数字图像处理

武汉理工大学　信息学院

第7章 图像压缩编码（Image Compression Coding Technology）

7.1 概述（Introduction）

7.2 无失真图像压缩编码（Lossless image compression）

7.3 有限失真图像压缩编码（Lossy image compression）

7.4 图像编码新技术（New Image Compression Technology）

7.5 图像压缩技术标准（Image Compression Standards）

为什么要对图像进行压缩

举例1：对于电视画面的分辨率640*480的彩色图像，每秒30帧，则一秒钟的数据量为：

640*480*24*30=221.12M ，1张CD可存640M，如果不进行压缩，1张CD则仅可以存放2.89秒的数据.

举例2：目前的WWW互联网包含大量的图像信息，如果图像信息的数据量太大，会使本来就已经非常紧张的网络带宽变得更加不堪重负（World Wide Web变成了World Wide Wait）

7.1.1、图像的信息量与信息熵（Information Content and Entropy）

1. 信息量

设信息源X可发出的消息符号集合为 $A=\{a_i \mid i=1,2,\cdots,m\}$ 并设X发出符号 a_i 的概率为 $p(a_i)$，则定义符号出现的自信息量为:

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

通常,上式中的对数取2为底，这时定义的信息量单位为“比特”(bit) 。

7.1.1、图像的信息量与信息熵（Information Content and Entropy）

2. 信息熵

对信息源X的各符号的自信息量取统计平均，可得每个符号的平均自信息量为：

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

这个平均自信息量$H(X)$称为信息源X的熵（$entropy$），单位为bit/符号，通常也称为X的零阶熵。由信息论的基本概念可以知道，零阶熵是无记忆信息源（在无失真编码时）所需数码率的下界。

7.1.2、图像数据冗余 (Image data redundancy)

- 通常一副图像中的各点像素点之间存在一定的相关性。特别是在活动图像中，由于两幅相邻图像之间的时间间隔很短，因此这两幅图像信息中包含了大量的相关信息。这些就是图像信息中的冗余。

7.1.2、图像数据冗余 (Image data redundancy)

1. 空间冗余

图7.2是一幅图像，其中心部分为一个灰色的方块，在灰色区域中的所有像素点的光强和彩色以及饱和度都是相同的，因此该区域中的数据之间存在很大的冗余度。

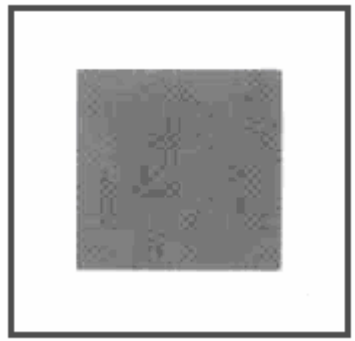


图7.2 空间冗余

空间冗余是图像数据中最基本的冗余。要去除这种冗余，人们通常将其视为一个整体，并用极少的数据量来表示，从而减少邻近像素之间的空间相关性，已达到数据压缩的目的 。

7.1.2、图像数据冗余 (Image data redundancy)

2. 时间冗余

由于活动图像序列中的任意两相邻的图像之间的时间间隔很短，因此两幅图像中存在大量的相关信息，如图7.3所示 。

时间冗余是活动图像和语音数据中经常存在的一种冗余。

图7.3 时间冗余

7.1.2、图像数据冗余 (Image data redundancy)

3. 信息熵冗余

信息熵冗余是针对数据的信息量而言的。设某种编码的平均码长为

$$L = \sum_{i=0}^{k-1} p(s_i) l(s_i)$$

式中，$l(s_i)$ 为分配给第符号 s_i 的比特数，$p(s_i)$ 为符号出现的概率。

这种压缩的目的就是要使L接近 $H(X)$

7.1.2、图像数据冗余 (Image data redundancy)

4. 结构冗余

图7.4表示了一种结构冗余。从图中可以看出。它存在着非常强的纹理结构，这使图像在结构上产生了冗余。

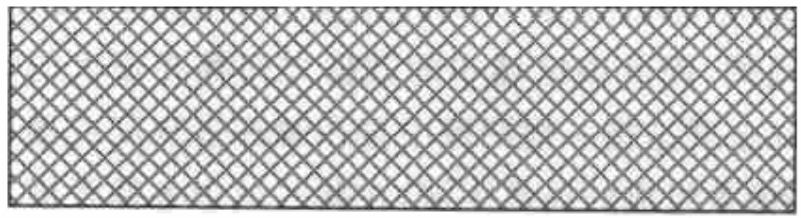


图7.4 结构冗余

7.1.2、图像数据冗余 (Image data redundancy)

5. 知识冗余

随着人们认识的深入，某些图像所具有的先验知识，如人脸图像的固有结构（包括眼、耳、鼻、口等）为人们所熟悉。这些由先验知识得到的规律结构就是知识冗余。

6. 视觉冗余

由于人眼的视觉特性所限，人眼不能完全感觉到图像画面的所有细小的变化。例如人眼的视觉对图像边缘的剧烈变化不敏感，而对图像的亮度信息非常敏感，因此经过图像压缩后，虽然丢了一些信息，但从人眼的视觉上并未感到其中的变化，而仍认为图像具有良好的质量。

7.1.3、图像压缩编码分类 (Coding methods of Image Compression

无失真图像压缩编码利用图像信源概率分布的不均匀性，通过变长编码来减少信源数据冗余，使编码后的图像数据接近其信息熵而不产生失真，因而也通常被称为熵编码。

有限失真编码则是根据人眼视觉特性，在允许图像产生一定失真的情况下（尽管这种失真常常不为人眼所觉察），利用图像信源在空间和时间上具有较大的相关性这一特点，通过某一种信号变换来消除信源的相关性、减少信号方差，达到压缩编码的目的。

7.1.4、压缩技术的性能指标(Evaluation Index of Image Compression approaches)

1. 压缩比

为了表明某种压缩编码的效率，通常引入压缩比这一参数，它的定义为：

$$c = \frac{b_1}{b_2}$$

其中 b_1 表示压缩前图像每像素的平均比特数，b_2 表示压缩后每像素所需的平均比特数，一般的情况下压缩比c总是大于等于1的，c愈大则压缩程度愈高。

7.1.4、压缩技术的性能指标(Evaluation Index of Image compressionapproaches)

2. 平均码字长度

平均码字长度: 设$l(c_k)$为数字图像第k个码字C_k 的长度(C_k编码成二进制码的位数)。其相应出现的概率为$p(c_k)$, 则该数字图像所赋予的平均码字长度为:

$$L = \sum_{k=1}^{m} p(c_k) l(c_k) \quad \text{单位为bit}$$

3. 编码效率

在一般情况下, 编码效率往往可用下列简单公式表示:

$$\eta = \frac{H}{L}$$

7.1.4、压缩技术的性能指标(Evaluation Index of Image compressionapproaches)

4. 冗余度

$$r = 1 - \eta$$

R越小，说明可压缩的余地越小。

7.2 无失真图像压缩编码（Lossless image compression）

无失真失真图像压缩编码就是指图像经过压缩、编码后恢复的图像与原图像完全一样，没有任何失真 .

