MeGUI 教程 0： 视频音频格式基础知识

本教程意在讲述一些视频音频制作的基础知识和术语。

1、封装格式(MP4/MKV…) vs 媒体格式(H.264/FLAC/AAC…)

你下载的视频文件最多的就是这些。这些文件其实类似一个包裹。它的后缀则是包裹的包装方式。这些包裹里面，包含了视频，音频，字幕等。当播放器在播放的时候，首先对这个包裹进行拆包（专业术语叫做分离/splitting）,把其中的视频、音频等拿出来，再进行播放。

既然它们只是一个包裹，就意味着这个后缀不能保证里面的东西是啥，也不能保证到底有多少东西。包裹里面的每一件物品，我们称之为轨道(track)，一般有这么些：

视频(Video): 一般来说肯定都有，但是也有例外，比如 mka 格式的外挂音轨，其实就是没视频的 mkv。
音频(audio)：一般来说也肯定有，但是有些情况是静音的，就没必要带了。
章节(Chapter): 蓝光原盘中自带的分段信息。如果文件带上了，那么你可以在播放器中看到带章节的效果：

potplayer 右键画面，选项-播放-在进度条上显示书签/章节标记
mpc-hc 右键画面，选项-调节-在进度条显示章节标记

字幕(Subtitles)：有些时候文件自带字幕，并且字幕并非是直接整合于视频的硬字幕，那么就是一起被打包在封装容器中。
其他可能还有附件等，不一一列举。每个类型也不一定只有一条轨道，比如经常见到带多音轨的 MKV。

每个轨道，都有自己的格式。比如大家常说的，视频是 H.264，音频是 AAC，这些就是每个轨道的格式。
视频的格式，常见的有 H.264(可以细分为 8bit/10bit)，H.265(当前也有 8bit/10bit 之分)，RealVideo(常见于早期 rm/rmvb)，VC-1(微软主导的，常见于 wmv)。基本上，H.264=AVC=AVC1, H.265=HEVC。
音频的格式，常见的有 FLAC/ALAC 这两种无损，和 AAC/MP3/AAC/DTS 这种有损。

MKV vs MP4，主要的区别在于：
1. MKV 支持封装 FLAC 作为音频，MP4 则不支持。但是 MP4 也可以封装无损音轨(ALAC，虽然普遍认为 ALAC 的效率不如 FLAC 优秀)
2. MKV 支持封装 ASS/SSA 格式的字幕，MP4 则不支持。一般字幕组制作的字幕是 ASS 格式，所以内封字幕多见于 MKV 格式

除此之外，这两个格式很大程度上可以互相代替。比如它们都支持封装 AVC 和 HEVC，包括 8bit/10bit 的精度。所以 MP4 画质不如 MKV 好，这种论断是非常无知的——它们完全可以封装一样的视频。

2、视频的基础参数：分辨率，帧率和码率。

视频是由连续的图像构成的。每一张图像，我们称为一帧(frame)。图像则是由像素(pixel)构成的。一张图像有多少像素，称为这个图像的分辨率。比如说 1920x1080 的图像，说明它是由横纵 1920x1080 个像素点构成。视频的分辨率就是每一帧图像的分辨率。

一个视频，每一秒由多少图像构成，称为这个视频的帧率(frame-rate)。常见的帧率有 24000/1001=23.976, 30000/1001=29.970, 60000/1001=59.940, 25.000, 50.000 等等。这个数字是一秒钟内闪过的图像的数量。比如 23.976，就是 1001 秒内，有 24000 张图像。视频的帧率是可以是恒定的(cfr, Const Frame-Rate)，也可以是变化的(vfr, Variable Frame-Rate)

