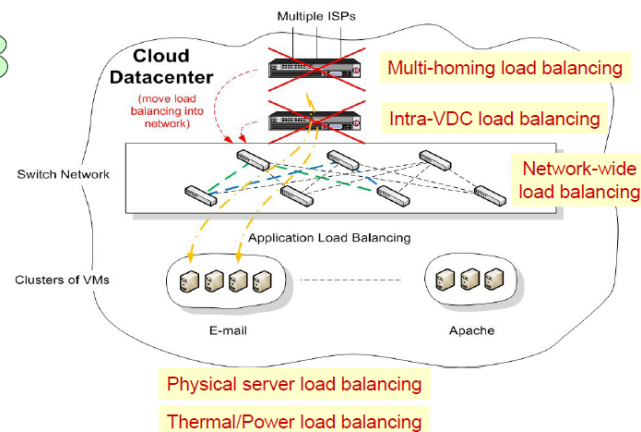
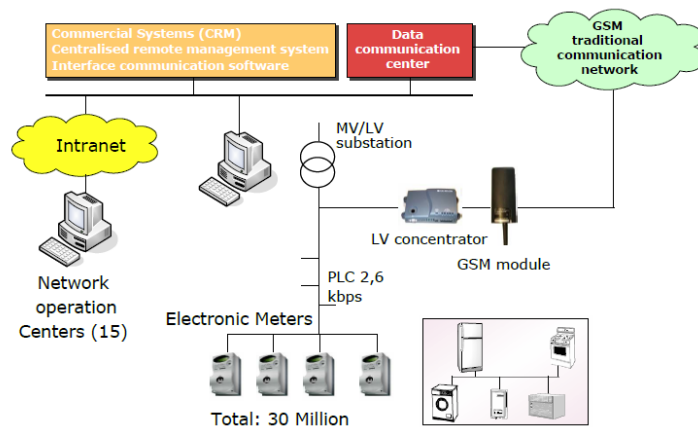
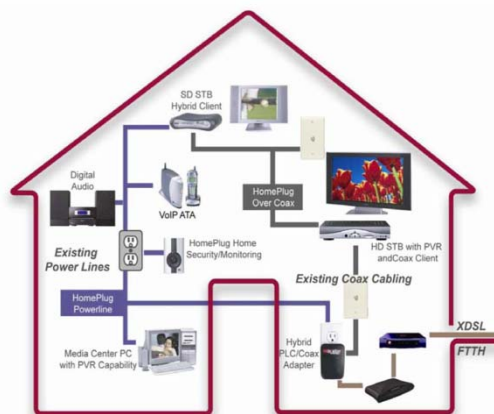




# 北京大学信息管理系



## 考前整理

03030910 多媒体技术

W. B. Huang



# 媒体(Media, Medium)

2

- 媒体（medium）在计算机领域有两种含义：即媒质和媒介。
  - 媒质：存储信息的实体，如磁盘、光盘、磁带、半导体存储器等。
  - 媒介：传递信息的载体，如数字、文字、声音、图形和图像等。



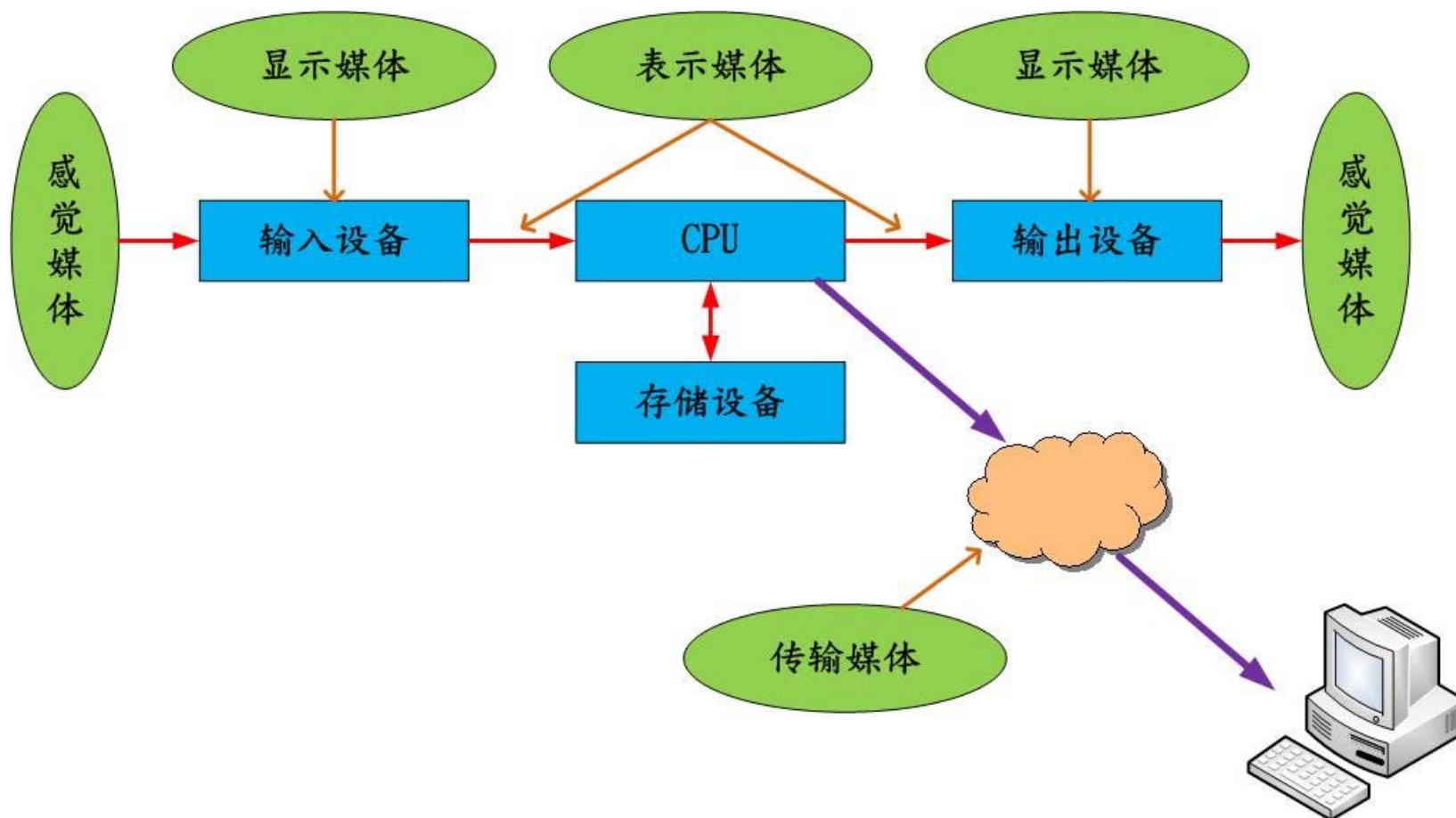
# CCITT:媒体的定义是...

3

- 感觉媒体 (Perception Medium)
  - 表示媒体 (Representation Medium)
  - 显示媒体 (Presentation Medium)
  - 存储媒体 (Storage Medium)
  - 传输媒体 (Transmission Medium)
- 
- CCITT(International Telephone and Telegraph Consultative Committee)是国际电报电话咨询委员会的简称，它是国际电信联盟 (ITU) 的常设机构之一。
  - 1993年3月1日起，国际电报电话咨询委员会 (CCITT) 改组为国际电信联盟 (ITU) 电信标准化部门，简称ITU-T。

# 媒体

4





# 感觉媒体 (Perception Medium)

5

- 感觉媒体是指能直接作用于人的感官，使人能直接产生感觉的一类媒体。感觉媒体有人类的各种语言、音乐，自然界的各种声音、图形、静止和运动的图像等。

类型	分类
视觉媒体	文字、景象
听觉媒体	语言、音乐、自然界的各种声音
触觉媒体	力、运动、温度
味觉媒体	滋味
嗅觉媒体	气味

# 表示媒体 (Representation Medium)



6

- 表示媒体是为了加工、处理和传输感觉媒体而人为地研究、构造出来的一种媒体。
- 其目的是能将感觉媒体从一个地方向另一个地方传送，以便于加工和处理。
- 表示媒体有各种编码方式如语音编码、文本编码、静止和运动图像编码等。

# 显示媒体（Presentation Medium）

7

- 显示媒体是指感觉媒体与用于通信的电信号之间转换用的一类媒体。它包括输入显示媒体（如键盘、摄像机、话筒等）和输出显示媒体（如显示器、喇叭和打印机等）。



# 存储媒体 (Storage Medium)

8

- 存储媒体是**用来存放表示媒体**，以方便计算机处理加工和调用，这类媒体主要是指与计算机相关的外部存储设备。



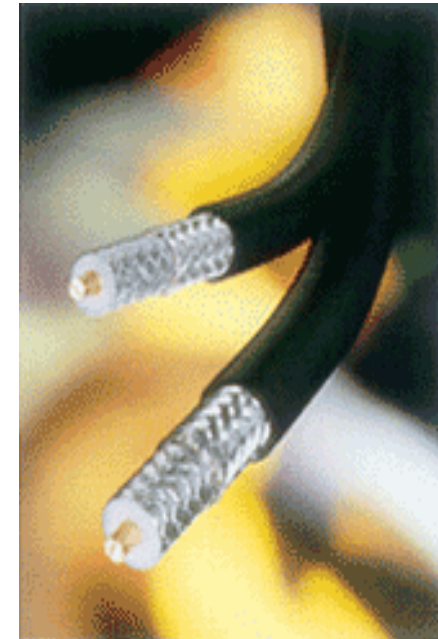


# 传输媒体 (Transmission Medium)



9

- 传输媒体是用来将媒体从一个地方传送到另一个地方的物理载体。传输媒体是通信的信息载体，如双绞线、同轴电缆、光纤等。





# 多媒体四个最重要的特征

10

- 多维化
  - ▣ 是指信息媒体的多样化。它使人们思想的表达不再限于顺序的、单调的、狭小的范围内，而有充分自由的余地。
- 集成性
  - ▣ 这不仅是指多媒体设备集成，而且也包含多媒体信息集成或表现集成。例如，仅有静态图像而无动态视频、仅有声音而无图形等。
- 交互性
  - ▣ 多媒体信息空间中的交互性可以增加对信息的注意和理解。
- 实时性
  - ▣ 实时性又称为动态性，是指多媒体技术中涉及的一些媒体，例如，音频和视频信息具有很强的时间特性，会随着时间的变化而变化。



# MPC标准

11

项目	MPC1	MPC2	MPC3
内存	2MB	4MB	8MB
运算处理器	16MHz 386SX	25 MHz 486SX	75 MHz Pentium 同等级X86
CD-ROM	150kB/s 最大寻址时间1s	300kB/s 最大寻址时间400ms CD-ROM XA	600kB/s 最大寻址时间200ms CD-ROM XA
声卡	8bit数字声音 8个合成音 MIDI	16bit数字声音 8个合成音 MIDI	8bit数字声音 Wavetable(波表) MIDI
显示	640×480 16色	640×480 65536色	640×480 65536色
硬盘容量	30MB	160MB	540MB
彩色视频播放	-	-	352×240 30fps
输入输出端口	MIDI I/O, 摇杆端口, 串并接口	MIDI I/O, 摇杆端口, 串并接口	MIDI I/O, 摇杆端口, 串并接口



# 数据压缩及编码技术

12

- 多媒体系统要求具有综合处理声、图、文的能力，面临的主要问题是巨大的数据量，尤其是对动态图形和视频图像
  - ▣ 例如:一幅分辨率为 $640 \times 480$ ，真彩色图像，8bit/像素，需存储空间为  $= 640 \times 480 \times 8 \times 3 = 7372800 \text{ bit} = 0.878906 \text{ MB}$
  - ▣ 650MB的CD光盘存放  $= 650 \div 0.879 \approx 739$ 张(帧)图像
  - ▣ 全动态视频画面(每秒25帧)，只能播放  $= 739 \div 25 \approx 29$ 秒图像信息。



# 声音质量和数据率

13

□ 数据率 = 采样频率 x 量化位数（精度） x 通道数目

例：电话语音 =  $8\text{k} \times 8\text{b} \times 1 = 64\text{kbps} = 8\text{kB/s} = 28\text{MB/h}$

质量	采样频率 (kHz)	样本精度 (bit/s)	单道声/ 立体声	数据率(kB/s) (未压缩)	频率范围
电话*	8	8	单道声	8	200~3,400 Hz
AM	11.025	8	单道声	11.0	20~15,000 Hz
FM	22.050	16	立体声	88.2	50~7,000 Hz
CD	44.1	16	立体声	176.4	20~20,000 Hz
DAT	48	16	立体声	192.0	20~20,000 Hz



# 1分钟数字视频信号需要的存储空间

14

□ CCIR 601 NTSC 数据量 =  $13.5 \text{ M} \times 16 \text{ bit} / 8 \times 60 = 1620 \text{ MB}$

数字电视格式	分辨率×速率	采样率	量化位数	存储容量(MB)
公用中间格式 (CIF)	352×288×30	3 MHz (4:1:1)	亮度、色差共12	270
CCIR 601号建议	NTSC 720×480×30 PAL 720×576×25	13.5 MHz (4:2:2)	亮度、色差共16	1620 1620
HDTV 亮度信号	1280×720×60	60 MHz	8	3600

国际无线电咨询委员会(International Radio Consultative Committee)，CCIR是简称。



# 数据压缩及编码技术

15

- 时间域压缩——迅速传输媒体信源
- 频率域压缩——并行开通更多业务
- 空间域压缩——降低存储费用
- 能量域压缩——降低发射功率



# 多媒体输入输出技术

16

- 多媒体输入/输出技术包括媒体变换技术、媒体识别技术、媒体理解技术和综合技术。
  - ▣ 媒体变换技术是指改变媒体的表现形式。如当前广泛使用的视频卡、音频卡（声卡）都属媒体变换设备。
  - ▣ 媒体识别技术是对信息进行一对一的映像过程。例如，**图像识别、语音识别技术和触摸屏技术**等。
  - ▣ 媒体理解技术是对信息进行更进一步的分析和理解信息内容。如自然语言理解、图像理解、模式识别等技术。
  - ▣ 媒体综合技术是把低维信息表示映像成高维的模式空间的过程。例如**语音合成器就可以把语音的内部表示综合为声音输出**。





