# 原 最简单的基于FFmpeg的AVfilter例子（水印叠加）

2014年06月09日 00:03:51 阅读数：83895

==========

最简单的基于FFmpeg的AVfilter例子系列文章：

最简单的基于FFmpeg的AVfilter例子（水印叠加）

最简单的基于FFmpeg的AVfilter的例子-纯净版

==========

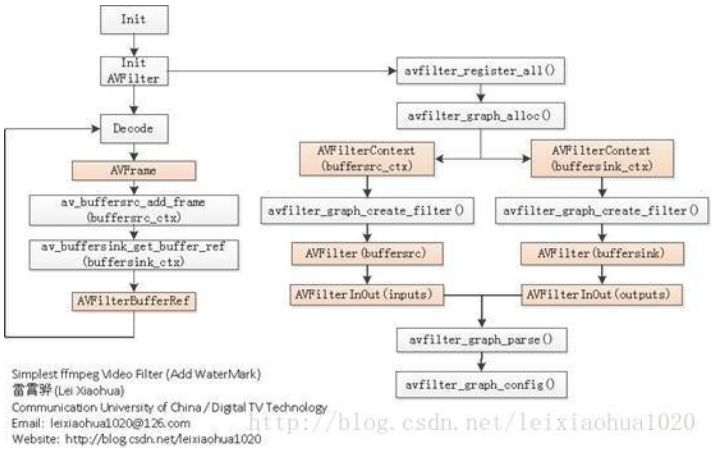
FFMPEG中有一个类库：libavfilter。该类库提供了各种视音频过滤器。之前一直没有怎么使用过这个类库，最近看了一下它的使用说明，发现还是很强大的，有很多现成的filter供使用，完成视频的处理很方便。在此将它的一个例子基础上完成了一个水印叠加器，并且移植到了VC2010下，方便开发人员学习研究它的使用方法。

该例子完成了一个水印叠加的功能。可以将一张透明背景的PNG图片作为水印叠加到一个视频文件上。需要注意的是，其叠加工作是在解码后的YUV像素数据的基础上完成的。程序支持使用SDL显示叠加后的YUV数据。也可以将叠加后的YUV输出成文件。

## 流程图

下面附一张使用FFmpeg的libavfilter的流程图。可以看出使用libavfilter还是需要做不少的初始化工作的。但是使用的时候还是比较简单的，就两个重要的函数：av_buffersrc_add_frame()和av_buffersink_get_buffer_ref()。

PS:这张图中只列出了和libavfilter有关的函数和结构体。代码中其它函数可以参考： 100行代码实现最简单的基于FFMPEG+SDL的视频播放器（SDL1.x）




