

最简单的基于FFmpeg的视频编码器-更新版 (YUV编码为HEVC(H.265))

2014年10月04日 14:12:45 阅读量: 40504

最简单的基于FFmpeg的视频编码器文章列表：

[最简单的基于FFMPEG的视频编码器 \(YUV编码为H.264\)](#)

[最简单的基于FFmpeg的视频编码器-更新版 \(YUV编码为HEVC\(H.265\)\)](#)

[最简单的基于FFmpeg的编码器-纯净版 \(不包含libavformat\)](#)

前一阵子做过一个基于FFmpeg的视频编码器的例子：

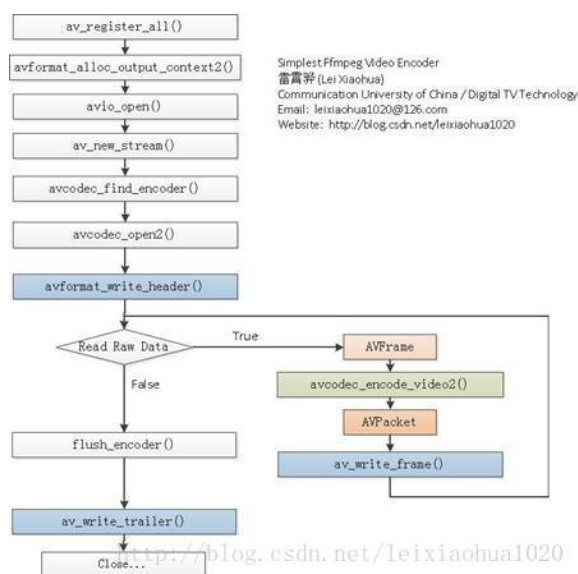
[最简单的基于FFMPEG的视频编码器 \(YUV编码为H.264\)](#)

在该例子中，可以将YUV像素数据 (YUV420P) 编码为H.264码流。因为如今FFmpeg已经实现了对libx265的支持，因此对上述编码H.264的例子进行了升级，使之变成编码H.265 (HEVC) 的例子。

比较早的FFmpeg的类库 (大约几个月以前的版本，我这里编译时间是2014.05.06) 对H.265的编码支持有问题。开始调试的时候，以为是自己的代码有问题，几经修改也没有找到解决方法。最终发现是类库本身的问题，更换新版本的类库 (我这里编译时间是2014.09.16) 后问题解决。

流程

下面附上一张FFmpeg编码视频的流程图。通过该流程，不仅可以编码H.264/H.265的码流，而且可以编码MPEG4/MPEG2/VP9/VP8等多种码流。实际上使用FFmpeg编码视频的方式都是一样的。图中蓝色背景的函数是实际输出数据的函数。浅绿色的函数是视频编码的函数。



简单介绍一下流程中各个函数的意义 (上一篇YUV编码为H.264的文章中已经写过一遍，这里复制粘贴一下)：

av_register_all()：注册FFmpeg所有编解码器。

avformat_alloc_output_context2()：初始化输出码流的AVFormatContext。

avio_open()：打开输出文件。

av_new_stream()：创建输出码流的AVStream。

avcodec_find_encoder()：查找编码器。

avcodec_open2()：打开编码器。

avformat_write_header()：写文件头 (对于某些没有文件头的封装格式，不需要此函数。比如说MPEG2TS)。

avcodec_encode_video2()：编码一帧视频。即将AVFrame (存储YUV像素数据) 编码为AVPacket (存储H.264等格式的码流数据)。



