Comparison of Picture Compression Methods: Wavelet, Wavelet Packet, and Local Cosine Transform Coding

Mladen Victor Wickerhauser*

Department of Mathematics
Washington University
St. Louis, Missouri 63130

Abstract

This is a survey of some new perspectives on transform coding image compression. We mentions some mathematical properties of sampling, entropy, and time-frequency localization of analysis-synthesis functions. Then we discuss several experiments in picture compression using wavelets, wavelet packets, local sines and cosines, and the adaptive "best-basis" method. We calculate the efficiency of several techniques to describe a fast transform from a library, with the aim of transforming each image into its own best-adapted coordinates for transmission with minimal overhead. Standard C algorithms for the local sine and cosine transforms are included in the appendix.

1 Introduction

In this summary I will describe several experiments in picture compression using wavelets and the local cosine transform of Coifman and Meyer. I will also describe an adaptive wavelet transform coding method and a local cosine transform algorithm based on the idea of a "best basis," and provide Standard C algorithms for computing some of the described transforms.

*Research supported in part by NFS Grant number DMS-9302828 and AFOSR contract number F49620-92-J-0106.

2 Relevant Notions from Mathematics

By a "picture" we will mean any function $S = S(x, y) \in l^2(\mathbf{Z}^2)$. In practice, of course, pictures will be nonzero only at finitely many points. Also, they must represent band-limited functions, since it is impossible to distinguish spatial frequencies above K cycles per unit distance from those below K if we use a sampling rate $2K$ samples per unit.

2.1 Sampled signals and approximation

Suppose $f \in L^2(\mathbf{R}^n)$ has d uniformly continuous derivatives, for $d \geq 0$, and that $\phi \in L^2(\mathbf{R})$ satisfies the additional conditions

$$\int_R \phi(x)\,dx = 0; \qquad \int_{\mathbf{R}} x^m \phi(x)\,dx = 0,\ 0 < m < d; \qquad \int_R x^d \phi(x)\,dx < \infty.$$

Then Taylor's theorem implies that the discrete values $f(k2^{-\nu})$ are very good approximations to the inner products $\langle f, \phi_{-\nu,k}\rangle$, for $k \in \mathbf{Z}^n$ and $\nu \in \mathbf{N}$. We obtain the estimate

$$\sup_{x \in I_{\nu,k}} |\langle f, \phi_{-\nu,k}\rangle - f(x)| < C2^{-\nu d},$$

where $I_{\nu,k} = \times \prod_{i=1}^n [2^{-\nu}k_i, 2^{-\nu}(k_i + 1)[$, and $0 < C < \infty$ may be chosen independently of ν and k.

By a *signal* we shall mean a compactly supported function belonging to $C^d(\mathbf{R}^n)$, which thereby satisfies the hypothesis for f above. For simplicity, we will at first consider the 1-dimensional case. By rescaling we may assume that $\sup_{x \in [k,k+1[} |\langle f, \phi_{0,k}\rangle - f(x)| < \epsilon$ for some acceptably small $\epsilon > 0$. If $d > 1$, then this error decreases faster than the number of samples grows.

2.2 The Heisenberg uncertainty principle

The functions underlying transform coding techniques can be judged by their simultaneous localization in space and wavenumber. This localization cannot be better than a universal fixed amount, as shown by Heisenberg's inequality or the uncertainty principle.

A nonzero function of compact support cannot also have a compactly supported Fourier transform. The Fourier transform of a compactly supported function is entire analytic, and an entire function which vanishes on an open set must be zero everywhere. Various refinements of this result exist, posing difficulties for time-frequency analysis.

Heisenberg is credited with observation that in quantum mechanics, one cannot simultaneously specify both the location and momentum of a wave function to arbitrary accuracy. The product of the location and momentum uncertainties must exceed the quantization h. Without developing the whole of quantum mechanics, we can sketch the proof of this fact and relate it to the present discussion of time-frequency atoms.

Let $\psi \in L^2(\mathbf{R})$ be a wave function, normalized with $\|\psi\|^2 = 1$. The location and momentum operators are X and P, respectively, where $X\psi(x) = x\psi(x)$ and $P\psi(x) = i\hbar \frac{d\psi(x)}{dx}$. Assume that the wave function has a derivative in L^2 and that $x\psi(x)$ is in L^2. Then the location and momentum are calculated from the wave function by $\langle \psi, X\psi \rangle$ and $\langle \psi, P\psi \rangle$, where $\langle \cdot, \cdot \rangle >$ is the inner product in $L^2(\mathbf{R})$. Location evaluates to the integral $\int_{\mathbf{R}} x|\psi(x)|^2\, dx$, which may be interpreted as an expectation $E(x)$, since $|\psi(x)|^2$ is a probability density function. Momentum may be also be interpreted as an expectation by first applying the Fourier transform, a unitary change of variable:

$$\int_{\mathbf{R}} \bar{\psi}(x) \left(\frac{1}{ih} \frac{d\psi(x)}{dx} \right) dx = i\hbar \int_{\mathbf{R}} \xi |\hat{\psi}(\xi)|^2\, dx = E(\xi).$$

The uncertainties of location and momentum are $\Delta X = \sqrt{E(x^2) - E(x)^2}$ and $\Delta P = \sqrt{E(\xi^2) - E(\xi)^2}$, respectively. Now we ask the question, what is the minimum value of $\Delta X \Delta P$? Observe that we are asking how well we can localize a function in L^2 simultaneously in time and frequency.

Without loss, we may assume that the spatial coordinates and initial phase are chosen so that $E(x) = 0$ and $E(\xi) = 0$. Define $A(\psi) = E(x)^2 E(\xi^2)$ in this case; any minimum ψ for the functional A is also a minimum for $\Delta X \Delta P$.

Using the calculus of variations, we can compute critical points for A. We look for a stationary ψ, namely a function for which $\delta A(\psi) = 0$. The component of this variation in the direction of an arbitrary perturbation function $\eta(x)$ in L^2 is computed below, where we have adjusted the time units so that $\hbar = 1$:

$$\begin{aligned}
\langle \delta A(\psi), \eta \rangle &= \frac{d}{d\epsilon} \left(\int_{\mathbf{R}} x^2 |\psi(x) + \epsilon\eta(x)|^2\, dx \times \int_{\hat{\mathbf{R}}} \xi^2 |\hat{\psi}(\xi) + \epsilon\hat{\eta}(\xi)|^2\, d\xi \right) \Bigg|_{\epsilon=0} \\
&= \left(\int_{\mathbf{R}} x^2 \left[\psi(x)\overline{\eta}(x) + \overline{\psi}(x)\eta(x) \right] dx \right) \times E(\xi^2) \\
&\qquad + E(x^2) \times \left(\int_{\hat{\mathbf{R}}} \xi^2 \left[\hat{\psi}(\xi)\overline{\hat{\eta}}(\xi) + \overline{\hat{\psi}}(\xi)\hat{\eta}(\xi) \right] d\xi \right) \\
&= 2\Re \left(E(\xi^2) \int_{\mathbf{R}} x^2 \psi(x)\overline{\eta}(x)\, dx + E(x^2) \int_{\hat{\mathbf{R}}} \xi^2 \hat{\psi}(\xi)\overline{\hat{\eta}}(\xi)\, d\xi \right) \\
&= \Re \langle \Delta P X^2 \psi + \Delta X P^2 \psi, \eta \rangle
\end{aligned}$$

Now $\delta A = 0$ implies that the last expression vanishes for all η, which implies that

$$\Delta P X^2 \psi + \Delta X P^2 \psi = 0.$$

2.3 The entropy of a sequence

Let $p = \{p_i : i = 0, 1, 2, \ldots\}$ be a discrete probability distribution function (pdf), that is, $0 \leq p_i \leq 1$ for all i and $\sum_i p_i = 1$. The *entropy* [SW64] of p, $H(p)$, is defined to be the following sum:

$$H(p) = \sum_i p_i \log \frac{1}{p_i}.$$

In this sum, $0 \log 0$ is taken to be 0. Since all the terms in the sum are positive, $H(p) \geq 0$. Roughly speaking, entropy measures the logarithm of the number of meaningful coefficients in the signal. It is not hard to show (see, for example, [Wic94]) that if only N of the values p_i are nonzero, then $H(p) \leq \log N$. Such a pdf is said to be concentrated into at most N values, but we can use entropy to define the theoretical dimension $d(p)$ of a pdf:

$$d(p) = e^{H(p)}.$$

This is a more general notion of concentration of a pdf, one that might be finite even if $p_i > 0$ for infinitely many values of i, so long as all but finitely many of those values are negligibly small.

Entropy and theoretical dimension are preserved by renumbering the probabilities p_i and thus we may assume that the pdf is strictly decreasing: $p_0 \geq p_1 \geq p_2 \geq \ldots$. We may measure the rate at which a pdf decreases in terms of its partial sum sequence $S_n = S_n(p) = \sum_{i=0}^{n} p_i$; since S_n increases and $\lim_{n\to\infty} S_n = 1$, we will say that the pdf p decreases faster than the pdf q if and only if $S_n(p) \geq S_n(q)$ for all $n = 0, 1, 2, \ldots$. Again, it is not hard to show (see [Wic94]) that if p decreases faster than q by this definition, then $H(p) \leq H(q)$ and thus $d(p) \leq d(q)$.

Unfortunately, the converse is not true because rate of decrease gives only a partial order on pdfs. There are pairs of pdfs p and q for which $H(p)$ is finite and $H(q)$ is infinite, but the rates of decrease are not comparable because $S_n(q)$ starts out by dominating $S_n(p)$ only to fall behind as $n \to \infty$. However, theoretical dimension is still useful as an indicator of which pdf has the longer "tail" of nonnegligible probabilities.

Now let $x = \{x_k : k = 1, 2, \ldots\}$ be an arbitrary sequence of positive integers. This is what we obtain, for example, after approximating a sequence of positive real numbers by their integer parts. There are two distinct notions of entropy for such sequences, which happen to be roughly equivalent in the special cases we will be considering. Both require forming a pdf $p = p(x)$ from the sequence x, but this is done in different ways.

The first notion is the traditional entropy of a source. We consider x_k to an independent Bernoulli trial of some process taking the value $i \geq 0$ with proability p_i. By the law of large numbers, this may be approximated by

$$\lim_{N\to\infty} \frac{\#\{1 \leq k \leq N : x_k = i\}}{N}.$$

Then $H(p)$ gives the lower bound for the expected number of bits per trial required to describe the sequence x in the most efficient alphabet, one which uses short code sequences for the most common values and longer sequences for rare events. (To be completely honest here, when talking about "bits" we must use $\log_2$ rather than log in the definition of entropy and 2 rather than e for the base of the exponential in the definition of theoretical dimension). If $H(p)$ is small, then x is concentrated into a few values i, with all other values being rare. If we renumber p into decreasing order, then the common values of x_k will be labeled 0, 1, etc., the "small" values. The "large" values will be rare; we expect few of them in any finite subsequence of x.

The second notion is geometric. We form probabilities from the x_k's themselves by letting $\|x\| = \left(\sum_k |x_k|^2\right)^{1/2}$ and defining $p_i = |x_i|^2/\|x\|^2$ whenever $\|x\|$ is finite. This obviously gives a pdf, and if $H(p)$ is small then we may conclude that large values of x_k are rare since there can only be a small number of k for which $|x_k|^2$ has a significant portion of the total energy $\|x\|^2$. This notion of entropy can be generalized to arbitrary real or complex valued sequences. Furthermore, we can calculate $H(p)$ directly from x with the following formula:

$$H(p) = \log \|x\|^2 + \frac{\ell(x)}{\|x\|^2}; \qquad \ell(x) = \sum_k |x_k|^2 \log \frac{1}{|x_k|^2}.$$

Note that $H(p)$ is monotonic with $\ell(x)$, so that they are equally useful for comparison purposes.

3 Transform Coding Methods

These work by performing a change of variable on the pixel values, with the expectation that most of the new coordinates can be discarded with minimal loss of image quality. The transforms must be invertible in exact arithmetic, but since some information will be discarded by rounding or quantization, they will not be perfectly invertible in practice.

