第3章 多媒体数据压缩技术

3.1 数据压缩工作原理

数字化音频和视频信号的数据量正常情况下都是在每秒兆位级以上，在目前的技术下，对如此巨量的数据存储和实时传输困难重重。但是，在多媒体系统中，为了取得满意的视听效果，对多媒体数据进行实时存取和传输又是必要的。因此，对多媒体数据必须进行压缩处理。音频、图像和视频等多媒体数据又确实有很大的压缩潜力。以图像为例，按照统计学分析结果，一帧图像，其画面灰度的分布具有块状结构，图像内容在整体上具有结构性，像素与像素之间在行列上有很大的相关性，因此一帧图像在整体上包含大量的冗余数据，还有一些视频数据具有时间和空间交叉的冗余度，具有分形性质，声音等媒体也有着类似的情形。因此，多媒体数据压缩技术的发展潜力十分巨大，具有广阔的应用前景。

本章主要介绍多媒体数据压缩技术的基本原理和方法，并介绍目前得到广泛应用和影响巨大的相关图像、视频压缩编码国际标准及其新技术。

3.1.1 数据压缩概述

数据压缩是指在不丢失有用信息的前提下，缩减数据量以减少存储空间，提高其传输、存储和处理效率，或按照一定的算法对数据进行重新组织，减少数据的冗余和存储空间的一种技术方法。

数据压缩分为有损压缩和无损压缩。无损压缩是指使用压缩后的数据进行重构（或者叫作还原、解压缩），重构后的数据与原来的数据完全相同；无损压缩用于要求重构的信号与原始信号完全一致的场合。无损压缩算法一般可以把普通文件的数据压缩到原来的1/2～1/4，常用的无损压缩算法有哈夫曼（Huffman）算法和LZW（Lenpel-Ziv & Welch）压缩算法。

有损压缩是指使用压缩后的数据进行重构，重构后的数据与原来的数据有所不同，但不影响人对原始资料表达的信息造成误解。有损压缩适用于重构信号不一定非要和原始信号完全相同的场合。例如，图像和声音的压缩就可以采用有损压缩，因为其中包含的数据往往多于人们的视觉系统和听觉系统所能接收的信息，丢掉一些数据而不至于对声音或者图像所表达的意思产生误解，但可大大提高压缩比。

在多媒体计算系统中，信息从单一媒体转到多种媒体；若要表示传输和处理大量数字化了的声音、图片、影像视频信息等，数据量是非常大的。例如，一幅具有中等分辨率(640×480 像素)的真彩色图像(24 位/像素)，它的数据量约为每帧 7.37Mb。若要达到每秒 25 帧的全动态显示要求，每秒所需的数据量为 184Mb，而且要求系统的数据传输速率必须达到 184Mb/s，这在目前是无法达到的。对于声音也是如此。若用 16 位/样值的 PCM 编码，采样速率选为 44.1kHz，则双声道立体声声音每秒将有 176kb 的数据量。由此可见，音频、视频的数据量巨大，如果不进行处理，计算机系统几乎无法对它进行存取和交换。因此，在多媒体计算机系统中，为了达到令人满意的图像、视频画面质量和听觉效果，必须解决视频、图像、音频信号数据的大容量存储和实时传输问题。解决的方法，除了提高计算机本身的性能及通信信道的带宽外，更重要的是对多媒体进行有效的压缩。

3.1.2 数据压缩的基本原理

1. 数据压缩原理概述

1948 年，Shannon 的经典论文——《通信的数学原理》首次用数学语言阐明了概率与信息冗余度的关系，提出并建立了信息率失真函数概念。Shannon 借鉴热力学的概念，把信息中排除了冗余后的平均信息量称为“信息熵”，并给出了计算信息熵的数学表达式。1955 年，Shannon 进一步确立了码率失真理论，以上工作奠定了信息编码的理论基础。Shannon 创立的信息论正是经典数据压缩技术的理论基础，既给出了数据压缩的理论极限，同时又指出了数据压缩技术的实现途径。从本质上讲，数据压缩的目的就是要消除信息中的冗余，而信息熵及相关的定理恰好用数学手段精确地描述了信息冗余的程度。利用信息熵公式，人们可以计算出信息编码的极限，即在一定的概率模型下，无损压缩的编码长度不可能小于信息熵公式给出的结果。本节对信息论中与数据压缩原理相关的一些相关概念简要介绍如下。

1) 信息量

只考虑连续型随机变量的情况。设 p 为随机变量 X 的概率分布，即 $p(x)$为随机变量 X 在 $X=x$ 处的概率密度函数值，随机变量 X 在 x 处的 Shannon 信息量定义为：

$$I=\log_2(1/p(x))=-\log_2 p(x) \tag{3.1}$$

单位为 b。由于在计算机上是二进制，一般都采用 b，也就是编码 x 所需要的位数。Shannon 信息量用于刻画消除随机变量 X 在 x 处的不确定性所需的信息量的大小。如随机事件“中国足球进不了世界杯”不需要多少信息量(例如，多观察几场球赛的表现)就可以消除不确定性，因此该随机事件的 Shannon 信息量就少。再例如“抛一个硬币出现正面”，要消除它的不确定性，通过简单计算，需要 1b 的信息量，这意味着随机实验完成后才能消除不确定性。可以近似地将不确定性视为信息量，一个消息带来的不确定性大，就是带来的信息量大。例如，带来一个信息 $x=$sun raise in east，其概率 $p(x)=1$，信息量视为 0。带来另一个信息 $y=$明天有一个老师要抽查作业，有很多不确定性——8 个老师，其中 1 个要抽查，另外 7 个不抽查，那么就得去思索判断推理这其中的信息了，这就是高不确定性，高信息量。

2）信息熵

对于相互独立、没有相关性的离散无记忆信源 X，其符号的出现概率不受它前面符号是否出现的影响，它所携带的平均信息量 $H(X)$ 定义为：

$$H(X)=-\sum_i p(x_i)\log_2[p(x_i)] \tag{3.2}$$

其中，$p(x_i)$ 是符号 x_i 在信源 X 中出现的概率。

式 3.2 的平均信息量也称为信息熵（简称熵），信息量度量的是一个具体事件发生了所带来的信息，而熵则是在结果出来之前对可能产生的信息量的期望——考虑该随机变量的所有可能取值，即所有可能发生事件所带来的信息量的期望。信息熵还可以作为一个系统复杂程度的度量，系统越复杂，出现不同情况的种类越多，那么它的信息熵是比较大的。一个系统越简单，出现情况种类很少（极端情况为 1 种情况，那么对应概率为 1，对应的信息熵为 0），此时的信息熵较小。

对于离散无记忆信源而言，只要其源符号的熵值不等于等概率分布的熵值，就一定存在着信息冗余，也就存在着数据压缩的可能性，这就是熵编码的基本依据。

3）率失真函数

率失真函数是信息理论的基本概念之一。在信源给定时，总希望在满足一定失真 D 的情况下信息传输率 R 尽可能小。当信源符号与码字之间存在映射关系时，实际信息传输速率 R 的最小值为 $R(D)$，即信息传输速率 R 满足关系 $R\geqslant R(D)$，此时，函数 $R(D)$ 称为率失真函数，单位为 b/符号。

2. 数据压缩处理的过程

数据压缩处理一般由两个过程组成：一是编码过程，即对原始数据进行编码压缩，以便存储和传输；二是解码过程，即对压缩的数据进行解压，恢复成可用的数据。压缩处理系统的流程框图如图 3.1 所示。编码子系统的核心是源编码器，经过源编码器的压缩编码，输入的信源数据被减少到存储设备和传输介质所能支持的水平。实用系统中，在使用了源编码器之后还要进行下一层次的编码，图 3.1 中使用了通道编码器，其作用是把压缩位流翻译成一种既适合于存储又适合于传输的信号。源编码和通道编码是两种不同的编码过程，已经开发出能综合地进行源编码和通道编码的方法。由通道编码器和源解码器构成的解码子系统执行通道编码和源编码的逆过程，以便重构多媒体信号。所谓对信源数据进行编码，就是依某种数据压缩算法对原始数据进行压缩处理，形成压缩编码数据，随后对压缩编码数据进行存储与传输；解码则是对进行了压缩处理的数据进行解压缩，还原成可以使用的数据。

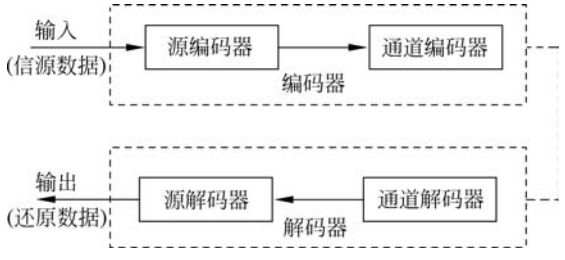


图 3.1　数据压缩处理流程图

因此，解码是编码的逆过程。

3. 有损压缩和无损压缩

目前常用的压缩编码可以分为有损压缩和无损压缩两大类。

1）无损压缩

无损压缩是能无失真地重建信源信号的一种压缩方法，无损压缩利用数据的统计冗余进行压缩，可完全恢复原始数据而不引起任何失真，但压缩率受到数据统计冗余度理论限制，一般为 2∶1～5∶1。这类方法广泛应用于文本数据、程序和特殊场合的图像数据压缩。由于压缩比的限制，仅使用无损压缩方法不可能解决多媒体数据的存储和传输问题。

2）有损压缩

有损压缩方法利用了人类视觉或者听觉的掩盖效应，即视觉对于图像边缘部分的急剧变化不敏感和对亮度信息敏感、但对色度分辨力弱等特点，以及听觉的生理特性，在压缩一部分信息后，由压缩后端数据恢复的原数据虽然信息量减少，但仍然有满意的主观效果。这种压缩的基本特点就是压缩后减少了信息量，即减少了熵，这是一种有失真的压缩，故称为有损压缩或者熵压缩编码。它的优点就是在保留满意的主观效果的基础上，能够获得比较高的压缩比，较显著地减少了多媒体数据的存储量，提高了实时传输速度。有损压缩是不可逆的编码方法。

衡量一种压缩算法的主要技术指标有以下几个。

（1）高压缩比，即压缩前后数据量的比值要大。

（2）算法简单，压缩、解压速度要快，尤其是解压缩速度要快。

（3）恢复效果好，即还原数据的失真小。

从系统设计角度看，可以把压缩处理归结为求位速率的问题，然而，这通常受到所要求的信号质量水平、算法实现复杂度以及端到端的通道时延等多种因素制约。

在多媒体应用中常用的数据压缩算法有：PCM（脉冲编码调制）、预测编码、变换编码（主成分变换或 K-L 变换、离散余弦变换等）、插值和外推法（空域亚采样、时域亚采样、自适应）、统计编码（哈夫曼编码、算术编码、Shannon-Fano 编码、行程编码）、矢量量化和子带编码等。混合编码是近年来广泛采用的方法，新一代的数据压缩方法，如基于模型的压缩方法、分形压缩和小波变换方法等也已经接近实用化水平。

接下来，本章将分别讲解数据压缩中涉及的 4 大编码技术，即统计编码、量化编码、变换编码和预测编码。

3.2 统计编码

3.2.1 统计编码的基础

统计编码是根据消息出现概率的分布特性而进行的压缩编码，它有别于预测编码和变换编码，这种编码的宗旨在于，在消息和码字之间找到明确的一一对应关系。统计编码又称为匹配编码，因为编码过程需要匹配信源的统计特性，对于无记忆信源，其剩余度主要体现在各个信源符号概率分布的不均匀性上。统计编码能实现压缩的不均性在编码后得以消

除，对于统计特性已知的有记忆离散信源可以采用算术码，马尔可夫信源可以按照状态采用哈夫曼编码，而对于统计特性未知的离散信源可以采用通用编码。

无论是进行等长编码还是变长编码，无失真信源压缩的理论极限值就是信源熵。对于这个极限值，就无法做到无失真编码。所以，香农第一定理对于无失真信源编码具有很重要的理论指导意义。如前所述，变长码往往在信源符号序列长度 N 不大时能编出效率很高而且无失真的信源编码，因此，统计编码通常采用变长码。

3.2.2　游程编码

游程编码又称“运行长度编码”或“行程编码”，是一种统计编码和与资料性质无关的无损数据压缩技术。该编码属于无损压缩编码，是栅格数据压缩的重要编码方法。变动长度编码法为一种“使用固定长度的码来取代连续重复出现的原始资料”的压缩技术。

