

最简单的基于FFmpeg的编码器-纯净版（不包含libavformat）

2015年01月03日 12:10:48 阅读数：17390

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）](#)

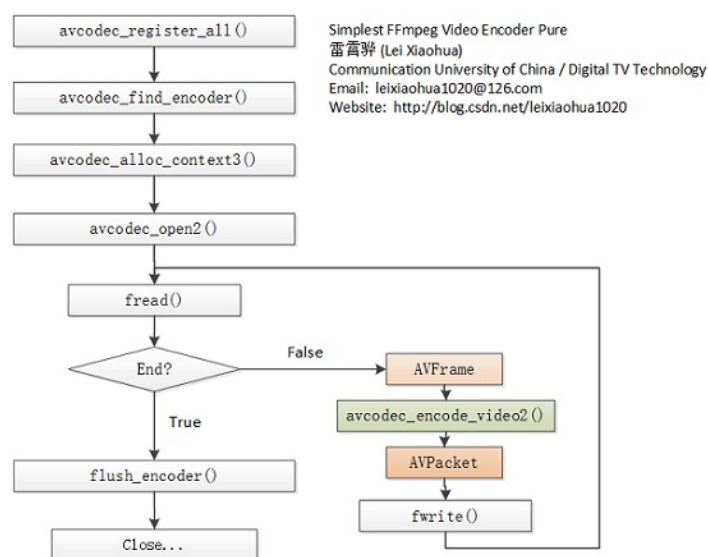
本文记录一个更加“纯净”的基于FFmpeg的视频编码器。此前记录过一个基于FFmpeg的视频编码器：

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

这个视频编码器调用了FFmpeg中的libavformat和libavcodec两个库完成了视频编码工作。但是这并不是一个“纯净”的编码器。上述两个库中libavformat完成封装格式处理，而libavcodec完成编码工作。一个“纯净”的编码器，理论上说只需要使用libavcodec就足够了，并不需要使用libavformat。本文记录的编码器就是这样的一个“纯净”的编码器，它仅仅通过调用libavcodec将YUV数据编码为H.264/HEVC等格式的压缩视频码流。

流程图

仅使用libavcodec（不使用libavformat）编码视频的流程图如下图所示。



流程图中关键函数的作用如下所列：

- avcodec_register_all()：注册所有的编解码器。
- avcodec_find_encoder()：查找编码器。
- avcodec_alloc_context3()：为AVCodecContext分配内存。
- avcodec_open2()：打开编码器。
- avcodec_encode_video2()：编码一帧数据。

两个存储数据的结构体如下所列：

- AVFrame：存储一帧未编码的像素数据。
- AVPacket：存储一帧压缩编码数据。

对比

简单记录一下这个只使用libavcodec的“纯净版”视频编码器和使用libavcodec+libavformat的视频编码器的不同。

PS：使用libavcodec+libavformat的编码器参考文章 [《最简单的基于FFmpeg的视频编码器-更新版（YUV编码为HEVC\(H.265\)）》](#)

下列与libavformat相关的函数在“纯净版”视频编码器中都不存在。

av_register_all()：注册所有的编解码器，复用/解复用器等等组件。其中调用了avcodec_register_all()注册所有编解码器相关的组件。

avformat_alloc_context()：创建AVFormatContext结构体。

avformat_alloc_output_context2()：初始化一个输出流。

avio_open()：打开输出文件。

avformat_new_stream()：创建AVStream结构体。avformat_new_stream()中会调用avcodec_alloc_context3()创建AVCodecContext结构体。

avformat_write_header()：写文件头。

av_write_frame()：写编码后的文件帧。

av_write_trailer()：写文件尾。

(2)

