# 20131108-20131205讨论总结

### １.郭建伟：

**Acquiring 3D Indoor Environments with Variability and Repetition**

**作者：Young Min Kim，　Niloy J. Mitra，　Dong-Ming Yan，　Leonidas Guibas**

**An Interactive Approach to Semantic Modeling of Indoor Scenes with an RGBD Camera**

**从场景中通过机器学习，分析及聚类，从数据库中找出物体，进行场景三维生成。**

**L1-Medial skeleton of Point cloud**

**从不完整点云中提取骨架。**

**Analyzing Growing Plants from 4D Point Cloud Data**

对带有时间维度的点云进行分析，得到植物的生长过程

**3D Object Recognition based on Correspondence Grouping**

**基本过程：**

* 空间分辨率的计算（spatial resolution computation）：计算点云中每个点到其它点的最小距离，求和后平均即得此值。可选项，控制后续操作的参数大小。
* 法向估计：对于模型和场景中每个点，利用其周围邻域10个点的位置估计法向（ least-square plane fitting ）。
* 提取keypoints: 对模型和场景进行均匀下采样，构造三维体网格（3D voxel grid），用每个格子中所有点的质心代表该网格中的所有点。
* 计算描述符：为每个keypoints关联一个描述信息，使用Signature of Histograms of OrienTations descriptor（SHOT，ECCV2010）。
* 关键点匹配：对场景中的所有关键点，在模型的关键点中进行查找匹配，若距离最小且小于某一阈值的，则将该点对放入到一个数组中去。
* 最后，对5中的结果进行聚类。可使用Hough Voting process方法或geometric consistency clustering algorithm（ PRL2007 , Hui Chen et al. ） 。

其中，可以在参数的调整，算法不同环节，以及测试上进行应用和进一步改进。对于自己的点云数据有一定的效果。但不是特别稳定。

总结：第一、二三次讲的东西用pcl可能实现其部分功能，第四、五次讲的有源代码，但是并不稳定和鲁棒，可考虑在算法细节上进行改进。

另外，对于树的重复性检测，可以做为一个研究的方向。

1. **室内场景点云重建，包括：**

***[1] Acquiring 3D Indoor Environments with Variability and Repetition***

***[2] A Search-Classify Approach for Cluttered Indoor Scene Understanding***

***[3] An Interactive Approach to Semantic Modeling of Indoor Scenes with an RGBD Camera***

这三篇论文都是Siggraph Asia 2012的文章，做的工作是从手持扫描仪（kinect、MantisVision Inc）采集的室内场景点云重建出室内场景模型。其中[1]和[2]采用了机器学习的方法，包括两个阶段：在学习阶段，对场景中的每一类物体单独扫一遍，然后训练学习得到该类物体的识别分类器；在识别阶段，输入场景点云，利用已经训练的分类器对场景中的每个物体进行识别，并从已存在的模型数据库中找到对应的模型进行替换，最后得到重建的三维场景。这两篇论文的学习过程都是基于对单个点云物体的形状分析，抽取单个物体的形状特征表示，但是当输入的点云数据缺失严重的时候，两个方法的识别过程不鲁棒。

论文[3]从RGBD图像（kinect采集）中重建室内场景，该方法首先对RGB图像进行分割，将图像分割为具有语义信息的区域，然后对每个区域，利用其深度信息在已存在的模型数据库中进行检索，找到合适的模型并根据深度信息进行相对位置的摆放。这篇论文的创新点在于利用语义分割区域的深度信息（点云）在模型库中进行检索，但是对数据库的规模有一定要求。

1. **点云处理，包括：**

***[4] L1-Medial skeleton of Point cloud*** 该论文提出一种从点云提取骨架的方法，对点云的质量要求不高。该方法的创新点在于，利用L1-中值的概念，迭代的从邻域内计算中心点代替该邻域内的所有点，和以前的方法相比，不需要计算点云的连接结构，也不需要计算任何法向信息，利用点的空间位置就可以直接计算，而且通过设置惩罚项可以处理不完整的点云数据。

***[5] Analyzing Growing Plants from 4D Point Cloud Data*** 该论文在点云数据的基础上，加上时间维度的信息，跟踪和分析植物的生长过程，该方法首次对植物的生长过程进行动态和真实的分析，应用比较新颖。论文的主要难点在于对于事件的检测，比如对新长出的部分和枯萎消亡部分的检测，因此需要对点云数据进行语义分割和标注。该方法对室内简单的植物适用，对树木等高大的室外植物不适用。

1. **点云物体识别**

***[6] Object recognition in 3D scenes with occlusions and clutter by Hough voting*** 给定一个物体模型点云和一个场景点云，如果场景中存在该物体，该方法可以在场景中识别和找到物体，并得到对应的变换矩阵。该算法的很多步骤比如检测关键点、计算法向和计算描述符比较通用，可以用在其他方面。

### ２.刘欣莹：

**黄慧的文章：Edge-Aware Point Set Resampling**

**Siggraph 2013**

**对采样点云进行边恢复，使得重采样的点云具有良好的保边性。**

Edge-Aware Point Set Resampling

关于边缘重采样从而对噪声点和离群点的处理方法（EAR）

优点：

* 具有良好的保边性
* 排除干扰点噪声
* 点集密度可以自行进行调整

输入为有噪声、有异常点且采样过疏的无规则三维点云，点的方向会比较混乱无序

输出期望是与所在平面尽量贴附的整齐并且最大的保留边缘特性的点集



中心步骤：

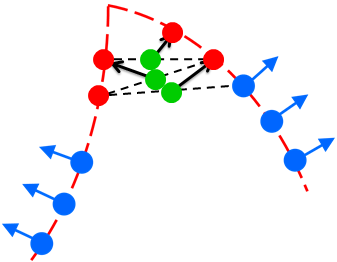
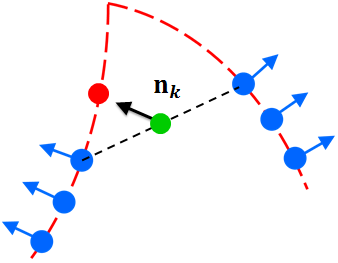
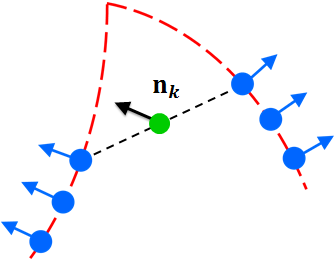
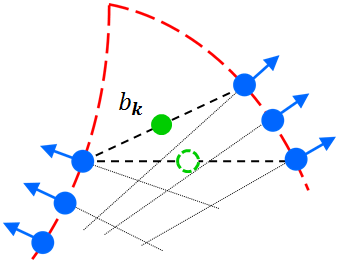
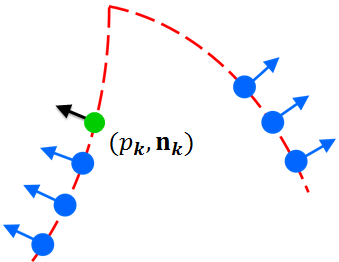
* 对远离边缘点进行采样
* 计算相应可靠法线方向
* 基于已经计算出的法线数据值逐步向边缘拐点进行计算



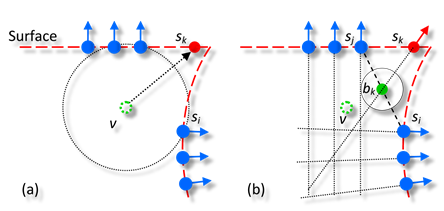
在此过程中比较核心的是对已经采样完成的远离边缘的点进行处理并进一步得到边缘点法线方向。可以看做一个迭代的过程：

