Real-Time Imaging **1**, 49–67 (1995)

A Survey of Multimedia Compression Techniques and Standards. Part I: JPEG Standard

This paper is the first part of a comprehensive survey of compression techniques and standards for multimedia applications. It covers the JPEG compression algorithm which is primarily used for full-color still image applications. The paper describes all the components of the JPEG algorithm including discrete cosine transform, quantization, and entropy encoding. It also describes both encoder and decoder architectures. The main emphasis is given to the sequential mode of operation which is the most typical use of JPEG compression; however, the other three modes of operation, progressive, lossless, and hierarchical JPEG, are described as well. A number of experimental data for both grayscale and color image compression is provided in the paper.

Borko Furht

Department of Computer Science & Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA
borko@cse.fau.edu

Introduction

Multimedia computing has emerged in the last few years as a major area of research. Multimedia computer systems have opened the wide range of potential applications by combining a variety of information sources, such as voice, graphics, animation, images, audio, and full-motion video. Looking at the big picture, multimedia can be viewed as the combination of three industries: computer, communication, and broadcasting industries.

Research and development efforts in multimedia computing have been divided into two groups. As the first group of research, much effort has been centered on the stand-alone multimedia workstation and associated software systems and tools, such as music composition, computer-aided learning, and interactive video. However, the combination of multimedia computing with distributed systems offers even greater potential. Potential new applications based on distributed multimedia systems include multimedia information systems, collaboration and conferencing systems, on-demand multimedia services, and distance learning.

The fundamental characteristics of multimedia systems is that they incorporate continuous media, such as voice, video, and animated graphics. This implies the need for multimedia systems to handle data with strict timing requirements and at high rate. The use of continuous media in distributed systems implies the need for multimedia systems to handle data with strict timing requirements and at a high rate. The use of continuous media in distributed systems implies the need for continuous data transfer over relatively long periods of time (e.g. playout of video stream from a remote camera). Additional important fundamental issues are: media synchronization, very large storage required, and need for special indexing and retrieval techniques, tuned to multimedia data types(1).

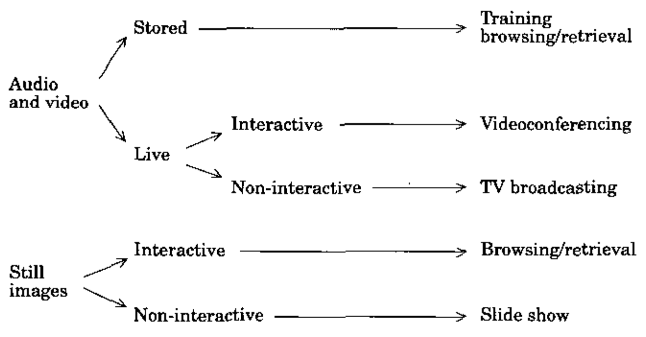


Figure 1. Various modes of operations on multimedia data

Figure 1 illustrates basic principles in dealing with continuous media (audio and video), and still images, and several examples of operations on these media. Audio and video information can be either stored and then used in an application, such as training, or can be transmitted live in real-time. Live audio and video can be used interactively, such as in multimedia conferencing, or non-interactively, in TV broadcast applications. Similarly, still images, which are stored, can be used in an interactive mode (using operations such as browsing and retrieval), or in non-interactive mode (slide show).

It is interesting to note that multimedia stresses all the components of a computer system. At the processor level, multimedia requires very high processing power in order to implement software codes, and multimedia file systems and corresponding file formats. At the architecture level, new solutions are needed to provide high bus bandwidth, and efficient I/O. At the operating systems level, there is a need for high-performance real-time multimedia OS, which will support new data types, real-time scheduling, and fast interrupt processing. Storage and memory requirements include very high capacity, fast access times and high transfer rates. At the network level, new networks and protocols are needed to provide high bandwidth, and low latency and jitter required for multimedia. At the software tool level, there is a need for new object-oriented, user-friendly multimedia development tools, and tools for retrieval and data management, important especially for large, heterogeneous, networked and distributed multimedia systems.

Researchers in multimedia fields are currently working in existing computer areas with the goals to transform existing technologies or develop new technologies, which will be suitable for multimedia. These research areas practically include most of the areas in the computer field e.g.: fast processors, high-speed networks, large-capacity storage devices, new algorithms and data structures, video and audio compression algorithms, graphics systems, human-computer interaction, real-time operating systems, object-oriented programming, information storage and retrieval, hypertext and hypermedia, languages for scripting, parallel processing methods, and complex architectures for distributed systems.

In this paper, we present the JPEG (Joint Photographic Expert Group) compression standard which is primarily intended for full-color still image applications; however, it can be also used for some real-time, full-motion video applications (Motion JPEG). In the second part of the paper, which will appear in a future issue of *Real-Time Imaging*, we describe the other two compression standards–px64 (or H.261) standard for video-based communications and MPEG standard for motion-intensive applications.

Storage Requirements for Multimedia Applications

Audio, image, and video signals require a vast amount of data for their representation. Table 1 illustrates the mass storage requirements for various media types, such as text, image, audio, animation, and video.

There are three main reasons why present multimedia systems require data to be compressed. These reasons are related to: (a) large storage requirements of multimedia data, (b) relatively slow storage devices which does not allow playing multimedia data (specifically video) in real-time, and (c) the present network's bandwidth, which do not allow real-time video data transmission.

As an example, a typical multimedia application may require the storage of more than 30 min of video, 2,000 images, and 40 min of stereo sound on each laser disc side. This application would require about 50 GB storage for video, 15 GB storage for images, and 0·4 GB storage for audio, that gives a total of 65·4 GB of storage.

In another example, assuming color video frames with 620 × 560 pixels, and 24 bits per pixels, it would be necessary to save about 1 Mbytes per frame. For a motion video requiring 30 frames/s, it gives a total of 30 Mbytes for 1 s of motion video. Even if there is enough storage available, we won't be able to playback the video

Table 1. Storage requirements for various media types

	Text	Image	Audio	Animation	Video
Object type	-ASCII -EBCDIS	-Bitmapped graphics -Still photos -Faxes	Noncoded stream of digitized audio or voice	Synched image and audio stream at 15–19 frames/s	TV analog or digital image with synched streams at 24–30 frames/s
Size and bandwidth	2KB per page	-Simple: 64KB per image -Detailed (color) 7·5 MB per image	Voice/Phone 8 KHz/8 bits (mono) 6–44 KB/s AUDIO CD 44·1 KHz/ 16 bit/stereo 176 KB/s	2·5 MB/s for 320 × 640 × 16 pixels/frame (16 bit color) 16 frames/s	27·7 MB/s for 640 × 480 × 24 pixels per frame (24-bit color) 30 frames/s

in real-time due to insufficient bit rate storage devices. According to the previous calculation, the required speed of a storage device should be 30 Mbytes/s; however today's technology provides about 300 Kbytes/s transfer rate of CD ROMs. Therefore, at the present state of technology of storage devices the only solution is to compress the data before the storage and decompress it before playback.

Modern image and video compression techniques offer a solution to this problem, which reduces these tremendous storage requirements. Advanced compression techniques can compress a typical image ranging from 10:1 to 50:1. Very high compression ratios of up to 2000:1 can be achieved in compressing of video signals. Figure 2 illustrates storage requirements for a multimedia application consisting of various media types, assuming that the image is compressed by ratio 15:1, and video by factors 30:1 and 200:1. The total storage requirement for this storage-intensive application becomes little over 2 GB, which is feasible.

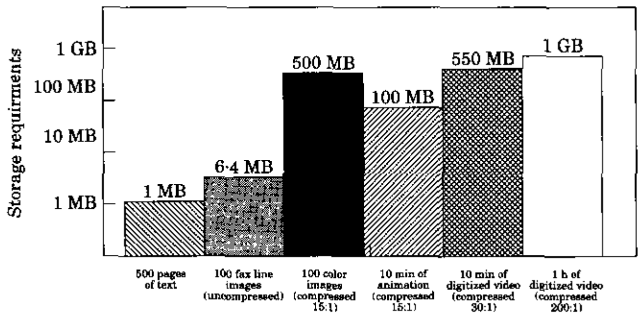


Figure 2. Storage requirements for a multimedia application, assuming that compression techniques are applied for images and video

Classification of Compression Techniques

Compression of digital data is based on various computational algorithms, which can be implemented either in software or in hardware. Compression techniques are classified into two categories: (a) lossless, and (b) lossy approaches (2). Lossless techniques are capable to recover the original representation perfectly. Lossy techniques involve algorithms which recover the presentation to be similar to the original one. The lossy techniques provide higher compression ratios, and therefore they are more often applied in image and video compression than lossless techniques. The classification schemes for lossless and lossy compression are presented in Figures 3(a) and (b), respectively.

The lossy techniques are classified into: (i) prediction-based techniques, (ii) frequency-oriented techniques, and (iii) importance- oriented techniques. Predictive-based techniques, such as ADPCM, predict subsequent values by observing previous values. Frequency-oriented techniques apply the Discrete Cosine Transform (DCT), which relates to fast Fourier transform. Importance-oriented techniques use other characteristics of images as the basis for compression. For example, DVI tèchnique uses color look-up tables and data filtering.

