Hypersen
HYPERSEN TECHNOLOGIES CO., LTD. 海伯森技术

HPS-3D640

OpenNI2 使用手册

目录

一、SDK 说明

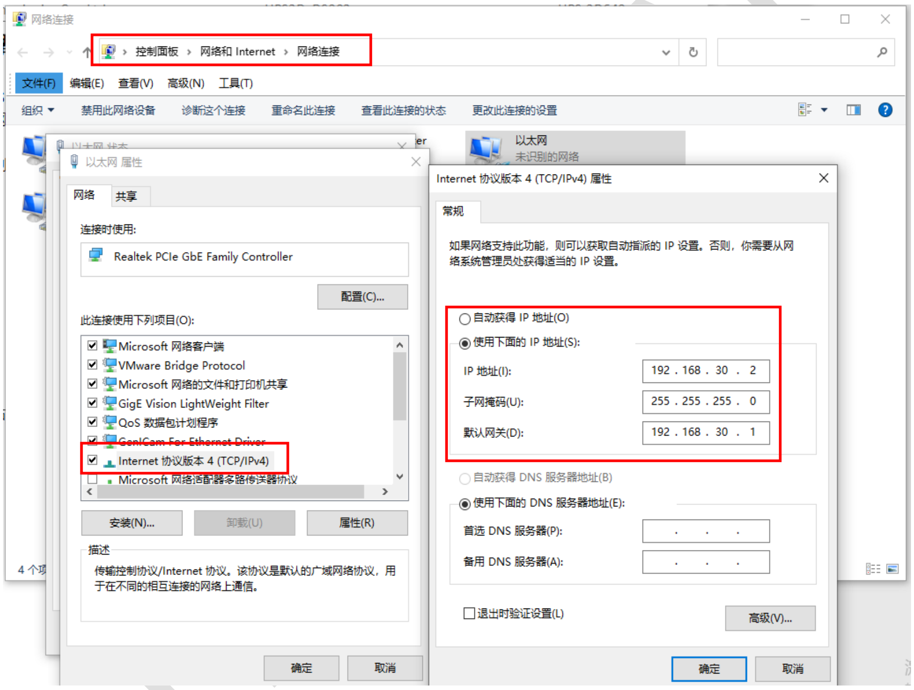
HPS-3D640 系列已适配 OpenNI2.2 以及 OpenNI2.3 库，支持 Windows、Linux 平台开发使用。

二、设备连接

HPS-3D640-L 通过 LAN 接口与主机通讯。

传感器默认 IP 地址是 192.168.30.202， 端口是 12345。

传感器网线接入主机的以太网口后，需要在主机配置 ip 地址和子网掩码，其中 IP 地址必须与传感器 IP 配置在同一网段下，如下图。


网络连接
控制面板 › 网络和 Internet › 网络连接
文件(F) 编辑(E) 查看(V) 高级(N) 工具(T)
组织 ▾ 禁用此网络设备 诊断这个连接 重命名此连接 查看此连接的状态 更改此连接的设置
以太网
未识别的网络
4 个项

以太网 属性

网络 共享

连接时使用:

Realtek PCIe GbE Family Controller

配置(C)...

此连接使用下列项目(O):

- ☑ Microsoft 网络客户端
- ☑ VMware Bridge Protocol
- ☑ Microsoft 网络的文件和打印机共享
- ☑ GigE Vision LightWeight Filter
- ☑ QoS 数据包计划程序
- ☑ GenICam For Ethernet Driver
- ☑ Internet 协议版本 4 (TCP/IPv4)
- □ Microsoft 网络适配器多路传送器协议

安装(N)... 卸载(U) 属性(R)

描述

传输控制协议/Internet 协议。该协议是默认的广域网络协议，用于在不同的相互连接的网络上通信。

确定 取消

Internet 协议版本 4 (TCP/IPv4) 属性

常规

如果网络支持此功能，则可以获取自动指派的 IP 设置。否则，你需要从网络系统管理员处获得适当的 IP 设置。

○ 自动获得 IP 地址(O)

● 使用下面的 IP 地址(S):

IP 地址(I): 192 . 168 . 30 . 2

子网掩码(U): 255 . 255 . 255 . 0

默认网关(D): 192 . 168 . 30 . 1

○ 自动获得 DNS 服务器地址(B)

● 使用下面的 DNS 服务器地址(E):

首选 DNS 服务器(P): . . .

备用 DNS 服务器(A): . . .

□ 退出时验证设置(L) 高级(V)...

