Objectives of Image Coding

- Representation of an image with acceptable quality, using as small a number of bits as possible

Applications:

- Reduction of channel bandwidth for image transmission
- Reduction of required storage

Objectives of Image Coding cont.

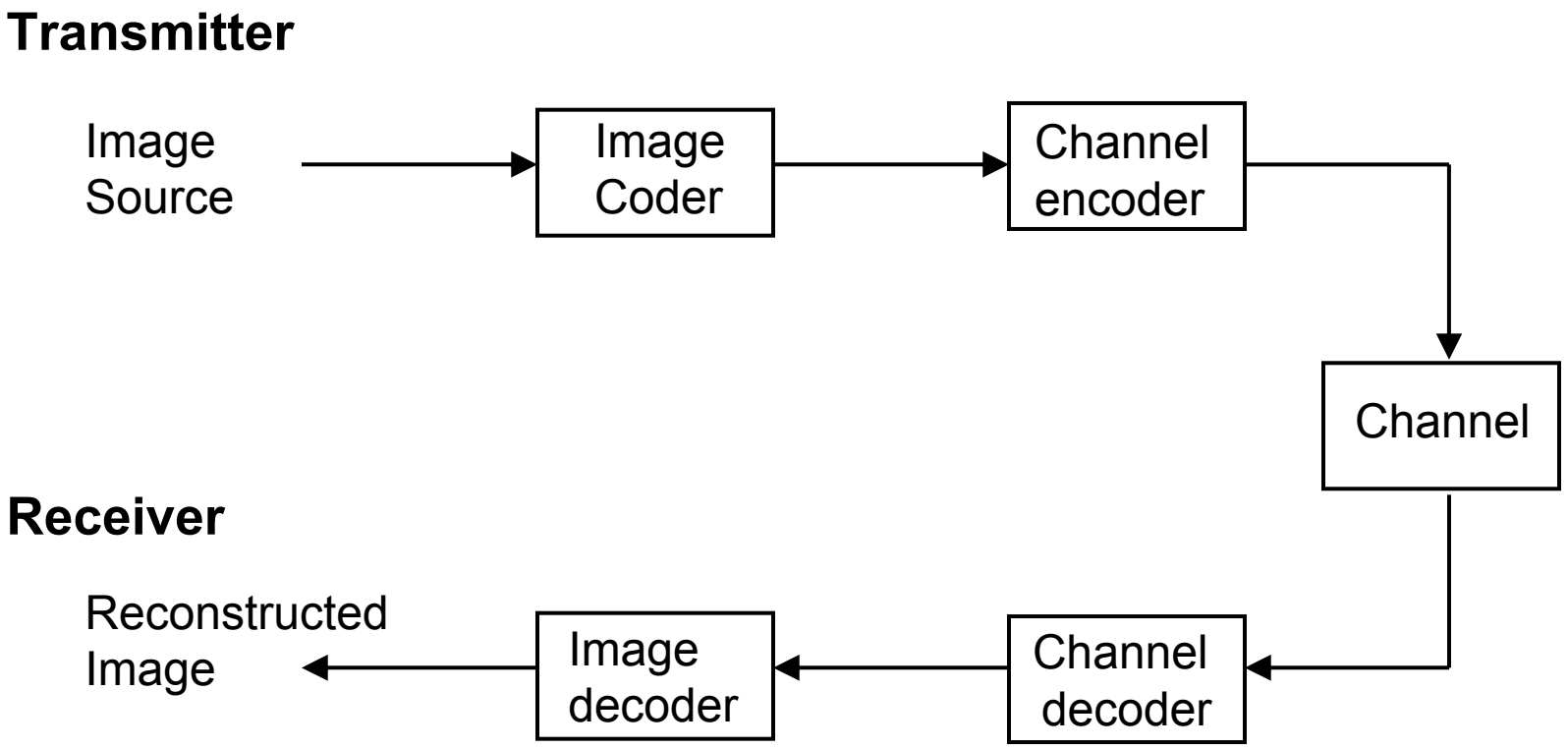


Figure 10.1 Typical environment for image coding.

Issues in Image Coding

1. What to code?
   a. Image density
   b. Image transform coefficients
   c. Image model parameters

2. How to assign reconstruction levels
   a. Uniform spacing between reconstruction levels
   b. Non-uniform spacing between reconstruction levels

3. Bit assignment
   a. Equal-length bit assignment to each reconstruction level
   b. Unequal-length bit assignment to each reconstruction level

Issues in Image Coding cont.

Figure 10.2 Three major components in image coding.

Methods of Reconstruction Level Assignments

Assumptions:

- Image intensity is to be coded
- Equal-length bit assignment

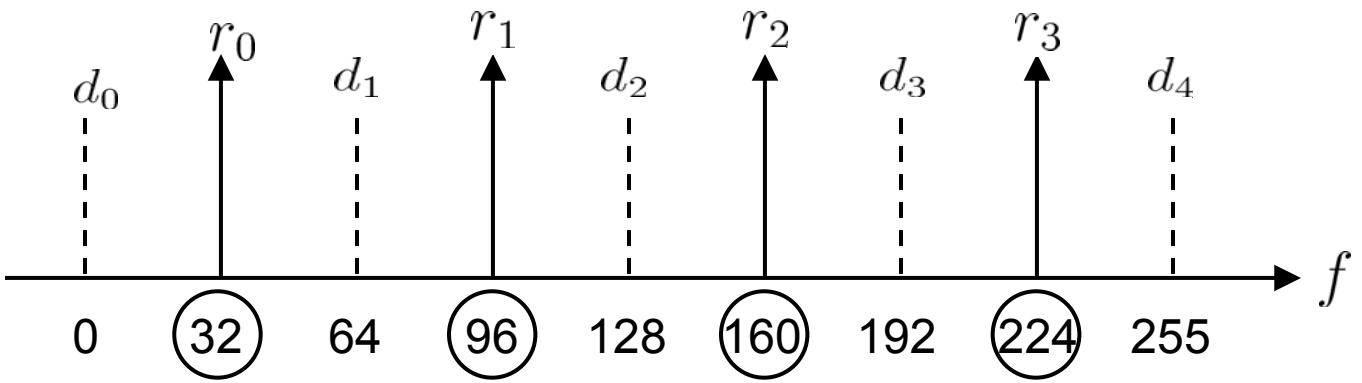
Scalar Case

1. Equal spacing of reconstruction levels (Uniform Quantization)

   (Ex): Image intensity *f:* *0 ~ 255*

Methods of Reconstruction Level Assignments cont.

Number of reconstruction levels: 4 (2 bits for equal bit assignment)



Methods of Reconstruction Level Assignments cont.

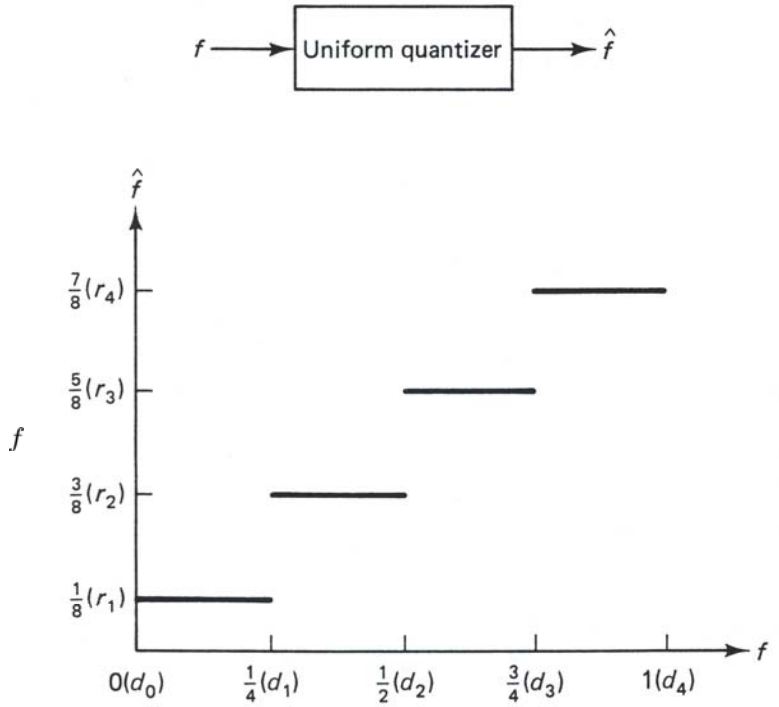


Figure 10.3
Example of uniform quantizer. The number of reconstruction levels is 4, f is assumed to be between 0 and 1, and $\hat{f}$ is the result of quantizing f The reconstruction levels and decision boundaries are denoted by r_i and d_i, respectively.

Scalar Case (cont.)

2. Spacing based on some error criterion

r_i: reconstruction levels (32, 96, 160, 224)

d_i: decision boundaries (0, 64, 128, 192, 256)

Optimally choose r_i and d_i.

To do this, assume $f_{min} \leq f \leq f_{max}$

J $\triangleq$ the number of reconstruction levels

$p(f)$: probability density function for *f*

Minimize

r_i, d_i, : $\epsilon = E[(f-\hat{f})^2] = \int_{f=f_{min}}^{f_{max}} (f-\hat{f})^2 \cdot p(f) \cdot df$ = $\sum_{j=0}^{J-1} \int_{d_j}^{d_{j+1}} (f-r_j)^2 \cdot p(f) \cdot df$

$\Rightarrow \frac{\partial \epsilon}{\partial r_j} = 0 \Rightarrow r_j = 2d_j - r_{j-1}$

$\frac{\partial \epsilon}{\partial d_j} = 0 \qquad r_j = \frac{\int_{d_j}^{d_{j+1}} f \cdot p(f) \cdot df}{\int_{d_j}^{d_{j+1}} p(f) \cdot df}$ ==> Lloyd-Max Quantizer

These are not simple linear equations.

Scalar Case (cont.)

2. Spacing based on some error criterion

   r_i: reconstruction levels (32, 96, 160, 224)

   d_i: decision boundaries (0, 64, 128, 192, 256)

   Optimally choose r_i and d_i.

   To do this, assume $f_{min} \leq f \leq f_{max}$

   $J \triangleq$ the number of reconstruction levels

   $p(f)$: probability density function for *f*

Scalar Case (cont.)

Minimize

r_i, d_i, :

$$\epsilon = E[(f-\hat{f})^2] = \int_{f=f_{min}}^{f_{max}} (f-\hat{f})^2 \cdot p(f) \cdot df = \sum_{j=0}^{J-1} \int_{d_j}^{d_{j+1}} (f-r_j)^2 \cdot p(f) \cdot df$$

$$\Rightarrow \frac{\partial \epsilon}{\partial r_j} = 0 \Rightarrow r_j = 2d_j - r_{j-1}$$

$$\frac{\partial \epsilon}{\partial d_j} = 0 \qquad r_j = \frac{\int_{d_j}^{d_{j+1}} f \cdot p(f) \cdot df}{\int_{d_j}^{d_{j+1}} p(f) \cdot df}$$

==> Lloyd-Max Quantizer

These are not simple linear equations.

Scalar Case (cont.)
Solution to Optimization Problem

TABLE 10.1 PLACEMENT OF RECONSTRUCTION AND DECISION LEVELS FOR LLOYD-MAX QUANTIZER. FOR UNIFORM PDF, $p_f(f_0)$ IS ASSUMED UNIFORM BETWEEN -1 AND 1. THE GAUSSIAN PDF IS ASSUMED TO HAVE MEAN OF 0 AND VARIANCE OF 1. FOR THE LAPLACIAN PDF, $p_f(f_0) = \frac{\sqrt{2}}{2\sigma} e^{-\frac{\sqrt{2}|f_0|}{\sigma}}$ with $\sigma = 1$.

	Uniform		Gaussian		Laplacian	
Bits	r_i	d_i	r_i	d_i	r_i	d_i
1	−0.5000	−1.0000	−0.7979	−∞	−0.7071	−∞
	0.5000	0.0000	0.7979	0.0000	0.7071	0.0000
		1.0000		∞		∞
2	−0.7500	−1.0000	−1.5104	−∞	−1.8340	−∞
	−0.2500	−0.5000	−0.4528	−0.9816	−0.4198	−1.1269
	0.2500	0.0000	0.4528	0.0000	0.4198	0.0000
	0.7500	0.5000	1.5104	0.9816	1.8340	1.1269
		1.0000		∞		∞
3	−0.8750	−1.0000	−2.1519	−∞	−3.0867	−∞
	−0.6250	−0.7500	−1.3439	−1.7479	−1.6725	−2.3796
	−0.3750	−0.5000	−0.7560	−1.0500	−0.8330	−1.2527
	−0.1250	−0.2500	−0.2451	−0.5005	−0.2334	−0.5332
	0.1250	0.0000	0.2451	0.0000	0.2334	0.0000
	0.3750	0.2500	0.7560	0.5005	0.8330	0.5332
	0.6250	0.5000	1.3439	1.0500	1.6725	1.2527
	0.8750	0.7500	2.1519	1.7479	3.0867	2.3769
		1.0000		∞		∞
4	−0.9375	−1.0000	−2.7326	−∞	−4.4311	−∞
	−0.8125	−0.8750	−2.0690	−2.4008	−3.0169	−3.7240
	−0.6875	−0.7500	−1.6180	−1.8435	−2.1773	−2.5971
	−0.5625	−0.6250	−1.2562	−1.4371	−1.5778	−1.8776
	−0.4375	−0.5000	−0.9423	−1.0993	−1.1110	−1.3444
	−0.3125	−0.3750	−0.6568	−0.7995	−0.7287	−0.9198
	−0.1875	−0.2500	−0.3880	−0.5224	−0.4048	−0.5667
	−0.0625	−0.1250	−0.1284	−0.2582	−0.1240	−0.2664
	0.0625	0.0000	0.1284	0.0000	0.1240	0.0000
	0.1875	0.1250	0.3880	0.2582	0.4048	0.2644
	0.3125	0.2500	0.6568	0.5224	0.7287	0.5667
	0.4375	0.3750	0.9423	0.7995	1.1110	0.9198
	0.5625	0.5000	1.2562	1.0993	1.5778	1.3444
	0.6875	0.6250	1.6180	1.4371	2.1773	1.8776
	0.8125	0.7500	2.0690	1.8435	3.0169	2.5971
	0.9375	0.8750	2.7326	2.4008	4.4311	3.7240
		1.0000		∞		∞

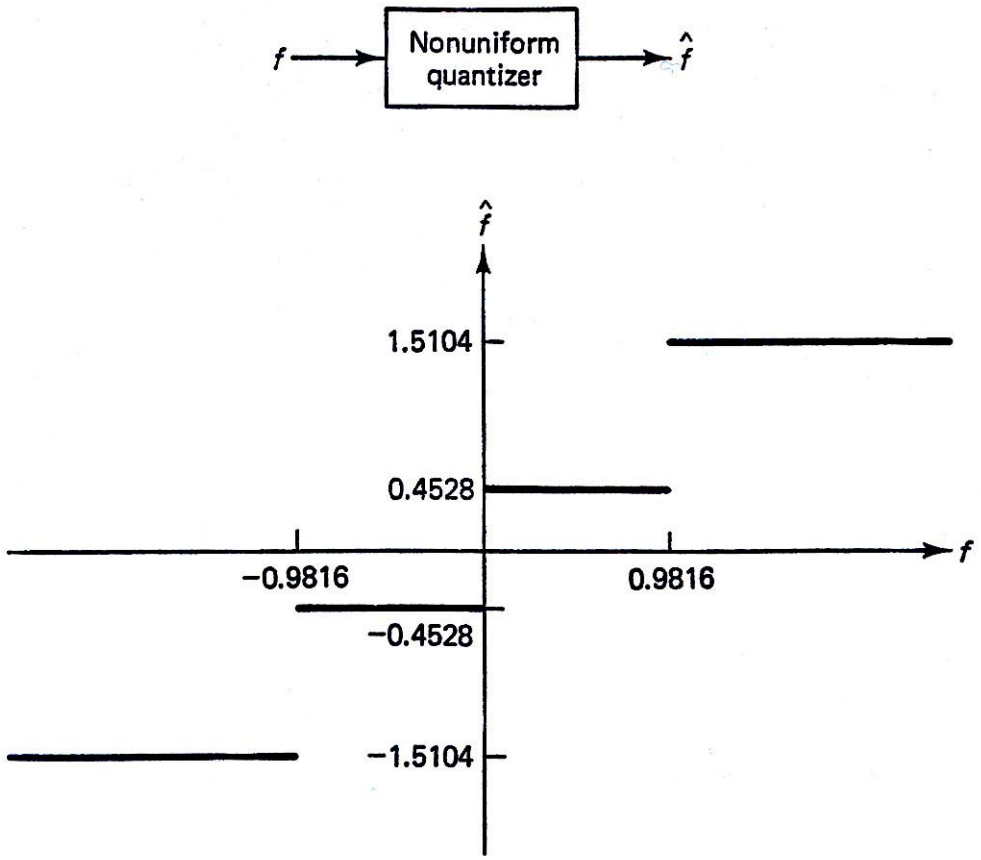


Figure 10.5
Example of a Lloyd-Max quantizer. The number of reconstruction levels is 4, and the probability density function for f is Gaussian with mean of 0 and variance of 1.

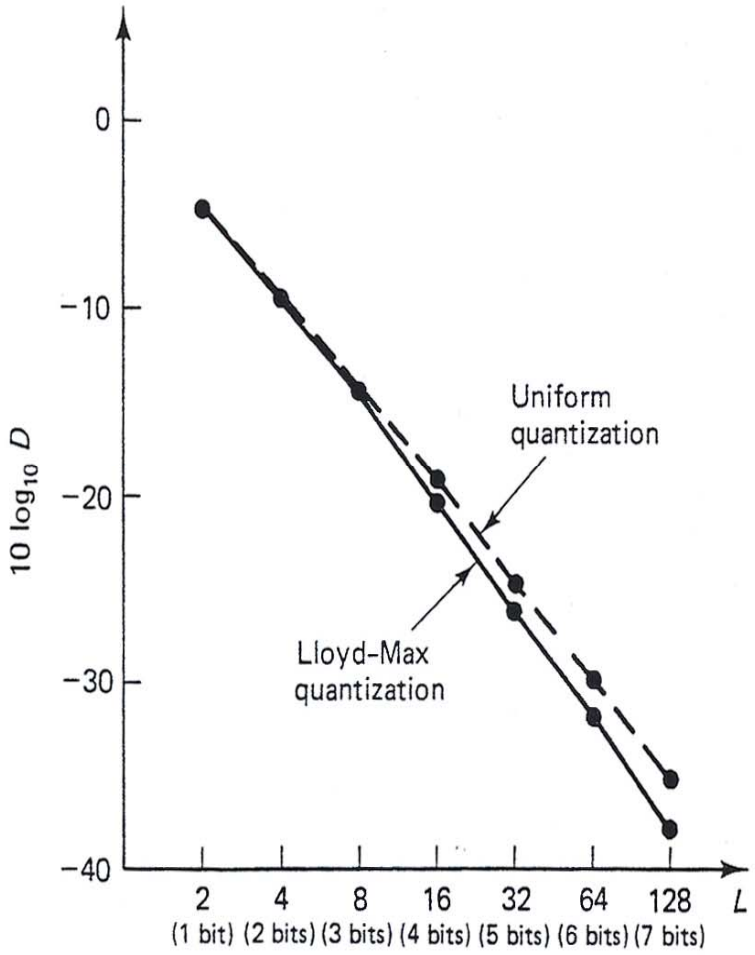


Figure 10.6

Comparison of average distortion $D = E[(\hat{f} - f)^2]$ as a function of L, the number of reconstruction levels, for a uniform quantizer (dotted line) and the Lloyd-Max quantizer (solid line). The vertical axis is 10 $\log_{10}$ D. The probability density function is assumed to be Gaussian with variance of 1.

Scalar Case (cont.)

For some densities, the optimal solution can be viewed as follows:

Figure 10.7 Nonuniform quantization by companding.

P(g) is flat density equally likely between g_{max}, g_{min}

Scalar Case (cont.)

For some densities, the optimal solution can be viewed as follows:

Figure 10.7 Nonuniform quantization by companding.

P(g) is flat density equally likely between g_{max}, g_{min}

Table ? 1-2 Companding quantization by transformation

	Probability Density	Forward Transformation	Inverse Transformation
Gaussian	$p(f) = (2\pi\sigma^3)^{-1/2} exp - \left\{\frac{f^2}{2\sigma^2}\right\}$	$g = \frac{1}{2} erf\left\{\frac{f}{\sqrt{2}\sigma}\right\}$	$\dot{f} = \sqrt{2\sigma} erf^{-1}\{2\dot{g}\}$
Rayleigh	$p(f) = \frac{f}{\sigma^2} exp\left\{-\frac{f^2}{2\sigma^2}\right\}$	$g = \frac{1}{2} - exp\left\{-\frac{f^2}{2\sigma^2}\right\}$	$\dot{f} = [\sqrt{2\sigma^2} ln[1/(\frac{1}{2} - \dot{g})]]^{1/2}$
Lapiacian	$p(f) = \frac{\alpha}{2} exp\{-\alpha \lvert f \rvert\}$	$g = \frac{1}{2}[1 - exp\{-\alpha f\}] \quad f \geq 0$	$\dot{f} = -\frac{1}{\alpha} ln[1 - 2\dot{g}] \quad \dot{g} \geq 0$
	$\alpha = \frac{\sqrt{2}}{\sigma}$	$g = -\frac{1}{2}[1 - exp\{\alpha f\}] \quad f < 0$	$\dot{f} = \frac{1}{\alpha} ln[1 + 2\dot{g}] \quad \dot{g} < 0$

Where $erf(x) = \frac{2}{\sqrt{\pi}} \int_0^d exp(-y^2) dy$

Vector Case

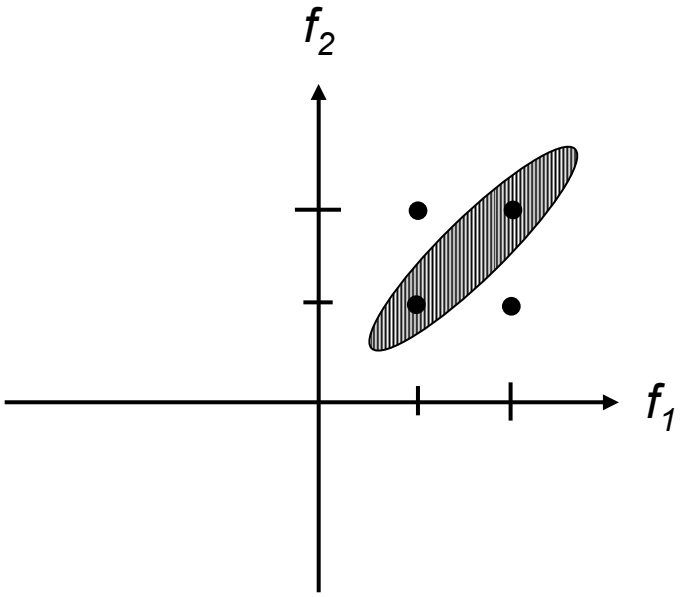
f_1, f_2: two image intensities

One approach: Separate level assignment for f_1, f_2

- previous discussions apply

Another approach: Joint level assignment for (f_1, f_2)

- typically more efficient than separate level assignment



Vector Case (cont.)

Minimize reconstruction levels/decision boundaries:

$$\epsilon = \int_{f_1} \int_{f_2} ((f_1 - \hat{f}_1)^2 + (f_2 - \hat{f}_2^{\,2}) \cdot p(f_1, f_2) \cdot df_1 \cdot df_2$$

- efficiency depends on the amount of correlation between f_1 and f_2

- finding joint density is difficult

  (Ex): Extreme case – vector level assignment for 256 x 256

  Joint density $P(x_1, x_2, \ldots, x_{256})$ is difficult to get

  Law of diminishing returns comes into play

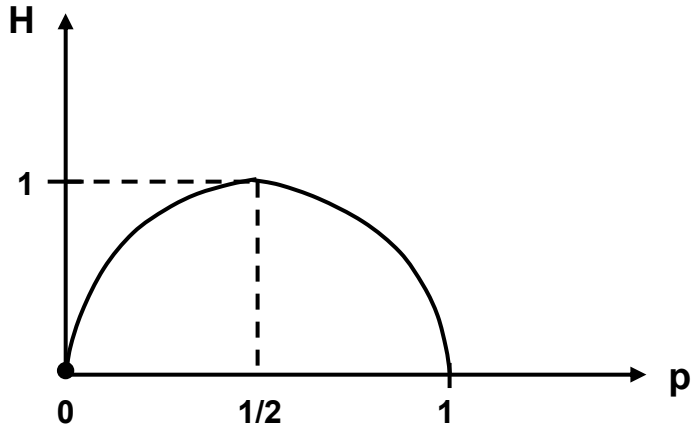
Codeword Design: Bit Allocation

- After quantization, need to assign *binray* codeword to each quantization symbol.
- Options:
  - Uniform: equal number of bits to each symbol $\rightarrow$ inefficient
  - Non-uniform: short codewords to more probable symbols, and longer codewords for less probable ones.
- For non-uniform, code has to be uniquely decodable:
  - Example: $L = 4, r_1 \rightarrow 0, r_2 \rightarrow 1, r_3 \rightarrow 10, r_4 \rightarrow 11$
  - Suppose we receive 100.
  - Can decode it two ways: either $r_3 r_1$ or $r_2 r_1 r_1$.
  - Not uniquely decodable.
  - One codeword is a *prefix* of another one.
- Use Prefix codes instead to achieve unique decodability.

Overview

- Goal: Design variable length codewords so that the average bit rate is minimized.
- Define entropy to be: $H \equiv \sum_{i=1}^{L} p_i log_2 p_i$

  pi is the probability that the ith symbol is a_i.
- Since $\sum_{i=1}^{L} p_i = 1$ we have $0 \leq H \leq log_2 L$
- Entropy: average amount of information a message contains.
- Example: L = 2, p_1 = 1, p_2 = 0 $\rightarrow$ H = 0. A symbol contains NO information.
- *p1 = p2 = ½* $\rightarrow$ *H = 1* Maximum possible value.



Overview

- Information theory: *H* is the theoretically minimum possible average bite rate.
- In practice, hard to achieve
- Example: *L* symbols, $p_i = \frac{1}{L} \longrightarrow H = log_2 L$ uniform length coding can achieve this.
- Huffman coding tells you how to do non-uniform bit allocation to different codewords so that you get unique decodability and get pretty close to entropy.

Methods of Bit Assignment (cont.)

Huffman Coding: A practical method to achieve near-optimum performance.

Example:

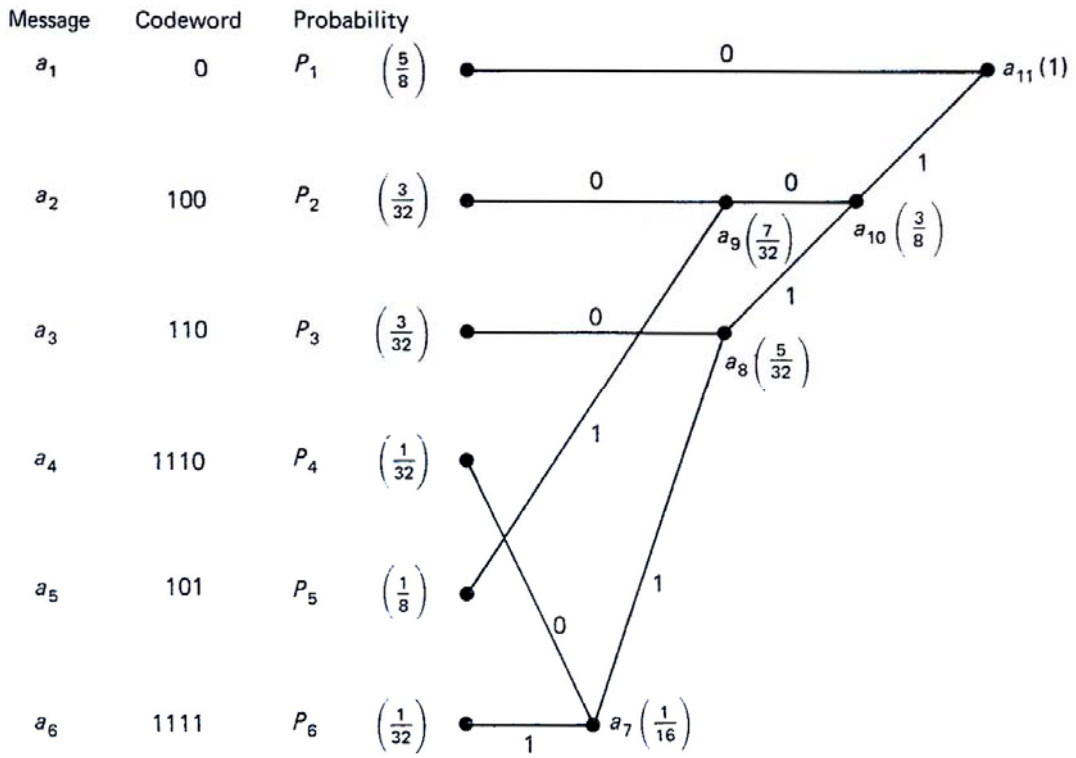


Figure10.16
Illustration of codeword generation in Huffman cding. Message possiblities with higher probabilities are assigned with shorter codewords.

