图像压缩

陈小潘
计算机与信息工程学院

目录

图像压缩编码的必要性

- 大数据量的图像信息会给存储器的存储、网络的传输以及计算机的处理增加极大的压力。
- 单纯靠增加存储器容量，提高网络带宽以及计算机的处理速度等方法来解决这个问题是不现实
- 因此，图像数据在传输和存储中，数据的压缩都是必不可少的。

概述

■ 图像为存储、传输带来的挑战举例

例子1：

电话线传输速率一般为56kbit/s（波特率）

一幅彩色图像640×480×24bit = 7Mbit大小

✓ 传输一幅图像：时间约2分钟左右；

✓ 实时传送： 640×480×24bit×25帧/s=175Mbit/s，时间为50min左右；

✓ 如压缩20倍，传一幅图6s左右，可以接受，实用

概述

- **图像为**存储、传输带来的挑战举例

举例2：

- 一幅1920×1080像素RGBA图像，32bit/像素，1920×1080×4= 8294400 Byte≈8MB
- 如果视频是1920×1080，30fps，1小时。不压缩需要的内存：

8MB×30×60×60 = 864000M=864GB

图像压缩编码的必要性

- **数据压缩的研究内容：**数据的表示、传输、变换和编码方法，目的是减少存储数据所需的空间和传输所用的时间。
- **图像编码与压缩：**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。

图像压缩编码的可能性

数字图像本身的特征为数据压缩提供可能性

- 图像数据是高度相关的，或者说存在冗余信息，去掉这些冗余信息后可以有效压缩图像，同时又不会损害图像的有效信息。
- 图像能量在变换域内分布的不均匀性

例如：大部分能量集中在低频部分，而小部分能量集中在高和较高的频率部分。

概述

- 图像编码: 对图像信息进行压缩和编码，在存储、处理和传输前进行，也称图像压缩；
- 图像解码：对压缩图像进行解压以重建原图像或其近似图像。
- **数据和信息：**数据是信息的载体，对给定量的信息可用不同的数据量来表示。
- **对给定量的信息，设法减少表达这些信息所需的数据量称为数据压缩。**

数据冗余

- 冗余：
  - ✓ 数据表达了无用的信息。
  - ✓ 数据表达了已表达的信息。
  - ✓ 相对不重要的信息。
- 压缩数据量的重要方法是消除冗余数据。如果能减少或消除其中的一种或多种冗余，就能达到数据压缩的目的。

数据冗余

■ 数字图像冗余的主要表现形式

✓ 空间冗余

✓ 时间冗余

✓ 视觉冗余

✓ 编码冗余

✓ 知识冗余

✓ **应用需求冗余**

数据冗余

- **空间冗余（像素间冗余、几何冗余）**

邻近像素灰度分布的相关性强。例如，图像数据中，大量的相邻像素完全一样或十分接近，或者图像由有规律的图案组成。

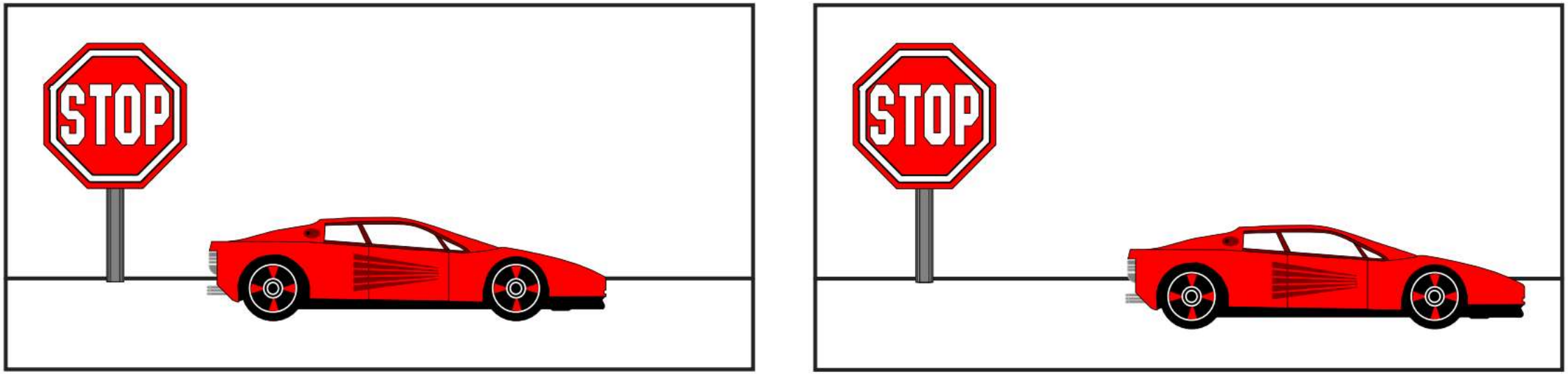
数据冗余

- **时间冗余。**

序列图像（电视图像、运动图像）帧间画面对应像素灰度相关性强。图像序列中一组连续的画面之间往往存在着时间和空间的相关性，相邻两帧的大部分数据无变化，只有少量数据变化。

数据冗余

- 时间冗余。



(a) F1帧　　(b) F2帧

如图所示，Fl帧中有一个小汽车和一个路标，在时间T后的F2图像中仍包含以上两个物体，只是小车向前行驶了一段路程，此时F1和F2中的路标和背景都是时间相关的，小车也是时间相关的，因而F2和Fl具有时间冗余。

数据冗余

编码冗余

- ✓编码是用于表示信息主体或事件集合的符号（字母、数字、比特等）系统。
- ✓每条信息或事件被赋予一系列编码符号，称为码字。
- ✓每个码字中符号的数量就是该码字的长度，称为码长。
- ✓**编码冗余，也称为信息熵冗余：如果图像中平均每个像素使用的比特数大于该图像的信息熵，则图像中存在冗余，称为信息熵冗余。**

数据冗余

编码冗余

例如，二值图像的编码。

- ✓ 二值图像只有黑色和白色，也就是每个像素仅两个灰度级，用1比特即可表示。
- ✓ 若用大于1的比特位表示二值图像的像素，则存在编码冗余。

数据冗余

编码冗余

162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
164	164	158	156	161	160	159	160
161	161	163	158	160	162	159	156
159	159	156	157	159	159	156	157

例如，大多数图像的灰度值呈不均匀分布。

- ✓ 若等长编码，对最大和最小概率可能性的灰度值分配相同比特数，产生编码冗余。
- ✓ 可根据灰度概率分配编码长度，则有可能使编码总长度下降。

数据冗余

编码冗余

✓ 如果用[0,1]内的一个随机变量r_k表示图像的灰度级，则每个灰度级r_k出现的概率为：

$$p_r(r_k) = \frac{n_k}{n} \quad k = 1,2,3 \cdots, L$$

式中：n_k是图像中出现第k级灰度的次数，n是图像中像素总数。

✓ 如果用于表示每个 r_k 的比特数为 $\boldsymbol{l_k}$ ，则表示每个像素所需的平均比特数：

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

编码一幅$M \times N$的图像需要的总比特数：$M \times N \times L_{\text{avg}}$

数据冗余

编码冗余

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

$$L_{avg} = \sum_{k=0}^{7} l_2(r_k) p_r(r_k) = 2 \times 0.19 + 2 \times 0.25 + \cdots + \cdots + 6 \times 0.03 + 6 \times 0.02 = 2.7$$

压缩比 $C_R = 3/2.7 = 1.11$　　相对数据冗余 $R_D = 1 - \frac{1}{1.11} = 0.099$

编码时如果不能使L_{avg}达到最小，就存在编码冗余。

数据冗余

- 知识冗余

图像的理解与某些基础知识有相当大的相关性。例如，人脸的图像有固定的结构，比如说嘴的上方有鼻子，鼻子的上方有眼睛，鼻子位于正脸图像的中线上等等，这类规律性的结构可由先验知识和背景知识得到，称此类冗余为知识冗余。

数据冗余

心理视觉冗余

心理视觉冗余：被人类视觉系统忽略的数据所导致的冗余（即视觉上不重要的信息）。人的眼睛并不是对所有信息都有相同的敏感度，有些信息在通常的视觉感觉过程中与另外一些信息相比并不那么重要，这些信息可认为是心理视觉冗余。

数据冗余

心理视觉冗余

- ✓ 去除心理视觉冗余并不会明显地降低所感受到的图像质量或所期望的图像作用。
- ✓ 例如：人的视觉系统一般的分辨能力约为2^6灰度等级，而图像量化一般采用2^8灰度等级。
- ✓ 心理视觉冗余与人观察图像的方式有关，因人而异，因应用要求而异。例如，教师上课给学生演示用的CT影像和医生看病诊断用的CT影像可以使用不同分辨率大小的图像。

数据冗余

心理视觉冗余

根据人的视觉特性对不敏感区进行降分辨率编码。

- ✓ 人眼对静态物体的敏感程度大于对动态物体敏感程度，可减少表示动态物体bit数；
- ✓ 人眼对图像中心信息敏感程度大于对图像边缘信息敏感程度，可对边缘信息少分配bit数。
- ✓ 人眼对亮度比对色度更敏感，对色度信息进行下采样，去掉冗余信息。
- ✓ 人眼对高频信号不敏感，可分配较少bit。

数据冗余

应用需求冗余

应用环境允许图像有一定程度地失真。

- ✓ 接收端设备分辨率降低，则可降低图像分辨率。
- ✓ 应用方关心图像区域有限，可对其余部分图像可采用空间和灰度级上的粗化。
- ✓ 对于识别，图像特征抽取和描述也是数据压缩。

数据冗余的量化

- **绝对数据冗余**：数据量R与信息量r之差。

$$D = R - r$$

- **相对数据冗余**：绝对数据冗余D与数据量R之比。

$$R_D = D/R = 1 - r/R$$

- **压缩比**：数据量R与信息量r之比，取值范围：$(0, \infty)$。

$$C_R = R/r$$

MATLAB函数：cr = imratio(f1, f2)

- 相对数据冗余：

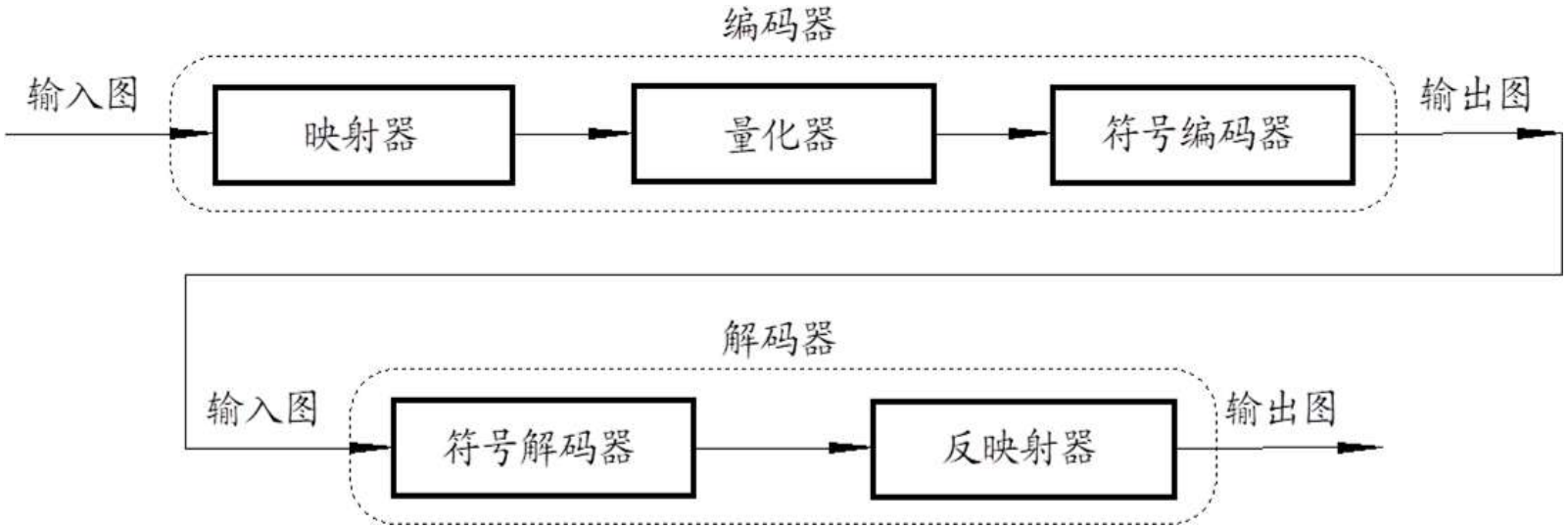
$R_D = 1 - 1/C_R$，取值范围：$(-\infty, 1)$

数据冗余的量化

n_1相对于n_2	C_R	R_D	对应的情况
$n_1 = n_2$	1	0	第 1 种表达相对第 2 种表达不含冗余数据
$n_1 \gg n_2$	$\to \infty$	1	第 1 个数据集合含相当多的冗余数据
$n_1 \ll n_2$	$\to 0$	$\to -\infty$	第 2 个数据集合包括比原始表达多得多的数据

n_1和n_2代表两个数据集合中的信息载体单位的个数。

图像压缩的过程



- 映射器对输入数据进行变换以减少空间/时间冗余。
- 量化器通过减少映射器输出的精度来减少心理视觉冗余。
- 符号编码器通过将最短的码赋给最频繁出现的量化器输出值以减少编码冗余。
- 解码器进行符号编码和映射的逆操作（符号解码和反映射）。量化操作是不可逆的，因此没有反量化过程。

图像压缩的过程

- 图像压缩可分为两类：
  - ✓ **无损压缩**（亦称无失真、无误差、信息保持型）：在压缩和解压缩过程中没有信息损失，这类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。无损压缩用于要求重建后图像严格地和原始图像保持相同的场合，例如复制、保存十分珍贵的历史、文物图像等。压缩比一般在2~10之间。
  - ✓ **有损压缩**（有失真、有误差、信息非保持型）：这类算法删除了不相干的信息，导致在解压缩时只能对原始图像进行近似的重建，而不能精确的复原。有损压缩适合大多数用于存储数字化了的模拟数据。压缩后并不能经解压缩完全恢复原始信息，常能取得较高的压缩比（几十~几百）。

图像压缩的过程

（1）无损压缩编码

哈夫曼（Huffman）编码，算术编码，行程（RLE）编码，Lempel zev 编码。

（2）有损压缩编码

预测编码，DPCM，运动补偿

频率域方法：正交变换编码(如DCT)，子带编码；

空间域方法：统计分块编码；

模型方法：分形编码，模型基编码；

基于重要性：滤波，子采样，比特分配，向量量化；

（3）混合编码

JBIG，H261，JPEG，MPEG等技术标准。 JBIG(Joint Bi-level Image Experts Group)是一种图像压缩技术，由联合双层图像专家组织开发，是ISO/IEC标准的一部分，用于压缩黑白图像。

图像保真度

■ **图像保真度：描述解码图像相对于原始图像的偏离程度。**

主观保真度准则：用主观的方法来测量图像的质量，是观察者对图像综合评价的平均。

一种常用的方法是让一组（不少于20人）观察者观察图像并给该图像评分，将他们对该图像的评分取平均，作为这幅图像的质量。

- ✓ 损伤程度：不能察觉、刚察觉、不讨厌、有点讨厌、很讨厌，不能用；
- ✓ 质量：优、良、中、次、 劣；
- ✓ 比较： +2，好的多； +1，好； 0，相同； -1，差； -2，差的多。

图像保真度

客观保真度准则：用编码输入图与解码输出图的某个确定性函数（准则）来表示图像编解码所损失的信息量。

- 点误差：$e(x,y)=\hat{f}(x,y)-f(x,y)$
- 输入图和输出图间总误差：

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

图像保真度

- 均方根误差：

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

- 归一化信噪比 **SNR**（单位：分贝 **dB**）：

$$SNR = 10 \lg \left[\frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[f(x,y) - \bar{f} \right]^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left[\hat{f}(x,y) - f(x,y) \right]^2} \right]$$

图像保真度

- **峰值信噪比 PSNR：**

$$PSNR = 10\lg\left[MN \times f_{\max}^2 \Big/ \sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left[\hat{f}(x,y) - f(x,y)\right]^2\right]$$

目录

信息量

- 从信息论的角度来看，数据压缩就是去掉数据中的冗余，即保留不确定的信息，去掉确定的信息（可推知的）。
- 数据是用来记录和传送信息的，或者说数据是信息的载体。真正有用的不是数据本身，而是数据所携带的信息。
- 信息由不确定性程度进行度量。
- 确定事件的信息量为零，不确定性程度越高信息量越大。

信息量

- 离散信源可以分为无记忆信源和有记忆信源两种，无记忆信源的当前输出与以前的输出无关。
- 假设无记忆信源符号集：$A = \{a_1, a_2, \ldots, a_N\}$，其概率分布：$\{P(a_1), P(a_2), \ldots, P(a_N)\}$，且满足

$$\sum_{k=1}^{N} P(a_k) = 1$$

- 信源符号a_k的自信息（或信息量）定义为：

$$I(a_k) = log[1/P(a_k)] = -logP(a_k)$$

- 信息量的单位由式中所用对数的底数来确定，对数以2为底时信息量的单位为比特，以e为底时单位是奈特（nit）。

信息熵

- 当两个相等可能性的事件之一发生时，其信息量就是1比特（bit）。
- 当$P(a_k)=1$（即事件总发生）时，$\boldsymbol{I(a_k)=0}$
- 一个概率小的符号出现将带来更大的信息量。
- 将信源*A*所有符号的信息量进行平均就得到信息熵（entropy）：

$$H(A)=-\sum_{k=1}^{J}P(a_k)\log P(a_k)$$

- 信息熵定义了观察到单个信源符号输出时所获得的平均信息量，表示不确定性，单位：比特/符号
- 如果信源各符号的出现概率相等，则熵达到最大。

信息熵

设 $X = \{a, b, c, d\}$，$p(a) = p(b) = p(c) = p(d) = 1/4$

则各信源符号自信息量：

$I(a) = I(b) = I(c) = I(d) = \log_2 4 = 2$

信源熵：$H(X) = \frac{1}{4} \times 2 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2 + \frac{1}{4} \times 2 = 2$

