distortion D_Q can be expressed as

$$D_Q = \sum_{k=1}^{K} \int_{b_{k-1}}^{b_k} (x - \hat{x}_k)^2 f_X(x) dx.$$

The optimum quantizer, called the Lloyd-Max quantizer, is obtained by minimizing D_Q with respect to the sets of boundary points $\{b_k\}_1^{K-1}$ and the representation (quantized) values $\{\hat{x}_k\}_1^K$.

The solution is given by the following two equations:

$$b_k = \frac{1}{2}(\hat{x}_k + \hat{x}_{k+1})$$

which means that the boundary points are midway between the quantized values, and

$$\hat{x}_k = \frac{\int_{b_{k-1}}^{b_k} x f_X(x) dx}{\int_{b_{k-1}}^{b_k} f_X(x) dx}$$

which means that the quantized value $\hat{x}_k$ is the centroid, or the conditional expected mean, of the associated quantization interval (b_{k-1}, b_k). This set of of equations is iteratively solved to arrive at the optimum quantizer. In the absence of a probability description, one needs training data to which suitable clustering algorithms are applied.

sectionQuantization

The first unit in the general compression system shown in Figure 1 computes the DCT of 8×8 blocks of the image. The output of this unit usually can take on a large number of possible values that in turn require a large number of bits for an accurate representation. The precise values cannot be represented uniquely under a constraint of a limited budget of bits available for the application.

Figure 2: Scalar Quantizer

Therefore the representation of the DCT coefficients calls for some loss in information by mapping a large set of output values into a smaller set allowed under the constraint.

This many-to-one (non-invertible) mapping is called *quantization*. The goal in quantization is to design the mapping to minimize the distortion measure such as mean squared error, while remaining within the allowed coding rate in bits/symbol discussed earlier. The trade-off between the coding rate and the distortion is given by the fundamental limit of a rate-distortion function [**?**], in order to approach which one needs to operate on a large block of input values. The quantization unit can be chosen to operate on a single value or on a block of values applied as input.

3 Symbol coding

The data generated by the quantizer output is mapped into the coded output bit-stream by defining suitable symbols that can be coded efficiently. Since the quantizer output contains a large percentage of zero DCT coefficient values then one can define symbols as consecutive occurrences (runs) of zeros terminated by an element from a finite set of possible non-zero values. Let us denote the symbols so defined by $\{a_i, i = 1, \ldots, M\}$. We wish to represent this symbol sequence in a lossless manner by assigning a unique binary codeword to each symbol. The probability of the symbols is not uniform. In such a case it is advantageous to assign shorter codewords to the more frequently occurring symbols. Based on analysis of representative material, the symbol source S is modeled as one that produces symbols with probabilities $\{P_i, i = 1, \ldots, M\}$. Suppose a codeword of length L_i is assigned to the symbol a_i (that occurs with probability P_i). The average (expected) length of the of the codeword is

$$L_{av} = \sum_{i=1}^{M} P_i L_i$$

From a result in information theory [**?**, **?**], the average length L_{av} is bounded by the entropy of the source S defined by

$$H(S) = \sum_{i=1}^{M} P_i \log_2 \frac{1}{P_i}$$

Clearly the goal of the symbol coding unit is to achieve an average codeword length as close to the entropy as possible. There are systematic procedures of coding that perform very close to the entropy bound. These coding procedures include Huffman coding and arithmetic coding. We will elaborate on Huffman coding and arithmetic coding, which are commonly used in video compression. The Lempel-Ziv algorithm belongs to the class of universal coding methods, where explicit models of source statistics are not required.

3.1 Huffman coding

Huffman coding is based on the knowledge of the probabilities of occurrence of symbols. It leads to minimum average codeword length under the condition that no codeword is a prefix of another. Huffman code is optimal (achieves entropy bound) in the case where all symbols probabilities are integral powers of 1/2.

Given the probabilities of occurrence of symbols, the following procedure can be used to construct a Huffman code:

(1) Arrange the symbols in a rank-ordered list according to the probability of occurrence.

(2) Perform the following iterations to reduce the size of the list by creating composite symbols until a reduced list with only two composite symbols is reached.

