

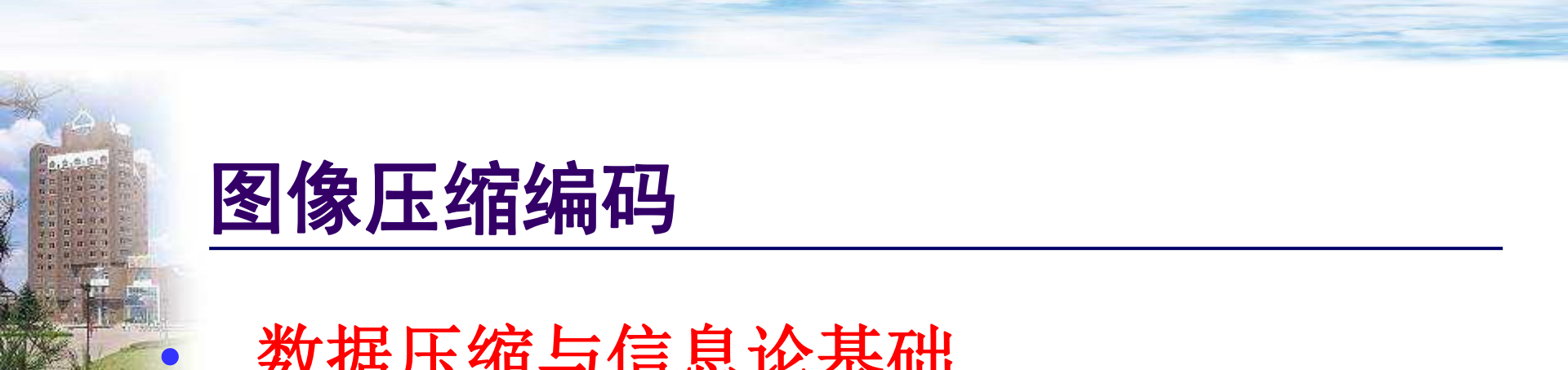
信号与图像处理基础

Image Compression

中国科学技术大学 自动化系

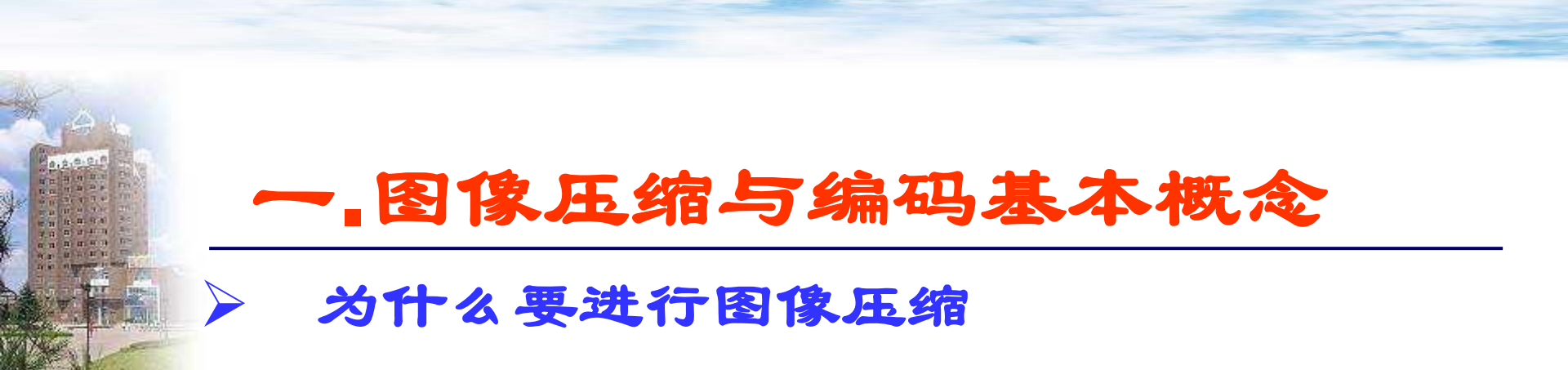
曹 洋





图像压缩编码

- 数据压缩与信息论基础
 - 图像压缩与编码基本概念
 - 信息论基础
- 图像压缩编码
 - 无损压缩
 - 有损压缩
- 图像压缩编码主要国际标准
 - 静止图像压缩编码标准-JPEG



一. 图像压缩与编码基本概念

- 为什么要进行图像压缩
- 图像数据压缩的可能性
- 数据冗余
- 图像压缩的目的
- 图像数据压缩技术的重要指标
- 图像数据压缩的应用领域
- 图像编码中的保真度准则
- 信息论基础
- 图像压缩模型



1. 为什么要进行图像压缩？

数字图像通常要求很大的比特数，这给图像的传输和存储带来相当大的困难。要占用很多的资源，花很高的费用。

如一幅512*512的灰度图象的比特数为

$$512*512*8=256k$$

再如一部90分钟的彩色电影，每秒放映24帧。把它数字化，每帧512*512象素，每象素的R、G、B三分量分别占8 bit，总比特数为

$$90*60*24*3*512*512*8\text{bit}=97,200M。$$



2. 图像数据压缩的可能性

- 一般原始图像中存在很大的冗余度。
- 用户通常允许图像失真。
- 当信道的分辨率不及原始图像的分辨率时，降低输入的原始图像的分辨率对输出图像分辨率影响不大。
- 用户对原始图像的信号不全都感兴趣，可用特征提取和图像识别的方法，丢掉大量无用的信息。提取有用的信息，使必须传输和存储的图像数据大大减少。



3. 数据冗余

1) 数据冗余的基本概念

描述信源的数据是信息量（信源熵）和信息冗余量之和。

设： n_1 和 n_2 是在两个表达相同信息的数据集中，所携带的单位信息量。

- 压缩率：——描述压缩算法性能

$$C_R = n_1 / n_2$$

其中， n_1 是压缩前的数据量， n_2 是压缩后的数据量

- 相对数据冗余：

$$R_D = 1 - 1/C_R$$

例： $C_R=20$ ； $R_D = 19/20$



3. 数据冗余

2) 常见的数据冗余

在数字图像压缩中，常有3种基本的数据冗余：编码冗余、像素间的冗余以及心理视觉冗余

- A. 编码冗余：

为表达图像数据需要用一系列符号，用这些符号根据一定的规则来表达图像就是对**图像编码**。

对每个信息或事件所赋的符号序列称为**码字**，而每个码字里的符号个数称为**码字的长度**。



3. 数据冗余

设定义在 $[0,1]$ 区间的离散随机变量 s_k 代表图像的灰度值，每个 s_k 以概率 $p_s(s_k)$ 出现

$$P_s(s_k)=n_k/n \quad k=0,1,2,\dots,L-1$$

其中 L 为灰度级数， n_k 是第 k 个灰度级出现的次数， n 是图像中像素总个数。设用来表示 s_k 的每个数值的比特数是 $l(s_k)$ ，那么为表示每个像素所需的平均比特数就是

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

编码所用的符号构成的集合称为**码本**。



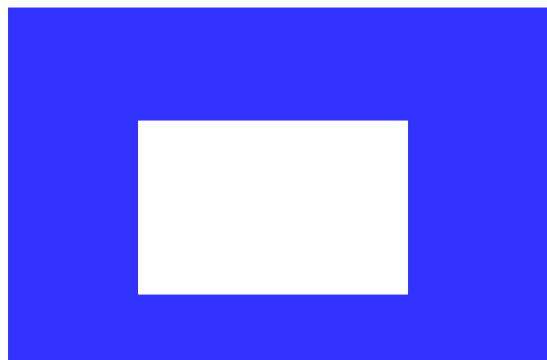
3. 数据冗余

等长码：对于一个消息集合中的不同消息，用相同长度的不同码字表示，**编解码简单，编码效率不高。**

变长码：与等长码相对应，对于一个消息集合中的不同消息，也可以用不同长度的码字表示，**编码效率高，编码解码复杂。**

3. 数据冗余

如果一个图像的灰度级编码，使用了多于实际需要的编码符号，就称该图像包含了编码冗余。



例：如果用8位表示该图像的像素，我们就说该图像存在着编码冗余，因为该图像的像素只有两个灰度，用一位即可表示。



3. 数据冗余

- B. 像素冗余:

由于任何给定的像素值，原理上都可以通过它的邻居预测到，单个像素携带的信息相对是小的。

对于一个图像，很多单个像素对视觉的贡献是冗余的。这是建立在对邻居值预测的基础上。

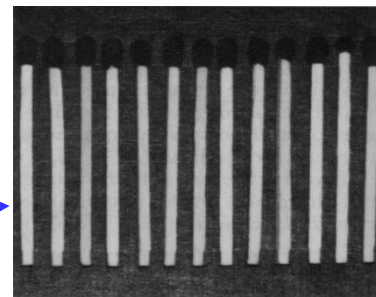
原始图像越有规则，各像素之间的相关性越强，它可能压缩的数据就越多。

例：原图像数据：234 223 231 238 235

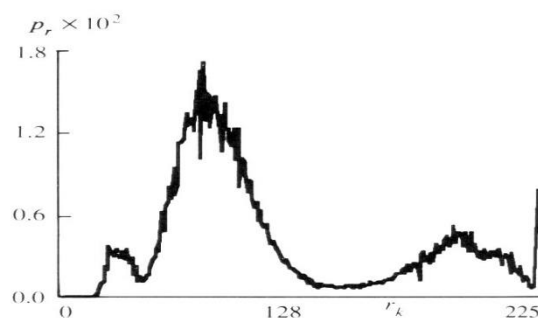
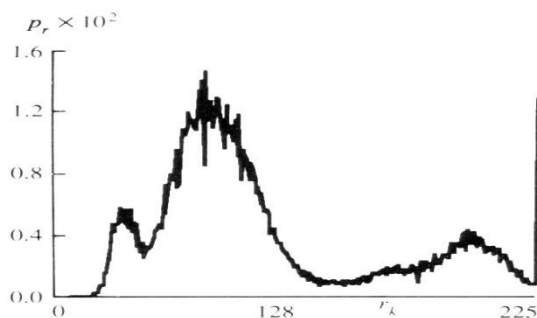
压缩后数据：234 11 -8 -7 3

3. 数据冗余

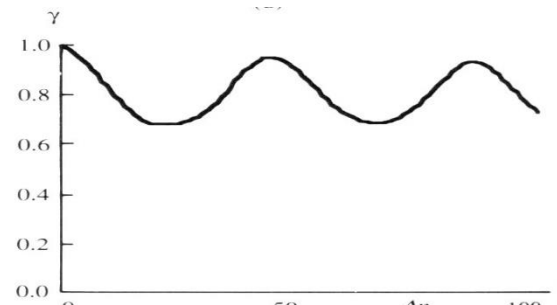
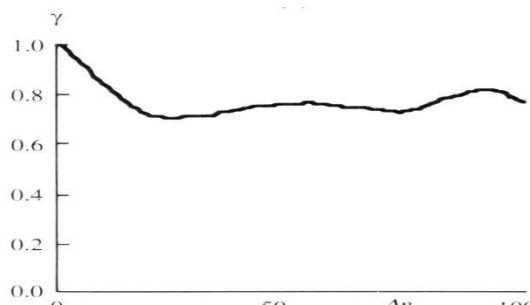
相同的目标



相同的直方图



像素间的相
关性不同

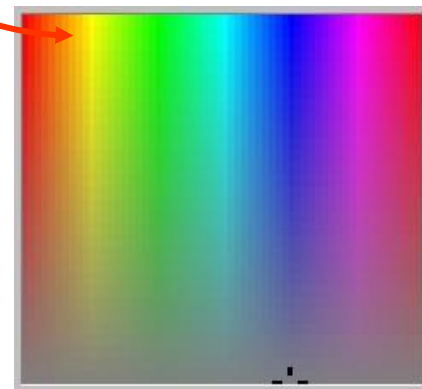


3. 数据冗余

类似还有：

图像彩色光谱空间的冗余；

视频图像信号在时间上的冗余；



3. 数据冗余

- 视觉心理冗余：

一些信息在一般视觉处理中比其它信息的相对重要程度要小，这种信息就被称为视觉心理冗余。





4. 图像压缩的目的

图像数据压缩的目的是在满足一定图像质量条件下，用尽可能少的比特数来表示原始图像，以提高图像传输的效率和减少图像存储的容量。在信息论中称为信源编码。

图像从结构上大体上可分为两大类，一类是具有一定图形特征的结构，另一类是具有一定概率统计特性的结构。

基于不同的图像结构特性，应采用不同的压缩编码方法。



5. 图像数据压缩技术的重要指标

- (1) **压缩比**: 图像压缩前后所需的信息存储量之比, 压缩比越大越好。
- (2) **压缩算法**: 利用不同的编码方式, 实现对图像的数据压缩。
- (3) **失真性**: 压缩前后图像存在的误差大小。



5. 图像数据压缩技术的重要指标

全面评价一种编码方法的优劣，除了看它的**编码效率**、**实时性**和**失真度**以外，还要看它的**设备复杂程度**，是否**经济与实用**。

常采用混合编码的方案，以求在性能和经济上取得折衷。

随着计算方法的发展，使许多高效而又比较复杂的编码方法在工程上有实现的可能。



6. 图像编码中的保真度准则

图像信号在编码和传输过程中会产生误差，尤其是在有损压缩编码中，产生的误差应在允许的范围之内。在这种情况下，保真度准则可以用来衡量编码方法或系统质量的优劣。通常，这种衡量的尺度可分为**客观保真度准则**和**主观保真度准则**。



6. 图像编码中的保真度准则

(1) 客观保真度准则

通常使用的客观保真度准则有输入图像和输出图像的**均方根误差**；输入图像和输出图像的**均方根信噪比**两种。

均方根误差：设输入图像是由 $N \times N$ 个像素组成，令其为 $f(x, y)$ ，其中 $x, y=0, 1, 2, \dots, N-1$ 。这样一幅图像经过压缩编码处理后，送至受信端，再经译码处理，重建原来图像，这里令重建图像为 $g(x, y)$ 。它同样包含 $N \times N$ 个像素，并且 $x, y=0, 1, 2, \dots, N-1$ 。



6. 图像编码中的保真度准则

在 $0, 1, 2, \dots, N-1$ 范围内 x, y 的任意值，输入像素和对应的输出图像之间的误差可用下式表示：

$$e(x, y) = g(x, y) - f(x, y)$$

而包含 $N \times N$ 像素的图像之均方误差为：

$$\overline{e^2} = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} e^2(x, y) = \frac{1}{N^2} \sum_{N=0}^{N-1} \sum_{N=0}^{N-1} [g(x, y) - f(x, y)]^2$$

