点云的理解与应用

by [redacted]；邮箱 zhangxian@zerozero.cn；2019-7-11

1 知识来源

```
@PhDThesis{RusuDoctoralDissertation, author = {Radu Bogdan Rusu}, title =
{Semantic 3D Object Maps for Everyday Manipulation in Human Living
Environments}, school = {Computer Science department, Technische Universitaet
Muenchen, Germany}, year = {2009}, month = {October} }
```

第一个 release 版本：2010年[1]；

当前最新的稳定版本是2018年出的 `pcl.1.9.1` ，编程语言是 `C++` [2]；

期待中的版本：`pcl 2.0` 目前尚未 release[3]。

2 点云背景

点云定义：论文中原名 point cloud，官方定义为 it is a data structure used to represent a collection of multi-dimensional points and is commonly used to represent three-dimensional data. In a 3D point cloud, the points usually represent the X, Y, and Z geometric coordinates of an underlying sampled surface. When color information is present, the point cloud becomes 4D；笔者译为多维点的集合，**一般用来表示三维数据**。例如图1，一个三维点云，其中的点代表其下样本表面的 X, Y, Z 坐标；**如果有颜色信息，则点云是四维的**。

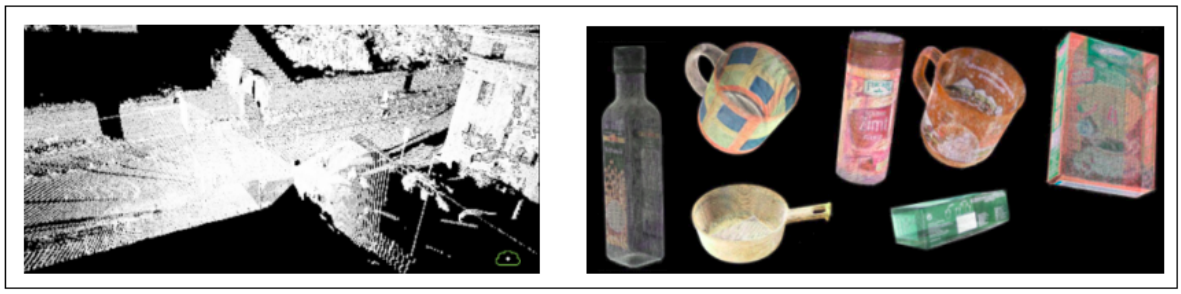

图1 - 3D点云和4D点云

引入背景：$2D$ 图像无法描述真实空间的**语义**[4，5]，一方面是由于单目相机本身设备缺陷（$e.g.$ 无法得到深度图像）和数据局限，另一方面机器视觉如果利用 $2D$ 图像直接匹配 $3D$ 空间的物体，准确率低[6]。

作者贡献：

- 通过点云创建新型的 $3D$ 语义模型框架，包括数据配准（或对齐），特征点提取，分割，建图等；
- 提出和基于 $3D$ Point Feature Histogram Descriptor，使用点云数据为每个点建模局部表面几何形状。

笔者理解：

- 一个三维语义建图框架，包含不同模块的接口和具体实现；
- 研究着重特征点提取。

每个人理解的侧重点和兴趣点不同，笔者后续着重特征点相关概念和应用。

3 概念理解

3.1 获取点云

获取方式如下[6]

- 直接方式：TOF 系统，如激光测量系统，LiDAR，TOF相机，声纳；
- 间接方式：三角化技术，例如需要标定的双目；

优缺点[6]：

- 结构光传感器没有被作者用以采集点云数据，因为论文是室内机器人应用，而结构光传感器不用在此方面[6]，主要用在受约束条件下的 $3D$ 建模（**此处不用还是很少用，建议查看原论文，以方便我们相机选型**）；
- 激光测量系统配合云台，获取速度快，频率高达500 HZ，但得到的原生数据是 $2D$ 图像，需要转换才能生成$3D$ 点云，且结果在真实空间的应用不好；
- TOF相机获取点云速度较快，但容易受材质不反射或其他光线影响；
- 双目得到的点云非完整或稠密的。