确定 取消

三、将 OpenNI 集成到 IDE 中

以下是在 Microsoft Visual Studio 平台下 OpenNI 的集成。

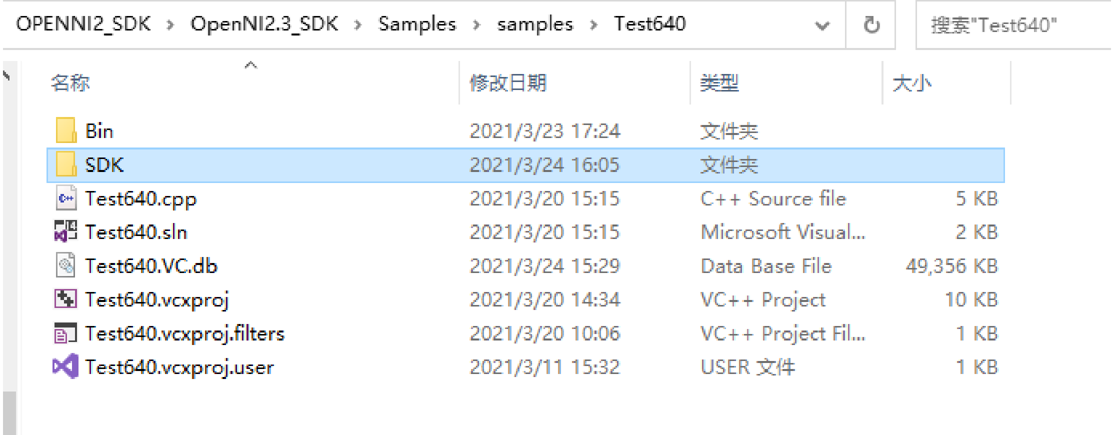
3.1 Visual Studio 平台下环境配置及集成到 IDE 中

xxx.lib 和 xxx.dll 适合在 Windows 操作系统平台使用，这里以 VS2015 环境为例。

3.1.1 工程环境配置与集成

1、添加头文件包含路径

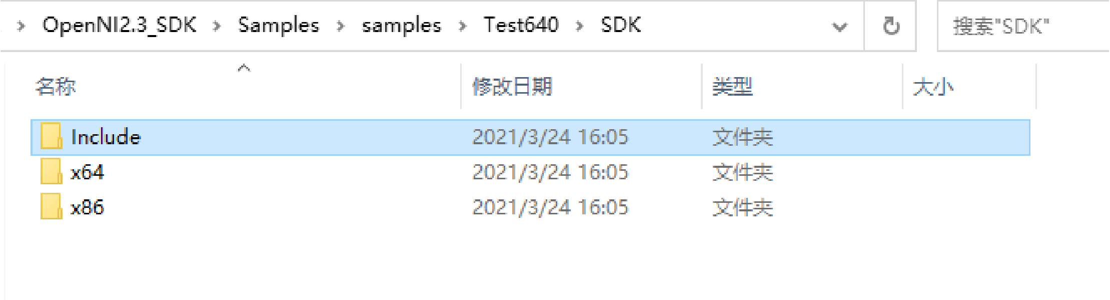
a、将 OPENI 的 SDK 拷贝到用户指定的路径下。



OPENNI2_SDK › OpenNI2.3_SDK › Samples › samples › Test640

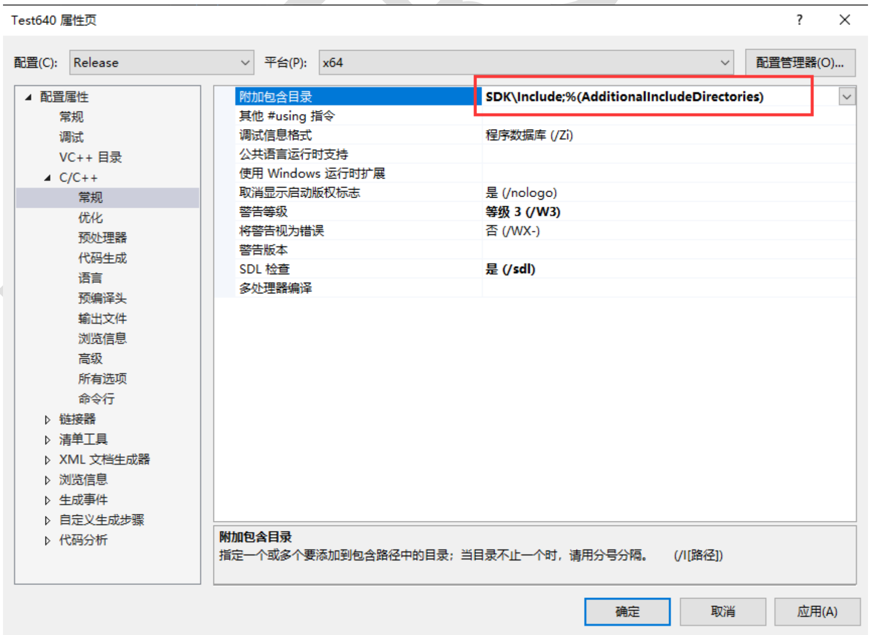
名称	修改日期	类型	大小
Bin	2021/3/23 17:24	文件夹	
SDK	2021/3/24 16:05	文件夹	
Test640.cpp	2021/3/20 15:15	C++ Source file	5 KB
Test640.sln	2021/3/20 15:15	Microsoft Visual...	2 KB
Test640.VC.db	2021/3/24 15:29	Data Base File	49,356 KB
Test640.vcxproj	2021/3/20 14:34	VC++ Project	10 KB
Test640.vcxproj.filters	2021/3/20 10:06	VC++ Project Fil...	1 KB
Test640.vcxproj.user	2021/3/11 15:32	USER 文件	1 KB

b、点击项目 - 属性 - C/C++ -常规。在附加包含目录下指定头文件的路径。（SDK 文件夹中的 Include 文件夹）



› OpenNI2.3_SDK › Samples › samples › Test640 › SDK

名称	修改日期	类型	大小
Include	2021/3/24 16:05	文件夹	
x64	2021/3/24 16:05	文件夹	
x86	2021/3/24 16:05	文件夹	



Test640 属性页

配置(C): Release 平台(P): x64

附加包含目录	SDK\Include;%(AdditionalIncludeDirectories)
其他 #using 指令	
调试信息格式	程序数据库 (/Zi)
公共语言运行时支持	
使用 Windows 运行时扩展	
取消显示启动版权标志	是 (/nologo)
警告等级	等级 3 (/W3)
将警告视为错误	否 (/WX-)
警告版本	
SDL 检查	是 (/sdl)
多处理器编译	

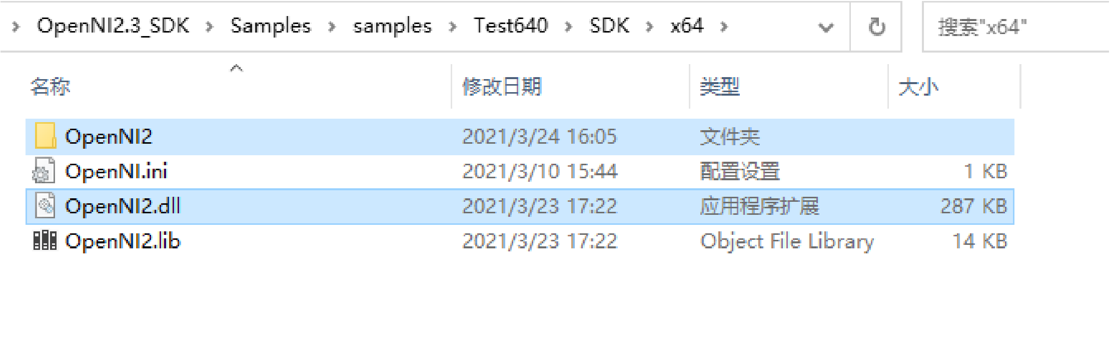
附加包含目录

指定一个或多个要添加到包含路径中的目录；当目录不止一个时，请用分号分隔。 (/I[路径])

2、添加库引用和库的路径

a、根据用户 Visual Studio 的平台，选择 x64 或 x86 目录下的 SDK 文件，这里以 x64 为例。

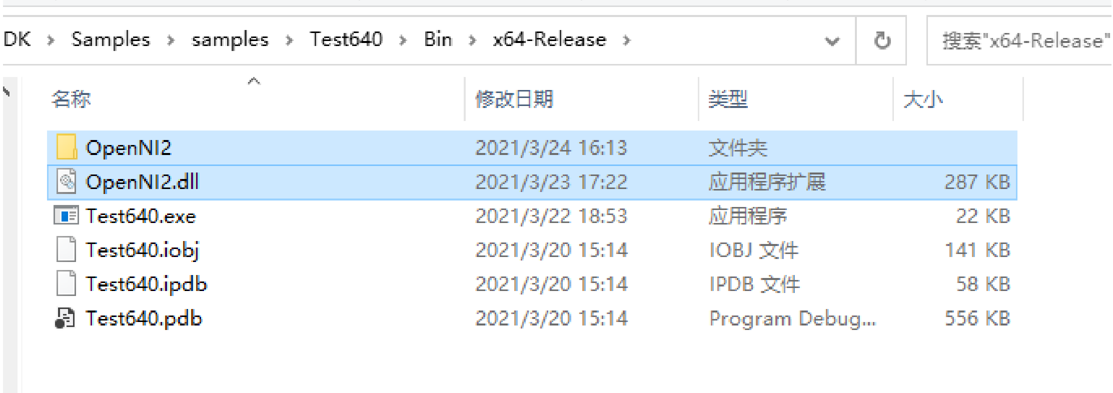
b、将 x64 目录下 OpenNI2.dll 以及 OpenNI2 文件夹拷贝到程序运行的目录下；