对先前得到的已知点进行插值加入新的点，新点需要满足三个条件：

1. 它所依靠的上一点要依附在已知表面上，也就是是有用的点
2. 它的新法线方向要与上一点的所在面垂直
3. 插入的新点要在邻域内均匀分布



针对每一个新插入的点，采用最优化计算找到密度最小的插入位置：



缺点及不足：

当输入噪声过多或输入附近包含其他面片产生干扰时，计算会有影响。另外在输入内部有交大空洞的情况下，此方法不能进行较有效的处理边缘过程。

我没有进行相关实验，在上次组会大家提到的如何从原始输入的点云中判定出哪里是边缘此篇文章没做相关介绍，作者之前的文章中也没有具体解释。

### ３.孔彦：

**High Quality Shape from a Single RGB-D Image under Uncalibrated Natural Illumination**

ICCV2013

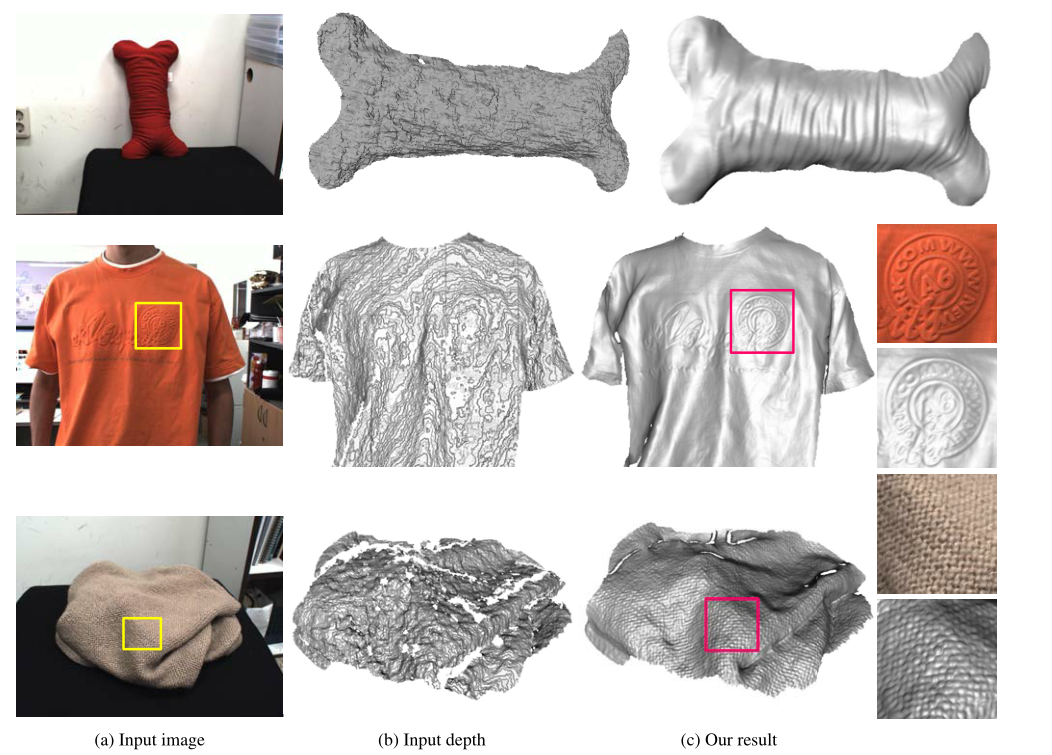
基本过程：

输入: RGB image and low quality depth

1. 对深度图双边滤波
2. 估计一个全局光照模型
3. 估计一个局部光照模型
4. 进行法向估计
5. 深度恢复

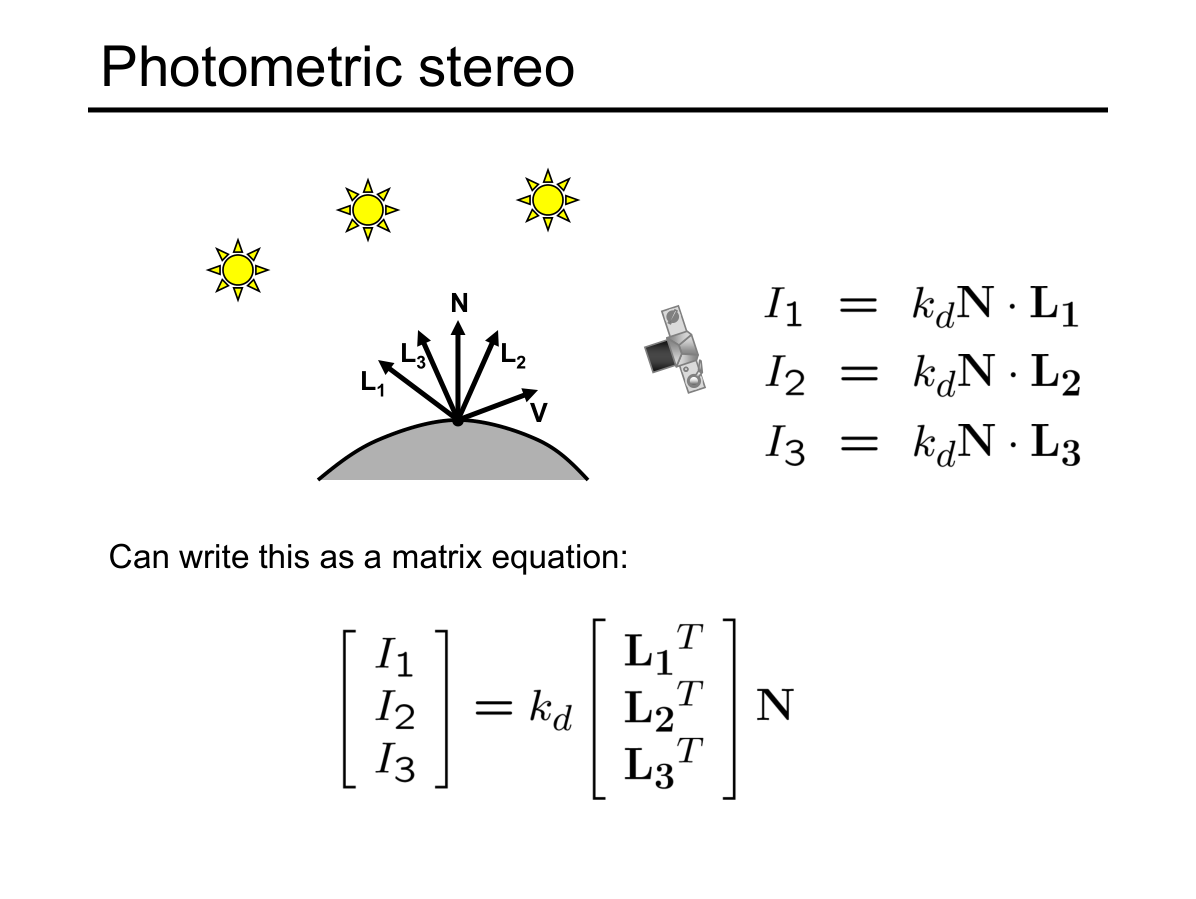
结论：应该比较容易实现，但效果未必有说的那么好。

**High Quality Shape from a Single RGB-D Image under Uncalibrated Natural Illumination**



这篇文章发表于ICCV2013，用shape from shading的方法解决Kinect深度图分辨率低导致的三维重建模型粗糙问题，文章展示的效果还是很好的。

Shape from shading是用光学方法(photometric method)来进行形状估计的主要方法之一，核心思想是用检测到的图像中光强度的变化，通过假设的一个光照模型来反算出模型的法向，进而恢复模型的三维形状信息。



如上图所示，方程组中的*I*可通过采集的图像颜色值得到，右侧L为光源信息，可通过对光源的标定得到，由此，可以求解模型的法向量N。

用shape from shading来增强传统双/多目视觉得到的三维重建结果之前就有人做过很多工作，CVPR2011就有一篇文章，High-quality shape from multi-view stereo and shading under general illumination，用shape from shading来增强多视图像的重建结果。、

这篇文章主要的主要流程是

|  |
| --- |
| Input: RGB image and low quality depth   1. Apply bilateral filtering to the given depth map 2. Estimate a global lighting model 3. Estimate a local lighting model 4. Normal Estimation 5. Depth Recovery |

其特点是不要求场景中的光源事先标定。一般shape from shading需要实现知道光源的信息来进行法向计算，这篇文章直接用Kinect的粗糙深度来初始化优化函数，省去了光源标定的过程。方法的缺点是速度较慢，800×600的图像需要20多分钟的计算。

### ４.姜海勇：

Supervoxels for Point Clouds，

有对应的PCL代码，主要是对带颜色的点云进行有意义的分块。

代码在PCL1.7.1下可编译成功。输入要求点云有颜色。没有对其它点云进行过测试。此方法可用来应用或进一步改进。

创新点：

1.对原始的输入点云进行处理，从而得到对点云的过分割super voxels。 以前没有相关的工作，类似的工作为图像上的super-pixels。

2. 作用： 在分割得到superVoxels后可以利用superVoxels计算相关的信息，从而得到一个更高层次的信息。如果对原始点云直接进行处理，可能计算量会很大，在过分割后减少了这种计算量，同时由于supervoxels内的点云是极为相似，所以对于点云自身的语义信息也没有太大的损失。

3. 进一步思考：

How to combine the multiple segmented 2d images?

