PanSight SDK 使用说明

目录

[一、SDK 简介 3](#_Toc12496)

[二、开发环境 3](#_Toc11976)

[三、函数使用流程 5](#_Toc23332)

[1. 连接相机 5](#_Toc7415)

[2. 固定数量点云抓取 6](#_Toc5558)

[3. 实时图像上报 7](#_Toc30722)

[4. 实时轮廓上报 9](#_Toc5621)

[四、示例代码 10](#_Toc11844)

[1. 单机 10](#_Toc3282)

[2. 双机 11](#_Toc16612)

[五、同步接口与异步接口的区别 13](#_Toc8936)

[六、C#调用SDK说明 14](#_Toc31198)

[七、SDK上报的数据说明 14](#_Toc22249)

[八、 不常用参数获取 15](#_Toc20277)

[九、 SDK使用注意事项 16](#_Toc22647)

[十、 SDK 配置文件 17](#_Toc17642)

## 一、SDK 简介

PanSight软件开发工具包（SDK）包括头文件、lib库、dll动态库以及C#、MFC、QT等编写的demo代码；

SDK对外开放的API接口函数都是标准C方式编译实现（调用约定为 \_\_cdecl），支持C#、MFC、QT 等应用程序集成调用。目前只开放了X64版本的相关库文件，所以应用程序必须编译成**X64**的版本长能正常使用SDK。

目前SDK支持windows x64、Linux(ubuntu)平台使用。

注：*sdk在使用时，****需要先安装驱动****（PanSight软件目录下vc2017\_redist\_x64.exe ）*

## 二、开发环境

**1.** 需要**VS2017**或以上版本（未全部测试过，如有问题，请使用VS2017的版本）；*VS2010 也可以兼容（不推荐）*。

**2.** SDK 包的目录（.\SDK\_DEMOS\）中包含了解决方案文件SDK\_DEMOS.sln（vs2017），打开后可以看到5个工程，其中3个是C#的demo，一个简单的C++ demo （SimpleConsole），一个是MFC的demo。如下图：



*注：SDK\_DEMOS\SDK\_DEMOS\_2010.sln 是使用vs2010编译的demo。由于SDK本身是vs2017编译的，所以该版本不能使用debug版本进行调试，但是可以使用release版本进行调试。*

* CSharpDemo: C#版本基本功能demo。该demo程序主要是指定如何使用SDK的API 进行相机的连接、参数获取、参数修改、以及数据的采集。
* CSharpVisionProDemo: 实现了visionpro SDK的基本功能调用。包含了点云数据转visionpro的图像格式。
* HalconDemo:实现相机图像获取，以及halcon SDK 的调用。包含点云数据、灰度数据转halcon图像功能。
* MFC\_DEMO: MFC 版本的demo，包含了调试图像显示，数据采集，编码器设置等功能。
* SimpleConsole: C++ 版本的控制台程序，简单的实现了相机探测、连接、数据采集功能。

1. Qt demo 工程文件位于(SDK\_DEMOS\QT\_DEMO\QtDemo\QtDemo.pro) 简单实现了相机连接及数据采集的功能。（该demo在windows、linux下都可以使用；windows下使用qt5.14.2+vs2017x64 ；linux下使用qt5.14.2）

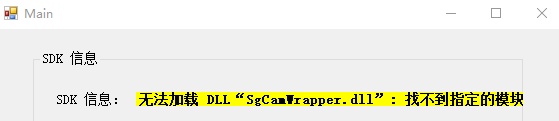
**4.** 所有的Demo编译后生成文件的路径：CamWrapper\bins\X64\，SDK 的相关dll 也在该路径下。

**5.** lib库路径：CamWrapper\libs\ *（使用C#则不需要关注这个）*

**6.** 头文件路径：CamWrapper\3rdParty\ 和 CamWrapper\include\ *（使用C#则不需要关注这个）*

*注：*

* *如果安装的vs版本非2017版，直接对C# 的****Debug****版本进行代码调试时，会出现如下错误：*

**

*解决办法：*

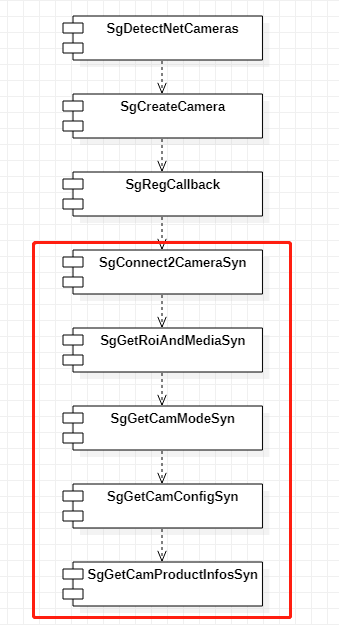
* *第一种：安装vs2017*
* *第二种：拷贝release 目录下的dll ，放到debug 目录下（有可能还是会有问题）*
* *如果安装的vs版本非2017版，直接对MFC 的****Debug****版本进行代码调试时，可能会提示一些系统的dll 不存在。*

