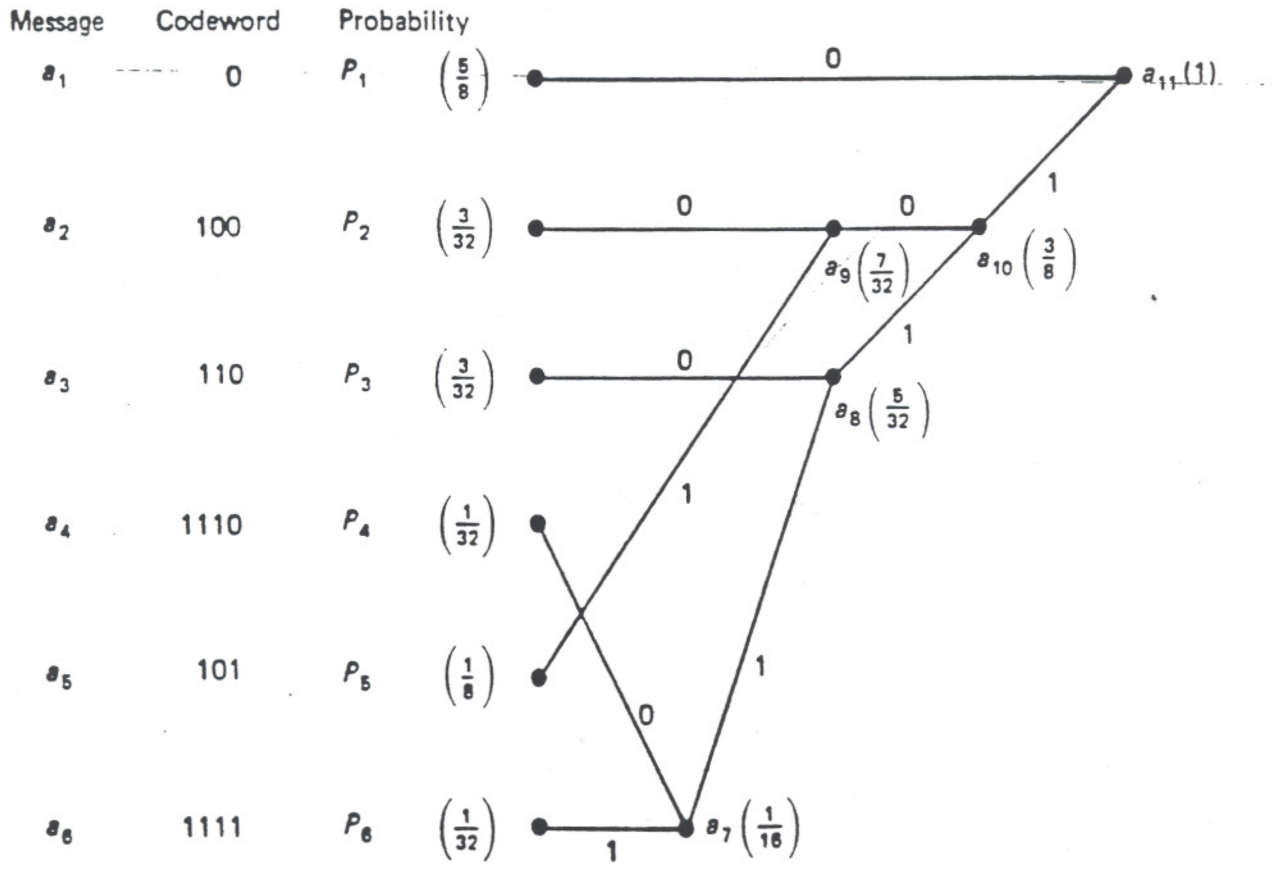
Methods of Bit Assignment (cont.)

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

Example:



Message	Codeword	Probability
a_1	0	P_1 $\left(\frac{5}{8}\right)$
a_2	100	P_2 $\left(\frac{3}{32}\right)$
a_3	110	P_3 $\left(\frac{3}{32}\right)$
a_4	1110	P_4 $\left(\frac{1}{32}\right)$
a_5	101	P_5 $\left(\frac{1}{8}\right)$
a_6	1111	P_6 $\left(\frac{1}{32}\right)$

a_{11} (1)
a_{10} $\left(\frac{3}{8}\right)$
a_9 $\left(\frac{7}{32}\right)$
a_8 $\left(\frac{5}{32}\right)$
a_7 $\left(\frac{1}{16}\right)$
0
1

Figure 10.16 Illustration of codeword generation in Huffman coding. Message possibilities with higher probabilities are assigned with shorter codewords.

- 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 $\simeq 1.813$ bits/message
- Entropy: $-(\frac{5}{8} \log_2 \frac{5}{8} + \frac{3}{32} \log_2 \frac{3}{32} + \frac{3}{32} \log_2 \frac{3}{32} + \frac{1}{32} \log_2 \frac{1}{32} + \frac{1}{8} \log_2 \frac{1}{8} + \frac{1}{32} \log_2 \frac{1}{32}) = 1.752$ bits/message

Methods of Bit Assignment (cont.)

Vector Case

f_1, f_2	Prob.
r_0, r_0'	p_0
r_1, r_1'	p_1
r_2, r_2'	p_2
$\cdot$	$\cdot$
$\cdot$	$\cdot$
$\cdot$	$\cdot$
$\cdot$	$\cdot$

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

Comments:

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

Arithmetic Coding

- Why not use Huffman?

