

# 三维视觉与理解课程

## 实验一

### 环境配置

```
In [74]: !pip install open3d
```

^C

```
In [46]: import open3d as o3d
import copy #点云拷贝库
import numpy as np
```

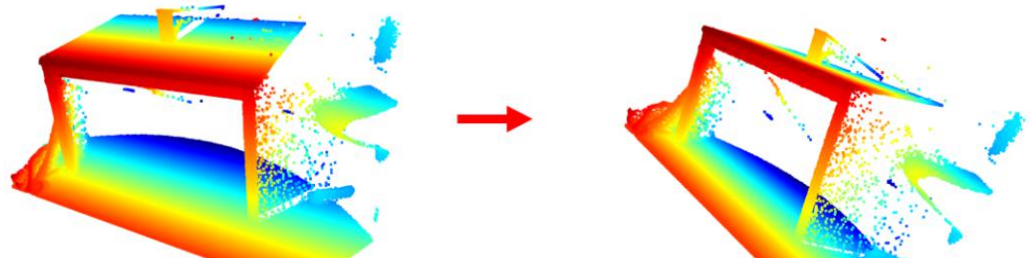
### 点云读写与可视化

```
In [71]: #读点云: 从文件路径处读取pcd格式的的点云文件 (table.pcd)
point_cloud = o3d.io.read_point_cloud("table.pcd")

#查看点云中, 点的个数
print("原始点云: ", point_cloud)
```

原始点云: PointCloud with 460400 points.

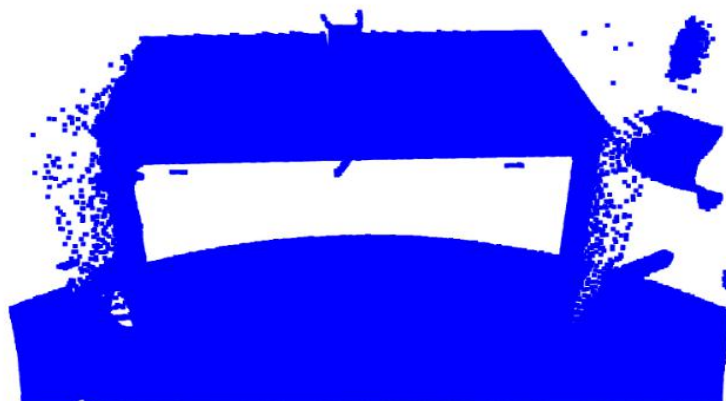
```
In [72]: # 将点云复制, 并将复制后的点云在x方向上平移若干个单位
o3d.visualization.draw_geometries([point_cloud], window_name="task1")
point_cloud_copy = copy.deepcopy(point_cloud).translate((50, 0, 0, 0, 0))
o3d.visualization.draw_geometries([point_cloud_copy], window_name="task2")
```

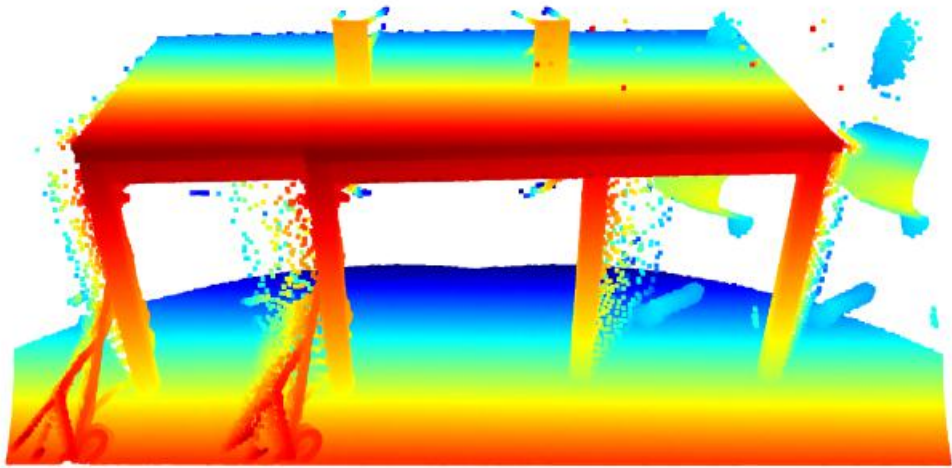


```
In [6]: # 将复制后的点云统一染成任意颜色(与之前颜色不同)
point_cloud_copy.paint_uniform_color([0, 0, 1, 0])
o3d.visualization.draw_geometries([point_cloud_copy], window_name="task1")

# 将染色后点云与原始点云合并为一个点云(相加), 并显示.
multi_point_cloud = point_cloud_copy + point_cloud
print("合并后的点云: ", multi_point_cloud)
o3d.visualization.draw_geometries([multi_point_cloud], window_name="task1")
```