新增了如下几个函数

avcodec_register_all()：只注册编解码器有关的组件。

avcodec_alloc_context3()：创建AVCodecContext结构体。

可以看出，相比于“完整”的编码器，这个纯净的编码器函数调用更加简单，功能相对少一些，相对来说更加的“轻量”。

源代码

```
[cpp]
1.  /**
2.   * 最简单的基于FFmpeg的视频编码器（纯净版）
3.   * Simplest FFmpeg Video Encoder Pure
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（H264，MPEG2，VP8等等）。
12.  * 它仅仅使用了libavcodec（而没有使用libavformat）。
13.  * 是最简单的FFmpeg视频编码方面的教程。
14.  * 通过学习本例子可以了解FFmpeg的编码流程。
15.  * This software encode YUV420P data to video bitstream
16.  * (Such as H.264, H.265, VP8, MPEG2 etc).
17.  * It only uses libavcodec to encode video (without libavformat)
18.  * It's the simplest video encoding software based on FFmpeg.
19.  * Suitable for beginner of FFmpeg
20.  */
21.
22.
23. #include <stdio.h>
24.
25. #define __STDC_CONSTANT_MACROS
26.
27. #ifdef _WIN32
28. //Windows
29. extern "C"
30. {
31. #include "libavutil/opt.h"
32. #include "libavcodec/avcodec.h"
33. #include "libavutil/imgutils.h"
34. };
35. #else
36. //Linux...
37. #ifdef __cplusplus
38. extern "C"
39. {
40. #endif
41. #include <libavutil/opt.h>
42. #include <libavcodec/avcodec.h>
43. #include <libavutil/imgutils.h>
44. #ifdef __cplusplus
45. };
46. #endif
47. #endif
48.
49. //test different codec
50. #define TEST_H264 1
51. #define TEST_HEVC 0
52.
53.
54. int main(int argc, char* argv[])
55. {
56.     AVCodec *pCodec;
57.     AVCodecContext *pCodecCtx= NULL;
58.     int i, ret, got_output;
59.     FILE *fp_in;
60.     FILE *fp_out;
61.     AVFrame *pFrame;
62.     AVPacket pkt;
63.     int y_size;
```

```

64.     int framecnt=0;
65.
66.     char filename_in[]="../ds_480x272.yuv";
67.
68.     #if TEST_HEVC
69.         AVCodecID codec_id=AV_CODEC_ID_HEVC;
70.         char filename_out[]="ds.hevc";
71.     #else
72.         AVCodecID codec_id=AV_CODEC_ID_H264;
73.         char filename_out[]="ds.h264";
74.     #endif
75.
76.
77.     int in_w=480,in_h=272;
78.     int framenum=100;
79.
80.     avcodec_register_all();
81.
82.     pCodec = avcodec_find_encoder(codec_id);
83.     if (!pCodec) {
84.         printf("Codec not found\n");
85.         return -1;
86.     }
87.     pCodecCtx = avcodec_alloc_context3(pCodec);
88.     if (!pCodecCtx) {
89.         printf("Could not allocate video codec context\n");
90.         return -1;
91.     }
92.     pCodecCtx->bit_rate = 400000;
93.     pCodecCtx->width = in_w;
94.     pCodecCtx->height = in_h;
95.     pCodecCtx->time_base.num=1;
96.     pCodecCtx->time_base.den=25;
97.     pCodecCtx->gop_size = 10;
98.     pCodecCtx->max_b_frames = 1;
99.     pCodecCtx->pix_fmt = AV_PIX_FMT_YUV420P;
100.
101.     if (codec_id == AV_CODEC_ID_H264)
102.         av_opt_set(pCodecCtx->priv_data, "preset", "slow", 0);
103.
104.     if (avcodec_open2(pCodecCtx, pCodec, NULL) < 0) {
105.         printf("Could not open codec\n");
106.         return -1;
107.     }
108.
109.     pFrame = av_frame_alloc();
110.     if (!pFrame) {
111.         printf("Could not allocate video frame\n");
112.         return -1;
113.     }
114.     pFrame->format = pCodecCtx->pix_fmt;
115.     pFrame->width = pCodecCtx->width;
116.     pFrame->height = pCodecCtx->height;
117.
118.     ret = av_image_alloc(pFrame->data, pFrame->linesize, pCodecCtx->width, pCodecCtx->height,
119.                         pCodecCtx->pix_fmt, 16);
120.     if (ret < 0) {
121.         printf("Could not allocate raw picture buffer\n");
122.         return -1;
123.     }
124.     //Input raw data
125.     fp_in = fopen(filename_in, "rb");
126.     if (!fp_in) {
127.         printf("Could not open %s\n", filename_in);
128.         return -1;
129.     }
130.     //Output bitstream
131.     fp_out = fopen(filename_out, "wb");
132.     if (!fp_out) {
133.         printf("Could not open %s\n", filename_out);
134.         return -1;
135.     }
136.
137.     y_size = pCodecCtx->width * pCodecCtx->height;
138.     //Encode
139.     for (i = 0; i < framenum; i++) {
140.         av_init_packet(&pkt);
141.         pkt.data = NULL; // packet data will be allocated by the encoder
142.         pkt.size = 0;
143.         //Read raw YUV data
144.         if (fread(pFrame->data[0],1,y_size,fp_in)<= 0|| // Y
145.             fread(pFrame->data[1],1,y_size/4,fp_in)<= 0|| // U
146.             fread(pFrame->data[2],1,y_size/4,fp_in)<= 0){ // V
147.             return -1;
148.         }else if(feof(fp_in)){
149.             break;
150.         }
151.
152.         pFrame->pts = i;
153.         /* encode the image */
154.         ret = avcodec_encode_video2(pCodecCtx, &pkt, pFrame, &got_output);

