第四章作业讲评

主讲人 陆一帆

作业1（方法选型）

地面点云分割

1. 一帧点云数据中有很多非地面点云，Outliers数量较大。
   - 排除直接最小二乘拟合
   - Hough Transform、RANSAC 二选一

Hough Transform – Summary

Advantage

- Robust to noise
- Robust to missing points of the shape
- Can be extended to lots of models

Disadvantage

- Doesn't scale well with complicated models
  - Usually works for models with less than 3 unknown parameters

RANSAC - Summary

Advantages

- Simple and general
- Usually works well in practice, even with low inlier ratio like 10%

Disadvantages

- Need to determine the inlier threshold τ
- Need large number of samples when inlier ratio is low

作业1（代码思路）

深蓝学院 shenlanxueyuan.com

RANSAC平面拟合

1. 确定迭代次数N、*inlier ratio r*和阈值*tau*
2. 对每一次迭代
    2.1 随机选取三个点(过滤掉3点共线的case)，求这3个点构成的平面的法向量(3个点构成的2个向量叉乘)
    2.2 遍历所有点，计算点Pt到平面距离(即平面上某点到Pt的向量在平面法向量上的投影长度)
        2.2.1 距离小于阈值*tau*为内点
        2.2.2 距离大于阈值*tau*为外点
    2.3 内点比例达到*r*停止迭代，否则返回2继续迭代
3. 确定使得内点数量最多的平面(这么做的前提是：地面一般情况下是包含点数量最多的平面)

作业1(核心代码提示)

深蓝学院 shenlanxueyuan.com

1

```python
# 取三个随机点
idxs = [np.random.randint(0, data.shape[0]) for _ in range(3)]
pts = data[idxs]
# 计算平面法向量
p0p1 = pts[1] - pts[0]
p0p2 = pts[2] - pts[0]
```

2

求平面法向量

```python
nor_vec = np.cross(p0p2, p0p1)
norm = np.linalg.norm(nor_vec)
nor_vec /= norm
```

3

根据平面上某个点pts[0]和平面法向量，计算任意点pt到平面的距离

```python
pt = data[i]
dist = math.fabs(np.matmul(nor_vec, (pt - pts[0]).T) / np.linalg.norm(nor_vec))
if dist < tau:
    inlier_vote += 1
```

4

记录下得票最高的平面的法向量和该平面上任意一点的坐标。

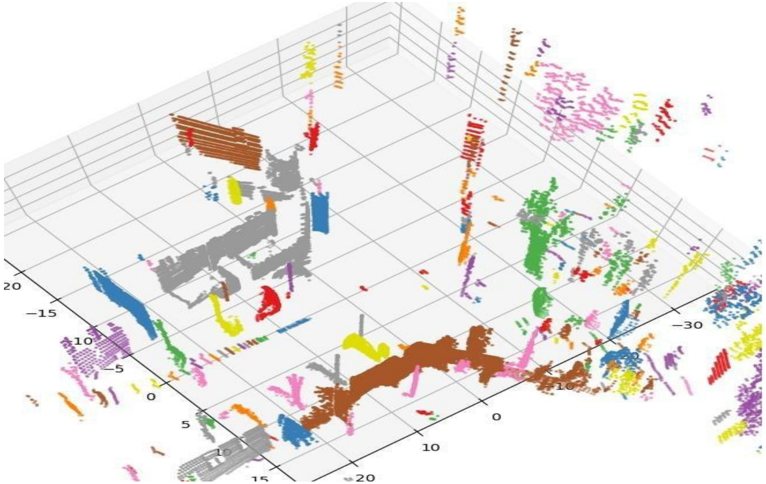
```python
if inlier_vote > max_inlier_vote:
    nor_vec_final = nor_vec
    max_inlier_vote = inlier_vote
    pt_final = pts[0]
```

作业2 (方法选型)

	K-Means	GMM	Spectral	Mean Shift	DBSCAN
Metric	Euclidean	Euclidean	Similarity	Density /Euclidean	Density /Euclidean
# of clusters	Pre-defined	Pre-defined	Heuristic	Automatic	Automatic
Robustness to outlier	Bad	Medium	Good	Good	Good
High dimension data	Medium	Medium	Good	Bad	Bad
Complexity	$O(t \cdot k \cdot n \cdot d)$ t: iteration k: # of clusters n: # of data d: dimension	$O(t \cdot k \cdot n \cdot d)$ t: iteration k: # of clusters n: # of data d: dimension	$O(n^3)$ n: # of data	$O(Tnlog(n))$ n: # of data T: # of centers	$O(n \cdot \log(n))$ n: # of data ✔

作业2(代码思路)

1. 将所有的点都标记为未被访问
2. 构造Kd-Tree，确定radius(大概设置为0.5到1.0)和min_samples (大概设置为4到10)两个参数
3. 从未访问点集合中随机取一个点p，标记p为被访问，radius-NN查找所有邻居
   3.1 若邻居数小于min_samples，标记p为噪点;
   3.2 若邻居数大于等于min_samples，则p为core point，创建新簇C，转步骤4
4. 遍历p的所有邻居n，若n未被访问，将n的类别标记为C，若邻居n也为core point， 重复步骤4
5. 重复步骤3和4，直到所有点都被访问



作业2（核心代码提示）

深蓝学院
shenlanxueyuan.com

1. Dbscan相关参数初始化

```
class DBSCAN(object):
    def __init__(self, radius=0.5, Min_Pts=10):
        self.radius = radius
        self.Min_Pts = Min_Pts
```

2. 随机从一个点开始分类

```
def fit(self, data):
    N = data.shape[0] # data:点云输入, N*3
    lables = -1 * np.ones(N) # 记录每个点的类别
    visited = np.zeros(N) # 记录点是否被访问到
    unvisited = list(range(N)) # 待访问点的id队列
    neighbor_unvisited =[] # 优先访问队列
    label = -1 # 类别初始化
    # 建立kdtree
    tree_root = kdtree.kdtree_construction(data, leaf_size=32)
    # 只要还有点没访问到，就不退出
```

3. 处理该点的邻居点

```
    # 只要还有点没访问到，就不退出
    while len(unvisited) > 0:
        ind = unvisited.pop()
        # TODO: 检查并修改访问状态

        # 搜索附近点
        n_ids = get_neighbor_ids(tree_root, data[ind, :])
        # 判断是否为噪声点
        if len(n_ids) < self.Min_Pts:
            lables[ind] = -1 # 躁点统一归为 "label = -1"
            continue
        else:
            label += 1
            lables[ind] = label
            neighbor_unvisited.extend(n_ids)
            while (len(neighbor_unvisited) > 0):
                ind = neighbor_unvisited.pop()
                # TODO: 检查并修改访问状态，设置label

                # kdtree邻近搜索
                nn_ids = get_neighbor_ids(tree_root, data[ind, :])
                # TODO: 如果是核心点，把临近点加入neighbor_unvisited
```

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听！

Thanks for Listening