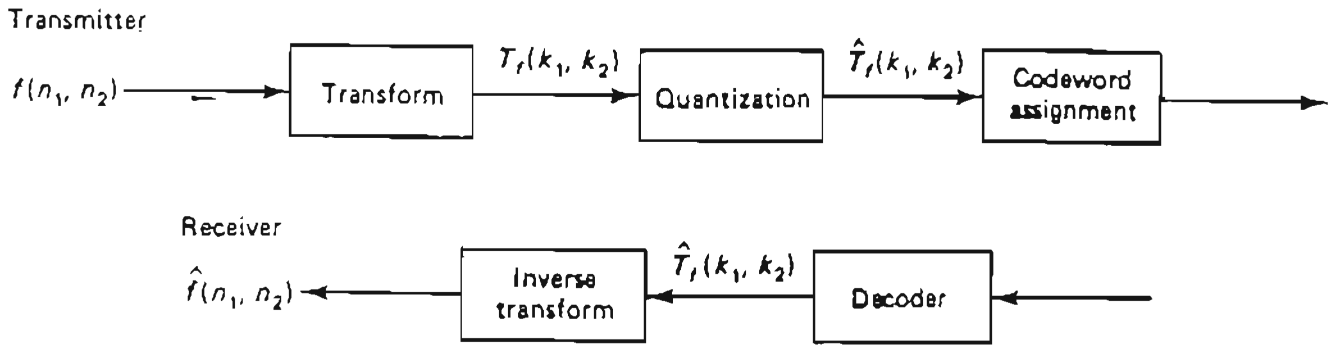
Transform Image Coding



What is exploited: Most of the image energy is concentrated in a small number of coefficients for some transforms

- the more energy compaction, the better

Some considerations:

- energy compaction in a small number of coefficients
- computational aspect: important (subimage by sub-image coding — 8×8 - 16×16)
- transform should be invertible
- Correlation Reduction

Examples of Transforms

1. Karhunen-Loeve Transform

$$F_k(k_1, k_2) = \sum_{n=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} f(n_1, n_2) \cdot A(n_1, n_2;\ k_1, k_2)$$

$$\lambda(k_1, k_2) \cdot A(n_1, n_2;\ k_1, k_2) =$$

$$\sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} K_f(n_1, n_2;\ l_1, l_2) \cdot A(l_1, l_2;\ k_1, k_2)$$

Covariance $K_f(n_1, n_2;\ l_1, l_2) =$

$$E\ [(x(n_1, n_2) - \overline{x}(n_1, n_2)) \cdot (x(l_1, l_2) - \overline{x}(l_1, l_2))]$$

Comments:

- optimal in the sense that the coefficients are completely uncorrelated
- finding $K_f(n_1, n_2;\ l_1, l_2)$ is hard
- no simple computational algorithm
- seldom used in practice
- On average, first M coefficients have more energy than any other transform
- KL is best among all linear transforms from: (a) compaction (b) decorrelation

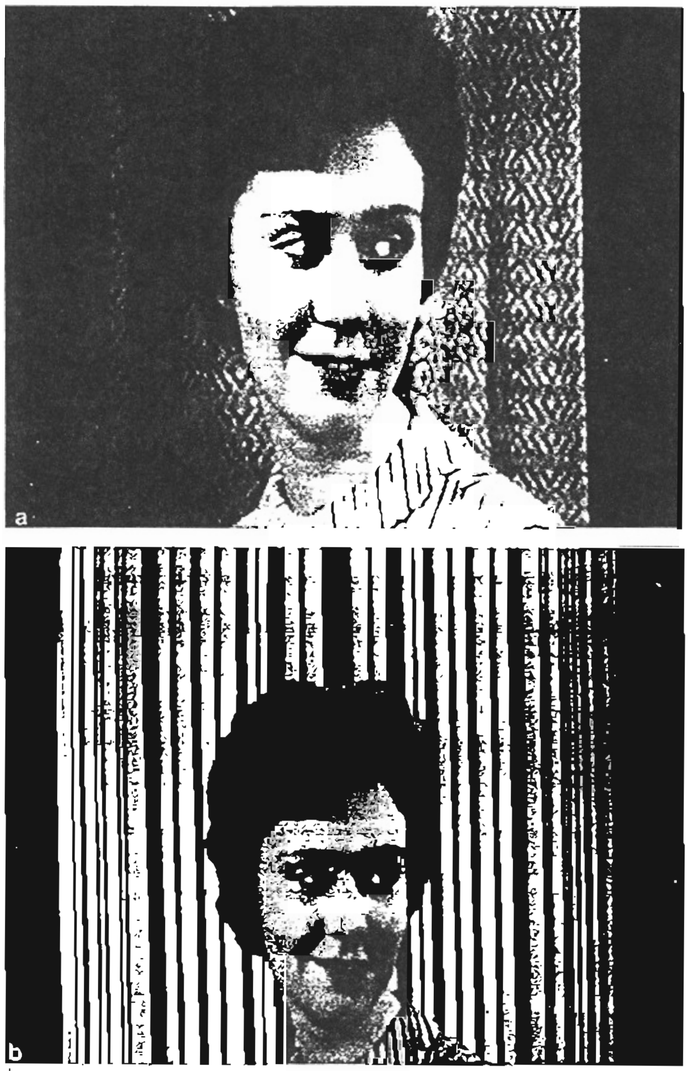


Fig. 5.3.7 Images used for coding and statistics. (a) "Karen" has much more stationary statistics than (b) "Stripes."

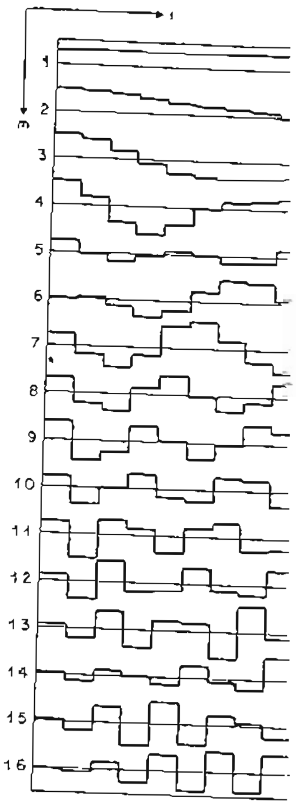


Fig. 5.3.8 KLT basis vectors for the image "
N=16 and μ = 0. For each m the c

orthonormalized eigenvectors of $\boldsymbol{R}$, i.e.,

$$\boldsymbol{R}\boldsymbol{t}_m = \lambda_m \boldsymbol{t}$$

$$\boldsymbol{t}_m'\boldsymbol{t}_n = \delta_{mn}$$

where the eigenvalues $\{\lambda_m\}$ are nonnegati

For example, Fig. 5.3.8 shows KL "Karen" in Fig. 5.3.7a using one-dimensi and μ = 0. The eigenvectors are arr eigenvalue. Note that, for the most part, according to increasing frequency, i.e., i basis vector. Fig. 5.3.9 shows similar c correlation model defined in Chapter

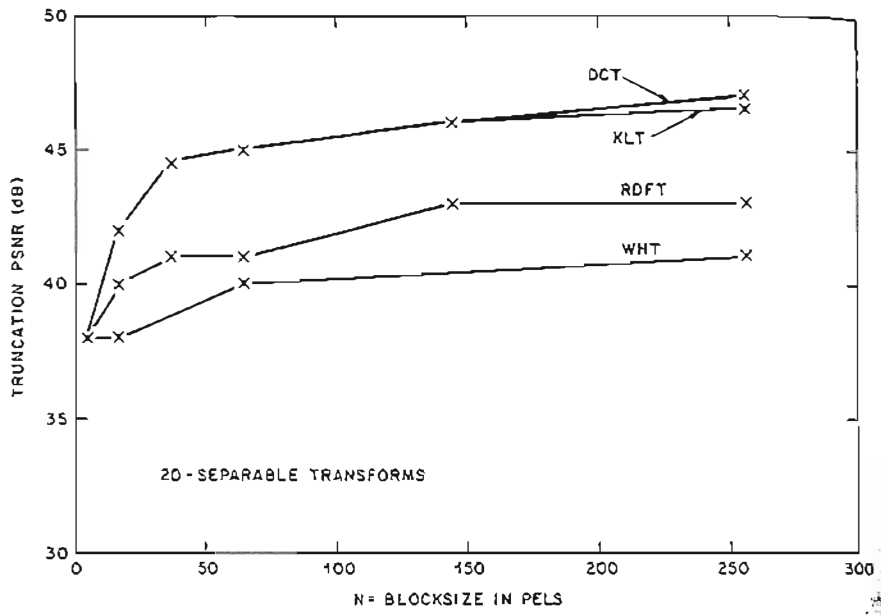


Fig. 5.3.22 Truncation PSNR versus block size for separable transforms with the image "Karen" when 60 percent of the coefficients are kept (p=0.6).

size, the higher the energy compaction achieved by the transform. Also two-dimensional blocks achieve more compaction than one-dimensional blocks. Experience has shown that over a wide range of pictures there is not much improvement in average energy compaction for two dimensional block sizes above 8×8 pels. However, individual pictures with higher nonstationary statistics can always be found for which this rule of thumb is violated (for example, compare the KLT curves of Fig. 5.3.17 and Fig.5.3.19). Also, considerable correlation may remain between blocks, even though the correlation between pels within a block is largely removed.[5.3.21] We shall return to this point in a later section.

5.3.1f Miscellaneous Transforms

Several other transforms have been studied. For example, the Haar Transform[5.3.8] can be computed from an orthogonal (but not orthonormal) matrix T that contains only +1's, −1's and zeros as shown in Fig. 5.3.23. This enables rather simple and speedy calculation, but at the expense of energy compaction performance.

Fig. 5.3.23 Haar transform matri
multiplications.

The Slant Transform[5.3.9]
vector t_1, a basis vector t_2 give

$$t_2 = \alpha(N-1,\ N-$$

where α is a normalization c
approximate the local behavior
compaction. However, the t_2
overall performance in most c
been developed for the Slant Tr
The Sine Transform[5.3.10]

$$t_{mi} = \sqrt{}$$

$$m,i = 1 \ldots$$

Its main utility arises when ima
sum of two uncorrelated images
statistics with a KLT that is app
The Singular Value Decor
the separable inverse transform

where U and V are unitary L t

constructed from a two-
: the L-pel rows end-to-end.
ot only high adjacent pel
ls with separation L. Thus
y large not only at low
L, etc. Fig. 5.3.19 shows
e transform coded in this
ing the largest MSV are
ian those of Fig. 5.3.17 by

els the most often used
using two L-dimensional
Recall that with separable
ıd transform the rows and

(5.3.56)

ansform coefficients. For
; when the separable DCT
lata from "Stripes". Note
iencies compared with the
ertical correlation in this

the results of separable
icture "Karen" when only
LT was derived from the
; of pels. We see that the
separable KLT. This is
ınot adapt very well to
all block size of $1\times L$ pels.
e DCT is only a few dB
9. It is this characteristic
ısform of choice in many

NR results for $p = 60\%$.
proves as the block size
ely small for block sizes

: important parameters
nts although the picture
ılly, the larger the block

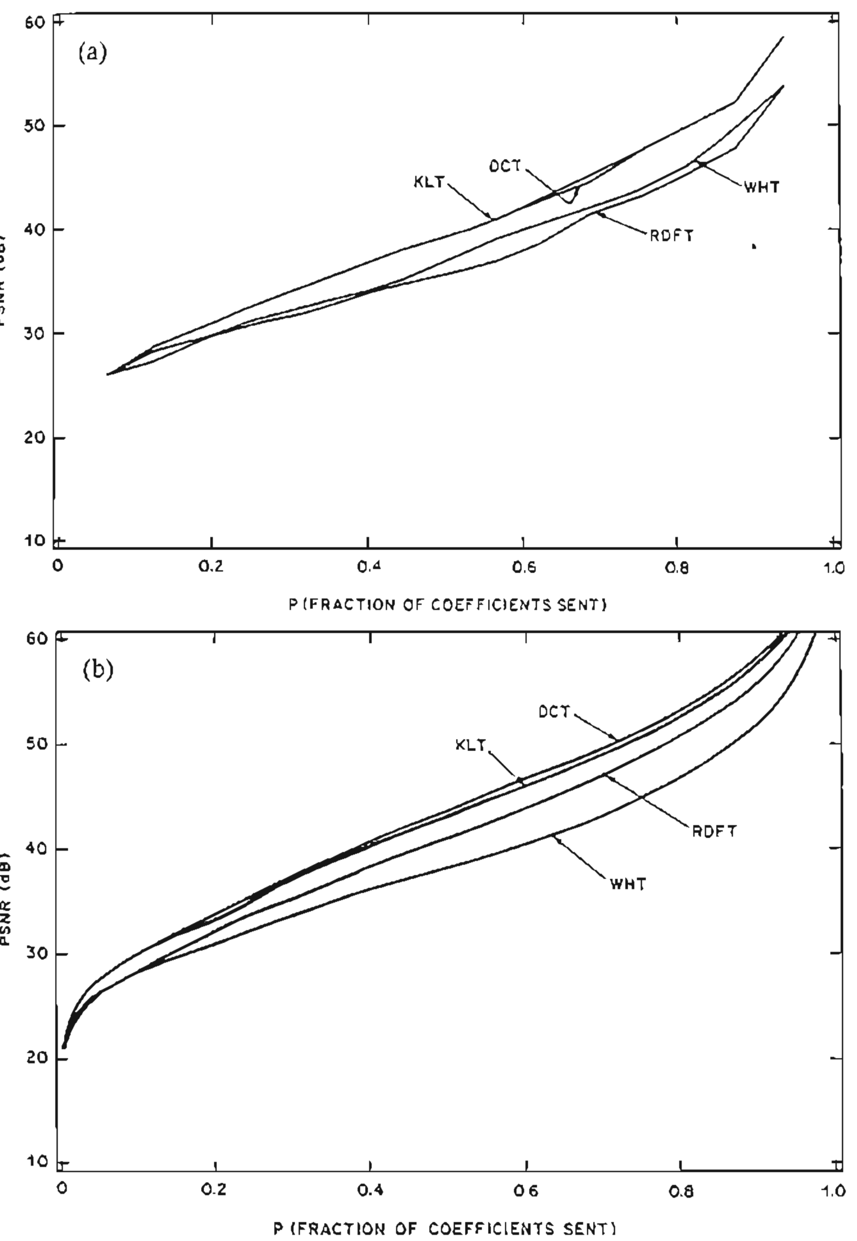


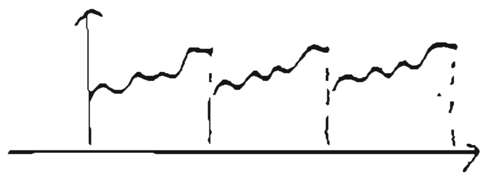
Fig. 5.3.21 Comparison of truncation errors using separable, two-dimensional blocks with the image "Karen". The coefficients having the largest MSV are transmitted. (a) 4×4 blocks, $N=16$. (b) 16×16 blocks, $N=256$.

