深蓝学院
shenlanxueyuan.com

三维点云处理
第一期第一章作业讲评

主讲人 王对武

纲要

- 作业
- 评阅细则
- 作业1 PCA的应用
- 作业2 法向量估计
- 作业3 栅格下采样

评阅细则

深蓝学院 shenlanxueyuan.com

- 总体原则
  - 根据作业1、2、3的完成度进行评阅，作业中有自己的思考，或者用C++实现，代码优秀的，酌情加分。
- 细则
  - 及格：PCA/法向量估计做对一个，并且下采样的两种方法（random或者centroid）至少对一个；
  - 良好：PCA/法向量估计做对一个，并且下采样的两种方法都对；
  - 优秀：PCA/法向量估计/下采样的两种方法（random或者centroid）都对；

作业1 PCA的应用

深蓝学院 shenlanxueyuan.com

原理部分

给定点云$xi \in \mathbb{R}n, i = 1,2, \cdots m$, PCA的步骤为：

1、归一化

$$\tilde{X} = [\tilde{x}_1, \cdots, \tilde{x}_m], \tilde{x}_i = x_i - \bar{x}, i = 1, \cdots, m \qquad \bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i.$$

2、构造协方差矩阵，进行SVD分解，求解特征值与特征向量：

$$H = \tilde{X}\tilde{X}^T = U_r \Sigma^2 U_r^T$$

3、选取Ur最大特征值对应的特征向量作为主成分，并进行可视化。

作业1 PCA的应用

深蓝学院
shenlanxueyuan.com

需要编写的核心代码

（1）标准化

```
# 计算均值
data_mean = np.mean(data, axis=0)
# 归一化
normalize_data = data - data_mean
```

（2）SVD分解

```
# 构造协方差矩阵
H = np.dot(normalize_data.T, normalize_data)
# SVD分解
eigenvectors, eigenvalues, eigenvectors_t = np.linalg.svd(H)
```

作业1 PCA的应用

深蓝学院 shenlanxueyuan.com

需要编写的核心代码

（3）选取主成分并显示

```
# TODO: 此处只显示了点云，还没有显示PCA
v, u = PCA(points)
print('the main orientation of this pointcloud is: ', u[:, 0])
# 画出PCA主方向
point = [[0, 0, 0], u[:, 0], u[:, 1]]  # 提取第一和第二主成分
lines = [[0, 1], [0, 2]]  # 由点构成线，[0,1]代表以点集中序号为0和1的点组成线
colors = [[1, 0, 0], [0, 1, 0]]  # 为不同的线添加不同颜色
#构造open3D中的LineSet对象，用于主成分的显示
line_set = o3d.geometry.LineSet(
    points=o3d.utility.Vector3dVector(point),
    lines=o3d.utility.Vector2iVector(lines)
)
line_set.colors = o3d.utility.Vector3dVector(colors)
o3d.visualization.draw_geometries([point_cloud_o3d, line_set])
```

作业1 PCA的应用

深蓝学院 shenlanxueyuan.com

易错点

从步骤上来看，由于我们可以直接使用SVD，或者特征值分解的API，所以整个主成分分析的实现并不难。但是，具体细节有不少错误。例如很多同学直接照搬作业中的代码框架，没有主动分析，错误地选择了最小特征值对应的特征向量作为主成分。还有特征矩阵的行或者列选择错误。

```
if sort:
    sort = eigenvalues.argsort()[::-1]
    eigenvalues = eigenvalues[sort]
    eigenvectors = eigenvectors[:, sort]

return eigenvalues, eigenvectors
        ⋮
# TODO: 此处只显示了点云，还没有显示PCA
v, u = PCA(points)
print('the main orientation of this pointcloud is: ', u[:, 0])
# 画出PCA主方向
point = [[0, 0, 0], u[:, 0], u[:, 1]]  # 提取第一和第二主成分
lines = [[0, 1], [0, 2]]  # 由点构成线，[0,1]代表以点集中序号为0和1的点组成线
colors = [[1, 0, 0], [0, 1, 0]]  # 为不同的线添加不同颜色
#构造open3D中的LineSet对象，用于主成分的显示
line_set = o3d.geometry.LineSet(
    points=o3d.utility.Vector3dVector(point),
    lines=o3d.utility.Vector2iVector(lines)
)
line_set.colors = o3d.utility.Vector3dVector(colors)
o3d.visualization.draw_geometries([point_cloud_o3d, line_set])
```

u[:, 2]