# 多媒体输入输出技术

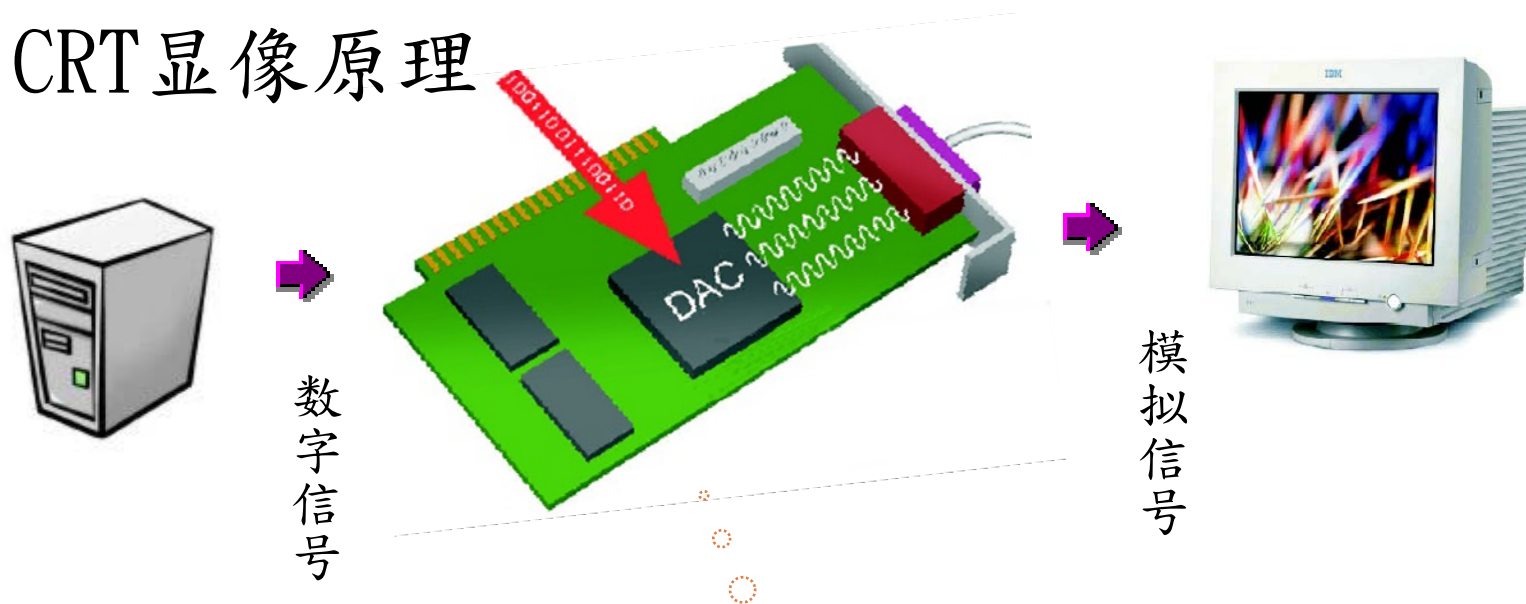
17

- 综合地利用输入输出技术实现用户和计算机之间更加自然的交互是人机界面设计的目标。为了达到这个目的，需要在以下几个方面进行研究：
  - ▣ 稳健的语言处理模式，包括语音识别和自然语言理解。
  - ▣ 手势分析和理解模型的设计。
  - ▣ 上述两个方面的通信模式的融合，因为二者之间在对用户需求的理解上是相互补充的。
  - ▣ 多模式环境中的对话管理。这是保证一个连续的对话过程所必需的。

# 显像输出

18

## CRT显像原理



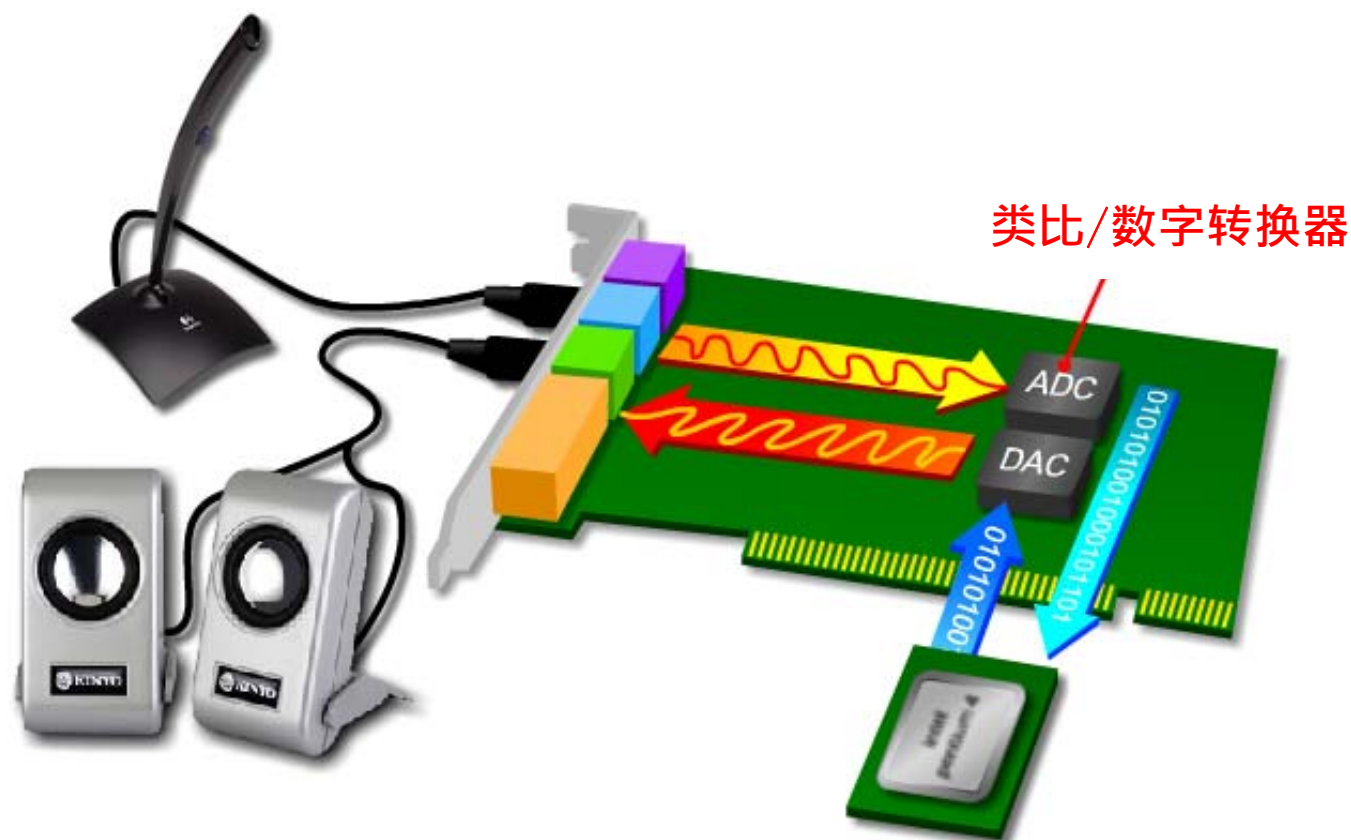
显示适配器—产生影像讯号并输出至屏幕上

DAC (digital-to-analog converter)

数字信号转换为模拟信号

# 模拟讯号转换成数字讯号

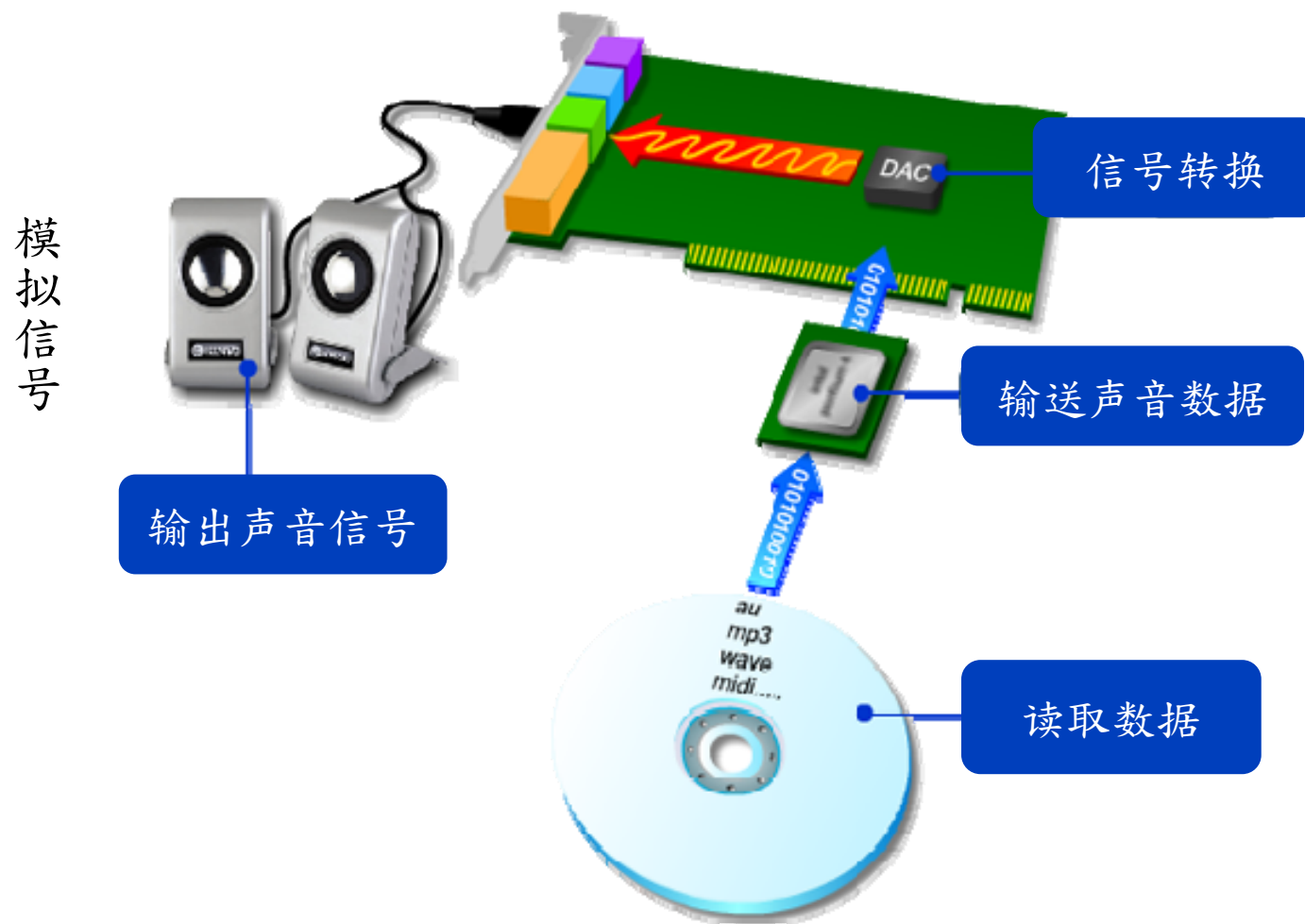
19



DAC (digital-to-analog converter)  
数字信号转换为模拟信号

# 声卡输出声音到喇叭的运作图

20





# 多媒体数据准备软件

21

- 多媒体数据准备软件是指用于采集多种多媒体数据的软件，如声音录制、编辑软件；图像扫描及预处理软件、全动态视频采集软件、动画生成编辑系统等。
  - 例：Windows环境下声音和视频数据的采集



# 多媒体创作工具

22

- 多媒体数据库和创作工具为多媒体应用提供资源和信息加工。
- 多媒体创作工具的分类：
  - ▣ 媒体创作软件工具，用于建立媒体模型，产生媒体数据
  - ▣ 多媒体节目写作工具，提供不同的编辑、写作方式
  - ▣ 媒体播放工具：可以在电脑上播出，有的甚至能在消费类电子产品中播出
  - ▣ 其他各类媒体处理工具
- 例：音频处理软件；图形、图像及动画制作与编辑软件；网上音、视频文件制作



# 多媒体应用软件

23

- 应用软件主要为用户提供在各个具体领域中的辅助功能，它也是绝大多数用户学习、使用计算机时最感兴趣的内容。
- 应用软件的内容很广泛，涉及到社会的许多领域，很难概括齐全，也很难确切地进行分类。常见的应用软件有如下几种：
  - ▣ 各种信息管理软件
  - ▣ 办公自动化系统
  - ▣ 各种文字处理软件
  - ▣ 各种辅助设计软件以及辅助教学软件
  - ▣ 各种软件包，如数值计算程序库、图形软件包等



# 多媒体播放器

24

- 多媒体播放器是指那些能够回放不同编码格式音视频文件的软件。这类软件一般分为两类：
  - ▣ 一类是运行在个人计算机上并用来播放本地存储的音视频文件的播放器，目前，这类播放软件很多
  - ▣ 另一类就是播放基于Web的音视频流的播放器。
- 能够播放基于Web的音视频流的播放器包括
  - ▣ Apple公司的QuickTime
  - ▣ Microsoft公司的Windows Media Player
  - ▣ Real Networks推出的RealPlayer。





# 矢量图(vector graphics)

25

- 根据数学规则描述而生成的图
  - ▣ 一幅图用数学描述的**点、线、弧、曲线、多边形**和其他几何实体和几何位置来表示，创建的图是对象的集合而不是点或像素模式的图
- 优点
  - ▣ **目标图像的移动、缩小或放大、旋转、拷贝、属性(如线条变宽变细、颜色)变更都很容易做到**
  - ▣ 相同或类似的图可以把它们当作图的构造块，并把它们存到图库中，这样不仅可加速矢量图的生成，而且可减小矢量图的文件大小
- 局限性
  - ▣ **很难用数学方法来描述真实世界的彩照，这就要用位图法表示**

# 位图(bitmap, bitmapped image)



26

- 用像素值阵列表示的图
  - ▣ 对位图进行操作时，只能对图中的像素进行操作，而不能把位图中的物体作为独立实体进行操作。
  - ▣ 画位图或编辑位图的软件称为画图程序(paint programs)；存放位图的格式称为位图格式；存储的内容是描述像素的数值
- 特性
  - ▣ 位图的获取通常用扫描仪、数码相机、摄像机、录像机、视像光盘和相关的数字化设备
  - ▣ 位图文件占据的存储空间比较大
  - ▣ 影响位图文件大小的因素
    - 图像分辨率：分辨率越高，表示组成一幅图的像素就越多，图像文件就越大
    - 像素深度：像素深度越深，表达单个像素的颜色和亮度的位数越多，图像文件就越大

# 灰度图(gray-scale image或intensity image)

27

- 只有明暗不同的像素而没有彩色像素组成的图像
- 只有黑白两种颜色的图像称为单色图像(monochrome/bit image)
  - ▣ 每个像素的像素值用一位存储，其值是“0”或“255”(通常用“1”来储存)
- 用一个字节表示一个像素的灰度图(256级灰度)
  - ▣ 一幅640×480的灰度图像需要占据300 KB的存储空间



