第一章 绪论

1、什么是数据压缩

数据压缩，就是以最少的数码表示信源所发的信号，减少容纳给定消息集合或数据采样集合的信号空间。

所谓信源空间即被压缩对象，是指：物理空间、时间区间、电磁频段。

2、为什么要进行数据压缩

数据压缩的好处：

（课本）

①较快地传输各种信源——时间域的压缩

②在现有通信干线上开通更多的并行业务——频率域的压缩

③降低发射功率——能量域的压缩

④紧缩数据存储容量——空间域的压缩

（老师PPT）

①减少存储空间——文件压缩、图像压缩、语音压缩、视频压缩……

②减少传输时间

③渐进传输——有些编码算法允许先发送重要的比特，使得我们在看到更逼真的内容之前，能先看到更低分辨率的版本

④降低计算——用较少的数据得到近似的结果

3、为什么可以数据压缩

自然界中的大多数数据都是冗余的：任何非随机选择的数据都有一定结构，可利用这种结构得到数据的更小表示。

例如：统计冗余（概率不均等、文本冗余、图像冗余）、数据的物理产生过程（如利用人类的发声系统，设计语音压缩算法）、感知冗余（听觉冗余、视觉冗余）

4、怎样进行数据压缩

利用统计冗余，引入可接受的偏差

数据压缩的一般步骤



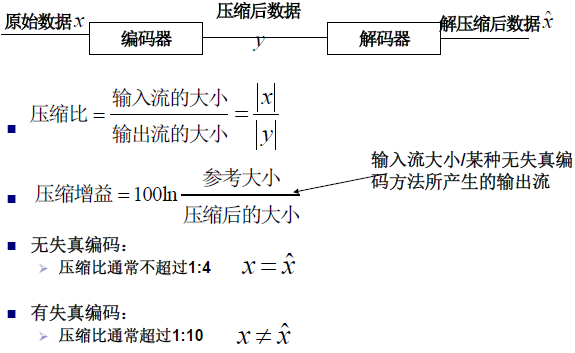
①建立一个数学模型，以便更有效地重新表达原始数据

②对模型参数进行量化（二次量化，对模拟信号进行的量化称为一次量化）

③更紧凑地表示量化后的模型参数，此时的编码要求能“忠实地”再现模型参数的量化符号（熵编码）

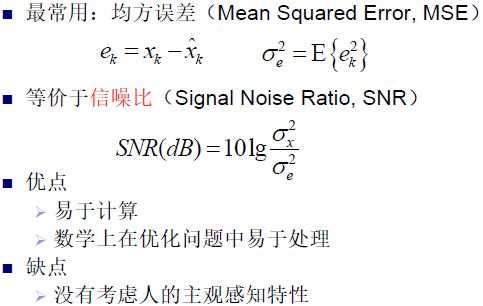
5、编码系统的性能比较

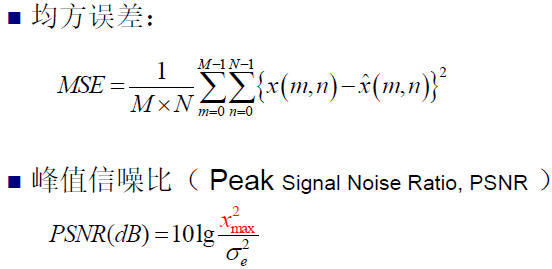
效率：压缩比



质量：对有失真编码而言

客观质量：失真度量





主观质量

复杂度

算法的运算量和存储量

编码器和解码器的复杂度：对称/非对称

时延

5、本课程的主要内容

第一部分 视音频媒体的特性与表示

图像的属性和常见文件格式，视频信号的数字化及相关标准，AVI等常见的视频文件格式，音频的特性和常见文件格式

第二部分 无失真压缩的理论基础

信息论基础：信源的建模与分析

变长码的编码基础——无失真信源编码定理

编码的基本途径

第三部分 统计编码方法

Huffman编码，Golomb编码和通用变长码，算数编码，游程编码

词典编码（LZ77、LZ78、LZW算法，词典编码在文件压缩和图像压缩中的应用）

第四部分 有失真压缩的理论基础

失真的度量，率失真理论简介

有失真压缩中常用的模型：概率模型和线性系统模型

第五部分 有失真信源编码（熵压缩）

量化

基于变换的编码、JPEG标准

预测编码

子带编码、小波编码、JPEG2000

第六部分 信源编码标准及应用

视频编码标准（MPEG-2、MPEG-4、H.264、AVS）

音频编码标准（MP2、MP3、G.721、G.729、……）

应用——码率控制、转码、传输

第二章 媒体的特性与表示

1、图形与图像属性及常用文件格式

1）视觉系统对颜色的感知

从人的主观感觉角度，颜色包含三个要素：色调、饱和度、明亮度

混合色的色调饱和度由三基色的混合比例决定，亮度是三基色亮度之和

2）图像的颜色模型（彩色空间color space）

颜色模型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、CMYK 色、亮分离颜色空间：HSL、YUV

3）图像的基本属性

①分辨率：

显示分辨率、图像分辨率

②像素深度：

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。

③真彩色、伪彩色、直接色：

真彩色：在组成一幅在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：每个像素的颜色不是由每个基色分量的数值直接决定，而是把像素值当做彩色查找表（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：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

4）图像的分类

①矢量图：计算生成，显示精度高，操作灵活性大。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

②点位图：适合表现细致、色彩和层次比较丰富的图像。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。

③灰度图

④彩色图

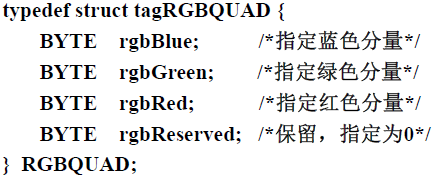
5）常见的图像格式

①BMP









紧跟在调色板之后的是图像数据字节阵列，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须是4的整数倍，也就是DWORD对齐的。扫描行是由底向上存储的，阵列中的第一个字节表示位图左下角的像素，最后一个字节表示位图右上角的像素。

②GIF

③JPG

2、视频的特性及常用文件格式

1）视频信号的数字化及标准

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分量信号数字化的方式）

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

并行数据传输：R=13.5MHz+6.75MHz×2=27Mb/s

①亮度信号的取样频率：13.5MHz

按照奈奎斯特取样定理，取样频率至少为信号上线频率的两倍

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信号，取样频率应大于13.2MHz

为了获取正交结构，取样频率必须是行频的整数倍

为了使两种扫描制式实现兼容，应采用同一取样频率，625行频为12.625Hz，525行频为15.734Hz，两者最小公倍数2.25MHz

②色差信号的取样频率：6.75MHz

③视频信号量化位数：8bit

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

PAL制的每行取样点数为13500kHz×64μs=864

NTSC制的每行取样点数为13500kHz×63.55μs=858

有效水平取样点：为了使两种制式兼容，有效部分取53.3μs，亮度信号53.3μs÷0.074μs=720（T），色差信号360（T）

有效垂直取样点：PAL每帧576行，每场288行。NTSC每帧480行，每场240行

2）AVI文件解析

3、音频的特性及常用文件格式

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。

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。

③音频：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。音高与声音频率的关系大体上呈对数关系。

④音频带宽：20Hz~20kHz是人类的听觉频带。人耳对不同频率的敏感程度有很大差别，中频段2kHz~4kHz最为敏感，幅度很低的信号都能被人耳听到，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。

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1kHz信号的声压级数定义的响度的值，单位是“方”（Phon）

⑥绝对听阈：在安静环境中，能被人耳听到的纯音的最小值。

⑦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。

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。

⑨音频信号在时域和频域中的表现形式：在时域中表现为赋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离或连续的谱线

2）音频信号的数字化

①音频信号的动态范围

②噪声容限

③采样定理及音频采样频率标准

采样频率fs必须高于被采样信号所含最高频率的两倍

④量化

⑤编码

第三章 无失真数据压缩

1、数据压缩的理论极限与基本途径

1）离散无记忆信源及其压缩基本途径

非等概率→统计匹配：统计编码  
 2）联合信源及其压缩基本途径

不独立的各分量→独立的各分量：彩色空间转换，变换编码

3）随机序列信源及其压缩基本途径

利用条件概率：预测编码

利用联合概率：对多个符号合并编码，如矢量量化

将非平稳信源转换为平稳信源：分析/综合编码，分成多个平稳子带

2、统计编码方法

1）霍夫曼编码

①递归Huffman算法

如果只有一个符号，则有一个节点的树是最佳的，否则找到两个概率最小的符号，其概率分别为p和q，用一个概率为p+q的新符号代替上述两个符号，用递归法对新的符号集求解，用一个有两个旧符号孩子的中间节点表示新符号。