笔者只是知识的搬运工，对上述设备的应用场景目前在了解过程中，当下不做评论；但作者论文完成于2009年，受限于当时技术手段，论文中的观点有待查证，这里不做深究。

3.2 特征点

$3D$ **点**：论文原文 $3D$ points，指 $3D$ 感知系统所需的基本输入数据，提供离散且有意义的 $3D$ 空间信息。此处大家可以想想，输入数据是什么，离散是什么，何为有意义等。

点云表征： 点云数据除了包含 $3D$ 空间的坐标信息，也可以包含传感器到物体表面的距离信息以及其他信息，例如颜色，标签，几何形状等。

邻域点：要想知道离散点云的特征信息，需要查询点云中每个点形成的特征信息。而邻域点恰好可以提供该点邻域组成的几何形状特征信息。数学表达式

$$||p_i^k - p_q||_x \leq d_m \qquad (1)$$

d_m 指邻域点到查询点最大距离，这里用 $\mathrm{Minkovski}$ 距离，大家可以用其他距离。

搜索邻域点的方法：作者采用的是

- k-search[6]
- radius-search[6]

但他建议采用 Approximate Nearest Neighbor Search, ANN[7，8]。k 和 r 选择会涉及到尺度问题，作者的解决方案考虑了密度一致性和多尺度统计估计。

3.3 局部几何信息

局部几何信息 Local Geometric Information：根据邻域点定义，局部几何信息定义为**特征向量 Feature Vector**, F

$$F(p_q, p^k) = x_1, x_2, x_3, \ldots, x_n$$

$x_i, i \in (1, 2, \ldots, n)$ 代表生成的特征向量第 i 维特征表示，对比两个点 p_1, p_2 会产生特征向量 F_1, F_2 的差异，定义相似性度量公式如下

$$\text{similarity} = d(F_1, F_2)$$

d 代表差异上的度量，如果 $d = 0$ 则表示 p_1, p_2 两点关于他们的特征相似，两个点也是相似的，反之则差异较大。一般好点和坏点特征存在明显差异，好点特征满足：

- 刚性变换：点在$3D$ 空间平移或者旋转，该点的特征向量 F 差异不大；
- 样本密度：局部表面采样点云密度不同时，不论稀疏或稠密，特征向量签名一致；
- 噪音：好点即便有噪音点影响，特征向量差异不大。

3.4 法线和曲率

法线引入背景：对于局部几何特征差异而言，邻域点可以用来描述查询点下方样本表面的几何特征。而描述样本表面的几何特征首要的就是推测坐标系的方向，即估计法线。

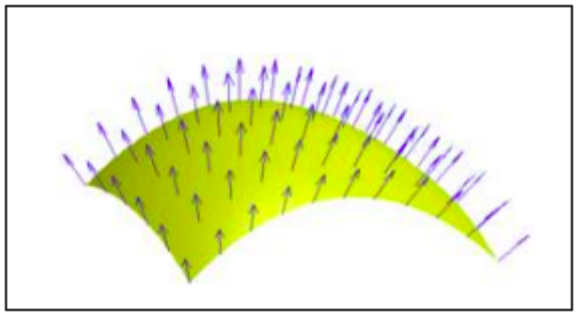

图2 法线示意图

公式

$$x = \bar{p} = \frac{1}{k}\sum_{i=1}^{k} p_i \qquad (2)$$

x 是 P^k 的质心， 求解法线向量 $\vec{n}$ 就成了求解经过质心平面上质心处的法线，利用到了协方差

$$\begin{aligned} C \qquad &= \tfrac{1}{k}\textstyle\sum_{i=1}^{k} w_i (p_i - \bar{p}) \dot{c} (p_i - \bar{p})^t \\ C \cdot \vec{v}_j \quad &= \lambda_j \cdot \vec{v}_j, \quad j \in \{0, 1, 2\} \end{aligned} \qquad (3)$$

w_i 一般为1，是p_i的权重；C 是半正定矩阵，其特征值是 $\lambda_j \in \mathbf{R}$. 当 $\lambda_0, \lambda_1, \lambda_2$ 算出时，可以计算出对应的特征向量 $\vec{v}_0, \vec{v}_1, \vec{v}_2$. 而它们就是法线向量 $+\vec{n} = \{n_x, n_y, n_z\}$ 或 $-\vec{n} = \{n_x, n_y, n_z\}$ 。根据上式

$$\begin{aligned} \phi \quad &= \arctan(\tfrac{n_z}{n_y}) \\ \theta \quad &= \arctan\frac{\sqrt{(n_y^2 + n_z^2)}}{n_x} \end{aligned} \qquad (4)$$