Table 1: Huffman coding

Stage 1 (5 symbols)			Stage 2 (4 Symbols)			Stage 3 (3 symbols)			Stage 4 (2 symbols)		
Prob	Sym	Code	Prob	Sym	Code	Prob	Sym	Code	Prob	Sym	Code
0.500	a_1	0	0.500	a_1	0	0.500	a_1	0	0.500	a_1	0
0.125	a_2	110	0.250	a_{45}	10	0.250	a_{45}	10	0.500	a_{2-5}	1
0.125	a_3	111	0.125	a_2	110	0.250	a_{23}	11			
0.125	a_4	100	0.125	a_3	111						
0.125	a_5	101									

(i) Combine the two symbols with the lowest probability to form a composite symbol, and add the probabilities.

(ii) Create a new list from the old by deleting the two symbols that were combined, and adding the composite symbol in a rank-ordered manner.

(3) Construct the binary tree in which each node represents the probability of all nodes beneath it.

(4) Following a convention of assigning a "0" or "1" to the direction of movement in the tree, assign a code by traversing a path to each leaf.

Consider the case of coding the output of a source with $M = 5$ possible symbols $\{a_i, i = 1, \ldots, 5\}$, with probabilities $P_1 = \frac{1}{2}$, $P_2 = P_3 = P_4 = P_5 = \frac{1}{8}$. The entropy in this case is given by

$$H(S) = \sum_{i=1}^{M} P_i \log_2 \frac{1}{P_i} = \frac{1}{2}.1 + 4.\frac{1}{8}.3 = 2\text{bits/symbol}.$$

Huffman coding is performed as shown in Table 1. In stage 1 the symbol probabilities are arranged in decreasing order. The lowest two probabilities corresponding to symbols a_4 and a_5 are added and assigned to a new symbol a_{45}, which then appears in the shorter list of stage 2. Note that the symbols a_4 and a_5 are deleted, while a_{45} is inserted in the correct rank-ordered position. The procedure is repeated until in stage reduced to two symbols. These two symbols are assigned bits 0 and 1 to represent them. Next the component symbols a_{4-5} and a_{23} of the last new symbol a_{2-5} are assigned an additional bit 0 and one respectively. The procedure is repeated to traverse back to the first stage.

In the above case it can be verified that the average length of the codeword is $L_{av} = 2$ bits. A rate of 2 bits/symbol is equal to the entropy, and this is achieved because the symbol probabilities are integral powers of 1/2.

Now consider a case where the condition that the probabilities are integral powers of 1/2 does not hold. Assume that a source generates $M = 3$ symbols $\{a_i, i = 1, \ldots, 3\}$, with probabilities $P_1 = P_2 = P_3 = 1/3$. In this case the source entropy is 1.585 bits/symbol. With Huffman coding, it can be verified that a code $\{0, 10, 11\}$ is generated to represent the symbols with an average length

$$L_{av} = \frac{1}{3}(1 + 2 + 2) = 1.67 \text{ bits/symbol}$$

A question that can be asked is whether one can improve over this performance of 1.67 bits/symbol. This improvement can be achieved by jointly coding a block of symbols. Toward this end, let us define a new symbol as a block of N occurrences of original symbols. Let us denote

these M^N possible new symbols by $\{b_i, i = 1, \ldots, M^N\}$. By applying Huffman coding to the new symbols, it is possible to reduce the average number of bits per original symbol. For example, in the above case of $M = 3$ consider a block of $N = 5$ occurrences of original symbols $\{a_i, i = 1, \ldots, 3\}$, generating new symbols $\{b_i, i = 1, \ldots, 243\}$. Even without formally applying the Huffman coding procedure, we note a fixed codeword length of $L = 8$ bits can be used to uniquely represent the 243 possible blocks of $N = 5$ original symbols. This gives us a length of $8/5 = 1.6$ bits/original symbol, significantly closer to the bound of 1.585 bits/symbol.

In this simple case it was possible to use a suitable block size to get close to the bound. With a large set of symbols and widely varying probabilities, an alternative method called arithmetic coding proves to be more effective in approaching the bound in a systematic way.

3.2 Arithmetic Coding

As mentioned above Huffman coding is optimal only when the symbol probabilities are integral powers of 1/2. As this is not the case in practice, alternative techniques are needed to provide a better performance in the more general case. The technique of *arithmetic coding* [**?**] provides a solution to attaining the theoretical bound of the source entropy.