1. 游程和游程序列

当信源是二元相关信源时，往往输出的信源符号序列中会连续出现多个“0”或“1”符号，为了提高编码的效率，科学家努力地寻找一种更为有效的编码方法。游程编码就是一种针对相关信源的有效编码方法，尤其适用于二元相关信源。游程编码在图文传真、图像传输等实际通信工程技术中得到应用，有时实际工程技术常常将游程编码和其他一些编码方法混合使用，能获得更好的压缩效果。

游程是指字符序列中各个字符连续重复出现而形成的字符串的长度，又称游程长度或游长。游程编码就是将这些字符序列映射成字符串的长度和串的位置的标识序列。知道了字符串的长度和串的位置的标志序列，就可以完全恢复出原来的字符序列。所以，游程编码不但适用于一维字符序列，也适用于二维字符序列。

对于二元相关信源，其输出只有两个符号，即“0”或“1”。在信源输出的一维二元序列中，连续出现“0”的这一段称为“0”游程，连续出现“1”的这一段称为“1”游程。对应段中的符号个数就是“0”游程长度和“1”游程长度。因为信源输出是随机的，所以游程长度是随机变量，其取值可为 1,2,3,…，直至无穷值。在输出的二元序列中，“0”游程和“1”游程总是交替出现的，若规定二元序列总是从“0”游程开始，那么第一个为“0”游程，接着第二个必定是“1”游程，以此类推，游程交替出现。这样，只需对串的长度(即游程长度)进行标记，然后可以将信源输出的任意二元序列一一对应地映射成交替出现的游程长度的标志序列。当然，一般游程长度都用自然数标记，所以就映射成交替出现的游程长度序列，简称游程序列两者中映射是可逆的，是无失真的。例如，某二元序列为

000 011 111 001 111 110 000 000 111 111…

对应的游程长度序列为 452676…

如果规定二元序列从“0”游程开始，那么给定上面的游程序列就很容易恢复出原来的二元序列。游程长度序列是多元序列，如果计算出各个游程长度的概率，就可以对其采用其他编码方法进行处理，从而进一步提高压缩效率。多元信源序列也存在相应的游程长度序列。R 元序列有 R 种游程，并且某游程的前后符号对应的游程是无法确定的，因此这种变换必须再加一些符号，才能使 R 元序列和其对应的游程长度序列是可逆的。

2. 游程编码

二元序列中，不同的“0”游程长度对应不同的概率，不同的“1”游程长度也对应不同的概率，这些概率叫作游程长度概率。游程编码的基本思想就是对不同的游程长度，按其不同的发生概率，分配不同的码字。游程编码可以将两种符号游程分别按其概率进行编码，也可以将两种游程长度混合起来一起编码。下面讨论游程长度编码后的平均码长的极限值。

考虑两种游程分开编码的情况。为了讨论方便，规定两种游程分别用白、黑表示。

白游程熵：

$$H_W = -\sum_{l_W=1}^{L} p(l_W)\lg p(l_W) \tag{3.3}$$

式中，l_W 代表白游程长度；$p(l_W)$表示白游程长度概率；L 表示白游程最大长度。

根据信源编码定理可知，白游程平均码长 $\bar{L}_W$ 应该满足式 3.4：

$$H_W \leqslant \bar{L}_W < H_W + 1 \tag{3.4}$$

若 $\bar{l}_W$ 为白游程长度的平均像素值，那么：

$$\bar{l}_W = -\sum_{l_W}^{L} l_W p(l_W) \tag{3.5}$$

由式 3.4 和式 3.5 得：

$$\frac{H_W}{\bar{l}_W} \leqslant \frac{\bar{L}_W}{\bar{l}_W} < \frac{H_W}{\bar{l}_W} + \frac{1}{\bar{l}_W} \tag{3.6}$$

令每个白像素的熵值为 h_W，则有 $h_W = \dfrac{H_W}{\bar{l}_W}$，每个白像素的平均码长为 $\bar{k}_W = \dfrac{\bar{L}_W}{\bar{l}_W}$，代入式 3.6：

$$h_W \leqslant \bar{k}_W < h_W + \frac{1}{\bar{l}_W} \tag{3.7}$$

同理，对黑游程可求得：

$$H_B = -\sum_{l_B=1}^{L} p(l_B)\lg p(l_B) \tag{3.8}$$

$$h_B \leqslant \bar{k}_B < h_B + \frac{1}{\bar{l}_B} \tag{3.9}$$

式中，H_B 为黑游程熵；l_B 代表黑游程长度；h_B 代表每个黑像素的熵值；$\bar{k}_B$ 代表每个黑像素的平均码长；$\bar{l}_B$ 为黑游程长度的平均像素值。

经过黑白平均可得到每个像素的熵值为：

$$h_{WB} = P_W h_W + P_B h_B \tag{3.10}$$

式中，P_W 为白像素的出现概率；P_B 为黑像素的出现概率。每个像素的平均码长为：

$$\bar{k}_{WB} = P_W \bar{k}_W + P_B \bar{k}_B \tag{3.11}$$

将式 3.7 乘以 P_W，则有：

$$P_W h_W \leqslant P_W \bar{k}_W < P_W h_W + \frac{P_W}{\bar{l}_W} \tag{3.12}$$

将式 3.9 乘以 P_B，则有：

$$P_B h_B \leqslant P_B \bar{k}_B < P_B h_B + \frac{P_B}{\bar{l}_B} \tag{3.13}$$

将式 3.12 和式 3.13 相加，则有：

$$h_{WB} \leqslant \bar{k}_{WB} < h_{WB} + \frac{P_W}{\bar{l}_W} + \frac{P_B}{\bar{l}_B} \tag{3.14}$$

所以，黑白游程分别是最佳编码后的平均码长仍以信源熵为极限，此时的熵 h_{WB} 已经是考虑了信源序列中符号之间的依赖关系了，这个熵 h_{WB} 小于信源序列之间无依赖情况下的熵值，所以游程编码压缩效率较高。

3.2.3　算术编码

由信源编码定理可知，仅对信源输出的单个符号进行编码其效率比较低，对信源输出的符号序列进行编码，并且当序列长度充分长时编码效率才达到香农定理的极限。假设对于只有两个出现概率很悬殊的消息符号$\{a_1, a_2\}$所组成的无记忆信源，若对单个符号进行编码，其效率很低，应该把它组成的长的消息队列来编码，如对输入长度为 n 的信源字符序列 x^n 进行不等长编码。但对长序列进行编码时必须考虑到编译码的可实现性，即编译码的复杂性、实时性和灵活性。

考虑一个离散平稳无记忆二元随机变量 X，信源字符表为$\{a_1, a_2\}$，概率分布为$\{p(X=a_1)=2/3, p(X=a_2)=1/3\}$，对输出序列 $x^n = x_1 x_2 \cdots x_n$ 进行编码。首先把单位区间[0,1)按 a_1, a_2 出现的概率分为子区间[0,2/3)和[2/3,1)，根据 $x_1 = a_1$ 还是 $x_2 = a_1$ 来选取这两个子区间中的一个，如此继续。对于 $n=3$，整个[0,1)区间被分为 8 个不同长度的子区间，对于信源序列对应一个子区间，例如，$x_1 x_2 x_3 = a_1 a_2 a_0$，则应选中的区间为[4/9,16/27)。

如果对信源序列 $x_1 x_2 x_3$ 进行编码，因为信源序列与子区间一一对应，所以编码也就是对这些子区间给出标号。可以把与信源序列 $x_1 x_2 x_3$ 对应的码字 $\phi(x_1 x_2 x_3)$选为相应子区间中的点，例如，可以选这个子区间中比特数最少的二进制小数作为码字。例如，$x_1 x_2 x_3 = a_1 a_2 a_1$ 所对应的码字 $\phi(a_1 a_2 a_1)=1$，它是相应区间[4/9,16/27)中具有最少比特数的二进制小数，因为 1/2 属于该区间，所以它的二进制表示为 0.1，其中小数点以后部分是 1。图 3.2 最右边一列表示相应的码字，这样的编码一般不是前缀码，甚至当这些码字级联成码字序列时不是非常重要的，因为算术编码采用非常长、甚至是无限长的信源序列进行编码。它根据每次信源输出数据 x_i 来更新子区间，然后选子区间中的点来代表新序列，这些子区间是不重叠的，因而子区间中的点唯一地确定了相应的码字。可以发现，这些区间的划分点实际上是某种积累概率，子区间的宽度就是相应信源序列出现的概率，人们希望得到这些概率的递归计算方法。

如果信源输出字符表 X 和编码字符表 Y 只含有两个符号 $X=Y=\{0,1\}$，对于由 n 个信源符号组成的序列 x^n 按自然字典次序排序，即两个序列 $x^n = x_1 x_2 \cdots x_n$ 和 $y^n = y_1 y_2 \cdots y_n$，

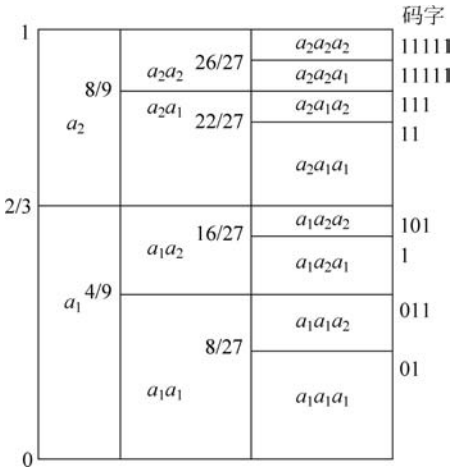


图 3.2 算术编码中信源输出序列与[0,1]中的一个子区间对应

假如 $x^n > y^n$ 是指

$$\sum_i x_i . 2^{-i} > \sum_i y_i . 2^{-i} \tag{3.15}$$

则对离散无记忆信源，有

$$p(x^n) \overset{\text{def}}{=} \prod_{i=1}^{n} p(x_i) \tag{3.16}$$

$$F(x^n) \overset{\text{def}}{=} \sum_{y^n < x^n} p(y^n) \tag{3.17}$$

信源序列 x^n 对应的子区间为$[F(x^n), F(x^n)+p(x^n))$，与 x^n 对应的码字把 $F(x^n)$ 按二进制小数展开，取其前 $l(x^n)$ 位，其中，$l(x^n)=\left\lceil \ln \frac{1}{p(x^n)} \right\rceil$，假如后面有尾数，就进位到第 $l(x^n)$ 位。这个码字记为 $C(x^n)$，即 $C(x^n) \in [F(x^n), F(x^n)+p(x^n))$，所以码字 $C(x^n)$ 与子区间$[F(x^n), F(x^n)+p(x^n))$相呼应。

算术编码的关键思想在于它有一套非常有效的计算信源序列 x^n 的出现概率 $p(x^n)$ 和累积概率 $F(x^n)$ 的递推算法。图 3.3 是高度为 n 的完整二元树，把 2^n 个长度为 n 的消息序列按照路径方式安排在 2^n 个树叶上，是按字典顺序来排列 x^n。此时，x^n 点左边的树叶上的概率和 $F(x^n)$ 可以看成是 x^n 左边所有子树的概率和。

若 $T_{x_1 x_2 \cdots x_{k-1}}$ 表示从支点 $x_1 x_2 \cdots x_{k-1}$ 长出的子树，该子树的概率为 $P(T_{x_1 x_2 \cdots x_{k-1}}) = x_1 x_2 \cdots x_{k-1}$。其中，$p(x_1 x_2 \cdots x_{k-1})$ 代表序列 $x_1 x_2 \cdots x_{k-1}$ 出现的概率。由式 3.17 可知，其累积概率为

$$F(x^n) = \sum_{y^n < x^n} p(y^n) = \sum_{\text{在}x^n\text{左边的}T} P(T) \tag{3.18}$$

