Pyramid Coding and Subband Coding

- **Basic Idea:** Successive lowpass filtering and subsampling.

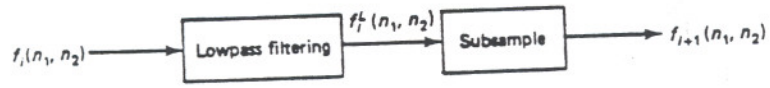


Figure 10.33 Process of generating the $i + 1$th-level image $f_{i+1}(n_1, n_2)$ from the ith-level image $f_i(n_1, n_2)$ in Gaussian pyramid image representation.

- 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

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.

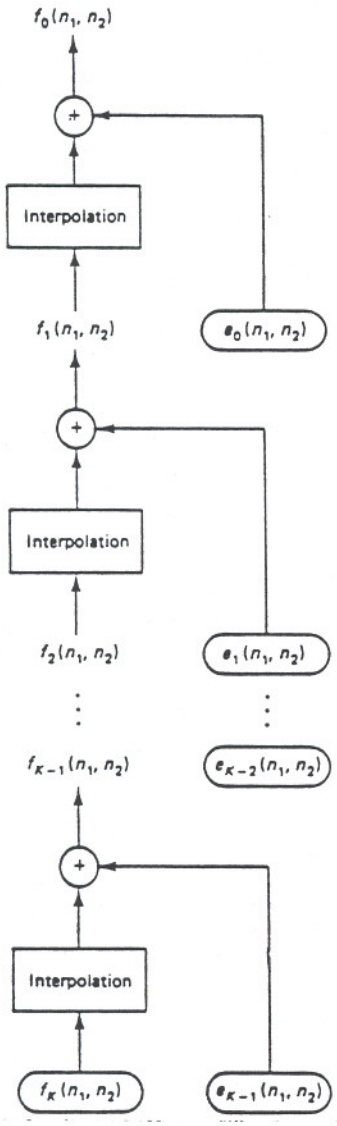


Figure 10.37 Laplacian pyramid generation. The base image $f_0(n_1, n_2)$ can be reconstructed from $e_i(n_1, n_2)$ for $0 \le i \le K - 1$ and $f_K(n_1, n_2)$.

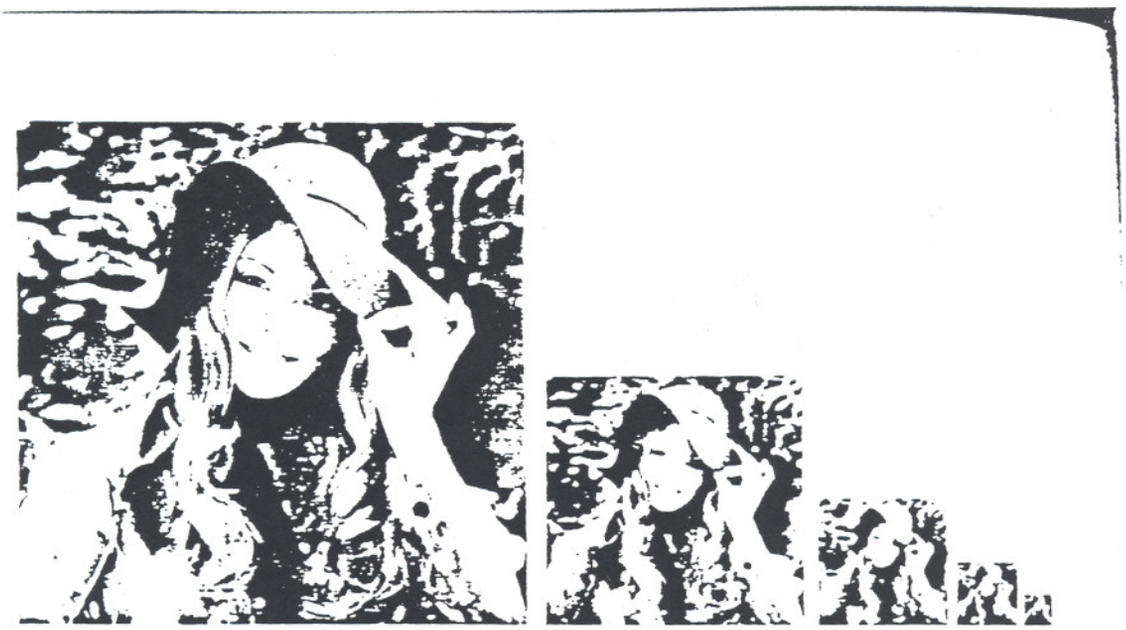


Figure 10.36 Example of the Gaussian pyramid representation for image of 513 × 513 pixels with $K = 4$.

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.

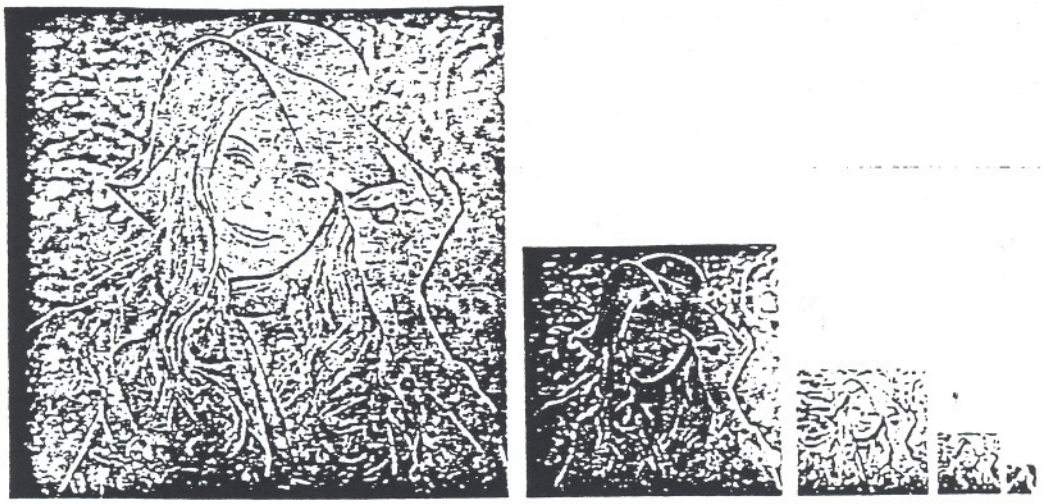


Figure 10.38 Example of the Laplacian pyramid image representation with $K = 4$. The original image used is the 513 × 513-pixel image $f_0(n_1, n_2)$ in Figure 10.36. $e_i(n_1, n_2)$ for $0 \le i \le 3$ and $f_4(n_1, n_2)$.

