Imperial College of Science Technology and Medicine

Department of Electrical and Electronic Engineering

Digital Image Processing

PART 4

IMAGE COMPRESSION

LOSSY COMPRESSION

NOT EXAMINABLE MATERIAL

Academic responsible

Dr. Tania STATHAKI

Room 811b

Ext. 46229

Email: t.stathaki@ic.ac.uk

1 INTRODUCTION

Lossy compression of images deals with compression processes where decompression yields an imperfect reconstruction of the original image data. A wide range of lossy compression methods has been developed for compressing still image data. In this section we describe some of the basic concepts of lossy compression that have been adopted in practice and that form the basis of the image and video compression standards.

As already discussed, the selection of a particular compression method involves many tradeoffs. However, regardless of the compression method that being used, **given the level of image loss (or distortion), there is always a bound on the minimum bit rate of the compressed bit stream**. The class of theoretical methods that attempt to relate the allowed distortion of the original signal with the minimum bit rate constitute what is referred to as Rate-Distortion or Distortion-Rate theory.

2 RATE-DISTORTION THEORY

At various points in their study of the image coding literature readers will come across with the application of rate-distortion $R(D)$ theory, or its equivalent, distortion-rate $D(R)$ theory. The significant introduction by Shannon (1959) of a fidelity criterion into his work on coding theory led to an extensive body of work which sought to characterize the relationship between **coding rate** and **measured distortion** in a consistent way.

Briefly, the general form of the $R(D)$ curve is as shown in the following figure where, as expected, for smaller levels of distortion we require a higher coding rate. If the attainable level of distortion is no smaller than $D_{\max}$ then no information need be sent anyway.

For an initially analogue input signal, as the distortion falls to zero the quantisation intervals must have a width which tends to zero and so the rate curve moves towards infinity (dotted curve in Figure 2.3). For a discrete signal we know that we can encode at a rate equivalent to the entropy and incur zero distortion, at least in principle $[R(0) = H]$. We may therefore set our operating point anywhere within the region $0 \le D \le D_{\max}$. Deviations between the results achieved with practical algorithms and the theory are to be expected, and are usually due to lack of knowledge of the true source distribution (and the associated $R(D)$ relation) and an inability to allocate fractional numbers of bits to encoded symbols (some kinds of block schemes, vector quantisation, for example, allow this to be done).

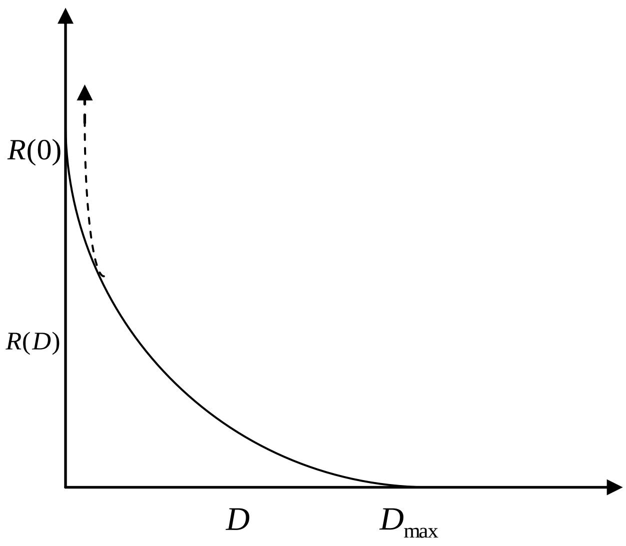


Figure 2.1: Rate-distortion $R(D)$ relationship. For a discrete signal zero distortion coding is achieved when $R(0) =$ the source entropy. For a continuous source zero distortion implies that the rate rises without limit (---).

3 BASIC CODING SCHEMES FOR LOSSY COMPRESSION

There are two classes of lossy compression schemes for images: sample-based coding and block-based coding.

Sample-Based Coding

In sample-based coding, the image samples are compressed on a sample by-sample basis. The samples can be either in the spatial domain or in the frequency domain. A simple sample-based compression method is a scalar predictive coder such as the differential pulse code modulation (DPCM) method shown in Figure 3.1. This scheme is very similar to the one used in lossless JPEG, except that in lossless JPEG there is no quantization. For each input sample x_{ij}, a residual signal e_{ij} is formed using the prediction signal P_{ij}. P_{ij} is typically a weighted sum of previously decoded pixels near x_{ij}. If the image is highly correlated, P_{ij} will track x_{ij}, and e_{ij} will consequently be quite small. A typical uniform quantization function is shown in Figure 3.1b. The quantizer maps several of its inputs into a single output. This process is irreversible and is the main cause of information loss.
Since the variance of e_{ij} is lower than the variance of x_{ij} quantizing e_{ij} will not introduce significant distortion. Furthermore, the lower variance corresponds to lower entropy and thus to higher compression. For typical images, the covariance function decays rapidly at distances longer than eight pixels. (The covariance function is a strong indication of the degree of dependency between

neighboring pixels). This implies that there is no benefit in using more than an eight-pixel neighborhood when forming P_{ij}. Practical DPCM systems use three previously decoded pixels near x_{ij}. For example,

$$P_{ij} = w_1 \hat{x}_{ij-1} + w_2 \hat{x}_{i-1j-1} + w_3 \hat{x}_{i-1j}$$

where w_1, w_2, w_3 are constant weights and

$$\hat{x}_{ij} = P_{ij} + q_{ij}$$

denotes the decoder's estimate for x_{ij}.

If x_{ij} represents 8-bit pixels then e_{ij} will be in the range [-255, 255]. The quantizer remaps e_{ij} into the quantized data q_{ij}, which may occupy the same dynamic range as e_{ij}, but require fewer bits in their representation. For example, if there are only 16 quantization levels, then the quantized data can be represented by only four bits.

DPCM coders do not yield performance close to the $R(D)$ bound that we described in the previous section. This motivates the use of other pre-processing techniques that are based on coding blocks of samples.

Block-Based Coding

From rate-distortion theory, the source-coding theorem (refer to Image Compression: Part I) shows that as the size of the coding block increases, we can find a coding scheme of rate $R > R(D)$ whose distortion is arbitrarily close to D. In other words, if a source is coded as an infinitely large block, then it is possible to find a block-coding scheme with rate $R(D)$ that can achieve distortion D. This implies that a sample-based compression method, such as DPCM, cannot attain the $R(D)$ bound and that block-based coding schemes yield better compression ratios for the same level of distortion. Many compression techniques have been developed to operate in a blockwise manner. These techniques fall into one of two classes: spatial domain block coding and transform-domain block coding.

In spatial-domain block coding, the pixels are grouped into blocks and the blocks are then compressed in the spatial domain. Vector quantization based methods fall into this category. Vector quantization (VQ) methods require far more processing in the encoder than in the decoder. From a rate-distortion viewpoint, for the same rate, such coders yield images with at least 3dB better SNR than DPCM methods.

In transform-domain block coding, the pixels are grouped into blocks and the blocks are then transformed to another domain, such as the frequency domain.

The motivation for transforming the original image X into a transformed image Y is to obtain in Y

a more compact representation of the data in X. Lossy transform based compression methods achieve compression by first performing the transformation from X to Y and then by discarding the less important information in Y.
Some of the most commonly used transforms are the Discrete Fourier Transform (DFT), the Discrete Cosine Transform (DCT), the Discrete Sine Transform (DST), the Discrete Hadamard Transform (DHT), and the Karhunen-Loeve Transform (KLT).