Arithmetic coding assigns a single variable-length code to a given realization of N symbols from the set $\{a_i, i = 1, \ldots, M\}$. Let us denote this realization as $\mathbf{x} = \{x_1, \ldots, x_N\}$. The code to represent $\mathbf{x}$ is obtained by associating with $\mathbf{x}$ a subinterval of $[0, 1)$. The length of the subinterval is equal to the probability of occurrence of the sequence $\mathbf{x}$. The binary representation of the end points of the intervals is used to represent $\mathbf{x}$ by a binary codeword that uniquely points to the subinterval associated with a symbol string.

A simple example is presented to explain the underlying idea. Consider an alphabet with $N = 3$ symbols, $\{a_1, a_2, a_3\}$. with probabilities $P_1 = \frac{1}{2}$, $P_2 = \frac{3}{8}$, and $P_3 = \frac{1}{8}$. Initially the symbol are assigned to subintervals of $[0.0, 1.0)$ whose lengths are proportional to their probabilities. Symbol a_1 is assigned to the interval $[0.0, 0.5)$, a_2 to $[0.5, 0.875)$, and a_3 to $[0.875, 1.0)$.

Let us determine the arithmetic code corresponding to the sequence of symbols:

$$a_1 \; a_2 \; a_1 \; a_3 \; \ldots$$

Since the first symbol is a_1, the sequence is initially assigned to the interval $I_1 = [0.0, 0.5)$. This interval is now subdivided into three intervals, again in proportion to the probabilities of the symbols: $[0.0, 0.25)$, $[0.25, 0.4375)$, $[0.4375, 0.5)$. Since the second symbol in the sequence is a_2, the sequence is now associated with the interval, $I_2 = [0.25, 0.4375)$. Continuing as before interval I_2 is split into three subintervals in proportion to the symbol probabilities. Since the third symbol is a_1, the new interval associated with the sequence is $I_3 = [0.25, 0.34375)$. Similarly the fourth symbol a_3 gives $I_4 = [0.33203125, 0.34375)$. Knowledge of the two boundaries of interval I_n at the n-th stage uniquely determines the sequence. This information has to be communicated to the decoder in a systematic way. This accomplished by first expressing the interval values in binary form. The bits corresponding to the lower bound of interval are compared with those of the upper bound, and the newly matching most significant bits at each stage are sent.

The binary expansion of [0.33203125 and 0.34375) are 0.01010101 and 0.01011. At stage $n = 4$, the first four bits, 0101 are matching, and constitute the coded bit-stream up to that stage.

The decoder operates on the bit-stream to successively narrow the interval, and determines the symbol sequence corresponding to that range. When the first bit ("0") is received, the interval is restricted to [0.0,0.5), sufficient to decide that the first symbol is a_1. The second bit ("1") restricts the interval to $[0.25, 0.5)$. This allows the second symbol to be either a_2 or a_3, and so there is

insufficient information to decide. The third bit ("0") narrows the interval to $[0.25, 0.375)$, helping decide that the second symbol is a_2. And so on.

====

Example: expalin each step with images

===

4 Variable Quantization

One of the main limitations of the JPEG standard is the fact that visible artifacts can often appear in the decompressed image at moderate to high compression ratios. This is especially true for parts of the image containing graphics, text, or some other such synthesized component. Artifacts are also common in smooth regions and in image blocks containing a single dominant edge. One approach to deal with this problem is to change the "coarseness" of quantization as a function of image characteristics in the block being compressed.

In order to alleviate this problem, the latest extension of the JPEG standard, called JPEG Part-3, allows rescaling of quantization matrix Q on a block by block basis, thereby potentially changing the manner in which quantization is performed for each block. The scaling operation is not done on the DC coefficient $Y[0,0]$, which is quantized in the same manner as baseline JPEG. The remaining 63 AC coefficients $Y[u,v]$ are quantized as follows:

$$\hat{Y}[u,v] = \left[\frac{Y[u,v] \times 16}{Q[u,v] \times \text{Qscale}}\right].$$

Where QScale is a parameter that can take on values from 1 to 112 (default 16). In order for the decoder to correctly recover the quantized AC coefficients, it needs to know the value of QScale used by the encoding process. The standard specifies the exact syntax by which the encoder can specify change in QScale values. If no such change is signaled then the decoder continues using the QScale value that is in current use. The overhead incurred in signaling a change in the scale factor is approximately 15 bits depending on the Huffman table being employed.

