National University of Singapore

College of Design and Engineering

ME5413 Autonomous Mobile Robotics: Homework 1

Author:
Yihui Chen A0263115N
Renjie Wang A0263387U

Supervisor:
Prof. Marcelo H Ang Jr

Email Address: *e1010473@u.nus.edu*
e1010745@u.nus.edu

February 8, 2023

1 Task 1: Lidar Clustering

In task 1, we need to perform lidar clustering for all objects in the scene given the set of 10 point cloud lidar samples. Different methods are tried and compared including DBSCAN algorithm that introduced in class. The clustering results are saved as the form of dictionary in a json file.

1.1 Ground Point Cloud Segmentation

Before doing lidar clustering, ground point cloud should first be segmented from the scene to avoid producing error in the following process. Since the ground point clouds are almost on the same plane, we selfly write a simple RANdom SAmple Consensus (RANSAC) model to detect and filter it.

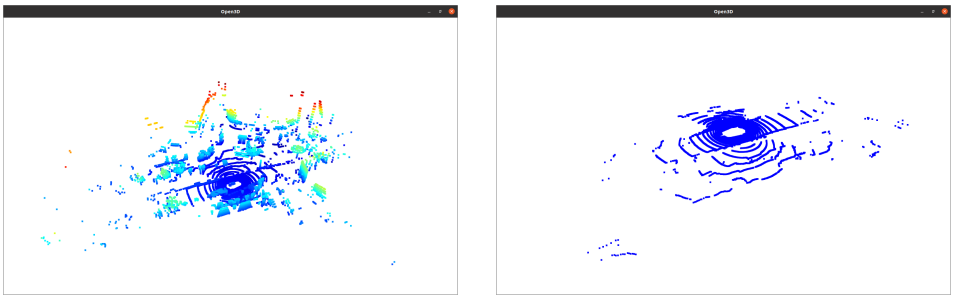
According to RANSAC algorithm (Alg. 1), we can regard the groud points as inliers while the surrounding points as outliers. The plane model can be explained as $aX + bY + cZ + d = 0$, which has 4 parameters and only need 3 points to solve it. In Step3, we assume the data point to be inliers if its distance to the model plane is less than σ ($\sigma = 0.4$ in our results). To accelerate the iteration, we also change the maximum iteration number when updating the model parameters according to Eq. 1:

Algorithm 1 RANSAC Algorithm

while $iter < iter_{max}$ **do**
  Step1: Randomly choose the smallest datset to estimate the model;
  Step2: Solve the model by the chosen dataset;
  Step3: Use all the other data to count the number of inliers;
  Step4: Compare the number of inliers of the current model and the best model, update the model;
  if the model is good enough **then**
    break;
  end if
end while

$$iter = \frac{log(1-P)}{log(1-t^n)} \tag{1}$$

where P is the probability that we hope RANSAC algorithm to get the correct model (0.99 in our case), t is the ratio of inliers in the whole dataset, and n is the number of points. We consider the model is good enough if the ratio of inliers is more than 0.4. Take the frame1 as example, Fig. 1 visualize the cloud points before and after ground segmentaion. The performance is quite good in this case, but it may sometimes show some error because of the randomness of the algorithm.



(a) Initial pointcloud (b) Segmented ground pointcloud

Figure 1: Cloud points before and after ground segmentaion in frame1

1.2 DBSCAN Algorithm

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm clusters the objects based on density. It defines a cluster as the maximal set of density-reachable points and is able to detect clusters of any shape. DBSCAN has only two parameters *eps* and *min_samples*, which defines the core samples that are in areas of high density. As shown in Fig. 2, the red points are core samples since there are more than *min_samples* (5) points around them in *eps* distance. Those density-reachable core samples and non-core samples (black ones) around them forms a cluster. Therefore, there are 2 clusters in Fig. 2, and those black points outside circles are detected as noise points.

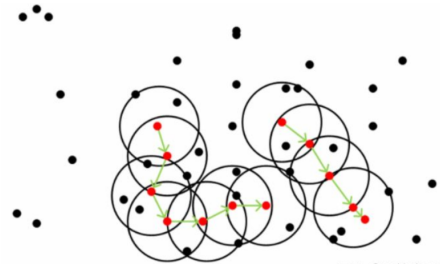
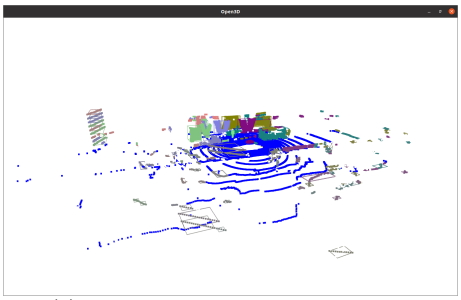
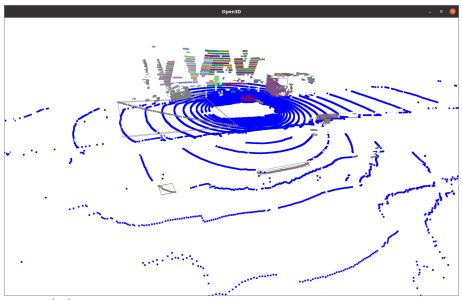


Figure 2: DBSCAN clustering principle

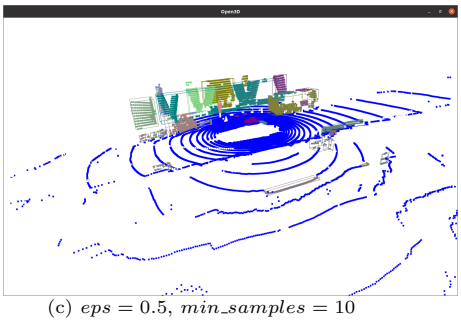
We paint the segmented ground point cloud blue and don't use them for clustering. We label different clusters in the scene with different colors, and mark noise points white. Since there may be large number of clusters, the colors of different clusters can be close, we plot bounding boxes for each cluster and store their information in the json file. We test the DBSCAN algorithm with different *eps* and *min_samples*, the results are shown in Fig. 3.