How to use supervoxels to get the segmentation of the point clouds?

Can the seeding procedure be replaced by mean-shift algorithm?

### ５.徐士彪：

多视图立体视觉（Multiview Stereo, MVS）配准和重建是基于图像的三维建模中的核心步骤。MVS算法通常用来处理以下三种数据：

* 物体：单一紧密的物体，在其周围拍摄到的图像中能够完全看到这个物体，且能够比较直接的提取其轮廓图并计算其可视外壳。
* 静态场景：目标物体也许部分遮挡或者嵌入在杂乱的环境中，并且视角范围被限制在较少的几个视点，导致无法进行有效的计算（特别是具有建筑物和墙体的外景）。
* 动态场景：移动的障碍物在多张图像中出现在相对静止目标物体的不同位置。

同时，当前的MVS算法可以根据隐含的对象模型被划分成4个类型：

1. **基于体素（voxel-based）的方法：**通过将待重建物体划分为一个个小立体栅格，然后基于这些体素来进行物体的三维重建，即将重建场景的包围盒体素化；选择重建表面的体素投影到可视图像中；利用内容一致性剔除不一致的体素；重复迭代直到收敛。重建前提需要获取三维场景的包围盒信息，重建的精度受到体素分辨率的限制，同时，由于计算的复杂度和内存限制，该方法往往只能用来恢复单个物体的三维模型。（该类方法适用于小型三维重建）

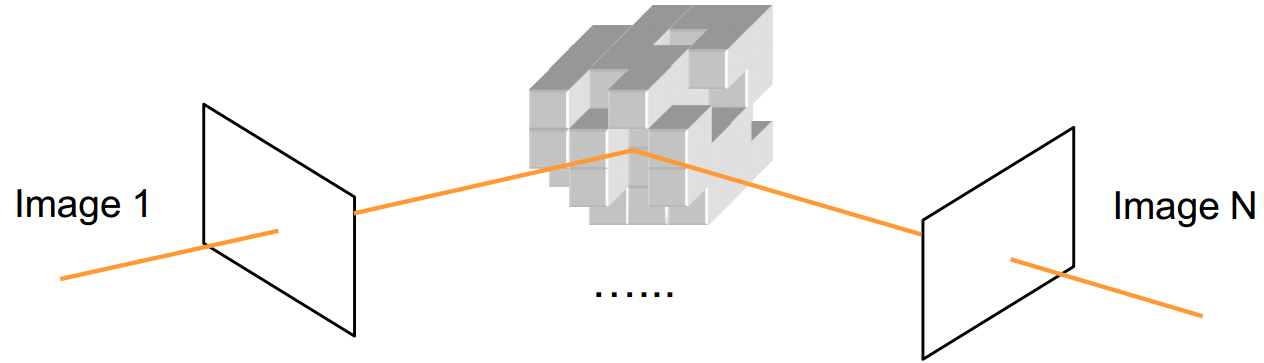
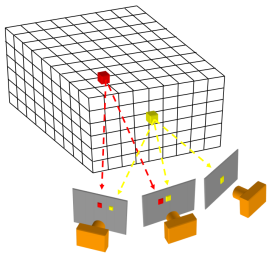


图1 基于体素重建流

**相关文献：**

[1] Seitz S.M., Dyer C.R.. Photorealistic scene reconstruction by voxel coloring. International Journal of Computer Vision, 1999, 35(2):151-173.

[2] Vogiatzis G., Torr P.H.S., Cipolla R.. Multi-view stereo via volumetric graphcuts. IEEE Conference of Computer Vision and Pattern Recognition, pp.391-398.

1. **基于可变多边形网格（deformable polygonal meshes）的方法：**同基于体素的方法一样，需要一个初始的三维模型（虚拟壳）作为优化进程的起点，采用动态规划的方法逐渐收缩初始模型直到收敛。优化的过程中需要计算大量非模型点，严重限制的方法的可扩展性。（该类方法适用于小型三维重建）

**相关文献：**

[3] Faugeras O.D., Keriven R.. Variational principles, surface evolution, PDES, level set method, and the stereo problem. IEEE Transactions on Image Processing, 1998, 7(3):336-344.

[4] Aaharescu A., Boyer E., Horaud R.. Transformesh: A topology-adaptive mesh-based approach to surface evolution. Proceedings of the 8th Asian conference on Computer vision, pp166-175.

1. **基于多深度图像（multiple depth maps）的方法：**基于深度恢复的方法比较灵活，且局限性较少.通常的技术路线为先为每张视图恢复深度图像，然后将不同视图的深度图像整合成一个完整的三维模型。不过需要专门的深度融合算法将多帧深度图像融合成一致的三维表面。（能够生成多个三维模型；能够快速重建模型）

(3.1)运动结构恢复（Structure-from-Motion）

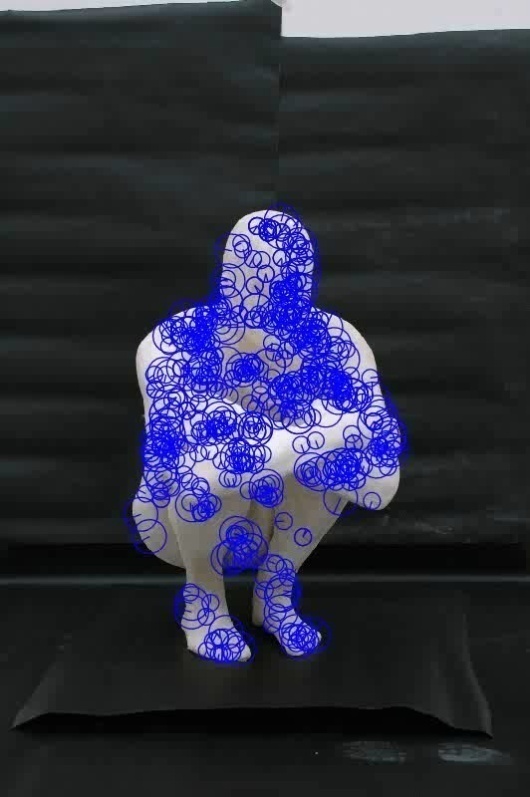
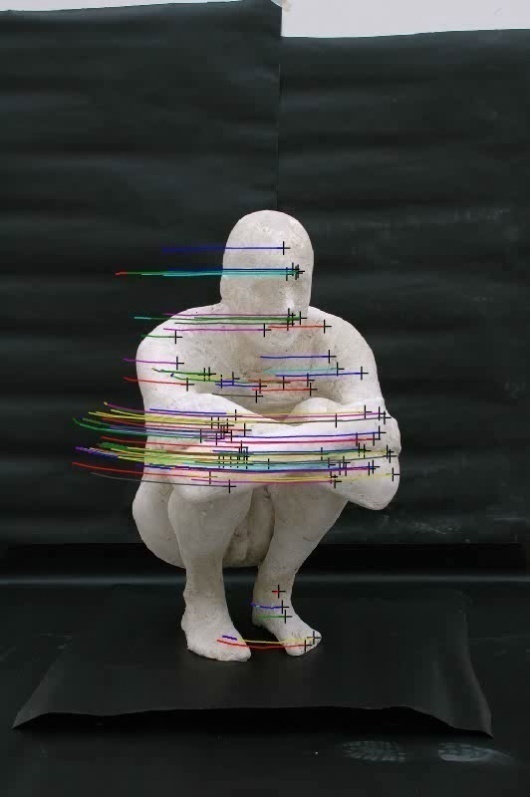
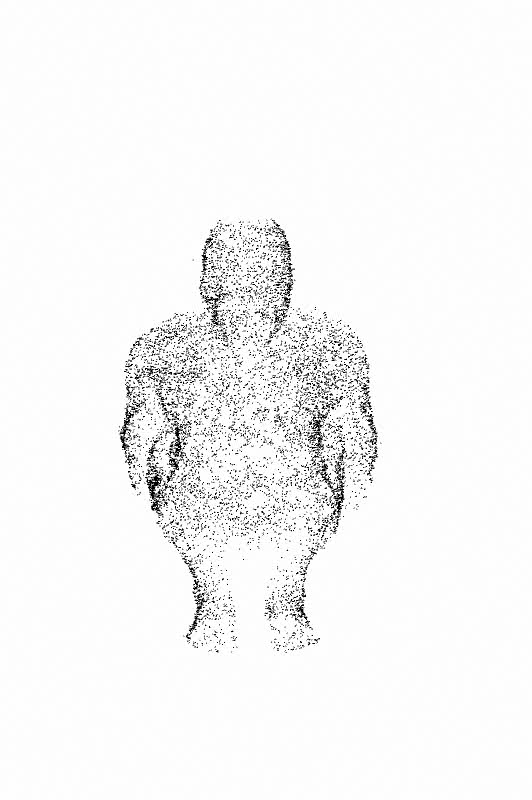


图2 运动结构恢复流程

为了对视频场景进行重建，首先需要恢复摄像机的运动参数，这通常需要借助于运动结构恢复技术。运动结构恢复技术指的是根据物体或场景的运动，分析和推断出其三维结构，主要包括特征点匹配跟踪、摄像机内参数标定和外参数求解等，如图2所示。假定场景静止不变，仅摄像机在运动，那么SFM技术可以从摄像机拍摄的图像序列中恢复出摄像机内外参数及场景的三维结构。

**相关文献：**

[5]Guofeng Zhang, Xueying Qin, Wei Hua, Tien-Tsin Wong, Pheng-Ann Heng and Hujun Bao. Robust Metric Reconstruction from Challenging Video Sequences. IEEE Conference on Computer Vision and Pattern Recognition, 2007.

