

## 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](#)

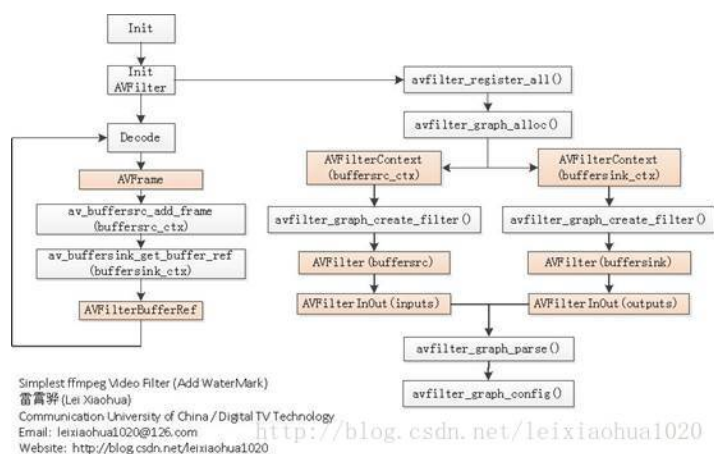
FFMPEG中有一个类库：libavfilter。该类库提供了各种视音频过滤器。之前一直没有怎么使用过这个类库，最近看了一下它的使用说明，发现还是很强大的，有很多现成的filter供使用，完成视频的处理很方便。在此将它的一个例子基础上完成了一个水印叠加器，并且移植到了VC2010下，方便开发人员学习研究它的使用方法。

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]
1.  /**
2.   * 最简单的基于FFmpeg的AVFilter例子（叠加水印）
3.   *  Simplest FFmpeg AVfilter Example (Watermark)
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.  *  本程序使用FFmpeg的AVfilter实现了视频的水印叠加功能。
12.  *  可以将一张PNG图片作为水印叠加到视频上。
13.  *  是最简单的FFmpeg的AVFilter方面的教程。
14.  *  适合FFmpeg的初学者。
15.  *
16.  *  This software uses FFmpeg's AVFilter to add watermark in a video file.
17.  *  It can add a PNG format picture as watermark to a video file.
18.  *  It's the simplest example based on FFmpeg's AVFilter.
19.  *  Suitable for beginner of FFmpeg
20.  *
21.  */
```

```

22. #include <stdio.h>
23.
24. #define __STDC_CONSTANT_MACROS
25.
26. #ifdef _WIN32
27. #define snprintf _snprintf
28. //Windows
29. extern "C"
30. {
31. #include "libavcodec/avcodec.h"
32. #include "libavformat/avformat.h"
33. #include "libavfilter/avfiltergraph.h"
34. #include "libavfilter/buffersink.h"
35. #include "libavfilter/buffersrc.h"
36. #include "libavutil/avutil.h"
37. #include "libswscale/swscale.h"
38. #include "SDL/SDL.h"
39. };
40. #else
41. //Linux...
42. #ifdef __cplusplus
43. extern "C"
44. {
45. #endif
46. #include <libavcodec/avcodec.h>
47. #include <libavformat/avformat.h>
48. #include <libavfilter/avfiltergraph.h>
49. #include <libavfilter/buffersink.h>
50. #include <libavfilter/buffersrc.h>
51. #include <libavutil/avutil.h>
52. #include <libswscale/swscale.h>
53. #include <SDL/SDL.h>
54. #ifdef __cplusplus
55. };
56. #endif
57. #endif
58.
59. //Enable SDL?
60. #define ENABLE_SDL 1
61. //Output YUV data?
62. #define ENABLE_YUVFILE 1
63.
64. const char *filter_descr = "movie=my_logo.png[wm];[in][wm]overlay=5:5[out]";
65.
66. static AVFormatContext *pFormatCtx;
67. static AVCodecContext *pCodecCtx;
68. AVFilterContext *buffersink_ctx;
69. AVFilterContext *buffersrc_ctx;
70. AVFilterGraph *filter_graph;
71. static int video_stream_index = -1;
72.
73.
74.
75. static int open_input_file(const char *filename)
76. {
77.     int ret;
78.     AVCodec *dec;
79.
80.     if ((ret = avformat_open_input(&pFormatCtx, filename, NULL, NULL)) < 0) {
81.         printf( "Cannot open input file\n");
82.         return ret;
83.     }
84.
85.     if ((ret = avformat_find_stream_info(pFormatCtx, NULL)) < 0) {
86.         printf( "Cannot find stream information\n");
87.         return ret;
88.     }
89.
90.     /* select the video stream */
91.     ret = av_find_best_stream(pFormatCtx, AVMEDIA_TYPE_VIDEO, -1, -1, &dec, 0);
92.     if (ret < 0) {
93.         printf( "Cannot find a video stream in the input file\n");
94.         return ret;
95.     }
96.     video_stream_index = ret;
97.     pCodecCtx = pFormatCtx->streams[video_stream_index]->codec;
98.
99.     /* init the video decoder */
100.    if ((ret = avcodec_open2(pCodecCtx, dec, NULL)) < 0) {
101.        printf( "Cannot open video decoder\n");
102.        return ret;
103.    }
104.
105.    return 0;
106. }
107.
108. static int init_filters(const char *filters_descr)
109. {
110.     char args[512];
111.     int ret;
112.     AVFilter *buffersrc = avfilter_get_by_name("buffer");
113.     AVFilter *buffersink = avfilter_get_by_name("buffersink");

```