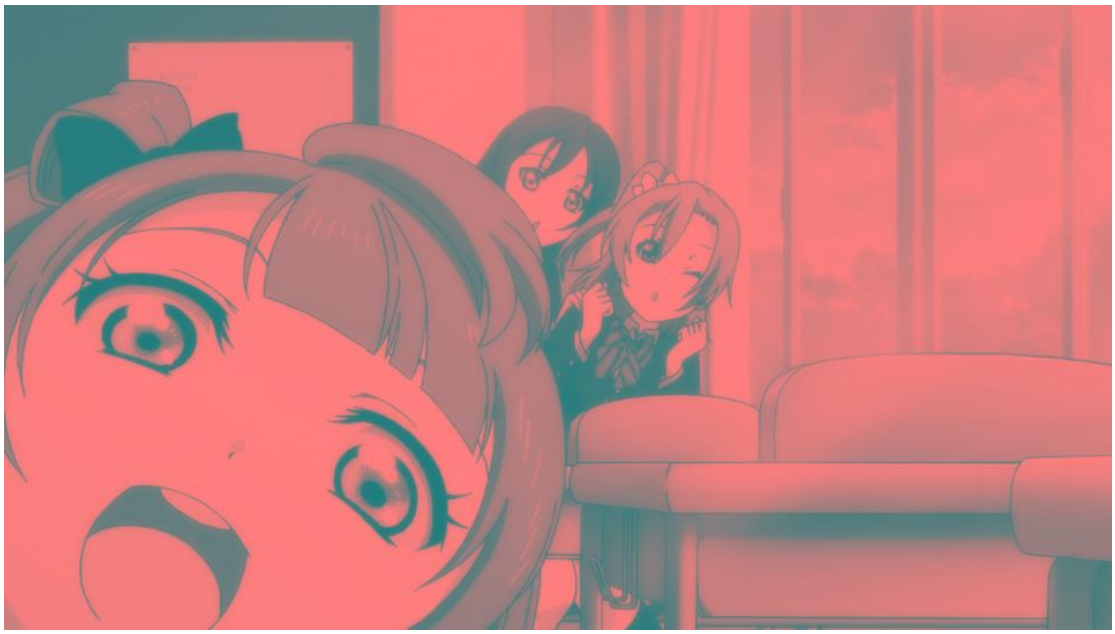
码率的定义是视频文件体积除以时间。单位一般是 Kbps(Kbit/s)或者 Mbps(Mbit/s)。注意 1B(Byte)=8b(bit)。所以一个 24 分钟，900MB 的视频：
体积：900MB = 900MByte = 7200Mbit
时间：24min = 1440s
码率：7200/1440 = 5Mbps = 5000 Kbps

3、图像的表示方法：RGB 模型 vs YUV 模型

光的三原色是红(Red)、绿(Green)、蓝(Blue)。现代的显示器技术就是通过组合不同强度的三原色，来达成任何一种可见光的颜色。图像储存中，通过记录每个像素红绿蓝强度，来记录图像的方法，称为 RGB 模型 (RGB Model)
常见的图片格式中，PNG 和 BMP 这两种就是基于 RGB 模型的。

比如说原图：

分别只显示 R G B 通道的强度，效果如下：



三个通道下，信息量和细节程度不一定是均匀分布的。比如说可以注意南小鸟脸上的红晕，在 3 个平面上的区分程度就不同——红色平面下几乎无从区分，造成区别的主要是绿色和蓝色的平面。外围白色的脸颊，三色都近乎饱和；但是红晕部分，只有红色饱和，绿色和蓝色不饱和。这是造成红色凸显的原因。

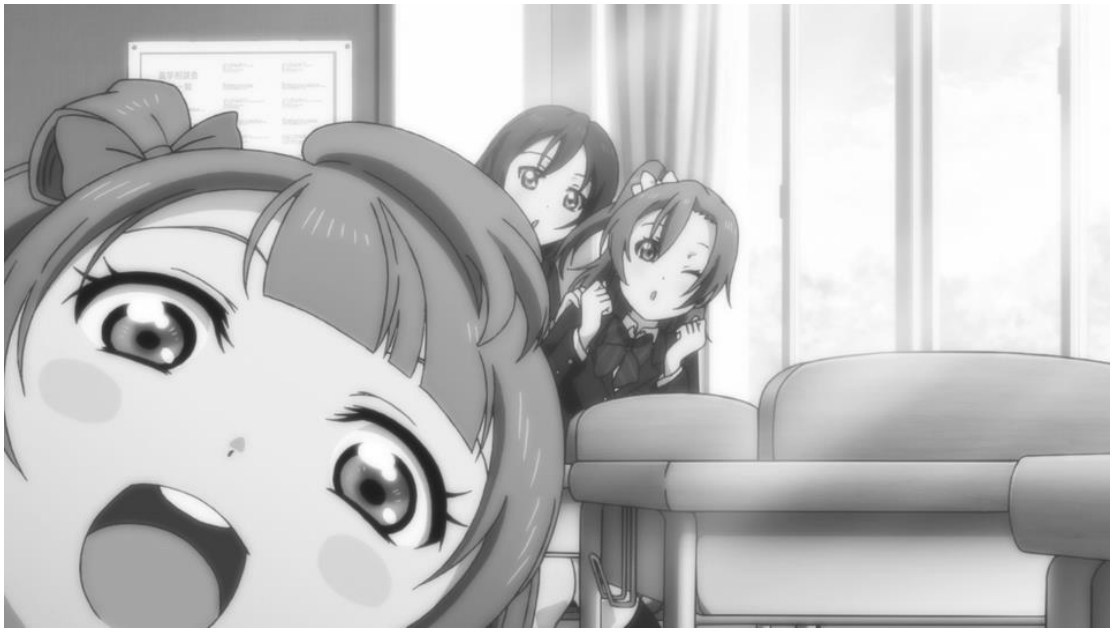
除了 RGB 模型，还有一种广泛采用的模型，称为 YUV 模型，又被称为亮度-色度模型（Luma-Chroma）。它是通过数学转换，将 RGB 三个通道，转换为一个代表亮度的通道(Luma)，和两个代表色度的通道(并成为 Chroma)。

