5.5

Lossy Image Compression: JPEG and JPEG2000 Standards

Rashid Ansari
University of Illinois at Chicago

Christine Guillemot
INRIA, Rennes

Nasir Memon
Polytechnic University

1 Introduction

JPEG is currently a worldwide standard for compression of digital images. The standard is named after the committee that created it and continues 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 standardization effort is led by the International Standards Organization (ISO) and the International Telecommunications Union—Telecommunication Standardization Sector (ITU—T). 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 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 multipart set of ISO standards IS 10918-1, 2, 3 (ITU-T recommendations T.81, T.83, T.84).

The JPEG standardization activity commenced in 1986, which generated 12 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) submitted in 1991 culminated in the International Standard (IS) approved in 1992.

Although the JPEG Standard defines both lossy and lossless compression algorithms, the focus in this chapter is on the lossy compression component of the JPEG standard. The JPEG lossless standards are described in detail in chapter 5.6 of this handbook [32]. JPEG lossy compression entails an irreversible mapping of the image to a compressed bit stream, but the standard provides mechanisms for a controlled loss of information. Lossy compression produces a bit stream that is usually much smaller in size than that produced with lossless compression.

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

- 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 on a 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 about 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, and so forth, are permitted.
- A graceful tradeoff in bit rate and quality is offered, except at very low bit rates.
- A hierarchical mode with multiple levels of resolution is allowed.
- Bit resolution of 8 to 12 bits is permitted.
- A recommended file format, JPEG file interchange format (JFIF), enables the exchange of JPEG bit streams among a variety of platforms.

A JPEG-compliant decoder has to support a minimum set of requirements, the implementation of which is collectively referred to as *baseline implementation*. Additional features are supported in the extended implementation of the standard. The features supported in the baseline implementation include the ability to provide

- A sequential build up
- Custom or default Huffman tables
- 8-bit precision per pixel for each component
- Image scans with one to four components
- Both interleaved and noninterleaved scans

A JPEG extended system includes all features in a baseline implementation and supports many additional features. It allows sequential build up as well as an optional progressive build up. Either Huffman coding or arithmetic coding can be used in the entropy coding unit. Precision of up to 12 bits per pixel is allowed. The extended system includes an option for lossless coding.

The JPEG standard suffers from shortcomings in compression efficiency and progressive decoding. This led the JPEG committee to launch an effort in late 1996 and early 1997 to create a new image compression standard. The initiative has resulted in the 15444/ITU-T Recommendation T.8000 known as the JPEG2000 standard that is based on wavelet analysis and encoding. The new standard is described in some detail in this chapter.

The rest of this chapter is organized as follows. In Section 2, we describe the structure of the JPEG codec and its units. In Section 3, the role and computation of the discrete cosine transform is examined. Procedures for quantizing the DCT coefficients are presented in Section 4. In Section 5, the mapping of the quantized DCT coefficients into symbols suitable for entropy coding is described. Syntactic issues and organization of data units are discussed in Section 6. Section 7 describes alternative modes of operation like the progressive and hierarchical modes. In Section 8, some extensions made to the standard, collectively known as JPEG Part 3, are described. Sections 9 and 10 provide a description of the the new JPEG2000 standard and its coding architecture. The performance of JPEG2000 and the extensions included in part 2 of the standard are briefly described in Section 11. Finally, Section 12 lists further sources of information on the standards.

2 Lossy JPEG Codec Structure

It should be noted that in addition to defining an encoder and decoder, the JPEG standard also defines a syntax for representing the compressed data along with the associated tables and parameters. In this chapter, however, we largely ignore these syntactic issues and focus instead on the encoding and decoding procedures. We begin by examining the structure of the JPEG encoding and decoding systems. The discussion centers on the encoder structure and the building blocks comprising an encoder. The decoder essentially consists of the inverse operations of the encoding process carried out in reverse.

2.1 Encoder Structure

The JPEG encoder and decoder are conveniently decomposed into units that are shown in Fig. 1. Note that the encoder shown in Fig. 1 is applicable in open-loop/unbuffered environments where the system is not operating under a

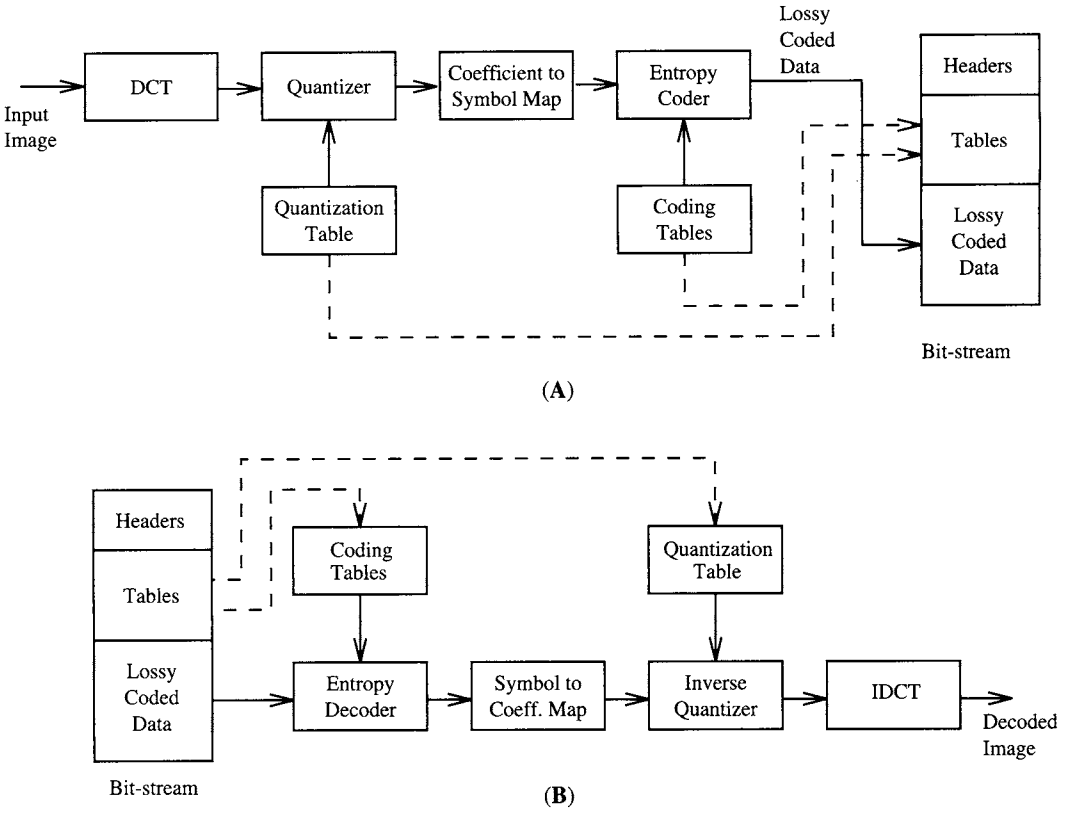

FIGURE 1 Constituent units of (A) Joint Photographic Experts Group (JPEG) encoder (B) JPEG decoder.

constraint of a prescribed bit rate/budget. The units constituting the encoder are described next.

2.1.1 Signal Transformation Unit: DCT

In JPEG image compression, each component array in the input image is first partitioned into 8×8 rectangular blocks of the data. A signal transformation unit computes the DCT of each 8×8 block 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 to present the information in a "machine-friendly" form that is convenient for disregarding the perceptually least relevant content. The DCT captures the spatial redundancy and packs the signal energy into a few DCT coefficients. The coefficient with zero frequency in both dimensions is called direct current (DC) coefficient, and the remaining 63 coefficients are called alternating current (AC) coefficients.

2.1.2 Quantizer

If we wish to recover the original image exactly from the DCT coefficient array, it is necessary to represent the DCT coefficients with high precision. Such a representation requires a large number of bits. In lossy compression, the DCT coefficients are mapped into a relatively small set of possible values that are represented compactly by defining and coding suitable symbols. The quantization unit performs this task of a many-to-one mapping of the DCT coefficients, so that the possible outputs are limited in number. A key feature of the quantized DCT coefficients is that many of them are zero, making them suitable for efficient coding.

2.1.3 Coefficient-to-Symbol Mapping Unit

The quantized DCT coefficients are mapped to new symbols 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. An effective definition of symbols for representing AC coefficients in JPEG is the "runs" of zero coefficients followed by a nonzero terminating coefficient. For representing DC coefficients, symbols are defined by computing the difference between the DC coefficient in the current block and that in the previous block.

2.1.4 Entropy 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. Huffman coding is usually used for variable-length coding of the symbols, with arithmetic coding allowed as an option.

2.2 Decoder Structure

In a decoder, the inverse operations are performed and in an order that is the reverse of that in the encoder. The coded bit stream contains coding and quantization tables that are first extracted. The coded data are then applied to the entropy decoder that determines the symbols coded. The symbols are then mapped to an array of quantized DCT coefficients, which are then "de-quantized" by multiplying each coefficient with the corresponding entry in the quantization table. The decoded image is then obtained by applying the inverse two-dimensional (2D) DCT to the array of the recovered DCT coefficients in each block of the image.

In the next three sections we consider each of the above encoder operations—DCT, quantization, and symbol mapping and coding, in more detail.

3 Discrete Cosine Transform

Lossy JPEG compression is based on the use of transform coding using the DCT [2]. In DCT coding, each component of the image is subdivided into blocks of 8×8 pixels. A 2D DCT is applied to each block of data to obtain an 8×8 array of coefficients. If $x[m, n]$ represents the image pixel values in a block, then the DCT is computed for each block of the image data as follows:

$$X[u,v] = \frac{C[u]C[v]}{4} \sum_{m=0}^{7} \sum_{n=0}^{7} x[m,n] \cos\frac{(2m+1)u\pi}{16} \cos\frac{(2n+1)v\pi}{16} \qquad 0 \le u, v \le 7$$

where

$$C[u] = \begin{cases} \frac{1}{\sqrt{2}} & u = 0, \\ 1 & 1 \le u \le 7. \end{cases}$$

The original image samples can be recovered from the DCT coefficients by applying the inverse discrete cosine transform (IDCT) as follows:

$$x[m,n] = \sum_{u=0}^{7} \sum_{v=0}^{7} \frac{C[u]C[v]}{4} X[u,v] \cos\frac{(2m+1)u\pi}{16} \cos\frac{(2n+1)v\pi}{16}, \qquad 0 \le m, n \le 7.$$

The DCT, which belongs to the family of sinusoidal transforms, has received special attention due to its success in compression of real-world images. It is seen from the definition of the DCT that an 8×8 image block being transformed is being represented as a linear combination of real-valued basis vectors that consist of samples of a product of one-dimensional (1D) cosinusoidal functions. The 2D transform can be expressed as a product of 1D DCT transforms applied separably along the rows and columns of the image block. The coefficients $X(u, v)$ of the linear combination are referred to as the *DCT coefficients*. For real-world digital images in which the interpixel correlation is reasonably high and that can be characterized with first-order autoregressive models, the performance of the DCT is very close to that of the Karhunen-Loeve transform [2]. The discrete Fourier transform (DFT) is not as efficient as DCT in representing an 8×8 image block. This is because when the DFT is applied to each row of the image, a periodic extension of the data, along with concomitant edge discontinuities, produces high-frequency DFT coefficients that are larger than the DCT coefficients of corresponding order. On the other hand, there is a mirror-periodicity implied by the DCT that avoids the discontinuities at the edges when image blocks are repeated. As a result, the "high-frequency" or "high-order AC" coefficients are on the average smaller than the corresponding DFT coefficients.