上述是用一对角度表示**法线**，然而，法线向量涉及 $+/-$ 号，无法通过数学方式求解。作者利用到了图优化理论，把正负号的确定当作模型一致性优化的参数。总之，上述法线计算既适用于光滑表面，也适用于不光滑表面。

曲率：计算法线向量需要求解协方差矩阵的特征值和特征向量，而这个过程也伴随着Principal Components Analysis, PCA 分析，可以进一步求解查询点周围的曲率。假设存在一点 $p_i \in P^k$ ，根据协方差矩阵求得$\lambda_0, \lambda_1, \lambda_2$ 且 λ_0 是最小的

$$\sigma_{p_i} = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \qquad (5)$$

σ_{p_i} 近似等于 p_i 的邻域点的曲率变化，且有尺度不变性。该值越小，说明邻域点几乎都在该表面 p_i 处的切面上。作者此处提出协方差矩阵创建时引入外点判断机制和优化 k 参数[9]，则

$$w_i = \begin{cases} \exp(-\frac{d_i^2}{\mu^2}), & p_i^k = \text{outlier} \\ 1, & p_i^k = \text{inlier} \end{cases} \tag{6}$$

d_i 指查询点 p_q 到邻域点的距离，必须满足 $d_i \leq d_{\max}$ ；μ 指查询点 p_q 到其 k 个邻域点 p_i^k 的平均距离。协方差引入外点判断机制，使尖边处法线计算也能正常进行，同时使边界处物体分割更加精确。作者得到的结果如下图

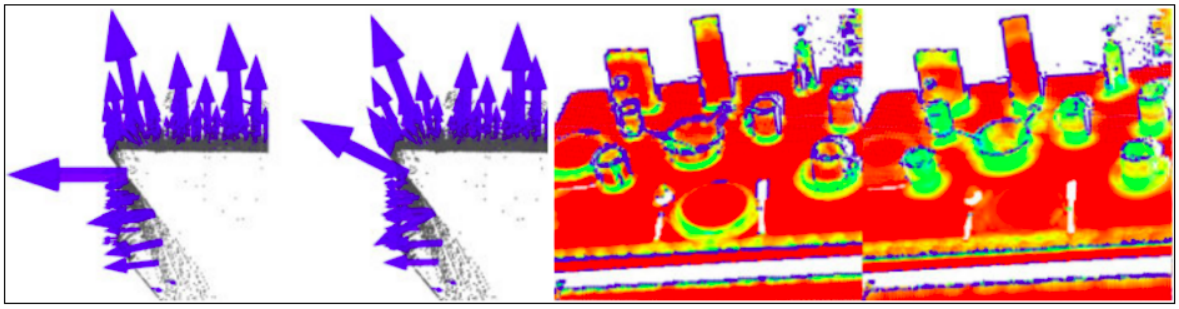

图3 优化 k 参数和协方差矩阵带来的结果

3.5 Point Feature Histograms

PFH **引入背景**：用法线和曲率表征查询点附近表面的几何形状是原理简单且计算很快的一种方法。然而它能提供的信息有限且点的数据过多时，特征都极为相似。为了更好区分不同特征，提出了**Point Feature Histogram**

原理：定义一特征空间，该空间叫做**双环邻域 Dual-Ring Neighborhood**. 假设 P 是点云数据集，存在一点 $p_i \in P$ 满足

$$\text{双环邻域} = \begin{cases} r_1 & \rightarrow P^{k_1} \\ r_2 & \rightarrow P^{k_2} \end{cases} \quad \text{with} \quad 0 < k_1 < k_2, r_1 < r_2 \tag{7}$$

r_1, r_2 分别代表以查询点为中心，不同的两个半径。由于 $r_2 > r_1$ ，故可想象查询点邻域第二层包含着第一层，类似同心圆。其中，其第一层点云是 P^{k_1} ，是 PCA 分析法线和曲率的 k_1 个邻域点；第二层点云是 P^{k_2} ，是 PFH 特征表征的组成部分。但PFH需要用高维直方图生成查询点 p_q 邻域点的平均曲率才能编码 P^{k_2} 的几何属性。而高维直方图和高维超空间相关联，能对特征信息提供签名，对邻域点下方表面的 $6DoF$ 姿态具有不变性（姿态不变局部特征）且对样本密度和噪音都有鲁棒性。签名通俗点讲就是专属或身份，故 PFH签名过程是下面这样的

