Chapter 2
Data Coding and Image Compression

—The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.

Shannon, Claude

2.1 Introduction

The need for compression of images becomes apparent when one counts the number of bits needed to represent the information content within each image. Let us consider, for example, the storage space required by the following types of images:

- A color image of compact size, say 512×512 pixels at 8 bpp and 3 color channels requires about 6.3×10^6 bits (768 KB).
- A digitized at 12 μm color negative photo of 24×36 mm (35 mm), 3000×2000 pixels at 8 bpp, requires 144×10^6 bits (about 17.5 MB).
- A digitized at 70 μm radiogram of 11×17 inch, 5000×6000 pixels at 12 bpp, requires 360×10^6 bits (about 44 MB).
- A multispectral LANDSAT image of 6000×6000 pixels per spectral band at 8 bpp for 6 spectral bands, requires 1.73×10^9 bits (about 211 MB).

These examples, and those mentioned in the Preface indicate the need for a method to tackle with this massive amount of data. In the following paragraphs an introduction into the realm of data compression is given and approaches for image compression are presented.

G. Pavlidis, *Mixed Raster Content*, Signals and Communication Technology,
DOI 10.1007/978-981-10-2830-4_2

2.2 Quantification of Information

Since *Information Theory* can provide answers to two fundamental questions, "what is the ultimate data compression" (Entropy), and "what is the ultimate transmission rate of communication" (channel capacity) (Cover and Thomas 2006), questions which are also fundamental in the explorations in this treatise, it is important to include an introduction on these subjects. A need that naturally arises is to model and quantify 'information'. Fortunately, this issue has been addressed back one century ago. Hartley in his seminal paper in 1928 (Hartley 1928) realized that

> *...as commonly used, information is a very elastic term, and it will first be necessary to set up for it a more specific meaning...*

so his first care was about measuring information. Information communication was the context in this work. Hartley noticed that during the communication of a message, the precision of the information depends upon what other sequences might have been chosen. He realized that whenever new information became available towards completing the communicated message, ambiguity lowered, by excluding any other possibilities for the communicated meaning. Furthermore, by realizing that the symbols that comprise the information, in terms of quality and quantity, constitute an integral part of a process to measure the information, he first tried to eliminate undesirable side effects of this insight that are imposed by non-physical (like psychological) quantities. For example, the number of symbols available to two persons that speak different languages to communicate is negligible as compared to the available number of symbols to two persons speaking the same language. Hartley's approach was to propose that any physical system that transmits information should be indifferent to the content and interpretation and should focus on the ability of the system to distinguish the transmitted symbols.

Hartley assumed a system able to transmit s symbols by doing a number of n possible selections (arrangements of s symbols into sequences of n members). He deduced that the amount of information should be proportional to the number of selections n

$$H = Kn \tag{2.1}$$

where K a constant that depends on the number of symbols s available at each selection. By comparing two such systems that produce the same amount of possible sequences he found that

$$\left.\begin{array}{l} s_1^{n_1} = s_2^{n_2} \\ H = K_1 n_1 = K_2 n_2 \end{array}\right\} \Rightarrow \frac{K_1}{\log s_1} = \frac{K_2}{\log s_2} \tag{2.2}$$

where $s_k^{n_k}$ the number of possible distinguishable sequences of system k. Apparently this relation holds only if

$$K = K_0 \log s \tag{2.3}$$

with K_0 being the same for all systems, which may be omitted by making the base of the logarithm arbitrary. Then connecting (2.3) with (2.1),

$$H = n \log s \Rightarrow H = \log s^n \tag{2.4}$$

This is equivalent to stating that *a practical measure of information is the logarithm of the number of possible symbol sequences.* If for example $n = 1$ (that is there is a single selection available) then the amount of information is equal to the logarithm of the number of symbols. Hartley, by introducing a logarithmic measure of information for communication (the logarithm of the alphabet size), had hacked his ways into the world of information and defined the first means to quantify it.

Shannon took on the work by Hartley and build a complete *Mathematical Theory of Communication* (Shannon 1948). Shannon approved Hartley's choice of a logarithmic measure for various reasons and pointed out that the choice of the logarithmic base corresponds to the choice of a unit for measuring information. So, if the base is two (2) then information is measured in *binary digits, or bits* (suggested by J.W. Tukey). Shannon distinguished the systems into two major categories of study, the *Discrete Noiseless Systems* and the *Discrete Systems with Noise*. He defined the *Discrete Channel* as

> *the system in which a sequence of choices from a finite set of elementary symbols can be transmitted from one point to another, where each symbol is assumed to have a certain duration in time*

Shannon then proceeded to define the *Discrete Source* of information using mathematical formalism, by realizing that such a source would produce a message symbol by symbol according to certain probabilities depending on the particular symbols and preceding symbol choices. Apparently what this formulation describes is a *stochastic process*. Shannon turned it also the other way around and stated that

> *...any stochastic process that produces a discrete sequence of symbols chosen from a finite set may be considered a discrete source.*

This was a crucial step in defining a mathematical description of a source of information. Building on this, Shannon defined the various series of approximations, like the zero-order, first-order, etc., in symbols and symbol sequences, bringing the notions of *Markov processes*[1] into the new information theory. Shannon further restricted the domain of interest into those Markov processes which exhibit a means of statistical homogeneity. The special case of processes that correspond to this domain are the *ergodic processes*, which exhibit the property that any collection of random samples from an ergodic process must represent the average statistical properties of the entire process. Subsequently, *an ergodic source* produces sequences with the same statistical properties.

[1] Shannon used the older spelling 'Markoff' for the famous Russian mathematician Andrei Andreyevich Markov.

After representing the discrete information sources as ergodic Markov sources, Shannon searched for a quantity to measure the amount of information produced by such sources, which can also be the rate at which information is produced by these sources. Supposing a set of possible events with corresponding probabilities $p_1, p_2, ..., p_n$ he searched for a measure $H(p_1, p_2, ..., p_n)$ with the following properties:

- H should be continuous in the p_i.
- If all the p_i are equal, $p_i = \frac{1}{n}$, then H should be a monotonic increasing function of n.
- If a choice be broken down into two successive choices, the original H should be the weighted sum of the individual values of H.

By elaborating on these desired properties, Shannon concluded that this measure is

$$H = -K \sum_{i=1}^{n} p_i \log p_i \tag{2.5}$$

Shannon realized that quantities of the form

$$H = -\sum_{i=1}^{n} p_i \log p_i \tag{2.6}$$

play a central role in quantifying information and thus the name *entropy* was coined to describe them, as they are directly related to the same definition of entropy in statistical mechanics. According to this definition, the entropy of two random variables with probabilities p and $q = 1 - p$ is $H = -(p \log p + q \log q)$ which gives rise to the famous bell-shaped graph shown in Fig. 2.1.

This quantity, the entropy H, has some interesting properties that further support its usability as an information measure:

1. It is a positive value except for one single case, where a single event is certain, thus the entropy collapses to zero,

$$H = \begin{cases} = 0, & p_i = 0, \ \forall i, \ \text{except for one } p_j = 1 \\ > 0, & \text{otherwise} \end{cases} \tag{2.7}$$

2. The entropy is maximized for equally probable events, thus the most uncertain situation is successfully captured,

$$H = H_{max} = \log n, \qquad p_i = \frac{1}{n} \tag{2.8}$$

3. The entropy of a joint event of events x and y with $p(i, j)$ probability of joint occurrence of i for the first and j for the second is,

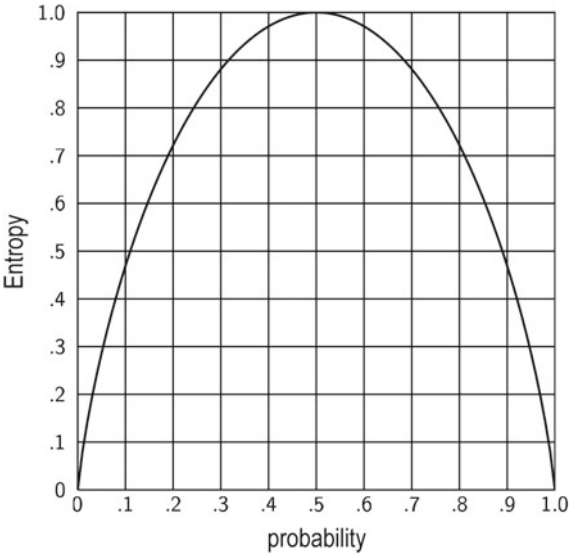

Fig. 2.1 Entropy of two random variables with probabilities p and $(1-p)$

$$H(x,y) = -\sum_{i,j} p(i,j)\ \log p(i,j)$$

$$\text{while } \begin{cases} H(x) = -\sum_{i,j} p(i,j)\ \log \sum_j p(i,j) \\ H(y) = -\sum_{i,j} p(i,j)\ \log \sum_i p(i,j) \end{cases} \tag{2.9}$$

and it is easily shown that,

$$H(x,y) \leq H(x) + H(y) \tag{2.10}$$

with equality holding for the case of the events being independent ($p(i,j) = p(i)p(j)$).

4. Any equalization of the probabilities $p_i, p_2, ..., p_n$ or averaging of the form,

$$p_i{}' = \sum_j a_{ij} p_j, \qquad \sum_i a_{ij} = \sum_j a_{ij} = 1,\ \forall a_{ij} \geq 0 \tag{2.11}$$

increases the entropy H.

5. Supposing two random events as in the 3rd case, the *conditional entropy* of y, $H_x(y)$ (modern notation for conditional entropy using the symbolism $H(y|x)$, but the original Shannon's notation is used here), is defined as the average of the entropy of y for each value of x, weighted according to the probability of getting that particular x,

$$H_x(y) = -\sum_{i,j} p(i,j)\ \log p_i(j) \tag{2.12}$$

where

$$p_i(j) = \frac{p(i,j)}{\sum_j p(i,j)} \tag{2.13}$$

is the conditional probability that y has the value j. Apparently, (2.12) quantifies the amount of uncertainty regarding y when x is known, on the average. Combining (2.12) with (2.13),

$$\begin{aligned} H_x(y) &= -\sum_{i,j} p(i,j)\ \log p(i,j) + \sum_{i,j} p(i,j)\ \log \sum_j p(i,j) \\ &= H(x,y) - H(x) \Leftrightarrow H(x,y) = H(x) + H_x(y) \end{aligned} \tag{2.14}$$

6. Combining the 3rd (2.10) and the 5th (2.14) properties,

$$H(x) + H(y) \geq H(x,y) = H(x) + H_x(y) \Leftrightarrow H(y) \geq H_x(y) \tag{2.15}$$

which indicates that the knowledge of x decreases the uncertainty of y, unless the events x and y are independent.

The theory of entropy expanded to describe the uncertainty in an information source, which produced two very significant theorems limiting the data rates that can be attained in a compression system. By defining,

$$G_N = -\frac{1}{N} \sum_i p(B_i)\ \log p(B_i) \tag{2.16}$$

$p(B_i)$ being the probability of a sequence B_i of source symbols, and the summation defined over all sequences containing N symbols. This G_N is a monotonic decreasing function of N and

$$\lim_{N \to \infty} G_N = H \tag{2.17}$$

This describes the zero-order case, where there is no dependence on previously produced symbols. In all other cases, where the probability of a sequence B_i being followed by a symbol s_j is denoted by $p(B_i, s_j)$, where also the conditional probability of s_j after B_i is $p_{B_i}(s_j) = \frac{p(B_i, s_j)}{p(B_i)}$,

$$F_N = -\sum_{i,j} p(B_i, s_j)\ \log p_{B_i}(s_j) \tag{2.18}$$

summing over all blocks B_i of $N-1$ symbols and over all symbols s_j. Then F_N is also a monotonic decreasing function of N, and by combining (2.16) with (2.18) holds that,

$$
\begin{aligned}
F_N &= NG_N - (N-1)G_{N-1} \\
G_N &= \frac{1}{N}\sum_{n=1}^{N} F_N \\
F_N &\leq G_N, \quad \lim_{N\to\infty} F_N = H
\end{aligned}
\tag{2.19}
$$

These very important results show that a series of approximations to H can be obtained by considering only the statistical structure of the sequences over the N symbols, and F_N is a better approximation, in fact being the N-th order approximation to the information source. F_N is the *conditional entropy* of the next symbol when knowing the $N-1$ preceding symbols. Shannon defined the ratio of the entropy of a source to the maximum value as the *relative entropy* of the source, identifying this as *the maximum compression possible*, and defined as *redundancy* the one minus the relative entropy.

At the time Shannon published this theory it was thought to be impossible to send information at a positive rate with negligible probability of error. Shannon proved that the probability of error could be made nearly zero for all communication rates below channel capacity. Shannon's *fundamental theorem for a noiseless channel*, today referenced as *the noiseless coding theorem*, stated that it is impossible to encode the output of a source in such a way as to transmit at an average rate less than or equal to $\frac{C}{H}$, where C the channel capacity (bits per second) and H the entropy of the source (bits per sample). In this theory Shannon defined the upper limit of any possible compression (the entropy H) for a random processes such as music, speech or images, by identifying that in such processes there is an irreducible complexity below which the signal cannot be compressed. Shannon argued that if the entropy of the source is less than the capacity of the channel, asymptotically error-free communication can be achieved (Cover and Thomas 2006).

Some typical examples of source entropy computations are as follows (bit-rates are reported typically in Bits Per Sample (bps)). Given $X = \{a, b, c, d, e\}$ with all symbols equiprobable ($p(a) = ... = p(e) = 1/5$), then the entropy of X would be,

$$
H(X) = -5 \times \frac{1}{5}\log_2\frac{1}{5} = 2.32\,\text{bps} \tag{2.20}
$$

and a typical message produced by such a source could be (Shannon 1948),

B D C B C E C C C A D C B D D A A E C E E A
A B B D A E E C A C E E B A E E C B C E A D

When the probabilities of the symbols change, say, $p(a) = 1/2.5$, $p(b) = p(e) = 1/10$, $p(c) = p(d) = 1/5$, then the entropy of X becomes

$$
\begin{aligned}
H(X) = -\frac{1}{2.5}\log_2\frac{1}{2.5} - \frac{1}{10}\log_2\frac{1}{10} - \frac{1}{5}\log_2\frac{1}{5} - \\
- \frac{1}{5}\log_2\frac{1}{5} - \frac{1}{10}\log_2\frac{1}{10} = 2.12\,\text{bps}
\end{aligned}
\tag{2.21}
$$

and a typical message produced by such a source could be (Shannon 1948),

A A A C D C B D C E A A D A D A C E D A
E A D C A B E D A D D C E C A A A A A D

Results change dramatically when joint and conditional probabilities are at work,

$$
\begin{aligned}
X &= \begin{cases} a & p(a) = 9/27 \\ b & p(b) = 16/27 \\ c & p(c) = 2/27 \end{cases} \\
p_i(j) &= \begin{pmatrix} p_a(a) = 0 & p_a(b) = 4/5 & p_a(c) = 1/5 \\ p_b(a) = 1/2 & p_b(b) = 1/2 & p_b(c) = 0 \\ p_c(a) = 1/2 & p_c(b) = 2/5 & p_c(c) = 1/10 \end{pmatrix} \\
p(i,j) &= \begin{pmatrix} p(a,a) = 0 & p(a,b) = 4/15 & p(a,c) = 1/15 \\ p(b,a) = 8/27 & p(b,b) = 8/27 & p(b,c) = 0 \\ p(c,a) = 1/27 & p(c,b) = 4/135 & p(c,c) = 1/135 \end{pmatrix}
\end{aligned}
\tag{2.22}
$$

a typical message produced by such a source would be (Shannon 1948),

A B B A B A B A B A B A B A B B B A B B
B B B A B A B A B A B A B B B A C A C A
B B A B B B B A B B A B A C B B B A B A

This scales up considerably when moving from a source of symbols to a source of more complicated output, say words. Furthermore, by imposing a second-order word approximation and using the total English alphabet, Shannon was able to produce sequences such as (Shannon 1948),

> THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED

which represents a sentence that could, actually, make some sense, even though Shannon's primary concern had to do with the decoupling of the message and the meaning.

When Shannon's attention switched to the case of channels with noise, then the focus had to be shifted from the compact representation of information to the reliable communication. Building on entropy as the measure of uncertainty, Shannon found it reasonable to use *the conditional entropy of the message, knowing the received signal*, as a measure of the missing information, which represents the amount of information needed at the received to correct the corrupted message,

$$R = R(x) - R_y(x) \tag{2.23}$$

R being the transmission rate, $R(x)$ the rate that corresponds to the $H(x)$ entropy of the source and $R_y(x)$ the rate that corresponds to the $H_y(x)$ conditional entropy of the

source given the message received at the receiver. In a typical example, supposing two possible symbols 0 and 1, and a transmission at a rate of 1000 symbols per second with probabilities $p_0 = p_1 = 1/2$, and assuming that during transmission the noise alters 1 in 100 received symbols, if a 0 is received the *a posteriori* probability that a 0 was transmitted is 0.99, and that a 1 was transmitted is 0.01. Hence,

$$H_y(x) = -[0.99 \log_2 0.99 + 0.01 \log_2 0.01] = 0.081 \text{ bits per symbol} \quad (2.24)$$

which corresponds to $R_y(x) = 81$ bits per second in this case. Then the system is transmitting at a rate $R = R(x) - R_y(x) = 1000 - 81 = 919$ bits per second. In the extreme case where a 0 is equally likely to be received as a 0 or 1 and similarly for 1, the a posteriori probabilities are $1/2$, $1/2$ and

$$H_y(x) = [0.5 \log_2 0.5 + 0.5 \log_2 0.5] = 1 \text{ bits per symbol} \quad (2.25)$$

which amounts to $R_y(x) = 1000$ bits per second. The rate of transmission is then $R = R(x) - R_y(x) = 1000 - 1000 = 0$ as expected. On this intuition, Shannon built his definition of the *capacity of a channel* as $C = max(H(x) - H_y(x))$ (with the 'max' in respect to all possible information sources used as input to the channel) and stated his *fundamental theorem for a discrete channel with noise*, today referenced also as *the noisy coding theorem*, which defines the limits for the rate of information that can be transmitted over a noisy channel keeping the error at a specific range. More formally, If the rate of the source is less than or equal to the capacity of the channel there exists a coding system such that the output of the source can be transmitted over the channel with an arbitrarily small frequency of errors.

A major achievement of the theory set out by Hartley and Shannon was the fundamental modeling of information as a probabilistic process, which can be measured in a manner that agrees with intuition. According to this modeling, a random event x with a probability of occurrence $p(x)$ is said to convey an amount of information content defined as (Gonzalez and Woods 1992)

$$I(x) = \log \frac{1}{p(x)} = -\log p(x) \quad (2.26)$$

where $I(x)$ is the *self-information* of x. Apparently, the self-information of an event is inversely related to the probability of its occurrence. So, for example, in the case of a certain event ($p(x) = 1$) there is no self-information ($I(x) = 0$) to attribute to this event. Intuitively, there is no meaning in transmitting a message about this event since its occurrence is certain. In any other case, (2.26) states that *the more surprising the event is, the more the information content it conveys*.

Other scientists built upon those theories or took their own path into the realm of information theory, such as Fisher who defined *the notion of sufficient statistic* (Fisher 1922) ($T(X)$ is sufficient relative to $\{f_\theta(x)\} \Leftrightarrow$ if $I(\theta; X) = I(\theta; T(X))$ for all distributions on θ), Lehmann and Scheffe who introduced the minimal sufficient statistic (Lehmann and Scheffe 1950), Kullback and Leibler who defined *the rela-*

tive entropy (or Kullback–Leibler distance, information divergence, cross entropy) (Kullback and Leibler 1951) ($D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$), and Fano who proved his *Fano's inequality*, giving a lower bound on the error probability of any decoder (Fano 1952) ($P_e = Pr\hat{X}(Y) \neq X \Rightarrow H(P_e) + P_e \log |\mathcal{X}| \geq H(X|Y)$, $\mathcal{X}$ denoting the support of X).

Any process that produces information can be considered as a source of symbol sequences, which are pre-selected from a finite set. The very text of this treatise, for example, is written using a source that includes all the American Standard Code for Information Interchange (ASCII) symbols (basically, Unicode symbols). Similarly, a computer performs calculations using binary data, which can be considered as sequences of symbols created by a source with a binary alphabet (0 and 1). Thus an n-bpp digital image is created by a source with an alphabet of 2^n symbols (which represents all possible values). The classification of terms of the sequence generated by a source-image can be based on the values of adjacent pixels resulting in a one-dimensional scanning (1-D raster scan), or may be based on values resulting from two-dimensional image regions (2-D blocks of pixels). The development of models for different input images makes it easier to measure the information carried by the symbol sequences (Rabbani and Jones 1991a). Let us consider a couple of simple examples from the image processing domain to see how the very basic theory of entropy applies in two-dimensional signals. Suppose there is a simple graylevel (or grayscale) image shown in Fig. 2.2.

This image comprises of the following pixel values

$$\begin{pmatrix} 40 & 40 & 40 & 80 & 80 & 160 & 200 & 200 \\ 40 & 40 & 40 & 80 & 80 & 160 & 200 & 200 \\ 40 & 40 & 40 & 80 & 80 & 160 & 200 & 200 \\ 40 & 40 & 40 & 80 & 80 & 160 & 200 & 200 \\ 40 & 40 & 40 & 80 & 80 & 160 & 200 & 200 \\ 40 & 40 & 40 & 80 & 80 & 160 & 200 & 200 \\ 40 & 40 & 40 & 80 & 80 & 160 & 200 & 200 \\ 40 & 40 & 40 & 80 & 80 & 160 & 200 & 200 \end{pmatrix}$$

Suppose also that the image is represented with an 8 bpp resolution, which corresponds to a source producing 256 (2^8) different integer values ranging in [0, 255]. If all these values are equiprobable then this particular image is one of the $2^{8\times8\times8} =$

Fig. 2.2 Simple graylevel image

$2^5 12 \approx 1.34 \times 10^{154}$ equally probable 8×8 images that can be produced by the specific source. In this image, four symbols (gray values) are present,

$$p(g): \begin{cases} g_1 = 40 & p(40) = 0.375 \quad \text{(counts of '40' divided by all image pixels)} \\ g_2 = 80 & p(80) = 0.250 \\ g_3 = 160 & p(160) = 0.125 \\ g_4 = 200 & p(200) = 0.250 \end{cases}$$

where g the variable for the gray values. If a uniform distribution is assumed (equiprobable gray values) then apparently the source the produces such images is characterized by an entropy of 8 bpp. If the entropy of the specific image is examined then according to the fundamental law of entropy (2.6),

$$H = -\sum_g p(g) \log_2 p(g) \approx 1.9056 \text{ bpp}$$

which amounts to $1.9056 \times 8 \times 8 = 122$ total bits and is the first-order entropy approximation, since all pixel values are supposed to be independent. A second-order approximation may be attained if the image is converted to one line of pixels and all pairs be counted,

40 40 40 80 80 160 200 200 40 40 40 80 80 160 200 200
40 40 40 80 80 160 200 200 40 40 40 80 80 160 200 200
40 40 40 80 80 160 200 200 40 40 40 80 80 160 200 200
40 40 40 80 80 160 200 200 40 40 40 80 80 160 200 200

It is easy to identify the gray level pairs listed in Table 2.1. In this case, the entropy is estimated by (2.6) as $2.75/2 = 1.375$ bpp, with the division by 2 imposed by the fact that pixels are taken in pairs. Apparently, a better estimate of the entropy is achieved, and it is expected that higher-order approximations provide even better estimates for a considerable computational cost, though. It should be noted that if

Table 2.1 Second-order entropy approximation with gray level pairs

Gray level pair	Counts	Probabilities
(40, 40)	16	0.250
(40, 80)	8	0.125
(80, 80)	8	0.125
(80, 160)	8	0.125
(160, 200)	8	0.125
(200, 200)	8	0.125
(200, 40)	8	0.125

the pixels are statistically independent any higher-order approximation collapses to the first-order approximation.

The case changes even more if the image pixels are subtracted to form a new 'difference' image,[2]

$$\begin{pmatrix} 40\ 0\ 0\ 40\ 0\ 80\ 40\ 0 \\ 40\ 0\ 0\ 40\ 0\ 80\ 40\ 0 \\ 40\ 0\ 0\ 40\ 0\ 80\ 40\ 0 \\ 40\ 0\ 0\ 40\ 0\ 80\ 40\ 0 \\ 40\ 0\ 0\ 40\ 0\ 80\ 40\ 0 \\ 40\ 0\ 0\ 40\ 0\ 80\ 40\ 0 \\ 40\ 0\ 0\ 40\ 0\ 80\ 40\ 0 \\ 40\ 0\ 0\ 40\ 0\ 80\ 40\ 0 \end{pmatrix}$$

In this case the symbols are reduced to three (0, 40, 80) with probabilities,

$$p(g): \begin{cases} g_1 = 0 & p(0) = 0.500 \\ g_2 = 40 & p(40) = 0.375 \\ g_3 = 80 & p(80) = 0.125 \end{cases}$$

and the estimated entropy is 1.4056 bpp. Thus, by taking a 'difference' image, an image that encodes the difference of the adjacent pixel values, even with a first-order approximation, a lower entropy is measured, which corresponds to fewer bits to describe the original image data (90 bits in this example).

It should be noted here that the entropy estimate provided by (2.6) can be computed using matrix operations. In case the probabilities of the symbols $p(x_i)$ are arranged in a row vector $\mathbf{P}$ then the computation of entropy can become a matrix multiplication,

$$H = -\mathbf{P}^T \cdot \log \mathbf{P} = \left[p(x_1) \cdots p(x_n)\right] \cdot \begin{bmatrix} \log p(x_1) \\ \vdots \\ \log p(x_n) \end{bmatrix} \tag{2.27}$$

where the ^{T} notation denotes the transpose and the logarithm operates on each of the elements of the matrix, producing a row vector of the logarithms of the probabilities.

2.2.1 Discrete Memoryless Sources

The simplest form of an information source is what is referenced as a *Discrete Memoryless Source (DMS)*, wherein the output symbols are statistically independent. A DMS S is characterized by its alphabet $S = \{s_1, s_2, ..., s_n\}$ and the corresponding

[2] Each new pixel (to the right of the first) represents the difference of its original value and the value of the previous pixel.

probabilities $P = p(s_1), p(s_2), ..., p(s_n)$. An important concept is the average information provided by this DMS, but before defining it, one should define a way of quantifying information in an event. Quantification of information content should, of course, have some practical consequences; for example, it is reasonable to accept that the emergence of a less probable event (or symbol) contains more information than the appearance of a more probable (expected). It should be also noted that the information content of several independent events, regarded as a new fact, is the sum of the information of independent events (Rabbani and Jones 1991a).

Defining $I(s_i)$ the information received by the appearance of a particular symbol s_i in terms of probability,

$$I(s_i) = \log_k \left(\frac{1}{p(s_i)} \right) \tag{2.28}$$

where the base of the logarithm, k, determines the information unit. If $k = 2$, the information is binary (in the form of bits).

Taking the average value of this quantity for all possible symbols of the DMS, the average information per symbol, $H(S)$, known as *entropy* may be computed as:

$$H(S) = \sum_{i=1}^{n} p(s_i) I(s_i) = -\sum_{i=1}^{n} p(s_i) \log_2 p(s_i) \text{ bps} \tag{2.29}$$

As an example, consider one DMS S producing four symbols, $S = \{A, B, C, D\}$ with probabilities $P(A) = 0.60$, $P(B) = 0.30$, $P(C) = 0.05$, $P(D) = 0.05$. By using (2.29) the source entropy is $H(S) = -[0.6 \log_2 0.6 + 0.3 \log_2 0.3 + 2 \times 0.05 \log_2 0.05] \approx 1.4$ bps.

The entropy of a source can be interpreted in two ways. By definition is the average value of information per symbol of the input source. It can also be defined as the average value of information per input source symbol that an observer needs to spend in order to alleviate the uncertainty of the source. For example, an observer is likely to want to recognize a symbol of the unknown source asking simple questions that require YES/NO (or TRUE/FALSE) answers. Each question is equivalent to an information bit. The observer needs to devise smart methods to minimize the average number of questions, which are required for the disclosure of a symbol or even a number of symbols. As intelligent though a method could be, it could never reveal symbols with an average number of binary questions less than the entropy of the source (Rabbani and Jones 1991a).

2.2.2 *Extensions of Discrete Memoryless Sources*

It is often useful to deal with symbol tables rather than pure symbols. Consider one DMS S with an alphabet of length n. Consider also that the output of the source is

grouped in sets of N symbols. Each group is considered to be a new symbol of an extended new source S^N, which has an alphabet size n^N. This new source is called *the N-th extension of S*. Since the sources 'have no memory', the probability of a symbol $\sigma_i = (s_{i1}, s_{i2}, ..., s_{iN})$ in S^N is given (Rabbani and Jones 1991a):

$$p(\sigma_i) = p(s_{i1})p(s_{i2})...p(s_{iN}) \tag{2.30}$$

It can therefore be shown that the entropy per symbol of the extended source S^N is N times the entropy per symbol of the source S:

$$H(S^N) = N \times H(S) \tag{2.31}$$

2.2.3 Markov Sources

The model of a DMS is too restrictive for many applications. In practice, it is observed that there is a previous section of a message to major influence the likelihood of the next symbol (i.e. the source has a 'memory'). Thus, in digital images, the probability of a pixel to get a specific value may depend significantly on the values of the pixels preceding. Sources with this feature are modeled as Markov or Markovian Sources (MS). An mth-order MS is a source, at which the probability of a symbol depends on the previous m symbols (m being a finite number). This probability in this case is a conditional probability expressed as $p(s_i|s_{j_1}, s_{j_2}, ..., s_{j_m})$, where $i, j_p (p = 1, .., m) = 1, 2, ..., n$. So an mth-order MS has n^m states. For an *ergodic MS* there is a unique probability distribution on the set of states, called stationary or equilibrium distribution (Rabbani and Jones 1991a). For the computation of the entropy of an mth-order MS the work is as follows (Rabbani and Jones 1991a):

- First calculate the entropy given that the source is in a particular state $s_{j_1}, s_{j_2}, ..., s_{j_m}$:

$$H(S|s_{j_1}, s_{j_2}, ..., s_{j_m}) = -\sum_{i=1}^{n} p(s_{j_1}, s_{j_2}, ..., s_{j_m}) \log_p(s_{j_1}, s_{j_2}, ..., s_{j_m}) \tag{2.32}$$

- Then sum, for all possible states, the products of the probabilities that the source are in that state with the corresponding entropy:

$$H(S) = \sum_{S^m} p(s_{j_1}, s_{j_2}, ..., s_{j_m}) H(S|s_{j_1}, s_{j_2}, ..., s_{j_m}) \tag{2.33}$$

Figure 2.3 depicts an example of a state diagram of a second order Markov process with binary alphabet $S = 0, 1$. The conditional probabilities are

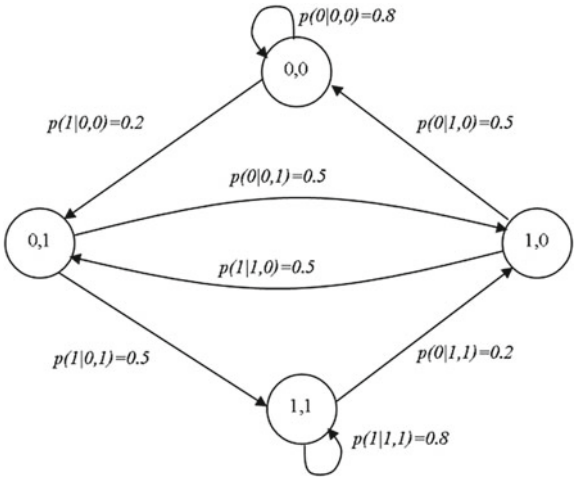

Fig. 2.3 Example of a states diagram for a Markov process

$$\begin{aligned} p(0|0,0) = p(1|1,1) &= 0.8 \\ p(1|0,0) = p(0|1,1) &= 0.2 \\ p(0|0,1) = p(0|1,0) = p(1|1,0) = p(1|0,1) &= 0.5 \end{aligned} \tag{2.34}$$

There are four possible states (0, 0), (0, 1), (1, 0) and (1, 1). Because of the symmetry the stationary probability distribution for these states satisfies

$$\begin{aligned} p(0,0) &= p(1,1) \\ p(0,1) &= p(1,0) \end{aligned} \tag{2.35}$$

In addition, the source will be in one of the given states at any time. So,

$$p(0,0) + p(0,1) + p(1,0) + p(1,1) = 1 \tag{2.36}$$

Apparently, there are two ways to reach state (0, 0),

- either by encountering a 0 symbol while the system is in state (0, 0) with 0.8 probability
- or by encountering a 0 symbol while the system is in state (1, 0) with 0.5 probability

Similarly, for state (0, 1),

$$\begin{aligned} p(0,0) &= 0.5\,p(0,1) + 0.8\,p(0,0) \\ p(0,1) &= 0.2\,p(0,0) + 0.5\,p(0,1) \end{aligned} \tag{2.37}$$

The solution of these equations leads to

$$\begin{aligned} p(0,0) &= p(1,1) = 5/14 \\ p(0,1) &= p(1,0) = 2/14 \end{aligned} \tag{2.38}$$

Then the entropy can be calculated as

$$H(S) = -\sum_{2^3} p(s_{i_1}, s_{i_2}, s_i) \log_2(s_i | s_{i_1}, s_{i_2}) = 0.801 \text{ bps} \tag{2.39}$$

2.2.4 The Theorem of Lossless Coding

According to the theory proposed by Shannon, there is a direct relation between the entropy and the information content. In this approach any source S can be simply considered being *ergodic* with an alphabet of size n, and entropy $H(S)$. Whenever segments of N input symbols from this source are being encoded into binary words, $\forall \delta > 0$, it is possible, by choosing a large enough N, to create a code in such a way that the average number of bits per input symbol $\overline{L}$, will satisfy the inequality (Rabbani and Jones 1991a)

$$H(S) \leq \overline{L} < H(S) + \delta \tag{2.40}$$

This inequality is usually referenced as the *theorem of lossless coding* and expresses the fact that *each source may be losslessly encoded with a code, in which the average number of bits per symbol for any input symbol is close, but never less than the entropy of the source*. But while the theorem assures of the existence of a code that can achieve a value close to the entropy of the source, it does not provide any information on the way of its creation. In practice, variable sized codes with extended source models are being used to achieve the desired performance (Rabbani and Jones 1991a). A code is called *compact* (for a given source), when its average length is less than or equal to the average length of all other codes that satisfy the *prefix codes condition*[3] for the same source and the same alphabet.

2.3 Digital Image Compression

After the necessary introduction to the information theory based essentially on the probability theory, the transition to the study of more complex sources can be done with two-dimensional data: the digital images. The examples presented thus far

[3]The prefix condition, which represents a necessary and sufficient condition for the creation of variable length codes, states that no code can be the beginning of another code for the same alphabet.

indicate that the digitization of images leads to a massive amount of data. The amount, though, of bits actually required to describe the information of an image can be much smaller due to the existence of information *redundancy*. The main objective of research in image compression is to reduce the number of bits required for the representation, by removing such redundancies. These redundancies are either of *statistical nature* and can be identified through proper modeling of the source, or of *optical nature* directly connected to the HVS, identified by appropriate modeling of the HVS itself. At the same time, establishing fundamental limits for the performance of each type of compression method to a specific kind of image, through the use of the basic principles of information theory are also considered. Apart from these basic objectives of the relevant research it is necessary to develop various algorithms that can 'fit' in different applications. There are several approaches to image compression, but generally all fall into two basic categories, *lossless* and *lossy* image compression (Fig. 2.4):

- *Lossless compression*—or *reversible coding*, in which the reconstructed image (decompressed) after compression is numerically identical to the original image (at pixel level). Obviously, lossless compression is ideal, since no part of the original information is likely to be lost or distorted. In this way, however, only small compression ratios can be achieved, typically of the order of 2 : 1. This type of compression has to do with the modeling of statistical characteristics of the image.

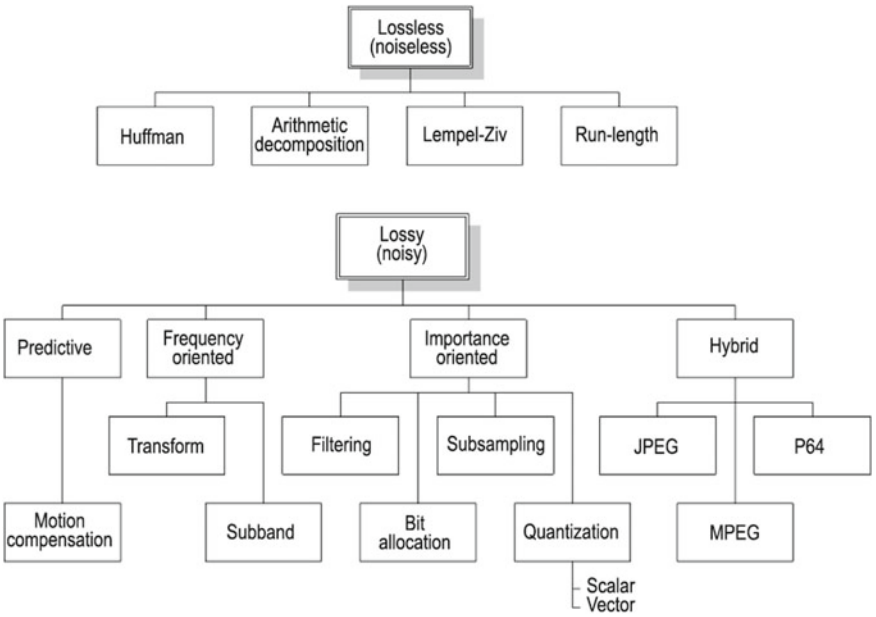

Fig. 2.4 General categorization of compression methods

- *Lossy compression*—or *irreversible coding*, wherein the reconstructed image is degenerate with respect to the original. Lossy compression can achieve significantly higher compression ratios. However, compression is achieved at the cost of increasing distortion, i.e. losing parts of the original data. It is important that the degeneration of the reconstructed image may or may not be noticeable, or it is possible to select the level and nature of distortions that might be acceptable. This type of compression makes heavy use of the concept of HVS modeling and exploitation of redundancies of an optical nature. The term *compression "without noticeable losses"* is often used to describe compression methods with losses that result in images with no noticeable degeneration (under normal or specific viewing conditions). It should be noted that the scope of the term "noticeable loss" in the definition of "compression without noticeable loss" is quite subjective and should be used with caution. Clearly, a method that implements compression with "no noticeable losses" that is specifically designed for display (e.g. for a computer screen of 19 in. and for an observer at a distance of about one meter) could be largely insufficient for other purposes, such as e.g. for printing on paper or on film.

In lossy compression it is allowed (or is tacitly accepted) to produce errors in order to increase the compression ratio. These errors can be quantified by distortion estimators. Usually this distortion is denoted as $D(I, \hat{I})$ and expresses the qualitative difference between an original image $I \equiv i[x, y]$ and a reconstruction $\hat{I} \equiv \hat{i}[x, y]$ from the compressed image.[4]

A widespread distortion estimator is the Mean Squared Error (MSE) defined as:

$$\text{MSE} = \frac{1}{N_1 \cdot N_2} \sum_{x=0}^{N_1-1} \sum_{y=0}^{N_2-1} (i[x, y] - \hat{i}[x, y])^2 \tag{2.41}$$

In image compression, this estimator is expressed through another equivalent estimator, PSNR, which takes the place of an objective image quality estimator, and is defined as:

$$\text{PSNR} = 10 \log_{10} \frac{(2^B - 1)^2}{\text{MSE}} = 20 \log_{10} \frac{2^B - 1}{\text{RMSE}} \tag{2.42}$$

where B is the color depth of the image in bpp and Root Mean Squared Error (RMSE) is the square root of the MSE. *PSNR is expressed in dB and is the most widespread mathematical formulation of the qualitative difference between two images.*

Along with the assessment of the distortion, there is also the assessment of the efficiency of compression. Since the objective of compression is the representation of an original image by a string consisting of bits (usually referenced as a bitstream), the aim is to maintain the length of this string as short as possible. In the general case an image of dimensions $N_1 \times N_2$ with a color depth of B bpp requires $N_1 \times N_2 \times B$

[4]This is a typical symbolism for digital images. Every image consists of a sequence of two-dimensional samples. The parameters x and y represent these two dimensions: $0 \leq x < N_1, 0 \leq y < N_2$, N_1 and N_2 being the image dimensions in the horizontal and vertical directions respectively.

Table 2.2 Typical compression performance values

Lossless	High quality lossy	Medium quality lossy	Low quality lossy
(B-3) bpp	1 bpp	0.5 bpp	0.25 bpp

bits for a complete representation. After compressing this image, the compression ratio may be defined as:

$$\text{compression ratio} = \frac{N_1 N_2 B}{\|c\|} \tag{2.43}$$

where $\|c\|$ is the length of the final bitstream c.

Equally, the data rate (or bit-rate) in bpp is defined as (Taubman and Marcellin 2002b):

$$\text{bit} - \text{rate} = \frac{\|c\|}{N_1 N_2} \tag{2.44}$$

For the case of lossy compression, the compression rate (bit-rate) is an important performance measure, particularly since the least significant bits of large color depth images can be removed without significant (noticeable) optical distortion. The measure, therefore, of the average number of bits spent for each sample of the image is usually the most important measure of compression efficiency, regardless of the accuracy with which these samples were represented initially. Table 2.2 presents indicative values of compression rate for natural images (photographs of the real world) in bpp, both for the case of lossless and lossy compression, adopting a convention that the encoded lossy images are being viewed in typical monitor resolution of 90 pixels per inch (or 22 pixels per mm) (Taubman and Marcellin 2002b). It should be noted that the values are indicative for each color channel (i.e. directly applicable to images of gray tones) and can show considerable variations depending on the image content.

2.3.1 Exploiting Redundancy for Compression

Without compression, an image is represented, generally, by a large amount of bits. Compression is only possible if some of those bits can be regarded as redundant or in some cases irrelevant. Two main categories of redundancy can be distinguished:

- spatial or statistical redundancy, and
- spectral redundancy or color correlation

In parallel, various application specifics render some of the information content of images irrelevant. This is a very important aspect relating to a potentially large amount of data able to be discarded, but it will not be discussed here, as it is application-dependent.

2.3.1.1 Spatial and Statistical Redundancy

Consider the simple case of a binary image $N_1 \times N_2$ consisting of white and black pixels, and suppose each pixel is treated as an integer of B bits long. The image will then be represented by $N_1 \times N_2 \times B$ bits. If an encoding-decoding system knows a priori that the image is binary, it is then able to use only one bit for each pixel and thereby achieve a compression ratio of $B : 1$. But in the general case, the system may just assume that some subset of combinations of pixels exhibits a higher incidence rate than others. So if it is known that the next sample belongs with high probability to the set {0, 1} then it is expected to spend a little more than 1 bit for the actual value of the sample. This is indicated by the theory of entropy, which shows the smallest amount of data expected to be consumed for the representation of a random variable (image samples in this case). This is the overall of the a priori knowledge for the input image regarded as a random variable.

Additionally, the system may assume that a small pixel neighborhood is very likely to exhibit small and smooth variation. Making this assumption, the system can benefit from previous pixel values to reduce the number of bits required to represent the next. The assumption is based on probability theory. This relative a priori knowledge can be described by the joint probability of two random variables, and thus the expected number of bits are already known through conditional entropy.

So what is readily understood is that if in a given image the number of pixels needed to extract robust statistics to predict new pixel values, then the compression rate is expected to be high.

2.3.1.2 Spectral Redundancy

Considering any digital color image as a union of numerous point spread functions (on each pixel) on the three dimensions color dimensions R, G and B, leads to an obvious recognition of inter-pixel color dependencies. That is, color variations in small regions of natural color images are expected to be limited or at least gradual. The consequence of this is that color images do not exhibit abrupt color changes, since there is usually a significant amount of color correlation. The obvious way to tackle with such correlation would be to apply a transformation to another color space that is decorrelated, minimizing the amount of redundancy in the color space of the image. Theoretically, the optimum (in a least squares sense) transformation in this case would be the Karhunen-Loeve Transform (KLT), or otherwise to apply the Principal Component Analysis (PCA) on the color space. This typically means that the covariance matrix of the image color pixels be computed, the eigenvalues of the covariance matrix be extracted and the corresponding eigenvectors be identified. The three eigenvectors, which are orthogonal, define a newcolor space for the image which has the least possible color redundancy (from a statistical point of view). Figure 2.5 shows how the principal components of a color image would look like after the application of PCA on the image color space.

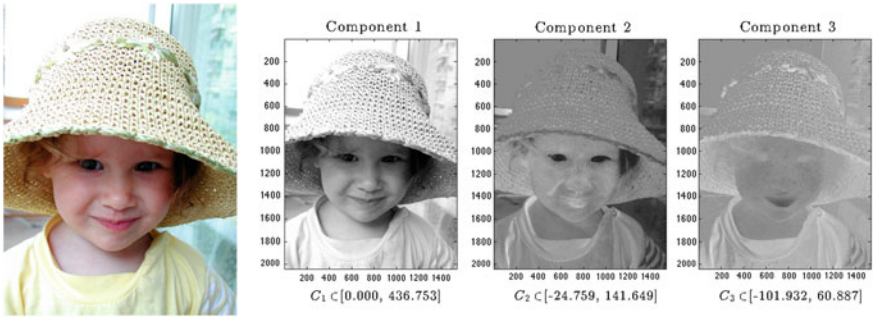

Fig. 2.5 KLT color space representation of a color image

In addition, it is known, according to many studies, that a human observer is significantly less sensitive to sudden changes in chromaticity than in luminance. This feature of the HVS is usually modeled by adopting a linear and sometimes a non-linear transformation of the original RGB representation of an image into a luminance-chromaticity representation (typically followed by a downsampling of the chromaticity channels). An example of this transformation is the transformation of RGB to YC_bC_r (Taubman and Marcellin 2002b):

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.1140 \\ -0.169 & -0.331 & 0.5000 \\ 0.500 & -0.419 & -0.0813 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.45}$$

Usually, after the application of the transformation, subsampling of the chromaticity channels (C_b, C_r) takes place either in the horizontal, or in the vertical direction or both. Important research in the field of modeling of the HVS (as already reported in the previous chapter), has made other similar transformations possible, which lead to different color systems. In those systems (such as CIELAB and sCIELAB) a linearization of the distances between colors as perceived by the HVS is targeted. In any case, conversion to another color system and sub-sampling of the channels with less visual importance leads to discarding of a significant amount of samples, without imposing a significant (noticeable) visual distortion, by just exploiting the characteristics (limitations) of the HVS.

