CHAPTER

Compression 10

Reducing a liter of orange juice to a few grams of concentrated powder is what lossy compression is about. The taste of the restored beverage is similar to the taste of orange juice but has often lost some subtlety. In this book we are more interested in sounds and images, but we face the same trade-off between quality and compression. Saving money for data storage and improving transmissions through channels with limited bandwidth are major compression applications.

A transform coder decomposes a signal in an orthogonal basis and quantizes the decomposition coefficients. The distortion of the restored signal is minimized by optimizing the quantization, the basis, and the bit allocation. The basic information theory necessary for understanding quantization properties is introduced. Distortion rate theory is first studied at high bit rates, in a Bayes framework, where signals are realizations of a random vector that has a probability distribution that is known a priori. This applies to audio coding, where signals are often modeled with Gaussian processes.

High signal-compression factors are obtained in sparse representations, where few nonzero coefficients are kept. Most of the bits are devoted to code the geometry of these nonzero coefficients, and the distortion is dominated by the resulting nonlinear approximation term. Wavelet image transform codes illustrate these properties. JPEG and JPEG-2000 image-compression standards are described.

10.1 TRANSFORM CODING

When production models are available, as in speech signals, compression algorithms can code parameters that produce a signal approximation. When no such model is available, as in general audio signals or images, then transform codes provide efficient compression algorithms with sparse representations in orthonormal bases. Section 10.1.1 reviews different types of compression algorithms and Section 10.1.2 concentrates on transform codes.

10.1.1 Compression State of the Art

Speech

Speech coding is used for telephony, where it may be of limited quality but good intelligibility, and for higher-quality teleconferencing. Telephone speech is limited to the frequency band of 200–3400 Hz and is sampled at 8 kHz. A pulse code modulation (PCM), which quantizes each sample on 8 bits, produces a code with 64 kb/s ($64\,10^3$ bits per second). This can be considerably reduced by removing some of the speech redundancy.

The production of speech signals is well understood. Model-based analysis–synthesis codes give intelligible speech at 2 kb/s. This is widely used for defense telecommunications [316, 460]. Digital cellular telephony uses 8 kb/s or less to reproduce more natural voices. Linear predictive codes (LPCs) restore speech signals by filtering white noise or a pulse train with linear filters defined by parameters that are estimated and coded. For higher bit rates, the quality of LPC speech production is enhanced by exciting the linear filters with waveforms chosen from a larger family. These code-excited linear prediction (CELP) codes provide nearly perfect telephone quality at 16 kb/s.

Audio

Audio signals include speech but also music and all types of sounds. On a compact disc, the audio signal is limited to a maximum frequency of 20 kHz. It is sampled at 44.1 kHz and each sample is coded on 16 bits. The bit rate of the resulting PCM code is 706 kb/s. For compact discs and digital audio tapes, signals must be coded with hardly any noticeable distortion. This is also true for multimedia CD-ROM and digital television sounds.

No models are available for general audio signals. At present, the best compression is achieved by transform coders that decompose the signal in a local time-frequency basis. To reduce perceived distortion, perceptual coders [317] adapt the quantization of time-frequency coefficients to our hearing sensitivity. Compact disc–quality sounds are restored with 128 kb/s; nearly perfect audio signals are obtained with 64 kb/s.

Images

A gray-level image typically has 512×512 pixels, each coded with 8 bits. Like audio signals, images include many types of structures that are difficult to model. Currently, the best image-compression algorithms are the JPEG and JPEG-2000 compression standards, which are transform codes in cosine bases and wavelet bases.

The efficiency of these bases comes from their ability to construct precise nonlinear image approximations with few nonzero vectors. With fewer than 1 bit/pixel, visually perfect images are reconstructed. At 0.25 bit/pixel, the image remains of good quality.

Video

Applications of digital video range from low-quality videophones, teleconferencing, and Internet video browsing, to high-resolution television. The most effective compression algorithms remove time redundancy with a motion compensation. Local image displacements are measured from one frame to the next, and are coded as motion vectors. Each frame is predicted from a previous one by compensating for the motion. An error image is calculated and compressed with a transform code. The MPEG video-compression standards are based on such motion compensation [344] with a JPEG-type compression of prediction error images.

Standard-definition television (SDTV) format has interlaced images of 720×426 (NTSC) or 720×576 pixels (PAL) with, respectively, 50 and 60 images per second. MPEG-2 codes these images with typically 5 Mb/s. Internet videos are often smaller images of 320×240 pixels with typically 20 or 30 images per second, and are often coded with 200 to 300 kb/s for real-time browsing. The full high-definition television (HDTV) format corresponds to images of 1920×1080 pixels. MPEG-2 codes these images with 12 to 24 Mb/s. With MPEG-4, the bit rate goes down to 12 Mb/s or less.

10.1.2 Compression in Orthonormal Bases

A transform coder decomposes signals in an orthonormal basis $\mathcal{B} = \{g_m\}_{0 \leq m < N}$ and optimizes the compression of the decomposition coefficients. The performance of such a transform code is first studied from a Bayes point of view, by supposing that the signal is the realization of a random process $F[n]$ of size N, which has a probability distribution that is known a priori.

Let us decompose F over $\mathcal{B}$:

$$F = \sum_{m=0}^{N-1} F_{\mathcal{B}}[m]\, g_m .$$

Each coefficient $F_{\mathcal{B}}[m]$ is a random variable defined by

$$F_{\mathcal{B}}[m] = \langle F, g_m \rangle = \sum_{n=0}^{N-1} F[n]\, g_m^*[n].$$

To center the variations of $F_{\mathcal{B}}[m]$ at zero, we code $F_{\mathcal{B}}[m] - E\{F_{\mathcal{B}}[m]\}$ and store the mean value $E\{F_{\mathcal{B}}[m]\}$. This is equivalent to supposing that $F_{\mathcal{B}}[m]$ has a zero mean.

Quantization

To construct a finite code, each coefficient $F_{\mathcal{B}}[m]$ is approximated by a quantized variable $\tilde{F}_{\mathcal{B}}[m]$, which takes its values over a finite set of real numbers. A scalar quantization approximates each $F_{\mathcal{B}}[m]$ independently. If the coefficients $F_{\mathcal{B}}[m]$ are highly dependent, quantizer performance is improved by vector quantizers that approximate the vector of N coefficients $\{F_{\mathcal{B}}[m]\}_{0 \leq m < N}$ together [27]. Scalar quantizers require fewer computations and are thus more often used. If the basis

$\{g_m\}_{0 \leq m < N}$ can be chosen so that the coefficients $F_{\mathcal{B}}[m]$ are nearly independent, the improvement of a vector quantizer becomes marginal. After quantization, the reconstructed signal is

$$\tilde{F} = \sum_{m=0}^{N-1} \tilde{F}_{\mathcal{B}}[m]\, g_m.$$

Distortion Rate

Let us evaluate the distortion introduced by this quantization. Ultimately, we want to restore a signal that is perceived as nearly identical to the original signal. Perceptual transform codes are optimized with respect to our sensitivity to degradations in audio signals and images [317]. However, distances that evaluate perceptual errors are highly nonlinear and thus difficult to manipulate mathematically. A mean-square norm often does not properly quantify the perceived distortion, but reducing a mean-square distortion generally enhances the coder performance. Weighted mean-square distances can provide better measurements of perceived errors and are optimized like a standard mean-square norm.

In the following, we try to minimize the average coding distortion, evaluated with a mean-square norm. Since the basis is orthogonal, this distortion can be written as

$$d = E\{\|F - \tilde{F}\|^2\} = \sum_{m=0}^{N-1} E\{|F_{\mathcal{B}}[m] - \tilde{F}_{\mathcal{B}}[m]|^2\}.$$

The average number of bits allocated to encode a quantized coefficient $\tilde{F}_{\mathcal{B}}[m]$ is denoted as R_m. For a given R_m, a scalar quantizer is designed to minimize $E\{|F_{\mathcal{B}}[m] - \tilde{F}_{\mathcal{B}}[m]|^2\}$. The total mean-square distortion d depends on the average total bit budget

$$R = \sum_{m=0}^{N-1} R_m.$$

The function $d(R)$ is called the *distortion rate*. For a given R, the bit allocation $\{R_m\}_{0 \leq m < N}$ must be adjusted in order to minimize $d(R)$.

Choice of Basis

The distortion rate of an optimized transform code depends on the orthonormal basis $\mathcal{B}$. We see in Section 10.3.2 that the Karhunen-Loève basis minimizes $d(R)$ for high-resolution quantizations of signals that are realizations of a Gaussian process. However, this is not true when the process is non-Gaussian.

To achieve a high compression rate, the transform code must produce many zero-quantized coefficients, and thus define a sparse signal representation. Section 10.4 shows that $d(R)$ then depends on the precision of nonlinear approximations in the basis $\mathcal{B}$.

10.2 DISTORTION RATE OF QUANTIZATION

Quantized coefficients take their values over a finite set and can thus be coded with a finite number of bits. Section 10.2.1 reviews entropy codes of random sources. Section 10.2.2 studies the optimization of scalar quantizers, in order to reduce the mean-square error for a given bit allocation.

10.2.1 Entropy Coding

Let X be a random source that takes its values among a finite alphabet of K symbols $\mathcal{A} = \{x_k\}_{1 \leq k \leq K}$. The goal is to minimize the average bit rate needed to store the values of X. We consider codes that associate to each symbol x_k a binary word w_k of length l_k. A sequence of values produced by the source X is coded by aggregating the corresponding binary words.

All symbols x_k can be coded with binary words of the same size $l_k = \lceil \log_2 K \rceil$ bits. However, the average code length may be reduced with a *variable-length code* using smaller binary words for symbols that occur frequently. Let us denote with p_k the probability of occurrence of a symbol x_k:

$$p_k = \Pr\{X = x_k\}.$$

The average bit rate to code each symbol emitted by the source X is

$$R_X = \sum_{k=1}^{K} l_k \, p_k. \tag{10.1}$$

We want to optimize the code words $\{w_k\}_{1 \leq k \leq K}$ in order to minimize R_X.

Prefix Code

Codes with words of varying lengths are not always uniquely decodable. Let us consider the code that associates to $\{x_k\}_{1 \leq k \leq 4}$ the code words

$$\{w_1 = 0,\ w_2 = 10,\ w_3 = 110,\ w_4 = 101\}. \tag{10.2}$$

The sequence 1010 can either correspond to $w_2\, w_2$ or to $w_4\, w_1$. To guarantee that any aggregation of code words is uniquely decodable, the *prefix* condition imposes that no code word may be the prefix (beginning) of another one. The code (10.2) does not satisfy this condition since w_2 is the prefix of w_4. The code

$$\{w_1 = 0,\ w_2 = 10,\ w_3 = 110,\ w_4 = 111\}$$

satisfies this prefix condition. Any code that satisfies the prefix condition is clearly uniquely decodable.

A prefix code is characterized by a binary tree that has K leaves corresponding to the symbols $\{x_k\}_{1 \leq k \leq K}$. Figure 10.1 shows an example for a prefix code of $K = 6$ symbols. The left and right branches of the binary tree are, respectively, coded by 0 and 1. The binary code word w_k associated to x_k is the succession of 0 and 1

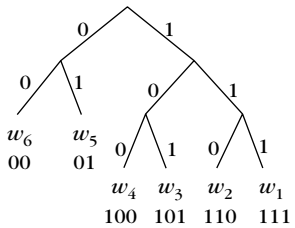


FIGURE 10.1

Prefix tree corresponding to a code with six symbols. The code word w_k of each leaf is indicated below it.

corresponding, respectively, to the left and right branches along the path from the root to the leaf x_k. The binary code produced by such a binary tree is always a prefix code. Indeed, w_m is a prefix of w_k if and only if x_m is an ancestor of x_k in the binary tree. This is not possible since both symbols correspond to a leaf of the tree. Conversely, we can verify that any prefix code can be represented by such a binary tree.

The length l_k of the code word w_k is the depth in the binary tree of the corresponding leaf. Thus, the optimization of a prefix code is equivalent to the construction of an optimal binary tree that distributes the depth of the leaves in order to minimize

$$R_X = \sum_{k=1}^{K} l_k p_k. \tag{10.3}$$

Therefore, higher-probability symbols should correspond to leaves higher in the tree.

Shannon Entropy

The Shannon theorem [429] proves that entropy is a lower bound for the average bit rate R_X of any prefix code.

Theorem 10.1: *Shannon.* Let X be a source with symbols $\{x_k\}_{1 \leqslant k \leqslant K}$ that occur with probabilities $\{p_k\}_{1 \leqslant k \leqslant K}$. The average bit rate R_X of a prefix code satisfies

$$R_X \geqslant \mathcal{H}(X) = -\sum_{k=1}^{K} p_k \log_2 p_k. \tag{10.4}$$

Moreover, there exists a prefix code such that

$$R_X \leqslant \mathcal{H}(X) + 1 \tag{10.5}$$

and $\mathcal{H}(X)$ is called the *entropy* of X.

Proof. This theorem is based on the Kraft inequality given by Lemma 10.1.

Lemma 10.1: *Kraft.* Any prefix code satisfies

$$\sum_{k=1}^{K} 2^{-l_k} \leqslant 1. \tag{10.6}$$

Conversely, if $\{l_k\}_{1\leqslant k\leqslant K}$ is a positive sequence that satisfies (10.6), then a sequence of binary words $\{w_k\}_{1\leqslant k\leqslant K}$ of length $\{l_k\}_{1\leqslant k\leqslant K}$ exists that satisfies the prefix condition.

To prove (10.6), we construct a full binary tree T the leaves of which are at the depth $m = \max\{l_1, l_2, \ldots, l_K\}$. Inside this tree, we can locate node n_k at depth l_k that codes the binary word w_k. We denote T_k to the subtree with the root of n_k, as illustrated in Figure 10.2. This subtree has a depth $m - l_k$ and thus contains 2^{m-l_k} nodes at the level m of T. There are 2^m nodes at the depth m of T and the prefix condition implies that the subtrees $T_1, \ldots, T_K$ have no node in common, so

$$\sum_{k=1}^{K} 2^{m-l_k} \leqslant 2^m,$$

which proves (10.6).

Conversely, we consider $\{l_k\}_{1\leqslant k\leqslant K}$ that satisfies (10.6), with $l_1 \leqslant l_2 \leqslant \cdots \leqslant l_K$ and $m = \max\{l_1, l_2, \ldots, l_K\}$. Again, we construct a full binary tree T with leaves at depth m. Let S_1 be the 2^{m-l_1} first nodes at level m, S_2 be the next 2^{m-l_2} nodes, and so on, as illustrated in Figure 10.2. Since $\sum_{k=1}^{K} 2^{m-l_k} \leqslant 2^m$, the sets $\{S_k\}_{1\leqslant k\leqslant K}$ have fewer than 2^m elements and can thus be constructed at level m of the tree. The nodes of a set S_k are the leaves of a subtree T_k of T. The root n_k of T_k is at depth l_k and corresponds to a binary word w_k. By construction, all these subtrees T_k are distinct, so $\{w_k\}_{1\leqslant k\leqslant K}$ is a prefix code where each code word w_k has a length l_k. This finishes the lemma proof.

To prove the two inequalities (10.4) and (10.5) of the theorem, we consider the minimization of

$$R_X = \sum_{k=1}^{K} p_k l_k$$

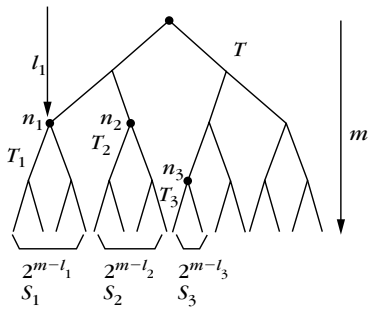


FIGURE 10.2

The leaves at depth m of tree T are regrouped as sets S_k of 2^{m-l_k} nodes that are the leaves of tree T_k, having its root n_k at depth l_k. Here, $m = 4$ and $l_1 = 2$, so S_1 has 2^2 nodes.

under the Kraft inequality constraint

$$\sum_{k=1}^{K} 2^{-l_k} \leqslant 1.$$

If we admit noninteger values for l_k, we can verify with Lagrange multipliers that the minimum is reached for $l_k = -\log_2 p_k$. The value of this minimum is the entropy lower bound:

$$R_X = \sum_{k=1}^{K} p_k l_k = -\sum_{k=1}^{K} p_k \log_2 p_k = \mathcal{H}(X),$$

which proves (10.4).

To guarantee that l_k is an integer, the Shannon code is defined by

$$l_k = \lceil -\log_2 p_k \rceil, \tag{10.7}$$

where $\lceil x \rceil$ is the smallest integer larger than x. Since $l_k \geqslant -\log_2 p_k$, the Kraft inequality is satisfied:

$$\sum_{k=1}^{K} 2^{-l_k} \leqslant \sum_{k=1}^{K} 2^{\log_2 p_k} = 1.$$

Lemma 10.1 proves that there exists a prefix code with binary words w_k that have length w_k. For this code,

$$R_X = \sum_{k=1}^{K} p_k l_k \leqslant \sum_{k=1}^{K} p_k(-\log_2 p_k + 1) = \mathcal{H}(X) + 1,$$

which proves (10.5). ■

The entropy $\mathcal{H}(X)$ measures the uncertainty as to the outcome of the random variable X, and

$$0 \leqslant \mathcal{H}(X) \leqslant \log_2 K.$$

The maximum value $\log_2 K$ corresponds to a sequence with a uniform probability distribution $p_k = 1/K$ for $1 \leqslant k \leqslant K$. Since no value is more probable than any other, the uncertainty as to the outcome of X is maximum. The minimum entropy value $\mathcal{H}(X) = 0$ corresponds to a source where one symbol x_k occurs with probability 1. There is no uncertainty as to the outcome of X because we know in advance that it will be equal to x_k.