```

113.     AVFilter *buffersink = avfilter_get_by_name("buffersink");
114.     AVFilterInOut *outputs = avfilter_inout_alloc();
115.     AVFilterInOut *inputs = avfilter_inout_alloc();
116.     enum AVPixelFormat pix_fmts[] = { AV_PIX_FMT_YUV420P, AV_PIX_FMT_NONE };
117.     AVBufferSinkParams *buffersink_params;
118.
119.     filter_graph = avfilter_graph_alloc();
120.
121.     /* buffer video source: the decoded frames from the decoder will be inserted here. */
122.     snprintf(args, sizeof(args),
123.              "video_size=%dx%d:pix_fmt=%d:time_base=%d/%d:pixel_aspect=%d/%d",
124.              pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt,
125.              pCodecCtx->time_base.num, pCodecCtx->time_base.den,
126.              pCodecCtx->sample_aspect_ratio.num, pCodecCtx->sample_aspect_ratio.den);
127.
128.     ret = avfilter_graph_create_filter(&buffersrc_ctx, buffersrc, "in",
129.                                       args, NULL, filter_graph);
130.     if (ret < 0) {
131.         printf("Cannot create buffer source\n");
132.         return ret;
133.     }
134.
135.     /* buffer video sink: to terminate the filter chain. */
136.     buffersink_params = av_buffersink_params_alloc();
137.     buffersink_params->pixel_fmts = pix_fmts;
138.     ret = avfilter_graph_create_filter(&buffersink_ctx, buffersink, "out",
139.                                       NULL, buffersink_params, filter_graph);
140.     av_free(buffersink_params);
141.     if (ret < 0) {
142.         printf("Cannot create buffer sink\n");
143.         return ret;
144.     }
145.
146.     /* Endpoints for the filter graph. */
147.     outputs->name = av_strdup("in");
148.     outputs->filter_ctx = buffersrc_ctx;
149.     outputs->pad_idx = 0;
150.     outputs->next = NULL;
151.
152.     inputs->name = av_strdup("out");
153.     inputs->filter_ctx = buffersink_ctx;
154.     inputs->pad_idx = 0;
155.     inputs->next = NULL;
156.
157.     if ((ret = avfilter_graph_parse_ptr(filter_graph, filters_descr,
158.                                       &inputs, &outputs, NULL)) < 0)
159.         return ret;
160.
161.     if ((ret = avfilter_graph_config(filter_graph, NULL)) < 0)
162.         return ret;
163.     return 0;
164. }
165.
166.
167. int main(int argc, char* argv[])
168. {
169.     int ret;
170.     AVPacket packet;
171.     AVFrame *pFrame;
172.     AVFrame *pFrame_out;
173.
174.     int got_frame;
175.
176.     av_register_all();
177.     avfilter_register_all();
178.
179.     if ((ret = open_input_file("cuc_ieschool.flv")) < 0)
180.         goto end;
181.     if ((ret = init_filters(filter_descr)) < 0)
182.         goto end;
183.
184.     #if ENABLE_YUVFILE
185.         FILE *fp_yuv=fopen("test.yuv", "wb+");
186.     #endif
187.     #if ENABLE_SDL
188.         SDL_Surface *screen;
189.         SDL_Overlay *bmp;
190.         SDL_Rect rect;
191.         if(SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO | SDL_INIT_TIMER)) {
192.             printf("Could not initialize SDL - %s\n", SDL_GetError());
193.             return -1;
194.         }
195.         screen = SDL_SetVideoMode(pCodecCtx->width, pCodecCtx->height, 0, 0);
196.         if(!screen) {
197.             printf("SDL: could not set video mode - exiting\n");
198.             return -1;
199.         }
200.         bmp = SDL_CreateYUVOverlay(pCodecCtx->width, pCodecCtx->height, SDL_YV12_OVERLAY, screen);
201.
202.         SDL_WM_SetCaption("Simplest FFmpeg Video Filter", NULL);
203.     #endif
204.

```

