FFMPEG读取关键帧（c++）

FFMPEG读取关键帧（**c++**）

设法使用 **av_read_frame** 获得所有框架，其中逐帧地顺序读取，再用**AVFrame-> key_frame**判断关键帧，将其进行保存。

主要提取代码如下：

```
//获得帧图大小
    PictureSize = avpicture_get_size(AV_PIX_FMT_BGR24, pCodecCtx->width, pCodecCtx->height);
    buf = (uint8_t*)av_malloc(PictureSize);
    if (buf == NULL) {
        printf("av malloc failed!\n");
        system("pause");
        exit(1);
    }
    avpicture_fill((AVPicture *)pFrameRGB, buf, AV_PIX_FMT_BGR24, pCodecCtx->width, pCodecCtx->height);
    //设置图像转换上下文
    pSwsCtx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt, pCodecCtx->width, pCodecCtx->height, AV_PIX_FM
    i = 0;
    while (av_read_frame(pFormatCtx, &packet) >= 0) {
        if (packet.stream_index == videoStream) {
            //解码
            avcodec_decode_video2(pCodecCtx, pFrame, &frameFinished, &packet);
            if (frameFinished) {
                if (pFrame->key_frame) {
                    //转换图像格式，将解压出来的YUV420P的图像转换为BRG24的图像
                    sws_scale(pSwsCtx, (const uint8_t* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height, pFrameRGB->data, pFrameR(
                    //保存为bmp图
                    SaveAsBMP(pFrameRGB, pCodecCtx->width, pCodecCtx->height, i, 24);
                    i++;
                }
            }
            av_free_packet(&packet);
        }
    }
```

> 注意① ffmpeg 中的函数 av_read_frame 是用来完成解封装的过程，它会从码流里面提取每一个音频、视频帧，它使用了结构体 AVPacket 来记录每一帧的信息。

```
AVPacket avpkt;
av_init_packet(&avpkt);

while (!interrupt) {
    int ret = av_read_frame(ic, &avpkt);
    if (ret < 0) {
        break;
    }
    // processing
}

av_free_packet(&avpkt);
```

每循环一次，就从码流中解封装并且提取了一帧数据，并存放在了 AVPacket 结构体中。

注意② 用 AVPacket 的成员变量stream_index判断Packet 到底是音频还是视频：

```
int video_stream_idx = av_find_best_stream(ic, AVMEDIA_TYPE_VIDEO, -1, -1, NULL, 0);
int audio_stream_idx = av_find_best_stream(ic, AVMEDIA_TYPE_AUDIO, -1, -1, NULL, 0);

if (avpkt.stream_index == video_stream_idx) {
    LOGD("read a video frame");
} else if (avpkt.stream_index == audio_stream_idx) {
    LOGD("read audio frame);
}
```