常用的无失真图像压缩编码有许多种。如哈夫曼（*Huffman*）编码、游程编码和算术编码。

7.2.1、哈夫曼编码(Huffman coding)

哈夫曼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。

1. 可变长最佳编码定理

对于一个无记忆离散信源中每一个符号，若采用相同长度的不同码字代表相应符号，就叫做等长编码。若对信源中的不同符号用不同长度的码字表示就叫做不等长或变长编码 。

7.2.1、哈夫曼编码(Huffman coding)

2. 哈夫曼（*Huffman*）编码的编码思路

实现哈夫曼编码的基本步骤如下:

(1) 将信源符号出现的概率按由大到小地顺序排列。

(2) 将两处最小的概率进行组合相加,形成一个新概率。并按第(1)步方法重排,如此重复进行直到只有两个概率为止。

(3) 分配码字,码字分配从最后一步开始反向进行,对最后两个概率一个赋于“0”码字,一个赋于“1”码字。如此反向进行到开始的概率排列,在此过程中,若概率不变采用原码字。

7.2.1、哈夫曼编码(Huffman coding)

举例： 设输入图像的灰度级{*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，并计算编码效率、压缩比、冗余度 。

按照上述的编码过程和例题所给出的参数，其哈夫曼编码过程及其编码的结果如图7.6所示。图像信源熵为：

$$H=-\sum_{K=1}^{M}P_K\log_2 P_K=-(0.4\times\log_2 0.4+0.18\times\log_2 0.18+2\times0.1\log0.1+$$

$$0.07\times\log_2 0.07+0.06\times\log_2 0.06+0.05\times\log_2 0.05+0.04\times\log_2 0.04)=2.55$$

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：

$$L=\sum_{K=1}^{M}l_K P_K=0.40\times1+0.18\times3+0.10\times3+0.10\times4+0.07\times4+$$

$$0.06\times4+0.05\times5+0.04\times5=2.61$$

7.2.1、哈夫曼编码(Huffman coding)

编码效率:

$$\eta = H / L = 2.55 / 2.61 = 97.8\%$$

压缩比:

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:

$$\gamma = 1 - \eta = 2.2\%$$

7.2.1、哈夫曼编码(Huffman coding)

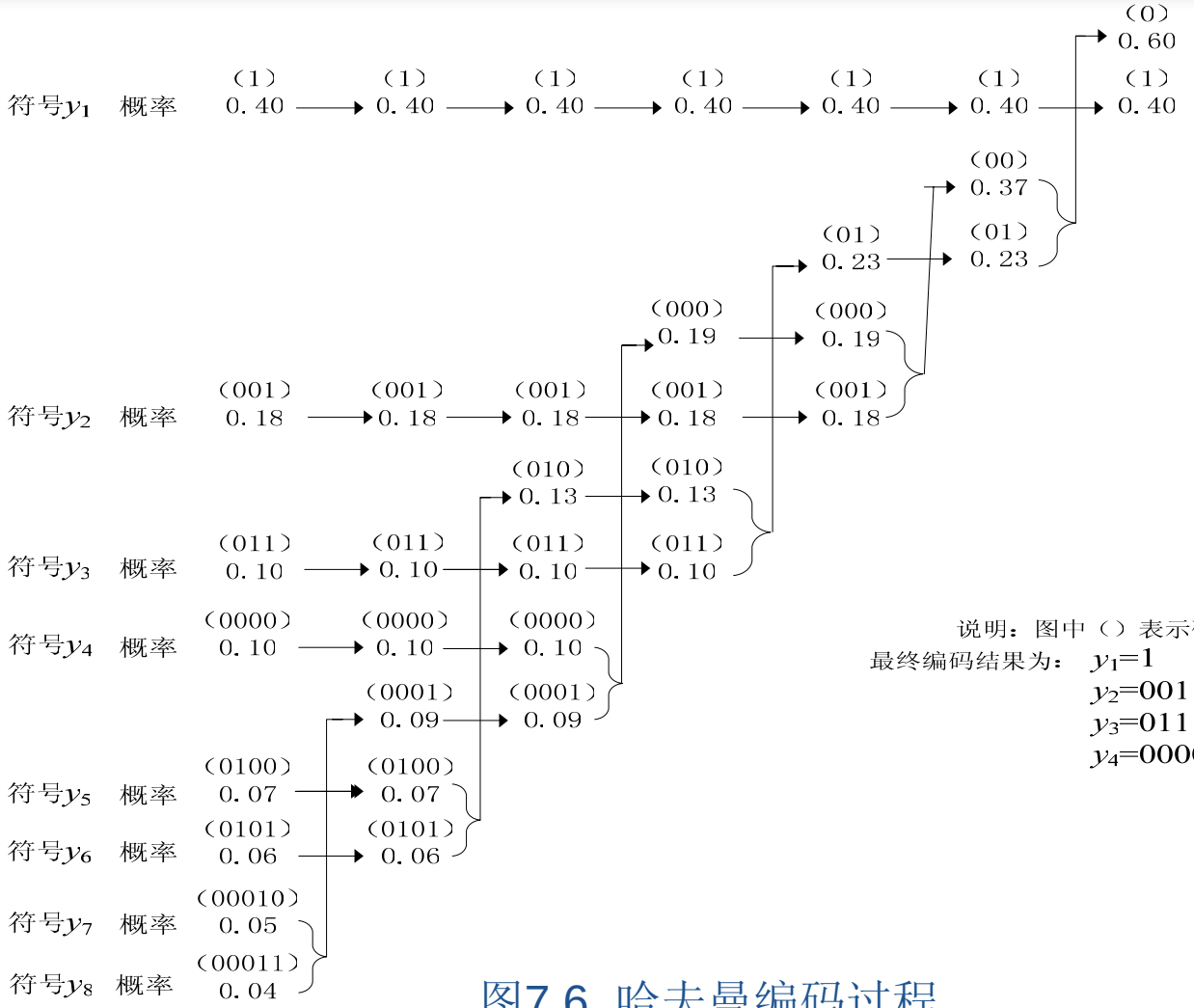


图7.6 哈夫曼编码过程

7.2.1、哈夫曼编码(Huffman coding)

