DoubleLi
qq: 517712484 wx: ldbgliet

2021年10月

日	一	二	三	四	五	六
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

公告

昵称: DoubleLi
园龄: 11年9个月
粉丝: 2080
关注: 29
+加关注

搜索
找找看
谷歌搜索

常用链接
我的随笔
我的评论
我的参与
最新评论
我的标签

随笔分类 (5164)
android(2)
ASP.NET(30)
ASP.NET MVC(11)
Boost(118)
c#(10)
C++/C(778)
c++11(15)
cmake/autotool(66)
com/ATL/Activex(75)
Css(16)
CxImage(12)
darwin stream server(3)
DataBase(32)
DirectX(16)
Extjs(13)
更多

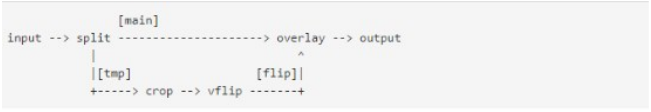
随笔档案 (3864)
2021年10月(33)
2021年9月(4)
2021年8月(10)
2021年7月(43)
2021年6月(1)
2021年5月(29)
2021年4月(15)
2021年3月(13)
2021年2月(96)
2021年1月(47)

FFMPEG 最简滤镜filter使用实例（实现视频缩放，裁剪，水印等）

FFMPEG官网给出了FFMPEG 滤镜使用的实例，它是将视频中的像素点替换成字符，然后从终端输出。我在该实例的基础上稍微的做了修改，使它能够保存滤镜处理过后的文件。在上代码之前先明白几个概念：

Filter:代表单个filter
FilterPad:代表一个filter的输入或输出端口，每个filter都可以有多个输入和多个输出，只有输出pad的filter称为source,只有输入pad的filter称为sink
FilterLink：若一个filter的输出pad和另一个filter的输入pad名字相同，即认为两个filter之间建立了link
FilterChain:代表一串相互连接的filters，除了source和sink外，要求每个filter的输入输出pad都有对应的输出和输入pad

经典示例：



图中的一系列操作共使用了四个filter，分别是
splite：将输入的流进行分裂复制，分两路输出。
crop：根据给定的参数，对视频进行裁剪
vflip：根据给定参数，对视频进行翻转等操作
overlay：将一路输入覆盖到另一路之上，合并输出为一路视频

下面上代码：

```
/*================================================================================
#     FileName: filter_video.c
#         Desc: an example of ffmpeg fileter
#       Author: licaibiao
#   LastChange: 2017-03-16
================================================================================*/
#define _XOPEN_SOURCE 600 /* for usleep */
#include <unistd.h>

#include "avcodec.h"
#include "avformat.h"
#include "avfiltergraph.h"
#include "avcodec.h"
#include "buffersink.h"
#include "buffersrc.h"
#include "opt.h"

#define SAVE_FILE

const charchar *filter_descr = "scale=iw*2:ih*2";
static AVFormatContext *fmt_ctx;
static AVCodecContext *dec_ctx;
AVFilterContext *buffersink_ctx;
AVFilterContext *buffersrc_ctx;
AVFilterGraph *filter_graph;
static int video_stream_index = -1;
static int64_t last_pts = AV_NOPTS_VALUE;

static int open_input_file(const charchar *filename)
```

2020年12月(2)
2020年11月(27)
2020年10月(44)
2020年9月(14)
2020年8月(4)
更多

文章分类 (2)
SilverLight(1)
sql server(1)

参考博客
linux驱动
回忆未来-向东
Nginx模块开发与原理剖析
大坡3D软件开发
Dean Chen的专栏
Sloan
音视频FFmpeg等
opencv教程
个人开发历程知识库
关注DirectX
chenyujing1234
morewindows
雷霄骅(leixiaohua1020)的专栏
ffmpeg参考
webrtc参考一
更多

阅读排行榜
1. Nginx之location 匹配规则详解(240898)
2. cmake使用方法详解(178488)
3. MinGW安装和使用(103448)
4. RTMP、RTSP、HTTP视频协议详解（附：直播流地址、播放软件）(102153)
5. C语言字符串操作总结大全(超详细)(94264)