We consider an example of the computation of the 2D DCT of an 8×8 block in the 512×512 gray-scale image *Lena*. The specific block chosen is shown in the image in Fig. 2top where the block is indicated with a black boundary with one corner

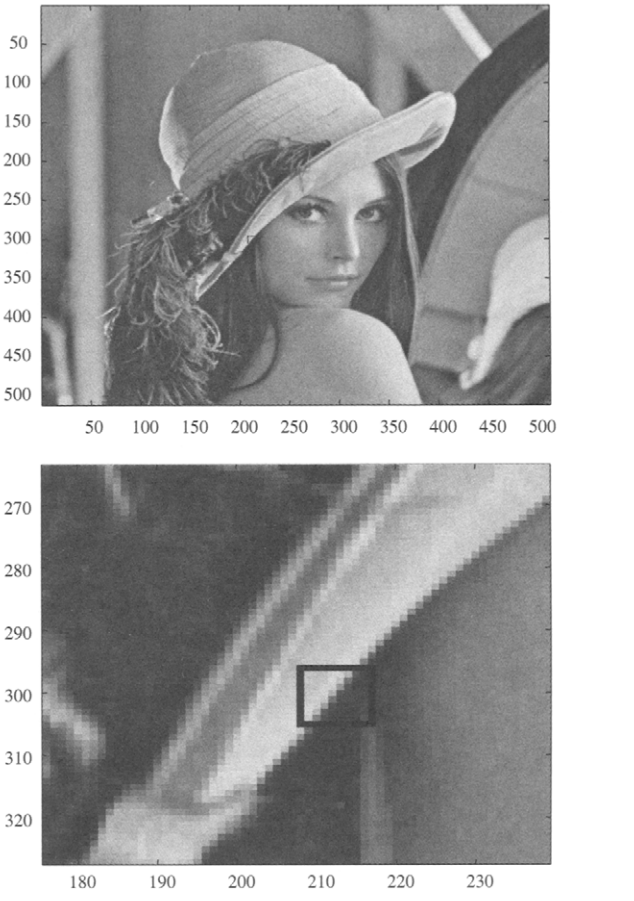

FIGURE 2 The original 512×512 *Lena* image (**top**) with an 8×8 block (**bottom**) identified with black boundary and with one corner at [208, 296].

at [208, 296]. A close up of the block enclosing part of the hat is shown Fig. 2bottom.

The 8-bit pixel values of the block chosen are shown in Fig. 3. After the DCT is applied to this block, the 8×8 DCT coefficient array obtained is shown in Fig. 4.

The magnitude of the DCT coefficients exhibits a pattern in their occurrences in the coefficient array. Also their contribution to the perception of the information is not uniform across the array. The DCT coefficients corresponding to the lowest frequency basis functions are usually large in magnitude and are also deemed to be perceptually most significant. These qualities are exploited in developing methods of quantization and symbol coding. The bulk of the compression achieved in transform coding occurs in the quantization step. The compression level is controlled by changing the total number of bits available to encode the blocks. The coefficients are quantized more coarsely when a large compression factor is required.

4 Quantization

Each DCT coefficient $X[m, n]$, $0 \leq m, n \leq 7$, is mapped into one of a finite number of levels determined by the compression factor desired.

187	188	189	202	209	175	66	41
191	186	193	209	193	98	40	39
188	187	202	202	144	53	35	37
189	195	206	172	58	47	43	45
197	204	194	106	50	48	42	45
208	204	151	50	41	41	41	53
209	179	68	42	35	36	40	47
200	117	53	41	34	38	39	63

FIGURE 3 The 8×8 block identified in Fig. 2.

915.6	451.3	25.6	-12.6	16.1	-12.3	7.9	-7.3
216.8	19.8	-228.2	-25.7	23.0	-0.1	6.4	2.0
-2.0	-77.4	-23.8	102.9	45.2	-23.7	-4.4	-5.1
30.1	2.4	19.5	28.6	-51.1	-32.5	12.3	4.5
5.1	-22.1	-2.2	-1.9	-17.4	20.8	23.2	-14.5
-0.4	-0.8	7.5	6.2	-9.6	5.7	-9.5	-19.9
5.3	-5.3	-2.4	-2.4	-3.5	-2.1	10.0	11.0
0.9	0.7	-7.7	9.3	2.7	-5.4	-6.7	2.5

FIGURE 4 Discrete cosine transform of the 8×8 block in Fig. 3.

4.1 DCT Coefficient Quantization Procedure

Quantization is done by dividing each element of the DCT coefficient array by a corresponding element in an 8×8 *quantization matrix* and rounding the result. Thus, if the entry $q[m, n]$, $0 \leq m, n \leq 7$, in the m-th row and n-th column of the quantization matrix, is large then the corresponding DCT coefficient is coarsely quantized. The values of $q[m, n]$ are restricted to be integers with $1 \leq q[m, n] \leq 255$, and they determine the quantization step for the corresponding coefficient. The quantized coefficient is given by:

$$qX[m, n] = \left[\frac{X[m, n]}{q[m, n]}\right]_{\text{round}}.$$

A quantization table(or matrix) is required for each image component. However a quantization table can be shared by multiple components. For example in a luminance-plus-chrominance $Y - Cr - Cb$ representation, the two chrominance components usually share a common quantization matrix. JPEG quantization tables given in Annex K of the standard for luminance and components are shown in Fig. 5. These tables were obtained from a series of psycho-visual experiments to determine the visibility thresholds for the DCT basis functions for a 760×576 image with chrominance components down-sampled by two in the horizontal direction and at a viewing distance equal to six times the screen width. On examining the tables, it is seen that the quantization table for the chrominance components has larger values in general implying a coarser quantization of the chrominance planes as compared with the luminance plane. This is done to exploit the human visual system's relative insensitivity to chrominance components as compared with luminance components. The tables shown have been known to offer satisfactory performance, on the average, over a wide variety of applications and viewing conditions. Hence, they have been widely accepted and over the years have become known as the "default" quantization tables.

Quantization tables can also be constructed by casting the problem as one of optimum allocation of a given budget of bits based on the coefficient statistics. The general principle is to estimate the variances of the DCT coefficients and assign more bits to coefficients with larger variances.

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

FIGURE 5 Example quantization tables for luminance (**left**) and chrominance (**right**) components provided in the informative sections of the standard.

57	41	2	0	0	0	0	0
18	1	-16	-1	0	0	0	0
0	-5	-1	4	1	0	0	0
2	0	0	0	-1	0	0	0
0	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

FIGURE 6 8 × 8 discrete cosine transform block in Fig. 4 after quantization with the luminance quantization table shown in Fig. 5.

181	185	196	208	203	159	86	27
191	189	197	203	178	118	58	25
192	193	197	185	136	72	36	33
184	199	195	151	90	48	38	43
185	207	185	110	52	43	49	44
201	198	151	74	32	40	48	38
213	161	92	47	32	35	41	45
216	122	43	32	39	32	36	58

FIGURE 7 The block selected from the *Lena* image recovered after decoding.

We now examine the quantization of the DCT coefficients given in Fig. 4 using the luminance quantization table in Fig. 5. Each DCT coefficient is divided by the corresponding entry in the quantization table and the result is rounded to yield the array of quantized DCT coefficients in Fig. 6. We observe that a large number of quantized DCT coefficients are zero making the array suitable for runlength coding as described in Section 6. The block recovered after decoding is shown in Fig. 7.

4.2 Quantization Table Design

With lossy compression, the amount of distortion introduced in the image is inversely related to the number of bits (bit rate) used to encode the image. The higher the rate, the lower the distortion. Naturally, for a given rate, we would like to incur the minimum possible distortion. Similarly, for a given distortion level, we would like to encode with the minimum rate possible. Hence, lossy compression techniques are often studied in terms of their rate-distortion performance. That is, the distortion they introduce over different bit rates. The rate-distortion performance of JPEG is determined mainly by the quantization tables. As mentioned before, the standard does not recommend any particular table or set of tables and leaves their design completely to the user. While the image quality obtained from the use of the "default" quantization tables described earlier is very good, there is a need to provide flexibility to adjust the image quality by changing the overall bit rate. In practice, scaled versions of the "default" quantization tables are very commonly used to vary the quality and compression performance of JPEG. For example, the popular IJPEG implementation, freely available in the public domain, allows this adjustment through the use of *quality factor* Q for scaling all elements of the quantization table. The scaling factor is then computed as

$$\text{Scale factor} = \begin{cases} \dfrac{5000}{Q} & \text{for} \quad 1 \le Q < 50 \\ 200 - 2 * Q & \text{for} \quad 50 \le Q \le 99 \\ 1 & \text{for} \quad Q = 100 \end{cases} \tag{1}$$

Although varying the rate by scaling a base quantization table according to some fixed scheme is convenient, it is clearly not optimal. Given an image and a bit rate, there exists a quantization table that provides the "optimal" distortion at the given rate. Clearly, the "optimal" table would vary with different images and different bit rates and even different definitions of distortion (for example mean square error (MSE), perceptual distortion, and so forth). To get the best performance from JPEG in a given application, custom quantization tables may need to be designed. Indeed, there has been a lot of work reported in the literature addressing the issue of quantization table design for JPEG. Broadly speaking, this work can be classified into three categories. The first deals with explicitly optimizing the rate-distortion performance of JPEG on the basis of statistical models for DCT coefficient distributions. The second attempts to optimize the visual quality of the reconstructed image at a given bit rate, given a set of display conditions and a perception model. The third addresses constraints imposed by applications, such as optimization for printers.

An example of the first approach is provided by the work of Ratnakar and Livny [12] who propose RD-OPT, an efficient algorithm for constructing quantization tables with optimal rate-distortion performance for a given image. The RD-OPT algorithm uses DCT coefficient distribution statistics from any given image in a novel way to optimize quantization tables simultaneously for the entire possible range of compression-quality trade-offs. The algorithm is restricted to the MSE-related distortion measures as it exploits the property that the DCT is a unitary transform, that is, mean-square error in the pixel domain is the same as mean-square error in the DCT-domain. The RD-OPT essentially consists of the following three stages:

1. Gathers DCT statistics for given image or set of images. Essentially this step involves counting how many times the nth coefficient gets quantized to the value v when the quantization step-size is q and what is the mean squared error for the nth coefficient at this step size.
2. Use statistics collected above to calculate $R_n(q)$, the rate for the nth coefficient when the quantization step size is q and $D_n(q)$, the corresponding distortion for each possible q. The rate $R_n(q)$ is estimated from the corresponding first order entropy of the coefficient at the given quantization step size.

3. Compute $R(Q)$ and $D(Q)$ the rate and distortions for a quantization table Q as

$$R(Q) = \sum_{n=0}^{63} R_n(Q[n]); \text{ and } D(Q) = \sum_{n=0}^{63} D_n(Q[n])$$

respectively. Use dynamic programming to optimize $R(Q)$ against $D(Q)$.

Optimizing quantization tables with respect to MSE may not be the best strategy when the end image is to be viewed by a human. A better approach is to match the quantization table to the a human visual system (HVS) model. As mentioned before, the "default" quantization tables were arrived at in an *image-independent* manner, on the basis of the visibility of the DCT basis functions. Clearly, better performance could be achieved by an *image-dependent* approach that exploits HVS properties like frequency, contrast and texture masking, and sensitivity. A number of HVS model–based techniques for quantization table design have been proposed in the literature [3, 7, 17]. Such techniques perform an analysis of the given image and arrive at a set of thresholds, one for each coefficient, called the just noticeable distortion (JND) thresholds. The underlying idea is that if the distortion introduced is at or just below these thresholds, the reconstructed image will be perceptually distortion free.