## 代码

下面直接贴上代码：

```cpp
/**
 * 最简单的基于FFmpeg的AVFilter例子（叠加水印）
 * Simplest FFmpeg AVfilter Example (Watermark)
 *
 * 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
 *
 * 本程序使用FFmpeg的AVfilter实现了视频的水印叠加功能。
 * 可以将一张PNG图片作为水印叠加到视频上。
 * 是最简单的FFmpeg的AVFilter方面的教程。
 * 适合FFmpeg的初学者。
 *
 * This software uses FFmpeg's AVFilter to add watermark in a video file.
 * It can add a PNG format picture as watermark to a video file.
 * It's the simplest example based on FFmpeg's AVFilter.
 * Suitable for beginner of FFmpeg
 *
 */
```

```
#include <stdio.h>

#define __STDC_CONSTANT_MACROS

#ifdef _WIN32
#define snprintf _snprintf
//Windows
extern "C"
{
#include "libavcodec/avcodec.h"
#include "libavformat/avformat.h"
#include "libavfilter/avfiltergraph.h"
#include "libavfilter/buffersink.h"
#include "libavfilter/buffersrc.h"
#include "libavutil/avutil.h"
#include "libswscale/swscale.h"
#include "SDL/SDL.h"
};
#else
//Linux...
#ifdef __cplusplus
extern "C"
{
#endif
#include <libavcodec/avcodec.h>
#include <libavformat/avformat.h>
#include <libavfilter/avfiltergraph.h>
#include <libavfilter/buffersink.h>
#include <libavfilter/buffersrc.h>
#include <libavutil/avutil.h>
#include <libswscale/swscale.h>
#include <SDL/SDL.h>
#ifdef __cplusplus
};
#endif
#endif

//Enable SDL?
#define ENABLE_SDL 1
//Output YUV data?
#define ENABLE_YUVFILE 1

const char *filter_descr = "movie=my_logo.png[wm];[in][wm]overlay=5:5[out]";

static AVFormatContext *pFormatCtx;
static AVCodecContext *pCodecCtx;
AVFilterContext *buffersink_ctx;
AVFilterContext *buffersrc_ctx;
AVFilterGraph *filter_graph;
static int video_stream_index = -1;



static int open_input_file(const char *filename)
{
    int ret;
    AVCodec *dec;

    if ((ret = avformat_open_input(&pFormatCtx, filename, NULL, NULL)) < 0) {
        printf( "Cannot open input file\n");
        return ret;
    }

    if ((ret = avformat_find_stream_info(pFormatCtx, NULL)) < 0) {
        printf( "Cannot find stream information\n");
        return ret;
    }

    /* select the video stream */
    ret = av_find_best_stream(pFormatCtx, AVMEDIA_TYPE_VIDEO, -1, -1, &dec, 0);
    if (ret < 0) {
        printf( "Cannot find a video stream in the input file\n");
        return ret;
    }
    video_stream_index = ret;
    pCodecCtx = pFormatCtx->streams[video_stream_index]->codec;

    /* init the video decoder */
    if ((ret = avcodec_open2(pCodecCtx, dec, NULL)) < 0) {
        printf( "Cannot open video decoder\n");
        return ret;
    }

    return 0;
}

static int init_filters(const char *filters_descr)
{
    char args[512];
    int ret;
    AVFilter *buffersrc  = avfilter_get_by_name("buffer");
    AVFilter *buffersink = avfilter_get_by_name("ffbuffersink");
```

```
    AVFilter *buffersink = avfilter_get_by_name("buffersink");
    AVFilterInOut *outputs = avfilter_inout_alloc();
    AVFilterInOut *inputs  = avfilter_inout_alloc();
    enum AVPixelFormat pix_fmts[] = { AV_PIX_FMT_YUV420P, AV_PIX_FMT_NONE };
    AVBufferSinkParams *buffersink_params;

    filter_graph = avfilter_graph_alloc();

    /* buffer video source: the decoded frames from the decoder will be inserted here. */
    snprintf(args, sizeof(args),
            "video_size=%dx%d:pix_fmt=%d:time_base=%d/%d:pixel_aspect=%d/%d",
            pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt,
            pCodecCtx->time_base.num, pCodecCtx->time_base.den,
            pCodecCtx->sample_aspect_ratio.num, pCodecCtx->sample_aspect_ratio.den);

    ret = avfilter_graph_create_filter(&buffersrc_ctx, buffersrc, "in",
                                       args, NULL, filter_graph);
    if (ret < 0) {
        printf("Cannot create buffer source\n");
        return ret;
    }

    /* buffer video sink: to terminate the filter chain. */
    buffersink_params = av_buffersink_params_alloc();
    buffersink_params->pixel_fmts = pix_fmts;
    ret = avfilter_graph_create_filter(&buffersink_ctx, buffersink, "out",
                                       NULL, buffersink_params, filter_graph);
    av_free(buffersink_params);
    if (ret < 0) {
        printf("Cannot create buffer sink\n");
        return ret;
    }

    /* Endpoints for the filter graph. */
    outputs->name       = av_strdup("in");
    outputs->filter_ctx = buffersrc_ctx;
    outputs->pad_idx    = 0;
    outputs->next       = NULL;

    inputs->name       = av_strdup("out");
    inputs->filter_ctx = buffersink_ctx;
    inputs->pad_idx    = 0;
    inputs->next       = NULL;

    if ((ret = avfilter_graph_parse_ptr(filter_graph, filters_descr,