②霍夫曼编码的实现过程

先从指定文件中读取数据，统计每个符号发生的概率，并建立相应的树叶节点。然后构建霍夫曼树及霍夫曼码，按字符概率由小到大将对应节点排序，得到文件出现的字符种类数，继而构建霍夫曼树，对码树编码。

③霍夫曼解码

比特串行解码、基于查找表的解码

④规范霍夫曼编码

只要符合是及时码、某一字符编码长度和使用二叉树建立的该字符的编码长度相同这两个条件的编码都可以叫做Huffman编码

规范霍夫曼编码的提出：

小量数据的压缩，霍夫曼码表的存储是问题

使用某些强制的约定，仅通过很少的数据便能重构处Huffman编码树的结构

编码步骤：

统计每个符号的频率并求出该符号所需的位数/编码长度

统计从最大编码长度到1的每个长度对应多少个符号，然后为每个符号递增分配编码

编码输出压缩信息

2）算术编码

①基本思路

算术编码是从另一种角度对很长的信源符号序列进行有效编码的方法，是一种非分组码。

它不是将单个的信源符号映射成一个码字，而是将整个输入符号序列映射为实轴[0,1)内的一个小区间，其长度等于该序列的频率；再在该小区间内选择一个代表性的二进制小数作为编码输出，从而达到了高效编码的目的。

②编码方法

将符号序列的累计函数写成二进位的小数，取小数点后L位，若后面还有尾数，就进到第L位。令L满足L=log[1/P(S)]

③自适应算术编码

3）词典编码

①基本思路

如果用一些简单的代号代替这些字符串，就可以实现压缩。字符串与代号的对应表就是词典。

②第一类词典编码

第一类词典法的想法是企图查找正在压缩的字符序列是否在以前输入的数据中出现过，然后用已经出现过的字符串替代重复的部分，其输出代号定义为指向早期出现过的字符串的“指针”。

LZ77“滑动窗口压缩”

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。

LZSS

如果匹配串的长度比指针本身的长度长就输出指针，否则就输出真实字符。需要输出额外的标志位区分是 指针还是字符。

③第二类词典编码

第二类算法的想法是企图从输入的数据中创建一个“短语词典”，这种短语可以是任意字符的组合。编码数据过程中当遇到已经在词典中出现的“短语”时，编码器就输出这个词典中的短语的“索引号”，而不是短语本身。

LZW

只输出代表词典中的字符串的码字。在开始时词典不能是空的，它必须包含可能在字符流出现中的所有单个字符。即在编码匹配时，至少可以在词典中找到长度为1的匹配串。

4）游程编码

①游程编码的基本思路

游程长度是指字符（或信号采样值）构成的数据流中各字符重复出现而形成字符串的长度。如果给出了形成串的数据字符X、串的长度RL及串的位置，就能恢复原来的数据流。游程长度编码就是用二进制码字给出上述信息的一类方法。

基本的RLC方法最初需要加一个“异字头”前缀，因而低效且不使用。但是，对于二值图像和连续色调图像，该前缀可以省去。因此，改进的RLC在图像编码中得到了广泛的应用。

②二值图像的游程编码

二值图像是指仅有黑（用“1”代表）、白（用“0”代表）两个亮度值的图像。

编码思路：

分别对“黑”、“白”的不同游程长度进行Huffman编码，形成黑、白两张码表，编译码通过查表进行。

编码规则：

对0~63的黑白游程，用结尾码表示

对64~1728的黑白游程，用构造码+结尾码表示

规定每行都从白游程开始，若实际扫描行由黑开始，则需在行首加零长度白游程。每行结束要加行同步码EOL

（构造码、结尾码查表）

5）Golomb编码

①提出的背景

变长码对差错极其敏感，即时码的特性导致1位出错就可能使解码器失去同步

解决思路：

基于某个预先假定的概率模型设计出最佳变长码，码字具有某种固定的结构。癌牺牲一定压缩效率的前提下，提高解码器的可靠性。

②一元码

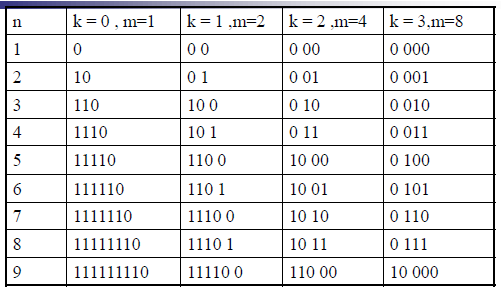
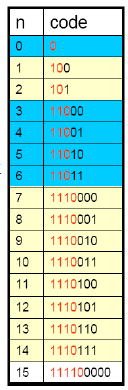
对非负整数n，用n个1和1个0表示，或者用n个0和1个1表示。为即时码

③Golomb编码

适用对象：服从几何分布的正整数数据流，数据流中整数n出现的概率为P（n）=（1-p）n-1p

使用一个可调参数b将输出码字分为若干个组，每组内均有码长相近的m个码字，每个码字有两个部分组成，前缀码和尾码。前缀码是q+1位的一元码字，q=INT[n-1/b]。尾码是r=n-1-qb的二进制编码

b=2k时刻简化为G（k），例如



m较小，Golomb码初始很短，增长很快，适合大游程很少的情况

m较大，Golomb码初始很长，增长很慢，适合大游程较多的情况

④指数Golomb编码（EGC）

尾码增加1位，前缀码增加1位

可视为G（0）码加上q+m位尾码（m为阶数，q为1的个数），例如

第四章 量化

1、标量量化

1）量化器的描述

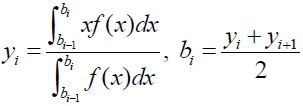
量化：用一个很小的集合表示一个大集合

量化区间的数目M，决策边界bi，重构水平yi，量化误差x-y



2）均匀量化

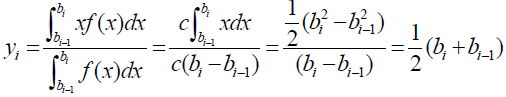
3）Lloyd-Max算法



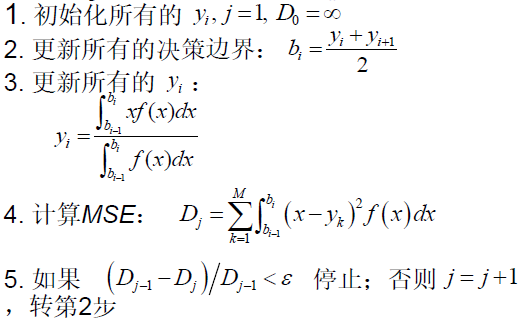
①门限（判决）电平应取在相邻量化输出电平的中点

②量化电平（重建电平）应取在量化间隔的质心上

当量化器的输入为均匀分布时，f(x)=c，Lloyd-Max量化器变为均匀量化器



迭代Lloyd-Mac算法（已知f（x））



4）熵约束量化（ECSQ）

Lloyd-Max量化器：

对索引用固定码率编码：log2M（R）比特

熵约束标量量化器：

对量化索引用变长码编码：

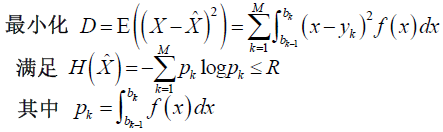
对量化索引用熵编码技术编码

平均码率~重构水平的熵≤log2M



比Llyod-Max量化器的性能更好

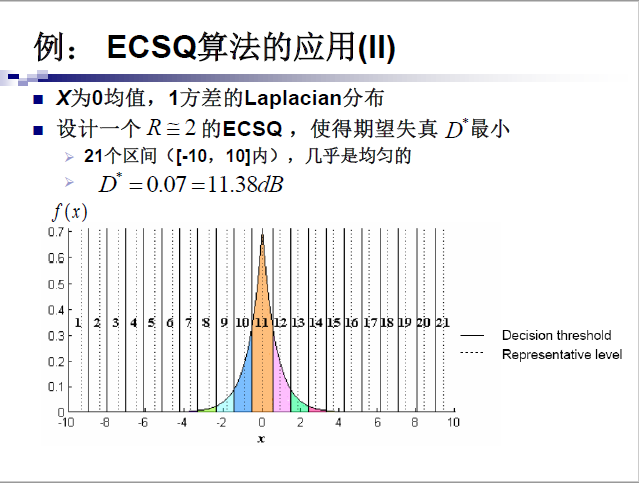
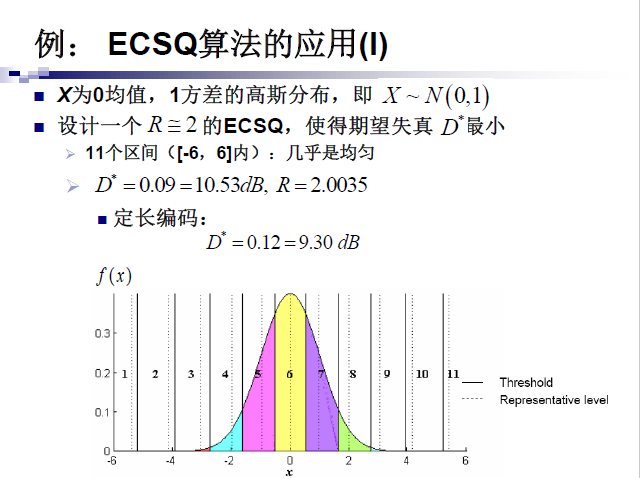
问题的形式化描述



