HW03

Question 3: - Apply RANSAC algorithm (or any others you prefer) to the 3D voxel space points to find a ground plane model. Print out your plane model parameter values result, visualize the plane with the points in the 3D

In [1]:
```
import numpy as np
import argparse

filename   = str("bin_files/002_00000001.bin")
pointcloud = np.fromfile(filename, dtype=np.float32)
pointcloud = pointcloud.reshape([-1,4])

print('LiDAR data loaded as a variable pointcloud')

str1= str('\nLidar data file : ') + str(filename) + str('\nSize of pointclo
ud data = ') + str(pointcloud.shape)
print(str1)
```

```
LiDAR data loaded as a variable pointcloud

Lidar data file : bin_files/002_00000001.bin
Size of pointcloud data = (92246, 4)
```

In [14]:
```
def visualize_3d(pointcloud,cloud_color,Point_size):
    import pptk
    import numpy as np

    # Extract first three points as x y z inputs and 4th for reflectivity v
alue
    P = pointcloud[:,0:3]

    a = pointcloud.shape[0]
    R = np.ones((a))*20

    if pointcloud.shape[1]==4:
        R   = pointcloud[:,3]

    # define color channels
    rgb = np.ones((P.shape))*cloud_color    # for grayish effect [200,200,2
00]

    rgb[:,0] = rgb[:,0]*(255-R)/255
    rgb[:,1] = rgb[:,1]*(255-R)/255
    rgb[:,2] = rgb[:,2]*(255-R)/255

    if len(cloud_color)>3:
        rgb = cloud_color

    # Visualize point cloud
    v = pptk.viewer(P)
    v.attributes(rgb / 255, R)
    v.set(floor_color = [0,0,0,0.5])
    v.set(lookat = [0,0,0])             # set zero /ego vehicle coordinate
    v.set(point_size=Point_size)        # for better visualization point_si
ze = 0.001
```

In [3]:
```
import pcl

cloud = pcl.PointCloud(np.array(pointcloud[:,0:3], dtype=np.float32))
seg   = cloud.make_segmenter_normals(ksearch=50)

seg.set_optimize_coefficients(True)
seg.set_model_type(pcl.SACMODEL_PLANE)
seg.set_normal_distance_weight(0.07)
seg.set_method_type(pcl.SAC_RANSAC)

seg.set_max_iterations(100)
seg.set_distance_threshold(0.25)

inliers, model = seg.segment()

if len(inliers) == 0:
        print('Could not estimate a planar model for the given dataset.')
        exit(0)

#Points here is a nx3 numpy array with n 3d points.
#Model will be [a, b, c, d] such that ax + by + cz + d = 0
print('Model coefficients: ' + str(model[0]) + ' ' + str(model[1]) + ' ' +
str(model[2]) + ' ' + str(model[3]))
print('Model inliers: ' + str(len(inliers)))
print(len(model))
```

```
Model coefficients: 0.0145805468783 0.00373710296117 0.99988669157 1.558219
90967
Model inliers: 39196
4
```

Visualize the plane with the points in the 3D

In [10]:
```
# Visualize ground plane points
ground_pointcloud = pointcloud[inliers,:]
visualize_3d(ground_pointcloud,[200,200,200],0.001)
```

Remove all the ground planes points in the 3D voxel space points, visualize all the off-ground points in the 3D

In [11]:
```
abc = np.arange(0,pointcloud.shape[0],1)
outliers = np.delete(abc,inliers)
# Visualize off ground plane points
off_ground_pointcloud = pointcloud[outliers,:]
visualize_3d(off_ground_pointcloud,[200,200,200],0.001)
```

Visualize ground points and off groud points

In [12]:
```
ground_pointcloud.shape
off_ground_pointcloud.shape

final_pointcloud       = np.empty([0,4])
cloud_color            = np.empty([0,3])

final_pointcloud       = np.append(final_pointcloud,ground_pointcloud,axis=
0)
ground_cloud_color     = np.ones([ground_pointcloud.shape[0],3])*[0,150,0]
cloud_color            = np.append(cloud_color,ground_cloud_color,axis=0)

final_pointcloud       = np.append(final_pointcloud,off_ground_pointcloud,a
xis=0)
off_ground_cloud_color = np.ones([off_ground_pointcloud.shape[0],3])*[200,2
00,200]
cloud_color            = np.append(cloud_color,off_ground_cloud_color,axis=
0)
```

In [15]:
```
visualize_3d(final_pointcloud,cloud_color,0.001)
```

In [ ]:
```

```