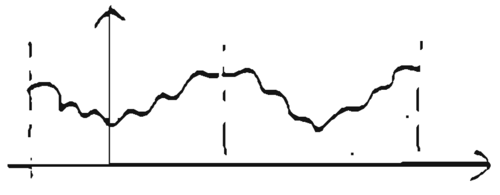
Discrete Cosine Transform

$$\begin{tikzcd} f(n_1,n_2) \arrow[r] & f'(n_1,n_2) = \dfrac{f(n_1,n_2)+f(n_1,-n_2)+f(-n_1,n_2)+f(-n_1,-n_2)}{4} \arrow[d] \\ & \boxed{\text{DFT}} \arrow[d] \\ & F_c(\omega_1,\omega_2) \end{tikzcd}$$

Comments:

- good energy compaction (better than DFT)


sharp discontinuity


no sharp discontinuity

- fast algorithms
- all real coefficients
- most often used in practice (good quality image at bit rate less than 1 bit/pixel)
- other transforms: Hadamard, Haar, Slant, Sine, ...

The sequence $Y(k)$ is related to $y(n)$ through the $2N$-point inverse DFT relation given by

$$y(n) = \frac{1}{2N} \sum_{k=0}^{2N-1} Y(k) W_{2N}^{-kn}, \quad 0 \le n \le 2N - 1. \tag{3.28}$$

From (3.20), $x(n)$ can be recovered from $y(n)$ by

$$x(n) = \begin{cases} y(n), & 0 \le n \le N - 1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.29}$$

From (3.27), (3.28), and (3.29), and after some algebra,

$$x(n) = \begin{cases} \dfrac{1}{N}\left[\dfrac{C_x(0)}{2} + \displaystyle\sum_{k=1}^{N-1} C_x(k) \cos \frac{\pi}{2N} k(2n + 1)\right], & 0 \le n \le N - 1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.30}$$

Equation (3.30) can also be expressed as

$$x(n) = \begin{cases} \dfrac{1}{N} \displaystyle\sum_{k=0}^{N-1} w(k) C_x(k) \cos \frac{\pi}{2N} k(2n + 1), & 0 \le n \le N - 1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.31a}$$

where

$$w(k) = \begin{cases} \dfrac{1}{2}, & k = 0 \\ 1, & 1 \le k \le N - 1. \end{cases} \tag{3.31b}$$

Equation (3.31) is the inverse DCT relation. From (3.25) and (3.31),

Discrete Cosine Transform Pair

$$C_x(k) = \begin{cases} \displaystyle\sum_{n=0}^{N-1} 2x(n) \cos \frac{\pi}{2N} k(2n + 1), & 0 \le k \le N - 1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.32a}$$

$$x(n) = \begin{cases} \dfrac{1}{N} \displaystyle\sum_{k=0}^{N-1} w(k) C_x(k) \cos \frac{\pi}{2N} k(2n + 1), & 0 \le n \le N - 1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.32b}$$

From the derivation of the DCT pair, the DCT and inverse DCT can be computed by

Computation of Discrete Cosine Transform

Step 1. $y(n) = x(n) + x(2N - 1 - n)$

Step 2. $Y(k) = \text{DFT}\,[y(n)]$ ($2N$-point DFT computation)

Step 3. $C_x(k) = \begin{cases} W_{2N}^{k/2} Y(k), & 0 \le k \le N - 1 \\ 0, & \text{otherwise} \end{cases}$

Step 1. $Y(k) = \{$

Step 2. $y(n) =$ II

Step 3. $x(n) = \{$

In computing the
simple. Most of th
for the DCT and a
DFT and inverse
algorithms. In ad
DFT can be comp
N-point inverse D
volved in using the
In the deriva
$y(n)$ that has sym
called an even sym
DCT pair in the s
sequence $y(n)$ use
shown in Figure 3
length of $y(n)$ is
points, has no arti
metrical DCT is co
commonly used, si
DFT, which is not

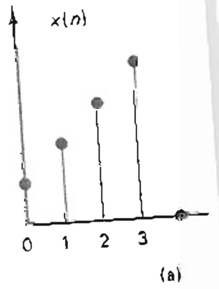


Figure 3.9
$x(N - 1)\delta(n$
defining the

Computation of Inverse Discrete Cosine Transform

Step 1. $$Y(k) = \begin{cases} W_{2N}^{-k/2}C_x(k), & 0 \le k \le N-1 \\ 0, & k = N \\ -W_{2N}^{-k/2}C_x(2N-k), & N+1 \le k \le 2N-1 \end{cases}$$

Step 2. $y(n) = \text{IDFT}\,[Y(k)]$ ($2N$-point inverse DFT computation)

Step 3. $$x(n) = \begin{cases} y(n), & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases}$$

In computing the DCT and inverse DCT, Steps 1 and 3 are computationally quite simple. Most of the computations are in Step 2, where a $2N$-point DFT is computed for the DCT and a $2N$-point inverse DFT is computed for the inverse DCT. The DFT and inverse DFT can be computed by using fast Fourier transform (FFT) algorithms. In addition, because $y(n)$ has symmetry, the $2N$-point DFT and inverse DFT can be computed (see Problem 3.20) by computing the N-point DFT and the N-point inverse DFT of an N-point sequence. Therefore, the computation involved in using the DCT is essentially the same as that involved in using the DFT.

In the derivation of the DCT pair, we have used an intermediate sequence $y(n)$ that has symmetry and whose length is even. The DCT we derived is thus called an even symmetrical DCT. It is also possible to derive the odd symmetrical DCT pair in the same manner. In the odd symmetrical DCT, the intermediate sequence $y(n)$ used has symmetry, but its length is odd. For the sequence $x(n)$ shown in Figure 3.9(a), the sequence $y(n)$ used is shown in Figure 3.9(b). The length of $y(n)$ is $2N - 1$, and $\tilde{y}(n)$, obtained by repeating $y(n)$ every $2N - 1$ points, has no artificial discontinuities. The detailed derivation of the odd symmetrical DCT is considered in Problem 3.22. The even symmetrical DCT is more commonly used, since the odd symmetrical DCT involves computing an odd-length DFT, which is not very convenient when one is using FFT algorithms.

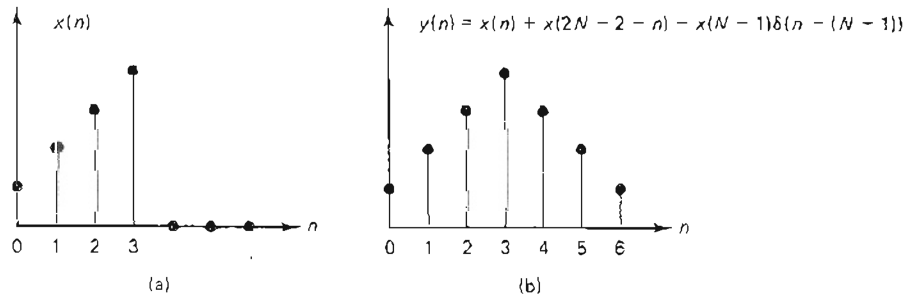


Figure 3.9 Example of (a) $x(n)$ and (b) $y(n) = x(n) + x(2N-2-n) - x(N-1)\delta(n-(N-1))$. The sequence $y(n)$ is used in the intermediate step in defining the odd symmetrical discrete cosine transform of $x(n)$.

DCT

- Signal independent

- ρ ---> 1 : KLT ---> DCT

  for first order
  Markov Image model

- Type II DCT:

$$S(K_1, K_2) = \sqrt{\frac{4}{N^2}} C(K_1) C(K_2)$$

$$\sum_{n_1 = o}^{N-1} \sum_{n_2 = o}^{N-1} s(n_1, n_2) \cos\left(\frac{\pi 2(n_1 + 1)K_1}{2N}\right)$$

$$\cos\left(\frac{\pi 2(n_2 + 1)K_2}{2N}\right)$$

$$C(K) = \begin{cases} \frac{1}{\sqrt{2}} & K = o \\ 1 & \text{otherwise} \end{cases}$$

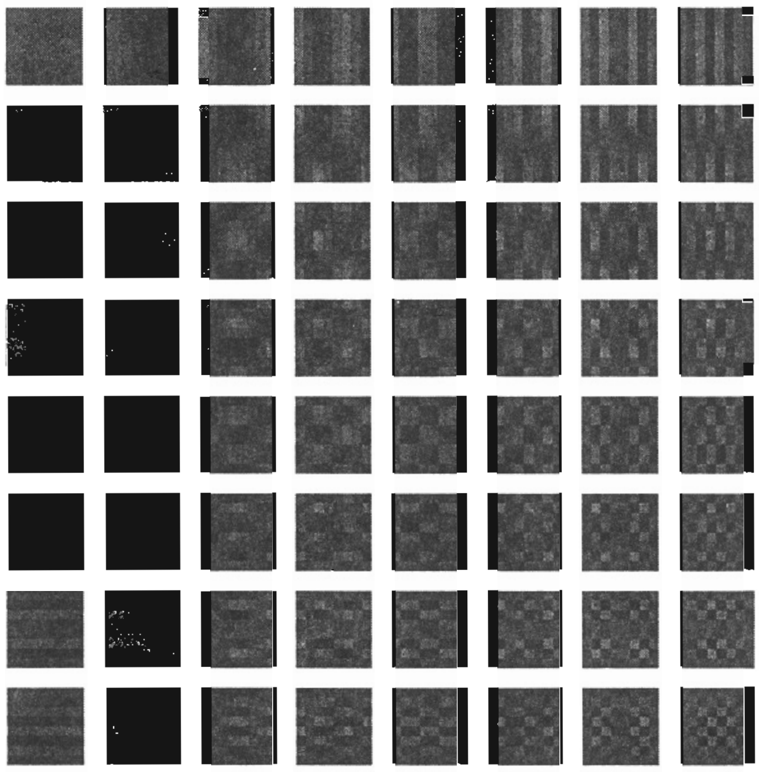


FIGURE 12.4 The basis matrices for the DCT.

The outer products of the rows are shown in Figure 12.4. Notice that the basis matrices show increased variation as we go from the top left matrix, corresponding to the θ_{00} coefficient, to the bottom right matrix, corresponding to the $\theta_{(N-1)(N-1)}$ coefficient.

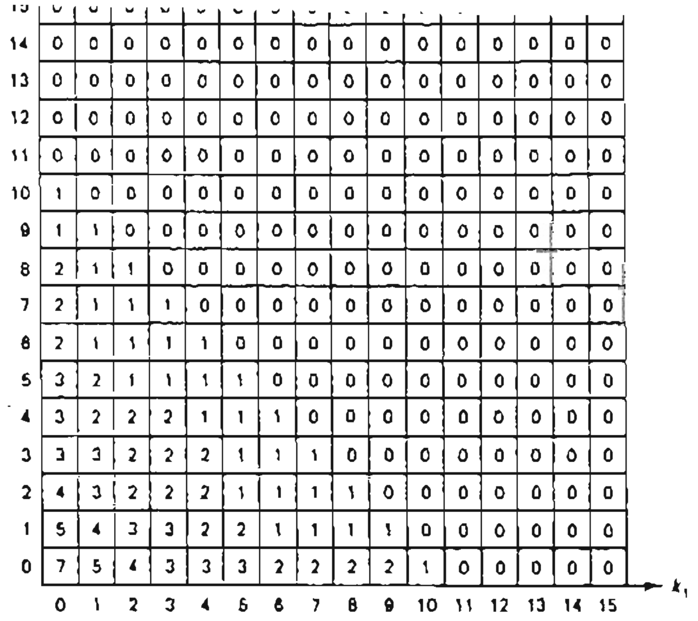
The DCT is closely related to the discrete Fourier transform (DFT) mentioned in Chapter 11, and in fact can be obtained from the DFT. However, in terms of compression, the DCT performs better than the DFT.

Recall that when we find the Fourier coefficients for a sequence of length N, we assume that the sequence is periodic with period N. If the original sequence is as shown in Figure 12.5a, the DFT assumes that the sequence outside the interval of interest behaves in the manner shown in Figure 12.5b. This introduces sharp discontinuities, at the beginning and end of the sequence. In order to represent these sharp discontinuities the DFT needs nonzero coefficients for the high-frequency components. As these components are needed only at the two endpoints of the sequence, their effect needs to be cancelled out at other points in the sequence. Thus, the DFT adjusts other coefficients accordingly. When we discard the high-frequency coefficients (which should not have been there anyway) during the compression process, the coefficients that were cancelling out the high-frequency effect in other parts of the sequence result in the introduction of additional distortion.