3. 哈夫曼（*Huffman*）编码的特点

（1）*Huffman*编码所构造的码并不是唯一的，但其编码效率是唯一的。

（2）对不同信源，其编码效率是不同的。

（3）实现电路复杂，且存在误码传播问题。

（4）Huffman编码只能用近似的整数而不是理想的小数来表示单个符号，这也是Huffman编码无法达到最理想的压缩效果的原因.

7.2.2、游程编码(Run-length coding)

下面以一个具体的二值序列为例进行说明。已知一个二值序列00101110001001 ，根据游程编码的规则，可知其游程序列为21133121 。

可见图像中具有相同灰度（或颜色）的图像块越大、越多，压缩的效果就越好，反之当图像越复杂，即其中的颜色层次越多时，则其压缩效果越不好，因此对于复杂的图像，通常采用游程编码与Huffman编码的混合编码方式，

即首先进行二值序列的游程编码，然后根据“0”游程与“1”游程长度的分布概率，再进行*Huffman*编码。

7.2.3 算术编码(Arithmetic coding)

- 算术编码不是将单个信源符号映射成一个码字，而是把整个信源表示为实数线上的0到1之间的一个区间，其长度等于该序列的概率。再在该区间内选择一个代表性的小数，转化为二进制作为实际的编码输出。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。

7.2.3 算术编码(Arithmetic coding)

算术编码不是将单个信源符号映射成一个码字，而是把整个信源表示为实数线上的0到1之间的一个区间，其长度等于该序列的概率。再在该区间内选择一个代表性的小数，转化为二进制作为实际的编码输出。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。

7.2.3 算术编码(Arithmetic coding)

举例： 假设信源符号为X={00，01，10，11}，其中各符号的概率为P(X)={0.1，0.4，0.2，0.3}。对这个信源进行算法编码的具体步骤如下：

1）已知符号的概率后，就可以沿着“概率线”为每个符号设定一个范围：[0，0.1)，[0.1，0.5)，[0.5，0.7)，[0.7，1.0)。把以上信息综合到表7.1中。

表 7.1 信源符号、概率和初始区间

符号	00	01	10	11
概率	0.1	0.4	0.2	0.3
初始区间	[0，0.1)	[0.1，0.5)	[0.5，0.7)	[0.7，1.0)

7.2.3 算术编码(Arithmetic coding)

2） 假如输入的消息序列为：10、00、11、00、10、11、01，其算术编码过程为：

第一步：初始化时，范围*range*为1.0，低端值*low*为0。下一个范围的低、高端值分别由下式计算：

$$\begin{cases} low = low + range * range\_low \\ high = low + range * range\_high \end{cases}$$

其中等号右边的*range*和*low*为上一个被编码符号的范围和低端值；*range_low*和*range_high*分别为被编码符号已给定的出现概率范围的低端值和高端值。

7.2.3 算术编码(Arithmetic coding)

对第一个信源符号10编码:

$$\begin{cases} low = low + range * range\_low = 0 + 1 * 0.5 = 0.5 \\ high = low + range * range\_high = 0 + 1 * 0.7 = 0.7 \end{cases}$$

所以，信源符号10将区间 $[0,1) \Rightarrow [0.5,0.7)$。

下一个信源符号的范围为 $range = range\_high - range\_low = 0.2$

第二步：对第二个信源符号00编码:

$$\begin{cases} low = low + range * range\_low = 0.5 + 0.2 * 0 = 0.5 \\ high = low + range * range\_high = 0.5 + 0.2 * 0.1 = 0.52 \end{cases}$$

所以信源符号00将区间 $[0.5,0.7) \Rightarrow [0.5,0.52)$。

下一个信源符号的范围为

$$range = range\_high - range\_low = 0.02$$

7.2.3 算术编码(Arithmetic coding)

第三步：对第三个信源符号11编码:

$$\begin{cases} low = low + range * range\_low = 0.5 + 0.02 * 0.7 = 0.514 \\ high = low + range * range\_high = 0.5 + 0.02 * 1 = 0.52 \end{cases}$$

所以信源符号11将区间 $[0.5, 0.52) \Rightarrow [0.514, 0.52)$。

下一个信源符号的范围为 $range = range\_high - range\_low = 0.006$

第四步：对信源符号00编码:

$$\begin{cases} low = low + range * range\_low = 0.514 + 0.006 * 0 = 0.514 \\ high = low + range * range\_high = 0.514 + 0.006 * 0.1 = 0.5146 \end{cases}$$

下一个信源符号的范围为

$$range = range\_high - range\_low = 0.0006$$

7.2.3 算术编码(Arithmetic coding)

第五步：对第五个信源符号10编码：

$$\begin{cases} low = low + range * range\_low = 0.514 + 0.0006 * 0.5 = 0.5143 \\ high = low + range * range\_high = 0.514 + 0.0006 * 0.7 = 0.51442 \end{cases}$$

所以，信源符号10将区间 $[0.514, 0.5146) \Rightarrow [0.5143, 0.51442)$。

下一个信源符号的范围为

$$range = range\_high - range\_low = 0.00012$$

第六步：对第六个信源符号11编码：

$$\begin{cases} low = low + range * range\_low = 0.5143 + 0.00012 * 0.7 = 0.514384 \\ high = low + range * range\_high = 0.5143 + 0.00012 * 1 = 0.51442 \end{cases}$$

所以，信源符号11将区间 $[0.5143, 0.51442) \Rightarrow [0.514384, 0.51442)$。

7.2.3 算术编码(Arithmetic coding)