# 彩色图像(color image)

28

- 每个像素包含颜色信息的图像。
- 可按照颜色的数目划分
  - 256色图像：每个像素的R、G和B值用一个字节来表示，一幅640×480的彩色图像需要300 KB的存储空间
  - 真彩色图像：每个像素的R，G，B分量分别用一个字节表示，一幅640×480的真彩色图像需要900 KB的存储空间



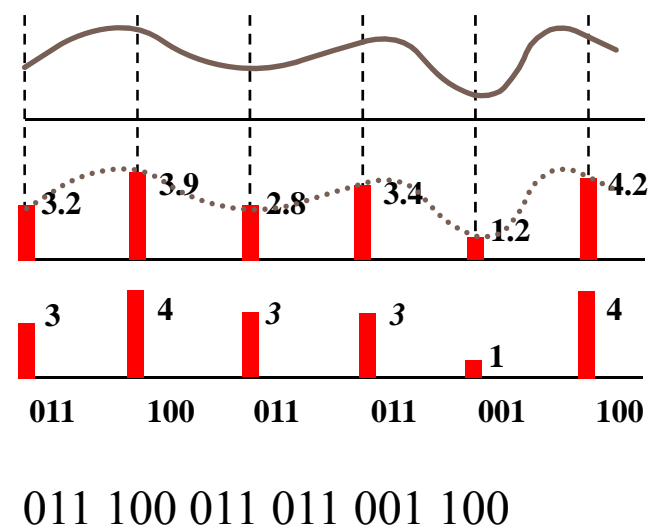
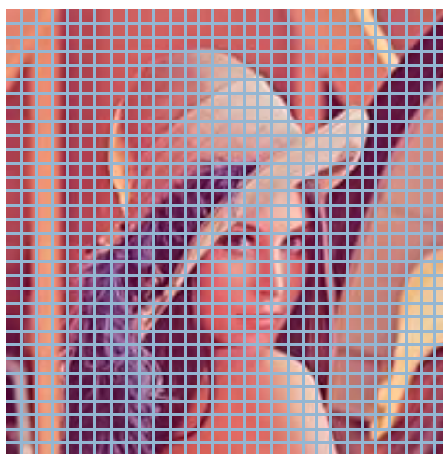
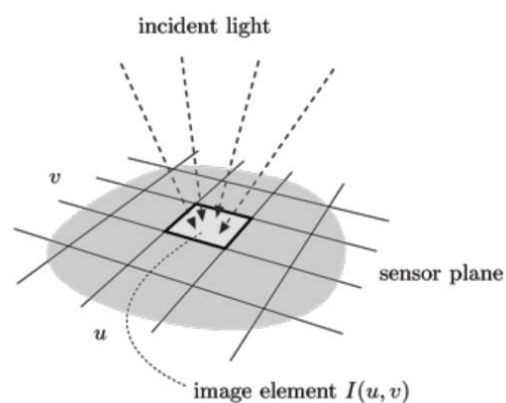
256色标准图像



24位标准图像

# 图像数字化

29





# 图像的清晰度

30

- 与清晰度相关的主要因素
  - ▣ 亮度
  - ▣ 对比度
  - ▣ 尺寸大小
  - ▣ 细微层次(灰度)
  - ▣ 颜色饱和度



# 图像分辨率

31

- 在图像显示应用中的图像分辨率表示法
  - ▣ (1)物理尺寸：每毫米线数(或行数)
  - ▣ (2)行列像素：像素/行 $\times$ 列/幅，如640像素/行 $\times$ 480列/幅
  - ▣ (3)像素总数：如数码相机上标的500万像素
  - ▣ (4)单位长度上的像素：如像素每英寸(pixels per inch, PPI)
  - ▣ (5)线对(line pair)数：以黑白相邻的两条线为一对，如5对线
- 在图像数字化和打印应用中的图像分辨率表示法
  - ▣ 通常用多少点每英寸(dots per inch, DPI)表示，如300 DPI
  - ▣ 分辨率越高，图像质量就越高，像素就越多，要求存储容量就越大
- 图像分辨率与屏幕分辨率是两个不同的概念
  - ▣ 从行列像素角度看，图像分辨率是构成一幅图像的像素数目，而屏幕分辨率是显示图像的区域大小



# 真彩色、伪彩色与直接色

32

- 真彩色(true color)
  - ▣ 每个像素的颜色值用红(R)、绿(G)和蓝(B)表示的颜色
  - ▣ 通常用24位表示，其颜色数 $2^{24} = 16\,777\,216$ 种。也称24位颜色(24-bit color)或全彩色(full color)
- 伪彩色(pseudo color)
  - ▣ 不是物体固有的而是人为的颜色
- 直接色(direct color)
  - ▣ 每个像素值由R，G，B分量构成，每个分量作为单独的索引值对它做变换，用变换后的R，G，B值产生的颜色。





# 伪彩色

33

## □ Pseudo color

- ▣ 每个像素的颜色不是由每个基色分量的数值直接决定，而是把像素值当作彩色查找表CLUT(color look-up table)的表项入口地址，去查找一个显示图像时使用的R，G，B强度值。

## □ 彩色查找表CLUT

- ▣ 也称为：colormap（颜色图），palette（调色板）
- ▣ 是一个事先做好的表，表项入口地址也称为索引号，根据该索引号可查找出包含实际R、G、B的强度值。
- ▣ 例如16种颜色的查找表，0号索引对应黑色，...，15号索引对应白色。



# 直接色

34

- 每个像素值分成R，G，B分量，每个分量作为单独的索引值对它做变换。
  - ▣ 从彩色索引表找出基色强度
  - ▣ 用矩阵变换后得到的R，G，B强度值产生的彩色称为直接色。
- 真彩色与直接色的比较
  - ▣ 相同之处是都采用R，G，B分量决定基色强度
  - ▣ 不同之处是前者的基色强度直接用R，G，B决定，而后者的基色强度由R，G，B经变换后决定。
  - ▣ 产生的颜色有差别。

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



# 抽样系统

35

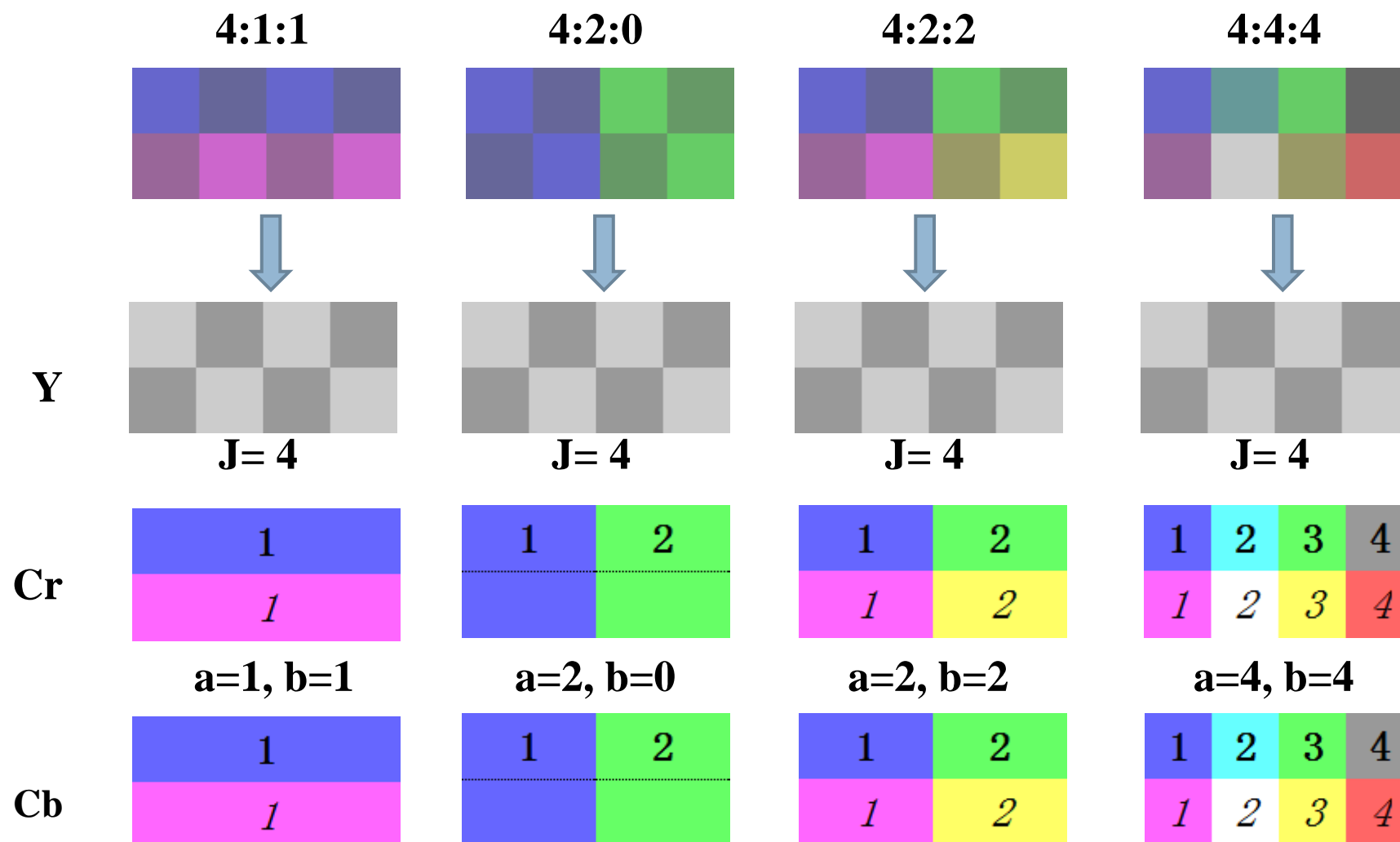
- 视频系统的抽样系统中通常用一个三分比值表示：
  - ▣  $J:a:b$ （例如4:2:2），形容一个以J个像素宽及两个像素高的概念上区域。
  - ▣ 依序列出为：
    - J：水平抽样参照（概念上区域的宽度）。通常为4。
    - a：在J个像素第一行中的色度抽样数目(Cr, Cb)。
    - b：在J个像素第二行中的额外色度抽样数目(Cr, Cb)。



# YCrCb

## 常用色度抽样系统的范例

36





# 直方图均衡化

37

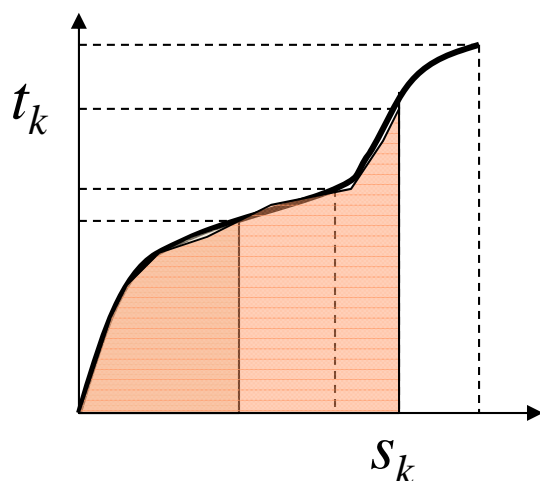
设  $p_s(s_k) = n_k/n$ ,  $0 \leq s_k \leq 1, k=0,1,\dots,L-1$ . 即灰度  $s_k$  的像素占全体像素的比例  
令

$$t_k = E_H(s_k) = \sum_{i=0}^k \frac{n_i}{n} = \sum_{i=0}^k p_s(s_i), 0 \leq s_k \leq 1, k = 0, 1, \dots, L-1.$$

$E_H(s_k)$  是单调增加函数，它等于灰度在  $s_k$  以下的像素所占的比例。

可以写出反函数：

$$s_k = E_H^{-1}(t_k), 0 \leq t_k \leq 1, k = 0, 1, \dots, L-1.$$



假设  $t_k = E_H(s_k) = 1/4$ ，那么，灰度  $s_k$  映射到  $t_k$  意味着  $t_k = 1/4$  以下的灰度占像素总数的  $1/4$ 。

如果  $t_k = E_H(s_k) = 1/2$ ，那么  $t_k = 1/2$  以下的灰度占像素总数的  $1/2$ 。

即  $t_k$  的直方图是均匀分布。



# 直方图均衡化的计算

38

- 假设原图像有 $L$ 个灰度，不妨记 $s_k$ 就是灰度 $k$ 。
- 依次计算频率 $p(k) = n_k/n, k=0,1,\dots,L-1$ .
- 计算累积直方图 $E_H(k), k=0,1,\dots,L-1$ .
- 用下式计算 $t_k$ 的近似值：

$$t_k = E_H(k) = \sum_{i=0}^k \frac{n_i}{n} = \sum_{i=0}^k p(k), k = 0, 1, \dots, L-1.$$

- 由于 $t_k$ 在0与1之间，应该把 $t_k$ 映射到在 $[0, L-1]$ 范围内的一个整数，所以用 $L-1$ 乘以 $t_k$ 后取整，即 $[(L-1) t_k]$ 。



# 直方图均衡化的范例

39

序号	运 算	步骤和结果							
1	列出原始图灰度级 $f_k, k=0, 1, \dots, 7$	0	1	2	3	4	5	6	7
2	列出原始直方图	0.02	0.05	0.09	0.12	0.14	0.2	0.22	0.16
3	用式(3.2.2)计算原始累积直方图	0.02	0.07	0.16	0.28	0.42	0.62	0.84	1.00
4	取整 $g_k = \text{int}[(L-1)g_k+0.5]$	0	0	1	2	3	4	6	7
5	确定映射对应关系 ( $f_k \rightarrow g_k$ )	0, 1 $\rightarrow$ 0		2 $\rightarrow$ 1	3 $\rightarrow$ 2	4 $\rightarrow$ 3	5 $\rightarrow$ 4	6 $\rightarrow$ 6	7 $\rightarrow$ 7
6	计算新直方图	0.07		0.09	0.12	0.14	0.2	0.22	0.16

