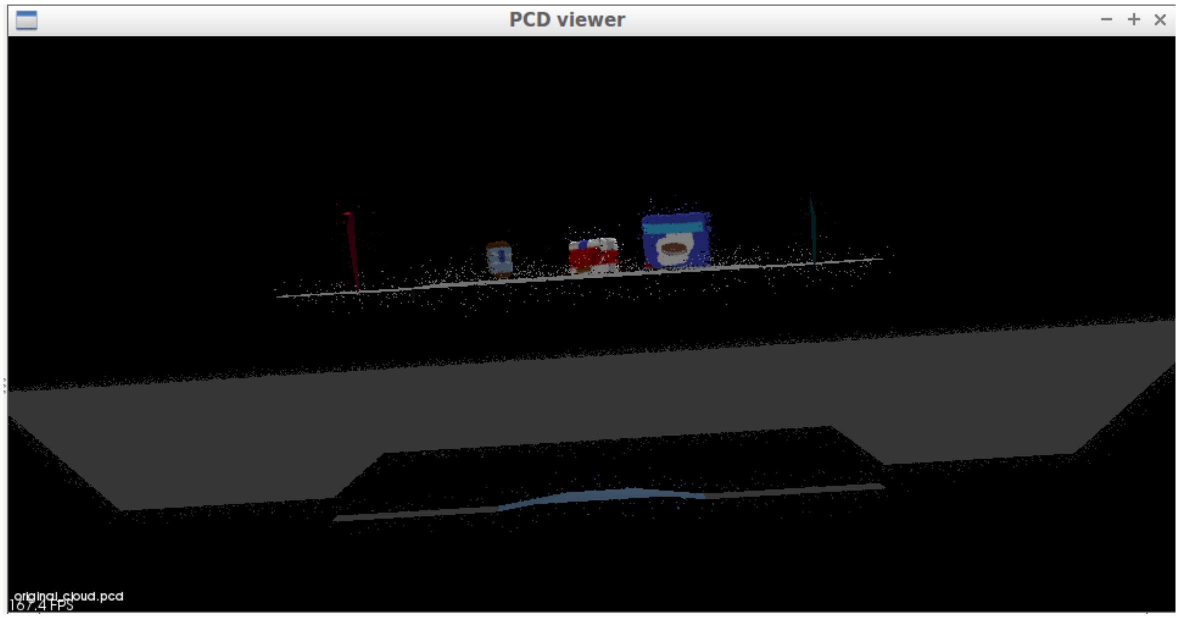
RoboND Perception Project

Exercise 1, 2 and 3 pipeline implemented

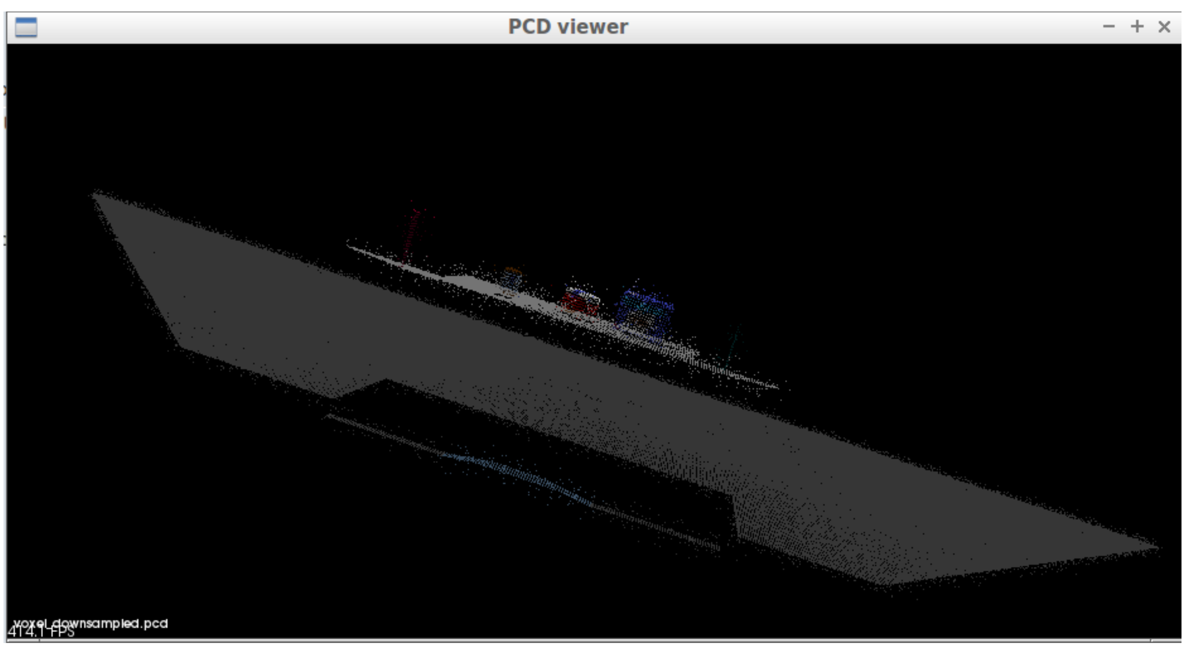
1. Complete Exercise 1 steps. Pipeline for filtering and RANSAC plane fitting implemented.