用Lagrange费用函数



太复杂，不能直接求解，要用迭代法求解



5）Deadzone Midtread量化器

0附近的量化区间大小为其余量化区间的两倍，其他量化区间仍是均匀的

6）嵌入式量化器

动机：可伸缩解码

随着比特流的解码，渐进地精化重构数据

对低带宽连接有用

是JPEG2000的一个关键特征

嵌套量化：低码率器的区间被再分割，以产生更高码率的量化器

可以通过截断量化索引获得较粗燥的量化

7）对于已知概率模型及其数字特征的随机过程，比较容易根据概率分布安排量化器的决策边界，以得到最小量化失真的优化量化器。

如果概率分布是均匀的，则采用均匀量化比较理想。

对于分布概率模型未知的随机过程，其优化量化器的设计较为困难，可以采用Lloyd-Max算法来解决，但实现时还有一定困难，不宜硬件实现，执行时间也因初始值选取的不同而不同。

2、矢量量化（VQ）

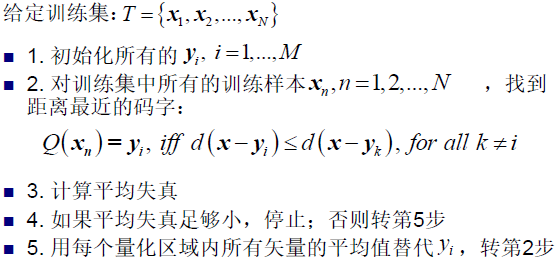
1）矢量量化的基本思想

量化时不是处理单个符号，而是一次处理一组符号（矢量）

假设块大小为a×b，码书中共有M个码字（码字也是长度为a×b的矢量），则码率R=logM/（a×b）bpp

2）LBG算法

将Lloyd算法推广到矢量量化，所以亦称为推广的Lloyd算法



优化过程中可能限于局部最小值

依赖于初始码书的选取

初始码书的选取

随机选择：重复多次，取失真最小的结果

分裂：从一个类开始，每次将失真最大/数量最多的类分裂成两个

合并：从N个类开始，每次将两个失真最小的类合并

码字中缺少结构

编码复杂性高：需要全搜索

存储要求高，码书指数增长

第五章 预测编码

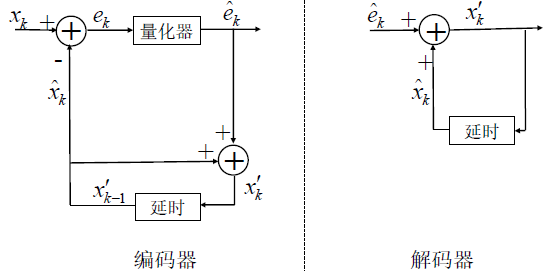
1、预测编码的基本原理

利用信源相邻符号之间的相关性。

根据某一模型利用以往的样本值对新样本进行预测，然后将样本的实际值与其预测值相减得到一个误差值，最后对这一误差值进行编码。

如果模型足够好，且样本序列在时间上相关性较强，则无差信号的幅度将远远小于原始信号，从而得到较大的数据压缩。

1）DPCM（差分脉冲编码调制）的基本原理



预测误差为



预测误差的方差为



若用MSE作为失真度量，则最小化失真的最佳预测器为



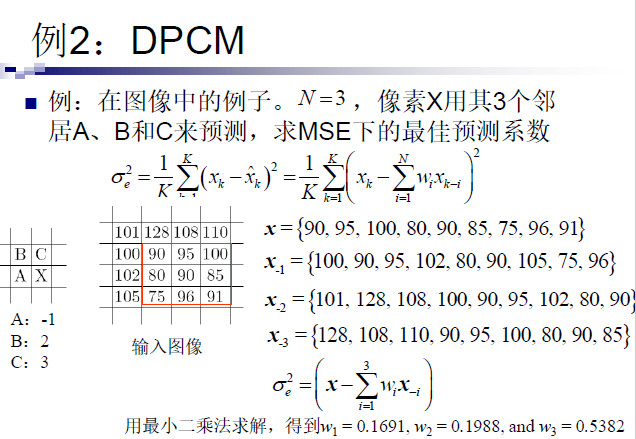
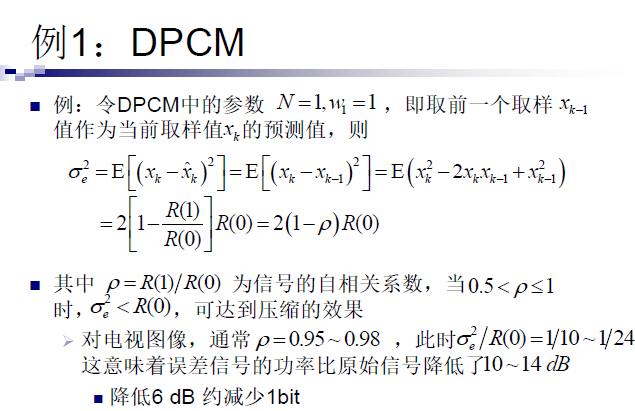
若采用前N个样本预测，且每个样本为B比特，则上述条件概率需要一个有2BN项的表

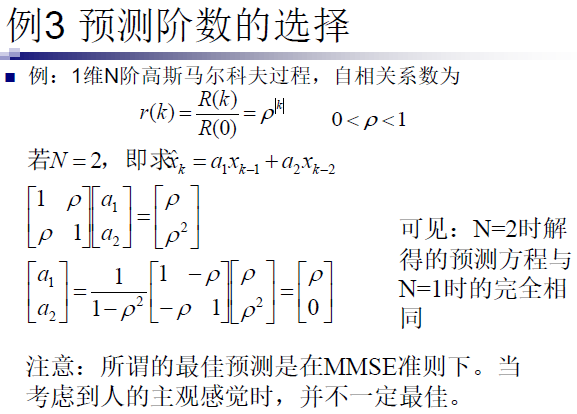
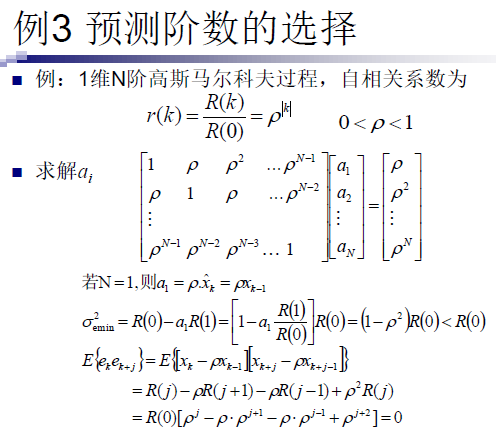
在一个DPCM系统中需要设计两个部分，预测器和量化器。在理想情况下，应同时优化预测器和量化器，但实际应用中，采用一种次优化方法，在这种方法中，量化电平数必须足够大。

线性预测器的设计：

具有最小均方预测误差的预测器——最大的预测增益

即在最小E[d²]的条件下，确定一组最佳预测系数（略去量化误差）





2）ADPCM

DPCM是以输出数据为平稳随机过程为依据。预测系数和量化器参数一次设计好后即不再改变。但在一些应用中，如在图像平坦区和边缘处要求量化器的输出差别很大。否则会导致图像出现噪声