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 $\frac{1}{4}$ 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 $\frac{1}{4}$ 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.

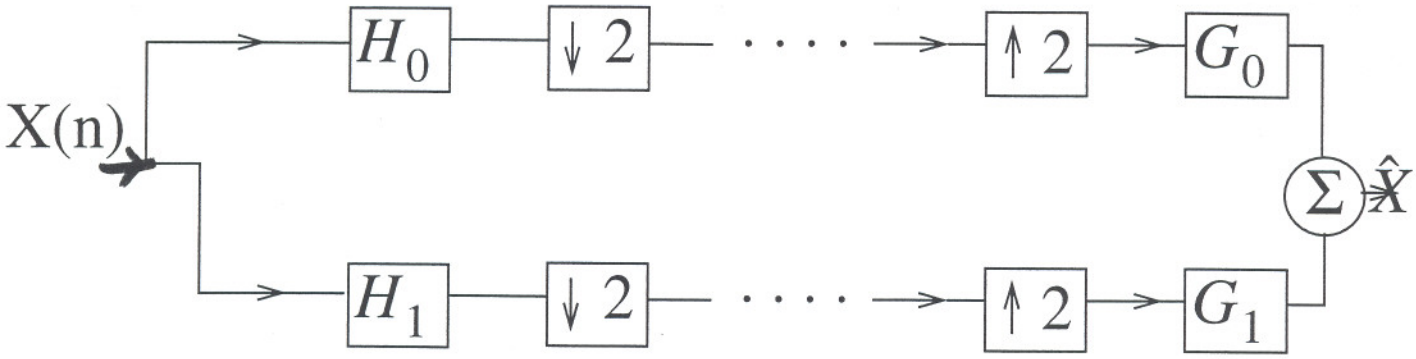
In addition to image coding, the Laplacian pyramid can also be used in other applications. For example, as we discussed above, the result of repetitive interpolation of $e_i(n_1, n_2)$ such that its size is the same as that of $f_0(n_1, n_2)$ can be viewed as approximately the result of filtering $f_0(n_1, n_2)$ 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

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:

$$H_0(\omega) = G_0(-\omega) = F(\omega)$$

$$H_1(\omega) = G_1(-\omega) = e^{j\omega}F(-\omega+\pi)$$

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

IMPOSE: $|F(\omega)|^2 + |F(\omega+\pi)|^2 = 2$

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

Filter Design:

- QMF filters:

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

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

- Johnston's filter coefficients

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

$$\longrightarrow \text{ symetric } \longrightarrow \text{ NPR}$$

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

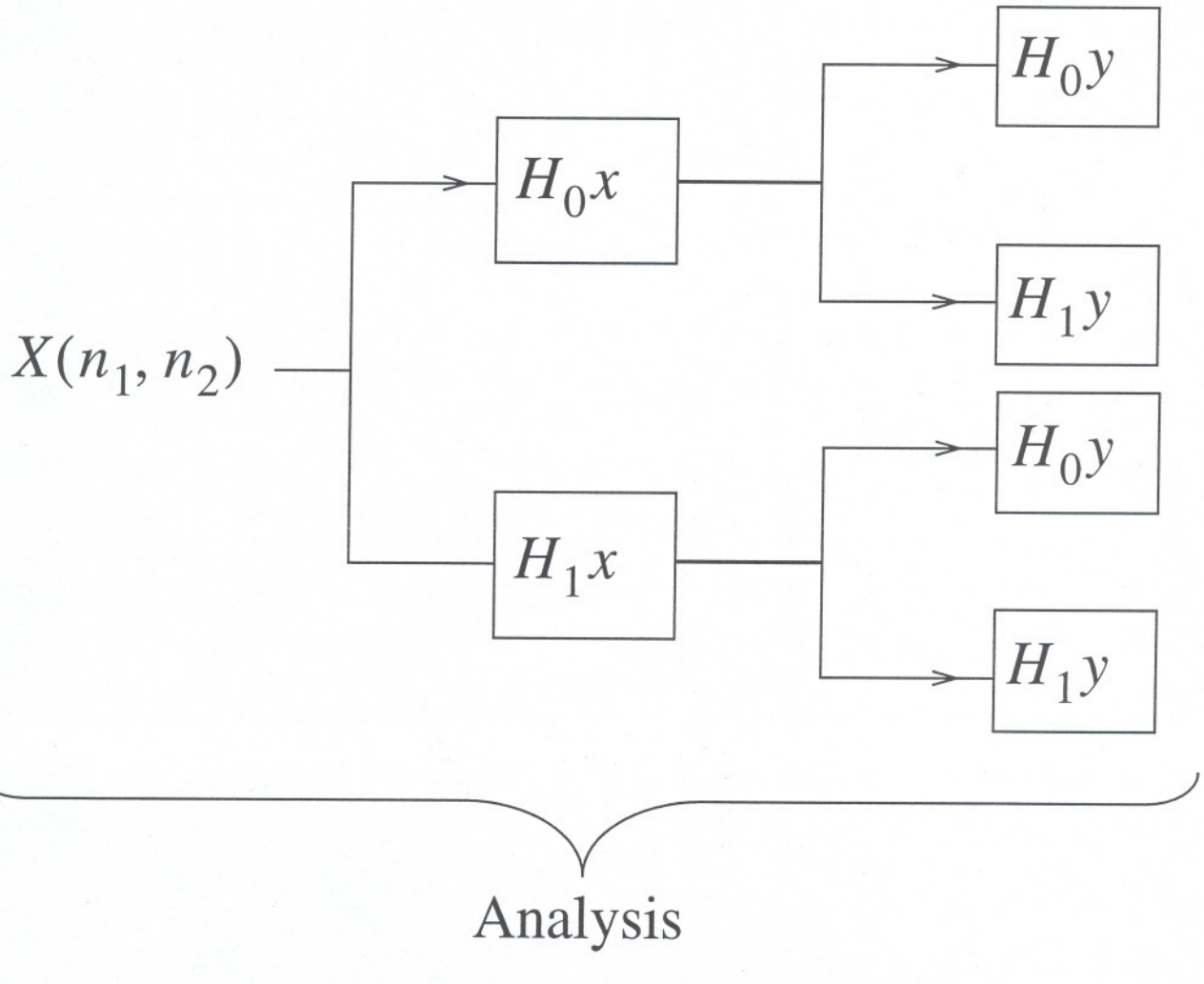
σ_K^2 = variance of coefficients in subband K:

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

2D Subband Coding

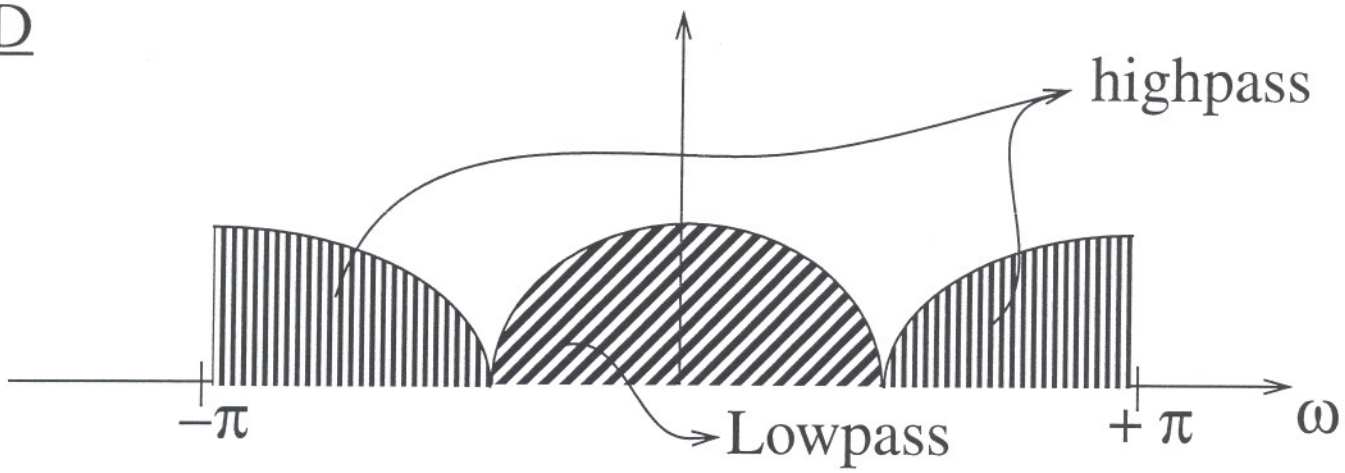
- Separable -----> Easy to implement
- Nonseparable

Separable subband Coding:

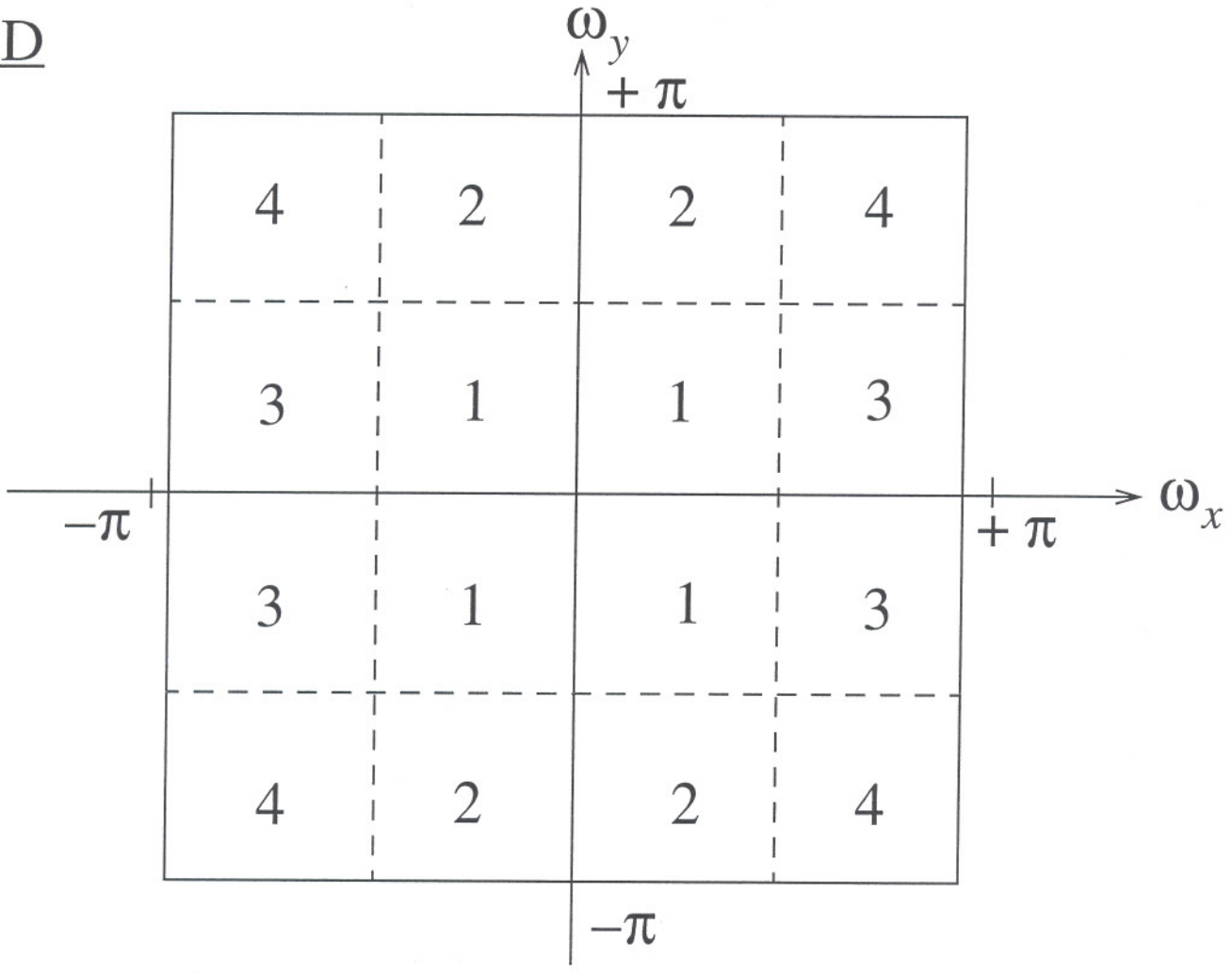


FREQUENCY DOMAIN

1D



2D

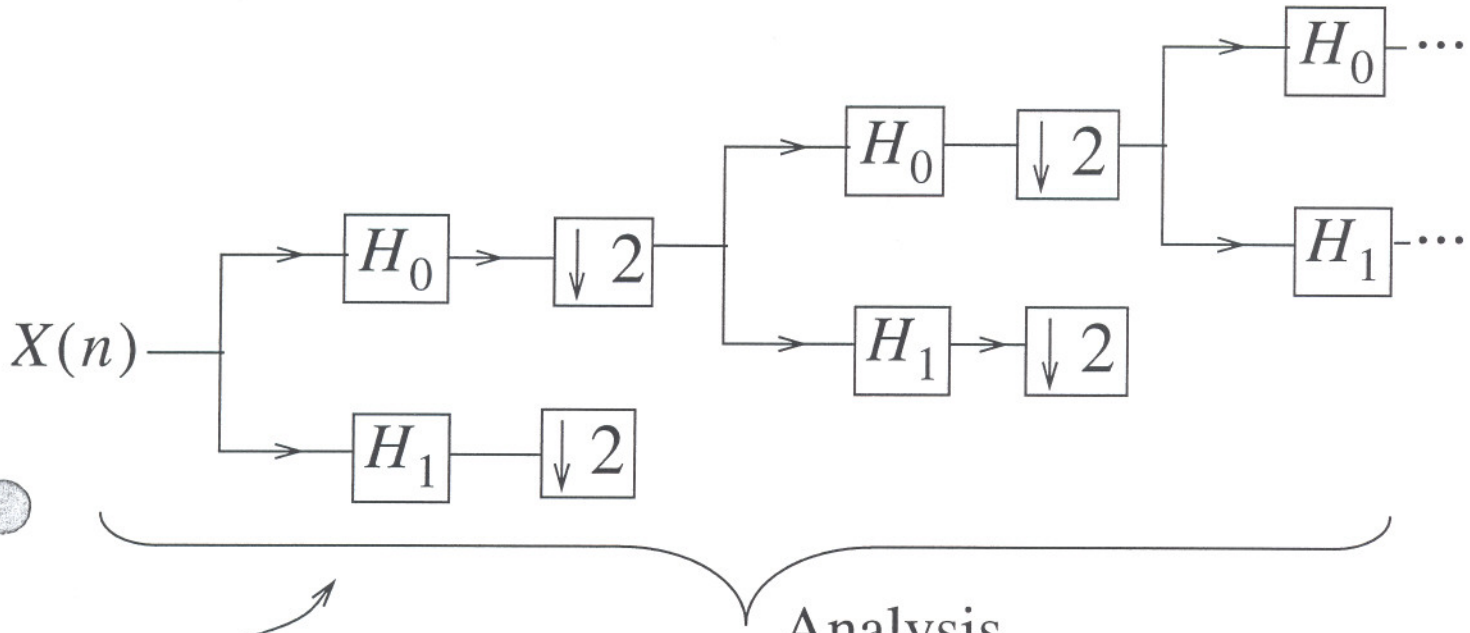


$1 = L_x L_y$ $\quad$ $3 = H_x L_y$

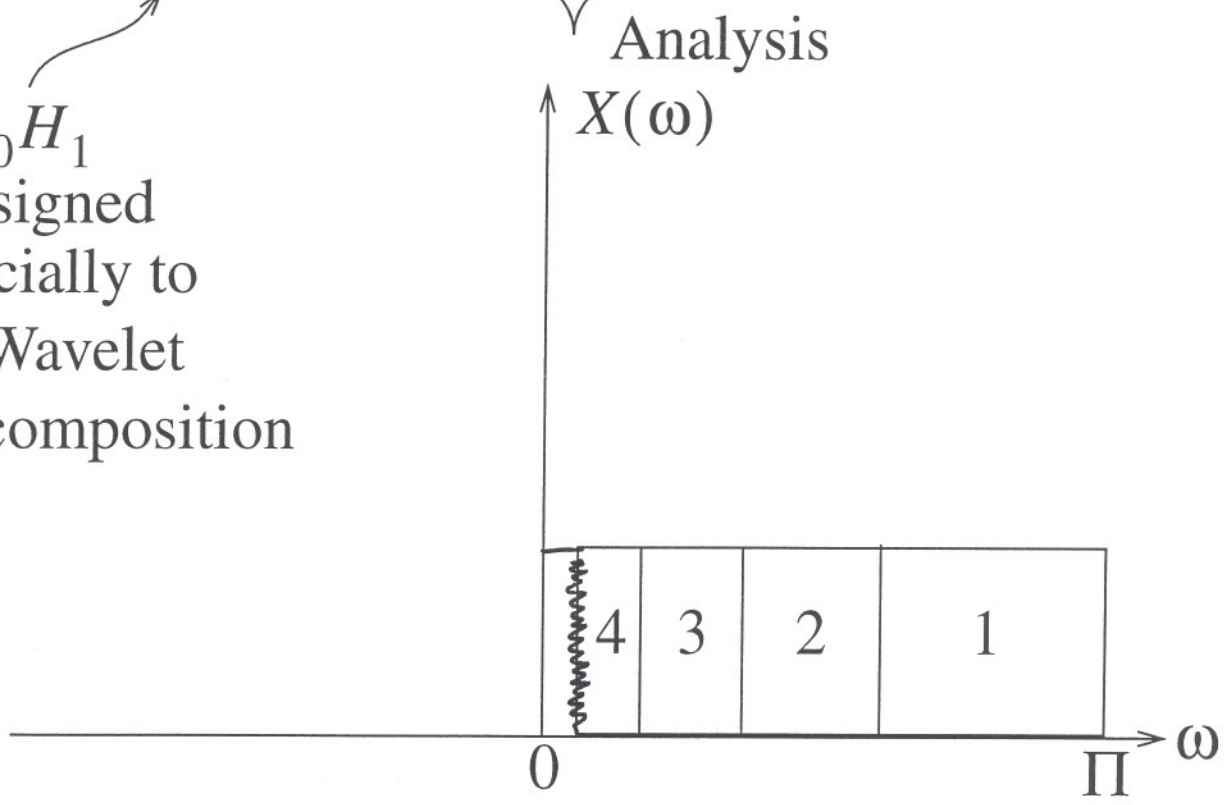
$2 = L_x H_y$ $\quad$ $4 = H_x H_y$

Wavelets

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



$H_0 H_1$
Designed specially to be Wavelet Decomposition



Famous Wavelet Filters

- Daubechies
- Haar
- Coiflet

4 Tap Daubechies Low Pass

$$h(0) = 0.4829$$

$$h(1) = 0.8365$$

$$h(2) = 0.22414$$

$$h(3) = -0.1294$$

Fractal Compression

- Founders: Mandelbrot and Barnsley.
- Basic idea: fixed point transformation
- X_0 is fixed point of function f if

$$f(X_0) = X_0$$

- Example: Transformation ax + b has a fixed point X_0 given by:

$$X_0 = aX_0 + b$$

- To transmit X_0, send a , b

Then iterate:

$$X_0^{(n+1)} = aX_0^{(n)} + b$$

will converge regardless of initial guess.

Image Compression

- Think of Image I as array of numbers
- Find a function f such that

$$f(I) = I$$

- If # of bits to transmit f is smaller than I, achieve Compression
- In practice, hard to come up with one transformation f, for the whole image.
- Divide up the image into domain and domain and Range blocks

Image Compression

- Main idea:

  - Divide up image into M x M "Range" blocks

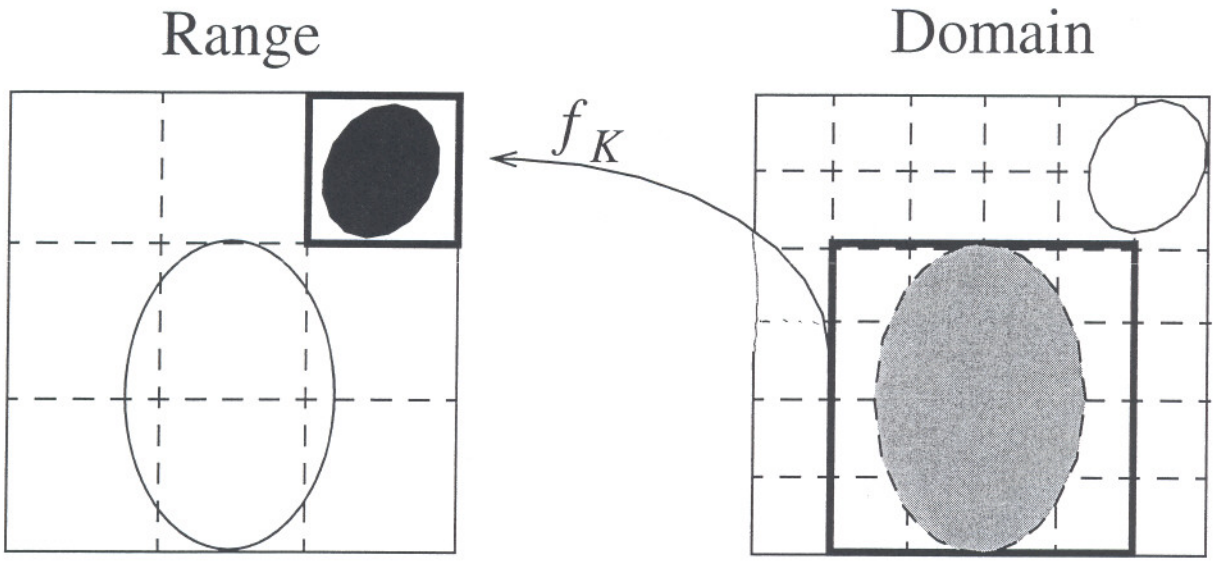