Discarding Transform Coefficients (cont.)

Threshold coding: Coefficients with values above a given threshold are coded

- location as well as amplitude has to be coded
- run-length coding is useful (many zeroes)



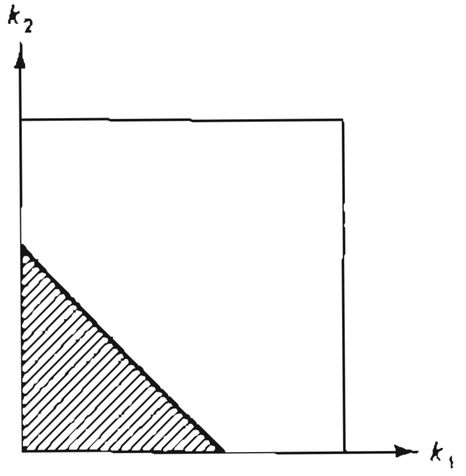
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
7	2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
6	2	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
5	3	2	1	1	1	1	0	0	0	0	0	0	0	0	0	0
4	3	2	2	2	1	1	1	0	0	0	0	0	0	0	0	0
3	3	3	2	2	2	1	1	1	0	0	0	0	0	0	0	0
2	4	3	2	2	2	1	1	1	1	0	0	0	0	0	0	0
1	5	4	3	3	2	2	1	1	1	1	0	0	0	0	0	0
0	7	5	4	3	3	3	2	2	2	2	1	0	0	0	0	0
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15 k_1

Figure 10.44 Example of bit allocation map at ½ bit/pixel for zonal discrete cosine transform image coding. Block size = 16 × 16 pixels.

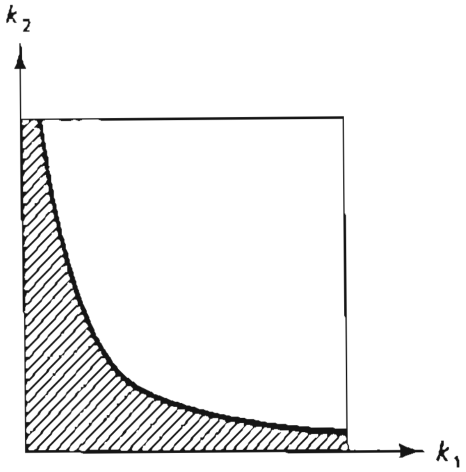
Discarding Transform Coefficients

Zonal coding: Eliminate coefficients in a fixed zone

(Ex)



(a)



(b)

Bits for coefficient i with variance σ_i^2

$$b_i = \frac{B}{M} + \frac{1}{2}\log_2 \sigma_i^2 - \frac{1}{2M}\sum_{i=1}^{M} \log_2 \sigma_i^2$$

M = # of coefficients to be coded

B = Total # of bits

Scalar Quantization of a Vector Source

- Assume N scalars: $f_i \qquad 1 \leq i \leq N$
- Each f_i is quantized to L_i reconstruction levels.
- Total of B bits to code N scalars.
- Optimum bit allocation strategy depends on (a) error criterion and (b) pdf of each random variable.
- Assume we minimize MSE : $\Sigma_{i=1}^{N} E[(f\prime_i - f_i)^2]$ with respect to B_i the number of bits for the *ith* scalar for $1 \leq i \leq N$.
- Assume pdf of all f_i is the same except they have different variances.
- Use Lloyd Max quantizer.
- Under these conditions we have:

$$B_i = \frac{B}{N} + \frac{1}{2} log \frac{\sigma_i^2}{[\Pi_{j=1}^{N} \sigma_j^2]^{1/N}}$$

- σ_i^2 is the variance of f_i

$$L_i = \frac{\sigma_i}{[\Pi_{j=1}^{N} \sigma_i]^{1/N}} 2^{B/N}$$

- L_i is the number of reconstruction levels for source i

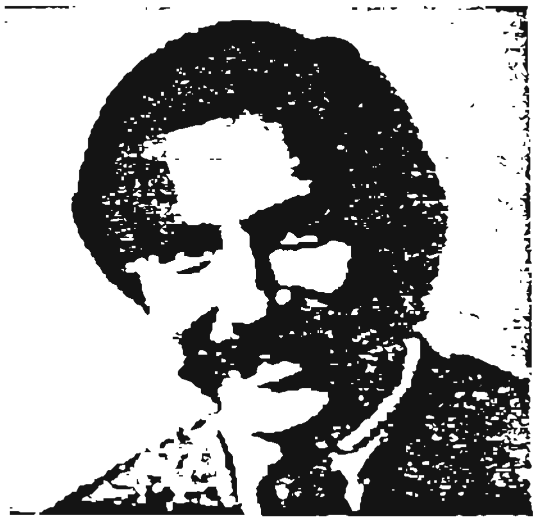


Figure 10.47 DCT-coded image with visible blocking effect

and (b) show the results of DCT image coding at 1 bit/pixel and $\frac{1}{2}$ bit/pixel, respectively. The original image is the 512 × 512-pixel image shown in Figure 10.45(a). In both examples, the subimage size used is 16 × 16 pixels, and adaptive zonal coding with the zone shape shown in Figure 10.43(b) and the zone size adapted to the local image characteristics has been used.

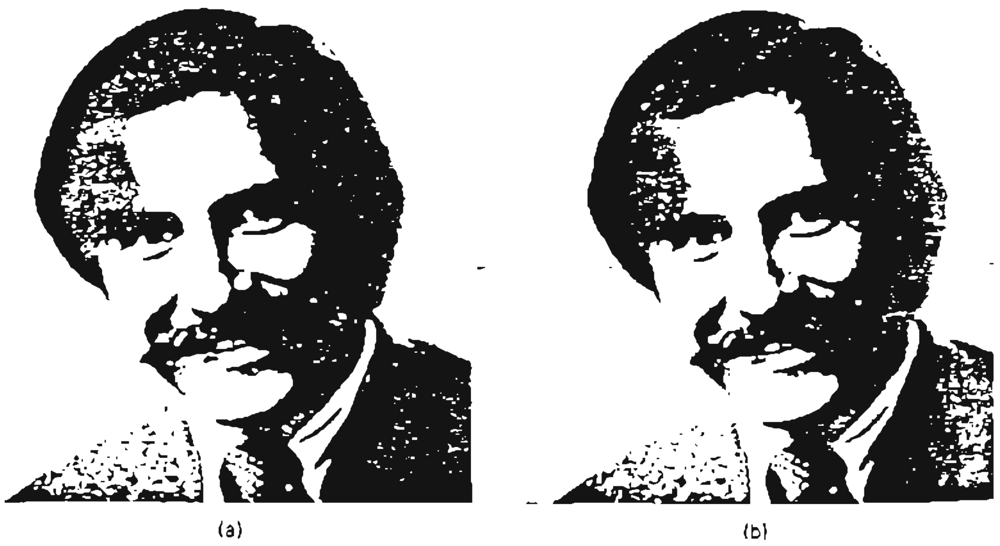


Figure 10.48 Example of DCT image coding. (a) DCT-coded image at 1 bit/pixel. NMSE = 0.8%, SNR = 20.7 dB. (b) DCT-coded image at $\frac{1}{2}$ bit/pixel. NMSE = 0.9%, SNR = 20.2 dB.

on is
512
and
size
zed.

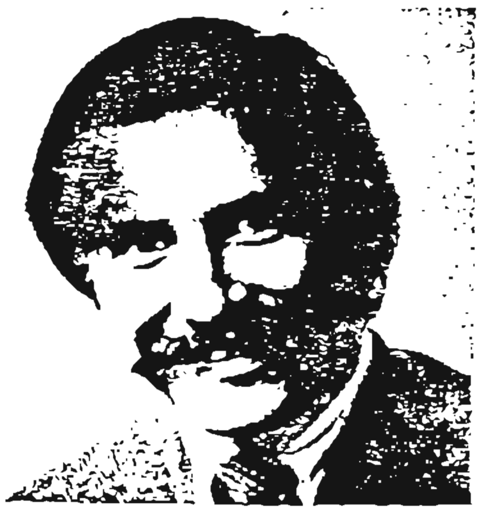


Figure 10.46 Illustration of graininess increase due to quantization of DCT coefficients. A 2-bit pixel uniform quantizer was used to quantize each DCT coefficient retained to reconstruct the image in Figure 10.45(b)

and are selected from a zone of triangular shape shown in Figure 10.43(a). From Figure 10.45, it is clear that the reconstructed image appears more blurry as we retain a smaller number of coefficients. It is also clear that an image reconstructed from only a small fraction of the transform coefficients looks quite good, illustrating the energy compaction property.

Another type of degradation results from quantization of the retained transform coefficients. The degradation in this case typically appears as graininess in the image. Figure 10.46 shows the result of coarse quantization of transform coefficients. This example is obtained by using a 2-bit uniform quantizer for each retained coefficient to reconstruct the image in Figure 10.45(b).

A third type of degradation arises from subimage-by-subimage coding. Since each subimage is coded independently, the pixels at the subimage boundaries may have artificial intensity discontinuities. This is known as the *blocking effect*, and is more pronounced as the bit rate decreases. An image with a visible blocking effect is shown in Figure 10.47. A DCT with zonal coding, a subimage of 16 × 16 pixels, and a bit rate of 0.15 bit/pixel were used to generate the image in Figure 10.47.

Examples. To design a transform coder at a given bit rate, different types of image degradation due to quantization have to be carefully balanced by a proper choice of various design parameters. As was discussed, these parameters include the transform used, subimage size, selection of which coefficients will be retained, bit allocation, and selection of quantization levels. If one type of degradation dominates other types of degradation, the performance of the coder can usually be improved by decreasing the dominant degradation at the expense of some increase in other types of degradation.

Figure 10.48 shows examples of transform image coding. Figure 10.48(a)

0

279

n the blocking effect at
ansforms called lapped
used are overlapped.
nains the same as the
:presenting a subimage
subimage size. Even
: transform coefficients
:tly from the transform
s reduces the blocking
:d, the total number of
reconstruction remains

) filter the image at the
›roach, the coding pro-
utually exclusive. The
:aused by segmentation
iency components. In
.ubimage boundaries to
) procedure used at the
ıl image discontinuities
thod does not increase
ing method was shown
DCT in reducing the
w the filtering method
the lapped orthogonal
n example of reduction
).50(a) shows an image
.e processed by applying
only the pixels at the

rate applications, while
. A hybrid transform/
aveform and transform
`orm-coding methods at
ıan true 2-D transform

'y a 1-D transform, such
$T_f(k_1, n_2)$ is then coded
(or row). This is illus-
each row of data well.
Due to the transform,
ıave been by waveform
ıentation issues such as

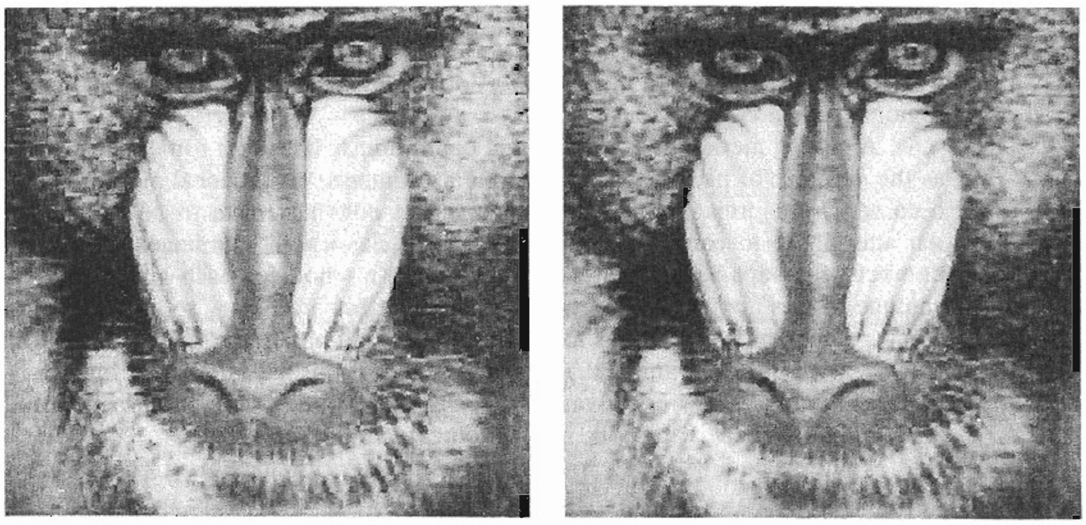


Figure 10.50 Example of blocking effect reduction using a filtering method. (a) Image of 512 × 512 pixels with visible blocking effect. The image is coded by a zonal DCT coder at 0.2 bit/pixel. (b) Image in (a) filtered to reduce the blocking effect. The filter used is a 3 × 3-point $h(n_1, n_2)$ with $h(0, 0) = \frac{1}{5}$ and $h(n_1, n_2) = \frac{1}{10}$ at the remaining eight points.

selection of the zone shape and size in zonal coding are simpler than those with a 2-D transform coder. Hybrid coding of a single image frame has not been used extensively in practice, perhaps because the method does not reduce the correlation in the data as much as a 2-D transform coder and the complexity in a 2-D transform coder implementation is not much higher than a hybrid coder. As will be discussed in Section 10.6, however, hybrid coding is useful in interframe image coding.

