0 资料汇总

官网：https://pointclouds.org/

知乎 -> PCL(Point Cloud Library)学习指南&资料推荐（2021版）：https://zhuanlan.zhihu.com/p/268524083

语雀 -> PCL(Point Cloud Library)学习记录（2021）：https://www.yuque.com/huangzhongqing/pcl

1 PCL介绍

1.1 PCL简介

- 点云相关研究基础上建立起来的大型跨平台开源C++ 编程库
- 涉及点云获取、滤波、分割、配准、检索、特征提取、识别 、追踪、曲面重建、可视化等，
- 做SLAM、激光雷达LOAM常用的一个库
- 支持多种操作系统，可在Windows 、Linux:， Android 、MacOS 、部分嵌入式实时系统上运行.
- OpenCV 是2 D信息获取与处理的结晶，那么PCL 就在3 D 信且获取与处理上具有同等地位.PCL 是BSD 握权方式，可以免费进行商业和学术应用.

应用领域

机器人领域

- 深度信息结合2D信息的应用研究。深度信息的引人能够使机器人
  更好地时环境进行认知、辨识，与图像信息在机器人领域的应用一样， 需要强大智能算法支撑，PCL就为此而生。

CAD /CAM 、逆向工程

- 目前在CAD/CAM 领域利用激光点云进行商精度测量与重建成为趋势
- 逆向工程技术能够对产品实物进行测绘，重构产品表面三维几何模型， 生成产品制造所需的数字化文档

激光遥感测量

- 激光遥感测量系统是目前最先进的实时获取地形表面三维空间信息和影像的遥感系统，但现在算法结果和实际结果差别较大，PCL中的模块有助于解决此问题。

虚拟现实、人机交

- PCL是基于RGBD设备的虚拟现实和人机交互应用生态链中最重要的一个环节。

1.2 PCL各模块功能

1.Filters

2.Features

3.Keypoints

4.Registration

5.Kd-tree

6.Octree

7.Segmentation

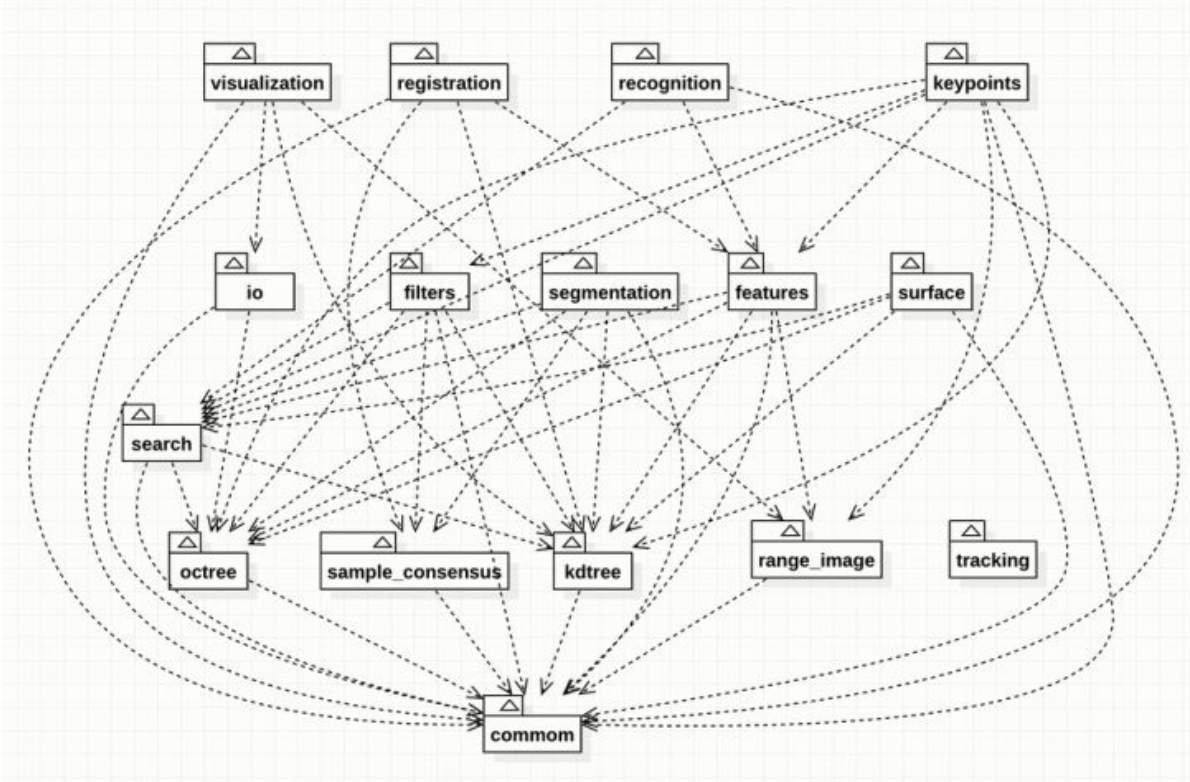
8.Sample Consensus

9.Surface

10.Range Image

11.I/O

12.Visualization



这是一个b.....站up画的图

参考博客：https://blog.csdn.net/yfy1127yfy/article/details/88943001

2 PCL库的安装编译

2.1 PCL在win10下安装编译与配置（测试）

///推荐CMake方式///

2.1.1PCL的安装与配置（vs2015+PCL1.8.1）

1. 下载PCL1.8.1，PCL官网自从1.6后的版本都托管到了github上，附上下载地址：http://unanancyowen.com/en/pcl181，这里选择对应版本

Visual Studio 2017

- **PCL 1.8.1 All-in-one Installer MSVC2017 Win32**
- PCL 1.8.1 PDB MSVC2017 Win32 (symbol files)

- **PCL 1.8.1 All-in-one Installer MSVC2017 x64**
- PCL 1.8.1 PDB MSVC2017 x64 (symbol files)