- Can show rate of Huffman code is within $P_{max} + 0.\cancel{0}86$ 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 of Arithmetic Coding

- Two steps:

  1) Generate a tag

  2) Assign a binary code to the tag

- Tag Generation:

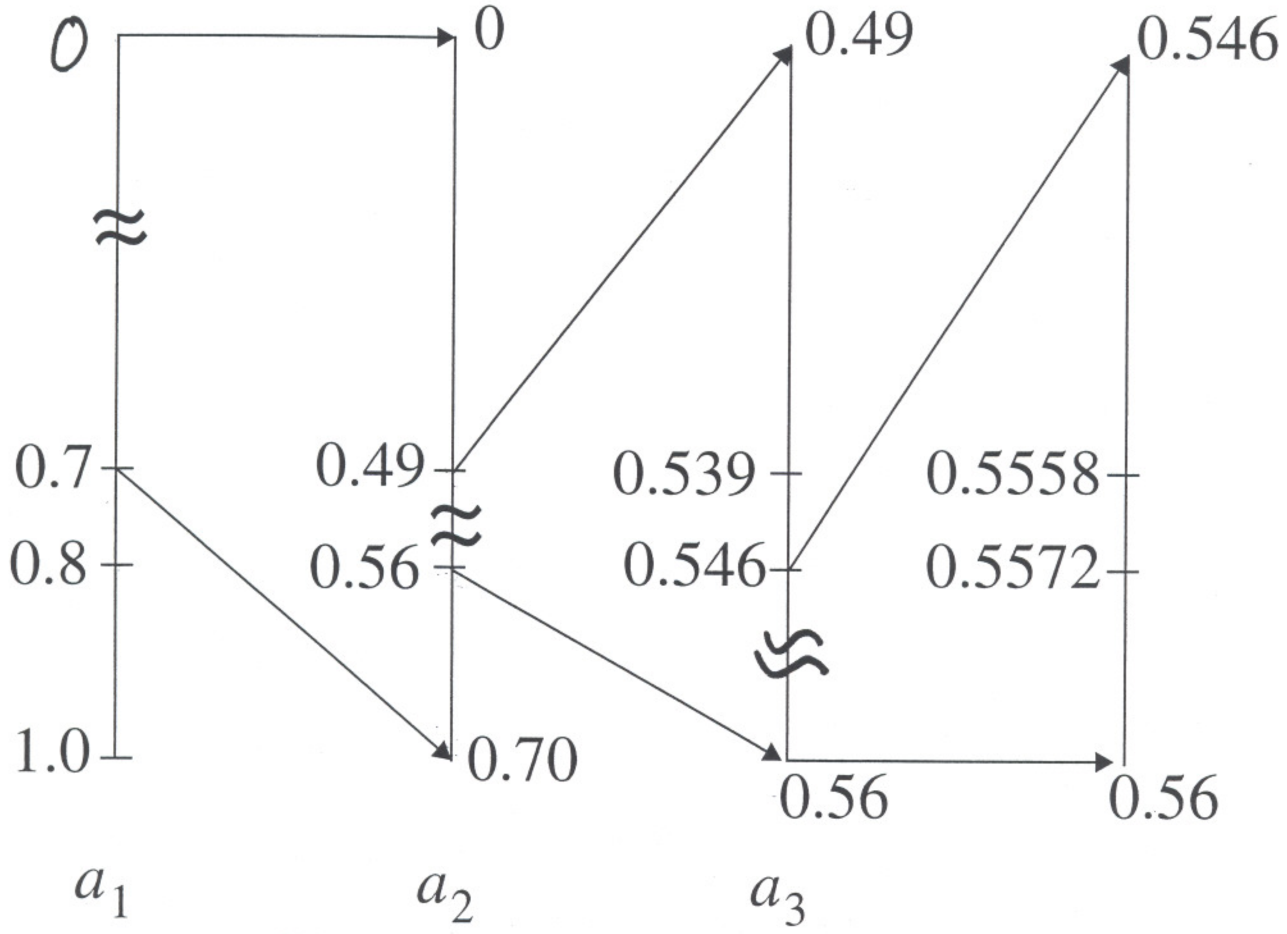
  A sequence of symbols from an alphabet source with a given Pdf results in a unique sub-interval in [0,1].

Example of Tag Generation:

- $A = \{a_1, a_2, a_3\}$ $\quad P(a_1) = 0.7$

  $P(a_2) = 0.1 \quad P(a_3) = 0.2$

- Suppose we encode a_1, a_2, a_3

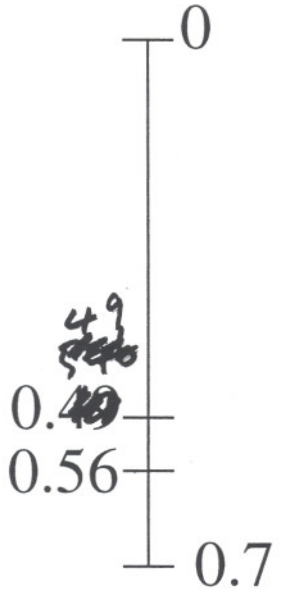


- Interval [0.546, 0.56] uniquely specifies sequence (a_1, a_2, a_3)

Deciphering the Tag:

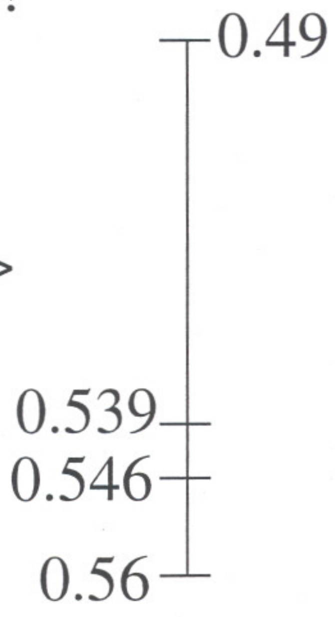
- Suppose we know 3 symbols resulted in subinterval [0.546, 0.56]
- Decode the symbols?
- Is the first symbol a_1 or a_2 or a_3 ?

$$\left.\begin{array}{l} a_1 \longrightarrow [0, 0.7] \\ a_2 \longrightarrow [0.7, 0.8] \\ a_3 \longrightarrow [0.8, 1] \end{array}\right\} \Longrightarrow \boxed{a_1} \Longrightarrow$$



