# 视频花屏和卡顿原因（音视频基础）

## **前言：**

大家晚上好，今天给大家分享一些关于音视频里面一些基础的知识点，基础知识点非常重要！

## **一、I、B、P帧：**

视频的播放过程可以简单理解为一帧一帧的画面按照时间顺序呈现出来的过程，就像在一个本子的每一页画上画，然后快速翻动的感觉。

但是在实际应用中，并不是每一帧都是完整的画面，因为如果每一帧画面都是完整的图片，那么一个视频的体积就会很大，这样对于网络传输或者视频数据存储来说成本太高，所以通常会对视频流中的一部分画面进行压缩（编码）处理。由于压缩处理的方式不同，视频中的画面帧就分为了不同的类别，其中包括：I 帧、P 帧、B 帧。

下面是关于I 帧、P 帧、B 帧的区别：

I 帧（Intra coded frames）：I 帧图像采用帧内编码方式，即只利用了单帧图像内的空间相关性，而没有利用时间相关性。I 帧使用帧内压缩，不使用运动补偿，由于 I 帧不依赖其它帧，所以是随机存取的入点，同时是解码的基准帧。I 帧主要用于接收机的初始化和信道的获取，以及节目的切换和插入，I 帧图像的压缩倍数相对较低。I 帧图像是周期性出现在图像序列中的，出现频率可由编码器选择。

P 帧（Predicted frames）：P 帧和 B 帧图像采用帧间编码方式，即同时利用了空间和时间上的相关性。P 帧图像只采用前向时间预测，可以提高压缩效率和图像质量。P 帧图像中可以包含帧内编码的部分，即 P 帧中的每一个宏块可以是前向预测，也可以是帧内编码。

B 帧（Bi-directional predicted frames）：B 帧图像采用双向时间预测，可以大大提高压缩倍数。值得注意的是，由于 B 帧图像采用了未来帧作为参考，因此 MPEG-2 编码码流中图像帧的传输顺序和显示顺序是不同的。

也就是说，一个 I 帧可以不依赖其他帧就解码出一幅完整的图像，而 P 帧、B 帧不行。P 帧需要依赖视频流中排在它前面的帧才能解码出图像。B 帧则需要依赖视频流中排在它前面或后面的帧才能解码出图像。

这就带来一个问题：在视频流中，先到来的 B 帧无法立即解码，需要等待它依赖的后面的 I、P 帧先解码完成，这样一来播放时间与解码时间不一致了，顺序打乱了，那这些帧该如何播放呢？这时就需要我们来了解另外两个概念：DTS 和 PTS(详见下边说明)。

两个I frame之间形成一个GOP，在x264中同时可以通过参数来设定bf的大小，即：I 和p或者两个P之间B的数量。

通过上述基本可以说明如果有B frame 存在的情况下一个GOP的最后一个frame一定是P.

看x264代码，感觉GOP 就是IDR帧到另一个IDR帧之间 就是一个GOP.在视频编码序列中，GOP即Group of picture（图像组），指两个I帧之间的距离，Reference（参考周期）指两个P帧之间的距离（）。一个I帧所占用的字节数大于一个P帧，一个P帧所占用的字节数大于一个B帧（）。

所以在码率不变的前提下，GOP值越大，P、B帧的数量会越多，平均每个I、P、B帧所占用的字节数就越多，也就更容易获取较好的图像质量；Reference越大，B帧的数量越多，同理也更容易获得较好的图像质量。

需要说明的是，通过提高GOP值来提高图像质量是有限度的，在遇到场景切换的情况时，H.264编码器会自动强制插入一个I帧，此时实际的GOP值被缩短了。另一方面，在一个GOP中，P、B帧是由I帧预测得到的，当I帧的图像质量比较差时，会影响到一个GOP中后续P、B帧的图像质量，直到下一个GOP开始才有可能得以恢复，所以GOP值也不宜设置过大。

同时，由于P、B帧的复杂度大于I帧，所以过多的P、B帧会影响编码效率，使编码效率降低。另外，过长的GOP还会影响Seek操作的响应速度，由于P、B帧是由前面的I或P帧预测得到的，所以Seek操作需要直接定位，解码某一个P或B帧时，需要先解码得到本GOP内的I帧及之前的N个预测帧才可以，GOP值越长，需要解码的预测帧就越多，seek响应的时间也越长。

## **二、DTS、PTS 的概念：**

DTS（Decoding Time Stamp）：即解码时间戳，这个时间戳的意义在于告诉播放器该在什么时候解码这一帧的数据。

PTS（Presentation Time Stamp）：即显示时间戳，这个时间戳用来告诉播放器该在什么时候显示这一帧的数据。

需要注意的是：虽然 DTS、PTS 是用于指导播放端的行为，但它们是在编码的时候由编码器生成的。

当视频流中没有 B 帧时，通常 DTS 和 PTS 的顺序是一致的。但如果有 B 帧时，就回到了我们前面说的问题：解码顺序和播放顺序不一致了。

比如一个视频中，帧的显示顺序是：I B B P，现在我们需要在解码 B 帧时知道 P 帧中信息，因此这几帧在视频流中的顺序可能是：I P B B，这时候就体现出每帧都有 DTS 和 PTS 的作用了。DTS 告诉我们该按什么顺序解码这几帧图像，PTS 告诉我们该按什么顺序显示这几帧图像。顺序大概如下：

PTS:   480   640  560  520  600  800  720  680  760  960 ...  
DTS:   400   440  480  520  560  600  640  680  720  760 ...  
Stream: I     P    B    B    B    P    B    B    B    P  ...  
播放序：1     5    3    2    4    9    7    6    8   10  ...                 
PTS >= DTS

