

数字图像与视频处理

卢官明 唐贵进 崔子冠 编著

机械工业出版社
CHINA MACHINE PRESS



“十三五”江苏省高等学校重点教材 (编号: 2017-2-029)
高等院校通信与信息专业规划教材

数字图像 与视频处理

DIGITAL IMAGE AND VIDEO PROCESSING



卢官明 唐贵进 崔子冠 编著

机械工业出版社
CHINA MACHINE PRESS

教学资源下载网站
<http://www.cmpedu.com>

第5章 数字图像与视频压缩 编码原理

熊健

南邮通信学院



第5章 数字图像与视频压缩编码原理

- 5.1 数字图像与视频压缩编码概述
- 5.2 熵编码
- 5.3 预测编码
- 5.4 变换编码
- 5.5 MATLAB编程实例



5.1 数字图像与视频压缩编码概述

- 5.1.1 数字图像与视频压缩的必要性和可能性
- 5.1.2 数字图像与视频压缩编码的主要方法及其分类



5.1.1 数字图像与视频压缩的必要性和可能性

- 数据压缩的理论基础是信息论。从信息论的角度来看，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），也就是用一种更接近信息本质的描述来代替原有冗余的描述。
- 在一般的图像和视频数据中，主要存在以下几种形式的冗余。

5.1.1 数字图像与视频压缩的必要性和可能性

- **空间冗余：**也称为空域冗余，是一种与像素间相关性直接联系的数据冗余。

例：图像中包含许多规则物体，它们的亮度、饱和度及颜色可能都一样，因此，图像在空间上具有很强的相关性。例如 Lenna 图像的脸部和肩部。



5.1.1 数字图像与视频压缩的必要性和可能性

- **时间冗余：**也称为时域冗余，它是针对视频序列图像而言的。

视频序列每秒有25 ~ 30帧图像，相邻帧之间的时间间隔很小；同时实际生活中的运动物体具有运动一致性，使得视频序列图像之间有很强的相关性。



5.1.1 数字图像与视频压缩的必要性和可能性

■ 统计冗余

- **信源熵**：如果将信源所有可能事件的信息量进行平均，就得到了信源熵(entropy)。熵就是平均信息量。

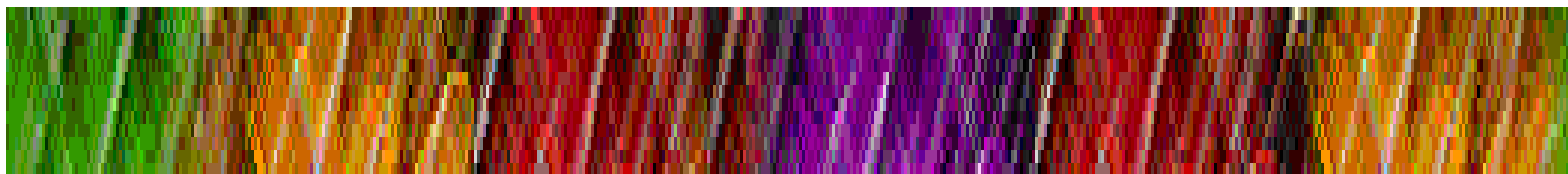
$$H(X) = E\{I(x_j)\} = \sum_{j=1}^n P(x_j) \cdot I(x_j) = -\sum_{j=1}^n P(x_j) \cdot \log_2 P(x_j)$$

- 当 x_j 等概率时， $H(X)$ 最大。
- 当 x_j 非等概率时， $H(X)$ 不是最大，就存在冗余。

采用可变长编码技术，对出现概率大的符号用短码字表示，对出现概率小的符号用长码字表示，则可去除符号冗余，从而节约码字，这就是**熵编码**的思想。

5.1.1 数字图像与视频压缩的必要性和可能性

- **结构冗余：**在有些图像的部分区域内有着很相似的纹理结构，或是图像的各个部分之间存在着某种关系，例如自相似性等，这些都是结构冗余的表现。



分形图像编码的基本思想就是利用了结构的自相似性。



5.1.1 数字图像与视频压缩的必要性和可能性

- **知识冗余：**在某些特定的应用场合，编码对象中包含的信息与某些先验的基本知识有关。例如：人脸的图像有同样的结构：嘴的上方有鼻子，鼻子上方有眼睛，鼻子在中线上……

可以利用这些先验知识为编码对象建立模型。通过提取模型参数，对**参数**进行编码而不是对图像像素值直接进行编码，可以达到非常高的压缩比。这是**模型基编码**（或称**知识基编码**、**语义基编码**）的基本思想。

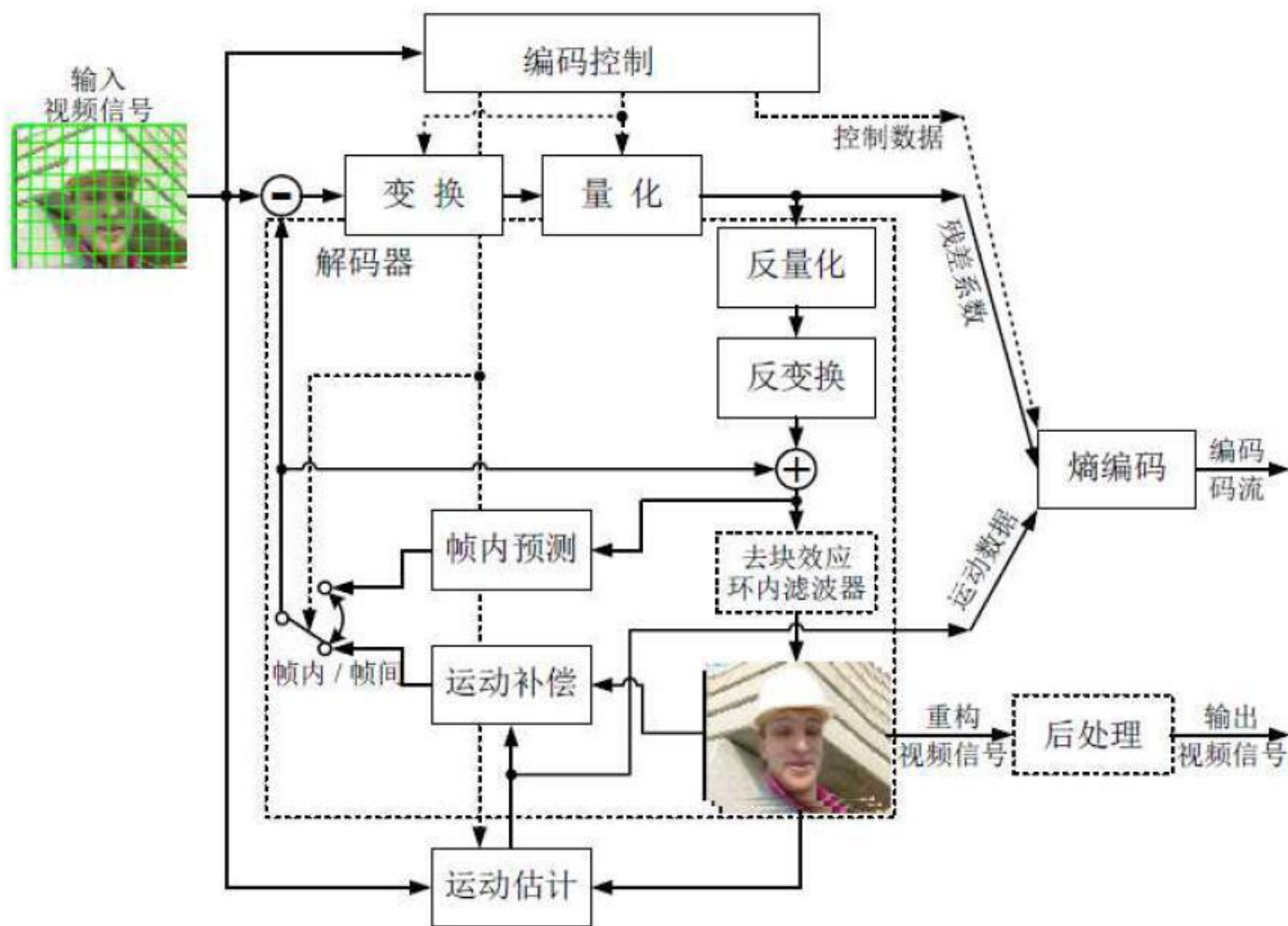


5.1.1 数字图像与视频压缩的必要性和可能性

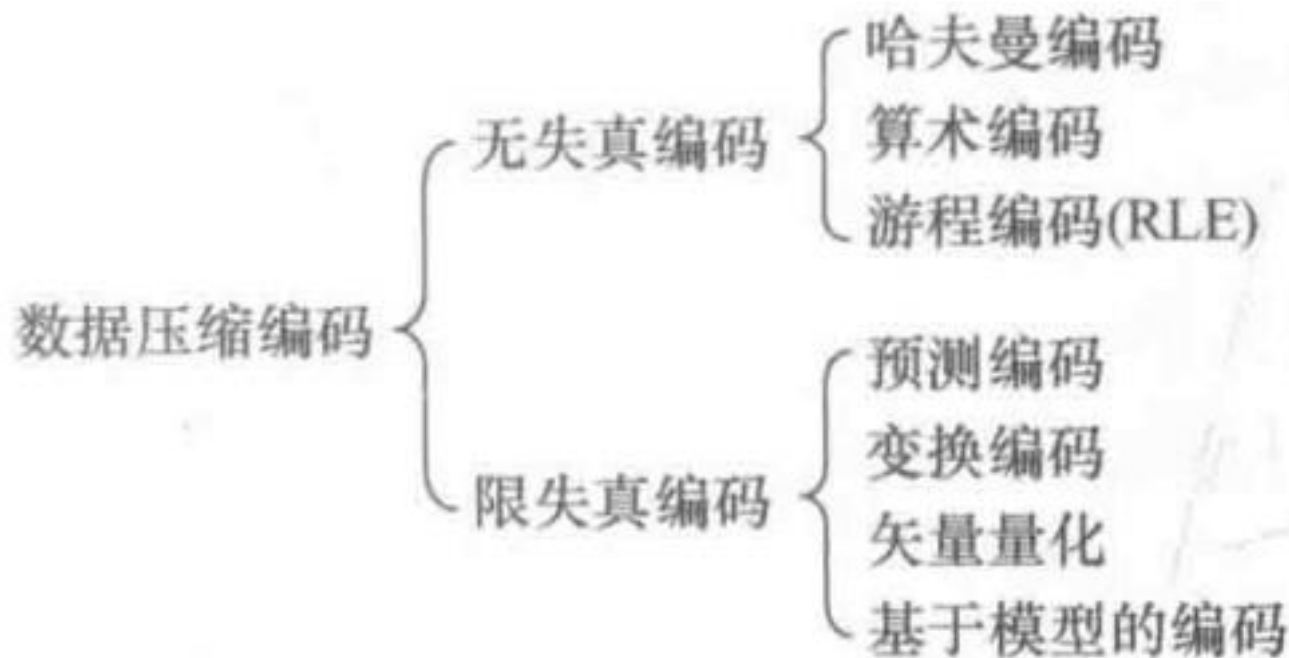
■ 人眼的视觉冗余

视觉冗余度是相对于人眼的视觉特性而言的。压缩视觉冗余的核心思想是去掉那些相对人眼而言是看不到的或可有可无的图像数据。对视觉冗余的压缩通常反映在各种具体的压缩编码过程中。

5.1.2 数字图像与视频压缩主要方法



5.1.2 数字图像与视频压缩主要方法





5.1.2 数字图像与视频压缩编码的主要方法及其分类

■ 熵编码

熵编码又称无损编码、信息保持编码、无失真编码。

熵编码是纯粹基于信号统计特性的一种编码方法，它利用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，解码后还原的数据与压缩编码前的原始数据完全相同而不引入任何失真。

熵编码的压缩比较低，可达到的最高压缩比受到信源熵的理论限制，一般为2：1到5：1。

最常用的熵编码方法有哈夫曼(Huffman)编码、算术编码和游程编码(Run-Length Encoding, RLE)等。



5.1.2 数字图像与视频压缩编码的主要方法及其分类

■ 限失真编码

限失真编码也称有损编码、非信息保持编码、熵压缩编码。

限失真编码方法利用了人类视觉的感知特性，允许压缩过程中损失一部分信息，虽然在解码时不能完全恢复原始数据，但是如果把失真控制在视觉阈值以下或控制在可容忍的限度内，则不影响人们对图像的理解，却换来了高压缩比。在限失真编码中，允许的失真愈大，则可达到的压缩比愈高。

