Image Compression

Data compression

- The term data compression refers to the process of reducing the amount of data required to represent a given quantity of Information.

- Data and information are not synonymous.
- Data are the means by which information is conveyed. Data is used to represent information.

- Various amounts of data may be used to represent the same amount of information.

Data Redundancy

- **Data redundancies** in image processing refers to the data which are no longer required.
- Let b and b’ denote the number of bits in two representation of the same information, the **relative data redundancy R** of the representation with b bit is

$$R = 1 - \frac{1}{C}$$

- Where **C denotes the compression ratio.**

$$C = \frac{b}{b'}$$

Redundancy in Digital Images

- **Coding Redundancy**
- **Spatial and temporal redundancy**
- **Irrelevant information**

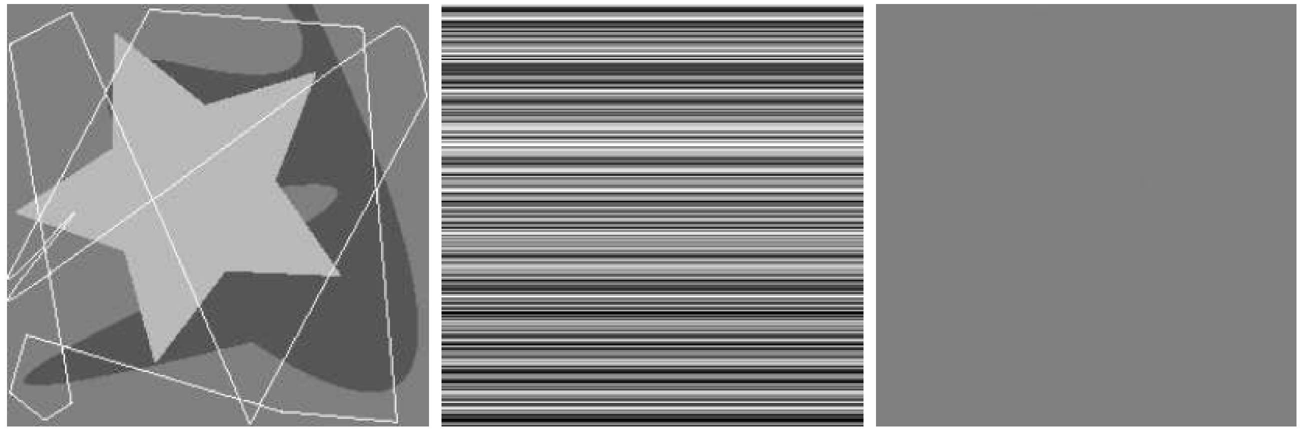

a b c

FIGURE 8.1 Computer generated 256 × 256 × 8 bit images with (a) coding redundancy, (b) spatial redundancy, and (c) irrelevant information. (Each was designed to demonstrate one principal redundancy but may exhibit others as well.)

Coding Redundancy

Coding Redundancy

- A code is a system of symbols (letters, numbers, bits, and the like) used to represent a body of information or set of events.
- Each piece of information or event is assigned a sequence of code symbols, called a code word. The number of symbols in each code word is its length.

Coding Redundancy

- Assume that a discrete random variable r_k in the interval $[0 \sim L\text{-}1]$ is used to represent the intensities of an M×N image and that each occurs with probability $P_r(r_k)$.

$$p_r(r_k) = \frac{n_k}{MN} \qquad k = 0, 1, 2, \ldots, L - 1$$

where L is the number of intensity values, and n_k is the number of times that the kth intensity appears in the image. If the number of bits used to represent each value of r_k is $l(r_k)$, then the average number of bits required to represent each pixel is

$$L_{\text{avg}} = \sum_{k=0}^{L-1} l(r_k) p_r(r_k)$$

Example of variable-length coding.

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	10000000	8	1	1
$r_{186} = 186$	0.25	11000100	8	000	3
$r_{255} = 255$	0.03	11111111	8	001	3
r_k for $k \neq 87, 128, 186, 255$	0	—	8	—	0

if the scheme designated as code 2, the average length of the encoded pixels is

$$L_{\text{avg}} = 0.25(2) + 0.47(1) + 0.25(3) + 0.03(3) = 1.81 \text{ bits}$$

The total number of bits needed to represent the entire image is $MNL_{\text{avg}} = 256*256*1.81 = 118{,}621$.

Spatial and temporal redundancy

- Information is unnecessarily replicated in the representations of the correlated pixels.
- In a video sequence, temporally correlated pixels (i.e., those similar to or dependent on pixels in nearby frames) also duplicate information.

In 8.1(b)
its pixels are independent of one another in the vertical direction.
Because the pixels along each line are identical, they are maximally correlated in the horizontal direction.

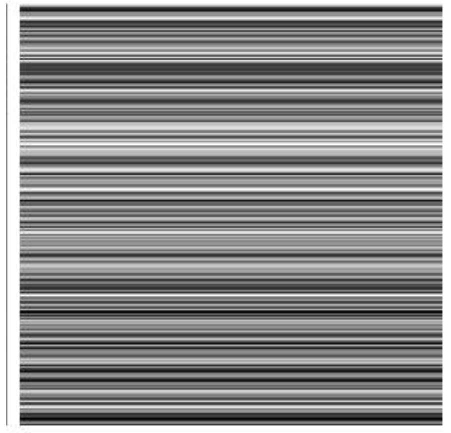


Fig 8.1(b)

- Spatial redundancy that can be eliminated, for instance, by representing the image in Fig. 8.1(b) as ***a sequence of run-length pairs***

- **Each run-length pair** specifies the start of a new intensity and the number of consecutive pixels that have that intensity.

- Each 256-pixel line of the original representation is replaced by a single 8-bit intensity value and length 256 in the run-length representation.

