National University of Singapore

College of Design and Engineering

ME5413 Autonomous Mobile Robotics: Homework 1

Group 9:

Chen Yihui A0263115N

Wang Renjie A0263387U

Supervisor:

Prof. Marcelo H Ang Jr

Email Address: e1010473@u.nus.edu

e1010745@u.nus.edu

February 11, 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.

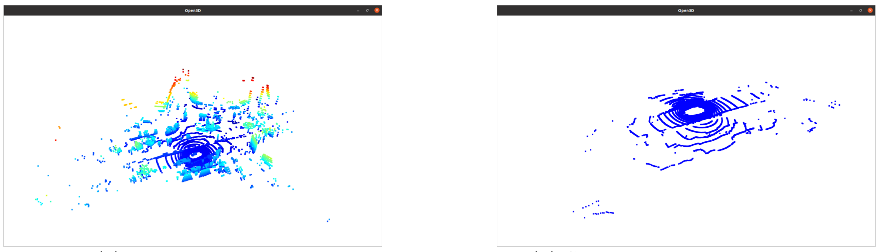

(a) Initial pointcloud (b) Segmented ground pointcloud

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.

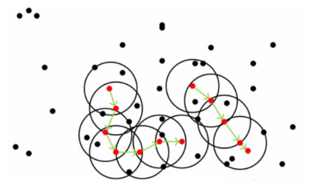


Figure 2: DBSCAN clustering principle

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.

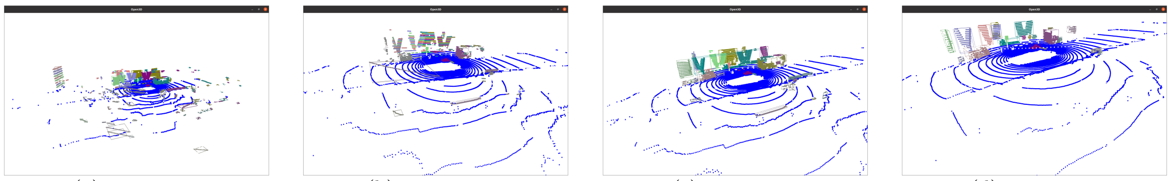

(a) $eps = 0.5$, $min\_samples = 5$ $n\_cluster = 166$, $n\_noise = 3256$
(b) $eps = 0.25$, $min\_samples = 5$ $n\_cluster = 200$, $n\_noise = 5835$
(c) $eps = 0.5$, $min\_samples = 10$ $n\_cluster = 44$, $n\_noise = 4851$
(d) $eps = 0.5$, $min\_samples = 20$ $n\_cluster = 21$, $n\_noise = 6750$

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.

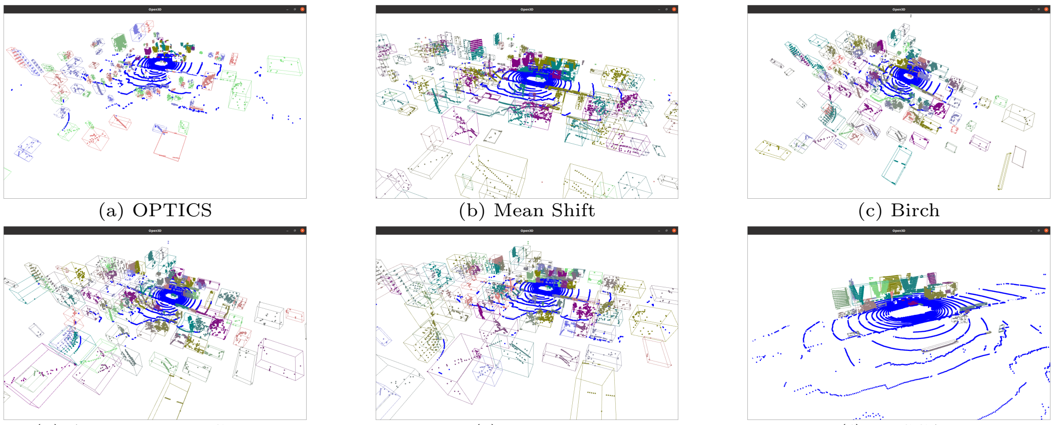

(a) OPTICS (b) Mean Shift (c) Birch
(d) Agglomerative Cluster (e) KMeans (f) DBSCAN

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 (Fig. 4f) & OPTICS perform well in our scenes.