3.1 Local cosine transform

Let $\{I_k\}$ be a collection of disjoint compact intervals of $\mathbf{R}$, indexed by the integers k. Coifman and Meyer showed how to construct an orthonormal basis of $L^2(\mathbf{R})$ subordinate to the partition $\mathbf{R} = \cup_k I_k$ in which the basis vectors are cosines multiplied by smooth bumps. For definiteness we will use a particular symmetric bump function

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

It is easy to see that 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 $\sin \pi x$: n iterations will yield at least 2^{n-1} vanishing derivatives at $-\frac{1}{2}$ and $\frac{3}{2}$.

We may translate and dilate the bump b onto an interval I_k by the formula $b_k(x) = b(\frac{x-a_k}{|I_k|})$, where a_k is the left endpoint of I_k. Let $c(n,x) = \sqrt{2}\cos \pi n x$, and consider the set of functions $c(n+\frac{1}{2}, x)$ for $n \geq 0$, $x \in [0,1]$. These form an orthonormal basis for $L^2([0,1])$. They may also be dilated and translated to the interval I_k by the formula $c_k(n,x) = \frac{1}{\sqrt{|I_k|}} c(n, \frac{x-a_k}{|I_k|})$. Now define $\psi_{nk}(x) = b_k(x)c_k(n+\frac{1}{2}, x)$ for integers $n \geq 0$ and k. These form an orthonormal basis for $L^2(\mathbf{R})$: the proof is a direct calculation. Cosine may be replaced by sine, and there are some other modifications possible, but this set of functions is sufficiently general for our present purposes.

Observe that ψ_{nk} is well localized in both space and frequency. In space, it is compactly supported on a proper subset of $3I_k$, which is defined as that interval centered at the center of I_k but having three times the width. In frequency, $\hat{\psi}_{nk}$ consists of two modulated bumps centered at $n+\frac{1}{2}$ and $-n-\frac{1}{2}$, respectively, with spread equal to that of $\hat{b}_k$. But $\hat{b}_k$ is well localized because b_k is smooth.

The construction generalizes to higher dimensions. For multi-indices n and k, define $\Psi_{nk}(x) = \psi_{n_1k_1}(x_1)\ldots\psi_{n_dk_d}(x_d)$ to obtain an orthonormal basis for $\mathbf{R}^d$ made of tensor products. Of course, it is possible to use a different partition in each dimension, as well as different bump functions. For the present exposition, we will concentrate on the special case of 2 dimensions, all intervals I_k of equal width, and always the same $b(x)$, as defined above. At the end of the paper, we will consider dyadic intervals with a restricted range of widths, as well.

3.2 Discrete local cosine transform

The functions ψ_{nk} have discrete analogues which form a basis of $l^2(\mathbf{Z})$, or $l^2(\mathbf{T})$. For the former, let I_k be a finite interval of integers with least element a_k, and let $|I_k|$ be the number of elements in I_k. The following vectors form a basis of $l^2(\mathbf{Z})$:

$$\psi_{nk}(j) = b_k(j+\frac{1}{2})c_k(n+\frac{1}{2}, j+\frac{1}{2}), \qquad \text{for integer } k \text{ and } 0 \leq n < |I_k|$$

Apart from the bells b_k, these are evidently the basis functions for the so-called DCT-IV transform. The particular bells chosen allow cosines on adjacent intervals to overlap while remaining orthogonal. A similar basis may be constructed over equally spaced points on the circle $\mathbf{T}$.

3.3 Implementation by folding

Rather than calculate inner products with the sequences ψ_{nk}, 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.

Suppose we wish to fold a function across 0, onto the intervals $[-1/2, 0)$ and $(0, 1/2]$, using the bell b defined above. Then folding replaces the function $f = f(x)$ with the left and right parts f_- and f_+:

$$\begin{aligned} f_-(x) &= b(-x)f(x) - b(x)f(-x), & \text{if } x \in [-\tfrac{1}{2}, 0); \\ f_+(x) &= b(x)f(x) + b(-x)f(-x), & \text{if } x \in (0, \tfrac{1}{2}]. \end{aligned}$$

The symmetry of b allows us to use $b(-x)$ instead of introducing the bell attached to the left interval.

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

$$f(x) = \begin{cases} b(x)f_+(-x) + b(-x)f_-(x), & \text{if } x \in [-\frac{1}{2}, 0); \\ \\ b(x)f_+(x) - b(-x)f_-(-x), & \text{if } x \in (0, \frac{1}{2}]. \end{cases}$$

Composing these relations yields $f(x) = \big(b(x)^2 + b(-x)^2\big)f(x)$, which is verified by the bell b defined above, for which the sum of the squares is 1. We can translate and dilate these relations to all adjacent pairs of intervals, and of course it works for sequences as well. These details are best understood by reading the comments in the computer program in Section 7

3.4 Block discrete local cosine transform coding

We propose a modification of the JPEG picture compression algorithm. We will replace the block discrete cosine transform (DCT) on 8x8 blocks with the block discrete local cosine transform (LCT). By the results of the previous section we expect to de-correlate adjacent samples about as well as with DCT, without introducing discontinuities at block boundaries. We can perform experiments of two types in order to evaluate this modification. First we calculate the rate distortion curves of some example pictures with this method and compare it to JPEG output. Results of one such calculation are attached to Figure 7. Second, we compare reconstructions from encodings at equal bit rates, but different methods, and judge them subjectively. This will verify the correlation between calculated and perceived distortion. One such comparison is available in Figures 8 through 24.

3.5 Adaptive block discrete local cosine transform coding

A further modification of the JPEG algorithm involves allowing the block size to grow if sections of the picture are regular on scales spanning many blocks. The main idea is that of the “best-basis” search algorithm, which compares the information cost of describing several adjacent blocks with that of describing their joint parent. In this case, the information cost is the entropy of the transformed coefficients, which can be computed for blocks of size 8x8, 16x16, 32x32, and so on up to the dimensions of the picture itself.

The cost of adding adaptivity to the transform coding scheme is a header describing the decomposition of the picture into blocks. Suppose for definiteness that we will consider only dyadic blocks of sizes 8x8 or bigger in a picture of size 256x256. Then each block needs no more than 2.585 bits to describe which of the 6 levels it comes from, and there are 1024 blocks, so the overhead of the adaptive scheme is 2647 bits for 65536 pixels, or about 0.04 bits per pixel. It remains to be seen whether this will be regained by the improved fit of the adapted basis.

4 Wavelet and Wavelet Packet Methods

We introduce a generalization of subband coding which may be used to compress digitized pictures or sequences of pictures. The new method uses 2-dimensional perfect reconstruction quadrature mirror filters (QMFs) to recursively decompose an image into finer and finer subbands. This produces an overabundant set of transform coefficients. The complete set of coefficients may be found in $O(N \log N)$ operations.

4.1 Wavelets and scaling or sampling functions

Let ϕ be a scaling function for a multiresolution decomposition of $L^2(\mathbf{R})$. Such a function may be constructed with an arbitrary finite number of vanishing moments by iteration of a sufficiently long finite impulse response quadrature mirror filter. Let $h = \{h_j\}_{j=-R}^{R-1}$ be such a filter. Then ϕ satisfies

$$\phi(x) = \sum_{j=-R}^{R-1} h_j \phi(2x - j).$$

As in an earlier chapter, we construct a mother wavelet ψ from ϕ by taking

$$\psi(x) = \sum_{j=-R}^{R-1} g_j \phi(2x - j),$$

where $g = \{g_j\}_{j=-R}^{R-1}$ is determined from h by the formula $g_j = (-1)^j h_{-(j+1)}$.

The wavelet expansion $f = \sum_{\nu,k} \langle f, \psi_{\nu,k}\rangle \psi_{\nu,k}$ is calculated by the pyramid scheme of Mallat. Write $s_{\nu,k} = \langle f, \phi_{\nu,k}\rangle$, and $d_{\nu,k} = \langle f, \psi_{\nu,k}\rangle$. Denote by H and

G the convolution-decimations with filters h and g, respectively. We calculate:

$$\begin{tikzcd}
f(k) \approx \langle f, \phi_{0k} \rangle \arrow[r, "I"] & s_{0,k} \arrow[r, "H"] \arrow[d, "G"'] & s_{1,k} \arrow[r, "H"] \arrow[d, "G"'] & s_{2,k} \arrow[r, "H"] \arrow[d, "G"'] & \cdots \\
& d_{1,k} & d_{2,k} & d_{3,k} &
\end{tikzcd}$$

Each arrow costs $2R$ multiplications per coefficient.

If f is supported in $[1, N]$, then a sequence at scale ν has $N2^{-\nu}$ nonzero coefficients. We may assume for technical simplicity that N is a positive power of 2. Hence the pyramid reaches its top after $\log_2 N$ scales, requiring $4R(N-1)$ multiplications. Since R is independent of N (and is typically much smaller), this algorithm's complexity is linear. It yields N coefficients $\{d_{\nu,k} : 1 \leq \nu \leq \log_2 N, 1 \leq k \leq N2^{-\nu}\} \cup \{s_{(\log_2 N),1}\}$.

By the orthogonality of quadrature mirror filters, we have

$$\sum_{k=1}^{N} |\langle f, \phi_{0k} \rangle|^2 = |s_{(\log_2 N),1}|^2 + \sum_{\nu=1}^{\log_2 N} \sum_{k=1}^{N2^{-\nu}} |d_{\nu,k}|^2$$

Denote by $s'_{\nu,k}$ and $d'_{\nu,k}$ the coefficients obtained from the pyramid scheme using $f(k)$ rather than $\langle f, \phi_{0k} \rangle$. If $|s'_{0k} - s_{0k}|^2 < \epsilon$ for $1 \leq k \leq N$, then $|d'_{\nu,k} - d_{\nu,k}|^2 < \epsilon N$ for $1 \leq \nu \leq \log_2 N$ and $1 \leq k \leq N2^{-\nu}$. If f has more than one derivative and we use wavelets with more than one vanishing moment, we can arrange that $\epsilon N \to 0$ as f is rescaled and $N \to \infty$.

Suppose that f is a signal with d continuous derivatives, and ψ is a mother wavelet with at least d vanishing moments. Then f is in the Sobolev space L^2_d, and its Littlewood-Paley characterization yields the formula:

$$\sum_{\nu=-\infty}^{\infty} \left(2^{-2\nu d} \sum_k |\langle f, \psi_{\nu k} \rangle|^2 \right) < \|f\|^2_{2,d}$$

Since f has compact support, there is some sufficiently large ν above which $\sum_k |\langle f, \psi_{\nu k} \rangle|^2 < C2^{\nu}$. This part of the series sums to a constant if $2d > 1$.

The sum converges, so that $|\langle f, \psi_{\nu k} \rangle|^2 < 2^{2\nu d} \|f\|^2_{2,d}$ as $\nu \to -\infty$. On the other hand, there are at most $C2^{-\nu}$ translates $\psi_{\nu k}$ for which $\langle f, \psi_{\nu k} \rangle$ is nonzero. We deduce that a decreasing rearrangement of the wavelet coefficients of f, written $\{c_j\}_{j=1}^{\infty}$ with corresponding wavelet ψ_j, will satisfy $|c_j|^2 < Cj^{-2d}$. This gives an estimate

$$\left\| f - \sum_{|c_j|^2 \geq \epsilon} c_j \psi_j \right\|^2 = \sum_{|c_j|^2 < \epsilon} |c_j|^2 \quad < \sum_{j > (\frac{\epsilon}{C})^{-1/2d}} j^{-2d} \quad < \quad C' \epsilon^{1-\frac{1}{2d}}.$$

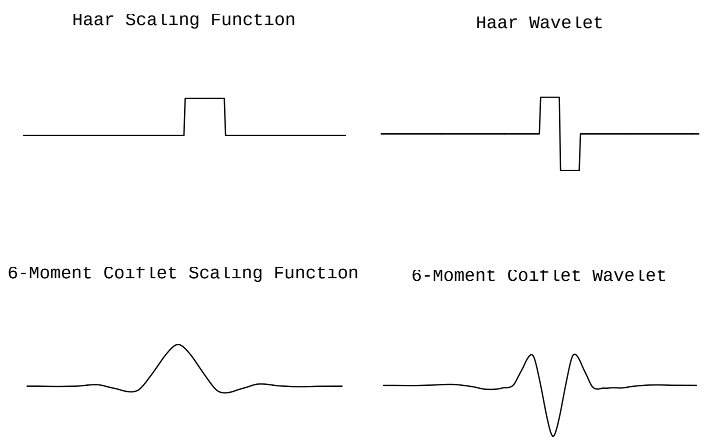

Figure 1: Haar and Coiflet wavelets and scaling functions.

