# 点云PCL估计一个点云的表面法线

表面法线是几何体表面的重要属性，在很多领域都有大量应用，例如：在进行光照渲染时产生符合可视习惯的效果时需要表面法线信息才能正常进行，对于一个已知的几何体表面，根据垂直于点表面的矢量，因此推断表面某一点的法线方向通常比较简单。然而，由于我们获取的点云数据集在真实物体的表面表现为一组定点样本，这样就会有两种解决方法：

使用曲面重建技术，从获取的点云数据集中得到采样点对应的曲面，然后从曲面模型中计算表面法线；

直接从点云数据集中近似推断表面法线。

本小节将针对后一种情况进行讲解，已知一个点云数据集，在其中的每个点处直接近似计算表面法线。

**理论基础**

尽管有许多不同的法线估计方法，本教程中着重讲解的是其中最简单的一个，表述如下，确定表面一点法线的问题近似于估计表面的一个相切面法线的问题，因此转换过来以后就变成一个最小二乘法平面拟合估计问题。

注意：更多信息，包含最小二乘法问题的数学方程式，请见[RusuDissertation](http://pointclouds.org/documentation/tutorials/how_features_work.php#rusudissertation)。

因此估计表面法线的解决方案就变成了分析一个协方差矩阵的特征矢量和特征值（或者PCA—主成分分析），这个协方差矩阵从查询点的近邻元素中创建。更具体地说，对于每一个点Pi,对应的协方差矩阵C，如下：

$$C=\frac{1}{k}\sum_{i=1}^{k}\cdot(\mathcal{P}_i-\bar{\mathcal{P}})\cdot(\mathcal{P}_i-\bar{\mathcal{P}})^{T},\ C\cdot\vec{v}_j=\lambda_j\cdot\vec{v}_j,\ j\in\{0,1,2\}\quad(1)$$

此处，k是点Pi邻近点的数目，$\bar{\mathcal{P}}$表示最近邻元素的三维质心。这里应该是三维的，即包括x，y，z坐标，建立坐标系，假设第i个点的坐标是ri，质心的坐标就是Σmi*ri/(Σmi)[1]，姑且认为他是正确的，$\lambda_j$是协方差矩阵的第j 个特征值，$\vec{v}_j$是第j个特征向量。这里，特征向量的意义是经过过这种特定的变换后保持方向不变。只是进行长度上的伸缩而已。而特征值的意义是一个变换（矩阵）可由它的所有特征向量完全表示，而每一个向量所对应的特征值，就代表了矩阵在这一向量上的贡献率——说的通俗一点就是能量（power）[2]。

在PCL内估计一点集对应的协方差矩阵，可以使用以下函数调用实现：

```
//定义每个表面小块的3x3协方差矩阵的存储对象
```

```
Eigen::Matrix3fcovariance_matrix;
```

```
//定义一个表面小块的质心坐标16-字节对齐存储对象
```

```
Eigen::Vector4fxyz_centroid;
```

```
//估计质心坐标
```

```
compute3DCentroid(cloud,xyz_centroid);
```

```
//计算3x3协方差矩阵
```

```
computeCovarianceMatrix(cloud,xyz_centroid,covariance_matrix);
```

matlab中求协方差矩阵

**cov(X)** 求矩阵X的协方差矩阵。diag(cov(X))得到每一个列向量的方差。sqrt(diag(cov(X)))得到每一个列的标准差。这里与（1）还有些不同，（1）式是由给定一点，协方差矩阵是由其周围的k邻近个点决定。

通常，没有数学方法能解决法线的正负向问题，如上所示，通过主成分分析法（PCA）来计算它的方向也具有二义性，无法对整个点云数据集的法线方向进行一致性定向。图1显示出对一个更大数据集的两部分产生的影响，此数据集来自于厨房环境的一部分，很明显估计的法线方向并非完全一致，图2展现了其对应扩展的高斯图像（EGI），也称为法线球体（normal sphere），（我的理解是所有点云的方向在一个球体中的表示。）它描述了点云中所有法线的方向。由于数据集是2.5维，其只从一个单一的视角获得，因此法线应该仅呈现出一半球体的扩展高斯图像（EGI）。然而，由于定向的不一致性，它们遍布整个球体，如图2所示。