Methods of Bit Assignment (cont.)

- Uniform-length codeword: 3 bits/message
- Huffman coding:

$$\frac{5}{8} \cdot 1 + \frac{3}{32} \cdot 3 + \frac{3}{32} \cdot 3 + \frac{1}{32} \cdot 4 + \frac{1}{8} \cdot 3 + \frac{1}{32} \cdot 4 = \frac{29}{16} \ bits/message$$
$$\approx 1.813 \ bits/message$$

- Entropy:

$$-(\frac{5}{8} log_2 \frac{5}{8} + \frac{3}{32} log_2 \frac{3}{32} + \frac{3}{32} log_2 \frac{3}{32} + \frac{1}{32} log_2 \frac{1}{32} + \frac{1}{8} log_2 \frac{1}{8} + \frac{1}{32} log_2 \frac{1}{32})$$
$$\approx 1.752 \ bits/message$$

Methods of Bit Assignment (cont.)

Vector Case

f_1, f_2	Prob.
$r_0, r_{\acute{0}}$	p_0
$r_1, r_{\acute{1}}$	p_1
$r_2, r_{\acute{2}}$	p_2
•	•
•	•
•	•

$$Entropy = -\sum_{i=0}^{N-1} p_i \cdot log_2 p_i$$

Comments:

1. **Finding p_i for a large number of elements in the vector is difficult.**
2. **Law of diminishing returns comes to play.**
3. **Huffman Coding can be used.**

Arithmetic Coding

- **Why not use Huffman?**
- **Can show rate of Huffman code is within** P_{max} + *0.86 of* ***Entropy***.

  P_{max} **= *Prob. of most frequent symbol***
- ***Example:***

  iid source = $\{a_1, a_2, a_3\}$

  $P(a_1) = 0.95$ $P(a_2) = 0.02$ $P(a_3) = 0.03$

  Entropy = 0.335 bits/symbol

  Huffman : 1.05 bits/symbol

⇨ ***Huffman 213% of Entropy!!***

Mechanics fo Arithmetic Coding

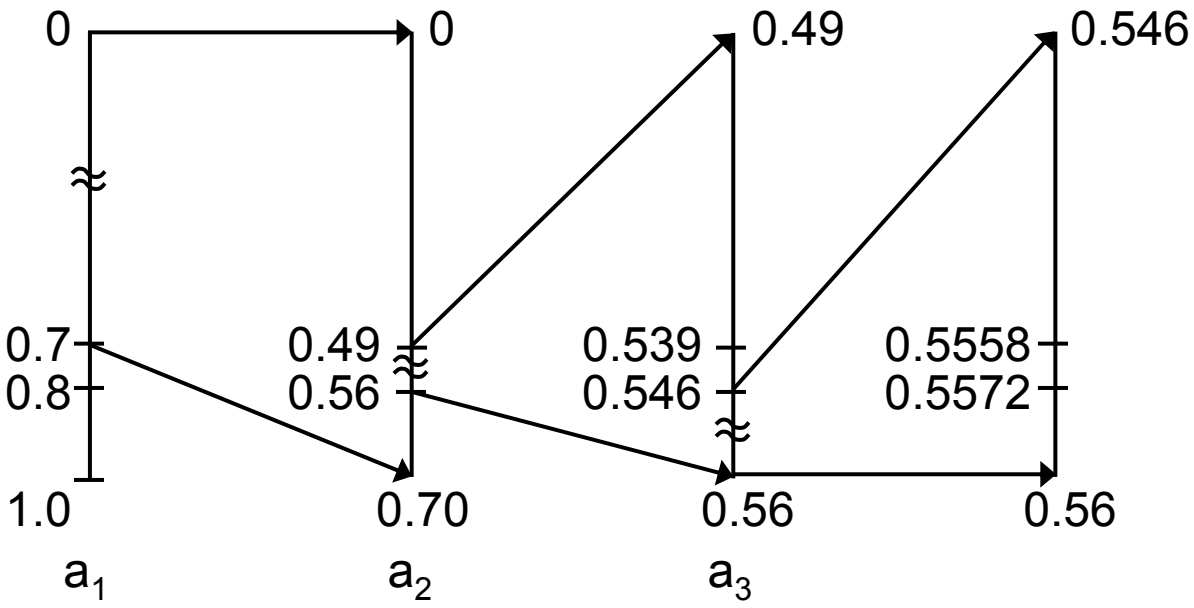
- **Two Steps:**
  1. Generate a tag
  2. Assign a binary code to the tag

- **Tag Generation:**

  A sequence of symbols from an alphabet source with a given Pdf results in a unique sub-interval in [0, 1].

Example of Tag Generation:

- $A = \{a_1, a_2, a_3\}$ $P(a_1) = 0.7$
  $P(a_2) = 0.1$ $P(a_3) = 0.2$
- **Suppose we encode** a_1, a_2, a_3

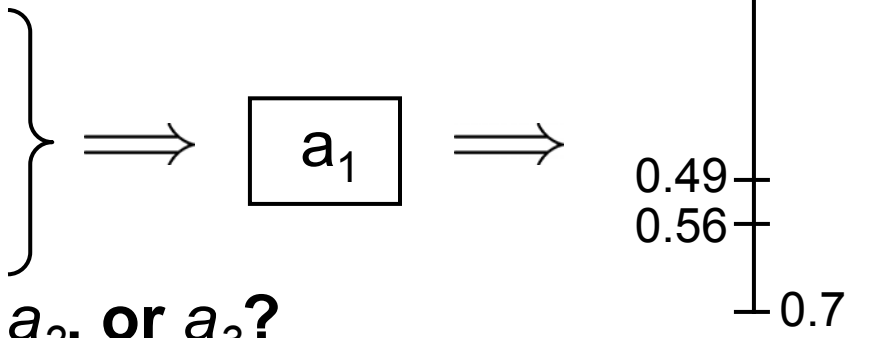


- **Interval [0.546, 0.56] uniquely specifies sequence** (a_1, a_2, a_3)

Deciphering the Tag:

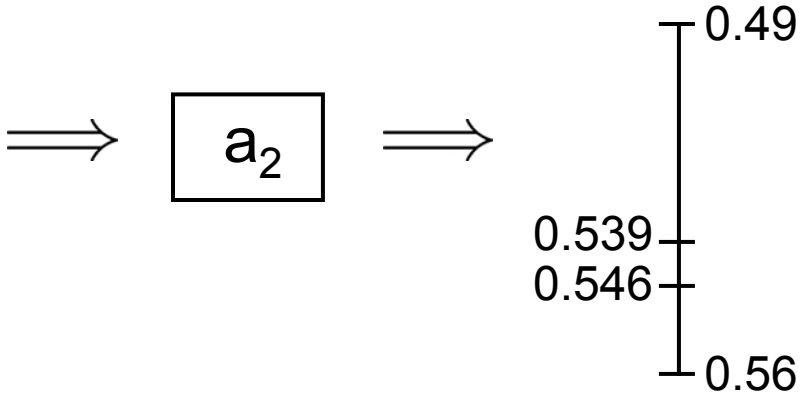
- **Suppose we know 3 symbols resulted in subinterval [0.546, 0.56]**
- **Decode the symbols?**
- **Is the first symbol a_1 or a_2 or a_3?**

a_1	[0, 0.7]
a_2	[0.7, 0.8]
a_3	[0.8, 1]



- **Is the second symbol a_1, a_2, or a_3?**

a_1	[0, 0.49]
a_2	[0.49, 0.56]
a_3	[0.56, 0.7]



Generating a binary code:

- **Uniqueness:**
- **Consider sequence of symbols**

$$\vec{X} = a_1, a_3, a_2, \ldots, a_1, \ldots a_3, a_2.$$

- **Compute** $P(\vec{X})$.
  If iid source ⇒ easy to compute
- **Write down binary representation of the point in the middle of subinterval for** $\vec{X}$**.**

- **Truncate it to** $\left\lceil log \frac{1}{P(\vec{X})} \right\rceil + 1 \; bits$

- **Can show it is unique.**

Example Generating a binary code:

- **Consider subinterval [0.546, 0.56]**
- **Midpoint: 0.553**
- $P(\vec{X}) = P(a_1)P(a_2)P(a_3) = 0.014$
- **# of bits =** $\left\lceil log\frac{1}{P(\vec{X})} \right\rceil + 1 = 8\ bits$
- **Binary Representation of 0.553**

$$0.553 = \frac{1}{2^1} + \frac{1}{2^5} + \frac{1}{2^6} + \frac{1}{2^8} + \frac{1}{2^9} + \ldots = 0.1\ \ 00011011\ldots$$

- **Truncate to 8 bits ⇒ Binary Representation:**

  10001101

Efficiency of Arithmetic Code

- **We can show** $H(X) \leq l_A \leq H(X) + \frac{2}{m}$
  - **m # of symbols in sequence**
  - l_A **= average length per symbol**
  - *H(X)* **= entropy of source**
  - **Assumes iid source.**

- **Observe: By increasing length of sequence, can get arbitrarily close to entropy.**

Dictionary Techniques

- Huffman and arithmetic coding assumed i.i.d. sources
- Most sources are correlated
- Main idea:
    1. Build a list of commonly occurring patterns
    2. Transmit index in the list.
- Exploits fact that certain patterns recur frequently.
- Consider 2 cases:
    1. Static Dictionary
    2. Dynamic Dictionary

Static Dictionary:

- **Suppose five letter alphabet source:**

  $A = \{a, b, c, d, r\}$

- **Using statistics of source, build dictionary**

Entry	ad	ac	ab	r	d	c	b	a
Code	111	110	101	100	011	010	001	000

- **Encode a b r a c a d a b r a**
  1. **Read ab → 101**
  2. **Read ra → not in dictionary**
  3. **Read r → 100**
  4. **Read ac → 110**
  5. **……..**

101	100	110	111	101	100	000
ab	r	ac	ad	ab	r	a

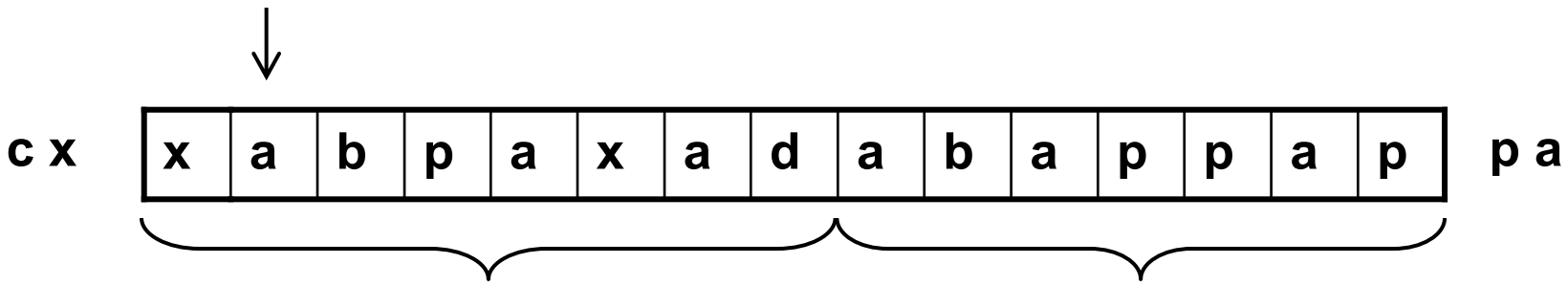
- **Opposite of Huffman coding:**

Adaptive Dictionary:

- Ziv Lempel 1977 + 1978
- LZ1 $\rightarrow$ 1977 LZ2 $\rightarrow$ 1978
- LZ1 discussed Here.
- Basic idea:

  Dictionary portion of previously encoded sequence
- Sliding window:
  1. Search buffer
  2. Lookahead buffer

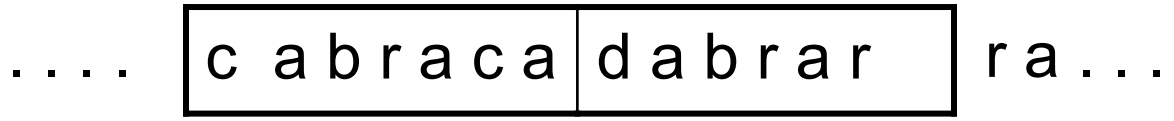
  Match pointer



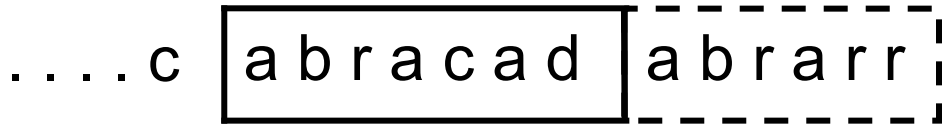
Example

. . . c a b r a c a d a b r a r r a r r a d . . .

– Window = 13
– Look ahead buffer = 6
– Search buffer = 7



1. No match to d $\rightarrow$ < 0, 0, C(d) >



Example (cont.)

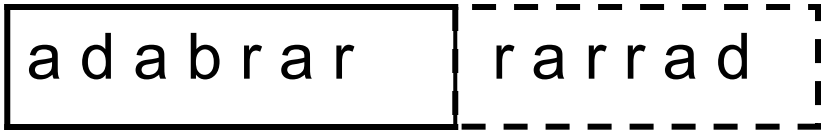
2. Match for a:

o = 2 ----> *l* = 1

0 = 4 ----> *l* = 1

0 = 7 ----> *l* = 4

< 7, 4, C (r) >



Example (cont.)

3. Match for r:

$$\begin{cases} 0 = 1 \dashrightarrow l = 1 \\ 0 = 3 \dashrightarrow l = 5 \end{cases}$$

< 3, 5, C (d) >

Exceeds the boundary between search and look ahead buffer

Encoding steps:

1. Move search pointer back until match in search buffer.
2. Offset $\triangleq$ distance of pointer from look ahead buffer.
3. Do consecutive symbols of pointer match also?
4. Search the search buffer for the longest match.
5. Length of match $\triangleq$ # of consecutive symbol match.
6. Send <o, *l*, c)

   o = offset

   l = match length

   C = codeword of symbol in LA buffer, following match.

Example: <7, 2, codeword for a >

Adaptive Dictionary

- **Why send C?**
  - Just in case no match
- **Total # of bits:**

$$\lceil log\ _2 S \rceil + \lceil log\ _2 W \rceil + \lceil log\ _2 A \rceil$$

S = size of search buffer

W = Size of window (search + LA)

A = size of source alphabet

What to Code
(Classification of Image Coding Systems)

1. **Waveform Coder (code the intensity)**
   - PCM (Pulse Code Modulation) and its improvements
   - DM (Delta Modulation)
   - DPCM (Differential Pulse Code Modulation)
   - Two-channel Coder

What to Code
(Classification of Image Coding Systems) (cont.)

2. **Transform Coder (code transform coefficients of an image)**
   - Karhunen-Loeve Transform
   - Discrete Fourier Transform
   - Discrete Cosine Transform
3. **Image Model Coder**
   - Auto-regressive Model for texture
   - Modelling of a restricted class of images

NOTE: Each of the above can be made to be adaptive

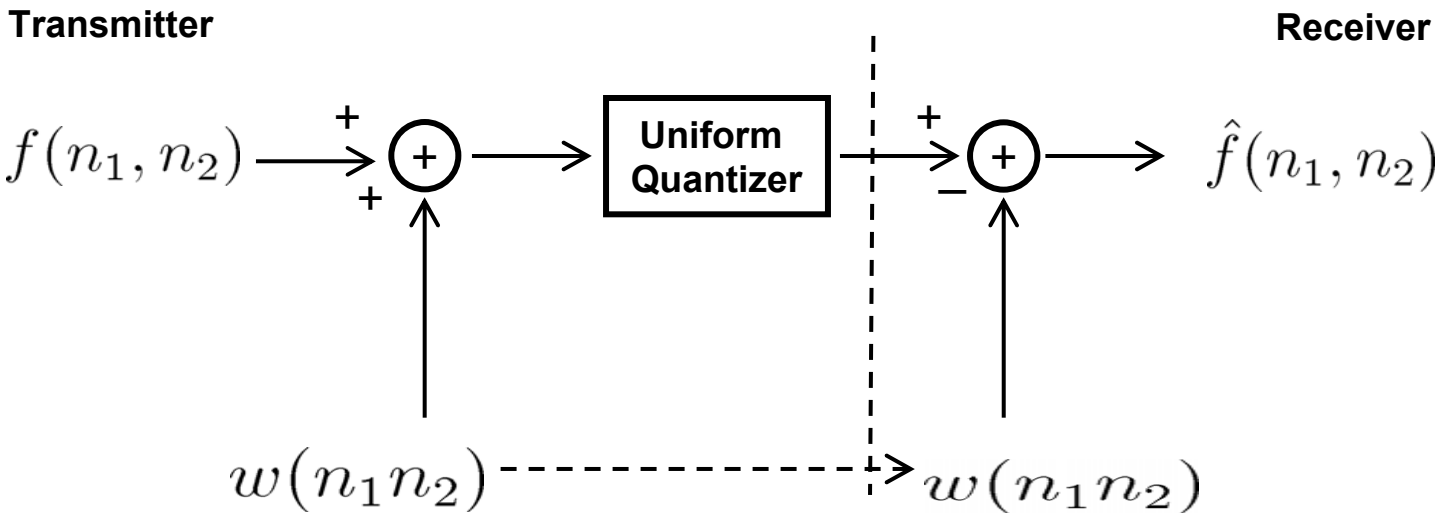
Waveform Coder

- **PCM Coding**

- **Very simple**
- **Typically requires over 5-6 bits/pixel for good quality**
- **False contours for low-bit rate case**

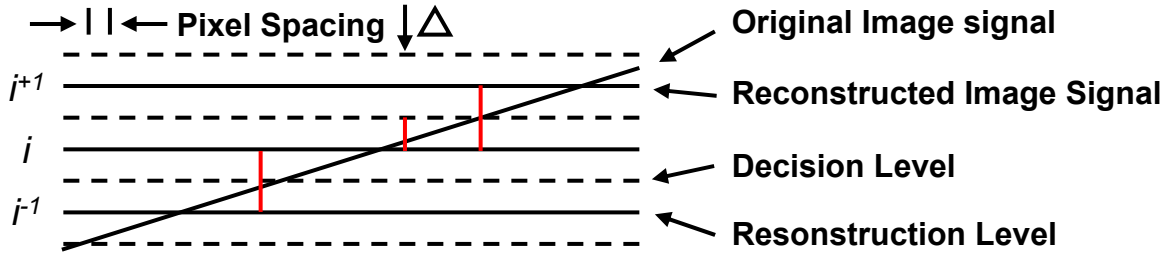
Improvements of PCM

1. **Roberts' Pseudo-Noise Technique:**



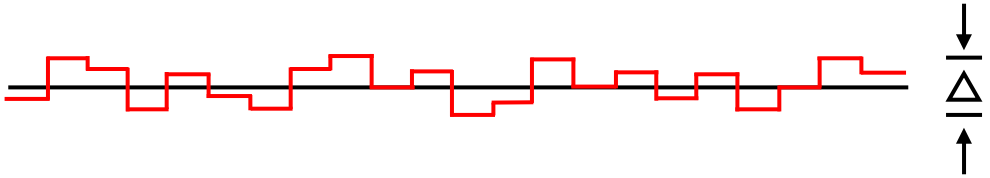
$$P(w) = \begin{cases} \frac{1}{\Delta} & -\frac{\Delta}{2} \leq w_0 \leq \frac{\Delta}{2} \\ 0 & \text{otherwise} \end{cases}$$

Improvements of PCM (cont.)



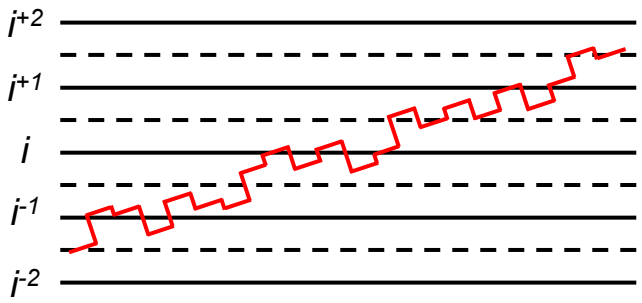
(a) Nominal Quantization

- False contours disappear – replaced by additive random noise

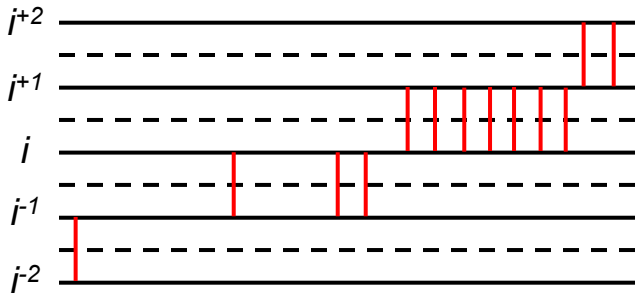


(b) One bit Pseudonoise

Improvements of PCM (cont.)

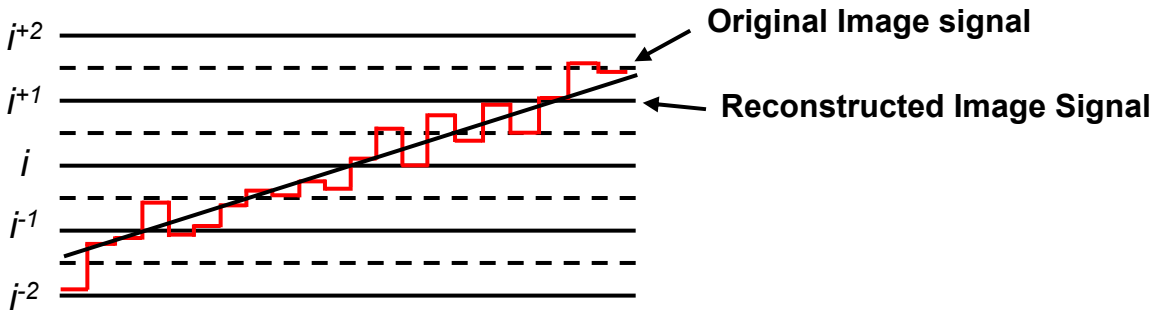


(c) Original Image Signal Plus Noise



(d) Quantized Image Signal Plus Noise

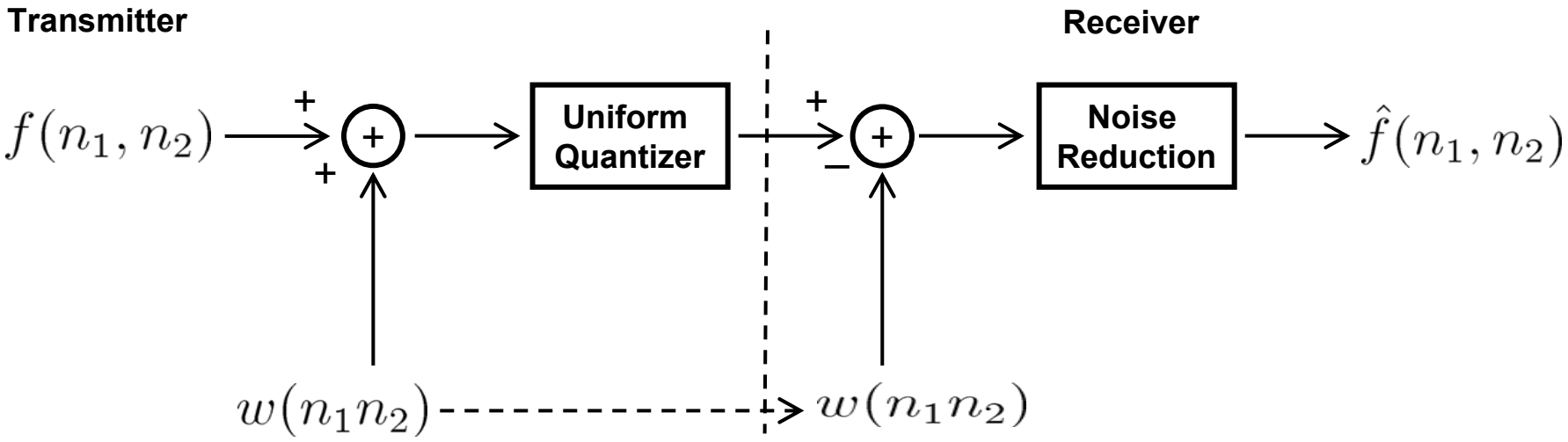
Improvements of PCM (cont.)



(e) Pseudonoise Quantization

Improvements of PCM (cont.)

2. Roberts' Pseudo-Noise Technique with Noise Reduction:



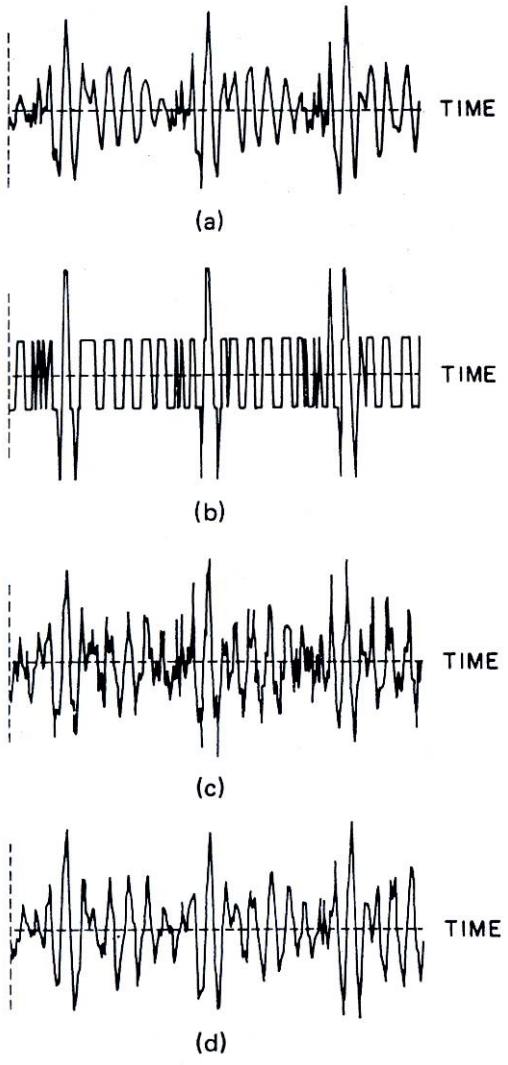


Figure 10.21
Example of quantization
Noise reduction in PCM speech coding.
(a) Segment of noise-free voiced speech;
(b) PCM-coded speech at 2 bits / sample;
(c) PCM-coded speech at 2 bits / sample by Roberts' pseudonoise technique;
(d) PCM-coded speech at 2 bits / sample with Quantization noise reduction.

(a) (b)

Figure 10.22
Example of quantization noise reduction in PCM image coding. (a) Original image of 512 x 512 pixels; (b) PCM-coded image at 2 bits/pixel.

(c)

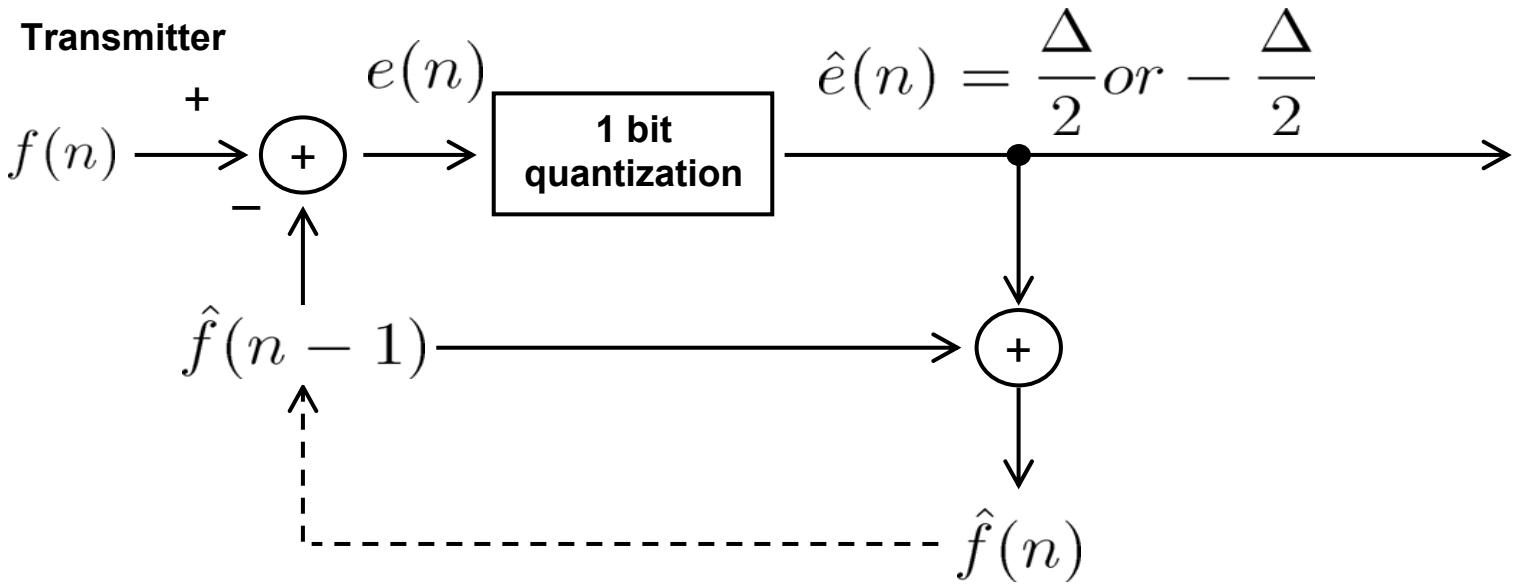
(d)

Figure 10.22
Example of quantization noise reduction in PCM image coding. (c) PCM-coded image at 2 bits/pixel by Roberts's pseudonoise technique; (d) PCM-coded image at 2 bits/pixel with quantization noise reduction.

