Chapter 8 Image Compression

第八章　图像压缩

引言

考虑大小为780×480 ×3，2小时的标准清晰度视频，30帧/秒速率，每帧图片

$$(780 \times 480) \times 30 \times 3 = 31\ 104\ 000 \text{字节}$$

2小时的电影

$$31\ 104\ 000 \times 3600 \times 2 \approx 2.24 \times 10^{11} \text{字节}$$

224GB（千兆字节），需要27张8.5GB的双层DVD来存储。对于高清视频如1920×1080，占用空间更大。

8.1 基础知识

数据间冗余是数字图像压缩的主要动机。

n1和n2代表两个代表相同信息的两种表示中的比特数

C_R 为压缩率=n1/n2

R_D为相对数据冗余=1-1/ C_R

- 相对数据冗余和压缩率的一些特例

n1相对于n2	C_R	R_D	对应的情况
n1 = n2	1	0	第1种表达相对第2种表达不含冗余数据
n1 >> n2	$\to\infty$	$\to1$	第1种数据集合包含相当多的冗余数据
n1 << n2	$\to0$	$\to\infty$	第2种数据集合包含相当多的冗余数据

8.1 基础知识

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.)

8.1.1 编码冗余

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

TABLE 8.1 Example of variable-length coding.

每像素编码后平均长度

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

压缩率和相关冗余分别为

$$C = \frac{8}{1.81} \approx 4.42$$

$$R = 1 - \frac{1}{4.42} = 0.774$$

8.1.2 空间冗余和时间冗余

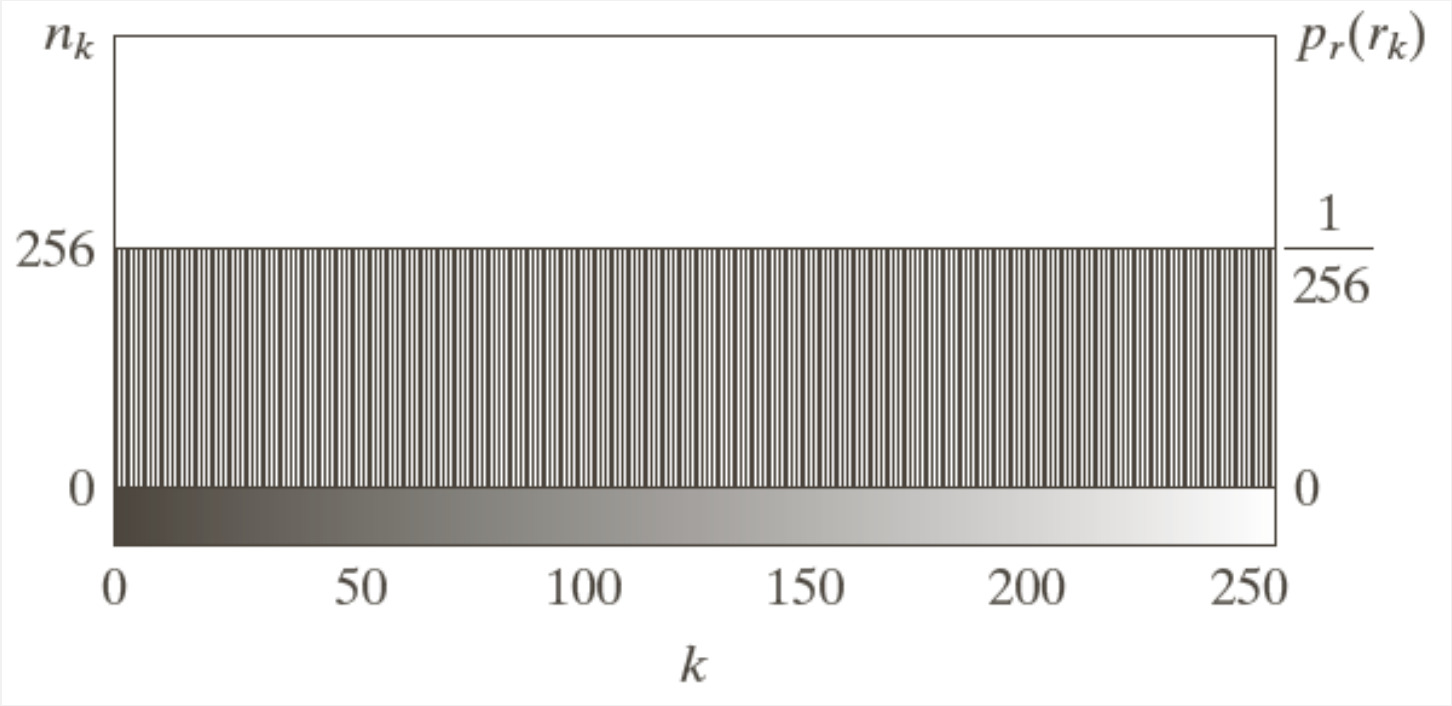


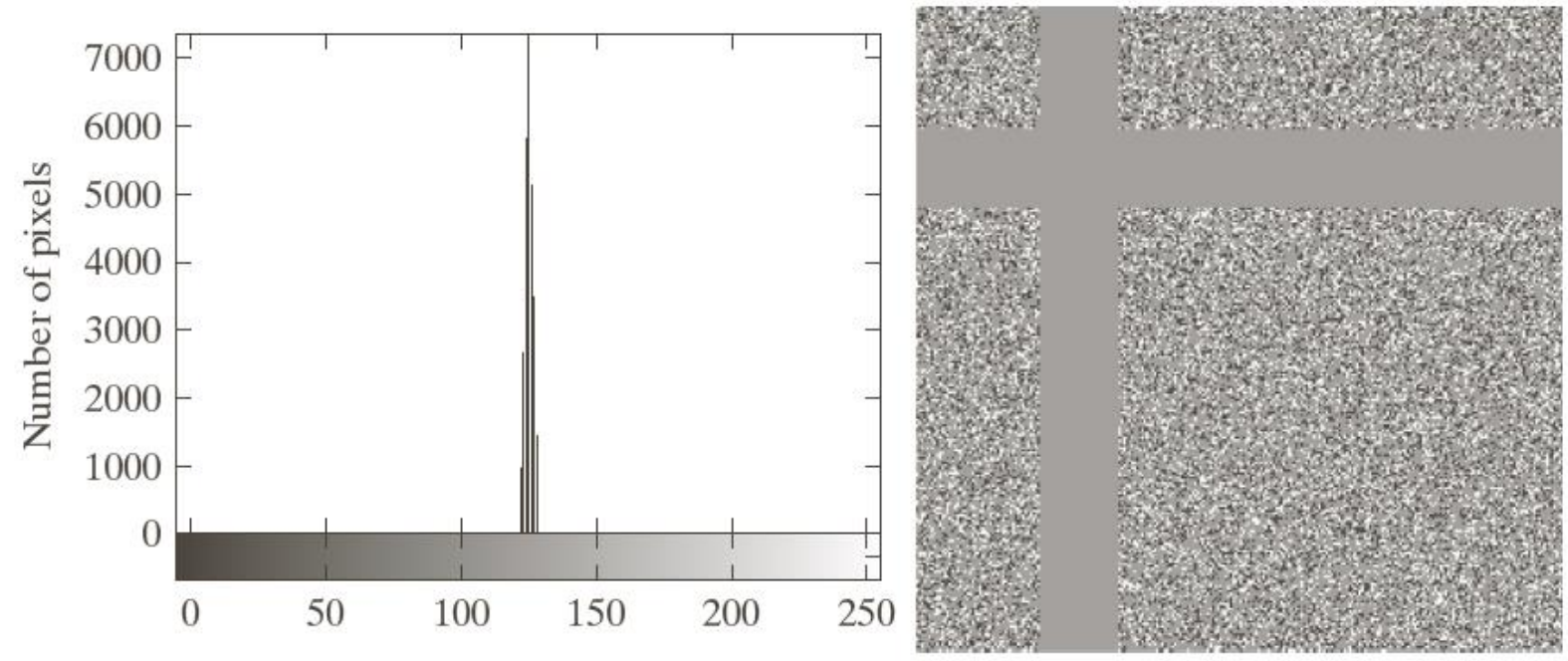
FIGURE 8.2 The intensity histogram of the image in Fig. 8.1(b).

基于行程对（游程对）的方式来表达图**8.1(b)**的图像。游程队中第一个单元为灰度值，第二个单元是具有该灰度值的像素的个数。故图**8.1(b)**的压缩率可以达到**(256×256×8)/((256+256) ×8)=128:1.**

原始图像尺寸

压缩图像尺寸

8.1.3 不相关信息



a b

FIGURE 8.3
(a) Histogram of the image in Fig. 8.1(c) and (b) a histogram equalized version of the image.

