# Individual Lab Report - 10

Team D: CuBi

Nithin Subbiah Meganathan

Teammates:

Bobby

Laavanye Bahl

Paulo Camasmie

Jorge Anton Garcia

November 7th, 2019

Carnegie Mellon University

## Individual Progress

This progress review was focused on implementing one possible pipeline for the obstacle detection subsystem. The goal of the obstacle detection pipeline is to detect any object in the camera scene range and attach a bounding box over it. The object detection pipeline then classifies the toys as objects that can be picked. Couple of approaches to obstacle detection were tried, ranging from point cloud detection to deep learning methods.

The point cloud detection approach is as follows. The Intel RealSense ROS wrapper is used to publish the point cloud data of the scene. This is the raw point cloud data which needs to be processed. The processing and filtering steps are shown in the figure 1. This is achieved using Point Cloud Library (PCL). The ROS nodes were written in C++ for fast processing in real time.

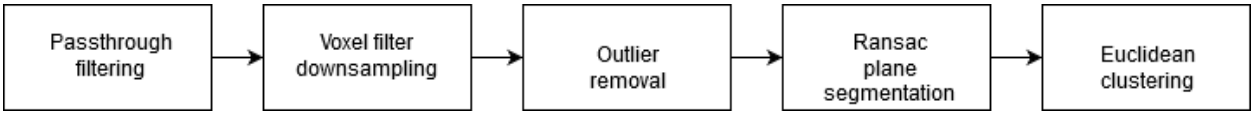


*Figure 1. Point Cloud Data Processing Steps*

The passthrough filter is used to filter along a specified dimension (x, y, and/or z) – that is, cut off values that are either inside or outside a given range. Voxel filter is done to down sample the dense input point cloud. The down sampling is done by taking a cuboid of user specified dimension in space and calculating the centroid of point clouds within the cuboid. This acts as a good approximation of the dense input which is expensive to process. Next, outliers are removed using Statistical Outlier Removal filter. This is performed by calculating the distance of neighboring points. A Gaussian distribution is then fit for the distances and a limit is set outside which the points are outliers.

Plane segmentation is done through RANSAC in which random points are selected, plane formed using those points, and inliers are calculated. The best estimate of a plane is the one with most inliers and it is removed. Now the processed point cloud is supposed to have only the obstacles. Clustering of this point cloud to determine each obstacle is done using Euclidean Clustering.

However, this implementation of obstacle detection did not work. The output point cloud was too sparse and thus, no obstacle could be observed. There are too many parameters involved and even after a considerable amount of tuning them it did not work.

I went on to the Deep Learning approach [1] that I had described in my previous ILR. In that approach a single 2D image is taken as an input and a 3D bounding box is

regressed from the network as output. But after going through the implementation it was found that it worked only for specific classes of objects pertaining to the self-driving car industry, for example cars, pedestrians.

All the other Deep Learning networks like YOLO, SSD work only for 2D object detection. In that case the depth information is lost. One way forward was to use the stereo pair image from the RealSense camera, perform a 2D object detection, and use epipolar geometry to map it to the 3D point clouds. But it was beyond the scope of this PR and so we resorted to extending the object detection pipeline for obstacles too.

## Challenges

The biggest challenge during the point cloud implementation was the number of parameter tuning. Each filter had a user-defined parameter and tuning it affects the result in a huge way. Also, it was hard to obtain a general solution using a classical vision approach meaning the implementation had to be overfit to environment subject to the parameters. Environmental factors like lighting, floor color can break the algorithm resulting in sparse point clouds. Deep Learning techniques overcome a lot of these issues, but time was a constraint to extend the 2D detection to 3D. Had we used a 3D LiDAR, this issue would have been easier to solve owing to the abundant research and literature in that area. Industry and researchers alike tend to use LiDAR or fusion of multiple sensors for 3D detection.

## Teamwork

Jorge and I discussed about how once obstacles are detected and its bounding boxes estimated, its dimensions can be used to map the obstacle like filling an occupancy grid. We also discussed about estimating the 3D point cloud correspondences from a stereo pair image. Laavanye helped me with the point cloud implementation for obstacle as he had done a very similar approach for objects. And finally, he adapted that pipeline for obstacles too. Paulo and I had discussions over the local planning strategy after the traversable cells are obtained.

## Plans

Next step for me would be to align CuBi with respect to the container while dropping. This is to be done using AprilTag. CuBi needs to orient itself whenever it goes to dropping the toy in the box. For the team, integration of all sub systems is the critical aspect. Integration of exploration with planning and finally object picking is a huge task ahead of us and is going to get us very close to the final goal of our project.

## References


1. “3D Bounding Box Estimation Using Deep Learning and Geometry” by Mousavian et al.