Assignment 2

Pointcloud data preprocessing overview:

- The point cloud data stored in pcd format is analyzed using Open3D library.
- In the first pcd file there are 45785 points. These points are downsampled using voxel grid method by tuning the hyperparameter voxel_size.
- After downsampling we perform segmentation on the point cloud data to separate the inliers and outliers. In our case the inliers are points that lie on the ground plane, and outliers are point having considerable height from the ground plane.
- The outlier cloud is used in the DBSCAN algorithm to form clusters and detect obstacles.
- The DBSCAN algorithm yields a list of cluster indexes, associating each point cloud coordinate with its respective cluster.
- Later, we visualize the clusters using the Open3d library. To avoid bounding box formation around small clusters, one more hyperparameter is defined as Min_points. Bounding box formation will only take place when the number of points inside a box is greater than min_points.

DBSCAN algorithm overview:

It classifies the data points into three categories: core points, border points, and noise.

1. Core point: A data point that has at least a specified number of data points (MinPts) within a specified radius (eps). They expand the cluster
2. Border point: A border point is a point that is within the specified radius (eps) of a core point but does not have the required number of neighbors.
3. Noise Point: A point that is neither a core point nor a border point.

Algorithm:

Select a Data Point: Start with an arbitrary data point that has not been visited.

Check Density: Determine if the data point is a core point by counting how many points are within the specified radius (epsilon).

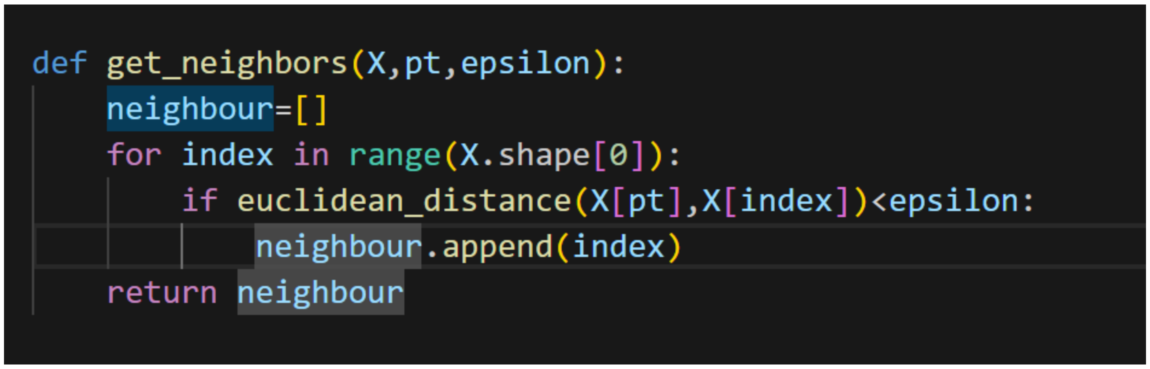
Expand Cluster: If the data point is a core point, start a new cluster and expand it by adding all reachable points within the specified radius. Mark each point as visited.

Continue Iteratively: Repeat the process for unvisited data points until all points have been processed.

Identify Noise: Any unvisited point that is not part of a cluster is considered noise.

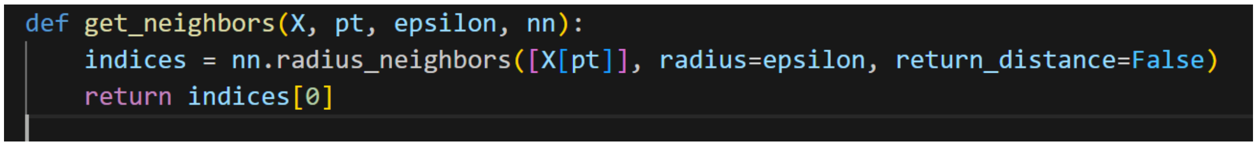
Hyperparameters: The two main hyperparameters of the DBSCAN algorithm are epsilon and MinPts.

In the initial approach of DBSCAN, neighbors of a point were calculated by comparing the distance with all points using a for loop. This significantly reduced the computational efficiency.