常见的限失真编码方法有：预测编码、变换编码、矢量量化、基于模型的编码等。



第5章 数字图像与视频压缩编码原理

- 5.1 数字图像与视频压缩编码概述
- **5.2 熵编码**
- 5.3 预测编码
- 5.4 变换编码
- 5.5 MATLAB编程实例



5.2 熵编码

信息熵 (Shannon) :

$$Entropy (S) = - \sum_{i=1}^n P_i \log P_i$$

单位: Bit/字符

1. 信源S的平均信息量;
2. 编码所有符号S平均所需要的位数。



5.2 熵编码

什么要压缩编码？

例：把100MB的原始数据，压缩成10MB的数据

那什么是熵编码？ 在信息熵的极限范围内进行编码。例如信息熵算出来是3bit/字符，用4bit/字符来编码，就是熵编码，用2bit/字符来编码，就不叫熵编码，这种情况下会失真了

信源熵是编码这个信源平均所需要的最小位数。所以，熵编码是无损压缩。



5.2 熵编码

熵编码有很多种：

- **游程编码(Run-Length Encoding, RLE)**
- **哈夫曼(Huffman)编码**
- **算术编码**
- **基于上下文的自适应可变长编码 (CAVLC)**
- **基于上下文的自适应二进制算术编码 (CABAC)**



5.2.1 游程编码

游程编码 (RLE) , 也称行程编码或游程(行程) 长度编码, 其基本思想是将具有相同数值 (例如, 像素的灰度值) 的、连续出现的信源符号构成的符号序列用其数值及串的长度表示。

以图像编码为例, 灰度值相同的相邻像素的连续长度 (像素数目) 称为连续的游程, 又称游程长度, 简称游程。



5.2.1 游程编码

下面以**二值图像**为例进行说明。二值图像是指图像中的像素值只有两种取值，即“0”和“1”，因而在图像中这些符号会连续地出现，我们通常将连“0”这一段称为“0”游程，而连“1”的一段则称为“1”游程，它们的长度分别表示为 $L(0)$ 和 $L(1)$ ，往往“0”游程与“1”游程会**交替出现**，即第一游程为“0”游程。第二游程为“1”游程。第三游程又为“0”游程。下面我们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.....，根据游程编码规则，可知其游程序列为21133121.....。



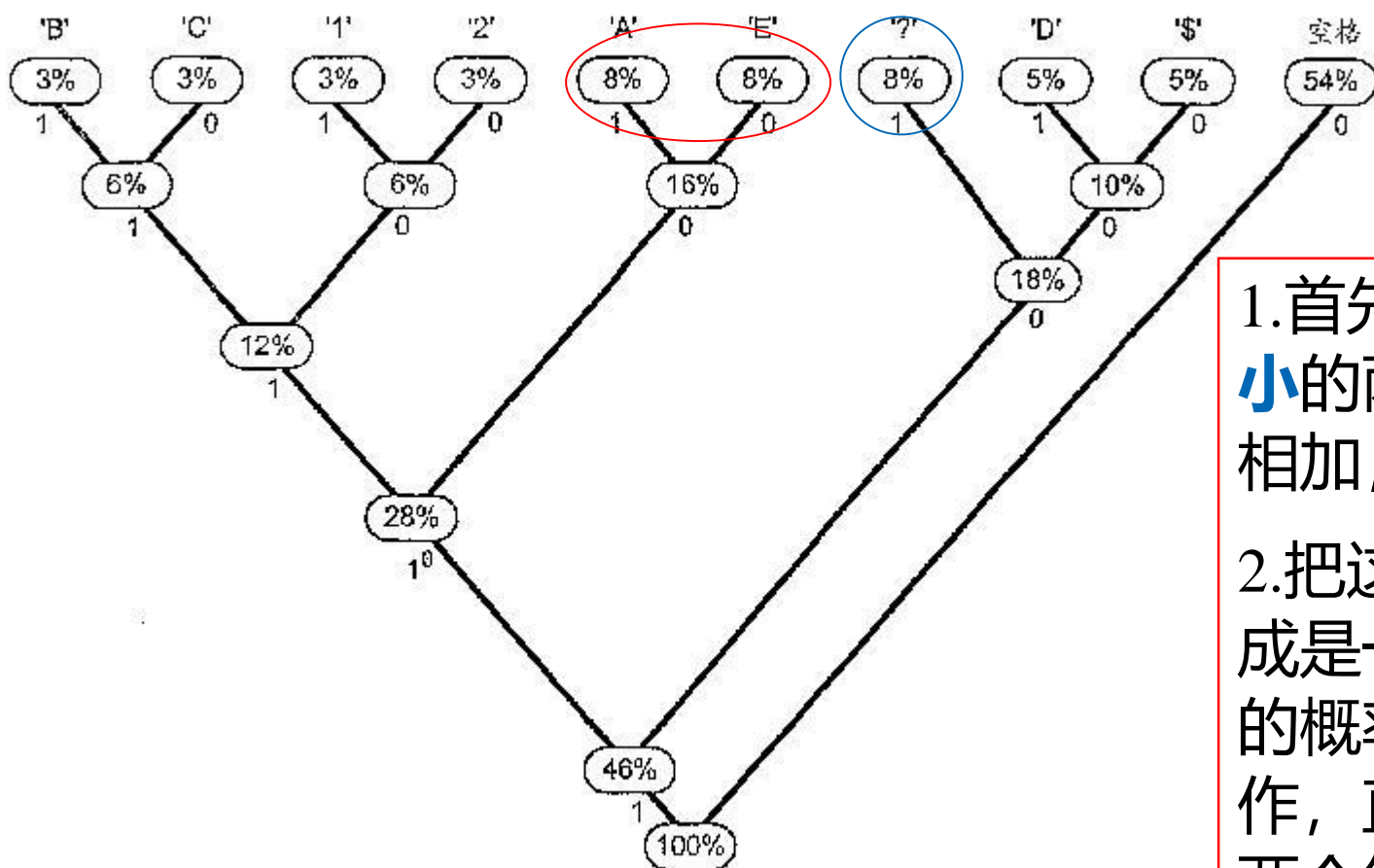
5.2.2 哈夫曼编码

哈夫曼(Huffman)于1952年提出一种编码方法，完全依据符号出现概率来构造异字头（前缀）的平均长度最短的码字，有时称之为最佳编码。

哈夫曼编码是一种可变长度编码(Variable Length Coding, VLC)，各符号与码字一一对应，是一种分组码。

5.2.2 哈夫曼编码

● Huffman编码过程 (1)





5.2.3 算术编码

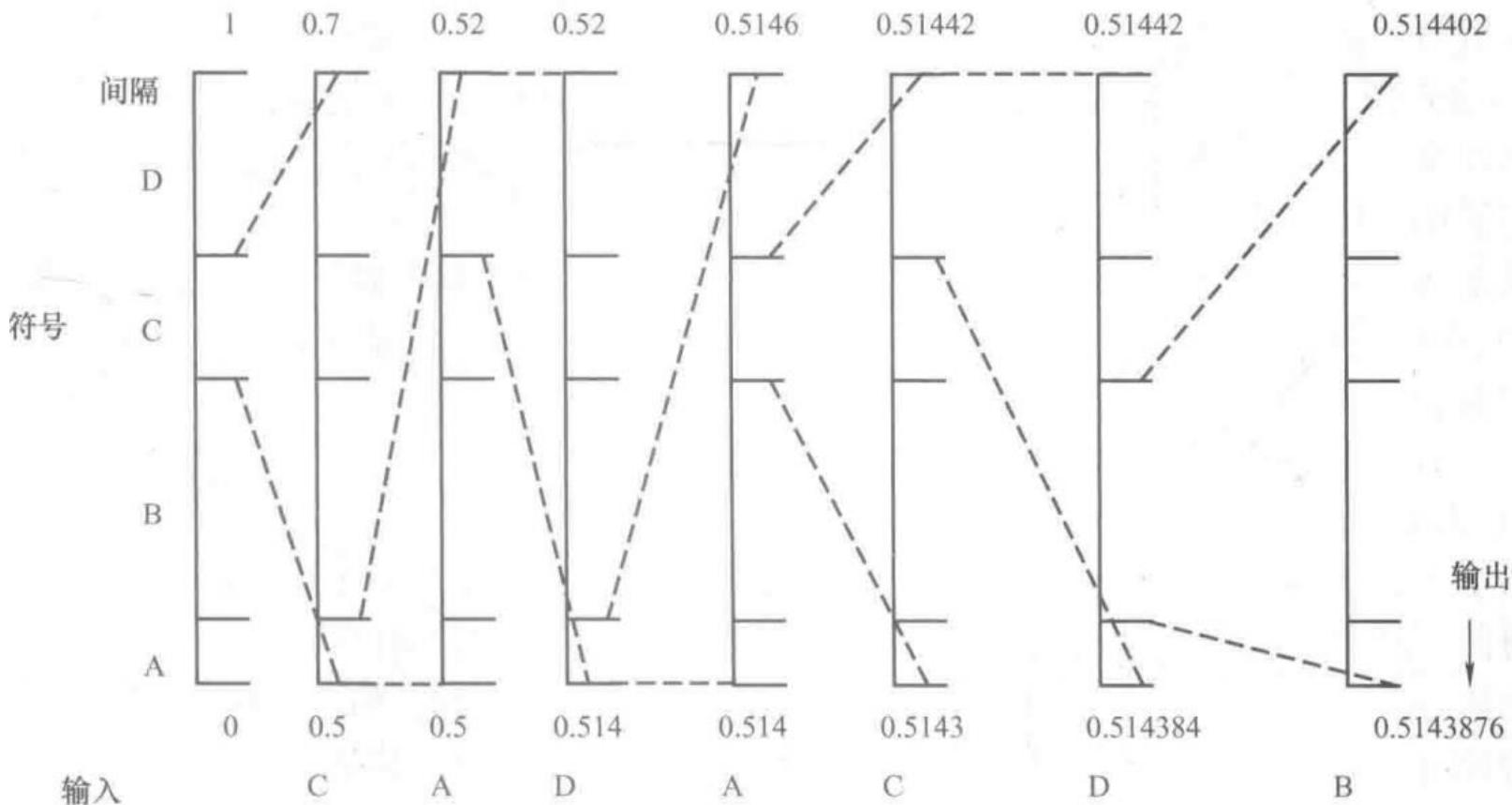
算术编码是一种非分组编码，它用一个浮点数值表示**整个信源符号序列**。算术编码将被编码的信源符号序列表示成**实数半开区间 $[0, 1)$** 中的一个数值间隔。这个间隔随着信源符号序列中每一个信源符号的加入逐步减小，每次减小的程度取决于当前加入的信源符号的先验概率。

5.2.3 算术编码

例：假设信源符号为{00, 01, 10, 11}, 这些符号的概率分别为{0.1, 0.4, 0.2, 0.3}。根据这些概率, 可把间隔 $[0, 1)$ 分成4个子间隔: $[0, 0.1)$, $[0.1, 0.5)$, $[0.5, 0.7)$ 和 $[0.7, 1)$ 。

