## 图像

YUV色彩模型来源于RGB模型，该模型的特点是将亮度和色度分离开，从而适合于图像处理领域。YCbCr来源于YUV。

H.264以YCbCr对图像进行描述，Y为亮度分量，CbCr为两个色差分量。

Y为颜色的亮度(luma)成分，表示光的浓度且为非线性，使用伽马修正(gamma correction)编码处理。而Cb和Cr则为蓝色和红色的浓度偏移量成份。

H．264支持取样格式为4：2：0的连续或隔行视频的编码和解码，4：2：0表示在水平和垂直取样时，Cr和Cb都是Y的一半。

一个彩色影像的图和一个彩色影像只有 Y 成份的图、只有 Cb 成份的图、和只有 Cr 成份的图。同时注意的是只有 Y 成份的图基本上来说就是彩色影像那张图的灰阶本。



RGB->YCbCr

                     Y = 0.299R + 0.587G + 0.114B

                     Cb = 0.564\*(B-Y)

                     Cr = 0.713\*(R-Y),

其中 Y 表示亮度， Cb Cr 表示色度。

YCbCr->RGB

                      R = Y + 1.402\*Cr

                      G = Y - 0.344\*Cb - 0.714\*Cr

                      B = Y + 1.772\*Cb

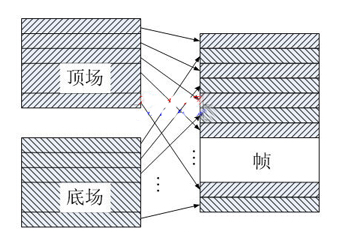


YPbPr：YPbPr也叫色差分量接口，采用的是美国电子工业协会EIA-770.2a标准，还有一种接口被称为YCbCr接口，他们两者的区别在于前者是逐行扫描色差输出，后者是隔行扫描色差输出。

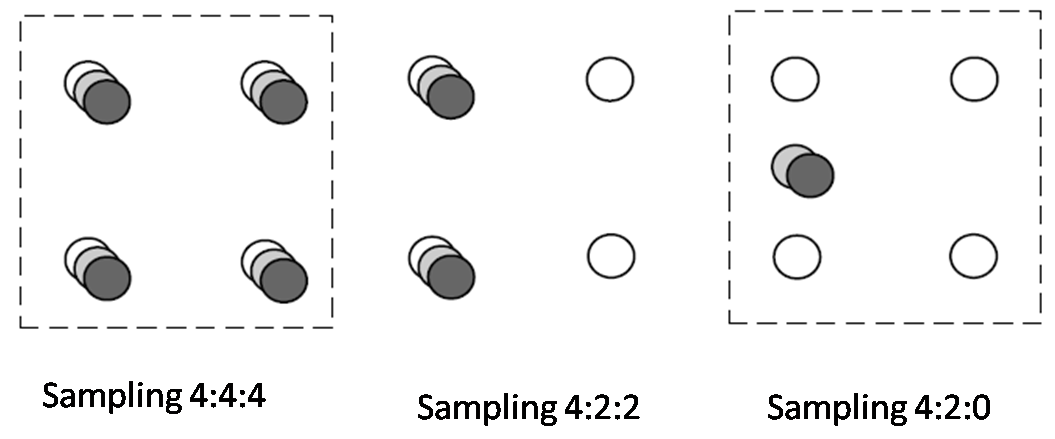
Google画质效果排序如下：

HDMI=DVI > 色差YPbPr =VGA(大约等于) > 色差YCbCr > S端子 > AV端子

**场和帧：**



**YUV的采样格式：**



4:4:4：Y,Cb,Cr具有同样的水平和垂直取样率，每4个亮度像素，相应的就有4个Cb和4个Cr。

4:2:2：Y,Cb,Cr具有同样的垂直取样率，但水平取样率色度分量是亮度分量的一半，即水平方向上，每4个亮度像素，相应的就有2个Cb和2个Cr。

4:2:0：水平和垂直取样率色度分量都是亮度分量的一半，即水平垂直方向上，每4个亮度像素，相应的就有1个Cb和1个Cr。

人眼对亮度分量Y值非常敏感，而对UV分量相对会弱一些，所以我们会对UV进行**下采样(down simpling)，**以此来减少数据量。

## 为什么要进行压缩：

**原始数据过于庞大：**

一帧标清分辨率（720x576）原始视频数据要占用720x576x3=1,244,160Bytes 1M

一秒的实时数据就是近29M (1,244,160x25) Bytes

视音频有很多信息的冗余，具有进行高比例压缩的潜力

重要指标：

**码率bitrate：**

当前移动互联网时代，带宽资源越来越昂贵，对于视频在网络传输尤其关注，而码率是影响视频传输是否流畅的关键因素。

影响码率的参数：分辨率，fps，

**图像质量：峰值信噪比PSNR(Peak Signal-to-Noise Ratio)/结构相似性SSIM(structural similarity index)/主观质量评价**

PSNR是使用最广泛的客观视频质量的度量方法，图像压缩中典型的峰值信噪比值(PSNR)在 30 到 40dB 之间，愈高愈好。

相较于传统所使用的影像品质衡量指标，像是峰值信噪比(英文：PSNR)，结构相似性在影像品质的衡量上更能符合人眼对影像品质的判断

**压缩速度：**如果编码过程占用大量的CPU，将很大影响到机房服务器的利用，手机终端执行速度。

**视频编码主要思路：**

**主要思想是通过预测去除时空冗余，再用变换将信息集中到少数的几个部分，然后通过量化来精简数据量，最后用无损压缩的方法来压量化的结果。**

## 无损压缩

一般无损压缩的压缩率只有3倍左右，对于象视频这样的大数据流，纯粹的无损压缩不能满足要求。

### 游程编码（RLE编码——Run Length Encoding）

游程编码又称“运行长度编码”或“行程编码”，是一种统计编码，该编码属于无损压缩编码。对于二值图有效。

行程编码的基本原理是：用一个符号值或串长代替具有相同值的连续符号（连续符号构成了一段连续的“行程”。行程编码因此而得名），使符号长度少于原始数据的长度。

例如：5555557777733322221111111

行程编码为：（5，6）（7，5）（3，3）（2，4）（l，7）。可见，行程编码的位数远远少于原始字符串的位数。

在对图像数据进行编码时，沿一定方向排列的具有相同灰度值的像素可看成是连续符号，用字串代替这些连续符号，可大幅度减少数据量。

行程编码是连续精确的编码，在传输过程中，如果其中一位符号发生错误，即可影响整个编码序列，使行程编码无法还原回原始数据。

游程编码所能获得的压缩比有多大，主要取决于图像本身的特点。如果图像中具有相同颜色的图像块越大，图像块数目越少，获得的压缩比就越高。反之，压缩比就越小。

游程长度编码是一种无损数据压缩，非常适合基于调色板的图标图像。但是它并不适用于连续色调图像的压缩，例如日常生活中的照片。

游程长度编码用于**传真机**（并和其他技巧一起组成了修改过的huffman编码）。相对而言，游程长度编码是比较有效的，因为传真的文档主要是黑白的（二值文档）。

### 熵编码entropy coding（比如变长编码和算术编码）

JPEG/MPEG/H.264主流视频标准基本都有用到

算术编码比变长编码效率高10%～15%，计算复杂度也高很多

**算术编码:**

算术编码是一种无失真的编码方法，能有效地压缩信源冗余度，属于熵编码的一种。算术编码的一个重要特点就是可以按分数比特逼近信源熵，突破了Huffman 编码每个符号只不过能按整数个比特逼近信源熵的限制。对信源进行算术编码，往往需要两个过程，第一个过程是建立信源概率表，第二个过程是对信源发出的符号序列进行扫描编码。而自适应算术编码在对符号序列进行扫描的过程中，可一次完成上述两个过程，即根据恰当的概率估计模型和当前符号序列中各符号出现的频率，自适应地调整各符号的概率估计值，同时完成编码。尽管从编码效率上看不如已知概率表的情况，但正是由于自适应算术编码具有实时性好、灵活性高、适应性强等特点，在图像压缩、视频图像编码等领域都得到了广泛的应用。

实现算术编码首先需要知道信源发出每个符号的概率大小，然后再扫描符号序列，依次分割相应的区间，最终得到符号序列所对应的码字。整个编码需要两个过程，即概率模型建立过程和扫描编码过程。