编码方法：*a*，*b*，*c*，*d* 用码字00、01、10、11来编码，每个符号用 2 个比特，平均码长也是 2 比特。

平均码长大于信源的熵

信息熵

设 $X = \{a, b, c, d\}$，

$p(a) = 1/2, p(b) = 1/4, p(c) = p(d) = 1/8$

则各信源符号自信息量：

$I(a) = \log_2 2 = 1, I(b) = \log_2 4 = 2，\ I(c) = I(d) = \log_2 8 = 3$

信源熵：$H(X) = \frac{1}{2} \times 1 + \frac{1}{4} \times 2 + \frac{1}{8} \times 3 + \frac{1}{8} \times 3 = 1.75$

平均码长大于信源的熵

- 信源的平均码长$l_{avg} >= H(X)$，也就是说熵是无失真编码的下界。
- 对非等概率分布的信源，采用不等长编码其平均码长小于等长编码的平均码长。

编码定理

图像压缩涉及到的相关编码定理

- 信息保持编码定理：信源熵是进行无失真编码的理论极限，低于此极限的无失真编码方法是不存在的，这是熵编码（无失真编码）的理论基础。
- 变长码信源编码定理（香农第一定理）：采用无失真最佳信源编码可使得用于每个信源符号的编码位数尽可能地小，但它的极限是原始信源的熵值。超过了这一极限就不可能实现无失真的译码。
- 变长最佳编码定理：在变长编码中对出现概率大的信息赋予短码字，而对于出现概率小的信息赋予长码字。

目录

熵编码

信息论是图像编码的主要理论依据之一，它给出无失真编码所需比特数的下限，为逼近这些下限提出了一系列熵编码算法。

熵编码（Entropy encoding）：利用数据的统计信息进行压缩的无损编码。

常见的熵编码：香农-法诺编码（Shannon-Fano coding）、哈夫曼编码（Huffman coding）和算术编码（arithmetic coding）。

哈夫曼编码

- Huffman于1952年提出的一种编码方法，目的是对每个信源符号产生最少的编码符号。
- 所谓最佳编码方法是指采用Huffman编码方法得到的单元像素的比特数最接近图像的实际熵值。
- Huffman编码是根据可变长最佳编码定理，应用哈夫曼算法而产生的一种编码方法。
- Huffman编码是一种无损压缩编码（编码后的图像经过译码后完全恢复为原图像的压缩编码，在编码系统中无失真编码也称为熵编码）

哈夫曼编码

Huffman编码步骤

（1）将信源符号按概率由大到小排列，概率相同的可以任意排列。

（2）两个最小概率相加，形成新的概率集合，并按（1）的原则重新排列。

（3）重复（2）的过程，直到仅剩下两个概率为止。

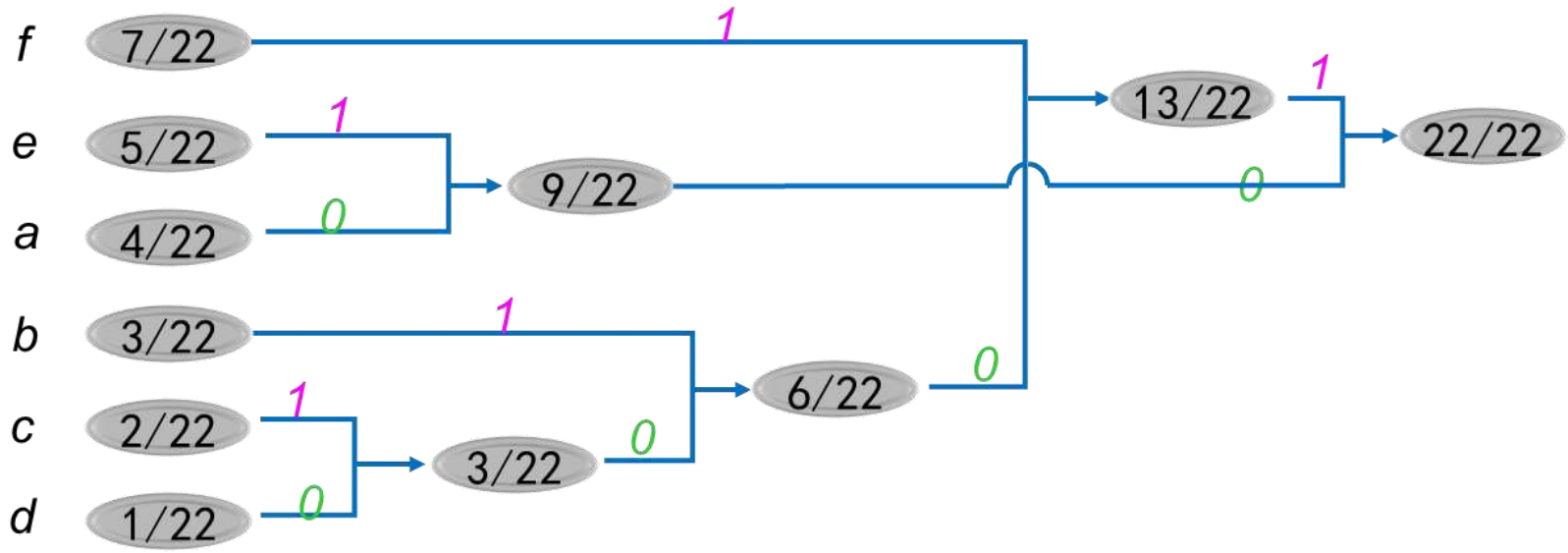
（4）分配码字进行编码，原则是从后到前，上 0下 1（或上1下0）。也即大的赋0，小的赋1（或者相反）

（5）从最后一次概率相加到第一次参与相加，依次读取所赋码元，构造哈夫曼码字，编码结束。

哈夫曼编码

例：22 位数据序列aaaa bbb cc d eeeee ffffff

概率分布为： a: 4/22， b: 3/22，c: 2/22，d: 1/22，e: 5/22，f: 7/22



哈夫曼编码：f=11， e=01， a=00， b=101， c=1001， d=1000

哈夫曼编码

例：22位数据序列 aaaa bbb cc d eeeee ffffff

如果每个字符用8bits存储，需要22×8=176bits

哈夫曼编码：a=00 b=101 c=1001 d=1000 e=01 f=11

编码后的数据为：

00000000101101101100110011000010101010111111111111111

共 4×2+3×3+2×4+1×4+5×2+7×2=53 bits

压缩比为 3.32:1

平均码长：$L_{avg} = \sum_{k=1}^{6} l_k P_k = 2.4089$

熵：$H = -\sum_{k=1}^{6} P_k \log P_k = 2.3678$

哈夫曼编码

哈夫曼编码特点：

- 无歧义性，能正确地恢复原信号。
- 构造出来的码不唯一，硬件实现困难。
  - ✓ 有两种赋值方式：概率大的赋 1，小的赋 0，反之亦可。
  - ✓ 两符号概率相等时，排列顺序随机，造成编码不唯一。
- 必须先统计得到信源的概率特征才能编码。
- 对不同的信号源，编码效率不同。等概率信源，效率最低。
- 编码后形成一个哈夫曼编码表，若要正确解码必须有此码表，于是在传送图像过程中也要传送此码表。

哈夫曼编码

```
%哈夫曼编码
clc;clear,close all;
Image=imread('lena.bmp');
subplot(1,2,1),imshow(Image);title('(a)原图像
','FontSize',14);
[h,w]=size(Image);
len=255;%处理的是一个灰度图像，范围是
0~255
symbols=0:len;%获取编码符号
totalpixelnum=h*w;
p=imhist(Image)/totalpixelnum;%获取编码符号
的概率
dict=huffmandict(symbols,p);%生成编码符号表
sig=reshape(Image,1,h*w);
enco=huffmanenco(sig,dict);%对图像进行huffman编码
deco=huffmandeco(enco,dict);%huffman解码
deImage=uint8(reshape(deco,h,w));
subplot(1,2,2),imshow(deImage);title('(b)哈夫曼编码解码图像
','FontSize',14);
diff=imsubtract(Image,deImage);%原图与重构图的差
maxdiff=max(diff(:));%差的最大值
```

哈夫曼编码

(a)原图像

(b)霍夫曼编码解码图像

maxdiff=0，说明原图像和解码图像一样。

算术编码

算术编码是20世纪80年代发展起来的一种熵编码方法，其从整个符号序列出发，采用递推形式连续编码。不是将单个信源符号映射成一个码字，而是直接将整个输入的消息编码为一个满足[0, 1)的小数。

- 消息序列中的每个元素都要缩短为一个区间，消息序列中元素越多，所得到的区间就越小。
- 当区间变小时，就需要更多的数位来表示这个区间，采用算术编码，每个符号的平均编码长度可以为小数。

算术编码

- 算术编码是一种无损数据压缩方法，属于熵编码。
- 在给定符号集和符号概率的情况下，算术编码可以给出接近最优的编码结果（优于哈夫曼编码）。
- 假设某个字符的出现概率为80%，该字符事实上只需要 $-log_2(0.8) = 0.322$ 位，但Huffman编码需要1位。由此可知，整个数据的80%的信息在哈夫曼编码中用的是理想长度3倍（≈1/0.322）的码字，导致压缩效果不够理想。
- 算术编码可以解决这个问题，算法编码在图像数据压缩标准（如JPEG2000）中起到重要作用。

算术编码

算术编码有两种模式

- 基于信源概率统计特性的固定编码模式和针对未知信源概率模型的自适应模式。
- 自适应模式中各个符号的概率初始值都相同，它们依据出现的符号而相应地改变。只要编码器和解码器都使用相同的初始值和相同的改变值的方法，那么它们的概率模型将保持一致。
- 上述两种形式的算术编码均可用硬件实现，其中自适应模式适用于不进行概率统计的场合。有关实验数据表明，在未知信源概率分布的情况下，算术编码一般要优于Huffman编码。在JPEG扩展系统中，用算术编码取代了哈夫曼编码。

算术编码

基本步骤如下：

1. 初始化数据，为每个可能输入的符号分配一个概率。设置初始 **编码区间** 为 $[0.0, 1.0)$，左区间记为 low，右区间记为 $high$，当前的编码区间的长度记为 $range = high - low$
2. 对于每个输入的编码，找到当前符号对应的 **初始概率区间** $[low_p, high_p)$，进行 **编码区间** 的更新：

$$\begin{cases} low = low + range \times low_p \\ high = low + range \times high_p \end{cases}$$

即

- **更新后的** low 等于 **上一次的** low 加上 **上一次的编码区间长度** $range$ 乘 **当前符号对应的初始概率区间** 的 low_p
- **更新后的** $high$ 等于 **上一次的** low 加上 **上一次的编码区间长度** $range$ 乘 **当前符号对应的初始概率区间** 的 $high_p$

3. 对于输入序列的符号，重复步骤2，不断减小编码区间
4. 当所有输入符号已经被编码，选择编码区间内的任何一个数作为整个输入序列的编码。为了解码方便，**通常选择区间的中点。**

算术编码

已知四个符号 $X1$、$X2$、$X3$、$X4$，它们出现的概率分别为 $1/5$、$1/5$、$2/5$、$1/5$，假设输入的消息序列为：$X1$、$X3$、$X4$、$X3$，求其算术编码的小数形式。

由题意可以将符号初始概率区间划分为：

$X1:[0,0.2)$

$X2:[0.2,0.4)$

$X3:[0.4,0.8)$

$X4:[0.8,1)$

初始情况编码区间 $low=0$，$high=1, range=1$。

1. 第一个输入的符号 $X1$，其对应的初始概率区间是 $[low_p, high_p)=[0,0.2)$：

$$\begin{cases} low = low + range \times low_p = 0 + 1 \times 0 = 0 \\ high = low + range \times high_p = 0 + 1 \times 0.2 = 0.2 \end{cases}$$

   更新得到当前的编码区间为 $[low, high)=[0,0.2), range=0.2$

2. 第二个输入的符号 $X3$，其对应的初始概率区间是 $[low_p, high_p)=[0.4,0.8)$：

$$\begin{cases} low = low + range \times low_p = 0 + 0.2 \times 0.4 = 0.08 \\ high = low + range \times high_p = 0 + 0.2 \times 0.8 = 0.16 \end{cases}$$

算术编码

3. 第三个输入的符号 $X4$，其对应的初始概率区间为 $[low_p, high_p) = [0.8, 1)$:

$$\begin{cases} low = low + range \times low_p = 0.08 + 0.08 \times 0.8 = 0.144 \\ high = low + range \times high_p = 0.08 + 0.08 \times 1 = 0.16 \end{cases}$$

更新得到当前的编码区间为 $[low, high) = [0.144, 0.16), range = 0.016$

4. 第四个输入的符号 $X3$，其对应的初始概率区间为 $[low_p, high_p) = [0.4, 0.8)$:

$$\begin{cases} low = low + range \times low_p = 0.144 + 0.016 \times 0.4 = 0.1504 \\ high = low + range \times high_p = 0.144 + 0.016 \times 0.8 = 0.1568 \end{cases}$$

更新得到最后的编码区间为 $[0.1504, 0.1568)$

故可以选取这个区间内的任意值作为最终的算术编码，可取 0.1536

算术编码

算术编码示例：编码来自一个 4 符号信源 $\{a_1, a_2, a_3, a_4\}$ 的由 5 个符号组成的符号序列：$b_1b_2b_3b_4b_5 = a_1a_2a_3a_3a_4$

符号	a_1	a_2	a_3	a_4
概率	0.2	0.2	0.4	0.2
初始区间	[0,0.2)	[0.2,0.4)	[0.4,0.8)	[0.8,1)

初始化时，被分割的范围为 **[0,1)**，下一个范围的低端和高端分别由下式计算：

Low=low + range×RangeLow

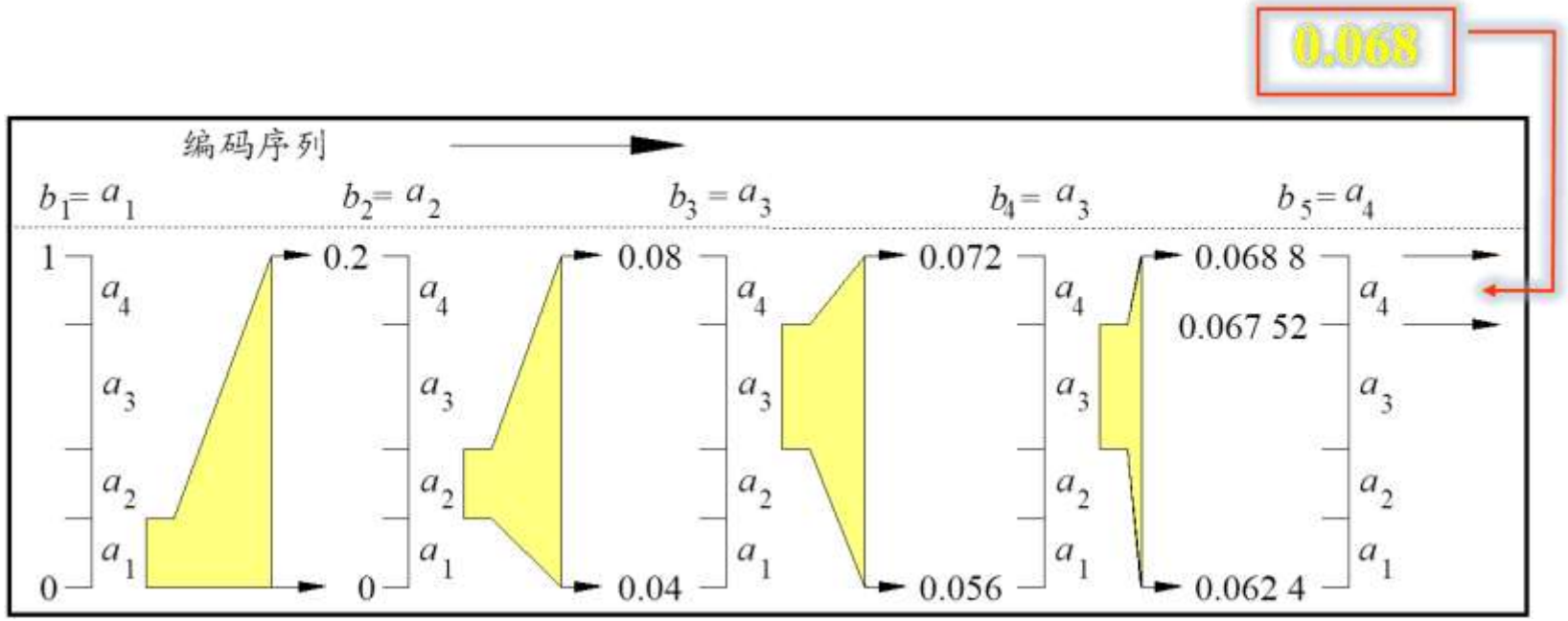
High=low + range×RangeHigh

其中**low**和**range**为上一个被编码字符的低端和范围。

RangeLow和**RangeHigh**分别为本次范围的低端和高端。

算术编码

符号	a_1	a_2	a_3	a_4
概率	0.2	0.2	0.4	0.2
初始区间	[0,0.2)	[0.2,0.4)	[0.4,0.8)	[0.8,1)



算术编码

算术编码计算过程示例

b1：初始化符号区间为[0 , 1)（理解为前一符号的空间），而符号b1=a1，为[0 , 0.2)

Low=0+1×0=0 $High= 0 + 1 \times 0.2$ =0.2

因此，b1的编码空间为[0,0.2)

b2：前符号区间为[0 , 0.2)，而符号b2为[0.2 , 0.4)

Low=0+0.2×0.2=0.04 $High= 0 + 0.2 \times 0.4$ =0.08

因此，b1的编码空间为[0.04,0.08)