下一个信源符号的范围为

$$range = range\_high - range\_low = 0.000036。$$

第七步：对第七个信源符号01编码：

$$\begin{cases} low = low + range * range\_low = 0.514384 + 0.000036 * 0.1 = 0.5143876 \\ high = low + range * range\_high = 0.514384 + 0.000036 * 0.5 = 0.514402 \end{cases}$$

所以，信源符号01将区间

$$[0.51484, 0.51442) \Rightarrow [0.5143876, 0.514402)。$$

最后从[0.5143876，0.514402]中选择一个数作为编码输出，这里选择0.5143876。

7.2.3 算术编码(Arithmetic coding)

综上所述，算术编码是从全序列出发，采用递推形式的一种连续编码，使得每个序列对应该区间内一点，也就是一个浮点小数；这些点把[0，1）区间分成许多小段，每一段长度则等于某序列的概率。再在段内取一个浮点小数，其长度可与序列的概率匹配，从而达到高效的目的。

7.2.3 算术编码(Arithmetic coding)

解码是编码的逆过程，通过编码最后的下标界值0.5143876得到信源“10 00 11 00 10 11 01”是唯一的编码。解码操作过程综合如下：

$$\frac{0.5143876-0}{1}=0.5143876\Rightarrow 10$$

$$\frac{0.5143876-0.5}{0.2}=0.071938\Rightarrow 00$$

$$\frac{0.071938-0}{0.1}=0.71938\Rightarrow 11$$

$$\frac{0.71938-0.7}{0.3}=0.0646\Rightarrow 00$$

$$\frac{0.0646-0}{0.1}=0.646\Rightarrow 10$$

$$\frac{0.646-0.5}{0.2}=0.73\Rightarrow 11$$

$$\frac{0.73-0.7}{0.3}=0.1\Rightarrow 01$$

$$\frac{0.1-0.1}{0.4}=0\Rightarrow \text{结束}$$

7.2.3 算术编码(Arithmetic coding)

从以上算术编码算法可以看出，算术编码具有以下特点:

1、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。

2、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。

3、算术编码也是一种对错误很敏感的方法。

7.3 有限失真图像压缩编码（Lossy Image Compression）

从前面的分析可知，无失真图像压缩编码的平均码长存在一个下限，这就是信源熵。换句话说，如果无失真图像编码的压缩效率越高，那么编码的平均码长越接近信源的熵。因此，无失真编码的压缩比不可能很高，而在有限失真图像编码方法中，则允许有一定的失真存在，因而可以大大提高压缩比，压缩比越大，引入的失真也就越大，但同样提出了一个新的问题，这就是在失真不超过某种极限的情况下，所允许的编码比特率的下限是多少，率失真函数回答的便是这一问题。

7.3.1、率失真函数(Rate distortion function)

率失真函数是指在信源一定的情况下使信号的失真小于或等于某一值D所必需的最小的信道容量，常用$R(D)$表示，D代表所允许的失真，对连续信源的编码与传输，可以用失真度$d(x,y)$和失真函数$D(x,y)$表示，即

$$D(x,y)=\iint p(x,y)d(x,y)dxdy$$

通常采用以下几种失真度量:

(1) 均匀误差

(2) 绝对误差

(3) 频域加权误差

(4) 超视觉阈值均方误差

在图7.7中给出了率失真函数$R(D)$与失真D的关系曲线。可见对于离散信源，当$D=0$（即无失真情况下）时，所需的比特数为$R(0)$，并且小于收到信号的熵值$H(Y)$；当D逐渐增大时，所需的率失真函数则随之下降，因此我们可以总结出率失真函数$R(D)$的性质。

7.3.1、率失真函数(Rate distortion function)

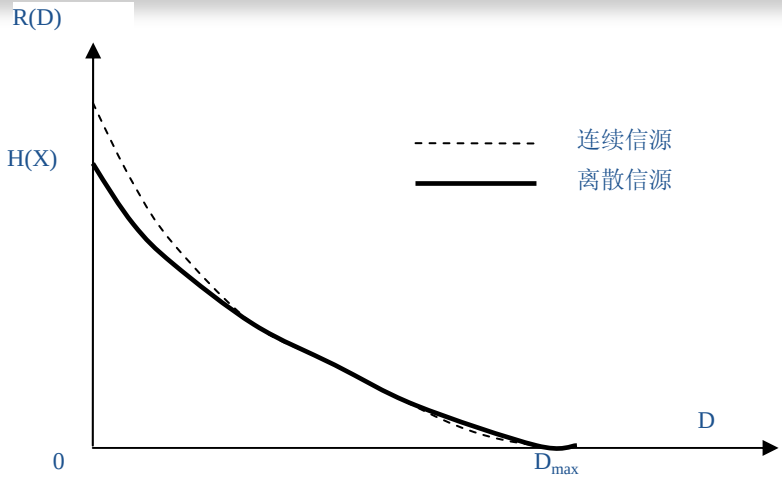


图7.7 *R*(*D*)的典型曲线

率失真函数具有以下性质

(1) 当D<0时，不存在R(D).

(2(当D≥Dmax（Dmax为正值，其数值上等于信号方差σ^2）时，R(D)=0,表示此时所传输的数据信息毫无意义。

(3) 当0<D<Dmax时，R(D)是一个下凸型连续函数。

7.3.1、率失真函数(Rate distortion function)

率失真函数具有以下性质

(1) 当D<0时，不存在R(D).

(2(当D≥Dmax（Dmax为正值，其数值上等于信号方差σ^2）时，R(D)=0,表示此时所传输的数据信息毫无意义。

(3) 当0<D<Dmax时，R(D)是一个下凸型连续函数。

7.3.2 线性预测编码和变换编码 (Linear Prediction and Transform Coding)

1. 线性预测编码

目前用得较多的线性预测方法,全称为差值脉冲编码调制(*DPCM:DifferentialPulseCodeModulation*),简称为*DPCM*。

图7.9是一个4阶预测器的示意图,图*(a)*表示预测器所用的输入像素和被预测像素之间的位置关系,图*(b)*表示预测器的结构。

7.3.2 线性预测编码和变换编码 (Linear Prediction and Transform Coding)