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

$$\left.\begin{array}{l} a_1 \longrightarrow [0, 0.49] \\ a_2 \longrightarrow [0.49, 0.56] \\ a_3 \longrightarrow [0.56, 0.7] \end{array}\right\} \Longrightarrow a_2 \Longrightarrow$$



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

$$\left\lceil \log \frac{1}{P(\vec{X})} \right\rceil + 1 \text{ bits}$$

- 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

$$0.553 = \frac{1}{2^1} + \frac{1}{2^5} + \frac{1}{2^6} + \frac{1}{2^8} + \frac{1}{2^9} + \dots$$

$$= 0.1\ 00011011\dots$$

- Truncate to 8 bits ===>
  Binary Representation:

10001101

Efficiency of Arithmetic Code

- We can show

$$H(X) \le l_A \le H(X) + \frac{2}{m}$$

  - 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 occuring 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

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

- 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 ---------> ~~0~~ 110
5. . . . .

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

- Opposite of Huffman coding:

Ziv

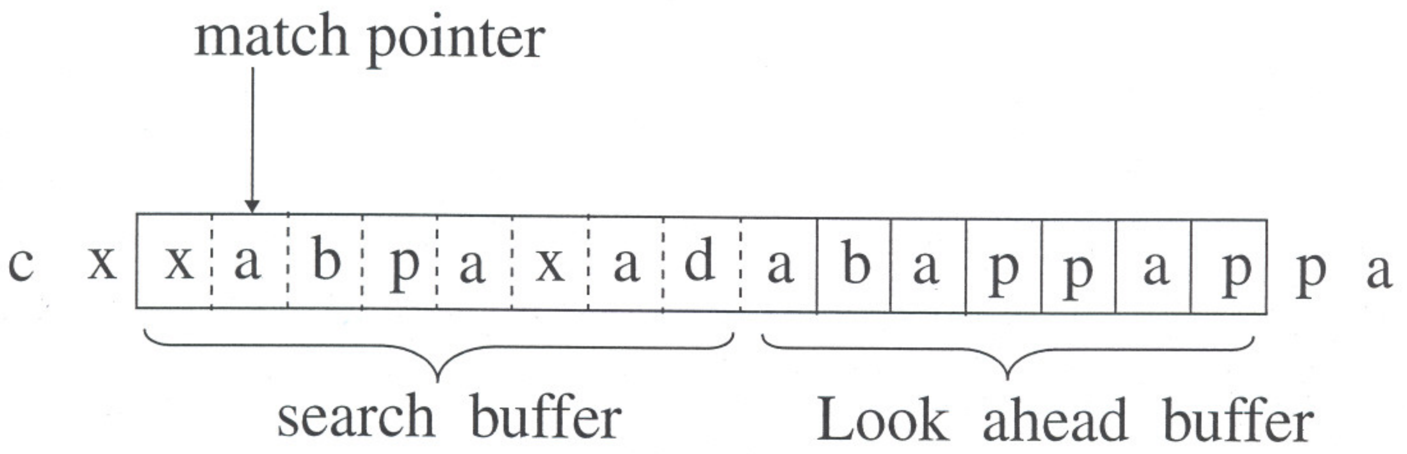
Adaptive Dictionary:

- ~~Zir~~ Lempel 1977 + 1978
- LZ1 -----> 1977 LZ2 -----> 1978
- LZ1 discussed Here.
- Basic idea:

  Dictionary portion of previously encoded sequence

- Sliding window:

  1) search buffer
  2) lookahead buffer



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

```
. . . . | c a b r a c a | d a b r a r | r a . . .
```

1. No match to d ---> $< \overset{0 \leftarrow \text{zero}}{\text{ø}}, 0, C(d) >$

```
. . . . c | a b r a c a d | a b r a r r ¦ a r . . .
```

2. Match for a : $\left\{ \begin{array}{l} \overset{\text{o (letter o)}}{\text{ø}} = 2 \dashrightarrow l = 1 \\ \text{ø} = 4 \dashrightarrow l = 1 \\ \boxed{\text{ø} = 7 \dashrightarrow l = \cancel{1}} \; 4 \end{array} \right.$

< 7, 4, C (r) >

```
| a d a b r a r | r a r r a d ¦
```

3. Ma$\overset{t}{\vee}$ch for r :

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

$$< 3,\ \underset{\substack{\uparrow \\ \text{exceeds the} \\ \text{boundary between} \\ \text{search and} \\ \text{look ahead buffer}}}{5},\ C\,(d) >$$

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 $\cancel{c}$ = codeword of symbol in LA buffer, following match

Example: < 7, 2, codeword for a >

Adaptive Dictionary

- Why send ~~c~~ C?
    - just in case no match
- Total # of bits:

$W \rightarrow$ capital W

$$\lceil \log_2 S \rceil + \lceil \log_2 \overset{W}{\cancel{w}} \rceil + \lceil \log_2 A \rceil$$

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
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