Huffman Code

The entropy lower bound $\mathcal{H}(X)$ is nearly reachable with an optimized prefix code. The *Huffman algorithm* is a dynamical programming algorithm that constructs a binary tree that minimizes the average bit rate $R_X = \sum_{k=1}^{K} p_k l_k$. This tree is called

an *optimal prefix-code tree*. Theorem 10.2 gives an induction rule that constructs the tree from bottom up by aggregating lower-probability symbols.

Theorem 10.2: *Huffman.* Let us consider K symbols with their probability of occurrence sorted in increasing order $p_k \leq p_{k+1}$:

$$\big\{(x_1, p_1), (x_2, p_2), (x_3, p_3), \ldots, (x_K, p_K)\big\}. \tag{10.8}$$

We aggregate the two lower-probability symbols x_1 and x_2 in a single symbol $x_{1,2}$ of probability

$$p_{1,2} = p_1 + p_2.$$

An optimal prefix–code tree for the K symbols (10.8) is obtained by constructing an optimal prefix–code tree for the $K-1$ symbols,

$$\{(x_{1,2}, p_{1,2}), (x_3, p_3), \ldots, (x_K, p_K)\}, \tag{10.9}$$

and by dividing the leaf $x_{1,2}$ into two children nodes corresponding to x_1 and x_2.

The proof of this theorem [27, 307] is left to the reader. The Huffman rule reduces the construction of an optimal prefix-code tree of K symbols (10.8) to the construction of an optimal code of $K-1$ symbols (10.9) plus an elementary operation. The Huffman algorithm iterates this regrouping $K-1$ times to grow an optimal prefix-code tree progressively from bottom to top. The Shannon theorem (10.1) proves that the average bit rate of the Huffman optimal prefix code satisfies

$$\mathcal{H}(X) \leq R_X \leq \mathcal{H}(X) + 1. \tag{10.10}$$

As explained in the proof of Theorem 10.1, the bit rate may be up to 1 bit more than the entropy lower bound because this lower bound is obtained with $l_k = -\log_2 p_k$, which is generally not possible since l_k must be an integer. In particular, lower bit rates are achieved when one symbol has a probability close to 1.

EXAMPLE 10.1

We construct the Huffman code with six symbols $\{x_k\}_{1 \leq k \leq 6}$ of probabilities

$$\big\{p_1 = 0.05,\ p_2 = 0.1,\ p_3 = 0.1,\ p_4 = 0.15,\ p_5 = 0.2,\ p_6 = 0.4\big\}.$$

The symbols x_1 and x_2 are the lower-probability symbols, which are regrouped in a symbol $x_{1,2}$ with a probability of $p_{1,2} = p_1 + p_2 = 0.15$. At the next iteration, the lower probabilities are $p_3 = 0.1$ and $p_{1,2} = 0.15$, so we regroup $x_{1,2}$ and x_3 in a symbol $x_{1,2,3}$ with a probability of 0.25. The next two lower-probability symbols are x_4 and x_5, which are regrouped in a symbol $x_{4,5}$ with a probability of 0.35. We then group $x_{4,5}$ and $x_{1,2,3}$, which yields $x_{1,2,3,4,5}$ with a probability of 0.6, which is finally aggregated with x_6. This finishes the tree, as illustrated in Figure 10.3. The resulting average bit rate (10.3) is $R_X = 2.35$, whereas the entropy is

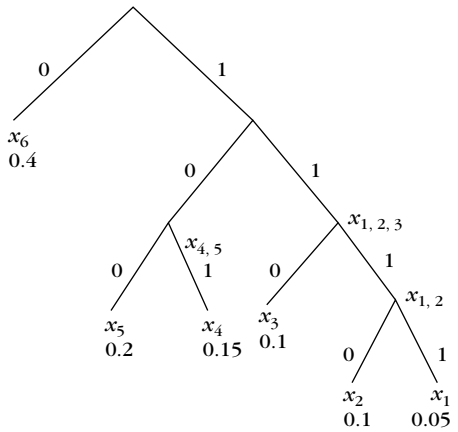


FIGURE 10.3

Prefix tree grown with the Huffman algorithm for a set of $K = 6$ symbols x_k, with probabilities p_k indicated at the leaves of the tree.

$\mathcal{H}(X) = 2.28$. This Huffman code is better than the prefix code in Figure 10.1, which has an average bit rate of $R_X = 2.4$.

Block Coding

As mentioned earlier, the inequality (10.10) shows that a Huffman code may require 1 bit above the entropy because the length l_k of each binary word must be an integer, whereas the optimal value $-\log_2 p_k$ is generally a real number. To reduce this overhead the symbols are coded together in blocks of size n.

Let us consider the block of n independent random variables $\vec{X} = X_1, \ldots, X_n$, where each X_k takes its values in the alphabet $\mathcal{A} = \{x_k\}_{1 \leqslant k \leqslant K}$ with the same probability distribution as X. The vector $\vec{X}$ can be considered as a random variable taking its values in the alphabet $\mathcal{A}^n$ of size K^n. To each block of symbols $\vec{s} \in \mathcal{A}^n$, we associate a binary word of length $l(\vec{s})$. The average number of bits per symbol for such a code is

$$R_X = \frac{1}{n} \sum_{\vec{s} \in \mathcal{A}^n} p(\vec{s})\, l(\vec{s}).$$

Theorem 10.3 proves that the resulting Huffman code has a bit rate that converges to the entropy of X as n increases.

Theorem 10.3. The Huffman code for a block of size n requires an average number of bits per symbol that satisfies

$$\mathcal{H}(X) \leqslant R_X \leqslant \mathcal{H}(X) + \frac{1}{n}. \tag{10.11}$$

Proof. The entropy of $\vec{X}$ considered as a random variable is

$$\mathcal{H}(\vec{X}) = \sum_{\vec{s} \in \mathcal{A}^n} p(\vec{s})\, \log_2 p(\vec{s}).$$

Denote by $R_{\vec{X}}$ the average number of bits to code each block $\vec{X}$. Applying (10.10) shows that with a Huffman code, $R_{\vec{X}}$ satisfies

$$\mathcal{H}(\vec{X}) \leqslant \vec{R}_{\vec{X}} \leqslant \mathcal{H}(\vec{X}) + 1. \tag{10.12}$$

Since the random variables X_i that compose $\vec{X}$ are independent,

$$p(\vec{s}) = p(s_1, \ldots, s_n) = \prod_{i=1}^{n} p(s_i).$$

Thus, we derive that $\mathcal{H}(\vec{X}) = n\,\mathcal{H}(X)$, and since $R = \vec{R}/n$, we obtain (10.11) from (10.12). ■

Coding together the symbols in blocks is equivalent to coding each symbol x_k with an average number of bits l_k that is not an integer. This explains why block coding can nearly reach the entropy lower bound. The Huffman code can also be adaptively modified for long sequences in which the probability of occurrence of the symbols may vary [18]. The probability distribution is computed from the histogram (cumulative distribution) of the N most recent symbols that were decoded. The next N symbols are coded with a new Huffman code calculated from the updated probability distribution. However, recomputing the Huffman code after updating the probability distribution is computationally expensive. Arithmetic codes have a causality structure that makes it easier to adapt the code to a varying probability distribution.

Arithmetic Code

Like a block Huffman code, an arithmetic code [411] records the symbols $\{x_k\}_{1 \leqslant k \leqslant K}$ in blocks to be coded. However, an arithmetic code is more structured; it progressively constructs the code of a whole block as each symbol is taken into account. When the probability p_k of each symbol x_k is not known, an adaptive arithmetic code progressively learns the probability distribution of the source and adapts the encoding.

We consider a block of symbols $\vec{s} = s_1, s_2, \ldots, s_n$ produced by a random vector $\vec{X} = X_1, \ldots, X_n$ of n independent random variables. Each X_k has the same probability distribution $p(x)$ as the source X, with $p(x_j) = p_j$. An arithmetic code represents each $\vec{s}$ by an interval $[a_n, a_n + b_n]$ included in $[0, 1]$, with a length equal to the probability of occurrence of this sequence:

$$b_n = \prod_{k=1}^{n} p(s_k).$$

This interval is defined by induction as follows. We initialize $a_0 = 0$ and $b_0 = 1$. Let $[a_i, a_i + b_i]$ be the interval corresponding to the first i symbols $s_1, \ldots, s_i$. Suppose that the next symbol s_{i+1} is equal to x_j so that $p(s_{i+1}) = p_j$. The new interval $[a_{i+1}, a_{i+1} + b_{i+1}]$ is a subinterval of $[a_i, a_i + b_i]$ with a size reduced by p_j:

$$a_{i+1} = a_i + b_i \sum_{k=1}^{j-1} p_k \quad \text{and} \quad b_{i+1} = b_i\, p_j.$$

The final interval $[a_n, a_n + b_n]$ characterizes the sequence $s_1, \ldots, s_n$ unambiguously because the K^n different blocks of symbols $\vec{s}$ correspond to K^n different intervals that make a partition of $[0, 1]$. Since these intervals are nonoverlapping, $[a_n, a_n + b_n]$ is characterized by coding a number $c_n \in [a_n, a_n + b_n]$ in binary form. The binary expression of the chosen numbers c_n for each of the K^n intervals defines a prefix code so that a sequence of such numbers is uniquely decodable. The value of c_n is progressively calculated by adding refinement bits when $[a_i, a_i + b_i]$ is reduced in the next subinterval $[a_{i+1}, a_{i+1} + b_{i+1}]$ until $[a_n, a_n + b_n]$.

There are efficient implementations that avoid numerical errors caused by the finite precision of arithmetic calculations when calculating c_n [488]. The resulting binary number c_n has d_n digits with

$$-\lceil \log_2 b_n \rceil \leq d_n \leq -\lceil \log_2 b_n \rceil + 2.$$

Since $\log_2 b_n = \sum_{i=1}^{n} \log_2 p(s_i)$ and $\mathcal{H}(X) = E\{\log_2 X\}$, one can verify that the average number of bits per symbol of this arithmetic code satisfies

$$\mathcal{H}(X) \leq R_X \leq \mathcal{H}(X) + \frac{2}{n}. \tag{10.13}$$

When the successive values X_k of the blocks are not independent, the upper and lower bounds (10.13) remain valid because the successive symbols are encoded as if they were independent.

An arithmetic code has a causal structure in the sense that the first i symbols of a sequence $s_1, \ldots, s_i, s_{i+1}, \ldots, s_n$ are specified by an interval $[a_i, a_i + b_i]$ that does not depend on the value of the last $n - i$ symbols. Since the sequence is progressively coded and decoded, one can implement an adaptive version that progressively learns the probability distribution $p(x)$ [377, 412]. When coding s_{i+1}, this probability distribution can be approximated by the histogram (cumulative distribution) $p_i(x)$ of the first i symbols. The subinterval of $[a_i, a_i + b_i]$ associated to s_{i+1} is calculated with this estimated probability distribution. Suppose that $s_{i+1} = x_j$; we denote $p_i(x_j) = p_{i,j}$. The new interval is defined by

$$a_{i+1} = a_i + b_i \sum_{k=1}^{j-1} p_{i,k} \quad \text{and} \quad b_{i+1} = b_i\, p_{i,j}. \tag{10.14}$$

The decoder is able to recover s_{i+1} by recovering the first i symbols of the sequence and computing the cumulative probability distribution $p_i(x)$ of these symbols. The interval $[a_{i+1}, a_{i+1} + b_{i+1}]$ is then calculated from $[a_i, a_i + b_i]$ with (10.14). The initial distribution $p_0(x)$ can be set to be uniform.

If the symbols of the block are produced by independent random variables, then as i increases, the estimated probability distribution $p_i(x)$ converges to the probability distribution $p(x)$ of the source. As the total block size n increases to $+\infty$, one can prove that the average bit rate of this adaptive arithmetic code converges to the entropy of the source. Under weaker Markov random-chain hypotheses this result also remains valid [412].

Noise Sensitivity

Huffman and arithmetic codes are more compact than a simple fixed-length code of size $\log_2 K$, but they are also more sensitive to errors. For a constant-length code, a single bit error modifies the value of only one symbol. In contrast, a single bit error in a variable-length code may modify the whole symbol sequence. In noisy transmissions where such errors might occur, it is necessary to use an error correction code that introduces a slight redundancy in order to suppress the transmission errors [18].

10.2.2 Scalar Quantization

If the source X has arbitrary real values, it cannot be coded with a finite number of bits. A scalar quantizer Q approximates X by $\tilde{X} = Q(X)$, which takes its values over a finite set. We study the optimization of such a quantizer in order to minimize the number of bits needed to code $\tilde{X}$ for a given mean-square error

$$d = E\{(X - \tilde{X})^2\}.$$

Suppose that X takes its values in $[a, b]$, which may correspond to the whole real axis. We decompose $[a, b]$ in K intervals $\{(y_{k-1}, y_k]\}_{1 \leq k \leq K}$ of variable length, with $y_0 = a$ and $y_K = b$. A scalar quantizer approximates all $x \in (y_{k-1}, y_k]$ by x_k:

$$\forall x \in (y_{k-1}, y_k], \quad Q(x) = x_k.$$

The intervals $(y_{k-1}, y_k]$ are called *quantization bins*. Rounding off integers is a simple example where the quantization bins $(y_{k-1}, y_k] = (k - \frac{1}{2}, k + \frac{1}{2}]$ have size 1 and $x_k = k$ for any $k \in \mathbb{Z}$.

High-Resolution Quantizer

Let $p(x)$ be the probability density of the random source X. The mean-square quantization error is

$$d = E\{(X - \tilde{X})^2\} = \int_{-\infty}^{+\infty} \Big(x - Q(x)\Big)^2 p(x)\, dx. \tag{10.15}$$

A quantizer is said to have a *high resolution* if $p(x)$ is approximately constant on each quantization bin $(y_{k-1}, y_k]$ of size $\Delta_k = y_k - y_{k-1}$. This is the case if the sizes Δ_k are sufficiently small relative to the rate of variation of $p(x)$, so that one can neglect these variations in each quantization bin. We then have

$$p(x) = \frac{p_k}{\Delta_k} \quad \text{for } x \in (y_{k-1}, y_k], \tag{10.16}$$

where

$$p_k = \Pr\{X \in (y_{k-1}, y_k]\}.$$

Theorem 10.4 computes the mean-square error under this high-resolution hypothesis.

Theorem 10.4. For a high-resolution quantizer, the mean-square error d is minimized when $x_k = (y_k + y_{k-1})/2$, which yields

$$d = \frac{1}{12} \sum_{k=1}^{K} p_k \Delta_k^2. \tag{10.17}$$

Proof. The quantization error (10.15) can be rewritten as

$$d = \sum_{k=1}^{K} \int_{y_{k-1}}^{y_k} (x - x_k)^2 p(x)\, dx.$$

Replacing $p(x)$ by its expression (10.16) gives

$$d = \sum_{k=1}^{K} \frac{p_k}{\Delta_k} \int_{y_{k-1}}^{y_k} (x - x_k)^2\, dx. \tag{10.18}$$

One can verify that each integral is minimum for $x_k = (y_k + y_{k-1})/2$, which yields (10.17). ■

Uniform Quantizer

The uniform quantizer is an important special case where all quantization bins have the same size

$$y_k - y_{k-1} = \Delta \quad \text{for} \quad 1 \leqslant k \leqslant K.$$

For a high-resolution uniform quantizer, the average quadratic distortion (10.17) becomes

$$d = \frac{\Delta^2}{12} \sum_{k=1}^{K} p_k = \frac{\Delta^2}{12}. \tag{10.19}$$

It is independent of the probability density $p(x)$ of the source.

Entropy Constrained Quantizer

We want to minimize the number of bits required to code the quantized values $\tilde{X} = Q(X)$ for a fixed distortion $d = E\{(X - \tilde{X})^2\}$. The Shannon theorem (10.1) proves that the minimum average number of bits to code $\tilde{X}$ is the entropy $\mathcal{H}(\tilde{X})$. Huffman and arithmetic codes produce bit rates close to this entropy lower bound. Thus, we design a quantizer that minimizes $\mathcal{H}(\tilde{X})$.

The quantized source $\tilde{X}$ takes K possible values $\{x_k\}_{1 \leqslant k \leqslant K}$ with probabilities

$$p_k = \Pr(\tilde{X} = x_k) = \Pr(X \in (y_{k-1}, y_k]) = \int_{y_{k-1}}^{y_k} p(x)\, dx.$$

Its entropy is

$$\mathcal{H}(\tilde{X}) = -\sum_{k=1}^{K} p_k \log_2 p_k.$$

For a high-resolution quantizer, Theorem 10.5 by Gish and Pierce [273] relates $\mathcal{H}(\tilde{X})$ to the *differential entropy* of X defined by

$$\mathcal{H}_d(X) = -\int_{-\infty}^{+\infty} p(x)\log_2 p(x)\,dx. \tag{10.20}$$

Theorem 10.5: *Gish, Pierce.* If Q is a high-resolution quantizer with respect to $p(x)$, then

$$\mathcal{H}(\tilde{X}) \geqslant \mathcal{H}_d(X) - \frac{1}{2}\log_2(12\,d). \tag{10.21}$$

This inequality is an equality if and only if Q is a uniform quantizer.