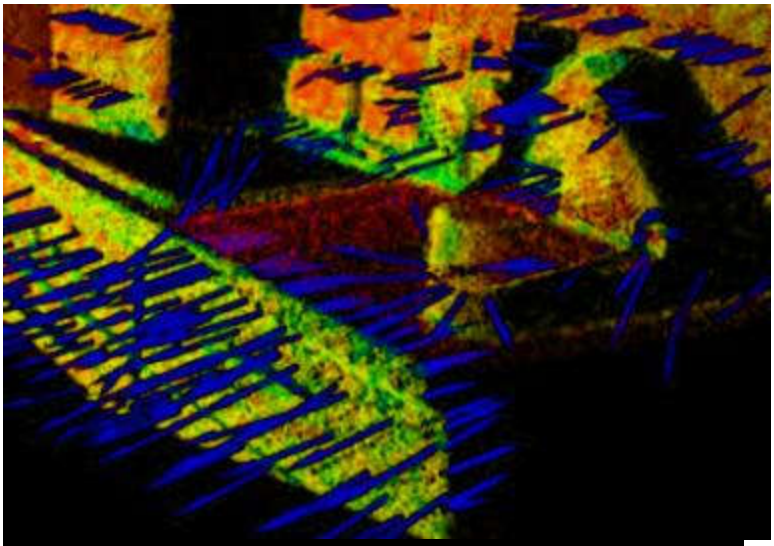


图1 估计厨房环境的表面法线

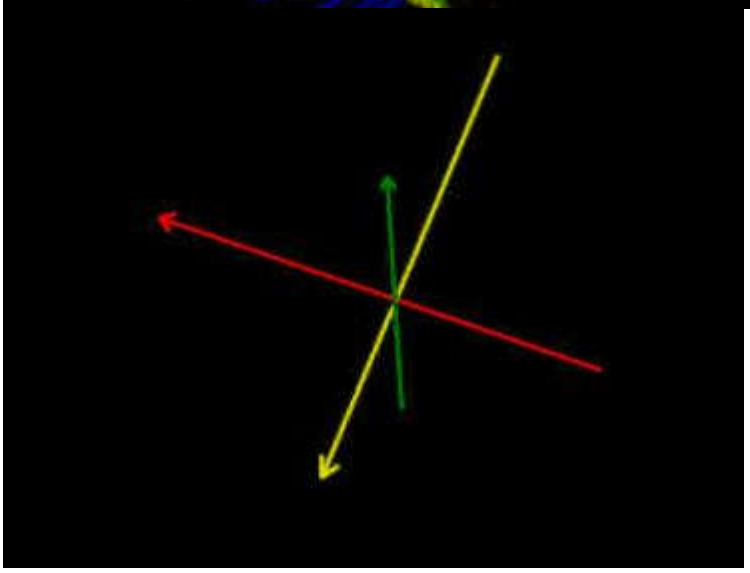


图2 法线球体

如果实际知道视点Vp，那么这个问题的解决是非常简单的。对所有$\overrightarrow{n_i}$法线定向只需要使它们一致朝向视点方向，满足下面的方程式：

$$\overrightarrow{n_i} \cdot (\mathcal{V}_{\mathrm{P}} - \mathcal{P}_i) > 0$$ （两个向量点积大于0）