2.3.2 Structure of a Basic Compression System

An abstraction of a compression-decompression system is shown in Fig. 2.6 as two mappings M and $\overline{M^{-1}}$ respectively. To achieve lossless compression, $\overline{M^{-1}} = M^{-1}$ should hold. In lossy compression, mapping M is not reversible, so the notation $\overline{M^{-1}}$ is used to indicate that the decompression system is an approximate inverse.

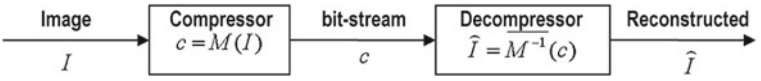

Fig. 2.6 Compression as a mapping

The compression system may be considered as one large Look-Up Table (LUT) with $2^{N_1 N_2 B}$ values. In the general case of designing an encoder M of fixed-length code words, an obvious way is to use the relation

$$c = M(I) = \arg\min_{c'} D(I, \overline{M^{-1}}(c')) \tag{2.46}$$

where D with a measure of the distortion and $\|c\|$ the fixed coding length imposed by the particular encoder, for which the decoder can be regarded as a LUT with $2^{\|c\|}$ values.

This condition is sufficient to maintain the LUT in a small number of elements both to the compressor and the decompressor. Thus, the compressor produces the string whose reconstruction by the decompression will be the most equivalent to the original image, based on the measure of distortion estimation. This approach results in that the mapping $\overline{M^{-1}}$ is the reverse of M:

$$M(\overline{M^{-1}}(c)) = c \tag{2.47}$$

Such a compression system is immune to distortion due to a repetitive compression-decompression.

This is basically the idea behind Vector Quantization (VQ), the generality of which is very attractive. However, the exponential increase in the size of the LUT with the increase in complexity, renders it an impractical solution (except for small images with few samples). It is therefore necessary to implement a separation of the various complementary structures within the mappings M and $\overline{M^{-1}}$. The basic structure of a general compression scheme, as it stands following this distinction, and is the basis for almost all today's compression systems, is presented in Fig. 2.7. The first step is to transform the input into a new set of samples, particularly suitable for compression. In the second step, the new samples are quantized. During the final third step, the quantized samples are encoded to form the final compression bitstream.

In the following paragraphs, a description of the key parts of a structured compression system is presented. In this presentation it is considered that any color space transformation has already occurred on the input image data.

2.3.2.1 Transformation

The term *image transform* typically refers to a class of unitary matrices used for the representation of images. Similarly to the one-dimensional case, where a signal may

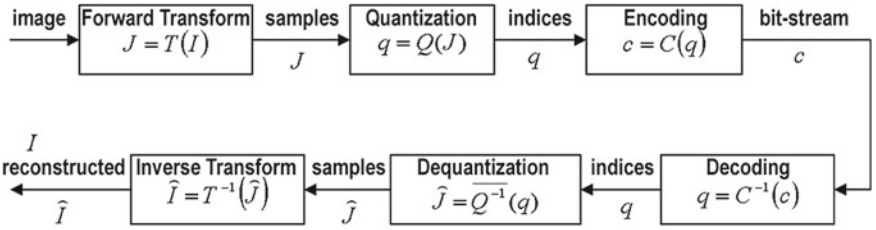

Fig. 2.7 Elements of a structured compression system

be represented by a series of *(orthogonal) basis functions*, images are represented by a series of two-dimensional basis functions or *basis images*. In the one-dimensional case, a continuous function may be expressed as a series expansion using orthonormal (orthogonal unitary) coordinate vectors, in order to produce a set of coefficients, which may be used to represent the original function,

$$\mathbf{J} = \mathbf{TI} \Leftrightarrow J(u) = \sum_{x=0}^{N-1} T(u,x)I(x), \quad 0 \le u \le N-1 \tag{2.48}$$

where $\mathbf{T}$ is the orthonormal transformation matrix and $\mathbf{I}$ the input data matrix. Since the transformation is orthonormal it holds that $\mathbf{T}^{-1} = \mathbf{T}^{*T}$, and the inversion of the transformation is,

$$\mathbf{I} = \mathbf{T}^{-1}\mathbf{J} \Leftrightarrow I(x) = \sum_{u=0}^{N-1} J(u)T^{*}(u,x), \quad 0 \le x \le N-1 \tag{2.49}$$

Apparently, Eq. (2.49) can be viewed as a series expansion of $I(x)$. In this representation, the columns of $\mathbf{T}^{-1}$ are the basis vectors of $\mathbf{I}$. It is also apparent in this expansion that the elements $J(u)$ are the coefficients that are needed to scale the basis vectors in order to reconstruct the original data (Jain 1988).

In image processing, the signals are two-dimensional and the above approach scales up to the following pair of transformations,

$$\begin{aligned} J(u,v) &= \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} T_{u,v}(x,y)I(x,y), \quad 0 \le u,v \le N-1 \\ I(x,y) &= \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} J(u,v)T^{*}_{u,v}(x,y), \quad 0 \le x,y \le N-1 \end{aligned} \tag{2.50}$$

where $T_{u,v}(x,y)$ is a set of orthonormal basis functions satisfying the *orthogonality* and the *completeness* properties,

$$\begin{aligned}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} T_{u,v}(x,y)T^{*}_{u',v'}(x,y) &= \delta(u-u', v-v')\\ \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T_{u,v}(x,y)T^{*}_{u,v}(x',y') &= \delta(x-x', y-y')\end{aligned} \tag{2.51}$$

'δ' denoting the impulse function. The $J(u, v)$ elements are the *transform coefficient* and the set of all transform coefficients $\mathbf{J} \triangleq \{J(u, v)\}$ is the *transformed image* (Jain 1988). In this context, the transformation matrix $\mathbf{T}$ is also called the *transformation kernel* and a two-dimensional transformation is said to be *separable* if its kernel can be written as a multiplication of two kernels acting on the two spatial dimensions of the data,

$$T_{u,v}(x, y) = T_C(u, x) \cdot T_R(v, y) \tag{2.52}$$

where the $_C$, $_R$ indices denote the column- and row-wise one-dimensional operation respectively. Thus an image transform can be written as,

$$J(u, v) = \sum_{y=0}^{N-1}\Big[\sum_{x=0}^{N-1} T_C(u, x)I(x, y)\Big]T_R(v, y) \tag{2.53}$$

With all these representations converted into matrix form,

$$\mathbf{J} = \mathbf{T_C} \cdot \mathbf{I} \cdot \mathbf{T_R^T} \tag{2.54}$$

while the inverse transforms follow the same formalism (Pratt 1991).

In image compression, the *transformation* (or simply *transform*) is responsible for the conversion of input samples to a format that allows for easier and more efficient quantization and encoding. A transformation captures inherent statistical dependencies among samples in the original data, so that the transformed sample would include only small local correlation. In the ideal case, the samples after the application of the transformation must be statistically independent. Additionally, the transformation must separate the information into parts that can be characterized as 'important' or not, so that the redundant (less or non-important) information be quantized with higher quantization level or even be eliminated. Specifically, in the unitary transformation $\mathbf{J} = \mathbf{T} \cdot \mathbf{I}$ holds that,

$$\|\mathbf{J}\|^2 \triangleq \mathbf{J}^{*T}\mathbf{J} = \mathbf{I}^{*T}\mathbf{T}^{*T}\mathbf{T}\mathbf{I} = \mathbf{I}^{*T}\mathbf{I} \triangleq \|\mathbf{I}\|^2 \Rightarrow \|\mathbf{J}\|^2 = \|\mathbf{I}\|^2 \tag{2.55}$$

which states that the *signal energy is preserved*. In addition, this equation shows that a unitary transformation is a *simple rotation in the N-dimensional vector space*, or the components of $\mathbf{J}$ are the *projections* of $\mathbf{I}$ on the new basis (Jain 1988).

In addition, since in unitary transforms the energy is preserved, if the transformation concentrates the energy of the signal to a small amount of coefficients would significantly aid in the compression. The fact is that unitary transforms actually do pack a large fraction of the energy into few coefficients. If $\boldsymbol{\mu}_I$ and $\boldsymbol{\Sigma}_I$ represent the mean value and the covariance of an input data vector $\mathbf{I}$ then the corresponding quantities for the transform coefficients are,

$$\begin{aligned}\boldsymbol{\mu}_J &\triangleq E[\mathbf{J}] = E[\mathbf{TI}] = \mathbf{T}E[\mathbf{I}] = \mathbf{T}\boldsymbol{\mu}_I \\ \boldsymbol{\Sigma}_J &= E\left[(\mathbf{J}-\boldsymbol{\mu}_J)(\mathbf{J}-\boldsymbol{\mu}_J)^{*T}\right] \\ &= \mathbf{T}\left(E\left[(\mathbf{I}-\boldsymbol{\mu}_I)(\mathbf{I}-\boldsymbol{\mu}_I)^{*T}\right]\right)\mathbf{T}^{*T} = \mathbf{T}\boldsymbol{\Sigma}_I\mathbf{T}^{*T}\end{aligned} \tag{2.56}$$

In the diagonal of $\boldsymbol{\Sigma}_I$ lie the variances of the transform coefficients,

$$\sigma_J^2(k) = [\boldsymbol{\Sigma}_J]_{k,k} = \left[\mathbf{T}\boldsymbol{\Sigma}_I\mathbf{T}^{*T}\right]_{k,k} \tag{2.57}$$

and since $\mathbf{T}$ is unitary,

$$\sum_{k=0}^{N-1}|\mu_J(k)|^2 = \boldsymbol{\mu}_J^{*T}\boldsymbol{\mu}_J = \boldsymbol{\mu}_I^{*T}\mathbf{T}^{*T}\mathbf{T}\boldsymbol{\mu}_I = \sum_{n=0}^{N-1}|\mu_I(n)|^2 \tag{2.58}$$

$$\sum_{k=0}^{N-1}\sigma_J^2(k) = Tr\left[\mathbf{T}\boldsymbol{\Sigma}_I\mathbf{T}^{*T}\right] = Tr[\boldsymbol{\Sigma}_I] = \sum_{n=0}^{N-1}\sigma_I^2(n) \tag{2.59}$$

so,

$$\sum_{k=0}^{N-1}E[|J(k)|^2] = \sum_{n=0}^{N-1}E[|I(n)|^2] \tag{2.60}$$

and the average energy $E[|J(k)|^2]$ of the transform coefficients $J(k)$ tends to be unevenly distributed, even though the energy in the input might be evenly distributed. In addition, the off-diagonal elements of the covariance matrix $\boldsymbol{\Sigma}_J$ tend to become small compared to the diagonal elements, which indicates that *the transform coefficients tend to be uncorrelated* (Jain 1988).

Last but certainly not least, the eigenvalues and the determinant of unitary transform matrices have unity magnitude and the entropy of the input signal is preserved during such a transformation, which states that *unitary transformations preserve the information content of a signal* (Jain 1988).

Known transformations that satisfy these requirements are either those which usually act on parts of the images (block transforms) as the *Karhunen-Loeve transform*, the *Fourier transform* and the *cosine transform*, or global transforms applied into spectral regions (sub-band transforms), as the *Wavelet transform*. In subsequent paragraphs a brief and comprehensive description of these transformations as applied in (2D data) image compression is included.

In order to adopt a more practical description and notation in the context of image compression, the operators of transformations are being defined in relation to their Point Spread Function (PSF). The PSF of an operator T is the result of the application of the operator on a point source, and for the case of 2-D are

$$\begin{aligned} T\left[\text{point source}\right] &= \text{ point spread function, or} \\ T\left[\delta(x-\alpha, y-\beta)\right] &= h(x, \alpha, y, \beta) \end{aligned} \tag{2.61}$$

where $\delta(x-\alpha, y-\beta)$ is an impulse function that expresses a point source of intensity 1 at point (α, β). The physical meaning of function $h(c, \alpha, y, \beta)$ is that it expresses how the input value in position (x, y) affects the output value at position (α, β).

When the operator is linear and the point source is c times more intense, then the result of the operator is c times larger, thus is

$$T[c \cdot \delta(x-\alpha, y-\beta)] = c \cdot h(x, \alpha, y, \beta) \tag{2.62}$$

Considering an image as a collection of such point sources each of different intensity, one may, ultimately, represent the image as the sum of all these point sources. In this case, the result of applying an operator characterized by a PSF $h(x, \alpha, y, \beta)$ to a square $N \times N$ image $I(x, y)$ can be written as

$$J(\alpha, \beta) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I(x, y) h(x, \alpha, y, \beta) \tag{2.63}$$

where the value of (α, β) is a linear combination of the values at all image locations (x, y) weighted by the PSF of the transformation.

Apparently, the problem in image compression is to define a transformation $h(x, \alpha, y, \beta)$ such that the input image can be losslessly represented with fewer bits. Additionally, if the effect expressed by the PSF of a linear transformation is independent of the actual position but depends on the relative position of pixels that affect and are affected, then the PSF is *translation invariant*,

$$h(c, \alpha, y, \beta) = h(x-\alpha, y-\beta) \tag{2.64}$$

and then (2.63) expresses a convolution

$$J(\alpha, \beta) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I(x, y) h(x-\alpha, y-\beta) \tag{2.65}$$

Further, if the columns are influenced independently of the rows of the image, the PSF is *separable*,

$$h(c, \alpha, y, \beta) = h_c(x, \alpha) \cdot h_r(y, \beta) \tag{2.66}$$

where the index c is used for the columns, while the index r for the rows, and thus (2.63) is written as the sequential application of two one-dimensional transformations,

$$J(\alpha, \beta) = \sum_{x=0}^{N-1} h_c(x, \alpha) \sum_{y=0}^{N-1} I(x, y) h_r(y, \beta) \tag{2.67}$$

And if the PSF is *translation invariant and separable*, then (2.63) is written as a sequence of two one-dimensional convolutions,

$$J(\alpha, \beta) = \sum_{x=0}^{N-1} h_c(x - \alpha) \sum_{y=0}^{N-1} I(x, y) h_r(y - \beta) \tag{2.68}$$

To transform these equations into matrix form, (2.67) can be expressed as

$$\mathbf{J} = \mathbf{h}_c^T \cdot \mathbf{I} \cdot \mathbf{h}_r \tag{2.69}$$

The transformation in this equation is *unitary* if matrices $\mathbf{h}_c$ and $\mathbf{h}_r$ are unitary. The concept of unitary is similar to that of the *orthogonal* except that it relates to complex-valued quantities (of unit magnitude).

Reversing (2.69), the image $\mathbf{I}$ can be recovered as

$$\mathbf{I} = \mathbf{h}_c \cdot \mathbf{J} \cdot \mathbf{h}_r^T \tag{2.70}$$

which in the form of elements is written as

$$I(x, y) = \sum_{\alpha=0}^{N-1} \sum_{\beta=0}^{N-1} J(\alpha, \beta) \mathbf{h}_r(y, \beta) \mathbf{h}_c(x, \alpha) \tag{2.71}$$

where for constants α and β, $\mathbf{h}_c$ and $\mathbf{h}_r$ are vectors. Their product is of $N \times N$ dimension. Taking all possible combinations of columns of $\mathbf{h}_r$ and rows of $\mathbf{h}_c$ the *basis matrices* are produced. Thus, Eq. (2.71) is the expression of the original image with respect to these basis matrices. Finally, the data obtained from the transformation of the original image are the coefficients with which each of the basis images (matrices) must be multiplied before being summed with the others to reconstruct the entire image.

Karhunen-Loeve Transform

Supposing a Gaussian random variable $\mathbf{I} = (I_1, I_2, ..., I_k)$, the covariance $\sigma_{i,j}$ between I_i and I_j is defined as

$$\sigma_{i,j} = E\left[(I_i - \mu_i)(I_j - \mu_j)\right] \tag{2.72}$$

where μ_i, μ_j, are the mean values of I_i and I_j respectively. The $k \times k$ covariance matrix of the random variable $\mathbf{I}$ is

$$\mathbf{\Sigma}_I = [\sigma_{i,j}]_{i,j=1}^{k} \tag{2.73}$$

If $\mathbf{A}$ is the $k \times k$ orthogonal matrix whose rows are the eigenvectors of $\mathbf{\Sigma}_I$, then the Karhunen-Loeve Transform (KLT) of $\mathbf{I}$ is defined as

$$\mathbf{Y}^T = \mathbf{A} \cdot \mathbf{I}^T \tag{2.74}$$

Vector $\mathbf{Y}$ is a Gaussian variable with statistically independent elements.

In practice, the question is to find the set of basis images, which are optimal for a family of images with similar statistical characteristics. To answer this question, the assumption that images are *ergodic processes* is made, namely:

- the mean value over the total family of images equals to the spatial mean in each of the images (Fig. 2.8)
- the covariance matrix over the total family of images is equal to the covariance matrix of each of the images

Of course, the ergodicity assumption is correct only in the case of uniform noise. But it can be approximately correct for small portions of an image. The application of the transformation on a graylevel (one-color-channel) image $\mathbf{I}$ of $N \times N$ dimensions involves:

1. Conversion of the image $\mathbf{I}$ to a column vector i of $N^2 \times 1$ dimensions, by putting the columns one below the other
2. Calculation of the covariance matrix $\mathbf{\Sigma}$ that is of size $N^2 \times N^2$
3. Calculation of the eigenvalues of the covariance matrix
4. Calculation of the corresponding eigenvectors
5. Creation of matrix $\mathbf{A}$ of the eigenvectors (as rows) using an ordering imposed by sorting the eigenvalues in a decreasing order
6. Compute the transformed image is: $\tilde{\mathbf{I}} = \mathbf{A}(\mathbf{I} - \boldsymbol{\mu})$ where $\boldsymbol{\mu}$ a vector with all elements equal to the mean value of the image

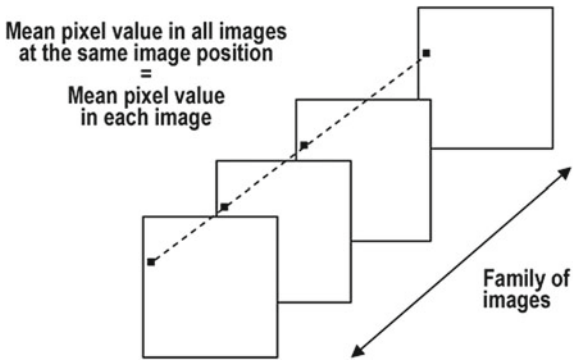

Fig. 2.8 Ergodicity in images

This transformation is optimal from the energy standpoint; if only a limited number of the KLT transform coefficients are stored, they are expected to hold the most part of the total energy in comparison to all other transformations. Unfortunately, the basis functions of the KLT dependent on the image and require the calculation of the covariance. Thus, a fast KLT algorithm is not possible if the transformation is applied to the entire image (global). Therefore, the KLT is limited to only few applications in image compression (Rabbani and Jones 1991a). Nevertheless, alternative implementations have already been proposed based on the assumed ergodicity in natural digital images and that KLT is equivalent to Singular Value Decomposition (SVD), which decomposes the original data into three matrices (factorization) in the typical form $\mathbf{I} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, representing the new coordinate system to project the original image (columns and rows of matrices $\mathbf{U}$, $\mathbf{V}$) and the amount of scaling to apply (on the diagonal of $\boldsymbol{\Sigma}$—the variances). Figure 2.9 shows an example of the KLT on a graylevel image. The original image is shown on the left; the two noisy images on the right represent the $\mathbf{U}$ and $\mathbf{V}^T$ matrices. The graph below the noisy images is a portion of the graphical representation of the variances in the diagonal of $\boldsymbol{\Sigma}$, which illustrates how the most of the energy is concentrated in the first values. The image can then be represented by a limited amount of basis vectors and variances to impose efficient coding and, consequently, significant compression. Figure 2.10 shows the reconstruction that can be achieved by using a limited number of dimensions, by discarding (setting to zero) a number of variances in $\boldsymbol{\Sigma}$. Specifically, the first reconstruction is attained by keeping only the first element in $\boldsymbol{\Sigma}$ containing almost no

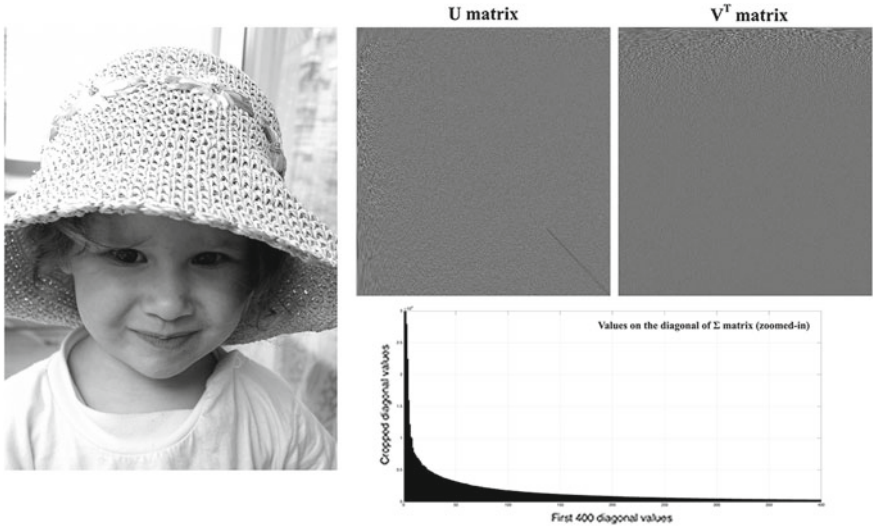

Fig. 2.9 KLT (PCA) on a graylevel image

Fig. 2.10 Progressive reconstruction of the graylevel image undergone KLT

valuable information; what is really surprising is that by keeping only about 100 elements in $\mathbf{\Sigma}$ a very good reconstruction can be achieved, like in the case of 118 elements that results in 30.57 dB PSNR, which is fairly a threshold above which images are considered of an acceptable quality. It should be noted that all elements of the diagonal of $\mathbf{\Sigma}$ are 1536 for this example, so the 290 elements are just a small 19 % fraction of the original data producing a significant 37.67 dB quality.

Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is often used in the spectral analysis and filter design. Following the line of unitary transforms, for an $M \times N$ input matrix $\mathbf{I}$ (image), the two-dimensional DFT is defined as

$$J(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} I(x,y) e^{-2\pi i (\frac{ux}{M} + \frac{vy}{N})}, \quad i = \sqrt{-1} \tag{2.75}$$

whereas the inverse transform is defined as:

$$I(x,y) = \frac{1}{M \cdot N} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} J(u,v) e^{2\pi i (\frac{ux}{M} + \frac{vy}{N})}, \quad i = \sqrt{-1} \tag{2.76}$$

The DFT decomposes an image into its spectral bands. The parameters u, v are called spatial frequencies of the transformation. The kernel of the transformation may be separated as follows

$$e^{-2\pi i(\frac{ux}{M}+\frac{vy}{N})} = e^{-\frac{2\pi iux}{M}} \cdot e^{-\frac{2\pi ivy}{N}} \tag{2.77}$$

so that the two-dimensional transformation is implemented by two successive one-dimensional transformations,

$$\begin{aligned} J(u,v) &= \sum_{x=0}^{M-1} e^{-\frac{2\pi iux}{M}} \sum_{y=0}^{N-1} I(x,y) e^{-\frac{2\pi ivy}{N}} \\ &= \sum_{x=0}^{M-1} J(x,v) e^{-\frac{2\pi iux}{M}} \\ \text{where } J(x,v) &= \sum_{y=0}^{N-1} I(x,y) e^{-\frac{2\pi ivy}{N}} \end{aligned} \tag{2.78}$$

Extensive study has been performed on fast implementations of DFT (such as the Fast Fourier Transform (FFT)), which typically exhibit a computational complexity of $O(N \log_2 N)$ for a transformation of N-points (Rabbani and Jones 1991a). In general, the transform coefficients generated by the DFT are complex numbers so their storage and handling is problematic. Actually there are $2N^2$ transform coefficients, but because of the symmetry of the conjugate elements, it holds that

$$J(u,v) = J^*(-u + lN, -v + mN) \tag{2.79}$$

for $l, m = 0, 1, 2, \ldots$. and J^* the complex conjugate of J. Thus, almost half of the elements are redundant, which is a disadvantage of the DFT. Another disadvantage of the DFT is the occurrence of false spectral components due to indirect periodicity in the image boundaries. When the DFT is used for compression with high compression ratio, these spurious components can lead to the appearance of sudden changes (*blocking artifacts*) between the parts of the image in which the transformation is applied. To express the transformation in the form of matrices, according to what has been defined in the introductory paragraphs, it suffices to define the functions h_r and h_c

$$h_c(x,u) = e^{-i\frac{2\pi xu}{M}}, \quad h_r(y,v) = e^{-i\frac{2\pi yv}{N}} \tag{2.80}$$

Figures 2.11, 2.12, 2.13, 2.14, and 2.15 show an example of the application of the DFT and the possible reconstructions using a limited amount of the transform coefficients. In this example only the magnitude of the complex coefficients is considered. The reconstruction is begin done by applying a filtering on the transform domain, zeroing out most of the transform coefficients as shown in the figures. It is apparent that the transform successfully compacts the image data into only a few

Fig. 2.11 DFT representation of a graylevel image by applying an FFT

Fig. 2.12 Reconstruction using 0.014% of the DFT coefficients at 19.78 dB PSNR

Fig. 2.13 Reconstruction using 0.34% of the DFT coefficients at 22.70 dB PSNR

Fig. 2.14 Reconstruction using 1.34% of the DFT coefficients at 26.07 dB PSNR

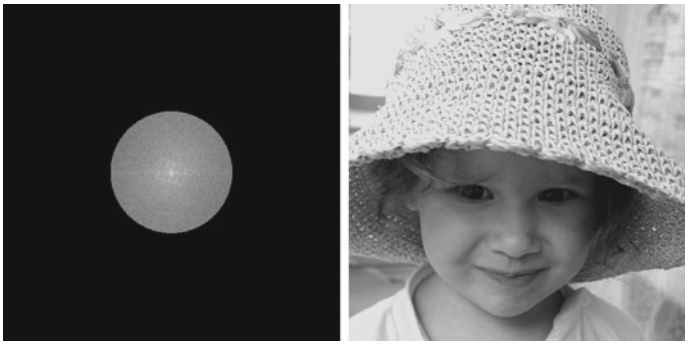

Fig. 2.15 Reconstruction using 10.25% of the DFT coefficients at 33.05 dB PSNR

transform coefficients, achieving an acceptable image quality (a good approximation of the original image data) with a very limited amount of data.

Discrete Cosine Transform

For practical applications in image processing the DFT cannot be the first choice as it poses computational difficulties by producing complex valued coefficients. This issue along with a need for fast and practical implementations of KLT gave rise to real-valued related transformations, such as the Discrete Cosine Transform (DCT). The DCT is defined by the transformation matrix $\mathbf{C} = \{C(u, x)\}$,

$$C(u,x) = \begin{cases} \sqrt{\frac{1}{N}} & u = 0,\ 0 \le x \le N-1 \\ \sqrt{\frac{2}{N}} \cos \frac{(2x+1)u\pi}{2N} & 1 \le u \le N-1,\ 0 \le x \le N-1 \end{cases} \tag{2.81}$$

The two-dimensional DCT of a square $N \times N$ matrix $\mathbf{I}$ is defined as

$$J(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} C(u,x)I(x,y)C(v,y)$$

$$J(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} I(x,y)c(u)c(v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \tag{2.82}$$

$$J(u,v) = c(u)c(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} I(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N}$$

with,

$$c(\xi) = \begin{cases} \sqrt{\frac{1}{N}}, & \xi = 0 \\ \sqrt{\frac{2}{N}}, & \text{otherwise} \end{cases}$$

Alternatively, the forward DCT can be found in the literature defined as,

$$J(u,v) = \frac{2}{N}c(u)c(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} I(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \tag{2.83}$$

with,

$$c(\xi) = \begin{cases} \sqrt{\frac{1}{2}}, & \xi = 0 \\ 1, & \text{otherwise} \end{cases}$$

The inverse two-dimensional DCT is defined as:

$$I(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} c(u)c(v)J(u,v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \tag{2.84}$$

with,

$$c(\xi) = \begin{cases} \sqrt{\frac{1}{N}}, & \xi = 0 \\ \sqrt{\frac{2}{N}}, & \text{otherwise} \end{cases}$$

or by using the alternative formulation,

$$I(x,y) = \frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} c(u)c(v)J(u,v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \tag{2.85}$$

with,

$$c(\xi) = \begin{cases} \sqrt{\frac{1}{2}}, & \xi = 0 \\ 1, & \text{otherwise} \end{cases}$$

Fig. 2.16 DCT representation of a graylevel image

Fig. 2.17 Reconstruction using 50 of the DCT coefficients (0.05%) at 20.69 dB PSNR

A practical example of the application of DCT on a graylevel image is shown in Figs. 2.16, 2.17, 2.18, 2.19, and 2.20. The first image shows a representation of the original image and the transform coefficients after the transformation. The following images present gradual reconstructions using an increasing amount of transform coefficients.[5] Apparently, DCT is also good at producing compact representations of images as with only a small fraction of the coefficients it is able to produce a high quality reconstruction, as shown in Fig. 2.20, in which a reconstruction using just 35 % of the coefficients led to an image with 39.8 dBs of quality. It is noted that the black regions in the transform domain denote the coefficient being zeroed out. The diagonal scheme used is in line with the ordering of increasing frequency in two dimensions.

For natural images that usually exhibit a high correlation among neighboring pixels, the performance of the DCT approaches that of KLT. It can be shown that for a source that is modeled as a first-order Markovian, and since the correlation of

[5] It should be noted that in this example the transform has been applied once on the total image area, which is not the usual case. Usually, these transforms are being applied in a block-by-block basis, typically of 8×8 pixels.

Fig. 2.18 Reconstruction using 200 of the DCT coefficients (0.85%) at 24.88 dB PSNR

Fig. 2.19 Reconstruction using 546 of the DCT coefficients (6.3%) at 31.68 dB PSNR

Fig. 2.20 Reconstruction using 1296 of the DCT coefficients (35.6%) at 39.82 dB PSNR

adjacent pixels tends to unity, the DCT basis functions are similar to those of the KLT. Theoretically, one of the properties of random processes that are stationary in the wide sense (Wide-Sense Stationary (WSS)) is that their Fourier coefficients are uncorrelated. As the number of coefficients increases, DFT and DCT, as well as other similar transformations, in the frequency domain, *diagonalize the covariance matrix of the source*. Therefore, these transformations are asymptotically equivalent to the KLT for WSS sources, up to an extent of a special reordering of the transform coefficients (like the well known *zig-zag ordering* of the DCT coefficients during the application of JPEG compression—as will be described in following paragraphs). Although most of the images cannot be readily modeled as WSS processes, the DCT has proven to be a robust approximation to the KLT for natural images.

Following the definitions in previous paragraphs and in accordance with the definition of the inverse DCT in (2.85), the DCT basis functions (or basis images in this case) are defined as

$$b(u,v,x,y)_{x,y=0,\ldots,N-1} = c(u)c(v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \tag{2.86}$$

The basis functions of DCT for an image of size 8×8 pixels are shown in Fig. 2.21. Like the DFT, the DCT has a fast implementation with computational complexity $O(N\log N)$ for a transformation of N points. Nevertheless, DCT is more efficient in compression applications, as it does not exhibit the phenomenon of occurrence of spurious spectral components. The characteristic transformation matrices, as defined in (2.70), are calculated for the case of an 8×8 pixels image, are

$$\mathbf{h}_c = \mathbf{h}_r = \begin{pmatrix} 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 & 0.3536 \\ 0.4904 & 0.4157 & 0.2778 & 0.0975 & -0.0975 & -0.2778 & -0.4157 & -0.4904 \\ 0.4619 & 0.1913 & -0.1913 & -0.4619 & -0.4619 & -0.1913 & 0.1913 & 0.4619 \\ 0.4157 & -0.0975 & -0.4904 & -0.2778 & 0.2778 & 0.4904 & 0.0975 & -0.4157 \\ 0.3536 & -0.3536 & -0.3536 & 0.3536 & 0.3536 & -0.3536 & -0.3536 & 0.3536 \\ 0.2778 & -0.4904 & 0.0975 & 0.4157 & -0.4157 & -0.0975 & 0.4904 & -0.2778 \\ 0.1913 & -0.4619 & 0.4619 & -0.1913 & -0.1913 & 0.4619 & -0.4619 & 0.1913 \\ 0.0975 & -0.2778 & 0.4157 & -0.4904 & 0.4904 & -0.4157 & 0.2778 & -0.0975 \end{pmatrix} \tag{2.87}$$

The DFT is a mapping into a Discrete Fourier Series (DFS) of a sequence of finite length and therefore, exhibits an indirect periodicity, which is illustrated in Fig. 2.22a (Rabbani and Jones 1991a). This periodicity is the result of the sampling in the frequency domain. During the reconstruction of the original sequence in this way, discontinuities between the encoded parts emerge. These discontinuities lead to spurious high frequency components, which can cause significant degradation in the efficiency of the transformation. Though these spurious components are not part of the original sequence, they are required for reconstructing the boundaries in the periodic sequence.

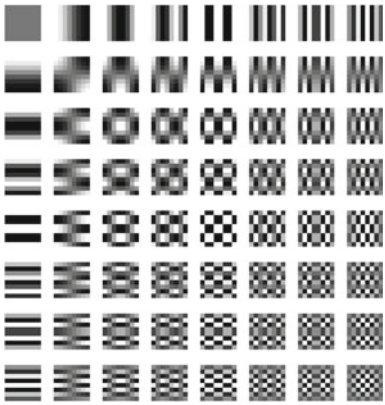

Fig. 2.21 The 8 × 8 2-D DCT basis images

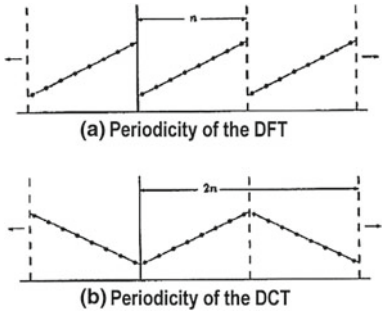

Fig. 2.22 Periodicities of **a** DFT and **b** DCT

Striving for improved efficiency of transformation through the rejection of these components leads to errors in the reconstruction at the region boundaries. In image coding, wherein the image is divided into non-overlapping parts to form two-dimensional sequences, such errors in the reconstruction result in the appearance of intense deformation at the boundaries between adjacent segments. To eliminate these inconsistencies in the boundaries, the original sequence of N points can be extended to a $2N$-point sequence by mirroring the original image at the boundaries. The extended sequence is repeated to form a periodic sequence that is required for the DFS representation. As shown in Fig. 2.22b, this periodic sequence exhibits no such discontinuities in the boundaries and no spurious spectral components in the DCT.

The process of calculating the $2N$-point DFT of the extended sequence (from the N-point original) is identical to the calculation of the DCT of the original sequence. Actually, a DCT can be computed as a $2N$-point FFT. Due to its symmetry DCT exhibits two main advantages over DFT, namely,

- The DCT does not generate spurious spectral components, and thus, the efficiency of coding remains high, while simultaneously, diminishing the *blocking artifacts* at the boundaries of the coding regions.

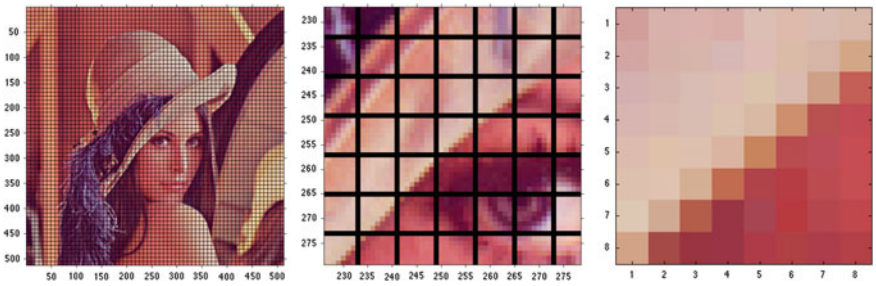

Fig. 2.23 Tiling of test image '*lena*' into 8 × 8 non-overlapping image blocks, zoomed-in portion of the image and a magnified selected image block

- The DCT calculations require only real numbers, while the DFT calculations involve complex numbers.

These advantages rendered DCT the most widely-used compression transform for both static and moving images (video). In overall, DCT can lead to efficient compression when the spatial correlation between neighboring pixels is significant, usually the case in natural images.

Finally, since the transform has been defined in 2-D and the corresponding basis images have been shown let us consider how the transform actually represents a color image. In practice, DCT (like many of the transforms) is being applied on non-overlapping image blocks, which simplifies the computations and make extremely fast parallel implementations possible. Figure 2.23 presents such a tiling on the color image '*lena*', which separates the image into 8 × 8 non-overlapping image blocks. Suppose during a sequential encoding process, the current block is the one in the pixel locations (250–257, 250–257) as shown in the figure. The values of the pixels in this image block, shown only for the green (G) channel,[6] are,

$$G = \begin{bmatrix} 157 & 176 & 177 & 174 & 190 & 186 & 187 & 183 \\ 166 & 177 & 180 & 183 & 187 & 188 & 181 & 167 \\ 177 & 181 & 188 & 189 & 194 & 185 & 160 & 95 \\ 181 & 184 & 189 & 193 & 185 & 143 & 81 & 74 \\ 188 & 192 & 192 & 179 & 133 & 75 & 77 & 76 \\ 193 & 195 & 175 & 108 & 67 & 61 & 74 & 77 \\ 200 & 169 & 93 & 53 & 69 & 57 & 70 & 72 \\ 163 & 74 & 52 & 53 & 59 & 65 & 61 & 65 \end{bmatrix}$$

[6]The green channel was selected because it is easy to follow; the upper left triangular region of the image, which is yelowish is expected to exhibit higher values in this channel than those in the rest of the block, which shows up redish.

Application of the transform on this image block and channel results in the matrix,

$$DCT_G = \begin{bmatrix} 1112 & 204 & 5 & 20 & 1 & -2 & -4 & 10 \\ 285 & -112 & -112 & -12 & -23 & -6 & -6 & -3 \\ -60 & -118 & 78 & 40 & 5 & 2 & 3 & 2 \\ 17 & 37 & 48 & -61 & -15 & -12 & -9 & 3 \\ -21 & -8 & -37 & -5 & 35 & 2 & 9 & -3 \\ 2 & 16 & 2 & 24 & -7 & -22 & -4 & 1 \\ -4 & -3 & -8 & -2 & -17 & 16 & 8 & 3 \\ 7 & 2 & 1 & 7 & 0 & 6 & -17 & 7 \end{bmatrix}$$

with the values rounded to the nearest integer for better visualization.

Apparently, the first coefficient (top-left) is significantly higher than all the other (in absolute values). So, what do actually these coefficient values represent? Clearly, they represent the weight (multiplier) of the appropriate basis image so that the weighted summation of all coefficients and basis images would approximate the original image. This is graphically depicted in Fig. 2.24 where above each of the basis images the corresponding weight has been added. This is equivalent to a series expansion, in which the image block is decomposed into,

$$I = \sum_{x=1}^{8} \sum_{y=1}^{8} c(x, y) \cdot b(x, y) = [c_1 \ \ldots \ c_{x \cdot y}] \begin{bmatrix} b_1 \\ \vdots \\ b_{x \cdot y} \end{bmatrix}$$

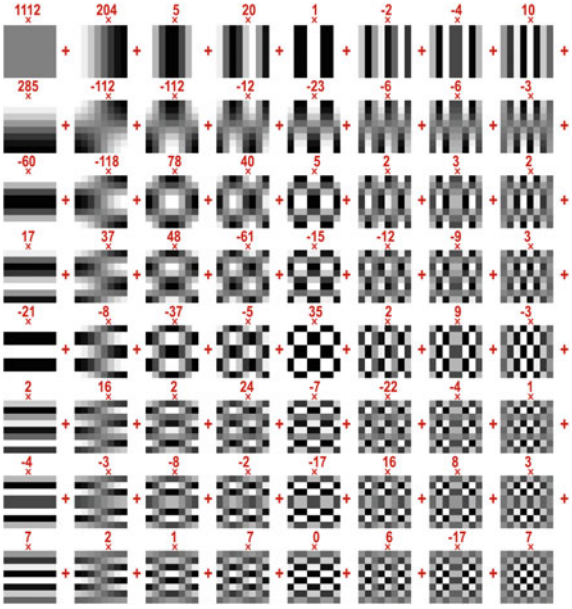

Fig. 2.24 DCT domain representation of image

where $c(x, y)$ the coefficient from the *DCT* matrix and $b(x, y)$ the basis image (Fig. 2.21). *Geometrically, this can be viewed as the representation in a linear image space, in which the basis images are the unit vectors of the axes and the coefficients are the coordinates.*

Four entirely different scenarios of reconstruction are being reviewed in the example depicted in Fig. 2.25. Specifically, the first scenario represents the reconstruction possible if only the first row of DCT coefficients matrix are being used. In the second scenario, the main diagonal coefficients have been selected and all others were discarded. In the third scenario the coefficients from the first row and first column of the matrix have been used for the reconstruction. Finally, for the fourth scenario the upper-left triangular part of the matrix has been used for a reconstruction. In all scenarios, Fig. 2.25 presents the original image block data, the reconstructed data using the limited number of coefficients and the difference between the two, with values exaggerated for emphasis. Each reconstruction is titled by the RMSE and the quality in PSNR dBs.

Discrete Sine Transform

The Discrete Sine Transform (DST) is defined similarly to the definition of DCT by the transformation matrix $\mathbf{C} = \{C(u, x)\}$,

$$C(u, x) = \sqrt{\frac{2}{N+1}} \sin \frac{(x+1)(u+1)\pi}{N+1} \tag{2.88}$$

The two-dimensional DST of a square $N \times N$ matrix $\mathbf{I}$ is defined as

$$\begin{aligned} J(u, v) &= \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} C(u, x) I(x, y) C(v, y) \\ J(u, v) &= \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I(x, y) \sqrt{\frac{2}{N+1}} \sin \frac{(x+1)(u+1)\pi}{N+1} \sqrt{\frac{2}{N+1}} \sin \frac{(y+1)(v+1)\pi}{N+1} \\ J(u, v) &= \frac{2}{N+1} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I(x, y) \sin \frac{(x+1)(u+1)\pi}{N+1} \sin \frac{(y+1)(v+1)\pi}{N+1} \end{aligned} \tag{2.89}$$

The inverse two-dimensional DST is defined as:

$$I(u, v) = \frac{2}{N+1} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} J(x, y) \sin \frac{(x+1)(u+1)\pi}{N+1} \sin \frac{(y+1)(v+1)\pi}{N+1} \tag{2.90}$$

A practical example of the application of DST on a graylevel image is shown in Figs. 2.26, 2.27, 2.28, 2.29, and 2.30. The first image shows a representation of the original image and the transform coefficients after the transformation. The following images present gradual reconstructions using an increasing amount of transform

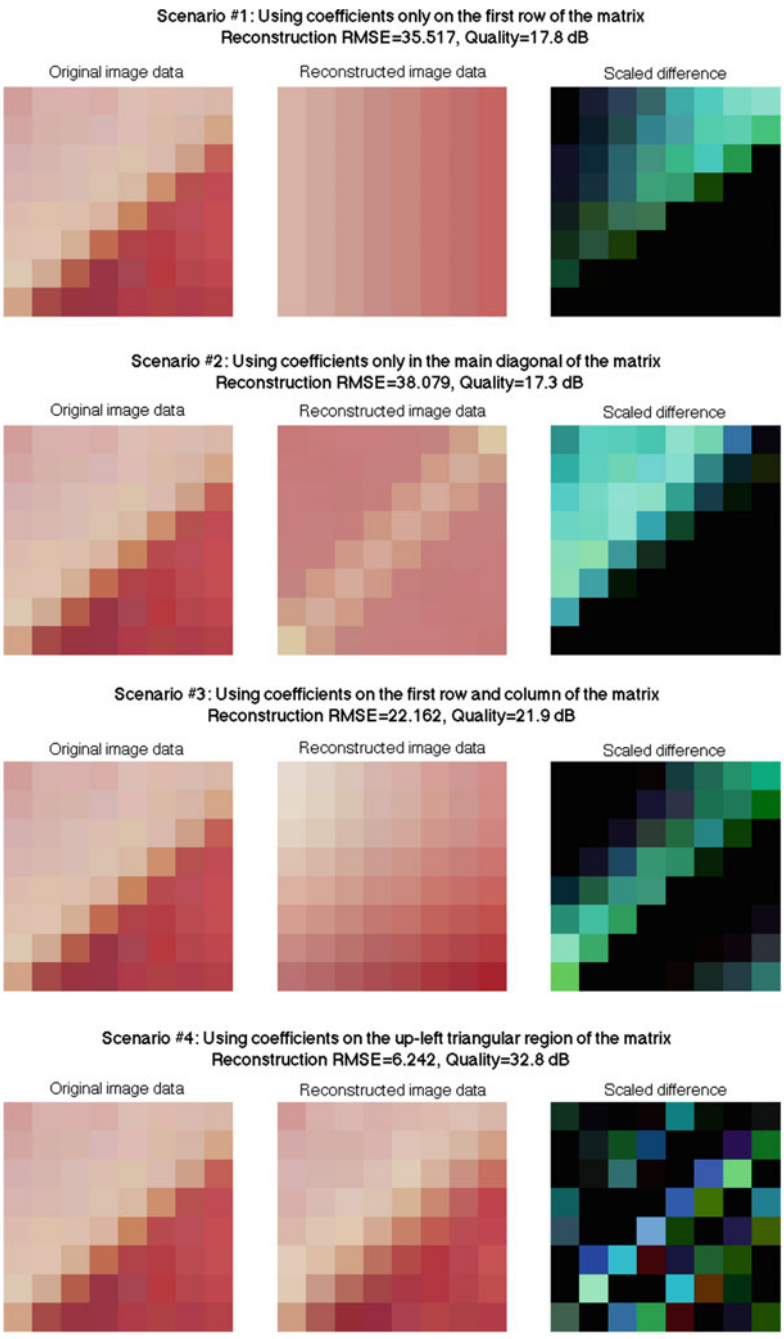

Fig. 2.25 Reconstructions of a block from image '*lena*' from a limited number of DCT coefficients

Fig. 2.26 DST representation of a graylevel image

Fig. 2.27 Reconstruction using 50 of the DST coefficients (0.05%) at 19.95 dB PSNR

Fig. 2.28 Reconstruction using 200 of the DST coefficients (0.85%) at 24.52 dB PSNR

Fig. 2.29 Reconstruction using 546 of the DST coefficients (6.3%) at 31.37 dB PSNR

Fig. 2.30 Reconstruction using 1296 of the DST coefficients (35.6%) at 39.82 dB PSNR

coefficients.[7] Apparently, DST is also good at producing compact representations of images as with only a small fraction of the coefficients it is able to produce a high quality reconstruction, as shown in Fig. 2.30, in which a reconstruction using just 35 % of the coefficients led to an image with 39.8 dBs of quality, very close the quality achieved in DCT. Also in this example, the black regions in the transform domain denote the coefficient being zeroed out using the usual diagonal scheme, in line with the ordering of increasing frequency in two dimensions.

