

## 最简单的基于FFmpeg的AVfilter的例子-纯净版

2016年02月01日 13:55:04 阅读数：52898

最简单的基于FFmpeg的AVfilter例子系列文章：

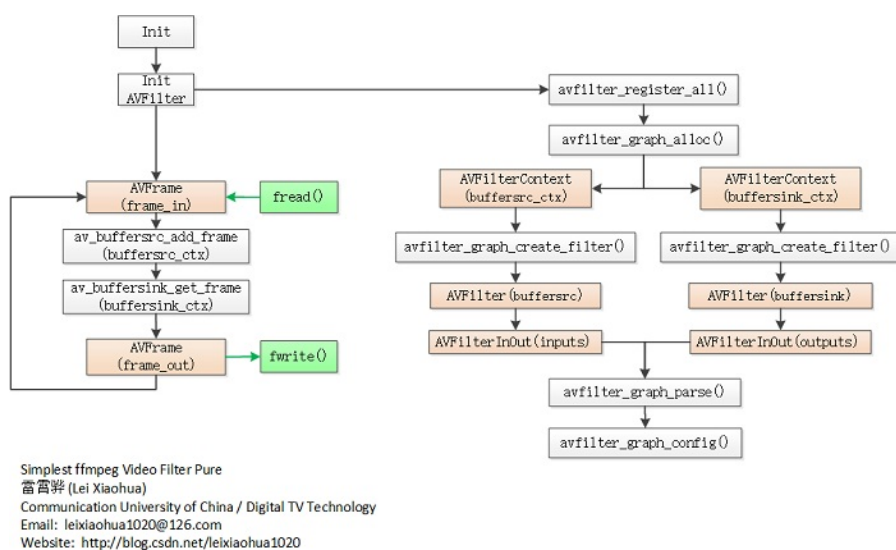
[最简单的基于FFmpeg的AVfilter例子（水印叠加）](#)

[最简单的基于FFmpeg的AVfilter的例子-纯净版](#)

有关FFmpeg的avfilter已经写过一个水印叠加的例子《[最简单的基于FFmpeg的AVfilter例子（水印叠加）](#)》，本文作为补充再记录一个纯净版的avfilter的例子。此前libavfilter一直是结合着libavcodec等类库的接口函数使用的，因此我一直以为libavfilter库与libavcodec等类库是高度耦合的（也就是如果想使用libavfilter的视音频特效功能的话必须使用libavcodec等类库的函数）。这两天空闲的时候研究了一下libavfilter的代码后发现实际情况不是这样的：libavfilter可以独立于libavcodec等类库的接口函数作为一个“纯粹”的视音频特效类库进行使用。本文记录的“纯净版”的avfilter的例子即实现了一个纯粹的视频特效添加的功能。该例子输入为一个YUV文件，输出也是一个YUV文件，通过avfilter的功能可以处理该YUV文件实现去色调、模糊、水平翻转、裁剪、加方框、叠加文字等功能。

### 流程图

该程序的流程图如下所示。AVFilter的初始化比较复杂，而使用起来比较简单。初始化的时候需要调用avfilter\_register\_all()到avfilter\_graph\_config()一系列函数。而使用的时候只有两个函数：av\_buffersrc\_add\_frame()用于向FilterGraph中加入一个AVFrame，而av\_buffersink\_get\_frame()用于从FilterGraph中取出一个AVFrame。



流程中的关键函数如下所示：

- avfilter\_register\_all()：注册所有AVFilter。
- avfilter\_graph\_alloc()：为FilterGraph分配内存。
- avfilter\_graph\_create\_filter()：创建并向FilterGraph中添加一个Filter。
- avfilter\_graph\_parse\_ptr()：将一串通过字符串描述的Graph添加到FilterGraph中。
- avfilter\_graph\_config()：检查FilterGraph的配置。
- av\_buffersrc\_add\_frame()：向FilterGraph中加入一个AVFrame。
- av\_buffersink\_get\_frame()：从FilterGraph中取出一个AVFrame。