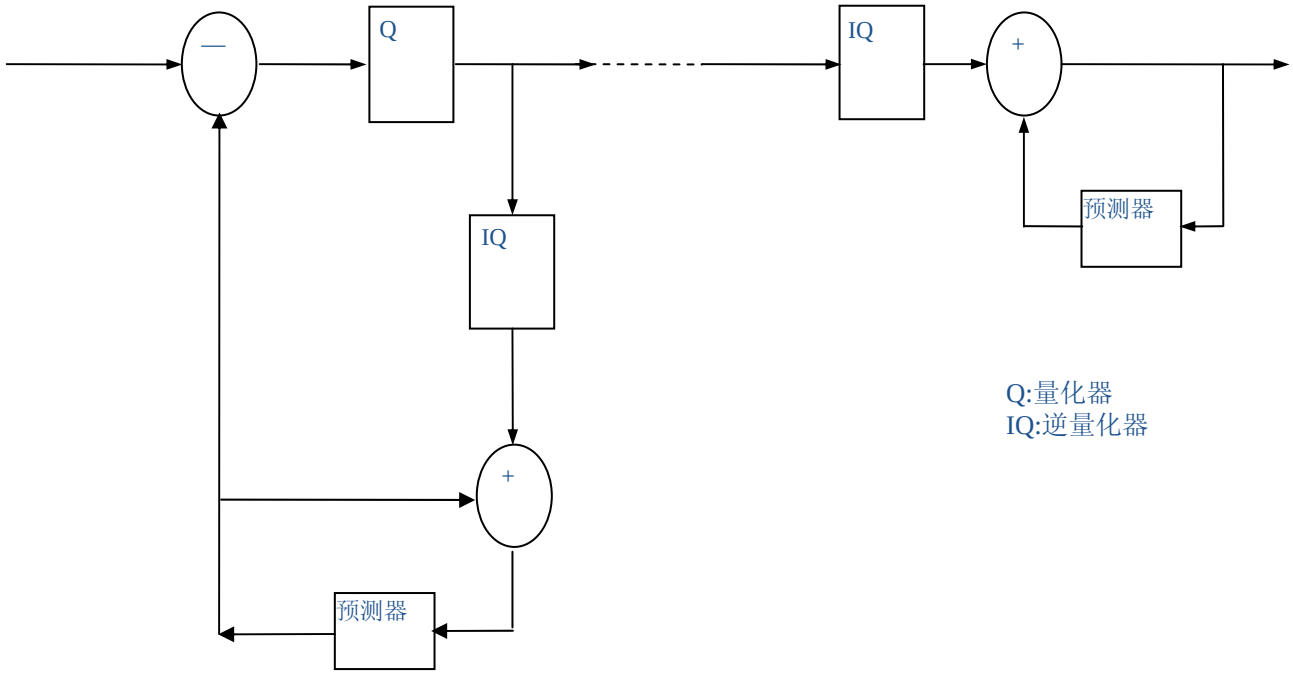


图7.8 DPCM系统框图

7.3.2 线性预测编码和变换编码 (Linear Prediction and Transform Coding)

图7.9是一个4阶预测器的示意图，图 *(a)*表示预测器所用的输入像素和被预测像素之间的位置关系，图 *(b)*表示预测器的结构.

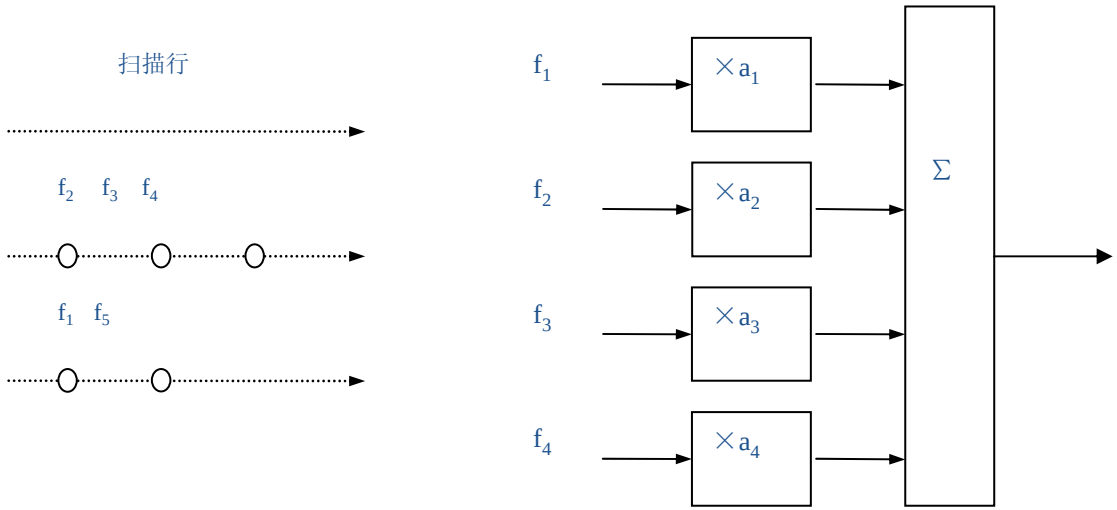


图7.9 预测像素和预测器

7.3.2 线性预测编码和变换编码 (Linear Prediction and Transform Coding)

（1）变换编码的工作原理

变换编码的工作原理如图所示

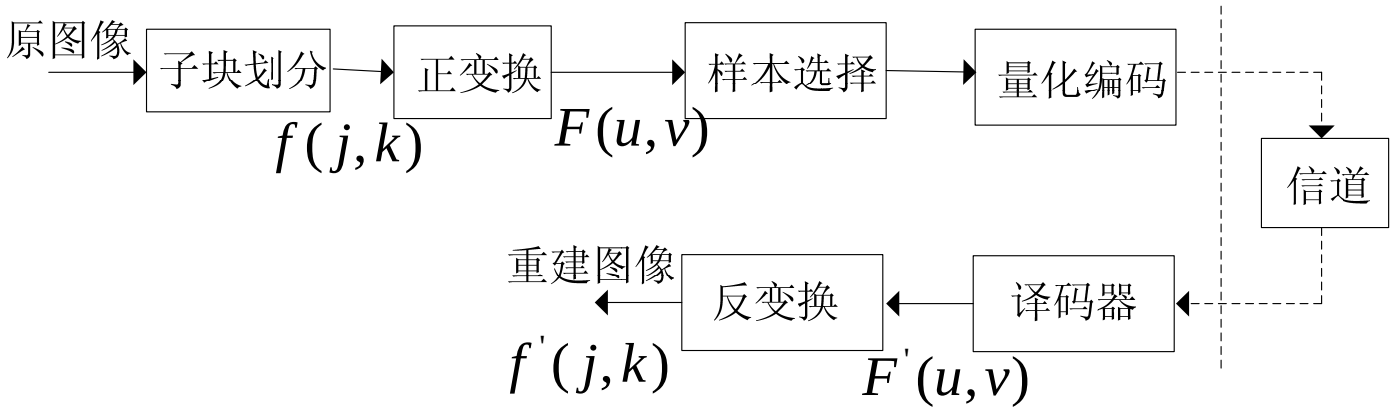


图7.11 变换编码系统原理

7.3.2 线性预测编码和变换编码 (Linear Prediction and Transform Coding)

