DIGITAL SIGNAL PROCESSING 2, 204–226 (1992)

High-Resolution Still Picture Compression*

Mladen Victor Wickerhauser

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

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 is 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 [4], 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 superefficient 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 two-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 at high and 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 three 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.

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

* Research supported in part by AFOSR Grant F49620-92-J-0106 and by FBI Contract A107183.

1051-2004/92 $4.00

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 represent only 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 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.

Figure 1 depicts a generic image compression transform coder. It embodies a three-step algorithm. 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 block (Quantize) 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 that 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 consists of some collection of pictures $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$, where M is a large number. The intensity of the nth pixel $S(n)$, $1 \leqslant n \leqslant N$, 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 $A = (A(i,j))_{i,j=1}^{N}$ of the pixels in $\mathcal{S}$,

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

where $\hat{S}_m = S_m - (1/M) \sum_m S_m$. A can be diagonalized because it is symmetric (see [2, Theorem 5.4, page 120] for a proof of this very general fact). We can write T for the orthogonal matrix that diagonalizes A;

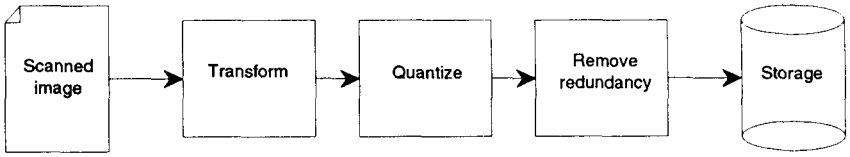

FIG. 1. Idealized transform coder.

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

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

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 DCT. In one dimension, this transform is an inner product with functions such as the one in the Fig. 2. 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 JPEG algorithm uses this transform and one other simplification. N is limited to 64 by taking 8×8 subblocks of the picture. JPEG applies two-dimensional DCT to the subblocks, and then treats the 64 vectors of amplitudes individually in a manner which we will discuss in the next section.

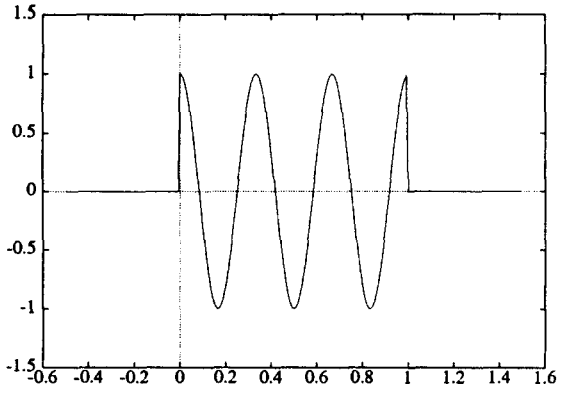

FIG. 2. Example DCT basis function.

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 [7]) or *lapped orthogonal transforms* (LOT, as in [23]) 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

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

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 $\mathbf{R}$. Note that we can modify b to obtain additional continuous derivatives by iterating the innermost $\sin \pi x$. Let $b_1(x) = b(x)$ and define

$$b_{n+1}(x) = b_n(\tfrac{1}{2}\sin \pi x). \tag{4}$$

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 $I = \{0, 1, 2, \ldots, N-1\}$, 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 $I' = \{\cdots, -2, -1\}$ and future samples $I'' = \{N, N+1, \cdots\}$ 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

$$\psi_k(j) = \frac{1}{\sqrt{2N}} b\left(\frac{j+\frac{1}{2}}{N}\right) \cos\left(\pi\left(k+\frac{1}{2}\right)\left[\frac{j+\frac{1}{2}}{N}\right]\right). \quad (5)$$

Apart from b, these are evidently the basis functions for the so-called DCT-IV transform. Figure 3 shows one such function, with N chosen large enough so that the smoothness is evident. The orthogonality of such functions may be checked by verifying the equations

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

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

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

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+}:

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

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

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:

$$f_{1-}(j) = \begin{cases} f(j), & \text{if } j < N/2, \\ b\left(\frac{-j-\frac{1}{2}}{N} - 1\right) f(j) - b\left(\frac{j+\frac{1}{2}}{N} - 1\right) & \\ \quad \times f(2N-j-1), & \text{if } j \in \{N/2, \ldots, N-1\}, \end{cases}$$

$$f_{1+}(j) = \begin{cases} b\left(\frac{j+\frac{1}{2}}{N} - 1\right) f(j) & \\ \quad + b\left(\frac{-j-\frac{1}{2}}{N} - 1\right) f(2N-j-1), & \\ & \text{if } j \in \{N, N+1, \ldots, 3N/2-1\}, \\ f(j), & \text{if } j \geq 3N/2 \end{cases} \quad (9)$$

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

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

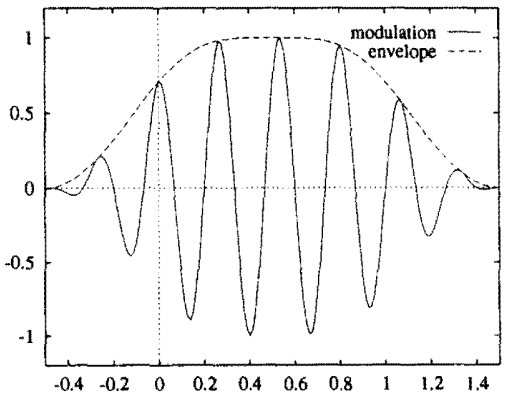

FIG. 3. Example LCT basis function.

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 formulas

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

Composing these relations yields $f(j) = [b(j+\frac{1}{2}/N)^2 + b(-j-\frac{1}{2}/N)^2] f(j)$. This equation is verified by the bell b defined above, 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. It will then 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 [13].

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

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

H and G are defined on square-summable signal sequences of any length. They are also 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

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

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

$$HG^* = GH^* = 0; \quad H^*H + G^*G = I. \quad (14)$$

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

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

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

$$\begin{pmatrix} m_0(\xi) & m_0(\xi+\pi) \\ m_1(\xi) & m_1(\xi+\pi) \end{pmatrix}. \quad (16)$$

This fact is proved in [11]. 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 in Fig. 4. We have plotted the absolute values of m_0 and m_1, respectively, over one period $[-\pi, \pi]$. Note that m_0 attenuates frequencies far from 0, while m_1 attenuates those near 0.

The traditional block diagram describing the action of a pair of quadrature mirror filters is shown in Fig. 5. 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.

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 $e_0 = \{\cdots, 0, 0, 1, 0, 0, \cdots\}$ produce 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

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.