  - For each Range block find another 2 M x 2 M "Domain block" from the same image such that for some transformation f_K we get $f_K(D_K) = R_K$

    D_K = Domain block k

    R_K = Range block k

- First publicly discussed by Jacquin in 1989 thesis + 1992 paper

- Works well if there is self similarity in image.



- What should f_K do?

  - change size of domain block
  - change orientation of domain block
  - change intensity of pixels

- f_K consists of

  - geometric transformation : g_K
  - massic transformation : m_K

- g_K : displacement + size + intensity

- m_K : orientation + ~~intensity~~

Transformations:

- g_K : displaces + adjusts intensity

  $\longrightarrow$ easy

- m_K : $m_K(t_{ij}) = i(\alpha_K t_{ij} + \Delta_K)$

  i can be

  * Rotation by 90, 180, -90
  * Reflection about
    horizontal, vertical, diagonal
  * identity map

- Finding transformations is
  compute intensive

- Search through all domain
  blocks + all transformations to
  find "BEST" one

- Encoding more time than decoding

- If Image is divided $\dot{m}t_o$

  N Range blocks $\longrightarrow$ N

  transformations $f_K \qquad k = 1, \ldots N$

  are its representation.

$$f = \bigcup_k f_k$$

$$\hat{I} = f(\hat{I})$$

$\hat{I}$ is approximation to I.

- Collage theorem guarantees convergence to $\hat{I}$ using <u>any</u> arbitrary initial guess for image.

