第二章
媒体的特性与表示

主要内容

- **2.1** 图形与图像属性及常用文件格式
- **2.2** 视频的特性及常用文件格式
- **2.3** 音频的特性及常用文件格式

2.1 图形图像属性及文件格式

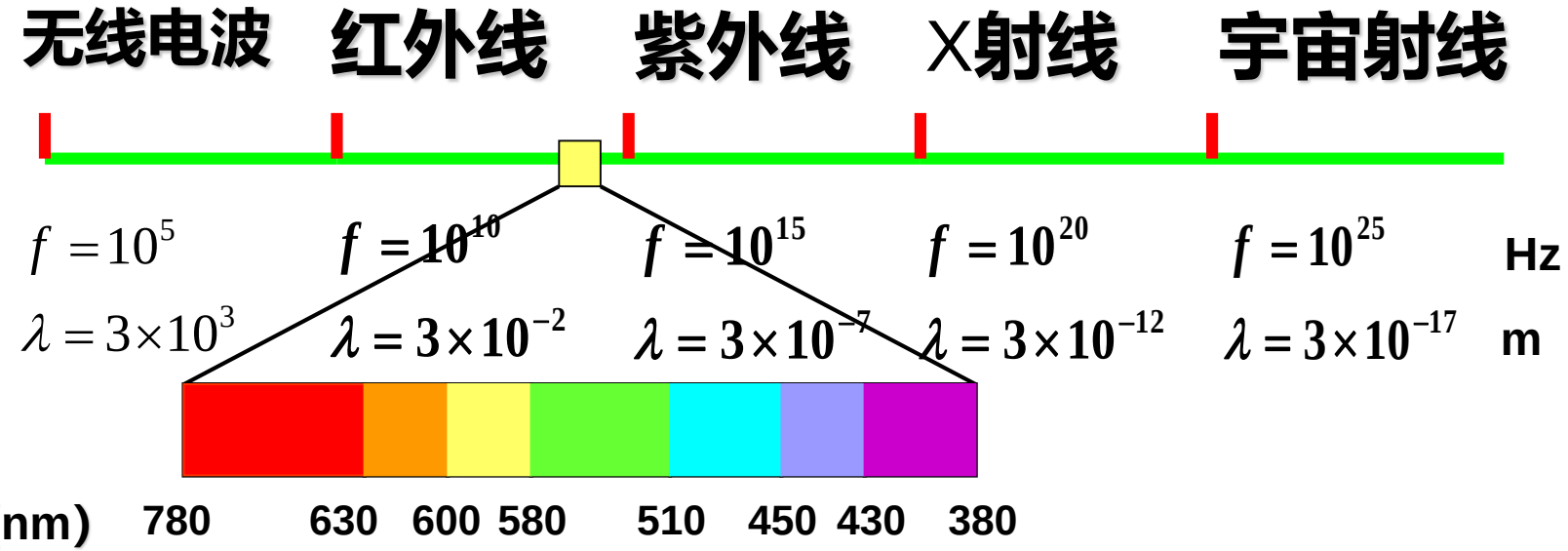
- 2.1.1 视觉系统对颜色的感知
- 2.1.2 图像的颜色模型（彩色空间color space）
- 2.1.3 图像的基本属性
- 2.1.4 图像的分类
- 2.1.5 常见的图像格式

2.1.1视觉系统对颜色的感知

- 彩色视觉
- 彩色三要素
- 三基色原理

2.1.1 视觉系统对颜色的感知

电磁波谱



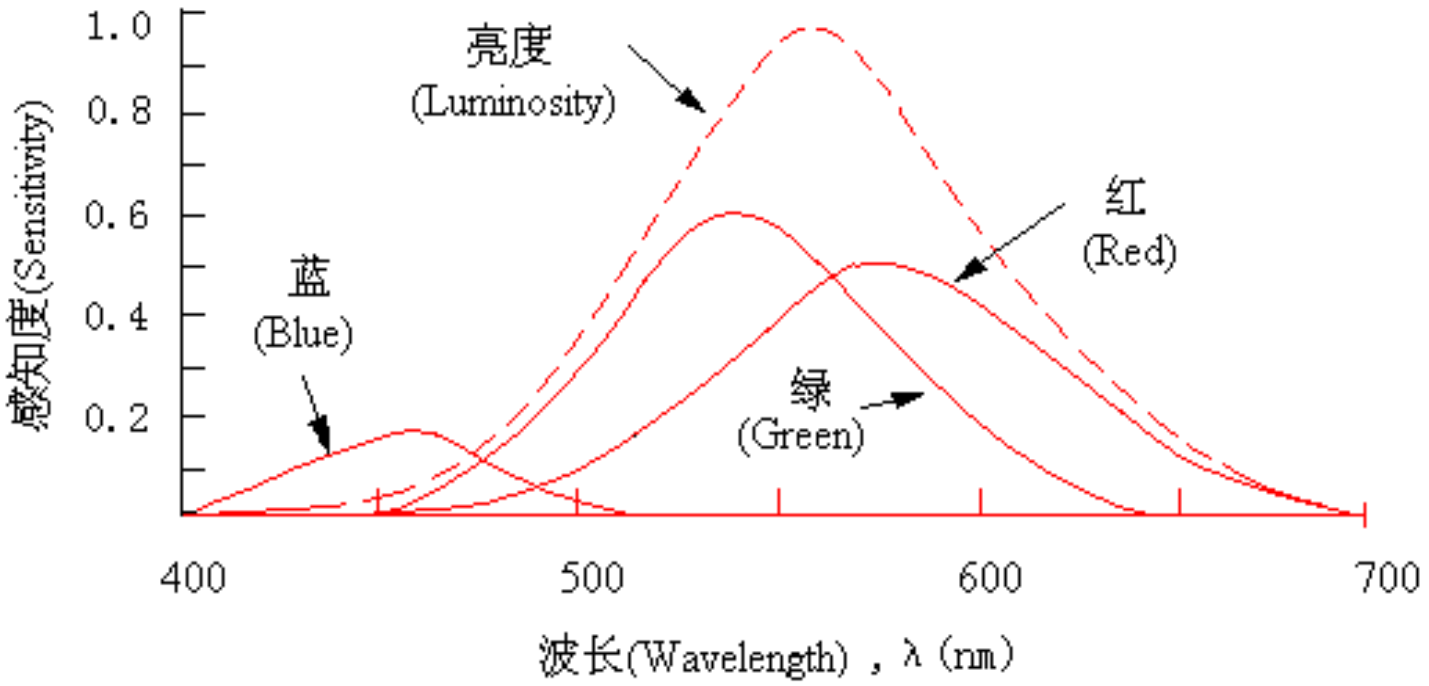
可见光

彩色视觉

光敏细胞
- 杆状细胞（暗视觉细胞）：感光灵敏度强，不能辨色
- 锥状细胞（明视觉细胞）：感光灵敏度弱，能辨色

- 人的视网膜有对红、绿、蓝颜色敏感程度不同的三种锥体细胞，另外还有一种在光功率极端低的条件下才起作用的杆状体细胞，因此颜色只存在于眼睛和大脑中。颜色是视觉系统对可见光的感知结果。
- 红、绿和蓝三种锥体细胞对不同频率的光的感知程度不同，对不同亮度的感知程度也不同，因此不同组成成分的可见光就呈现出不同的颜色

2.1.1 视觉系统对颜色的感知

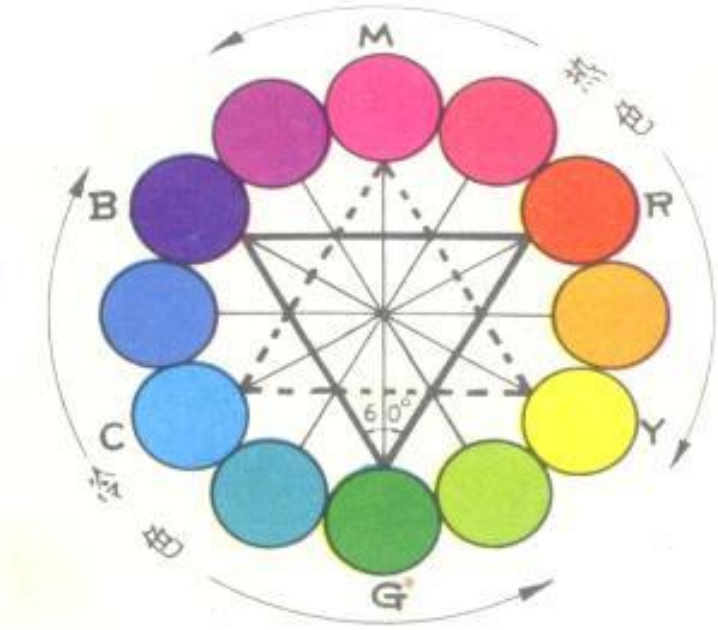


视觉系统对颜色和亮度的响应特性曲线
（各个波长的光的强度相等）

彩色三要素

从人的主观感觉角度，颜色包含三个要素：

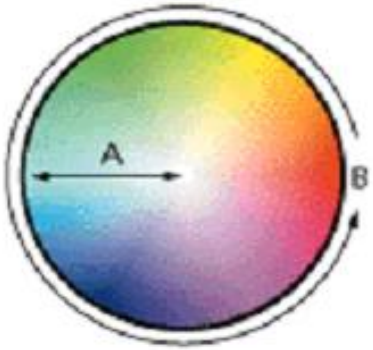
1. 色调（hue）：色调反映颜色的类别，如红色、绿色、蓝色等。色调大致对应光谱分布中的主波长。



彩色三要素

2. 饱和度（Saturation）

饱和度是指彩色光所呈现颜色的深浅或纯洁程度。对于同一色调的彩色光，其饱和度越高，颜色就越深，或越纯；而饱和度越小，颜色就越浅，或纯度越低。高饱和度的彩色光可因掺入白光而降低纯度或变浅，变成低饱和度的色光。**100%**饱和度的色光就代表完全没有混入白光的纯色光。

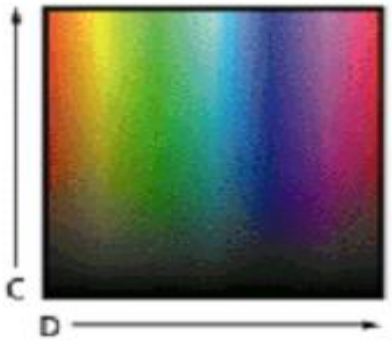


彩色三要素

3. 明亮度（luminance）

明亮度是光作用于人眼时引起的明亮程度的感觉。一般来说，彩色光能量大则显得亮，反之则暗。

大量试验表明，人的眼睛能分辨128种不同的色调，10－30种不同的饱和度，而对亮度非常敏感。人眼大约可以分辨35万种颜色。



三基色原理

- 自然界中绝大多数彩色都可以由三基色按一定比例混合而得；反之，这些彩色也可以分解成三基色；
- 三基色必须是相互独立的，即其中任何一种基色都不能由其它两种基色混合得到；
- 混合色的色调和饱和度由三基色的混合比例决定；
- 混合色的亮度是三基色亮度之和。

电视系统的三基色为：红、绿、蓝

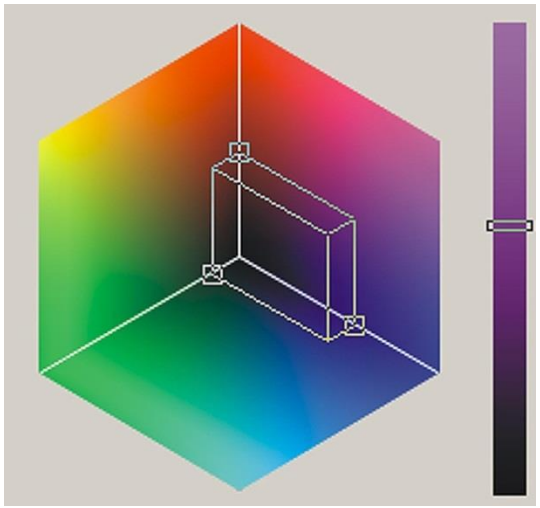
补色：若A + B = 白色，则A和B互为补色

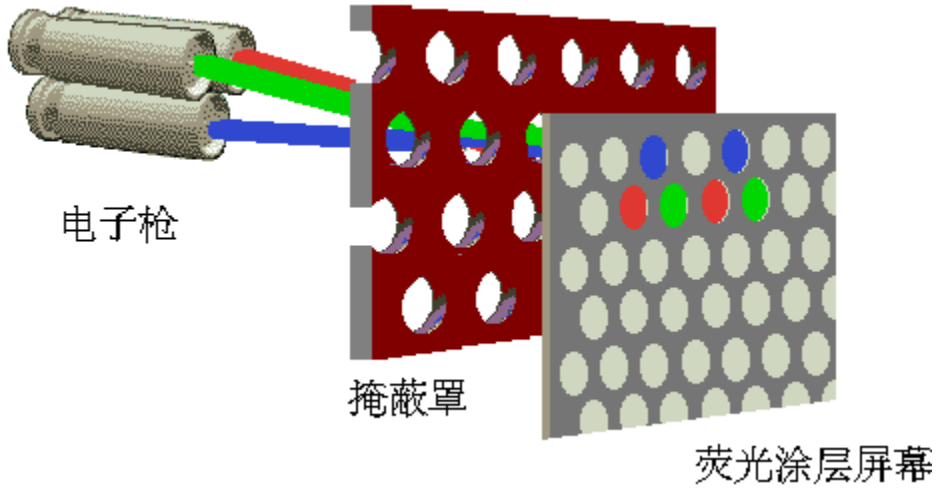
2.1.2 颜色模型

- 颜色模型（**color model**）是用来精确标定和生成各种颜色的一套规则和定义。某种颜色模型所标定的所有颜色就构成了一个颜色空间。
- 颜色空间通常用三维模型表示，空间中的颜色通常使用代表三个参数的三维坐标来指定
- 基色颜色空间
  - **RGB**颜色模型
  - 对于打印设备来说，可以使用青色**(Cyan)**、品红**(Magenta)**、黄色和黑色颜料的用量来指定颜色（**CMYK**颜色模型）
- 色、亮分离颜色空间
  - 通过色调、饱和度和亮度来定义颜色（**HSL**颜色模型）
  - **YUV**（数据压缩的对象！）

RGB颜色模型

- 理论上绝大部分可见光谱都可用红、绿和蓝(RGB) 三色光按不同比例和强度的混合来表示。颜色C＝R(红色的百分比)＋G(绿色的百分比)＋B(蓝色的百分比)
- RGB模型称为相加混色模型，用于光照、视频和显示器。例如，显示器通过红、绿和蓝荧光粉发射光线产生彩色。





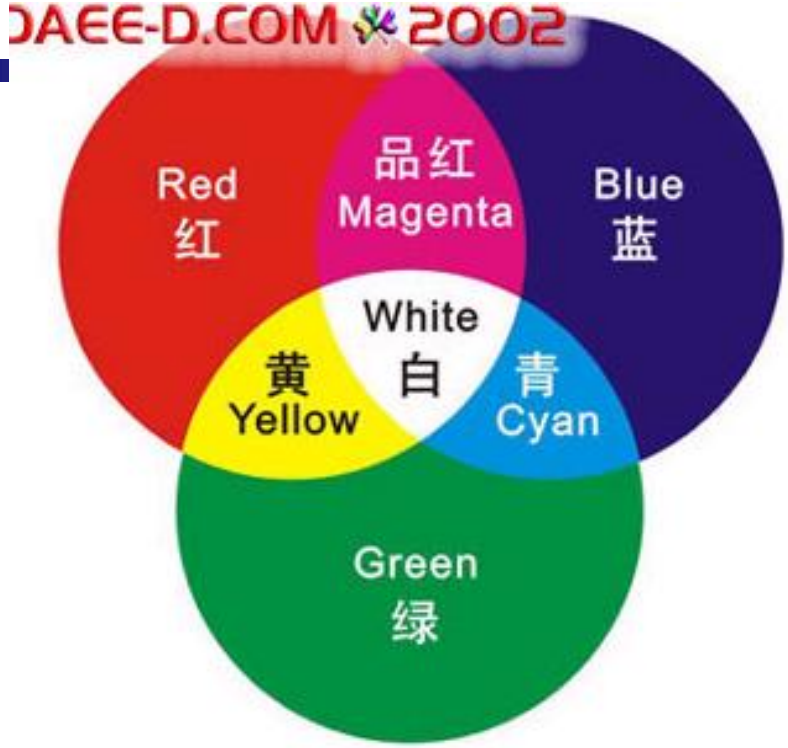
CMYK颜色模型

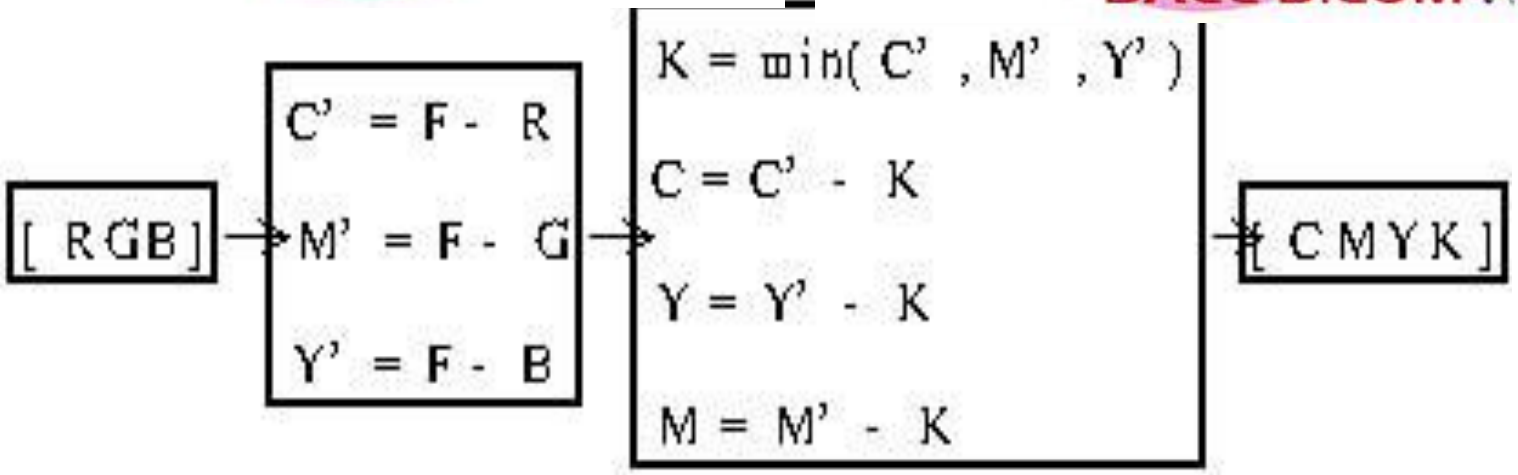
- 在理论上，绝大多数颜色都可以用三种基本颜料（青色**cyan**、品红**magenta**、和黄色**yellow**）按一定比例混合得到。
- 理论上，青色、品红和黄色三种基本色素等量混合能得到黑色。但实际上，因为所有打印油墨都会包含一些杂质，这三种油墨混合实际上产生一种土灰色，必须与黑色 **(K)** 油墨混合才能产生真正的黑色，所以再加入黑色作为基本色形成**CMYK**颜色模型。
- **CMYK**模型称为相减混色模型。