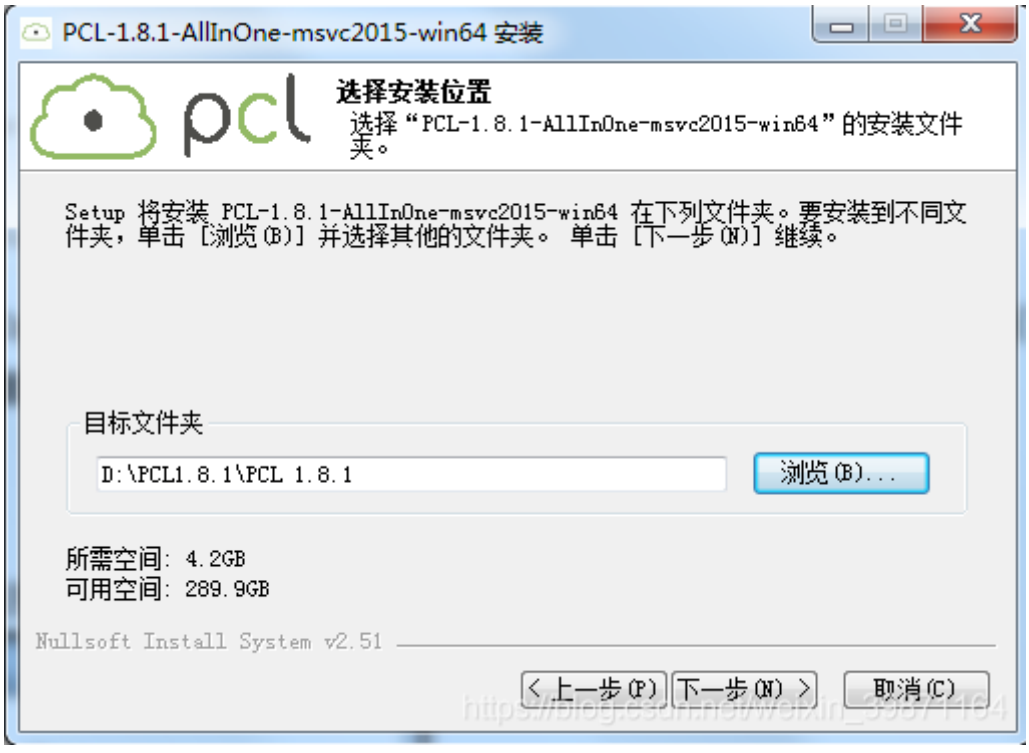
Visual Studio 2015

- **PCL 1.8.1 All-in-one Installer MSVC2015 Win32**
- PCL 1.8.1 PDB MSVC2015 Win32 (symbol files)

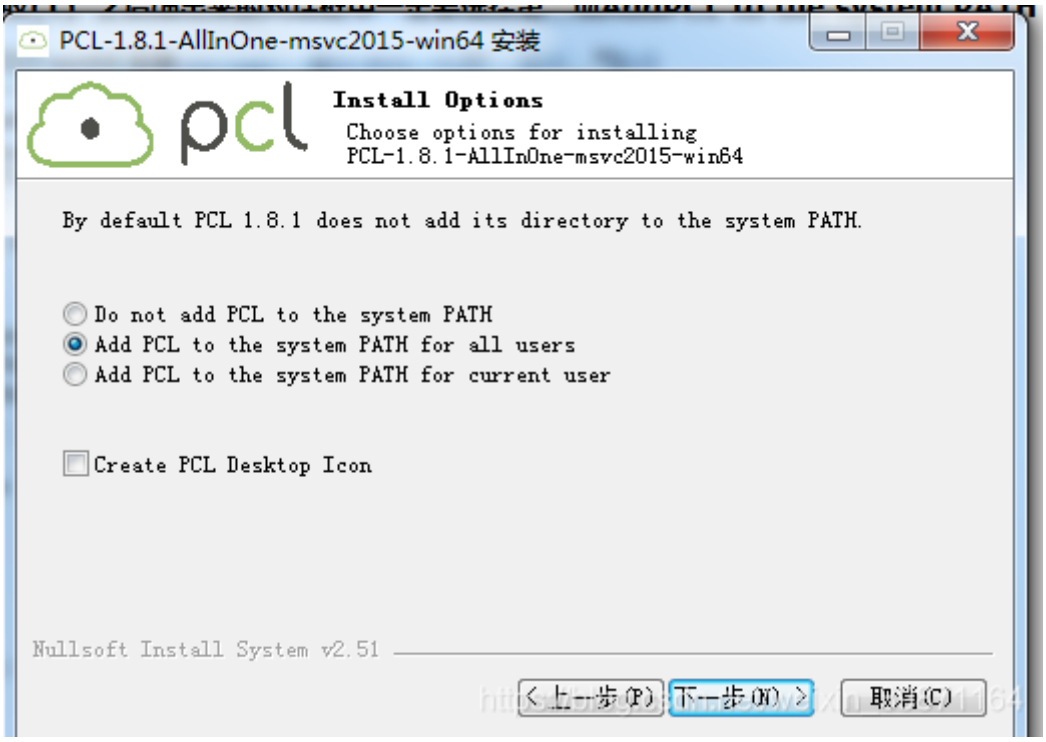
- **PCL 1.8.1 All-in-one Installer MSVC2015 x64**
- PCL 1.8.1 PDB MSVC2015 x64 (symbol files)

注意：这里提供的是All-in-one版本的安装文件，以及下面的PDB文件（调试文件）都下载下来，根据自己电脑的系统位数来安装，应该一般都是64位的

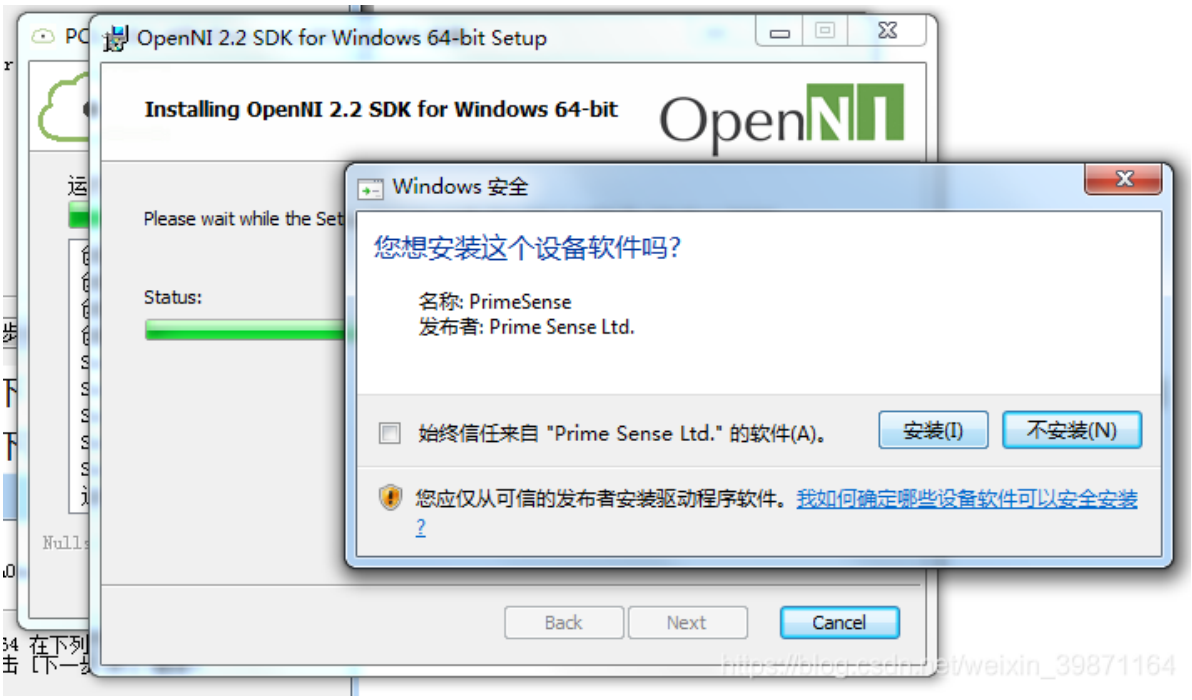
2. 安装配置，将下好的exe文件直接运行安装：
   选择一个目录安装，这里最好保证**路径没有中文和空格**