第二个求和是对一切位于 x^n 左边的子树 T 求和。

例如，若输入二元符号序列 $x^n = 0111$，则

$$F(0111) = P(T_1) + P(T_2) + P(T_3) = P(00) + P(010) + P(0110)$$

若再在 $x^n = 0111$ 以后输入符号“0”，则由树图得到：

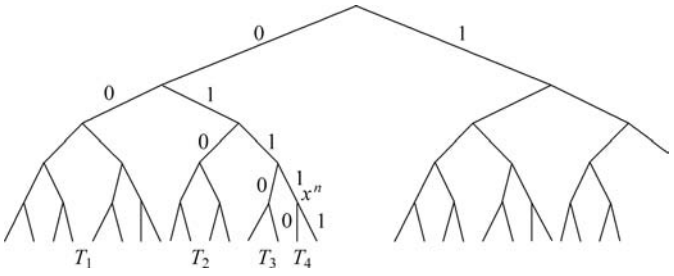


图 3.3　算术编码累积概率的计算树

$$F(x^n0)=P(x^{n+1})=F(01110)$$

若再在 $x^n=0111$ 以后输入符号“1”，则由树图得到：

$$\begin{aligned}F(x^n1)&=P(T_1)+P(T_2)+P(T_3)+P(T_4)\\&=P(00)+P(010)+P(0110)+P(01110)\end{aligned}$$

算术编码的优势就是当消息序列长度由 n 变为 $n+1$ 时，很容易计算序列概率 $p(x^nx_{n+1})$ 和累积概率 $F(x^nx_{n+1})$，所以编码可以序贯地进行，事实上

$$p(x^nx_{n+1})=p(x^n).p(x_{n+1}) \tag{3.19}$$

$$F(x^{n+1})=F(x^nx_{n+1})=\begin{cases}F(x^n), & x_{n+1}=0\\ F(x^n)+p(x^n), & x_{n+1}=1\end{cases} \tag{3.20}$$

不难看出，随着 n 的增长，子区间$[F(x^n),F(x^n)+p(x^n))$的下限单调增加，而上限单调减少，子区间长度区域为零。

3.2.4　香农编码

香农第一定理的证明过程给出了一种编码方法，称为香农编码，其编码方法是选择每个码字长度 L_i 满足

$$-\log(p_i)\leqslant L_i<-\log(p_i)+1 \quad (i=1,2,\cdots,q) \tag{3.21}$$

可以证明，这样的码长一定满足 Kraft 不等式，所以一定存在这样的码长唯一可译码和即时码。

香农编码的基本思想是概率匹配原则，即概率大的信源符号用短码，概率小的信源符号用长码，以减小平均码长，提高编码效率。香农编码的步骤如下。

(1) 将信源发出的 q 个消息，按其概率递减顺序排列。

(2) 计算各个消息的$-\log(p_i)$，确定满足式 3.21 的码字长度 L_i。

(3) 计算第 i 个消息的累积分布函数 $F_i=\sum_{k=1}^{i-1}p(s_k)$。

(4) 将累积分布函数 F_i 转换为二进制数。

(5) 取 F_i 二进制数的小数点后 L_i 位作为第 i 个符号的二进制码字 W_i。

在香农编码中，累积分布函数 F_i 将区间$[0,1)$分为许多互不相叠的小区间，每个信源符号 s_i 对应的码字 W_i 位于不同区间$[F_i,F_{i+1})$内。根据二进制小数的特性，在区间$[0,1)$，不重叠区间的二进制小数的前缀部分是不相同的，所以，这样编出来的码一定满足异前缀条

件，一定是即时码。

假如对有 6 个符号的信源序列进行香农编码，其信源概率为：

$$\begin{bmatrix} S \\ P(s) \end{bmatrix} = \begin{bmatrix} s_1 & s_2 & s_3 & s_4 & s_5 & s_6 \\ 0.2 & 0.19 & 0.18 & 0.17 & 0.15 & 0.11 \end{bmatrix}$$

下面以消息 s_3 为例介绍香农编码过程。

首先计算 $-\log(0.18)=2.47$，取整数 $L_3=3$ 作为 s_3 的码长。计算 s_1，s_2 的累积分布函数，有：

$$F_3=\sum_{k=1}^{2}p(s_k)=0.2+0.19=0.39$$

将 0.39 转换为二进制数 $(0.39)_{10}=(0.0110001)_2$，取小数点后面三位 011 作为 s_3 的代码。以此类推，其余符号的码字也可以通过计算得到，如表 3.1 所示。

表 3.1　香农编码结果

消息符号 s_i	消息概率 $p(s_i)$	累积分布函数 F_i	$-\log p(s_i)$	码字长度 L_i	码字 W_i
s_1	0.2	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	0.11	0.89	3.18	4	1110

香农编码是依据香农第一编码定理而来的，有着重要的理论意义。但香农编码的冗余度稍大，实用性不强。例如，信源有三个符号，概率分布分别是 0.5，0.4 和 0.1，根据香农编码方法求出的消息码长为 1，2 和 4，码字分别为 0，10 和 1110。但是同样的消息符号如果采用哈夫曼编码方法，则可以构造出平均码长更短的即时码 0，01 和 11。

3.2.5　哈夫曼编码

对于某一信源和某一码元集来说，若有一个唯一的可译码，它的平均码长不大于其他唯一可译码的平均长度，则称此码为最佳码，也称为紧致码。可以证明，最佳码具有以下性质。

(1) 若 $p_i>p_j$，则 $L_i\leqslant L_j$。即概率大的信源符号所对应的码长不大于概率小的信源符号所对应的码长。

(2) 对于二元最佳码，两个最小概率的信源符号所对应的码字具有相同的码长，而且这两个码字，除了最后一位码元不同以外，前面各位码元都相同。

1952 年，哈夫曼提出了一种构造最佳码的方法，所得的码字是即时码，且在所有的唯一可译码中，它的平均码长最短，是一种最佳变长码。

1. 二元哈夫曼

二元哈夫曼码的编码步骤如下：

(1) 将 q 个信源符号 s_i 按出现概率 $P(s_i)$ 递减次序排列起来。

(2) 取两个概率最小的符号，其中一个符号编为 0，另一个符号编为 1，并将这两个概率

相加作为一个新符号的概率，从而得到包含$(q-1)$个符号的新信源，称为缩减信源。

(3) 把缩减信源中的$(q-1)$个符号重新以概率递减的次序排列，重复步骤(2)。

(4) 依次继续下去，直至所有概率相加得到1为止。

(5) 从最后一级开始，从前返回，得到各个信源符号所对应的码元序列，即相应的码字。

例如，对某包含8个消息符号离散无记忆信源编二进制哈夫曼码，其概率分别为：

S	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
$P(s)$	0.2	0.19	0.18	0.17	0.15	0.1	0.007	0.003

编码过程如图3.4所示，各个符号的码长和码字计算结果如表3.2所示。

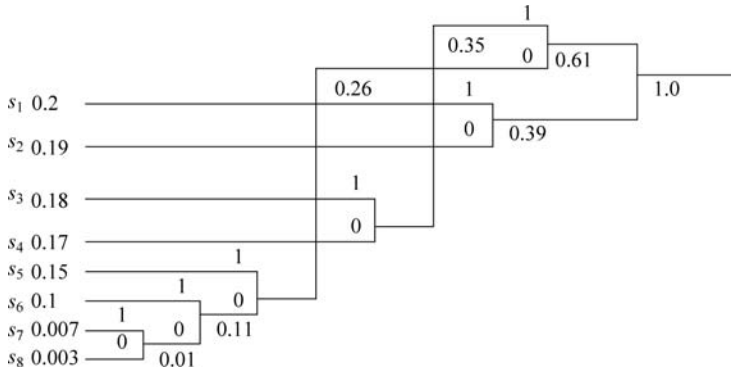


图3.4　该消息的哈夫曼编码

表3.2　哈夫曼编码结果

消息符号 s_i	码字长度 L_i	码字 W_i	消息符号 s_i	码字长度 L_i	码字 W_i
s_1	2	01	s_5	3	101
s_2	2	00	s_6	4	1001
s_3	3	111	s_7	5	10001
s_4	3	110	s_8	5	10000

从图3.4中读取码字时，一定要从后向前读，此时编出来的码字才是分离的异前置码。若从前向后读取码字，则码字不可分离。

该消息的熵：$H(S)=2.62$(比特/符号)

平均码长：$\bar{L}=2.73$(码元/符号)

编码效率：$\eta=95.8\%$

哈夫曼编码结果并不唯一。首先，由于0和1的指定是任意的，故由上述过程编出的哈夫曼码不是唯一的，但其平均长度总是一样的，故不影响编码效率。但每次排列必须严格按照大小次序，特别是最后一次只有两个概率，也要遵守相同的规则，不可以随意排列，否则会出现奇异码。其次，缩减信源时，若合并后的新符号概率与其他符号概率相等，从编码方法上来说，这几个符号的次序可任意排列，编出的码都是正确的，但得到的码字不相同，不同的编码方法得到的码字长度也不尽相同。

2. r 元哈夫曼码

上面讨论的二元哈夫曼编码方法可以推广到 r 元编码中来，不同的是每次将 r 个概率最小的符号合并成一个新的信源符号，并分别用 0，1，…，$(r-1)$ 等码元表示。

为了使短码得到充分利用，平均码长最短，必须使最后一步的缩减信源有 r 个信源符号，即构造出整树。因此，对于 r 元编码，信源的符号个数 q 必须满足

$$q=(r-1)\theta+r \tag{3.22}$$

其中，θ 表示缩减的次数，$(r-1)$ 为每次缩减所减少的信源符号个数，对于二元码 $(r=2)$，信源符号个数 q 必须满足 $q=\theta+r$。

因此，对于二元码，q 等于任何正整数时，总能找到一个 θ 满足式 3.22。而对于 r 元码，q 为任意整数时不一定能找到一个 θ 满足式 3.22。若 q 不满足时，不妨人为地增加一些概率为 0 的符号。假设增加 t 个信源符号 $s_{q+1}, s_{q+2}, \cdots, s_{q+t}$，并使它们对应的概率为零，即 $P(s_{q+1})=P(s_{q+2})=\cdots=P(s_{q+t})=0$。

假设 $n=q+t$，则 n 满足

$$n=(r-1)\theta+r \tag{3.23}$$

取概率最小的 r 个符号合并成一个新符号，并把这些符号的概率相加作为该节点的概率，重新按概率由大到小顺序排队，再取概率最小的 r 个符号合并，如此下去直至树根，这样得到的 r 元哈夫曼一定是最佳码。

下面给出 r 元哈夫曼编码步骤。

(1) 验证所给的 q 是否满足式 3.22，若不满足，可以人为地增加 t 个概率为零的符号，满足式 3.22，以使最后一步有 r 个信源符号。

(2) 取概率最小的 r 个符号合并成一个新符号，并分别用 0，1，…，$(r-1)$ 给各分支赋值，把这些符号的概率相加作为新符号的概率。

(3) 将新符号和剩下的符号重新排队，重复步骤(2)，如此下去直至树根。

(4) 取各树枝上的赋值，得到各符号的码字。

后来新加的概率为零的符号，虽也赋予码字，但因为概率为零，实际上并未用上，这样编成的码仍为最佳的，也就是平均码长最短。另外，等概率符号排队时应注意到顺序，以使码方差最小。

例如，对某包含 5 个消息符号离散无记忆信源编三元和四元哈夫曼码，其概率分别为

S	s_1	s_2	s_3	s_4	s_5
$P(s)$	0.4	0.3	0.2	0.05	0.05

三元哈夫曼编码过程如图 3.5 所示，各个符号的码长和码字计算结果如表 3.3 所示。四元哈夫曼编码过程如图 3.6 所示，各个符号的码长和码字计算结果如表 3.4 所示。

表 3.3　三元哈夫曼编码结果

消息符号 s_i	码字长度 L_i	码字 W_i	消息符号 s_i	码字长度 L_i	码字 W_i
s_1	1	0	s_4	2	21
s_2	1	1	s_5	2	22
s_3	2	20			

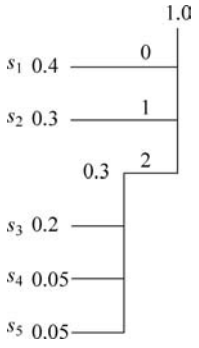


图 3.5　三元哈夫曼编码

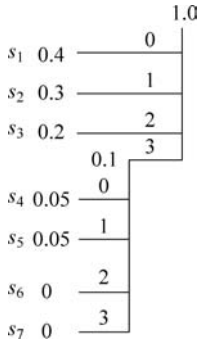


图 3.6　四元哈夫曼编码

表 3.4　四元哈夫曼编码结果

消息符号 s_i	码字长度 L_i	码字 W_i	消息符号 s_i	码字长度 L_i	码字 W_i
s_1	1	0	s_4	2	30
s_2	1	1	s_5	2	31
s_3	1	2			