后面依次类推

算术编码

算术编码的解码：

（1）根据信源符号及其概率，将区间[0,1)划分为若干子区间，互不重叠，各子区间长为各信源符号概率。

（2）将接收到的码字转换为小数Start。

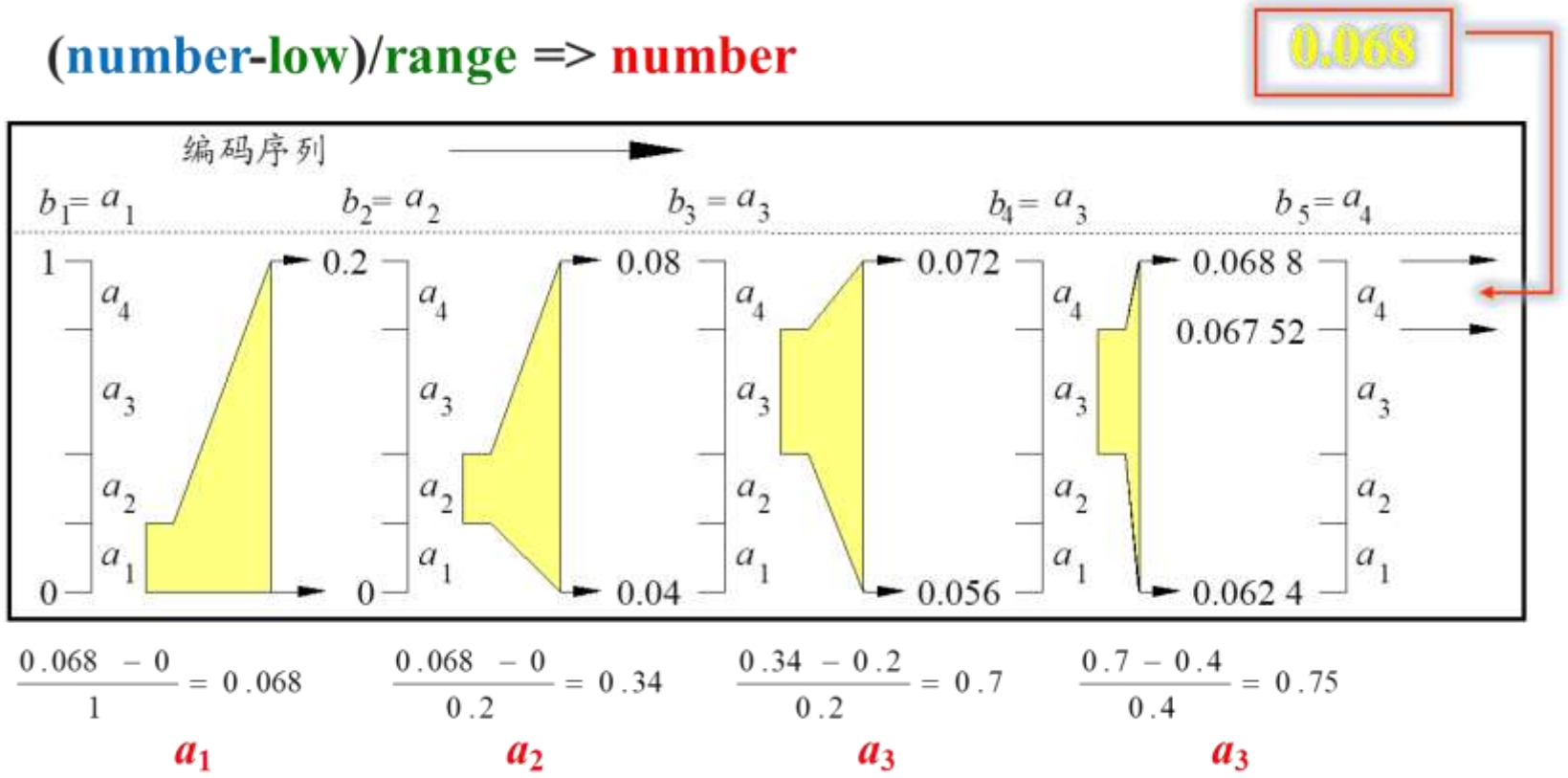
（3）根据该小数对应概率范围，确定字符串第一个字符i在[0,1)区间的起始位置$Left_i$和终止位置$Right_i$，区间长$L_i = Right_i - Left_i$。

（4）计算$(Start_i - Left_i)/L_i$，根据结果对应概率范围，确定字符串下一个字符j。

（5）重复上述过程，直到所有字符解码完毕。

算术编码

- 算术编码的解码：



算术编码

由于0.068在区间[0,0.2)，因此可知第一个信号源符号为a_1，得到信息源符号a_1后，由于已知a_1的上界和下界，利用编码可逆性，减去a_1的下界0，得到0.068，再用信源a_1的范围0.2去除，得到0.34，已知0.34落在信号源a_2的区间，所以得到第二个符号的信源符号为a_2。后续以此类推。

$$\frac{0.068-0}{1-0}=0.068\rightarrow a_1 \qquad \frac{0.068-0}{0.2-0}=0.34\rightarrow a_2 \qquad \frac{0.34-0.2}{0.4-0.2}=0.7\rightarrow a_3$$

$$\frac{0.7-0.4}{0.8-0.4}=0.75\rightarrow a_3 \qquad \frac{0.75-0.4}{0.8-0.4}=0.875\rightarrow a_4$$

算术编码

算术编码的特点：

- 由于实际的计算机精度不可能无限长，运算中会出现溢出问题。
- 算术编码器对整个消息只产生一个码字，这个码字是在[0, 1)之间的一个实数，因此译码器必须在接收到这个实数后才能译码。
- 算术编码也是一种对错误很敏感的方法。如果某一位发生错误会导致整个序列被译错，实际使用中，最好能在编码过程中估算信源概率。
- 使用算术编码的压缩算法通常先要对输入符号的概率进行估计，然后再编码。这个估计越准，编码结果就越接近最优的结果。

算术编码

【例】利用Matalab中的函数对Lena图像进行算术编解码

统计图像中各灰度出现的概率，通常先将图像转换为序列，调用函数编码、解码。由于图像中可能有部分灰度未曾出现，对应数量取值为0，调用函数会出错。因此，需要将这一部分灰度剔除。【Eg12_7】

```
%算术编码
clc;clear,close all;
Image=imread('lena.bmp');
subplot(1,2,1),imshow(Image);title('(a)原图像
','FontSize',14);
[h,w]=size(Image);
map=zeros(256,2);
map(:,1)=1:256;%map中第一列存放1:256灰度级
counts=imhist(Image);
num=1;
for i=1:256
    if counts(i)
        map(i,2)=num;
        num=num+1;
    end
end
```

算术编码

【例】利用Matalab中的函数对Lena图像进行算术编解码

```
counts(counts==0)=[];%去掉未曾出现灰度的出现次数0
In=double(Image)+1;%灰度值范围由0~255 变为1~256
for i=1:256
    In(In==i)=map(i,2);%修改图像中的灰度值
end
seq=reshape(In,1,h*w);%变为向量
code=arithenco(seq,counts);%算术编码
dseq=arithdeco(code,counts,h*w);%算术解码
flag=isequal(seq,dseq);%判断解码后字符串和原字符是否一致
deImage=reshape(dseq,h,w);
for i=256:-1:1
    if map(i,2)~=0
        deImage(deImage==map(i,2))=map(i,1);%%恢复图像中的灰度值
    end
end
subplot(1,2,2),imshow(uint8(deImage-1));title('(b)算术编码解码图像
','FontSize',14);
disp(flag)
```

- code=arithenco(seq,counts)：根据指定向量seq对应的符号序列产生二进制算术代码， 向量counts 是代码信源中指定符号在数据集合中出现的次数统计。
- dseq=arithdeco (code,counts,len)：解码二进制算术代码code，恢复相应的len 符号列。

算术编码

- ✓ 程序运行，编码后的code为487885位二进制码，比原图需要524288位二进制码略有压缩。flag变量为1，解码后图像和原图像一致，实现了无损编码。
- ✓ 在实际应用中，算术编码一般也和其他的方法结合在一起使用，能提高效率。

算术编码

【例】利用Matalab中的函数对Lena图像进行算术编解码

(a)原图像

(b)算术编码解码图像

香农-范诺编码

香农-范诺编码（Shannon–Fano coding）是一种基于一组符号集及其出现的概率构建前缀码的技术。

香农-范诺编码名称来自于以克劳德·香农和罗伯特·法诺。

香农-范诺编码属于**不等长编码**，通常将经常出现的消息变成短码，不经常出现的消息编成长码，从而提高通信效率。香农-范诺编码严格意义上来说不是最佳码，它是采用信源符号的累计概率分布函数来分配码字。只有当信源符号的概率分布使不等式左边的等号成立时，编码效率才达到最高。

- **符号的码字长度N_i完全由该符号出现的概率来决定，**

$$-\log_2 P_i \le N_i \le -\log_2 P_i + 1$$

香农-范诺编码

香农-范诺编码的步骤

步骤1：将信源符号按其出现概率从大到小排序；

步骤2：计算出各概率对应的码字长度N_i；

步骤3：计算累加概率A_i，即

$$A_i = A_{i-1} + P_{i-1}, i = 1,2,\ldots,N-1; \quad A_0 = 0$$

步骤4：将各个累加概率A_i由十进制转化为二进制，取该二进制数的前N_i位作为对应信源符号的码字。

香农-范诺编码

例1对如下信源编码：

信源符号	S1	S2	S3	S4	S5	S6	S7
符号概率	0.20	0.19	0.18	0.17	0.15	0.10	0.01

香农编码如表所示，

信源符号 s_i	符号概率 $p(s_i)$	累加概率 P_i	$-\log p(s_i)$	码长 l_i	码字
s_1	0.20	0	2.34	3	000
s_2	0.19	0.2	2.41	3	001
s_3	0.18	0.39	2.48	3	011
s_4	0.17	0.57	2.56	3	100
s_5	0.15	0.74	2.74	3	101
s_6	0.10	0.89	3.34	4	1110
s_7	0.01	0.99	6.66	7	1111110

平均码长为：(0.2+0.19+0.18+0.17+0.15)x3+0.1×4+0.01×7=3.14

香农-范诺编码

累加概率，变成二进制数，为0.1001…。转换的方法为：用Pi乘以2，如果整数部分有进位，则小数点后第一位为1，否则为0，将其小数部分再做同样的处理，得到小数点后的第二位，依此类推，直到得到了满足要求的位数，或者没有小数部分了为止。可以看出，编码所得的码字，没有相同的，所以是非奇异码，也没有一个码字是其他码字的前缀，所以是即时码，也是唯一可译码。

香农-范诺编码

例. 设一幅灰度级为8（分别用S_0、 S_1、 S_2、 S_3、 S_4、 S_5、 S_6、S_7表示） 的图像中，各灰度所对应的概率分别为0.40、 0.18、0.10、 0.10、 0.07、 0.06、 0.05、 0.04。 现对其进行香农-范诺编码。

香农-范诺编码

信源符号	出现概率 P_i	码字长度 N_i	累加概率 A_i	转换为二进制	分配码字 B_i
x_1	0.40	2	0	0	00
x_2	0.18	3	0.40	0110	011
x_3	0.10	4	0.58	10010	1001
x_4	0.10	4	0.68	10100	1010
x_5	0.07	4	0.78	11000	1100
x_6	0.06	5	0.85	110110	11011
x_7	0.05	5	0.91	111010	11101
x_8	0.04	5	0.96	111101	11110
平均码长 R=3.17		图像熵 H=2.55		编码效率 η = 80.4%	

二分法香农-范诺编码

二分法香农-范诺编码算法

步骤1：首先统计出每个符号出现的概率；

步骤2：从左到右对上述概率从大到小排序；

步骤3：将概率集合从某个位置分为两个子集合，尽量使两个子集合的概率和近似相等，前面子集合赋值为0，后面为1；

步骤4：重复步骤3，直到各个子集合中只有一个元素为止；

步骤5：将每个元素所属的子集合的值依次串起来即可。

二分法香农-范诺编码

码字	符号	出现概率				
00	S_0	0.40	0.58(0)	0.40(0)		
01	S_1	0.18		0.18(1)		
100	S_2	0.10	0.42(1)	0.20(0)	0.10(0)	
101	S_3	0.10			0.10(1)	
1100	S_4	0.07		0.21(1)	0.13(0)	0.07(0)
1101	S_5	0.06				0.06(1)
1110	S_6	0.05			0.09(1)	0.05(0)
1111	S_7	0.04				0.04(1)
平均码长 $R=2.64$			图像熵 $H=2.55$		编码效率 $\eta=96.59\%$	

二分法香农-范诺编码

- 在理想意义上，香农-范诺编码与哈夫曼编码一样，并未实现码词（code word）长度的最低预期
- 与哈夫曼编码不同的是，它确保了所有的码词长度在一个理想的理论范围$-\log P(x)$之内。
- 香农-范诺编码不应该与香农编码混淆

目录

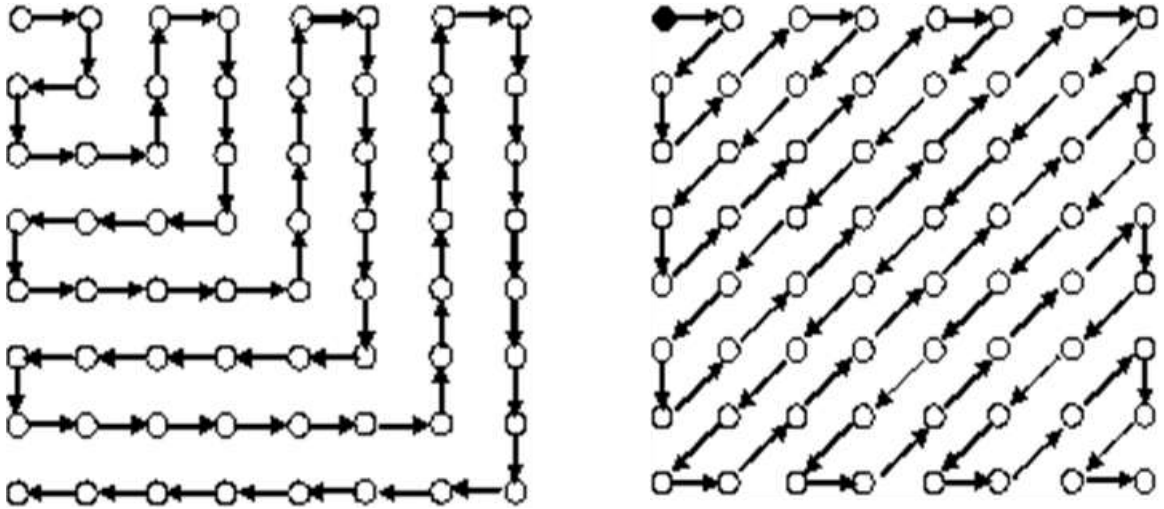
行程编码

- 行程编码（Run Length Encoding）：一种利用空间冗余度压缩图像的方法，也称为游程编码，属于无损编码。
- 行程编码将具有相同值的连续串用其**串长和一个代表值**来代替，该连续串就称为行程，串长称为行程长度。

例如：aaabccccccddeee，不压缩存储需要15×8=120位，采用行程长度编码为：3a1b6c2d3e，需要10×8=80位，减少存储位数。

行程编码

- 图像行程编码：二维排列的像素转化成一维排列的方式，再按照一维行程编码方式进行编码。



行程编码

- 如果图像是由很多颜色或灰度相同的大面积区域组成的，例如灰度图像和二值图像，那么采用行程编码可以达到很高的压缩比。
- 如果图像中的数据**非常分散**，则行程编码不但不能压缩数据，反而会增加图像文件的大小。行程编码不适用于连续色调图像的压缩，例如日常生活中的照片。
- 一般和其他编码方法结合使用以期达到更好的压缩效果。

行程编码

图 一幅二值图像

10	8			
0	7	1	3	0
6	1	3	0	6
1	3	0	6	1
3	0	6	1	3
0	6	1	3	0
6	1	3	0	6
1	3	U	6	

图 二值图像对应的行程编码

其中10和8表示图像的宽和高。

行程编码

在上述例子中行程编码并没有起到压缩图像的作用。这是由于这个图像的尺寸过小，当图像尺寸较大时行程编码还是不错的无损压缩方法。

行程编码方法实现起来很容易，对于具有长重复值的串的压缩编码很有效。例如：对于有大面积的阴影或颜色相同的图像，使用这种方法压缩效果很好。很多位图文件格式都采用行程编码，如TIFF，PCX，GEM，BMP等。

行程编码

```
%行程编码
clc;clear all;
I=imread('lena.jpg');
if 3 == ndims(I)
    I=rgb2gray();
end
I = imbinarize(I);
[m,n] = size(I);
sig = I(:);  % 转化为一维数据
% 行程编码
enco = runlengthenco(sig);  % 编码
dsing = runlengthdeco(enco);  % 解码
ide = col2im(dsing,[m,n],[m,n],‘distinct’);  % 将向量重新转换成图像块
figure;
subplot(121),imshow(I,[]),title('原图');
subplot(122),imshow(ide,[]),title('解压图像');

Cr=imratio(ide,I)
```

行程编码

```
% 行程编码
function enco = runlengthenco(sig)
    enco = zeros(1);%初始化编码结果数组enco，创建一个1x1的全零矩阵
    k = 1;%初始化指针k=1，用于跟踪编码数组的当前位置
    enco(k) = sig(1);%将编码数组的第一个元素设置为输入信号的第一个值
    enco(k+1) = 1;%将编码数组的第二个元素设置为1（表示第一个值的连续出现次数）
    i = 2;
    while (i<=length(sig))
        if (sig(i)==enco(k))
            enco(k+1) = enco(k+1)+1;
        else
            k = k+2;
            enco(k) = sig(i);
            enco(k+1) = 1;
        end
        i = i+1;
    end
    enco = enco';
end
```

```
% 行程编码的解码
function dsing = runlengthdeco(enco)
    dsing = (-1);
    for i = 1:2:length(enco)-1
        for j = 1:enco(i+1)
            dsing = [dsing enco(i)];
        end
    end
    dsing(dsing==-1) = [];
    dsing = dsing';
end
```

行程编码

原图

解压图像

LZW编码

- LZW编码（Lempel-Ziv & Welch）又称字串表编码，属于无损编码。是一种基于字典的编码；能减少或消除图像像素间冗余。
  - ✓ 70年代末，以色列研究者A.Lempel和J.Ziv提出LZ77和LZ78算法。。

LZ77算法：Ziv J , Lempel A. A universal algorithm for sequential data compression[J].IEEE Transactions on Information Theory, 1977, 23(3):337-343.DOI:10.1109/TIT.1977.1055714.

LZ78算法：Ziv J , Lempel A. Compression of Individual Sequences Via Variable-Rate Coding[J].IEEE Transactions on Information Theory, 1978, 24(5):530-536.DOI:10.1109/TIT.1978.1055934.

A Universal Algorithm for Sequential Data Compression

JACOB ZIV, FELLOW, IEEE, AND ABRAHAM LEMPEL, MEMBER, IEEE

Abstract—**A universal algorithm for sequential data compression is presented. Its performance is investigated with respect to a nonprobabilistic model of constrained sources. The compression ratio achieved by the proposed universal code uniformly approaches the lower bounds on the compression ratios attainable by block-to-variable codes and variable-to-block codes designed to match a completely specified source.**

I. INTRODUCTION

IN MANY situations arising in digital communications and data processing, the encountered strings of data display various structural regularities or are otherwise subject to certain constraints, thereby allowing for storage and time-saving techniques of data compression. Given a discrete data source, the problem of data compression is first to identify the limitations of the source, and second to devise a coding scheme which, subject to certain performance criteria, will best compress the given source.