Such an estimate is also valid for the finite sequences which arise when sampling a function supported on $[1, N]$. The N samples $\{f(j)\}_{j=1}^{N}$ may be compressed into a smaller number by replacing them with the largest few coefficients c_j. For a given error ϵ, we take the top $N' = (C/\epsilon)^{1/2d}$ coefficients. For very smooth signals where C is small and d large, we will obtain $N' \ll N$.

To compress a signal with d continuous derivatives, it is optimal to use an analyzing wavelet with d vanishing moments. With too few, the wavelet coefficients will not decrease as fast as possible. With too many, calculating the coefficients by filter convolution-decimation will become too expensive.

Figure 1 shows two wavelets and their scaling functions. The Haar wavelet has one vanishing moment, and has been used for a long time to represent nonsmooth (digital) signals. The "Coiflet" of 6 vanishing moments is a recent discovery, and is part of a family of almost symmetric smooth wavelets well suited to certain smooth signals.

4.2 Subband coding

A signal may be divided into frequency subbands by repeated application of convolution and decimation operators. For a 2-dimensional signal or "picture," these are usually tensor products of one-dimensional QMFs (separable filters), as defined below. There has also been considerable recent interest in nonseparable filters [CD92], for which the following discussion is equally valid, but for simplicity we used only separable filters in our fingerprint compression experiments.

Let $\{h_k\}, \{g_k\}$ belong to l^1, and define two decimating convolution operators

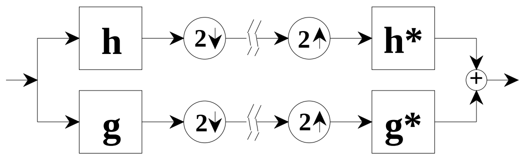

Figure 2: Block diagram of convolution-decimation and reconstruction

$H : l^2 \to l^2$, $G : l^2 \to l^2$ as follows:

$$Hf_k = \sum_{j=-\infty}^{\infty} h_j f_{j+2k}, \qquad Gf_k = \sum_{j=-\infty}^{\infty} g_j f_{j+2k}.$$

H and G are called *quadrature mirror filters (or QMFs)* if they satisfy an orthogonality condition:

$$HG^* = GH^* = 0,$$

where H^* denotes the adjoint of H, and G^* the adjoint of G. They are further called *perfect reconstruction filters* if they satisfy the condition

$$H^*H + G^*G = I,$$

where 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

$$m_0(\xi) = \sum_{k=-\infty}^{\infty} h_k e^{ik\xi}, \qquad m_1(\xi) = \sum_{k=-\infty}^{\infty} g_k e^{ik\xi}.$$

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

$$\begin{pmatrix} m_0(\xi) & m_0(\xi+\pi) \\ m_1(\xi) & m_1(\xi+\pi) \end{pmatrix}$$

This fact is proved in [Dau88].

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

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

$$F_0 \overset{\text{def}}{=} H \otimes H, \qquad F_0 v(x,y) \;=\; \sum_{i,j} v(i,j) h_{2x+i} h_{2y+j}$$

$$\begin{aligned}
F_1 &\overset{\text{def}}{=} H \otimes G, & F_1 v(x,y) &= \sum_{i,j} v(i,j) h_{2x+i} g_{2y+j} \\
F_2 &\overset{\text{def}}{=} G \otimes H, & F_2 v(x,y) &= \sum_{i,j} v(i,j) g_{2x+i} h_{2y+j} \\
F_3 &\overset{\text{def}}{=} G \otimes G, & F_3 v(x,y) &= \sum_{i,j} v(i,j) h_{2x+i} h_{2y+j}
\end{aligned}$$

These convolution-decimations have the following adjoints:

$$\begin{aligned}
F_0^* v(x,y) &= \sum_{i,j} v(i,j) h_{2i+x} h_{2j+y} \\
F_1^* v(x,y) &= \sum_{i,j} v(i,j) h_{2i+x} g_{2j+y} \\
F_2^* v(x,y) &= \sum_{i,j} v(i,j) g_{2i+x} h_{2j+y} \\
F_3^* v(x,y) &= \sum_{i,j} v(i,j) h_{2i+x} h_{2j+y}
\end{aligned}$$

The orthogonality relations for this collection are as follows:

$$F_n F_m^* = \delta_{nm} I; \quad F_0^* F_0 \oplus F_1^* F_1 \oplus F_2^* F_2 \oplus F_3^* F_3 = I$$

The space $l^2(\mathbf{Z}^2)$ of pictures may be decomposed into a partially ordered set $\mathbf{W}$ of subspaces $W(n,m)$, where $m \geq 0$, and $0 \leq n < 4^m$. These are the images of orthogonal projections composed of products of convolution-decimations. Put $W(0,0) = l^2$, and define recursively

$$W(4n+i, m+1) = F_{2i+j}^* F_{2i+j} W(n,m) \qquad \text{for } i = 0,1,2,3.$$

These subspaces may be partially ordered by a relation which we define recursively as well. 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 $V_1, \ldots, V_n$ of subspaces in $\mathbf{W}$ such that $W \prec V_1 \prec \ldots \prec V_n \prec W'$. 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,

$$W = F_0^* F_0 W \oplus F_1^* F_1 W \oplus F_2^* F_2 W \oplus F_3^* F_3 W$$

The right hand side contains all the children of W.

As in [Wic93], the coordinates with respect to the standard basis of $W(n,m)$ are in $F_{(1)} \ldots F_{(m)} W(0,0)$, where the particular filters $F_{(1)} \ldots F_{(m)}$ are determined uniquely by n. Therefore we can express in standard coordinates the

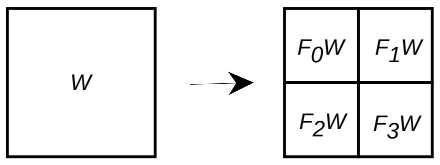

Figure 3: Four child subbands of a picture.

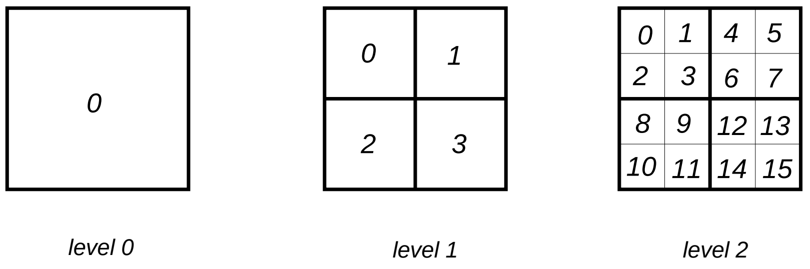

Figure 4: Ancestor subband and two generations of descendents.

orthogonal projections of $W(0,0)$ onto the complete tree of subspaces $\mathbf{W}$ by recursively convolving and decimating with the filters.

We may think of the quadrature mirror filters H and G as low-pass and high-pass filters, respectively. Their tensor products form a partition of unity in the Fourier transform space. 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 Figure 3.

Figure 4 shows 2 generations of descendents, labeled as the complete decomposition of $\mathbf{R}^4 \times \mathbf{R}^4$. 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.

4.3 Two-dimensional wavelet packets

An earlier paper [CMQW90] presented a method for generating a library of orthonormal basis by recursive QMF convolution-decimation. From an algorithmic standpoint, this is equivalent to recursive subband coding while retaining all intermediate subband decompositions. The coefficients produced at each stage are correlations of the signal with compactly-supported oscillatory functions called “wavelet packets.” The analytic properties of these functions have been

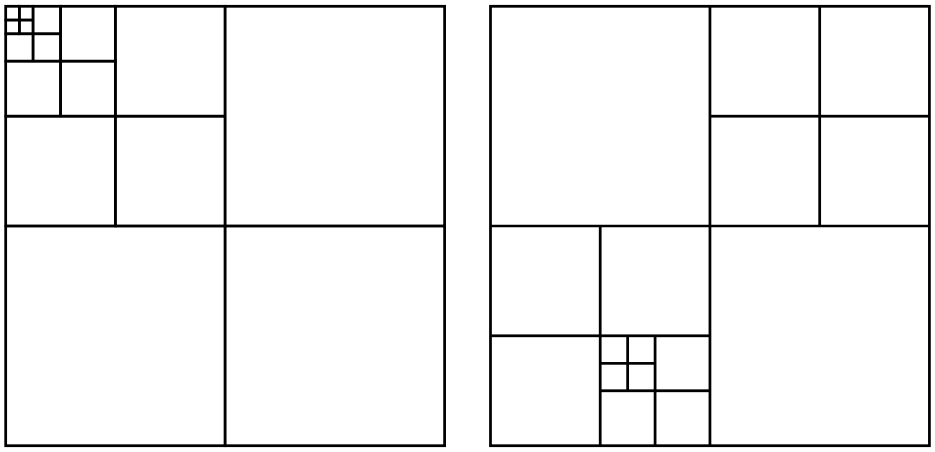
Figure 5: Octave and custom subbands.

extensively studied [CMW92b], [CMW92a]. If perfect reconstruction QMFs are used, then the wavelet packets satisfy some remarkable orthogonality properties.

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 decomposition of the unit square. Classical subband coding takes coefficients from a fixed set of subbands, usually from a single level of the quadtree in Figure 6. Wavelet transform coding also extracts coefficients from a fixed collection of blocks, the octave subbands, which are schematically represented in the left part of Figure 5. The right hand part shows a typical dyadic decomposition down to level 4; its basis subset consists of the 16 pairs (0,1), (3,1), (4,2), (5,2), (6,2), (7,2), (8,2), (9, 2), (10,2), (45,3), (46,3), (47,3), (176,4), (177,4), (178,4), and (179,4). We may call such a basis subset B.

Proposition 1 *The number of basis subsets grows exponentially with the size of the tree* $\mathbf{W}$*, which grows like* $O(N \log N)$ *with the number of pixels* N*.*

Proof: Let A_n be the number of bases in the library corresponding to a tree of $1 + n$ levels, namely levels $0 \ldots n$. A decomposition to level n is only possible for a picture of size at least $N = 4^n$ pixels. Then $A_0 = 1$, and we can calculate $A_{n+1} = 1 + A_n^4$, namely the root and combinations of the 4 independent children subtrees with A_n bases each. Simplifying this by discarding the 1 gives the estimate $A_{n+1} \geq 2^{4^n} = 2^N$. $\square$

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

$$\sum_{W \in B} H_W(S)$$

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

Some examples of information cost functions are listed in [CW92]. 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 components of S in the direction of the standard basis vectors 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 coefficients above the threshold.

Set a predetermined deepest level L. Label each subspace at level L as "kept," 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 coefficients 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. 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.

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

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

with S_W as above. This is related to the classical measure of the concentration of a probability distribution function.

4.4 Wavelet packets and the best basis

The "best-basis" method is to select a most efficient orthogonal representation of the picture from among these coefficients, using an efficiency functional chosen according to the application. Compression occurs when the coefficients are quantized and coded to remove redundancy, then stored together with an efficient representation of the chosen basis.

The overabundant set of coefficients is naturally organized into a quadtree of subspaces by frequency. Every connected subtree containing the root corresponds to a different orthonormal basis, and the totality of such bases forms a "library" of fast transforms. The most efficient of all the transforms in the library may be found by recursive comparison. The efficiency functionals are characterized by additivity across orthogonal decompositions. For such functionals, the choice algorithm will find the global minimum in $O(N)$ operations, where N is the number of pixels in the image.