要发挥哈夫曼编码的优势，一般情况下，信源符号数应远大于码元数。本例中，若编五元码，只能对每个信源符号赋予一个码元，相当于没有编码，当然无压缩可言。

3. 马尔可夫信源的哈夫曼编码

根据马尔可夫信源的特点，当前发出的符号的概率取决于当前的状态，同样的符号输出在不同状态下提供的信息量可能很大，也可能很小。如果类似于无记忆信源编码，一个信源符号对应于一个码字，将使编码效率降低。马尔可夫信源编码可以采用按状态编码。

设马尔可夫信源在 l 时刻的状态 $s_l \in E=\{E_1,E_2,\cdots,E_M\}$，输出符号 $x_l \in X=\{a_1,a_2,\cdots,a_q\}$，符号条件概率 $P(a_k/E_i)$，$k=1,2,\cdots,q$，$i=1,2,\cdots,M$。对每个状态 E_i，根据 $P(a_k/E_i)$进行哈夫曼编码。

例如，某一阶马尔可夫信源有 3 个状态，$E_1=a_1$，$E_2=a_2$，$E_3=a_3$，状态转移概率矩阵如式 3.24 所示。

$$[P(E_j/E_i)=\begin{bmatrix}1/2 & 1/4 & 1/4\\ 1/4 & 1/2 & 1/4\\ 0 & 1/2 & 1/2\end{bmatrix} \tag{3.24}$$

对三种状态 $E_1=a_1$，$E_2=a_2$，$E_3=a_3$ 进行哈夫曼编码如表 3.5 所示。

表 3.5　马尔可夫信源哈夫曼编码结果

状态	码字 $W(a_1)$	码字 $W(a_2)$	码字 $W(a_3)$
E_1	0	10	11
E_2	10	0	11
E_3		0	1

设马尔可夫信源输出的符号序列为 $x_1,x_2,\cdots,x_n$，信源的初始状态为 s_0，马尔可夫信源的编码过程如下。

(1) 给定初始状态 $s_0=E_i$，选取与状态 E_i 对应的码字 C_i，则获得信源符号 $x_1=a_k$ 对应的码字。

(2) 根据当前状态 E_i 和待发送的符号 a_k，得到下一个状态 $s_1=E_j$。选取与状态 E_j 对应的码字 C_j，编出信源符号 $x_2=a_k$ 对应的码字。

(3) 重复步骤(2)，直至处理完最后一个信源符号 x_k。

例如，信源初始状态为 $E_1=a_1$。输出符号序列为 $a_1a_3a_3a_2a_3a_2a_2a_1$，则编码输出为 01110110010。接下来讨论该信源的哈夫曼编码的编码效率和平均码长。

根据马尔可夫信源的状态转移概率，计算得到状态极限概率为：

$$P(E_1)=2/9,\quad P(E_2)=4/9,\quad P(E_3)=1/3$$

对马尔可夫信源的每个状态 E_i，根据式 3.25：

$$\bar{L}(E_i)=\sum_{k=1}^{q}p(a_k/E_i)L_k,\quad (i=1,2,\cdots,M;k=1,2,\cdots,q) \tag{3.25}$$

可以计算得它的平均码长为：

状态 E_1：$\bar{L}(E_1)=\sum_{k=1}^{3}p(a_k/E_1)L_k=\frac{1}{2}\times1+\frac{1}{4}\times2+\frac{1}{4}\times2=\frac{3}{2}$ 码元 / 符号

状态 E_2：$\bar{L}(E_2)=\sum_{k=1}^{3}p(a_k/E_2)L_k=\frac{1}{4}\times2+\frac{1}{2}\times1+\frac{1}{4}\times2=\frac{3}{2}$ 码元 / 符号

状态 E_3：$\bar{L}(E_3)=\sum_{k=1}^{3}p(a_k/E_3)L_k=\frac{1}{2}\times1+\frac{1}{2}\times1=1$ 码元 / 符号

所以该马尔可夫信源的哈夫曼编码的平均码长为：

$$\bar{L}=\sum_{i=1}^{M}p(E_i)\bar{L}(E_i)=\frac{2}{9}\times\frac{3}{2}+\frac{4}{9}\times\frac{3}{2}+\frac{1}{3}=\frac{4}{3}\text{ 码元 / 符号}$$

而该马尔可夫信源的极限熵为：

$$H_\infty=\frac{2}{9}H\left[\frac{1}{2},\frac{1}{4},\frac{1}{4}\right]+\frac{4}{9}H\left[\frac{1}{4},\frac{1}{2},\frac{1}{4}\right]+\frac{1}{3}H\left[0,\frac{1}{2},\frac{1}{2}\right]=\frac{4}{3}\text{ 比特 / 符号}$$

所以其编码效率为：

$$\eta=\frac{H_\infty}{\bar{L}}=1$$

由此可见，该编码方法能使得平均码长达到理论极限值——信源熵。所以对于马尔可夫信源来讲，它是一种最佳压缩的编码方法。

3.3 预测编码

3.3.1 预测编码的基本原理

若有一个离散信号序列，序列中各离散信号之间有一定的关联性，则利用这个序列中若干个信号作为依据，对下一个信号进行预测，然后将实际的值与预测的值的差进行编码。利

用过去的信号样值来预测当前值，并仅对当前样值的实际值与其预测值之差进行量化、编码后进行传输，这就是预测编码的基本原理。

1952 年，Bell 实验室的 C. C. Cutler 提出了差值脉冲编码调制系统，它是利用样本与样本之间存在的信息冗余来进行编码的一种数据压缩技术，其基本思想是：根据过去的样本去估算下一个样本信号的幅度大小，这个值称为预测值，然后对实际信号值与预测值之差进行量化编码，从而就减少了表示每个样本信号的位数。它与脉冲编码调制不同的是，PCM 是直接对采样信号进行量化编码，而差分脉冲编码调制（DPCM）是对实际信号值与预测值之差进行量化编码，存储或者传送的是差值而不是幅度绝对值，这就降低了传送或存储的数据量，可适应大范围变化的输入信号。DPCM 的基本出发点就是对相邻样值的差值进行量化编码。由于此差值比较小，可以为其分配较少的比特数，进而达到了压缩数码率的目的。

DPCM 差分脉冲编码系统的基本原理框图如图 3.7 所示。

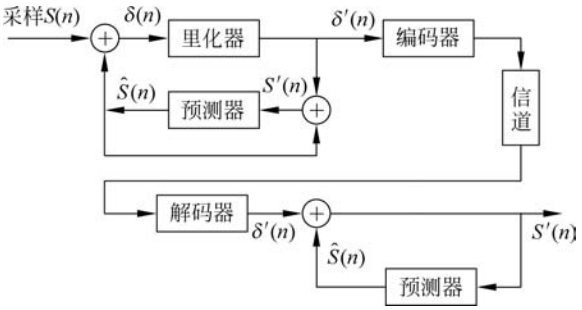


图 3.7　DPCM 编码系统原理图

图中，$S(n)$——信源数据；$S'(n)$——引入了量化误差的信源数据；$\hat{S}(n)$——预测器对的预测值；$\delta(n)$——量化误差；$\delta'(n)$——误差的量化值。

在 DPCM 系统中，发送端先发送一个初始值 S，然后就只发送预测误差 δ，接收端收到量化后的预测误差 δ' 和本地计算出的预测值 $\hat{S}$ 相加，得到恢复后的信号 S'。如果信道传输无误，则接收端重建信号 S' 与发送端原始信号 S 间的误差为 $S_k - S''_{kk} = S_k - (\hat{S}_k + \delta'_k) = (S_k - \hat{S}_k) - \delta'_k = \delta_k - \delta'_k = q_k$，这就是量化误差，即预测编码系统的失真来自发送端量化器。

预测器的设计是 DPCM 系统的核心问题，因为预测越准确，预测误差越小，码率就能压缩得越多。常用的线性预测公式是 $\hat{S}_k = \sum_{i=1}^{N} a_i(k) s'_i$，$N < k$，预测误差 $q_k = s_k - \sum_{i=1}^{N} a_i(k) s'_i$，为了使预测误差在某种测度下最小，要按照一定的准则对线性预测系数进行优化，采用最小均方误差准则来设计是很经典的方法。

下面主要介绍图像和视频中的预测编码技术。

3.3.2　帧内预测技术

帧内预测技术是利用图像信号的空间相关性来压缩图像的空间冗余，根据前面已经传送的同一帧内的像素值来预测当前的像素值。

将图像分割成为相互独立的块后，图像中的物体通常位于相邻区域的数个或数十个子块中。正是由于物体对象的这种全局性，造成了当前块与其相邻块的纹理方向往往是高度一致的，且各像素值相差不大。即不但子块内部的像素具有空间冗余，子块与子块间也存在空间冗余。另一方面，自然场景图像中的前景和背景通常具有一定的纹理特征，按其方向性可以划分为水平纹理、垂直纹理和倾斜纹理等。这两个特性为空域的帧内预测创造了条件。

在 MPEG-1/2 等编码标准中，帧内编码都采用直接做离散余弦变换、量化和熵编码的方法。在 H.263＋和 MPEGE-4 中，编码 I 帧采用了基于频域的帧内预测。H.264 中使用了精度更高的帧内预测算法，该算法基于空间的像素值进行预测，对于每个 4×4 块，每个像素都用 17 个最接近的已编码的像素的加权和来预测。这种帧内预测不是在时间上，而是在空间域上进行的预测编码算法，可以除去相邻块之间的空间冗余度，取得更有效的 I 帧压缩。有关 H.264 的帧内预测技术将在后面的章节中详细介绍。

3.3.3 帧间预测技术

对于视频图像序列，相邻帧之间存在着较强的相关性，即时间冗余。如果能够降低视频信号的时间冗余，就能够大幅提高视频序列的压缩效率。在当前的各类视频编码标准中，运动估计是去除时间冗余最基础和最有效的方法，也是各类视频编码算法所普遍采用的一项核心技术。运动估计的优劣直接决定编码效率和重构视频质量。一般而言，运动估计越准确，补偿的残差图像越小，编码效率也就越高，在相同码率下的解码视频就具有更好的图像质量；另一方面，运动估计计算复杂度占到了编码器的 50％以上，为了保证视频编解码的实时性，运动估计应当具有尽可能低的计算复杂度。因此，如何提高运动估计算法的性能，使得运动估计更快速、精确和健壮也一直受到学术界和工业界的广泛关注。

运动估计算法多种多样，大体上可以分为 4 类：块匹配法、递归估计法、贝叶斯估计法和光流法。其中，块匹配运动估计因其算法简单、便于 VLSI 实现等优点已被广泛应用，它已经被许多视频编码标准所采纳，如 MPEG-1、MPEG-2、MPEG-4、H.261、H.263 和 H.261 等。块匹配算法中，首先将图像分成 $N\times N$ 的宏块，并假设块内像素做相同的运动，且只做平移运动，然后用当前图像的每一个宏块在上一帧的一定范围内搜索或者穷举，得到一个最优的运动矢量。在块匹配运动估计算法中，全搜索算法精度最高，但是因为它要对搜索区域内的每个块进行检测，所以运算量太大，软硬件实现困难。人们相继提出了许多快速搜索算法，如三步法、四步法、二维对数法、基于块的梯度下降法、交叉法、菱形法和 MVFAST 算法等。下面简单介绍块匹配算法的基本原理。

通常情况下，自然场景的视频图像只有其中的部分区域在运动，同一场景相邻的两帧图像之间差异也不会太大。因而，编码器端无须将视频序列中每帧目标的运动信息告知解码器端，解码器就可以根据运动信息和前一帧图像的内容来更新当前帧图像，获得当前帧的真实数据，这样便能够有效地降低对每帧图像进行编码所需要的数据量。从序列图像中提取有关物体运动信息的过程就是运动估计，运动估计研究的主要内容就是如何快速、有效地获得足够精度的运动矢量，而把前一帧相应的运动部分信息根据运动矢量补偿过来的过程称为运动补偿。