Irrelevant Information

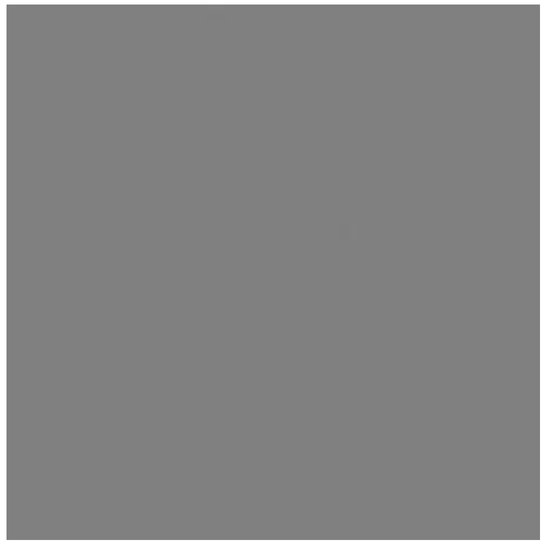
Most 2-D intensity arrays contain information that is ignored by the human visual system and/or extraneous to the intended use of the image. It is redundant in the sense that it is not used.

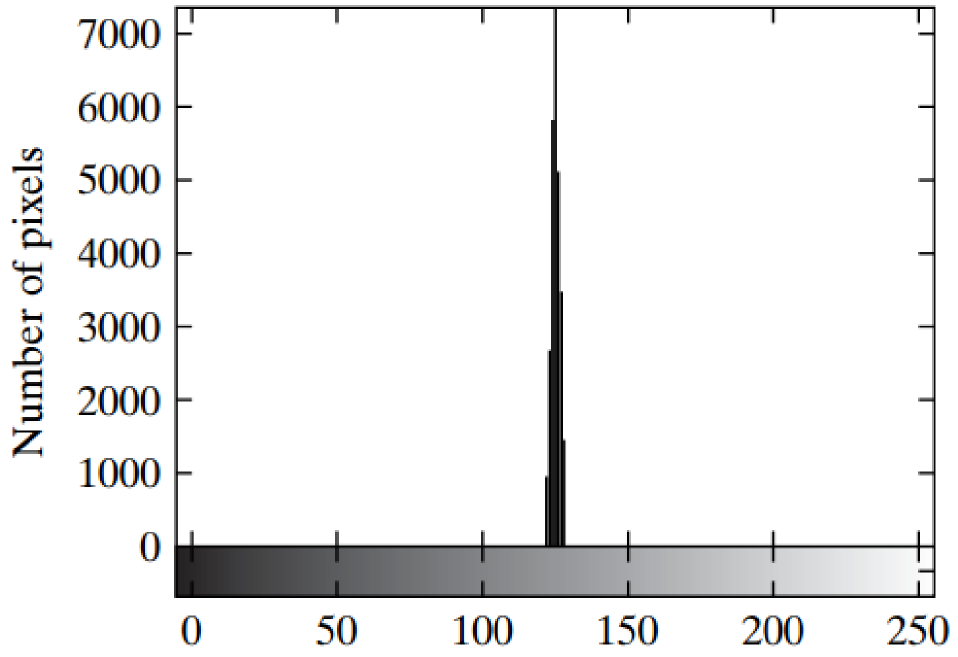
Solution: because it appears to be a homogeneous field of gray, can be represented by its average intensity alone—a single 8-bit value. The original 256*256*8 bit intensity array is reduced to a single byte;

Irrelevant Information

- Note that there are several intensity values (intensities 125 through 131) actually present. The human visual system averages these intensities, perceives only the average value, and ignores the small changes in intensity that are present in this case.

Fig 8.1(C)





Measuring Image Information

- Information theory provides the mathematical framework to know the minimum amount of data that is sufficient to describe an image without losing information

- The average information per source output, called the *entropy* of the source, is

$$\tilde{H} = -\sum_{k=0}^{L-1} p_r(r_k) \log_2 p_r(r_k)$$

- It is not possible to code the *intensity values* of the imaginary source (and thus the sample image) with fewer than H bits/ pixel.

$$\begin{aligned} \tilde{H} &= -[0.25 \log_2 0.25 + 0.47 \log_2 0.47 + 0.25 \log_2 0.25 + 0.03 \log_2 0.03] \\ &= -[0.25(-2) + 0.47(-1.09) + 0.25(-2) + 0.03(-5.06)] \\ &\approx 1.6614 \text{ bits/pixel} \end{aligned}$$

Fidelity Criteria

- Because information is lost, a means of quantifying the nature of the loss is needed. Two types of criteria can be used for such an assessment: (1) objective fidelity criteria, and (2) subjective fidelity criteria.

- For any value of x and y, the error $e(x,y)$ between $f(x, y)$ and $\hat{f}(x, y)$ is