The advantages of this method over traditional subband coding or fixed wavelet transform methods is that the basis is chosen automatically to best represent the particular image. Hence the name "best-basis." In this sense the transform is highly nonlinear. However, the transform is well-conditioned in the following sense: the basis choice has an exact representation and contributes no error, while the coefficient transform is orthonormal, i.e., has condition number 1. Best-basis differs from statistical compression methods in that no statistical model of the ensemble of images is used. Compression can occur because pixel values are correlated by the smoothness of the sampled band-limited image. The exact nature of the correlation need not be known *a priori*, and deviant images will be automatically represented by deviant best bases.

Some related adaptive methods are adaptive quantization subband coding of images [RV91], and adaptive vector quantization of wavelet coefficients [MBA90].

We present some results of using the method to compress high-resolution fingerprint images, using various notions of entropy as the efficiency functional and using mean-square deviation as the error criterion.

4.5 Compressing a best-basis representation

Suppose that B is a best-basis subset of W, chosen by counting coefficients above a predetermined threshold. We may then extract just these non-negligible coefficients and transmit them, together with their locations in the tree. This number of coefficients is no greater than the number of pixels, since it is chosen after comparison with the original basis, among others. Define the *compression ratio* to be the ratio of retained coefficients to the original pixels. With thresholding 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 coefficients as we can afford. This may be accomplished by first sorting into decreasing order by absolute value, then reading off the desired number of coefficients. Alternatively, since we know in advance how many coefficients we can use, it may be more efficient to bubble up the top few coefficients and discard the rest of the array. The second method is better if the number of retained coefficients is less than $\log N$, where N is the number of pixels.

For 2-dimensional signals (i.e., pictures), we may use 3 analyzing wavelets and one scaling function formed from tensor products of 1-dimensional functions. Define

$$\begin{aligned}
ss_{\nu k} &= \int_{\mathbf{R}^2} f(x,y)\phi_{-\nu,k_x}(x)\phi_{-\nu,k_y}(y)\,dxdy,\\
sd_{\nu k} &= \int_{\mathbf{R}^2} f(x,y)\phi_{-\nu,k_x}(x)\psi_{-\nu,k_y}(y)\,dxdy,\\
ds_{\nu k} &= \int_{\mathbf{R}^2} f(x,y)\psi_{-\nu,k_x}(x)\phi_{-\nu,k_y}(y)\,dxdy,\\
dd_{\nu k} &= \int_{\mathbf{R}^2} f(x,y)\psi_{-\nu,k_x}(x)\psi_{-\nu,k_y}(y)\,dxdy.
\end{aligned}$$

The 2-dimensional pyramid scheme is then:

$$\begin{array}{lcccccl}
f(k) \approx \langle f\,,\,\phi_{0k_x}\otimes\phi_{0k_y}\rangle & \xrightarrow{I} & ss_{0k} & \xrightarrow{H\otimes H} & ss_{1k} & \xrightarrow{H\otimes H} & \cdots\\
H\otimes G, H\otimes G, G\otimes G: & & \downarrow\downarrow\downarrow & & \downarrow\downarrow\downarrow & & \\
 & & sd_{1k},\ldots & & sd_{2k},\ldots & &
\end{array}$$

Each arrow costs $(2R)^2$ multiplications per coefficient. The total complexity is $(2R)^2(2N-2)^2$, which is linear in the number of samples. The algorithm may be readily applied to million-sample signals, or pictures of 1024×1024 pixels, using a contemporary desk top computer.

4.6 Relationships among the methods

The wavelet basis is one particular basis subset of $\mathbf{W}$, namely all elements in the standard bases of the subspaces $W(1,m)$, $W(2,m)$, and $W(3,m)$ for $m = 1,2,\ldots$, together with $W(0,L)$ if we fix a deepest level L. Numerous other authors have investigated representing signals in the wavelet basis; a small sampling of this work is the papers [CMQW90, BBH93, DrJL92, Don93, MF91, MBA90, Wic92, ZSW91]. While this will obviously yield no better choice of basis than the "best-basis," the savings in computation and the absence of side information may warrant its consideration when the signals are of a particular class.

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 labeled 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 [Wic94].

4.7 Efficient coding of best-basis coefficients

The transformation from a 2-dimensional signal to its best-basis representation is nonlinear since the choice of basis depends upon the signal itself. Together with the coefficients we must include the extra information describing which basis was used, and we must somehow indicate which coefficient each quantized value represents.

There are at least 2 ways to include this latter information. Best-basis coefficients may be individually tagged with their coordinates in the best-basis tree. Suppose this tree 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 coefficients, 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 [pasnt].

Alternatively, we may agree upon an ordering of the coefficients, write out the coefficients in this order after quantization, and then entropy code the entire list. If we were to use a single basis like wavelets or DCT, then we need never explicitly tag any coefficients. We obtain compression because the quantized stream of coefficients has a lower entropy that the original stream of bytes.

We shall use a variation of the second method to code best basis elements. Namely, we will include some side information which describes the chosen basis, and we shall then write all the (quantized) coefficients from that basis out into a stream for entropy coding. This method is substantially more efficient, and is essential for a competitive picture compression algorithm.

4.7.1 Describing the basis, quantizing all the coefficients

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 $0 \leq m \leq L \leq \log_2 N$. Picture these arrays stacked one atop the other, as in Figure 6 below:

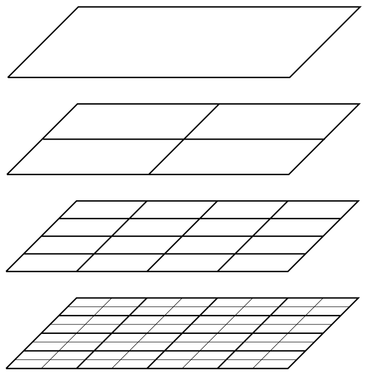

Figure 6: Quadtree of two-dimensional wavelet packet subbands.

Suppose that from this collection of arrays we have chosen a best basis. This will be a subset of the coefficients having the property that if one element of a subspaces is in the basis, then that whole subspace is in the basis. Also, if a subspace is in the basis, then none of its descendent or ancestor subspaces are in the basis. Such a subset can be identified with a cover by dyadic subarrays. Looking down through the stack of arrays, 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.

4.7.2 Levels map

In this scheme, we use 2 arrays to describe the transformation, a "levels map" and a "coefficients list." The first has 2^{2L} integers of length $\log_2(1 + L)$ bits each, and describes the level from which a corresponding $N2^{-L} \times N2^{-L}$ block of coefficients in the next list was chosen. The second array contains the $N \times N$ coefficients from the best basis, scanned row-by-row or in some other agreed-upon pattern. We illustrate with 3 examples:

If the original signal turns out to be the best-basis representation, then every coefficient will be chosen from level 0, the levels map will consist of $2^L \times 2^L$ 0's, and the coefficients list will contain the original signal in the order it was scanned.

If the complete bottom level L turns out to give the best basis, then the levels map will contain $2^L \times 2^L$ L's and the coefficients list will contain all the coefficients from the bottom level L, scanned in some canonical order.

If the wavelet basis turns out to be the best, then the levels map will contain a description of the wavelet basis, and the coefficients list will contain the wavelet coefficients of the signal. To illustrate, suppose that $N = 16$, $L = 3$, and the signal is in fact the scaling function of amplitude 1, level 3 and position

$(0, 0)$. Then the levels map and the coefficients list will look respectively like the following:

```
1 1 1 1 1 1 1 1           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 2 2 2 2           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 2 2 2 2           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 2 2 3 3           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 2 2 3 3           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
                          0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

The number of coefficients and side data is large, but the information content is low, and entropy coding of this list will greatly shorten it.

We can obtain an estimate of the overhead cost of this method prior to entropy coding. There are 4^L additional integers each of length $\log_2(1+L)$ bits. This gives a total of $4^L \log_2(1+L)/N^2$ bits per pixel, which we can control by making $L < \log_2 N$.

4.7.3 Subspace lists

A basis may also be described by listing the subspaces it contains. One method is to list subspaces by level. We use 3 arrays:

1. an array `num[`L`]` of integers giving the number of subspaces chosen at each level,
2. an array of arrays `subspace[][]` listing the subspaces chosen from each level, and
3. an array containing the complete set of chosen coefficients.

The array `num[]` in item 1 contains L entries of varying length: 1 bit for entry 0, which describes whether level 0 is chosen; $2m$ bits for entry m which tells how many of the 4^m subspaces on level m are in the best-basis; and so on up to $2L$ bits for level L. Together the subspaces must account for all of the coefficients, so we have the relation:

$$\texttt{num[0]} * N^2 + \texttt{num[1]} * N^2/4 + \texttt{num[2]} * N^2/4^2 + \ldots + \texttt{num[}L\texttt{]} * N^2/4^L = N^2,$$

which implies that $\texttt{num[}L\texttt{]} = 4^L(1 - \texttt{num[0]} - \ldots - \texttt{num[}L-1\texttt{]}/4^{L-1})$, so it is not necessary to transmit this value to describe the basis. The total number of bits in the array `num[]` is thus $1 + 2\sum_{m=1}^{L-1} m = 1 + (L-1)L = L^2 - L + 1$.

The array `subspace[]` of arrays in item 2 contains L entries of length `num[0]`, `num[1]`, ..., `num[L-1]`, respectively. Array `subspace[m][]` contains integers of length $2m$ bits; array `subspace[0]` need not be allocated, since there is a unique subspace at level 0. Also, array `subspace[L][]` need not be allocated. Suppose that the subspace numbering scheme assigns the indices $4k$, $4k+1$, $4k+2$, and $4k+3$ to the subspaces descended from k. The subspaces in level L will be labeled by the integers $0, \ldots, 4^L$, and the ones which are actually present are the survivors after the indices $4^{L-m}k, \ldots, 4^{L-m}(k+1)-1$ are deleted for each subspace k at level $0 \leq m < L$.

With this numbering scheme for the subspaces, the wavelet basis of the example above would be represented by `num[]`$=\{0, 3, 3\}$, and the following subspaces list:

```
subspace[1][] = { 0, 1, 2 }
subspace[2][] = { 12, 13, 14 }
```

The coefficients list will look the same as the above.

Hence the total number of bits in `subspace[][]` is

$$\sum_{m=1}^{L-1} 2m * \texttt{num[}m\texttt{]} \leq 2(L-1)4^{L-1}.$$

This coding method requires an extra $(L^2 - L + 1 + 2(L-1)4^{L-1})/N^2$ bits per pixel, which we again control by limiting L.

4.7.4 Coding the tree

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 `next[]` of integers which describe at which level the next node in the best basis resides. The nodes themselves are traversed in depth-first order, and the level of the next node is determined by a very short integer of only $log_2 L$ bits.

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 $m, \ldots, L$. In an extreme case of this phenomenon, the first 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, since level 0 (the root of the tree, or original signal) is essentially meaningless we can use that value instead of L to represent the deepest level, thus using only the range $0, \ldots, L-1$ in the coding of the tree.

In the wavelet basis case above, the depth-first-search encounter order list would be $\{1, 1, 1, 2, 1, 1, 0\}$. We can either write the coefficients list in the same manner as before, or we can dump the coefficients from each subspace as it is encountered in depth-first order. In the example case, that would produce a stream of small square arrays:

```
0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0     0 0 0 0 0 0 0 0

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

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 in the bottom row is chosen and the description is `next[]` $= \{0, 0, \ldots, 0\}$ (4^{L-1} 0's), or when the next-to-bottom row is chosen and we have `next[]`$=\{L-1, \ldots, L-1\}$ (4^{L-1} $L-1$'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.

4.8 Reconstruction

A picture represented as coefficients may be reconstructed by calculating the value at each point of the appropriate linear combination of wavelet packets. We cascade this computation for efficiency, as follows. First we allocate enough memory for the deepest level of the tree of subspaces that contains retained coefficients, and insert the coefficients into their appropriate locations. Then we reconstruct the parent subspaces by applying the adjoints of the convolution and decimation operators, which produces part of the next deepest level. Into this we add the retained coefficients which belong in that level, at their respective locations, and reconstruct the parents of this level. We continue in this manner until we have reconstructed the root, which now contains the picture.