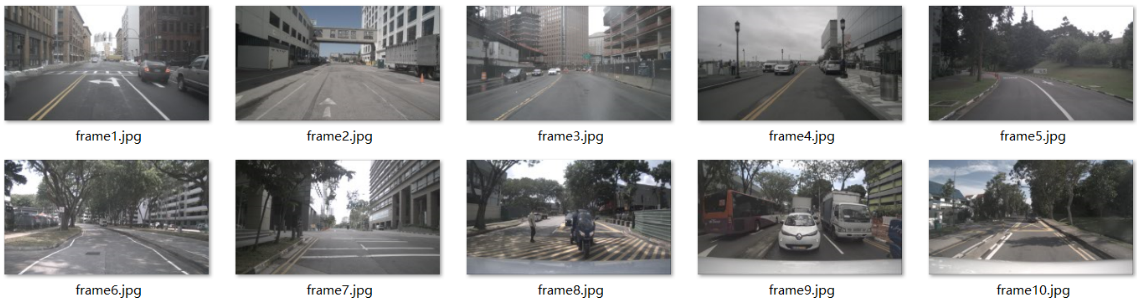
2 Task2: Image Segmentation

In task 2, we need to perform image segmentation on the given ten street scene images, give our own analysis on the algorithm and suggest improvement to it. The method of DeepLabv3 is mainly used to execute the code on MATLAB. The segmented result image is saved in the "Result" folder.

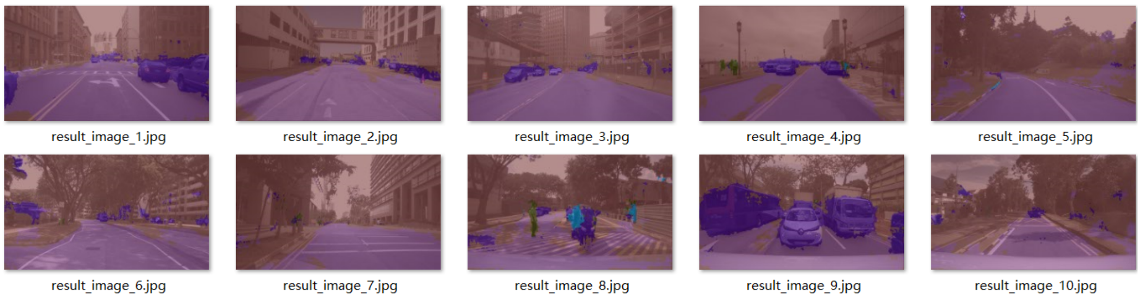
2.1 Deeplabv3 Retrained Model

Deeplabv3+ belongs to the semantic segmentation network, which classifies each pixel in the image to segment the image. Semantic segmentation has a wide range of applications, including road segmentation for autonomous driving. This pre-trained model uses the CamVid dataset from the University of Cambridge, and the weights are initialized by a pre-trained Resnet-18 network.

In this question, we directly perform image segmentation using DeepLabv3+ pretrained model on the 10 sample images. And the output classes of the network are modified to 5 classes: 1. vehicles 2. bicycles and motorcycles 3. pedestrians 4. others 5. drivable surface. The segmentation result is shown in Fig. 5.



(a) Original sample images



(b) Segmented Images

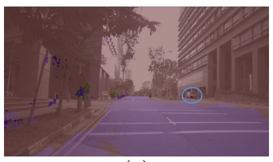
Figure 5: Segmentation performed with DeepLabv3+

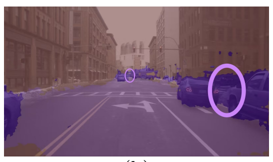
2.2 Analysis on the segmentation performance

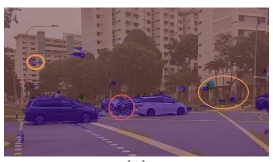
It can be seen that the performance of image segmentation is not completely accurate, and there are still some failure points. To illustrate the failure points, we also tested 5 images of our own.