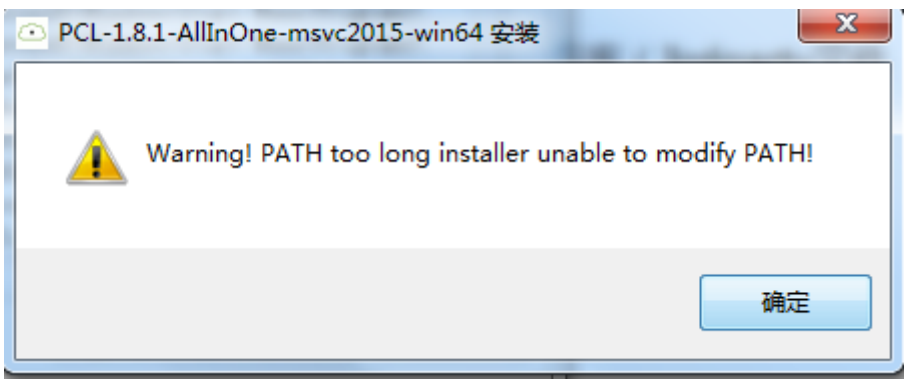
这里最好选择安装到系统所有用户，安装完后会自动添加pcl的环境变量



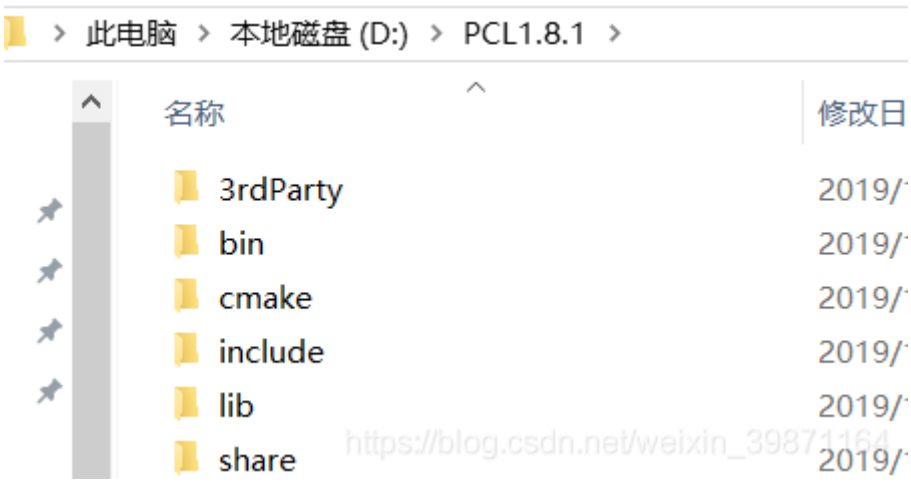
安装过程中会弹出窗口提示安装OpenNI，将OpenNI的安装路径更改为自己选定的安装PCL的第三方库（3rdparty文件夹）的路径下即：D:\PCL1.8.1\PCL1.8.1\3rdparty\OpenNI2，然后同意安装



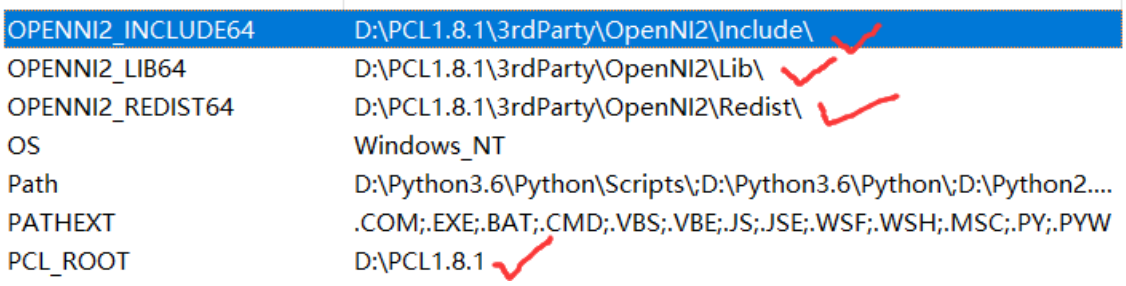
若出现这个警告，直接确认忽略



安装完后的目录样纸：



3. 把下载的PDB文件解压全部放在刚安装完PCL下的bin目录下面去
4. 环境变量的设置
    - 首先**计算机—>属性—>高级系统属性—>高级—>环境变量**
    - 在系统子菜单里面找到“Path”