评论排行榜
1. 非IE内核浏览器支持activex插件(37)
2. Nginx之location 匹配规则详解(19)
3. Javascript中定义类(15)
4. C++中的头文件和源文件(9)
5. RTSP协议详解(8)

推荐排行榜
1. C++中的头文件和源文件(25)
2. Nginx之location 匹配规则详解(22)
3. Javascript中定义类(12)
4. JavaScript中typeof知多少？(11)
5. MinGW安装和使用(9)

最新评论
1. Re:windows下搭建nginx-rtmp服务器
configuration-nginx.bat执行报错啊 'auto' 不是内部或外部命令，也不是可运行的程序 或批处理文件。 '--conf-path' 不是内部或外部命令，也不是可运行的程序 或批...
--猫爷°
2. Re:深入理解linux系统下proc文件系统内容
怎么联系作者
--ㅇㅋ
3. Re:go mod模式下引用本地包/模块(module)的方法
go mod用法，不错

```
{
    int ret;
    AVCodec *dec;

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

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

    /* select the video stream  判断流是否正常 */
    ret = av_find_best_stream(fmt_ctx, AVMEDIA_TYPE_VIDEO, -1, -1, &dec, 0);
    if (ret < 0) {
        av_log(NULL, AV_LOG_ERROR, "Cannot find a video stream in the input file\n");
        return ret;
    }
    video_stream_index = ret;
    dec_ctx = fmt_ctx->streams[video_stream_index]->codec;
    av_opt_set_int(dec_ctx, "refcounted_frames", 1, 0); /* refcounted_frames 帧引用计数 */

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

    return 0;
}

static int init_filters(const charchar *filters_descr)
{
    char args[512];
    int ret = 0;
    AVFilter *buffersrc  = avfilter_get_by_name("buffer");     /* 输入buffer filter */
    AVFilter *buffersink = avfilter_get_by_name("buffersink"); /* 输出buffer filter */
    AVFilterInOut *outputs = avfilter_inout_alloc();
    AVFilterInOut *inputs  = avfilter_inout_alloc();
    AVRational time_base = fmt_ctx->streams[video_stream_index]->time_base;   /* 时间基数 */

#ifndef SAVE_FILE
    enum AVPixelFormat pix_fmts[] = { AV_PIX_FMT_GRAY8, AV_PIX_FMT_NONE };
#else
    enum AVPixelFormat pix_fmts[] = { AV_PIX_FMT_YUV420P, AV_PIX_FMT_NONE };
#endif

    filter_graph = avfilter_graph_alloc();                    /* 创建graph  */
    if (!outputs || !inputs || !filter_graph) {
        ret = AVERROR(ENOMEM);
        goto end;
    }

    /* 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",
            dec_ctx->width, dec_ctx->height, dec_ctx->pix_fmt,
            time_base.num, time_base.den,
            dec_ctx->sample_aspect_ratio.num, dec_ctx->sample_aspect_ratio.den);

    /* 创建并向FilterGraph中添加一个Filter */
    ret = avfilter_graph_create_filter(&buffersrc_ctx, buffersrc, "in",
                                       args, NULL, filter_graph);
    if (ret < 0) {
        av_log(NULL, AV_LOG_ERROR, "Cannot create buffer source\n");
        goto end;
```

--立志做一个好的程序员

4. Re:谷歌浏览器Chrome播放rtsp视频流解决方案

目前市面上已经有很成熟且商用Chrome播放海康威视大华的H.264或H.265的RTSP视频流解决方案了，就是猿大师中间件，底层调用VLC的ActiveX控件可实现在网页中内嵌播放多路RTSP的实时...

--喵大侠

5. Re:如何使用UDP进行跨网段广播

主机A：192.168.3.100 子网掩码255.255.0.0（手动临时修改）主机B：192.168.120.100 子网掩码255.255.255.0 主机A 广播192.168.255...