变换编码首先将一幅N × N大小的图像分割成 $(N/n)^2$ 个子图像。然后对子图像进行变换操作，解除子图像像素间的相关性，达到用少量的变换系数包含尽可能多的图像信息的目的。接下来的量化步骤是有选择地消除或粗量化带有很少信息的变换系数，因为它们对重建图像的质量影响很小。

7.3.2 线性预测编码和变换编码 (Linear Prediction and Transform Coding)

（2）基于DCT的图像压缩编码

图像的DCT(离散余弦变换)已在4.6节中阐明. DCT具有把高度相关数据能量集中的能力，这一点和傅里叶变换相似，且DCT得到的变换系数是实数，因此广泛用于图像压缩。下面是用二维离散余弦变换进行图像压缩说明。

例如一个图像信号，$f(j,k)_{8\times8}$其灰度值为

$$f(j,k)_{8\times8}=\begin{bmatrix}79&75&79&82&82&86&94&94\\76&78&76&82&83&86&85&94\\72&75&67&78&80&78&74&82\\74&76&75&75&86&80&81&79\\73&70&75&67&78&78&79&85\\69&63&68&69&75&78&82&80\\76&76&71&71&67&79&80&83\\72&77&78&69&75&75&78&78\end{bmatrix}$$

得到*DCT*系数矩阵为

$$F(\mu,v)_{8\times8}=\begin{bmatrix}619 & -29 & 8 & 2 & 1 & -3 & 0 & 1\\ 22 & -6 & -4 & 0 & 7 & 0 & -2 & -3\\ 11 & 0 & 5 & -4 & -3 & 4 & 0 & -3\\ 2 & -10 & 5 & 0 & 0 & 7 & 3 & 2\\ 6 & 2 & -1 & -1 & -3 & 0 & 0 & 8\\ 1 & 2 & 1 & 2 & 0 & 2 & -2 & -2\\ -8 & -2 & -4 & 1 & 5 & 1 & -1 & 1\\ -3 & 1 & 5 & -2 & 1 & -1 & 1 & -3\end{bmatrix} \quad (7.29)$$

由（7.29）式可以看出*DCT*系数集中在低频区域、越是高频区域系数值越小，根据人眼的视觉特性，通过设置不同的视觉阈值或量化电平，将许多能量较小的高频分量量化为0，可以增加变换系数中“0”的个数，同时保留能量较大的系数分量，从而获得进一步的压缩。在*JPEG*的基本系统中，就是采用二维*DCT*的算法作为压缩的基本方法。

（a）原始图像　　（b）压缩后的图像

图7.12 图像压缩前后的比较

7.3.3 矢量量化编码（Vector Quantifiation Coding）

矢量量化(Vector Quantifiation ，VQ)技术是一种有损压缩技术，它根据一定的失真测度在码书中搜索出与输入矢量失真最小的码字的索引，传输时仅传输这些码字的索引，接收方根据码字索引在码书中查找对应码字，再现输入矢量。

1. 矢量量化原理

矢量量化的过程如图7-13所示，可以分为量化和反量化两部分。在矢量量化中，根据某种失真最小原则，来分别决定如何对n维矢量空间进行划分，以得到合适的C个分块，以及如何从每个分块选出它们各自合适的代表Xi′ 。

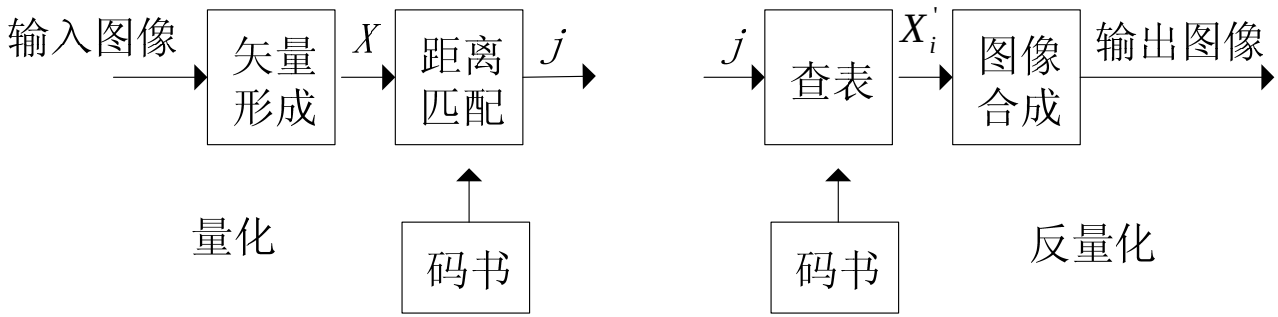


图7.13 矢量量化编译码

2. 码书的设计

矢量量化中的一个关键问题就是码书的设计。码书的设计越适合待编码的图像的类型，矢量量化器的性能就越好，因为实际中不可能为每一幅待编码的图像单独设计一个码书，所以通常是以一些代表性的图像构成的训练集为基础，为一类图像设计出一个码书 。