*解决方法：*

* 第一种：安装vs2017
* 第二种：下载相应的dll（可能需要好几个）放到debug目录下*（不推荐，很多情况下会还是会失败）*
* *拷贝release 目录下的dll ，放到debug 目录下（有可能还是会有问题）*

## 三、函数使用流程

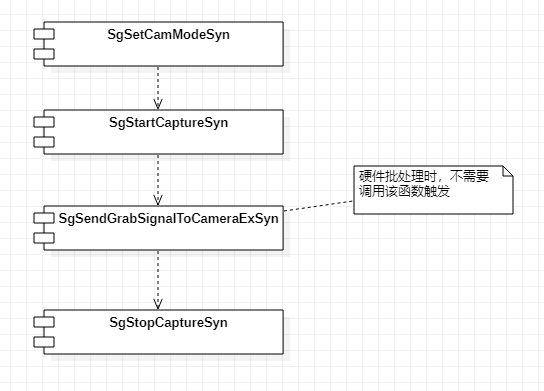
### 1. 连接相机



注：

* **相机上电后，**通过SgDetectNetCameras可以探测到相机IP。
* 按照需要进行回调函数注册。如果指令操作使用异步接口，则需要注册对应指令响应的回调函数。如果需要获取调试图像或者点云数据，则也要先注册好回调函数；具体回调函数定义请见接口说明文档。
* 连接成功后，必须调用4个获取相机参数的函数（顺序如上图）；如果使用异步接口，则使用SgGetCamParam 来获取相机参数（该函数的第2个参数分别是PARAM\_OPTION\_GET\_ROI、PARAM\_OPTION\_GET\_MODE、PARAM\_OPTION\_GET\_CONFIG、PARAM\_OPTION\_GET\_PRODINFO）
* **连接成功并且相关参数都获取成功后，才能调用修改配置的API或者StartCaptureSyn等API**（根据实际情况而定是否需要修改相机参数）
* 不可以同时有多个软件（客户自己调用SDK二次开发的软件、PanSight软件等同时连接同一个相机）

### 2. 固定数量点云抓取



注：

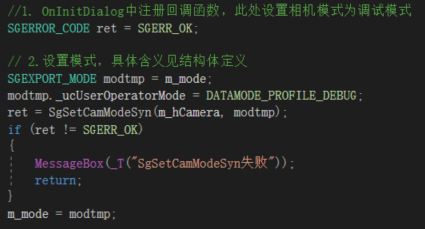
* 先设置好要抓取的线数（调用函数SgSetCamModeSyn）；
* 然后启动开始准备抓取（SgStartCaptureSyn调用后，SDK进入等待状态，等待相机数据，当采集的数据达到预设的线数之后，SDK将调用预先注册好的回调函数SG\_CAP\_DEPTH\_CB）。
* 在调用SgStartCaptureSyn后，**请勿修改相机参数**；如需修改，请先调用SgStopCaptureSyn停止数据抓取，然后再修改。
* 不要在注册的回调函数中直接调用SDK的API，如有需要，请进行线程切换到主线程（防止SDK内部线程死锁）
* 如果通过SgSendGrabSignalToCameraSyn同步抓取数据，返回的内存中的数据只包含Z值，这和回调上报的数据格式不一样！
* 如果当前是硬件批处理方式，则不需要调用SgSendGrabSignalToCameraExSyn来触发一幅图像的抓取。
* SgStartCaptureSyn和SgStopCaptureSyn必须配对使用。
* 如果使用编码器时，请注意编码器的脉冲数量较少（少于预设的要抓取的线数对应的脉冲）或者中途停止，都可能导致SDK上报的线数和预设的不一致。*（SDK 默认1秒钟无信号，则认为已经抓取结束）。*

