个人资料

scalerzhangjie

访问： 5540次
积分： 205分
排名： 千里之外

原创： 12篇 转载： 0篇
译文： 0篇 评论： 18条

文章搜索

文章分类

Linux (3)
C (7)
C_Language_Programming_Learn

文章存档

2013年11月 (1)
2013年10月 (2)
2013年09月 (2)
2013年08月 (2)
2012年12月 (3)

展开

阅读排行

视频编解码---x264用于编 (2440)
微软2014笔试题21题解答 (1045)
有序双链表的插入问题 (223)
select在socket中的使用 (213)
通过Framebuffer刷屏使 (206)
找工作之面试题练习2 (183)
完美洗牌算法的多种实现 (179)
关于迭代和递归的一个小 (169)
找工作之面试题练习1 (164)
找工作之"整数转化为字符 (146)

评论排行

视频编解码---x264用于编码，ffmpeg用于解码

分类： Linux 2012-12-08 20:53 2440人阅读 评论(14) 收藏 举报

codec x264 ffmpeg

项目要用到视频编解码，最近半个月都在搞，说实话真是走了很多弯路，浪费了很多时间。将自己的最终成果记录于此，期望会给其他人提供些许帮助。

参考教程：

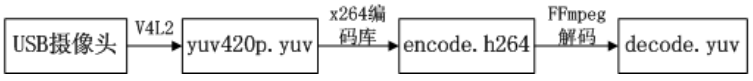
http://ffmpeg.org/trac/ffmpeg/wiki/UbuntuCompilationGuide安装ffmpeg和x264，官方权威教程（注意不要用命令行安装，会少很多库的。编译安装最保险）

http://blog.csdn.net/zgyulongfei/article/details/7526249采集与编码的教程

http://www.cnblogs.com/fojian/archive/2012/09/01/2666627.html编码的好文章

http://my.oschina.net/u/555701/blog/56616?p=2#comments-解码的好文章

整体过程流程如下：



显而易见，整个过程分为三个部分：采集、编码、解码。

1. 采集视频

我是利用USB摄像头采集视频的，我的摄像头只支持YUV422格式的图像采集，因为x264编码库只能编码YUV420P(planar)格式，因此在采集到yuv422格式的图像数据后要变换成yuv420p格式。

采集视频使用官方的那个采集程序，稍加修改即可，具体点说就是修改

static void process_image (const char * p) ;函数

参数p指向一帧采集图像的yuv数据。

关于YUV格式和RGB格式，网上有很多教程。

在这儿，我讲一下自己的理解。

假设有一幅4*4分辨率的图片，如下：

1	2	3	4
5	6	7	8
9	10	11	12

视频编解码---x264用于编 (14)
微软2014笔试题21题解答 (3)
找工作之面试题练习2 (1)
有序双链表的插入问题 (0)
完美洗牌算法的多种实现 (0)
找工作之面试题练习1 (0)
找工作之“整数转化为字符 (0)
找工作之“二叉搜索树” (0)
通过Framebuffer刷屏使得 (0)
select在socket中的使用方 (0)

推荐文章

最新评论

视频编解码---x264用于编码，ffm
sunnyrainab: @scalerzhangjie: 谢谢啦，我已经搞定了，呵呵

视频编解码---x264用于编码，ffm
scalerzhangjie: @bg2bkk:厉害！有空我试试。谢谢分享。

视频编解码---x264用于编码，ffm
scalerzhangjie: @smilestone322:谢谢喽

视频编解码---x264用于编码，ffm
scalerzhangjie: @smilestone322:谢谢喽

视频编解码---x264用于编码，ffm
scalerzhangjie: @u012767067:不好意思，这么晚才回复，最近一直在忙，没怎么上网。我觉得pyuv播放器你没设...

视频编解码---x264用于编码，ffm
sunnyrainab: http://download.csdn.net/detail/scal

微软2014笔试题21题解答（有点
垚山行者: O(1)的思路是头插法，数据结构上的生成 链表的一个方法

微软2014笔试题21题解答（有点
swording路人: @Weirenren_027:先把后一半的元素逆序，就是将指针反转为指向前一个元素。然后把中间那个元...

