深蓝学院
shenlanxueyuan.com

三维点云处理第四章作业讲评

主讲人 冯拓

题目

Object detection pipeline for lidar

- Use KITTI 3D object detection dataset, select 3 point clouds, do the followings.
- Step 1. Remove the ground from the lidar points. Visualize ground as blue.
  - Any method you want – LSQ, Hough, RANSAC
- Step 2. Clustering over the remaining points. Visualize the clusters with random colors.
  - Any method you want
- Step 3. Classification over the clusters
  - Homework of Lecture 5
- Step 4. Report the detection precision-recall for three categories: vehicle, pedestrian, cyclist
  - Homework of Lecture 5

评分原则

深蓝学院
shenlanxueyuan.com

- 优秀：地面分割正确；地物聚类正确；
- 良：地面分割或地物聚类正确；
- 不合格：其他情况。

整体流程

深蓝学院
shenlanxueyuan.com

- 读取数据；
- 预处理；
- 地面分割；
- 删除地面点，做聚类。

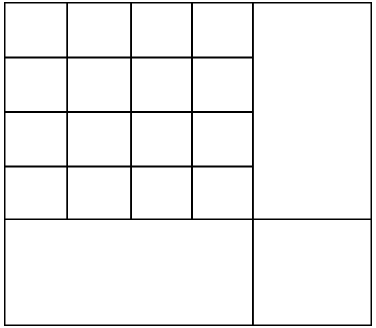
预处理

- 点云降采样（Voxel）；
- 高度滤波；
- 分块分割；

```
z_filter = data[:, 2] < lidar_height
z_filter_down = data[:, 2] > lidar_height_down
filt = np.logical_and(z_filter_down, z_filter)
data_filtered = data[filt, :]
```

参考yuanxun的作业

x >= 0, y >= 0
x >= 0, y < 0
x < 0, y < 0
x < 0, y >= 0



4个大区域

19个小区域

每个子区域中分别使用RANSAC

参考陈贤波的作业

第一题 地面分割 RANSAC

- 算法流程

1、确定迭代次数；

2、在迭代次数内：

  2.1 随机选择三个点组成平面（判断三个点是否共线）；

  2.2 构造坐标矩阵；

  2.3 求平面方程；

  2.4 求所有点到平面的距离；

  2.5 统计inlier个数（距离小于阈值）；

3、迭代选择inlier个数最多的平面。

确定迭代次数

深蓝学院 shenlanxueyuan.com

- 指定迭代次数；
- 计算理论迭代次数；

$$N = \frac{log(1-p)}{log(1-(1-e)^s)}$$

```
inlier_ratio = 0.5
iteration_num = math.ceil(math.log(1-0.99) / math.log(1-pow(inlier_ratio, 3)))
print(iteration_num)
```

判断三个点的关系

- 判断三个点是否共线；

  1、满足满秩矩阵

  2、利用比例关系

```python
while True:
    sample_index = random.sample(range(sz),3)
    p = data[sample_index,:]
    if np.linalg.matrix_rank(p)==3:
        break
```

```python
vector1 = xyz[1,:] - xyz[0,:]
vector2 = xyz[2,:] - xyz[0,:]

# 共线性检查 ; 0过滤
if not np.all(vector1):
    # print('will divide by zero..', vector1)
    return None
dy1dy2 = vector2 / vector1
# 2向量如果是一条直线，那么必然它的xyz都是同一个比例关系
if  not ((dy1dy2[0] != dy1dy2[1])  or  (dy1dy2[2] != dy1dy2[1])):
    return None
```

求平面方程 计算abcd

- 计算平面方程：

```
#求由x点组成的的平面的方程
a = (X[1,1] - X[0,1])*(X[2,2] - X[0,2]) - (X[2,1] - X[0,1])*(X[1,2] - X[0,2])
b = -(X[1,0] - X[0,0])*(X[2,2] - X[0,2]) + (X[2,0] - X[0,0])*(X[1,2] - X[0,2])
c = (X[1,0] - X[0,0])*(X[2,1] - X[0,1]) - (X[2,0] - X[0,0])*(X[1,1] - X[0,1])
ABC = np.zeros((3,1))
ABC[0] = a
ABC[1] = b
ABC[2] = c
d = np.dot(X[0,:],ABC)[0]
print('a',a,'b',b,'c',c,'d',d)
```