由式可得到均方根误差为

$$e_{rms} = [\overline{e^2}]^{1/2}$$



6. 图像编码中的保真度准则

如果把输入、输出图像间的误差看作是噪声，那么，重建图像 $g(x, y)$ 可由下式表示：

$$g(x, y) = f(x, y) + e(x, y)$$

在这种情况下，另一个客观保真度准则——重建图像的均方信噪比如下式表示：

$$\left(\frac{S}{N}\right)_{ms} = \frac{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} g^2(x, y)}{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} e^2(x, y)} = \frac{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} g^2(x, y)}{\sum_{x=0}^{N-1} \sum_{y=0}^{N-1} [g(x, y) - f(x, y)]^2}$$



6. 图像编码中的保真度准则

(2) 主观保真度准则

图像处理的结果,大多是给人观看,由研究人员来解释的,因此,图像质量的好坏,既与图像本身的客观质量有关,也与视觉系统的特性有关。

有时候,客观保真度完全一样的两幅图像可能会有完全不同的视觉质量,所以又规定了主观保真度准则,这种方法是把图像显示给观察者,然后把评价结果加以平均,以此来评价一幅图像的主观质量。



6. 图像编码中的保真度准则

表6.1 电视图像质量评价尺度

评分	评价	说明
1	优秀的	优秀的具有极高质量的图像
2	好的	是可供观赏的高质量的图像，干扰并不令人讨厌
3	可通过的	图像质量可以接受，干扰不讨厌
4	边缘的	图像质量较低，希望能加以改善，干扰有些讨厌
5	劣等的	图像质量很差，尚能观看，干扰显著地令人讨厌
6	不能用	图像质量非常之差，无法观看



7. 信息论

(一)、信源空间概述

- 1、信息：事物运动状态或存在方式的不确定性的描述；
- 2、信源空间：随机符号及其出现概率的空间；
- 3、信源的分类：
 - (1) 连续信源—离散信源—混合信源；
 - (2) 无记忆信源—有记忆信源（相关信源）—有限长度记忆信源（Markov信源）



7. 信息论

(二)、信息的度量

1、信息公理

(1) 信息由不确定性程度进行度量;

确定事件的信息量为零。

(2) 不确定性程度越高信息量越大;

(3) 相互独立性与信息量可加性;

独立事件的联合信息等于两个独立事件的信息总和。

满足上述公理的函数为:

$$I(a) = -\log P(a)$$



7. 信息论

2、离散无记忆信源（**DNMS**）的信息量度量：

(1) 信源符号 a_i 的自信息量定义为：

$$I(a_i) = -\log P(a_i)$$

(a)非负性；

(b)信息量的单位：

底为2时——单位为：比特（bit）

底为e时——单位为：奈特（Nat）

底为10时——单位为：哈特



7. 信息论

(2)、信源平均自信息量（信息熵）

离散无记忆信源**A**的平均自信息量（信息熵）定义为：

$$\begin{aligned} H(A) &= \sum_{i=1}^m P(a_i) I(a_i) \\ &= - \sum_{i=1}^m P(a_i) \log P(a_i) \end{aligned}$$



7. 信息论

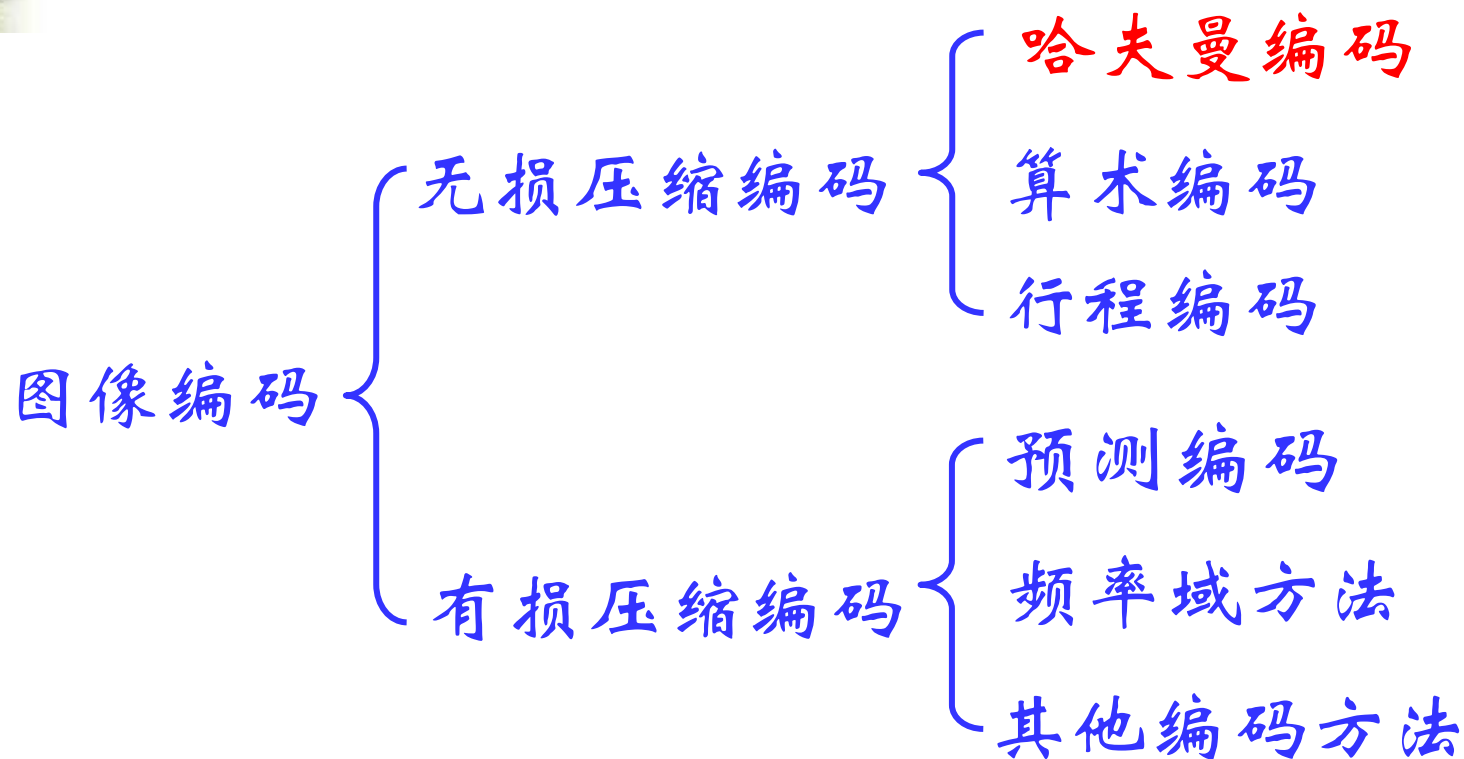
例：设8个随机变量具有同等概率为 $1/8$ ，计算信息熵 H 。

$$\text{解：} H = 8 * [-1/8 * (\log_2(1/8))] = 8 * [-1/8 * (-3)] = 3$$

图像熵指该图像的平均信息量，即表示图像中各个灰度级比特数的统计平均值，等概率事件的熵最大。



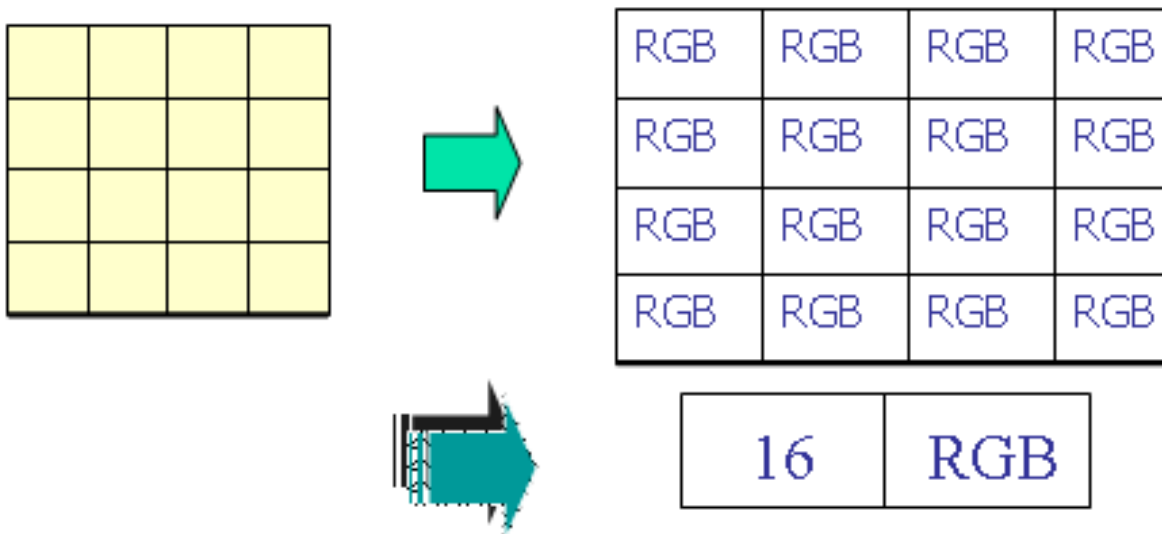
二. 常用的图像压缩编码方法



1.无损压缩编码

✂ 无损压缩算法中删除的仅仅是图像数据中冗余的信息，因此在解压缩时能精确恢复原图像，无损压缩的压缩比很少有能超过3:1的，常用于要求高的场合。

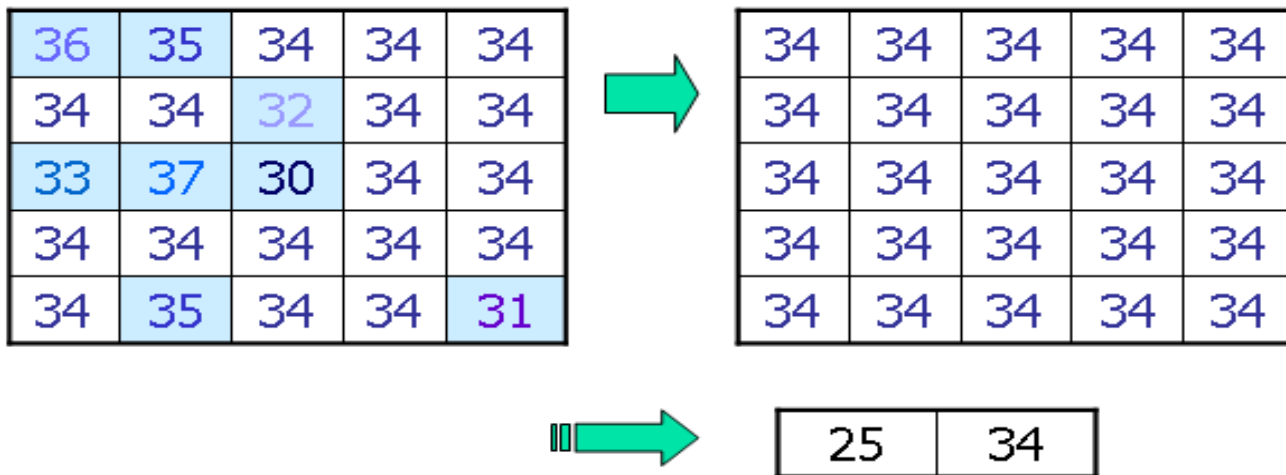
■ 图像冗余无损压缩的原理



2. 有损压缩编码

※有损压缩是通过牺牲图像的准确率以实现较大的压缩率，如果容许解压图像有一定的误差，则压缩率可显著提高。有损压缩在压缩比大于30:1时仍然可重构图像，而如果压缩比为10:1到20:1，则重构的图像与原图几乎没有差别

■ 图像冗余有损压缩的原理





3. 哈夫曼编码

等长码：对于一个消息集合中的不同消息，用相同长度的不同码字表示，**编解码简单，编码效率不高。**

变长码：与等长码相对应，对于一个消息集合中的不同消息，也可以用不同长度的码字表示，**编码效率高，编码解码复杂。**

哈夫曼编码是一种利用信息符号概率分布特性的**变字长**的编码方法。对于出现概率大的信息符号编以短字长的码，对于出现概率小的信息符号编以长字长的码。



3. 哈夫曼编码

- I. 将信源符号按出现概率从大到小排成一行，然后把最末两个符号的概率相加，合成一个概率。
- II. 把这个符号的概率与其余符号的概率按从大到小排列，然后再把最末两个符号的概率加起来，合成一个概率。
- III. 重复上述做法，直到最后剩下两个概率为止。
- IV. 从最后一步剩下的两个概率开始逐步向前进行编码。每步只需对两个分支各赋予一个二进制码，如对概率大的赋予码0，对概率小的赋予码1。