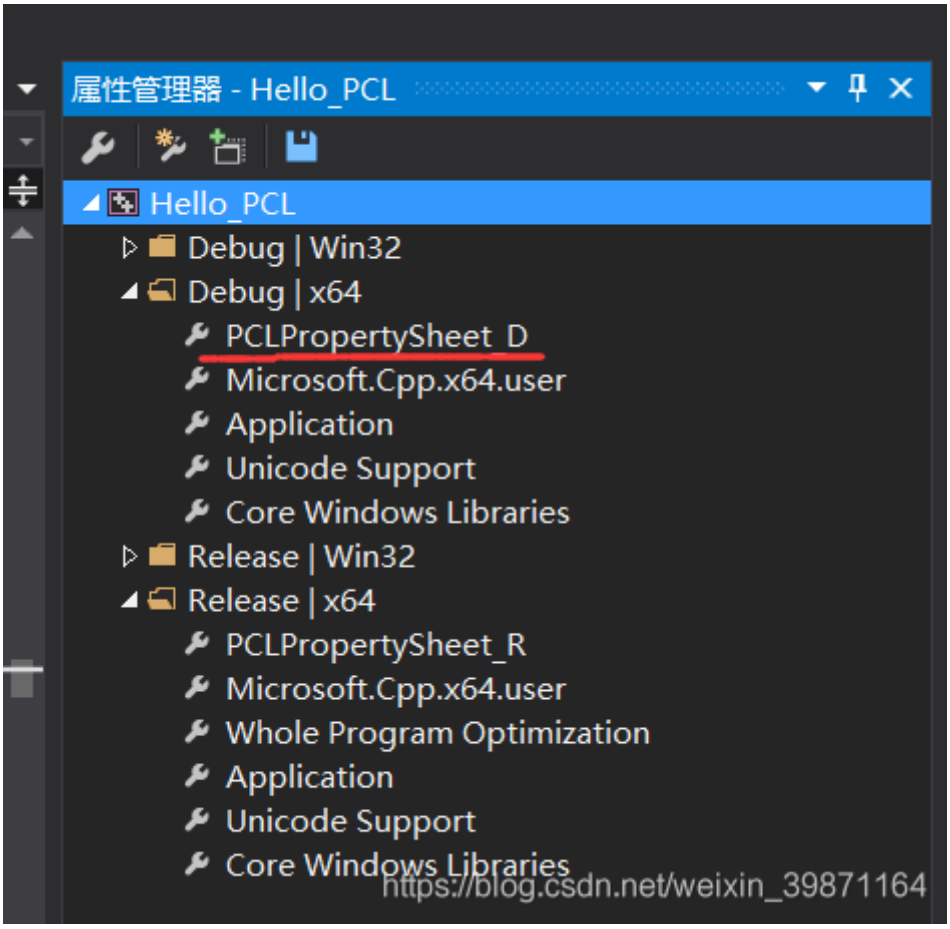


OPENNI2_INCLUDE64	D:\PCL1.8.1\3rdParty\OpenNI2\Include\
OPENNI2_LIB64	D:\PCL1.8.1\3rdParty\OpenNI2\Lib\
OPENNI2_REDIST64	D:\PCL1.8.1\3rdParty\OpenNI2\Redist\
OS	Windows_NT
Path	D:\Python3.6\Python\Scripts\;D:\Python3.6\Python\;D:\Python2....
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
PCL_ROOT	D:\PCL1.8.1

图示的系统环境变量是之前pcl安装时自动添加的，若没有的话可自行对照添加
现在还要做的就是，将第三方的bin文件添加到系统文件中去，如下
在系统的Path环境变量中手动添加：
%PCL_ROOT%\bin
%PCL_ROOT%\3rdParty\FLANN\bin
%PCL_ROOT%\3rdParty\Qhull\bin
%PCL_ROOT%\3rdParty\OpenNI2\Tools
%PCL_ROOT%\3rdParty\VTK\bin

2.1.2在vs2015中配置pcl库

1. 在VS中新建一个项目，并打开属性管理器，找到Debug|x64，新建属性页命名为PCLPropertySheet_D（命名可随意）



2. 双击打开属性页，进行配置（这里的路径根据自己的安装情况填写）：
   ①点击“VC++目录”->“包含目录”，进行编辑。
   添加下列路径：（按照自己的目录来设置）
   D:\PCL1.8.1\include\pcl-1.8.
   D:\PCL1.8.1\3rdParty\Boost\include\boost-1_64
   D:\PCL1.8.1\3rdParty\Eigen\eigen3
   D:\PCL1.8.1\3rdParty\FLANN\include
   D:\PCL1.8.1\3rdParty\FLANN\include\flann
   D:\PCL1.8.1\3rdParty\OpenNI2\Include
   D:\PCL1.8.1\3rdParty\Qhull\include
   D:\PCL1.8.1\3rdParty\VTK\include\vtk-8.0
   ②点击“VC++目录”->“库目录”，进行编辑。
   添加下列路径：（按照自己的目录来设置）
   D:\PCL1.8.1\lib
   D:\PCL1.8.1\3rdParty\Boost\lib
   D:\PCL1.8.1\3rdParty\FLANN\lib
   D:\PCL1.8.1\3rdParty\OpenNI2\Lib
   D:\PCL1.8.1\3rdParty\Qhull\lib
   D:\PCL1.8.1\3rdParty\VTK\lib
   ③点击“链接器”->“输入”->“附加依赖项”，进行编辑，添加lib文件
   这里的lib文件太多，基本上是pcl安装目录下所有lib目录的lib文件：

   pcl_common_debug.lib
   pcl_features_debug.lib
   pcl_filters_debug.lib
   pcl_io_debug.lib
   pcl_io_ply_debug.lib
   pcl_kdtree_debug.lib
   pcl_keypoints_debug.lib

pcl_ml_debug.lib
pcl_octree_debug.lib
pcl_outofcore_debug.lib
pcl_people_debug.lib
pcl_recognition_debug.lib
pcl_registration_debug.lib
pcl_sample_consensus_debug.lib
pcl_search_debug.lib
pcl_segmentation_debug.lib
pcl_stereo_debug.lib
pcl_surface_debug.lib
pcl_tracking_debug.lib
pcl_visualization_debug.lib
flann_cpp_s-gd.lib
flann_s-gd.lib
flann-gd.lib
libboost_atomic-vc140-mt-gd-1_64.lib
libboost_chrono-vc140-mt-gd-1_64.lib
libboost_container-vc140-mt-gd-1_64.lib
libboost_context-vc140-mt-gd-1_64.lib
libboost_coroutine-vc140-mt-gd-1_64.lib
libboost_date_time-vc140-mt-gd-1_64.lib
libboost_exception-vc140-mt-gd-1_64.lib
libboost_filesystem-vc140-mt-gd-1_64.lib
libboost_graph-vc140-mt-gd-1_64.lib
libboost_iostreams-vc140-mt-gd-1_64.lib
libboost_locale-vc140-mt-gd-1_64.lib
libboost_log-vc140-mt-gd-1_64.lib
libboost_log_setup-vc140-mt-gd-1_64.lib
libboost_math_c99-vc140-mt-gd-1_64.lib
libboost_math_c99f-vc140-mt-gd-1_64.lib
libboost_math_c99l-vc140-mt-gd-1_64.lib
libboost_math_tr1-vc140-mt-gd-1_64.lib
libboost_math_tr1f-vc140-mt-gd-1_64.lib
libboost_math_tr1l-vc140-mt-gd-1_64.lib
libboost_mpi-vc140-mt-gd-1_64.lib
libboost_prg_exec_monitor-vc140-mt-gd-1_64.lib
libboost_program_options-vc140-mt-gd-1_64.lib
libboost_random-vc140-mt-gd-1_64.lib
libboost_regex-vc140-mt-gd-1_64.lib
libboost_serialization-vc140-mt-gd-1_64.lib
libboost_signals-vc140-mt-gd-1_64.lib
libboost_system-vc140-mt-gd-1_64.lib
libboost_test_exec_monitor-vc140-mt-gd-1_64.lib
libboost_thread-vc140-mt-gd-1_64.lib
libboost_timer-vc140-mt-gd-1_64.lib
libboost_unit_test_framework-vc140-mt-gd-1_64.lib
libboost_wave-vc140-mt-gd-1_64.lib
libboost_wserialization-vc140-mt-gd-1_64.lib
qhull_d.lib
qhull_p_d.lib
qhull_r_d.lib