av_write_frame()：将编码后的视频码流写入文件。

flush_encoder()：输入的像素数据读取完成后调用此函数。用于输出编码器中剩余的AVPacket。

av_write_trailer()：写文件尾 (对于某些没有文件头的封装格式，不需要此函数。比如说MPEG2TS)。

代码

下面直接贴上代码

```
[cpp]  

1.  /**
2.   * 最简单的基于FFmpeg的视频编码器
3.   * Simplest FFmpeg Video Encoder
4.   *
5.   * 雷霄骅 Lei Xiaohua
6.   * leixiaohua1020@126.com
7.   * 中国传媒大学/数字电视技术
8.   * Communication University of China / Digital TV Technology
9.   * http://blog.csdn.net/leixiaohua1020
10.  *
11.  * 本程序实现了YUV像素数据编码为视频码流（HEVC(H.265), H264, MPEG2, VP8等等）。
12.  * 是最简单的FFmpeg视频编码方面的教程。
13.  * 通过学习本例子可以了解FFmpeg的编码流程。
14.  * This software encode YUV420P data to HEVC(H.265) bitstream (or
15.  * H.264, MPEG2, VP8 etc.).
16.  * It's the simplest video encoding software based on FFmpeg.
17.  * Suitable for beginner of FFmpeg
18.  */
19.
20. #include <stdio.h>
21.
22. extern "C"
23. {
24.     #include "libavutil\opt.h"
25.     #include "libavcodec\avcodec.h"
26.     #include "libavformat\avformat.h"
27.     #include "libswscale\swscale.h"
28. };
29.
30.
31. int flush_encoder(AVFormatContext *fmt_ctx,unsigned int stream_index)
32. {
33.     int ret;
34.     int got_frame;
35.     AVPacket enc_pkt;
36.     if (!(fmt_ctx->streams[stream_index]->codec->capabilities &
37.         CODEC_CAP_DELAY))
38.         return 0;
39.     while (1) {
40.         printf("Flushing stream #%u encoder\n", stream_index);
41.         //ret = encode_write_frame(NULL, stream_index, &got_frame);
42.         enc_pkt.data = NULL;
43.         enc_pkt.size = 0;
44.         av_init_packet(&enc_pkt);
45.         ret = avcodec_encode_video2 (fmt_ctx->streams[stream_index]->codec, &enc_pkt,
46.             NULL, &got_frame);
47.         av_frame_free(NULL);
48.         if (ret < 0)
49.             break;
50.         if (!got_frame){
51.             ret=0;
52.             break;
53.         }
54.         printf("Succeed to encode 1 frame! 编码成功1帧!\n");
55.         /* mux encoded frame */
56.         ret = av_write_frame(fmt_ctx, &enc_pkt);
57.         if (ret < 0)
58.             break;
59.     }
60.     return ret;
61. }
62.
63. int main(int argc, char* argv[])
64. {
65.     AVFormatContext* pFormatCtx;
66.     AVOutputFormat* fmt;
67.     AVStream* video_st;
68.     AVCodecContext* pCodecCtx;
69.     AVCodec* pCodec;
70.
71.     uint8_t* picture_buf;
72.     AVFrame* picture;
73.     int size;
74.
75.     //FILE *in_file = fopen("src01_480x272.yuv", "rb"); //Input YUV data 视频YUV源文件
76.     FILE *in_file = fopen("ds_480x272.yuv", "rb"); //Input YUV data 视频YUV源文件
77.     int in_w=480,in_h=272;//宽高
78.     //Frames to encode
79.     int framenum=100;
80.     //const char* out_file = "src01.h264"; //Output Filepath 输出文件路径
81.     //const char* out_file = "src01.ts";
82.     //const char* out_file = "src01.hevc";
83.     const char* out_file = "ds.hevc";
```

```

84.
85.     av_register_all();
86.     //Method1 方法1.组合使用几个函数
87.     pFormatCtx = avformat_alloc_context();
88.     //Guess Format 猜格式
89.     fmt = av_guess_format(NULL, out_file, NULL);
90.     pFormatCtx->oformat = fmt;
91.
92.     //Method 2 方法2.更加自动化一些
93.     //avformat_alloc_output_context2(&pFormatCtx, NULL, NULL, out_file);
94.     //fmt = pFormatCtx->oformat;
95.
96.
97.     //Output Format 注意输出路径
98.     if (avio_open(&pFormatCtx->pb, out_file, AVIO_FLAG_READ_WRITE) < 0)
99.     {
100.         printf("Failed to open output file! 输出文件打开失败");
101.         return -1;
102.     }
103.
104.     video_st = avformat_new_stream(pFormatCtx, 0);
105.     video_st->time_base.num = 1;
106.     video_st->time_base.den = 25;
107.
108.     if (video_st==NULL)
109.     {
110.         return -1;
111.     }
112.     //Param that must set
113.     pCodecCtx = video_st->codec;
114.     //pCodecCtx->codec_id = AV_CODEC_ID_HEVC;
115.     pCodecCtx->codec_id = fmt->video_codec;
116.     pCodecCtx->codec_type = AVMEDIA_TYPE_VIDEO;
117.     pCodecCtx->pix_fmt = PIX_FMT_YUV420P;
118.     pCodecCtx->width = in_w;
119.     pCodecCtx->height = in_h;
120.     pCodecCtx->time_base.num = 1;
121.     pCodecCtx->time_base.den = 25;
122.     pCodecCtx->bit_rate = 400000;
123.     pCodecCtx->gop_size=250;
124.     //H264
125.     //pCodecCtx->me_range = 16;
126.     //pCodecCtx->max_qdiff = 4;
127.     //pCodecCtx->qcompress = 0.6;
128.     pCodecCtx->qmin = 10;
129.     pCodecCtx->qmax = 51;
130.
131.     //Optional Param
132.     pCodecCtx->max_b_frames=3;
133.
134.     // Set Option
135.     AVDictionary *param = 0;
136.     //H.264
137.     if(pCodecCtx->codec_id == AV_CODEC_ID_H264) {
138.         av_dict_set(&param, "preset", "slow", 0);
139.         av_dict_set(&param, "tune", "zerolatency", 0);
140.     }
141.     //H.265
142.     if(pCodecCtx->codec_id == AV_CODEC_ID_H265){
143.         av_dict_set(&param, "x265-params", "qp=20", 0);
144.         av_dict_set(&param, "preset", "ultrafast", 0);
145.         av_dict_set(&param, "tune", "zero-latency", 0);
146.     }
147.
148.     //Dump Information 输出格式信息
149.     av_dump_format(pFormatCtx, 0, out_file, 1);
150.
151.     pCodec = avcodec_find_encoder(pCodecCtx->codec_id);
152.     if (!pCodec){
153.         printf("Can not find encoder! 没有找到合适的编码器!\n");
154.         return -1;
155.     }
156.     if (avcodec_open2(pCodecCtx, pCodec, &param) < 0){
157.         printf("Failed to open encoder! 编码器打开失败!\n");
158.         return -1;
159.     }
160.
161.
162.
163.     picture = avcodec_alloc_frame();
164.     size = avpicture_get_size(pCodecCtx->pix_fmt, pCodecCtx->width, pCodecCtx->height);
165.     picture_buf = (uint8_t *)av_malloc(size);
166.     avpicture_fill((AVPicture *)picture, picture_buf, pCodecCtx->pix_fmt, pCodecCtx->width, pCodecCtx->height);
167.
168.     //Write File Header 写文件头
169.     avformat_write_header(pFormatCtx, NULL);
170.
171.     AVPacket pkt;
172.     int y_size = pCodecCtx->width * pCodecCtx->height;
173.     av_new_packet(&pkt, y_size*3);
174.