10.4.5 Adaptive Coding and Vector Quantization

Transform coding techniques can be made adaptive to the local characteristics within each subimage. In zonal coding, for example, the shape and size of the

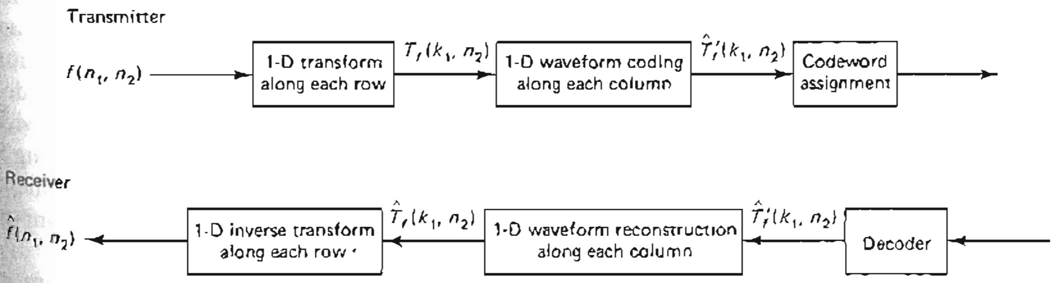


Figure 10.51 Hybrid transform/waveform coder.

Iterative Procedures for Reduction of Blocking Effects in Transform Image Coding

Ruth Rosenholtz and Avideh Zakhor

Abstract—We propose a new iterative block reduction technique based on the theory of projection onto convex sets. The basic idea behind this technique is to impose a number of constraints on the coded image in such a way as to restore it to its original artifact-free form. One such constraint can be derived by exploiting the fact that the transform-coded image suffering from blocking effects contains high-frequency vertical and horizontal artifacts corresponding to vertical and horizontal discontinuities across boundaries of neighboring blocks. Since these components are missing in the original uncoded image, or at least can be guaranteed to be missing from the original image prior to coding, one step of our iterative procedure consists of projecting the coded image onto the set of signals that are bandlimited in the horizontal or vertical directions. Another constraint we have chosen in the restoration process has to do with the quantization intervals of the transform coefficients. Specifically, the decision levels associated with transform coefficient quantizers can be used as lower and upper bounds on transform coefficients, which in turn define boundaries of the convex set for projection. Thus, in projecting the "out-of-bound" transform coefficient onto this convex set, we will choose the upper (lower) bound of the quantization interval if its value is greater (less) than the upper (lower) bound. We present a few examples of our proposed approach.

Manuscript received June 10, 1991; revised February 3, 1992. This work has been supported by IBM, Eastman Kodak Company, TRW, and the National Science Foundation contract MIP-9057466. This paper was recommended by Associate Editor Dimitris Anastassiou.

The author is with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

IEEE Log Number 9107519.

I. Introduction

Transform coding is one of the most widely used image compression techniques. It is based on dividing an image into small blocks, taking the transform of each block and discarding high-frequency coefficients and quantizing low-frequency coefficients. Among various transforms, the discrete cosine transform (DCT) is one of the most popular because its performance for certain class of images is close to that of the Karhunen-Loeve transform (KLT), which is known to be optimal in the mean squared error sense.

Although DCT is used in most of today's standards such as JPEG and MPEG, its main drawback is what is usually referred to as the "blocking effect." Dividing the image into blocks prior to coding causes blocking effects—discontinuities between adjacent blocks—particularly at low bit rates. In this paper, we present an iterative technique for the reduction of blocking effects in coded images.

II. Iterative Restoration Method

The block diagram of our proposed iterative approach is shown in Fig. 1. The basic idea behind our technique is to impose a number

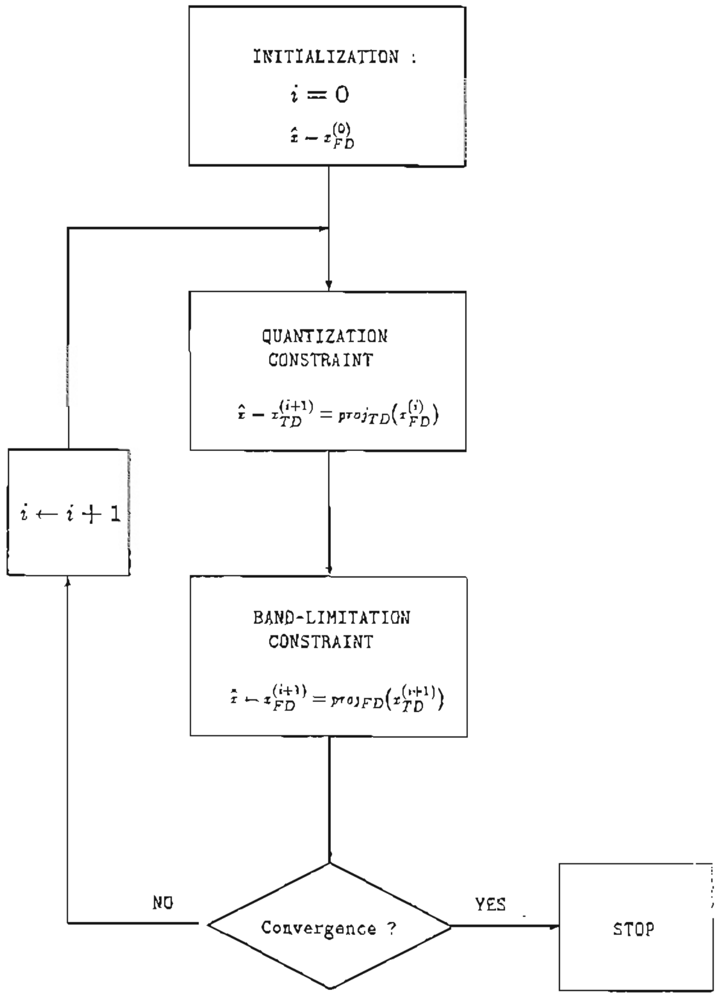

INITIALIZATION :
$i = 0$
$\hat{x} \leftarrow x_{FD}^{(0)}$
QUANTIZATION CONSTRAINT
$\hat{x} \leftarrow x_{TD}^{(i+1)} = proj_{TD}(x_{FD}^{(i)})$
BAND-LIMITATION CONSTRAINT
$\hat{x} \leftarrow x_{FD}^{(i+1)} = proj_{FD}(x_{TD}^{(i+1)})$
$i \leftarrow i+1$
NO
Convergence ?
YES
STOP

Fig. 1. Block diagram of the iterative algorithm.

of constraints on the coded image in such a way as to restore it to its original artifact-free-form. We derive one such constraint from the fact that the coded image with $N \times N$ blocks has high-frequency horizontal and vertical artifacts corresponding to the discontinuities at the edges of the $N \times N$ blocks. Therefore, one step of our procedure consists of bandlimiting the image in the horizontal and vertical directions. We refer to this constraint as the filtering constraint.

We derive the second constraint from the quantizer and thus refer to it as the quantization constraint. Because the quantization intervals for each DCT coefficient is assumed to be known in decoding a DCT encoded image, the quantization constraint ensures that in restoring images with blocking effects, DCT coefficients of $N \times N$ blocks remain in their original quantization interval.

If S_1 denotes the set of bandlimited images, and S_2 denotes the set of images whose $N \times N$ DCT coefficients lie in specific quantization intervals, our goal can be stated as that of finding an image in the intersection of S_1 and S_2. One way to achieve this is to start with an arbitrary element in either of the two sets and iteratively map it back and forth to the other set, until the process converges to an element in the intersection of the two sets. Under these conditions convergence can be guaranteed by the theory of projection onto convex sets (POCS) if sets S_1 and S_2 are convex, and if the mapping from each set to the other is a projection [6]. By definition, the projection of an element x in set A onto set B is equivalent to finding the closest element, according to some metric, in B to x.

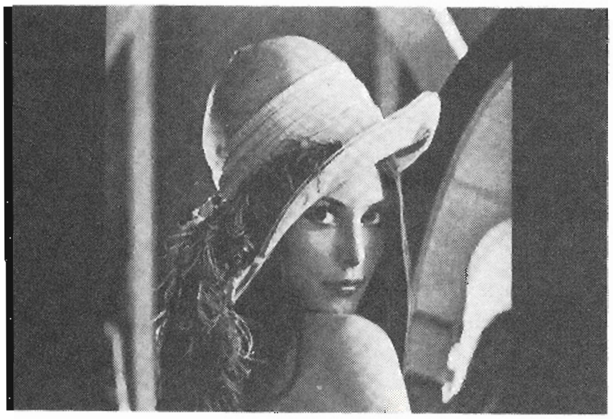
To apply the above idea to our problem, we first notice that two sets S_1 and S_2 are both convex. We also choose the mean squared error as our metric of closeness. This implies that a projection from S_2 to S_1 can be accomplished by any bandlimitation algorithm such as ideal low-pass filtering. It also implies that projection from S_1 to S_2 can be accomplished by moving $N \times N$ DCT coefficients that are outside their designated quantization interval to the closest boundary of their respective quantization intervals. For instance, if a particular $N \times N$ DCT coefficient, which is supposed to be in the range $[a, b]$, takes on a value greater than b, it is projected to b. Alternatively, if it takes on a value smaller than a it is projected onto a.

Having explained the constraints, convex sets, and projections, we now summarize our proposed iterative procedure shown in Fig. 1. In the first part of each iteration, we low pass filter, or bandlimit, the image that has high-frequency horizontal and vertical components corresponding to the discontinuities between $N \times N$ blocks. In the second part of each iteration we apply the quantization constraint as follows. First we divide the image into $N \times N$ blocks and take the DCT of each. Then we project any coefficient outside its quantization range onto its appropriate value. Under these conditions, the POCS theory guarantees that iterative projection between the sets S_1 and S_2 results in convergence to an element in the intersection of the two sets.

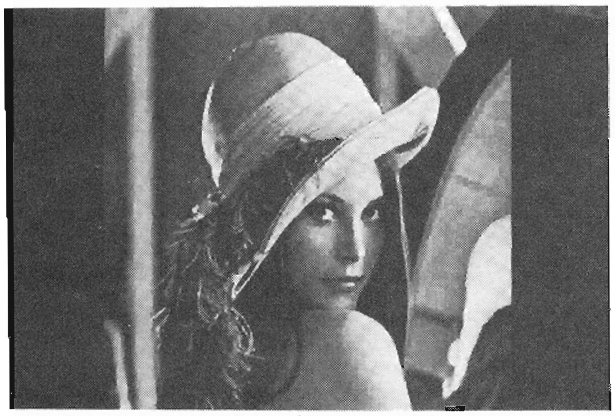
III. Experimental Results

Fig. 2(a) shows the original, unquantized 512×512 Lena, and (b), (c), and (d) show its JPEG encoded version to 0.43, and 0.24,

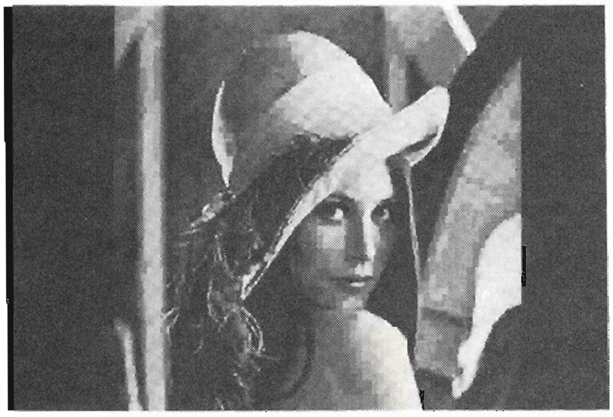
(a)



(b)



(c)



(d)

Fig. 2(a) Original 512 × 512 image, Lena. 2(b) Lena quantized to 0.43 bpp. 2(c) Lena quantized to 0.24 bpp. 2(d) Lena quantized to 0.15 bpp.

and 0.15 bpp, respectively. The quantization tables for Figs. 2(b), (c), and (d) are included in the Appendix.

Strictly speaking, the band-limitation portion of our algorithm corresponds to a true projection if the image under consideration is convolved with an ideal low-pass filter. Since an ideal low-pass filter cannot be implemented in practice, we have chosen to approximate it with a 3 × 3 finite impulse response (FIR) filter of the form

$$\begin{aligned} h(0,0) &= 0.2042, \\ h(0,1) &= h(0,-1) = h(1,0) = h(-1,0) = 0.1239 \quad (1) \\ h(0,2) &= h(0,-2) = h(2,0) = h(2,0) = 0.0751. \end{aligned}$$

We now show examples of our iterative algorithm. Fig. 3(a) shows five iterations of our algorithm applied to the 0.43-bpp quantized image of Fig. 2(b). The FIR filter of (1) was used for the band-limitation step. As Fig. 2(b) shows, blocking artifact has been removed without introducing excessive blurring. For comparison purposes, the result of applying the low-pass filter in (1) to Fig. 2(b) for five times, without applying the quantization constraint, is also shown in Fig. 3(b). Although consecutive low-pass filtering removes most of the blocking effect, it blurs the image in a noticeable way. We have found that applying the low-pass filter of (1) once rather than five times, results in a less blurry image than in Fig. 3(b), but at the same time does not remove all the blocking effect.

Figs. 4(a) and (b) show application of our algorithm to the 0.24-bpp quantized image of Fig. 2(c) for 5 and 20 iterations, respectively. The FIR filter of (1) was used for the band-limitation step. As seen, the blocking artifact is better removed in Fig. 4(b) than in 4(a), while they are as sharp as each other. For comparison purposes, Fig. 4(c) and (d) show the result of applying the low-pass filter of (1) to Fig. 2(c), 5 and 20 times, respectively. Comparing Fig. 4(c) and 4(d) to Fig. 4(a) and (b), respectively, we find that the latter pair are more blurry than the former. Thus, applying the quantization constraint prevents the images from becoming excessively blurry.

Fig. 5(a) shows application of our algorithm to the 0.15-bpp quantized image of Fig. 2(d) for 20 iterations. The FIR filter of (1) was used for the band-limitation step. For comparison purposes, Fig. 5(b) shows the result of applying the low-pass filter of (1) to Fig. 2(d), 20 times. Comparing Fig. 5(b) to 5(a), we find that the latter is considerably more blurry than the former.