表 5-1 信源符号、概率和初始编码区间

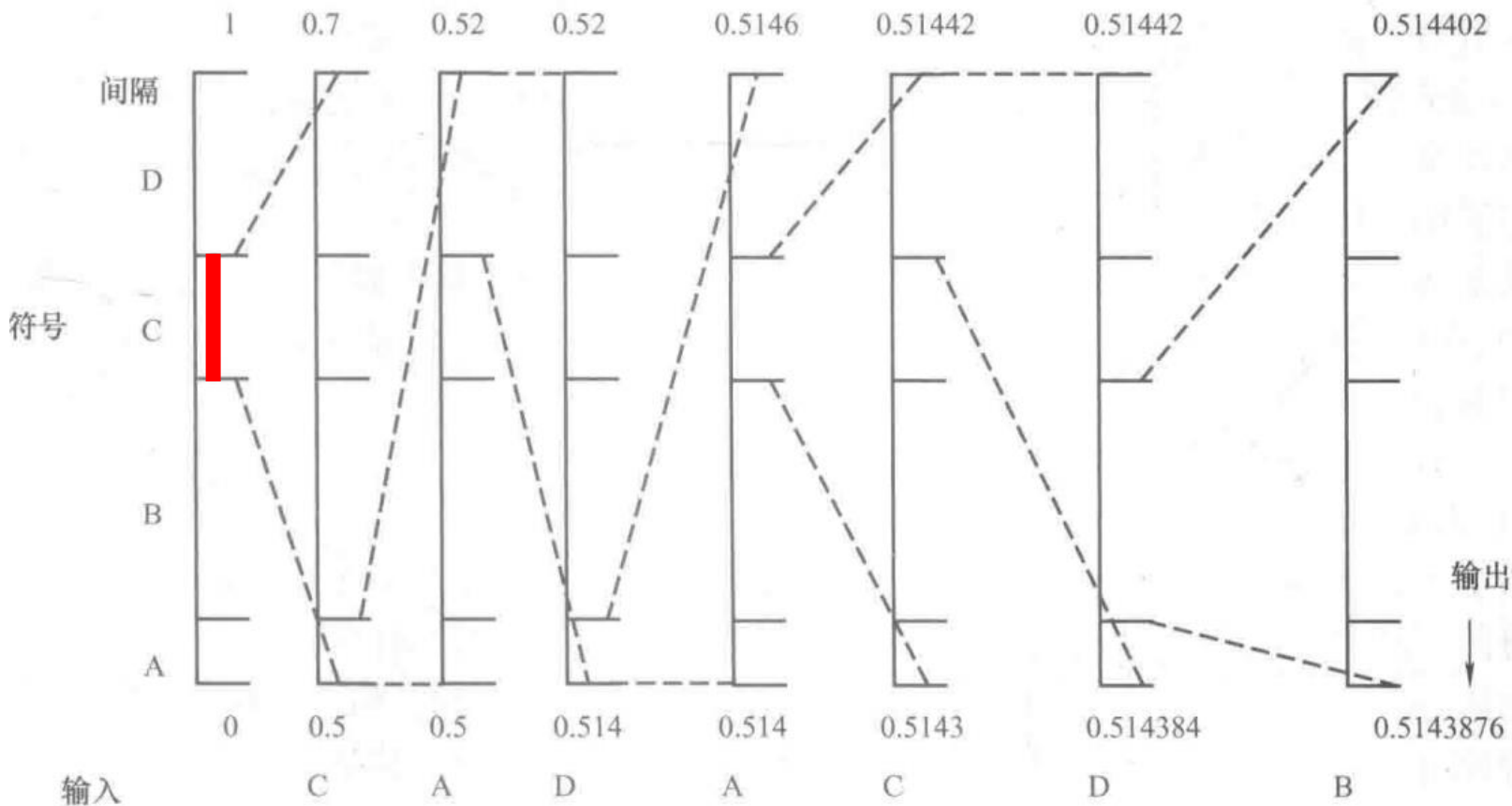
符 号	A	B	C	D
概率	0.1	0.4	0.2	0.3
初始编码子区间	$[0, 0.1)$	$[0.1, 0.5)$	$[0.5, 0.7)$	$[0.7, 1)$



- ① 初始化：设置当前区间的左端点值 $low = 0$ ，右端点值 $high = 1.0$ ，当前区间长度 $length = 1.0$ 。
- ② 对符号序列中每一个输入的信源符号进行编码，采用式(5-6) 的递推形式。

$$\begin{cases} low = low + length \times symbol_low \\ high = low + length \times symbol_high \end{cases} \quad (5-6)$$

式中，等号右边的 low 和 $length$ 分别为前面已编码符号序列所对应编码区间的左端点值和区间长度；等号左边的 low 和 $high$ 分别为输入待编码符号后所对应的“当前区间”的左端点值和右端点值。



“当前区间”的区间长度为

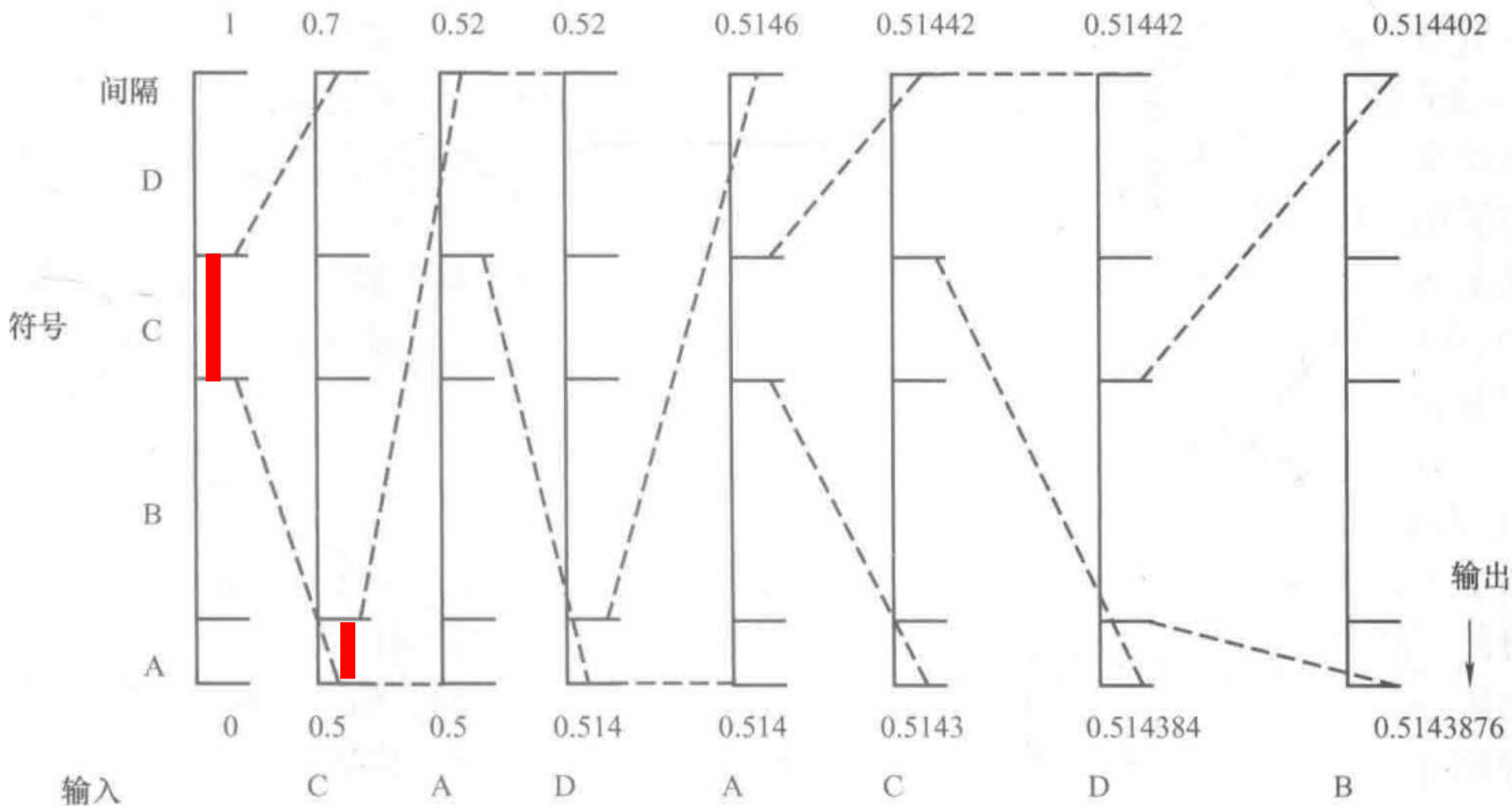
$$length = high - low \quad (5-7)$$

- 对输入的第1个信源符号C编码，有

$$\begin{cases} low = low + length \times symbol_low = 0 + 1 \times 0.5 = 0.5 \\ high = low + length \times symbol_high = 0 + 1 \times 0.7 = 0.7 \end{cases}$$

所以,输入第1个信源符号C后,编码区间从 $[0,1)$ 变成 $[0.5,0.7)$,”当前区间”的区间长度为

$$length = high - low = 0.7 - 0.5 = 0.2$$

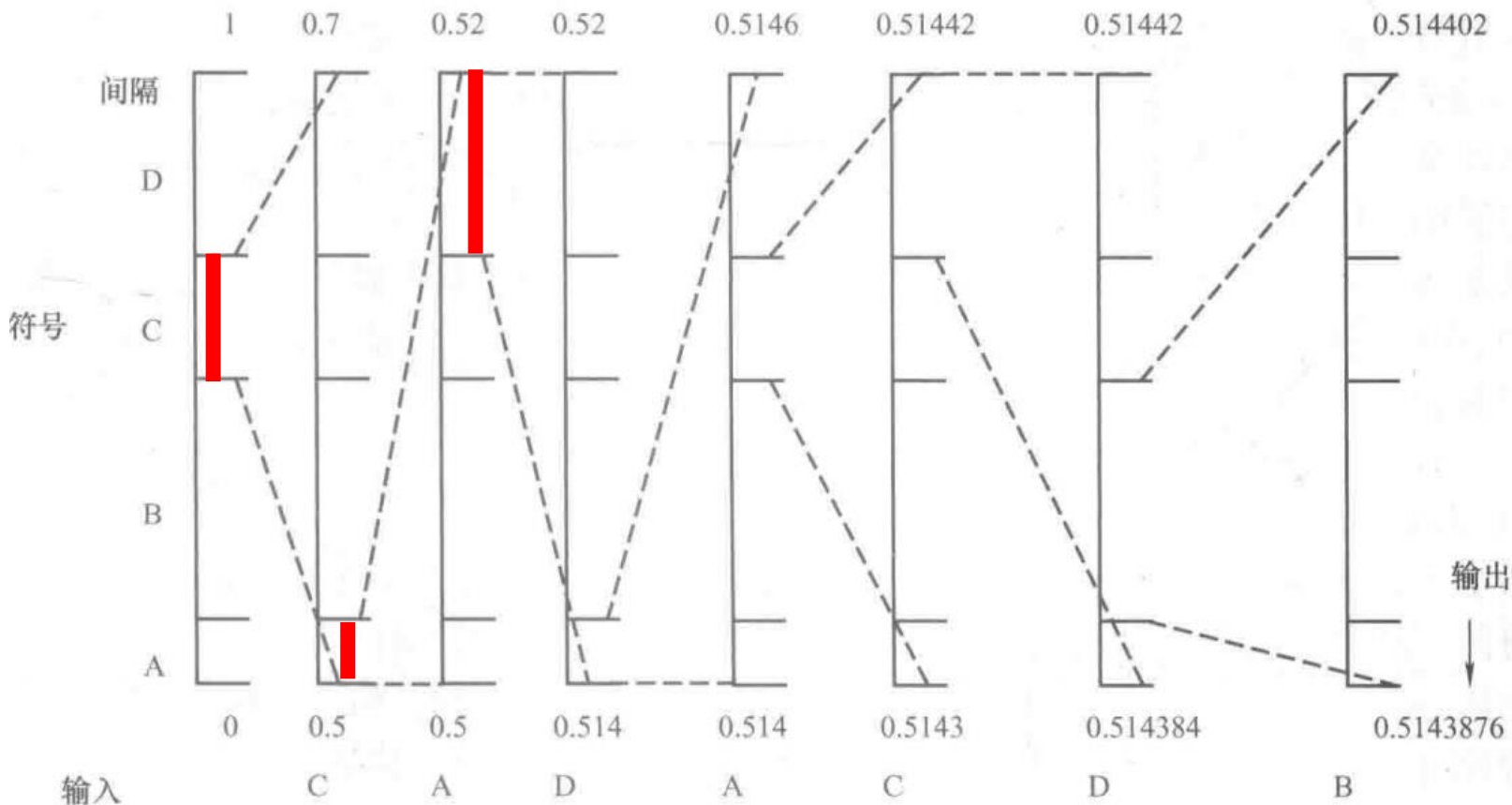


- 对输入的符号序列 CA 进行编码,有

$$\begin{cases} low = low + length \times symbol_low = 0.5 + 0.2 \times 0 = 0.5 \\ high = low + length \times symbol_high = 0.5 + 0.2 \times 0.1 = 0.52 \end{cases}$$