3. 哈夫曼编码

输入 输入概率

S_1 0.4

S_2 0.3

S_3 0.1

S_4 0.1

S_5 0.06

S_6 0.04

3. 哈夫曼编码

输入 输入概率 第一步

S_1 0.4 0.4

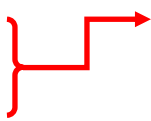
S_2 0.3 0.3

S_3 0.1 0.1

S_4 0.1 0.1

S_5 0.06 0.1

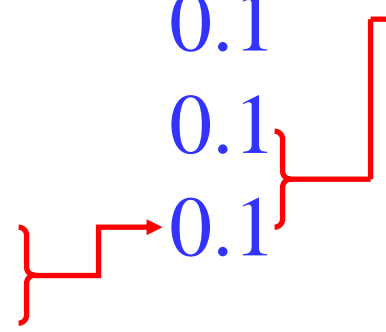
S_6 0.04



3. 哈夫曼编码

输入 输入概率 第一步 第二步

S_1	0.4	0.4	0.4
S_2	0.3	0.3	0.3
S_3	0.1	0.1	0.2
S_4	0.1	0.1	0.1
S_5	0.06	0.1	
S_6	0.04		



3. 哈夫曼编码

输入 输入概率 第一步 第二步 第三步

S_1	0.4	0.4	0.4	0.4
S_2	0.3	0.3	0.3	0.3
S_3	0.1	0.1	0.2	0.3
S_4	0.1	0.1	0.1	
S_5	0.06	0.1		
S_6	0.04			

3. 哈夫曼编码

输入 输入概率 第一步 第二步 第三步 第四步

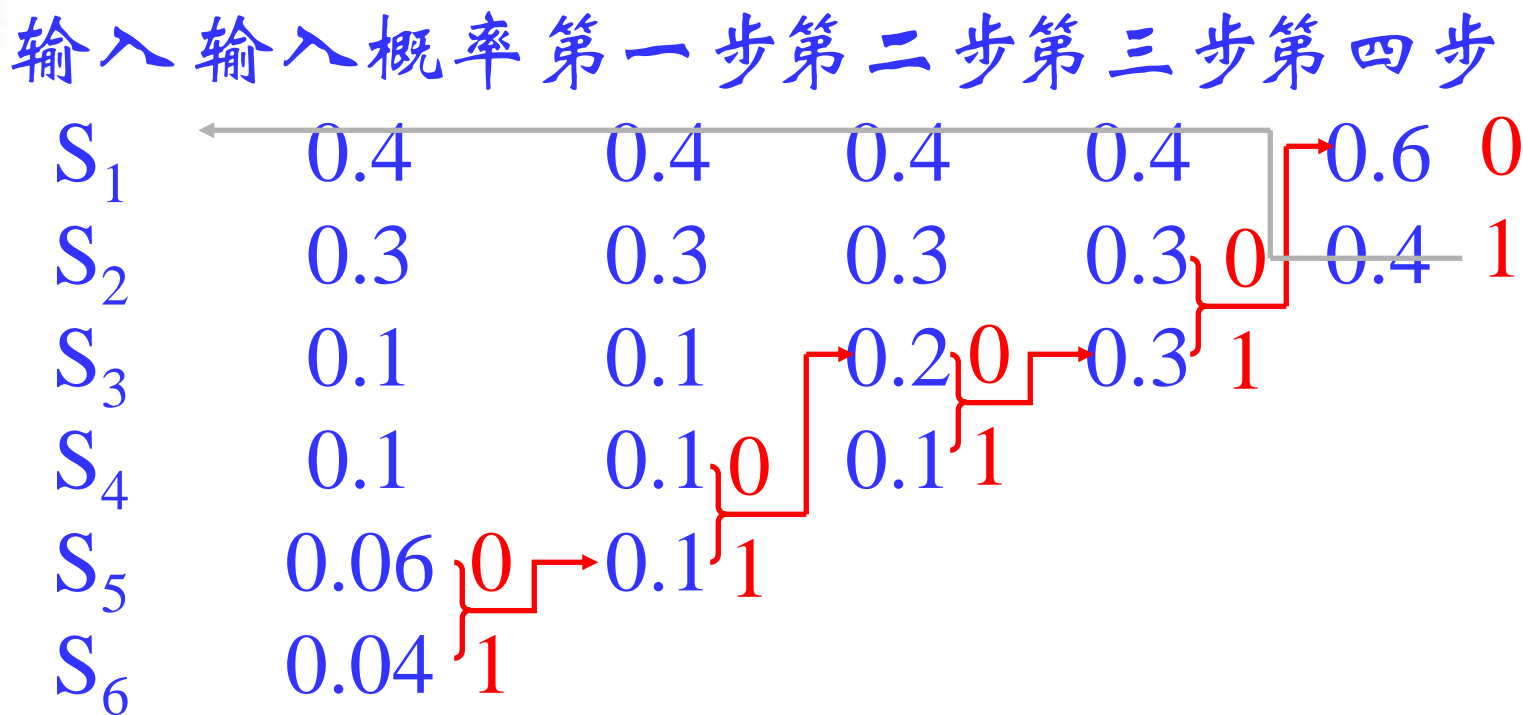
S_1	0.4	0.4	0.4	0.4	0.6
S_2	0.3	0.3	0.3	0.3	0.4
S_3	0.1	0.1	0.2	0.3	
S_4	0.1	0.1	0.1		
S_5	0.06	0.1			
S_6	0.04				

Huffman 编码

输入 输入概率 第一步 第二步 第三步 第四步

S_1	0.4	0.4	0.4	0.4	0.6	0
S_2	0.3	0.3	0.3	0.3	0.4	1
S_3	0.1	0.1	0.2	0.3		
S_4	0.1	0.1	0.1			
S_5	0.06	0.1				
S_6	0.04					

3. 哈夫曼编码



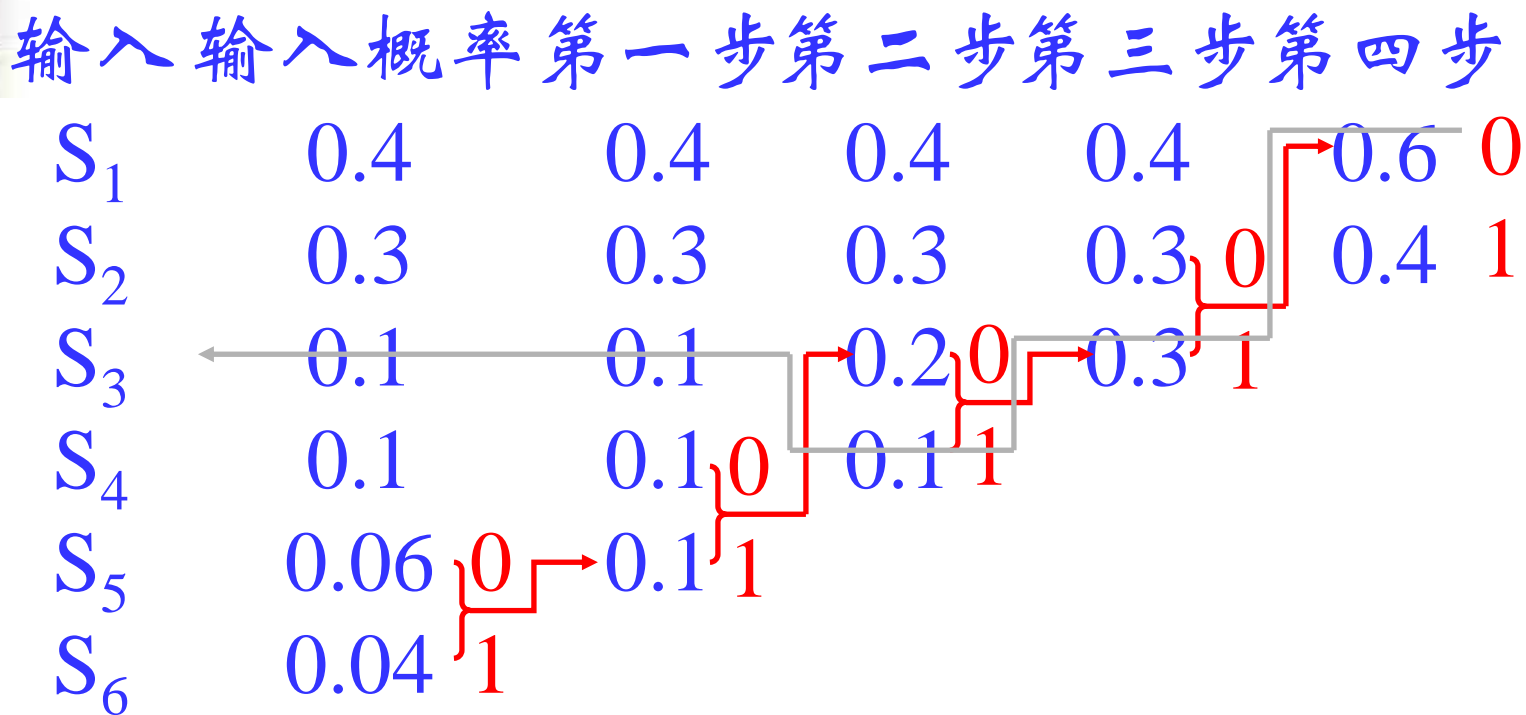
$S_1=1$

3. 哈夫曼编码

输入	输入概率	第一步	第二步	第三步	第四步
S_1	0.4	0.4	0.4	0.4	0.6 0
S_2	0.3	0.3	0.3	0.3	0.4 1
S_3	0.1	0.1	0.2	0.3	
S_4	0.1	0.1	0.1		
S_5	0.06	0.1			
S_6	0.04				

$S_2=00$

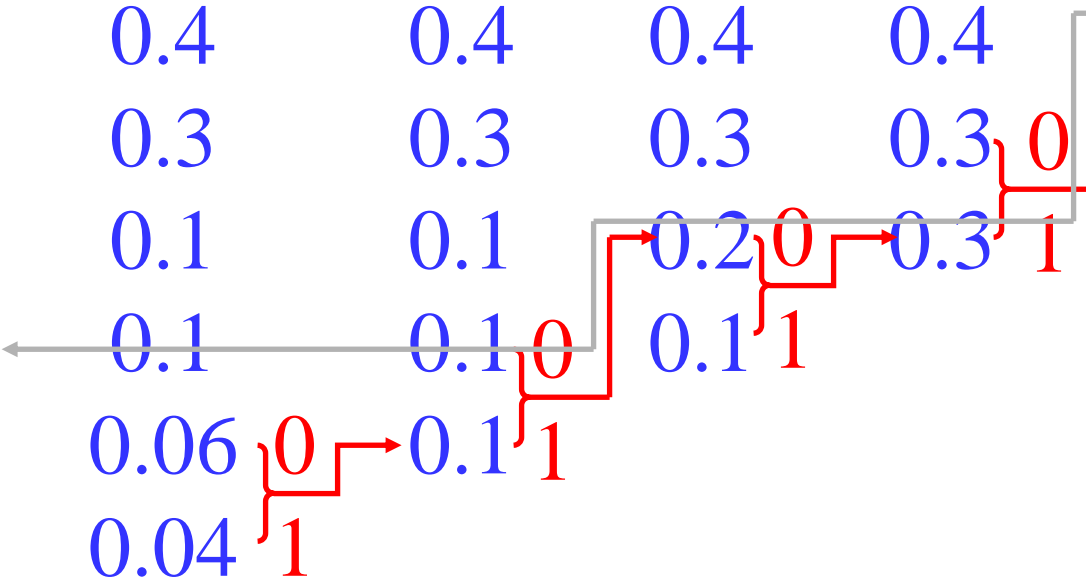
3. 哈夫曼编码



$S_3=011$

3. 哈夫曼编码

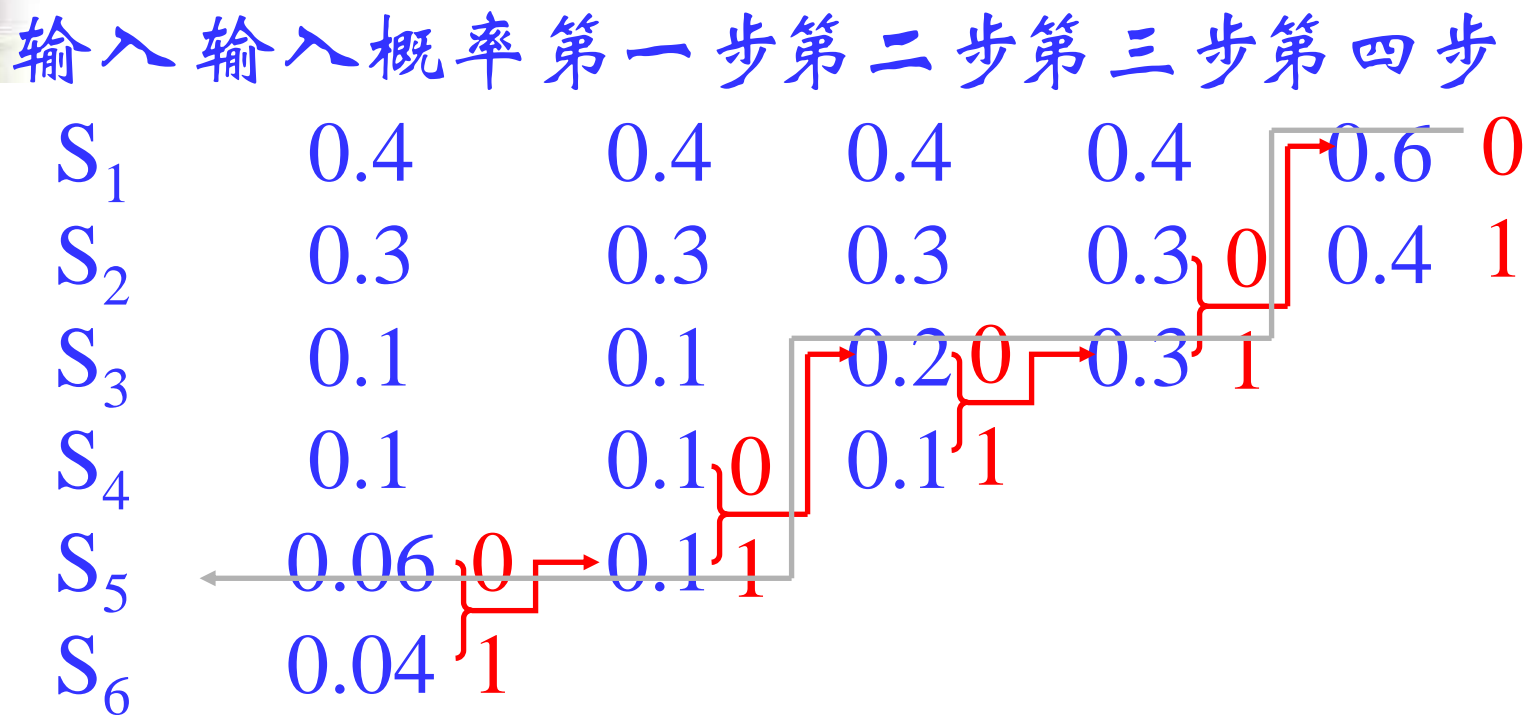
输入	输入概率	第一步	第二步	第三步	第四步
S_1	0.4	0.4	0.4	0.4	0.6 0
S_2	0.3	0.3	0.3	0.3	0.4 1
S_3	0.1	0.1	0.2	0.3	
S_4	0.1	0.1	0.1		
S_5	0.06	0.1			
S_6	0.04				



The diagram illustrates the Huffman tree construction. Red lines and brackets show the merging of nodes at each step. A grey arrow points from S_4 to the left, indicating its final code path.