Delta Modulation (DM)

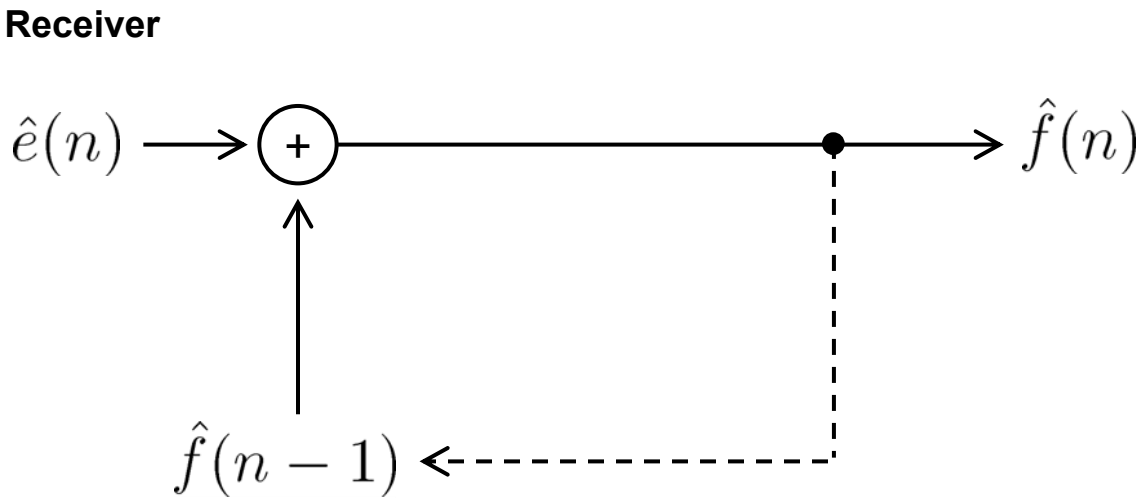
$$fn_1, n_2) : \quad signal, \qquad \hat{f}(n_1, n_2) : \quad coded\ signal$$

Transmitter



Delta Modulation (DM) (cont.)

Receiver



Delta Modulation (DM) (cont.)

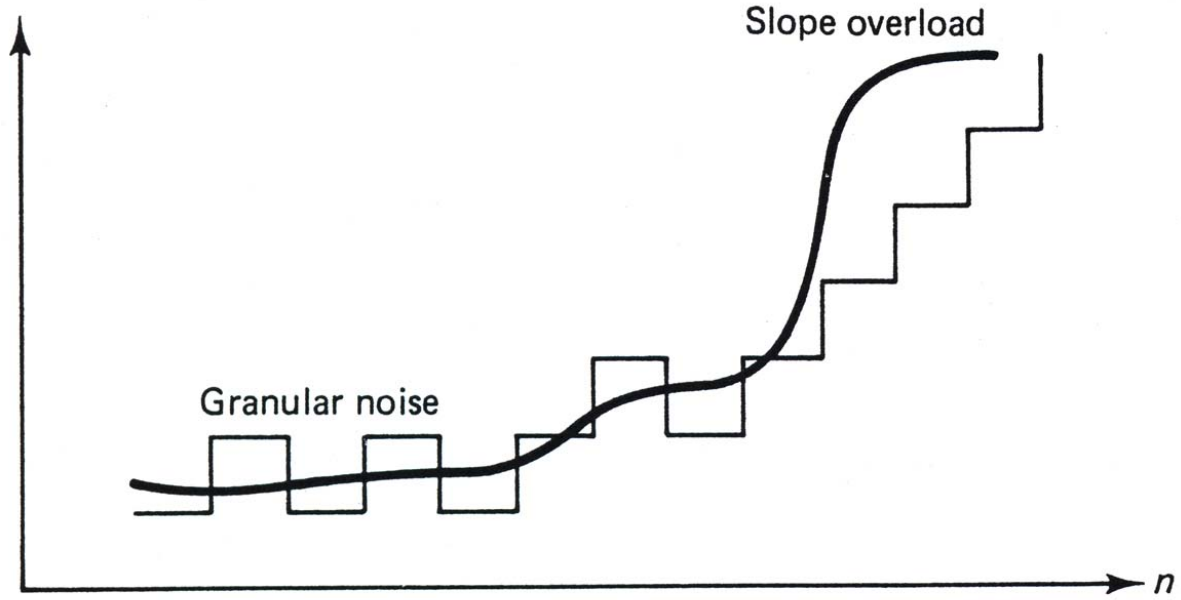


Figure 10.26
Granular noise and slope-overload distortion in delta modulation.

- Needs over 2-3 bits/pixel to get good quality

(a)

(b)

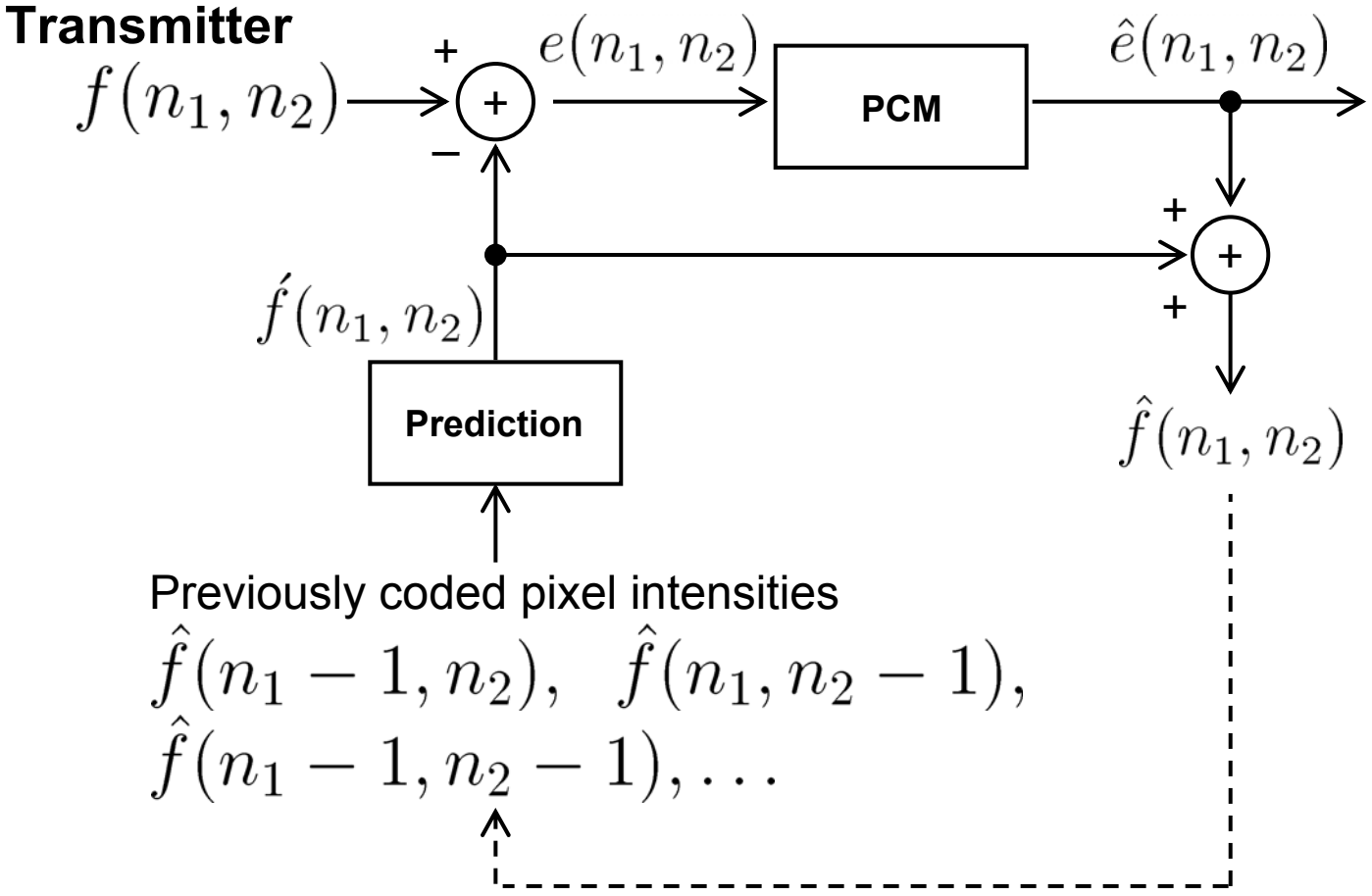
Figure 10.27
Example of delta-modulation (DM)-coded image. The original image used is the image in Figure 10.22(a). (a) DM-coded image with = 8% of the overall dynamic range. NMSE = 14.8%, SNR = 8.3 dB; (b) DM-coded image with = 15%, NMSE = 9.7%, SNR = 10.1 dB.

Figure 10.28
DM-coded image at 2 bits/pixel. The original image used is the image in Figure 10.22(a). NMSE = 2.4%, SNR = 16.2 dB.

Differential Pulse Code Modulation (DPCM)

$f(n_1, n_2)$: *original image*

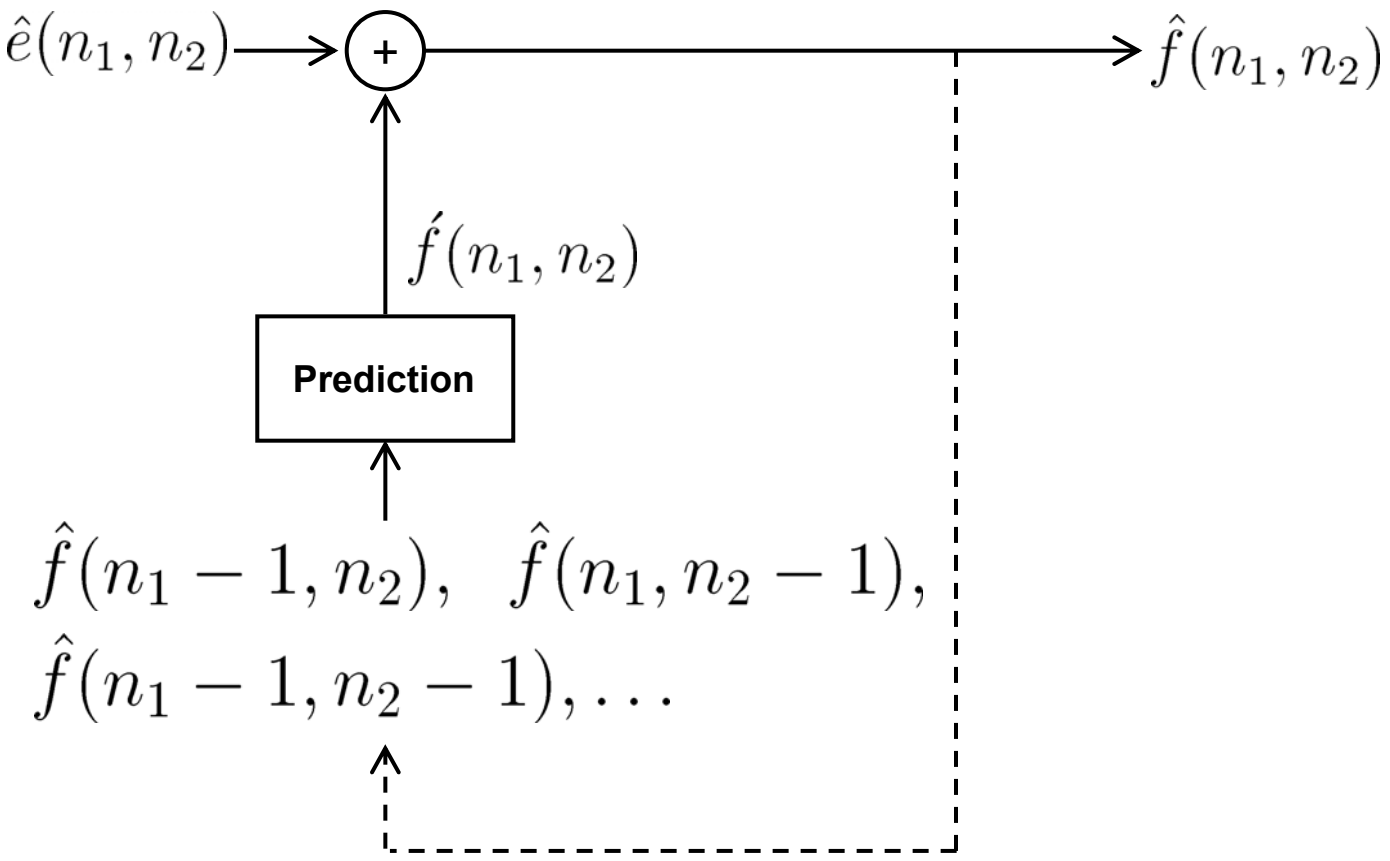
$\hat{f}(n_2, n_2)$: *reconstructed image*



- **The Auto-regressive Model parameters are obtained from the image by solving a linear set of equations or by a Markov process assumption**

Differential Pulse Code Modulation (DPCM) (cont.)

Receiver

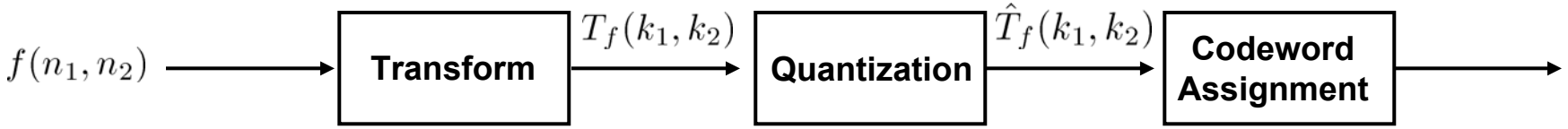


- **Requires 2-3 bits/pixel for good quality image**

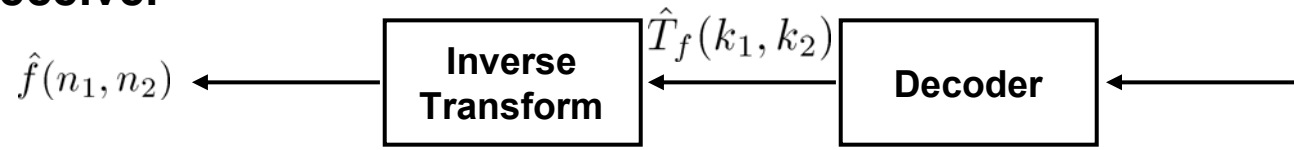
Figure 10.30
Example of differential pulse code modulation (DPCM)-coded image at 3 bits/pixel. Original image used is the image in Figure 10.22(a). NMSE = 2.2%, SNR = 16.6 db.

Transform Image Coding

Transmitter



Receiver



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

Transform Image Coding

Some considerations:

- **Energy compaction in a small number of coefficients**
- **Computational aspect: important (subimage by subimage coding – 8 x 8 – 16 x 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

$$K_f(n_1,n_2;l_1,l_2) = E[(x(n_1,n_2)-\bar{x}(n_1,n_2))\cdot(x(l_1,l_2)-\bar{x}(l_1,l_2))]$$

Examples of Transforms (cont.)

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**

Need better scan of Figure 5.3.7

Figure 5.3.7
Images used for coding and statistics. (a) “Karen” has much more stationary Statistics than (b) “Stripes.”

Figure 5.3.7
Images used for coding and statistics. (a) “Karen” has much more stationary Statistics than (b) “Stripes.”

Basic Compression Techniques

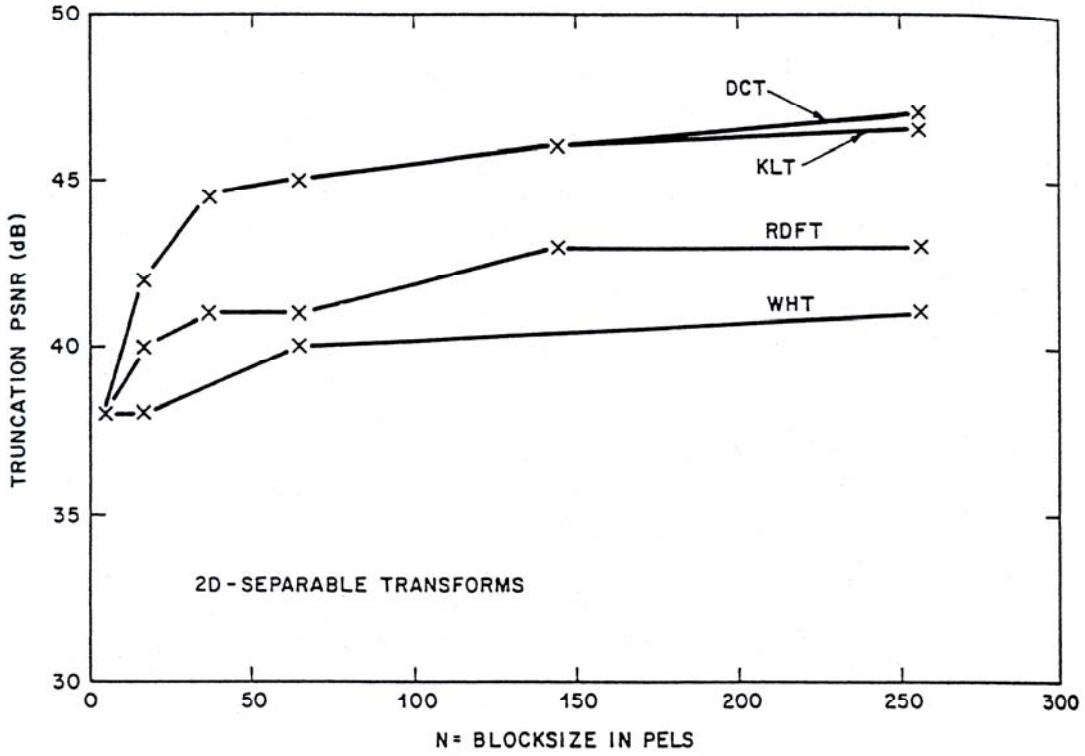
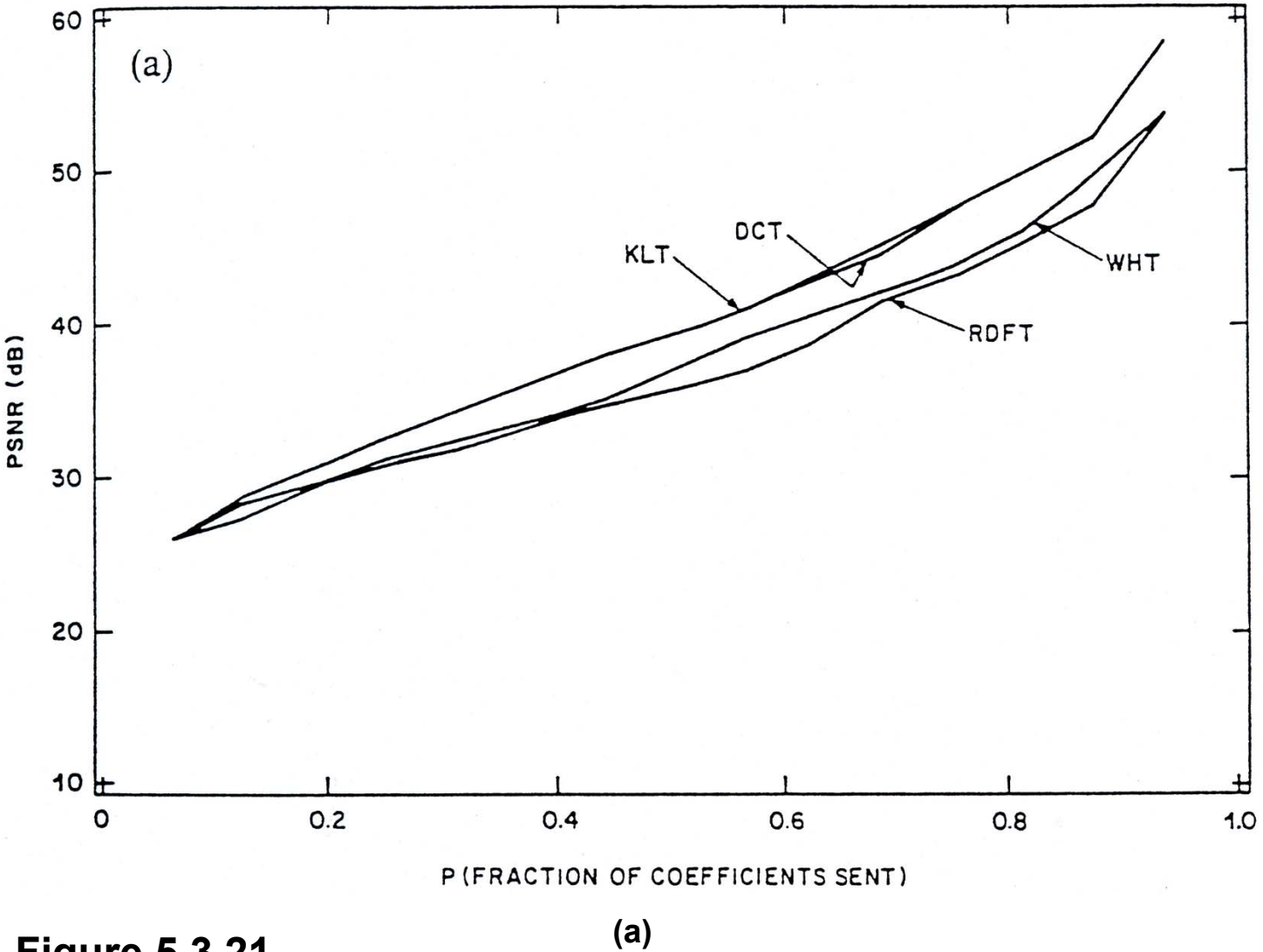


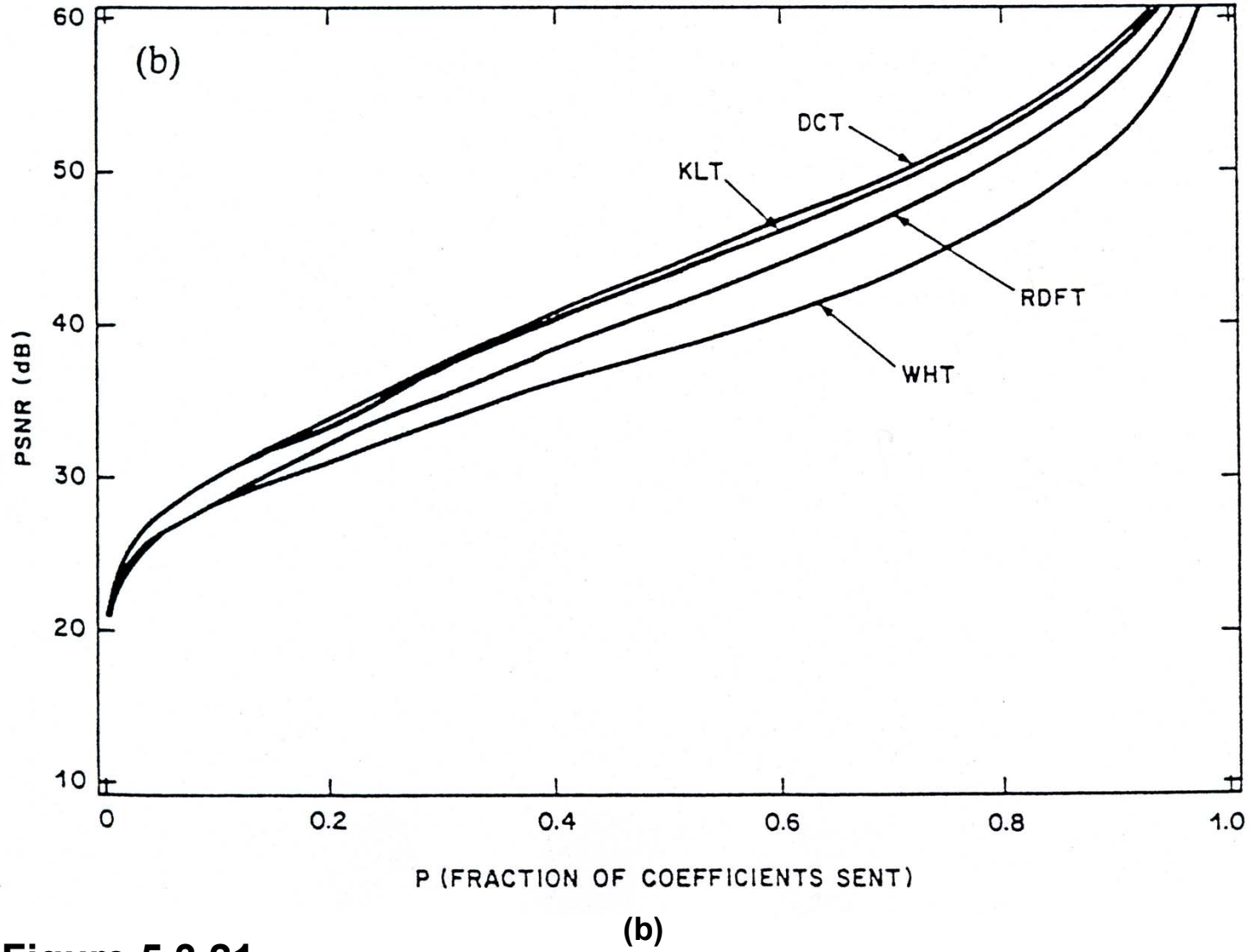
Figure 5.3.22
Truncation PSNR versus block size for separable transforms with the image "Karen" when 60 percent of the coefficients are kept (p=0.6).



(a)

Figure 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 x 4 blocks, *N = 16*. (b) 16 x 16 blocks, *N = 256*.



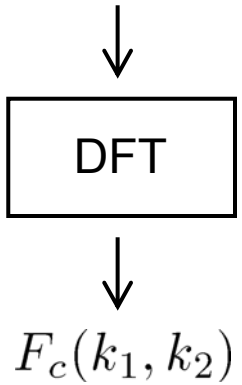
(b)

Figure 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 x 4 blocks, *N = 16*. (b) 16 x 16 blocks, *N = 256*.

Discrete Cosine Transform

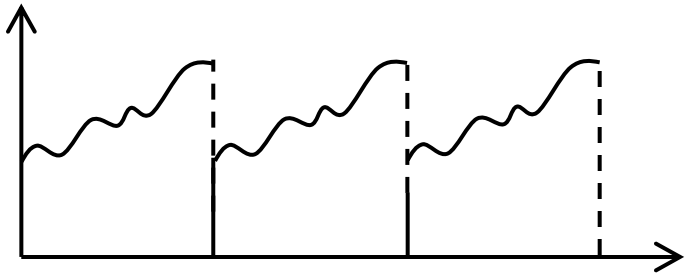
$$f(n_1, n_2) \rightarrow \acute{f}(n_1, n_2) = \frac{f(n_1, n_2) + f(n_1, -n_2) + f(-n_1, n_2) + f(-n_1, -n_2)}{4}$$



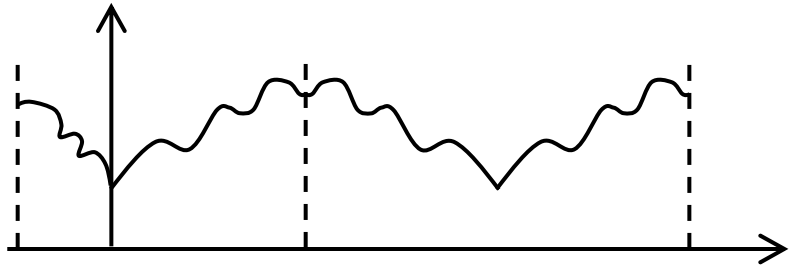
$F_c(k_1, k_2)$

Comments:

- Good energy compaction (better than DFT)



Sharp discontinuity



No sharp discontinuity

Discrete Cosine Transform (cont.)