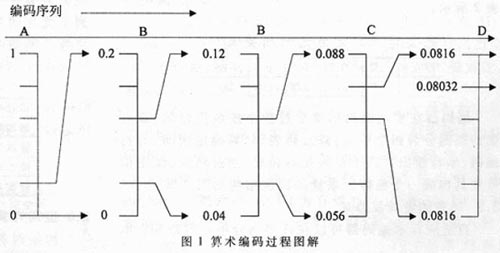


图1 给出一个实现算术编码的示例。要编码的是一个来自四符号信源{A，B，C，D}的由五个符号组成的符号序列：ABBCD。

假设已知各信源符号的概率分别为: P(A)=0.2,P(B)=0.4,P(C)=0.2,P(D)=0.2。编码时，首先根据各个信源符号的概率将区间[0，1]。分成四个子区间。符号A 对应[0，0.2]，符号B 对应[0.2,0.6]，符号C 对应[0.6,0.8]，符号D 对应[0.8,1.0]。

符号序列中第一个符号是A，其对应的区间为[0,0.2]，接下来将这个区间扩展为整个高度，再根据各个信源符号的概率将这个间扩展为整个高度，再根据各个信源符号的概率将这个新区间分成四段；第二个符号是B，它对应新的子区间的第二个子区间，即对应区间[0.04,0.12]；再将该区间扩展为整个高度，再根据这个过程直接最后一个符号得到一个区间[0.08032,0.0816]，这样该区间内的任何一个实数就可以表示整个符号序列，如0.081。

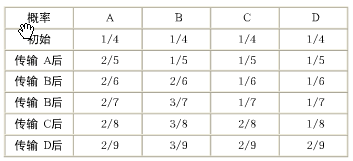
通过0.081我们就可以通过概率表推算出来ABBCD.

CABAC(基于上下文的自适应二进制算术编码)

Context-Based A daptive Binary Arithmetic Codeing in the H.264/AVC

自适应概率调整：

以ABBCD为例，概率调整值：



CABAC是H264的一种熵编码方案，相比如H264的另外一种熵编码方案CAVLC而言，在可接受的视频质量（30dB到38dB之间）内变化时，前者可节约平均9%到14%的码流。CABAC有以下几个特性：

1. 对多数符号使用了自适应概率模型。
2. 通过使用上下文关系，利用了符号相关性。
3. 限制为二进制算术编码（binary arithmetic coding），基于只用查表和移位方式的快速二进制算术编解码器。
4. 399 种预定义的上下文模型，分成了各种不同的模型组，例如模型 14-20 用于帧间宏块类型的编码，模型的选择基于前面编码的信息（上下文关系），每个上下文模型适应实验分布。

熵编码小组主讲的内容有：UVLC&CAVLC，哥伦布码的编码方法，CAVLC的具体过程并做了编码的具体说明。CABAC，整个过程的介绍，包括二进制化，上下文建模，算术编码。

### 字典序压缩 LZW算法

主要有ZIP压缩，GIF图片等

LZW算法中，首先建立一个字符串表，把每一个第一次出现的字符串放入串表中，并用一个数字来表示，这个数字与此字符串在串表中的位置有关，并将这个数字存入压缩文件中，如果这个字符串再次出现时，即可用表示它的数字来代替，并将这个数字存入文件中。

原始数据： ABCCAABCDDAACCDB，

ABCD可以用0～3的数来表示。那么注意这个符串中出现了好几个重复的字串：

ABCCAABCDDAACCDB

那么就可以用 4来代表AB，5来代表CC等等，原来的字符串就变为

压缩后的数据：45A4CDDAA5DB

## 有损压缩

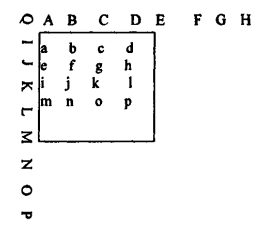
主要思想是通过**预测**去除时空冗余，再用**变换**将信息集中到少数的几个部分，然后通过**量化**来精简数据量，最后用**无损压缩**的方法来压量化的结果。

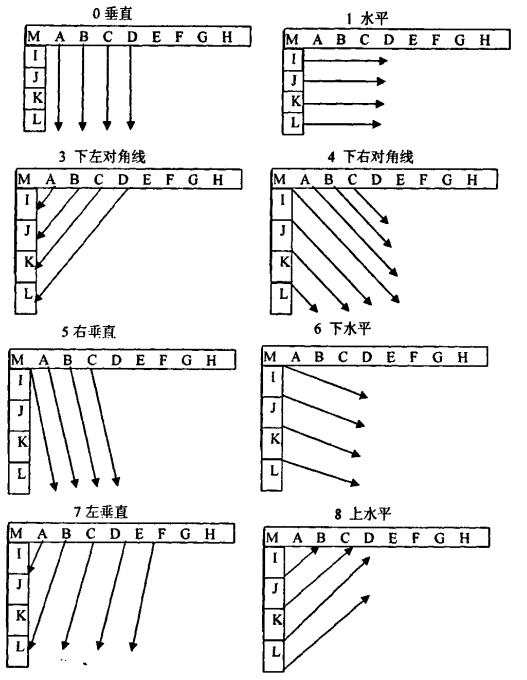
### 预测编码(Prediction Coding)：

#### 空域预测/帧内预测编码(Intra Predictive Coding)

空域预测一般是利用一帧图像内部的信息，在需要预测编码的像素周边 寻找相近像素来作为预测来减少数据量；H.264帧内编码参考预测块左方或上方已编码块的邻近像素点，引入空间域。

Intra4X4亮度预测模式下，将宏块分割成16个4X4子块进行帧内预测，每个4X4块中的像素可通过相邻块中已经解码的17个像素进行预测。帧内的预测模式分为9种(模式0到模式8)，其中模式2为DC预测，表示该宏块内所有像素值相等，是像素A～D和l～L的平均值。其它几种模式的预测方向如下图所示。





判断哪种模式的准则一般是用当前待编码宏块与每种预测块相减并计算绝对误差和(SAE，Sum of Absolute Errors)，选取SAE值最小的预测模式为预测的最佳模式。

#### 时域预测/帧间预测编码(Inter Predictive Coding)

时域预测一般是通过前后几帧的图像信息来预测需要编码的帧，从而达到减少数据量的作用。帧间预测编码在视频编码方法中占有很重要的地位，通常是针对图像块的预测编码，它采用的技术有帧重复法，阈值法，运动补偿法等。其中运动补偿法效果为最佳。缺点：分块边缘不连续。

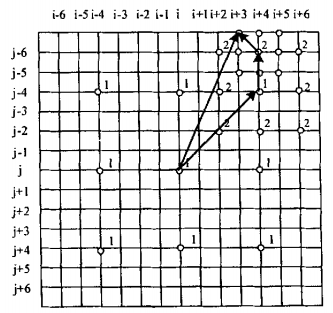
运动估计：

基本思想是将图像序列的每一帧分成许多互不重叠的宏块，并认为宏块内所有象素的位移量都相同， 然后对每个宏块到参考帧某一给定特定搜索范围内根据一定的匹配准则找出与当前块最相似的块，即匹配块，匹配块与当前块的相对位移即为运动矢量。 视频压缩的时候，只需保存运动矢量和残差数据就可以完全恢复出当前块。

运动估计（Motion Estimation, ME):其表达方式是运动矢量（Motion Vector, MV），研究的主要内容就是如何快速、有效的获得有足够精度的运动矢量(MV)．



三步快速搜索法:



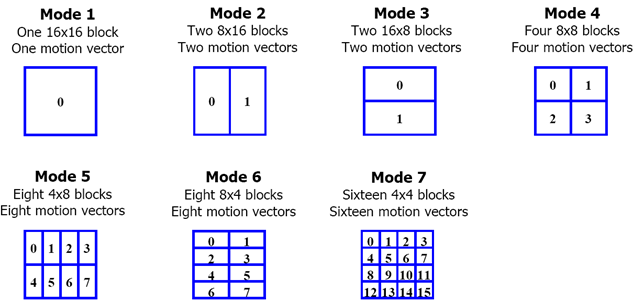
可改进

会议电视，可视电视通常不应用双向预测，。

帧间预测主要是利用连续图像序列之间的相关性，通过运动补偿的预测编码方法消除视频图像的时间冗余。H. 264 除了具有原有标准（ 如 H. 263、MPEG - 4）中的 P 帧，B 帧预测方法外，还增加了许多新的功能：使用不同大小的预测块进行运动估计；使用高精度的亚像元运动矢量补偿算法；使用多参考帧进行帧间预测编码；在预测环路使用去除块效应滤波器。