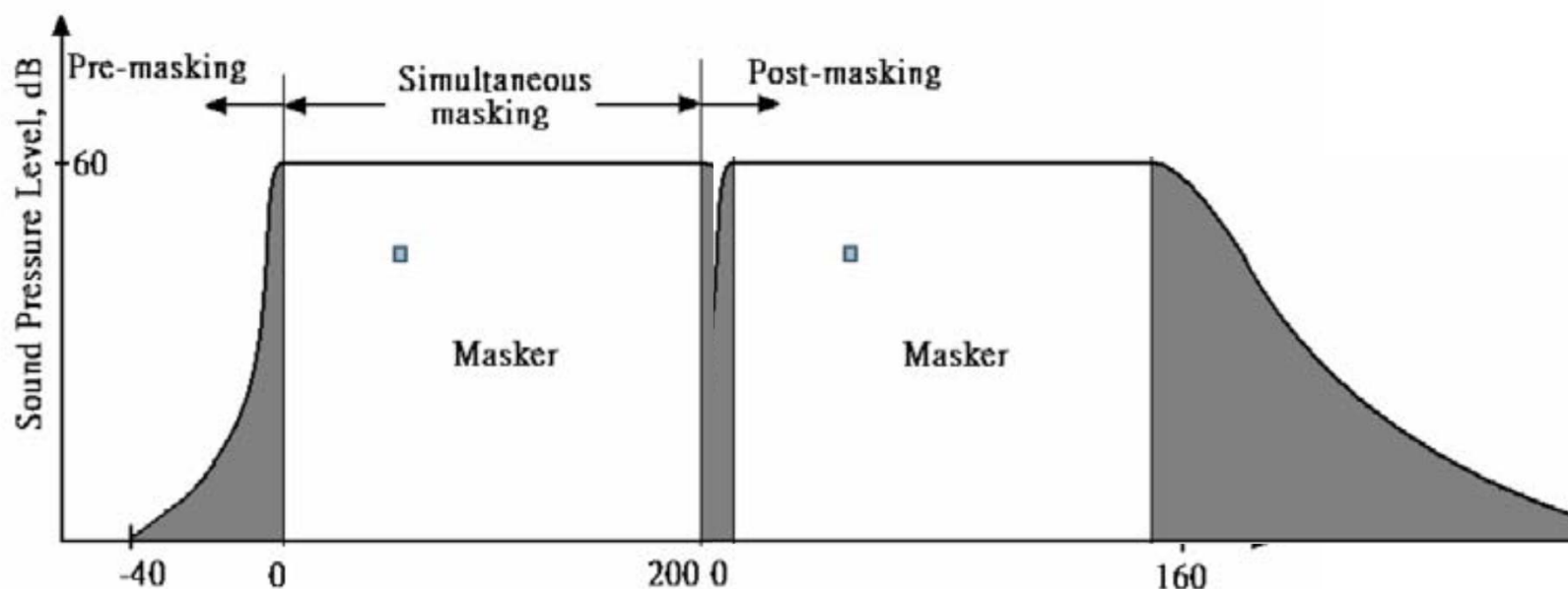
$$f_k = k$$

$$g_k = \text{int} \left[ (7-1) \sum_{i=0}^k \frac{n_i}{n} + 0.5 \right]$$

# 遮蔽效应

40

- 称听不到的声音为**被掩蔽声**，而起掩蔽作用的声音为**掩蔽声**。
- 掩蔽效应的实质是掩蔽声的出现使人耳听觉的等响度曲线的最小可听阈抬高。
- 掩蔽效应的一般规律是**强音压低音**、**低频率声音压高频率**。







# 音频特征提取-时域

41

## □ 平均能量：

- ▣ 说明了音频信号的强度，可用于静音检测
- ▣ 如这个音频例子中的某一短时帧的平均能量低于一个事先设定的阈值，则可判定该短时帧为静音。

## □ 过零率：

- ▣ 指每秒内信号值通过零值的次数，一定程度上说，它说明了平均信号频率。

## □ 静音比：

$$Z_0 = \frac{1}{2} \left\{ \sum | \text{sgn}[s_w(n)] - \text{sgn}[s_w(n-1)] | \right\}$$

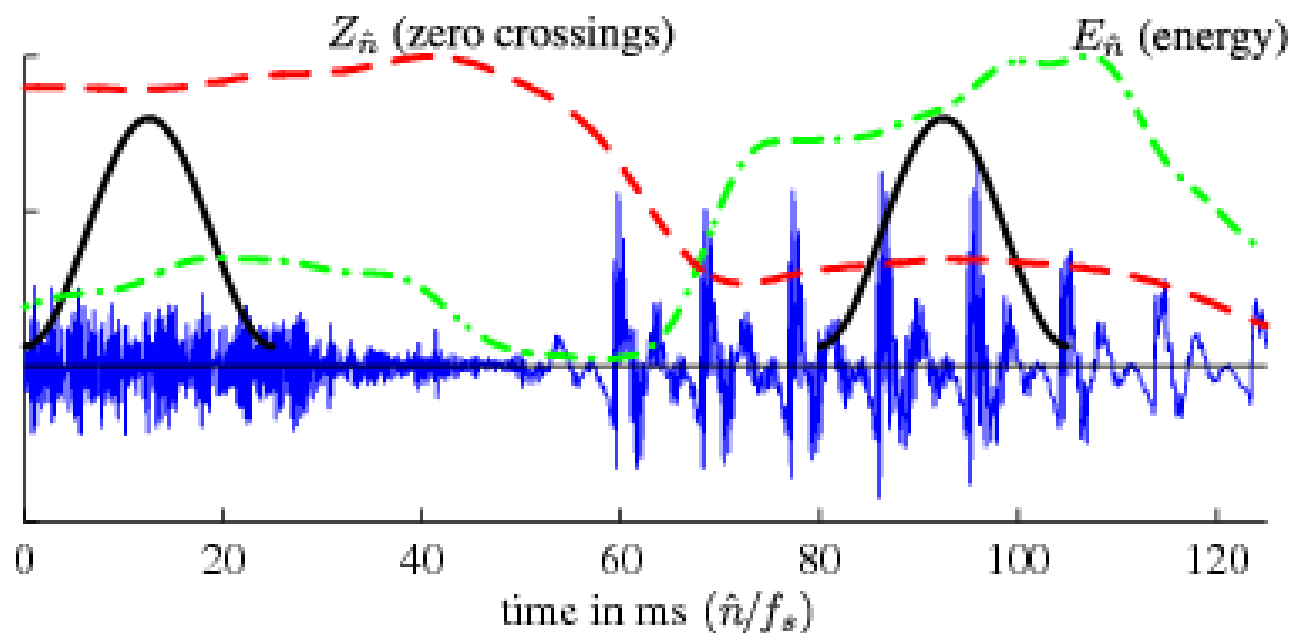
- ▣ 表示静音的声音片段的比例。
- ▣ 可计算为静音时段的总和与音频片段总长度之间的比值。



# 过零率、能量、静音

42

- 一般而言，噪声的过零大于气音的过零率，而气音的过零率又大于有声音的过零率。
- 可断点侦测，估测气音的起始位置和结束位置。
- 区分语音和音乐。





# 音频特征提取-频域

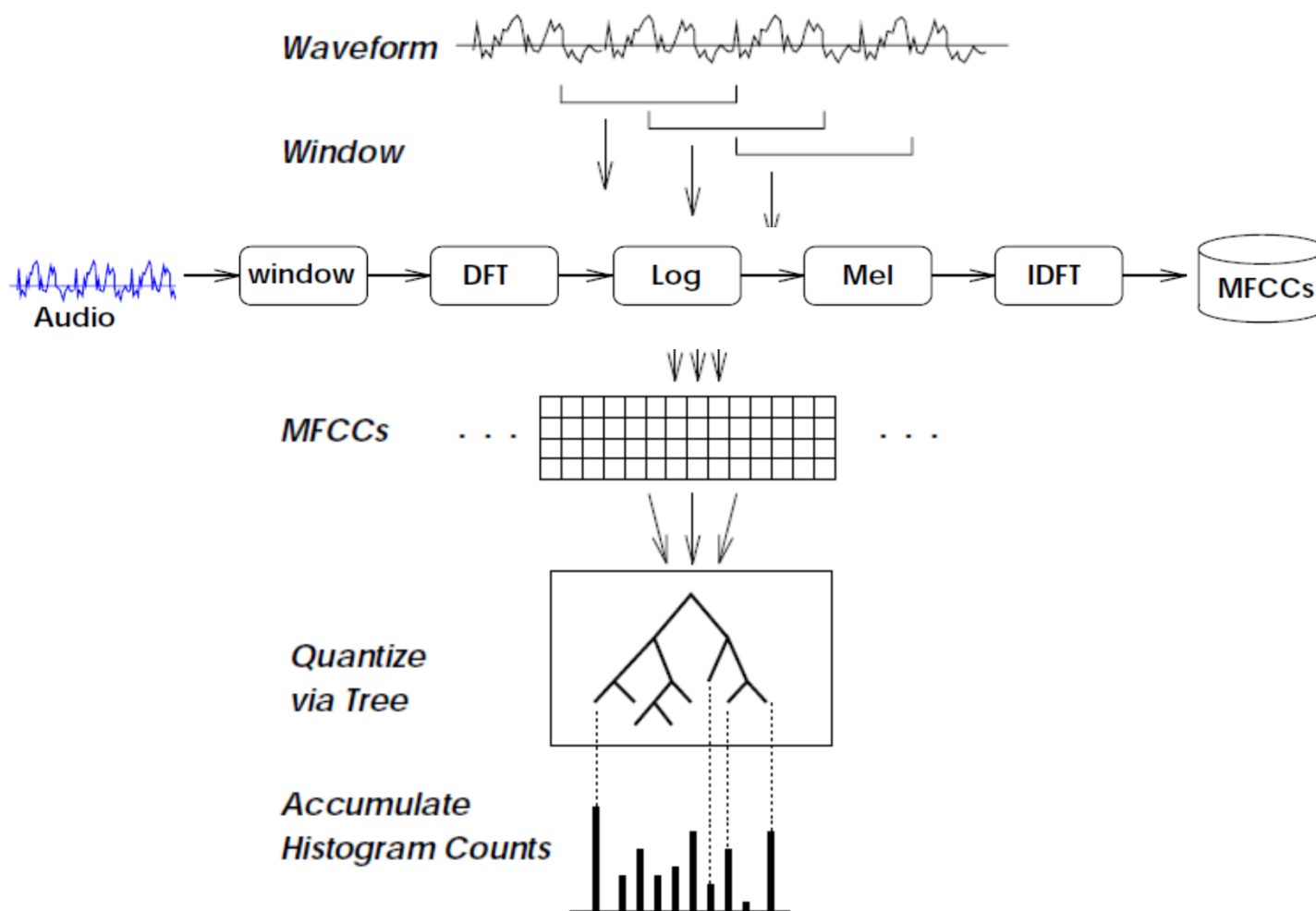
43

- 带宽：
  - ▣ 说明了声音的频率范围，音乐通常比语音信号具有更高的带宽。
- 频谱中心：
  - ▣ 也称亮度，是一个声音频谱能量分布的中心点。
  - ▣ 语音与音乐相比，频谱中心较低。
- 谐音：
  - ▣ 频率为最低频率的倍数的频谱成分称为谐音。
  - ▣ 在有谐音的声音中，频谱成分大部分是最低频率的整数倍数，音乐通常比其他声音具有更多的谐音。
- 音调：
  - ▣ 是听觉分辨声音高低的特性，完全由频率决定，可通过频谱估计。
  - ▣ 是一个主观特征，与基本频率有关，但不等同于基本频率。

# 音频处理



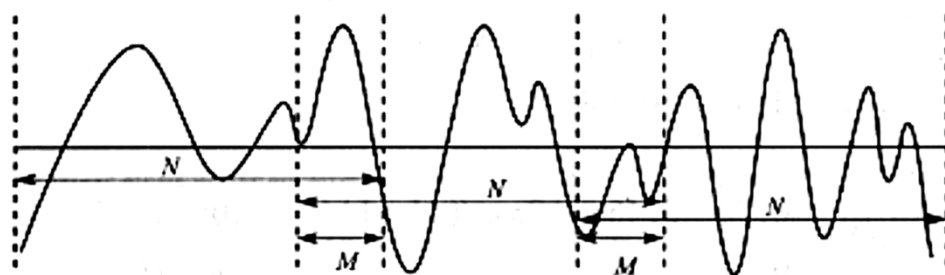
44



# 帧和加窗的概念

45

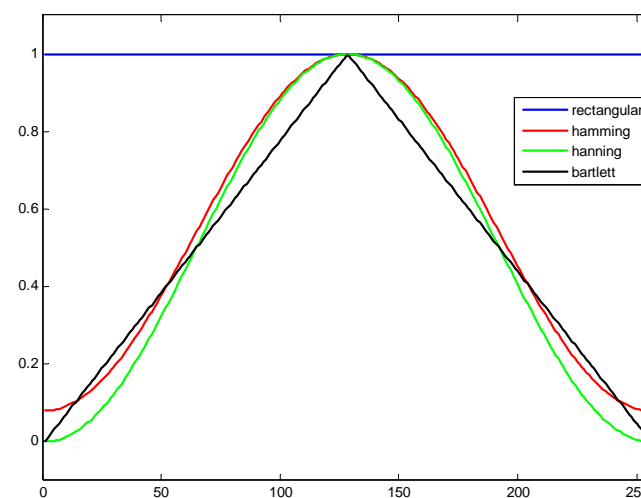
- 短时分析将语音流分为一段一段来处理，每一段称为一“帧”；
  - ▣ 帧长：10~30ms；
  - 帧移：0~1倍帧长，帧与帧之间的平滑过渡；
- 语音识别中常用的帧长为20~30ms,帧移为10ms
- 为了减小语音帧的截断效应，需要加窗处理；



a)  $N$ 为帧长,  $M$ 为帧间重叠长度



b) 帧长和帧移的示例





# 窗选择

46

- 不同的窗选择，将决定短时语音分析结果的好坏
  - 首先是窗的长度，长度 $N$ 将起决定性的作用。
    - $N$ 选得太大，不能保证每一帧的语音的平稳特性
    - $N$ 太小，不能保证信号的统计特性，容易产生统计噪声
    - 对于频域分析而言，窗长 $N$ 还直接决定了信号频谱的分辨率
  - 其次是窗口的形状，不同的窗，其频率特性是不一样的，这在短时频域分析时尤为重要。
- 窗口的形状和长度对分析影响很大，不同的分析方法对窗函数的要求不尽一样。



# 量测标准

47

- PSNR是“Peak Signal to Noise Ratio”的缩写
  - ▣ 峰值信噪比经常用作图像压缩等领域中信号重建质量的测量方法
  - ▣ 主要是利用影像信号的最大值与影像中噪声的比值作为评估的标准。
  - ▣ 通常PSNR值越高表示质量越好，一般而言当PSNR<30db时，代表以人的肉眼看起来是不能容忍的范围。
  - ▣ 因此大部分PSNR值皆要>30db。但PSNR高，并不代表影像质量一定好，有时候还是必须靠人的肉眼辅助来判断影像的质量才较为正确。

$$\text{PSNR} = 10 \times \log \left( \frac{255^2}{\text{MSE}} \right) \quad \text{MSE} = \frac{\sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2}{m \cdot n}$$