8.1.4 图像信息的度量

发生概率为$P(E)$的随机事件E被认为是包含：

$$I(E) = log\frac{1}{P(E)} = -logP(E)$$

的信息单元。

一个可能的离散集合$\{a_1, a_2, \ldots, a_J\}$,其对应的概率$\{P(a_1), P(a_2), \ldots, P(a_J)\}$, 熵的定义为：

$$H = -\sum_{j=1}^{J} P(a_j)log(P(a_j))$$

对图像而言，其熵可以表示为

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

熵越小，通常越有利于图像压缩

图像被编码后，每个像素所需要的bit数不小于$\tilde{H}$。

8.1.4 图像信息的度量

例 8.2　图像的熵的估计。

图 8.1(a)所示图像的熵可通过将表 8.1 中的灰度概率代入式(8.1-7)中来估计：

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

以类似的方法，图 8.1(b)和(c)中图像的熵可以分别表示为 8 比特/像素和 1.566 比特/像素。注意，图 8.1(a)中的图像似乎具有更多的视觉信息，但计算出的熵却几乎是最低的——1.66 比特/像素。图 8.1(b)中图像的熵几乎是图 8.1(a)中图像的熵的 5 倍，但似乎有相同(或更少)的视觉信息；图 8.1(c)中似乎有很少的信息或没有信息，却具有与图 8.1(a)中图像几乎相同的熵。因而明显的结论是一幅图像中熵的数量和信息与直觉相差甚远。

8.1.5 保真度准则

1、客观保真准则

$f(x,y)$表示输入图像，$\hat{f}(x,y)$ 表示由对输入先压缩后解压缩的得到的估计量，则输出图像的均方信噪比表示：

$$MSE = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left(\hat{f}(x,y)-f(x,y)\right)^2$$

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

$$PSNR = 10\times log_{10}\left(\frac{(2^n-1)^2}{MSE}\right)$$

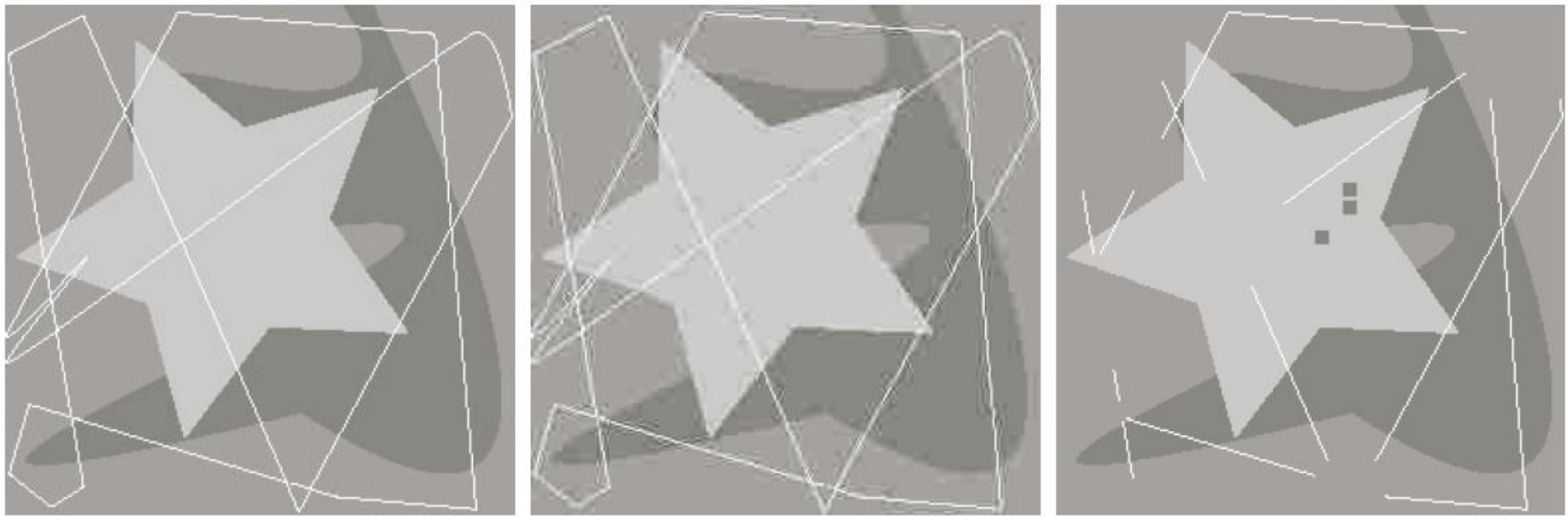
8.1.5 保真度准则

2、主观保真准则

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.

TABLE 8.2
Rating scale of the Television Allocations Study Organization. (Frendendall and Behrend.)

8.1.5 保真度准则



a b c

FIGURE 8.4 Three approximations of the image in Fig. 8.1(a).

主观评价顺序：**abc**
客观评价顺序**(SNR)**：**acb**

8.1.5 保真度准则

	
PSNR(dB)	PSNR(dB)
23.5084	23.2079

8.1.6 图像压缩模型

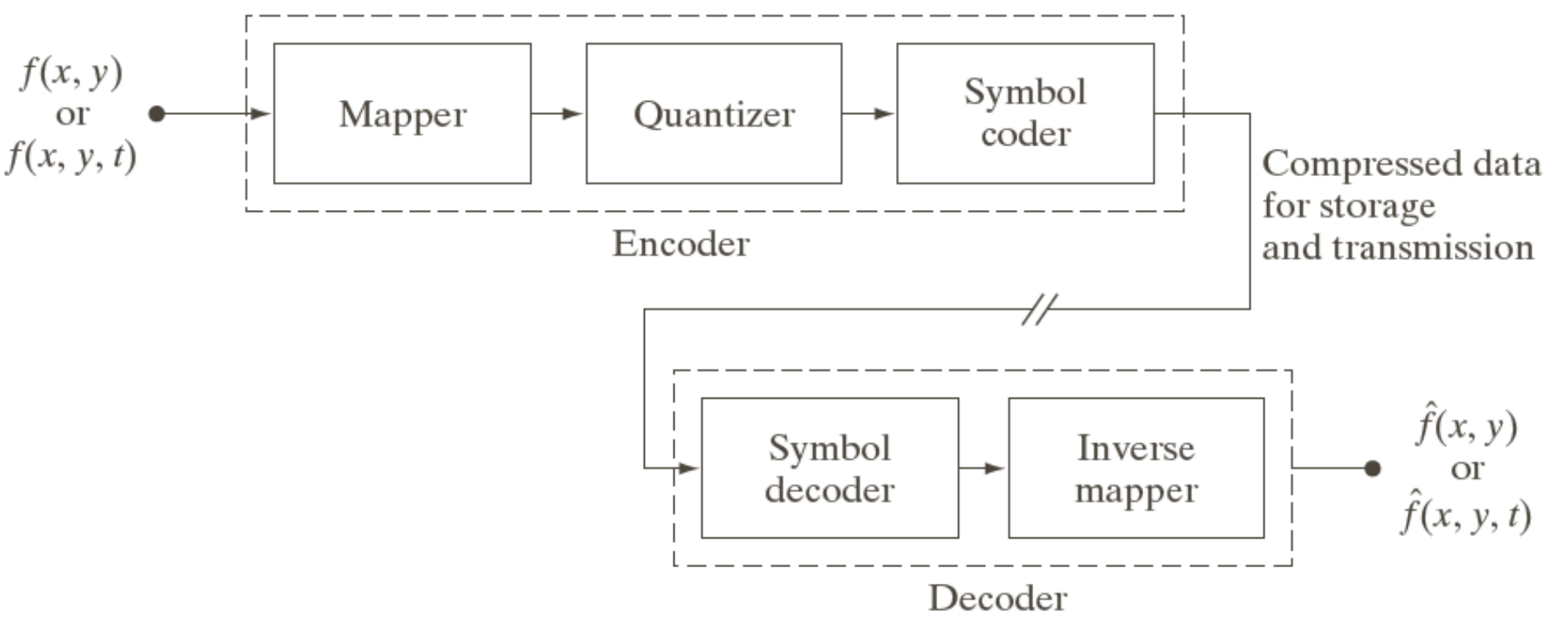


FIGURE 8.5 Functional block diagram of a general image compression system.

8.1.7 图像格式、容器和压缩标准

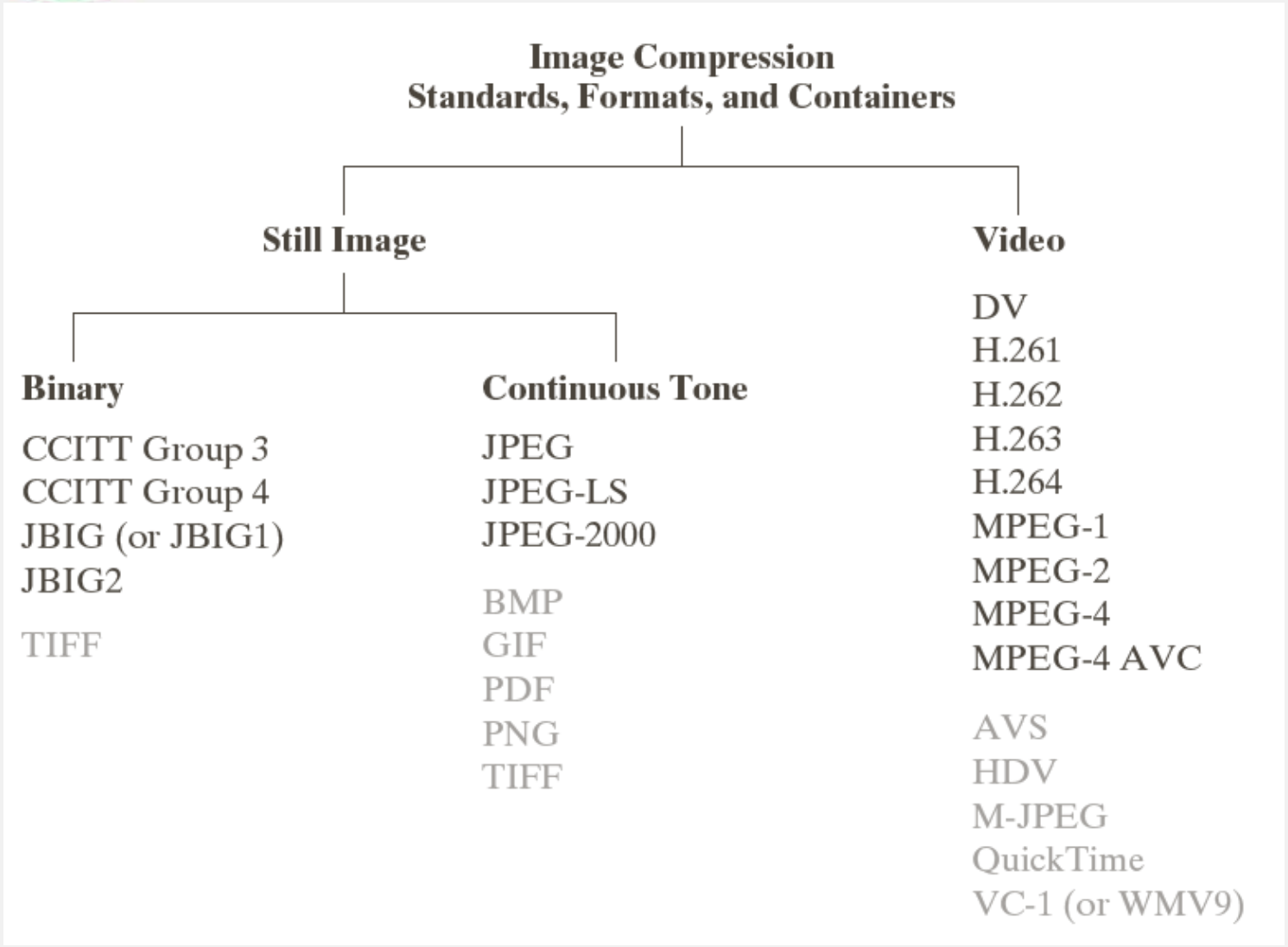


FIGURE 8.6 Some popular image compression standards, file formats, and containers. Internationally sanctioned entries are shown in black; all others are grayed.