```

```

155.         if (ret < 0) {
156.             printf("Error encoding frame\n");
157.             return -1;
158.         }
159.         if (got_output) {
160.             printf("Succeed to encode frame: %5d\tsize:%5d\n", framecnt, pkt.size);
161.             framecnt++;
162.             fwrite(pkt.data, 1, pkt.size, fp_out);
163.             av_free_packet(&pkt);
164.         }
165.     }
166.     //Flush Encoder
167.     for (got_output = 1; got_output; i++) {
168.         ret = avcodec_encode_video2(pCodecCtx, &pkt, NULL, &got_output);
169.         if (ret < 0) {
170.             printf("Error encoding frame\n");
171.             return -1;
172.         }
173.         if (got_output) {
174.             printf("Flush Encoder: Succeed to encode 1 frame!\tsize:%5d\n", pkt.size);
175.             fwrite(pkt.data, 1, pkt.size, fp_out);
176.             av_free_packet(&pkt);
177.         }
178.     }
179.
180.     fclose(fp_out);
181.     avcodec_close(pCodecCtx);
182.     av_free(pCodecCtx);
183.     av_freep(&pFrame->data[0]);
184.     av_frame_free(&pFrame);
185.
186.     return 0;
187. }

```

运行结果

通过设定定义在程序开始的宏，确定需要使用的编码器。

```

1.  //test different codec
2.  #define TEST_H264 0
3.  #define TEST_HEVC 1

```

当TEST_H264设置为1的时候，编码H.264文件“ds.h264”。

当TEST_HEVC设置为1的时候，解码HEVC文件“ds.hevc”。

输入文件是“ds_480x272.yuv”。

程序运行的截图如下所示。

```

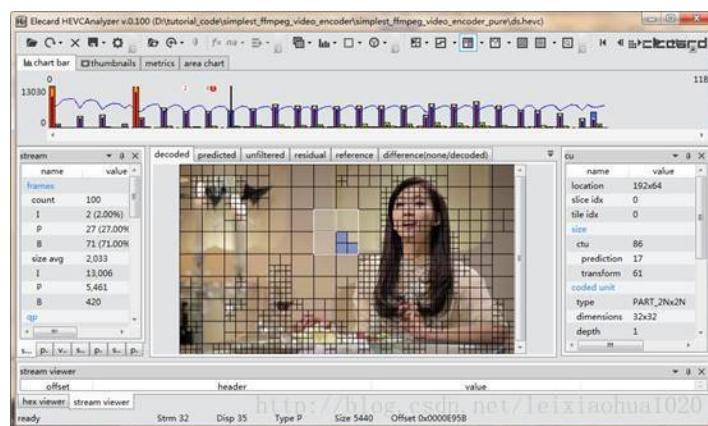
D:\tutorial_code\simplest_ffmpeg_video_encoder\Debug\simplest_ffmpeg_video_encoder...
x265 [info]: HEVC encoder version 1.3+55-c21ddfe6a6d9e234
x265 [info]: build info (Windows)[GCC 4.8.3][32 bit] 8hpp
x265 [info]: using cpu capabilities: MMX2 SSE2Fast SSSE3 SSE4.2 AVX
x265 [info]: VPP streams / pool / frames : 5 / 4 / 1
x265 [info]: Main profile, Level-2.1 (Main tier)
x265 [info]: CU size : 64
x265 [info]: Max RQI depth inter / intra : 1 / 1
x265 [info]: ME / range / subpel / merge : hex / 57 / 2 / 2
x265 [info]: KeyFrame min / max / scenecut : 25 / 250 / 40
x265 [info]: Lookahead / bframes / badapt : 20 / 4 / 2
x265 [info]: b-pyramid / weightp / weightb / refs : 1 / 1 / 0 / 3
x265 [info]: Rate Control / AQ-Strength / CUIree : ABR-400 kbps / 1.0 / 1
x265 [info]: tools: rd=3 lft sao-lcu signhide
Succeed to encode frame: 0 size:13030
Succeed to encode frame: 1 size: 1251
Succeed to encode frame: 2 size: 104
Succeed to encode frame: 3 size: 52
Succeed to encode frame: 4 size: 45
Succeed to encode frame: 5 size: 3716
Succeed to encode frame: 6 size: 127
Succeed to encode frame: 7 size: 53
Succeed to encode frame: 8 size: 72
Succeed to encode frame: 9 size: 4041
Succeed to encode frame: 10 size: 198
Succeed to encode frame: 11 size: 71

```

输入的YUV文件如下图所示。



输出的HEVC文件如下图所示。



下载

Simplest ffmpeg encoder pure工程被作为子工程添加到了simplest ffmpeg video encoder工程中。新版的simplest ffmpeg video encoder的信息如下。

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

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。

version 1.1

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_pure.cpp /link avcodec.lib avutil.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_pure.cpp -g -o simplest_ffmpeg_video_encoder_pure.exe \
2.  -I /usr/local/include -L /usr/local/lib \
3.  -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_pure.cpp -g -o simplest_ffmpeg_video_encoder_pure.out \
2.  -I /usr/local/include -L /usr/local/lib -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/42181271>

文章标签 : [ffmpeg](#) [视频](#) [编码](#) [libavcodec](#)

个人分类 : [FFMPEG](#) [我的开源项目](#)

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