1(a) Let's represent L to be a 3 x 3 matrix

$$x^\top Lx = x^\top (D - A)x = x^\top (\begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} - \begin{bmatrix} 0 & a_{12} & a_{13} \\ a_{21} & 0 & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix})x$$

$$= x^\top \begin{bmatrix} d_1 & -a_{12} & -a_{13} \\ -a_{21} & d_2 & -a_{23} \\ -a_{31} & -a_{32} & d_3 \end{bmatrix} x$$

$$= [x_i, x_j, x_k] \begin{bmatrix} d_1 & -a_{12} & -a_{13} \\ -a_{21} & d_2 & -a_{23} \\ -a_{31} & -a_{32} & d_3 \end{bmatrix} \begin{bmatrix} x_i \\ x_j \\ x_k \end{bmatrix}$$

$$= \begin{bmatrix} x_i d_1 - x_j a_{21} - x_k a_{31} & -x_i a_{12} + x_j d_2 - x_k a_{32} & -x_i a_{13} - x_j a_{23} + x_k d_3 \end{bmatrix} \begin{bmatrix} x_i \\ x_j \\ x_k \end{bmatrix}$$

By definition d is the degree of the vertex meaning that how many edges does this vertex have and a_{ij} is 1 if there is a edge between vertex i and j else 0. So we can easily verify that $\sum_{(i,j)} \| x_i - x_j \|^2 = x^\top Lx$

1(b) Let's still represent L by 3 x 3 matrix

$$L = D - A = \begin{bmatrix} d_1 & -a_{12} & -a_{13} \\ -a_{21} & d_2 & -a_{23} \\ -a_{31} & -a_{32} & d_3 \end{bmatrix}$$

$$Let\ \lambda_i\ represents\ eigenvalues\ and\ v_i\ represents\ eigenvectors$$

$$\lambda_i = v_i^\top L v_i$$

$$= v_i^\top M^\top M v_i$$

$$= (Mv_i)^\top (Mv_i)$$

It's symmetrix as we can see in the expression of L. Because λ_i can be written as the inner product of the vector Mv_i with itself, it shows that $\lambda_i \geq 0$ and so the eigenvalues of L are all non-negative. Thus it postive semi definite.

1(c)

$$\sum_{(i,j)} \| p_i - p_j \|^2 = P^\top LP$$

$$= [x, y, z]^\top L[x, y, z]$$

$$= x^\top Lx + y^\top Ly + z^\top Lz$$

2(a) Let's assume it to be 3 x 3

$$L_{norm} = D^{-1}L = I - D^{-1}A$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 1/d_1 & 0 & 0 \\ 0 & 1/d_2 & 0 \\ 0 & 0 & 1/d_3 \end{bmatrix} \begin{bmatrix} 0 & a_{12} & a_{13} \\ a_{21} & 0 & a_{23} \\ a_{31} & a_{32} & 0 \end{bmatrix}$$

$$= I - \begin{bmatrix} 0 & \frac{1}{d_1}a_{12} & \frac{1}{d_1}a_{13} \\ \frac{1}{d_2}a_{21} & 0 & \frac{1}{d_2}a_{23} \\ \frac{1}{d_3}a_{31} & \frac{1}{d_3}a_{32} & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -\frac{1}{d_1}a_{12} & -\frac{1}{d_1}a_{13} \\ -\frac{1}{d_2}a_{21} & 1 & -\frac{1}{d_2}a_{23} \\ -\frac{1}{d_3}a_{31} & -\frac{1}{d_3}a_{32} & 1 \end{bmatrix}$$

For each row the sum is the 1 - the sum of adjacency of this vertex divided by the degree of the vertex, the adjacency can only be 1 or 0 depending on the degree of this vertex. So it must be 1 - 1 for each 0. Thus Sum of each row of L_{norm} is zero