### 代码

```
[cpp]
1. /**
2.  * 最简单的基于FFmpeg的AVFilter例子 - 纯净版
3.  * Simplest FFmpeg AVfilter Example - 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. * 本程序使用FFmpeg的AVfilter实现了YUV像素数据的滤镜处理功能。
12. * 可以给YUV数据添加各种特效功能。
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 process YUV raw data.
17. * It can add many excellent effect to YUV data.
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 "libavfilter/avfiltergraph.h"
32. #include "libavfilter/buffersink.h"
33. #include "libavfilter/buffersrc.h"
34. #include "libavutil/avutil.h"
35. #include "libavutil/imgutils.h"
36. };
37. #else
38. //Linux...
39. #ifdef __cplusplus
40. extern "C"
41. {
42. #endif
43. #include <libavfilter/avfiltergraph.h>
44. #include <libavfilter/buffersink.h>
45. #include <libavfilter/buffersrc.h>
46. #include <libavutil/avutil.h>
47. #include <libavutil/imgutils.h>
48. #ifdef __cplusplus
49. };
50. #endif
51. #endif
52.
53.
54.
55.
56. int main(int argc, char* argv[])
57. {
58.     int ret;
59.     AVFrame *frame_in;
60.     AVFrame *frame_out;
61.     unsigned char *frame_buffer_in;
62.     unsigned char *frame_buffer_out;
63.
64.     AVFilterContext *buffersink_ctx;
65.     AVFilterContext *buffersrc_ctx;
66.     AVFilterGraph *filter_graph;
67.     static int video_stream_index = -1;
68.
69.     //Input YUV
70.     FILE *fp_in=fopen("sintel_480x272_yuv420p.yuv", "rb+");
71.     if(fp_in==NULL){
72.         printf("Error open input file.\n");
73.         return -1;
74.     }
75.     int in_width=480;
76.     int in_height=272;
77.
78.     //Output YUV
79.     FILE *fp_out=fopen("output.yuv", "wb+");
80.     if(fp_out==NULL){
81.         printf("Error open output file.\n");
82.         return -1;
83.     }
84.
85.     //const char *filter_descr = "lutyuv=u=128:v=128";
86.     const char *filter_descr = "boxblur";
87.     //const char *filter_descr = "hflip";
88.     //const char *filter_descr = "hue=h=60:s=-3";
89.     //const char *filter_descr = "crop=2/3*in_w:2/3*in_h";
90.     //const char *filter_descr = "drawbox=x=100:y=100:w=100:h=100:color=pink@0.5";
91.     //const char *filter_descr = "drawtext=fontfile=arial.ttf:fontcolor=green:fontsize=30:text='Lei Xiaohua'";
92.
93.     avfilter_register_all();
94.

```

```

95.     char args[512];
96.     AVFilter *buffersrc = avfilter_get_by_name("buffer");
97.     AVFilter *buffersink = avfilter_get_by_name("ffbuffersink");
98.     AVFilterInOut *outputs = avfilter_inout_alloc();
99.     AVFilterInOut *inputs = avfilter_inout_alloc();
100.     enum PixelFormat pix_fmts[] = { AV_PIX_FMT_YUV420P, PIX_FMT_NONE };
101.     AVBufferSinkParams *buffersink_params;
102.
103.     filter_graph = avfilter_graph_alloc();
104.
105.     /* buffer video source: the decoded frames from the decoder will be inserted here. */
106.     snprintf(args, sizeof(args),
107.              "video_size=%dx%d:pix_fmt=%d:time_base=%d/%d:pixel_aspect=%d/%d",
108.              in_width,in_height,AV_PIX_FMT_YUV420P,
109.              1, 25,1,1);
110.
111.     ret = avfilter_graph_create_filter(&buffersrc_ctx, buffersrc, "in",
112.                                         args, NULL, filter_graph);
113.     if (ret < 0) {
114.         printf("Cannot create buffer source\n");
115.         return ret;
116.     }
117.
118.     /* buffer video sink: to terminate the filter chain. */
119.     buffersink_params = av_buffersink_params_alloc();
120.     buffersink_params->pixel_fmts = pix_fmts;
121.     ret = avfilter_graph_create_filter(&buffersink_ctx, buffersink, "out",
122.                                         NULL, buffersink_params, filter_graph);
123.     av_free(buffersink_params);
124.     if (ret < 0) {
125.         printf("Cannot create buffer sink\n");
126.         return ret;
127.     }
128.
129.     /* Endpoints for the filter graph. */
130.     outputs->name = av_strdup("in");
131.     outputs->filter_ctx = buffersrc_ctx;
132.     outputs->pad_idx = 0;
133.     outputs->next = NULL;
134.
135.     inputs->name = av_strdup("out");
136.     inputs->filter_ctx = buffersink_ctx;
137.     inputs->pad_idx = 0;
138.     inputs->next = NULL;
139.
140.     if ((ret = avfilter_graph_parse_ptr(filter_graph, filter_descr,
141.                                         &inputs, &outputs, NULL)) < 0)
142.         return ret;
143.
144.     if ((ret = avfilter_graph_config(filter_graph, NULL)) < 0)
145.         return ret;
146.
147.     frame_in=av_frame_alloc();
148.     frame_buffer_in=(unsigned char *)av_malloc(av_image_get_buffer_size(AV_PIX_FMT_YUV420P, in_width,in_height,1));
149.     av_image_fill_arrays(frame_in->data, frame_in->linesize,frame_buffer_in,
150.                         AV_PIX_FMT_YUV420P,in_width, in_height,1);
151.
152.     frame_out=av_frame_alloc();
153.     frame_buffer_out=(unsigned char *)av_malloc(av_image_get_buffer_size(AV_PIX_FMT_YUV420P, in_width,in_height,1));
154.     av_image_fill_arrays(frame_out->data, frame_out->linesize,frame_buffer_out,
155.                         AV_PIX_FMT_YUV420P,in_width, in_height,1);
156.
157.     frame_in->width=in_width;
158.     frame_in->height=in_height;
159.     frame_in->format=AV_PIX_FMT_YUV420P;
160.
161.     while (1) {
162.
163.         if(fread(frame_buffer_in, 1, in_width*in_height*3/2, fp_in)!= in_width*in_height*3/2){
164.             break;
165.         }
166.         /*input Y,U,V
167.         frame_in->data[0]=frame_buffer_in;
168.         frame_in->data[1]=frame_buffer_in+in_width*in_height;
169.         frame_in->data[2]=frame_buffer_in+in_width*in_height*5/4;
170.
171.         if (av_buffersrc_add_frame(buffersrc_ctx, frame_in) < 0) {
172.             printf("Error while add frame.\n");
173.             break;
174.         }
175.
176.         /* pull filtered pictures from the filtergraph */
177.         ret = av_buffersink_get_frame(buffersink_ctx, frame_out);
178.         if (ret < 0)
179.             break;
180.
181.         /*output Y,U,V
182.         if(frame_out->format==AV_PIX_FMT_YUV420P){
183.             for(int i=0;i<frame_out->height;i++){
184.                 fwrite(frame_out->data[0]+frame_out->linesize[0]*i,1,frame_out->width,fp_out);
185.             }