--zzhilling

```
    }

    /* buffer video sink: to terminate the filter chain. */
    ret = avfilter_graph_create_filter(&buffersink_ctx, buffersink, "out",
                                       NULL, NULL, filter_graph);
    if (ret < 0) {
        av_log(NULL, AV_LOG_ERROR, "Cannot create buffer sink\n");
        goto end;
    }

     /* Set a binary option to an integer list. */
    ret = av_opt_set_int_list(buffersink_ctx, "pix_fmts", pix_fmts,
                              AV_PIX_FMT_NONE, AV_OPT_SEARCH_CHILDREN);
    if (ret < 0) {
        av_log(NULL, AV_LOG_ERROR, "Cannot set output pixel format\n");
        goto end;
    }

    /*
     * Set the endpoints for the filter graph. The filter_graph will
     * be linked to the graph described by filters_descr.
     */

    /*
     * The buffer source output must be connected to the input pad of
     * the first filter described by filters_descr; since the first
     * filter input label is not specified, it is set to "in" by
     * default.
     */
    outputs->name       = av_strdup("in");
    outputs->filter_ctx = buffersrc_ctx;
    outputs->pad_idx    = 0;
    outputs->next       = NULL;

    /*
     * The buffer sink input must be connected to the output pad of
     * the last filter described by filters_descr; since the last
     * filter output label is not specified, it is set to "out" by
     * default.
     */
    inputs->name       = av_strdup("out");
    inputs->filter_ctx = buffersink_ctx;
    inputs->pad_idx    = 0;
    inputs->next       = NULL;

    /* Add a graph described by a string to a graph */
    if ((ret = avfilter_graph_parse_ptr(filter_graph, filters_descr,
                                        &inputs, &outputs, NULL)) < 0)
        goto end;

    /* Check validity and configure all the links and formats in the graph */
    if ((ret = avfilter_graph_config(filter_graph, NULL)) < 0)
        goto end;

end:
    avfilter_inout_free(&inputs);
    avfilter_inout_free(&outputs);

    return ret;
}

#ifndef SAVE_FILE
static void display_frame(const AVFrame *frame, AVRational time_base)
{
    int x, y;
    uint8_t *p0, *p;
    int64_t delay;

    if (frame->pts != AV_NOPTS_VALUE) {
        if (last_pts != AV_NOPTS_VALUE) {
            /* sleep roughly the right amount of time;
```

```
             * usleep is in microseconds, just like AV_TIME_BASE. */
             /* 计算 pts 是用来把时间戳从一个时基调整到另外一个时基时候用的函数 */
            delay = av_rescale_q(frame->pts - last_pts,
                                 time_base, AV_TIME_BASE_Q);
            if (delay > 0 && delay < 1000000)
                usleep(delay);
        }
        last_pts = frame->pts;
    }

    /* Trivial ASCII grayscale display. */
    p0 = frame->data[0];
    puts("\033c");
    for (y = 0; y < frame->height; y++) {
        p = p0;
        for (x = 0; x < frame->width; x++)
            putchar(" .-+#"[*(p++) / 52]);
        putchar('\n');
        p0 += frame->linesize[0];
    }
    fflush(stdout);
}
#else
FILEFILE * file_fd;
static void write_frame(const AVFrame *frame)
{
    static int printf_flag = 0;
    if(!printf_flag){
        printf_flag = 1;
        printf("frame widht=%d,frame height=%d\n",frame->width,frame->height);

        if(frame->format==AV_PIX_FMT_YUV420P){
            printf("format is yuv420p\n");
        }
        else{
            printf("formet is = %d \n",frame->format);
        }

    }

    fwrite(frame->data[0],1,frame->width*frame->height,file_fd);
    fwrite(frame->data[1],1,frame->width/2*frame->height/2,file_fd);
    fwrite(frame->data[2],1,frame->width/2*frame->height/2,file_fd);
}

#endif

int main(int argc, charchar **argv)
{
    int ret;
    AVPacket packet;
    AVFrame *frame = av_frame_alloc();
    AVFrame *filt_frame = av_frame_alloc();
    int got_frame;

#ifdef SAVE_FILE
    file_fd = fopen("test.yuv","wb+");
#endif

    if (!frame || !filt_frame) {
        perror("Could not allocate frame");
        exit(1);
    }
    if (argc != 2) {
        fprintf(stderr, "Usage: %s file\n", argv[0]);
        exit(1);
    }

    av_register_all();
    avfilter_register_all();