qhullcpp_d.lib
qhullstatic_d.lib
qhullstatic_r_d.lib
vtkalglib-8.0-gd.lib
vtkChartsCore-8.0-gd.lib
vtkCommonColor-8.0-gd.lib
vtkCommonComputationalGeometry-8.0-gd.lib
vtkCommonCore-8.0-gd.lib
vtkCommonDataModel-8.0-gd.lib
vtkCommonExecutionModel-8.0-gd.lib
vtkCommonMath-8.0-gd.lib
vtkCommonMisc-8.0-gd.lib
vtkCommonSystem-8.0-gd.lib
vtkCommonTransforms-8.0-gd.lib
vtkDICOMParser-8.0-gd.lib
vtkDomainsChemistry-8.0-gd.lib
vtkexoIIc-8.0-gd.lib
vtkexpat-8.0-gd.lib
vtkFiltersAMR-8.0-gd.lib
vtkFiltersCore-8.0-gd.lib
vtkFiltersExtraction-8.0-gd.lib
vtkFiltersFlowPaths-8.0-gd.lib
vtkFiltersGeneral-8.0-gd.lib
vtkFiltersGeneric-8.0-gd.lib
vtkFiltersGeometry-8.0-gd.lib
vtkFiltersHybrid-8.0-gd.lib
vtkFiltersHyperTree-8.0-gd.lib
vtkFiltersImaging-8.0-gd.lib
vtkFiltersModeling-8.0-gd.lib
vtkFiltersParallel-8.0-gd.lib
vtkFiltersParallelImaging-8.0-gd.lib
vtkFiltersProgrammable-8.0-gd.lib
vtkFiltersSelection-8.0-gd.lib
vtkFiltersSMP-8.0-gd.lib
vtkFiltersSources-8.0-gd.lib
vtkFiltersStatistics-8.0-gd.lib
vtkFiltersTexture-8.0-gd.lib
vtkFiltersVerdict-8.0-gd.lib
vtkfreetype-8.0-gd.lib
vtkGeovisCore-8.0-gd.lib
vtkhdf5-8.0-gd.lib
vtkhdf5_hl-8.0-gd.lib
vtkImagingColor-8.0-gd.lib
vtkImagingCore-8.0-gd.lib
vtkImagingFourier-8.0-gd.lib
vtkImagingGeneral-8.0-gd.lib
vtkImagingHybrid-8.0-gd.lib
vtkImagingMath-8.0-gd.lib
vtkImagingMorphological-8.0-gd.lib
vtkImagingSources-8.0-gd.lib
vtkImagingStatistics-8.0-gd.lib
vtkImagingStencil-8.0-gd.lib

vtkInfovisCore-8.0-gd.lib
vtkInfovisLayout-8.0-gd.lib
vtkInteractionImage-8.0-gd.lib
vtkInteractionStyle-8.0-gd.lib
vtkInteractionWidgets-8.0-gd.lib
vtkIOAMR-8.0-gd.lib
vtkIOCore-8.0-gd.lib
vtkIOEnSight-8.0-gd.lib
vtkIOExodus-8.0-gd.lib
vtkIOExport-8.0-gd.lib
vtkIOGeometry-8.0-gd.lib
vtkIOImage-8.0-gd.lib
vtkIOImport-8.0-gd.lib
vtkIOInfovis-8.0-gd.lib
vtkIOLegacy-8.0-gd.lib
vtkIOLSDyna-8.0-gd.lib
vtkIOMINC-8.0-gd.lib
vtkIOMovie-8.0-gd.lib
vtkIONetCDF-8.0-gd.lib
vtkIOParallel-8.0-gd.lib
vtkIOPLY-8.0-gd.lib
vtkIOSQL-8.0-gd.lib
vtkIOVideo-8.0-gd.lib
vtkIOXML-8.0-gd.lib
vtkIOXMLParser-8.0-gd.lib
vtkjpeg-8.0-gd.lib
vtkjsoncpp-8.0-gd.lib
vtklibxml2-8.0-gd.lib
vtkmetaio-8.0-gd.lib
vtkNetCDF-8.0-gd.lib
vtknetcdf_c++-gd.lib
vtkoggtheora-8.0-gd.lib
vtkParallelCore-8.0-gd.lib
vtkpng-8.0-gd.lib
vtkproj4-8.0-gd.lib
vtkRenderingAnnotation-8.0-gd.lib
vtkRenderingContext2D-8.0-gd.lib
vtkRenderingCore-8.0-gd.lib
vtkRenderingFreeType-8.0-gd.lib
vtkRenderingImage-8.0-gd.lib
vtkRenderingLabel-8.0-gd.lib
vtkRenderingLOD-8.0-gd.lib
vtkRenderingVolume-8.0-gd.lib
vtksqlite-8.0-gd.lib
vtksys-8.0-gd.lib
vtktiff-8.0-gd.lib
vtkverdict-8.0-gd.lib
vtkViewsContext2D-8.0-gd.lib
vtkViewsCore-8.0-gd.lib
vtkViewsInfovis-8.0-gd.lib
vtkzlib-8.0-gd.lib
OpenNI2.lib

3. 同理在release下重复以上三个步骤，需要注意的是添加的lib是release下的lib

   如下：

   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_s.lib
   flann.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_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_mpi-vc140-mt-1_64.lib
   libboost_prg_exec_monitor-vc140-mt-1_64.lib
   libboost_program_options-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_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
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

