2022春

数字图像处理

图像压缩

曹劲舟

助理教授

深圳技术大学

大数据与互联网学院

2022年4月

本章内容

- ✓图像压缩的概念
- ✓编码冗余
- ✓空间冗余
- ✓不相关信息
- ✓JPEG压缩

图像数据压缩

- **图像压缩**是一种减少描绘一幅图像所需数据量的技术和科学， 它是数字图像处理领域最有用、商业上最成功的技术之一。
- 图像编码与压缩是数据压缩的一个分支，就是对图像数据按一定的规则进行变换和组合，达到以尽可能少的代码（符号）来表示尽可能多的图像信息。
- **图像压缩是为了减少描述数字图像数据量的问题，除去图像中多余的数据而对信息没有本质的影响。**
- 从数学角度看，这一过程实际上就是将二维像素矩阵变换为一个在统计上无关联的数据集合。

图像压缩的必要性

✓数字图像通常文件尺寸比较大，这给图像的传输和存储带来相当大的困难。数据的压缩是必不可少的。

✓优点：节省存储空间；减少传输时间；利于处理，降低处理成本

图像（未压缩）总的文件大小:
420x280x3= 352,800 bytes

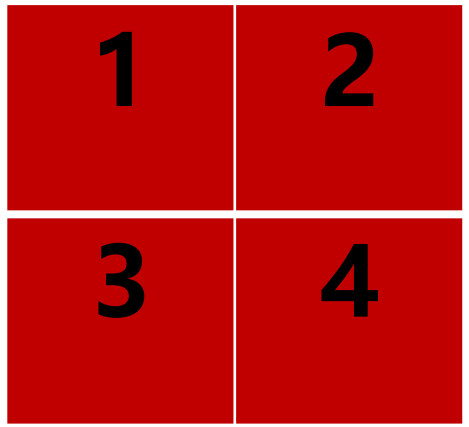
宽：420像素 高：280像素

每个像素由8比特R、G、B分量组成

一张4k x 3k的照片4000x3000x3 = 36,000,000 bytes -> 36MB

图像压缩的必要性

- 描述语言
  - 1） “这是一幅 2*2的图像，图像的第一个像素是红的，第二个像素是红的，第三个像素是红的，第四个像素是红的”。
  - 2） “这是一幅2*2的图像，整幅图都是红色的”。



由此我们知道，**整理图像的描述方法可以达到压缩的目的。**

图像数据冗余

- **数据冗余的概念**
  - 数据是用来表示信息的。
  - 如果不同的方法为表示给定量的信息使用了不同的数据量，那么使用较多数据量的方法中，有些数据必然是代表了无用的信息，或者是重复地表示了其它数据已表示的信息，这就是数据冗余的概念。

图像数据冗余

■ 相对数据冗余的定义

如果n_1和n_2代表两个表示相同信息的数据集合中所携载信息单元的数量，则n_1表示的数据集合的相对数据冗余R_D定义为：

$$R_D = 1 - \frac{1}{C_R}$$

C_R称为压缩比，定义为

$$C_R = \frac{n_1}{n_2}$$

图像数据冗余

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

图像压缩的方法

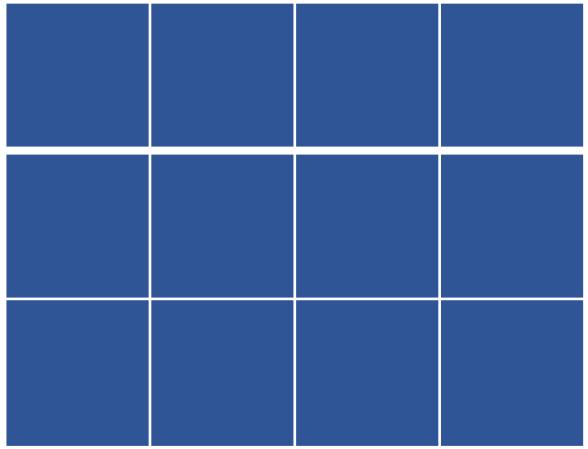
✓消除冗余数据，从数学角度看，将原始图像转化为从统计角度看尽可能不相关的数据集。

✓一般分为两类：

- ✓**无损压缩：**在压缩和解压缩过程中没有信息损失。主要用于图像存档，但压缩比有限。
- ✓**有损压缩：**能取得较高的压缩率，但压缩后不能通过解压缩恢复原状。数字电视，图像传输和多媒体应用场合常用这类压缩。
- ✓其它：如根据需要，即可进行无损，也可进行有损压缩的技术；准无损技术。

图像压缩的方法

✓无损压缩原理



RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB
RGB	RGB	RGB	RGB

16	RGB

从原来的16*3*8=284bits压缩为： (1+3)*8=32bits

图像压缩的方法

✓有损压缩原理

36	35	34	34	34
34	34	32	34	34
33	37	30	34	34
34	34	34	34	34
34	35	34	34	31

34	34	34	34	34
34	34	34	34	34
34	34	34	34	34
34	34	34	34	34
34	34	34	34	34

25	34

图像数据冗余

图像压缩是通过去除一个或三个基本数据冗余来实现。

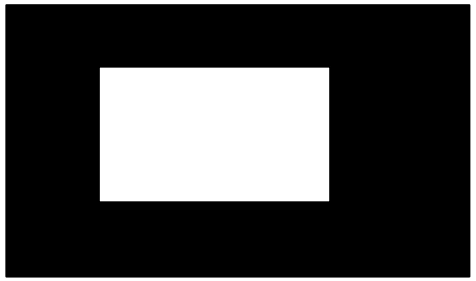
- ✓**编码冗余**：**图像所用码字少于最佳编码长度**
- ✓**空间/时间冗余**：**图像像素间相关造成的冗余（像素冗余）**
  - ✓图像中相邻像素间的相关性引起的**空间冗余**；
  - ✓图像序列中不同帧之间存在相关性引起的**时间冗余**
- ✓**不相关信息**：**人类视觉系统忽略的数据所造成的冗余（视觉冗余）**

✓如果能减少或消除上述三种冗余的1种或 多种冗余，就能取得数据压缩的效果

编码冗余

什么是编码冗余?

- 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。
- 黑白二值图像编码



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

图像编码与压缩，本质上说，就是对图像原数据按一定的规则进行变换和组合，从而达到以尽可能少的代码来表示尽可能多的数据信息。
压缩通过编码来实现，或者说**编码带来压缩的效果**，所以，一般把此项处理称之为**压缩编码**。

编码冗余

平均编码长度L_{avg}：每个像素所需的平均比特数：

变长编码：$$L_{avg} = \sum_{k=0}^{7} l_2(r_k)p_r(r_k) = 2.7 \text{ bits}$$

定长编码：$L_{avg} = 3$ bits

$l_2(r_k)$是对应像素灰度级的编码长度

例：

r_k	$p_r(r_k)$	Code 1	$l_1(r_k)$	Code 2	$l_2(r_k)$
$r_0 = 0$	0.19	000	3	11	2
$r_1 = 1/7$	0.25	001	3	01	2
$r_2 = 2/7$	0.21	010	3	10	2
$r_3 = 3/7$	0.16	011	3	001	3
$r_4 = 4/7$	0.08	100	3	0001	4
$r_5 = 5/7$	0.06	101	3	00001	5
$r_6 = 6/7$	0.03	110	3	000001	6
$r_7 = 1$	0.02	111	3	000000	6

压缩比为：$C = 3/2.7 = 1.11$　　R=1-1/C=0.1

编码冗余

不同的编码方法可能会有不同的L_{avg}，由此引出两种编码冗余。

1.相对编码冗余：L_{avg}大的编码相对于L_{avg}小的编码就存在相对于编码冗余。

2.绝对编码冗余：使$L_{avg} > L_{\min}$的编码就存在绝对编码冗余。

图像压缩冗余度和编码效率

如何度量编码方法的优劣？（速度，效率， 保真度）

在不丢失信息的条件下，**是否存在**一个最小数据量来足够充分描述一幅图像？

图像信息熵H(d)

