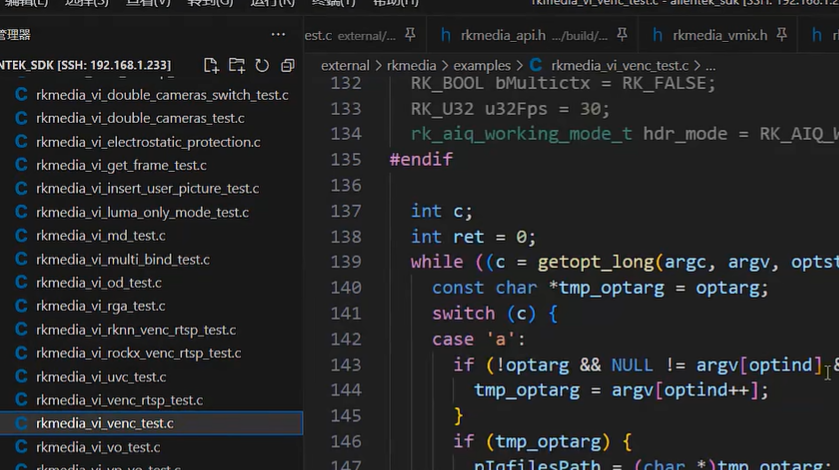
本节开始学习RKMedia的VENC（视频编码）模块：

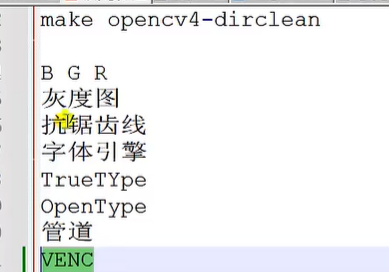
参考RKMedia手册：

"C:\Users\zhongqing\Desktop\笔记\08、RV1126参考资料\RV1126\_RV1109\Multimedia\Rockchip\_Developer\_Guide\_Linux\_RKMedia\_CN.pdf"

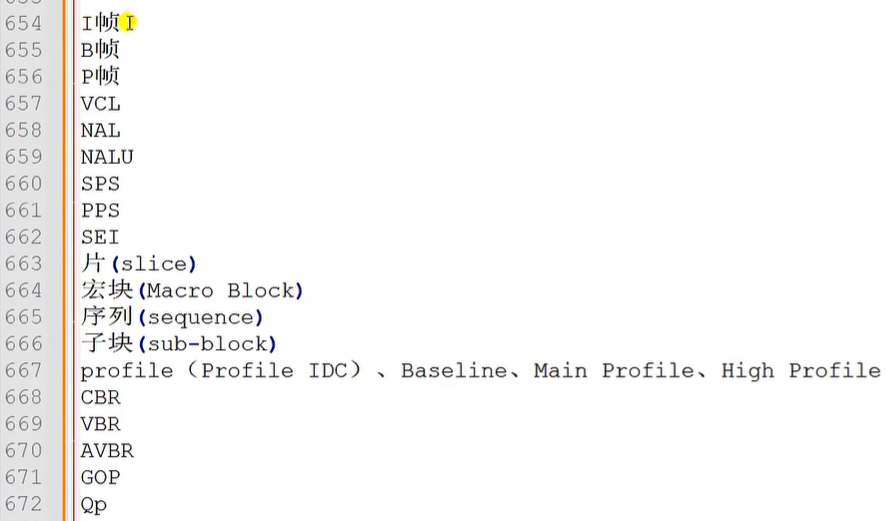
以及VENC例程：

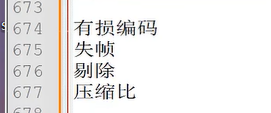


里面会出现一些词汇：

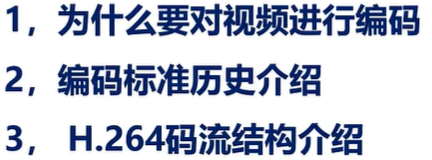


以及这些描述：

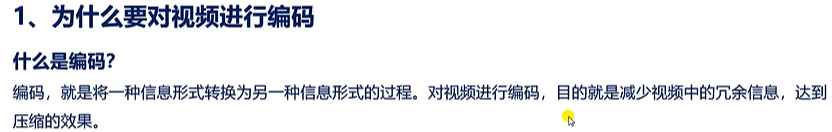




IMG_256



/\*

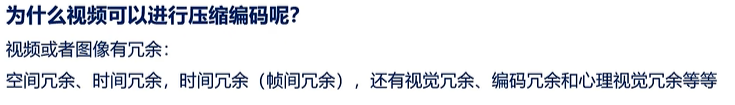


在计算机中，编码通常用于将文本图像、音频、视频不同类型的信息转化为计算机可识别处理的二进制数据。对视频进行编码实际就是进行压缩。（采取各种算法减少数据的容量，减少bit流，目的就是让体积变小从而减少存储空间，利于存储并减少带宽，便于传输。）