所以, 输入第 2 个信源符号 A 后, 编码区间从 $[0.5, 0.7)$ 变成 $[0.5, 0.52)$, “当前区间” 的区间长度为

$$length = high - low = 0.52 - 0.5 = 0.02$$

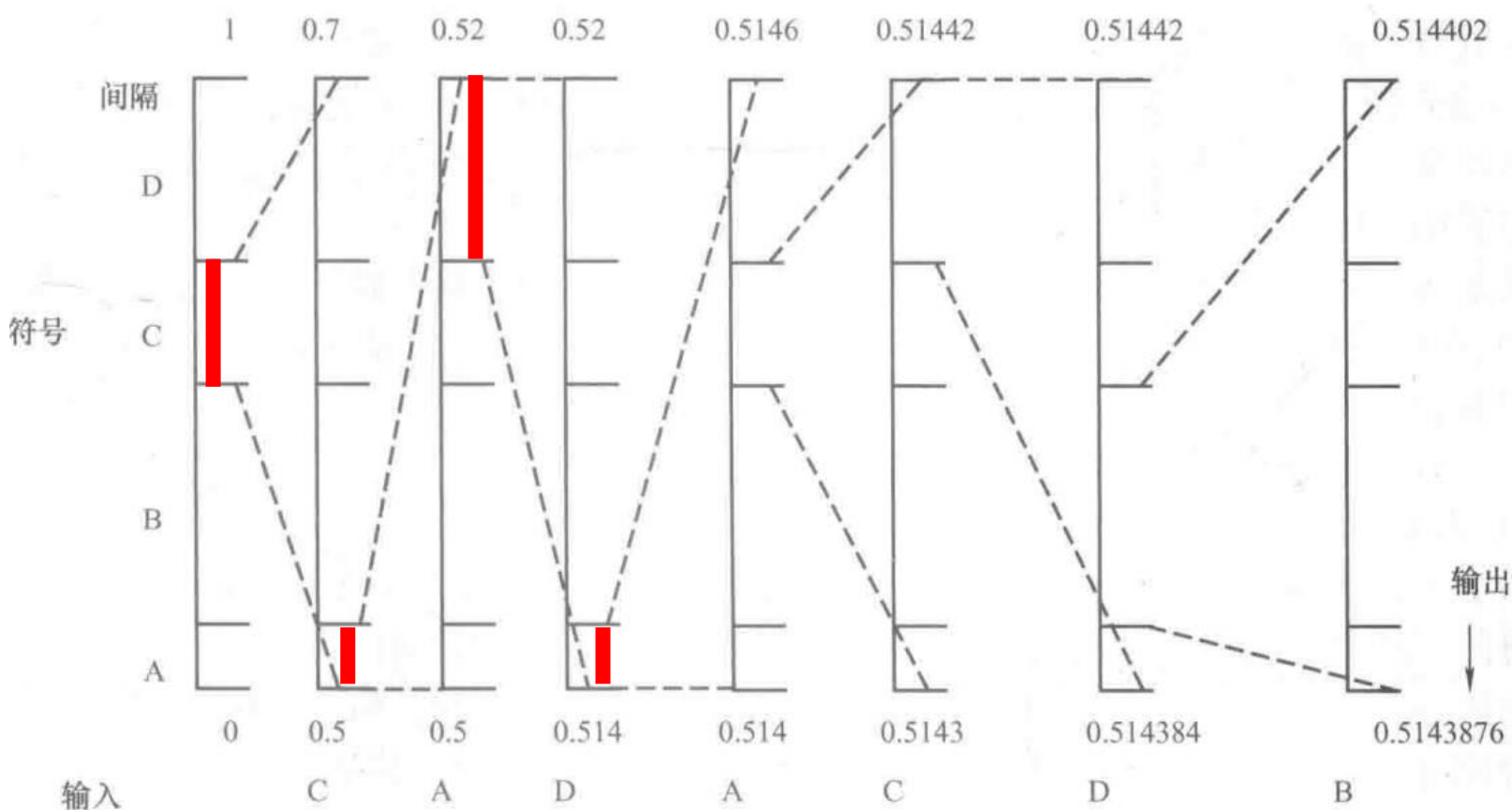


- 对输入的符号序列 CAD 进行编码，有

$$\begin{cases} low = low + length \times symbol_low = 0.5 + 0.02 \times 0.7 = 0.514 \\ high = low + length \times symbol_high = 0.5 + 0.02 \times 1 = 0.52 \end{cases}$$

所以，输入第 3 个信源符号 D 后，编码区间从 $[0.5, 0.52)$ 变成 $[0.514, 0.52)$ ，“当前区间”的区间长度为

$$length = high - low = 0.52 - 0.514 = 0.006$$

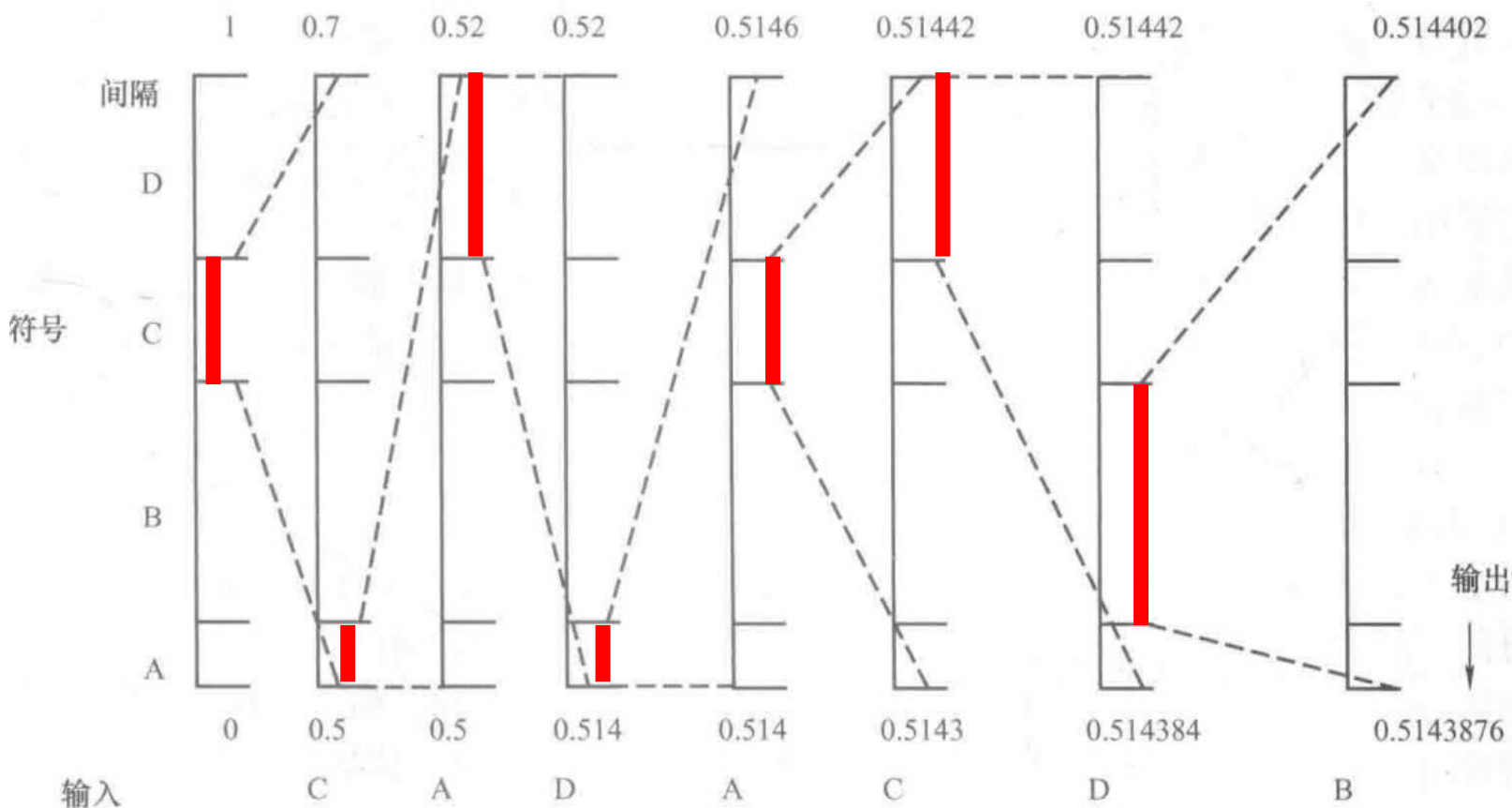


- 对输入的符号序列 CADA 进行编码，有

$$\begin{cases} low = low + length \times symbol_low = 0.514 + 0.006 \times 0 = 0.514 \\ high = low + length \times symbol_high = 0.514 + 0.006 \times 0.1 = 0.5146 \end{cases}$$

所以，输入第 4 个信源符号 A 后，编码区间从 $[0.514, 0.52)$ 变成 $[0.514, 0.5146)$ ，“当前区间”的区间长度为

$$length = high - low = 0.5146 - 0.514 = 0.0006$$



- 对输入的符号序列 CADACDB 进行编码，有

$$\begin{cases} low = low + length \times symbol_low = 0.514384 + 0.000036 \times 0.1 = 0.5143876 \\ high = low + length \times symbol_high = 0.514384 + 0.000036 \times 0.5 = 0.514402 \end{cases}$$



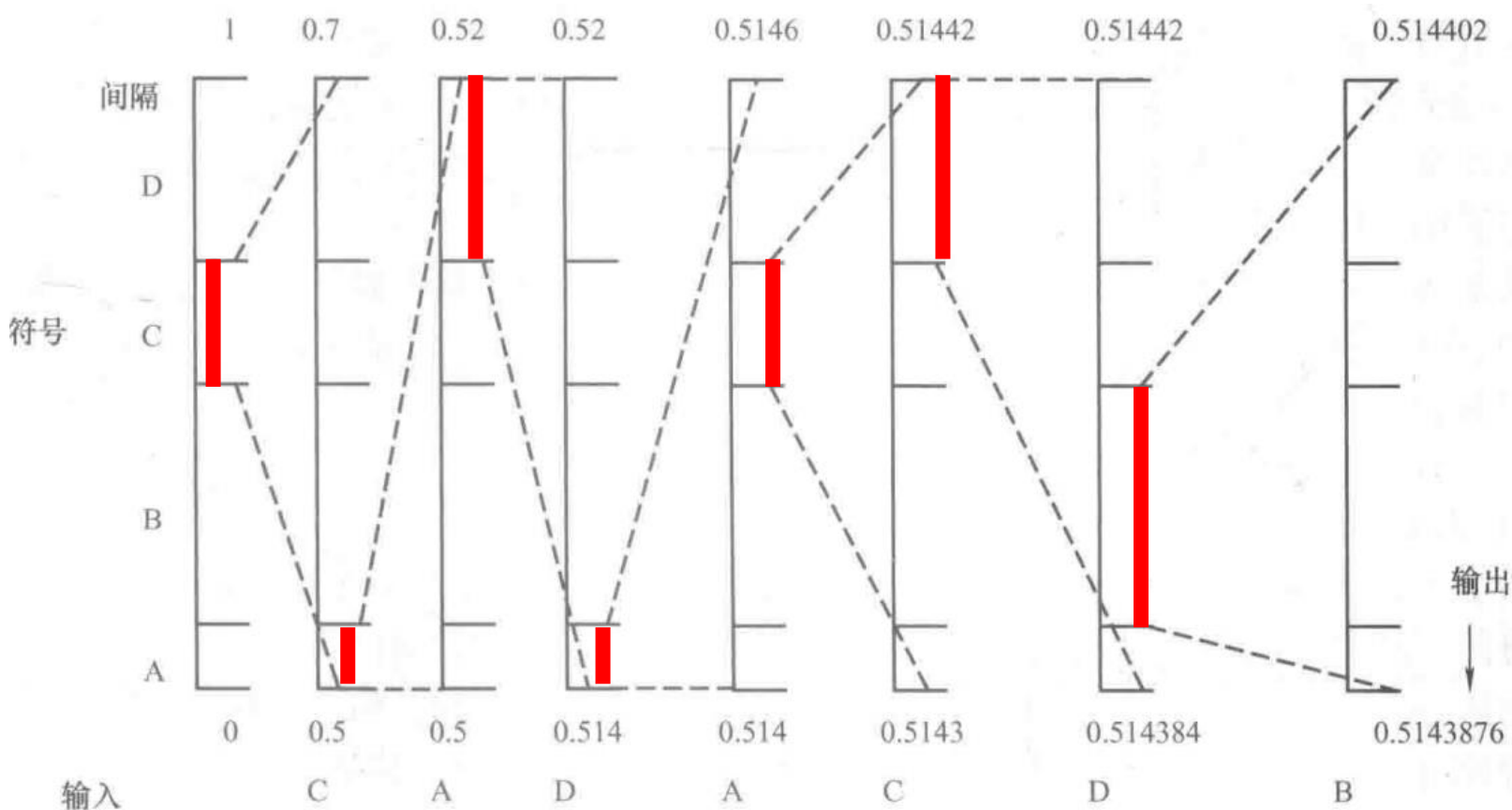
5.2.3 算术编码

$$\begin{cases} low' = low + length \times symbol_low \\ high' = low + length \times symbol_high \end{cases}$$

$$length' = high' - low'$$

$$length' = length \times (symbol_high - symbol_low)$$

$$length' = length \times (symbol_length)$$



$$\frac{0.5143876 - 0}{1} = 0.5143876 \in [0.5, 0.7) \Rightarrow C$$

$$\frac{0.5143876 - 0.5}{0.2} = 0.071938 \in [0, 0.1) \Rightarrow A$$

$$\frac{0.071938 - 0}{0.1} = 0.71938 \in [0.7, 1.0) \Rightarrow D$$

$$\frac{0.71938 - 0.7}{0.3} = 0.0646 \in [0, 0.1) \Rightarrow A$$

$$\frac{0.0646 - 0}{0.1} = 0.646 \in [0.5, 0.7) \Rightarrow C$$

$$\frac{0.646 - 0.5}{0.2} = 0.73 \in [0.7, 1.0) \Rightarrow D$$

$$\frac{0.73 - 0.7}{0.3} = 0.1 \in [0.1, 0.5) \Rightarrow B$$

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



5.2.3 算术编码

算术编码的主要特点有：

- (1) 当信源符号的出现概率比较接近时，算术编码的效率比哈夫曼编码高。
- (2) 算术编码的实现比哈夫曼编码复杂。

算术编码是一种相对比较新的编码，它在许多方面比哈夫曼编码优越：算术编码按照分数比特逼近熵，而哈夫曼编码是按照整数比特逼近熵的；算术编码可以有效地从模型中分离出来，而哈夫曼编码是与统计模型强相关的。



