CSDN 博客 下载 学习 社区 GitCode InsCode GPU ffmpeg 搜索 会员中心 消息 历史 创作中心 AI发文

原创 Mango酱 于 2020-12-04 14:31:01 发布 969 收藏 9 版权
分类专栏：FFMPEG 文章标签：ffmpeg 视频处理

FFMPEG 专栏收录该内容 9 订阅 27 篇文章 订阅专栏

12关于FFmpeg的四种时基和时基相关函数的分析及其场景用法

前言：
AVRational类型：是一个分数。例如{1,25}表示1除以25即1/25。
时基：时间的单位，在ffmpeg下被描述成时基。
时间戳：某个时刻的时间。

1 FFmpeg的四种时基
抛开解码器上下文的时基即stream->codec->time_base(因为编码时基本不用，所以我这里不讲它，它也叫tbc,称之为流中的时基)。我们FFmpeg实际上编码经常使用的时基有四种：

- 1）输入流封装上下文的时基，也叫容器的时基，即AVFormatCtx->stream->time_base。
- 2）输出流封装上下文的时基，也叫容器的时基，即AVFormatCtx->stream->time_base。(注意：编码时输入输出各有一个封装上下文，当然也有可能没有输入流的封装上下文，但是这里为了分析四种时基特意不考虑)
- 3）我们人类实际的时基，也就是以秒为单位。
- 4）FFmpeg的内部时基AV_TIME_BASE，百万,故单位为微妙。

解释：
1）输入流封装上下文的时基
例如flv为1000，ts的输入流为90k。

2）输出流封装上下文的时基
这个是我们想要输出时的容器格式的时基，同上面输入一样，如果你想输出flv格式的，那么时基为1000，想输出ts格式的那么时基为90k。

3）人类实际的时基
没什么好讲的，就是转来给人类看的。

4）内部时基AV_TIME_BASE
定义如下：

```
#define          AV_TIME_BASE            1000000 //百万,故单位为微妙
#define          AV_TIME_BASE_Q          (AVRational){1, AV_TIME_BASE} //上面倒数
```

我这里为什么将FFmpeg的时基列为四种呢?
主要是我们在编码的时候使用的时基只有下面四种。
何为转时基：转时基就是将输入容器中以输入容器时基为单位的时间戳，转换成输出容器中的以输出容器时基为单位的时间戳。
例如输入容器的时间戳为80,时基为1000，想要输出成ts格式的流，那么由于输入输出帧率是要一样的，所以1000/80=90000/x得出x=7200(实际上我下一篇关于推流时pts的运算步骤也讲到)。
有人会想不懂为什么时基/时间戳=帧率，例如1s下帧率为25那么每帧时间戳为0.04，那么时基1除以时间戳0.04=帧率25。
所以这里我们编码时已经用到了两种时基。

那么还有两种为什么要包含进来呢?
因为另外两种时基在转时基时非常常用，例如人类的实际时间经常依靠帧率运算出每帧显示的时长，而FFmpeg的内部时基被用于转换人类实际的时长，当然你也可以不转，但是转了能更精确表示，但需要舍弃小数，难度相对大点。

例如ffmpeg内部时基与人类标准的时间转换方法：

```
timestamp(ffmpeg内部的时间戳) = AV_TIME_BASE * time(秒)//乘以一百万即获取到微秒单位的内部时间戳，人为认为它是微秒单位
time(秒) = AV_TIME_BASE_Q * timestamp(ffmpeg内部的时间戳)//上面公式转换得出。即对应除以一百万即可得出原来的标准秒数。
```

2 与时基转换相关的函数
1）av_q2d函数。

```
/*
 * 作用：下面看到，该函数非常简单，只是将传入的AVRational类型转为double类型返回。
*/
static inline double av_q2d(AVRational a){
    return a.num / (double) a.den;
}
```

但是该函数的用于主要是将某个时基转成doule(可以有小数保留，更加精确)，以作计算。
例如将标准时间戳转为内部时间的做法是标准时间乘以对应时基，那么下面利用该函数同样也可以做到。

```
packet播放时刻值：timestamp(单位秒) = packet.pts × av_q2d(stream.time_base);//除法通常是变大单位，例如stream.time_bas
packet播放时长值：duration(单位秒) = packet.duration × av_q2d(stream.time_base);

//实际上上面内部时间戳与标准时间戳的转换也可以写成(并且通常这样做)：
timestamp(ffmpeg内部的时间戳) = av_q2d({AV_TIME_BASE,1}) * time(秒) //乘以一百万
time(秒) = av_q2d(AV_TIME_BASE_Q) * timestamp(ffmpeg内部的时间戳) //除以一百万
```