相加色与相减色的关系

RGB	CMY	生成的颜色
000	111	黑
001	110	蓝
010	101	绿
011	100	青
100	011	红
101	010	品红
110	001	黄
111	000	白

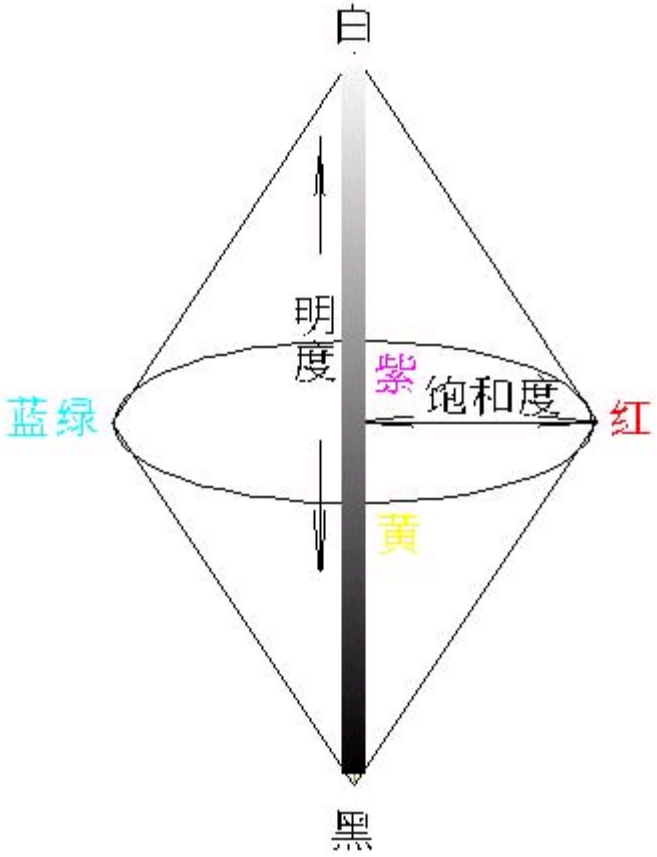
RGB模型到CMYK模型的转换



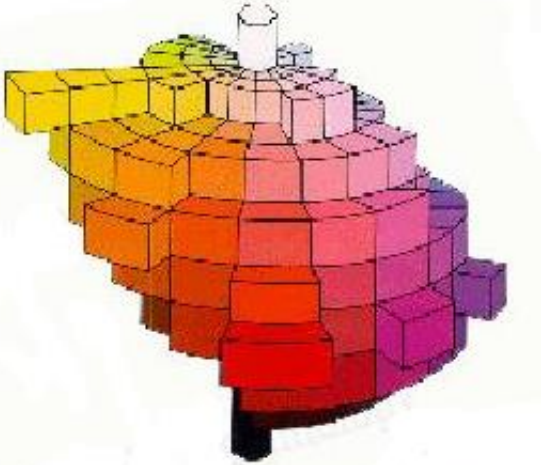


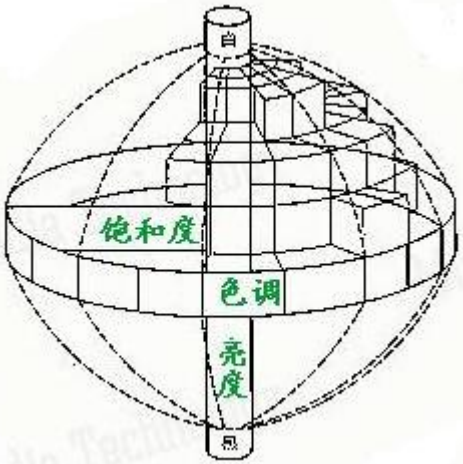
HSL颜色模型

- 在HSL模型中，H定义色调；S定义颜色的深浅程度或饱和度；L（Luminance/Lightness）定义亮度。
- RGB模型和CMYK模型主要是面向设备的，而HSL模型更容易被人理解和控制。

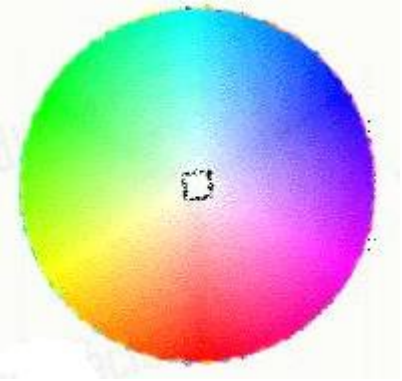


HSL用圆锥空间模型表示





- 纵轴表示亮度，沿圆锥轴线上的点表示完全不饱和色
- 圆锥纵切面：同一色调的不同亮度和饱和度关系
- 圆锥横切面：色调H为绕着圆锥截面度量的色环，圆周上的颜色为完全饱和的纯色，色饱和度为穿过中心的半径横轴
- **The hue is an angle from 0 to 360 degrees, typically 0 is red, 60 degrees yellow, 120 degrees green, 180 degrees cyan, 240 degrees blue, and 300 degrees magenta**



代码	R	G	B	H	S	L	颜色
0	0	0	0	160	0	0	黑(Black)
1	0	0	128	160	240	60	蓝(Blue)
2	0	128	0	80	240	60	绿(Green)
3	0	128	128	120	240	60	青(Cyan)
4	128	0	0	0	240	60	红(Red)
5	128	0	128	200	240	60	品红(Magenta)
6	128	128	0	40	240	60	褐色(Dark yellow)
7	192	192	192	160	0	180	白(Light gray)
8	128	128	128	160	0	120	深灰(Dark Gray)
9	0	0	255	160	240	120	淡蓝(Light blue)
10	0	255	0	80	240	120	淡绿(Light green)
11	0	255	255	120	240	120	淡青(Light cyan)
12	255	0	0	0	240	120	淡红(Light Red)
13	255	0	255	200	240	120	淡品红(Light Magenta)
14	255	255	0	40	240	120	黄(yellow)
15	255	255	255	160	0	240	高亮白(Bright white)

16色VGA调色板的值

HSL色彩空间和RGB色彩空间转换公式：

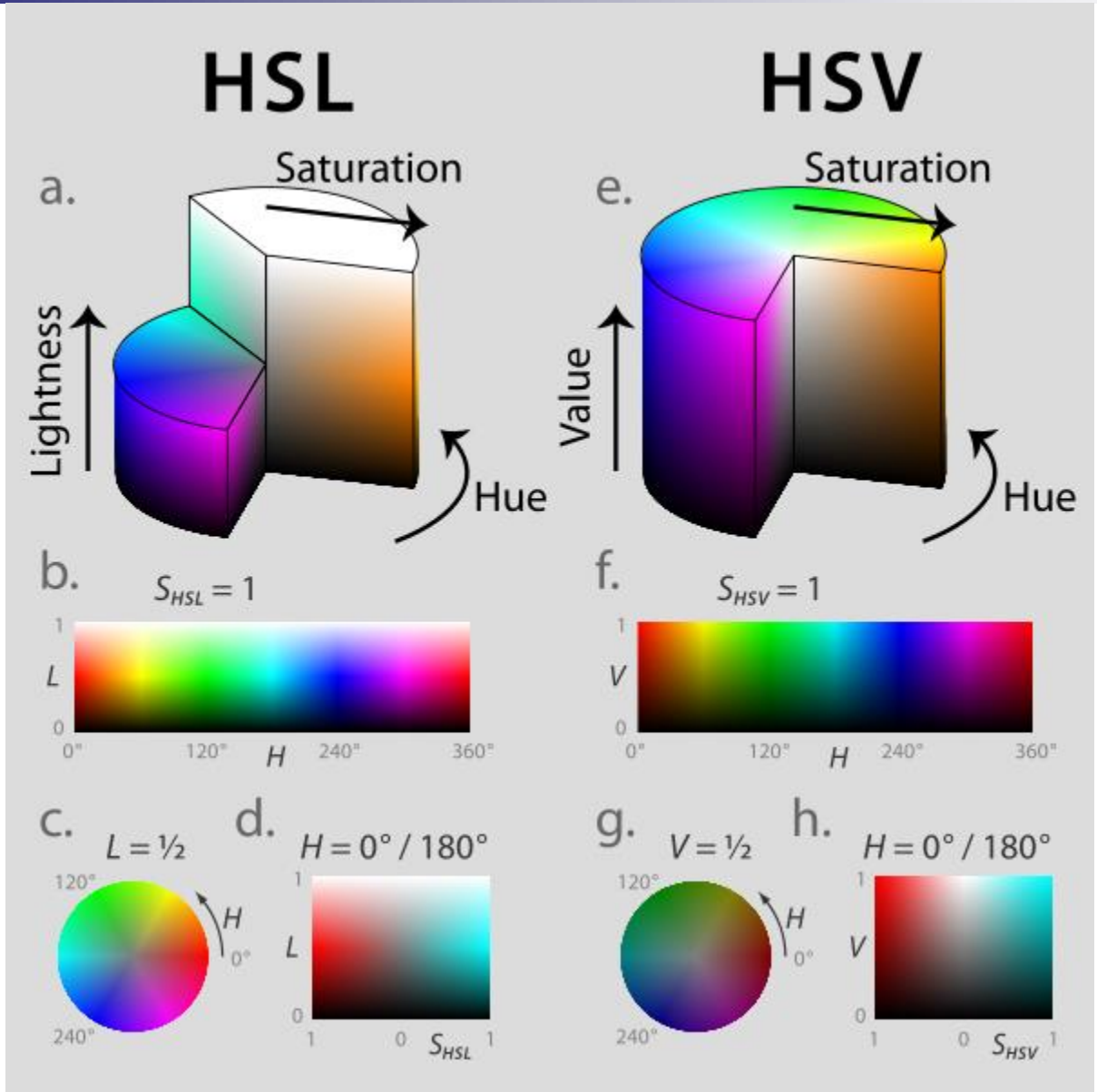
$$\begin{cases} I = \dfrac{R+G+B}{3} \\ H = \dfrac{1}{360}[90 - Arc\tan(\dfrac{F}{\sqrt{3}}) + \{0, G > B \ \ ; \ \ 180, G < B\}] \\ S = 1 - [\dfrac{\min(R,G,B)}{I}] \end{cases}$$

其中，$F = \dfrac{2R-G-B}{G-B}$

Q（1）在RGB颜色空间中，当R=G=B且为任意数值时，计算机显示器显示什么颜色？

Q（2）在HSL 颜色空间中，当H 为任意值，S=L=0 时，R，G 和B 的值是多少？当H＝0, S=0, L＝5，R，G 和B 的值是多少?

HSL与HSB(HSV)模型的差别



2.1.3 图像的属性

- 分辨率
  - 显示分辨率：指显示屏上能够显示出的象素数目。同样大小显示屏能够显示的象素越多，说明显示设备的分辨率越高，显示的图像质量也就越高。（640×480，1024 × 768）
  - 图像分辨率：指组成一副图像的像素的密度，一般用单位长度上包含像素的个数来衡量。常用单位为DPI（dots per inch），即每英寸多少点。
- 像素深度
  - 像素深度是指存储每个像素所用的位数。像素深度决定彩色图像每个像素可能有的颜色数，或者确定灰度图像每个像素可能有的灰度级数。
  - RGB16位（565或555），24位，32位

例 图像深度3，则可显示8色

R	G	B	颜色
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

2.1.3 图像的属性（续）

- 真彩色、伪彩色与直接色
  - 真彩色(true color)：真彩色是指在组成一幅彩色图像的每个像素值中，有R，G，B三个基色分量，每个基色分量直接决定显示设备的基色强度，这样产生的彩色称为真彩色。
  - 伪彩色(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(调色板)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，用查找出的R，G，B强度值产生的彩色称为伪彩色。
  - 直接色(direct color)：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。也就是通过相应的彩色变换表找出基色强度，用变换后得到的R，G，B强度值产生的彩色称为直接色。

2.1.4 图形与图像的类型

- 从生成、显示、处理和存储的角度划分
  - 矢量图
    - 矢量图是用一系列计算机指令来表示一幅图，如画点、画线、画曲线、画圆、画矩形等。这种方法实际上是用数学方法来描述一幅图。
    - 矢量图文件中存储的是数学表达式和指令，绘制和显示矢量图时，可以采用绘图程序进行处理操作
    - 将每个图元对象作为一个独立体处理，缩放、旋转、移动时不会影响其它对象。特别适用于要求创建和操作单个对象的图例和几何造型

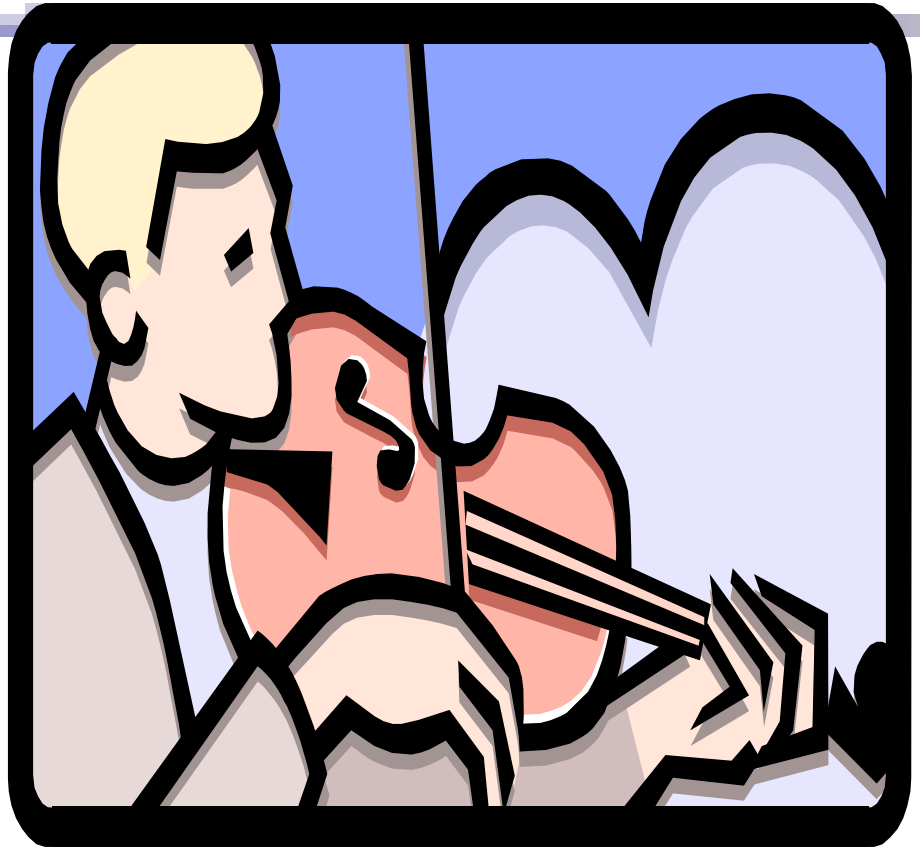
2.1.4 图形与图像的类型

- 矢量图
  - 优点：计算生成，显示精度高、操作灵活性大。常常用来设计精细的线框形图案、商标、标志等适合数学运算表示的美术作品和图形的生成技术
  - 缺点：图像显示时花费时间比较长；真实世界的彩色图像难以转化为矢量图 （可以把相同或类似的图独立作为图的构造模块，存到图库中以便于加速图的生成和减少文件大小）
  - 常用文件格式：如**Adobe Illustrator的*.AI、*.EPS和SVG(Scalable Vector Graphics) 、AutoCAD的*.dwg和dxf、Corel DRAW的*.cdr、windows标准图元文件*.wmf和增强型图元文件*.emf** 等
  - **AutoCAD、CoreDraw** 等绘图软件都是基于矢量的系统。

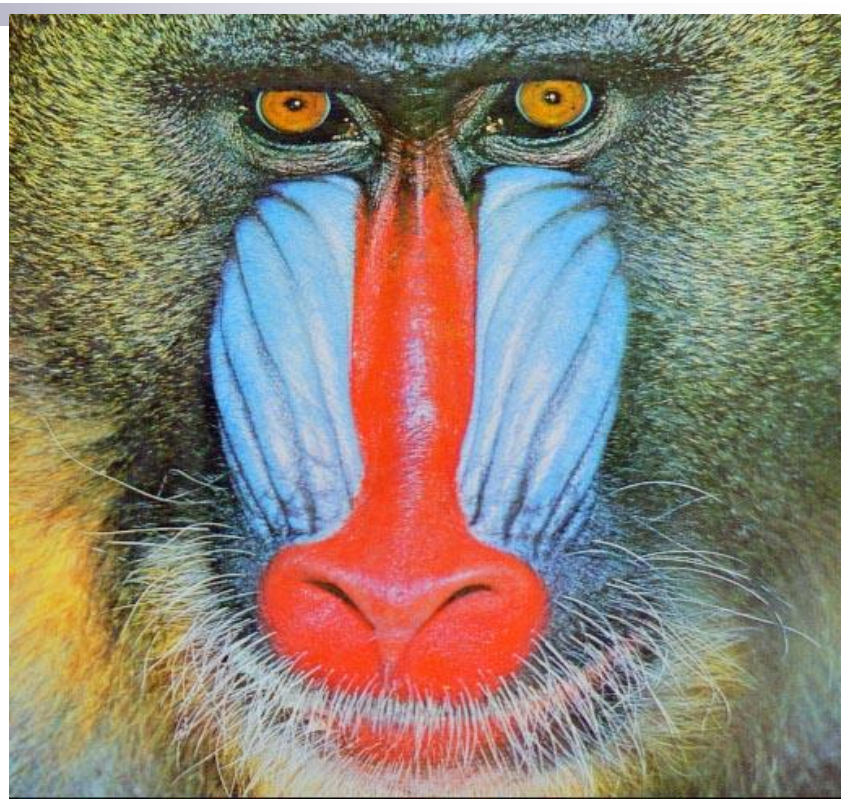
2.1.4 图形与图像的类型（续）

- 点位图
  - 点位图是将一副图像在空间上离散化，即将图像分成许许多多的像素，每个象素用若干个二进制位来指定该像素的颜色或灰度值。
  - 位图即为用图像数据集合的形式对构成图像的各个像素点的颜色和亮度等相关信息进行描述和存储
  - 优点：适合表现细致、色彩和层次比较丰富的图像，是所有视觉图像表示方法的基础。显示速度快。真实世界的图像可以通过扫描仪、数码相机、摄像机等设备方便的转化为点位图。
  - 缺点：存储和传输时数据量比较大。缩放、旋转时算法复杂且容易失真。