Optimizing quantization tables with respect to MSE may also not be appropriate when there are constraints on the type of distortion that can be tolerated. For example, on examining Fig. 5, it is clear that the "high-frequency" AC quantization factors (i.e., $q[m, n]$ for larger values of m and n) are significantly greater than the DC coefficient $q[0, 0]$ and the "low-frequency" AC quantization factors. There are applications in which the information of interest in an image may reside in the high-frequency AC coefficients. For example, in compression of radiographic images [14], the critical diagnostic information is often in the high-frequency components. The size of microcalcification in mammograms is often so small that a coarse quantization of the higher AC coefficients will be unacceptable. In such cases, JPEG allows custom tables to be provided in the bit streams.

Finally, quantization tables can also be optimized for hard-copy devices like printers. JPEG was designed for compressing images that are to be displayed on cathode ray tube-like (CRT) display devices that can represent a large range of pixel intensities. Hence, when an image is rendered through a half-tone device like a printer, the image quality could be far from optimal (For information on half-toning please see Chapter 8.1.) Vander Kam and Wong [15] give a closed-loop procedure to design a quantization table that is optimum for a given half-toning and scaling method chosen. The basic idea behind their algorithm is to code more coarsely frequency components that are corrupted by half-toning and to code more finely components that are left untouched by half-toning. Similarly, to take into account the effects of scaling, their design procedure assigns higher bit rate to the frequency components that correspond to a large gain in the scaling filter response and lower bit rate to components that are attenuated by the scaling filter.

5 Coefficient-to-Symbol Mapping and Coding

The quantizer makes the coding lossy, but it provides the major contribution in compression. However, the nature of the quantized DCT coefficients and the preponderance of zeros in the array, leads to further compression with the use of lossless coding. This requires that the quantized coefficients be mapped to symbols in such a way that the symbols lend themselves to effective coding. For this purpose, JPEG treats the DC coefficient and the set of AC coefficients in a different manner. Once the symbols are defined, they are represented with Huffman coding or arithmetic coding.

In defining symbols for coding, the DCT coefficients are scanned by traversing the quantized coefficient array in a zig-zag fashion shown in Fig. 8. The zig-zag scan processes the DCT coefficients in increasing order of spatial frequency. Recall that the quantized high-frequency coefficients are zero with high probability. Hence, scanning in this order leads to a sequence that contains a large number of trailing zero values and can be efficiently coded as shown in section 5.2.

The $[0, 0]$-th element or the quantized DC coefficient is separated from the remaining string of 63 AC coefficients, and symbols are defined next as shown in Fig. 9.

5.1 DC Coefficient Symbols

The DC coefficients in adjacent blocks are highly correlated. This fact is exploited to differentially code them. Let $qX_i[0, 0]$ and $qX_{i-1}[0, 0]$ denote the quantized DC coefficient in blocks

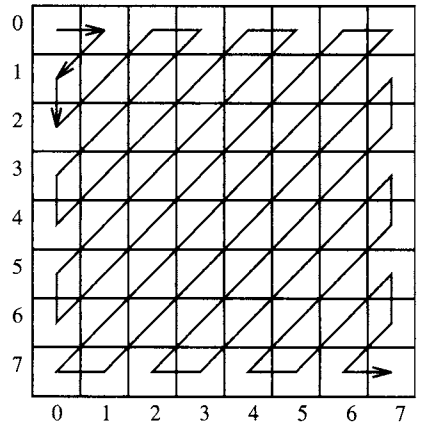

FIGURE 8 Zig-zag scan procedure.

(A) DC coding

Difference δ_i	[category, amplitude]	code
-2	[2,-2]	01101

(B) AC coding

Terminating value	run/ categ.	Code length	Code	Total Bits	Amplitude bits
41	0/6	7	1111000	13	010110
18	0/5	5	11010	10	10010
1	1/1	4	1100	5	1
2	0/2	2	01	4	10
-16	1/5	11	11111110110	16	01111
-5	0/3	3	100	6	010
2	0/2	2	01	4	10
-1	2/1	5	11100	6	0
-1	0/1	2	00	3	0
4	3/3	12	111111110101	15	100
-1	1/1	4	1100	5	1
1	5/1	7	1111010	8	1
-1	5/1	7	1111010	8	0
EOB	EOB	4	1010	4	-
Total bits for block		112			
Rate = 112/64 = 1.75 bits per pixel					

FIGURE 9 **A**: Coding of DC coefficient with value 57, assuming that the previous block has a DC coefficient of value 59. **B**: Coding of AC coefficients.

i and $i-1$. The difference $\delta_i = qX_i[0,0] - qX_{i-1}[0,0]$ is computed. Assuming a precision of 8 bits/pixel for each component, it follows that the largest DC coefficient value (with $q[0,0] = 1$) is less than 2048, so that values of δ_i are in the range [–2047, 2047]. If Huffman coding is used, these possible values would require a very large coding table. To limit the size of the coding table, the values in this range are grouped into 12 size categories, which are assigned labels 0 through 11. Category k contains 2^k elements $\{\pm 2^{k-1}, \ldots, \pm 2^k - 1\}$. The difference δ_i is mapped to a symbol described by a pair (category, amplitude). The 12 categories are Huffman coded. To distinguish values within the same category, extra k bits are used to represent the one of the possible 2^k "amplitudes" of symbols within category k. The amplitude of δ_i, $\{2^{k-1} \leq \delta_i \leq 2^k - 1\}$ is simply given by its their binary representation. On the other hand, the amplitude of δ_i, $\{-2^k - 1 \leq \delta_i \leq -2^{k-1}\}$ is given by the one's complement of the absolute value $|\delta_i|$, or simply by the binary representation of $\delta_i + 2^k - 1$.

5.2 Mapping AC Coefficient to Symbols

As observed before, most of the quantized AC coefficients are zero. The zig-zag–scanned string of 63 coefficients contains many consecutive occurrences or "runs of zeros," making the quantized AC coefficients suitable for run-length coding. The symbols in this case can be defined as [runs, nonzero terminating value], which can then be entropy coded. However, the number of possible values of AC coefficients is large as is evident from the definition of DCT. For 8-bit pixels, the allowed range of AC coefficient values is $[-1023, 1023]$. In view of the large coding tables this entails, a procedure similar to that discussed above for DC coefficients is used. Categories are defined for suitable grouped values that can terminate a run. Thus a run/category pair together with the amplitude within a category is used to define a symbol. The category definitions and amplitude bits generation use the same procedure as in DC difference coding. Thus, a 4-bit category value is concatenated with a 4-bit run length to get an 8-bit [run/category] symbol. This symbol is then encoded using either Huffman or arithmetic coding. There are two special cases that arise when coding the [run/category] symbol. First, since the run value is restricted to 15, the symbol (15/0) is used denote 15 zeroes followed by a zero. A number of such symbols can be cascaded to form specify larger runs. Second, if after a nonzero AC coefficient all the remaining coefficients are zero, then special symbol (0/0) denoting end of block (EOB) is encoded. Figure 9 continues our example and shows the sequence of symbols generated for coding the example quantized DCT block of Fig. 6.

5.3 Entropy Coding

The symbols defined for DC and AC coefficients can be entropy coded using mostly Huffman coding, or optionally and infrequently, arithmetic coding based on the probability estimates of the symbols. Huffman coding is a method of variable-length coding (VLC) in which shorter codewords are

assigned to the more frequently occurring symbols to achieve an average symbol codeword length that is as close to the symbol source entropy as possible. Huffman coding is optimal (meets the entropy bound) only when the symbol probabilities are integral powers of 1/2. The technique of *arithmetic coding* [18] provides a solution to attaining the theoretical bound of the source entropy. The baseline implementation of the JPEG standard uses Huffman coding only.

If Huffman coding is used, then Huffman tables, up to a maximum of eight in number, are specified in the bit stream. The tables constructed should not contain codewords that (a) are more that 16 bits long or (b) consist of all ones. Recommended tables are listed in Annex K of the standard. If these tables are applied to the output of the quantizer shown in the first two columns of Fig. 9, then the algorithm produces output bits shown in the following columns of the Figure. The procedures for specification and generation of the Huffman tables are identical to the ones used in the lossless standard explained in chapter 5.6 [32].

6 Image Data Format and Components

The JPEG standard is intended for the compression of both gray scale and color images. In a gray scale image there is a single "luminance" component. However a color image is represented with multiple components and the JPEG standard sets stipulations on the allowed number of components and data formats. The standard permits a maximum of 255 color components, which are rectangular arrays of pixel values represented with 8 to 12 bit precision. For each color component, the largest dimension supported in either the horizontal or the vertical direction is $2^{16} = 65{,}536$.

All color component arrays do not necessarily have the same dimensions. Assume that an image contains K color components denoted by C_n, $n = 1, 2, \ldots, K$. Let the horizontal and vertical dimensions of the n-th component array be equal to X_n and Y_n, respectively. Define dimensions X_{max}, Y_{max} and X_{min}, Y_{min} as:

$$X_{max} = max_{n=1}^{K}\{X_n\}, \quad Y_{max} = max_{n=1}^{K}\{Y_n\}$$

and

$$X_{min} = min_{n=1}^{K}\{X_n\}, \quad Y_{min} = min_{n=1}^{K}\{Y_n\}.$$

With each color component C_n, $n = 1, 2, \ldots, K$, one associates relative horizontal and vertical sampling factors, denoted by H_n and V_n, respectively, where

$$H_n = \frac{X_n}{X_{min}}, \quad V_n = \frac{Y_n}{Y_{min}}.$$

The standard restricts the possible values of H_n and V_n to the set of four integers 1, 2, 3, 4. The largest values of relative sampling factors are given by $H_{max} = \max\{H_n\}$ and $V_{max} = \max\{V_n\}$.

According to the JPEG file interchange format, the color information is specified by $[X_{max}, Y_{max}, H_n$ and V_n, $n = 1, 2, \ldots, K, H_{max}, V_{max}]$. The dimensions of the components are computed by the decoder as:

$$X_n = \left\lceil X_{max} \times \frac{H_n}{H_{max}} \right\rceil.$$

Example 1: Consider a raw image in a luminance-plus-chrominance representation consisting of $K = 3$ components, $C_1 = Y$, $C_2 = Cr$, and $C_3 = Cb$. Let the dimensions of the luminance matrix (Y) be $X_1 = 720$ and $Y_1 = 480$, and the dimensions of the two chrominance matrices (Cr and Cb) be $X_2 = X_3 = 360$ and $Y_2 = Y_3 = 240$. In this case $X_{max} = 720$ and $Y_{max} = 480$. The relative sampling factors are $H_1 = V_1 = 2$, and $H_2 = V_2 = H_3 = V_3 = 1$.

When images have multiple components, the standard describes formats for organizing the data for the purpose of storage. In storing components, the standard provides the option of using either interleaved or noninterleaved formats. Processing and storage efficiency is aided, however, by interleaving the components where the data are read in a single scan. Interleaving is performed by defining a *data unit* for lossy coding as a single block of 8×8 pixels in each color component. This definition can be used to partition the nth color component C_n, $n = 1, 2, \ldots, K$, into rectangular blocks, each of which contains $H_n \times V_n$ data units. A *minimum coded unit* (MCU) is then defined as the smallest interleaved collection of data units obtained by successively picking $H_n \times V_n$ data units from nth color component. Certain restrictions are imposed on the data to be stored in the interleaved format:

- The number of interleaved components should not exceed four, and
- An MCU should contain no more than ten data units, i.e.,

$$\sum_{n=1}^{K} H_n V_n \leq 10.$$

If the above restrictions are not met, the data are stored in a noninterleaved format, where each component is processed in successive scans.

