Jerry Chen

Qimin Ren

HW3

Part 1: Voxel Construction

To start off, we defined the dimensions of our voxel grid which we can use to determine the size of each voxel depending on the amount of voxels we wish to use. Then, we loop through each voxel point and project them through each calibration matrix to see if they fall within the silhouette images. If they pass each silhouette successfully, the voxel point would be added to the voxel grid.

Part 2: Determine Surface Points

Looping through each plane-axis (xy, yz, and xz), we determine the minimum and maximum voxel points which would constitute the surface of the model. These points all get added into the surface grid, and because there are overlapping surface points in each plane-axis, with each pass the points added previously are cleared from the original voxel list to avoid duplicates. There was a second method we tried which was to loop through the voxel grid and see if it's neighbors were also within the grid and if it were surrounded by valid neighbors, the point would not be a surface point. However, that method took too much time computationally since it had to search through the entire grid multiple times for its neighbors and float number precision was also a problem as the voxel size became smaller.

Part 3: Color Sampling

To determine which image to sample when coloring the surface points, in the previous step of determining surface points, we also record the axis or direction where we get the point from. So for example, in the z-axis the minimum points are bottom views and maximum point are top views. Then look through the images provided to correspond the image views and have each point sample from their respective images. Another way to do this would be to find the two nearest neighbors for each surface point and use them to determine the point's surface plane and it's normal vertex which we can use to determine the point's viewing plane and image to sample color from.

```
import numpy as np
from PIL import Image
import open3d as o3d

P = np.array([[776.649963,-298.408539,-
32.048386,993.1581875,132.852554,120.885834,-
759.210876,1982.174000,0.744869,0.662592,-0.078377,4.629312012],
    [431.503540,586.251892,-137.094040,1982.053375,23.799522,1.964373,-
657.832764,1725.253500,-0.321776,0.869462,-0.374826,5.538025391],
    [-153.607925,722.067139,-127.204468,2182.4950,141.564346,74.195686,-
637.070984,1551.185125,-0.769772,0.354474,-0.530847,4.737782227],
```

```
    [-823.909119,55.557896,-82.577644,2498.20825,-31.429972,42.725830,-
777.534546,2083.363250,-0.484634,-0.807611,-0.335998,4.934550781],
    [-715.434998,-351.073730,-147.460815,1978.534875,29.429260,-2.156084,-
779.121704,2028.892750,0.030776,-0.941587,-0.335361,4.141203125],
    [-417.221649,-700.318726,-27.361042,1599.565000,111.925537,-169.101776,-
752.020142,1982.983750,0.542421,-0.837170,-0.070180,3.929336426],
    [94.934860,-668.213623,-331.895508,769.8633125,-549.403137,-58.174614,-
342.555359,1286.971000,0.196630,-0.136065,-0.970991,3.574729736],
    [452.159027,-658.943909,-279.703522,883.495000,-262.442566,1.231108,-
751.532349,1884.149625,0.776201,0.215114,-0.592653,4.235517090]])

c_dict = {}
s_dict = {}
p_mat = np.zeros((3,4,8))
for i in range(8):
    c_dict[i] = np.asarray(Image.open("cam0" + str(i) + "_00023_0000008550.png"))
    s_dict[i] = np.asarray(Image.open("silh_cam0" + str(i) +
"_00023_0000008550.pbm"))
    p_mat[:,:,i] = np.reshape(P[i],(3,4))

x_range = 5
y_range = 6
z_range = 2.5
volume = x_range*y_range*z_range
vox_num = 10000000
vox_size = np.power((volume/vox_num),1/3)
vox_grid = []
surf_grid = []
d_grid = []

for x in np.arange(-2.5, 2.5, vox_size):
    for y in np.arange(-3, 3, vox_size):
        minz = 3
        maxz = -3
        for z in np.arange(0, 2.5, vox_size):
            pass_mat = np.zeros(8)
            coord = [x, y, z, 1]
            for i in range(8):
                point = np.dot(coord,np.transpose(p_mat[:,:,i]))
                point = point/point[2]
                # check if point is within bounds
                if((0<=point[1]<582) and (0<=point[0]<780)):
                    pass_mat[i] = s_dict[i][int(point[1]),int(point[0])]
            # if point is in the silhouette for all 8 views, mark as occupied
            if(np.sum(pass_mat) == 8):
```

```
                vox_grid.append([x,y,z])
                if(z<minz):
                    minz = z
                if(z>maxz):
                    maxz = z
        if(minz!=3 and maxz!=-3):
            surf_grid.append([x,y,minz])
            surf_grid.append([x,y,maxz])
            d_grid.append('zbot')
            d_grid.append('ztop')
def clear_grid(vox,surf):
    for s in surf:
        if(s in vox):
            vox.remove(s)
clear_grid(vox_grid,surf_grid)
for z in np.arange(0, 2.5, vox_size):
    for y in np.arange(-3, 3, vox_size):
        minx = 3
        maxx = -3
        for x in np.arange(-2.5, 2.5, vox_size):
            if([x,y,z] in vox_grid):
                if(x<minx):
                    minx = x
                if(x>maxx):
                    maxx = x
        if(minx!=3 and maxx!=-3):
            surf_grid.append([minx,y,z])
            surf_grid.append([maxx,y,z])
            d_grid.append('xleft')
            d_grid.append('xright')
clear_grid(vox_grid,surf_grid)
for z in np.arange(0, 2.5, vox_size):
    for x in np.arange(-2.5, 2.5, vox_size):
        miny = 3
        maxy = -3
        for y in np.arange(-3, 3, vox_size):
            if([x,y,z] in vox_grid):
                if(y<miny):
                    miny = y
                if(y>maxy):
                    maxy = y
        if(minz!=3 and maxz!=-3):
            surf_grid.append([x,miny,z])
            surf_grid.append([x,maxy,z])
            d_grid.append('yback')
```

```
            d_grid.append('yfront')

color_grid = []
d_dict = {
    "ztop": 6,
    "zbot": 3,
    "yback": 3,
    "yfront": 0,
    "xright": 2,
    "xleft": 5
}
for i in range(0,len(d_grid)):
    view = d_dict[d_grid[i]]
    coord = surf_grid[i]+[1]
    point = np.dot(coord,np.transpose(p_mat[:,:,view]))
    point = point/point[2]
    rgb = c_dict[view][int(point[1]),int(point[0])]
    color_grid.append([float(rgb[0])/255,float(rgb[1])/255,float(rgb[2])/255])

print(vox_size,len(color_grid),len(surf_grid))
pcd = o3d.geometry.PointCloud()

pcd.points = o3d.utility.Vector3dVector(surf_grid)
pcd.colors = o3d.utility.Vector3dVector(color_grid)
o3d.io.write_point_cloud("./surf.ply", pcd)
o3d.visualization.draw_geometries([pcd])
```

Image of final model