目录

[视频制作基本知识 1](#_Toc518744554)

[1.封装格式与媒体格式 1](#_Toc518744555)

[2.分辨率，帧率，码率 1](#_Toc518744556)

[3.RGB模型 vs YUV模型 1](#_Toc518744557)

[4.色深 2](#_Toc518744558)

[5.色度半采样 2](#_Toc518744559)

[6.空间上的低频与高频：平面，纹理和线条 2](#_Toc518744560)

[7.时间上的低频与高频：动态 2](#_Toc518744561)

[8.清晰度与画质简述 2](#_Toc518744562)

[播放器相关技术知识 2](#_Toc518744563)

[1.播放器的工作流程：分离，解码，渲染 2](#_Toc518744564)

[2.硬解的定义与分类 3](#_Toc518744565)

[3.YUV->RGB转换过程中的细节 4](#_Toc518744566)

[4.硬解的优劣与选择 4](#_Toc518744567)

[5. 图像格式的标识与查看方法 4](#_Toc518744568)

# 视频制作基本知识

## 1.封装格式与媒体格式

封装格式是一个包裹，包含了各种音频、视频、字幕等独立子内容，子内容也称轨道track。

常见封装格式：  
MP4，MKV，WMV

常见视频轨道媒体格式：  
H.264（AVC，AVC1），H.265（HEVC），RealVideo，VC-1（微软）。

常见音频轨道媒体格式：  
无损：FLAC，ALAC，TrueHD，DTS-HD  
有损：AAC，MP3，AC3，DTS（Core）

## 2.分辨率，帧率，码率

码率：视频文件体积除以时间。单位一般是Kbps(Kbit/s)或者Mbps(Mbit/s)

## 3.RGB模型 vs YUV模型

RGB：定义略，PNG和BMP图片基于此模型

YUV：又被称为亮度-色度模型（Luma-Chroma）。它是通过数学转换，将RGB三个通道，转换为一个代表亮度的通道(Y,又称为Luma)，和两个代表色度的通道(UV，并成为Chroma)。YUV模型下，还有不同的实现方式。举个用的比较多的YCbCr模型：它把RGB转换成一个亮度(Y)，和 蓝色色度(Cb) 以及 红色色度(Cr)。JPEG图像和大多数视频格式基于YCbCr模型。

YUV模型的优势：  
1、人眼对亮度的敏感度远高于色度，因此人眼看到的有效信息主要来自于亮度。YUV模型可以将绝大多数的有效信息分配到Y通道。UV通道相对记录的信息少的多。相对于RGB模型较为平均的分配，YUV模型将多数有效信息集中在Y通道，不但减少了冗余信息量，还为压缩提供了便利  
2、保持了对黑白显示设备的向下兼容  
3、图像编辑中，调节亮度和颜色饱和度，在YUV模型下更方便。

## 4.色深

色深(bit-depth)，就是我们通常说的8bit和10bit，是指每个通道的精度。实际上就是每个通道表达数值所占的空间大小。色深越大，表达数值所占空间越大，颜色的分级越细腻。

## 5.色度半采样

色度半采样(Chroma Sub-Sampling)：在YUV模型的应用中，Y和UV的重要性是不等同的。图像视频的实际储存和传输中，通常将Y以全分辨率记录，UV以减半甚至1/4的分辨率记录。

420采样：配合YUV格式，常常被写作yuv420。这种采样是Y保留全部，UV只以(1/2) x (1/2)的分辨率记录。

444采样：或者yuv444。YUV三个平面全是满分辨率。H264格式编码444格式，需要High 4:4:4 Predictive Profile（简称Hi444pp）。

## 6.空间上的低频与高频：平面，纹理和线条

在视频处理中，空间(spatial)的概念指的是一帧图片以内（你可以认为就是一张图所呈现的二维空间/平面）。跟时间(temporal)相对；时间的概念就强调帧与帧之间的变换。

亮度变化较快，变动幅度大的区域，我们称之为高频区域。否则，亮度变化缓慢且不明显的区域，我们称为低频区域。典型的低频区域，或者就叫做平面（平坦的部分）。亮度几乎没有变化。亮度呈现跳跃式的突变，这种高频区域我们称之为线条。亮度频繁变化，幅度有高有低，这种高频区域我们称为纹理。有时候，线条和纹理（高频区域）统称为线条，平面（低频区域）又叫做非线条。色度平面，高频低频，线条等概念也同样适用，就是描述色度变化的快慢轻重。一般我们所谓的“细节”，就是指图像中的高频信息。