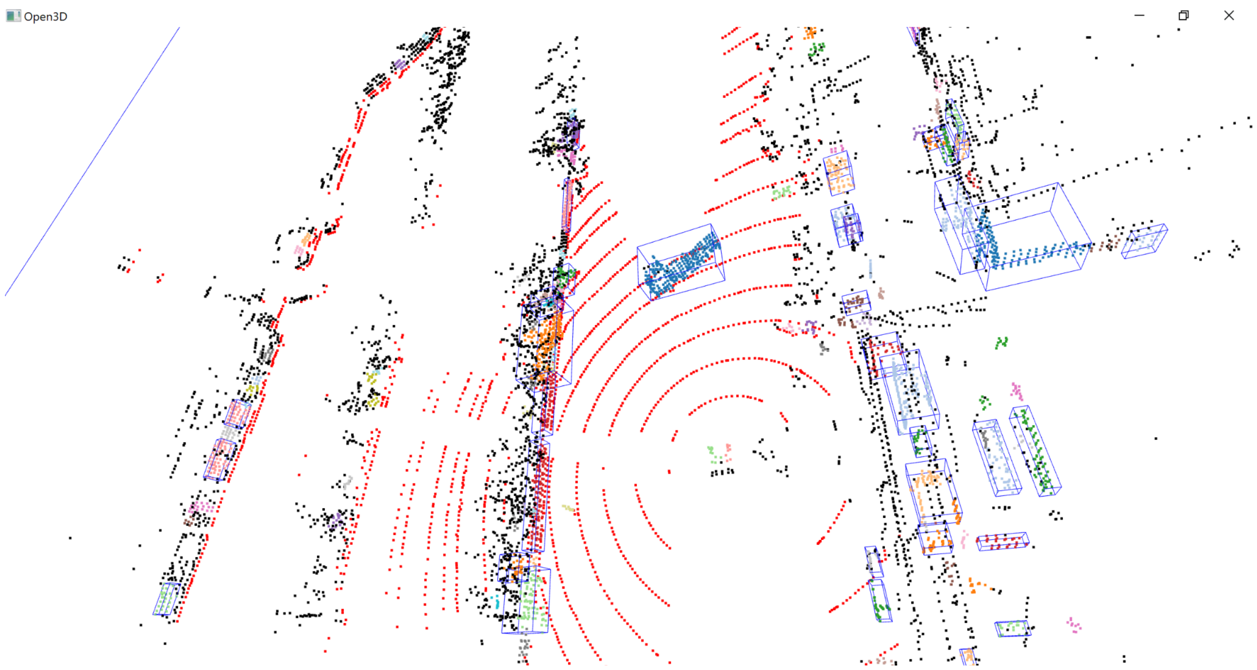
```python
def get_neighbors(X,pt,epsilon):
    neighbour=[]
    for index in range(X.shape[0]):
        if euclidean_distance(X[pt],X[index])<epsilon:
            neighbour.append(index)
    return neighbour
```

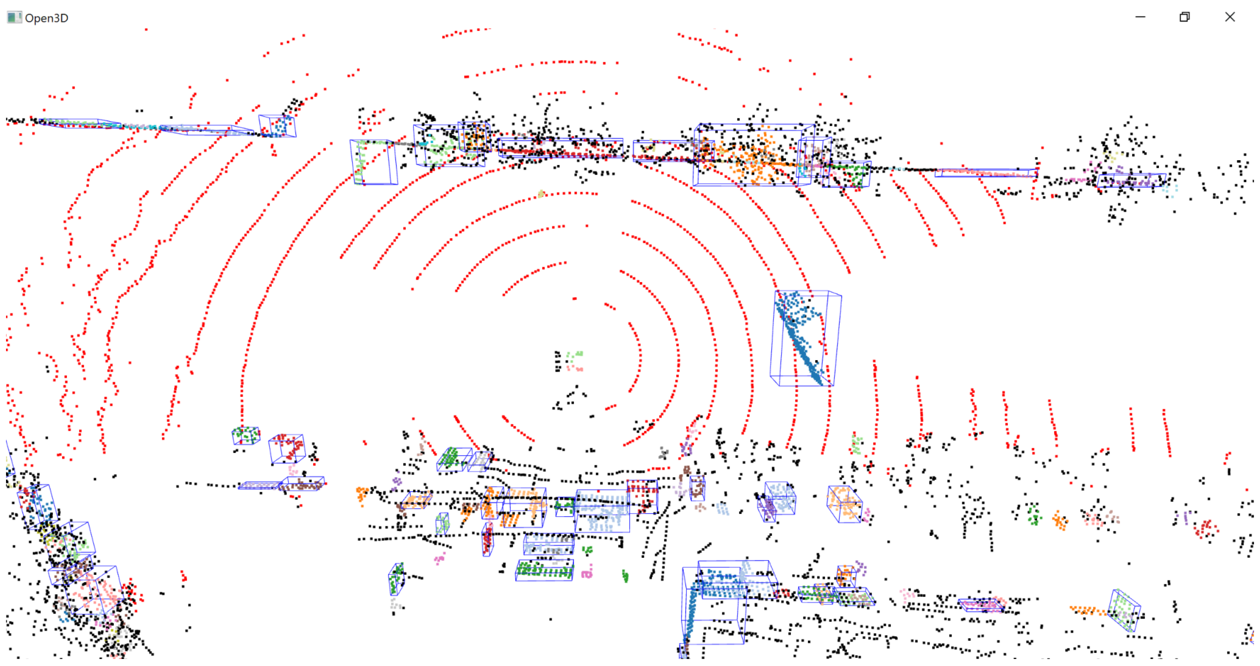
In the second approach, I optimized it by adding a fast region query using an index structure with the help scikit-learn library's NearestNeighbours. This significantly improved the computational time. The time taken to perform clustering over 10600 points by the optimized algorithm is approximately 6s much less than simple dbscan approach.