Example 2: We consider the case of storage of the Y, Cr, Cb components in Ex. 1. The luminance component contains 90×60 data units and each of the two chrominance components contains 45×30 data units. Figure 10 shows both an noninterleaved and a interleaved arrangement of the data for $K = 3$ components, $C_1 = Y$, $C_2 = Cr$, and $C_3 = Cb$,

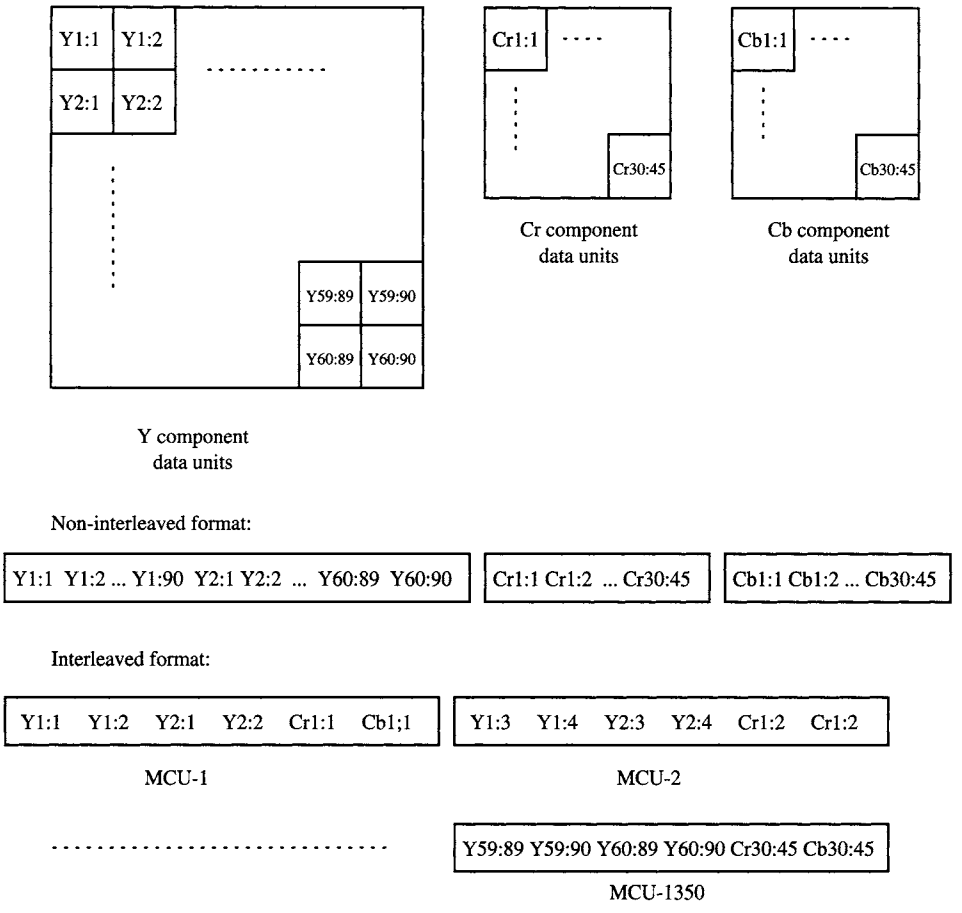

FIGURE 10 Organizations of the data units in the *Y*, *Cr*, and *Cb* components into noninterleaved and interleaved formats.

with $H_1 = V_1 = 2$, and $H_2 = V_2 = H_3 = V_3 = 1$. The MCU in the interleaved case contains six data units, consisting of $H_1 \times V_1 = 4$ data units of the Y component and $H_2 \times V_2 = H_3 \times V_3 = 1$ each of the Cr and Cb components.

7 Alternative 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 Fig. 11 we show how the different modes can be used. For example, the hierarchical mode could be used in conjunction with 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 chapter 5.6. 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.

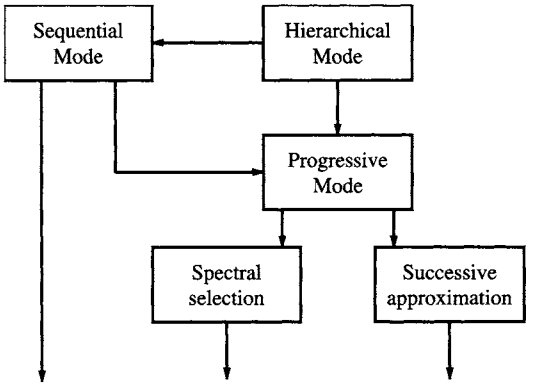

FIGURE 11 Joint Photographic Experts Group (JPEG) modes of operation.

7.1 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 generates additional bits that 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 that would benefit from progressive transmission, is provided by the Web, 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, telemedicine 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 of the example codes the first three AC coefficients in zig-zag order, the third scan encodes the next four AC coefficients, and the fourth (and 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. 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 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, that in the sequential mode, the EOB code indicates that the rest of 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 achieve further reduction in bit rate, JPEG defines an additional set of 15 symbols EOB_n each representing a run of 2^n EOB codes. After each EOB_i 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.

Despite the advantages of progressive transmission, there have not been many implementations of progressive JPEG codecs. There has been some interest in them recently due to the proliferation of images on the Web. It is expected that many more public domain progressive JPEG codecs will be available in the future.

7.2 Hierarchic Mode

The hierarchic mode defines another form of progressive transmission where the image is decomposed into a pyramidal structure of increasing resolution. The top-most layer in the pyramid represents the image at the lowest resolution and the base of the pyramid represents the image at full resolution. There is a doubling of resolutions both in the horizontal and vertical dimensions, between successive levels in the pyramid. Hierarchic coding is useful when an image could be

displayed at different resolutions in units such as handheld devices, computer monitors of varying resolutions, and high-resolution printers. In such a scenario, a multiresolution representation allows the transmission of the appropriate layer to each requesting device, thereby making full use of available bandwidth.

In the JPEG hierarchic 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 interpolating the low resolution image I_{i-1} from the layer immediately above. Then the difference between I_i' and I_i is encoded using modifications of the DCT-based modes or the lossless mode. If the lossless mode is used to code each refinement, the final reconstruction at the base layer is lossless. The interpolating filter used is a bilinear interpolating filter that is specified by the standard and cannot be specified by the user. Starting from the high-resolution image, successive low-resolution images are created essentially by decimating by two in each direction. The exact decimating filter to be used is not specified, but the standard cautions that the decimating filter used be consistent with the fixed interpolation filter. Note that the decoder does not need to know what decimating filter was used to decode a bit stream. Figure 12 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 used. For details, the user is referred to [11].

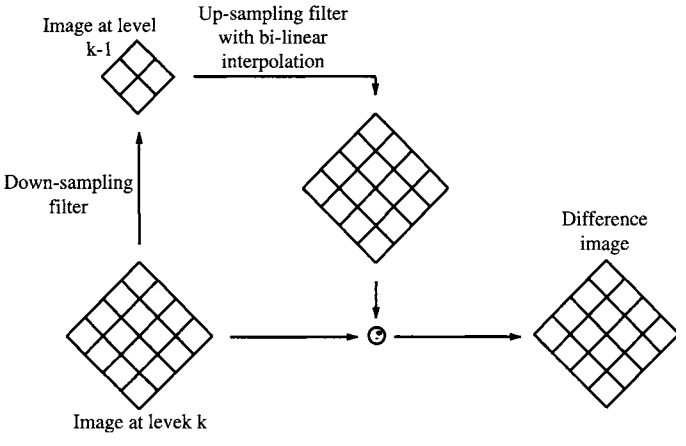

FIGURE 12 Joint Photographic Experts Group (JPEG) hierarchic mode.

8 JPEG Part 3

JPEG has made some recent extensions to the original standard described in [1]. These extensions are collectively known as JPEG Part 3. The most important elements of JPEG Part 3 are variable quantization and tiling, as described in greater detail in section 8.1.

8.1 Variable Quantization

One of the main limitations of the original JPEG standard was 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. We consider compression of a 24 bit/pixel color version of the *Lena* image. In Fig. 13 we show the reconstructed *Lena* image with different compression ratios. At 24-to-1 compression, we see little artifacts. But, as the compression ratio is increased to 96 to 1, noticeable artifacts begin to appear. Especially annoying is the "blocking artifact" in smooth regions of the image.

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. 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:

$$\hat{Y}[u,v] = \left\lfloor \frac{Y[u,v] \times 16}{Q[u,v] \times QScale} \right\rfloor,$$

where QScale is a parameter that can take on values from 1 to 112 (default 16). 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, 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 used.

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 proposed in the literature use the fact

FIGURE 13 *Lena* image at 24-to-1 (**top**) and 96-to-1 (**bottom**) compression ratios. (See color insert.)

that the HVS 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, a few simple features for each block are computed. These features are used to classify the block as either smooth, edge, texture, and so forth. On the basis of this classification, and a simple activity measure computed for the block, a QScale value is computed.

For example, Konstantinides and Tretter [8] 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:

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

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 discriminate smooth blocks where artifacts often appear first.

Algorithms for variable quantization that perform a more extensive classification have been proposed for video coding but nevertheless are also applicable to still image coding. One such technique has been proposed by Chun et al. [6] who classify blocks as being either smooth, edge, or texture based on several parameters defined 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 low-valued threshold. 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 thresholds. All blocks that fail to satisfy the smoothness and texture tests are classified as edge blocks.

8.2 Tiling

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 of an image region on a given screen size
- Fast access to image subregions
- Region-of-interest refinement
- Protection of large images from copying by giving access to only a part of it

As shown in Fig. 14, the different types of tiling allowed by JPEG are

- Simple tiling: This form of tiling is essentially used for dividing a large image into multiple subimages that are of the same size (except for edges) and are nonoverlapping. In this mode, all tiles are required to have the same sampling factors and components. Other parameters like quantization tables and Huffman tables are allowed to change from tile to tile.
- Composite tiling: This allows multiple resolutions on a single image display plane. Tiles can overlap within a plane.
- Pyramidal tiling: This is used for storing multiple resolutions of an image. Simple tiling as described above is used in each resolution. Tiles are stored in raster order, left to right, top to bottom, and low resolution to high resolution.

Another Part 3 extension is selective refinement. This feature permits a scan in a progressive mode, or a specific level of a hierarchic sequence, to cover only part of the total image area. Selective refinement could be useful, for example, in telemedicine applications where a radiologist could request refinements to specific areas of interest in the image.

9 The JPEG2000 Standard

The JPEG standard has proved to be a tremendous success over the past decade in many digital imaging applications. However, as the needs of multimedia and imaging applications evolved in areas such as medical imaging, reconnaissance, Internet, and mobile imaging, it became evident that the JPEG standard suffered from shortcomings in compression efficiency and progressive decoding. This led the JPEG committee to launch an effort in late 1996 and early 1997 to create a new image compression standard. The intent was to provide a method that would support a range of features in a single compressed bit stream for different types of still images such as bilevel, gray level, color, multicomponent—in particular multispectral—or other types of imagery. A call for technical contributions was issued in

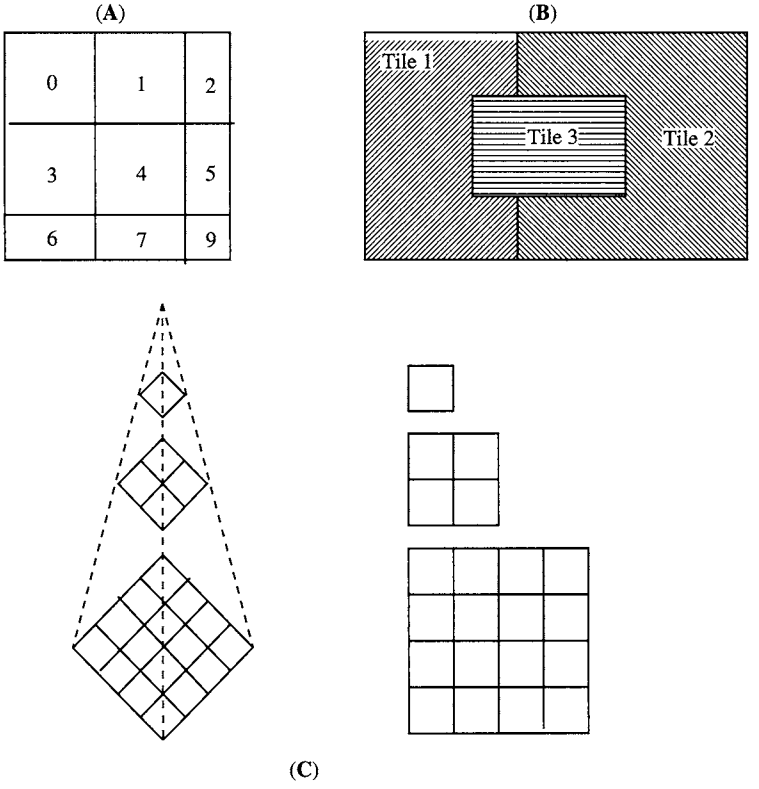

FIGURE 14 Different types of tilings allowed in Joint Photographic Experts Group (JPEG) Part 3. **A**: Simple. **B**: Composite. **C**: Pyramidal.