8.2 一些基本的压缩方法

图像压缩技术包含两种；

- 无损压缩（哈夫曼编码属于无损压缩编码）
- 有损压缩（主要是量化过程中）

8.2.1 哈夫曼编码

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				

FIGURE 8.7 Huffman source reductions.

8.2.1 哈夫曼编码

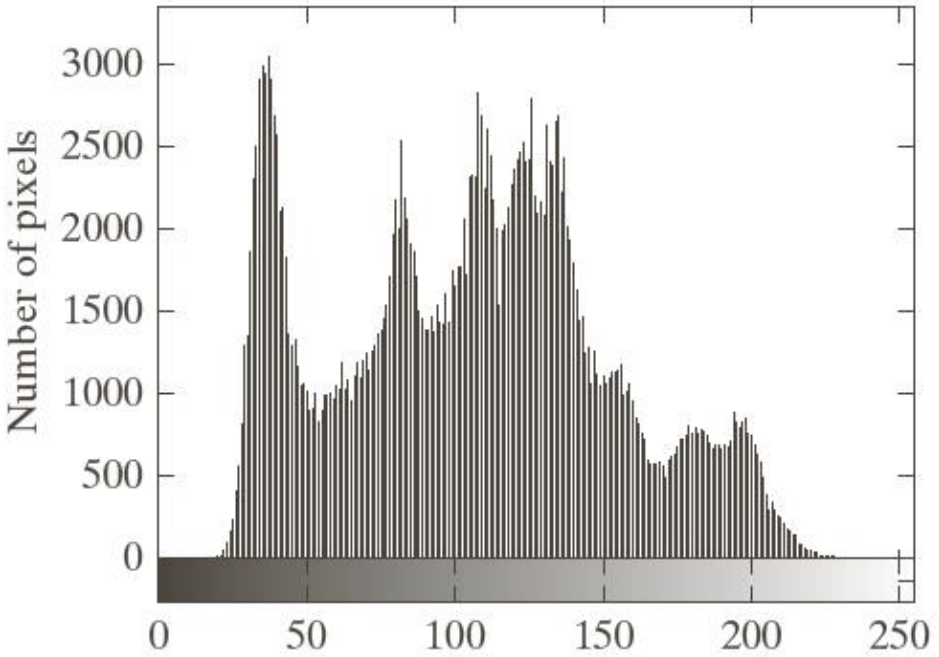
Original source			Source reduction							
Symbol	Probability	Code	1		2		3		4	
a_2	0.4	1	0.4	1	0.4	1	0.4	1	0.6	0
a_6	0.3	00	0.3	00	0.3	00	0.3	00	0.4	1
a_1	0.1	011	0.1	011	0.2	010	0.3	01		
a_4	0.1	0100	0.1	0100	0.1	011				
a_3	0.06	01010	0.1	0101						
a_5	0.04	01011								

FIGURE 8.8 Huffman code assignment procedure.

平均码长：

0.4×1+0.3×2+0.1×3+0.1×4+0.06×5 +0.04×5 = 2.2比特/像素

8.2.1 哈夫曼编码



a b

FIGURE 8.9 (a) A 512 × 512 8-bit image, and (b) its histogram.

8.2.2 Golomb编码

Golomb编码是针对（具有指数衰减概率分布的）非负整数的编码，在计算上比Huffman简单，且在Shannon第一定理的基础上是最优的。

给定一个非负整数n和一个正整数m, n关于m的Golomb编码$G_m(n)$是$\lfloor n/m \rfloor$和$n \bmod m$的二进制表示的组合。$G_m(n)$构建如下：

步骤一：完成关于$\lfloor n/m \rfloor$的一元编码（整数q的一元编码定义为q个1后面紧跟一个0）；

步骤二：令$k = \lceil log_2 m \rceil, c = 2^k - m, r = n \bmod m$,计算截断后的余数$r'$如下：

$$r' = \begin{cases} r\text{截断至}k-1\text{比特}, & 0 \le r < c \\ r + c\text{截断至}k\text{比特}, & \text{其他} \end{cases} \quad (8.2\text{-}1)$$

步骤三：链接步骤一和步骤二的结果。

例如：计算$G_4(9)$，第一步$\lfloor 9/4 \rfloor = 2$，得到一元编码为110；

第二步, $k = \lceil log_2 4 \rceil = 2, c = 2^2 - 4 = 0, r = 9 \bmod 4 = 1$(转换为二进制为0001)，根据(8.2-1)，截断至2比特，即01。

第三步：连接步骤一和步骤二的结果，得到11001，即为$G_4(9)$。

8. 2. 2 Golomb编码

n	$G_1(n)$	$G_2(n)$	$G_4(n)$	$G^0_{\exp}(n)$
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

TABLE 8.5 Several Golomb codes for the integers 0 – 9.

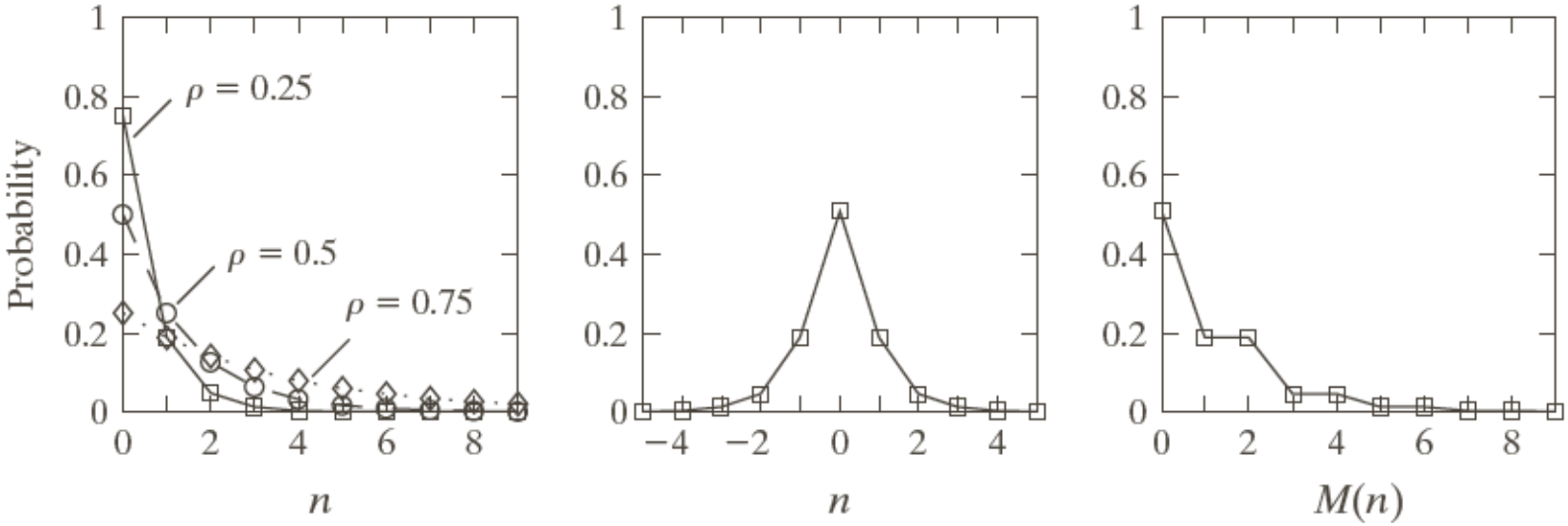
8. 2. 2 Golomb编码

Golomb只能用于表示非负整数，并且有许多Golomb码可供选择，在应用中一个关键步骤是关于除数m的选择。通常，当所需要编码的整数具有如下PMF(probability mass function)

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

对部分$1 < \rho < 1$，Golomb码$G_m(n)$可以达到最优—即平均码长最短，此时m的取值为：

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



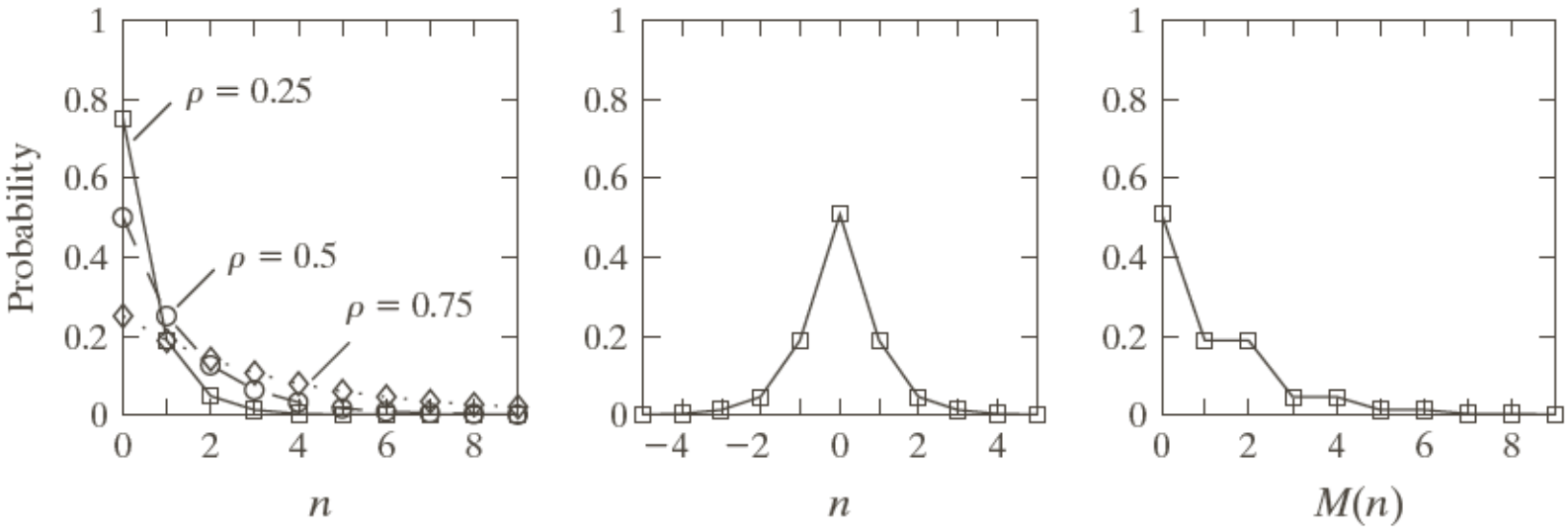
a b c

FIGURE 8.10 (a) Three one-sided geometric distributions from Eq. (8.2-2); (b) a two-sided exponentially decaying distribution; and (c) a reordered version of (b) using Eq. (8.2-4).

