001\_视频基础概念

编辑历史

|  |  |  |
| --- | --- | --- |
| 更新日期 | 作者 | 更新内容 |
| 2024.11.12 | 胡益华 | Initialization |
|  |  |  |
|  |  |  |

目录

[001\_视频基础概念 I](#_Toc16647)

[编辑历史 I](#_Toc23667)

[目录 II](#_Toc392)

[1. 引言 1](#_Toc31916)

[1.1 概述 1](#_Toc6854)

[1.2 基础概念 1](#_Toc26074)

[2. 比特率 2](#_Toc22693)

[2.1 概念 2](#_Toc30625)

[2.2 比特率的应用 2](#_Toc6723)

[2.2.1 h264 比特率 3](#_Toc17034)

[2.2.2 hevc 比特率 3](#_Toc11362)

[2.2.3 av1 比特率 3](#_Toc19775)

[2.2.4 编码比特率配置 3](#_Toc31790)

[3. 关键帧 5](#_Toc29887)

[3.1 概念 5](#_Toc3360)

[3.2 使用特点 6](#_Toc18226)

**1.** 引言

**1.1** 概述

视频的压缩算法十分复杂，不过现在，各种开源库的完善使处理视频变得简单起来。最著名的如 ffmpeg，以及被 ffmpeg 引用的第三方组件，如 libx264、libx265 等。

虽然 api 的调用使得处理视频难度骤降，不过了解一些视频的基础概念，或者编解码器的基础工作原理还是非常有必要的，否则在使用相关 api 时，对很多配置参数都会感到陌生。

**1.2** 基础概念

实际上，我对很多音视频相关的概念也只是听说，并没有真正地了解这些概念。这些概念在通信领域会被广泛提及，在编写代码时，也许不用了解也不影响结果。所以这应征那句话：有兴趣可自行了解。

这篇文档会记录一些我在使用 ffmpeg 库编码时遇到的配置参数，如比特率，关键帧等等。

**2.** 比特率

**2.1** 概念

使用 ffmpeg 时，它会要求使用者填写一个参数，名为 `bit\_rate`，即比特率，也称为码率。通常对其的定义为：

单位时间内传送或处理的比特数量。

这句话的含义也体现在比特率的单位上：bps，即 bit per second 的缩写。

不过每当我看到这行定义，都会感到莫名其妙，我认为这个定义讲的不算好。传输或处理的比特数量，所以这个视频到底在传输或处理什么？我语文向来不是很好，实在不能理解。但是基本上所有地方，甚至询问 gpt，它们都会用这句话回答我，比特率是这个意思。

把比特率的概念先抛到一边。找到一些视频文件，打开属性可以看到视频的一些信息。视频的大小一般可以根据以下方式计算：

video\_size(KB) ≈ (bit\_rate(bps) / 8 / 1024) \* video\_duration(s)

主流编码格式的视频都可以根据这个估算大小，显然，比特率是一个平均数的概念。举例来说，对于一个比特率为400000 bps，30帧的视频，每一帧分配的比特数就是：400000 / 30 (bits) = 13333 (bits)。所以对于视频而言，我认为这才是比特率的核心意义，即体现视频编码后，平均每一帧分配到的比特数大小。此时再回过头来看比特率的定义，也就不难理解了。

比特率只是一个平均值，实际编码过程比较复杂，聪明的编码器会基于某个设定的比特率数值进行编码，一般情况如下：

(1) 对于相对静态的画面，编码器会分配比较少的比特数；

(2) 对于相对动态的画面，如场景切换或快速运动的画面，编码器会分配比较多的比特数，从而保证这些画面的质量。

实际，还有如可变码率(VBR)、恒定码率(CBR)、恒定平均码率(AVR)等概念，不过只要理解了比特率，这些都不难理解。

**2.2** 比特率的应用

一定范围内，比特率越大，视频质量越好。通过网络传输视频数据时，基于网络带宽，可以选择合适的比特率。若网络状况不佳，又选择了比较高的比特率，尤其是比如 VBR 模式下，可能会遇到视频的高比特区段，这可能导致“拥塞”问题，导致延迟、卡顿、丢帧等。尽管一些缓存应对机制应运而生，不过尽量还是选择合适的比特率。

不同的编码格式对于比特率的要求相差很大。

**2.2.1 h264** 比特率

h264 的压缩率不如 hevc，所以比特率相对较大。参考如下，未必准确：

|  |  |  |
| --- | --- | --- |
| **Resolution(pixel \* pixel)** | **Fps (fps)** | **Suggested Bitrate (Mbps)** |
| 640 \* 480 | 30 fps | 0.5 ~ 2.0 |
| 1280 \* 720 | 30 fps | 1.5 ~ 5.0 |
| 1920 \* 1080 | 30 fps | 3.0 ~ 12.0 |