Proof. By definition, a high-resolution quantizer satisfies (10.16), so $p_k = p(x)\Delta_k$ for $x \in (y_{k-1}, y_k]$. Thus,

$$\begin{aligned}\mathcal{H}(\tilde{X}) &= -\sum_{k=1}^{K} p_k \log_2 p_k \\ &= -\sum_{k=1}^{K}\int_{y_{k-1}}^{y_k} p(x)\log_2 p(x)\,dx - \sum_{k=1}^{K} p_k \log_2 \Delta_k \\ &= \mathcal{H}_d(X) - \frac{1}{2}\sum_{k=1}^{K} p_k \log_2 \Delta_k^2. \end{aligned} \tag{10.22}$$

The Jensen inequality for a concave function $\phi(x)$ proves that if $p_k \geqslant 0$ with $\sum_{k=1}^{K} p_k = 1$, then for any $\{a_k\}_{1\leqslant k\leqslant K}$,

$$\sum_{k=1}^{K} p_k\,\phi(a_k) \leqslant \phi\Big(\sum_{k=1}^{K} p_k\,a_k\Big). \tag{10.23}$$

If $\phi(x)$ is strictly concave, the inequality is an equality if and only if all a_k are equal when $p_k \neq 0$. Since $\log_2(x)$ is strictly concave, we derive from (10.17) and (10.23) that

$$\frac{1}{2}\sum_{k=1}^{K} p_k \log_2(\Delta_k^2) \leqslant \frac{1}{2}\log_2\left(\sum_{k=1}^{K} p_k\,\Delta_k^2\right) = \frac{1}{2}\log_2(12\,d).$$

Inserting this in (10.22) proves that

$$\mathcal{H}(\tilde{X}) \geqslant \mathcal{H}_d(X) - \frac{1}{2}\log_2(12\,d).$$

This inequality is an equality if and only if all Δ_k are equal, which corresponds to a uniform quantizer. ■

This theorem proves that for a high-resolution quantizer, the minimum average bit rate $R_X = \mathcal{H}(\tilde{X})$ is achieved by a uniform quantizer and

$$R_X = \mathcal{H}_d(X) - \frac{1}{2}\log_2(12\,d). \tag{10.24}$$

In this case, $d = \Delta^2/12$, so

$$R_X = \mathcal{H}_d(X) - \log_2 \Delta. \tag{10.25}$$

The distortion rate is obtained by taking the inverse of (10.24):

$$d(R_X) = \frac{1}{12} 2^{2\mathcal{H}_d(X)} 2^{-2R_X}. \tag{10.26}$$

10.3 HIGH BIT RATE COMPRESSION

Section 10.3.1 studies the distortion rate performance of a transform coding computed with high-resolution quantizers. For Gaussian processes, Section 10.3.2 proves that the optimal basis is the Karhunen-Loève basis. An application to audio compression is studied in Section 10.3.3.

10.3.1 Bit Allocation

Let us optimize the transform code of a random vector $F[n]$ decomposed in an orthonormal basis $\{g_m\}_{0 \leq m < N}$:

$$F = \sum_{m=0}^{N-1} F_{\mathcal{B}}[m]\, g_m.$$

Each $F_{\mathcal{B}}[m]$ is a zero-mean source that is quantized into $\tilde{F}_{\mathcal{B}}[m]$ with an average bit budget R_m. For a high-resolution quantization, Theorem 10.5 proves that the error $d_m = E\{|F_{\mathcal{B}}[m] - \tilde{F}_{\mathcal{B}}[m]|^2\}$ is minimized with a uniform scalar quantization, and $R_m = \mathcal{H}_d(X) - \log_2 \Delta_m$ where Δ_m is the bin size.

In many applications, the overall bit budget R is fixed by some memory or transmission bandwidth constraints. Thus, we need to optimize the choice of the quantization steps $\{\Delta_m\}_{0 \leq m < N}$ to minimize the total distortion

$$d = \sum_{m=0}^{N-1} d_m$$

for a fixed-bit budget

$$R = \sum_{m=0}^{N-1} R_m.$$

The following bit allocation theorem (10.6) proves that the transform code is optimized when all Δ_m are equal, by minimizing the distortion rate Lagrangian

$$\mathcal{L}(R, d) = d + \lambda R = \sum_{m=0}^{N-1} (d_m + \lambda R_m). \tag{10.27}$$

Theorem 10.6. For a fixed-bit budget R with a high-resolution quantization, the total distortion d is minimum for

$$\Delta_m^2 = 2^{2\overline{\mathcal{H}}_d}\, 2^{-2\bar{R}} \quad \text{for} \quad 0 \leq m < N \tag{10.28}$$

with

$$\overline{R} = \frac{R}{N} \quad \text{and} \quad \overline{\mathcal{H}}_d = \frac{1}{N} \sum_{m=0}^{N-1} \mathcal{H}_d(F_{\mathcal{B}}[m]).$$

The resulting distortion rate is

$$d(\bar{R}) = \frac{N}{12} 2^{2\overline{\mathcal{H}}_d} 2^{-2\bar{R}}. \tag{10.29}$$

Proof. For uniform high-resolution quantizations, (10.26) proves that

$$d_m(R_m) = \frac{1}{12} 2^{2\mathcal{H}_d(F_{\mathcal{B}}[m])} 2^{-2R_m} \tag{10.30}$$

is a convex function of R_m, and the bit budget condition can be written as

$$R = \sum_{m=0}^{N-1} R_m = \sum_{m=0}^{N-1} \mathcal{H}_d(F_{\mathcal{B}}[m]) - \sum_{m=0}^{N-1} \frac{1}{2} \log_2(12\, d_m). \tag{10.31}$$

The minimization of $d = \sum_{m=0}^{N-1} d_m$ under the equality constraint $\sum_{m=0}^{N-1} R_m = R$ is thus a convex minimization obtained by finding the multiplier λ, which minimizes the distortion rate Lagrangian $\mathcal{L}(R, d) = \sum_{m=0}^{N-1} (d_m(R_m) + \lambda\, R_m)$. At the minimum, the relative variation of the distortion rate for each coefficient is constant and equal to the Lagrange multiplier:

$$\lambda = -\frac{\partial d_m(R_m)}{\partial R_m} = 2 d_m \log_e 2 \quad \text{for} \quad 0 \leqslant m < N.$$

Thus,

$$\Delta_m^2/12 = d_m = \frac{d}{N} = \frac{\lambda}{2 \log_e 2}. \tag{10.32}$$

Since $d_m = d/N$, the bit budget condition (10.31) becomes

$$R = \sum_{m=0}^{N-1} \mathcal{H}_d(F_{\mathcal{B}}[m]) - \frac{N}{2} \log_2 \left(\frac{12\, d}{N} \right).$$

Inverting this equation gives the expression (10.29) of $d(R)$ and inserting this result in (10.32) yields (10.28). ■

This theorem shows that the transform code is optimized if it introduces the same expected error $d_m = \Delta_m^2/12 = d/N$ along each direction g_m of the basis $\mathcal{B}$. Then, the number of bits R_m used to encode $F_{\mathcal{B}}[m]$ depends only on its differential entropy:

$$R_m = \mathcal{H}_d(F_{\mathcal{B}}[m]) - \frac{1}{2} \log_2 \left(\frac{12d}{N} \right). \tag{10.33}$$

Let σ_m^2 be the variance of $F_{\mathcal{B}}[m]$, and let $F_{\mathcal{B}}[m]/\sigma_m$ be the normalized random variable of variance 1. A simple calculation shows that

$$\mathcal{H}_d(F_{\mathcal{B}}[m]) = \mathcal{H}_d(F_{\mathcal{B}}[m]/\sigma_m) + \log_2 \sigma_m.$$

The "optimal bit allocation" R_m in (10.33) may become negative if the variance σ_m is too small, which is clearly not an admissible solution. In practice, R_m must be a positive integer, but with this condition the resulting optimal solution has no simple analytic expression (Exercise 10.8). If we neglect the integer bit constraint, (10.33) gives the optimal bit allocation as long as $R_m \geqslant 0$.

Weighted Mean-Square Error

We mentioned that a mean-square error often does not measure the perceived distortion of images or audio signals very well. When the vectors g_m are localized in time and frequency, a mean-square norm sums the errors at all times and frequencies with equal weights. Thus, it hides the temporal and frequency properties of the error $F - \tilde{F}$. Better norms can be constructed by emphasizing certain frequencies more than others in order to match our audio or visual sensitivity, which varies with the signal frequency. A weighted mean-square norm is defined by

$$d = \sum_{m=0}^{N-1} \frac{d_m}{w_m^2}, \tag{10.34}$$

where $\{w_m^2\}_{0 \leqslant m < N}$ are constant weights.

Theorem 10.6 applies to weighted mean-square errors by observing that

$$d = \sum_{m=0}^{N-1} d_m^w,$$

where $d_m^w = d_m / w_m^2$ is the quantization error of $F_{\mathcal{B}}^w[m] = F_{\mathcal{B}}[m]/w_m$. Theorem 10.6 proves that bit allocation is optimized by uniformly quantizing all $F_{\mathcal{B}}^w[m]$ with the same bin size Δ. This implies that coefficients $F_{\mathcal{B}}[m]$ are uniformly quantized with a bin size $\Delta_m = \Delta\, w_m$; it follows that $d_m = w_m^2 d/N$. As expected, larger weights increase the error in the corresponding direction. The uniform quantization Q_{Δ_m} with bins of size Δ_m can be computed from a quantizer Q that associates to any real number its closest integer:

$$Q_{\Delta_m}\left(F_{\mathcal{B}}[m]\right) = \Delta_m\, Q\left(\frac{F_{\mathcal{B}}[m]}{\Delta_m}\right) = \Delta\, w_m\, Q\left(\frac{F_{\mathcal{B}}[m]}{\Delta\, w_m}\right). \tag{10.35}$$

10.3.2 Optimal Basis and Karhunen-Loève

Transform code performance depends on the choice of an orthonormal basis $\mathcal{B}$. For high-resolution quantizations, (10.29) proves that the distortion rate $d(\bar{R})$ is optimized by choosing a basis $\mathcal{B}$ that minimizes the average differential entropy

$$\overline{\mathcal{H}}_d = \frac{1}{N} \sum_{m=0}^{N-1} \mathcal{H}_d(F_{\mathcal{B}}[m]).$$

In general, we do not know how to compute this optimal basis because the probability density of $F_{\mathcal{B}}[m] = \langle F, g_m \rangle$ may depend on g_m in a complicated way.

Gaussian Process

If F is a Gaussian random vector, then the coefficients $F_{\mathcal{B}}[m]$ are Gaussian random variables in any basis. In this case, the probability density of $F_{\mathcal{B}}[m]$ depends only on the variance σ_m^2:

$$p_m(x) = \frac{1}{\sigma_m \sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma_m^2}\right).$$

With a direct integration, we verify that

$$\mathcal{H}_d\left(F_{\mathcal{B}}[m]\right) = -\int_{-\infty}^{+\infty} p_m(x) \log_2 p_m(x)\, dx = \log_2 \sigma_m + \log_2 \sqrt{2\pi \mathrm{e}}.$$

Inserting this expression in (10.29) yields

$$d(\bar{R}) = N \frac{\pi\, \mathrm{e}}{6} \rho^2\, 2^{-2\bar{R}}, \tag{10.36}$$

where ρ^2 is the geometrical mean of all variances:

$$\rho^2 = \left(\prod_{m=0}^{N-1} \sigma_m^2\right)^{1/N}.$$

Therefore, the basis must be chosen to minimize ρ^2.

Theorem 10.7. The geometrical mean variance ρ^2 is minimized in a Karhunen-Loève basis of F.

Proof. Let K be the covariance operator of F,

$$\sigma_m^2 = \langle K g_m, g_m \rangle.$$

Observe that

$$\log_2 \rho^2 = \frac{1}{N} \sum_{m=0}^{N-1} \log_2(\langle K g_m, g_m \rangle). \tag{10.37}$$

Lemma 10.2 proves that since $C(x) = \log_2(x)$ is strictly concave, $\sum_{m=0}^{N-1} \log_2(\langle K g_m, g_m \rangle)$ is minimum if and only if $\{g_m\}_{0 \leqslant m < N}$ diagonalizes K, and thus if it is a Karhunen-Loève basis.

Lemma 10.2. Let K be a covariance operator and $\{g_m\}_{0 \leqslant m < N}$ be an orthonormal basis. If $C(x)$ is strictly concave, then $\sum_{m=0}^{N-1} C(\langle K g_m, g_m \rangle)$ is minimum if and only if K is diagonal in this basis.

To prove this lemma, let us consider a Karhunen-Loève basis $\{h_m\}_{0 \leqslant m < N}$ that diagonalizes K. As in (9.26), by decomposing g_m in the basis $\{h_i\}_{0 \leqslant i < N}$, we obtain

$$\langle K g_m, g_m \rangle = \sum_{i=0}^{N-1} |\langle g_m, h_i \rangle|^2\, \langle K h_i, h_i \rangle. \tag{10.38}$$

Since $\sum_{i=0}^{N-1} |\langle g_m, h_i\rangle|^2 = 1$, applying the Jensen inequality (A.2) to the concave function $C(x)$ proves that

$$C\left(\langle Kg_m, g_m\rangle\right) \geqslant \sum_{i=0}^{N-1} |\langle g_m, h_i\rangle|^2 \, C\left(\langle Kh_i, h_i\rangle\right). \tag{10.39}$$

Thus,

$$\sum_{m=0}^{N-1} C\left(\langle Kg_m, g_m\rangle\right) \geqslant \sum_{m=0}^{N-1}\sum_{i=0}^{N-1} |\langle g_m, h_i\rangle|^2 \, C\left(\langle Kh_i, h_i\rangle\right).$$

Since $\sum_{m=0}^{N-1} |\langle g_m, h_i\rangle|^2 = 1$, we derive that

$$\sum_{m=0}^{N-1} C\left(\langle Kg_m, g_m\rangle\right) \geqslant \sum_{i=0}^{N-1} C\left(\langle Kh_i, h_i\rangle\right).$$

This inequality is an equality if and only if for all m (10.39) is an equality. Since $C(x)$ is strictly concave, this is possible only if all values $\langle Kh_i, h_i\rangle$ are equal as long as $\langle g_m, h_i\rangle \neq 0$. Thus, we derive that g_m belongs to an eigenspace of K and is also an eigenvector of K. Thus, $\{g_m\}_{0\leqslant m<N}$ diagonalizes K as well. ■

Together with the distortion rate (10.36), this result proves that a high bit rate transform code of a Gaussian process is optimized in a Karhunen-Loève basis. The Karhunen-Loève basis diagonalizes the covariance matrix, which means that the decomposition coefficients $F_{\mathcal{B}}[m] = \langle F, g_m\rangle$ are uncorrelated. If F is a Gaussian random vector, then the coefficients $F_{\mathcal{B}}[m]$ are jointly Gaussian. In this case, being uncorrelated implies that they are independent. The optimality of a Karhunen-Loève basis is therefore quite intuitive since it produces coefficients $F_{\mathcal{B}}[m]$ that are independent. The independence of the coefficients justifies using a scalar quantization rather than a vector quantization.

Coding Gain

The Karhunen-Loève basis $\{g_m\}_{0\leqslant m<N}$ of F is a priori not well structured. The decomposition coefficients $\{\langle f, g_m\rangle\}_{0\leqslant m<N}$ of a signal f are thus computed with N^2 multiplications and additions, which is often too expensive in real-time coding applications. Transform codes often approximate this Karhunen-Loève basis by a more structured basis that admits a faster decomposition algorithm. The performance of a basis is evaluated by the coding gain [34]

$$G = \frac{E\{\|F\|^2\}}{N\,\rho^2} = \frac{\sum_{m=0}^{N-1} \sigma_m^2}{N\left(\prod_{m=0}^{N-1} \sigma_m^2\right)^{1/N}}. \tag{10.40}$$

Theorem 10.7 proves that G is maximum in a Karhunen-Loève basis.

Non-Gaussian Processes

When F is not Gaussian, the coding gain G no longer measures the coding performance of the basis. Indeed, the distortion rate (10.29) depends on the average differential entropy factor $2^{2\overline{\mathcal{H}}_d}$, which is not proportional to ρ^2. Therefore, the Karhunen-Loève basis is not optimal.

Circular stationary processes with piecewise smooth realizations are examples of non-Gaussian processes that are not well compressed in their Karhunen-Loève basis, which is the discrete Fourier basis. In Section 10.4 we show that wavelet bases yield better distortion rates because they can approximate these signals with few nonzero coefficients.

10.3.3 Transparent Audio Code

The compact disc standard samples high-quality audio signals at 44.1 kHz. Samples are quantized with 16 bits, producing a PCM of 706 kb/s. Audio codes must be "transparent," which means that they should not introduce errors that can be heard by an "average" listener.

Sounds are often modeled as realizations of Gaussian processes. This justifies the use of a Karhunen-Loève basis to minimize the distortion rate of transform codes. To approximate the Karhunen-Loève basis, we observe that many audio signals are locally stationary over a sufficiently small time interval. This means that over this time interval, the signal can be approximated by a realization of a stationary process. One can show [192] that the Karhunen-Loève basis of locally stationary processes is well approximated by a local cosine basis with appropriate window sizes. The local stationarity hypothesis is not always valid, especially for attacks of musical instruments, but bases of local time-frequency atoms remain efficient for most audio segments.

Bases of time-frequency atoms are also well adapted to matching the quantization errors with our hearing sensitivity. Instead of optimizing a mean-square error as in Theorem 10.6, perceptual coders [317] adapt the quantization so that errors fall below an auditory threshold, which depends on each time-frequency atom g_m.