合并后的点云: PointCloud with 920800 points.





```
In [34]: # 写点云：将合并后的点云保存在桌面位置，命名为t1.pcd
o3d.io.write_point_cloud('t1.pcd', multi_point_cloud)
```

Out[34]: True

## 点云的体素化

```
In [61]: #以斯坦福兔子点云 (bunny.pcd) 为基础，生成点云体素化之后的数据
bunny = o3d.io.read_point_cloud("bunny.pcd")
print("斯坦福兔子点云个数:", bunny)

bunny_3D = o3d.geometry.VoxelGrid.create_from_point_cloud(bunny, voxel_size=0.1)

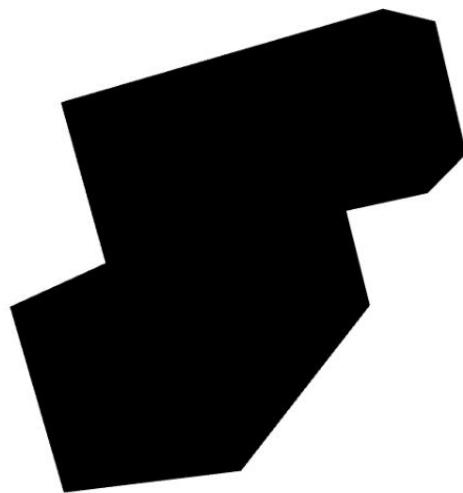
print(bunny_3D)

o3d.visualization.draw_geometries([bunny_3D], window_name="task2")
```

斯坦福兔子点云个数: PointCloud with 397 points.  
VoxelGrid with 7 voxels.

---

斯坦福兔子点云个数: PointCloud with 397 points.  
VoxelGrid with 7 voxels.



In [62]: # 调整voxel\_size, 分别令其取值为0.01、0.04、0.07与0.1, 并将体素化的结果进行展示。

```
voxel_size = [0.01, 0.04, 0.07, 0.1]
```

```
for v_size in voxel_size:
    b = o3d.geometry.VoxelGrid.create_from_point_cloud(bunny, voxel_size=v_size)
    print(f"当前VoxelSize为:{v_size}, {b}")
    o3d.visualization.draw_geometries([b], window_name="task")
```

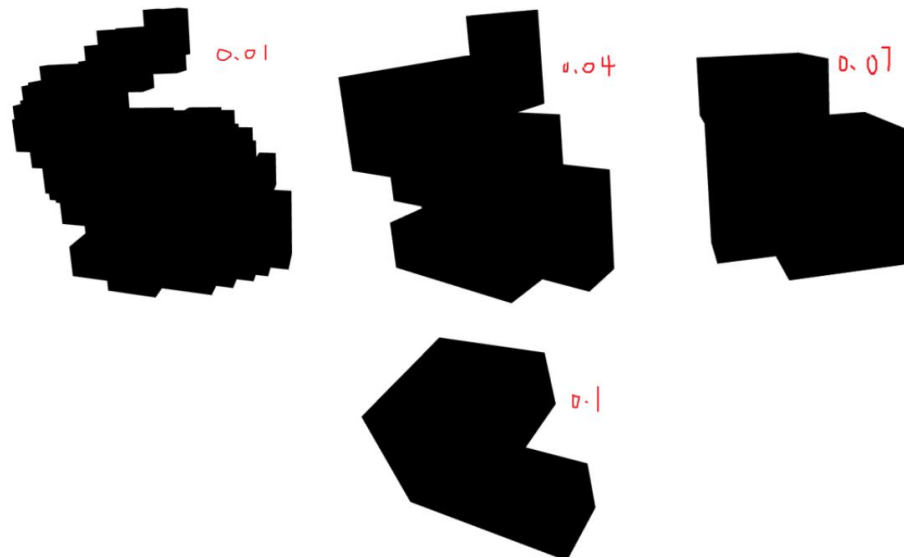
当前VoxelSize为:0.01, VoxelGrid with 333 voxels.