8. 2. 2 Golomb编码

但图像直方图通常不满足上述分布，故Golomb一般用于对差分信号进行编码。然而差分信号存在负值，故需要进一步变换

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



a b c

FIGURE 8.10 (a) Three one-sided geometric distributions from Eq. (8.2-2); (b) a two-sided exponentially decaying distribution; and (c) a reordered version of (b) using Eq. (8.2-4).

8.2.3 算术编码

非分块码，给信源符号的整个序列分配一个单一的算术码字。具体如下：

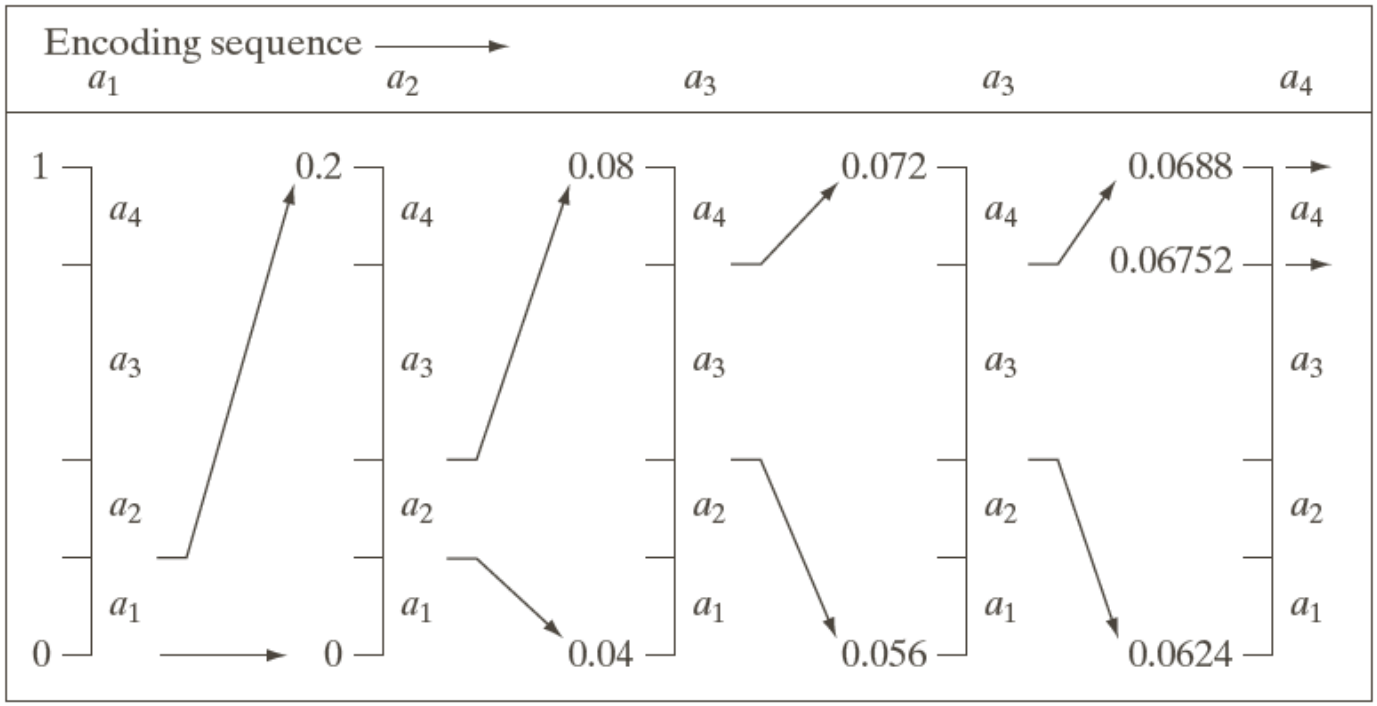


FIGURE 8.12 Arithmetic coding procedure.

Source Symbol	Probability	Initial Subinterval
a_1	0.2	[0.0, 0.2)
a_2	0.2	[0.2, 0.4)
a_3	0.4	[0.4, 0.8)
a_4	0.2	[0.8, 1.0)

TABLE 8.6 Arithmetic coding example.

8.2.3 算术编码

例1：假设信源符号为{A， B， C， D}，这些符号的概率分别为{ 0.1， 0.4， 0.2， 0.3 }，根据这些概率可把间隔[0， 1]分成4个子间隔：[0， 0.1]， [0.1， 0.5]， [0.5， 0.7]， [0.7， 1]，其中[x，y)表示半开放间隔，即包含x不包含y。

符号	A	B	C	D
概率	0.1	0.4	0.2	0.3
初始编码间隔	[0， 0.1)	[0.1， 0.5)	[0.5， 0.7)	[0.7， 1]

如果二进制消息序列的输入为：C A D A C D B。编码时首先输入的符号是C，找到它的编码范围是[0.5，0.7]。由于消息中第二个符号A的编码范围是[0， 0.1]，因此它的间隔就取[0.5， 0.7]的第一个十分之一作为新间隔[0.5，0.52]。依此类推，编码第3个符号D时取新间隔为[0.514， 0.52]，编码第4个符号A时，取新间隔为[0.514， 0.5146]，…。消息的编码输出可以是最后一个间隔中的任意数。

8.2.3 算术编码

编码过程：

步骤	输入符号	编码间隔	编码判决
1	C	[0.5， 0.7]	符号的间隔范围[0.5，0.7]
2	A	[0.5， 0.52]	[0.5， 0.7]间隔的第一个1/10
3	D	[0.514， 0.52]	[0.5， 0.52]间隔的最后一个1/10
4	A	[0.514， 0.5146]	[0.514， 0.52]间隔的第一个1/10
5	C	[0.5143， 0.51442]	[0.514， 0.5146]间隔的第五个1/10开始，二个1/10
6	D	[0.514384， 0.51442]	[0.5143， 0.51442]间隔的最后3个1/10
7	B	[0.5143836， 0.514402]	[0.514384，0.51442]间隔的4个1/10，从第1个1/10开始
8	**从[0.5143876， 0.514402]中选择一个数作为输出：0.5143876**		

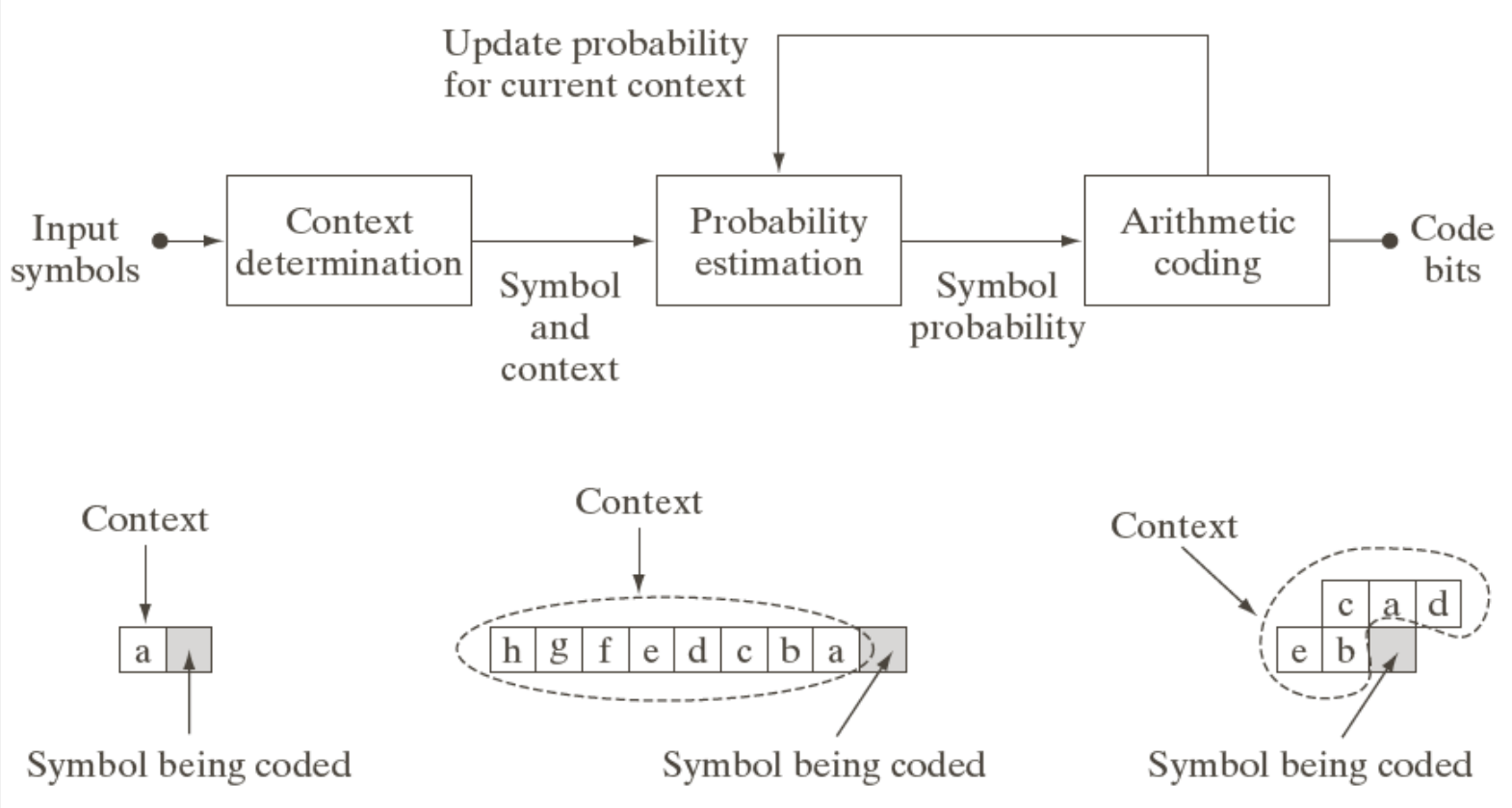
8.2.3 算术编码

解码过程：

步骤	间隔	译码符号	译码判决
1	[0.5， 0.7]	C	0.51439在间隔 [0.5， 0.7)
2	[0.5， 0.52]	A	0.51439在间隔 [0.5， 0.7)的第1个1/10
3	[0.514， 0.52]	D	0.51439在间隔[0.5， 0.52)的第7个1/10
4	[0.514， 0.5146]	A	0.51439在间隔[0.514， 0.52]的第1个1/10
5	[0.5143， 0.51442]	C	0.51439在间隔[0.514， 0.5146]的第5个1/10
6	[0.514384， 0.51442]	D	0.51439在间隔[0.5143， 0.51442]的第7个1/10
7	[0.51439， 0.5143948]	B	0.51439在间隔[0.51439，0.5143948]的第1个1/10
8	**译码的消息：C A D A C D B**		

