Homework1

Author	WesleyHuang
Task	Done🙌

PCA

核心代码：

```
def PCA(data):
    if type(data) is o3d.cpu.pybind.utility.Vector3dVector:
        pcd = np.asarray(data)
    else:
        pcd = data

    if pcd.ndim is not 3:
        pcd = np.transpose(pcd)

    # Normalization
    mean = pcd.mean(axis=1, keepdims=True)
    pcd = pcd - mean

    # Implement SVD
    H = np.dot(pcd, pcd.T)
    u, s, vt = np.linalg.svd(H, hermitian=True)

    ##or directly use SVD to decompose pcd matrix, u is the same as above
    #u, s, vt = np.linalg.svd(pcd, hermitian=False)

    eigenvalues = s
    eigenvectors = u

    return eigenvalues, eigenvectors
```

备注：

其中，主方向的计算可直接对X_{3*n}作SVD分解，其左奇异向量即为特征向量，或作XX^T特征分解，其左矩阵即为特征向量。

结果：

对ModelNet40中飞机模型（0001，0005）作PCA分析，可视化结果如下所示。红线/绿线/蓝线分别表示第一/第二/第三主要方向。

Normal Estimation

核心代码：

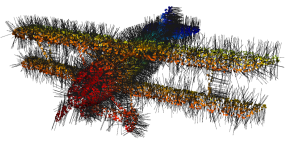
```
# Build KD-Tree
pcd_tree = o3d.geometry.KDTreeFlann(pcd)
normals = []
neighbor_size = 10
for i in range(pcd_array.shape[0]):
    # Get the neighbor points' index within neighbor_size
    _, idx, _ = pcd_tree.search_knn_vector_3d(pcd.points[i], neighbor_size)
    # Perform PCA
    _, v = PCA(np.asarray(pcd.points)[idx, :])
    normals.append(v[:,2])
```

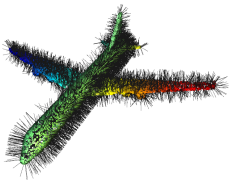
备注：

- 主要步骤：
  - 1-对点云作KD树划分
  - 2-遍历点云中的每个点，对每个点作如下操作
    - i-利用KD树获得该点邻域点的index
    - ii-提取邻域点并作PCA

- iii-最次要方向认为是该点的法向，即$\min n^T X X^T n$。将该方向赋值到该点的normal属性

结果：





Downsample

核心代码：

```
pcd = np.asarray(point_cloud.points)
pcd_max = pcd.max(axis=0)
pcd_min = pcd.min(axis=0)
# Compute dimention of the voxel grid
D = np.ceil((pcd_max - pcd_min) / leaf_size)
# Compute voxel index for each point
d = np.round((pcd - pcd_min) / leaf_size)

# Perform: h = hx + hy * Dx + hz * Dx * Dy
h_helper = np.array([[1], [D[0]], [D[0] * D[1]]])
hs = np.dot(d, h_helper)

# Store points within the same voxel into dictionary with same key
dict = {}
for idx, h in enumerate(hs):
    x = h[0]
    if x in dict.keys():
        dict[x] = np.vstack((dict[x], pcd[idx]))
    else:
        dict[x] = np.array([pcd[idx]])

filtered_points = []
# Iterate the dictionary. For each voxel, select the centroid / one point randomly.
for v in dict.values():
```

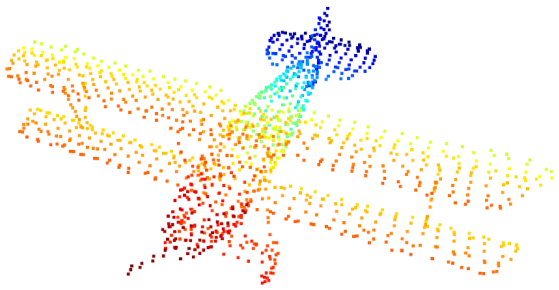
```
        if v.shape[0] is 1:
            filtered_points.append(v[0])
        else:
            if random_select:
                random_idx = np.random.choice(v.shape[0], 1)
                filtered_points.append(v[random_idx][0])
            else:
                filtered_points.append(v.mean(axis=0))
```

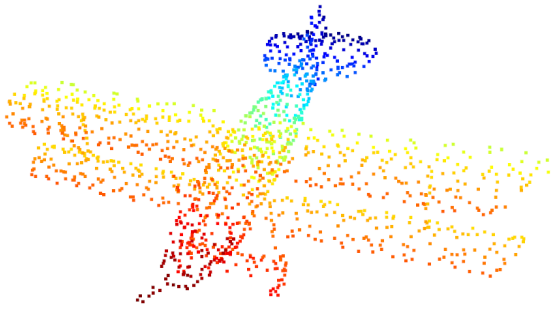
备注：

计算得到h后，利用h值作为key，将相同h的点储存到一个容器中，利用centroid或random的方法对点进行选取。

结果：

- leafsize = 0.05 (上：Centroid，下：Random)





- leafsize = 0.05 (上：Centroid，下：Random)