- 计算距离：

```
#求所有点到平面的距离
vector = data - X[0,:]
distance = np.dot(vector,ABC)/np.linalg.norm(ABC)
distance = np.abs(distance)
```

$$d = \frac{|Ax_1 + By_1 + Cz_1 + D|}{\sqrt{A^2 + B^2 + C^2}}$$

参考ESOman的作业

求平面方程　点法式

深蓝学院 shenlanxueyuan.com

- 点法式：

平面 π：

π 上一点：　$M_0\,(x_0, y_0, z_0)$

垂直于 π 的法向量：　$n = (A, B, C)$

则：　$n \bullet \overline{M_0 M} = (A, B, C) \bullet (x - x_0, y - y_0, z - z_0) = 0$

- 计算距离：

$$d = \frac{\overrightarrow{M_0 M_1} \cdot \vec{n}}{\|\vec{n}\|}$$

```python
# 2. solve model: 计算平面单位法向量 n
p12 = p2 - p1
p13 = p3 - p1
n = np.cross(p12, p13)
n = n / np.linalg.norm(n)  # 单位化

# 3. computer distance(error function):
count = 0
for point in data:
    d = abs(np.dot((point-p1), n))
```

善用numpy，减少循环的使用

参考小我吃辣不加价的作业

利用平面法向量与Z轴夹角

- 只使用inlier作为判断条件的不足；

  导致某个点数较多的非地面平面占据inlier个数；

  避免将平直墙面检测为地面，必须将夹角加入判断条件；

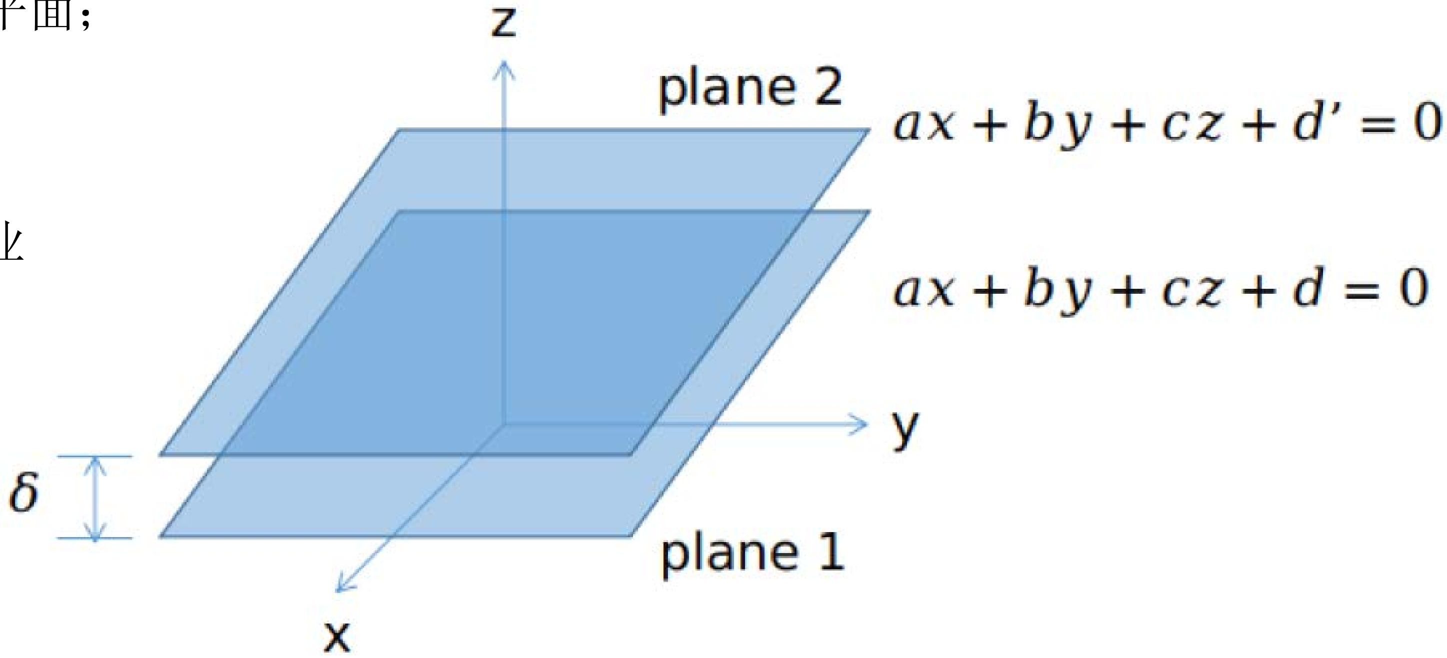
参考Blackest的作业

```
# 法向量与Z轴(0,0,1)夹角
alphaz = math.acos(abs(coeffs[2]) / r)

......

if near_point_num > max_point_num and alphaz < alpha_threshold:
    max_point_num = near_point_num

    ......

    alpha = alphaz
```

