

 流媒体技术 专栏收录该内容

0 订阅 7 篇文章 订阅专栏

之前一直是采用V4L2的基础协议进行USB数据采集，然而并不是非常的方便，在移植了FFMPEG之后，有了另外一种选择。相对于直接采集方便了很多。

ffmpeg.cpp

```
1 #include "ffmpeg.h"
2
3
4 #define FFMPEG_MJPEG
5 // #define FFMPEG_H264
6 // #define FFMPEG_YUV
7
8 #define TIMEMS      qPrintable(QTime::currentTime().toString("HH:mm:ss zzz"))
9
10 ffmpeg::ffmpeg(QWidget *parent) :
11     QThread(parent)
12 {
13     framCount = 0;
14     frameFinish = 0;
15     saveFile = true;
16     framIndex = 0;
17     isOutputFileOpen = false;
18 }
19
20
21 ffmpeg::~ffmpeg()
22 {
23 }
24
25
26 /* 功能：初始化解封装上下文，解码器上下文，和格式转换上下文 (yuv转rgb)
27 *      1 解封装
28 *      2 解码
29 *      3 格式转换
30 * 参数：无
31 * 返回值：成功返回零，失败返回-1
32 */
33 int ffmpeg::initDecodeVideo()
34 {
35     //注册库中所有可用的文件格式和解码器
36     av_register_all();
37     //注册所有设备，主要用于本地摄像机播放支持
38     avdevice_register_all();
39
40     qDebug() << TIMEMS << "init ffmpeg lib ok" << " version:" << FFMPEG_VERSION;
41
42
43     AVDictionary *options = NULL;
44     AVCodec *deCodec = NULL; //解码器
45
46 }
```

分类专栏

	STM32开发	22篇
	笔记	1篇
	机器人原理与实践	3篇
	数字图像处理	18篇
	边缘计算	1篇
	流媒体技术	7篇
	LINUX	21篇
	嵌入式UI开发	2篇
	QT	6篇
	C++	4篇
	数据库技术	6篇
	转载	1篇
	随笔	4篇
	WEB与LOT	1篇
	Android开发	2篇
	嵌入式实时操作系统	13篇
	开发环境搭建	3篇

```
47 //为解封装上下文开辟空间
48 ifmt_ctx = avformat_alloc_context();
49 //解封装对象
50 AVInputFormat *ifmt = av_find_input_format("video4linux2");
51
52 //打开输入视频流, 进行解封装
53 av_dict_set(&options, "framerate", "30", 0);
54 av_dict_set(&options, "video_size", "1280x720", 0);
55 #ifdef FFmpeg_MJPEG
56 av_dict_set(&options, "input_format", "mjpeg", 0);
57 #endif
58
59 #ifdef FFmpeg_YUV
60 av_dict_set(&options, "input_format", "yuyv422", 0);
61 #endif
62 int result = avformat_open_input(&ifmt_ctx, inputFilename, ifmt, &options);
63 if (result < 0) {
64     qDebug() << TIMEMS << "open input error" << inputFilename;
65     return false;
66 }
67 //释放设置参数
68 if(options != NULL) {
69     av_dict_free(&options);
70 }
71
72 //获取流信息
73 result = avformat_find_stream_info(ifmt_ctx, NULL);
74 if (result < 0) {
75     qDebug() << TIMEMS << "find stream info error";
76     return false;
77 }
78 videoStreamIndex = -1;
79
80 videoStreamIndex = av_find_best_stream(ifmt_ctx, AVMEDIA_TYPE_VIDEO, -1, -1, &deCodec, 0);
81
82 if (videoStreamIndex < 0) {
83     qDebug() << TIMEMS << "find video stream index error";
84     return false;
85 }
86
87 //从输入封装上下文获取输入视频流
88 in_stream = ifmt_ctx->streams[videoStreamIndex];
89 if (!in_stream)
90 {
91     printf("Failed get input stream\n");
92     return false;
93 }
94
95 //获取视频流解码器上下文
96 deCodecCtx = in_stream->codec;
97
98 //获取分辨率大小
99 videoWidth = in_stream->codec->width;
100 videoHeight = in_stream->codec->height;
101
102 //如果没有获取到宽高则返回
103 if (videoWidth == 0 || videoHeight == 0) {
104     qDebug() << TIMEMS << "find width height error";
105     return false;
106 }
107
```

```

108 //获取视频流的帧率 fps,要对0进行过滤,除数不能为0,有些时候获取到的是0
109 int num = in_stream->codec->framerate.num;
110 int den = in_stream->codec->framerate.den;
111 if (num != 0 && den != 0) {
112     videoFps = num / den ;
113 }
114
115 QString videoInfo = QString("视频流信息 -> 索引: %1 格式: %2 时长: %3 秒 fps: %4 分辨率: %5*%6")
116     .arg(videoStreamIndex).arg(ifmt_ctx->iformat->name)
117     .arg((ifmt_ctx->duration) / 1000000).arg(videoFps).arg(videoWidth).arg(videoHeight);
118 qDebug() << TIMEMS << videoInfo;
119
120 //打开视频解码器
121 result = avcodec_open2(deCodecCtx, deCodec, NULL);
122 if (result < 0) {
123     qDebug() << TIMEMS << "open video codec error";
124     return false;
125 }
126
127 avDePacket = av_packet_alloc();
128 avDeFrameYuv = av_frame_alloc();
129 avDeFrameRgb = av_frame_alloc();
130
131 //比较上一次文件的宽度高度,当改变时,需要重新分配内存
132 if (oldWidth != videoWidth || oldHeight != videoHeight) {
133     int byte = avpicture_get_size(AV_PIX_FMT_RGB32, videoWidth, videoHeight);
134     buffer = (uint8_t *)av_malloc(byte * sizeof(uint8_t));
135     oldWidth = videoWidth;
136     oldHeight = videoHeight;
137 }
138
139 //定义像素格式
140 AVPixelFormat srcFormat = AV_PIX_FMT_YUV420P;
141 AVPixelFormat dstFormat = AV_PIX_FMT_RGB32;
142
143 #ifdef FFmpeg_MJPEG
144     srcFormat = AV_PIX_FMT_YUV420P;
145 #endif
146
147 #ifdef FFmpeg_YUV
148     srcFormat = AV_PIX_FMT_YUV422;
149 #endif
150
151 #ifdef FFmpeg_H264
152     srcFormat = AV_PIX_FMT_YUV420P;
153 #endif
154 av_image_fill_arrays(avDeFrameRgb->data, avDeFrameRgb->linesize, buffer, dstFormat, videoWidth, videoHeight,
155 int flags = SWS_FAST_BILINEAR;
156
157 swsContextYuvtoRgb = sws_getContext(videoWidth, videoHeight, srcFormat, videoWidth, videoHeight, dstFormat,
158
159 //打开输出视频的文件
160 outFile.setFileName(outputFilename);
161 outFile.open(QIODevice::WriteOnly);
162
163
164 qDebug() << TIMEMS << "init ffmpegVideo ok";
165
166 return 0;
167 }
168