令集合为$\{d_1, d_2, \ldots, d_m\}$图像像素灰度级的集合,其对应概率为$p(d_1), p(d_2),\ldots, p(d_m)$，则熵定义为：

$$H(d) = -\sum_{i=1}^{m} p(d_i)\log_2 p(d_i)$$

单位：比特/像素

H表示对输入信源元素集合进行编码时所需要的平均位数的下限。 d_i 出现的概率相等时，熵最大。

Matlab 计算熵值

考虑一幅简单的 4x4图像，下面的命令行顺序生成一幅这样的图像并计算熵的一阶估计：

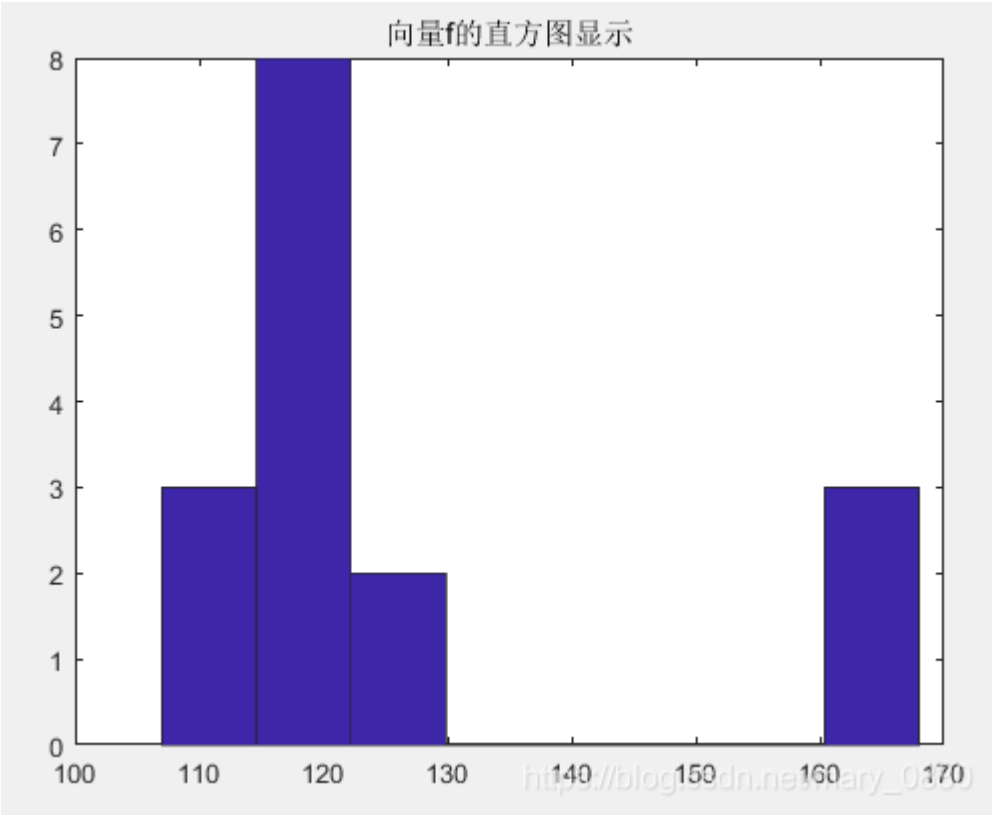
调用**函数ntrop**来处理

```
f = [119 123 168 119;123 119 168 168];
f = [f;119 119 107 119;107 107 119 119]
p =hist(f(:),8);
[p x]=hist(f(:),8),title('向量f的直方图显示')
p = p/sum(p) %直方图各个灰度级概率
h = ntrop(f)
```

```
p =

    0.1875    0.5000    0.1250         0         0         0         0    0.1875
```

```
f =

   119   123   168   119
   123   119   168   168
   119   119   107   119
   107   107   119   119
```

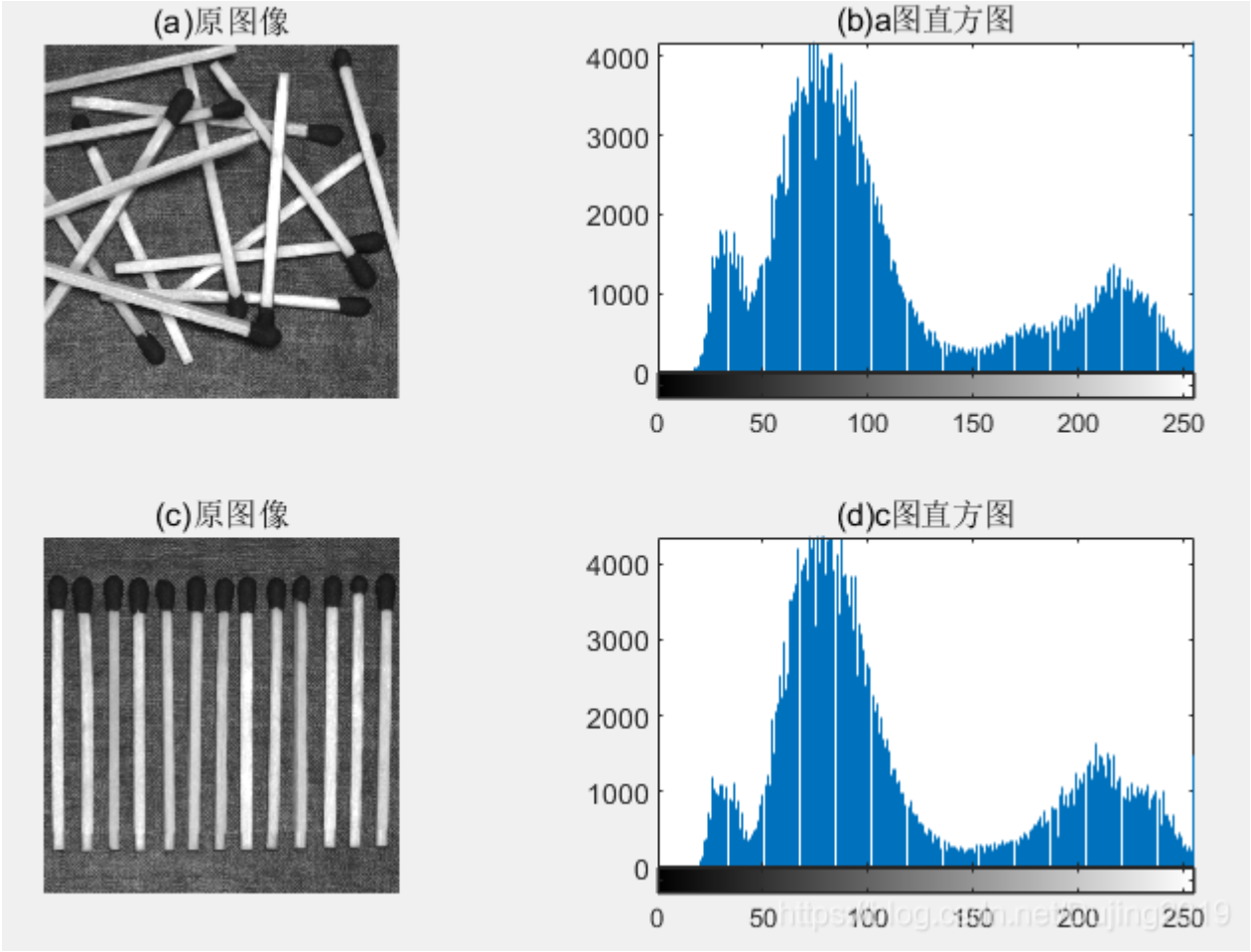
```
>> h=ntrop(f)

h =

    1.7806
```



空间冗余

两张图片的直方图非常相似，且图片内容都是火柴，只不过第一张图是随意摆放的，第二张图是按照顺序摆放的

```
f1=imread('D:\数字图像处理\第八章学习
\Fig0807(a).tif');
f2=imread('D:\数字图像处理\第八章学习
\Fig0807(c).tif');
subplot(2, 2, 1), imshow(f1);title('(a)原图像');
subplot(2, 2, 2), imhist(f1);title('(b)a图直方图');
subplot(2, 2, 3), imshow(f2);title('(c)原图像');
subplot(2, 2, 4), imhist(f2);title('(d)c图直方图');
```



空间冗余

```
c1 = mat2huff(f1);
ntrop(f1)
imratio(f1,c1)
c2 = mat2huff(f2);
ntrop(f2)
imratio(f2,c2)
```

```
ans =

    7.4253


ans =

    1.0704
```

```
ans =

    7.3505


ans =

    1.0821
```

- 不同图像具有**大致相同的直方图和熵(分别为7.4253和 7.3505 比特/像素)**，且其编码压缩比也大致一样(压缩比为 1.0704 和 1.0821)。
- 表明**变长编码的设计**并不适用于内容整齐，及像素排列整齐的这类图片。
- 任何一幅图像的像素都可以合理地从它们的相邻像素值预测，这些相关性是像素间冗余的潜在基础。

空间冗余Spatial Redundancy

- **什么是空间冗余?**
  - 空间冗余是静态图像中存在的最主要的一种数据冗余。
  - 同一景物表面上采样点的颜色之间通常存在着空间相关性，相邻各点的取值往往相近或者相同，这就是空间冗余。
    - 例如：图片中有一片连续区域，这个区域的像素都是相同的颜色。
  - 为了减少像素间冗余，通常必须把由人观察和解释的二维像素数组变换为更有效的格式 (但通常是“不可视的”) 。
    - 例如，邻近像素点之间的差值可以用来表示一幅图像。**移走像素间冗余**的变换称为**映射**。如果原始图像可以从变换的数据集重建，它们就被称为**可逆映射**。
  - 对于一幅图像，很多单个像素的值可以通过与**它相邻的像素值为基础进行预测**。

心理视觉冗余

■什么是心理视觉冗余?