图像的分类



矢量图



点位图

Q: 矢量图和点位图可以互换吗？

2.1.4 图形与图像的类型（续）

- 图形与图像的区别与联系
  - 从图像显示内容的角度
    - 一般来说，图像所表现的显示内容是自然界的真实景物，或利用计算机技术逼真地绘制出的带有光照、阴影等特性的自然界景物，而图形实际上是对图像的抽象，组成图形的画面元素主要是点、线、面或简单立体图形等，与自然界景物的真实感相差很大。
    - 图形更多的用矢量图表示，图像更多的用位图表示
  - 一旦在屏幕上显示，图形与图像无异
    - 都是以一定的分辨率和颜色深度在屏幕上以点阵的形式显示出来

图像的分类

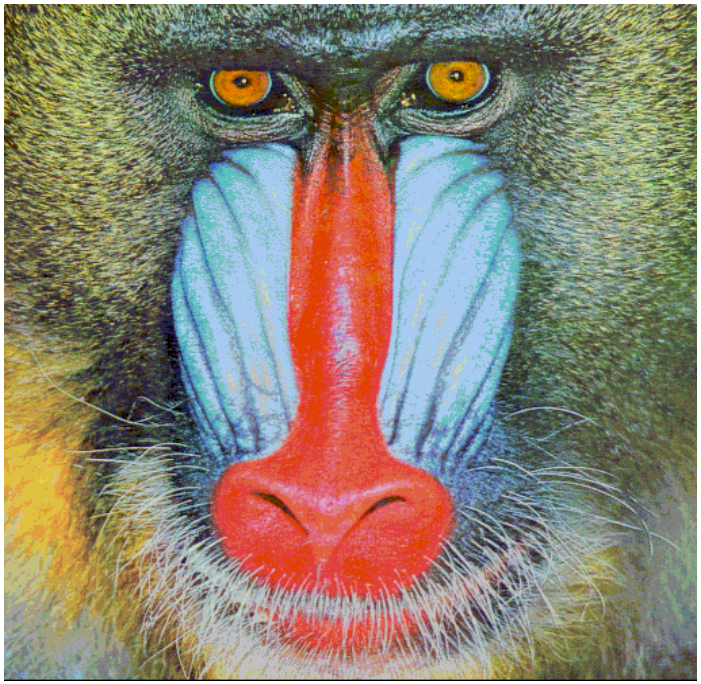
- 灰度图

标准单色图

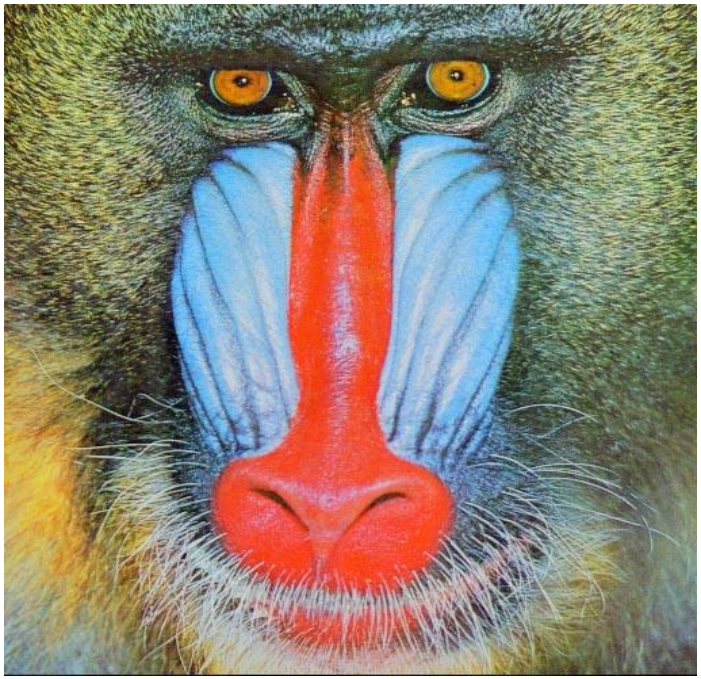
标准灰度图

图像的分类

- 彩色图



256色标准图像



24位标准图像

2.1.5 常见的图形与图像文件格式

- BMP文件格式
- GIF文件格式
- JPG文件格式（第三章中讲述）
- 需要说明的：
  - 图像存储时一般由两部分组成：图像说明部分和图像数据部分
    - 图像说明部分：图像的格式、深度、高度、宽度、调色板、压缩方法等
    - 图像数据部分：描述图像每个像素的数据
- 多媒体文件的设计思想
  - 媒体信息以灵活和可扩展的方式组织，以便于媒体的交换、管理、编辑和呈现

BMP图像文件格式

位图文件(Bitmap-File，BMP)格式是Windows采用的图像文件存储格式，在Windows环境下运行的所有图像处理软件都支持这种格式。BMP位图文件默认的文件扩展名是bmp或者dib。

BMP文件大体上分为四个部分：

位图文件头BITMAPFILEHEADER
位图信息头BITMAPINFOHEADER
调色板Palette
实际的位图数据ImageData

BMP图像文件格式

```
typedef struct tagBITMAPFILEHEADER {
        WORD        bfType;          /* 说明文件的类型 */
        DWORD       bfSize;          /* 说明文件的大小，用字节为单位 */
                                     /*注意此处的字节序问题
        WORD        bfReserved1;     /* 保留，设置为0 */
        WORD        bfReserved2;     /* 保留，设置为0 */
        DWORD       bfOffBits;       /* 说明从BITMAPFILEHEADER结构
                                        开始到实际的图像数据之间的字 节
                                        偏移量 */
}   BITMAPFILEHEADER;
```

BMP图像文件格式

```
typedef struct tagBITMAPINFOHEADER {
    DWORD   biSize;       /* 说明结构体所需字节数 */
    LONG    biWidth;    /* 以像素为单位说明图像的宽度 */
    LONG    biHeight;  /* 以像素为单位说明图像的高度*/
    WORD    biPlanes;    /* 说明位面数，必须为1 */
    WORD    biBitCount;  /* 说明位数/像素，1、2、4、8、24 */
    DWORD   biCompression;  /* 说明图像是否压缩及压缩类型
                BI_RGB，BI_RLE8，BI_RLE4，BI_BITFIELDS */
    DWORD   biSizeImage;    /*  以字节为单位说明图像大小，必须是4的
                                整数倍*/
    LONG    biXPelsPerMeter;    /*  目标设备的水平分辨率，像素/米 */
    LONG    biYPelsPerMeter;    /*目标设备的垂直分辨率，像素/米 */
    DWORD   biClrUsed;    /* 说明图像实际用到的颜色数，如果为0
                            则颜色数为2的biBitCount次方 */
    DWORD   biClrImportant;  /*说明对图像显示有重要影响的颜色
                索引的数目，如果是0，表示都重要。 */
} BITMAPINFOHEADER;
```

BMP图像文件格式

调色板实际上是一个数组，它所包含的元素与位图所具有的颜色数相同，决定于biClrUsed和biBitCount字段。数组中每个元素的类型是一个RGBQUAD结构。真彩色无调色板部分。

```
typedef struct tagRGBQUAD {
    BYTE    rgbBlue;        /*指定蓝色分量*/
    BYTE    rgbGreen;       /*指定绿色分量*/
    BYTE    rgbRed;         /*指定红色分量*/
    BYTE    rgbReserved;    /*保留，指定为0*/
} RGBQUAD;
```

BMP图像文件格式

紧跟在调色板之后的是图像数据字节阵列。对于用到调色板的位图，图象数据就是该象素颜色在调色板中的索引值（*逻辑色*）。对于真彩色图，图象数据就是实际的**R**、**G**、**B**值。（对**1**位，**4**位，**8**位彩色一个字节各表示多少个像素？）

图像的每一扫描行由表示图像像素的连续的字节组成，每一行的字节数取决于图像的颜色数目和用像素表示的图像宽度。规定每一扫描行的字节数必需是**4**的整倍数，也就是**DWORD**对齐的（如果不是整数倍怎么办？）

扫描行是由底向上存储的，这就是说，阵列中的第一个字节表示位图左下角的像素，而最后一个字节表示位图右上角的像素。（只针对于倒向**DIB**，如果是正向**DIB**，则扫描行是由顶向下存储）

关于BMP文件可进一步扩展的问题

- 24位转1位，4位，8位，16位(r5g5b5,r5g6b5)位图如何操作？
- 8位图像转24位图像如何操作？
- 更改调色板？☺
- 假设取得某526色BMP位图数据和调色板，如何显示？
- C#中建立了个对话框，然后打开了一幅BMP图片，并且显示在了对话框上，怎样获得该位图上指定坐标点的象素值?
- 如何将视图窗口中的自绘图，以BMP格式方式保存？

2.2 视频的特性及文件格式

- 2.2.1 视频信号的数字化及标准
- 2.2.2 AVI文件解析

2.2.1 视频信号数字化及标准

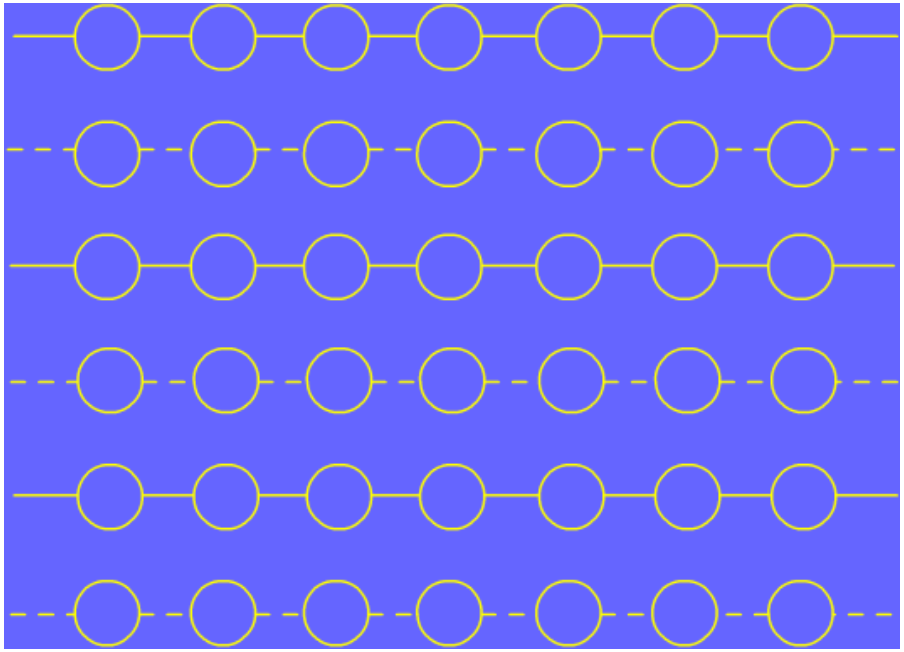
全信号数字化：直接对复合信号数字化的方式。

分量数字化：分别对Y、U、V分量信号数字化方式。

优点：Y、U、V分量互不干扰，不需要反复的编码与解码，图像质量高。

2.2.1 视频信号数字化及标准

正交取样结构：每一场中的样点都重合，而且都对齐



2.2.1 视频信号数字化及标准

注：为了获得这种取样结构，就要求取样频率是行频的整倍数

1. 亮度信号的取样频率

亮度信号对取样频率的要求如下：

（1）按照奈奎斯特取样定理，取样频率至少应为信号上限频率的两倍。

（2）为了在取样后保证产生足够小的混叠噪声，要求取样频率是信号带宽的2.2～2.7倍。对PAL制信号，取样频率应大于13.2 MHz。

2.2.1 视频信号数字化及标准

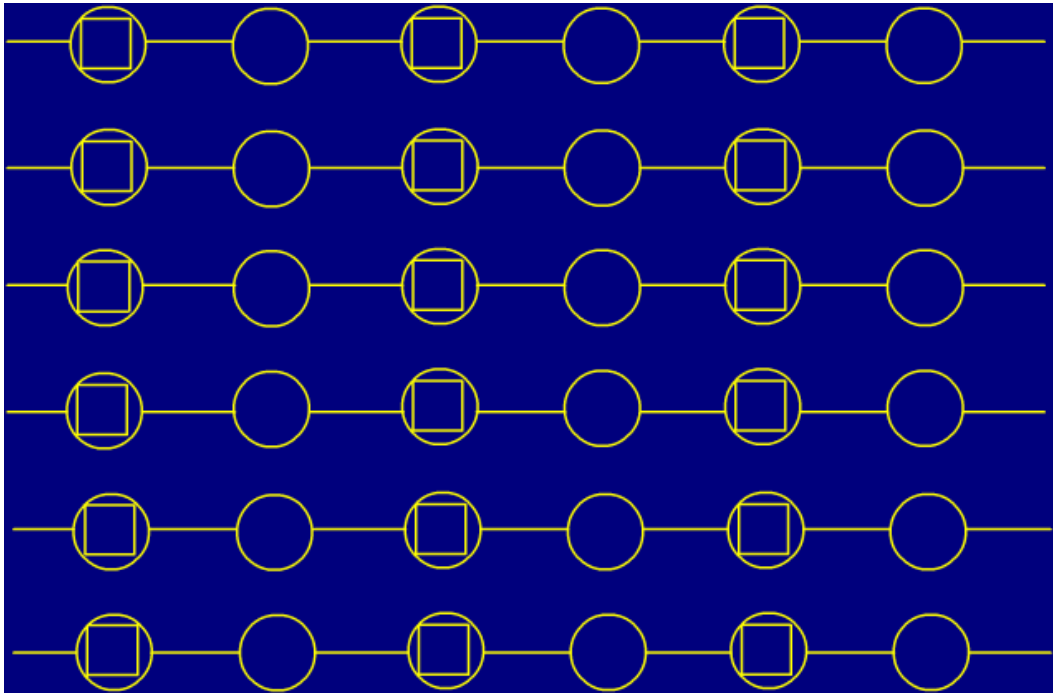
（3）为了获得正交取样结构，取样频率必须是行频的整数倍。

（4）为了使两种扫描制式实现兼容，应采用同一种取样频率，625行制的行频为15.625Hz，525行制的行频为15.734Hz，两者的最小公倍数为2.25 MHz。

结论：取样频率应大于13.2 MHz，故将亮度信号的取样频率定为2.25 MHz的6倍，为13.5 MHz。

2.2.1 视频信号数字化及标准

2．色差信号的取样频率：将色差信号的取样频率定为6.75 MHz。取样结构见图所示。



2.2.1 视频信号数字化及标准

(1)视频信号量化位数的确定：8位（bit）

$S/N=6n+10.8$(dB) 为视频信号的信噪比计算公式。如果把视频信号的信噪比定为大于50dB，则量化位数应不低于7位。当然位数越高，信噪比也越高，每增加一位，信噪比可提高6dB，但电路的复杂性和设备的成本也会大大提高。对视频信号采用8位（bit）量化位数显然是较为合理的。这样，经一次量化处理后其信噪比可以达到59dB。