- 假设 $p_i, p_j \in P^{k_2}$, 计算点 p_i, p_j 对应的表面法线 $\vec{n}_i, \vec{n}_j$ ；
- 如果满足

$$\begin{aligned} &\arccos(\vec{n}_i \cdot \vec{p}_{ji}) \leq \arccos(n_j, \vec{p}_{ij}) \\ &\text{where} \quad p_{ji} = p_j - p_i, p_{ij} = p_i - p_j \end{aligned} \tag{8}$$

则

$$\begin{cases} p_s = p_i, n_s = n_i \\ p_t = p_j, n_t = n_j \end{cases} \tag{9}$$

p_s, p_t分别是源点和目标点，否则

$$\begin{cases} p_s = p_j, n_s = n_j \\ p_t = p_i, n_t = n_i \end{cases} \tag{10}$$

根据作者定义的坐标系

$$\begin{cases} u = n_s \\ v = u \times \frac{p_t - p_s}{||p_t - p_s||_2} \\ w = u \times v \end{cases} \quad (11)$$

见下图

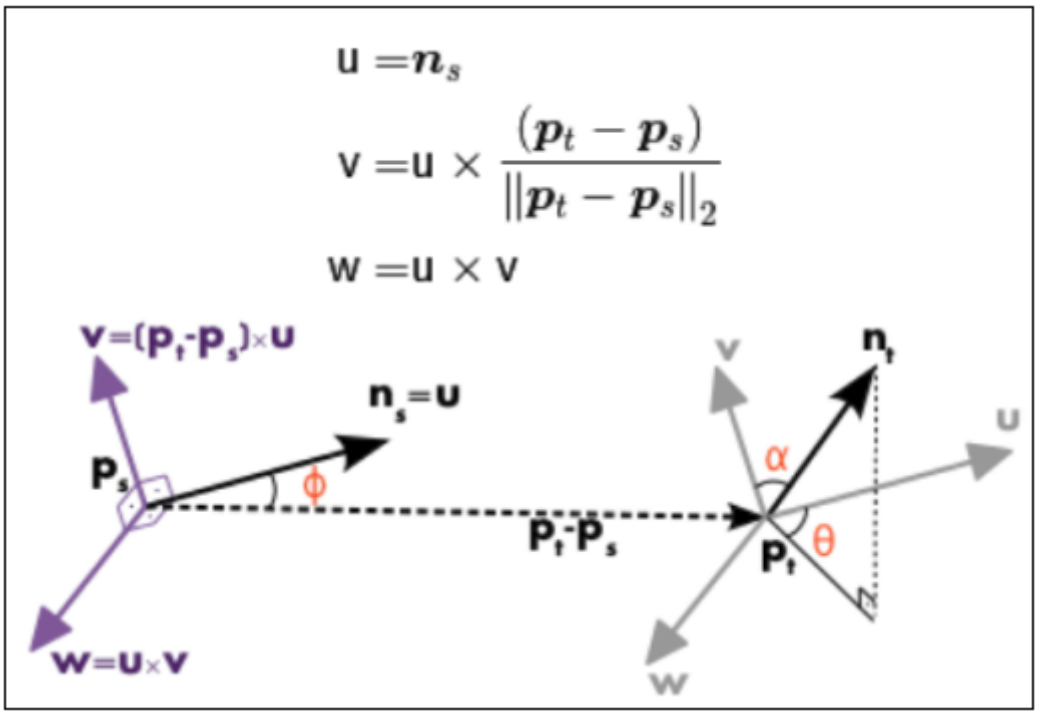

图4 作者定义的uvw坐标系

转换成四元组

$$\begin{aligned} \alpha &= v \cdot n_t \\ \phi &= u \cdot \frac{p_t - p_s}{d} \quad (4.20) \\ \theta &= \arctan(w \cdot n_t, u \cdot n_t) \\ d &= ||p_t - p_s||_2 \end{aligned} \quad (12)$$