# 统计冗余与视听冗余

48

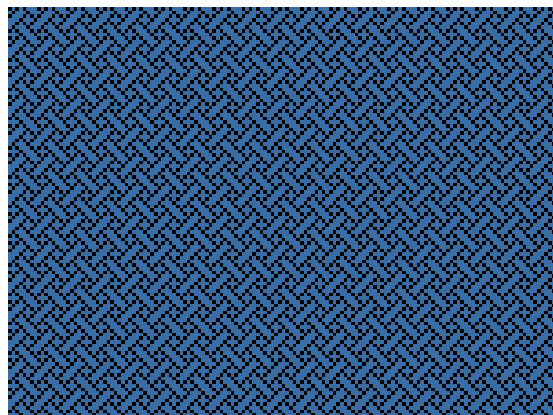
- 图像数据存在大量的统计特征的重复，这种重复包括静态单帧图像数据在空间上的冗余和音频、视频数据在时间上的冗余。
- 由于人的视觉系统和听觉系统的局限性，在图像数据和声音数据中，有些数据确实是多余的，使用算法将其去掉后并不会丢失实质性的信息或含义，对理解数据表达的信息几乎没有影响。



# 结构冗余与知识冗余

49

- 图像从大面积上或整体上看存在着重复出现的相同或相近的纹理结构，例如布纹图像和草席图像，**被称为结构冗余**。
- 图像的理解与图像所表现内容的基础知识有相当大的相关性，从这种知识出发可以归纳出图像的某种规律性变化，**这类冗余称为知识冗余**。比如，鼻子上方有眼睛，鼻子在嘴的上方等。





# 编码方法

50

- 统计编码
  - 香农-范诺编码
  - 霍夫曼编码
  - 算术编码
- 行程长度编码(Run-Length Coding)
- 词典编码(dictionary coding)
  - LZ77算法
  - LZ78算法
  - LZW算法



# 香农-范诺编码

51

## □ 程序

- (1) 计算该数据可能获得的压缩比的理论值
- (2) 对5个符号进行编码
- (3) 计算该数据可能获得的压缩比的实际值

## □ 举例

- 有一幅40个像素组成的灰度图像，灰度共有5级，分别用符号A，B，C，D和E表示。40个像素中出现灰度A的像素数有15个，出现灰度B的像素数有7个，出现灰度C的像素数有7个，出现灰度D的像素数有6个，出现灰度E的像素数有5个

符号	A	B	C	D	E
出现的次数	15	7	7	6	5
出现的概率	15/40	7/40	7/40	6/40	5/40



# 香农-范诺编码

52

## □ 压缩比的理论值计算

- 按照常规方法，表示5个符号最少需要3位，如用000表示A，001表示B，...，100表示E，其余3个代码(101，110，111)不用。
- 这就意味每个像素用3位，**编码这幅图像总共需要120位**。按照香农理论，这幅图像的熵为

$$\begin{aligned} H(X) &= -\sum_{i=1}^5 p(x_i) \log_2 p(x_i) \\ &= (15/40) \log_2 (40/15) + (7/40) \log_2 (40/7) + \dots + (5/40) \log_2 (40/5) \\ &\approx 2.196 \end{aligned}$$

- 这个数值表明，**每个符号不需要用3位构成的代码表示**，而用2.196位就可以，因此40个像素只需用87.84位就可以。
- 因此在理论上，**这幅图像的的压缩比为120:87.84≈1.37**



# 香农-范诺编码

53

## □ 符号编码

- 对每个符号进行编码时采用“从上到下”的方法。
- 首先按照符号出现的频度或概率排序，如A，B，C，D和E。
- 然后使用递归方法分成两个部分，每一部分具有近似相同的次数。

符号	出现的次数( $p(x_i)$ )	$-\log_2 p(x_i)$	分配的代码	需要的位数
A	15(0.375)	1.4150	00	30
B	7(0.175)	2.5145	01	14
C	7(0.175)	2.5145	10	14
D	6(0.150)	2.7369	110	18
E	5(0.125)	3.0000	111	15

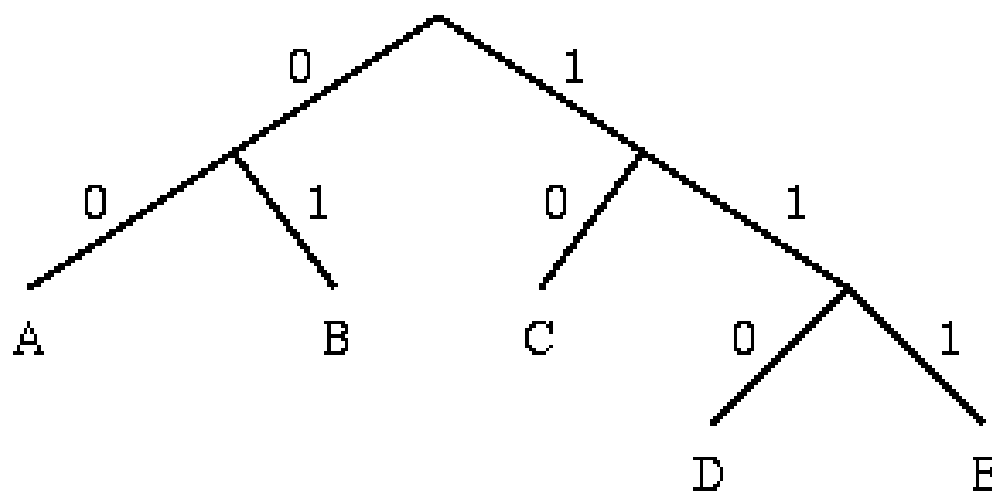


# 香农-范诺编码

54

## □ 压缩比的实际值

- 按照这种方法进行编码需要的总位数为  $30+14+14+18+15=91$ ，实际的压缩比为  $120:91 \approx 1.32$





# 编码方法

55

- 统计编码
  - 香农-范诺编码
  - 霍夫曼编码
  - 算术编码
- 行程长度编码(Run-Length Coding)
- 词典编码(dictionary coding)
  - LZ77算法
  - LZ78算法
  - LZW算法



# 霍夫曼编码

56

- 霍夫曼(D.A. Huffman)在1952年提出和描述的“从下到上”的熵编码方法
- 根据给定数据集中各元素所出现的频率来压缩数据的一种统计压缩编码方法。这些元素(如字母)出现的次数越多，其编码的位数就越少
- 广泛用在JPEG, MPEG, H.26X等各种信息编码标准中
- 程序
  - ▣ (1)计算该字符串的霍夫曼码
  - ▣ (2)计算该字符串的信息量与熵
  - ▣ (3)计算该字符串的平均码长
  - ▣ (4)计算编码前后的压缩比





# 霍夫曼编码

57

## □ 霍夫曼编码举例

■ 现有一个由5个不同符号组成的30个符号的字符串：

**BABACACADADABBCBABEBEDDABEEEBB**

符号	出现的次数( $p(x_i)$ )	$-\log_2 p(x_i)$	分配的代码	需要的位数
A	10	1.585		
B	8	1.907		
C	3	3.322		
D	4	2.907		
E	5	2.585		



# 霍夫曼编码

58

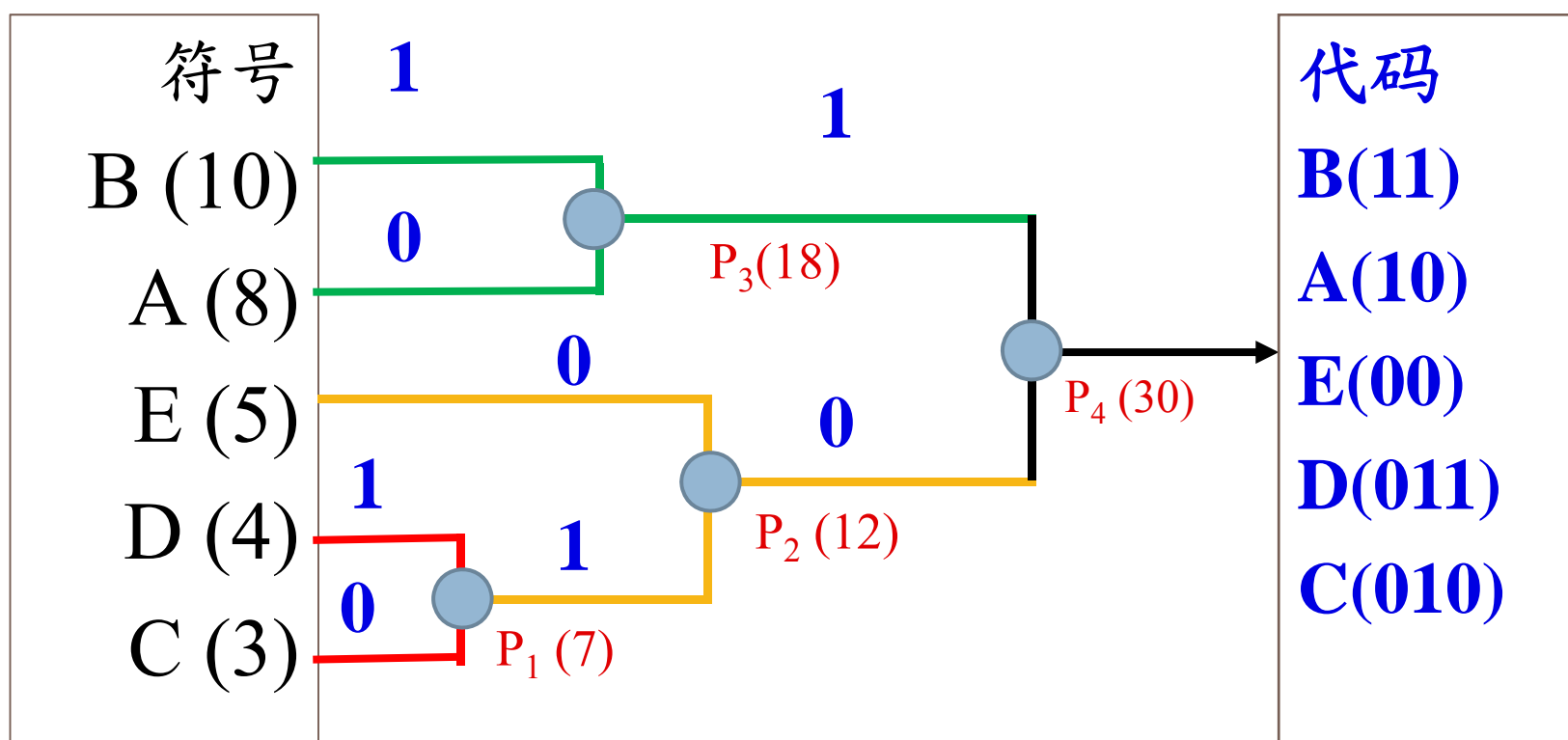
## □ 计算该字符串的霍夫曼码

- 1) 按照符号出现概率大小的顺序对符号进行排序
- 2) 把概率最小的两个符号组成一个节点P1
- 3) 重复步骤2，得到节点P2，P3，P4，.....，PN，形成一棵树，其中的PN称为根节点
- 4) 从根节点PN开始到每个符号的树叶，从上到下标上0(上枝)和1(下枝)，至于哪个为1哪个为0则无关紧要，但通常把概率大的标成1，概率小的标成0
- 5) 从根节点PN开始顺着树枝到每个叶子分别写出每个符号的代码



# 霍夫曼树

59





# 霍夫曼编码

60

符号	出现的次数( $p(x_i)$ )	$-\log_2 p(x_i)$	分配的代码	需要的位数
A	10	1.585	11	20
B	8	1.907	10	16
C	3	3.322	010	9
D	4	2.907	011	12
E	5	2.585	00	10

30个字符组成的字符串需要67位



# 霍夫曼编码

61

## □ 计算该字符串的熵

$$\begin{aligned} \square \quad H(S) &= (8/30) \times \log_2(30/8) + (10/30) \times \log_2(30/10) + \\ &\quad (3/30) \times \log_2(30/3) + (4/30) \times \log_2(30/4) + (5/30) \times \log_2(30/5) \\ &= (44.3136 - 24.5592) / 9.0308 = 2.1874 \quad (\text{sh}) \end{aligned}$$

## □ 计算该字符串的平均码长

$$\square \quad \text{平均码长：} \quad l = \sum_{i=1}^N l_i p(l_i)$$

$$\begin{aligned} &= (2 \times 8 + 2 \times 10 + 3 \times 3 + 3 \times 4 + 2 \times 5) / 30 \\ &= 2.233 \text{ 位/符号} \end{aligned}$$



# 霍夫曼编码

62

- 计算编码前后的压缩比
  - ▣ 编码前：5个符号需3位，30个字符，需要90位
  - ▣ 编码后：共67位
  - ▣ 压缩比： $90/67=1.34$



# 编码方法

63

- 统计编码
  - 香农-范诺编码
  - 霍夫曼编码
  - 算术编码
- 行程长度编码(Run-Length Coding)
- 词典编码(dictionary coding)
  - LZ77算法
  - LZ78算法
  - LZW算法

# 算术编码(arithmetic coding)



64

- Arithmetic coding是一个比较新的编码方法，通常效能会比Huffman coding来的好。
- Huffman coding给每一个符号一个编码，而**这个编码是整数的位长度**。而Arithmetic coding可以**将一整个讯息当作一个编码单元**。
- 一个讯息有半开区间 $[a, b)$ 来代表，而 $a$ 跟 $b$ 是介于0与1之间的实数。一开始的区间是 $[0, 1)$ 。当这个讯息变得越来越长时，区间会越来越短，而且需要越多的位来代表这个区间。





# 算术编码

65

## □ 范例

- ▣ 假设信源符号为  $\{00, 01, 10, 11\}$ ，它们的概率分别为  $\{0.1, 0.4, 0.2, 0.3\}$
- ▣ 对二进制消息序列  $10\ 00\ 11\ 00\ 10\ 11\ 01\ \dots$  进行算术编码



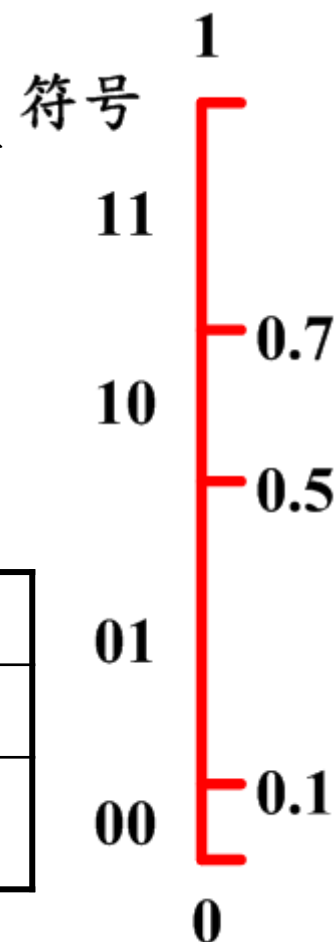
# 算术编码

66

## □ 初始化

- 根据信源符号的概率把间隔 $[0, 1)$ 分成的4个子间隔： $[0, 0.1)$ ,  $[0.1, 0.5)$ ,  $[0.5, 0.7)$ ,  $[0.7, 1)$ 。
- 其中 $[a, b)$ 的表示半开放间隔，即包含 $a$ 不包含 $b$ ， $a$ 称为低边界或左边界， $b$ 称为高边界或右边界