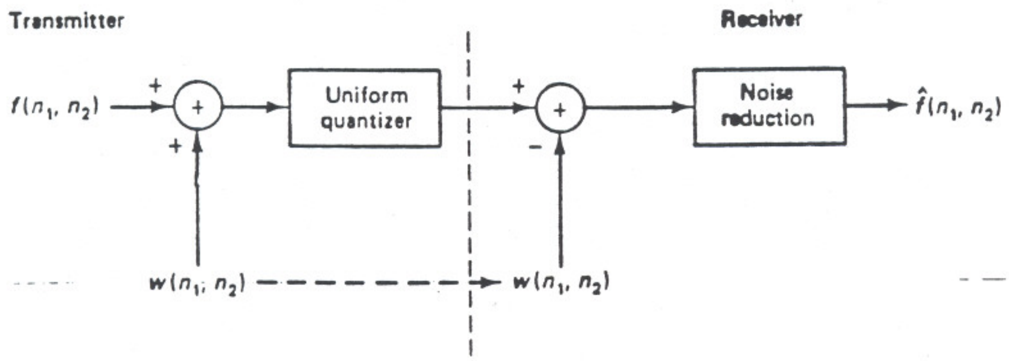
$$\begin{tikzcd}
f(n_1,n_2) \arrow[r] & \boxed{\text{Uniform Quantizer}} \arrow[r] & \hat{f}(n_1,n_2)
\end{tikzcd}$$

$$\begin{tikzcd}
f(n_1,n_2) \arrow[r] & \boxed{\text{Non-Linearity}} \arrow[r] & \boxed{\text{Uniform Quantizer}} \arrow[r] & \boxed{\text{Non-Linearity}^{-1}} \arrow[r] & \hat{f}(n_1,n_2)
\end{tikzcd}$$

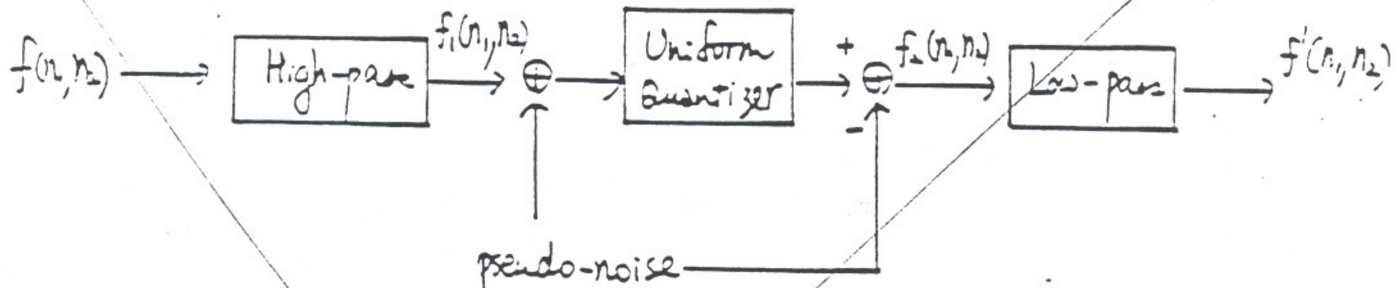
- very simple
- typically requires over 5-6 bits/pixel for good quality
- false contours for low-bit rate case

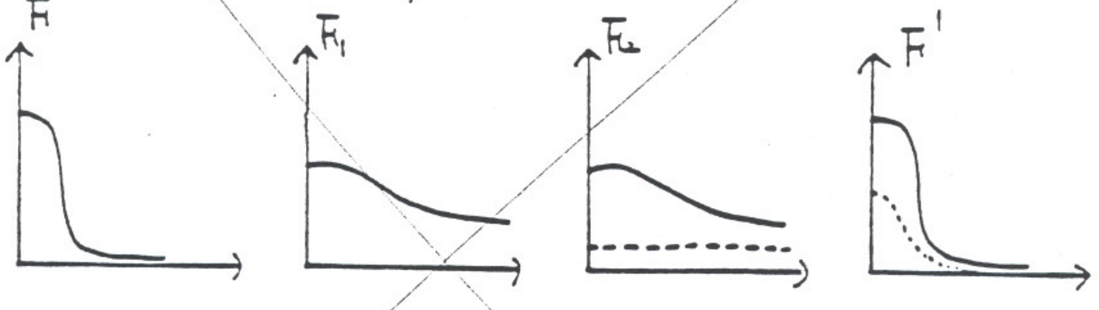
Improvements of PCM (cont.)

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

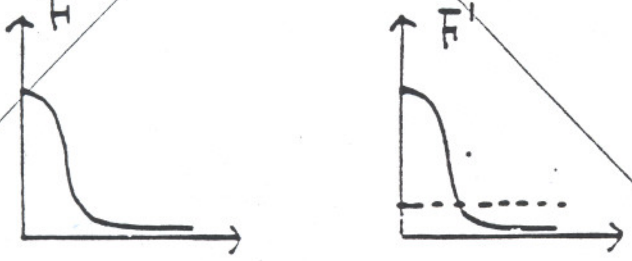


~~3. Roberts' Pseudo-Noise Technique and Highpass/Lowpass Filtering:~~



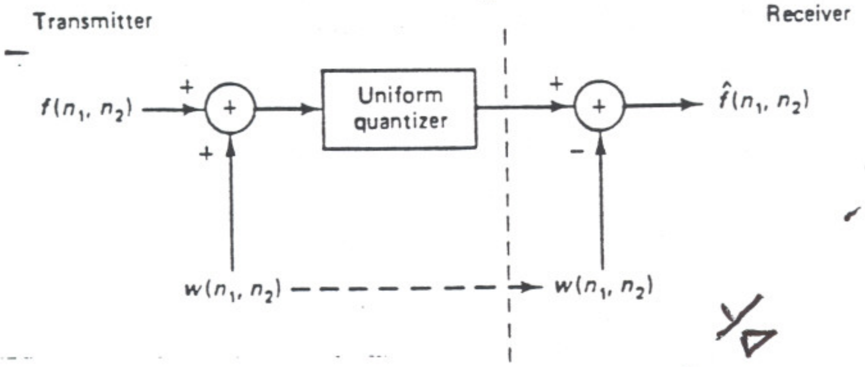


~~If there were not Highpass/Lowpass Filtering:~~

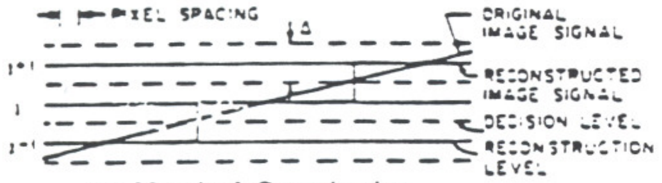


Improvements of PCM

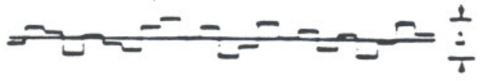
1. Roberts' Pseudo-Noise Technique:



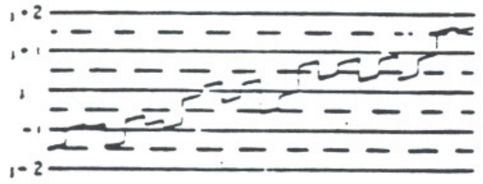
$$p(w) = \begin{cases} \frac{1}{\Delta} & -\frac{\Delta}{2} \le w_0 \le \Delta/2 \\ 0 & \text{otherwise} \end{cases}$$



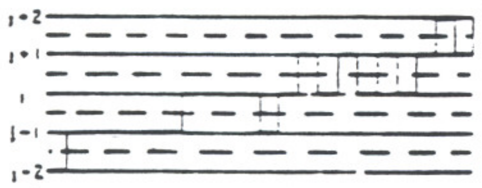
(a) Nominal Quantization



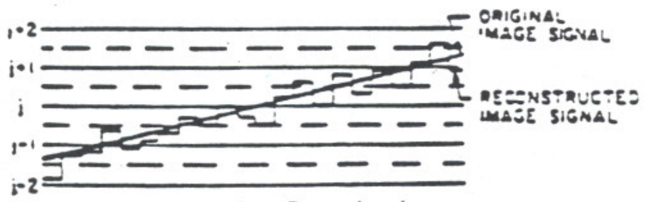
(b) One bit Pseudonoise



(c) Original Image Signal Plus Noise



(d) Quantized Image Signal Plus Noise



(e) Pseudonoise Quantization

- false contours disappear — replaced by additive random noise

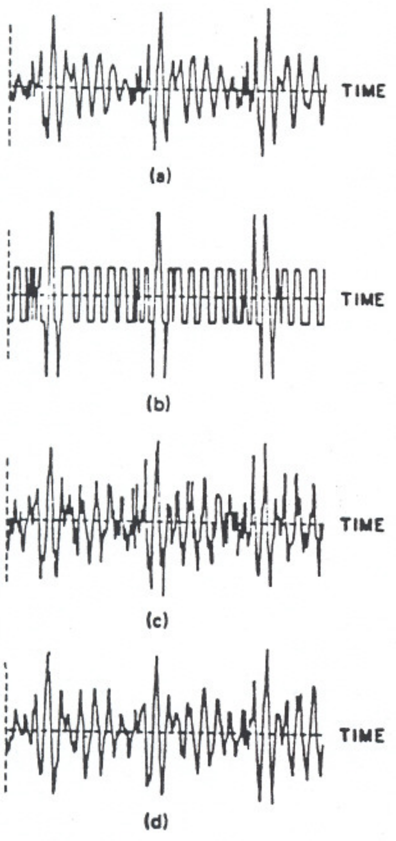


Figure 10.21 Example of quantization noise reduction in PCM speech coding. (a) Segment of noise-free voiced speech; (b) PCM-coded speech at 2 bits/sample; (c) PCM-coded speech at 2 bits/sample by Roberts's pseudonoise technique; (d) PCM-coded speech at 2 bits/sample with quantization noise reduction.