Comments (cont.):

- 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

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

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

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

Discrete Cosine Transform (cont.)

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

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

Equation (3.30) can also be expressed as

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

where

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

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

Discrete Cosine Transform (cont.)

Discrete Cosine Transform Pair

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

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

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

Computation of Discrete Cosine Transform

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

Discrete Cosine Transform (cont.)

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

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

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

Discrete Cosine Transform (cont.)

Computation of Inverse Discrete Cosine Transform (cont.)

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 DFT.

Discrete Cosine Transform (cont.)

Computation of Inverse Discrete Cosine Transform (cont.)

In the derivation of the DCT pair, we have used an intermediate sequence 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 $y(n)$, 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.

Discrete Cosine Transform (cont.)

Computation of Inverse Discrete Cosine Transform (cont.)

$$y(n) = x(n) + x(2N-2-n) - x(N-1)\delta(n-(N-1))$$

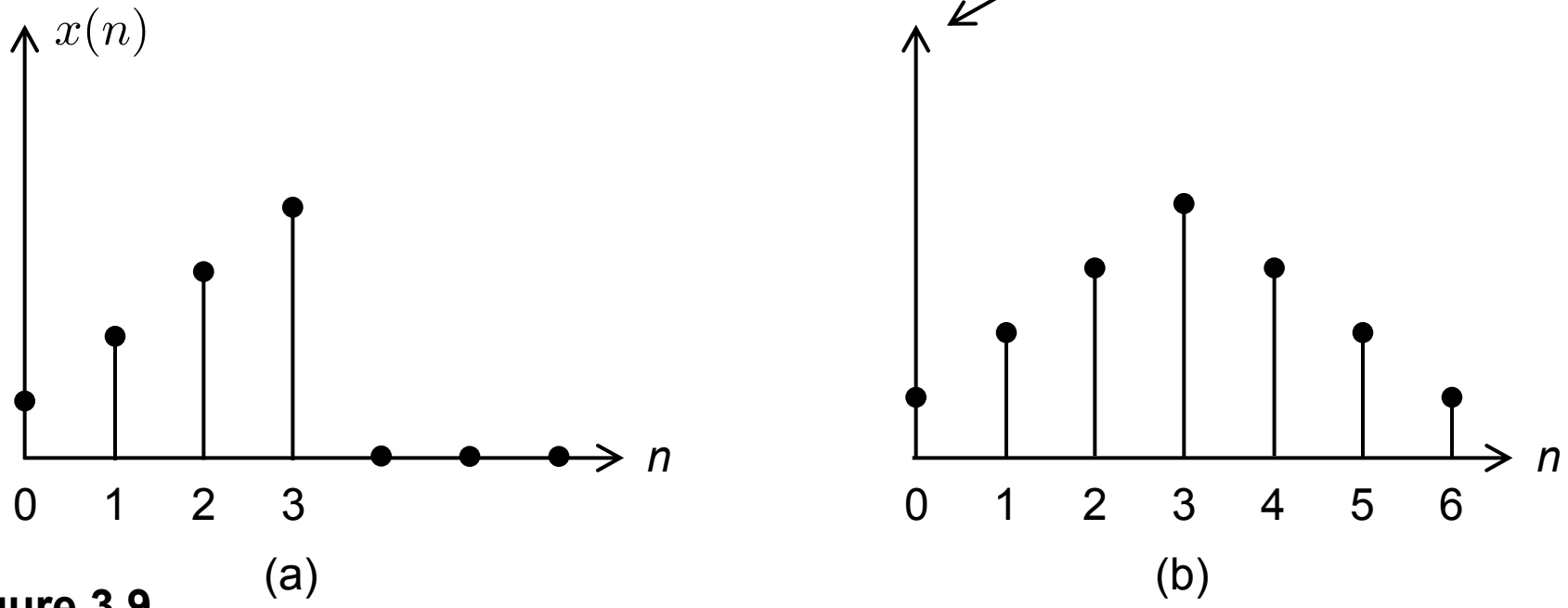


Figure 3.9
Example of (a) $x(n)$ and (b) $y(n) = x(n) + x(2N-2-n) - x(N-1)\delta(n-(N-1))$
The sequence $y(n)$ is used in the intermediate step in defining the odd symmetrical discrete cosine transform of $x(n)$.

DCT

- Signal independent
- ρ - - -> 1 : KLT - - -> DCT

  For first order Markov Image model
- Type II DCT:

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

$$\sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} s(n_1, n_2) cos\left(\frac{\pi 2(n_1+1)K_1}{2N}\right)$$

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

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

DCT (cont.)

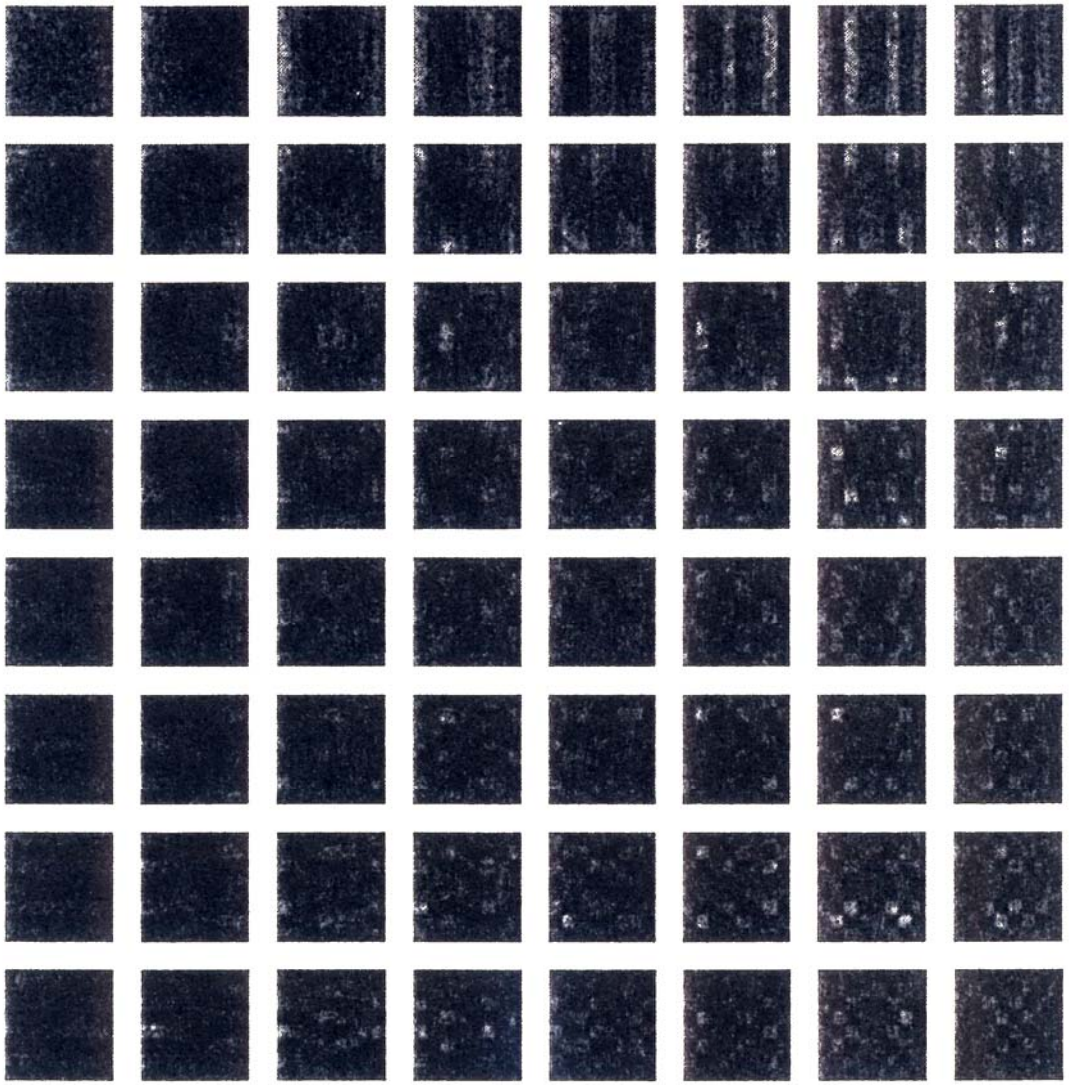


Figure 12.4 The basis matrices for the DCT

DCT (cont.)

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 of the 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 sequnece 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.

DCT (cont.)

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

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 zeros)

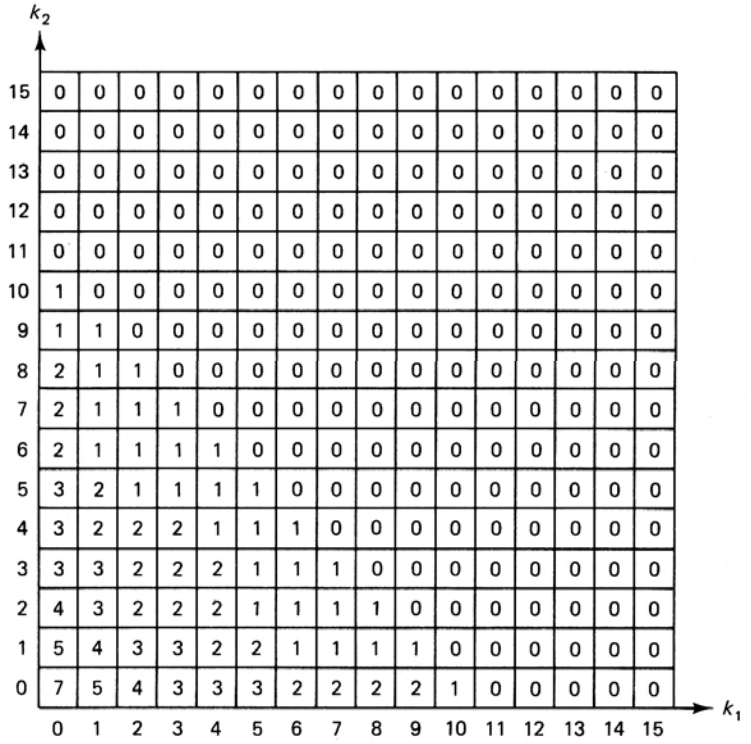


Figure 10.44
Example of a bit allocation map at 1 bit/pixel for zonal discrete cosine transform image coding. Block size = 16 x 16 pixels.

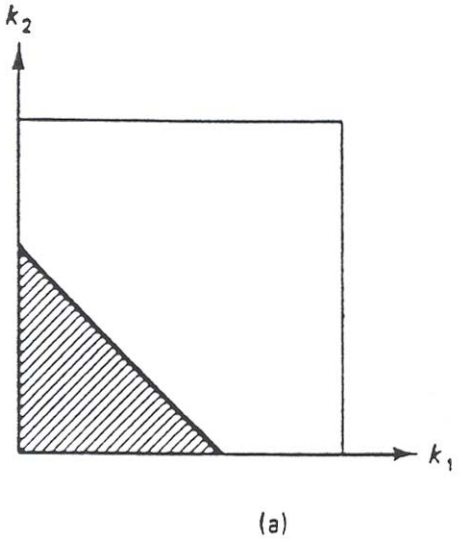
Discarding Transform Coefficients

Zonal coding: Eliminate coefficients in a fixed zone.

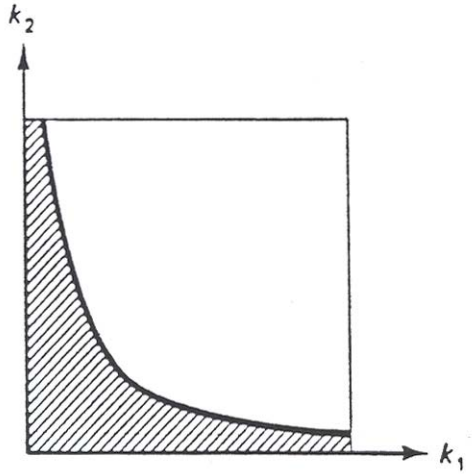
# of Bits for coefficient ***i*** with variance 6_i^2

$$bi = \frac{B}{M} + \frac{1}{2} log_2 6i^2 - \frac{1}{2M} \sum_{i=1}^{M} log_2 6_i^2$$

M = # of coefficients to be coded
B = total # of bits



(a)



(b)

Scalar Quantization of a Vector Source

- Assume N scalars: $f_i \quad 1 \leq i \leq N$
- 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: $\sum_{i=1}^{N} E\left[(f_i' - f_i)^2\right]$ 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: $B_i = \frac{B}{N} + \frac{1}{2} log \frac{\sigma_i^2}{\left[\Pi_{j=1}^{N} \sigma_j^2\right]^{1/N}}$
- σ_i^2 is the variance of f_i : $L_i = \frac{\sigma_i}{\left[\Pi_{j=1}^{N} \sigma_i\right]^{1/N}} 2^{B/N}$
- L_i is the number of reconstruction levels for source i.

DCT-Coded Images

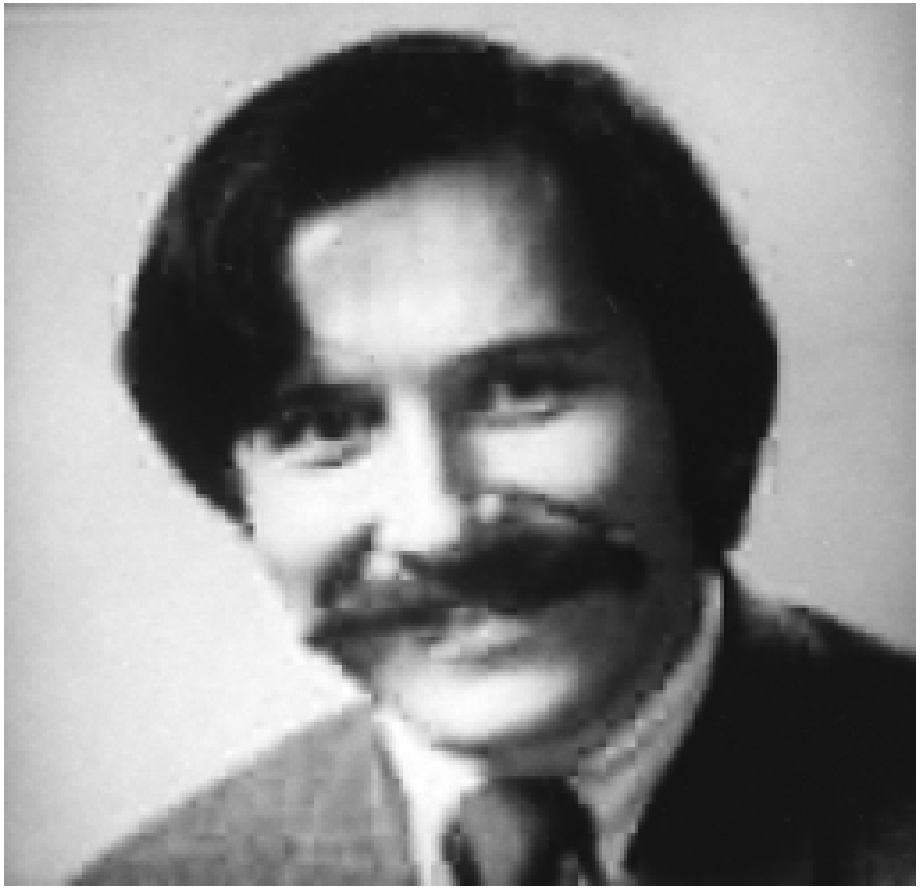
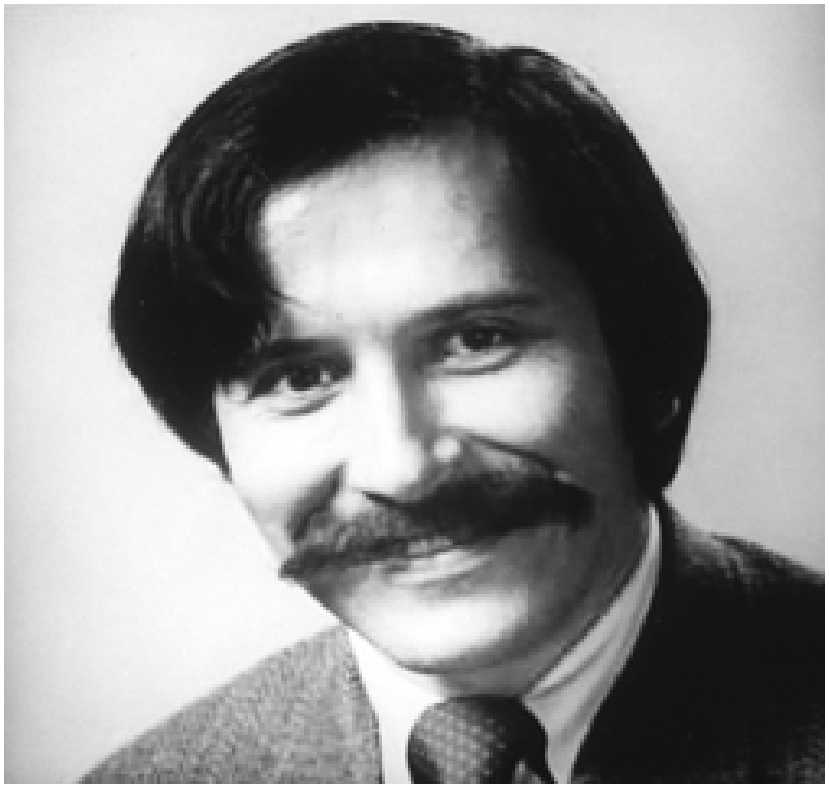
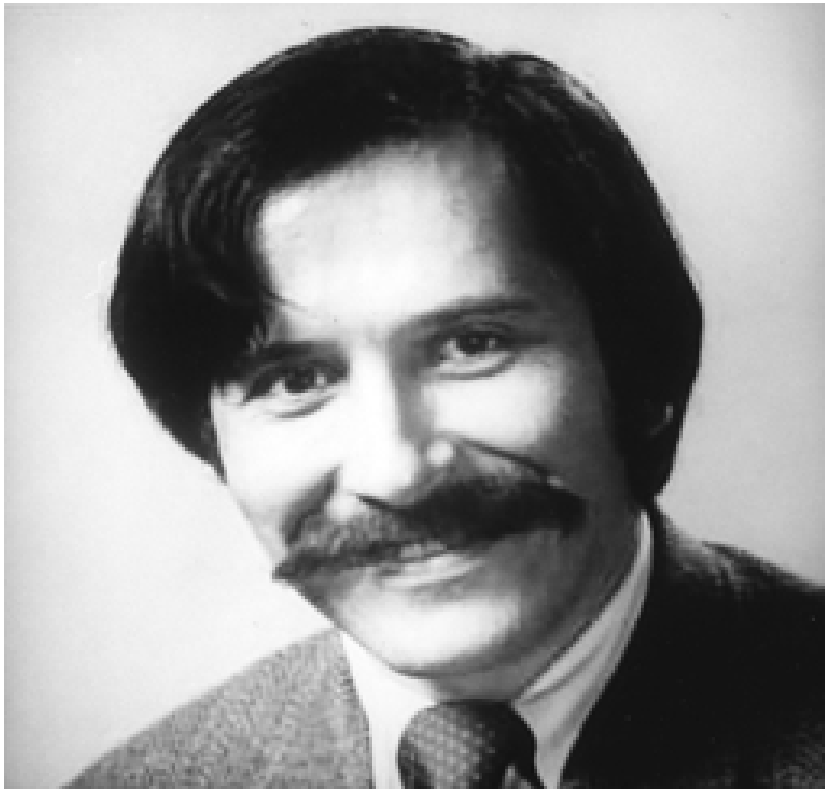


Figure 10.47
DCT-coded image with visible blocking effect.

DCT-Coded Images (cont.)



(a)



(b)

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 ½ bit/pixel. NMSE = 0.9%, SNR = 20.2 dB.

Quantization of DCT Coefficients

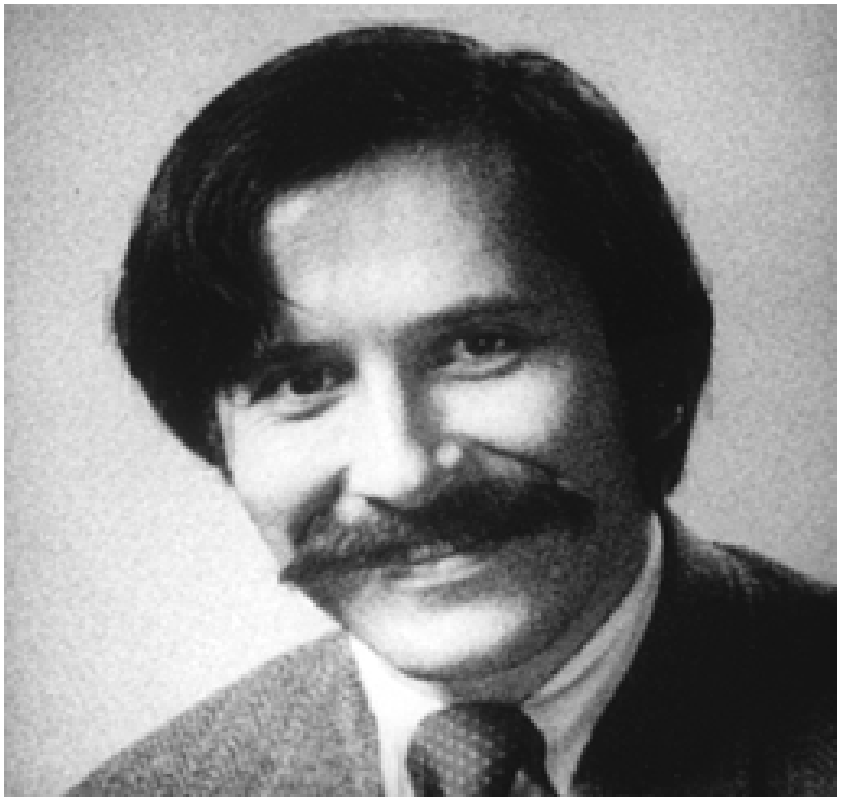
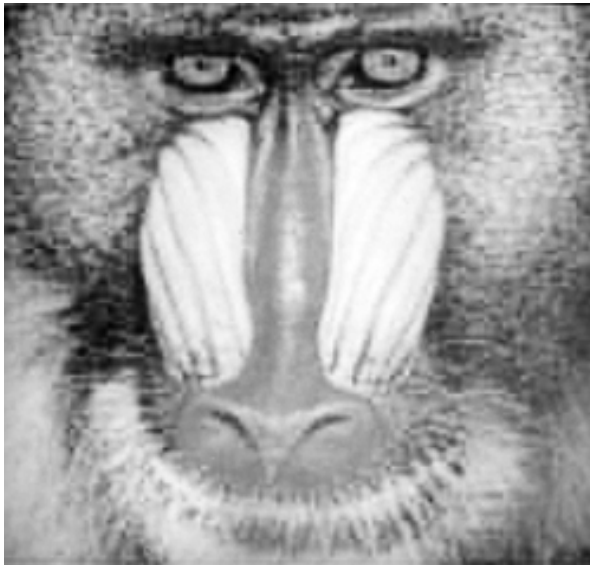


Figure 10.46
Illustration of graininess increase due to quantization of DCT coefficients. A 2-bit/pixel uniform quantizer was used to quantize each DCT coefficient retained to reconstruct the image in Figure 10.45(b)

Blocking Effect Reduction



(a)

(b)

Figure 10.50

Example of blocking effect reduction using a filtering method. (a) Image of 512 x 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 x 3-point $h(n_1, n_2)$ with $h(0, 0)=1/5$ and $h(n_1, n_2) = 1/10$ at the remaining eight points.

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 zone can be adapted.

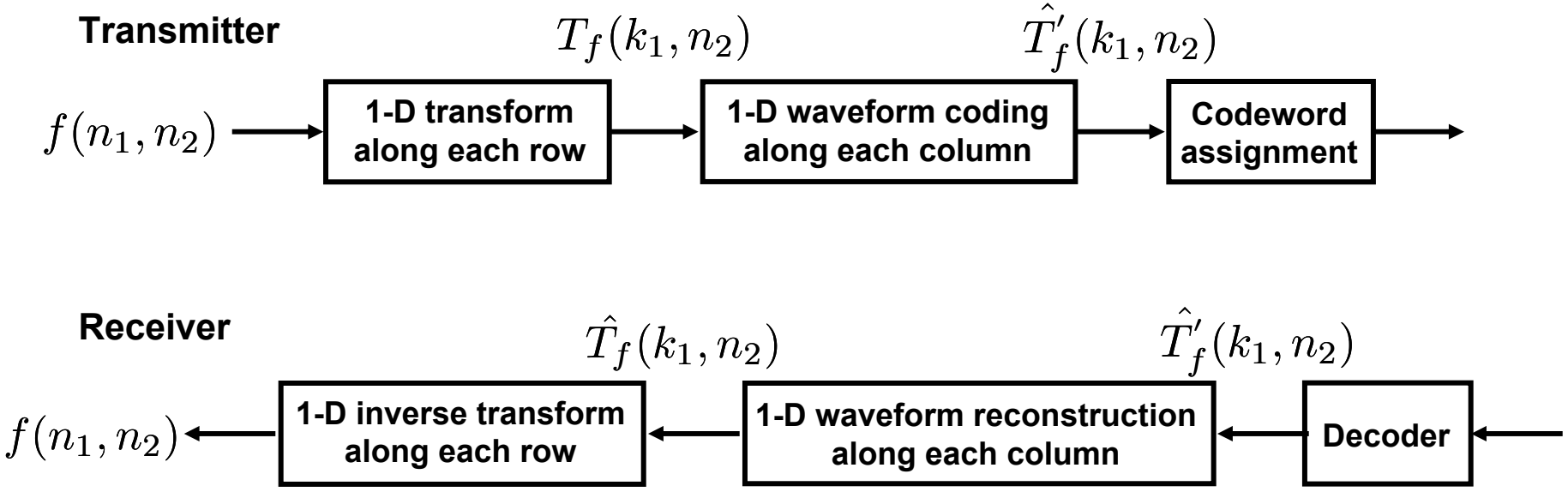


Figure 10.51 Hybrid transform/waveform coder.

Iterative Procedures for Reduction of Blocking Effects in Transform Image Coding

by

Ruth Rosenholtz and Avideh Zakhor

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 (cont.)

- **Hybrid coding useful in interframe coding.**

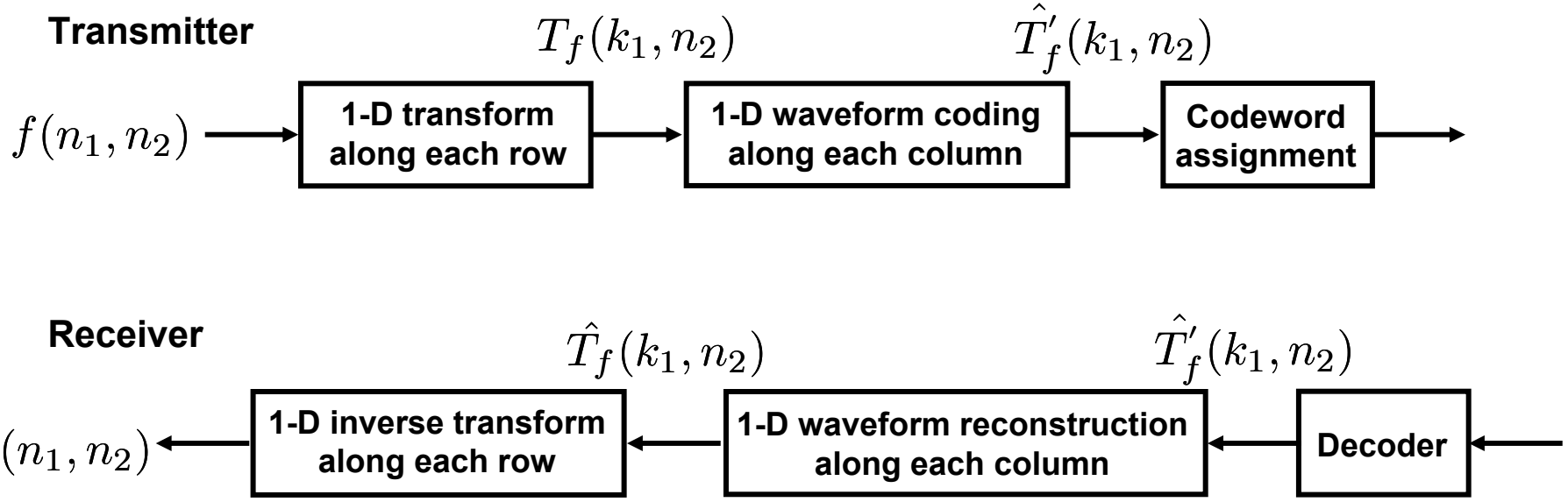


Figure 10.51 Hybrid transform/waveform coder.