2(b) From L_{norm} representation above, each row of $L_{norm}P$ is

$$L_{norm}P = \begin{bmatrix} 1 & -\frac{1}{d_1}a_{12} & -\frac{1}{d_1}a_{13} \\ -\frac{1}{d_2}a_{21} & 1 & -\frac{1}{d_2}a_{23} \\ -\frac{1}{d_3}a_{31} & -\frac{1}{d_3}a_{32} & 1 \end{bmatrix} \begin{bmatrix} p_{1x} & p_{1y} & p_{1z} \\ p_{2x} & p_{2y} & p_{2z} \\ p_{3x} & p_{3y} & p_{3z} \end{bmatrix}$$

$[L_{norm}P]_1 = \begin{bmatrix} p_{1x} - \frac{1}{d_1}a_{12}p_{2x} - \frac{1}{d_1}a_{13}p_{3x} & p_{1y} - \frac{1}{d_1}a_{12}p_{2y} - \frac{1}{d_1}a_{13}p_{3y} & p_{1z} - \frac{1}{d_1}a_{12}p \end{bmatrix}$

$[p_{1x}, p_{1y}, p_{1z}]$ forms p_1 .

By definition, we can easily simplify it as $p_i - \frac{1}{N(p_i)} \sum_{p_j} p_j$

Thus, $\Delta p_i = [L_{norm}P]_i$

Type *Markdown* and LaTeX: α^2

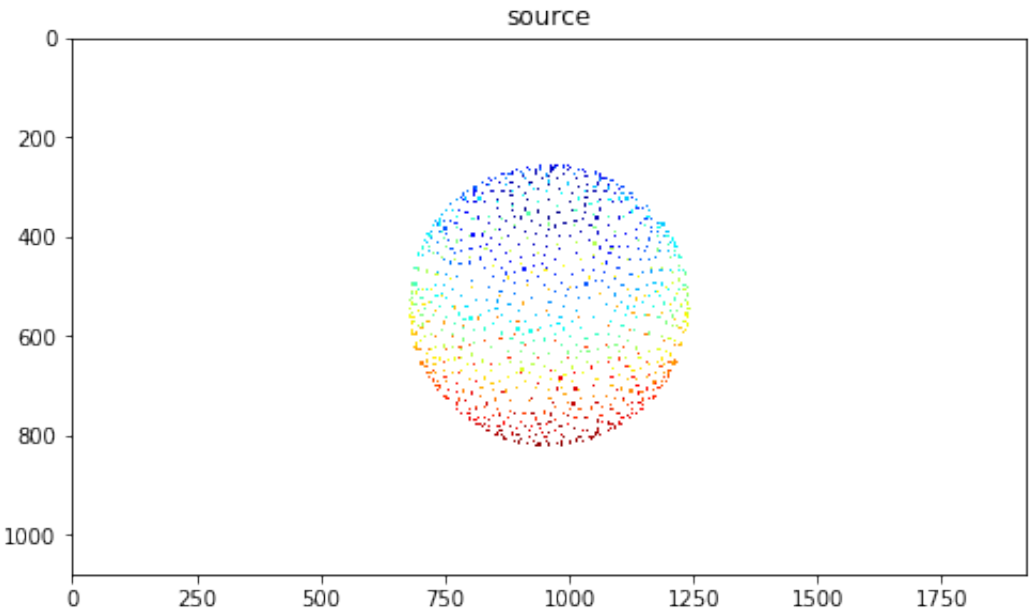
In [1]:
```
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt
import open3d

vis = open3d.visualization.Visualizer()
vis.create_window(visible = False)
def draw_geometries(geoms):
    for g in geoms:
        vis.add_geometry(g)
    view_ctl = vis.get_view_control()
    view_ctl.set_up((0, 1, 0))
    view_ctl.set_front((0, 2, 1))
    view_ctl.set_lookat((0, 0, 0))
    view_ctl.set_zoom(1)
    # do not change this view point
    vis.update_renderer()
    img = vis.capture_screen_float_buffer(True)
    plt.figure(figsize=(8,6))
    plt.imshow(np.asarray(img))
    for g in geoms:
        vis.remove_geometry(g)
```

```
Bad key "text.kerning_factor" on line 4 in
/opt/anaconda3/lib/python3.7/site-packages/matplotlib/mpl-data/stylel
ib/_classic_test_patch.mplstyle.
You probably need to get an updated matplotlibrc file from
https://github.com/matplotlib/matplotlib/blob/v3.1.3/matplotlibrc.tem
plate
(https://github.com/matplotlib/matplotlib/blob/v3.1.3/matplotlibrc.te
mplate)
or from the matplotlib source distribution
```