基于四元组，生成查询点的 PFH 描述子。大致过程是：

- 遍历点云，将计算得到的四元组按照四个值的范围分组，假设分成了 m 组（PCL默认5组）；
- 由于前三个都是角度，因此可以归以化处理，则分组步长值相等（例如弧度，角度等）；
- 竖坐标设定为四个值出现的次数（**某组竖坐标不为0，就说明该组有特征值；如果多个组不为 0，说明 PFH 有多个特征值，代表不同的几何形状或属性**），横坐标设定为 m^4 个分组，最终形成直方图。

总之，

$$d(PFH_1, PFH_2) = \sum_{i=1}^{m^3} \min(PFH_1^i, PFH_2^i) \quad (13)$$

就可以计算差异了。

3.6 衍生特征描述子

由于 PFH 时间复杂度很高，提出了 Fast Point Feature Histograms，$FPFH$，降低时间复杂度的同时，保留了 PFH的区分能力。

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k}\sum_{i=1}^{1}\frac{1}{w_k}\cdot SPFH(p_k) \quad (14)$$

作者也曾发表了View Point Histograms，VFH 描述子，但论文没有涉及 $NARF, SIFT, BRIEF$ 描述子。

3.7 特征一致性

如何正确选择 k 和 r . **这是个难点，2011年前都是难点**。该理论叫 Feature Persistence or Saliency，主要研究尺度选择问题。作者用到的方法是

- 曲率突变或法线方向偏差确定 k [10]；
- 利用不同半径下 $FPFH$ 直方图的特征均值和标准差计算最优 r，进而计算 k [11]。

3.8 过滤器

过滤背景：**对同一不变场景**，尽管有上述方法确定查询点周围的邻域点，但由于遮挡，深度不连续，跳跃边界等因素，得到的邻域点在短暂的相邻时间内仍然会有部分离群点，需要剔除。

原理：

$$p^* = p_q^* \in P|(\mu_k - \alpha\cdot\sigma_k) \leq \bar{d}^* \leq (\mu_k + \alpha\cdot\sigma_k) \qquad (15)$$

简而言之，利用的是距离和统计信息[9]。

3.9 配准

背景：传感器在不同位置扫描生成的点云数据不同，如何将不同数据整合起来形成完整的模型，该过程叫 Registration 配准。

原理：迭代最近点算法 iterative closest point，其思想是通过旋转、平移使得两个点集之间的距离最小[12]。

3.10 分割

简单几何模型：主要包括平面，线条，柱面，球面，锥面，二元多项式或样条等。

模型拟合算法：RANSAC，事实上特征匹配遇到错误匹配时，需要用RANSAC算法进行过滤。例如平面模型匹配，作者使用不共线的三个点和RANSAC进行 （$ax + by + cz + d = 0$）平面的匹配。

原理：作者使用的是区域增长分割，利用曲率和法线不同进行分割[13]。步骤是

- step 1: 对点云每个点计算曲率，结果排序，找到最小曲率值（可能很多点的曲率值的最小值十分相似）并将其放进种子点集；
- step 2: 对于点集中的种子点（1）计算该种子点与其邻域点里的每个点的法线角度差，如果小于设定阈值，则（2）计算他们的曲率之差，如果小于设定阈值，则添加到新的种子集，该种子集即为当前平面；
- step 3: 设置最小点簇的最小点数和最大点数；
- step 4: 基于第3步，重复1-2。未通过检验，则直接划分为其他集合的种子点集；通过则将该点从原始点云中去除；
- step 5: 算法运行直到不满足最小点数，且对最终生成的平面标记不同的颜色加以区分。

4 实际应用

4.1 测试目的

两个：

- 验证PCL库运行效率，看是否可以在消费级无人机上协助建图和避障；
- 测试点云聚类计算效率。

4.2 测试方法

针对不同目的，提出2种方法：

- 针对 Features，Registration的一些官方实例，看测试标准能否满足；
- 假设点云数据有20000个点或一定数量的点，依次对每个点进行聚类分析，最终得到的聚类是有多少，PC上耗时多少；

补充：**测试标准**的指标两个（可根据个人选择，此处供参考）

- 时间：相同程序迭代5次，计算平均运行时间，本文用clock_t，参考[14]
- CPU使用率：空闲CPU百分比变化[15]，由于浮动变化，采取估算（也可以用平均值或上下限）