## **三、音视频的同步：**

上面说了视频帧、DTS、PTS 相关的概念。我们都知道在一个媒体流中，除了视频以外，通常还包括音频。音频的播放，也有 DTS、PTS 的概念，但是 音频没有类似视频中 B 帧，不需要双向预测，所以音频帧的 DTS、PTS 顺序是一致的。

音频视频混合在一起播放，就呈现了我们常常看到的广义的视频。在音视频一起播放的时候，我们通常需要面临一个问题：怎么去同步它们，以免出现画不对声的情况。

要实现音视频同步，通常需要选择一个参考时钟，参考时钟上的时间是线性递增的，编码音视频流时依据参考时钟上的时间给每帧数据打上时间戳。在播放时，读取数据帧上的时间戳，同时参考当前参考时钟上的时间来安排播放。这里的说的时间戳就是我们前面说的 PTS。实践中，我们可以选择：同步视频到音频、同步音频到视频、同步音频和视频到外部时钟。

## **四、PTS和DTS的时间基：**

首先我们思考一下：PST和DTS的单位是什么?

为了回答这个问题，先引入FFmpeg中 时间基 的概念，也就是time\_base。它也是用来度量时间的。

如果把1秒分为25等份，你可以理解就是一把尺，那么每一格表示的就是1/25秒。此时的time\_base={1，25} 如果你是把1秒分成90000份，每一个刻度就是1/90000秒，此时的time\_base={1，90000}。

所谓时间基 表示的就是 每个刻度是多少秒 。

pts的值就是占多少个时间刻度（占多少个格子）。它的单位不是秒，而是时间刻度。只有pts加上time\_base两者同时在一起，才能表达出时间是多少。

好比我只告诉你，某物体的长度占某一把尺上的20个刻度。但是我不告诉你，这把尺总共是多少厘米的，你就没办法计算每个刻度是多少厘米，你也就无法知道物体的长度。

pts=20个刻度

time\_base={1,10} 每一个刻度是1/10厘米

所以物体的长度=pts*time\_base=20*1/10 厘米

在ffmpeg中。av\_q2d(time\_base)=每个刻度是多少秒

此时你应该不难理解 pts\*av\_q2d(time\_base)才是帧的显示时间戳。

下面理解时间基的转换，为什么要有时间基转换。

首先，不同的封装格式，timebase是不一样的。另外，整个转码过程，不同的数据状态对应的时间基也不一致。拿mpegts封装格式25fps来说（只说视频，音频大致一样，但也略有不同）。非压缩时候的数据（即YUV或者其它），在ffmpeg中对应的结构体为AVFrame,它的时间基为AVCodecContext 的time\_base ,AVRational{1,25}。

压缩后的数据（对应的结构体为AVPacket）对应的时间基为AVStream的time\_base，AVRational{1,90000}。

因为数据状态不同，时间基不一样，所以我们必须转换，在1/25时间刻度下占10格，在1/90000下是占多少格。这就是pts的转换。

根据pts来计算一桢在整个视频中的时间位置：

timestamp(秒) = pts \* av\_q2d(st->time\_base)

duration和pts单位一样，duration表示当前帧的持续时间占多少格。或者理解是两帧的间隔时间是占多少格。一定要理解单位。

pts：格子数

av\_q2d(st->time\_base): 秒/格

* 计算视频长度：

time(秒) = st->duration \* av\_q2d(st->time\_base)

ffmpeg内部的时间与标准的时间转换方法：ffmpeg内部的时间戳 = AV\_TIME\_BASE \* time(秒) AV\_TIME\_BASE\_Q=1/AV\_TIME\_BASE

av\_rescale\_q(int64\_t a, AVRational bq, AVRational cq)函数

这个函数的作用是计算a\*bq / cq来把时间戳从一个时间基调整到另外一个时间基。在进行时间基转换的时候，应该首先这个函数，因为它可以避免溢出的情况发生。函数表示在bq下的占a个格子，在cq下是多少。

* 关于音频pts的计算：

音频sample\_rate:samples per second，即采样率，表示每秒采集多少采样点。

比如44100HZ，就是一秒采集44100个sample. 即每个sample的时间是1/44100秒

一个音频帧的AVFrame有nb\_samples个sample，所以一个AVFrame耗时是nb\_samples\*（1/44100）秒 即标准时间下duration\_s=nb\_samples\*（1/44100）秒，

转换成AVStream时间基下 duration=duration\_s / av\_q2d(st->time\_base)

基于st->time\_base的num值一般等于采样率,所以duration=nb\_samples. pts=n*duration=n*nb\_samples

## **视频花屏和卡顿原因：**

这个原因只能作为参考，具体实际应用开发遇到的问题，视情况而定：

视频花屏原因：如果GOP分组中有B帧、P帧丢失，会造成解码端的图像发生错误，这会出现马赛克（也就是花屏的现象出现！）

视频卡顿的原因：为了避免花屏的问题发生，当出现有帧丢失时，就丢弃GOP内所有的帧，直到下一个IDR正重新刷新图像；I帧是按照帧周期来的，需要一个比较长的时间周期，如果在下一个I帧来之前，不显示后来的图像，那么视频就静止不动了，这就是出现了所谓的卡顿现象

补充：

什么是GOP：（group of picture）,它是一组帧，也就是一组图片。

IDR帧：(Instantanous Decoder Resfresh)解码器立即刷新，每当遇到IDR帧时，解码器就会清空解码器参考buffer中的内容。每个GOP中的第一个帧就是IDR帧，它是一种特殊的I帧，他在解码过程中，防止解码错误传播

## **五、总结：**

文章参考来源：

<https://blog.csdn.net/knowledgebao/article/details/84326649>