To complete exercise 1 the following steps were followed:

1) The original point cloud scene was viewed.

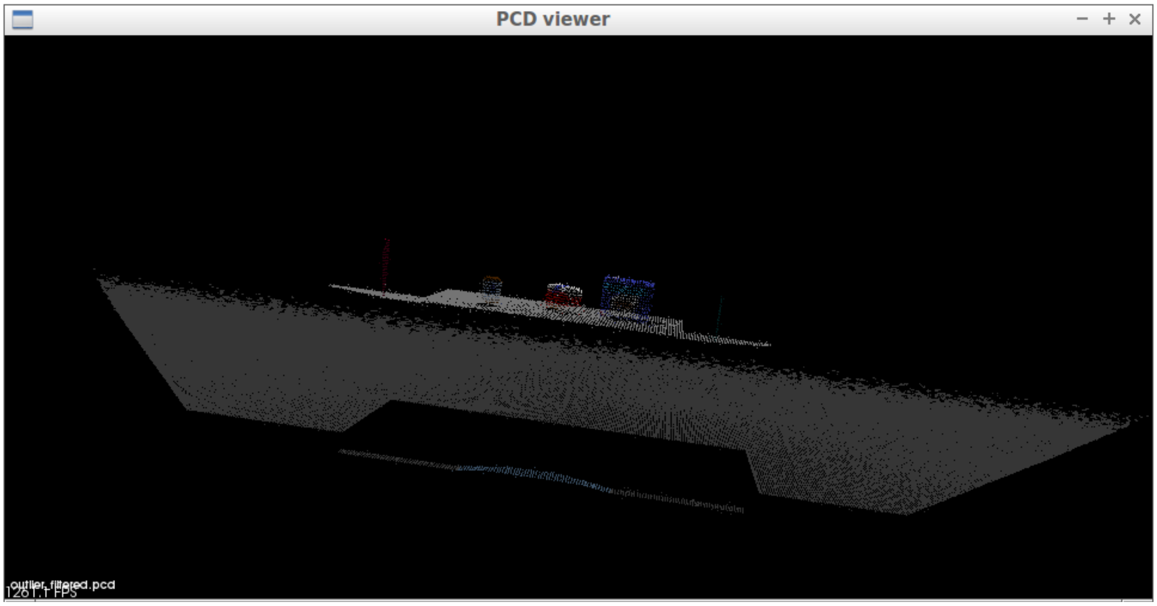


2) A Voxel Grid Downsampling filter was implemented to lower the original RGB-D camera data resolution so that the processing of this data can be faster and remain efficient. The Leaf size was chosen to be 0.01.

