原 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：17488

==========

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)）》

(1)
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
/**
 * 最简单的基于FFmpeg的视频编码器（纯净版）
 * Simplest FFmpeg Video Encoder Pure
 *
 * 雷霄骅 Lei Xiaohua
 * leixiaohua1020@126.com
 * 中国传媒大学/数字电视技术
 * Communication University of China / Digital TV Technology
 * http://blog.csdn.net/leixiaohua1020
 *
 * 本程序实现了YUV像素数据编码为视频码流（H264，MPEG2，VP8等等）。
 * 它仅仅使用了libavcodec（而没有使用libavformat）。
 * 是最简单的FFmpeg视频编码方面的教程。
 * 通过学习本例子可以了解FFmpeg的编码流程。
 * This software encode YUV420P data to video bitstream
 * (Such as H.264, H.265, VP8, MPEG2 etc).
 * It only uses libavcodec to encode video (without libavformat)
 * It's the simplest video encoding software based on FFmpeg.
 * Suitable for beginner of FFmpeg
 */


#include <stdio.h>

#define __STDC_CONSTANT_MACROS

#ifdef _WIN32
//Windows
extern "C"
{
#include "libavutil/opt.h"
#include "libavcodec/avcodec.h"
#include "libavutil/imgutils.h"
};
#else
//Linux...
#ifdef __cplusplus
extern "C"
{
#endif
#include <libavutil/opt.h>
#include <libavcodec/avcodec.h>
#include <libavutil/imgutils.h>
#ifdef __cplusplus
};
#endif
#endif

//test different codec
#define TEST_H264  1
#define TEST_HEVC  0


int main(int argc, char* argv[])
{
    AVCodec *pCodec;
    AVCodecContext *pCodecCtx= NULL;
    int i, ret, got_output;
    FILE *fp_in;
    FILE *fp_out;
    AVFrame *pFrame;
    AVPacket pkt;
    int y_size;
    int framecnt=0;

    char filename_in[]="../ds_480x272.yuv";

#if TEST_HEVC
    AVCodecID codec_id=AV_CODEC_ID_HEVC;
    char filename_out[]="ds.hevc";
#else
    AVCodecID codec_id=AV_CODEC_ID_H264;
    char filename_out[]="ds.h264";
#endif


    int in_w=480,in_h=272;
    int framenum=100;

    avcodec_register_all();

    pCodec = avcodec_find_encoder(codec_id);
    if (!pCodec) {
        printf("Codec not found\n");
        return -1;
    }
    pCodecCtx = avcodec_alloc_context3(pCodec);
```

```
    if (!pCodecCtx) {
        printf("Could not allocate video codec context\n");
        return -1;
    }
    pCodecCtx->bit_rate = 400000;
    pCodecCtx->width = in_w;
    pCodecCtx->height = in_h;
    pCodecCtx->time_base.num=1;
    pCodecCtx->time_base.den=25;
    pCodecCtx->gop_size = 10;
    pCodecCtx->max_b_frames = 1;
    pCodecCtx->pix_fmt = AV_PIX_FMT_YUV420P;

    if (codec_id == AV_CODEC_ID_H264)
        av_opt_set(pCodecCtx->priv_data, "preset", "slow", 0);

    if (avcodec_open2(pCodecCtx, pCodec, NULL) < 0) {
        printf("Could not open codec\n");
        return -1;
    }

    pFrame = av_frame_alloc();
    if (!pFrame) {
        printf("Could not allocate video frame\n");
        return -1;
    }
    pFrame->format = pCodecCtx->pix_fmt;
    pFrame->width  = pCodecCtx->width;
    pFrame->height = pCodecCtx->height;

    ret = av_image_alloc(pFrame->data, pFrame->linesize, pCodecCtx->width, pCodecCtx->height,
                         pCodecCtx->pix_fmt, 16);
    if (ret < 0) {
        printf("Could not allocate raw picture buffer\n");
        return -1;
    }
    //Input raw data
    fp_in = fopen(filename_in, "rb");
    if (!fp_in) {
        printf("Could not open %s\n", filename_in);
        return -1;
    }
    //Output bitstream
    fp_out = fopen(filename_out, "wb");
    if (!fp_out) {
        printf("Could not open %s\n", filename_out);
        return -1;
    }

    y_size = pCodecCtx->width * pCodecCtx->height;
    //Encode
    for (i = 0; i < framenum; i++) {
        av_init_packet(&pkt);
        pkt.data = NULL;    // packet data will be allocated by the encoder
        pkt.size = 0;
        //Read raw YUV data
        if (fread(pFrame->data[0],1,y_size,fp_in)<= 0||       // Y
            fread(pFrame->data[1],1,y_size/4,fp_in)<= 0|| // U
            fread(pFrame->data[2],1,y_size/4,fp_in)<= 0){ // V
            return -1;
        }else if(feof(fp_in)){
            break;
        }

        pFrame->pts = i;
        /* encode the image */
        ret = avcodec_encode_video2(pCodecCtx, &pkt, pFrame, &got_output);
        if (ret < 0) {
            printf("Error encoding frame\n");
            return -1;
        }
        if (got_output) {
            printf("Succeed to encode frame: %5d\tsize:%5d\n",framecnt,pkt.size);
            framecnt++;
            fwrite(pkt.data, 1, pkt.size, fp_out);
            av_free_packet(&pkt);
        }
    }
    //Flush Encoder
    for (got_output = 1; got_output; i++) {
        ret = avcodec_encode_video2(pCodecCtx, &pkt, NULL, &got_output);
        if (ret < 0) {
            printf("Error encoding frame\n");
            return -1;
        }
        if (got_output) {
            printf("Flush Encoder: Succeed to encode 1 frame!\tsize:%5d\n",pkt.size);
            fwrite(pkt.data, 1, pkt.size, fp_out);
            av_free_packet(&pkt);
        }
    }
```

```
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。

```
[cpp]
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。

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
::VS2010 Environment
call "D:\Program Files\Microsoft Visual Studio 10.0\VC\vcvarsall.bat"
::include
@set INCLUDE=include;%INCLUDE%
::lib
@set LIB=lib;%LIB%
::compile and link
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
g++ simplest_ffmpeg_video_encoder_pure.cpp -g -o simplest_ffmpeg_video_encoder_pure.exe \
-I /usr/local/include -L /usr/local/lib \
-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
gcc simplest_ffmpeg_video_encoder_pure.cpp -g -o simplest_ffmpeg_video_encoder_pure.out \
-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。

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