Audio Masking

A large-amplitude stimulus often makes us less sensitive to smaller stimuli of a similar nature. This is called a *masking effect*. In a sound, a small-amplitude quantization error may not be heard if it is added to a strong signal component in the same frequency neighborhood. Audio masking takes place in critical frequency bands $[\omega_c - \Delta\omega/2, \omega_c + \Delta\omega/2]$ that have been measured with psychophysical experiments [425]. A strong narrow-band signal having a frequency energy in the interval $[\omega_c - \Delta\omega/2, \omega_c + \Delta\omega/2]$ decreases the hearing sensitivity within this frequency interval. However, it does not influence the sensitivity outside this frequency range. In the frequency interval [0, 20 kHz], there are approximately 25 critical bands. Below 700 Hz, the bandwidths of critical bands are on the order of 100 Hz. Above 700 Hz

the bandwidths increase proportionally to the center frequency ω_c:

$$\Delta\omega \approx \begin{cases} 100 & \text{for } \omega_c \leq 700 \\ 0.15\,\omega_c & \text{for } 700 \leq \omega_c \leq 20,000. \end{cases} \tag{10.41}$$

The masking effect also depends on the nature of the sound, particularly its tonality. A tone is a signal with a narrow-frequency support as opposed to a noiselike signal with a frequency spectrum that is spread out. A tone has a different masking influence than a noise-type signal; this difference must be taken into account [442].

Adaptive Quantization

To take advantage of audio masking, transform codes are implemented in orthogonal bases of local time-frequency atoms $\{g_m\}_{0 \leq m < N}$, with frequency supports inside critical bands. To measure the effect of audio masking at different times, the signal energy is computed in each critical band. This is done with an FFT over short time intervals, on the order of 10 ms, where signals are considered to be approximately stationary. The signal tonality is estimated by measuring the spread of its Fourier transform. The maximum admissible quantization error in each critical band is estimated depending on both the total signal energy in the band and the signal tonality. This estimation is done with approximate formulas that are established with psychophysical experiments [335]. For each vector g_m having a Fourier transform inside a given critical band, the inner product $\langle f, g_m \rangle$ is uniformly quantized according to the maximum admissible error. Quantized coefficients are then entropy coded.

Although the SNR may be as low as 13 db, such an algorithm produces a nearly transparent audio code because the quantization error is below the perceptual threshold in each critical band. The most important degradations introduced by such transform codes are *pre-echoes*. During a silence, the signal remains zero, but it can suddenly reach a large amplitude due to a beginning speech or a musical attack. In a short time interval containing this attack, the signal energy may be quite large in each critical band. By quantizing the coefficients $\langle f, g_m \rangle$ we introduce an error both in the silent part and in the attack. The error is not masked during the silence and will clearly be heard. It is perceived as a "pre-echo" of the attack. This pre-echo is due to the temporal variation of the signal, which does not respect the local stationarity hypothesis. However, it can be detected and removed with postprocessings.

Choice of Basis

The MUSICAM (Masking-pattern Universal Subband Integrated Coding and Multiplexing) coder [203] used in the MPEG-I standard [121] is the simplest perceptual subband coder. It decomposes the signal in 32 equal frequency bands of 750-Hz bandwidth, with a filter bank constructed with frequency-modulated windows of 512 samples. This decomposition is similar to a signal expansion in a local cosine basis, but the modulated windows used in MUSICAM are not orthogonal. The quantization levels are adapted in each frequency band in order to take into account the masking properties of the signal. Quantized coefficients are not entropy coded. This

system compresses audio signals up to 128 kb/s without audible impairment. It is often used for digital radio transmissions where small defects are admissible.

MP3 is the standard MPEG-1 layer 3, which operates on a large range of audio quality: Telephone quality is obtained at 8 kb/s on signals sampled at 2.5 kHz; FM radio quality is obtained at 60 kb/s for a stereo signal sampled at 11 kHz; and CD quality is obtained for bit rates going from 112 to 128 kb/s for stereo signals sampled at 22.1 kHz. To maintain a compatibility with the previous standards, the signal is decomposed both with a filter bank and a local cosine basis. The size of the windows can be adapted as in the MPEG-2 AAC (Advanced Audio Coding) standard described next. The size of the quantization bins is computed with a perceptual masking model. The parameters of the models are not specified in the standard and are thus transmitted with the signal. A Huffman code stores the quantized coefficients. For a pair of stereo signals f_1 and f_2, the compression is improved by coding an average signal $a = (f_1 + f_2)/2$ and a difference signal $d = (f_1 - f_2)/2$, and by adapting the perceptual model for the quantization of each of these signals.

MPEG-2 AAC is a standard that offers a better audio quality for a given compression ratio. The essential difference with MP3 is that it directly uses a decomposition in a local cosine basis. In order to reduce pre-echo distortions, and to adapt the basis to the stationarity intervals of the signal, the size 2^j of the windows can vary from 256 to 2048. However, on each interval of 2048 samples, the size of the window must remain constant, as illustrated in Figure 10.4. Each window has a raising and a decaying profile that is as large as possible, while overlapping only the two adjacent windows. The profile used by the standard is

$$\beta(t) = \sin\left(\frac{\pi}{4}(1+t)\right).$$

Section 8.4.1 explains in (8.85) how to construct the windows $g_p(t)$ on each interval from this profile. The discrete windows of the local cosine basis are obtained with a uniform sampling: $g_p[n] = g_p(n)$. The choice of windows can also be interpreted as a best-basis choice, further studied in Section 12.2.3. However, as opposed to the bases of the local cosine trees from Section 8.5, the windows have raising and decaying profiles of varying sizes, which are best adapted to the segmentation. The strategy to choose the window sizes is not imposed by the standard, and the

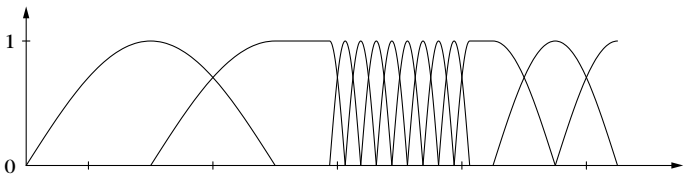


FIGURE 10.4

Succession of windows of various sizes on intervals of 2048 samples, which satisfy the orthogonality constraints of local cosine bases.

code transmits the selected window size. A best-basis algorithm must measure the coding efficiency for windows of different sizes for a given audio segment. In the neighborhood of an attack, it is necessary to choose small windows in order to reduce the pre-echo. Like in MP3, the local cosine coefficients are quantized with a perceptual model that depends on the signal energy in each critical frequency band, and an entropy code is applied.

The AC systems produced by Dolby are similar to MPEG-2 AAC. The signal is decomposed in a local cosine basis, and the window size can be adapted to the local signal content. After a perceptual quantization, a Huffman entropy code is used. These coders operate on a variety of bit rates from 64 kb/s to 192 kb/s.

To best match human perception, transform code algorithms have also been developed in wavelet packet bases, with a frequency decomposition that matches the critical frequency bands [483]. Sinha and Tewfik [442] propose the wavelet packet basis shown in Figure 10.5, which is an $M = 4$ wavelet basis. The properties of M-band wavelet bases are explained in Section 8.1.3. These four wavelets have a bandwidth of $1/4$, $1/5$, $1/6$, and $1/7$ octaves, respectively. The lower-frequency interval $[0, 700]$ is decomposed with eight wavelet packets of the same bandwidth in order to match the critical frequency bands (10.41). These wavelet packet coefficients are quantized with perceptual models and are entropy coded. Nearly transparent audio codes are obtained at 70 kb/s.

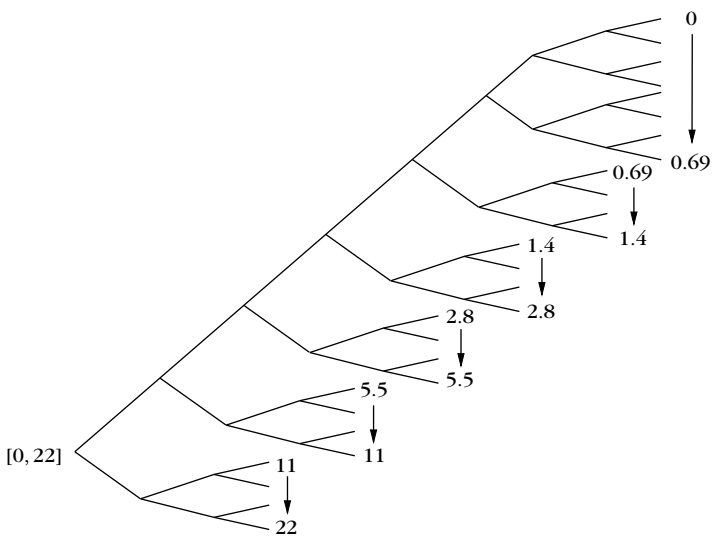


FIGURE 10.5

Wavelet packet tree that decomposes the frequency interval $[0, 22$ kHz$]$ in 24 frequency bands covered by $M = 4$ wavelets dilated over six octaves, plus 8 low-frequency bands of the same bandwidth. The frequency bands are indicated at the leaves in kHz.

Wavelets produce smaller pre-echo distortions compared to local cosine bases. At the sound attack, the largest wavelet coefficients appear at fine scales. Because fine-scale wavelets have a short support, a quantization error creates a distortion that is concentrated near the attack. However, these bases have the disadvantage of introducing a bigger coding delay than local cosine bases. The coding delay is approximately equal to the maximum time support of the vector used in the basis. It is typically larger for wavelets and wavelet packets than for local cosine vectors.

Choice of Filter

Wavelet and wavelet packet bases are constructed with a filter bank of conjugate mirror filters. For perceptual audio coding, the Fourier transform of each wavelet or wavelet packet must have its energy well concentrated in a single critical band. Second-order lobes that may appear in other frequency bands should have a negligible amplitude. Indeed, a narrow-frequency tone creates large-amplitude coefficients for all wavelets with a frequency support covering this tone, as shown in Figure 10.6. Quantizing the wavelet coefficients is equivalent to adding small wavelets with amplitude equal to the quantization error. If the wavelets excited by the tone have important second-order lobes in other frequency intervals, the quantization errors introduce some energy in these frequency intervals that is not masked by the energy of the tone, thereby introducing audible distortion.

To create wavelets and wavelet packets with small second-order frequency lobes, the transfer function of the corresponding conjugate mirror filter $\hat{h}(\omega)$ must have a zero of high order at $\omega = \pi$. Theorem 7.7 proves that conjugate mirror filters with p zeros at $\omega = \pi$ have at least $2p$ nonzero coefficients, and correspond to wavelets of size $2p - 1$. Thus, increasing p produces a longer coding delay. Numerical experiments [442] show that increasing p up to 30 can enhance the perceptual quality of the audio code, but the resulting filters have at least 60 nonzero coefficients.

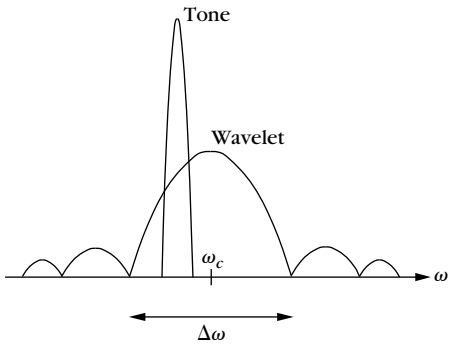


FIGURE 10.6

A high-energy, narrow-frequency tone can excite a wavelet having a Fourier transform with second-order lobes outside the critical band of width $\Delta\omega$. The quantization then creates audible distortion.

10.4 SPARSE SIGNAL COMPRESSION

Sparse representations provide high signal-compression factors by only coding a few nonzero coefficients. The high-resolution quantization hypothesis is not valid anymore. Section 10.4.1 shows that the distortion is dominated by the nonlinear approximation term, and most of the bits are devoted to coding the position of nonzero coefficients. These results are illustrated by a wavelet image transform code. Section 10.4.2 refines such transform codes with an embedding strategy with progressive coding capabilities.

The performance of transform codes was studied from a Bayes point of view, by considering signals as realizations of a random vector that has a known probability distribution. However, there is no known stochastic model that incorporates the diversity of complex signals such as nonstationary textures and edges in images. Classic processes and, in particular, Gaussian processes or homogeneous Markov random fields are not appropriate. This section introduces a different framework where the distortion rate is computed with deterministic signal models.

10.4.1 Distortion Rate and Wavelet Image Coding

The signal is considered as a deterministic vector $f \in \mathbb{C}^N$ that is decomposed in an orthonormal basis $\mathcal{B} = \{g_m\}_{0 \leq m < N}$:

$$f = \sum_{m=0}^{N-1} f_{\mathcal{B}}[m]\, g_m \quad \text{with} \quad f_{\mathcal{B}}[m] = \langle f, g_m \rangle.$$

A transform code quantizes all coefficients and reconstructs

$$\tilde{f} = \sum_{m=0}^{N-1} Q(f_{\mathcal{B}}[m])\, g_m. \tag{10.42}$$

Let R be the number of bits used to code the N quantized coefficients $Q(f_{\mathcal{B}}[m])$. The coding distortion is

$$d(R, f) = \|f - \tilde{f}\|^2 = \sum_{m=0}^{N-1} |f_{\mathcal{B}}[m] - Q(f_{\mathcal{B}}[m])|^2. \tag{10.43}$$

We denote by $p(x)$ the histogram of the N coefficients $f_{\mathcal{B}}[m]$, normalized so that $\int p(x)\, dx = 1$. The quantizer approximates each $x \in (y_{k-1}, y_k]$ by $Q(x) = x_k$. The proportion of quantized coefficients equal to x_k is

$$p_k = \int_{y_{k-1}}^{y_k} p(x)\, dx. \tag{10.44}$$

Suppose that the quantized coefficients of f can take at most K different quantized values x_k. A variable-length code represents the quantized values equal to x_k with an average of l_k bits, where the lengths l_k are specified independently from f. It is implemented with a prefix code or an arithmetic code over blocks of quantized

values that are large enough so that the l_k can be assumed to take any real values that satisfy the Kraft inequality (10.6) $\sum_{k=1}^{K} 2^{-l_k} \leq 1$. Encoding a signal with K symbols requires a total number of bits

$$R = N \sum_{k=1}^{K} p_k \, l_k. \tag{10.45}$$

The bit budget R reaches its minimum for $l_k = -\log_2 p_k$ and thus,

$$R \geq \mathcal{H}(f) = -N \sum_{k=1}^{K} p_k \, \log_2 p_k. \tag{10.46}$$

In practice, we do not know in advance the values of p_k, which depend on the signal f. An *adaptive variable-length code*, as explained in Section 10.2.1, computes an estimate $\tilde{p}_k$ of p_k with an empirical histogram of the already coded coefficients. It sets $l_k = -\log_2 \tilde{p}_k$ and updates the estimate $\tilde{p}_k$ and thus l_k as the coding progresses. Under appropriate ergodicity assumptions, the estimated $\tilde{p}_k$ converge to p_k and R to the entroy $\mathcal{H}$. The N quantized signal coefficients can be modeled as values taken by a random variable X with a probability distribution equal to the histogram $p(x)$. The distortion (10.43) can then be rewritten as

$$d(R, f) = N \, E\{|X - Q(X)|^2\},$$

and the bit budget of an adaptive variable-length code converges to entropy $R = \mathcal{H}(Q(X)) = \mathcal{H}(f)$.

If the high-resolution quantization assumption is valid, which means that $p(x)$ is nearly constant over quantization intervals, then Theorem 10.5 proves that $d(R, f)$ is minimum if and only if Q is a uniform quantizer. The resulting distortion rate computed in (10.26) is

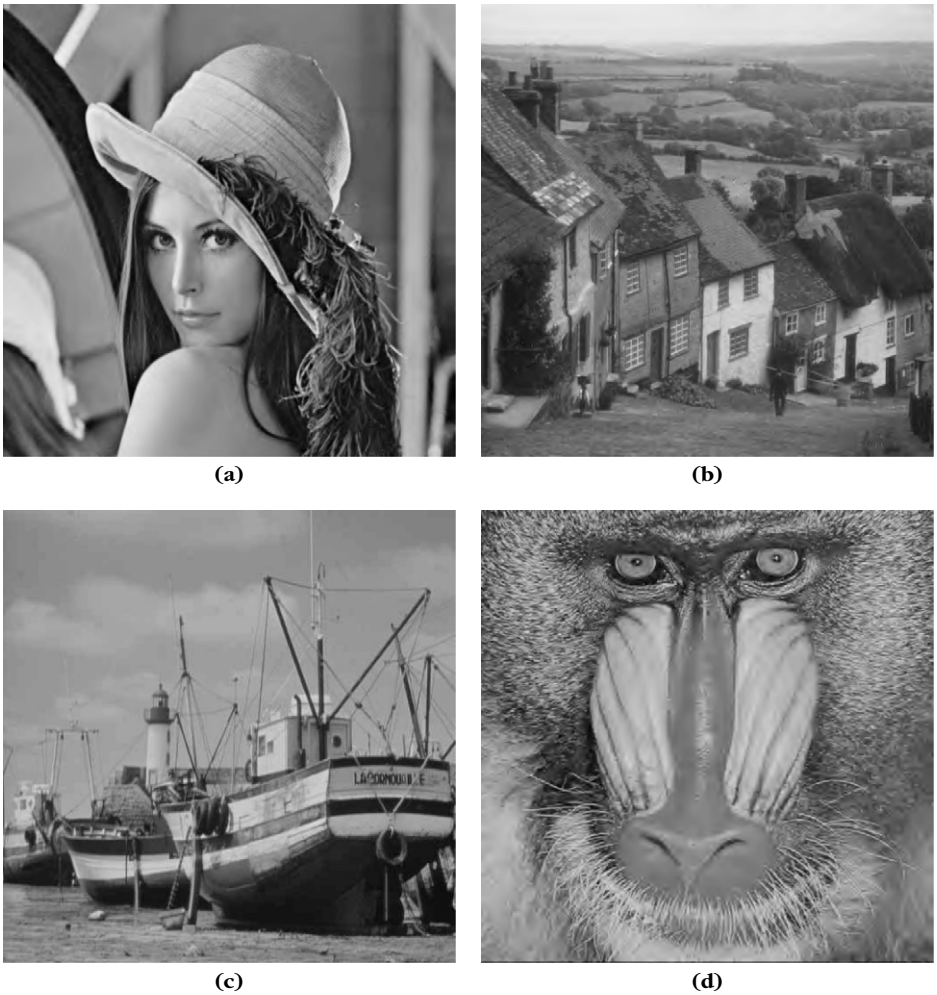
$$d(R, f) = \frac{N}{12} \, 2^{2\mathcal{H}_d(f)} \, 2^{-2R/N}, \tag{10.47}$$

with $\mathcal{H}_d(f) = -\int p(x) \log_2 p(x) \, dx$. It predicts an exponential distortion rate decay. The high-resolution quantization assumption is valid if the quantization bins are small enough, which means that R/N is sufficiently large.

Coded sequences of quantized coefficients are often not homogeneous and ergodic sources. Adaptive variable-length codes can then produce a bit budget that is below the entropy (10.46). For example, the wavelet coefficients of an image often have a larger amplitude at large scales. When coding coefficients from large to fine scales, an adaptive arithmetic code progressively adapts the estimated probability distribution. Thus, it produces a total bit budget that is often smaller than the entropy $\mathcal{H}(f)$ obtained with a fixed code globally optimized for the N wavelet coefficients.

Wavelet Image Code

A simple wavelet image code is introduced to illustrate the properties of low bit rate transform coding in sparse representations. The image is decomposed in a



(a) **(b)** **(c)** **(d)**

FIGURE 10.7

Result of a wavelet transform code with an adaptive arithmetic coding using $\bar{R} = 0.5$ bit/pixel for images of $N = 512^2$ pixels: **(a)** Lena, **(b)** GoldHill, **(c)** boats, and **(d)** mandrill.

separable wavelet basis. All wavelet coefficients are uniformly quantized and coded with an adaptive arithmetic code. Figure 10.7 shows examples of coded images with $R/N = 0.5$ bit/pixel.

The peak signal-to-noise ratio (PSNR) is defined by

$$PSNR(R, f) = 10 \log_{10} \frac{N\, 255^2}{d(R, f)}.$$

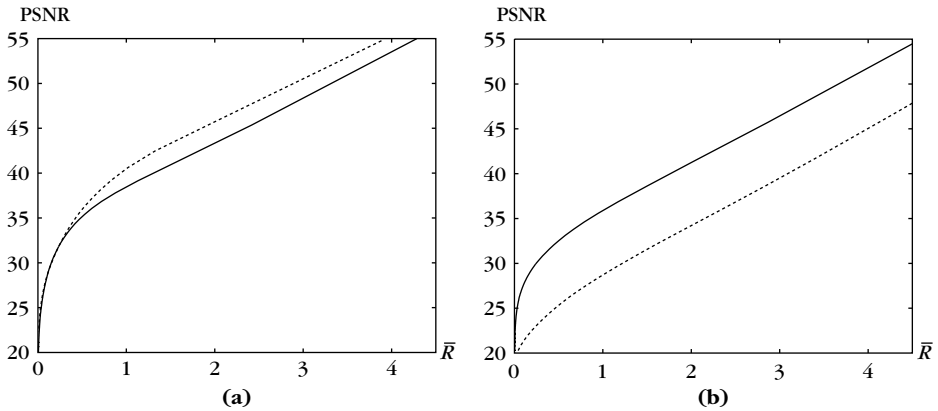


FIGURE 10.8

PSNR as a function of R/N: **(a)** Lena image (*solid line*) and boats image (*dotted line*); **(b)** GoldHill image (*solid line*) and mandrill image (*dotted line*).

The high-resolution distortion rate formula (10.47) predicts that there exists a constant K such that

$$PSNR(R, f) = (20 \log_{10} 2)\, \bar{R} + K \quad \text{with} \quad \bar{R} = R/N.$$

Figure 10.8 shows that $PSNR(R, f)$ has indeed a linear growth for $\bar{R} \geqslant 1$, but not for $\bar{R} < 1$.

At low bit rates $\bar{R} \leqslant 1$, the quantization interval Δ is relatively large. The normalized histogram $p(x)$ of wavelet coefficients in Figure 10.9 has a narrow peak in the neighborhood of $x = 0$. Thus, $p(x)$ is poorly approximated by a constant in the zero bin $[-\Delta/2, \Delta/2]$ where $Q(x) = 0$. The high-resolution quantization hypothesis is not valid in this zero bin, which explains why the distortion rate formula (10.47) is incorrect. For the mandrill image, the high-resolution hypothesis remains

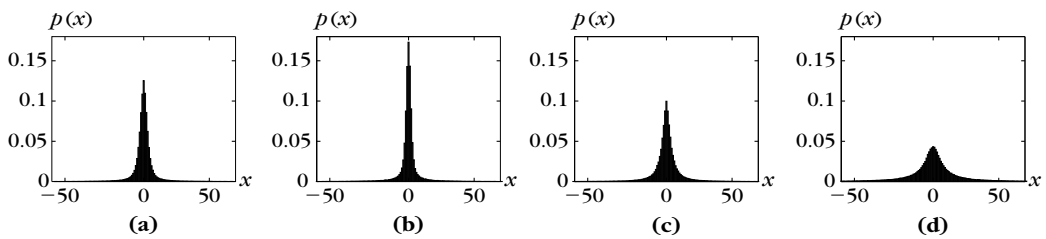


FIGURE 10.9

Normalized histograms of orthogonal wavelet coefficients for each image: **(a)** Lena, **(b)** boats, **(c)** GoldHill, and **(d)** mandrill.

valid up to $\bar{R} = 0.5$ because the histogram of its wavelet coefficients is wider in the neighborhood of $x = 0$.

Geometric Bit Budget

If the basis $\mathcal{B}$ is chosen so that many coefficients $f_{\mathcal{B}}[m] = \langle f, g_m \rangle$ are close to zero, then the histogram $p(x)$ has a sharp high-amplitude peak at $x = 0$, as in the wavelet histograms shown in Figure 10.9. The distortion rate is calculated at low bit rates, where the high-resolution quantization does not apply.

The bit budget R is computed by considering separately the set of *significant coefficients*

$$\Lambda_{\Delta/2} = \{m \,:\, Q(|f_{\mathcal{B}}[m]|) \neq 0\} = \{m \,:\, |f_{\mathcal{B}}[m]| \geqslant \Delta/2\}.$$

This set is the approximation support that specifies the geometry of a sparse transform coding. Figure 10.10 shows the approximation support of the quantized wavelet coefficients that code the four images in Figure 10.7. The total bit budget R to code all quantized coefficients is divided into the number of bits R_0 needed to code the coefficients quantized to zero, plus the number of bits R_1 to code significant coefficients:

$$R = R_0 + R_1.$$

The bit budget R_0 can also be interpreted as a geometric bit budget that codes the position of significant coefficients and thus $\Lambda_{\Delta/2}$. Let $M = |\Lambda_{\Delta/2}| \leqslant N$ be the number of significant coefficients that is coded with $\log_2 N$ bits. There are $\binom{N}{M}$ different sets of M coefficients chosen among N. To code an approximation support $\Lambda_{\Delta/2}$ without any other prior information requires a number of bits

$$R_0 = \log_2 N + \log_2 \binom{N}{M} \sim M \left(1 + \log_2 \frac{N}{M}\right).$$

Theorem 10.8 shows that the number of bits R_1 to code M significant is typically proportional to M with a variable-length code. If $M \ll N$, then the overall bit budget $R = R_0 + R_1$ is dominated by the geometric bit budget R_0.

This approximation support $\Lambda_{\Delta/2}$ can be coded with an entropy coding of the binary *significance map*

$$b[m] = \begin{cases} 0 & \text{if } Q(f_{\mathcal{B}}[m]) = 0 \\ 1 & \text{if } Q(f_{\mathcal{B}}[m]) \neq 0. \end{cases} \tag{10.48}$$

The proportions of 0 and 1 in the significance map are, respectively, $p_0 = (N - M)/N$ and $p_1 = M/N$. An arithmetic code of this significance map yields a bit budget of

$$R_0 \geqslant \mathcal{H}_0 = -N \left(p_0 \log_2 p_0 + p_1 \log_2 p_1\right), \tag{10.49}$$

which is of the same order as $\log_2 \binom{N}{M}$ (Exercise 10.5).

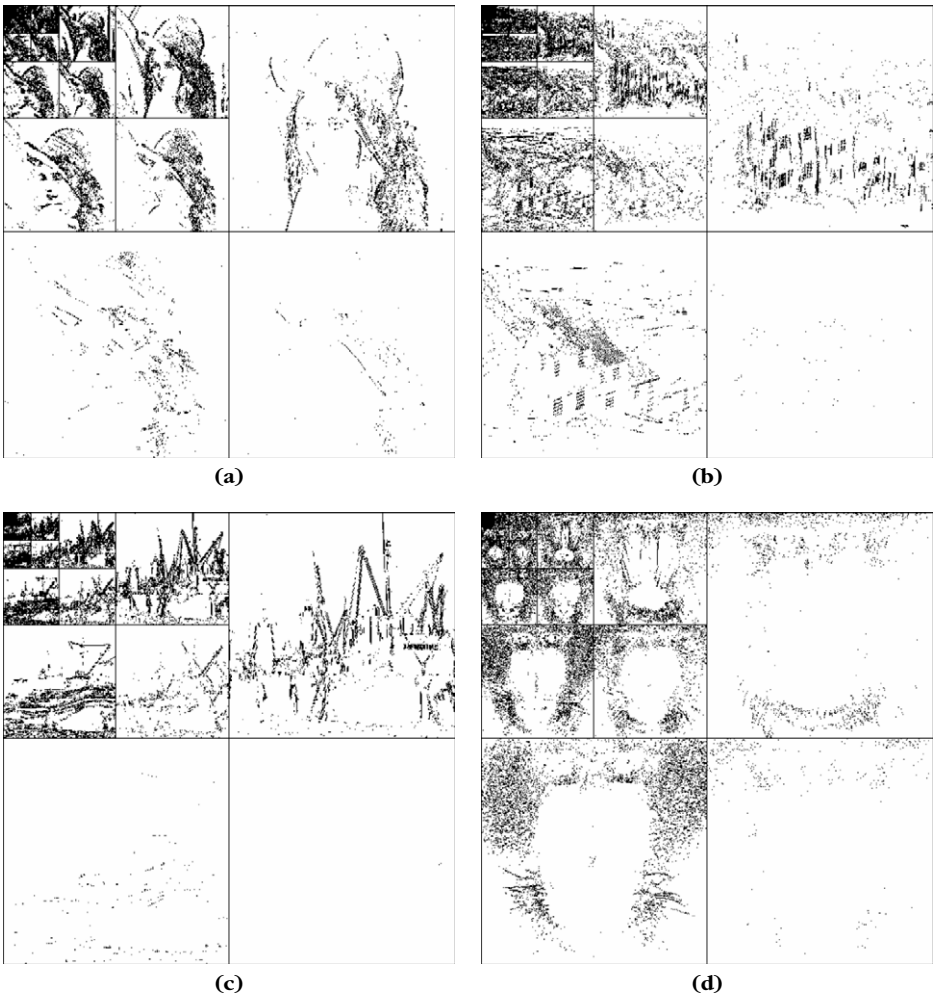


FIGURE 10.10

Significance maps of quantized wavelet coefficients for images coded with $\bar{R} = 0.5$ bit/pixel: **(a)** Lena, **(b)** GoldHill, **(c)** boats, and **(d)** mandrill.

Distortion and Nonlinear Approximation

The distortion $d(R, f)$ is calculated by separating the significant coefficients in $\Lambda_{\Delta/2}$ from other coefficients for which $Q(f_{\mathcal{B}}[m]) = 0$:

$$d(R,f) = \sum_{m=0}^{N-1} |f_{\mathcal{B}}[m] - Q(f_{\mathcal{B}}[m])|^2 \tag{10.50}$$

$$= \sum_{m \notin \Lambda_{\Delta/2}} |f_{\mathcal{B}}[m]|^2 + \sum_{m \in \Lambda_{\Delta/2}} |f_{\mathcal{B}}[m] - Q(f_{\mathcal{B}}[m])|^2. \tag{10.51}$$

Let $f_M = \sum_{m \in \Lambda_{\Delta/2}} f_{\mathcal{B}}[m]\, g_m$ be the best M-term approximation of f from the M significant coefficients above $\Delta/2$. The first sum of $d(R,f)$ can be rewritten as a nonlinear approximation error:

$$\|f - f_M\|^2 = \sum_{m \notin \Lambda_{\Delta/2}} |f_{\mathcal{B}}[m]|^2.$$

Since $0 \leqslant |x - Q(x)| \leqslant \Delta/2$, (10.50) implies that

$$\|f - f_M\|^2 \leqslant d(R,f) \leqslant \|f - f_M\|^2 + M\,\frac{\Delta^2}{4}. \tag{10.52}$$

Theorem 10.8 shows that the nonlinear approximation error $\|f - f_M\|^2$ dominates the distortion rate behavior [208].

In Section 9.2.1 we prove that nonlinear approximation errors depend on the decay of the sorted coefficients of f in $\mathcal{B}$. We denote by $f^r_{\mathcal{B}}[k] = f_{\mathcal{B}}[m_k]$ the coefficient of rank k, defined by $|f^r_{\mathcal{B}}[k]| \geqslant |f^r_{\mathcal{B}}[k+1]|$ for $1 \leqslant k \leqslant N$. We write $|f^r_{\mathcal{B}}[k]| \sim C\, k^{-s}$ if there exist two constants $A, B > 0$ independent of $C, k,$ and N such that $A\, C\, k^{-s} \leqslant |f^r_{\mathcal{B}}[k]| \leqslant B\, C\, k^{-s}$. Theorem 10.8 computes the resulting distortion rate [363].

Theorem 10.8: *Falzon, Mallat.* Let Q be a uniform quantizer. There exists a variable-length code such that for all $s > 1/2$ and $C > 0$, if $|f^r_{\mathcal{B}}[k]| \sim C\, k^{-s}$, then

$$d(R,f) \sim C^2\, R^{1-2s} \left(1 + \log_2 \frac{N}{R}\right)^{2s-1} \quad \text{for} \quad R \leqslant N. \tag{10.53}$$

Proof. Since the sorted coefficients satisfy $|f^r_{\mathcal{B}}[k]| \sim C\, k^{-s}$ and $|f^r_{\mathcal{B}}[M]| \sim \Delta$, we derive that

$$M \sim C^{1/s}\, \Delta^{-1/s}. \tag{10.54}$$

Since $s > 1/2$, the approximation error is

$$\|f - f_M\|^2 = \sum_{k=M+1}^{N} |f^r_{\mathcal{B}}[k]|^2 \sim \sum_{k=M+1}^{N} C^2\, k^{-2s} \sim C^2\, M^{1-2s}. \tag{10.55}$$

But (10.54) shows that $M\, \Delta^2 \sim C^2\, M^{1-2s}$, so (10.52) yields

$$d(R,f) \sim C^2\, M^{1-2s}. \tag{10.56}$$

We now relate the bit budget $R = R_0 + R_1$ to M. The number of bits R_0 to code M and the significance set $\Lambda_{\Delta/2}$ of size M is

$$R_0 = \log_2 N + \log_2 \binom{N}{M} \sim M \left(1 + \log_2 \frac{N}{M}\right).$$

Let us decompose $R_1 = R_a + R_s$, where R_a is the number of bits that code the amplitude of the M significant coefficients of f, and R_s is the number of bits that code their sign. Clearly, $0 \leqslant R_s \leqslant M$. The amplitude of coefficients is coded with a logarithmic variable-length code, which does not depend on the distribution of these coefficients.

Let p_j be the fraction of M significant coefficients such that $|Q(f_{\mathcal{B}}^0[k])| = j\Delta$, and thus $|f_{\mathcal{B}}^r[k]| \in [(j-1/2)\Delta, (j+1/2)\Delta)$. Since $|f_{\mathcal{B}}^r[k]| \sim C\, k^{-s}$,

$$p_j M \sim C^{1/s}\, \Delta^{-1/s}\, (j-1/2)^{-1/s} - C^{1/s}\, \Delta^{-1/s}\, (j+1/2)^{-1/s} \sim s^{-1}\, C^{1/s}\, \Delta^{-1/s}\, j^{-1/s-1}.$$

But $M \sim\sim C^{1/s}\, \Delta^{-1/s}$ so

$$p_j \sim s^{-1}\, j^{-1/s-1}. \tag{10.57}$$

Let us consider a logarithmic variable-length code

$$l_j = \log_2(\pi^2/6) + 2\log_2 j.$$

which satisfies the Kraft inequality (10.6) because

$$\sum_{j=1}^{+\infty} 2^{-l_j} = \frac{6}{\pi^2} \sum_{j=1}^{+\infty} j^{-2} = 1.$$

This variable-length code produces a bit budget

$$R_a = -M \sum_{j=1}^{+\infty} p_j\, l_j \sim M\, s^{-1} \sum_{j=1}^{+\infty} j^{-1-1/s} (\log_2(\pi^2/6) + 2\log_2 j) \sim M. \tag{10.58}$$

As a result, $R_1 = R_s + R_a \sim M$, and thus

$$R = R_0 + R_1 \sim M \left(1 + \log_2 \frac{N}{M}\right). \tag{10.59}$$

Inverting this equation gives

$$M \sim R \left(1 + \log_2 \frac{N}{R}\right)^{-1},$$

and since $d(R,f) \sim C^2\, M^{1-2s}$ in (10.56), it implies (10.53). ■

The equivalence sign $\sim$ means that lower and upper bounds of $d(R,f)$ are obtained by multiplying the right expression of (10.53) by two constants $A, B > 0$ that are independent of C, R, and N. Thus, it specifies the increase of $d(R,f)$ as R decreases. Theorem 10.8 proves that at low bit rates, the distortion is proportional to R^{1-2s}, as opposed to $2^{-2R/N}$ in the high bit rate distortion formula (10.47). The bit budget is dominated by the geometric bit budget R_0, which codes the significance map. At low bit rates, to minimize the distortion one must find a basis $\mathcal{B}$ that yields the smallest M-term approximation error.

Twice-Larger Zero Bin

Since the high-resolution quantization hypothesis does not hold, a uniform quantizer does not minimize the distortion rate. The wavelet coefficient histograms in Figure 10.9 are highly peaked and can be modeled in a first approximation by

Laplacian distributions having an exponential decay $p(x) = \mu/2\, e^{-\mu|x|}$. For Laplacian distributions, one can prove [394] that optimal quantizers that minimize the distortion rate with an entropy coder have a zero bin $[-\Delta, \Delta]$ that is twice larger than other quantization bins, which must have a constant size Δ.

Doubling the size of the zero bin often improves the distortion at low bit rates. This is valid for wavelet transform codes but also for image transform codes in block cosine bases [363]. It reduces the proportion of significant coefficients and improves the bit budget by a factor that is not offset by the increase of the quantization error. A larger zero bin increases the quantization error too much, degrading the overall distortion rate. Thus, the quantizer becomes

$$Q(x) = \begin{cases} 0 & \text{if } |x| < \Delta \\ \text{sign}(x)\, (\lfloor x/\Delta \rfloor + 1/2)\, \Delta & \text{if } |x| \geqslant \Delta \end{cases} \tag{10.60}$$

and the significance map becomes $\Lambda_\Delta = \{m : |f_{\mathcal{B}}[m]| \leqslant \Delta\}$. Theorem 10.8 remains valid for this quantizer with a twice-larger zero bin. Modifying the size of other quantization bins has a marginal effect. One can verify that the distortion rate equivalence (10.53) also holds for a nonuniform quantizer that is adjusted to minimize the distortion rate.

Bounded Variation Images

Section 2.3.3 explains that large classes of images have a bounded total variation that is proportional to the length of their contours. Theorem 9.17 proves that the sorted wavelet coefficients $f^r_{\mathcal{B}}[k]$ of bounded variation images satisfy $|f^r_{\mathcal{B}}[k]| = O(\|f\|_V\ k^{-1})$. If the image is discontinuous along an edge curve, then it creates large-amplitude wavelet coefficients and $|f^r_{\mathcal{B}}[k]| \sim \|f\|_V\, k^{-1}$. This decay property is verified by the wavelet coefficients of the Lena and boat images, which can be considered as discretizations of bounded variation functions. Theorem 10.8 derives for $s = 1$ that

$$d(R, f) \sim \|f\|_V^2\, R^{-1} \left(1 + \log_2 \frac{N}{R}\right). \tag{10.61}$$

Figure 10.11(a) shows the PSNR computed numerically from a wavelet transform code of the Lena and boats images. Since $PSNR = 10 \log_{10} d(R, f) + K$, it results from (10.61) that it increases almost linearly as a function of $\log_2 \bar{R}$, with a slope of $10 \log_{10} 2 \approx 3$ db/bit for $\bar{R} = R/N$.

More Irregular Images

The mandrill and GoldHill are examples of images that do not have a bounded variation. This appears in the fact that their sorted wavelet coefficients satisfy $|f^r_{\mathcal{B}}[k]| \sim C\, k^{-s}$ for $s < 1$. Since $PSNR = 10 \log_{10} d(R, f) + K$, it results from (10.53) that it increases with a slope of $(2s - 1)\ 10 \log_{10} 2$ as a function of $\log_2 \bar{R}$. For the GoldHill image, $s \approx 0.8$, so the PSNR increases by 1.8 db/bit. Mandrill is even more irregular, with $s \approx 2/3$, so at low bit rates $\bar{R} < 1/4$ the PSNR increases by only 1 db/bit. Such images can be modeled as the discretization of functions in Besov spaces with a regularity index $s/2 + 1/2$ smaller than 1.

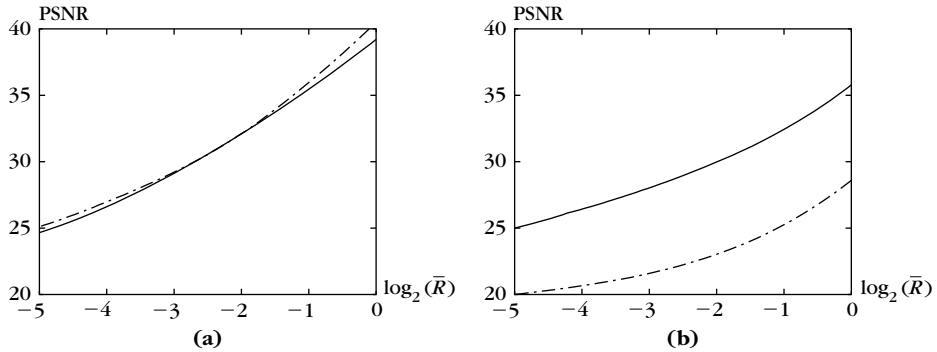


FIGURE 10.11

PSNR as a function of $\log_2(\bar{R})$. **(a)** Lena image (*solid line*) and boats image (*dotted line*) **(b)** GoldHill image (*solid line*) and mandrill image (*dotted line*).

Distortion Rate for Analog Signals

The input signal $f[n]$ is most often the discretization of an analog signal $\bar{f}(x)$, and can be written as $f[n] = \langle \bar{f}, \phi_n \rangle$ where $\{\phi_n\}_{0 \leq n < N}$ is a Riesz basis of an approximation space $\mathbf{U}_N$. To simplify explanations, we suppose that this basis is orthonormal. Let $\bar{f}_N$ be the orthogonal projection of f in $\mathbf{U}_N$ that can be recovered from $f[n]$. At the end of the processing chain, the coded signal $\tilde{f}[n]$ is converted into an analog signal in $\mathbf{U}_N$:

$$\tilde{\bar{f}}_N(x) = \sum_{n=0}^{N-1} \tilde{f}[n]\, \phi_n(x).$$

Since $\{\phi_n\}_{0 \leq n < N}$ is orthonormal, the norms over analog and discrete signals are equal: $\|\bar{f}_N - \tilde{\bar{f}}_N\| = \|f - \tilde{f}\|$.

The overall analog distortion rate $d(R, \bar{f}) = \|\bar{f} - \tilde{\bar{f}}_N\|^2$ satisfies

$$d(R, \bar{f}) = \|\bar{f}_N - \bar{f}\|^2 + \|\bar{f}_N - \tilde{\bar{f}}_N\|^2 = \|\bar{f}_N - \bar{f}\|^2 + \|f - \tilde{f}\|^2.$$

The sampling resolution N can be chosen so that the linear approximation error $\|\bar{f}_N - \bar{f}\|^2$ is smaller or of the same order as the compression error $\|f - \tilde{f}\|^2$. For most functions, such as bounded variation functions or Besov space functions, the linear approximation error satisfies $\|\bar{f}_N - \bar{f}\|^2 = O(N^{-\beta})$ for some $\beta > 0$. If discrete distortion rate $\|f - \tilde{f}\|^2$ satisfies the decay (10.53) of Theorem 10.8, then for $N = R^{(2s-1)/\beta}$, we get

$$d(R, \bar{f}) = O\left(R^{1-2s} |\log_2 R|^{2s-1}\right). \tag{10.62}$$

This result applies to bounded variation images coded in wavelet bases. Donoho [215] proved that for such functions the decay exponent R^{-1} cannot be improved

by any other signal coder. In that sense, a wavelet transform coding is optimal for bounded variation images. Functions in Besov spaces coded in wavelet bases also have a distortion rate that satisfies (10.62) for an exponent s that depends on the space. The optimality of wavelet transform codes in Besov spaces is also studied in [173].

10.4.2 Embedded Transform Coding

For rapid transmission or fast browsing from a database, a coarse signal approximation should be quickly provided, and progressively enhanced as more bits are transmitted. Embedded coders offer this flexibility by grouping the bits in order of significance. The decomposition coefficients are sorted and the first bits of the largest coefficients are sent first. A signal approximation can be reconstructed at any time from the bits already transmitted.

Embedded coders store geometric bit planes and can thus take advantage of any prior information about the location of large versus small coefficients. Such prior information is available for natural images decomposed on wavelet bases. Section 10.5.2 explains how JPEG-2000 uses this prior information to optimize wavelet coefficients coding.

Embedding

The decomposition coefficients $f_{\mathcal{B}}[m] = \langle f, g_m \rangle$ are partially ordered by grouping them in index sets Θ_k defined for any $k \in \mathbb{Z}$ by

$$\Theta_k = \{m \,:\, 2^k \leqslant |f_{\mathcal{B}}[m]| < 2^{k+1}\} = \Lambda_{2^k} - \Lambda_{2^{k+1}},$$

which are the difference between two significance maps for twice-larger quantization steps. The set Θ_k is coded with a binary significance map $b_k[m]$:

$$b_k[m] = \begin{cases} 0 & \text{if } m \notin \Theta_k \\ 1 & \text{if } m \in \Theta_k. \end{cases} \tag{10.63}$$

An embedded algorithm quantizes $f_{\mathcal{B}}[m]$ uniformly with a quantization step $\Delta = 2^n$ that is progressively reduced. Let $m \in \Theta_k$ with $k \geqslant n$. The amplitude $|Q(f_{\mathcal{B}}[m])|$ of the quantized number is represented in base 2 by a binary string with nonzero digits between bit k and bit n. Bit k is necessarily 1 because $2^k \leqslant |Q(f_{\mathcal{B}}[m])| < 2^{k+1}$. Thus, $k - n$ bits are sufficient to specify this amplitude, to which is added 1 bit for the sign.

The embedded coding is initiated with the largest quantization step that produces at least one nonzero quantized coefficient. Each coding iteration with a reduced quantization step 2^n is called a *bit plane coding pass*. In the loop, to reduce the quantization step from 2^{n+1} to 2^n, the algorithm first codes the significance map $b_n[m]$. It then codes the sign of $f_{\mathcal{B}}[m]$ for $m \in \Theta_n$. Afterwards, the code stores the nth bit of all amplitudes $|Q(f_{\mathcal{B}}[m])|$ for $m \in \Theta_k$ with $k > n$. If necessary, the coding

precision is improved by decreasing n and continuing the encoding. The different steps of the algorithm can be summarized as follows [422]:

1. *Initialization*: Store the index n of the first nonempty set Θ_n:
$$n = \left\lfloor \sup_m \log_2 |f_{\mathcal{B}}[m]| \right\rfloor. \tag{10.64}$$
2. *Significance coding*: Store the significance map $b_n[m]$ for $m \notin \Theta_{n+1}$.
3. *Sign coding*: Code the sign of $f_{\mathcal{B}}[m]$ for $m \in \Theta_n$.
4. *Quantization refinement*: Store the nth bit of all coefficients $|f_{\mathcal{B}}[m]| > 2^{n+1}$. These are coefficients that belong to some set Θ_k for $k > n$, having a position already stored. Their nth bit is stored in the order in which its position was recorded in the previous passes.
5. *Precision refinement*: Decrease n by 1 and go to step 2.

This algorithm may be stopped at any time in the loop, providing a code for any specified number of bits. The decoder then restores the significant coefficients up to a precision $\Delta = 2^n$. In general, only part of the coefficients are coded with a precision 2^n. Valid truncation points of a bit stream correspond to the end of the quantization refinement step for a given pass, so that all coefficients are coded with the same precision.

Distortion Rate

The distortion rate is analyzed when the algorithm is stopped at step 4. All coefficients above $\Delta = 2^n$ are uniformly quantized with a bin size $\Delta = 2^n$. The zero-quantization bin $[-\Delta, \Delta]$ is therefore twice as big as the other quantization bins, which improves coder efficiency as previously explained.

Once the algorithm stops, we denote by M the number of significant coefficients above $\Delta = 2^n$. The total number of bits of the embedded code is

$$R = R_0^e + R_1^e,$$

where R_0^e is the number of bits needed to code all significance maps $b_k[m]$ for $k \geqslant n$, and R_1^e the number of bits used to code the amplitude of the quantized significant coefficients $Q(f_{\mathcal{B}}[m])$, knowing that $m \in \Theta_k$ for $k > n$.

To appreciate the efficiency of this embedding strategy, let us compare the bit budget $R_0^e + R_1^e$ to the number of bits $R_0 + R_1$ used by the direct transform code from Section 10.4.1. The value R_0 is the number of bits that code the overall significance map

$$b[m] = \begin{cases} 0 & \text{if } |f_{\mathcal{B}}[m]| \leqslant \Delta \\ 1 & \text{if } |f_{\mathcal{B}}[m]| > \Delta \end{cases} \tag{10.65}$$

and R_1 is the number of bits that code the quantized significant coefficients.

An embedded strategy codes $Q(f_{\mathcal{B}}[m])$ knowing that $m \in \Theta_k$ and thus that $2^k \leqslant |Q(f_{\mathcal{B}}[m])| < 2^{k+1}$, whereas a direct transform code knows only that $|Q(f_{\mathcal{B}}[m])| > \Delta = 2^n$. Thus, fewer bits are needed for embedded codes:

$$R_1^e \leqslant R_1. \tag{10.66}$$

However, this improvement may be offset by the supplement of bits needed to code the significance maps $\{b_k[m]\}_{k>n}$ of the sets $\{\Theta_k\}_{k>n}$. A direct transform code records a single significance map $b[m]$, which specifies $\Lambda_{2^n} = \cup_{k \geqslant n} \Theta_k$. It provides less information and is therefore coded with fewer bits:

$$R_0^e \geqslant R_0. \tag{10.67}$$

An embedded code brings an improvement over a direct transform code if

$$R_0^e + R_1^e \leqslant R_0 + R_1.$$

This happens if there is some prior information about the position of large coefficients $|f_{\mathcal{B}}[m]|$ versus smaller ones. An appropriate coder can then reduce the number of bits needed to encode the partial sorting of all coefficients provided by the significance maps $\{b_k[m]\}_{k>n}$. The use of such prior information produces an overhead of R_0^e relative to R_0 that is smaller than the gain of R_1^e relative to R_1. This is the case for most images coded with embedded transform codes implemented in wavelet bases [422] and for the block cosine I basis [492].

Figure 10.12 compares the PSNR of the SPIHT wavelet–embedded code by Said and Pearlman [422] with the PSNR of the direct wavelet transform code that performs an entropy coding of the significance map and of the significance coefficients, described in Section 10.4.1. For any quantization step, both transform codes yield the same distortion but the embedded code reduces the bit budget:

$$R_0^e + R_1^e \leqslant R_0 + R_1.$$

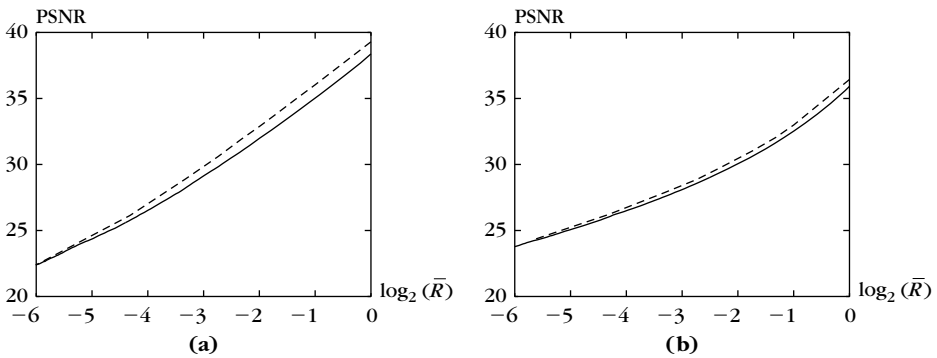


FIGURE 10.12

Comparison of the PSNR obtained with an embedded wavelet transform code (dotted line) and a direct wavelet transform code (solid line): (**a**) Lena image, and (**b**) GoldHill image.

As a consequence, the PSNR curve of the embedded code is a translation to the left of the PSNR of the direct transform code. For a fixed bit budget per pixel $1 \geqslant R/N \geqslant 2^{-8}$, the embedded SPIHT coder gains about 1 db on the Lena image and 1/2 db on the GoldHill image. The GoldHill is an image with more texture. Its wavelet representation is not as sparse as the Lena image, and therefore its distortion rate has a slower decay.

10.5 IMAGE-COMPRESSION STANDARDS

The JPEG image-compression standard is a transform code in an orthogonal block cosine basis described in Section 10.5.1. It is still the most common image standard used by digital cameras and for transmission of photographic images on the Internet and cellular phones. JPEG-2000 is the most recent compression standard performing a transform coding in a wavelet basis, summarized in Section 10.5.2. It is mostly used for professional imaging applications.

10.5.1 JPEG Block Cosine Coding

The JPEG image-compression standard [478] is a transform coding in a block cosine I basis. Its implementation is relatively simple, which makes it particularly attractive for consumer products.

Theorem 8.12 proves that the following cosine I family is an orthogonal basis of an image block of $L \times L$ pixels:

$$\left\{ g_{k,j}[n,m] = \lambda_k \lambda_j \frac{2}{L} \cos\left[\frac{k\pi}{L}\left(n+\frac{1}{2}\right)\right] \cos\left[\frac{j\pi}{L}\left(m+\frac{1}{2}\right)\right] \right\}_{0 \leqslant k,j < L} \tag{10.68}$$

with

$$\lambda_p = \begin{cases} 1/\sqrt{2} & \text{if } p = 0 \\ 1 & \text{otherwise.} \end{cases} \tag{10.69}$$

In the JPEG standard, images of N pixels are divided in $N/64$ blocks of 8×8 pixels. Each image block is expanded in this separable cosine basis with a fast separable DCT-I transform.

JPEG quantizes the block cosine coefficients uniformly. In each block of 64 pixels, a significance map gives the position of zero versus nonzero quantized coefficients. Lower-frequency coefficients are located in the upper right of each block, whereas high-frequency coefficients are in the lower right, as illustrated in Figure 10.13. Many image blocks have significant coefficients only at low frequencies and thus in the upper left of each block. To take advantage of this prior knowledge, JPEG codes the significance map with a run-length code. Each block of 64 coefficients is scanned in zig-zag order as indicated in Figure 10.13. In this scanning order, JPEG registers the size of the successive runs of coefficients quantized to zero, which are efficiently coded together with the values of the following nonzero quantized coefficients.

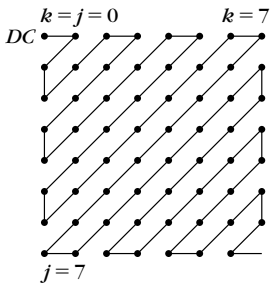


FIGURE 10.13

A block of 64 cosine coefficients has the zero-frequency (DC) coefficient at the upper left. The run-length makes a zig-zag scan from low to high frequencies.

Insignificant high-frequency coefficients often produce a long sequence of zeros at the end of the block, which is coded with an end-of-block (EOB) symbol.

In each block i, there is one cosine vector $g^i_{0,0}[n, m]$ of frequency zero, which is equal to $1/8$ over the block and 0 outside. The inner product $\langle f, g^i_{0,0} \rangle$ is proportional to the average of the image over the block. Let $DC^i = Q(\langle f, g^i_{0,0} \rangle)$ be the quantized zero-frequency coefficient. Since the blocks are small, these averages are often close for adjacent blocks, and JPEG codes the differences $DC^i - DC^{i-1}$.

Weighted Quantization

Our visual sensitivity depends on the frequency of the image content. We are typically less sensitive to high-frequency oscillatory patterns than to low-frequency variations. To minimize the visual degradation of the coded images, JPEG performs a quantization with intervals that are proportional to weights specified in a table, which is not imposed by the standard. This is equivalent to optimizing a weighted mean-square error (10.34). Table 10.1 is an example of an 8×8 weight matrix that is used in JPEG [478]. The weights at the lowest frequencies, corresponding to the upper left portion of Table 10.1, are roughly 10 times smaller than at the highest frequencies, corresponding to the bottom right portion.

Distortion Rate

At 0.25 to 0.5 bit/pixel, the quality of JPEG images is moderate. At 0.2 bit/pixel, Figure 10.14 shows that there are blocking effects due to the discontinuities of the square windows. At 0.75 to 1 bit/pixel, images coded with the JPEG standard are of excellent quality. Above 1 bit/pixel, the visual image quality is perfect. The JPEG standard is often used for $\bar{R} \in [0.5, 1]$.

At low bit rates, the artifacts at the block borders are reduced by replacing the block cosine basis by a local cosine basis [40, 87], designed in Section 8.4.4. If the image is smooth over a block, a local cosine basis creates lower-amplitude, high-frequency coefficients, which slightly improves the coder performance. The quantization errors for smoothly overlapping windows also produce more regular

Table 10.1 Matrix of Weights $w_{k,j}$

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	108	103	77
24	35	55	64	81	194	113	92
49	64	78	87	103	121	120	101
72	92	95	98	121	100	103	99

Note: These weights are used to quantize the block cosine coefficient corresponding to each cosine vector $g_{k,j}$ [69]. The order is the same as in Figure 10.13.

gray-level image fluctuations at the block borders. However, the improvement has not been significant enough to motivate replacing the block cosine basis by a local cosine basis in the JPEG standard.

Implementation of JPEG

The baseline JPEG standard [478] uses an intermediate representation that combines run-length and amplitude values. In each block, the 63 (nonzero frequency) quantized coefficients indicated in Figure 10.13 are integers that are scanned in zig-zag order. A JPEG code is a succession of symbols $S_1 = (L, B)$ of 8 bits followed by symbols S_2. The L variable is the length of a consecutive run of zeros, coded on 4 bits. Thus, its value is limited to the interval $[0, 15]$. Actual zero-runs can have a length greater than 15. The symbol $S_1 = (15, 0)$ is interpreted as a run length of size 16 followed by another run length. When the run of zeros includes the last 63rd coefficient of the block, a special EOB symbol $S_1 = (0, 0)$ is used, which terminates the coding of the block. For high compression rates, the last run of zeros may be very long. The EOB symbol stops the coding at the beginning of this last run of zeros.

The B variable of S_1 is coded on 4 bits and gives the number of bits used to code the value of the next nonzero coefficient. Since the image gray-level values are in the interval $[0, 2^8]$, one can verify that the amplitude of the block cosine coefficients remains in $[-2^{10}, 2^{10} - 1]$. For any integers in this interval, Table 10.2 gives the number of bits used by the code. For example, 70 is coded on $B = 7$ bits. There are 2^7 different numbers that are coded with 7 bits. If B is nonzero, after the symbol S_1, the symbol S_2 of length B specifies the amplitude of the following nonzero coefficient. This variable-length code is a simplified entropy code. High-amplitude coefficients appear less often and are thus coded with more bits.

For DC coefficients (zero frequency), the differential values $DC^i - DC^{i-1}$ remain in the interval $[-2^{11}, 2^{11} - 1]$. They are also coded with a succession of two symbols.

FIGURE 10.14

Image compression with JPEG: left column, 0.5 bit/pixel; right column, 0.2 bit/pixel.

Table 10.2 The Value of Coefficients Coded on B Bits

B	Range of Values
1	$-1, 1$
2	$-3, -2, 2, 3$
3	$-7 \ldots -4, 4 \ldots 7$
4	$-15 \ldots -8, 8 \ldots 15$
5	$-31 \ldots -16, 16 \ldots 31$
6	$-63 \ldots -32, 32 \ldots 63$
7	$-127 \ldots -64, 64 \ldots 127$
8	$-255 \ldots -128, 128 \ldots 255$
9	$-511 \ldots -256, 256 \ldots 511$
10	$-1023 \ldots -512, 512 \ldots 1023$

Note: The values belong to sets of 2^B values that are indicated in the second column.

In this case, S_1 is reduced to the variable B that gives the number of bits of the next symbol S_2, which codes $DC^i - DC^{i-1}$.

For both DC and the other coefficients, the S_1 symbols are encoded with a Huffman entropy code. JPEG does not impose the Huffman tables, which may vary depending on the type of image. An arithmetic entropy code can also be used. For coefficients that are not zero frequency, the L and B variables are lumped together because their values are often correlated, and the entropy code of S_1 takes advantage of this correlation.

10.5.2 JPEG-2000 Wavelet Coding

The JPEG-2000 image-compression standard is a transform code in a wavelet basis. It introduces typically less distortions than JPEG but this improvement is moderate above 1 bit/pixel. At low bit rates, JPEG-2000 degrades more progressively than JPEG. JPEG-2000 is implemented with an embedded algorithm that provides scalable codes for progressive transmission. Region of interest can also be defined to improve the resolution on specific image parts. Yet, the algorithmic complexity overhead of the JPEG-2000 algorithm has mostly limited its applications to professional image processing, such as medical imaging, professional photography, or digital cinema.

Although mostly inspired by the EBCOT algorithm by Taubman and Marcellin [65], JPEG-2000 is the result of several years of research to optimize wavelet image codes. It gives elegant solutions to key issues of wavelet image coding that will be reviewed together with the description of the standard. Taubman and Marcellin's

book [65] explains all the details. JPEG-2000 brings an improvement of typically more than 1 db relatively to an adaptive arithmetic entropy coding of wavelet coefficients, described in Section 10.4.1. Good visual-quality images are obtained in Figure 10.15 with 0.2 bit/pixel, which considerably improves the JPEG compression results shown in Figure 10.14. At 0.05 bit/pixel the JPEG-2000 recovers a decent approximation, which is not possible with JPEG.

Choice of Wavelet

To optimize the transform code one must choose a wavelet basis that produces as many zero-quantized coefficients as possible. A two-dimensional separable wavelet basis is constructed from a one-dimensional wavelet basis generated by a mother wavelet ψ. The wavelet choice does not modify the asymptotic behavior of the distortion rate (10.61) but it influences the multiplicative constant. Three criteria may influence the choice of ψ: number of vanishing moments, support size, and regularity.

High-amplitude coefficients occur when the supports of the wavelets overlap a brutal transition like an edge. The number of high-amplitude wavelet coefficients created by an edge is proportional to the width of the wavelet support, which should thus be as small as possible. For smooth regions, wavelet coefficients are small at fine scales if the wavelet has enough vanishing moments to take advantage of the image regularity. However, Theorem 7.9 shows that the support size of ψ increases proportionally to the number of vanishing moments. The choice of an optimal wavelet is therefore a trade-off between the number of vanishing moments and support size.

The wavelet regularity is important for reducing the visibility of artifacts. A quantization error adds a wavelet multiplied by the amplitude of the quantized error to the image. If the wavelet is irregular, the artifact is more visible because it looks like an edge or a texture patch [88]. This is the case for Haar wavelets. Continuously differentiable wavelets produce errors that are less visible, but more regularity often does not improve visual quality.

To avoid creating large-amplitude coefficients at the image border, it is best to use the folding technique from Section 7.5.2, which is much more efficient than the periodic extension from Section 7.5.1. However, it requires using wavelets that are symmetric or antisymmetric. Besides Haar, there is no symmetric or antisymmetric wavelet of compact support that generates an orthonormal basis. Biorthogonal wavelet bases that are nearly orthogonal can be constructed with symmetric or antisymmetric wavelets. Therefore, they are used more often for image compression.

Overall, many numerical studies have shown that the symmetric 9/7 biorthogonal wavelets in Figure 7.15 give the best distortion rate performance for wavelet image-transform codes. They provide an appropriate trade-off between the vanishing moments, support, and regularity requirements. This biorthogonal wavelet basis is nearly orthogonal and thus introduces no numerical instability. They have an efficient lifting implementation, described in Section 7.8.5. JPEG-2000 offers the choice to use the symmetric 5/3 biorthogonal wavelets, which can be implemented

FIGURE 10.15

JPEG-2000 transform coding: left column, 0.2 bit/pixel; right column, 0.05 bit/pixel.

with fewer lifting operations and exact integer operations. The 5/3 wavelet coefficients can be coded as scaled integers and thus can be restored exactly if the quantization step is sufficiently small. This provides a lossless coding mode to JPEG-2000, which means a coding with no error. Lossless coding yields much smaller compression ratios, typically of about 1.7 for most images [65].

Intra- and Cross-Scale Correlation

The significance maps in Figure 10.10 show that significant coefficients tend to be aggregated along contours or in textured regions. Indeed, wavelet coefficients have a large amplitude where the signal has sharp transitions. At each scale and for each direction, a wavelet image coder can take advantage of the correlation between neighbor wavelet coefficient amplitude, induced by the geometric image regularity. This was not done by the wavelet coder from Section 10.4.1, which makes a binary encoding of each coefficient independently from its neighbors. Taking advantage of this intrascale amplitude correlation is an important source of improvement for JPEG-2000.

Figure 10.10 also shows that wavelet coefficient amplitudes are often correlated across scales. If a wavelet coefficient is large and thus significant, the coarser scale coefficient located at the same position is also often significant. Indeed, the wavelet coefficient amplitude often increases when the scale increases. If an image f is uniformly Lipschitz α in the neighborhood of (x_0, y_0), then (6.58) proves that for wavelets $\psi^l_{j,p,q}$ located in this neighborhood, there exists $A \geqslant 0$ such that

$$|\langle f, \psi^l_{j,p,q} \rangle| \leqslant A\, 2^{j(\alpha+1)}.$$

The worst singularities are often discontinuities, so $\alpha \geqslant 0$. This means that in the neighborhood of singularities without oscillations, the amplitude of wavelet coefficients decreases when the scale 2^j decreases. This property is not always valid, in particular for oscillatory patterns. High-frequency oscillations create coefficients at large scales 2^j that are typically smaller than at the fine scale that matches the period of oscillation.

To take advantage of such correlations across scales, wavelet zero-trees have been introduced by Lewis and Knowles [348]. Shapiro [432] used this zero-tree structure to code the embedded significance maps of wavelet coefficients by relating these coefficients across scales with quad-trees. This was further improved by Said and Pearlman [422] with a set partitioning technique. Yet, for general natural images, the coding improvement obtained by algorithms using cross-scale correlation of wavelet coefficient amplitude seems to be marginal compared to approaches that concentrate on intrascale correlation due to geometric structures. This approach was, therefore, not retained by the JPEG-2000 expert group.

Weighted Quantization and Regions of Interest

Visual distortions introduced by quantization errors of wavelet coefficients depend on the scale 2^j. Errors at large scales are more visible than at fine scales [481]. This can be taken into account by quantizing the wavelet coefficients with intervals

$\Delta_j = \Delta\, w_j$ that depend on the scale 2^j. For $\bar{R} \leqslant 1$ bit/pixel, $w_j = 2^{-j}$ is appropriate for the three finest scales. The distortion in (10.34) shows that choosing such weights is equivalent to minimizing a weighted mean-square error.

Such a weighted quantization is implemented like in (10.35) by quantizing weighted wavelet coefficients $f_{\mathcal{B}}[m]/w_j$ with a uniform quantizer. The weights are inverted during the decoding process. JPEG-2000 supports a general weighting scheme that codes weighted coefficients $w[m]f_{\mathcal{B}}[m]$ where $w[m]$ can be designed to emphasize some region of interest $\Omega \subset [0, 1]^2$ in the image. The weights are set to $w[m] = w > 1$ for the wavelet coefficients $f_{\mathcal{B}}[m] = \langle f, \psi^I_{j,p,q} \rangle$ where the support of $\psi^I_{j,p,q}$ intersects Ω. As a result, the wavelet coefficients inside Ω are given a higher priority during the coding stage, and the region Ω is coded first within the compressed stream. This provides a mechanism to more precisely code regions of interest in images—for example, a face in a crowd.

Overview of the JPEG-2000 Coder

The JPEG-2000 compression standard [65] implements a generic embedded coder, described in Section 10.4.2, and takes advantage of the intrascale dependencies between wavelet coefficients of natural images. The three primitive operations of the algorithm, which code the significance, the sign, and the amplitude refinement, are implemented with a binary adaptive arithmetic coder. This coder exploits the aggregation of large-amplitude wavelet coefficients by creating a small number of context tokens that depend on the neighbors of each coefficient. The coding of a binary symbol uses a conditional probability that depends on the context value of this coefficient.

Wavelet coefficients are subdivided in squares (code blocks) that are coded independently. Each code block S_k is a square of $L \times L$ coefficients $\{f_{\mathcal{B}}[m]\}_{m \in S_k}$, with typical sizes $L = 32$ or $L = 64$. The coding of each code block S_k follows the generic embedded coder detailed in Section 10.4.2 and generates a binary stream c^k. This stream is composed of substreams $c^k = (c^k_1, c^k_2, \ldots)$ where each c^k_n corresponds to the bit plane of the nth bit. The whole set of substreams $\{c^k_n\}_{k,n}$ is reorganized in a global embedded binary stream c that minimizes the rate distortion for any bit budget. The square segmentation improves the scalability of the bit stream, since the generated code can be truncated optimally at a very large number of points in the stream. It also provides more efficient, parallel, and memory friendly implementation of the wavelet coder. Figure 10.16 illustrates the successive coding steps of JPEG-2000.

Conditional Coding of Bit Planes

The coefficients $f_{\mathcal{B}}[m]$ in each code block are processed sequentially using an ordering of the positions $m \in S_k$. The specific choice of ordering used in the JPEG-2000 standard is depicted in Figure 10.17. It scans each square in bands of size $4 \times L$ coefficients.

The conditional coding of coefficients uses an instantaneous significance $\sigma[m]$ for each bit plane n. If the coefficient has not yet been processed, then $\sigma[m]$ carries

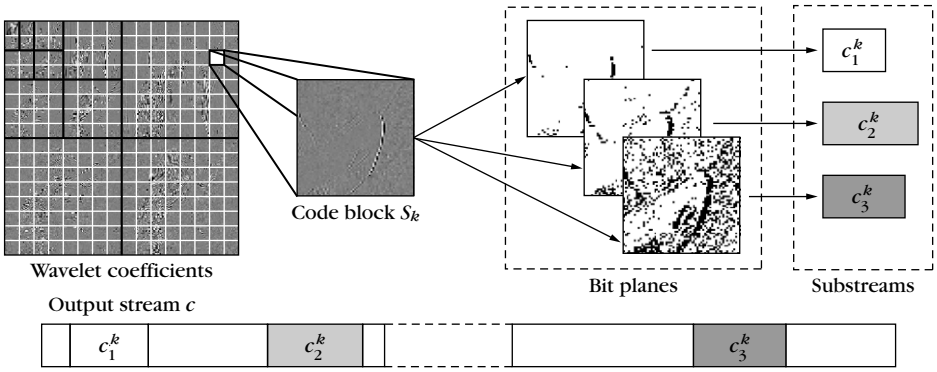


FIGURE 10.16

Overview of JPEG-2000 compression process.

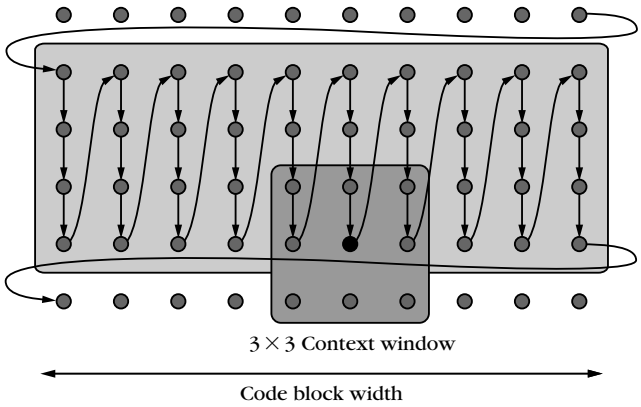


FIGURE 10.17

JPEG-2000 scans square blocks of $L \times L$ wavelet coefficients, as bands of size $4 \times L$.

the value of the previous pass, and $\sigma[m] = 0$ if $|f_{\mathcal{B}}[m]| \geqslant 2^{n+1}$, and $\sigma[m] = 1$ otherwise. If the coefficient has already been processed, then $\sigma[m] = 0$ if $|f_{\mathcal{B}}[m]| \geqslant 2^{n}$, and $\sigma[m] = 1$ otherwise. Observe that $\sigma[m]$ can be computed by both the coder and the decoder from the information that has already been processed.

In the following, we explain how the three primitive operations—significance, sign, and amplitude refinement coding—of the generic embedded coder are implemented using a conditional arithmetic coder.

Significance Coding

If the coefficient of index m was insignificant at the previous bit planes, meaning $|f_{\mathcal{B}}[m]| \leqslant 2^{n+1}$, JPEG-2000 encodes the significance bit $b_n[m] \in \{0, 1\}$. JPEG-2000

takes advantage of redundancies between neighboring coefficients in the square S_k. A conditional arithmetic coder stores the value of $b_n[m]$ by using a conditional probability distribution,

$$p_{\varepsilon,\eta} = P\left(b_n[m] = \varepsilon \,|\, \kappa^0[m] = \eta\right) \quad \text{for} \quad \varepsilon \in \{0, 1\} \quad \text{and} \quad \eta \in \{0, \ldots, \kappa^0{}_{\max} - 1\},$$

where the context $\kappa^0[m]$ for a coefficient $f_{\mathcal{B}}[m] = \langle f,\ \psi^l_{j,p,q} \rangle$ at position (p, q) is evaluated by both the compressor and decompressor from a set of fixed rules using the significance $\sigma[(p, q) + \varepsilon]$ with $\varepsilon = (\pm 1, \pm 1)$ in a 3×3 context window around the position (p, q). These rules can be found in [65].

Sign Coding

If the coefficient of index m has been coded as significant, which means that $b_n[m] = 1$, then JPEG-2000 encodes its sign $s[m] = \text{sign}(f_{\mathcal{B}}[m]) \in \{+1, -1\}$ with an adaptive arithmetic coder that also uses a conditional probability:

$$P\left(s[m] = \varepsilon \,|\, \kappa^{\text{s}}[m] = \eta\right) \quad \text{for} \quad \varepsilon \in \{+1, -1\} \quad \text{and} \quad \eta \in \{0, \ldots, \kappa^{\text{s}}{}_{\max} - 1\}.$$

The context depends on the 3×3 neighboring coefficients, which can be significant and positive, significant and negative, or insignificant, thus allowing 81 unique configurations. JPEG-2000 uses a reduced set of only $\kappa^{\text{s}}{}_{\max} = 5$ context values for $\kappa^{\text{s}}[m]$ that are calculated from horizontal and vertical sign agreement indicators [65].

Amplitude Refinement Coding

If the coefficient is already significant from the previous bit plane, which means that $|f_{\mathcal{B}}[m]| > 2^{n+1}$, then JPEG-2000 codes $\gamma_n[m]$, which is the nth bit of $|f_{\mathcal{B}}[m]|$, to refine the amplitude of the quantized coefficients, with an adaptive arithmetic coder using a conditional probability:

$$P\left(\gamma_n[m] = \varepsilon \,|\, \kappa^{\text{a}}[m] = \eta\right) \quad \text{for} \quad \varepsilon \in \{0, 1\} \quad \text{and} \quad \eta \in \{0, \ldots, \kappa^{\text{a}}{}_{\max} - 1\}.$$

Figure 10.9 shows that the histograms of wavelet coefficients of natural images are usually highly peaked around zero. For a small quantized value $Q(|f_{\mathcal{B}}[m]|)$, the event $\gamma_n[m] = 0$ is thus more likely than the event $\gamma_n[m] = 1$. In contrast, for a large quantized value $Q(|f_{\mathcal{B}}[m]|)$, both events have approximately the same probability. JPEG-2000 uses a context $\kappa^{\text{a}}[m]$ that discriminates these two cases by checking whether $|f_{\mathcal{B}}[m]| > 2^{n+2}$ (one already knows that $|f_{\mathcal{B}}[m]| > 2^{n+1}$). This context also takes into account the significance of the neighboring coefficients by using the significance context $\kappa^0[m]$.

Optimal Truncation Points and Substream Packing

The bit plane coding algorithm generates an embedded bit stream c^k for each code block $\{f_{\mathcal{B}}[m]\}_{m \in S_k}$ of wavelet coefficients. Each bit stream c^k has a set of valid truncation points $\{R^k_{n_{\max}}, R^k_{n_{\max}-1}, \ldots\}$ that corresponds to the end of the coding pass for each bit plane. The quantity $R^k_n - R^k_{n+1}$ is the number of bits of the substream c^k_n generated by the coding of the bit plane of index n of the code block S_k. The goal

of the rate distortion optimization is to pack together all the substreams $\{c_n^k\}_{k,n}$ in an optimized order so as to minimize the resulting global distortion when the code is interrupted at a truncation point.

JPEG-2000 offers a mechanism of fractional bit plane coding that further increases the number of these valid truncation points. These fractional coding passes do not code the whole significance map b_n at once, but rather begin by coding only those coefficient positions (p, q) that have at least one significant neighbor $(p, q) + \varepsilon$ such that $\sigma[(p, q) + \varepsilon] = 1$. The sign and magnitude refinement pass are then applied, and afterward the coefficients that have not been processed in the current bit plane are encoded. In the following, we assume a set of valid truncation points $\{R_n^k\}_n$ for each code block S_k.

Assuming an orthogonal wavelet transform, the distortion at the truncation point R_n^k, after the coding of the bit plane of index n, is

$$d_n^k = \sum_{m \in S_k} |Q(f_{\mathcal{B}}[m]) - f_{\mathcal{B}}[m]|^2, \tag{10.70}$$

where Q is the quantizer of bin size 2^n. This distortion can be computed by the encoder from the already coded bit planes of indexes greater than n. JPEG-2000 uses the biorthogonal 9/7 wavelet transform, which is close to being orthogonal, so the distortion computation 10.70, although not exact, is accurate enough in practice.

For a given number of bits R, we must find for each code block S_k an optimized truncation point $R_{n_k}^k$ of index n_k that solves

$$\min_{\{n_k\}_k} \sum_k d_{n_k}^k \quad \text{subject to} \quad \sum_k R_{n_k}^k \leqslant R. \tag{10.71}$$

This optimal set $\{n_k\}_k$ depends on the desired total number of bits R. To obtain an embedded stream c, the rate distortion optimization (10.70) must be computed for an increasing set of $\ell_{\max}$ bit budgets $\{R^{(\ell)}\}_{0 \leqslant \ell < \ell_{\max}}$. For each number of bits $R^{(\ell)}$, (10.70) is solved with $R = R^{(\ell)}$, defining a set $n_k^{(\ell)}$ of optimal truncation points. The final stream c is obtained by successively appending the substreams c_n^k for $n_k^{(\ell-1)} < n \leqslant n_k^{(\ell)}$.

Rate Distortion Optimization

To build the final stream c, the optimization (10.70) is solved for a large number of bit budgets $R = R^{(\ell)}$. As in Section 10.3.1, following a classic distortion rate approach [392, 435], the constraint minimization is replaced by a Lagrangian optimization. The resulting rate distortion Lagrangian over all square blocks S_k is

$$\mathcal{L}\left(\{d_{n_k}^k\}_k, \{R_{n_k}^k\}_k\right) = \sum_k \left(d_{n_k}^k + \lambda R_{n_k}^k\right).$$

A set of indexes $\{n_k\}_k$ that minimizes $\mathcal{L}$ is necessarily a minimizer of (10.70) for a bit budget $R = \sum_k R_{n_k}^k$. The optimization (10.70) is performed by minimizing $\mathcal{L}$ for many values of λ and by retaining the smallest value of λ that guarantees $\sum_k R_{n_k}^k \leqslant R$.

For a fixed λ, to minimize $\mathcal{L}$, an independent minimization is first done over each block S_k of the partial Lagrangian $d_n^k + \lambda R_n^k$. In each block S_k, it gives truncation points that are associated to a λ. These truncation points are then globally recombined across all blocks S_k by ordering them according to their Lagrange variable λ, which provides the global sequence of all truncation points and defines the final embedded JPEG-2000 stream.

10.6 EXERCISES

10.1 [1] Let X be a random variable that takes its values in $\{x_k\}_{1 \leq k \leq 7}$ with probabilities $\{0.49, 0.26, 0.12, 0.04, 0.04, 0.03, 0.02\}$.

(a) Compute the entropy $\mathcal{H}(X)$. Construct a binary Huffman code and calculate the average bit rate R_X.

(b) Suppose that the symbols are coded with digits that may take three values $(-1, 0, 1)$ instead of two as in a bit representation. Variable-length ternary prefix codes can be represented with ternary trees. Extend the Huffman algorithm to compute a ternary prefix code for X that has a minimal average length.

10.2 [1] Let x_1 be the symbol of highest probability of a random variable X, and l_1 the length of its binary word in a Huffman code. Show that if $p_1 > 2/5$, then $l_1 = 1$. Verify that if $p_1 < 1/3$, then $l_1 \geq 2$.

10.3 [1] Let X be a random variable equal to x_1 or x_2 with probabilities $p_1 = 1 - \varepsilon$ and $p_2 = \varepsilon$. Verify that $\mathcal{H}(X)$ converges to 0 when ε goes to 0. Show that the Huffman code has an average number of bits that converges to 1 when ε goes to 0.

10.4 [2] Prove the Huffman code Theorem 10.2.

10.5 [2] Let $\mathcal{H}_0 = (N - M) \log_2(N/N - M) + M \log_2(N/M)$ be the entropy of a binary coding of the position of M significant coefficients among N as in (10.49). Show that $\mathcal{H}_0 \leq \log_2 \binom{N}{M}$ and compute the difference between the two, by using the Stirling formula $\lim_{n \to \infty} (2\pi n)^{-1/2} (n/e)^{-n}\, !n = 1$.

10.6 [2] Let X be a random variable with a probability density $p(x)$. Let Q be a quantizer with quantization bins that are $\{(y_{k-1}, y_k]\}_{1 \leq k \leq K}$.

(a) Prove that $E\{|X - Q(X)|^2\}$ is minimum if and only if

$$Q(x) = x_k = \frac{\int_{y_{k-1}}^{y_k} x\, p(x)\, dx}{\int_{y_{k-1}}^{y_k} p(x)\, dx} \quad \text{for } x \in (y_{k-1}, y_k].$$

(b) Suppose that $p(x)$ is a Gaussian with variance σ^2. Find x_0 and x_1 for a "1 bit" quantizer defined by $y_0 = -\infty$, $y_1 = 0$, and $y_2 = +\infty$.

10.7 [1] Consider a pulse code modulation that quantizes each sample of a Gaussian random vector $F[n]$ and codes it with an entropy code that uses the same number of bits for each n. If the high-resolution quantization hypothesis is satisfied, prove that the distortion rate is

$$d(\bar{R}) = \frac{\pi e}{6} E\{\|F\|^2\} 2^{-2\bar{R}}.$$

10.8 [3] Let $d = \sum_{m=0}^{N-1} d_m$ be the total distortion of a transform code. We suppose that the distortion rate $d_m(r)$ for coding the mth coefficient is convex. Let $R = \sum_{m=0}^{N-1} R_m$ be the total number of bits.

(a) Prove with the distortion rate Lagrangian that there exists a unique bit allocation that minimizes $d(R)$ for R fixed, and that it satisfies $\frac{\partial d_m(R_m)}{\partial r} = -\lambda$ where λ is a constant that depends on R.

(b) To impose that each R_m is a positive integer, we use a greedy iterative algorithm that allocates the bits one by one. Let $\{R_{m,p}\}_{0 \leq m < N}$ be the bit allocation after p iterations, which means that a total of p bits have been allocated. The next bit is added to $R_{k,p}$ such that

$$\left| \frac{\partial d_k(R_{k,p})}{\partial r} \right| = \max_{0 \leq m < N} \left| \frac{\partial d_m(R_{m,p})}{\partial r} \right|.$$

Justify this strategy. Prove that this algorithm gives an optimal solution if all curves $d_m(r)$ are convex and if $d_m(n+1) - d_m(n) \approx \frac{\partial d_m(n)}{\partial r}$ for all $n \in \mathbb{N}$.

10.9 [2] Let $X[m]$ be a binary first-order Markov chain, which is specified by the transition probabilities $p_{01} = \Pr\{X[m] = 1 \mid X[m-1] = 0\}$, $p_{00} = 1 - p_{01}$, $p_{10} = \Pr\{X[m] = 0 \mid X[m-1] = 1\}$, and $p_{11} = 1 - p_{10}$.

(a) Prove that $p_0 = \Pr\{X[m] = 0\} = p_{10}/(p_{10} + p_{01})$ and that $p_1 = \Pr\{X[m] = 1\} = p_{01}/(p_{10} + p_{01})$.

(b) A run-length code records the length Z of successive runs of 0 values of $X[m]$ and the length I of successive runs of 1. Show that if Z and I are entropy coded, the average number of bits per sample of the run-length code, denoted $\bar{R}$, satisfies

$$\bar{R} \geqslant \bar{R}_{\min} = p_0 \frac{\mathcal{H}(Z)}{E\{Z\}} + p_1 \frac{\mathcal{H}(I)}{E\{I\}}.$$

(c) Let $\mathcal{H}_0 = -p_{01} \log_2 p_{01} - (1 - p_{01}) \log_2 (1 - p_{01})$ and $\mathcal{H}_1 = -p_{10} \log_2 p_{10} - (1 - p_{10}) \log_2 (1 - p_{10})$. Prove that

$$\bar{R}_{\min} = \mathcal{H}(X) = p_0 \mathcal{H}_0 + p_1 \mathcal{H}_1,$$

which is the average information gained by moving one step ahead in the Markov chain.

(d) Suppose that the binary significance map of the transform code of a signal of size N is a realization of a first-order Markov chain. We denote

$\alpha = 1/E\{Z\} + 1/E\{I\}$. Let M be the number of significant coefficients (equal to 1). If $M \ll N$, then show that

$$\bar{R}_{\min} \approx \frac{M}{N}\left(\alpha \log_2 \frac{N}{M} + \beta\right) \tag{10.72}$$

with $\beta = \alpha \log_2 e - 2\alpha \log_2 \alpha - (1-\alpha)\log_2(1-\alpha)$.

(e) Implement a run-length code for the binary significance maps of wavelet image coefficients $d_j^l[n, m] = \langle f, \psi_{j,n,m}^l \rangle$ for j and l fixed. See whether (10.72) approximates the bit rate $\bar{R}$ calculated numerically as a function of N/M for the Lena and Barbara images. How does α vary depending on the scale 2^j and the orientation $l = 1, 2, 3$?

10.10 [4] Implement a transform code in a block cosine basis with an arithmetic code and with a local cosine transform over blocks of the same size. Compare the compression rates in DCT-I and local cosine bases, as well as the visual image quality for $\bar{R} \in [0.2, 1]$.

10.11 [4] Implement a wavelet transform code for color images. Transform the red, green, and blue channels in the color Karhunen-Loève basis calculated in Exercise 9.2 or with any standard color-coordinate system such as Y, U, V. Perform a transform code in a wavelet basis with the multichannel decomposition (12.157), which uses the same approximation support for all color channels. Use an arithmetic coder to binary encode together the three color coordinates of wavelet coefficients. Compare numerically the distortion rate of the resulting algorithm with the distortion rate of a wavelet transform code applied to each color channel independently.

10.12 [4] Develop a video compression algorithm in a three-dimensional wavelet basis [474]. In the time direction, choose a Haar wavelet in order to minimize the coding delay. This yields zero coefficients at locations where there is no movement in the image sequence. Implement a separable three-dimensional wavelet transform and an arithmetic coding of quantized coefficients. Compare the compression result with an MPEG-2 motion-compensated compression in a DCT basis.