```

```

175.     for (int i=0; i<framenum; i++){
176.         //Read YUV 读入YUV
177.         if (fread(picture_buf, 1, y_size*3/2, in_file) < 0){
178.             printf("Failed to read YUV data! 文件读取错误\n");
179.             return -1;
180.         }else if(feof(in_file)){
181.             break;
182.         }
183.         picture->data[0] = picture_buf; // 亮度Y
184.         picture->data[1] = picture_buf+ y_size; // U
185.         picture->data[2] = picture_buf+ y_size*5/4; // V
186.         //PTS
187.         picture->pts=i;
188.         int got_picture=0;
189.         //Encode 编码
190.         int ret = avcodec_encode_video2(pCodecCtx, &pkt,picture, &got_picture);
191.         if (ret < 0){
192.             printf("Failed to encode! 编码错误!\n");
193.             return -1;
194.         }
195.         if (got_picture==1){
196.             printf("Succeed to encode 1 frame! 编码成功1帧!\n");
197.             pkt.stream_index = video_st->index;
198.             ret = av_write_frame(pFormatCtx, &pkt);
199.             av_free_packet(&pkt);
200.         }
201.     }
202.     //Flush Encoder
203.     int ret = flush_encoder(pFormatCtx,0);
204.     if (ret < 0) {
205.         printf("Flushing encoder failed\n");
206.         return -1;
207.     }
208.
209.     //Write file trailer 写文件尾
210.     av_write_trailer(pFormatCtx);
211.
212.     //Clean 清理
213.     if (video_st){
214.         avcodec_close(video_st->codec);
215.         av_free(picture);
216.         av_free(picture_buf);
217.     }
218.     avio_close(pFormatCtx->pb);
219.     avformat_free_context(pFormatCtx);
220.
221.     fclose(in_file);
222.
223.     return 0;
224. }