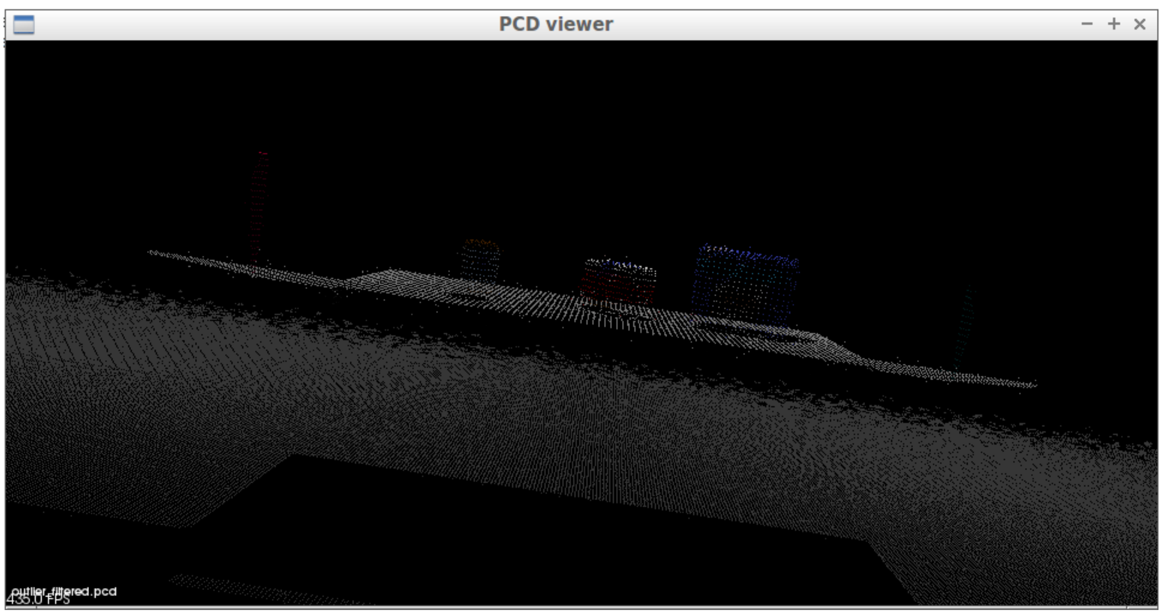


3) An outlier Filter was used to remove noise from the scene.

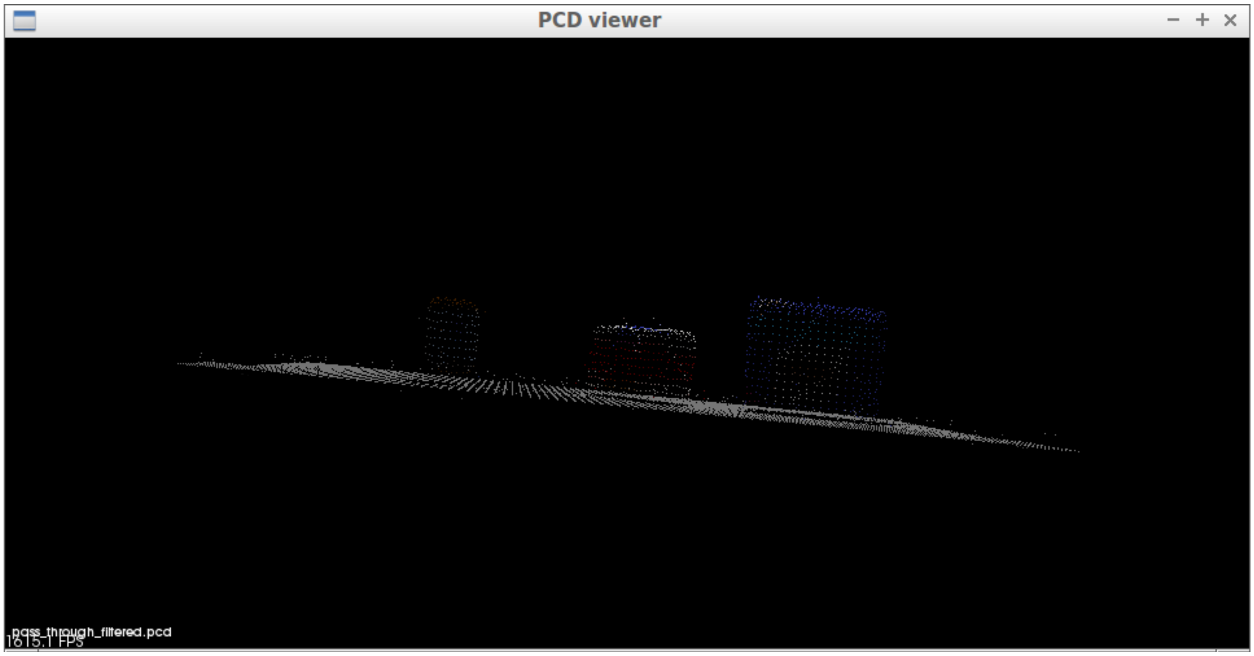
The number of neighboring points to analyze for any given point was set to 20 and the threshold scale factor was set to 0.1.