### 3. 实时调试图像上报

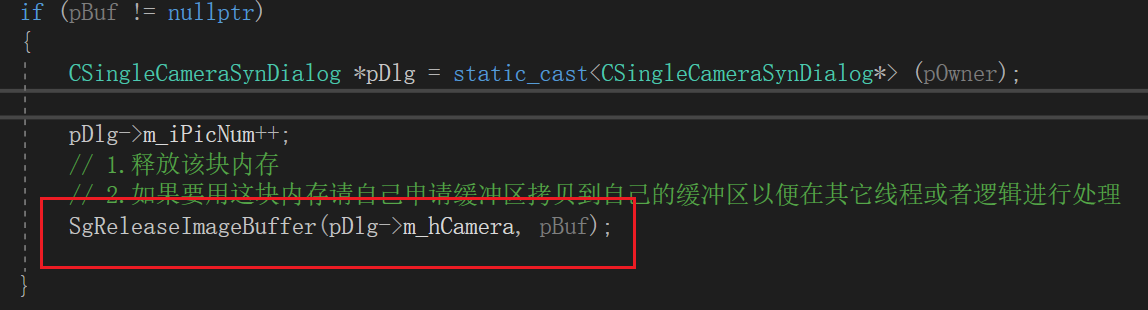


注：

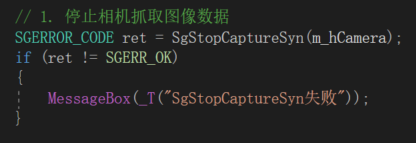
* 注册的接收图像数据回调函数会每收到一幅图像就调用一次。格式与上报的实时轮廓数据与点云数据格式是不一样的。
* 设置模式具体，请查看SGEXPORT\_MODE结构体成员的具体定义；该处的m\_mode是连接相机后获取mode保存下来的，



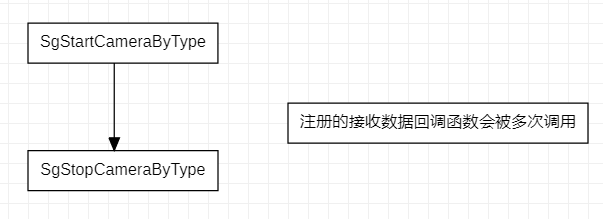
* 需要格外注意的是，在实时图像数据回调函数中，处理完**buf一定要调用SgReleaseImageBuffer**，否则会导致后面的数据拿不到内存而不上报图像数据；



* 停止抓取。

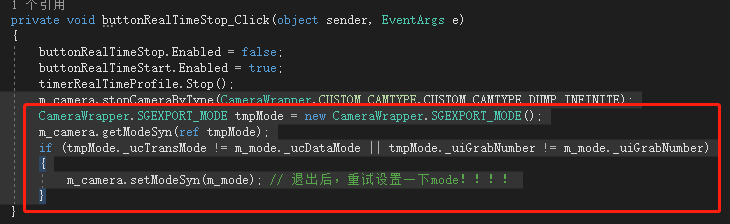


### 4. 实时轮廓上报 （SDK暂不支持该功能）



注：

* 注册的接收数据回调函数会每收到一条轮廓线就调用一次。此时上报的轮廓数据格式和点云格式是一样的。
* 也可以通过SgSendGrabSingleProfileSignalToCameraSyn函数抓取当前一条轮廓（替代回调上报的方式，不建议这么使用；统一使用回调的方式更好理解）。
* 在调用SgStartCameraByType后，**请勿修改相机参数**；如需修改，请先调用SgStopCameraByType停止数据抓取.
* 如果项目中需要实时轮廓上报、固定数量点云抓取功能反复切换时，在调用SgStopCameraByType后，最好从新获取下SGEXPORT\_MODE，并确认相关参数是否需要重新设置！！！



## 四、示例代码

### 1. 单机

使用同步接口方式实现数据抓取，详细代码实现请参考MFC\_DEMO中的CSingleCameraSynDialog

class CameraListener

{

public:

CameraListener();

~CameraListener();

// 上报抓取数据 （注意不要在该回调函数中调用SDK 的API）

static void SG\_CAP\_DEPTH\_CBFUN(void \*pBuf, void\*pGrayBuf, SG\_DEPTHDATA\_PARAM \*pParam, void \*pOwner);

};

void CameraListener::SG\_CAP\_DEPTH\_CBFUN(void \* pBuf, void \* pGrayBuf, SG\_DEPTHDATA\_PARAM \* pParam, void \* pOwner)

{

if (!pBuf || !pParam || !pOwner)

{

return;

}

float \* pfBuf = (float \*)pBuf;