打开终端，执行

```
top

//观察%Cpu(s)一行，注意空闲百分比的变化
- 6.7% us — 用户空间占用CPU的百分比
- 0.4% sy — 内核空间占用CPU的百分比
- 0.0% ni — 改变过优先级的进程占用CPU的百分比
- 92.9%id — 空闲CPU百分比
- 0.0% wa — IO等待占用CPU的百分比
- 0.0% hi — 硬中断（Hardware IRQ）占用CPU的百分比
- 0.0% si — 软中断（Software Interrupts）占用CPU的百分比
```

4.3 电脑配置

笔者是 Thinkpad E490，参数如下

```
cat /proc/cpuinfo                   //Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz
free -m                             //内存: total 7821 MB
getconf LONG_BIT                    //CPU型号：64位
cat /proc/cpuinfo | grep "physical id" | sort | uniq | wc -l     //CPU 1个
cat /proc/cpuinfo | grep "cpu cores" | wc -l                     //每个CPU 8个
core
```

4.4 测试结果

【1 特征提取】

序号	测试项目	模块	名称	耗时（s）	整体耗时（s）
1	程序运行时间	模块1	生成点云数据	0.000779	0.02465
		模块2	生成深度图像	0.0197	
		模块3	提取NARF特征+索引	0.003957	
2	CPU使用率(%)	8%~13%			

表1 NARF特征提取实际操作效率

【2 法线估计】

- 官方数据集 Table_scene_mug_stereo_textured， 307200个数据点，没有过滤，分割等额外操作

table_scene_mug_stereo_textured					
序号	测试项目	模块	名称	耗时（s）	整体耗时（s）
1	程序运行时间	模块1	下载点云数据	0.0901	1.342
		模块2	法线计算	1.241	
2	CPU使用率(%)	7%~12%			

表2 使用积分图进行法线估计（2万点+）实际操作效率

- 个人数据集：test.pcd，5个点，没有过滤，分割等额外操作

test_pcd					
序号	测试项目	模块	名称	耗时（s）	整体耗时（s）
1	程序运行时间	模块1	下载5个点云数据	0.0004114	0.0004358
		模块2	法线计算	0.0000202	
2	CPU使用率(%)	7%~16%			

表3 5个点计算法线实际操作效率

【3 配准】

- 官方数据集：capture0001.pcd ~ capture0004.pcd，ICP配准，点个数见备注

capture0001.pcd~capture0005.pcd						
序号	测试项目	模块	名称	耗时（s）	整体耗时（s）	备注
1	程序运行时间 capture0001-capture0002	模块1	downsample	1.098	1.8973	1.pcd有2499647个点
		模块2	计算法线和曲率	0.4921		2.pcd有249931个点
		模块3	ICP配准	0.3072		3.pcd有248494个点
2	程序运行时间 capture0002-capture0003	模块1	downsample	1.078	2.1086	4.pcd有244573个点
		模块2	计算法线和曲率	0.4839		
		模块3	ICP配准	0.5467		
3	程序运行时间 capture0003-capture0004	模块1	downsample	1.073	2.0495	
		模块2	计算法线和曲率	0.4989		
		模块3	ICP配准	0.4776		
4	CPU使用率(%)	7%~12%				

表4 ICP配准官方数据实际操作效率

- 个人数据集：5个点，ICP配准

序号	测试项目	模块	名称	耗时（s）	整体耗时（s）
1	程序运行时间	模块1	创建5个点云数据	0.001671	0.0007164
		模块2	平移或旋转点云	0.0000022	
		模块2	ICP配准	0.0006916	
2	CPU使用率(%)	7%~16%			

表5 ICP 5个点配准实际操作效率

【4 过滤器】

【1】滤波								
序号	方法	英文名	数据来源	操作前数据个数	操作后数据个数	耗时 (ms)	平均耗时 (ms)	效率排序
1	直通过滤器	passthrough filter	无序点云，随机生成	20000	9948	2	4.34405E-06	1
2	体素格过滤器	VoxelGrid filter	table_scene_lms400.pcd	460400	41049	1556	0.00337967	4
3	统计过滤器	statical filter	table_scene_lms400.pcd	460400	451400	8069	0.017526064	6
4	投影模型过滤器	project_inliers	无序点云，随机生成	20000	20000	37	0.00185	4
5	提取点云索引	extract indices	table_scene_lms400.pcd	460400	41049	1631	0.003542572	5
6	条件过滤器	ConditionalRemoval filter	无序点云，随机生成	20000	20000	11	0.00055	2
7	截取框过滤器	CropHull filter	table_scene_lms400.pcd	460400	13211	1431	0.003108167	4