对于视频文件来说压缩编码时会有图像信息丢失，这种就称为有损编码。

如果编码时丢失的信息较少，编码后得到的图像仍然很清晰，编码时候丢失的关键信息越多，图像失帧越严重。也就是说，压缩编码的强度越大，剔除的冗余信息就越多，得到的编码文件就越小。那么在解码的时候，图像失帧越严重。

所以在压缩的时候既要考虑图像的质量，也要考虑压缩比。压缩比和图像质量是一个互斥的过程。



视频编码就是为了减少视频中的冗余信息的。

视频和图像之所以可以编码，是因为它有冗余信息。

冗余可以分为：像素空间冗余、时间冗余（帧间冗余）、视觉冗余、编码冗余和心理冗余等。

我们主要看像素空间冗余和时间冗余（帧间冗余和帧内冗余）。

/\*

像素空间冗余（静态的图像中）：对于一帧图像（一个静态的图像），例如：如下的一张证件照，其后面背景中 相近的像素点之间的 像素差 是很小的。其亮度和色度也是比较平稳的。也就是说后面背景的像素值基本是一样的。

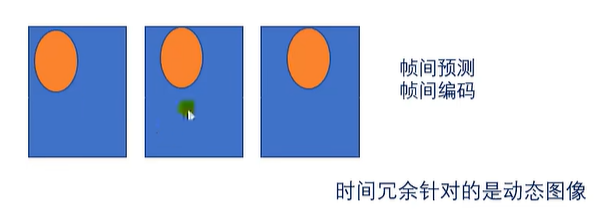


那么我们在预测这张图像的时候，可以用一些相邻的像素值来推断相邻的像素值。也就是说我们可以用特定大小的一个像素矩阵来表示相邻的像素。例如：假如图中左上角的黄色、黑色、绿色区域是已经编码好的像素块。我们可以利用这几个相邻的像素块作为参考来预测当前红色像素块。这种就称为帧内编码（帧内预测）。帧内预测是在一帧图像内进行的，帧内预测可以对一部分区域的相邻像素进行编码，然后利用已经编码的这部分像素的值来预测待编码区域的值。这样就减少了像素间空间冗余的目的，实现压缩。一张图像内有很多冗余的区域，如果在压缩图像的时候，把静态图像中相似像素点的冗余数据剔除了（不记录这部分数据），我们只记录具有代表性的数据，这个压缩后的数据量就大大减少了，在解压的时候我们就可以根据这些已经记录了的具有代表性的像素值来推断像素。

\*/

/\*

帧间冗余（时间冗余）：一般用于动态的图像中，连续帧的像素之间，具有相关性，就造成了冗余，因为在许多时候，在连续帧之间，大部分图片之间对应点的像素值是缓慢平稳过度的。