ADPCM工作原理：预测器的预测系数和量化器的量化参数能够根据信源符合的局部区域分布特点而自动调整。

2、测编码在语音编码中的应用

1）语音信号的特性

浊音信号有长时间周期性，这是语音压缩的关键。

根据时间域的统计分析，语音信号中存在着多种冗余度：

①幅度非均匀分布

②样本间的相关

③周期之间的相关

④基音之间的相关

⑤静止系数（语音间隔）

⑥长时间自相关函数

如果从频率域考察语音信号的功率谱密度，可以发现：

①非均匀的长时功率谱密度

②语音特有的短时功率谱密度

2）波形编码

尽量保持重建语音与原始语音连信号波形都基本相同。

3）声码编码/参数编码

提取信源信号的特征参数并以数字代码传输；接收端从数字代码中恢复特征参数，由特征参数重建语音信号。

4）混合编码

既保留参数编码的声道模型，又像波形编码那样传递预测误差。

最流行的混合编码为时域合成-分析AbS算法，码激励线性预测编码（CELP）是AbS编码的一种。

3、预测编码在视频编码中的应用

1）视频信号中的冗余

各成分之间的冗余（彩色图像不同分量之间相关）、空间冗余（同一帧内相邻像素值相似）、时间冗余（相邻帧的内容通常比较相似）

2）帧内预测

去除空间冗余。对视频某一帧中的像素，可用其相邻像素的值来预测，然后对原信号与预测信号的差值进行编码。

3）帧间预测：运动估计/运动补偿

去除时间冗余。

运动估计的分类：

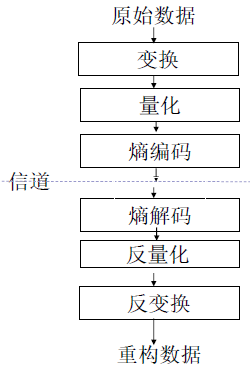
全局运动估计、基于像素点的运动估计、基于块的运动估计、基于区域的运动估计

基于块的运动估计、基于网格的运动估计

块匹配：

视频压缩时，只需保存运动矢量和残差数据就可以完全恢复处当前块

第六章 变换编码



1、变换编码基本原理

如果利用映射变换来实现对数据的建模表达，就称为变换编码。

其中映射变换是把原始信号中的各个样值从一个域变换到另一个域，然后针对变换后的数据再进行量化（二次量化）与编码操作。

接收端首先对收到的信号进行解码和反量化，然后再进行反变换以恢复原来信号（在一定的保真度下）。

2、离散正交变换

1）基本概念

令相邻的n个信号样本看作在n维线性空间中的一个列向量X=（x1,x2,...,xn）T

对它进行线性变换，Y=AX，称A为变换矩阵，Y是X的一个线性变换

如果矩阵A具有性质AAT=ATA=I，则称A为正交矩阵，从X到Y的变换为正交变换

对于正交变换，反变换可以唯一的得到复原信号X=ATY

对正交变换的总能量保持y=Ax，||y||²=yyT=xTATAx=||x||²，总的能量保持，但是通常能量在各系数上分布并不均匀

X的协方差矩阵 Φ11 Φ12 ... Φ1N

ΦX=E{[X-E(X)][X-E(X)]T}= Φ21 Φ22 ... Φ2N

... ... ...

ΦN1 ΦN2 ... ΦNN

其中Φij=E{[xi-E(xi)][xj-E(xj)]}=Φji

Y的协方差矩阵ΦY=E{[Y-E(Y)][Y-E(Y)]T}=E{A[X-E(X)][X-E(X)]TAT}=AΦXAT

为使得变换后各系数不相关，即要令Y的协方差矩阵中只存在对角线元素。

对于一个实对称矩阵Φ，必存在一个正交矩阵Q，使得QΦQ-1=diag[λ1λ2...λN]=Λ，其中对角阵Λ=diag[λ1λ2...λN]的N个对角元λ1λ2...λN是Φ的特征根，Q中的第i个行向量是Φ的第i个特征根λN所对应的满足归一化正交条件的特征向量，即Φqi=λiqi，qiTqj= 1，i=j

0，i≠j

只要选择正交矩阵Q作为变换核矩阵，其行向量是X的协方差矩阵的特征向量，则变换后的矢量Y的协方差矩阵为ΦY=E{[Y-E(Y)][Y-E(Y)]T}=E{[QX-E(QX)][QX-E(QX)]T}=QE{[X-E(X)][X-E(X)]T}QT=QΦXQT=Λ，则变换后各分量之间的相关性被全部去除。

2）KL变换

以矢量X的协方差矩阵的归一化正交特征向量（qi）所构成的正交矩阵Q，对该矢量所做的正交变换Y=QX称作KL变换。要实现KTL，必须要先知道输入信号的协方差矩阵，然后再求出协方差矩阵的特征值和特征向量。用特征向量构成该输入信号矢量的正交变换核矩阵。

A为n阶方阵，若存在数λ和n维非0列向量x使Ax=λx，则λ为x的特征值，x为A对应于特征值λ的特征向量，方程组形式（A-λI）x=0。

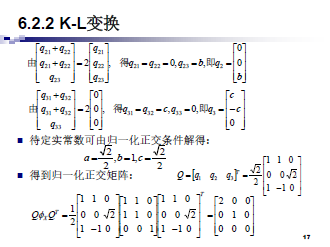
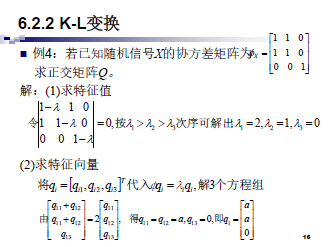
如果特征值和特征向量存在，则上述方程必有非0解x，即要求系数矩阵A-λI的秩<n，所以系数行列式|A-λI|=0，该式是关于λ的n次多项式，即矩阵A的特征多项式。

一个特征值可对应无穷个不同的特征向量。某一特征向量只对应于某一特征值。所以方阵A的不同特征值所对应的特征向量线性无关。

求法：（1）|A-λI|=λ的多项式，求出特征值

（2）把每个特征值带入（A-λI）x=0，求出齐次线性方程组的一个基础解系的全部解ξ1,ξ2,...,ξs，则

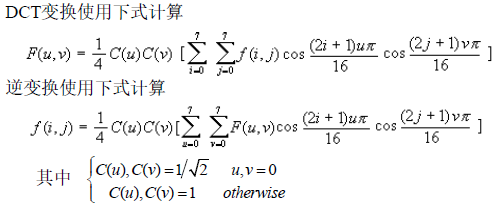
∑kiξi是A对应于特征值λ的全部特征向量，其中k1,k2,...,ks不全为0



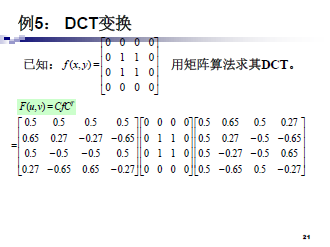
KLT：基函数为输入信号的协方差矩阵的特征向量。KLT产生的系数不相关（输出信号协方差矩阵为对角阵），KLT达到最佳的能量集中，KLT取决于信号的统计性质，没有快速计算法，寻找结构化的变换，使得其性能接近KLT

3）DCT变换

二维DCT变换



变换核可分离，先进行垂直方向的DCT变换，再进行水平方向的DCT变换



DFT在边界不连续，DCT更连续

3、JPEG图像压缩标准

1）压缩标准

JPEG规定了4种运行模式，以满足不同需要：

基于DPCM的无损编码模式：压缩比可达2:1

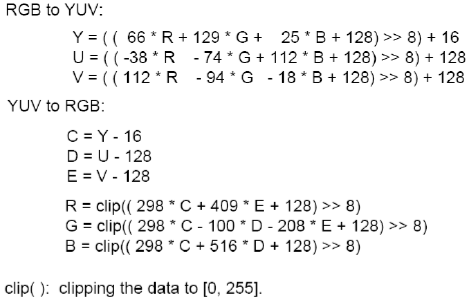
基于DCT的有损顺序编码模式：压缩比可达10:1以上

基于DCT的递增编码模式

基于DCT的分层编码模式

2）颜色空间

JPEG标准本身并没有规定具体的颜色空间，只是对各分量分别进行编码。实现中通常将高度相关RGB颜色空间转换到相关性较小的YUV颜色空间



3）零偏置

对于灰度级是2n的像素，通过减去2n-1，将无符号的整数值编程有符号数

对于n=8，即将0~255的值域，通过减去128，转换为值域在-128~127之间的值

目的：使像素的绝对值出现3位10进制的概率大大减少

4）量化

因为人眼对亮度信号比对色差信号更敏感，因此使用了两种量化表：亮度量化值和色差量化值

根据人烟的视觉特性，对低频分量采取较细的量化，对高频分量采取较粗的量化

真正的量化表=缩放因子×基本量化表

质量因子≤50：缩放因子=50/质量因子

质量因子>50：缩放因子=2-质量因子/50

5）DC系数的差分编码

8×8图像块经过DCT变换之后得到的DC直流系数有两个特点：系数的数值比较大，相邻8×8图像块的DC系数值变化不大：冗余

根据这个特点，JPEG算法使用了差分脉冲调制编码（DPCM）技术，对相邻图像块之间量化DC系数的差值DIFF进行编码：DIFFk=DCk-DCk-1

对DIFF用Huffman编码：分成类别，类似指数Golomb编码

类别ID：一元码编码 类内索引：采用定长码

6）AC系数

Z字扫描。由于经DCT变换后系数大多数集中在左上角，即低频分量区，因此采用Z字形按频率的高低顺序读出，可以出现很多连0的机会。可以使用游程编码。尤其在最后，如果都是0，给出EOB（End Of Block）即可。