表6 过滤器实际操作效率

【5 分割】

【2】Segmentation											
序号	方法	英文名	数据来源	操作前数据个数	操作后数据个数	删除外点耗时 (ms)	平均耗时 (ms)	分割(ms)	分割平均耗时ms	备注	分割效率排序
1	平面分割	planar_seg.	无序点云，随机生成	20000	19997	74	0.0037	包含在0.0037	包含在0.0037	无	2
2	柱面分割	cylinder seg.	table_scene_mug_stereo_textured	307200	139897	20	6.51042E-05	13176	0.094183578	计算法线耗时	3
3	欧式聚类分割	eulidean clustering seg.	table_scene_lms400.pcd	460400	41049	1069	0.002321894	790	0.019245292	无	3
4	区域增长分割	region growing seg.	region_growing_segmentation.pcd	108104	-	7264	0.067194553	2121	0.019619996	计算法线耗时	3
5	最小切割分割	min cut seg.	min_cut_segmentation_tutorial.pcd	9311	-	0	0	667	667	直通过滤器太快	未知
6	条件欧式聚类分割	condtional eulidean seg.	Status_4.pcd	19553780	202529	18430	0.000942529	11273	0.055661165	发现耗时0.0825	3
7	法线不同分割	difference of normal seg.	don.pcd	108104	2611	1235	0.011424184	38	0.000351513	kdtree搜索	1
8	超体聚类	supervoxels seg.	milk_cartoon_all_small_clorox.pcd	307200	413 supervoxels	5530	0.018001302	899	2.176755448	分割耗时存疑	4
9	地面点云分割	ProgressiveMorphologicalFilter seg.	samp11-utm	38010	18667	99741	2.624072612	包含在平均耗时	包含在平均耗时	无	4
10	模型外点移除分割	ModelOutlierRemoval	无序点云，随机生成，加噪	20005	20000	11	0.000549863	包含在平均耗时	包含在平均耗时	无	1
11	密度聚类	DBSCAN	随机生成2万个点 p(x,y)	20000	20000	19233	0.96165	包含在平均耗时	包含在平均耗时	分割精度差	5
12	机器学习聚类分析	KNN/K-Means	-	-	-	-	-	-	-	效果同DBSCAN	-

表7 分割算法实际操作效率

5 总结

- 非官方的聚类分析算法（如KNN，K-means, DBSCAN等）效率低；
- 法线计算很耗时，聚类分析相对高效，实时性强；
- 过滤较简单，具体工程应用要根据情况写。

参考文献

[1] https://en.wikipedia.org/wiki/Point_Cloud_Library

[2] https://github.com/PointCloudLibrary/pcl/releases

[3] https://github.com/PointCloudLibrary/pcl2

[4] https://en.wikipedia.org/wiki/Semantic_mapping_(literacy)

[5] https://blog.csdn.net/shelden_/article/details/79941783

[6] http://mediatum.ub.tum.de/doc/800632/941254.pdf

[7] https://blog.csdn.net/u011736771/article/details/85103293

[8] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale Feature Extraction on Point-Sampled Surfaces. Computer Graphics Forum, 22(3):281–289, September 2003

[9] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3D Point Cloud Based Object Maps for Household Environments. Robotics and Autonomous Systems Journal (Special Issue on Semantic Knowledge), 2008

[10] Mark Pauly, Richard Keiser, and Markus Gross. Multi-scale Feature Extraction on Point-Sampled Surfaces. Computer Graphics Forum, 22(3):281–289, September 2003

[11] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning Point Cloud Views using Persistent Feature Histograms. In Proceedings of the 21st IEEE International Conference on Intelligent Robots and Systems (IROS), Nice, France, September 22-26 2008

[12] http://www-evasion.inrialpes.fr/people/Franck.Hetroy/Teaching/ProjetsImage/2007/Bib/besl_mckay-pami1992.pdf

[13] https://blog.csdn.net/liukunrs/article/details/80482788

[14] https://blog.csdn.net/yuyin86/article/details/6616566

[15] https://blog.csdn.net/qingrenufo/article/details/79076599