[6]Guofeng Zhang, Zilong Dong, Jiaya Jia, Tien-Tsin Wong, and Hujun Bao. Efficient Non-Consecutive Feature Tracking for Structure-from-Motion. European Conference on Computer Vision, 2010.

(3.2)稠密深度恢复（Dense Depth Recovery）

基于上述摄像机自动跟踪方法，可以鲁棒的恢复出摄像机的参数和一些稀疏三维点，后续的工作需要从视频序列中恢复出高质量的稠密深度信息。然而，由于实际拍摄的图像或视频数据不可避免地存在着图像噪声、无特征区域以及遮挡等情况，导致自动恢复出高质量的稠密深度依然非常困难。

**相关文献：**

[7]Guofeng Zhang, Jiaya Jia, Tien-Tsin Wong, and Hujun Bao. Consistent Depth Maps Recovery from a Video Sequence. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(6):974-988, 2009.

[8]Guofeng Zhang, Zilong Dong, Jiaya Jia, Liang Wan, Tien-Tsin Wong, and Hujun Bao. Refilming with Depth-Inferred Videos. IEEE Transactions on Visualization and Computer Graphics, 15(5):828-840,2009.

1. **基于面片（patched-based）的方法：**以小块集合的形式来展现场景。这种方法简单有效，并且能够符合基于点的绘制技术的视觉要求，但是需要一个后续的处理步骤将块的集合转化成网格模型，从而能够更加适应基于图像的建模应用。（能够生成单个全景模型；很难快速重建模型）

通常情况下，基于体素和基于可变多边形的重建方法只能对上述第一类数据进行操作；基于深度图和面片的重建方法则可以很好的处理第二类数据；而基于面片的重建方法对第三类数据也能取得较好的处理效果。

实际上，"Accurate, Dense, and Robust Multi-View Stereopsis"报告中所讲到的方法属于上述MVS算法中的第四类，它针对标定好的多视图图片，不需要包围盒、深度等初始化信息，只需要估计点的深度，再通过局部光学连续性估计点的法向。该算法准确、简单和高效，能够自动检测和忽略外部点和障碍点，输出具有方向的小矩形面片密集集合。这些面片获得与像素级别的对应，并且紧密覆盖被观察物体的表面。算法中包括差异函数的定义、图像模型的定义及面片深度不连续的定义都具有一定的新意，值得借鉴。



图3 基于面片重建流程

**相关文献：**

[9] Noah S., Steven M.S., Richard S.. Photo Tourism: Exploring image collections in 3D. ACM Transactions on Graphics, 2006.

[10] Noah S., Steven M.S., Richard S.. Modeling the World from Internet Photo Collections. International Journal of Computer Vision, 2007.

[11] Yasutaka F., Jean P.. Accurate, Dense, and Robust Multi-View Stereopsis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(8):1362-1376, 2009.

[12] Yasutaka F., Brian C., Steven M.S., Richard S.. Towards Internet-scale Multi-view Stereo. IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[13] Jancosek M., Pajdla T.. Multi-View Reconstruction Preserving Weakly-Supported Surfaces. IEEE Conference on Computer Vision and Pattern Recognition, 2011.

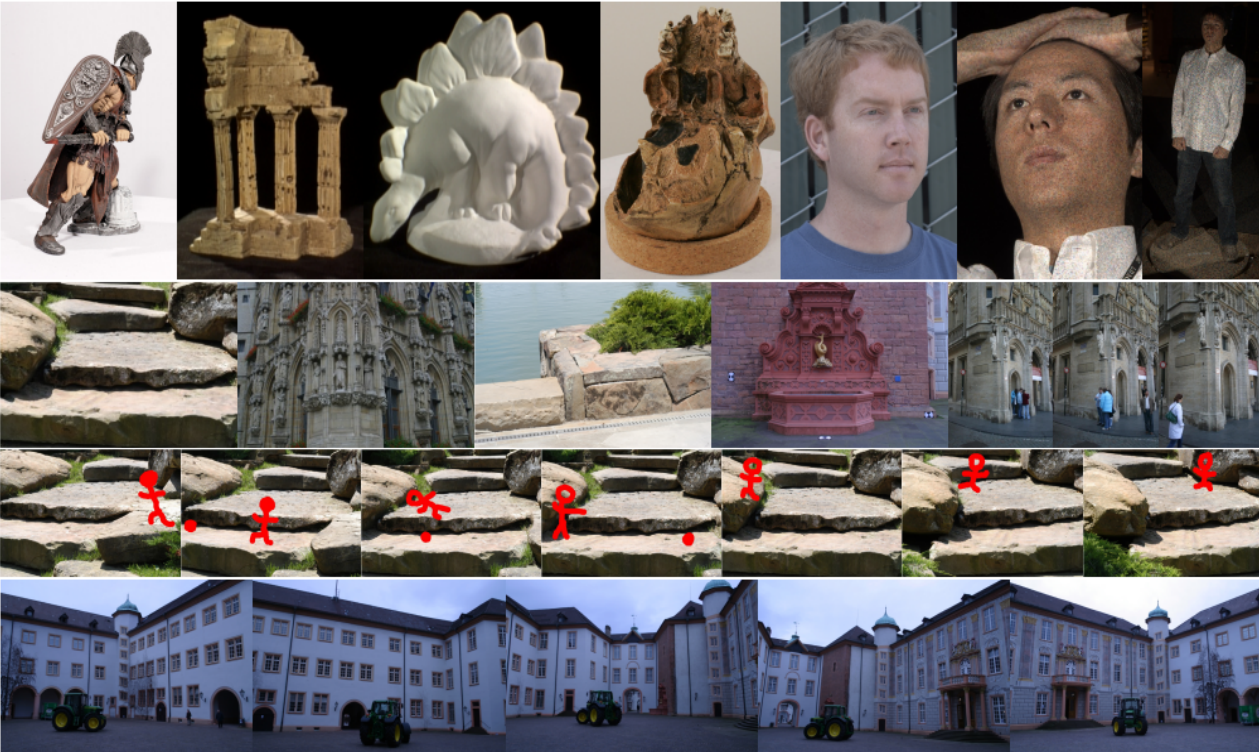


图4 部分输入图像



图5 部分实验结果

### ６.ＰＣＬ相关调研汇总：

PCL可以快速完成以下功能：

1. 给定一个点云，和一个点p的位置，可以快速找出p的N个最近点，或是距离p半径小于r的所有点，N和r由用户定义。使用kdtree或是octree,PCL库用几行代码就可以实现 。octree还可以实现体元检索，给某个点检索其所在个体元内的其它点。Octree树构建时可指定其分辨率，并检测每个体元的变化。
2. 对两个点云进行连结：包括将两个点云并成一个点云，或是在点云上添加新的属性，比如位置点云上添加颜色属性等，使用io操作可快速实现。
3. 可以进行点云的压缩和解压，使用octree。主要用来解决传输问题。
4. 点云模型场景检测匹配过程：输入：模型和场景点云
5. 空间分辨率的计算（spatial resolution computation）：计算点云中每个点到其它点的最小距离，求和后平均即得此值。可使用kdtree
6. 法向估计：利用每个点周围10个点的位置估计法向，算出模型和场景每个点的法向估计值，使用pcl::NormalEstimationOMP
7. 寻找keypoints:对模型和场景进行均匀采样，设置采样半径，得到采样点。
8. 根据keypoints计算描述符，包括模型和场景。
9. 对模型和场景中的所有关键点进行配对。距离最小且小于某一阈值的，放入到一个数组中去。
10. 最后，对5中的结果进行聚类。可使用Hough Voting process过程。它需要每个keypoint的局部坐标系Local Reference Frame (LRF)作为输入参数.返回的是场景中每个模型的变换平移矩阵向量组， 和对应关系向量组，其中每个向量都是场景中的一个模型实例。
11. 最后，可根据这些值进行显示。
12. 点云识别：给定一系列点云作为训练，可识别一个新的点云属于这些点云中的哪类。
13. 点云局部特征的计算：