在JPEG和MPEG编码中规定（run，level）为（连续run个0，后面跟值为level的系数）。

编码：Run最多15个，用4位表示RRRR；Level分成16个类别，用4位表示SSSS表示类别符号。对（RRRR，SSSS）联合用Huffman编码

解码重构：与编码相反。解码Huffman数据，解码DC差值，重构量化后的系数，DCT逆变换，丢弃填充的行/列，反0偏置，对丢失的CbCr分量插值（下采样的逆过程），YCbCr→RGB

7）JPEG文件的解析

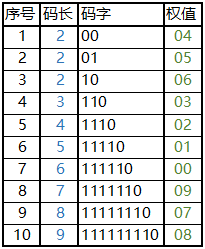
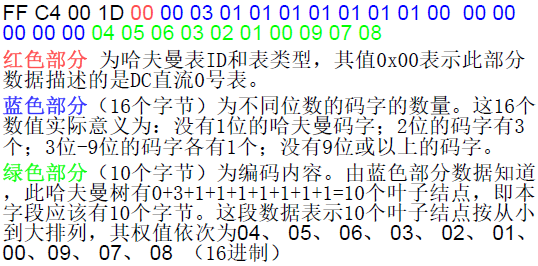
由若干个必不可少的标记顺序连接构成整个文件

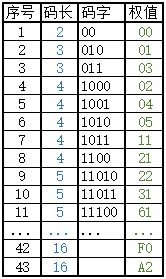
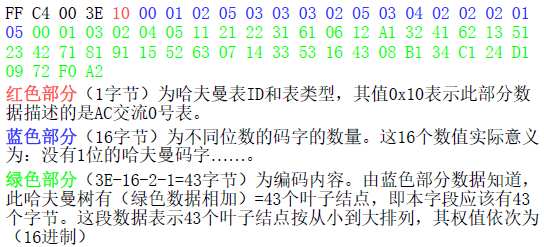
一定以0xFFD8开始，即表示图像开始SOL（Start of Image）标记

一定以0xFFD9结束，表示图像结束EOL（End of Image）标记

8）Huffman表存储

Huffman表的长度、类型（AC或DC）、索引（Index）、位表（bit table）、值表（value table）





建立Huffman树/表

在独处Huffman表的数据后，就要建立Huffman树。具体方法为：

①第一个码字必定为0

如果第一个码字位数为1，则码字为0；如果第一个码字位数为2，则码字为00；如此类推。

②从第二个码字开始，如果它和它前面的码字位数相同，则当前码字为它前面的码字加1；如果它的位数比它前面的码字位数大，则当前码字是前面的码字加1后再在后边添若干个0，直到满足位数长度为止。

直流：第一个例题建立的是DC（0）Huffman表，其权值就是解码时再需要读入的bit位数。这个再读入的位数通过查表得到真正的码值，表示直流系数

交流：对于交流系数，用交流Huffman树/表查的该码字对应的权值。权值的高4位表示当前数值前面有多少个连续的0，低4位表示该交流分量数值的二进制位数，也就是接下来需要读入的位数。例如权值为0X31，克表示为（3,1），表明此交流系数前面有3个0，而此交流系数的具体值还需要再读入1个bit的码字，才能得到。

第七章 分析-综合编码

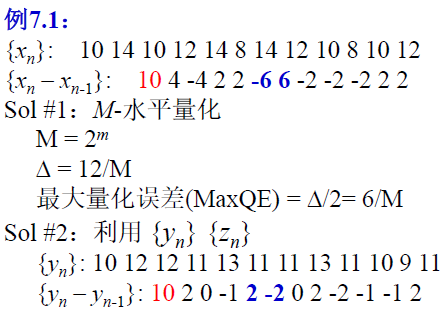
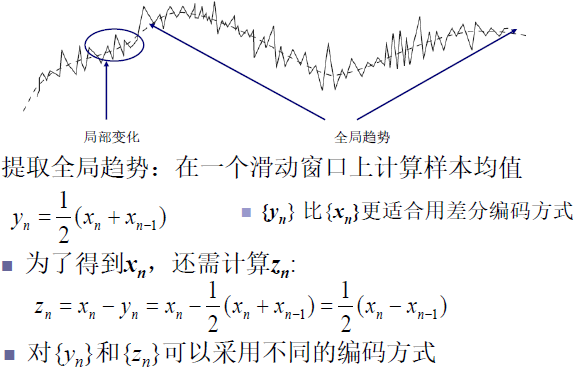
1、引言

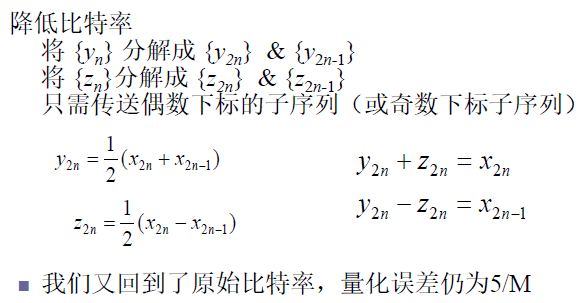
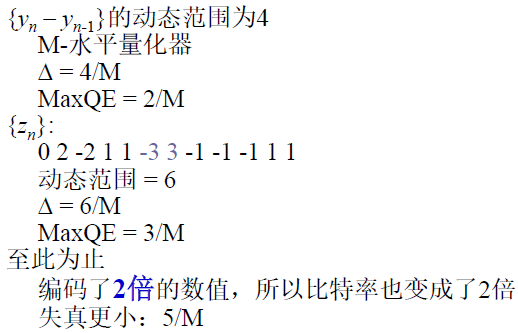
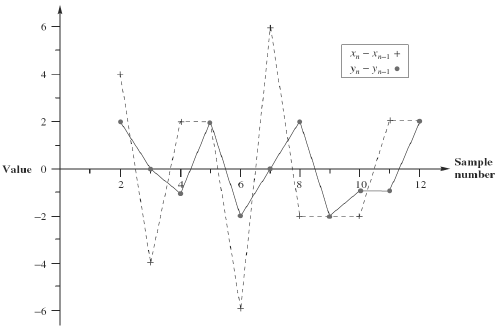
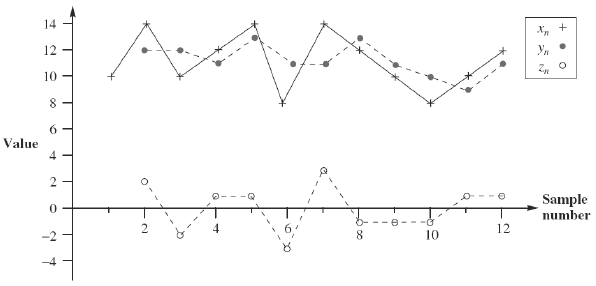
基于块变换的编码将信源输出分解为不同频率子带，然后对不同频率的子带编码