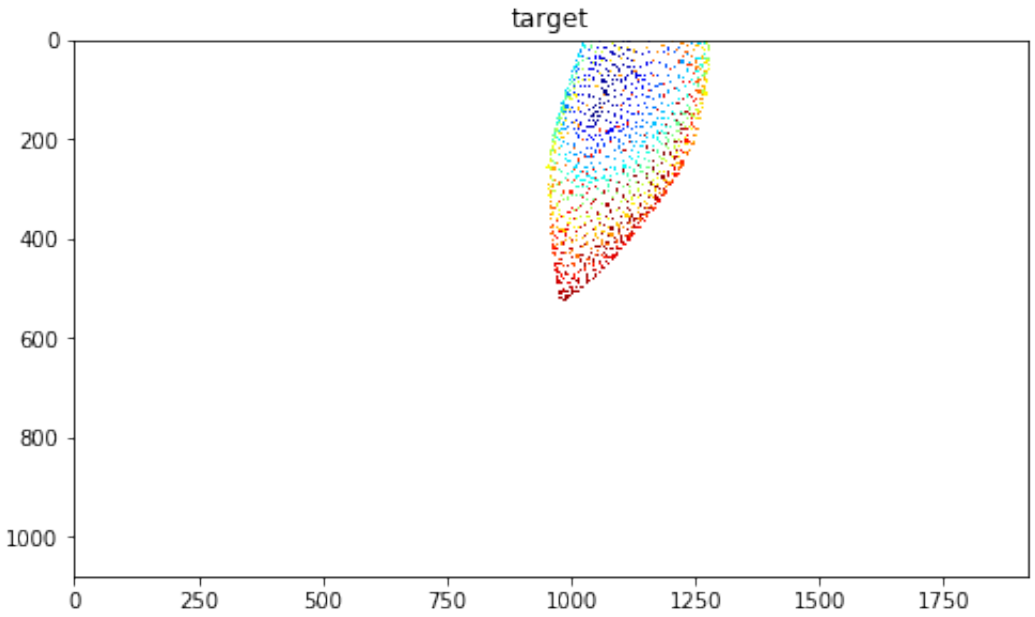
In [3]:
```
import trimesh
source_mesh = trimesh.load('source.obj')
pcd = open3d.geometry.PointCloud()
pcd.points = open3d.utility.Vector3dVector(source_mesh.vertices)
draw_geometries([pcd])
plt.title("source")
```

Out[3]: Text(0.5, 1.0, 'source')



In [4]:
```
target_mesh = trimesh.load('target.obj')
pcd = open3d.geometry.PointCloud()
pcd.points = open3d.utility.Vector3dVector(target_mesh.vertices)
draw_geometries([pcd])
plt.title("target")
```

Out[4]: Text(0.5, 1.0, 'target')



In [11]:
```
len(np.asarray(open3d.utility.Vector3dVector(source_mesh.vertices)))
```