微软2014笔试题21题解答（有点
Weirenren_027: 思路是怎样的啊 博主能不能描述下

找工作之面试题练习2
scalerzhangjie: 前两种是递归实现，第三种是迭代实现。迭代实现快很多！！！

13	14	15	16

每个像素是由YUV数据构成，假设如下：

Y1	U1	V1	Y2	U2	V2	Y3	U3	V3	Y4	U4	V4
Y5	U5	V5	Y6	U6	V6	Y7	U7	V7	Y8	U8	V8
Y9	U9	V9	Y10	U10	V10	Y11	U11	V11	Y12	U12	V12
Y13	U13	V13	Y14	U14	V14	Y15	U15	V15	Y16	U16	V16

YUV422图像是这样的，每个像素采集Y，UV每隔两个像素采集一次：

Y1	U1		Y2		V2	Y3	U3		Y4		V4
Y5	U5		Y6		V6	Y7	U7		Y8		V8
Y9	U9		Y10		V10	Y11	U11		Y12		V12
Y13	U13		Y14		V14	Y15	U15		Y16		V16

Packed格式的YUV420是这样的，每个像素采集Y，UV隔行采集，每行是每两个像素采集一次：

Y1	U1		Y2			Y3	U3		Y4		
Y5			Y6		V6	Y7			Y8		V8
Y9	U9		Y10			Y11	U11		Y12		
Y13			Y14		V14	Y15			Y16		V16

以上几种格式存储就是按照从左到右，从上到下顺序存储的。

我想要得到是planar格式的YUV420，即在一段连续的内存中，先存储所有的Y，接着是所有的U，最后是所有的V。

修改后的 process_image函数如下：

```cpp
static void
process_image                   (const char *           p)
{
        //fputc ('.', stdout);
    //convert yuv422 to yuv420p
            char *y=yuv420p;
            char *u=&yuv420p[IMAGE_WIDTH*IMAGE_HEIGHT];
            char *v=&yuv420p[IMAGE_WIDTH*IMAGE_HEIGHT+IMAGE_WIDTH*IMAGE_HEIGHT/4];

            int i=0,j=0,l=0;
            for(j=0;j<IMAGE_HEIGHT;j++)
                for(i=0;i<IMAGE_WIDTH*2;i++,l++){

                    if(j%2==0){//even line to sample U-Chriminance
                        if(l==1){//sample U-Chriminance
                            *u=p[j*IMAGE_WIDTH*2+i];
                            u++;
                         }
                         else if(l==3){//abandon V-Chroma
                            l=-1;
                            continue;

                         }
                         else{
                            *y=p[j*IMAGE_WIDTH*2+i];
                            ++y;
                         }
                    }

                    else if(j%2==1){//odd lines to sample  V-Chroma
```

```
                        if(l==1){
                            continue;
                        }
                        else if(l==3){
                            l=-1;
                            *v=p[j*IMAGE_WIDTH*2+i];
                            ++v;
                        }
                        else {
                            *y=p[j*IMAGE_WIDTH*2+i];
                            ++y;
                        }

                    }

                }

            fwrite(yuv420p,IMAGE_WIDTH*IMAGE_HEIGHT*3>>1,1,fp_yuv420p);

            fflush (stdout);

}
```