In DST the 2-D basis functions (images) are

$$b(u, v, x, y)_{x,y=0,\ldots,N-1} = \frac{2}{N+1} \sin \frac{(x+1)(u+1)\pi}{N+1} \sin \frac{(y+1)(v+1)\pi}{N+1} \quad (2.91)$$

[7] Again, as in the case of the example given for DCT, the transform has been applied once on the total image area, which is not the usual case; in the usual case the transform is being applied in a block-by-block basis, typically of 8×8 pixels.

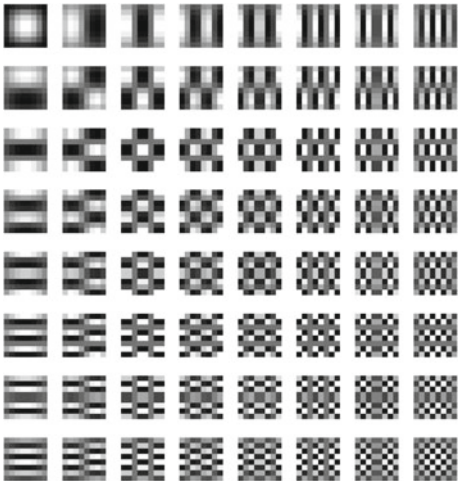

Fig. 2.31 The 8 × 8 2-D basis functions (images) of DST

A representation of these basis functions (basis images) appears in Fig. 2.31. The characteristic transformation matrix, as in the case of the DCT, is defined, for the case of an 8 × 8 pixels image, as

$$\mathbf{h}_c = \mathbf{h}_r = \begin{pmatrix} 0.1612 & 0.3030 & 0.4082 & 0.4642 & 0.4642 & 0.4082 & 0.3030 & 0.1612 \\ 0.3030 & 0.4642 & 0.4082 & 0.1612 & -0.1612 & -0.4082 & -0.4642 & -0.3030 \\ 0.4082 & 0.4082 & 0.0000 & -0.4082 & -0.4082 & 0.0000 & 0.4082 & 0.4082 \\ 0.4642 & 0.1612 & -0.4082 & -0.3030 & 0.3030 & 0.4082 & -0.1612 & -0.4642 \\ 0.4642 & -0.1612 & -0.4082 & 0.3030 & 0.3030 & -0.4082 & -0.1612 & 0.4642 \\ 0.4082 & -0.4082 & 0.0000 & 0.4082 & -0.4082 & 0.0000 & 0.4082 & -0.4082 \\ 0.3030 & -0.4642 & 0.4082 & -0.1612 & -0.1612 & 0.4082 & -0.4642 & 0.3030 \\ 0.1612 & -0.3030 & 0.4082 & -0.4642 & 0.4642 & -0.4082 & 0.3030 & -0.1612 \end{pmatrix} \tag{2.92}$$

Discrete Hartley Transform

The Discrete Hartley Transform (DHT) has been proposed by an alternative to the Fourier transforms by Bracewell (1983), based on a previous work by Hartley (1942) based on the DFT (2.75) as,

$$\mathbf{J} = Re\{\mathbf{J}_F\} - Im\{\mathbf{J}_F\} \tag{2.93}$$

where $\mathbf{J}$ the Hartley transformed signal, $\mathbf{J}_F$ the Fourier transformed signal, and $Re\{\}$ and $Im\{\}$ the real and imaginary parts of the transform. Formulating this equation using the notation in (2.75), requires to consider two approaches as pointed out by (Watson and Poirson 1986). These two approaches involve a slight variation in the definition of the basis functions (in the general case of $M \times N$ 2-D signals) as,

$$b(u,v,x,y) = \begin{cases} \cos\left(2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)\right) \\ \cos\left(\frac{2\pi ux}{M}\right) case \left(\frac{2\pi vy}{N}\right) \end{cases} \tag{2.94}$$

where $\cos(\theta) = \cos(\theta)+\sin(\theta)$, which has been shown to be an orthogonal function. The first definition gives the following definition for the transform,

$$J(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} I(x,y)\cos\left(\frac{2\pi ux}{M}+\frac{2\pi vy}{N}\right) \tag{2.95}$$

The alternative definition of the basis functions gives rise to the following definition of the transform,

$$J(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} I(x,y)\cos\frac{2\pi ux}{M}\cos\frac{2\pi vy}{N} \tag{2.96}$$

The inverse transform is defined as,

$$I(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} J(u,v)\cos\frac{2\pi ux}{M}\cos\frac{2\pi vy}{N} \tag{2.97}$$

which is typically a separable 2-D transform and was the preferred representation by Hartley. Apparently, the Hartley Transform is equivalent to the typical Fourier Transform and the choice between them is a matter of the application.

A representation of the basis functions (basis images) for an 8×8 transform appears in Fig. 2.32. An example of the application of DHT is shown in Figs. 2.33 and 2.34 for the case of a graylevel image. The coefficients in Fig. 2.33 have been shifted to the centered of the spatial range exactly as in the case of the DFT. The three-dimensional representation of the transform coefficients magnitude shown in Fig. 2.34 reveals how this transform concentrates the image energy near the mean value and practically only in low spatial frequencies, achieving a rather compact representation. This graph shows the pseudo-colored logarithm of the coefficient magnitudes for better representation.

Walsh-Hadamard Transform

There is a family of transformations, which generalize the Fourier transforms, in which the basis functions are non-sinusoidal. The are all based on discretized versions of *Walsh functions*,[8] to formulate the transformation matrices $\mathbf{h}_c$ and $\mathbf{h}_r$. These matrices may be defined in various ways, the simpler being by the use of *Hadamard matrices*,

[8] A family of special piecewise constant functions assuming only the two values, ± 1.

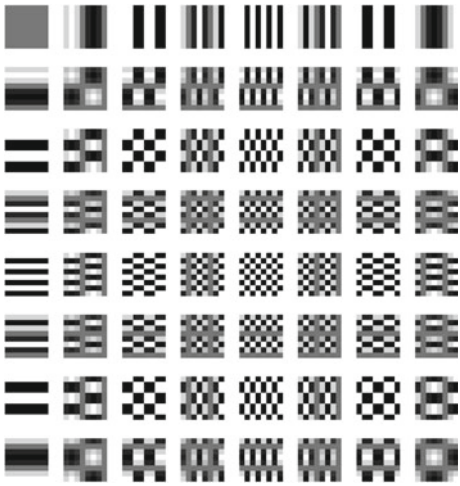

Fig. 2.32 The 8×8 2-D basis functions (images) of DHT

Fig. 2.33 Example of the application of DHT on a gray level image

$$\mathbf{H}_{2^n} = \sqrt{\frac{1}{2}} \begin{pmatrix} \mathbf{H}_{2^{n-1}} & \mathbf{H}_{2^{n-1}} \\ \mathbf{H}_{2^{n-1}} & -\mathbf{H}_{2^{n-1}} \end{pmatrix}, \quad \mathbf{H}_0 = 1, \quad n > 0 \tag{2.98}$$

where the $\sqrt{\frac{1}{2}}$ normalization factor is sometimes omitted. This definition produces $2^n \times 2^n$ matrices for $n > 0$ with all elements ± 1 and a normalization factor that changes as $1/2^{\frac{n}{2}}$, so as to have the elements defined as

$$(H_{2^n})_{i,j} = \frac{1}{2^{\frac{n}{2}}}(-1)^{i \cdot j} \tag{2.99}$$

with the dot product being the bitwise dot product of the binary representations of i and j. The rows of such a matrix are Walsh functions.

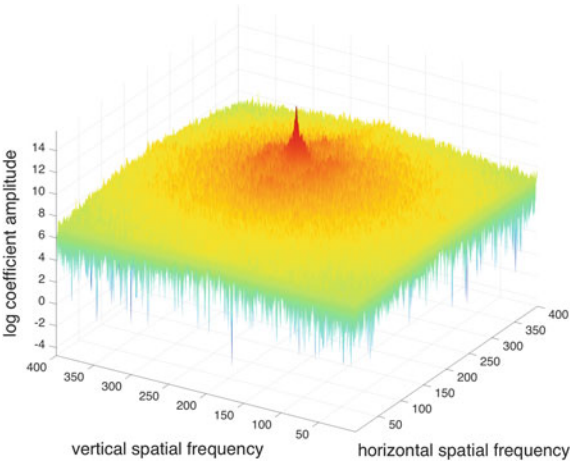

Fig. 2.34 Three-dimensional pseudo-colored representation of the DHT transform coefficients

According to this definition,

$$\mathbf{H}_2 = \sqrt{\frac{1}{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \mathbf{H}_4 = \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \tag{2.100}$$

Scaling up to get the typical 8×8 transformation matrix for $n = 3$ produces,

$$\mathbf{h}_c = \mathbf{h}_r = \mathbf{H}_8 = \frac{1}{2\sqrt{2}}\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix} \tag{2.101}$$

A 'curiosity' in the formation of the basis images (the 2-D case) is that a reordering of the computed function should be imposed in order to get the basis images in gradually incrementing spatial frequency in both dimensions. This reordering is imposed on the bit-levels of the coordinates—that is all coordinates are converted to their binary representation and everything is computed on their bits using a right-hand MSB notation—(leaving out the scaling factor) as

$$(H_{2^n})_{u,v,x,y} = (-1)^{\sum_{j=0}^{b-1}\left(g(u_j)x_j + g(v_j)y_j\right)} \quad \text{with} \quad g(\xi_j) = \begin{cases} \xi_0, \\ \xi_{b-1} + \xi_{b-2} \\ \vdots \quad \vdots \\ \xi_0 + \xi_1 \end{cases} \tag{2.102}$$

where b are the total bitplanes needed to describe the numbers involved and all summations are considered in modulo-2 arithmetic, that is the modulo-2 is taken after each summation. If the binary representations are taken as column vectors then these summations are transformed to dot products as,

$$(H_{2^n})_{u,v,x,y} = (-1)^{\left(g(\mathbf{u})^T \cdot \mathbf{x} + g(\mathbf{v})^T \cdot \mathbf{y}\right) \bmod 2} \tag{2.103}$$

with g defined as in (2.102) and the reordering imposed on the elements of the corresponding vectors.

A representation of the basis images appears in Figs. 2.35 and 2.36 for the 4×4 and 8×8 transform respectively. A practical example of the application of WHT on a graylevel image is shown in Figs. 2.37, 2.38, 2.39, 2.40, 2.41 and 2.42. The first image shows a representation of the original image and the transform coefficients produced by the transformation, whereas the other images present gradual reconstructions using an increasing amount of transform coefficients. As in the previous examples, the black regions in the transform domain denote the coefficient being zeroed out.

Due to the computational simplicity of Walsh-Hadamard Transform (WHT) in comparison to Fourier transforms, since the WHT does not involve multiplications or divisions (only ± 1 factors), this transform is significantly more appealing in applications that have access to low computational resources. It has been used for image compression when the computational power of computer was not strong enough to support images of high dimensions in limited time. NASA made heavy use of the WHT during the 1960s and early 1970s space missions for photo compression, 'real-time' video transmission and error correction over unreliable channels.

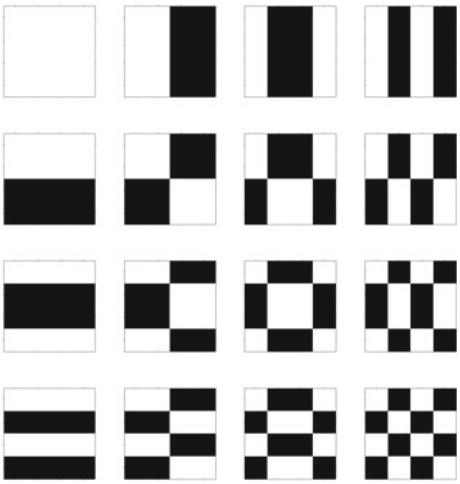

Fig. 2.35 The 4×4 2-D basis images of WHT

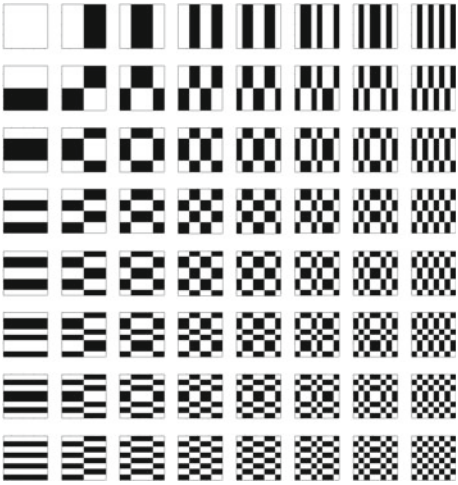

Fig. 2.36 The 8×8 2-D basis images of WHT

Fig. 2.37 WHT representation of a graylevel image

Fig. 2.38 Reconstruction using 2% of the WHT coefficients at 23.75 dB PSNR

Fig. 2.39 Reconstruction using 8% of the WHT coefficients at 27.09 dB PSNR

Fig. 2.40 Reconstruction using 18% of the WHT coefficients at 30.27 dB PSNR

Fig. 2.41 Reconstruction using 32% of the WHT coefficients at 32.52 dB PSNR

Fig. 2.42 Reconstruction using 50% of the WHT coefficients at 36.79 dB PSNR

Haar Transform

Alfred Haar in his seminal 1910 *theory of orthogonal function systems* (Haar 1910), defined the so called *orthogonal function system* χ, as the most representative of the class of orthogonal systems as,

$$\begin{aligned}
\chi_0(s) &= 1, \quad \forall s \in [0,1] \\
\chi_1(s) &= \begin{cases} 1 & 0 \le s < \frac{1}{2} \\ -1 & \frac{1}{2} \le s < 1 \end{cases} \\
\chi_2^{(1)}(s) &= \begin{cases} \sqrt{2} & 0 \le s < \frac{1}{4} \\ -\sqrt{2} & \frac{1}{4} < s < \frac{1}{2} \\ 0 & \frac{1}{2} < s \le 0 \end{cases} \\
\chi_2^{(2)}(s) &= \begin{cases} 0 & 0 \le s < \frac{1}{2} \\ \sqrt{2} & \frac{1}{2} < s < \frac{3}{4} \\ -\sqrt{2} & \frac{3}{4} < s \le 0 \end{cases}
\end{aligned} \tag{2.104}$$

so, by dividing the interval $[0, 1]$ into 2^n equal parts and by denoting these subintervals by $i_n^{(1)}, i_n^{(2)}, \ldots, i_n^{(2^n)}$,

$$\begin{aligned}
\chi_n^{(k)}(s) &= \begin{cases} 0 & s \in i_n^{(1)}, i_n^{(2)}, \ldots, i_n^{(2k-2)} \\ \sqrt{2^{n-1}} & s \in i_n^{(2k-1)} \\ -\sqrt{2^{n-1}} & s \in i_n^{(2^k)} \\ 0 & s \in i_n^{(2k+1)}, \ldots, i_n^{(2^n)} \end{cases} \\
k &= 1, 2, \ldots, 2^{n-1}
\end{aligned} \tag{2.105}$$

At points 0 and 1, each function $\chi_n^{(k)}(s)$ is assigned the value it assumes in the intervals $\left[0, \frac{1}{2^n}\right]$ or $\left[1 - \frac{1}{2^n}, 1\right]$. According to this definition, $\chi_n^{(k)}(s)$ is a piecewise constant

function with discontinuities only in the points $\frac{2k-2}{2^n}$, $\frac{2k-1}{2^n}$, $\frac{2k}{2^n}$, where $\chi_n^{(k)}(s)$ assumes (by agreement) a value equal to the arithmetic means of the values in the intervals adjoining at those points. Using this definition, infinitely many function χ form *a complete orthogonal function system*. Haar proved that *this system is orthogonal and complete*.

The Haar Transform (HT) was developed as a family of transformations, in which the characteristic transformation tables $\mathbf{h}_c$ and $\mathbf{h}_r$ are discretized forms of *Haar functions*. The HT is essentially a simple compression process. In one dimension, the HT transforms a vector of two elements $\mathbf{x} = (x_1\ x_2)^T$ to a vector $\mathbf{y} = (y_1\ y_2)^T$, as follows

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \mathbf{H} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \text{ where } \mathbf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.106}$$

or, in simple form $\mathbf{y} = \mathbf{H} \cdot \mathbf{x}$, and according to the definition of $\mathbf{H}$, y_1 and y_2 are the sum and the difference of x_1 and x_2, divided by $\sqrt{2}$ for energy conservation. It should be noted that the matrix $\mathbf{H}$ is orthonormal (comprises of orthogonal unit-length vectors), and therefore, $\mathbf{H}^{-1} = \mathbf{H}^T = \mathbf{H}$ (as $\mathbf{H}$ is symmetric, $\mathbf{H}^T = H$). Therefore, the inverse transformation can be solved by (2.106) as follows

$$\begin{gathered} \mathbf{y} = \mathbf{H} \cdot \mathbf{x} \Leftrightarrow \mathbf{H}^{-1} \cdot \mathbf{y} = \mathbf{H}^{-1} \cdot \mathbf{H} \cdot \mathbf{x} \Leftrightarrow \mathbf{H}^T \cdot \mathbf{y} = \mathbf{I} \cdot \mathbf{x} \Leftrightarrow \mathbf{H} \cdot \mathbf{y} = \mathbf{x} \\ \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \mathbf{H}^T \cdot \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \text{ where } \mathbf{H}^T = \mathbf{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \end{gathered} \tag{2.107}$$

In two dimensions, $\mathbf{x}$ and $\mathbf{y}$ are 2×2 matrices. Since the HT is separable it can be applied first to the columns and then the rows of x. The relation which expresses the process is $\mathbf{y} = \mathbf{H} \cdot \mathbf{x} \cdot \mathbf{H}^T$, and the inverse HT becomes $\mathbf{x} = \mathbf{H}^T \cdot \mathbf{y} \cdot \mathbf{H}$. Specifically, what happens during the HT is that given the input matrix $\mathbf{x}$

$$\mathbf{x} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{2.108}$$

then the output of the transform is

$$\mathbf{y} = \frac{1}{2} \begin{pmatrix} a+b+c+d & a-b+c-d \\ a+b-c-d & a-b-c+d \end{pmatrix} \tag{2.109}$$

Equations (2.108) and (2.109) correspond to the following filtering procedures (which take into account the halving factor):

- Top-Left: $a + b + c + d$ = mean value or a 2D low-pass filter (LL—Low in both dimensions)
- Top-Right: $a - b + c - d$ = average horizontal gradient or horizontal high-pass and vertical low-pass filter (HL)
- Bottom-Left: $a + b - c - d$ = average vertical gradient or horizontal low-pass and vertical high-pass filter (LH)

- Bottom-Right: $a-b-c+d$ = diagonal gradient or 2D high-pass filter (HH—High in both dimensions)

An example of the HT on a graylevel image is shown in Fig. 2.43, where only one step in the overall possible decomposition is being depicted. Apparently, the application of the transform results in a band-based decomposition of the original image. One step of the transform produces four bands, which correspond to a low-pass filtered replica of the original image (upper-left quarter) usually denoted as the LL band, and three high-pass filtered replicas (the three dark quarter images) that capture the horizontal (HL), vertical (LH) and diagonal (HH) high spatial frequencies. The three high frequency bands are displayed in log-scale for better illustration. Figure 2.44 shows the histograms of all these images. Apparently the histogram of the LL band is very similar to that of the original image, whereas the histograms of the high frequency bands are similar to two-sided geometric distributions centered

Fig. 2.43 The result of the application of a 1-step HT on a digital image

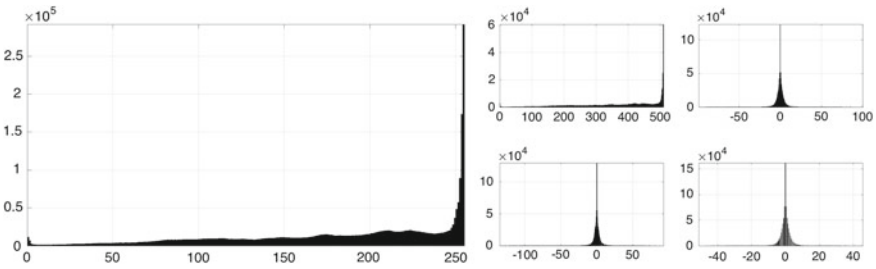

Fig. 2.44 The histograms of the images in Fig. 2.43

Table 2.3 Energy levels per image band after HT

LL (%)	HL (%)	LH (%)	HH (%)
96.5	2.2	0.9	0.4

around zero. The energy is unevenly distributed among the transform bands, as shown in Table 2.3, which lists the relative energy levels in each band normalized to the total energy of the image.

In order to create transform matrices of any size, one has to begin with the Haar functions,

$$h_{p,q}(x) = \frac{1}{\sqrt{N}} \begin{cases} 1 & p = q = 0,\ x \in [0,1] \\ 2^{\frac{p}{2}} & \frac{q-1}{2^p} \le x < \frac{q-\frac{1}{2}}{2^p} \\ -2^{\frac{p}{2}} & \frac{q-\frac{1}{2}}{2^p} \le x < \frac{q}{2^p} \\ 0 & \text{otherwise for } x \in [0,1] \end{cases} \tag{2.110}$$

where,

$$\begin{aligned} &0 \le p \le n-1 \\ &q = 0, 1 \quad p = 0 \\ &1 \le q \le 2^p \quad p \ne 0 \\ &N = 2^n \end{aligned} \tag{2.111}$$

By discretizing the values of x at $m/N,\ m = 0, ..., N-1$ these functions produce the transformation matrices. This way, the characteristic 8×8 transformation matrix, as defined in previous cases, is given for $N = 8$ as

$$\mathbf{h}_c = \mathbf{h}_r = \frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 \end{pmatrix} \tag{2.112}$$

Wavelet Transform

Supposing a real or complex function $\psi(t)$ with the following properties:

- the integral of the function equals zero:

$$\int_{-\infty}^{\infty} \psi(t)dt = 0 \tag{2.113}$$

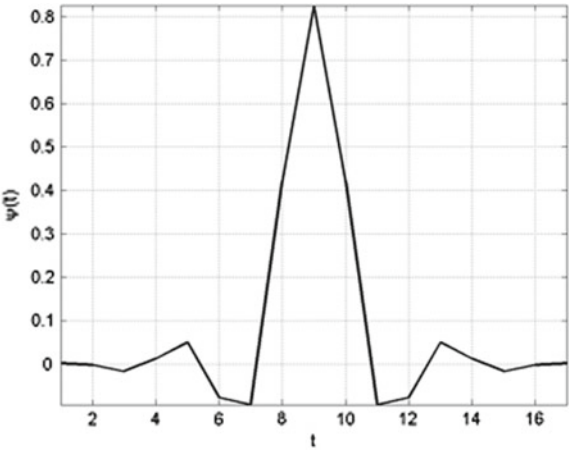

Fig. 2.45 The biorthogonal 6.8 wavelet

- the function is square integrable (or has finite energy):

$$\int_{-\infty}^{\infty} |\psi(t)|^2 dt < \infty \tag{2.114}$$

- the function satisfies the convention:

$$C \equiv \int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega \tag{2.115}$$

where $\Psi(\omega)$ the Fourier transform of $\psi(t)$, then the function $\psi(t)$ is a *mother wavelet or simply a wavelet*.

And while the assumption expressed in (2.115) is useful for the formulation of the inverse transformation, the first two conditions are sufficient for the definition of the Continuous-time Wavelet Transform (CWT) and explain the reason why the function is called wavelet (wavelet). The first condition (2.113) contains the information that the function itself is characterized as an oscillation or has a wavy nature. Unlike a continuous sinusoidal function, it is a 'small' wave. The second condition (2.114) ensures that most of the function energy is limited to a finite temporal duration. These two conditions are satisfied relatively easy and there is a multitude of functions that are suitable as mother wavelets (Rao and Bopardikar 1998).

Figure 2.45 illustrates the 'famous' *biorthogonal 6.8* wavelet.[9] This wavelet takes values in a closed region, i.e. it has a finite duration. On the contrary, there is the possibility of infinite duration wavelets, such as the *Morlet wavelet*, as illustrated in Fig. 2.46. This wavelet is created through the modulation of a cosine function by a Gaussian function. While it is of infinite duration, most of the energy is confined to

[9] 'Famous' due to its acceptance and usage in image compression applications, like in the Joint Photographic Experts Group 2000 (JPEG2000) image compression standard.

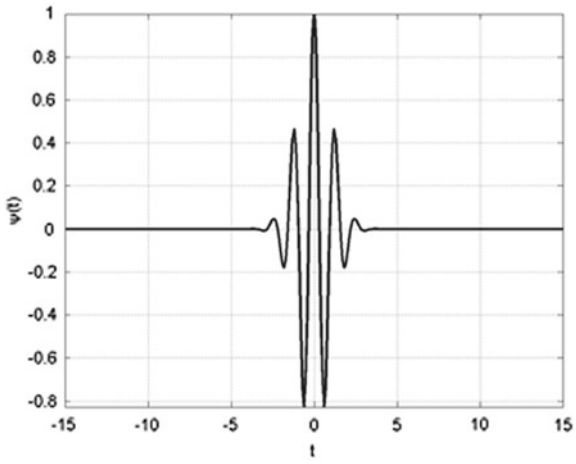

Fig. 2.46 The Morlet wavelet

a finite duration. As shown in the graph, more than 99 % of the energy is confined within the time frame $|t| \leq 2.5$ s (Rao and Bopardikar 1998).

Consider a square integrable function $f(t)$. The CWT of this function relative to the wavelet is defined (Rao and Bopardikar 1998)

$$W(a,b) \equiv \int_{-\infty}^{\infty} f(t) \frac{1}{\sqrt{|a|}} \psi^* \left(\frac{t-b}{a} \right) \mathrm{d}t \tag{2.116}$$

where a and b are real numbers and * indicates the complex conjugate. Accordingly, this transformation is a function of two variables. It should be noted that both $f(t)$ and $\psi(t)$ belong to the $L^2(R)$ space of the square integrable functions, which is called the *set of energy signals*. Defining:

$$\psi_{a,b}(t) \equiv \frac{1}{\sqrt{|a|}} \psi^* \left(\frac{t-b}{a} \right) \tag{2.117}$$

(2.116) can be written as:

$$W(a,b) \equiv \int_{-\infty}^{\infty} f(t) \psi_{a,b}^*(t) \mathrm{d}t \tag{2.118}$$

It should also be noted that:

$$\psi_{1,0}(t) = \psi(t) \tag{2.119}$$

The normalization factor $1/\sqrt{|a|}$ guarantees that the energy remains constant for all a and b, so

$$\int_{-\infty}^{\infty} |\psi_{a,b}(t)|^2 \mathrm{d}t = \int_{-\infty}^{\infty} |\psi(t)|^2 \mathrm{d}t \tag{2.120}$$

for every a and b. For a given value of a, the function $\psi_{a,b}(t)$ expresses the translation (or shifting) of $\psi_{a,0}(t)$ by b on the horizontal time axis. In addition, from

$$\psi_{a,0}(t) \equiv \frac{1}{\sqrt{|a|}} \psi^* \left(\frac{t}{a} \right) \tag{2.121}$$

can be concluded that $\psi_{a,0}(t)$ is an extension in time and scale (magnitude) of $\psi(t)$. Since a determines the size of the time extension or expansion, it is called the expansion or scale variable. When $a > 1$, $\psi(t)$ is expanded on the time axis, and when $0 < a < 1$ it is retracted. Negative values of a result in a time reversal while scaling. Thus, since the CWT is created by scaling and shifting of a function, the wavelet transform is called mother wavelet (Rao and Bopardikar 1998). It should be emphasized that the biggest advantage of Wavelet-based transforms over Fourier-based transforms is that they *provide a better time and frequency domain localization*, obeying, of course, the uncertainty principle, as expressed in time-frequency analysis

$$\Delta t_\psi \Delta \omega_\psi = c_\psi \tag{2.122}$$

where c_ψ is a constant dependent on the wavelet used, Δt_ψ the mean squared duration and $\Delta \omega_\psi$ the mean squared frequency range. It is understood that the smaller the value of the constant c_ψ the more accurate time-frequency analysis can this wavelet achieve (Rao and Bopardikar 1998).

Finally, when the third condition of the mother wavelet, as expressed by Eq. (2.115) is satisfied for $0 < C < \infty$, then the inverse CWT is defined as

$$f(t) = \frac{1}{C} \int_{a=-\infty}^{\infty} \int_{b=-\infty}^{\infty} \frac{1}{|a|^2} W(a,b) \psi_{a,b}(t) \mathrm{d}a \mathrm{d}b \tag{2.123}$$

It should of course be noted that this is a sufficient but not a necessary condition for achieving a mapping of all CWT in $L^2(R)$. To discretize CWT, the following representation is adopted

$$f(t) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} d(k,l) 2^{-k/2} \psi(2^{-k} t - 1) \tag{2.124}$$

It is worth noting that unlike Eq. (2.123), in which continuous shifts and scalings are involved, Eq. (2.124) involves discrete values. Scaling takes values of the form

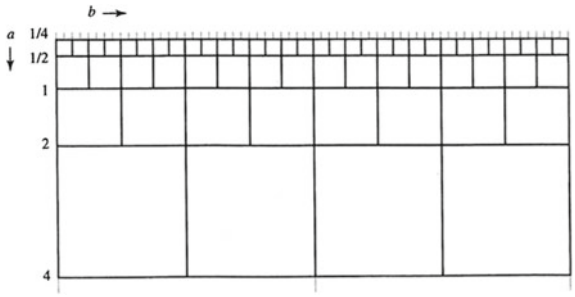

Fig. 2.47 The 'cells' of time-frequency analysis that correspond to a dyadic sampling

$a = 2^k$ (where k is an integer), whereas for each scaling 2^k, the shift takes values of the form $b = 2^k l$ (where l is an integer). Thus, the values of $d(k, l)$ relate to values of the transformation at $a = 2^k$ and $b = 2^k l$, which corresponds to sampling the coefficients (a, b) on a grid, as shown in the Fig. 2.47. This process is called *dyadic sampling* because successive discrete values of scale and the corresponding sampling intervals differ by a factor of two (2) (Rao and Bopardikar 1998).

The two-dimensional sequence $d(k, l)$ is called the Discrete Wavelet Transform (DWT) of $f(t)$. As is evident, the DWT remains a transformation of a continuous signal. The discretization refers only to the variables a and b. In this sense, it is analogous to Fourier series, which represent in a discrete frequency domain (periodic) a continuous (in time) signal. For this reason, the DWT is referred to as *Continuous-Time Wavelet Series*.

For the application of Wavelet Transform (WT) in images, a two-dimensional transformation is required. It turns out, however, that the WT is separable, and therefore, it is possible to sequentially apply two one-dimensional transformations (e.g., first on the rows and then the columns of the image), which greatly simplifies the calculations. Key features of use of the WT in compression are (Rao and Bopardikar 1998):

- it involves the idea of the multi-resolution image representation
- it applies (or at least could apply) to the entire image and, therefore, it does not suffer from blocking artifacts (evident in DCT)
- it can be used (by applying integer wavelet filter coefficients) for simultaneous lossless and lossy coding, and embedding of both in the same output file
- it can provide decomposition of the image in spectral bands, where each band can be quantized according to the importance of the visual content

The one-dimensional transform is implemented by applying two analysis filters followed by downsampling, as shown in Fig. 2.48. The one-dimensional sample sequence is filtered by a low pass and a high pass filter, is then sub-sampled by a factor of 2, which ultimately results in a representation of a low and a high frequency band. Of course, the inverse transform is the exact reverse procedure (Rabbani and Joshi 2002; Rabbani and Cruz 2001).

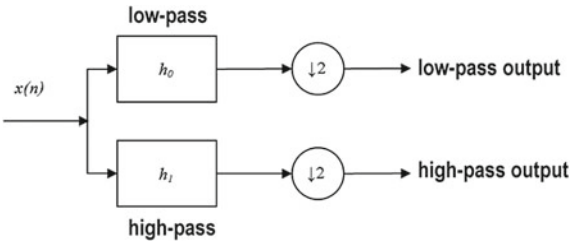

Fig. 2.48 Analysis filter bank

Two methods leading to the same result can be applied to compute the transformation (Rabbani and Joshi 2002; Rabbani and Cruz 2001):

- *convolution*
- *lifting*: an alternative method to compute the transform coefficients using the following three-step process:

1. *separation*: initially the input signal is divided into two sequences, taking samples in even locations (or times) for one sequence and samples in odd locations for the other. This step is commonly called the *lazy wavelet transform*
2. *prediction and update*: assuming that the input signal shows a significant correlation between successive samples, one may deduce with some certainty that an even sample can predict the value of the neighboring odd sample. Then the output is updated with corrections in the prediction that preceded. The process in this step can be expressed as:

$$(s_{odd_{i-1}}, s_{even_{i-1}}) \leftarrow S(s_i)s_{odd_{i-1}} - = P(s_{even_{i-1}})s_{even_{i-1}} + = U(s_{odd_{i-1}}) \quad (2.125)$$

   and graphically represented in the Fig. 2.49, and can be textually description as follows:
   - separation S of the input into even and odd samples
   - prediction P of odd samples from previous even samples
   - update U of predicted value
3. *normalization*: at the final stage, the output of the previous step is normalized to form the final transform coefficients.

Figure 2.50 shows the result of applying the DWT on an image. The two steps are shown, in which the transformation is applied sequentially due to the *separability*. During the first step the DWT is applied to the columns of the original image resulting in an image like the one shown in Fig. 2.50a. At the second step, the transformation is applied to the rows of the image produced by the first step resulting in the image shown in Fig. 2.50b. It should be noted that the values of the parts of the images that reflect the high frequency content (the 'noisy' gray regions) have been normalized to allow a better representation.

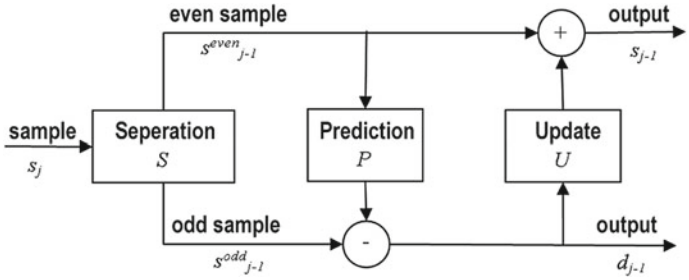

Fig. 2.49 Representation of the lifting method for WT

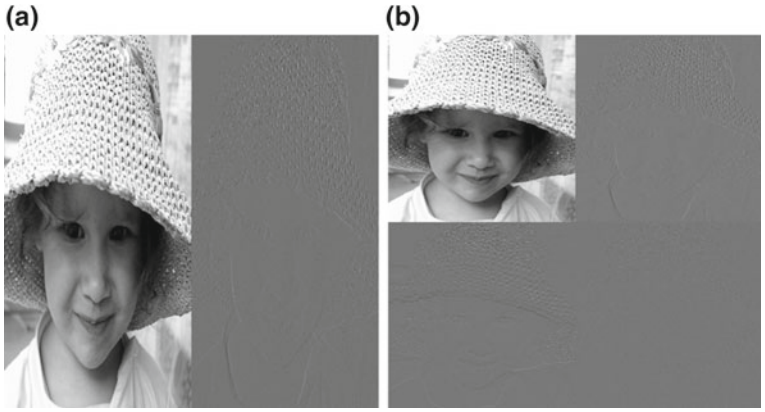

Fig. 2.50 DWT as a separable image transformation: **a** the transformation on the columns of the image and **b** the transformation on the rows of the image after the first step

Statistics in the Transform Domain

Both Pratt (1991) and Jain (1988) in their analysis of the image transforms provide further insight on the transform domain statistics in an attempt to support the need for those transforms in order to decorrelate the input data before further processing for compression. In this analysis, the *first and second moments* of the transform coefficients have been analyzed. If a square image **I** of size $N \times N$ is considered to be a two-dimensional ergodic process with known mean and covariance function then its unitary transform

$$J(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I(x, y) T(x, y; u, v) \tag{2.126}$$

is also a stochastic process with a mean value

$$E\{J(u,v)\} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} E\{I(x,y)\}\, T(x,y;u,v) \tag{2.127}$$

with $E\{I(x,y)\}$ being the mean of $I(x,y)$. Thus, the covariance function can be written as

$$\Sigma_J(u_1,v_1;u_2,v_2) = \sum_{x_1}\sum_{x_2}\sum_{y_1}\sum_{y_2} \Sigma_I(x_1,y_1;x_2,y_2) T(x_1,y_1;u_1,v_1) T^*(x_2,y_2;u_2,v_2) \tag{2.128}$$

where $\Sigma_I(x_1,y_1;x_2,y_2)$ is the covariance function of $I(x,y)$. Also, the variance function of $J(u,v)$ is

$$\sigma_J^2(u,v) = \Sigma_J(u,v;u,v) \tag{2.129}$$

These equations can be simplified in notation by adopting a matrix representation, thus

$$\begin{aligned} \mathbf{J} &= \mathbf{T}\mathbf{I} && \text{is the unitary transform} \\ \mathbf{m}_J &= \mathbf{T}\mathbf{m}_I && \text{is the mean of the transform coefficients} \\ \mathbf{\Sigma}_J &= \mathbf{T}\mathbf{\Sigma}_I\mathbf{T}^{*T} && \text{is the covariance matrix of the coefficients} \\ \mathbf{V}_J &= diag\,[\mathbf{\Sigma}_J] && \text{is the vector of variances of the coefficients} \end{aligned} \tag{2.130}$$

Analysis of the performance of KLT reveals that this is the only unitary transform that performs a complete decorrelation for an arbitrary image achieving the best energy compaction of all unitary transforms (Pratt 1991). Apparently this is due to the fact that the variance function of the KLT coefficients equals the corresponding eigenvalue, that is

$$\sigma_J^2(u,v) = \lambda(u,v) \tag{2.131}$$

Transformations other than the KLT result in at least some residual correlation between the transform coefficients, thus

$$\sum_{w=0}^{W} \lambda(w) \geq \sum_{w=0}^{W} \sigma^2(w) \quad W < N^2 \tag{2.132}$$

In addition, in order to derive analytic representations for the first and second moments of the transform coefficients of arbitrary image transforms, it is usually assumed that images are WSS; thus the mean value of an image is a constant $E\{\mathbf{I}\}$ and the covariance function assumes the functional form $\Sigma_I(x_1 - x_2, y_1 - y_2)$. In addition, if the transform is orthogonal the summation $\sum_x \sum_y T(x,y;u,v)$ yields zero for all non-zero-th basis functions

$$E\{J(u,v)\} = E\{\mathbf{I}\}\sum_x\sum_y T(x,y;u,v) = 0 \quad \forall u,v \neq 0 \tag{2.133}$$

Even though the input image is considered to be WSS this is not expected to be true for the transform coefficients, unless the transform kernel is space invariant. It can be shown that the Fourier transform is such a transform and an analytic representation for the first and second moments is possible, whereas for Hadamard, Haar and other transforms no closed form expressions have been developed for the covariance functions of those transforms.

Furthermore, based on the *central limit theorem* it is possible to approximate the probability densities of the transform coefficients by adopting a typical Gaussian distribution with the previously mentioned first and second moments. Thus the probability densities of the Fourier coefficients can be defined,

$$\begin{aligned} p\left(Re\left\{J_F(u,v)\right\}\right) &= \sqrt{2\pi\sigma_J^2(u,v)}\, e^{\frac{-Re\left\{J_F^2(u,v)\right\}}{2\sigma_J^2(u,v)}} \\ p\left(Im\left\{J_F(u,v)\right\}\right) &= \sqrt{2\pi\sigma_J^2(u,v)}\, e^{\frac{-Im\left\{J_F^2(u,v)\right\}}{2\sigma_J^2(u,v)}} \end{aligned} \tag{2.134}$$

where $Re\{\}, Im\{\}$ represent the real and imaginary parts of the transform. The assumed Gaussian probability density of the real and imaginary parts imply that the magnitude of the transform is modeled by a Rayleigh distribution and the phase is modeled by a uniform distribution, that is

$$\begin{aligned} p\left(\|J_F(u,v)\|\right) &= \frac{\|J_F(u,v)\|}{\sigma_J^2(u,v)}\, e^{\frac{-\|J_F^2(u,v)\|}{2\sigma_J^2(u,v)}} \\ p\left(\angle J_F(u,v)\right) &= \frac{1}{2\pi} \qquad -\pi \le \angle J_F(u,v) \le \pi \end{aligned} \tag{2.135}$$

The probability density of coefficients of unitary transforms other than the Fourier transform are usually modeled by Gaussian or Laplacian densities as follows,

$$\begin{aligned} p\left(J(u,v)\right) &= \sqrt{2\pi\sigma_J^2(u,v)}\, e^{\frac{-J^2(u,v)}{2\sigma_J^2(u,v)}} \\ p\left(J(u,v)\right) &= \sqrt{2\pi\sigma_J^2(u,v)}\, e^{\frac{-\sqrt{2}|J(u,v)|}{\sigma_J(u,v)}} \end{aligned} \tag{2.136}$$

To illustrate the effect of the image transforms and to provide a representation of the covariance matrices of the transform coefficients Fig. 2.51 displays the covariance matrices of the transform coefficients of various image transforms. In order to make the illustration printable and readable the logarithm of the covariances of the coefficients are being displayed. In all these graphical representations of covariance matrices, a more distinguishable and highly contrasted diagonal denotes a better decorrelation by the corresponding transform. The figure includes the original image and the covariance matrix of the initial image pixels, which exhibit high correlation as indicated by the multiple bright regions throughout the whole surface of its covariance matrix. The DFT coefficients are represented only by their magnitude and not any information regarding the phase. In addition, the (1-step) Haar coefficients have

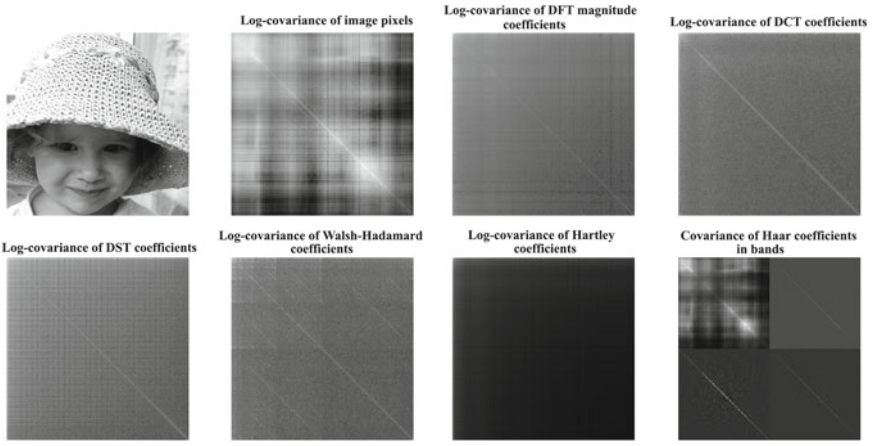

Fig. 2.51 Covariance matrices of various image transform coefficients

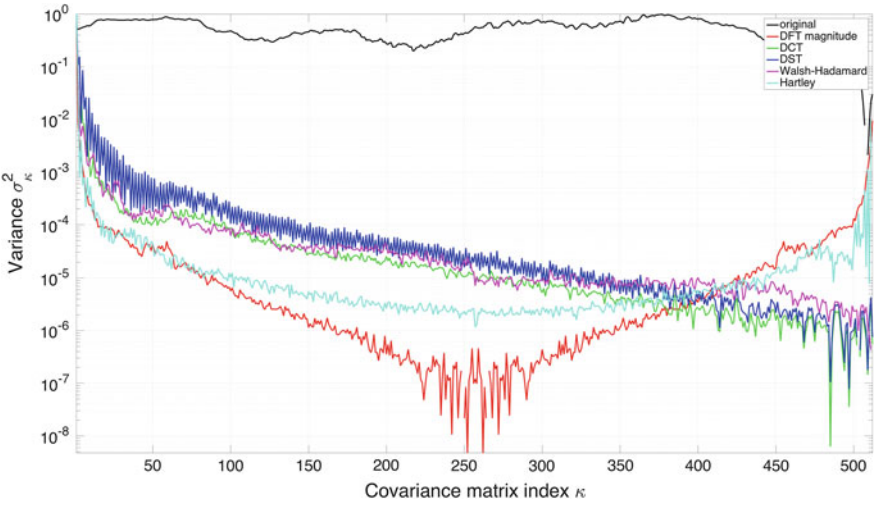

Fig. 2.52 Normalized log variance in each of the image transforms

been normalized individually in each of the four bands purely for illustration purposes; as expected the upper-left quarter of the matrix is equivalent to the covariance matrix of the initial image as it is simply a low-pass filtered replica of the original data, thus displays a large amount of correlation in bright regions. In addition, Fig. 2.52 shows a plot of the variances computed in each of the transforms, in normalized logarithmic scale; variances are shown as found in the covariance matrices without reordering or sorting (thus the 'mirroring' in the DFT magnitude is apparent).

2.3.2.2 Quantization

Quantization is usually defined as *the dividing of a quantity into a discrete number of small pieces, which are usually integer multiples of a common base quantity*. The oldest (and most common) example of quantization is that of *rounding*: every real number x may be rounded to the nearest integer $q(x)$ with a quantization error $e = |q(x) - x|$. Generally, it can be assumed that a quantizer consists of a set of intervals or 'cells' $S = \{S_i; i \in \mathfrak{I}\}$, where the set of indices $\mathfrak{I}$ is often a collection of consecutive integers (often starting with 0 or 1), along with a set of reconstruction values or points or levels $C = \{y_i; i \in \mathfrak{I}\}$, so that in overall the quantizer q is defined by the equation $q(x) = y_i$ for $x \in S_i$, which may eventually be represented by the formula (Gray and Neuhoff 1998)

$$q(x) = \sum_i y_i l_{s_i}(x) \tag{2.137}$$

where

$$l_{s_i} = \begin{cases} 1 & x \in S \\ 0 & \text{otherwise} \end{cases}$$

In order for this definition to have a substantial meaning, an assumption is made that S is a partition on the axis of the real numbers. This means that the intervals are independent and complete. The general definition also applies in the simplest case of rounding when $S_i = (i - 1/2, i + 1/2]$ and $y_i = i$ for all integers i. Even more generally, the intervals may take the form $S_i = (a_{i-1}, a_i]$ in which a_i (called *thresholds*) form an ascending sequence. The width of the interval S_i is the length of the $a_i - a_{i-1}$. The function $q(x)$ is usually called *quantization rule*. A simple quantizer with five levels is shown in Fig. 2.53. A quantizer is called *uniform* when the levels are equally spaced by a fixed distance δ among them, and the thresholds a_i are at the centers of the intervals. In all other cases the quantizer is called *non-uniform*.