下面的图3展现了上面图1中的数据集的所有法线被一致定向到视点后的结果演示。

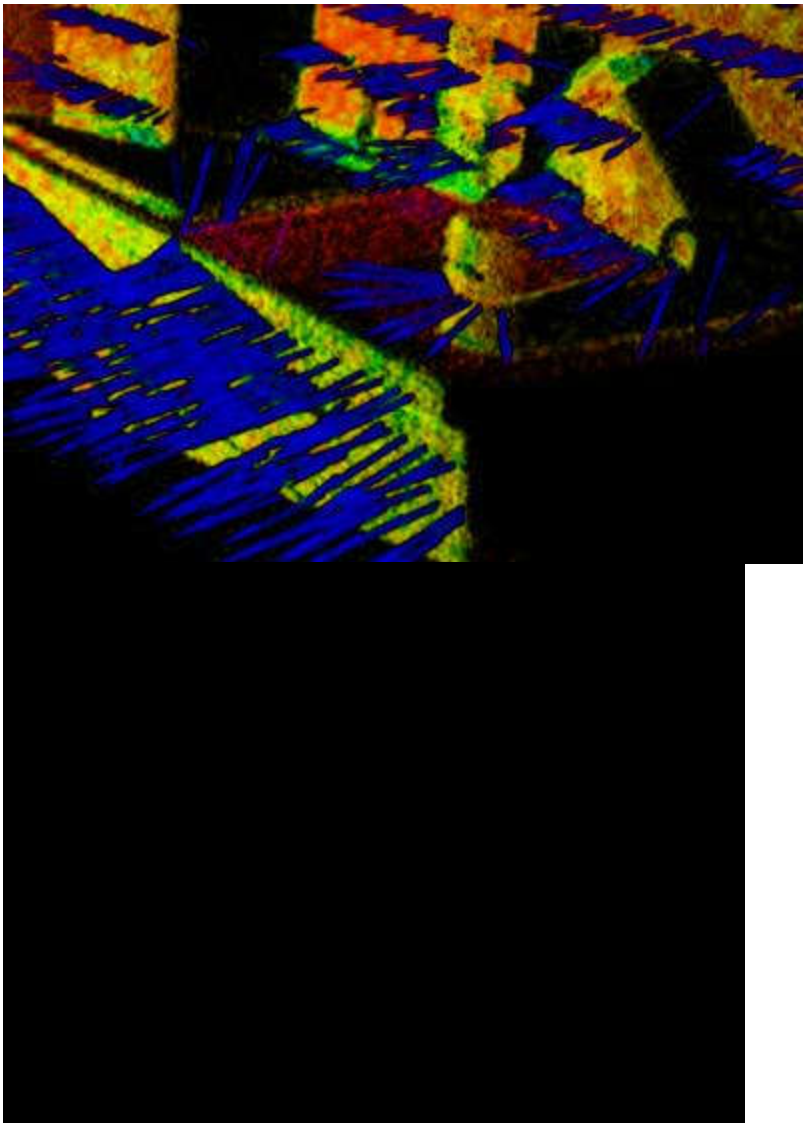


图3 将所有法线一致定向到后视点的结果

在PCL中对一个已知点的法线进行手动重新定向，你可以使用：

**flipNormalTowardsViewpoint (const PointT &point, float vp_x, float vp_y, float vp_z, Eigen::Vector4f &normal);**

注意：如果数据集是从多个捕获视点中配准后集成的，那么上述法线的一致性定向方法就不适用了。需要使用更复杂的算法 (http://www.xuebuyuan.com/category/%E7%AE%97%E6%B3%95)，更多信息，请见[RusuDissertation] (http://pointclouds.org/documentation/tutorials/how_features_work.php#rusudissertation)。

**选择合适的尺度**

如之前介绍的，在估计一个点的表面法线时，我们需要从周围支持这个点的邻近点着手（也称作k邻域）。最近邻估计问题的具体内容又提出了另一个问题“合适的尺度”：已知一个取样点云数据集，k的正确取值是多少（k通过pcl::Feature::setKSearch给出）或者确定一个点r为半径的圆内的最近邻元素集时使用的半径r应该取什么值（r通过pcl::Feature::setRadiusSearch给出）。这个问题非常重要，并且在一个点特征算子的自动估计时（例如：用户没有给定阈值）是一个限制因素。为了更好地说明这个问题，以下图示表现了选择更小尺度（如：r值或k取相对

小）与选择更大尺度（如：r值或k值比较大）时的两种不同效果。图4和图5分别为近视图和远视图，两图中左边部分展示选择了一个合理的比例因子，估计的表面法线近似垂直于两个平面，即使在互相垂直的边沿部分，可明显看到边沿。如果这个尺度取的太大（右边部分，k选择的过大或者r选择的过大），这样邻近点集将更大范围地覆盖邻近表面的点，估计的点特征表现就会扭曲失真，在两个平面边缘处出现旋转表面法线，以及模糊不清的边界，这样就隐藏了一些细节信息。