Once the relevant source parameters have been identified, the problem reduces to one of minimum-redundancy coding. This phase of the problem has received extensive treatment in the literature [1]-[7].

When no *a priori* knowledge of the source characteristics is available, and if statistical tests are either impossible or unreliable, the problem of data compression becomes considerably more complicated. In order to overcome these difficulties one must resort to universal coding schemes whereby the coding process is interlaced with a learning process for the varying source characteristics [8], [9]. Such coding schemes inevitably require a larger working memory space and generally employ performance criteria that are appropriate for a wide variety of sources.

In this paper, we describe a universal coding scheme which can be applied to any discrete source and whose performance is comparable to certain optimal fixed code book schemes designed for completely specified sources. For lack of adequate criteria, we do not attempt to rank the proposed scheme with respect to other possible universal coding schemes. Instead, for the broad class of sources defined in Section III, we derive upper bounds on the compression efficiency attainable with full *a priori* knowledge of the source by fixed code book schemes, and then show that the efficiency of our universal code with no *a priori* knowledge of the source approaches those bounds.

The proposed compression algorithm is an adaptation of a simple copying procedure discussed recently [10] in a study on the complexity of finite sequences. Basically, we employ the concept of encoding future segments of the source-output via maximum-length copying from a buffer containing the recent past output. The transmitted codeword consists of the buffer address and the length of the copied segment. With a predetermined initial load of the buffer and the information contained in the codewords, the source data can readily be reconstructed at the decoding end of the process.

The main drawback of the proposed algorithm is its susceptibility to error propagation in the event of a channel error.

Manuscript received June 23, 1975; revised July 6, 1976. Paper previously presented at the IEEE International Symposium on Information Theory, Ronneby, Sweden, June 21-24, 1976.

J. Ziv was with the Department of Electrical Engineering, Technion-Israel Institute of Technology, Haifa, Israel. He is now with the Bell Telephone Laboratories, Murray Hill, NJ 07974.

A. Lempel was with the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa, Israel. He is now with the Sperry Research Center, Sudbury, MA 01776.

II. THE COMPRESSION ALGORITHM

The proposed compression algorithm consists of a rule for parsing strings of symbols from a finite alphabet A into substrings, or words, whose lengths do not exceed a prescribed integer L_s, and a coding scheme which maps these substrings sequentially into uniquely decipherable codewords of fixed length L_c over the same alphabet A.

The word-length bounds L_s and L_c allow for bounded-delay encoding and decoding, and they are related by

$$L_c = 1 + \lceil \log (n - L_s) \rceil + \lceil \log L_s \rceil, \qquad (1)$$

where $\lceil x \rceil$ is the least integer not smaller than x, the logarithm base is the cardinality α of the alphabet A, and n is the length of a buffer, employed at the encoding end of the process, which stores the latest n symbols emitted by the source. The exact relationship between n and L_s is discussed in Section III. Typically, $n \simeq L_s \alpha^{hL_s}$, where $0 < h < 1$. For on-line decoding, a buffer of similar length has to be employed at the decoding end also.

To describe the exact mechanics of the parsing and coding procedures, we need some preparation by way of notation and definitions.

Consider a finite alphabet A of α symbols, say $A = \{0,1,\cdots,\alpha-1\}$. A string, or word, S of length $\ell(S) = k$ over A is an ordered k-tuple $S = s_1 s_2 \cdots s_k$ of symbols from A. To indicate a substring of S which starts at position i and ends at position j, we write $S(i,j)$. When $i \leq j$, $S(i,j) = s_i s_{i+1} \cdots s_j$, but when $i > j$, we take $S(i,j) = \Lambda$, the null string of length zero.

The concatenation of strings Q and R forms a new string $S = QR$; if $\ell(Q) = k$ and $\ell(R) = m$, then $\ell(S) = k + m$, $Q = S(1,k)$, and $R = S(k+1, k+m)$. For each j, $0 \leq j \leq$

Compression of Individual Sequences via Variable-Rate Coding

JACOB ZIV, FELLOW, IEEE, AND ABRAHAM LEMPEL, MEMBER, IEEE

Abstract—**Compressibility of individual sequences by the class of generalized finite-state information-lossless encoders is investigated. These encoders can operate in a variable-rate mode as well as a fixed-rate one, and they allow for any finite-state scheme of variable-length-to-variable-length coding. For every individual infinite sequence x a quantity $\rho(x)$ is defined, called the compressibility of x, which is shown to be the asymptotically attainable lower bound on the compression ratio that can be achieved for x by any finite-state encoder. This is demonstrated by means of a constructive coding theorem and its converse that, apart from their asymptotic significance, also provide useful performance criteria for finite and practical data-compression tasks. The proposed concept of compressibility is also shown to play a role analogous to that of entropy in classical information theory where one deals with probabilistic ensembles of sequences rather than with individual sequences. While the definition of $\rho(x)$ allows a different machine for each different sequence to be compressed, the constructive coding theorem leads to a universal algorithm that is asymptotically optimal for all sequences.**

I. INTRODUCTION

IN A RECENT paper [1], data-compression coding theorems and their converses were derived for the class of finite-state encoders that map at a fixed rate input strings drawn from a source of α letters into equally long strings over an alphabet of $\beta < \alpha$ letters. In the context of data-compression, the aim is to minimize the number of bits/symbol $\log_2 \beta$, while securing zero or negligibly small distortion. For every individual infinite sequence x, this minimal bit/symbol rate was shown in [1] to be equal to a quantity $H(x)$ that, in analogy with the Shannon entropy (which is defined for probabilistic ensembles rather than

Manuscript received June 10, 1977; revised February 20, 1978.

J. Ziv is with Bell Laboratories, Murray Hill, NJ 07974, on leave from the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa, Israel.

A. Lempel is with Sperry Research Center, Sudbury, MA 01776, on leave from the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa, Israel.

0018-9448/78/0900-0530$00.75

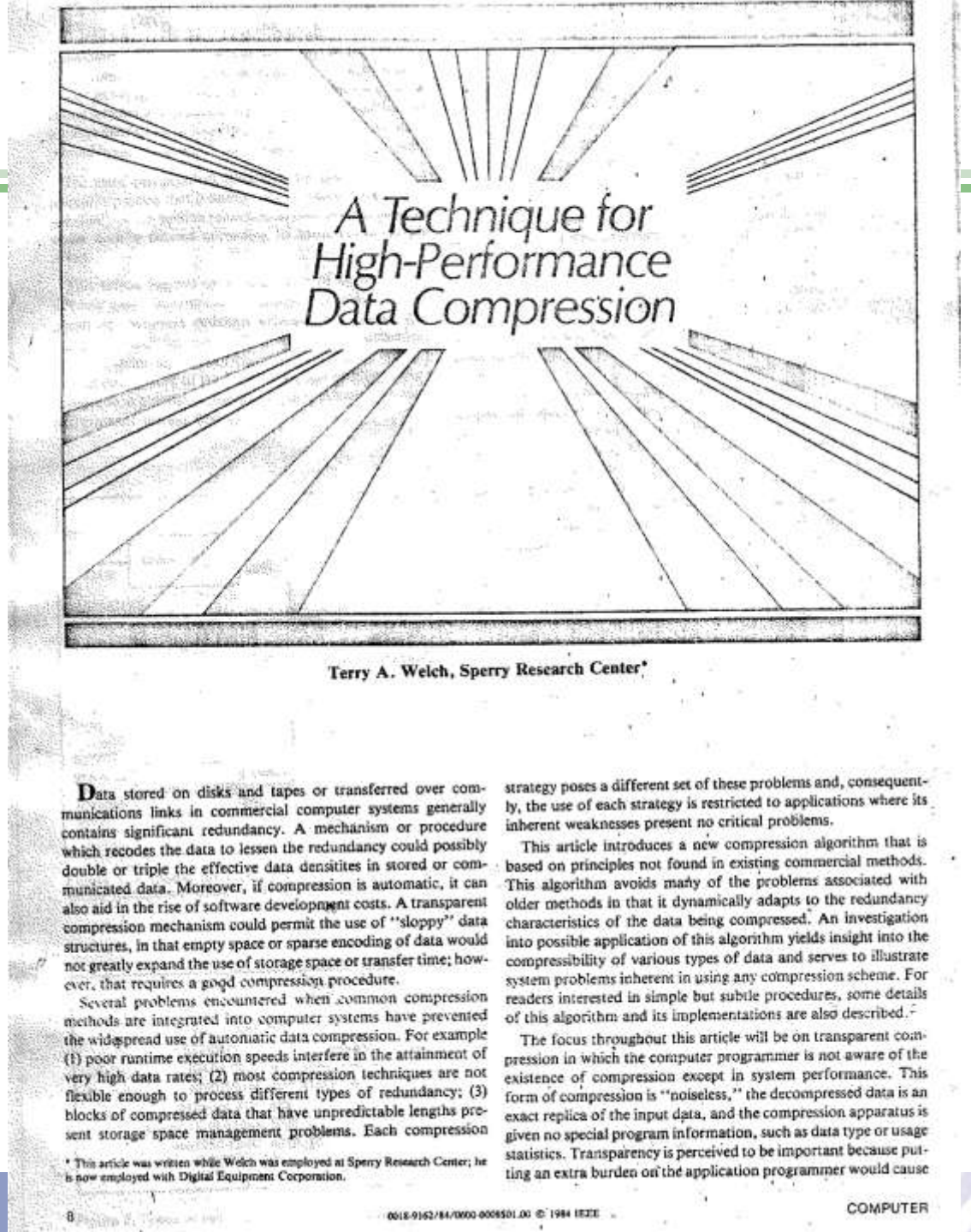
LZW编码

LZ77算法和LZ78算法的核心思想：利用数据本身的冗余性（重复出现的模式或字符串），通过构建一个字典（码表） 来用较短的代码（codeword） 表示这些重复出现的字符串。LZ77/LZ78 算法奠定了字典压缩基础，形成通用数据压缩算法体系，该系列算法通过构建动态字典实现重复序列的高效压缩，突破Huffman编码需预知信源统计特性的限制，LZ77采用滑动窗口更新字典，LZ78改进为显式字典结构。算法将输入序列划分为互不重复的短语，编码时用短语索引和新增字符构成码字，解码时通过索引重建字典。通过动态扩展字典捕捉序列冗余，随着序列增长，压缩效率逐渐逼近信源熵的理论极限。

LZW编码

1984 年 6 月，Terry A. Welch发表论文——
Welch, T. A. A Technique for High-Performance Data Compression[J].Computer, 1984, 17(6):8-19.DOI:10.1109/MC.1984.1659158.

Welch对LZ78算法做出了几项关键的、使其更高效和实用的改进，Welch的改进极大地提升了算法的性能、效率和实现的简洁性，使其真正适合于实际应用。LZW算法曾为Unisys公司专利技术，2003年后解除使用限制。



A Technique for High-Performance Data Compression

Terry A. Welch, Sperry Research Center*

Data stored on disks and tapes or transferred over communications links in commercial computer systems generally contains significant redundancy. A mechanism or procedure which recodes the data to lessen the redundancy could possibly double or triple the effective data densitites in stored or communicated data. Moreover, if compression is automatic, it can also aid in the rise of software development costs. A transparent compression mechanism could permit the use of "sloppy" data structures, in that empty space or sparse encoding of data would not greatly expand the use of storage space or transfer time; however, that requires a good compression procedure.

Several problems encountered when common compression methods are integrated into computer systems have prevented the widespread use of automatic data compression. For example (1) poor runtime execution speeds interfere in the attainment of very high data rates; (2) most compression techniques are not flexible enough to process different types of redundancy; (3) blocks of compressed data that have unpredictable lengths present storage space management problems. Each compression strategy poses a different set of these problems and, consequently, the use of each strategy is restricted to applications where its inherent weaknesses present no critical problems.

This article introduces a new compression algorithm that is based on principles not found in existing commercial methods. This algorithm avoids many of the problems associated with older methods in that it dynamically adapts to the redundancy characteristics of the data being compressed. An investigation into possible application of this algorithm yields insight into the compressibility of various types of data and serves to illustrate system problems inherent in using any compression scheme. For readers interested in simple but subtle procedures, some details of this algorithm and its implementations are also described.

The focus throughout this article will be on transparent compression in which the computer programmer is not aware of the existence of compression except in system performance. This form of compression is "noiseless," the decompressed data is an exact replica of the input data, and the compression apparatus is given no special program information, such as data type or usage statistics. Transparency is perceived to be important because putting an extra burden on the application programmer would cause

* This article was written while Welch was employed at Sperry Research Center; he is now employed with Digital Equipment Corporation.

0018-9162/84/0600-0008$01.00 © 1984 IEEE

LZW编码

- LZW算法
  - ✓ 垄断数据压缩领域—gz、zip、arj
  - ✓ 在标准图像数据格式中得到广泛应用—GIF、TIFF
- **思路**：用数据中字符序列构建字典；给出字典的索引作为编码结果。

LZ77、LZ78和LZW均为无损压缩编码算法，旨在不产生信息失真的同时降低信息冗余度。LZ压缩算法系列还有LZSS、LZH、LZB、DEFLATE、LZMA/LZMA2、LZO、LZ4、LZF、Snappy、zstd、Brotli 等算法。

LZW编码

- LZW编码对信源符号的可变长度序列分配固定长度的码字，且不需要了解被编码符号的概率等先验知识。
- LZW编码基本思想：在编码过程中，将所遇到的字符串建立一个字符串表（码书、字典），表中的每个字符串都对应一个索引，编码时用该字符串在字典中的索引来代替原始的数据串（类似于调色板图像）。
- 字典是在压缩过程中**动态生成**，不必保存在压缩文件里，解压缩时字典可以由压缩文件中的信息重新生成。

LZW编码

基本思想

- ✓ 构建初始编码字典（码本）。每个字符序列对应一个字典索引值。
- ✓ 动态更新字典。每发现字典中没有的字符序列，则存储。
- ✓ 用字典索引值作为该字符序列的编码码字。
- ✓ 下次再遇到相同字符序列，则用字典索引值进行代替。
- ✓ 压缩结果不保留压缩过程中形成的字典。

LZW编码

算法步骤

步骤1：构建初始字典，包含图像信源中所有可能的字符串

步骤2：定义变量S为“输入字符像素”，变量R为“当前识别字符序列”，初始化R为空。

步骤3：判断生成的新连接字串“RS”是否在字典中：若“RS ”在字典中，则无编码输出，令R=RS；若“RS”不在字典中，则添加“RS ”到字典末尾，编码输出为R在字典中位置，令R=S；

步骤4：判断图像信源数据流中是否还有码字要译。如果“是”，则返回到步骤2，继续编码。如果“否”，则将代表当前识别字符序列R在字典中位置作为编码输出。

LZW编码

对一如下一幅图像f 进行 LZW 编码。

$$\begin{bmatrix} 30 & 30 & 30 & 30 \\ 110 & 110 & 110 & 110 \\ 30 & 30 & 30 & 30 \\ 110 & 110 & 110 & 110 \end{bmatrix}$$

LZW编码

解：建立初始化字符串表。

- 一个有512字（位置）的字典；其中字典中前256个字符位置分别分配给灰度度0，1，……，255，位置256~511还没有用到，暂时为空。

位置	0	1	…	255	256	…	511
索引值	0	1	…	255			

LZW编码

图像通过从左到右、从上到下的顺序处理器像素并进行编码，编码过程如下表所示。

当前识别序列R	被处理的像素S	编码输出	字典条目	字典位置（码字）
	30（1）	不输出（因为30已经在字典中）	不添加	
30（1）	30（2）	30（30对应的字典位置）	30-30	**256**
30（2）	30（3）	不输出		
30-30	30（4）	256	30-30-30	**257**
30（4）	110（5）	30	30-110	**258**
110（5）	110（6）	110	110-110	**259**
110（6）	110（7）	不输出		
110-110	110（8）	259	110-110-110	**260**
110（8）	**30（9）**	**110**	**110-30**	**261**

LZW编码

当前识别序列R	被处理的像素S	编码输出	字典条目	字典位置（码字）
30 （9）	30 （10）	不输出		
30-30	30 （11）	不输出		
30-30-30	30 （12）	257	30-30-30-30	**262**
30 （12）	110 （13）	不输出		
30-110	110 （14）	258	30-110-110	**263**
110 （14）	110 （15）	不输出		
110-110	**110 （16）**	**260**		

最后的编码结果为 {30, 256, 30, 110, 259, 110, 257, 258, 260}

进行PCM(Pulse Code Modulation）**)自然二进制编码输出共：9 × 9 = 81bit ， 而原始数据为：16 × 8 = 128bit， 压缩比为：** $\frac{128}{81} \approx 1.58$

LZW编码

- **自适应性**

  构建的字典（字符串表），从初始空状态到建立生成表中内容，算法具有自适应性。

- **前缀性**

  字典（字符串表）中任意一字符串的前缀字符串也在字典（字符串表）中。

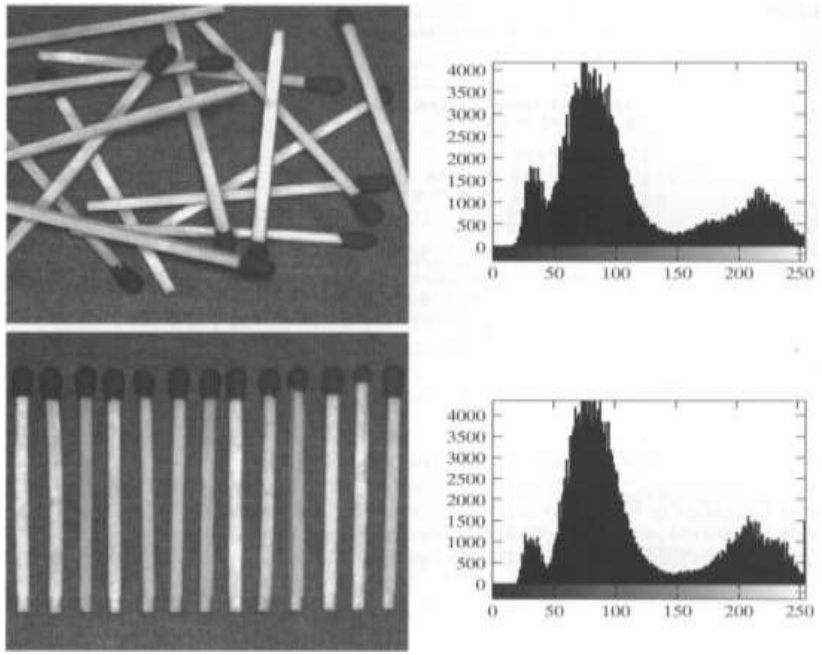
- **动态性**

  LZW编码过程中，动态生成建立字典（字符串表）

空间冗余

- 变长编码难以消除像素间相关造成的空间冗余。

```
%空间冗余
f1 = imread( 'matches-random.tif');
c1 = mat2huff(f1);  % 哈夫曼
ntrop(f1)   % 计算熵
%（去除了 0 概率，因为 log20 =-∞）
imratio(f1, c1)
% 压缩比
f2 = imread( 'matches-aligned.tif') ;
c2 = mat2huff(f2)
ntrop(f2)
imratio(f2, c2)
```