March 1997. Twenty four proposals were submitted for consideration by the committee in November 1997. Their evaluation led to the selection of a wavelet-based coding architecture as the backbone for the emerging coding system. The initial solution, inspired by the wavelet trellis-coded quantization (WTCQ) algorithm [36] based on wavelets and trellis-coded quantization (TCQ) [21, 31], has been refined via a series of core experiments over the ensuing 3 years. The initiative has resulted in the ISO 15444/ITU-T Recommendation T.8000 known as the JPEG2000 standard. It comprises six parts that are either complete or nearly complete at the time of writing this chapter, together with four new parts that are under development.

Part 1, in the spirit of the JPEG baseline system, specifies the core compression system together with a minimal file format [26]. JPEG2000 Part 1 addresses some limitations of existing standards by supporting the following features:

- Lossless and lossy compression of continuous tone and bilevel images with superior low bit-rate distortion and subjective performance.
- Progressive transmission and decoding based on resolution scalability by pixel accuracy [i.e., based on quality or signal-to-noise ratio (SNR) scalability]. The bytes extracted are identical to those that would be generated if the image had been encoded targeting the desired resolution or quality, the latter being directly available without the need for decoding and re-encoding.
- Random access to spatial regions (or regions of interest) as well as to components. Each region can be accessed at a variety of resolutions and qualities.
- Robustness to bit errors (e.g., for mobile image communication).
- Encoding capability for sequential scan, thereby avoiding the need to buffer the entire image to be encoded. This is especially useful when manipulating images of very large dimensions such as those encountered in reconnaissance (satellite and radar) images.

Some of the above features are supported to a limited extent in the JPEG standard. For instance, as described earlier, the JPEG standard has four modes of operation: sequential, progressive, hierarchic, and lossless. These modes use different techniques for encoding (e.g., the lossless compression mode relies on predictive coding, whereas the lossy compression modes rely on the DCT). One drawback is that if the JPEG Joint lossless mode is used, lossy decompression using the lossless encoded bitstream is not possible. One major advantage of JPEG2000 is that these four operation modes are integrated in it in a "compress once, decompress many" paradigm, with superior rate distortion and subjective performance over a large range of rate-distortion operating points.

Part 2 specifies extensions to the core compression system and a more complete file format [27]. These extensions address additional coding features such as generalized and variable quantization offsets, TCQ, visual masking, multiple component transformations, and so forth. In addition, it includes features such as image editing, such as cropping in the compressed domain or mirroring and flipping in partially compressed domain.

Parts 3, 4, and 5 provide a specification for motion JPEG 2000, conformance testing, and a description of a reference software implementation, respectively [28–30]. Four parts, numbered 8 through 11, are still under development at the time of writing. Part 8 deals with security aspects, Part 9 specifies an interactive protocol and application programming interface (API) for accessing JPEG2000 compressed images and files via a network, Part 10 deals with volumetric imaging, and Part 11 specifies the tools for wireless imaging.

The remainder of this chapter provides a brief overview of the JPEG2000 Part 1 and outlines the main extensions provided in Part 2. The JPEG2000 standard embeds efficient lossy, near-lossless, and lossless representations within the same stream. However, while some coding tools (e.g., color transformations, discrete wavelet transforms) can be used both for lossy and lossless coding, others can be used for lossy coding only. This led to the specification of two coding paths or options referred to as the reversible (embedding lossy and lossless representations) and irreversible (for lossy coding only) paths with common and path-specific building blocks. This chapter presents the main components of the two coding paths that can be used for lossy coding whereas chapter 5.6 [32] focuses on the components specific to JPEG2000 lossless coding. A detailed description of the JPEG2000 coding tools and system can be found in [39]. Tutorials and overviews are presented in [23, 35, 37].

10 The JPEG2000 Part 1: Coding Architecture

The coding architecture comprises two paths, the irreversible and the reversible paths as shown in Fig. 15. Both paths can be used for lossy coding by truncating the compressed codestream at the desired bit rate. The input image may comprise one or more (up to 16384) signed or unsigned components, to accommodate for various forms of imagery (e.g., multispectral imagery). The various components may have different bit depth, resolution, and signed specifications.

10.1 Preprocessing: Tiling, Level Offset, and Color Transforms

The first steps in both paths are optional and can be regarded as preprocessing steps. The image is first, optionally, partitioned into rectangular and nonoverlapping tiles of equal size. If the sample values are unsigned and represented with B bits, an offset of -2^{B-1} is added leading to a signed

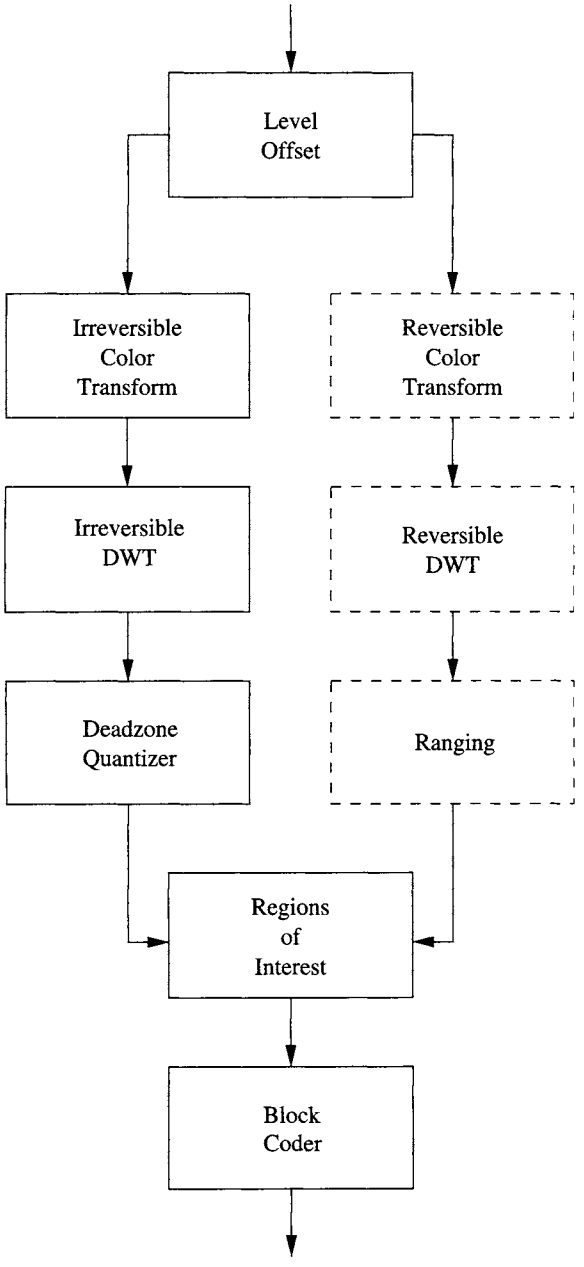

FIGURE 15 Main building blocks of the JPEG2000 coder. The path with boxes in dotted lines corresponds to the JPEG2000 lossless coding mode (see Chapter 5.6 [32]).

representation in the range $[-2^{B-1}, 2^{B-1}]$ (i.e., to a symmetric distribution about 0). The color component samples may be converted into luminance and color difference components via an irreversible color transform (ICT) or reversible color transform (RCT) in the irreversible or reversible paths, respectively. The ICT is identical to the conversion from *RGB* to YC_bC_r.

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

and can be used for lossy coding only. The RCT is a reversible integer-to-integer transform that approximates the ICT. This color transform is required for lossless coding and is described in chapter 5.6 [32] of this *Handbook*. The RCT can also be used for lossy coding, thereby allowing the embedding of both a lossy and lossless representation of the image in a single codestream.

10.2 Discrete Wavelet Transform

After the tiling, each tile component is decomposed with a forward discrete wavelet transform (DWT) into a set of $L = 2^l$ resolution levels using a dyadic decomposition. A detailed and complete presentation of the theory and implementation of filter banks and wavelets is beyond the scope of this chapter. The reader is referred to Chapter 4.2 in this *Handbook* and to [40] for additional insight on these issues.

The forward DWT is based on separable wavelet filters and can be irreversible or reversible. The transforms are then referred to, respectively, as reversible discrete wavelet transform (RDWT) and irreversible discrete wavelet transform (IDWT). As for the color transform, lossy coding can make use of both the IDWT and the RDWT. In the case of RDWT the codestream is truncated to reach a given bit rate. The use of the RDWT allows for both lossless and lossy compression to be embedded in a single compressed codestream. In contrast, lossless coding restricts us to the use of only RDWT.

The default RDWT is based on the spline 5/3 wavelet transform first introduced in [24]. The RDWT filtering kernel is presented in Chapter 5.6 [32] in this *Handbook*. The default irreversible transform, IDWT, is implemented with the Daubechies 9/7 wavelet kernel [19]. The coefficients of the analysis and synthesis filters are given in Table 1. Note however that, in JPEG2000 Part 2, other filtering kernels specified by the user can be used to decompose the image.

These filtering kernels are of odd length. Their implementation at the boundary of the image or subbands requires a symmetric signal extension. Two filtering modes are possible: convolution and lifting-based [33].

10.3 Quantization and Inverse Quantization

JPEG2000 adopts a scalar quantization strategy, similar to that in the JPEG baseline system. One notable difference is in the use of a central deadzone quantizer. A detailed description of the procedure can be found in [39]. This section provides only an outline of the algorithm. In Part 1, the subband samples are quantized with a deadzone scalar quantizer with a central interval that is twice the quantization step size. The quantization of $y_i(n)$ is given by:

$$\hat{y}_i(n) = sign(\ , y_i(n)) \left\lfloor \frac{|y_i(n)|}{\Delta_i} \right\rfloor \tag{4}$$

TABLE 1 Irreversible discrete wavelet transform analysis and synthesis filters coefficients

Index	Low-Pass Analysis Filter Coefficient	High-Pass Analysis Filter Coefficient	Low-Pass Synthesis Filter Coefficient	High-Pass Synthesis Filter Coefficient
0	0.602949018236360	1.115087052457000	1.115087052457000	0.602949018236360
+/–1	0.266864118442875	–0.591271763114250	0.591271763114250	–0.266864118442875
+/–2	–0.078223266528990	–0.057543526228500	–0.057543526228500	–0.078223266528990
+/–3	–0.016864118442875	0.091271763114250	–0.091271763114250	0.016864118442875
+/–4	+0.026748757410810			0.026748757410810

where Δ_i is the quantization step size in the subband i. The parameter Δ_i is chosen so that $\Delta_i = \Delta\sqrt{1/G_i}$ where G_i is the squared norm of the DWT synthesis basis vectors for subband i and where Δ is a parameter to be adjusted to meet given rate-distortion constraints. The step-size Δ_i is represented with two bytes, and consists of an 11-bit mantissa μ_i and a 5-bit exponent ϵ_i:

$$\Delta_i = 2^{R_i - \epsilon_i}\left(1 + \frac{\mu_i}{2^{11}}\right), \tag{5}$$

where R_i is the number of bits corresponding to the nominal dynamic range of the coefficients in subband i. In the reversible path, the step size Δ_i is set to 1 by choosing $\mu_i = 0$ and $\epsilon_i = R_i$.