It should be noted that the standard only specifies the syntax by means of which the encoding process can signal changes made to the QScale value. It does not specify how the encoder may determine if a change in QScale is desired and what the new value of QScale should be. Typical methods for variable quantization propsed in the literature utilize the fact that the human visual system is less sensitive to quantization errors in highly active regions of the image. Quantization errors are frequently more perceptible in blocks that are smooth or contain a single dominant edge. Hence, prior to quantization, they compute a few simple features for each block. These features are used to classify the block as either smooth, edge or texture etc. Based on this classification, and a simple activity measure computed for the block, a QScale value is computed.

Chun, et. al [**?**] have proposed a block classification scheme in the context of video coding. Their scheme also classifies blocks as being either smooth, edge, or texture, and defines several parameters in the DCT domain as shown below:

E_h: horizontal energy	E_v: vertical energy	E_d: diagonal energy
E_a: $avg(E_h, E_v, E_d)$	E_m: $min(E_h, E_v, E_d)$	E_M: $max(E_h, E_v, E_d)$
$E_{m/M}$: ratio of E_m and E_M		

E_a represents the average high frequency energy of the block, and is used to distinguish between low activity blocks and high activity blocks. Low activity (smooth) blocks satisfy the relationship, $E_a \leq T_1$, where T_1 is a small constant. High activity blocks are further classified into texture blocks and

edge blocks. Texture blocks are detected under the assumption that they have relatively uniform energy distribution in comparison with edge blocks. Specifically, a block is deemed to be a texture block if it satisfies the conditions: $E_a > T_1$, $E_{min} > T_2$, and $E_{m/M} > T_3$, where T_1, T_2 and , T_3 are experimentally determined constants. All blocks which fail to satisfy the smoothness and texture tests are classified as edge blocks.

Tan, Pang and Ngan [**?**] have developed an algorithm for variable quantization for the H.263 video coding standard. They compute quantization scale factors for a macroblock based on a perceptual classification in the DCT domain. Macroblocks are classified as flat, edge, texture or fine-texture. The classification algorithm first computes the texture energy $T_E(k)$ of the $k'th$ macro-block to be

$$T_E(k) = \rho \left[\sum_{i=0(i,j)\neq(0,0)}^{N-1} \sum_{j=0}^{N-1} \hat{H}^{-1}(i,j)^2 \cdot X[i,j]^2 \right]^{\gamma} \tag{1}$$

where $\hat{H}^{-1}(f)$ is a weighting function modeling the sensitivity of the Human Visual System (HVS) and γ and ρ are constants. After computing the texture energy, macro-block classification is done by a complex process which may often require more than one pass of the data.

Konstantinides and Tretter [**?**] give an algorithm for computing QScale factors for improving text quality on compound documents. They compute an activity measure M_i for each image block as a function of the DCT coefficients as follows:

$$M_i = \frac{1}{64} \left[\log_2 |Y_i[0,0] - Y_{i-1}[0,0]| + \sum_{j,k} \log_2 |Y_i[j,k]| \right], \tag{2}$$

The QScale value for the block is then computed as:

$$\text{QScale}_i = \begin{cases} a \times M_i + b & if 2 > a \times M_i + b \geq 0.4 \\ 0.4 & a \times M_i + b \geq 0.4 \\ 2 & a \times M_i + b > 2. \end{cases} \tag{3}$$

The technique is only designed to detect text regions and will quantize high activity textured regions in the image part at the same scale as text regions. Clearly, this is not optimal as high activity textured regions can be quantized very coarsely leading to improved compression. In addition, the technique does not address smooth blocks where artifacts are often the first to appear.

5 JPEG Modes of Operation

What has been described thus far in this chapter represents the JPEG *sequential DCT mode*. The sequential DCT mode is the most commonly used mode of operation of JPEG and is required to be supported by any baseline implementation of the standard. However, in addition to the sequential DCT mode, JPEG also defines a *progressive DCT mode*, *sequential lossless mode*, and a *hierarchical mode*. In figure 3 we show how the different modes can be used. For example, the hierarchical mode could be used in conjunction of any of the other modes as shown in the figure. In the lossless mode, JPEG uses an entirely different algorithm based on predictive coding as described in detail in the next chapter. In this section we restrict ourselves to lossy compression and describe in more detail the DCT based progressive and hierarchical modes of operation in more detail.