7.4 图像编码新技术（New Image Compression Technology）

7.4.1 子带编码（Subband coding）

7.4.2 模型基编码（Model-based coding）

7.4.3 分形编码（Fractal coding）

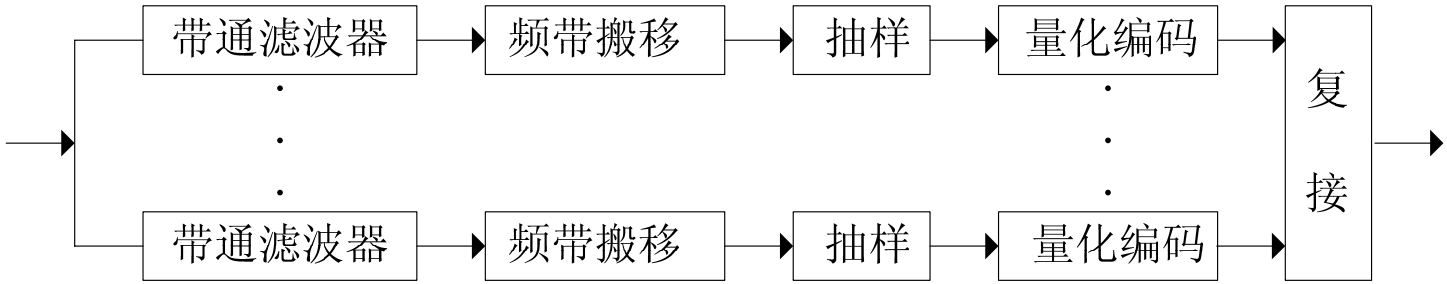
7.4.1 子带编码（Subband coding）

1. 子带编码

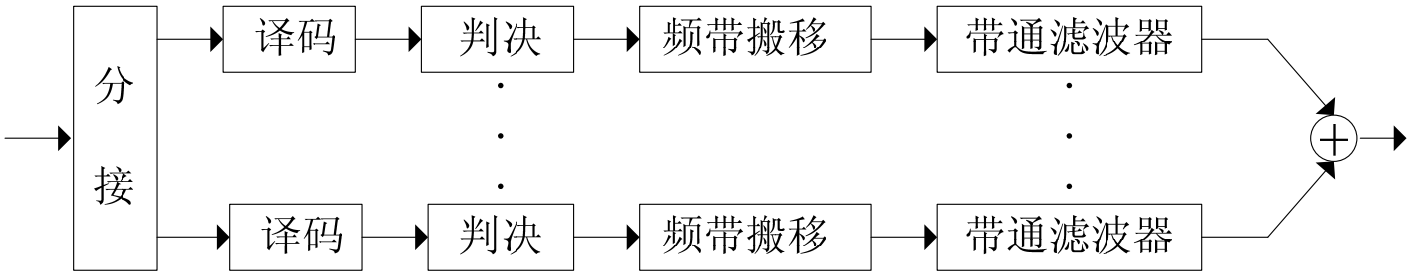
子带编码是一种在频率域中进行数据压缩的算法。其指导思想是首先在发送端将图像信号在频率域分成若干子带，然后分别对这些子带信号进行频带搬移，将其转换成基带信号，再根据奈奎斯特定理对各基带信号进行取样、量化和编码，最后合并成为一个数据流进行传送。

7.4.1 子带编码（Subband coding）

子带编码的工作原理



(a) 编码器



(b)解码器

图7.15 子带编码器工作原理

7.4.2 模型基编码（Model-based codin

模型基编码主要是一种参数编码方法，因此它与基于保持信号原始波形的所谓波形编码相比有着本质区别。相对于对像素进行编码而言，对参数的编码所需的比特数要少得多，因此可以节省大量的编码数据。

7.4.2 模型基编码（Model-based coding）

编码原理

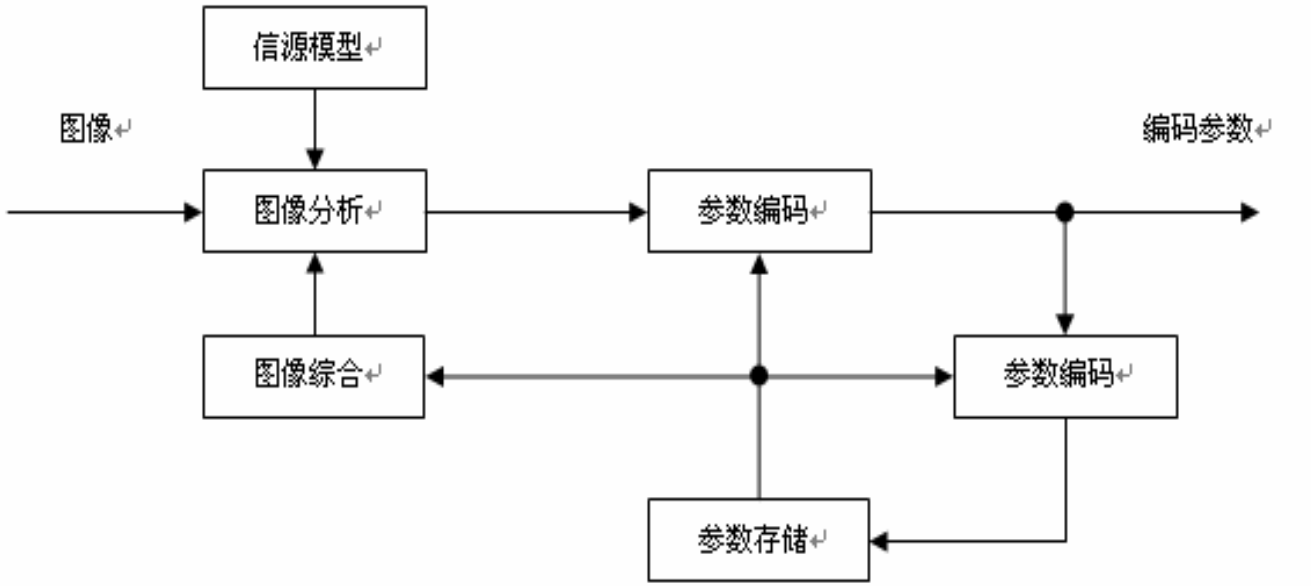


图7.16分析综合编码器原理图

7.4.3 分形编码（Fractal coding）

在分形编码中，使用了迭代函数（*IFS*）理论、仿射理论和拼贴定理，具体应用过程如下：首先采用如颜色分割、边缘检测、频谱分析等，将原始图像分割成一系列子图像，如一棵树、一片树叶然后在分形集合中查找这些子图像，但分形集所存储的并不是具体的子图像，而是迭代函数，因此分形集中包含许多迭代函数。由于迭代只需要几个参数来表示，因此能够得到高压缩比。由此可见，分形编码中存在两大难点，这就是如何进行图像分割和构造迭代 。