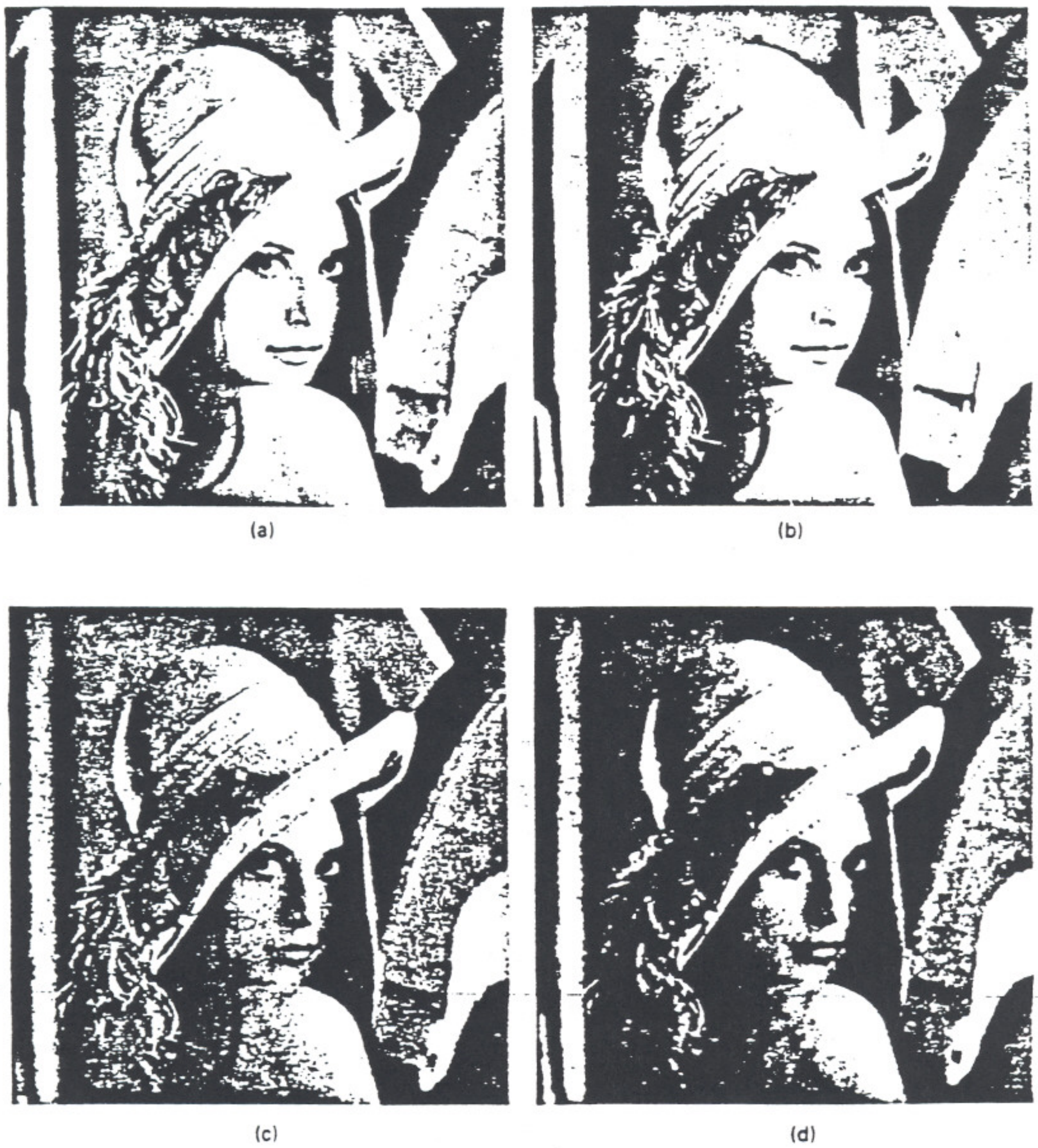
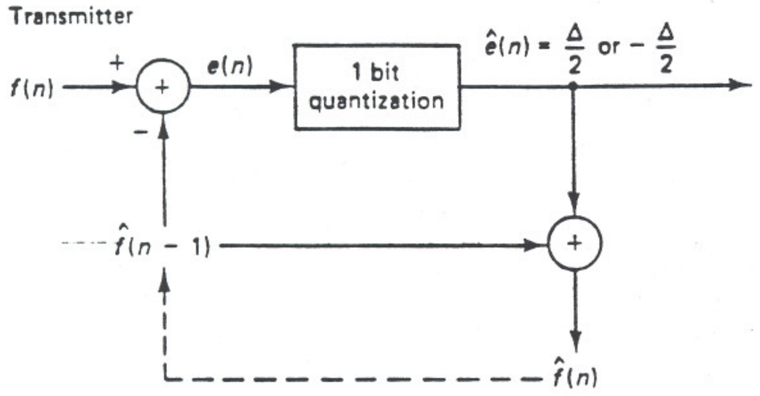

(a) (b) (c) (d)

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

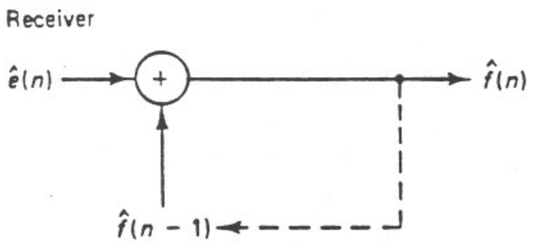
Delta Modulation (DM)

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

Transmitter



Receiver



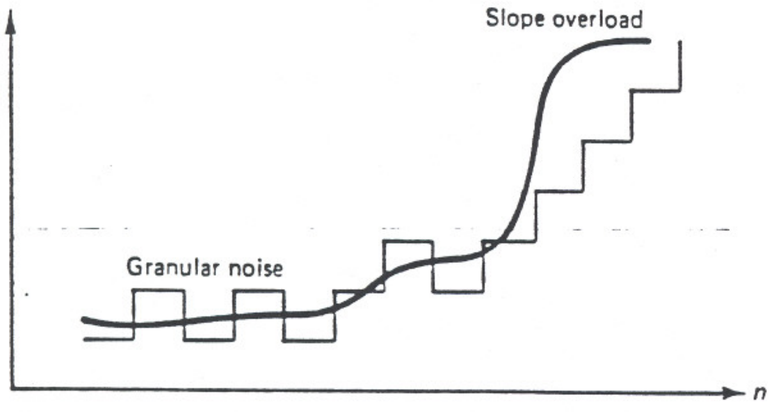


Figure 10.26 Granular noise and slope-overload distortion in delta modulation.

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

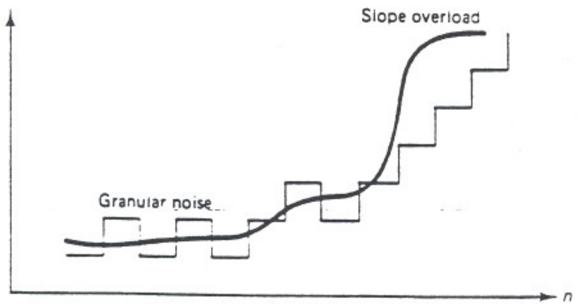


Figure 10.26 Granular noise and slope-overload distortion in delta modulation.

quirements, and the step size Δ is chosen through some compromise between the two requirements.

Figure 10.27 illustrates the performance of a DM system. Figures 10.27(a) and (b) show the results of DM with step sizes of $\Delta = 8\%$ and 15%, respectively, of the overall dynamic range of $f(n_1, n_2)$. The original image used is the 512×512-pixel image in Figure 10.22(a). When Δ is small [Figure 10.27(a)], the granular noise is reduced, but the slope overload distortion problem is severe and the resulting image appears blurred. As we increase Δ [Figure 10.27(b)], the slope overload distortion is reduced, but the graininess in the regions where the signal varies slowly is more pronounced.