- 相邻像素之间具有较强的相关性，可以根据已知的像素来估计、预测新像素。

预测编码

- 预测编码（Predictive Coding)是根据离散信号之间存在一定**相关性**的特点，利用前面一个或多个信号预测当前信号，然后对实际值和预测值的预测误差进行编码。如果预测比较准确，误差就会很小。在同等精度要求的条件下，就可以用比较少的比特进行编码，达到压缩数据的目的。
- 在图像压缩中，预测编码建立在去除图像**空间冗余和时间冗余**的基础上，利用邻近像素间或相邻帧之间图像的高度相关性，在编码时，只对新信息(预测误差信息)进行编码从而提高压缩率。
- 新信息定义为该像素的当前值与对它预测值的差。

预测编码

- **预测编码器由预测器、量化器和编码器构成**。
- 预测器的目的是由过去的信息预测当前的信息，在这一步并没有减少数据量。在图像编码中，预测器分为帧内预测和帧间预测。帧内预测是利用若干个像素点的值来预测当前像素点的灰度值，**目的是去除空间冗余**；帧间预测是利用过去的帧来预测当前帧，目的是去除时间冗余。

预测编码

- **量化器**是用来表示如何看待误差的问题，由于人眼存在心理视觉冗余，在图像压缩时，可以忽略较小的误差，减少数据量而不影响图像视觉效果，但这种损失**不可恢复**。将带有量化器的预测编码称为有损预测编码，而将不带量化器的预测编码称为无损预测编码。
- **编码器**的目的在于对量化后的误差进行压缩减少数据量。

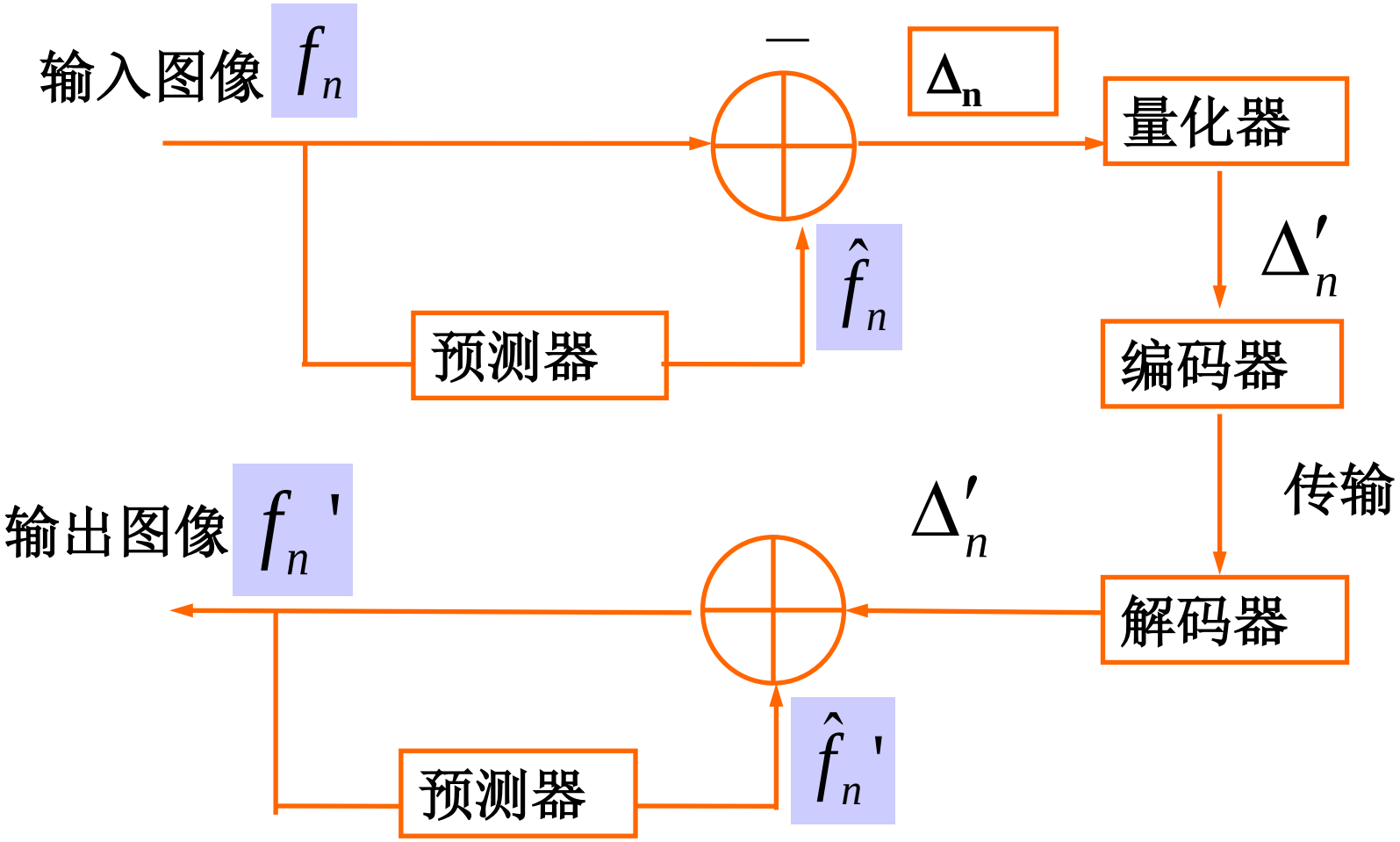
预测编码

- 预测器可分为线性预测器和非线性预测器。
- 利用非线性方程计算预测值的预测器称为非线性预测器;
- 用线性方程计算预测值的预测器称为线性预测器。

预测编码

- 预测编码非常适合声音和图像的压缩。
  - ✓ 对于声音来讲，预测的对象是声波的下一个幅度、下一个音色。
  - ✓ 对于图像而言，预测的对象是下一个像点、下一条线或下一帧。
- 特点：声音和图像中通常都存在冗余的信号，而且在相邻的音色或相邻像素点之间的相关性比较强，它们的差值比较小。这样任何音色或像点都可以通过已知样本值进行预测。对于连续的多帧图像，上下帧通常具有一些相同的部分内容，如背景和静止的物体，可以预计在一定的时间内将不会发生变化。

预测编码



预测编码示意图

预测编码

- 预测器： $\hat{f}_n = F(f_{n-1}, f_{n-2}, \cdots, f_{n-k})$
- $\hat{f}_n$是根据前面k个像素的亮度值$f_{n-1}, f_{n-2}, \cdots, f_{n-k}$预测，进而得到预测误差$\Delta_n = f_n - \hat{f}_n$
- 量化器：对Δ_n进行四舍五入取整量化
- 编码器：可采用成熟的编码技术，如Huffman编码等
- 解码器：编码器的逆
- 线性预测器： $\hat{f}_n = F(f_{n-1}, f_{n-2}, \cdots, f_{n-k}) = \sum_{k=l}^{n-1} a_k f_k$， $\sum a_k = 1$

预测编码

例：

	2	4	6	8	8	4	2	10
$\hat{f}$	2	4	3	5	7	8	6	3
Δ	2	4	3	3	1	-4	-4	7
	2	4	6	8	8	4	2	10

预测器

$$\hat{f}_n = F(f_{n-1}, f_{n-2}) = \sum_{k=n-2}^{n-1} a_k f_k, \quad a_k = 0.5$$

前两个元素的平均值

预测编码

- 接收端解码时的预测过程与发端相同，所用预测器也相同，收端输出的信号是发端的近似值，二者的误差是

$$f_n' - f_n = \hat{f}_n + \Delta'_n - f_n = \Delta'_n - \Delta_n$$

- 每行的最开始的几个像素无法预测，这些像素需要用其他方式编码，这是采用预测编码所需要的**额外操作**。
- 预测系数随着不同的图像而不同，但对每幅图像都计算预测系数太麻烦，也不现实，可参考前人得到的数据选择使用。在静止图像压缩的国际标准(JPEG)中，对这种方法的前置点形式以及预测系数有一些推荐值可供参考。

无损预测编码

无损预测的基本思想

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。（去除像素冗余）。

操作：用当前像素值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，对差进行编码，作为压缩数据流中的下一个元素。

由于差比原数据要小，因而编码要小，可用变长编码。

无损预测编码

在多数情况下，预测是由前m个样本的线性组合生成的，即

$$\hat{f}(n) = \text{round}\left[\sum_{i=1}^{m} \alpha_i f(n-i)\right]$$

m是线性预测器的阶数，round表示四舍五入或取最接近整数的一个函数，i=1，2，…，m是预测系数。

对于一维线性预测图像编码，上式可以写为

$$\hat{f}(x,y) = round\left[\sum_{i=1}^{m} \alpha_i f(x,y-i)\right]$$

式中，每个样本现在都显式地表示为输入图像空间坐标x和y的函数。注意上式表明一维线性预测只是当前扫描行上前几个像素的函数。

无损预测编码

编码与解码过程

编码过程：

第1步：压缩头处理

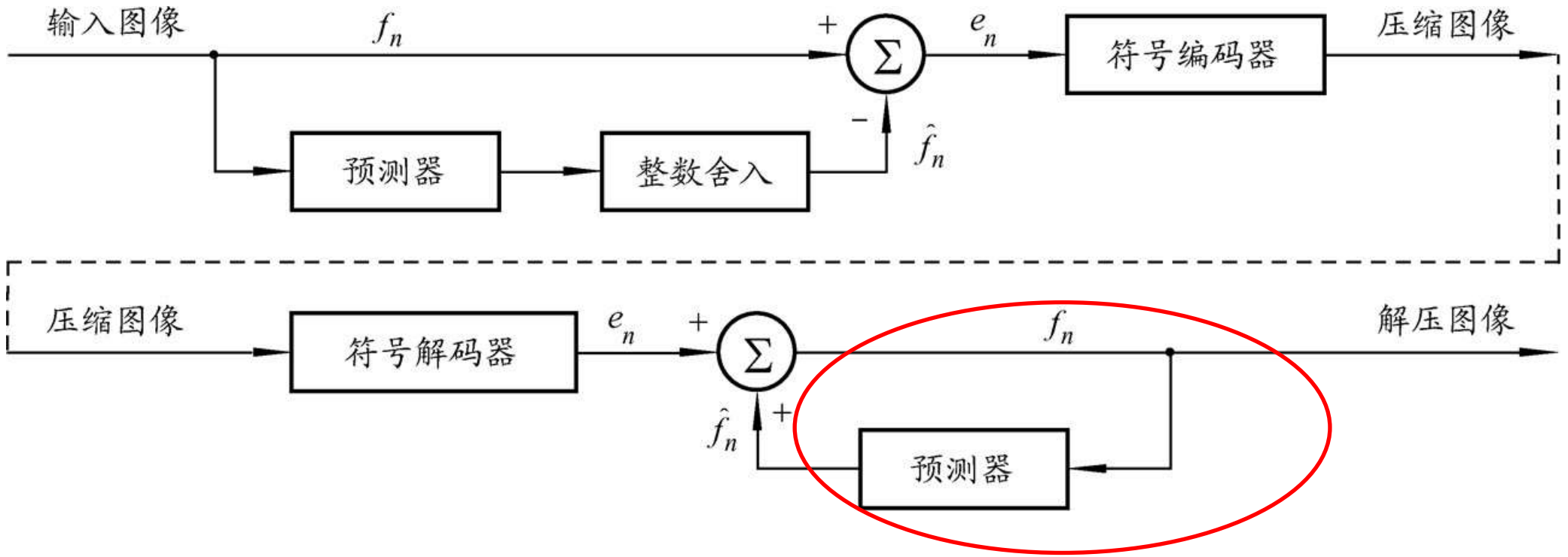
第2步：对每一个符号$f(x,y)$，根据规定数量的以往样本生成每个样本的预测值。然后预测器的输出被四舍五入为最接近的整数， 即预测值$\hat{f}(x,y)$

第3步：求出预测误差：$e(x,y) = f(x,y) - \hat{f}(x,y)$

第4步：对误差$e(x,y)$进行变长编码（符号解码器），生成压缩数据流的下一个元素（作为压缩值）。

重复步骤2、3、4。

无损预测编码



此处并不是表示输入了原始图像，而是一个根据预测的值进行循环的过程，最终得到是f_n。

无损预测编码

编码与解码过程

解码过程：

第1步：对头解压缩

第2步：对每一个预测误差的编码解码，得到预测误差$e(x,y)$。

第3步：由前面的值，得到预测值$\hat{f}(x,y)$。

第4步：误差$e(x,y)$与预测值$\hat{f}(x,y)$相加， 得到解码$f(x,y)$。

重复步骤2、3、4。

无损预测编码

```
% 预测编码Main函数（主程序）
%对peppers.bmp图像进行无损一阶预测编码，采用前值预测。
clc;clear all;
X=imread('peppers.png');
if ndims(X)>2
    X= rgb2gray(X)
end
figure;subplot(231);
imshow(X)
title('原始图像')
X=double(X);
Y=Yucebianma(X);
XX=Yucejiema(Y);
subplot(232);imshow(mat2gray(Y));
title('预测误差图像')
```

```
e=double(X)-double(XX);
[m,n]=size(e);
erms=sqrt(sum(e(:).^2)/(m*n));
[h,x]=hist(X(:));
subplot(233);
bar(x,h,'k');
title('原图直方图')
[h,x]=hist(Y(:));
subplot(234);
bar(x,h,'k');
title('预测误差直方图')
XX=uint8(XX);
subplot(235);
imshow(XX);
title('解码图像')
```

无损预测编码

```
%% Yucebianma预测编码函数
% 一维无损预测编码压缩图像x， f为预测系数，如果f默认，则f=1，即为前值预测。
function Y=Yucebianma(x,f)
error(nargchk(1,2,nargin))
if nargin<2
   f=1;
end
x=double(x);
[m,n]=size(x);
p=zeros(m,n);
xs=x;
zc=zeros(m,1);
for j=1:length(f)
   xs=[zc xs(:,1:end-1)]; %end运算符(末尾)
   p=p+f(j)*xs;
end
Y=x-round(p);
end
```

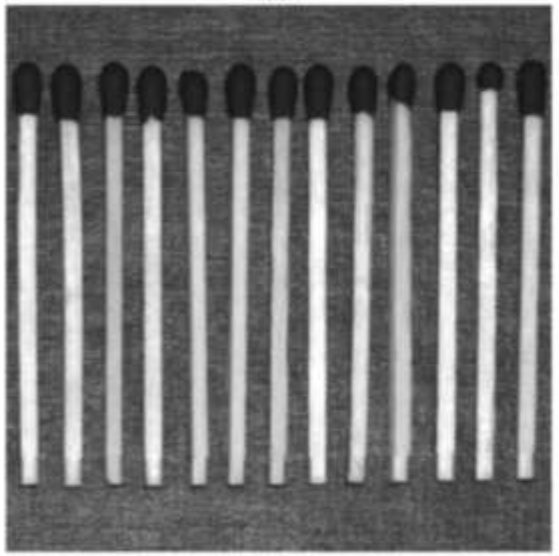
```
%% Yucejiema是解码程序，与编码程序用的是同一个预测器。
function x = Yucejiema(Y,f)
error(nargchk(1,2,nargin));
if nargin < 2
   f = 1;
end
f = f(end:-1:1);
[m,n] = size(Y);
order = length(f);
f = repmat(f,m,1);
x = zeros(m,n+order);
for j = 1:n
   jj = j + order;
   x(:,jj) = Y(:,j)+round(sum(f(:,order:-1:1).*x(:,(jj-1):-1:(jj-order)),2));
end
x = x(:,order+1:end);
end
```

无损预测编码

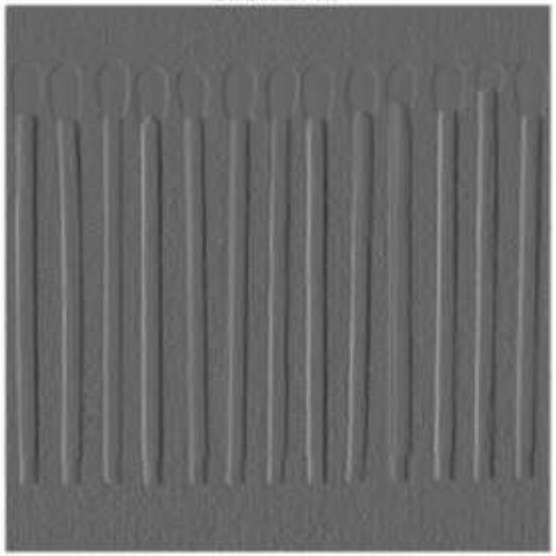
```
%无损预测
clc;clear all;
f = imread( 'matches-aligned.tif')
figure;
subplot(1,3,1);imshow(f),title('原图');
e = mat2lpc(f);% 一阶线性预测器（前像素预测器）：
% f'(x, y) = round [af(x, y) - 1)]
subplot(1,3,2);imshow(mat2gray(e));title('预测误差图像');% 显示预测误差图像
ntrop(e)
c = mat2huff(e);
% 对误差进行霍夫曼编码
cr = imratio(f, c);
%预测误差的直方图
g = lpc2mat(huff2mat(c));   % 解码
subplot(1,3,3);imshow(g,[]);title('恢复图像');% 显示预测误差图像
compare(f, g)   %
```