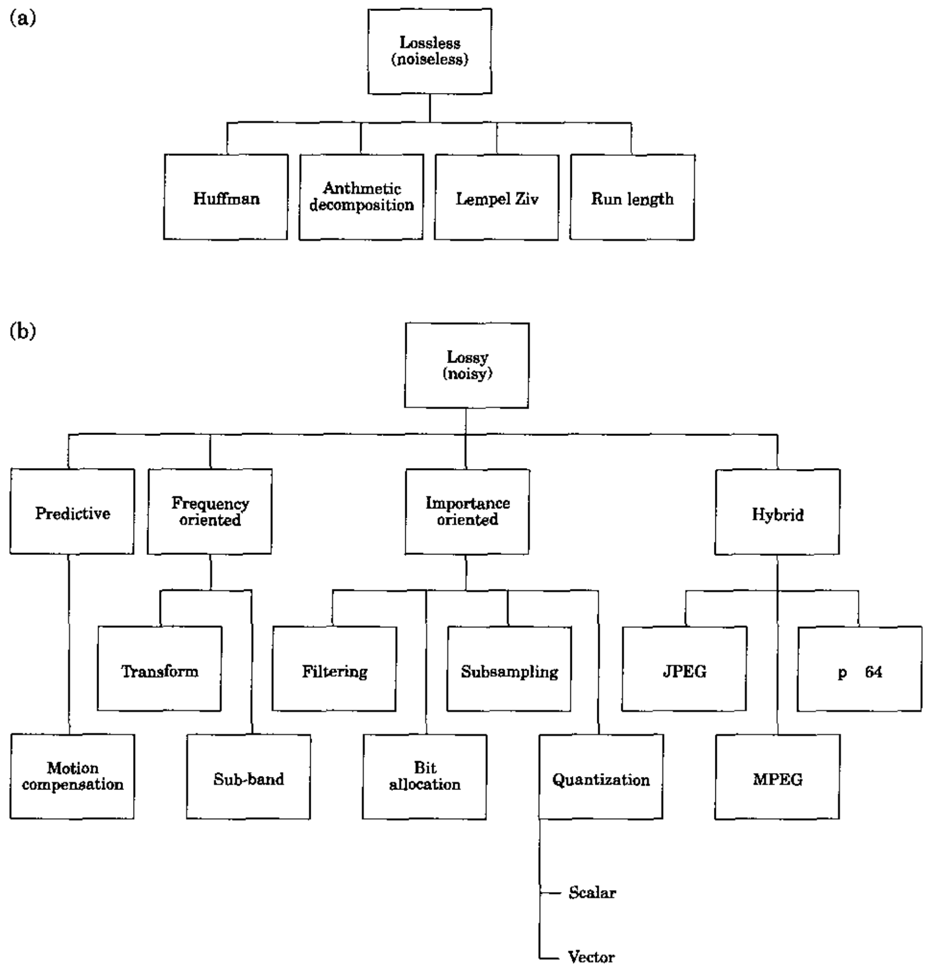


Figure 3. Classification schemes for (a) lossless and (b) lossy compression techniques

The hybrid compression techniques, such as JPEG, MPEG, and px64, combine several approaches, such as DCT and vector quantization or differential pulse code modulation. Recently, standards for digital multimedia have been established based on these three techniques, as illustrated in Table 2.

Image Concepts and Structures

A digital image represents a two-dimensional array of samples, where each sample is called a pixel. Precision determines how many levels of intensity can be represented, and is expressed as the number of bits/sample. According to precision, the images can be classified into:

Binary images, represented by 1 bit/sample. Examples include black/white photographs and facsimile images.

Computer graphics, represented by a lower-precision, as 4 bits/sample.

Grayscale images, represented by 8 bits/sample.

Color images, represented with 16, 24 or more bits/sample.

According to the trichromatic theory, the sensation of color is produced by selectively exciting three classes of receptors in the eye. In a RGB color representation system, shown in Figure 4, a color is produced by adding three primary colors: red, green and blue (RGB). The straight line, where R = G = B, specifies the gray values ranging from black to white.

Figure 5 illustrates how a three-sensor RGB color video camera operates and produces colors at a RGB monitor (4).

Table 2. Multimedia compression standards

Short name	Official name	Standards group	Compression ratios
JPEG	Digital compression and coding of continuous-tone still images	Joint Photographic Experts Group	15:1 (full color still-frame applications)
H.261 px64	Video encoder/decoder for audio–visual services at px64 Kbps	Specialist Group on Coding for Visual Telephone	100;1 to 2000:1 (video–based tele-communications)
MPEG	Coding of moving pictures and associated audio	Moving Pictures Experts Group	200:1 Motion-intensive applications

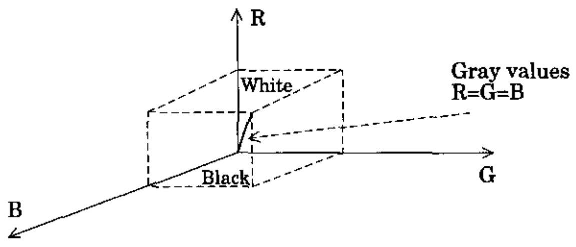


Figure 4. RGB representation of color images

Lights for sources of different colors are added together to produce the prescribed color.

Another representation of color images, YUV representation, describes luminance and chrominance components of an image. The luminance component provides a grayscale version of the image, while two chrominance components give additional informaion that converts the grayscale image to a color image.

The YUV representation is more natural for image compression. An image represented in RGB color, can be converted into YUV systems using the following transformations(4):

$$Y = 0{\cdot}3R + 0{\cdot}6G + 0{\cdot}1B \quad [1]$$

$$U = B - U \quad [2]$$

$$V = R - Y \quad [3]$$

where Y is the luminance component, and U and V are two chrominance components. When $R = G = B$, then $Y = R = G = B$, and $U = V = 0$, which represents a grayscale image.

Another color format, referred to as YCbCr; similar to the YUV format, is intentively used for image compression. In YCbCr format, Y is the same as in YUV system, however, U and V are scaled and zero-shifted to produce C_b and C_r, respectively, as follows:

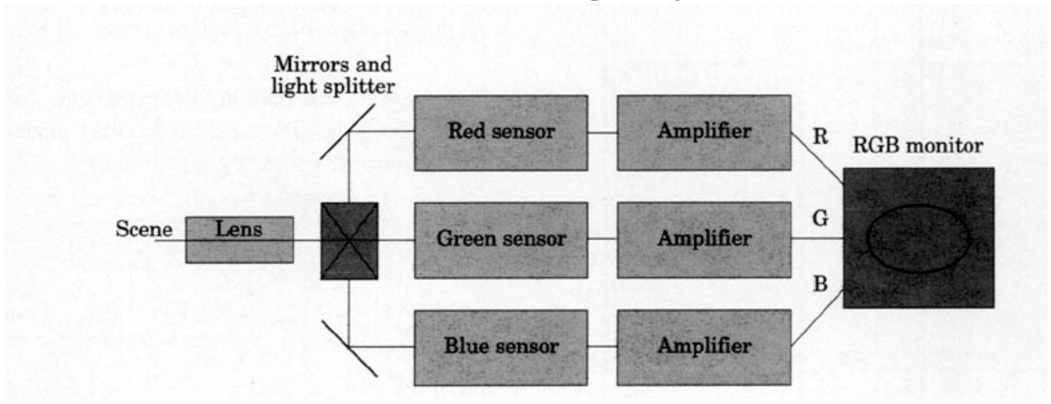


Figure 5. Three-sensor RGB color video camera

$$C_b = \frac{U}{2} + 0{\cdot}5 \qquad [4]$$

$$C_r = \frac{V}{1{\cdot}6} + 0{\cdot}5 \qquad [5]$$

In this way, chrominance components, C_b and C_r are always in the range (0,1).

Resolutions of an image system refers to its capability to reproduce fine detail (4). Higher resolution requires more complex imaging systems to represent these images in real-time. In computer systems, resolution is characterized by number of pixels (for example, VGA has a resolution of 640 × 480 pixels). In video systems, resolution refers to the number of line pairs resolved on the face of the display screen, expressed in cycles per picture height, or cycles per picture width. For example, the NTSC broadcast system in North America and Japan, denoted 525/59·94, has about 483 picture lines. (525 denotes the total number of lines in its rates, and 59·94 is its field rate in Hertz). The HDTV system, will approximately double the number of lines of current broadcast television at approximately the same field rate. The system will have 937 total lines and a frame rate of 65·95 Hz.

Figure 6 compares various image structures, showing the vertical and horizontal pixel counts (4), and the approximate total number of pixels.

The CCITT has adopted two picture formats for video-based telecommunications, Common Intermediate Format (CIF), and Quarter-CIF (QCIF), described in detail later in this paper.

The full-motion video is characterized with at least 24 Hz frame rate (or 24 frames/s), up to 30, or even 60 frames/s for

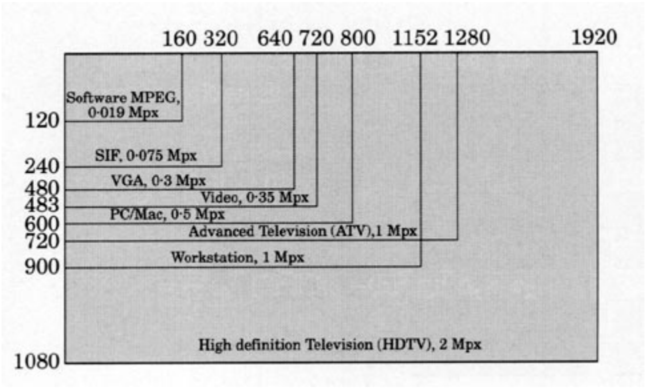


Figure 6. Various image structures

HDTV. For animation, acceptable frame rate is in the range 15/19 frames/s, while for video telephone is 5–10 frames/s. Videoconferencing and interactive multimedia applications require a rate of 15–30 frames/s.