块匹配运动估计的思想是将视频序列的每一帧都划分为大小相同、互不重叠的子块，为简单起见，做如下假设：子块内的所有像素具有运动一致性，并且只做平移运动，不包含旋

转、伸缩；在参考帧的一定范围内，按照一定的匹配准则搜索与之最接近的块。该预测块与到当前块的位移就是运动矢量，预测块和当前块之间的差值称为残差图像，因而每个原始图像宏块都可以使用残差块和一个运动矢量表示。预测越准确，意味着残差中的数值越小，编码后所占用的比特数也越少。利用搜索到的运动矢量在参考帧上进行运动补偿，补偿残差经 DCT 变换、量化、游程编码后与运动矢量共同经过熵编码，然后以比特流形式传送出去。图 3.8 显示了块匹配运动估计中当前宏块、预测块和运动矢量的关系。其中，Frame k 为当前帧，Frame $k-1$ 为参考帧，箭头所指示的即为该块的运动矢量。

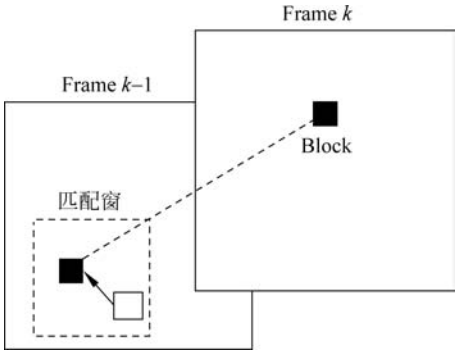


图 3.8　块匹配算法中与运动矢量的关系

匹配窗的大小通常设定为 $S=(M+2d)\times(N+2d)$，其中，M、N 是宏块的长和宽，d 为垂直和水平方向上的最大位移。例如，当宏块的最大矢量为(−7,7)时，则搜索范围为 15×15px。在传统的块匹配算法中，块的大小一般取为 16×16，这是一个折中的选择，因为在宏块大小的选择上存在以下两个矛盾。

(1) 宏块必须足够大，如果太小，很可能发生匹配到有相同像素值但与场景无关的块；并且因块小增加运算量，同时也增加了所需传输的运动矢量信息。

(2) 宏块必须足够小，如果在一个块里存在不同的运动矢量，匹配块就不能提供准确有效的估计。

1. 提高搜索效率的主要技术

运动估计算法的效率主要体现在图像质量、压缩码率和搜索速度三个方面。运动估计越准确，预测补偿的图像质量越高，补偿的残差就越小，补偿编码所需的位数就越少，比特率越小；运动估计速度越快，越有利于实时应用。提高图像质量、加快估计速度、减小比特率是运动估计算法研究的目标。通常，通过研究初始搜索点的选择、匹配准则、运动搜索策略来提高算法效率。

1) 初始搜索点的选择

(1) 直接选择参考帧对应的(0,0)位置。这种方法简单，但易陷入局部最优点。如果采用的算法初始步长太大，而原点又不是最优点，有可能使快速搜索跳出原点周围可能性比较大的区域而去搜索远距离的点，导致搜索方向的不确定性，故有可能陷入局部最优。

(2) 选择预测的起点。由于相邻块之间和相邻帧之间具有很强的相关性，许多算法都利用这种相关性先对初始搜索点进行预测，以预测点作为搜索起点。大量的实验证明，预测

点越靠近最优匹配点，即加强了运动矢量中心偏置分布，使得搜索次数越少。

下面举例说明起点预测的几种常见方法。

方法 1：基于求和绝对误差值的起点预测方法。分别求出当前块与其相邻块间的 SAD 值，然后选取 SAD 最小的块的运动矢量作为预测值。这种方法预测精度高，但计算 SAD 值的时间开销大。

方法 2：利用相邻块和相邻帧对应块的运动矢量来预测当前块的搜索起点。序列图像的运动矢量在空间、时间上具有很强的相关性。由于保存前一帧运动矢量信息在解码端要占用大量内存，这使得系统复杂化，故大多数算法仅考虑同帧块的空间相关性来预测运动。

方法 3：基于相邻运动矢量相等的起点预测方法。如果当前块的各相邻块的运动矢量相等，则以其作为当前块运动矢量的预测值；否则，使用方法 1 求出当前块与其相邻块间的 SAD 值，然后选取 SAD 最小的块作为预测起点。这种方法在保证精度的基础上利用运动矢量相关性大大减少了计算量。因为，在图像序列中存在大量的静止块和缓动块，而且属于同一对象的块在运动中常保持一致。

2）块匹配准则选取

运动估计算法中能够用的匹配准则有三种，即最小绝对差、最小均方误差和归一化互相关函数。

（1）最小绝对差。

$$\mathrm{MAD}(i,j)=\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left|f_k(m,n)-f_{k-1}(m+i,n+j)\right| \tag{3.26}$$

其中，(i,j)是位移矢量，f_k 和 f_{k-1} 分别是当前帧和上一帧的灰度值，$M\times N$ 为宏块的大小，若在某一个点(i_0,j_0)处 $\mathrm{MAD}(i_0,j_0)$达到最小，则该点是要找的最优匹配点。

（2）最小均方误差。

$$\mathrm{MSE}(i,j)=\frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left[f_k(m,n)-f_{k-1}(m+i,n+j)\right]^2 \tag{3.27}$$

最小的 MSE 值对应的是最优匹配点。

（3）归一化互相关函数。

$$\mathrm{NCCF}(i,j)=\frac{\sum_{m=1}^{M}\sum_{n=1}^{N}f_k(m,n)f_{k-1}(m+i,n+j)}{\left[\sum_{m=1}^{M}\sum_{n=1}^{N}f_k^2(m,n)\right]^{1/2}\left[\sum_{m=1}^{M}\sum_{n=1}^{N}f_{k-1}^2(m+i,n+j)\right]^{1/2}} \tag{3.28}$$

最大的 NCCF 值对应的是最优匹配点。在运动估计中，匹配准则对匹配的精度影响不是很大，由于 MAD 准则不需作乘法运算，实现简单、方便，所以使用的最多，通常使用 SAD 代替 MAD。SAD 的定义如下。

$$\mathrm{SAD}(i,j)=\sum_{m=1}^{M}\sum_{n=1}^{N}\left|f_k(m,n)-f_{k-1}(m+i,n+j)\right| \tag{3.29}$$

3）搜索策略

搜索策略选择得恰当与否对运动估计的准确性、运动估计的速度都有很大的影响。有关搜索策略的研究主要是解决运动估计中存在的计算复杂度和搜索精度这一对矛盾。目前运动估计快速搜索算法很多，下面将重点介绍几种典型算法的原理和搜索步骤。

2. 典型运动估计算法研究

目前，搜索精度最高的是全搜索法，但它的计算复杂度高，不宜实时实现，为此，研究人员提出了各种改进的快速算法。下面将对一些典型的运动估计算法进行简单介绍并分析各自的优缺点。

1) 全搜索法

(1) 算法思想：全搜索算法也称为穷尽搜索法，是对搜索范围内所有可能的候选位置计算 SAD(i,j)值，从中找出最小的 SAD(i,j)，其对应的偏移量即为所求运动矢量。此算法虽计算量大，但最简单、可靠，找到的必为全局最优点。

(2) 算法步骤。

① 从原点出发，按顺时针方向由近及远，在逐个像素处计算 SAD 值，直到遍历搜索范围内所有的点。

② 在所有的 SAD 中找到最小块误差点，该点所对应的位置即为最佳运动矢量。

(3) 算法分析。FS 算法是最简单、最原始的块匹配算法。由于该算法可靠，且能够得到全局最优的结果，通常是其他算法性能比较的标准，但它的计算量很大，这就限制了其在需要实时压缩场合下的应用，所以有必要进一步研究其他快速算法。

2) 三步搜索法

三步搜索法是应用相当广泛的一种次优的运动估计搜索算法，它的搜索区间一般为[−7,7]，即在候选区中与编码块相同坐标位置处为原点，将参考块在其上下左右距离为 7 的范围内，按照一定规律移动到一个位置就做匹配计算，它总共进行三步搜索，在下一次搜索时步长减半以前一步搜索得到的最优点为中心。其算法的中心思想是，采用一种由粗到细的搜索模式，从原点开始，按一定步长取周围 8 个点构成每次搜索的点群，然后进行匹配计算，利用上一步搜索得到的最小块误差 MBD 点作为当前搜索的中心位置，每做一步，搜索的步长减 1。图 3.9 为三步法的搜索示意图。

三步搜索法搜索窗选取[−7,7]，最多要做 25 个位置的匹配计算，相对于全搜索来讲，大大减少了匹配运算的复杂度，而且数据读取比较规则。三步法是一种比较典型的快速搜索算法，所以被研究得较多，后来又相继有许多改进的新三步法出现，改进了它对小运动的估计性能。

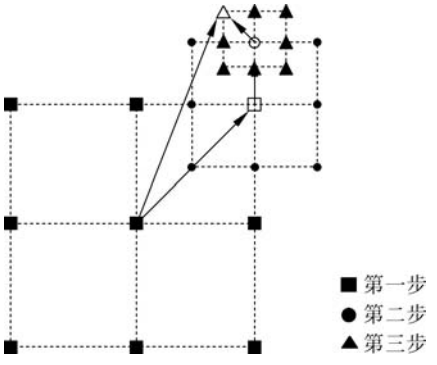


图 3.9　三步法的搜索示意图

3) 新三步法

TSS 假定运动矢量分布特点是在搜索窗口中均匀分布的，但事实证明运动矢量是偏置中心的，Renxiang Lin 等人在 TSS 的基础上提出了一种增强运动矢量中心偏置搜索和减小步长误差的新三步法。

NTSS 是对 TSS 的一个改进，对运动量比较小的视频序列如可视电话序列有比较好的性能。对于绝大多数的视频序列，运动矢量的分布都是在中心位置上的概率最大，随着与中心位置的距离增大，概率会急剧地下降，这也就是前面所说的运动矢量的中心偏移特性。运

动量比较小的视频序列的这一特性会更加明显。

NTSS 算法在最好的情况下只需要做 17 个点的匹配，在最坏的情况下需要做 33 个点的匹配，由于运动矢量中心偏置在现实视频序列中是普遍存在的，通常情况下，NTSS 算法需要做 33 点匹配的概率比较小，因此，在低速率视频应用中，如视频电话或视频会议中，NTSS 算法的优点可以得到更好的发挥。

图 3.10 为新三步搜索示意图。

4）四步搜索算法

四步搜索法是由 Po Lai-man Ma Wing-chung 等人提出的，FSS 也是基于视频序列图像的运动矢量的中心偏置特征，以原点为中心，在 5×5 大小的正方形中构造 9 个检测点的搜索模型。每一步将搜索模型的中心移向 MBD 点处，且后两步搜索模式取决于 MBD 点的位置。与 NTSS 一样，当运动较小时，FSS 也会很快结束搜索过程，只需要 2～3 步即可。

四步搜索算法考虑了块的中心匹配的特性，同时兼顾了物体的大范围运动。这种改进在物体既有小范围运动又有大范围运动时可以得到较好的性能。它在搜索速度上不一定快于 TSS，搜索范围为[－7，7]时，FSS 最多需要进行 27 次块匹配。但是 FSS 计算复杂度比 TSS 低，它的搜索幅度比较平滑，不至于出现方向上的误导，所以获得了较好的搜索效果。这种效果同样出现在摄像机镜头伸缩、有快速运动物体的图像序列中，因此，这是一种吸引人的运动估值算法。

图 3.11 为四步搜索法示意图。

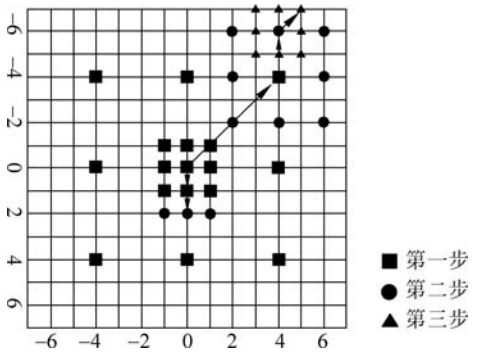


图 3.10　新三步搜索示意图

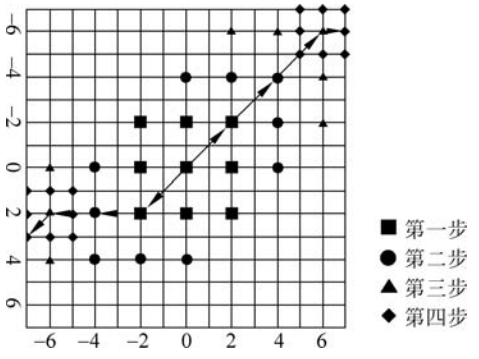


图 3.11　四步搜索法示意图

5）菱形搜索算法

菱形搜索算法最早由 Shan Zhu 和 Kai-Kuang Ma 两人提出，经过多次改进，已成为目前快速块匹配运动估计算法中性能最好的算法之一。1999 年 10 月，菱形搜索算法（DS）被 MPEG-4 国际标准采纳并被收入验证模型（VM）。