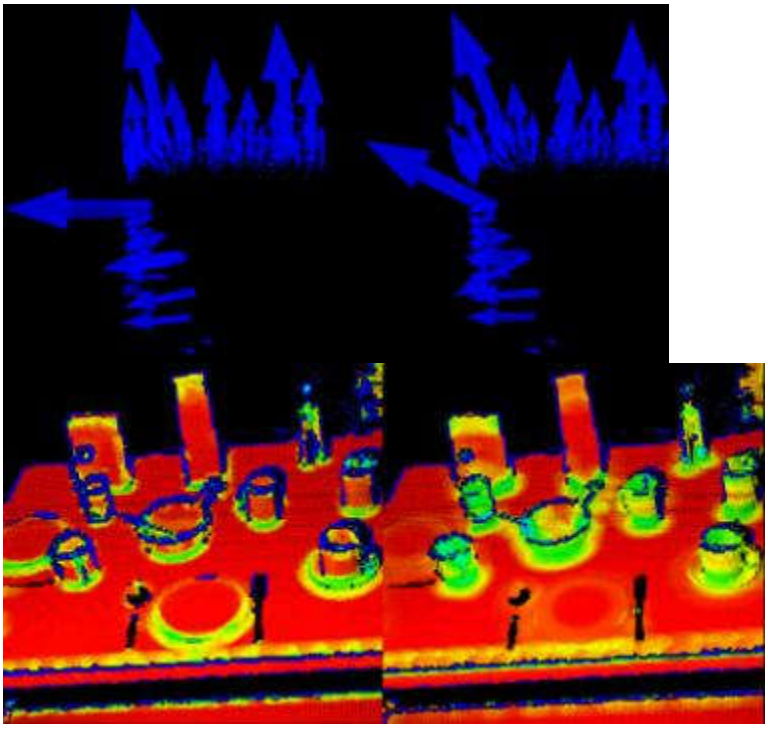


图4选择合理的比例因子　　图5选择较大的比例因子

无法深入探究更多讨论，现在可粗略假设，以应用程序所需的细节需求为参考，选择确定点的邻域所用的尺度。简言之，如果杯子手柄和圆柱体部分之间边缘的曲率是重要的，那么需要足够小的尺度来捕获这些细节信息，而在其他不需要细节信息的应用中可选择
`的尺度。

**估计法线实例详解**

虽然法线估计的例子已经在上小节（http://pointclouds.org/documentation/tutorials/index.php#features-tutorial)中给过了，这里我们还是复习一下其中的一部分，以便更好地解释说明本节的后续部分，在PCL（Point Cloud Learning）中国协助发行的书[1]本书提供光盘的第12章例1文件夹中，打开名为normal_estimation.cpp的代码文件，同文件夹下可以找到相关的测试点云文件table_scene_lms400.pcd。下面详细介绍一些源代码中相关的函数和类。

法线估计类NormalEstimation的实际计算调用程序内部执行以下操作：

**对点云P中的每个点p**

**1.得到p点的最近邻元素**

**2.计算p点的表面法线n**

### 3.检查n的方向是否一致指向视点，如果不是则翻转

视点坐标默认为（0,0,0），可以使用以下代码进行更换：

**setViewPoint (float vpx, float vpy, float vpz);**

计算单个点的法线，使用：

**computePointNormal**
**(const pcl::PointCloud<PointInT>&cloud, const std::vector<int>&indices,**
**Eigen::Vector4f &plane_parameters, float&curvature);**

此处，cloud是包含点的输入点云，indices是点的k-最近邻元素集索引，plane_parameters和curvature是法线估计的输出，plane_parameters前三个坐标中，以(nx, ny, nz)来表示法线，第四个坐标D = nc . p_plane (centroid here) + p(这个坐标值D不是很理解，可以参考开源代码的说明)。输出表面曲率curvature通过协方差矩阵的特征值之间的运算估计得到，如：