The quality of a quantizer is evaluated by comparing the final outcome with the initial input. A common way of doing this is by defining a *distortion measure*—which quantifies the cost or the distortion that occurs in the reconstruction of the input to the quantizer output—and taking the average distortion as a measure of assessing the quality of the system, wherein lower average distortion means better quality. The most commonly used estimator is that of the squared error $d(x, \hat{x}) = |x - \hat{x}|^2$, ($\hat{x}$ representing the quantized value of x). In practice, the average value is a sample

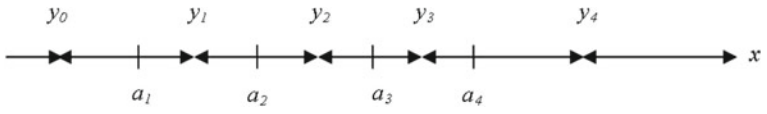

Fig. 2.53 A quantizer of five levels ($a_0 = -\infty, a_5 = \infty$)

average when the quantization is applied to sequences of real numbers, but in the general case data are taken as elements that share a common probability density function $f(x)$ corresponding to a random variable X, and the average distortion is converted to a statistical expectation (Gray and Neuhoff 1998)

$$D(q) = E[d(X, q(X))] = \sum_i \int_{S_i} d(x, y_i) f(x) dx \tag{2.138}$$

When the distortion is estimated by means of a squared error, $D(q)$ is converted to MSE, a special case, which is often used.

It is desirable that the average distortion is maintained as low as possible by increasing the number of intervals (negligible average distortion may be achieved when there are many small quantization intervals). However, additional cost is imposed in the form of extra bits of information that must be created to describe the quantizer, which creates a digital representation problem in light of a limited capacity. A simple method for estimating this cost is as follows: the quantizer encodes an input x to a binary representation or codeword corresponding to a quantization index i, thus determining the quantization level to be used for the reconstruction. If there are N possible levels and all the binary representations or codewords are of the same length, the binary vectors will need $(\lfloor \log_2 N \rfloor + 1)$ bits. This results to an estimate rate in bps

$$R(q) = \log_2 N \tag{2.139}$$

This quantizer that leads to fixed length binary representations is called *fixed-rate quantizer*.

In overall, the target of quantization is the encoding of input samples, characterized by a probability density function, into a form with the least number of bits (low rate) in such a way as to ensure the reconstruction of the input signal at the greatest possible precision (with low distortion). What immediately becomes clear is that there is a trade-off in quantization: distortion against data rate (and vice versa). This trade-off can be quantified by a rate-distortion function $\delta(R)$, which is defined as the minimum distortion by a quantizer with a rate less or equal to R

$$\delta(R) = \inf_{q:R(q) \le R} D(q) \tag{2.140}$$

It is also possible to define, alternatively, the function $r(D)$, as the lowest rate of a fixed-rate quantizer with distortion less or equal to D.

Up to this point, the description was about the so-called *fixed-rate scalar quantizer*, in which each sample of the input is encoded independently into a binary string of fixed length. There, of course, are many other alternative forms of quantization, which achieve better results in terms of rate-distortion. Some indicative cases are (Gray and Neuhoff 1998):

- *scalar quantizer with memory*, where a prediction technique is applied to predict the current sample by storing one or more previous samples (like the Differential

Pulse Code Modulation (DPCM) method, which is considered by many to belong to the quantization rather than the coding methods)

- *variable-rate quantizer*, which has a partition at intervals and a dictionary of quantization levels just as the fixed-rate quantizer, but uses a variable length binary representations of the intervals. In this category belong the *entropy constrained scalar quantizers*, which are designed to introduce the smallest average distortion under a pre-conditioned entropy
- *vector (or multidimensional) quantizer*, which is the generalization of the scalar quantizer in n-dimensions, providing a general model of quantization, which can be applied to vectors without structural constraints.

To generate optimal scalar quantizers the Lloyd-Max method (Lloyd 1982) is typically applied, in which:

- to create an optimum quantizer with L quantization intervals, the quantization levels are initialized as y_i, $i = 1, .., L$ and the decision end-points are

$$x_0 = -\infty, \quad x_{L+1} = \infty, \quad x_i = \frac{y_i + y_{i+1}}{2}, \quad i = 1, ..., L \tag{2.141}$$

- then the system performs iterations to achieve convergence or until the error is within desired limits, following the equations

$$y_i = \frac{\int_{x_{i-1}}^{x_i} xp(x)\mathrm{d}x}{\int_{x_{i-1}}^{x_i} p(x)\mathrm{d}x}, \quad x_i = \frac{y_i + y_{i+1}}{2}, \quad i = 1, ..., L \tag{2.142}$$

This expression of y_i indicates that it should be selected iteratively as the centroid of probability densities in the quantization interval $[x_{i-1}, x_i]$. It is worth noting that when the probability density function satisfies the condition $\mathrm{d}^2 p(x)/\mathrm{d}x^2 < 0$, which is satisfied for Gaussian and Laplacian distributions, the algorithm converges to a global minimum (in respect to distortion). The Lloyd-Max method generalizes for the case of optimum vector quantizers into the *generalized Lloyd-Max* method or the *Linde-Buzo-Gray (LBG)* method (Linde et al. 1980), in which the following occur:

- initialization of levels y_i, $i = 1, .., L$
- segmentation of the set of all learning vectors t_j into S_i sets, for which the vector y_i is the closest:

$$t_j \in S_i \iff d(y_i, t_j) \leq d(y_k, t_j), \forall k \tag{2.143}$$

- redefinition of y_i as the center of mass of S_i:

$$y_i = \frac{1}{\#S_i} \sum_{x_i \in S_i} x_j \tag{2.144}$$

- repetition of the last two steps until convergence

Although vector quantizers are generally better in performance, scalar quantizers are preferable in compression methods due to their significantly easy implementation and the much lower memory requirements. Scalar quantizers can be actually found at the heart of most wide-spread image compression standards, in many forms and alternatives, but with the same aim, in truncating the image data. It is the step within lossy image compression systems that generates the most (if not all) the degeneration of the original data, which is counter-balanced by the strong positive effect it produces for the entropy coding steps that usually follow. In the JPEG image compression standard, quantization is practically implemented in the form of an element-wise matrix operation on the 8×8 block it applies, using the following 8×8 reference quantization matrices for the luminance (q_Y) and chrominance (q_C) channels respectively (ISO-IEC-CCITT 1993b; Pennebaker and Mitchell 1993; Wallace 1991)

$$
\begin{aligned}
q_Y &= \begin{pmatrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\
14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\
14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\
18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\
24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\
49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\
72 & 92 & 95 & 98 & 112 & 100 & 103 & 99
\end{pmatrix} \\
q_C &= \begin{pmatrix}
17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\
18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\
24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\
47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\
99 & 99 & 99 & 99 & 99 & 99 & 99 & 99
\end{pmatrix}
\end{aligned}
\tag{2.145}
$$

The open source implementation of JPEG from the Independent JPEG Group (IJG) (Independent JPEG Group 2000) includes the following formula for the generation of the quantization matrices based on the reference matrices (2.145), depending on the coefficient of the desired quality defined by the user (f)

$$
Q_\xi = q_\xi \times e^k, \qquad k = 6 \ln 2 \frac{50 - f}{50} \tag{2.146}
$$

where Q_ξ the estimated quantization matrix, f the quality factor defined by the user and $ln \equiv \log_e$ the operator for the natural logarithm and $\xi = \{Y, C\}$. For f ranging in the percentage interval ([0, 100]), it holds that the intermediate variable $k \in [-4.159, 4.159]$, while its exponential representation $e^k \in [0.0156, 64.0000]$. As f increases, k and e^k decrease to allow for a consequent milder quantization. A graphical representation of e^k is shown in Fig. 2.54.

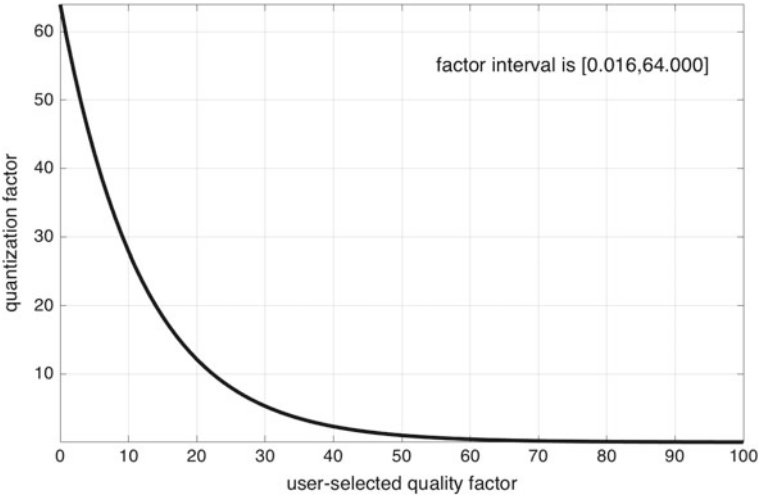

Fig. 2.54 Plot of the quantization factor e^k against the user-selected quality factor f

The quantization levels in the reference matrices (2.145) emerged from observing that after the DCT, which is applied to the original image data when compressing with JPEG, there is a particular spectral behavior, as expressed by the occurrence of large or small values to transform coefficients, which corresponds to specific frequencies, and therefore, other coefficients should be quantized with more and others with less tolerance.

The latest standard in image compression, JPEG2000, the quantizer is a typical *dead-zone uniform scalar quantizer* (Rabbani and Joshi 2002; Rabbani and Cruz 2001), as it is called due to that the central interval $(-1, 1]$ is twice as long as the other quantization intervals, and centered around zero (0). In this case, and because the transformation that is applied before quantization is the DWT, which leads to an analysis to multiple spectral bands, within the quantizer there is a special configuration on the spectral band in which it is acting (taking advantage of the features present in each zone). So for a given zone b a different quantization step Δ_b is defined. The choice of the quantization step in each zone is essentially defined by the user and may be based on modeling of the HVS, similarly to how the quantization matrices are defined in JPEG. The quantization rule in JPEG2000 is

$$q = sign(c) \left\lfloor \frac{|x|}{\Delta_\mathrm{b}} \right\rfloor \tag{2.147}$$

where x is the sample at the input of the quantizer, $sign(x)$ the sign of the input sample and Δ_b the quantization step in the respective spectral band.

In conclusion, quantization is responsible for the distortion introduced in a compression process. So when *lossless compression is required quantization should not*

be applied. In the process of quantization each sample or group of samples is mapped to a quantization level or a pointer to the dictionary of the quantizer and the input signal is trimmed substantially.

A simple example of image-domain-based quantization is shown in Figs. 2.55 and 2.56, which present typical examples of un-dithered uniform graylevel and color quantization into various number of levels each. In addition, Fig. 2.57 shows an example of the application of non-uniform, minimum variance color quantization. Each graylevel/color-quantized image is titled with the number of levels, the error estimation and the estimated quality in PSNR dBs.

A simple example of transform-domain-based quantization is shown in Figs. 2.56, 2.58, and 2.60, which present the effect of the quantization of the DCT transform coefficients using three different quantization matrices, based on the IJG JPEG quantization recommendation (2.145) and (2.146). The original image has been initially converter to the YC_bC_r color space and then transformed and quantized using an 8×8

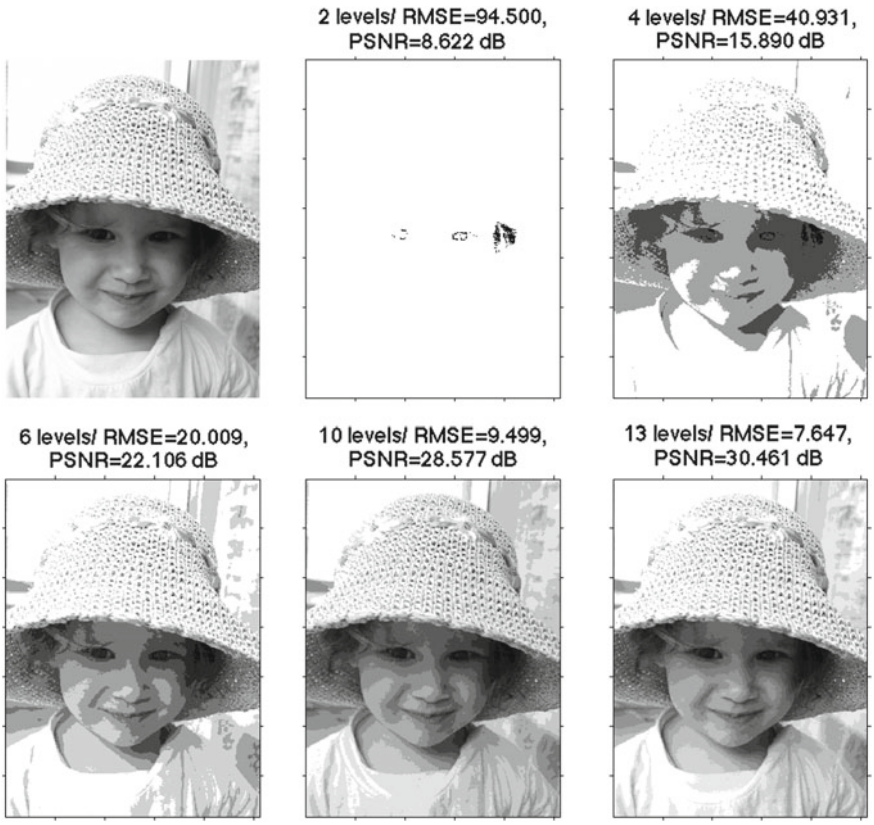

Fig. 2.55 Uniform quantization of a full-range graylevel image into various number of levels

Fig. 2.56 Uniform quantization of a full-range color image into various number of colors

pixels block partitioning, then de-quantized and inverse transformed, to provide a reconstruction and an estimate of the distortion imposed by the quantization.

Figure 2.58 presents the result of quantization using quality factor 1, which produces a reconstructed image with an average per-channel quality of about 22 dB, with a maximum pixel difference of 190 and an average RMSE 20, as detailed in Table 2.4. (2.145) and (2.146) produce the quantization matrices in (2.148).

$$q_Y = q_C = \begin{pmatrix} 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{pmatrix} \tag{2.148}$$

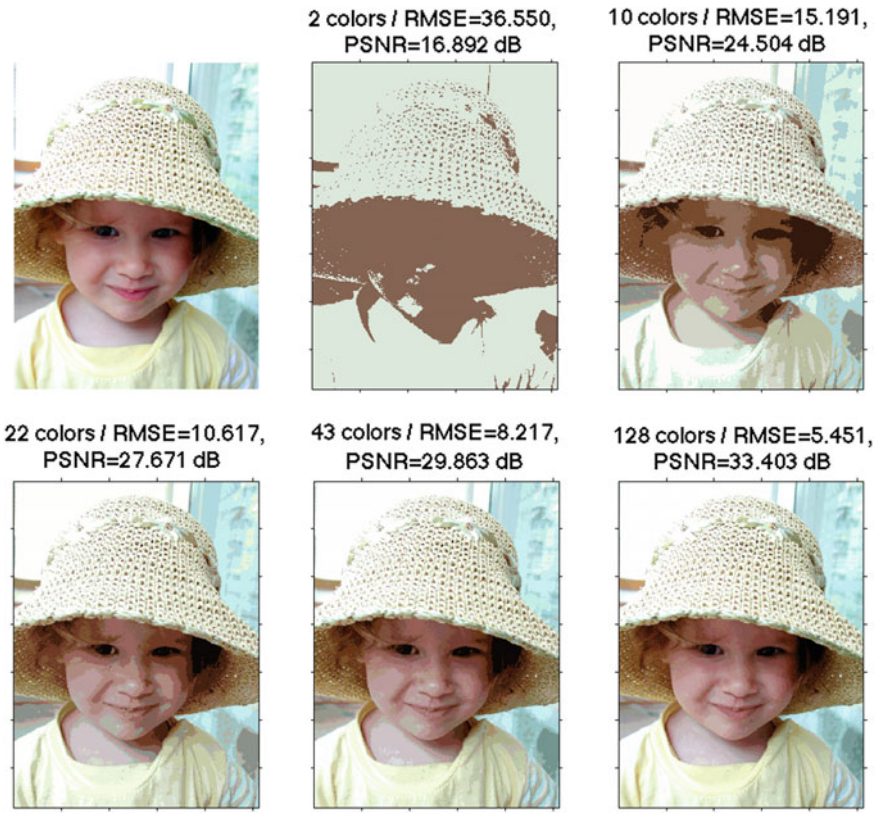

Fig. 2.57 Minimum variance quantization of a full-range color image into various number of colors

Fig. 2.58 Quantization in the DCT domain using quality factor 1

Table 2.4 Comparison between the original and the transform-domain-quantized image for quality factor 1

Channel	Max difference	RMSE	PSNR (dB)
Red	194	19.817	22.190
Green	185	16.983	23.531
Blue	198	23.359	20.762

Figure 2.59 shows the result of quantization using quality factor 25, which produces a reconstructed image with an average per-channel quality of about 26 dB, with a maximum pixel difference of 133 and an average RMSE 12, as detailed in Table 2.5. In this case the quantization matrices are differentiated and defined as shown in (2.149).

$$
\begin{aligned}
q_Y &= \begin{pmatrix}
128 & 88 & 80 & 128 & 192 & 255 & 255 & 255 \\
96 & 96 & 112 & 152 & 208 & 255 & 255 & 255 \\
112 & 104 & 128 & 192 & 255 & 255 & 255 & 255 \\
112 & 136 & 176 & 232 & 255 & 255 & 255 & 255 \\
144 & 176 & 255 & 255 & 255 & 255 & 255 & 255 \\
192 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\
255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\
255 & 255 & 255 & 255 & 255 & 255 & 255 & 255
\end{pmatrix} \\
q_C &= \begin{pmatrix}
136 & 144 & 192 & 255 & 255 & 255 & 255 & 255 \\
144 & 168 & 208 & 255 & 255 & 255 & 255 & 255 \\
192 & 208 & 255 & 255 & 255 & 255 & 255 & 255 \\
255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\
255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\
255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\
255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\
255 & 255 & 255 & 255 & 255 & 255 & 255 & 255
\end{pmatrix}
\end{aligned}
\tag{2.149}
$$

Finally, Fig. 2.60 shows the result of quantization using quality factor 50, which produces a reconstructed image with an average per-channel quality of about 35 dB (with a very small RMSE of about 4.5 on the average), as detailed in Table 2.6. According to the IJG definition (2.145) and (2.146), in this case the quantization matrices are exactly equal to the reference matrices in (2.145).

It is really obvious that in all cases of the transform-domain-quantization a best reconstruction is guaranteed for the Green channel, then the Red and last the Blue, which is actually in line with the human vision modeling (the HVS model) and the predominant role of the Green channel.

Fig. 2.59 Quantization in the DCT domain using quality factor 25

Table 2.5 Comparison between the original and the transform-domain-quantized image for quality factor 25

Channel	Max difference	RMSE	PSNR (dB)
Red	118	12.358	26.292
Green	110	10.227	27.936
Blue	133	13.543	25.497

2.3.2.3 Encoding

The purpose of encoding (many times referred to simply as 'coding') is to exploit statistical redundancy among quantized samples (samples that have undergone quantization) to minimize the length of the final codestream (or bitstream). The stages preceding coding, namely transformation and quantization, limit the correlations on a strictly local level. In the ideal case all the input samples to the encoder are statistically independent. In this case these samples can be encoded independently and the only statistical redundancy that can be considered is the one related to the variance in the distribution of their probabilities. In the general case, however, it is almost impossible to ensure statistical independence of quantized samples. Nevertheless, so long as these inter-dependencies are limited to a narrow (spatial or temporal) interval, it is usually possible to design efficient coding systems with easily managed complexity. Known encoders produce codes that can either be of fixed or of variable length; among the most famous are the *Huffman* and *Arithmetic encoders*. Various widely used coding methods are listed in the following paragraphs, some of the accompanied by example implementations and examples; Huffman and Arithmetic coding are presented in more detail.

Fig. 2.60 Quantization in the DCT domain using quality factor 50

Table 2.6 Comparison between the original and the transform-domain-quantized image for quality factor 50

Channel	Max difference	RMSE	PSNR (dB)
Red	48	4.764	34.572
Green	43	3.932	36.238
Blue	67	5.230	33.760

Pulse Code Modulation

Pulse Code Modulation (PCM) was introduced during the development of a digital audio broadcasting standard (Sayood 1996). Today, the term is used for every encoding method related to encoding of an analog signal. The method is not connected to any specific method of compression: it simply implies the quantization and digitization of an analog signal. The signal range is divided into intervals, and each interval is assigned a unique index. For the encoding of an input sample, the interval to which it belongs is being tracked and the appropriate interval index is stored. The sampling of the signal occurs at regular intervals. Thus, the sequence of the signal in time or space can be stored as a sequence of digits (bits) of a pre-determent rate. An example of the application of PCM is shown in Figs. 2.61, 2.62, 2.63, and 2.64. The original signal is a 8 Kbps (8192 bps) one-dimensional signal.

Apparently in PCM the first process is sampling with the sampling rate being based on the Nyquist rate ($f_s \geq 2f_{max}$). The output of the sampling process is a discrete (in time) signal. This discrete signal is fed to a quantizer which, ultimately, makes the signal digital. At the final stage, an encoder converts the digital signal to a sequence of binary digits or bits. The quantization step size can be simply defined as $\Delta = \frac{\Re(x)}{q}$, with $\Re(x)$ the range of the input signal x and q the quantization levels; this

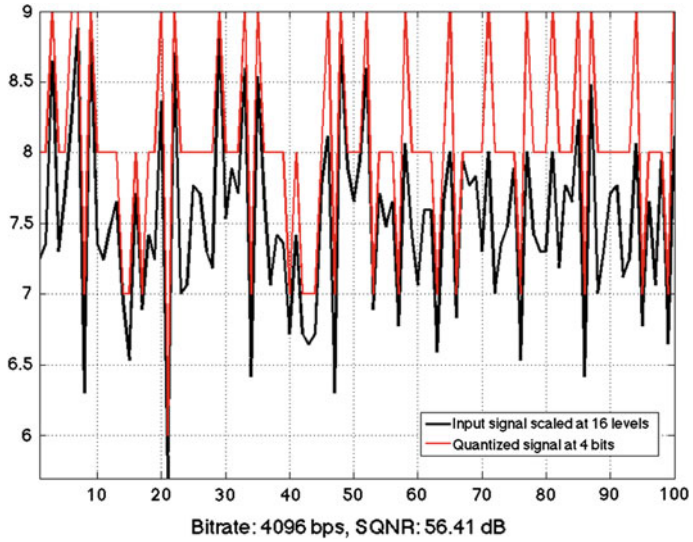

Fig. 2.61 PCM on an 8 Kbps signal using 4 bits at 4 Kbps rate and SQNR 56.41 dB

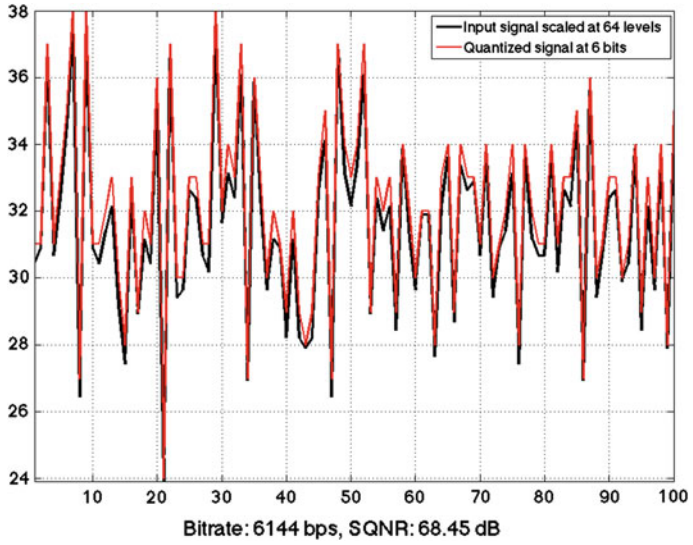

Fig. 2.62 PCM on an 8 Kbps signal using 6 bits at 6 Kbps rate and SQNR 68.45 dB

simplifies to $\Delta = \frac{2}{q}$ if $-1 \leq x \leq 1$. The error introduced by PCM is the quantization error (or quantization noise) $\epsilon = q(x) - x$ and it can be shown that the quality of the encoding can be estimated by the *signal to quantization noise ratio* as,

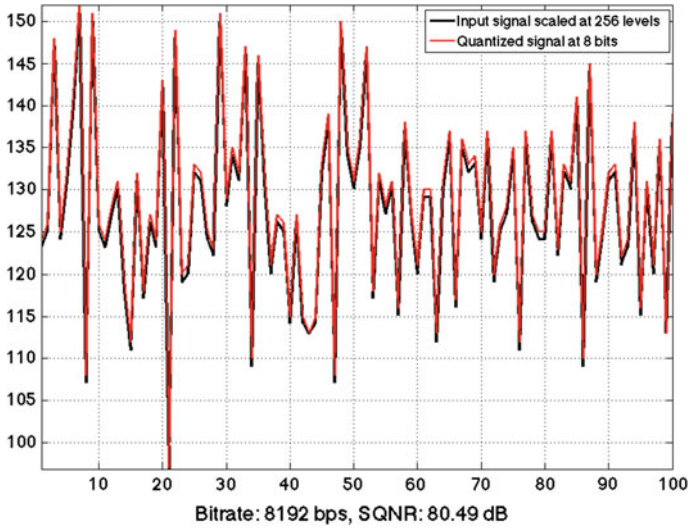

Fig. 2.63 PCM on an 8 Kbps signal using 8 bits at 8 Kbps rate and SQNR 80.49 dB

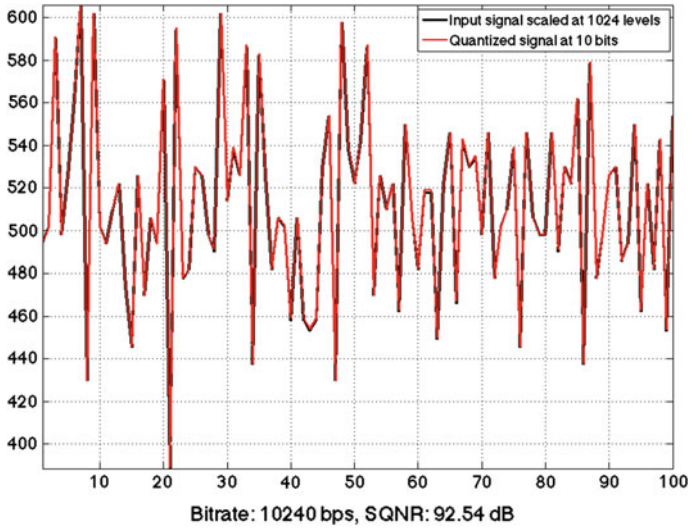

Fig. 2.64 PCM on an 8 Kbps signal using 10 bits at 10 Kbps rate and SQNR 92.54 dB

$$SQNR = 10 \log \left(\frac{\sum q_x^2}{\frac{\Delta^2}{12}} \right) \text{ dB} \tag{2.150}$$

where $\sum q_x^2$ is the quantized signal power and $\frac{\Delta^2}{12}$ is the quantization noise power.

Differential Pulse Code Modulation

Differential Pulse Code Modulation (DPCM) is an encoding scheme based on PCM which adds a prediction functionality. Prediction in DPCM is considered in two approaches; in the first approach, prediction of the current value is made by knowing the previous value; in the second approach the difference relative to the output of a local model of the decoder process is considered (a decoder is incorporated in the encoder). In the simple (first) case the indices do not correspond to sample values (as in PCM), but to differences between successive samples (Sayood 1996). If, for example, a horizontal row of pixels of an image is encoded, an index may represent a difference in luminance between the current pixel to the previous one. It is known that there are many types of signals in which mostly small changes between successive samples appear.

When this method is applied to such signals, the indices that represent small differences are very frequent. By using entropy coding these indices can be greatly reduced in value and thus lead to better compression ratios, since, as will later be explained, a new distribution of samples emerges that approximates the *Laplacian distribution* or the so-called *two-sided geometric distribution*. This method is a simple example of predictive encoding, since, essentially, a predicting of the next value is being made according to the current. If the prediction is correct, the result is an index corresponding to a very small value, while if the prediction is wrong, the size of the index may be bigger than what the simple PCM would produce. An example of the application of DPCM on a full color (24-bpp) image is shown in Fig. 2.65. The DPCM image levels have been normalized for better illustration. The histograms of the luminance of the original image and the DPCM image are shown in Fig. 2.66.

Run-Length Encoding

Certain types of data may exhibit repetitive sequences of samples. This is strongly evident in binary images of digital documents, in which very large white regions (page background) may be found. A simple idea to efficiently compress such signals is to replace the sequence of repeated samples by a quantity that represents their number (Sayood 1996; Golomb 1966). According to the method of *Run-Length Encoding (RLE)*, an original sample represented by a particular symbol is considered the start of a sequence, which lasts as long as consecutive samples have the same value. In the end, this sequence is replaced by the sample value and the sequence length.

A typical example would be as follows: given a sequence of two symbols ‘W’ and ‘B’

```
WWWWWWWWWBBBBBWWWWWWWWWWWW
BBBWWWWWWWWWBBBBBBBBBB
WWWWWWWBBBBBBBBB
WWWWWWWWWWWWWWWWWWWW
BBBWWWWWWWWWWB
```

Fig. 2.65 DPCM representation of a color image; the range of values is [−255, 255]

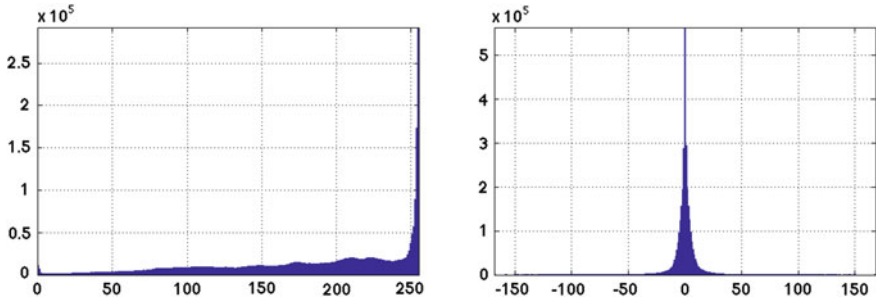

Fig. 2.66 Histograms of luminance channel in the original and the DPCM images

then according to RLE this sequence would be encoded as

```
W9B5W12B3W9B10W7B10W20B3W10B2
```

which represents the same message of the original 100 symbols using only 29 symbols (which is a 71 % compression). Clearly RLE results in very compact representations, especially in cases of large sequences of repeating symbols. It is effectively being applied in the standard JPEG encoding scheme for the efficient formation of the final bitstream of the compressed data, as will be shown in the corresponding section. In a way, RLE could be viewed as a simplified special case of a dictionary-based encoding scheme (described in the following paragraphs), in which only single symbols represent the dictionary and no patterns of consecutive different symbols (like words) are being exploited along with lookup tables for reference.

Table 2.7 Average results of RLE test on 100 natural and 100 text/graphics images

Image type	Gain (%)	Bitrate (bpp)
Natural images	17.3	6.61
Text/graphics images	97.7	0.18

A test on 100 graylevel natural images and on 100 graylevel and binary textual, line-art and synthetic graphics printed or handwritten images reveals the apparent advantage of using RLE in non-natural images as summarized in Table 2.7. In the reported results, 'gain' refers to the conservation of storage that can be achieved (compression), which attains the enormous amount of 97.7 %, (a data rate less than 0.2 bpp) for the case of non-natural images. The method is only of limited interest when applied to natural images, as only a 16 % gain has been achieved (6.61 bpp average data rate). Original images were treated as 8 bpp (luminance) images.

Figure 2.67 shows a typical example of two binary images where only the black pixels are considered to convey information (as in traditional fax applications). In these images, a simple RLE coding would result a significant amount of data compression up to a 87 %. The particular example used an alternative implementation

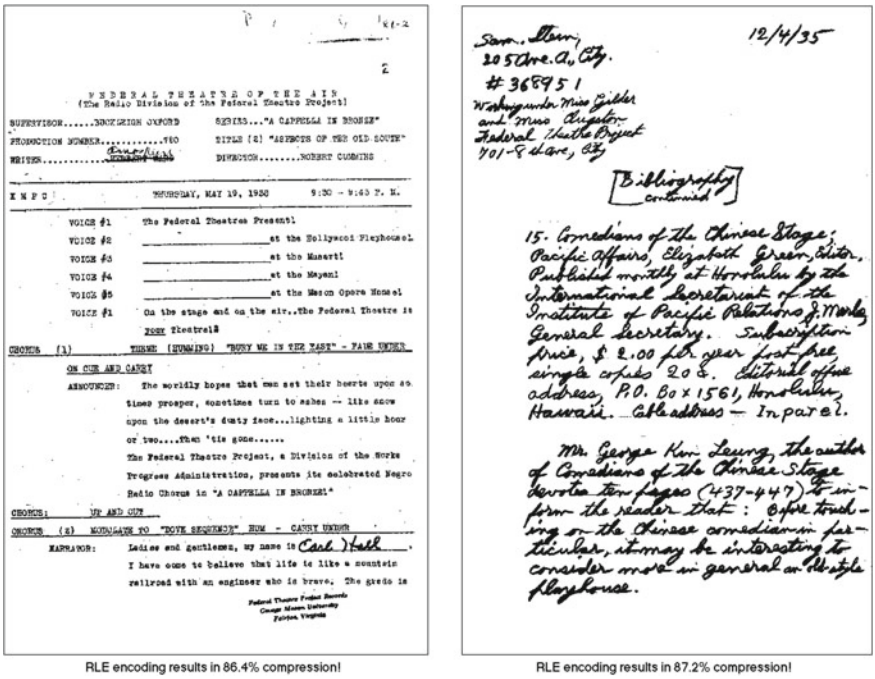

Fig. 2.67 Two typical binary images encoded using RLE

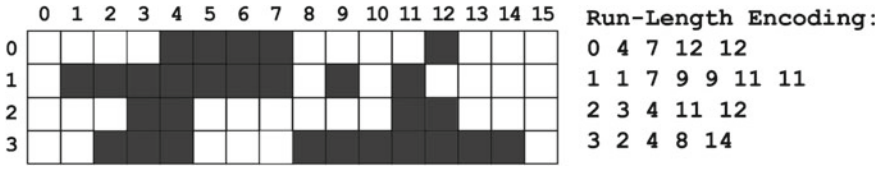

Fig. 2.68 RLE encoding rule and example

of RLE which is graphically depicted in Fig. 2.68, in which the encoding starts by the current row number (zero being the first row and column) complemented by the column positions than mark the beginning and ending of consecutive black pixels (the data).

Figure 2.69 presents a closeup in just 100 rows of the first image, along with a vertically stretched replica of the image rows for a better illustration of the corresponding row-wise and column-wise pixel counts (the histogram-like graphs at the bottom and right of the stretched image). Taking the first row in the image the RLE output of this single row is

```
500 486 490 495 497 502 506 547 549 557 559 563 567 609
612 623 629 664 666 673 676 680 685 722 725 738 745 790
792 797 799 847 851 930 934 941 944 949 955 991 995 997
1010 1052 1064 1119 1121 1170 1173 1181 1183 1231 1247
1249 1251 1292 1306 1376 1381 1397 1404 1443 1455 1459
1463 1537 1541 1548 1550 1589 1594 1604 1608 1649 1652
1658 1662 1745 1757 1808 1812 1869 1875
```

with the first entry denoting the image row index and each pair of entries denoting the location (column) of the beginning and ending of a series of black pixels. This single

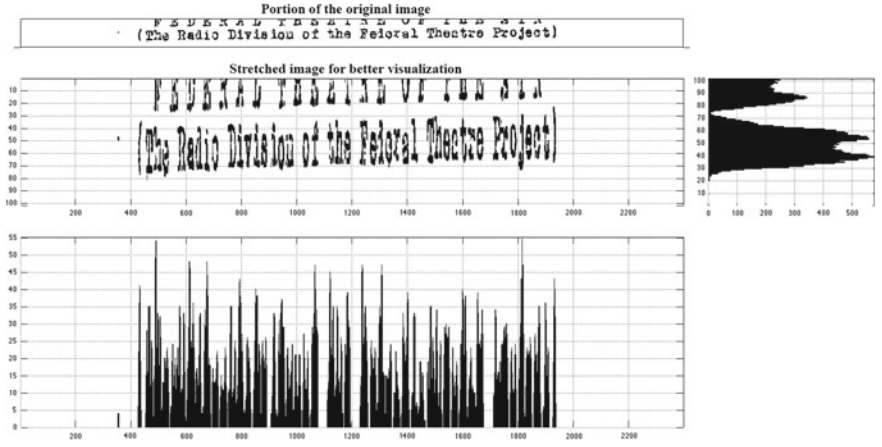

Fig. 2.69 Part of a binary image and the horizontal and vertical pixel counts

row is represented by 83 integers in the range [486, 1875], so apparently 16-bit integer representations are required. This means that this row requires $83 \times 2 \times 8 = 1{,}328$ bits to be perfectly reconstructed. The original image row is of 2,396 binary pixels (2,396 bits). Thus, RLE of this row results to a 44.6 % compression, or a 1.8:1 compression ratio, which is relatively low, although expected, since this particular image row contains a significant amount of information (black pixels). In the overall image portion, 7,497 (16-bit) RLE values are needed to represent the 239,600 binary pixels, which amounts to 49.9 % compression or an approximate 2:1 compression ratio. Apparently, other RLE rules and approaches are expected yield significantly different compression results.

Shannon-Fano Coding

For a given resource, the optimum compression rate that can be achieved is the one determined by the entropy of the source. Entropy, as already mentioned, is the measure of uncertainty of a source of information. Based on this principle, the basic idea behind the *Shannon-Fano Coding (SFC)* (Sayood 1996) is as follows: through the use of variable length codes, symbols are encoded according to their probabilities, using less bits to the symbols with the highest probability of occurrence, in compliance with the original Shannon theory. The method is considered to be suboptimal in the sense that it does not achieve the lowest possible code length but it guarantees that all code lengths are within one bit of their theoretical ideal set by the Shannon entropy. Although based on Shannon's theory, the method is co-attributed to Robert Fano, who published it as a technical report at MIT (Fano 1949).

SFC is a recursive process in which all symbols are assigned a codeword. Initially the symbols are arranged according to their probability in descending order, and then separated into two sets with total probabilities close to being equal. Then the first digit of the codeword of all symbols is being created, using "0" for the first set and "1" for the second. This process is repeated until no more sets remain with more than one members and all codewords have been assigned to all symbols. The process results in symbol with prefix codes (no other symbol has the same prefix).

The method is efficient in producing variable-length codes when the partitioning is as close as possible to being of equal probability. Although computationally simple, the method is almost never being used because it requires the a partitioning to equal probability sets is possible, in order to be efficient. Nevertheless, SFC is being used in the *IMPLODE compression method* in the ZIP compression.

The implementation of SFC is a simple tree-defining algorithm as follows:

1. Definition of the underlying probability mass function, by counting the occurrence of symbols
2. Ordering of the sequence of symbols in the order of descending probability
3. Division of the sequence into two parts having approximately equal total probability
4. Assignment of the binary digit "0" to the first set and of the digit "1" to the second set, so that all set symbols start with the same digit

5. Repetition of steps 3 and 4 for each of the two sets, to subdivide the sets and assign the appropriate additional digits to the corresponding codes until all symbols have become leaves on the tree that is being constructed

In a simple example of 10 symbols $S_I = \{A, B, C, D, E, F, G, H, I, J\}$ with corresponding occurrences $f_I = \{3, 8, 10, 5, 1, 2, 1, 7, 4, 9\}$, $I = 1, ..., N$, $N = 10$, the probabilities are shown in Table 2.8. The symbols are ordered in descending order of their probability and the sequence becomes $\{C, J, B, H, D, I, A, F, E, G\}$. Then the recursive splitting of the sequence into two sets takes place until all sets have only one symbol. Figure 2.70 shows graphically the overall process that results in the assignment of a single prefix code for each of the input symbols as shown in the table embedded in the figure. In this figure the red color denotes the "0" code paths,

Table 2.8 SFC example using 10 symbols with various probabilities

A	B	C	D	E	F	G	H	I	J
3	8	10	5	1	2	1	7	4	9
0.06	0.16	0.2	0.1	0.02	0.04	0.02	0.14	0.08	0.18

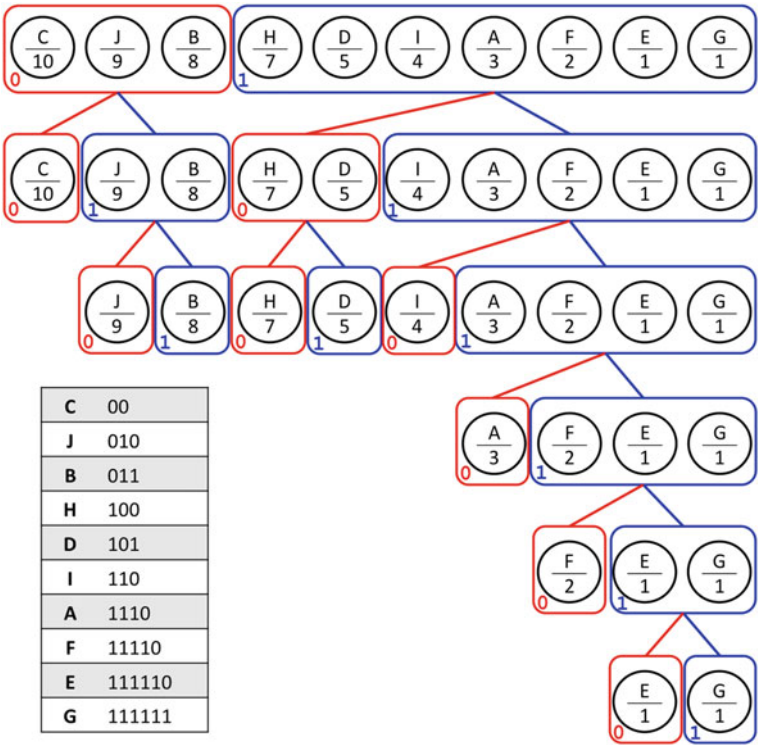

Fig. 2.70 Example of the formation of a Shannon-Fano tree of codes

whereas the blue color denotes the "1" code paths. Using (2.29) the entropy of this source is $H_i = 3.009$ bps. The data rate is estimated

$$\frac{\sum_{i \in S_I} bits(i) \times f_i}{\sum_{i \in S_I} f_i} = \\ = \frac{1}{50} [2\ 3\ 3\ 3\ 3\ 3\ 4\ 5\ 6\ 6] [10\ 9\ 8\ 7\ 5\ 4\ 3\ 2\ 1\ 1]^T = 3.06 \text{ bps} \quad (2.151)$$

Dictionary Methods

Dictionary Methods exploit the fact that many types of data exhibit repeated identical symbol sequences (patterns). Thus, it becomes possible to construct a 'dictionary' of 'model' sequences or structures from the original data and to replace the data with corresponding indices. Dictionary methods apply efficiently in text compression and generally ASCII files with a specific structure. They can also operate efficiently on graphics images or digitized text, but not in natural images (such as photographs). The most widespread form is expressed with the known Lempel-Ziv (LZ) algorithm (Ziv and Lempel 1978), and can be found in standard image storage formats such as TIFF for lossless compression (which yields a compression ration of around 2:1).

Huffman Coding

Huffman codes (Huffman 1952) are a subset of a large family of uniquely decodable variable-length codes. A Huffman code is generated by calculating the probabilities of the source symbols and the corresponding efficiency-indices so that the resultant code is minimized in length. It has been proven that Huffman codes have the minimum average length as compared to all other codes. Given a source with finite alphabet of symbols

$$S_I = s_0, s_1, ..., s_{k-1} \quad (2.152)$$

following a corresponding probability distribution function f_I, it is reasonable to search for an optimal coding method, where the average length of the codeword be minimal through all uniquely decodable codes. This optimal encoder is not expected to be unique. David Huffman developed an algorithm for finding a set of lengths that satisfy the relation

$$\sum_{i \in S_I} 2^{-l+i} \leq 1 \quad (2.153)$$

which minimizes the mean codeword rate. This equation expresses the characteristics which the lengths (l_i) must meet so that the codeword be uniquely decodable, as a necessary condition (Kraft–McMillan theorem (McMillan 1956)). Assuming the alphabet is sorted so that $f_I(s_0) \leq f_I(s_1) \leq ... \leq f_I(s_{k-1})$, Huffman coding is based on the observation that

Among all the optimal codes, at least one has $l_{a_0} = l_{a_1} = ... = l_{max}$, the maximum codeword length, such that the codes c_{s_0} and c_{s_1} differ only in their last bit.

This observation indicates that the optimization problem can be reduced to finding only $K-1$ codewords. This problem can now be expressed as follows: Determine the lengths that satisfy

$$\sum_{k=1}^{K-1} 2^{-l_{s_k}} \leq 1 \tag{2.154}$$

which minimize the quantity

$$\begin{aligned} R &= (f_I(s_0) + f_I(s_1)) \cdot (l_{s'_1} + 1) + \sum_{k=2}^{K-1} f_I(s_k) l_{s_k} = \\ &= f_I(s_0) + f_I(s_1) + \sum_{i' \in S_I} f'_I(i') l_i \end{aligned} \tag{2.155}$$

where I' a new random variable with symbols alphabet

$$S_{I'} = s'_1, s'_2, ..., s'_{K-1} \tag{2.156}$$

and

$$f_{I'}(i) = \begin{cases} f_I(a_0) + f_I(a_1) & i = a'_1 \\ f_I(i) & \text{otherwise} \end{cases} \tag{2.157}$$

Thus, the problem is identical to the original, but with a reduced alphabet. This reasoning leads naturally to the following algorithm, which iteratively reduces the problem of generating optimal codewords to the simplest problem, where there is only a binary alphabet (Cover and Thomas 2006; Taubman and Marcellin 2002b).