$S_4=0100$

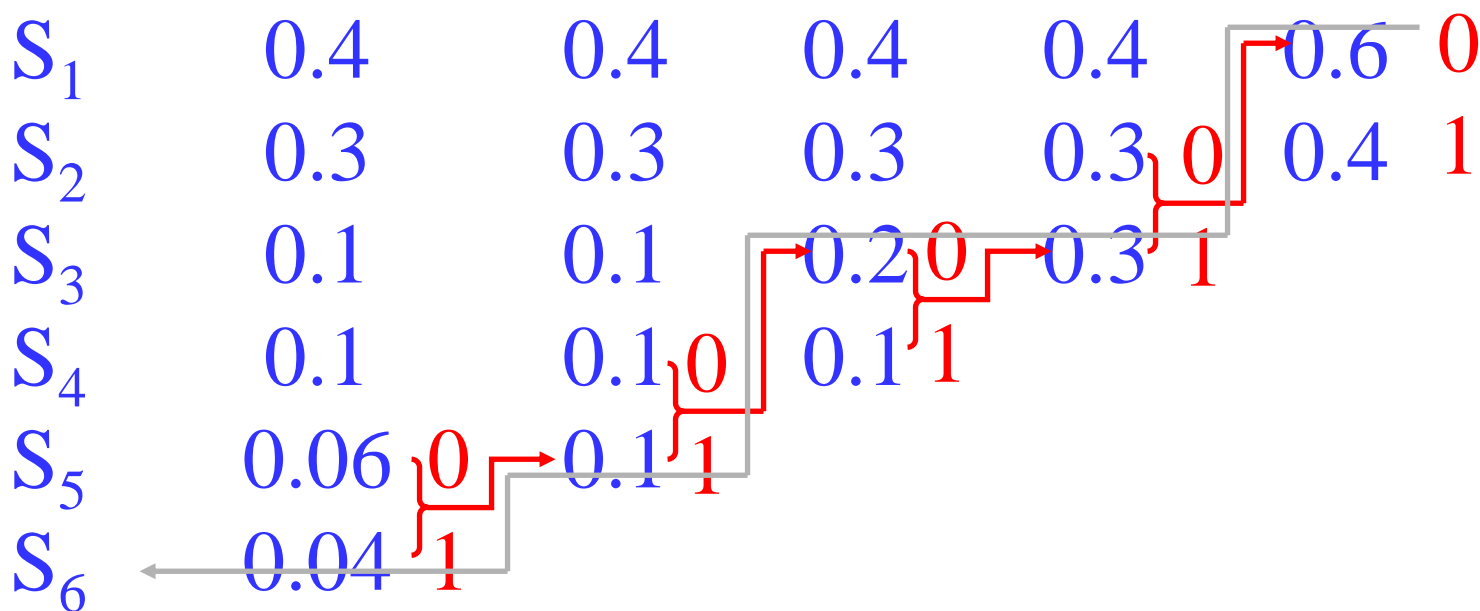
3. 哈夫曼编码



$S_5=01010$

3. 哈夫曼编码

输入 输入概率 第一步 第二步 第三步 第四步



$S_6=01011$



3. 哈夫曼编码

- 哈夫曼编码效率

信源熵为:

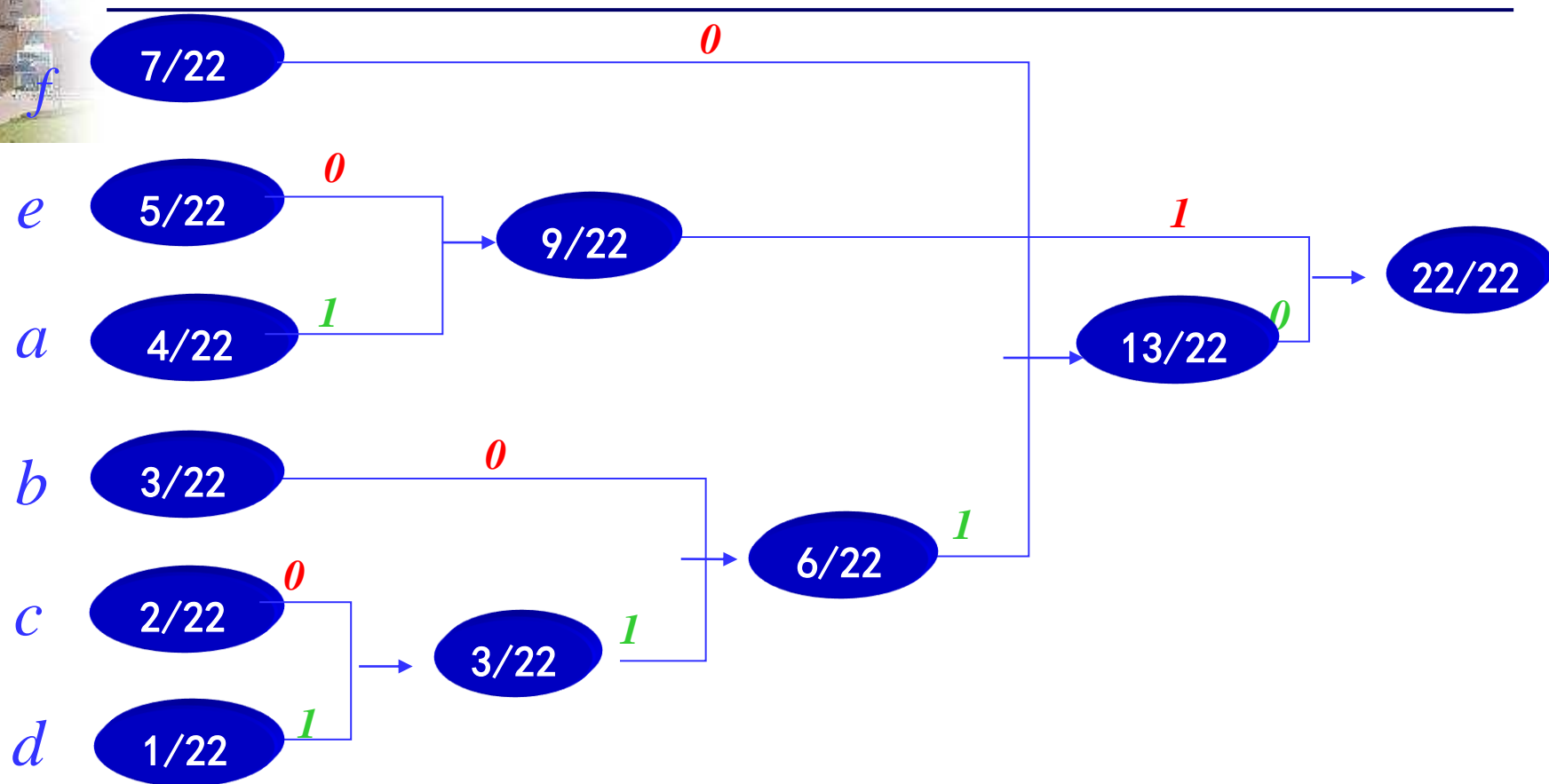
$$H = -\sum P_i \log_2 P_i$$

$$= -(0.4 \log_2 0.4 + 0.3 \log_2 0.3$$

$$+ 2 * 0.1 \log_2 0.1 + 0.06 \log_2 0.06 + 0.04 \log_2 0.04)$$

$$= 2.14 \text{ 比特/符号}$$

编码举例



$f=00$ $e=10$ $a=11$ $b=010$ $c=0110$ $d=0111$



4. 算术编码

- 从理论上分析，采用哈夫曼编码可以获得最佳信源字符编码效果；
- 实际应用中，由于信源字符出现的概率并非满足2的负幂次方，因此往往无法达到理论上的编码效率和信息压缩比；



4. 算术编码

以信源字符序列{x, y}为例

- 设字符序列{x, y}对应的概率为{1/3, 2/3}, N_x 和 N_y 分别表示字符x和y的最佳码长, 则根据信息论有:

$$N_x = -\log_2\left(\frac{1}{3}\right) = 1.58$$

$$N_y = -\log_2\left(\frac{2}{3}\right) = 0.588$$



4. 算术编码

- 字符 x 、 y 的最佳码长分别为1.58bit和0.588bi;
- 这表明, 要获得最佳编码效果, 需要采用小数码字长度, 这是不可能实现的;
- 即采用哈夫曼方法对 $\{x, y\}$ 的码字分别为0和1, 也就是两个符号信息的编码长度都为1。对于出现概率大的字符 y 并未能赋予较短的码字;
- 实际编码效果往往不能达到理论效率;
- 为提高编码效率, Elias等人提出了算术编码算法。

4. 算术编码

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.

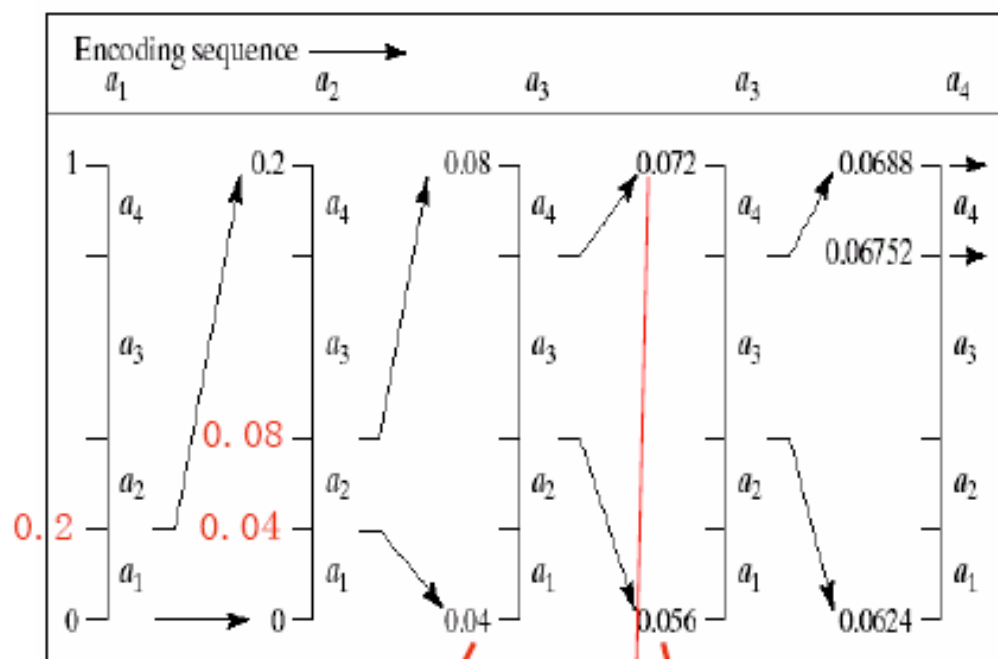


FIGURE 8.13

Arithmetic coding procedure.

$$0.04 = 0 + (0.2 - 0) / 5 \times 1$$

$$0.08 = 0 + (0.2 - 0) / 5 \times 2$$

$$0.056 = 0.04 + (0.08 - 0.04) / 5 \times 2$$

$$0.072 = 0.04 + (0.08 - 0.04) / 5 \times 4 = 0.04 + 0.032$$



5. 行程编码

RLE 编码——Run Length Encoding

—概念：

- 行程：具有相同灰度值的像素序列。

— 编码思想：去除像素冗余。

- 用行程的灰度和行程的长度代替行程本身。



5. 行程编码

0	1	1	0	0	1	1	1	0	0	0	0	1	1	1	1	1	1	0	1
0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	1	1	1

1的游程: (2,2) (6,3) (13,6) (20,1)
(4,6) (11,10)
(1,5) (11, 1) (17, 4)

1和0的游程长度: 0, 1, 2, 2, 3, 4, 6, 1, 1
0, 3, 6, 1, 10
1, 5, 5, 1, 5, 4



5. 行程编码

— 分析：

- 对于有大面积色块的图像，压缩效果很好
- 直观，经济，是一种无损压缩
- 对于纷杂的图像，压缩效果不好，最坏情况下，会加倍图像



6. 无损预测编码

■ 无损预测编码

1. 编码思想

- 1) 去除像素冗余。
- 2) 认为相邻像素的信息有冗余。当前像素值可以用以前的像素值来获得。
- 3) 当前像素值 f_n ，通过预测器得到一个预测值 \hat{f}_n ，对当前值和预测值求差，对残差编码，作为压缩数据流中的下一个元素。由于残差比原数据要小，因而编码要小，可用变长编码。大多数情况下， f_n 的预测是通过 m 个以前像素的线性组合来生成的。



6. 无损预测编码

■ 无损预测编码步骤

第一步：压缩头处理

第二步：对每一个符号 $f(x;y)$ 通过预测器求出预测值 $\hat{f}(x;y)$

第三步：求出预测误差

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

第四步：对误差 $e(x,y)$ 编码，作为压缩值。

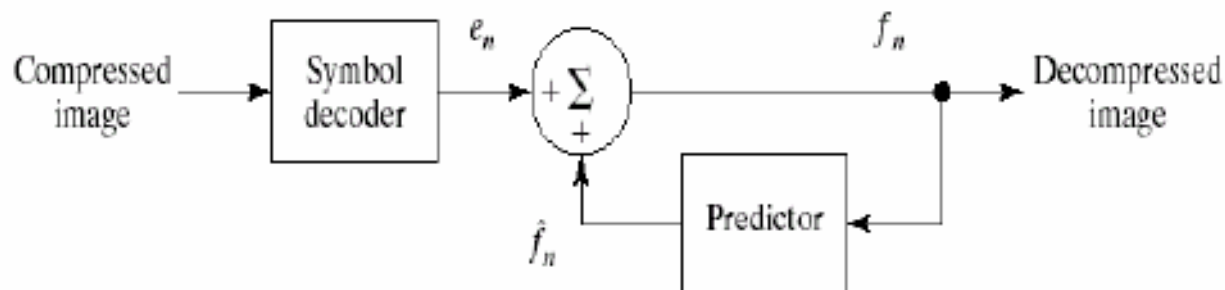
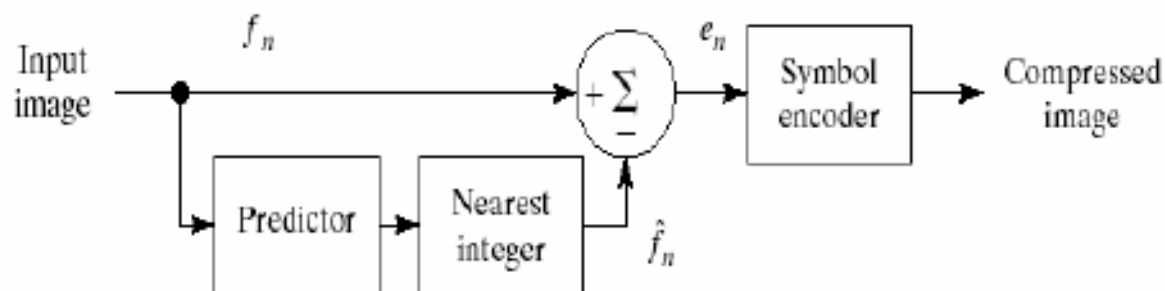
重复二、三、四步

6. 无损预测编码

■ 无损预测编码流程图示意图

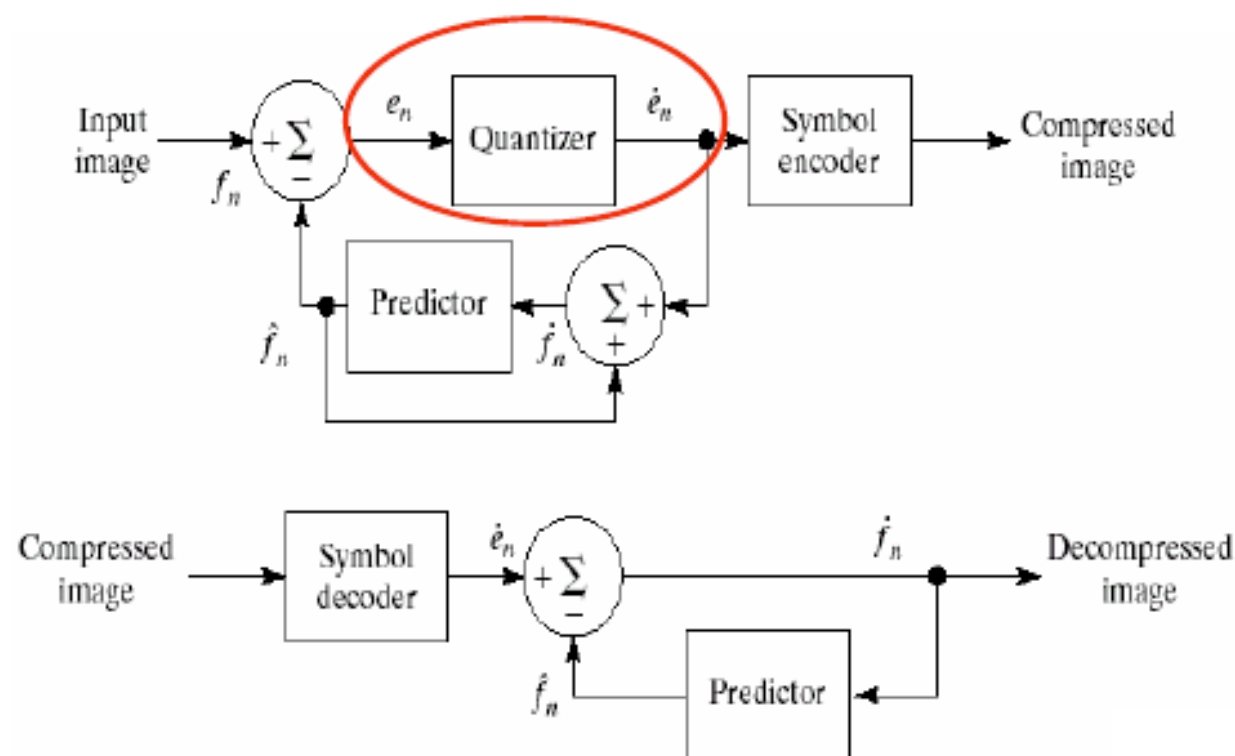
a
b

FIGURE 8.19 A lossless predictive coding model:
(a) encoder;
(b) decoder.