4.9 Operation counts

Suppose that S is an N-element picture. Applying convolution-decimations to generate the tree of coefficient sequences requires $O(N \log N)$ operations. Calculating information costs has complexity $O(N \log N)$. 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 coefficients from the best basis takes an additional $O(N)$ operations.

We can perform a radix sort to determine the largest coefficients in the output register: this has complexity $O(N \log N)$. The alternative, extracting the top t coefficients, requires $O(tN)$ operations. We choose the more efficient method: in either case the total complexity of the compression algorithm is $O(N \log N)$.

Reconstruction from the retained coefficients has the same complexity as generating all the coefficients, since we must in general reproduce the entire tree, and the convolution-decimations have the same complexity as their adjoints. The constant in $O(N \log N)$ is smaller because no additional steps (such as searching for a best basis) are needed during reconstruction.

4.10 Tricks and optimizations

There are several enhancements commonly used in practice. Prior to sorting, wavelet packet coefficients can be weighted by their visibility, a psychophysiological observable measuring the human eye's spatial frequency response. They must then be unweighted prior to the reconstruction of the signal. This permutes the decreasing rearrangement so as to minimize the weighted error. Note that such bounded weighting amounts to conjugating the compression projection by a Calderón–Zygmund operator.

It is also possible to optimize the storage of the selected wavelet packet coefficients. For example, not all coefficients need to be quantized at the same precision. The relative quantizations are determined by experiment. Finally, the resulting bit stream can be entropy-coded to minimize the number of bits per picture.

5 How to Compare Coding Methods

All "lossy" transform coding methods introduce errors and distortion, but different methods produced radically different kinds and quantities of artifacts. Even when the errors can be quantified, their magnitude may not adequately describe the "visibility" of the artifacts they represent, so it is always necessary to judge the quality of a lossy compression through experiments involving its ultimate consumer. In the case of images to be examined by humans, this might be done with subjective evaluations by a large number of observers. In the case of machine analyzed images, a coding method may be judged by its transparency: how little it distorts subsequent computations.

5.1 Transform coding gain

One way to judge a compression method is by its effectiveness at concentrating pixel energy into a small number of codewords. This concentration is called "coding gain" and it can be quantified using yet another notion of entropy.

Suppose $\{f_n\}_{n=1}^N$ is a finite time series with autocovariance matrix $m_{ij} = \rho^{|i-j|}$, where ρ is close to 1. This corresponds to a first-order Gauss-Markov process with adjacent correlation coefficient ρ; such time series have been used to model pictures and speech signals for purposes of comparing transform coding schemes. An ideal scheme would de-correlate the samples, or equivalently diagonalize m; this is done by the Karhunen-Loéve transformation, which is impractically slow for our purposes. Instead, we shall use fast orthogonal transformations and measure the quality of approximate diagonalization by the so-called energy compaction or maximum transform coding gain, which is defined as

$$E(m) = \frac{\frac{1}{N}\sum_{i=1}^N m_{ii}}{(\prod_{i=1}^N m_{ii})^{1/N}}$$

If m is diagonal, then the time series coefficients are independent random variables with variances m_{ii}. The entropy of the source of the time series is then $H(m) = \sum_i \log m_{ii}$, and in this case $E(m)$ is seen to be a decreasing function of $H(m)$.

Now let U and V be orthogonal matrices, whose columns may be considered an orthonormal basis. We call U a better basis than V if $E(UmU^*) > E(VmV^*)$, or equivalently if $H(UmU^*) < H(VmV^*)$. For $\rho = 0.95$ and a matrix of order 32, the energy compaction for the local cosine transform is 9.47, while that for the Karhunen-Loéve basis is 9.54. By contrast, the discrete cosine transform gives 9.43.

5.2 Rate-distortion curves

Since the coefficients produced by the transform coding method will ultimately be quantized or approximated by a small number of chosen values, methods can be compared by the signal to noise ratio (distortion) produced by quantization and entropy coding giving a particular compression ratio (rate). Each coding method thereby yields a rate-distortion curve for each image, and one method is better than another if its rate-distortion curves consistently or on average have lower distortion at the desired rates.

Below are the sample picture and rate-distortion curves for various coding methods applied to a typical picture intended for human consumption:

5.3 Reduction in entropy

Write S for the original picture sequence and S' for the sequence written in the best-basis. We may ask what is the expected reduction in entropy obtained by going from S to S'. This is equivalent to evaluating

$$\int_{|x|^2=1} [H(x') - H(x)]\, d\omega(x),$$

where x' is x written in best-basis coordinates, and $\omega(x)$ is normalized surface measure on the $(N-1)$-sphere.

The simplest example is a 2-point signal $s = (s_0, s_1)$, periodized, so that the only available filters are the Haar filters $\sqrt{2}p = \{1, 1\}$, $\sqrt{2}q = \{1, -1\}$. If we suppose that $\|s\| = 1$, then the original entropy is just $H(s) = -|s_0|^2 \log |s_0|^2 - |s_1|^2 \log |s_1|^2$. The only other basis gives the coefficients $s'_0 = (s_0 + s_1)/2$, $s'_1 = (s_0 - s_1)/2$, with corresponding entropy $H(s') = -|s'_0|^2 \log |s'_0|^2 - |s'_1|^2 \log |s'_1|^2$. We parameterize $s_0 = \cos x$, $s_1 = \sin x$ and evaluate the integral numerically to get

$$\frac{1}{2\pi} \int_0^{2\pi} [H(s) - H(s')]\, dx = \frac{2}{\pi} \int_{\pi/8}^{3\pi/8} [H(s) - H(s')]\, dx = -0.210393231149915$$

Now the expected entropy of a 2-point distribution is

$$\frac{1}{2\pi} \int_0^{2\pi} H(s)\, dx = 0.3862942192715892$$

so that the expected entropy in the best basis is 0.1759009881216742.

Roughly speaking, entropy measures the logarithm of the number of meaningful coefficients in the signal. The above entropies correspond to 1.47 and 1.19 coefficients by this analogy, a reduction of 19%.

We have computed the entropy of the picture, and its entropy in the best basis obtainable with filters of 1, 2, 3, 5, and 10 vanishing moments. Hence we have also computed the exponential function of the entropy. These results are listed in Table 2 below.

5.4 Subjective evaluation

For the subjective evaluation, Figure 8 compare the perceived distortion for two methods which have the same distortion energy.

Figure 9 is a 256×256-pixel computer-rendered picture, created by Craig Kolb with a ray-tracing program. Pixels are stored as 8-bit integers.

The remaining pictures in Figures 10 to 24 are reconstructed from compressions performed with the author's implementation of the 2-dimensional pyramid algorithm. Coefficients are sorted then selected in decreasing order of absolute value. The definition of compression ratio as it is used in the captions is the number of pixels divided by the number of coefficients used to represent the picture.

Filter Moments	Compression Ratio	MSE (%)	MSE (dB)
10	100	6.58048	11
10	50	3.85890	14
10	20	1.54101	18
10	10	0.61173	22
10	5	0.16766	27
10	2	0.00755	41
5	100	5.87643	12
5	50	3.39416	14
5	20	1.34303	18
5	10	0.50403	22
5	5	0.12961	28
5	2	0.00376	44
3	100	6.18733	12
3	50	3.51632	14
3	20	1.36315	18
3	10	0.50358	22
3	5	0.12848	28
3	2	0.00332	44
2	100	6.32141	11
2	50	3.70738	14
2	20	1.46627	18
2	10	0.54608	22
2	5	0.14227	28
2	2	0.00370	44
1	100	7.36173	11
1	50	4.38014	13
1	20	1.83089	17
1	10	0.73331	21
1	5	0.18761	27
1	2	0.00317	44

Table 1: Rate versus distortion for compressions with Daubechies filters.

Filter Moments	Entropy	Reduction	Theoretical Dimension)
(original)	9.449	-	12700
10	4.371	-5.078	79
5	4.265	-5.184	71
3	4.397	-5.052	81
2	4.313	-5.136	75
1	4.345	-5.104	77

Table 2: Entropy reduction by wavelet transformation.

6 Transforming compressed pictures

As described in [CW93], the wavelet packet coefficients of a picture contain a mixture of spatial and spectral information. 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 coefficients. Some examples include spatial filtering and local image enhancement, edge and texture detection, 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 a coefficient in the complete wavelet packet expansion of a picture S. Here $m = 0,1,\ldots,L$. We may divide $0 \leq n < 4^m$ into n_x and n_y by taking the odd and even bits in its binary expansion, respectively. These can be arranged in "sequency" order (by Gray-encoding; see [Wic94]) 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 (n_x,n_y,m,k_x,k_y) as follows:

Spatial filtering: to remove (or attenuate) high frequency components in a particular direction $\alpha = \tan\frac{y}{x}$, simply discard any coefficient 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.

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 coefficients 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.

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

Texture detection: textures may be characterized by constant ratios between wavelet packet coefficients 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 a coefficient at k to its neighbor at $k+1$ would have 0's in its range at k, indicating where that texture was located.

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.

The survey article [Wic92] describes these and other operations and their fast implementations in greater detail.

7 Source programs

Below is a printout of three C programs which perform block discrete LCT on a 256x256 pixel grey scale image. The first is a header file "fold.h" which contains most of the subroutines. A standard library package "interface.o" (not listed here) contains standard procedures for reading file names from the command line and so on. Its procedures are declared in "interface.h", but only `open_file()` and `open_different_file()` are used in these programs, and their purpose is self-evident:

```
/*** fold.h Header file for 2-D LCT and iLCT  programs. ***/
#include "interface.h"
char *inputname, *outputname;
FILE *inputfile, *outputfile;

/*** Trivial functions to read and write single floats: */
void getv( float *value ) { fscanf(inputfile, "%f", value); }
void putv( float value )  { fprintf(outputfile, "%f\n", value); }

/*** Take a sequence X[] and put B[]-determined combinations into it: */
void fold_in_place(float *X, float *B, int bellReach) {
  int i, j;  float temp;
  for(i=0, j= -1; i<bellReach; i++, j--) {
    temp = B[i]*X[i] + B[j]*X[j];
    X[j] = B[i]*X[j] - B[j]*X[i];
    X[i] = temp;
  }
}

/*** Inverse of fold_in_place(): */
void unfold_in_place( float *X, float *B, int bellReach ) {
  int i, j;  float temp;
  for(i=0, j=-1; i<bellReach; i++, j--) {
    temp = B[i]*X[i] - B[j]*X[j];
    X[j] = B[i]*X[j] + B[j]*X[i];
    X[i] = temp;
  }
}
```

```
/*** Fill an array of arrays with cosines: */
float **make_cos_tab( int n ) {
  float **pcosine;  int i, j;  double freq, norm;
  pcosine = (float **) malloc(n*sizeof(float *));
  norm = sqrt(2.0/(double)n);     /* ...to get unit vector */
  for (i=0; i<n; i++)     {
    pcosine[i] = (float *)calloc(n, sizeof(float));
    freq = (1.570796326794897)*(double)(2*i + 1)/(double)n;
    for(j= 0; j<n; j++)
      pcosine[i][j] = (float)(norm*cos(freq*(0.5+(double)j)));
  }
  return(pcosine);
}

/*** Find DCT-IV in place by inner products: */
void dctiv( float *x, float *y, int n, float **pcosine ) {
  int k, j;  float sum;
  for(k=0; k<n; k++) {  /* Loop over the frequency parameter */
    sum = 0.0;
    for(j= 0; j<n; j++) sum += pcosine[k][j]*x[j];
    y[k] = sum;
  }
  for(k=0; k<n; k++) x[k] = y[k];
}

/*** Allocate and assign an array with the left edge of a bell */
float *make_bell( int bellReach, int zeroContact ) {
  double x;  int i, j;  float *bell;
  bell = (float *)calloc((unsigned)(2*bellReach),sizeof(float));
  bell = &bell[bellReach]; /* Fill bell[] with cutoff function */
  for(j= -bellReach; j<bellReach; j++) { /* iterate  sin() */
    x =    ((double)j+0.5)/(double)bellReach; /* real no. in [-1, 1] */
    for(i=0; i<zeroContact; i++) x = sin( HALFPI*x );
    bell[j] = (float)sin( (1.570796326794897)*(0.5 + 0.5*x) );
  }
  return(bell);
}
```