Hybrid Coding (cont.)

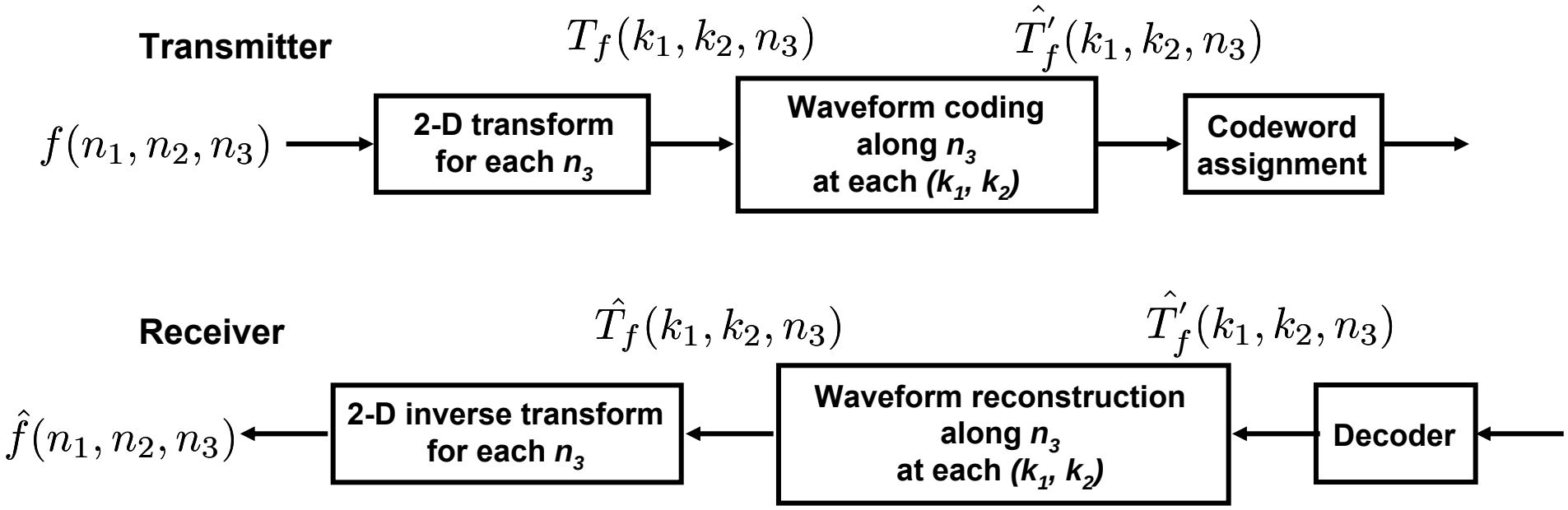
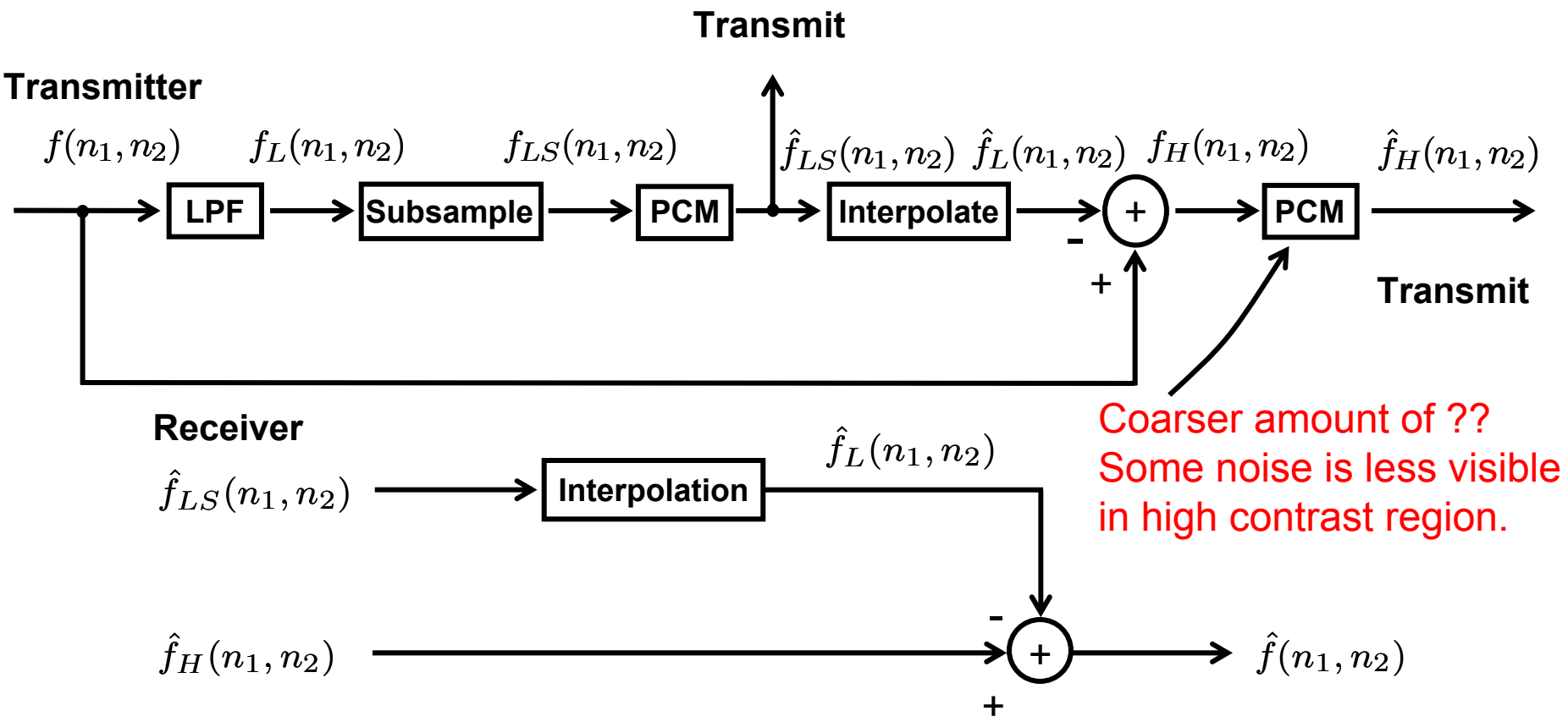


Figure 10.51 Interframe hybrid coder.

Two-Channel Image Coder



$f_L(n_1, n_2)$: Can be under-sampled (typically by 8 x 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).

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

Two-Channel Image Coding (cont.)

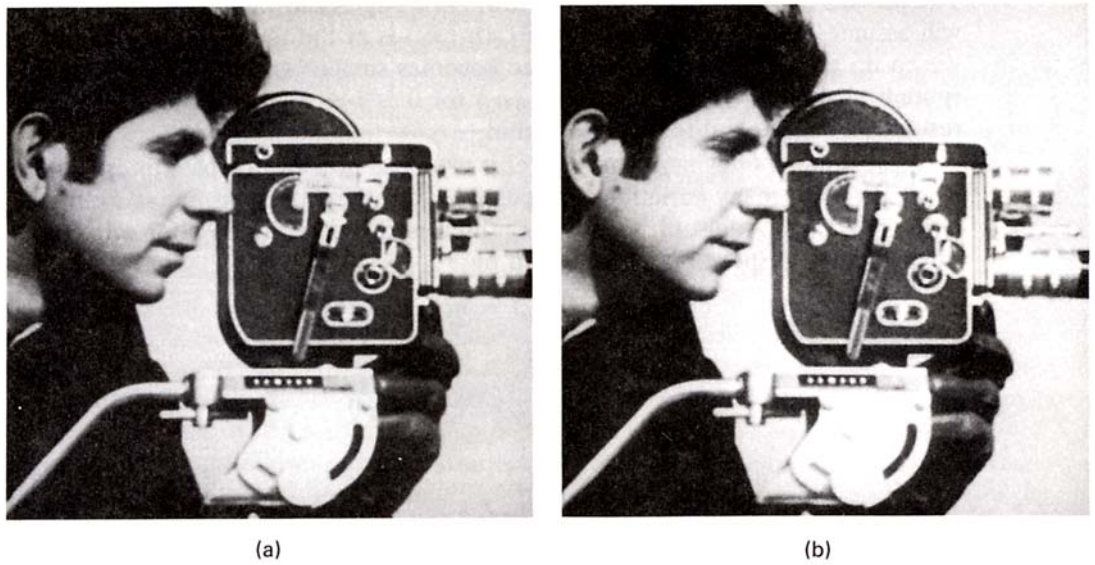


Figure 10.32
Example of image coding by at two-channel coder. (a) Original image of 512 x 512 pixels; (b) coded image at 3 1/8 bits/pixel. NMSE = 1.0%, SNR = 19.8 dB.

Pyramid Coding and Subband Coding

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

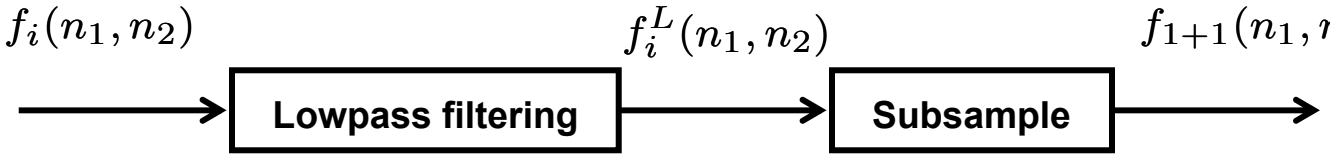


Figure 10.33
Process of generating the i + 1th-level image $f_{i+1}(n_1, n_2)$ from the ith-level image $f_i(n_1, n_2)$ in Gaussian pyramid image representation.

- **Filtering:**

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

- **Subsampling:**

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

Pyramid Coding and Subband Coding (cont.)

- **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 (cont.)

- **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.**

Laplacian Pyramid Generation

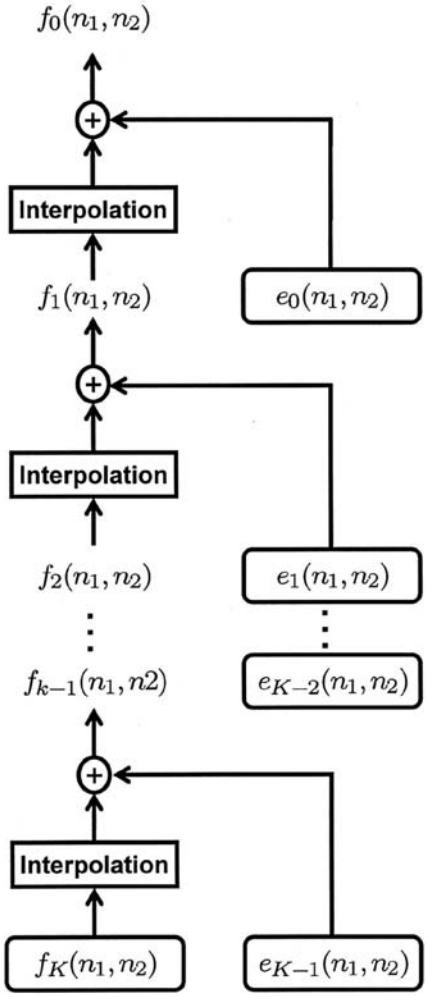


Figure 10.37
Laplacian pyramid generation. The base image $f_0(n_1, n_2)$ can be reconstructed from $e_i(n_1, n_2)$ For $0 \le i \le K-1$ and $f_K(n_1, n_2)$.

Gaussian Pyramid Representation

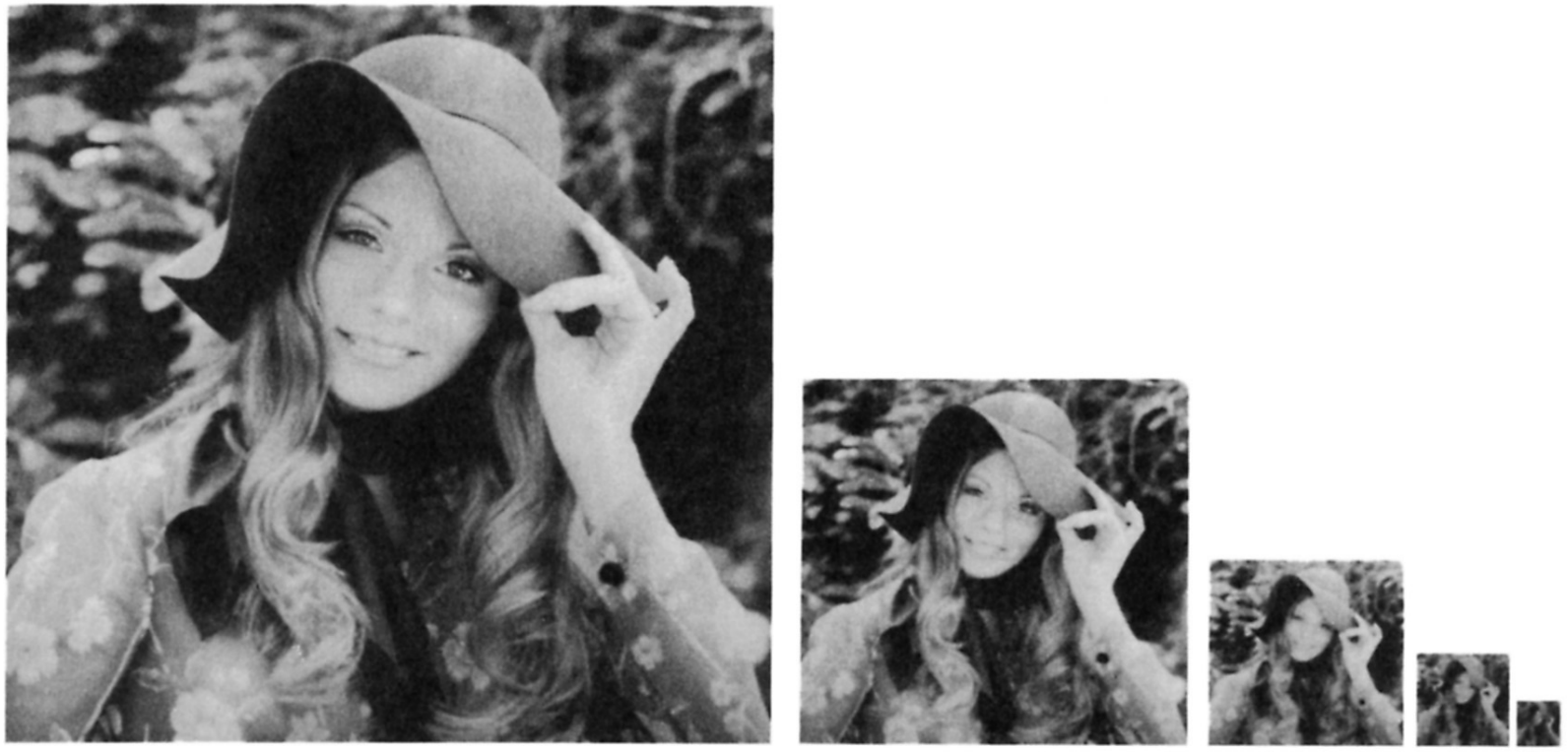


Figure 10.36
Example of the Gaussian pyramid representation for image of 513 x 513 pixels with $K = 4$.

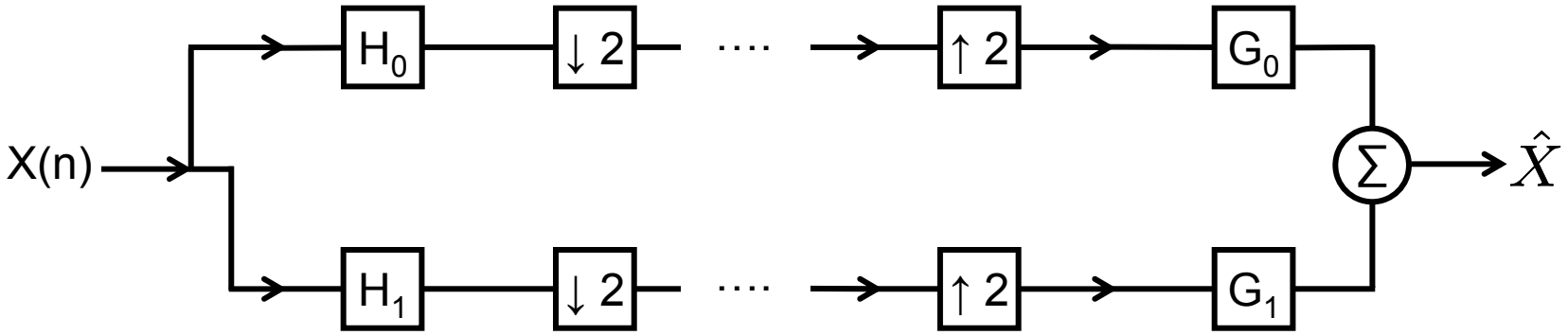
Laplacian Pyramid Representation

Figure 10.38
Example of the Laplacian pyramid image representation with $K = 4$. The original image used is the 513 x 513-pixel image $f_0(n_1, n_2)$ in Figure 10.36 $e_1(n_1, n_2)$ for $0 \leq i \leq 3$ and $f_4(n_1, n_2)$.

Laplacian Pyramid Image Coding

Figure 10.39
Example of the Laplacian pyramid image coding with $K = 4$ at ½ bit/pixel. The original image used is the 513 x 513-pixel image $f_0(n_1, n_2)$ in Figure 10.36.

Subband Coding



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

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

Consider QMF Filters:

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

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

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

Subband Coding (cont.)

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

$$\rightarrow \hat{X}(\omega) = X(\omega) \rightarrow \mathrm{P}erfect \;\; Reconstruction$$

Filter Design

- **QMF filters:** $h_1(n) = (-1)^n h_0(N-1-n)$

N = # of taps

- **Johnston's filter coefficients:** $h_0(N-1-n) = h_0(n)$

$\rightarrow$ Symetric $\rightarrow$ 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 (cont.)

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

  $\rightarrow$ **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]

$\rightarrow$ **Simple to implement**

Proposed by LeGall

Filter Design (cont.)

- **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.

RK = Average # of bits per sample of subband K

M = # of subbands

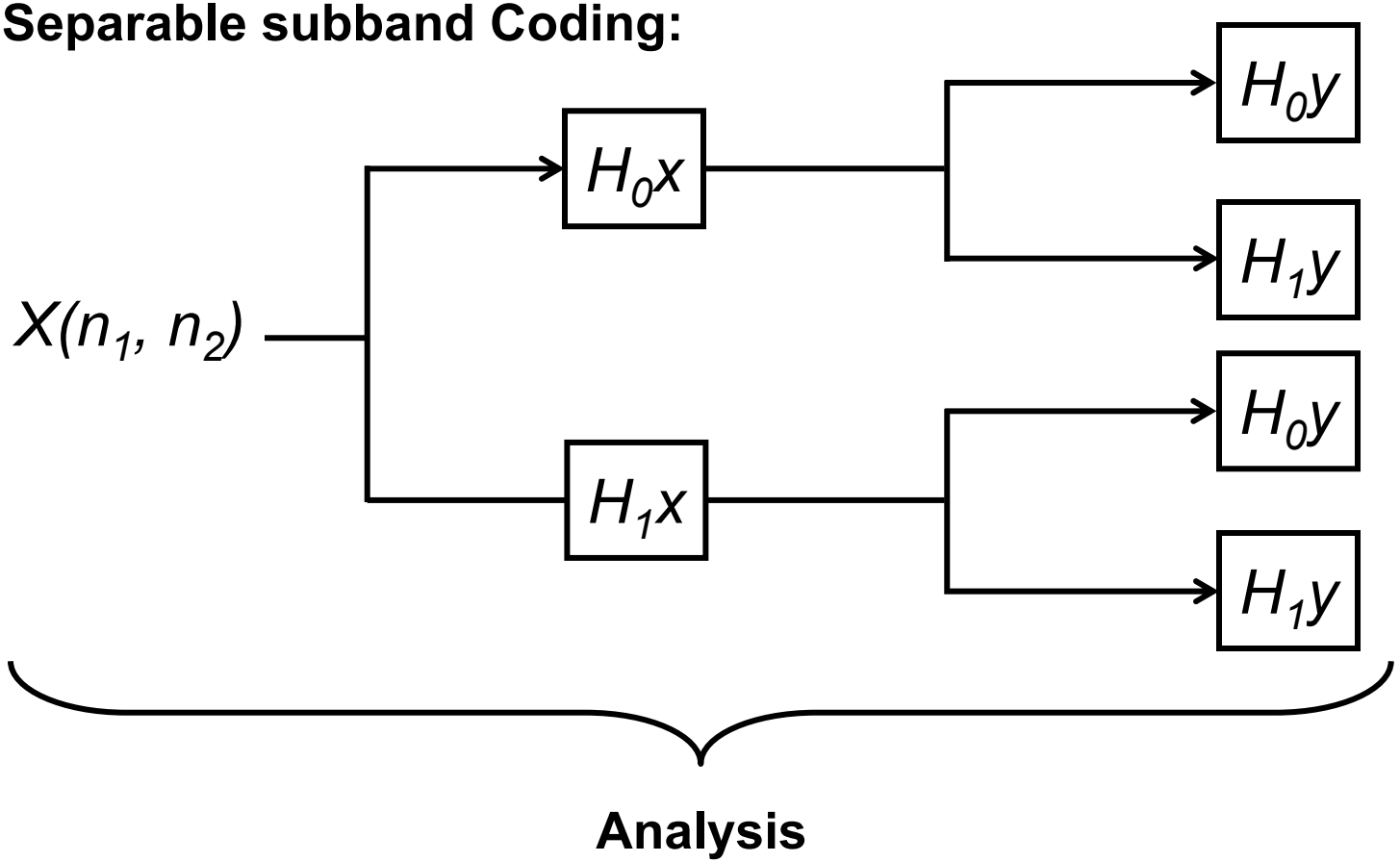
σ_K^2 **= variance of coefficients in subband K:**

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

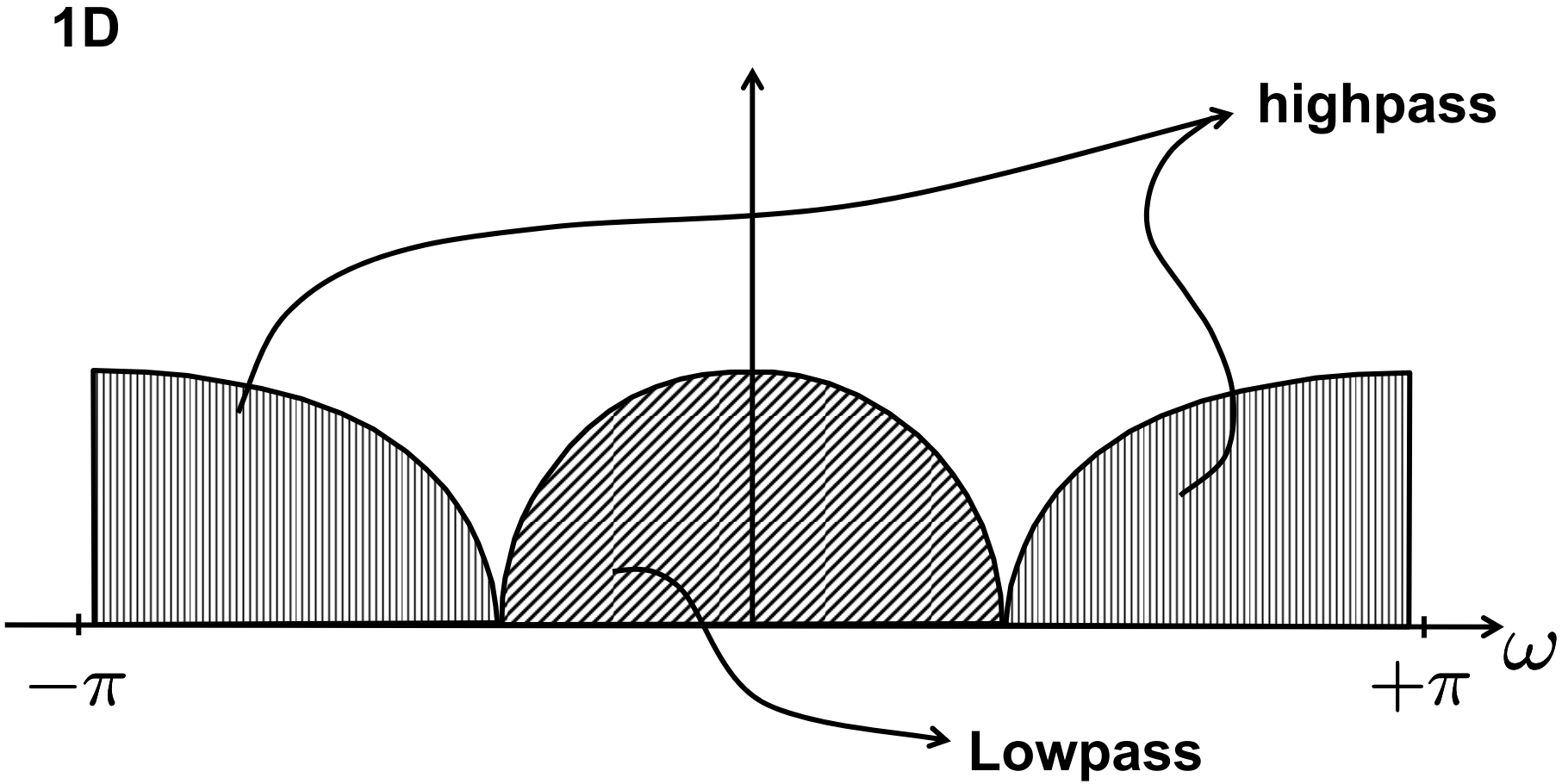
2D Subband Coding

- **Separable $\rightarrow$ Easy to implement**
- **Nonseparable**

Separable subband Coding:

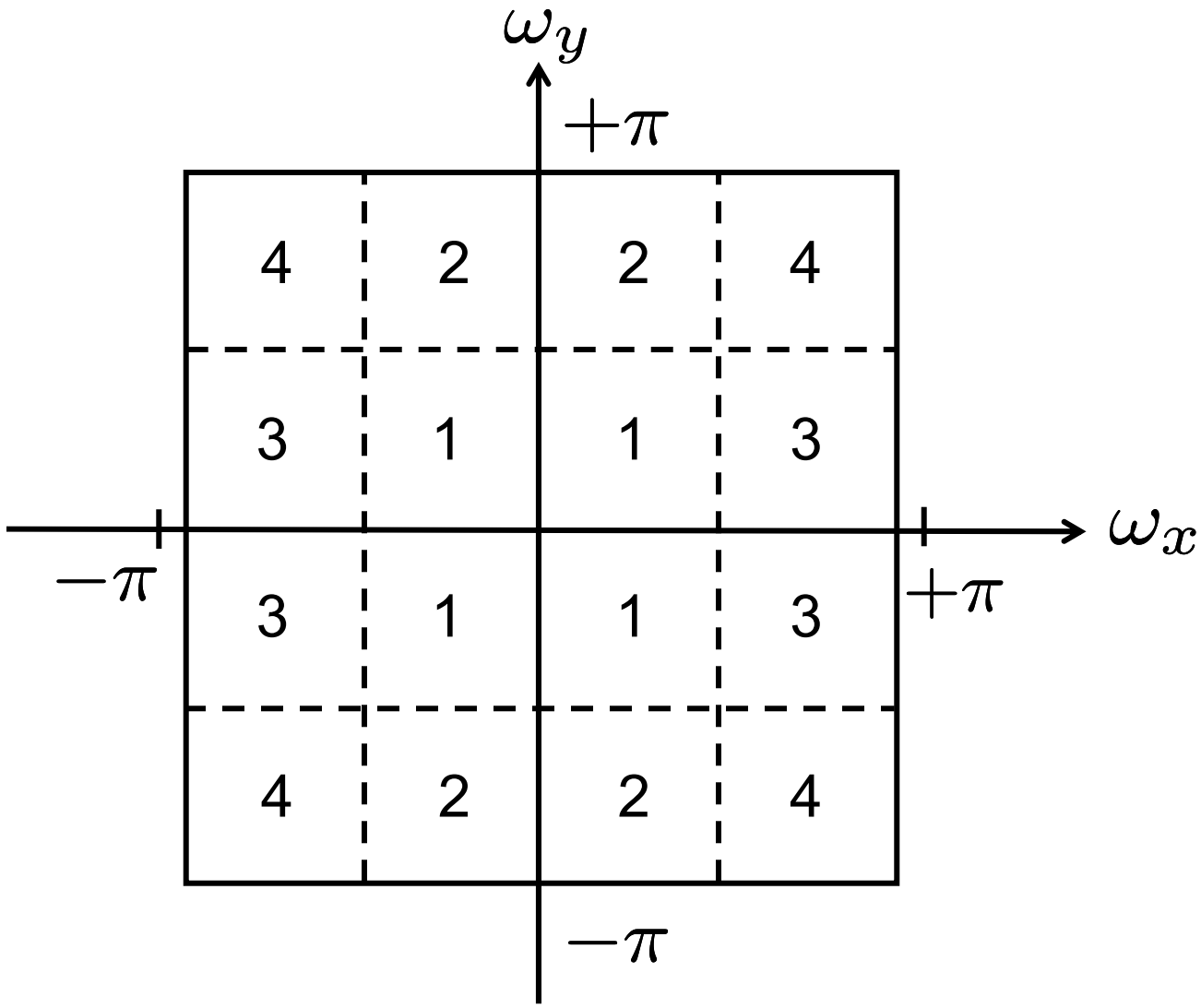


Frequency Domain



Frequency Domain (cont.)