主成分选择错误示例

```
if sort:
    sort = eigenvalues.argsort()[::-1]
    eigenvalues = eigenvalues[sort]
    eigenvectors = eigenvectors[:, sort]

eigenvectors = list(eigenvectors)
# 分别得到三个特征向量并返回
point_cloud_vector  = eigenvectors[0]  # 点云主方向对应的向量
point_cloud_vector1 = eigenvectors[1]  # 点云次方向对应的向量
point_cloud_vector2 = eigenvectors[2]  # 点云第二次方向对应的向量
return point_cloud_vector, point_cloud_vector1, point_cloud_vector2
```

错误代码（行列选错）

```
0 = {ndarray} [-9.959788e-01  5.536268e-04 -8.958811e-02] ...View as Array
1 = {ndarray} [-0.08958723  0.00144826  0.99597794] ...View as Array
2 = {ndarray} [ 6.8114733e-04  9.9999881e-01 -1.3928424e-03] ...View as Array
```

```
#eigenvectors = list(eigenvectors)
# 分别得到三个特征向量并返回
point_cloud_vector  = eigenvectors[:, 0]  # 点云主方向对应的向量
point_cloud_vector1 = eigenvectors[:, 1]  # 点云次方向对应的向量
point_cloud_vector2 = eigenvectors[:, 2]  # 点云第二次方向对应的向量
return point_cloud_vector, point_cloud_vector1, point_cloud_vector2
```

正确代码

主成分向量的行或者列选择错误示例

作业1 PCA的应用

深蓝学院 shenlanxueyuan.com

易忽略的点

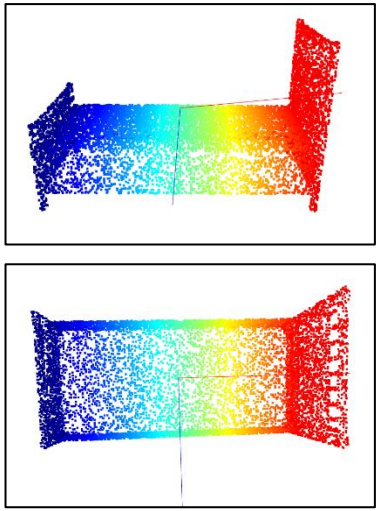
没有对主成分进行可视化，也没验证PCA的结果，导致在粗心或者没有完全弄清出PCA原理的情况下，容易选错主成分。