- 人眼感觉到的图像区域亮度不仅取决于该区域的反射光，例如根据马赫带效应，在灰度值为常数的区域也能感觉到灰度值的变化，这是由于眼睛对所有视觉信息感受的灵敏度不同。
- 在正常视觉处理过程中各种信息的相对重要程度不同，有些信息在通常的视觉过程中与另外一些信息相比并不那么重要，这些信息被认为是心理视觉冗余的，需要被删除。
- **去除这些信息并不会明显降低图像质量。**

心理视觉冗余

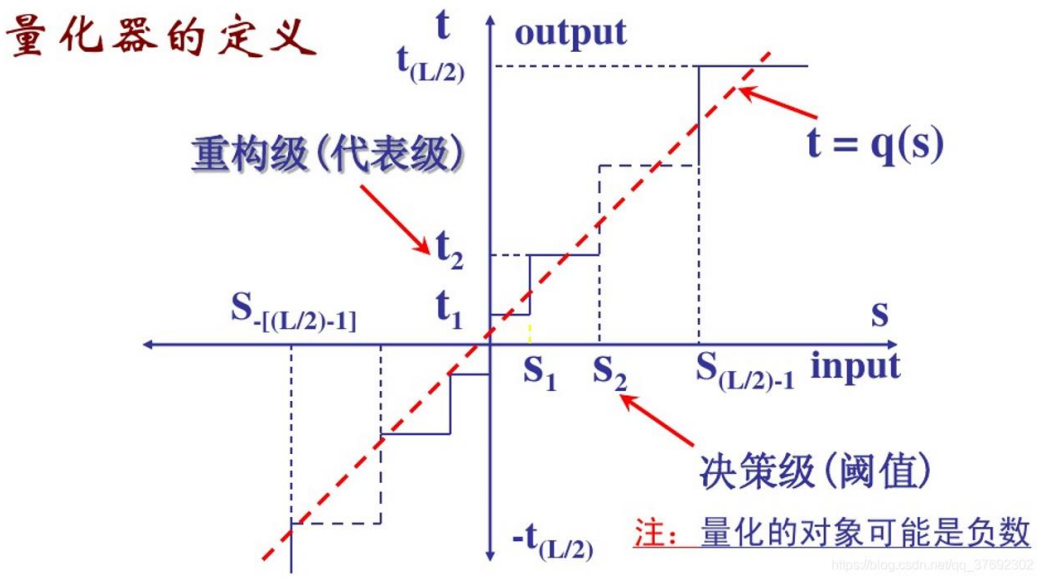
什么是心理视觉冗余?

- 由于消除心理视觉冗余数据会导致一定量信息的丢失，所以这一过程通常称为**量化**。
- 心理视觉冗余压缩是**不可恢复的**，量化的结果导致了数据**有损压缩**。

心理视觉冗余

量化

- 减少数据量的最简单的办法是将图像量化成较少的灰度级，通过减少图像的灰度级来实现图像的压缩
- 这种量化是不可逆的，因而解码时图像有损失。

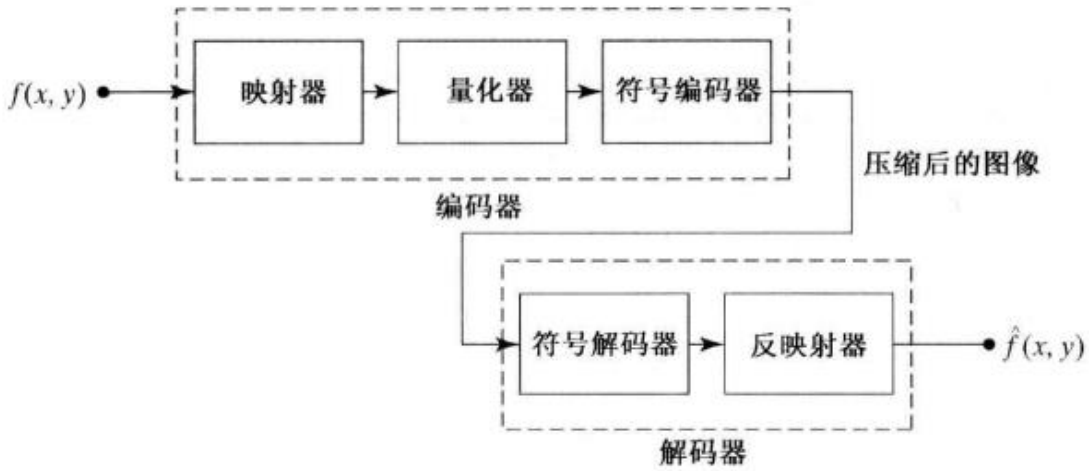


如果输入时256个灰度级，对灰度级量化后输出，只剩下4个层次，数据量被大大减少。

图像压缩系统

- **图像压缩系统由两个截然不同的结构块组成的：编码器和解码器。**

- **编码器**
  - **映射器**：降低（像素冗余）空间和时间冗余**（可逆）**
  - **量化器**：减少视觉心理冗余（不可逆）
  - **符号编码器**：减少编码冗余，如使用哈夫曼编码
  - 并不是每个图像压缩系统都必须包含这3种操作，如**进行无误差压缩时，必须去掉量化器**
- **解码器**
  - **符号解码器：**进行编码的逆操作
  - **反向映射器：**进行转换器的逆操作
  - 为什么没有反向量化器?

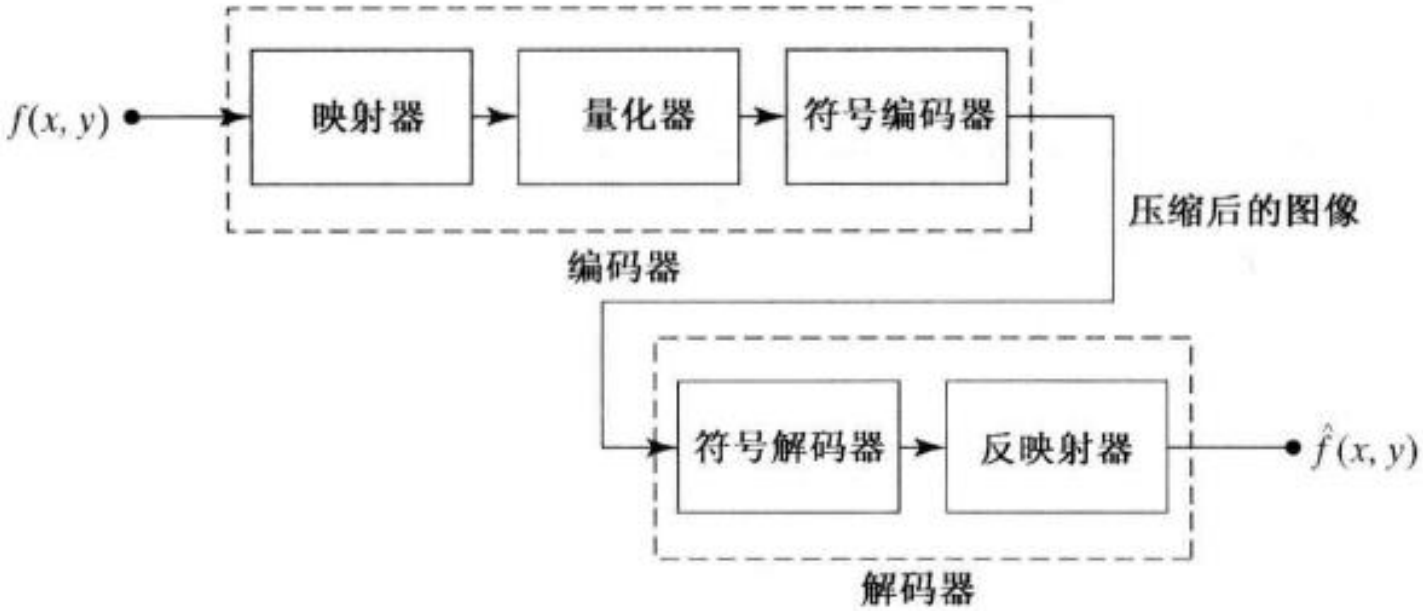


一个通用图像压缩系统的功能方框图

图像压缩系统

- 一般而言，$f\hat{}(x,y)$可能是，也可能不是$f\hat{}(x,y)$的精确表示。
  - 如果是，系统就是无误差的、信息保持的或无损的；
  - 如果不是，在重建图像中会有一部分失真。
- 对于后一种情况，被称为有损压缩，可以对x和y的任意取值在f(x,y)和f^(x,y)之间定义误差e(x,y):

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



一个通用图像压缩系统的功能方框图

压缩的(也就是编码后的)图像，把图像发送到解码器，产生重建的输出图像$f\hat{}(x,y)$ 。

图像压缩保真度准则（评价压缩算法的标准）

a）输入图和输出图之间的均方根（e_{rms}）误差

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

b）输入图和输出图的均方根信噪比

$$SNR_{ms}=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}\,(x,y)^2/\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}\,(x,y)-f(x,y)]^2$$

c）峰值信噪比 PSNR

$$\mathbf{PSNR}=\mathbf{10}\log_{10}\{f_{max}{}^2/\textstyle\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}\,(x,y)-f(x,y)]^2\}$$