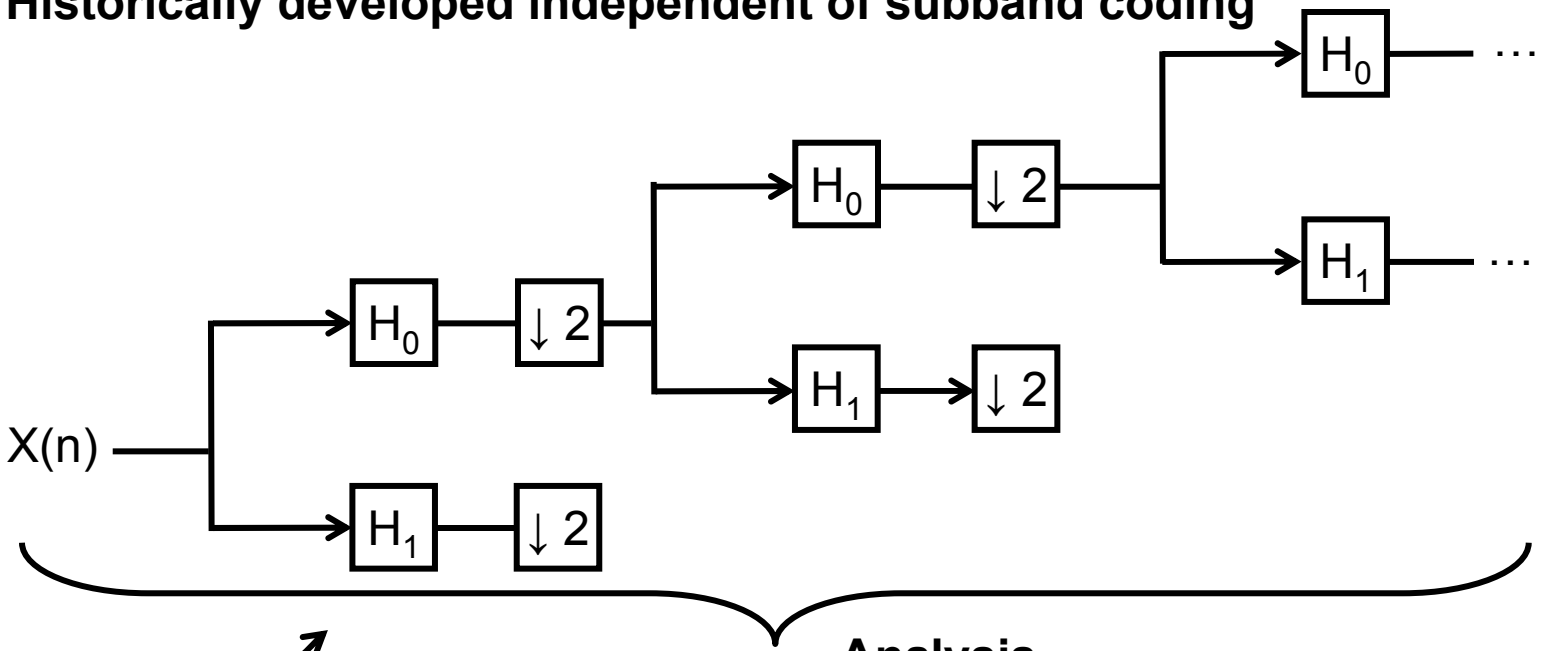
2D



1 = L_xL_y
2 = L_xH_y
3 = H_xL_y
4 = H_xH_y

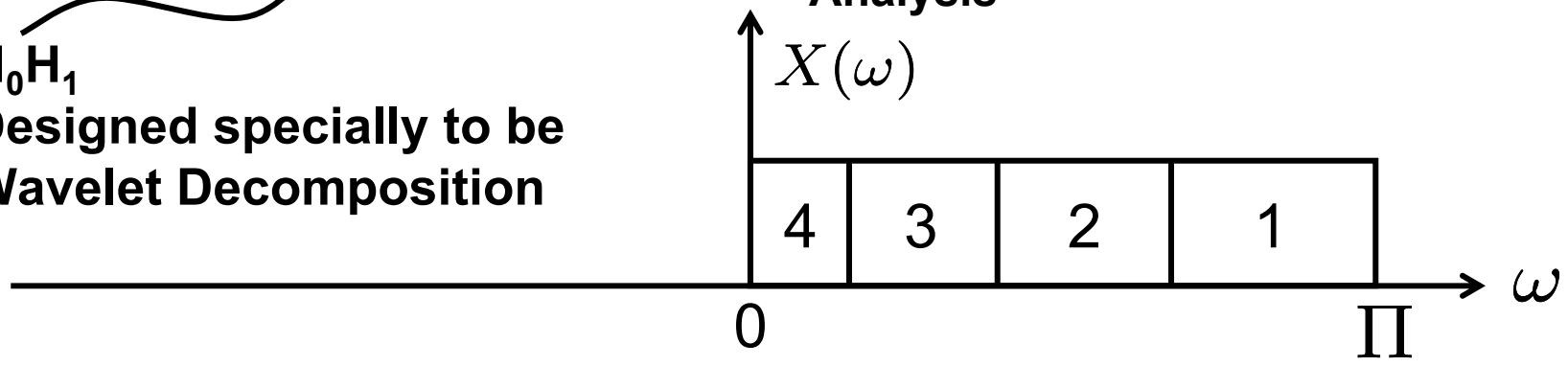
Wavelets

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



H_0H_1
Designed specially to be Wavelet Decomposition

Analysis



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: Manderbroth 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.**

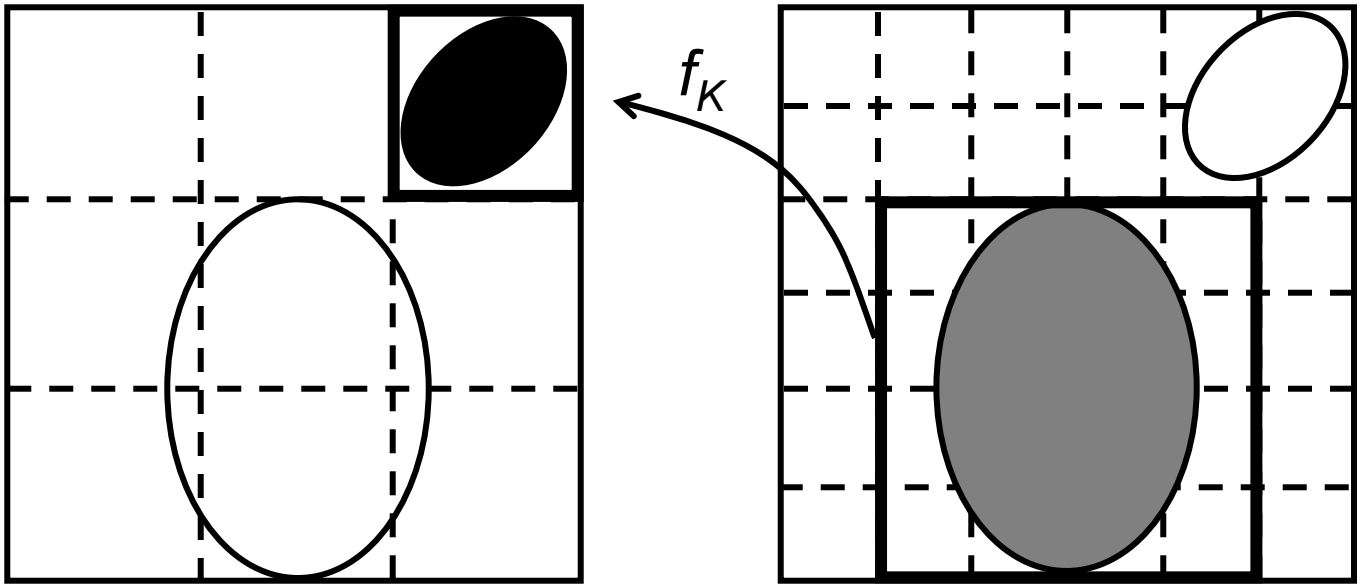
Image Compression (cont.)

- **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.**

Image Compression (cont.)



- What should f_K do?
  - Change size of domain block
  - Change orientation of domain block
  - Change intensity of pixels

Image Compression (cont.)

- f_K consists of
  - Geometric transformation: g_K
  - Massic transformation: m_K
- g_K: displacement + size + intensity
- m_K: orientation

Transformations:

- g_K: displaces + adjusts intensity

  $\rightarrow$ 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

Transformations (cont.):

- If image is divided into
  N Range blocks $\rightarrow$ N transformations f_K k = 1, …N
  are its representation.

$$f = \overset{\cup}{k}\ f_k$$

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

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

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

Insert Fig. 13.11

Vector Quantization

- Let $\vec{f}$ denote *N* dimensional vectors consisting of *N* real valued, continuous amplitude scalars.

- Basic Idea: Map $\vec{f}$ into *L* possible *N* dimensional reconstruction vectors $\vec{r_i}$ for $1 \leq i \leq L$.

- Define a distortion measure:

$$D = E\left[\left(\hat{\vec{f}} - \vec{f}\right)^T \left(\hat{\vec{f}} - \vec{f}\right)\right] = \sum_{i=1}^{L} \int_{\vec{f_0}} \in C_i \left(\vec{r_i} - \hat{\vec{f}}_0\right) d\vec{f_0}$$

Vector Quantization (cont.)

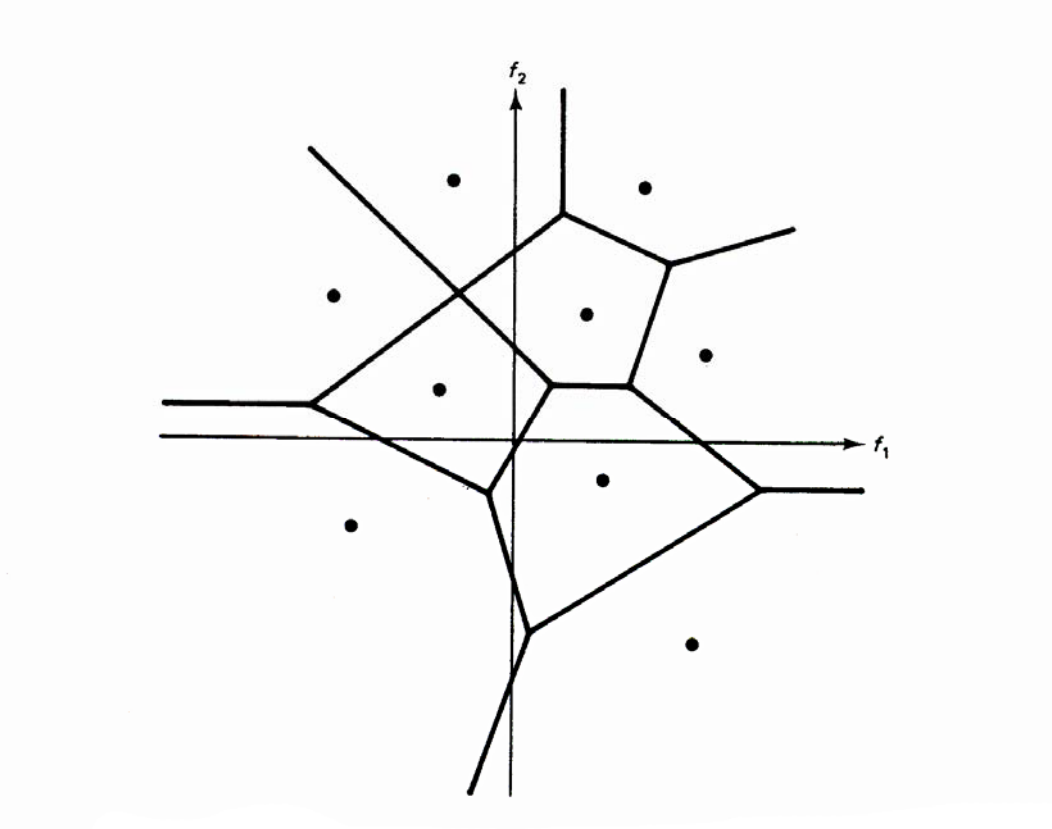


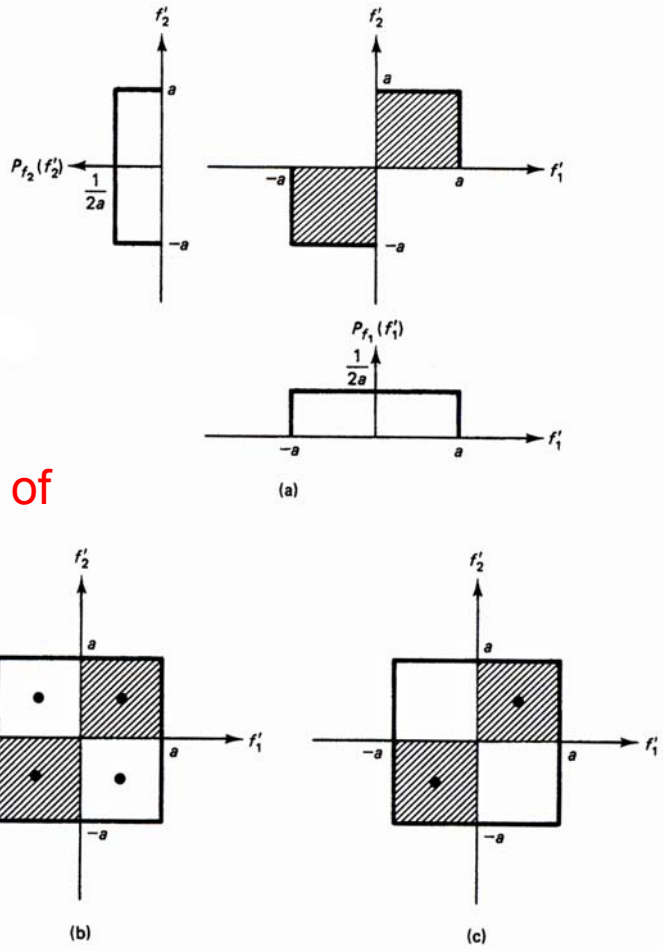
Figure 10.8
Example of vector quantization. The number of Scalars in the vector is 2, and the number of reconstruction levels is 9.

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 levels.

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.

VQ Removes Linear Dependency (cont.)

Some distortion but, reduce # of reconstruction levels.

Rotate (linear transformation) f_1, f_2 to get g_1, g_2 uncorrelated.

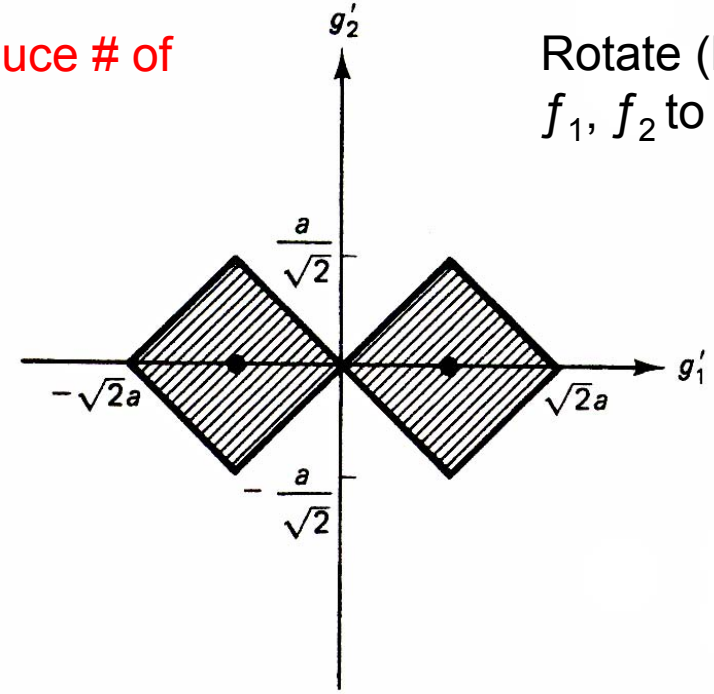
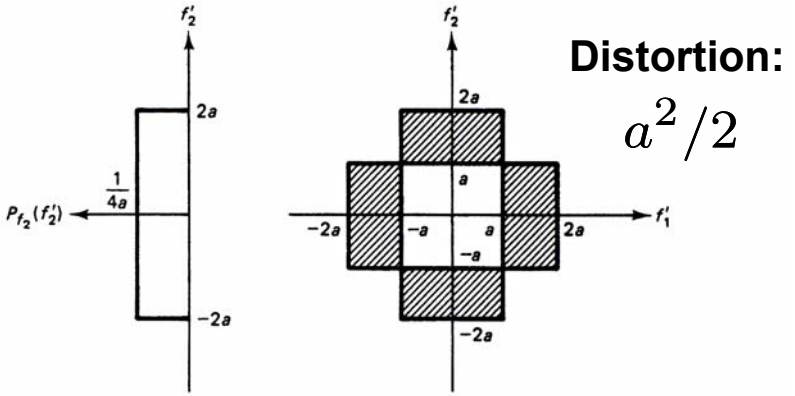


Figure 10.10
Result of eliminating linear dependence of the two scalars f_1 and f_2 in Figure 10.9 by linear transformation of f_1 and f_2.

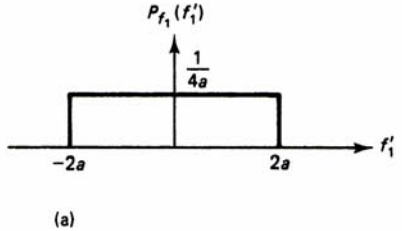
VQ Removes Nonlinear Dependency

- Nonlinear dependence cannot be eliminated by a linear operator.



Distortion:

$a^2/2$

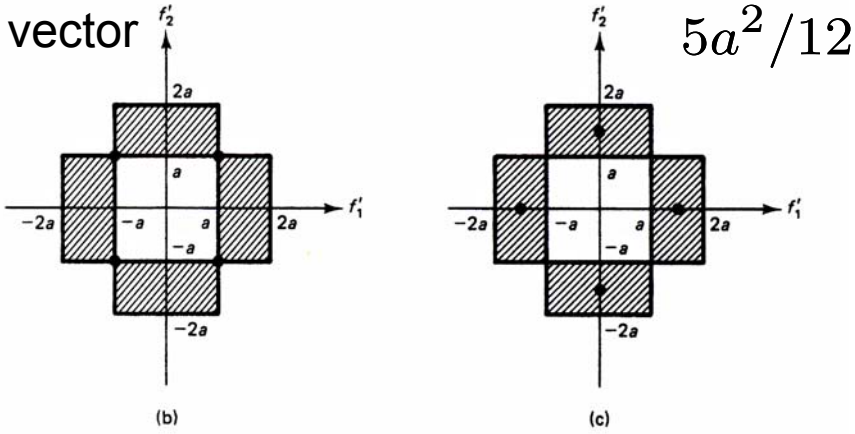


(a)

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.

Distortion:

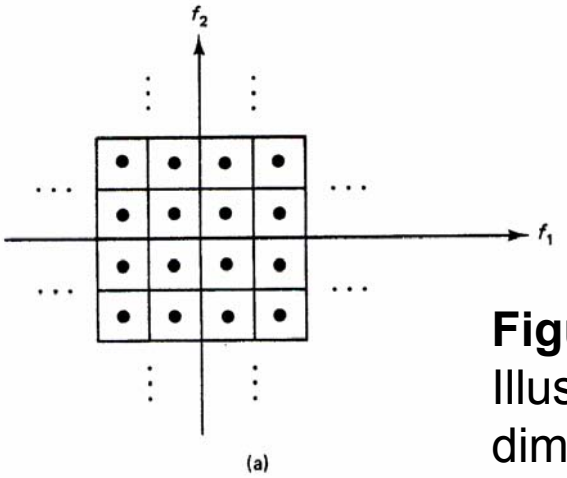
$5a^2/12$



(b) (c)

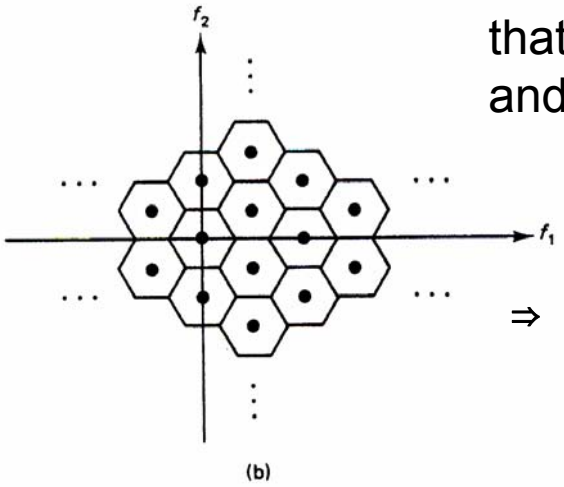
VQ Exploits the Increase in Dimensionality

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



# of reconstruction levels is 2% lower than scalar Quantization with same MSE.

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 bits per scalar with VQ _____ < 1. Look at Figure 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 $\vec{r_i}$, *1 ≤ i ≤ L*, which has the smallest distortion between $\vec{f}$ and $\vec{r_i}$. 10(*f*) = r; i*f f* d(*f*, r_i) < d(*f*,r_g) jf_1
  - Each reconstruction level $\vec{r_i}$ must minimize the average distortion *D* in C_i. Minimize $E\left[d\left(\vec{f},\vec{r_i}\right)\middle|\vec{f}\in C_i\right]$ *w.r.t.* $\vec{r_i}$.
- Find $\vec{r_i}$ and C_i iteratively $\rightarrow$ Problem: local versus global minimum $\rightarrow$ initial guess important.

Codebook Design via K-means (cont.)

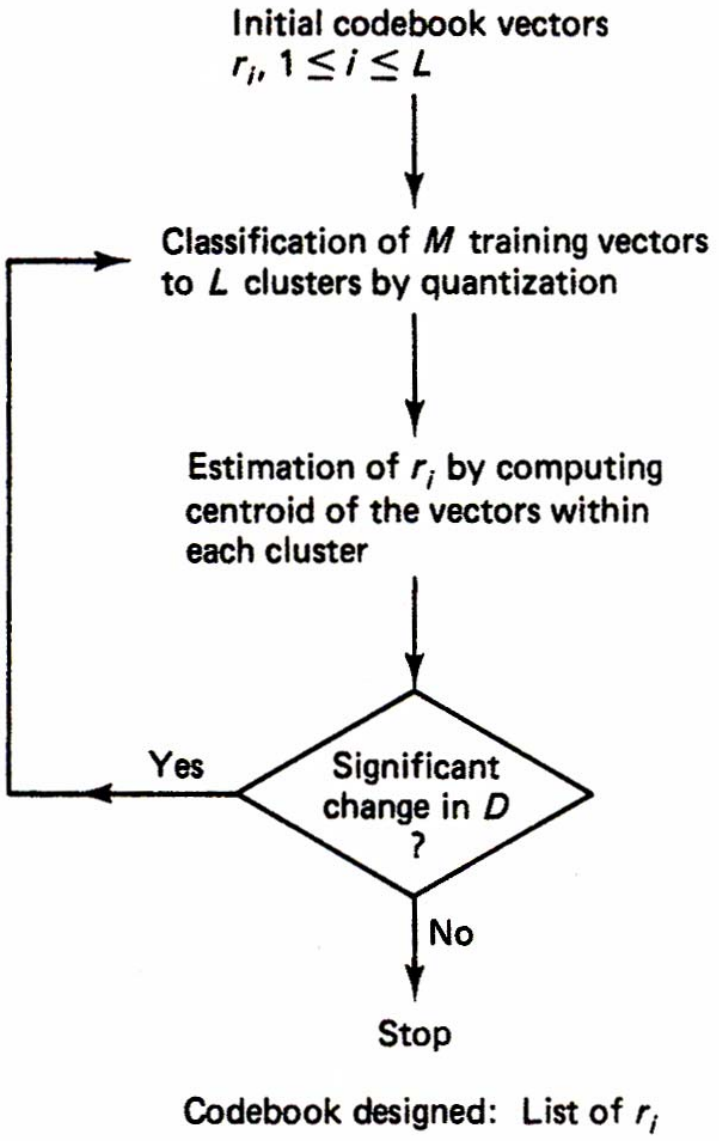


Figure 10.14
Codebook design by the K-means algorithm for vector quantization

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.

    $L = 2^{NR} = 2^{B.}$
  - $MLN = NM2^{NR}$ additions and mults per iteration.
  - Example: $N = 10, R = 2, M = 10L$ results in 100 trillion operations per iteration.
  - Storage: *MN* for training vectors, *LN* for reconstruction levels $\rightarrow$ $(M + 2^{NR})N$.
- **Complexity of operation at the transmitter.**
  - Storage of reconstruction levels: $N2^{NR} = NL$. If $N = 10$ and $R = 2$, storage is 10 million.
  - Number of arithmetic operations $N2^{NR} = NL$. 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 transmitter $\rightarrow$ 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.

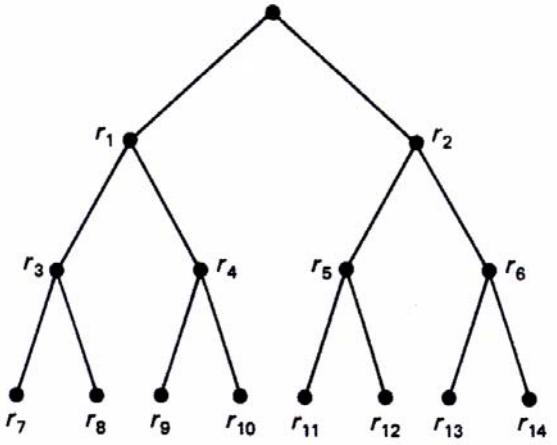


Figure 10.15
Example of a tree codebook.

Complexity of Tree Codebook

- **Terms:** ***M* = # *of training vectors*, *L* = # *of codewords and* *N* = *Dimension*.**
- **Design complexity:**
  - **Number of arithmetic operators per iteration is $2NM\log_2 L$, where the 2 is the distortion measure evaluated twice and $\log_2 L$ is the # of stages. 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. (Storing training data dominates.)**
- **Operation complexity at transmitter:**
  - **Number of arithmetic operations is $2N^2R = 2N\log_2 L$. 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 construction 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.**

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.**

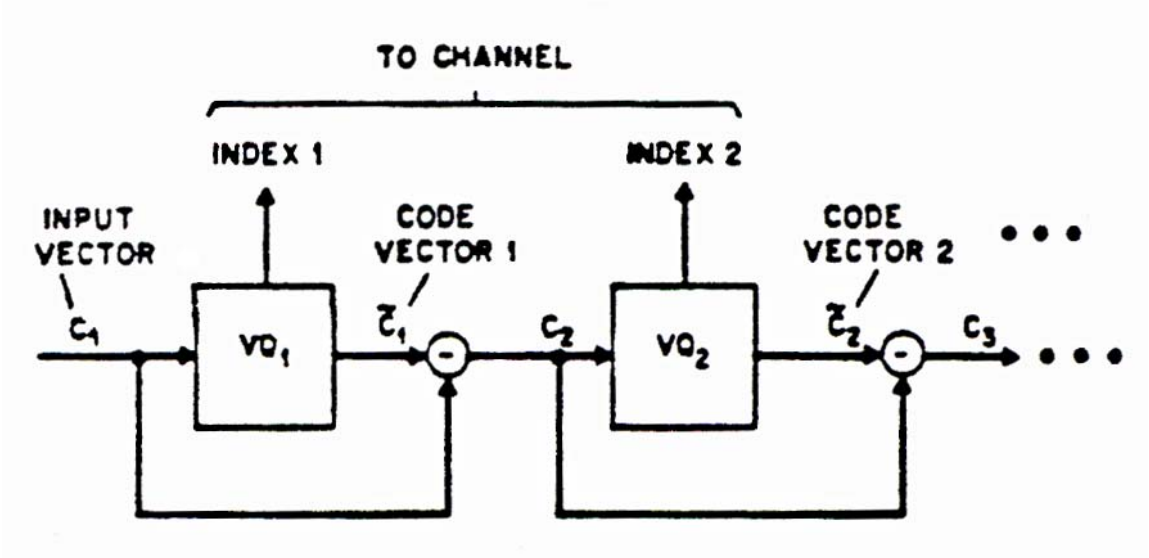


Figure 5.5.2
Multistage Vector Quantization. At each state an error vector is computed, which is then used as the input to the next stage of VQ. The decoder merely computes a summation of the code vectors corresponding to the received indices.