配置完成，测试配置是否正确
代码如下：

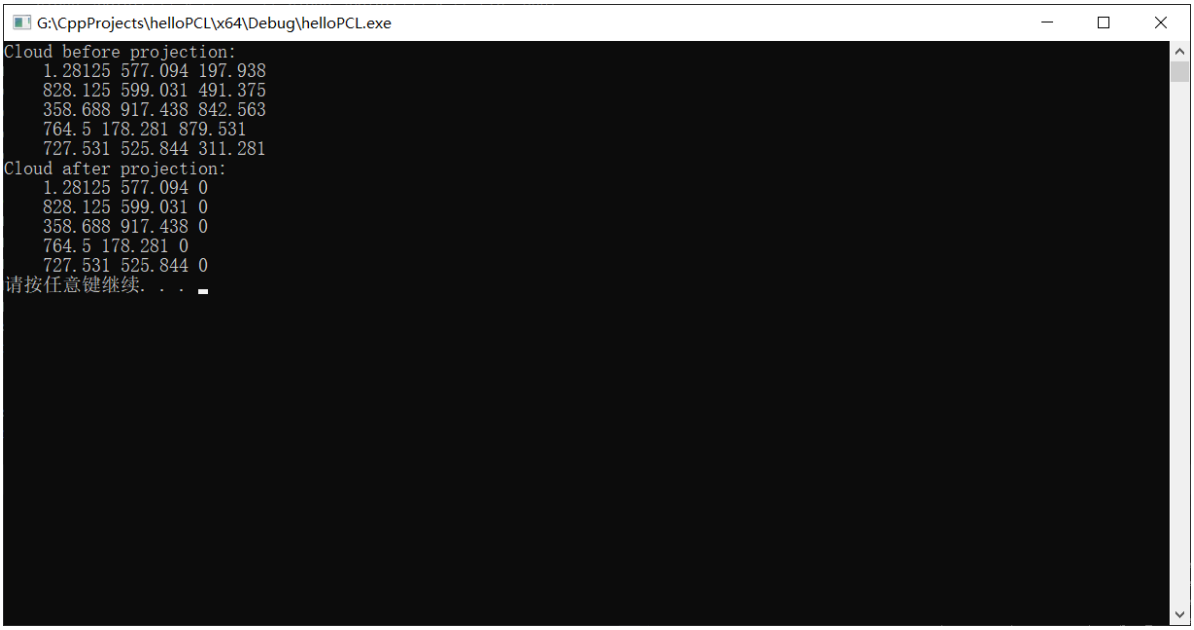
```
#include <iostream>
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/ModelCoefficients.h>
#include <pcl/filters/project_inliers.h>
int main(int argc, char** argv)
{
    pcl::PointCloud<pcl::PointXYZ>::Ptr cloud(new
pcl::PointCloud<pcl::PointXYZ>);
    pcl::PointCloud<pcl::PointXYZ>::Ptr cloud_projected(new
pcl::PointCloud<pcl::PointXYZ>);
    cloud->width = 5;
    cloud->height = 1;
    cloud->points.resize(cloud->width * cloud->height);
    for (size_t i = 0; i < cloud->points.size(); ++i)
    {
        cloud->points[i].x = 1024 * rand() / (RAND_MAX + 1.0f);
        cloud->points[i].y = 1024 * rand() / (RAND_MAX + 1.0f);
        cloud->points[i].z = 1024 * rand() / (RAND_MAX + 1.0f);
    }
    std::cerr << "Cloud before projection: " << std::endl;
    for (size_t i = 0; i < cloud->points.size(); ++i)
        std::cerr << "    " << cloud->points[i].x << " " <<
        cloud->points[i].y << " " << cloud->points[i].z << std::endl;

    // Create a set of planar coefficients with X=Y=0,Z=1
    pcl::ModelCoefficients::Ptr coefficients(new pcl::ModelCoefficients());
    coefficients->values.resize(4);
    coefficients->values[0] = coefficients->values[1] = 0;
    coefficients->values[2] = 1.0;
    coefficients->values[3] = 0;

    // Create the filtering object
    pcl::ProjectInliers<pcl::PointXYZ> proj;
    proj.setModelType(pcl::SACMODEL_PLANE);
    proj.setInputCloud(cloud);
    proj.setModelCoefficients(coefficients);
    proj.filter(*cloud_projected);
    std::cerr << "Cloud after projection: " << std::endl;
    for (size_t i = 0; i < cloud_projected->points.size(); ++i)
        std::cerr << "    " << cloud_projected->points[i].x << " " <<
        cloud_projected->points[i].y << " " << cloud_projected->points[i].z <<
std::endl;
```

```
    system("pause");
    return (0);
}
```

运行结果：



编译过程报错情况及解决方法：

a)编译的时候记得将编译器变为x64的环境下运行，然后估计会报这样的错误：

error C4996: 'pcl::SAC_SAMPLE_SIZE': This map is deprecated and is kept only to prevent breaking existing user...

解决方式：属性"->"C/C++"->"SDL检查"：改为否(/sdl)