所选数据：bed_0003.txt

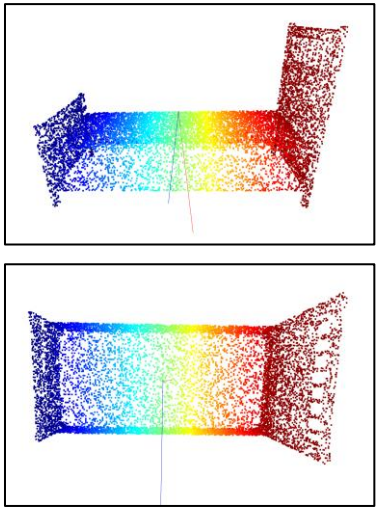
PCA结果显示与错误对比

侧面

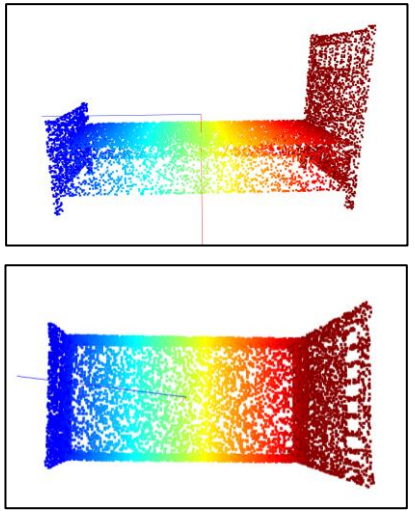
俯视



正确结果



错误示例（成分选错）



错误示例（行列选错）

作业2 法向量估计

深蓝学院 shenlanxueyuan.com

需要编写的核心代码

遍历所有点云，寻找与之邻近的k个点，然后对这些点进行PCA处理，得到最小特征值对应的特征向量即为法向量

```
# 循环计算每个点的法向量
pcd_tree = o3d.geometry.KDTreeFlann(point_cloud_o3d)
normals = []
# 利用kd_tree搜索最近10个点，并且利用PCA函数计算最小特征向量即为点的法向量
for i in range(points.shape[0]):
    [_, idx, _] = pcd_tree.search_knn_vector_3d(point_cloud_o3d.points[i], 10)
    k_nearest_point = np.asarray(point_cloud_o3d.points)[idx, :]
    v, u = PCA(k_nearest_point)
    normals.append(u[:, 2])
normals = np.array(normals, dtype=np.float64)
# TODO: 此处把法向量存放在了normals中
point_cloud_o3d.normals = o3d.utility.Vector3dVector(normals)
o3d.visualization.draw_geometries([point_cloud_o3d, line_set])
```