In order to create Huffman codewords (Taubman and Marcellin 2002b) classification of elements of the alphabet should take place, so that

$$f_I(s_0) \leq f_I(s_1) \leq \cdots \leq f_I(s_{K-1}) \tag{2.158}$$

If $K = 2$, $c_{s_0} =$ "0" and $c_{s_1} =$ "1" are instantly defined, otherwise,

- a new alphabet is being created $S_{I'} = s'_1, s'_2, ..., s'_{K-1}$, and a probability distribution function such that (2.157) holds.
- the creation algorithm is being applied recursively to find the optimal codewords $c_{s'_1}, c_{s'_2}, ..., c_{s'_{K-1}}$ for S'_I, f'_I
- the codeword $c_{s'_1}$ is being expanded by complementing either with "0" or "1" for the production of c_{s_0}, c_{s_1} respectively.

Re-using the simple 10-symbols example defined and presented for the case of Shannon-Fano Coding, and reviewing that the symbols are $S_i = \{A, B, C, D, E, F, G, H, I, J\}$ with corresponding occurrence frequencies $f_i = \{3, 8, 10, 5, 1, 2, 1, 7, 4, 9\}$, $i = 1, ..., N$, $N = 10$, as shown in Table 2.8, and that the symbols are ordered in a descending frequency of occurrence ordering $\{C, J, B, H, D, I, A, F, E, G\}$, then the

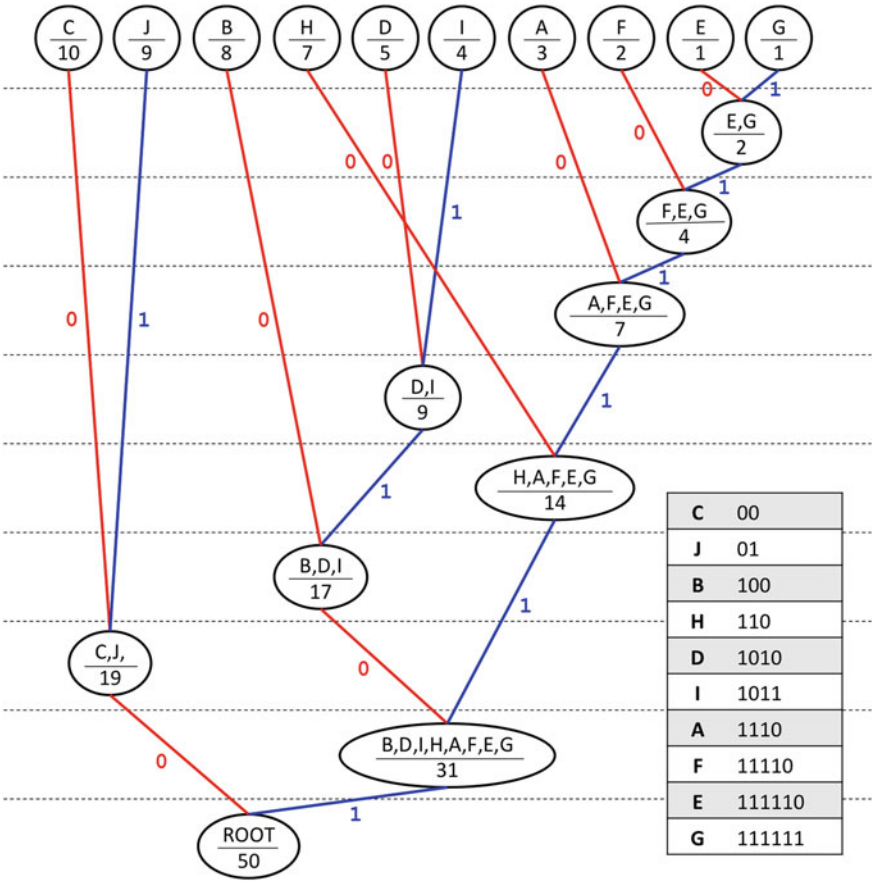

Fig. 2.71 Example of creating Huffman codes

method proposed by Huffman for generating the codes of those symbols is depicted in Fig. 2.71. In this figure the red color denotes the "0" code paths, whereas the blue color denotes the "1" code paths. The final resulting codes are shown as a table embedded in the same figure. Typically, using (2.29) the entropy of this source is $H_i = 3.009$ bps. The data rate of the Huffman-encoded symbols is estimated as

$$\frac{\sum_{i \in S_I} bits(i) \times f_i}{\sum_{i \in S_I} f_i} = \\ = \frac{1}{50} [2\ 2\ 3\ 3\ 4\ 4\ 4\ 5\ 6\ 6] [10\ 9\ 8\ 7\ 5\ 4\ 3\ 2\ 1\ 1]^T = 3.06 \text{ bps} \qquad (2.159)$$

Apparently, in this example the resulting compact representation is equivalent to the one provided by Shannon-Fano Coding, although the codes are different, and is very close to the entropy limit.

Golomb Coding

In 1966, Solomon Golomb presented a mental experiment in which a special roulette game is being played in a way that it produces binary outcomes (reporting whether a "0" is encountered or not—a "0" would have a probability of 1/37 in this case). He realized that in cases of sequences of infinite outcomes, or more generally of largely un-balanced event probabilities, a Huffman coding would not be applicable (Golomb 1966). His insight was that a run length of n favorable events[10] of probability p between successive unfavorable events with probability $q = 1 - p$ would have a probability of $p^n q,\ \ n = 0, 1, 2, ...$, which is approximated by no other than the *geometric distribution*.[11] Golomb realized that a shift towards the study of distributions for the discrete case should initiate, at a time when Gaussian distributions dominated the representations of every process being studied. He tried to shape the form of Huffman coding when applied to the geometric distribution, which is a more accurate representation of processes in the discrete case, and especially in the case in which $p \gg q$. Apparently, Golomb was concerned about encoding run-lengths, so this method is primarily applicable to positive integer values, and may exhibit slightly different implementations whether the encoded inputs are natural numbers (including zero) or strictly integer numbers.

Golomb introduced a parameter m, which controls the coding process by dividing any input x into two parts, namely the *quotient* q and the *remainder* r of the division of the input by the parameter m as

$$\begin{aligned} q &= \left\lfloor \frac{x}{m} \right\rfloor \\ r &= x - qm \end{aligned} \tag{2.160}$$

The quotient q undergoes a typical *unary encoding*, whereas the remainder r undergoes a *truncated binary encoding* and the two binary representations are concatenated to form the final Golomb code.

In *unary encoding* a natural number n is assigned a binary codeword of n 1's followed by a 0, thus $U(3) = 1110$, or $U(5) = 111110$. In the case of a strictly positive integer n the assignment corresponds to $(n-1)$ 1's, thus $U_{>0}(3) = 110$. The original Golomb coding accepts the first approach and considers natural numbers (including zero).

In *truncated binary encoding*, things get complicated, as a number is represented by its binary code that is truncated to the number of bits actually needed for a correct representation and one-to-one reconstruction, following a *prefix code* creation process. For example by encoding $n = 6$ coming from an alphabet of $k = 5$ bits,

[10] It should be noted that the definition of what is 'favorable' purely depends on the context.

[11] For simplicity, *geometric* is a distribution sharply concentrated around the mean value with only small values around it being probable (small variance).

truncated encoding outputs $T(6;5) = 110$, whereas a typical *binary encoding* would output $B(6;5) = 00110$. Practically, truncated binary encoding requires the size of the alphabet from which the input number is selected. Then it computes the bits needed to encode the length of the alphabet and an offset of numbers to advance while converting to the binary representation. Given an input number x and an alphabet of length m,

$$\begin{aligned} k &= \lfloor \log_2 m \rfloor \\ o &= 2^{k+1} - m \\ \text{CODE} &= \begin{cases} \text{bin}(x, k), & x < u \\ \text{bin}(x + o, k + 1), & x \geq u \end{cases} \end{aligned} \tag{2.161}$$

where $\text{bin}(\xi, b)$ is the binary representation or binary encoding of ξ using b bits. Apparently, the first o-bits of the code are assigned k bits whereas the last $(m - o)$-bits are assigned $(k + 1)$ bits, by skipping o numbers in the middle of the alphabet range. Any number less than the offset will be simply binary represented using k bits, whereas any number larger than the offset will be increased by this offset and then binary encoded using $(k + 1)$ bits. It should be noted that the offset purely depends upon the alphabet size.

Two MATLAB function included in this section implement the Golomb coding process; the first implements the truncated binary encoding algorithm and the second the complete Golomb coding procedure. These functions can either encode a single natural number or a vector of natural numbers. Obviously, the truncated encoding function is generic, and it can be used in other settings also. The results of the Golomb coding function have been tested against all examples provided by Golomb himself in the original 1966 paper (Golomb 1966).

```
function code = truncated_binary_code( x, varargin )
   if nargin==1,
      if length( x ) > 1
         m = length(x);
      else
         error('You need to specify the alphabet size!');
      end
   else
      m = varargin{1};
   end
   if find(x > m-1)
      error('Cannot compute code for X>M-1!');
   end
   % initialize
   code = cell(length(x),1);
   % compute the bits needed to encode the alphabet
   k = floor(log2(m));
   % compute the offset of numbers to skip while advancing
   o = 2^(k+1)-m;
   % since u is known then the assignmens is as follows:
   % the first o-bits are assigned typical binary codes of number X
   % whereas the last (m-o)-bits are assigned (k+1)-length binary codes of number
        (X+1)
   for i=1:length(x)
      if x(i) < o,
```

```
        code{i} = dec2bin(x(i),k);
      else
        code{i} = dec2bin(x(i)+o,k+1);
      end
   end
```

```
function code = golomb_code( num, param )
   % get the two parts of the number to encode
   q = dix(num/param);
   r = rem(num,param);
   % produce unary of q
   ql = length(q);
   code1 = cell(ql,1);
   for j=1:ql
      code1{j} = '';
      for i=1:q(j)
         code1{j}(i) = '1';
      end
      code1{j}(end+1) = '0';
   end
   % produce the truncated binary code of r
   code2 = truncated_binary_code( r, param);
   % concatenate code1 and code2 to produce final code
   code = cell(ql,1);
   for j=1:ql
      code{j} = [ code1{j}, code2{j}];
   end
```

The Golomb coding method was independently rediscovered by Tanaka and Garcia in 1982 (Tanaka and Leon-Garcia 1982) and One et al. in 1989 as MELCODE (Ono et al. 1989).

Elias and Arithmetic Coding

A special case of codes, the *universal codes*, are those which map source messages to codewords with an average length that is bounded by a linear combination of the entropy, $L \leq \alpha(H + \beta)$; that is universal codes achieve an average length that is less than a multiple of the optimal code. Apparently, in universal codes, the amount of compression depends on the values of α and β, so these codes are usually defined as *asymptotically optimal codes*, with the obvious asymptotically optimal code corresponding to $\alpha = 1$, $\beta = 0$. In these codes it is sufficient to know the probability distribution to the extent that the source messages can be ranked in probability order, so that decreasing probability symbols get increasingly larger codeword lengths, thus achieving *universality*.

A universal code may be considered as providing an enumeration of a sequence of symbols, giving, actually, a representation of the integers that enumerate the sequence. Peter Elias, in his seminal paper in 1975 (Elias 1975), gave rise to a whole family of universal coding methods that map positive integers onto binary codewords.

The simplest of the Elias codes is the *Elias gamma code*, which simply maps a natural number $x \in N$ by a binary code that is constructed by concatenating $\lfloor \log_2 x \rfloor$ "0" with the binary representation of the number; $\lfloor \log_2 x + 1 \rfloor$ is the total number

of bits required to represent the natural number. For example, the Elias gamma code for number $x = 15$ is produced by concatenating $\lfloor \log_2 15 = 3 \rfloor$ zeros with its binary representation 1111, thus $C_\gamma = 0001111$. Decoding starts by counting the number of zeros n at the beginning of each decoded sequence of bits. If there is no zero ($n = 0$) at the beginning then the number to be decoded is obviously "1"; in any other case ($n > 0$), the decoder reads the following $n + 1$ bits and decodes the corresponding binary number. A MATLAB implementation of the Elias gamma coding method is presented in the following listing.[12]

```
function y = elias_gamma_code( x )
   bts = floor( log2( x ) );
   dts = dec2bin( x );
   nbts = bts+1;
   nums = length( x );
   for i = 1:nums
      sbts = '';
      if bts(1,i)>0
         sbts = num2str( zeros(1,bts(i)));
         sbts = strrep( sbts, ' ', '');
      end
      sz = length(dts(i,:));
      bg = sz-nbts(i)+1; if bg<1, bg=1; end;
      snum = [ sbts dts(i,bg:sz)];
      y(i,1:length(snum)) = snum;
   end
```

A more complex and sophisticated Elias code is the *Elias delta code*, which builds upon the gamma code. Encoding of a natural number using delta coding requires to concatenate $C_\gamma\left(\lfloor \log_2 x \rfloor + 1\right)$ with a modified binary representation of the number. The modification of the binary representation consists in discarding the first bit of the representation which is expected to be always a "1" for any number. For example, the Elias delta code for $x = 15$, the binary representation is $1111 \rightarrow 111$, whereas $C_\gamma\left(\lfloor \log_2 x \rfloor + 1\right) = C_\gamma(4+1) = 00100$, thus $C_\delta(15) = 00100111$. To decode delta codes, the procedure begins with the decoding of the embedded gamma code, which is at the beginning of the delta code (using the procedure described for the gamma codes); this provides with knowledge regarding the number of bits b in the binary representation that follows. Then the decoder simply decodes the following $b - 1$ bits after complementing with a preceding "1". A MATLAB implementation of the Elias delta coding method is presented in the following listing.[13]

```
function y = elias_delta_code( x )
   bts = elias_gamma_code( floor( log2( x ) )+1 );
   dts = dec2bin( x );
   nums = length( x );
   for i = 1:nums
      sdts = deblank( dts(i, strfind(dts(i,:),'1'):end));
      sdts = sdts(2:end);
```

[12] This MATLAB function can be executed either for a scalar (single) or for a vector input and will respond with the appropriate results.

[13] This MATLAB function can be executed either for a scalar (single) or for a vector input and will respond with the appropriate results.

```
    b = deblank( bts(i,:) );
    d = sdts;
    snum = [ b d];
    y(i,1:length(snum)) = snum;
end
```

Table 2.9 presents some representative examples of gamma and delta codes for selected natural numbers. Clearly the number of bits required by each of these codes (both gamma and delta) is

$$\begin{aligned} L_\gamma(x) &= 2\lfloor \log_2 x \rfloor + 1 \\ L_\delta(x) &= L_\gamma(\lfloor \log_2 x \rfloor + 1) + \lfloor \log_2 x \rfloor \\ &= 2\lfloor \log_2(\lfloor \log_2 x \rfloor + 1) \rfloor + 1 + \lfloor \log_2 x \rfloor \end{aligned} \tag{2.162}$$

In the case of a random variable X with uniform distribution on $\{1, 2, ..., N\}$ and entropy $H(X) = \log_2 N$ bps, the expected lengths for the gamma and delta codes are respectively

$$\begin{aligned} E\left[L_\gamma(X)\right] &= \frac{1}{N}\sum_{x=1}^{N}\left(2\lfloor \log_2 x \rfloor + 1\right) \\ E\left[L_\delta(X)\right] &= \frac{1}{N}\sum_{x=1}^{N}\left(2\lfloor \log_2(\lfloor \log_2 x \rfloor + 1) \rfloor + 1 + \lfloor \log_2 x \rfloor\right) \end{aligned} \tag{2.163}$$

An even more advanced method that is the third in the family of Elias coding methods, is the one that generates the *Elias omega codes*, or the *recursive Elias codes*, due to their recursive generation. These codes can also be applied to natural numbers and are more efficient in compact representations of small numbers. There are two alternative methods to generate the omega codes.

According to the first method, in order to encode a natural number x the code starts with a "0" and iteratively builds up as now bits are concatenated at the beginning of the code, as follows

1. Start by placing a "0" at the end of the code
2. Stop encoding if $x = 1$
3. Insert the binary representation of x at the beginning of the code
4. Change x, to be equal the number of bits just inserted minus one ($x = \lfloor \log_2 x \rfloor$)
5. Go back to step 2

The second method is an iterative application of Elias gamma coding and update of the input, which builds the code by changing bits at the beginning of it, as follows

1. Apply Elias gamma coding on x and generate code
2. Set $x = \lfloor \log_2 x \rfloor$ and go back to step 1 to replace the preceding "0"s until only one "0" is left
3. Move the left "0" to the end of the code

As an example, the Elias omega code for $x = 15$ is $C_\omega = 1111110$. Omega codes that can be generated by both these methods are listed in Table 2.9. A MATLAB implementation of the Elias omega coding method is presented in the following listing.[14]

```
function y = elias_omega_code( x, varargin )
   method = 1;
   if nargin>1,
      method = varargin{1};
   end
   if method == 1,
      % METHOD 1:
      % To code a number N:
      % Place a "0" at the end of the code.
      % If N=1, stop; encoding is complete.
      % Prepend the binary representation of N to the beginning of the code.
      %       This will be at least two bits, the first bit of which is a 1.
      % Let N equal the number of bits just prepended, minus one.
      % Return to step 2 to prepend the encoding of the new N.
      % As described in: https://en.wikipedia.org/wiki/Elias_omega_coding
      y = [ '0' ];
      while x ~= 1,
         dts = dec2bin( x );
         y = [ dts y ];
         x = length( dts )-1;
      end
   else
      % METHOD 2:
      % Call Elias gamma coding
      % Recursively apply Elias gamma coding on the num of zeros at front
      % Move the first zero to the end
      y = elias_gamma_code( x );
      N = strfind( y, '1'); N = N(1)-1;
      while N>1
         yy = elias_gamma_code( N );
         y = [ yy y(N+1:end)];
         N = strfind( y, '1'); N = N(1)-1;
      end
      y = [ y(2:end) y(1) ];
   end
```

Elias actually suggested the method now known as *arithmetic coding*; arithmetic coding was presented by Abramson in his *Information Theory and Coding* in 1963 (Abramson 1963), but remained almost a scientific curiosity until practical implementations were presented by researchers among which Rissanen, Pasco, Rubin and Witten (Rissanen 1976; Pasco 1976; Rubin 1979; Rissanen and Langdon 1979; Witten et al. 1987) can be identified as the most prominent. Arithmetic coding plays an important role in various image compression standards, such as the famous Joint Bilevel Image Group (JBIG) and JPEG (and respectively the newer Joint Bilevel Image Group 2 (JBIG2) and JPEG2000). There are two main factors in arithmetic coding: the probabilities of symbols and coding intervals. The probabilities of symbols define and yield compression efficiency and they determine the size of the intervals for the encoding process. Through an iterative process of updating prob-

[14] This MATLAB function can be executed only for scalar inputs.

Table 2.9 Elias gamma, delta and omega codes of natural numbers

x	$C_\gamma(x)$	$C_\delta(x)$	$C_\omega(x)$
1	1	1	0
2	010	0100	100
3	011	0101	110
4	00100	01100	101000
5	00101	01101	101010
6	00110	01110	101100
7	00111	01111	101110
8	0001000	00100000	1110000
9	0001001	00100001	1110010
10	0001010	00100010	1110100
20	000010100	001010100	10100101000
30	000011110	001011110	10100111100
40	00000101000	0011001000	101011010000
50	00000110010	0011010010	101011100100
100	0000001100100	00111100100	1011011001000

abilities of symbols and intervals, all symbols go through the encoder and the end result converges to a real number (theoretically of infinite accuracy between 0 and 1). An arithmetic encoder can be regarded as an encoding device that receives as input the symbols to be encoded and the corresponding estimate for the probability distribution, and produces a string of a length equal to the combination of the ideal code-lengths of the input symbols (Rissanen and Langdon 1979).

Simply described, in arithmetic coding *a message* is represented by the interval [0, 1) on the real axis, and is supposed to be governed by an underlying probability distribution. Each symbol of the message has a probability of occurrence (described by a source model) and is being processed sequentially. During this process each symbol narrows the overall interval according to its probability. Apparently, as the interval becomes smaller and smaller with the intercepted symbols, the amount of binary digits required to define it grows respectively. Since the narrowing of the interval depends upon the probability of the processed symbol, high probability symbols narrow the interval less than low probability, which implies that high probability symbols contribute fewer bits to the final code. Obviously, in arithmetic coding there exists no one-to-one correspondence between source symbols and output codes, as an entire sequence of symbols is encoded to a single arithmetic code. After the encoding of all message symbols the initial interval [0, 1) is diminished to an interval represented by the product of all probabilities of encoded symbols

$$P = p(s_1) \times p(s_2) \times p(s_3) \times \ldots \times p(s_N)$$

The interval precision expressed as the number of bits required to represent the interval is

$$-\log_2 P = -\sum_i \log_2(p(s_i))$$

The algorithm originally attributed to Elias (on a DMS) can be understood as a mapping of each symbol of a source to a distinct interval on the real axis $[c_n, c_n+a_n) \subseteq [0, 1)$, so that the length of this interval is equal to $f_I(s_n)$ the probability of the symbol s_n in the message I. The algorithm is recursive and is summarized in the following steps (Taubman and Marcellin 2002b)

- Initialize $c_0 = 0$ and $a_0 = 1$ so that the initial interval is $[0, 1)$
- For each symbol intercepted $n = 0, 1, ...$
  - Update $a_{n+1} \leftarrow a_n f_I(s_n)$
  - Update $c_{n+1} \leftarrow c_n + a_n F_I(s_n)$

where F_I is the cumulative distribution

$$F_I(s_j) = \sum_{k=0}^{j-1} f_I(s_k), \quad S_I = s_0, s_1, ...$$

In order for this to work, both the encoder and the decoder need to have access to the same distribution function f_I (and hence to the F_I), otherwise they should both be able to make the same assumptions about the underlying distribution. In practice the iterative process of the generation of arithmetic codes has been simplified as a recursive update of the interval start point and its length; thus for a message expressed by the sequence of symbols $S = \{s_1, s_2, ..., s_N\}$ governed by a source model that includes the symbol probabilities

$$p(l) = p(s_k = l), \quad l = 0, 1, ..., M-1, \quad k = 1, 2, ..., N$$

M being the number of symbols the source can produce, and a cumulative distribution practically defined as

$$c(l) = \sum_{j=0}^{l-1} p(j), \quad j = 0, 1, ..., M, \quad c(0) \equiv 0, c(M) \equiv 1$$

the process defines the interval

$$\Phi_k(S) = [\alpha_k, \beta_k), \quad k = 0, 1, ..., N, \quad 0 \le \alpha_k \le \alpha_{k+1}, \quad \beta_{k+1} \le \beta_k \le 1$$

This interval is represented for simplicity as $|b, l\rangle$ with b its starting point (also referenced as the 'base') and l its length,

$$|b, l\rangle = [\alpha, \beta), \quad b = \alpha, l = \beta - \alpha$$

Then this interval is being iteratively narrowed according to the symbols intercepted using

$$\begin{aligned} \Phi_0(S) &= |b_0, l_0\rangle = |0, 1\rangle \\ \Phi_k(S) &= |b_k, k_k\rangle = |b_{k-1} + c(s_k) l_{k-1},\ p(s_k) l_{k-1}\rangle, \quad k = 1, 2, ..., N \end{aligned} \tag{2.164}$$

A MATLAB implementation of the arithmetic coding method is presented in the following listings, first for the encoder and then the decoder.

```
%
% implementation of the Elias/Arithmetic encoding
%
% implementation is based on
%    Amir Said, "Introduction to Arithmetic Coding Theory and Practice",
%    Hewlett-Packard Laboratories Report, HPL-2004-76, Palo Alto, CA, April 2004
%    https://software.intel.com/sites/default/files/m/b/6/3/HPL-2004-76.pdf
%
function varargout = elias_arithmetic_encode( symbol_probabilities, symbol_indices )
   %% check input and initialize
   if (sum( symbol_probabilities )- 1) > eps,
      error( 'Probabilities of symbols do not add up to 1!');
   end
   c = [0 cumsum( symbol_probabilities )];
   l = 1;
   b = 0;
   %% run the main algorithm
   for i=1:length( symbol_indices )
      j = symbol_indices(i);
      b = b + c(j)*l;
      l = l*symbol_probabilities(j);
   end
   %% produce the output
   code = b+l/2;                                  % compute the code
   bin_code = num2bin( code );           % convert the code to binary form
   %% handle the output
   switch nargout
      case 0
         fprintf( ['| The code is in [%.12f,%.12f] --> [%.12f]\n',...
            '| Represented by the binary code [%s]\n'],...
            b,b+l,code,bin_code);
      case 1
         varargout{1} = code;
      otherwise
         varargout{1} = code;
         varargout{2} = bin_code;
   end
```

```
%
% implementation of the Elias/Arithmetic decoding
%
% implementation is based on
%    Amir Said, "Introduction to Arithmetic Coding Theory and Practice",
%    Hewlett-Packard Laboratories Report, HPL-2004-76, Palo Alto, CA, April 2004
%    https://software.intel.com/sites/default/files/m/b/6/3/HPL-2004-76.pdf
%
function varargout = elias_arithmetic_decode( symbol_probabilities, elias_code,
     num_symbols )
   %% check input and initialize
   if (sum( symbol_probabilities )- 1) > eps,
      error( 'Probabilities of symbols do not add up to 1!');
   end
   c = [0 cumsum( symbol_probabilities )];
   s = [];
   %% run the main algorithm
   for i=1:num_symbols
      % find the upper bound of the interval that includes the current code
      s(end+1) = find(c<elias_code, 1, 'last' );
      % normalize the code to [0,1)
      elias_code = (elias_code-c(s(end))) / symbol_probabilities(s(end));
   end
   %% handle the output
   switch nargout
      case 0
         disp( s );
      otherwise
         varargout{1} = s;
   end
```

If for example, there is a source that produces four (4) symbols $S = \{1, 2, 3, 4\}$ with a corresponding probabilities $p(s_k) = \{0.2, 0.4, 0.3, 0.1\}$ and this source produces the message $M = \{2, 2, 1, 3, 3, 2, 1, 4, 3, 2\}$ then the encoding of this message would proceed as follows:

- The cumulative distribution is computed by the probabilities resulting $c = \{0, 0.2, 0.6, 0.9, 1\}$. The lower bound of the initial interval is set to zero (0) and the length of this interval to unity (1).
- The first symbol in the message is a "2" with a probability of 0.4, which points to the interval $|0.2, 0.4\rangle$.
- The second symbol is also a "2", which narrows the interval to $|0.28,\ 0.16\rangle$.
- The third symbol is a "1", which narrows the interval to $|0.28,\ 0.032\rangle$.
- The fourth symbol is a "3", which narrows the interval to $|0.2992,\ 0.0096\rangle$.
- The fifth symbol is also a "3", which narrows the interval to $|0.30496,\ 0.0028\rangle$.
- Then comes a "2", which zooms in the interval $|0.305536,\ 0.001152\rangle$.
- A "1", narrows the interval to $|0.305536,\ 2.304 \times 10^{-4}\rangle$.
- A "4", narrows the interval to $|0.30574336,\ 2.304 \times 10^{-5}\rangle$.
- A "3", narrows the interval to $|0.305757184,\ 6.912 \times 10^{-6}\rangle$.

- And the final symbol "2", narrows the interval to $|0.3057585664,\ 2.7648 \times 10^{-6}\rangle$. This corresponds to the interval $[0.305758566400,\ 0.305761331200)$.
- The easiest way to get a single number in the final interval is to get the middle point, which in this case is 0.3057599488, or, in binary form, is 0.0100111001000110 0100100010110100.

Typically this process is followed by another step to minimize the length of the binary code by estimating the number of bits to keep from the length of the final interval.

Decoding of this output to reconstruct the original message (the sequence of symbols $M = \{2, 2, 1, 3, 3, 2, 1, 4, 3, 2\}$) is straightforward. The output of the encoder (0.3057599488) passes though an iterative process that tracks the interval the code belongs to (in the cumulative distribution) and normalizes the interval to become $[0, 1)$. Apparently *the decoder has no clue on when to stop the process*, so additional data, like the number of encoded symbols, have to be stored and passed to the decoder for a proper decoding.

An in-depth analysis of the theory and practical application of arithmetic coding can be found in (Said 2004), where important implementation concepts are being presented.

Practical implementations of arithmetic encoders produce strings that exceed the ideal length (at approximately 6 %). A typical practical arithmetic encoder is shown in Fig. 2.72. In general, the probability of a sample having a specific value is influenced by the values of adjacent samples. Thus, the probabilities of the symbols can be estimated, provided that the values of neighboring symbols are known. For a given neighborhood of symbols, every possible combination of symbols reflects a *contextual pattern*. The arithmetic coder is very efficient in coding sequences where the probabilities are changing for every contextual pattern. Figure 2.73 shows a causal neighborhood of pixels in an image.

Statistical Modeling

The entropy coder presented in previous paragraphs, along with others not referenced here, use a statistical model of the source that produces the symbols being encoded. Apparently the assumption of such an underlying model is crucial for any

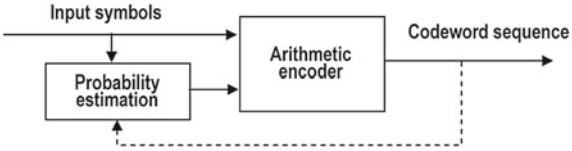

Fig. 2.72 Typical arithmetic encoder

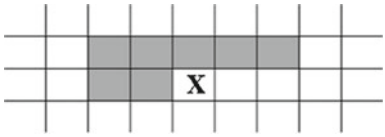

Fig. 2.73 Causal region for the conditional probability estimation

coding process, as it is responsible for determining both the alphabet of the source symbols and the entropy bound for any possible coding rate achievable. Presupposing a *stationary model* where the probabilities of the symbols do not change then, according to the theory of Shannon, the entropy of the source is the ultimate limit for any possible data compression. Nevertheless, this model is far from being accurate and in practice many techniques are being employed to alter the description of the symbols in order to achieve a lower limit and be able to design far more efficient coding schemes. In the classic textbook on JPEG, Pennebaker and Mitchell (Pennebaker and Mitchell 1993) briefly and intuitively presented four possible ways to improve the entropy and achieve better compression rates by,

- *Changing the alphabet*
  According to this approach, the source alphabet may be altered using possibly very simple observations on the occurrences of the symbols in intercepted messages produced by the source. For example, in natural images it is typical that run-lengths of identical pixel values are very probably so instead of using an alphabet consisting of the values of those pixels, one might consider encode the state of intercepting the same value. This way, the 'same value state' would be assigned significantly less bits (being the most frequent) and the new alphabet would be expected to be of reduced entropy.
- *Removing redundancy*
  In addition to the identification of patterns in the occurrence of the symbols of the alphabet presented in the previous approach, it is highly likely that some of the state transitions (described by the newly adopted alphabet) are improbable (there is no way of them to happen), apparently like each state change to itself. Removing these cases from the alphabet, a new, reduced alphabet is created and the entropy is further reduced as redundant symbols are being discarded.
- *Grouping symbols*
  Pushing further the development of the alphabet created so far with the two previous approaches, another improvement is expected to stem from the grouping of consecutive occurrences of alphabet symbols. A complete alphabet for all possible combinations could be created this way, lowering even more the number of bits per symbol needed for highly likely combined symbol occurrences. Typically, the probability for the occurrence of two symbols in a row (joint occurrence) is represented by the product of the probabilities of the individual symbols, thus less probable symbol transitions are highlighted whereas high probable transitions are suppressed in terms of the bits needed to represent them, which has a significant positive impact in the entropy of the new complex alphabet.
- *Using conditional probabilities*
  If the assumed *statistical independence of the symbols of the alphabet* in the previous approaches is discarded, then the source is no longer memoryless, and becomes an *x-order Markov process*. In this modeling of the source, the occurrence of a symbol is expected to be influenced (up to an order) by previously intercepted symbols. By employing conditional probabilities of the symbols these influences are being captured and efficiently represented as newly defined (more complex)

alphabet symbols that can impose a further reduction in the entropy. Actually, it has long been recognized that this approach yield very efficient alphabets and significantly reduced redundancy that guarantees a more compact representation in a compression attempt. Widely used methods, like JPEG, heavily rely on this approach to define what is usually referenced *the context* of a decision during entropy coding.

The approaches are not simple theoretical curiosities, but rather important practical strategies to remove large portions of the redundancy inherent in any source of information that produces sequences of symbols to be encoded. By targeting to remove this redundancy, it is expected to achieve a reduction in the entropy of the source, and create a highly efficient alphabet for use in an encoding process. In addition, the approaches aim at providing a more accurate statistical model of the source (either by supposing a statistical independence or by using Markov modeling), which is of paramount importance in many data encoding schemes that rely on probabilities, such as the entropy coding schemes, which have gained high acceptance in numerous applications over many decades of data compression developments.

2.3.3 Lossy Coding and Rate-Distortion Theory

Transformation, quantization and encoding of arbitrary real numbers is at the core of any data coding method. Unfortunately, real numbers require an infinite amount of bits to be accurately represented. Thus, it is expected that a finite representation of a continuous random variable cannot be perfect. In such a context, one has to consider to trade-off between accuracy and size of any possible representation. Upon this basic notion, a *rate-distortion theory* has been created that relates the minimum expected distortion achievable at a particular data rate, given the source statistics and an arbitrary distortion measure. Assuming a source that produces a sequence of *independent identically distributed* random variables with probability $p(x),\ x \in \mathcal{X}$, and assuming that the alphabet is finite, the encoder describes the source sequence X^n by and index $f_n(X^n) \in \{1, 2, ..., 2^{nR}\}$ and the decoder represents X^n by and estimate $\hat{X}^n \in \mathcal{X}$. In this respect, a *distortion function* or *distortion measure* is defined as a mapping from the set of source and reproduction alphabet pairs to the set of non-negative real numbers $d : \mathcal{X} \times \hat{\mathcal{X}} \rightarrow \mathcal{R}^+$. Accordingly, the distortion $d(x, \hat{x})$ is a measure of the cost of representing x by $\hat{x}$. Typical distortion measures usually being used are the *Hamming distortion*

$$d(x, \hat{x}) = \begin{cases} 0 & x = \hat{x} \\ 1 & x \neq \hat{x} \end{cases} \tag{2.165}$$

or the *squared-error distortion* $d(x, \hat{x}) = (x - \hat{x})^2$. In addition, the distortion between sequences x^n and $\hat{x}^n$ is defined as

$$d(x^n, \hat{x}^n) = \frac{1}{n} \sum_{i=1}^{n} d(x_i, \hat{x}_i) \tag{2.166}$$

thus the average per symbol distortion is equivalent to the distortion for a sequence.

The rate-distortion function $R(D)$ is the infimum of rates R such that (R, D) is in the rate-distortion region of the source for a given distortion D. Looking at it the other way around, the distortion-rate function $D(R)$ is the infimum of all distortions D such that (R, D) is in the rate-distortion region of the source for a given rate R, as already presented in (2.140). Accordingly, the *information rate-distortion function* $R^{(I)}(D)$ is defined as,

$$R^{(I)}(D) = \min_{p(\hat{x}|x): \sum_{(x,\hat{x})} p(x)p(\hat{x}|x)d(x,\hat{x}) \leq D} I(X; \hat{X}) \tag{2.167}$$

with the minimization over all conditionals $p(\hat{x}|x)$ for which $p(x, \hat{x}) = p(x)p(\hat{x}|x)$ satisfies the expected distortion constraint. Additionally, it can be proved that the definition of the rate-distortion function is equal to the latter information rate-distortion function definition, thus $R(D) = R^{(I)}(D)$ for an independent identically distributed source X with a distribution $p(x)$ and a bounded distortion function $d(x, \hat{x})$ (Cover and Thomas 2006).

In (Cover and Thomas 2006) is shown that for a binary (Bernoulli) source and considering a Hamming distortion, the rate-distortion function is defined as

$$R(D) = \begin{cases} H(p) - H(D) & 0 \leq D \leq \min\{p, 1-p\} \\ 0 & D > \min\{p, 1-p\} \end{cases} \tag{2.168}$$

whereas for a Gaussian source $\mathcal{N}(0, \sigma^2)$ with a squared-error distortion, the rate-distortion function is defined as

$$R(D) = \begin{cases} \frac{1}{2} \log \frac{\sigma^2}{D} & 0 \leq D \leq \sigma^2 \\ 0 & D > \sigma^2 \end{cases} \tag{2.169}$$

Apparently, there are cases in which 'good' approximations of an input are considered 'enough' as regards an expected outcome. In those cases, one trades off information 'quality' for a more compact (compressed) representation, as it is in those cases only that one is able to *surpass the entropy barriers*. Shannon already described these cases in his basic information theory, where he realized and put forward that in such conditions one should considered a balance between the compression rate achieved and the distortion that is imposed on the data.

As already mentioned on the general compression methods in the previous section, when applying lossy compression methods the degeneration of the original image is

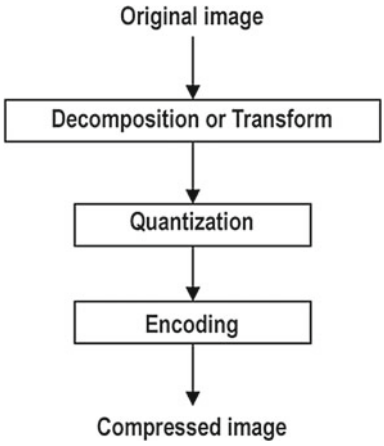

Fig. 2.74 The basic framework of lossy compression

permitted in order to further reduce the length of the generated compression string. Distortions can either be visible or not. Generally, a higher compression is achieved as the quality of the final image is continuously reduced (or respectively as the requirements for image quality are being reduced the compression efficiency is increased). The basic framework of a lossy compression method is shown in Fig. 2.74.

It includes three basic processes: transformation, quantization and coding to an output code-string. The value of each process in the overall compression depends on the technique used. Of course, the more *intelligent*[15] a process, the better the quality that can be achieved for a given number of bits. The compression performance limit of each information source, as defined by its entropy, applies only to the case of lossless compression. In the case of lossy compression this is not true: the problematic in this case refers to *how much the compression ratio should at least be to keep the distortion within certain acceptable limits*. The level of distortion is usually a parameter defined by the user by controlling certain parameters, such as the level of quantization. This concern arises and is being studied through a class of theories known as *rate-distortion theory*. This theory puts theoretical performance limits for lossy compression according to some fidelity criteria. For a very large class of distortion measures and source models, the theory provides a rate-distortion function $R(D)$ (or respectively $D(R)$), as developed in previous sections on quantization, which has the following properties:

- For any given distortion level D, it is possible to find a coding method with efficiency close to R and average deformation close to D.
- It is impossible to find an encoder that achieves a compression ratio less than R, with distortion D (or less).

[15]The term is used metaphorically to denote the ability of a method to adapt to the input data and the exploitation of statistical, spectral or any other inherent redundancies.

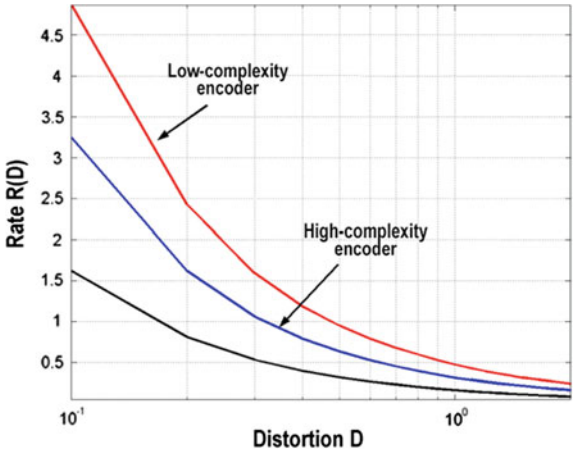

Fig. 2.75 Example of rate-distortion curves and typical encoders efficiency

It turns out that the curve $R(D)$ is convex, continuous and decreasing function of D. Figure 2.75 illustrates a typical rate-distortion function for a discrete source with finite alphabet. The minimum degree of compression required for compression without distortion is the value of R for $D = 0$, and is less than or equal to the entropy of the source and depends on the distortion estimator D. In the image ideal curves of encoders with high and low complexity are also shown. Generally, the better the compression method the better the source statistics are being modeled and the better the efficiency, which approaches the $R(D)$ threshold (Rabbani and Jones 1991a).

2.4 JPEG Compression

JPEG (Rabbani and Jones 1991a; ISO-IEC-CCITT 1993b; Pennebaker and Mitchell 1993; Wallace 1991) is a standardized image compression mechanism. Its name comes from the *Joint Photographic Experts Group*, the name of the group that introduced this standardization. JPEG was designed for compressing of either color or graylevel images of scenes that reflect the real world. It is a compression method which guarantees good compression results, primarily when running on real-world photos (or natural images). JPEG cannot provide good compression results on text images, simple graphics and line art. Also, is is mainly used for compressing still images (although there is a Motion-JPEG compression scheme for video sequences). JPEG is a lossy compression method. It was designed to exploit known properties of the HVS, with particular emphasis on its capacity to perceive changes in brightness a lot better than changes in chromaticity. A useful feature of JPEG is that the degree of distortion can be easily adjusted by changing the compression parameters. This means that one may choose (at least approximately) between image quality or the size of the final compressed file. Another important feature is that the JPEG decoders can

Table 2.10 Advantages and disadvantages of JPEG

Advantages	Disadvantages
Low memory requirements	A single image resolution (size)
Low complexity	A single image quality
Compression efficiency	Inability to predefine the compression rate
HVS modeling	No lossless compression
Robustness	No partial compression
	No region of interest
	Blocking artifacts in block boundaries
	Low noise resilience

use less accurate approaches in the calculations in order to achieve higher decoding speed (which, of course, affects the quality of the final image). The JPEG standard is applicable mainly in cases of compression of true color still images, achieving an average compression around 15 : 1. The advantages and disadvantages of JPEG compression standard are summarized in Table 2.10. The JPEG standard defines four modes of the basic algorithm:

- *Sequential coding*, in which each image block and each picture element is encoded following a scan path from top-left.
- *Progressive coding*, in which each image block and each picture element is encoded through multiple scans, to enable a progressive decoding of the image when there is a limitation in transmission times.
- *Lossless coding*,[16] in which the encoded image is coded so as to ensure accurate reproduction of the original image.
- *Hierarchical coding*, in which the image is encoded at multiple resolutions.

2.4.1 The Sequential JPEG

The sequential JPEG or baseline JPEG is the most basic of the modes that JPEG supports and it is the most usual mode of coding in JPEG. The block diagram of a typical sequential JPEG encoder is shown in Fig. 2.76. A decoder for such an encoder is follows the exact opposite path and the symmetric operations. Coupled, an encoder and a decoder are usually referenced as a *codec*. In the following paragraphs a description is given of the operation of each basic block of the baseline codec.

[16]There is a misunderstanding on this case, as JPEG does not support lossless compression. This mode does not have anything in common with the classical JPEG algorithm and can only support compression rates of about 2:1, using prediction in a causal neighborhood of pixels. It is essentially an entirely different method.

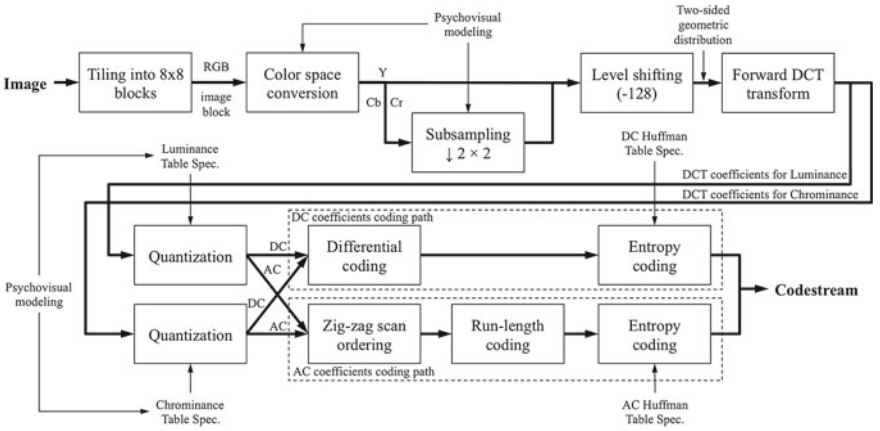

Fig. 2.76 Detailed block diagram of a sequential JPEG encoder

2.4.1.1 The Encoder

The encoding process in the sequential JPEG is performed in rectangular non-overlapping tiles of the image, usually called *blocks*, typically of 8×8 pixels in size. Initially, a color image is tiled and appropriately padded to the right and bottom, if needed. The following process is performed on each of the 8×8 image blocks separately. First, the color image undergoes a color space transformation, from the RGB color space, in which is usually represented, to the YC_bC_r color space, which, as previously analyzed, decorrelates the pixel values by providing a luminance-chrominance representation. According to the psychovisual modeling implied by the HVS modeling, the two chrominance channels are being subsampled by 2×2 (half samples in both directions), as it is expected that discarding data in these channels will not have a significant impact in the final reconstructed image quality, while it will greatly facilitate a better compression outcome. The luminance channel does not undergo any subsampling, due to its significance for the HVS. Then, the samples with values in the interval $[0, 2^b - 1]$ are shifted in the interval $[-2^{b-1}, 2^{b-1} - 1]$ by subtracting 128 from all samples. For each image channel in the case that the data rate is initially $b = 8$ bps which corresponds to all values being in the interval $[0, 255]$, these values are shifted to the interval $[-128, 127]$, thereby creating a two-sided geometric distribution with zero mean, apparently, increasing, in parallel, the number of bits needed for the new representation by 1 ($b' = 9$ bps). These shifted values of the luminance and subsampled chrominance pixels pass through DCT using (2.83) (which further increases the *bit-budget* of the data by outputing large and small floating point coefficients),

$$\left.\begin{aligned} J(u,v) &= \frac{2}{N}c(u)c(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} I(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \\ c(\xi):\ c(\xi=0) &= \sqrt{\frac{1}{2}},\quad c(\xi\neq 0)=1 \\ N &= 8 \end{aligned}\right\} \Rightarrow$$

$$J(u,v) = \frac{1}{4}c(u)c(v)\sum_{x=0}^{7}\sum_{y=0}^{7} I(x,y)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16} \tag{2.170}$$

The original discrete signal of 64 integer samples is a function of two dimensions x and y, whereas the real values of the DCT coefficients are a function of u and v. The first transform coefficient $J(0,0)$ is called the *DC coefficient* and the remaining 63 coefficients are called the *AC coefficients*. In a classic case of a natural (continuous-tone) image tile of 8×8 pixels, many AC coefficients are expected to have low or zero magnitudes, thus, need not be encoded. This fact, complemented by the psychovisual modeling that guides to rely only on low-frequency coefficients, forms the foundation for achieving compression in JPEG.