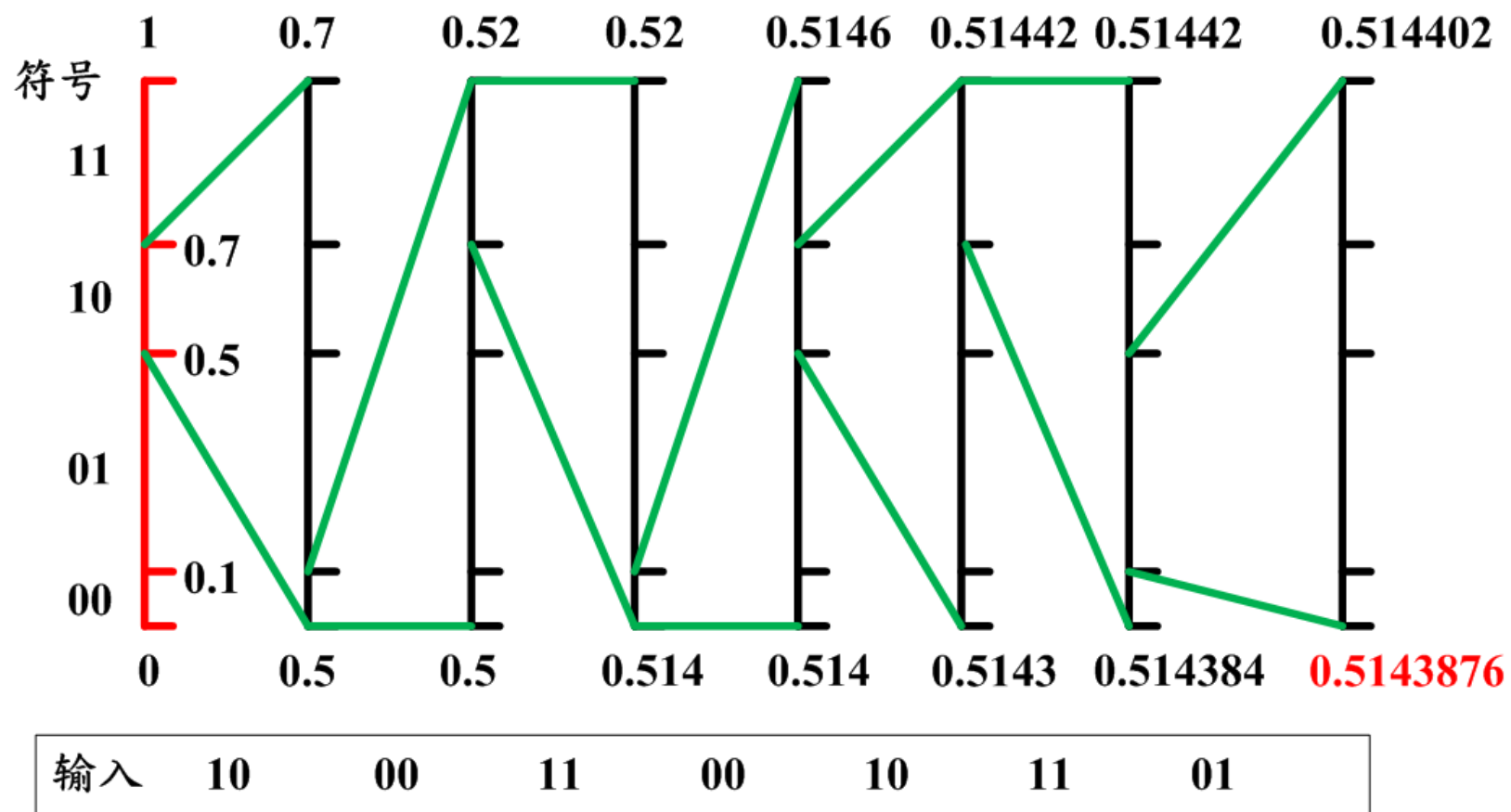
符号	00	01	10	11
概率	0.1	0.4	0.2	0.3
初始编码间隔	$[0, 0.1)$	$[0.1, 0.5)$	$[0.5, 0.7)$	$[0.7, 1]$





# 算术编码

67



# 算术编码



68

Step	Input	间隔选择	编码判决
1	10	$[0.5, 0.7)$	选取符号的间隔范围 $[0.5, 0.7)$
2	00	$[0.5, 0.52)$	选取 $[0.5, 0.7)$ 间隔的第一个子间隔
3	11	$[0.514, 0.52)$	选取 $[0.5, 0.52)$ 间隔的第四个子间隔
4	00	$[0.514, 0.5146)$	选取 $[0.514, 0.52)$ 间隔的第一个子间隔
5	10	$[0.5143, 0.51442)$	选取 $[0.514, 0.5146)$ 间隔的第三个子间隔
6	11	$[0.514384, 0.51442)$	选取 $[0.5143, 0.51442)$ 间隔的第四个子间隔
7	01	$[0.5143876, 0.514402)$	选取 $[0.514384, 0.51442)$ 间隔的第二个子间隔
8	从 $[0.5143876, 0.514402]$ 中选择一个数( $x$ , 0.5143876)作为输出		

# 算术编码算法



69

```
BEGIN
    low = 0.0;    high = 1.0;    range = 1.0;

    while (symbol != terminator)
    {
        get (symbol);
        low = low + range * Range_low(symbol);
        high = low + range * Range_high(symbol);
        range = high - low;
    }

    output a code so that low <= code < high;
END
```



# 算术编码-译码过程

70

输入 0.5143876

Step	间隔选择	输出	编码判决
1	$[0.5, 0.7)$	10	在间隔范围 $[0.5, 0.7)$
2	$[0.5, 0.52)$	00	在 $[0.5, 0.7)$ 间隔的第一个子间隔
3	$[0.514, 0.52)$	11	在 $[0.5, 0.52)$ 间隔的第四个子间隔
4	$[0.514, 0.5146)$	00	在 $[0.514, 0.52)$ 间隔的第一个子间隔
5	$[0.5143, 0.51442)$	10	在 $[0.514, 0.5146)$ 间隔的第三个子间隔
6	$[0.514384, 0.51442)$	11	在 $[0.5143, 0.51442)$ 间隔的第四个子间隔
7	$[0.5143876, 0.514402)$	01	在 $[0.514384, 0.51442)$ 间隔的第二个子间隔
8	$[0.5143876, 0.514402)$ 的尾数与输入相同，停止		



# 编码方法

71

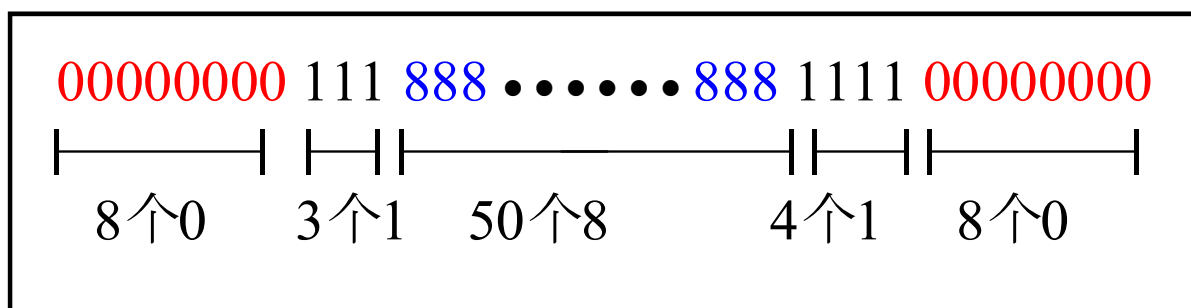
- 统计编码
  - ▣ 香农-范诺编码
  - ▣ 霍夫曼编码
  - ▣ 算术编码
- 行程长度编码(Run-Length Coding)
- 词典编码(dictionary coding)
  - ▣ LZ77算法
  - ▣ LZ78算法
  - ▣ LZW算法



# 行程长度编码

72

- 一种无损压缩数据编码技术，它利用重复的数据单元有相同的数值这一特点对数据进行压缩。对相同的数值只编码一次，同时计算出相同值重复出现的次数。在JPEG，MPEG，H.261和H.263等压缩方法中，RLE用来对图像数据变换和量化后的系数进行编码
- 例: 假设有一幅灰度图像第n行的像素值。用RLE编码方法得到的代码为：80315084180







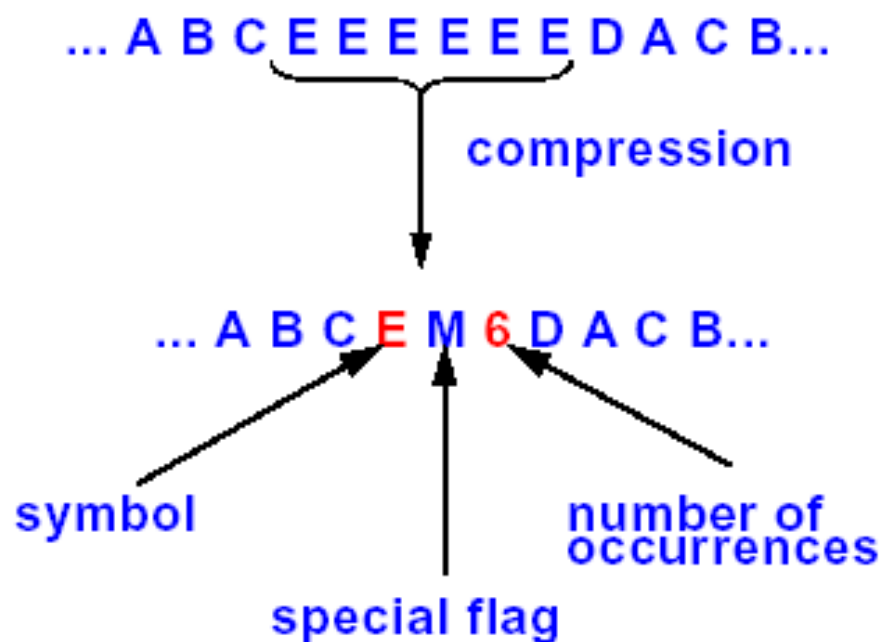
# 行程长度编码

73

□ Assumption:

▣ Long sequences of identical symbols.

▣ Example,





# 编码方法

74

- 统计编码
  - 香农-范诺编码
  - 霍夫曼编码
  - 算术编码
- 行程长度编码(Run-Length Coding)
- 词典编码(dictionary coding)
  - LZ77算法
  - LZ78算法
  - LZW算法

# 词典编码(dictionary coding)



75

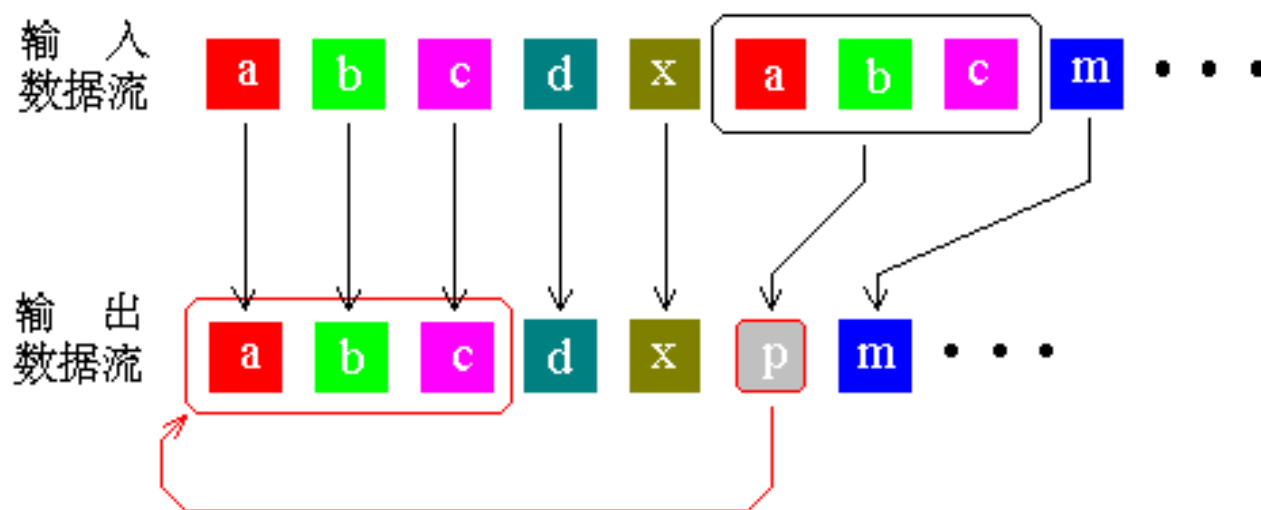
- 采用静态词典编码技术时，编码器需要事先构造词典，解码器要事先知道词典。
- 采用动态词典编码技术时，编码器将从被压缩的文本中自动导出词典，解码器解码时边解码边构造解码词典，具体算法
  - ▣ LZ77算法
  - ▣ LZW算法



# 词典编码技术

76

- 用已经出现过的字符串替代重复的部分
- 编码器的输出仅仅是指向早期出现过的字符串的“指针”





# LZ77算法

77

- LZ77和LZ78算法，是以Abraham Lempel和Jakob Ziv分在1977年和1978年开发和发表
  - ▣ Jacob Ziv, Abraham Lempel, A Universal Algorithm for Sequential Data Compression, IEEE Transactions on Information Theory, 23(3):337-343, May 1977.
- LZ77 算法在某种意义上又可以称为“滑动窗口压缩”，该算法将一个虚拟的，可以跟随压缩进程滑动的窗口作为词典，要压缩的字符串如果在该窗口中出现，则输出其出现位置和长度。