CameraListener \*pThis = (CameraListener \*)pOwner;

pThis->SaveCloud(pfBuf, pParam->\_iCapturedProfileLineNum, pParam->\_iPointNumPerLine, 1);

}

CameraListener m\_CameraListener;

CAMHANDLE m\_hCamera;

SGEXPORT\_ROI m\_roi;

SGEXPORT\_MODE m\_mode;

SGEXPORT\_CAMCONFIG m\_cfg;

SGEXPORT\_PRODUCTINFO m\_product;

const char \*pIPs = SgDetectNetCameras(); // 返回探测到的相机IP及对应的本地IP和网卡信息

m\_hCamera = SgCreateCamera();

// 注册接收轮廓数据的回调函数

SgRegCallback(m\_hCamera, CBTYPE\_CAP\_DEPTH, CameraListener::SG\_CAP\_DEPTH\_CBFUN, &m\_CameraListener);

SGERROR\_CODE ret = SgConnect2CameraSyn(m\_hCamera, “192.168.2.3”, “192.168.2.101”);

ret = SgGetRoiAndMediaSyn(m\_hCamera, m\_roi);

ret = SgGetCamModeSyn(m\_hCamera, m\_mode);

ret = SgGetCamConfigSyn(m\_hCamera, m\_cfg);

ret = SgGetCamProductInfosSyn(m\_hCamera, m\_product);

m\_mode.\_uiGrabNumber = 200; // 设置为抓取200条轮廓

ret = SgSetCamModeSyn(m\_hCamera, m\_mode);

m\_cfg.\_uiExpo = 100; // 曝光修改为100

ret = SgSetCamConfigSyn(m\_hCamera, m\_cfg);

// 准备开始抓取数据

ret = SgStartCaptureSyn(m\_hCamera);

// 软件触发一次数据抓取 （反复调用则可以多次抓取，请在回调函数中收到数据后，再进行下一次触发）

ret = SgSendGrabSignalToCameraExSyn(m\_hCamera);

// 多次触发抓取后，退出数据抓取

//...

ret = SgStopCaptureSyn(m\_hCamera); // stop 之后，未抓完的数据也不会上报！

## 五、同步接口与异步接口的区别

* 对应连接相机、获取相机各种参数、设置相机参数、启动相机等接口函数，同时支持同步接口和异步接口；这些函数如果是 XXXSyn 格式的函数名称，则表示同步接口。
* 异步接口使用前，要先注册对应的响应回调函数，例如：SgRegCallback(m\_hCamera, CBTYPE\_CONNECT, SG\_CONNECT\_RSP\_CBFUN, this); 当调用函数SgConnect2Camera后（该函数立即返回），等SDK和相机真正连接上之后，回调函数SG\_CONNECT\_RSP\_CBFUN会被调用；
* 同步接口使用简单，但是如果对UI要求很高时，可能这些函数的触发会使得UI稍有点卡顿；异步函数使用相对复杂，需要自行管理好调用的顺序，并且所有的回调函数中都不可以直接调用SDK的其他函数（目前还未做强制要求，需要自行切换到工作线程中调用SDK的其他api）
* 如果使用异步接口的方式，建议所用的接口函数的调用统一在一个线程中调用执行。（SgDetectNetCameras函数除外）

注：对于项目开发，**推荐使用同步接口（开发简洁、要求低）**。

## 六、C#调用SDK说明

需要引用CameraEx.dll ，并使用其中封装的CameraSyn类。该类是对C++ SDK 的同步接口的二次封装。函数在命名时，去除了Sg前缀和Syn后缀。具体使用，请参考demo代码

## 七、SDK上报的数据说明

上报数据的回调函数定义如下：

typedef void(\*SG\_CAP\_DEPTH\_CB)(void \*pBuf, void\*pGrayBuf, SG\_DEPTHDATA\_PARAM \*pParam, void \*pOwner);

typedef void(\*SG\_PAIRCAM\_CAP\_DEPTH\_CB)(void \*pBuf1, void\*pGrayBuf1, SG\_DEPTHDATA\_PARAM \*pParam1, void \*pBuf2, void\*pGrayBuf2, SG\_DEPTHDATA\_PARAM \*pParam2, void \*pOwner);

参数说明：

* pBuf、pBuf1、pBuf2，指向深度数据内存，其内存指向为3维点数组stPoint。如下：

struct stPoint

{

float y; // 时间方向，运动方向；同一条轮廓线该值相同

float x;// 沿激光方向

float z; // 高度值

};