```

```
    if ((ret = open_input_file(argv[1])) < 0)
        goto end;
    if ((ret = init_filters(filter_descr)) < 0)
        goto end;

    /* read all packets */
    while (1) {
        if ((ret = av_read_frame(fmt_ctx, &packet)) < 0)
            break;

        if (packet.stream_index == video_stream_index) {
            got_frame = 0;
            ret = avcodec_decode_video2(dec_ctx, frame, &got_frame, &packet);
            if (ret < 0) {
                av_log(NULL, AV_LOG_ERROR, "Error decoding video\n");
                break;
            }

            if (got_frame) {
                frame->pts = av_frame_get_best_effort_timestamp(frame);    /* pts: Presentation Time Stamp */

                /* push the decoded frame into the filtergraph */
                if (av_buffersrc_add_frame_flags(buffersrc_ctx, frame, AV_BUFFERSRC_FLAG_KEEP_REF) < 0) {
                    av_log(NULL, AV_LOG_ERROR, "Error while feeding the filtergraph\n");
                    break;
                }

                /* pull filtered frames from the filtergraph */
                while (1) {
                    ret = av_buffersink_get_frame(buffersink_ctx, filt_frame);
                    if (ret == AVERROR(EAGAIN) || ret == AVERROR_EOF)
                        break;
                    if (ret < 0)
                        goto end;
#ifndef SAVE_FILE
                    display_frame(filt_frame, buffersink_ctx->inputs[0]->time_base);
#else
                    write_frame(filt_frame);
#endif
                    av_frame_unref(filt_frame);
                }
                /* Unreference all the buffers referenced by frame and reset the frame fields. */
                av_frame_unref(frame);
            }
        }
        av_packet_unref(&packet);
    }
end:
    avfilter_graph_free(&filter_graph);
    avcodec_close(dec_ctx);
    avformat_close_input(&fmt_ctx);
    av_frame_free(&frame);
    av_frame_free(&filt_frame);

    if (ret < 0 && ret != AVERROR_EOF) {
        fprintf(stderr, "Error occurred: %s\n", av_err2str(ret));
        exit(1);
    }
#ifdef SAVE_FILE
    fclose(file_fd);
#endif
    exit(0);
}
```

该工程中，我的Makefile文件如下：

```
OUT_APP      = test
INCLUDE_PATH = /usr/local/include/
```

```
INCLUDE = -I$(INCLUDE_PATH)libavutil/ -I$(INCLUDE_PATH)libavdevice/ \
          -I$(INCLUDE_PATH)libavcodec/ -I$(INCLUDE_PATH)libswresample \
          -I$(INCLUDE_PATH)libavfilter/ -I$(INCLUDE_PATH)libavformat \
          -I$(INCLUDE_PATH)libswscale/

FFMPEG_LIBS = -lavformat -lavutil -lavdevice -lavcodec -lswresample -lavfilter -lswscale
SDL_LIBS    =
LIBS        = $(FFMPEG_LIBS)$(SDL_LIBS)

COMPILE_OPTS = $(INCLUDE)
C            = c
OBJ          = o
C_COMPILER   = cc
C_FLAGS      = $(COMPILE_OPTS) $(CPPFLAGS) $(CFLAGS)

LINK         = cc -o
LINK_OPTS    = -lz -lm  -lpthread
LINK_OBJ     = test.o

.$(C).$(OBJ):
    $(C_COMPILER) -c $(C_FLAGS) $<


$(OUT_APP): $(LINK_OBJ)
    $(LINK)$@  $(LINK_OBJ)  $(LIBS) $(LINK_OPTS)

clean:
        -rm -rf *.$(OBJ) $(OUT_APP) core *.core *~ *yuv
```