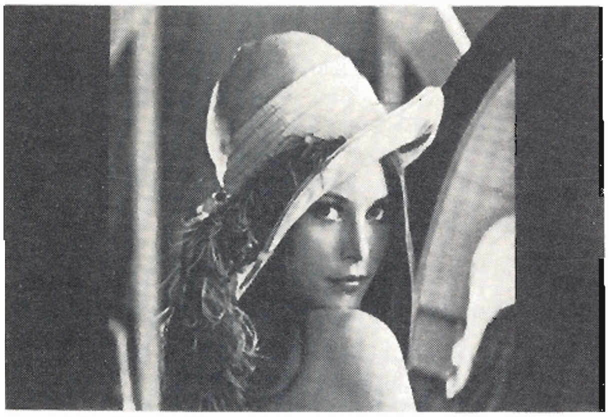
IV. Conclusions

The major conclusions to be drawn from this paper are as follows: 1) the proposed iterative algorithm using a 3 × 3 low-pass filtering of (1) results in images that are free of blocking artifacts and excessive blurring; 2) low-pass filtering by itself could remove blockiness but at the expense of increased blurriness.

It is conceivable to generate images similar to Figs. 5(a) and 4(b) without having to apply our algorithm for as many as 20 iterations. Our conjecture is that this could be achieved by increasing the region of support of the impulse response of the filter of (1). In practical hardware implementations however, 3 × 3 convolvers are more readily available than, say, 30 × 30 ones.

We have checked the convergence of our algorithm and found that it converges after 20 iterations or so. This is encouraging since there is no guarantee that the intersections of our particular convex sets is nonempty, and the theory of POCS only guarantees convergence in situations where the intersection is nonempty.

One way to increase the likelihood of convergence is to vary the confidence with which the ideal solution is in the chosen constraint set, by varying its size. For example, if we choose prototype constraint sets as in [10], using the statistics of the


(a)

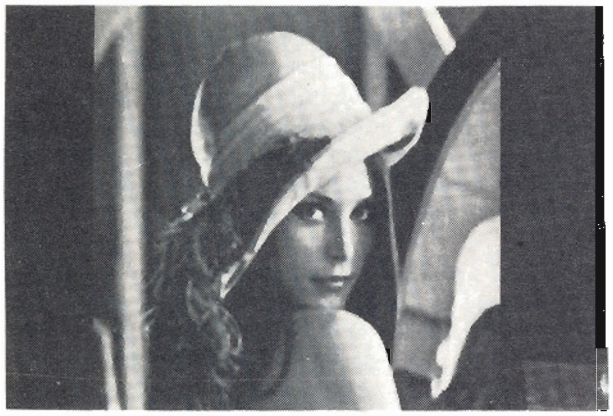
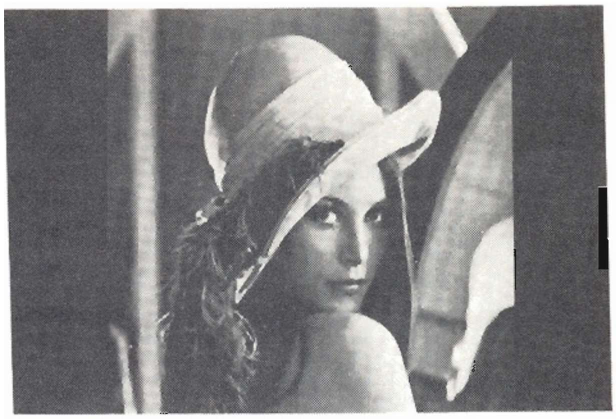
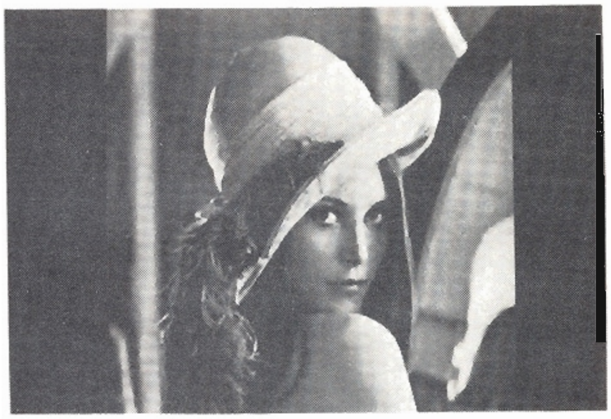
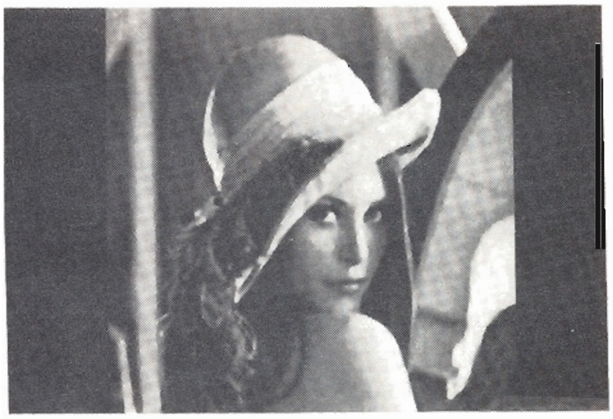

(b)

Fig. 3(a) Result of applying the iterative algorithm to Fig. 2(b) for five iterations with the low-pass filter of (1) used for bandlimitation. (b) Result of low-pass filtering Fig. 2(b) five times using the filter in (1).


(a)


(b)


(c)

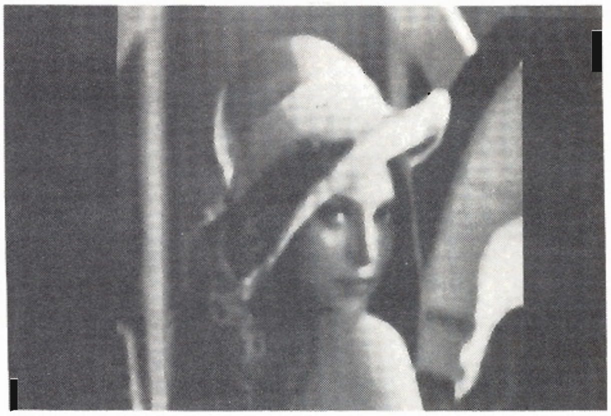

(d)

Fig. 4(a) Result of applying the iterative algorithm to Fig. 2(c) for 5 iterations with the low-pass filter of (1) used for bandlimitation. (b) Result of applying the iterative algorithm to Fig. 2(c) for 20 iterations with the low-pass filter of (1) used for bandlimitation. (c) Result of low-pass filtering Fig. 2(c) five times using the filter in (1). (d) Result of low-pass filtering Fig. 2(c) 20 times using the filter in (1).

quantization noise, we can change the boundaries and the size of the constraint set in a controlled fashion and therefore increase the likelihood of a solution in the intersection of the constraint sets. Examples of such prototype constraint sets include bounded variation from the Weiner solution and pointwise adaptive smoothness. The latter constraint has the obvious advantage of being locally adaptive to changes in the characteristics of the image. Projection onto fuzzy sets is another way of increasing the size of our convex sets [9].

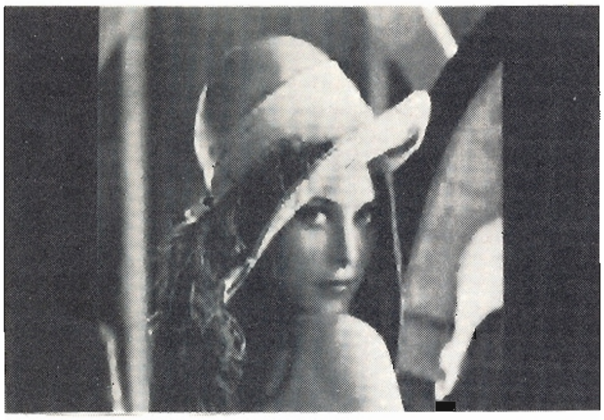
Appendix

The quantization table for Fig. 2(b) is

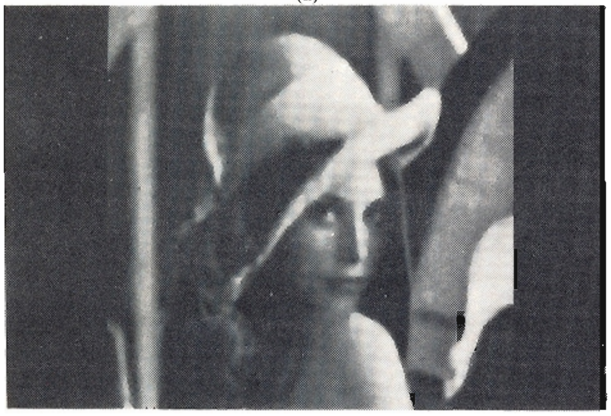
20	24	28	32	36	80	98	144
24	24	28	34	52	70	128	184
28	28	32	48	74	114	156	190
32	34	48	58	112	128	174	196
36	52	74	112	136	162	206	224
80	70	114	128	162	208	242	200
98	128	156	174	206	242	240	206
144	184	190	196	224	200	206	208

For Fig. 2(c) it is

50	60	70	70	90	120	255	255
60	60	70	96	130	255	255	255
70	70	80	120	200	255	255	255
70	96	120	145	255	255	255	255



(a)



(b)

Fig. 5(a) Result of applying the iterative algorithm to Fig. 2(d) for 20 iterations with the low-pass filter of (1) used for bandlimitation. (b) Result of low pass filtering Fig 2(d) 20 times using the filter in (1).

90	130	200	255	255	255	255	255
120	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

and for Fig. 2(d) it is:

110	130	150	192	255	255	255	255
130	150	192	255	255	255	255	255
150	192	255	255	255	255	255	255
192	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

The 255 entry in the above tables indicates that the coefficient was discarded.

References

[1] H. C. Reeve and J. S. Lim, "Reduction of blocking effects in image coding," *Optical Eng.*, vol. 23, no. 1, pp. 34-37, Jan./Feb. 1984.

[2] J. Biemond, R. L. Lagendijk, and R. M. Mersereau, "Iterative methods for image deblurring," *Proc. IEEE*, vol. 78, no. 5, pp. 856-883, May 1990.

[3] S. N. Efstratiadis and A. K Katsaggelos, "Adaptive iterative image restoration with reduced computational load," *Optical Eng.*, vol. 29, no. 12, pp. 1458-1468, Dec. 1990.

[4] S. J. Reeves and R. M. Mersereau, "Optimal estimation of the regularization parameter and stabilizing functional for regularized image restoration," *Opt. Eng.*, vol. 29, no. 5, pp. 446-454, May 1990.

[5] B. J. Sullivan and A. K. Katsaggelos, "New termination rule for linear iterative image restoration algorithms," *Opt. Eng.*, vol. 29 no. 5, pp. 471-477, May 1990.

[6] D. C. Youla and H. Webb, "Image restoration by the method of convex projections: Part I—Theory," *IEEE Trans. Med. Imag.* vol. 1, no. 2, pp. 81-94, Oct. 1982.

[7] A. Zakhor and A. V. Oppenheim, "Reconstruction of two-dimensional signals from level crossings," *Proc. IEEE*, vol. 78, no. 1, pp. 31-55, Jan. 1990.

[8] A. Abo-Taleb and M. M. Fahmy, "Design of FIR two-dimensional digital filters by successive projections," *IEEE Trans. Circuits Syst.*, vol. CAS-31, no. 9, pp. 801-805, Sept. 1984.

[9] M. R. Civanlar and H. J. Trussell, "Digital image restoration using fuzzy sets," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. 34, pp. 919-936, 1986.

[10] "Set-theoretic image restoration," *IEEE Trans. Signal Processing*," *vol.* 39, *no.* 10, *pp.* 2275-2285, *Oct.* 1991.

Hybrid Coding

- Combines waveform and transform coding.
  - Implementation is simpler than 2-D transform coding.
  - Better performance than waveform coding.
- Basic Idea:
  - Transform an image $f(n_1, n_2)$ by a 1-D transform such as a 1-D DCt along each row to obtain $T_f(k_1, n_2)$.
  - Remove more redundancy along each column by DPCM.
- Hybrid coding useful in interframe coding.

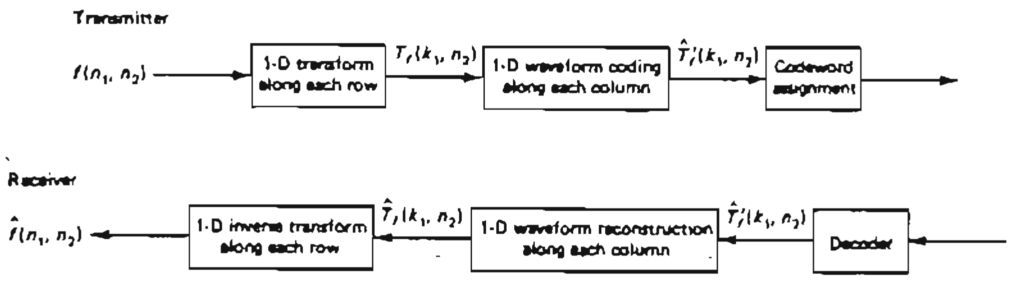


Figure 10.51 Hybrid transform/waveform coder.