菱形搜索模板的形状和大小不但影响整个算法的运行速度，而且也影响它的性能。块匹配的误差实际上是在搜索范围内建立了误差表面函数，全局最小点即对应着最佳运动矢量。由于这个误差表面通常并不是单调的，所以搜索窗口太小，就容易陷入局部最优。例如，BBGDS 算法，其搜索窗口仅为 3×3。若搜索窗口太大，又容易产生错误的搜索路径，如 TSS 算法的一步。另外，统计数据表明，视频图像中进行运动估值时，最优点通常在零矢量周围（以搜索窗口中心为圆心，2px 为半径的圆内）。

基于这两点事实，DS 算法采用了两种搜索模板，分别是 9 个检测点的大模板（Large Diamond Search Pattern）和有 5 个检测点的小模板（Small Diamond Search Pattern）。搜索先用大模板计算，当最小块误差 MBD 点出现时，将大模板 LDSP 换为 SDSP，再进行匹配计算，这时 5 个点中的 MBD 即为最优匹配点。图 3.12 显示了一个用 DS 算法搜索到运动矢量（－4，－2）的例子。搜索共有 5 步，MBD 点分别为（－2，0）、（－3，－1）、（－4，－2），使用了 4 次 LDSP 和 1 次 SDSP，总共搜索了 24 个点。

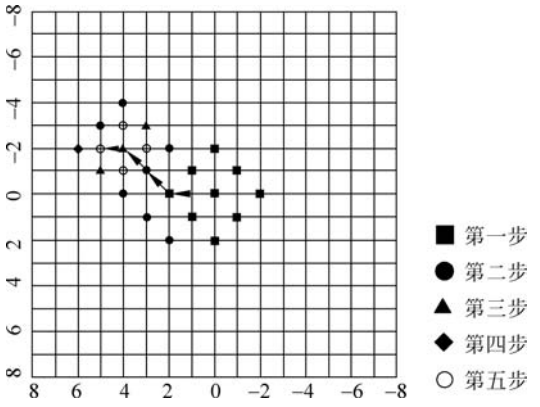


图 3.12　菱形搜索法图示

DS 算法的特点在于它分析了视频图像中运动矢量的基本规律，选用了大小两种形状的搜索模板 LDSP 和 SDSP。先用 LDSP 搜索，由于步长大，搜索范围广，可以进行粗定位，使搜索过程不会陷于局部最小；当粗定位结束后，可以认为最优点就在 LDSP 周围 8 个点所围的菱形区域中，这时再用 SDSP 来准确定位，使搜索不至于有大的起伏，所以它的性能优于其他算法。另外，DS 搜索时各步骤之间有很强的关联性，模板移动时只需要在几个新的检测点处进行匹配计算，所以也提高了搜索速度。

6）六边形搜索算法

六边形（Hexagon-based Search，HEXBS）算法于 2002 年提出，针对 DS 算法的不足，提出了六边形的大模板结构。在菱形搜索算法的大菱形模板（LDSP）中，四周的 8 个匹配点到中心点位置距离是不同的：水平和垂直方向的相邻搜索点间距为 2px，而中心点和对角方向的相邻搜索点间距为$\sqrt{2}$px。因此，使用 LDSP 进行粗定位时，沿不同方向移动的匹配速度也是不同的，当 LDSP 的顶点为本次匹配的 MBD 点时，模板沿对角方向移动时，其速度为$\sqrt{2}$像素/步。另一方面，在大模板移动的每一步中，不同的搜索方向上只需检测 3 个新的搜索点即可。根据 LDSP 模板的问题，在 HEXBS 算法中，用六边形模板（HSP）代替 DS 算法中的 LDSP 模板，该模板在各个搜索方向都具有相同的梯度下降速度，搜索速度优于 LDSP 模板。

六边形搜索法的两个搜索模板如图 3.13 所示，其中，图 3.13(a)为六边形模板（HSP），图 3.13(b)为小菱形模板（SDSP）。搜索时先用 HSP 进行计算，当 MBD 点出现在中心处时，可认为最优点位于 HSP 所包围的六边形区域内，此时将 HSP 换为 SDSP 时，这 5 个点中的 MBD 就是最优匹配点。

用 HSP 在搜索区域中心及周围 6 个点处进行匹配计算，若 MBD 点位于中心点，则以

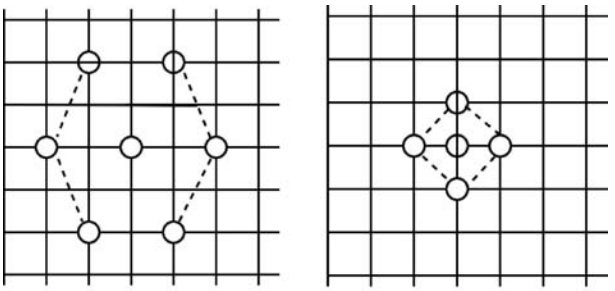


图 3.13 HEXBS 算法模板

上次找到的 MBD 点作为中心点，用新的 HSP 来计算，若 MBD 点位于中心点内，则以上一次找到的 MBD 点作为中心点，将 HSP 换为 SDSP，在 5 个点处计算，找出 MBD 点，该点所在位置即对应最佳运动矢量。

六边形模板包含 7 个搜索点，比 LDSP(9 个搜索点)减少了两个搜索点，因此，在粗定位的过程中比 LDSP 的计算复杂度低。HSP 比 LDSP 更接近于圆，其水平方向的顶点到中心点距离为 2px，对角方向的顶点距离中心点为$\sqrt{5}$ px。使用 HSP 在匹配窗内移动时，在各个方向上的移动也非常接近，沿水平方向模板的梯度下降速度为 2 像素/步，在对角方向为$\sqrt{5}$像素/步，高于 LDSP 的搜索速度，且新增搜索点数与方向无关，无论本次 MBD 点位于模板的何处位置，下次匹配只存在 3 个新匹配点需要计算。从上述分析可以看出，HEXBS 算法因为六边形模板的引入，所以其搜索速度比 DS 有所提高，但搜索精度与 DS 相比略有下降。

3. 新型运动估计技术

当前运动估计技术正在向分数像素精度、多参考帧和可变块等方向发展，在新一代视频编码标准 H.264 中充分利用了这几种运动估计技术。H.264 对一个宏块理论上的运动估计过程如下：对宏块所有允许划分下的每一子块、在所有允许的参考帧中进行分数像素精度运动估计，并根据估计结果选择最佳参考帧、最佳宏块划分以及与之对应的估计结果。这些新技术能有效地提高运动估计的精度、提高编码效率，但也增加了整个编码器的计算复杂度。下面对 H.264 中使用的这三种技术进行简略介绍。

1) 分数像素精度运动估计

一般而言，运动估计越精确，则对应的参考图像中的零值越多，因而编码后所占用的比特数就会更小，从而提高编码效率。在 H.263 中，提出了使用半像素精度的运动估计技术，H.264 中明确提出了运动估计采用亚像素运动估计的方法，并制定了 1/4px 精度和 1/8px 精度可选的运动估计方法。

下面以 1/2px 精度为例说明分数像素精度的运动估计搜索过程：首先在搜索窗内利用前面介绍的经典运动估计算法进行整像素精度的运动估计，得到整像素级的最优运动矢量；然后在整像素最优运动矢量点周围进行 1/2px 精度的插值，并对得到的 8 个 1/2px 点进行 SAD 计算，并与整像素点的 SAD 值进行比较，得到当前的最优匹配点，此最优匹配点即为 1/2px 精度的运动矢量。如果要得到更加精确的运动矢量，则在前一步的基础上继续进行

插值，并进行 SAD 比较，得到最佳匹配点。

2）多参考帧运动估计

多参考帧运动估计就是在进行运动估计过程中可以参考前面已经编码的多幅数据帧，在所有可能的参考帧中搜索出率失真最小的宏块和运动矢量。H.264 标准中规定，对 P 帧和 B 帧编码时最多可采用 5 个参考帧进行帧间预测。一般多参考帧运动估计首先从距离当前编码帧最近的参考帧开始，一直搜索到距离当前编码帧最远的参考帧为止。

3）可变块运动估计

固定块匹配算法在运动估计中遇到下面几个问题。当匹配块较小时，块匹配准确，残差数据量减小，但块的数量增加，运动矢量的数据量增加；当匹配块较大时，块的数量减小，运动矢量的数据量减小，但块匹配准确度下降，残差数据量增加。若采用 16×16 固定的块的大小，对于背景区域的块，可以找到较小误差的匹配，但对于运动区域的块则不能。为了解决上述问题，学者提出了可变块运动估计技术，其核心思想是，对运动不剧烈的背景区域的块使用较大的块进行运动估计，而对于运动较剧烈的前景对象则使用较小的块进行运动估计。可变块运动估计的块模式如图 3.14 所示。

在 H.264 标准中每个宏块可根据语法分为 16×16，8×16，16×8 和 8×8 的块；当采用 8×8 块时，还可以进一步分为更小的 8×4，4×8 和 4×4 子块进行运动估计。在对图像进行帧间编码时（P 帧和 B 帧），编码器需要对整个预测模式集合{SKIP，16×16，8×16，16×8，8×8，8×4，4×8，4×4}的每种分割模式单独进行运动估计，得到各自的运动矢量。采用可变块进行帧间预测，使得运动估计模型能够更接近物体的实际运动，因此运动补偿精度更高，使用这种方法比单独 16×16 块的预测方法提高大约 15%的编码效率。

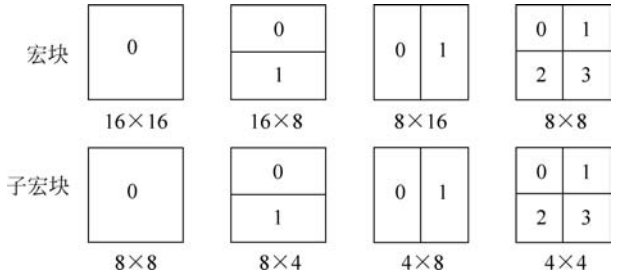


图 3.14　H.264 标准中的可变块运动估计的块模式

3.4　变换编码

3.4.1　变换编码工作原理

变换编码不是直接对空域图像信号进行编码，而是首先将空域图像信号映射变换到另一个正交矢量空间（变换域或频域），产生一批变换系数，然后对这些变换系数进行编码处理。变换编码是一种间接编码方法，其中关键问题是在时域或空域描述时，数据之间相关性大，数据冗余度大，经过变换在变换域中描述，数据相关性大大减少，数据冗余量减少，参数独立，数据量少，再进行量化，编码就能得到较大的压缩比。

为什么信号通过正交变换就能压缩数据量呢？下面通过一个例子来说明。

x_1、x_2 为两个相邻的数据样本，每个样本有 $2^3=8$ 个幅度等级，用 3b 编码。两个样本的联合事件，有 $8\times8=64$ 种可能性，如图 3.15 所示。由于信号变换缓慢，x_1、x_2 同时出现相近等幅等级的可能性较大，故图 3.15 阴影区内 45°斜线 $x_1=x_2$ 附近的联合事件出现的概率也就较大，将此阴影区的边界称为相关圈：信源的相关性越强，相关圈就越“扁长”，x_1 与 x_2 呈现出“水涨船高”的紧密关联特性，此时编码圈内各点的位置，就要对两个差不多大的坐标值分别进行编码；信源的相关性越弱，此相关圈就越“方圆”，说明 x_1 处于某一幅度等级时，x_2 可能出现在不相同的任意幅度等级上。