5.2.3 算术编码

■ 为什么说算术编码压缩率更接近最优编码？

从熵编码角度：概率越小的字符，用更多的bit去表示。

反映到概率区间上就是，概率小的字符所对应的区间也小，因此这个区间的上下边际值的**差值越小（区间的上下界很接近）**，为了唯一确定当前这个区间，则需要更多的数字去表示它

例如：大区间0.2到0.3，需要0.2来确定，一位（十进制）足以表示；但如果是小的区间0.11112到0.11113，则需要0.11112才能确定这个区间，编码时就需要5位才能将这个字符确定。

5.2.3 算术编码

为什么算术编码压缩率更接近最优编码？



进制转换

0.2(十进制) = 0.0011001100110011(二进制)



进制转换

0.24(十进制) = 0.0011110101110000(二进制)



进制转换

0.3(十进制) = 0.0100110011001100(二进制)



进制转换

0.6(十进制) = 0.1001100110011001(二进制)

超出计算精度，结果保留十六位小数

0.6

十进制



转换为

二进制



转换



第5章 数字图像与视频压缩编码原理

- 5.1 数字图像与视频压缩编码概述
- 5.2 熵编码
- **5.3 预测编码**
- 5.4 变换编码
- 5.5 MATLAB编程实例



5.3 预测编码

预测编码：是根据离散信号之间存在着一定关联性的特点，利用前面一个或多个信号预测对下一个信号进行预测，然后对实际值和预测值的差（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。



5.3 预测编码

就图像而言，预测编码的基本原理就是利用图像数据的相关性，利用已传输的像素值对当前需要传输的像素值进行预测，然后对当前像素的实际值与预测值的**差值**（即**预测误差**）进行编码传输，而不是对当前像素值本身进行编码传输，以去除图像数据中的**空间**相关冗余或**时间**相关冗余。



5.3.1 预测编码基本原理

- **预测编码**：根据某一**模型**，利用信号以往的样本值对新样本值进行预测，对**预测误差**进行编码。
- 对于**相关性较强的信号**，如果建立**合适的模型**，**预测误差**的幅值将远远小于原始信号，从而可以用较少的**量化级**对其误差信号进行**量化**，得到较大的数据压缩效果。



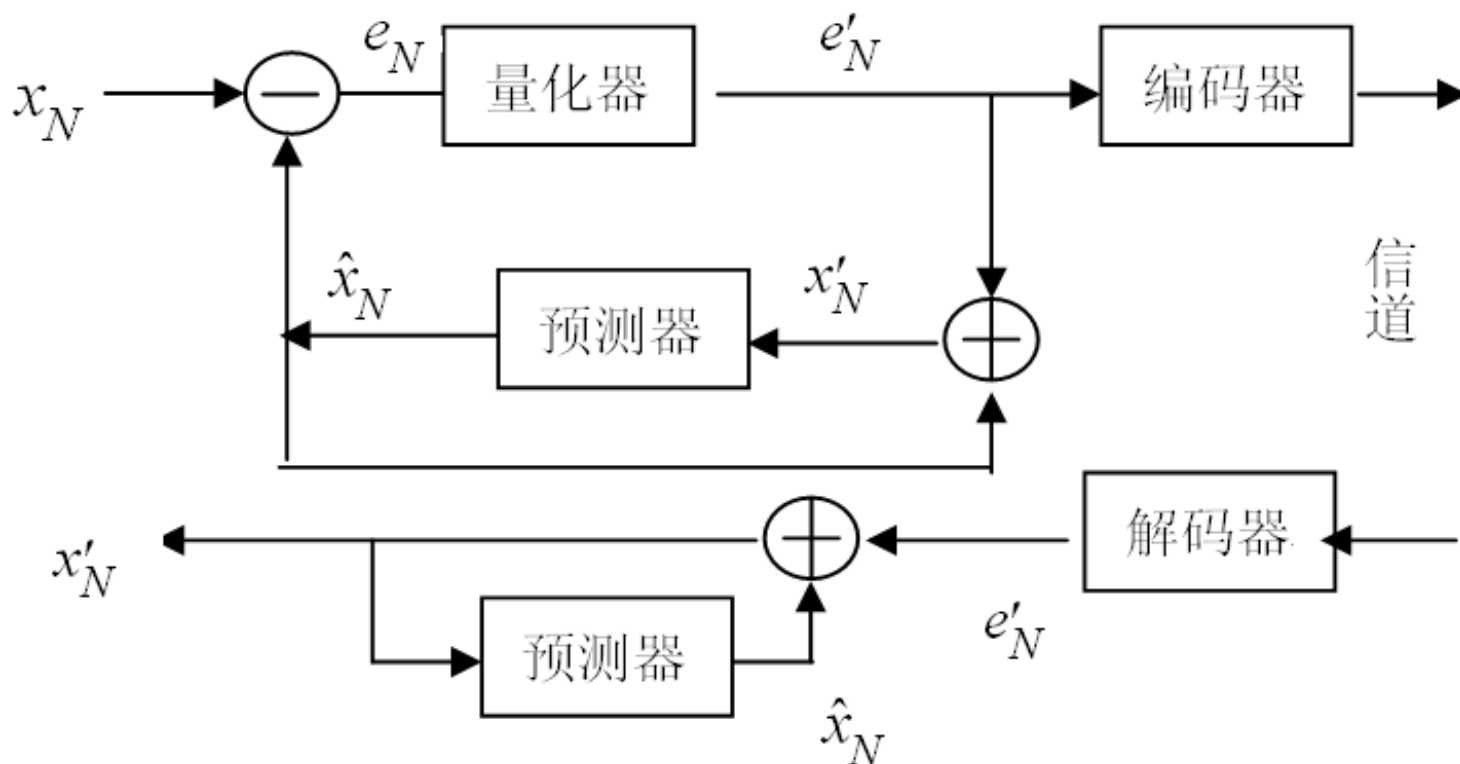
5.3.1 预测编码基本原理

- 对于**静止图像**，由于相邻像素具有很强的相关性，这样当前像素的灰度（颜色）值可用前面已经出现的像素值进行预测，得到一个预测值，对实际值与预测值的差值进行编码，
- 对于**视频信号**，图像帧间的相关性具有很强的相关性，通过帧间预测，对残差图像编码。
- 预测编码是当今主流技术并且还会流行于未来(BPG,H.264,HEVC,VVC,AVS2,AVS3)。

5.3.2 帧内预测编码

■ 1. DPCM系统的基本原理

DPCM(Differential Pulse Code Modulation, 差分脉冲编码调制)



5.3.2 帧内预测编码

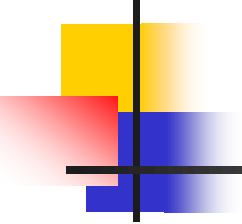
■ 2. 预测模型

设 t_N 时刻之前的样本值 x_1, x_2, \dots, x_{N-1} 与预测值之间的关系呈现某种函数形式

∞ 线性预测编码器

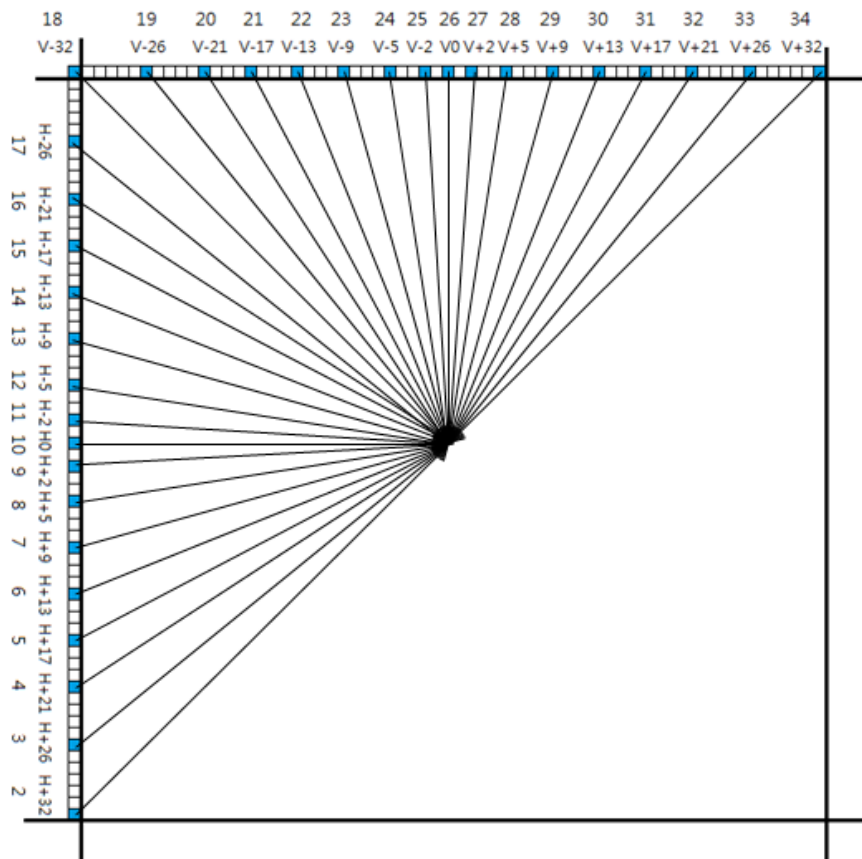
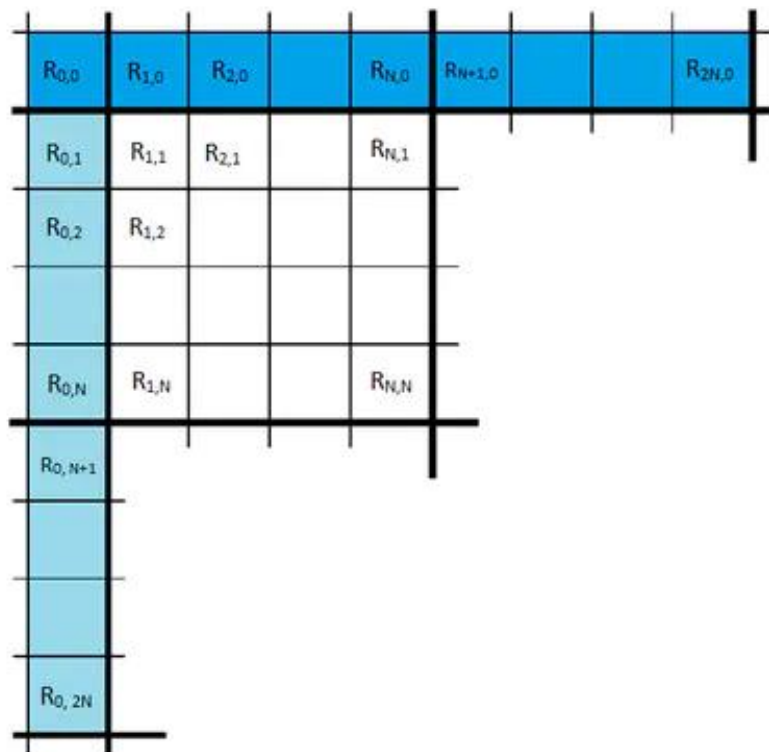
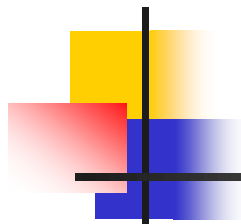
$$\hat{x}_N = \sum_{i=1}^{N-1} a_i x_i$$

∞ 非线性预测编码器



在图像数据压缩中，常用如下几种**线性预测方案**：

- * **前值预测**，即 $\hat{x}_N = x_{N-1}$
- * **一维预测**，即采用同一扫描行中前面已知的若干个样值来预测。
- * **二维预测**，即不但用同一扫描行中的前面几个样值，而且还要用以前几行扫描行中样值来预测。