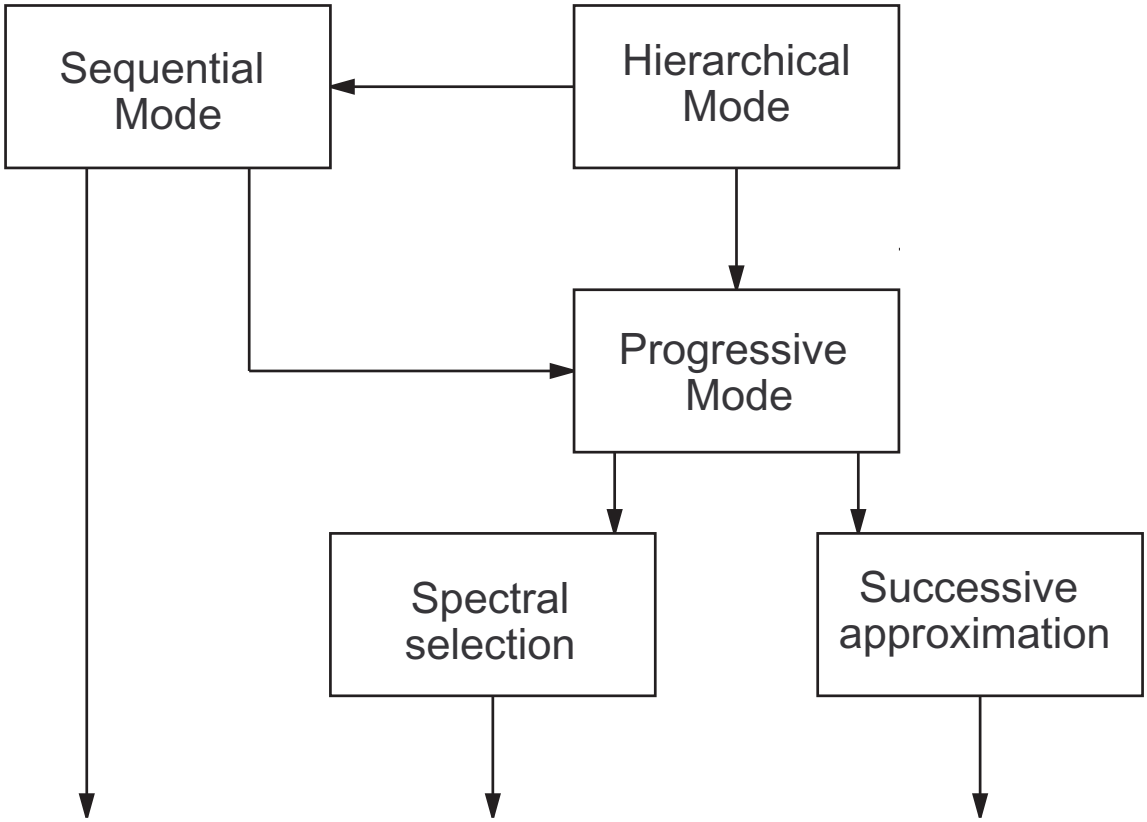

Figure 3: JPEG modes of operation

5.1 Progressive Mode

In some applications it may be advantageous to transmit an image in multiple passes, such that after each pass an increasingly accurate approximation to the final image can be constructed at the receiver. In the first pass very few bits are transmitted and the reconstructed image is equivalent to one obtained with a very low quality setting. Each of the subsequent passes contain an increasing number of bits which are used to refine the quality of the reconstructed image. The total number of bits transmitted is roughly the same as would be needed to transmit the final image by the sequential DCT mode. One example of an application which would benefit from progressive transmission, is provided by the World-Wide-Web (WWW), where a user might want to start examining the contents of the entire page without waiting for each and every image contained in the page to be fully and sequentially downloaded. Other examples include remote browsing of image databases, tele-medicine and network centric computing in general. JPEG contains a progressive mode of coding that is well suited to such applications. The disadvantage of progressive transmission of course is that the image has to be decoded multiple number of times and only makes sense if the decoder is faster than the communication link.