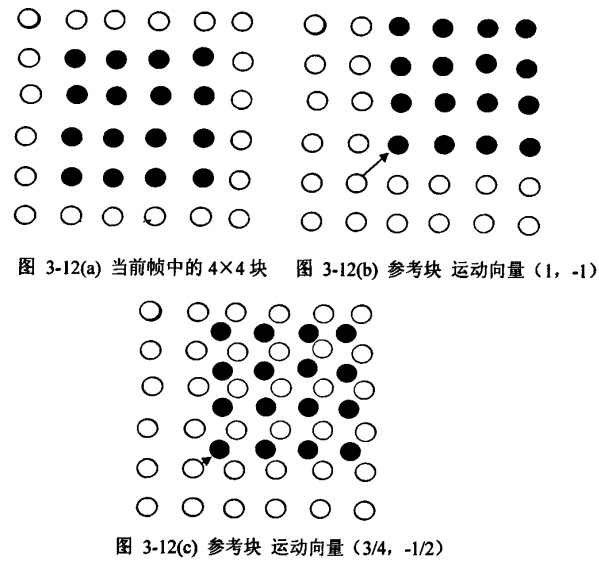
**不同大小的预测块进行运动估计：**

宏块上的运动补偿可通过使用不同块大小和形状来实现。H. 264 标准中根据不同的语法，对帧间预测时每个 16 × 16 宏块，分为 16 × 16、16 × 8、8 ×16、8 × 8、8 × 4、4 × 8 和 4 × 4 或更小的块进行变换编码，共七种块大小模式，如图 3 所示。由于采用不同大小的块进行帧间预测，使得运动估计模型更接近物体的实际运动，一般来说，使用小的运动补偿块可以提高预测精度，具体地说，小的块增强了编码模型处理精细运动细节能力，而且不会产生大的块效应，同时提高了主观视觉质量，这种方法比单独 16× 16 块的预测方法提高大于 15%的编码率。



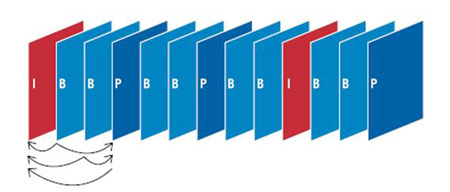
**高精度运动估计**

H. 264 中使用比现有标准更高的空间精度来决定运动矢量（H. 263 中使用半像素精度），从而大大地提高了运动补偿算法的预测能力。目前在 H. 264的运动估计精度可以达到 1 / 4、1 / 8 像素，与整数精度的空间预测相比它可以提高大于 20%的编码率。



**多参考图像运动补偿**

与仅采用前一幅图像作为参考图像相比，H.264允许编码器在已经编码的一组图像中选取一幅或者多幅图像来做为参考图像进行运动补偿，这样编码器可以在更多的图像中进行最佳块的匹配。这在很多自然场景的周期变换以及镜头在两个场景中交替转换的情况下尤其有效。例如，对于鸟类飞翔的视频序列，由于翅膀的扇动具有周期性，因而采用多参考图像的运动估计方式就有机会选择时问间隔为一个或者接近一个周期的图像作为更有效的参考。因此，这种多参考图像机制可以为当前编码宏块或宏块分割搜索到更好的匹配。

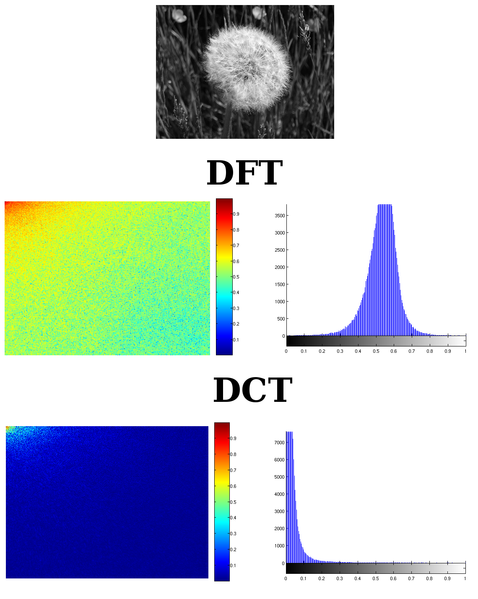


I帧（帧内编码帧）是一种自带全部信息的独立帧，视频序列中的第一个帧始终都是I帧。I帧可以用来实现快进、快退以及其它随机访问功能。

P帧（帧间预测编码帧）P帧属于前向预测的帧间编码，它只参考前面最靠近它 的I帧或者P帧。

### 变换编码(transform coding)：

为达到某种目的将原始图象变换映射到另一个空间上，使得图象的某些特征得以突出，以便于后面的处理和识别。一般变换后的图象，大部分能量都分布于低频谱段，这对以后图象的压缩、传输都比较有利。



变换编码，系指欲编码之数值经过一数学转换后映射至另一值域后再进行编码处理。变换编码经常与量化一起使用，进行有损数据压缩。如将时域信号变换到频域信号，因为声音图像大部分信号都是低频信号，在频域中信号的能量较集中，再进行采样、编码，就可以压缩数据。

最常用的变换是离散余弦变换，其次还有小波变换、Hadamard变换等等。离散余弦变换DCT在性能上接近K-L变换(Karhunen-Loève 变换)，能够很好的实现能量集中，广泛的应用于几乎所有的视频压缩标准中。

DCT(Discete Cosine Transfor)：基于分块的离散余弦变换。缺点是会出现块效应和飞蚊噪声。

主要变换方法：

* 离散傅里叶变换
* 离散余弦变换DCT
* 小波变换

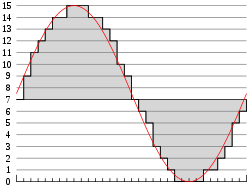
有损的8x8DCT主要被应用在 JPEG/MPEG1/MPEG2/MPEG4 part2/H.261/H.263

无损的4x4/8x8整数DCT及哈达吗变换被应用于H.264

小波变换被应用于JPEG2000及还在制定中的H.265

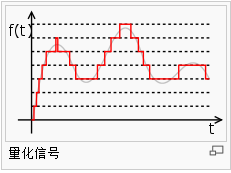
### 矢量量化：

数字电视的PCM原理：



量化在有损数据压缩中起着相当重要的作用。很多情况下，量化可以被当作将有损数据压缩同无损数据压缩相区别的标志之一。量化的目的通常是为了减少数据量。

量化在转码过程中对视频质量的影响比较大，在转码中如果选择了【恒定量化值】就需要对这它的最量化值做设置，H.264的量化值从0到51，通常默认为51，转化出来的效果就会很差。



JPEG是一种利用了量化的图像有损压缩。JPEG的编码过程对原始的图像数据作离散余弦变换，然后对变换结果进行量化并作熵编码。通过量化可以降低变换值的精度，从而减少图像的数据量。当然，精度的损失意味着图像质量的下降。然而图像的质量可以通过量化位数的选择加以控制。例如，JPEG在每像素3比特的精度下得到的图像质量还让人可以接受的，相对于PCM抽样得到的每个像素24比特的原始图像来说，数据量大大下降了。

<http://zh.wikipedia.org/wiki/%E8%84%88%E8%A1%9D%E7%B7%A8%E8%99%9F%E8%AA%BF%E8%AE%8A>

H.264的算法在概念上可以分为两层：视频编码层（VCL：Video Coding Layer）负责高效的视频内容表示，网络提取层（NAL：Network Abstraction Layer）负责以网络所要求的恰当的方式对数据进行打包和传送。