HEVC帧内预测



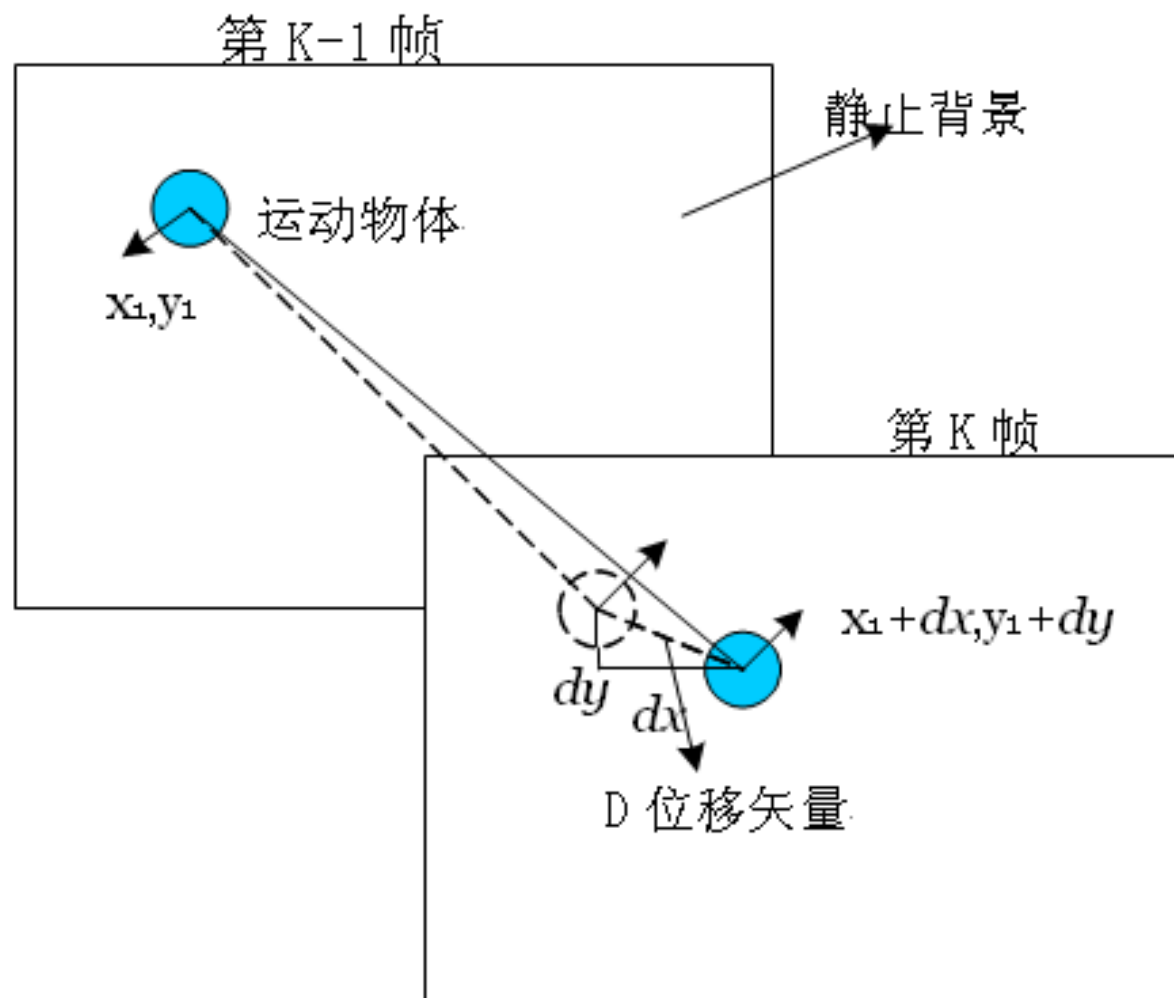
5.3.3 帧间预测编码

序列图像在时间上的冗余情况可分为如下几种：

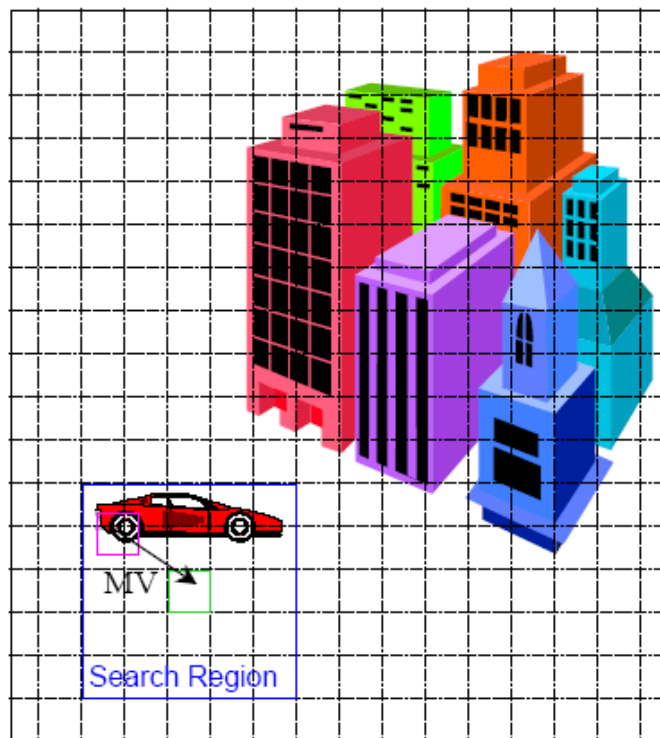
- ✧ **对于静止不动的场景**，当前帧和前一帧的图像内容是完全相同的。
- ✧ **对于运动的物体**，只要知道其运动规律，就可以从前一帧图像推算出它在当前帧中的位置。
- ✧ **摄像机**对着场景的横向移动、焦距变化等操作会引起整个图像的平移、放大或缩小。对于这种情况，只要**摄像机的运动**规律和镜头改变的参数已知，图像随时间所产生的变化也是可以推算出来的。

5.3.3 帧间预测编码

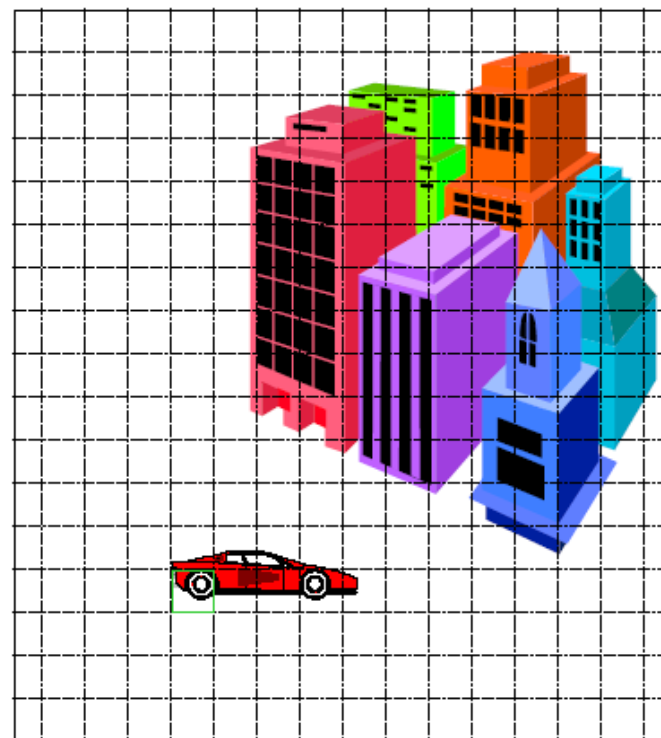
■ 运动补偿预测



5.3.3 帧间预测编码



Frame $t-1$
(Reference Frame)



Frame t
(Predicted frame)

◆ 对当前子块进行运动估计，就是找在上一帧图像中哪一个子块和当前子块最相似，估计它的位移矢量。

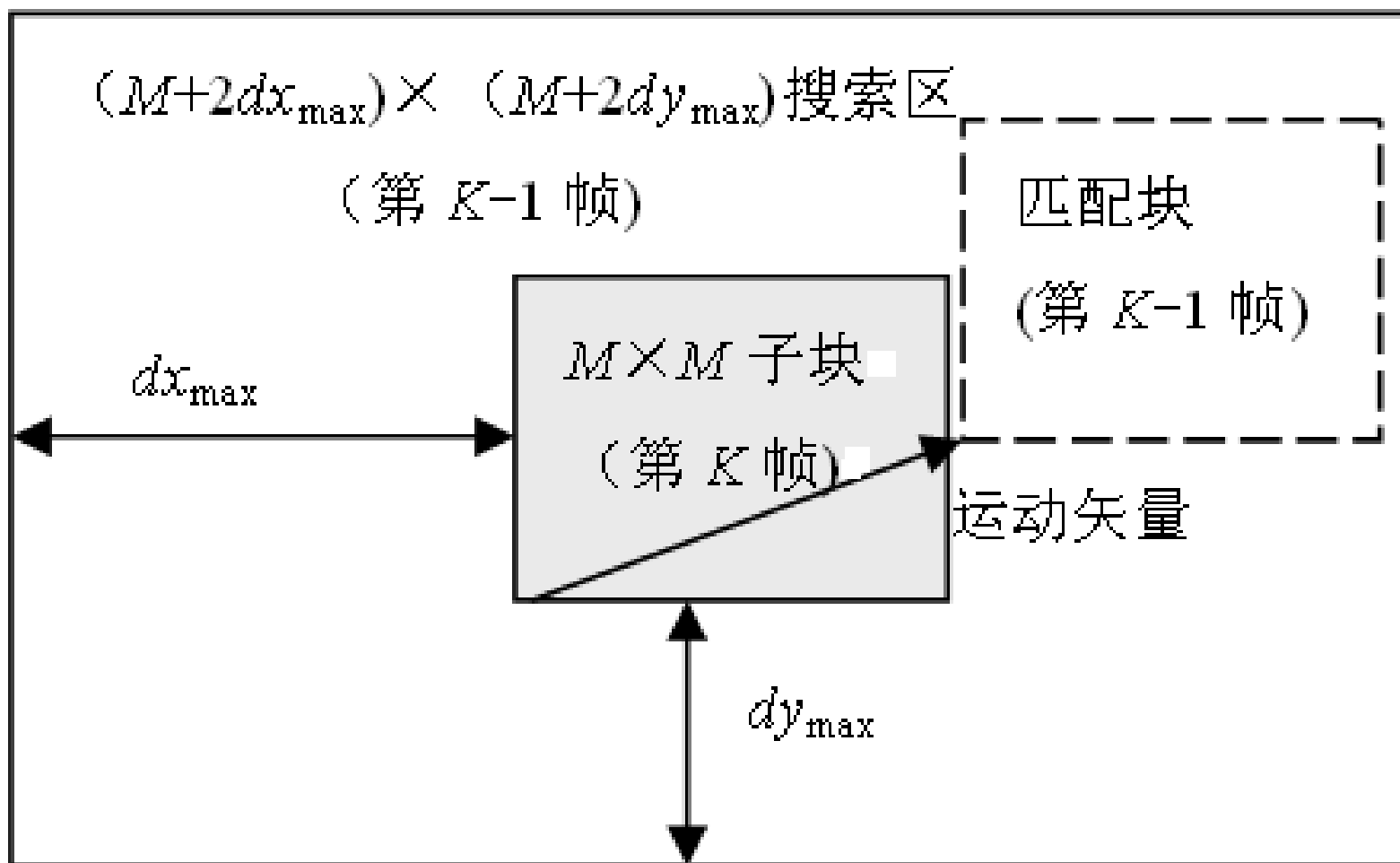
5.3.3 帧间预测编码

■ 运动估计方法：

- **像素递归法**：根据像素间亮度的**变化和梯度**，通过递归修正的方法来估计每个像素的运动矢量。接收端在与发送端同样的条件下，用与发送端相同的方法进行运动估值。像素递归法估计精度高，可以满足运动补偿帧内插的要求。但接收端较复杂，不利于一发多收（如数字电视广播等）的应用。
- **块匹配算法**：块匹配算法对当前帧图像的每一子块，在前一帧（第 $K-1$ 帧）的一定范围内搜索最优匹配，并认为本图像子块就是从前一帧最优匹配块位置处平移过来的。块匹配算法虽然作了一定假设（假设位于同一图像子块内的所有像素都作相同的运动，且只作平移运动），但满足了计算复杂度和实时实现的要求。

5.3.3 帧间预测编码

➤ 块匹配算法





块匹配算法(BMA)

➤ 方块大小的选取

块大时，一个方块可能包含多个作不同运动的物体，块内各像素作相同平移运动的假设难以成立，影响估计精度。

若块太小，则估计精度容易受噪声干扰的影响，不够可靠；而且传送运动矢量所需的附加比特数过多，不利于数据压缩。

一般都用 16×16 像素的块作为匹配单元。



块匹配算法(BMA)

➤ 最优匹配准则

- 绝对差均值 (MAD, Mean Absolute Difference) 最小准则

$$MAD(i, j) = \frac{1}{MM} \sum_{m=1}^M \sum_{n=1}^M |S_K(m, n) - S_{K-1}(m+i, n+j)|$$

- 均方误差 (MSE, Mean Squared Error) 最小准则

$$MSE(i, j) = \frac{1}{MM} \sum_{m=1}^M \sum_{n=1}^M [S_K(m, n) - S_{K-1}(m+i, n+j)]^2$$