In the next step, all coefficients are quantized using a quantization matrix, which is determined according to the compression ratio required and is different for the luminance and the chrominance channels. The quantization is nonlinear and has been described in the previously presented section on *Quantization*; it uses the reference quantization matrices in (2.145) and applies the rule described in (2.146) to convert the reference matrices to the matrices that correspond to the quality factor used in the JPEG compression.

The purpose of this quantization is to reduce the values of the transform coefficients and especially of those coefficients that are expected to have minimal contribution to the reconstruction of the image; an ultimate goal at this step is to increase the number of coefficients with zero value in a way that complies with the psychovisual studies. Quantization in JPEG is typically defined as,

$$J_q(u,v) = \left\lfloor \frac{J(u,v)}{Q(u,v)} \right\rceil \tag{2.171}$$

for each of the luminance-chrominance channels, where $\lfloor\rceil$ denotes the rounding operator, and $Q(u,v)$ are the elements of the quantization matrices, which are rounded and clipped within the interval $[1, 255]$.

After the quantization, the quantized DCT coefficients are being reordered according to a zig-zag ordering as shown in Fig. 2.77. This ordering is required for the next step, the entropy encoding, since the low-frequency coefficients that have high probability of being non-zero are being ordered before the coefficients of high frequencies, whose probability of being zero is high. A simple test on various natural (continuous-tone) images reveals that the zig-zag reordering is very close to being the

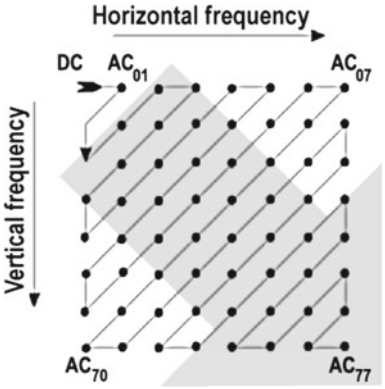

Fig. 2.77 The zig-zag ordering of the DCT coefficients

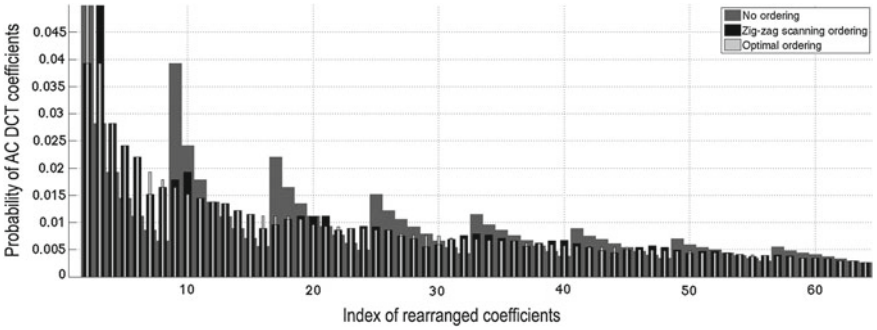

Fig. 2.78 Probability of non-zero AC coefficients of DCT and coefficients ordering

optimum possible reordering—actually the one being absolutely adaptive, adapted to the dataset.

Figure 2.78 aggregates the results of such a test on the Wang database of 1,000 natural color images,[17] where three different orderings of the AC coefficients of the DCT are being presented. The horizontal axis represents the index of the coefficients and the vertical axis the probability of each coefficient over the total amount of 8×8 image blocks in all 1,000 images; the dark grey bar-graph shows the probabilities without any reordering; the black bar-graph shows the probabilities of the coefficients after zig-zag ordering; the light grey bar-graph shows the 'optimum' reordering, where all coefficient probabilities are in a monotonically decreasing order. Apparently the zig-zag ordering is vary close to being 'optimum' and it does not need to be adaptive (it still applies for any other images). Purely for reference, Table 2.11 shows the indices for the coefficients required by the zig-zag ordering and those imposed by the optimum ordering (for this test set). The ordering is shown in a column-wise setting

[17] The Wang database is accessible at http://wang.ist.psu.edu/docs/related/.

Table 2.11 Comparison of the indices in zig-zag and optimum (adaptive) ordering

Zig-zag ordering								Optimum (adaptive) ordering							
1	9	2	3	10	17	25	18	1	3	2	4	5	6	10	9
11	4	5	12	19	26	33	41	8	7	11	12	13	14	15	19
34	27	20	13	6	7	14	21	21	20	18	17	24	25	23	16
28	35	42	49	57	50	43	36	22	26	33	34	32	27	35	28
29	22	15	8	16	23	30	37	31	40	39	36	38	41	30	47
44	51	58	59	52	45	38	31	37	29	42	46	48	45	49	43
24	32	39	46	53	60	61	54	52	51	53	44	50	54	57	56
47	40	48	55	62	63	56	64	58	55	59	60	61	63	62	64

just like the presented indices are in a column-wise ordering in the 8×8 grid, both beginning from the top-left corner and walking towards the right to the end of the columns before advancing one row.

The DC coefficients, which constitute the average of the 64 image samples, are being encoded using reversible encoding with prediction, DPCM, as shown in the Fig. 2.79. The reason for the use of such a coding scheme is that in adjacent image blocks the DC coefficients are expected to be highly correlated, and thus the value of their difference is expected to be very close to zero. Entropy coding of such small values are expected to produce small codewords, and, ultimately, the compression is significantly improved, as only the first DC coefficient has to be encoded to represent its full value and all others are encoded as very small differences.

Final stage in JPEG compression is the entropy coding, which achieves additional compression to create the final codestring via the coding of quantized DCT coefficients into a more 'compact' binary form. The JPEG standard specifies two entropy coding methods: *Huffman coding* and *arithmetic coding*. The basic sequential coding algorithm uses Huffman coding, mainly due to patents in the arithmetic coding proposed in JPEG. The Huffman encoder converts the quantized coefficients into a compact binary format by performing a two-step process, which creates a sequence of intermediate symbols before assigning the final Huffman codes to produce the codestream. The first step produces a set of *intermediate symbols*, which are a *category code* for the DPCM encoded DC coefficients, and a more complex *run-length and category code* for the AC coefficients.

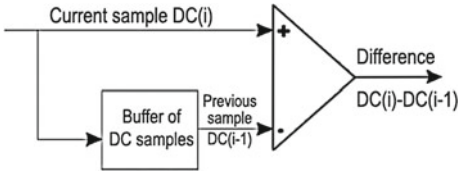

Fig. 2.79 Predictive coding (DPCM) for the DC coefficients

Table 2.12 Coding categories for DCT coefficients in JPEG standard for up to 12-bpp images

DC category	AC category	Range of values	
0	N/A	0	
1	1	(-1,1)	
2	2	(-3,-2)	(2,3)
3	3	(-7...-4)	(4...7)
4	4	(-15...-8)	(8...15)
5	5	(-31...-16)	(16...31)
6	6	(-63...-32)	(32...63)
7	7	(-127...-64)	(64...127)
8	8	(-255...-128)	(128...255)
9	9	(-511...-256)	(256...511)
A	A	(-1023...-512)	(512...1023)
B	B	(-2047...-1024)	(1024...2047)
C	C	(-4095...-2048)	(2048...4095)
D	D	(-8191...-4096)	(4096...8191)
E	E	(-16383...-8192)	(8192...16383)
F	N/A	(-32767...-16384)	(16384...32767)

Table 2.13 Reference table of Huffman codes for luminance DC coefficients

DC category	Code length	Huffman codeword
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
A	8	11111110
B	9	111111110

Specifically, the DC coefficient differences are mapped to difference categories in accordance with the definition in the JPEG standard shown in Table 2.12. Then this category is checked against another reference table from the JPEG standard, which assigns Huffman codes to the category codes as shown in Table 2.13 for the luminance and in Table 2.14 for the chrominance channels respectively.[18] Thus, a DC difference of -13, would be mapped to the category 4 in the DC coefficients categories, according to Table 2.12, which outputs the Huffman codeword "101" for luminance (length of 3 bits). In addition, since a category 4 coefficient is being encoded 4 more bits are assigned, either as the 4 least significant bits of a positive

[18] These are Tables K.3 and K.4 in the JPEG standard.

Table 2.14 Reference table of Huffman codes for chrominance DC coefficients

DC category	Code length	Huffman codeword
0	2	00
1	2	01
2	2	10
3	3	110
4	4	1110
5	5	11110
6	6	111110
7	7	1111110
8	8	11111110
9	9	111111110
A	10	1111111110
B	11	11111111110

coefficient, or the 4 least significant bits of a negative coefficient minus one (1).[19] In this case the coefficient is negative, thus, the most appropriate complement of the codeword would be "(0011)-1" or "0010". So the final codeword for the DC difference −13 would be "1010010". During decoding, the codeword's first three bits "101" represent specifically and beyond any doubt (since the codes are prefix there is a one-to-one mapping with) the category in which the coefficient should be searched. Decoding "101" as category 4, guides to use the next four bits for the final decoding of the coefficient. Specifically, the next four bits "(0010) + 1" result in "0011", which is the twos-complement representation for the negative number −13.

Encoding of the AC coefficients is somehow more complex. As there is a large probability that null AC coefficients are to be encoded, their encoding considers run-lengths of zero coefficients in addition to the category coding applied to the DC coefficients. The encoding start by identifying the category of the non-zero AC coefficient using Table 2.12. Then the encoding assigns a codeword for the pair of the number of zero AC coefficients that precede the encoded non-zero coefficient and its category according to Table 2.15 (Tables K.5 and K.6 in the JPEG standard) for the luminance or Table 2.16 for the chrominance channels respectively. If a run-length of more than 16 zero AC coefficients are encountered, then the run-length reports a code for each 15 and continues normally for the next coefficients. Applying this coding method to the AC coefficient sequence,

$$0, 0, 0, 0, 0, 0, 0, 0, 123$$

[19] It should be stressed that the negative values are represented in the twos-complement format. Twos complement representation requires to flip all bits of the absolute value binary representation and then add 1.

Table 2.15 Reference table of Huffman codes for luminance AC coefficients

Zero runs	Category	Huffman codeword	Zero runs	Category	Huffman codeword
0	0	1010 (=EOB)			
0	1	00	8	1	111111000
0	2	01	8	2	1111111111000000
0	3	100	8	3	1111111110110110
0	4	1011	8	4	1111111110110111
0	5	11010	8	5	1111111110111000
0	6	1111000	8	6	1111111110111001
0	7	11111000	8	7	1111111110111010
0	8	1111110110	8	8	1111111110111011
0	9	1111111110000010	8	9	1111111110111100
0	A	1111111110000011	8	A	1111111110111101
1	1	1100	9	1	111111001
1	2	11011	9	2	1111111110111110
1	3	11110001	9	3	1111111110111111
1	4	111110110	9	4	1111111111000000
1	5	11111110110	9	5	1111111111000001
1	6	1111111110000100	9	6	1111111111000010
1	7	1111111110000101	9	7	1111111111000011
1	8	1111111110000110	9	8	1111111111000100
1	9	1111111110000111	9	9	1111111111000101
1	A	1111111110001000	9	A	1111111111000110
2	1	11100	A	1	111111010
2	2	11111001	A	2	1111111111000111
2	3	1111110111	A	3	1111111111001000
2	4	111111110100	A	4	1111111111001001
2	5	1111111110001001	A	5	1111111111001010
2	6	1111111110001010	A	6	1111111111001011
2	7	1111111110001011	A	7	1111111111001100
2	8	1111111110001100	A	8	1111111111001101
2	9	1111111110001101	A	9	1111111111001110
2	A	1111111110001110	A	A	1111111111001111
3	1	111010	B	1	1111111001
3	2	111110111	B	2	1111111111010000
3	3	111111110101	B	3	1111111111010001
3	4	1111111110001111	B	4	1111111111010010
3	5	1111111110010000	B	5	1111111111010011
3	6	1111111110010001	B	6	1111111111010100

(continued)

Table 2.15 (continued)

Zero runs	Category	Huffman codeword	Zero runs	Category	Huffman codeword
3	7	1111111110010010	B	7	1111111111010101
3	8	1111111110010011	B	8	1111111111010110
3	9	1111111110010100	B	9	1111111111010111
3	A	1111111110010101	B	A	1111111111011000
4	1	111011	C	1	1111111010
4	2	1111111000	C	2	1111111111011001
4	3	1111111110010110	C	3	1111111111011010
4	4	1111111110010111	C	4	1111111111011011
4	5	1111111110011000	C	5	1111111111011100
4	6	1111111110011001	C	6	1111111111011101
4	7	1111111110011010	C	7	1111111111011110
4	8	1111111110011011	C	8	1111111111011111
4	9	1111111110011100	C	9	1111111111100000
4	A	1111111110011101	C	A	1111111111100001
5	1	1111010	D	1	11111111000
5	2	11111110111	D	2	1111111111100010
5	3	1111111110011110	D	3	1111111111100011
5	4	1111111110011111	D	4	1111111111100100
5	5	1111111110100000	D	5	1111111111100101
5	6	1111111110100001	D	6	1111111111100110
5	7	1111111110100010	D	7	1111111111100111
5	8	1111111110100011	D	8	1111111111101000
5	9	1111111110100100	D	9	1111111111101001
5	A	1111111110100101	D	A	1111111111101010
6	1	1111011	E	1	1111111111101011
6	2	111111110110	E	2	1111111111101100
6	3	1111111110100110	E	3	1111111111101101
6	4	1111111110100111	E	4	1111111111101110
6	5	1111111110101000	E	5	1111111111101111
6	6	1111111110101001	E	6	1111111111110000
6	7	1111111110101010	E	7	1111111111110001
6	8	1111111110101011	E	8	1111111111110010
6	9	1111111110101100	E	9	1111111111110011
6	A	1111111110101101	E	A	1111111111110100
7	1	11111010	F	1	1111111111110101
7	2	111111110111	F	2	1111111111110110
7	3	1111111110101110	F	3	1111111111110111

(continued)

Table 2.15 (continued)

Zero runs	Category	Huffman codeword	Zero runs	Category	Huffman codeword
7	4	1111111110101111	F	4	1111111111111000
7	5	1111111110110000	F	5	1111111111111001
7	6	1111111110110001	F	6	1111111111111010
7	7	1111111110110010	F	7	1111111111111011
7	8	1111111110110011	F	8	1111111111111100
7	9	1111111110110100	F	9	1111111111111101
7	A	1111111110110101	F	A	1111111111111110
			F	0	11111111001 (=ZRL)

the encoding process packs together the zeros before the 123 into the intermediate representation,

$$(8, 7)\ (123)$$

The zeros-runs/category code for (8, 7) in the Table of codes for luminance AC coefficients (Table 2.15) is

1111111110111010

whereas the binary code for 123 is 1111011, thus the codeword for the whole sequence would be

111111111011101011111011

which amounts to 23 bits for the encoding of 8 zeros and 1 integer.

If in the sequence of coefficients more than 16 consecutive zero coefficients are found, then the symbol (F, 0) is reported, which represents the maximum amount of consecutive zeros that can be grouped together into one codeword. Thus, the sequence

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, 23

would be considered as a grouping and coding of

$$(15, 0)\ (15, 0)\ (3, 5)\ (23)$$

and would be encoded as

11111111001 11111111001 1111111110010000 10110

where the spaces are shown only to illustrate the four parts that are being encoded.

Table 2.16 Reference table of Huffman codes for chrominance AC coefficients

Zero runs	Category	Huffman codeword	Zero runs	Category	Huffman codeword
0	0	1010 (=EOB)			
0	1	01	8	1	11111001
0	2	100	8	2	1111111110110111
0	3	1010	8	3	1111111110111000
0	4	11000	8	4	1111111110111001
0	5	11001	8	5	1111111110111010
0	6	111000	8	6	1111111110111011
0	7	1111000	8	7	1111111110111100
0	8	111110100	8	8	1111111110111101
0	9	1111110110	8	9	1111111110111110
0	A	111111110100	8	A	1111111110111111
1	1	1011	9	1	111110111
1	2	111001	9	2	1111111111000000
1	3	11110110	9	3	1111111111000001
1	4	111110101	9	4	1111111111000010
1	5	11111110110	9	5	1111111111000011
1	6	111111110101	9	6	1111111111000100
1	7	1111111110001000	9	7	1111111111000101
1	8	1111111110001001	9	8	1111111111000110
1	9	1111111110001010	9	9	1111111111000111
1	A	1111111110001011	9	A	1111111111001000
2	1	11010	A	1	111111000
2	2	11110111	A	2	1111111111001001
2	3	1111110111	A	3	1111111111001010
2	4	111111110110	A	4	1111111111001011
2	5	111111111000010	A	5	1111111111001100
2	6	1111111110001100	A	6	1111111111001101
2	7	1111111110001101	A	7	1111111111001110
2	8	1111111110001110	A	8	1111111111001111
2	9	1111111110001111	A	9	1111111111010000
2	A	1111111110010000	A	A	1111111111010001
3	1	11011	B	1	111111001
3	2	11111000	B	2	1111111111010010
3	3	1111111000	B	3	1111111111010011
3	4	111111110111	B	4	1111111111010100
3	5	1111111110010001	B	5	1111111111010101
3	6	1111111110010010	B	6	1111111111010110
3	7	1111111110010011	B	7	1111111111010111

(continued)

Table 2.16 (continued)

Zero runs	Category	Huffman codeword	Zero runs	Category	Huffman codeword
3	8	1111111110010100	B	8	1111111111011000
3	9	1111111110010101	B	9	1111111111011001
3	A	1111111110010110	B	A	1111111111011010
4	1	111010	C	1	111111010
4	2	111110110	C	2	1111111111011011
4	3	1111111110010111	C	3	1111111111011100
4	4	1111111110011000	C	4	1111111111011101
4	5	1111111110011001	C	5	1111111111011110
4	6	1111111110011010	C	6	1111111111011111
4	7	1111111110011011	C	7	1111111111100000
4	8	1111111110011100	C	8	1111111111100001
4	9	1111111110011101	C	9	1111111111100010
4	A	1111111110011110	C	A	1111111111100011
5	1	111011	D	1	11111111001
5	2	1111111001	D	2	1111111111100100
5	3	1111111110011111	D	3	1111111111100101
5	4	1111111110100000	D	4	1111111111100110
5	5	1111111110100001	D	5	1111111111100111
5	6	1111111110100010	D	6	1111111111101000
5	7	1111111110100011	D	7	1111111111101001
5	8	1111111110100100	D	8	1111111111101010
5	9	1111111110100101	D	9	1111111111101011
5	A	1111111110100110	D	A	1111111111101100
6	1	1111001	E	1	11111111100000
6	2	11111110111	E	2	1111111111101101
6	3	1111111110100111	E	3	1111111111101110
6	4	1111111110101000	E	4	1111111111101111
6	5	1111111110101001	E	5	1111111111110000
6	6	1111111110101010	E	6	1111111111110001
6	7	1111111110101011	E	7	1111111111110010
6	8	1111111110101100	E	8	1111111111110011
6	9	1111111110101101	E	9	1111111111110100
6	A	1111111110101110	E	A	1111111111110101
7	1	1111010	F	1	1111111010
7	2	11111111000	F	2	1111111111000011
7	3	1111111110101111	F	3	1111111111110110
7	4	1111111110110000	F	4	1111111111110111
7	5	1111111110110001	F	5	1111111111111000
7	6	1111111110110010	F	6	1111111111111001

(continued)

Table 2.16 (continued)

Zero runs	Category	Huffman codeword	Zero runs	Category	Huffman codeword
7	7	1111111110110011	F	7	1111111111111010
7	8	1111111110110100	F	8	1111111111111011
7	9	1111111110110101	F	9	1111111111111100
7	A	1111111110110110	F	A	1111111111111101
			F	0	1111111111111110 (=ZRL)

Finally, the symbol (0, 0) (i.e. no zeros and null coefficient) denotes the end of the sequence and is encoded as "1010" consuming 4 more bits in the final codestream (constant for each block in the image).

One might wonder why this coding stage is complicated, while it would be possible to simply apply Huffman coding of the quantized DCT coefficients. The answer is simple, if one considers the amount of data required to encode the coefficients; DC differentials are within the interval $[-2047, 2047]$ and AC coefficients in the interval $[-1023, 1023]$ for 8 bpp images, which corresponds to a requirement of code tables with 4,095 and 2,047 entries respectively. By adopting the encoding proposed in the JPEG standard, these tables are reduced to 12 and 160 entries respectively (with two more for the ZRL and EOB codes), which is a significant amount of complexity reduction.

Let us consider a full process example using a block selected from image 'lena' as shown in Fig. 2.80. This is a block from a color image so it is expected to have three channels, that is, three 8×8 matrices have to be encoded. First, the pixels of the image block undergo a color space conversion from RGB to YC_bC_r. Then the values are level shifted by subtracting 128. The level shifted values are transformed using DCT and quantized using the default quantization matrix in JPEG. The quantized transform coefficients are ordered according to the zig-zag scanning scheme and then each of them is entropy coded with the specific Huffman coding method in JPEG, using the

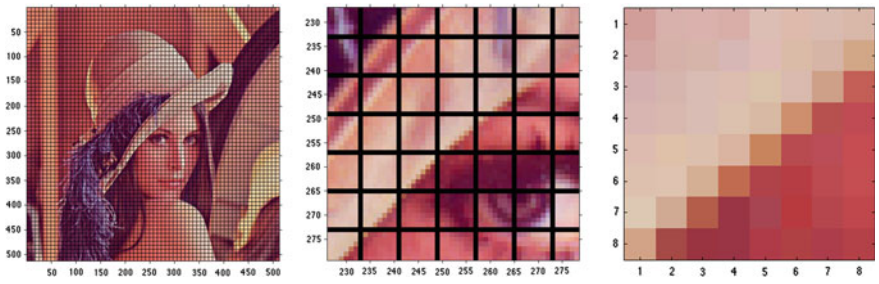

Fig. 2.80 Selected image block for JPEG compression experiments

default Huffman tables. The final produced codestreams $CODE_Y$, $CODE_{C_b}$, $CODE_{C_r}$ are shown with the individual codewords separated for illustration and verification.

$$R = \begin{bmatrix} 210 & 216 & 218 & 217 & 220 & 222 & 217 & 217 \\ 211 & 214 & 215 & 215 & 223 & 219 & 214 & 211 \\ 213 & 214 & 217 & 220 & 219 & 218 & 206 & 196 \\ 216 & 217 & 2188 & 219 & 217 & 206 & 183 & 192 \\ 221 & 223 & 222 & 218 & 201 & 184 & 187 & 198 \\ 222 & 223 & 213 & 193 & 176 & 182 & 187 & 196 \\ 221 & 209 & 179 & 154 & 168 & 184 & 181 & 192 \\ 211 & 165 & 153 & 154 & 175 & 175 & 173 & 180 \end{bmatrix} \quad Y = \begin{bmatrix} 164 & 177 & 178 & 176 & 186 & 184 & 183 & 181 \\ 170 & 177 & 178 & 181 & 185 & 183 & 179 & 168 \\ 177 & 179 & 184 & 185 & 187 & 182 & 163 & 123 \\ 180 & 182 & 185 & 187 & 181 & 152 & 112 & 111 \\ 185 & 187 & 187 & 178 & 144 & 109 & 111 & 113 \\ 187 & 189 & 173 & 128 & 102 & 100 & 109 & 113 \\ 191 & 169 & 116 & 89 & 102 & 98 & 105 & 109 \\ 166 & 103 & 88 & 89 & 98 & 100 & 98 & 102 \end{bmatrix}$$

$$G = \begin{bmatrix} 157 & 176 & 177 & 174 & 190 & 186 & 187 & 183 \\ 166 & 177 & 180 & 183 & 187 & 188 & 181 & 167 \\ 177 & 181 & 188 & 189 & 194 & 185 & 160 & 95 \\ 181 & 184 & 189 & 193 & 185 & 143 & 81 & 74 \\ 188 & 192 & 192 & 179 & 133 & 75 & 77 & 76 \\ 193 & 195 & 175 & 108 & 67 & 61 & 74 & 77 \\ 200 & 169 & 93 & 53 & 69 & 57 & 70 & 72 \\ 163 & 74 & 52 & 53 & 59 & 65 & 61 & 65 \end{bmatrix} \xRightarrow[YC_bC_r]{RGB} C_b = \begin{bmatrix} 164 & 177 & 178 & 176 & 186 & 184 & 183 & 181 \\ 170 & 177 & 178 & 181 & 185 & 183 & 179 & 168 \\ 177 & 179 & 184 & 185 & 187 & 182 & 163 & 123 \\ 180 & 182 & 185 & 187 & 181 & 152 & 112 & 111 \\ 185 & 187 & 187 & 178 & 144 & 109 & 111 & 113 \\ 187 & 189 & 173 & 128 & 102 & 100 & 109 & 113 \\ 191 & 169 & 116 & 89 & 102 & 98 & 105 & 109 \\ 166 & 103 & 88 & 89 & 98 & 100 & 98 & 102 \end{bmatrix}$$

$$B = \begin{bmatrix} 153 & 172 & 173 & 169 & 181 & 173 & 176 & 169 \\ 164 & 170 & 169 & 181 & 180 & 168 & 167 & 136 \\ 176 & 175 & 180 & 176 & 171 & 173 & 135 & 86 \\ 176 & 176 & 180 & 174 & 163 & 109 & 80 & 84 \\ 175 & 169 & 173 & 161 & 93 & 79 & 88 & 81 \\ 174 & 176 & 147 & 81 & 74 & 67 & 80 & 82 \\ 179 & 147 & 74 & 69 & 82 & 62 & 78 & 77 \\ 135 & 70 & 64 & 70 & 73 & 68 & 74 & 72 \end{bmatrix} \quad C_r = \begin{bmatrix} 152 & 146 & 146 & 147 & 142 & 145 & 142 & 144 \\ 148 & 145 & 144 & 142 & 144 & 143 & 143 & 150 \\ 144 & 143 & 141 & 143 & 141 & 143 & 150 & 173 \\ 144 & 143 & 141 & 141 & 144 & 158 & 173 & 179 \\ 143 & 143 & 143 & 146 & 161 & 176 & 176 & 181 \\ 142 & 142 & 147 & 167 & 175 & 181 & 177 & 180 \\ 139 & 147 & 167 & 171 & 171 & 183 & 176 & 180 \\ 151 & 168 & 172 & 171 & 178 & 176 & 176 & 178 \end{bmatrix}$$

$$Y = \begin{bmatrix} 36 & 49 & 50 & 48 & 58 & 56 & 55 & 53 \\ 42 & 49 & 50 & 53 & 57 & 55 & 51 & 40 \\ 49 & 51 & 56 & 57 & 59 & 54 & 35 & -5 \\ 52 & 54 & 57 & 59 & 53 & 24 & -16 & -17 \\ 57 & 59 & 59 & 50 & 16 & -19 & -17 & -15 \\ 59 & 61 & 45 & 0 & -26 & -28 & -19 & -15 \\ 63 & 41 & -12 & -39 & -26 & -30 & -23 & -19 \\ 38 & -25 & -40 & -39 & -30 & -28 & -30 & -26 \end{bmatrix} \quad C_Y = \begin{bmatrix} 11 & 12 & 1 & 1 & 0 & 0 & 0 & 0 \\ 16 & -6 & -6 & -1 & -1 & 0 & 0 & 0 \\ -3 & -6 & 3 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\xRightarrow[shift]{Level} C_b = \begin{bmatrix} -10 & -8 & -8 & -9 & -8 & -11 & -9 & -11 \\ -8 & -9 & -10 & -6 & -8 & -13 & -11 & -20 \\ -6 & -8 & -8 & -10 & -14 & -10 & -18 & -19 \\ -7 & -8 & -8 & -12 & -14 & -24 & -16 & -13 \\ -11 & -15 & -13 & -14 & -28 & -14 & -11 & -16 \\ -13 & -12 & -18 & -24 & -13 & -15 & -14 & -15 \\ -12 & -16 & -21 & -8 & -9 & -17 & -13 & -16 \\ -19 & -15 & -10 & -8 & -11 & -15 & -11 & -14 \end{bmatrix} \xRightarrow[Quantization]{DCT} C_{C_b} = \begin{bmatrix} -6 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$C_r = \begin{bmatrix} 24 & 18 & 18 & 19 & 14 & 17 & 14 & 16 \\ 20 & 17 & 16 & 14 & 16 & 15 & 15 & 22 \\ 16 & 15 & 13 & 15 & 13 & 15 & 22 & 45 \\ 16 & 15 & 13 & 13 & 16 & 30 & 45 & 51 \\ 15 & 15 & 15 & 18 & 33 & 48 & 48 & 53 \\ 14 & 14 & 19 & 39 & 47 & 53 & 49 & 52 \\ 11 & 19 & 39 & 43 & 43 & 55 & 48 & 52 \\ 23 & 40 & 44 & 43 & 50 & 48 & 48 & 50 \end{bmatrix} \quad C_{C_r} = \begin{bmatrix} 13 & -4 & 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

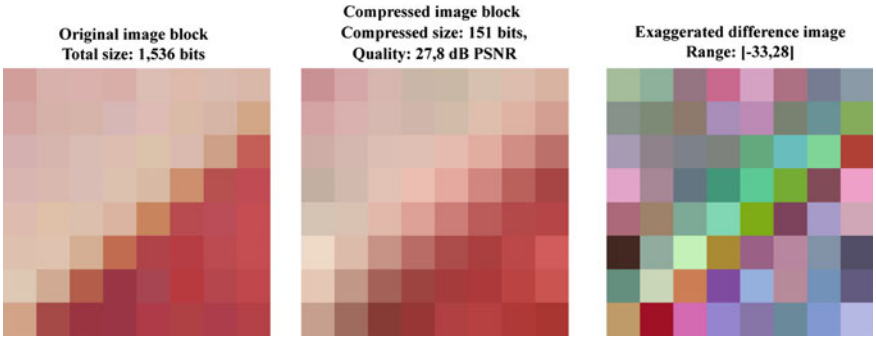

Fig. 2.81 Comparison of the original and reconstructed image block

$$
\begin{aligned}
C_Y^{zz} &= \{11, 12, 16, -3, -6, 1, 1, -6, -6, 1, -1, 1, 3, -1, 0, 0, -1, 1, 2, 0, 0, 0, 0, -1, -1, \\
&\quad 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\} \\
\xrightarrow[ordering]{Zig-zag}\ C_{C_b}^{zz} &= \{-6, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \\
&\quad 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\} \\
C_{C_r}^{zz} &= \{13, -4, -4, 0, 2, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, \\
&\quad 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\}
\end{aligned}
$$

$$
\begin{aligned}
CODE_Y &= 1011011\ 10111100\ 1101010000\ 0100\ 100001\ 001\ 001\ 100001\ 100001 \\
&\quad 001\ 000\ 001\ 0111\ 000\ 111000\ 001\ 0110\ 1110110\ 000\ 1010 \\
\xrightarrow[(JPEG\ Tables)]{Huffman\ coding}\ CODE_{C_b} &= 100001\ 001\ 001\ 1010 \\
CODE_{C_r} &= 1011101\ 100011\ 100011\ 1101110\ 111001\ 001\ 1010
\end{aligned}
$$

Apparently, the selected image block, which was a 3-channel 8×8 block of 8-bit pixels amounting 1,536 bits is now compactly represented by only 151 bits; this corresponds to a compression data rate of 0.786 bpp, or equivalently a compression ratio of about 10:1. Since the quantization was applied by using the reference quantization matrices (2.145) the reconstruction of the image by the 151-bits codestream is expected to produce only minimal or non-noticeable artifacts. Figure 2.81 shows side-by-side the original image block, the reconstructed block (after the JPEG compression) and the difference of the two.

2.4.1.2 The Decoder

In sequential decoding, all the steps of the encoding are being applied in the reverse order. Initially, the codestring representing the compressed data undergoes entropy decoding. The binary sequence is converted into a symbol sequence, and then the symbols are converted into DCT coefficients. Reversing the quantization is performed by a coefficient-wise multiplication with the quantization matrix,

$$J(u, v) = J_q(u, v) \times Q(u, v) \tag{2.172}$$

Inverse-DCT is performed on the quantized DCT coefficients in order to reconstruct the data from the frequency domain back to the two-dimensional spatial image domain. Using the definition of the inverse DCT in (2.85),

$$\left.\begin{aligned} I(x, y) &= \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} c(u)c(v)J(u, v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \\ c(\xi) &: \; c(\xi = 0) = \sqrt{\frac{1}{2}}, \quad c(\xi \neq 0) = 1 \\ N &= 8 \end{aligned}\right\} \Rightarrow$$

$$I(x, y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} c(u)c(v)J(u, v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16} \tag{2.173}$$

Finally, the decompressed data are level-shifted back to the interval $[0, 2^b - 1]$ (i.e. by adding 128 in the case of 8-bpp images) to reconstruct the original image data.

2.4.1.3 JPEG Compression Efficiency

The most basic estimator of the efficiency of a compression algorithm is the *Compression Ratio*, C_r, which is typically defined as the ratio of the original data to the compressed data and is expected to be greater that one,

$$C_r = \frac{\text{Original Data Size}}{\text{Compressed Data Size}} \tag{2.174}$$

Typically, the compression ratio is reported as a ratio $C_r : 1$, such as $2 : 1$.

As already pointed out, in image compression applications there is an interactive relationship presented as a trade-off between the compression ratio and the reconstructed image quality. A high compression ratio usually implies a low quality reconstructed image. Quality and compression may depend on the characteristics of the original image or the visual content of the scene. An image quality estimator, proposed by Wallace (1991), is the number of bits for a pixel (bpp) in the compressed image domain, the *Compression Rate*,

$$N_b = \frac{\text{Number of encoded bits}}{\text{Number of pixels}} \tag{2.175}$$

Table 2.17 presents four different image quality estimates derived by applying the estimator in (2.175). The results are accompanied by a subjective characterization

Table 2.17 Typical image quality related to various bit-rates

N_b [bits/pixel]	Image quality
0.25–0.5	Moderate to good
0.5–0.75	Good to very good
0.75–1.0	Excellent
1.5–2.0	Subtle difference from original

to 'moderate', 'good' or 'excellent', to denote what is roughly expected to be the judgment of an average observer.

Another estimator, which can be used for a variety of compression algorithms, is the classic RMSE, defined as a typical normalized Euclidean distance between the original and the compressed image,

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(I_i - \hat{I}_i)^2} \tag{2.176}$$

where I_i the original pixel values, $\hat{I}_i$ the compressed image pixel values and n the total number of pixels in the image.[20]

Figures 2.82 and 2.83 show examples of compression outcomes using the sequential JPEG encoding algorithm for a graylevel image (8 bpp) and a true color image (24 bpp). The results show the compressed image size, the compression ratio C_r, the compression rate N_b and the PSNR estimates, for five different compression quality factors. In the case of the color image, the quality estimate is based on the one proposed in (Kang and Leou 2003), in which the quality is assessed in the YUV color space, and the measurements of each channel ($PSNR_Y$, $PSNR_U$, and $PSNR_V$) are combined by using appropriate weights to create a single measurement value as,

$$PSNR = \frac{4 \cdot PSNR_Y + PSNR_U + PSNR_V}{6} \quad \text{dB} \tag{2.177}$$

The examples clearly demonstrate the strength of JPEG in compressing continuous-tone images, either graylevel or true color. Even for compression ratios of more than 16:1 the quality of the reconstructed image peaks high above 30 dBs of PSNR (which is usually used as a 'psychological' limit for noticeable artifacts). The graylevel image reconstructed from the compressed stream using the reference quality factor (50) presents a ratio of 16:1 and a data rate of 0.49 bpp with a high 37.4 dBs quality. The corresponding color image reports an impressive 42:1 ratio at 0.58 bpp with a 38.1 dBs quality. In both cases the visual appearance of the images is acceptable for most observers. In the extreme case of using the lowest quality

[20] It is worth noting that there are cases in which it is possible high RMSE values to correspond to visually acceptable quality.

Fig. 2.82 Baseline JPEG coding example for a graylevel image and five compression ratios

Fig. 2.83 Baseline JPEG coding example for a true color image and five compression ratios

factor (imposing the highest quantization), the images appear severely influenced by the compression but the basic image structure is still preserved and the quantization practically damages the tones of the images. The compression ratios are impressive at 70:1 and 163:1 for the graylevel and color image respectively. In addition, in the extreme case of using the highest quality factor, the outcome is a 2:1 and 4:1 compression for the graylevel and the color image respectively.

2.5 JPEG2000 Compression

JPEG2000 is the new standard (as of 2000) in digital image compression (Taubman and Marcellin 2002b; Rabbani and Joshi 2002; Rabbani and Cruz 2001; ISO-IEC 2000a; Christopoulos et al. 2002; Skodras et al. 2001; Christopoulos et al. 2000b; Santa-Cruz and Ebrahimi 2000a, b; Santa-Cruz et al. 2000). It appeared as a solution to issues in the existing standards, and as a response to the challenges posed by the new forms of information systems and the modern user requirements. The main axes on which the development of this new standard initiated were:

- Higher compression efficiency
- Multiple image resolutions within the same compressed file
- Presetting of the final data rate
- Quality scalability, and support for progressive decoding and scaling of the decoded image quality
- Lossy and lossless compression
- Partial or 'total' image compression
- Enhanced noise resilience
- Flexible codestream to support processing in the transform domain
- Robustness against recursive compression
- New type of file format to meet the requirements of modern digital photography, transmission among heterogeneous devices, better internal organization, code embedding, etc.

Even though it might sound paradoxical, the axis that played the least significant role in the development of JPEG2000 was that of the improved compression performance. Essentially, the motivation for this new compression standard was the addition of multiple functionalities and flexibility. The model on which this standard developed, was that of a typical compression system that is based on transform coding as typically depicted in Fig. 2.84.

The data transformation adopted and used in this standard was the wavelet transform (DWT). Quantization is this method is based on a conventional dead-zone scalar quantizer. The entropy coder is based on an implementation of a binary arithmetic encoder. In overall, the standard is quite open, and supports extensions and alternatives in almost all stages of the method, incorporating virtually all the solutions that were proposed at the stage of development by the researchers of the Working Groups, as well as other solutions that can meet specific future user needs. One of the

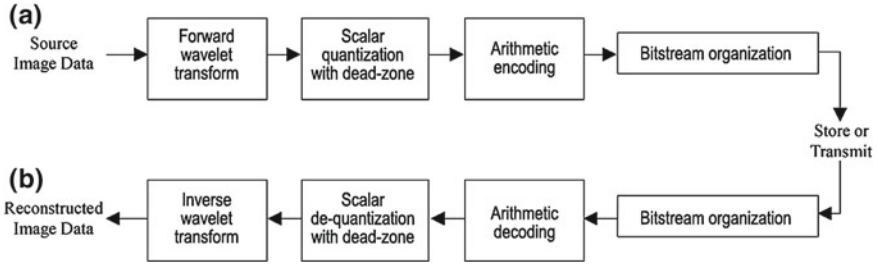

Fig. 2.84 The basic structure of the JPEG2000 codec, **a** the encoder and **b** the decoder

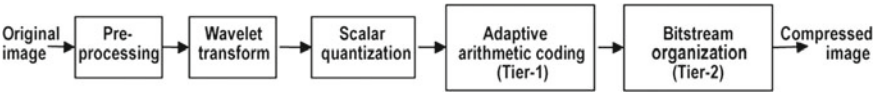

Fig. 2.85 The basic structure of the JPEG2000 encoder

most celebrated features is the flexible progressive coding, which can be performed either based on the position in the image, or on the size, or the quality, or the color component.

2.5.1 The JPEG2000 Encoder

A basic diagram of the encoder is shown in Fig. 2.85. A comprehensive graphical representation of the JPEG2000 encoding operation is shown in Fig. 2.86. Even thought the encoder usually constitutes an *informative* part of a compression standard, it is in this case highly sophisticated; it introduced several new concepts and ideas towards the optimization of the overall process, in addition to maintaining and supporting a number of options for alternatives. It is therefore worth studying the encoder proposed in JPEG2000 even for purely educational purposes, in order the grasp the insight of the researchers that made it possible.

2.5.1.1 Coding Preparation

At the beginning of the encoding process the image data undergo pre-processing. The pre-processing includes imperative (termed *normative* in JPEG2000) and optional (termed *informative* in JPEG2000) steps, including,

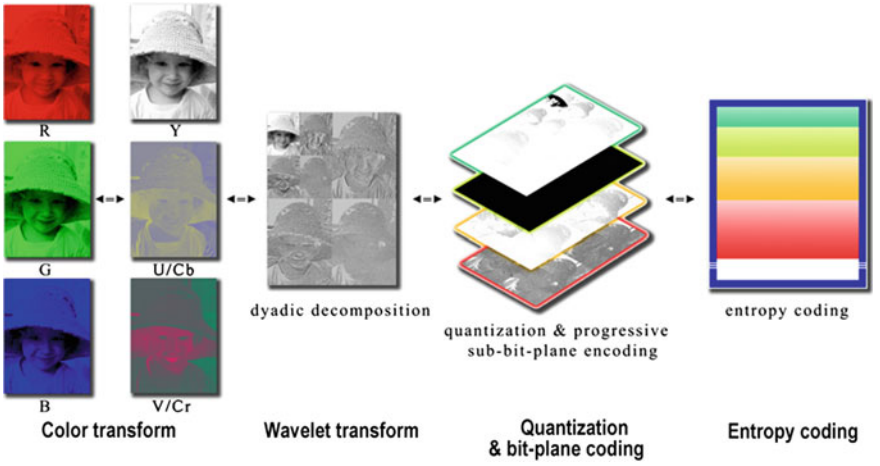

Fig. 2.86 Graphical representation of the JPEG2000 encoder operation

- *Tiling* (optional) of the image into non-overlapping regularly organized rectangular blocks for separate processing. This tiling is especially recommended for hardware implementations, where memory limitations and parallel processing requirements are of utmost importance. Figure 2.87 shows a random image tiling before the encoding process.
- *Level shifting of the values* of the samples (also termed *DC level shifting*) by subtracting a fixed quantity, the mean of the range of the values ($2^{b-1}, b = \text{bit} - \text{depth}$) from each sample, to generate a distribution symmetrically distributed around zero, in order to support the following functions without affecting the coding itself.
- *Color transformation* (optional) of the image for initial spectral decorrelation, provided that there are more than one color channels, and all the channels are of the same color depth (bpp) and dimensions. Figure 2.88 illustrates the color transformation from an initial RGB color space to a typical luminance-chrominance color space (either YUV or YC_bC_r). The transformations that are indicatively referenced in the JPEG2000 standard (informative) are the RGB to YUV (reversible transformation) and the RGB to YC_bC_r (irreversible transformation), defined as,

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 1/4 & 1/2 & 1/4 \\ 1 & -1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$
$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} = \begin{pmatrix} 0.29900 & 0.58700 & 0.11400 \\ -0.16875 & -0.33126 & 0.50000 \\ 0.50000 & -0.41869 & -0.08131 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.178}$$

Fig. 2.87 Tiling of an image before encoding

2.5.1.2 Image Transform

The wavelet transform (DWT) has been selected to replace the cosine transform (DCT) in JPEG2000, since it has been proven that it can be beneficial to coding for a number of reasons including,

- The DWT can be applied to the entire image and not only to distinct image tiles, and thus the appearance of discontinuities in the tile boundaries can be avoided
- The use of integer arithmetic DWT filters allows both lossy and lossless compression in the same codestream
- The DWT leads, by default, to multi-resolution representations, thus enabling a new mode of progressive coding
- The DWT results in representations of the image in various spectral bands and permits the application of different quantization in each band according to any selected HVS model

The application of the two-dimensional DWT is implemented as a separable transform (that is, two one-dimensional transforms), as shown in Fig. 2.50. The process is recursively applied to the LL band of the transform up to a particular desired level or up to the point in which it is not possible to further decompose the LL band (the

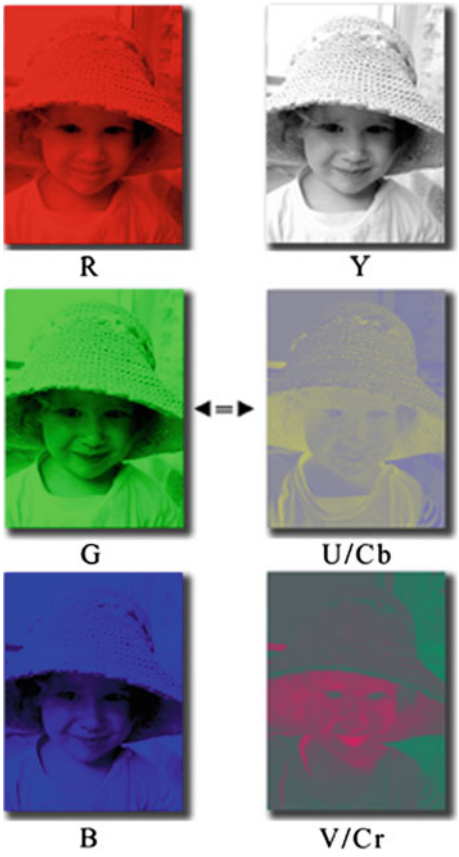

Fig. 2.88 Representation of the color space transformation

LL band consists only of one pixel). The result is a spectral representation of the original image into various zones as shown in Fig. 2.89.

After extensive study on various wavelet filtering options for use in theJPEG2000, researchers who participated in its development concluded in the adoption of two basic analysis-synthesis sets of filters, namely the *Le Gall or Integer (5, 3)* and the *Daubechies (9, 7) filter banks*.[21] The numbering in the names of the filter banks indicates the number of coefficients in the analysis and synthesis filters used in each pair. A representation of the application of the one-dimensional transform in terms of a filter bank pair is depicted in Fig. 2.90. By imposing the DWT to be biorthogonal, h_0 is orthogonal to to g_1 and h_1 is orthogonal to g_0. The absolute summations

[21] In the context of signal processing, a filter bank is a set of band-pass filters, each of which separates an input signal into one single frequency sub-band of the overall spectrum of the original signal. A graphic equalizer is usually referenced as one illustrative example of a filter bank, in which various components of the input signal are being attenuated separately and all components are recombined to form the modified signal at the output. Apparently this process includes two steps, a decomposition or an analysis and a reconstruction or a synthesis step.