FIGURE 13.11 **The first six iterations of the fractal decoding process.**

VECTOR QUANTIZATION

- Let $\vec{f}$ denote N dimensional vectors constisitng of N real valued, continuous amplitude scalars.
- **Basic Idea:** Map $\vec{f}$ into L possible N dimensional reconstruction vectors $\vec{r}_i$ for $1 \leq i \leq L$.

$$\hat{\vec{f}} = VQ(\hat{f}) = \vec{r}_i \qquad \vec{f} \in C_i$$

- Define a distortion measure:

$$\begin{aligned} D &= E[(\hat{\vec{f}} - \vec{f})^T(\hat{\vec{f}} - \vec{f})] \\ &= \sum_{i=1}^{L} \int_{\vec{f}_0 \in C_i} (\vec{r}_i - \hat{\vec{f}}_0) d\vec{f}_0 \end{aligned}$$

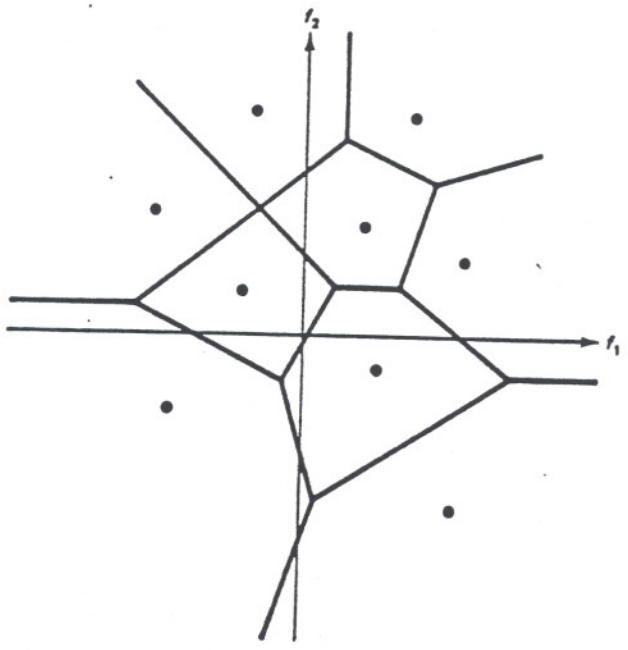


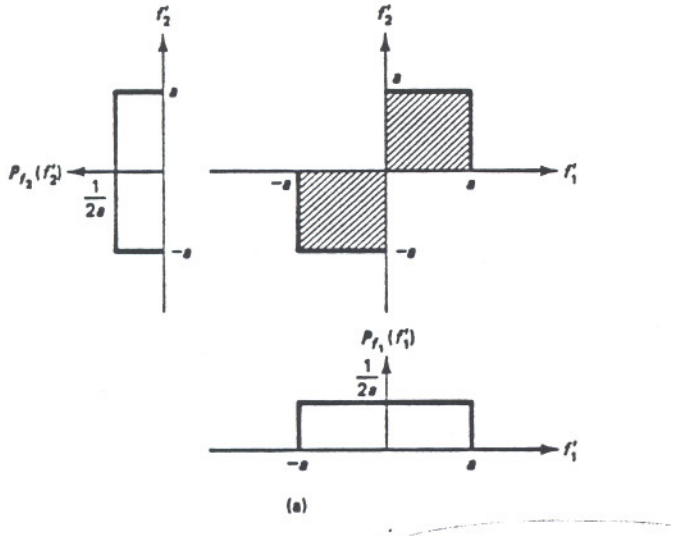
Figure 10.8 Example of vector quantization. The number of scalars in the vector is 2, and the number of reconstruction levels is 9.

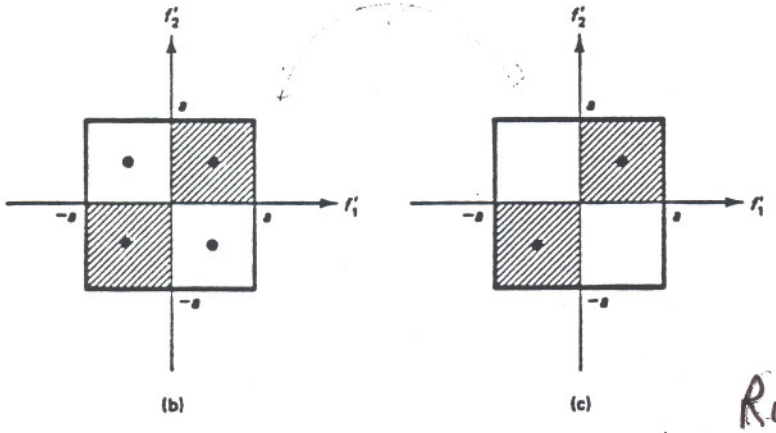
Properties of Vector Quantization

- Removes linear dependency between random variables.
- Removes nonlinear dependency between random variables.
- Explits increase in dimensionality.
- Allows us to code a scalar with less than one bit.
- Computational and storage requirements are far greater than scalar quantization.

VQ Removes Linear Dependency

- Linear transformation can decorrelate linearly dependent (correlated) random variables.





Some distortion
But reduce # of reconstruction level
(linear Transformation)
Rotate f_1, f_2 to get g_1, g_2 uncorrelated.

Figure 10.9 Illustration that vector quantization can exploit linear dependence of scalars in the vector. (a) Probability density function $p_{f_1 f_2}(f_1', f_2')$; (b) reconstruction levels (filled-in dots) in scalar quantization; (c) reconstruction levels (filled-in dots) in vector quantization.

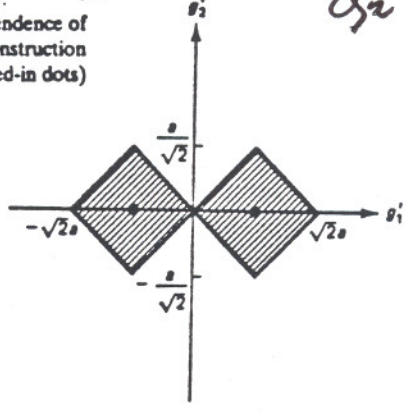
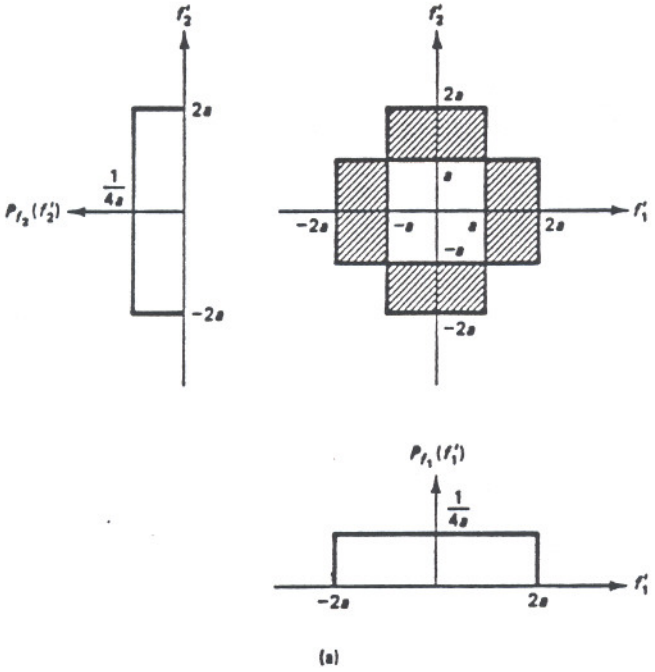