举个形象点的例子：一家养殖场饲养猪和牛，一种记数方式是：（猪的数量，牛的数量）
但是也可以这么记录：（总数量=猪的数量+牛的数量，相差=猪的数量-牛的数量）。两种方法

之间有公式可以互转。

YUV 模型干的是类似的事儿。通过对 RGB 数据的合理转换，得到另一种表示方式。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)。转换背后复杂的公式大家不需要了解，只需要看看效果：

只有亮度通道：



只有蓝色色度：

只有红色色度：

在图像视频的加工与储存中，YUV 格式一般更受欢迎，理由如下：

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。

几乎所有的视频格式，以及广泛使用的 JPEG 图像格式，都是基于 YCbCr 模型的。播放的时候，播放器需要将 YCbCr 的信息转换为 RGB。这个步骤称为渲染（Rendering）

每个通道的记录，通常是用整数来表示。比如 RGB24，就是 RGB 各 8 个 bit，用 0~255 (8bit 的二进制数范围)来表示某个颜色的强弱。YUV 模型也不例外，也是用整数来表示每个通道的高低。

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。8bit 就是每个通道用一个 8bit 整数(0~255)代表，10bit 就是用 10bit 整数(0~1023)来显示。16bit 则是 0~65535

你的显示器是 8bit 的，代表它能显示 RGB 每个通道 0~255 所有强度。但是视频的色深是 YUV 的色深，播放的时候，YUV 需要通过计算转换到 RGB。因此，10bit 的高精度是间接的，它使得运算过程中精度增加，以让最后的颜色更细腻。

如何理解 8bit 显示器，播放 10bit 是有必要的呢：

一个圆的半径是 12.33m, 求它的面积，保留两位小数。
半径的精度给定两位小数，结果也要求两位小数，那么圆周率精度需要给多高呢？也只要两位小数么？
取 pi=3.14, 面积算出来是 477.37 平方米
取 pi=3.1416，面积算出来是 477.61 平方米
取 pi 精度足够高，面积算出来是 477.61 平方米。所以取 pi=3.1416 是足够的，但是 3.14 就不够了。

换言之，即便最终输出的精度要求较低，也不意味着参与运算的数字，以及运算过程，可以保持较低的精度。在最终输出是 8bit RGB 的前提下，10bit YUV 比起 8bit YUV 依旧具有精度优势的原因就在这里。事实上，8bit YUV 转换后，覆盖的精度大概相当于 8bit RGB 的 26%，而 10bit 转换后的精度大约可以覆盖 97%——你想让你家 8bit 显示器发挥 97%的细腻度么？看 10bit 吧。

8bit 精度不足，主要表现在亮度较低的区域，容易形成色带：

注意这图右边那一圈圈跟波浪一样的效果。这就是颜色精度不足的表现。如果你看的是 pdf 版本，你还应该可以注意到左边背景那网格状的效果，那也是精度不足造成的颜色过渡不自然。

10bit 的优势不只在于显示精度的提高，在提高视频压缩率，减少失真方面，相对 8bit 也有优势。这方面就不展开了。

5、色度半采样

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。这个手段被称为色度半采样(Chroma Sub-Sampling)。色度半采样可以有效减少传输带宽，和加大 UV 平面的压缩率。

我们平常的视频，最常见的是 420 采样。这种采样是 Y 保留全部，UV 只以(1/2) x (1/2)的分辨率记录。比如说 1920x1080 的视频，其实只有亮度平面是 1920x1080。两个色度平面都只有 960x540 的分辨率。

