[PointCloud - Open3D 0.18.0 documentation](https://www.open3d.org/docs/release/tutorial/t_geometry/pointcloud.html)

pcd.cluster\_dbscan: get the labels of cluster

pcd.crop(boundybox): crop

pcd.estimate\_normals: estimate the normals to each point

pcd.normalize\_normals(): Normalize point normals to length 1

pcd.compute\_mean\_and\_covariance(self): get the mean and covariance

pcd.get\_center(): get the center of mass

compute\_point\_cloud\_distance(target): compute the distance to another point clouds

pcd.get\_axis\_aligned\_bounding\_box(): get the largest bounding box of point clouds

pcd.get\_minimal\_oriented\_bounding\_box(): get minimal oriented bounding box

paint\_uniform\_color(color): paint color to points

remove\_non\_finite\_points(): remove non-finite or Nan points

remove\_statistical\_outlier(nb\_neighbors, std\_ratio, print\_progress=False): Removes points that are further away from their neighbors in average.

pcd.select\_by\_index(indices, invert=False): select points according to indices

**Conversion between tensor and legacy point cloud**

legacy\_pcd = pcd.to\_legacy()

print(legacy\_pcd, "\n")

tensor\_pcd = o3d.t.geometry.PointCloud.from\_legacy(legacy\_pcd)

print(tensor\_pcd, "\n")

## Bounding volumes

aabb = pcd.get\_axis\_aligned\_bounding\_box()

aabb.set\_color(o3c.Tensor([1, 0, 0], o3c.float32))

obb = pcd.get\_oriented\_bounding\_box()

obb.set\_color(o3c.Tensor([0, 1, 0], o3c.float32))

o3d.visualization.draw\_geometries(

[pcd.to\_legacy(), aabb.to\_legacy(),

obb.to\_legacy()],

zoom=0.7,

front=[0.5439, -0.2333, -0.8060],

lookat=[2.4615, 2.1331, 1.338],

up=[-0.1781, -0.9708, 0.1608])

## DBSCAN clustering

Given a point cloud from e.g. a depth sensor we want to group local point cloud clusters together. For this purpose, we can use clustering algorithms. Open3D implements DBSCAN [[Ester1996]](https://www.open3d.org/docs/release/tutorial/reference.html#Ester1996) that is a density based clustering algorithm. The algorithm is implemented in cluster\_dbscan and requires two parameters: eps defines the distance to neighbors in a cluster and min\_points defines the minimum number of points required to form a cluster. The function returns labels, where the label -1 indicates noise.

[21]:

ply\_point\_cloud = o3d.data.PLYPointCloud()

pcd = o3d.t.io.read\_point\_cloud(ply\_point\_cloud.path)

**with** o3d.utility.VerbosityContextManager(

o3d.utility.VerbosityLevel.Debug) **as** cm:

labels = pcd.cluster\_dbscan(eps=0.02, min\_points=10, print\_progress=**True**)

max\_label = labels.max().item()

print(f"point cloud has *{*max\_label + 1*}* clusters")

colors = plt.get\_cmap("tab20")(

labels.numpy() / (max\_label **if** max\_label > 0 **else** 1))

colors = o3c.Tensor(colors[:, :3], o3c.float32)

colors[labels < 0] = 0

pcd.point.colors = colors

o3d.visualization.draw\_geometries([pcd.to\_legacy()],

zoom=0.455,

front=[-0.4999, -0.1659, -0.8499],

lookat=[2.1813, 2.0619, 2.0999],

up=[0.1204, -0.9852, 0.1215])