Variations of VQ (cont.)

- **Parameter extraction techniques:**
  - Mean and variance of each input vector are computed and sent separately.
  - Mean and variance might be coded with DPCM
- **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 x 4 pixels as vectors.**

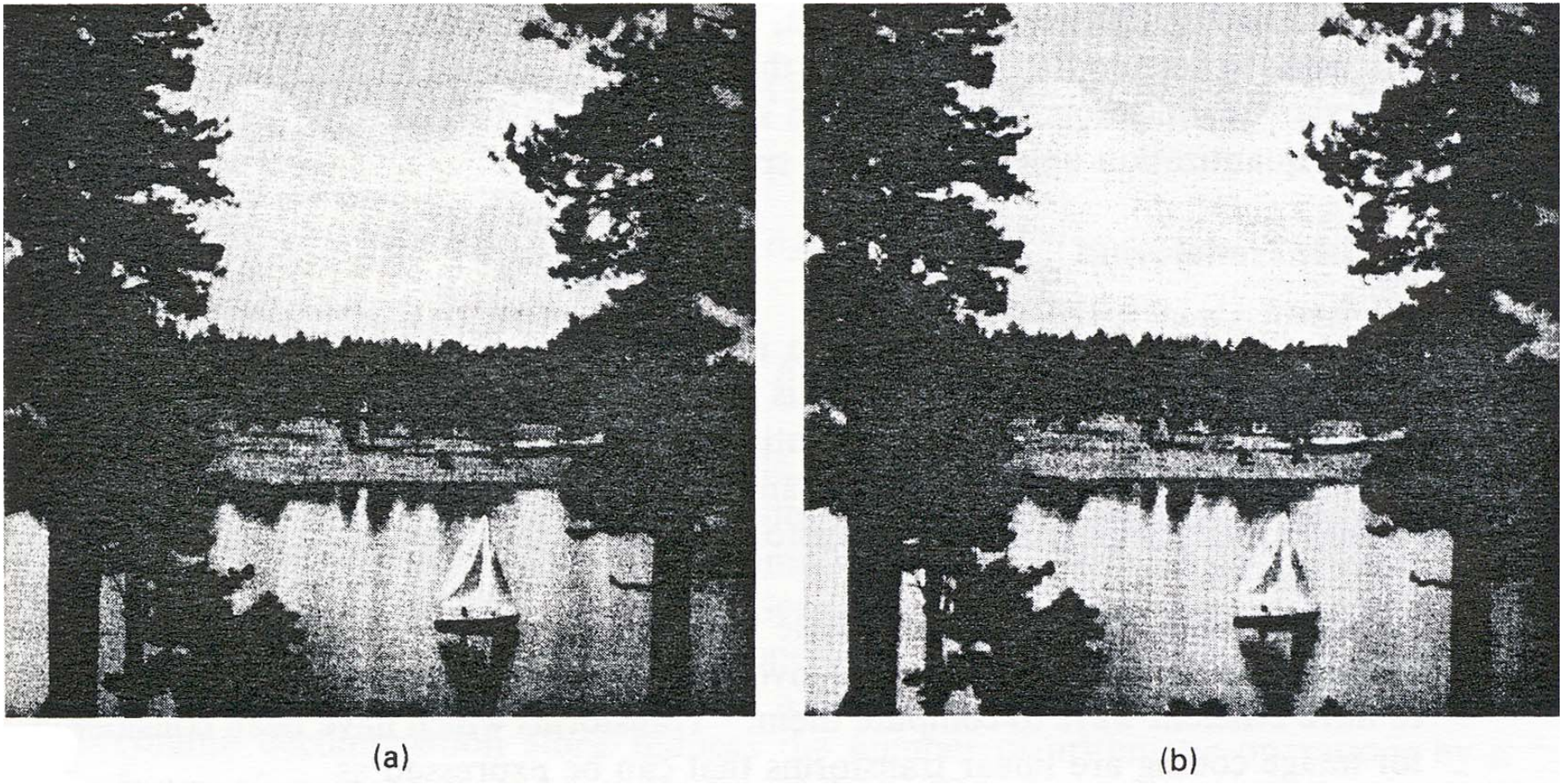


Figure 10.40
Example of an image coded by vector quantization. Courtesy of William Equitz. (a) Original image of 512 x 512 pixels; (b) coded image by vector quantization at ½ bit/pixel. The block size used is 4 x 4 pixels and the codebook is designed by using a variation of the K-means algorithm. NMSE = 2.7% , SNR = 15.7 dB.

Second Generation Image Coding

- Exploits the fact that images consist of distinct objects with well defined and abrupt boundaries with textured interiors.
- **Basic Idea:** Decompose a single image into edges and texture.

$$B(x,y) = d(x,y) + r(x,y)$$

$d(x,y)$ is an image containing basically objects with their interior texture removed. $r(x,y)$ is a remainder image containing texture, surface roughness and other irregularities of the object interiors.

- Use contour coding for the shape of $d(x,y)$. Send coarsely quantized amplitudes of $d(x,y)$ separately.
- Use transform coding type techniques for $r(x,y)$. Or, approximate the regions inside the boundaries with polynomials.

Second Generation Image Coding (cont.)

- If the smallest allowable object is too small, then $d(x, y)$ requires too many bits.
- If only large objects are allowed, then $r(x, y)$ will require too many bits.
- Potential for 100:1 compression ratio.

Comparison of Image Coding Methods

	Waveform Coding	Transform Coding	Image Model Coding
Performance (bits/pixel)	Worst	Very good	Best (potentially)
Computations	Best	Good	Worst

Interframe Image Coding

- **Similar to intraframe techniques.**
- **Extend DPCM from 2-D to 3-D:** Predict each pixel in frame ($n + 1$) from the neighboring ones in frame n and $n + 1$.
- **Extend 2-D DCT to 3-D DCT:** Set high frequency temporal coefficients to zero $\rightarrow$ Impractical due to storage requirements and delay.
- **Hybrid Transform / Waveform Coding:**
  - Compute a 2-D Transform for each frame.
  - Apply a waveform coder such as DPCM along the temporal direction.

Motion Estimation

- **Applications of motion estimation:**
  - Commercial Problems: Bandwidth compression of TV conferencing and picture phone video signals.
  - Industrial Problems: Dynamic monitoring of industrial processes. Dynamic robot vision.
  - Medical Problems: Study of heart motion from X-ray movies.
  - Meteorology: Cloud tracking.
  - Transportation: Highway traffic monitoring.
- **Approaches to motion estimation:**
  - Region matching methods.
  - Recursive methods.
  - Transform domain methods.
  - Method of differentials.

Region Matching Technique

- Choose the two dimensional vector $\vec{d}$ in order to minimize the cost function given by:

$$C\left(\vec{d}, \vec{x_0}, t_0\right) = F\left[E\left(\vec{x_0}, t_0\right), E\left(\vec{x_0} - \vec{d}, t_0 + \delta t\right)\right]$$

- *F[.]* is a function measuring the similarity between two frames displaced with respect to each other.
- $E\left(\vec{x_0}, t\right)$ is the intensity of the frame at time t and location $\vec{d}$.
- Objective: Search over a 2-D space to find $\vec{d}$ to minimize the cost function at $\left(\vec{x_0}, t_0\right)$
- Options for cost function:
  - Cross correlation.
  - Mean squared error.

Region Matching Technique (cont.)

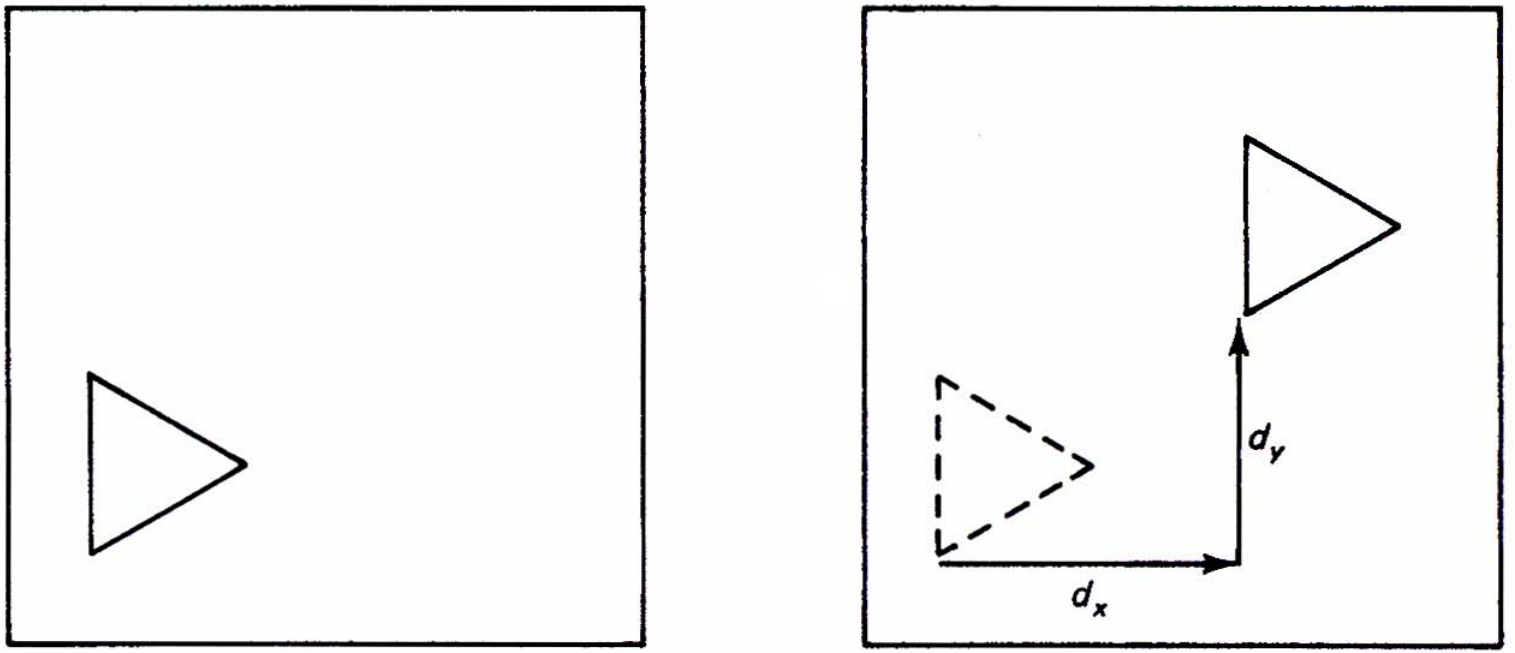


Figure 8.43
Image translated with displacement of (d_x,d_y) (a) $f(x, y, t_{-1})$; (b) $f(x, y, t_0)$.

Cross Correlation Method

- Define cross correlation between frame *k* and *k – 1*:

$$R_{S_k S_{k-1}}^{(d_x, d_y)} = E\left[s_k(x, y)\, s_{k-1}(x - d_x, y - d_y)\right]$$

- Define *NCCF* to be the normalized cross correlation function:

$$NCCF\left(\vec{d}\right) = \frac{R_{S_k S_{k-1}}(\vec{d})}{\sqrt{R_{S_k S_k}\left(\vec{0}\right) R_{S_k S_{k-1}}\left(\vec{0}\right)}}$$

- Basic Idea: Position of the correlation peak is the displacement estimate.
- Let d_{max} denote the maximum horizontal or vertical displacement pixels.
- Search for the correlation peak requires an evaluation of NCCF at $(2d_{max} + 1)^2$ different horizontal and vertical shifts. $\rightarrow$ too many operations.

Logarithmic Search

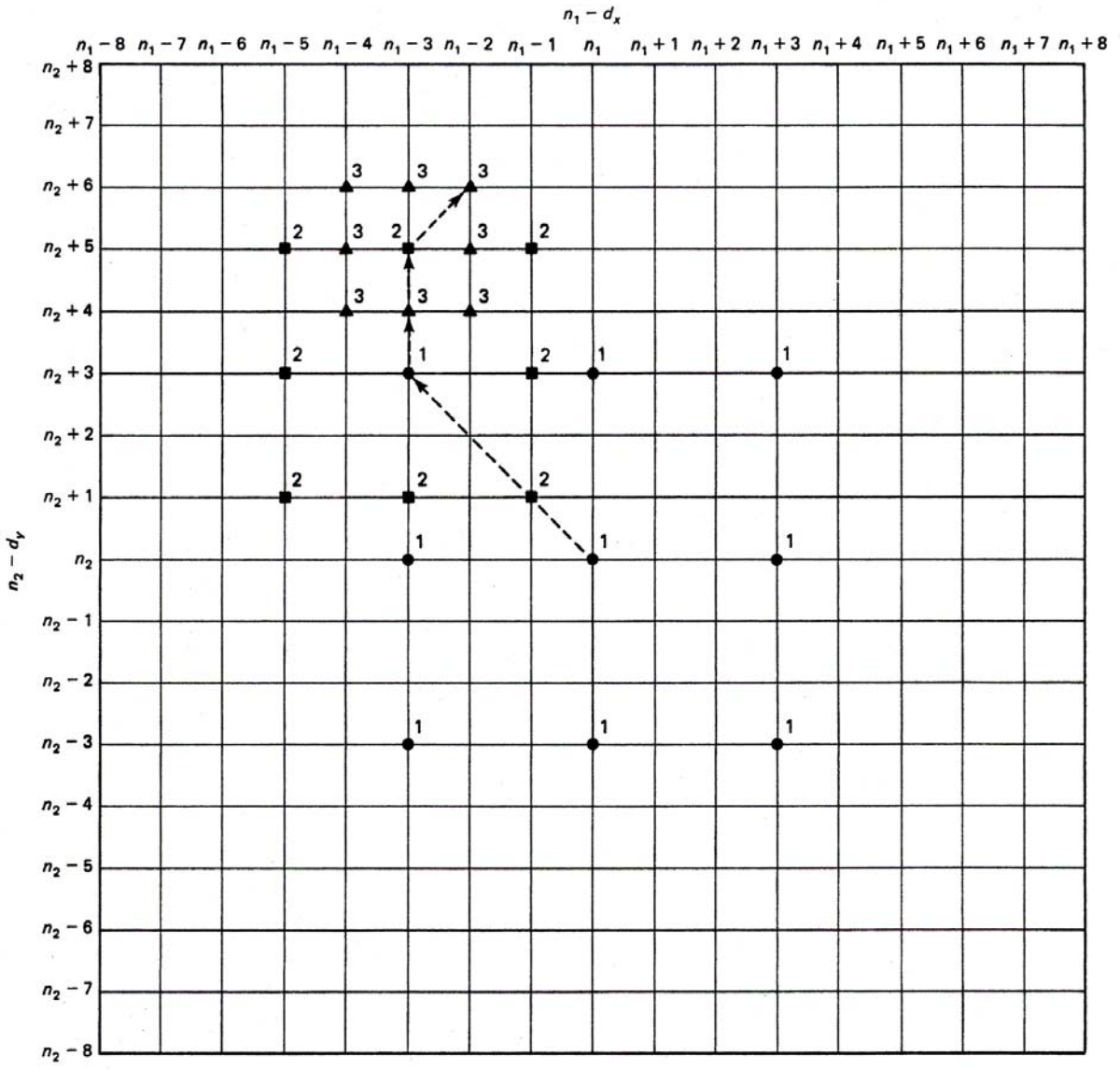


Figure 8.44
Illustration of three-step search method.

MSE as Region Matching Criterion

- Jain and Jain have proposed the MSE criterion defined by:

$$MSE(i,j) = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \left[S_k(m,n) - S_{k-1}(m+i, n+j) \right]^2$$

- Too many matchings $\rightarrow$ too many operations needed to estimate displacement.
- 3 step search algorithm $\Rightarrow$ logarithmic.
- Hierarchical
- Cost Comparison: 720 x 480, 30 fps

  Search range ± 15

  Full Search 30 GOPS

  Logarithmic 1 GOPS

  Hierarchical ½ GOPS
- Must sacrifice accuracy if give up exhaustive search.

Recursive Motion Estimation

- Basic Idea: Minimize the MSE criterion via steepest descent or other recursive methods.
- Let $\left(\hat{d}_x(k), \hat{d}_y(k)\right)$ denote the estimate of (d_x, d_y) after the kth iteration:

$$\hat{d}_x(k+1) = \hat{d}_x(k) - \epsilon \frac{\partial Error(d_x, d_y)}{\partial d_x}\Big|_{(d_x, d_y)} = \left(\hat{d}_x(k), \hat{d}_y(k)\right)$$

$$\hat{d}_y(k+1) = \hat{d}_y(k) - \epsilon \frac{\partial Error(d_y, d_y)}{\partial d_y}\Big|_{(d_x, d_y)} = \left(\hat{d}_x(k), \hat{d}_y(k)\right)$$

- ε is the step size and error is the MSE.

Recursive Motion Estimation (cont.)

- **Observations:**
  - Evaluation of partial derivatives is noisy $\rightarrow$ Presmoothing helps.
  - Can potentially achieve subpixel accuracy $\rightarrow$ need to estimate the derivatives at subpixel accuracy $\rightarrow$Interpolation is needed.
  - Iteration can be repeated many times over the same pixel or it can be used only once for a pixel and then move on to the next pixel.

Frequency Domain Methods for Motion Estimation

- **Basic Idea:** Fourier transform of the shifted version of *E(x,y)* is given by:

  $$E(x - d_x, y - d_y) \Longleftrightarrow S(w_x, w_y)\, exp\left[-j2\pi\left(w_x d_x + w_y d_y\right)\right]$$

  $S(w_x, w_y)$ is the Fourier Transform of *E(w,y).*
- Solve a linear system of equations corresponding to different (w_x, w_y) to find d_x and d_y.
- Shortcomings: Only applies to the case where all the objects move in the same direction and by the same amount against a uniform background.
- Involves phase unwrapping $\rightarrow$ computationally expensive.

Method of Differential for Motion Estimation

- **Basic Idea: Brightness Constraint Equation:**

$$u\frac{\partial E}{\partial x} + v\frac{\partial E}{\partial y} + \frac{\partial E}{\partial t} = 0$$

- *u* and *v* denote the components of motion along *x* and *y* directions.
- **Comments:**
  - Computing derivatives is noisy particularly when there is aliasing due to temporal under sampling.
  - Brightness constraint equation only determines the component of motion perpendicular to the edges.
  - Determination of the component of motion parallel to the edges is an ill conditioned problem. $\rightarrow$ Not a shortcoming of the method.

Method of Differential for Motion Estimation (cont.)

- **Ways to overcome ill posedness:**
  - Assume a whole block has moved with the same velocity and has edges both along and perpendicular to the direction of motion.
  - Regularize the ill conditioned problem by introducing smoothness constraints. Define:

$$\epsilon_b = u\frac{\partial E}{\partial x} + v\frac{\partial E}{\partial v} + \frac{\partial E}{\partial t}$$

$$\epsilon_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2$$

Minimize:

$$\epsilon^2 = \int\int \left(\alpha^2\epsilon_c^2 + \epsilon_b^2\right) d_x d_y$$

Color Image Coding

RGB → YIQ

$$\begin{bmatrix} Y(n_1,n_2) \\ I(n_1,n_2) \\ Q(n_1,n_2) \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \bullet \begin{bmatrix} R(n_1,n_2) \\ G(n_1,n_2) \\ B(n_1,n_2) \end{bmatrix}$$

Y: most of the energy is compacted here

I, Q: **relatively small amount of energy**

- **Requires about 50% but rate relative to Y component.**

Black and white image	**Color image**
Y bits/pixel	**≈ 1,5 y bits/pixel**

$$\begin{bmatrix} \hat{R}(n_1,n_2) \\ \hat{G}(n_1,n_2) \\ \hat{B}(n_1,n_2) \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & 1.703 \end{bmatrix} \bullet \begin{bmatrix} \hat{Y}(n_1,n_2) \\ \hat{I}(n_1,n_2) \\ \hat{Q}(n_1,n_2) \end{bmatrix}$$

Figure 7.9
Y, I and Q components of the color image in Figure 7.8(d). (a) Y component; (b) I component; (c) Q component.

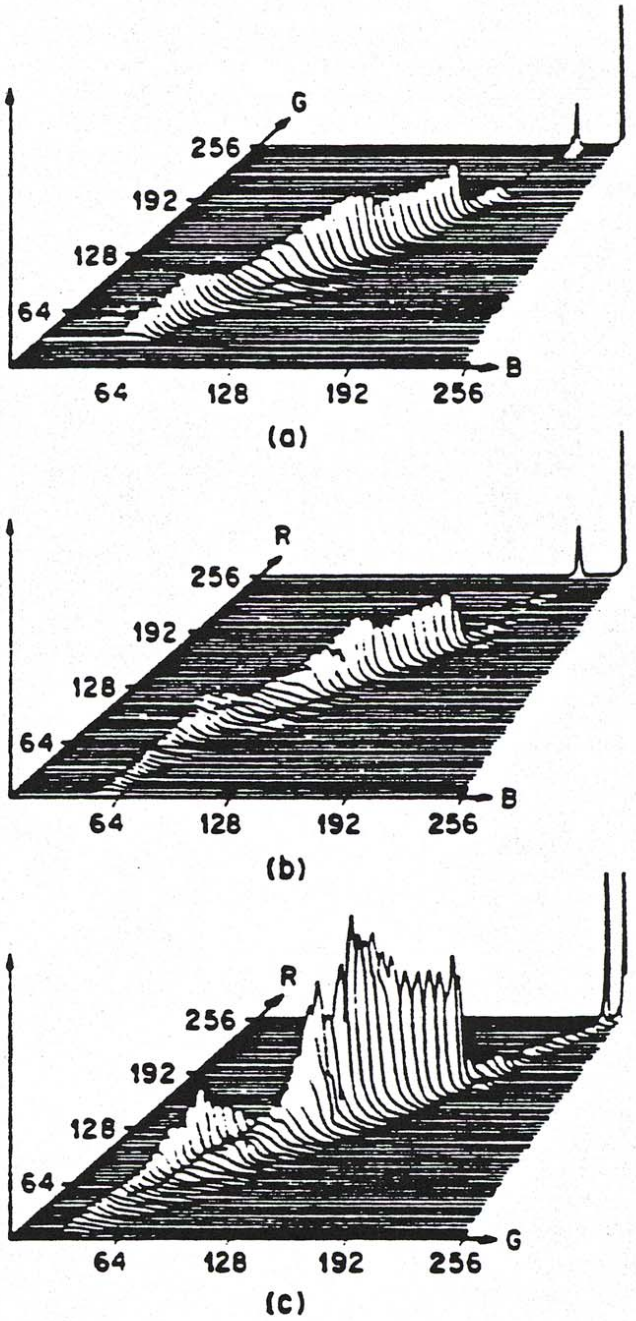
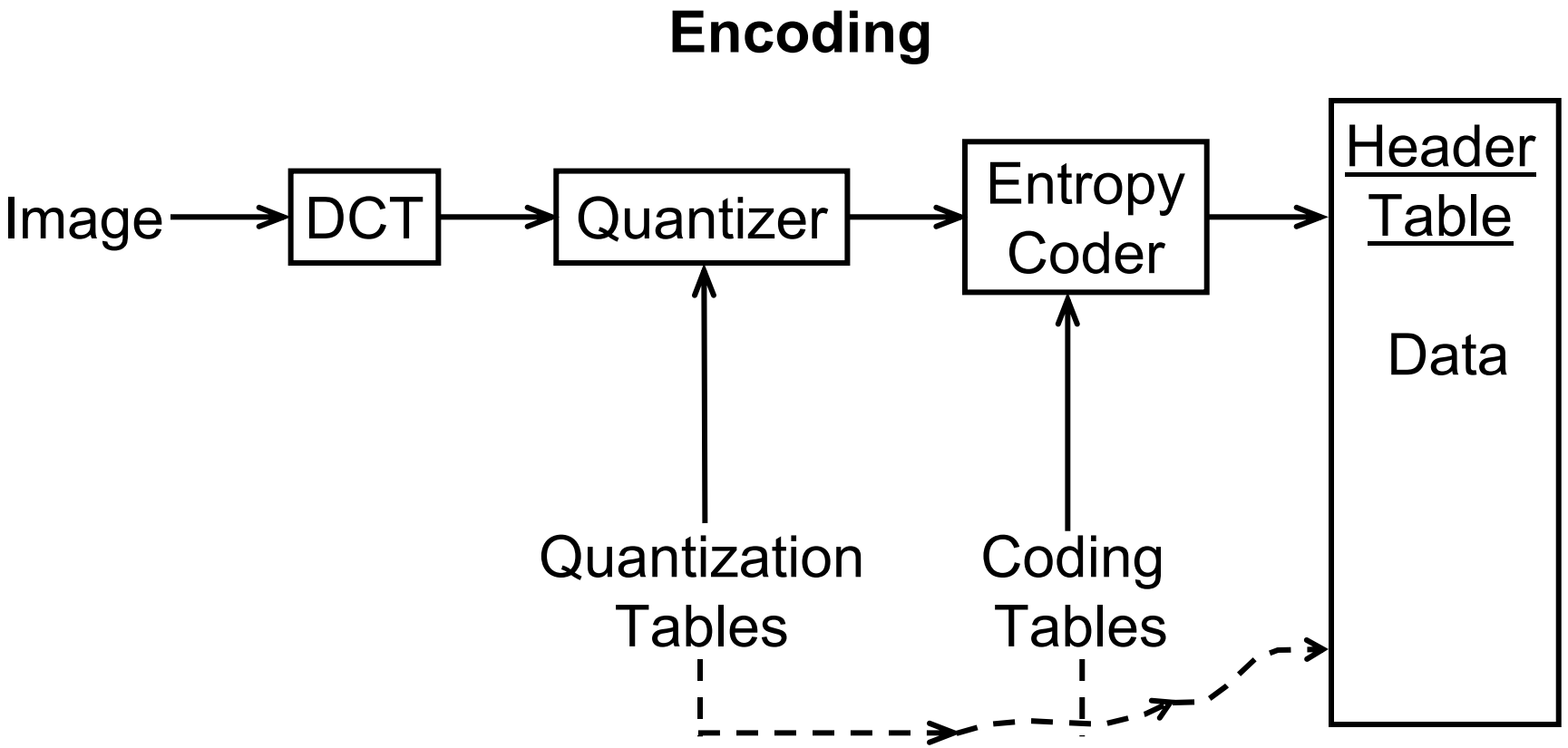


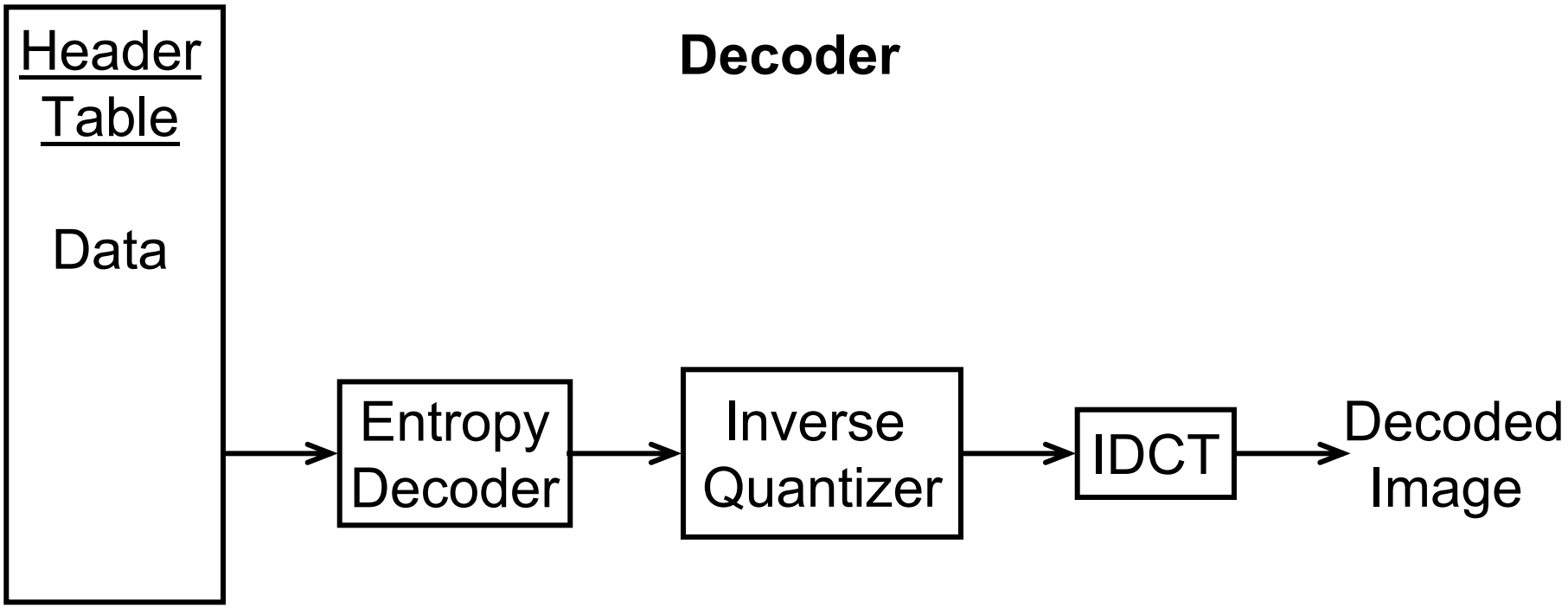
Figure 3.3.3
Two-dimensional joint probability densities of the three-dimensional RGB signal, for a test picture, showing the strong correlation between components. (a) B vs. G, (b) R vs. B, and (c) R vs. G (from frei et al. [3.3.9]).