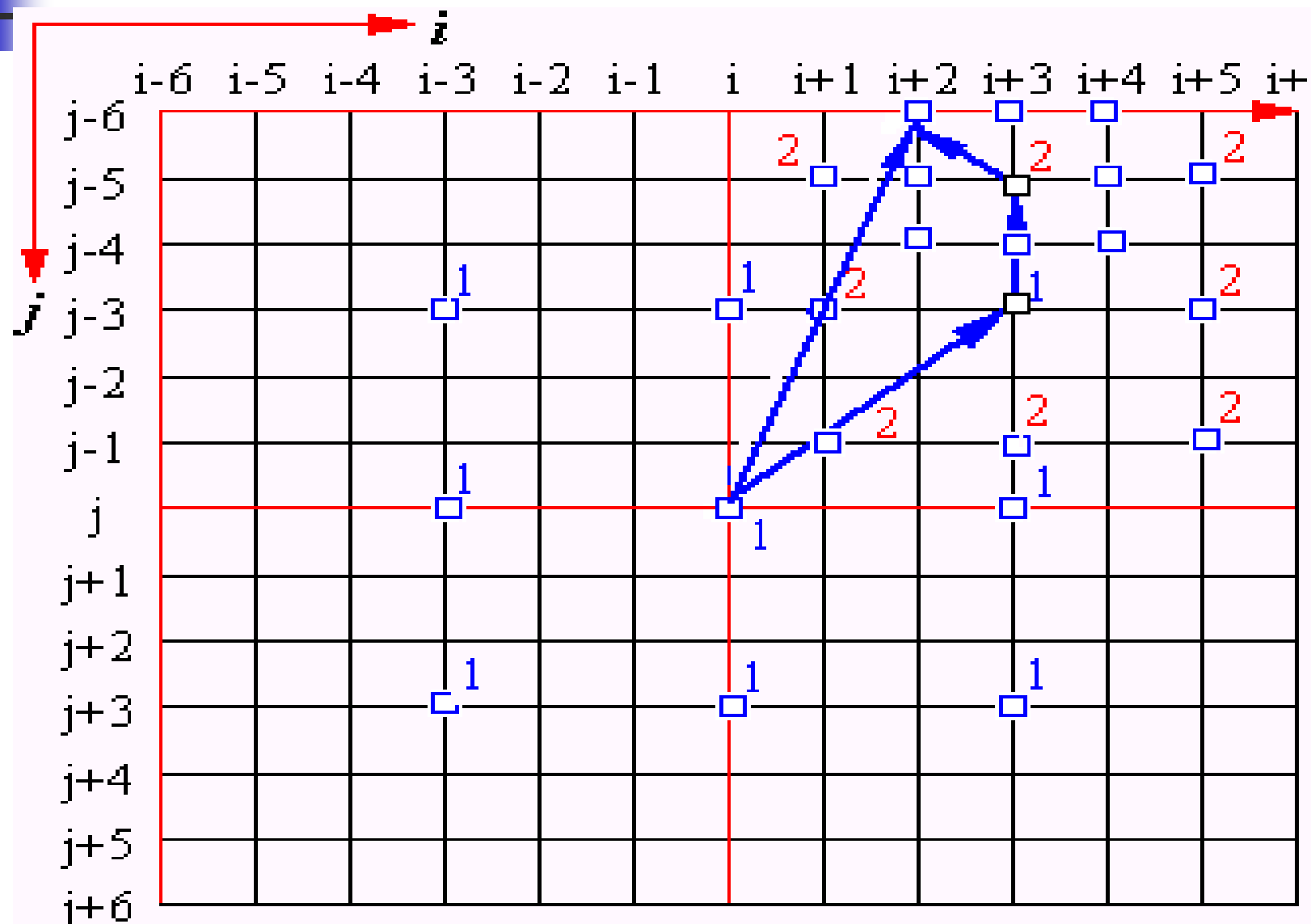
- 归一化互相关函数最大准则

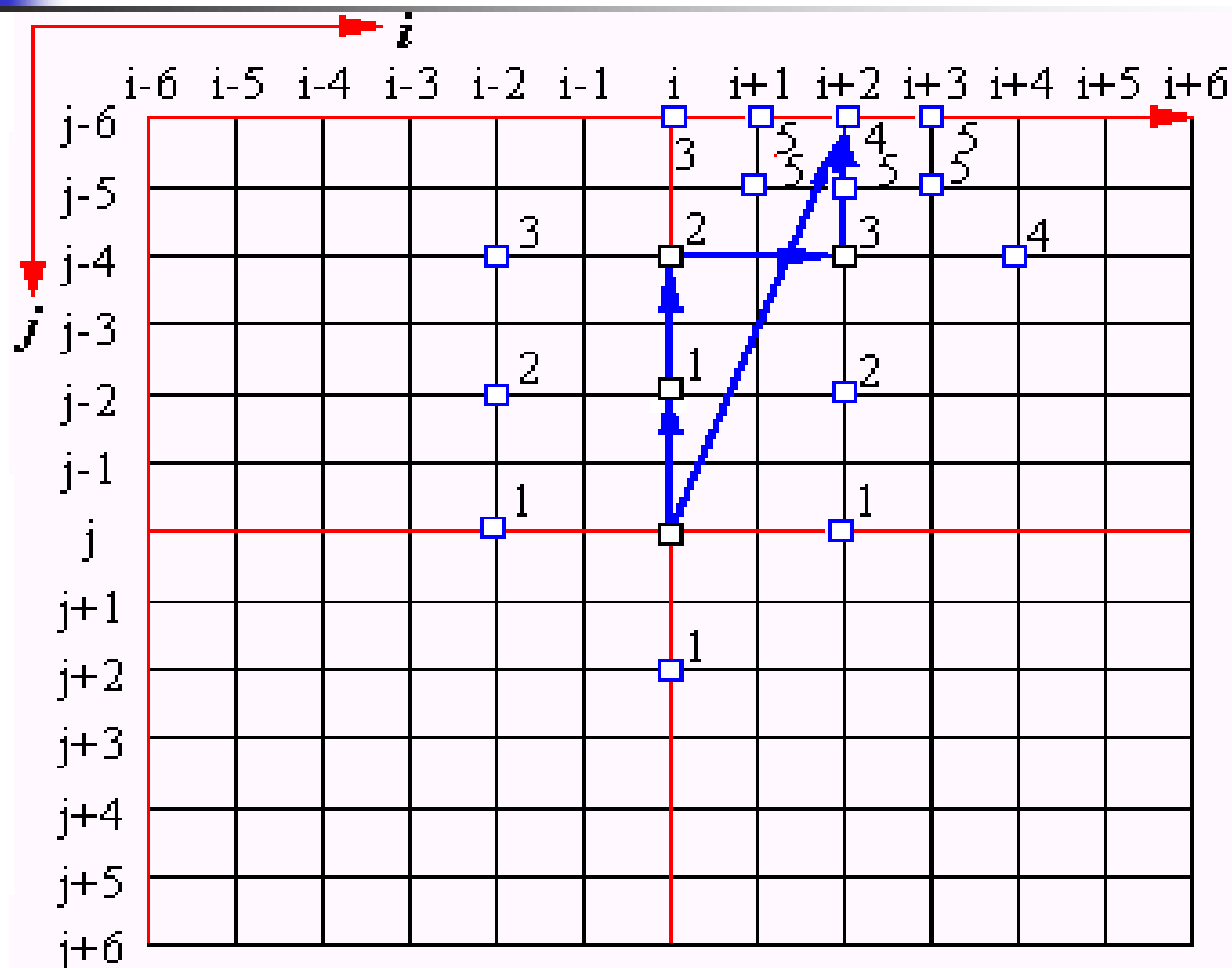


块匹配算法(BMA)

➤ 最优匹配点的搜索方法

- **穷尽搜索** (full search, 也称**全搜索**)
- **快速搜索**: 其算法共同之处在于它们把使准则函数 (例如, MAD) 趋于极小的方向视同为最小失真方向, 并假定准则函数在偏离最小失真方向时是单调递增的, 即认为它在整个搜索区内是 (i,j) 的单极点函数, 有唯一的极小值, 而快速搜索是从任一猜测点开始沿最小失真方向进行的。
- **分级搜索**: 先通过对原始图像滤波和亚采样得到一个图像序列的低分辨率表示, 再对所得低分辨率图像进行全搜索。由于分辨率降低, 使得搜索次数成倍减少, 这一步可以称为粗搜索。然后, 再以低分辨率图像搜索的结果作为下一步细搜索的起始点。经过粗、细两级搜索, 便得到了最终的运动矢量估值。







第5章 数字图像与视频压缩编码原理

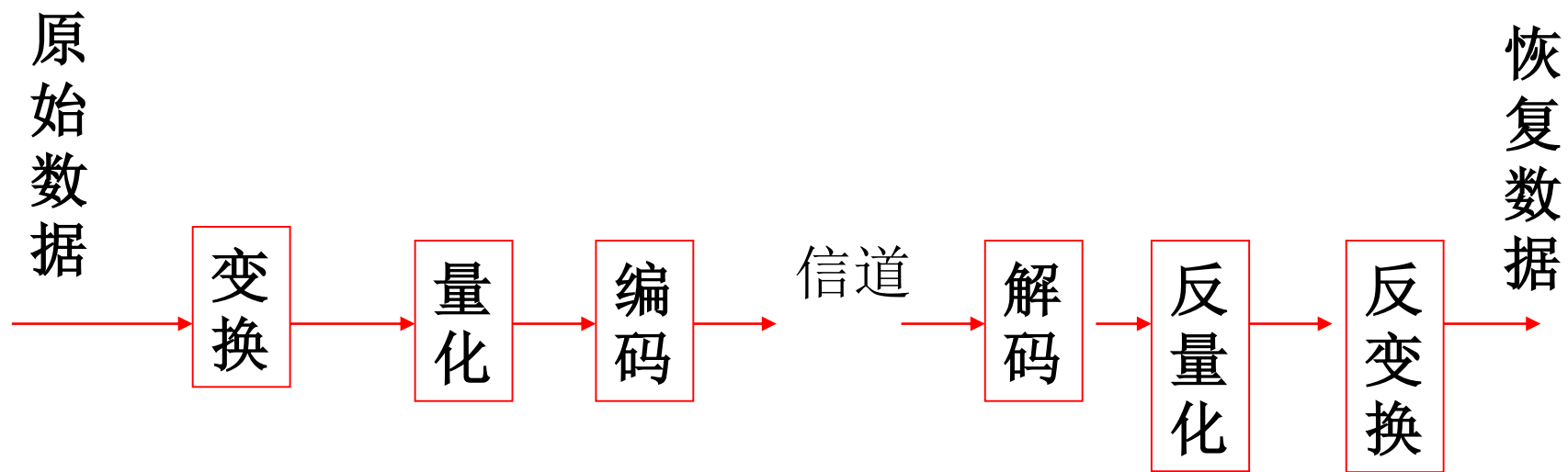
- 5.1 数字图像与视频压缩编码概述
- 5.2 熵编码
- 5.3 预测编码
- **5.4 变换编码**
- 5.5 MATLAB编程实例



5.4.1 变换编码的基本原理

- 预测编码希望通过对信源建模尽可能精确地预测数据，然后对预测误差进行编码。
- 变换编码的思路：将原始数据从时间域或者空间域“变换”到另一个更为紧凑表示、适合于压缩的变换域（通常为频域），从而得到比预测编码更高效的数据表示（压缩）。
- 预测编码消除相关性的能力有限，变换编码是一种更高效的压缩编码。

变换编码的通用模型



5.4.1 变换编码的基本原理

- 变换编码不直接对空间域图像数据进行编码，而是首先将空间域图像数据映射变换到另一个正交向量空间（变换域），得到一组变换系数，然后对这些变换系数进行量化和编码。
- 变换编码系统通常包括正交变换、变换系数选择和量化编码3个模块。
- 为了保证平稳性和相关性，同时也为了减少运算量，在变换编码中，一般在发送端的编码器中，先将一帧图像划分成若干个 $N \times N$ 像素的图像块，然后对每个图像块逐一进行变换编码，最后将各个图像块的编码比特流复合后再传输。在接收端，对收到的变换系数进行相应的逆变换，再恢复成图像数据。



5.4.2 正交变换基的选择

- 选择不同的正交基向量，可以得到不同的正交变换。常用的正交变换包括：

离散傅里叶变换 (**DFT**)

离散余弦变换 (**DCT**)

Karhunen—Loeve变换 (**K-L**)

沃尔什-哈达玛变换 (**WHT**)、

离散小波变换 (**DWT**)