2.2.1 视频信号数字化及标准

(2)量化电平（码电平）：

对应模拟信号电平的量化级电平,即量化级所对应的电平值。

(3)码电平的分配：

为防止信号电平的过载，而将视频信号严格地调整到范围内，并且不把8bit视频量化的256个量化级都分配给满幅度信号，而在上下各留一个保护带。

亮度信号的码电平分配:（单极性信号）上留20级、下留16级为保护带。（16-235）

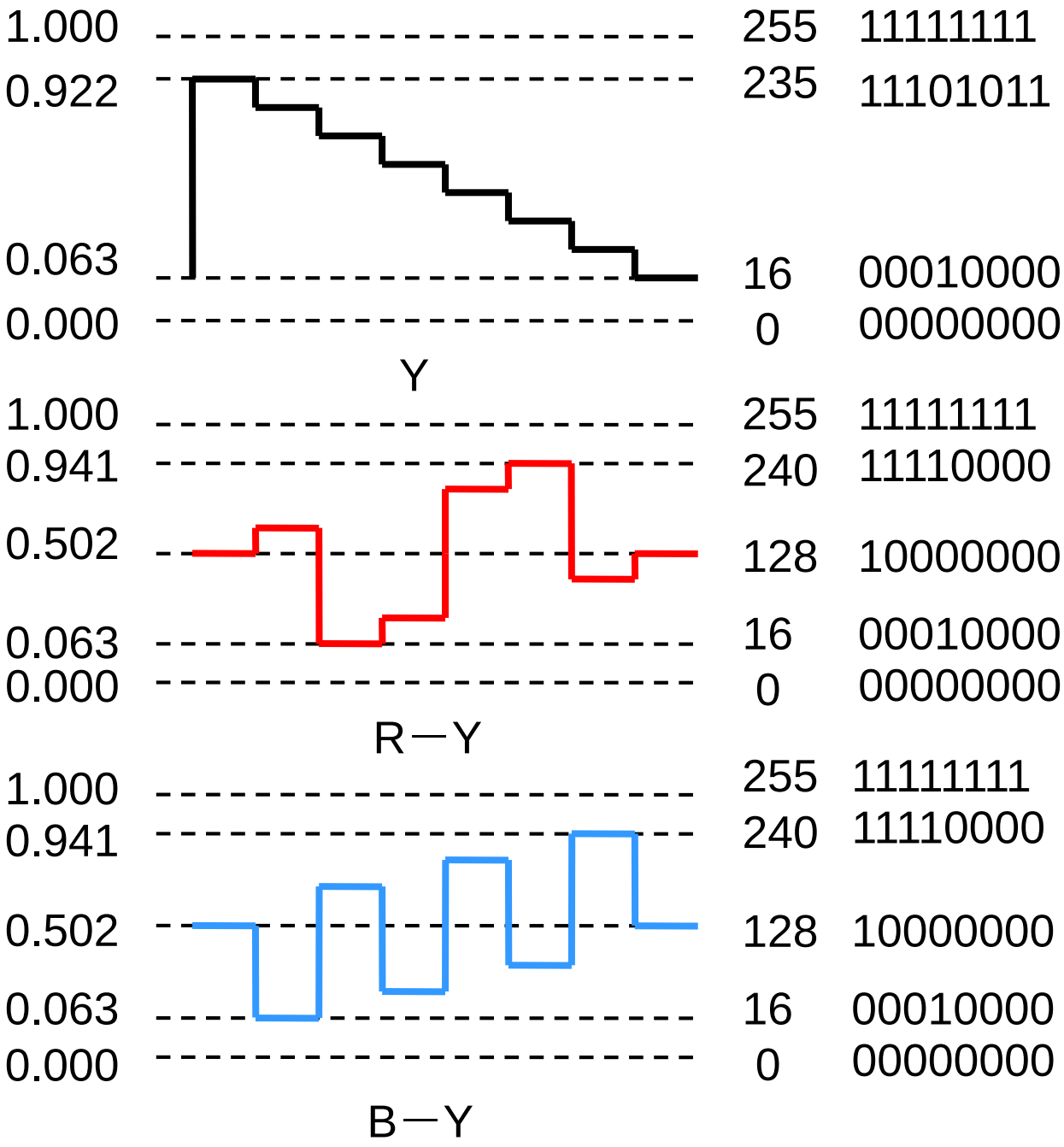
色差信号的码电平分配:（双极性信号）上下各16级为保护带。（16-240）

YCbCr和RGB的转换关系

$$\begin{bmatrix} Y \\ B-Y \\ R-Y \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.299 & -0.587 & 0.866 \\ 0.701 & -0.587 & -0.114 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- 亮度分量的取值范围为[16，235]。16表示黑电平值，235表示白电平值。数值范围可认为[0，1]。
- 色差信号的数值范围为[16，240]，使用128的偏移量时取值范围为[-112，112]。数值范围应为[-0.5，0.5]
- 因此对B-Y和R-Y分别用(0.5/0.866)和(0.5/0.701)相乘，就将色差信号的范围转换为[-0.5，0.5]
- 转换程序多用部分查表法实现。*例子*

$$\begin{bmatrix} Y \\ B-Y \\ R-Y \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$


1.000
0.922
0.063
0.000
255 11111111
235 11101011
16 00010000
0 00000000
Y
1.000
0.941
0.502
0.063
0.000
255 11111111
240 11110000
128 10000000
16 00010000
0 00000000
R—Y
1.000
0.941
0.502
0.063
0.000
255 11111111
240 11110000
128 10000000
16 00010000
0 00000000
B—Y

2.2.3 视频信号数字化及标准

码率R=13.5 MHz×8bit+（6.75 MHz×8bit）×2=216Mb/s

并行数据传输： R=13.5 MHz+6.75 MHz×2=27 Mb/s

数字行:数字化后的一行信号称为数字行。

1.每行取样点数

每秒的取样点数（取样频率13.5 MHz）×每行扫描时间(64μs)=每行取样点数

PAL制的每行取样点数为：13500 kHz×64 μs=864

NTSC制的每行样点数为：13500 kHz×63.55 μs=858

2.2.3 视频信号数字化及标准

2.有效水平取样点

为了使两种制式兼容，有效部分均取53.3μs，由于在13.5 MHz的取样频率下，一次取样需0.074μs，所以每行有效部分取样点均为:

亮度信号:　53.3μs÷0.074μs＝720（T）

色差信号：色差信号的采样频率为亮度信号的一半（6.75 MHz）故为360（T）

2.2.3 视频信号数字化及标准

3.有效垂直取样点（扫描行数）

在数字化视频中：

定义：PAL制每帧有效行为576行，每场有效行均为288行。

NTSC制每帧有效扫描行数为480行，每场有效行数均为240行。

2.2.3 视频信号数字化及标准

PAL制与NTSC制视频数字化扫描行参数表

	PAL（625H/50V）		NTSC（525H/60V）	
每帧中行数	625H		525H	
每帧有效数	576		480	
多场正程	奇数场	偶数场	奇数场	偶数场
	288H	288H	240H	240H
每场逆程	24H	25H	22H	23H

2.2.3 视频信号数字化及标准

为了规范国际上存在的多种视频数字化格式，国际无线电咨询委员会（CCIR）于1982年2月在其第15届全体会上通过了CCIR－601标准，后又改名为ITU－601标准，这是用于电视演播室等级的标准，又称为4：2：2标准，在美国称为“D1标准”，表中写出了标准的基本内容。

2.2.3 视频信号数字化及标准

4：2：2标准参数表

参数名称		PAL（625/50）	NTSC(525/60)
编码信号（分量信号）		Y,U,V	Y,I,Q
全行亮度信号采样点数		864	858
全行色差信号采样点数		432	429
取样结构		正交结构，即行、场和帧重复的样点同位，并和每行第奇数个Y样点同位	
取样频率	亮度信号	13.5MHz	
	色差信号	6.75MHz	
编码方式		均采用每采样点8bit均匀量化脉码调制（PCM）	
全数字行有效点数	亮度信号	720	
	色差信号	360	
图象分辨率	亮度信号	720×576	720×480
	色度信号	360×576	360×480
数码传输速率（R）		216Mbit/s 或 27MB/s	
视频信号电平与量化级间的对应关系	亮度信号	共220个量化级（黑电平对应第16级，峰值白电平对应235级）	
	色差信号	共224个量化级	

取样参数

		4：4：4	4：2：2	4：2：0	4：1：1
取样频率（MHz）	Y	13.5	13.5	13.5	13.5
	R-Y	13.5	6.75	6.75	3.375
	B-Y	13.5	6.75	6.75	3.375
每帧有效行数	Y	576	576	576	576
	R-Y	576	576	288	576
	B-Y	576	576	288	576
每行有效象素数	Y	720	720	720	720
	R-Y	720	360	360	180
	B-Y	720	360	360	180
帧频（Hz）		25	25	25	25
宽高比		4:3	4:3	4:3	4:3

AVI

A Microsoft Standard for Video

Audio Video Interleave

- 微软开发的音频/视频格式
- “interleave” 指视频数据和音频数据以交织形式保存在文件中
- 通常，AVI文件包含多个不同类型的音频和视频数据流。

AVI is a special case of RIFF

- AVI 是Resource Interchange File Format的子集或特例。
- 此规范用来捕获、编辑和播放音频和视频序列。

RIFF 和AVI格式的历史

- Electronic Arts 首先于1984年发明了 IFF format
- Electronic Arts 将 IFF 格式用于在Amiga系统中绘制图画,且该格式成为该平台上的视频和音频交换标准。
- IBM和Microsoft加入，RIFF标准开始成为视频和音频在DOS/Windows平台上互换的标准。
- RIFF格式和IFF格式非常类似。

AVI 格式的历史

- 在 1991到1992时表示视频
- Microsoft在windows的早期版本中使用。
- 许多评论家认为AVI格式之所以流行，是由于它与Microsoft的操作系统绑在一起。
- 标准由Microsoft拥有。也即，该标准是由Microsoft开发和修改，而非一个开放的、由标准委员会制定。

RIFF 文件包含

- RIFF FORM HEADER – ASCII字符‘RIFF’，后跟特定格式的相应标识符。
- 文件剩下的部分是特定格式的数据。

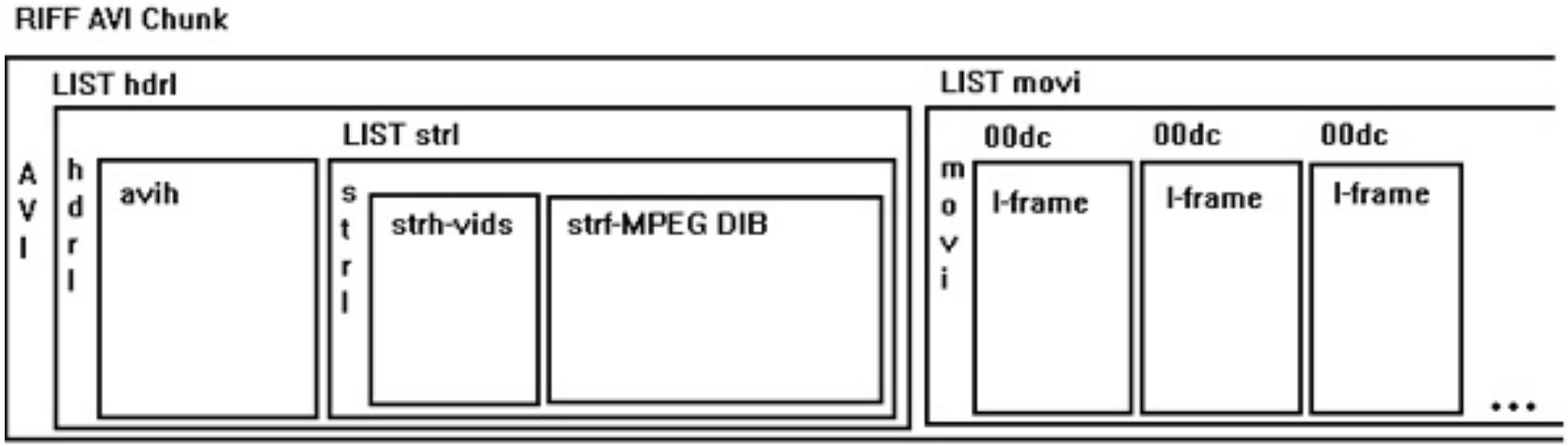
RIFF 数据格式

- 有两种类型:
  - **Chunks** – 包含4字节的标识符，4字节表示该chunk的大小，以及数据。
  - **Lists** –一个chunk的子集，它包含关键字“LIST,”4字节表示该 list的大小，4字节标识符，以及数据。

RIFF AVI 格式

- 所有AVI文件以‘RIFF’标识符后跟的标识码‘AVI’开始
- 所有AVI文件包含两个必须的 LIST chunk和一个索引 chunk:
  - ‘hdrl’ LIST – 主要定义数据的格式。它包含主要的AVI格式头。
  - ‘movi’ LIST – 这包含实际的音频和视频数据。
  - ‘idxl’ chunk – 需要此chunk以索引文件。

RIFF AVI 格式


RIFF AVI Chunk
AVI
LIST hdrl
hdrl
avih
LIST strl
strl
strh-vids
strf-MPEG DIB
LIST movi
movi
00dc
I-frame
00dc
I-frame
00dc
I-frame
...

RIFF AVI 格式

- RIFF xxxx ‘AVI ’
  { LIST xxxx ‘hdrl’
    { ‘avih’ (<Main AVI Header>)
      LIST xxxx ‘strl’
          {   ‘strh’(<Stream header>)
              ‘strf’(<Stream format>)
              ‘strd’(additional header data)
          }
    }
     LIST xxxx ‘movi’
     {SubChunk1
      SubChunk2…
     }
   [‘idx1’ <AVIIndex>]
  }

The Main AVI Header LIST

```
typedef struct {
    DWORD  dwMicroSecPerFrame;   // 定义视频帧之间的时间间隔，这个值确定文件
                                    的总体时速。ns
    DWORD  dwMaxBytesPerSec;     // 定义文件的大致数据率，这个值确定要显示由
                                    此Main Header  及流的Header chunks所含
                                    其他参数定义的AVI系列，系统必输速率（每
                                    秒字节数）
    DWORD  dwPaddingGranularity;  //记录块的长度需为此值的倍数，通常是2048
    DWORD  dwFlags;               // 包含文件中的任何标志字。（如有无”idx1”
                                    块，为确定数据的显示次序所要用到的索引，
                                    AVI文件是否Interlaced,是否含版权 信息等）
    DWORD  dwTotalFrames;         // 文件中含的数据帧个数。
    DWORD  dwInitialFrames;       // 对interlaced文件，确定在文件中在AVI系列
                                    初始帧之前所含的  数据帧的个数。
    DWORD  dwStreams;             // 文件中所含流的个数，如既有视频又有音频
                                    的文件含两个流。
    DWORD  dwSuggestedBufferSize; // 定义读文件所建议的Buff大小，大于最大的
                                    Chunk的大小。
    DWORD  dwWidth;               //定义AVI文件就像素而言的宽与高
    DWORD  dwHeight;              //
    DWORD  dwScale;               //与dwRate一起定义文件所要用到的通用时间
                                  刻度，dwRate/dwScale为每秒的采样时间刻度。
    DWORD  dwRate;                //
    DWORD  dwStart;               // 定义AVI文件的起始时间。通常设为0
    DWORD  dwLength;              //定义AVI文件的长度。
} MainAVIHeader;
```

The Stream Header ("strl") Chunks

主文件头后紧跟一个或几个"strl"块(每个数据流需要一个"strl" 块)。包含文件中数据流的信息。每个"strl"块必须包含一个Stream头和Stream格式块。Stream头由四个字符"strh"标记，Stream格式块由四个字符"strf"标记，此外"strl"块也可能包含一个流数据块，流数据块由四个字符”strd”来标识。

Stream头的数据格式定义为:

```
typedef struct {
  FOURCC  fccType;     //若此流含的是video数据，此域值为"vids"，若为audio数据,则为”auds”.
   FOURCC  fccHandler;  //为四个字符，描述数据所用的压缩、解压缩算法。
   DWORD   dwFlags;        //数据流属性）
   WORD    wPriority;   //此数据流的播放优先级
    WORD    wLanguage;   //音频的语言代号
DWORD   dwInitialFrames; //用于interlaced文件，定义在文件中在AVI系列初始帧之前所含的数据帧的个数。
   DWORD   dwScale;                  //与dwRate一起定义回放速率。
   DWORD   dwRate;                   //
   DWORD   dwStart;       //序列的起始时间
   DWORD   dwLength;      //系列的长度
   DWORD   dwSuggestedBufferSize;  //回放所需的Buff大小
   DWORD   dwQuality;     //数据质量标志字
   DWORD   dwSampleSize;  //采样大小} AVIStreamHeader;
```

The Stream Header ("strl") Chunks

- 在stream头（”strh”）块之后紧接着出现的是stream格式块("strf")，它描述流中数据的格式。对视频流而言，这一块中的信息为一个BITMAPINFO结构。

```
typedef struct tagBITMAPINFO
{
    BITMAPINFOHEADER    bmiHeader;
    RGBQUAD       bmiColors[ ]; //颜色表
}BITMAPINFO;
```

The Stream Header ("strl") Chunks

- 在stream头（”strh”）块之后紧接着出现的是stream格式块("strf")，它描述流中数据的格式。对音频流而言，这一块中的信息为一个WAVEFORMATEX或PCMWAVEFORMAT结构（WAVEFORMATEX结构为WAVEFORMAT结构的扩展版本）。

```
typedef struct
{
    WORD   wFormatTag;
    WORD   nChannels;   //声道数
    DWORD nSamplesPerSec; //采样率
    DWORD nAvgBytesPerSec; //WAVE声音中每秒的数据量
    WORD   nBlockAlign;    //数据块的对齐标志
    WORD   biSize;         //此结构的大小
}WAVEFORMAT
```

The Stream Header ("strl") Chunks

- "strl"块也可能包含一个流数据块（"strd" chunk），它一般在stream格式块("strf")之后，这一块的格式和内容由压缩或解压缩的驱动程序来定义。一般驱动程序利用这些信息来配置读写RIFF文件的应用程序，而不必解码它们。“strd”子块不一定存在，也没有固定的结构。
- AVI播放器将LIST "hdrl"块中的Stream头结构与LIST "movi"块中的stream数据**通过利用"strl"块的次序联系起来.**第一个"strl"块用于stream 0, 第二个用于stream 1, 依次下去。例如，如果第一个"strl" 块描述wave audio数据,则wave audio数据在stream 0中。同样,如果第二个"strl" 块描述video数据,则此video数据在stream 1中。

The LIST "movi" Chunk

- 头信息之后是LIST “movi”块，它们包含流中的实际数据块，即图象和声音本身。这些数据块可以直接放在LIST “movi” 块中。
- 同任何RIFF块类似,数据块包含一个四字符的代码来表示其类型。此四字符的代码通过一个流编号和一个两字符的代码来定义该数据块中的信息。例如，一个wave块可以用 “wb”这两个字符来表识，**如果这个wave块与第二个LIST “hdrl”流标识对应，它的四字符代码就是 “01wb”。**
- 因为所有的**格式信息都在头结构中**，这些数据块中包含的audio数据不含其格式的任何信息。一个audio数据块包含如下信息 (##代表流编号):

```
WAVE  Bytes   '##wb'
        BYTE    abBytes[];
```

The LIST "movi" Chunk

- Video数据可以是压缩后或未压缩的DIBs数据。未压缩的DIB在其对应的BITMAPINFO结构中biCompression域为BI_RGB，一个压缩后的DIB在其biCompression 域为非BI_RGB的值。
- 包含RGB视频数据的未压缩DIB对应的数据块由 “db”两字符表示 (db为DIB简写)。压缩后DIB对应的数据块由”dc”两字符表示（为DIB compressed的简写)。**这两种数据块都不含有关DIBs的任何头信息。**未压缩DIB对应的数据块有如下的形式：

DIB Bits '##db'

BYTE abBits[];

压缩的DIB对应的数据块的形式为：

Compressed DIB '##dc'