```

205.     pFrame=av_frame_alloc();
206.     pFrame_out=av_frame_alloc();
207.
208.     /* read all packets */
209.     while (1) {
210.
211.         ret = av_read_frame(pFormatCtx, &packet);
212.         if (ret< 0)
213.             break;
214.
215.         if (packet.stream_index == video_stream_index) {
216.             got_frame = 0;
217.             ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_frame, &packet);
218.             if (ret < 0) {
219.                 printf("Error decoding video\n");
220.                 break;
221.             }
222.
223.             if (got_frame) {
224.                 pFrame->pts = av_frame_get_best_effort_timestamp(pFrame);
225.
226.                 /* push the decoded frame into the filtergraph */
227.                 if (av_buffersrc_add_frame(buffer_src_ctx, pFrame) < 0) {
228.                     printf("Error while feeding the filtergraph\n");
229.                     break;
230.                 }
231.
232.                 /* pull filtered pictures from the filtergraph */
233.                 while (1) {
234.
235.                     ret = av_buffersink_get_frame(buffer_sink_ctx, pFrame_out);
236.                     if (ret < 0)
237.                         break;
238.
239.                     printf("Process 1 frame!\n");
240.
241.                     if (pFrame_out->format==AV_PIX_FMT_YUV420P) {
242. #if ENABLE_YUVFILE
243.                         //Y, U, V
244.                         for(int i=0;i<pFrame_out->height;i++){
245.                             fwrite(pFrame_out->data[0]+pFrame_out->linesize[0]*i,1,pFrame_out->width,fp_yuv);
246.                         }
247.                         for(int i=0;i<pFrame_out->height/2;i++){
248.                             fwrite(pFrame_out->data[1]+pFrame_out->linesize[1]*i,1,pFrame_out->width/2,fp_yuv);
249.                         }
250.                         for(int i=0;i<pFrame_out->height/2;i++){
251.                             fwrite(pFrame_out->data[2]+pFrame_out->linesize[2]*i,1,pFrame_out->width/2,fp_yuv);
252.                         }
253. #endif
254.
255. #if ENABLE_SDL
256.                         SDL_LockYUVOverlay(bmp);
257.                         int y_size=pFrame_out->width*pFrame_out->height;
258.                         memcpy(bmp->pixels[0],pFrame_out->data[0],y_size); //Y
259.                         memcpy(bmp->pixels[2],pFrame_out->data[1],y_size/4); //U
260.                         memcpy(bmp->pixels[1],pFrame_out->data[2],y_size/4); //V
261.                         bmp->pitches[0]=pFrame_out->linesize[0];
262.                         bmp->pitches[2]=pFrame_out->linesize[1];
263.                         bmp->pitches[1]=pFrame_out->linesize[2];
264.                         SDL_UnlockYUVOverlay(bmp);
265.                         rect.x = 0;
266.                         rect.y = 0;
267.                         rect.w = pFrame_out->width;
268.                         rect.h = pFrame_out->height;
269.                         SDL_DisplayYUVOverlay(bmp, &rect);
270.                         //Delay 40ms
271.                         SDL_Delay(40);
272. #endif
273.                     }
274.                     av_frame_unref(pFrame_out);
275.                 }
276.                 av_frame_unref(pFrame);
277.             }
278.             av_free_packet(&packet);
279.         }
280.     }
281. #if ENABLE_YUVFILE
282.     fclose(fp_yuv);
283. #endif
284.
285. end:
286.     avfilter_graph_free(&filter_graph);
287.     if (pCodecCtx)
288.         avcodec_close(pCodecCtx);
289.     avformat_close_input(&pFormatCtx);
290.
291.
292.     if (ret < 0 && ret != AVERROE_EOF) {
293.         char buf[1024];
294.         av_strerror(ret, buf, sizeof(buf));
295.         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：

```

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](https://github.com/leixiaohua1020/simplest_ffmpeg_video_filter)

开源中国 : [http://git.oschina.net/leixiaohua1020/simplest\\_ffmpeg\\_video\\_filter](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

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/29368911>

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](#)

此PDF由spygy生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com