## 视频处理：

视频编码的主要技术核心

1. 运动估计（ME）算法
2. 码率控制（RC）算法

目前的N-PASS算法主要是针对这部分来优化的，多次的反复处理。Pass1,pass2

3. 帧/场/宏块的模式选择算法等

*视频质量/带宽/性能*

### 视频缺陷的纠正

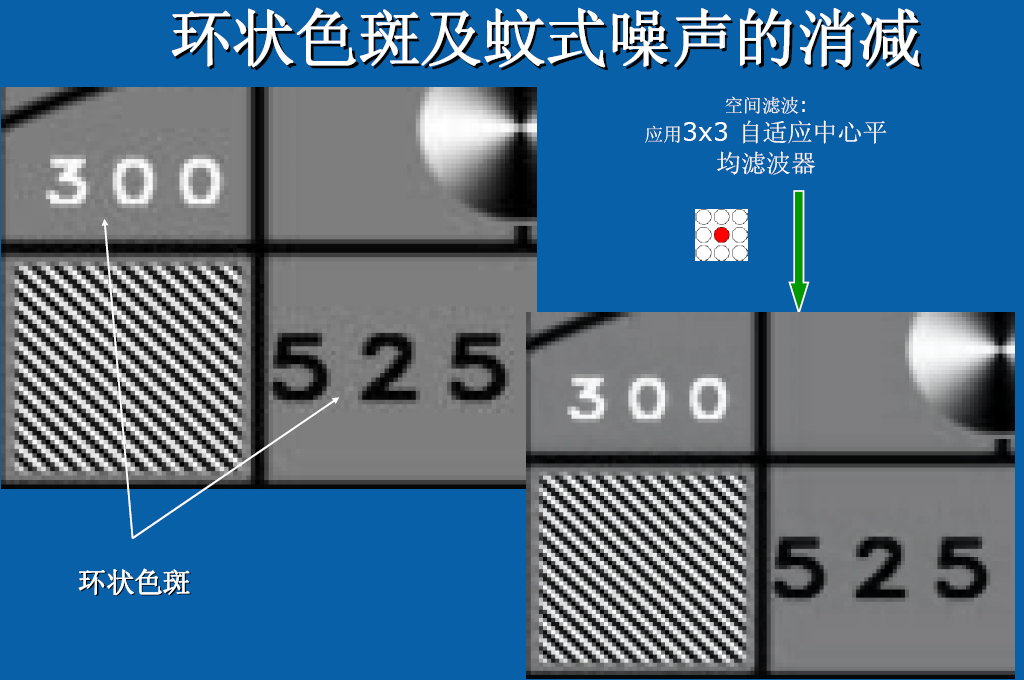
去块效应， 去噪等



基于块编码的图像在低码率压缩时都容易产生块效应或者说马赛克现象，而去块滤波是去除块效应的主要手段。去块滤波又分为**环路滤波**和**后置滤波**，后置滤波器只处理编码环路外的显示缓冲器中的数据，不会影响后续图像的处理，是H.264标准的可选项，而环路滤波处理编码环路中的数据，滤波后的数据将可能作为后续图像处理时的参考数据，并有助于提高主客观图像质量，是H.264标准中必须实现的部分。

上述内容是我对去块效应滤波的最新认识，看来以前我对去块滤波的作用看的太简单了，事实上，去块滤波能够有效提高压缩图像的主客观质量，是整个视频编解码中非常重要的一环。