当然了，你也可以选择不做缩减。这种称为 444 采样。YUV 三个平面全是满分辨率。

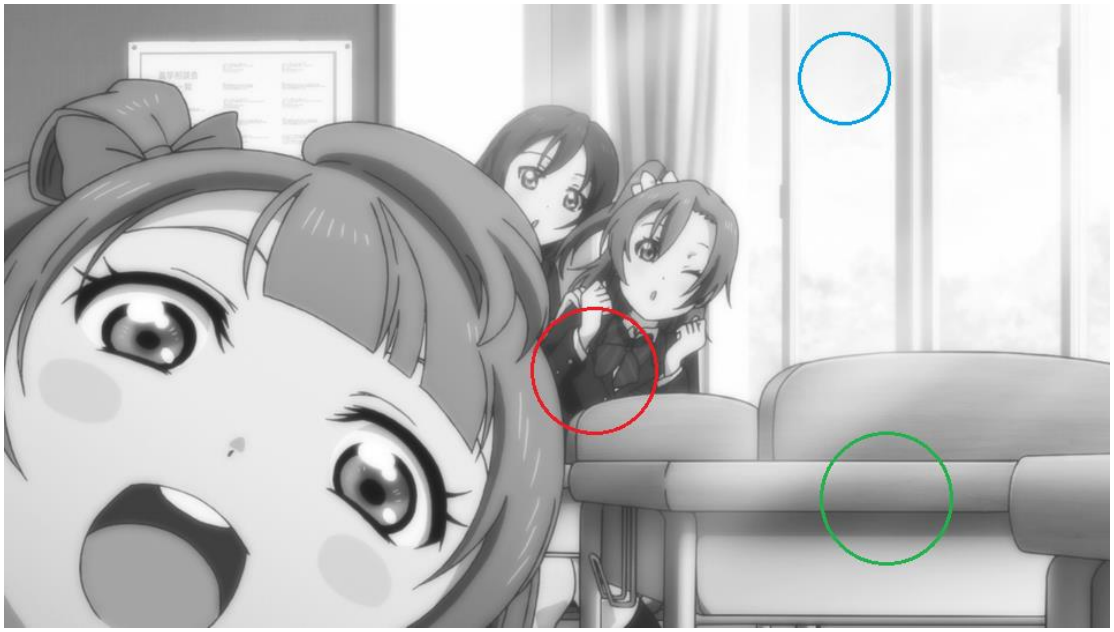
在做 YUV->RGB 的时候，首先需要将 UV 分辨率拉升到 Y 的分辨率（madVR 中允许自定义算法，在 Chroma Upscaling 当中），然后再转换到 RGB。做 RGB->YUV 的转换，也是先转换到 444，再将 UV 分辨率降低。

一般能拿到的片源，包括所有蓝光原盘，都是 420 采样的。所以成品一般也保留 420 采样。（除非需要在 444/RGB 平面下做处理，这样成品有必要保留 444 采样）

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。跟时间(temporal)相对；时间的概念就强调帧与帧之间的变换。

于是我们重新来看这张亮度的图：



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。

图中的蓝圈就是一块典型的低频区域。亮度几乎没有变化
绿圈中，亮度呈现跳跃式的突变，通常是一般亮度较高，但是线条部分的一排像素亮度明显低于周围，这种高频区域我们称之为线条。
红圈中，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，线条和纹理统称为线条，平面又叫做非线条。

这是亮度平面。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。

一般来说，一张图的高频信息越多，意味着这张图信息量越大，所需要记录的数据量就越多，编码所需要的运算量也越大。如果一个视频包含的空间性高频信息很多（通俗点说就是每一帧内细节很多），意味着这个视频的空间复杂度很高。

记录一张图片，编码器需要决定给怎样的部分多少码率。码率在一张图内不同部分的分配，叫做码率的空间分配。分配较好的时候，往往整幅图目视观感比较统一；分配不好常见的后果，就是线条纹理尚可；背景平面区域出现大量色带色块；或者背景颜色过渡自然，纹理模糊，线条烂掉。

7、时间上的低频与高频：动态

在视频处理中，时间(temporal)的概念强调帧与帧之间的变换。跟空间(spatial)相对。

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。

一般来说，一段视频的时域高频信息多，动态的信息量就大，所需要记录的数据量就越多，编码所需要的运算量也越大。但是另一方面，人眼对高速变化的场景，敏感度不如静态的图片来的高（你没有时间去仔细观察细节），所以动态场景的优先度可以低于静态场景。如何权衡以上两点去分配码率，被称为码率的时间分配。分配较好的时候，看视频无论动态还是静态效果都较好；分配不好的时候往往是静态部分看着还行，动态部分糊烂掉；或者动态部分效果过分的好，浪费了大量码率，造成静态部分欠码，瑕疵明显。