```

结果

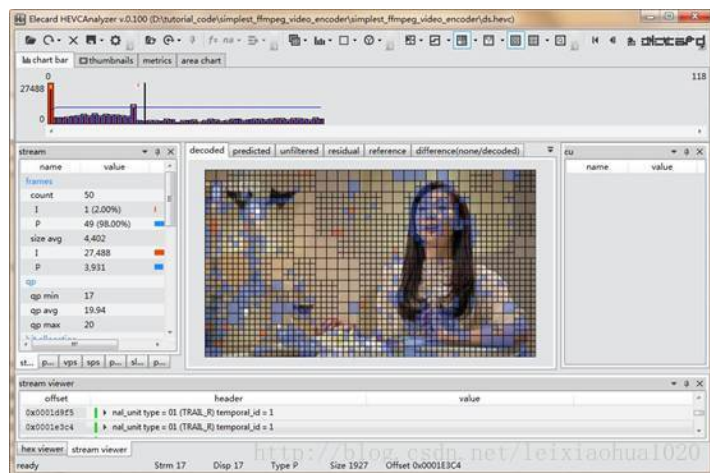
软件运行截图（受限于文件体积，原始YUV帧数只有100帧）：

[illegible]

这次换了个有趣点的YUV序列。之前总是看YUV标准测试序列都已经看烦了，这次换个电视剧里的序列相对更加生动一些。YUV序列如下图所示。



编码后的HEVC (H.265) 码流：



下载

Simplest ffmpeg video encoder

项目主页

SourceForge：<https://sourceforge.net/projects/simplestffmpegvideoencoder/>

Github：https://github.com/leixiaohua1020/simplest_ffmpeg_video_encoder

开源中国：http://git.oschina.net/leixiaohua1020/simplest_ffmpeg_video_encoder

CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/8001515>

本程序实现了YUV像素数据编码为视频码流（H.265，H264，MPEG2，VP8等等）。是最简单的FFmpeg视频编码方面的教程。它包含以下两个子项目：

simplest_ffmpeg_video_encoder：最简单的基于FFmpeg的视频编码器。使用libavcodec和libavformat编码并且封装视频。

simplest_ffmpeg_video_encoder_pure：最简单的基于FFmpeg的视频编码器-纯净版。仅使用libavcodec编码视频，不使用libavformat。

更新-1.1 (2015.1.03)=====

增加了《最简单的基于FFmpeg的编码器-纯净版（不包含libavformat）》中的simplest_ffmpeg_video_encoder_pure工程。

CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/8322003>

更新-1.2 (2015.2.13)=

这次考虑到了跨平台的要求，调整了源代码。经过这次调整之后，源代码可以在以下平台编译通过：

VC++：打开sln文件即可编译，无需配置。

cl.exe：打开compile_cl.bat即可命令行下使用cl.exe进行编译，注意可能需要按照VC的安装路径调整脚本里面的参数。编译命令如下。

```
[plain]
1.  ::VS2010 Environment
2.  call "D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat"
3.  ::include
4.  @set INCLUDE=include;%INCLUDE%
5.  ::lib
6.  @set LIB=lib;%LIB%
7.  ::compile and link
8.  cl simplest_ffmpeg_video_encoder.cpp /link avcodec.lib avformat.lib avutil.lib ^
9.  avdevice.lib avfilter.lib postproc.lib swresample.lib swscale.lib /OPT:NOREF
```

MinGW：MinGW命令行下运行compile_mingw.sh即可使用MinGW的g++进行编译。编译命令如下。

```
[plain]
1.  g++ simplest_ffmpeg_video_encoder.cpp -g -o simplest_ffmpeg_video_encoder.exe \
2.  -I /usr/local/include -L /usr/local/lib \
3.  -lavformat -lavcodec -lavutil
```

GCC：Linux或者MacOS命令行下运行compile_gcc.sh即可使用GCC进行编译。编译命令如下。

```
[plain]
1.  gcc simplest_ffmpeg_video_encoder.cpp -g -o simplest_ffmpeg_video_encoder.out \
2.  -I /usr/local/include -L /usr/local/lib -lavformat -lavcodec -lavutil
```

PS：相关的编译命令已经保存到了工程文件夹中

CSDN下载地址：<http://download.csdn.net/detail/leixiaohua1020/8444967>

SourceForge上已经更新。

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/39770947>

文章标签：[ffmpeg](#) [HEVC](#) [h.265](#) [编码](#) [YUV](#)

个人分类：[我的开源项目](#) [FFMPEG](#)

所属专栏：[FFmpeg](#)

此PDF由spygg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com