8.2.3 算术编码

自适应算术编码：



a
b c d

FIGURE 8.13
(a) An adaptive, context-based arithmetic coding approach (often used for binary source symbols). (b)–(d) Three possible context models.

8.2.4 LZW编码

- 对信源符号的可变长度序列分配固定长度的码字,且不需要了解有关被编码符号的出现概率的知识.

- 基本思想是: 先构造一个对信源符号进行编码的编码本或“字典”,将输入字符串映射成定长的码字输出, 算法在产生输出串的同时更新编码表,这样编码表可以更好地适应所压缩图像的特殊性质.

8.2.5 行程编码

对二值图像压缩特别有效，具体实现如下：

- 用一长度序列表示图像或位平面的每一行，这些长度描绘了对黑色和白色像素的连续行程，这称为行程编码是传真编码的标准压缩方法

- 对从左到右扫描一行时所遇到的1或0的连接组，使用这些连接组的长度进行编码

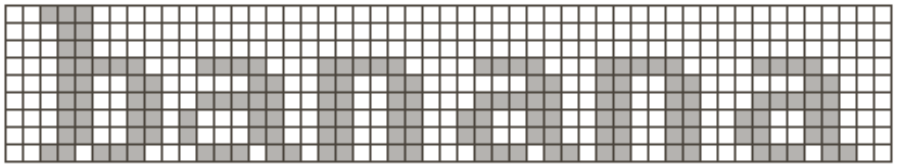
- 决定行程长度值的常用方法：
  1. 指定每一行第一次行程的值
  2. 假设每一行从白色行程开始

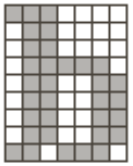
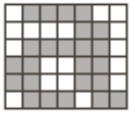
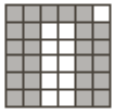
- 对行程本身进行变长编码可以实现额外的压缩，近似熵为：

$$H_{RL} = \frac{H_0 + H_1}{L_0 + L_1}$$

其中H_0，H_1表示黑色和白色行程的熵，L_0，L_1表示黑色和白色行程的均值。

8.2.6 基于符号的编码



Token	Symbol
0	
1	
2	

Triplet
(0, 2, 0) (3, 10, 1) (3, 18, 2) (3, 26, 1) (3, 34, 2) (3, 42, 1)

a b c

FIGURE 8.17
(a) A bi-level document, (b) symbol dictionary, and (c) the triplets used to locate the symbols in the document.

8.2.7 位平面编码

- 位平面编码：消除像素间冗余
- 将一幅图像分解为一系列二值图像并通过二值图像压缩方法对每幅二值图像进行压缩
- 位平面分解的两种方法

  二值图像位平面

  格雷编码位平面

二值图像位平面

- 一幅m比特的灰度图像具有的灰度级表示如下

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

- 零级位平面是通过收集每个像素的 a_0位生成，第(m-1)级位平面包含a_{m-1} 位

- 缺点：图像在灰度级上稍有变化就会对位平面的复杂性产生显著影响，如亮度127(01111111)和亮度128(10000000)的转换

格雷编码位平面

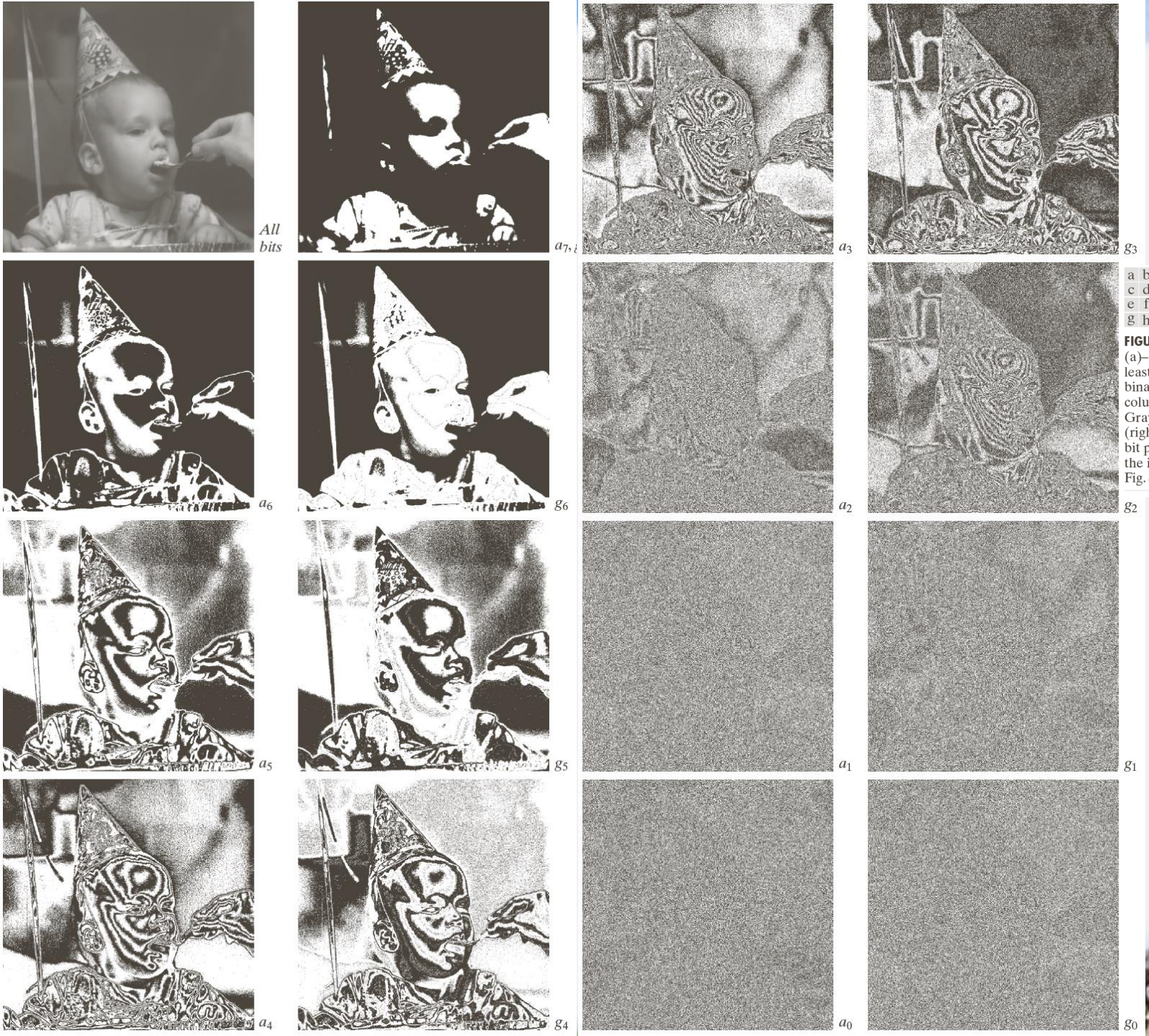
- 这种分解方法可以减少小的灰度级变化带来的影响，首先用一个**m**比特的灰度编码表示图像。这个**m**比特的灰度级编码 $g_{m-1}\cdots g_2 g_1 g_0$

- 图像的灰度编码根据下列方法得到：

$$g_{m-1} = a_{m-1}$$

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

- 避免二值图像位平面的问题，连续码字只在**1**位位置上不同,如亮度**127 (01000000)**和亮度**128(11000000)**的转换



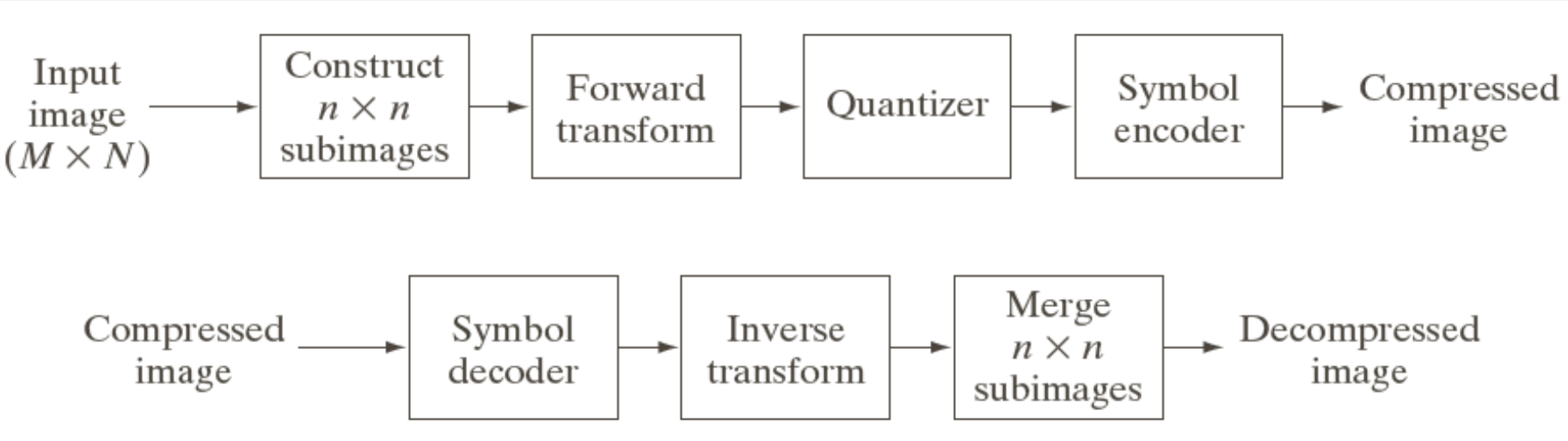
a b
c d
e f
g h

FIGURE 8.19
(a) A 256-bit monochrome image. (b)–(h) The four most significant binary and Gray-coded bit planes of the image in (a).

a b
c d
e f
g h

FIGURE 8.20
(a)–(h) The four least significant binary (left column) and Gray-coded (right column) bit planes of the image in Fig. 8.19(a).