2.编码

采用x264编码库编码yuv420p文件。

程序如下：

```cpp
#include <stdint.h>
#include <x264.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>

#define DEBUG 0

#define CLEAR(x) (memset((&x),0,sizeof(x)))
#define IMAGE_WIDTH   320
#define IMAGE_HEIGHT  240
#define ENCODER_PRESET "veryfast"
#define ENCODER_TUNE   "zerolatency"
#define ENCODER_PROFILE  "baseline"
#define ENCODER_COLORSPACE X264_CSP_I420

typedef struct my_x264_encoder{
    x264_param_t  * x264_parameter;
    char parameter_preset[20];
    char parameter_tune[20];
    char parameter_profile[20];
    x264_t  * x264_encoder;
    x264_picture_t * yuv420p_picture;
    long colorspace;
    unsigned char *yuv;
    x264_nal_t * nal;
} my_x264_encoder;

char *read_filename="yuv420p.yuv";
char *write_filename="encode.h264";

int
main(int argc ,char **argv){
    int ret;
    int fd_read,fd_write;
    my_x264_encoder * encoder=(my_x264_encoder *)malloc(sizeof(my_x264_encoder));
    if(!encoder){
        printf("cannot malloc my_x264_encoder !\n");
        exit(EXIT_FAILURE);
    }
    CLEAR(*encoder);


    /*****************************************************************************
     * Advanced parameter handling functions
```

```
     *****************************************************************************/

    /* These functions expose the full power of x264's preset-tune-profile system for
     * easy adjustment of large numbers //free(encoder->yuv420p_picture);of internal parameters.
     *
     * In order to replicate x264CLI's option handling, these functions MUST be called
     * in the following order:
     * 1) x264_param_default_preset
     * 2) Custom user options (via param_parse or directly assigned variables)
     * 3) x264_param_apply_fastfirstpass
     * 4) x264_param_apply_profile
     *
     * Additionally, x264CLI does not apply step 3 if the preset chosen is "placebo"
     * or --slow-firstpass is set. */
    strcpy(encoder->parameter_preset,ENCODER_PRESET);
    strcpy(encoder->parameter_tune,ENCODER_TUNE);

    encoder->x264_parameter=(x264_param_t *)malloc(sizeof(x264_param_t));
    if(!encoder->x264_parameter){
        printf("malloc x264_parameter error!\n");
        exit(EXIT_FAILURE);
    }
    CLEAR(*(encoder->x264_parameter));
    x264_param_default(encoder->x264_parameter);

    if((ret=x264_param_default_preset(encoder->x264_parameter,encoder->parameter_preset,encoder->parameter_tune))<0){
        printf("x264_param_default_preset error!\n");
        exit(EXIT_FAILURE);
    }

    encoder->x264_parameter->i_fps_den         =1;
    encoder->x264_parameter->i_fps_num         =25;
    encoder->x264_parameter->i_width       =IMAGE_WIDTH;
    encoder->x264_parameter->i_height      =IMAGE_HEIGHT;
    encoder->x264_parameter->i_threads         =1;
    encoder->x264_parameter->i_keyint_max    =25;
    encoder->x264_parameter->b_intra_refresh =1;
    encoder->x264_parameter->b_annexb      =1;

    strcpy(encoder->parameter_profile,ENCODER_PROFILE);
    if((ret=x264_param_apply_profile(encoder->x264_parameter,encoder->parameter_profile))<0){
        printf("x264_param_apply_profile error!\n");
        exit(EXIT_FAILURE);
    }

#if DEBUG
    printf("Line --------%d\n",__LINE__);
#endif

    encoder->x264_encoder=x264_encoder_open(encoder->x264_parameter);

    encoder->colorspace=ENCODER_COLORSPACE;

#if DEBUG
    printf("Line --------%d\n",__LINE__);
#endif

    encoder->yuv420p_picture=(x264_picture_t *)malloc(sizeof(x264_picture_t ));
    if(!encoder->yuv420p_picture){
        printf("malloc encoder->yuv420p_picture error!\n");
        exit(EXIT_FAILURE);
    }
    if((ret=x264_picture_alloc(encoder->yuv420p_picture,encoder->colorspace,IMAGE_WIDTH,IMAGE_HEIGHT))<0){
        printf("ret=%d\n",ret);
        printf("x264_picture_alloc error!\n");
        exit(EXIT_FAILURE);
    }

    encoder->yuv420p_picture->img.i_csp=encoder->colorspace;
    encoder->yuv420p_picture->img.i_plane=3;
    encoder->yuv420p_picture->i_type=X264_TYPE_AUTO;



#if DEBUG
```

```
    printf("Line --------%d\n",__LINE__);
#endif

    encoder->yuv=(uint8_t *)malloc(IMAGE_WIDTH*IMAGE_HEIGHT*3/2);
    if(!encoder->yuv){
        printf("malloc yuv error!\n");
        exit(EXIT_FAILURE);
    }
    CLEAR(*(encoder->yuv));

#if DEBUG
    printf("Line --------%d\n",__LINE__);
#endif

    encoder->yuv420p_picture->img.plane[0]=encoder->yuv;
    encoder->yuv420p_picture->img.plane[1]=encoder->yuv+IMAGE_WIDTH*IMAGE_HEIGHT;
    encoder->yuv420p_picture->img.plane[2]=encoder->yuv+IMAGE_WIDTH*IMAGE_HEIGHT+IMAGE_WIDTH*IMAGE_HEIGHT/4;

    if((fd_read=open(read_filename,O_RDONLY))<0){
        printf("cannot open input file!\n");
        exit(EXIT_FAILURE);
    }

    if((fd_write=open(write_filename,O_WRONLY | O_APPEND | O_CREAT,0777))<0){
        printf("cannot open output file!\n");
        exit(EXIT_FAILURE);
    }

#if DEBUG
    printf("Line --------%d\n",__LINE__);
#endif
    int n_nal;
    x264_picture_t pic_out;
    x264_nal_t *my_nal;
    encoder->nal=(x264_nal_t *)malloc(sizeof(x264_nal_t ));
    if(!encoder->nal){
        printf("malloc x264_nal_t error!\n");
        exit(EXIT_FAILURE);
    }
    CLEAR(*(encoder->nal));

    while(read(fd_read,encoder->yuv,IMAGE_WIDTH*IMAGE_HEIGHT*3/2)>0){
        encoder->yuv420p_picture->i_pts++;
        if((ret=x264_encoder_encode(encoder->x264_encoder,&encoder->nal,&n_nal,encoder->yuv420p_picture,&pic_out))<0){
            printf("x264_encoder_encode error!\n");
            exit(EXIT_FAILURE);
        }

        unsigned int length=0;
        for(my_nal=encoder->nal;my_nal<encoder->nal+n_nal;++my_nal){
            write(fd_write,my_nal->p_payload,my_nal->i_payload);
            length+=my_nal->i_payload;
        }
        printf("length=%d\n",length);
    }

    /*clean_up functions*/
    //x264_picture_clean(encoder->yuv420p_picture);
    //free(encoder->nal);//???? confused conflict with x264_encoder_close(encoder->x264_encoder);

    free(encoder->yuv);
    free(encoder->yuv420p_picture);
    free(encoder->x264_parameter);
    x264_encoder_close(encoder->x264_encoder);
    free(encoder);
    close(fd_read);
    close(fd_write);

    return 0;
}
```