地面点选取

1、选择inlier点；
2、利用分割平面；

参考Alex-Su的作业



LSQ

LSQ可以精修RANSAC的分割结果

```python
def LSQ(data):
    # model: ax + by + cz + d = 0
    H = np.cov(data.T)
    eigenvalues, eigenvectors = np.linalg.eig(H)
    sorted_idx = np.argsort(eigenvalues)
    a, b, c = eigenvectors[sorted_idx[0]]
    print("means of xyz: ", data.mean(axis=0))
    xyz_means = data.mean(axis=0)
    d = -(a*xyz_means[0] + b*xyz_means[1] + c*xyz_means[2])
    params = [a, b, c, d]
    print("params = ", params)
    return params
```

```python
params = LSQ(data=data)
for idx, point in enumerate(data):
    dist = point2plane(point, params)
    if dist < tao:
        ground_index.append(idx)
    else:
        no_ground_index.append(idx)
        segmengted_cloud.append(point)
```

参考小我吃辣不加价的作业

第二题 地物聚类 DBSCAN

深蓝学院 shenlanxueyuan.com

● 算法流程

1、创建访问记录矩阵；

2、循环直到所有点标记为visited；

　　2.1 在未标记的点中随机选择初始点，并修改状态；

　　2.2 获取初始点r半径范围内的近邻；

　　2.3 若近邻数量小于min_samples，则标记为noise；大于等于min_samples，标记为核心点；

　　2.4 从初始点创建新的聚类；

　　3、 遍历其近邻；

地物聚类 DBSCAN

深蓝学院 shenlanxueyuan.com

- 算法流程

- 遍历其近邻；

```
# 3.遍历其近邻
core_stack=[]
neighbors_stack=[]
while(iscore[index]):
    tic=time.time()
    for j in neighbors:
        # 3.1 若该点未visited, 标记该点为visited, 并将该点加入cluster
        if(isvisited[j]):
            continue
        else:
            isvisited[j]=True
            clusters_index[j]=cluster_num
            # 3.2 判断该点是否为core point
            neighbors=kd.query_ball_point(data[j],r)
            k=len(neighbors)
            if(k>=min_samples):
                iscore[j]=True
                # 将 core point的索引 和 其近邻索引 别入栈
                core_stack.append(j)
                neighbors_stack.append(neighbors)
    # 若栈非空, 说明近邻中有core point, 将该点作为p点, 遍历其近邻
    toc=time.time()
    print("遍历完一个点的所有近邻花费时间：%.2fs"%(toc-tic))
    if len(core_stack):
        index=core_stack.pop()
        neighbors=neighbors_stack.pop()
    #若栈空, 则该聚类搜索结束
    else:
        cluster_num+=1
        break
```