JPEG Algorithm for Full-Color Still Image Compression

Originally, JPEG standard is targeted for full-color still frame applications, achieving 15:1 average compression ratio (3,5). However, JPEG is also applied in some real-time, full-motion video applications (Motion JPEG–MJPEG). JPEG standard provides four modes of operation:

- *sequential DCT-based encoding*, in which each image component is encoded in a single left-to-right, top-to-bottom scan,
- *progressive DCT-based encoding*, in which the image is encoded in multiple scans, in order to produce a quick, rough decoded image when the transmission time is long,
- *lossless encoding*, in which the image is encoded to guarantee the exact reproduction, and
- *hierarchical encoding*, in which the image is encoded in multiple resolutions.

Sequential JPEG Encoder and Decoder

The block diagram of the JPEG sequential encoder and decoder is shown in Figure 7.

JPEG Encoder

The original samples, in the range $[0, 2^P - 1]$, are shifted to sign in the range $[-2^{P-1}, 2^{P-1} - 1]$. For a grayscale image, where $p = 8$, the original samples in the range [0, 511], are shifted in the range [-128. + 127].

These values are then transformed into the frequency domain using the Forward Discrete Cosine Transform (FDCT) using the following equations:

$$F(u, v) = \frac{C(u)}{2} \bullet \frac{C(v)}{2} \sum_{x=0}^{7} \sum_{y=0}^{7} f(x, y)$$

$$cos\,\frac{(2x + 1)u\pi}{16}\, cos\,\frac{(2y + 1)v\pi}{16} \qquad [6]$$

where

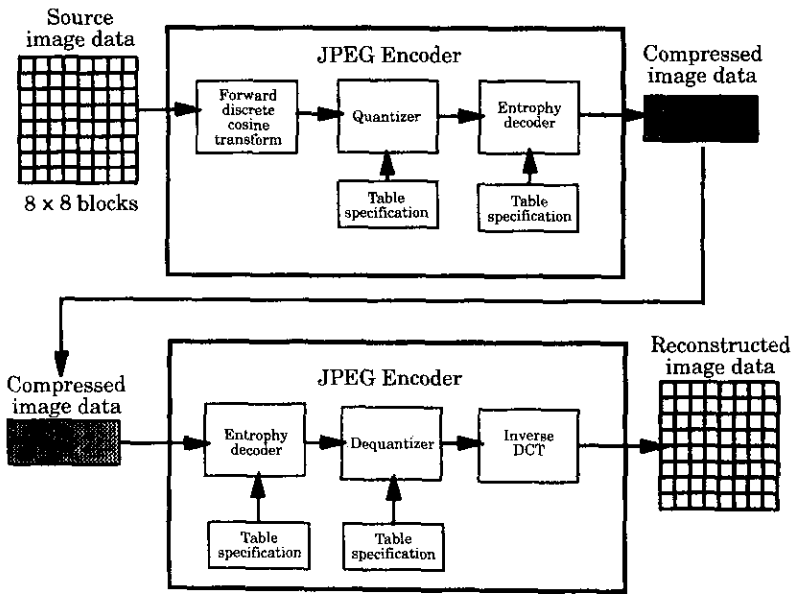


Figure 7. Block diagrams of sequential JPEG encoder and decoder

$C(u) = \frac{1}{\sqrt{2}}$ for $u = 0$

$C(u) = 1$ for $u > 0$

$C(v) = \frac{1}{\sqrt{2}}$ for $v = 0$

$C(v) = 1$ for $v > 0$

The transformed 64-point discrete signal is a function of two spatial dimensions x and y, and its components are called spatial frequencies or DCT coefficients.

The F(0, 0) coefficient is called the "DC coefficient", and the remaining 63 coefficients are called the "AC coefficients". A number of fast DCT algorithms are analysed (3).

For a typical 8 × 8 image block, most of the spatial frequencies have zero or near-zero values, and need not to be encoded. This is illustrated in JPEG example, presented later in this paper by Pennenbaker and Mitchell. This fact is the foundation for achieving data compression. In the next step, all the 64 DCT coefficients are quantized using a 64-element quantization table, specified by the application. The quantization reduces the amplitude of the coefficients which contribute little or nothing to the quality of the image, with the purpose of increasing the number of zero-value coefficients. Quantization also discards information which is not visually significant.

The quantization is performed according to the following equation:

$$F_q(u, v) = Round\left[\frac{F(u, v)}{Q(u, v)}\right] \qquad [7]$$

where $Q(u, v)$ are quantization coefficients specified in the Quantization Table. Each element $Q(u, v)$ is an integer from 1 to 255, which specifies the step size of the quantizer for its corresponding DCT coefficient.

A set of four Quantization Tables are specified by JPEG standard (3). In the JPEG example, a quantization formula is used to produce the Quantization Tables.

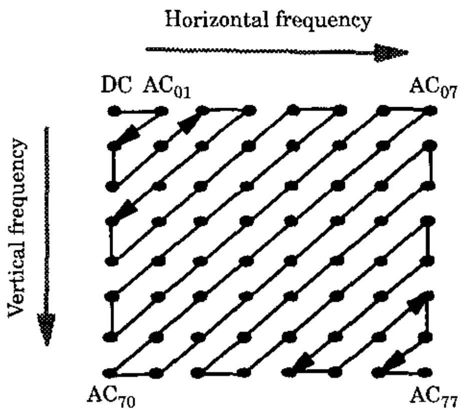


Figure 8. Zig-zag ordering of AC coefficients

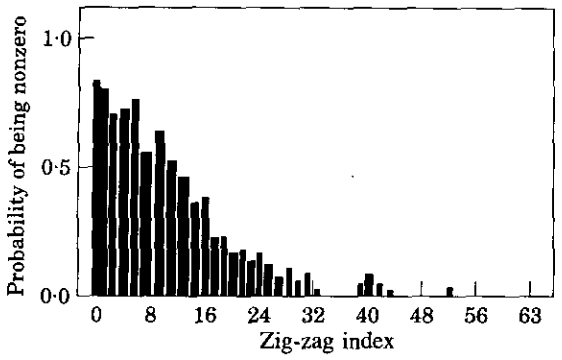


Figure 9. Probability of being nonzero of zig-zag ordered DCT coefficients

After quantization, the 63 AC coefficients are ordered into the 'zig-zag' sequence, as shown in Figure 8. This zig-zag ordering will help to facilitate the next phase, entropy encoding, by placing low-frequency coefficients, which are more likely to be nonzero, before high-frequency coefficients. This is confirmed by the experiment presented by Pennenbaker & Mitchell (3) and the results shown in Figure 9. These results show that when the coefficients are ordered zig-zag, the probability of coefficients being zero is an increasing monotonic function of the index. The DC coefficients, which represent the average values of the 64 image samples, are coded using the predictive coding techniques, as illustrated in Figure 10.

The reasons for predictive coding of DC coefficients is that there is usually a strong correlation between the DC coefficients of adjacent 8 × 8 blocks. As a consequence, the compression ratio will be improved. Finally, the last block in the JPEG encoder is the entropy coding, which provides additional compression by encoding the quantized DCT coefficients into more compact form. The JPEG standard specifies two entropy coding methods: Huffman coding and arithmetic coding (3). The baseline sequential JPEG encoder uses Huffman coding.

The Huffman coder converts the DCT coefficients after quantization into a compact binary sequence using two steps: (i) forming intermediate symbol sequence, and (ii) converting intermediate symbol sequence into binary sequence using Huffman tables.

In the intermediate symbol sequence, each AC coefficient is represented by a pair of symbols:

where:

Symbol-1	Symbol-2
(RUNLENGTH, SIZE)	(AMPLITUDE)

RUNLENGTH is the number of consecutive zero-lined AC coefficients preceding the nonzero AC coefficient. The value of RUNLENGTH is in the range 0 to 15, which requires 4 bits for its representation.

SIZE is the number of bits used to encode AMPLITUDE. The number of bits for AMPLITUDE is in the range of 0 to 10 bits, so there are 4 bits needed to code SIZE.

AMPLITUDE is the amplitude of the nonzero AC coefficient in the range of [+1024 to −1023], which requires 10 bits for its coding. In an example, if the sequence of AC coefficients is:

$$\underbrace{0, 0, 0, 0, 0, 0}_{6}, 476$$

the symbol representation of the AC coefficient 476 is:

(6, 9) (476)

where RUNLENGTH = 6, SIZE = 9, and AMPLITUDE = 476.

If RUNLENGTH is greater than 15, then symbol-1 (15, 0) is interpreted as the extension symbol with runlength = 16. These can be up to three consecutive (15, 0) extensions.