The nominal dynamic range in subband i depends on the number of bits used to represent the original tile component and on the wavelet transform used.

The choice of a deadzone that is twice the quantization step size allows for an optimal bitstream embedded structure (i.e., for SNR scalability). The decoder can, by decoding up to any truncation point, reconstruct an image identical to what would have been obtained if encoded at the corresponding target bit rate. All image resolutions and qualities are directly available from a single compressed stream (also called codestream) without the need for decoding and reencoding the existing codestream. In Part 2, the size of the deadzone can have different values in the different subbands.

Two modes have been specified for signaling the quantization parameters: *expounded* and *derived*. In the expounded mode, the pair of values (ϵ_i, μ_i) for each subband are explicitly transmitted. In the derived mode, codestream markers quantization default (QCD) and quantization component (QCC) supply step size parameters only for the lowest frequency subband. The quantization parameters for other subbands i are then derived according to

$$(\epsilon_i, \mu_i) = (\epsilon_0 + l_i - L, \mu_0) \tag{6}$$

where L is the total number of wavelet decomposition levels and l_i is the number of levels required to generate the subband i.

The inverse quantization allows for a reconstruction bias from the quantizer midpoint for nonzero indices to accommodate skewed probability distributions of wavelet coefficients. The reconstructed values are thus computed as

$$\tilde{y}_i = \begin{cases} (\hat{y}_i + \gamma)\Delta_i 2^{M_i - N_i} & \text{if } \hat{y}_i > 0, \\ (\hat{y}_i - \gamma)\Delta_i 2^{M_i - N_i} & \text{if } \hat{y}_i < 0, \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

Here γ is a parameter that controls the reconstruction bias; a value of $\gamma = 0.5$ results in midpoint reconstruction. The term "M_i" denotes the maximum number of bits for a quantizer index in subband i. N_i represents the number of bits to be decoded in the case where the embedded bitstream is truncated prior to decoding.

10.4 Precincts and Codeblocks

Each subband, after quantization, is divided into nonoverlapping rectangular blocks, called codeblocks, of equal size. The dimensions of the codeblocks are powers of 2 (e.g., of size 16×16 or 32×32) and the total number of coefficients in a codeblock should not exceed 4096. The codeblocks formed by the quantizer indexes corresponding to the quantized wavelet coefficients constitute the input to the entropy coder. Collections of spatially consistent codeblocks taken from each subband at each resolution level are called precincts and will form a packet partition in the bit stream structure. The purpose of precincts is to enable spatially progressive bitstreams. This point is further elaborated in Section 10.6.

10.5 Entropy Coding

The JPEG2000 entropy coding technique is based on the EBCOT (Embedded Block Coding with Optimal Truncation) algorithm [38]. Each codeblock B_i is encoded separately, bit plane by bit plane, starting with the most significant bit plane (MSB) with a non-zero element and progressing towards the least significant bit-plane. The data in each bit-plane are scanned along the stripe pattern shown in Fig. 16 (with a stripe height of four samples) and encoded in three passes. Each pass collects contextual information that first helps decide which primitives to encode. The primitives are then provided to a context-dependent arithmetic coder.

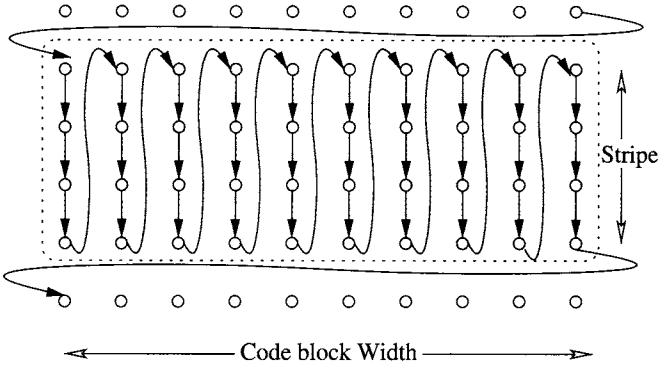

FIGURE 16 Stripe bit-plane scanning pattern.

The bit-plane encoding procedure is well suited for creating an embedded bit stream. Note that the approach does not exploit interscale dependencies. This potential loss in compression efficiency is compensated by beneficial features such as spatial random access, geometric manipulations in the compression domain, and error resilience.

10.5.1 Contexts Formation

Let $s_i[\mathbf{k}] = s_i[k_1, k_2]$ be the subband sample belonging to the block B_i at the horizontal and vertical positions k_1 and k_2. Let $\chi_i[\mathbf{k}] \in \{-1, 1\}$ denote the sign of $s_i[\mathbf{k}]$ and $v_i[\mathbf{k}] = |s_i[\mathbf{k}]|/\delta_{\beta_i}$ the amplitude of the quantized samples represented with M_i bits, where δ_{β_i} is the quantization step of the subband β_i containing the block B_i. Let $v_i^b[\mathbf{k}]$ be the b^{th} bit of the binary representation of $v_i[\mathbf{k}]$.

A sample $s_i[\mathbf{k}]$ is said to be *nonsignificant* ($\sigma[s_i[\mathrm{R}] = 0$) if the first nonzero bit $v_i^b[\mathbf{k}]$ of $v_i[\mathbf{k}]$ is yet to be encountered. The statistical dependencies between neighboring samples are captured via the formation of contexts which depend upon the *significance* state variable $\sigma[s_i[\mathrm{R}]$ associated with the eight-connect neighbors depicted in Fig. 17. These contexts are grouped in the following categories:

- h: Number of significant horizontal neighbors, $0 \leq h \leq 2$.
- v: Number of significant vertical neighbors, $0 \leq v \leq 2$.
- d: Number of significant diagonal neighbors, $0 \leq d \leq 4$

Neighbors that lie beyond the codeblock boundary are considered to be nonsignificant to avoid dependence between codeblocks.

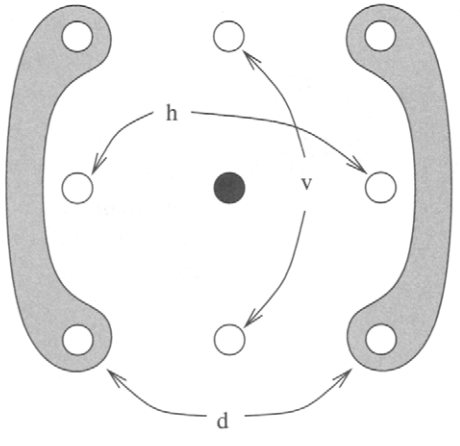

FIGURE 17 Neighbors involved in the contexts formation. (See color insert.)

10.5.2 Coding Primitives

Different subsets of the possible *significance* patterns form the contextual information (or state variables) that is used to decide upon the primitive to code as well as the probability model to use in arithmetic coding.

If the sample significance state variable is in the state *nonsignificant*, a combination of the zero coding (ZC) and run-length coding (RLC) primitives is used to encode whether the symbol is significant or not in the current bit plane. If the four samples in a column defined by the column-based stripe scanning pattern (see Fig. 16) have a zero significance state value ($\sigma[s_i[\mathrm{R}] = 0$), with zero-valued neighborhoods, then the *RLC* primitive is coded. Otherwise, the value of the sample $v_i^b[\mathbf{k}]$ in the current bit plane b is coded with the primitive *ZC*. In other words, *RLC* coding occurs when all four locations of a column in the scan pattern are nonsignificant and each location has only nonsignificant neighbors.

Once the first nonzero bit $v_i^b[\mathbf{k}]$ has been encoded, the coefficient becomes *significant* and its sign $\chi_i[\mathbf{k}]$ is encoded with the sign coding (SC) primitive. The binary-valued sign bit $\chi_i[\mathbf{k}]$ is encoded conditionally to five different context states depending upon the sign and significance of the immediate vertical and horizontal neighbors.

If a sample significance state variable is already *significant* (i.e., ($\sigma[s_i[\mathrm{R}] = 1$)), when scanned in the current bit plane, the magnitude refinement (MR) primitive encodes the bit value $v_i^b[\mathbf{k}]$. Three contexts are used depending on whether or not (a) the immediate horizontal and vertical neighbors are significant and (b) the *MR* primitive has already been applied to the sample in a previous bit plane.

10.5.3 Bit-Plane Encoding Passes

Briefly, the different passes proceed as follows. In a first *significance propagation* pass ($\mathcal{P}_i^{b,1}$), the insignificant coefficients that have the highest probability of becoming significant are encoded. A nonsignificant coefficient is considered to have a high probability of becoming significant if at least one of its eight-connect neighbors is significant. For each sample $s_i[\mathbf{k}]$ that is nonsignificant with a significant neighbor, the primitive *ZC* is encoded followed by the primitive *SC* if $v_i^b[\mathbf{k}] = \mathbf{1}$.

Once the first nonzero bit has been encoded, the coefficient becomes *significant* and its sign is encoded. All subsequent bits are called *refinement* bits. In the second pass, referred to as the *refinement* pass ($\mathcal{P}_i^{b,2}$), the significant coefficients are refined by their bit representation in the current bit plane. Following

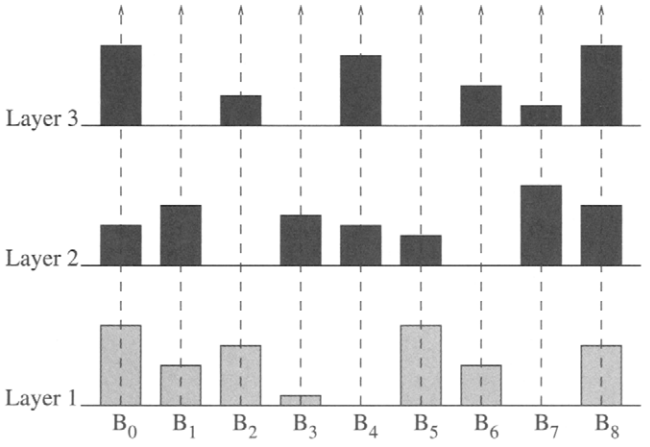

FIGURE 18 Codeblock contributions to layers.

the stripe-based scanning pattern, the primitive *MR* is encoded for each significant coefficient for which no information has been encoded yet in the current bit-plane b.

In a final *normalization* or *clean-up* pass ($\mathcal{P}_3^b$), all the remaining coefficients in the bit plane (i.e., the nonsignificant samples for which no information has yet been coded) are encoded with the primitives *ZC*, *RLC* and, if necessary, *SC*. The *clean-up* pass $\mathcal{P}_3^b$ corresponds to the encoding of all the b bit-plane samples.

The encoding in three passes $\mathcal{P}_1^b, \mathcal{P}_2^b, \mathcal{P}_3^b$, leads to the creation of distinct subsets in the bit stream (Fig. 18). This structure in *partial bit planes* allows a fine granular bitstream representation providing a large number of rate-distortion truncation points. The standard allows the placement of bitstream truncation points at the end of each coding pass (this point is revisited in the sequel). The bitstream can thus be organized in such a way that the subset leading to a larger reduction in distortion is transmitted first.