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}$$

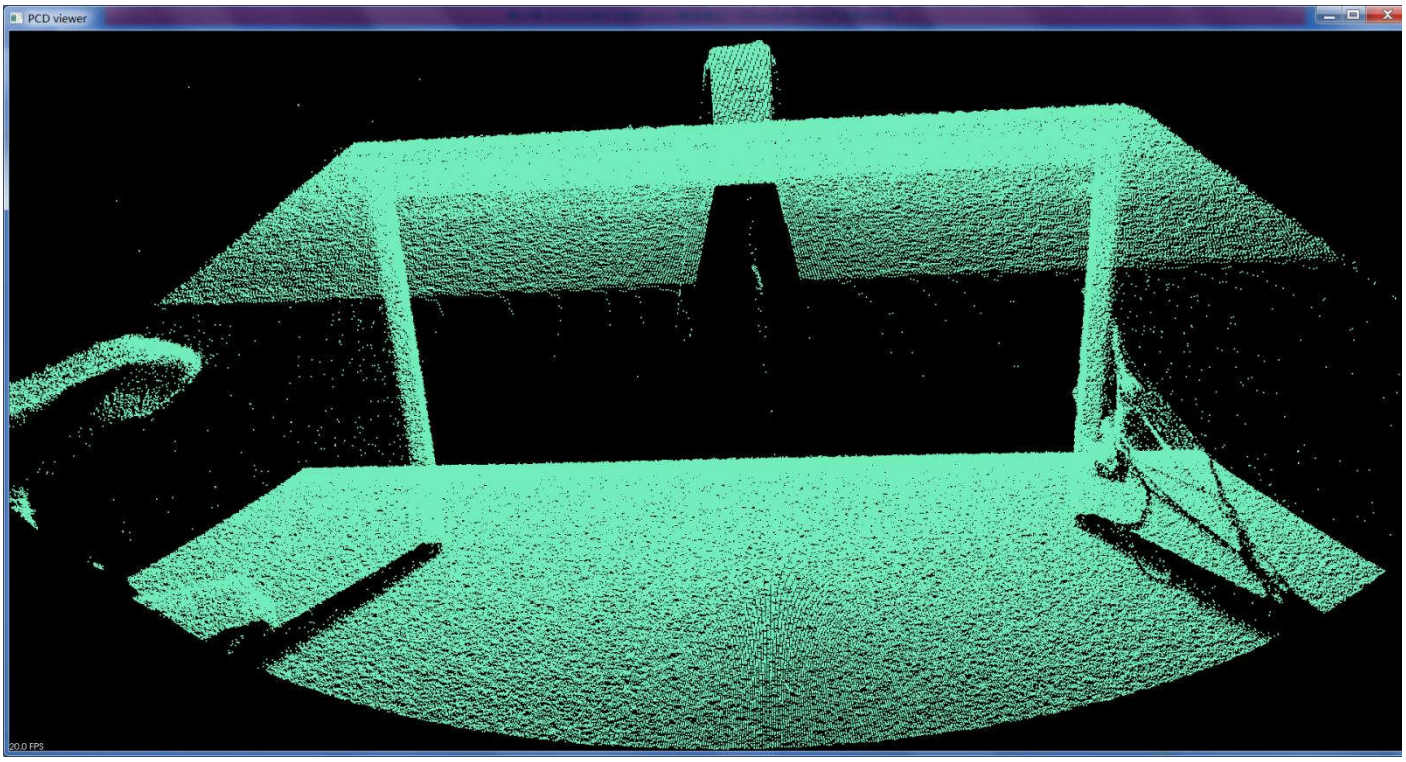



图6法线估计前原始点云

最后利用光盘提供的CMakeLists.txt文件，在cmake中建立工程文件，并生成相应的可执行文件,生成执行文件后，将测试点云文件拷贝到与可执行文件相同的路径下或者改变源代码中点云的路径，就可以运行了。如图6所示，为估计所用的点云可视化效果，法线估计后运行结果如图7所示，图中的小线段表示法线。