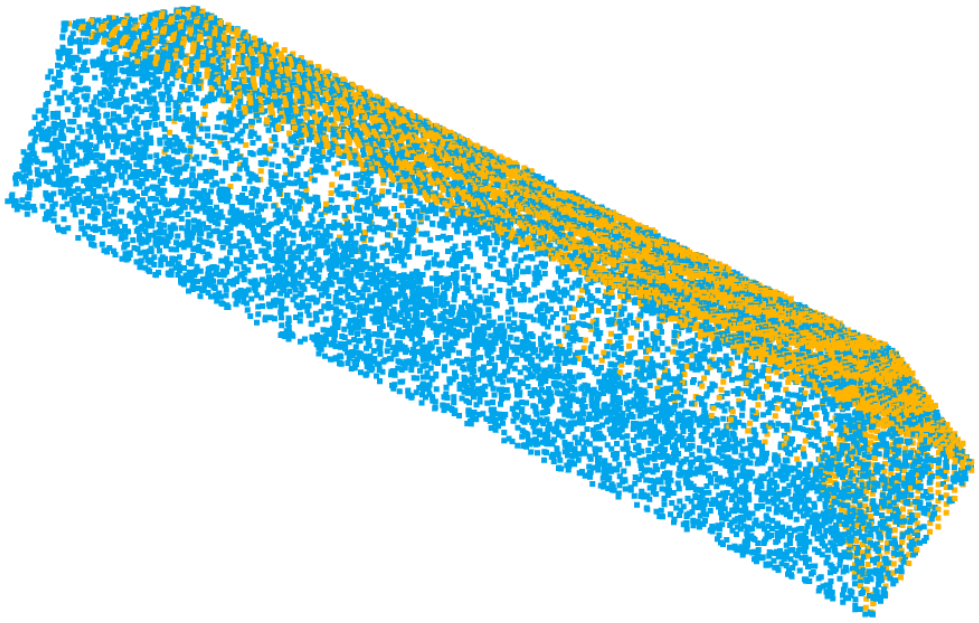
Out[11]: 962

In [ ]:
```
model = resnet18()
optim = torch.optim.Adam(model.parameters(), lr=0.0001)
model.cuda()
for epoch in range(epochs): print_count = 0
print_loss = 0 epoch_step = 0
for data in train_loader:
        epoch_step += 1
        print_count += 1
        result = model(data["image"].cuda())
        optim.zero_grad()
        loss = F.binary_cross_entropy(result, data["edge"].cuda())
        print_loss += loss.item()
        loss.backward()
        optim.step()
```

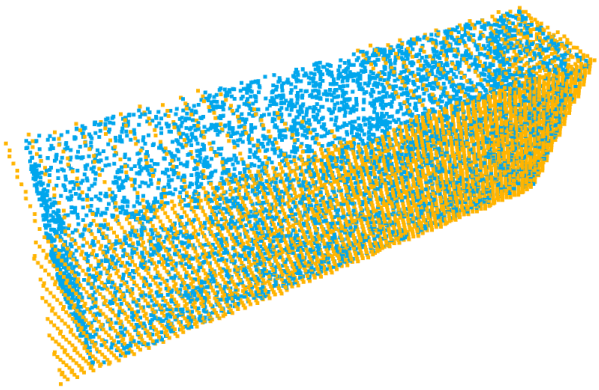
ICP Report

Username: gordonhu

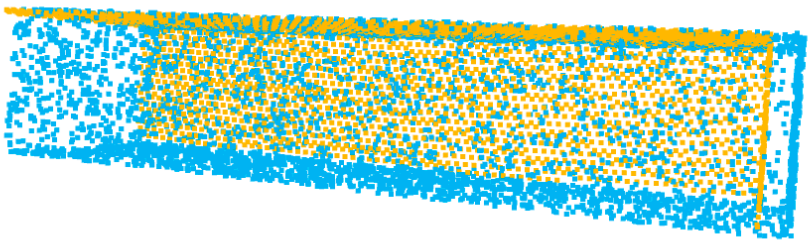
I first implemented all the preprocessing part of pointcloud, including scaling, transfer point clouds into worldframe. The source point cloud I used is the pointcloud lifted from the depth image and the target point cloud is by loading dae trimesh and then sampling 10000 points from the surface to be a pointcloud. I selected the first object -- jenga to experiment with. Without using icp first, I tried the ground truth pose and check if the visulization after the ground truth transformation will be aligned. Here is the result



And it shows that the two point clouds are aligned perfectly. Then I tried to use icp method in open3d by initilizing the transformation matrix as identity. The alignment is very bad since I used a very small throld. After tuning the threshold to 0.3, here's the result I get. It shows that the icp is working well.

