REDBACK NMP 2021 T3 – AV PERCEPTION

by DARYL LEE

1. Downsampling

Downsampling involves the generation of a standard grid within a point cloud. Points within a particular grid space are collected into a singular voxel to minimise the points in the point cloud. With a lower voxel size, the following operations will take longer, while a larger voxel size may not provide enough resolution for an autonomous vehicle to safely navigate.

Multiple voxel sizes were trialed until a suitable balance was found. A voxel size of 0.2 was used, although voxels up to 0.5 in size also produced satisfactory resolution.

2. Crop/FOV trimming

The first process was to remove any statistical outliers. This process compared the density around a voxel to the global mean, and removed the least dense clusters. The two variables that determined the extent of outlier removal were neighbours and standard deviation.

An aggressive outlier trimming was applied to also remove the clusters on the edge of the LIDAR scan, which displayed low point density and is redundant to the autonomous vehicle. A standard deviation of 0.1 was applied and a nb_neighbours of 200 were used to have a decent sample size.

I attempted to crop the point cloud to remove unwanted clusters that were not in the immediate view of the vehicle. However, this turned out to be difficult to execute as the Open3D library does not provide a scale or coordinate system visible to us. Thus, it was decided that the outlier removal was sufficient enough in removing redundant clusters.

3. Segmentation

Segmentation using the RANSAC Algorithm was carried out and made up the majority of the processing time. The variables that affected the quality of the process were the threshold distance that determined the voxels that fit the plane (distance_threshold), the minimum number of data points (ransac_n) required to estimate model parameters, and the number of iterations run by the algorithm (num_iterations).

The threshold distance was interpreted to be the “thickness” of the plane segment, which was kept to 0.4 to prevent any road obstacles from being segmented out. Meanwhile, it was more difficult to balance ransac_n and num_iterations. Ideally, the number of iterations should be kept to a minimal to avoid lengthy process times. It was found that a combination of a low ransac_n and above 1000 iterations were suitable for most point clouds. However, the process sometimes suffered in parts of the plane close to dense clusters, such as other vehicles.

Due to this, an iteration count of 1500 was used with a minimum ransac_n of 3 to ensure greater accuracy. I prioritised this as an unsegmented section of the plane can cause the autonomous vehicle to wrongly identify it as an obstacle. The vehicle may swerve to avoid it unnecessarily, potentially being dangerous to surrounding vehicles and pedestrians.

4. Clustering

The DBSCAN algorithm was used to separate the point cloud into clusters. This process depended on two variables, the range from a voxel to measure from (eps) and the minimum number of points in a cluster (min_points). From this, it can be assumed that eps has to be bigger than the voxel size for the DBSCAN algorithm to correctly identify dense areas. The minimum number of points can be set to an arbitrary number as most low-density clusters would have already been detected and removed as outliers.

After testing, it seemed that an eps of 1.5 and a min_points of 3 worked well in differentiating the different obstacles present in the LIDAR scan, although a higher eps can be used as the real-life distances between obstacles are relatively large.

Conclusion and Reflection

The resultant visualisation of the 3D radar data showed distinct clusters that can be easily interpreted as cars, buildings, and other street obstacles. This was also consistent across multiple datasets.

I attempted to optimise the processing time of the perception program as much as possible. A low processing time is important to autonomous vehicles as a large amount of data has to be processed in real time to safely navigate an area.

The entire process ranged between 0.3 and 0.6 seconds to complete for each dataset, which is not ideal for real time perception. Each process was also timed using python's timeit module. As mentioned earlier, the RANSAC Segmentation took up majority of the processing time while leaving some of the plane unsegmented in some cases. This can be improved upon with potentially a more efficient segmentation algorithm or a better combination of input variables.

Future iterations of AV perception programs could use a compiled language such as C++ to further reduce processing time per dataset. In addition, adaptive voxel sizes and processing techniques can also be looked into to produce a more robust perception system.

The plane equation found may also be used to crop the point cloud and remove any points below the road plane. This would reinforce the segmentation algorithm in making sure it is completely removed.