当前VoxelSize为:0.04, VoxelGrid with 34 voxels.

当前VoxelSize为:0.07, VoxelGrid with 15 voxels.

当前VoxelSize为:0.1, VoxelGrid with 7 voxels.

[Open3D WARNING] [ViewControl] SetViewPoint() failed because window height and width are not set.



请回答：结合可视化实验结果，用自己的语言描述，点云体素化是什么意思？

体素是什么？体素即体积像素。二维图片最小单元是像素，对应到三位空间最小单元即体素。点云体素化，即把每个拥有三个坐标值的点云映射成单个体素。点云数据三维格网化通常用于基于体素的深度学习数据预处理、局部相对特征计算以及投影。

请回答：Voxel\_size参数在点云体素化中起到了什么作用？

Voxel\_size是体素网格的分辨率，分辨率越高(voxel\_size数低)体素化后的图形越接近真实形状，分辨率越低(比如上图0.1)体素化后的形状越接近方块，越抽象。

## 代码解读练习

color.py代码中，实现了以点云（maize.pcd）坐标为依据，对点云进行渐变式赋色。请结合代码内容、可视化结果、控制台输出等

```
In [47]: #读文件
cloud = o3d.io.read_point_cloud("maize.pcd")

#调颜色: 绿色
cloud.paint_uniform_color([0, 1, 0])

#可视化
o3d.visualization.draw_geometries([cloud])

# 点云转换为数组
pts = np.asarray(cloud.points)
print("点云矩阵")
print(pts)
```

点云矩阵

[	7.0967999	-4.8207002	0.4303	]
[	7.0918002	-4.8204002	0.43009999	]
[	7.0840001	-4.8213	0.4298	]
...				
[	6.2782998	-3.3692999	-1.6933	]
[	6.2722001	-3.3685999	-1.692	]
[	6.2512002	-3.3620999	-1.6869	]]



```
In [58]: # 改变轴
Axis = [0,1,2] ##### 按照Axis轴填颜色####

# 平分程度
Avg = [0,1,2,3] #####调整颜色的多样性####

def fun(axis,avg):
    # 根据高度生成色彩
    colors = np.zeros([pts.shape[0], 3]) # 生成一个空的与点云相同大小的ndarray

    height_max = np.max(pts[:, axis]) # s轴最大

    height_min = np.min(pts[:, axis]) # s轴最小

    delta_c = abs(height_max - height_min) / (255 * avg)

    for j in range(pts.shape[0]):
        color_n = (pts[j, axis] - height_min) / delta_c
        if color_n <= 255:
            colors[j, :] = [0, 1 - color_n / 255, 1]
        else:
            colors[j, :] = [(color_n - 255) / 255, 0, 1]
    print(f"当前axis:{axis}, avg:{avg}")
    print("颜色矩阵")
    print(colors)
    print("点云矩阵+颜色矩阵")
    print(np.concatenate((pts, colors), axis=1))
    cloud.colors = o3d.utility.Vector3dVector(colors) # 给点云逐点赋色
    o3d.visualization.draw_geometries([cloud]) # 可视化

for axis in Axis:
    for avg in Avg:
        fun(axis, avg)
```