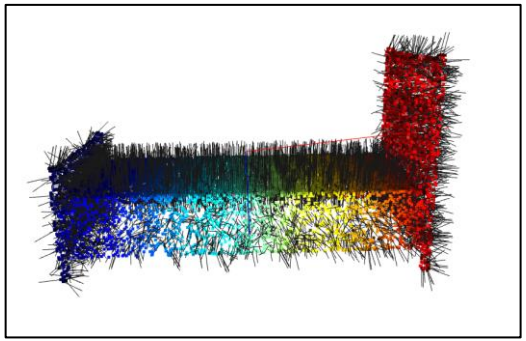
需注意的地方

作业2 法向量估计

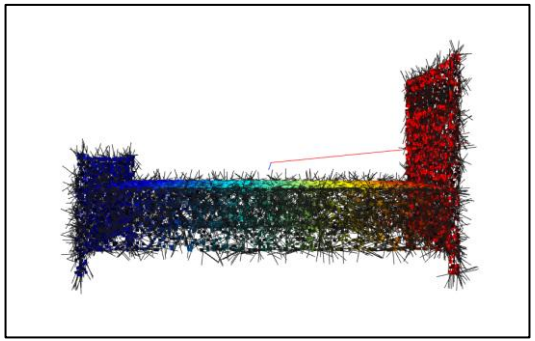
深蓝学院 shenlanxueyuan.com

法向量估计结果显示与错误对比

所选数据：bed_0003.txt

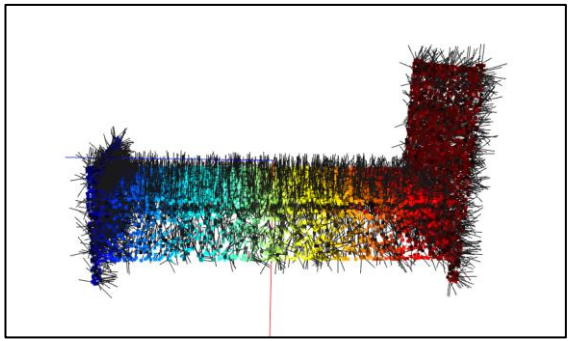


正确结果



错误示例（成分选错）

法向量全都贴着平面，而不是垂直平面



错误示例（行列选错）

该错误比较隐蔽，不细看难以发现平面的法向量中很多歪的，所以对比正确结果看起来更稀疏

作业3 栅格下采样

深蓝学院 shenlanxueyuan.com

原理部分

给定点集$\{p_1, p_2, \cdots, p_i, \cdots, p_N\}, p_i = (x_i, y_i, z_i)$

1、求最大栅格的包围框，即x， y， z三轴上的各自的最大、最小值。

$$x_{max} = \max(x_1, x_2, \cdots, x_N), x_{min} = \min(x_1, x_2, \cdots, x_N), y_{max} = \cdots\cdots$$

2、选取栅格大小r

3、计算各个轴能够划分的栅格维度大小

$$D_x = (x_{max} - x_{min})/r$$
$$D_y = (y_{max} - y_{min})/r$$
$$D_z = (z_{max} - z_{min})/r$$

作业3 栅格下采样

深蓝学院 shenlanxueyuan.com

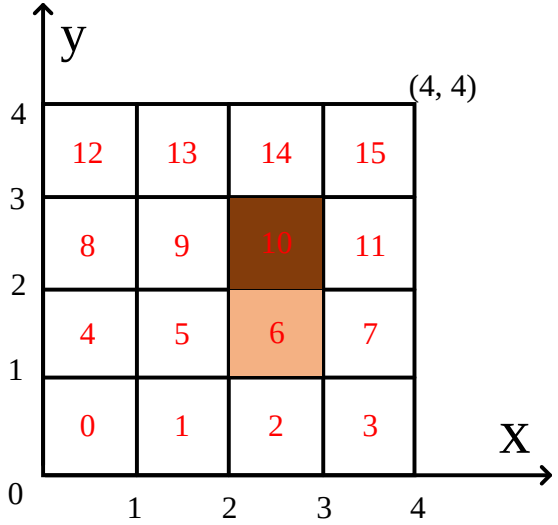
原理部分

4、计算某个点被划分进的栅格序号

$$h_x = \lfloor (x - x_{min})/r \rfloor$$
$$h_y = \lfloor (y - y_{min})/r \rfloor$$
$$h_z = \lfloor (z - z_{min})/r \rfloor$$
$$h = h_x + h_y * D_x + h_z * D_x * D_y$$

example in 2D

Input
P=(2.5, 2.5) r = 1
comput
Dx = D_y = 4
x_min = y_min =0
x_max = y_max =4
h_x = 2
h_y = 2
output
h = 2+2*4 = 10



作业3 栅格下采样

深蓝学院 shenlanxueyuan.com

原理部分

5、根据步骤4得到的序号进行排序，取具有相同序号的点集进行采样，采样方式为random和centroid两种。

6、基于Hash表的栅格采样方法

注意点

r的大小选取，过小容易造成溢出

易忽略的点

审题问题，很多同学只实现了一种采样方式

作业3 栅格下采样

深蓝学院 shenlanxueyuan.com

需要编写的核心代码(排序算法)

1、求最大栅格的包围框

整理自KaiXin的优秀代码

```
filtered_points = []
data = point_cloud.values
# 求出x,y,z三轴各自的最小值、最大值
min_d = data.min(axis=0)
max_d = data.max(axis=0)
```

2、计算各个轴能够划分的栅格维度大小

```
# 求出x,y,z三轴各自的维度
D = (max_d - min_d) / leaf_size
```

作业3 栅格下采样

深蓝学院 shenlanxueyuan.com

需要编写的核心代码(排序算法)

3、计算点的栅格序号

整理自KaiXin的优秀代码

```
# 求出所有点所在栅格序号h
point_x, point_y, point_z = np.array(point_cloud.x), np.array(point_cloud.y), np.array(point_cloud.z)
h_x, h_y, h_z = np.floor((point_x - min_d[0]) / leaf_size), \
                np.floor((point_y - min_d[1]) / leaf_size), \
                np.floor((point_z - min_d[2]) / leaf_size)
h = np.array(np.floor(h_x + h_y * D[0] + h_z * D[0] * D[1]))
```