主观的保真度准则

- 具有相同客观保真度的不同图像，在人的视觉系统中有不同的视觉效果。因此，在某些应用下用**主观**的方法来测量图像的质量。

- 主观评价通常的方法是让一组人（不少于20）观察者观看图像并打分，来评价图像的主观质量。

电视图像质量评价尺度

评分	1	2	3	4	5	6
评价	优秀	良好	可用	刚可看	差	不能用

图像压缩系统

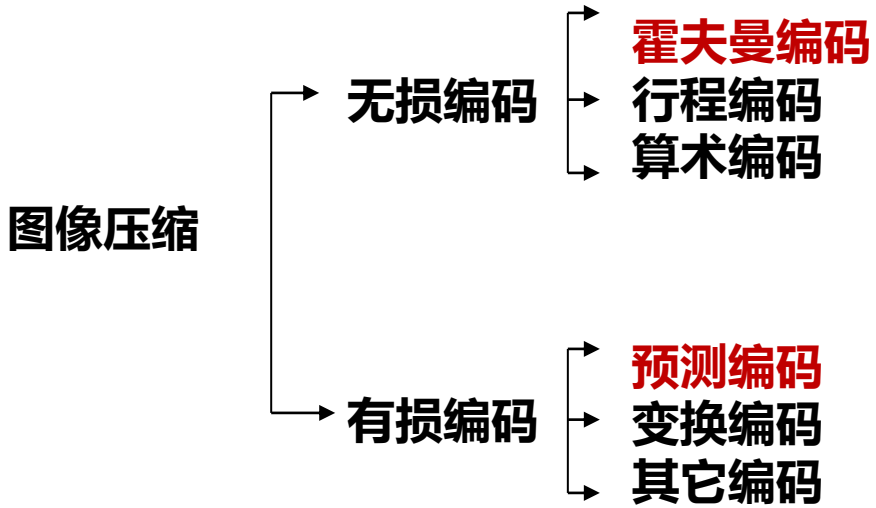
- 图像f(x,y)被送入编码器，编码器从输入数据建立一组符号，并用它们描述图像。
- 如果令n1和n2分别表示原始及编码后的图像携带的信息单元的数量(通常是比特)，达到的压缩可以通过压缩比，用数字进行量化：

$$C_R = \frac{n_1}{n_2}$$

```
cr = imratio(f1, f2)
```

图像压缩编码的分类

- 根据解压重建后的图像和原始图像之间是否具有误差，图像编码压缩分为无误差（亦称无失真、无损、信息保持）编码和有误差（有失真或有损）编码两大类。
- 无误差图像压缩的最简单方法是**减少编码冗余**。一般编码冗余通常存在于**灰度的二进制编码**中。
- 根据编码作用域划分，图像编码为空间域编码和变换域编码两大类。



无损压缩

- 无损压缩的必要性
  - 在医疗或商业文件的归档，有损压缩因为法律原因而被禁止。
  - 卫星成像的收集，考虑数据使用和所花费用，不希望有任何数据损失
  - X光拍片，信息的丢失会导致诊断的正确性
  - ……
- 无损压缩技术
  - 减少像素间冗余
  - 减少编码冗余

无损编码压缩

■ 霍夫曼编码

- Huffman于1952年提出一种编码方法，该方法完全依据字符出现概率来构造异字头的平均长度最短的码字，有时称之为最佳编码，一般就叫做Huffman编码（有时也称为霍夫曼编码）。
- **通过减少编码冗余来达到压缩的目的。**
- 基本原理：变长编码，即**将在图像中出现次数多的像素值给一个短的编码，将出现次数少的像数值给一个长的编码。**

霍夫曼编码步骤

信号源 $a=\{a_1, a_2, a_3, a_4, a_5, a_6\}$，其概率分布为$p_1=0.1$ $p_2=0.4$ $p_3=0.06$ $p_4=0.1$ $p_5=0.04$ $p_6=0.3$，求最佳Huffman码。

i. 将信源符号按出现概率从大到小排成一列，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。

建立概率统计表和编码树

符号	概率	1	2	3	4
a2	0.4	0.4	0.4	0.4	0.6
a6	0.3	0.3	0.3	0.3	0.4
a1	0.1	0.1	0.2	0.3	
a4	0.1	0.1	0.1		
a3	0.06	0.1			
a5	0.04				

霍夫曼编码举例

ii. 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元0，对概率小的赋予码元1。

编码过程：

符号	概率	编码	1	2	3	4
a2	0.4	1	0.4 1	0.4 1	0.4 1	0.6 0
a6	0.3	00	0.3 00	0.3 00	0.3 00	0.4 1
a1	0.1	011	0.1 011	0.2 010	0.3 01	
a4	0.1	0100	0.1 0100	0.1 011		
a3	0.06	01010	0.1 0101			
a5	0.04	01011				

霍夫曼编码举例

信号源 $a=\{a_1, a_2, a_3, a_4, a_5, a_6\}$，其概率分布为$p_1=0.1$ $p_2=0.4$ $p_3=0.06$ $p_4=0.1$ $p_5=0.04$ $p_6=0.3$，求最佳Huffman码。

a2	a6	a1	a4	a3	a5
1	00	011	0100	01010	01011

编码的平均长度:

$$L_{avg} = 0.4 \times 1 + 0.3 \times 2 + 0.1 \times 3 + 0.1 \times 4 + 0.06 \times 5 + 0.04 \times 5 = 2.2\ bits/symbol$$

信源熵为：2.14比特/符号(熵表示每个像素的平均信息量为多少比特，是编码所需比特数的下限)

$$H = -\sum_{K=1}^{M} P_K \log_2 P_K = -(0.4 \times \log_2 0.4 + 0.3 \times \log_2 0.3 + 2 \times 0.1 \times \log 0.1 + 0.06 \times \log_2 0.06 + 0.04 \times \log_2 0.04) = 2.14$$

编码效率为：0.973

$$\eta = H/L = 2.14/2.2 = 97.3\%$$

霍夫曼解码举例

例子：将0101001111100解码

解码通过查询表的方式完成

a2	a6	a1	a4	a3	a5
1	00	011	0100	01010	01011

解码过程：

01010	011	1	1	00
a3	a1	a2	a2	a6

霍夫曼编码步骤

霍夫曼编码具体步骤：

1.统计像素出现的概率——得到由大到小排列的像素概率表。

2.构建霍夫曼树——a.从2个概率最小的开始做父节点，合并成一个接节点，从而使节点的消息数减少一个；b.循环操作a，最终做到根节点1的位置结束。

3.对图像进行编码——从父节点开始到根节点结束，排序后进行逆序，即为编码，并建编码模式表。

随堂作业

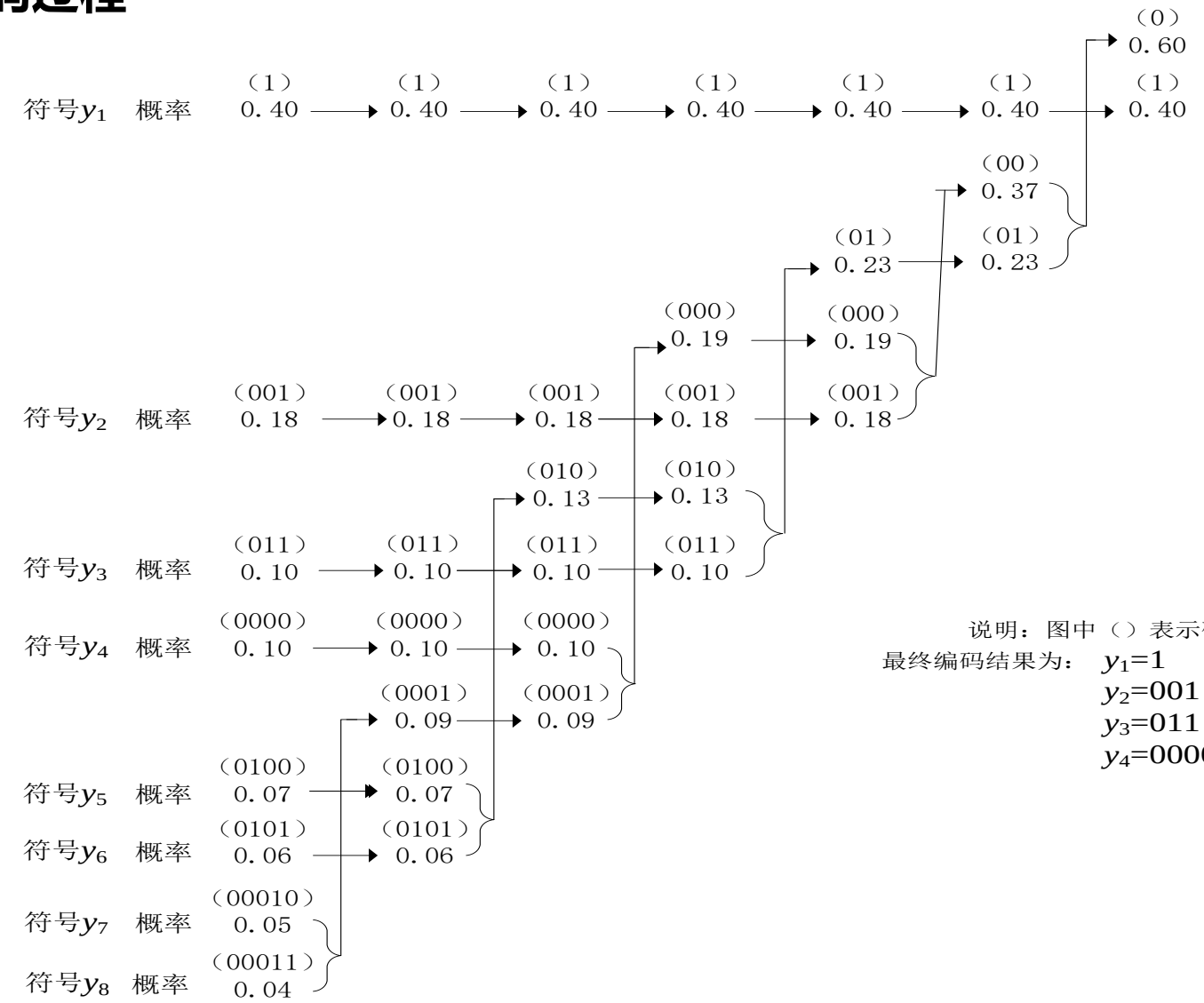
设输入图像每像元用3bit表示其灰度值，灰度级{*y1,y2,y3,y4,y5,y6,y7,y8*}，