* pGrayBuf、pGrayBuf1、pGrayBuf2, 指向**unsigned char 数组**，记录的是每个点的灰度值。*（如要使用灰度数据，请先打开configs.ini中的supportGrayMode配置项，重启软件后再调用API设置模式中的\_ucDataMode=1）*
* pParam、pParam1、pParam2，包含其他信息（例如抓取的线数、每条轮廓的点数等等）
* pOwner，同注册回调函数时的pOwner参数。

**重要声明：**回调函数是SDK中的线程回调上来的，请务必不要在回调函数里面进行耗时、等待的操作。

**注：**

1. MFC demo 中SaveCloud函数有转换数据为点云文件的实例，如果不清楚如何处理点云数据的，请自行阅读该函数代码。
2. 当使用相机使用编码器，并且软件触发批处理时，如果param->\_iCapturedProfileLineNum < param->\_iWantCaptureProfileLineNum，则需要进行如下分析处理：

2.1 在UI或者其他工作线程中调用函数SendGrabSignalToCamera(true)； 避免后续相机中数据包序号不对。

2.2 请分析一下为什么会出现这种情况。如果是人为停止了运控，则忽略。否则可能就是编码器的脉冲信号不足，设置的抓取线数偏大了，请减少抓取线数（确保设置的抓取线数小于等于实际相机的线数）

2.3 有些编码器运行时不是很稳定，特别是运动即将停止时。那么可能会出现一种现象：一次触发抓取，SDK上报了多次，并且其中有几次上报的线数很少，就几条线。 对于出现这种情况，请在业务层面直接忽略这些上报的数据。SDK 无法判断是否为异常信号导致。

## 不常用参数获取

1. 获取最大帧率 SgGetMaxFrame

注：因为最大帧率是动态可变的，SDK 也支持自动上报的方式。获取方式如下，注册一个获取的回调函数，然后在回调函数内计算得到最大帧率。（可参考MFC demo）

// 注册回调函数

SgRegCallback(m\_hCamera, CBTYPE\_REPORT\_MAX\_FRAME, SG\_MAXFRAME\_REPORT\_CBFUNC, this);

// 回调函数实现如下

void CSingleCameraSynDialog::SG\_MAXFRAME\_REPORT\_CBFUNC(unsigned short usMaxFrame, void \*pOwner)

{

}

1. 获取x方向点间距

注：x方向的点间距可能会发生改变，所以没有单独获取x方向点间距的接口。如果要获取，可以在数据上报的回调函数中得到该值SG\_DEPTHDATA\_PARAM::\_fXSpace。

某些相机支持bin功能(PanSight软件调试模式下可打开），打开x bin后， x方向点数变少，点间距变大。

## SDK使用注意事项

1. 采集数据期间（调用了SgStartCaptureSyn、SgStartCapture、SgStartCameraByType后），不能调用修改模式的接口（SgSetCamMode、SgSetMod、SgSetCamModeSyn、SgSetModSyn）! 否则可能会引起采集不到数据、数据错位等等奇怪问题。
2. 如果使用异步接口SgSetCamMode、SgSetMod，务必要在收到响应后，才能调用SgStartCaptureSyn、SgStartCapture、SgStartCameraByType函数！否则可能会引起采集不到数据或者数据错位等问题
3. 使用异步接口，调用者额外判断接口调用是否成功！ 任何接口都可能失败，调用者必须对失败情况进行处理！（例如修改相机的参数时，刚好网线被触碰，导致相关参数没能发送到相机，如果调用者不进行判断，直接进行后续操作，可能结果就不是想要的了）

## SDK 配置文件

1. 配置文件configs.ini 很重要，并且和动态库dll需要配套！如果升级动态库则必须升级对应的configs.ini （如果之前是使用灰度模式，记得要修改*supportGrayMode*）
2. 重要配置项说明

**LogLevel** ：调节SDK的日志级别。调节后软件重启生效。打印的日志在当前Logs文件夹中，如：coreLog\_\*.log

**iDataTimeout**: 深度或灰度模式时，抓取相机数据的超时时间，毫秒为单位。当一次抓取未满线数时，超过该设定的时间，数据回调自动上报。（使用编码器时，可能有些编码器信号不稳定，同样的距离产生的有效脉冲不一致，导致相机抓取线数不满。如果没有该超时机制，则回调就一直无法上报。）

**supportGrayMode**：是否支持灰度模式。如果需要使用灰度数据，在运行软件之前，请先打开此开关。