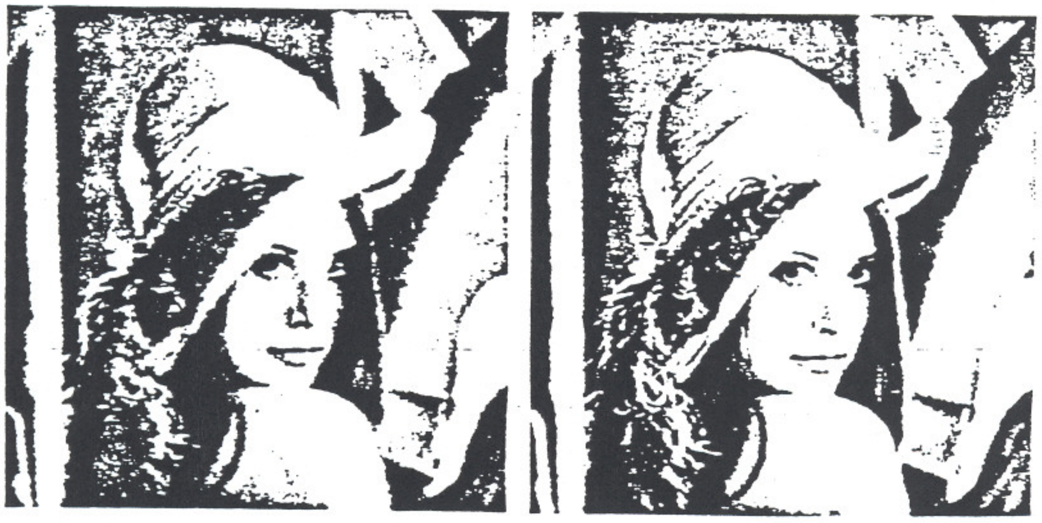


Figure 10.27 Example of delta-modulation (DM)-coded image. The original image used is the image in Figure 10.22(a). (a) DM-coded image with $\Delta = 8\%$ of the overall dynamic range, NMSE = 14.8%, SNR = 8.3 dB; (b) DM-coded image with $\Delta = 15\%$, NMSE = 9.7%, SNR = 10.1 dB.

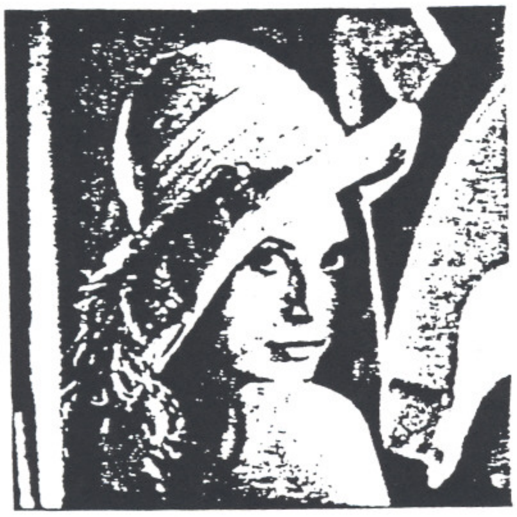


Figure 10.28 DM-coded image at 2 bits/pixel. The original image used is the image in Figure 10.22(a). NMSE = 2.4%, SNR = 16.2 dB.

To obtain good quality image reconstruction using DM without significant graininess or slope overload distortion, 3–4 bits/pixel is typically required. A bit rate higher than 1 bit/pixel can be obtained in DM by oversampling the original analog signal relative to the sampling rate used in obtaining $f(n_1, n_2)$. Oversampling reduces the slope of the digital signal $f(n)$ so a smaller Δ can be used without increasing the slope overload distortion. An example of an image coded by DM at 2 bits/pixel is shown in Figure 10.28. To obtain the image in Figure 10.28, the size of the original digital image in Figure 10.22(a) was increased by a factor of two by interpolating the original digital image by a factor of two along the horizontal direction. The interpolated digital image was coded by DM with $\Delta = 12\%$ of the dynamic range of the image and the reconstructed image was undersampled by a factor of two along the horizontal direction. The size of the resulting image is the same as the image in Figure 10.27, but the bit rate in this case is 2 bits/pixel.

10.3.3 Differential Pulse Code Modulation

Differential pulse code modulation (DPCM) can be viewed as a generalization of DM. In DM, the difference signal $e(n) = f(n) - \hat{f}(n-1)$ is quantized. The most recently coded $\hat{f}(n-1)$ can be viewed as a prediction of $f(n)$ and $e(n)$ can be viewed as the error between $f(n)$ and a prediction of $f(n)$. In DCPM, a prediction of the current pixel intensity is obtained from more than one previously coded pixel intensity. In DM, only one bit is used to code $e(n)$. In DPCM, more than one bit can be used in coding the error.

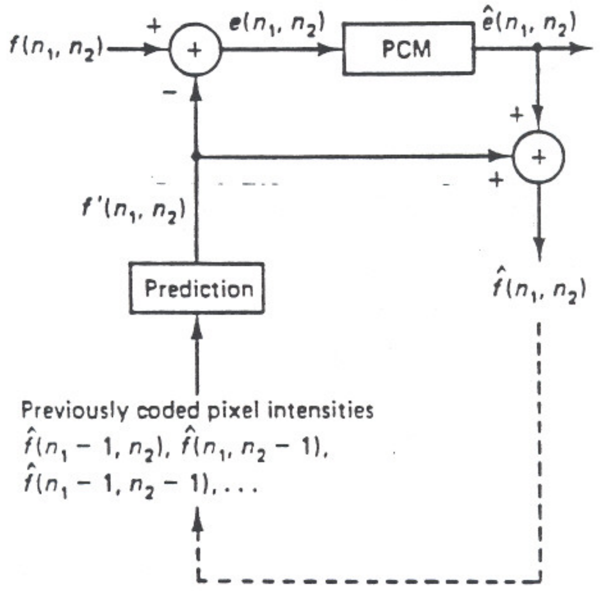
A DPCM system is shown in Figure 10.29. To code the current pixel intensity $f(n_1, n_2)$, $f(n_1, n_2)$ is predicted from previously reconstructed pixel intensities. The predicted value is denoted by $f'(n_1, n_2)$. In the figure, we have assumed that $\hat{f}(n_1 - 1, n_2)$, $\hat{f}(n_1, n_2 - 1)$, $\hat{f}(n_1 - 1, n_2 - 1)$, . . . were reconstructed prior to coding $f(n_1, n_2)$. We are attempting to reduce the variance of