4、排序

```
# 根据h对点云进行排序
data = np.c_[h, point_x, point_y, point_z]
data = data[data[:, 0].argsort()]
```

作业3 栅格下采样

深蓝学院 shenlanxueyuan.com

需要编写的核心代码(排序算法)

5、采样

整理自KaiXin的优秀代码

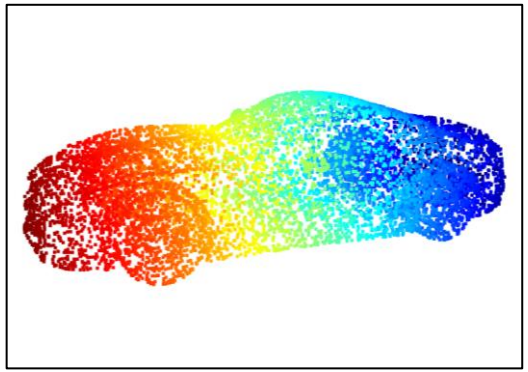
```
# 随机采样
if mode == 'random':
    filtered_points = []
    for i in range(data.shape[0] - 1):
        if data[i][0] != data[i + 1][0]:  # 判断h是否相等
            # 选取序号为h的栅格内第1点作为随机采样点
            filtered_points.append(data[i][1:])
    filtered_points.append(data[data.shape[0]-1][1:])
# 均值采样
if mode == 'centroid':
    filtered_points = []
    data_points = []
    for i in range(data.shape[0] - 1):
        if data[i][0] == data[i + 1][0]:  # 判断h是否相等
            data_points.append(data[i][1:])
            continue
        if data_points == []:
            continue
        # 取序号为h的栅格内所有点的中心
        filtered_points.append(np.mean(data_points, axis=0))
        data_points = []
    # print(filtered_points)
    filtered_points = np.array(filtered_points)
```