无损预测编码

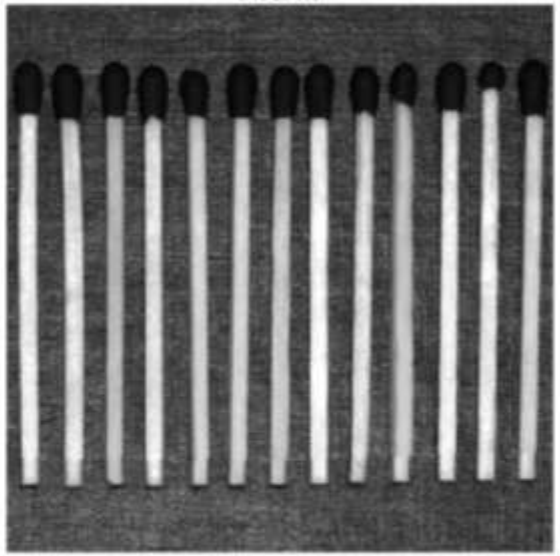
原图



预测误差图像



恢复图像



无损预测编码

- dpcmenco函数用于实现差分码调制编码；dpcmdeco函数用于实现差分码调制解码。
- indx=dpcmenco(sig,codebook,partition,predictor)：参数sig为输入信号，codebook为预测误差量化码本，partition为量化阈值，predictor为预测期的预测传递函数系数向量，返回参数indx为量化序号。
- [indx,quants] =dpcmenco(sig,codebook,partition,predictor)：返回参数quants为量化的预测误差。
- sig=dpcmdeco(indx,codebook,predictor)：返回参数为输出信号，indx为量化序号。
- [sig,quanterror] =dpcmdeco(indx,codebook,predictor)：参数quanterror为量化的预测误差。

无损预测编码

图像是二维信息，调用函数要转换为一维信号，预测编码利用的定以一维序列中相邻值应该也是空间上相邻的像素，一般采用Z形扫描，本例中仅将图像各列首尾相接。

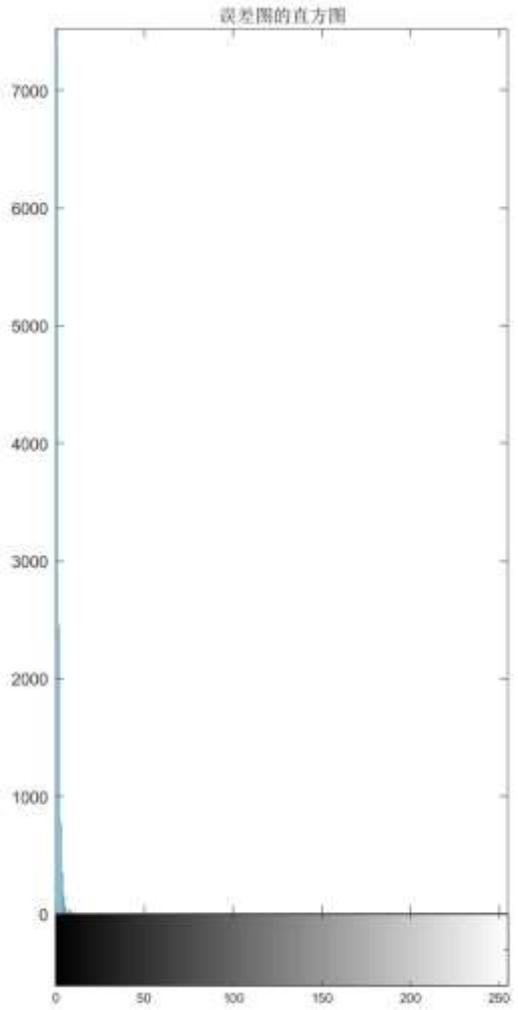
```
%使用matlab自带函数实现无损预测
clear,clc,close all;
Image= imread('lena.bmp');
[height,width]= size( Image);
sig=double(Image);
temp= flipud(sig);%现了矩阵X的上下翻转
sig = double(Image);
for i=2:2:width
    sig(:,i)= temp(:,i);
end
%将图像各列首尾相接形成一维信号
sig= reshape(sig,1,[ ]);
maxV=max(sig);
minV=min(sig);
step=10;
```

```
%设置初始码书
initcodebook= minV:step:maxV;
[predictor,codebook,partition]=dpcmopt(sig,2,initcodebook);%参数优化
encoded=dpcmenco(sig,codebook,partition,predictor);% DPCM 解码
decoded= dpcmdeco(encoded,codebook,predictor);
decodedI=uint8(reshape(round(decoded),height,width));%转换为二维图像
temp= flipud(decodedI);
for i=2:2:width
    decodedI(:,i)=temp(:,i);%调整偶数列上下像素顺序
end
diffI = abs(decodedI - Image);%原图与解码图的误差图
subplot(131),imshow(Image),title('原图');
subplot(132),imshow(decodedI),title('解码图');
subplot(133),imhist(diffI),title('误差图的直方图');
```

无损预测编码

原图

解码图



有损预测编码

有损压缩：

- 通过牺牲图像的准确率来达到提升压缩率的目的
- 如果能够容忍解压缩后的结果中有一定的误差，那么压缩率可以显著提高

有损压缩方法的压缩比：

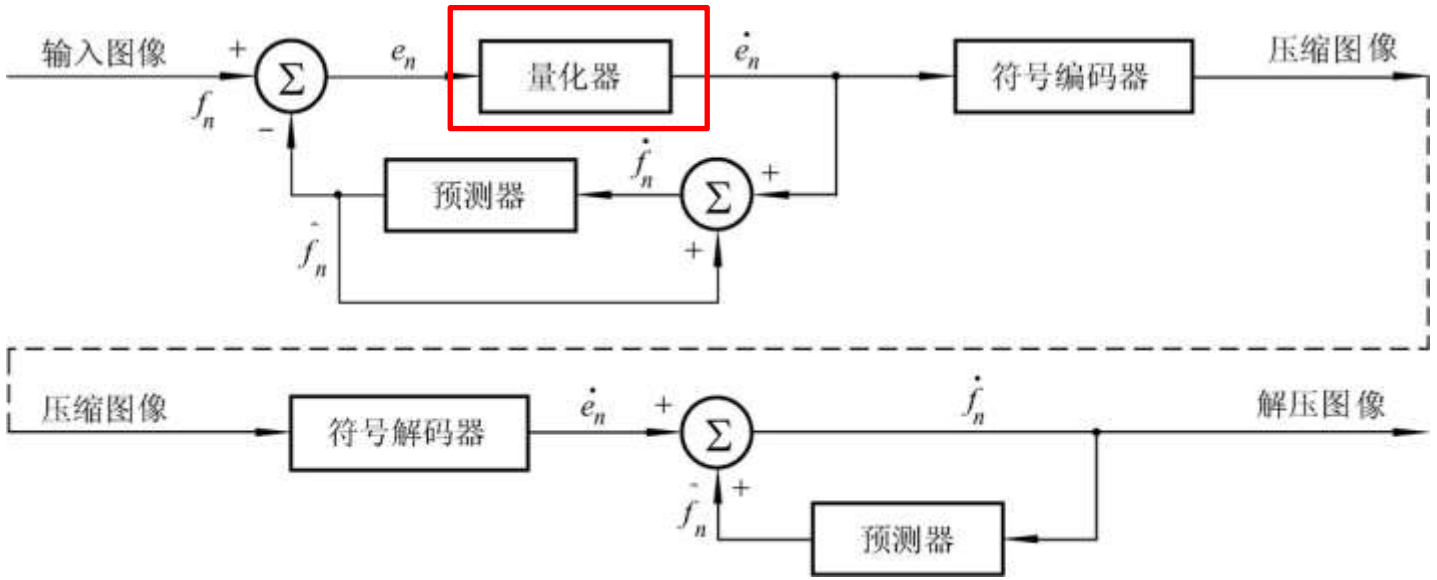
- 在图像压缩比大于30:1时，仍然能够重构图像
- 在图像压缩比为10:1到20:1时，重构图像与原图几乎没有差别
- 无损压缩的压缩比很少有能超过3:1的

有损预测编码

- 有损压缩增加了一个量化器（量化是不可逆的，有损压缩）
- 量化器基本思想：
  - ✓ 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

有损预测编码

- 常见的有损预测编码：增量调制DM、差分脉冲调制DPCM、自适应差分脉冲调制ADPCM



有损预测编码

- 输入序列：$f_n(n = 1, 2, \ldots)$
- 量化误差：$\dot{e}_n = Q(e_n)$
- 预测输入：$\dot{f}_n = \dot{e}_n + \hat{f}_n$
- 误差编码：$\dot{e}_n = Q(\dot{f}_n - \hat{f}_n) \approx Q(f_n - \hat{f}_n)$
- 解压序列：$\dot{f}_n = \dot{e}_n + \hat{f}_n$
- 编码误差：$f_n - \dot{f}_n$

目录

离散余弦变换

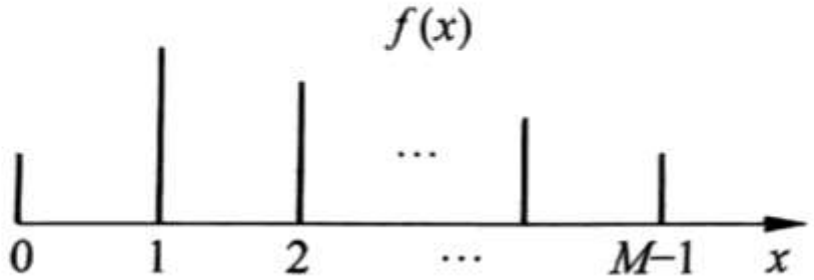
- **傅立叶变换存在的缺陷**：它的参数是复数，在数据描述上相当于实数的两倍，导致计算量大。
- 希望有一种能够达到相同功能但**数据量又不大**的变换。
- 离散余弦变换（Discrete Cosine Transform, DCT）是傅里叶变换的一种特殊情况，实际上是利用了傅立叶变换的实数部分构成的变换。
- 在傅里叶变换过程中，如果被展开的函数是实偶函数，那么其傅里叶变换中只包含余弦项。 基于这一特点，人们提出了离散余弦变换。
- 离散余弦变换首先将图像函数变换成**偶函数**形式，DCT变换核是实数的余弦函数，因而DCT的计算速度比DFT快得多。

离散余弦变换

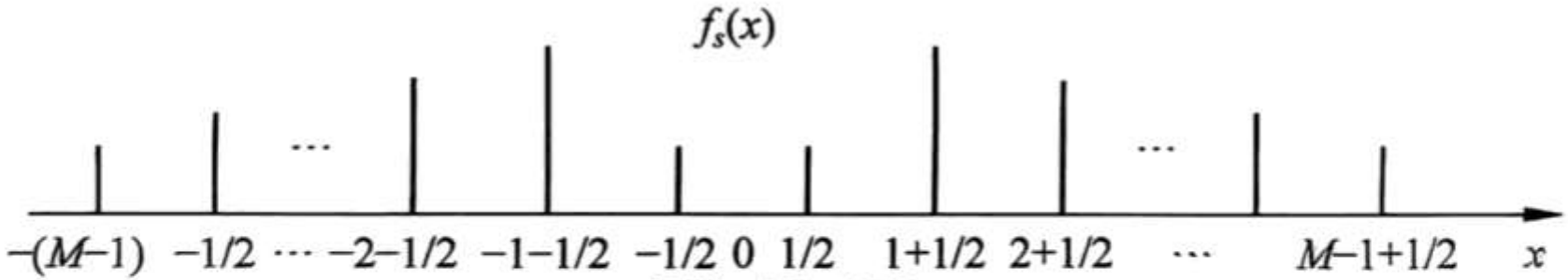
- 离散余弦变换通过一组**频率和幅度不同的余弦函数之和**来近似地表示图像的方式，实际上是傅里叶变换的实数部分。
- 离散余弦变换有一个重要的性质，即对于一幅图像而言，其大部分可视化信息都集中在少数的变换系数。因此，离散余弦变换经常用于图像压缩。 例如国际压缩标准JPEG就采用了离散余弦变换。
- 离散余弦变换是一种**可分离的正交变换**，并且是对称的。**DCT的变换阵的基向量能很好地描述人类语音信号和图像信号的相关特征。**

离散余弦变换

设 $f(x)$为一实数离散序列，且 $x=0,1,2,\cdots,M-1$，如图 7.1(a)所示。将其延拓为偶对称序列 $f_s(x)$，如图 7.1(b)所示。



(a) 实数离散序列 $f(x)$



(b) 偶数对称序列 $f_s(x)$

图　　以 $x=0$ 为中心的偶对称序列

离散余弦变换

则有

$$f_s(x)=\begin{cases} f(x-\frac{1}{2}) & \text{对于} x=\frac{1}{2},1+\frac{1}{2},\cdots,(M-1)+\frac{1}{2} \\ f(-x-\frac{1}{2}) & \text{对于} x=-\frac{1}{2},\ -1-\frac{1}{2},\ \cdots,\ -(M-1)-\frac{1}{2} \end{cases}$$

显然， $f_s(x)$ 是以x=0为中心的偶对称函数。

离散余弦变换

对$f_s(x)$求2M个点的一维离散傅里叶变换(DFT)，有

$$F_s(u)=\frac{1}{\sqrt{2M}}\sum_{x=-(M-1)-1/2}^{M-1+1/2} f_s(x)\exp\left(-\frac{j2\pi xu}{2M}\right)$$

$$=\frac{1}{\sqrt{2M}}\sum_{x=-(M-1)-1/2}^{-1/2} f_s(x)exp\left(-\frac{j2\pi xu}{2M}\right)+\frac{1}{\sqrt{2M}}\sum_{x=1/2}^{M-1+1/2} f_s(x)exp\left(-\frac{j2\pi xu}{2M}\right)$$

用y=-x对上式的第1项作变量代换，并仍用x表示可得

$$F_s(u)=\frac{1}{\sqrt{2M}}\sum_{x=1/2}^{M-1+1/2} f_s(x)\,exp\left(\frac{j\pi xu}{M}\right)+\frac{1}{\sqrt{2M}}\sum_{x=1/2}^{M-1+1/2} f_s(x)\,exp\left(-\frac{j\pi xu}{M}\right)$$

离散余弦变换

考虑到$f_s(x)$为偶函数，即$f_s(x)=f_s(-x)$，并对式$F_s(u)$运用欧拉公式可得

$$F_s(u)=\frac{1}{\sqrt{2M}}\sum_{x=1/2}^{M-1+1/2} f_s(x)\left(\cos\left(\frac{\pi xu}{M}\right)+i\cdot\sin\left(\frac{\pi xu}{M}\right)\right)$$

$$+\frac{1}{\sqrt{2M}}\sum_{x=1/2}^{M-1+1/2} f_s(x)\left(\cos\left(\frac{\pi xu}{M}\right)-i\cdot\sin\left(\frac{\pi xu}{M}\right)\right)$$

$$=\frac{2}{\sqrt{2M}}\sum_{x=1/2}^{M-1+1/2} f_s(x)\cos\left(\frac{\pi xu}{M}\right)$$

离散余弦变换

由定义：

$$f_s(x)=\begin{cases} f(x-\frac{1}{2}) & \text{对于} x=\frac{1}{2},1+\frac{1}{2},\cdots,(M-1)+\frac{1}{2} \\ f(-x-\frac{1}{2}) & \text{对于} x=-\frac{1}{2},\ -1-\frac{1}{2},\ \cdots,\ -(M-1)-\frac{1}{2} \end{cases}$$

即 $$F_s(u)=\sqrt{\frac{2}{M}}\sum_{x=1/2}^{M-1+1/2} f\left(x-\frac{1}{2}\right)\cos\left(\frac{\pi x u}{M}\right)$$

离散余弦变换

$$F_s(u) = \sqrt{\frac{2}{M}} \sum_{x=1/2}^{M} f\left(x - \frac{1}{2}\right) \cos\left(\frac{\pi x u}{M}\right)$$

用y=x-1/2对上式作变量代换，即设 y=x-1/2，则 x=y+1/2，且 x=(2y+1)/2

这样，当 x=1/2 时，y=0；当 x=M-1+1/2 时，y=M-1

则有：$F_s(u) = \sqrt{\frac{2}{M}} \sum_{y=0}^{M-1} f\,(y) \cos\left(\frac{\pi(2y+1)u}{2M}\right)$

再用x代替y得：

$$F_s(u) = \sqrt{\frac{2}{M}} \sum_{x=0}^{M-1} f\,(x) \cos\left(\frac{\pi(2x+1)u}{2M}\right)$$

离散余弦变换

$$F_s(u) = \sqrt{\frac{2}{M}} \sum_{x=0}^{M-1} f\,(x) \cos\left(\frac{\pi(2x+1)u}{2M}\right)$$

对上式乘以$K(u)$，以便将其表示成归一正交矩阵形式，可得$f(x)$的一维DCT为：

$$F(u) = \sqrt{\frac{2}{M}} K(u) \sum_{x=0}^{M-1} f\,(x) \cos\left(\frac{\pi(2x+1)u}{2M}\right)$$

$$K(u) = \begin{cases} \dfrac{1}{\sqrt{2}} & u = 0 \\ 1 & u = 1,2,\cdots,M-1 \end{cases}$$

离散余弦变换

将式$K(u)$代入式$F(u)$，可得到一种更直观地一维正DCT表示形式为：

$$F(0)=\frac{1}{\sqrt{M}}\sum_{x=0}^{M-1}f\,(x)$$

$$F(u)=\sqrt{\frac{2}{M}}\sum_{x=0}^{M-1}f\,(x)\cos\left(\frac{\pi(2x+1)u}{2M}\right)$$

其中： $F(u)$是第u个余弦变换系数，u是广义频率变量，u=1,2,…,M-1，f(x)是时域上的M点实序列，且x=0,1,2,…,M-1。

一维离散余弦变换的正变换核为

$$H(u,x)=\sqrt{\frac{2}{M}}K(u)\cdot\cos\left[\frac{\pi(2x+1)u}{2M}\right]$$

离散余弦变换

一维DCT

对于有限长数字序列$f(x)$，$x = 0,1,\cdots N-1$

✓ 一维**DCT**定义：$F(u) = C(u)\sqrt{\frac{2}{N}}\sum_{x=0}^{N-1} f(x)\cos\frac{(2x+1)u\pi}{2N}$　　$u = 0,1,\cdots,N-1$