去块滤波最好的实现方案是在某宏块重建完成后立即对该宏块进行去块滤波，这样不但有助于减少编码码流还可以充分利用CPU的缓冲机制，实现更快的编解码。



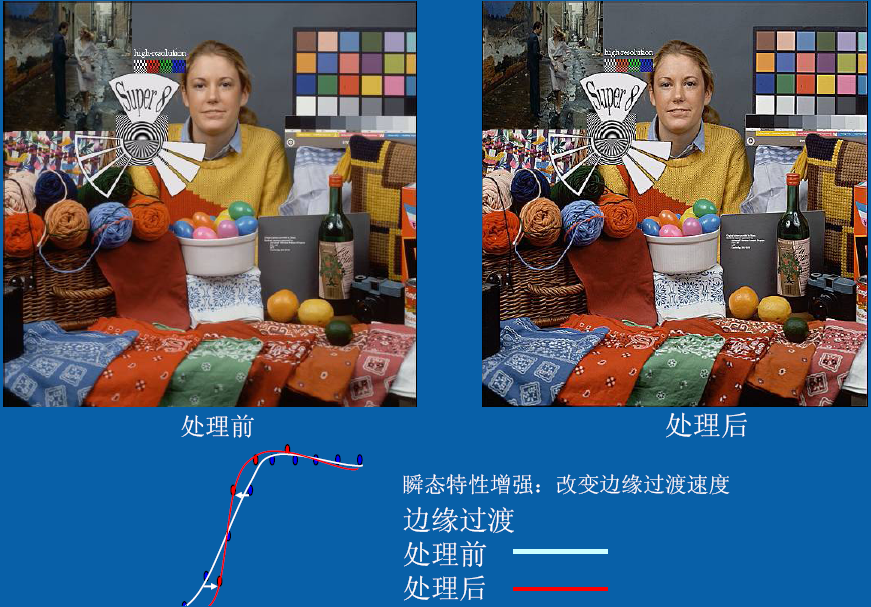
### 视频格式的转换

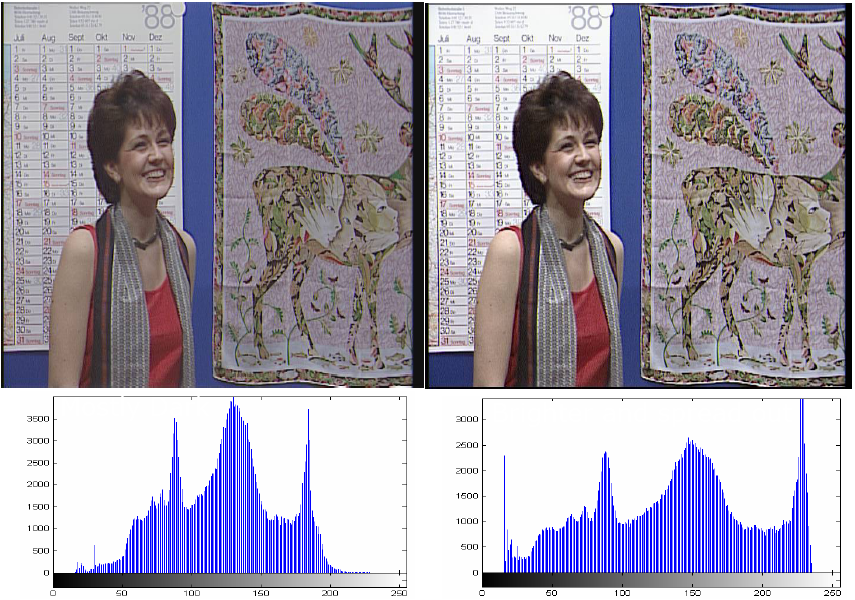
去交错，缩放， 变帧率等

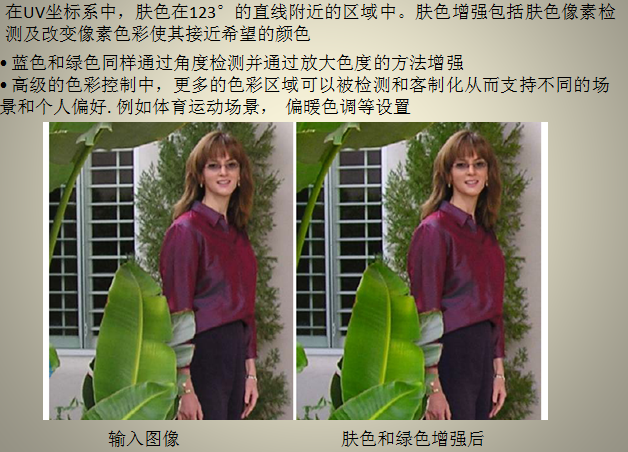


### 视频增强

锐化， 色彩饱和度在增强，对比度增强等



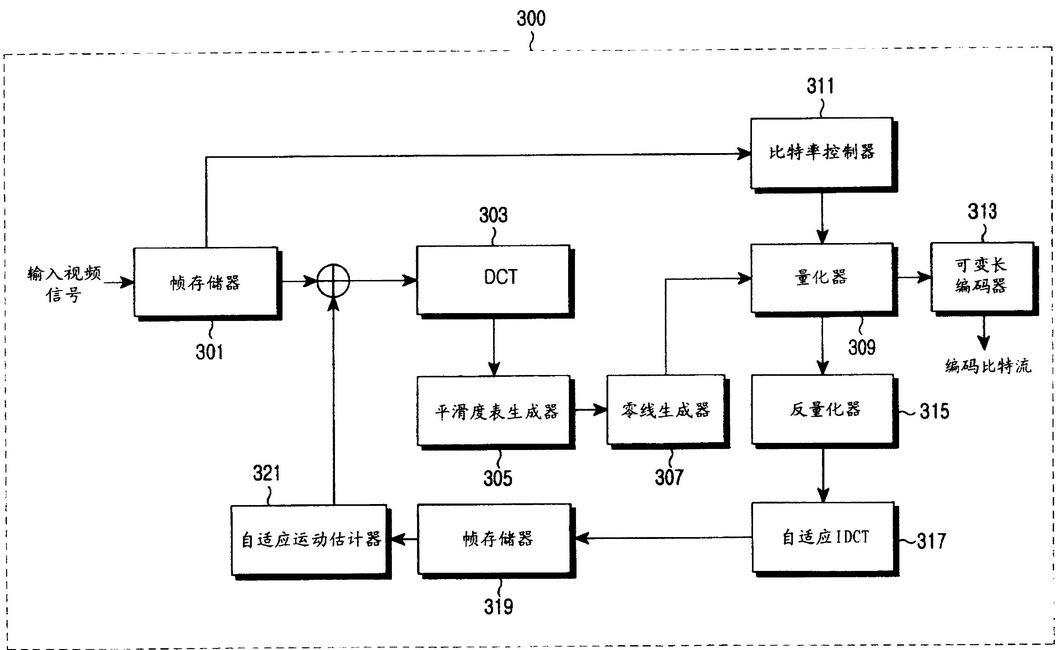


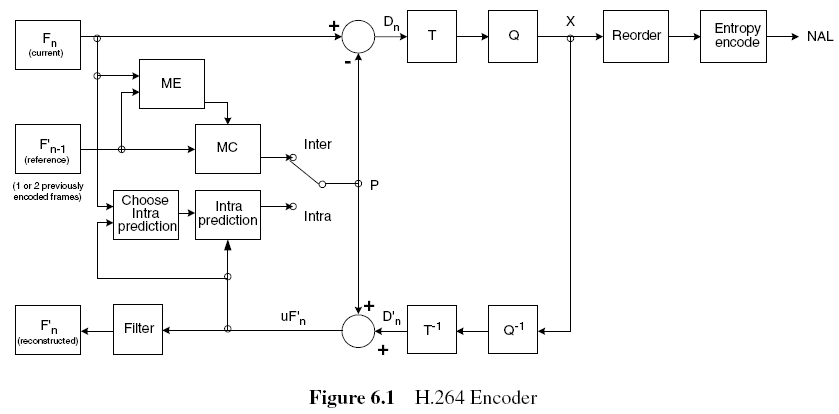


视频解码的主要关键

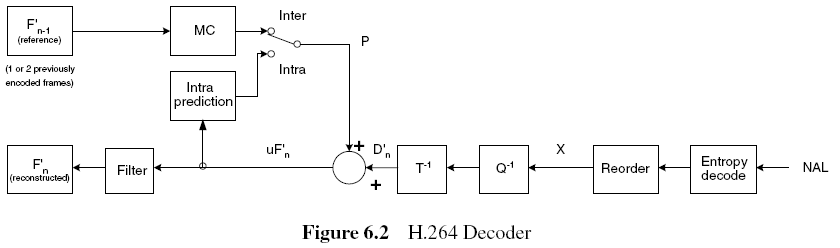
1. 兼容性
2. 容错性
3. 运动补偿（MC）/熵解码/后处理算法等

*视频质量/性能*





编码器采用的是变换和预测的混合编码法。在图6.1中，输入的帧或场Fn以宏块为单位被编码器处理。首先，按帧内或者帧间预测编码的方法进行处理。如果采用帧间预测编码，其预测值PRED是由当前片中前面已编码的参考图像经运动补偿（MC）后得到，其中参考图像用F'n-1表示。预测值PRED和当前块相减后，产生一个残差块Dn，经块变换、量化后产生一组量化后的变换系数X，再经熵编码，与解码所需的一些头信息一起组成压缩后的码流，经NAL（网络自适应层）供传输和存储用。



将编码器的NAL输出的H264比特流经熵解码得到量化后的一组变换系数X，再经反量化、反变换，得到残差D'n。利用从该比特流中解码出的头信息，解码器就产生一个预测块PRED，它和编码器中的原始PRED是相同的。当该解码器产生的PRED与残差D'n相加后，就得到了uF'n，再经滤波后，最后就得到滤波后的解码输出图像F'n。

## 视频解码

### 解码

I frame：

帧内编码帧 又称intra picture，I 帧通常是每个 GOP（MPEG 所使用的一种视频压缩技术）的第一个帧，经过适度地压缩，做为随机访问的参考点，可以当成图象。I帧可以看成是一个图像经过压缩后的产物。

P frame：

前向预测编码帧 又称predictive-*frame*，通过充分将低于图像序列中前面已编码帧的时间冗余信息来压缩传输数据量的编码图像，也叫预测帧；

B frame：

双向预测内插编码帧 又称bi-directional interpolated prediction frame，既考虑与源图像序列前面已编码帧，也顾及源图像序列后面已编码帧之间的时间冗余信息来压缩传输数据量的编码图像，也叫双向预测帧；

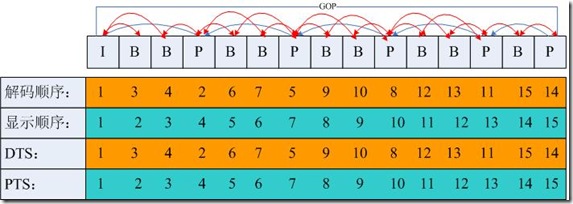
DTS：Decode Time Stamp。主要是标识读入内存中的bit流在什么时候开始送入解码器中进行解码。

PTS：Presentation Time Stamp。主要用于度量解码后的视频帧什么时候被显示出来

DTS主要用于视频的解码,在解码阶段使用.PTS主要用于视频的同步和输出.在display的时候使用.在没有B frame的情况下.DTS和PTS的输出顺序是一样的.

**例子:**

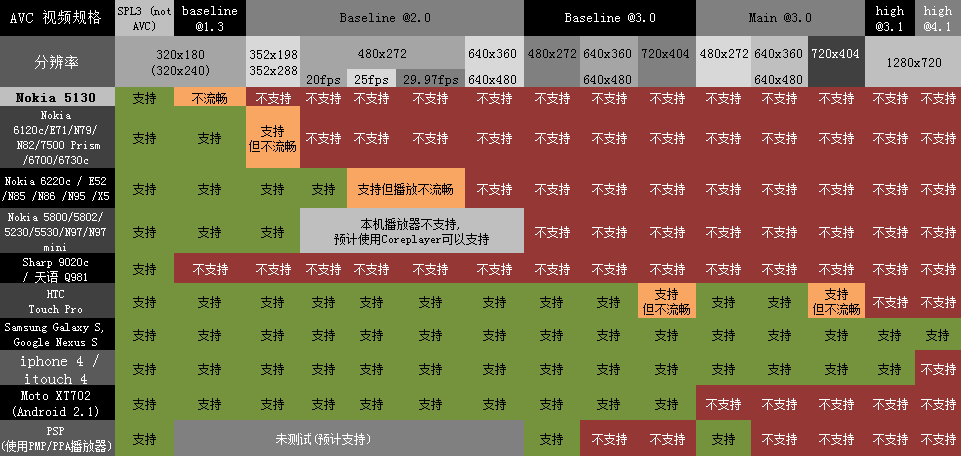
下面给出一个GOP为15的例子,其解码的参照frame及其解码的顺序都在里面:



如上图：I frame 的解码不依赖于任何的其它的帧.而p frame的解码则依赖于其前面的I frame或者P frame.B frame的解码则依赖于其前的最近的一个I frame或者P frame 及其后的最近的一个P frame.

### 移动终端

智能机移动终端应按Baseline @2.0来转码视频，如下为测试表：



<http://hi.baidu.com/%D2%BB%C9%FD%CF%E3%C3%D7/blog/item/0937ec4518a3e847510ffe09.html>

移动终端的UGC视频通常视频质量比较差，但文件却又非常大，所以在处理时会牺牲一些质量。

如youku上面的一些数据：



## 流媒体

流化技术(hint) 按照相应的标准在mp4文件中建立hint track[建立mdat(media data)的rtp分割,index等]，从而使文件适合Internet发布。 流化采用的标准:

– rfc3640: RTP Payload Format for Transport of MPEG-4 Elementary

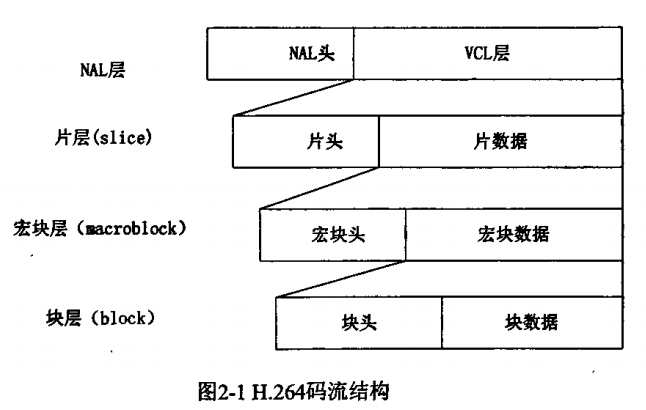
– rfc3016: RTP Payload Format for MPEG-4 Audio/Visual Streams

– rfc2250: RTP Payload Format for MPEG1/MPEG2 Video

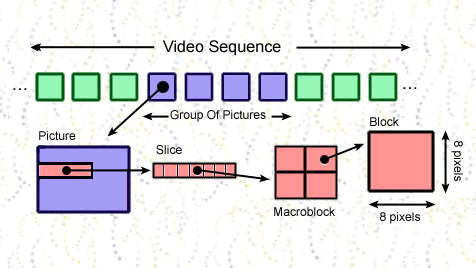
ISMA的处理方法是：从编码端到解码端的所有环节，均需建立多个音视频和其他数据流的RTPSession，因此在做StreamingServer的I/O时，需要管理多个输入输出，多个Buffer的管理以及它们之间的同步，将极大地增加StreamingServer的处理能力要求，也带来了较大的算法复杂性，同时降低了系统的稳定性和可靠性。而MPEG2 TS则将多个音视频和其他数据流复用在一起，仅需建立一个RTPSession，因此在做I/O，Buffer管理和音视频同步等方面将会简单和容易得多。