10.5.4 Arithmetic Coding

Entropy coding is done by means of an arithmetic coder that encodes binary symbols (the primitives) using adaptive probability models conditioned by the corresponding contextual information. A reduced number of contexts, up to a maximum of nine, is used for each primitive. The corresponding probabilities are initialized at the beginning of each codeblock and then updated using a state automaton. The reduced number of contexts allows for rapid probability adaptation. In a default operation mode, the encoding process starts at the beginning of each codeblock and terminates at the end of each codeblock. However, it is also possible to start and terminate the encoding process at the beginning and the end, respectively, of a *partial* bit plane in a codeblock. This allows increased error resilience of the codestream.

An arithmetic coder proceeds with recursive probability interval subdivisions. The arithmetic coding principles are described in chapter 5.6 [32]. In brief, the interval $[0, 1]$ is partitioned into two cells representing the binary symbols of the alphabet. The size of each cell is given by the stationary probability of the corresponding symbol. The partition, and hence the bounds of the different segments, of the unit interval is given by the cumulative stationary probability of the alphabet symbols. The interval corresponding to the first symbol to be encoded is chosen. It becomes the current interval that is again partitioned into different segments. The subinterval associated with the more probable symbol (MPS) is ordered ahead of the subinterval corresponding to the less probable symbol (LPS). The symbols are thus often recognized as *MPS* and *LPS* rather than as 0 or 1. The bounds of the different segments are hence driven by the statistical model of the source. The codestream associated with the sequence of coded symbols points to the lower bound of the final subinterval. The decoding of the sequence is performed by reproducing the coder behavior to determine the sequence of subintervals pointed by the codestream.

Practical implementations use fixed precision integer arithmetic with integer representations of fractional values. This potentially forces an approximation of the symbols probabilities leading to some coding sub-optimality. The corresponding states of the encoder (interval values that cannot be reached by the encoder) are used to represent markers that contribute to improving the error resilience of the codestream [39]. One of the early practical implementations of arithmetic coding is known as the Q-coder [34]. The JPEG2000 standard has adopted a modified version of the Q-coder, called the MQ-coder, introduced in the JBIG2 standard [35] and available on a license and royalty-free basis. The various versions of arithmetic coders inspired from the Q-coder often differ by their stuffing procedure and the way they handle the carry-over.

To reduce the number of symbols to encode, the standard specifies an option that allows the by-passing of some coding passes. Once the fourth bit plane has been coded, the data corresponding to the first and second passes are included as raw data without being arithmetically encoded. Only the third pass is encoded. This coding option is referred to as the *lazy* coding mode.

10.6 Bitstream Organization

The compressed data resulting from the different coding passes can be arranged in different configurations to accommodate a rich set of progression orders that are dictated by the application needs of random access and scalability. This flexible progression order is enabled by essentially four bitstream structuring components: codeblock, precinct, packet, and layer.

10.6.1 Packets and Layers

The bitstream is organized as a succession of layers, each one being formed by a collection of packets. The layer gathers sets

of compressed *partial* bit plane data from all the codeblocks of the different subbands and components of a tile. A packet is formed by an aggregation of compressed *partial* bit planes of a set of codeblocks that correspond to one spatial location at one resolution level and that define a precinct. The number of bit-plane coding passes contained in a packet varies for different codeblocks. Each packet starts with a header that contains information about the number of coding passes required for each codeblock assigned to the packet. The codeblock compressed data are distributed across the different layers in the codestream. Each layer contains the additional contributions from each codeblock. The number of coding passes for a given codeblock that are included in a layer is determined by rate-distortion optimization and it defines truncation points in the codestream [38].

Notions of precincts, codeblocks, packets and layers are well suited to allowing the encoder to arrange the bitstream in an arbitrary progression manner (i.e., to accommodate the different modes of scalability that are desired). Four types of progression, namely resolution, quality, spatial, and component, can be achieved by an appropriate ordering of the packets in the bitstream. For instance, layers and packets are key components for allowing quality scalability (i.e., packets containing less significant bits can be discarded to achieve lower bit rates and higher distortion). This flexible bitstream structuring gives application developers a high degree of freedom. For example, images can be transmitted over a network at arbitrary bit rates by using a layer-progressive order; lower resolutions, corresponding to low-frequency subbands, can be sent first for image previewing; and spatial browsing of large images is also possible through appropriate tile and/or partition selection. All these operations do not require any reencoding but only byte-wise copy operations. Additional information on the different modes of scalability is provided in the standard.

10.6.2 Truncation Points Rate-Distortion Optimization

The problem now is to find the packet length for all codeblocks (i.e., define truncation points, that will minimize the overall distortion). The recommended method to solve this problem, which is not part of the standard, makes use of a rate-distortion (R-D) optimization procedure. Under certain assumptions about the quantization noise, the distortion is additive across codeblocks. The overall distortion can thus be written as $D = \sum_i D_i^{n_i}$. There is thus a need to search for the packet lengths n_i so that the distortion is minimized under the constraint of an overall bit rate $R = \sum_i R_i^{n_i} \leq R^{max}$. The distortion measure $D_i^{n_i}$ is defined as the mean-square error weighted by the square of the L_2 norm of the wavelet basis functions used for the subband i to which the codeblock B_i belongs. This optimization problem is solved using a Lagrangian formulation.

10.7 Additional Features

10.7.1 Region of Interest Coding

The JPEG2000 standard has a provision for defining the so-called *regions of interest* (ROI) in an image. The objective is to encode the ROIs with a higher quality and possibly to transmit them first in the bitstream so that they can be rendered first in a progressive decoding scenario. To allow for ROI coding, an ROI mask must first be derived. A mask is a map of the ROI in the image domain with nonzero values inside the ROI and zero values outside. The mask identifies the set of pixels (or the corresponding wavelet coefficients) that should be reconstructed with higher fidelity. ROI coding thus consists of encoding the quantized wavelet coefficients corresponding to the ROI with a higher precision. The ROI coding approach in JPEG2000 Part 1 is based on the MaxShift method [22], which is an extension of the ROI scaling–based method introduced in [20]. The ROI scaling method consists of scaling up the coefficients belonging to the ROI, or scaling down the coefficients corresponding to non-ROI regions in the image. The goal of the scaling operation is to place the bits of the ROI in higher bit planes than the bits associated with the non-ROI regions as shown in Fig. 19. Thus, the ROI will be decoded before the rest of the image and, if the bitstream is truncated, the ROI will be of higher quality. The ROI scaling method described in [20] requires the coding and transmission of the ROI shape information to the decoder. To minimize the decoder complexity, the MAXSHIFT method adopted by JPEG2000 Part 1 shifts down all the coefficients not belonging to the ROI by a certain number s of bits chosen so that 2^s is

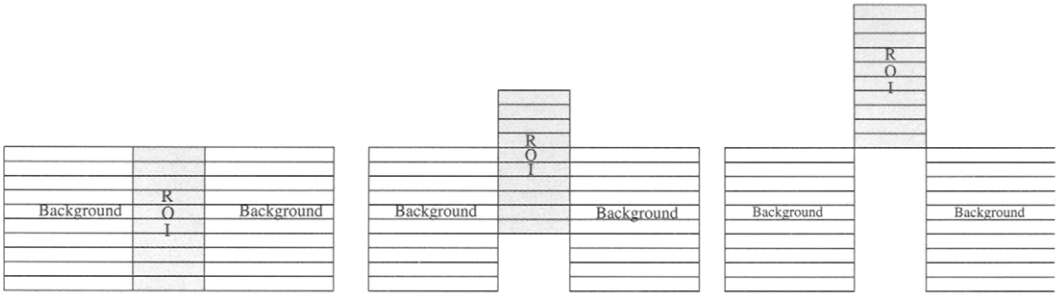

FIGURE 19 No region-of-interest (ROI) (**left**); coding scaling method (**middle**), and MaxShift method for ROI coding (**right**).

larger than the largest non-ROI coefficients. This ensures that the minimum value contained in the ROI is higher than the maximum value of the non-ROI area. The compressed data associated with the ROI will then be placed first in the bitstream. With this approach the decoder does not need to generate the ROI mask. All the coefficients lower than the scaling value belong to the non-ROI region. Therefore the ROI shape information does not need to be encoded and transmitted. The drawback of this reduced complexity is that the ROI cannot be encoded with multiple quality differentials with respect to the non-ROI area.

10.7.2 File Format

Part 1 of the JPEG2000 standard also defines an optional file format referred to as JP2. It defines a set of data structures used to store information that may be required to render and display the image such as the colorspace (with two methods of color specification), the resolution of the image, the bit-depth of the components, and the type and ordering of the components. The JP2 file format also defines two mechanisms for embedding application-specific data or metadata using either a universal unique identifier (UUID) or XML [41].

10.7.3 Error Resilience

Arithmetic coding is very sensitive to transmission noise: when some bits are altered by the channel, synchronization losses can occur at the receiver leading to error propagation that results in dramatic symbol error rates. JPEG2000 Part 1 provides several options to improve the error resilience of the codestream. First, the independent coding of the code blocks limits error propagation across code block boundaries. Certain coding options such as terminating the arithmetic coding at the end of each coding pass and reinitializing the contextual information at the beginning of the next coding pass further confine error propagation within a partial bit plane of a code block. The optional *lazy* coding mode, that bypasses arithmetic coding for some passes, can also help to protect against error propagation. In addition, at the end of each cleanup pass, segmentation symbols are added in the codestream. These markers can be exploited for error detection. If the segmentation symbol is not decoded properly, the data in the corresponding bit plane and of the subsequent bit planes in the code block should be discarded. Finally, resynchronization markers, including the numbering of packets, are also inserted in front of each packet in a tile.

11 JPEG2000 Performance and Extensions

The performance of JPEG2000 when compared with the JPEG baseline algorithm is briefly discussed in this section. The extensions included in Part 2 of the JPEG2000 standard are also listed.

11.1 Comparison of Performance

The efficiency of the JPEG2000 lossy coding algorithm in comparison with the JPEG baseline compression standard has been extensively studied and key results are summarized in [4, 5, 9]. The superior rate-distortion and error-resilience performance, together with features such as progressive coding by resolution, scalability, or ROI, clearly demonstrate the advantages of JPEG2000 over the baseline JPEG (with optimum Huffman codes). For coding common test images such as *Foreman* and *Lena* in the range of 0.125 to 1.25 bits/pixel, an improvement in peak signal-to-noise ratio (PSNR) for JPEG2000 is consistently demonstrated at each compression ratio. For example, for the *Foreman* image, an improvement of 1.5 to 4 dB is observed as the bits per pixel are reduced from 1.2 to 0.12 [4].

11.2 Part 2: Extensions

Most of the technologies that have not been included in Part 1 due to their complexity or because of intellectual property rights (IPR) issues have been included in Part 2 [27]. These extensions concern the use of:

- Different offset values for the different image components
- Different deadzone sizes for the different subbands
- Trellis-coded quantization (TCQ) [31]
- Visual masking based on the application of a nonlinearity to the wavelet coefficients [42, 43]
- Arbitrary wavelet decomposition for each tile component
- Arbitrary wavelet filters
- Single sample tile overlap
- Arbitrary scaling of the ROI coefficients with the necessity to code and transmit the ROI mask to the decoder
- Nonlinear transformations of component samples and transformations to de-correlate multiple component data
- Extensions to the JP2 file format

12 Additional Information

Some sources and links for further information on the standards are provided here.

12.1 Useful Information and Links for the JPEG Standard

A key source of information on the JPEG compression standard is the book by Pennebaker and Mitchell [11]. This book also contains the entire text of the official committee draft international standard ISO DIS 10918-1 and ISO DIS

10918-2. The official standards document [1] contains information on JPEG Part 3.