7. 有损预测编码

■ 有损预测编码流程示意图



a
b

FIGURE 8.21 A lossy predictive coding model: (a) encoder and (b) decoder.

$$\hat{f}_n = \hat{e}_n + \hat{f}_{n-1}$$

7. 有损预测编码

- Δ 调制

■ Δ 调制

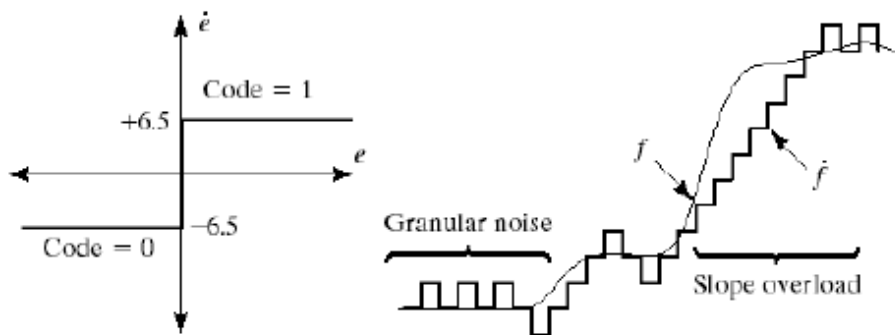


FIGURE 8.22 An example of delta modulation.

[illegible]

7. 有损预测编码

■ 熵调制

The residual values should be very small, so

- ▶ Use fewer bits for smaller values (entropy coding), or
- ▶ Use finer quantization (less loss) for smaller values

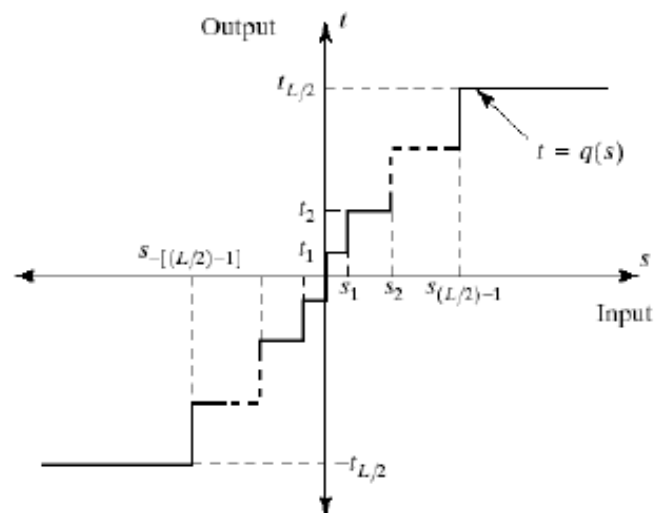


FIGURE 8.25 A typical quantization function.



三. 静止图像压缩编码标准 – JPEG

- 图像标准的制定:

ISO和CCITT (国际电报电话咨询委员会) 联合制定

- 标准的类型:

- 连续图像压缩标准:

静止帧黑白、彩色压缩: (1) 面向静止的单幅图像
- JPEG

连续帧黑白、彩色压缩: (2) 面向连续的视频影像
- MPEG



➤ JPEG标准简述

由ISO/IEC与CCITT联合发起的联合图像专家组，在过去十几年图像编码研究成果的基础上于20世纪90年代初制定了静止图像(包括8bit/像素的灰度图像与24bit/像素的彩色图像)的编码标准。

JPEG标准在较低的计算复杂度下，能提供较高的压缩比与保真度。在视觉效果不受到严重损失的前提下，算法可以达到15到20的压缩比。如果在图像质量上稍微牺牲一点的话，可以达到40:1或更高的压缩比。



➤ JPEG标准简述

JPEG定义了一个基本系统，一个符合JPEG标准的编解码器至少要满足基本系统的技术指标。JPEG基本系统其核心属于**分块变换编码**。JPEG编码时，对原始图像的每一个分量首先分割成互不重叠的 8×8 像素块，然后对每个像素块的编码过程可分为二维DCT变换。

为什么要进行分块变换编码？



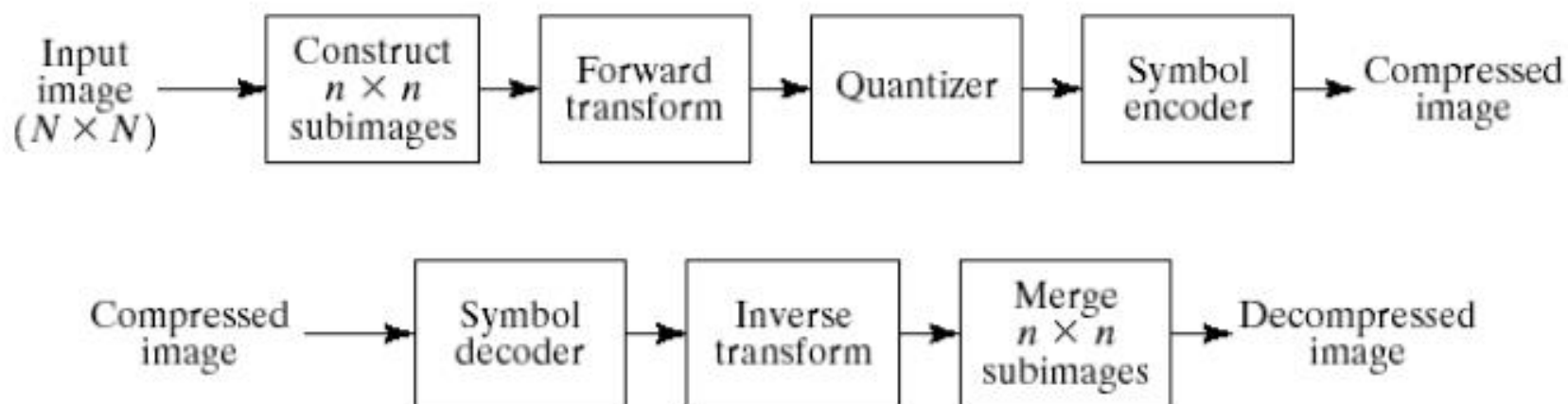
➤ JPEG标准简述

JPEG定义了一个基本系统，一个符合JPEG标准的编解码器至少要满足基本系统的技术指标。JPEG基本系统其核心属于**分块变换编码**。JPG编码时，对原始图像的每一个分量首先分割成互不重叠的 8×8 像素块，然后对每个像素块的编码过程可分为二维DCT变换。

分块变换编码的优势：

- (1) 使得图像信号的能量更加集中；
- (2) 利用了人类视觉心理冗余性
- (3) 分块变换使得图像的artifact不会扩展到整幅图像，保证了图像编码质量。

分块变换编码通用框架



a
b

FIGURE 8.28 A transform coding system: (a) encoder; (b) decoder.



变换编码

There are many other transforms besides the Fourier Transform, all with the same structure:

- Forward transform (general form):

$$T(u, v) = \sum_{x=0}^M \sum_{y=0}^N f(x, y) g(x, y, u, v)$$

- Inverse transform (general form):

$$f(x, y) = \sum_{u=0}^M \sum_{v=0}^N T(u, v) h(x, y, u, v)$$

Most of the time, $g(x, y, u, v) = h(x, y, u, v)$
(possible normalized, or complex conjugates)

变换编码

Other basis sets:

Walsh-Hadamard

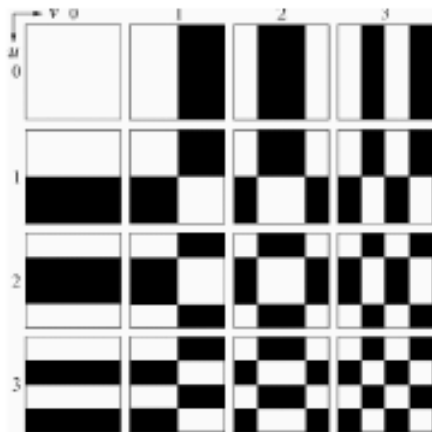


FIGURE 8.29 Walsh-Hadamard basis functions for $N = 4$. The origin of each block is at its top left.

Cosine

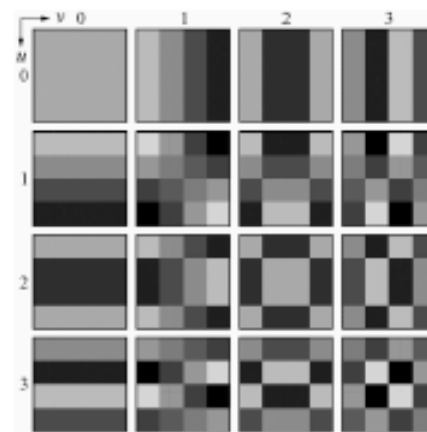


FIGURE 8.30 Discrete-cosine basis functions for $N = 4$. The origin of each block is at its top left.



离散余弦变换

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

where

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } u = 0 \\ \sqrt{\frac{2}{N}} & \text{otherwise} \end{cases}$$

离散余弦变换

How can we get away with cosines and no sines?

Assumes alternating periodicity instead of periodicity

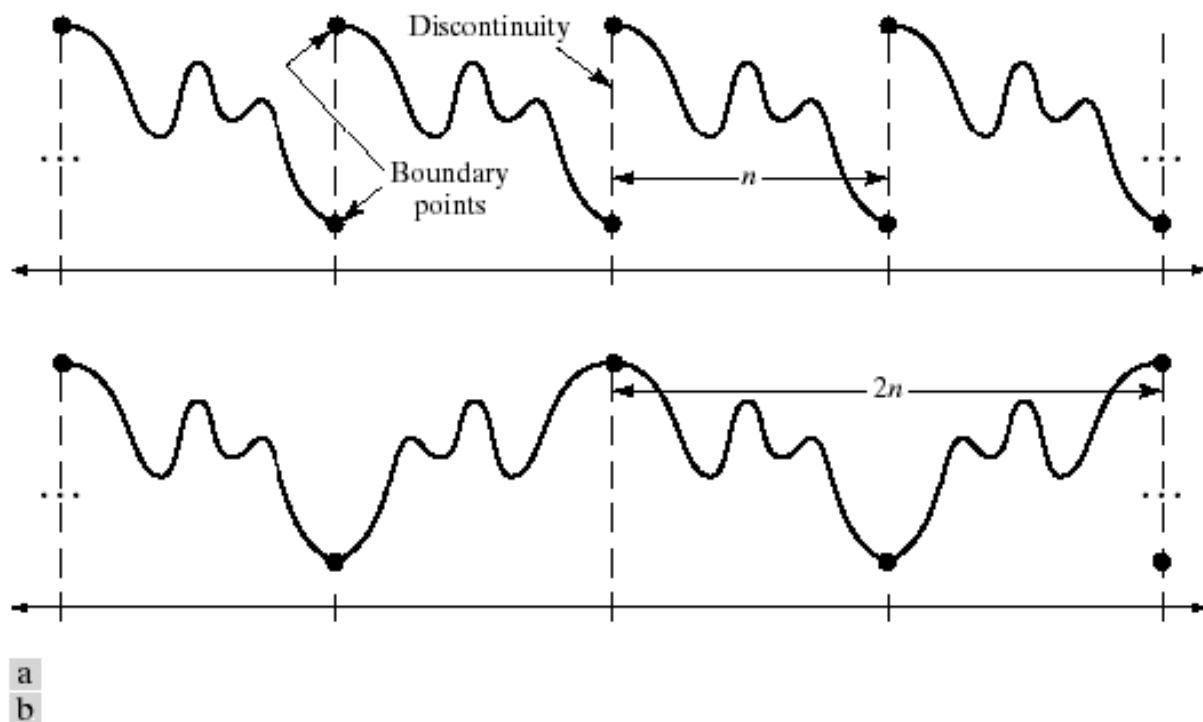


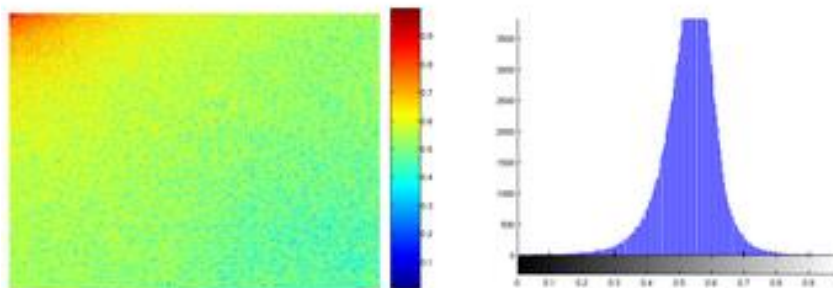
FIGURE 8.32 The periodicity implicit in the 1-D (a) DFT and (b) DCT.