                                        &inputs, &outputs, NULL)) < 0)
        return ret;

    if ((ret = avfilter_graph_config(filter_graph, NULL)) < 0)
        return ret;
    return 0;
}


int main(int argc, char* argv[])
{
    int ret;
    AVPacket packet;
    AVFrame *pFrame;
    AVFrame *pFrame_out;

    int got_frame;

    av_register_all();
    avfilter_register_all();

    if ((ret = open_input_file("cuc_ieschool.flv")) < 0)
        goto end;
    if ((ret = init_filters(filter_descr)) < 0)
        goto end;

#if ENABLE_YUVFILE
    FILE *fp_yuv=fopen("test.yuv","wb+");
#endif
#if ENABLE_SDL
    SDL_Surface *screen;
    SDL_Overlay *bmp;
    SDL_Rect rect;
    if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
        printf( "Could not initialize SDL - %s\n", SDL_GetError());
        return -1;
    }
    screen = SDL_SetVideoMode(pCodecCtx->width, pCodecCtx->height, 0, 0);
    if(!screen) {
        printf("SDL: could not set video mode - exiting\n");
        return -1;
    }
    bmp = SDL_CreateYUVOverlay(pCodecCtx->width, pCodecCtx->height,SDL_YV12_OVERLAY, screen);

    SDL_WM_SetCaption("Simplest FFmpeg Video Filter",NULL);
#endif
```

```
    pFrame=av_frame_alloc();
    pFrame_out=av_frame_alloc();

    /* read all packets */
    while (1) {

        ret = av_read_frame(pFormatCtx, &packet);
        if (ret< 0)
            break;

        if (packet.stream_index == video_stream_index) {
            got_frame = 0;
            ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_frame, &packet);
            if (ret < 0) {
                printf( "Error decoding video\n");
                break;
            }

            if (got_frame) {
                pFrame->pts = av_frame_get_best_effort_timestamp(pFrame);

                /* push the decoded frame into the filtergraph */
                if (av_buffersrc_add_frame(buffersrc_ctx, pFrame) < 0) {
                    printf( "Error while feeding the filtergraph\n");
                    break;
                }

                /* pull filtered pictures from the filtergraph */
                while (1) {

                    ret = av_buffersink_get_frame(buffersink_ctx, pFrame_out);
                    if (ret < 0)
                        break;

                    printf("Process 1 frame!\n");

                    if (pFrame_out->format==AV_PIX_FMT_YUV420P) {
#if ENABLE_YUVFILE
                        //Y, U, V
                        for(int i=0;i<pFrame_out->height;i++){
                            fwrite(pFrame_out->data[0]+pFrame_out->linesize[0]*i,1,pFrame_out->width,fp_yuv);
                        }
                        for(int i=0;i<pFrame_out->height/2;i++){
                            fwrite(pFrame_out->data[1]+pFrame_out->linesize[1]*i,1,pFrame_out->width/2,fp_yuv);
                        }
                        for(int i=0;i<pFrame_out->height/2;i++){
                            fwrite(pFrame_out->data[2]+pFrame_out->linesize[2]*i,1,pFrame_out->width/2,fp_yuv);
                        }
#endif

#if ENABLE_SDL
                        SDL_LockYUVOverlay(bmp);
                        int y_size=pFrame_out->width*pFrame_out->height;
                        memcpy(bmp->pixels[0],pFrame_out->data[0],y_size);   //Y
                        memcpy(bmp->pixels[2],pFrame_out->data[1],y_size/4); //U
                        memcpy(bmp->pixels[1],pFrame_out->data[2],y_size/4); //V
                        bmp->pitches[0]=pFrame_out->linesize[0];
                        bmp->pitches[2]=pFrame_out->linesize[1];
                        bmp->pitches[1]=pFrame_out->linesize[2];
                        SDL_UnlockYUVOverlay(bmp);
                        rect.x = 0;
                        rect.y = 0;
                        rect.w = pFrame_out->width;
                        rect.h = pFrame_out->height;
                        SDL_DisplayYUVOverlay(bmp, &rect);
                        //Delay 40ms
                        SDL_Delay(40);
#endif
                    }
                    av_frame_unref(pFrame_out);
                }
            }
            av_frame_unref(pFrame);
        }
        av_free_packet(&packet);
    }
#if ENABLE_YUVFILE
    fclose(fp_yuv);
#endif

end:
    avfilter_graph_free(&filter_graph);
    if (pCodecCtx)
        avcodec_close(pCodecCtx);
    avformat_close_input(&pFormatCtx);


    if (ret < 0 && ret != AVERROR_EOF) {
        char buf[1024];
        av_strerror(ret, buf, sizeof(buf));
        printf("Error occurred: %s\n", buf);
```

```
296.         return -1;
297.     }
298.
299.     return 0;
300. }
```