法向：对点的局部点云作特征值分解，最小特征值对应的特征向量即为法向

曲率：最小特征值除以三个特征值之和，即为曲率。

1. 点云处理：过滤（去掉outlier和noise）->Feature提取（法向和曲率）->提取关键点(keypoints)->registration(通过关键点的对应关系)->点云组织(kd tree, octree)->点云分割(分割成独立的，有意义的部分)->一致性采样（查找具体的模型和及对应参数（圆柱，立方体等等））－>表面重建(根据点云重建原始3D表面，包括三角化和convex　hull等)
2. 根据range image和相机参数生成点云。
3. 输入输出点云格式及快速查看。可实现pcd2ply,ply2pcd,ply2obj功能，不能实现obj2ply功能，可借助meshlab软件实现obj2ply格式转换。
4. 在点云特征方面，使用shape descriptor或是geometry feature作为point feature representations.特征一般要满足：•rigid transformations - that is, 3D rotations and 3D translations in the data should not influence the resultant feature vector F estimation;

•varying sampling density - in principle, a local surface patch sampled more or less densely should have the same feature vector signature;

•noise - the point feature representation must retain the same or very similar values in its feature vector in the presence of mild noise in the data.

使用kd-tree，常用两种操作：k搜索和半径搜索。

•determine the k (user given parameter) neighbors of a query point (also known as k-search);

•determine all the neighbors of a query point within a sphere of radius r (also known as radius-search).

1. PCL1.0中的pcl::PCLPointCloud2在后来的版本中被取代为：sensor\_msgs::PointCloud2
2. 可计算沿ax+by+cz+d=0平面的投影点。使用pcl::ProjectInliers<pcl::PointXYZ> proj;
3. 可以去掉outliers,如在某个半径范围内，点数小于指定个数时，将被作为outlier移去，或是满足用户指定的某个条件。
4. 使用pcl的工具时，按h可在命令窗口显示所有可用的选项。这里选项指按键盘的响应选项。
5. 在使用normalestimation时，点的曲率已经被计算完毕了。如果要计算主曲率，可以再使用如下代码：

pcl::PrincipalCurvaturesEstimation<pcl::PointXYZ, pcl::Normal, pcl::PrincipalCurvatures> pc;

pc.setInputNormals (cloud\_normals1);

vector<int> indices;

indices.resize (basic\_cloud\_ptr->points.size ());

int indices\_size = static\_cast<int> (indices.size ());

for (size\_t i = 0; i < indices.size (); ++i)

indices[i] = static\_cast<int> (i);

tree.reset (new pcl::search::KdTree<pcl::PointXYZ> (false));

tree->setInputCloud (basic\_cloud\_ptr);

boost::shared\_ptr<vector<int> > indicesptr (new vector<int> (indices));

pcl::PointCloud<pcl::PrincipalCurvatures>::Ptr pcs (new pcl::PointCloud<pcl::PrincipalCurvatures> ());

pc.setIndices (indicesptr);

pc.setSearchMethod (tree);

pc.setKSearch (indices\_size);

// estimate

pc.compute (\*pcs);

实践发现，对树木进行曲率计算时无效。

1. 做模型格式转化时，使用-format 0转成文本格式，-format 1转成二进制格式。
2. 在做region grow segment的时候，模型需要颜色信息。

# pcl\_viewer可视化：一目了然

如题所示，可视化的重要性不必多说。在点云数据预处理中，要想知道点云的形状需要可视化; 要想了解精简/去噪/简化/压缩 的结果需要可视化; 配准中，对应点对的显示/对应点对的去除结果/配准变化的过程 需要可视化 ...

### 1.pCL viewer使用方法说明

(1)p,P 键：　切换到点表示方式

(2) w, W键：　切换到wireframe方式（如果存在的话）

(3)s, S 键：　切换到surface显示方式（如果存在的话）

(4)j, J键：　对显示窗口进行截图，存成.png格式

(5)c,C 键：　显示当前的相机/窗口　参数

(6)f, F: 飞行到点模式

(7) e, E: 退出交互方式

(8)q, Q: 退出程序

(9)+/- : 增加/减少点的大小

(10) +/-同时按下alt: 放大/缩小

(11) g, G: 显示尺度网格(开/关)

(12) u, U: 显示查找表（开/关）

(13) r, R同时按下alt　：重置相机到视点　（视点＝(0,0,0)-中心（x,y,z）

(14) Alt加s, S:　启动双目（开/关）

(15)Alt加f, F: 在全屏和原始窗口大小之间进行切换。

(16)l, L:列出所有的几何体和当前映射的颜色句柄，对应的数值可在下面两个命令可用（每个句柄不可能超过10个，对应0到9的数值

(17)alt+0…,9 [+ ctrl] :在不同的几何体句柄切换

(18)0…9[+ctrl] ：在不同的颜色句柄之间进行切换。

(19)shift+左击：　选取一个点

(20)x,X: 开启或关闭左键的选取模式。

### 2. 网上说法：pcl\_viewer

linux 下可直接在命令行输入 pcl\_viewr path/to/.pcd或.vtk可直接显示pcl中的点云文件。

pcl\_viewr几个常用的命令：  
r键: 重现视角。如果读入文件没有在主窗口显示，不妨按下键盘的r键一试。  
j键：截图功能。  
g键：显示/隐藏 坐标轴。   
鼠标：左键，使图像绕自身旋转; 滚轮, 按住滚轮不松，可移动图像，滚动滚轮，可放大/缩小 图像; 右键,“原地” 放大/缩小。  
-/+：-（减号）可缩小点; +(加号)，可放大点。  
pcl\_viewe -bc r,g,b /path/to/.pcd:可改变背景色.  
pcl\_viewer还可以用来直接显示pfh，fpfh（fast point feature histogram），vfh等直方图。  
常用的pcl\_viewer 好像就这些，其他未涉及到的功能可通过pcl\_viewer /path/.pcd 打开图像，按键盘h（获取帮助）的方式获得.

#### 程序中的可视化

##### 简单可视化类

所谓简单可视化类，是指直接在程序中使用，而且不支持多线程。  
必须包含的头文件 #include<pcl/visualization/cloud\_viewer.h>，声明一个可视化类直接 pcl::visualization::CloudViewer viewer ("test"); 即可，它的意思是说，我创建了一个CloudViewer的可视化类，这个可视化窗口的名字叫做test; 显示用viewer.showCloud(cloud) , 要想让自己所创窗口一直显示，则加上 while (!viewer.wasStopped()){ };即可， 或者直接viewr.spin(0);

##### "复杂的"可视化类

以一段程序为例：

**#include <pcl/visualization/pcl\_visualizer.h> //包含基本可视化类**

**#include <pcl/visualization/pcl\_visualizer.h>**

//设置键盘交互函数,按下`space`键，某事发生

**void** **keyboardEvent**(**const** pcl::visualization::KeyboardEvent &event,**void** \*nothing)

{

**if**(event.getKeySym() == "space" && event.keyDown())

next\_iteration = **true**;

}

**int** **main** (**int** argc, **char** \*\*argv)

{

1. 读入点云 source, target

2. 处理读入的数据文件

boost::shared\_ptr<pcl::visualization::PCLVisualizer> view (**new** pcl::visualization::PCLVisualizer("test")); //创建可视化窗口，名字叫做`test`

view->setBackgroundColor(0.0,0,0); //设置背景色为黑色

viewer->addCoordinateSystem(1.0); //建立空间直角坐标系

// viewer->setCameraPosition(0,0,200); //设置坐标原点

viewer->initCameraParameters(); //初始化相机参数

\*\*\*`\*显示的”处理的数据文件“的具体内容\*`\*\*\*

view->registerKeyboardCallback(&keyboardEvent,(**void**\*)NULL); //设置键盘回吊函数

**while**(!viewer->wasStopped())

{

viewer->spinOnce(100); //显示

boost::this\_thread::sleep (boost::posix\_time::microseconds (100000)); //随时间

}

}

在主程序2中，处理显示数据文件包含以下几种:  
一. 计算并显示法向量，具体在自己的笔记[pcl法向量的计算与显示](https://segmentfault.com/n/1330000005761876)点击预览  
二. 我的笔记[画线与显示](https://segmentfault.com/n/1330000006645681)点击预览,可用于配准计算中对应点对的显示，不过用画线的办法***很不好***.  
三. 单纯的自定义的显示点云有如下常用函数：