› OpenNI2.3_SDK › Samples › samples › Test640 › SDK › x64 ›

搜索"x64"

名称	修改日期	类型	大小
OpenNI2	2021/3/24 16:05	文件夹	
OpenNI.ini	2021/3/10 15:44	配置设置	1 KB
OpenNI2.dll	2021/3/23 17:22	应用程序扩展	287 KB
OpenNI2.lib	2021/3/23 17:22	Object File Library	14 KB

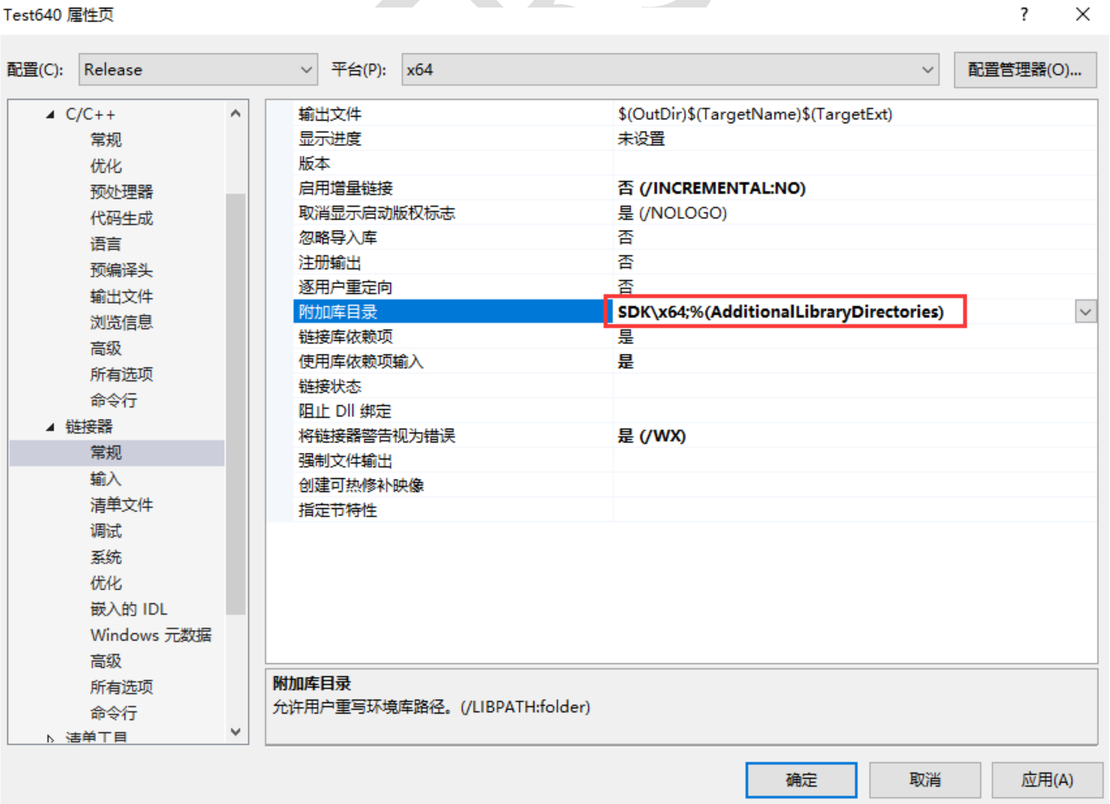


DK › Samples › samples › Test640 › Bin › x64-Release ›

搜索"x64-Release"

名称	修改日期	类型	大小
OpenNI2	2021/3/24 16:13	文件夹	
OpenNI2.dll	2021/3/23 17:22	应用程序扩展	287 KB
Test640.exe	2021/3/22 18:53	应用程序	22 KB
Test640.iobj	2021/3/20 15:14	IOBJ 文件	141 KB
Test640.ipdb	2021/3/20 15:14	IPDB 文件	58 KB
Test640.pdb	2021/3/20 15:14	Program Debug...	556 KB

c、点击项目 - 属性 - 链接器 - 常规。在附加库目录中指定 OpenNI2.lib 的路径。



Test640 属性页

配置(C): Release 平台(P): x64 配置管理器(O)...

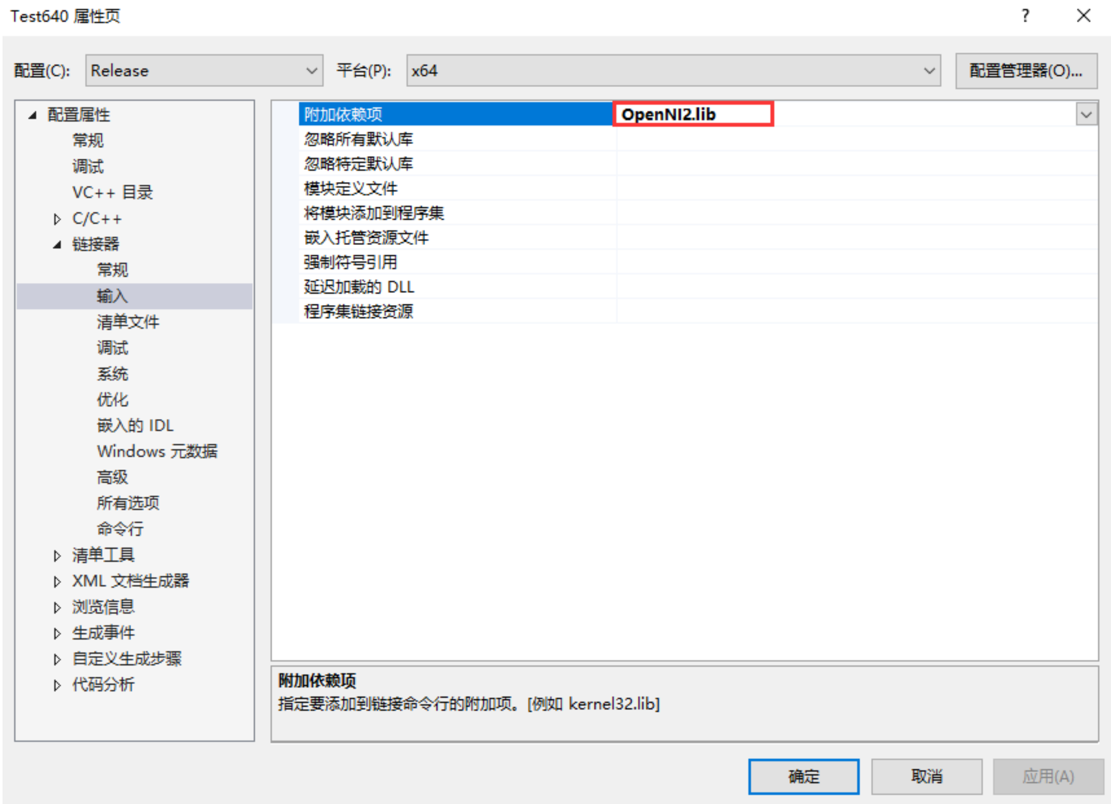
- C/C++
  - 常规
  - 优化
  - 预处理器
  - 代码生成
  - 语言
  - 预编译头
  - 输出文件
  - 浏览信息
  - 高级
  - 所有选项
  - 命令行
- 链接器
  - 常规
  - 输入
  - 清单文件
  - 调试
  - 系统
  - 优化
  - 嵌入的 IDL
  - Windows 元数据
  - 高级
  - 所有选项
  - 命令行
- 清单工具