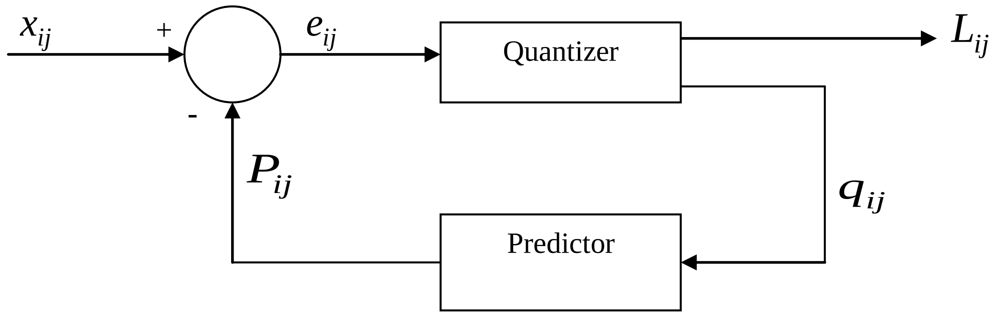


Figure 3.1a: A generic diagram of a scalar predictive coder

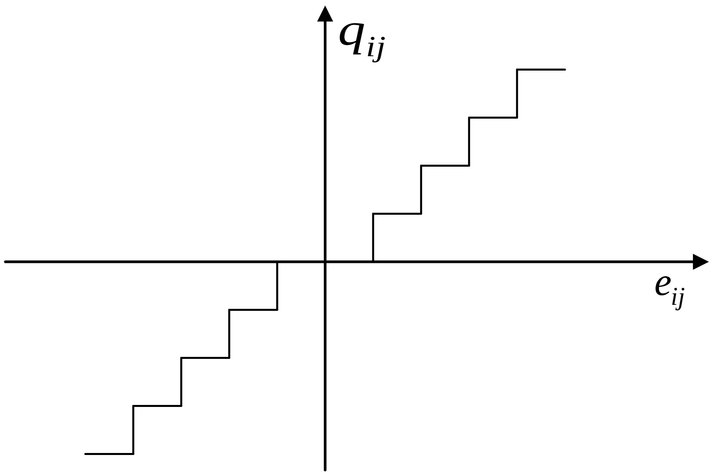


Figure 3.1b: Typical uniform quantization function

4 DCT-BASED CODING

DCT based image coding is the basis for all the image and video compression standards. The basic computation in a DCT-based system is the transformation of an $N \times N$ image block from the spatial domain to the DCT domain. For the image compression standards, $N = 8$.
An 8×8 block size is chosen for several reasons. From a hardware or software implementation viewpoint, an 8×8 block size does not impose significant memory requirements; furthermore the computational complexity of an 8×8 DCT is manageable on most computing platforms. From a

compaction efficiency viewpoint, a block size larger than 8×8 does not offer significantly better compression; this is attributable to the drop-off in spatial correlation when a pixel neighborhood is larger than eight pixels.
The advantages of DCT are already mentioned in Image Transforms.

A Generic DCT-Based Image Coding System

Figure 4.1 shows the key functional blocks in a generic DCT based image coding system. In the encoder, the DCT process transforms each 8×8 block X into a set of DCT coefficients Y. In the lossy compression mode, some of the weights will be deleted and thus the corresponding waveforms will not be used during decompression. The process of deleting some of the weights is referred to as the quantization process in Figure 4.1. The quantization process is an irreversible process and is the only source of loss in a DCT coding scheme. Strictly speaking, even with no quantization, there may be additional losses related to the implementation accuracy of the DCT and IDCT. Furthermore, in a transmission application, there may be additional losses due to noise in the transmission link. After quantization, the nonzero quantized values are then compressed in a lossless manner using an entropy coder. In most applications, the entropy coder combines a run/length coder with a Huffman coder.

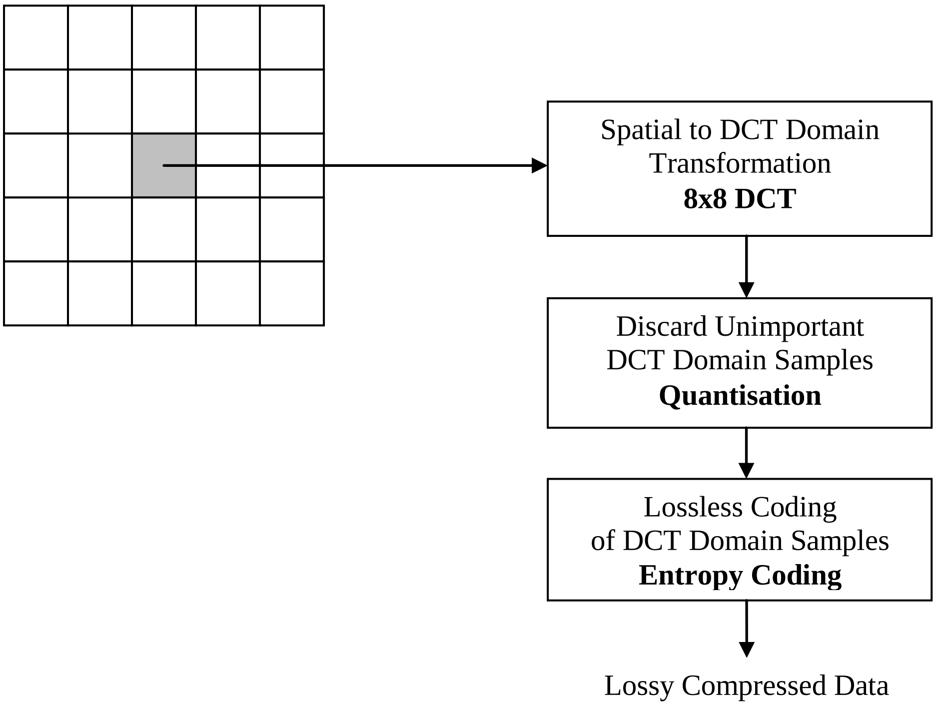


Figure 4.1: A Generic DCT-Based Image Lossy Coding System

DCT Based Coding Examples

Example 1

The DCT coding for a typical image is shown in Figure 4.2.