3. 解码

利用ffmpeg进行解码

程序如下：

[cpp]

```cpp
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <libavcodec/avcodec.h>
#include <libavformat/avformat.h>
#include <libavutil/mathematics.h>

#define DECODED_OUTPUT_FORMAT  AV_PIX_FMT_YUV420P
#define INPUT_FILE_NAME "encode.h264"
#define OUTPUT_FILE_NAME "decode.yuv"
#define IMAGE_WIDTH  320
#define IMAGE_HEIGHT 240

void
error_handle(const char *errorInfo ){
    printf("%s error!\n",errorInfo);
    exit(EXIT_FAILURE);
}


int
main(int argc,char ** argv){
    int  write_fd,ret,videoStream;
    AVFormatContext * formatContext=NULL;
    AVCodec * codec;
    AVCodecContext * codecContext;
    AVFrame * decodedFrame;
    AVPacket packet;
    uint8_t *decodedBuffer;
    unsigned int decodedBufferSize;
    int finishedFrame;


    av_register_all();


    write_fd=open(OUTPUT_FILE_NAME,O_RDWR | O_CREAT,0666);
    if(write_fd<0){
        perror("open");
        exit(1);
    }

    ret=avformat_open_input(&formatContext, INPUT_FILE_NAME, NULL,NULL);
    if(ret<0)
        error_handle("avformat_open_input error");

    ret=avformat_find_stream_info(formatContext,NULL);
    if(ret<0)
        error_handle("av_find_stream_info");


    videoStream=0;
    codecContext=formatContext->streams[videoStream]->codec;

    codec=avcodec_find_decoder(AV_CODEC_ID_H264);
    if(codec==NULL)
        error_handle("avcodec_find_decoder error!\n");

    ret=avcodec_open2(codecContext,codec,NULL);
    if(ret<0)
        error_handle("avcodec_open2");

    decodedFrame=avcodec_alloc_frame();
    if(!decodedFrame)
        error_handle("avcodec_alloc_frame!");

    decodedBufferSize=avpicture_get_size(DECODED_OUTPUT_FORMAT,IMAGE_WIDTH,IMAGE_HEIGHT);
    decodedBuffer=(uint8_t *)malloc(decodedBufferSize);
    if(!decodedBuffer)
        error_handle("malloc decodedBuffer error!");
```