The JPEG committee maintains an official Web site 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. The JPEG FAQ is located at http://www.faqs.org/faqs/jpeg-faq/part1/preamble.html.

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/graphics/jpegsr6b.zip.

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 and has been used successfully on many machines ranging from personal computers to supercomputers. The IJG code is free for noncommercial and commercial use; only an acknowledgment in your documentation is required to use it in a product. A different free JPEG implementation, written by the portable video research group (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.

12.2 Useful Information and Links for the JPEG2000 Standard

A useful source of information on the JPEG2000 compression standard is the book by Taubman and Marcellin [39]. Further information on the different parts of the JPEG2000 standard can be found on the JPEG Web site http://www.jpeg.org/jpeg2000.html. This Web site provide links to sites from which various official standard and other documents can be downloaded. It also provides links to sites from which software implementations of the standard can be downloaded. Some software implementations are available at the following addresses:

- JJ2000 software that can be accessed at http://jpeg2000.epfl.ch. The JJ2000 software is a Java implementation of JPEG2000 Part 1.
- Kakadu software that can be accessed at http://www.ee.unsw.edu.au/taubman/kakadu. The Kakadu software is a C++ implementation of JPEG2000 Part 1. The Kakadu software is provided with the book [39].
- Jasper software that can be accessed at http://www.ece.ubc.ca/mdadams/jasper/. Jasper is a C implementation of JPEG2000 that is free for commercial use.

References

[1] ISO/IEC JTC 1/SC 29/WG 1 N 993. Information technology—Digital compression and coding of continuous-tone still images. Recommendation T.84 *ISO/IEC CD 10918-3*. Nov 1994.

[2] N. Ahmed, T. Natrajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comp.* 23, 90–93, (1974).

[3] A. J. Ahumada and H. A. Peterson, "Luminance model based DCT quantization for color image compression," *Human Vision, Visula Processing, and Digital Display, III* SPIE, pp. 365–374, 1992.

[4] D. Chai and A. Bouzerdoum, "JPEG2000 image compression: an overview," Australian and New Zealand Intelligent Information Systems Conference (ANZIIS'2001), Perth, Australia, pp. 237–241 Nov. 2001.

[5] C. Christopoulos, IEEE, A. Skodras, and T. Ebrahimi, "The JPEG still image coding system: an overview," *IEEE Trans. Cons. Elect.* 46, 1103–1127 (2000).

[6] K. W. Chun, K. W. Lim, H. D. Cho, and J. B. Ra, "An adaptive perceptual quantization algorithm for video coding," *IEEE Trans. Cons. Elect.* 39, 555–558 (1993).

[7] N. Jayant, R. Safranek, and J. Johnston, "Signal compression based on models of human perception," in *Proceedings of the IEEE*, 83, 1385–1422 (1993).

[8] K. Konstantinides and D. Tretter, "A method for variable quantization in JPEG for improved text quality in compound documents," in *Proceedings of the IEEE International Conference on Image Processing* (Chicago, IL, October 1998).

[9] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek, "An overview of JPEG2000," *Proc. of IEEE Data Compression Conference* (2000), pp. 523–541.

[10] W. B. Pennebaker and J. L. Mitchell, "An overview of the basic principles of the Q-Coder adaptive binary arithmetic coder," *IBM Journal of Research and Development*, 32(6), 717–726, 1988. Technical Report JPEG-18, ISO/IEC/JTC1/SC2/WG8, International Standards Organization, 1988. Working group document.

[11] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard* (Van Rostrand, Reinhold, 1993).

[12] V. Ratnakar and M. Livny, "RD-OPT: An efficient algorithm for optimizing DCT quantization tables," in *IEEE Proceedings of the Data Compression Conference (DCC)* (Snowbird, Utah, 1995), pp. 332–341.

[13] K. R. Rao and P. Yip, *Discrete Cosine Transform—Algorithms, Advantages, Applications* (Academic Press, San Diego, California, 1990).

[14] B. J. Sullivan, R. Ansari, M. L. Giger, and H. MacMohan, "Relative effects of resolution and quantization on the quality of compressed medical images," in *Proc. IEEE International Conference on Image Processing* (Austin, TX, 1994), pp. 987–991.

[15] R. VanderKam and P. Wong, "Customized JPEG compression for grayscale printing," in *Proceedings of the Data Compression Conference (DCC)* (Snowbird, Utah, 1994), pp. 156–165.

[16] G. K. Wallace, "The JPEG still picture compression standard," *Communic. ACM*, 34, 31–44 (1991).

[17] A. B. Watson, "Visually optimal DCT quantization matrices for individual images," in *Proceedings of the IEEE Data Compression Conference (DCC)* (Snowbird, Utah, 1993), pp. 178–187.

[18] I. H.Witten, R. M. Neal and J. G. Cleary, "Arithmetic coding for data compression," *Communic. ACM*, 30, 520–540 (1987).

[19] A. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, "Image coding using the wavelet transform," *IEEE Trans. Image Process.* 205–220 (1992).

[20] E. Atsumi and N. Farvardin, " Lossy/lossless region-of-interest image coding based on set partitioning in hierarchical trees," in *Proc. IEEE Intl. Conf. on Image Processing*, Chicago, IL (1998), pp. 87–91.

[21] A. Bilgin, P. J. Sementilli, and M. W. Marcellin, " Progressive image coding using trellis coded quantization," *IEEE Trans. Image Process.* 8, 1638–1643 (1999).

[22] C. Christopoulos, J. Askelof, and M. Larsson, "Efficient methods for encoding regions of interest in the upcoming JPEG2000 still image coding standard," *IEEE Signal Process. Letters* (Sept. 2000).

[23] C. Christopoulos, A. Skodras, and T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *IEEE Trans. Consum. Elect.* 46, 1103–1127 (2000).

[24] D. Le Gall and A. Tabatabai, "Subband coding of digital images using symmetric short kernel filters and arithmetic coding techniques," in *Proc. Intl. Conf. on Acoustics, Speech and Signal Processing, ICASSP'88* (1988), pp. 761–764.

[25] ISO/IEC International standard 14492 and ITU recommendation T.88. *JBIG2 Bi-Level Image Compression Standard* (2000).

[26] ISO/IEC International standard 15444-1 and ITU recommendation T.800. *Information Technology—JPEG2000 Image Coding System* (2000).

[27] ISO/IEC International standard 15444-2 and ITU recommendation T.801. *Information Technology—JPEG2000 Image Coding System: Part 2, Extensions* (2001).

[28] ISO/IEC International standard 15444-3 and ITU recommendation T.802. *Information Technology—JPEG2000 Image Coding System: Part 3, Motion JPEG2000* (2001).

[29] ISO/IEC International standard 15444-4 and ITU recommendation T.803. *Information Technology—JPEG2000 Image Coding System: Part 4, Compliance Testing* (2001).

[30] ISO/IEC International standard 15444-5 and ITU recommendation T.804. *Information Technology—JPEG2000 Image Coding System: Part 5, Reference Software* (2001).

[31] M. W. Marcellin and T. R. Fisher, "Trellis coded quantization of memoryless and gauss-markov sources," *IEEE Trans. Communic.* 38, 82–93 (1990).

[32] N. Memon and R. Ansari. The JPEG Lossless Compression Standards. Chapter 5.6 in this Handbook.

[33] P. Moulin. Multiscale Image Decomposition and Wavelets, Chapter 4.2 in this Handbook.

[34] W. B. Pennebaker, J. L. Mitchell, G. G. Langdon, and R. B. Arps, "An overview of the basic principles of the q-coder adaptive binary arithmetic coder," *IBM J. Res. Develop.* 32, 717–726 (1988).

[35] M. Rabbani and R. Joshi, "An overview of the JPEG2000 still image compression standard," *Elsevier J. Sign. Process.* 17, 3–48 (2002).

[36] P.J. Sementilli, A. Bilgin, J. H. Kasner, and M. W. Marcellin, "Wavelet TCQ: submission to JPEG2000," in *Proc. SPIE, Applications of Digital Processing* (1998), pp. 2–12.

[37] A. Skodras, C. Christopoulos, and T. Ebrahimi. The jpeg2000 still image compression standard. *IEEE Signal Processing Magazine*, pp. 36–58, Sept. 2001.

[38] D. Taubman, "High performance scalable image compression with ebcot," *IEEE Trans. Image Process.* 9, 1158–1170 (1999).

[39] D. Taubman and M. W. Marcellin, *JPEG2000: Image Compression Fundamentals: Standards and Practice* (Kluwer Academic Publishers, 2002).

[40] M. Vetterli and J. Kovacevic. *Wavelet and subband coding* (Prentice-Hall, Englewood Cliffs, NJ, 1995).

[41] World Wide Web Consortium (W3C), "Extensible Markup Language (XML) 1.0," (Third Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, Eds., http://www.w3.org/TR/REC-xml (2004).

[42] W. Zeng, S. Daly, and S. Lei, "Point-wise extended visual masking for JPEG2000 image compression," *Proc. IEEE International Conference on Image Processing*, Vol. 1, (Vancouver, BC, Canada, 2000), 657–660.

[43] W. Zeng, S. Daly, and S. Lei, "Visual optimization tools in JPEG2000," in *Proc. IEEE Intl. Conf. on Image Processing*, 37–40 (2000).

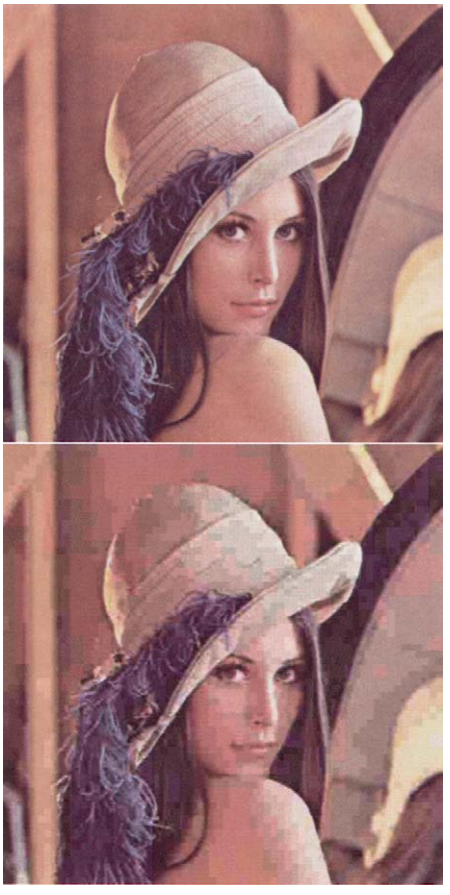

FIGURE 5.5.13 *Lena* image at 24-to-1 (top) and 96-to-1 (bottom) compression ratios.

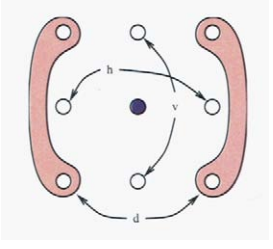

FIGURE 5.5.17 Neighbors involved in the contexts formation.

FIGURE 6.2.20 (a) Frame 141 in *Foreman* with OBMC, IBLOCK and color HVSBM at rate 512 Kbps @ 30 fps. (b) Frame 141 in *Foreman* without OBMC and IBLOCK at rate 512 Kbps @ 30 fps.

FIGURE 6.2.21 (a) Frame 43 in *Foreman* with OBMC, IBLOCK and color HVSBM (corresponding to original frame 86) at rate 256 Kbps @ 15 fps, half the original frame rate. (b) Frame 43 in *Foreman* without OBMC and IBLOCK, but with old YUV HVSBM (corresponding to original frame 86) at rate 256 Kbps @ 15 fps, half the original frame rate.