原 最简单的基于FFmpeg的移动端例子：IOS 视频解码器

2015年07月28日 19:02:45　阅读数：26020

==========

最简单的基于FFmpeg的移动端例子系列文章列表：

最简单的基于FFmpeg的移动端例子：Android HelloWorld

最简单的基于FFmpeg的移动端例子：Android 视频解码器

最简单的基于FFmpeg的移动端例子：Android 视频解码器-单个库版

最简单的基于FFmpeg的移动端例子：Android 推流器

最简单的基于FFmpeg的移动端例子：Android 视频转码器

最简单的基于FFmpeg的移动端例子附件：Android 自带播放器

最简单的基于FFmpeg的移动端例子附件：SDL Android HelloWorld

最简单的基于FFmpeg的移动端例子：IOS HelloWorld

最简单的基于FFmpeg的移动端例子：IOS 视频解码器

最简单的基于FFmpeg的移动端例子：IOS 推流器

最简单的基于FFmpeg的移动端例子：IOS 视频转码器

最简单的基于FFmpeg的移动端例子附件：IOS自带播放器

最简单的基于FFmpeg的移动端例子：Windows Phone HelloWorld

==========

本文记录IOS平台下基于FFmpeg的视频解码器。该示例C语言的源代码来自于《 最简单的基于FFMPEG+SDL的视频播放器 》。相关的概念就不再重复记录了。

源代码

项目的目录结构如图所示。

C代码位于ViewController.m文件中，内容如下所示。

[cpp]

```cpp
/**
 * 最简单的基于FFmpeg的视频解码器-IOS
 * Simplest FFmpeg IOS Decoder
 *
 * 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
 *
 * 本程序是IOS平台下最简单的基于FFmpeg的视频解码器。
 * 它可以将输入的视频数据解码成YUV像素数据。
 *
 * This software is the simplest decoder based on FFmpeg in IOS.
 * It can decode video stream to raw YUV data.
 *
 */

#import "ViewController.h"
#include <libavcodec/avcodec.h>
#include <libavformat/avformat.h>
#include <libavutil/imgutils.h>
#include <libswscale/swscale.h>

@interface ViewController ()
```

```
26.
27. @end
28.
29. @implementation ViewController
30.
31. - (void)viewDidLoad {
32.     [super viewDidLoad];
33.     // Do any additional setup after loading the view, typically from a nib.
34. }
35.
36. - (void)didReceiveMemoryWarning {
37.     [super didReceiveMemoryWarning];
38.     // Dispose of any resources that can be recreated.
39. }
40.
41. - (IBAction)clickDecodeButton:(id)sender {
42.     AVFormatContext *pFormatCtx;
43.     int             i, videoindex;
44.     AVCodecContext  *pCodecCtx;
45.     AVCodec         *pCodec;
46.     AVFrame *pFrame,*pFrameYUV;
47.     uint8_t *out_buffer;
48.     AVPacket *packet;
49.     int y_size;
50.     int ret, got_picture;
51.     struct SwsContext *img_convert_ctx;
52.     FILE *fp_yuv;
53.     int frame_cnt;
54.     clock_t time_start, time_finish;
55.     double  time_duration = 0.0;
56.
57.     char input_str_full[500]={0};
58.     char output_str_full[500]={0};
59.     char info[1000]={0};
60.
61.     NSString *input_str= [NSString stringWithFormat:@"resource.bundle/%@",self.inputurl.text];
62.     NSString *output_str= [NSString stringWithFormat:@"resource.bundle/%@",self.outputurl.text];
63.
64.     NSString *input_nsstr=[[[NSBundle mainBundle]resourcePath] stringByAppendingPathComponent:input_str];
65.     NSString *output_nsstr=[[[NSBundle mainBundle]resourcePath] stringByAppendingPathComponent:output_str];
66.
67.     sprintf(input_str_full,"%s",[input_nsstr UTF8String]);
68.     sprintf(output_str_full,"%s",[output_nsstr UTF8String]);
69.
70.     printf("Input Path:%s\n",input_str_full);
71.     printf("Output Path:%s\n",output_str_full);
72.
73.     av_register_all();
74.     avformat_network_init();
75.     pFormatCtx = avformat_alloc_context();
76.
77.     if(avformat_open_input(&pFormatCtx,input_str_full,NULL,NULL)!=0){
78.         printf("Couldn't open input stream.\n");
79.         return ;
80.     }
81.     if(avformat_find_stream_info(pFormatCtx,NULL)<0){
82.         printf("Couldn't find stream information.\n");
83.         return;
84.     }
85.     videoindex=-1;
86.     for(i=0; i<pFormatCtx->nb_streams; i++)
87.         if(pFormatCtx->streams[i]->codec->codec_type==AVMEDIA_TYPE_VIDEO){
88.             videoindex=i;
89.             break;
90.         }
91.     if(videoindex==-1){
92.         printf("Couldn't find a video stream.\n");
93.         return;
94.     }
95.     pCodecCtx=pFormatCtx->streams[videoindex]->codec;
96.     pCodec=avcodec_find_decoder(pCodecCtx->codec_id);
97.     if(pCodec==NULL){
98.         printf("Couldn't find Codec.\n");
99.         return;
100.    }
101.    if(avcodec_open2(pCodecCtx, pCodec,NULL)<0){
102.        printf("Couldn't open codec.\n");
103.        return;
104.    }
105.
106.    pFrame=av_frame_alloc();
107.    pFrameYUV=av_frame_alloc();