运行结果如下：

```
licaibiao@ubuntu:~/test/FFMPEG/filter$ ls
Makefile  school.flv  test  test.c  test.o
licaibiao@ubuntu:~/test/FFMPEG/filter$ ./test school.flv
[flv @ 0x12c16c0] video stream discovered after head already parsed
[flv @ 0x12c16c0] audio stream discovered after head already parsed
frame widht=1024,frame height=576
format is yuv420p
licaibiao@ubuntu:~/test/FFMPEG/filter$ ls
Makefile  school.flv  test  test.c  test.o  test.yuv
```

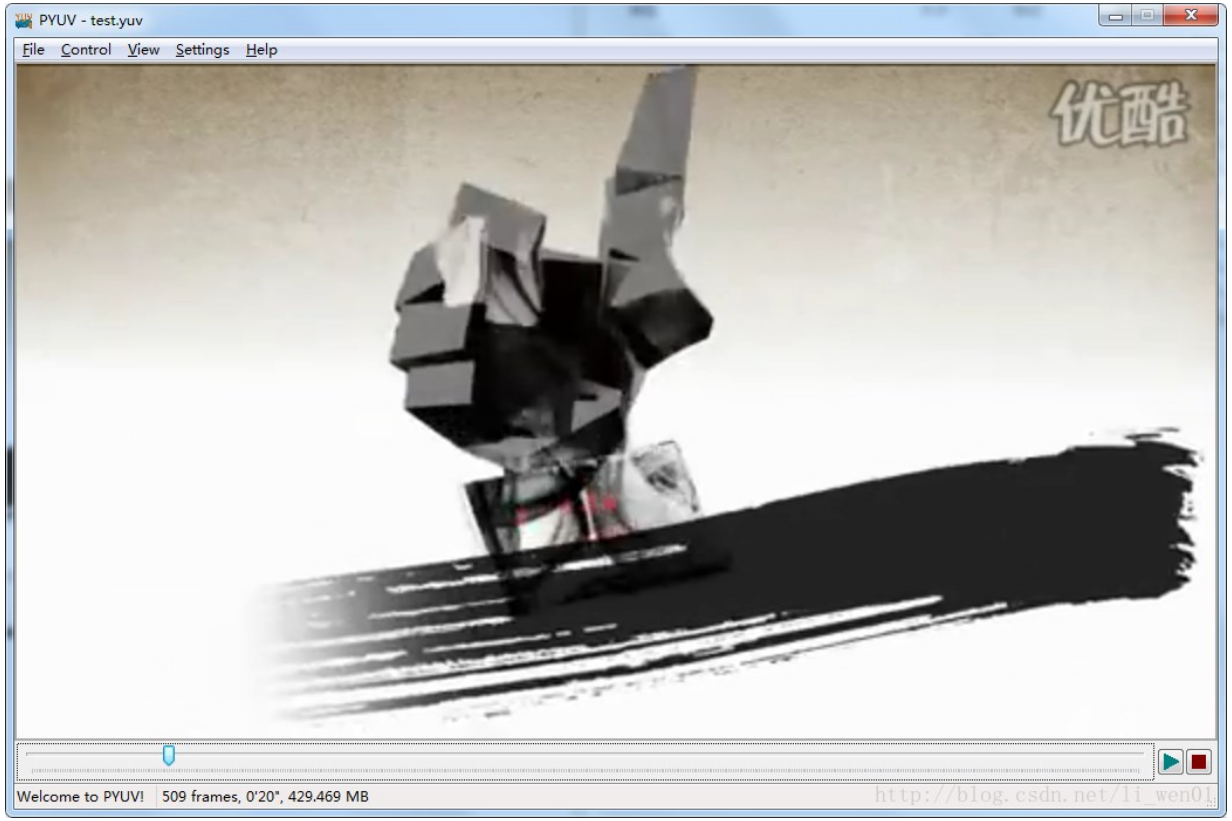
在这里，我打印出来了输出视频的格式和图片的长和宽，该实例生成的是一个YUV420 格式的视频，使用YUV播放器播放视频的时候，需要设置正确的视频长度和宽度。在代码中通过设置enum AVPixelFormat pix_fmts[] = { AV_PIX_FMT_YUV420P, AV_PIX_FMT_NONE };来设置输出格式。