作业3 栅格下采样

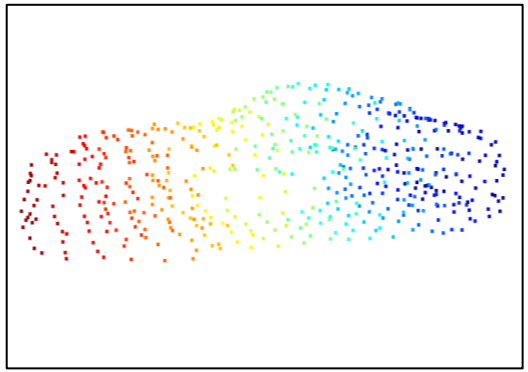
深蓝学院 shenlanxueyuan.com

采样结果显示

所选数据：car_0005.txt

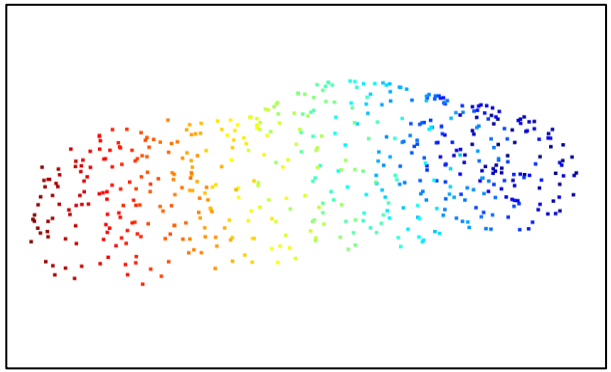


原始点云



Centroid采样

采样后的点相对而言排列整洁有序，点与点之间的间隔大致相同。



Random采样

该采样后的点排列杂乱无序，点与点之间的间隔不等。

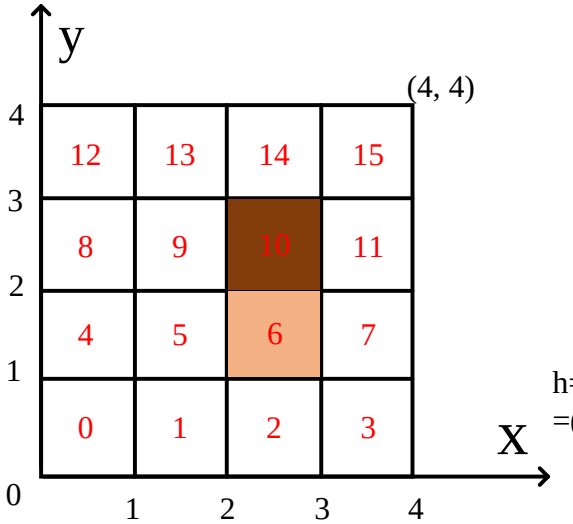
作业3 栅格下采样

深蓝学院 shenlanxueyuan.com

基于Hash表的算法

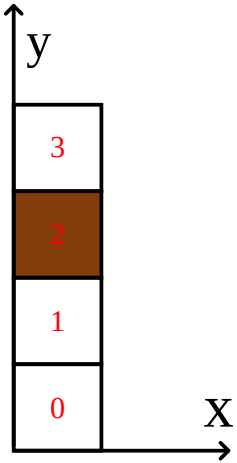
5、不进行排序，而是直接将其存入对应的栅格表中

Input
P=(2.5, 2.5) *r* = 1
comput
Dx = D_y = 4
x_min = y_min =0
x_max = y_max =4
h_x = 2
h_y = 2
output
h = 2+2*4 = 10



降维或映射

h=hash(hx,hy,xz)
=(hx+hy*Dx+hz*Dx*Dy)%4



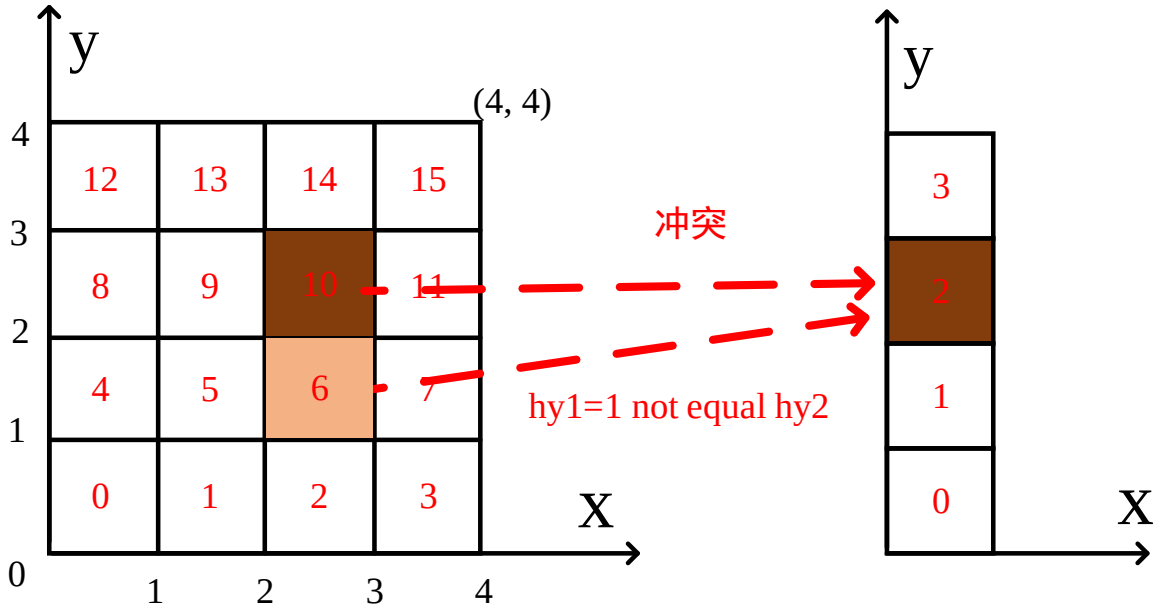
满足一个容器包含其内部的点，都可算做基于Hash表的采样，
但经过降维后的需要注意冲突判断