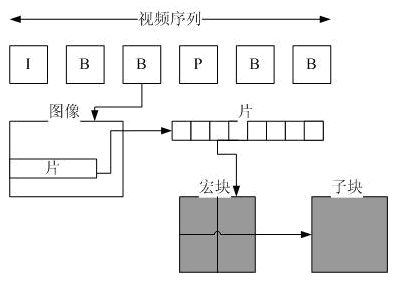
## 转码

### H264

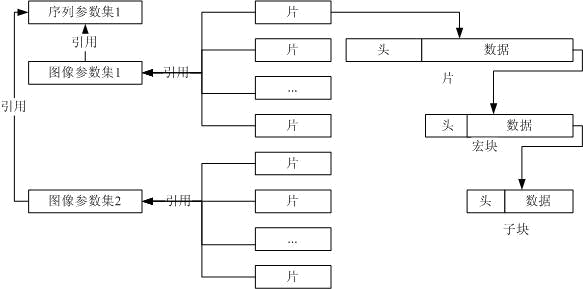


NAL层是外围层，它根据视频信号传输的媒介把VCL的内容封装起来，形成NAL单元。所有的视频内容最终都是以NAL单元的形式存储和传送的。NAL依据VCL的内容提供不同的NAL头信息。头信息中包括同步码字或者VCL单元长度，VCL层类型等。NAL单元序列能够在基于包的网络或比特流传输链路中传输，也能够储存为文件。从VCL层自上而下包括片层、宏块层和块层。每一层包含了头信息和数据。





H.264分层结构由五层组成，分别是序列参数集、图像参数集、片（Slice）、和宏块和子块。参数集是一个独立的数据单位，不依赖于参数集外的其它句法元素。下图描述了参数集与参数集外的句法元素之间的关系。



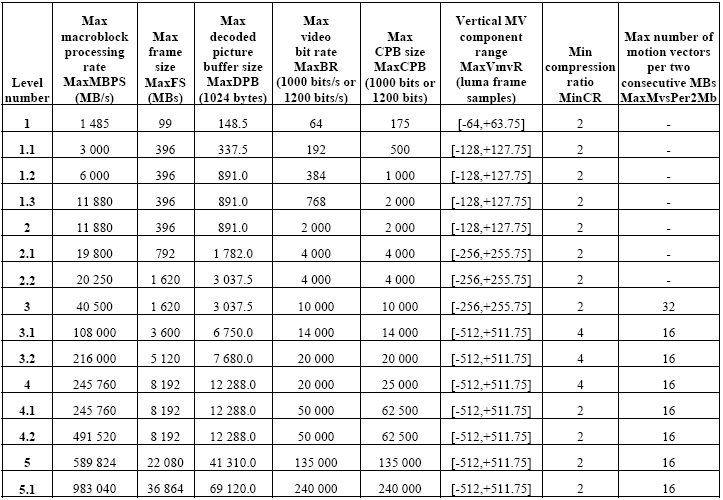
在H.264中，图像以序列为单位进行组织。一个序列的第一个图像叫做IDR图像，IDR图像都是I帧，H.264引入IDR图像为了解码的同步，当解码器解码到IDR图像时，立即将参考帧队列清空，将已解码的数据全部输出或抛弃，重新查找参数集，开始一个新的序列。这样，如果前一个序列出现重大错误，在这里可以获得重新同步的机会。IDR图像之后的图像永远不会使用IDR之前的图像的数据来解码。

IDR是I帧，但I帧不一定是IDR。I帧之后的图像有可能会使用I帧之前的图像做运动参考。

AVC/H.264定义了4种不同的Profile(类)：Baseline(基线类), Main(主要类), Extended(扩展类)和High Profile(高端类)（它们各自下分成许多个层）：

* Baseline Profile 提供I/P帧，仅支持progressive(逐行扫描)和CAVLC
* Extended Profile 提供I/P/B/SP/SI帧，仅支持progressive(逐行扫描)和CAVLC
* Main Profile 提供I/P/B帧，支持progressive(逐行扫描)和interlaced(隔行扫描)，提供CAVLC或CABAC
* High Profile （也就是FRExt）在Main Profile基础上新增：8x8 intra prediction(8x8 帧内预测), custom quant(自定义量化), lossless video coding(无损视频编码), 更多的yuv格式（4:4:4...）

层：



From: <http://forum.doom9.org/showthread.php?p=678891#post678891>

### 工具：

**测试工具：MSU VQMT**

下载URL: <http://compression.ru/video/quality_measure/video_measurement_tool_en.html>

FFMPEG：FFmpeg是一个开源免费跨平台的视频和音频流方案，它提供了录制、转换以及流化音视频的完整解决方案。它包含了非常先进的音频/视频编解码库libavcodec，为了保证高可移植性和编解码质量，libavcodec里很多codec都是从头开发的。

　　libpostproc：用于后期效果处理；

编译加入：

--enable-libpostproc \

--with-libpostproc-includes=/usr/local/include/postproc \

libav:

　　2011年3月13日，FFmpeg 项目新的维护开发组决定 ffmpeg 项目将改名为 libav，同时制定了一套关于项目继续发展和维护的规则。

X264:

X264是一款从2004年有法国大学生发起的开源H.264编码器，对PC进行汇编级代码优化，舍弃了一些性能效率比不高的功能来提高编码效率，它被FFMPEG作为引入的.264编码库，也被移植到很多DSP嵌入平台。

用jeeb版的x264可以连着音频同时编码,但是AAC是FAAC:

<http://x264.fushizen.eu/>

mencode:

mencoder 是一款命令行方式的视频处理软件，是Mplayer自带的编码工具。mencoder支持几乎所有的格式的视频转换，可以将任意格式转换到任意格式，转换功能可以说是相当强大。目前市面上流行的格式转换器，都是基于mencoder开发的GUI，比如暴风转码，格式工厂等。

FFDShow(后期更好):

是一款全能的DirectShow解码、编码器,可以解压缩常见的视频格式和几乎所有的音频格式, 它还提供了丰富的加工处理选项,可以锐化画面,调节画面的亮度,它还支持诸多字幕格式.它能让音频和视频播放更流畅。

MP4Box :

GPAC 包括一个多媒体播放器 Osmo4 以及多媒体打包工具 MP4Box

MP4BOX——用于给3GP、MP4加hint，合并视频等

可视化的转码工具：

Mediacoder，MeGui

### 示例：

FFmpeg -y -threads 2 -i test.avi -vcodec libx264 -psnr -s 480x272 -r 25 -flags +loop -cmp +chroma -deblockalpha 0 -deblockbeta 0 -crf 24 -bt 256k -refs 1 -coder 0 -me umh -me\_range 16 -subq 5 -partitions +parti4x4+parti8x8+partp8x8 -g 250 -keyint\_min 25 -level 30 -qmin 10 -qmax 51 -trellis 2 -sc\_threshold 40 -i\_qfactor 0.71 -acodec libfaac -ab 128k -ar 48000 -ac 2 test120711\_1.mp4

使用FFmpeg压制H.264视频 ，非常棒，官网源码：

FFmpeg -threads 4 -i INPUT -r 25 -vcodec libx264 -s 576x422 -flags +loop -cmp +chroma -deblockalpha 0 -deblockbeta 0 -crf 24 -bt 256k -refs 1 -coder 0 -me umh -me\_range 16 -subq 5 -partitions +parti4x4+parti8x8+partp8x8 -g 250 -keyint\_min 25 -level 30 -qmin 10 -qmax 51 -trellis 2 -sc\_threshold 40 -i\_qfactor 0.71 -acodec libfaac -ab 128k -ar 48000 -ac 2 OUTPUT

第一阶段可行(改进版)：

FFmpeg -y -threads 2 -i test.avi -vcodec libx264 -psnr -s 480x272 -r 20 -flags +loop -deblockalpha 2 -deblockbeta 3 -partitions +parti4x4+parti8x8+partp8x8+partb8x8 -qmin 2 -qmax 41 -acodec libfaac -ab 128k -ar 48000 -ac 2 test120711\_6.mp4

块效应相对少

更进版：

x264 --preset fast --vf resize:576,422 --fps 25 --crf 26 --threads 2 --bitrate 500 --tune film --profile high -o x264\_e\_ok.mp4 test.avi