$$e(x,y) = \hat{f}(x,y) - f(x,y)$$

- The total error between the two images is

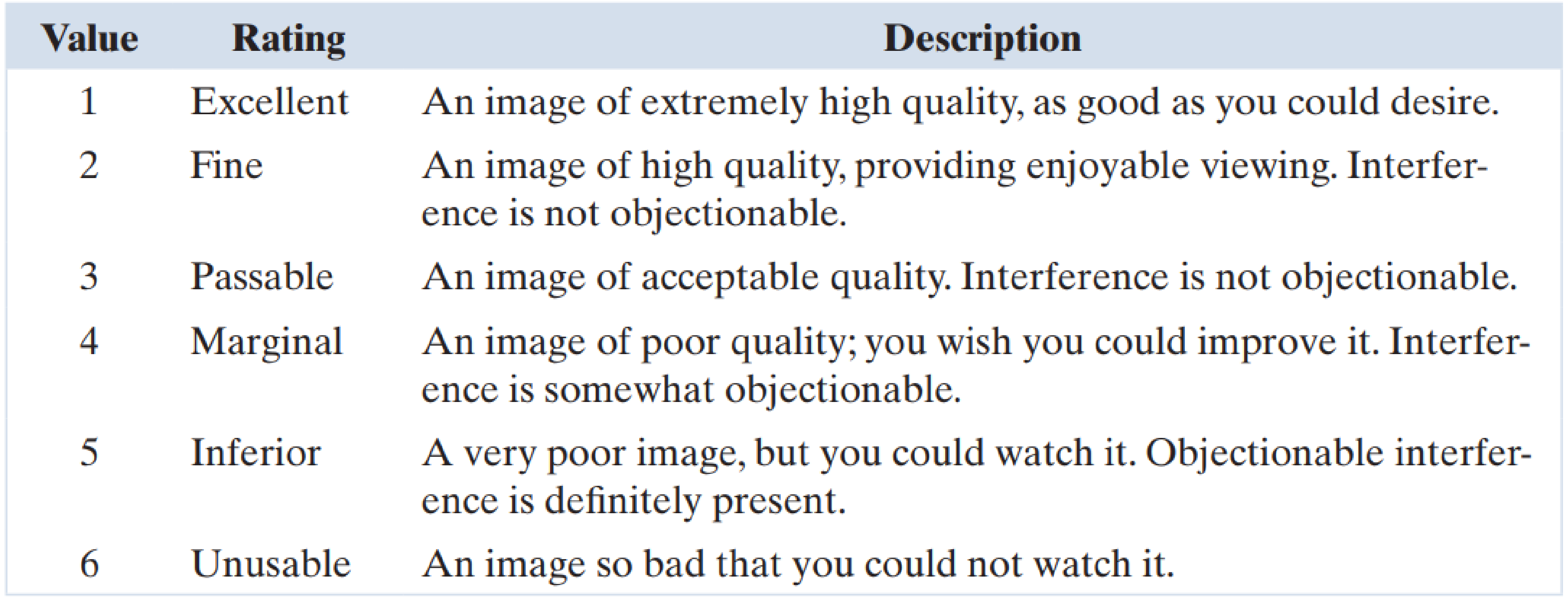
$$\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]$$

- *signal-to-noise ratio* of the image, denoted SNR_{ms}

$$\text{SNR}_{\text{ms}} = \frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}(x,y)^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y)-f(x,y)\right]^2}$$

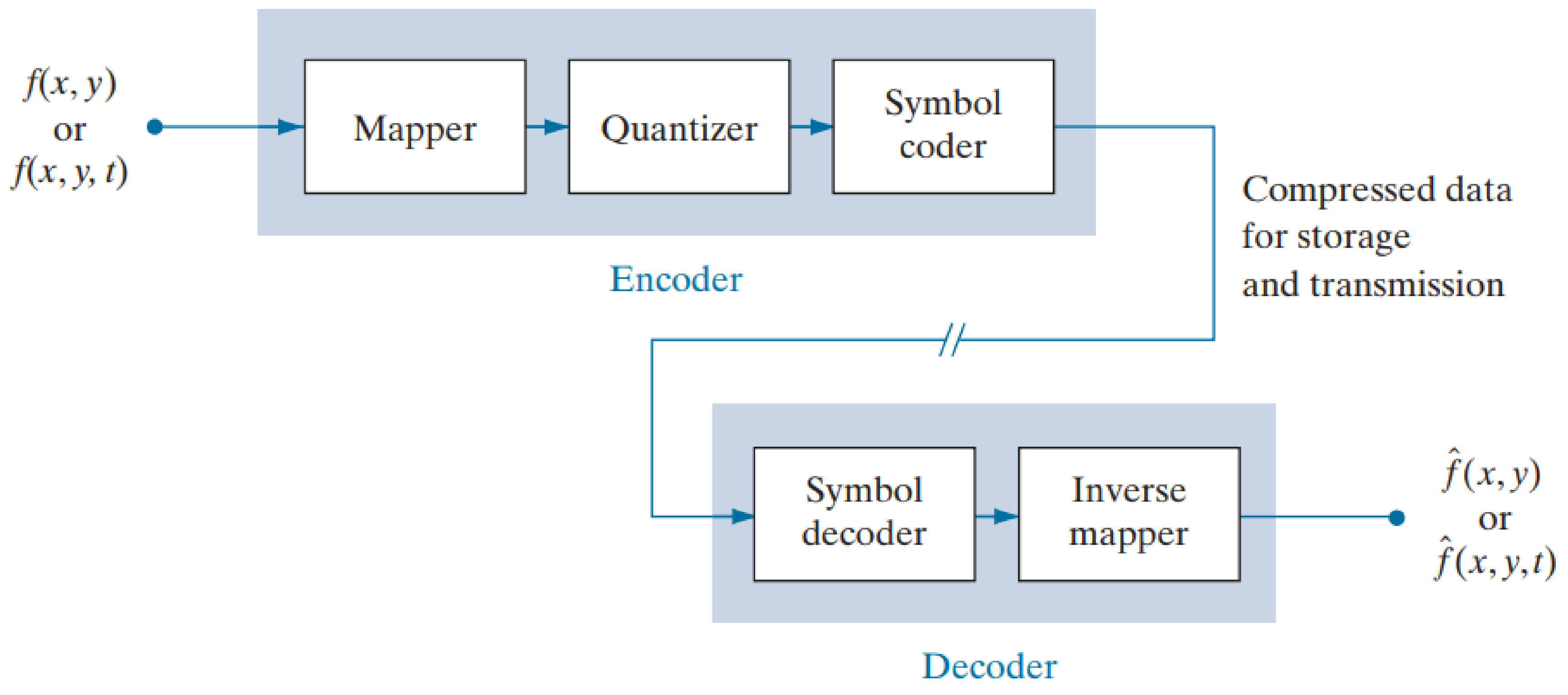
- rms value of the signal-to-noise ratio, denoted SNRrms, is obtained by taking the square root of SNR_{ms}

- subjective evaluations can be done by presenting a decompressed image to a cross section of viewers and averaging their evaluations.
- Side-by-side comparisons can be done with a scale such as {-3,-2,-1,0, 1, 2, 3} to represent the subjective evaluations {*much worse*, *worse*, *slightly worse*, *the same*, *slightly better*, *better*, *much better*}



Value	Rating	Description
1	Excellent	An image of extremely high quality, as good as you could desire.
2	Fine	An image of high quality, providing enjoyable viewing. Interference is not objectionable.
3	Passable	An image of acceptable quality. Interference is not objectionable.
4	Marginal	An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.
5	Inferior	A very poor image, but you could watch it. Objectionable interference is definitely present.
6	Unusable	An image so bad that you could not watch it.

General image compression system



$f(x, y)$ or $f(x, y, t)$
Mapper
Quantizer
Symbol coder
Encoder
Compressed data for storage and transmission
Symbol decoder
Inverse mapper
Decoder
$\hat{f}(x, y)$ or $\hat{f}(x, y, t)$

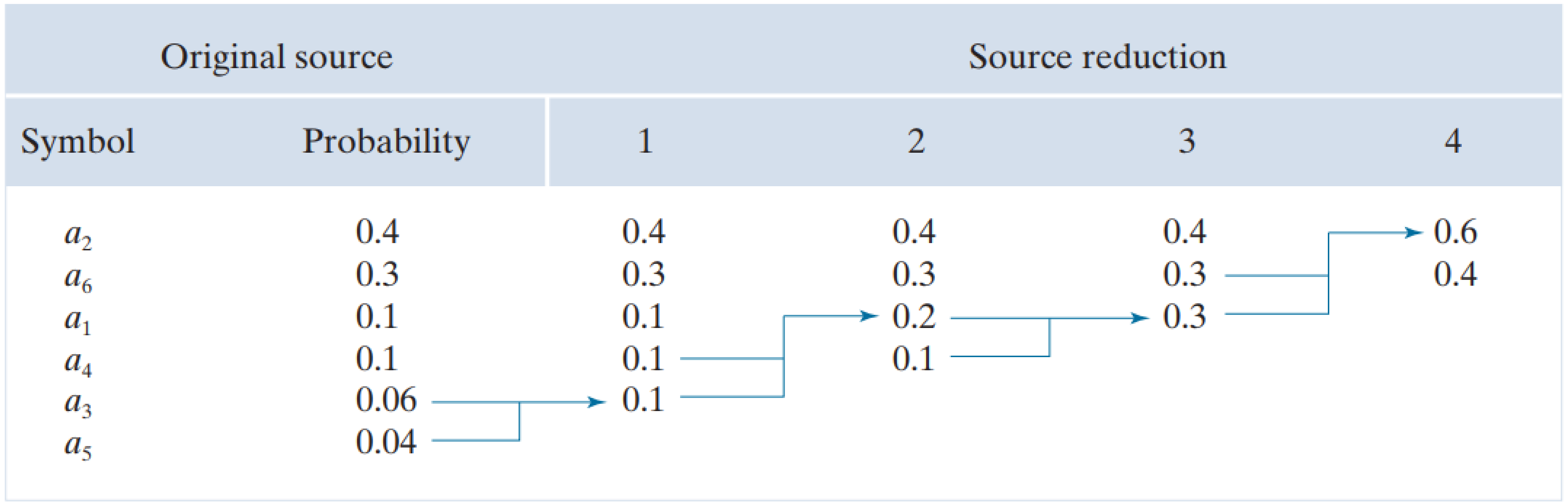
General image compression system

- Mapper transforms into a (usually nonvisual) format designed to reduce spatial and temporal redundancy.
- Quantizer reduces the accuracy of the mapper's output in accordance with a pre-established fidelity criterion. The goal is to keep irrelevant information out of the compressed representation. As noted earlier, this operation is irreversible
- Symbol coder generates a fixed- or variable-length code to represent the quantizer output

Huffman Coding

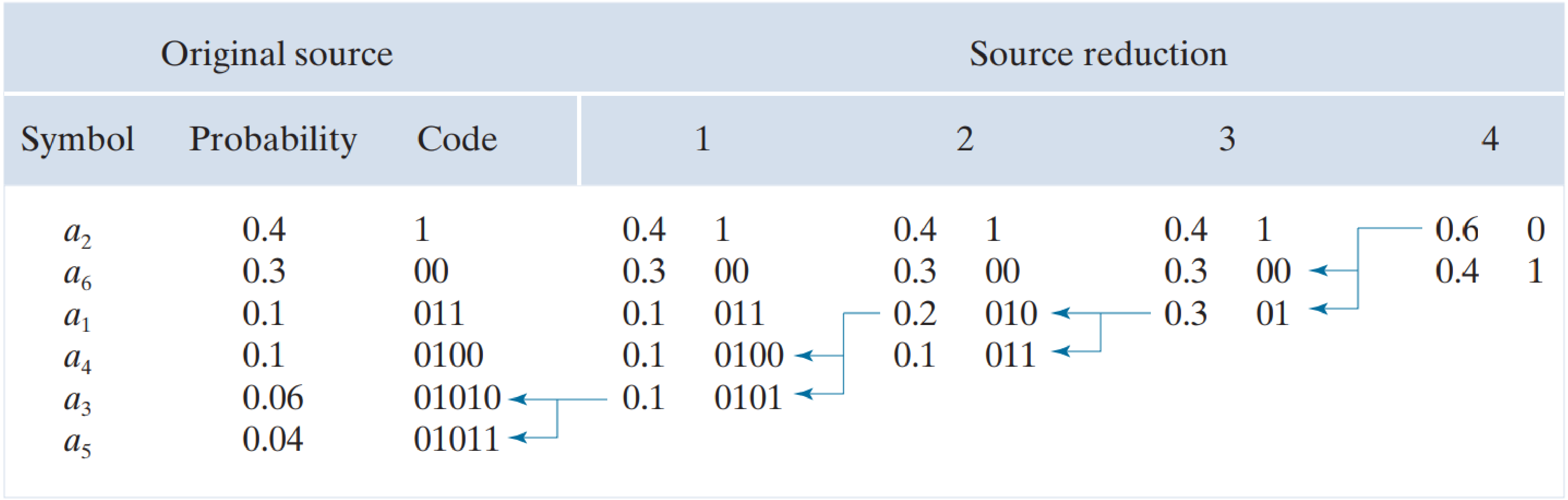
r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_{87} = 87$	0.25	01010111	8	01	2
$r_{128} = 128$	0.47	01010111	8	1	1
$r_{186} = 186$	0.25	01010111	8	000	3
$r_{255} = 255$	0.03	01010111	8	001	3
r_k for $k = 87, 128, 186, 255$	0	—	8	—	0

Huffman Coding



Original source		Source reduction			
Symbol	Probability	1	2	3	4
a_2	0.4	0.4	0.4	0.4	0.6
a_6	0.3	0.3	0.3	0.3	0.4
a_1	0.1	0.1	0.2	0.3	
a_4	0.1	0.1	0.1		
a_3	0.06	0.1			
a_5	0.04				

Huffman Code Assignment Procedure



Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

$$L_{\text{avg}} = (0.4)(1) + (0.3)(2) + (0.1)(3) + (0.1)(4) + (0.06)(5) + (0.04)(5)$$
$$= 2.2 \text{ bits/pixel}$$

- The higher the probability the shorter the code.
- Uniquely decodable
- Left-to-right scan
- 010100111100 = $a_3a_1a_2a_2a_6$

Golomb Coding

- Nonnegative integer inputs with exponentially decaying probability distributions.

Given a nonnegative integer n and a positive integer divisor m>0, the Golomb code of n with respect to m, denoted $G_m(n)$, is a combination of the unary code of quotient $\lfloor n/m \rfloor$ and the binary representation of remainder n mod m. $G_m(n)$ is constructed as follows:

1. Form **the unary code of quotient $\lfloor n/m \rfloor$** . (The unary code of an integer q is defined as q 1's followed by a 0.)

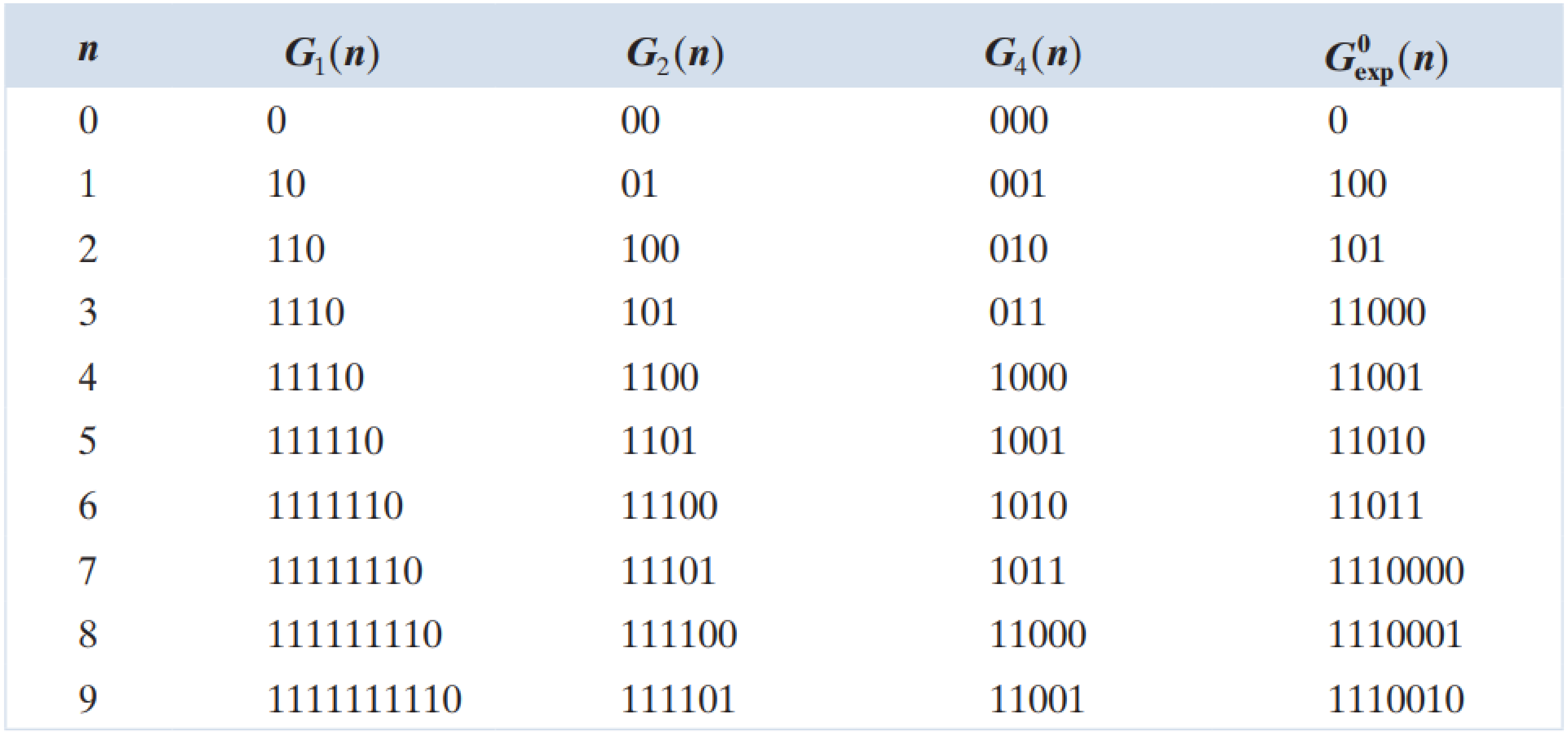
2. Let **$k = \lceil \log_2 m \rceil$ log , $c = 2^k - m$, $r = n \bmod m$** , and compute truncated **remainder r'** such that

$$r' = \begin{cases} r \text{ truncated to } k-1 \text{ bits} & 0 \le r < c \\ r + c \text{ truncated to } k \text{ bits} & \text{otherwise} \end{cases}$$

3. Concatenate the results of Steps 1 and 2

$$\boxed{\text{special case of } m = 2^k,\ c = 0 \text{ and } r' = r = n \bmod m \text{ truncated to } k \text{ bits}}$$

Several Golomb codes for the integers (0–9)



n	$G_1(n)$	$G_2(n)$	$G_4(n)$	$G_{\text{exp}}^{0}(n)$
0	0	00	000	0
1	10	01	001	100
2	110	100	010	101
3	1110	101	011	11000
4	11110	1100	1000	11001
5	111110	1101	1001	11010
6	1111110	11100	1010	11011
7	11111110	11101	1011	1110000
8	111111110	111100	11000	1110001
9	1111111110	111101	11001	1110010

- There are many Golomb codes to choose from, **a key step in their effective application is the selection of divisor *m*.**
- When the integers to be represented are *geometrically* distributed with a *probability mass function* (PMF)

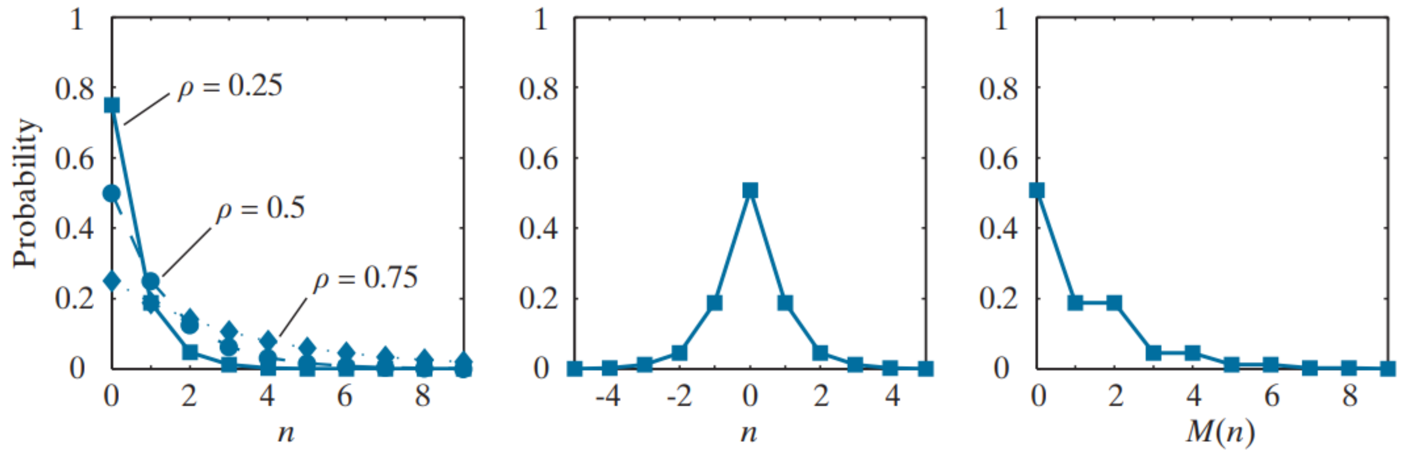
$$P(n) = (1-\rho)\rho^n$$

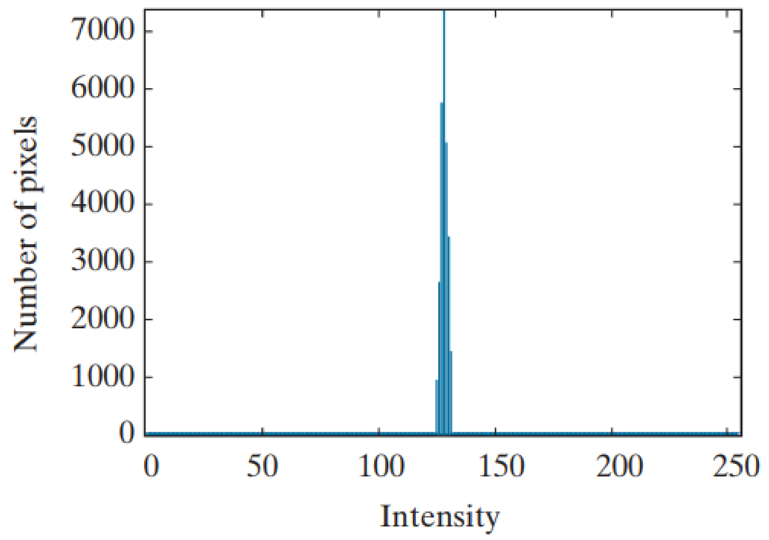
- for some Golomb codes can be shown to be optimal in the sense that $G_m(n)$ provides the shortest average co $0 < \rho < 1$, of all uniquely decipherable codes when

$$m = \left\lceil \frac{\log_2(1+\rho)}{\log_2(1/\rho)} \right\rceil$$

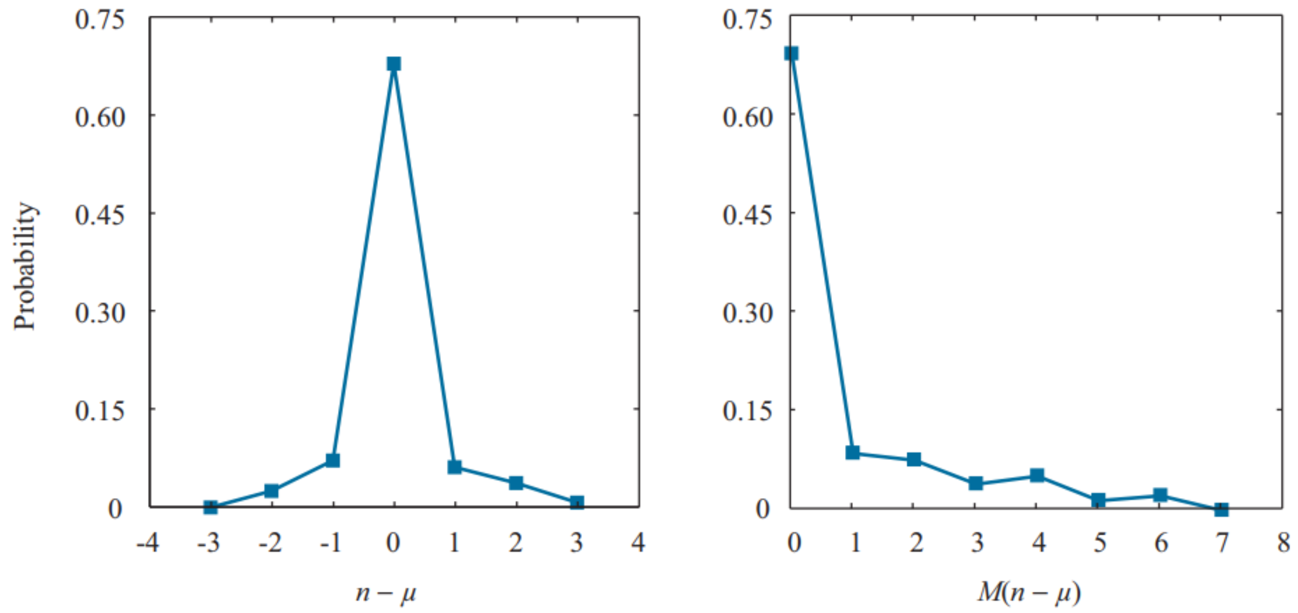
- To handle negative differences in Golomb coding, which can only represent nonnegative integers, a mapping like

$$M(n) = \begin{cases} 2n & n \geq 0 \\ 2|n| - 1 & n < 0 \end{cases}$$





Number of pixels
Intensity



Probability
$n - \mu$
$M(n - \mu)$

Zeroth-order exponential Golomb code

- An order-*k* exponential Golomb code $\mathbf{G^{k}_{exp}(n)}$ is computed as follows:

1. Find an integer $i \geq 0$ such that

$$\sum_{j=0}^{i-1} 2^{j+k} \leq n < \sum_{j=0}^{i} 2^{j+k} \tag{8-16}$$

   and form the unary code of i. If $k = 0$, $i = \lfloor \log_2 (n+1) \rfloor$ and the code is also known as the *Elias gamma code*.
2. Truncate the binary representation of

$$n - \sum_{j=0}^{i-1} 2^{j+k} \tag{8-17}$$

   to $k + i$ least significant bits.
3. Concatenate the results of Steps 1 and 2.

To find $G_{\exp}^{0}(8)$, for example, we let $i = \lfloor \log_2 9 \rfloor$ or 3 in Step 1 because $k = 0$. Equation (8-16) is then satisfied because

$$\sum_{j=0}^{3-1} 2^{j+0} \le 8 < \sum_{j=0}^{3} 2^{j+0}$$

$$\sum_{j=0}^{2} 2^{j} \le 8 < \sum_{j=0}^{3} 2^{j}$$

$$2^0 + 2^1 + 2^2 \le 8 < 2^0 + 2^1 + 2^2 + 2^3$$
$$7 \le 8 < 15$$

The unary code of 3 is 1110 and Eq. (8-17) of Step 2 yields

$$8 - \sum_{j=0}^{3-1} 2^{j+0} = 8 - \sum_{j=0}^{2} 2^{j} = 8 - \left(2^0 + 2^1 + 2^2\right) = 8 - 7 = 1 = 0001$$

which when truncated to its 3 + 0 least significant bits becomes 001. The concatenation of the results from Steps 1 and 2 then yields 1110001.

Bit-plane Coding

- The intensities of an m-bit monochrome image can be represented in the form of the base-2 polynomial

$$a_{m-1}2^{m-1} + a_{m-2}2^{m-2} + \ldots + a_1 2^1 + a_0 2^0$$

- A simple method of decomposing the image into a collection of binary images is to separate the m coefficients of the polynomial into m 1-bit bit planes.

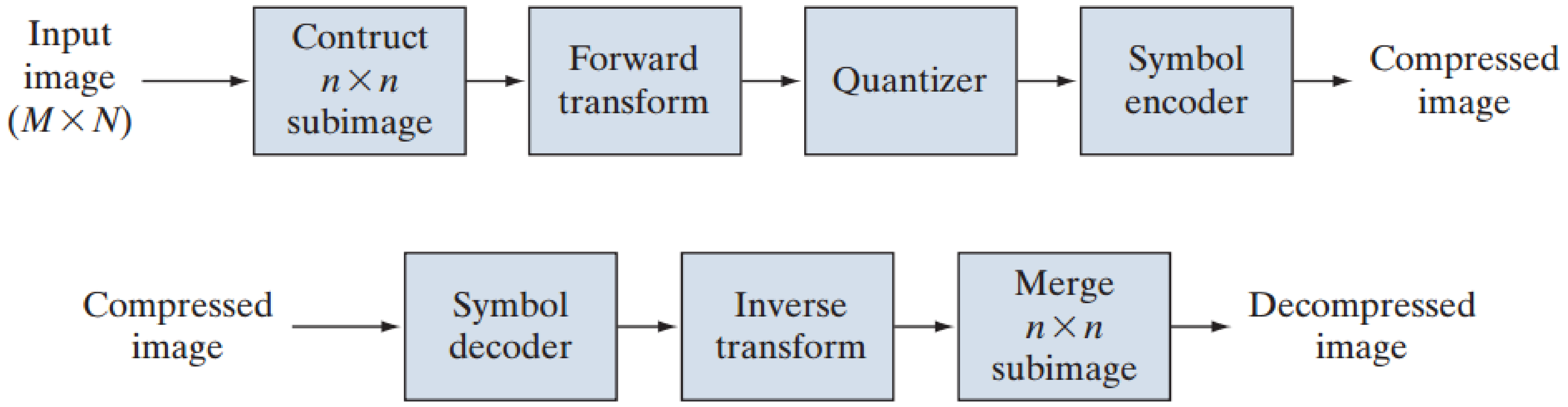
- Disadvantage : small changes in intensity can have a significant impact on the complexity of the bit planes. **127 (01111111) is adjacent to a pixel of intensity 128 (10000000), for instance, every bit plane will contain a corresponding 0 to 1 (or 1 to 0) transition**.

- Alternate Approach: first represent the image by an m-bit Gray code. The m-bit Gray code $g_{m-1}\ldots g_2 g_1 g_0$ that corresponds to the polynomial:

$$g_i = a_i \oplus a_{i+1} \qquad 0 \le i \le m-2$$
$$g_{m-1} = a_{m-1}$$

- For instance, when intensity levels 127 and 128 are adjacent, only the highest-order bit plane will contain a 0 to 1 transition, because the Gray codes that correspond to 127 and 128 are 01000000 and 11000000, respectively

Block Transform Coding



Transform Selection

- Consider a sub image g(x,y) of size n*n whose forward discrete transform, T(u,v)

$$T(u, v) = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} g(x, y) r(x, y, u, v)$$

- Inverse discrete Transform

$$g(x, y) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) s(x, y, u, v)$$

- In these equations, and r(x,y,u,v) and s(x,y,u,v) are called the *forward* and *inverse transformation kernels*, respectively.

- The best known transformation kernel pair is

$$r(x, y, u, v) = e^{-j2\pi(ux+vy)/n}$$

$$s(x, y, u, v) = \frac{1}{n^2} e^{j2\pi(ux+vy)/n}$$

- One of the transformations used most frequently for image compression is the *discrete cosine transform* (DCT).

$$\begin{aligned} r(x, y, u, v) &= s(x, y, u, v) \\ &= \alpha(u)\alpha(v) \cos\left[\frac{(2x+1)u\pi}{2n}\right] \cos\left[\frac{(2y+1)v\pi}{2n}\right] \end{aligned}$$

where

$$\alpha(u) = \begin{cases} \sqrt{\dfrac{1}{n}} & \text{for } u = 0 \\ \sqrt{\dfrac{2}{n}} & \text{for } u = 1, 2, \ldots, n-1 \end{cases}$$

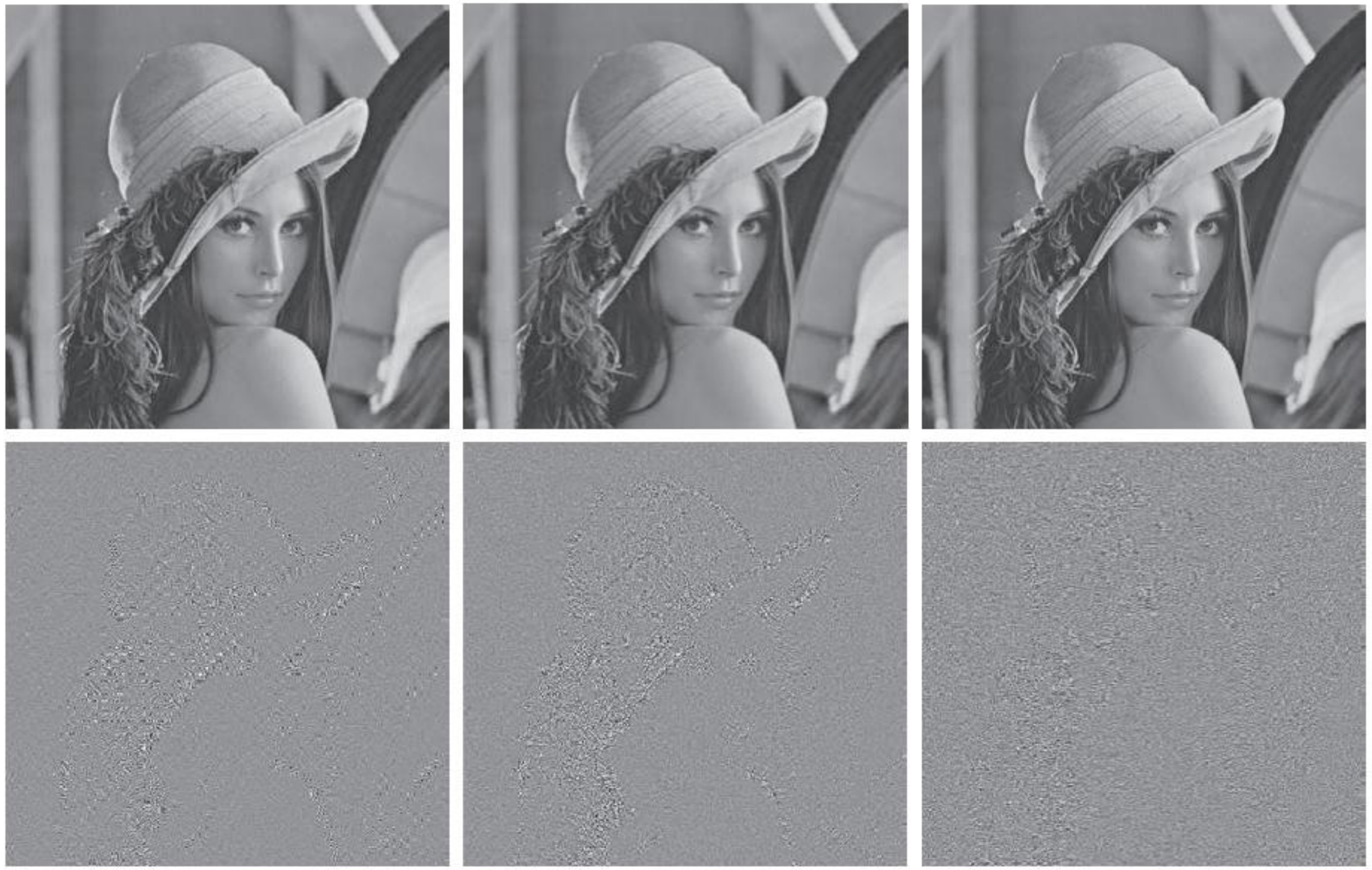


FIGURE 8.22 Approximations of Fig. 8.9(a) using the (a) Fourier, (b) Walsh-Hadamard, and (c) cosine transforms, together with the corresponding scaled error images in (d)–(f).

The actual rms errors were 2.32, 1.78, and 1.13 intensities, respectively