## 结果

程序的运行效果如图所示。

需要叠加的水印为一张PNG（透明）图片（在这里是my_logo.png）。


需要叠加的视频为一个普通的FLV格式的视频（在这里是cuc_ieschool.flv）。


程序运行的时候，会通过SDL显示水印叠加的结果，如图所示。此外，也可以将水印叠加后的解码数据输出成文件。

注：SDL显示和输出YUV可以通过程序最前面的宏控制：

```
[cpp]
1. #define ENABLE_SDL 1
2. #define ENABLE_YUVFILE 1
```


输出的YUV文件如图所示。


## 下载

**simplest ffmpeg video filter**

**项目主页**

SourceForge： https://sourceforge.net/projects/simplestffmpegvideofilter/

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

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

CSDN下载地址：

http://download.csdn.net/detail/leixiaohua1020/7465861

[一个小错误]

注：由于失误，CSDN上的项目少了一个SDL.dll文件，去SDL官网
http://www.libsdl.org/download-1.2.php
下载一个Runtime Libraries即可

PUDN下载地址（修复了SDL问题）：

http://www.pudn.com/downloads644/sourcecode/multimedia/detail2605264.html

SourceForge上已经修正该问题。

**更新-1.1 (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_filter.cpp /MD /link SDL.lib SDLmain.lib avcodec.lib ^
9. avformat.lib avutil.lib avdevice.lib avfilter.lib postproc.lib swresample.lib swscale.lib ^
10. /SUBSYSTEM:WINDOWS /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_filter.cpp -g -o simplest_ffmpeg_video_filter.exe \
2. -I /usr/local/include -L /usr/local/lib \
3. -lmingw32 -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lavfilter -lswscale
```

*GCC（Linux）：Linux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_filter.cpp -g -o simplest_ffmpeg_video_filter.out \
2. -I /usr/local/include -L /usr/local/lib \
3. -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lavfilter -lswscale
```

*GCC（Mac）：终端下运行compile_gcc_mac.sh即可使用GCC进行编译。编译命令如下。*

```
[plain]
1. gcc simplest_ffmpeg_video_filter.cpp -g -o simplest_ffmpeg_video_filter.out \
2. -framework Cocoa -I /usr/local/include -L /usr/local/lib \
3. -lSDLmain -lSDL -lavformat -lavcodec -lavutil -lavfilter -lswscale
```

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/leixiaohua1020/8445551

SourceForge上已经更新。

**更新-1.2 (2016.2.1)==========**

新增了“最简单的基于FFmpeg的avfilter的例子-纯净版”工程



文章标签： ffmpeg avfilter 水印 视频

个人分类： FFMPEG 我的开源项目

所属专栏： FFmpeg