该函数可以使平常的秒按照自己的算法转换成一定的时间戳表示。
注意下面，不能简单的认为是秒转成time_base然后再转成微秒AV_TIME_BASE，毕竟若是秒转成time_base按照上面内部时基的转法，需要在此之前乘以time_base，这里是直接转了，所以我们这里可以认为是一种自定义运算方法。实际上这里也是雷神博主的做法：

```
int64_t pts = sec / av_q2d(ifmt_ctx->streams[videoindex]->time_base)*AV_TIME_BASE;
```

2）av_rescale_q函数
作用是将时间戳从一个时基转到另一个时基。常用于将输入容器的时基转成输出容器的时基。例如：
注意：in_stream->time_base，out_stream->time_base在我这里指的是容器的时基(但是很多人也叫视频流的时基,所以很容易歧义)，stream->codec->time_base指的是编解码器上下文的时基。

```
pkt.pts = av_rescale_q(pkt.pts, in_stream->time_base, out_stream->time_base);
pkt.dts = av_rescale_q(pkt.dts, in_stream->time_base, out_stream->time_base);
pkt.duration = av_rescale_q(pkt.duration, in_stream->time_base, out_stream->time_base);
```

3）av_rescale_q_rnd函数
这个函数实际上和上面的av_rescale_q函数一样，但是多了一个参数4可以四舍五入小数，对于FFmpeg的pts需要比较精确，这个函数确实是不错的选择，但是要求比较高，该参数设置需要按照不同视频设置去设置，否则容易报错。
例如：

```
pkt.pts = av_rescale_q_rnd(pkt.pts, in_stream->time_base, out_stream->time_base, (AVRounding)(AV_ROUND_NEAR_INF
pkt.dts = av_rescale_q_rnd(pkt.dts, in_stream->time_base, out_stream->time_base, (AVRounding)(AV_ROUND_NEAR_INF
```

好了，关于时基和其相关函数的讲述差不多了，下一篇我们将讲到使用FFmpeg编码推流时，pts的计算步骤，和运用这些时基及其相关函数理解，因为pts的计算实际上并无固定的算法，你的视频流畅你就是牛逼。
pts的计算有两种可能，一是没有输入流的封装上下文的计算(即没有相关属性，需要自己初始化一些内容)，只有输出流的封装上下文；二是有输入流和输出流的封装上下文的计算。但两种的计算步骤是差不多的，只不过内部处理方式有点区别。

分类专栏 发布首篇原创文章，原力分+10，点亮新秀勋章
C++数据结构 30篇
C/C++功能函数 31篇
python学习 11篇
#模板学习 8篇
面试相关 8篇
C++新特性相关 12篇
C++11多线程编程 13篇
音视频流媒体开发 18篇
ONVIF开发 8篇
软件库下载与安装 18篇
Linux网络编程 28篇
Windows编程 1篇
GDB调试，相关命令及环... 21篇
FFMPEG 27篇
代码管理工具及相关命令 13篇
C++STL专栏 9篇
Go 32篇
JavaScript学习笔记 121篇
docker 7篇
WebRTC
redis专题 11篇
MYSQL 36篇
Qt遇到的BUG 6篇
C++设计模式 4篇
黑马Qt笔记 61篇
Linux系统编程 83篇
Tcpdump与Wireshark抓包 2篇
Nginx源码分析 9篇
MFC教程 30篇
rapidJson库 5篇
Linux_C++_Server 1篇

利用ffmpeg获取视频封面图，时长信息 05-23
利用ffmpeg获取视频封面图，时长信息

ffmpeg编码的代码 08-20
网上买的关于ffmpeg的编码部分

(音视频学习笔记):FFmpeg库简介、常用函数及数据结构解析_ffmpeg库函数... 8-27
FFmpeg 3.x 组件注册方式 使用ffmpeg,首先要执行av_register_all,把全局的解码器、编码器等结构体注册到各自全局的对象链表里,以便后面查找调用 FF...