In the progressive mode, the DCT coefficients are encoded in a series of scans. JPEG defines two ways for doing this: *spectral selection* and *successive approximation*. In the spectral selection mode, DCT coefficients are assigned to different groups according to their position in the DCT block and during each pass, the DCT coefficients belonging to a single group are transmitted. For example, consider the following grouping of the 64 DCT coefficients numbered from 0 to 63 in the zig-zag scan order

$$\{0\}, \{1, 2, 3\}, \{4, 5, 6, 7\}, \{8, \ldots, 63\}.$$

Here, only the DC coefficient is encoded in the first scan. This is a requirement imposed by the standard. In the progressive DCT mode, DC coefficients are always sent in a separate scan. The second scan codes the first three AC coefficients in zig-zag order, the third scan encodes the next four AC coefficients and the fourth and the last scan encodes the remaining coefficients. JPEG provides the syntax for specifying the starting coefficient number and the final coefficient number being encoded in a particular scan. This limits a group of coefficients being encoded in any given scan to be successive in the zig-zag order. As figure **??** shows, the first few DCT coefficients are often sufficient to give a reasonable rendition of the image. In fact, just the DC coefficient can serve to essentially identify the contents of an image, although the reconstructed image contains severe blocking artifacts. It should be noted that after all the scans are decoded, the final image quality is the same as that would be obtained by a sequential mode of operation. The bit rate however, can be different as the entropy coding procedures for the progressive mode are different as described later in this section.

In successive approximation coding, the DCT coefficients are sent in successive scans with increasing level of precision. The DC coefficient however, just as in spectral selection coding, is sent in the first scan with full precision. The AC coefficients are sent bit-plane by bit-plane, starting from the most significant bit plane to the least significant bit plane.

The entropy coding techniques used in the progressive mode are slightly different than those used in the sequential mode. Since the DC coefficient is always sent as a separate scan, the Huffman and arithmetic coding procedures used remain the same as those in the sequential mode. However, coding of the AC coefficients is done a bit differently. In spectral selection coding (without selective refinement) and in the first stage of successive approximation coding, a new set of symbols are defined to indicate runs of End-Of-Block (EOB) codes. Recall, in the sequential mode, the EOB code indicates that the rest if the block contains zero coefficients. With spectral selection, each scan contains only a few AC coefficients and the probability of encountering EOB is significantly higher.

Similarly, in successive approximation coding each block consists of reduced precision coefficients, leading again to a large number of EOB symbols being encoded. Hence, to exploit this fact and acheive further reduction in bit rate, JPEG defines an additional set of fifteen symbols $EOBn$ each representing a run of 2^n EOB codes. After each $EOBi$ run-length code, extra i bits are appended to specify the exact run-length.

It should be noted that the two progressive modes, spectral selection and successive refinement can be combined to give successive approximation in each spectral band being encoded. This results in a quite a complex codec, which to our knowledge is rarely used.

It is possible to transcode between progressive JPEG and sequential JPEG without any loss in quality and approximately maintaining the same bit rate. Spectral selection results in bit-rates slightly higher the sequential mode whereas successive approximation often results in lower bit rates. The differences however, are small. There have not been many implementation of progressive JPEG codecs. There has been some interest in them recently due to the proliferation of images on the World-Wide-Web. It is expected that more public domain progressive JPEG codec will be available in the future.

5.2 Hierarchical Mode

The hierarchical mode defines another from of progressive transmission where the image is decomposed into a pyramidal structure of increasing resolution. The top most layer in the pyramid representing the image at the lowest resolution and the base of the pyramid representing the image at full resolution. There is a doubling of resolutions both in the horizontal and vertical dimensions, between successive levels in the pyramid. Hierarchical coding is useful when an image could be displayed at different resolutions devices, like hand held devices, computer monitors of varying resolutions and high resolution printers. In such a scenario, having a multi-resolution representation allows the transmission of the appropriate layer to each requesting device, thereby making full use of available bandwidth.

In the JPEG hierarchical mode, each image component is encoded as a sequence of frames. The lowest resolution frame (level 1) is encoded using one of the sequential or progressive modes. The remaining levels are encoded differentially. That is, an estimate I_i' of the image, I_i, at the $i'th$ level ($i \geq 2$) is first formed by up-sampling the low resolution image I_{i-1} from the layer immediately above. Then the difference between I_i' and I_i is then encoded using modifications of the DCT based modes or the lossless mode. If lossless mode is used to code each refinement, then the final reconstruction at the base layer is lossless. The up-sampling filter used is a bi-linear interpolating filter that is specified by the standard and cannot be specified by the used. Starting from the high resolution image, successive low resolution images are created essentially by down-sampling by two in each direction. The exact down-sampling filter to be used is not specified but the standard cautions that the down-sampling filter used be consistent with the fixed up-sampling filter. Note that the decoder does not need to know what down-sampling filter was used in order to decode a bit stream. Figure 4 depicts the sequence of operations performed at each level of the hierarchy.