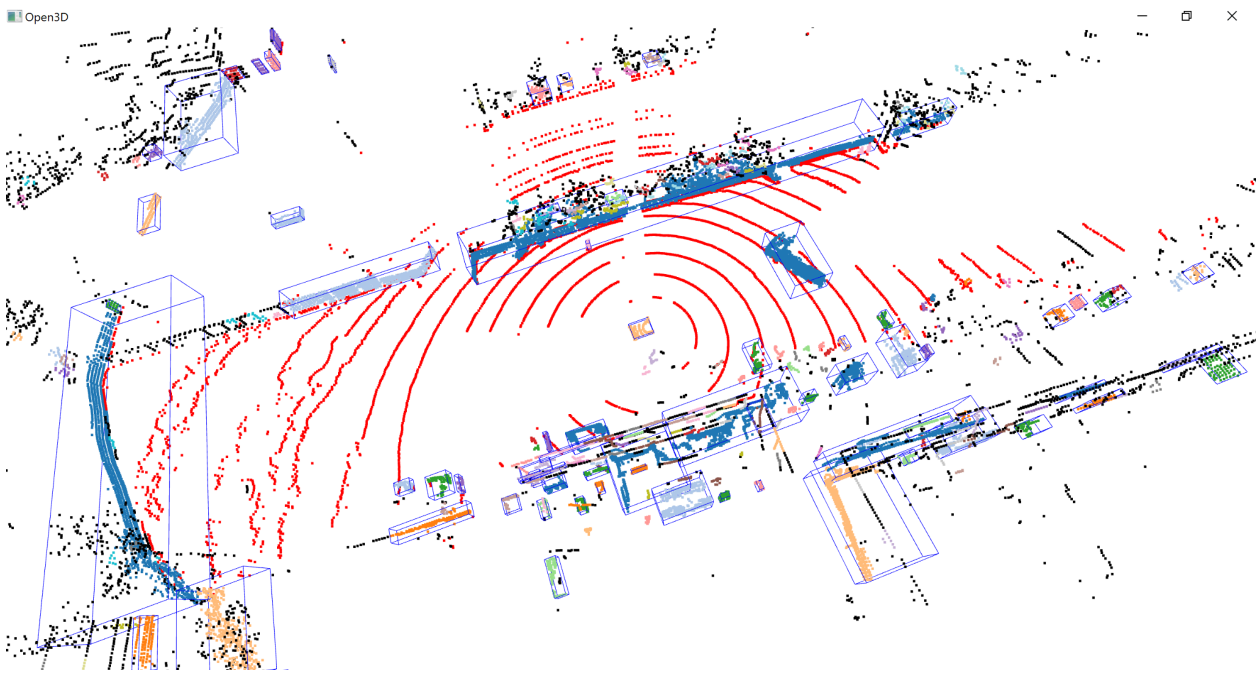
```python
def get_neighbors(X, pt, epsilon, nn):
    indices = nn.radius_neighbors([X[pt]], radius=epsilon, return_distance=False)
    return indices[0]
```