```



```

169 int ffmpeg::playVideo()
170 {
171     initDecodeVideo();
172     while(true)
173     {
174         if (av_read_frame(ifmt_ctx, avDePacket) >= 0) {
175             //判断当前包是视频还是音频
176             int index = avDePacket->stream_index;
177             in_stream = ifmt_ctx->streams[index];
178
179             if (index == videoStreamIndex) {
180                 avcodec_decode_video2(deCodecCtx, avDeFrameYuv, &frameFinish, avDePacket);
181                 if (frameFinish)
182                 {
183
184                     //将数据转成一张图片YuvtoRgb
185                     sws_scale(swsContextYuvtoRgb, (const uint8_t *const *)avDeFrameYuv->data, avDeFrameYuv->linesize,
186                             0, videoHeight, avDeFrameRgb->data, avDeFrameRgb->linesize);
187
188                     QImage image((uchar *)buffer, videoWidth, videoHeight, QImage::Format_RGB32);
189
190                     if (!image.isNull()) {
191                         emit receiveImage(image);
192                     }
193
194                     framIndex ++;
195                     qDebug()<< "解码到第" << framIndex << "帧";
196                     qDebug() << TIMEMS;
197                     if(framIndex > 200)
198                     {
199                         framIndex = 0;
200                         break;
201                     }
202
203                     for(int i = 0;i < avDeFrameYuv->height;i++){
204                         outFile.write((char *) (avDeFrameYuv->data[0] + i * avDeFrameYuv->linesize[0]),avDeFrame'
205                     }
206
207                     int loop = avDeFrameYuv->height / 2;
208                     int len_uv = avDeFrameYuv->width / 2;
209
210                     for(int i = 0;i < loop;i++){
211                         outFile.write((char *) (avDeFrameYuv->data[1] + i * avDeFrameYuv->linesize[1]),len_uv);
212                     }
213                     for(int i = 0;i < loop;i++){
214                         outFile.write((char *) (avDeFrameYuv->data[2] + i * avDeFrameYuv->linesize[2]),len_uv);
215                     }
216
217                 }
218                 av_packet_unref(avDePacket);
219                 av_freep(avDePacket);
220             }
221         }
222     }
223 }
224
225 outFile.close();
226
227 avformat_free_context(ifmt_ctx);
228 //关闭编码和解码器
229 avcodec_close(deCodecCtx);