Since the differential frames are already signed values they are not level shifted prior to FDCT. Also, the DC coefficient is coded directly rather than differentially. Other than these two facts the Huffman coding model in the progressive mode is the same as that used in the sequential mode. Arithmetic coding is however, done a bit differently with conditioning states based on differences with pixel to the left as well as the one above being utilized. For details the user is referred to [**?**].

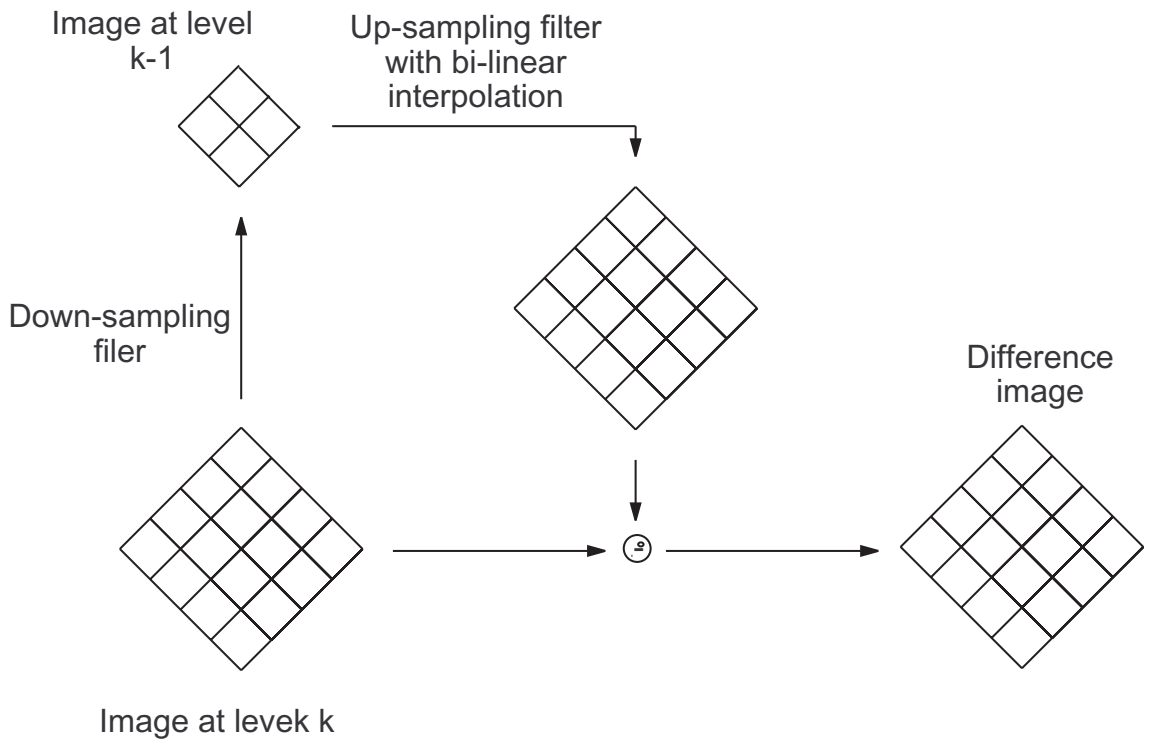

Figure 4: JPEG modes of operation

6 Additional features

JPEG Part 3 defines a tiling capability whereby an image is subdivided into blocks or tiles, each coded independently. Tiling facilitates the following features

- Display image region on given screen size
- Accelerate access to image sub-region
- Refine region of interest Protect copying of large images

As shown in figure xxx, tiling can be

- Simple - Non-overlapping , same size (except edges)
- Pyramidal - may overlap across resolutions
- Composite

7 Discussion and Future Directions

It has been almost 10 years now since the development of the JPEG standard and during these years JPEG has rapidly become the most widely used image compression standard.