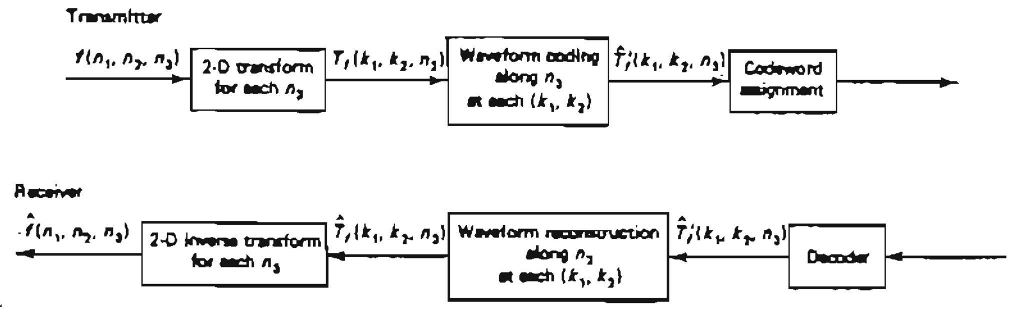
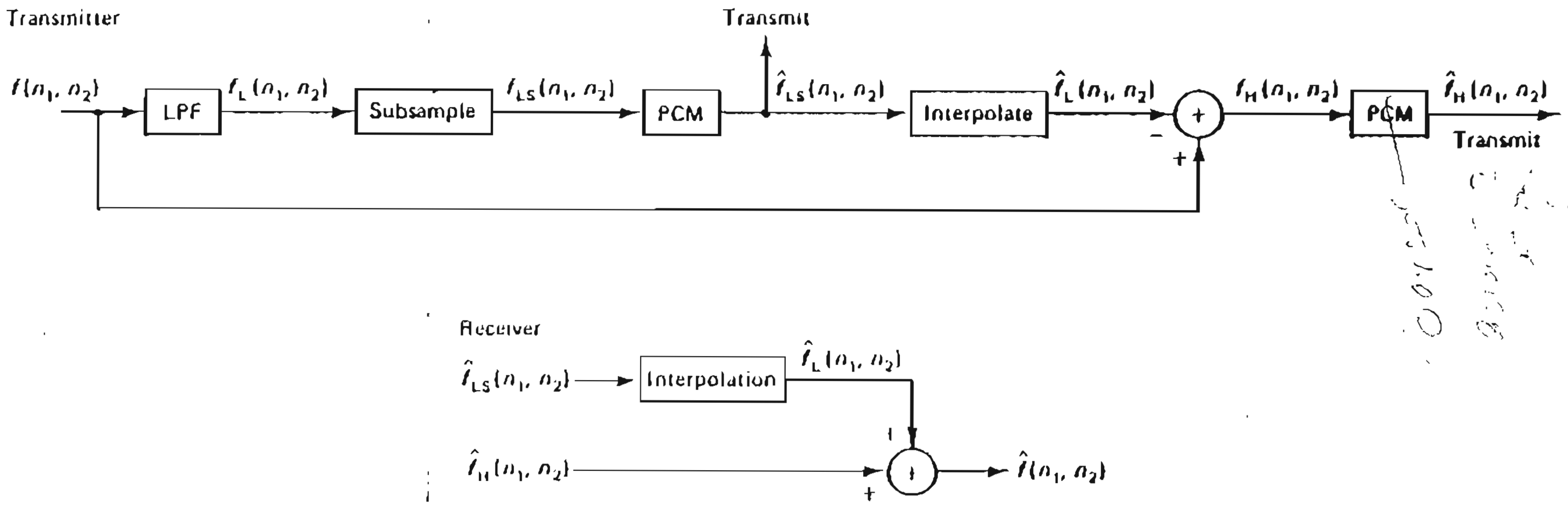

Transmitter
$f(n_1, n_2, n_3)$
2-D transform for each n_3
$T_f(k_1, k_2, n_3)$
Waveform coding along n_3 at each (k_1, k_2)
$\hat{T}_f'(k_1, k_2, n_3)$
Codeword assignment
Receiver
$\hat{f}(n_1, n_2, n_3)$
2-D inverse transform for each n_3
$\hat{T}_f(k_1, k_2, n_3)$
Waveform reconstruction along n_3 at each (k_1, k_2)
$\hat{T}_f'(k_1, k_2, n_3)$
Decoder

Figure 10.55 Interframe hybrid coder.

Two-Channel Image Coder



[illegible]

$f_L(n_1, n_2)$: Can be under-sampled (typically by 8×8), but requires above 5 bits/sample

$f_H(n_1, n_2)$: Cannot be under-sampled, but can be coarsely quantized (2-3 bits/pixel)

$$\text{Bit rate} \approx \frac{5}{64} + 2\text{–}3 \text{ bits/pixel} \approx 2\text{–}3 \text{ bits/pixel}$$

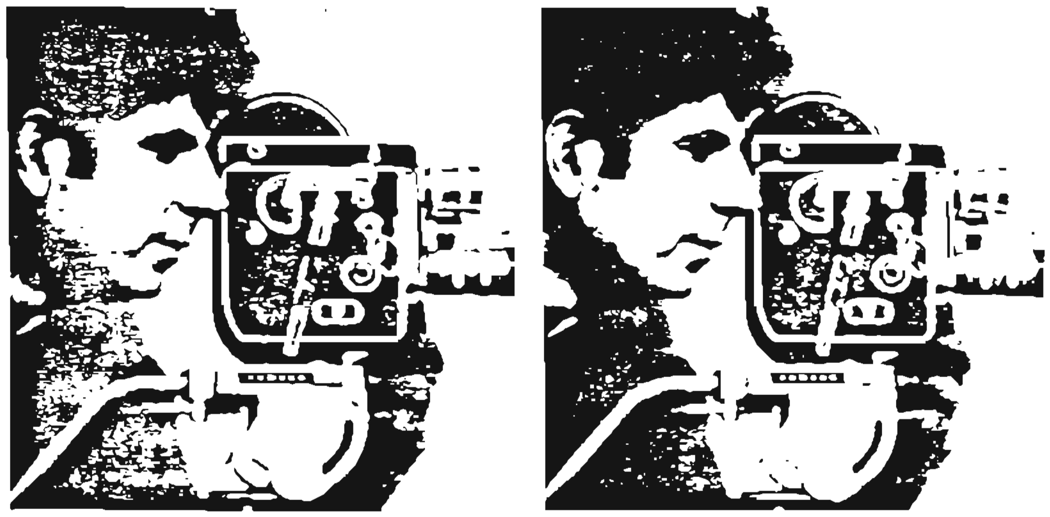


Figure 10.32 Example of image coding by a two-channel coder. (a) Original image of 512 × 512 pixels. (b) coded image at 3/4 bits/pixel. NMSE = 1.0%, SNR = 19.8 dB.

It is possible to develop many image representations [Rosenfeld] that can be viewed as pyramids. In this section, we discuss one particular representation developed by [Burt and Adelson]. This pyramid representation consists of an original image and successively lower resolution (blurred) images and can be used for image coding.

Let $f_0(n_1, n_2)$ denote an original image of $N \times N$ pixels where $N = 2^M + 1$, for example, 129 × 129, 257 × 257, 513 × 513, and so forth. It is straightforward to generate an image of $(2^M + 1) \times (2^M + 1)$ pixels from an image of $2^M \times 2^M$ pixels, for example, by simply repeating the last column once and the last row once. We assume a square image for simplicity. We will refer to $f_0(n_1, n_2)$ as the base level image of the pyramid. The image at one level above the base is obtained by lowpass filtering $f_0(n_1, n_2)$ and then subsampling the result. Suppose we filter $f_0(n_1, n_2)$ with a lowpass filter $h_0(n_1, n_2)$ and denote the result by $f_0^L(n_1, n_2)$ so that

$$f_0^L(n_1, n_2) = L[f_0(n_1, n_2)] = f_0(n_1, n_2) * h_0(n_1, n_2) \tag{10.43}$$

where $L[\cdot]$ is the lowpass filtering operation. Since $f_0^L(n_1, n_2)$ has a lower spatial resolution than $f_0(n_1, n_2)$ due to lowpass filtering, we can subsample $f_0^L(n_1, n_2)$. We denote the result of the subsampling operation by $f_1(n_1, n_2)$. The image $f_1(n_1, n_2)$ is smaller in size than $f_0(n_1, n_2)$ due to subsampling and is the image at one level above the base of the pyramid. We will refer to $f_1(n_1, n_2)$ as the first-level image of the pyramid. The second-level image, $f_2(n_1, n_2)$, is obtained by lowpass filtering the first-level image $f_1(n_1, n_2)$ and then subsampling the result. This procedure can be repeated to generate higher level images $f_3(n_1, n_2)$, $f_4(n_1, n_2)$, and so forth.

Pyramid Coding and Subband Coding

- Basic Idea: Successive lowpass filtering and subsampling.

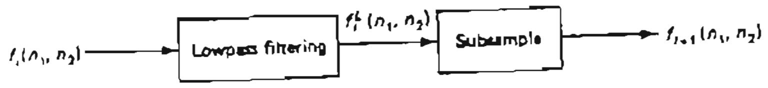


Figure 10.33 Process of generating the $i+1$th-level image $f_{i+1}(n_1, n_2)$ from the ith-level image $f_i(n_1, n_2)$ in Gaussian pyramid image representation.

- Filtering:

$$f_i^L(n_1, n_2) = f_i(n_1, n_2) * h(n_1, n_2)$$

- Subsampling.

$$f_{i+1}(n_1, n_2) = \begin{cases} f_i^L(2n_1, 2n_2) & 0 \le n_1, n_2 \le 2^{M-1} \\ 0 & Otherwise \end{cases}$$

- Type of filter determines the kind of pyramid.
- Gaussian pyramid: $h(n_1, n_2) = h(n_1)h(n_2)$

$$h(n) = \begin{cases} a & n = 0 \\ \frac{1}{4} & n = \pm 1 \\ \frac{1}{4} - \frac{a}{2} & n = \pm 2 \end{cases}$$

a is between .3 and .6

26

Pyramid Coding and Subband Coding

- Application to image coding:
  - Code successive images down the pyramid from the ones above it.
  - Use intrafram coding techniques to code the image at top of the pyramid.
  - Interpolate $f_{i+1}(n_1, n_2)$ to obtain a prediction for $f_i(n_1, n_2)$.

$$\hat{f}_i(n_1, n_2) = I[f_{i+1}(n_1, n_2)]$$

  - Code the prediction error:

$$e_i(n_1, n_2) = f_i(n_1, n_2) - \hat{f}_i(n_1, n_2)$$

    to construct $f_i(n_1, n_2)$.
  - Repeat until the bottom level image, i.e. the original is reconstructed.
- The sequence $f_i(n_1, n_2)$ is a *Gaussian Pyramid.*
- The sequence $e_i(n_1, n_2)$ is a *Laplacian Pyramid.*
- Other examples of Pyramid coding:
  - Subband coding.
  - Wavelet coding.

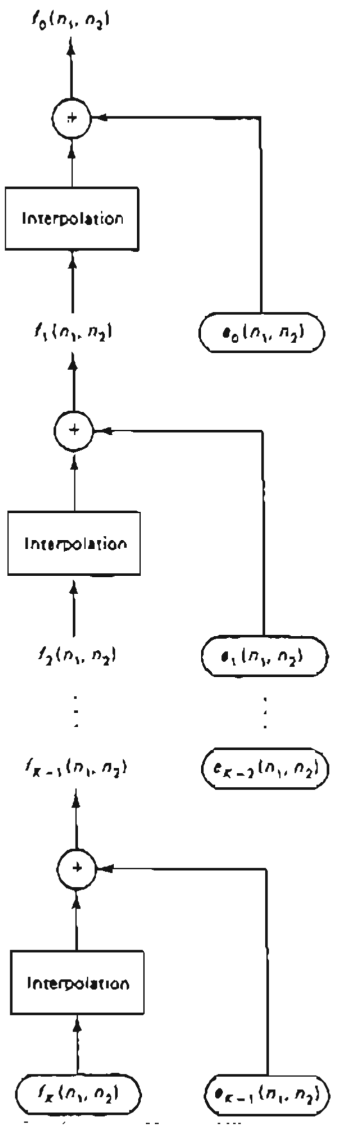


Figure 10.37 Laplacian pyramid generation. The base image $f_0(n_1, n_2)$ can be reconstructed from $e_i(n_1, n_2)$ for $0 \le i \le K - 1$ and $f_K(n_1, n_2)$.

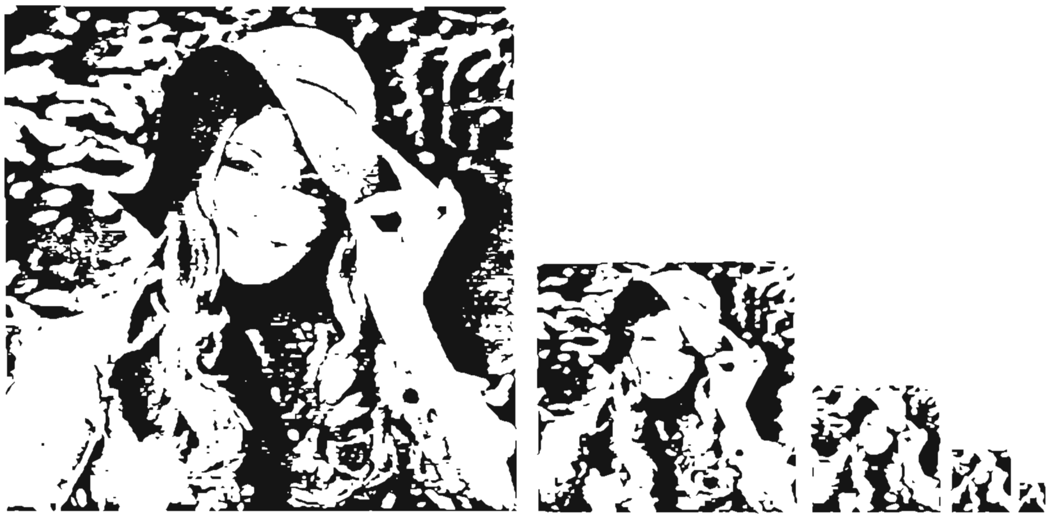


Figure 10.36 Example of the Gaussian pyramid representation for image of 513 × 513 pixels with $K = 4$.