输出文件	$(OutDir)$(TargetName)$(TargetExt)
显示进度	未设置
版本	
启用增量链接	否 (/INCREMENTAL:NO)
取消显示启动版权标志	是 (/NOLOGO)
忽略导入库	否
注册输出	否
逐用户重定向	否
附加库目录	SDK\x64;%(AdditionalLibraryDirectories)
链接库依赖项	是
使用库依赖项输入	是
链接状态	
阻止 Dll 绑定	
将链接器警告视为错误	是 (/WX)
强制文件输出	
创建可热修补映像	
指定节特性	

附加库目录

允许用户重写环境库路径。(/LIBPATH:folder)

d、点击项目 - 属性 - 链接器 – 输入。在附加依赖项中加入 OpenNI2.lib。


Test640 属性页

配置(C): Release   平台(P): x64

附加依赖项: OpenNI2.lib

附加依赖项
指定要添加到链接命令行的附加项。[例如 kernel32.lib]

3.1.2 在用户项目中使用 OpenNI2

以下为获取传感器数据的简单步骤，更具体的请参考 OpenNI_SDK 中的示例代码。

1、 包含头文件

```
#include <OpenNI.h>
#include <conio.h> //非必须
```

2、 初始化 OpenNI 的 API

```
//初始化OpenNI
Status nRetVal = openni::OpenNI::initialize();
if (nRetVal != STATUS_OK)
{
    return 1;
}
```

3、 打开设备

```
Device device;
//使用URI打开设备，URI格式IP=x,PORT=x(x根据设备参数设置)
nRetVal = device.open("IP=192.168.30.202,PORT=12345");
if (nRetVal != STATUS_OK)
{
    printf("Device open failed! %s\n", OpenNI::getExtendedError());
    OpenNI::shutdown();
    return 2;
}
```

4、 创建流

```
//创建流
VideoStream depth;
//判断设备是否支持深度流
if (device.getSensorInfo(SENSOR_DEPTH) != NULL)
{
    nRetVal = depth.create(device, SENSOR_DEPTH);
    if (nRetVal != STATUS_OK)
    {
        printf("Couldn't create depth stream! %s\n", OpenNI::getExtendedError());
        return 3;
    }
}
```

注意：HPS-3D640 只支持创建深度流

5、 启动深度流开始输出数据

```
//开始输出深度数据
nRetVal = depth.start();
if (nRetVal != STATUS_OK)
{
    printf("Couldn't start the depth stream! %s\n", OpenNI::getExtendedError());
    return 4;
}
```

6、 获取深度流中的数据

```
//创建单帧
VideoFrameRef frame;
//敲击键盘退出
while (!_kbhit())
{
    int ReadyStreamIndex;
    VideoStream* pStream = &depth;
    //等待当前流的一帧数据
    nRetVal = OpenNI::waitForAnyStream(&pStream, 1, &ReadyStreamIndex, 300);
    if (nRetVal != STATUS_OK)
    {
        printf("Wait Time Out! %s\n", OpenNI::getExtendedError());
        continue;
    }
    //读取当前流中的一帧数据
    nRetVal = depth.readFrame(&frame);
    if (nRetVal != STATUS_OK)
    {
        printf("Read failed! %s\n", OpenNI::getExtendedError());
        continue;
    }
    //获取当前帧的数据
    DepthPixel* depthData = (DepthPixel*)frame.getData();
    //打印中心点数据
    printf("Timestamp:%d depthData:%d\n", frame.getTimestamp(),
depthData[307200/2]);
}
```

7、运行结束，释放资源

```
//停止流
depth.stop();
//销毁流
depth.destroy();
//关闭设备
device.close();
//卸载驱动
openni::OpenNI::shutdown();
return openni::STATUS_OK;
```

四、 通过 PCL 框架调用 OpenNI 并集成到 IDE 中

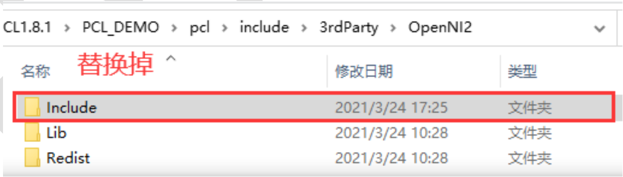
4.1 Visual Studio 平台下环境配置及集成到 IDE 中

以 VS2015 环境为例。

4.1.1 工程环境配置与集成

1、将 PCL 第三方库中的 OpenNI2 替换成 Hypersen 提供的 OpenNI2 中的内容

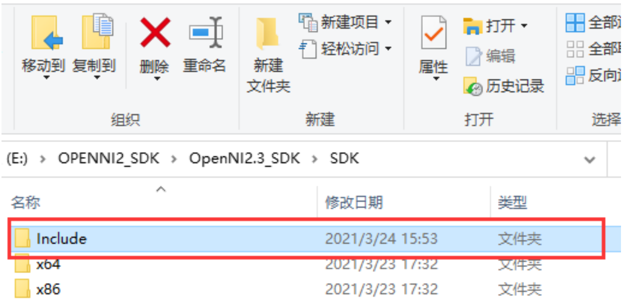
a、替换 Include 中的所有文件。



CL1.8.1 › PCL_DEMO › pcl › include › 3rdParty › OpenNI2

替换掉

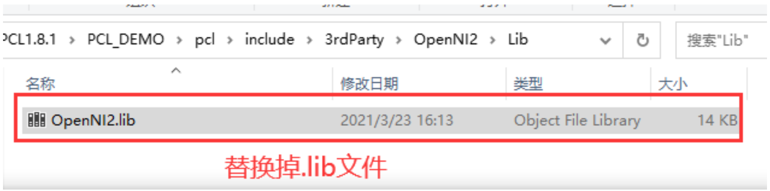
名称	修改日期	类型
Include	2021/3/24 17:25	文件夹
Lib	2021/3/24 10:28	文件夹
Redist	2021/3/24 10:28	文件夹