The program “fold.c” performs the transform to BDLCT coefficients, listing them in order in the output file specified on the command line or interactively:

```
/*** fold.c
     Perform a 2-D local cosine transform on an 256x256 picture divided into
square blocks of dimension 8x8 which overlap (orthogonally) by 4 samples.
```

```
Read samples, row by row, from specified 'inputfile' via the function getv(),
which is defined in fold.h. Output the coefficients column by column into
'outputfile'. */
#include "fold.h"

main( int argc, char **argv ) {
  float *row, *col[256], *bell, y[8], **pcosine;  int i, j;
  open_file("input","rb",&inputname,&inputfile,argc,argv);
  open_different_file(inputname, "output","wb",
                  &outputname,&outputfile,argc,argv);
  row = (float *)calloc((unsigned)(256+2*4), sizeof(float));
  row = &row[4];    /* Set array range to [-4, 256+4) */
  for(j=0; j<256; j++) {
    col[j] = (float *)calloc((unsigned)(256+2*4), sizeof(float));
    col[j] = &col[j][4]; /* Set array range to [-4, 256+4) */
  }
  bell = make_bell(4, 1); /* Make and store the bell[] */
  pcosine = make_cos_tab(8); /* Make the table of cosines */
  for(i=0; i<256; i++) {    /* Loop over rows */
    for(j=0; j<256; j++) getv(&row[j]);  /* Read a row */
    for(j= -4; j<0; j++) row[j] = row[256+j];    /* Periodize */
    for(j= 0; j< 4; j++) row[256+j] = row[j];
    for(j=0; j<=256; j+= 8) fold_in_place(&row[j], bell, 4);
    for(j=0; j<256; j+= 8) dctiv(&row[j], y, 8, pcosine);
    for(j=0; j<256; j++) col[j][i] = row[j]; /* Transpose */
  }
  for(j=0; j<256; j++) {    /* Loop over columns */
    for(i= -4; i<0; i++) col[j][i] = col[j][256+i]; /* Periodize */
    for(i= 0; i< 4; i++) col[j][256+i] = col[j][i];
    for(i=0; i<=256; i+= 8) fold_in_place(&col[j][i], bell, 4);
    for(i=0; i<256; i+= 8) dctiv(&col[j][i], y, 8, pcosine);
  }
  for(j=0; j<256; j++) for(i=0; i<256; i++)  putv(col[j][i]);
  exit(0);
}
```

The program "unfold.c" inverts the BDLCT and dumps the reconstructed picture into a specified output file:

```
/*** unfold.c
    Invert the 2-D local cosine transform on the encoding of a picture  of size
256x256 pixels divided into square blocks of dimension 8x8 which overlap
(orthogonally) by 4 samples.  Assume that the coefficients are presented
column-by-column in 'inputfile'. Output the recontructed samples row-by-row
to 'outputfile'. */
```

```
#include "fold.h"

main(int argc, char **argv) {
  float *row[256], *col, *bell, y[8], **pcosine;  int i, j;
  open_file("input","rb",&inputname,&inputfile,argc,argv);
  open_different_file(inputname, "output","wb",
                    &outputname,&outputfile,argc,argv);
  col = (float *)calloc((unsigned)(256+2*4), sizeof(float));
  col = &col[4];    /* Set array range to [-4, 256+4) */
  for(j=0; j<256; j++) {
    row[j] = (float *)calloc((unsigned)(256+2*4), sizeof(float));
    row[j] = &row[j][4]; /* Set array range to [-4, 256+4) */
  }
  bell = make_bell(4, 1); /* Make and store the bell[] */
  pcosine = make_cos_tab(8); /* Make the table of cosines */
  for(i=0; i<256; i++) {    /* Loop over colums */
    for(j=0; j<256; j++) getv(&col[j]); /* Read column */
    for(j=0; j<256; j+= 8) dctiv(&col[j], y, 8, pcosine);
    for(j= −4; j<0; j++) col[j] = col[256+j];    /* Periodize */
    for(j= 0; j< 4; j++) col[256+j] = col[j];
    for(j=0; j≤256; j+= 8) unfold_in_place(&col[j], bell, 4);
    for(j=0; j<256; j++) row[j][i] = col[j]; /* Transpose*/
  }
  for(j=0; j<256; j++) {    /* Loop over rows */
    for(i=0; i<256; i+= 8) dctiv(&row[j][i], y, 8, pcosine);
    for(i= −4; i<0; i++) row[j][i] = row[j][256+i]; /* Periodize */
    for(i= 0; i< 4; i++) row[j][256+i] = row[j][i];
    for(i=0; i≤256; i+= 8) unfold_in_place(&row[j][i], bell, 4);
  }
  for(j=0; j<256; j++) for(i=0; i<256; i++) putv(row[j][i]);
  exit(0);
}
```

References

[BBH93] Jonathan N. Bradley, Christopher M. Brislawn, and Thomas E. Hopper. The FBI wavelet/scalar quantization standard for gray-scale fingerprint image compression. In *Visual Information Processing II*, volume 1961, Orlando, Florida, April 1993. SPIE.

[CD92] Albert Cohen and Ingrid Daubechies. Non-separable bidimensional wavelet bases. *Revista Matemática Iberoamericana*, 1992. To appear.

[CMQW90] Ronald R. Coifman, Yves Meyer, Stephen R. Quake, and Mladen Victor Wickerhauser. Signal processing and compression with wavelet packets. In Yves Meyer and Sylvie Roques, editors, *Progress in Wavelet Analysis and Applications*, Procedings of the International Conference “Wavelets and Applications”, pages 77–93. Editions Frontieres, Toulouse, France, 8–13 June 1990.

[CMW92a] Ronald R. Coifman, Yves Meyer, and Mladen Victor Wickerhauser. Size properties of wavelet packets. In Ruskai et al. [R$^+$92], pages 453–470.

[CMW92b] Ronald R. Coifman, Yves Meyer, and Mladen Victor Wickerhauser. Wavelet analysis and signal processing. In Ruskai et al. [R$^+$92], pages 153–178.

[CW92] Ronald R. Coifman and Mladen Victor Wickerhauser. Entropy based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 32:712–718, March 1992.

[CW93] Ronald R. Coifman and Mladen Victor Wickerhauser. Wavelets and adapted waveform analysis: A toolkit for signal processing and numerical analysis. In Daubechies [Dau93], pages 119–153. Minicourse lecture notes.

[Dau88] Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, XLI:909–996, 1988.

[Dau93] Ingrid Daubechies, editor. *Different Perspectives on Wavelets*, number 47 in Proceedings of Symposia in Applied Mathematics, San Antonio, Texas, 11-12 January 1993. American Mathematical Society.

[Don93] David L. Donoho. Unconditional bases are optimal bases for data compression and for statistical estimation. *Applied and Computational Harmonic Analysis*, 1(1):100–115, December 1993.

[DrJL92] Ron Devore, Bjørn Jawerth, and Bradley J. Lucier. Image compression through wavelet transform coding. *IEEE Transactions on Information Theory*, 38:719–746, March 1992.

[MBA90] P. Mathieu, M. Barlaud, and M. Antonini. Compression d’images par transformée en ondelette et quantification vectorielle. *Traitment du Signal*, 7(2):101–115, 1990.

[MF91] Stéphane G. Mallat and Jacques Froment. Compression d’images et ondelettes: une double approche contours et textures. In

Pierre-Louis Lyons, editor, *Problémes Non-Linéaires Appliqués, Ondelettes et Paquets D'Ondes*, pages 227–247. INRIA, Roquencourt, France, 17–21June 1991. Minicourse lecture notes.

[pasnt] pascal. Yale University Department of Mathematics. InterNet Anonymous File Transfer (ftp) Site pascal.math.yale.edu [128.36.23.1], 1989–present.

[R$^+$92] Mary Beth Ruskai et al., editors. *Wavelets and Their Applications.* Jones and Bartlett, Boston, 1992.

[RV91] Kannan Ramchandran and Martin Vetterli. Efficient quantization for adaptive wave-packet tree structures. Preprint, Columbia University, 1 June 1991.

[SW64] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication.* The University of Illinois Press, Urbana, 1964.

[Wic92] Mladen Victor Wickerhauser. High-resolution still picture compression. *Digital Signal Processing: a Review Journal*, 2(4):204–226, October 1992.

[Wic93] Mladen Victor Wickerhauser. Best-adapted wavelet packet bases. In Daubechies [Dau93], pages 155–171. Minicourse lecture notes.

[Wic94] Mladen Victor Wickerhauser. *Adapted Wavelet Analysis from Theory to Software.* AK Peters, Ltd., Wellesley, Massachusetts, 9 April 1994. 486+xii pages. To appear.

[ZSW91] Lareef Zubair, Kannan R. Sreenivasan, and Mladen Victor Wickerhauser. Compression of turbulence data and images using wavelet packets. In T. Gadsky, S. Sirkar, and C. Speziale, editors, *Studies in Turbulence.* Springer Verlag, New York, 1991.

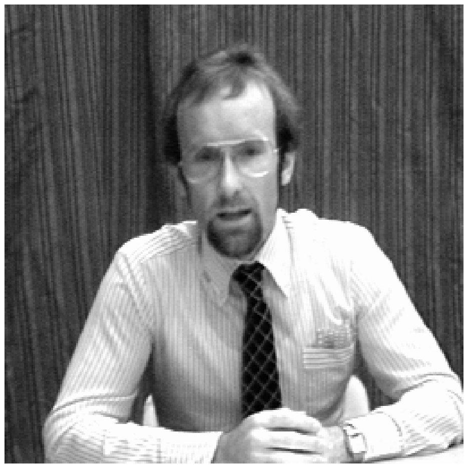
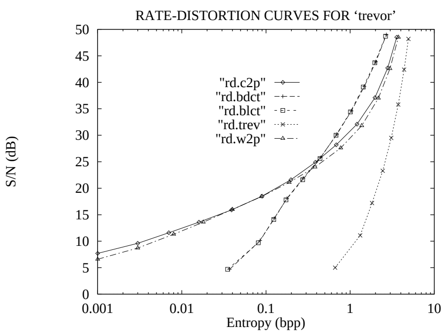

Figure 7: Original picture and rate-distortion curves for 5 compression methods

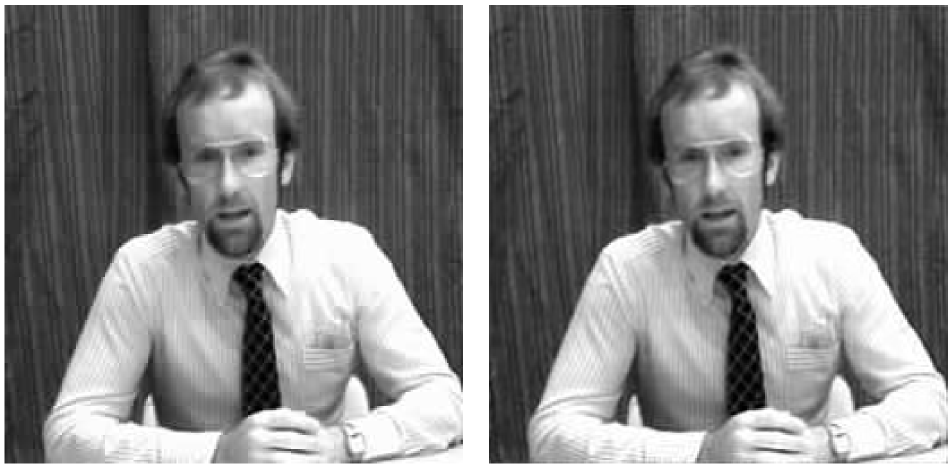

Figure 8: DCT versus LCT, 8 bits per coefficient.