b)为了防止弹出很多的warning，加入预处理器

在属性目录"C/C++ "– >"预处理器"–>"预处理器定义" 加入：

_SCL_SECURE_NO_WARNINGS

_CRT_SECURE_NO_WARNINGS

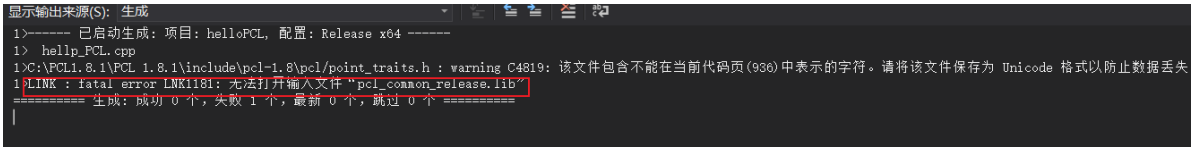
PCL_NO_PRECOMPILE

c)在Debug下编译会出现大量的如下warning（Release下没有）：

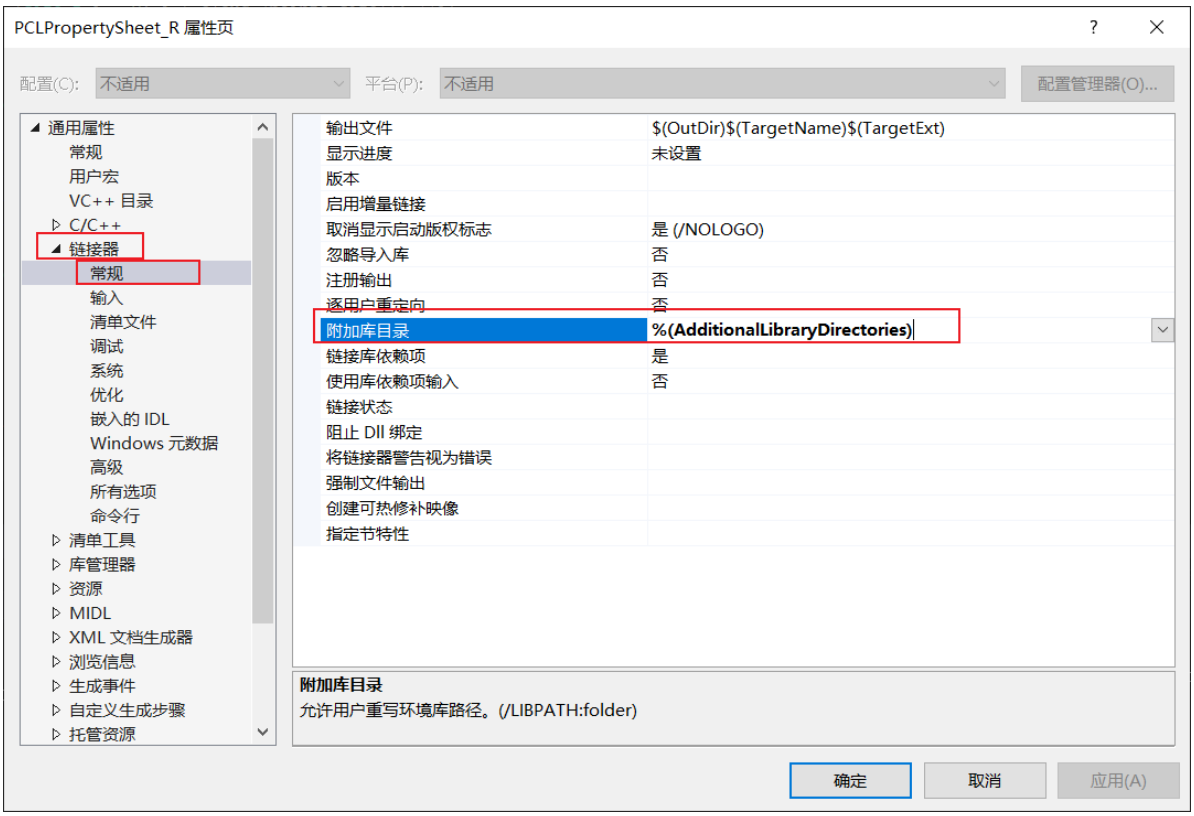
warning LNK4099: 未找到 PDB".pdb"正在链接对象,如同没有调试信息一样

解决方法："视图"->"解决方案资源管理器"->"属性"->"链接器"->"调试"，在右侧的"生成调试信息"改为"否"，确定即可。

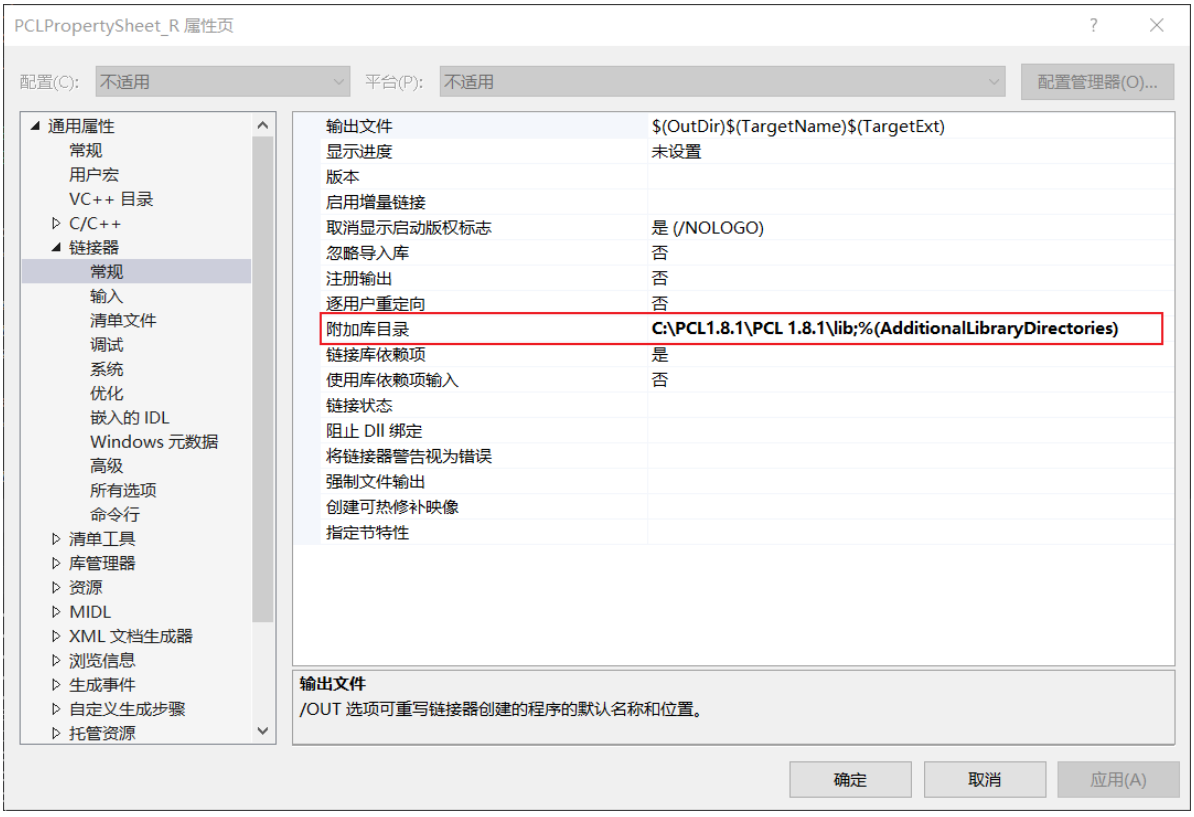
d)如在release下出现无法打开输入文件"pcl_common_release.lib"错误