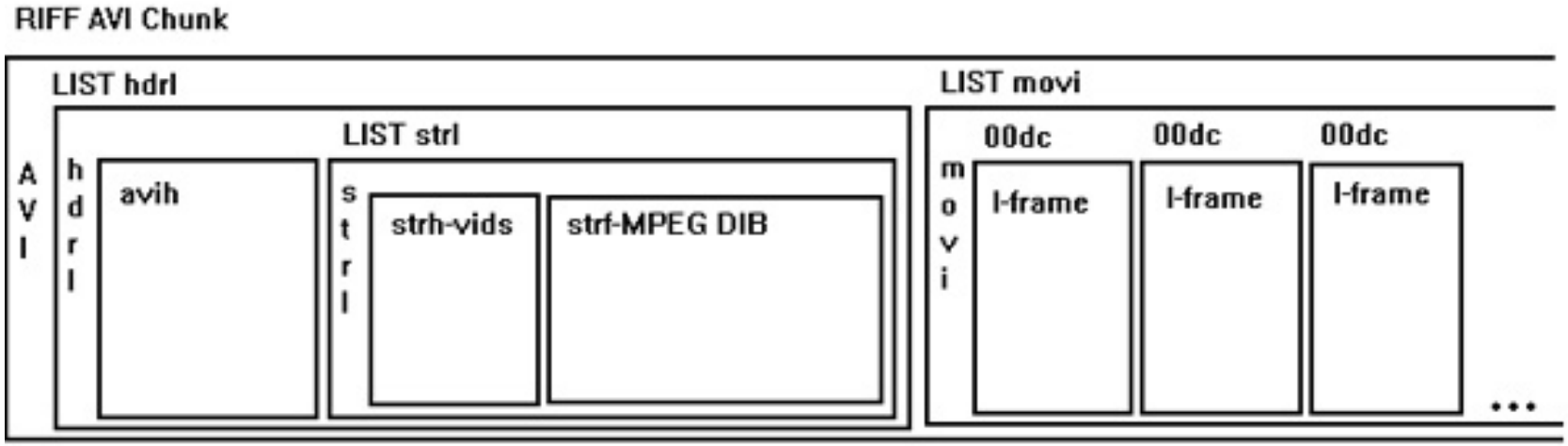
BYTE abBits[];

The "idx1" Chunk

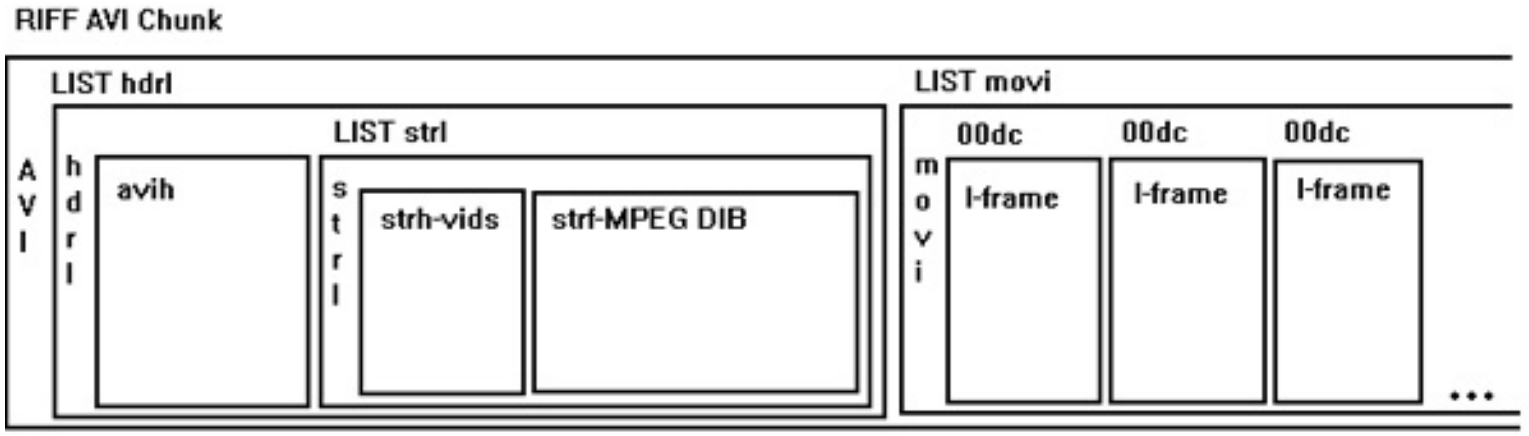
- 在LIST "movi"块之后，AVI文件可以有一个索引块。索引块主要包含数据块列表和它们在文件中的位置。这样就可以随机访问文件中的数据。
- 索引块的四字符代码为"idx1"。其结构定义为：

```
typedef struct {
    DWORD  ckid;       //用四字符来标识数据块。
    DWORD  dwFlags;    //标志位
    DWORD  dwChunkOffset;  //定义数据块相对于'movi' list的位置。
    DWORD  dwChunkLength;  //定义数据块除去8字节的RIFF头信息后的长度。
} AVIINDEXENTRY
```

RIFF AVI 格式

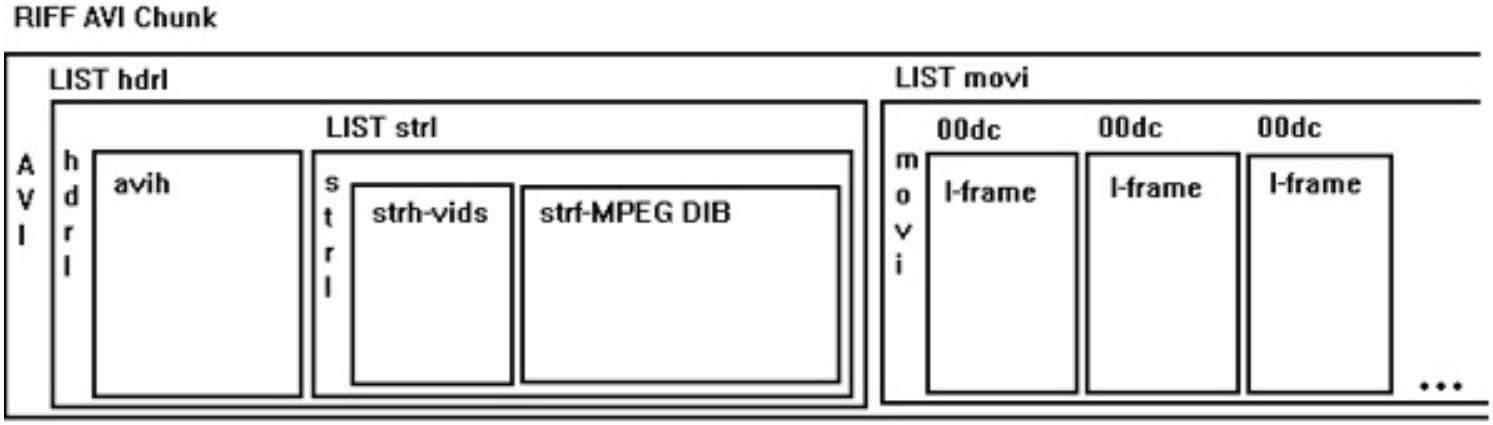

RIFF AVI Chunk
AVI
LIST hdrl
hdrl
avih
LIST strl
strl
strh-vids
strf-MPEG DIB
LIST movi
movi
00dc
I-frame
00dc
I-frame
00dc
I-frame
...

HDRL Header



- Header包含必须的AVI header chunk或‘avih.’ ‘hdrl’ header将‘hdrl’ LIST chunk和 ‘movi’ chunk内包含的内容组织起来。
- 在 ‘avih’ chunk中嵌套一个或多个 Stream List, 或‘strl’ LIST，给出有关文件中流的信息。
- 流可以包括视频和音频，每个数据流都有一个chunk。

Interleaving the Chunks



- AVI播放器将LIST ‘hdrl’内包含的格式与 LIST ‘movi’ chunk联系起来。.
- 当播放器解析数据时，‘strl’中的第一个chunk与第一个“流”联系在一起。
- 例如，如果第一个 ‘strl’描述音频而第二个描述视频，则 AVI播放器将 ‘movi’ list 中第一个chunk中的原始数据认为是音频，第二个chunk 认为是视频。

Stream Data Chunk

- LIST ‘movi’ 可简单地包含标识为视频或音频 chunk的数据。
- LIST ‘movi’ 可包含一个 ‘rec’ LIST. 在 ‘rec’ LIST中有单帧的音频和视频。
- 之所以设计 ‘rec’ list，是因为 AVI播放器可以从disk上一次读出chunk。因此可以从CD-ROM上播放视频。

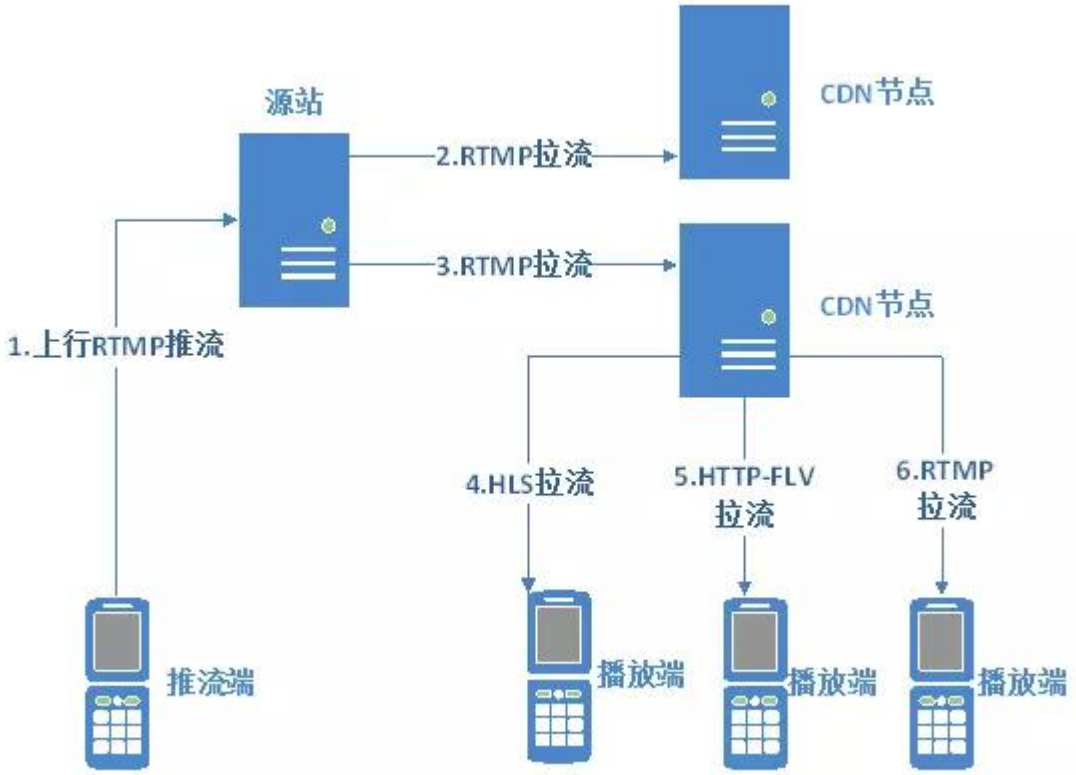
关于AVI文件

- AVI RIFF文件格式只规定了文件的组成方式,即各种数据如何在文件中排列等,对于文件中的数据并没有做出编码格式的约束。
- AVI文件结构可以看成是一个数据容器，AVI格式只规定了将数据装入这个容器的方法,但对数据本身的编码格式并不涉及。要操作一个AVI文件（播放、抓图、编辑）,必须在系统中安装与该AVI文件中的编码器对应的解码器,才能进行正确的工作

关于AVI文件进一步扩展的问题

- 按照目前所讲的AVI规范，AVI文件大小有限制吗？
- 针对不同视频压缩标准的AVI是否需要制定扩展规范？
- …

流媒体的相关背景

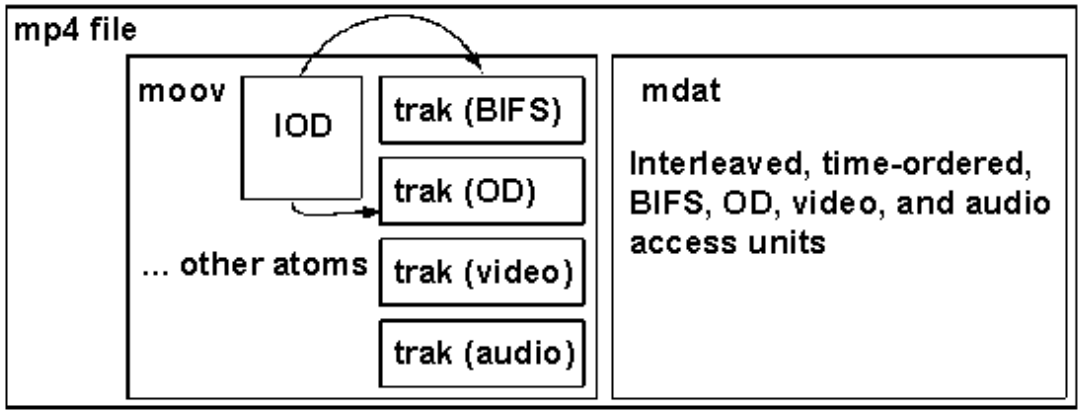


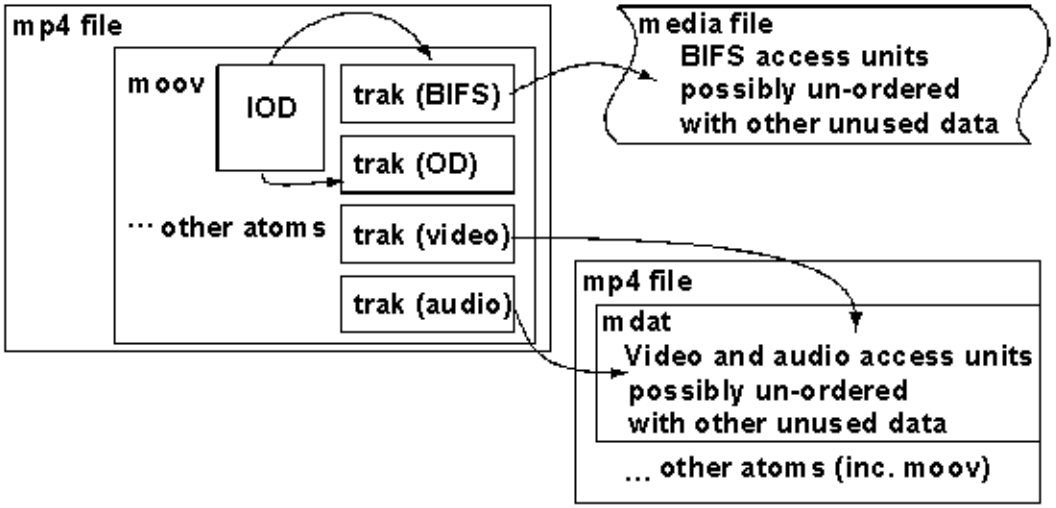
- RTMP是Real Time Messaging Protocol（实时消息传输协议）的缩写，是Adobe公司为Flash/AIR平台和服务器之间音、视频及数据传输开发的实时消息传送协议。RTMP协议基于TCP。
- Http Live Streaming是由Apple公司定义的基于HTTP的流媒体实时传输协议。是将整个流分为多个小的文件来下载，每次只下载若干个。服务器端会将最新的直播数据生成新的小文件，客户端只要不停的按顺序播放从服务器获取到的文件，就实现了直播。

MP4文件格式

- 基于APPLE公司的QuickTime格式
- 采用面向对象的结构，称为Atom（原子）。通过唯一的标记和长度来标识每个原子。绝大多数原子描述的是分层的元数据。原子的集合包含在称为Movie atom的原子中。
- 媒体数据自身可以存放在任何位置，既可以在MP4文件中的一个或多个媒体数据原子中，也可以位于MP4文件外，通过URL（统一资源定位）来访问

MP4文件格式


mp4 file
moov
IOD
trak (BIFS)
trak (OD)
... other atoms
trak (video)
trak (audio)
mdat
Interleaved, time-ordered,
BIFS, OD, video, and audio
access units


mp4 file
moov
IOD
trak (BIFS)
trak (OD)
... other atoms
trak (video)
trak (audio)
media file
BIFS access units
possibly un-ordered
with other unused data
mp4 file
mdat
Video and audio access units
possibly un-ordered
with other unused data
... other atoms (inc. moov)

MP4文件格式

```
aligned(8)class Box(unsigned int(32)boxtype,optional unsigned int(8)
[16]extended  type){
    unsigned int(32)size;
    unsigned int(32)type=boxtype;
    if(size==1){
        unsigned int(64)largesize;
    }else if (size==0){
        //box extends to end of file
    }
    if(boxtype==‘uuid’){
        unsigned int(8)[16]usertype=extended_type;
    }
}
```