参考SISE的作业

C++实现

- 地面分割 RANSAC 核心代码

```cpp
#include "ransac.h"

std::pair<Eigen::Vector3d,PointXYZ> ransac(pcl::PointCloud<PointXYZ>::Ptr db,int max_iter){
    srand(time(0));
    std::vector<int> index_final;
    PointXYZ plat_point;
    Eigen::Vector3d ABC;
    index_final.clear();
    while(max_iter--){
        std::vector<int> index;
        index.clear();
        for(int k =0;k<3;k++){
            index.push_back(rand()%(db->size()));
        }
        double x1, y1,z1, x2, y2,z2, x3, y3,z3;
        auto idx = index.begin();
        x1 = (*db)[*idx].x;
        y1 = (*db)[*idx].y;
        z1 = (*db)[*idx].z;
        idx++;
        x2 = (*db)[*idx].x;
        y2 = (*db)[*idx].y;
        z2 = (*db)[*idx].z;
        idx++;
        x3 = (*db)[*idx].x;
        y3 = (*db)[*idx].y;
        z3 = (*db)[*idx].z;
        Platform p;
        p.a = (y2 - y1)*(z3 - z1) - (z2-z1)*(y3 - y1);
        p.b = (z2 - z1)*(x3 - x1) - (x2-x1)*(z3 - z1);
        p.c = (x2 - x1)*(y3 - y1) - (y2-y1)*(x3 - x1);
        p.d = -(p.a*x2 + p.b*y2 + p.c*z2);
        for(int i=0;i < db->size();i++){
            double x4 = (*db)[i].x;
            double y4 = (*db)[i].y;
            double z4 = (*db)[i].z;
            double dis = fabs((x4-x2)*p.a+(y4-y2)*p.b+(z4-z2)*p.c)/sqrt(p.a*p.a+p.b*p.b+p.c*p.c);
            if(dis<0.12){
                index.push_back(i);
            }
        }
        if(index.size()>index_final.size()){
            index_final = index;
            plat_point = PointXYZ(x1,y1,z1);
            ABC = Eigen::Vector3d(p.a,p.b,p.c);
        }
    }
    return make_pair(ABC,plat_point);
}
```

参考ESOman的作业

C++实现

● 地面分割 RANSAC 核心代码

```cpp
x3 = (*db)[*idx].x;
y3 = (*db)[*idx].y;
z3 = (*db)[*idx].z;
Platform p;
p.a = (y2 - y1)*(z3 - z1) - (z2-z1)*(y3 - y1);
p.b = (z2 - z1)*(x3 - x1) - (x2-x1)*(z3 - z1);
p.c = (x2 - x1)*(y3 - y1) - (y2-y1)*(x3 - x1);
p.d = -(p.a*x2 + p.b*y2 + p.c*z2);
for(int i=0;i < db->size();i++){
    double x4 = (*db)[i].x;
    double y4 = (*db)[i].y;
    double z4 = (*db)[i].z;
    double dis = fabs((x4-x2)*p.a+(y4-y2)*p.b+(z4-z2)*p.c)/sqrt(p.a*p.a+p.b*p.b+p.c*p.c);
    if(dis<0.12){
        index.push_back(i);
    }
}
//更新集合
if(index.size()>index_final.size()){
    index_final = index;
    plat_point = PointXYZ(x1,y1,z1);
    ABC = Eigen::Vector3d(p.a,p.b,p.c);
```

```cpp
/*对降采样的点云进行进行ransac*/
int point_size = points->size()-1;
chrono::steady_clock::time_point t5 = chrono::steady_clock::now();
std::pair<Eigen::Vector3d,PointXYZ> plat_point;
cout << "滤波后的点云数量：" << cloud_filtered->size() << endl;
plat_point = ransac(cloud_filtered,300);
/*对所有点云进行计算离平面距离,并提取对应的index*/
std::vector<int> platform_index;//存放平面点的容器
for(int i = 0 ;i<points->size();i++){
    Eigen::Vector3d dis_vector;
    dis_vector[0] = (*points)[i].x - plat_point.second.x;
    dis_vector[1] = (*points)[i].y - plat_point.second.y;
    dis_vector[2] = (*points)[i].z - plat_point.second.z;
    double dis;
    dis = fabs(dis_vector.dot(plat_point.first))/ (double)sqrt(plat_point.first.squaredNorm());
    if(dis < 0.28){
        platform_index.push_back(i);
    }
}
```