过滤器的参数设置是通过const char *filter_descr = "scale=iw*2:ih*2"; 来设置。它表示将视频的长和框都拉伸到原来的两倍。具体的filter参数可以通过命令：ffmpeg -filters 来查询。结果如下：

```
Filters:
  T.. = Timeline support
  .S. = Slice threading
  ..C = Command support
  A = Audio input/output
  V = Video input/output
  N = Dynamic number and/or type of input/output
  | = Source or sink filter
 ... abench            A->A       Benchmark part of a filtergraph.
 ... acompressor       A->A       Audio compressor.
 ... acrossfade        AA->A      Cross fade two input audio streams.
 ... acrusher          A->A       Reduce audio bit resolution.
 ... ...................................................................
```

在上面的代码中，我们设置的是将图片拉升到原来图像的两倍，其显示效果如下，可能是截图的问题，这里看好像没有拉伸到两倍。

原图



拉伸后

在上面的代码中，我们设置的是：

const char *filter_descr = "scale=iw*2:ih*2"; iw 表示输入视频的宽，ih表示输入视频的高。可以任意比例的缩放视频。这里*2 表示放大两倍,如果是/2表示缩小两倍。

视频缩放还可以直接设置：

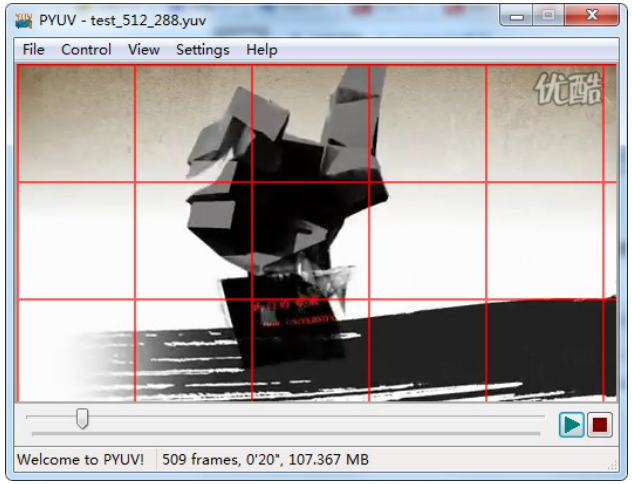
const char *filter_descr = "scale=320:240"; 设置视频输出宽为320，高位240，当然也是可以随意的设置其他的参数。

视频的裁剪可以设置为：

const char *filter_descr = "crop=320:240:0:0"; 具体含义是 crop=width:height:x:y，其中 width 和 height 表示裁剪后的尺寸，x:y 表示裁剪区域的左上角坐标。

视频添加一个网格水印可以设置为：

const char *filter_descr = "drawgrid=width=100:height=100:thickness=2:color=red@0.5"; 具体含义是 width 和 height 表示添加网格的宽和高，thickness表示网格的线宽，color表示颜色 。其效果如下：



更多filter参数的使用，可以直接参考ffmpeg 的官方文档：http://www.ffmpeg.org/ffmpeg-filters.html

分类: ffmpeg、ffplay

DoubleLi
关注 - 29
粉丝 - 2080
+加关注

« 上一篇： FFmpeg 基本用法
» 下一篇： FFMPEG (一) 从V4L2捕获摄像头数据

posted on 2017-08-11 14:31 DoubleLi 阅读(1412) 评论(1) 编辑 收藏 举报

· 以终为始：如何让你的开发符合预期

最新新闻：
· 何小鹏：争取2024年实现飞行汽车量产 价格100万以内（2021-10-24 23:35）
· 供应链危机提振美国在线二手市场 全年销售额预计超650亿美元（2021-10-24 22:00）
· CityTree：一款利用苔藓和机器学习来捕捉空气污染的设备（2021-10-24 20:53）
· 1024程序员节各家怎么过：送霸王洗发水、集体穿格子衫、盲人按摩（2021-10-24 20:00）
· 新卫星图展示泰国季风洪水所带来的巨大影响（2021-10-24 19:00）
» 更多新闻...