当前axis:0, avg:0

颜色矩阵

```
[[0. 1. 1.]
 [0. 1. 1.]
 [0. 1. 1.]
```

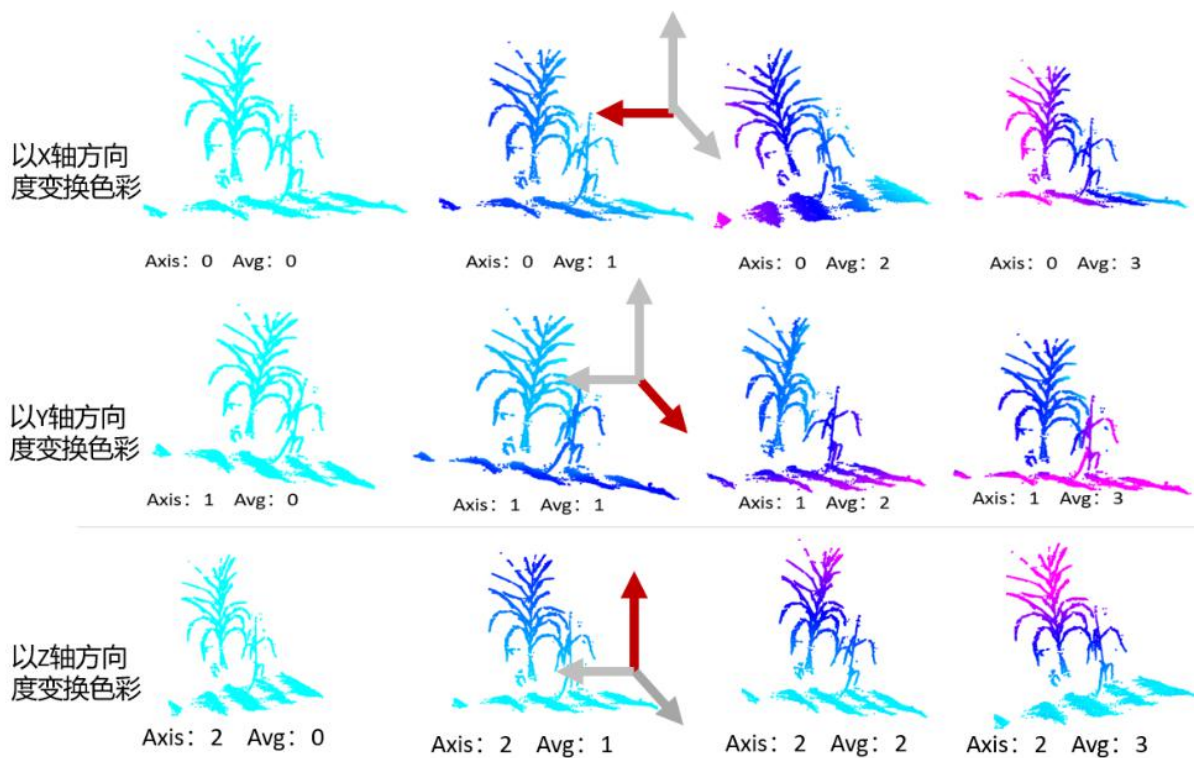
...

```
[0. 1. 1.]
 [0. 1. 1.]
 [0. 1. 1.]
```

点云矩阵+颜色矩阵

```
[[ 7.0967999 -4.8207002  0.4303      0.          1.          1.          ]
 [ 7.0918002 -4.8204002  0.43009999 0.          1.          1.          ]
 [ 7.0840001 -4.8213     0.4298      0.          1.          1.          ]
 ...
 [ 6.2782998 -3.3692999 -1.6933     0.          1.          1.          ]
 [ 6.2722001 -3.3685999 -1.692      0.          1.          1.          ]
 [ 6.2512002 -3.3680002 -1.6928     0.          1.          1.          ]
```