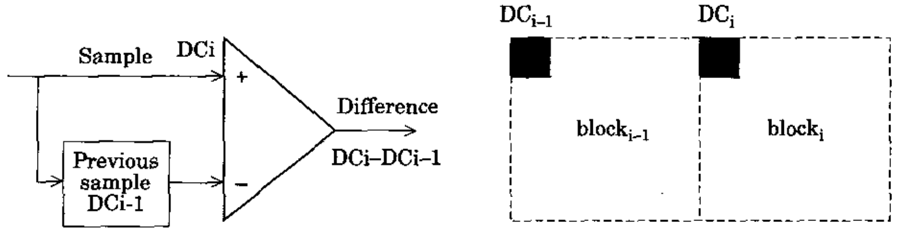


Figure 10. Predictive coding for DC coefficients

In the following example:

(15, 0) (15, 0) (7, 4) (12)

RUNLENGTH is equal to 16 + 16 + 7 = 39, SIZE = 4, and AMPLITUDE = 12. The symbol (0,0) means 'End of block' (EOB) and terminates each 8×8 block. For DC coefficients, the intermediate symbol representation consists of:

Symbol-1	Symbol-2
(SIZE)	(AMPLITUDE)

Because DC coefficients are differentially encoded, this range is double the range for AC coefficients, and is [−2048, +2047].

The second step in Huffman coding is converting the intermediate symbol sequence into binary sequence. In this phase, symbols are replaced with variable length codes, beginning with the DC coefficients, and continuing with AC coefficients.

Each Symbol-1 (both for DC and AC coefficients) is encoded with a Variable-Length Code (VLC), obtained from the Huffman table set specified for each image component. The generation of Huffman tables is discussed by Pennenbaker & Mitchell (3). Symbols-2 are encoded using a Variable-Length Integer (VLI) code, whose length in bits is given in Table 3.

Table 3. Huffman Coding of Symbols-2

Size	Amplitude range
1	(−1, 1)
2	(−3, −2) (2,3)
3	(−7..−4) (4..7)
4	(−15..−8) (8..15)
5	(−31..−16) (16..31)
6	(−63..−32) (32..63)
7	(−127..−64) (64..127)
8	(−255..−128) (128..255)
9	(−511..−256) (256..511)
10	(−1023..−512) (512..1023)

For example, for an AC coefficient presented as the symbols:

(1, 4) (12)

the binary presentation will be: (1111101101100), where (111110110) is VLC obtained from the Huffman table, and (1100) is VLI code for 12.

JPEG Decoder

In the JPEG sequential decoding, all the steps from the encoding process are inversed and implemented in reverse order, as shown in Figure 7.

First, an entropy decoder (such as Huffman) is implemented on the compressed image data. The binary sequence is converted to a symbol sequence using Huffman tables (VLC coefficients) and VLI decoding, and then the symbols are converted into DCT coefficients. Then, the dequantization is implemented using the following function:

$$F_q(u, v) = F_q(u, v) \times Q(u, v) \qquad [8]$$

Then, the Inverse Discrete Cosine Transform (IDCT) is implemented on dequantized coefficients in order to convert the image from frequency domain into spatial domain. The IDCT equation is defined as:

$$F(x, y) = \frac{1}{4} [\sum_{u=0}^{7} \sum_{v=0}^{7} C(u)C(v)F(u, v)$$

$$cos \frac{(2x + 1)u\pi}{16} cos \frac{(2y + 1)v\pi}{16} \qquad [9]$$

where

$C(u) = \frac{1}{\sqrt{2}}$ for $u = 0$

$C(u) = 1$ for $u > 0$

$C(v) = \frac{1}{\sqrt{2}}$ for $v = 0$

$C(v) = 1$ for $v > 0$

The last step consists of shifting back the decompressed samples in the range $[0, 2^P - 1]$.

Compression Measures
The basic measure for the performance of a compression algorithm is Compression Ratio (C_r), defined as:

$$C_r = \frac{\text{Original data size}}{\text{Compressed data size}} \qquad [10]$$

There is a trade-off between the compression ratio and the picture quality. Higher compression ratios may produce lower picture quality. Quality and compression can also vary according to source image characteristics and scene content. One measure for the quality of the picture, proposed by Wallace (5), is the number of bits per pixel in the compressed image (N_b) which is defined as the total number of bits in the compressed image divided by the number of pixels:

$$N_b = \frac{\text{Encoded number of bits}}{\text{Number of pixels}} \qquad [11]$$

According to this measure, four different picture qualities are defined (5), as shown in Table 4.

Another statistical measure that can be used to evaluate various compression algorithms is the Root Mean Square (RMS) error, calculated as:

$$RMS = \frac{1}{n}\sqrt{\sum_{i=1}^{n} (X_i - \hat{X}_i)^2} \qquad [12]$$

where

X_i – original pixel values
$\hat{X}_i$ – pixel values after decompression
n – total number of pixels in an image

Table 4. Picture quality characteristics

Nb [bits/pixel]	Picture quality
0·25–0·5	Moderate to good quality
0·5–0·75	Good to very good quality
0·75–1·0	Excellent quality
1·5–2·0	Usually indistinguishable from the original

The RMS shows the statistical difference between the original and decompressed images. However, in some cases it may happen that the quality of a decompressed image with higher RMS is better than one with lower RMS.

In the next two sections, we will calculate these measures in several examples.

Sequential JPEG Encoding Example

In order to illustrate all the steps in baseline sequential JPEG encoding, we present step-by-step results obtained in encoding an 8 × 8 block of 8-bit samples, as illustrated in Figure 11. The original 8 × 8 block is shown in Figure 11(a), and after shifting, the obtained block is given in Figure 11(b). After applying the FDCT, the obtained DCT coefficients are given in Figure 11(c). Note that, except for low-frequency coefficients, all the other coefficients are close to zero.

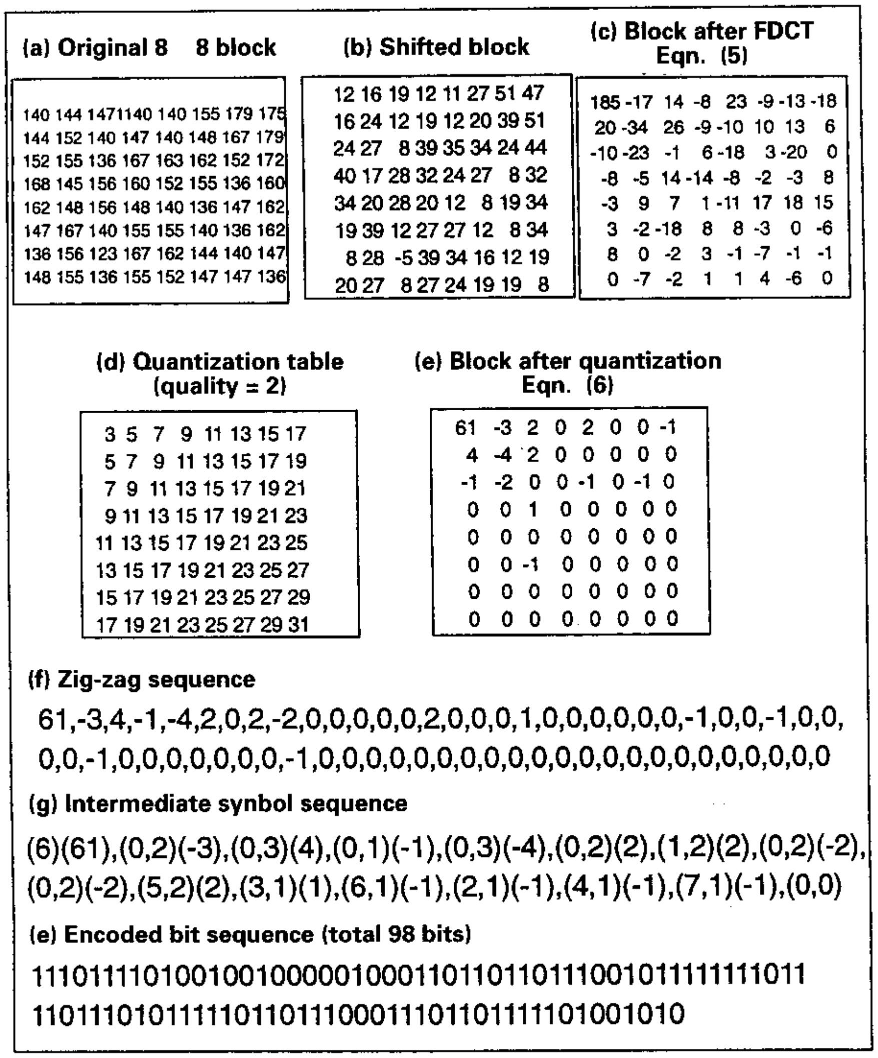
For the generation of the Quantization Table, we used the program proposed by Nelson (6):

```
for i = 0 to n;
  for j = 0 to n;
    Q[i,j] = 1 + (1 + i + j)*quality;
  end j;
end i;
```

The parameter 'quality' specifies the quality factor and recommended range is from 1 to 25, where quality = 1 gives the best quality, but the lowest compression rate, and quality = 25 gives the worst quality and the highest compression rate. In our example, we used quality = 2, which generates the Quantization Table in Figure 11(d).

After implementing quantization, the obtained quantized coefficients are shown in Figure 11(e). Note that a number of high-frequency AC coefficients are zero.

The zig-zag ordered sequence of quantized coefficients is shown in Figure 11(f), and the intermediate symbol sequence in Figure 11(g). Finally, after implementing Huffman codes, the obtained encoded bit sequence is shown in Figure 11(h). The Huffman table used in this example is proposed in the JPEG standard for luminance AC coefficients (3), and the partial table needed to code the symbols from Figure 11(g) is given in Table 5.