## 7.时间上的低频与高频：动态

动态的概念无需多解释；就是帧与帧之间图像变化的强弱，变化频率的高低。一段视频如果动态很高，变化剧烈，我们称为时间复杂度较高，时域上的高频信息多。否则如果视频本身舒缓多静态，我们称为时间复杂度低，时域上的低频信息多。

## 8.清晰度与画质简述

清晰度的定义我更倾向于这样一个说法：图像或视频中，原生、有效的高频信息量的多少。  
原生，强调这种清晰度是非人工添加的；有效；强调细节本身有意义，而不是毫无意义的噪点特效。

# 播放器相关技术知识

## 1.播放器的工作流程：分离，解码，渲染

分离，指的是拿到媒体文件(MKV/MP4/MKA)等，先收集相关的文件（包括外挂音轨、字幕），然后将所有轨道拆开，拆分成单独的内容。视频流、音频流、字幕、章节信息，等等。负责执行分离的模块滤镜，叫做分离器(splitter/demuxer)。分离器现在能用的基本上只有LAV/ffmpeg了（这俩几乎可以算一家），以前还有个Haali，然而停止更新已久，不能适应HEVC时代了。分离器一般不耗费运算性能。因为它只是简单地收集、拆分和选择。

解码，指的是将分离器丢来的各种原生压缩格式，比如H264/H265的视频，FLAC/AAC的音频，解码为非压缩的格式，比如视频是YUV/RGB（相当于bmp），音频是PCM（相当于wav），然后丢给下游模块。负责解码的模块滤镜称为解码器(decoder)。常见的有LAV/ffmpeg, ffdshow(同样停止更新了)。当解码器能完全解码一个轨道中所有有效信息的时候，我们成为完全解码（现在绝大多数情况是如此），否则称为不完全解码。如果解码器可以将最原始的数据，或者更高精度（比如有时候为了方便，将10bit转为16bit）输出给下游，我们称为全精度输出；否则，解码器会试图降低精度输出，我们称为低精度输出。少数时候，我们会让解码器做一些转换（比如vcb-s新播放器教程中，让lav解码器做YUV->RGB的转换），我们称为转换输出。

渲染，指的是将解码后的数据，在pc硬件上（显示器、扬声器）进行播放。负责渲染的模块我们称之为渲染器(Render)，视频渲染器主流有EVR（Enhanced Video Render, 微软送的）以及madVR(madshi Video Render)。音频渲染器一般都是系统自带的（同样是微软送的），也有可以自定义的。比如MPC播放器有MPC Audio Render，可以支持类似wasapi输出等其他功能。因为显示器是RGB显示，而解码后的视频多为YUV格式，渲染器一般也需要负责将YUV转换为RGB，并保证输出的图像大小跟播放窗口吻合。多数播放器自带的滤镜（mpc/pot都有很多调色之类的功能），显卡的加成，以及SVP，都作用于解码器和渲染器中间。它们接过解码器解码的数据，对其进行处理，然后将处理后的数据送给渲染器。因为渲染器是需要借助显卡进行图形运算，YUV数据基本上需要先进入显存，所以显卡可以检测到丢来的YUV数据，对其进行“优化”。同样需要当心的是，这些滤镜和处理，往往入口精度低，处理精度也低。导致的后果就是解码器被迫低精度输出，给这些滤镜低精度处理，从而大幅降低视频精度，导致色带色块问题。

## 2.硬解的定义与分类

硬解是为了缓解高分辨率新编码面世初期，CPU不堪重负的解码压力，而诞生的技术。如果说软解的定义是：利用CPU通用运算能力，进行解码，那么硬解的定义可以这么说：不利用CPU通用运算能力，而是依赖其他集成电路，无论是否特制，来进行解码。

硬解现在比较常见的是以下种类：

1）更古老的时候，有些显卡没办法进行完全解码，只是帮助计算部分解码过程中的运算，那么可以归为“硬件加速”。

2）DXVA(DirectX Video Acceleration)，比较古老的方案了。Windows XP以及之前系统上流行的。上古ffdshow的硬解就是利用DXVA。DXVA规范下容易出现不完全解码，导致画质降低。Vista以后，渐渐地被抛弃。