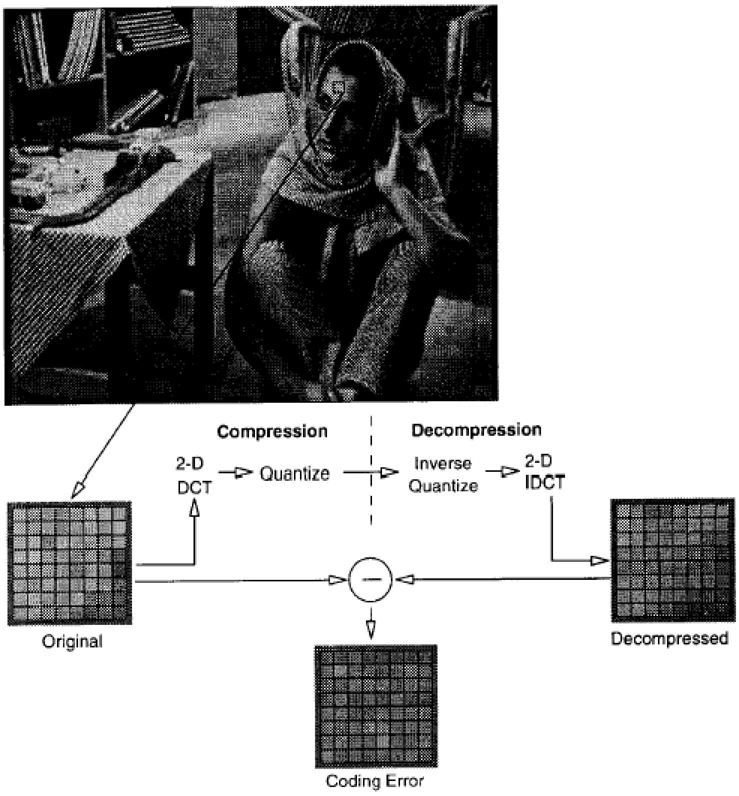


Figure 4.2: DCT coding-an example

Here, the input 8×8 block (labeled **original**) is taken from a low activity region; that is there are very small differences among pixel values in that area. The pixel values for this block are given by

$$X = \begin{bmatrix} 168 & 161 & 161 & 150 & 154 & 168 & 164 & 154 \\ 171 & 154 & 161 & 150 & 157 & 171 & 150 & 164 \\ 171 & 168 & 147 & 164 & 164 & 161 & 143 & 154 \\ 164 & 171 & 154 & 161 & 157 & 157 & 147 & 132 \\ 161 & 161 & 157 & 154 & 143 & 161 & 154 & 132 \\ 164 & 161 & 161 & 154 & 150 & 157 & 154 & 140 \\ 161 & 168 & 157 & 154 & 161 & 140 & 140 & 132 \\ 154 & 161 & 157 & 150 & 140 & 132 & 136 & 128 \end{bmatrix}$$

Depending on the color space, image pixels of a color component may have zero or nonzero average values. For example, in the RGB color space, all color components have a mean value of 128 (assuming 8-bit pixels). However, in the YC_bC_r color space (look Appendix), the Y component has an average value of 128, but the chroma components have an average value of zero. For uniform processing, most standard DCT coders require that image pixels are preprocessed so that their expected mean value is zero. The subtracted (or added) bias is then added (or subtracted) back by the decoder after the inverse DCT. After subtracting 128 from each element of X, the 8×8 DCT output block is given by

$$Y=\begin{bmatrix} 214 & 49 & -3 & 20 & -10 & -1 & 1 & -6 \\ 34 & -25 & 11 & 13 & 5 & -1 & 15 & -6 \\ -6 & -4 & 8 & -9 & 3 & -3 & 5 & 10 \\ 8 & -10 & 4 & 4 & -15 & 10 & 6 & 6 \\ -12 & 5 & -1 & -2 & -15 & 9 & -5 & -1 \\ 5 & 9 & -8 & 3 & 4 & -7 & -14 & 2 \\ 2 & -2 & 3 & -1 & 1 & 2 & -3 & -4 \\ -1 & 1 & 0 & 2 & 3 & -2 & -4 & -2 \end{bmatrix}$$

At this point, no compression has been achieved. Note that, compared to X, the DCT-transformed data Y has large amplitudes clustered close to y_{00}, commonly referred to as the DC coefficient. In general, for a low-activity block, most of the high-amplitude data will be in the low-order coefficients, as is the case in this example. It is the process of quantization of y_{kl} which leads to compression in DCT domain and this is expressed as

$$z_{kl}=\mathrm{round}\left[\frac{y_{kl}}{q_{kl}}\right]=\left\lfloor\frac{y_{kl}\pm\left\lfloor\frac{q_{kl}}{2}\right\rfloor}{q_{kl}}\right\rfloor,\ k,l=0,1,\ldots,7 \tag{4.1}$$ where

q_{kl} denotes the $kl-\mathrm{th}$ element of an 8×8 quantization matrix Q. ($\lfloor x\rfloor$ denotes the largest integer smaller or equal to x.) In order to ensure that the same type of clipping is performed for either positive or negative valued y_{kl} in (4.1), if $y_{kl}\geq 0$, then the two terms in the nominator are added; otherwise they are subtracted. For this example, if the 8×8 quantization matrix is given by

$$Q=\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

then the quantized DCT output is given by

$$Z = \begin{bmatrix} 13 & 4 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & -2 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The process of quantization has resulted in the zeroing out of many of the DCT coefficients y_{kl}. The design of Q depends on psychovisual characteristics and compression-ratio considerations. All compression standards provide default values for Q. The quantized DCT domain representation, Z, has resulted in significant savings, since only 11 values are needed to represent Z compared to the 64 values needed to represent X; this represents a compression ratio of 5.8. The matrix Z can be efficiently represented using a combination of a run/length coding scheme and a Huffman coding scheme; we will describe the specifics of such an entropy coder later. Decompression begins with the entropy decoding of the coded bit stream. Since entropy coding is a lossless compression scheme, the decoder should be able to reconstruct an exact version of Z. Inverse quantization on Z is simply performed as

$$\hat{z}_{kl} = z_{kl} q_{kl} .$$

In this example, the inverse quantizer output matrix Z is

$$\hat{Z} = \begin{bmatrix} 208 & 44 & 0 & 16 & 0 & 0 & 0 & 0 \\ 36 & -24 & 14 & 19 & 0 & 0 & 0 & 0 \\ 0 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 14 & -17 & 0 & 0 & 0 & 0 & 0 & 0 \\ -18 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

After the inverse quantization, the decoder computes an 8×8 IDCT given by

$$\hat{X} = \begin{bmatrix} 171 & 160 & 149 & 149 & 158 & 166 & 166 & 162 \\ 174 & 164 & 155 & 154 & 160 & 164 & 161 & 156 \\ 171 & 164 & 157 & 156 & 158 & 158 & 151 & 145 \\ 161 & 157 & 154 & 154 & 155 & 151 & 144 & 137 \\ 156 & 155 & 155 & 156 & 156 & 152 & 145 & 140 \\ 159 & 160 & 160 & 160 & 157 & 153 & 148 & 145 \\ 161 & 161 & 160 & 156 & 150 & 144 & 141 & 139 \\ 159 & 158 & 155 & 148 & 139 & 132 & 129 & 128 \end{bmatrix}$$

Observe that $\hat{X} \neq X$ and that the only cause of this coding error is the quantization of the DCT coefficients. However, in this example, as evidenced in Figure 4.2, the coding error is not perceptually significant.

Example 2

To better illustrate the properties of the DCT, we include another example as depicted in Figure 4.3.