计算每个点的距离时的区别

参考ESOman的作业

C++实现

● 地物聚类 DBSCAN 核心代码

```cpp
void dbscan::run(){
    for(int i=0;i<db_->size();i++){
        if((*cluster_state)[i] != -1) continue;
        if(isCore(i)){//如果圆内数量大于min_sample
            explore(i,++cluster_id);
        }else{
            (*cluster_state)[i] = 0;//噪点
        }
    }
}
void dbscan::explore(int index,int cluster_idx){
    (*cluster_state)[index] = cluster_idx;//核心点
    result a(eps_);
    octree_radius_search(root_,db_,(*db_)[index],a);
    if(a.only_index.size() <= min_sample_) return;
    for(auto &idx:a.only_index){
        if((*cluster_state)[idx] != -1) continue;//已访问过的点跳出
        explore(idx,cluster_idx);
    }

}
bool dbscan::isCore(int index){
    result a(eps_);
    octree_radius_search(root_,db_,(*db_)[index],a);
    if(a.only_index.size()>=min_sample_){
        return true;
    }else{
        return false;
    }
}
```

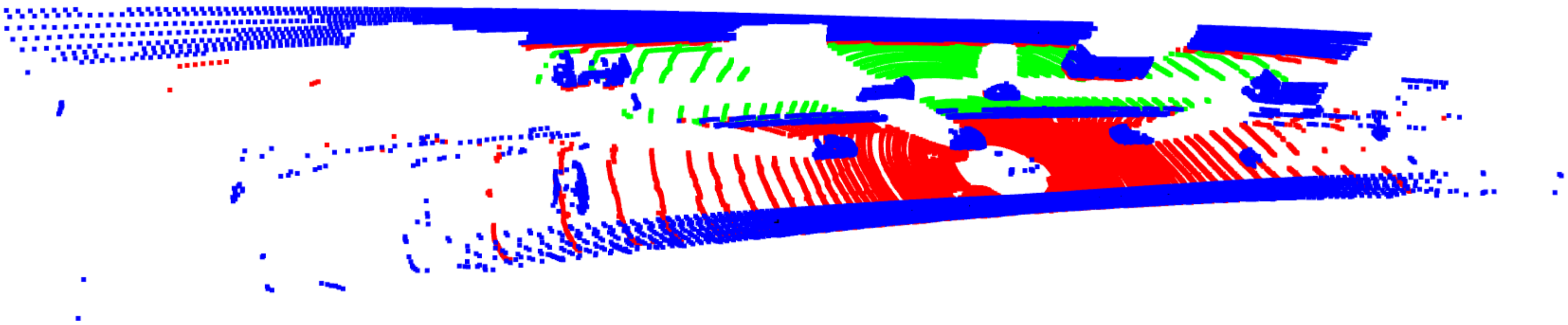
```python
for i,cand_idx in enumerate(candidate_idx):
    result_set_2 = RadiusNNResultSet(radius=r)
    kdtree.kdtree_radius_search(kd_root, data,result_set_2, data[cand_idx])
    for i in range(result_set_2.size()):
        if(result_set_2.dist_index_list[i].index not in unmarked_point_idx):
            continue
        result_set_3 = RadiusNNResultSet(radius=r)
        kdtree.kdtree_radius_search(kd_root, data,result_set_3, data[cand_idx])
        unmarked_point_idx.remove(result_set_3.dist_index_list[i].index)
        clusters_index[result_set_3.dist_index_list[i].index]=label
        if(result_set_3.size()>min_number):
            candidate_idx.append(result_set_3.dist_index_list[i].index)
```

通过list的append
动态的增加元素。

参考ESOman的作业

地面分割效果举例

深蓝学院
shenlanxueyuan.com



参考SISE的作业

聚类效果举例

参考SISE的作业

在线问答

Q&A

深蓝学院
shenlanxueyuan.com

感谢各位聆听!

Thanks for Listening