(a) Original 8 8 block

140	144	147	140	140	155	179	175
144	152	140	147	140	148	167	179
152	155	136	167	163	162	152	172
168	145	156	160	152	155	136	160
162	148	156	148	140	136	147	162
147	167	140	155	155	140	136	162
136	156	123	167	162	144	140	147
148	155	136	155	152	147	147	136

(b) Shifted block

12	16	19	12	11	27	51	47
16	24	12	19	12	20	39	51
24	27	8	39	35	34	24	44
40	17	28	32	24	27	8	32
34	20	28	20	12	8	19	34
19	39	12	27	27	12	8	34
8	28	-5	39	34	16	12	19
20	27	8	27	24	19	19	8

(c) Block after FDCT Eqn. (5)

185	-17	14	-8	23	-9	-13	-18
20	-34	26	-9	-10	10	13	6
-10	-23	-1	6	-18	3	-20	0
-8	-5	14	-14	-8	-2	-3	8
-3	9	7	1	-11	17	18	15
3	-2	-18	8	8	-3	0	-6
8	0	-2	3	-1	-7	-1	-1
0	-7	-2	1	1	4	-6	0

(d) Quantization table (quality = 2)

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

(e) Block after quantization Eqn. (6)

61	-3	2	0	2	0	0	-1
4	-4	2	0	0	0	0	0
-1	-2	0	0	-1	0	-1	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	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

(f) Zig-zag sequence

61,-3,4,-1,-4,2,0,2,-2,0,0,0,0,2,0,0,0,1,0,0,0,0,0,-1,0,0,-1,0,0,
0,0,-1,0,0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

(g) Intermediate synbol sequence

(6)(61),(0,2)(-3),(0,3)(4),(0,1)(-1),(0,3)(-4),(0,2)(2),(1,2)(2),(0,2)(-2),
(0,2)(-2),(5,2)(2),(3,1)(1),(6,1)(-1),(2,1)(-1),(4,1)(-1),(7,1)(-1),(0,0)

(e) Encoded bit sequence (total 98 bits)

1110111101001001000001000110110110111001011111111011
1101110101111101101110001110110111101001010

Figure 11. Step-by-step procedure in JPEG sequential encoding of a 8 × 8 block

Note that the DC coefficient is threatened as being from the first 8 × 8 block in the image, and therefore, it is coded directly (not using predictive coding as all the remaining DC coefficients).

For this block the compression ratio can be calculated as:

$$C_r = \frac{\text{Original number of bits}}{\text{Encoded number of bits}}$$

$$= \frac{64 \times 8}{98} = \frac{512}{98} = 5{\cdot}22$$

and the number of bits/pixel in the compressed form is:

$$N_b = \frac{\text{Encoded number of bits}}{\text{Number of pixels}} = \frac{98}{64} = 1{\cdot}53$$

Sequential JPEG Experiments

In this section we present results of experiments obtained by implementing the sequential JPEG algorithms for grayscale images (7). The JPEG algorithm has been implemented and run on two Sun machines: Sparc 10 Model 41, characterized by 109 MIPS and 22 MFLOPS, and Sparc IPC, characterized with 17 MIPS and 2.1 MFLOPS. We used the algorithm, described previously, to generate different Quantization Tables, selecting the following values for parameter quality: 1, 2, 4, 8, 16 and 25. Several grayscale images

Table 5. Partial Huffman table for Luminance AC coefficients

(Runlength, size)	Code word
(0,,0) EOB	1010
(0,1)	00
(0,2)	01
(0,3)	100
(1,2)	11011
(2,1)	11100
(3,1)	111010
(4,1)	111011
(5,2)	11111110111
(6,1)	1111011
(7,1)	11111010

(320 × 200 pixels, 8 bits/pixel) were compressed and decompressed using the JPEG algorithm.

Complete results are given by Kessler *et al.* (7); here results for the grayscale image 'Lisa' are presented. The empirical results are given in Table 6, and the original and decompressed images, for different quality factors, are shown in Figure 12.

From Table 6 and Figure 13 it is clear that higher compression ratios produce a lower quality image, and consequently the number of bits/pixel will decrease and RMS error will increase. Execution times, which include both encoder and decoder times, are very consistent at different quality levels. The average processing at Sparc has been over six times faster than at the IPC, which is consistent with the reported MIPS performance of these two machines.

JPEG Compression of Color Images

The described sequential JPEG algorithm can be easily expanded for compression of color images, or in a general case for compression of multiple-component images. The JPEG source image model consists of 1 to 255 image components (5, 8), called color or spectral bands, as illustrated in Figure 13.

Table 6. Results of JPEG Compression for Grayscale Image 'Lisa' (320 ×240 pixels)

Quality factors	Original number of bits	Compressed number of bits	Compression ratio (Cr)	Bits/pixel (Nb)	RMS error	Execution times [ms]	
						Sun Sparc 10/41	Sun IPC
1	512,000	48,021	10·66	0·75	2·25	0·59	6·31
2	512,000	30,490	16·79	0·48	2·75	0·59	6·22
4	512,000	20,264	25·27	0·32	3·43	0·58	6·39
8	512,000	14,162	36·14	0·22	4·24	0·59	6·44
15	512,000	10,479	48·85	0·16	5.36	0·58	6·45
25	512,000	9,034	56·64	0·14	6·40	0·58	6·32
DC only	512,000	7,688	66·60	0·12	7·92	0·57	6·25

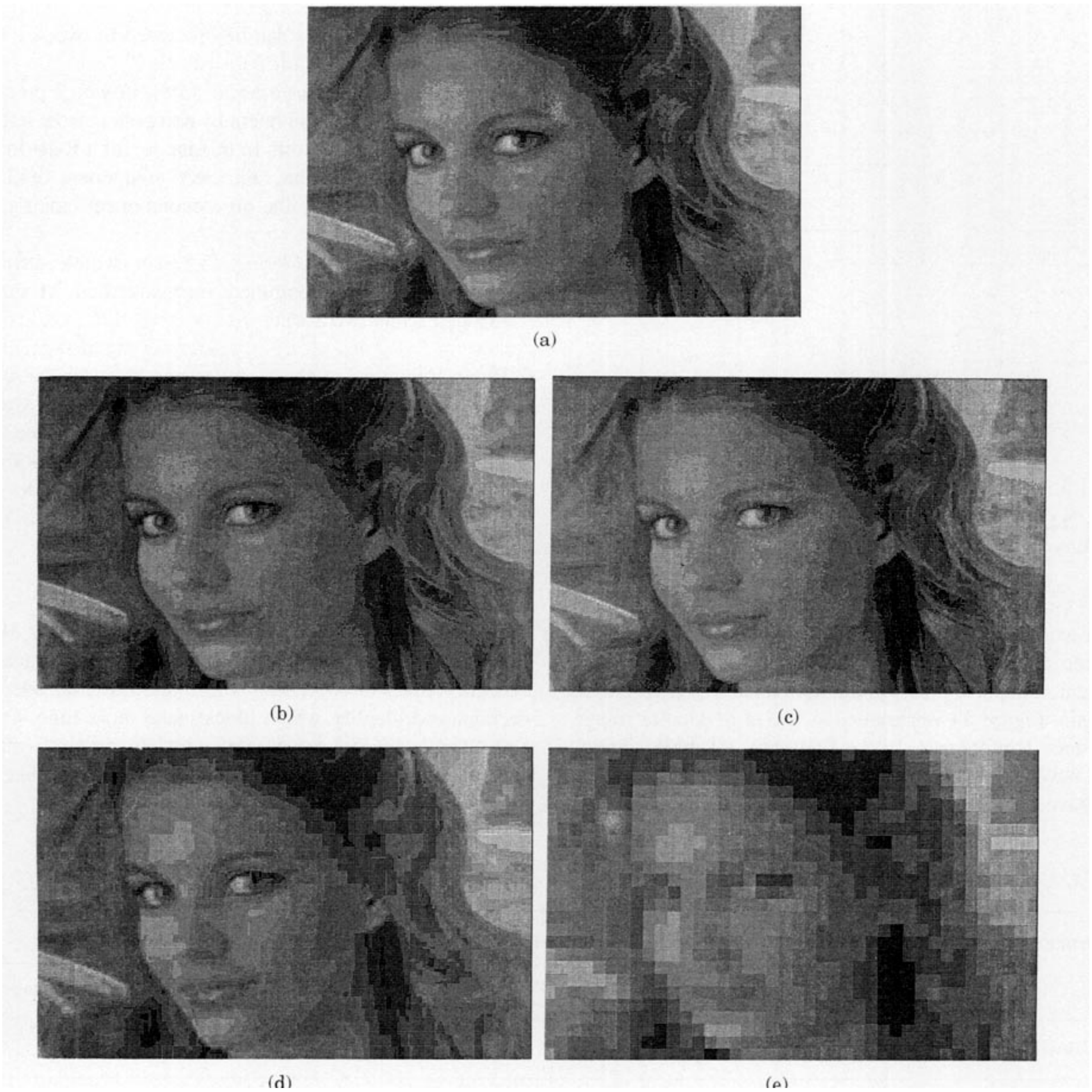


Figure 12. Original image 'Lisa' and after decompression for different quality factors (a) Original image; (b) quality 1, CR = 10·66; (c) quality = 2, Cr = 16·79; (d) quality = 15, Cr = 48·85; (e) DC coefficient only, Cr 66·60