子带编码：将原始信号分解为若干个子频带，对其分别进行编码处理后再合成为全频带信号

小波变换编码：使用更强大的非均匀分辨率滤波器对信号进行时间-频率局部分析与综合

2、信号的分解与合成





在两种情况下，最后传送数值的数目相同

两个子序列有不同的特征

因此可以采用不同的编码机制

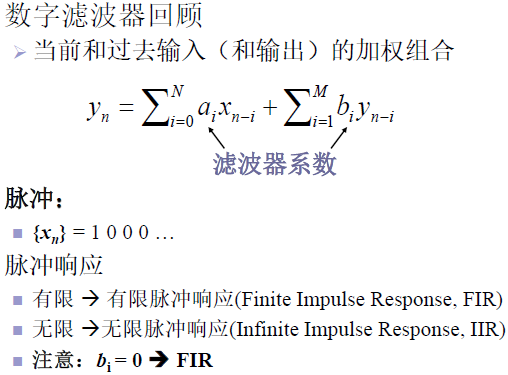
编码更灵活，从而编码效率更高

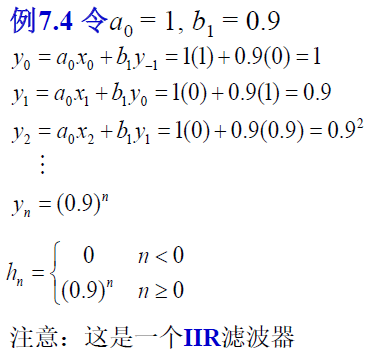
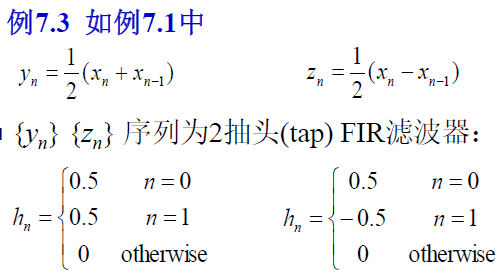
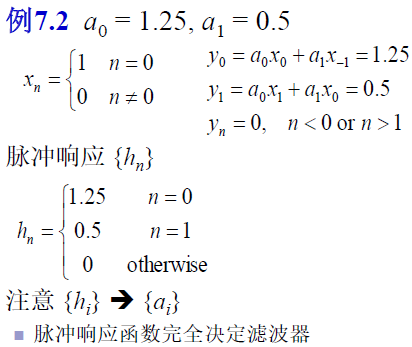
还可以递归使用该分解方法，得到子序列

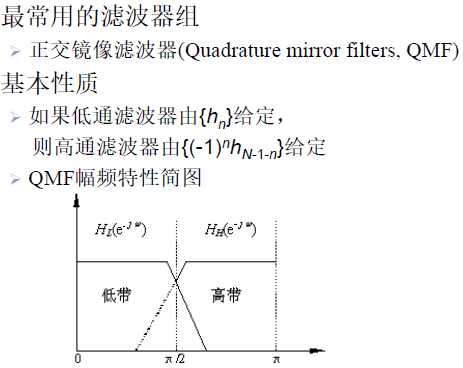
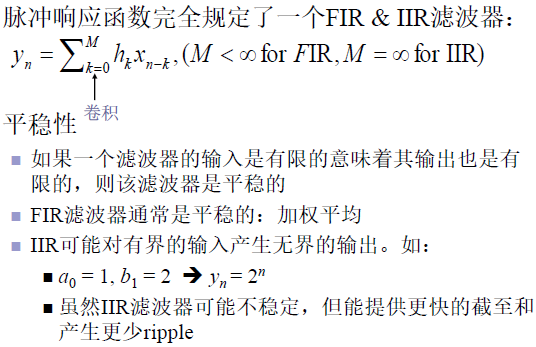
亦称为分析——综合

信号分解可利用数字滤波器实现

3、滤波器组的设计







4、基本的子带编码系统

三大要素：

分析——综合滤波器组，比特分配机制，编码机制：PCM、ADPCM、VQ...

量化&编码

不同子带运载不同数量的感知信息

比特分配对总质量影响极大

感知边界之外的带宽可以被抛弃

不太重要的子带可以用较低码率下采样，且可以采用较粗糙的量化粗量化

运载最重要信息的子带用高质量编码

计算比特分配Rk，使得给定码率R下失真D最小

5、MPEG音频压缩

1）MPEG-1声音的主要性能

输入为PCM信号，采样率为32、44.1或48kHz，输出为32kbps到384kbps

三个独立的压缩层次：Layer1、2、3，编码器复杂程度增加

2）编码器说明

输入声音信号经过一个多相滤波器组变换到多个子带，同时经过“心理声学模型”计算以频率为自变量的噪声掩蔽阈值。量化和编码部分用信掩比SMR决定分配给子带信号的量化位数，使量化噪声<掩比域值。最后通过数据帧包装将量化的自带样本和其它数据按照规定而帧格式组装成比特数据流。

①多相滤波器组，用来分割子带

划分子带的方法有两种：线性划分和非线性划分

线性划分可能一个子带覆盖好几个临界频带

②量化和编码——比例因子的取值和编码

对各个子带每12个样点金鼎一次比例因子计算。先定出12个样点中绝对值的最大值。查比例因子表中比这个最大值大的最小值作为比例因子。用6比特表示。

③数据帧包装

彩色空间转换实验

基本原理：

RGB和YUV彩色空间的基础知识；数据类型的分析

1）不带参数的主函数定义void main（）

带参数的主函数定义void main（int argc，char\*argv[ ]）

调用：命令名 参数1 参数2 ... 参数n

参数值：argc=n+1 argv[0]=命令名 argv[1]=参数1 argv[2]=参数2 ... argv[n]=参数n

2）文件相关指针

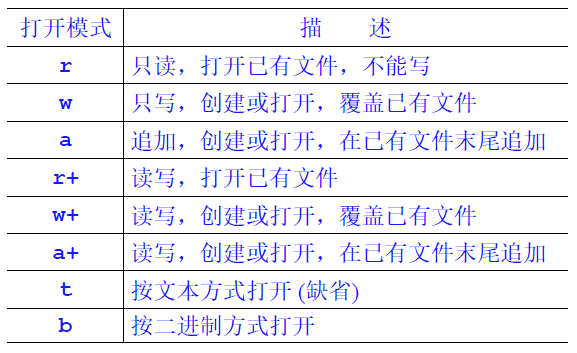
FILE \*fp;

FILE \*fopen（char \*filename，char \*mode）;

filename是要打开的文件路径

mode是要打开的模式

若成功，返回指向被打开文件的指针；若出错，返回空指针NULL

举例：

//指针定义

char \*rgbFileName = NULL;

FILE \*rgbFile = NULL;

//参数获取

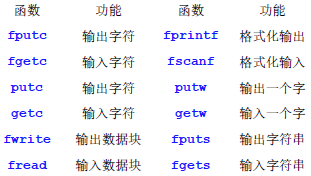
rgbFileName = argv[1];

//打开函数

rgbFile = fopen(rgbFileName, "rb");

if (rgbFile == NULL)

{

 printf("cannot find rgb file\n");

exit(1);

}

else

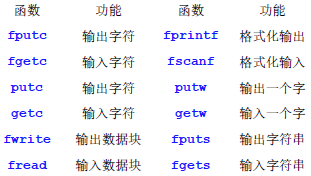
{

printf("The input rgb file is %s\n", rgbFileName);

}

fclose（FILE \*fp）;

fp是要关闭的文件指针

若成功，返回0；若出错，返回EOF（-1）

不用的文件应

feof（FILE \*fp）;

fp是文件指针

若文件结束，返回非零值；若文件尚未结束，返回0

fputc（int c, FILE \*fp）;

c是要输出到文件的字符

fp是文件指针

若成功，返回输出的字符；若失败，返回EOF

fgetc（FILE \*fp）;

fp是文件指针

若成功，返回输入的字符；若失败或文件结束，返回EOF

fwrite/fread（void \*buffer, size\_t size, size\_t count, FILE \*fp）;

buffer是要写/读的数据块地址

size是要写/读的每个数据项的字节数

count是要写/读的数据项数量

fp是文件指针

若成功，返回实际写/读的数据项数量；若失败，一般返回0

举例：

//开辟缓存空间

rgbBuf = (u\_int8\_t\*)malloc(frameWidth \* frameHeight \* 3);

yBuf = (u\_int8\_t\*)malloc(frameWidth \* frameHeight);

uBuf = (u\_int8\_t\*)malloc((frameWidth \* frameHeight) / 4);

vBuf = (u\_int8\_t\*)malloc((frameWidth \* frameHeight) / 4);

//读取数据

fread(rgbBuf, 1, frameWidth \* frameHeight \* 3, rgbFile);