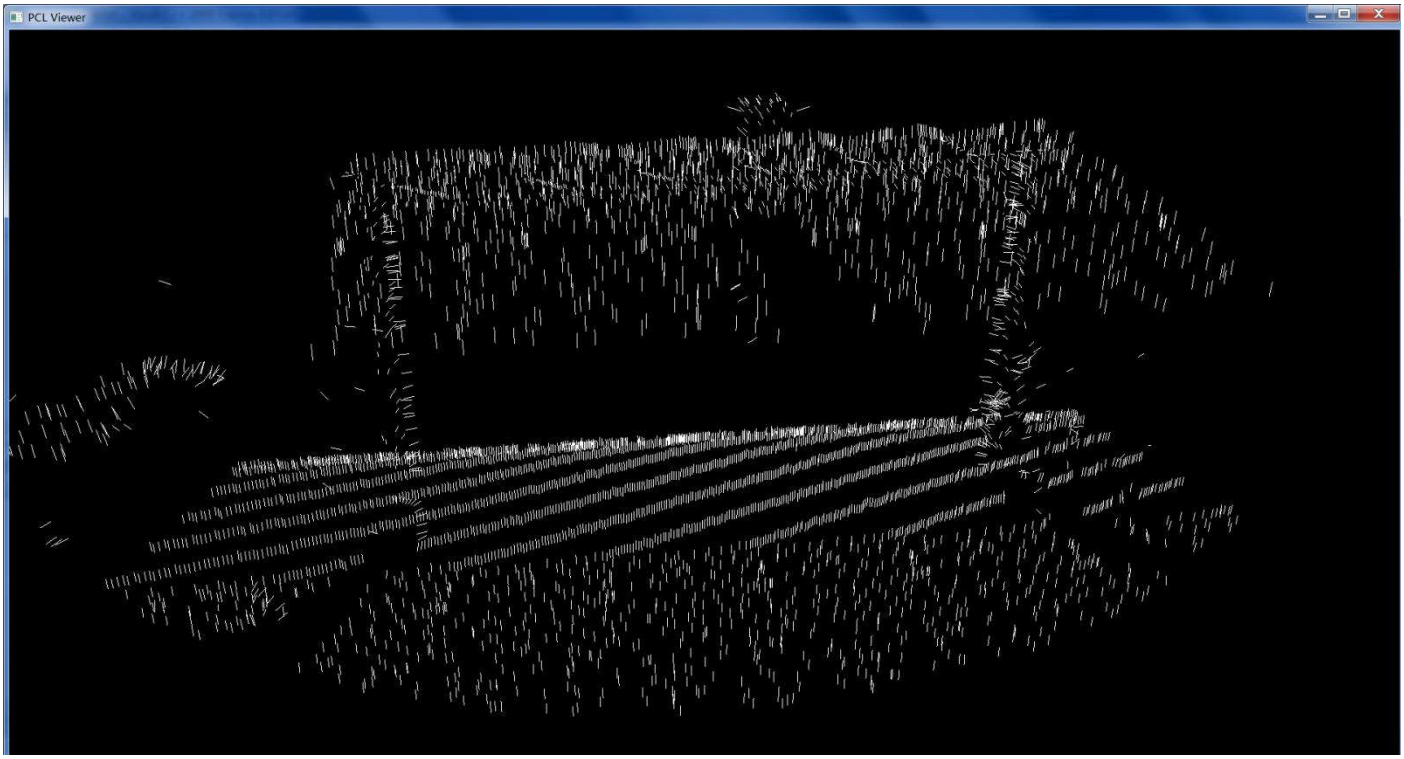



图7 法线估计结果可视化

**使用OpenMP加速法线估计**

对于对运算速度有要求的用户，PCL点云库提供了一个表面法线的附加实现程序，它使用多核/多线程开发规范，利用OpenMP来提高计算速度。它的类命名为pcl::NormaleEstimationOMP，并且它的应用程序接口（API）100%兼容单线程pcl::NormalEstimation，这使它适合作为一个可选提速方法。在8核系统中，可以轻松提速6-8倍。

参考资料:

[1] http://zhidao.baidu.com/link?url=e2XzUad5EULxhNCW7_7zHh9UKmQobW809H8VXixboioojGDma-T0MYWBG9YTo4eoUGj6KHePCGkpIXJ3sSIDUP5okdzKY3nMOrlRDp0X9Cy

[2] http://wenku.baidu.com/view/5587010a581b6bd97f19ea74.html

[3] 朱德海、郭浩、苏伟.点云库*PCL*学习教程（*ISBN 978-7-5124-0954-5*）北京航空航天出版社*2012-10*

转自：*http://www.xuebuyuan.com/2228903.html*



## 智能推荐

(/article/38431324244/)

## 点云库PCL模块简介（一） (/article/38431324244/)

1、I/O模块 PCL 中I/O库提供了点云文件输入输出相关的操作类，并封装了 OpenNI 兼容的设备源数据获取接口，可直接从众多感知设备获取点云图像等数据。I/O模块利用21个类与28个函数实现了对点云的获取、读人、存储等相关操作，其依赖于 pcl_common 和 pcl_octree 模块以及OpenNI 外部开发包。 &...

---

(/article/3736132098/)

## PCL（点云库）的源码安装 (/article/3736132098/)

pcl的官网 方法一( 通过PPA安装) （本人没成功） 方法二(源码安装) 安装依赖： 官网：boost、Eigen、FLANN、VTK是必须的 （注意依赖版本） 根据自己环境安装，我只安装了必须的依赖 下载： 进入目录、创建build文件夹 创建makefile 默认编译、安装（-j+线程数） 编译、安装的过程可能会出错，我遇到的问题： （1）No rule to make target '/...