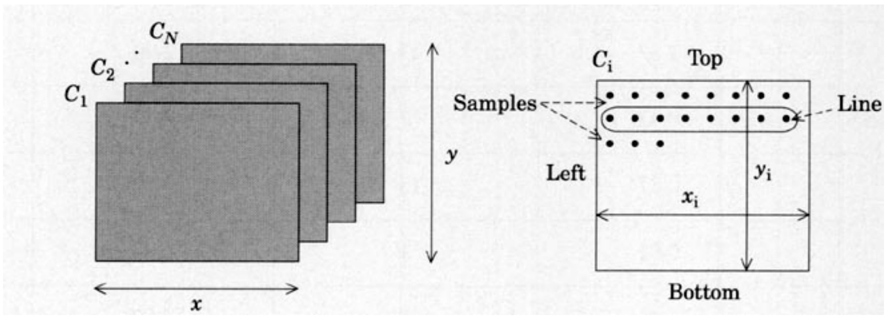


Figure 13. JPEG source image model

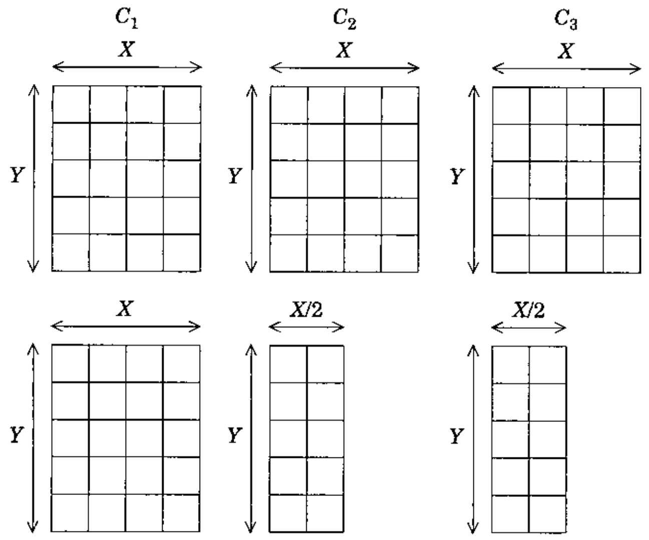


Figure 14. Color image with three components: (a) with same resolutions; (b) with different resolutions

For example, RGB and YUV representations both consist of three color components. Each component may have a different number of pixels in the horizontal (X_i) and vertical (Y_i) axis. Figure 14 illustrates two cases of a color image with three components. In the first case, all three components have the same resolutions, while in the second case they have different resolutions.

The color components can be processed in two ways:

(a) *non-interleaved data ordering* (5,8), in which processing is performed component by component from left-to-right and top-to-bottom. In this mode, for a RGB image with high resolution, the red component will be displayed first, then the green component, and finally the blue component.
(b) *interleaved data ordering* (5,8), in which different components are combined into so-called Minimum Coded Units (MCUs).

Block diagrams of the encoder and decoder for color JPEG compression are identical to those for grayscale image compression, shown in Figure 7, except the first block into encoder is a color space conversion block (for example, RGB to YUV conversion), and at the decoder side the last block is the inverse color conversion, such as YUV to RGB.

Color JPEG Experiment
In this experiment, the execution time of a color JPEG algorithm was analysed (9). The goal was to measure execution times of different blocks in the JPEG encoder and decoder and identify which blocks take more time. Originally, this experiment was performed in order to design some of the steps in JPEG algorithm and parallelize the most effective parallel algorithm.

Table 7. Color JPEG experiment

Operation	JPEG Compression		JPEG Decompression	
	Time (s)	Percentage	Time (s)	Percentage
Read/write file	0·75	11	0·34	6
RGB/YUV conversion	1·40	21	1·14	19
Recorder from/for YUV	0·28	4	0·49	8
FDCT/IDCT	2·87	43	2·90	48
Quantization/dequantiz.	0·47	7	0·37	6
Huffman enc./dec.	0·87	13	0·70	12
Write/read file I/O	0·01	1	0·06	1
Total time	6·65	100%	6·00	100%

The color JPEG algorithm was run on an i486/33MHz machine with no maths co-processor. The 320 × 240 'Targa' image, with 24 bits/pixel, was compressed and then decompressed. The results are presented in Table 7.

From Table 7 it can be seen that the JPEG algorithm spends 48% of decompression time performing IDCT, and similarly 42% of compression time performing FDCT. Color conversion also requires about 20% of total time execution, and Huffman coding about 12%.

Assuming a Pentium personal computer with maths co-processor, the expected performance of the JPEG algorithm can be improved about sixfold which will give about 1 s for decompression.

Progressive JPEG Compression

In some applications, an image may have large numbers of pixels and the decompression process, including transmission of the compressed image over the network, may take several minutes. In such applications, there may be a need to produce a quick rough image quickly, and then improve its quality using multiple scans (3,5,8). The progressive JPEG mode of operation produces a sequence of scans, each scan coding a subset of DCT coefficients. Therefore, the progressive JPEG encoder must have an additional buffer at the output of the quantizer and before the entropy encoder. The size of the buffer should be large enough to store all DCT coefficients of the image, each of which is 3 bits larger than the original image samples.

Figure 15 illustrates the differences in displaying a decompressed image in the progressive and sequential JPEG.

Progressive JPEG compression can be achieved using three algorithms:

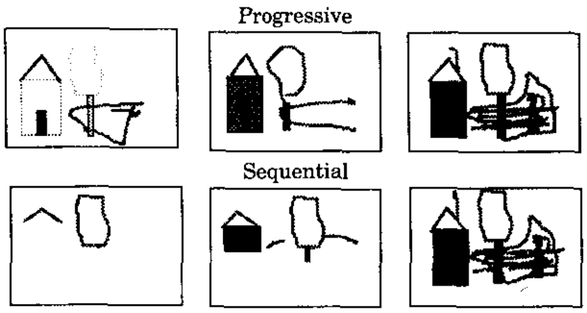


Figure 15. Progressive versus sequential JPEG decoding

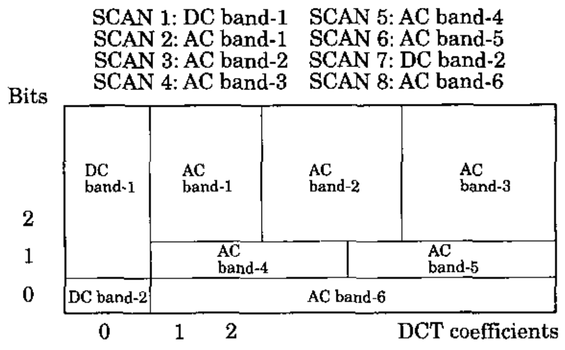


Figure 16. An example of image encoded using the combined progressive JPEG algorithm

(1) Progressive spectral selection algorithm
(2) Progressive successive approximation algorithm
(3) Combined progressive algorithm

In the *progressive spectral selection algorithm*, the DCT coefficients are grouped into several spectral bands. Typically, low-frequency DCT coefficients bands are sent first, and then higher-frequency coefficients. For example, a sequence of four spectral bands may look like this:

Band 1: DC coefficient only
Band 2: AC_1 and AC_2 coefficients
Band 3: AC_3, AC_4, AC_5, AC_6, coefficients
Band 4: $AC_7 \ldots AC_{63}$, coefficients

In the *progressive successive approximation algorithm*, all DCT coefficients are sent first with lower precision, and then refined in later scans. For example, a sequence of three successive approximation bands may be as follows:

Band 1: All DCT coefficients (divided by four)
Band 2: All DCT coefficients (divided by two)
Band 3: All DCT coefficients (full resolution)

Combined progressive algorithm combines both spectral selection and successive approximation algorithms. Figure 16 illustrates an image divided into eight combined scans. For example, in the first scan only DC coefficients divided by two (will lower resolution) will be sent, and so on.

Progressive JPEG Experiment

In this experiment, we implemented both progressive algorithms, spectral selection (SS), and successive approximation (SA) to compress and decompress a 320 × 200 grayscale image "Cheetah". In both cases, we used four scans, as follows:

	Spectral selection	Successive approximation
Scan 1	DC, AC1, AC2	All DCT – divided by 8
Scan 2	AC3–AC9	All DCT – divided by 4
Scan 3	AC10–AC35	All DCT divided by 2
Scan 4	AC 36–AC 63	All DCT – full resolution

The results of the experiments are presented in Tables 8 and 9. Figures 17 and 18 show results of sequential and progressive JPEG compression using EasyTech/Codec, respectively. In the case of sequential JPEG, compression ratios obtained are in the range 21 to 52. Assuming an image transmitted over a 64 Kbit/s ISDN network, the following four images are produced using progressive JPEG (Figure 18). The first image of reduced quality appears on the screen very quickly, in 0.9 s. Each subsequent pass improves the image quality and it is obtained after 1·6, 3·6, and 7·0 s, respectively.

Note that in both algorithms, the first scan will be transmitted 6–7 times faster than the sequential JPEG algorithm (26,215 bits in SA and 29,005 bits in SS versus 172,117 bits in sequential JPEG).