作业3 栅格下采样

深蓝学院 shenlanxueyuan.com

基于Hash表的算法

5、冲突检测

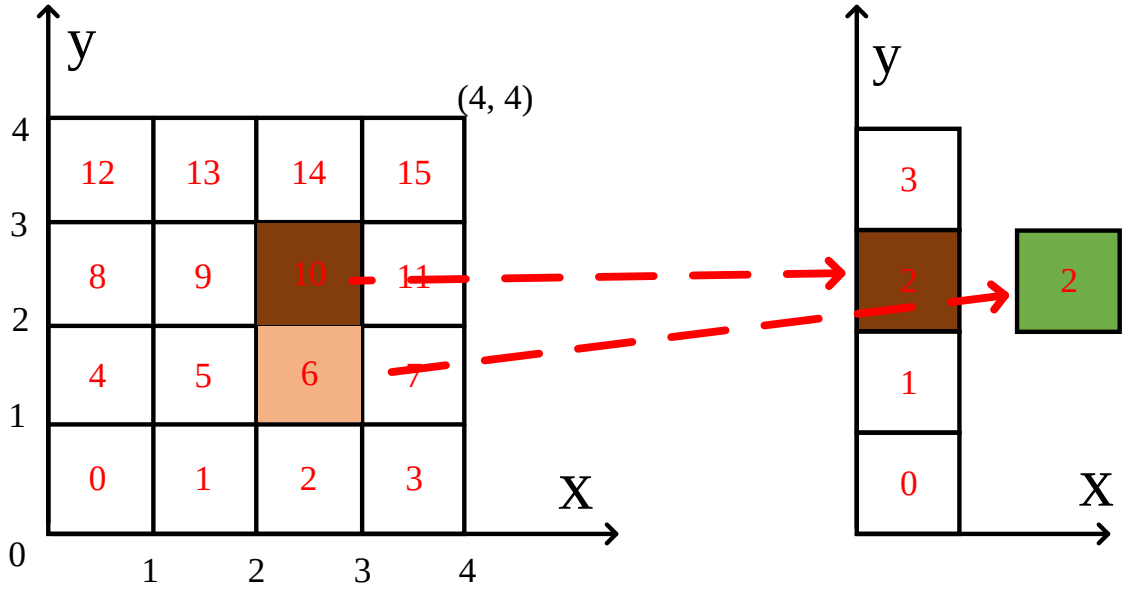


作业3 栅格下采样

深蓝学院 shenlanxueyuan.com

基于Hash表的算法

5、冲突检测



将冲突容器内的点进行centroid或random采样输出，并清空，用于存放新的点云

作业3 栅格下采样

深蓝学院 shenlanxueyuan.com

需要编写的核心代码(基于Hash表的算法)

5、采样

整理自Mr.robot的优秀代码

```
# 构建二维列表用于存储在不同栅格内的点云
voxel = [[] for _ in range(int(D[0]*D[1]*D[2]))]
for i, point in enumerate(data):
    h_xyz = (point - min_d) // leaf_size  # 求第i点的hx,hy,hz
    h = round(h_xyz[0] + h_xyz[1] * D[0] + h_xyz[2] * D[0] * D[1])
    voxel[int(h)].append(i)  # 在序号为point_h的栅格中存储当前点云
# 均值采样
if function == 'centroid':
    for v in voxel:
        if len(v) == 0:
            continue
        filtered_points.append(np.sum(data[v], axis=0)/len(v))
# 随机采样
if function == 'random':
    for v in voxel:
        if len(v) == 0:
            continue
        filtered_points.append(data[random.choice(v)])
filtered_points = np.array(filtered_points, dtype=np.float64)
```