(E:) › OPENNI2_SDK › OpenNI2.3_SDK › SDK

名称	修改日期	类型
Include	2021/3/24 15:53	文件夹
x64	2021/3/23 17:32	文件夹
x86	2021/3/23 17:32	文件夹

b、替换.lib 文件。

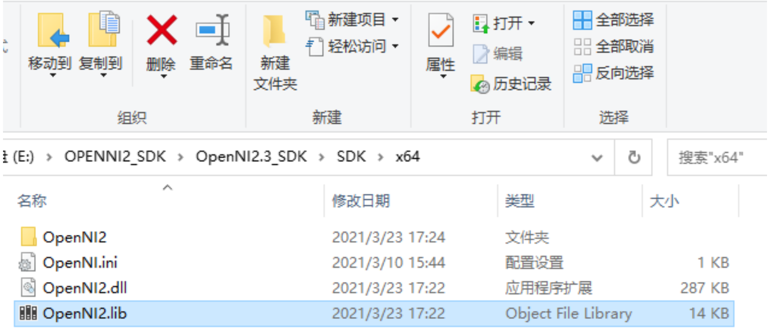


PCL1.8.1 › PCL_DEMO › pcl › include › 3rdParty › OpenNI2 › Lib

搜索"Lib"

名称	修改日期	类型	大小
OpenNI2.lib	2021/3/23 16:13	Object File Library	14 KB

替换掉.lib文件

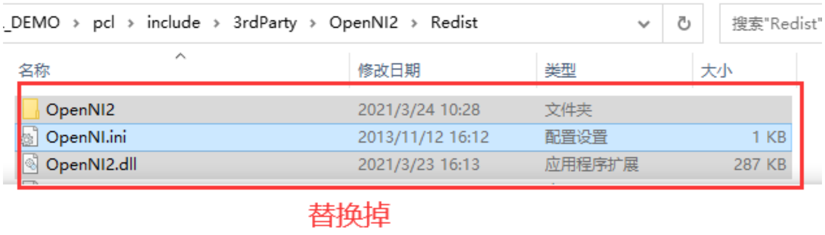


(E:) › OPENNI2_SDK › OpenNI2.3_SDK › SDK › x64

搜索"x64"

名称	修改日期	类型	大小
OpenNI2	2021/3/23 17:24	文件夹	
OpenNI.ini	2021/3/10 15:44	配置设置	1 KB
OpenNI2.dll	2021/3/23 17:22	应用程序扩展	287 KB
OpenNI2.lib	2021/3/23 17:22	Object File Library	14 KB

c、替换以下文件

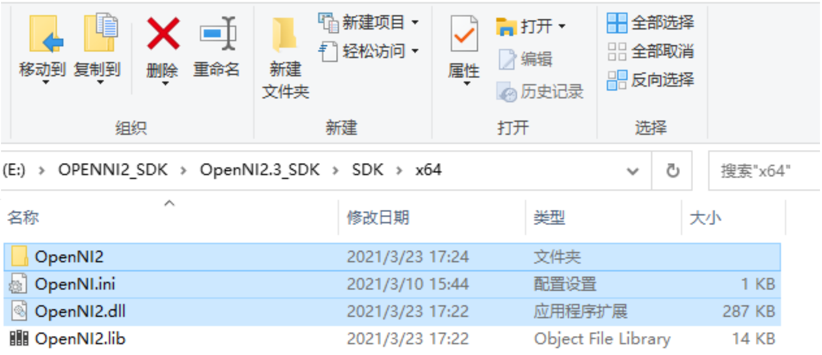


_DEMO › pcl › include › 3rdParty › OpenNI2 › Redist

搜索"Redist"

名称	修改日期	类型	大小
OpenNI2	2021/3/24 10:28	文件夹	
OpenNI.ini	2013/11/12 16:12	配置设置	1 KB
OpenNI2.dll	2021/3/23 16:13	应用程序扩展	287 KB

替换掉

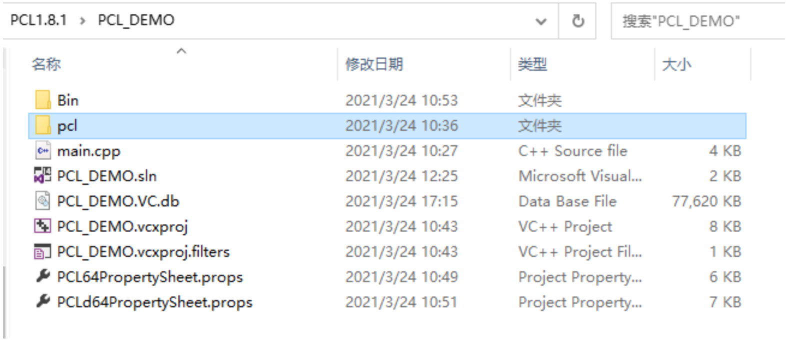


(E:) › OPENNI2_SDK › OpenNI2.3_SDK › SDK › x64

搜索"x64"

名称	修改日期	类型	大小
OpenNI2	2021/3/23 17:24	文件夹	
OpenNI.ini	2021/3/10 15:44	配置设置	1 KB
OpenNI2.dll	2021/3/23 17:22	应用程序扩展	287 KB
OpenNI2.lib	2021/3/23 17:22	Object File Library	14 KB

2、将 PCL 库拷贝到项目文件中

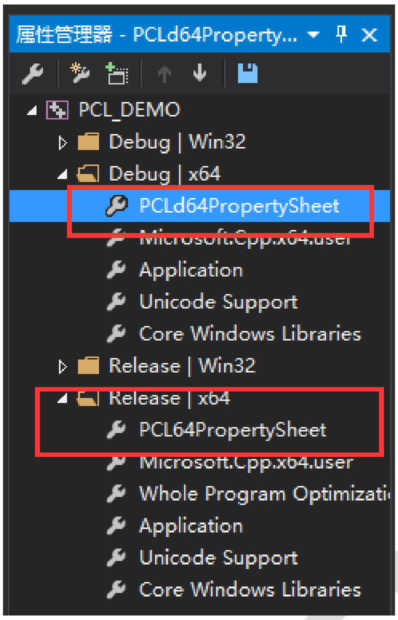


PCL1.8.1 › PCL_DEMO

搜索"PCL_DEMO"

名称	修改日期	类型	大小
Bin	2021/3/24 10:53	文件夹	
pcl	2021/3/24 10:36	文件夹	
main.cpp	2021/3/24 10:27	C++ Source file	4 KB
PCL_DEMO.sln	2021/3/24 12:25	Microsoft Visual...	2 KB
PCL_DEMO.VC.db	2021/3/24 17:15	Data Base File	77,620 KB
PCL_DEMO.vcxproj	2021/3/24 10:43	VC++ Project	8 KB
PCL_DEMO.vcxproj.filters	2021/3/24 10:43	VC++ Project Fil...	1 KB
PCL64PropertySheet.props	2021/3/24 10:49	Project Property...	6 KB
PCLd64PropertySheet.props	2021/3/24 10:51	Project Property...	7 KB