ffmpeg 时基timebase、时间戳pts/dts、延时控制delay_aworkholic的博客... 8-29
例如, 在时基bq = {1,1200000}有时间戳 a1 = 48000,需要转换到时基 cq ={1,1000000}下的时间戳a2, 计算过程应该为 a2 = 48000 *(1/1200000)/ (1/1000...

ffmpeg demo_ffmpegh264编码_ffmpeg_ 10-02
ffmefg实现的H264编码解码demo

iOS FFmpeg软编码demo 12-18
iOS ffmpeg软编码demo，能够正常运行，实现效果。值得学习。

FFmpeg解码详解(原理与函数使用说明)_ffmpeg 解码原理_OH,CGWLMXUP... 8-25
FFmpeg解码函数简介 ◦ av_register_all():注册所有组件。 ◦ avformat_open_input():打开输入视频文件。 ◦ avformat_find_stream_info():获取视频文件信...

FFmpeg基础知识_codecpar_&小菜子的博客 9-6
FFmpeg最常用的结构体 解协议(http,rtsp,rtmp,mms) 协议(文件)操作的顶层结构是AVIOContext,这个对象实现了带缓冲的读写操作;FFmpeg的输入对象AV...

2022 ffmpeg 编码器exe版本 06-17
2022 ffmpeg 编码器exe版本

FFmpeg时间戳详解 weixin_34072857的博客 1139
本文为作者原创，转载请注明出处：https://www.cnblogs.com/leisure_chn/p/10584910.html 1. I帧/P帧/B帧 I帧：I帧(Intra-coded picture, 帧内编码帧，常...

ffmpeg常用结构体及函数_lu_xiukun的博客 8-22
ffmpeg常用结构体及函数 libavformat:用于各种音视频封装格式的生成和解析,包括获取解码所需信息以生成解码上下文结构和读取音视频帧等功能;音视频...

FFMpeg学习笔记_av_codec_iterate_asiwxy的博客 8-29
1.2,编解码相关的结构体: I、FFmpeg编解码相关数据结构简介 AVFormatContext AVInputFormat AVStream AVCodecContext AVCodec AVPacket AVFra...

ffmpeg中的时间单位以及时间转换函数(av_q2d av_rescale_q) Linux C/C++ 领域内容 1781
ffmpeg 时间单位

ffmpeg时间戳相关函数 Lce的专栏 785
ffmpeg中与时间戳相关的函数主要有：av_rescale_q，av_rescale_q_rnd，av_compare_ts。= 4);a0 += r;for(i = 63;i >= 0;a1 -= c;t1 ++;1= 4);a0 += r;for(...

FFmpeg常见数据结构_ffmpeg 数据结构_EricsZhou周靖的博客 8-31
这个结构体,是FFmpeg中有关io操作的顶层结构体,是avio的核心。FFmpeg支持打开本地文件路径和流媒体协议的URL。 虽然AVIOContext时avio操作的核...

ffmpeg学习 函数分析swr_convert_swr_convert例子_aworkholic的博客-CSD... 9-5
有关ffmpeg中主要的api函数源码解析参考雷神系列文章,整理如下 ffmpeg学习(2)获取和使用,源码分析。libswresample主要是用于音频的重采样和格式转...

ffmpeg av_rescale_q_rnd 6110
int64_t av_rescale_q_rnd(int64_t a, int64_t b, int64_t c, enum AVRounding rnd); 它的作用是计算 "a * b / c" 的值并分五种方式来取整. 用在FFmpeg中,...

av_rescale_q 产业互联网&消费类电子 3314
时间戳，DTS(decoding time stamp)，PTS(presention time stamp)，CTS（current time stamp）。 ffmepg中的时间戳，是以微秒为单位，关乎timebase...

ffmpeg-5.0.1 07-06
ffmpeg-5.0.1

Qt基于FFmpeg读取摄像头并进行H264编码 01-20
Qt5基于FFmpeg读取摄像头生成yuv和rgb数据两份数据，利用解码线程类实现边解码生成rgb数据边在窗口类播放，利用生成的yuv数据进行编码生成h26...