8.2.8 块变换编码（JPEG压缩）



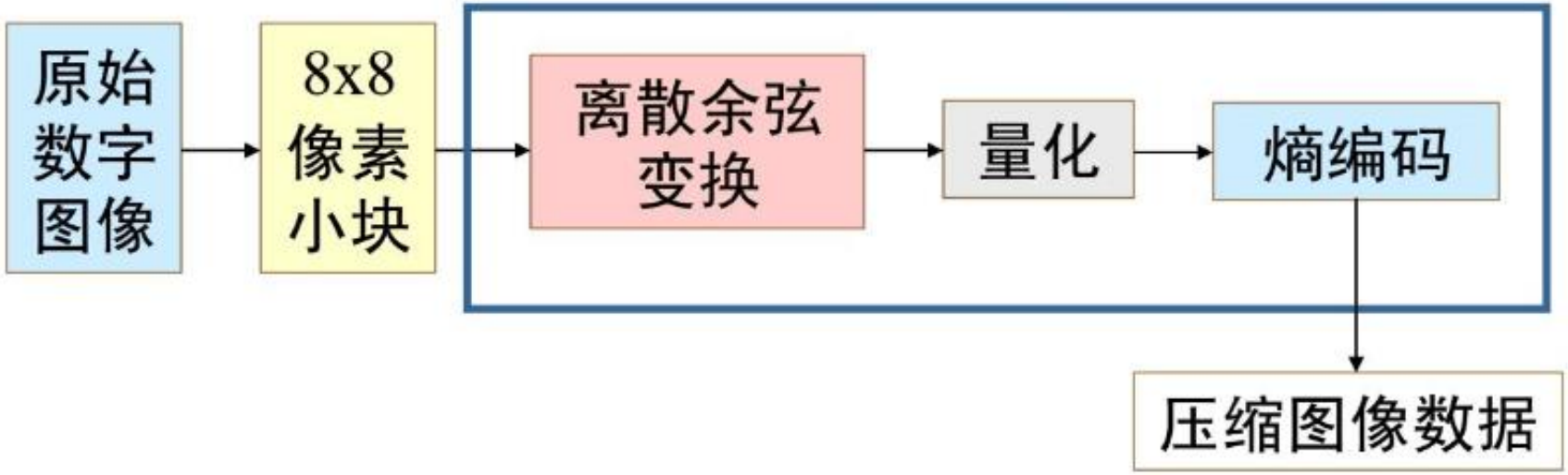
a
b

FIGURE 8.21
A block transform coding system:
(a) encoder;
(b) decoder.

8.2.8 块变换编码（JPEG压缩）

- JPEG编码：

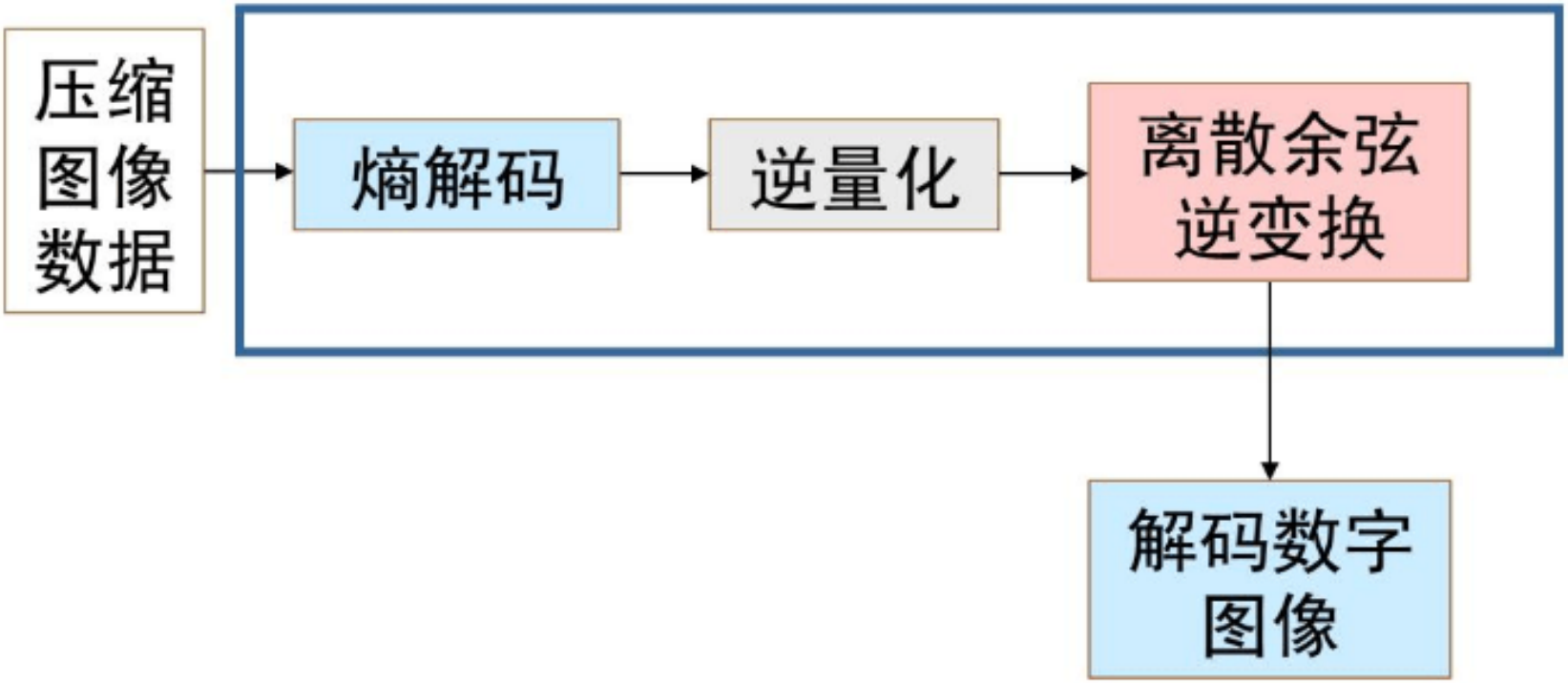
  原始图像数据➔压缩图像数据



8.2.8 JPEG压缩

- JPEG解码：

  压缩图像数据➔解压缩图像数据



8.2.8 JPEG压缩

JPEG采用的是YCrCb颜色空间，而BMP采用的是RGB颜色空间，要想对BMP图片进行压缩，首先需要进行颜色空间的转换。YCrCb颜色空间中，Y代表亮度，Cr,Cb则代表色调和饱和度(也有人将Cb,Cr两者统称为色度)，三者通常以Y,U,V来表示，即用U代表Cb，用V代表Cr。RGB和YCrCb之间的转换关系如下所示：

$$Y = 0.299R+0.587G+0.114B$$

$$Cb = -0.1687R-0.3313G+0.5B+128$$

$$Cr = 0.5R=0.418G-0.0813B+128$$

8.2.8 JPEG压缩

JPEG压缩编码算法的主要计算步骤

- (1) 正向离散余弦变换(FDCT)
- (2) 量化(quantization)
- (3) Z字形编码(zigzag scan)。
- (4) 使用差分脉冲编码调制(DPCM)对直流系数(DC)进行编码
- (5) 使用行程长度编码(RLE)对交流系数(AC)进行编码
- (6) 熵编码(entropy coding)

8.2.8 JPEG压缩

DCT变换使用下式计算

$$F(u,v)=\frac{1}{4}C(u)C(v)\left[\sum_{i=0}^{7}\sum_{j=0}^{7}f(i,j)\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}\right]$$

逆变换使用下式计算

$$F(i,j)=\frac{1}{4}C(u)C(v)\left[\sum_{u=0}^{7}\sum_{v=0}^{7}f(u,v)\cos\frac{(2i+1)u\pi}{16}\cos\frac{(2j+1)v\pi}{16}\right]$$

其中，$C(u),\ C(v)=1/\sqrt{2}$ 当 $u,\ v=0$；

$C(u),\ C(v)=1$ 其他

8.2.8 JPEG压缩

量化

- 对FDCT变换后的(频率的)系数进行量化
- 量化目的是降低非“0”系数的幅度以及增加“0”值系数的数目
- 用图5-4所示的均匀量化器量化
- 量化是造成图像质量下降的最主要原因
- 量化用右式计算