# LZ77算法

78

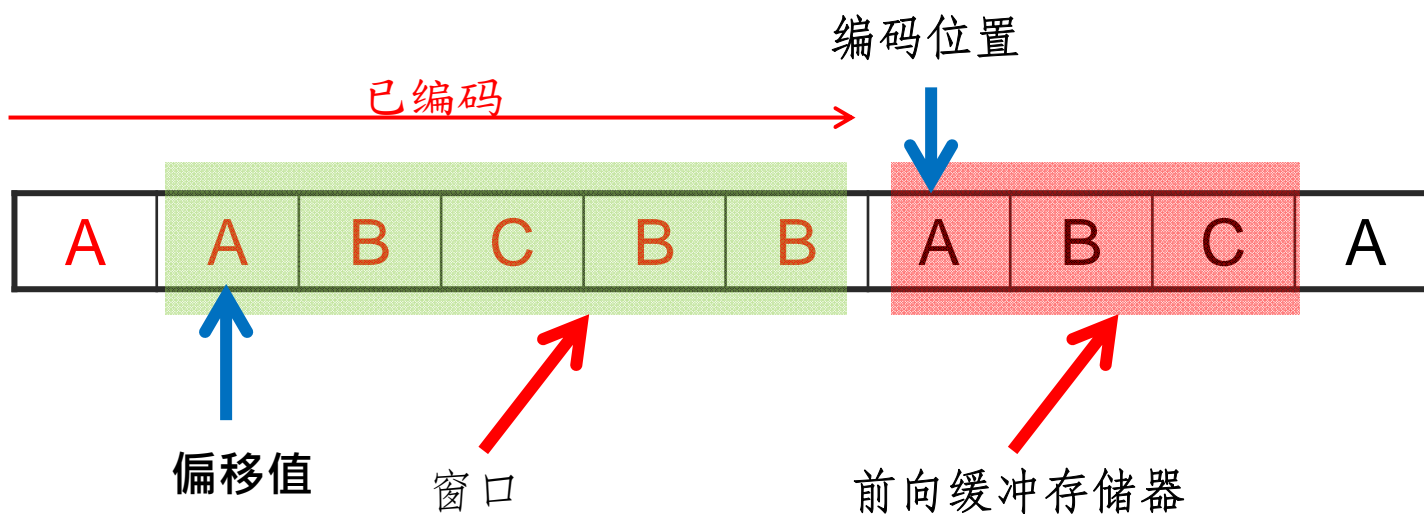
- 输入数据流(input stream)：要被压缩的字符序列。
- 字符(character)：输入数据流中的基本单元。
- 编码位置(coding position)：输入数据流中当前要编码的字符位置，指前向缓冲存储器中的开始字符。
- 前向缓冲存储器(Lookahead buffer)：存放从编码位置到输入数据流结束的字符序列的存储器。
- 窗口(window)：指包含W个字符的窗口，字符从编码位置开始向后数，也就是最后处理的字符数。
- 偏移值(offset)：为窗口中匹配字符串相对窗口边界的偏移



# LZ77算法-范例

79

- 以编码AABCBBABCA为例
- 前向缓冲存储器=3
- 窗口=5





# LZ77算法-編碼范例

80

A	A	B	C	B	B	A	B	C	A
---	---	---	---	---	---	---	---	---	---

步骤	字符输入	缓冲存储器	窗口	匹配串	边界的偏移值	输出 (ptr, len, ch)
1	A	A	(-, -, -, -, -)	0	0	(0, 0, A)
2	A	A	(-, -, -, -, A)	A	1	-
3	B	AB	(-, -, -, -, A)	A(0)	1(0)	(1, 1, B)
4	C	C	(-, -, A, A, B)	0	0	(0, 0, C)
5	B	B	(-, A, A, B, C)	B	2	-
6	B	BB	(-, A, A, B, C)	B(0)	2(0)	(2, 1, B)
7	A	A	(A, B, C, B, B)	A	5	-
8	B	AB	(A, B, C, B, B)	A B	5	-
9	C	ABC	(A, B, C, B, B)	ABC	5	-
10	A	ABC	(A, B, C, B, B)	ABC(0)	5(0)	(5, 3, A)





# 编码方法

81

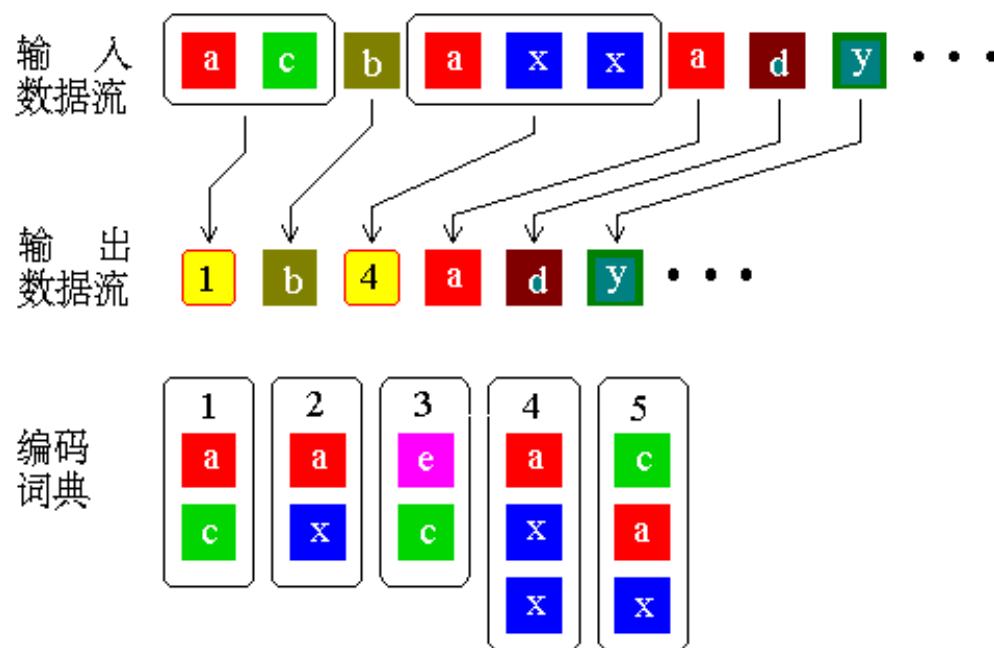
- 统计编码
  - 香农-范诺编码
  - 霍夫曼编码
  - 算术编码
- 行程长度编码(Run-Length Coding)
- 词典编码(dictionary coding)
  - LZ77算法
  - **LZ78算法**
  - LZW算法



# 词典编码技术(LZ78)

82

- 从输入的数据中创建一个“短语词典(dictionary of the phrases)”
- 编码器输出词典中的短语“索引号”，而不是短语





# LZ78原理

83

- LZ78的编码思想是不断地从字符流中提取新的字符串(String)，通俗地理解为新“词条”，然后用“代号”也就是码字(Code word)表示这个“词条”。
- 这样一来，对字符流的编码就变成了用码字(Code word)去替换字符流(Char stream)，生成码字流(Code stream)，从而达到压缩数据的目的。
- LZ77与LZ78比较
  - ▣ LZ77利用滑动窗口里的内容作为字典，找最长子串
  - ▣ LZ78动态构建字典



# LZ78编码算法

84

- 步骤1：在开始时，词典和当前前缀P都是空的。
- 步骤2：当前字符C：字符流中的下一个字符。
- 步骤3：判断P+C是否在词典中：
  - ▣ (1) 如果“是”： $P = P + C$ ；
  - ▣ (2) 如果“否”：
    - 输出与当前前缀P相对应的码字和当前字符C；
    - 字符串P+C添加到词典中。
    - $P = \text{空值}$ 。
  - ▣ (3) 判断字符流中是否还有字符需要编码
    - 如果“是”：返回到步骤2。
    - 如果“否”：若当前前缀P不是空的，输出相应于当前前缀P的码字，然后结束编码。



# LZ78算法-編碼范例

85

A	B	B	C	B	C	A	B	A
---	---	---	---	---	---	---	---	---

步骤	字符 (c)	前缀P (执行前)	前缀P (执行后)	添加词典	词典编号 (ref)	输出 (ref, c)
1	A	0	0	A	1	(0, A)
2	B	0	0	B	2	(0, B)
3	B	0	B	-	-	-
4	C	B	0	BC	3	(2, C)
5	B	0	B	-	-	-
6	C	B	BC	-	-	-
7	A	BC	0	BCA	4	(3, A)
8	B	0	B	-	-	-
9	A	B	0	BA	5	(2, A)



# LZ78译码算法

86

- 步骤1：在开始时词典是空的。
- 步骤2：当前码字 $W$ 码字流中的下一个码字。
- 步骤3：当前字符 $C$ 随码字之后的字符。
- 步骤4：把当前码字的缀-字符串(string.W)输出到字符流(Charstream)，然后输出字符 $C$ 。
- 步骤5：把string.W+C添加到词典中。
- 步骤6：判断码字流中是否还有码字要译
  - 如果“是”，就返回到步骤2。
  - 如果“否”，则结束。



# 编码方法

87

- 统计编码
  - ▣ 香农-范诺编码
  - ▣ 霍夫曼编码
  - ▣ 算术编码
- 行程长度编码(Run-Length Coding)
- 词典编码(dictionary coding)
  - ▣ LZ77算法
  - ▣ LZ78算法
  - ▣ **LZW算法**



# LZW编码算法

88

- 步骤1：在开始时，词典内已包含字符流出现中的所有单个字符，字符串S是字符流第一个字符。
- 步骤2：当前字符C是字符流中的下一个字符。
- 步骤3：判断S+C是否在词典中：
  - ▣ (1) 如果“是”： $S = S + C$ ；
  - ▣ (2) 如果“否”：
    - 输出当前词典编号；
    - 字符串S+C添加到词典中，并给与编号。
    - $S = C$ 。
  - ▣ (3) 判断字符流中是否还有字符需要编码
    - 如果“是”：返回到步骤2。
    - 如果“否”：若当前字符串S不是空的，输出相应于当前字符串S的词典编号，然后结束编码。





# LZW算法-編碼范例

89

A	B	A	B	B	A	B	C	A	B	A	B	B	A
---	---	---	---	---	---	---	---	---	---	---	---	---	---

步骤	字符串S	字符 C	添加字典	字典編號(ref)	输出 (ref)
0	-	-	A	1	-
0	-	-	B	2	-
0	-	-	C	3	-
1	A	B	AB	4	1
2	B	A	BA	5	2
3	A	B	-	-	-
4	AB	B	ABB	6	4
5	B	A	-	-	-
6	BA	B	BAB	7	5
7	B	C	BC	8	2
8	C	A	CA	9	3
9	A	B	-	-	-
10	AB	A	ABA	10	4
11	A	B	-	-	-
12	AB	B	-	-	-
13	ABB	A	ABBA	11	6
14	A	EOF	-	-	1



# LZW译码算法

90

- 步骤1：在开始时词典内已包含字符流出现中的所有单个字符。
- 步骤2：当前码字W码字流中的下一个码字。
- 步骤3：当前字符C随码字之后的字符。
- 步骤4：把当前码字的字符串(string.W)输出到字符流(Charstream)，然后输出字符C。
- 步骤5：把string.W+C添加到词典中。
- 步骤6：判断码字流中是否还有码字要译
  - ▣ 如果“是”，就返回到步骤2。
  - ▣ 如果“否”，则结束。



# LZW 与 LZ77

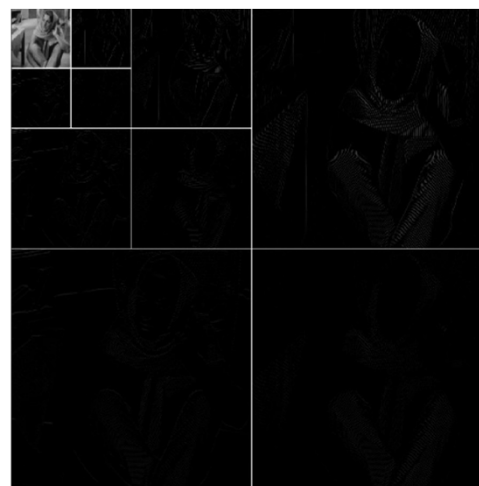
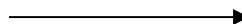
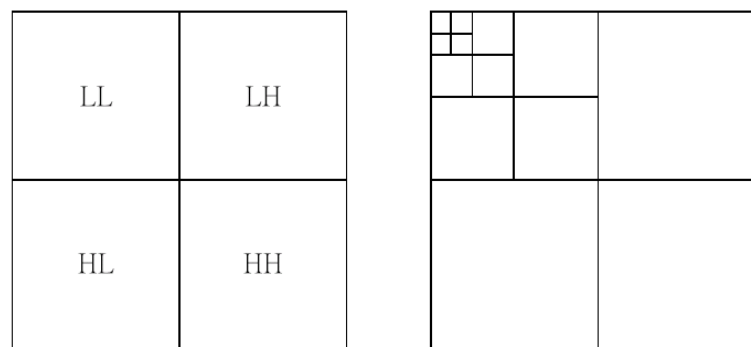
91

- 在多数情况下，**lz77 拥有更高的压缩率**，而在待压缩文件中占绝大多数的是些超长匹配，并且相同的**超长匹配高频率地反复出现时，lzw 更具优势**。
- **GIF 就是采用了 lzw 算法来压缩背景单一、图形简单的图片**。
- **zip 是用来压缩通用文件的**，这就是它采用对大多数文件有更高压缩率的 **lz77 算法的原因**。

# 使用小波转换在二维图像中



92





# 应用傅立叶变换需注意的问题

93

- 傅立叶变换需要**计算复数**，而不是实数。
- 傅立叶变换的**谱收敛性差**，不利于图像编码的应用。
  - ▣ 主要原因在于，**图像左右以及顶底之间的截然不连续性**，结果产生许多大幅度、高空间频率的分量。
- 傅立叶变换研究的是**信号总体的频谱分布规律**，**是一种纯频率域的分析方法**，无法给出信号在某一段时间其频率分布规律。