PHP中使用FFMPEG获取视频缩略图和视频总时长实例 10-25
主要介绍了PHP中使用FFMPEG获取视频缩略图和视频总时长实例,需要的朋友可以参考下

使用ffmpeg.exe获取文件属性信息,视频音频时长 12-19
使用ffmpeg.exe获取文件属性信息，ffmpeg是java开发的用于多媒体文件编辑的命令行工具，有多个版本，功能比较强大，C#中可以在进程外异步调用这...

ffmpeg.exe和jave-2.0.jar（获取视频时长） 03-12
Java获取视频文件时长，为后面根据视频时长随机获取缩略图做铺垫

ffmpeg框架函数分析 07-09
ffmpeg调用编解码器的流程函数，详细介绍ffmpeg中解码部分涉及函数，编码部分涉及的函数。

FFmpeg学习总结 weixin_43155199的博客 954
ffmpeg学习总结

FFMPEG之时间基 cchao985771161的博客 366
fps=29.97 fps=29.97这是一个小数啊，我如果直接利用公式 frame number = time * fps 得到了也不是一个整数啊，而帧号应该是一个整数才对。 首先，2...

ffmpeg音视频同步 不断debug，优化中…… 695
编码器到容器需要做时间转换，视频H264时间基{1，30}，转换成flv{1，1000}，音频的时间基{1,44100}转换成flv{1，1000}，使用av_packet_rescale_ts(...

ffmpeg http 最新发布 07-28
根据引用

1

，您提到您使用的是amlogic平台android6，其中amlogic平台播放器amplayer使用ffmpeg作为解析器。您最近遇到了播放http ts流时的卡顿问题，可能是...
于缓冲区处于缓冲状态导致的。然而，度娘上关于这个问题的资料并不多。 根据引用

2

，URLContext是一个结构体，其中包含了一些关于URL的信息，比如AVClass、URLProtocol、flags等等。这些信息在url_open()函数中设置。 根据引用

3

，在解析http ts流的场景中，相关的结构体主要在libavformat/http.c、libavformat/mpegts.c、libavformat/avio.c和avio.h中。这些结构体和文件可能与您遇...

“相关推荐”对你有帮助么？

非常没帮助 没帮助 一般 有帮助 非常有帮助

关于我们 招贤纳士 商务合作 寻求报道 400-660-0108 kefu@csdn.net 在线客服 工作时间 8:30-22:00

Mango酱 关注 2 9 0 专栏目录

Mango酱
码龄5年 暂无认证

670	17万+	8879	53万+	
原创	周排名	总排名	访问	等级

7920	211	321	175	1669
积分	粉丝	获赞	评论	收藏

私信 关注

12关于FFmpeg的四种时基和时基相关函数的分析及其场景用法 解决了你的问题么？
可以写篇文章记录加深印象哦~
写文章

搜博主文章

热门文章

0xC0000005: 写入位置 0x00000000 时发生访问冲突的解决办法(内存对齐) 19059

Linux下安装openssl 16811

04-----无法执行二进制文件: 可执行文件格式错误 16274

02-----音频通道数、采样频率、采样位数、采样个数(样本数)的概念及计算一帧音频的大小、每秒播放的音频字节大小、一帧的播放时长、音频重采样 9958

智能指针之unique_ptr(概述，初始化，成员函数用法，以及转化成shared_ptr)07 9909

最新评论

04-----关于Qt下编译大文件的源码时报to...
狮子取名字: 谢谢。我的是编译生成时，报的这错，看来不是同一个

04-----关于Qt下编译大文件的源码时报to...
Mango酱: qt的pro文件

MFC之列表控件24
姜姜不恰姜: m_list.SetBkColor(RGB(67, 67, 67)); // 设置背景颜色 m_list.SetTextBk...

04-----关于Qt下编译大文件的源码时报to...
狮子取名字: 我是cmake opencv时遇到的

04-----关于Qt下编译大文件的源码时报to...
狮子取名字: -Wa,-mbig-obj,请问这个在哪里添加呢

您愿意向朋友推荐“博客详情页”吗？

强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

python基础----11-----闭包、装饰器、单例、工厂、多线程、socket、正则表达式、递归

python基础----10-----python操作mysql

python基础----09-----类、对象、魔法方法、封装、继承、类型注解、多态、数据分析案例

2023年 11篇	2022年 54篇
2021年 356篇	2020年 256篇