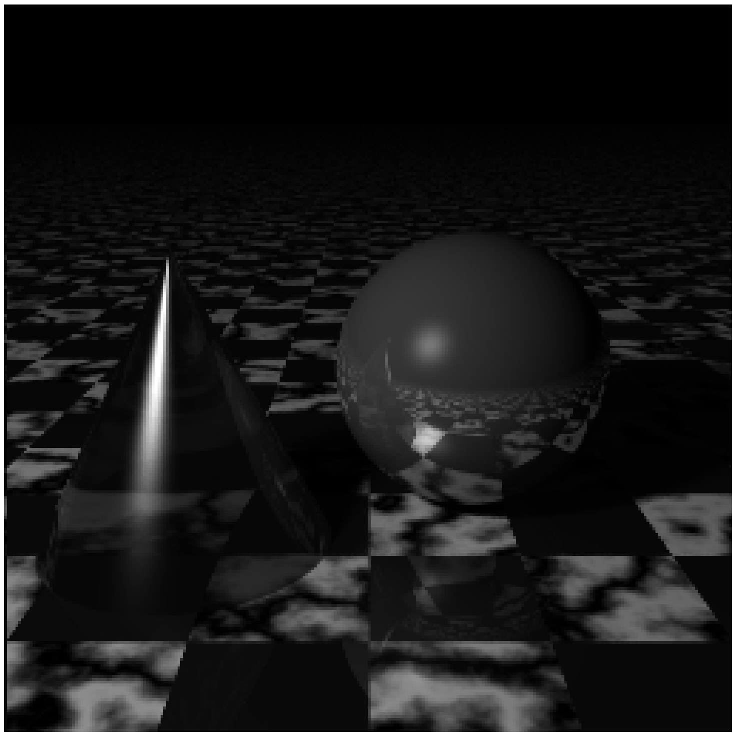

Figure 9: Original ray-traced picture.

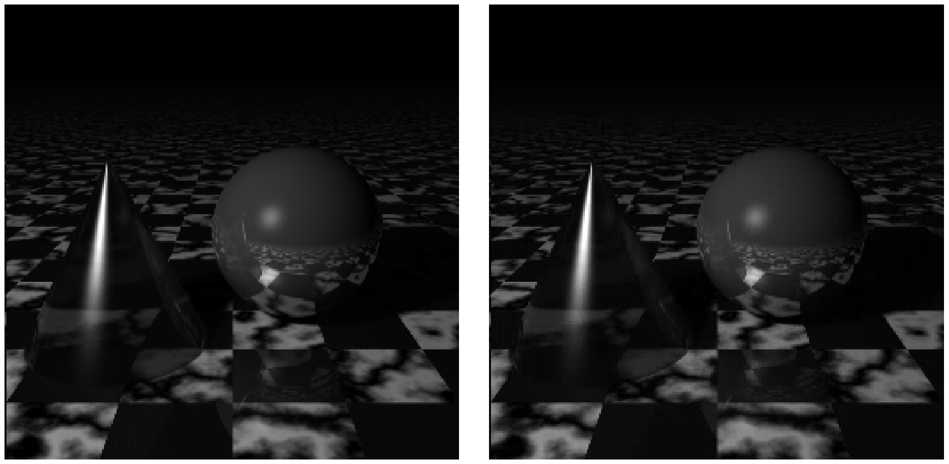

Figure 10: D 2 filter, compression ratios 2 and 5.

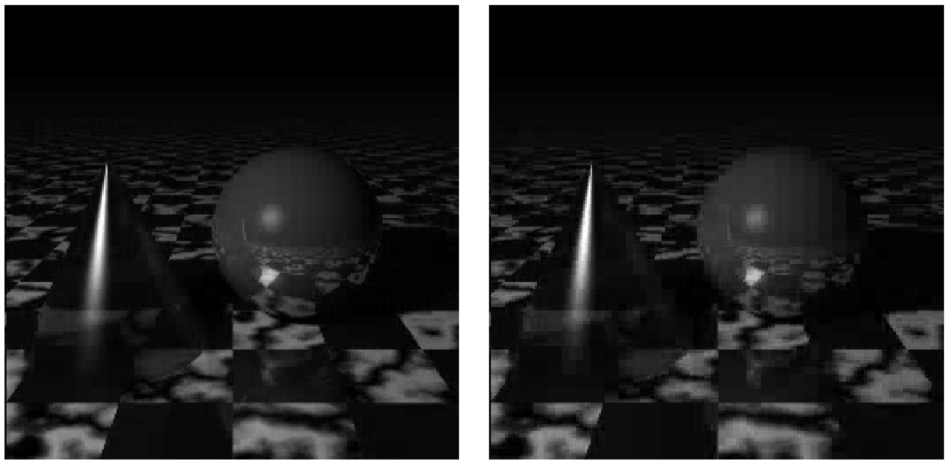

Figure 11: D 2 filter, compression ratios 10 and 20.

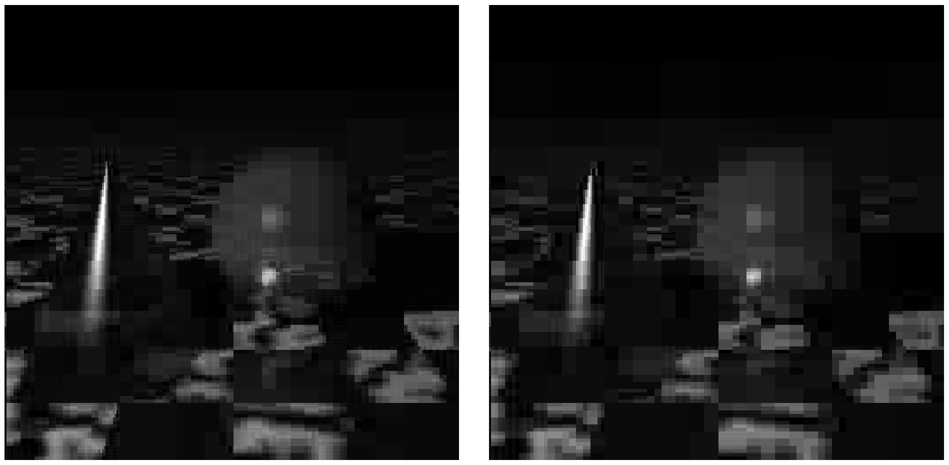

Figure 12: D 2 filter, compression ratios 50 and 100.

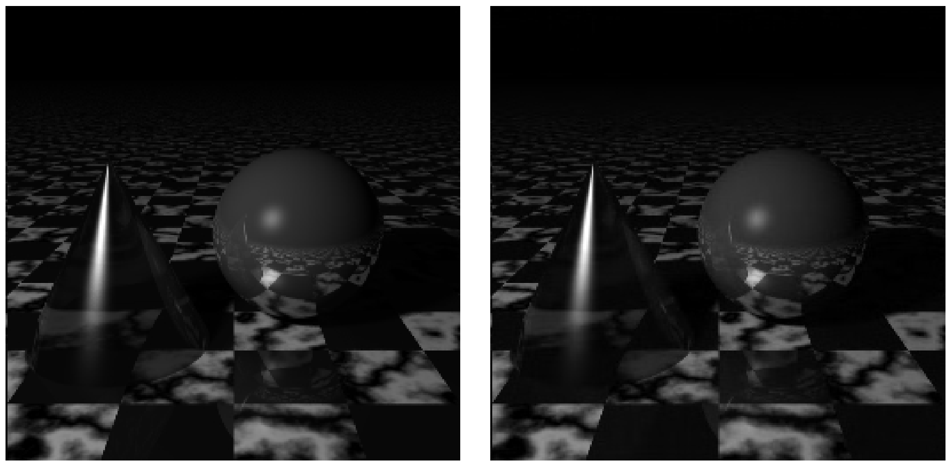

Figure 13: D 4 filter, compression ratios 2 and 5.

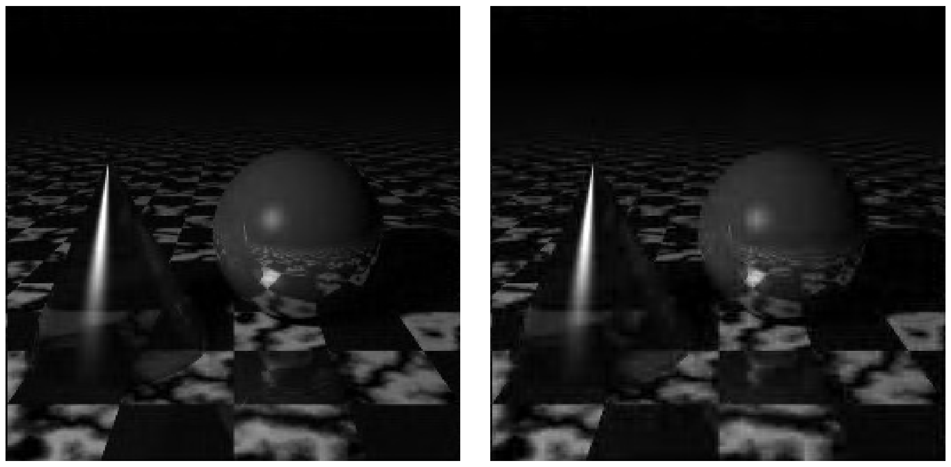

Figure 14: D 4 filter, compression ratios 10 and 20.

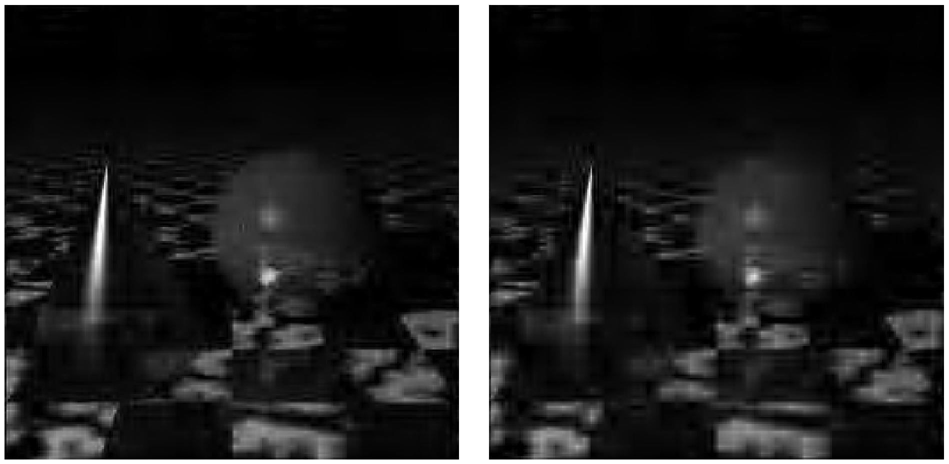

Figure 15: D 4 filter, compression ratios 50 and 100.

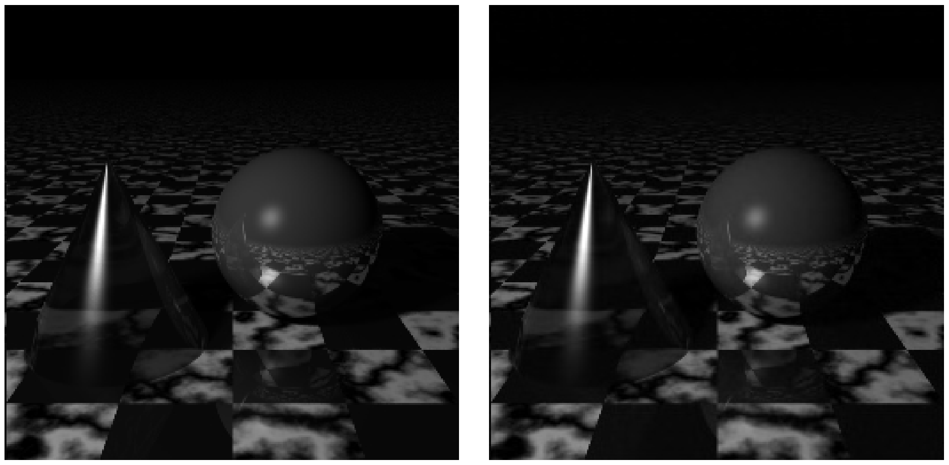

Figure 16: D 6 filter, compression ratios 2 and 5.