质量较满意：

x264 --preset fast --fps 25 --crf 19 --ref 9 --qpmin 0 --qpmax 28 --threads 2 --bitrate 500 --tune film --profile high -o youku\_x264\_4x4.mp4 youku.mp4

FFmpeg -y -threads 2 -i youku.mp4 -vcodec libx264 -psnr -b 500k -s 576x422 -r 25 -bf 2 -coder ac -flags +loop -cmp +chroma -deblockalpha 0 -deblockbeta 0 -crf 20 -refs 9 -me umh -me\_range 16 -subq 5 -partitions +parti4x4+parti8x8+partp8x8 -g 250 -keyint\_min 25 -level 30 -qmin 0 -qmax 26 -trellis 2 -sc\_threshold 40 -i\_qfactor 0.71 -acodec libfaac -ab 128k -ar 48000 -ac 2 test120713\_youku.mp4

接近YOUKU高清，文件稍大一点:

x264 --preset fast --fps 25 --crf 19 --ref 9 --qpmin 2 --qpmax 30 --threads 2 --bitrate 500 --tune film --profile high -o youku\_x264\_4x4.mp4 youku.mp4

**恒定质量方式CRF(constant ratefactor)**

　　默认参数——

--profile main --preset medium --tune ssim --crf 26.0 --threads 2 --bframes 3 --ref 3 --merange 16 --me umh --subme 9 --b-adapt 2 --trellis 2 --ssim

 　　实际的全部参数为——

cabac=1 / ref=3 / deblock=1:0:0 / analyse=0x1:0x111 / me=umh / subme=9 / psy=0 / mixed\_ref=1 / me\_range=16 / chroma\_me=1 / trellis=2 / 8x8dct=0 / cqm=0 / deadzone=21,11 / fast\_pskip=1 / chroma\_qp\_offset=0 / threads=2 / sliced\_threads=0 / nr=0 / decimate=1 / interlaced=0 / bluray\_compat=0 / constrained\_intra=0 / bframes=3 / b\_pyramid=2 / b\_adapt=1 / b\_bias=0 / direct=1 / weightb=1 / open\_gop=0 / weightp=2 / keyint=250 / keyint\_min=23 / scenecut=40 / intra\_refresh=0 / rc\_lookahead=40 / rc=crf / mbtree=1 / crf=26.0 / qcomp=0.60 / qpmin=0 / qpmax=69 / qpstep=4 / ip\_ratio=1.40 / aq=2:1.00

**指定码率方式VBR（Variable bitrate）**

 　　默认采用1000Kbps。为了避免其他干扰，采用一次编码模式（abr）。

 　　默认参数——

--profile main --preset medium --tune ssim --bitrate 1000 --threads 2 --bframes 3 --ref 3 --merange 16 --me umh --subme 9 --trellis 2 –ssim

x264 --profile high --tune film --preset fast --level 5.1 --psnr --ssim --vf resize:672,378 --bitrate 500 --ref 9 --bframe 2 --b-adapt 2 --me umh --subme 9 --trellis 2 --fps 25 --frames 8000 --output tt\_x264\_4.mp4 tong.mp4

(FFMPEG最好加入-qmin -qmax)

 　　实际的全部参数为——

cabac=1 / ref=3 / deblock=1:0:0 / analyse=0x1:0x111 / me=umh / subme=9 / psy=0 / mixed\_ref=1 / me\_range=16 / chroma\_me=1 / trellis=2 / 8x8dct=0 / cqm=0 / deadzone=21,11 / fast\_pskip=1 / chroma\_qp\_offset=0 / threads=2 / sliced\_threads=0 / nr=0 / decimate=1 / interlaced=0 / bluray\_compat=0 / constrained\_intra=0 / bframes=3 / b\_pyramid=2 / b\_adapt=1 / b\_bias=0 / direct=1 / weightb=1 / open\_gop=0 / weightp=2 / keyint=250 / keyint\_min=23 / scenecut=40 / intra\_refresh=0 / rc\_lookahead=40 / rc=abr / mbtree=1 / bitrate=1000 / ratetol=1.0 / qcomp=0.60 / qpmin=0 / qpmax=69 / qpstep=4 / ip\_ratio=1.40 / aq=2:1.00

以前的：

H.263

FFmpeg -y -i test.avi -vcodec h263 -acodec libfaac -s 352x288 -b 80k -maxrate 250k -minrate 30k -bufsize 100k -r 20 -ac 1 -ar 22050 -ab 32k -coder 1 -qscale 6 OUT-263.3gp

H.264

FFmpeg -y -i test.avi -vcodec libx264 -acodec libfaac -s 480x272 -b 80k -maxrate 250k -minrate 30k -bufsize 100k -r 20 -ac 1 -ar 22050 -ab 32k -coder 1 -qscale 6 OUT.mp4

**其他办法：**

也可以这样写（二次编码高级方法）：

x264 --pass 1 --slow-firstpass --tune psnr --crf 24 -o 100.mkv 100.avs

x264 --pass 2 ---tune psnr -bitrate 500 -o 100.mkv 100.avs

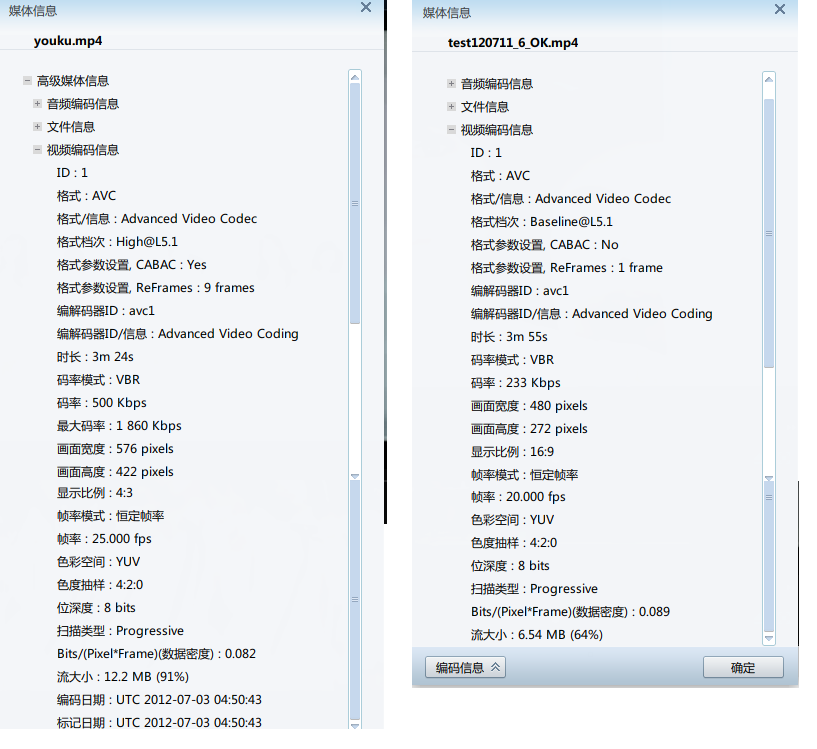
这段命令的意思是：将100.avs脚本视频，通过一次crf动态质量控制，二次平均码率控制，以psnr最佳量化信噪比，生成MKV文件。转码时间会长一些。

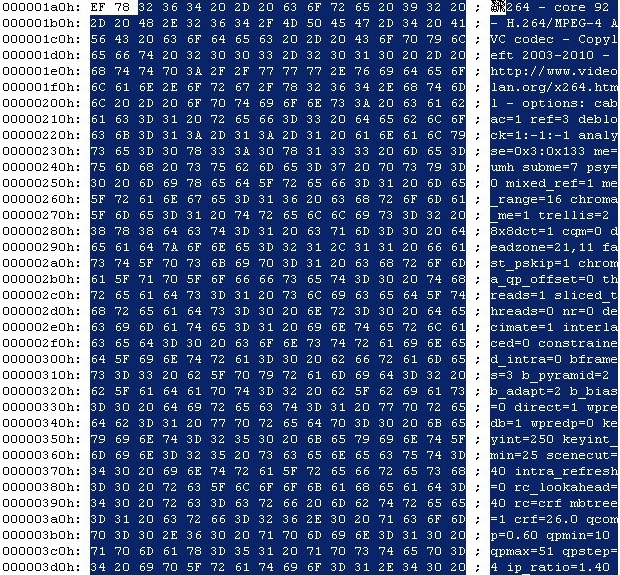
通过此命令，质量体积控制可以达到最佳状态。可以生成高质量低码率的网络视频。

X264使用说明：

<http://nmm-hd.org/doc/index.php?title=X264%E8%A8%AD%E5%AE%9A&variant=zh#qpmin>

<http://bbs.chinavideo.org/viewthread.php?tid=14145>





生产线：

PC转码：

x264 --preset faster --fps 25 --crf 19 --ref 9 --qpmin 2 --qpmax 30 --threads 2 --bitrate 500 --tune film --profile high --level 5.1 --psnr --ssim -o aa.mp4 a.flv