解决方法：打开release属性页，找到链接器-> 常规-> 添加库目录->编辑->添加pcl1.8.1\pcl1.8.1\lib



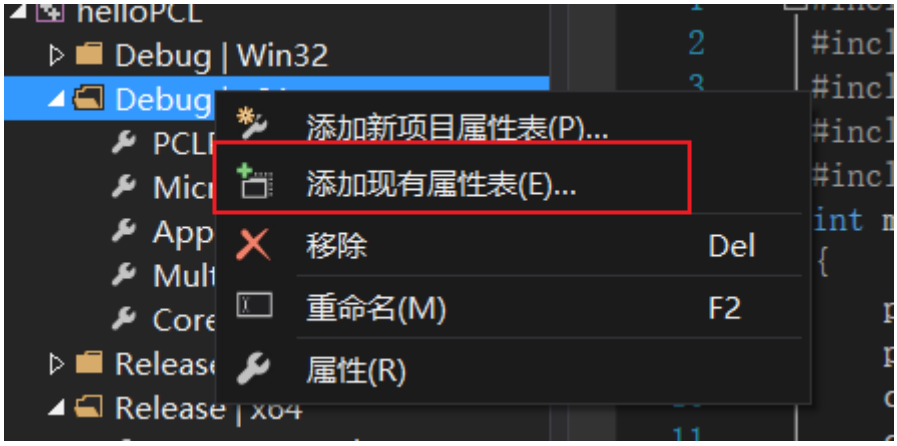
添加完成如下：



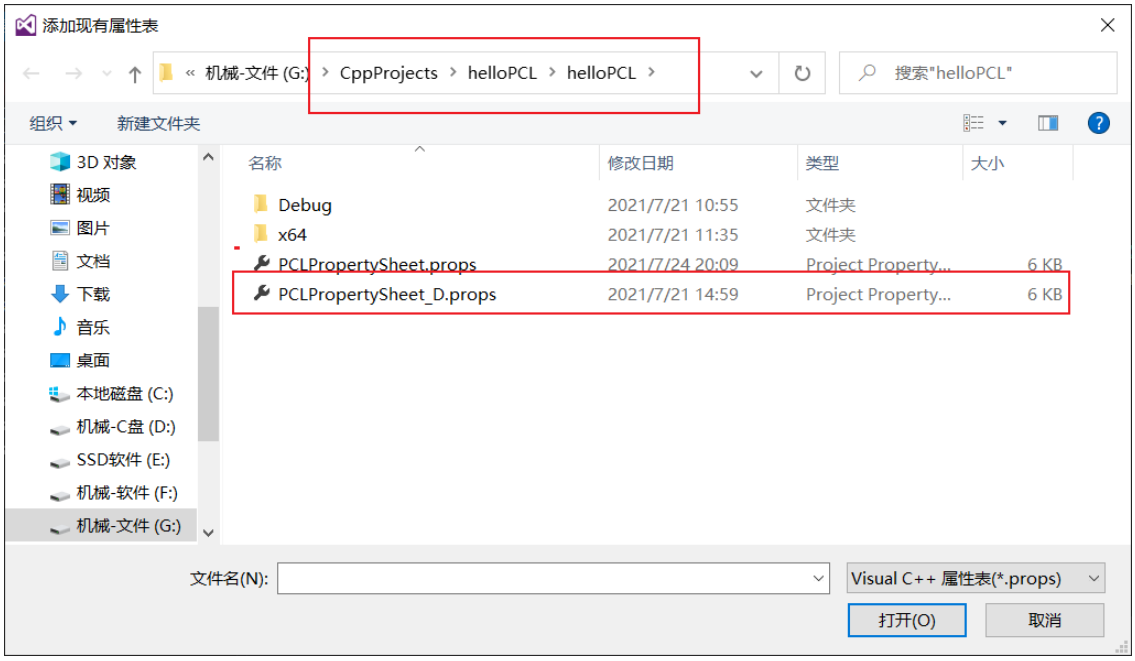
在进行编译即可。

4. 使用其他工程时如何配置

   在属性管理器：Debug|x64 中右键添加现有属性表



找到上一个测试工程中配置好的PCLPropertySheet_D.props，添加即可，Release|x64也是同样操作。添加完成即可使用了。



参考博客链接：https://blog.csdn.net/weixin_39871164/article/details/102879962

2.2 PCL在ubuntu系统下的安装编译

2.2.1 ROS中自带的PCL库（1.7）

ROS_PCL库的简单使用(第一个PCL程序)

1）创建ROS程序包

程序包是基于pcl_conversions pcl_ros pcl_msgs sensor_msgs

```
cd catkin_ws/src
```

```
catkin_create_pkg pcl_tutorials pcl_conversions pcl_ros pcl_msgs sensor_msgs
```

```
rospack profile
```

```
roscd pcl_tutorials
```

```
mkdir src
```

2）创建pcl_sample.cpp

其内容如下：

```
#include <ros/ros.h>
#include <pcl/point_cloud.h>
#include <pcl_ros/point_cloud.h>
#include <pcl_conversions/pcl_conversions.h>
#include <sensor_msgs/PointCloud2.h>

main (int argc, char** argv)
{
    ros::init (argc, argv, "pcl_sample");
    ros::NodeHandle nh;
    ros::Publisher pcl_pub = nh.advertise<sensor_msgs::PointCloud2>
("pcl_output", 1);
    sensor_msgs::PointCloud2 output;
    pcl::PointCloud<pcl::PointXYZ>::Ptr cloud (new
pcl::PointCloud<pcl::PointXYZ>);
    // Fill in the cloud data
    cloud->width = 100;
    cloud->height = 1;
    cloud->points.resize (cloud->width * cloud->height);
    //Convert the cloud to ROS message
    pcl::toROSMsg (*cloud, output);
    pcl_pub.publish(output);
    ros::spinOnce();
    return 0;
}
```

3) 添加PCL libraries 到 CMakeLists.txt

```
find_package(PCL REQUIRED)
include_directories(include ${PCL_INCLUDE_DIRS})
link_directories(${PCL_LIBRARY_DIRS})
```

4) 根据适当的库生成可执行文件和链接

```
add_executable(pcl_sample src/pcl_sample.cpp)
target_link_libraries(pcl_sample ${catkin_LIBRARIES} ${PCL_LIBRARIES})
```

5) 编译

```
cd catkin_ws
```

```
source devel/setup.bash
```

```
catkin_make
```

6) 编译没有啥问题，但也没啥显示的

上述教程原文链接：https://blog.csdn.net/hanshuning/article/details/52314108

更详细一点的官方教程：https://blog.csdn.net/muyiyushan/article/details/114592222

2.2.2 源码安装PCL库

（本人ubuntu16.04在有ROS的情况下未安装成功，可能是依赖版本冲突啥的不知道什么原因，所以这部分只提供一下资料参考）

（在0 资料汇总 第二个语雀的学习笔记中有ubuntu系统安装1.9版本的教程以及vscode配置pcl开发环境教程）