读取视频文件，提取关键帧并保存关键帧代码如下：

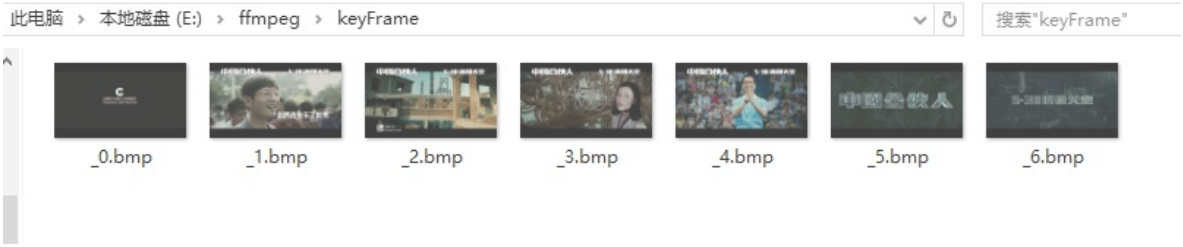
```
#include "stdafx.h"
#include <windows.h>
#include <WinGDI.h>
#include <iostream>
#include <iosfwd>
#include <fstream>

#define __STDC_CONSTANT_MACROS
extern "C"
{

    #include <libavcodec/avcodec.h>//处理原始音频和视频流的解码
    #include <libavutil/opt.h>
    #include <libavutil/channel_layout.h>
    #include <libavutil/common.h>
    #include <libavutil/imgutils.h>
    #include <libavutil/mathematics.h>
    #include <libavutil/samplefmt.h>
    #include <libavformat/avformat.h>//处理解析视频文件并将包含在其中的流分离出来
    #include <libswscale/swscale.h>
    #include <libavutil/imgutils.h>
    #include <libavutil/parseutils.h>
}
#pragma comment(lib,"avcodec.lib")
#pragma comment(lib,"avdevice.lib")
#pragma comment(lib,"avfilter.lib")
#pragma comment(lib,"avformat.lib")
#pragma comment(lib,"avutil.lib")
#pragma comment(lib,"swscale.lib")
#pragma comment(lib,"postproc.lib")
#pragma comment(lib,"swresample.lib")
using namespace std;
//定义BMP文件头



#ifndef _WINGDI_
#define _WINGDI_
typedef struct tagBITMAPFILEHEADER {
    WORD    bfType;
    DWORD   bfSize;
```

```
    WORD    bfReserved1;
    WORD    bfReserved2;
    DWORD   bfOffBits;
} BITMAPFILEHEADER, FAR *LPBITMAPFILEHEADER, *PBITMAPFILEHEADER;

typedef struct tagBITMAPINFOHEADER {
    DWORD      biSize;
    LONG       biWidth;
    LONG       biHeight;
    WORD       biPlanes;
    WORD       biBitCount;
    DWORD      biCompression;
    DWORD      biSizeImage;
    LONG       biXPelsPerMeter;
    LONG       biYPelsPerMeter;
    DWORD      biClrUsed;
    DWORD      biClrImportant;
} BITMAPINFOHEADER, FAR *LPBITMAPINFOHEADER, *PBITMAPINFOHEADER;
#endif
//保存BMP文件的函数
void SaveAsBMP(AVFrame *pFrameRGB, int width, int height, int index, int bpp)
{
    char buf[5] = { 0 };   //bmp头
    BITMAPFILEHEADER bmpheader;
    BITMAPINFOHEADER bmpinfo;
    FILE *fp;
    char *filename = new char[255];  //文件存放路径，根据自己的修改
    sprintf_s(filename, 255, "%s_%d.bmp", "E:/ffmpeg/keyFrame/", index);
    if ((fp = fopen(filename, "wb+")) == NULL) {
        printf("open file failed!\n");
        return;
    }
    bmpheader.bfType = 0x4d42;
    bmpheader.bfReserved1 = 0;
    bmpheader.bfReserved2 = 0;
    bmpheader.bfOffBits = sizeof(BITMAPFILEHEADER) + sizeof(BITMAPINFOHEADER);
    bmpheader.bfSize = bmpheader.bfOffBits + width*height*bpp / 8;
    bmpinfo.biSize = sizeof(BITMAPINFOHEADER);
    bmpinfo.biWidth = width;
    bmpinfo.biHeight = -height;
    bmpinfo.biPlanes = 1;
    bmpinfo.biBitCount = bpp;
    bmpinfo.biCompression = BI_RGB;
    bmpinfo.biSizeImage = (width*bpp + 31) / 32 * 4 * height;
    bmpinfo.biXPelsPerMeter = 100;
    bmpinfo.biYPelsPerMeter = 100;
    bmpinfo.biClrUsed = 0;
    bmpinfo.biClrImportant = 0;
    fwrite(&bmpheader, sizeof(bmpheader), 1, fp);
    fwrite(&bmpinfo, sizeof(bmpinfo), 1, fp);
    fwrite(pFrameRGB->data[0], width*height*bpp / 8, 1, fp);
    fclose(fp);
}

int main()
{
    //printf("%d\n", avcodec_version());

    unsigned int i = 0, videoStream = -1;
    AVFormatContext *pFormatCtx;
    AVCodecContext *pCodecCtx;