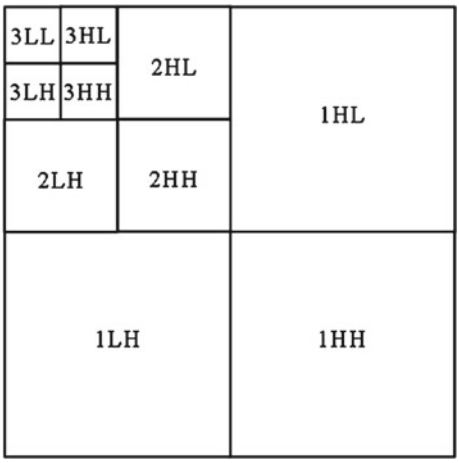

Fig. 2.89 Three-level 2-D DWT

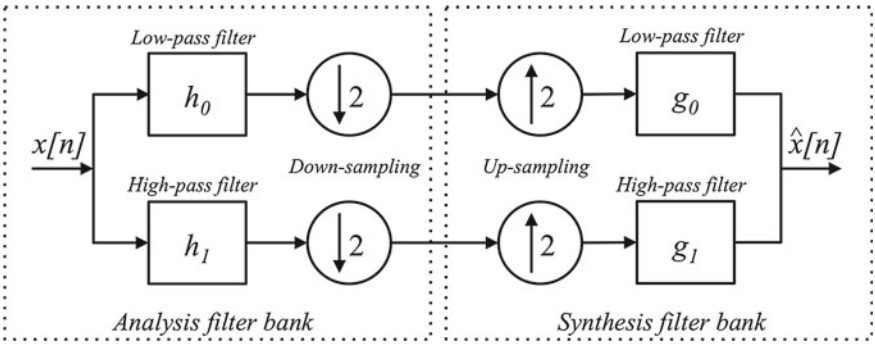

Fig. 2.90 DWT in terms of an analysis (*left*)—synthesis (*right*) filter bank

$$\left|\sum_n h_0[n]\right|, \qquad \left|\sum_n (-1)^n h_1[n]\right|$$

define the *DC gain* of the low-pass analysis filter and the *Nyquist gain* of the high-pass analysis filter respectively. The synthesis and analysis filters are related according to

$$\begin{aligned} g_0[n] &= a(-1)^n h_1[-n] \\ g_1[n] &= a(-1)^n h_0[-n] \end{aligned} \tag{2.179}$$

where

$$a = \frac{2}{\left(\sum_n h_0[n]\right)\left(\sum_n (-1)^n h_1[n]\right) + \left(\sum_n h_1[n]\right)\left(\sum_n (-1)^n h_0[n]\right)}$$

a normalization factor.

According to this definition, the analysis Le Gall (5, 3) filter bank consists of a low-pass filter of five (5) coefficients and a high-pass filter of three (3) coefficients,

$$\begin{aligned} &\text{low} - \text{pass filter} : h_1[n] \\ h_0 &= \left\{-\frac{1}{8}, \frac{2}{8}, \frac{6}{8}, \frac{2}{8}, -\frac{1}{8}\right\} \\ n &= -2, -1, 0, 1, 2 \\ &\text{high} - \text{pass filter} : h_1[n] \\ h_1 &= \left\{-\frac{1}{2}, 1, -\frac{1}{2}\right\} \\ n &= -2, -1, 0 \end{aligned} \tag{2.180}$$

Using (2.179) it is possible to compute the synthesis filter bank, as

$$\begin{aligned} &\text{low} - \text{pass filter} : g_0[n] \\ g_0 &= \left\{\frac{1}{2}, 1, \frac{1}{2}\right\} \\ n &= -2, -1, 0 \\ &\text{high} - \text{pass filter} : g_1[n] \\ g_1 &= \left\{\frac{1}{8}, \frac{2}{8}, -\frac{6}{8}, \frac{2}{8}, \frac{1}{8}\right\} \\ n &= -2, -1, 0, 1, 2 \end{aligned} \tag{2.181}$$

This filter bank uses integer arithmetic, whereas the analysis Daubechies (9, 7) filter, with a low-pass filter of nine (9) coefficients and a high-pass filter of seven (7) coefficients,

$$\begin{aligned} &\text{low} - \text{pass filter} : g_0[n] \\ g_0 &= \{0.026748757410, -0.016864118442, -0.078223266528, \\ &\quad 0.266864118442, 0.602949018236, 0.266864118442, \\ &\quad -0.078223266528, -0.016864118442, 0.026748757410\} \\ n &= -4, -3, -2, -1, 0, 1, 2, 3, 4 \\ &\text{high} - \text{pass filter} : g_1[n] \\ g_1 &= \{0.091271763114, -0.057543526228, -0.591271763114, \\ &\quad 1.115087052456, -0.591271763114, -0.057543526228, 0.091271763114\} \\ n &= -4, -3, -2, -1, 0, 1, 2 \end{aligned} \tag{2.182}$$

Using (2.179) it is possible to compute the synthesis filter bank, as

$$
\begin{aligned}
&\text{low} - \text{pass filter} : g_0[n] \\
g_0 &= \{-0.091271763114, -0.057543526228, 0.591271763114, \\
&\quad 1.115087052456, 0.591271763114, -0.057543526228, -0.091271763114\} \\
n &= -4, -3, -2, -1, 0, 1, 2 \\
&\text{high} - \text{pass filter} : g_1[n] \\
g_1 &= \{-0.026748757410, -0.016864118442, 0.078223266528, \\
&\quad 0.266864118442, -0.602949018236, 0.266864118442, \\
&\quad 0.078223266528, -0.016864118442, -0.026748757410\} \\
n &= -4, -3, -2, -1, 0, 1, 2, 3, 4
\end{aligned}
\tag{2.183}
$$

Figures 2.91 and 2.92 are graphical representations of the filter coefficients for the analysis and synthesis stage respectively, for both the filter banks in JPEG2000. It is noted that the analysis-synthesis filter bank scheme presented so far correspond to a one-dimensional one decomposition level wavelet transform. In case a multi-level decomposition is required then the transform is applied recursively on the low-pass filtered output at the analysis stage as shown in Fig. 2.93, where an N-level 1-D wavelet transform is presented. The output of the transform is a series of transform coefficients that includes L_N, H_N, H_{N-1}, ..., H_2, H_1. Since the two-dimensional DWT can be applied as a separable transform, it is usually applied as a sequence of two one-dimensional transforms, first applied on the columns of an image and then applied on the rows of the 'image' after the first step.

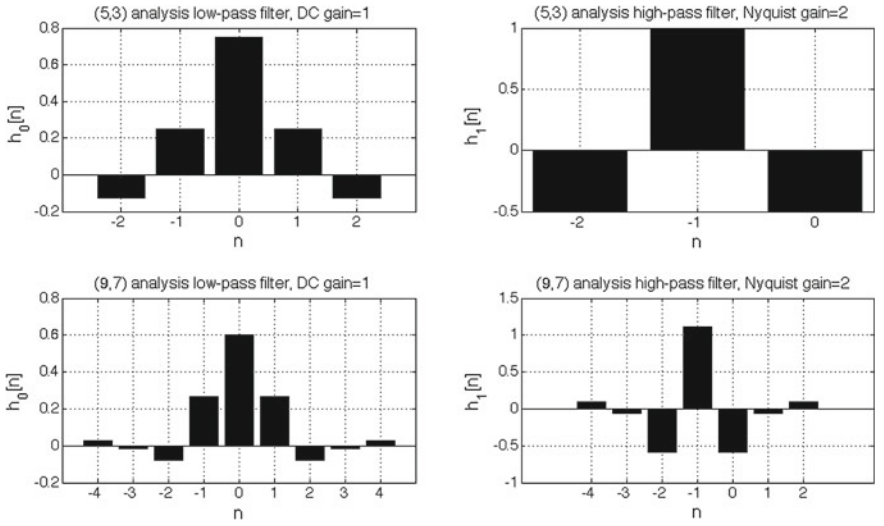

Fig. 2.91 Analysis stage filter banks (5, 3) and (9, 7) for JPEG2000

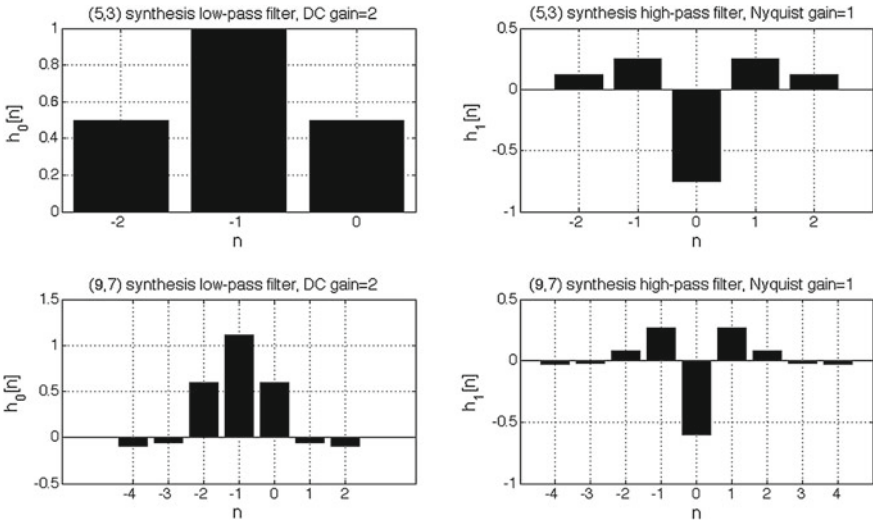

Fig. 2.92 Synthesis stage filter banks (5, 3) and (9, 7) for JPEG2000

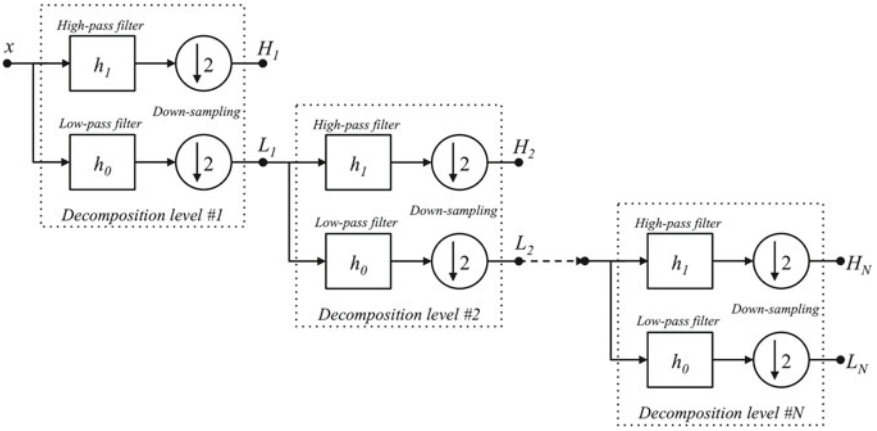

Fig. 2.93 Multi-level decomposition in 1-D DWT

Figure 2.94 shows a representation of the transform coefficients after the applications of one, two and three levels of two-dimensional DWT for a MRC image. It can de clearly seen in this representation that in each of the detail bands (all the bands except the LL at the upper-left part) the details captured correspond to a particular 'detail direction', vertical in the upper-right bands, horizontal in the lower-left bands and diagonal in the lower-right bands at each decomposition level. As depicted in Fig. 2.93 the decomposition at each new level is applied on the LL band (the low pass filtered part from the previous decomposition level). The double 'L' letters indicate the two-dimensional application of the low-pass analysis filter of the filter banks,

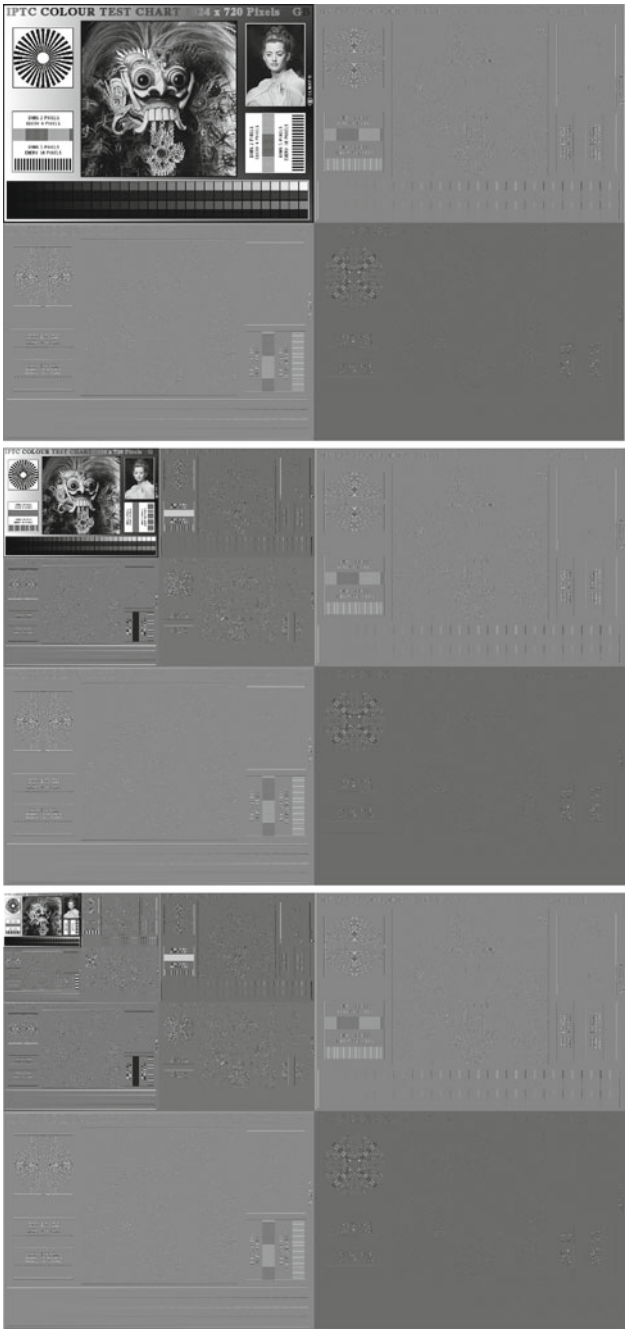

Fig. 2.94 Transform coefficients after a one/two/three-level 2-D DWT on a MRC image

column-wise and row-wise. In this example, the Le Gall (5, 3) filter bank has been used, and the resulting coefficient values have been scaled for a better illustration in Fig. 2.94.

2.5.1.3 Quantization

In sequential JPEG a uniform scalar quantizer is being used and a de-quantizer maps the quantized coefficients to the center of the quantization interval. For each of the DCT coefficients a different quantization step is being used in agreement with the HVS modeling, through the use of a quantization matrix of the same size as the part of the image gradually being encoded.

The same general principle applies in JPEG2000 with some variations introduced to meet the new requirements. One difference lies in the adoption of the central dead-zone quantizer. It has been shown (Sullivan 1996) that the optimal quantizer—from a rate-distortion point of view—for a continuous signal with Laplacian probability density (such as the the one of the transform coefficients) is a scalar quantizer with a central dead zone. The size of the dead zone as a fraction of the step increases with the reduction of the variance of the Laplacian distribution. Usually, its value does not exceed the value of two (2), being preferable to tend to one (1). In JPEG2000 the dead zone, typically, has two times the quantization step size, as shown in Fig. 2.95, although the standard supports any modification of the quantizer for each of the bands of the transform coefficients.

This specific quantizer is used in JPEG2000 because it exhibits efficient embedding features. This means that if a quantization index of M_b bits resulting from a quantizer with step size Δ_b is transmitted progressively starting with the most significant bit, the final index after decoding only N_b bits is identical to that which would have been produced by a similar quantizer with a step size of $\Delta_b 2^{M_b - N_b}$. This property ensures *progressiveness in quality*, which, from an optimization point of view, signifies that the decoder can stop decoding at any time having managed to reconstruct the same image that would have been produced by a quantization set to the same endpoint. It also allows the definition, in advance, of the compression data rate and the level of expected distortion.

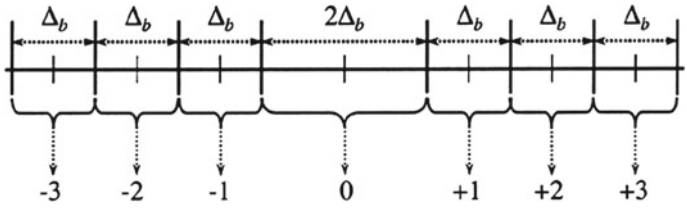

Fig. 2.95 Uniform scalar quantizer with central dead zone and step size Δ_b

Another significant feature of the JPEG2000 quantizer is that, in the inverse process, deviations from the middle point of the quantization interval are permitted for non-zero indices, in order to achieve a better adaptation to the skewness (asymmetry) in the probability distribution of the transform coefficients.

In the encoder, for each spectral band b of the DWT, a different quantization step size Δ_b is selected by the user, which is then used to quantize all the coefficients in that band. The selection of the step size may follow the perceptual significance of the content in that band (in terms of the HVS) (Albanesi and Bertoluzza 1995; Jones et al. 1995; O'Rourke and Stevenson 1995; Watson et al. 1997), or it may be a basis for achieving other goals, such as a better compression rate. The quantizer assigns a quantized value $q_b(u, v)$ to a transform coefficient $y_b(u, v)$ of band b, as shown in Fig. 2.95. Although, the quantization in an encoder is informative, it was initially proposed that a specific relation could be follows for quantization, that is

$$q_b(u, v) = sign[y_b(u, v)] \left\lfloor \frac{|y_b(u, v)|}{\Delta_b} \right\rfloor \tag{2.184}$$

b being the band and u, v, the spatial coordinates in the transform domain. The quantizing step size Δ_b is described by two bytes: a *mantissa* μ_b of 11-bits and an *exponent* ϵ_b of 5-bits, somehow complying with the ISO/IEC/IEEE 60559: 2011 or the IEEE Standard for Floating-Point Arithmetic (IEEE 754), as

$$\Delta_b = 2^{R_b - \epsilon_b} \left(1 + \frac{\mu_b}{2^{11}}\right) \tag{2.185}$$

where R_b is the number of bits representing the dynamic range of band b. It is noted that, when reversible encoding with a Le Gall (5, 3) filter-bank is being applied, then the quantization step size becomes unity (1).

The decoder becomes aware of the value of the quantization step size in two ways:

1. Transmission of the pair (ϵ_b, μ_b) for each band. This is called an *expounded quantization* and is similar to the approach being used in the older JPEG.
2. Transmission of the pair (ϵ_b, μ_b) only for the LL band as (ϵ_0, μ_0) and calculation of values for any other band. This is called a *derived quantization*, and involves the computation

$$(\epsilon_b, \mu_b) = (\epsilon_0 - N_L + n_b, \mu_0) \tag{2.186}$$

where N_L the total number of DWT decomposition levels, and n_b the decomposition level that corresponds to band b.

*The decoder needs not be signaled about the quantization step size in the case of reversible encoding (or lossless compression) as there is no quantization in this case ($\Delta = 1$).

In the decoder, when lossy encoding with a Daubechies (9, 7) filter-bank is applied, then a reconstructed coefficient $Rq_b(u, v)$ for a quantization step size Δ_b is estimated as

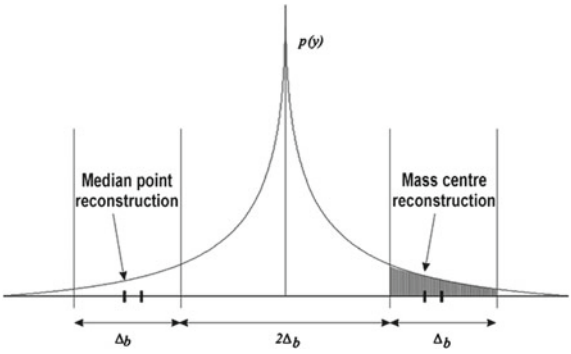

Fig. 2.96 Midpoint and mass center reconstruction in the quantization intervals, assuming a two-sided Laplacian distribution

$$Rq_b(u,v) = \begin{cases} \left[q_b(u,v) + \gamma\right]\Delta_b & q_b(u,v) > 0 \\ \left[q_b(u,v) - \gamma\right]\Delta_b & q_b(u,v) < 0 \\ 0 & \text{elsewhere} \end{cases} \tag{2.187}$$

where $\gamma \in [0, 1)$ is a reconstruction parameter selected by the decoder that controls the position in the quantization interval to report. Typically, when $\gamma = 0.5$ then a midpoint de-quantization occurs. Values for $\gamma < 0.5$ introduce a bias towards zero, which is expected to contribute to an improvement in the final decoded image quality, in cases in which the probability distribution of the coefficients is Laplacian. An empirical popular value is $\gamma = 0.375$, as this value corresponds roughly to the position of the centroid (mass center) of the quantization interval as shown in Fig. 2.96.

When a reversible Le Gall (5, 3) filter-bank is being used in the transform, the same principle in (2.187) applies with $\Delta_b = 1$; when lossless compression is required then the de-quantization involves virtually no operation, and the reconstructed coefficient is assigned the same value as the quantized coefficient,

$$Rq_b(u,v) = q_b(u,v) \tag{2.188}$$

$Rq_b(u, v)$ being the reconstructed (de-quantized) coefficient.

2.5.1.4 Entropy Coding

The final stage in the coding process is the entropy coding, which includes a novel bitsteam organization. In this stage the quantized transform coefficients pass through an entropy encoder to produce the final compression bitstream. In order to meet the requirements set for this standard, and especially to support codestream embedding so that progressive transmission and decoding is possible, a specific type of entropy encoder had to be selected. This is a bitplane encoder, which had already been tested in several transform coding schemes based on DWT, like in the Embedded image

coding using Zerotrees of Wavelet coefficients (EZW) (Shapiro 1993) and Set Partitioning In Hierarchical Trees (SPIHT) (Said and Pearlman 1996). In these codecs the correlation among the bands is being exploited to improve the compression efficiency, which, unfortunately hampers the error resilience during data transmission and the embedding functionality for a flexible progressiveness. To tackle with this issue, in JPEG2000 each wavelet band is encoded independently. In addition, JPEG2000 employs block coding in the transform domain as in Embedded Block Coding with Optimized Truncation (EBCOT) (Taubman 2000b). In this coding scheme each band is divided into small non-overlapping blocks, named *codeblocks*, which are independently encoded. Their dimensions are determined by the encoder and are restricted to be powers of 2, to have a height greater or equal to 4 and the number of coefficients in a codeblock not to exceed 4096. This coding scheme introduces a number of advantages since it enables

- easy random access to an image region
- parallel implementations
- improved segmentation and rotation capabilities
- improved error resilience
- efficient control of compression rate
- flexibility in shaping progressive forms

As will be explained in the following paragraphs, by adopting an efficient data rate strategy that enables the optimization of the percentage of participation of each codeblock in the final bitstream, JPEG2000 achieves a better compression efficiency than other existing standards (Taubman et al. 2002c).

According to the entropy coding stage in JPEG2000, the quantized transform coefficients are being encoded bitplane-by-bitplane (bit-by-bit) as shown in Fig. 2.97 (Rabbani and Joshi 2002), starting from the Most Significant Bit (MSB). During the process, each coefficient is considered *insignificant* as long as its already processed

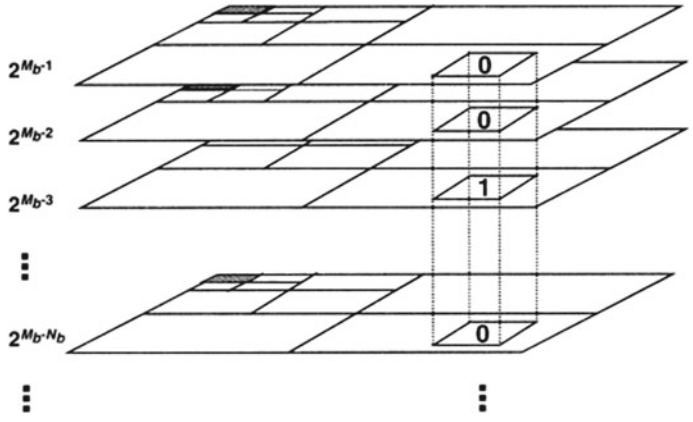

Fig. 2.97 Transform coefficients are encoded bit by bit starting from the MSB

bits are zero (for example, in Fig. 2.97 the coefficient is insignificant after the encoding of its first two significant bits). Once the process encounters a non-zero bit, then the coefficient becomes *significant* and its sign is encoded. Henceforth, the bits of this coefficient are called *precision bits*. Since the coefficients are expected to be of small values (due to the nature of the DWT to collect most of the energy of the image in low frequency coefficients), the quantized coefficients to be encoded are mostly *insignificant* in the early stages of coding, thereby producing very limited information about the specific bitplanes.

In JPEG2000 a very effective method is being used to exploit these redundancies, known as *adaptive binary arithmetic coding*.[22] One of the first implementations of an adaptive binary arithmetic coder was the *Q-coder* (Pennebaker et al. 1988) developed by IBM. A modified version of the Q-coder, the *QM-coder*, was selected for arithmetic coding in both JBIG and JPEG (Pennebaker and Mitchell 1993). Copyright issues, however, prevented its widespread in JPEG-based implementations. JPEG2000 adopted another modification of the Q-coder, the *MQ-coder*. This encoder is used in JBIG2 (ISO-IEC-ITU 2000) and its use was extended to JPEG2000.

Generally, the probability distribution of each bit symbol of a quantized coefficient is influenced by its previous coded bits, and the bits of the neighboring coefficients. The estimation of this probability is being done using contextual information generated by the current significance status of the coefficient, and the significance of the eight neighboring coefficients (in a typical 3×3 neighborhood), as defined by the current and previous bitplanes and based on the available, up to that point, encoded information. In arithmetic coding with contextual information, separate probabilities are being estimated and maintained for each context model, which are being updated based on a finite state machine, each time a symbol is encoded in the given context model. The MQ-coder selects among 46 modes for each context model: of these, modes 0 to 13 correspond to initialization and are used for fast convergence (fast attack) in robust probability estimation. Modes 14 to 45 correspond to probability estimates of the steady state. There is also a complementary non-adaptive mode (46), which is used for encoding symbols with equal probability distribution, and can neither enter or leave one of the other states. In practice, each band of coefficients is divided into codeblocks (typically, of size 64×64) and each bitplane of each codeblock is encoded in three passes (referenced also as *coding triplets*),

- *significance propagation pass*, in which the coefficients that are begin encoded are the ones that have not yet been marked as significant, and there are significant coefficients in their neighborhood; this is due to their high probability to become significant
- *magnitude refinement pass*, in which only significant coefficients are being encoded, so that their magnitude estimate is refined

[22] It is reminded that an *adaptive binary arithmetic coder* accepts the binary symbols of an input sequence, along with a corresponding probabilistic model, and outputs a codestream with a length of at most two bits greater than the combined ideal lengths of the code of the input symbols. By updating the probability estimate of symbols adaptivity is enabled (Pennebaker et al. 1988).

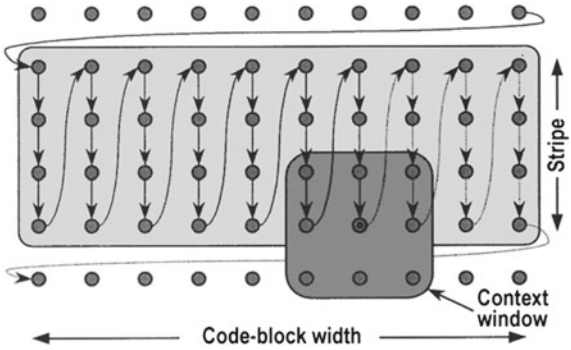

Fig. 2.98 Scanning flow in a codeblock

- *cleanup pass*, in which all non-processed coefficients of the bitplane are being encoded, keeping in mind that the first coding pass of the most significant bitplane is a cleanup phase; in addition, run-length encoding is also being applied at this phase

The order by which the coefficients are being encoded is shown in Fig. 2.98. The height of each vertical scan corresponds to four coefficients. At the end of this process that is called *Tier-1 coding*, a series of bitstreams for each encoded image codeblock are being produced, which should be properly arranged to form an *embedded bitstream*. This is the work of *Tier-2 coding* that follows.

The second phase of the final step of entropy coding, the *Tier-2 coding* consists virtually of a 'multiplexing' of the various bitstreams produced by *Tier-1 coding*, implemented through an efficient ordering. The aim of this step is to create a bitstream that allows easy access and flexible syntax control, guarantee progressiveness and enable region-of-interest coding. An important construct introduced in this phase is the *layer*, which is a collection of consecutive coding scans of all the codeblocks and coefficient bands. In this scheme, each codeblock contributes with a different number of coding passes. The organization of layers to achieve progressiveness in quality can be best shown through a graphic representations, as shown in Fig. 2.99 (Rabbani and Cruz 2001), which includes,

(a) a distribution and ordering of coding triplets by coefficient band and bitplane, where black squares correspond to the clean-up pass, the gray squares correspond to the significance propagation pass and white squares correspond to the refinement pass
(b) a selection of coding phases to fully reconstruct quality and resolution, indicated by a light gray cover over the total amount of data
(c) a selection of coding phases for lower resolution (small image) and maximum quality, indicated by a light gray cover over the the LL band and all coding passes and bitplanes
(d) a selection of coding phases for medium resolution and maximum quality, indicated by a light gray cover over the bands of the second decomposition level and all coding passes and bitplanes

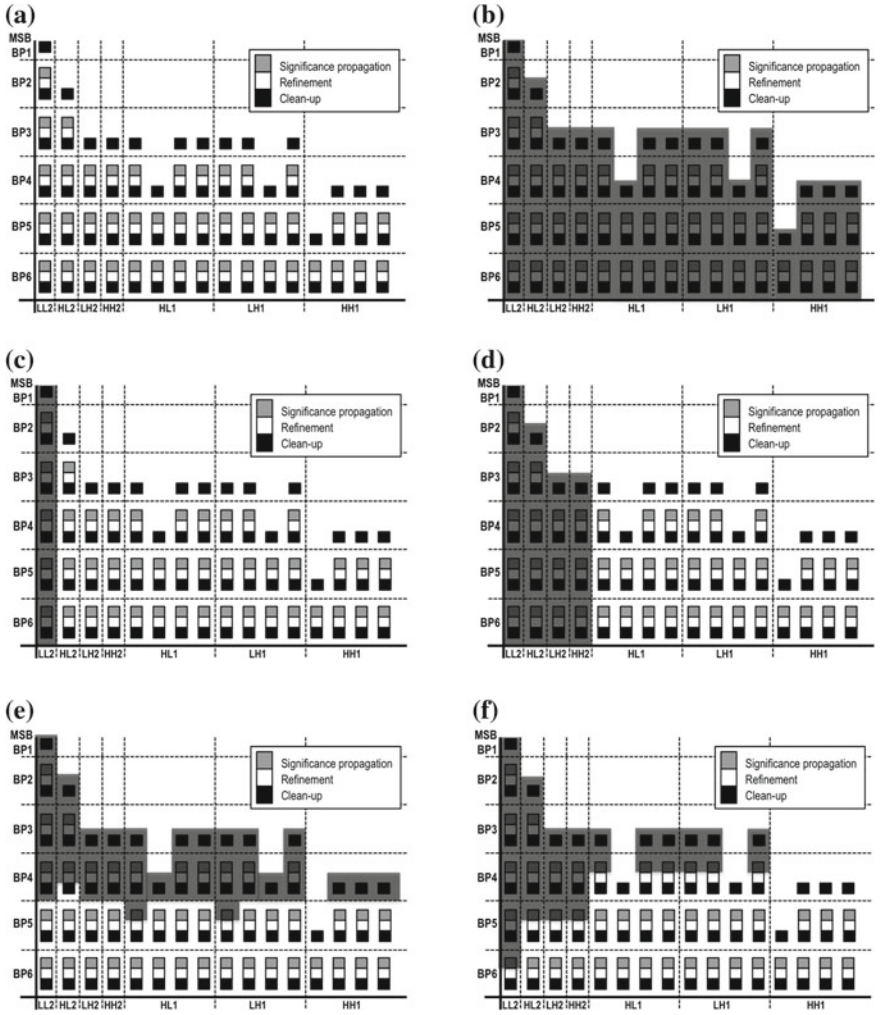

Fig. 2.99 Representation of various formations of layers by selecting results from Tier-1 coding

(e) a selection of coding phases for maximum resolution (full size) and a preset objective quality (SNR), indicated by a light gray cover over a selected part of the coding passes and bitplanes for all bands

(f) a selection of coding phases for maximum resolution and a preset subjective quality (data related to the HVS), indicated by a light gray cover over a selection of the coding passes, bitplanes and bands, using more data from higher decomposition levels

According to this organization, layers are being created using a collection of consecutive coding passes throughout the coefficient bands. Multiple layers may be

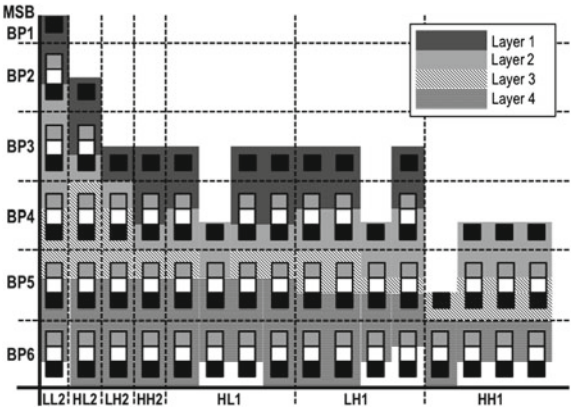

Fig. 2.100 An example of bitstream organization by the formation of four layers

created, one after another improving the image quality. The total number of layers may range from $2^0 = 1$ to $2^{16} - 1 = 65535$, with a popular selection being 20. Figure 2.100 shows an example of an organization that uses four layers (Rabbani and Cruz 2001). In practice, the codeblocks belonging to a *precinct* are encoded together. Precincts are regions in all the coefficient bands that correspond to the same image-domain pixel locations, and can be of arbitrarily large size, as long as their dimensions are powers of 2. By default, precincts are of size 15×15, which corresponds to a division of a resolution level of a component into rectangles of $2^{15} \times 2^{15}$ samples. Further, as the encoding finalizes the bitstream, *packets* are being formed, which represent a specific tile, layer, component, resolution level and precinct. Packets include a *packet head* which encodes informative data regarding the localization of the encoded coefficients, the number of all-zero bitplanes to be skipped, the number of included coding passes and the corresponding length of the data for each codeblock.

Progressiveness in the final encoded bitstream can be defined in terms of four parameters, namely

- Image quality (SNR) corresponding to parameter *Layer—L*
- Image size corresponding to parameter *Resolution—R*
- Color component corresponding to parameter *Component—C*
- Position in the image corresponding to parameter *Position—P*

Definition of the progressiveness during the encoding, which eventually results in reorganization of the packets, is accomplished by selecting the value of a specific byte, so as to indicate the priority of these four parameters. The standard supports five types of progression, namely

- Layer - Resolution - Component - Position (LRCP)
- Resolution - Layer - Component - Position (RLCP)
- Resolution - Position - Component - Layer (RPCL)
- Position - Component - Resolution - Layer (PCRL)
- Component - Position - Resolution - Layer (CPRL)

These progression priorities are switchable among the different tiles that the image has potentially been split. This series of parameters is expressed in their execution from the end to the beginning. For example, in LRCP, first the algorithm runs for Position, then for Component, Resolution, and finally the Layer. Therefore a priority in quality (Layer) denotes that all packets with a certain level of quality (or compression rate) will be arranged before the packets corresponding to the next level of quality.

2.5.2 Enhanced Features in JPEG2000

The JPEG2000 standard, apart from the basic coding structure, supports a series of optimizations and extensions, like those of the *region-of-interest (ROI) coding* and the advanced *error resilience*, which might be of utmost importance, especially in image transmission applications, in cases with requirements for fast communication using a limited bandwidth, or in cases in which corruption by noise occurs in addition to a transmission typically suffering from congestion and outage. These two functionalities are being reviewed in the following sections.

2.5.2.1 Region-Of-Interest Coding

Region-Of-Interest (ROI) Coding allows an uneven distribution of image quality and an arbitrary reorganization of the bitstream according to regions of higher interest. A ROI is encoded in better quality than the rest of the image, which is consequently considered to be *the background* (sometimes denotes *BG* for simplicity). Two classes of ROI are defined in JPEG2000, namely *static ROI* and *dynamic ROI*.

- *Static ROI*, which is determined during the phases of encoding; this ROI is preferable in cases of storage, static transmission, remote sensing applications, etc.
- *Dynamic ROI*, which is defined interactively by a user on a client-server application scenario within a progressive transmission process. This ROI is preferable in telemedicine applications, mobile communications, mobile devices, etc., and can be implemented through the creation of dynamic coding layers to better approach the requirements of the user.

In practice, ROI coding is implemented by using a ROI mask, which is a binary map that determines which of the DWT coefficients contribute to reconstruct the ROI. This mask undergoes a transformation that resembles the dyadic decomposition in DWT, in order to be transformed to a representation that actually maps the ROI in the transform domain, in all bands. In the simple case of a rectangular mask, it is not even required that the mask be an image, since there is an easy way for its definition. Figure 2.101 represents the mask generation process (following the dyadic decomposition in DWT) in an appropriate form to become a map for the coefficients in the ROI. Coding of a ROI is accomplished by shifting (Fig. 2.102)

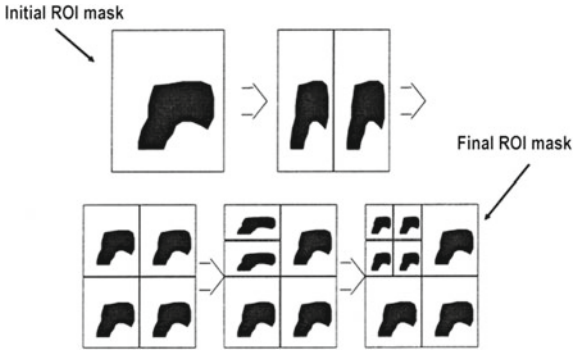

Fig. 2.101 ROI mask generation in JPEG2000

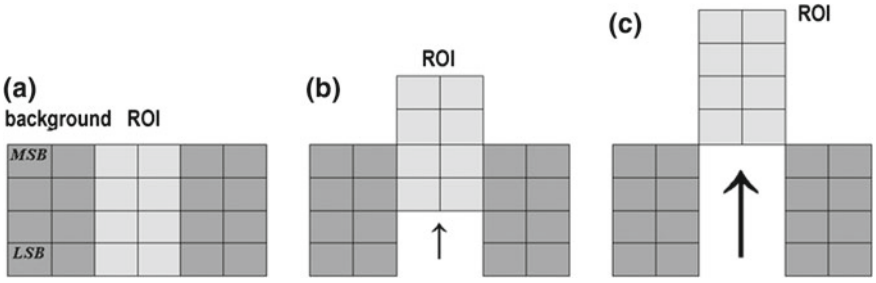

Fig. 2.102 Bit shifting for ROI coding: **a** definition of a ROI, **b** general scaling, **c** maxshift mode

only the transform coefficients that correspond to the ROI a number of bitplanes upwards (left shifting or multiplication by powers of two). As a consequence of this shifting, ROI coefficients are encoded first. The shifting of the bitplanes can vary for different ROIs, and the number of bitplanes that were shifted is stored in the header of the final bitstream to allow a proper decoding of the image. If the ROI coefficients rise above the MSB bitplane of the background, the method is called the *maxshift mode* (Rabbani and Cruz 2001; Christopoulos et al. 2000a), in which neither header data are required, nor the ROI mask itself, as the ROI is being decoded in its entirety before the decoding of the background, as the Least Significant Bit (LSB) of the ROI is higher than the MSB of the background.

2.5.2.2 Resilience to Transmission Noise

In general, typical image applications require the transmission of compressed image data through communication channels with various characteristics. For instance, wireless telecommunication networks are prone to random and sudden errors, whereas wired communications are vulnerable to data loss due to congestion; in addition, both are prone to outages. To address issues during image transmission, various

mechanisms for error resilience and data correction have been defined in JPEG2000. Error resilience may be achieved through various approaches, such as data partitioning and re-synchronization, error detection and concealment, and priority-based Quality of Service (QoS) transmission. Both the syntax and the native tools for error resilience are implemented at the entropy coding and packet transmission levels (Liang and Talluri 1999; Moccagata et al. 2000). Summarizing, the errors that may occur are

- *corruption in the body of a packet*, in the form of numerically affected (corrupted) encoded data for a codeblock, in which incorrect symbols are decoded and thus the context regions are updated with erroneous data in the subsequent bitplanes, and as a result *significant distortions occur*
- *corruption in the packet head*, in the form of corruption of sensitive data, such as the packet size, which may lead to a decoding of data that, possibly, do not correspond to the packet, increasing, at the same time, the uncertainty about subsequent packets, and as a result *loss of synchronization occurs*
- *loss of data*, such as packet loss at the network level, which may lead to combined error effects affecting both the packet body and the packet head

JPEG2000 supports a number of protection mechanisms against those issues, protecting both the data of the codeblocks and their headers,

- *Segmentation symbols*: in this mechanism, a special symbol sequence is encoded at the end of each bitplane. If a wrong symbol sequence is decoded an error has been detected and at least the latest bitplane is corrupt.
- *Regular predictable termination*: the arithmetic encoder is terminated at the end of each coding pass using a special predictable termination. The decoder reproduces the termination and if it does not detect the same unused bits at the end, an error is detected in at least the latest coding pass.
- *Simultaneous use of both segmentation symbols and regular predictable termination*: guarantees a better response but has an impact in compression performance, as the number of excess bits needed becomes significant.
- *Re-synchronization marker*: a special synchronization marker, the Start Of Packet (SOP), precedes each packet head, with a sequence index. If a SOP marker with a correct index is not found in the decoder, an error is detected and the decoder waits for the next unaffected packet in order for the decoding to resume.
- *Relocation of packet head symbols*: usage of Packed Packet Headers, Main Header (PPM) or Packed Packet Headers, Tile-Part Header (PPT) markers for the relocation of all packet head symbols to the main image or the corresponding tile header and transmission through a channel with lower error rate.
- *Use of precincts*: the use of precincts in the encoding leads to a reduction of the spatial coverage of the packets and thereby limits the effect of errors in a particular location in the image.

Any of these mechanisms or their combinations may be employed in an error protection strategy designed by a user for the purposes of specific applications. Each of the mechanisms provides an additional level of protection introducing, though, a

cost in compression efficiency of JPEG2000, as additional bits are required to enable each mechanisms. A user has to carefully study the scope of the application and the anticipated corruption, and balance the cost of introducing an appropriate mechanism with the data rate limits.

2.5.3 *Brief Evaluation of JPEG2000*

A typical example of a progressive decoding of a JPEG2000 compressed image up to full reconstruction (lossless) at 0.01, 0.025, 0.05, 0.1, 0.25, 2 bpp, is shown in Fig. 2.103. During this progression, the reconstructed image quality (measured as PSNR using (2.177)) increases from 27.55 dB up to 49.4 dB. It is apparent that

Fig. 2.103 Quality and data rate during decoding of a progressive-by-quality encoded image (0.01–2 bpp)

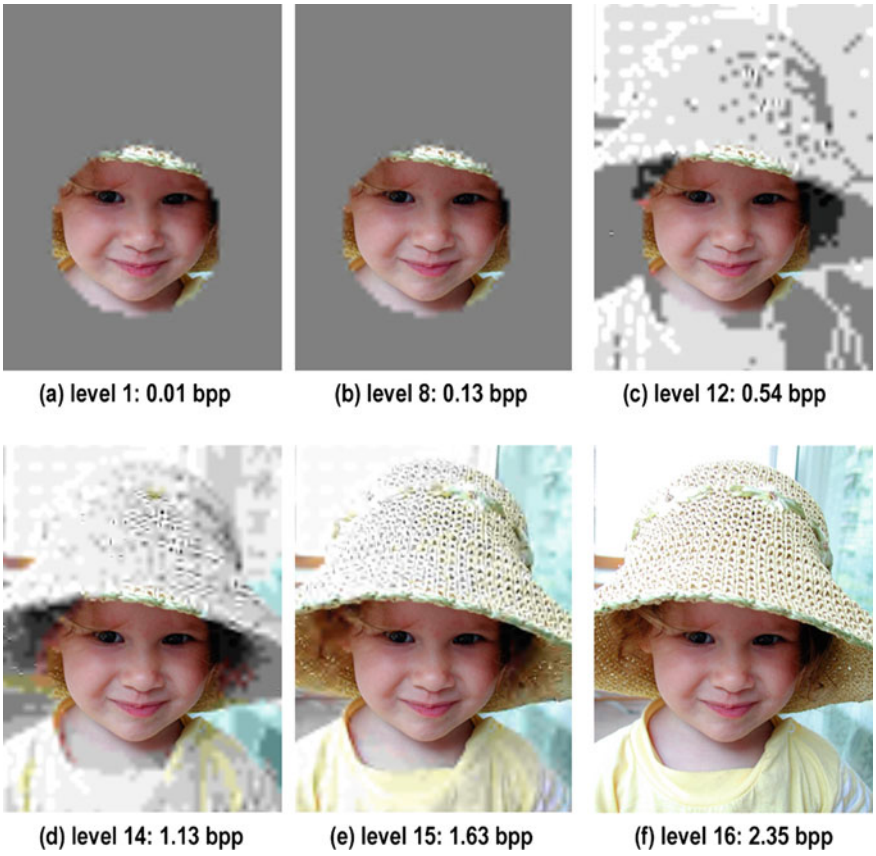

Fig. 2.104 Progressive image decoding of a maxshift mode ROI encoded image

the reconstruction that corresponds to 0.25 bpp marks the beginning of reconstructions with non-noticeable distortions, whereas at 2 bpp there is virtually no visible difference from the original.

In a typical example of ROI coding, Fig. 2.104 depicts the progressive decoding of an image, which has been encoded with layer (quality) priority and a ROI. The image was encoded using 20 quality layers and a circular ROI mask, while the coding strategy for the ROI was the *maxshift mode*. As shown in Fig. 2.104c, in which pixels of the background begin to be reconstructed, the reconstruction of the ROI has already completed. The uniform gray regions in the first two figures correspond to regions for which there were no available data (yet) for a reconstruction.

In a different example, priority was given to the progressiveness in *Position* and color *Component*, using a tiling into twelve image tiles, as shown in Fig. 2.105. In this example, the image with the same tiling was used to produce two compressed bitstreams, one with PCRL and the other with CPRL progressiveness. As shown

Fig. 2.105 Random image tiling into 12 tiles

in Fig. 2.106, the result of a gradual decoding is significantly different for the two progressive schemes, even when the samples that are considered correspond to the same compression rate (bpp). Again, the uniform gray regions in the images in Fig. 2.106 correspond to regions with no available data for reconstruction.

Numerous studies have been conducted to estimate the gain in compression efficiency attained by using the JPEG2000 standard (Skodras et al. 2001; Christopoulos et al. 2000b; Santa-Cruz and Ebrahimi 2000a, b; Santa-Cruz et al. 2000). Estimates tend towards a compression rate improvement of about 30 % compared to the conventional JPEG for the same objective image quality (Signal to Noise Ration (SNR)), while it is worth noting that the subjective quality is clearly better than that obtained by the conventional JPEG, since the distortion caused by the use of the DWT are less noticeable by human observers since they are of an entirely different nature compared to those caused by the DCT, which acts on small image blocks and introduces sharp and easily noticeable discontinuities at the block boundaries.

