文档编号:

Infrared Systems

StreamSDK
开发指南

文档说明

日期	版本	修订者	修订说明	参与评审人员	审核人
2018.8.27	1.2				
2017.9.14	1.1				
2017.8.21	1.0				

目录

一、概述

StreamSDK 包含了访问网络型热像仪流式数据的通用函数接口。本文档用示例对 SDK 的函数使用方式进行描述。头文件中有详细的函数接口说明。

本文基于的 StreamSDK 版本为 1.2.0.1

1. 文件列表

- StreamSDK.dll SDK 的动态链接库
- StreamSDK.h 标准 C 语言 SDK 接口头文件
- StreamSDK.lib 引导库

二、数据流

本文档将通过网络进行各种实时传输的具有时效的数据都称为数据流，不同类型的数据以流方式传输时会以数据帧的方式分隔，每个数据帧除了数据本身外还包含帧序列、时间戳等时序信息，具体的封装格式参见《热像仪软件接口手册》

StreamSDK 除了将原始信息接收到本地外，也提供了将数据帧进行解析的函数接口，如 H.264 视频流解码为 RGB 图像数据。

1. 网络重连

当获取了数据流句柄之后，StreamSDK 会将连接参数缓存并持续尝试连接数据流，当网络发生异常时，数据流句柄仍然有效并进入到重连尝试状态。

2. 回调（异步）方式获取数据

用户通过 set_grabber 向 StreamSDK 提供一个回调函数接口，当客户端接收到一帧完整数据时，调用该回调函数，用户在回调函数中完成所需要的数据处理，或者，如果处理时间较长时，将数据传递给另一个线程处理。

回调函数的方式保证数据接收的实时性，但该函数的处理最好在下一帧数据到达前完成，不然就会发生数据丢失。例如视频以 30fps 速度采集，那么回调函数处理时间不能超过 33.3ms

3. 抓取（同步）方式获取数据

用户通过 grab_xx 函数请求一帧数据，当数据还未接收到时，函数会阻塞等待，直到

数据到达，或者等待设定的超时时间，函数返回 STREAMSDK_EC_TIMEOUT。这种数据获取方式的程序实现更为简单，但在获取数据之前，用户需要自行分配足够的缓存空间可容纳至少一帧数据。

三、数据流操作

1. 创建和销毁流

每个数据流是作为 URL 资源来定位的，所以创建时需要指定 IP 地址或者使用域名，并且要指定正确的流服务端口号。

创建流后一定要在不需要流资源时销毁，否则会导致内存泄漏。

```
01    HSTREAM video_stream = NULL;
02    if(streamsdk_create_stream("192.168.1.100",10081,&video_stream)       !=
STREAMSDK_EC_OK)
03    {
04       std::cerr<<"Fail to create stream"<<std::endl;
05       return NULL;
06    }
07
08    ...
09    streamsdk_destroy_stream(video_stream)
```

2. 流接收及回调函数

数据流以数据帧的形式接收，SDK 每收到一个完整的数据帧会调用一次回调函数，用户可以通过访问 streamsdk_st_buffer 来读取数据帧的信息

```
01    HSTREAM video_stream = NULL;
02    if(streamsdk_create_stream("192.168.1.100",10081,&video_stream)!=STREAM
SDK_EC_OK)
03    {
04       std::cerr<<"Fail to create stream"<<std::endl;
05       return NULL;
06    }
07
08    /* max_packet_size 来自流属性信息，请参考相关接口手册 */
09    streamsdk_start_stream(video_stream,"/video/raw",max_packet_size,grabRa
w,NULL);
10    ...
11    streamsdk_stop_stream(video_stream);
```

```
12    streamsdk_destroy_stream(video_stream)
```

回调函数

```
13    void CALLBACK grabRaw(int ec, streamsdk_st_buffer * buf,void * )
14    {
15       std::cout<<"packet: "
16       <<buf->pts<<"size: "<<buf->buf_size<<std::endl;
17
18    ...
19    }
```