108.    out_buffer=(unsigned char *)av_malloc(av_image_get_buffer_size(AV_PIX_FMT_YUV420P,  pCodecCtx->width, pCodecCtx->height,1));
109.    av_image_fill_arrays(pFrameYUV->data, pFrameYUV->linesize,out_buffer,
110.                         AV_PIX_FMT_YUV420P,pCodecCtx->width, pCodecCtx->height,1);
111.    packet=(AVPacket *)av_malloc(sizeof(AVPacket));
112.
113.    img_convert_ctx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt,
114.                                     pCodecCtx->width, pCodecCtx->height, AV_PIX_FMT_YUV420P, SWS_BICUBIC, NULL, NULL, NULL);
115.
116.
```

```
    sprintf(info,   "[Input     ]%s\n", [input_str UTF8String]);
    sprintf(info, "%s[Output    ]%s\n",info,[output_str UTF8String]);
    sprintf(info, "%s[Format    ]%s\n",info, pFormatCtx->iformat->name);
    sprintf(info, "%s[Codec     ]%s\n",info, pCodecCtx->codec->name);
    sprintf(info, "%s[Resolution]%dx%d\n",info, pCodecCtx->width,pCodecCtx->height);


    fp_yuv=fopen(output_str_full,"wb+");
    if(fp_yuv==NULL){
        printf("Cannot open output file.\n");
        return;
    }

    frame_cnt=0;
    time_start = clock();

    while(av_read_frame(pFormatCtx, packet)>=0){
        if(packet->stream_index==videoindex){
            ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
            if(ret < 0){
                printf("Decode Error.\n");
                return;
            }
            if(got_picture){
                sws_scale(img_convert_ctx, (const uint8_t* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,
                          pFrameYUV->data, pFrameYUV->linesize);

                y_size=pCodecCtx->width*pCodecCtx->height;
                fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
                fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
                fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
                //Output info
                char pictype_str[10]={0};
                switch(pFrame->pict_type){
                    case AV_PICTURE_TYPE_I:sprintf(pictype_str,"I");break;
                    case AV_PICTURE_TYPE_P:sprintf(pictype_str,"P");break;
                    case AV_PICTURE_TYPE_B:sprintf(pictype_str,"B");break;
                    default:sprintf(pictype_str,"Other");break;
                }
                printf("Frame Index: %5d. Type:%s\n",frame_cnt,pictype_str);
                frame_cnt++;
            }
        }
        av_free_packet(packet);
    }
    //flush decoder
    //FIX: Flush Frames remained in Codec
    while (1) {
        ret = avcodec_decode_video2(pCodecCtx, pFrame, &got_picture, packet);
        if (ret < 0)
            break;
        if (!got_picture)
            break;
        sws_scale(img_convert_ctx, (const uint8_t* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height,
                  pFrameYUV->data, pFrameYUV->linesize);
        int y_size=pCodecCtx->width*pCodecCtx->height;
        fwrite(pFrameYUV->data[0],1,y_size,fp_yuv);    //Y
        fwrite(pFrameYUV->data[1],1,y_size/4,fp_yuv);  //U
        fwrite(pFrameYUV->data[2],1,y_size/4,fp_yuv);  //V
        //Output info
        char pictype_str[10]={0};
        switch(pFrame->pict_type){
            case AV_PICTURE_TYPE_I:sprintf(pictype_str,"I");break;
            case AV_PICTURE_TYPE_P:sprintf(pictype_str,"P");break;
            case AV_PICTURE_TYPE_B:sprintf(pictype_str,"B");break;
            default:sprintf(pictype_str,"Other");break;
        }
        printf("Frame Index: %5d. Type:%s\n",frame_cnt,pictype_str);
        frame_cnt++;
    }
    time_finish = clock();
    time_duration=(double)(time_finish - time_start);

    sprintf(info, "%s[Time      ]%fus\n",info,time_duration);
    sprintf(info, "%s[Count     ]%d\n",info,frame_cnt);

    sws_freeContext(img_convert_ctx);

    fclose(fp_yuv);

    av_frame_free(&pFrameYUV);
    av_frame_free(&pFrame);
    avcodec_close(pCodecCtx);
    avformat_close_input(&pFormatCtx);

    NSString * info_ns = [NSString stringWithFormat:@"%s", info];
    self.infomation.text=info_ns;

}
```

```
208. @end
```