```

```

230 //清理编码器和解码器上下文
231 avcodec_free_context(&deCodecCtx);
232 //清理格式转换上下文
233 sws_freeContext(swsContextYuvtoRgb);
234
235 qDebug() << TIMEMS << "stop ffmpeg thread";
236 }
237 void ffmpeg::run()
238 {
239     playVideo();
240 }

```

ffmpeg.h

```

1  #ifndef FFMPEG_H
2  #define FFMPEG_H
3
4  #include <QMainWindow>
5  #include <QMutex>
6  #include <QDateTime>
7  #include <QFile>
8  #include <QThread>
9  #include <QDebug>
10
11 //引入ffmpeg头文件
12 extern "C" {
13     #include "libavutil/opt.h"
14     #include "libavutil/time.h"
15     #include "libavutil/frame.h"
16     #include "libavutil/pixdesc.h"
17     #include "libavutil/avassert.h"
18     #include "libavutil/imgutils.h"
19     #include "libavutil/ffversion.h"
20     #include "libavcodec/avcodec.h"
21     #include "libswscale/swscale.h"
22     #include "libavdevice/avdevice.h"
23     #include "libavformat/avformat.h"
24     #include "libavfilter/avfilter.h"
25
26     #ifndef gcc45
27     #include "libavutil/hwcontext.h"
28     #endif
29 }
30
31
32 namespace Ui {
33 class ffmpeg;
34 }
35
36 class ffmpeg : public QThread
37 {
38     Q_OBJECT
39
40 public:
41     explicit ffmpeg(QWidget *parent = nullptr);
42     ~ffmpeg();
43     char *outputFilename;
44     char *inputFilename;
45
46 protected:

```

```
47 void run();
48 signals:
49 //收到图片信号
50 void receiveImage(const QImage &image);
51
52 private:
53     uint64_t framIndex;
54     int lastMsec;
55     int videoStreamIndex;           //视频流索引
56     int videoWidth;                 //视频宽度
57     int videoHeight;                //视频高度
58     int videoFps;                   //视频流帧率
59     int frameFinish;                //一帧完成
60     bool saveFile;
61     bool isOutputFileOpen;
62     uint64_t framCount;             //帧计数
63
64     uint8_t *buffer;                //存储解码后图片buffer
65     AVOutputFormat *ofmt = NULL;    //输出格式
66
67     AVPacket *avDePacket;           //解码包对象
68
69     AVFrame *avDeFrameYuv;           //解码帧对象YUV
70     AVFrame *avDeFrameRgb;           //解码帧对象RGB
71
72     AVFormatContext *ifmt_ctx;        //输入封装格式对象
73     AVFormatContext *ofmt_ctx;        //输出封装格式对象
74
75     AVStream *in_stream;              //输入视频流
76     AVStream *out_stream;             //输出视频流
77
78     AVCodecContext *deCodecCtx;       //解码器上下文
79
80     SwsContext *swsContextYuvtoRgb;    //格式转换上下文 ( YuvtoRgb )
81
82     int oldWidth;                     //上一次视频宽度
83     int oldHeight;                    //上一次视频高度
84
85     QFile outFile;
86
87 private:
88     Ui::ffmpeg *ui;
89     int initDecodeVideo();
90     int playVideo();
91
92 };
93
94 #endif // FFMPEG_H
```

这里需要移植QT，这是后面的硬件编码的RTMP推流的基础。

usb摄像头视频采集及格式转换yuv420

07-31

该资源是基于FFmpeg进行的二次开发，该压缩包包含两部分：[视频采集类](#)、[视频格式转换类](#)，[视频采集类](#)根据用户传入的设备名字、帧率、[视频大小](#)进...

FFmpeg实现获取USB摄像头视频流测试代码

网络资源是有限的 5285

通过USB摄像头(注：windows7/10下使用内置摄像头，linux下接普通的usb摄像头(Logitech))获取视频流用到的模块包括avformat和avdevice。头文件仅in...



优质评论可以帮助作者获得更高权重

抢沙发



评论



与光同程

码龄3年 武汉大学

110	14万+	1万+	11万+	
原创	周排名	总排名	访问	等级

2275	825	219	74	824
积分	粉丝	获赞	评论	收藏