✓ 一维**IDCT**定义：$f(x) = \sqrt{\frac{2}{N}}\sum_{u=0}^{N-1} C(u)F(u)\cos\frac{(2x+1)u\pi}{2N}$　　$x = 0,1,\cdots,N-1$

$$C(u) = \begin{cases} 1/\sqrt{2} & u = 0 \\ 1 & u = 1,2,\ldots,N-1 \end{cases}$$

离散余弦变换

二维DCT变换对

$$F(u,v)=\frac{2}{\sqrt{MN}}C(u)C(v)\sum_{x=0}^{M-1}\sum_{v=0}^{N-1}f\,(x,y)\cos\left[\frac{\pi(2x+1)u}{2M}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

$$f(x,y)=\frac{2}{\sqrt{MN}}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}C\,(u)C(v)F(u,v)\cos\left[\frac{\pi(2x+1)u}{2M}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right]$$

$$\begin{matrix}x,u=0,1,2,\cdots,M-1\\ y,v=0,1,2,\cdots,N-1\end{matrix}\quad C(u),C(v)=\begin{cases}1/\sqrt{2} & u,v=0\\ 1 & u,v=1,2,\ldots,N-1\end{cases}$$

IDCT变换的特点：

✓ 无虚数部分

✓ 正变换核与逆变换核相同，且是可分离的(二维 DCT可用两次一维 DCT 来完成)

离散余弦变换

二维离散余弦变换能够逐次应用一维离散余弦变换进行计算

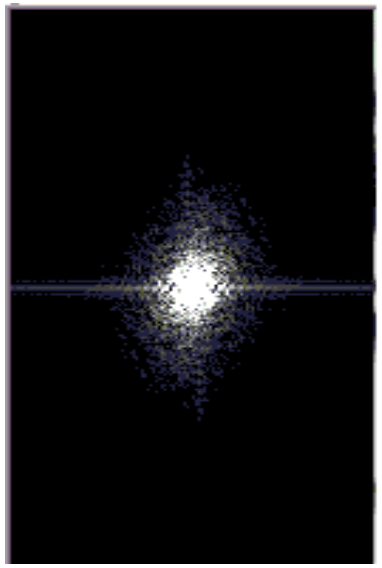
$$F(u,v)=\boldsymbol{c}(u)\sum_{x=0}^{N-1}\left\{\boldsymbol{c}(v)\sum_{y=0}^{N-1}f\,(x,y)\cos\left[\frac{(2y+1)v\pi}{2N}\right]\right\}\cos\left[\frac{(2x+1)u\pi}{2N}\right]$$

一维离散余弦变换实现步骤如下所示

$$F(u)=c(u)\sum_{x=0}^{N-1}f\,(x)\cos\frac{(2x+1)u\pi}{2N}=c(u)\mathrm{Re}\left\{\sum_{x=0}^{N-1}f\,(x)\mathrm{e}^{-\mathrm{j}\frac{(2x+1)u\pi}{2N}}\right\}$$

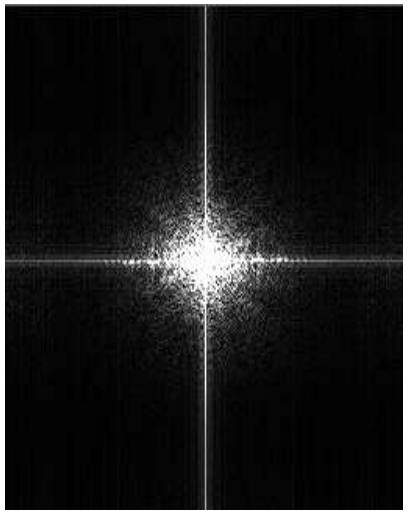
式中，Re{·}表示取复数的实部。

离散余弦变换



细节较少图片的傅立叶变换和离散余弦变换

二维离散余弦变换



细节中等图片的傅立叶变换和离散余弦变换

二维离散余弦变换

```
%图像压缩
I = imread('cameraman.tif');
I = im2double(I);
%
T = dctmtx(8);%返回N×N的DCT变换矩阵
dct = @(block_struct) T * block_struct.data * T';
B = blockproc(I,[8 8],dct);
%丢弃每个数据块中 64 个 DCT 系数的大部分系数，仅保留 10 个。

mask = [1   1   1   1   0   0   0   0
        1   1   1   0   0   0   0   0
        1   1   0   0   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
        0   0   0   0   0   0   0   0];
```

二维离散余弦变换

```
B2 = blockproc(B,[8 8],@(block_struct) mask .* block_struct.data);
%使用每个数据块的二维逆 DCT 重构图像。
invdct = @(block_struct) T' * block_struct.data * T;
I2 = blockproc(B2,[8 8],invdct);
%并排显示原始图像和重构图像。尽管几乎 85% 的 DCT 系数被丢弃，导致重构
图像的质量有所下降，但它仍是清晰可辨的。
subplot(1,2,1),imshow(I);title('原始图像');
subplot(1,2,2),imshow(I2);title('恢复图像');
```

二维离散余弦变换

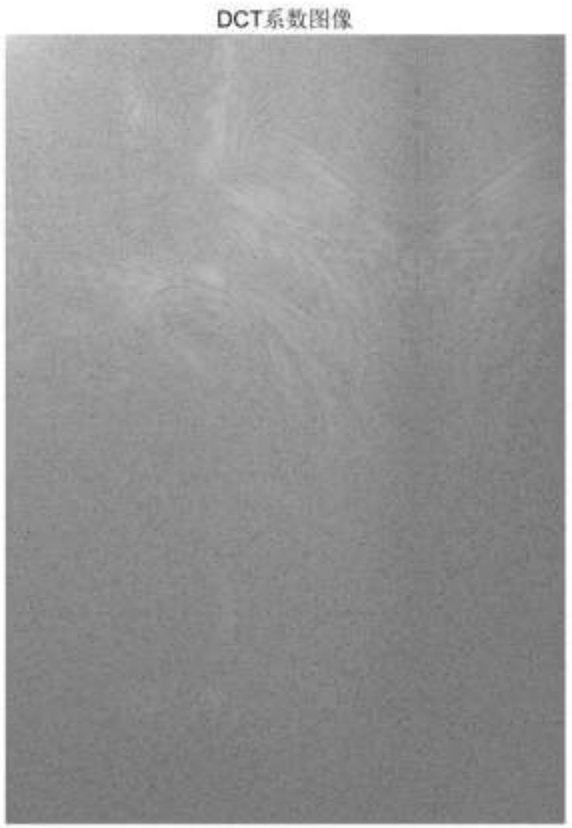
原始图像

恢复图像

二维离散余弦变换

```
%DCT变换Matlab程序
clc; clear all; close all;
img0=imread('barbara.tif');
subplot(1,3,1);  imshow(img0);  title('原图像');
[h, w, color]=size(img0);
if (color==3)        % 如果输入图像是彩色图像, 将其转换成灰度图像
   f_gray=rgb2gray(img0);
else
   f_gray=img0;
end
dct_coef=dct2(f_gray);    % 计算DCT系数
subplot(1,3,2); imshow(log(abs(dct_coef)),[]); title('DCT系数图像');
dct_coef(abs(dct_coef)<0.1)=0;   % 将DCT系数矩阵中小于0.1的值置为0
f_dct=idct2(dct_coef);                % 进行DCT逆变换重建图像
subplot(1,3,3); imshow(f_dct,[]); title('DCT变换解压缩图像');
```

二维离散余弦变换



变换域压缩编码

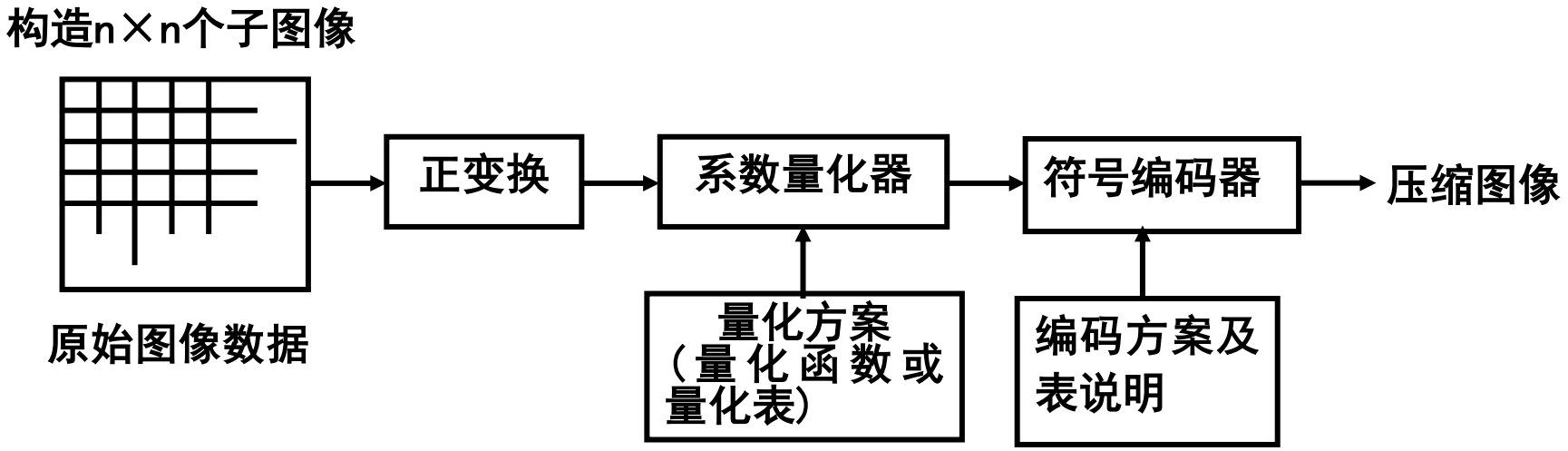
变换域编码指以某种可逆的正交变换将给定的图像变换到另一个数据域（如频域），从而利用新的数据域的特点，用一组非相关数据（系数）来表示原图像，并以此来去除或减小图像在空间域中的相关性，将尽可能多的信息集中到尽可能少的变换系数上，使多数系数只携带尽可能少的信息，实现用较少的数据表示较大的图像数据信息，进而达到压缩数据的目的。

通过变换将空间域图像像素映射为变换域变换系数，图像能量在空间域的分散分布变为变换域的能量相对集中分布，利用系数分布特点和人类的感觉特性，对系数进行量化、编码，舍弃能量很小的系数，达到压缩的目的。

变换域压缩编码

- 变换域压缩编码是一种有损编码。
- 优点：压缩比高，视觉效果好。
- 压缩并不是在变换域变换步骤取得的，而是在量化变换域系数时取得的。变换的能量集中特性会影响压缩效果。

变换域压缩编码



- 图像分解（划分子块）：构造子图像减少变换的计算复杂度。
- 图像变换：解除每个子图像内部像素之间的相关性，或者说将尽可能多的信息集中到尽可能少的变换系数上。
- 系数量化：要通过保留较大系数，舍弃较小的系数。

变换域压缩编码

（1）将待编码的N×N的图像分解成$(N/n)^2$个大小为n×n的子图像。通常选取的子图像大小为8×8或16×16，即n等于 8 或16。

（2）对每个子图像进行正交变换（如DCT变换等），得到各子图像的变换系数。这一步的实质是把空间域表示的图像转换成频率域表示的图像。

（3）对变换系数进行量化。

（4）使用霍夫曼变长编码或游程编码等无损编码器对量化的系数进行编码，得到压缩后的图像（数据）。

变换域压缩编码

子图像尺寸选择

子图像的大小与变换编码的误差和变换所需的计算量等有关。在大多数应用中，将图像进一步分割成子图像块要求满足以下两个条件：

- ✓ 一是相邻子图像块之间的相关性（冗余）要减少到某种可接受的程度；
- ✓ 二是子图的长和宽应是2的整数次幂，这个条件主要是为了简化对子图的计算。一般来说，当子图的尺寸增大时所计入的相关像素就越多，总的均方差性能改善可能越多。然而，从统计学上平均来说一列相似的像素通常会持续15~20个像素那么长，其后像素间的相关性就开始下降。按照子图的长和宽应是2的整数次幂的要求，即当n>16时，对性能的进一步改善作用不大，所以最常采用的子图像尺寸为8×8和16×16的块，JPEG编码选用的就是8×8的块。

变换域压缩编码

- 变换编码是一种有失真编码，失真表现在：高频信息丢失或减少导致可分辨性下降，在图像灰度平坦区有颗粒噪声出现，产生方块效应。

变换域压缩编码

编码常用两种思路：区域编码和阈值编码。

■ 区域编码

✓ 变换系数集中在低频区域，可对该区域的变换系数进行量化、编码、传输，对高频区域既不编码又不传输即可达到压缩目的，这称为变换区域编码。

✓ 即保留系数方阵中左上角区域的若干系数，而将其余系数置为零。这种保留和置0也即量化过程。

✓ 区域编码压缩比可达到5:1，缺点是高频分量被丢弃，图像可视分辨率下降。

变换域压缩编码

- 典型的区域编码量化模板：

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

（a）只保留左上角的6个系数

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

（b）只保留左上角的15个系数

变换域压缩编码

■区域编码

实现区域编码要求的保留左上角特定区域系数的方法：

用区域模板中的各个数据（1或0）乘以变换系数，也即乘以将各子图像进行变换后所得到的变换系数。

176	172	48	107	158	179	171	153
172	174	163	134	167	168	148	148
171	176	167	161	169	159	147	131
176	177	170	169	163	124	84	96
185	179	179	159	80	41	58	74
185	184	143	72	66	75	75	69
182	131	90	100	116	113	114	104
113	83	101	114	123	116	112	130

176	172	48	0	0	0	0	0
172	174	0	0	0	0	0	0
171	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	0

图 区域编码均匀量化方法示例

变换域压缩编码

■区域编码

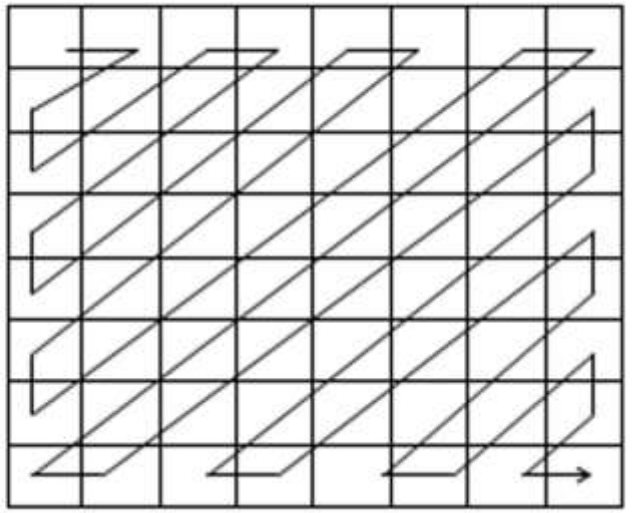
- 一般将上图所示的仅用区域模板进行系数量化的方法称为均匀量化。
- 变换系数量化（量化器——量化模板）的作用：
  - ✓ 减少视觉心理冗余。
  - ✓ 显然由于量化，所以区域编码是有损压缩编码。

变换域压缩编码

■ 区域编码

在进行均匀量化或非均匀量化后，还要对量化后的系数根据下图所示的顺序重新编排成一个具有n^2个元素的$1\times n^2$的系数序列（矢量）。也即，对量化后的系数进行矢量排序。利用哈夫曼变长编码或游程编码等无损编码器方式，对量化得到的一维矢量形式的量化系数进行编码。

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



变换域压缩编码

```
%DCT区域变换编码
clc;clear all;%清除命令窗口的内容,清除工作空间中的所有变量
close all;%关闭所有的figure窗口
img=imread('lena1.bmp');
subplot(2,2,1),imshow(img);title('(a)原图像');
f = double( img);
h= zeros(8,8);
%建立一个8x8的全零值图像块
%计算变换核
h,h(u,v)=sqrt(2/N)*k(u)*cos[(pi*(2x+1)* u)/2N]
```

```
for u=1:8%公式中u和x为0:7，这里程序控制变量u和x为1:8
   for x=1:8%程序通过u-1和x-1实现公式的0:7
      if u== 1
         h(u,x)= sqrt(1/8);
      else
         h(u,x)=sqrt(2/8)*cos((pi*(2*(x-1)+1)*(u-1))/16);
      end
   end
end
%按8x8的分块计算图像f的DCT变换，'P1*x*P2'为阵列计算模式
F= blkproc(f,[8 8],'P1 * x* P2',h, h');
subplot(2,2,2),imshow(uint8(F));title('(b)8x8分块DCT变换的DCT系数频谱');%显示图像f按8x8分块进行
```

变换域压缩编码

```
DCT变换的DCT系数
%构建8x8矩阵的上三角1值矩阵(28个1;对角线和下三角值均为0,36个0)
for u= 1:8%即构建区位编码模板
   for x=1:8
      if u<= 8-x
         M(x,u)=1;
      else
         M(x,u)= 0;
      end
   end
end
%用区域模板M保留每个8x8块的左上角28个系数,舍弃其余36个高频系数
F= blkproc(F,[8 8],'P1.*x',M);%%P1=M，计算F=F*.M
subplot(2,2,3),
imshow(uint8(F));title('(c)保留8x8块28个系数的DCT系数频谱');%显示每个8x8分块DCT系数舍弃高频系数后
的DCT系数
%按8x8的分块计算图像f的DCT反变换
F=blkproc(F,[8 8],'P1 * x* P2',h',h);
subplot(2,2,4),
imshow(uint8(F));
title('(d)区域编码压缩后的解压缩图像');
cc=double(uint8(F));
diff=imsubtract(f,cc);%原图与重构图的差
maxdiff=max(diff(:))%差的最大值
```

变换域压缩编码

maxdiff = 54

变换域压缩编码

阈值编码

阈值编码通过事先设定一个阈值，只对其变换系数的幅值大于此阈值的编码；在保留低频成分的同时，选择性地保留了高频成分，重建图像质量得到改善；但需要同时对系数所处的位置编码，复杂度提高。