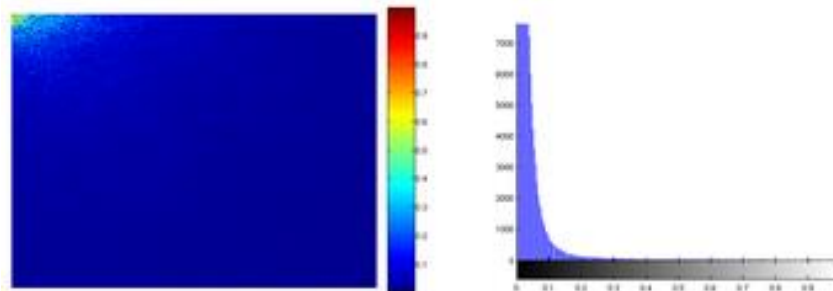
离散余弦变换



DFT



DCT



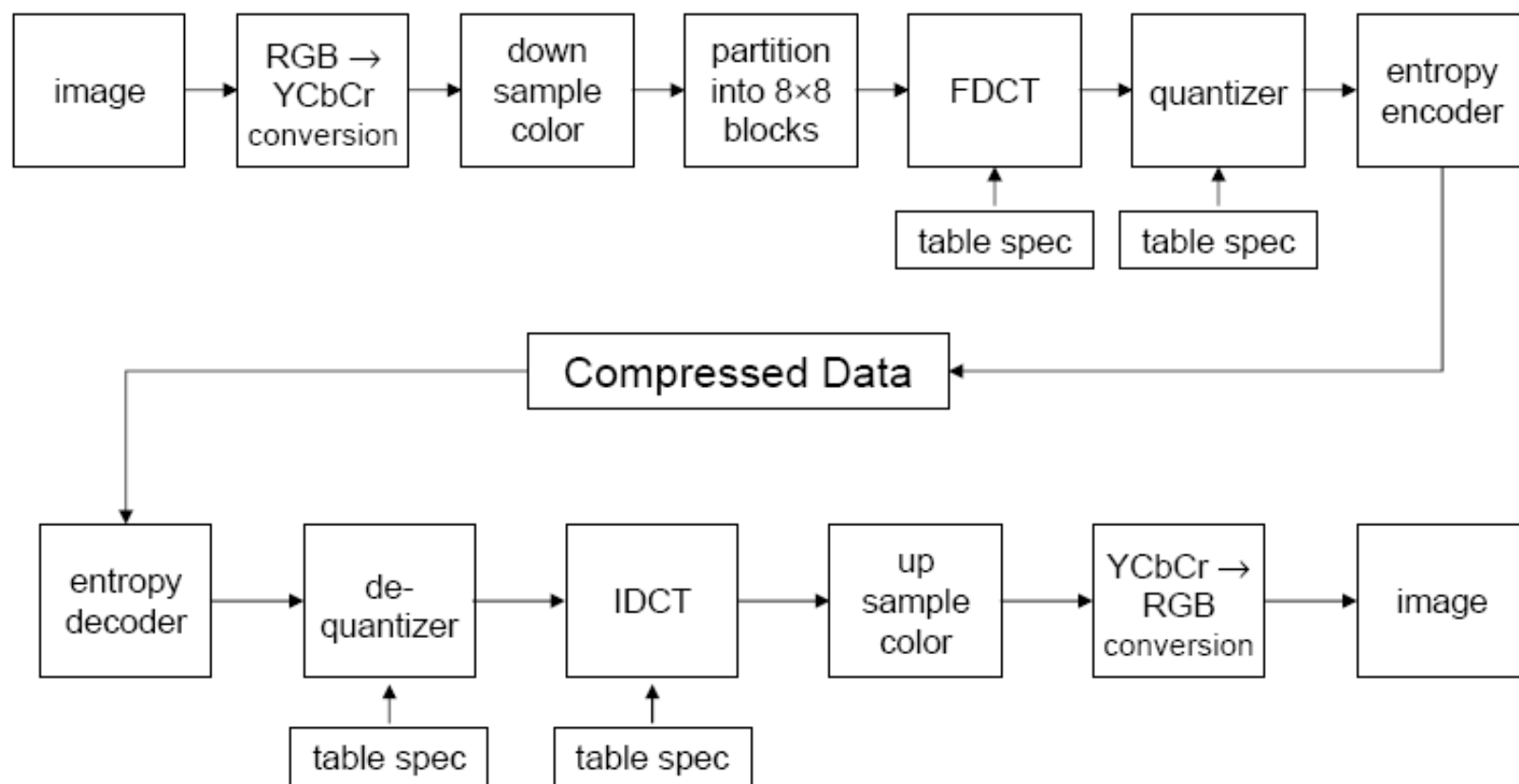


JPEG

From the Article by Wallace:

Title: The JPEG Still Picture Compression Standard
Source: Communications of the ACM archive
Volume 34, Issue 4 (April 1991)
Special issue on digital multimedia systems
Pages: 30-44
Year: 1991
ISSN: 0001-0782
Author: Gregory K. Wallace
Digital Equipment Corp., Maynard, MA
Publisher: ACM Press New York, NY, USA

JPEG 编码流程



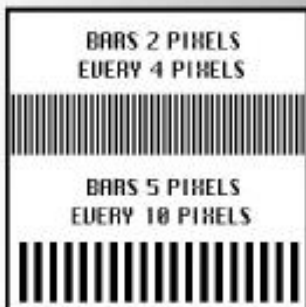
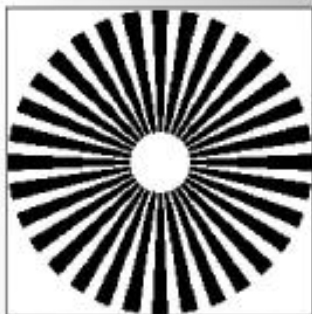


JPEG Baseline Process

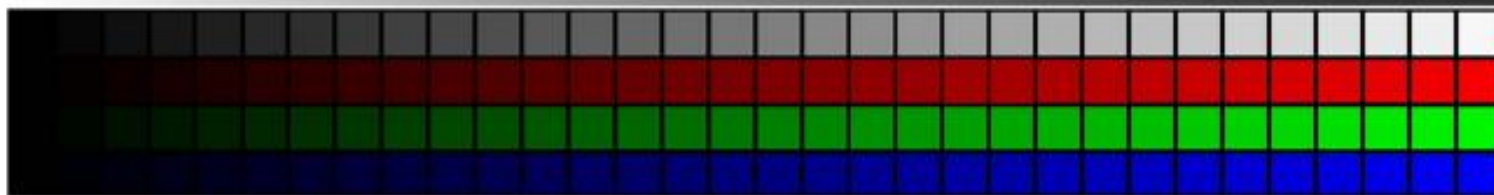
- DCT-based process
- Source image: 8-bit samples within each component
- Sequential
- Huffman coding: 2 AC and 2 DC tables
- Decoders process scans with 1, 2, 3, or 4 components
- Interleaved and non-interleaved scans

标准测试图像

IPTC COLOUR TEST CHART 2048 x 1440 Pixels RGB

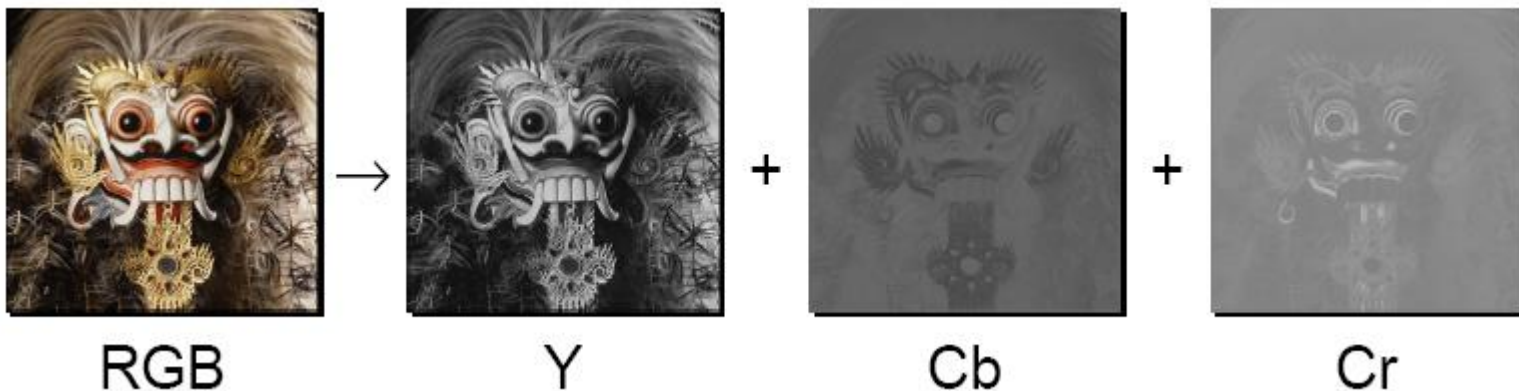


E.A. WABIT ©



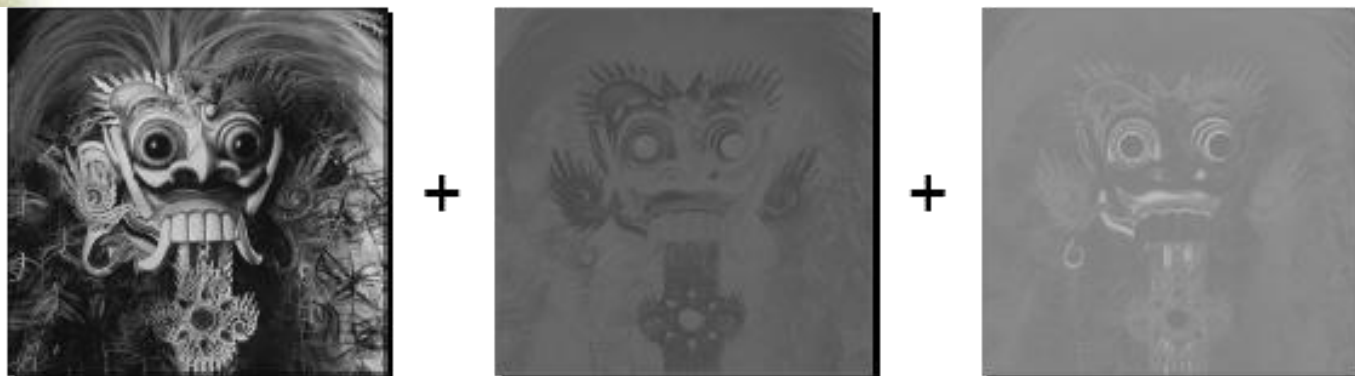
颜色转换

RGB \rightarrow YCbCr



$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.25678824 & 0.50412941 & 0.09790588 \\ -0.14822290 & -0.29099279 & 0.43921596 \\ 0.43921569 & -0.36778831 & -0.07142737 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$$

颜色通道下采样



+



+



by a factor
of 2

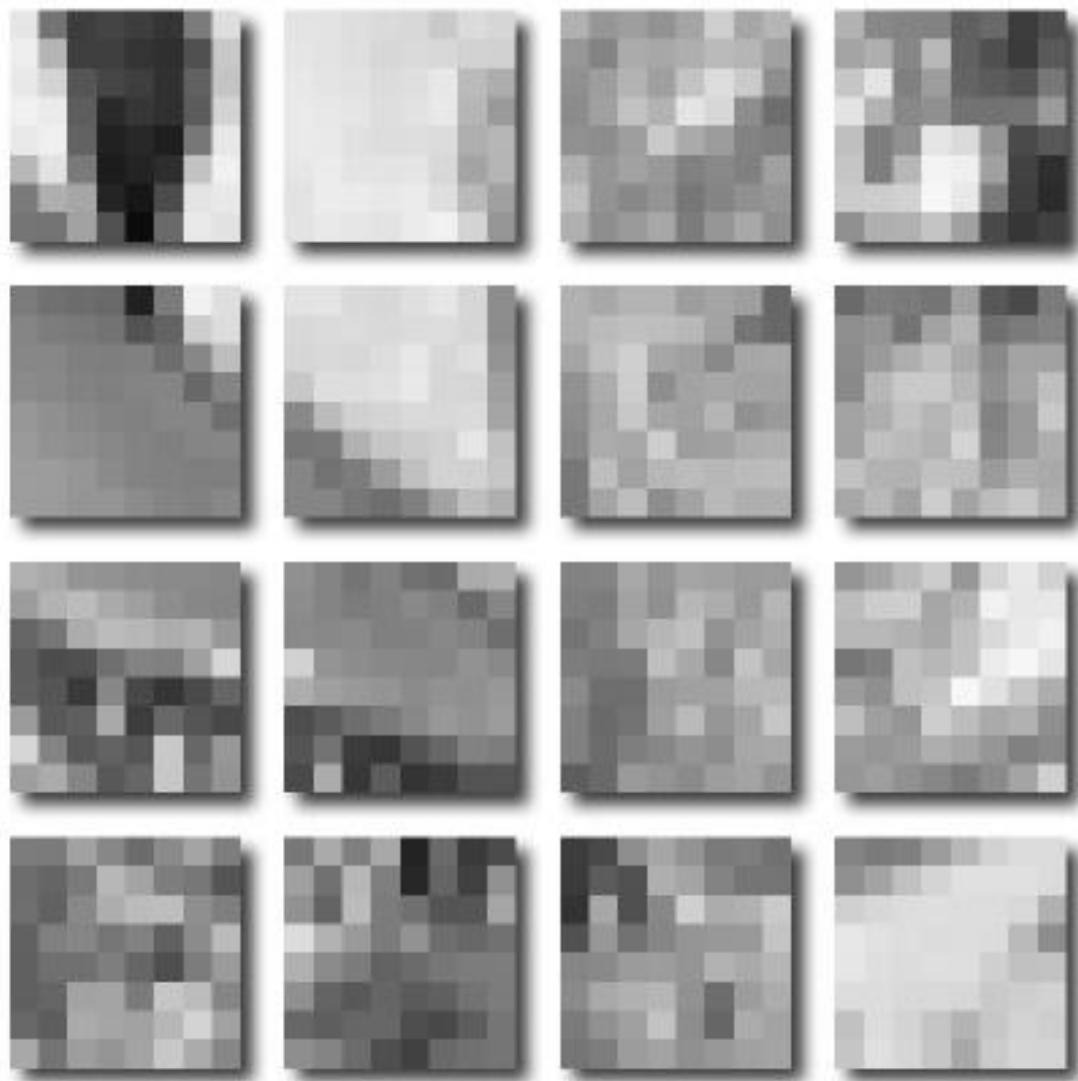
Y

Cb

Cr



各个通道分割成 8×8 小块





为离散余弦变换进行预处理

- Each image band is operated on independently.
- Each 8×8 image block is operated on independently.
- Shift all image values from $I \in [0, 2^P - 1]$ to $I \in [-2^{P-1}, 2^{P-1} - 1]$. E.g. $[0, 255] \rightarrow [-128, 127]$.
- The FDCT decomposes each block into a set of coefficients with respect to the 64 orthogonal basis functions shown on the next slide.