3、将 PCLd64PropertySheet 和 PCL64ProPertySheet 加入属性管理器中

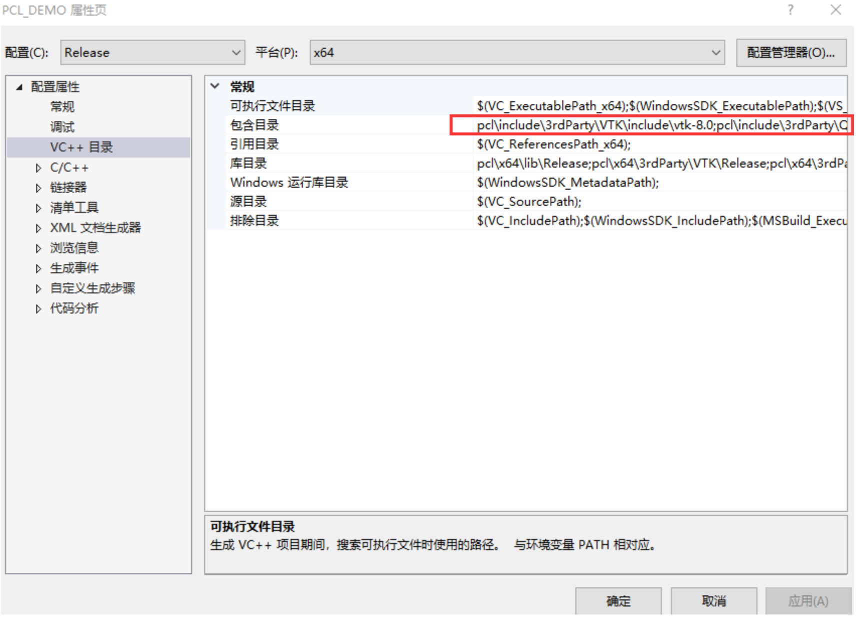


这两个文件已经配置好了依赖环境，直接导入到项目即可，若没有这里文件，则需要参照以下步骤自行配置环境。

4、设置库环境

a、点击项目 - 属性 - VC++目录。在包含目录下添加以下路径：

```
pcl\include\3rdParty\VTK\include\vtk-8.0
pcl\include\3rdParty\Qhull\include
pcl\include\3rdParty\OpenNI2\Include
pcl\include\3rdParty\FLANN\include\flann
pcl\include\3rdParty\FLANN\include
pcl\include\3rdParty\Eigen\eigen3
pcl\include\3rdParty\Boost\include\boost-1_64
pcl\include
pcl\include\pcl
```



配置(C): Release 平台(P): x64

常规

可执行文件目录 $(VC_ExecutablePath_x64);$(WindowsSDK_ExecutablePath);$(VS_

包含目录 pcl\include\3rdParty\VTK\include\vtk-8.0;pcl\include\3rdParty\C

引用目录 $(VC_ReferencesPath_x64);

库目录 pcl\x64\lib\Release;pcl\x64\3rdParty\VTK\Release;pcl\x64\3rdP

Windows 运行库目录 $(WindowsSDK_MetadataPath);

源目录 $(VC_SourcePath);

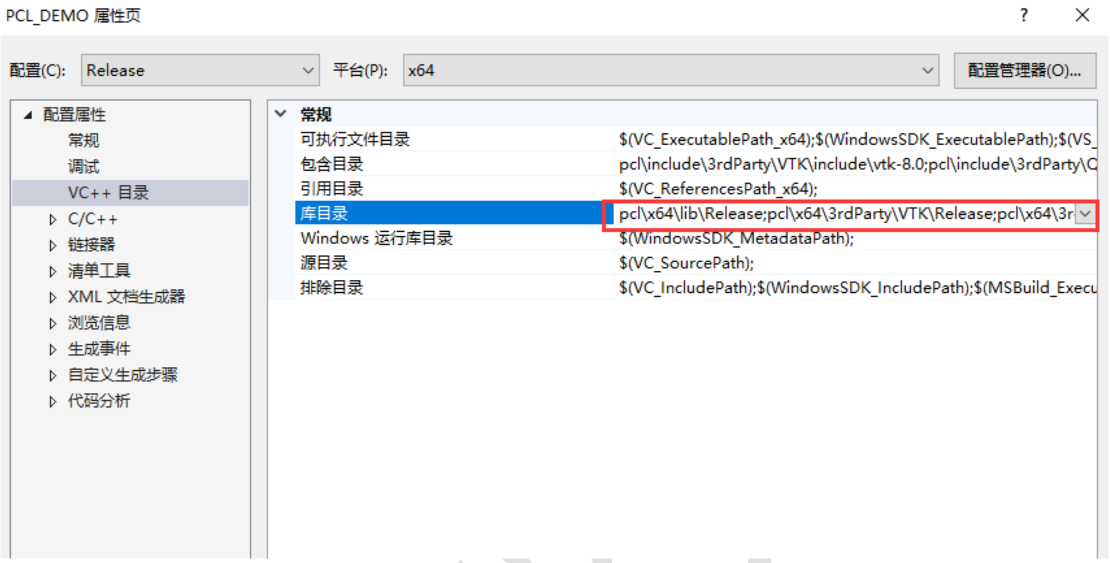
排除目录 $(VC_IncludePath);$(WindowsSDK_IncludePath);$(MSBuild_Execu

可执行文件目录

生成 VC++ 项目期间，搜索可执行文件时使用的路径。 与环境变量 PATH 相对应。

b、点击项目 - 属性 - 库目录。在库目录下添加以下路径：

```
pcl\x64\lib\Release
pcl\x64\3rdParty\VTK\Release
pcl\x64\3rdParty\Qhull\lib\Release
pcl\x64\3rdParty\OpenNI2\Lib
pcl\x64\3rdParty\FLANN\lib\Release
pcl\x64\3rdParty\Boost\lib\Release
```



配置(C): Release
平台(P): x64

可执行文件目录	$(VC_ExecutablePath_x64);$(WindowsSDK_ExecutablePath);$(VS_
包含目录	pcl\include\3rdParty\VTK\include\vtk-8.0;pcl\include\3rdParty\C
引用目录	$(VC_ReferencesPath_x64);
库目录	pcl\x64\lib\Release;pcl\x64\3rdParty\VTK\Release;pcl\x64\3r
Windows 运行库目录	$(WindowsSDK_MetadataPath);
源目录	$(VC_SourcePath);
排除目录	$(VC_IncludePath);$(WindowsSDK_IncludePath);$(MSBuild_Execu