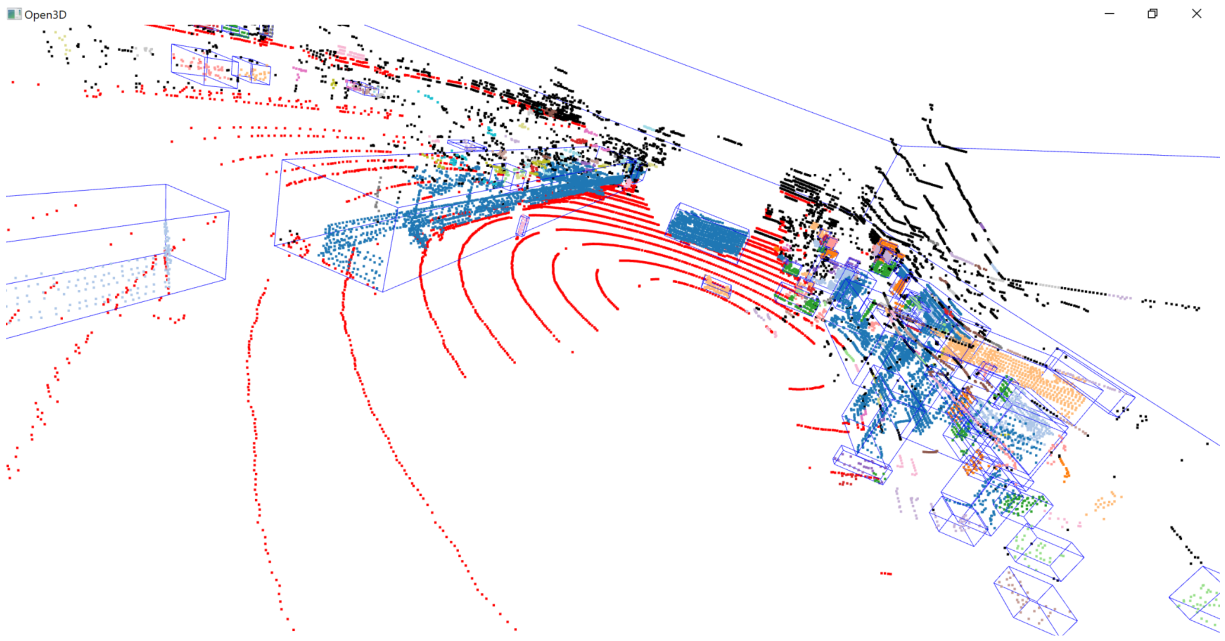
Results:



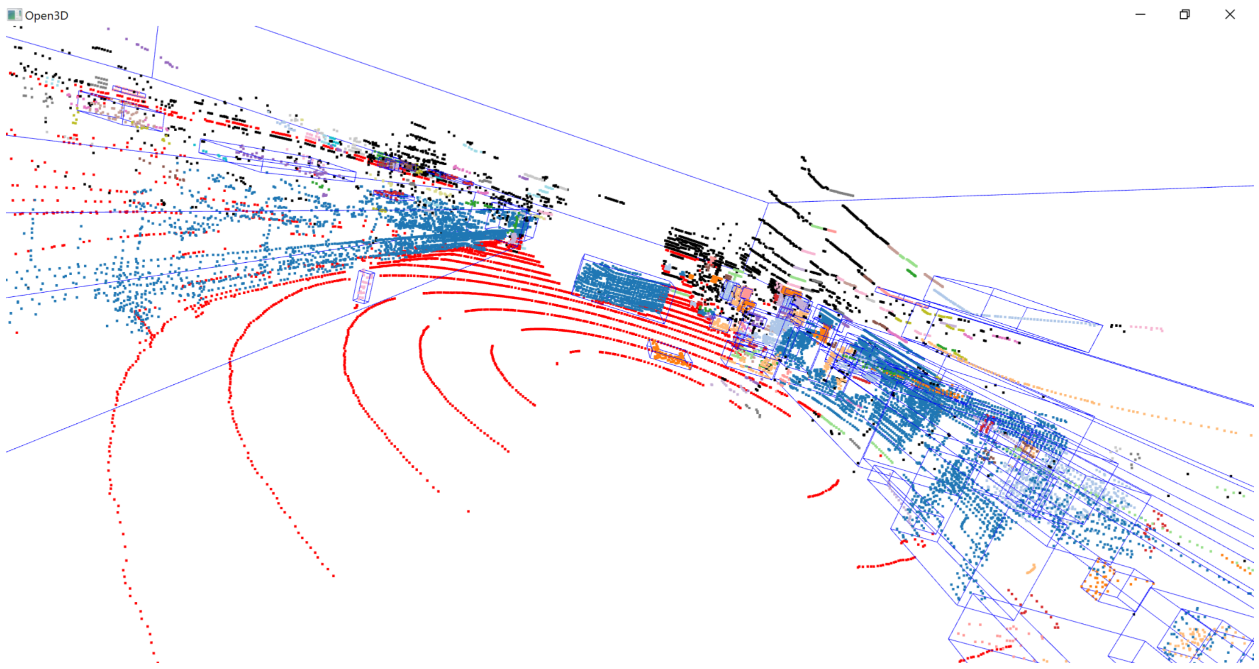


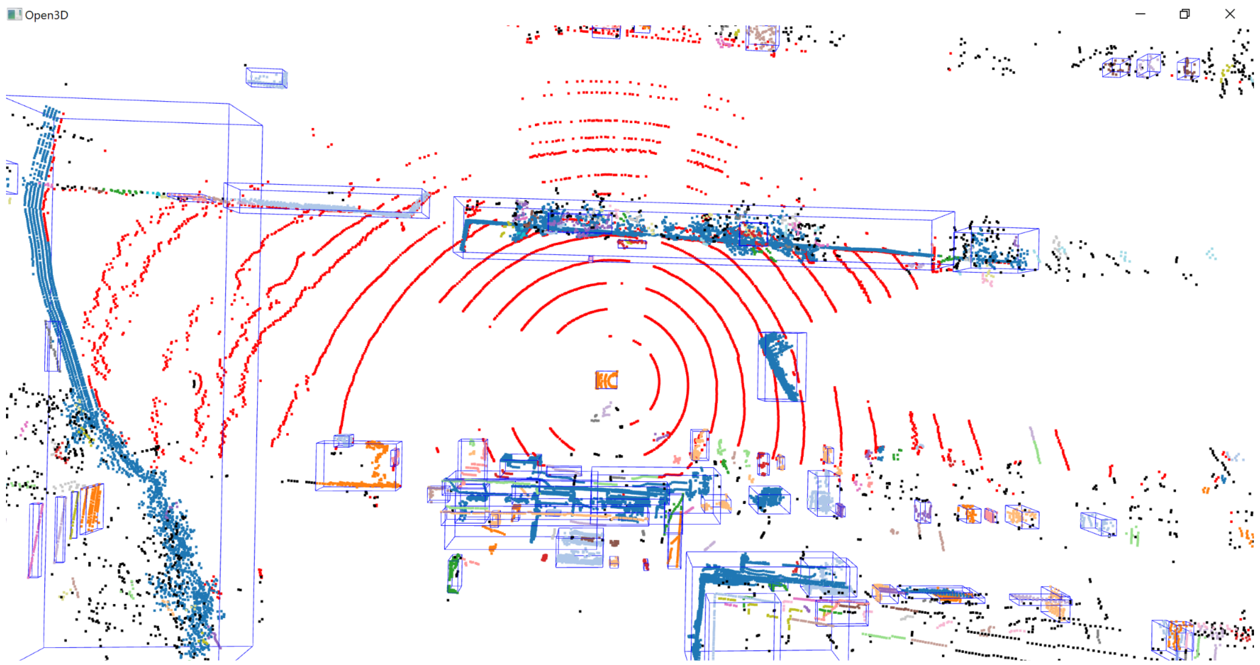
1. Eps = 0.55
   min_pts = 7
   min_points_in_cluster = 20
   voxel_size = 0.4
   no.pts = 15420





2. Eps = 0.55
   min_pts = 7
   min_points_in_cluster = 20
   voxel_size = 0.2
   no.pts = 24922





3. Eps = 0.65
   min_pts = 6
   min_points_in_cluster = 20
   voxel_size = 0.2
   no.pts = 24922

Inference:

- In larger epsilon, clusters are more expansive, and points that are farther apart may be considered part of the same cluster.
- For larger Minpts it requires a higher density of points to form a core point, resulting in smaller and denser clusters.
- In our pcd files, epsilon varies between (0.5,0.8), and Minpts lie in the integral range of 6-8.