(1)用自己的语言说明，从矩阵的角度看，点云中的每个点怎样被赋予不同颜色的？

染色点云矩阵 = 点云矩阵 X 颜色矩阵

(2)通过查阅代码，结合改变变量的实验结果，尝试阐明代码中的axis与avg变量的作用。

Axis: 按照Axis轴填颜色

Avg: 调整颜色的多样性

## 进阶任务

参照color.py代码，编写程序针对maize.pcd点云实现如下操作：x坐标≤7的点染为红色、x坐标>7但≤7.4的点染为绿色、x坐标>7.4的点染为蓝色，并将染色后的结果进行实时显示。

```
9]: maize = o3d.io.read_point_cloud("maize.pcd")
```

```
print("点云个数: ",maize)
```

点云个数: PointCloud with 39432 points.

```
0]: o3d.visualization.draw_geometries([maize]) # 可视化
```



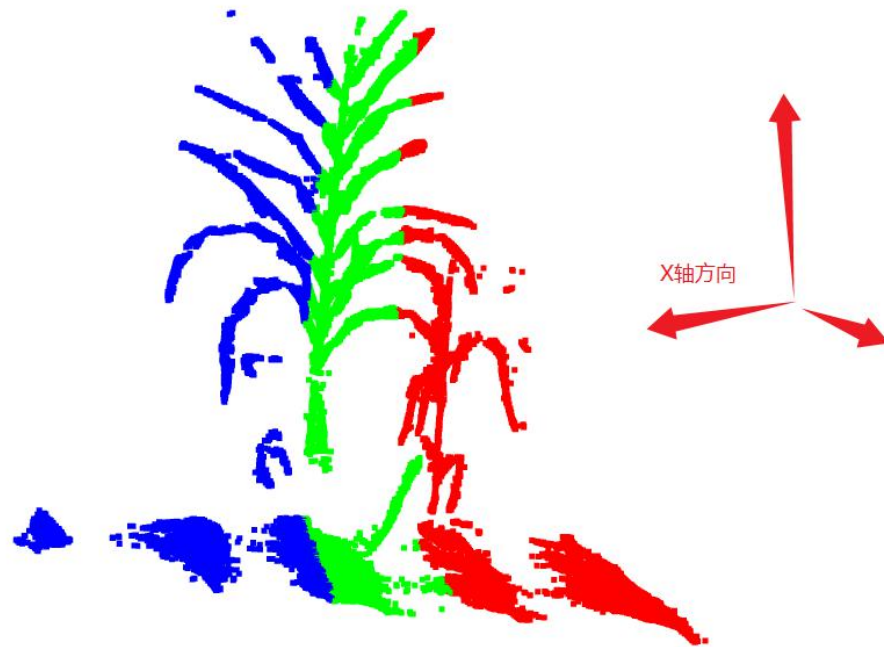
```
In [68]: # x坐标 $\leq 7$ 的点染为红色、x坐标 $>7$ 但 $\leq 7.4$ 的点染为绿色、x坐标 $>7.4$ 的点染为蓝色
```

```
#初始化颜色矩阵
colors = np.zeros([len(cloud.points), 3])

#修改逻辑
for i, point in enumerate(cloud.points):
    x_coord = point[0]
    if x_coord <= 7:
        colors[i] = [1, 0, 0] #red
    elif x_coord <= 7.4:
        colors[i] = [0, 1, 0] #green
    else:
        colors[i] = [0, 0, 1] #blue

cloud.colors = o3d.utility.Vector3dVector(colors)

o3d.visualization.draw_geometries([cloud])
```



## 实验总结

用自己的语言描述，点云和传统的2D图像有什么区别？从数据结构、数据特征、处理方法等角度，至少说出两点。

数据结构方面：点云包含点的三个维度，是空间点集，基本数据单元是体素，而传统2D图形是二维的，图片是二维像素点集，基本单元是像素。

数据特征方面：点云3D点的集和，2D图形是2D像素点的集和。

处理方法方面：点云可以对三个维度方向进行操作，有空间旋转、三维平移、翻转，而图像是两个维度操作，即普通的平移变换，对称变换。