容器大小在这里就是栅格数量
栅格数量=Dx*Dy*Dz
没有取余，故不需要进行冲突判断

作业3 栅格下采样

深蓝学院
shenlanxueyuan.com

易错点

1、由于粗心，导致h计算错误

2、采用Hash表并取余的同学没有做冲突判断

易忽略的点

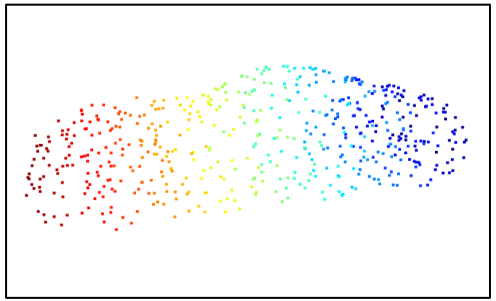
审题问题，很多同学只实现了一种采样方式

作业3 栅格下采样

深蓝学院 shenlanxueyuan.com

H计算错误

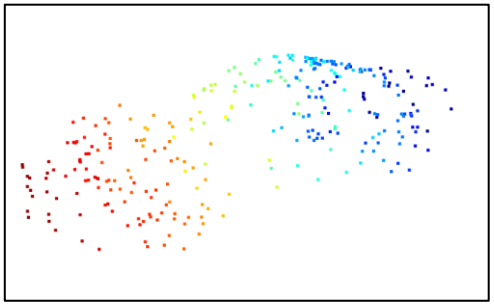
```
#正确计算
h = np.rint(hx + hy * Dx + hz * Dx * Dy)
```



正确结果

hz输入成hx，导致栅格划分错误，采样不均匀

```
#错误示例
h = np.rint(hx + hy * Dx + hx * Dx * Dy)
```



错误结果

作业3 栅格下采样

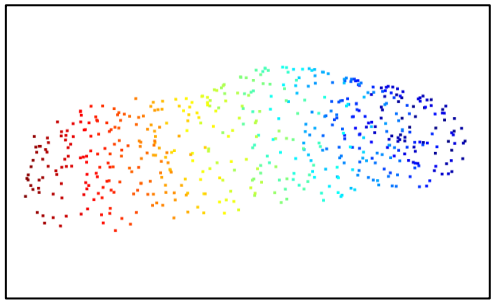
深蓝学院 shenlanxueyuan.com

基于Hash表的采样错误代码示例

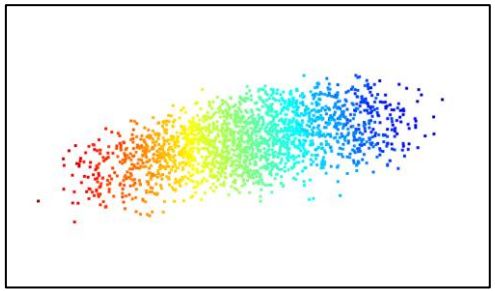
```
h = hxyz[0] +hxyz[1]*Dx+hxyz[2]*Dx*Dy
h_list.append(h)
h_list1[0].append(h%leaf_size)
while s <= len(hash_list[0]) - 1:
    # for point in points:
    #对不同hash值的点进行centroid采样
    if hash_list[0][s] - hash_list[0][s - 1] == 0:
        sum_num += 1
        sum_x += points[s][0]
        sum_y += points[s][1]
        sum_z += points[s][2]
    else:
        sum_x = sum_x / sum_num
        sum_y = sum_y / sum_num
        sum_z = sum_z / sum_num

        filtered_point = ([sum_x], [sum_y], [sum_z])
        filtered_points.append(filtered_point)
        sum_x = sum_y = sum_z = 0
        sum_num = 1
    s += 1
```

没有冲突检测



正确结果



错误结果

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