3）DXVA2，目前主流的硬解方式。主要由GPU来实现，但是并非利用GPU的流处理器，INA三家都是使用了单独附在GPU芯片上的一块专职电路来完成。GPU硬解能力往往不与显卡游戏性能相关，而与搭载的专职电路先进与否相关。使用DXVA解码，都需要先将视频数据（压缩的格式）传输到显存中，然后再让GPU进行解码。DXVA2有两种实现方式：native和copy-back。区别是解码后的数据是否还要传回内存。因为有些滤镜，比如SVP，比如LAV的转格式，必须依赖CPU+内存进行工作。不传回来没办法继续处理。copy-back保证了硬解的流程类似软解，可以不漏下任何后处理。而代价是传来传去必定降低性能，增加能耗。需要注意的是，即便用native，也可能导致解码后的数据被“优化”，因为有些处理，包括播放器、显卡驱动带的那些，是可以完全作用在GPU环境中的。

4）Intel Quick Sync是集成在CPU中的逻辑电路承担的。注意的是这玩意并非隶属于Intel的集显，而是CPU的直属。它直接读写内存，运行表现和软解非常类似。Intel Quick Sync堪称速度快，能耗低。

5）NVIDIA CUVID，是基于NV自己的接口，写的一个类似DXVA2(copy-back)的升级版。

## 3.YUV->RGB转换过程中的细节

如果解码过程是完全解码，也不主动添加播放器调校和驱动增强，渲染的环节决定了最终成品的画质。造成画质区别的可以说就三点：缩放算法，运算精度，和抖动算法。任何试图优化渲染器效果的尝试，都应该从这三个方面着手。

缩放算法造成的区别，比较好理解。用双线性算法（多数播放器默认算法）不如nnedi3算法。

精度，是指运算的过程中，参与运算的数，有效位数的高低。千万不要以为显示器是8bit，就认为8bit 整数 的片源精度/处理精度是足够的。播放过程中，应该尽量减少RGB-YUV互转的次数，每一次转换都要做一次计算与取整，都会导致实际精度降低。

抖动算法(Dithering Algorithm)，通常出现在高精度转低精度中。在数字图像高转低处理中，全部四舍五入不见得是好习惯。抖动算法通过科学的添加噪点，来掩盖精度的不足。如Floyd–Steinberg 抖动算法。

以下是渲染器的工作流程，并用蓝色标注出可能造成画质差别的地方：

1、渲染器从解码器那里获取YUV数据。注意拿到的数据可能是全精度，也可能是降精度，取决于渲染器接口类型；

2、播放器和显卡驱动可能会试图“优化”画面；

3、如果不是YUV444格式的，渲染器会先将UV平面放大到Y平面的大小。这个步骤称为Chroma upscaling；

4、将YUV444的数据，转为RGB。转换的过程势必需要浮点运算（YUV->RGB一些参与运算的常数是浮点数）；

5、播放器或者渲染器将RGB用特定的算法缩放到播放窗口大小。这个步骤称为Image Upscaling(图像放大)/Downscaling(图像缩小)；

6、因为4的步骤中，必须以浮点数运算，而输出结果一定是RGB 8bit整数，因此输出之前必须有一个高转低的过程。

2~6每一步都涉及数字运算，因此有运算精度的区别。

RGB 8bit 包括RGB24和RGB32。RGB32多一个透明层通道，看似带了个没用的东西，但是因为计算机更喜欢2的次方，所以部分运算下RGB32比RGB24快。在视频播放中，这两个格式几乎完全等同；互转也人畜无害（加一个空的透明度通道 vs 去掉透明度通道）。

## 4.硬解的优劣与选择

软解吃力的硬解解不了，硬解解得了的软解解的飞起，那我们为什么要冒着各种潜在风险去开硬解呢？

## 5. 图像格式的标识与查看方法

针对个别播放器而使用的一项功能，此处略。

## 6.动漫画面区别于常规录制视频的特殊性

区别于录制视频，比如电影之类的，动漫、CG等有着自己的特殊性。总结起来就两点：1、噪点少，2、线条非常突兀。视频拍摄，限制于器材水准，噪点是不可避免的，在后续制作等过程中也难以完全去除。而动漫天生可以0噪点，动漫中的噪点更多是数字图像处理中主动加上去的。

噪点的一大作用就是极大地降低视频处理和压制，对于精度的需要。说的简单点：高噪点的视频不怕低精度，反之亦然。噪点会使得人眼对图像锐利度等差异不敏感，或者说，缩放算法造成的区别，变得不太可见。当有噪点存在的时候，主打高精度、优秀缩放算法的播放器，优势将不再明显。从另一个方面讲，面对较少噪点、较为突出线条的动漫，对播放器的精度和缩放算法提出的要求就很高。编码器也是一样的道理，动漫非常需要10bit x264/x265这样原生高精度的编码器来提升画质。

噪点作为一种高频信息，需要浪费成倍的码率。在今天10bit编码可以不增加（甚至减少）码率完美解决问题的前提下，我们为什么要用10年前的理解呢？