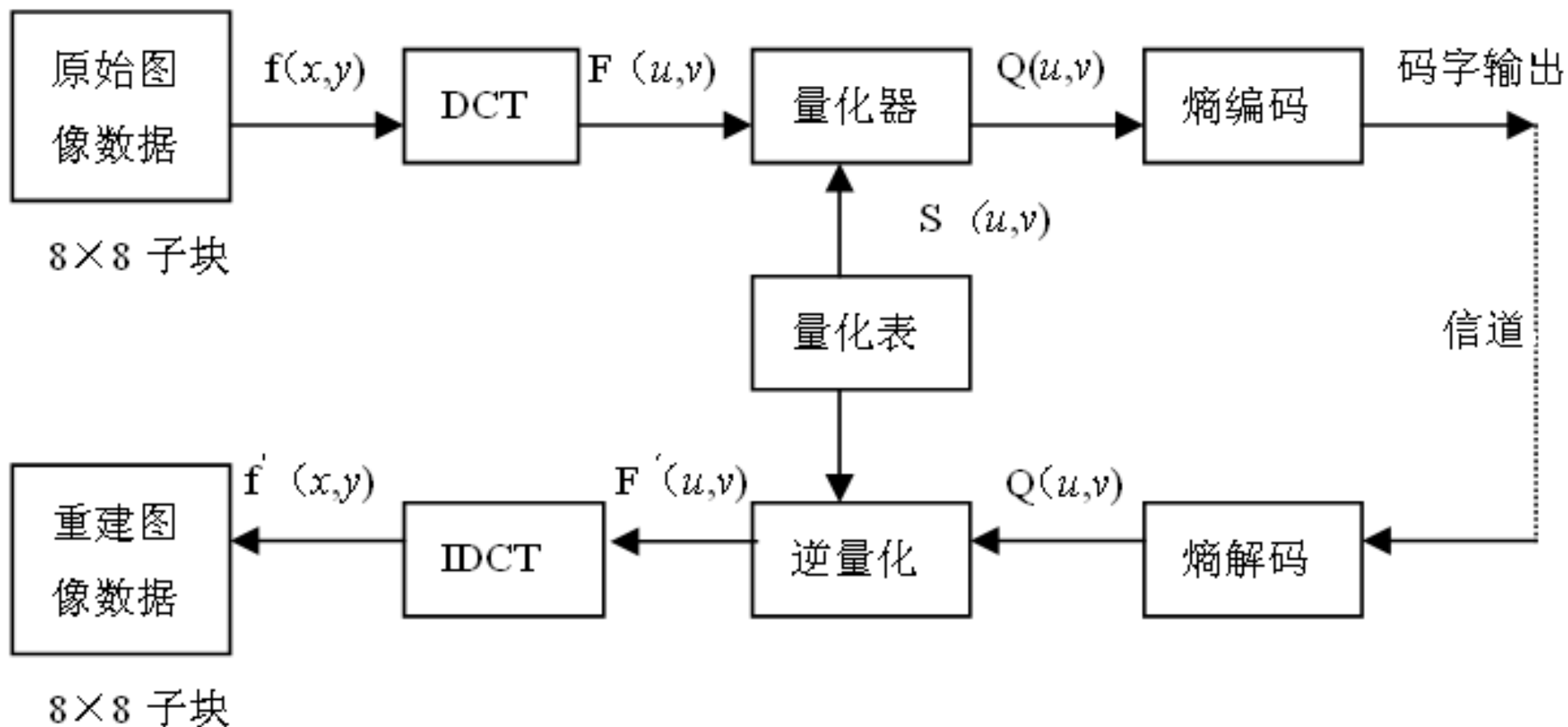


5.4.2 正交变换基的选择

- K-L变换能使变换后协方差矩阵为**对角阵**，并且有**最小均方误差 (MSE)**，因此称为在MSE最小准则下的**最佳变换**。
- 由于K-L变换是取原图像各子块的**协方差矩阵的特征向量**作为变换基向量，因此K-L变换的变换基是**不固定的**，且与编码对象的统计特性有关，**没有快速算法**，计算复杂性高，使得K-L变换的应用不现实。
- 对大多数图像信源来说，DCT的性能最接近K-L变换，同时其变换基向量是固定的，且有快速算法，故DCT广泛应用于图像/视频压缩。

5.4.3 基于DCT的图像编码

■ 基于DCT的编码和解码原理





5.4.3 基于DCT的图像编码

8 × 8 二维DCT变换

$$F(u, v) = \frac{1}{4} C(u) C(v) \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

8 × 8 二维DCT反变换

$$f(x, y) = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 C(u) C(v) F(u, v) \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

当 $u = v = 0$ 时, $C(u) = C(v) = \frac{1}{\sqrt{2}}$

当 u, v 为其他值时 $C(u) = C(v) = 1$



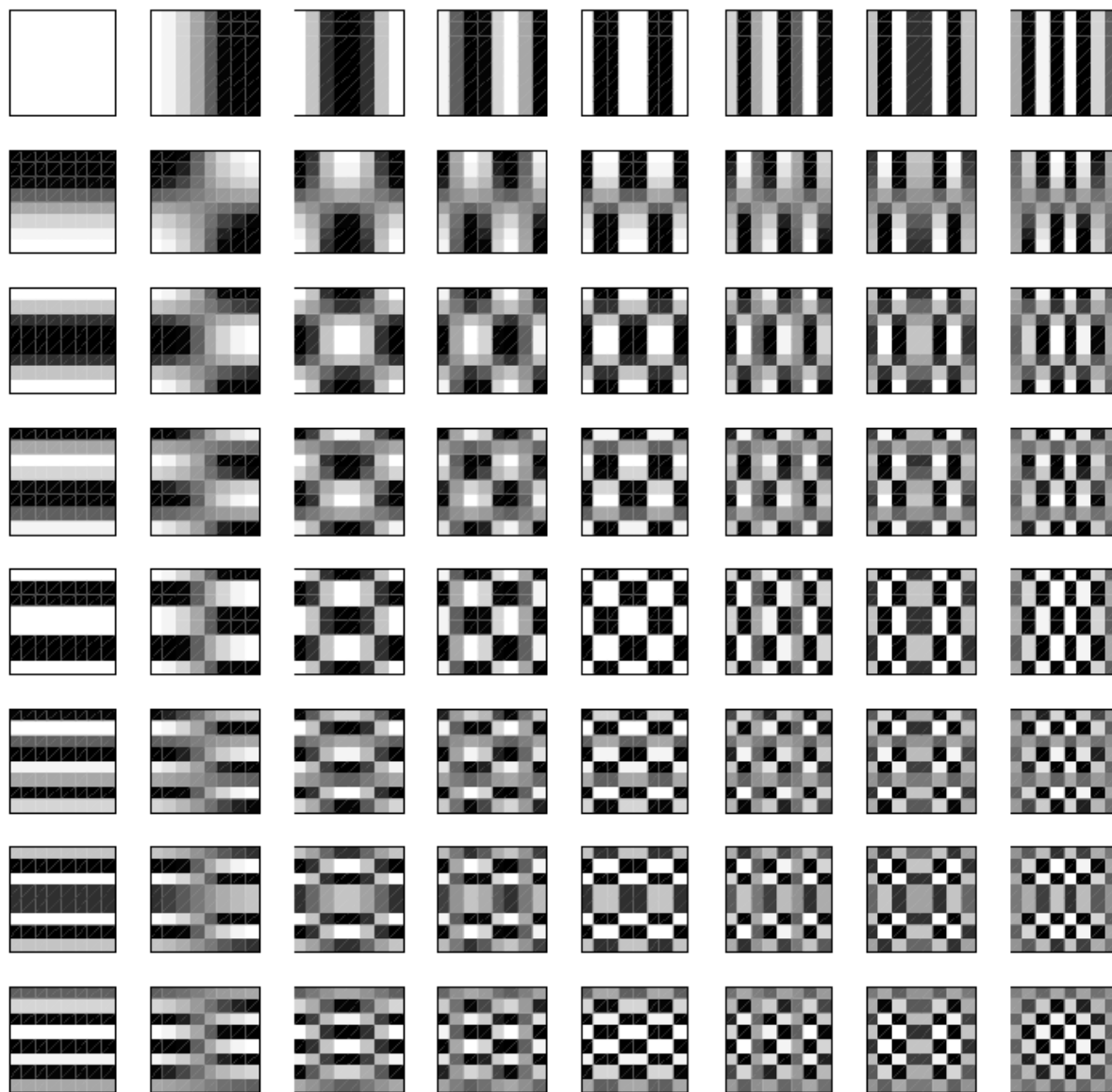
5.4.3 基于DCT的图像编码

8×8 二维DCT反变换的变换核函数为

$$C(u)C(v)\cos\frac{(2x+1)u\pi}{16}\cos\frac{(2y+1)v\pi}{16}$$

按 u ， v 分别展开后得到64个 8×8 像素的图像块组，称为基图像。

8×8二维DCT变换基图像





5.4.3 基于DCT的图像编码

量化

量化处理是一个多到一的映射，它是造成DCT编解码信息损失的根源。

根据人眼的视觉特性，对不同的变换系数设置不同的量化步长。

$$Q(u, v) = \text{round}\left[\frac{F(u, v)}{S(u, v)}\right]$$



JPEG标准中亮度DCT系数的量化步长

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

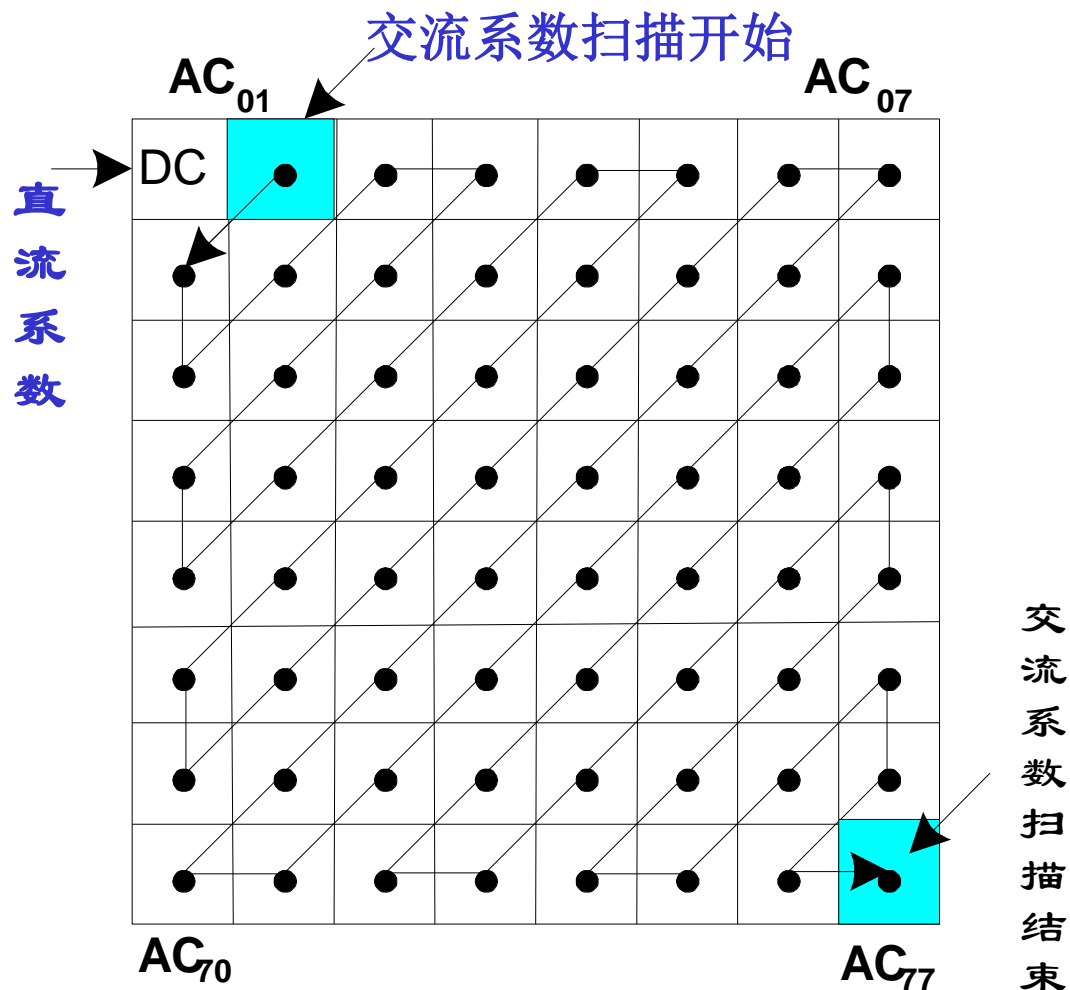


JPEG标准中色度DCT系数的量化步长

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

变换系数熵编码

❧ Zig-Zag（或称“Z”字形，“之”字形）扫描





变换系数熵编码

- ◆ **直流分量 (DC)** : 相邻图像子块的直流分量 (图像子块的平均样值) 也存在着相关性, 所以对DC的量化系数用**DPCM**编码较合适, 即对当前块和前一块的DC系数的差值进行编码。
- ◆ **交流分量 (AC)** : 把数值为0的连续长度 (即0游长) 和非0值结合起来构成**一个事件 (Run, Level)** , 然后再对事件 (Run, Level) 进行熵编码。