1. pcl::visualization::PointCloudColorHandlerCustom<pcl::PointXYZ> sources\_cloud\_color(source,250,0,0); //这句话的意思是：对输入为pcl::PointXYZ类型的点云，着色为红色。其中，source表示真正处理的点云，sources\_cloud\_color表示处理结果.
2. view->addPointCloud(source,sources\_cloud\_color,"sources\_cloud\_v1",v1); //将点云source,处理结果sources\_cloud\_color,添加到视图中，其中,双引号中的sources\_cloud\_v1,表示该点云的”标签“，我们依然可以称之为”名字“，之所以设置各个处理点云的名字，是为了在后续处理中易于区分; v1表是添加到哪个视图窗口（pcl中可设置多窗口模式）
3. view->setPointCloudRenderingProperties(pcl::visualization::PCL\_VISUALIZER\_POINT\_SIZE,3,"sources\_cloud\_v1"); //设置点云属性. 其中PCL\_VISUALIZER\_POINT\_SIZE表示设置点的大小为3,双引号中”sources\_cloud\_v1“,就是步骤2中所说的标签。
4. view->setPointCloudRenderingProperties(pcl::visualization::PCL\_VISUALIZER\_OPACITY,1,"sources\_cloud\_v1"); //主要用来设置标签点云的***不透明度***，表示对标签名字为"sources\_cloud\_v1"的标签点云设置不透明度为1,也就是说透明度为0. 默认情况下完全不透明。

四. 显示配准中的对应点对关系.

要想显示点对之间的对应关系, 首先必须计算出对应点对, pcl中对应点对的计算可通过pcl::registration::CorrespondenceEstimation<pcl::PointT,pcl::PointT> correspond\_est;计算,计算出对应点对后, source 和 target对应点的索引会存储在vector<int> A或pcl::Correspondences A中.要想显示点对的对应关系,只需 view->addCorrespondences<pcl::PointXYZ>(source,target,A,"correspond",v1); 其中，pcl::PointXYZ表示所添加对应点对的类型为PointXYZ类型的，参数中的前两个表示目标点云和源点云，A 存储从目标点云到源点云的对应点的索引，”correspond“依然是自定义的标签，v1表示添加到哪个窗口.   
为了使得对应点更加个性化，我们可以对它进行一下”定制“：

1. view->setShapeRenderingProperties(pcl::visualization::PCL\_VISUALIZER\_LINE\_WIDTH,2,"correspond"); //设置对应点连线的粗细.PCL\_VISUALIZER\_LINE\_WIDTH,表示线操作,线段的宽度为2（提醒一下自己: 线段的宽度最好不要超过自定义的点的大小）,"correspond"表示对 **对应的标签** 做处理.
2. view->setShapeRenderingProperties(pcl::visualization::PCL\_VISUALIZER\_COLOR,0,0,1,"correspond"); //设置对应点连线的颜色，范围从0-1之间。

五. 多窗口及人机交互设置.  
具体操作即设置请看[pcl之ICP实现](https://segmentfault.com/a/1190000005930422)  
六. 如果用pcl的可视化类显示直方图，则可以这样做(以fpfh为例)：

**#include <pcl/visualization/histogram\_visualizer.h> //直方图的可视化**

**#include <boost/thread/thread.hpp>**

**#include <pcl/visualization/pcl\_plotter.h>**

**int** **main** (**int** argc, **char** \*\*argv)

{

....

直方图计算

....

pcl::visualization::PCLHistogramVisualizer view;

view.setBackgroundColor(255,0,0);

view.addFeatureHistogram<pcl::FPFHSignature33> (\*fpfhs,"fpfh",1000); //对下标为1000的元素可视化

//view.spinOnce(10000); //循环的次数

view.spin(); //无限循环

**return** 0;

}

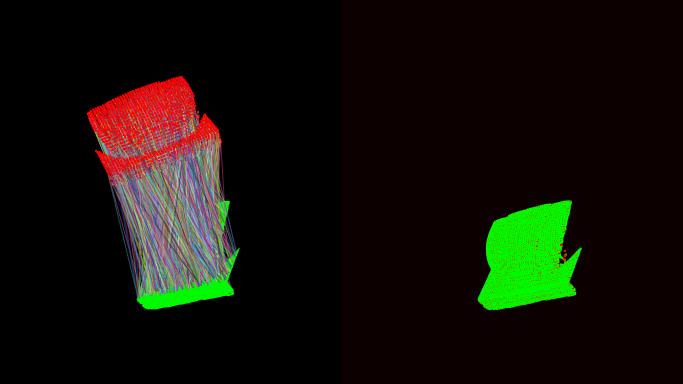
也可以这样显示直方图不过需要在添加头文件#include <pcl/visualization/pcl\_plotter.h>

pcl::visualization::PCLPlotter plotter;

// We need to set the size of the descriptor beforehand.

plotter.addFeatureHistogram(\*fpfhs, 300); //设置的很坐标长度，该值越大，则显示的越细致

plotter.plot();



* [2016年08月22日发布](https://segmentfault.com/a/1190000006685118)

# Ubuntu 16.04 安装PCL库以及测试

参考链接：https://blog.csdn.net/dantengc/article/details/78446600

参考博客，官网一直安装不成功，后来参照一篇博客终于安装成功了，记录如下。

1.需要预安装依赖库。官网上提供了使用PPA安装，比较简单，但是之后安装时有错误。（ubuntu16.04 使用ppa预安装时，最后一句apt-get install libpcl-all 应该更改为apt-get install libpcl-dev（好像前者针对ubuntu14.04，后者才 针对ubuntu 16.04）,虽然可以运行成功，仍然不推荐，因为我采用这种方法后面还是有错误了！)

采用一下方式依次安装依赖库：

sudo apt-get update

sudo apt-get install git build-essential linux-libc-dev

sudo apt-get install cmake cmake-gui

sudo apt-get install libusb-1.0-0-dev libusb-dev libudev-dev

sudo apt-get install mpi-default-dev openmpi-bin openmpi-common

sudo apt-get install libflann1.8 libflann-dev

sudo apt-get install libeigen3-dev

sudo apt-get install libboost-all-dev

sudo apt-get install libvtk5.10-qt4 libvtk5.10 libvtk5-dev

sudo apt-get install libqhull\* libgtest-dev

sudo apt-get install freeglut3-dev pkg-config

sudo apt-get install libxmu-dev libxi-dev

sudo apt-get install mono-complete

sudo apt-get install qt-sdk openjdk-8-jdk openjdk-8-jre

sudo apt-get install libproj-dev

2.下载源码并编译

git clone https://github.com/PointCloudLibrary/pcl.git

cd pcl

mkdir release

cd release

cmake -DCMAKE\_BUILD\_TYPE=None -DCMAKE\_INSTALL\_PREFIX=/usr \

-DBUILD\_GPU=ON -DBUILD\_apps=ON -DBUILD\_examples=ON \

-DCMAKE\_INSTALL\_PREFIX=/usr ..

make -j4 (线程数根据情况选择)

sudo make install

测试

按照以上步骤直接生成.cpp文件运行会发现找不到头文件，原因是需要使用Cmake 编译（包括连接到PCL库等）

具体操作直接参照官网（http://pointclouds.org/documentation/tutorials/using\_pcl\_pcl\_config.php#using-pcl-pcl-config）

步骤包括：

（1）.在project 文件夹下生成 pcd\_write.cpp（见官网上提供的链接）

（2）.生成 CMakeLists.txt,直接复制一下内容（不需要任何修改）：

cmake\_minimum\_required(VERSION 2.6 FATAL\_ERROR)

project(MY\_GRAND\_PROJECT)

find\_package(PCL 1.3 REQUIRED COMPONENTS common io)

include\_directories(${PCL\_INCLUDE\_DIRS})

link\_directories(${PCL\_LIBRARY\_DIRS})

add\_definitions(${PCL\_DEFINITIONS})

add\_executable(pcd\_write\_test pcd\_write.cpp)

target\_link\_libraries(pcd\_write\_test ${PCL\_COMMON\_LIBRARIES} ${PCL\_IO\_LIBRARIES})

（3）.编译运行

cd /PATH/TO/MY/GRAND/PROJECT （project 所在的文件夹）

mkdir build

cd build

cmake ..