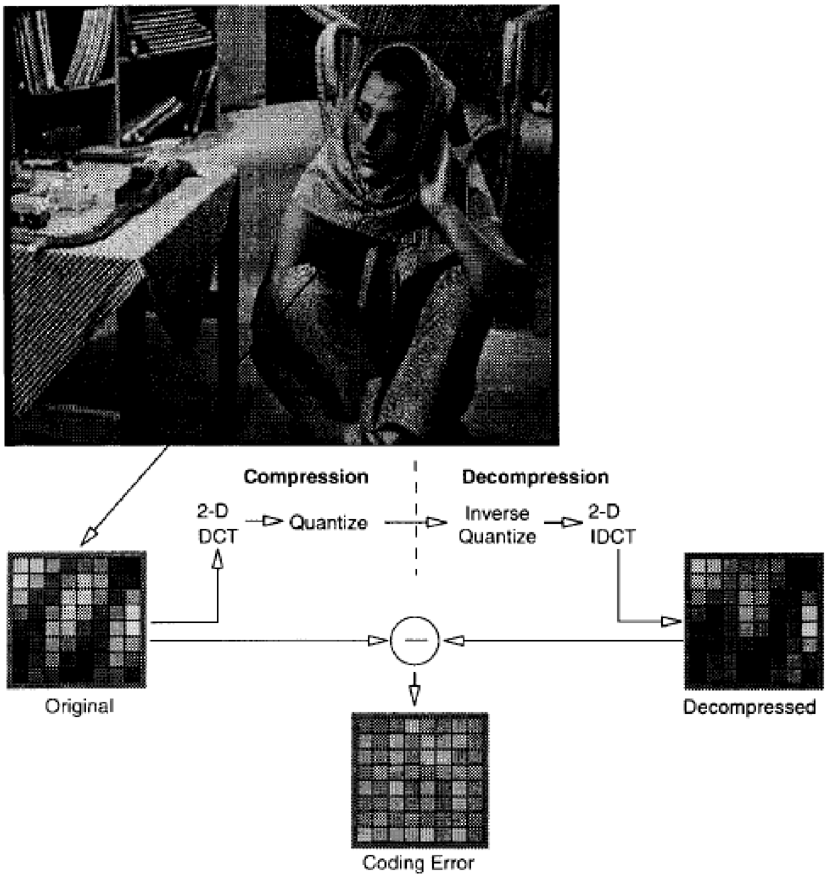


Figure 4.3: DCT coding-an example for a block in a high activity region

Here, we have chosen an 8×8 input block from a high-activity region. Zooming into this block, from Figure 4.3, we note that the block has two edge regions. The pixel values for this block are given by

$$X = \begin{bmatrix} 197 & 184 & 144 & 103 & 130 & 133 & 70 & 51 \\ 200 & 158 & 111 & 141 & 179 & 151 & 70 & 73 \\ 172 & 110 & 111 & 179 & 192 & 135 & 95 & 144 \\ 118 & 77 & 139 & 193 & 156 & 102 & 128 & 193 \\ 73 & 75 & 151 & 163 & 110 & 84 & 154 & 197 \\ 54 & 84 & 142 & 122 & 73 & 90 & 160 & 162 \\ 50 & 95 & 130 & 71 & 52 & 101 & 146 & 117 \\ 68 & 115 & 106 & 55 & 63 & 116 & 118 & 72 \end{bmatrix}$$

and the corresponding DCT output is

$$
Y = \begin{bmatrix}
-60 & -5 & -14 & -38 & 17 & 15 & 15 & 7 \\
127 & 139 & -40 & 103 & 102 & -41 & 12 & -13 \\
-76 & 123 & 22 & 110 & -105 & -46 & 1 & -8 \\
-20 & -5 & 29 & -53 & -54 & 18 & -1 & 11 \\
-4 & 4 & 5 & -25 & -6 & 4 & 2 & 5 \\
-3 & -10 & 9 & -19 & -5 & 8 & 7 & 6 \\
3 & 0 & 1 & 1 & 4 & 0 & -1 & -2 \\
-1 & -4 & 5 & -4 & -2 & -2 & 1 & 4
\end{bmatrix}
$$

Note that, compared with the Y output of the previous example, the dominant DCT values are not clustered close to y_{00}. This is usually the case when the spatial domain block is a high-activity block. For this example, we chose a quantization matrix that yields the same compression ratio as the one obtained in the previous example. Repeating the same calculations as in the previous example, the IDCT output is

$$
\hat{X} = \begin{bmatrix}
198 & 182 & 153 & 136 & 145 & 145 & 95 & 32 \\
182 & 159 & 146 & 153 & 152 & 129 & 98 & 81 \\
153 & 124 & 135 & 174 & 159 & 105 & 104 & 150 \\
120 & 95 & 125 & 180 & 153 & 86 & 112 & 203 \\
88 & 84 & 120 & 159 & 130 & 81 & 121 & 211 \\
62 & 93 & 120 & 114 & 92 & 92 & 131 & 173 \\
45 & 112 & 123 & 64 & 52 & 110 & 139 & 114 \\
37 & 127 & 126 & 31 & 27 & 123 & 143 & 72
\end{bmatrix}
$$

A comparison of the coding error images in Figure 4.2 and Figure 4.3 indicates that, for the same compression ratio, we have incurred more error for the high-activity block. However, from a perceptual viewpoint, the viewer may find the decompressed image block in Figure 4.3 to be the same as the original image block; this is because the eye is less sensitive to quantization noise on edges.

5 THE LOSSY JPEG STANDARD

Introduction

Until recently, the Group 3 and Group 4 standards for facsimile transmission were the only international standard methods for the compression of images. However, these standards deal only with bi-level images and do not address the problem of compressing continuous-tone color or grayscale images.

Since the mid 1980s, members from both the International Telecommunication Union (ITU) and the International Organization for Standardization (ISO) have been working together to establish a joint international standard for the compression of multilevel still images. This effort has been known as JPEG, the Joint Photographic Experts Group. Officially, JPEG corresponds to the ISO/IEC international standard 1091 8-I, *Digital compression and coding of continuous-tone still images*, or to the ITU-T Recommendation T. 81. The text in both these ISO and ITU-T documents is identical.

In recent years, there have been many new developments in the field of image compression. These include new compression schemes based on transform coding, vector quantization, sub-band filtering, wavelets, and fractals. The goal of JPEG has been to develop a general method for image compression that meets a number of diverse requirements, including the following:

- Be as close as possible to the state of the art in image compression.
- Allow applications (or a user) to tradeoff easily between desired compression and image quality.
- Work independently of the image type. That is, the method should not be restricted by the type of image source, image content, color spaces, dimensions, pixel resolution.
- Have modest computational complexity that would allow software-only implementations even on low-end computers. Low-complexity hardware implementations should also be feasible.
- Allow both sequential (single scan) and progressive coding (multiple scans).
- Offer the option for hierarchical encoding, in which a low-resolution version of the image can be accessed without a need to decompress the image at full resolution.

After evaluating a number of coding schemes, **the JPEG members selected a DCT-based method in 1988**. From 1988 to 1990, the JPEG group continued its work by simulating, testing, and documenting the algorithm. JPEG became a draft international standard in 1991 and an international standard in 1992. The JPEG group has not yet completed its mission. It continues to work on future enhancements. ISO 10918-3, which specifies recent extensions to JPEG, has already been approved as a draft international standard.

JPEG includes two basic compression methods: a DCT-based lossy compression method and a predictive method for lossless compression. We have already examined the lossless compression method earlier. We provide here an overview of the lossy JPEG method.