JPEG Standard

- Joint ITU ISO standard
- Still image compression



JPEG Standard (cont.)



JPEG

- Compressing color images
- Can compress R, G, B separately
- More efficient to reduce correlation

  convert RGB to YUV.
- How to Design Quantization Tables?
  - * Psychovisual Experiments
  - * Bit Rate Control

$$b_{ij} = r + \frac{1}{2} log_2 \frac{\sigma_{i,j}^2}{\left\{ \prod_{i,j} \sigma_{i,j}^2 \right\}^{\frac{1}{64}}}$$

8 x 8 blocks

r = total # of bits

b_{ij} = bits to (i, j) coefficient

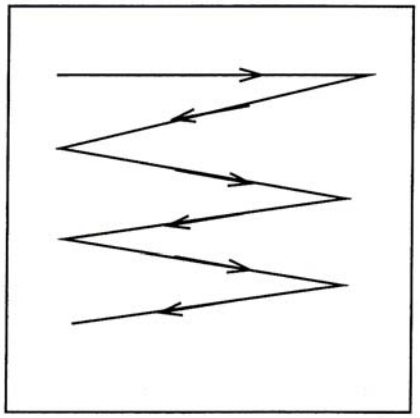
σ_{ij}^2 = variance of b_{ij}

Entropy Coding

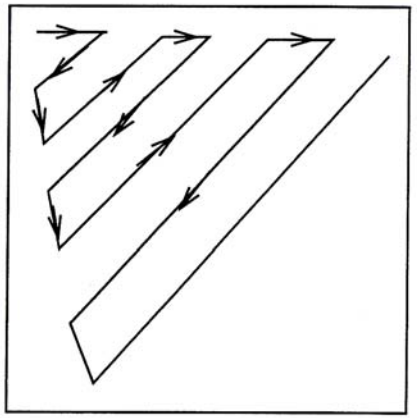
- DC coefficients:

  Difference between DC coefficients of neighboring blocks is quantized + coded.

- AC coefficients:
  - Scan coefficients
  - Threshold to set some to 0
  - Run length code position
  - Quantize + Huffman code amplitude



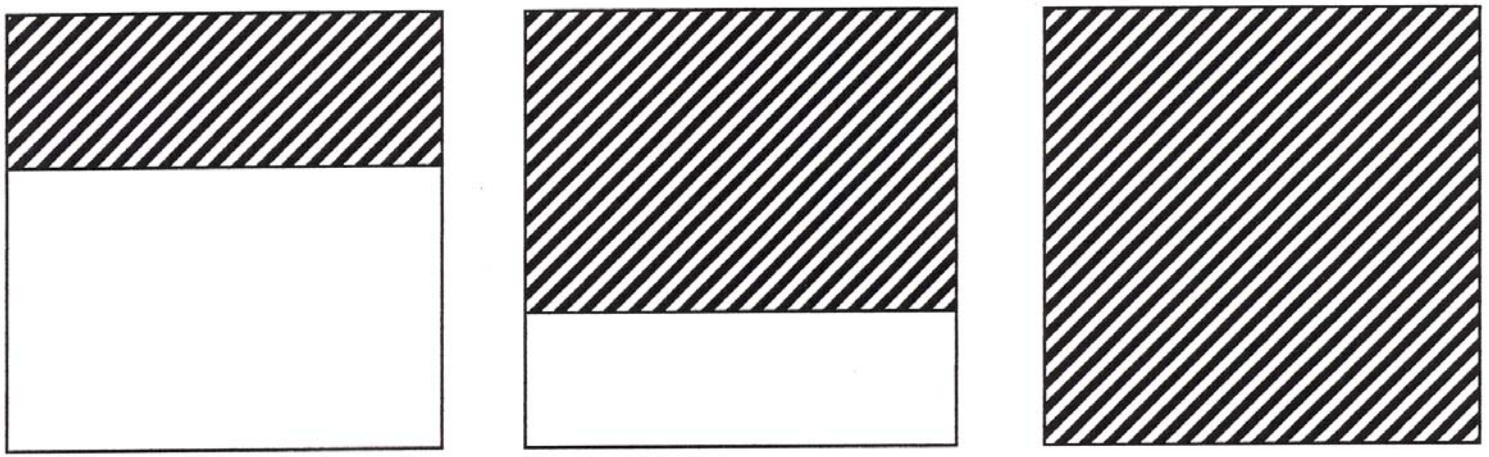
conventional



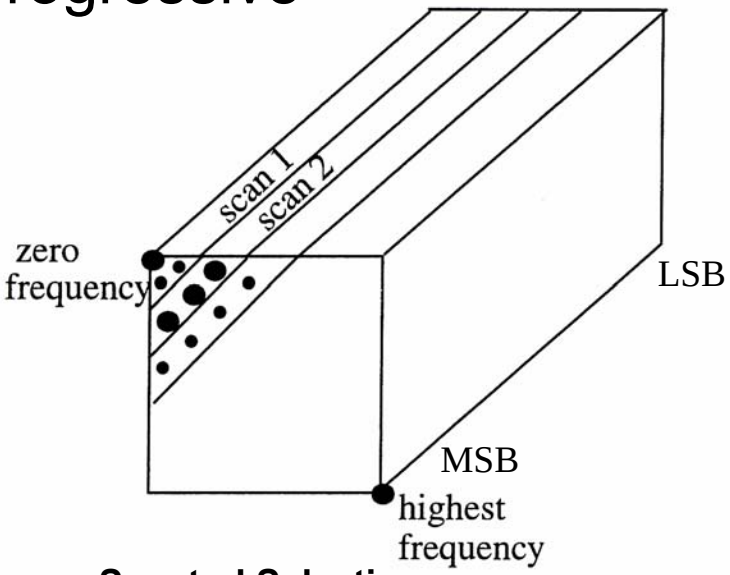
zig - zag order

JPEG MODES

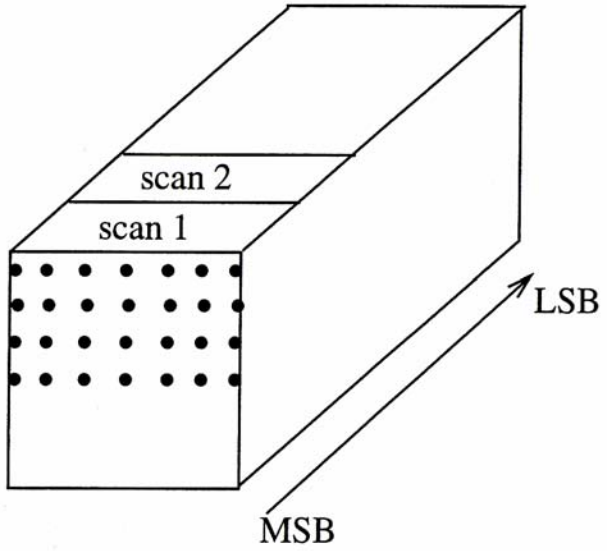
1. Sequential → most common



2. Progressive



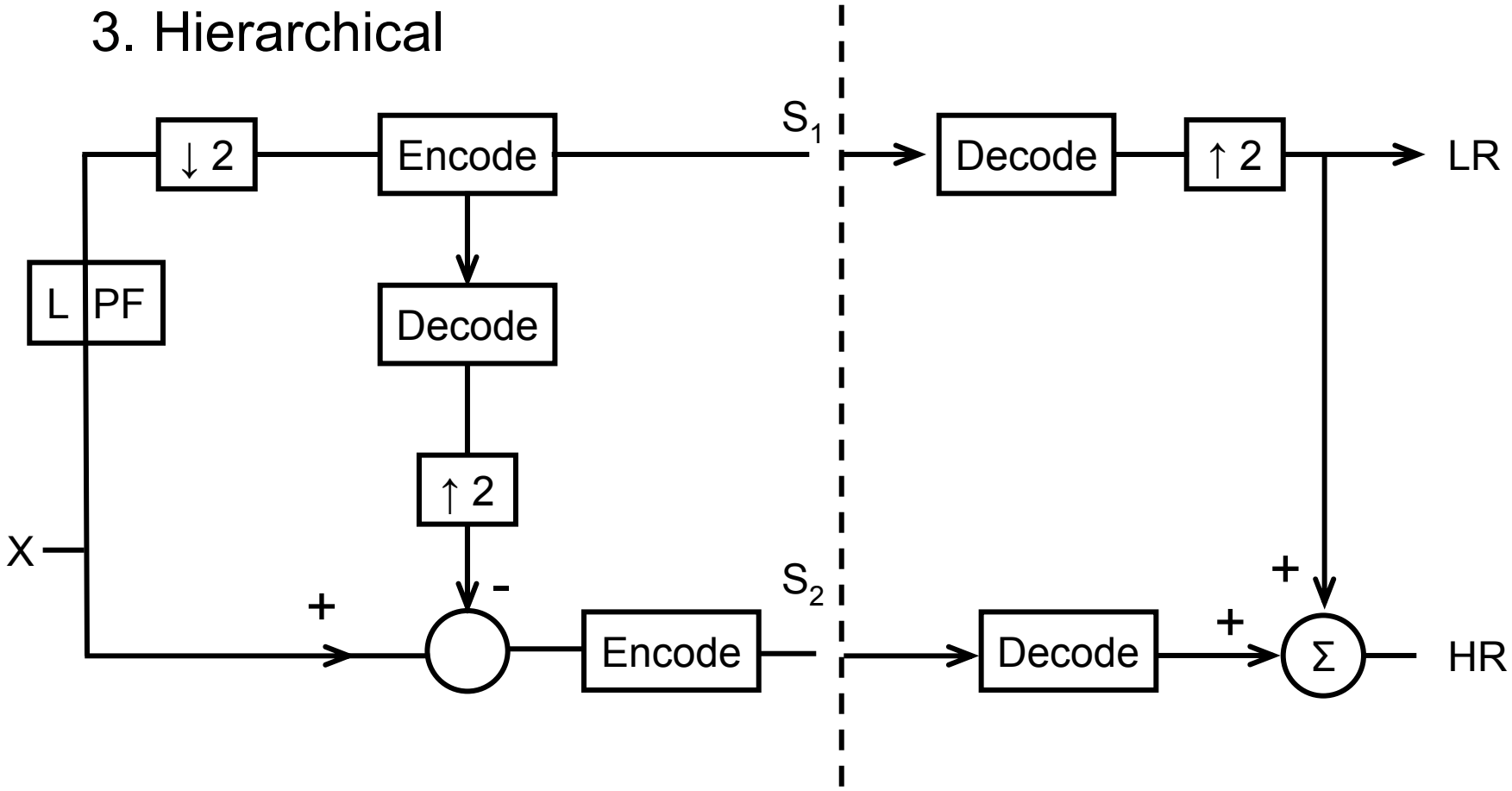
Spectral Selection



Successive Approximation

JPEG MODES (cont.)

3. Hierarchical



LR = Low Resolution

HR = High Resolution

MPEG Standard

- ISO standard 11172 = MPEG 1.
- MPEG = Moving Pictures Expert Group.
- MPEG 1:

  Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mb/s.
- ISO standard 13818 = ITU H.262.

  “Generic coding of moving Pictures and associated audio” = MPEG 2.
- Motivation for MPEG 2: Higher bit rate, greater input flexibility.
- MPEG 4 $\rightarrow$ very low bit Rate 1998.

MPEG

- Only specifies the syntax of the coded bit stream
- Lots of room for flexibility and optimization.
- MPEG standards are application independent, BUT
  - MPEG 1 $\rightarrow$ CDROM
  - MPEG 2 $\rightarrow$ Digital TV
- Many parts to standard
  - Systems
  - Audio
  - Video $\leftarrow$
  - Conformance testing

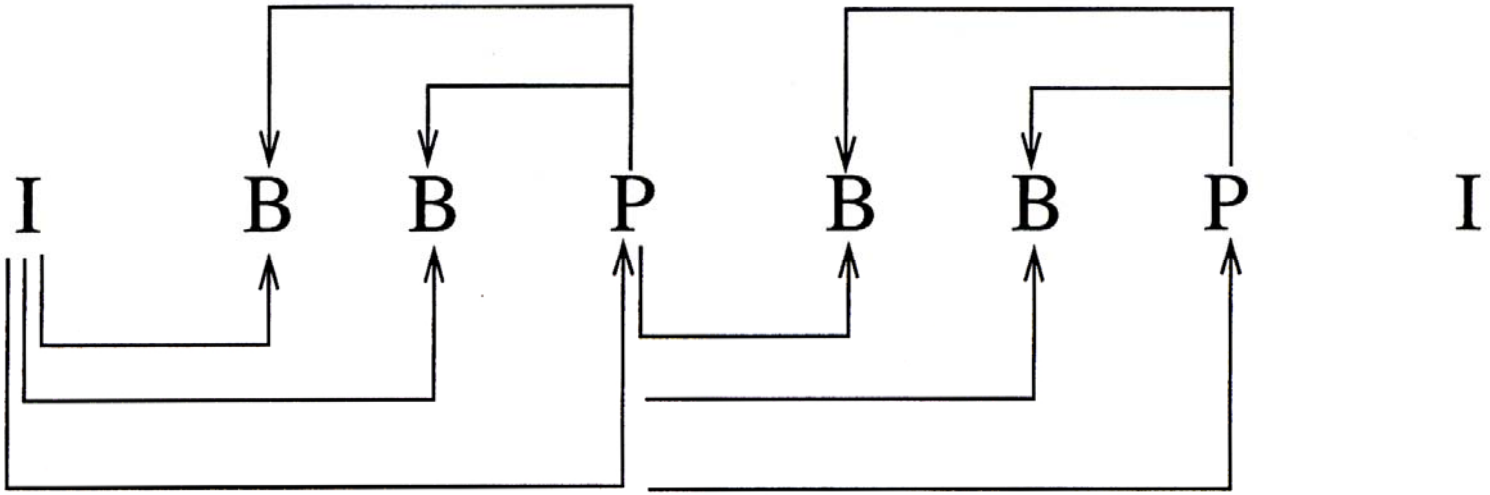
MPEG 1

- Does not recognize fields and frames.
- Only non-interlaced data
- Basic Concepts
  - Inter frame and Intra frame coding
  - DCT
  - Block based motion compensation
  - Huffman coding for motion vectors and quantized DCT coefficients
  - Inter frame coding:

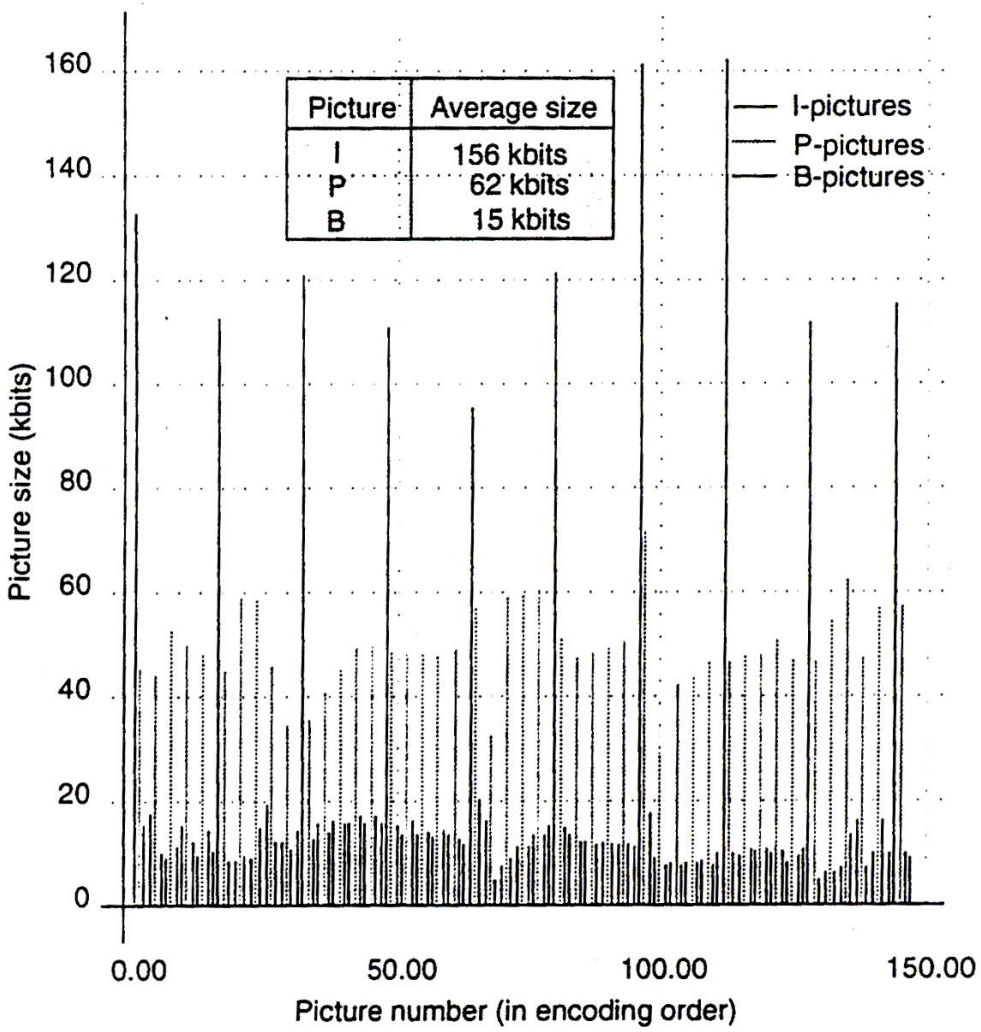
    Predictive AND Interpolative

Picture Types:

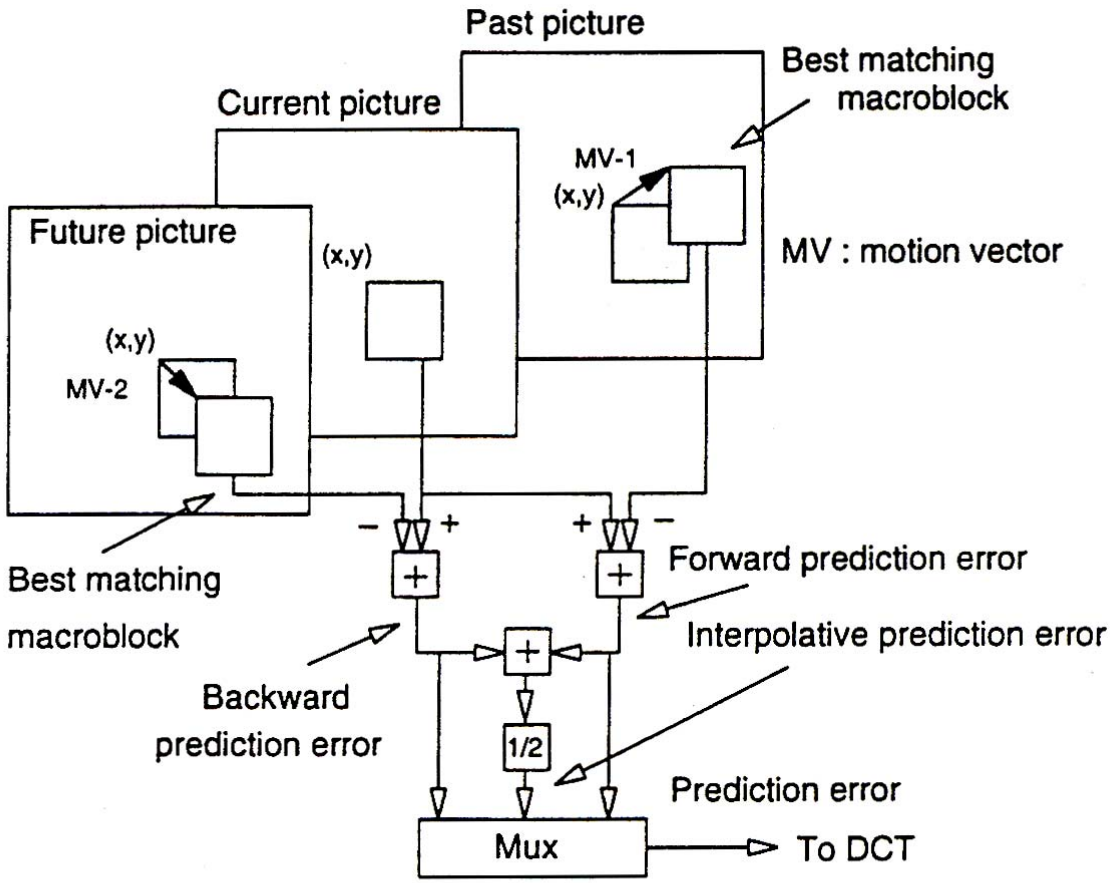
- Three types
- I pictures: Intra
- P pictures: Predicted from past I or P frames
- B pictures: Bidirection predicted either from past or future I or P frames
- Example



Picture Types (cont.)



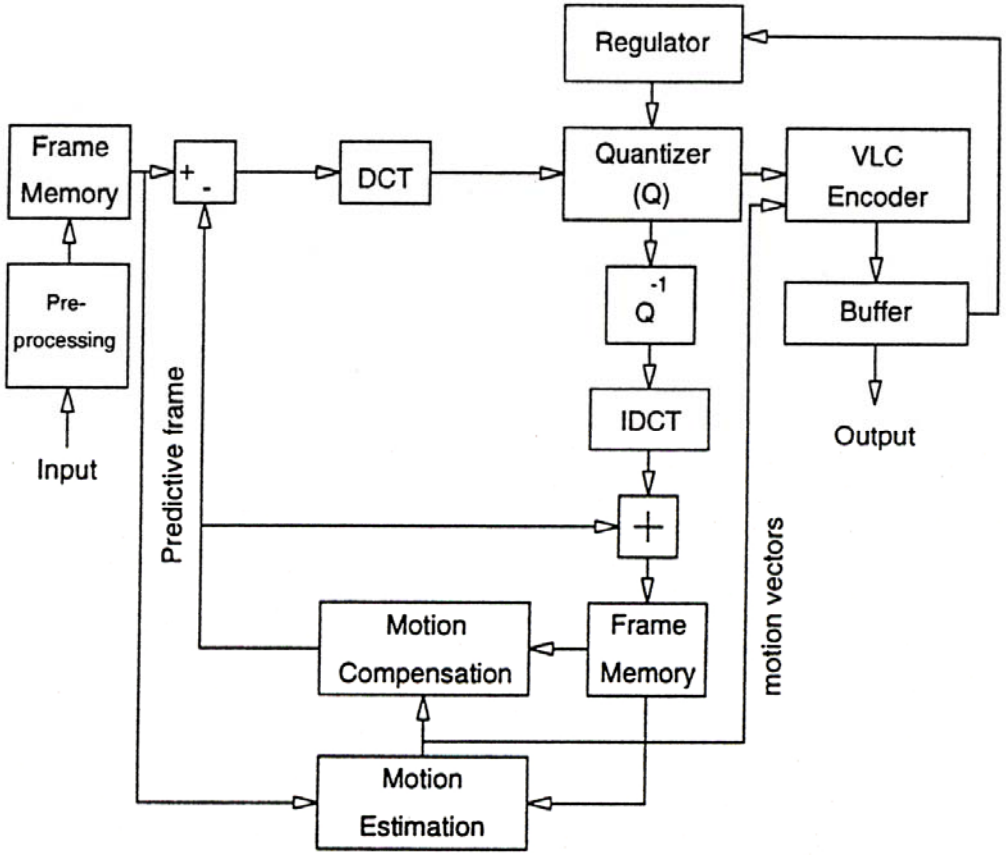
Bidirectional Motion Compensation



Bidirectional motion compensation.

Block Diagram of MPEG Encodes

- Assumes 4:2:0 color subsampling N x N luminance
  N/2 x N/2 each of chrominance



MPEG 2:

- Quite similar to MPEG 1
- Differences from MPEG 1:
  - Interlaced pictures $\rightarrow$ affects motion compensation
  - Color subsampling

    4 : 2 : 2 and 4 : 4 : 4 as well as 4 : 2 : 0

	Y	Cr	Cb
4 : 2 : 0	N x N	$\frac{N}{2} \times \frac{N}{2}$	$\frac{N}{2} \times \frac{N}{2}$
4 : 2: 2	N x N	$\frac{N}{2} \times N$	$\frac{N}{2} \times N$
4 : 4 : 4	N x N	N x N	N x N

- Profiles + levels

Profiles + Levels

- Profiles:
  - Simple 4 : 2 : 0
  - Main 4 : 2 : 0

  } → Nonscalable

  - Main + 4 : 2 : 0
  - Next 4 : 2 : 2

  } → Scalable
- Most applications → Main
- Next includes hierarchical representation → terrestrial broadcasting
- Main + between main and Next
- Simple, same as main but no bidirectional prediction → low cost
- Levels: specifies max spatio-temporal resolution + bit Rate.

Levels		Profiles			
		Nonscalable		Scalable	
		Simple 4:2:0	Main 4:2:0	Main+ 4:2:0	Next 4:2:2
High	Max resolution/ rate (Hz)	N/A	1920 x 1152/60	N/A	1920 x 1152/60
	Min. resolution/ rate (Hz)	N/A	N/A	N/A	960 x 576/30
	Bitrate (Mbits/s)	N/A	80	N/A	100 (all layers) 80 (base+mid) 25 (base layer)
High-1440	Max resolution/ rate (Hz)	N/A	1440 x 1152/60	1440 x 1152/60	1440 x 1152/60
	Min. resolution/ rate (Hz)	N/A	N/A	720 x 576/30	720 x 576/30
	Bitrate (Mbits/s)	N/A	60	60 (all layers) 40 (base+mid) 15 (base layer)	80 (all layers) 60 (base+mid) 20 (base layer)
Main	Max resolution/ rate (Hz)	720 x 576/30	720 x 576/30	720 x 576/30	720 x 576/30
	Min. resolution/ rate (Hz)	N/A	N/A	N/A	352 x 288/30
	Bitrate (Mbits/s)	15	15	15 (all layers) 10 (base layer)	20 (all layers) 15 (base+mid) 4 (base layer)
Low	Max resolution/ rate (Hz)	N/A	352 x 288/30	352 x 288/30	N/A
	Min. resolution/ rate (Hz)	N/A	N/A	N/A	N/A
	Bitrate (Mbits/s)	N/A	4	4 (all layers) 3 (base layer)	N/A

Scalability

- **Allows layered representation of Coded bit stream.**
- **4 modes**
  - Data Partitioning: 2 channels, header + data
  - SNR Scalability: Applications with video quality at multiple quality levels
  - Spatial scalability
  - Temporal scalability

H. 261 Video Coding

- Collaboration between Telecom operators and manufacturers of video conferencing equipment
- H. 261 or p x 64 kbps

  $P = 1 \rightarrow 30$
- H. 320 is complete family
  - H. 261 video
  - G. 722, G. 726, G. 728
  - H. 221 $\rightarrow$ Multiplexing
  - H. 230, H. 242 $\rightarrow$ Handshaking
  - H. 233 $\rightarrow$ Encryption

H. 261

- Format CIF OR QCIF
- CIF = Common Intermediate Format

  Y: 352 x 288

  C_b: 176 x 144

  C_r: 176 x 144
- QCIF ½ x ½ CIF
- Similar to MPEG
  - Motion Comp
  - DCT
  - VLC
- No bidirectional
- BCM codes for error detection