**2.2.2 hevc** 比特率

对于 hevc，我没有获取到太多的相关资料，参考如下，未必准确：

|  |  |  |
| --- | --- | --- |
| **Resolution(pixel \* pixel)** | **Fps (fps)** | **Suggested Bitrate (Mbps)** |
| 640 \* 480 | 30 fps | 0.35 ~ 0.85 |
| 1280 \* 720 | 30 fps | 0.7 ~ 1.5 |
| 1920 \* 1080 | 30 fps | 1.2 ~ 3.5 |
| 2560 \* 1440 | 30 fps | 2.5 ~ 8.0 |
| 3840 \* 2160 | 30 fps | 6.0 ~ 22.0 |

**2.2.3 av1** 比特率

对于 av1，我查到的资料少之又少，这里我基于 hevc 的推荐码率，用 ffmpeg 自己编码了一些视频，观察视频的质量。一般公认地，av1 的压缩率比 hevc 更加优秀一些，所以比特率应该也会小一些。因为是我自己测试的，各种配置可定没有控制变量，如 gop size，B 帧等参数。所以这里仅作记录，不能作为参考：

|  |  |  |
| --- | --- | --- |
| **Resolution(pixel \* pixel)** | **Fps (fps)** | **Suggested Bitrate (Mbps)** |
| 640 \* 480 | 30 fps | 0.35 ~ 0.8 |
| 1280 \* 720 | 30 fps | 0.75 ~ 1.4 |
| 1920 \* 1080 | 30 fps | 1.0 ~ 3.0 |
| 2560 \* 1440 | 30 fps | 2.0 ~ 6.0 |
| 3840 \* 2160 | 30 fps | 4.0 ~ 22.0 (upper limit untested) |

**2.2.4** 编码比特率配置

编码时，比特率一般仅作为参考值，编码器会基于这个配置去编码，但是结果未必是配置值。

并且不同的编码器对同种类型的视频编码格式，比特率的配置策略差异也很大。比如我发现通过 ffmpeg 调用第三方组件 lib265x、svt-av1 编码出的结果文件，比特率和配置值就比较吻合。而通过 lib264x 编码，结果文件的比特率一般都被调整的比配置大好多。关于这个我用 gdb 粗略看了一些，不是 ffmpeg 内部调整的比特率，应该是 lib264x 库内部做了一些调整，可能通过某些额外配置可以解决这个问题，我没怎么深入探究。

**3.** 关键帧

**3.1** 概念

大多数视频编码器都有关键帧这个概念，这是压缩时间数据冗余很好的一种方式。了解一些视频基本知识的都听过以下三种类型的帧(引用 chatgpt 的回答)：

**3.1.1 I 帧**

Intra-coded Frame

定义：

I 帧是关键帧，也称为自包含帧。采用帧内压缩技术。它包含完整的图像信息，类似于静态图片。这里打断一下 gpt 的回答，在处理视频时，经常能看到 gop size 这个配置，指的就是两个 I 帧之间的帧数间隔。

特点：

(1) 不依赖其他帧，解码时不需要参考其他帧的内容。

(2) 压缩率最低，但包含最多的信息，通常作为视频的起始帧。

(3) 在发生快进、倒退等操作时，I 帧有助于视频的快速定位。

(4) I 帧的数量越多，视频的压缩率会降低，但对随机访问的支持会更好。

**3.1.2 P 帧**

Predictive-coded Frame

定义：

P 帧是预测帧。采用帧间压缩技术。通过参考前面的 I 帧或 P 帧进行编码，只存储与参考帧的差异。

特点：

(1) 依赖于之前的 I 帧或 P 帧，不包含完整的图像信息，解码时需要参考之前的帧。

(2) 相较于 I 帧，P 帧的压缩率更高，因只需记录变化部分。

(3) 使用运动补偿等技术来描述帧间变化，从而减少冗余。

**3.1.3 B 帧**

Bidirectionally Predictive-coded Frame

定义：

B 帧是双向预测帧。采用帧间压缩技术。通过参考前后两帧(I 帧或 P 帧)进行编码。

特点：

(1) 同时参考前后的帧，因此压缩率最高。

(2) 因依赖两侧帧，在解码时通常需要先解码相邻的参考帧。

(3) B 帧的加入可以显著提高压缩效率，但增加了解码的复杂度和存储需求。

**3.2** 使用特点

压缩率：B 帧 > P 帧 > I 帧

编码复杂度：B 帧 > P 帧 > I 帧

随机访问性能：I 帧 > P 帧 > B 帧