Sequential Lossless JPEG Compression

JPEG standard also supports a lossless mode of operation, by providing a simple predictive compression algorithm, rather than DCT-based technique, which is a lossy one. Figure 19 shows the block diagram of the lossless JPEG encoder, in which a prediction block has replaced the FDCT

Table 8. Progressive spectral selection JPEG. (Image 'Cheetah': 320 × 240 pixels –> 512,000 bits)

Scan number	Bits transmitted	Compression ratio	Bits/pixel	RMS error
1	29,005	17·65	0·45	19·97
2	37,237	7·73	1·04	13·67
3	71,259	3·72	2·15	7·90
4	32,489	3·01	2·66	4·59
Sequential JPEG	172,117	2·97	2·69	4·59

Table 9. Progressive successive approximation JPEG. (Image 'Cheetah': 320 × 240 pixels –> 512,000 bits)

Scan number	Bits transmitted	Compression ratio	Bits/pixel	RMS error
1	26,215	19·53	0·41	22·48
2	34,506	8·43	0·95	12·75
3	63,792	4·11	1.95	7·56
4	95,267	2·33	2·43	4·59
Sequential JPEG	172,117	2·97	2·69	4·59

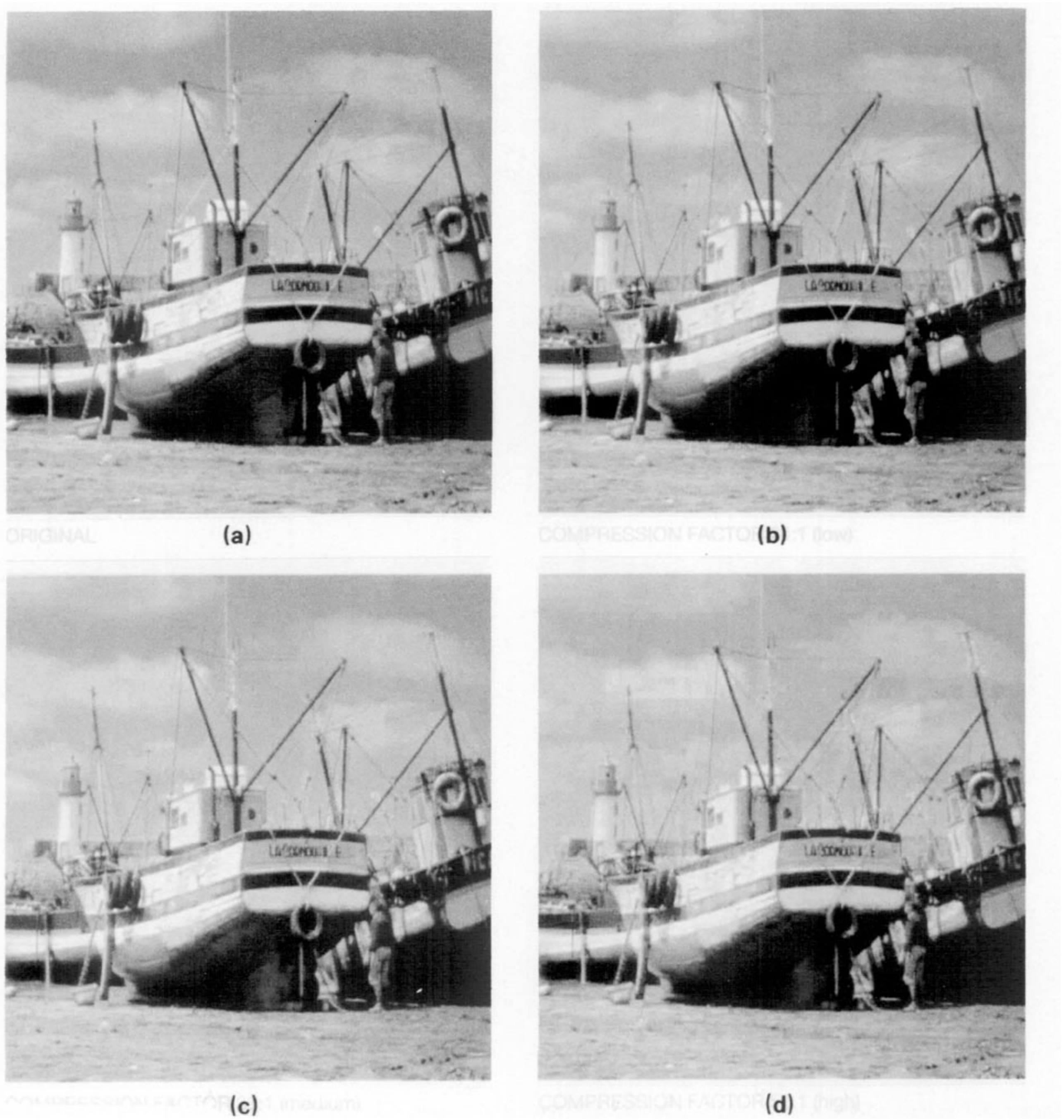


Figure 17. Sequential JPEG results using Easy/Tech Codec (courtesy of Autograph International). (a) Original; (b) Compression factor 21:1 (low); (c) Compression factor 33:1 (medium); (d) Compression factor 52:1 (high)

and the Quantization blocks from the baseline sequential DCT-based JPEG.

The predictor block works in such a way that a prediction of the sample $\hat{X}$ is calculated on the basis of previous samples A, B and C, and then the difference $\Delta X = X - \hat{X}$ is computed, where X is the actual value of the sample [Figures 20(a) and (b)]. Then, the difference, ΔX, is coded using the Huffman arithmetic encoder.

Table 10 illustrates several different predictor formulae that can be used for lossless prediction. Lossless JPEG

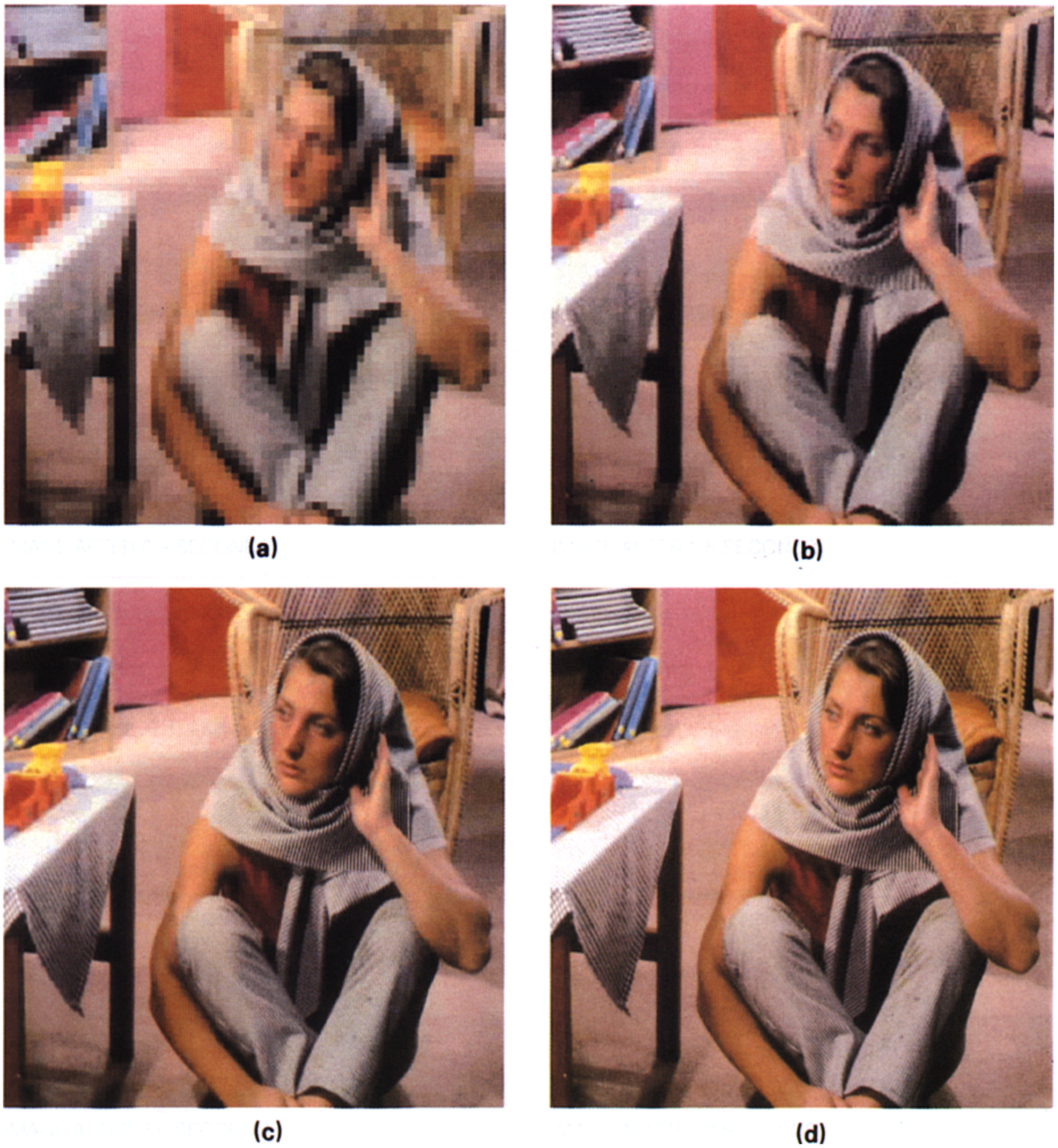


Figure 18. Progressive JPEG results using Easy/Tech Codec (courtesy of Autograph International) (a) Image after 0·9 s; (b) Image after 1·6 s; (c) Image after 3·6 s; (d) Image after 7·0 s

compression typically gives around 2:1 compression ratio for moderately complex color images.