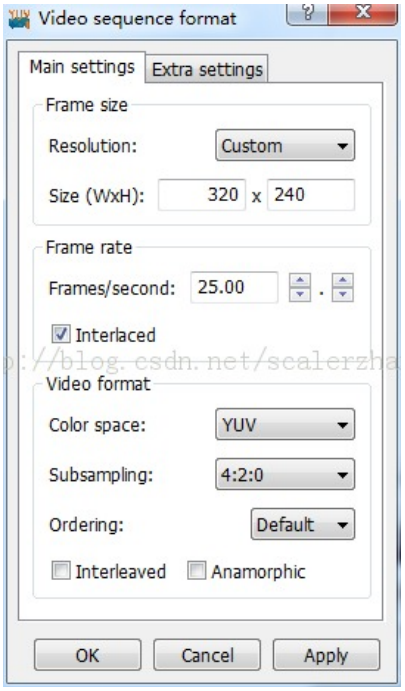
```
73.
74.     av_init_packet(&packet);
75.     while(av_read_frame(formatContext,&packet)>=0){
76.             ret=avcodec_decode_video2(codecContext,decodedFrame,&finishedFrame,&packet);
77.             if(ret<0)
78.                 error_handle("avcodec_decode_video2 error!");
79.             if(finishedFrame){
80.                 avpicture_layout((AVPicture*)decodedFrame,DECODED_OUTPUT_FORMAT,IMAGE_WIDTH,
81.                 ret=write(write_fd,decodedBuffer,decodedBufferSize);
82.                 if(ret<0)
83.                     error_handle("write yuv stream error!");
84.             }
85.
86.         av_free_packet(&packet);
87.     }
88.
89.     while(1){
90.         packet.data=NULL;
91.         packet.size=0;
92.         ret=avcodec_decode_video2(codecContext,decodedFrame,&finishedFrame,&packet);
93.         if(ret<=0 && (finishedFrame<=0))
94.             break;
95.         if(finishedFrame){
96.             avpicture_layout((AVPicture*)decodedFrame,DECODED_OUTPUT_FORMAT,IMAGE_WIDTH,IMAG
97.             ret=write(write_fd,decodedBuffer,decodedBufferSize);
98.             if(ret<0)
99.                 error_handle("write yuv stream error!");
100.         }
101.
102.         av_free_packet(&packet);
103.     }
104.
105.
106.     avformat_close_input(&formatContext);
107.     free(decodedBuffer);
108.     av_free(decodedFrame);
109.     avcodec_close(codecContext);
110.
111.     return 0;
112. }
```

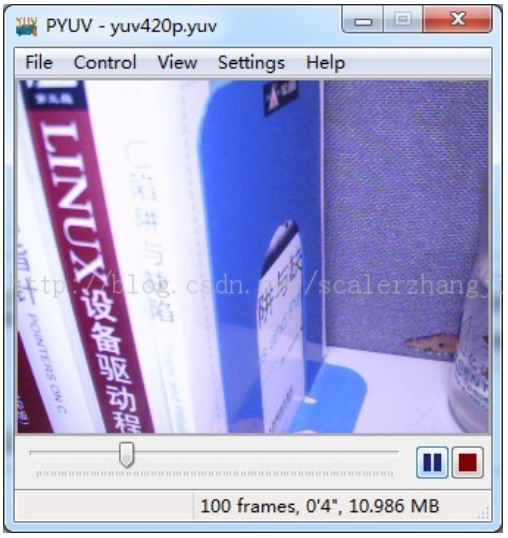
结果：

1. 利用USB摄像头采集的YUV420P，大小11.0MB，可以用pyuv播放器正常播放。
2. 编码后的文件encode.h264，大小262.4kb，可用vlc播放器正常播放。
3. 解码后的文件decode.yuv，大小11.0MB，可以用pyuv播放器正常播放。