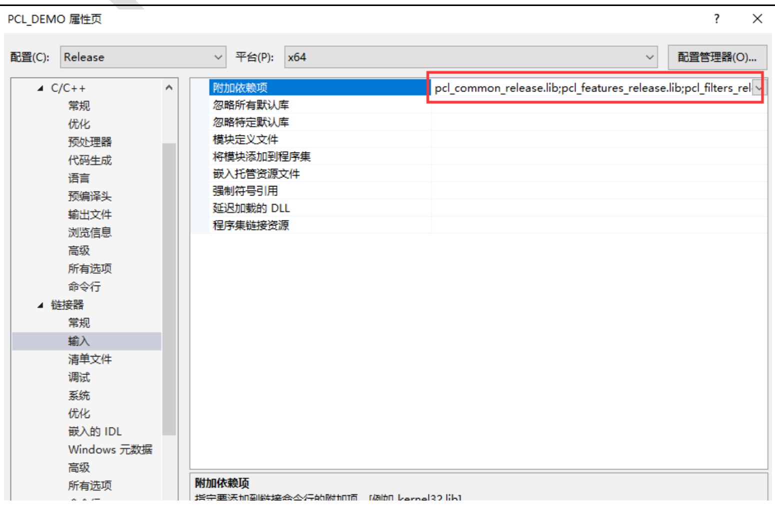
d、点击项目 - 属性 - 链接器 - 输入。在附加依赖选项中填入以下内容。

```
pcl_common_release.lib
pcl_features_release.lib
pcl_filters_release.lib
pcl_io_release.lib
pcl_io_ply_release.lib
pcl_kdtree_release.lib
pcl_keypoints_release.lib
pcl_ml_release.lib
pcl_octree_release.lib
pcl_outofcore_release.lib
pcl_people_release.lib
pcl_recognition_release.lib
pcl_registration_release.lib
pcl_sample_consensus_release.lib
pcl_search_release.lib
pcl_segmentation_release.lib
pcl_stereo_release.lib
pcl_surface_release.lib
pcl_tracking_release.lib
pcl_visualization_release.lib
flann_cpp_s.lib
flann_cpp.lib
flann_s.lib
flann.lib
libboost_bzip2-vc140-mt-1_64.lib
libboost_atomic-vc140-mt-1_64.lib
libboost_chrono-vc140-mt-1_64.lib
libboost_container-vc140-mt-1_64.lib
libboost_context-vc140-mt-1_64.lib
```

```
libboost_coroutine-vc140-mt-1_64.lib
libboost_date_time-vc140-mt-1_64.lib
libboost_exception-vc140-mt-1_64.lib
libboost_fiber-vc140-mt-1_64.lib
libboost_graph_parallel-vc140-mt-1_64.lib
libboost_filesystem-vc140-mt-1_64.lib
libboost_graph-vc140-mt-1_64.lib
libboost_iostreams-vc140-mt-1_64.lib
libboost_locale-vc140-mt-1_64.lib
libboost_log-vc140-mt-1_64.lib
libboost_log_setup-vc140-mt-1_64.lib
libboost_math_c99-vc140-mt-1_64.lib
libboost_math_c99f-vc140-mt-1_64.lib
libboost_math_c99l-vc140-mt-1_64.lib
libboost_math_tr1-vc140-mt-1_64.lib
libboost_math_tr1f-vc140-mt-1_64.lib
libboost_math_tr1l-vc140-mt-1_64.lib
libboost_numpy3-vc140-mt-1_64.lib
libboost_numpy-vc140-mt-1_64.lib
libboost_mpi-vc140-mt-1_64.lib
libboost_prg_exec_monitor-vc140-mt-1_64.lib
libboost_program_options-vc140-mt-1_64.lib
libboost_python3-vc140-mt-1_64.lib
libboost_python-vc140-mt-1_64.lib
libboost_random-vc140-mt-1_64.lib
libboost_regex-vc140-mt-1_64.lib
libboost_serialization-vc140-mt-1_64.lib
libboost_signals-vc140-mt-1_64.lib
libboost_system-vc140-mt-1_64.lib
libboost_type_erasure-vc140-mt-1_64.lib
libboost_zlib-vc140-mt-1_64.lib
libboost_test_exec_monitor-vc140-mt-1_64.lib
libboost_thread-vc140-mt-1_64.lib
libboost_timer-vc140-mt-1_64.lib
libboost_unit_test_framework-vc140-mt-1_64.lib
libboost_wave-vc140-mt-1_64.lib
libboost_wserialization-vc140-mt-1_64.lib
qhullstatic.lib
qhull.lib
qhull_p.lib
qhull_r.lib
qhullcpp.lib
qhullstatic_r.lib
vtkFiltersPoints-8.0.lib
vtkFiltersTopology-8.0.lib
vtkgl2ps-8.0.lib
vtkIOExportOpenGL-8.0.lib
vtkIOParallelXML-8.0.lib
vtkIOTecplotTable-8.0.lib
vtklibharu-8.0.lib
vtklz4-8.0.lib
vtkRenderingContextOpenGL-8.0.lib
vtkRenderingGL2PS-8.0.lib
vtkRenderingLIC-8.0.lib
vtkRenderingOpenGL-8.0.lib
vtkRenderingVolumeOpenGL-8.0.lib
vtkalglib-8.0.lib
vtkChartsCore-8.0.lib
```

```
vtkCommonColor-8.0.lib
vtkCommonComputationalGeometry-8.0.lib
vtkCommonCore-8.0.lib
vtkCommonDataModel-8.0.lib
vtkCommonExecutionModel-8.0.lib
vtkCommonMath-8.0.lib
vtkCommonMisc-8.0.lib
vtkCommonSystem-8.0.lib
vtkCommonTransforms-8.0.lib
vtkDICOMParser-8.0.lib
vtkDomainsChemistry-8.0.lib
vtkexoIIc-8.0.lib
vtkexpat-8.0.lib
vtkFiltersAMR-8.0.lib
vtkFiltersCore-8.0.lib
vtkFiltersExtraction-8.0.lib
vtkFiltersFlowPaths-8.0.lib
vtkFiltersGeneral-8.0.lib
vtkFiltersGeneric-8.0.lib
vtkFiltersGeometry-8.0.lib
vtkFiltersHybrid-8.0.lib
vtkFiltersHyperTree-8.0.lib
vtkFiltersImaging-8.0.lib
vtkFiltersModeling-8.0.lib
vtkFiltersParallel-8.0.lib
vtkFiltersParallelImaging-8.0.lib
vtkFiltersProgrammable-8.0.lib
vtkFiltersSelection-8.0.lib
vtkFiltersSMP-8.0.lib
vtkFiltersSources-8.0.lib
vtkFiltersStatistics-8.0.lib
vtkFiltersTexture-8.0.lib
vtkFiltersVerdict-8.0.lib
vtkfreetype-8.0.lib
vtkGeovisCore-8.0.lib
vtkhdf5-8.0.lib
vtkhdf5_hl-8.0.lib
vtkImagingColor-8.0.lib
vtkImagingCore-8.0.lib
vtkImagingFourier-8.0.lib
vtkImagingGeneral-8.0.lib
vtkImagingHybrid-8.0.lib
vtkImagingMath-8.0.lib
vtkImagingMorphological-8.0.lib
vtkImagingSources-8.0.lib
vtkImagingStatistics-8.0.lib
vtkImagingStencil-8.0.lib
vtkInfovisCore-8.0.lib
vtkInfovisLayout-8.0.lib
vtkInteractionImage-8.0.lib
vtkInteractionStyle-8.0.lib
vtkInteractionWidgets-8.0.lib
vtkIOAMR-8.0.lib
vtkIOCore-8.0.lib
vtkIOEnSight-8.0.lib
vtkIOExodus-8.0.lib
vtkIOExport-8.0.lib
vtkIOGeometry-8.0.lib
```

```
vtkIOImage-8.0.lib
vtkIOImport-8.0.lib
vtkIOInfovis-8.0.lib
vtkIOLegacy-8.0.lib
vtkIOLSDyna-8.0.lib
vtkIOMINC-8.0.lib
vtkIOMovie-8.0.lib
vtkIONetCDF-8.0.lib
vtkIOParallel-8.0.lib
vtkIOPLY-8.0.lib
vtkIOSQL-8.0.lib
vtkIOVideo-8.0.lib
vtkIOXML-8.0.lib
vtkIOXMLParser-8.0.lib
vtkjpeg-8.0.lib
vtkjsoncpp-8.0.lib
vtklibxml2-8.0.lib
vtkmetaio-8.0.lib
vtkNetCDF-8.0.lib
vtknetcdf_c++.lib
vtkoggtheora-8.0.lib
vtkParallelCore-8.0.lib
vtkpng-8.0.lib
vtkproj4-8.0.lib
vtkRenderingAnnotation-8.0.lib
vtkRenderingContext2D-8.0.lib
vtkRenderingCore-8.0.lib
vtkRenderingFreeType-8.0.lib
vtkRenderingImage-8.0.lib
vtkRenderingLabel-8.0.lib
vtkRenderingLOD-8.0.lib
vtkRenderingVolume-8.0.lib
vtksqlite-8.0.lib
vtksys-8.0.lib
vtktiff-8.0.lib
vtkverdict-8.0.lib
vtkViewsContext2D-8.0.lib
vtkViewsCore-8.0.lib
vtkViewsInfovis-8.0.lib
vtkzlib-8.0.lib
OpenNI2.lib
```