---

(/article/8407753023/)

## PCL点云的保存和显示 (/article/8407753023/)

1.pcl点云的保存 1.首先将获得的点云x，y，z分别存放在三个vector中 2.pcl 是一个命名空间，跟std类似，PointCloud是类模板，<pcl::PointXYZ>是模板类实例化的类型 3cloud.height用来判断是否为有序点云，=1则是无序点云也可以使用如下函数代替：if (!cloud.isOrganized ()), 4.对于无序点云来说：(...

---



(/article/92991415361/)

## 一个可以快速复制opencv的lib库和PCL点云库lib库到附加依赖项的方法 (/article/929914153611/)

我们知道如果opencv不是world.lib的话，那么配置的话需要一个一个在附加依赖项输入lib，这对于opencv可能还好点，因为最多就20多个lib，但是对于PCL的话，基本就不要想一个一个复制到依赖库，因为有40-60库，怎么快速复制呢，网上搞了一堆脚本，麻烦不说而且操作不方便，今天我为大家一款软件神器只需要打开软件把lib往里面拖就可以了，也可以拖拽文件夹，可以说一键搞定。首先我们打开软...

---

## PCL：旋转、平移点云 (/article/7398346470/)

(/article/7398346470/)

cmake： result：...

## 猜你喜欢


(/article/9810517876/)

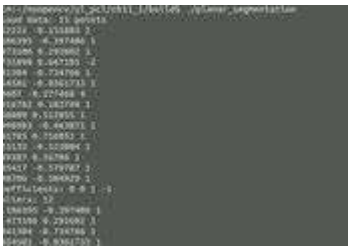
### PCL点云分割（3） (/article/9810517876/)

（1）Euclidean分割 欧几里德分割法是最简单的。检查两点之间的距离。如果小于阈值，则两者被认为属于同一簇。它的工作原理就像一个洪水填充算法：在点云中的一个点被“标记”则表示为选择在一个的集群中。然后，它像病毒一样扩散到其他足够近的点，从这些点到更多点，直到没有新的添加为止。这样，就是一个初始化的新的群集，并且该过程将以剩余的无标记点再次进行。 在PCL中，Eucli...

(/article/8911607377/)

### PCL点云曲面重建 (/article/8911607377/)

在测量较小的数据时会产生一些误差，这些误差所造成的不规则数据如果直接拿来曲面重建的话，会使得重建的曲面不光滑或者有漏洞，可以采用对数据重采样来解决这样问题，通过对周围的数据点进行高阶多项式插值来重建表面缺少的部分， （1）用最小二乘法对点云进行平滑处理 新建文件resampling.cpp 结果对比 (2)在平面模型上提取凸（凹）多边...



(/article/1824928410/)

### PCL点云分割（1） (/article/1824928410/)

点云分割是根据空间，几何和纹理等特征对点云进行划分，使得同一划分内的点云拥有相似的特征，点云的有效分割往往是许多应用的前提，例如逆向工作，CAD领域对零件的不同扫描表面进行分割，然后才能更好的进行空洞修复曲面重建，特征描述和提取，进而进行基于3D内容的检索，组合重用等。 案例分析 用一组点云数据做简单的平面的分割： planar_segmentation.cpp 结果如下：开始打印的数据为手动添加...

(/article/75261081585/)

## [PCL] 点云数据集 (/article/75261081585/)

1 The Stanford 3D Scanning Repository（斯坦福大学的3 d扫描存储库） 链接：http://graphics.stanford.edu/data/3Dscanrep/ 这应该是做点云数据最初大家用最多的数据集，其中包含最开始做配准的Bunny、Happy Buddha、Dragon等模型。 2 Sydney Urban Objects Dataset（悉尼城市目...

---

(/article/99981109577/)

## 硬件：宽带猫（光猫）的基础知识 (/article/99981109577/)

1、宽带猫的概念　　“猫”，又叫做:调制解调器(英文名MODEM)（记住，这才是正经名字），作用是把通过电话线或者光纤进入的信号还原为数据以及把网线产生的数据转换成模拟信号（电信号/光信号），所以，如果是电话线/光纤入户的，都需要安装调制解调器。首先你需要知道的是，常见的信号有两种，数字信号和模拟信号（光信号和电信号）。在计算机里运行的是数字信号，在网线（电话线/光纤）里...

---