make

./pcd\_write\_test

Qt5

On 64 Bit Ubuntu Systems

$ wget http://download.qt-project.org/official\_releases/qt/5.6/5.6.1/qt-opensource-linux-x64-5.6.1.run

$ chmod +x qt-opensource-linux-x64-5.6.1.run

$ ./qt-opensource-linux-x64-5.6.1.run

最简易的PCL安装方法（ubuntu16.04系统）

2018年01月16日 21:04:11

阅读数：1007

在第N次帮学弟学妹们装PCL之后，我发现……

编译个毛线的源码啊！明明已经有了更简单的安装方法，一行命令就搞定：

sudo apt-get install libpcl-dev

这是已经编译好的点云库（PCL-1.7），附带安装了VTK-6.2.0，以及各种杂七杂八的依赖库。

安装后的编译过程中出现了库文件vtkproj4.so丢失的问题。这个问题与我另一篇寻找缺失库文件的文章不同，这是VTK-6.2.0 (build10)中的BUG，而原作者已经在build11中修正了该问题。

这个问题有一个非常简单的解决方法：在/usr/lib文件夹下，我发现了vtk5.10的相关库文件，考虑到5.10与6.2版本间的差距应该不大，我直接进行了软链接……

sudo ln -s /usr/lib/libvtkproj4.so.5.10 /usr/lib/libvtkproj4.so

编译通过！告辞！

如果你不是linux小白，已经成功自己安装了Matlab，CUDA等“较为容易”安装的软件，那么很好，你就有看懂这篇文章的基础了。如果你是linux小白，连 “啥是sudo？哇sudo好神奇！卧槽咋干啥都要sudo” 这种三段式的感叹都能发出来的话，建议还是别先看我这篇文章。。。

我之所以使用ubuntu 14.04，是某个历史原因，我必须使用linux kernel版本低于或等于3.13的LTS OS。所以迫不得已使用14.04 LTS。以下所有安装都是基于14.04上。相信16.04 和12.04 应该也差不多吧。

第一步，打开PCL 官网linux 预安装教程 Prebuilt binaries for Linux. 按照他的说法把第一步完成。

Ubuntu

We currently support all Ubuntu via PPA. The installation instructions are:

sudo add-apt-repository ppa:v-launchpad-jochen-sprickerhof-de/pcl

sudo apt-get update

sudo apt-get install libpcl-all

第二步，打开 PCL Ubuntu documentation：Documentation 从 Experimental 开始按照它的步骤往下做。

Experimental

If you are eager to try out a certain feature of PCL that is currently under development (or you plan on developing and contributing to PCL), we recommend you try checking out our source repository, as shown below. If you’re just interested in browsing our source code, you can do so by visiting GitHub - PointCloudLibrary/pcl: Point Cloud Library (PCL).

Clone the repository:

git clone https://github.com/PointCloudLibrary/pcl pcl-trunk

Please note that above steps (3-5) are almost identical for compiling the experimental PCL trunk code:

cd pcl-trunk && mkdir build && cd build

cmake -DCMAKE\_BUILD\_TYPE=RelWithDebInfo ..

make -j2

sudo make -j2 install

咦？！怎么安装失败了？？？

原来少了一些dependencies。这也是我最讨厌PCL的地方。。各种依赖库。。关键是依赖库还不好找。。基本上大家都会缺少Boost依赖库和VTK依赖库，这些Mandatory是必装的，是运行PCL最基本的要求。

Mandatory

The following code libraries are required for the compilation and usage of the PCL libraries shown below:

pcl\_\* denotes all PCL libraries, meaning that the particular dependency is a strict requirement for the usage of anything in PCL.

LogoLibraryMinimum versionMandatoryBoost

1.40 (without OpenNI)

1.47 (with OpenNI)

pcl\_\*Eigen3.0pcl\_\*FLANN1.7.1pcl\_\*VTK5.6pcl\_visualization

安装Boost的时候比较easy的啦，根据我的仔细研究（StackOverFlow），发现原来安装Boost居然这么简单！运行一行apt-get install 即可！

You can use apt-get command (requires sudo)

sudo apt-get install libboost-all-dev

安装VTK就比较蛋疼了！VTK的文档简直难找……好在我这里提供了一个我能用的不错的文档：VTK/Building/Linux:

VTK/Building/Linux

< VTK‎ | Building

Contents

1 Download

2 Configure

2.1 Qt Setup

2.1.1 Qt5

2.1.2 Qt4

3 Build

Download

Download the source code from the VTK Download page or Clone with Git

git clone git://vtk.org/VTK.git VTK

Configure

Create a build directory which is separate from the source.

mkdir VTK-build

cd VTK-build

ccmake /path/to/VTK

Set any options you would like using the curses interface. Alternatively, set the options at the command line.

mkdir VTK-Release-build

cd VTK-Release-build

cmake -DCMAKE\_BUILD\_TYPE:STRING=Release /path/to/VTK

Qt Setup

Qt5

In order to build with the latest Qt5 release (Qt5.2.1), take the following steps:

Download the Qt.5.2.1 offline installer for linux and run it

mkdir qt5.2.1-install && cd qt5.2.1-install

wget http://download.qt-project.org/official\_releases/qt/5.2/5.2.1/qt-opensource-linux-x64-5.2.1.run

chmod +x qt-opensource-linux-x64-5.2.1.run

./qt-opensource-linux-x64-5.2.1.run

Install to a separate directory

Configure VTK with the following options

cd /path/to/VTK-Release-build

cmake -DVTK\_QT\_VERSION:STRING=5 \

-DQT\_QMAKE\_EXECUTABLE:PATH=/path/to/qt5.2.1-install/5.2.1/gcc\_64/bin/qmake \

-DVTK\_Group\_Qt:BOOL=ON \

-DCMAKE\_PREFIX\_PATH:PATH=/path/to/qt.5.2.1-install/5.2.1/gcc\_64/lib/cmake \

-DBUILD\_SHARED\_LIBS:BOOL=ON

/path/to/VTK

Qt4

The latest patch release (Qt4.8.6) does not appear to have an installer for linux on the Qt Downloads page so the code must be built from source.

mkdir qt-4.8.6-build && cd qt-4.8.6-build

wget http://download.qt-project.org/official\_releases/qt/4.8/4.8.6/qt-everywhere-opensource-src-4.8.6.tar.gz

tar xzf qt-everywhere-opensource-src-4.8.6.tar.gz

cd qt-everywhere-opensource-src-4.8.6

./configure # go through the dialogue

make -j<# of cores>

Then in the VTK build directory, configure VTK with the path to the Qt build

cmake -DQT\_QMAKE\_EXECUTABLE:PATH=/path/to/qt-4.8.6-build/qt-everywhere-opensource-src-4.8.6/bin/qmake \

-DVTK\_Group\_Qt:BOOL=ON \

-DBUILD\_SHARED\_LIBRARIES:BOOL=ON \

/path/to/VTK

Build

Once cmake finishes successfully configuring, use make inside the build directory.

make -j<# of cores>

根据文档，按顺序把VTK和Qt5 安装好（Qt4就不必安装了）。记住，Qt5是必装的！！！也就是PCL依赖于VTK，VTK依赖于Qt5！

而这个“算是不错的”文档里，还有一个错误！就是下文引用部分：

cd /path/to/VTK-Release-build

cmake -DVTK\_QT\_VERSION:STRING=5 \

-DQT\_QMAKE\_EXECUTABLE:PATH=/path/to/qt5.2.1-install/5.2.1/gcc\_64/bin/qmake \

-DVTK\_Group\_Qt:BOOL=ON \

-DCMAKE\_PREFIX\_PATH:PATH=/path/to/qt.5.2.1-install/5.2.1/gcc\_64/lib/cmake \

-DBUILD\_SHARED\_LIBS:BOOL=ON

/path/to/VTK

倒数第二行，少了一个反斜杠“\”！！！所以倒数第二行应该是：-DBUILD\_SHARED\_LIBS:BOOL=ON \

不然就会报错！尼玛啊！鬼知道啊！怕是pcl用的人太少了，在网上查了一圈居然也没有人po到stackoverflow上。。。。

好在我现在po到知乎上了。。。希望能够拯救一下大家宝贵的时间。

个人经验：最后一篇这个很有用，一定要编译VTK。不用生成VTK-build，生成VTK-Release-build就OK。不能两个都 生成。

# Ubuntu16.04 安装 Visual Studio Code之后启动不起来

2017年04月06日 15:05:12

阅读数：6735

一、写在前面的废话

网上有很多Ubuntu怎么安装vscode的博文,好多种方法,今天就不赘述了,记录一下我自己碰到的问题.