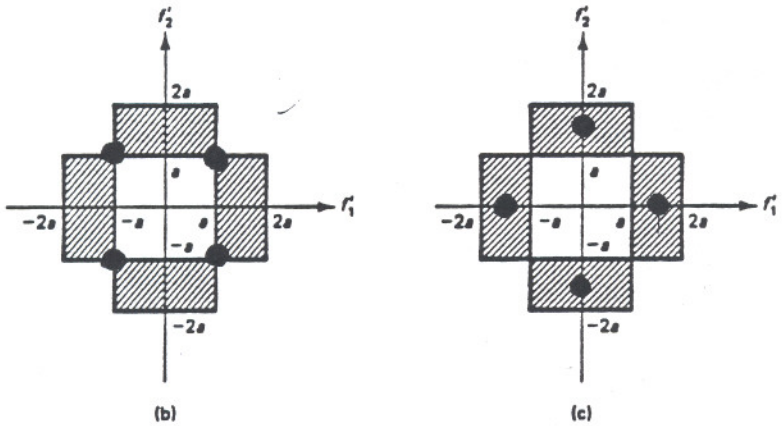
Figure 10.10 Result of eliminating linear dependence of the two scalars f_1 and f_2 in Figure 10.9 by linear transformation of f_1 and f_2.

VQ Removes Nonlinear Dependency

- Nonlinear dependence cannot be eliminated by a linear operator.



distortion $a^2/2$

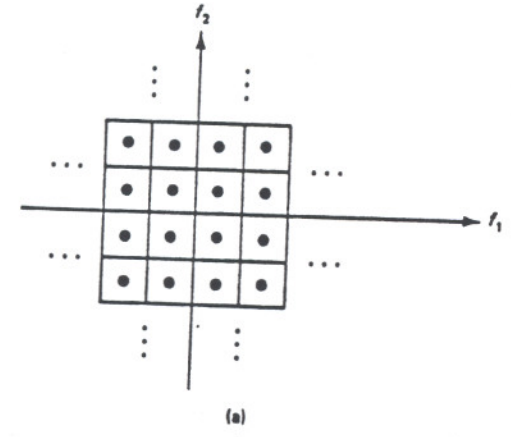


distortion $\frac{5a^2}{12}$

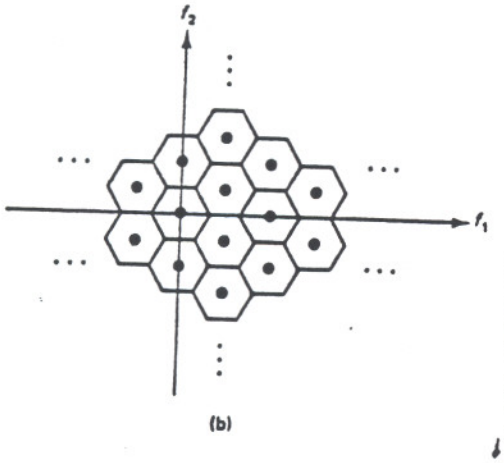
Figure 10.11 Illustration that vector quantization can exploit nonlinear dependence of scalars in the vector. (a) Probability density function $p_{f_1,f_2}(f_1', f_2')$; (b) reconstruction levels (solid dots) in scalar quantization; (c) reconstruction levels (solid dots) in vector quantization.

VQ Exploits the Increase in Dimensionality

- The mean square error due to VQ is approximately less than 4 percent than scalar quantization. with same # of reconstruction levels.



(a)



(b)

Figure 10.13 Illustration that vector quantization can exploit the dimensionality increase. In this case, the mean square error due to vector quantization is approximately 4% less than that due to scalar quantization. (a) Scalar quantization of f_1 and f_2; (b) vector quantization of f_1 and f_2.

# of Recon levels is 2% lower than Scalar Quantizer with same MSE.

⇒ # of bits per scalar with VQ < 1

look at 10.9

scalar 1 bit per scalar

VQ: ½ bit per scalar

Codebook Design Algorithms

- K-means algorithm.
- Tree codebooks and binary search.
- Nearest neighbor.

Codebook Design via K-means

- Exploit the following two necessary conditions for the optimal solution:
  - For a vector $\vec{f}$ to be quantized to one of the reconstruction levels, the optimal quantizer must choose the reconstruction level $\vec{r}_i$ which has the samllest distortion between $\vec{f}$ and $\vec{r}_i$.
  - Each reconstruction level $\vec{r}_i$ must minimize the average distortion D in C_i.

    $$Minimize \quad E[d(\vec{f}, \vec{r}_i) \mid \vec{f} \in C_i] \quad w.r.t. \quad \vec{r}_i$$

- Find $\vec{r}_i$ and C_i iteratively $\rightarrow$ Problem: local versus global minimum $\rightarrow$ initial guess important.

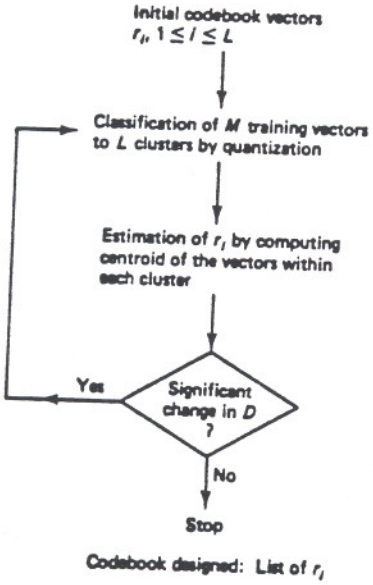


Figure 10.14 Codebook design by the K-means algorithm for vector quantization.