手机终端转码：

x264 --preset faster --fps 25 --crf 19 --ref 2 --qpmin 2 --qpmax 30 --threads 2 --bitrate 300 --tune film --profile baseline --level 3.0 --psnr --ssim -o aa\_mobile.mp4 a.flv

### 重要参数描述：

Coder ac: 使用CABAC

vlc variable length coder / huffman coder

ac arithmetic coder

raw raw (no encoding)

rle run-length coder

deflate deflate-based coder

-bf 3: 使用B帧，连续的3个P帧用B帧替换

-crf（Constant Ratefactor） 值18基本为无损，19-21.5为高质量，22-26为中等质量。与码率-bt互斥。crf会智能分析哪些是重要帧，哪些是次要帧，重要帧会得到更多的字节。

默认可以给20

refs：控制解码图片缓冲（DPB：Decoded Picture Buffer）的大小。范围是从0到16。总之，此值是每个P帧可以使用先前多少帧作为参照帧的数目（B帧可以使用的数目要少一或两个，取决于它们是否作为参照帧）。可以被参照的最小ref数是1。该参数对质量和压缩比都有提高，值越大它的压缩比越大。可选择9

qmin-qmax最小/大量化值: 范围0-51，值越小效果越好，qmax:通常30~40，默认为51是最大值，效果最差。

-partitions +parti4x4+parti8x8+partp8x8：定义宏块大小

-psnr：编码完成后输出：SSIM与PSNR值



-b 500k：码率

-r 25：FPS，每秒帧数

-g 250：两个I frame之间形成一个GOP，GOP的第一帧为I帧

-flags +loop -deblockalpha -1 –deblockbeta -1：去块效应

-me\_method umh 运动估算：

-me\_range 16 动估算模式选项中选择Uneven Multi-Hexagon或Exhaustive时的搜索半径，最大64

-cmp +chroma

移动终端：

--profile baseline

--level 3 PC终端最高4.1，手机终端3.0

--vbv-bufsize 2500

--vbv-maxrate 2500

最简单的编码方式：--crf=20

### 测试序列

常用测试序列选择：

Coastguard：物体的相对运动和镜头移动

Flower：物体的剧烈运动和镜头的快速移动

Garphone：物体的快速转换

Foreman：物体转换和镜头移动

Mobile&calendar：物体的多种运动和镜头移动。

football序列为物体的剧烈运动,

foreman为物体移动和镜头一般性的旋转和平移,

carphone为物体的一般运动,

container为物体的缓慢平移。

suzie序列运动较平缓,

football序列运动剧烈,

foreman序列运动剧烈程度居中

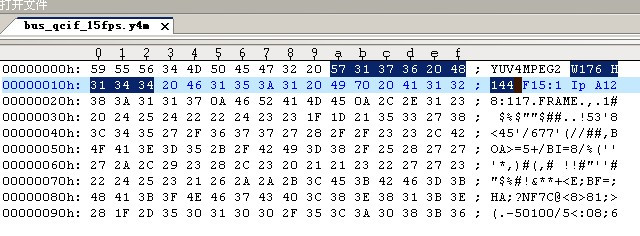
一个平缓，一个剧烈运动，一个适中来测试视频编码。

测试序列下载：

<http://media.xiph.org/video/derf/>

后缀.y4m文件描述：

资料：YUV4MPEG2这种文件格式是一种以头文件存储视频规格的未压缩视频序列。简单来说，在原始的yuv序列的起始和每一帧的头部都加入了纯文字形式的视频参数信息，包括分辨率、帧率、逐行/隔行扫描方式、高宽比(aspect ratio)，以及每一帧起始的”FRAME ”标志位。



mencoder \*\*.y4m -ovc raw -of rawvideo -vf format=i420 -o \*\*.yuv

### FFmpeg耻辱柱(Hall Of Shame)：

目前FFmpeg没有维护耻辱柱，转而在libav上面继续发布。<http://libav.org/shame.html>

由于FFmpeg是在LGPL/GPL协议下发布的（如果使用了其中一些使用GPL协议发布的模块则必须使用GPL协议），任何人都可以自由使用，但必须严格遵守LGPL/GPL协议。目前有很多播放软件都使用了FFmpeg的代码，但它们并没有遵守LGPL/GPL协议，没有公开任何源代码。我们应该对这种侵权行为表示耻辱。

2009年加入FFmpeg的播放软件：**暴风影音、QQ影音**都在其列。

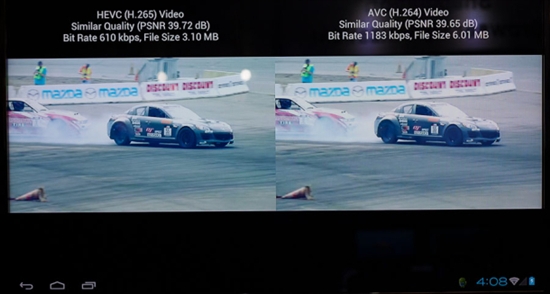
2009年5月，网友cehoyos下载了暴风影音软件，解压之后发现其安装程序内包含了大量的开源和私有解码器：avcodec，avformat，avutil，x264，xvid，bass，wmvdmod等，之后暴风影音被正式加入到FFmpeg耻辱名单。

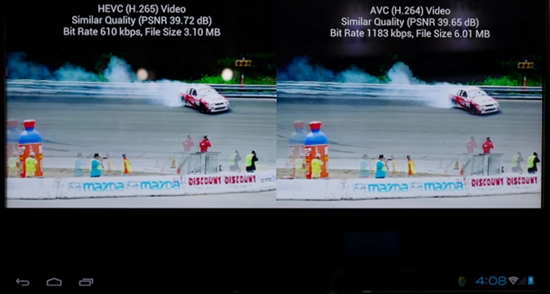
2009年11月，网友roo\_zhou向FFmpeg举报，指出QQ影音的credit只给出了修改的FFmpeg源码下载，声称是LGPL许可证。但实际是修改过的ffdshow，采用的是GPL许可证，之后QQ影音被正式加入到FFmpeg耻辱名单之列。

## H.265的发展:

巴塞罗那MWC 2012上，除了骁龙S4处理器之外，高通还展示了下一代视频编码标准H.264继任者H.265的效果。就像H.264另一个名称是AVC（Advanced Video Codec）一样，H.265的代号是HEVC（High Effciency Video Codec）。由于网络传输流媒体的分辨率越来越大，对于带宽要求也越来越高，H.265可提供类似质量下更小的码率。

高通演示的HEVC视频对比对象为同一片源的AVC，**在PSNR（峰值信噪比）这一目前最广泛使用的客观评价标准类似的情况下**（HEVC为39.72，AVC为39.65，分辨率同为800\*480），HEVC视频的码率为610kbps，文件大小3.10MB；而AVC视频的码率达到了1183kbps，文件大小也有6.01MB。据高通工作人员介绍，**HEVC可比AVC节省40-45%的码率**。





## 参考：

论坛：

http://forum.doom9.org/

<http://www.nmm-hd.org/newbbs/viewforum.php?f=8>

H.264学习笔记之一（层次结构，NAL，SPS）

<http://blog.163.com/ld081055@126/blog/static/11818691520117314196122/>

利用ffmpeg来进行视频解码的完整示例代码（H.264）

http://www.rosoo.net/a/201006/9659.html?1342220931

利用ffmpeg编码参数示例：

<http://bbs.chinavideo.org/viewthread.php?tid=14145>

X264設定

http://www.g0759.com/html/JingPinZhiZuo/92.html

<http://nmm-hd.org/doc/index.php?title=X264%E8%A8%AD%E5%AE%9A&variant=zh>

自适应算术编码的FPGA

<http://wenku.baidu.com/view/3147b6669b6648d7c1c7469f.html>

CABAC原理及其实现笔记

<http://wenku.baidu.com/view/10b2440a581b6bd97f19ea59.html>

AviSynth入门与应用指南

<http://meguizhcn.sourceforge.net/blog/?p=767>

<http://www.ffmpeg.com.cn/index.php/AviSynth%E5%85%A5%E9%97%A8%E4%B8%8E%E5%BA%94%E7%94%A8%E6%8C%87%E5%8D%97>

AviSynth脚本实例

<http://meguizhcn.sourceforge.net/blog/?p=393>