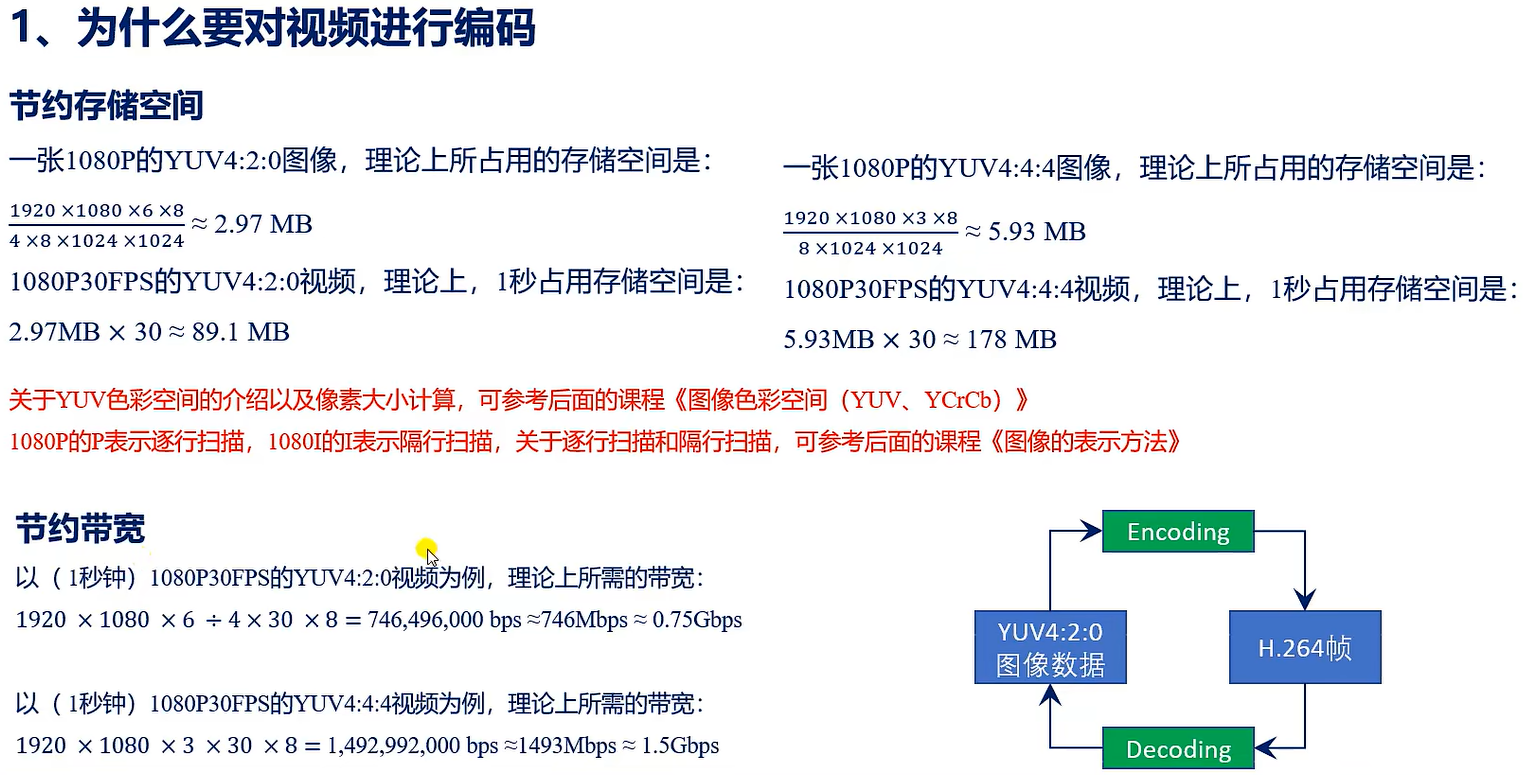


比如上面一段视频的相邻三帧图像，红色的部分在缓慢地向右运动，而剩余的相邻区域的像素都是接近的。比如下面的区域都是蓝色的。

连续的许多帧图像，相邻帧之间对应的一些区域像素点的值变化不大（缓慢过度的），我们就可以用其中某一帧为基础，来表示其他的帧信息。从而减少带宽的消耗。进而减小帧间冗余的目的。

也就是说视频序列之间也是有许多冗余的区域的，如时间上的冗余。

\*/



之所以要对视频进行压缩编码，是因为节省存储空间 以及 节省带宽。

首先，视频信号包含的信息量是很大的，如上图，一张1080P的YUV4:2:0的图像为例，宽1920，高1080。这张图像所占用的存储空间约为2.97MB

/\*公式由来：

1920\*1080像素点

每个像素点中Y取4个，U，V取一个。4+1+1=6所以 1920\*1080\*6

因为每4个像素点进行采样，所以除以4.就得到平均每个像素点占用的大小。

分子分母都乘以8是因为每个像素点是8位。转化为MB就又需要/1024/1024.