变换域压缩编码

■ 变换解码

解码是编码的逆过程。变换解码过程为：

（1）利用符号解码器对压缩的图像数据进行解码，得到用量化系数表示的图像数据。

（2）用与编码时相同的量化函数或量化值表对用量化系数表示的图像数据进行逆量化，得到每个子图像的变换系数。

（3）对逆量化得到的每个子图像的变换系数进行反向正交变换（如反向DCT变换等），得到$(N/n)^2$个大小为$n\times n$的子图像。

（4）将$(N/n)^2$个大小为$n\times n$的子图像重构成一个$N\times N$的图像。

变换域压缩编码

■ 变换解码

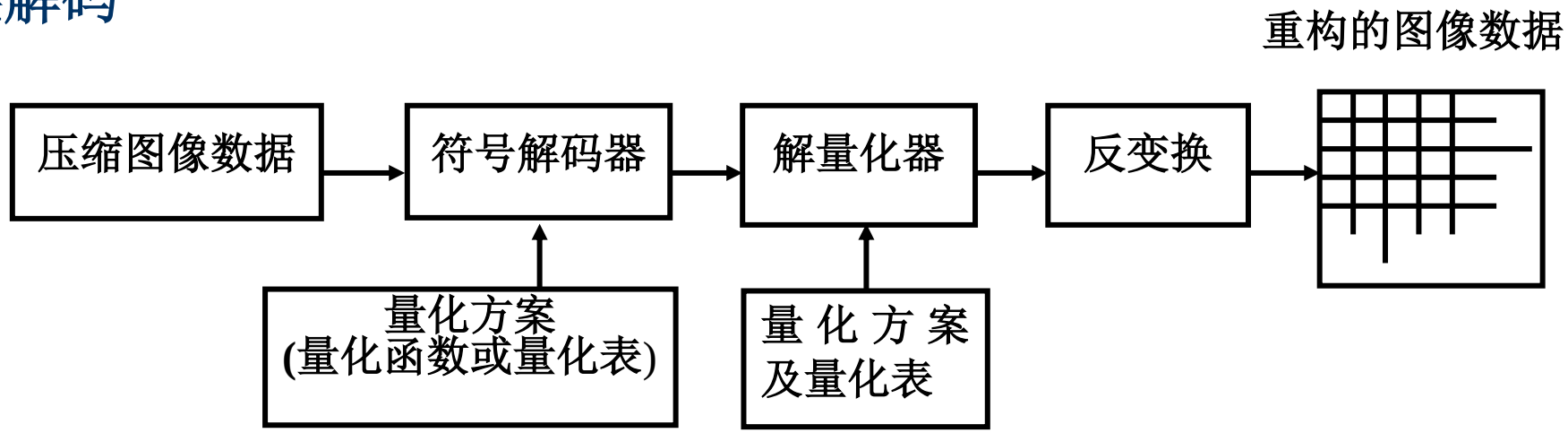


图 变换解码系统框图

变换域压缩编码

■ 变换解码

在变换编码中，变换本身并不产生信息压缩作用，而只是去除原图像中的相关性。只有通过对系数的量化和高效的符号编码才能产生对图像信息的压缩作用。

常见图像压缩标准

JPEG

- 静止图像压缩标准JPEG：Joint Picture Expert Group，联合图片专家组
- JPEG是一个应用广泛的**静态图像**数据压缩标准，其中包含多种压缩算法，并考虑了人眼的视觉特性，在量化和无损压缩编码方面综合权衡，可以达到较大的压缩比（25:1 以上）。
- JPEG既适用于灰度图像也适用于彩色图像，即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。
- JPEG算法**与彩色空间无关**，因此它可以压缩来自不同彩色空间的数据，如RGB，YCbCr和CMYK。

JPEG

- JPEG定义了三种不同的编码系统：

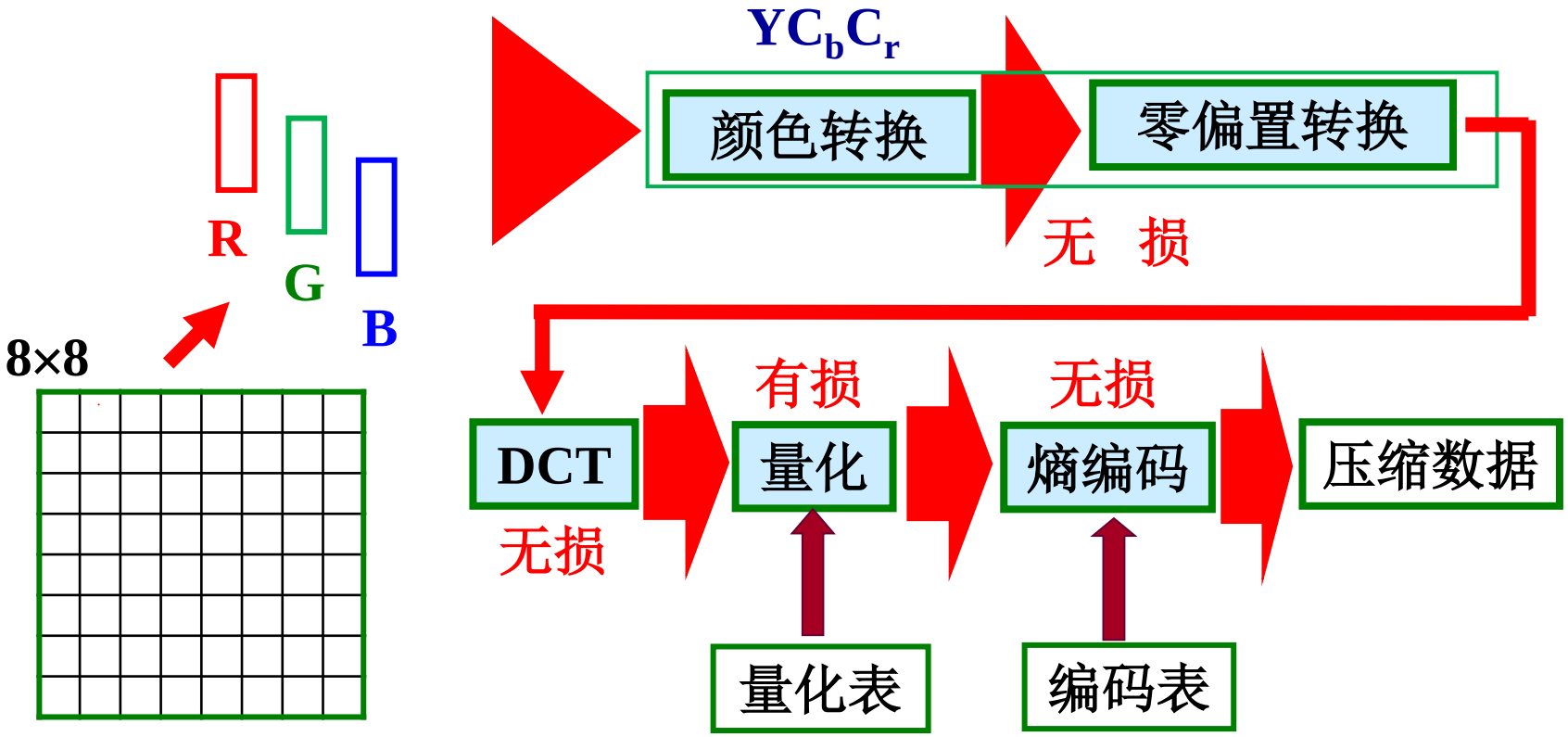
（1）基于DCT的有损编码基本系统：足够应付大多数压缩方面的应用

（2）扩展编码系统：面向的是更大规模的压缩、更高的精确性或逐渐递增的重构应用系统；

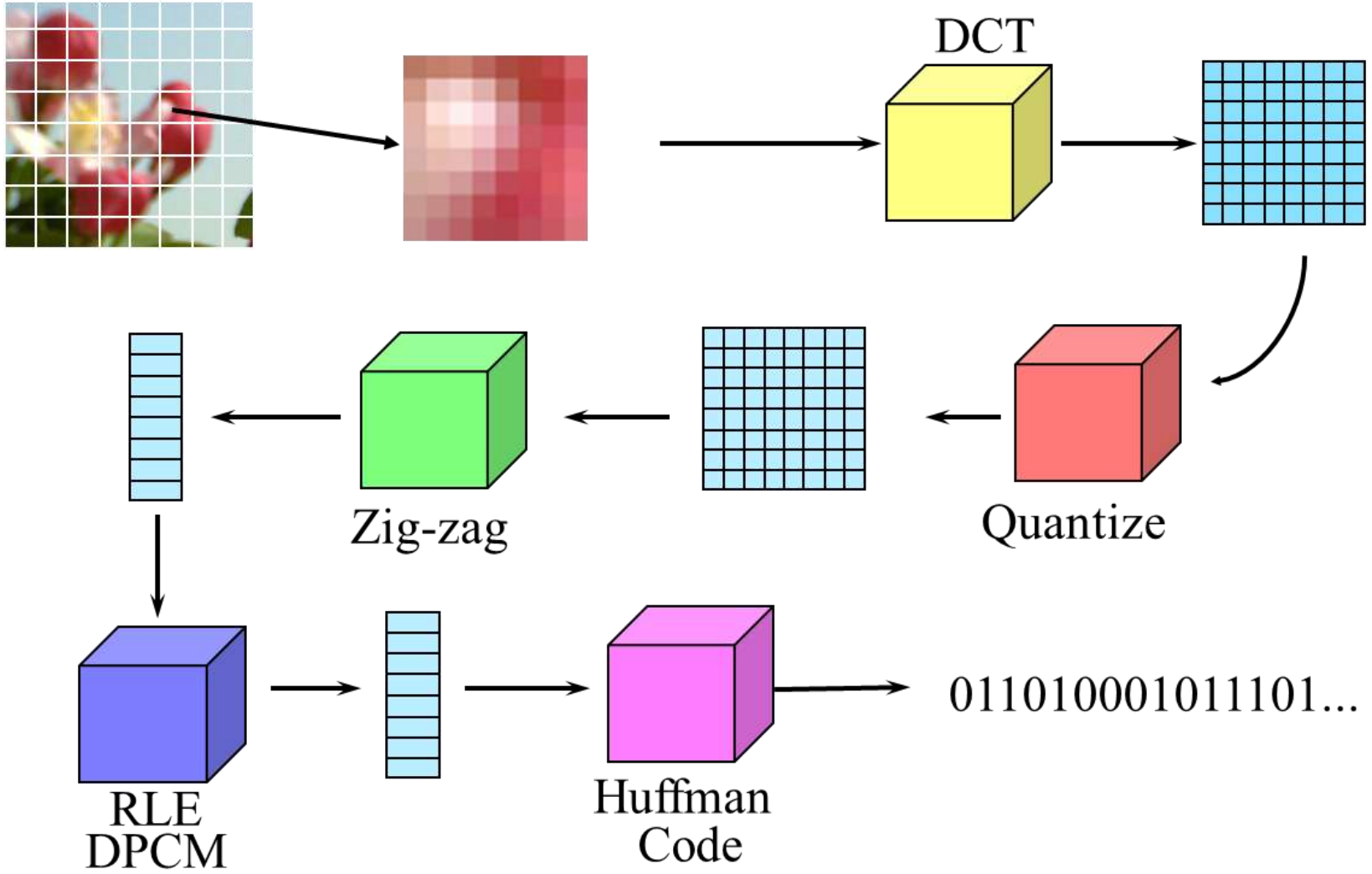
（3）基于差分脉冲编码调制（DPCM, Differential Pulse Code Modulation）的独立无损编码系统；DPCM是一种预测编码技术，通过对相邻采样点之间的差值（而非原始信号本身）进行量化与编码，从而减少数据冗余，实现信号的压缩。

JPEG

- 编码流程：



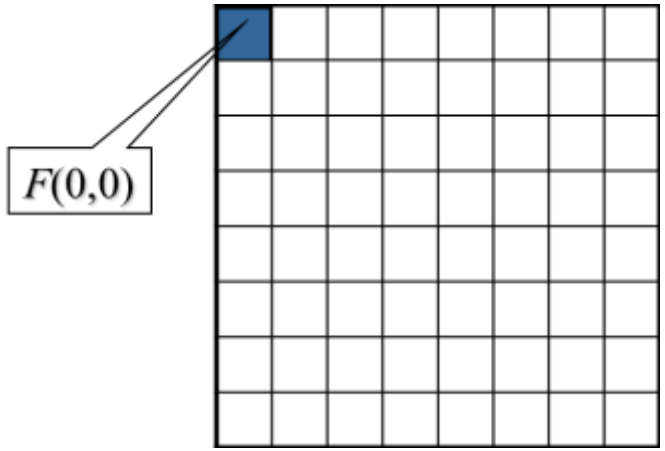
JPEG


DCT
Quantize
Zig-zag
RLE
DPCM
Huffman
Code
011010001011101...

JPEG

1、图像分解：将图像分成 8×8 的数据块。

2、DCT变换：每个图像数据块经过DCT变换后，将数据块从空间域变换到频率域，输出8×8的DCT变换系数T(u, v)。其中F(0, 0)称为直流系数DC（能量集中），其余的63个系数称为交流系数AC。



JPEG

3、量化：JPEG基本系统使用Huffman编码对DCT量化系数进行熵编码，进一步压缩码率。

根据人眼的视觉特性，

- 人眼对亮度信号比对色度信号更加敏感。
  - √ JPEG标准中使用两个量化表：亮度/色度量化表。
- 人眼对低频信号比对高频信号更加敏感。
  - √ 高频编码分配较少比特数。
  - √量化的原则是低频部分用小的值量化，高频部分用大的值量化，量化的结果将会在高频部分出现大量的0。

JPEG

3、量化：用DCT变换后的系数T(u, v)除以量化表Q(u, v)

通过标定量化表，可以改变压缩比和重建图像质量。

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

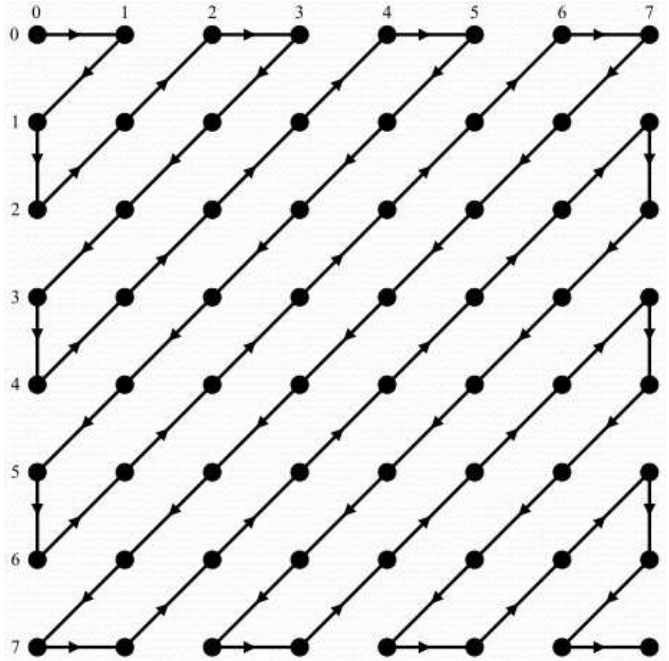
亮度量化表

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

色差量化表

JPEG

4、重排数据： 量化后$\hat{T}(u,v)$是个稀疏矩阵。用Zigzag模式将$\hat{T}(u,v)$重新排列变成一维数列，即将8×8的矩阵变成64×1的向量，数值较大的系数放在向量的顶部。重排数据的目的是为了产生长的零行程。



JPEG

5、编码：对于DC系数和AC系数采用不同的编码方式。

- DC系数编码：相邻8×8图像块的DC系数值变化不大，JPEG算法采用无失真的差分脉冲调制编码DPCM技术，对相邻图像块之间量化DC系数的差值进行编码。
- AC系数编码：量化的AC系数中包含有许多连续的0系数，因此采用行程编码（RLE）进行编码。
- 熵编码：JPEG对DPCM编码后的直流DC码字和RLE编码后的交流AC码字，再进行熵编码（Huffman编码，算术编码等）作进一步压缩。

JPEG2000

- JPEG2000是JPEG的升级版。
- 核心：采用离散小波变换DWT取代了DCT。
- 高压缩比：JPEG2000压缩率比JPEG高约30%左右，重建图无方块效应，同时支持有损和无损压缩。
- 速度快：DWT可一次变换整幅图像，速度比FFT快一个数量级。
- 多分辨率编码：利用DWT具有平移和缩放的数学显微镜的功能进行多分辨率编码，适应于不同分辨率的图像I/O设备和不同传输速率的通信系统。
- 其它特点：渐进传输、感兴趣区域（ROI）编码（交互式压缩）

JPEG2000

与JPEG的比较：相同压缩比，图像质量更好。相同图像质量，压缩比更高。

MPEG

- 序列图像压缩标准MPEG：Moving Picture Expert Group，运动图像专家组，它是对活动的视频图像压缩的国际标准的简称。
- 该专家组成立于1988年，它的工作不仅局限于活动图像编码，还将伴音和图像的压缩联系在一起，并且根据不同的应用场合，定义了不同的标准。
- **MPEG-1**：1993年8月正式通过，全称为“适用于约1.5Mbit/s以下数字存储媒体的运动图像及伴音的编码”。这里所指的数字存储媒体包括CD-ROM、DAT、硬盘、可写光盘等，同时利用该标准也可以在ISDN或局域网中进行远程通信。

MPEG

- **MPEG-2**：

1994年11月发布的“活动图像及伴音通用编码”标准，该标准可以应用于2.048Mbit/s ~ 20Mbit/s的各种速率和各种分辨率的应用场合之中，如多媒体计算机、多媒体数据库、多媒体通信、常规数字电视、高清晰度电视以及交互电视等。

MPEG

■ MPEG-4：

1999年1月公布V1.0版本，同年12月公布了V2.0版本。该标准主要应用于**超低速系统**之中，例如多媒体Internet、视频会议和视频电视等个人通信、交互式视频游戏和多媒体邮件、基于网络的数据业务、光盘等交互式存储媒体、远程视频监视及无线多媒体通信。特别是它能够满足基于内容的访问和检索的多媒体应用，且其编码系统是开放的，可随时加入新的有效算法模块。

MPEG

- **MPEG-7**：2000年11月颁布，“多媒体内容描述接口”的标准。定义该标准的目的是制定出一系列的标准描述符来描述各种媒体信息。这种描述与多媒体信息的内容有关，便于用户进行基于内容和对象的视听信息的快速搜索。 MPEG－7与其他MPEG标准的不同之处在于它提供了与内容有关的描述符，并不包括具体的视音频压缩算法，而且还未形成与内容提交有关的所有标准的总框架。
- **MPEG-21**：全称为“多媒体框架”。该标准的目的在于为多媒体用户提供透明而有效的电子交易和使用环境。

THE END

Thank You.