Hierarchical JPEG Compression

The hierarchical JPEG mode of operation creates a set of compressed images beginning with the small images and then increasing resolution. This process is called downsampling, or pyramidal coding (3,5,8). After the downsampling phase, each lower-resolution image is scaled up to the next resolution (upsampling process), and is used as a prediction for the following stage. Hierarchical JPEG encoder requires significantly more buffer space. However, the benefits are that the encoded image is immediately available at different resolutions (8).

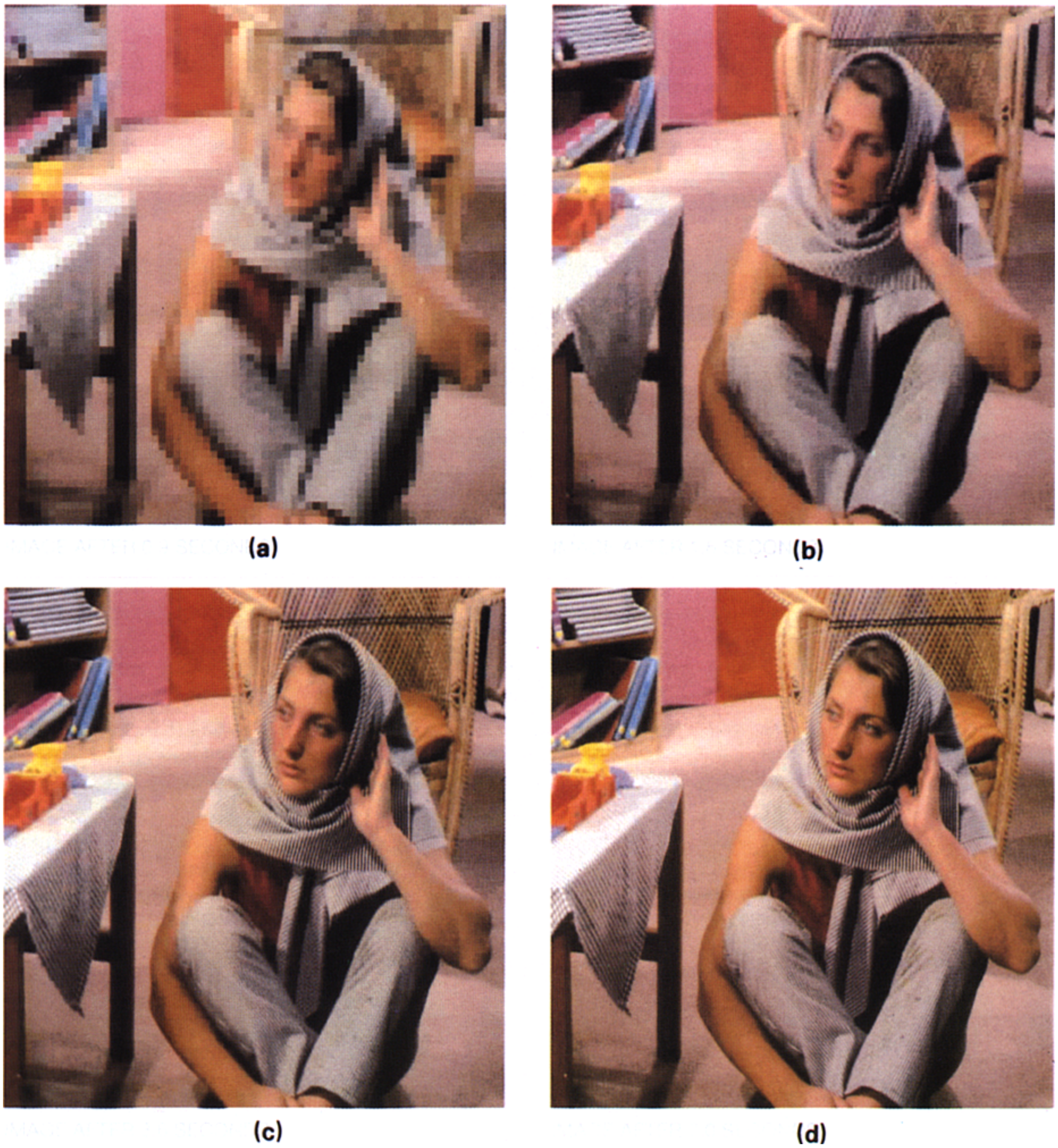


Figure 18. Progressive JPEG results using Easy/Tech Codec (courtesy of Autograph International) (a) Image after 0·9 s; (b) Image after 1·6 s; (c) Image after 3·6 s; (d) Image after 7·0 s

compression typically gives around 2:1 compression ratio for moderately complex color images.

Hierarchical JPEG Compression

The hierarchical JPEG mode of operation creates a set of compressed images beginning with the small images and then increasing resolution. This process is called down-sampling, or pyramidal coding (3,5,8). After the down-sampling phase, each lower-resolution image is scaled up to the next resolution (upsampling process), and is used as a prediction for the following stage. Hierarchical JPEG encoder requires significantly more buffer space. However, the benefits are that the encoded image is immediately available at different resolutions (8).

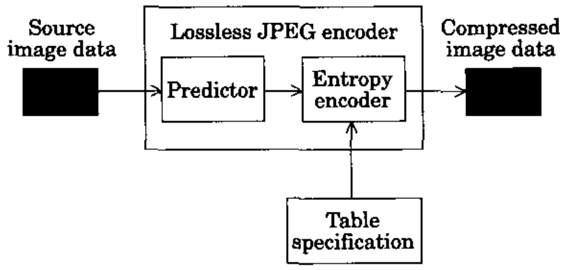


Figure 19. Block diagram of the lossless JPEG encoder

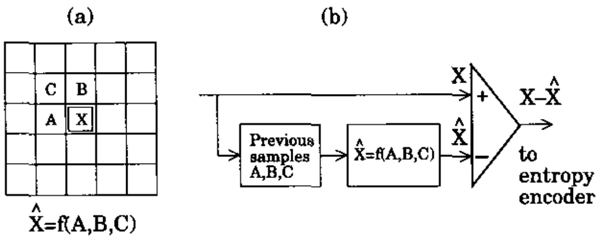


Figure 20. Lossless JPEG encoding: (a) location of four samples in the predictor (b) predictor block diagram

Conclusion

In this paper we presented JPEG compression standard, intended for full-color still image compression. A number of experiments have been performed with the aim of illustrating features and performance of various JPEG techniques, such as sequential, progressive, and lossless techniques. A number of software and hardware JPEG implementations have been done, and commercial systems include ALICE 200 from Telephoto, the Super Still-Frame Compression Card from New Medial Graphics, and Optipac 3250 and 5250 JPEG compression accelerator boards from Optivision. LSI Logic has developed L64702 JPEG compressor that can support real-time video applications.

The most popular use of JPEG image compression technology include its use in photo ID systems telecommunications of images, military image systems, and distributed image management systems.

Table 10. Predictors for lossless JPEG compression

Selection value	Predictor formula
0	No prediction
1	X = A
2	X = B
3	X = C
4	X = A + B − C
5	X = A + (B − C)/2
6	X = B + (A − C)/2
7	X = (A + B)/2

Acknowledgements

The author would like to thank the following graduate students from Multimedia Laboratory at Florida Atlantic University, who successfully completed compression projects under the author's supervision, and provided a number of useful results presented in this chapter.

Mark Kessler, Jan Alexander, Mauricio Cuervo, Lou Horowitz, and Henry Pesulima have developed sequential JPEG and provided results and images. Shen Huang and Zhong-Gang Li performed and obtained results for progressive JPEG compression and Peter Monnes analysed the JPEG performance.

Special thanks for typing, formatting, and finalizing this paper goes to Donna Rubinoff, whose expertize and dedication were invaluable in the completion of this work.

References

1. Fox, E.A. (1991) Advances in interactive digital multimedia systems. *IEEE Computer*, **24**: 9–21.
2. Furht, B. (1994) Multimedia systems: an overview. *IEEE Multimedia*, **1**: 47–59.
3. Pennenbaker, W.B. & Mitchell, J.L. (1993) *JPEG still image Data Compression Standard.* New York, Van Nostrand Reinhold.
4. Poynton, C.A. (1994) High definition television and desktop computing. In Koegel Buford, J.F. *Multimedia Systems*, ACM Press.
5. Wallace, G. (1991) The JPEG still picture compression standard. *Communication of the ACM*, **34**: 30–44.
6. Nelson, M. (1992) *The Data Compression Book.* San Mateo, CA, M&T Books.
7. Kessler, M., Alexander, J. & Furht, B. JPEG Still Image Compression Algorithm: Analysis and Implementation, *Technical Report TR-CSE-94-07*, Florida Atlantic University, Boca Raton, FL, March 1994.
8. Steinmetz, R. (1994) Data compression in multimedia computing—standards and systems, Part I and II. *J. Multimedia Syst*, **1**: 166–172 and 187–204.
9. Monnes, P. & Furht, B. Parallel JPEG Algorithms for Still Image Compression, *Proceedings of Southeastcon '94*, Miami, Florida, April 1994, 375–379.
10. Aravind, R., Cash, G.L., Duttweller, D.C., Hang, H.-M., Haskel, B.G. & Puri, A. (1993) Image and video coding standards. *AT&T Technical Journal*, **72**: 67–88.