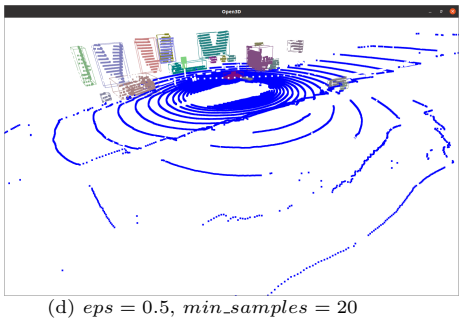
(a) $eps = 0.5$, $min\_samples = 5$
$n\_cluster = 166$, $n\_noise = 3256$



(b) $eps = 0.25$, $min\_samples = 5$
$n\_cluster = 200$, $n\_noise = 5835$



(c) $eps = 0.5$, $min\_samples = 10$
$n\_cluster = 44$, $n\_noise = 4847$



(d) $eps = 0.5$, $min\_samples = 20$
$n\_cluster = 21$, $n\_noise = 6750$

Figure 3: Clustering results by DBSCAN algorithm for frame1

It is easy to see that larger *min_samples* or smaller *eps* stands for higher density to form a cluster. In Fig. 3b, the density requirement is too high and causes bad clustering performance and large number of noise points. Also compare Fig. 3c and Fig. 3d, where we remain *eps* the same and increase the *min_samples* requirements, we can find the clustering accuracy increases but it loses more information and generates more noise points. We finally choose the case Fig. 3c as results written to the json file.

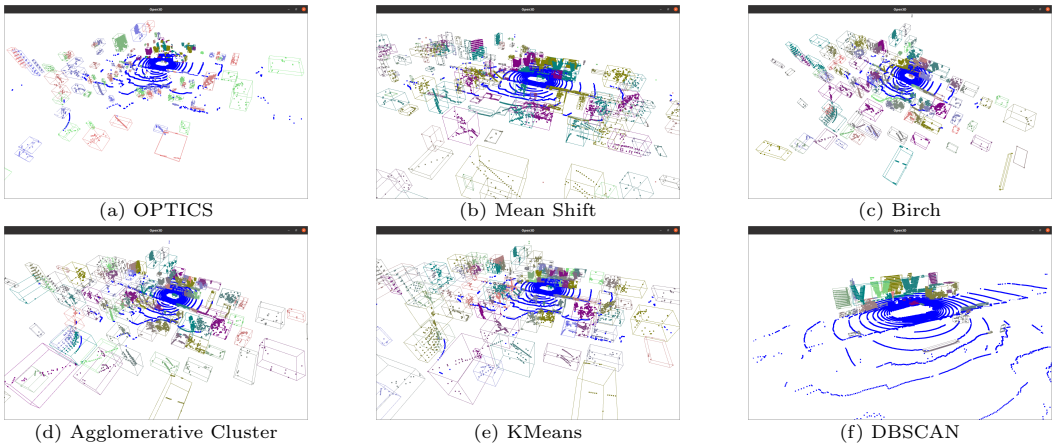
1.3 Comparison between Different Clustering Methods

OPTICS is an extension version of DBSCAN and make the algorithm no longer sensitive to the radius *eps*. Given the *min_samples*, OPTICS algorithm will produce an augmented cluster order for analysis and automatically choose the appropriate *eps*. Here we remain $min\_samples = 10$ and the result is shown in Fig. 4a, where more clusters are formed but some noise is also included.

Meanshift is another density-based clustering algorithm. It assumes that different clusters conform to different probability density distributions and aims at finding the fastest direction in which the sample density increases. The areas with high density correspond to the maximum value of the distribution, so data points should belong to one cluster if their density converge to the local maximum. Meanshift adjusts number of clusters by bandwidth, and here we set $quantile = 0.01, n\_samples = 1000$ to estimate the bandwidth, the result is shown in Fig. 4b

Birch algorithm builds an Clustering Feature Tree to realize quick clustering. It is computationally efficient and suits for large dataset. Usually Birch needs some priori knowledge about cluster number and cannot directly detect noise points, which requires removing some mini-cluster afterwards. The results is shown in Fig. 4c where we set estimated cluster number to be 100. The results of CF Tree can be optimized by hierarchical clustering methods such as AgglomerativeClustering (Fig. 4d, $n\_clusters = 100$).

K-Means is a partition-based clustering algorithm that uses distance as a measure of similarity between data objects. We also set $n\_clusters = 100$ and $tol = 0.001$, the results is shown in Fig. 4e.



(a) OPTICS (b) Mean Shift (c) Birch

(d) Agglomerative Cluster (e) KMeans (f) DBSCAN

Figure 4: Comparison between different clustering algorithm for frame1

From the experiment results, we find that Birch, AgglomerativeCluster and KMeans need prior knowledge of cluster numbers and are easy to include noise points. Among them, K-Means can only process convex clustering. Density-based methods instead can do clustering without estimating the total number of clusters, and DBSCAN & OPTICS perform well in our scenes.