```

```

186.         for(int i=0;i<frame_out->height/2;i++){
187.             fwrite(frame_out->data[1]+frame_out->linesize[1]*i,1,frame_out->width/2,fp_out);
188.         }
189.         for(int i=0;i<frame_out->height/2;i++){
190.             fwrite(frame_out->data[2]+frame_out->linesize[2]*i,1,frame_out->width/2,fp_out);
191.         }
192.     }
193.     printf("Process 1 frame!\n");
194.     av_frame_unref(frame_out);
195. }
196.
197. fclose(fp_in);
198. fclose(fp_out);
199.
200. av_frame_free(&frame_in);
201. av_frame_free(&frame_out);
202. avfilter_graph_free(&filter_graph);
203.
204. return 0;
205. }

```

## 结果

本程序输入为一个名称为"sintel\_480x272\_yuv420p.yuv"的YUV420P视频数据,输出为一个名称为"output.yuv" 的YUV420P视频数据。输入的视频数据的内容如下所示。



程序中提供了几种特效：

- lutyuv='u=128:v=128'
- boxblur
- hflip
- hue='h=60:s=-3'
- crop=2/3\*in\_w:2/3\*in\_h
- drawbox=x=100:y=100:w=100:h=100:color=pink@0.5
- drawtext=fontfile=arial.ttf:fontcolor=green:fontsize=30:text='Lei Xiaohua'

可以通过修改程序中的filter\_descr字符串实现上述几种特效。下面展示几种特效的效果图。

lutyuv='u=128:v=128'



boxblur



hflip



hue='h=60:s=-3'



crop=2/3\*in\_w:2/3\*in\_h



drawbox=x=100:y=100:w=100:h=100:color=pink@0.5



drawtext=fontfile=arial.ttf:fontcolor=green:fontsize=30:text='Lei Xiaohua'



## 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/leixiaohua1020/9424521>

本程序使用包含下面两个项目：

simplest\_ffmpeg\_video\_filter：可以将一张PNG图片作为水印叠加到视频上，结合使用了libavfilter，libavcodec等类库。

simplest\_ffmpeg\_video\_filter\_pure：可以给YUV像素数据加特效，只用了libavfilter库。

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/50618190>

文章标签： [ffmpeg](#) [特效](#) [yuv](#) [滤镜](#) [libavfilter](#)

个人分类： [FFMPEG](#) [我的开源项目](#)

此PDF由spygg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com