运行结果

App在手机上运行后的结果如下图所示。单击“Decode”，将会把位于resource.bundle中的“sintel.mov”文件解码为“sintel.yuv”文件并存储于相同的目录下。

生成的文件如下图所示。

下载

simplest ffmpeg mobile

项目主页

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

开源中国： https://git.oschina.net/leixiaohua1020/simplest_ffmpeg_mobile

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

CSDN工程下载地址： http://download.csdn.net/detail/leixiaohua1020/8924391

本解决方案包含了使用FFmpeg在移动端处理多媒体的各种例子：

[Android]
simplest_android_player: 基于安卓接口的视频播放器
simplest_ffmpeg_android_helloworld: 安卓平台下基于FFmpeg的HelloWorld程序
simplest_ffmpeg_android_decoder: 安卓平台下最简单的基于FFmpeg的视频解码器
simplest_ffmpeg_android_decoder_onelib: 安卓平台下最简单的基于FFmpeg的视频解码器-单库版
simplest_ffmpeg_android_streamer: 安卓平台下最简单的基于FFmpeg的推流器
simplest_ffmpeg_android_transcoder: 安卓平台下移植的FFmpeg命令行工具
simplest_sdl_android_helloworld: 移植SDL到安卓平台的最简单程序
[IOS]
simplest_ios_player: 基于IOS接口的视频播放器
simplest_ffmpeg_ios_helloworld: IOS平台下基于FFmpeg的HelloWorld程序
simplest_ffmpeg_ios_decoder: IOS平台下最简单的基于FFmpeg的视频解码器
simplest_ffmpeg_ios_streamer: IOS平台下最简单的基于FFmpeg的推流器
simplest_ffmpeg_ios_transcoder: IOS平台下移植的ffmpeg.c命令行工具
simplest_sdl_ios_helloworld: 移植SDL到IOS平台的最简单程序

文章标签： FFmpeg IOS 视频 YUV
个人分类： FFMPEG IOS多媒体
所属专栏： FFmpeg