Complexity of K-means

- M training vectors, L codewords, N dimensional, R bits per scalar.
- Complexity of Codebook design:
  - ML evaluation of distortion measure for each iteration.
  - $MLN = NM2^{NR}$ additions and mults per iteration.
  - Example: N= 10, R=2 , M = 10L results in 100 trillion operation per iteration.
  - Storage: MN for training vectors, LN for reconstruction levels $\rightarrow$ $(M + 2^{NR})N$.
- Complexity of operation at the ~~receiver~~.
  - Storage of reconstruction levels: $N2^{NR}$. If $N = 10$ and $R = 2$, storage is 10 million.
  - Number of artithemetic operations $N2^{NR}$. If $N = 10$ and $R = 2$, 10 million operations per look up.

Tree Codebook and Binary Search

- Full search is responsible for exponential growth of the number of operations at the ~~receiver~~ → Tx mtr. Tree codebook.
- Let L be a power of 2.
- Basic operation of tree codebook design:
  - Use K-means to divide the N dimensional space of $\vec{f}$ into two regions.
  - Divide each of the two regions into two more regions using the K-means algorithm.
  - Repeat step 2 until there are L reconstruction levels.

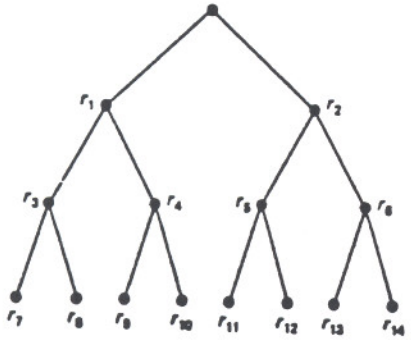


Figure 10.15 Example of a tree codebook.

243

M = # of Training vectors
L = # of codewords
N = Dimension

Complexity of Tree Codebook

- Design complexity:
  - Number of arithmetic operations per iteration is $2NM \log_2 L$. For $N = 10$ and $R = 2$, the reduction factor compared to the full search is 26,000.
  - Storage: approximately the same as full search algorithm.
- Operation complexity at ~~receiver~~:
  - Number of arithmetic operations Is $2N^2R$. For $N = 10$ and $R = 2$, the reduction factor compared to the full search is 26,000.
  - Storage: The codebook must store all the intermediate reconstruction levels as well as the final reconstruction levels. → Twice as much storage needed as full search.
- Distortion of full search is slightly smaller than that of tree search.

2-4

Nearest Neighbor Design Algorithm

- Initially proposed by Equitz.
- computational complexity grows linearly with the training set.
- Find the 2 vectors closest to each other, merge them into another vector equal to their mean, repeat this process until the number of vectors is L.
- Main efficiency is achieved by partitioning the training data into a K-D tree $\rightarrow$ multiple merges at each iteration.

Variations of VQ

- Multistage VQ reduces storage and search time.
  1. First stage a low rate VQ.
  2. Generate error by subtracting the codeword from the original.
  3. Code the error by a different VQ.
  4. Repeat steps 2 and 3.

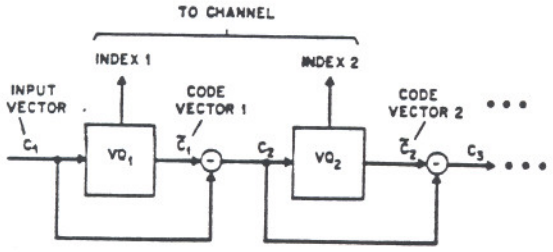


Fig. 5.5.2 Multistage Vector Quantization. At each state an error vector is computed, which is then used as the input to the next stage of VQ. The decoder merely computes a summation of the code vectors corresponding to the received indices.

- Parameter extraction techniques:
  - mean and variance of each input vector are computed and sent separately.
  - mean and variance might be coded with DPCM.

Variations of VQ (cont'd)

- Block classification:
  - Divide the blocks into several classes according to spatial activity.
  - Design a codebook for each class.
  - Overhead on transmitting the codebook is large.
- Combine prediction techniques with VQ:
  - Coded quantity is the prediction error rather than intensity values.
- VQ of color images exploits the correlation between color components.
- Typical rates: .1 to .5 bits per pixel for 4×4 pixels as vectors.

local measure, such as local image contrast. If the local measure used can be obtained from previously coded pixel intensities, then it does not have to be transmitted. If the local measure used is obtained directly from $f(n_1, n_2)$, it must be transmitted, since the receiver does not have access to the original image. Adaptive coding certainly adds complexity to an image coder, but can often significantly improve its performance.

PCM systems do not exploit the statistical dependence of neighborhood pixel intensities. One way to exploit the statistical dependence is to use methods such as DM, DPCM, and two-channel coders, where the difference between $f(n_1, n_2)$ and a prediction of $f(n_1, n_2)$ is coded. An alternate way is to use vector quantization. As we discussed in Section 10.2, vector quantization can exploit the statistical dependence of the parameters coded. Vector quantization has been considered in coding the waveform $f(n_1, n_2)$. The blocks used consist of neighborhood pixel intensities of a small size, typically 2×2, 3×3, and 4×4. Vector quantization has been primarily applied in low bit rate (below 1 bit/pixel) applications, since the computational and storage costs increase rapidly with the block size and the bit rate. Intelligible images can be reconstructed with some sacrifice in quality at bit rates below 1 bit/pixel, which is not possible with DM, DPCM, or two-channel coders with scalar quantization and uniform-length codeword assignment. For waveform coders, vector quantization is an effective way to code an image at a bit rate lower than 1 bit/pixel. Figure 10.40 illustrates the performance of vector quantization when applied to coding the waveform $f(n_1, n_2)$. Figure 10.40(a) shows an original image of 512×512 pixels. Figure 10.40(b) shows the

(a) (b)

Figure 10.40 Example of an image coded by vector quantization. Courtesy of William Equitz. (a) Original image of 512×512 pixels; (b) coded image by vector quantization at $\frac{1}{2}$ bit/pixel. The block size used is 4×4 pixels and the codebook is designed by using a variation of the K-means algorithm. NMSE = 2.7%, SNR = 15.7 dB.

342