//写入数据

fwrite(yBuf, 1, frameWidth \* frameHeight, yuvFile);

fwrite(uBuf, 1, (frameWidth \* frameHeight) / 4, yuvFile);

fwrite(vBuf, 1, (frameWidth \* frameHeight) / 4, yuvFile);

fprintf/fscanf（FILE \*fp, char \*format[,address/argument,...]）;

fputs(char \*s, FILE \*fp);

若成功，返回输出字符个数；若失败，返回EOF

fgets（char \*s, int n, FILE \*fp）;

若成功，返回s首地址；若失败，返回NULL

从fp输入字符串到s中，输入n-1个字符，或遇到换行符或EOF为止，读完后自动在字符串末尾添加 ’\0’

rewind（FILE \*fp）;

fp是文件指针

使文件位置指针重新返回文件开头

fseek（FILE \*fp, long offset, int whence）;

fp是文件指针

offset是偏移量

whence是起始位置

随机改变文件的位置指针

SEEK\_SET(0)是文件开始

SEEK\_CUR(1)是文件当前位置

SEEK\_END(2)是文件末尾

ftell（FILE \*fp）;

fp是文件指针

返回fp所指向文件中的读写位置

3）动态数组和指针

①申请空间

char \*name;

name = (char \*)malloc(20);

float \*pf;

pf = (float \*)malloc(sizeof(float)\*20);

double \*pd;

pd = (double \*)malloc(sizeof(double)\*50);

②赋值

int \*pi;

pi = (int \*)malloc(sizeof(int)\*100);

\*pi = 0;

\*(pi+1) = 1;

\*(pi+2) = 2;

int j;

for(j=0; j<100; j++)

\*(pi+j) = 0;

举例：

for (i = 0; i < size; i++)

{

g = b + 1;

r = b + 2;

\*y = (unsigned char)( RGBYUV02990[\*r] + RGBYUV05870[\*g] + RGBYUV01140[\*b]);

\*u = (unsigned char)(- RGBYUV01684[\*r] - RGBYUV03316[\*g] + (\*b)/2 + 128);

\*v = (unsigned char)( (\*r)/2 - RGBYUV04187[\*g] - RGBYUV00813[\*b] + 128);

b += 3;

y ++;

u ++;

v ++;

}

③空间回收

free（filename）;

4）RGB to YUV文件转换

①流程分析

程序初始化（打开两个文件、定义变量和缓冲区等）

读取RGB文件，抽取RGB数据写入缓冲区

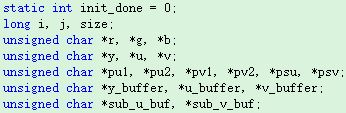
调用RGB2YUV的函数实现RGB到YUV数据的转换

写YUV文件

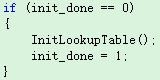
程序收尾工作（关闭文件，释放缓冲区）

②代码分析

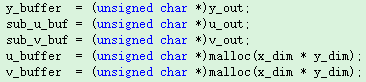
定义变量

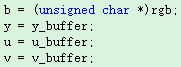


调用快速查找表

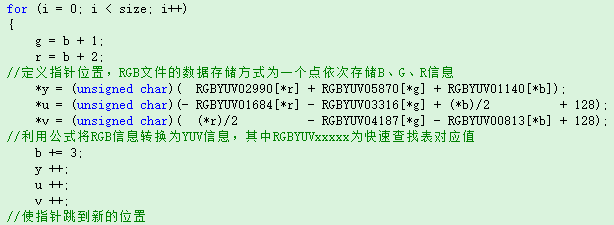


定义指针、开辟缓存空间

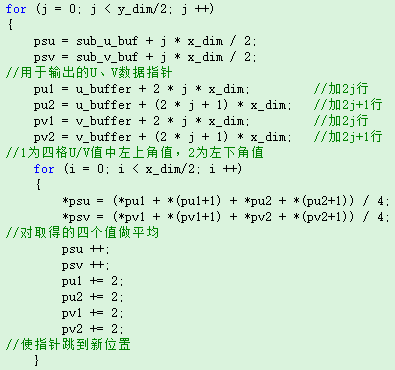




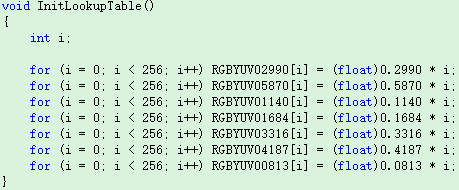
将RGB数据转换为YUV数据



将转换为的U、V数据转换为4：2：0格式



附查找表



5）YUV to RGB文件转换

①流程分析

程序初始化（打开两个文件、定义变量和缓冲区）

读取YUV文件，抽取YUV数据写入缓冲区

调用YUV2RGB的函数实现YUV到RGB数据的转换

写RGB文件

程序收尾工作（关闭文件，释放缓冲区）

②代码分析

定义变量

调用快速查找表

定义指针、开辟缓存空间

将YUV数据转换为RGB数据

附查找表

图像文件的读写和转换

1）BMP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;

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速 \*/

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;

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;

调用文件头信息头

#include<windows.h>

BITMAPFILEHEADER File\_header;

BITMAPINFOHEADER Info\_header;

// read file & info header

if(fread(&File\_header,sizeof(BITMAPFILEHEADER),1,bmpFile) != 1)

{

printf("read file header error!");

exit(0);

}

if (File\_header.bfType != 0x4D42)

{

printf("Not bmp file!");

exit(0);

}

else

{

printf("this is a %s\n",File\_header.bfType);

}

if(fread(&Info\_header,sizeof(BITMAPINFOHEADER),1,bmpFile) != 1)

{

printf("read info header error!");

exit(0);

}

// end read header

调用调色板

RGBQUAD \*pRGB=(RGBQUAD\*)malloc(sizeof(RGBQUAD)\*(unsigned char)pow(2,info\_h.biBitCount));

if(!MakePalette(pFile,file\_h,info\_h,pRGB))

printf("No palette!");

bool MakePalette(FILE \* pFile,BITMAPFILEHEADER &file\_h,BITMAPINFOHEADER & info\_h,RGBQUAD \*pRGB\_out)

{

if ((file\_h.bfOffBits - sizeof(BITMAPFILEHEADER) - info\_h.biSize) == sizeof(RGBQUAD)\*pow(2,info\_h.biBitCount))

{

fseek(pFile,sizeof(BITMAPFILEHEADER)+info\_h.biSize,0);

fread(pRGB\_out,sizeof(RGBQUAD),(unsigned int)pow(2,info\_h.biBitCount),pFile);

return true;

}

else

return false;

}

2）BMP to YUV文件转换

①转换流程

程序初始化（打开两个文件、定义变量和缓冲区等）

读取BMP文件，抽取或生成RGB数据写入缓冲区

读位图文件头（判断是否可读出、判断是否是BMP文件）

读位图信息头（判断是否读出）

判断像素的实际点阵数

开辟缓冲区，读数据，倒序存放

根据每像素位数的不同，执行不同的操作

（8bit以下，构造调色板，位与移位取像素数据查调色板，写RGB缓冲区

16bit，位与移位取像素数据转换为8bit/彩色分量，写RGB缓冲区

24/32bit，直接取像素数据，写RGB缓冲区）

16bitBMP：

for (Loop = 0;Loop < height \* width;Loop +=2)

{

\*rgbDataOut = (Data[Loop]&0x1F)<<3;

\*(rgbDataOut + 1) = ((Data[Loop]&0xE0)>>2) + ((Data[Loop+1]&0x03)<<6);

\*(rgbDataOut + 2) = (Data[Loop+1]&0x7C)<<1;

rgbDataOut +=3;

}

1~8bitBMP：

int shiftCnt = 1;

while (mask)

{

unsigned char index=mask

==0xFF?Data[Loop]:((Data[Loop]&mask)>>(8-shiftCnt\* info\_h.biBitCount));

\* rgbDataOut = pRGB[index].rgbBlue;

\* (rgbDataOut+1) = pRGB[index].rgbGreen;

\* (rgbDataOut+2) = pRGB[index].rgbRed;

if(info\_h.biBitCount == 8)

mask = 0;

Else

mask >>= info\_h.biBitCount;

rgbDataOut+=3;

shiftCnt ++;