Figure 2.107 shows an example comparison between the two standards where the objective quality is compared for the same compression ratio. The supremacy in the image quality of JPEG2000 is apparent, although the PSNR estimate reports only a 4

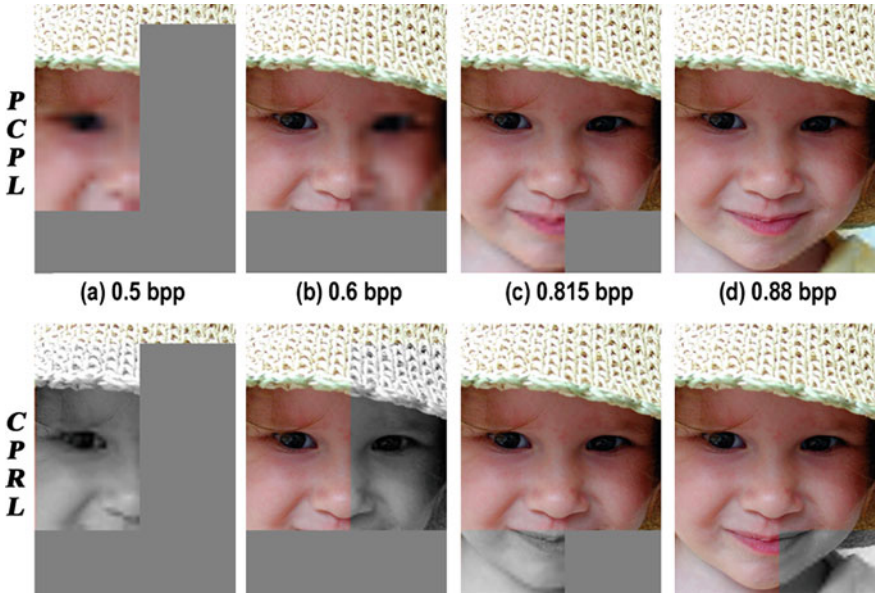

Fig. 2.106 Progressive decoding of two compressed bitstreams with different progressiveness priorities

Fig. 2.107 Quality comparison between JPEG and JPEG2000 for the same compression rate

Table 2.18 Lossless compression ratios for various compression standards

Image	J2K	JPEG-LS	L-JPEG	PNG
Aerial2	1.47	1.51	1.43	1.48
Bike	1.77	1.84	1.61	1.66
Café	1.49	1.57	1.36	1.44
Chart	2.60	2.82	2.00	2.41
Cmpnd1	3.77	6.44	3.23	6.02
Target	3.76	3.66	2.59	8.70
Us	2.63	3.04	2.41	2.94
Average	*2.50*	*2.98*	*2.09*	*3.52*

dBs difference. The quantization is clearly visible in JPEG, whereas the JPEG2000 image is nearly indistinguishable from the original.

Table 2.18 summarizes the lossless compression performance results in an experiment using seven standard graylevel test images with distinct characteristics (shown in Fig. 2.108). In (Skodras et al. 2001), one of the many works on the presentation of the JPEG2000 standard, there is an extensive reference to the comparative results of the compression method in comparison with other known standards.

2.5.4 Progressive Transmission on the Web

This section presents, as an educative example on the functionalities offered by JPEG2000, the development of a Web application aiming at the progressive transmission of large images stored in multimedia databases.[23] The technology employed is based on the classic *client-server architecture* and *TCP/IP communication* through *sockets*, in which at the request of a client, the image database server can easily respond with partial image data deriving from single image files. In this method, a significant reduction in the size of the data transmitted through the network is accomplished, since the server transmits only the portion of the encoded image that corresponds to the image data that are of the client's interest. In addition, since the client employs caching mechanisms, there is a possibility of using existing data in the client (previously transmitted), so that additional network traffic reductions are possible. This was among the first works towards the development of systems supporting progressive transmission of images using the JPEG2000 standard and Web browser plug-ins technology (Politou et al. 2004). Although since that work other

[23]It is noted that this section is provided here purely to illustrate the potential in adopting the JPEG2000 coding strategy and to highlight an example of successful engineering. The system that is described was implemented in a *pre-HTML5 era*, so the need for all those complementing technologies for client-server communication was imperative. The value of this section is purely illustrative and educative.

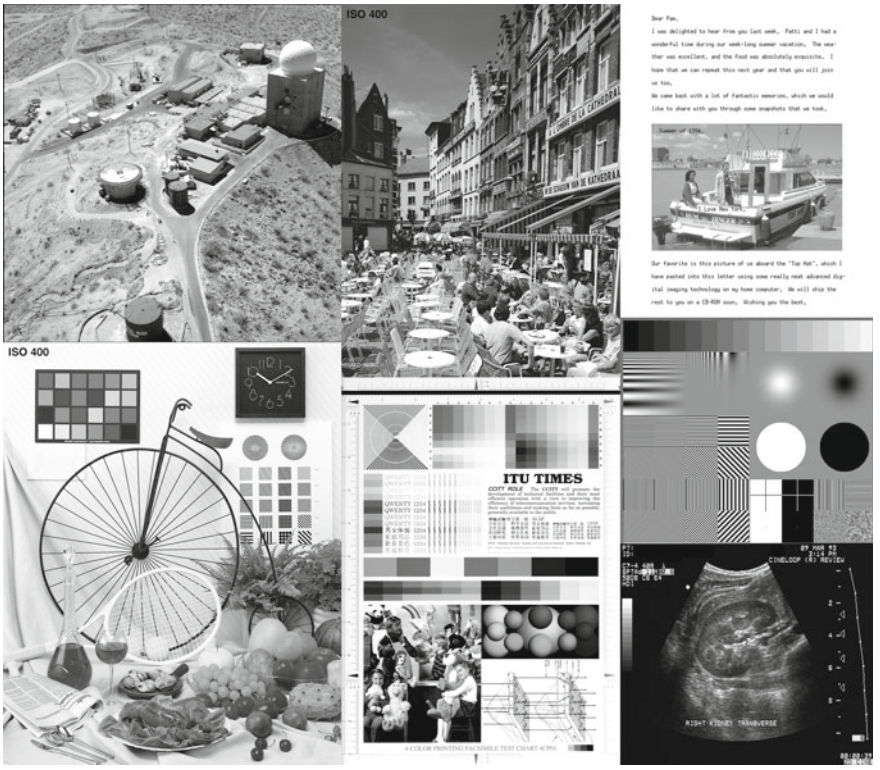

Fig. 2.108 Graylevel image set used for a comparative lossless compression study

methods and approaches have already been proposed, they either did not rely on a common interface such as a Web browsers (Deshpande and Zeng 2001), or they were not based on a generally accepted communication protocol (Taubman 2002a).

One of the serious technical challenges on the Internet—that is constantly under study and development—is the limited bandwidth, the limitations on data transmission speed. The challenge of having a limited bandwidth, in conjunction with large storage requirements were, after all, the main reasons for the development of different compression methods. A typical example of a compression method with wide acceptance and dissemination on the Internet is JPEG (Pennebaker and Mitchell 1993). As known, using JPEG (for example, by running the classic cjpeg codec Independent JPEG Group 2000) with a quality factor of 15 on a typical digital photograph of say 24 MPixels (6,000 × 4,000 pixels), the transmission time may be significantly reduced from minutes to some seconds, maintaining a satisfactory digital image quality (around 30 dB PSNR). The emergence of JPEG2000 with a better compression efficiency and a set of enhanced functionalities widens the possibilities for higher quality applications.

One typical application of multimedia databases, are the databases in the cultural heritage domain, in which the content includes all kinds of digital media. Among those media, one with a significant importance and large volumes is the image. An image provides information about the shape, texture, color and other details, properties that often can not be described (at least easily) by simple text. Common problem in cultural databases is the difficulty in managing and publishing of information that is stored in large images. Significant work on this subject has been reported by many research groups, and has been a subject of research and development activities of the author of this treatise for many years.

Among the various works one particular will be the subject of this section, the "Ark of Refugee Heirloom" (Politou et al. 2002), which recorded the cultural identity of a specific part of a community. Content from that database has been used for a pilot implementation of a large volume image data transmission. The goal was to explore and propose a new, interactive and economic (in terms of data transmission), way of accessing multimedia data. The basic idea was to implement a progressive image transmission method for the Web, and to ensure an efficient way for data exploration. After extensive experimentation on various compression methods and the way an average user navigates in cultural content online, this investigation resulted in the following scenario for an efficient interactive environment,

- A visitor that enters the home page receives a list of small versions of all (or a part of) the images in the database, the 'thumbnails'
- The visitor selects one of the thumbnail images for further study and receives a medium sized image
- If the visitor wishes to examine the image in full resolution, the image is transmitted in two steps using quality refinement
- The transmission in all steps is differential, so as to transmit only the additional information that is required to achieve the requested quality and resolution at a specific step.
- The connection between the user (client) and the server is controlled by sockets, while the JPEG2000 decoding and presentation of the images is controlled by a web browser plug-in

In this scenario, each time the user requests additional information either to increase the resolution or the quality of an image, the required information is extracted and transmitted from the same single image file. This implies that the image encoding should have been done in such a way as to ensure the possibility of extracting various resolutions from the same file.

In order to proceed further with the description of the implementation, a description of the main pieces that complete the overall puzzle will be defined and put together. These *pieces* are the *syntax of the JPEG2000 file format*, the *client-server technology* and the *browser plug-ins*. The *puzzle* is a system to provide efficient progressive transmission under a Web-based image database navigation scenario.

First, the syntax in the JPEG2000 encoded file should be examined. This syntax defines all the basic properties of the encoded image, such as dimensions, the size of image tiles, the number and the rates of sampling of the spectral components, as

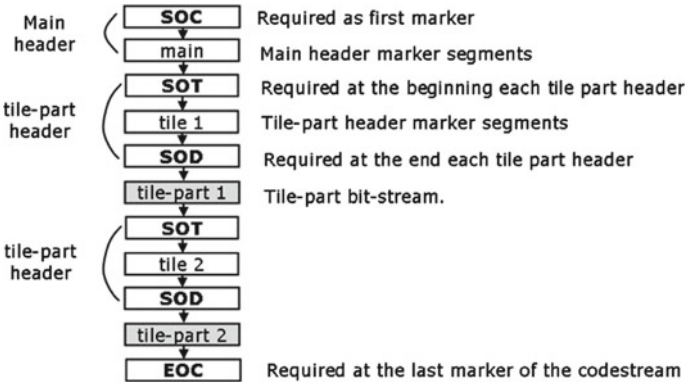

Fig. 2.109 Basic file structure of a JPEG2000 encoded image

well as the parameters of quantization and coding, the size of codeblocks and the transform that is applied. In addition, it includes information about the number of quality layers, resolutions and progressiveness priority, ROI encoding, as well as error correction settings. Most parameters can be selected at a tile level (ISO-IEC 2000a; Moccagata et al. 2000). In the simplest case, a file that represents an encoded JPEG2000 image has a structure that starts with a main header that is followed by a series of encoded information blocks as graphically shown in Fig. 2.109. The extension of this file, according to the standard (ISO-IEC 2000a) is jpc (**jp**eg **c**odestream). The main header contains global information necessary for the decoding of the whole file. Each image tile in the codestream consists of a header, which contains informations relating individually to the tile to which they belong. Each group of packets includes a sequence of encoded data (ISO-IEC 2000a; Moccagata et al. 2000). A progressive encoding would allow to be able to improve the quality, resolution, size and colors as more and more data are gradually being decoded from an encoded file that arrives at the receiver. The type or priority in progression is determined by the way in which information packets are arranged in the codestream. It is possible to determine progressiveness both at a global and at a tile-based level, which is expressed by the way tile-parts are being arranged. Each part has its own header (ISO-IEC 2000a; Moccagata et al. 2000). As mentioned in the previous paragraphs, the standard defined five different priorities for progressiveness.

In addition, the client-server technology is based on creating sockets to achieve communication between two remote computers. A socket operates as a phone. It is the terminal of a bidirectional communication channel. By connecting two sockets it is possible to transmit data to processes, even when operating on different computers, in a manner similar to the speech transmission in phones. In general, a software that is written to use the Transport Control Protocol (TCP) is developed using the client-server model, in which when two connected devices use TCP for data exchange, one of them assumes the role of the client and the other the role of the server. The application on the client is the one which starts what is called an *active open*. It creates

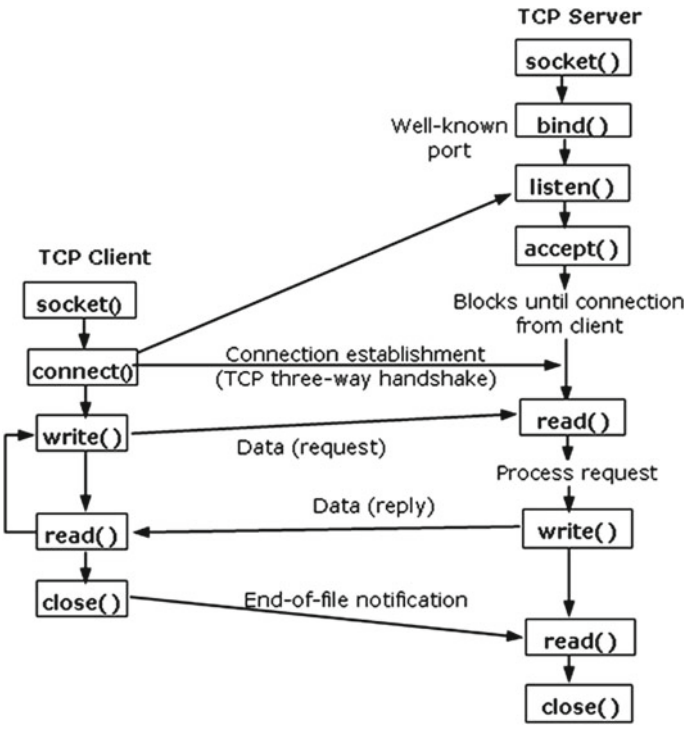

Fig. 2.110 Processes in a socket connection

a socket and tries to connect to a server. At the other end, the server application creates a socket that 'listens', waits, for incoming connections from clients, by performing what is called a *passive open*, as shown in Fig. 2.110. When the client initiates a connection, the server is informed that a process is trying to connect with it. By accepting the connection, the server completes the creation of a virtual circuit, which is nothing more than a logical communication path between two programs. It is important to note that the attempt to create a connection always creates a new socket; the original socket remains unchanged so that the server remains open to listen for new connections. When the server does not wish or is not required to await new connections then it may terminate the original passive socket (Comer and Stevens 2000).

Finally, a plug-in (a complementary software library) is a separate semi-autonomous part of software that behaves as part of a software, which it complements with new features and functionalities. In particular, a plug-in for a Web browser is a library that is activated and only works within a Web browser on the Web. The way the browser interacts with the plug-in is described by a set of programming tools, called the plug-in Application Programming Interface (API) (Netscape 1997). With the plug-in API it is possible to create dynamic plug-ins, which

- register one or more MIME types
- draw on a particular position on the canvas of the Web browser window
- receive and manage user input (eg. keyboard and mouse)
- receive and send data from/to the Web via URLs
- add hyperlinks or links, in general, to point to URLs

When a user opens a Web page that contains data types which activate a plug-in, then the Web browser responds by performing a number of predefined actions

- it searches for a registered plug-in that matches the specific requested MIME type
- it loads the dynamic plug-in into memory
- it activates the plug-in
- it creates an instance (copy) of the plug-in in memory
- it removes an instance of the plug-in from memory if not required
- it disables and removes the plug-in from memory when the removal of the last instance is requested

With the pieces of the puzzle defined, the presentation of the implementation begins with a *server application* and any additional *tools needed for image preparation* and concludes with a *client application*. The server application is essentially a service that runs on a Web server and has access to the images to be transmitted. The images were previously encoded using lossless JPEG2000, in an appropriate manner so as to ensure the progressive transmission in accordance with the predefined interactive navigation scenario. A special application was developed to properly prepare the images. On the other side of the connection, the client application is a browser plug-in, installed on the user's computer in order to enable processing of the user interactions and to decode and display the images requested by the user.

To be able to select multiple meaningful segments from a single encoded image file, a specific way of coding must be adopted. The data of the encoded bitstream must be arranged in code segments, so that the sequential 'addition' of arriving code parts to produce an increasing resolution (image size) as shown in Fig. 2.111. The shading in the various parts of this representation denotes the grouping of data that should be imposed for a progressive by resolution coding. Under the specific implementation scenario, and after a study on the requirements and practices in online multimedia databases, specific resolutions of the images were selected in this project to, somehow, optimize the experience of the users (based mainly on a better use of the network bandwidth). The final selection of the intermediate resolutions was based on empirical and experimental data from previous related activities (Cultural & Educational Technology Institute 2000a; Democritus University of Thrace 2000),

- 64×64 pixels for the lowest resolution (thumbnail size)
- 256×256 pixels for the 'medium' resolution (preview size)
- the full-sized original image

To achieve encoding of images so that they contain these specific resolutions, the number of levels of wavelet decompositions should be defined in advance. Specifically, to achieve the lowest resolution of 64×64 pixels, the number of wavelet decompositions n is estimated as

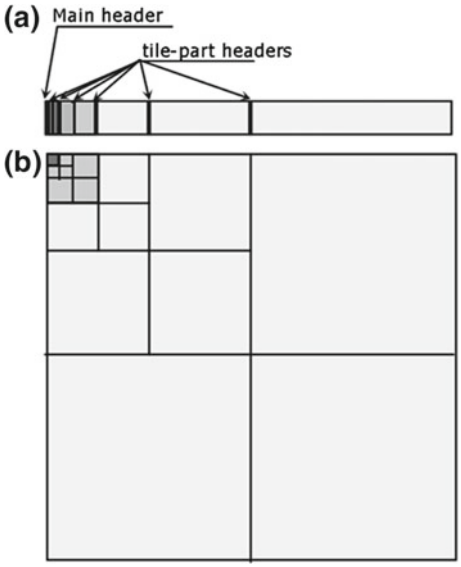

Fig. 2.111 Representation of code segments in a JPEG2000 compression bitstream: **a** the final bitstream and **b** the corresponding parts in the transform domain

$$\left.\begin{array}{l} n_w \leq \log_2 \frac{w}{w'} \\ n_h \leq \log_2 \frac{h}{h'} \end{array}\right\} \Rightarrow n = \lceil max\{n_w, n_h\} \rceil \tag{2.189}$$

where w and h are the width and height of the original image, w' and h' is the required width and height respectively, n_w and n_h the number of divisions by two in width and height for the production of the desired size and $\lceil\ \rceil$ is the ceiling operator. In this particular case the value for both w' and h' was 64. This way the images are encoded with $n+1$ levels of resolutions (there will be $n+1$ Start Of Tile (SOT) markers in the code-stream), ensuring that the lower resolution is of 64×64 pixels at maximum. A representation of the structure is shown in Fig. 2.112. Encoding was performed via the *kakadu software* (Taubman 2000a), the code of which was incorporated into

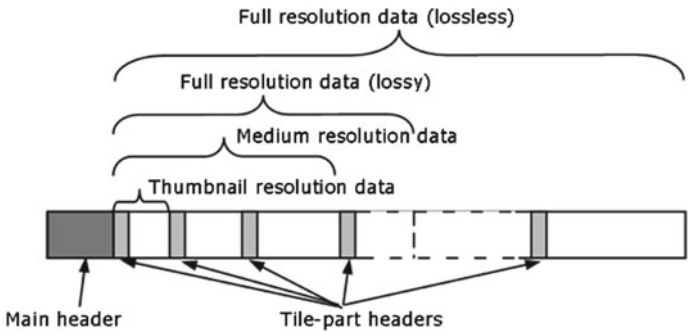

Fig. 2.112 Representation of the JPEG2000 bitstream for images with a predefined resolutions

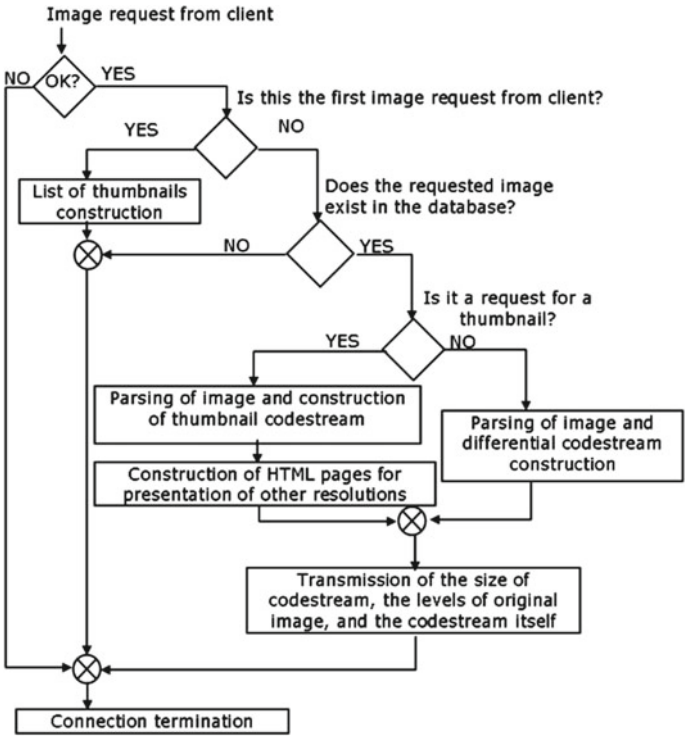

Fig. 2.113 Flowchart of a server response for progressive differential information transmission

the application created. The corresponding command line to compress the images according to the scenario adopted is

```
kdu_compress -rate - Corder=RLCP ORGparts=R ORGgen_plt=yes Clayers=3
     Creversible=yes Clevels=n -i input_image -i output_code-stream
```

After imposing this encoding on the images of the database, the images are ready to be used by the server to achieve progressive transmission on the Web according to the application scenario.

As the server should be able to handle multiple and concurrent client connections and requests, the server application was designed to operate in *multithreaded mode*. It was equipped with the ability to send images of any resolution by analyzing the syntax in the file of each encoded image. At the final stage of each processing cycle, the application creates the required HTML pages to be delivered to the client. The way in which the server application responds to a client request is shown in Fig. 2.113. Each time a client requests the lowest resolution of an image, the application sends the corresponding code-stream parts including the main header. In any other case (the client has already received one of the resolutions an the image) in which the

client requests further image information, then the application sends the differential information by extracting the appropriate parts of the code-stream without resending previously sent information. It is in the client's application responsibility to manage the differential information for proper reconstruction of the requested image. To automate the process, the HTML pages in which the images are being presented, are created at the server side, always one step ahead of the corresponding request of the connected client. The process of creating low-resolution images on the server is based on marker operators and syntax control on the code-stream of an image. Whenever reading data from the main header of the file, the data are copied to a temporary storage in memory while replacing the values for the markers *XRsiz*, *YRsiz SIZ* (0xFF51) and the wavelet transform marker *COD* (0xFF52), with new values for proper decoding of the image corresponding to the requested resolution. Whenever reading data that correspond to the content of a tile, then they are copied in a new temporary file. At the end of the process, all temporary data are synthesized into a single stream and are sent to the client.

The client application is essentially a Web browser plug-in. The plug-in receives the encoded data from the server and decodes and presents the data to the user within the Web browser. The syntax of the HTML *EMBED* tag, with which objects are embedded to integrate a particular plug-in to an HTML page, is being employed

```
<EMBED
src="name.jpc"
onclick="(javascript routines)"
type="MIME type"
width="width"
height="height"
host="Server IP"
size="Requested resolution">
```

where *src*, *onclick*, *type*, *width* and *height* are parameters and events that determine typical properties of *EMBED* tags, while *host* and *size* are parameters of the plug-in,

- the parameter *src* contains the image file name being requested
- the *onclick* event is a general HTML event that determines how the Web browser responds to the use of the mouse buttons within a Web page
- the parameter *type* specifies the MIME type of the data, determined during the development of a plug-in and in the specific application it was defined as *application/x-Netscape-jpc*[24]
- the parameters *width* and *height* define the dimensions of the display area (within the Web browser), which the plug-in will span, and must be greater than or equal to the image to be displayed

[24]Remember this was an implementation in an early pre-HTML5 era.

- the *host* parameter specifies the IP address of the server with which a connection is being made for receiving data
- the *size* parameter specifies the desired resolution of the image for display

Once the Web browser detects the MIME type in the EMBED tag, the corresponding plug-in is called and control passes to the plug-in. Specifically, when a user opens a Web browser, all available plug-ins are loaded into memory. The appropriate plug-in is called whenever the Web browser faces content that requests its activation. If there is no other Web browser windows with the selected plug-in activated, then the plug-in is activated and binds global memory for the management of shared data. Then a new instance (copy) of the plug-in is created in the part of the memory controlled by the Web browser, to which the input parameters are eventually transferred. These parameters are no other than the ones specified in the *EMBED* tag that triggered the plug-in. By using these parameters, the Web browser binds an area on the canvas of the current window with the output of the plug-in. The dimensions of this area are defined by the parameters *width* and *height*. Parameters *src*, *host* and *size* are used subsequently according to the requests to the server. After the process of the received data, the client decodes and presents the data in the reserved area of the plug-in. When the user chooses to close the page (the window), the Web browser removes the active instance of the plug-in from memory, but does not completely releases the memory of the global data. This is only done when all Web browser windows with the active specific plug-in are closed. The entire process is represented graphically in Fig. 2.114.

According to the adopted case study scenario, the client-server connection starts with a user verification and continues with the client triggering the server to create the thumbnail images Web page. The page is sent to the client and control passes to the client, waiting user input. When the user requests information for an image then a series of processes is activated, which is shown in Fig. 2.115.

- if the image requested by the user already exists on the client (cache memory of the client Web browser) then no request is sent to the server and the client simply decodes and presents the stored data
- if the image requested by the user has lower resolutions in the cache memory of the client's Web browser, then the client sends a request to the server for differential transmission of information and the data obtained are added to those already existing for the creation of the next higher resolution; at the same time, the main header, which already exists on the client is updated to reflect the change in the resolution
- if the image is not on the client at any resolution (it is the first time the user requests data about this image), the lowest resolution is being received and presented

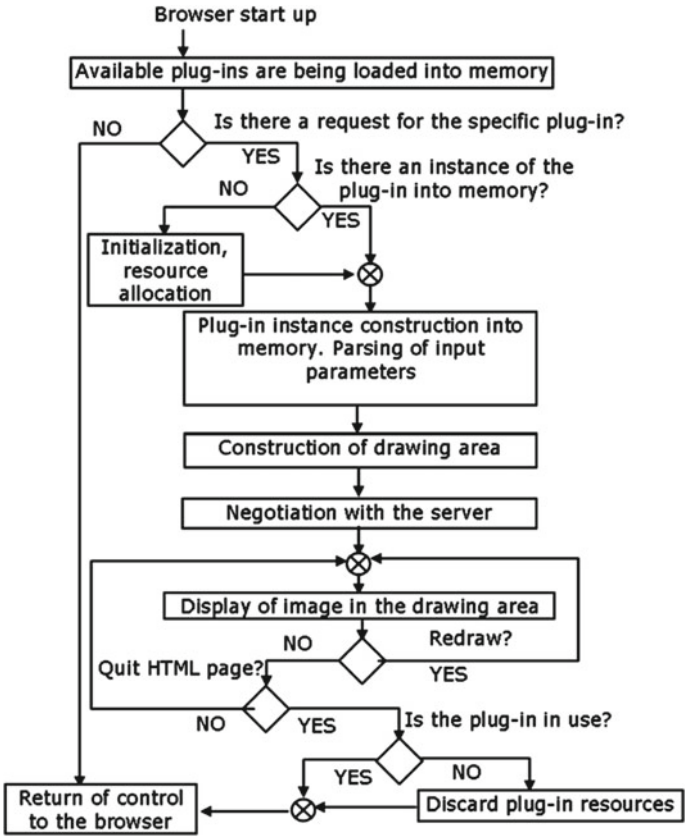

Fig. 2.114 Flowgram of the operation of a plug-in on a Web browser client

The described system has been put to the test by using a real multimedia database of cultural heritage, the one created for *The Ark of Refugee Heirloom* project (Cultural & Educational Technology Institute 2000a). A presentation of the results of this case study are in (Cultural & Educational Technology Institute 2000b) and are briefly presented here. A first step when a user is visiting *the Ark* for the first time, is the user authentication as shown in Fig. 2.116a. At the same time the server prepares the introductory page to present the list of images in low resolution (thumbnails). After the user authentication, the prepared page of thumbnails is sent from the server and displayed to the client as shown in Fig. 2.116b. If the user chooses to request further information regarding one of the images, the browser plug-in receives the additional information from the server and decodes and displays the medium resolution, accompanied by a link to the higher resolution as shown in Fig. 2.116c. If the

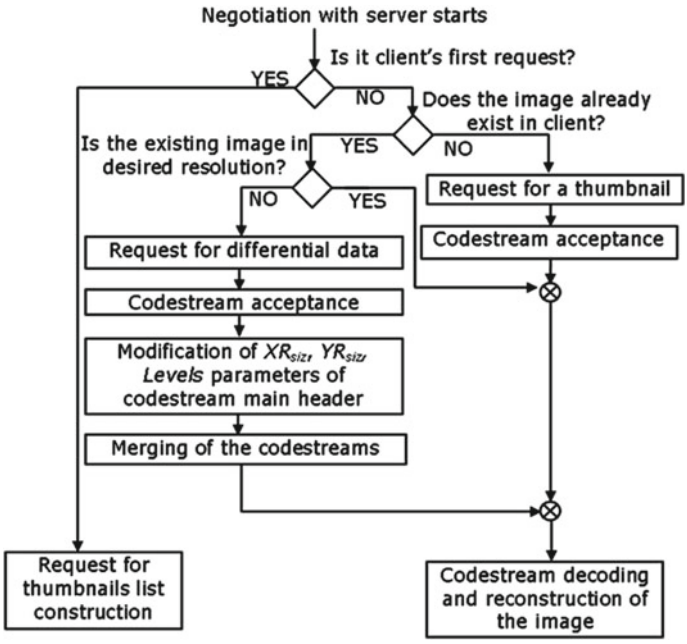

Fig. 2.115 Flowgram of the processes at the Web browser plug-in

user requests for even higher resolution, additional information that corresponds to the maximum resolution and a reduced quality is received and presented, as shown in Fig. 2.116d, providing, at the same time, a link to the full-resolution. At the final step, the user may request the full resolution and quality image (original image), as shown in Fig. 2.116e.

Table 2.19 provides comparative compression and transmission results for the image 'watch' (Fig. 2.116), which is a characteristic image from the database of the Ark in all resolutions.[25] Of interest in these results is the one that corresponds to the transmission of the maximum-resolution-low-quality (at 1 bpp), in which there is substantial differential data compression and transmission gain, while with the received image quality the user can have a *satisfactory* representation of the data. It is also worth noting that the JPEG2000 compression experiments at 1 bpp throughout the whole image set of the database resulted in an average quality of decoded image of about 35 dB PSNR, that backs the claim that it was a good selection for that progression step.

[25]Transmission times are reported for a noiseless transmission channel in a guaranteed constant 1 Mbps bandwidth.

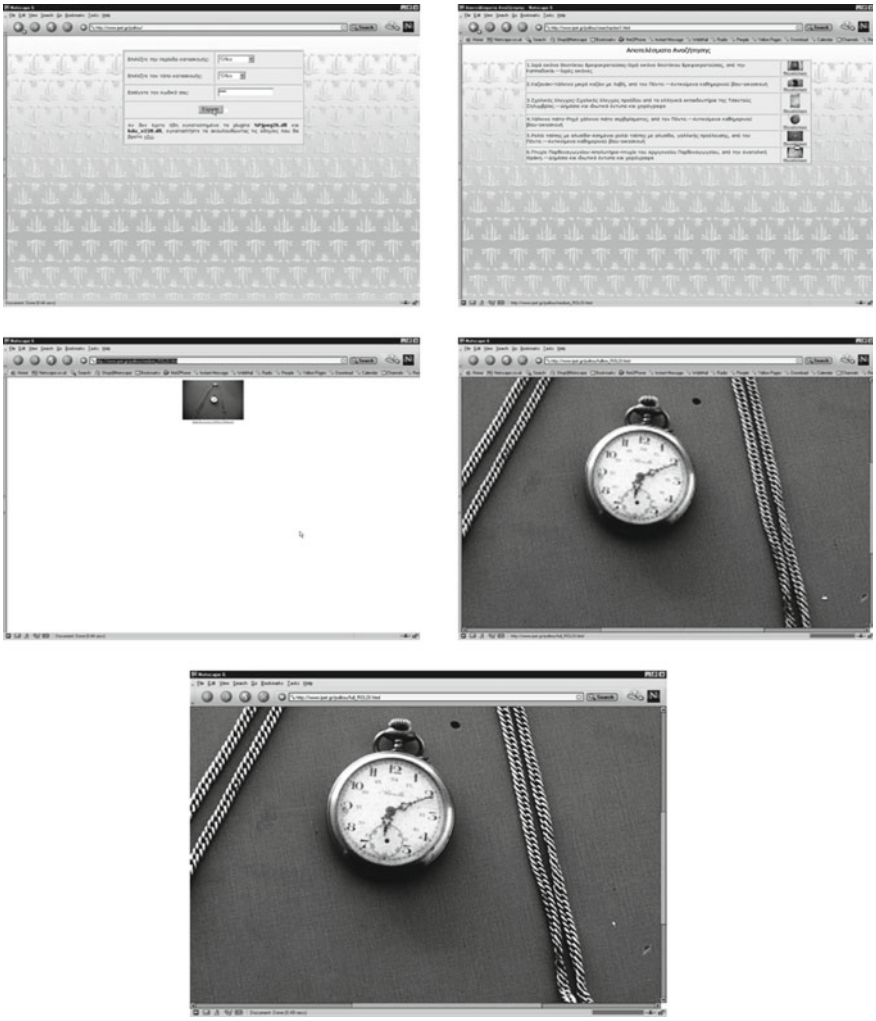

Fig. 2.116 Screenshots of the step-by-step interaction within the plug-in on the client

It is apparent that the JPEG2000 compression leads to image databases in which images can be compressed into multiple resolutions and quality layers within a single file each. These files consist of embedded data that can be used for the recovery and reconstruction of images in multiple ways. Some of the relevant features include progressive presentation in resolution and quality, zooming in a specific area, use of a dynamic Region Of Interest (ROI) and copyright control and management.

Table 2.19 Comparative compression and transmission results for four resolutions of the image 'Watch'

Image size (pixels)	Uncompressed data (Kbytes)	Compressed data (Kbytes)	Data rate (bpp)	Transmission data (Kbytes)	Data transmission	Compression ratio	Transmission times (s)
Thumbnail 43×28	3.5	2.27	16.21	2.27	15.57	1.54:1	0.02
Medium 169×110	54.5	26.59	12.20	24.32	10.71	2.05:1	0.19
Lossy-full 2696×1756	13,870	578	1.04	553.68	0.96	24.00:1	4.33
Lossless 2696×1756	13,870	4,152.29	7.48	3,598.61	6.23	3.34:1	28.11

References

Abramson, N. (1963). *Information theory and coding*. New York: McGraw-Hill.

Albanesi, M., & Bertoluzza, S. (1995). Human vision model and wavelets for high-quality image compression. In *Proceedings of the 5th International Conference in Image Processing and its Applications* (Vol. 410, pp. 311–315).

Bracewell, R. M. (1983). Discrete Hartley transform. *Journal of the Optical Society of America*, *73*(12), 1832–1835.

Christopoulos, C., Ebrahimi, T., & Lee, S. U. (2002). JPEG2000 Special Issue. *Elsevier Signal Processing: Image Communication17*.

Christopoulos, C., Askelof, J., & Larsson, M. (2000a). Efficient methods for encoding regions of interest in the upcoming JPEG2000 still image compression standard. *IEEE Signal Processing Letters*, *7*(9), 247–249.

Christopoulos, C., Skodras, A., & Ebrahimi, T. (2000b). The JPEG 2000 still image coding system: An overview. *IEEE Transactions on Consumer Electronics*, *46*(4), 1103–1127.

Comer, D. E., & Stevens, D. L. (2000). *Internetworking with TCP/IP: Client-Server programming and applications*. Prentice Hall. ASIN: B00LKKVXQ4.

Cover, T. M., & Thomas, J. A. (2006). *Elements of information theory* (2nd edn). John Wiley & Sons, ISBN: 978-0-471-24195-9.

Cultural & Educational Technology Institute, Greece. (2000a). *Ark of Refugee Heirloom—A cultural heritage database—Online*.

Cultural & Educational Technology Institute, Greece. (2000b). *Ark of Refugee Heirloom JPEG2000 prototype*.

Democritus University of Thrace, Greece. (2000). *Thracian Electronic Thesaurus*.

Deshpande, S., & Zeng, W. (2001). Scalable streaming of JPEG2000 images using hypertext transfer protocol. In *Proceedings of ACM* (pp. 72–281).

Elias, P. (1975). Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, *21*(2), 194–203.

Fano, R. M. (1949). *The transmission of information*. Technical report. 65. Research Laboratory of Electronics at MIT.

Fano, R. M. (1952). *Class notes for Transmission of Information, course 6.574*. Technical report. Cambridge, MA: MIT

Fisher, R. A. (1922). On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society, London A*, *222*, 309–368.

Golomb, S. (1966). Run-length encodings. *IEEE Transactions on Information Theory*, *12*, 399–401.

Gonzalez, R. C., & Woods, R. E. (1992). *Digital image processing* (3rd edn). Prentice Hall, ISBN: 978-0201508031.

Gray, R., & Neuhoff, D. (1998). Quantization. *IEEE Transactions on Information Theory*, *44*(6).

Haar, A. (1910). Zur Theorie der orthogonalen Funktionensysteme. *Mathematische Annalen*, *69*(3), 331–371.

Hartley, R. V. L. (1928). Transmission of information. *Bell System Technical Journal*, *7*(3), 535–563.

Hartley, R. V. L. (1942). A more symmetrical Fourier analysis applied to transmission problems. *Proceeding IRE*, *30*, 144–150.

Huffman, D. (1952). A method for the construction of minimum redundancy codes. *Proceedings IRE*, *40*, 1098–1101.

Independent JPEG Group, IJG. (2000). *JPEG reference software*.

ISO-IEC. (2000a). *Information technology— JPEG 2000 image coding system—Part 1: Core coding system, ISO/IEC International Standard 15444–1*. ISO/IEC: Technical report.

ISO-IEC-CCITT. (1993b). *JPEG: Information Technology—Digital compression and coding of continuous-tone still images—requirements and guidelines, ISO/IEC International Standard, CCITT Recommendation T.81*. Technical report. ISO/IEC/CCITT.

ISO-IEC-ITU. (2000). *JBIG2, ISO/CEI International Standard 14492 and ITU-T Recommendation T.88*. Technical report. ISO/IEC/ITU.

Jain, K. A. (1988). *Fundamentals of digital image processing*. New Jersey: Prentice-Hall.

Jones, P., Daly, S., Gaborski, R., & Rabbani, M. (1995). Comparative study of wavelet and DCT decompositions with equivalent quantization and encoding strategies for medical images. In *Proceedings of SPIE* (Vol. 2431, pp. 571–582).

Kang, L. W., & Leou, J. J. (2003). A new error resilient coding scheme for JPEG image transmission based on data embedding and vector quantization. *Proceedings of IEEE International Symposium on Circuits and Systems—ISCAS2003* (Vol. 2, pp. 532–535).

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, *22*, 79–86.

Lehmann, E. L., & Scheffe, H. (1950). Completeness, similar regions and unbiased estimation. *Sankhya*, *10*, 305–340.

Liang, J., & Talluri, R. (1999). Tools for robust image and video coding in JPEG2000 and MPEG-4 standards. In *Proceedings of the SPIE Visual Communications and Image Processing Conference* (Vol. 3653, pp. 40–51).

Linde, Y., Buzo, A., & Gray, R. M. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, *28*(1), 84–95.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137.

McMillan, B. (1956). Two inequalities implied by unique decipherability. *IEEE Transaction of Information Theory, IT-2*, 115–116.

Moccagata, I., Sodagar, S., Liang, J., & Chen, H. (2000). Error resilient coding in JPEG2000 and MPEG-4. *IEEE Journal of Selected Areas in Communications (JSAC)*, *18*(6), 899–914.

Netscape. (1997). *Plug-in guide*.

Ono, F., Kino, S., Yoshida, M., & Kimura, T. (1989). Bi-level image coding with MELCODE—comparison of block type code and arithmetic type code. In *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)* (pp. 255–260).

O'Rourke, T., & Stevenson, R. (1995). Human visual system based wavelet decomposition for image compression. *Journal of Visual Communication and Image Representation*, *6*, 109–131.

Pasco, R. C. (1976). *Source coding algorithms for fast data compression*. Ph.D. thesis, Stanford University.

Pennebaker, W. B., & Mitchell, J. L. (1993). *JPEG still image compression standard*. New York: Springer.

Pennebaker, W. B., Mitchell, J. L., Langdon, G., & Arps, R. B. (1988). An overview of the basic principles of the Q-coder adaptive binary arithmetic coder. *IBM Journal of Research and Development*, *32*(6), 717–726.

Politou, E., Tsevremes, I., Tsompanopoulos, A., Pavlidis, G., Kazakis, A., & Chamzas, C. (2002). Ark of refugee heirloom—A cultural heritage database. In *EVA 2002: Conference of Electronic Imaging and the Visual Arts* (pp. 25–29).

Politou, E. A., Pavlidis, G. P., & Chamzas, C. (2004). JPEG2000 and the dissemination of cultural heritage databases over the Internet. *IEEE Transactions on Image Processing*, *13*(3), 293–301.

Pratt, W. (1991). *Digital image processing* (2nd edn). Wiley-Interscience Publication. ISBN: 0-471-85766-1.

Rabbani, M., & Cruz, D. Santa. (2001). The JPEG2000 still-image compression standard, tutorial session. In *IEEE International Conference on Image Processing—ICIP 2001*.

Rabbani, M., & Jones, P. W. (1991a). *Digital image compression techniques* (Vol. TT7). SPIE-Tutorial Texts in Optical Engineering. ISBN: 978-0819406484.

Rabbani, M., & Joshi, R. (2002). An overview of the JPEG2000 still image compression standard. *Signal Processing: Image Communication*, *17*(1).

Rao, R. M., & Bopardikar, A. S. (1998). *Wavelet transforms: Introduction to theory and applications*. Prentice Hall. ASIN: B01A65JU7W.

Rissanen, J. (1976). Generalized kraft inequality and arithmetic coding of strings. *IBM Journal of Research and Development*.

Rissanen, J. J., & Langdon, G. G. (1979). Arithmetic coding. *IBM Journal of Resources and Development*, *23*(2), 146–162.

Rubin, F. (1979). Arithmetic stream coding using fixed precision registers. *IEEE Transactions on Information Theory*, *25*(6), 672–675.

Said, A. (2004). Introduction to arithmetic coding theory and practice. Technical report. Hewlett-Packard Laboratories Report, HPL-2004-76.

Said, A., & Pearlman, W. A. (1996). A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transaction on Circuits Systems and Video Technology*, *6*(3), 243–250.

Santa-Cruz, D., & Ebrahimi, T. (2000a). An analytical study of JPEG 2000 functionalities. In *Proceedings of IEEE International Conference on Image Processing—ICIP 2000*.

Santa-Cruz, D., & Ebrahimi, T. (2000b). A study of JPEG 2000 still image coding versus other standards. In *Proceedings of X European Signal Processing Conference* (Vol. 2, pp. 673–676).

Santa-Cruz, D., Ebrahimi, T., Askelof, J., Karsson, M., & Christopoulos, C. A. (2000). JPEG2000 still image coding versus other standards. In *Proceedings of SPIE, 45th annual meeting, Applications of Digital Image Processing XXIII* (Vol. 4115, pp. 446–454).

Sayood, K. (1996). *Introduction to data compression*. Morgan Kaufmann. ISBN: 978-1558603462.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell Systems Technology Journal*, *27*(379–423), 623–656.

Shapiro, J. M. (1993). Embedded image coding using zero trees of wavelet coefficients. *IEEE Transactions on Signal Processing*, *41*(12), 3445–3462.

Skodras, A., Christopoulos, C., & Ebrahimi, T. (2001). The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine*, 36–58.

Sullivan, G. (1996). Efficient scalar quantization of exponential and Laplacian variables. *IEEE Transactions of Information Theory*, *42*(5), 1365–1374.

Tanaka, H., & Leon-Garcia, A. (1982). Efficient run-length encodings. *IEEE Transactions on Information Theory*, *28*(November), 880–890.

Taubman, D. S. (2000a). *Kakadu Software—A comprehensive framework for JPEG2000*.

Taubman, D. S. (2000b). High performance scalable image compression with EBCOT. *IEEE Transaction on Image Processing*, *9*(7), 1158–1170.

Taubman, D. S. (2002a). Remote browsing of JPEG2000 images. In *IEEE International conference on Image Processing—ICIP2002* (pp. 22–25).

Taubman, D. S., & Marcellin, M. W. (2002b). *JPEG2000 image compression fundamentals, standards and practice*. Kluwer Academic Publishers. ASIN: B011DB6NGY.

Taubman, D. S., Ordentlich, E., Weinberger, M. J., & Seroussi, G. (2002c). Embedded block coding in JPEG2000. *Elsevier Signal Processing: Image Communication*, *17*(1), 49–72.

Wallace, G. (1991). The JPEG still picture compression standard. *Communications of the ACM*, *34*(4), 30–44.

Watson, A. B., & Poirson, A. (1986). Separable two-dimensional discrete Hartley transform. *Journal of the Optical Society of America A*, *3*(12), 2001–2004.

Watson, A. B., Yang, G. Y., Solomon, J. A., & Villasenor, J. (1997). Visibility of wavelet quantization noise. *IEEE Transactions on Image Processing*, *6*(8), 1164–1175.

Witten, I. H., Neal, R. M., & Cleary, K. G. (1987). Arithmetic coding for data compression. *Communications of the ACM*, *30*(6), 520–540.

Ziv, J., & Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, *24*, 530–536.

http://www.springer.com/978-981-10-2829-8

Mixed Raster Content
Segmentation, Compression, Transmission
Pavlidis, G.
2017, XXVIII, 354 p. 260 illus., 103 illus. in color.,
Hardcover
ISBN: 978-981-10-2829-8