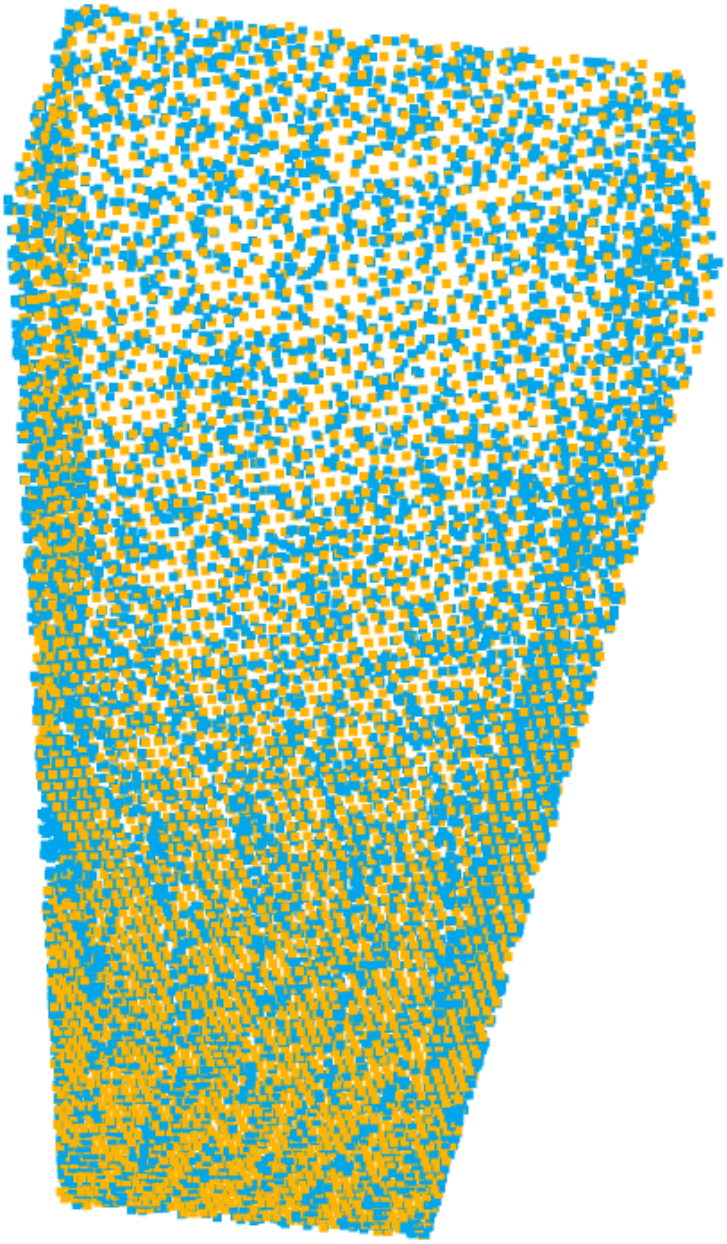


The open3d method is working pretty good and then I implemented my own icp method. Since the closest point may have the best correspondance. I used nearest neighbor to find the closet point and build the correspondance. Then I optimized the correspondance by minimizing the equation q - Rp and using svd to find the closed form solution of the equation. Update the source point cloud and then iteratively find the best correspondance and transformation matrix. Here's the result of my own icp



I also tried to use the global registration method to find a better initial transformation matrix since the result of icp is very depending on a better initilization. The fast version of global registration find the features of two pointcloud with correspondance. It downsample the point cloud to estimate normals then compute a FPFH feature for each point, which describes the local geometric property of a point. Then use nearest neighbor to find points with similar local geometric structures. I used a voxel size of 0.01 since 0.05 gives very few point. And the illustration of the downsampled point cloud forming correspondance and transformation shows that principal points were aligned really well.

And this is the final result after I used global registration initilization and icp method. The combined approach gives a really good result.



feedback:

time: 3days progress: very good professor explaiend really well.

In [ ]: