HIGH-RESOLUTION STILL PICTURE COMPRESSION

Mladen Victor Wickerhauser

Department of Mathematics, Washington University
Campus Box 1146, St. Louis, Missouri 63130

September 27, 1993

1: Introduction

We shall consider the problem of storing, transmitting, and manipulating digital electronic images. Because of the file sizes involved, transmitting images will always consume large amounts of bandwidth, and storing images will always require hefty resources. Because of the large number N of pixels in a high resolution image, manipulation of digital images is infeasible without low-complexity algorithms, i.e., $O(N)$ or $O(N\log(N))$. Our goal will be to describe some new methods which are firmly grounded in harmonic analysis and the mathematical theory of function spaces, which promise to combine effective image compression with low-complexity image processing. We shall take a broad perspective, but we shall also compare specific new algorithms to the state of the art.

Roughly speaking, most image compression algorithms split into three parts: invertible transformation, lossy quantization or rank-reduction, and entropy coding (or redundancy removal). There are a few algorithms which differ fundamentally from this scheme, e.g., the collage coding algorithm [Barnsley,Sloan], or pure vector quantization of the pixels. The former uses a deep observation that pictures of natural objects exhibit self-similarity at different scales; we prefer to avoid relying on this phenomenon, since our images may not be "natural." The latter uses a complex algorithm to build a super efficient empirical vocabulary to describe an ensemble of images; we prefer to avoid training our algorithm with any sample of images, to avoid the problem of producing a sufficiently large and suitable ensemble.

There has emerged an international standard for picture compression, promulgated by the Joint Photographic Experts Group (JPEG), which is remarkably effective in reducing the size of digitized image files. JPEG is 2-dimensional discrete cosine transform (DCT) coding of 8×8 blocks of pixels, followed by a possibly proprietary quantization scheme on the DCT amplitudes, followed by either Huffman, Lempel–Ziv–Welch or arithmetic coding of the quantized coefficients. It has some drawbacks; for example, several incompatible implementations are allowed under the standard. Also, JPEG degrades ungracefully a high ultrahigh compression ratios, and it makes certain assumptions about the picture that are violated by zooming in or out, or other transformations. It works so well on typical photographs and many other images, however, that it has become the algorithm to beat in most applications. JPEG fails most noticeably on high resolution (i.e., oversampled) data, and on images which must be closely examined by humans or machines.

Alternatives to JPEG have recently appeared, and we shall discuss 3 of these: the fast discrete wavelet transform, the local trigonometric or lapped orthogonal transform, and the best-basis algorithm. These differ in the transform coding step, i.e., instead of DCT they first apply the wavelet transform, lapped orthogonal transform, or wavelet packet transform, possibly followed by a best-basis search. The resulting stream of amplitudes is then quantized and coded to remove redundancy.

Research supported in part by ONR Grant N00014-88-K0020 and by FBI contract A107183

Existing image processing algorithms work on the original pixels or else on the (2-dimensional) Fourier transform of the pixels. If the image has been compressed, it must be uncompressed prior to such processing. Alternatively, we can try to devise algorithms which transform the compressed parameters. If compression is accomplished by retaining only a low-rank approximation to the signal, then we can use more complex algorithms for subsequent processing. To put this idea into practice, we need to retain useful analytic properties such as the large derivatives used in edge detection. These will not be preserved by purely information-theoretic coding such as pure vector quantization, but we can choose transform coding methods whose mathematical properties combine efficient compression with good analytic behavior.

2: Transform Coding Image Compression

A digitally sampled image can only represent a band-limited function, since there is no way of resolving spatial frequencies higher than half the pixel pitch. Band limited functions are smooth; in fact they are entire analytic, which means that at each point they can be differentiated arbitrarily often and the resulting Taylor series converges arbitrarily far away. Since digitally sampled images faithfully reproduce the originals as far as our eyes can tell, we may confidently assume that our images are in fact smooth and well approximated by band-limited functions. Another way of saying this is that adjacent pixels are highly correlated, or that there is a much lower rank description of the image which captures virtually all of the independent features. In transform coding, we seek a basis of these features, in which the the coordinates are less highly correlated or even uncorrelated. These coordinates are then approximated to some precision, and that approximate representation is further passed through a lossless redundancy remover.

The figure below depicts a generic image compression transform coder. It embodies a three-step algorithm:

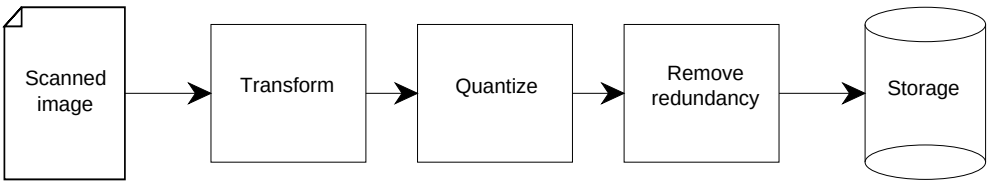

Figure 2-1.

Idealized transform coder.

The first block ("Transform") applies an invertible coordinate transformation to the image. We think of this transformation as implemented in real arithmetic, with enough precision to keep the truncation error below the quantization error introduced by the original sampling. The output of this block will be treated as a stream of real numbers, though in practice we are always limited to a fixed precision.

The second ("Quantize") block replaces the real number coordinates with lower-precision approximations which can be coded in a (small) finite number of digits. If the transform step is effective, then the new coordinates are mostly very small and can be set to zero, while only a few coordinates are large enough to survive. The output of this block is a stream of small integers, most of which are the same (namely 0). If our goal is to reduce the rank of the representation, we can now stop and take only the surviving amplitudes and tag them with some identifiers. If our goal is to reduce the number of bits we must transmit or store, then we should proceed to the next step.

The third block ("Remove redundancy") replaces the stream of small integers with some more efficient alphabet of variable-length characters. In this alphabet the frequently occurring letters (like "0") are represented more compactly than rare letters.

3: Decorrelation by transformation

We will consider six pixel transformations which have proven useful in decorrelating smooth pictures.

3.1: Karhunen–Loève.

Let us now fix an image size—say height H and width W, with $N = H \times W$ pixels—and treat the individual pixels as random variables. Our probability space will consist of some collection of pictures $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$, where M is a big number. The intensity of the nth pixel $S(n)$, $1 \le n \le N$, is a random variable that takes a nonnegative real value for each individual picture $S \in \mathcal{S}$. Nearby pixels in a smooth image are correlated, which means that the value of one pixel conveys information about the likelihood of its neighbors' values. This implies that having transmitted the one pixel value at full expense, we should be able to exploit this correlation to reduce the cost of transmitting the neighboring pixel values. This is done by transforming the picture into a new set of coordinates which are uncorrelated over the collection $\mathcal{S}$, and then transmitting the uncorrelated values.

More precisely, the collection of smooth pictures $\mathcal{S}$ has off-diagonal terms in the autocovariance matrix $A = \big(A(i,j)\big)_{i,j=1}^{N}$ of the pixels in $\mathcal{S}$:

$$A(i,j) = \frac{1}{M} \sum_{m=1}^{M} \hat{S}_m(i) \times \hat{S}_m(j), \qquad (3.1\text{-}1)$$

where $\hat{S}_m = S_m - \frac{1}{M}\sum_m S_m$. A can be diagonalized because it is symmetric (see [Apostol], theorem 5.4, page 120, for a proof of this very general fact). We can write T for the orthogonal matrix that diagonalizes A; then TAT^* is diagonal, and T is called the *Karhunen–Loève transform*, or alternatively the *principal orthogonal decomposition*. The rows of T are the vectors of the Karhunen–Loève basis for the collection S, or equivalently for the matrix A. The number of positive eigenvalues on the diagonal of TAT^* is the actual number of uncorrelated parameters, or degrees of freedom, in the collection of pictures. Each eigenvalue is the variance of its degree of freedom. TS_m is S_m written in these uncorrelated parameters, which is what we should transmit.

Unfortunately, the above method is not practical because of the amount of computation required. For typical pictures, N is in the range 10^4–10^6. To diagonalize A and find T requires $O(N^3)$ operations in the general case. Furthermore, to apply T to each picture requires $O(N^2)$ operations in general. Hence several simplifications are usually made.

3.2: DCT.

For smooth signals, the autocovariance matrix is assumed to be of the form

$$A(i,j) = r^{|i-j|} \qquad (3.2\text{-}2)$$

where r is the adjacent pixel correlation coefficient and is assumed to be just slightly less than 1. The expression $|i-j|$ should be interpreted as $|i_r - j_r| + |i_c - j_c|$, where i_r and i_c are respectively the row and column indices of pixel i, and similarly for j. Experience shows that this is quite close to the truth for small sections of large collections of finely sampled smooth pictures. It is possible to compute the Karhunen–Loève basis exactly in the limit $N \to \infty$: in that case A is the matrix of a two-dimensional convolution with an even function, so it is diagonalized by the two-dimensional discrete cosine transform (DCT). In one dimension, this transform is an inner product with functions such as the one in the figure below:

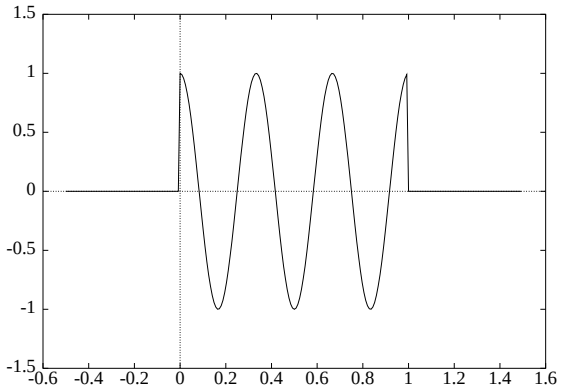

Figure 3.2-2.

Example DCT basis function.

This limit transform can be used instead of the exact Karhunen–Loève basis; it has the added advantage of being rapidly computable via the fast DCT derived from the fast Fourier transform. The Joint Photographic Experts Group (JPEG) algorithm uses this transform and one other simplification. N is limited to 64 by taking 8×8 sub-blocks of the picture. JPEG applies two-dimensional DCT to the sub-blocks, then treats the 64 vectors of amplitudes individually in a manner we will discuss in the next section.

3.3: LCT or LOT.

Rather than use disjoint 8×8 blocks as in JPEG, it is possible to use "lapped" or "localized" (but still orthogonal) discrete cosine functions which are supported on overlapping patches of the picture. These *local cosine transforms* (LCT, as in [Coifman,Meyer]) or *lapped orthogonal transforms* (LOT, as in [Malvar]) are modifications of DCT which attempt to solve the blockiness problem by using smoothly overlapping blocks. This can be done in such a way that the overlapping blocks are still orthogonal, i.e., there is no added redundancy from using amplitudes in more than one block to represent a single pixel. For the smooth blocks to be orthogonal we must use DCT-IV, which is the discrete cosine transform using half-integer grid points and half-integer frequencies. The formulas for the smooth overlapping basis functions in two dimensions are derived from the following formulas in one dimension.

For definiteness we will use a particular symmetric bump function

$$b(x) = \begin{cases} \sin\frac{\pi}{4}(1+\sin\pi x), & \text{if } -\frac{1}{2} < x < \frac{3}{2}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.3-3}$$

This function is symmetric about the value $x = \frac{1}{2}$. It is smooth on $(-\frac{1}{2}, \frac{3}{2})$ with vanishing derivatives at the boundary points, so that it has a continuous derivative on $\mathbf{R}$. Notice that we can modify b to obtain more continuous derivatives by iterating the innermost $\sin\pi x$. Let $b_1(x) = b(x)$ and define

$$b_{n+1}(x) = b_n\left(\tfrac{1}{2}\sin\pi x\right) \tag{3.3-4}$$

Then b_n will have (use L'Hôpital's rule!) at least 2^{n-1} vanishing derivatives at $-\frac{1}{2}$ and $\frac{3}{2}$.

Now consider the interval of integers $I = \{0, 1, 2, \dots, N-1\}$ where $N = 2^n$ is a positive integer power of 2. This can be regarded as the "current block" of N samples in an array; there are previous samples $I' = \{\dots, -2, -1\}$ and future samples $I'' = \{N, N+1, \dots\}$ as well. The lapped orthogonal functions are

mainly supported on I, but they take values on $\{-N/2, \ldots, -1\} \subset I'$ and $\{N, \ldots, N/2-1\} \subset I''$ as well; those are the overlapping parts. For integers $k \in \{0, 1, \ldots, N-1\}$, we can define the function

$$\psi_k(j) = \frac{1}{\sqrt{2N}}\, b\left(\frac{j+\frac{1}{2}}{N}\right) \cos\left(\pi\,(k+\frac{1}{2})\,\left[\frac{j+\frac{1}{2}}{N}\right]\right) \qquad (3.3\text{-}5)$$

Apart from b, these are evidently the basis functions for the so-called DCT-IV transform. The figure below shows one such function, with N chosen large enough so that the smoothness is evident:

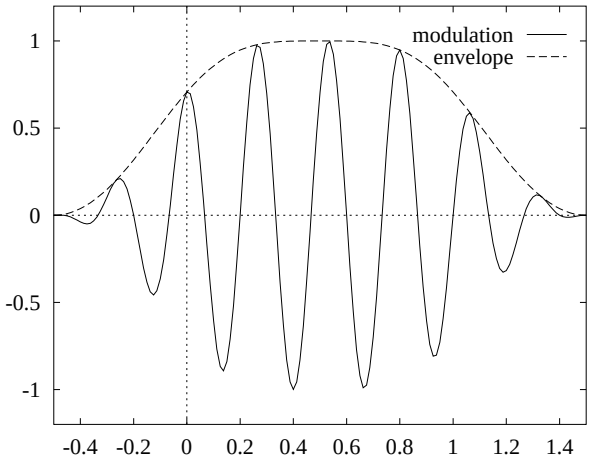

Figure 3.3-3.

Example LCT basis function.

The orthogonality of such functions may be checked by verifying the following equations:

$$\sum_{j=-\frac{N}{2}}^{\frac{3N}{2}-1} \psi_k(j)\psi_{k'}(j) = \begin{cases} 1, & \text{if } k = k', \\ 0, & \text{if } k \neq k'. \end{cases} \qquad (3.3\text{-}6)$$

The chosen window function or "bell" allows cosines on adjacent intervals to overlap while remaining orthogonal. For example, the function $\psi_k(j+N)$ is centered over the range $j \in \{-N, -N+1, \ldots, -1\}$ and overlaps the function $\psi_{k'}(j)$ at values $j \in \{-N/2, -N/2+1, \ldots, N/2-1\}$. Yet these two functions are orthogonal, which may be checked by verifying the equation below:

$$\sum_{j=-\frac{N}{2}}^{\frac{N}{2}-1} \psi_k(j+N)\psi_{k'}(j) = 0, \qquad \text{for all integers } k,\ k'. \qquad (3.3\text{-}7)$$

Of course, rather than calculate inner products with the sequences ψ_k, we can preprocess data so that standard fast DCT-IV algorithms may be used. This may be visualized as "folding" the overlapping parts of the bells back into the interval. This folding can be transposed onto the data, and the result will be disjoint intervals of samples which can be "unfolded" to produce smooth overlapping segments. This is best illustrated by an example. Suppose we wish to fold a smooth function across 0, onto the intervals $\{-N/2, \ldots, -1\}$ and $\{0, 1, \ldots, N/2-1\}$, using the bell b defined above. Then folding replaces the function

$f = f(j)$ with the left and right parts f_{0-} and f_{0+}:

$$\text{(3.3-8)}\qquad \begin{aligned} f_{0-}(j) &= \begin{cases} f(j), & \text{if } j < -N/2, \\ b(\frac{-j-\frac{1}{2}}{N})f(j) - b(\frac{j+\frac{1}{2}}{N})f(-j-1), & \text{if } j \in \{-N/2, \ldots, -1\}, \end{cases} \\ f_{0+}(j) &= \begin{cases} b(\frac{j+\frac{1}{2}}{N})f(j) + b(\frac{-j-\frac{1}{2}}{N})f(-j-1), & \text{if } j \in \{0, 1, \ldots, N/2-1\}, \\ f(j), & \text{if } j \geq N/2. \end{cases} \end{aligned}$$

The symmetry of b allows us to use $b(-x)$ instead of introducing the bell attached to the left interval. This action divides f into two independent functions (the even and odd parts of f) which merge smoothly around the grid point 0. The process is an orthogonal transformation. We can fold the smooth function around the grid point N in a similar manner:

$$\text{(3.3-9)}\qquad \begin{aligned} f_{1-}(j) &= \begin{cases} f(j), & \text{if } j < N/2; \\ b(\frac{-j-\frac{1}{2}}{N} - 1)f(j) - b(\frac{j+\frac{1}{2}}{N} - 1)f(2N-j-1), & \text{if } j \in \{N/2, \ldots, N-1\}, \end{cases} \\ f_{1+}(j) &= \begin{cases} b(\frac{j+\frac{1}{2}}{N} - 1)f(j) + b(\frac{-j-\frac{1}{2}}{N} - 1)f(2N-j-1), & \text{if } j \in \{N, N+1, \ldots, 3N/2-1\}, \\ f(j), & \text{if } j \geq 3N/2. \end{cases} \end{aligned}$$

The new function f_0 defined below is a smooth, independent segment of the original smooth function f, restricted to the interval of values $\{0, 1, \ldots, N-1\}$:

$$\text{(3.3-10)}\qquad f_0(j) = \begin{cases} f_{0+}(j), & \text{if } j \in \{0, 1, \ldots, N/2-1\}, \\ f_{1-}(j), & \text{if } j \in \{N/2, N/2+1, \ldots, N-1\}. \end{cases}$$

We can now apply the N-point DCT-IV transform directly to f_0.

We can likewise define $f_m(j)$ for the values $j \in \{mN, mN+1, \ldots, (m+1)N-1\}$ by the same folding process, which segments a smooth function f into smooth independent blocks. Folding to intervals of different lengths is easily defined as well. We can also generalize to two dimensions by separably folding in x and then in y.

Unfolding reconstructs f from f_{0-} and f_{0+} by the following formulas:

$$\text{(3.3-11)}\qquad f(j) = \begin{cases} b(\frac{-j-\frac{1}{2}}{N})f_{0-}(j) + b(\frac{j+\frac{1}{2}}{N})f_{0+}(-j-1), & \text{if } j \in \{-N/2, \ldots, -1\}, \\ b(\frac{j+\frac{1}{2}}{N})f_{0+}(j) - b(\frac{-j-\frac{1}{2}}{N})f_{0-}(-j-1), & \text{if } j \in \{0, 1, \ldots, N/2-1\}. \end{cases}$$

Composing these relations yields $f(j) = \left[b(\frac{j+1/2}{N})^2 + b(\frac{-j-1/2}{N})^2\right] f(j)$. This equation is verified by the bell b defined in Eq.(3.3-3), for which the sum of the squares is 1.

3.4: Adapted block cosines.

We can also build a library of block LCT bases (or block DCT bases) and search it for the minimum of some cost function. The chosen "best LCT basis" will be a patchwork of different-sized blocks, adapted to different-sized embedded textures in the picture. Again it will be necessary to encode the basis choice together with the amplitudes. A description of two versions of this algorithm and some experiments may be found in [Fang,Seré].

3.5: Subband coding.

A (one-dimensional) signal may be divided into frequency subbands by repeated application of convolution by a pair of digital filters, one high-pass and one low-pass, with mutual orthogonality properties.

Let $\{h_k\}_{k=0}^{M-1}, \{g_k\}_{k=0}^{M-1}$ be two finite sequences, and define two operators H and G as follows:

$$\text{(3.5-12)}\qquad (Hf)_k = \sum_{j=0}^{M-1} h_j f_{j+2k}, \qquad (Gf)_k = \sum_{j=0}^{M-1} g_j f_{j+2k}.$$

H and G are defined on square-summable signal sequences of any length. They are also be defined for periodic sequences of (even) period P, where we simply interpret the index of f as $j + 2k \pmod P$. In that case, the filtered sequences will be periodic with period $P/2$.

The adjoints H^* and G^* of H and G are defined by

$$(H^*f)_k = \sum_{0 \le k-2j < M} h_{k-2j} f_j, \qquad (G^*f)_k = \sum_{0 \le k-2j < M} g_{k-2j} f_j. \tag{3.5-13}$$

H and G are called *(perfect reconstruction) quadrature mirror filters (or QMFs)* if they satisfy a pair of orthogonality conditions:

$$HG^* = GH^* = 0; \qquad H^*H + G^*G = I. \tag{3.5-14}$$

Here I is the identity operator. These conditions translate to restrictions on the sequences $\{h_k\}, \{g_k\}$. Let m_0, m_1 be the bounded periodic functions defined by

$$m_0(\xi) = \sum_{k=0}^{M-1} h_k e^{ik\xi}, \qquad m_1(\xi) = \sum_{k=0}^{M-1} g_k e^{ik\xi}. \tag{3.5-15}$$

Then H, G are quadrature mirror filters if and only if the matrix below is unitary for all ξ:

$$\begin{pmatrix} m_0(\xi) & m_0(\xi+\pi) \\ m_1(\xi) & m_1(\xi+\pi) \end{pmatrix} \tag{3.5-16}$$

This fact is proved in [Daubechies]. QMFs can be obtained by constructing a sequence $\{h_k\}$ with the desired low-pass filter response, and then putting $g_k = (-1)^k h_{M-1-k}$. That reference also contains an algorithm for constructing a family of such $\{h_k\}$, one for each even filter length M.

The frequency response of one particular pair of QMFs ("C30") is depicted below. We have plotted the absolute values of m_0 and m_1, respectively, over one period $[-\pi, \pi]$:

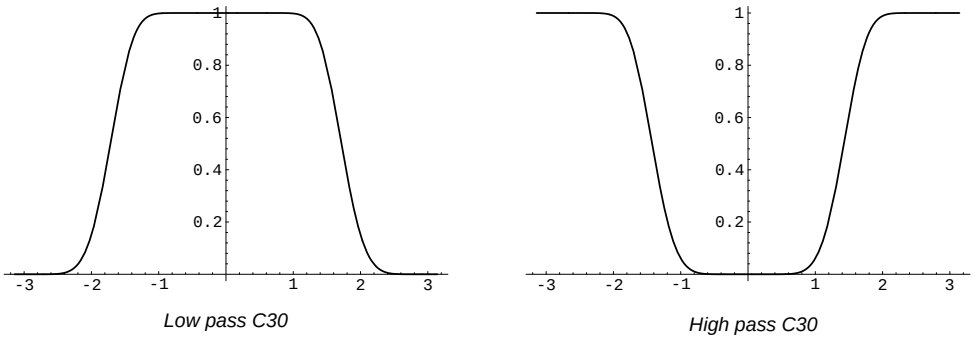

Figure 3.5-4.

Absolute values of m_0 and m_1.

Notice that m_0 attenuates frequencies away from 0, while m_1 attenuates those away from 0.

Below is the traditional block diagram describing the action of a pair of quadrature mirror filters. On the left is convolution and downsampling (by 2); on the right is upsampling (by 2) and adjoint convolution, followed by summing of the components. The broken lines in the middle represent either transmission or storage.

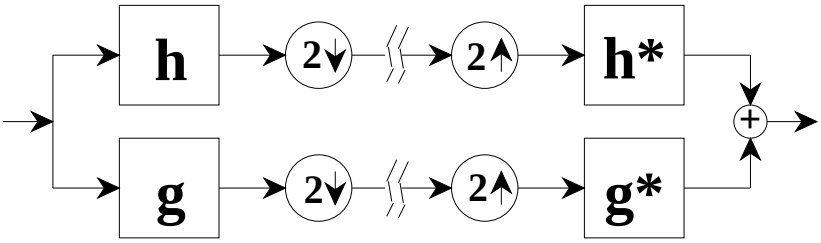

Figure 3.5-5.

Block diagram of subband filtering.

The underlying functions of subband filtering are produced by iterating H^* and G^* until we have enough points. For example, 10 iterations of H^* applied to the sequence $e_0 = \{\dots, 0, 0, 1, 0, 0, \dots\}$ produces a 1024-point approximation to the smooth function whose translates span the lowest-frequency subband. Likewise, a single G^* after 9 iterations of H^* applied to e_0 produces a 1024-point approximation to the next lowest-frequency function. These are distinguished examples; the first is called the "scaling" or "father" function, while the second is called the "wavelet" or "mother" function. They are depicted below for a particular QMF ("C 30," with 30 taps):

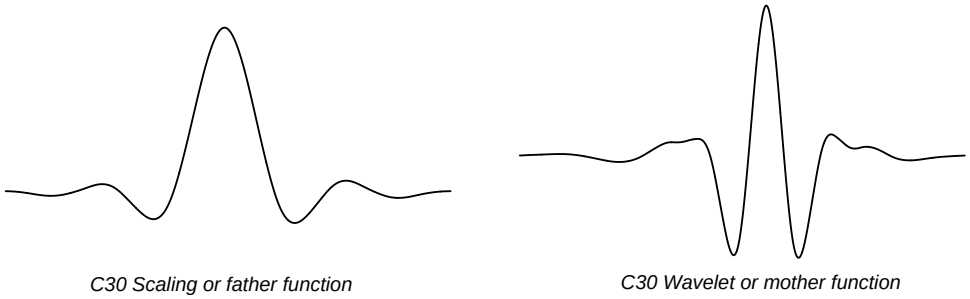

Figure 3.5-6.

Lowest frequency subband basis functions.

Higher-frequency subbands are spanned by functions with more oscillations, which are produced by using G^* earlier in the iteration. The sequence of filters used to generate a function can be converted to an integer in binary notation as follows. Put $F_0^* = H^*$ and $F_1^* = G^*$ in the formula for the function (respectively, put $F_0 = H$ and $F_1 = G$). Then for any pair of integers n and L with $0 \le n < 2^L$ we can write $n = n_0 2^0 + n_1 2^1 + \cdots + n_{L-1} 2^{L-1}$, where $n_i \in \{0, 1\}$ for all $i = 0, 1, \dots, L-1$. To that combination (n, L) we can associate a vector $F_{n_0}^* \circ F_{n_1}^* \circ \cdots \circ F_{n_{L-1}}^* e_0$.

For example, the following functions are 1024-point sequences $H^*G^*(H^*)^8 e_0$ and $G^*G^*(H^*)^8 e_0$, respectively given by $(n, L) = (2, 10)$ and $(n, L) = (3, 10)$:

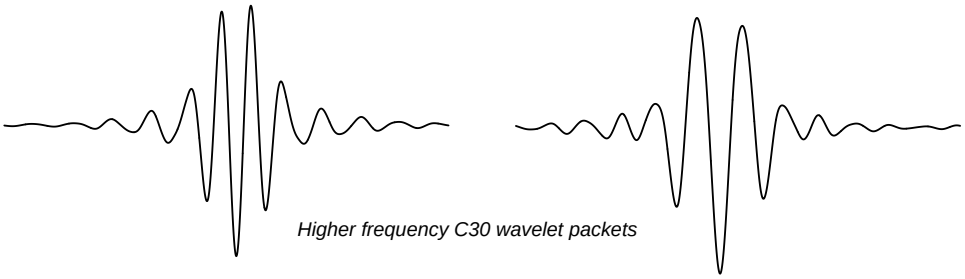

Figure 3.5-7.

Higher-frequency subband basis functions.

It is well known (and shown in [INRIA]) that the number of oscillations of the vector produced in this manner increases with n', where n is the Gray-code permutation of n'. The renumbering $n \to n'$ relates Paley order to sequency order for Walsh functions, and has an analogous effect in the smooth case. This fact can be used to analyze the spectrum of acoustic signals by measuring the amplitudes of wavelet packet coefficients. Acoustic signal compression by wavelet packet and best-basis methods was discussed in [W1].

Now we can define 4 2-dimensional convolution-decimation operators in terms of H and G, namely the tensor products of the pair of quadrature mirror filters:

(3.5-17) $$F_0 \stackrel{\text{def}}{=} H \otimes H, \qquad F_0 v(x,y) = \sum_{i,j} h_i h_j v(i+2x, j+2y)$$

(3.5-18) $$F_1 \stackrel{\text{def}}{=} H \otimes G, \qquad F_1 v(x,y) = \sum_{i,j} h_i g_j v(i+2x, j+2y)$$

(3.5-19) $$F_2 \stackrel{\text{def}}{=} G \otimes H, \qquad F_2 v(x,y) = \sum_{i,j} g_i h_j v(i+2x, j+2y)$$

(3.5-20) $$F_3 \stackrel{\text{def}}{=} G \otimes G, \qquad F_3 v(x,y) = \sum_{i,j} g_i g_j v(i+2x, j+2y)$$

These convolution-decimations have the following adjoints:

(3.5-21) $$F_0^* v(x,y) = \sum_{i,j} h_{x-2i} h_{y-2j} v(i,j)$$

(3.5-22) $$F_1^* v(x,y) = \sum_{i,j} h_{x-2i} g_{y-2j} v(i,j)$$

(3.5-23) $$F_2^* v(x,y) = \sum_{i,j} g_{x-2i} h_{y-2j} v(i,j)$$

(3.5-24) $$F_3^* v(x,y) = \sum_{i,j} g_{x-2i} g_{y-2j} v(i,j)$$

The orthogonality relations for this collection are as follows:

(3.5-25) $$F_n F_m^* = \delta_{nm} I;$$

(3.5-26) $$I = F_0^* F_0 \oplus F_1^* F_1 \oplus F_2^* F_2 \oplus F_3^* F_3.$$

By a "picture" we will mean a finite sequence indexed by two coordinates $S = S(x, y)$. It is convenient to regard pictures as periodic in botyh x and y, though this is not absolutely necessary. For simplicity of implementation, we shall also assume that the x-period (the "width" $N_x = 2^{n_x}$) and the y-period (the "height" $N_y = 2^{n_y}$) are both positive integer powers of 2, so that we can always decimate by two and get an integer period. The space of such pictures may be decomposed into a partially ordered set $\mathbf{W}$ of subspaces $W(n, m)$ called *subbands* (see below), where $m \geq 0$, and $0 \leq n < 4^m$. These are the images of orthogonal projections composed of products of convolution-decimations. Denote the space of $N_x \times N_y$ pictures by $W(0, 0)$ (it is $N_x \times N_y$ dimensional), and define recursively

$$W(4n + i, m + 1) = F_i^* F_i W(n, m) \subset W(0, 0), \qquad \text{for } i = 0, 1, 2, 3. \tag{3.5-27}$$

The orthogonality condition on the QMFs implies that the projections from $W(0, 0)$ onto $W(n, m)$ are orthogonal, i.e., they conserve energy. The subspace $W(n, m)$ is $(N_x 2^{-m}) \times (N_y 2^{-m})$-dimensional. These subspaces may be partially ordered by a relation which we define recursively as follows. We say W is a *precursor* of W' (write $W \prec W'$) if they are equal or if $W' = F^* F W$ for a convolution-decimation F in the set $\{F_0, F_1, F_2, F_3\}$. We also say that $W \prec W'$ if there is a finite sequence $V_1, \ldots, V_s$ of subspaces in $\mathbf{W}$ such that $W \prec V_1 \prec \cdots \prec V_s \prec W'$. This is well defined, since each application of $F^* F$ increases the index m.

Subspaces of a single precursor $W \in \mathbf{W}$ will be called its *descendents*, while the first generation of descendents will naturally be called *children*. By the orthogonality condition,

$$W = F_0^* F_0 W \oplus F_1^* F_1 W \oplus F_2^* F_2 W \oplus F_3^* F_3 W. \tag{3.5-28}$$

The right hand side contains all the children of W.

The subspaces $W(n, m)$ are called *subbands*, and the transform coding method that first transforms a signal into subband coordinates is called *subband coding*. If $S \in W(0, 0)$ is a picture, then its orthogonal projection onto $W(n, m)$ can be computed in the standard coordinates of $W(n, m)$ by the formula $F_{(1)} \ldots F_{(m)} W(0, 0)$, where the particular filters $F_{(1)} \ldots F_{(m)}$ are determined uniquely by n. Therefore we can express in standard coordinates the orthogonal projections of $W(0, 0)$ onto the complete tree of subspaces $\mathbf{W}$ by recursively convolving and decimating with the filters.

The quadrature mirror filters H and G form a partition of unity in the Fourier transform (or wavenumber) space. The same is true for the separable filters F_i. They can be described as nominally dividing the support set of the Fourier transform $\hat{S}$ of the picture into dyadic squares. If the filters were perfectly sharp, then this would be literally true, and the children of W would correspond to the 4 dyadic subsquares one scale smaller. We illustrate this in the figure below.

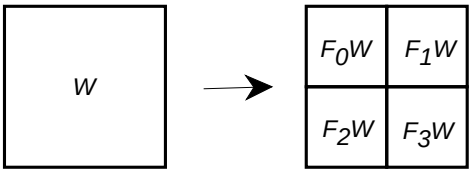

Figure 3.5-8.

Four subband descendents of a picture.

The next figure shows 2 generations of descendents, the complete decomposition of $\mathbf{R}^4 \times \mathbf{R}^4$. The subbands are labelled by the "n" index in $W(n, m)$. Within the dyadic squares are the n-indices of the corresponding

subspaces at that level. If we had started with a picture of $N \times N$ pixels, then we could repeat this decomposition process $\log_2(N)$ times.

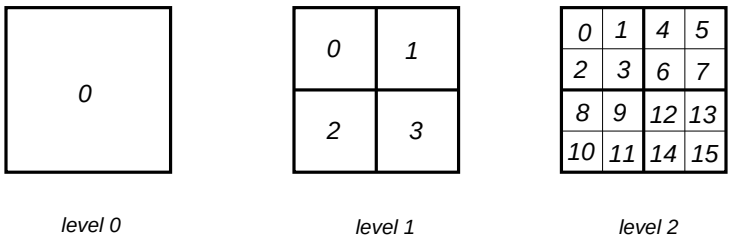

Figure 3.5-9.

Two levels of subband decomposition.

All subbands together form a quadtree, in which each subspace forms a node and directed edges go from precursors to descendents. The orthogonality relation among the subspaces implies that every connected subtree which contains the root $W(0,0)$ corresponds to an orthonormal subband decomposition of the original picture: the subbands correpond to the leaves of the subtree. Having stated this general nonsense result, let us consider specific examples.

The subbands $W(n,m)$ as we define them are in one-to-one correspondence with rectangular regions (in fact squares) in wavenumber space, and the quadtree stacks these regions one on top of the other. We can idealize various orthogonal subband bases as disjoint covers of wavenumber space. A few of these are schematically represented below:

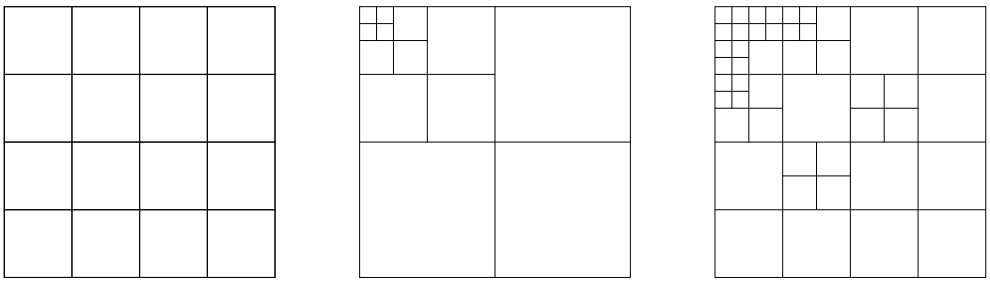

Figure 3.5-10.

Various decompositions into subbands.

The leftmost decomposition is a subband decomposition in which we have all the subbands at a fixed level—in this case, level 2. The subbands are labeled (0,2), (1,2), (2,2), (3,2), ..., (15,2) as in figure 3.5-9 above. The middle decomposition produces 2-dimensional "isotropic" wavelets, i.e., which have the same scale in both the x and y directions. The subbands in this decomposition are labelled by (0,4), (1,4), (2,4), (3,4), (1,3), (2,3), (3,3), (1,2), (2,2), (3,2), (1,1), (2,1), and (3,1).

The rightmost decomposition is an adapted subband basis such as might be discovered by minimizing

some cost function over all subband decompositions. In that particular example, the subspaces are labelled by (0,4), (1,4), (2,4), (3,4), (4,4), (5,4), (6,4), (7,4), (8,4), (9,4), (10,4), (11,4), (3,3), (16,4), (17,4), (18,4), (19,4), (5,3), (6,3), (7,3), (32,4), (33,4), (34,4), (35,4), (9,3), (10,3), (11,3), (3,2), (4,2), (5,2), (24,3), (25,3), (26,3), (27,3), (7,2), (8,2), (36,3), (37,3), (38,3), (39,3) (10,2), (11,2), (12,2), (13,2), (14,2), and (15,2). It is similar to a subband basis which is particularly effective at representing fingerprint images.

3.5.1: Complete subband level. We can divide a picture into patches and code the spatial frequencies of the patches by a decimation-in-space algorithm (subband coding) rather than a decimation-in-frequency algorithm (block DCT). This is computationally equivalent, but has the advantage of using smooth basis functions (if the filters are chosen appropriately) and also greater flexibility, since there are additional degrees of freedom available in the choice of filters. The drawback is poorer time-frequency localization. Subband basis functions spread out in the spatial domain in proportion to their smoothness, and they have rather complicated Fourier transforms.

The subband correlations are computed recursively as depicted below:

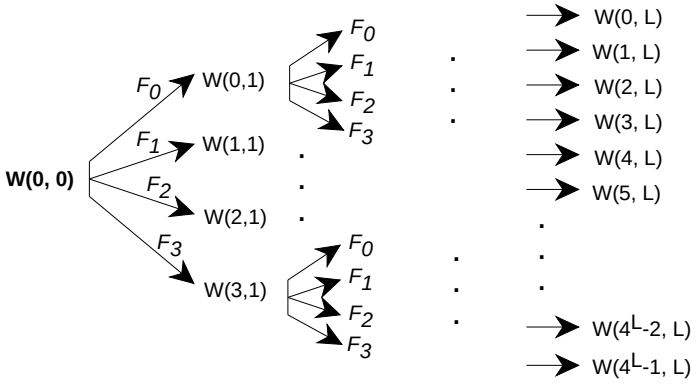

Figure 3.5.1-11.

Recursive filtering to fixed subbands.

This method is popular largely due to its simplicity. The amplitudes within subbands correspond to equal-sized basis functions distinguished only by their spatial frequencies. Hence variable bit-allocation is simple to implement, using the many transform values within a subband as the random variables, and allocating bits to a subband according to its variance. In addition, it is possible to use a simple visibility weighting formula based on the eye's spatial frequency response. See the quantization section below for further details. With these additional steps, subband coding becomes quite competitive with other transform coding compression algorithms.

3.5.2: Wavelets, or multiresolution analysis. We can attempt pixel decorrelation by assuming that the picture looks more or less the same at various different scales. This assumption underlies certain models of vision (see [Marr]), and partially accounts for the natural appearance of fractal computer images. In this case, the decorrelating transformation should do the same thing at each scale of the picture.

We can build an orthonormal basis with this property (in one dimension) by choosing a special "mother function" $\psi = \psi(x)$ and then generating its descendents $\psi_{ab} = \psi_{ab}(x) = 2^{a/2}\psi(2^a x - b)$, with integers a and b. A (separable) two-dimensional wavelet basis can be built by taking products $\psi_{ab}(x)\psi_{cd}(y)$, although recent work by several authors (for example [Cohen,Daubechies], [Madych,Gröchenig], and [Strichartz]) has produced nonseparable multidimensional wavelet bases which could also be used. It is not immediately

obvious that a function ψ with the requisite properties exists. The surprising and fortunate recent discovery of many such functions ([Daubechies], [Mallat], [Meyer] and others) also provided a fast $O(N)$ algorithm to compute the associated wavelet transforms.

The wavelet basis down to level L consists of the elements spanning the subbands $W(1,1)$, $W(2,1)$, $W(3,1)$, $W(1,2)$, $W(2,2)$, $W(3,2)$, ... ,$W(1,L-1)$, $W(2,L-1)$, $W(3,L-1)$, $W(1,L)$, $W(2,L)$, $W(3,L)$ and the largest-scale average $W(0,L)$. The pixel values may be transformed into this basis via the 2-dimensional version of the "pyramid scheme" described in [Mallat]. Graphically, this is the following:

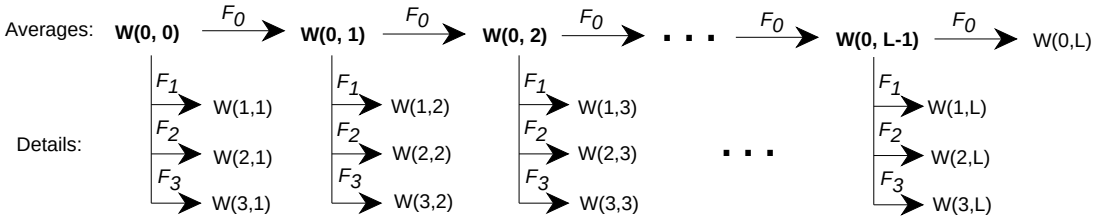

Figure 3.5.2-12.

Two dimensional pyramid scheme.

The pyramid scheme is also called a multiresolution analysis, and has been extensively studied. It provides an algorithm of complexity $O(N)$ for the transformation of an N-pixel picture.

3.5.3: Custom subbands. There may be features which are most efficiently described by expansion into certain subbands. For example, if the image contains textures of a particular size and frequency, then we will find large values in the corresponding subband. This is true for example in fingerprint image compression, where both the large-scale texture of the ridges and the fine-scale texture of the pores contribute large values. The first are not so important for identification since ridges can be easily deformed, while the latter must be preserved very accurately since pore shapes and distributions often provide strong clues for identification. In that case, we can choose to include both the ridge and pore subbands in our transform, de-emphasize the values of the first and amplify the values in the second.

3.6: Adapted subband coding.

Wavelets decorrelate pictures which are close to self-similar. Other subband bases decorrelate pictures composed of overlapping textures on different scales. Some combination is needed for pictures which are close to self-similar but contain embedded patches of textures of various sizes, but it is not clear that any fixed choice of subbands will contain suitable templates. But it is possible to use a library of bases of *wavelet packets* (this name is from [CMQW]), which are efficiently encoded superpositions of wavelets. These adapted subband bases come with a natural quad-tree organization and some remarkable orthogonality properties. It is possible to introduce a cost function and pick a "best" wavelet packet basis for one or many pictures. This basis and the resulting decorrelated pixel valuess can then be compactly coded, but also the analysis performed during the choice of representation provides some information about the picture and could be useful for feature recognition.

3.6.1: Adapted subbands or wavelet packets. As mentioned in an earlier paper [CMQW], we can build a large library of adapted subband bases by retaining all amplitudes in the quadtree. The amplitudes produced at each stage are correlations of the signal with compactly-supported oscillatory functions called *wavelet packets*. From the tree $\mathbf{W}$ of subspaces we may choose a *basis subset*, defined as a collection of mutually orthogonal subspaces $W \in \mathbf{W}$, or lists of pairs (n,m), which together span the root. Basis subsets

are in one-to-one correspondence with dyadic decompositions of the unit square. Classical subband coding takes amplitudes from a fixed set of subbands, usually from a single level of the quadtree. Wavelet transform coding also extracts amplitudes from a fixed collection of blocks, the octave subbands.

Even for a small tree, the library of wavelet packet bases is very large:

Proposition. *The number of wavelet packet basis subsets for N-pixel pictures is greater than 2^N. The number of operations needed to compute all the transformed pixel values in all these basis subsets, however, is no more than $N \log_4(N)$.*

Proof. A decomposition to level n is only possible for a picture of size at least $N = 4^n$ pixels, and in such a tree there can therefore be at most $N \log_4(N))$ transformed pixel values. Let A_n be the number of bases in the library corresponding to a tree of $1+n$ levels, namely levels $0, \ldots, n$. Then $A_0 = 1$, and we can calculate $A_{n+1} = 1 + A_n^4$, namely the root and combinations of the 4 children, which are independent subtrees with A_n bases each. Simplifying this by discarding the 1 gives the estimate $A_{n+1} > 2^{4^n} = 2^N$ for $n > 1$. □

3.6.2: The best-basis algorithm. To each subspace $W \in \mathbf{W}$ we may assign an information cost H_W. The quantity $H_W(S)$ measures the expense of including W in the decomposition used to represent the picture S. Define the *best basis* for representing S (with respect to H_W) to be the basis subset B_0 which minimizes

$$\sum_{W \in B} H_W(S) \qquad (3.6.2\text{-}29)$$

over all basis subsets $B \subset \mathbf{W}$.

Some examples of information cost functions are listed in [CW]. The simplest is the number of elements above a predetermined threshold ϵ, namely $H_W(S) = \#\{x \in S_W : |x| \geq \epsilon\}$, where S_W is the sequence of the pixel values of S as transformed into the standard basis of W. This sequence is $F_{i_m} \ldots F_{i_1} S$, where $W = F_{i_m}^* F_{i_m}^* \ldots F_{i_1}^* F_{i_1} W(0,0)$. The following algorithm finds the basis subset with the fewest amplitudes above the threshold.

Set a predetermined deepest level L. Label as "kept" each subspace at level L, i.e., the subspaces indexed by (n, L) for $0 \leq n < 4^L$. Next, set the level index m to $L-1$. Compare the information cost of the subspace $W(n,m)$ with the sum of the information costs of its children $W(4n, m+1)$, $W(4n+1, m+1)$, $W(4n+2, m+1)$, and $W(4n+3, m+1)$. If the parent is less than or equal to the sum of the children, then mark the parent as "kept." This means that by choosing the parent rather than the children, we will have fewer amplitudes above the threshold in the representation of S. On the other hand, if the sum of the children is less than the parent, leave the parent unmarked but attribute to her the sum of the children's information costs. By passing this along, prior generations will always have their information costs compared to the least costly collection of descendents.

After all the subspaces at level $m = L-1$ have been compared to their children, decrement the level index and continue the comparison. At each level, we are comparing the information cost of a node to the sum of the lowest information costs obtainable by any decompositions of its 4 children. We can proceed in this way until we have compared the root $W(0,0)$ to its 4 children. We claim that the topmost "kept" nodes in depth-first order constitute a best basis. I.e., the collection of "kept" nodes W with no "kept" precursors is a basis subset which minimizes information cost. But this is easily proved by induction on the level index (see [CW] for the details).

If we think of the amplitudes below ϵ as negligible, we now have a basis in which the fewest amplitudes are non-negligible. This cost accounting requres deciding in advance what negligible means, which in some applications may not be feasible. The decision may be postponed by using a different measure of the concentration of energy into the amplitudes. For example, there is an additive analog of Shannon entropy, namely,

$$H_W(S) = -\sum_{x \in S_W} x^2 \log x^2, \qquad (3.6.2\text{-}30)$$

with S_W as above. This is related to the classical measure of the concentration of a probability distribution function which we discussed in an earlier section.

3.7: Functions underlying the transforms.

Each of the transform methods correlates a picture with some underlying functions or "templates" and then stores the correlations. For JPEG, the templates are products of sampled cosine functions restricted to square blocks:

Figure 3.7-13.

Density plots of fixed-block and adapted-block DCT functions.

For LCT, the underlying functions are "smeared out," without the sharp edges of the DCT functions:

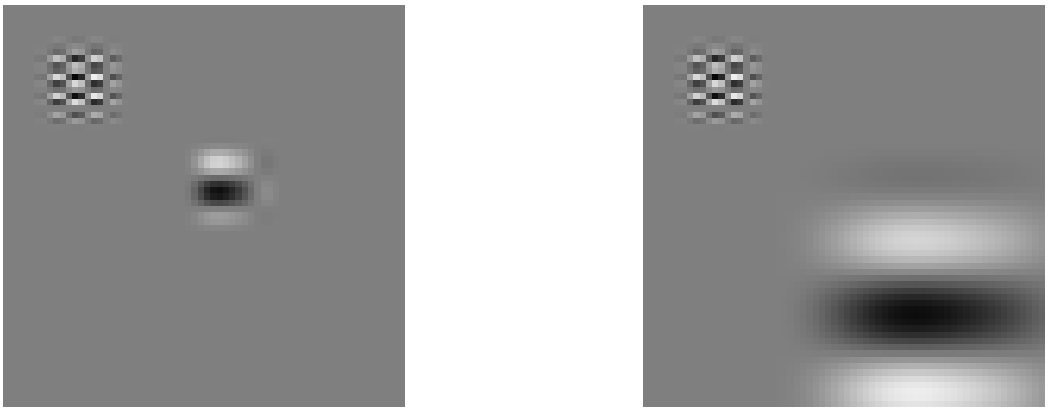

Figure 3.7-14.

Density plots of fixed-block and adapted-block LCT functions.

Two-dimensional wavelets and wavelet packets superpose in a different manner; wavelet packets of different scales can overlap in any manner, so long as their frequencies are distinct:

Figure 3.7-15.

Density plots of one wavelet and a superposition of three wavelets.

Wavelet packets, on the other hand, are superpositions of wavelets which are arranged so that they are efficient to describe. In particular, they correlate better with textures:

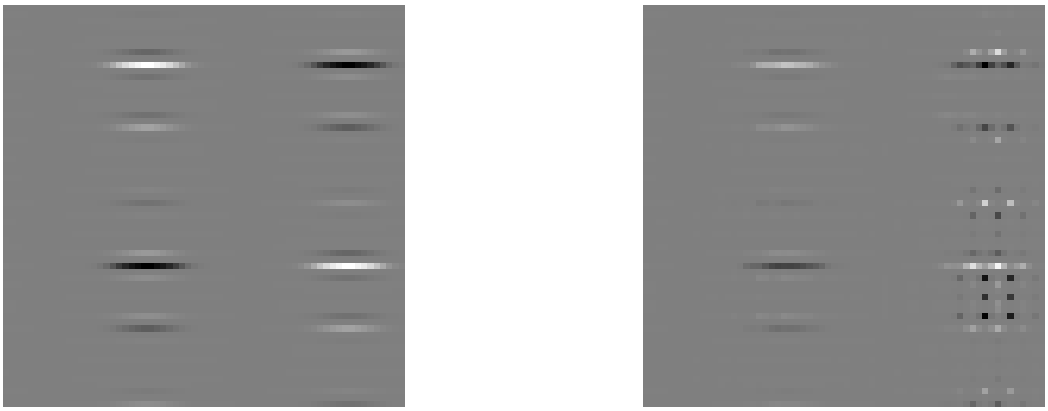

Figure 3.7-16.

Density plots of one wavelet packet and a superposition of three.

4: Quantization of transformed amplitudes

The transformed pixel values are real numbers which must be approximated in a (small) finite alphabet, or *quantized*, before they can be transmitted. All of the distortion from a lossy transform coding scheme is introduced at this step. The range of transformed values is divided up into numbered subintervals or *bins*. Any pixel value falling into a bin is approximated by the bin's index, as in the figure below:

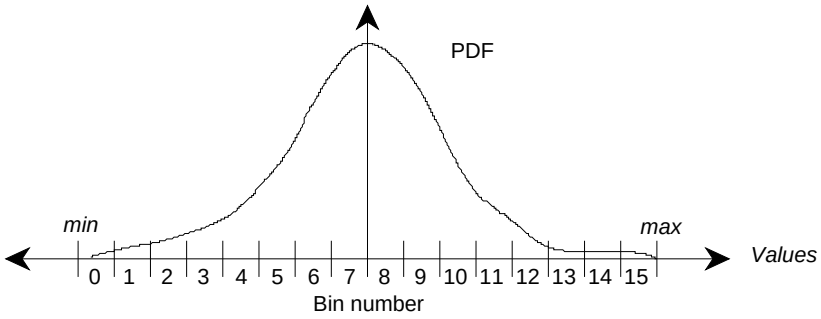

Figure 4-17.

Quantization into 16 equal-size bins.

In practice, the bin around 0 is often taken to be twice as wide as the other bins, and quantization is performed by scaling and truncation to the integer nearest to 0. The resulting integers are then biased into the range $[0, 1, \ldots, b-1]$, where b is the number of bins.

Quantization is undone by replacing the bin index with the value at the center of the bin. When the transformed pixel values will always have the same sign, or when 0 plays no special role, then quantization is done by dividing the full range between the maximum and minimum values into b bins. When the transformed pixel values can be both positive and negative and 0 is an important value, then it is vital that the bin around 0 be centered exactly at 0. Otherwise, the one-sided bias introduced by many values close to but not exactly equal to 0 will appear as artifacts in the reconstructed image.

The *quantization error* is the difference between the actual sequence of values and the sequence of bin center values, measured in some norm like mean-square-error or maximum absolute error. It is possible to vary the width of the bins, so that the more popular values (where the PDF is larger) are quantized more finely, into narrower bins. The distortion-minimizing choice of bin widths given a fixed number of bins can be found using the Lloyd-Max algorithm (see [Jayant,Noll]). This has the effect of reparametrizing the rate-distortion curve for the signal, but not improving the compression rate for a given degree of distortion. We can easily allow variable numbers of equal-sized bins, to adjust the compression rate.

4.1: Uniform quantization. In this method we use a single set of bins for all the transformed pixel values. This method is used when we have no *a priori* knowledge of the importance or relative visibility of a basis element.

4.2: Visibility quantization. In this method we use a model of the relative importance of a transformed pixel value to choose a weighting coefficient. The value is multiplied by this weight prior to quantization. It is known that the human eye is less sensitive to errors at certain spatial frequencies. When it is possible to determine the spatial frequency of the part of an image which will be reconstructed from an amplitude, this fact may be used to reduce the perceived distortion at a given level of compression.

4.3: Bit allocation. When transformed pixel values can be grouped, for example by subband, then we can allocate bits to the groups in a nonuniform manner to minimize the quantization error. The optimal allocation for a fixed number of subbands assigns bits in proportion to the variance within a subband (see [Jayant,Noll]). Another way to put this is, if q_i is the number of quantization bins to be assigned to subband W_i, then we should have $q_i/\sigma(W_i) =$constant for all i, where the constant depends upon the total variance of all the transformed pixel values as well as on the total number of bits we can afford to transmit or store.

Great competitive advantage may be gained from intelligent bit-allocation schemes, perhaps in combination with visibility weighted quantization based on acurate models of the kind of images to be transmitted. Such schemes are valuable property, and jealously guarded secrets in the industry.

5: Removing redundancy or entropy coding

Digitally coded data can often be reversibly transformed into a more efficient form, requiring fewer bits to store than its original representation. We shall refer to such an invertible transform as "lossless coding" because all the information in the original bits can be recovered. This is the only acceptable way to compress certain kinds of data sets such as compiled computer programs and data archives. Such methods have great practical significance and have been extensively studied.

It is a classical fact (see, for example, [Shannon,Weaver]) that there is a limiting rate of (lossless) compression achievable by such an alphabet substitution. Suppose we have an arbitrarily long message $M = (M(1), M(2), M(3), \dots)$ composed in an alphabet with finitely many letters: $M(j) \in a = \{a_1, a_2, \dots, a_n\}$ for all $j = 1, 2, \dots$. Coding these letters in the obvious manner requires $\log_2(n)$ bits per letter, and so the first L letters of the message will require $L \log_2(n)$ bits to transmit. Let $B(a, L)$ denote the number of bits required to transmit the first L letters $M(1) \dots M(L)$ using alphabet a; then we see that

$$\lim_{L \to \infty} \frac{B(a, L)}{L} = \log_2(n) \tag{5-31}$$

Shannon's theorem asserts that there is a nonnegative number H, the entropy of the probability distribution of the original alphabet, such that any new alphabet $\{b_i\}_{i=1}^n$ of variable-length characters satisfies:

$$H \leq \lim_{L \to \infty} \frac{B(b, L)}{L}, \tag{5-32}$$

and that for every $\epsilon > 0$ there in fact exists a particular alphabet b^ϵ which satisfies

$$\lim_{L \to \infty} \frac{B(b^\epsilon, L)}{L} \leq H + \epsilon. \tag{5-33}$$

Suppose that the original signal is written using the letters $a_1, a_2, \dots, a_n$, each occurring with probability

$$P(a_i) = p_i = \lim_{L \to \infty} \frac{\#\{j : 1 \leq j \leq L, M(j) = a_i\}}{L} \tag{5-34}$$

Then the entropy of the message is $H = -\sum_{i=1}^n p_i \log_2(p_i)$, and it is not too hard to show (as in [Ash], Theorem 1.4.2, p.17) that

$$0 \leq H \leq \log_2(n). \tag{5-35}$$

The left equality (best compression) holds if and only if the original message consists of a single letter of the alphabet repeated forever. The right equality (no compression) holds if and only if the original letters a_i are equally probable.

Several algorithms exist to construct the good alphabets, of which the earliest is probably static Huffman coding [Huffman]. In practice the probabilities p_i are determined empirically as the message is being sent, so there are refinements such as dynamic Huffman coding [Storer, p.40], arithmetic or Q-coding [Storer, p.47], data dictionary methods, and so on.

We can refine the application of Shannon's theorem by taking different lengths n for the initial alphabet. For example, it is natural to consider 8-bit characters ($n = 256$) for binary data emitted by a typical computer. Nevertheless, it may be that for a certain class of signals (such as Kanji text, which uses more

than 8 bits per character), the entropy of the alphabet consisting of 2-letter pairs is lower than the entropy of the original single letters. This idea may be extended to k-letter pairs, with the caveat that building the code grows in complexity exponentially with the number of bits. Hence there are some compromise algorithms which search a small range of alphabet lengths. Some of these, like Unix™ 'compress(1)' which is based on the Lempel-Ziv algorithm, are remarkably effective as well as quite fast [Ziv,Lempel77], [Ziv,Lempel78].

Shannon's theorem must be applied carefully, since in practice there are no infinite signals. The infinite-length signal assumption allows us to ignore the the fixed overhead of transmitting the new alphabet to the receiver. In practice, there must always be agreement between transmitter and receiver on both the original and the minimal alphabets, which must be established before any communication occurs. Lossless redundancy removers must always account this overhead. Indeed, if the alphabet were taken to be huge (i.e., large enough so that a particular finite signal consisted of only one "letter"), then the entropy of the message would be 0 and no bits would be required to transmit it. In that case, the receiver must know the original alphabet and also the minimal alphabet, which is equivalent to assuming that the receiver already knows the message, having received it as part of the overhead.

6: Image compression algorithms

6.1: JPEG.

The figure below describes, in block diagram form, the Joint Photographic Experts Group (JPEG) picture compression algorithm.

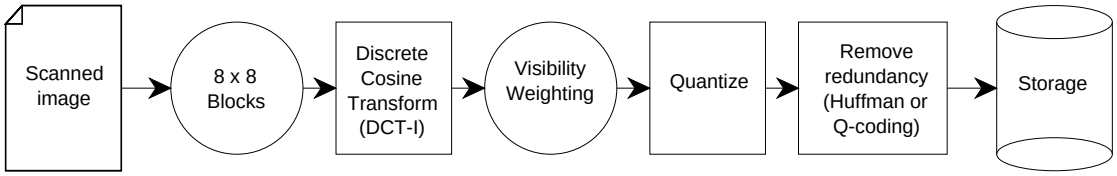

Figure 6.1-18.

JPEG visibility-weighted transform coding.

It specializes the generic transform coder by first cutting up the picture into independent 8×8 blocks, and also by multiplying the transformed pixel values by "visibility weights" determined from psychovisual measurements. Small sub-blocks are needed to limit the complexity of the algorithm. A striking visual justification for the weighting can be found in [DeVore,Jawerth,Lucier], p.734.

There are two main difficulties with JPEG. First, the algorithm cannot take advantage of large-scale correlations among pixels, such as those caused by regular textures covering large areas. This limitation is imposed by the choice of an 8×8 block size, and in some cases is overcome by using a larger block size. It is known, for example, that 16×16 or even 32×32 block sizes give better compression ratios for the same distortion when the images are high resolution fingerprint images. Second, the artifacts introduced by JPEG losses severely affect those image processing algorithms which detect and treat sharp edges. Namely, small pixels errors introduced along the sharp boundaries between blocks appear to be false edges because they line up. This artifact is called "blockiness" when it becomes noticeable to the eye at high compression rates, and is mitigated by over-quantizing the $(0,0)$-frequency or "DC" component of the transformed pixel values. The JPEG standard allows for proprietary algorithms to be used for visibility weighting, and various implementations use this flexibility to gain considerable reduction in visible distortion at fixed compression ratios.

The definitive description of the JPEG standard may be found in [JPEG1] and [JPEG2]. A more expository description which includes more details than this article may be found in [Wallace].

6.2: LCT or LOT.

The specialization of this method from the generic transform coder is depicted in the figure below:

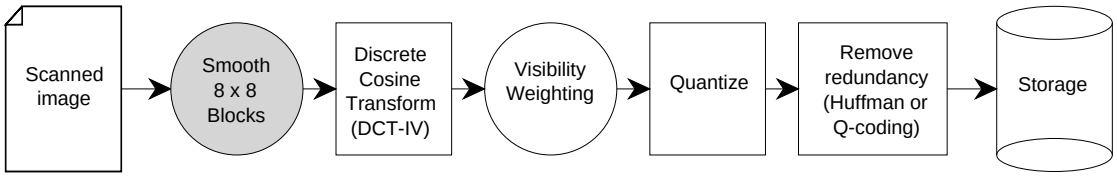

Figure 6.2-19.

Block LCT transform coding with frequency weighting.

The main difference (the use of smooth blocks) is noted by the shaded block. As with JPEG, we can take advantage of the varying frequency response of the eye and use frequency weighting of the transformed amplitudes to reduce perceived distortion at a given compression rate.

6.3: Orthogonal wavelets.

The special features which set the wavelet transform apart from the generic transform coder are depicted in the figure below:

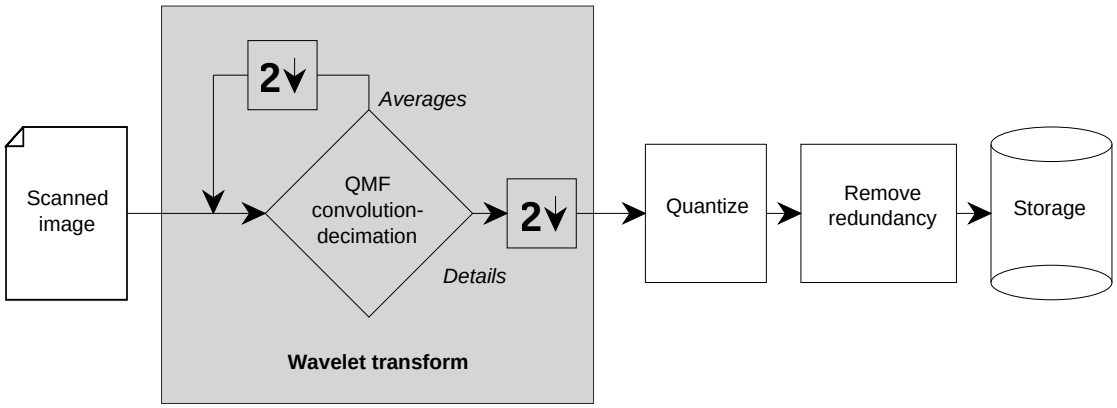

Figure 6.3-20.

Wavelet transform signal compression.

One of the image compression algorithms available on a VLSI microprocessor (for example from Aware, Inc. in Cambridge, Mass.) follows the above scheme, using short QMFs, a small number of decomposition levels, fixed-point arithmetic, variable bit-allocation, and arithmetic coding. The operations are optimized for speed, to make the implementation useful for motion-picture image compression.

Another algorithm is described in [Antonioni,Barlaud,Mathieu,Daubechies]. It uses short QMF wavelet transformation, followed by vector quantization of the triples of corresponding-position values from subbands $W(1,k)$, $W(2,k)$, and $W(3,k)$ for each level k.

6.4: Custom wavelet packets.

We can choose some other basis subset of subbands, rather than the one giving the wavelet basis. That will require refiltering the "detail" values, so the pyramid scheme is not enough. Schematically, this compression algorithm uses a subband selection criterion:

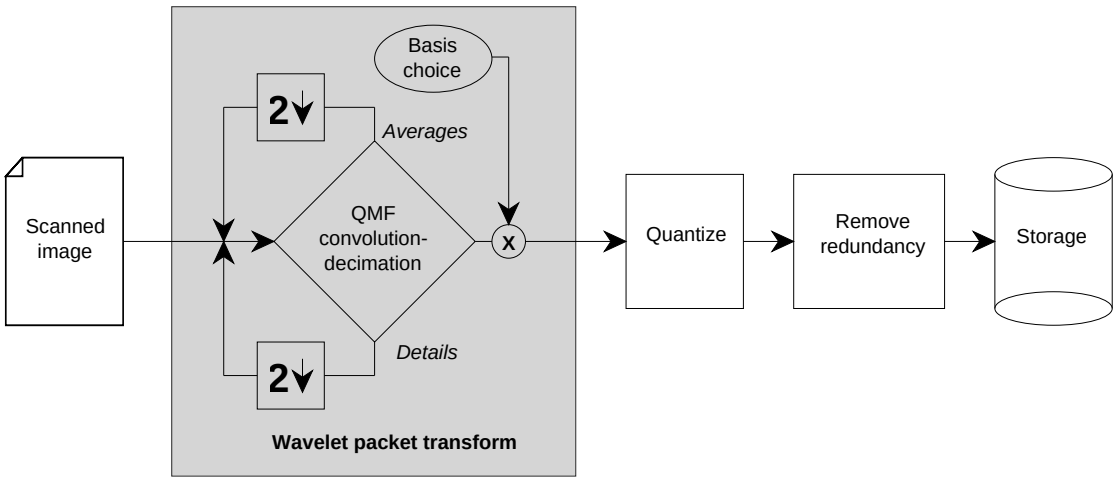

Figure 6.4-21.

Signal compression with custom wavelet packets.

The proposed FBI-Yale fingerprint image compression algorithm [Hopper] uses just such a custom selection of subbands, discovered by experiments with the best-basis method, and trials involving amplifying or attenuating the various subbands. Values from the custom subbands are weighted by visibility, then scaled and truncated to integers, with the integers passed to Lempel–Ziv or Huffman coding compression. Another feature of the FBI-Yale algorithm is its use of biorthogonal QMFs (see [Cohen]). These can be linear in phase so their underlying wavelet functions are reflection-symmetric, unlike the orthogonal wavelets which will always make an artificial distinction between left and right. Biorthogonal wavelets do not decorrelate the signal as well as orthogonal wavelets, but the difference appears to be of minor importance in practice. As can be seen from Figures 3.5-6 and 3.5-7 above, the orthogonal "C 30" wavelets are quite close to reflection symmetric; conversely, some of the symmetric biorthogonal wavelets are quite close to orthogonal.

6.5: Best-basis of wavelet packets.

We can also store all the amplitudes in the complete quadtree subband expansion, and then search for the most efficient representation by the "best-basis" algorithm. The stream of amplitudes thus produced can be quantized and coded as before. Schematically, this compression algorithm is depicted below:

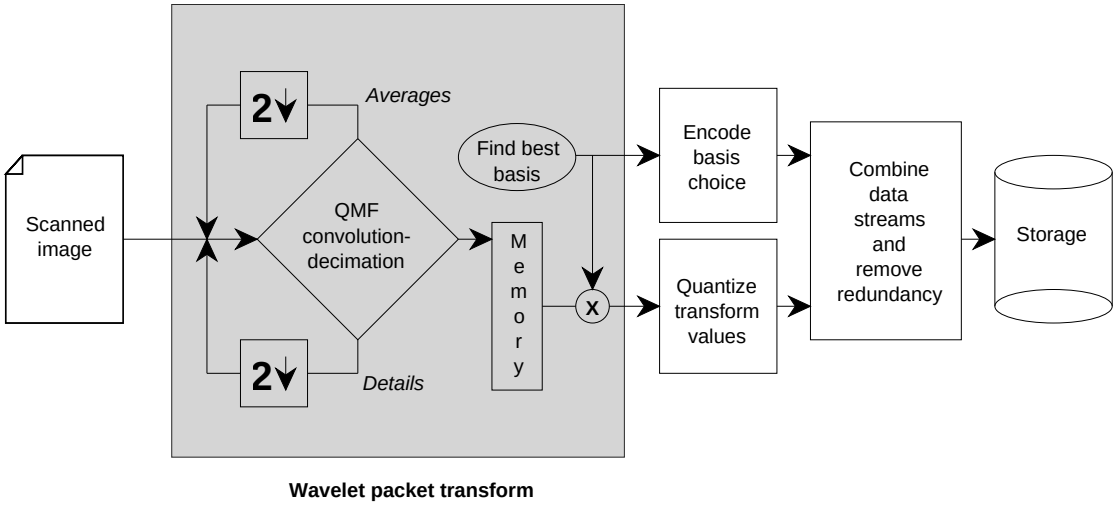

Figure 6.5-22.

Wavelet packet (best-basis) signal compression.

The transformation from a picture to its best-basis representation is nonlinear. Since the "best" basis choice depends upon the picture, it must also be encoded together with the transformed pixel values. Unlike those values, the basis description must be coded without any losses. After the lossless coding of the nonlinear information, the new pixel values are an orthogonal linear trasnformation of the original pixels. The whole process therefore has condition number 1.

The greater the number of bases in the library, the more side information must be transmitted to describe which one was selected. In [Ramchandran,Vetterli1], for example, only subbands close to the wavelet subbands are included in the best-basis search. In other schemes, the number of decomposition levels is kept small but all bases within that collection of levels are considered.

There are at least two ways to include the basis information. Best-basis amplitudes may be individually tagged with their coordinates in the best-basis tree. Unless the number of transformed pixel values is very small, or the precision very high, tagging large amplitudes will not be the most efficient compression method. Alternatively, we may agree upon an ordering of the amplitudes and a regular scheme for describing the basis. We will include some side information which describes the chosen basis, and we shall then write all the (quantized) amplitudes from that basis out into a stream for entropy coding. We obtain compression because the quantized stream of transformed pixels has a lower entropy that the original stream of pixels. There are various orderings one could choose; which one is most efficient depends upon the constraints in the basis search. This method is essential for a competitive picture compression algorithm.

Imagine $L+1$ arrays of $N \times N$ numbers. The first array represents the original signal, which we may call Z. The second is a concatenation of the 4 subspaces obtained via separable filter convolution-decimation, i.e., the spaces $F_0(X)F_0(Y)Z$, $F_1(X)F_0(Y)Z$, $F_0(X)F_1(Y)Z$, and $F_1(X)F_1(Y)Z$. Array m represents the concatenations of the 4^m subspaces that make up level m of the wavelet packet decomposition. Of course, we must have $0 \leq m \leq L \leq \log_2(N)$. To understand the quadtree coding schemes described below, it is helpful to visualize these arrays stacked one atop the other, as in the figure below:

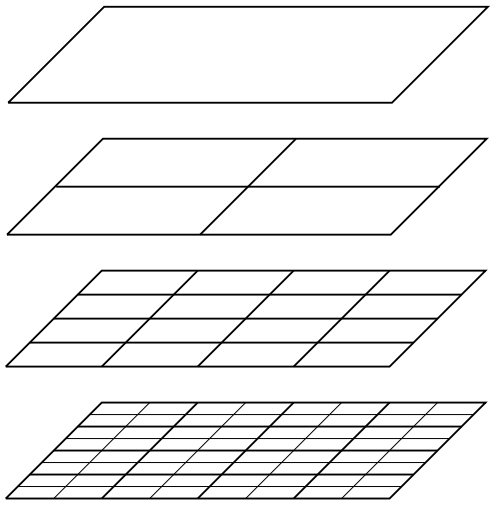

Figure 6.5-23.

Two-dimensional wavelet packet quadtree.

A basis subset of the subbands has the the property that if one element of a subbands is in the basis, then that whole subband is in the basis. Also, if a subspace is in the basis, then none of its descendent or ancestor subbands are in the basis. Such a subset can be identified with a cover by dyadic subarrays. Looking down through the stack of arrays in Figure 6.5-23, this cover gives a tiling of the original $N \times N$ array by square subarrays of size $2^{-m}N \times 2^{-m}N$, where m is the level from which that particular subspace was chosen.

6.5.1: Tag the large values. Suppose that B is a best-basis subset of $\mathbf{W}$, chosen by counting amplitudes above a predetermined threshold. We may then extract just these non-negligible amplitudes and transmit them, together with their locations in the tree. This number of amplitudes is no greater than the number of pixels, since it is chosen after comparison with the original basis, among others. Define the *compression ratio* in this case to be the ratio of the number of retained amplitudes to the number of original pixels. With thresholding to ϵ and counting, the compression ratio measures how well a library represents a picture S at a fixed precision ϵ.

In practice it is sometimes necessary to fix the compression ratio, for example due to bandwidth limitations. In that case we may use the entropy cost function to obtain the most concentrated representation, and then take only as many of the largest amplitudes as we can afford. This may be accomplished by first sorting into decreasing order by absolute value, then reading off the desired number of amplitudes. Alternatively, since we know in advance how many amplitudes we can use, it may be more efficient to bubble up the top few amplitudes and discard the rest of the array. The second method is better if the number of retained amplitudes is less than $\log_2(N)$, where N is the number of pixels.

Together with the transformed pixel value we must include the extra information describing which basis was used, and we must somehow indicate which basis vector each quantized value represents. Suppose the quadtree begins with an $N \times N$ signal and decomposes it down to level L, where $L \leq \log_2(N)$. Then there are LN^2 wavelet packet amplitudes, and it takes $\log_2(LN^2)$ bits to encode each individual one. This method is used and documented in the wavelet packet software programs available by anonymous ftp from the Yale Mathematics Department [pascal].

The overhead of this method is a constant number of bits per retained value. This technique is used in

numerical analysis, where the "picture" might be a matrix and the pixels are double-precision floating point values. Then we will retain the nonnegligible values to full precision and ignore the rest. We win because we have reduced the number of parameters in the problem, and thus the complexity of the calculation.

6.5.2: Coding the complete basis. In this scheme, we use two arrays to describe the transformation, a *levels list* and an *amplitudes list*. The subspaces in the best basis are encountered in depth-first order as they are selected, and this order can be used to code the quadtree. The side information consists of an array of integers which describe at which level the next subband in the best basis resides. The subbands themselves are traversed in depth-first order, and the level of the next subband is determined by a very short integer of at most $\log_2(L)$ bits (for an L-level decomposition). Some extra economy is possible, since the presence in the best basis of a node at level m sometimes implies that the following nodes can only be in levels $m, \ldots, L$. The extreme case, for example, is that each first occurence in the levels list of a subspace at the deepest level L implies that all of its siblings are also in the best basis. Hence whenever the deepest level appears, it is not necessary to follow it with 3 "L" symbols. Further, the only levels list which contains a marker for level 0 is the list $\{0\}$, which implies that the root of the tree, or the original signal, is the best representation. That unique situation can be encoded with a single bit, and we can use the value 0 instead of L to represent the deepest level, thus using only the range $0, 1, \ldots, L-1$ in the coding of the tree. It then becomes convenient to reverse the meaning of "level" so that it describes distance from the bottom level 0. The original signal will now be labeled with L, the first decomposition level $L-1$, and so on. This side information costs at most $4^{L-1} \log_2 L$ bits, or $4^{L-1} \log_2 L/N^2$ bits per pixel. The worst cases occur when every subspace at the deepest level is chosen and the levels list is $\{0, 0, \ldots, 0\}$ (4^{L-1} 0's), or when the next-to-deepest level is chosen and the levels list is $\{1, \ldots, 1\}$ (4^{L-1} 1's). The side information will also be compressed losslessly as part of the compression algorithm, and because of the redundancy in both of these cases the lossless compression will be extremely efficient.

Let us consider an example. Suppose that the picture consists of a single wavelet packet from subband $W(5,3)$ (i.e., frequency 5 and level 3). The best basis algorithm would have a choice of decompositions since many comparisons of children and parents would show equal information costs. If we use the convention that levels closer to the root (i.e., parents) are preferable, then the canonical best-basis decomposition of this picture includes the following collection of subbands:

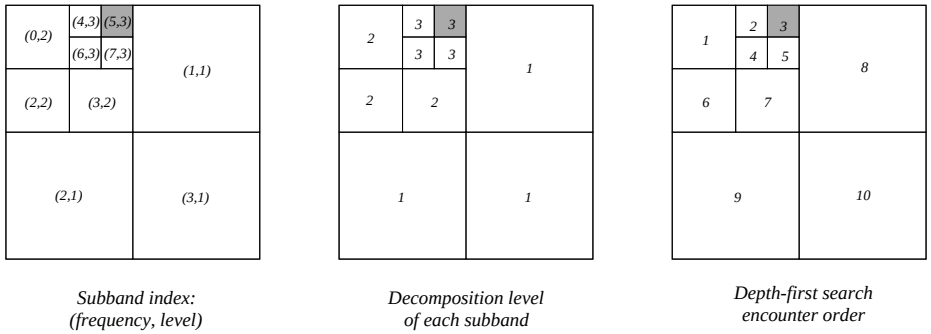

Figure 6.5.2-24.

Various descriptions of the subbands in the best basis.

Then the depth-first-search encounter order levels list would be $\{2, 3, 3, 3, 3, 2, 2, 1, 1, 1\}$. If the maximum level were declared to be 3, we could economize in the conventional manner above and write this as $\{1, 0, 1, 1, 2, 2, 2\}$.

Suppose also that the dimensions of the picture are 16×16. At level 3 each subband has 2×2 amplitudes, at positions $(k_x, k_y) \in \{(0,0),(1,0),(0,1),(1,1)\}$; let us suppose that the wavelet packet in our picture has amplitude 1 at position $(0,0)$. We can write the amplitudes list in any predefined manner, but it is convenient and conventional to dump the amplitudes from each subspace as it is encountered in depth-first order. This approach, in the example case, produces a stream of small square arrays, each transmitted as lists of rows:

```
 (0,2)  (4,3) (5,3) (6,3) (7,3)  (2,2)    (3,2)       (1,1)           (2,1)           (3,1)

0 0 0 0  0 0   1 0   0 0   0 0  0 0 0 0  0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
0 0 0 0  0 0   0 0   0 0   0 0  0 0 0 0  0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
0 0 0 0                       0 0 0 0  0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
0 0 0 0                       0 0 0 0  0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
                                       0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
                                       0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
                                       0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
                                       0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0  0 0 0 0 0 0 0 0
```

The various subbands present in the best-basis of wavelet packets provide a segmentation of the picture in the time-frequency domain. Some selection criterion can be applied to preselect (in a crude manner) a desired feature of the signal, such as a given texture or a feature at a selected scale. Then the amplitudes within the subband can be used as a signature of the selected feature, while the positions of large amplitudes can be used to more precisely locate the feature. In the example above, we could track down the position of a $(5,3)$ wavelet packet in a noisy signal by finding the largest amplitude in the $(5,3)$ subband. Features characterized by linear compinations of wavelet packets can likewise be detected. Since wavelet packets form a basis this idea privides a general feature-detection algorithm. Of course, its effectiveness depends upon the simplicity of the wavelet packet description of the desired feature.

6.6: Adapted LCT and DCT.

It is possible to apply the best-basis algorithm to the library of bases obtained from block DCT or LCT transforms with varying block sizes. The algorithms are depicted schematically below:

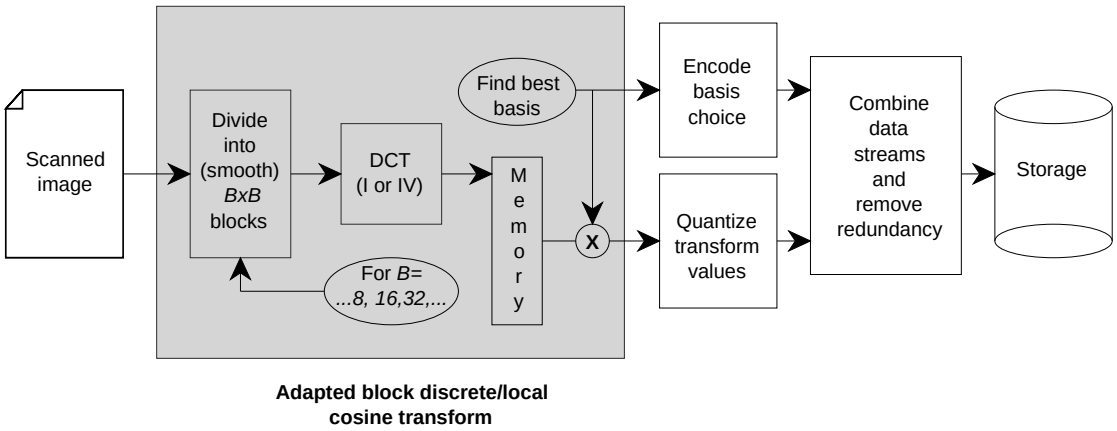

Figure 6.6-25.

Adapted block local cosine transform signal compression.

The structure of the subspaces in each decomposition is also a quadtree, only the descendents are produced by recursive subdivision into subblocks rather than recursive filtering and decimation. Each subblock contains all the frequencies up to the maximum possible with its number of samples, and each block is the direct sum

of its four children. Because the subspaces (now subblocks) have the same quadtree structure as subbands of a wavelet packet basis, the same basis-coding convention may be applied.

In the adapted DCT case (ADCT), the blocks account for independent regions of the original picture, with the usual sharp boundaries and attendant blocking artifacts. In the adapted LCT case (ALCT), adjacent blocks are smeared together. In both cases, the best-basis algorithm chooses a particular segmentation of the picture into blocks. This choice conveys information about the contents of the picture; the spectrum of a chosen block can then be used to identify what is present in that part of the picture.

A decomposition down to a deep level, where the subblocks are very small, results in rather sharp bells in the ALCT algorithm. This can be avoided by using "multiple folding," an alternative to ALCT which preserves the product of window width and window steepness. The cost is poorer spatial localization for a given block size. This algorithm is described in detail in [Fang,Seré].

6.7: Other methods.

One may also consider the correlation of a picture with the complete set of tensor products of wavelet packets. These form a larger nonhomogeneous tree of subspaces which must be labelled with an x-scale and y-scale, rather than with a single scale as above. There is a more general notion of admissible subset, and a best-basis search algorithm to find extrema. This basis will produce higher compression ratios at a given threshold, at a cost of greater computational complexity and increased overhead describing the basis. The practical disadvantages rule out this generalization for image compression; further details may be found in [INRIA].

7: Comparisons

The compression algorithms described above may be compared for: their effectiveness as measured by rate-distorion curves; the usefulness of the transformed pixel values as input to various transformations; the types of artifacts which appear when they are pushed to the point of visible distortion; and the complexity of computing the transforms. In addition, we would like to have an inherent measure of the "compressibility" of an image for use in practice; can we apply such a measure algorithmically or will it always be necessary to involve humans in judging acceptable distortion?

7.1: Transforming compressed pictures.

Wavelet packets contain a mixture of spatial and spectral information, as do wavelets, DCT and LCT values. A list of the most energetic subspaces used in a compressed picture conveys a signature for the picture. Certain operators are very efficiently represented by their action on the wavelet packet amplitudes. Some examples include spatial filtering and local image enhancement [INRIA], edge and texture detection [Hwang,Mallat], principal orthogonal factor classification [W2], and local rescaling.

For purposes of explanation we will assume that the picture is a function of 2 real variables supported in the region $[0,1] \times [0,1]$, with a resolution of 2^{-L}. Let (n,m,k) be the index of an amplitude in the complete wavelet packet expansion of a picture S. Here $m = 0, 1, \ldots, L$. We may divide $0 \le n < 4^m$ into n_x and n_y by taking the odd and even bits in its binary expansion, respectively. These are then arranged in "sequency" order (by Gray-encoding; see [INRIA]) and then will approximately correspond to x and y frequencies. Likewise, k may be divided into its x and y components k_x and k_y. Then the transforms described above may be defined by their action on the amplitudes $c = c(n,m,k)$. Since the functions underlying the wavelet packet transform are smooth, small errors in the transformed amplitudes c appear as smoothly varying deviations, i.e., low-contrast distortion. This is a desirable property for transformations that might be coupled, for example, with edge detection algorithms or other procedures which are sensitive to abrupt changes or large derivatives.

For some examples of one-dimensional signal processing algorithms which are defined by their actions on wavelet packet and local cosine coefficients, see [INRIA] or [CMW].

7.1.1: Spatial filtering. To remove (or attenuate) high frequency components in a particular direction $\alpha = \tan \frac{y}{x}$, simply discard any amplitude for which $|\alpha - \tan \frac{n_y}{n_x}| < \epsilon$ with $n_x > C$, $n_y > C$, where the cutoff frequency C and the directionality ϵ are parameters of the filter.

7.1.2: Local image enhancement. To remove high frequency noise from a particular region of the picture, employ spatial filtering as described above, but only on those amplitudes for which 2^m is less than the diameter of the region, and for which $2^m(k_x, k_y)$ is a point in the region.

7.1.3: Edge detection. Suppose we wish to find an edge of scale m_0 in a picture of resolution L, for example a white region which darkens to black in a distance 2^{m_0-L}. Such an edge will contribute large amplitudes to scales $1, 2, \ldots, m_0$ at high frequencies. We may graph it by selecting only thise amplitudes $c(n, m, k)$ above a (large) threshold, with $m < m_0$ and n greater than an appropriate monotone function of m, and then plot points at $2^m(k_x, k_y)$.

7.1.4: Texture detection. Textures may be characterized by linear dependencies among wavelet packet amplitudes at nearby translations. Suppose for example that we wish to detect a texture in which $c(n_0, m_0, k+1) = -c(n_0, m_0, k+1)$ for all k in some region. An operator which added an amplitude at k to its neighbor at $k+1$ would have 0's in its range at k, indicating where that texture was located.

7.1.5: Local rescaling. We simply replace $c(n, m, k)$ with $c(n', m', k')$ for a restricted range of k's. The map $n \mapsto n'$, etc., is determined by the rescaling. For example, if $n' = 2n$ and $m' = m + 1$, then we will increase the magnification of the picture locally with little change in the frequency content.

7.2: Theoretical dimension and expected compression.

Roughly speaking, entropy measures the logarithm of the number of meaningful amplitudes in the signal. The lower is this quantity, the better the compression for a given level of distortion. Some experimental results [Devore,Jawerth,Lucier] suggest that for pictures, there is a strong correlation between the rate of decrease of transformed pixel values and the entropy of the PDF of the values. This relationship depends upon the pictures having a definite degree of smoothness (technically, sampled functions from a smooth Besov space).

Define the *theoretical dimension* $d(x)$ of a sequence $x = \{x_i\}_{i=1}^N$ to be the exponential of its entropy:

$$d(x) = \exp\left[-\sum_{i=1}^{N} \frac{|x_i|^2}{\|x\|^2} \log_4\left(\frac{|x_i|^2}{\|x\|^2}\right)\right], \qquad (7.2\text{-}36)$$

where $\|x\|^2 = \sum_{i=1}^N |x_i|^2$. This quantity always lies between 1 and k if there are no more than k nonzero values x_i. It can serve as a measure of the compressibility of picture after transform coding. The higher the theoretical dimension $d(x)$ of the transformed pixel values x, the more quantization bins should be used and the lower the expected compression for a given distortion level.

7.3: Operation counts.

7.3.1: Transformation coomplexity. The complexity of JPEG is the complexity of the 2-dimensional factored DCT restricted to 8×8 blocks, multiplied by the number of such blocks in the picture. Restriction makes it an $O(N)$ algorithm, since the Huffman coding is $O(N)$. The constant is rather small, around 10 or so.

LCT adds one step which costs N operations in each dimension, so the complexity is the same $O(N)$ as for JPEG with a constant of 12 or so.

Adapted-block DCT and LCT require computing up to $\log_4(N)$ different block DCT/LCTs, with block sizes from 8×8 to $\sqrt{N}$. This makes the complexity $O(N[\log_4(N)]^2)$ with constants around 3. This can be controlled by limiting the range of block sizes.

Orthogonal wavelet transformation has complexity $O(N)$, with the constant controlled by the length of the QMFs. Really long QMFs can be factored and then the constant is controlled by the logarithm of QMF length, but such filters are more useful in acoustic signal compression than image compression. Typical constants are around 10.

Adapted wavelet packet algorithms are more expensive. Suppose that S is an N-element picture. The convolution-decimations to generate the tree of amplitudes require $O(N \log_4(N))$ operations. The information cost calculation has complexity $O(N \log_4(N))$. Labeling "kept" subspaces is equivalent to a breadth-first search through the tree, which has complexity $O(N)$. Locating topmost "kept" subspaces is equivalent to a depth-first search, with complexity $O(N)$, and filling an output register with amplitudes from the best basis takes an additional $O(N)$ operations.

Reconstruction from the retained amplitudes has no more than the same complexity as generating all the amplitudes, since the transformations are orthogonal and have the same complexity as their adjoints

In the adapted transforms we must in general reproduce the entire tree to reconstruct the signal, so the reconstruction complexity is of the order as the compression complexity. The constant is smaller, though, because on average the chosen basis resides in a strictly smaller subtree, and because no additional steps such as searching for a best basis are needed during reconstruction.

7.3.2: Quantization complexity. We can perform a fast sort to determine the largest amplitudes in the output register: this has complexity $O(N \log_2(N))$. The alternative, extracting the top t amplitudes, requires $O(tN)$ operations. We choose the more efficient method: in either case the total complexity of this step is $O(N \log_2(N))$.

On the other hand, if we choose to quantize all amplitudes to some fixed precision, it is clear that we need $O(N)$ operations to replace each amplitude with is bin number.

7.3.3: Entropy coding complexity. If we choose to transmit all the quantized amplitudes, then we need to employ a standard entropy coding algorithm to gain compression. Those algorithms we have mentioned (arithmetic, Huffman, and Lempel-Ziv) all have complexity $O(N)$, which is dominated by other steps in the algorithm.

8: Experimental comparisons

To put the algorithms into perspective, we have prepared compressions of three images: an artificial still life generated by the ray tracing program "rayshade" written by Craig Kolb; a fingerprint digitized to 500 dpi at 8 bits grayscale, and a fingerprint digitized to 900 dpi at 8 bits grayscale provided by Tom Hopper. The fingerprints can be called high resolution images; the smaller one is sampled to the draft FBI standard [Hopper], while the larger is sampled to a higher standard used for early evaluation. The first image is sampled much more coarsely but still contains a great deal of information at all scales.

The reproductions below are provided to indicate what the images look like, not for any kind of distortion analysis. The raw data files are available from the author by request.

A *Ray-traced image, 256 x 256 x 8bpp*

B *Fingerprint image at 500 dpi, 512 x 512 x 8bpp*

C *Fingerprint image at 900 dpi, 1024 x 1024 x 8bpp*

Figure 8-26.

Three gray-scale test images.

We will compare the rate-distortion curves for 4 transforms described above: JPEG, LCT, orthogonal wavelets, and best-basis of wavelet packets.

8.1: Implementation parameters. The JPEG implementation uses the following luminance visibility matrix, which was provided by Dan Fuhrmann and Arun Kumar:

(8.1-37)

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

The x-frequencies increase from left to right, and the y-frequencies increase from top to bottom, just as indices in a matrix. What is displayed are the reciprocals of the weights, that is, the algorithm specifies that corresponding amplitudes in each 8×8 block be divided by the numbers above before uniform quantization. In fact the weights are scaled so that weighted quantization can be implemented by division and truncation to the nearest integer.

The LCT implementation uses 8×8 blocks as well, and the bell function is the one given by Eq.(3.3-3) above. It also uses the frequency weighting matrix in Eq.(8.1-37).

The orthogonal wavelet and wavelet packet transforms were computed using the periodized "C 6" QMFs, whose nonzero coefficients are given below:

(8.1-38)

$$\begin{aligned} h_0 = \quad g_5 = \quad & 0.038580777747886749, \\ h_1 = -g_4 = & -0.126969125396205200, \\ h_2 = \quad g_3 = & -0.077161555495773498, \\ h_3 = -g_2 = & -0.607491641385684120, \\ h_4 = \quad g_1 = \quad & 0.745687558934434280, \\ h_5 = -g_0 = \quad & 0.226584265197068560, \end{aligned}$$

All pictures were decomposed only to level 8. Amplitudes were quantized uniformly. In the best-basis case, the information cost function was $H(x) = -\sum_i x_i^2 \log(x_i^2)$. The 46 subspaces chosen for image "A" by the best-basis algorithm are as follows: {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 4, 4, 4, 5, 5, 5, 6, 6, 6, 7, 7, 7}. There were 22,513 subbands in the best-basis chosen for image "B," contributing 7006 bytes or 0.026 bits per pixel (packed, but not entropy coded) of overhead on top of the storage requirements for the quantized amplitudes. There were 9349 subbands in the best-basis chosen for image "C," contributing 2995 bytes or 0.0028 bits per pixel (again, packed but not entropy coded) of overhead. Since the wavelet basis for all the images would contain only 25 subbands, it is clear that the best bases for images "B" and "C" are quite different from the wavelet basis. Image "A," on the other hand, is best represented by something quite close to the wavelet basis.

8.2: Rate-distortion comparisons.

Mean square error is the measure of distortion in this comparison. Some other error norms should be mentioned, although they are not used here. A method described in [Fang,Seré] allows arbitrary distortion measures in the adapted local cosine best-basis method. In another approach, (see [Ramchandran,Vetterli2]), the rate-distortion curve is used as a distortion measure for the transform. Also there are projects (for example [Fuhrmann]) to find a perceived distortion measure for the JPEG compression algorithm. For comparison of rate-distortion functions, the compression rate is computed from the entropy of the PDF of the transformed pixel values.

Below are the rate distortion curves for the transformed pixel values of the 3 pictures "A," "B," and "C," transformed using JPEG, LCT, orthogonal (C6) wavelets and the best-basis of (C6) wavelet packets.

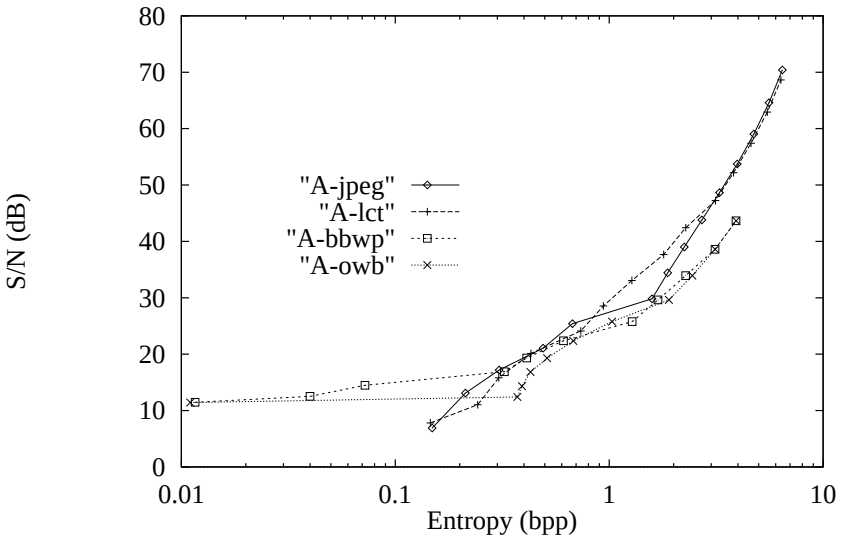

Figure 8.2-27.

Rate-distortion curves for image A.

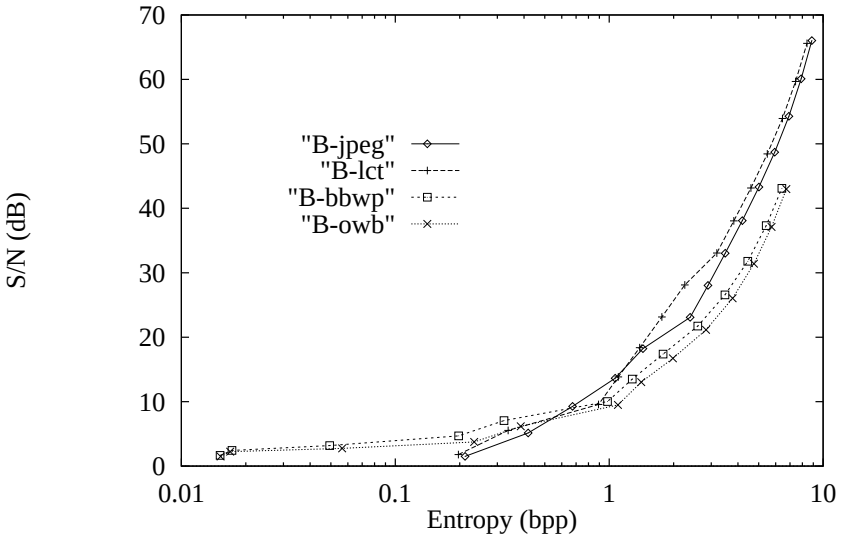

Figure 8.2-28.

Rate-distortion curves for image B.

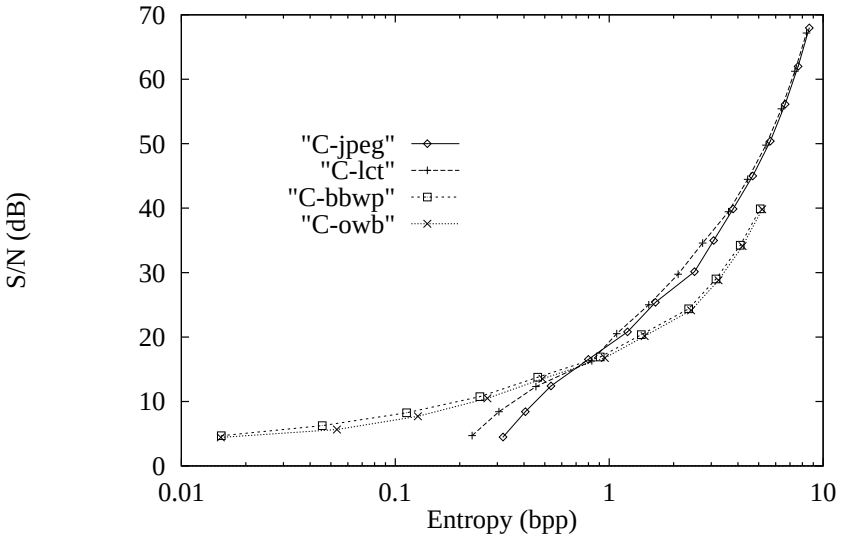

Figure 8.2-29.

Rate-distortion curves for image C.

Observe that at high bitrates (above 0.8 bits per pixel or so, compression ratio 10 or so) the rate-distortion curves for JPEG and LCT lies above that for orthogonal wavelets and best-basis of wavelet packets. At lower

bitrates, the wavelet and best-basis algorithms give better results. We can make several remarks.

First, the JPEG and LCT algorithms with their fixed block sizes cannot achieve ultra low bitrates. At compression ratios of 100 or more the wavelet packet algorithms still show a signal-to-nose ratio of 5–10dB.

Second, at nearly all bitrates the wavelet and best-basis curves are quite close. In the absence of other criteria, it appears that we can profitably save the overhead of the best-basis description (small though it is). It would nonetheless be worthwhile to keep that extra data (and to do the extra analysis) if we could use the basis description for feature recognition or classification.

Third, the frequency-weighted quantization used by both JPEG and LCT results in lower perceived distortion at the cost of freezing a viewing scale. Those subbands which were coarsely quantized because of the eye's contrast insensitivity at that spatial frequency might fall into a more sensitive region upon magnification. This accounts for the blockiness of JPEG in blown-up images.

8.3: Software.

All compressions were performed in software on an unloaded NeXTCube 68040-based desktop computer with 40 MB of RAM. In the table below, the VSIZE entry denotes virtual memory size, while RSIZE denotes actual memory being allocated (in megabytes). UTIME is user CPU time in seconds, while STIME is the operating system's CPU time.

	``A''				``B''				``C''			
CMD :	VSIZE	RSIZE	UTIME	STIME	VSIZE	RSIZE	UTIME	STIME	VSIZE	RSIZE	UTIME	STIME
bbwp :	3.66M	1.40M	14.4u	19.2s	5.88M	3.15M	55.6u	73.8s	14.4M	11.6M	209.8u	297.2s
owb :	3.44M	1.23M	4.1u	1.9s	4.94M	2.73M	18.0u	7.8s	10.9M	8.73M	71.1u	31.2s
jpeg :	2.23M	1.16M	4.6u	1.7s	3.76M	2.67M	21.2u	8.0s	9.73M	8.68M	86.2u	31.0s
lct :	2.15M	1.16M	5.9u	11.2s	3.68M	2.67M	25.6u	43.8s	9.65M	8.68M	101.7u	175.0s

The timings are approximately linear in the number of pixels, since the recursion depth for the subband methods was limited to 8 levels.

9: Acknowledgments

The ray traced image ("A") was provided by Craig Kolb of the Department of Mathematics at Yale University. The two fingerprint images ("B" and "C") are cropped versions of data files provided by Thomas E. Hopper at the Systems Technology Unit of the Federal Bureau of Investigation in Washington, DC. The Adapted Waveform Analysis Library of ANSI C functions, which was used to compute the transforms in the experimental trials, is available from FMA&H, Inc., in Hamden, CT.

References

M. Antonioni, M. Barlaud, P. Mathieu, and I. Daubechies, *Image coding using wavelet transform*, IEEE Transactions on Acoustics, Speech, and Signal Processing (1991).

Tom M. Apostol, *Calculus, Volume II*, second edition, John Wiley & Sons, New York, 1969.

Robert B. Ash, *Information Theory*, ISBN 0-486-66521-6, Dover Publications, New York, (1965/1990).

Michael F. Barnsley and A. D. Sloan, *A better way to compress images*, Byte Magazine (January, 1988), 215-223.

, *Biorthogonal wavelets*, preprint, AT&T Bell Laboratories (1991).

Albert Cohen and Ingrid Daubechies, *Non-separable bidimensional wavelet bases*, to appear, Revista Matemática Iberoamericana (1992).

R. R. Coifman and Y. Meyer, *Remarques sur l'analyse de Fourier à fenêtre*, série I, C. R. Acad. Sci. Paris **312** (1991), 259–261.

R. Coifman, Y. Meyer, S. Quake, and M. V. Wickerhauser, *Signal processing and compression with wavelet packets*, Proceedings of the conference on wavelets, Marseilles, Spring 1989.

Ronald R. Coifman, Yves Meyer, and M. Victor Wickerhauser, *Wavelet analysis and signal processing*, Wavelets and Their Applications, ed. Ruskai et al., ISBN 0-86720-225-4, Jones and Bartlett, Boston, 1992, pp. 153–178..

M. V. Wickerhauser and Ronald R. Coifman, *Entropy based methods for best basis selection*, IEEE Transactions on Information Theory (March, 1992).

Ingrid Daubechies, *Orthonormal bases of compactly supported wavelets*, Communications on Pure and Applied Mathematics **XLI** (1988), 909–996.

Ron Devore, Bjørn Jawerth, and Bradley J. Lucier, *Image compression through wavelet transform coding*, IEEE Transactions on Information Theory **38** (March, 1992), 719–746.

Xiang Fang, Eric Seré, *Multiple folding local sine transform*, preprint, Washington University in St. Louis (1991).

Thomas E. Hopper, *Grey-scale fingerprint image compression status report*, internal report, FBI/Systems Technology Unit, Washington, DC. (November 4, 1991).

D. A. Huffman, *A method for the construction of minimum-redundancy codes*, Proceedings of the IRE **40** (1952), 1098–1101.

Stphane Mallat and W. L. Hwang, *Singularity detection and processing with wavelets*, IEEE Transactions on Information Theory (March, 1992).

M. Victor Wickerhauser, *INRIA lectures on wavelet packet algorithms*, lecture notes, INRIA, Roquencourt, France, 1991.

Nuggehally S. Jayant and Peter Noll, *Digital coding of waveforms : principles and applications to speech and video*, Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

ISO/IEC JTC1 Draft International Standard 10918-1, *Digital compression and coding of continuous-tone still images, Part 1: requirements and guidelines*, document number ISO/IEC CD 10918-1 (alternate number SC2 N2215), available from ANSI Sales, (212)642-4900 (November, 1991).

ISO/IEC JTC1 Draft International Standard 10918-2, *Digital compression and coding of continuous-tone still images, Part 2: compliance testing*, document number ISO/IEC CD 10918-2, available from ANSI Sales, (212)642-4900 (December, 1991).

Wlodysław Madych and Karlheinz Gröchenig, *Multiresolution analysis, Haar bases and self-similar tilings of* $\mathbf{R}^n$, preprint, University of Connecticut (1990).

Stephane G. Mallat, *A Theory for Multiresolution Signal Decomposition: The Wavelet Decomposition*, IEEE Transactions on Pattern Analysis and Machine Intelligence **11** (1989), 674–693.

H. Malvar, *Lapped transforms for efficient transform/subband coding*, IEEE Trans. Acoustics, Speech, and Signal Processing **38** (1990), 969–978.

David Marr, *Vision: a computational investigation into the human representation and processing of visual information*, W. H. Freeman, San Francisco, 1982.

Anonymous file transfer site, *pascal.math.yale.edu*, InterNet number [128.36.23.1].

Kannan Ramchandran and Martin Vetterli, *Efficient quantization for adaptive wave-packet tree structures*, preprint, Columbia University (June 1, 1991).

Kannan Ramchandran and Martin Vetterli, *Best wavelet packet bases in a rate distortion sense*, preprint, Columbia University (January 29, 1992).

Claude E. Shannon and Warren Weaver, *The Mathematical Theory of Communication*, The University of Illinois Press, Urbana, 1964.

James Andrew Storer, *Data compression: methods and theory*, ISBN 0-88175-161-8, Computer Science Press, Rockville, Maryland, 1988.

Robert Strichartz, *Affine self-similar tilings of the plane*, preprint, Cornell University, Ithaca, NY (1991).

Gregory K. Wallace, *The JPEG still picture compression standard*, Communications of the ACM **34** (April 1991), 30–44.

M. Victor Wickerhauser, *Acoustic signal compression with wavelet packets*, Wavelets–A Tutorial in Theory and Applications, C. K. Chui (ed.) ISBN 0-12-174590-2, Academic Press, Boston, 1992, pp. 679–700.

M. Victor Wickerhauser, *Fast approximate factor analysis*, Curves and Surfaces in Computer Vision and Graphics II, SPIE Proceedings Volume 1610, ISBN 0-8194-0747-X, Boston, 1991, pp. 23–32.

, *A universal algorithm for sequential data compression*, IEEE Transactions in Information Theory **23** (1977), 337–343.

, *Compression of individual sequences via variable-rate coding*, IEEE Transactions in Information Theory **24** (1978), 530–536.