二、问题描述

照着网上的方法安装vscode,基本每一种方法(umake,dpkg,.zip包,.deb包,我能找到的所有方法,哦,对,大神说,能用安装包的安装的最好不要用umake)都尝试了,结果怎么都启动不起来,无奈,请教大神(总是去烦大神,大神的内心一定在咆哮,没办法,谁叫我解决不了呢),大神就是大神,10分钟都没用上就解决了(哎,我什么时候能成为大神,/憧憬脸).

三、发现问题

大神点来点去,没一会就发现:

vscode的配置文件被加上了root权限

(我发誓不是我故意的,我压根就不知道该怎么加root权限)

四、解决问题

把配置文件的root权限去掉就好了

cd ~/.config 番外(1)

sudo rm -rf ./Code/

然后输入密码就好了

猜测原因:vscode在打开的时候需要改动一些配置文件,但是启动的时候发现文件夹加了root权限,他改不了,所以就启动失败

番外

(1)~/.config

.config目录就在Home目录下,默认是被隐藏了的

显示方法:最上面菜单栏 -> 查看 -> 显示隐藏文件

好啦,到这我遇到的问题就解决啦,希望能帮到遇到同样问题的小伙伴~~~

# 对‘inflateValidate@ZLIB\_1.2.9’未定义的引用

2018年03月28日 11:25:05

阅读数：800

第一次遇见这个错误是编译caffe的时候，然后网上一大堆解决方案

比如

在 Makefile.config 中，加入下一句

LINKFLAGS := -Wl,-rpath,$(HOME)/anaconda/lib

不知道为什么，anaconda自己安装的opencv不能读取摄像头，然后就从源码安装

编译的时候又出现这个问题，之前的办法不能用了。

发现ubuntu的zlib版本是1.2.8

然后就去官网下载链接下载了

尴尬的是官网没有1.2.9,下了1.2.11,安装后居然就不报错了。。。。

zlib1g-dev\_1.2.11.dfsg-0ubuntu2\_amd64.deb

zlib1g\_1.2.11.dfsg-0ubuntu2\_amd64.deb

我大概就是下载的这两个文件

# **准确率与召回率（Precision & Recall）**

**准确率**和**召回率**是广泛用于信息检索和统计学分类领域的两个度量值，用来评价结果的质量。其中精度是检索出相关文档数与检索出的文档总数的比率，衡量的是检索系统的**查准率**；召回率是指检索出的相关文档数和文档库中所有的相关文档数的比率，衡量的是检索系统的**查全率**。

一般来说，Precision就是检索出来的条目（比如：文档、网页等）有多少是准确的，Recall就是所有准确的条目有多少被检索出来了。

正确率、召回率和 F 值是在鱼龙混杂的环境中，选出目标的重要评价指标。不妨看看这些指标的定义先：

    1. 正确率 = 提取出的正确信息条数 /  提取出的信息条数

    2. 召回率 = 提取出的正确信息条数 /  样本中的信息条数

**两者取值在0和1之间，数值越接近1，查准率或查全率就越高。**

    3. F值  = 正确率 \* 召回率 \* 2 / (正确率 + 召回率) （F 值即为正确率和召回率的调和平均值）

不妨举[这样一个例子](http://peghoty.blog.163.com/blog/static/49346409201302595935709/)：某池塘有1400条鲤鱼，300只虾，300只鳖。现在以捕鲤鱼为目的。撒一大网，逮着了700条鲤鱼，200只虾，100只鳖。那么，这些指标分别如下：

正确率 = 700 / (700 + 200 + 100) = 70%

召回率 = 700 / 1400 = 50%

F值 = 70% \* 50% \* 2 / (70% + 50%) = 58.3%

不妨看看如果把池子里的所有的鲤鱼、虾和鳖都一网打尽，这些指标又有何变化：

正确率 = 1400 / (1400 + 300 + 300) = 70%

召回率 = 1400 / 1400 = 100%

F值 = 70% \* 100% \* 2 / (70% + 100%) = 82.35%

由此可见，正确率是评估捕获的成果中目标成果所占得比例；召回率，顾名思义，就是从关注领域中，召回目标类别的比例；而F值，则是综合这二者指标的评估指标，用于综合反映整体的指标。

当然希望检索结果Precision越高越好，同时Recall也越高越好，但事实上这两者在某些情况下有矛盾的。比如极端情况下，我们只搜索出了一个结果，且是准确的，那么Precision就是100%，但是Recall就很低；而如果我们把所有结果都返回，那么比如Recall是100%，但是Precision就会很低。因此在不同的场合中需要自己判断希望Precision比较高或是Recall比较高。如果是做实验研究，可以绘制**Precision-Recall曲线**来帮助分析。

**2、综合评价指标（F-Measure）**

P和R指标有时候会出现的矛盾的情况，这样就需要综合考虑他们，最常见的方法就是F-Measure（又称为F-Score）。

**F-Measure是Precision和Recall加权调和平均**：

当参数α=1时，就是最常见的F1，也即

可知F1综合了P和R的结果，当F1较高时则能说明试验方法比较有效。

**3、E值**

E值表示查准率P和查全率R的加权平均值，当其中一个为0时，E值为1，其计算公式：

b越大，表示查准率的权重越大。

**4、平均正确率（Average Precision, AP）**

平均正确率表示不同查全率的点上的正确率的平均。

PCL1.8.0中，

# 错误：error C2653: “sensor\_msgs”: 不是类或命名空间名称

解决方法3：sensor\_msgs改为pcl

说明：在PCL1.8中将PCLPointCloud2加入了pcl名字空间，sensor\_msgs是旧的方式。注意原来 通常是：sensor\_msgs：：PointCloud2

现在应该改成：pcl:: PCLPointCloud2

对应的：

"pcl::fromROSMsg is deprecated, please use fromPCLPointCloud2 instead.

# 安**装tensorflow**

# 首先 1、conda create -n 环境名称 python=需要的版本号 ；2、source activate 环境名称 ；3、conda install tensorflow=1.4

虚拟环境一定要创建并激活，然后安装对应的tensorflow才行

然后在激活的虚拟环境中调用Python并import tensorflow才能成功

发现环境下的某个软件版本不匹配时，比如cudnn7.1.3需要改变为7。0。5

则可以执行

conda install cudnn=7.0.5

系统会自动安装软件进行降级 。

另外，tensorflow1.4.1和tensorflow-gpu 1.4.1不是同一版本，请注意 。

# Ubuntu 16 安装pycharm专业破解版方法

2018年03月20日 22:12:06

阅读数：2963

        之前转载了一个在Windows环境下pycharm专业破解的安装的文章，今天为了在Linux环境下安装使用odoo10，所以尝试在Linux环境下安装pycharm专业破解版看看。

windows下安装的文章在<http://blog.csdn.net/mzl87/article/details/79492684>

        言归正传，我现在来说说在ubuntu16下安装pycharm及破解的方法。首先我们要下载pycharm的安装包，

地址为<https://www.jetbrains.com/pycharm/download/#section=linux>

下载的默认地址为当前用户的下载目录下，如下图

可以直接右击文件=>提取到此处进行解压缩（喜欢用命令解压的同学自便）

解压好后打开终端设备，输入命令

sudo mv 下载/pycharm-2017.3.4 /opt

将pycharm解压包移动到opt文件夹下，然后在终端设置中输入命令对pycharm文件夹付权限

sudo chmod -R 744 /opt/pycharm-2017.3.4

在安装之前请首先修改hosts文件，在终端设备中输入命令

sudo gedit /etc/hosts

在打开的文件中加入0.0.0.0 account.jetbrains.com，如下图

接下来需要把文件JetbrainsCrack-2.6.10-release-enc.jar下载并放到pycharm-2017.3.4/bin文件夹中，文件下载地址为

<https://pan.baidu.com/s/1suF5f3byC1EoIjQRZNTIuA>  密码为：5rp7

然后在进入终端设备，输入命令

cd /opt/pycharm-2017.3.4/bin/

进入后输入命令

sh ./pycharm.sh

启动pycharm 的安装，在等待一段时间后会自安装完成并打开，我这里直接介绍输入验证码的部分。

如下图所示，选择第二个（activation code），然后进入<http://idea.lanyus.com/>,点击

获取注册码，将注册码复制到下面的方框里或者在激活页面的license server输入：

http://idea.liyang.io/，点击OK

ps：建议选择第二个，第三个试了失败

至此pycharm安装成功！！（初始化设置这里不介绍）