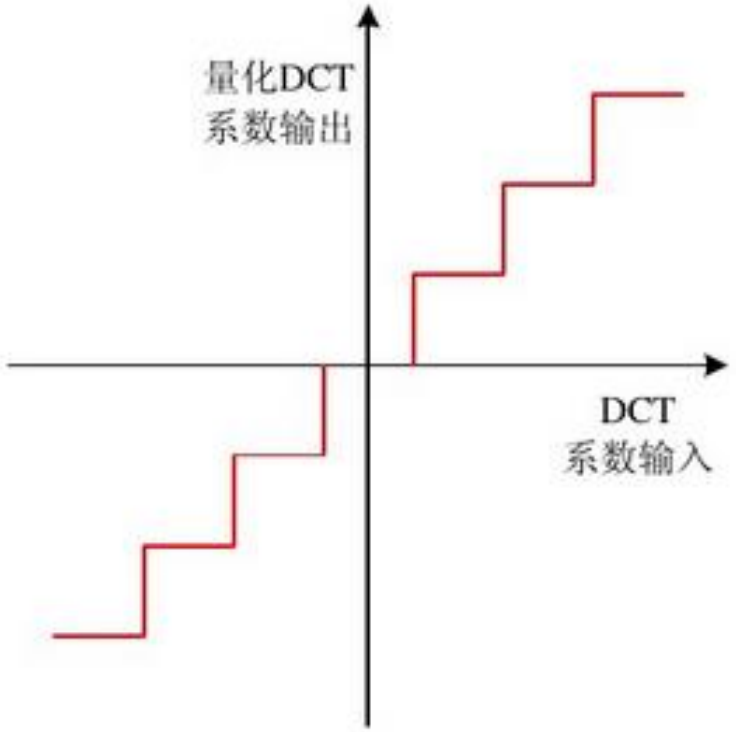


图5-4 均匀量化器

$$\hat{F}(u,v) = round(\frac{F(u,v)}{Q(u,v)})$$

8.2.8 JPEG压缩

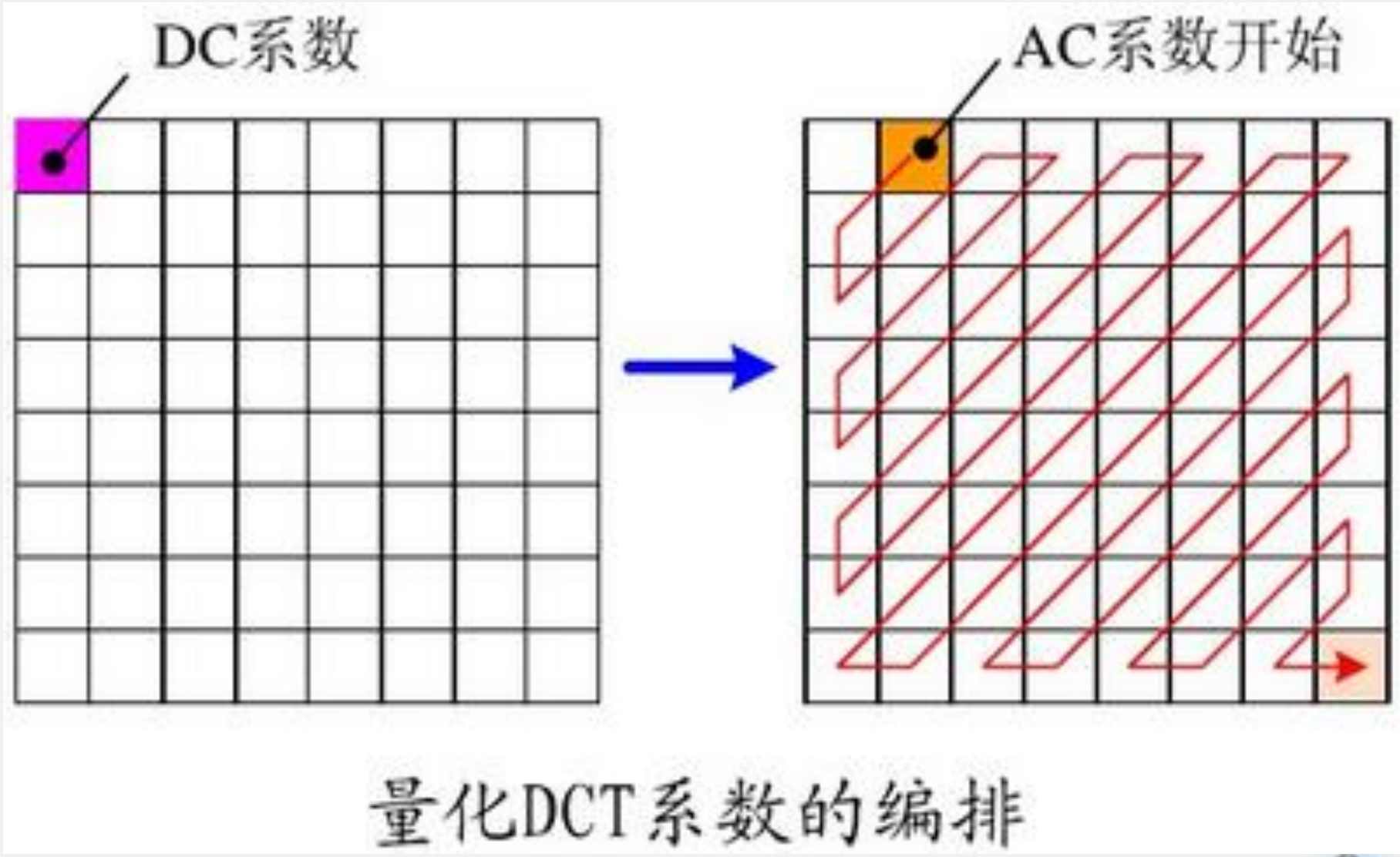
亮度量化表

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

色差量化表

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

8.2.8 JPEG压缩



8.2.8 JPEG压缩

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	63	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

EXAMPLE 8.17: JPEG baseline coding and decoding.

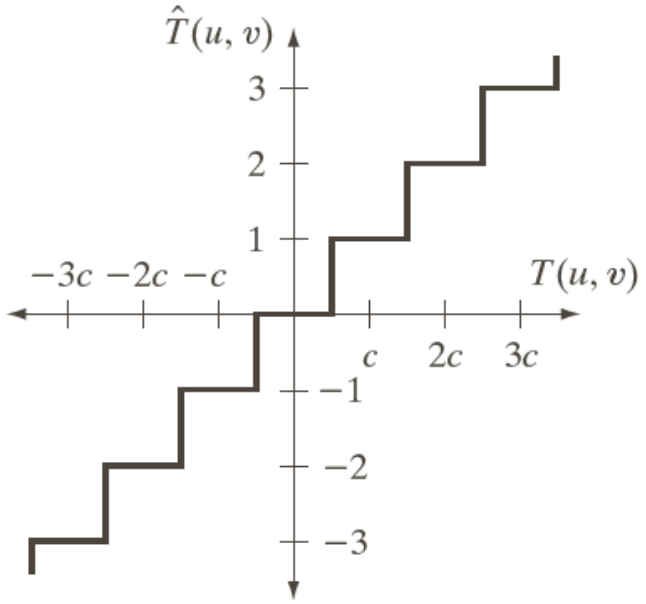
-128

−76	−73	−67	−62	−58	−67	−64	−55
−65	−69	−62	−38	−19	−43	−59	−56
−66	−69	−60	−15	16	−24	−62	−55
−65	−70	−57	−6	26	−22	−58	−59
−61	−67	−60	−24	−2	−40	−60	−58
−49	−63	−68	−58	−51	−65	−70	−53
−43	−57	−64	−69	−73	−67	−63	−45
−41	−49	−59	−60	−63	−52	−50	−34

FDCT

8.2.8 JPEG压缩

−415	−29	−62	25	55	−20	−1	3
7	−21	−62	9	11	−7	−6	6
−46	8	77	−25	−30	10	7	−5
−50	13	35	−15	−9	6	0	3
11	−8	−13	−2	−1	1	−4	1
−10	1	3	−3	−1	0	2	−1
−4	−1	2	−1	2	−3	1	−2
−1	−1	−1	−2	−1	−1	0	−1



16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

a b

FIGURE 8.30 (a) A threshold coding quantization curve [see Eq. (8.2-29)]. (b) A typical normalization matrix.

8.2.8 JPEG压缩

−26	−3	−6	2	2	0	0	0
1	−2	−4	0	0	0	0	0
−3	1	5	−1	−1	0	0	0
−4	1	2	−1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

C	AC Coefficients
1	-1,1
2	-3,-2,2,3
3	–7...–4,4...7
4	–15...–8,8...15
5	–31...–16,16...31
6	–63...–32,32...63
7	–127...–64,64...127
8	–255...–128,128...255
9	–511...–256,256...511
10	–1023...–512,512...1023

[−26 −3 1 −3 −2 −6 2 −4 1 −4 1 1 5 0 2 0 0 −1 2 0 0 0 0 0 −1 −1 EOB]

1010110 0100 001 0100 0101 100001 0110 100011 001 100011 001
001 100101 11100110 110110 0110 11110100 000 1010

8.2.8 JPEG压缩

−26	−3	−6	2	2	0	0	0
1	−2	−4	0	0	0	0	0
−3	1	5	−1	−1	0	0	0
−4	1	2	−1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

反量化

−416	−33	−60	32	48	0	0	0
12	−24	−56	0	0	0	0	0
−42	13	80	−24	−40	0	0	0
−56	17	44	−29	0	0	0	0
18	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

IDCT

8.2.8 JPEG压缩

−70	−64	−61	−64	−69	−66	−58	−50
−72	−73	−61	−39	−30	−40	−54	−59
−68	−78	−58	−9	13	−12	−48	−64
−59	−77	−57	0	22	−13	−51	−60
−54	−75	−64	−23	−13	−44	−63	−56
−52	−71	−72	−54	−54	−71	−71	−54
−45	−59	−70	−68	−67	−67	−61	−50
−35	−47	−61	−66	−60	−48	−44	−44

+128

58	64	67	64	59	62	70	78
56	55	67	89	98	88	74	69
60	50	70	119	141	116	80	64
69	51	71	128	149	115	77	68
74	53	64	105	115	84	65	72
76	57	56	74	75	57	57	74
83	69	59	60	61	61	67	78
93	81	67	62	69	80	84	84

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	144	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	63	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

8.2.8 JPEG压缩

压缩比
25:1

压缩比
52:1

a b c
d e f

FIGURE 8.32 Two JPEG approximations of Fig. 8.9(a). Each row contains a result after compression and reconstruction, the scaled difference between the result and the original image, and a zoomed portion of the reconstructed image.