MP4文件格式

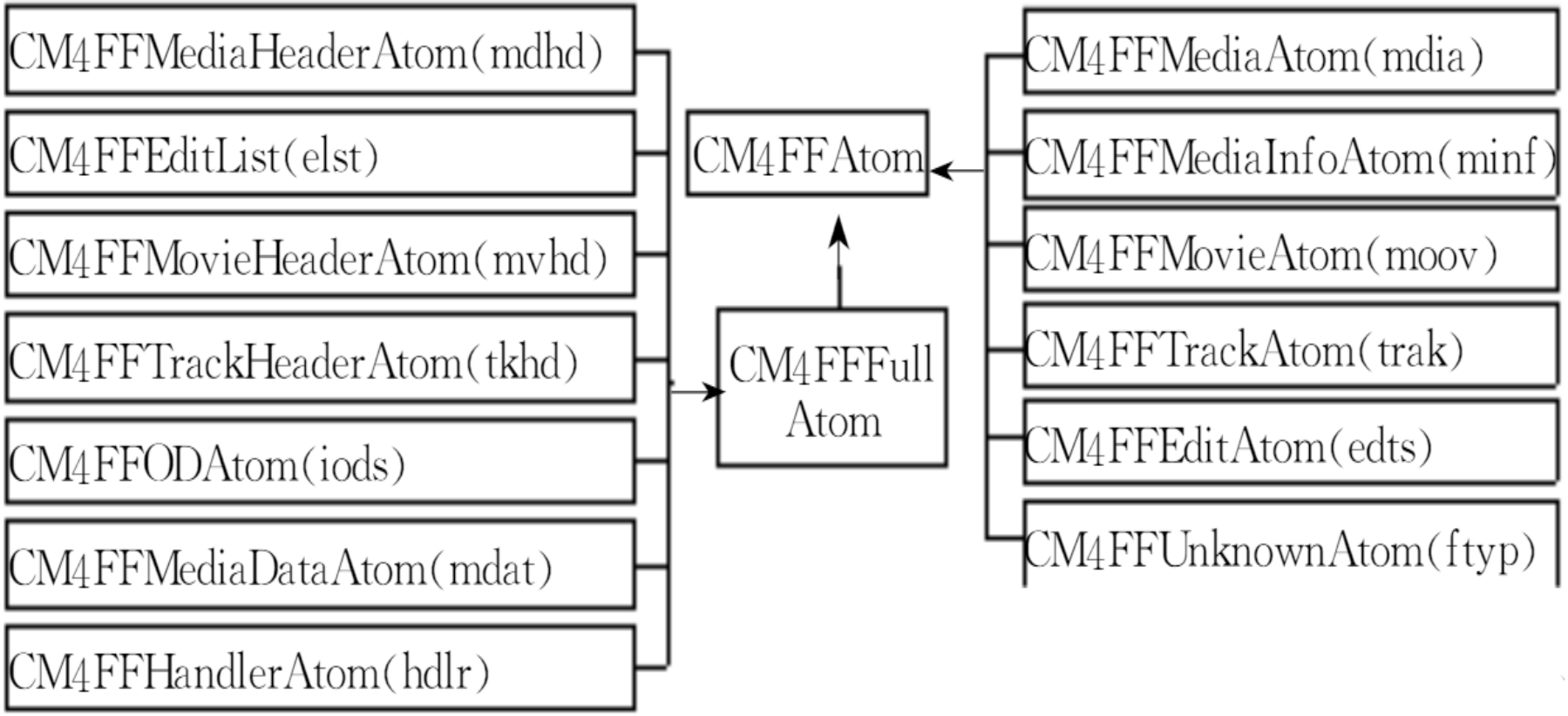


图3　部分原子类继承关系图

MP4 for streaming:

- 拉流时客户端在做什么？

得到每帧数据的显示时间，

得到该帧数据所对应的 chunk 的编号，

得到每个 chunk 相对文件的偏移地址，

得到每个 sample 在对应 chunk 内的偏移地址。

2.3 音频的特性与常用文件格式

2.3.1音频信号及其心理特征

1．声压（P）

声波引起某处媒质压强的变化量称为该处的声压。（单位为Pa（帕斯卡），即牛顿/米^{2}是压强的量纲）

也就是说：有声波时该处的压强值与没有声波时该处的压强值的差值

2.3.1音频信号及其心理特征

2．声压级（SPL）

将声压的有效值以对数的形式表示声音强弱的数值称为声压级。

$$SPL = 20\lg\frac{P_{rms}}{P_{ref}}$$ （单位用分贝dB）

P_{rms}----计量点的声压有效值

P_{ref}----零声级的参考声压值（$P_{ref}=2\times10^{-5}$帕）

a．P_{ref}为具有正常听力的年轻人对1 kHz的声音刚好能察觉的声压值。

b．声压级实际上是一种相对量，是某点的声压与零声压的比，是描述声音变化的动态范围的物理量

2.3.1音频信号及其心理特征

3. 音频与音高

音频是指声音信号的频率。人对于声音频率的感觉表现为音调的高低。

客观用频率表示声音的音高，单位为**Hz.**

主观感觉的音高$Mel=1000\log_2(1+f)$,测量音高时以**40dB**声强为基准。

音高与声音频率的关系大体上呈对数关系。

2.3.1音频信号及其心理特征

4．音频带宽　　20 Hz～20 kHz是人类的听觉频带

①人耳对不同频率的敏感程度有很大差别。

②中频段（2 kHz～4 kHz）最为敏感，幅度很低的信号都能被人耳听到。

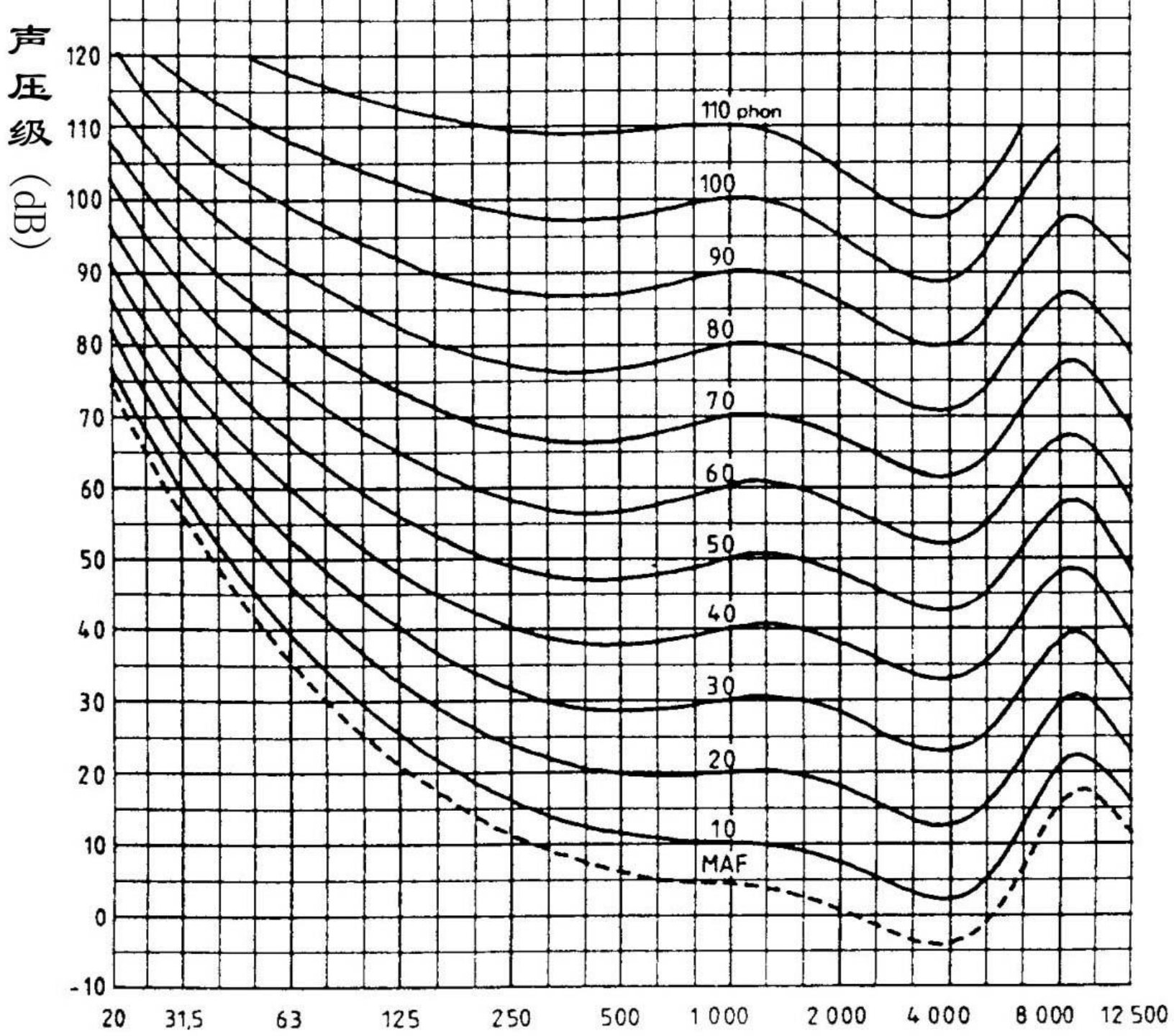
③低频区和高频区较不敏感，能被人耳听到的信号幅度比中频段要高得多。

2.3.1音频信号及其心理特征

5. 响度与响度级

响度是指人类所感受到声音大小的程度，而响度级则是以1 kHz信号的声压级数定义的响度的数值，单位是“方”（Phon）。

注：声压级是客观量，而响度级则是主观量。


声压级（dB）
120
110
100
90
80
70
60
50
40
30
20
10
0
-10
110 phon
100
90
80
70
60
50
40
30
20
10
MAF
20
31,5
63
125
250
500
1 000
2 000
4 000
8 000
12 500

2.3.1音频信号及其心理特征

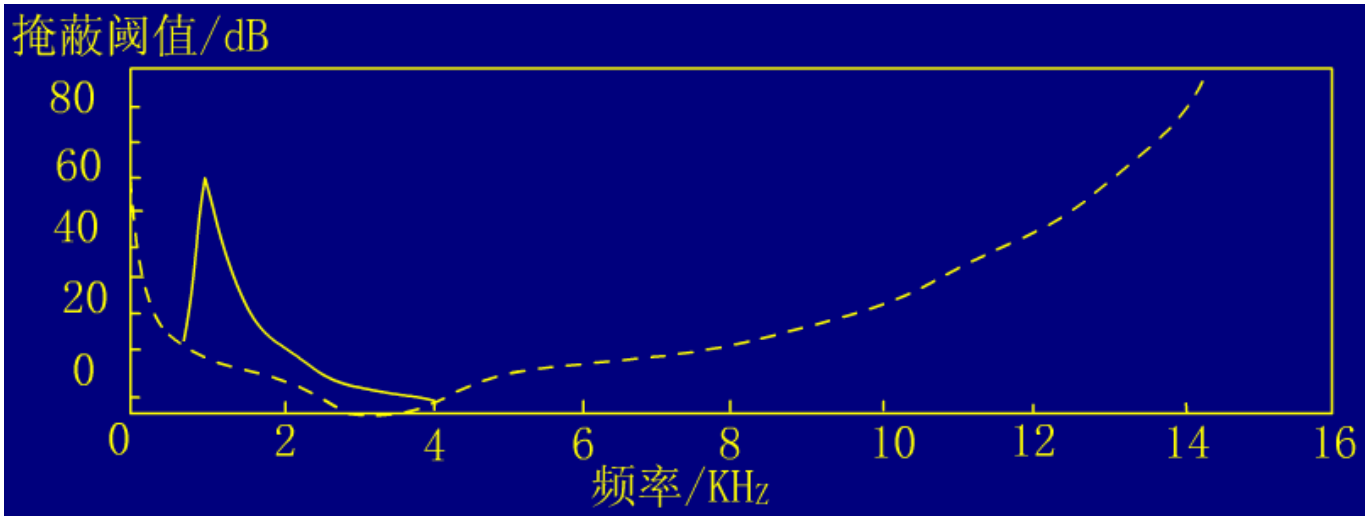
7．绝对听阈

在安静环境中，能被人耳听到的纯音的最小值(该曲线为0方响度级等响度曲线,即该曲线在1 kHz时声压级为0 db)

注：在绝对听阈曲线以下的各种声音将不能被人耳察觉。

2.3.1音频信号及其心理特征

掩蔽听阈：频域中的一个强音会掩蔽与之同时发声的附近的弱音。



注:①某一频率强音的存在会改变其附近的绝对听阈曲线,而改变部分称为掩蔽听阈曲线。

②改变后的听阈曲线以下的各种声音将不能被人耳察觉。

③某一频率的强音对较高频率弱音的影响大于较低频率。

2.3.1音频信号及其心理特征

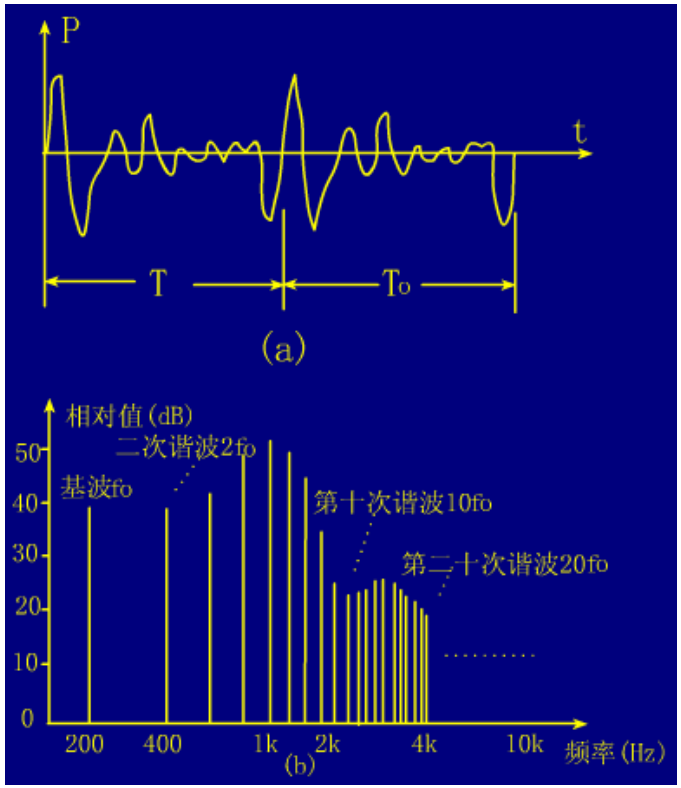
8．动态范围

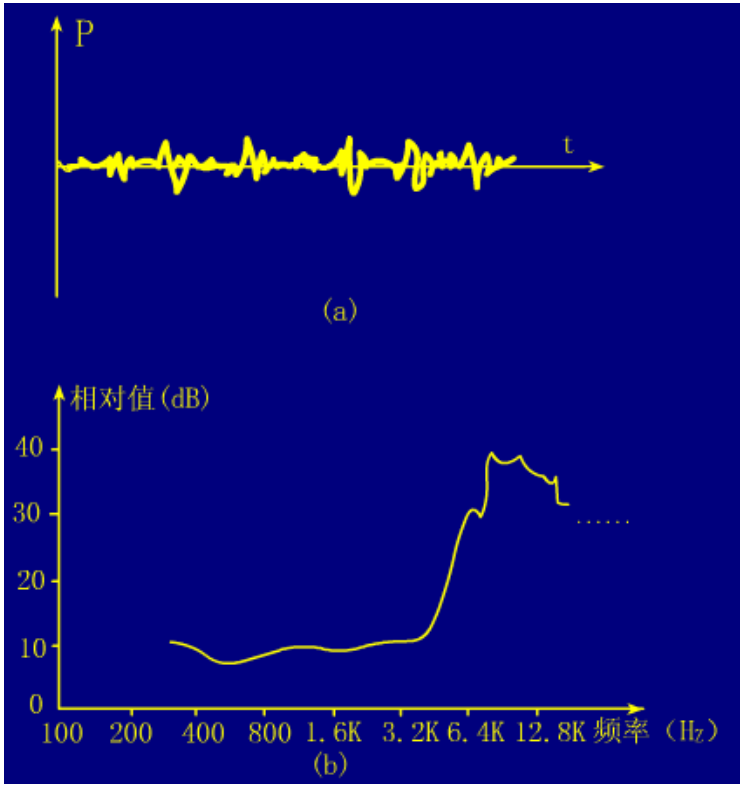
是某个声音的最强音与最弱音的强度差，用分贝表示。它是衡量声音强度变化的重要参数。

2.3.1音频信号及其心理特征

9．音频信号在时域和频域中的表现形式

在时域中表现为幅值随时间连续变化的曲线，在频域中则是将音频信号经傅里叶变换后在频率空间的分立或连续的谱线





2.3.2音频信号的数字化

1. 音频信号的动态范围

①实际声场中声音强弱的变化达120 dB。

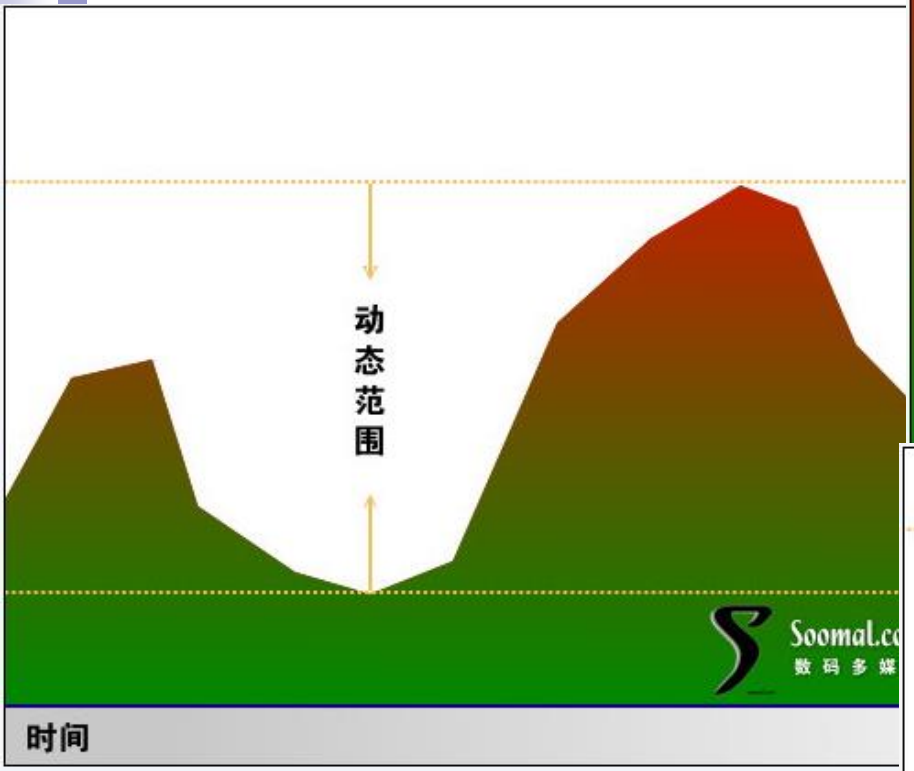
②传统的模拟音响设备的动态范围：采用模拟信号处理方式记录和重放音频信号。比如：磁带录音机等，其动态范围不会超过60 dB。