DCT-Based Coding

The JPEG standard specifies four modes of operation: sequential DCT-based, progressive DCT-based, lossless, and hierarchical. Under the lossless mode, a predictive coder followed by either a Huffman or an arithmetic coder is used instead of a DCT-based scheme. The details of operation under the lossless mode were already discussed. To better understand the other modes of operation we need first to review the DCT-based coder.

JPEG Encoding

The following Figure 5.1 shows a block diagram of the DCT-based JPEG encoder for an image with a single color component (grayscale).

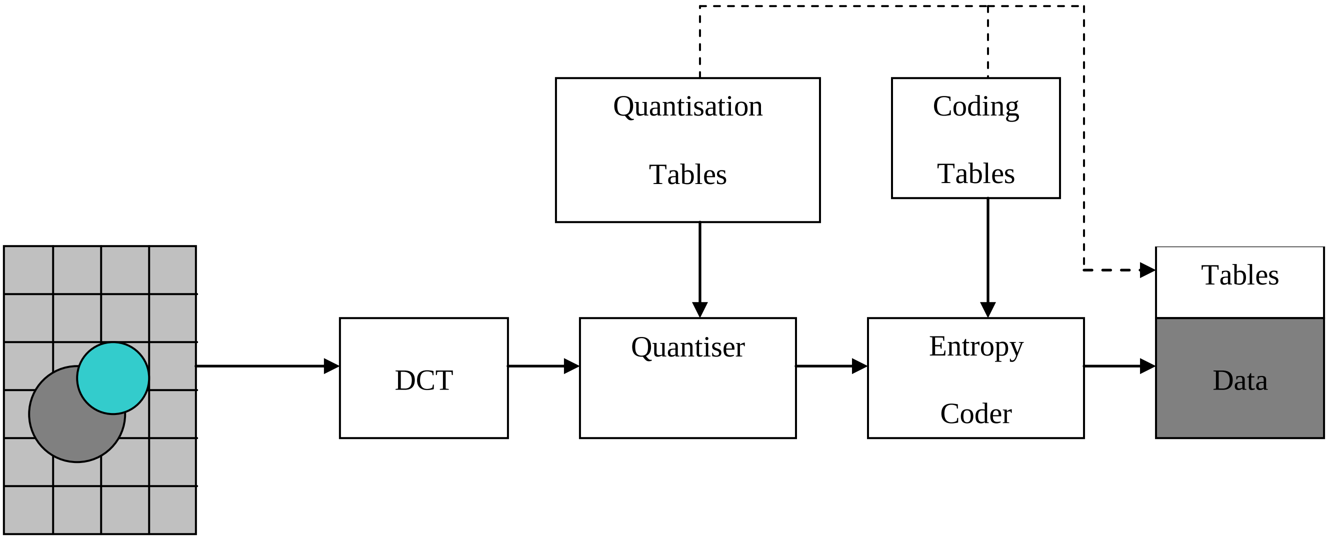


Figure 5.1: Block diagram of a JPEG encoder

For color images, the process is repeated for each of the color components. From Figure 5.1, the image is first divided into non-overlapping blocks. Each block has 8×8 pixels. If any of the dimensions of the image is not a multiple of eight, then the pixels of the last row or the last column in the image are duplicated appropriately.

Each block is transformed into the frequency domain by a 2-D DCT. The standard does not specify a unique DCT algorithm. Consequently, users may choose the algorithm that is best suited for their applications. The DCT output coefficients are quantized and entropy coded.

The entropy coder consists of two stages. The first stage is either a predictive coder for the DC (or [0,0]) coefficients or a run/length coder for the AC coefficients. The second stage is either a Huffman coder or an arithmetic coder. Arithmetic coding provides better compression than Huffman coding; however, there are very few JPEG implementations that support arithmetic coding. There are three main reasons for this. First, the improvement in compression (2 percent to 10 percent) does not justify the additional complexity (especially for hardware implementations). Second, many of the algorithms on arithmetic coding are covered by patents in the United States and Japan. Therefore, most implementors are reluctant to pay license fees for minimal gains in performance. Third, the baseline implementation, that is, the implementation with the minimum set of requirements for a JPEG compliant decoder, uses only Huffman coding. In order to facilitate the acceptance of JPEG as an international standard and because of the various options available during JPEG encoding, the JPEG committee also defined an interchange format. This interchange format embeds image and coding parameters (type of compression, coding tables, quantization tables, image size, etc.) within the compressed bit stream. This allows JPEG compressed bit streams to be interchanged among different platforms and to be decompressed without any ambiguity.

JPEG Decoding.

Figure 5.2 shows a block diagram of a JPEG decoder. After extracting the coding and the quantization tables from the compressed bit stream, the compressed data passes through an entropy decoder. The DCT coefficients are first dequantized and then translated to the spatial domain via a 2-D inverse DCT.

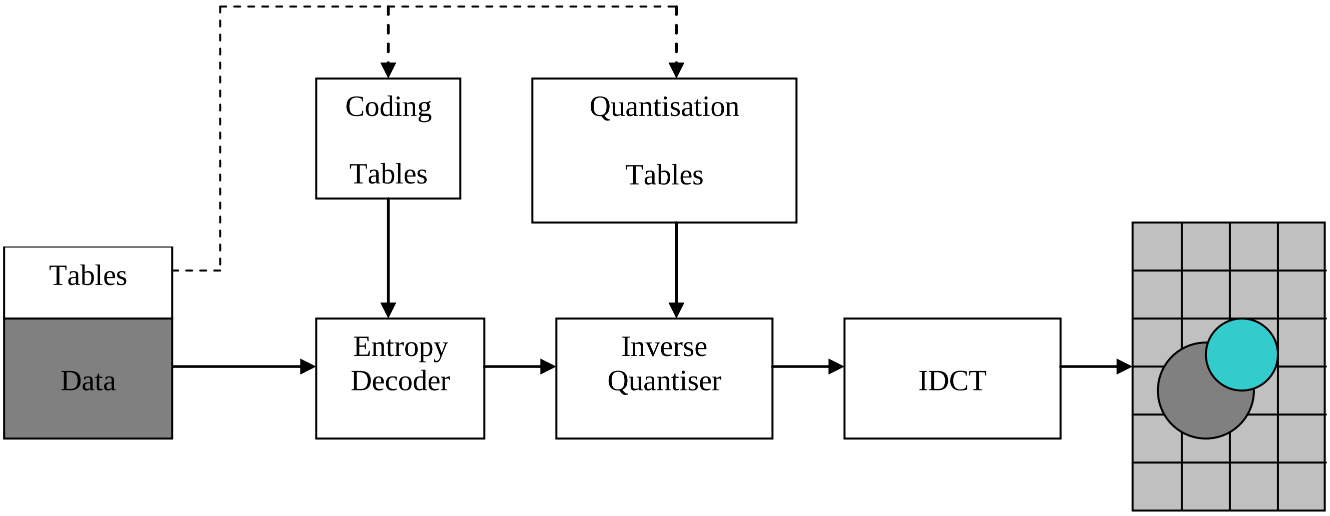


Figure 5.2: Block diagram of a JPEG decoder

The **Processing of Color Images** and the various methods for the **Design of Quantization Tables** are out of the scope of this course.