PCL_DEMO 属性页

配置(C): Release

平台(P): x64

附加依赖项: pcl_common_release.lib;pcl_features_release.lib;pcl_filters_rel

忽略所有默认库
忽略特定默认库
模块定义文件
将模块添加到程序集
嵌入托管资源文件
强制符号引用
延迟加载的 DLL
程序集链接资源

附加依赖项
指定要添加到链接命令行的附加项。[例如 kernel32.lib]

4.1.2 在用户项目中使用 PCL

以下为获取传感器数据并转换成点云的步骤，更具体的请参考 DOME 中的示例代码。

1、包含头文件

```
#include <pcl/common/common_headers.h>
#include <pcl/visualization/pcl_visualizer.h>
#include <iostream>
#include <OpenNI.h>
```

2、声明全局变量及宏定义

```
#define HPS_RESOLUTION_X_640 640
#define HPS_RESOLUTION_Y_480 480

openni::Device mDevice;
openni::VideoStream mDepth;
```

3、初始化 OpenNI 中的 API

```
//初始化OpenNI2中的API
openni::Status nRetVal = openni::OpenNI::initialize();
if (nRetVal != openni::STATUS_OK) {
    std::cerr << "OpenNI Initial Error: " << openni::OpenNI::getExtendedError() <<
std::endl;
    return -1;
}
```

4、使用 URI 打开设备

```
//使用URI打开设备，URI格式IP=x,PORT=x(x根据设备参数设置)
nRetVal = mDevice.open("IP=192.168.30.202,PORT=12345");
if (nRetVal != openni::STATUS_OK) {
    std::cerr << "Can't Open Device: " << openni::OpenNI::getExtendedError() <<
std::endl;
    return -1;
}
```

5、创建深度流

```
//判断设备是否支持深度流
if (mDevice.getSensorInfo(openni::SENSOR_DEPTH) != NULL)
{
    //关联设备
    nRetVal = mDepth.create(mDevice, openni::SENSOR_DEPTH);
    if (nRetVal != openni::STATUS_OK)
    {
    std::cerr << "Can't create depth stream: " << openni::OpenNI::getExtendedError()
<< std::endl;
    return -1;
    }
    //开启深度流
    mDepth.start();
    }
else {
    std::cerr << "ERROR: This device does not have depth sensor" << std::endl;
    return -1;
}
```

6、创建 PCL 点云对象

```
//创建pcl云
pcl::PointCloud<pcl::PointXYZ>::Ptr point_XYZ(new pcl::PointCloud<pcl::PointXYZ>());
point_XYZ->width = HPS_RESOLUTION_X_640;
point_XYZ->height = HPS_RESOLUTION_Y_480;
point_XYZ->points.resize(point_XYZ->width*point_XYZ->height);
```

7、 PCL 可视化

```
//pcl可视化
pcl::visualization::PCLVisualizer::Ptr m_pViewer(new
pcl::visualization::PCLVisualizer("Viewer"));
m_pViewer->setCameraPosition(0, 0, -2, 0, -1, 0, 0);
m_pViewer->addCoordinateSystem(0.3);
while (!m_pViewer->wasStopped())
{
    Depth2PointCloud(point_XYZ); //将深度数据转还成点云数据
    m_pViewer->addPointCloud<pcl::PointXYZ>(point_XYZ, "cloud");
    m_pViewer->spinOnce();
    m_pViewer->removeAllPointClouds();
}
```

其中，将深度数据转换成点云数据的示例如下：

```
//将深度数据转换成点云数据
bool Depth2PointCloud(pcl::PointCloud<pcl::PointXYZ>::Ptr cloud_XYZRGB)
{
    openni::VideoFrameRef frame;
    //获取一帧数据
    if (mDepth.readFrame(&frame) == openni::STATUS_OK)
    {
        float fx, fy, fz;
        int index = 0;
        openni::DepthPixel *pDepth = (openni::DepthPixel*)frame.getData();
        for (int y = 0; y < frame.getHeight(); y++)
        {
            for (int x = 0; x < frame.getWidth(); x++)
            {
                openni::CoordinateConverter::convertDepthToWorld(mDepth, x +
frame.getCropOriginX(), y + frame.getCropOriginY(), pDepth[index], &fx, &fy, &fz);
                //将无效数据过滤
                if (fz >= 60000)
                {
                    cloud_XYZRGB->points[index].x = 0;
                    cloud_XYZRGB->points[index].y = 0;
                    cloud_XYZRGB->points[index].z = 0;
                }
                else
                {
                    cloud_XYZRGB->points[index].x = fx / 1000; //将单位从MM转换成M
                    cloud_XYZRGB->points[index].y = fy / 1000;
                    cloud_XYZRGB->points[index].z = fz / 1000;
                }
                index++;
            }
        }
        return true;
    }
    else {
        std::cout << "Depth2PointCloud: fail to read frame from depth stream" <<
std::endl;
        return false;
    }
}
```

8、 运行结束，释放资源

```
//停止流
mDepth.stop();
//销毁流
mDepth.destroy();
//关闭设备
mDevice.close();
//卸载OPENNI驱动
openni::OpenNI::shutdown();
return 0;
```

五、修订历史记录

Date	Revision	Description
2021/04/09	1.0.0	初始版本。

Note:
The OpenNI2 SDK is available, please contact sales@hypersen.com for more information.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Hypersen Technologies Co., Ltd. reserve the right to make changes, corrections, enhancements, modifications, and improvements to Hypersen products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on Hypersen products before placing orders. Hypersen products are sold pursuant to Hypersen's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of Hypersen products and Hypersen assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by Hypersen herein.

Resale of Hypersen products with provisions different from the information set forth herein shall void any warranty granted by Hypersen for such product.

Hypersen and the Hypersen logo are trademarks of Hypersen. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.