Although JPEG gives excellent performance with continuous-tone (or photographic) images, it fails to provide adequate compression and/or good quality with synthetic images like cartoons, logos or computer graphic images like ray-traces scenes. This is because logos and cartoons have very few colors and large areas of constant intensity. Such images contain very little information and consequently require very few bits to represent. JPEG's block based approach requires a minimum number of bits to represent a block and this can result in excessive bits being used when dealing with low information images or regions.

Another limitation of JPEG that often precludes its choice in practice is the fact that the algorithm is not idempotent. That is, successive compression/decompression cycles can lead to increasing deterioration in image quality due to quantization round off errors. This is especially true if successive compression/decompression is done with different quality settings or if the image is altered in a significant way in between. This makes JPEG unsuitable for representing intermediate images generated while editing image content in professional production environments.

8 Additional Information

The best source of information on the JPEG compression standard is the excellent book by Pennebaker and Mitchell [**?**]. This book also contains the entire text of the official committee draft international standard ISO DIS 10918-1 and ISO DIS 10918-2. The book has not been revised since its first publication in 1993 and hence later extensions to the standard, incorporated in JPEG Part 3 are not covered. The official standards document xxx is the only source for JPEG part 3.

The JPEG committee maintains an official website at www.jpeg.org which contains general information about the committee and its activities, announcements and other useful links related to the different JPEG standards.

Free, portable C code for JPEG compression is available from the Independent JPEG Group (IJG). Source code, documentation, and test files are included. Version 6b is available from ftp.uu.net:/graphics/jpeg/jpegsrc.v6b.tar.gz and in ZIP archive format at ftp.simtel.net:/pub/simtelnet/msdos/gra

The IJG code includes a reusable JPEG compression/decompression library, plus sample applications for compression, decompression, transcoding and file format conversion. The package is highly portable; it has been used successfully on many machines ranging from Apple IIs to Crays. The IJG code is free for both noncommercial and commercial use; only an acknowledgement in your documentation is required to use it in a product. A different free JPEG implementation, written by the PVRG group at Stanford, is available from havefun.stanford.edu:/pub/jpeg/JPEGv1.2.1.tar.Z. The PVRG code is designed for research and experimentation rather than production use; it is slower, harder to use, and less portable than the IJG code, but the PVRG code is easier to understand.

"Discrete Cosine Transform—Algorithms, Advantages, Applications" by K.R. Rao and P. Yip (Academic Press, London, 1990),ISBN 0-12-580203-X.

Polynomial Transform Computation of the 2-D DCT, Duhamel Guillemot, ICASSP '90 p. 1515.

A Forward-Mapping Realization of the Inverse DCT, McMillan Westover, DCC '92 p. 219.

A Fast Algorithm for 2-D DCT, Cho, Yun Lee, ICASSP '91 p. 2197.

Fast Algorithm and Implementation of 2-D DCT, Cho Lee, Tr. CAS v38 p. 297.

A DCT Chip based on a new Structured and Computationally Efficient DCT Algorithm, Duhamel, Guillemot Carlach, ICCAS '90 p. 77.

Trade-offs in the Computation of Mono- and Multi-dimensional DCTs, Vetterli, Duhamel Guillemot, ICASSP '89 p. 999.

Practical Fast 1-D DCT Algorithms with 11 Multiplications, Loeffler, Ligtenberg Moschytz, ICASSP '89 p. 988.

New Scaled DCT Algorithms for Fused Multiply/Add Architectures, Linzer Feig, ICASSP '91 p. 2201.

Fast Algorithms for the 2-D Discrete Cosine Transform, Kamangar Rao, IEEE Tr. Computers, v C-31 p. 899.

Fast 2-D Discrete Cosine Transform, Vetterli, ICASSP '85 p. 1538.

A Two-Dimensional Fast Cosine Transform, Haque, Tr. ASSP v ASSP-33 p. 1532.

Real-Time Parallel and Fully Pipelined 2-D DCT Lattice Structures with Application to HDTV Systems, Chiu Liu, Tr. CAS for Video Tech, v 2 p. 25.

J.F. Blinn, "What's the Deal with the DCT", IEEE Computer Graphics and Applications, July 1993, pp.78-83.

A C Hung and TH-Y Meng, "A Comparison of fast DCT algorithms, Multimedia Systems", No. 5 Vol. 2, Dec 1994