相关文件在我的资源里，里面包含：

1. 采集、编码、解码程序、对应的可执行程序和Makefile文件；
2. Pyuv播放器（用于XP）
3. 实验文件-yuv420p.yuv 、encode.h264、 decode.yuv
4. 相关参考文档pdf版本





欢迎批评指正！

更多 0

上一篇 关于迭代和递归的一个小问题，求解答

下一篇 select在socket中的使用示例

主题推荐 视频 编码 ffmpeg 摄像头 存储

猜你在找

windows下使用MinGW和msys编译x264

windows 下FFMPEG的编译方法 附2012-9-19发布的

stdint.h 文件 int8_t uint8_t int16_t uint16_t（一）

ffmpeg for android 独立ndk以及android项目下编译 --

网络电话pjsip Getting Started: Building for Apple iPhone,
libjpeg 详解
C++字符数组函数深度剖析
CStatic空间的鼠标事件响应，以及...
h264检测是I帧还是P帧，解决录像第一帧有马赛克问题
ffmpeg Windows下采集摄像头一帧数据，并保存为bmp图

查看评论

6楼 sunnyrainab 2013-11-26 13:40发表

http://download.csdn.net/detail/scalerzhangjie/4862538
从这里下载的视频 yuv420p.yuv和decode.yuv 怎么打开是一片颜色在闪呢，看不到内容呀 。encode.h264 用 vlc可以播放

Re: scalerzhangjie 2013-12-13 16:42发表

回复u012767067：不好意思，这么晚才回复，最近一直在忙，没怎么上网。
我觉得pyuv播放器你没设置好，所以才会闪烁。

我刚刚重新编辑了一下文档，具体配置请见文档最后的图片。

Re: sunnyrainab 2013-12-19 11:51发表

回复scalerzhangjie：谢谢啦，我已经搞定了，呵呵

5楼 smilestone 2013-08-16 09:33发表

给个资源的下载地址

Re: scalerzhangjie 2013-08-20 15:35发表

回复smilestone322：http://download.csdn.net/detail/scalerzhangjie/4862538

4楼 smilestone 2013-08-16 09:31发表

好东西，支持下

Re: scalerzhangjie 2013-12-13 16:43发表

回复smilestone322：谢谢喽

3楼 smilestone 2013-08-16 09:30发表

支持

Re: scalerzhangjie 2013-12-13 16:44发表

回复smilestone322：谢谢喽

2楼 bg2bkk 2013-06-13 13:50发表

测试好用，楼主用心研究了，在此谢谢楼主。
另外想问个问题是，wine怎样能让pyuv在ubuntu上跑起来呢？

Re: scalerzhangjie 2013-06-14 10:48发表

回复bg2bkk：我没用wine，我是两台机器，一台Ubuntu，一台XP，pyuv装在XP机器上。

Re: bg2bkk 2013-06-14 11:03发表

回复scalerzhangjie： 哦这样，不过我已经找到解决办法了。
使用mplayer播放yuv视频，即从摄像头采集到的原始图像

mplayer -demuxer rawvideo -rawvideo w=320:h=240 yuv420p.yuv

使用mplayer可以播放264视频，比如视频文件encode.h264

mplayer encode.h264

ffplay播放yuv图像

ffplay -f rawvideo -video_size 320x240 *.yuv

Re: scalerzhangjie 2013-12-13 16:45发表

回复bg2bkk：厉害！有空我试试。谢谢分享。

1楼 scalerzhangjie 2012-12-08 20:59发表

还是不太喜欢csdn的编辑器，很多东西要么显示出错，要么干脆不显示。哎，什么时候改改啊！