出现的概率分别为 0.40,0.18,0.10,0.10,0.07,0.06,0.05,0.04。

试进行**哈夫曼编码**，并绘出二叉树；并计算**信源的熵、平均码长、编码效率、压缩比、冗余度** 。

答案

哈夫曼编码过程



说明：图中（）表示码字。

最终编码结果为：

y_1=1　　y_5=0100

y_2=001　　y_6=0101

y_3=011　　y_7=00010

y_4=0000　　y_8=00011

答案

按照上述的编码过程和例题所给出的参数，图像信源熵为:

$$H=-\sum_{K=1}^{M}P_K\log_2 P_K=-(0.4\times\log_2 0.4+0.18\times\log_2 0.18+2\times 0.1\log 0.1+$$
$$0.07\times\log_2 0.07+0.06\times\log_2 0.06+0.05\times\log_2 0.05+0.04\times\log_2 0.04)=2.55$$

根据哈夫曼编码过程图所给出的结果,可以求出它的平均码字长度:

$$L=\sum_{K=1}^{M}l_K P_K=0.40\times 1+0.18\times 3+0.10\times 3+0.10\times 4+0.07\times 4+$$
$$0.06\times 4+0.05\times 5+0.04\times 5=2.61$$

编码效率: $\eta=H/L=2.55/2.61=97.8\%$

压缩之前8个符号需3个比特量化，经压缩之后的平均码字长度为2.61，因此**压缩比**为:

$$C=3/2.61=1.15$$

冗余度为: $\gamma=1-\eta=2.2\%$

霍夫曼编码总结

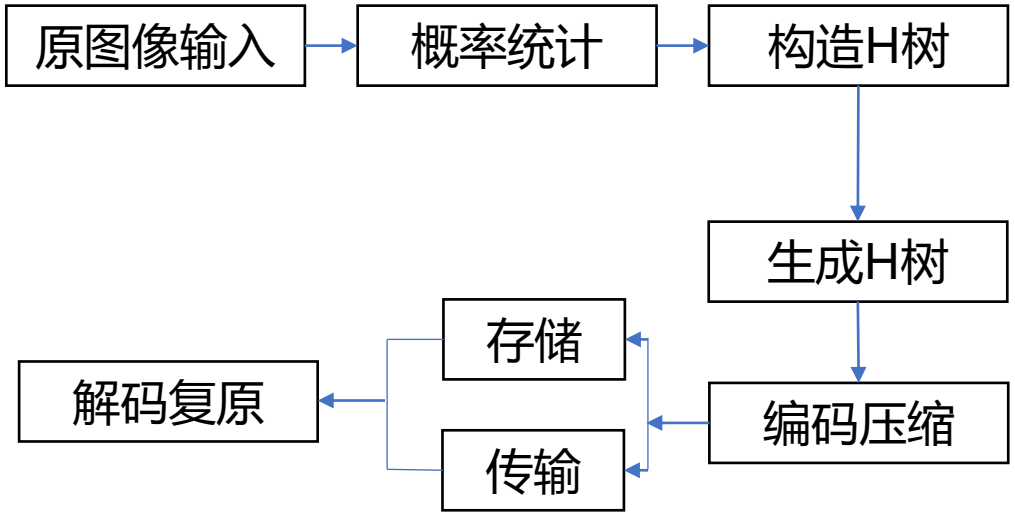
霍夫曼编码是一种瞬时唯一的可解块编码，解码时可简单地用查表方式实现。

该码有3个特点：

（1）Huffman方法构造出来的码不是惟一的。

（2）Huffman编码对不同的信源其编码效率是不同的。

（3） Huffman编码中，没有一个码字是另一个码字的前缀，因此，每个码字惟一可译。



基本霍夫曼编码示意图

Matlab

```
p = [0.1875 0.5 0.125 0.1875];
c = huffman(p)
```

```
c =

  4×1 cell 数组

    '011'
    '1'
    '010'
    '00'
```

```
f2 = uint8([2 3 4 2;3 2 4 4;2 2 1 2;1 1 2 2])
whos('f2')
```

```
4×4 uint8 矩阵

  2   3   4   2
  3   2   4   4
  2   2   1   2
  1   1   2   2
```

```
>> c = huffman (hist (double (f2 (:)), 4))

c =

  4×1 cell 数组

    {'011'}
    {'1'  }
    {'010'}
    {'00' }
```


输出是长度可变的字符数组，其中的每一行是由0和 1(对应索引符号 p中的二进制码)组成的字符串。例如， ‘010’ 是概率为0.125的灰度级的码字。

对于任何一幅图像，只要其概率是p = [0.1875 0.5 0.125 0.1875]，计算所得到的码字是一样的。

行程编码

基本原理：将一行中颜色值相同的相邻像素用一个计数值和该颜色值来代替。

举例说明：

aaaa bbb cc d eeeee fffffff (共22*8=176 bits)
→ 4a 3b 2c 1d 5e 7f (共12*8=96 bits)

这种方法在处理包含大量重复信息时可以获得很好的压缩效率。

一维行程编码只考虑了消除行内像素间的相关性。没有考虑其它方向的相关性。
二维行程编码就是利用图像二维信息的强相关性，按照一定的扫描路径遍历所有的像素形成一维的序列，然后对序列进行一维行程编码的方法。

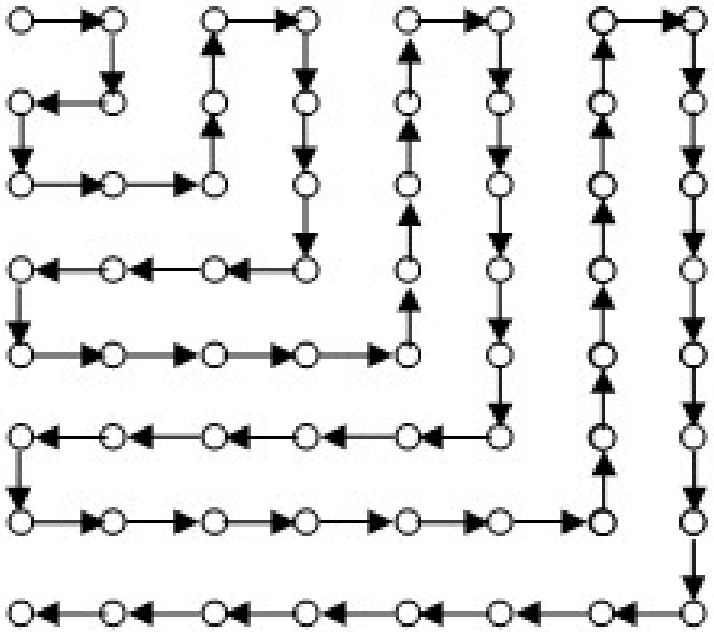


图 (a)是对图像按8×8划分子块的常用排列方式;

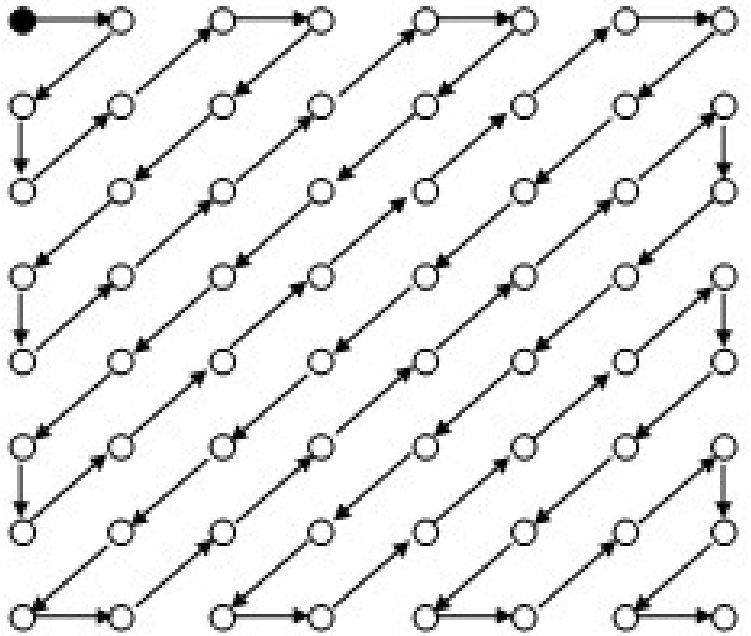


图 (b)是变换编码方式中采用的Zig-Zag排列方式，如JPEG图像编码就采用这种排列方式。

预测编码

预测编码（Predictive Coding），就是根据“过去”的时刻的像素值，运用一种模型，预测当前的像素值。预测编码通常不直接对信号编码，而是对预测误差进行编码。