Entropy Coding

We already studied in detail the first two parts of a lossy compression scheme, namely, the DCT followed by quantization. The last processing block in JPEG is the entropy coder. This block improves overall performance by performing lossless coding on the quantized DCT coefficients. The entropy coder employed in the JPEG standard is not a straightforward implementation of the Huffman or arithmetic coding methods already described; instead, the quantized data are preprocessed by a run/length coder whose operation will be described later in this section. If the entropy coder employs Huffman coding, then one or more sets of Huffman tables need to be specified by the application. There are no default tables, but most applications use the Huffman tables listed in the standard. JPEG imposes only two restrictions on the Huffman tables: (1) no codeword may exceed 16 bits and (2) no codeword may be the all ones sequence (that is, FF_{16}). The arithmetic coding option in JPEG requires no external table specifications since it is able to adapt to the image characteristics. However, for improved performance, optional statistical tables can he used.

The baseline JPEG implementation uses Huffman coding only. Details for the baseline Huffman coder are presented next.

Huffman Coding of the DC Coefficients

Figure 5.3 shows a block diagram of the Huffman coder in baseline JPEG. Let DC_i and DC_{i-1} denote the DC coefficients of blocks i and $i-1$. Due to the high correlation of DC values among adjacent blocks, JPEG uses differential coding for the DC coefficients. For 8-bit-per-pixel data, DC differentials $(DC_i - DC_{i-1})$ can take values in the range [-2047, 2047]. This range is divided into 12 size categories, where the $i-\text{th}$ category includes all differentials that can be represented by i bits. The entries for these categories are the same as the first 12 categories in Table 4.2 (Image Compression: Part II). Thus, after a table lookup, each DC differential can be described by the pair (***size***, ***amplitude***) where ***size*** defines the number of bits required to represent the amplitude and ***amplitude*** is simply the amplitude of the differential. Given a DC residual value, its amplitude is computed as follows: if the residual is positive, then the amplitude is simply its binary representation with ***size*** bits of precision; and if the residual is negative, then we take the one's complement of its absolute value. From this pair of values, only the first (the ***size***) is Huffman coded.

For example, if the DC differential has an amplitude of 195, then from Table 4.2, size=8. Thus, 195 is described by the pair (8,11000011). If the Huffman codeword for size=8 is 111110, then 195 is coded as 11111011000011. Similarly, -195 would he coded as 11111000111100. Huffman decoding is quite simple. From the input bit stream, we first decode the size=8 information. Then, the next eight bits in the input bit stream directly give the amplitude of the DC differential, which we decode according to the value of its most significant bit.

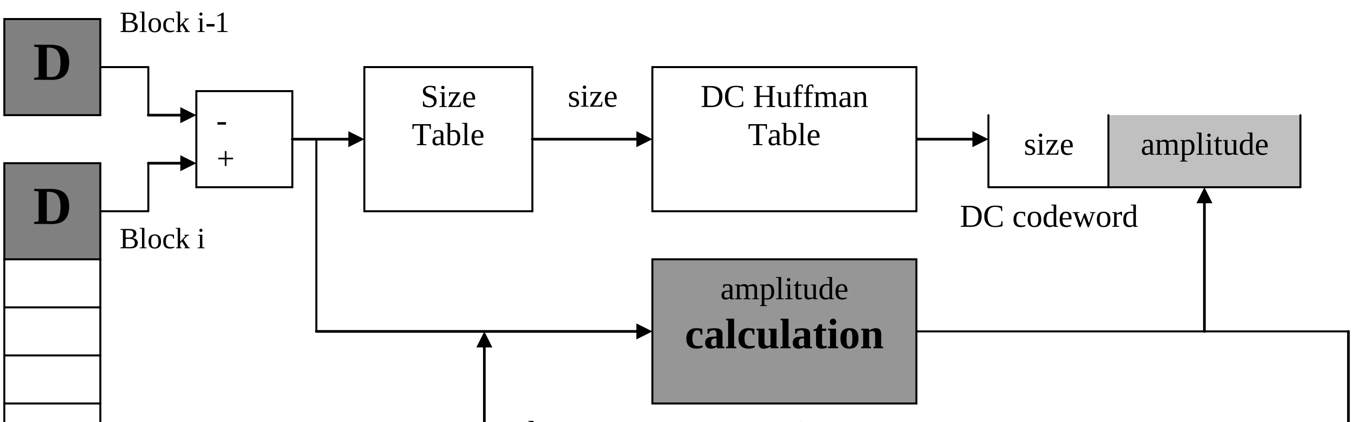


Figure 5.3: Huffman coding in baseline JPEG.

Huffman Coding of the AC Coefficients

For 8-bit pixels, AC coefficients make take any value in the range [-1023, 1023]. As before, this range is divided into 10 size categories. However, after quantization, most of the AC coefficients will be zero; thus, only the nonzero AC coefficients need to be coded. AC coefficients are processed in zig-zag order. Figure 5.4 shows the conventional and the zig-zag ordering of elements in an 8×8 matrix. Zig-zag ordering allows for a more efficient operation of the run/length coder.

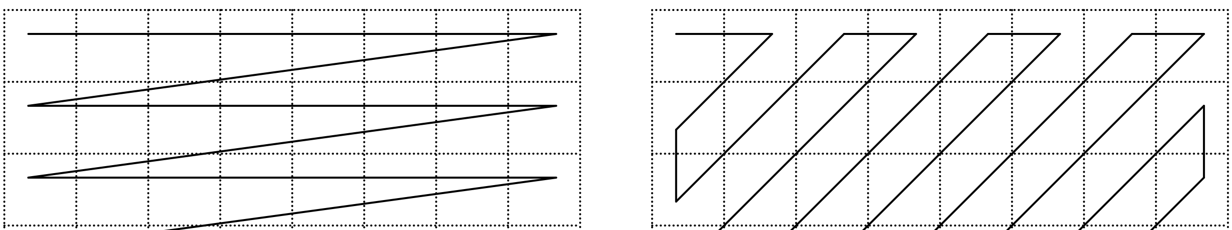
A run/length coder yields the value (***amplitude***) of the next nonzero AC coefficient and a ***run***. The run is the number of zero AC coefficients preceding this one. The number of bits required for the amplitude is again represented by the ***size*** (***length***). Hence, each nonzero AC coefficient can be described by the pair (***run***/***size***, ***amplitude***). The value of run/size is Huffman coded, and the value of the amplitude is appended to that code.

For example, assume an AC coefficient is preceded by six zeros and has a value of -18. From Table 4.2, -18 falls into category 5. The one's complement of -18 is 01101. Hence, this coefficient is represented by (6/5, 01101). The pair (6/5) is Huffman coded, and the 5-bit value of-18 is appended to that code. If the Huffman codeword for (6/5) is 1101, then the codeword for –18 is 110101101.

There are two special cases in the coding of AC coefficients as follows:

(1) The run/length value may be larger than 15. In that case, JPEG uses the symbol (15/0) to denote a run/length of 15 zeros followed by a zero. Such symbols can be cascaded as needed: however, the codeword for the last AC coefficient must have a non zero amplitude.

(2) If after a nonzero AC value all the remaining coefficients are zero, then the special symbol (0/0) denotes an end of block (EOB).