}

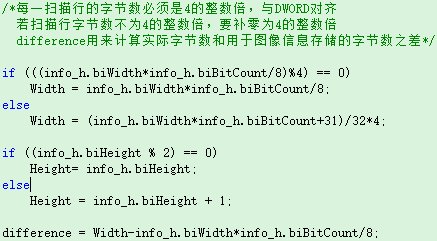
调用RGB2YUV的函数实现RGB到YUV数据的转换

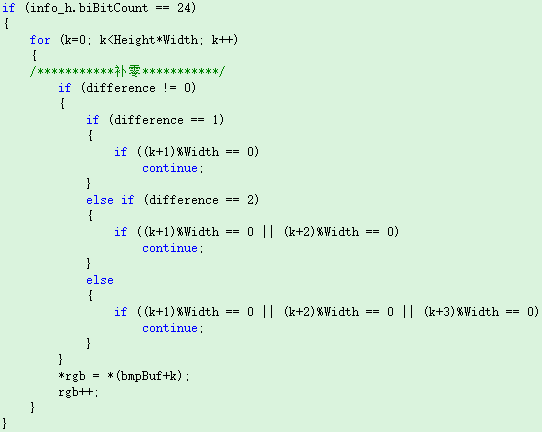
采用部分查找表法，提高运行效率

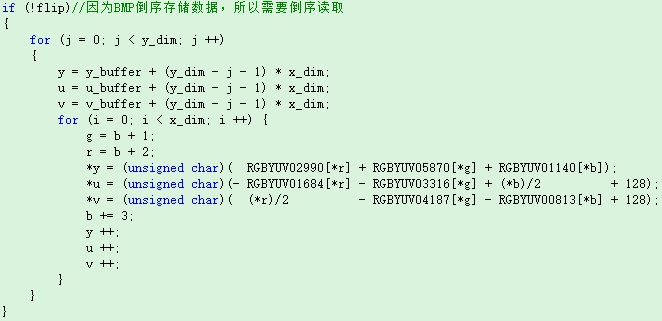
写YUV文件

程序收尾工作（关闭文件，释放缓冲区）

②代码分析







Huffman编解码算法实现与压缩效率分析

1）Huffman编码的数据结构

Typedef struct huffman\_node\_tag //Huffman节点标签

{

unsigned char isLeaf //是否为树叶

unsigned long count; //节点代表的符号加权和

struct huffman\_node\_tag \*parent; //父节点指针

union

{

struct

{

struct huffman\_node\_tag \*zero, \*one; //子节点指针,分别代表0,1子节点指针

};

unsigned char symbol; //节点代表的符号

};

} huffman\_node;

typedef struct huffman\_code\_tag //Huffman编码标签

{

unsigned long numbits; //该码所用的比特数

unsigned char \*bits; //指向该码比特串的指针

} huffman\_code;

2）Huffman编码的流程

从指定文件中读取数据，统计每个符号发生的概率，并建立相应的树叶节点

注意在此处pSF代入的是地址

第一次扫描：统计文件中各个字符出现频率

构建Huffman树及生成Huffman码

按字符概率由小到大将对应节点排序

得到文件出现的字符种类数

构建Huffman树

对码树编码

将码表及其他必要信息写入输出文件

第二次扫描：对源文件进行编码并输出

JPEG原理分析及JPEG解码器的调试

1）JPEG文件格式介绍

SOI start of image 图像开始

标记代码 2字节 固定值0xFFD8

APP0 appllication 应用程序保留标记0

标记代码 2字节 固定值0xFFE0

包含9个具体字段

①数据长度 2字节 ①~⑨9个字段的总长度  
 ②标识符 5字节 固定值0x4A46494600，即字符串“JFIF0”  
  ③版本号 2字节 一般是0x0102，表示JFIF的版本号1.2  
  ④X和Y的密度单位 1字节 只有三个值可选  
 0：无单位；1：点数/英寸；2：点数/厘米  
  ⑤X方向像素密度 2字节 取值范围未知  
  ⑥Y方向像素密度 2字节 取值范围未知     
  ⑦缩略图水平像素数目 1字节 取值范围未知  
  ⑧缩略图垂直像素数目 1字节 取值范围未知  
  ⑨缩略图RGB位图  长度可能是3的倍数 缩略图RGB位图数据

DQT define quantization table 定义量化表

标记代码 2字节 固定值0xFFDB

包含9个具体字段：  
  ①数据长度 2字节 字段①和多个字段②的总长度  
  ②量化表 数据长度-2字节

a) 精度及量化表ID 1字节

高4位：精度，只有两个可选值 0：8位；1：16位  
 低4位：量化表ID，取值范围为0～3

b) 表项 (64×(精度+1))字节

例如8位精度的量化表,其表项长度为64×（0+1）=64字节

本标记段中，字段②可以重复出现，表示多个量化表，但最多只能出现4次

SOF0 start of frame 帧图像开始

标记代码 2字节 固定值0xFFC0

包含9个具体字段：  
  ①数据长度 2字节 ①~⑥六个字段的总长度  
 ②精度 1字节 每个数据样本的位数  
 通常是8位，一般软件都不支持 12位和16位  
  ③图像高度 2字节 图像高度（单位：像素）

  ④图像宽度 2字节 图像宽度（单位：像素）

  ⑤颜色分量数 1字节 只有3个数值可选  
 1：灰度图；3：YCrCb或YIQ；4：CMYK  
 而JFIF中使用YCrCb，故这里颜色分量数恒为3  
  ⑥颜色分量信息 颜色分量数×3字节（通常为9字节） a)颜色分量ID 1字节

b)水平/垂直采样因子 1字节

高4位：水平采样因子  
 低4位：垂直采样因子  
 c)量化表 1字节 当前分量使用的量化表的ID

DHT define huffman table 定义哈夫曼表

标记代码 2字节 固定值0xFFC4

包含2个具体字段：

①数据长度 2字节

②huffman表 数据长度-2字节

表ID和表类型 1字节

高4位：类型，只有两个值可选

0：DC直流；1：AC交流  
 低4位：哈夫曼表ID，

注意，DC表和AC表分开编码

不同位数的码字数量 16字节

编码内容 16个不同位数的码字数量之和（字节）  
本标记段中，字段②可以重复出现（一般4次），也可以只出现1次。

SOS start of scan 扫描开始12字节

标记代码 2字节 固定值0xFFDA

包含2个具体字段：  
  ①数据长度 2字节 ①~④两个字段的总长度  
 ②颜色分量数 1字节 应该和SOF中的字段⑤的值相同，即：  
 1：灰度图是；3： YCrCb或YIQ；4：CMYK。

③颜色分量信息  
     a)颜色分量ID 1字节  
     b)直流/交流系数表号 1字节

高4位：直流分量使用的哈夫曼树编号  
 低4位：交流分量使用的哈夫曼树编号

  ④压缩图像数据  
        a)谱选择开始 1字节 固定值0x00  
        b)谱选择结束 1字节 固定值0x3F  
        c)谱选择 1字节 在基本JPEG中总为00

EOI end of image 图像结束

标记代码 2字节 固定值0xFFD9

2）JPEG文件解码流程

读入文件的相关信息

JPEG文件解码流程

SOI(0xFFD8)

APP0(0xFFE0)

[APPn(0xFFEn)]可选

DQT(0xFFDB)

SOF0(0xFFC0)

DHT(0xFFC4)

SOS(0xFFDA)

压缩数据

EOI(0xFFD9)

Huffman表存储方式

在标记段DHT内，包含了一个或多个的哈夫曼表。对于单个哈夫曼表，应该包括三部分：

①Huffman表ID和表类型

0x00表示DC直流0号表 0x01表示DC直流1号表  
 0x10表示AC交流0号表 0x11表示AC交流1号表

②不同位数的码字数量

JPEG的Huffman编码只能是1~16位。这个字段的16个字节分别表示1~16位的编码码字在Huffman树中的个数。

③编码内容

这个字段记录了Huffman树中各个叶子结点的权。所以，上一字段的16个数值之和就应该是本字段的长度，也就是Huffman树种叶子结点的个数。

初步了解图像数据流的结构

颜色分量单元的内部解码

直流系数的差分编码

反量化&反Zig-Zag编码

反离散余弦变化

3）JPEG文件解码程序实现

对Huffman码字解码前不知道码字的长度，解码有两种方法，Huffman码树遍历和lookup查找表

4）对函数build\_huffman\_table的分析

第一步：得到code\_size.（huffsize）

第二步：得到code.（huffcode）

第三步：得到code\_size查找表。

Table->code\_size[val] = code\_size;

第四步：得到value查找表（lookup）。

table->lookup[code++] = val

第五步：当码字长度>9，slowtable处理。

slowtable[0] = code; slowtable[1] = val; slowtable[2] = 0;