The Gaussian pyramid representation can be used in developing an approach to image coding. To code the original image $f_0(n_1, n_2)$, we code $f_1(n_1, n_2)$ and the difference between $f_0(n_1, n_2)$ and a prediction of $f_0(n_1, n_2)$ from $f_1(n_1, n_2)$. Suppose we predict $f_0(n_1, n_2)$ by interpolating $f_1(n_1, n_2)$. Denoting the interpolated image by $f_1^I(n_1, n_2)$, we find that the error signal $e_0(n_1, n_2)$ coded is

$$\begin{aligned} e_0(n_1, n_2) &= f_0(n_1, n_2) - I[f_1(n_1, n_2)] \\ &= f_0(n_1, n_2) - f_1^I(n_1, n_2) \end{aligned} \tag{10.46}$$

where $I[\cdot]$ is the spatial interpolation operation. The interpolation process expands the support size of $f_1(n_1, n_2)$, and the support size of $f_1^I(n_1, n_2)$ is the same as $f_0(n_1, n_2)$. One advantage of coding $f_1(n_1, n_2)$ and $e_0(n_1, n_2)$ rather than $f_0(n_1, n_2)$ is that the coder used can be adapted to the characteristics of $f_1(n_1, n_2)$ and $e_0(n_1, n_2)$. If we do not quantize $f_1(n_1, n_2)$ and $e_0(n_1, n_2)$, from (10.46) $f_0(n_1, n_2)$ can be recovered exactly by

$$f_0(n_1, n_2) = I[f_1(n_1, n_2)] + e_0(n_1, n_2). \tag{10.47}$$

In image coding, $f_1(n_1, n_2)$ and $e_0(n_1, n_2)$ are quantized and the reconstructed image $\hat{f}_0(n_1, n_2)$ is obtained from (10.47) by

$$\hat{f}_0(n_1, n_2) = I[\hat{f}_1(n_1, n_2)] + \hat{e}_0(n_1, n_2) \tag{10.48}$$

where $\hat{f}_0(n_1, n_2)$ and $\hat{e}_0(n_1, n_2)$ are quantized versions of $f_0(n_1, n_2)$ and $e_0(n_1, n_2)$. If we stop here, the structure of the coding method is identical to the two-channel coder we discussed in the previous section. The image $f_1(n_1, n_2)$ can be viewed as the subsampled lows component $f_{LS}(n_1, n_2)$ and $e_0(n_1, n_2)$ can be viewed as the highs component $f_H(n_1, n_2)$ in the system in Figure 10.31.

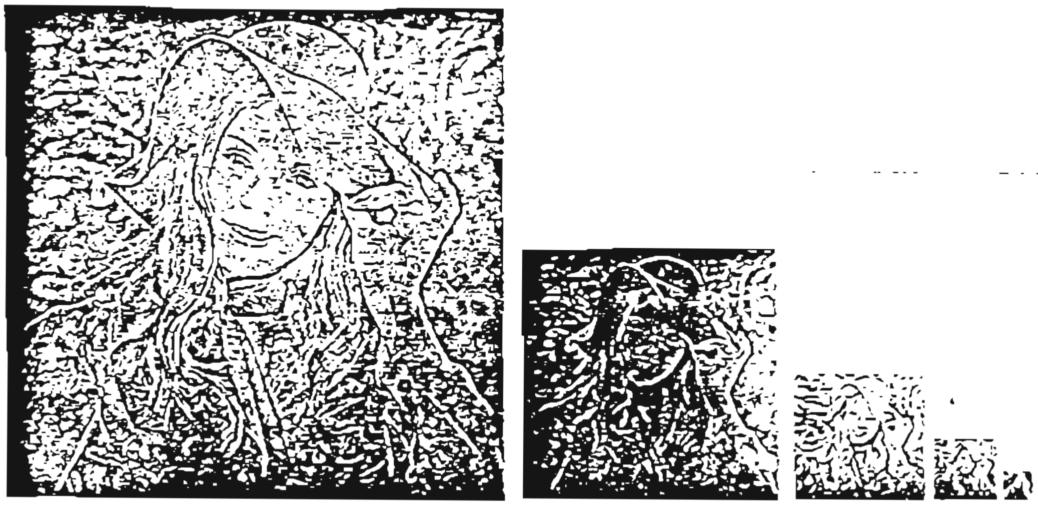


Figure 10.38 Example of the Laplacian pyramid image representation with $K = 4$. The original image used is the 513 × 513-pixel image $f_0(n_1, n_2)$ in Figure 10.36. $e_i(n_1, n_2)$ for $0 \le i \le 3$ and $f_4(n_1, n_2)$.

the difference of the two Gaussian functions. The difference of two Gaussians can be modeled [Marr] approximately by the Laplacian of a Gaussian, hence the name "Laplacian pyramid."

From the above discussion, the pyramid coding method we discussed can be viewed as an example of subband image coding. As we have stated briefly, in subband image coding, an image is divided into different frequency bands and each band is coded with its own coder. In the pyramid coding method we discussed, the bandpass filtering operation is performed implicitly and the bandpass filters are obtained heuristically. In a typical subband image coder, the bandpass filters are designed more theoretically [Vetterli; Woods and O'Neil].

Figure 10.39 illustrates the performance of an image coding system in which $f_K(n_1, n_2)$ and $e_i(n_1, n_2)$ for $0 \le i \le K-1$ are coded with coders adapted to the signal characteristics. Qualitatively, higher-level images have more variance and more bits/pixel are assigned. Fortunately, however, they are smaller in size. Figure 10.39 shows an image coded at $\frac{1}{2}$ bit/pixel. The original image used is the 513 × 513-pixel image $f_0(n_1, n_2)$ in Figure 10.36. The bit rate of less than 1 bit/pixel was possible in this example by entropy coding and by exploiting the observation that most pixels of the 513 × 513-pixel image $e_0(n_1, n_2)$ are quantized to zero.

One major advantage of the pyramid-based coding method we discussed above is its suitability for progressive data transmission. By first sending the top-level image $f_K(n_1, n_2)$ and interpolating it at the receiver, we have a very blurred image. We then transmit $e_{K-1}(n_1, n_2)$ to reconstruct $f_{K-1}(n_1, n_2)$, which has a higher spatial resolution than $f_K(n_1, n_2)$. As we repeat the process, the reconstructed image at the receiver will have successively higher spatial resolution. In some applications, it may be possible to stop the transmission before we fully

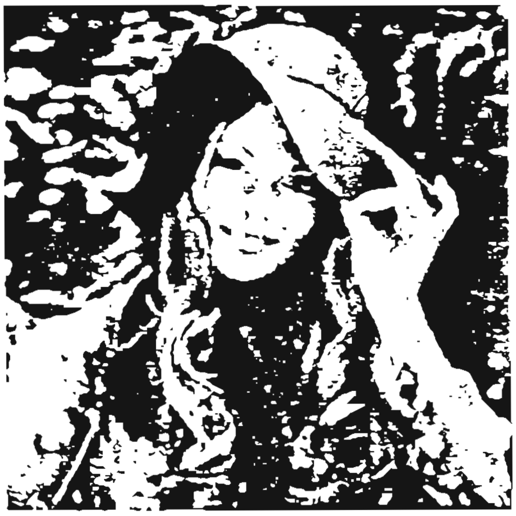


Figure 10.39 Example of the Laplacian pyramid image coding with $K = 4$ at $\frac{1}{2}$ bit/pixel. The original image used is the 513×513-pixel image $f_0(n_1, n_2)$ in Figure 10.36.

recover the base level image $f_0(n_1, n_2)$. For example, we may be able to judge from a blurred image that the image is not what we want. Fortunately, the images are transmitted from the top to the base of the pyramid. The size of images increases by approximately a factor of four as we go down each level of the pyramid

In addition to image coding, the Laplacian pyramid can also be used in other applications. For example, as we discussed above, the result of repetitive interpolation of $e_i(n_1, n_2)$ such that its size is the same as that of $f_0(n_1, n_2)$ can be viewed as approximately the result of filtering $f_0(n_1, n_2)$ with the Laplacian of a Gaussian. As we discussed in Section 8.3.3, zero-crossing points of the result of filtering $f_0(n_1, n_2)$ with the Laplacian of a Gaussian are the edge points in the edge detection method by Marr and Hildreth.

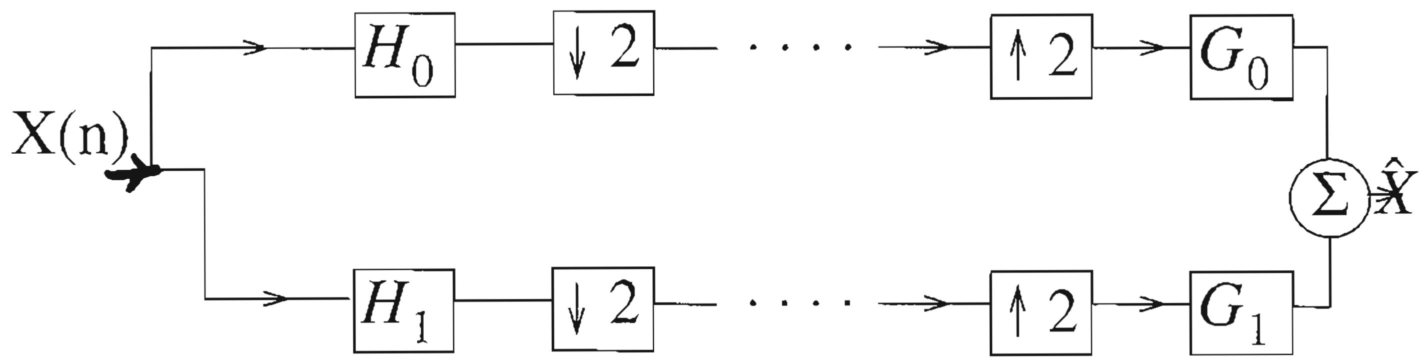
10.3.6 Adaptive Coding and Vector Quantization

The waveform coding techniques discussed in previous sections can be modified to adapt to changing local image characteristics. In a PCM system, the reconstruction levels can be chosen adaptively. In a DM system, the step size Δ can be chosen adaptively. In regions where the intensity varies slowly, for example, Δ can be chosen to be small to reduce granular noise. In regions where the intensity increases or decreases rapidly, Δ can be chosen to be large to reduce the slope overload distortion problem. In a DPCM system, the prediction coefficients and the reconstruction levels can be chosen adaptively. Reconstruction levels can also be chosen adaptively in a two-channel coder and a pyramid coder. The number of bits assigned to each pixel can also be chosen adaptively in all the waveform coders we discussed. In regions where the quantized signal varies very slowly, for example, we may want to assign a smaller number of bits/pixel. It is also possible to have a fixed number of bits/frame, while the bit rate varies at a pixel level.

In adaptive coding, the parameters in the coder are adapted based on some

271

Subband Coding



$$\hat{X}(\omega) = \frac{1}{2}[H_0(\omega)G_0(\omega) + H_1(\omega)G_1(\omega)]X(\omega) +$$

$$\frac{1}{2}[H_0(\omega + \pi)G_0(\omega) + H_1(\omega + \pi)G_1(\omega)]X(\omega + \pi)$$

Consider QMF Filters:

$$\left|\begin{aligned} H_0(\omega) &= G_0(-\omega) = F(\omega) \\ H_1(\omega) &= G_1(-\omega) = e^{j\omega}F(-\omega + \pi) \end{aligned}\right.$$

$$\longrightarrow \quad \hat{X}(\omega) = \frac{1}{2}[F(\omega)F(-\omega) + F(-\omega + \pi)F(\omega + \pi)]X(\omega)$$

$$\text{IMPOSE:} \quad |F(\omega)|^2 + |F(\omega + \pi)|^2 = 2$$

$$\longrightarrow \quad \hat{X}(\omega) = X(\omega) \longrightarrow \text{Perfect Reconstruction}$$

Filter Design:

- QMF filters:

$$h_1(n) = (-1)^n h_o(N-1-n)$$

$$N = \# \text{ of taps}$$

- Johnston's filter coefficients

$$h_0(N-1-n) = h_0(n)$$

$\longrightarrow$ symetric $\longrightarrow$ NPR

8 tap Johnston filters:

h (0) = h (7) = 0.00938

h (1) = h (6) = 0.06942

h (2) = h (5) = -0.07065

h (3) = h (4) = 0.489980

Filter Design

- Cannot have linear phase FIR filters for QMF condition except for trivial 2 tap filter

  $\longrightarrow$ amplitude distortion

- Well known filters

$$H_0(\omega) = A(\omega) \qquad G_0(\omega) = B(\omega)$$

$$H_1(\omega) = e^{j\omega} B(\omega + \pi)$$

$$G_1(\omega) = e^{-j\omega} A(\omega + \pi)$$

$$a(n) = [1, 2, 1]$$

$$b(n) = [-1, 2, 6, 2, -1]$$

$\longrightarrow$ simple to implement

proposed by LeGall

Filter Design:

* Smith and Barnwell

$h(0) = 0.03489$

$h(1) = -0.0109$

$h(2) = -0.0628$

$h(3) = 0.2239$

$h(4) = 0.55685$

$h(5) = 0.35797$

$h(6) = -0.0239$

$h(7) = -0.0759$

Bit Allocation in Subband Coding:

R = Average # of bits per sample

R_K = Average # of bits per sample of subband K

M = # of subbands

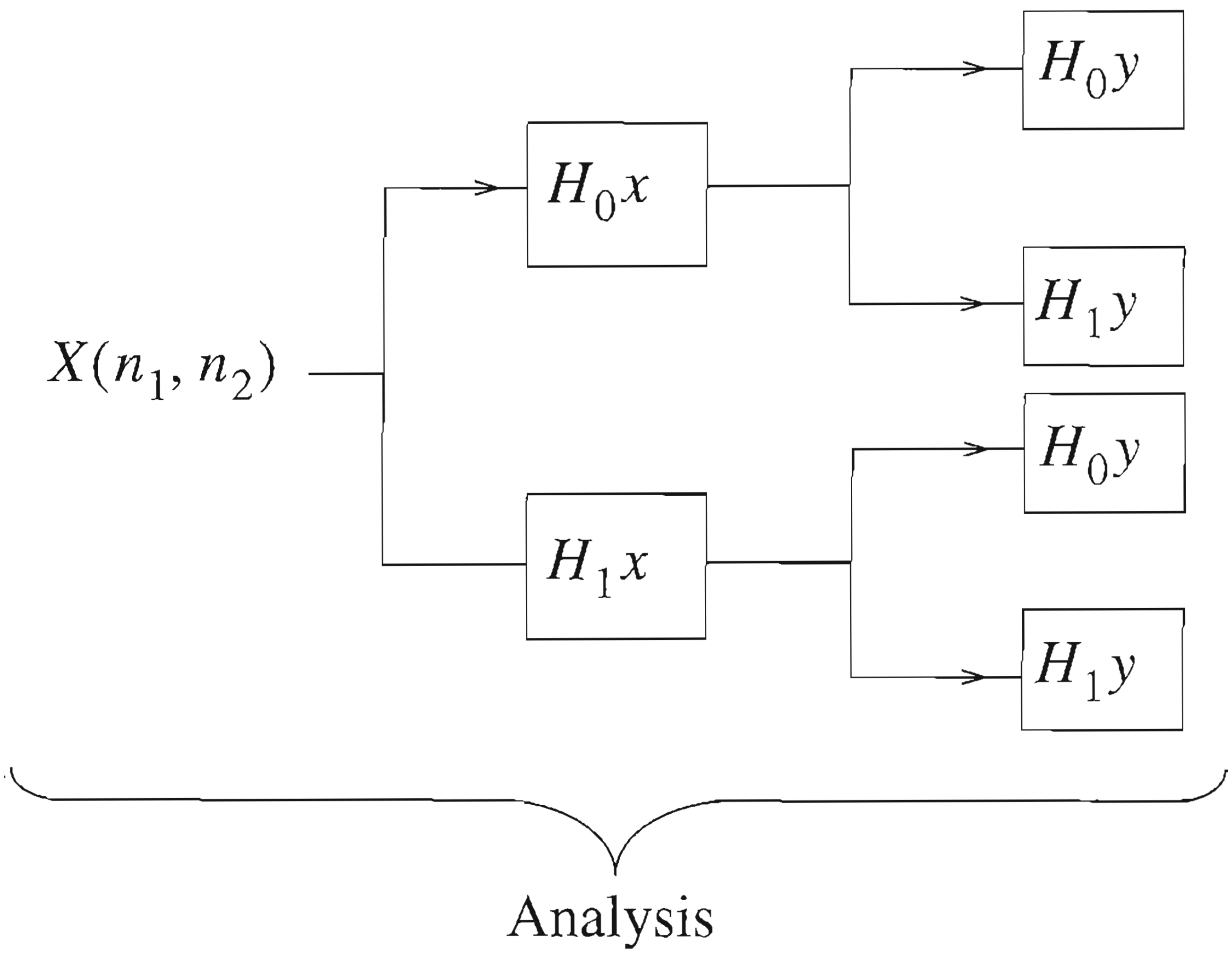
σ_K^2 = variance of coefficients in subband K:

$$R_K = R + \frac{1}{2}\log_2 \frac{\sigma_K^2}{\prod_{K=1}^{M} (\sigma_K^2)^{\frac{1}{M}}}$$

2D Subband Coding

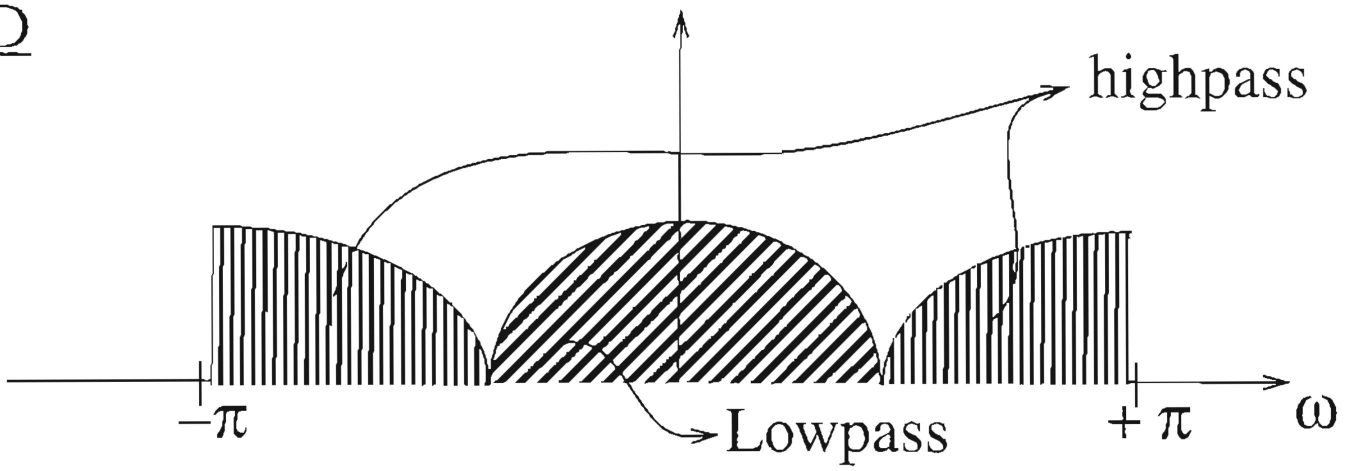
- Separable -----> Easy to implement
- Nonseparable

Separable subband Coding:

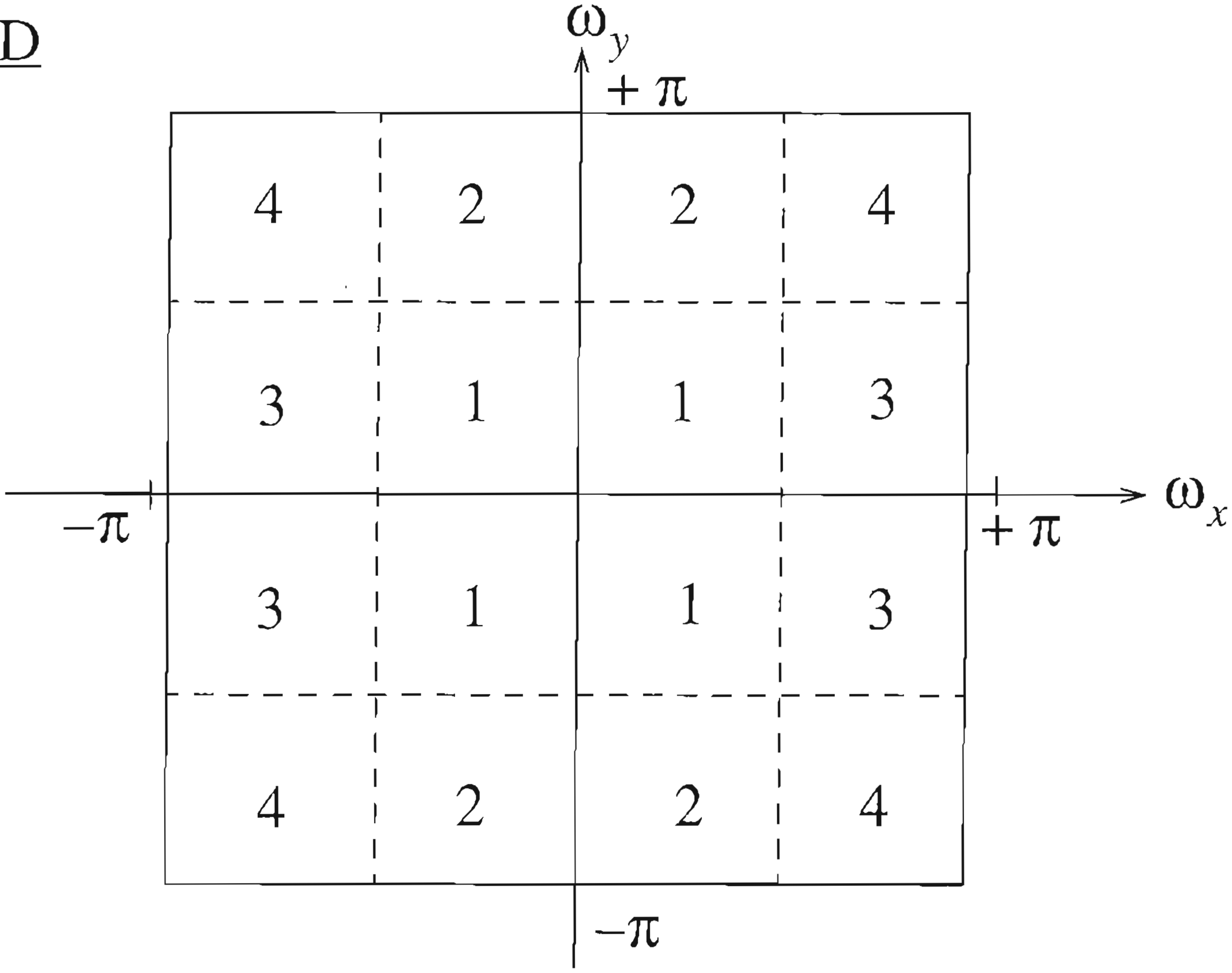


FREQUENCY DOMAIN

1D



2D

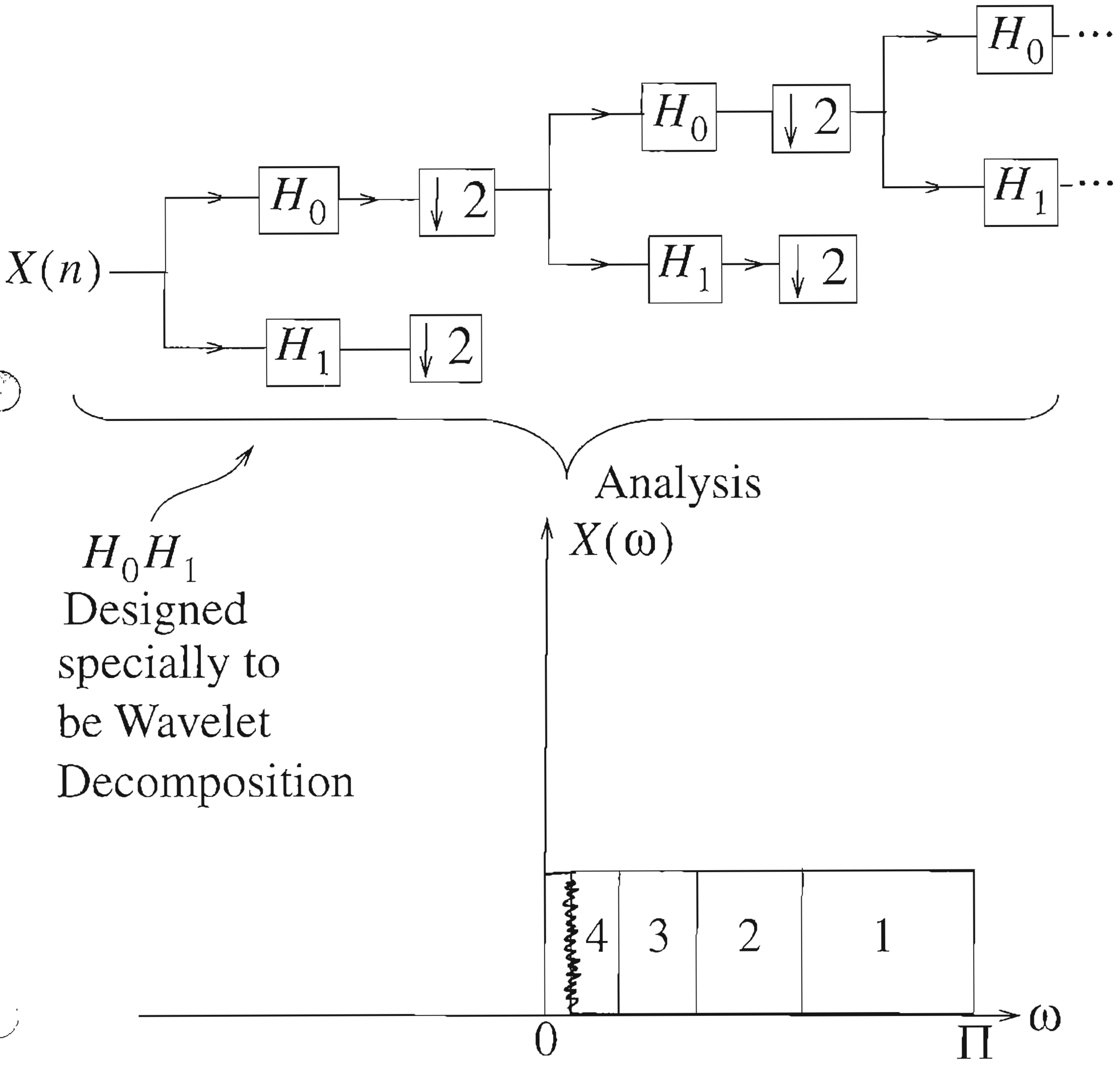


$1 = L_xL_y \qquad 3 = H_xL_y$

$2 = L_xH_y \qquad 4 = H_xH_y$

Wavelets

- A special kind of Subband Transform
- Historically developed independent of subband coding



Famous Wavelet Filters

- Daubechies
- Haar
- Coiflet

4 Tap Daubechies Low Pass

$$h(0) = 0.4829$$

$$h(1) = 0.8365$$

$$h(2) = 0.22414$$

$$h(3) = -0.1294$$