7.5 图像压缩技术标准（Image Compression Standards）

7.5.1 JPEG压缩（JPEG Compression）

7.5.2 JPEG 2000

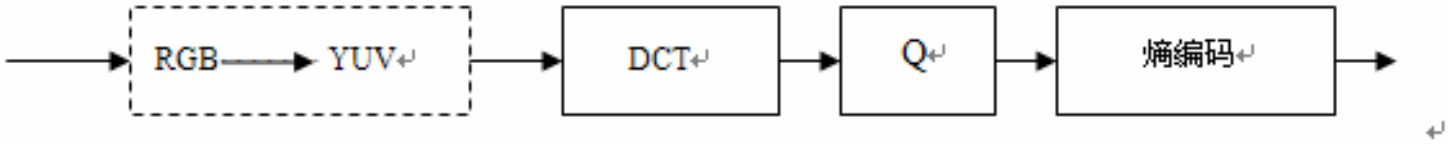
7.5.3 H.26X标准（*H.26X standards*）

7.5.4 MPEG标准（MPEG standards）

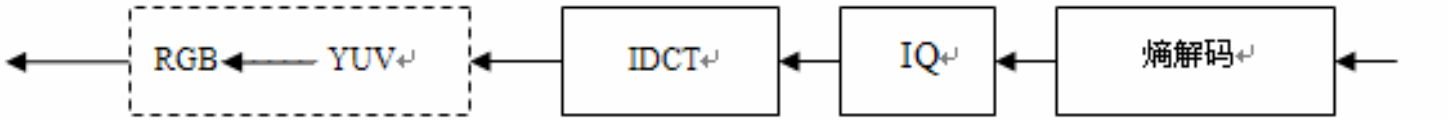
7.5.1、 JPEG压缩（JPEG Compression）

- *JPEG*是用于彩色和灰度静止图像的一种完善的有损/无损压缩方法，对相邻像素颜色相近的连续色调图像的效果很好，而用于处理二值图像效果较差。*JPEG*是一种图像压缩方法，它对一些图像特征如像素宽高比、彩色空间或位图行的交织方式等并未作严格的限制。
- 在*JPEG*基准编码系统中，输入和输出图像都限制为8比特图像，而量化的*DCT*系数值限制在11比特。图7.15给出了*JPEG*编码/解码方框图。

7.5.1、 JPEG压缩（JPEG Compression）



（a）



（b）

图 7.17 JPEG 基本系统的编码器的结构框图

7.5.2 JPEG 2000

*JPEG*2000主要由6个部分组成,其中,第一部分为编码的核心部分,具有最小的复杂性,可以满足80%的应用需要,其地位相当于JPEG标准的基本系统,是公开并可免费使用的。第二至第六部分则定义了压缩技术和文件格式的扩展部分,包括编码扩展(第二部分),*MotionJPEG2000*(*MJP2*,第三部分),一致性测试(第四部分),参考软件(第五部分),混合图像文件格式(第六部分)。

图7.18是*JPEG2000*的基本模块组成,其中包括预处理、*DWT*、量化、自适应算术编码以及码流组织等5个模块 。

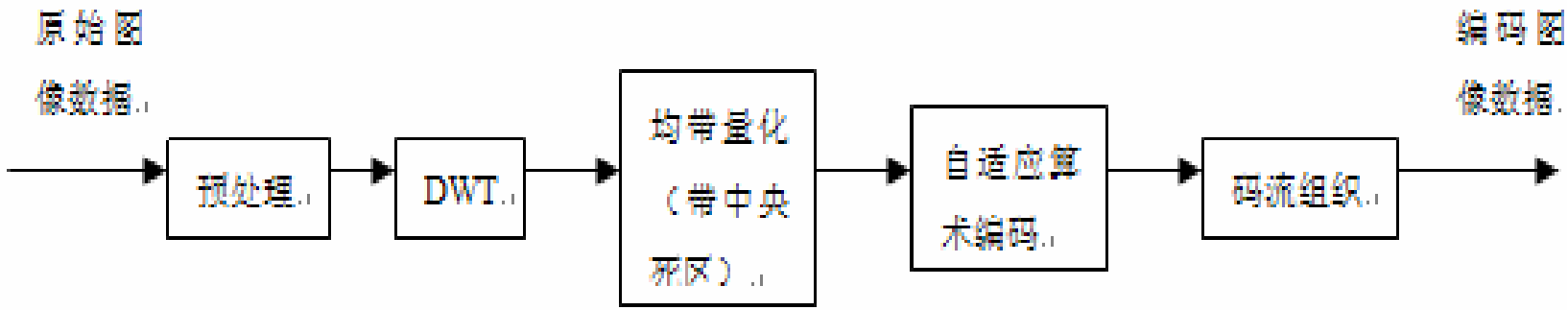


图 7.18 JPEG2000 基本编码模块组成

7.5.3 H.26X标准（*H.26X standards*）

1. H.261

TU-T(原CCITT)于1990年7月通过H.261建议——“$p\times 64kbit/s$视听业务的视频编解码器”,其中p的范围是1～30,覆盖了整个窄带*ISDN*基群信道速率。该标准的应用目标是会议电视和可视电话,通常p=1,2时适用于可视电话,p在6以上时可以适用于会议电视业务。

- **2. H.263**

H.263建议中仍采用*H*.261建议的混合编码器 ,但去掉了信道编码部分。在信源编码器中,*DCT*、量化器的种类,以及对*DCT*的量化系数的

*Z*字形扫描和二维*VLC*等处理与*H*.261建议是一致的, *H*.263的基本编码方法与*H*.261是相同的,均为混合编码方法。但H.263在编码的各个环节上考虑得更加细致,以便节省码字。

7.5.4 MPEG标准（MPEG standards）

MPEG是活动图像专家组（Moving Picture Expert Group）的缩写。它建立于1988年，属于ISO/IEC信息技术联合委员会第29研究组的第11工作组(JJCI／SC29，WG11)。MPEG专家组开始时完成三个工作项目，即压缩码率达1．5Mb／s的MPEG—1编码标难；压缩码率达10 Mb／s的MPEG—2编码标准及压缩码率达40 Mb／s的MPEG—3编码标准。但后来MPEG—2的工作内容扩大并包含了MPEG—3的内容，1 992年7月撤消了MPEG—3项目组。由于甚低码率(very lowbit-rate)的音视频编解码的需要，1993年7月成立了MPEG—4

第七章完


Thank You !


Thank You !