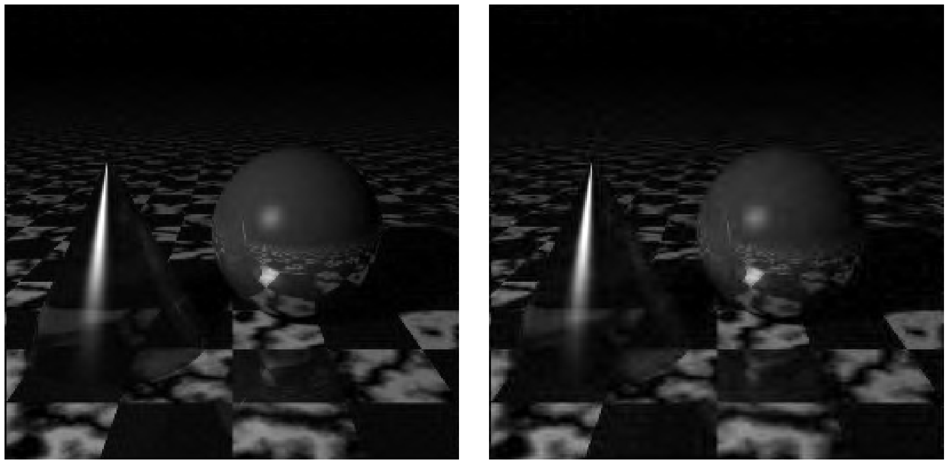

Figure 17: D 6 filter, compression ratios 10 and 20.

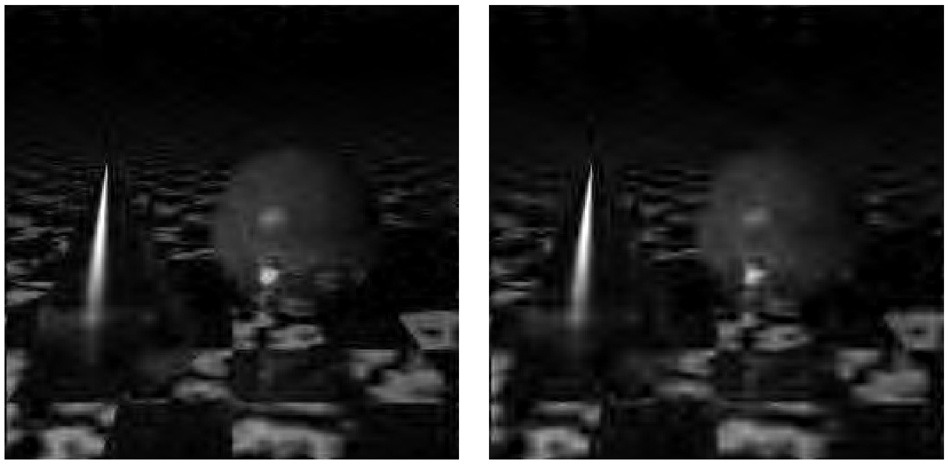

Figure 18: D 6 filter, compression ratios 50 and 100.

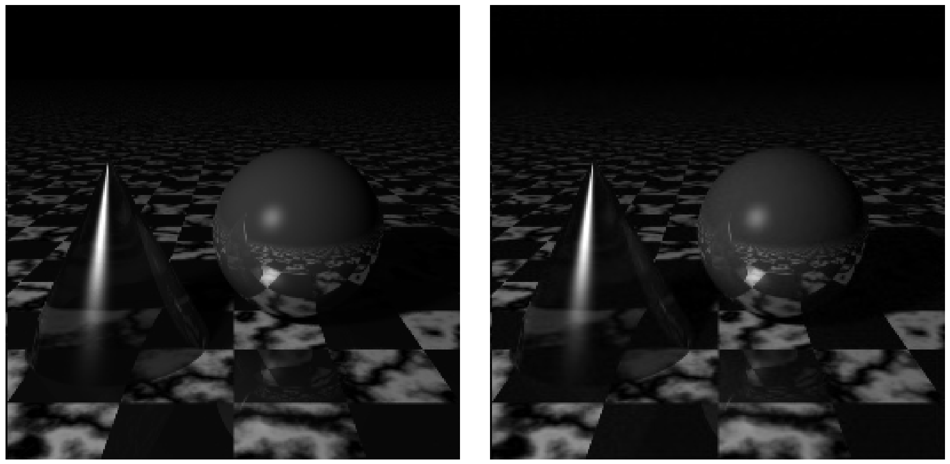

Figure 19: D 10 filter, compression ratios 2 and 5.

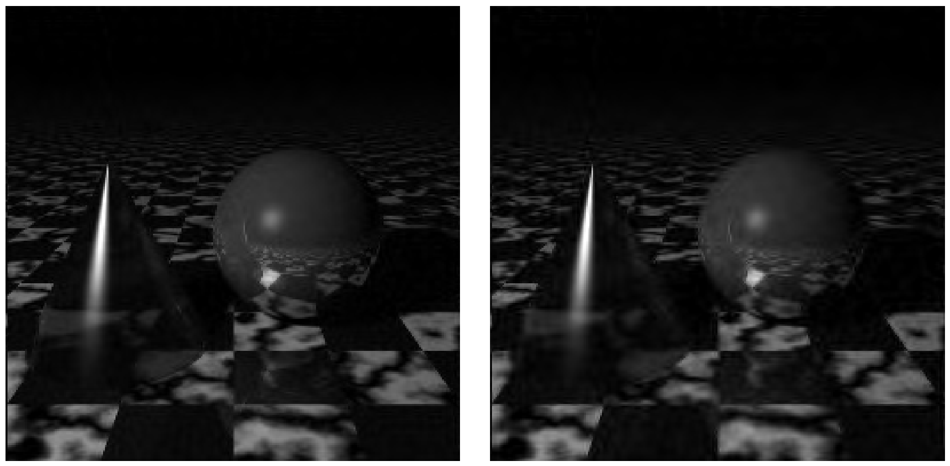

Figure 20: D 10 filter, compression ratios 10 and 20.

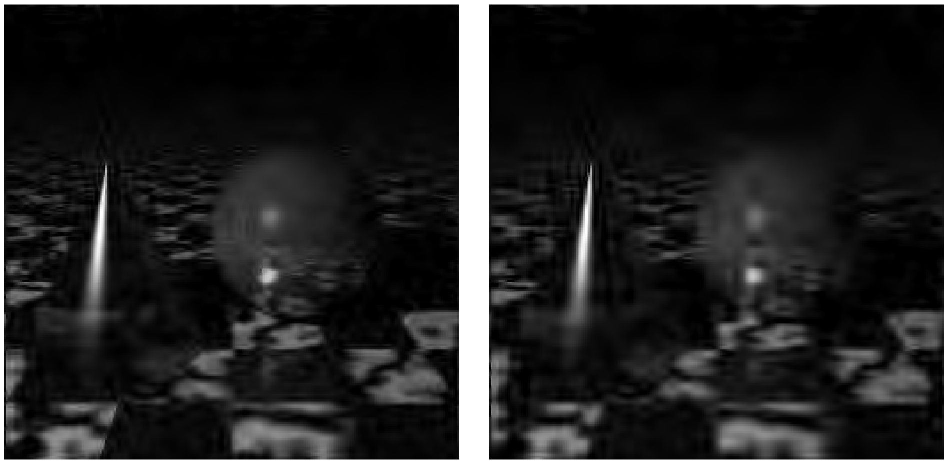

Figure 21: D 10 filter, compression ratios 50 and 100.

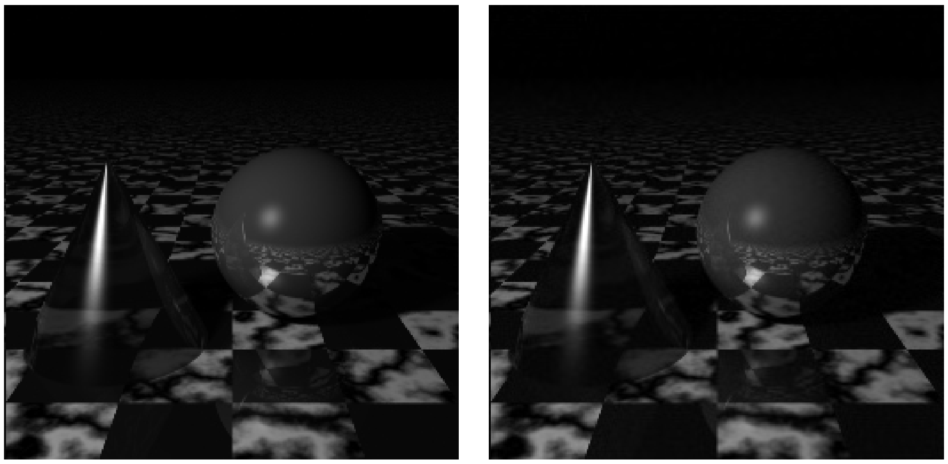

Figure 22: D 20 filter, compression ratios 2 and 5.

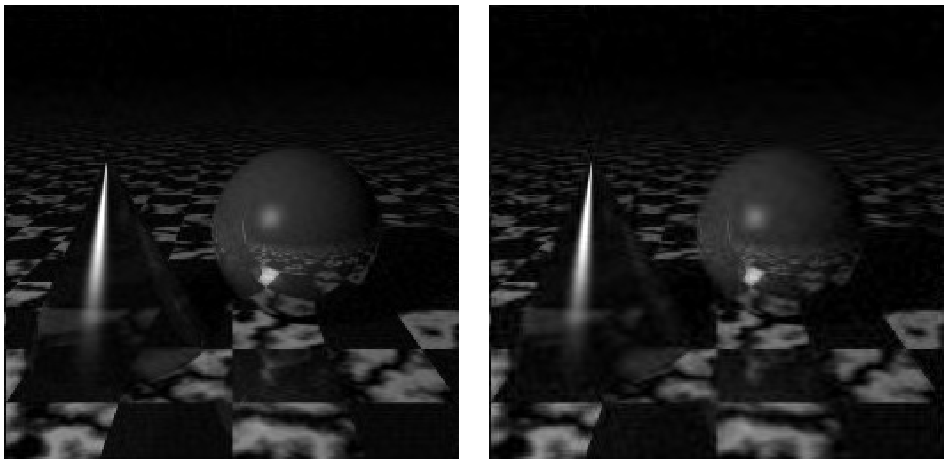

Figure 23: D 20 filter, compression ratios 10 and 20.

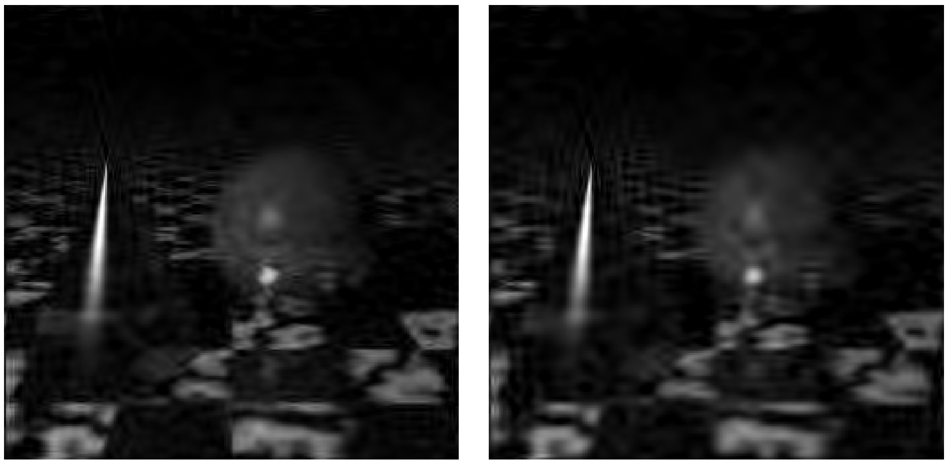

Figure 24: D 20 filter, compression ratios 50 and 100.