(a) Conventionl order (b) Zig-zag order

Figure 5.4: Conventional and zig-zag ordering in an 8×8 matrix

A Coding Example: Assume that the values of a quantized DCT matrix are given (in zig-zag order) by

$$
\begin{array}{rrrrrrrr}
42 & 16 & -21 & 10 & -15 & 0 & 0 & 0 \\
3 & -2 & 0 & 2 & -3 & 0 & 0 & 0 \\
0 & 0 & 2 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

If the DC value of the previous block is 40, then $DC_i - DC_{i-1} = 2$. This can be expressed as the (size, amplitude) pair (2, 2). If the Huffman codeword for size 2 is 011, then the codeword for the DC value is 01110.

Table 5.1 shows the codewords for the AC values. For Huffman codewords, we use tables taken from the JPEG standard. For this example, we need 82 bits to encode the AC coefficients and five bits to encode the DC coefficient, for a total of 87 bits or an average bit rate of $\frac{87}{64} = 1.36$ bits per pixel. If the input resolution was eight bits per pixel, then the compression ratio is $\frac{8}{1.36} = 5.88$.

Value	**Run/Size**	**Huffman Code**	**Amplitude**	**Total Bits**
16	0/5	11010	10000	10
-21	0/5	11010	01010	10

10	0/4	1011	1010	8
-15	0/4	1011	0000	8
3	3/2	111110111	11	11
-2	0/2	01	01	4
2	1/2	11011	10	7
-3	0/2	01	00	4
2	5/2	11111110111	10	13
-1	0/1	00	0	3
EOB	0/0	1010		4

Table 5.1: Example for the Huffman coding of AC coefficients

Compression Efficiency of Entropy Coding in JPEG

From the discussion on entropy coding of the DC and AC coefficients, we note that the entropy coder employed in the JPEG standard is not a straightforward implementation of the Huffman or arithmetic coding methods. In JPEG, Huffman or arithmetic coding is preceded by a run/length coder. Furthermore, entropy coding in the JPEG standard includes the following features:

1. The DC and AC coefficients are treated separately. This is motivated by the fact that the statistics for the DC and AC coefficients are quite dissimilar; hence, better coding efficiencies can be obtained using different Huffman tables.

2. For typical values of the quality factor q many of the AC coefficients within an 8×8 block will be zero-valued. Zig-zag scanning of the AC coefficients leads to an efficient (run/length, value) representation for the nonzero AC coefficients. Note that the quality factor is a scalar that is used to scale uniformly the original quantization table. The choice of the quality factor has been studied thoroughly in the design of quantization tables.

3. Values for the DC differentials range between -2047 and 2047, and for the AC coefficients range between -1023 and 1023. Direct Huffman coding of these values would require code tables with 4,095 and 2,047 entries, respectively. By Huffman coding only the ***size*** or the (***run/size***) information, the size of these tables is reduced to 12 and 162 entries, respectively.

To illustrate the benefits of run/length coding, Figure 5.5 shows for a typical grayscale image the output bit rate with and without a run/length coder. The top plot shows the output bit rate when an ideal Huffman or arithmetic coder is applied directly to the output of the DCT quantizer. The bottom plot shows the output bit rate when the ideal Huffman or arithmetic coder is preceded by a run/length coder. Bit rates are measured for various settings of the quality factor used to scale the quantization table. For a quality factor of one, the bit rate with a run/length coder is nearly four bits per pixel lower than the bit rate of an entropy coder alone. This is largely attributable to the efficient run/length representation of the zig-zag ordered AC coefficients. As the quality factor increases, more of the quantized AC values will be zero, and as expected the benefits from a run/length coder are even higher.

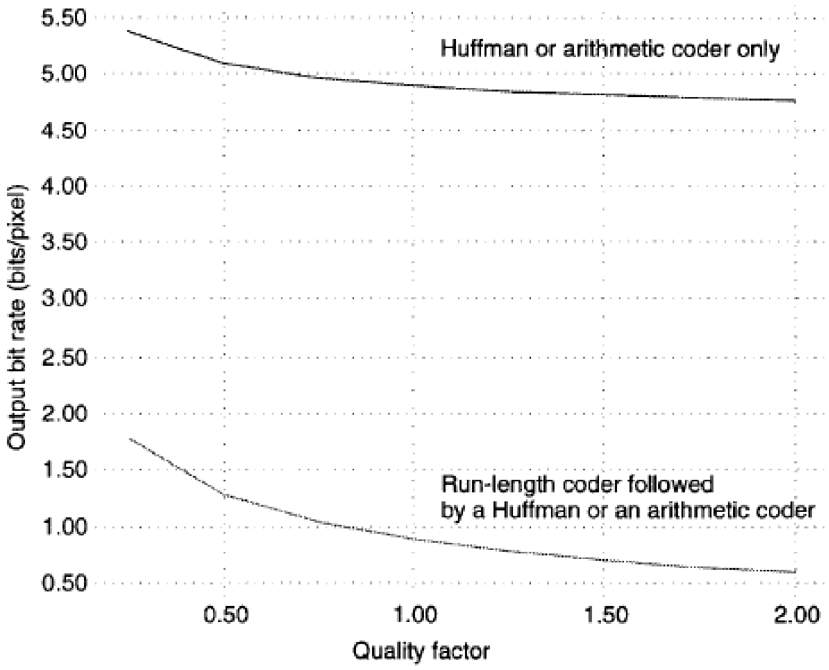


Figure 5.5: Effects of run/length coding on data compression

6 JPEG MODES OF OPERATION

As mentioned before, in addition to the lossless mode of operation, JPEG defines the following other modes: sequential, progressive, and hierarchical.

Sequential Coding

Sequential coding is the most common mode of operation. Image blocks are coded in a scan-like sequence, from left to right and from top to bottom. Transformed and encoded blocks can be transmitted before the end of the image. Similarly, the decoder may begin sequential decoding before it receives the complete compressed image. Figure 6.1 shows an example of sequential coding.

Figure 6.1: Example of sequential coding

Progressive Coding

In progressive mode, image blocks are also processed sequentially, but the coding is completed in multiple scans. The first scan yields the full image but without all the details, which are provided in successive scans. This mode requires that the output of the DCT is buffered so that during each scan only partial information from the DCT coefficients is encoded. Progressive coding allows a user to preview a rough version of an image and decode the additional information only if necessary. There are two procedures that are allowed for progressive coding: *spectral selection* and *successive approximation*.

Consider 8×8 blocks of quantized DCT coefficients as shown in Figure 6.2. We view each block as a three-dimensional (3-D) object, where depth denotes the arithmetic precision of the quantized coefficients. Under spectral selection, each block is divided into frequency bands, and each band is transmitted during a different scan. For example, in Figure 6.2a, the DCT output is divided into four scans.

Scan 1 includes the DC coefficient and the first two AC coefficients (counted in zig-zag order).