    AVCodec *pCodec;
    AVFrame *pFrame, *pFrameRGB;
    struct SwsContext *pSwsCtx;
    const char *filename = "E:/ffmpeg/中国合伙人.flv";
```

```
int frameFinished;
int PictureSize;
AVPacket packet;
uint8_t *buf;
//注册解码器
av_register_all();
avformat_network_init();
pFormatCtx = avformat_alloc_context();


//AVInputFormat *pInputFormt = av_find_input_format("dshow");
if (avformat_open_input(&pFormatCtx, filename, NULL, NULL) != 0) {

    printf("%s\n", "failed");
    system("pause");
}
//获取视频流信息
if (avformat_find_stream_info(pFormatCtx, NULL)<0) {
    printf("%s\n", "couldn`t find stream info");
    system("pause");
}

//获取视频数据
for (int i = 0; i<pFormatCtx->nb_streams; i++)

    if (pFormatCtx->streams[i]->codec->codec_type == AVMEDIA_TYPE_VIDEO) {
        //AVMEDIA_TYPE_VIDEO
        //AV_CODEC_ID_H264
        videoStream = i;
    }

if (videoStream == -1) {
    printf("%s\n", "find video stream failed");
    system("pause");
    exit(1);
}
pCodecCtx = pFormatCtx->streams[videoStream]->codec;
pCodec = avcodec_find_decoder(pCodecCtx->codec_id);

if (pCodec == NULL) {
    printf("%d\n", "avcode find decoder failed!");
    system("pause");
    exit(1);
}
//打开解码器
if (avcodec_open2(pCodecCtx, pCodec, NULL)<0) {
    printf("avcode open failed!\n");
    system("pause");
    exit(1);
}
//为每帧图像分配内存
pFrame = av_frame_alloc();
pFrameRGB = av_frame_alloc();

if (pFrame == NULL || pFrameRGB == NULL) {
    printf("av frame alloc failed!\n");
    system("pause");
    exit(1);
}
//获得帧图大小
PictureSize = avpicture_get_size(AV_PIX_FMT_BGR24, pCodecCtx->width, pCodecCtx->height);
buf = (uint8_t*)av_malloc(PictureSize);
if (buf == NULL) {
    printf("av malloc failed!\n");
    system("pause");
```

```
            exit(1);
        }
        avpicture_fill((AVPicture *)pFrameRGB, buf, AV_PIX_FMT_BGR24, pCodecCtx->width, pCodecCtx->height);
        //设置图像转换上下文
        pSwsCtx = sws_getContext(pCodecCtx->width, pCodecCtx->height, pCodecCtx->pix_fmt, pCodecCtx->width, pCodecCtx->height, AV_PIX_FM
        i = 0;
        while (av_read_frame(pFormatCtx, &packet) >= 0) {
            if (packet.stream_index == videoStream) {
                //解码
                avcodec_decode_video2(pCodecCtx, pFrame, &frameFinished, &packet);
                if (frameFinished) {
                    if (pFrame->key_frame) {
                        //转换图像格式，将解压出来的YUV420P的图像转换为BRG24的图像
                        sws_scale(pSwsCtx, (const uint8_t* const*)pFrame->data, pFrame->linesize, 0, pCodecCtx->height, pFrameRGB->data, pFrameR(
                        //保存为bmp图
                        SaveAsBMP(pFrameRGB, pCodecCtx->width, pCodecCtx->height, i, 24);
                        i++;
                    }
                }
                av_free_packet(&packet);
            }
        }
        sws_freeContext(pSwsCtx);
        av_free(pFrame);
        av_free(pFrameRGB);
        avcodec_close(pCodecCtx);
        avformat_close_input(&pFormatCtx);
        printf("关键帧已保存在设置路径！\n");
        system("pause");
        return 0;
    }
```

运行效果：



(o゜▽゜)o☆[BINGO!] 欢迎来个人网站做客**(o゜▽゜)o☆[BINGO!]**