当网络连接从异常中恢复时，回调函数会继续有效。

3. 流抓取

接收数据流的另一种方式是抓取方式，不需要使用回调函数这样的异步方法，通过 streasmsdk_grab_buffer 来进行，可参考后面图像抓取的示例。

四、H.264 视频流解码

SDK 内部可以对视频流进行处理后再返回给用户，视频流可以使用相关接口先进行解码

1. 解码示例

```
01    HSTREAM video_stream = NULL;
02    if(streamsdk_create_stream("192.168.1.100",10081,&video_stream)!=STREAM
SDK_EC_OK)
03    {
04       std::cerr<<"Fail to create stream"<<std::endl;
05       abort();
06    }
07
08    /* max_packet_size 来自流属性信息，请参考相关接口手册 **/
09    streamsdk_start_stream(video_stream,"/video/raw",max_packet_size,grabRa
w,NULL);
10
11    HDECODER dec;
12    streamsdk_st_decoder_param dp;
13    dp.dec_w = 384;
14    dp.dec_h = 288;
15    dp.pix_type = STREAMSDK_PIX_BGR;
```

```
16
17    if(streamsdk_create_h264_decoder(video_stream,&dp,&dec)!=STREAM_SDK_EC_
OK)
18    {
19       std::cerr<<"Fail to create decorder"<<std::endl;
20       abort();
21    }
22
23    streamsdk_start_h264_decoder(dec,grabImage,NULL);
24    /**等待用户中止**/
25    ...
26    streamsdk_stop_h264_decoder(dec);
27    streamsdk_stop_stream(video_stream);
28    streamsdk_destroy_stream(video_stream)
```

回调函数

```
29    void CALLBACK grabImage(int ec, streamsdk_st_image * img,void * )
30    {
31       std::cout<<"packet: "
32       <<img->pts<<"width: "<<img->img_w
33       <<"height: "<<img->img_h<<std::endl;
34
35    ...
36    }
```

2. 图像抓取

解码图像数据也支持抓取方式：

```
37    HSTREAM video_stream = NULL;
38    if(streamsdk_create_stream("192.168.1.100",10081,&video_stream)!=STREAM
SDK_EC_OK)
39    {
40       std::cerr<<"Fail to create stream"<<std::endl;
41       abort();
42    }
43
44    /* max_packet_size 来自流属性信息，请参考相关接口手册 **/
45    streamsdk_start_stream(video_stream,"/video/raw",max_packet_size,grabRa
w,NULL);
46
47    HDECODER dec;
48    streamsdk_st_decoder_param dp;
```

```
49    dp.dec_w = 384;
50    dp.dec_h = 288;
51    dp.pix_type = STREAMSDK_PIX_BGR;
52
53    if(streamsdk_create_h264_decoder(video_stream,&dp,&dec)!=STREAM_SDK_EC_
OK)
54    {
55       std::cerr<<"Fail to create decorder"<<std::endl;
56       abort();
57    }
58
59    streamsdk_start_h264_decoder(dec,grabImage,NULL);
60
61    streamsdk_st_image img;
62    img.img_w = dp.dec_w;
63    img.img_h = dp.dec_h;
64    img.img_linesize = dp.dec_w * 3;
65    img.img_type = STREAMSDK_PIX_BGR;
66    img.img_ptr = NULL;
67    streamsdk_create_image(&img);  // 根据图像信息分配足够的图像缓存
68
69
70    int ret;
71    bool is_quit = false
72    while(is_quit == false)
73    {
74       ret = streamsdk_grab_image(dec,&img,1000);
75       if(ret == STREAMSDK_EC_FAIL)
76           abort();
77
78       if(ret == STREAMSDK_EC_TIMEOUT)
79       {
80           std::cerr<<"grab timeout"<<std::endl;
81       }
82       else
83       {
84           /** TODO:处理接收到的图像 **/
85
86       }
87       if(keyPress() == 'q')
88           is_quit = true;
89       }
90    }
91    streamsdk_destroy_image(&img);
```

```
92    streamsdk_stop_h264_decoder(dec);
93    streamsdk_stop_stream(video_stream);
94    streamsdk_destroy_stream(video_stream)
```