现在若对该数据进行正交变换，从几何上相当于把如图 3.15 所示的(x_1，x_2)坐标系旋转 45°变换成(y_1，y_2)坐标系。那么此时该相关圈正好处于 y_1 上的投影就越大，而在 y_2 上的投影则越小。因而从 y_2 坐标来看，任凭 y_1 在较大范围内变换，而 y_2 却巍然不动或者仅仅微动，这就意味着变量 y_1 和 y_2 之间的联系，在统计上更加相互独立。经过多维坐标系中适当的旋转和变换，能够把散布在各个坐标轴上的原始数据在新的、适当的坐标系中集中到少数坐标轴上。因此可能用较少的编码维数来表示一组信号样本，实现高效率的压缩编码。

预测编码和变换编码都是利用去除信号自身的相关性消除冗余，实现数据压缩，但不同的是，预测编码直接在空间域内进行，而变换编码则在变换域内进行。这是因为变换后的系数矩阵更有利于压缩，在空间域，相邻采样点具有很强的相关性，能量一般平均分布，而利用正交变换的去相关性和能量集中性，就能去除数据间的相关性，将信号能量聚集到少数变换系数上，这便于在其后的量化过程中保留小部分的重要系数，去除大部分不重要的系数。

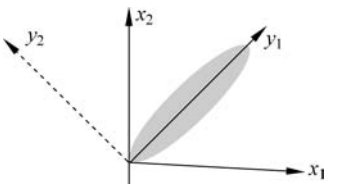


图 3.15　正交变换的几何意义

正交变换的种类很多，典型的准最佳变换有 DCT(离散余弦变换)、DFT(离散傅里叶变换)、WHT(Walsh Hadama 变换)、HrT(Haar 变换)等。基于变换的多媒体压缩编码技术已经有近 40 年发展历史，技术已经非常成熟，已广泛应用于各种图像、音频、视频等多媒体数据的压缩，现有的视频和图像编码标准中几乎都使用了变换编码的思想。

3.4.2　最佳的正交变换——K-L 变换

K-L(Karhunen-Loeve)变换是以多媒体数据的统计特性为基础的一种正交变换，其变换核矩阵是与待处理的数据相关的，即其变换核是变化的。由于其通过分析待处理数据统计特性来设计 K-L 变换核，所以其变换后的数据独立性更强。K-L 变换具有以下两个性质。

(1) 矢量信号的各个分量互不相关，即变换域信号的协方差矩阵为对角线型。

(2) K-L 变换是在均方误差准则下失真最小的一种最佳变换。

变换的选择很大程度上影响着编码系统的整体性能，最大限度地集中信号功率，取得最大的压缩效率。如果信号是一个平稳随机过程，经过 K-L 变换后，所有的变换系数都不相关，且数值较大的方差仅存于少数系数中，这就利于在一定的失真度限制下，将数据压缩至最小。虽然 K-L 变换具有均方误差准则下的最佳性能，但是要先知道信源的协方差矩阵，

并求特征值和特征向量，运算量相当大，即使借助计算机求解，也很难满足系统的实时性要求，因此 K-L 变换实用性差，一般情况下只是作为衡量其他正交变换性能的参考方式来使用。

对矢量信号 $\boldsymbol{X}$ 进行 K-L 变换的具体过程是：首先求出矢量信号 $\boldsymbol{X}$ 的协方差矩阵 $\phi_{\boldsymbol{X}}$，然后求出 $\phi_{\boldsymbol{X}}$ 的归一化正交特征向量 q_i 所构成的正交矩阵 $\boldsymbol{Q}$，最后用矩阵 $\boldsymbol{Q}$ 对该矢量信号进行正交变换 $\boldsymbol{Y}=\boldsymbol{Q}\boldsymbol{X}$。下面举例说明 K-L 变换的过程。

若已知随机信号 $\boldsymbol{X}$ 的协方差矩阵 $\phi_{\boldsymbol{X}}=\begin{bmatrix}110\\110\\001\end{bmatrix}$，求正交矩阵 $\boldsymbol{Q}$。

(1) 根据 $\phi_{\boldsymbol{X}}$ 求特征值。

令 $\begin{vmatrix}1&-\lambda&1&0\\1&1&-\lambda&0\\0&0&1&-\lambda\end{vmatrix}=0$，则有 $\lambda_1=2,\lambda_2=1,\lambda_3=0$。

(2) 求特征向量。

由 $\phi_{\boldsymbol{X}}q_i=\lambda_i q_i(i=1,2,3)$ 得到三个方程组。

当 $\lambda_1=2$ 时，有 $\begin{bmatrix}q_{11}+q_{12}\\q_{11}+q_{12}\\q_{13}\end{bmatrix}=2\begin{bmatrix}q_{11}\\q_{12}\\q_{13}\end{bmatrix}$，得到 $q_1=\begin{bmatrix}a\\a\\0\end{bmatrix}$；同理得到，$q_2=\begin{bmatrix}0\\0\\b\end{bmatrix}$、$q_3=\begin{bmatrix}c\\-c\\0\end{bmatrix}$。

其中，待定实常数可由归一化正交条件解得：$a=\frac{\sqrt{2}}{2},b=1,c=\frac{\sqrt{2}}{2}$。

(3) 归一化正交矩阵：

$$\boldsymbol{Q}=[q_1q_2q_3]^{\mathrm{T}}=\frac{\sqrt{2}}{2}1\begin{vmatrix}1&1&0\\0&0&\sqrt{2}\\0&-1&0\end{vmatrix}$$

3.4.3　离散余弦变换 DCT

DCT 是在图像和视频编码中应用最多的一种变换方法，DCT 的实质是通过线性变换 $\boldsymbol{X}=\boldsymbol{H}\boldsymbol{x}$，将一个 N 维向量 $\boldsymbol{x}$ 变换为变换系数向量 $\boldsymbol{X}$。DCT 变换核 $\boldsymbol{H}$ 的第 k 行第 n 列的元素定义为：

$$\boldsymbol{H}(k,n)=c_k\sqrt{\frac{2}{N}}\cos\frac{(2n+1)k\pi}{2N} \tag{3.30}$$

其中，$k=0,1,\cdots,N-1,n=0,1,\cdots,N-1,c_0=\sqrt{2},c_k=1$。由于 DCT 是线性正交变换，因此 DCT 是完全可逆的，且逆矩阵就是其转置矩阵。

为了降低运算量，图像编码中一般将图像分为相互独立的子块，以子块为单位做二维 DCT，二维 $N\times N$ 点 DCT 公式为：

$$F(u,v)=\frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \tag{3.31}$$

逆变换 IDCT 公式为：

$$f(x,y)=\frac{2}{N}C(u)C(v)\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}F(u,v)\cos\frac{(2x+1)u\pi}{2N}\cos\frac{(2y+1)v\pi}{2N} \quad (3.32)$$

其中，x,y 是空间坐标，$x,y=0,1,\cdots,N-1$；u,v 是 DCT 空间坐标，$u,v=0,1,\cdots,N-1$。可变系数 $C(0)=1/\sqrt{2}$，$C(i)=1$，$i=1,2,\cdots,N-1$。

下面将对现有几种 DCT 变换的编程实现方法进行比较分析。

1. 直接变换公式的算法实现

用最直接的算法实现 8×8 点 DCT 变换，如下。

```
for(u = 0;u < 8;u++)
  for (v = 0;v < 8;v++)
      for(x = 0;x < 8;x++)
        for(y = 0;y < 8;x++)
          MAT(F,u,v) += MAT(F,u,v) * c * cos(2 * x + 1) * u * pi/2(nrow)) *  cos(2 * y + 1) * v *
pi/(2 * ncol));
```

其中，MAT(m,i,j)表示 8×8 点矩阵 $\boldsymbol{m}$ 中第 i 行第 j 列的元素。

从算法可见，一共需要 8×8×8×7＝3584 次加法。用最直接的算法实现需要巨大的计算量，不具有使用价值。

2. 二维 DCT 变换分解为一维形式

二维 DCT 变换可以转换为一维形式，即 8×8 的二维 DCT 变换可以等效为 8 行一维的 8 点 DCT 变换和 8 列一维的 8 点 DCT 变换，算法实现如下。

```
for(v = 0;v < 8;v++)
  for (u = 0;u < 8;u++)
      for(x = 0;x < 8;x++)
        MAT(tempF,u,v) += MAT(f,x,v) * coff * cos(2 * x + 1) * u * pi/2(nrow));
  for(u = 0;u < 8;u++)
  for (v = 0;v < 8;v++)
      for(y = 0;y < 8;x++)
        MAT(F,u,v) += coff * MAT(tempF,u,y) * c * cos(2 * y + 1) * v * pi/(2 * ncol));
```

其中，MAT(m,i,j)表示 8×8 点矩阵 $\boldsymbol{m}$ 中第 i 行第 j 列的元素。

从算法可见，先 8 行一维 DCT 变换需要 64×8 次乘法和 56×8 次加法，再 8 列一维 DCT 变换需要 64×8 次乘法和 56×8 次加法，共需要 1024 次乘法和 896 次加法，计算量减少了原来的 3/4。

3. AAN 快速算法

AAN 快速算法是 Y. Arai、T. Agui 和 M. Nakjima 于 1988 年提出的一种快速算法。它也是将二维 DCT 分解成行列的一维变换，一维 8 点的 DCT 通过 16 点离散傅里叶变换来实现，而 16 点 DFT 又可以通过快速傅里叶变换(FFT)实现。从一维 ANN 变换的算法流程可以看出，其一维 8 点变换只需 11 次乘法和 29 次加法，如果将最后的尺度变换和量化结合在一起，则变换部分只需 5 次乘法和 29 次加法。二维 8×8 点 DCT 采用此方法需要 16×5＝80 次

乘法和 16×29＝464 次加法。

3.4.4　小波变换

小波变换(Wavelet Transform,WT)是一种新的变换分析方法,它继承和发展了短时傅里叶变换局部化的思想,同时又克服了窗口大小不随频率变化等缺点,能够提供一个随频率改变的“时间-频率”窗口,是进行信号时频分析和处理的理想工具。它的主要特点是通过变换能够充分突出问题某些方面的特征,能对时间(空间)频率的局部化分析,通过伸缩平移运算对信号(函数)逐步进行多尺度细化,最终达到高频处时间细分,低频处频率细分,能自动适应时频信号分析的要求,从而可聚焦到信号的任意细节,解决了傅里叶变换的困难问题,成为继傅里叶变换以来在科学方法上的重大突破。

小波分析用于数据或图像的压缩,目前绝大多数是对静止图像进行研究的。面向网络的活动图像压缩,长期以来主要是采用离散余弦变换(DCT)加运动补偿(MC)作为编码技术,然而,该方法存在两个主要的问题:方块效应和蚊式噪声。利用小波分析的多尺度分析,不但可以克服上述问题,而且可首先得到粗尺度上图像的轮廓,然后决定是否需要传输精细的图像,以提高图像的传输速度。因此,研究面向网络的低速率图像压缩的小波分析并行算法,具有较高的探索性和新颖性,同时也具有较高的应用价值和广泛的应用前景。下面给出小波变换的两种快速算法:Mallat 算法和提升算法。

1. Mallat 算法

小波变换的主要算法是由法国科学家 Stephane Mallat 在 1988 年提出的。他在构建正交小波基时提出了多分辨率的概念,从空间上形象地说明了小波的多分辨率的特性,提出了正交小波的构造方法和快速算法,叫作 Mallat 算法。该算法统一了在此之前构造正交小波基的所有方法,它的地位相当于快速傅里叶变换在经典傅里叶分析中的地位,为小波变换的实际应用奠定了坚实的基础,极大地促进了小波变换在数字信号处理中的工程应用。Mallat 算法也是在小波的实际软硬件实现中最常用的算法之一。采用 Mallat 算法对信号的正交小波分解与合成的原理如图 3.16 所示。Mallat 算法通过低通滤波器 h 和高通滤波器 g 对输入信号 X 进行滤波,然后对输出结果进行下二采样来实现正交小波分解,分解的结果是产生长度减半的两个分量,一个是经低通滤波器产生的原始信号平滑部分,另一个则是经高通滤波器产生的原始信号细节部分。重构时先要进行上二采样,再使用一组合成滤波器 $\tilde{h}$ 和 $\tilde{g}$ 对小波分解的结果滤波,以生成重构信号 Y。Y 相对于 X 来说可能是有损的,也可能是无损的,这取决于所采用的小波变换核(小波基)。多级小波分解可以通过级联的方式进行,每一级的小波变换都是在前一级分解产生的低频分量上的继续,而合成是分解的逆运算。