原理：对图象的一个像素的离散幅度的真实值，利用其相邻象素的相关性，预测它的下一个象素的可能值，再求两者差，对差值量化、编码，达到压缩的目的。

例：原图像数据：234　223　231　238　235

　　压缩后数据：234　-11　8　7　-3

预测编码

像素的新信息被定义为**实际值和预测值的差值**。

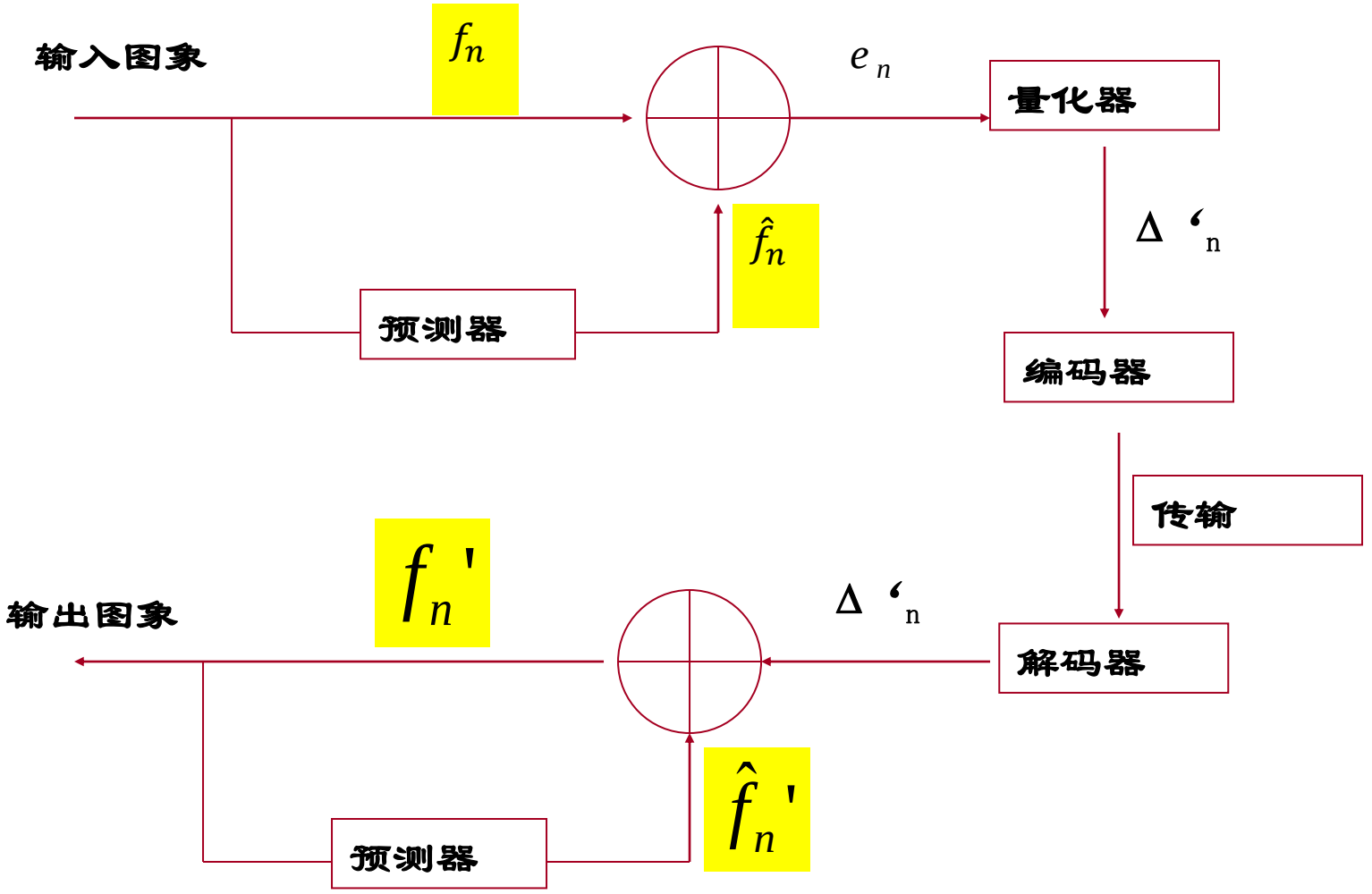
系统由编码器和解码器组成，每个都含有相同的预测器。

预测器：以过去某些输入为基础产生像素的预测值。预测器的输出被四舍五入成最接近的整数。表示为$\hat{f}_n$。m个先前的像素进行线性组合以得到预测：

$$\hat{f}_n = round\left[\sum_{i=1}^{m} a_i f_{n-i}\right]$$

最简单的一维线性预测编码是一阶（m=1）

$$\hat{f}_n = round\left[af(x, y-1)\right]$$



预测编码示意图

预测编码

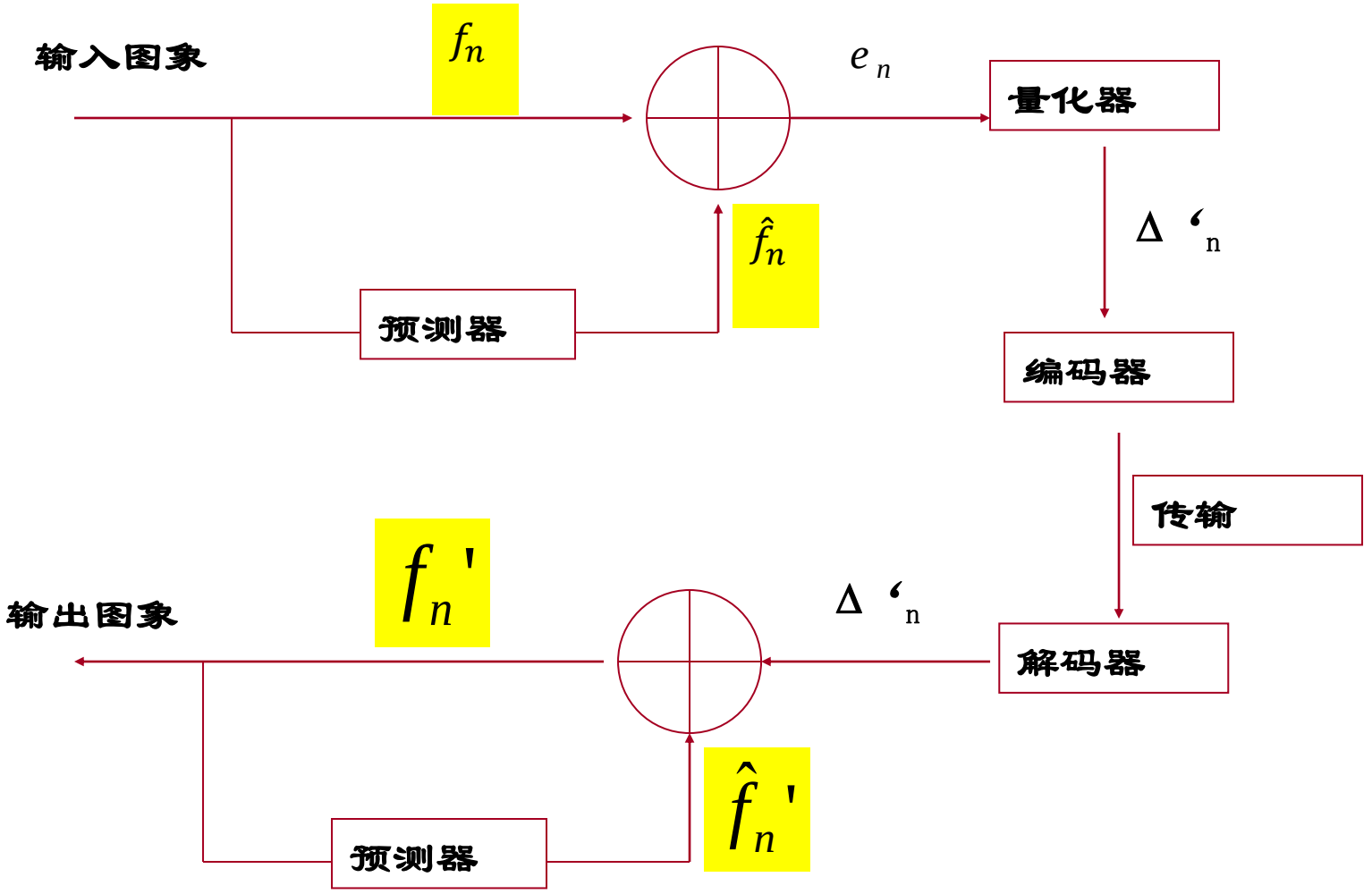
产生差或预测误差：

$$e_n = f_n - \hat{f}_n$$

该误差用符号编码器借助变长码进行编码以产生压缩数据流。

解码器：编码器的反操作，将压缩过的图像通过符号解码器进行解码，并重建预测误差。

$$f_n = e_n + \hat{f}_n$$



预测编码示意图

预测编码

编码过程：

1. 压缩头处理
2. 对每一个符号：f(x,y)，由前面的值，通过预测器，求出预测值f^(x,y)
3. 求出预测误差：e(x,y)=f(x,y)-f^(x,y)
4. 对误差e(x,y)编码，作为压缩值。
5. 重复二，三，四步。