离散余弦变换

$\mathcal{F}\{\mathbf{I}\}(v, u; r, c)$ - DCT of 8×8 block from \mathbf{I} starting at (r, c)

$\Lambda(\xi)$ - Normalization Factor

$\phi(v, u; \rho, \chi)$ - 2D Cosine Basis Function, (v, u)

r - image row index (vertical, increasing down)

c - image column index (horizontal, increasing right)

ρ - DCT row index (horizontal wave fronts, vertical propagation down)

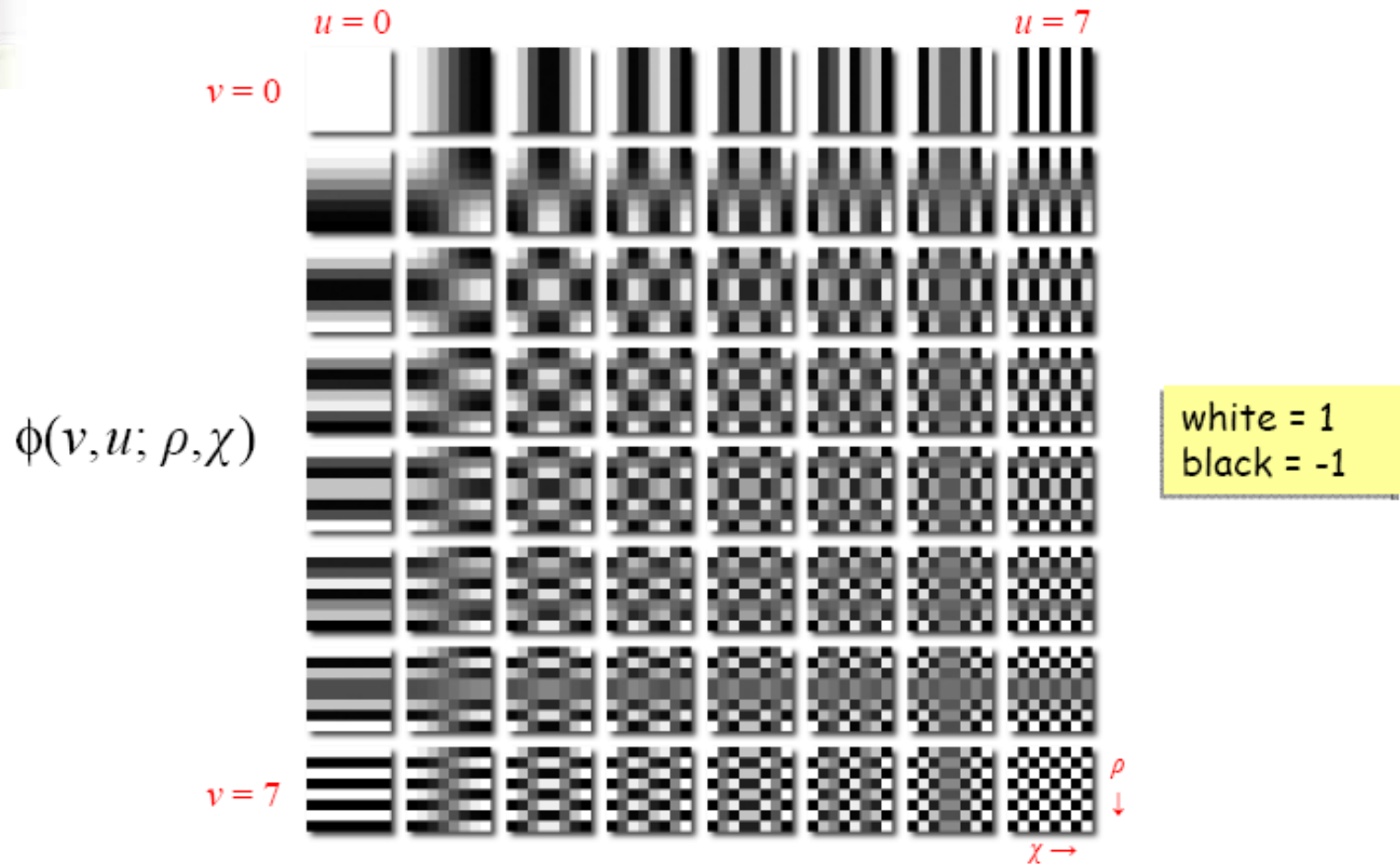
χ - DCT column index (vertical wave fronts, horizontal propagation right)

v - DCT row frequency index (vertical, increasing down)

u - DCT column frequency index (horizontal, increasing right)



离散余弦变换的基函数

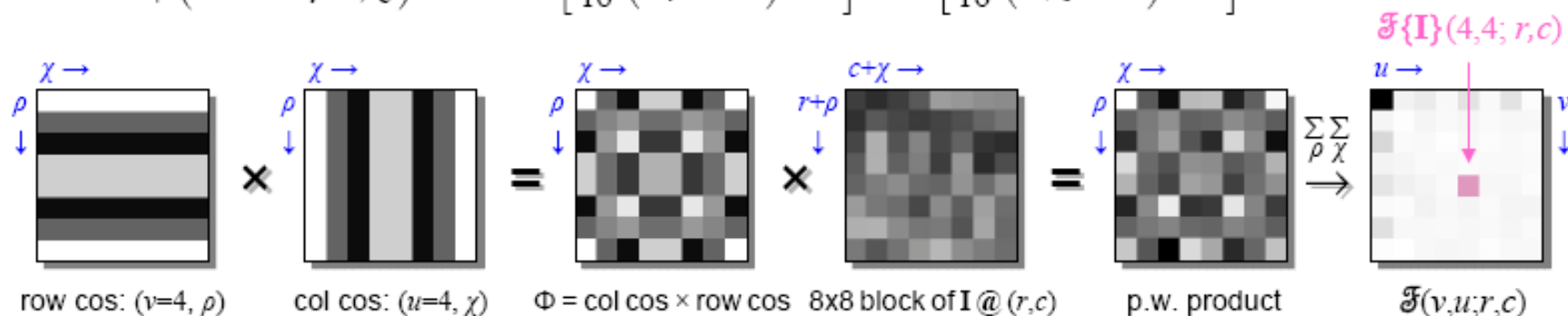


离散余弦变换

$$\mathcal{F}\{\mathbf{I}\}(v, u; r, c) = \sum_{\rho=0}^7 \sum_{\chi=0}^7 \frac{1}{4} \Lambda(v) \Lambda(u) \phi(v, u; \rho, \chi) \mathbf{I}(r + \rho, c + \chi)$$

$$\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } \xi = 0 \\ 1 & \text{otherwise} \end{cases} \quad \begin{array}{l} (r, c) \in \{0, 8, 16, \dots, R\} \times \{0, 8, 16, \dots, C\}, \\ (v, u), (\rho, \chi) \in \{0, \dots, 7\} \times \{0, \dots, 7\}. \end{array}$$

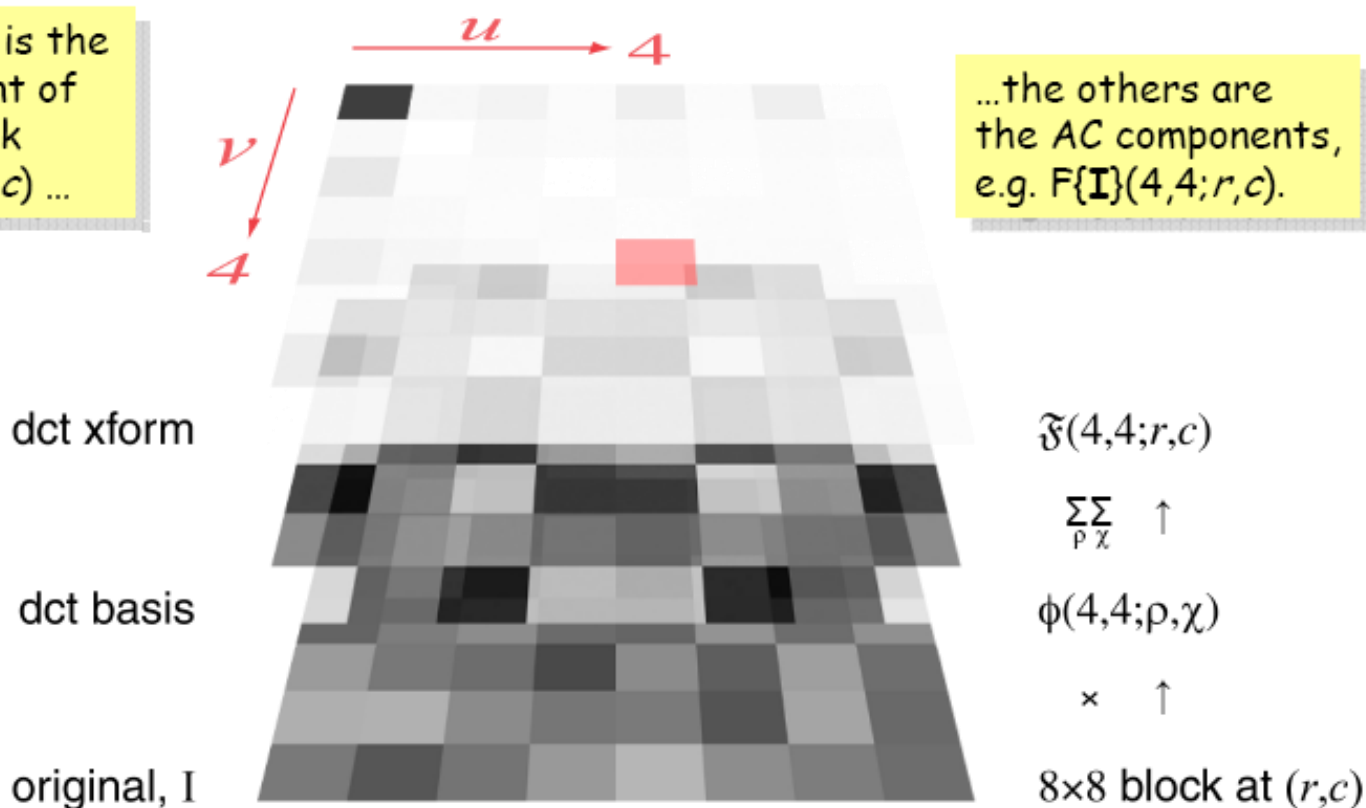
$$\phi(v, u; \rho, \chi) = \cos\left[\frac{1}{16}(2\rho + 1)\pi v\right] \cos\left[\frac{1}{16}(2\chi + 1)\pi u\right]$$

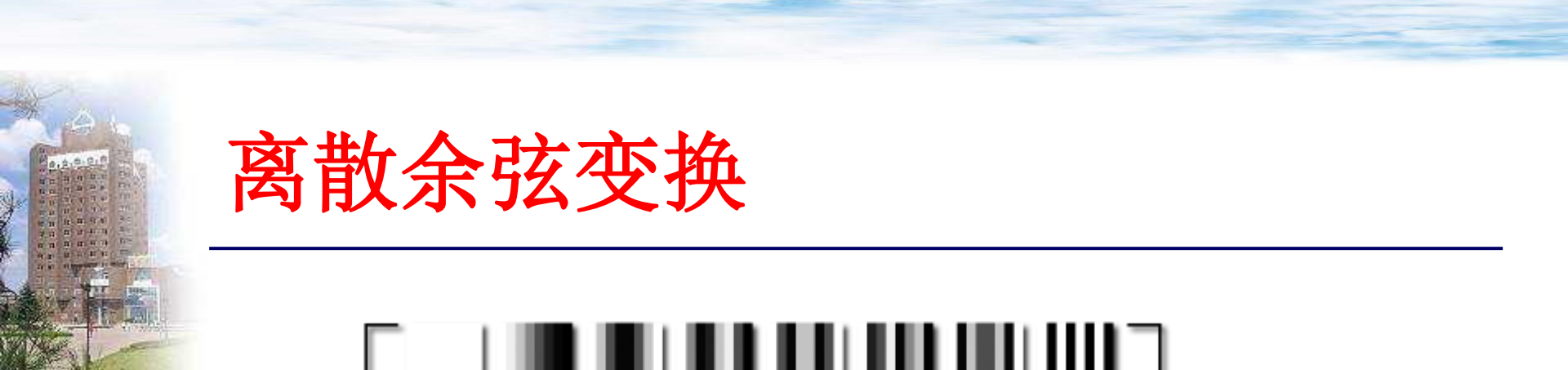


离散余弦变换

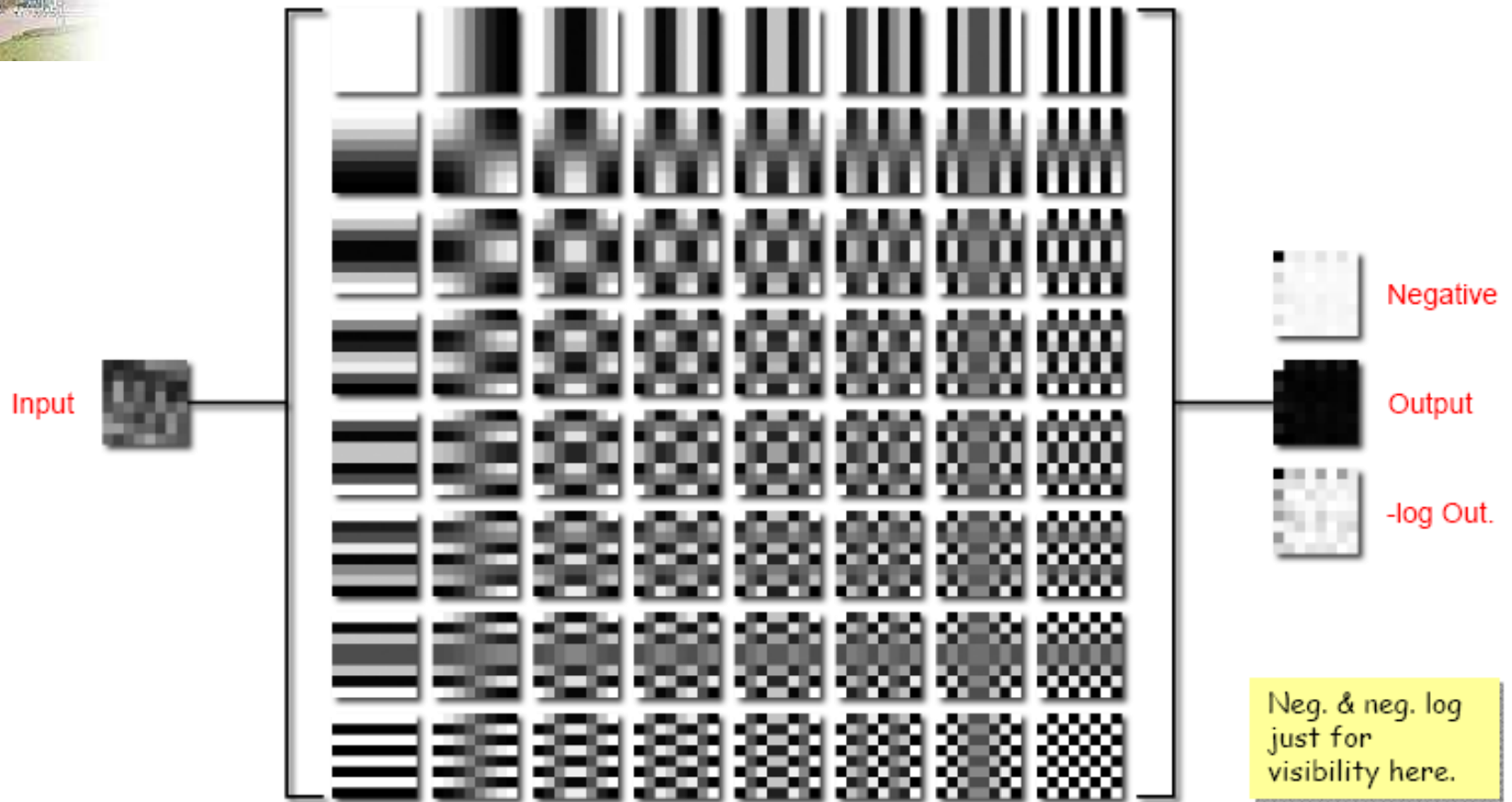
$F\{\mathbf{I}\}(0,0;r,c)$ is the DC component of the 8×8 block from \mathbf{I} at (r,c) ...