\*/

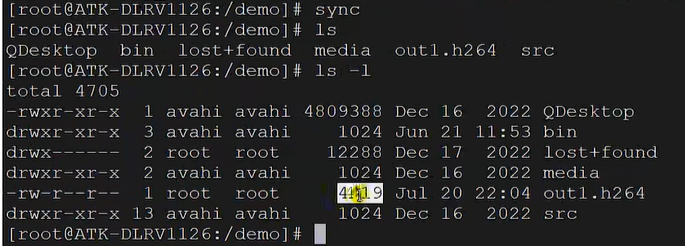
那么假设一段视频的帧率是30fps，那么一段1080P 30FPS YUV420的视频，1秒钟视频占用的存储空间就约为89.1MB。所以视频会占用极大空间。

对带宽需求也极大，见图中公式。

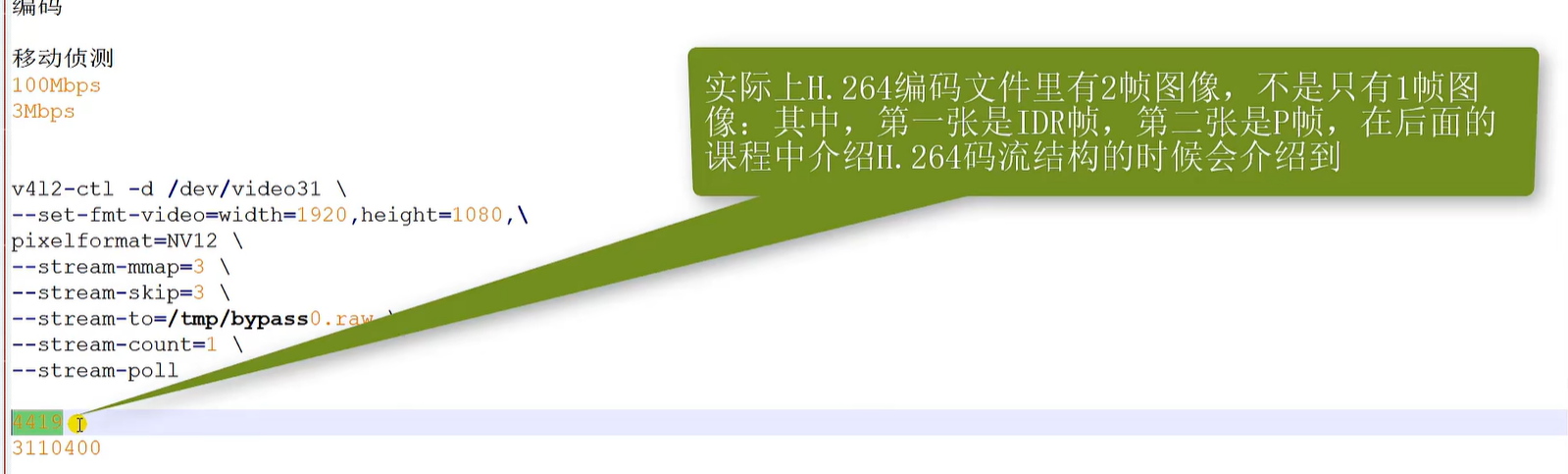
/\*抓取一帧图像：编码和没编码的区别：

H.264的大小：

IMG_256



使用v4l2命令抓取ISPP的图像，对应的大小：



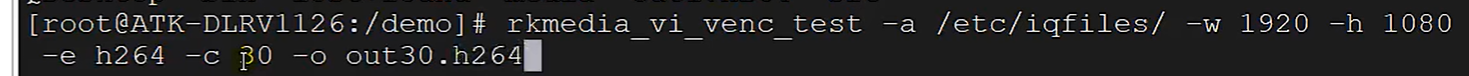
可见压缩后和没压缩的图像，二者相差极大。

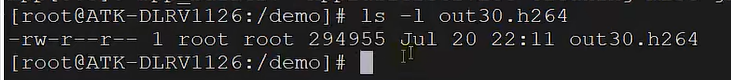
\*/

/\*

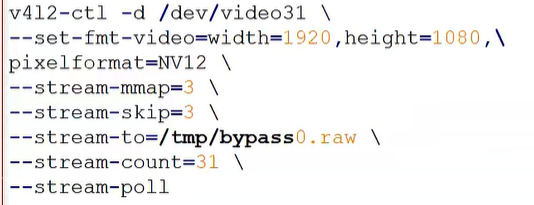
抓取31帧的图像，编码和没编码的区别：

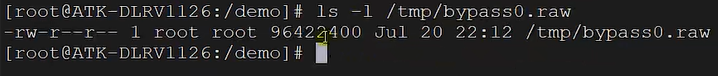
H264编码：





V4l2:





\*/

\*/