解码过程：

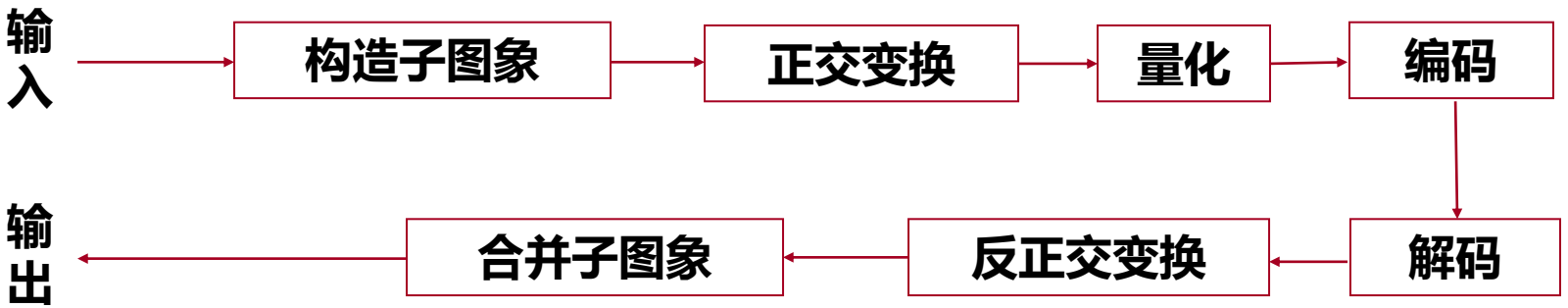
1. 对头解压缩
2. 对每一个预测误差的编码解码，得到预测误差e(x,y)。
3. 由前面的值，得到预测值f^(x,y)。
4. 误差e(x,y),与预测值f^(x,y)相加，得到解码f(x,y)。
5. 重复二，三，四步。

变换编码

- 空间域：对一幅图像像素进行直接操作
- 变换域：以修改图像的变换为基础
- 变换编码：一个可逆的线性变换或离散余弦变换（DCT），用于将一幅图像映射成一组变换系数，然后对系数进行量化和编码。
- 对大多数自然图像来说，多数系数具有较小的数组，可以粗糙地量化（或完全抛弃）而对图像造成的失真较小。

变换编码

原理：图象数据经过正交变换后，其变换系数具有一定的相互独立性，（例如，对于FT来说，频普系数大的变换系数均集中在低频部分，而高频部分的幅值均很小，因而可以对低频的变换系数量化、编码和传输，对高频部分不处理，这样可以达到图象压缩的目的。

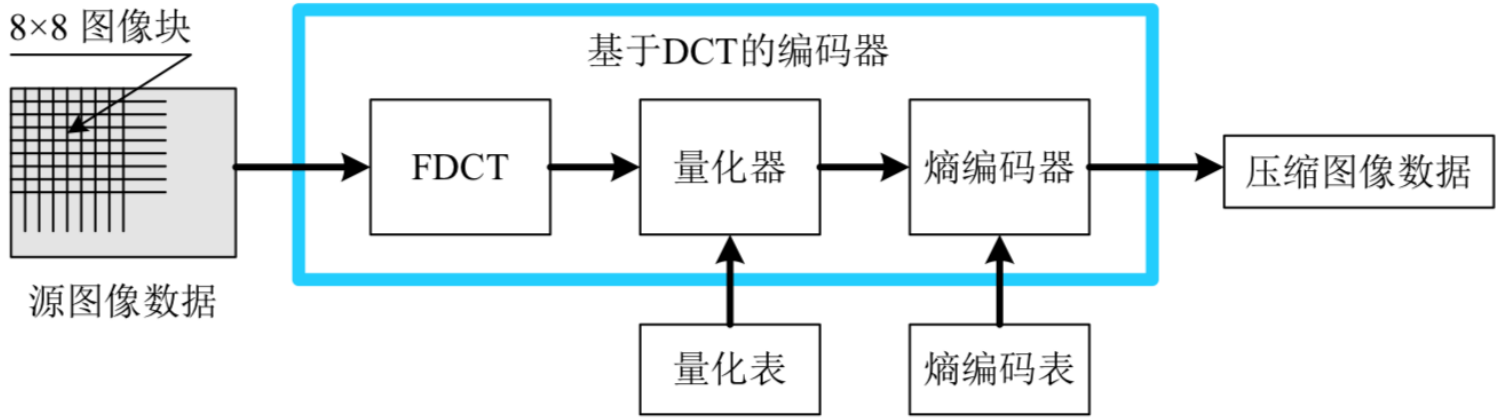


变换编码的一般系统框图

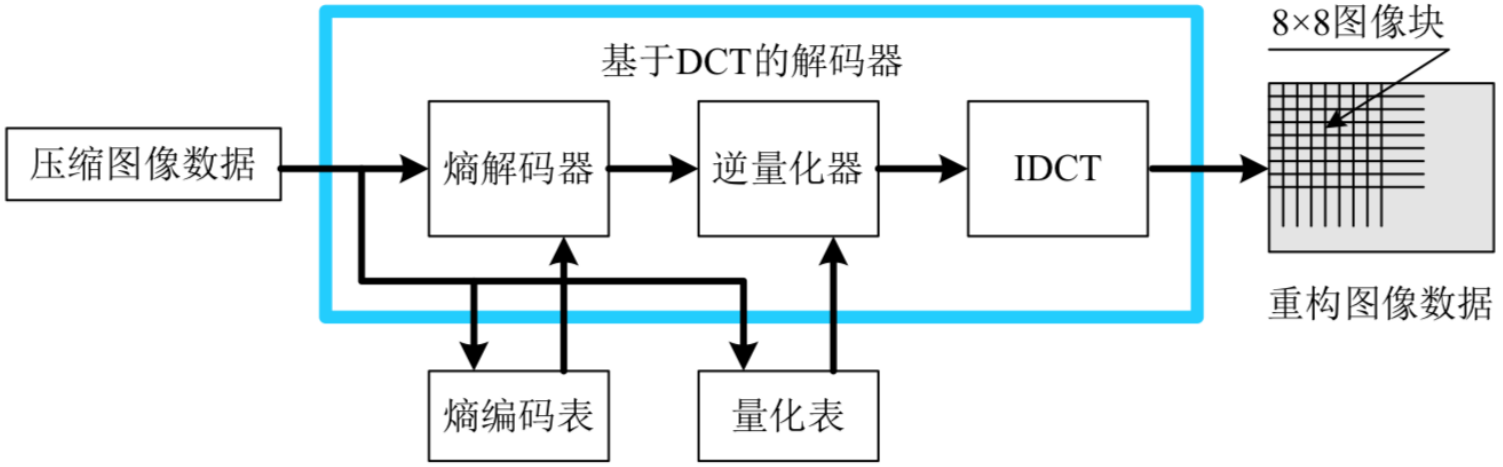
JPEG压缩

- 国际标准化组织（ID）和国际电报电话咨询委员会（CCITT）联合成立的专家组JPEG（Joint Photographic Experts Group）于1991年3月提出了ISO CDIO918号建议草案：多灰度静止图像的数字压缩编码（通常简称为JPEG标准）。这是一个适用于彩色和单色多灰度或连续色调静止数字图像的压缩标准。
- 定义了3种编码系统
  - 基于DCT的有损编码基本系统，可用于绝大多数压缩应用场合
  - 用于高压缩比、高精确度或渐进重建应用的扩展编码系统
  - 用于无失真应用场合的无损系统

JPEG基本系统



(a) JPEG压缩算法框图



(b) JPEG解压缩算法框图

JPEG基本系统编码器框图

■ JPEG压缩编码算法的主要计算步骤如下：

■先把整个图像分解成多个8×8的图像块；图像被分割成大小为8X8的小块，这些小块在整个压缩过程中都是单独被处理的。

■**DCT**：8×8的图像块经过DCT变换后，低频分量都集中在左上角，高频分量则分布在右下角（DCT变换类似于低通滤波器），因为低频分量包含了图像的主要信息，所以可以**忽略高频分量**，达到压缩的目的；

■**量化**：使用量化操作去掉高频分量。量化操作就是将某一个值除以量化表中的对应值。由于量化表中左上角的值较小，而右下角的值较大，这样达到保持低频分量，抑制高频分量的目的；

■**DC-差分编码**：在左上角的低频分量中，F(0,0)代表了直流(DC)系数，即8×8 子块的平均值。由于两个相邻图像块的DC系数相差很小，所以采用差分编码DPCM。