Differential Pulse Code Modulation (DPCM)

$f(n_1, n_2)$: original image

$\hat{f}(n_1, n_2)$: reconstructed image

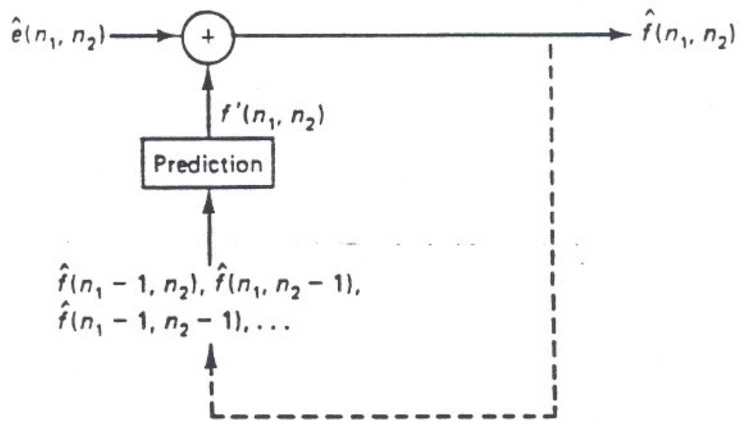
Transmitter



$f(n_1, n_2)$
+
−
$e(n_1, n_2)$
PCM
$\hat{e}(n_1, n_2)$
+
+
$f'(n_1, n_2)$
Prediction
$\hat{f}(n_1, n_2)$
Previously coded pixel intensities
$\hat{f}(n_1 - 1, n_2), \hat{f}(n_1, n_2 - 1),$
$\hat{f}(n_1 - 1, n_2 - 1), \ldots$

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

Receiver



$\hat{e}(n_1, n_2)$
$\hat{f}(n_1, n_2)$
$f'(n_1, n_2)$
Prediction
$\hat{f}(n_1 - 1, n_2), \hat{f}(n_1, n_2 - 1),$
$\hat{f}(n_1 - 1, n_2 - 1), \ldots$

- requires 2-3 bits/pixel for good quality image where R_a is the region of (k_1, k_2) over which $a(k_1, k_2)$ is nonzero. Typically, $f'(n_1, n_2)$ is obtained by linearly combining $\hat{f}(n_1 - 1, n_2)$, $\hat{f}(n_1, n_2 - 1)$, and $\hat{f}(n_1 - 1, n_2 - 1)$. Since the prediction of $f(n_1, n_2)$ is made in order to reduce the variance of $e(n_1, n_2)$, it is reasonable to estimate $a(k_1, k_2)$ by minimizing

$$E[e^2(n_1, n_2)] = E[(f(n_1, n_2) - \sum\sum_{(k_1, k_2) \in R_a} a(k_1, k_2)\hat{f}(n_1 - k_1, n_2 - k_2))^2]. \qquad (10.40)$$

Since $\hat{f}(n_1, n_2)$ is a function of $a(k_1, k_2)$ and depends on the specific quantizer used, solving (10.40) is a nonlinear problem. Since $\hat{f}(n_1, n_2)$ is the quantized version of $f(n_1, n_2)$, and is therefore a reasonable representation of $f(n_1, n_2)$, the prediction coefficients $a(k_1, k_2)$ are estimated by minimizing

$$E\left[(f(n_1, n_2) - \sum\sum_{(k_1, k_2) \in R_a} a(k_1, k_2)f(n_1 - k_1, n_2 - k_2))^2\right]. \qquad (10.41)$$

Since the function in (10.41) minimized is a quadratic form of $a(k_1, k_2)$, solving (10.41) involves solving a linear set of equations in the form of

$$R_f(l_1, l_2) = \sum\sum_{(k_1, k_2) \in R_a} a(k_1, k_2)R_f(l_1 - k_1, l_2 - k_2) \qquad (10.42)$$

where $f(n_1, n_2)$ is assumed to be a stationary random process with the correlation function $R_f(n_1, n_2)$. The linear equations in (10.42) are the same as those used in the estimation of the autoregressive model parameters discussed in Chapters 5 and 6.

Figure 10.30 illustrates the performance of a DPCM system. Figure 10.30 shows the result of a DPCM system at 3 bits/pixel. The original image used is the image in Figure 10.22(a). The PCM system used in Figure 10.30 is a nonuniform quantizer. The prediction coefficients $a(k_1, k_2)$ used to generate the example are

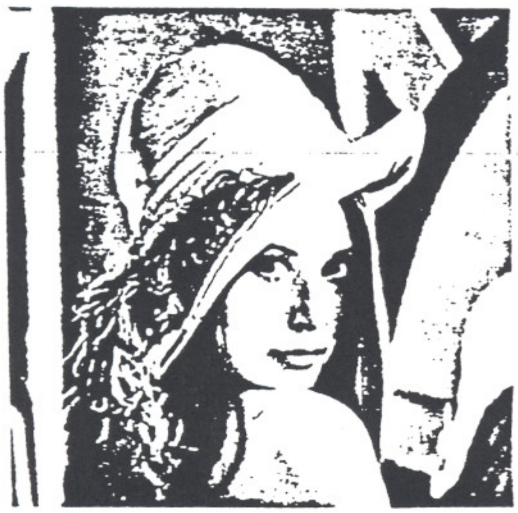


Figure 10.30 Example of differential pulse code modulation (DPCM)-coded image at 3 bits/pixel. Original image used is the image in Figure 10.22(a). NMSE = 2.2%, SNR = 16.6 dB.