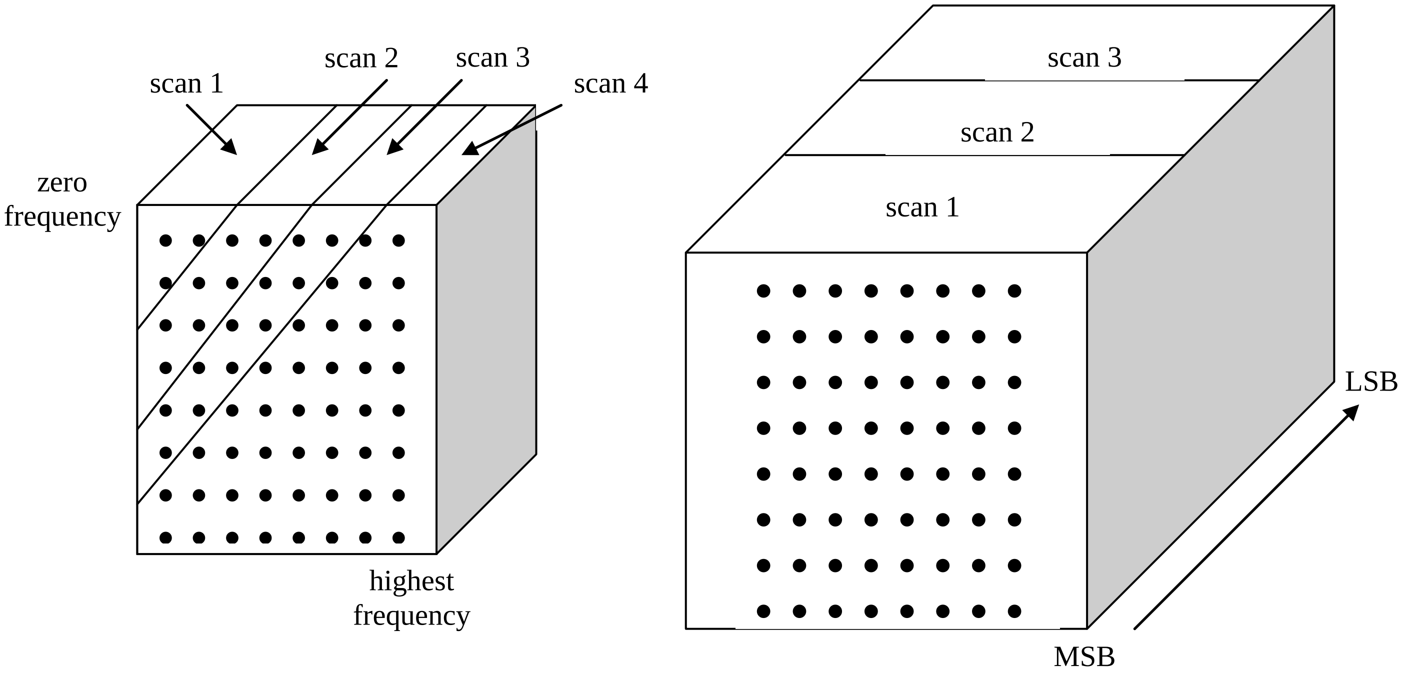
Scan 2 includes the next seven AC coefficients.

Scan 3 includes another 11 AC coefficients.

Scan 4 includes the remaining AC coefficients.

For most images, most of the information is contained in the DC and the first few AC coefficients. Thus, encoding and transmission of the first scan of coefficients will provide adequate information for a rough preview of the image. Encoding and transmission of the remaining scans just adds progressively additional detail.

Figure 6.3 shows an example of progressive coding based on spectral selection. Figure 6.3a shows the output image after decoding only the DC coefficients. The image is rather blocky, but we can still get a rough preview of the image. Figure 6.3b shows the output image after decoding the DC and the first three AC coefficients. The diagonal edges of the house are still blocky. Figure 6.3c shows the decoded image at full spectral resolution.



(a) Spectral selection

(b) Successive approximation

Figure 6.2: Description of progressive coding in JPEG

Under successive approximation, given a frequency band, the DCT coefficients are divided by a power of two before encoding. This scheme allows the encoder to transmit the most significant bits first, for a rough preview, and the least significant bits later, for decoding at full resolution. For example, in Figure 6.2b, the DCT output is encoded using three successive approximation scans. In the decoder, the coefficients are scaled back by the same power of two before computing the IDCT. The two progressive schemes may be combined or used separately.

Figure 6.3: Example of JPEG progressive coding using spectral selection

Hierarchical Coding

In hierarchical mode, each image component is encoded as a sequence of ***frames***. The first frame is usually a low-resolution version of the original image (possibly down sampled). Subsequent frames are differential frames between source components (possibly up sampled) and reference reconstructed components (possibly up sampled). Frames can be coded using either lossy JPEG or lossless JPEG. The two modes can be mixed only when lossless JPEG is used for the last stage of DCT-based hierarchical process. Hierarchical coding is useful when there are multiresolution requirements. For example, an application may support both high-resolution displays on workstations and low resolution displays on personal computers.

Figure 6.4 shows an example of a three level hierarchical coder. From the original image X, we generate two sub-sampled versions: X_2, where the image is sub-sampled by a factor of two on both dimensions; and X_4, where the image is sub-sampled by a factor of four on both dimensions. Note that, sub-sampling may also be preceded by a low-pass filtering operation to reduce aliasing effects. JPEG poses no requirements on these preprocessing operations.

The encoded image consists of three frames: S_1, S_2, and S_3. Frame S_1 is simply the X_4 image compressed. Using only S_1, the decoder can extract a low-resolution estimate of the original image $X_l^{'}$. S_2 (uncompressed) is the difference image between X_2 and an estimate of X_2 $(X_2^{'})$ after upsampling X_4 by a factor or two. Using S_1 and S_2, the decoder can extract a medium resolution estimate of the input X $(X_m^{'})$. Similarly, S_3 (uncompressed) is the difference image between X and an estimate of X based on X_2 and X_4. The reader can verify that, under lossless compression, $X^{'} = X$.

Figure 6.4: Three-level hierarchical coder

Figure 6.5 shows an example of hierarchical coding. The first image (a) is the original image shown at full resolution (200 dpi). Images (b) and (c) are sub sampled versions of the original (sub sampled by factors of two and four, respectively).

Figure 6.5: Example of hierarchical coding. (a) original; (b) subsanpled by a factor of two; (c) subsampled by a factor of four.

To summarize, the essential characteristics of the main JPEG coding processes are given as follows.

1. **Baseline Process**
   - Coding: DCT-based, sequential, one to four color components.
   - Resolution: eight bits per pixel.
   - Huffman coding; two AC and two DC tables.
2. **Extended DCT-based Process**
   - Coding: DCT-based, sequential or progressive.
   - Resolution: 8 or 12 bits per pixel.
   - Huffman coding or arithmetic coding; four AC and four DC tables.
3. **Lossless Process**
   - Coding: Predictive, sequential.
   - Resolution: From two bits per pixel to 16 bits per pixel. Huffman coding or arithmetic coding; four DC tables.
4. **Hierarchical Process**
   - Coding: DCT-based or lossless process.

- Multiple frames (non differential and differential).

JPEG has been developed for the compression of still-images; however, the proliferation of low-cost hardware for JPEG has led to the development of an additional mode of operation for video sequences: motion-JPEG. Under motion-JPEG, each frame of a video stream is compressed independently using the baseline JPEG algorithm. However, there is no standard syntax for motion-JPEG coded streams, and encoded data may not be able to be decoded across different platforms.

APPENDIX

YC_bC_r is a color co-ordinate system used for the transmission and storage of image and video signals. Given the primary RGB inputs in [0,1] then

$$Y = 0.299(R-G) + G + 0.114(B-G)$$

$$C_b = 0.564(B-Y)$$

$$C_r = 0.713(R-Y)$$

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$