■**AC-行程编码**：其它63个元素是交流（AC）系数，经过量化后，AC分量出现较多的0。JPEG采用对0系数的行程长度编码。而对非0值，则要保存所需位数和实际值采用之字型（zig-zag）顺序进行行程编码，使系数为0的值更集中。

■**霍夫曼编码**：在得到DC码字和AC行程码字后，为了进一步提高压缩比，采用了Huffman编码

■JPEG标准不规定FDCT和IDCT的算法。

应用举例

Lenna图像的一个8*8方块

$$f(x,y)=\begin{bmatrix}139 & 144 & 149 & 153 & 155 & 155 & 155 & 155\\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156\\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156\\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159\\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155\\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157\\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157\\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158\end{bmatrix}$$

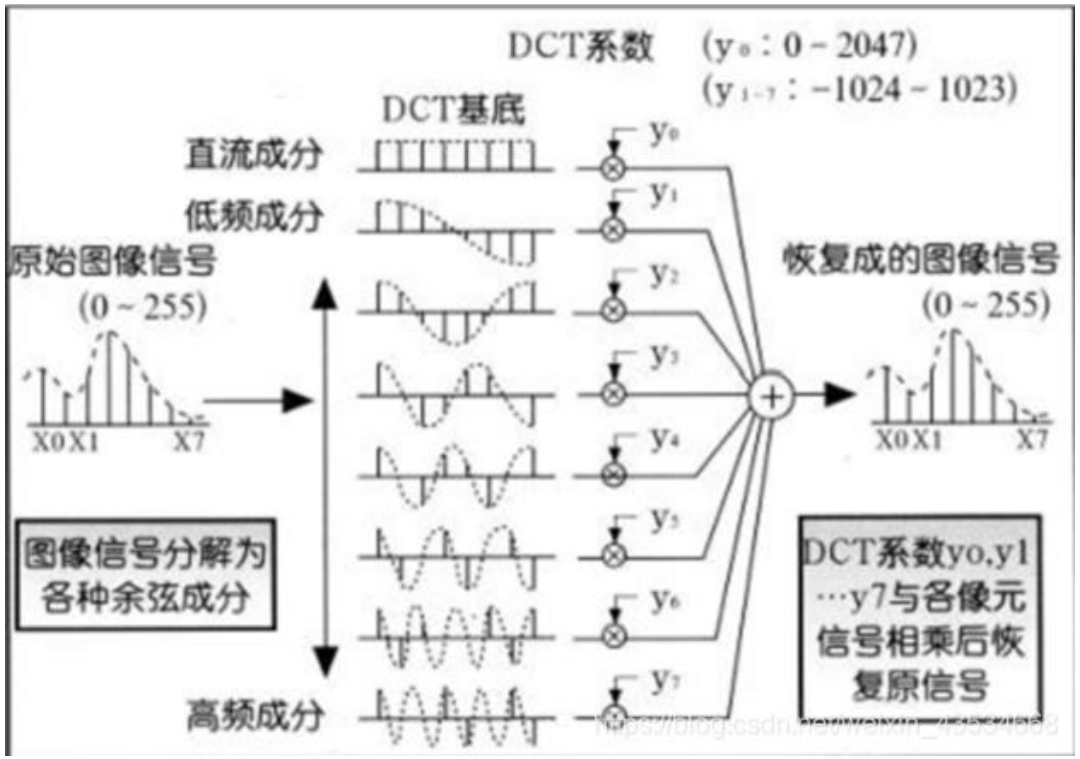
应用举例

原始的图像信号经过DCT变换后变成了8个波，其中第一个波为直流成分，其余7个为交流成分。

- F(0,0)：直流(DC)系数
- 其它63个元素：交流（AC）系数
- 二维DCT将图像的能量集中在极少的几个系数之上，这些系数大都集中在左上角，即低频分量区。

$$F(u,v)=\begin{bmatrix} 1260 & -1 & -12 & -5 & 2 & -2 & -3 & 1 \\ -23 & -17 & -6 & -3 & -3 & 0 & 0 & -1 \\ -11 & -9 & -2 & 2 & 0 & -1 & -1 & 0 \\ -7 & -2 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1 & 1 & 2 & 0 & -1 & 1 & 1 \\ 2 & 0 & 2 & 0 & -1 & 1 & 1 & -1 \\ -1 & 0 & 0 & -1 & 0 & 2 & 1 & -1 \\ -3 & 2 & -4 & -2 & 2 & 1 & -1 & 0 \end{bmatrix}$$

经过FDCT后的变换系数矩阵



应用举例

将频率系数除以量化矩阵的值之后取整，即完成了量化过程。

$$B_{i,j} = round(\frac{G_{i,j}}{Q_{i,j}})j, j = 0, 1, 2, \ldots, 7$$

G是我们需要处理的图像矩阵，Q称作量化系数矩阵，round函数是取整函数。

以这个结果而言，经常会把很多更高频率的成分舍位成为接近0，且剩下很多会变成小的正或负数。

$$F(u,v) = \begin{bmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

根据量化表量化后得到的量化系数矩阵

应用举例

- 8 * 8的图像块经过DCT变换之后的DC系数有两个特点：
（1）系数的数值比较大；
（2）相邻的88图像块的DC系数值变化不大；

- 根据这两个特点，DC系数一般采用差分脉冲调制编码DPCM（Difference Pulse Code Modulation），即：取同一个图像分量中每个DC值与前一个DC值的差值来进行编码。

- DPCM所需的位数会比对原值进行编码所需位数少很多。
  - 假设某一个88图像块的DC系数值为15，而上一个88图像块的DC系数为12，则两者之间的差值为3。如第一个块的DC系数为14，第二个块的DC系数为16，则记录第一块的DC系数为14，第二个块则只记录差值2，需要2bit;如果不采用DPCM则需要5bit。

DC系数

$$F(u,v)=\begin{bmatrix} 79 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

根据量化表量化后得到的量化系数矩阵

应用举例

- **Zig-zag扫描和行程编码(Run-Length Encoding, RLE)**

- DC不参与Z字形扫描，AC分量则采用Z字形扫描排列并进行行程编码(RLE)。

- 量化AC系数的特点是包含很多连续0，故使用RLE对其进行编码。JPEG使用了1个字节的高4位来表示连续的0的个数，而使用其低4位来表示下一个非0系数需要的位数，紧随其后的是量化AC系数的值。

假设Zig-zag扫描后的一组向量的AC分量为

$$31, 45, 0, 0, 0, 0, 23, 0, -30, -8, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, \ldots, 0$$

经RLE压缩后如下

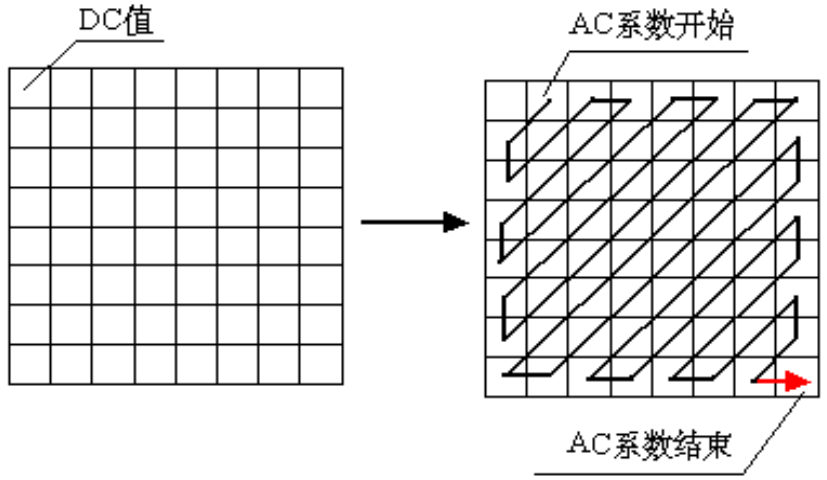
$$(0, 31); (0, 45); (4, 23); (1, -30); (0, -8); (2, 1); EOB$$

其中EOB表示后面都是0。实际上，用(0,0)表示EOB。
若这组数字不以0结束，则不需要EOB。

按如下图中的顺序依次保存和读取64个DCT的系数值，可以使0尽量连在一起，进一步压缩存储空间。

0	1	5	6	14	15	27	28
2	4	7	13	16	26	29	42
3	8	12	17	25	30	41	43
9	11	18	24	31	40	44	53
10	19	23	32	39	45	52	54
20	22	33	38	46	51	55	60
21	34	37	47	50	56	59	61
35	36	48	49	57	58	62	63

量化DCT系数的序号



Matlab

```
>> f = imread('Fig0804(a).tif');
>> c1 = im2jpeg(f);
>> f1= jpeg2im(c1);
>> imratio(f, c1)
ans =
   18.4116
>> compare(f, f1, 3)
ans =
    2.8659
>> c4 = im2jpeg(f, 4);
>> f4 = jpeg2im(c4);
>> imratio(f, c4)
ans =
   43.3153
>> compare(f, f4, 3)
ans =
    4.6657
```

本章小结

- ✓编码冗余
- ✓空间冗余
- ✓不相关信息
- ✓JPEG压缩