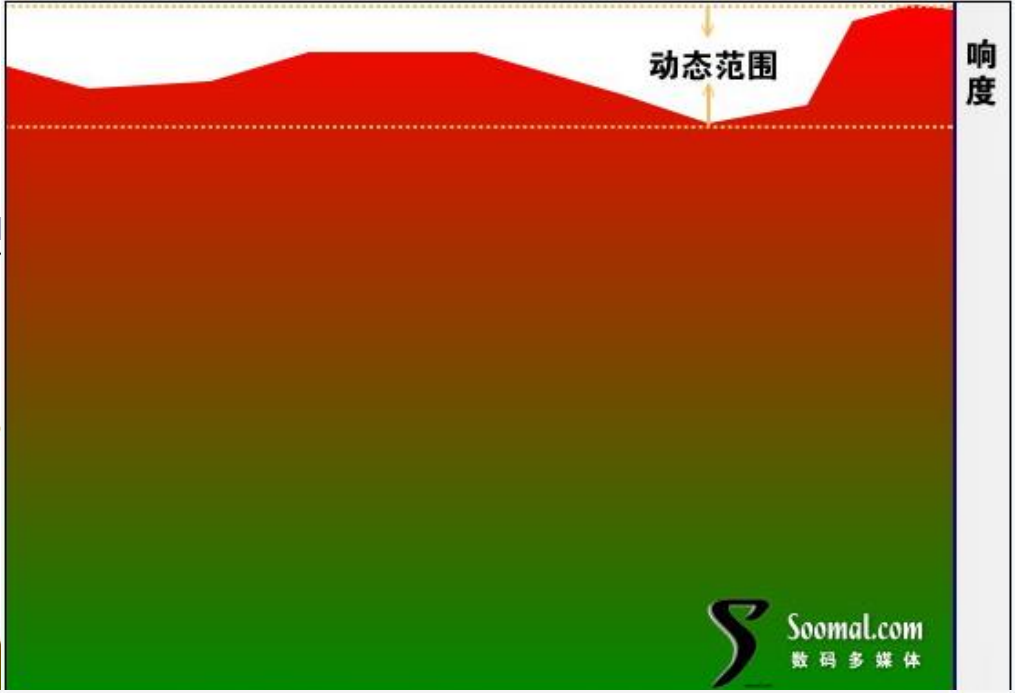
③数字音响设备的动态范围。

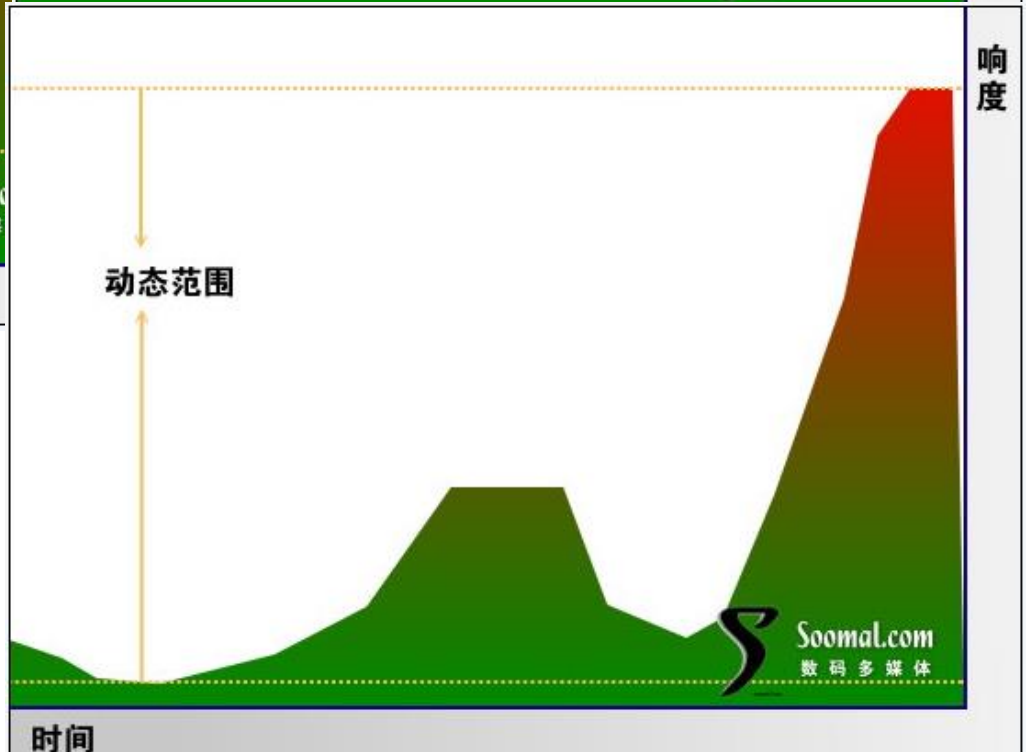
2.3.2音频信号的数字化

采用16 bit量化(如16位声卡)，则声音的强弱范围就可划分成：$20 \lg 2^{16} = 96$ (dB)个等级，因而动态范围较大。

结论：数字系统的音频信号动态范围比模拟系统提高了近一倍。这也是CD技术之所以获得高水准的音质的重要原因。


动态范围
时间


动态范围
响度
Soomal.com
数码多媒体


动态范围
响度
Soomal.com
数码多媒体
时间

2.3.2音频信号的数字化

2. 噪声容限 （对噪声的承受能力）

①传统的音响设备的失真情况

在重放时，由于失真、噪声、电机转速不匀等原因，重放效果大打折扣。

②数字音响设备的失真情况

数字系统只要能识别码的长短或脉冲的有无，即可再现出原来的信号。

结论：数字信号的噪声容限比较高

2.3.2音频信号的数字化

3. 采样定理及音频采样频率标准

采样定理

采样频率f_s必须高于被采样信号所含最高频率的两倍。

该定理指出：当对连续变化的信号波形进行采样时，若采样频率f_s高于该信号所含最高频率的两倍，那么可以由采样值通过插补技术正确地恢复原信号的波形，否则将会引起频谱混叠产生混叠噪声,而重叠的部分是不能恢复。这一定理不仅适用于模拟音频信号，也同样适用于模拟视频信号的采样。

2.3.2音频信号的数字化

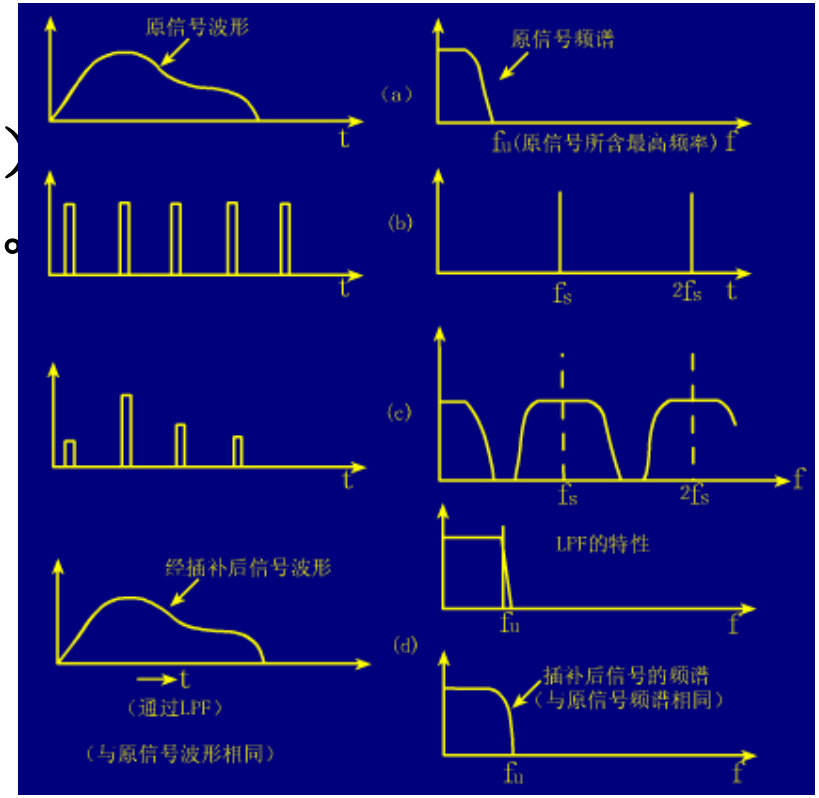
3. 采样定理及音频采样频率标准

被采样后的信号可恢复的原因

设有一音频信号$f(t)$,图（a）是对其在时域与频域中的描述。

设有一采样信号,图（b）其频谱为一个频率为nf_s的波列

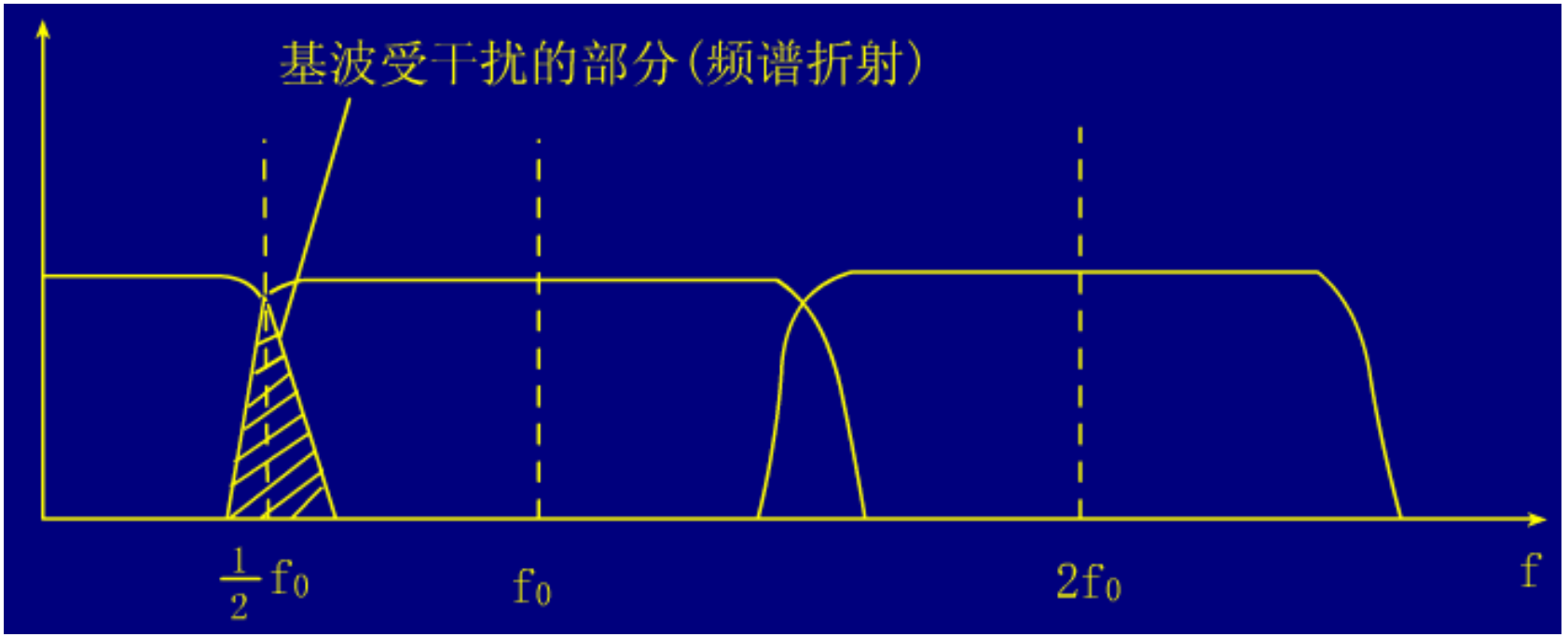
图（c）为采样后的波形与频谱。信号可恢复的原因：原信号的频谱完好保留，可以通过插补技术将原信号恢复



2.3.2音频信号的数字化

3. 采样定理及音频采样频率标准

如f_s低于信号中最高频率的两倍，将出现频谱混叠，原信号的频谱与下边带无法分开，破坏了原信号的频谱，原信号将无法恢复。



2.3.2音频信号的数字化

3. 采样定理及音频采样频率标准

音频信号的采样频率标准

选用了44.1 kHz作为CD声音的采样标准

音频信号频率上限为20 kHz，故采样信号频率f_s应大于40kHz以上，考虑到LPF在20kHz处大约衰减10%，为全频带高质量的还原，可以用22kHz的两倍频率作为音频信号的采样频率，但又为了能与电视信号同步，PAL制场频为50 Hz，NTSC制场频为60 Hz，所以取二者的整倍数，则选用了44.1 kHz作为CD声音的采样标准。

2.3.2音频信号的数字化

4. 量化

（1）量化过程

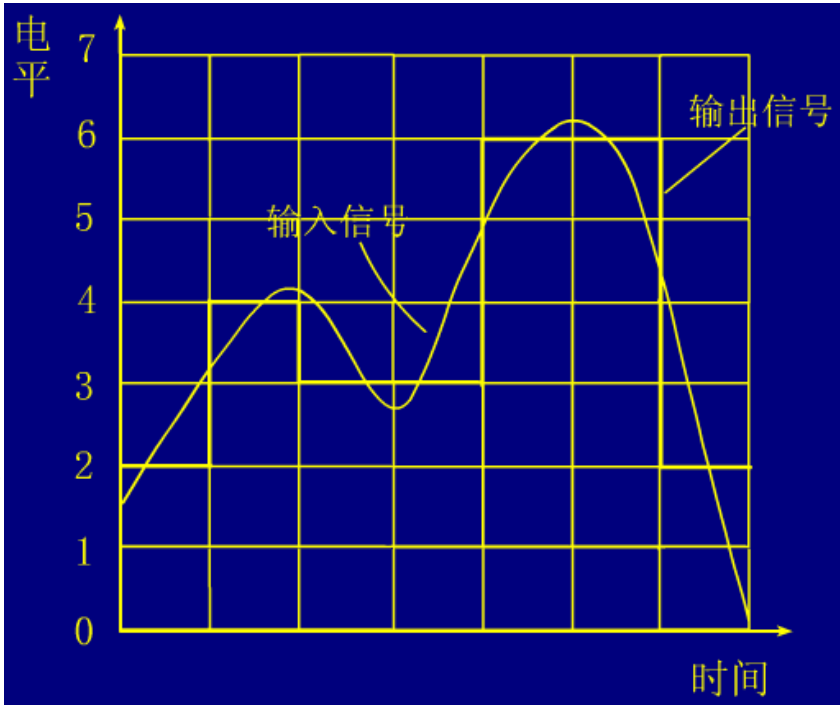
对非整数的采样值整数化（四舍五入）的过程

(2) 量化级

对满幅度信号所取的量化份数为量化级

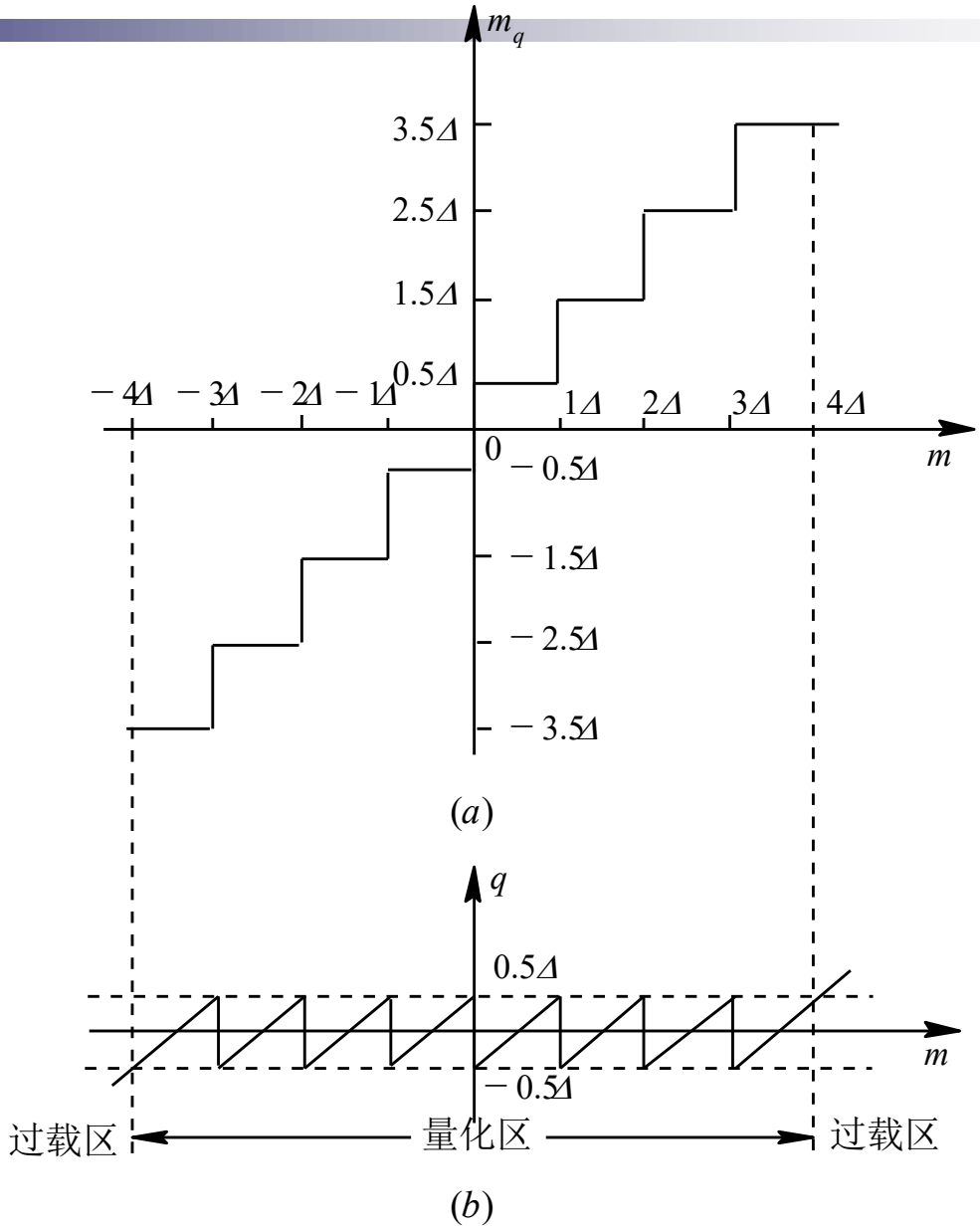
(3) 量化级差

量化分度的最小单位称为量化级差，用△表示



2.3.2音频信号的数字化

4. 量化



2.3.2音频信号的数字化

4. 量化

(4) 量化误差（量化噪声）

由四舍五入所引起的输入信号样值与量化后输出值的差，叫做量化误差，也称为量化噪声(N)。

由于量化值是在对应量化级内四舍五入得到的，所以量化误差应不大于：　|N|＜△/2

2.3.2音频信号的数字化

4. 量化

(5) 量化信噪比 $\frac{S}{N_q}=\frac{\sigma_x^{\ 2}}{E[(m-m_q)^2]}$

E表示求统计平均，*S*为信号平均功率，N_q为量化噪声平均功率。显然，（S/N_q）越大，量化性能越好

设输入的模拟信号*m*(*t*)是均值为零，概率密度为*f*(*x*)的平稳随机过程，*m*的取值范围为(*a*, *b*)，且设不会出现过载量化，则量化噪声功率N_q为

$$Nq=E\left[(m-m_q)^2\right]=\int_a^b(x-m_q)^2 f(x)dx$$

2.3.2音频信号的数字化

4. 量化

一般来说， 量化电平数M很大，量化间隔Δ很小，因而可认为在Δ内不变，以p_i表示，且假设各层之间量化噪声相互独立，则N_q表示为

$$N_q = \sum_{i=1}^{M} p_i \int_{m_{i-1}}^{m_i} (x - q_i)^2 dx$$

$$= \frac{\Delta^2}{12} \sum_{i=1}^{M} p_i \Delta = \frac{\Delta^2}{12}$$

p_i代表第i个量化间隔的概率密度，Δ为均匀量化间隔， 因假设不出现过载现象，故上式中 $p_i\Delta=1$。 均匀量化器不过载量化噪声功率N_q仅与Δ有关，而与信号的统计特性无关，一旦量化间隔Δ给定，无论抽样值大小，均匀量化噪声功率N_q都是相同的