8.3 数字图像水印

数字水印（利用图像信息的冗余）的作用：

1）版权认证；

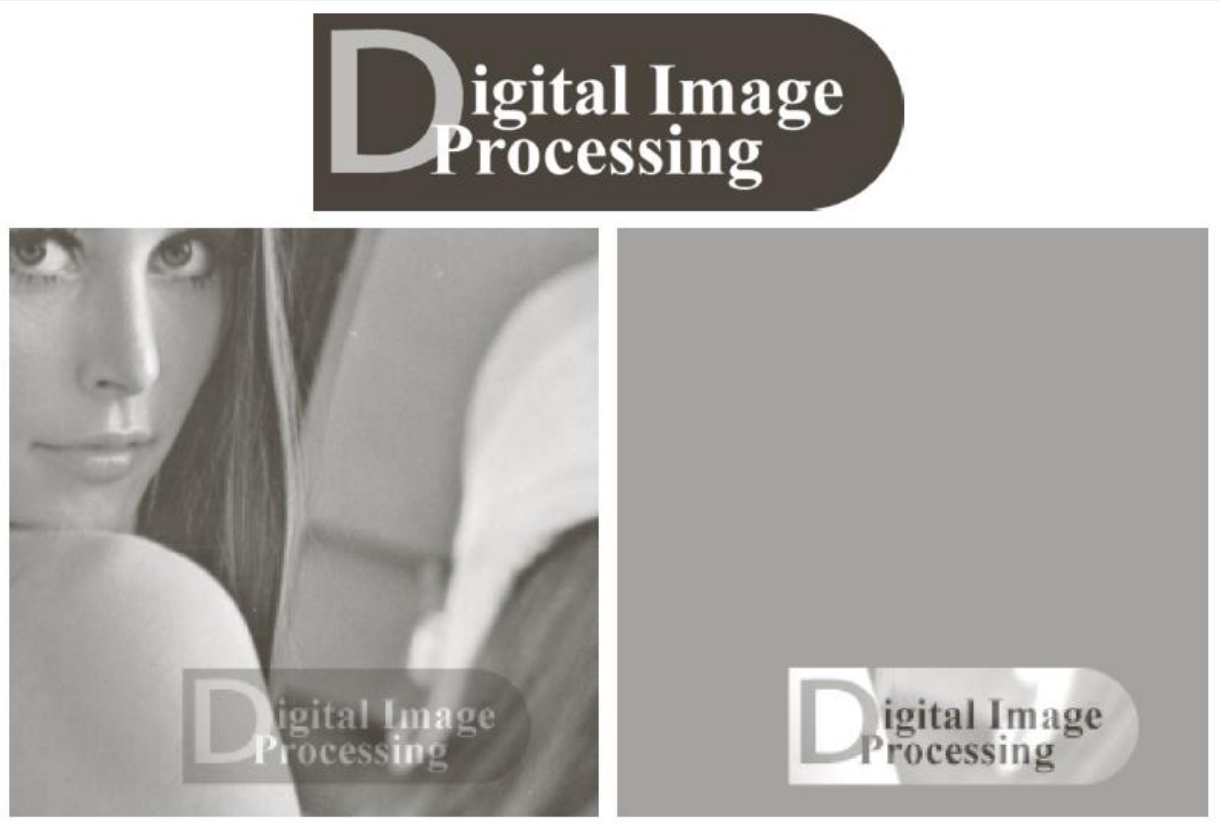
2）用户识别或指纹；

3）真实性鉴定（原始图片、完整图片的认定）；

4）自动监视；

5）拷贝保护。

8.3 数字图像水印



a
b c

FIGURE 8.50
A simple visible watermark: (a) watermark; (b) the watermarked image; and (c) the difference between the watermarked image and the original (non-watermarked) image.

$$f_w = (1-\alpha)f + \alpha w$$

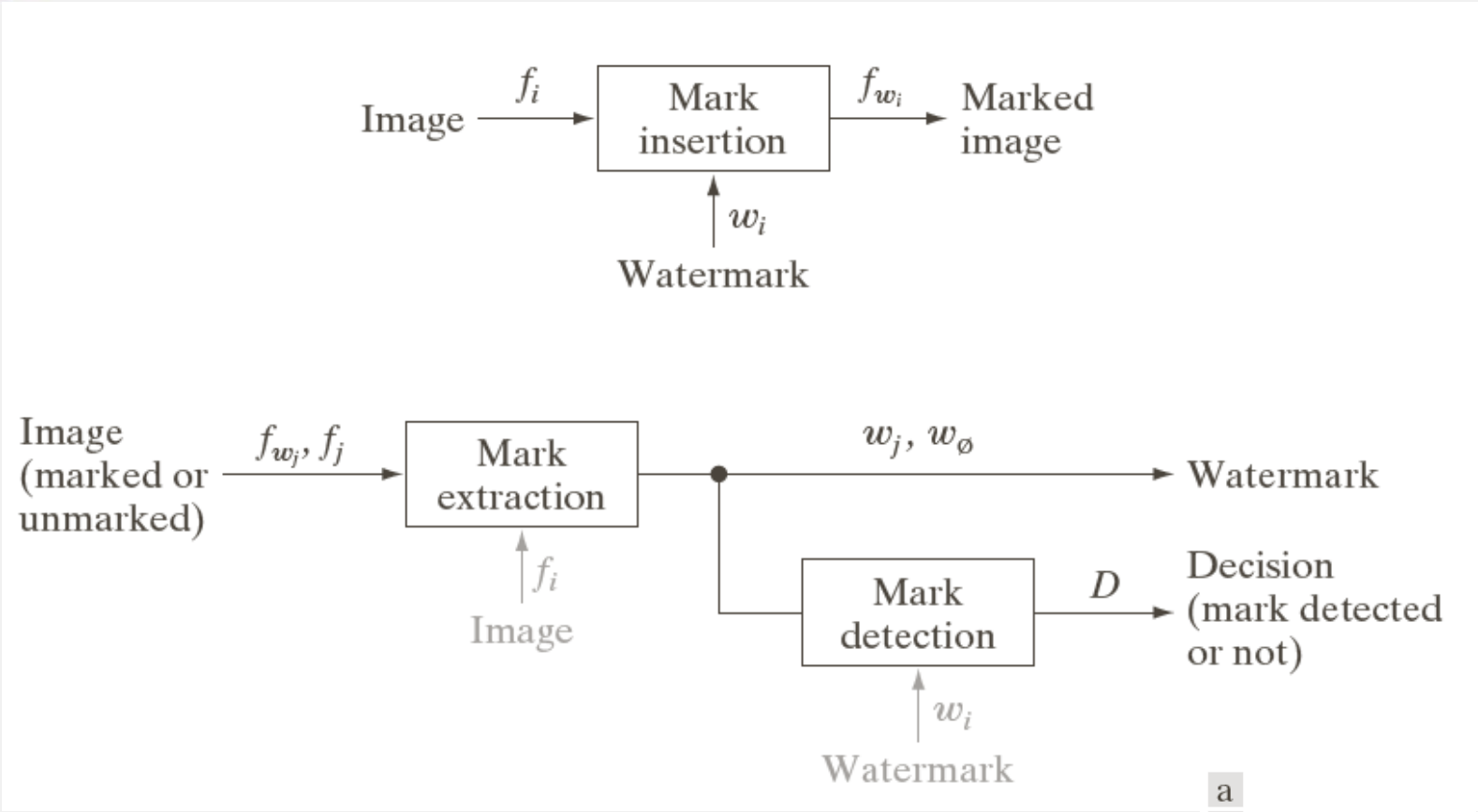
8.3 数字图像水印

a b
c d

FIGURE 8.51 A simple invisible watermark: (a) watermarked image; (b) the extracted watermark; (c) the watermarked image after high quality JPEG compression and decompression; and (d) the extracted watermark from (c).

$$f_w = 4\left(\frac{f}{4}\right) + \frac{w}{64}$$

8.3 数字图像水印



a
b

FIGURE 8.52 A typical image watermarking system: (a) encoder; (b) decoder.

8.3 数字图像水印

例8.30 基于DCT的不可见水印(Cox et al. [1997])

1. 计算载体图像的二维DCT变换。
2. 按幅值定位它的K个最大AC系数$c_1, c_2, \ldots, c_K$。
3. 生成K元素的伪随机序列$w_1, w_2, \ldots, w_K$(来源于均值为$\mu = 0$，方差为$\delta^2 = 1$的高斯序列)。
4. 嵌入方式为$c_i' = c_i(1 + \alpha w_i), 1 \le i \le K, \alpha > 0$。
5. 反DCT变换得到水印图片。

8.3 数字图像水印

水印提取过程：

1. 计算水印图像的二维DCT变换。
2. 定位K个用于水印嵌入的AC系数$\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_K$(如果水印图片未受攻击，有$\hat{c}_i = c'_i$)。
3. 使用下式计算水印$\hat{w}_1 = \frac{\hat{c}_i - c_i}{\alpha c_i}\ (1 \le i \le K)$。
4. 度量相似度$\gamma = \frac{\sum_{i=1}^{K}(\hat{w}_i - \overline{\hat{w}})(w_i - \overline{w})}{\sqrt{\sum_{i=1}^{K}(\hat{w}_i - \overline{\hat{w}})^2 \cdot \sum_{i=1}^{K}(w_i - \overline{w})^2}}, 1 \le i \le K$,其中$\overline{\hat{w}}$，$\overline{w}$为水印均值。
5. 将度量值与一个预定义阈值T进行比较

$D = \begin{cases} 1, \gamma \ge T \\ 0, \text{其它} \end{cases}$，$D$=1表示含有水印，反之则没有。

8.3 数字图像水印

a b
c d

FIGURE 8.53 (a) and (c) Two watermarked versions of Fig. 8.9(a); (b) and (d) the differences (scaled in intensity) between the watermarked versions and the unmarked image. These two images show the intensity contribution (although scaled dramatically) of the pseudo-random watermarks on the original image.

8.3 数字图像水印

a b c
d e f

FIGURE 8.54 Attacks on the watermarked image in Fig. 8.53(a): (a) lossy JPEG compression and decompression with an rms error of 7 intensity levels; (b) lossy JPEG compression and decompression with an rms error of 10 intensity levels (note the blocking artifact); (c) smoothing by spatial filtering; (d) the addition of Gaussian noise; (e) histogram equalization; and (f) rotation. Each image is a modified version of the watermarked image in Fig. 8.53(a). After modification, they retain their watermarks to varying degrees, as indicated by the correlation coefficients below each image.

作业：8.9，8.34（给出具体的Matlab实现及原始图片、加水印后的图片、提取的水印等）