# DWT的好处

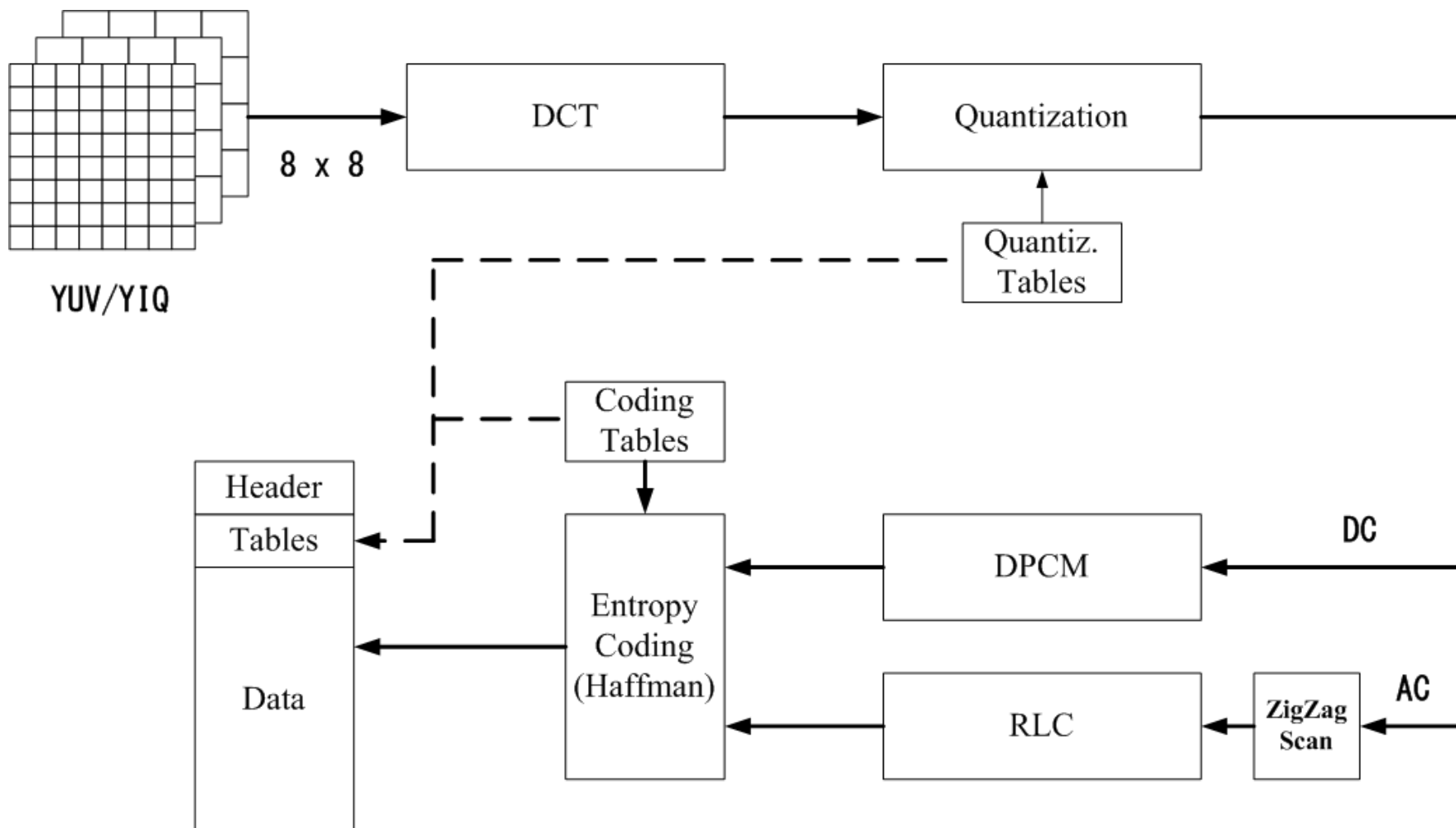
94

- DWT会有较高的压缩率，因为可以避免 blocking artifacts.
- DWT has good **localization** both in time and spatial frequency domain.
- DWT容易适应人眼的感知，有助于提高压缩率
- DWT灵活性高
  - ▣ Wavelet function(Basic function) can be freely chosen



# JPEG Encoding Overview

95



# JPEG Encoding Overview



96

- The main steps in JPEG encoding are the following
  - ▣ Transform RGB to YUV or YIQ
  - ▣ Chroma subsampling color (4:2:0)
  - ▣ DCT on 8x8 image blocks
  - ▣ Quantization
  - ▣ Zig-zag ordering and run-length encoding
  - ▣ Entropy coding

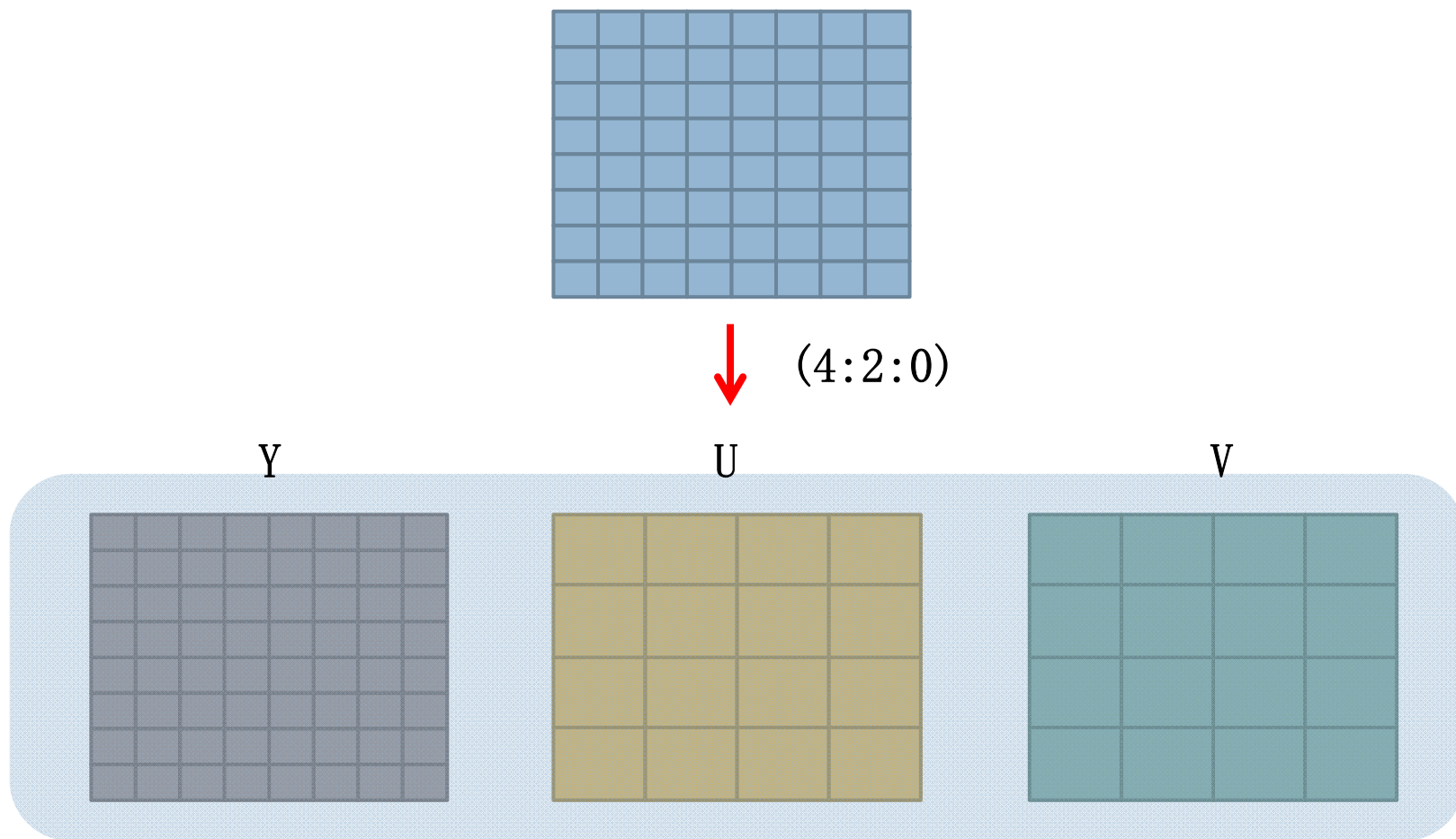


# YUV or YIQ

## Subsampling color (4:2:0)



97





# 8x8 2D DCT

98

200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	195	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

DCT

DC

AC

515	65	-12	4	1	2	-8	5
-16	3	2	0	0	-11	-2	3
-12	6	11	-1	3	0	1	-2
-8	3	-4	2	-2	-3	-5	-2
0	-2	7	-5	4	0	-1	-4
0	-3	-1	0	4	1	-1	0
3	-2	-3	3	3	-1	-1	3
2	5	-2	4	2	2	-3	0



# Quantization

99

- Quantization in JPEG aims at reducing the total number of bits in the compressed image
  - ▣ Divide each entry in the frequency space block by an integer, then round
  - ▣ Use a quantization matrix  $Q(u, v)$

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



# CCIR 601标准 Quantization

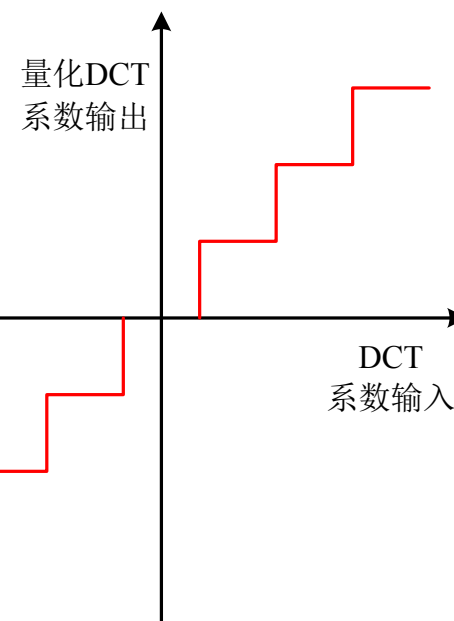
100

$$Q_Y(u, v) =$$

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

$$Q_{U,V}(u, v) =$$

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



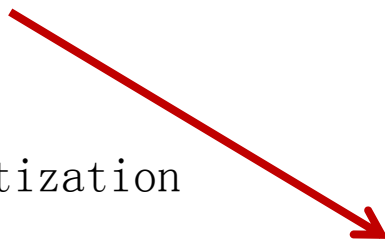
# 量化



101

515	65	-12	4	1	2	-8	5
-16	3	2	0	0	-11	-2	3
-12	6	11	-1	3	0	1	-2
-8	3	-4	2	-2	-3	-5	-2
0	-2	7	-5	4	0	-1	-4
0	-3	-1	0	4	1	-1	0
3	-2	-3	3	3	-1	-1	3
2	5	-2	4	2	2	-3	0

Quantization

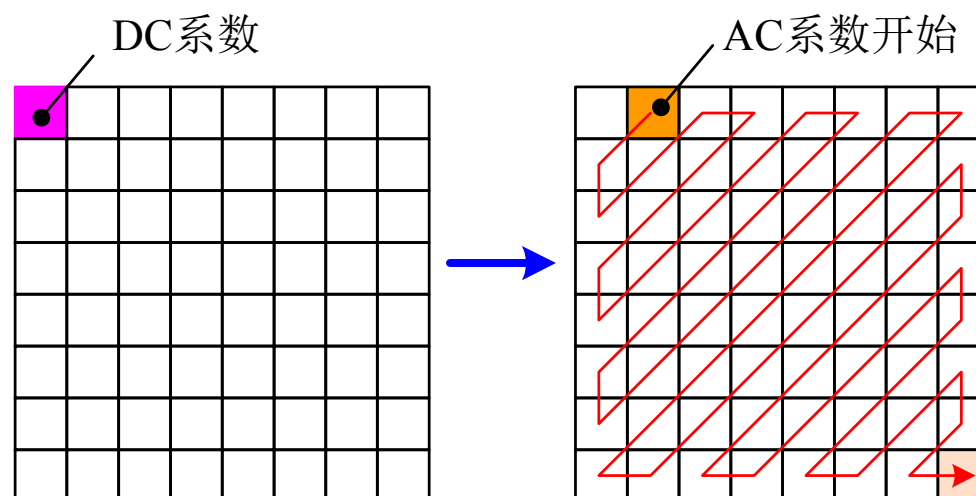


32	6	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

# JPEG压缩编码

102

- Z字形编排(ZigZag Scan)
  - ▣ 为增加连续的“0”值系数的个数，即“0”的游程长度
  - ▣ DCT系数的序号，序号小的位置表示频率较低，把一个 $8 \times 8$ 的矩阵变成一个 $1 \times 64$ 的矢量





# DC Entropy Coding

103

- 每个区块有一个DC值
- 假设有五个DC值，分别为150, 155, 149, 152, 144
- 经过DPCM后，则分别为150, 5, -6, 3, -8
- 将DPCM的值以一对符号表示(SIZE, AMPLITUDE)
  - ▣ SIZE = number of bits to represent coefficient
  - ▣ AMPLITUDE = the actual bits
  - ▣ 利用右表方式编码，则上述五个数值将分别表示为
  - ▣ (8, 10010110), (3, 101), (3, 001), (2, 11), (4, 0111)
- 再将数值编排

SIZE	AMPLITUDE
1	-1, 1
2	-3, -2, 2, 3
3	-7..-4, 4..7
4	-15..-8, 8..15
.	.
.	.
.	.
10	-1023..-512, 512..1023



# AC Entropy Coding

104

- 经由Z字形编排后
  - ▣ 假设数值(6, -1, -1, 0, -1, 0, 0, 0, -1, 0, 0, 1, 0, 0, ..., 0)
- 利用RLC(Run-Length Coding)方式编码记录
  - ▣ 将AC用Z字形编排的值以一对符号表示(RUNLENGTH, VALUE)
  - ▣ RUNLENGTH is the number of zeroes in the run
  - ▣ VALUE is the next non-zero value
- 因此这组数值为
  - ▣ (0,6) (0,-1) (0,-1)(1,-1)(3,-1)(2,1)(0,0)
- 然后将VALUE的数值利用符号表示(SIZE, AMPLITUDE)编排，如同DC值的做法，后排序。




# 反量化



105

32	6	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0
-1	0	1	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

Inverse Quantization



512	66	-10	0	0	0	0	0
-12	0	0	0	0	0	0	0
-14	0	16	0	0	0	0	0
-14	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

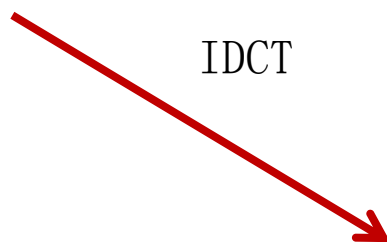


# 8x8 2D IDCT

106

512	66	-10	0	0	0	0	0
-12	0	0	0	0	0	0	0
-14	0	16	0	0	0	0	0
-14	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

IDCT



199	196	191	186	182	178	177	176
201	199	196	192	188	183	180	178
203	203	202	200	195	189	183	180
202	203	204	203	198	191	183	179
200	201	202	201	196	189	182	177
200	200	199	197	192	186	181	177
204	202	199	195	190	186	183	181
207	204	200	194	190	187	185	184

# 重建图与原始图并不相同(失真)



107

200	202	189	188	189	175	175	175	199	196	191	186	182	178	177	176
200	203	198	188	189	182	178	175	201	199	196	192	188	183	180	178
203	200	200	195	200	187	185	175	203	203	202	200	195	189	183	180
200	200	200	195	197	187	187	187	202	203	204	203	198	191	183	179
200	205	200	200	195	188	187	175	200	201	202	201	196	189	182	177
200	200	200	200	200	190	187	175	200	200	199	197	192	186	181	177
205	200	199	200	191	187	187	175	204	202	199	195	190	186	183	181
210	200	200	200	188	185	187	186	207	204	200	194	190	187	185	184

原始图



重建图

1	6	-2	2	7	-3	-2	-1
-1	4	2	-4	1	-1	-2	-3
0	-3	-2	-5	5	-2	2	-5
-2	-3	-4	-3	-1	-4	4	8
0	4	-2	-1	-1	-1	5	-2
0	0	1	3	8	4	6	-2
1	-2	0	5	1	1	4	-6
3	-4	0	6	-2	-2	2	2