2.3.2音频信号的数字化

4. 量化

$$\frac{S}{N_q}=\frac{\sigma_x^{\ 2}}{E[(m-m_q)^2]}$$

$$S=\int_{-a}^{a}x^2 f(x)dx=\int_{-a}^{a}x^2\frac{1}{2a}dx=\frac{a^2}{3}=\frac{M^2\Delta^2}{12}$$

$$\frac{\mathrm{S}}{\mathrm{N_q}}=\mathrm{M}^2 \qquad \left(\frac{S}{N_q}\right)_{dB}=10\lg\left(\frac{S}{N_q}\right)=20\lg M$$

2.3.2音频信号的数字化

4. 量化

(6) 音频信号的量化位数

CD和VCD中的音频采用16 bit量化器

(7) 音频码率

码率:为单位时间内传输的数据bit数

音频码率： $R=44.1\times10^3\times16\times2=1.41\times10^6$ b/s

(8) 均匀量化

无论信号大小，都采用同样的量化级差Δ的方法

2.3.2音频信号的数字化

4. 量化

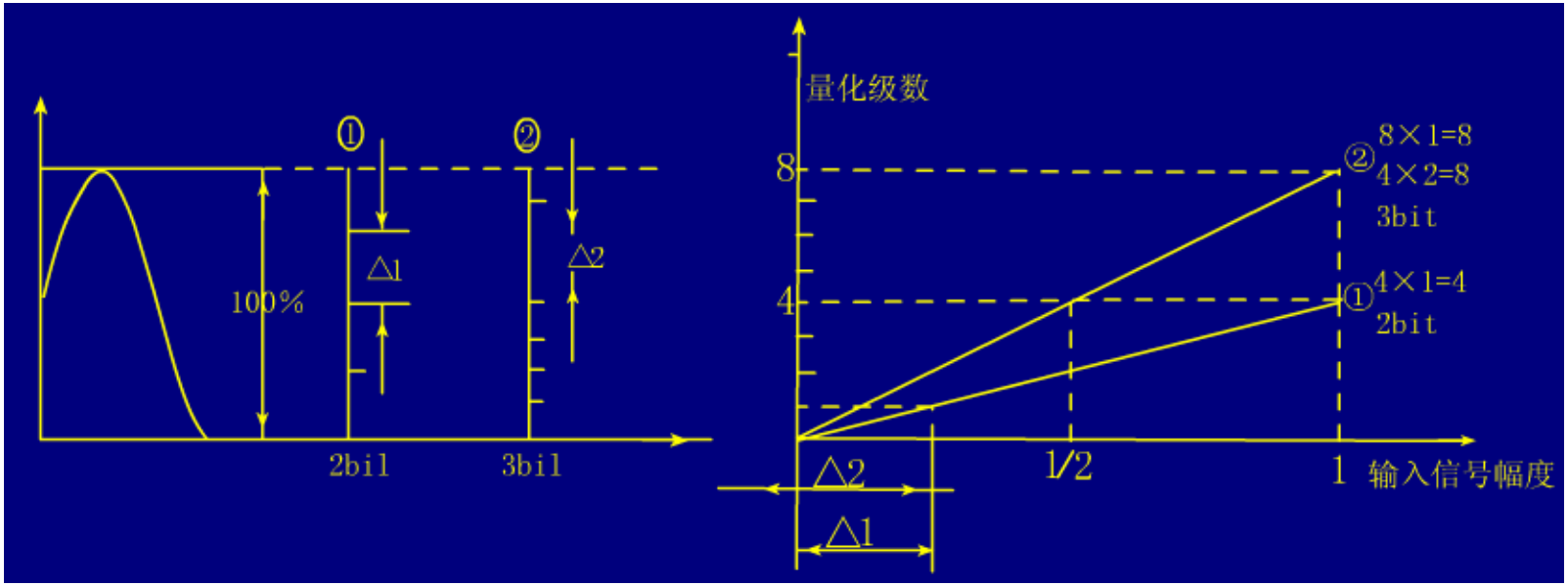
（9） 非均匀量化

对微小信号采用细量化（Δ小），对大幅度信号采用粗量化（Δ大）的方法

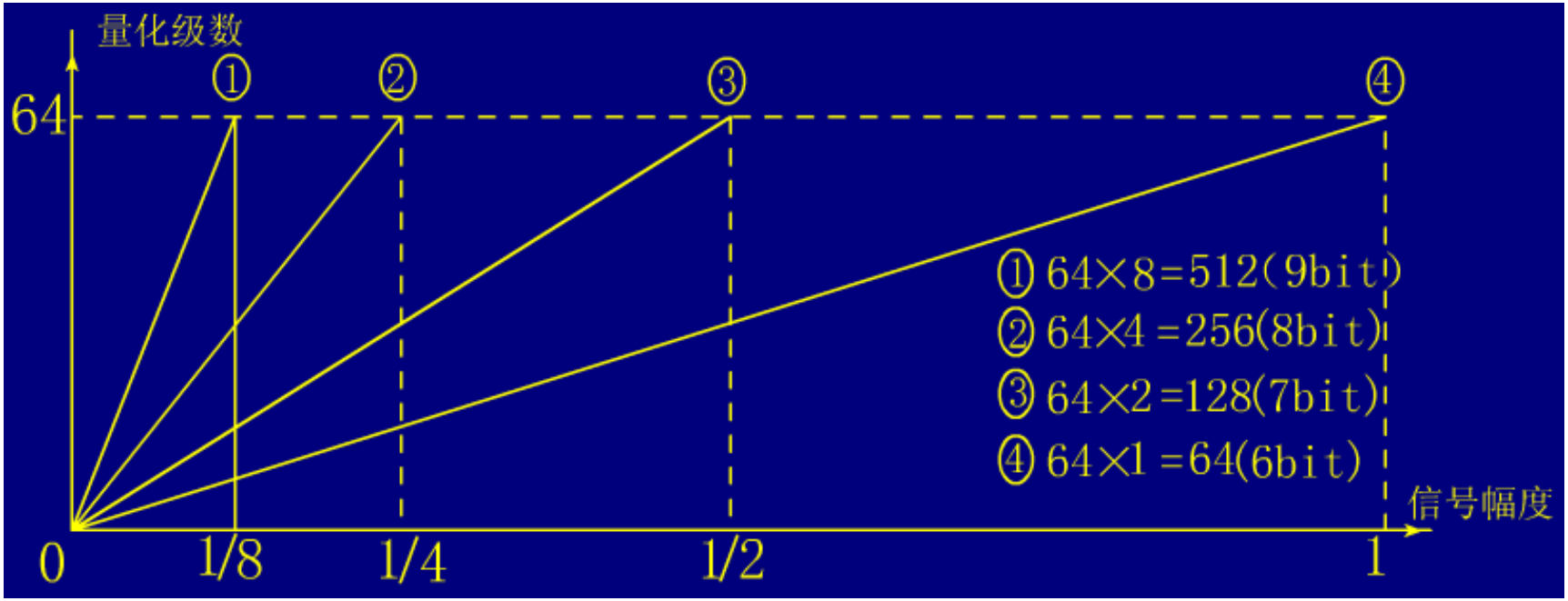
图（a）为幅值为1的信号进行2bit、3bit的均匀量化的情况，其中Δ不同

图（b）改变了坐标☆同样对满幅度量化，量化的级数越多对应斜线的斜率就越大，量化就越细

反之对同一量化级数，其描述的幅值越小，斜率越大，量化就越细



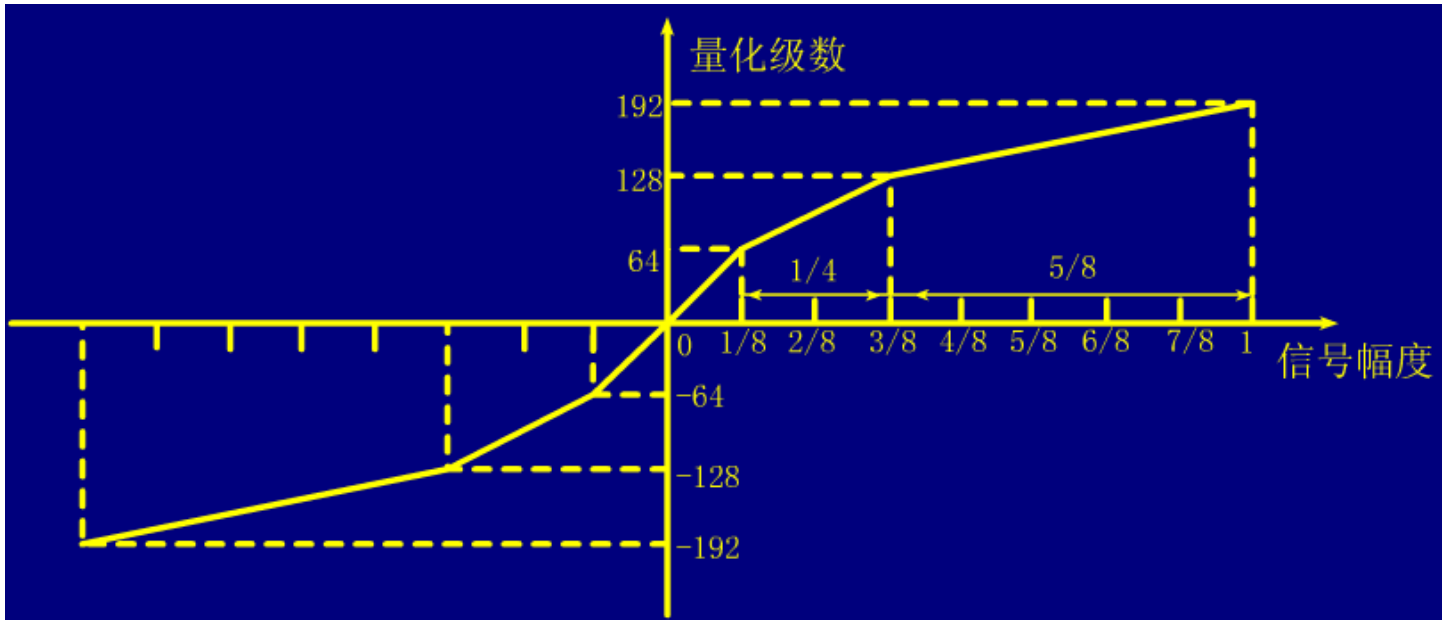
用同样的bit数去描述大小不同的信号，则可达到对小信号细量化，而对大信号粗量化的目的



2.3.2音频信号的数字化

4. 量化 (10) 非均匀量化的折线法

下图是一个对双极性信号进行5折线的量化图。



图中每段64个量化级共6段，所以总共应有6×64=384级，约为$2^{8.5}$，对于同样的动态范围，采用非均匀量化在可听见的声音强度范围提供了更高的信噪比。

2.3.2音频信号的数字化

5. 编码 编码就是把已经量化后的采样值用二进制数码表示出来

（1）自然码

无符号的二进制代码

（2） 符号 – 数值码

在自然码最高位（MSB）增加了一个符号位而构成，用以表示数值的正负，一般用“0”表示正，而用“1”表示负，一般用来表示双极性信号。

（3）2的补码

2.3.2音频信号的数字化

5. 编码

(4) 格雷码

每增加1个数值时只有一个码元变化的码

000	001	011	010	110	111	101	100	格雷码
0	1	2	3	4	5	6	7	十进制数

多媒体文件的设计思想和理解

- 该容器中数据是如何组织的？
- 该容器包含哪些编码格式的数据？这些数据是如何存储的？
- 该容器包含哪些元数据信息？包含哪些节目信息？
- 对于支持多节目的容器格式，如何找到对应的音频流、视频流、字幕流？
- 如何确定该容器的节目播放时长？
- 如何从该容器中提取音频、视频、字幕数据，并交给解码器解码，有时间戳否？通过什么方式保证同步？
- 该容器是否支持seek？有哪些辅助信息？
- 是否支持**直接流化**（streaming）？
- 哪里可以找到该容器格式最标准的文档资料？
- 有哪些可用的工具，方便分析容器格式异常或者错误？

作业：分析一个图像格式文件

- TGA、Tiff、PNG，……任选一个
- 设计需要回答的问题（体现多媒体文件的设计思想）
- 回答这些问题
- 整理成文档上传博客

多媒体文件处理小结-BMP文件

- 两个结构体
  - 位图文件头**BITMAPFILEHEADER**
  - 位图信息头BITMAPINFOHEADER
- 一个结构体数组
  - **struct tagRGBQUAD *rgbquad**
- 实际的位图数据
  - unsigned __int8 *bmpBuf

多媒体文件处理小结-BMP文件

- 顺序的程序结构
  - 打开位图文件
  - 解析位图文件头
  - 解析位图信息头
  - 构造调色板
  - 取出RGB数据或根据调色板构造RGB数据

- 很方便地扩展到**C++**，例如写一个对**bmp**文件读写和显示的类
- class CXBitmap
  {
  private:
  BITMAPFILEHEADER m_fileHead;
  BITMAPINFOHEADER m_infoHead;
  HGLOBAL m_hInfoHead;
  LPBITMAPINFOHEADER m_pInfoHead;
  CPalette m_pal;
  BYTE* m_bitmap;
  int m_nColors;
  int m_bitNum;
  public:
  CXBitmap();
  virtual ~CXBitmap();
  BOOL Read(LPCTSTR path);
  BOOL Show(CDC* pDC, int origin_x, int origin_y, int cx, int cy);
  int GetWidth();
  int GetHeight();

  };

```
class CFG_DIB : public CObject
{
public:
//默认构造函数
CFG_DIB();
//构造函数,根据图象宽和高,以及记录每个象素所需字节数来初始化
CFG_DIB(int width, int height, int nBitCounts);
virtual ~CFG_DIB();

public:
HB99vMAP m_hBitmap;
LPBYTE m_lpDIBits; //DIB位的起始位置
LPB99vMAPINFOHEADER m_lpBMPHdr; //B99vMAPINFOHEADER信息
LPVOID m_lpvColorTable; //颜色表信息
HPALETTE m_hPalette; //调色板

private:
DWORD m_dwImageSize; //非B99vMAPINFOHEADER或
B99vMAPFILEHEADER的位
int m_nColorEntries; //颜色表项的个数

//显示参数
public:
CPoint m_Dest; //目的矩形域的左上角坐标
CSize m_DestSize; //显示矩形的宽度和高度
CPoint m_Src; //原矩形左下角坐标
CSize m_SrcSize; //原矩形宽度和高度

public:
void InitDestroy(); //初始化变量
void ComputePaletteSize(int nBitCounts); //计算调色板大小
void ComputeImage(); //计算图象大小
```

- BOOL ReadFile(CFile* pFile); //从BMP文件中读入DIB信息
  //从BMP文件中读入DIB信息,与ReadFile不同的是使用CreateSection创建位图位
  BOOL ReadSection(CFile* pFile, CDC* pDC = NULL);
  //将DIB写入文件，保存成BMP图片格式
  BOOL WriteFile(CFile* pFile);
  //创建新的位图文件，根据参数width,height,nBitCounts分配内存空间
  BOOL NewFile(int width, int height, int nBitCounts);
  BOOL CloseFile(); //关闭位图文件

  BOOL Display(CDC* pDC); //显示位图

  HB99vMAP CreateBitmap(CDC* pDC); //用DIB创建DDB
  HB99vMAP CreateSection(CDC* pDC = NULL); //创建位图位数据，即象素数据
  //如果DIB没有颜色表，可以用逻辑调色板
  BOOL SetLogPalette(CDC* pDC);
  //如果DIB有颜色表，可以创建系统调色板
  BOOL SetWinPalette();
  //把DIB对象的逻辑调色板选进设备环境里，然后实现调色板
  UINT UseLogPalette(CDC* pDC);

  int GetHeaderSize() //得到BitmapInfoHeader的大小，包含颜色表数据
  {
  return sizeof(B99vMAPINFOHEADER) + sizeof(RGBQUAD) * m_nColorEntries;
  }
  int GetHeight() //得到图像的高度
  {
  if(m_lpBMPHdr == NULL) return 0;
  return m_lpBMPHdr->biHeight;
  }
  int GetWidth() //得到图像的宽度
   {
  if(m_lpBMPHdr == NULL) return 0;
  return m_lpBMPHdr->biWidth;
  }