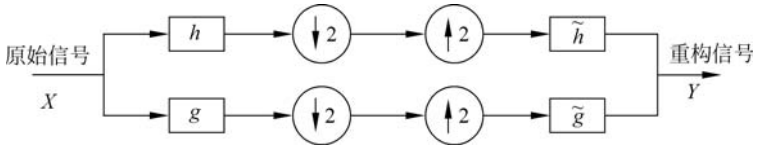


图 3.16　Mallat 算法的分解与合成

2. 小波提升算法

Swelden 提出了一种不依赖于傅里叶变换的新的小波构造方案——提升方案(Lifting Scheme),由于其计算法复杂度只是原有卷积方法的一半左右,因而成为计算离散小波变换的主流方法。提升方案为小波变换提供了一种新的更快速的实现方法。同时提升不但是构造第一代小波的难度,并且已经证明提升可以实现所有第一代小波变换。利用提升方案可以构造出不同的小波,例如,Daubechies 双正交小波和差值双正交小波。

提升方案的特点如下。

(1) 继承了第一代小波的多分辨率的特性。

(2) 不依赖傅里叶变换。

(3) 提升方案允许完全的原位计算,即在小波变换中不需要附加内存,原始信号数据可以直接被小波系数替换。

(4) 提升方案的反变换可以很容易由正变换得到,即只需要改变数据流的方向和正负号。小波提升方案由于其计算速度快、占用内存少、可以实现整数变换等特点而被 JPEG2000 标准推荐用来进行小波变换,是 JPEG2000 的核心算法之一。小波提升方案通过预测和更新两个提升环节实现信号高低频的分离。由于信号具有局部相关性,某一点的信号值可以根据相邻信号的值由适当的预测算子预测出来,而这种预测所产生的误差就是高频信息,这个过程称为预测环节。预测环节得到的高频信息又是通过更新算子来调整信号的下抽样,从而得到低频信息,这个过程称为更新环节。更新环节在提升算法中称为原始提升,而预测环节被称为对偶提升。

小波提升通常由一个预测环节和一个更新环节构成,图 3.17 表示了一个最简单的提升变换和提升反变换。提升变换和提升反变换结构对称、算子符号相反,由此可以保证提升变换是一种完全可逆的变换,可以实现精确重构。提升环节可以通过使用著名的欧几里得算法分解已经存在的小波滤波器得到,也可以完全重新构造。

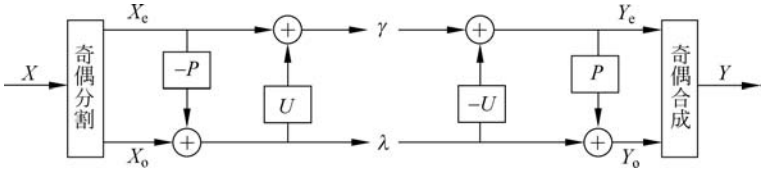


图 3.17 小波提升变换和反变换示意图

图 3.17 中 X 为输入信号,Y 为重构信号,λ 为小波系数(对应高频分量),γ 为尺度系数(对应低频分量),P 为预测算子,U 为更新算子,下标 e 表示偶数序列,下标 o 表示奇数序列。

通常,输入信号具有局部相关性,即某一点的信号值跟它相邻点的值是很相关的。可以利用这一点通过 X_e 来预测 X_o 的值,当然这个过程需要借助预测算子 P 来完成。然而,不管如何精巧地去选择 P,预测后得到的值跟真实的 X_o 总会有误差,这个误差就是小波系数(高频信息),所以有 $\lambda = X_o - P[X_e]$。

在小波变换中,除了要得到高频信息外,还需要低频信息。当然,不能仅让输入信号的下抽样 X_e 来表示低频信息,因为它没有反映 X_o 的信息。所以,需要借助得到的高频信息

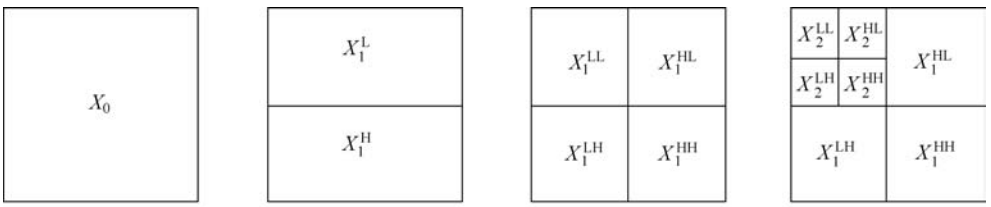
来修正 X_e，使之能充分反映 X_e 和 X_o 的影响，正确地表达信号的低频信息，而完成这个修正任务的就是更新算子 U。

小波提升的核心是更新算子和预测算子，通过预测算子可以分离出高频信息，而通过更新算子可以得到正确的低频信息。提升方案可以实现原位计算和整数提升，并且变换的中间结果是交织排列的，其中，原位计算和整数提升在硬件实现中很有价值，而交织排列则造成了对中间结果进行分析的稍许不便。另外，提升反变换是只需要将正变换中算子的符号改变并按照相反的顺序进行运算即可。

在提升变换中，预测环节实现了高频分量的分离，而更新环节则可以使得低频分量能正确地表达原始信号的统计特征。实际上，如果仅从可重构的角度来看，P 和 U 可以是任意的，而且也不需要同时使用。但考虑到实际的高频与低频的物理意义，也可以对中间结果进一步分析与处理，预测算子 P 应该以尽量平滑地拟合原输入信号为己任，而更新算子 U 则应以保持原信号的连续均值为目标。

3. 二维小波变换原理

当输入信号为二维信号时(如图像信号)，小波变换要在二维上进行。在图像编码中使用的二维离散小波变换等同于一个分层的子带系统，各子带的频率按对数划分，表示二倍程分解。因为二维小波可分离构造，也就是说，二维小波变换可通过一次列变换，如图 3.18(b)所示，然后再经过一次行变换，如图 3.18(c)所示，即总共两次一维小波变换来实现。这样一幅图像在二维频域可以分解为 4 个子带。进一步地，可以将 X_1^{LL} 分解为 4 个子带 X_2^{LL}、X_2^{LH}、X_2^{HL} 和 X_2^{HH}，如图 3.18(d)所示。就这样，分别对图像的列、行实施一维小波分解，并迭代以形成多级塔形分解结构。在多级塔形分解结构中，不同级的子带系数实质上反映了图像在不同尺度下的低频和高频分量。分解的级数越高，则相应子带对应的尺度越大，而子带本身的大小就越小，因为每进行一次分解都要进行下抽样，级数越高则抽样次数越多，子带自然就越小。



(a) 原始信号　(b) 小波变换(列)　(c) 小波变换(行)　(d) 二级小波变换

图 3.18　二维小波变换示意图

3.5　其他编码方法

3.5.1　矢量量化编码

矢量量化(Vector Quantization，VQ)是 20 世纪 70 年代后期发展起来的一种基于块编

码规则的有损数据压缩方法，事实上，在 JPEG 和 MPEG-4 等多媒体压缩格式里都有 VQ 这一步。它的基本思想是：将若干个标量数据组构成一个矢量，然后在矢量空间给以整体量化，压缩了数据而不损失多少信息。矢量量化编码技术流程如图 3.19 所示。

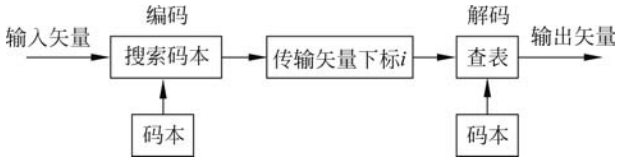


图 3.19　矢量量化编码与解码流程

在矢量量化过程中，对于给定的矢量 X_i，在码本中进行比较，得到一个与最为接近的矢量 Y_i，则码本矢量 Y_i 在码本中的矢量编码 i 即为 X_i 的量化值。这样，对于某个矢量 X_i，在编码时可以用一个编码 i 进行编码。矢量量化编码可以将多个复杂的采样点编码量化到一个码本矢量的编号，进而对这一矢量编号进行编码。

只要有码本和相应的编号，就可以快速解码，可以极大地压缩编码率。矢量量化的关键是设计一个能体现关键特征的码本。早期，VQ 运用的一个难点在于它要解决一个多维积分的问题。1980 年，Linde，Buzo 和 Gray(LBG)提出一种基于训练序列的 VQ 设计算法，对训练序列的运用绕开了多维积分的求解，使得世上又诞生了一种经典的 LBG-VQ 的算法。

3.5.2　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。

子带编码有以下几个突出的优点。

(1) 对不同的子带分配不同的比特数，可以很好地控制各个子带的量化电平数及重建信号时的量化误差方差值，进而获得更好的主观视听质量。

(2) 由于各个子带相互隔开，使各个子带的量化噪声也相互独立，互不影响，量化噪声被束缚在各自的子带内。这样，某些输入电平比较低的子带信号不会被其他子带的量化噪声所湮没。

(3) 子带划分的结果，使各个子带的采样频率大大降低。

因此，子带编码技术的关键是子带滤波器的设计，通常使用二维高、低通滤波器组成正交镜像滤波器组，采用复频移子带滤波措施，将高、中频段的信号移到低频段，再采用低通滤波器逐次进行过滤。

3.5.3　神经网络编码

人工神经网络 BP 运用于图像压缩编码中取得了较好的效果，一些人工神经网络模型可以直接提高数据压缩能力。

目前，在图像压缩中使用较多的是三层 BP 神经网络。其大致步骤为：先将图像分成 n 个小块，对应于输入的 n 个神经元，而压缩后的数据对应于隐含 m 个神经元；再用 BP 训练

算法调整网络权重，使重建图像尽可能相似于原图像，训练之后的 BP 神经网络便可直接用于数据压缩。人工神经网络 BP 用于图像数据压缩时，网络拓扑结构变换和算法修正对网络训练时间及重建图像质量都会有影响，因此，选择合适的网络结构和快速网络训练算法，可明显加速网络收敛，且网络易避开学习误差的局部极小点，可取得高压缩比和好的重建图像质量。

另外，除了神经网络直接用于图像压缩编码外，还可以将其与传统图像编码相结合，即将神经网络间接应用于图像编码。因为神经网络有较强的容错能力，因而有助于压缩有噪图像数据和恢复压缩后信息不全的图像。最后，由于神经网络有大规模并行处理的能力，也为神经网络图像编码的实时实现创造了条件。

3.5.4　模型编码

前面提到的预测编码和变换编码都是基于信息论的图像编码方法，它把图像信号看作随机信号，通过去除冗余度来压缩数据。模型编码和它们不同，是利用计算机视觉和计算图形学的知识，对图像信号进行分析与合成，涉及对图像结构和具体内容进行处理，利用与图像具体内容有密切关系的信息来压缩数据。编码的关键是对图像信号建立某种模型，并根据模型确定图像中景物的参数，如运动参数、形状参数等。解码时则根据参数和已知模型，运用图像合成技术重建图像。图像模型决定着图像编码方式，可以利用边缘、轮廓、区域等图像二维结构进行编码；可以利用编码对象的三维运动情形和三维形状估计数据进行编码；也可以利用涉及对象组织结构的三维物体模型进行编码。例如，编码器分析一个有头部运动的场景，可把头部作为一个三维目标建模，解码器中保留头部的三维模型，只要传输“移动”模型所需要的动态参数，并补偿模型场景和实际场景间的误差信号即可。因为只是针对对象的特征参数编码，而不是原始图像，因此，有可能实现比较大的压缩比，但是实时分析和综合一个视频场景的计算复杂度较高。目前，图像和视频编码标准都没有真正应用模型编码技术。

小结

多媒体系统要处理、传输、存储多媒体信息，由于这些媒体信息的数据量非常大，所以多媒体数据的高效压缩技术就成为多媒体系统的关键技术，也是目前多媒体应用中最为成熟的技术和标准化最完美的技术之一。本章介绍了多媒体压缩算法，如哈夫曼编码、算术编码、标准量化、DCT 和运动估计/运动补偿等是后续章节的基础，需要熟练掌握以上技术，才能更好地理解和应用音频、图像和视频等编码标准。

习题

1. 简述数据压缩的工作原理。
2. 请对字符串“555555777773332222111111”进行游程编码。
3. 设有 4 个字符 d，i，a，n，出现的频度为 7，5，2，4，请对此字符串用哈夫曼编码。
4. 运动图像编码中，帧内编码和帧间编码有什么区别？