The failure points can be concluded as follows:

1. When the colour of the object is close to the background, it may not be recognized by network. (Blue circle in Fig. 6a)
2. When two objects are too close, there is a chance that the background between objects will be segmented into object class. (Purple circle in Fig. 6b)
3. Inaccurate segmentation when objects overlap together. (Red circle in Fig. 6c)
4. Hard to be recognized by network when objects are too small in the image. (Green circle in Fig Fig. 6d)
5. Mis-segmentation noise in irrelevant places in the images. (Orange circle in Fig. 6c and Fig. 6d)


(a)


(b)


(c)

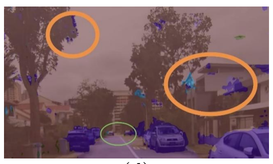

(d)

Figure 6: Example images showing failure points

2.3 Improvement to the segmentation method

Since the segmentation results of the neural network are not perfect, can we improve the segmentation method in the pre-processing and post-processing stages? We first realized that this is different from the traditional image segmentation problem. Methods such as filtering noise points, increasing image contrast, etc. are not effective.

Notice that we use the method of segmentation using the Deeplabv3 pre-trained neural network model. The principle is to update and converge the weights of the neural network in the learning of the dataset, so that the segmentation can be performed in new input images.

1. Therefore, we realized that in the pre-processing stage, we firstly should resize the picture to the size of the network's input image (the picture in the CamVid dataset) (720x960).
2. Secondly, considering that the images taken by ourselves may not be clear enough under different equipment and environment, we decide to preprocess the images to be input by properly increasing the image contrast. Notice that we use the edge-aware local contrast manipulation of images in MATLAB (localcontrast function). The reason is: although the method Traditional linear contrast boosting and histogram equalization are the most widely used global image enhancement methods and simple, it does not take local information into account.
3. In addition, we noticed that some images have too many surface details of some objects, which interferes with the accuracy of segmentation. Here we appropriately use the method of fast local Laplacian filtering of images in MATLAB (locallapfilt function) to smooth image details Without affecting edge sharpness.

Then we implemented it again on the 10 sample image. The results are good as noise points decreasing (Fig. 7. We can see the noise problem is better and segmentation is improved a little bit after this process.

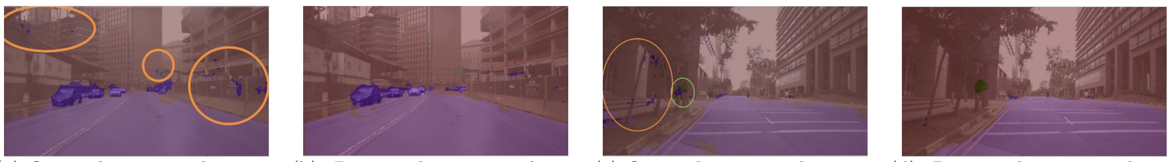

(a) Original segmented image1 (b) Processed segmented image1 (c) Original segmented image2 (d) Processed segmented image2

Figure 7: Improved segmented images

3 Bonus Task: Implementation in ROS

In this section, the clustering method in Task1 is implemented in ROS. By subscribing the topic /me5413/lidar_top, we can process the data and publish the clustered point cloud and the results are saved as in rosbag. We also find the initial data has 382 frames and is published at a rate of 20Hz, so some methods are used for accelerating the clustering algorithm.

We test our self-written RANCAC algorithm and find its segmentation performance is far from perfect and cost a long time. Therefore, we use the *segment_plane* function in open3d and set the parameters $distance\_threshold = 0.3, ransac\_n = 3$ and $num\_iterations = 1000$. Moreover, we use voxel down-sampling method (*voxel_down_sample* function) and successfully reduce by half of the initial points for clustering. However, the DBSCAN algorithm still needs about 0.35s and cannot process all frame. Therefore, we decide to realize real-time processing by dropping 8 out of every 9 frames. Though it's meaningless to play the initial rosbag slowly in real case, we offer the offline processing case and save the result to "BonusTask_rosbag_offlineProcess.bag".

The initial lidar data and clustered results are visualize in Rviz (Fig. 8).

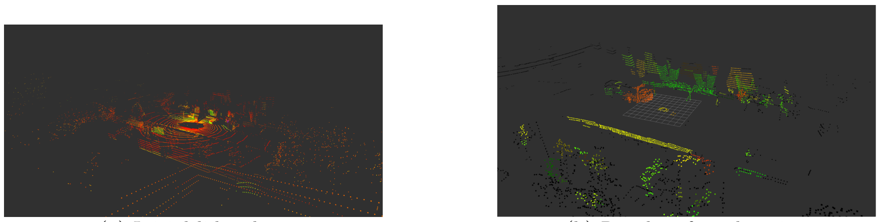

(a) Initial lidar data (b) Results after clutering

Figure 8: Initial and clustered point cloud visualized by Rviz