...the others are the AC components, e.g. $F\{\mathbf{I}\}(4,4;r,c)$.





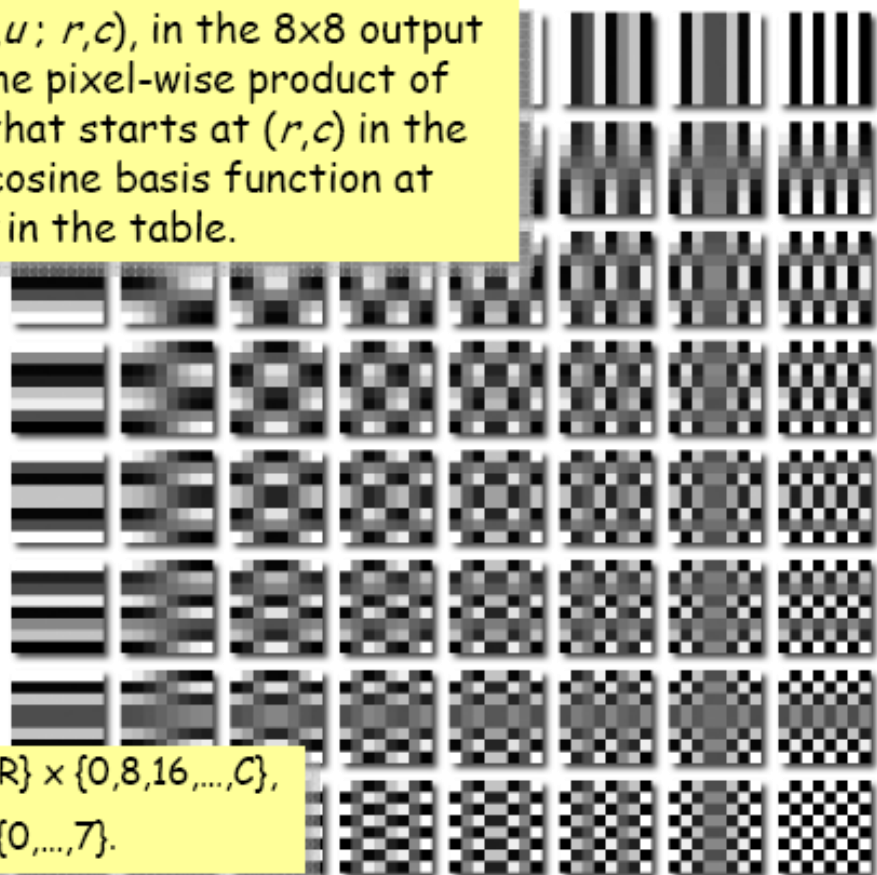
离散余弦变换



离散余弦变换

Each value, $F(v,u; r,c)$, in the 8×8 output is the sum of the pixel-wise product of the 8×8 block that starts at (r,c) in the image and the cosine basis function at row v , column u in the table.

Input



Negative



Output



$-\log$ Out.

$(r,c) \in \{0,8,16,\dots,R\} \times \{0,8,16,\dots,C\},$
 $(v,u) \in \{0,\dots,7\} \times \{0,\dots,7\}.$

Neg. & neg. log
just for
visibility here.



例子：离散余弦变换的量化表

Luminance Quantization Table

2	2	3	4	5	6	8	11
2	2	2	4	5	7	9	11
3	2	3	5	7	9	11	12
4	4	5	7	9	11	12	12
5	5	7	9	11	12	12	12
6	7	9	11	12	12	12	12
8	9	11	12	12	12	12	12
11	11	12	12	12	12	12	12

Precision: 8 bits

Approximate quality factor: 91.64

Scaling: 16.71 Variance: 22.54

Chrominance Quantization Table

3	3	7	13	15	15	15	15
3	4	7	13	14	12	12	12
7	7	13	14	12	12	12	12
13	13	14	12	12	12	12	12
15	14	12	12	12	12	12	12
15	12	12	12	12	12	12	12
15	12	12	12	12	12	12	12
15	12	12	12	12	12	12	12

Precision: 8 bits

Approximate quality factor: 92.57

Scaling: 14.85 Variance: 23.00

离散余弦变换的量化

255	23	34	13	44	11	44	6
19	4	19	12	18	9	16	15
58	10	11	2	14	6	12	7
22	18	9	12	7	11	12	9
42	26	19	9	23	15	16	6
9	6	23	14	20	10	19	21
48	13	13	11	15	12	18	10
3	0	13	18	16	4	11	14

Output of FDCT, $\mathcal{F}(u,v)$, at image pixel location (r,c) .

1	1	1	2	3	6	8	10
1	1	2	3	4	8	9	8
2	2	2	3	6	8	10	8
2	2	3	4	7	12	11	9
3	3	8	11	10	16	15	11
3	5	8	10	12	15	16	13
7	10	11	12	15	17	17	14
14	13	13	15	15	14	14	14

Quantization Table, $Q(u,v)$

255	23	34	7	15	2	6	1
19	4	10	4	5	1	2	2
29	5	6	1	2	1	1	1
11	9	3	3	1	1	1	1
14	9	2	1	2	1	1	1
3	1	3	1	2	1	1	2
7	1	1	1	1	1	1	1
0	0	1	1	1	0	1	1

Quantized Result, $\mathcal{F}^Q(u,v)$ at image pixel location (r,c) .

$$\mathcal{F}^Q(u,v) = \text{round} \left(\frac{\mathcal{F}(u,v)}{Q(u,v)} \right)$$

$(r,c) \in$
 $\{0,8,16,\dots,R\} \times \{0,8,16,\dots,C\},$

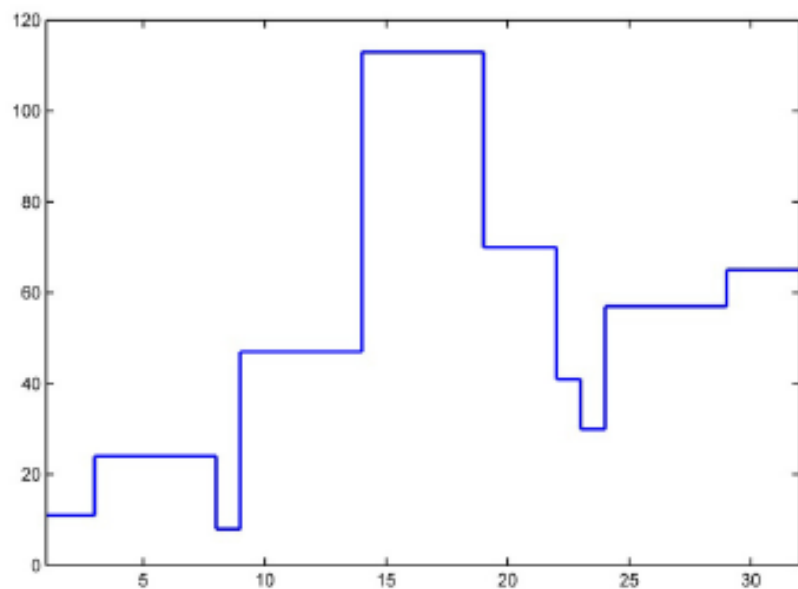
This $Q(u,v)$ is yet another quantization table.

图像直流成分(8×8 constant blocks)

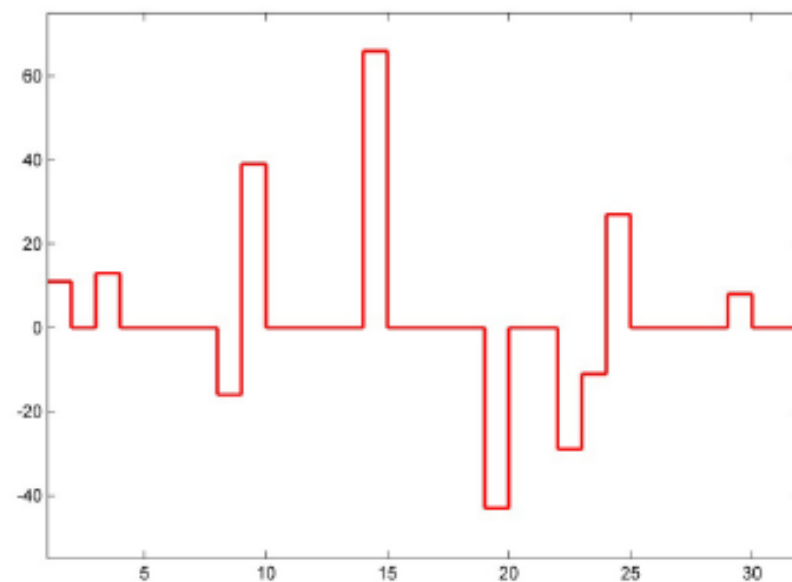




直流成分编码(DPCM差分脉冲编码调制)



DC components from 32 blocks



Differential PCM coding of same

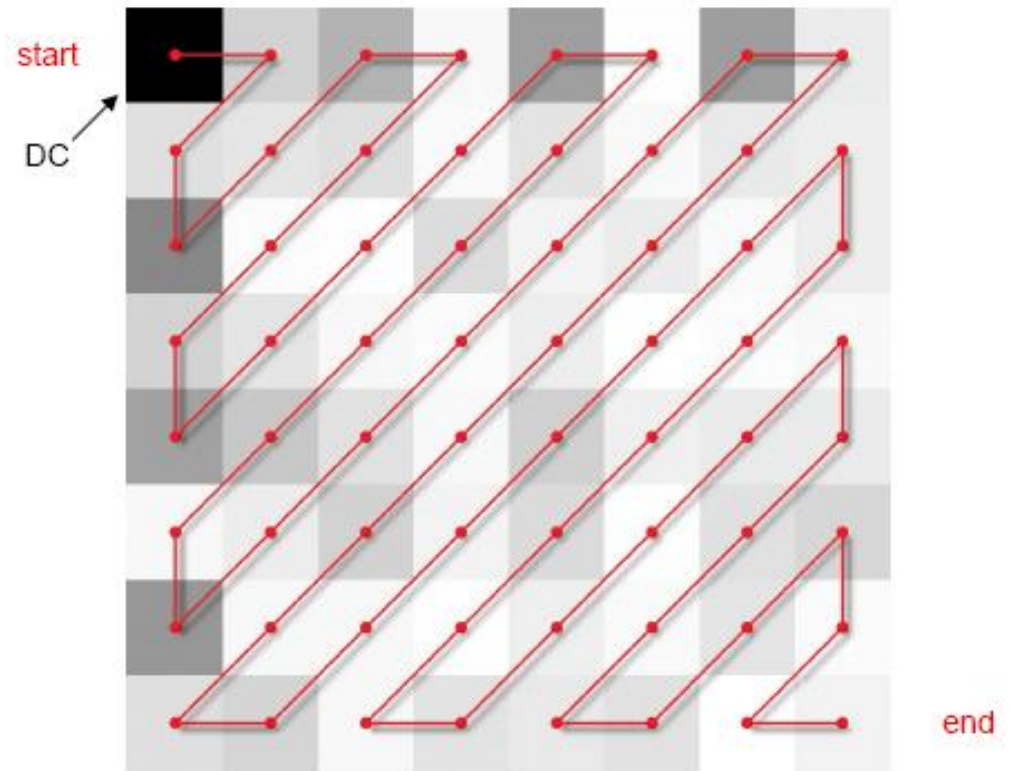
DC components are coded separately from the AC components.



直流成分编码(DPCM差分脉冲编码调制)

Quantized
coefficient
encoding
order

255	23	34	7	15	2	6	1
19	4	10	4	5	1	2	2
29	5	6	1	2	1	1	1
11	9	3	3	1	1	1	1
14	9	2	1	2	1	1	1
3	1	3	1	2	1	1	2
7	1	1	1	1	1	1	1
0	0	1	1	1	0	1	1



AC: 23 19 29 4 34 7 10 5 11 14 9 6 4 15 2 5 1 3 9 3 7 1 2 3 2 1 6 1 2 1 1 1 3 1 0 0 1 1 2 1 1 2 1 1 1 2 1 1 1 1 1



交流成分熵编码

Two step process:

1. Convert zigzag sequence of quantized coefficients into a sequence of symbols.
2. Convert symbols to a data stream of variable length codes.

Symbol 1 = (Runlength, Size) Symbol 2 = Amplitude
--



交流成分熵编码

Symbol 1: (Runlength, Size);

Symbol 2: Amplitude

AC:

Runlength: number of consecutive zero values (0-15)

Size: number of bits used to encode amplitude (1-10)

Amplitude: quantized value

DC:

Runlength: not included

Size: number of bits used to encode amplitude (1-11)

Amplitude: quantized value



交流成分熵编码

	size	amplitude
	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...128
	8	-255...-128,128...255
	9	-511...-256,256...511
	10	-1023...-512,512...1023
DC only	11	-2047...-1024,1024...2047

Symbol 2 Coding



效果展示

Quality vs File Size