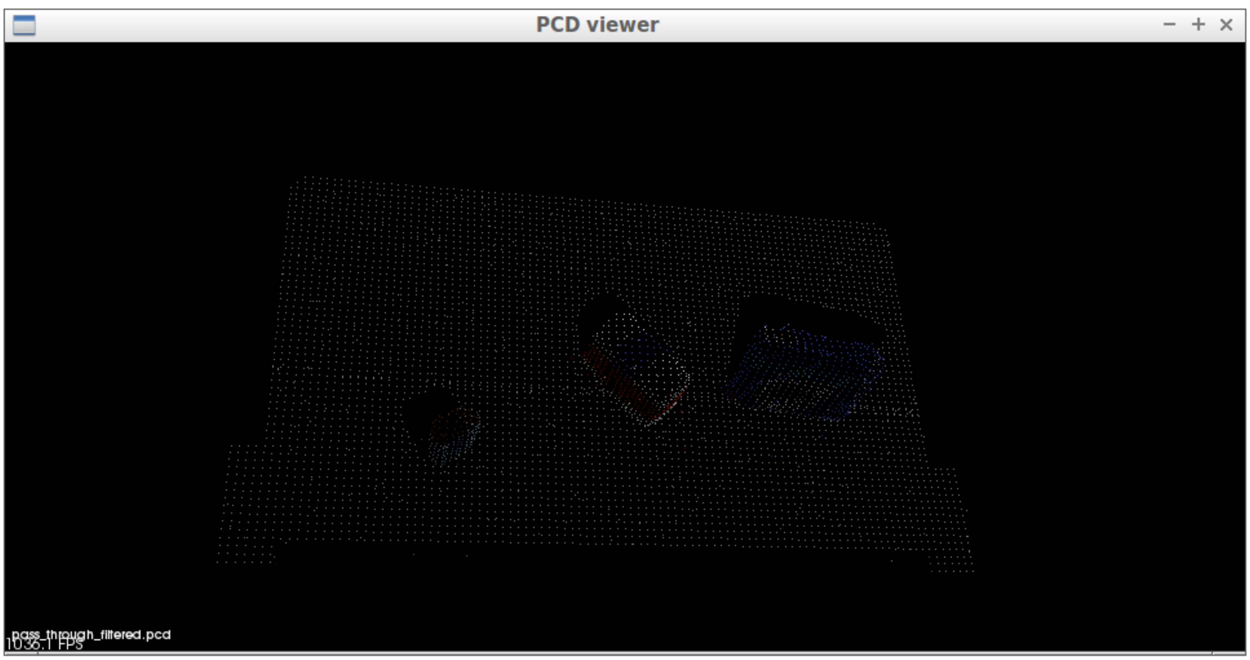


These snapshots from the PCD viewer after implementing the outlier filter show how much noise was reduced compared to the original cloud.



4) A pass-through filter was used remove unneeded data from the point cloud, leaving only the table and the objects above it.





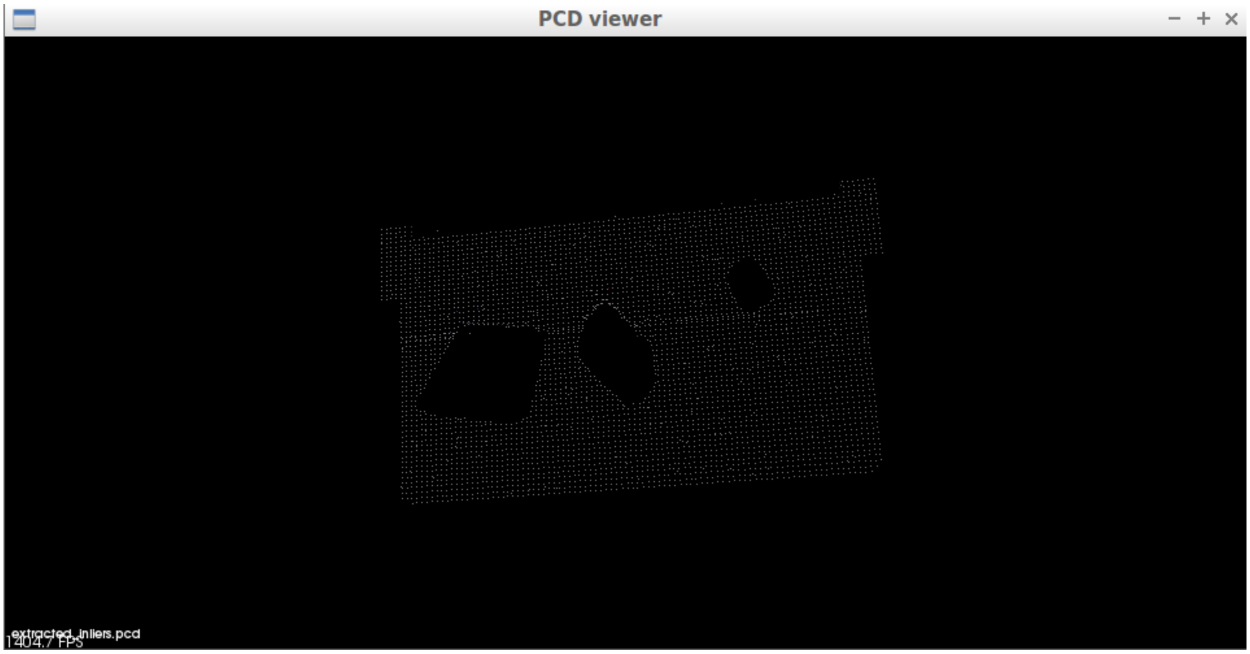
The axes were assigned and ranged as the following to the passthrough filter object:

filter_axis = 'x'	filter_axis = 'z'	filter_axis = 'y'
axis_min = 0.3	axis_min = 0.6	axis_min = -0.5
axis_max = 1.0	axis_max = 1.3	axis_max = 0.5

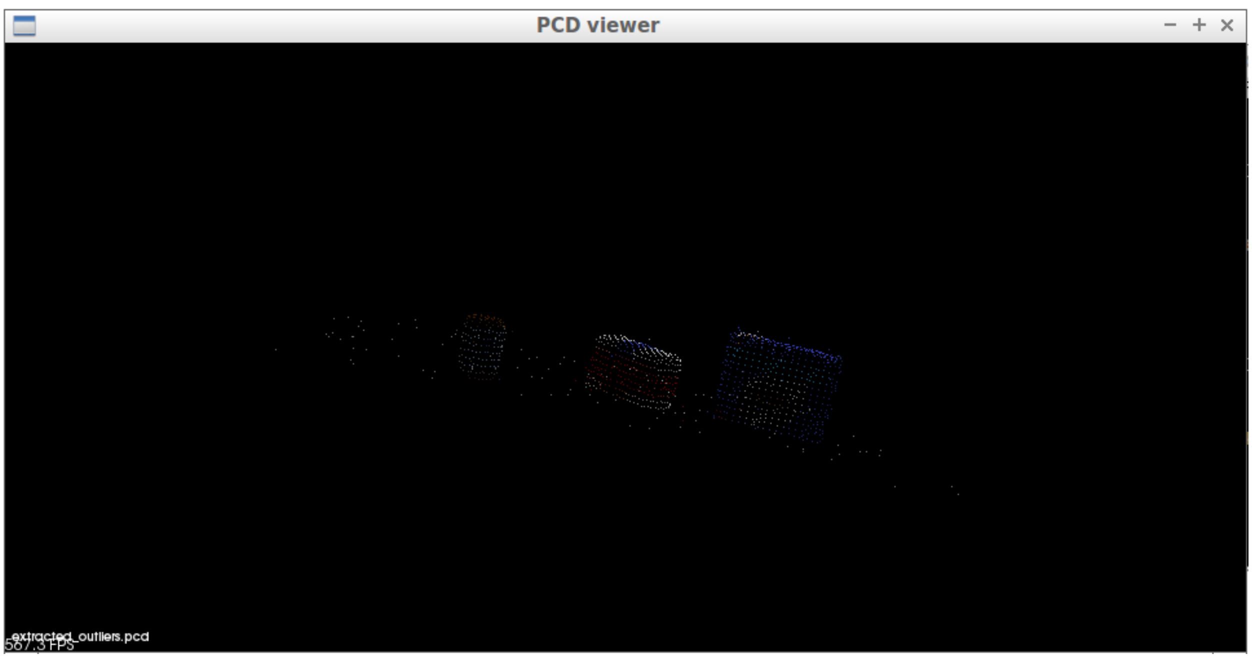
5) RANSAC Plane Segmentation was implemented to segment the objects and the table separately from the point cloud scene.

This is the PCD of the extracted_inliers from the RANSAC Segmentation (table).

The max distance for a point to be considered fitting the model was set to 0.01.



And this is the PCD of the extracted_outliers from the RANSAC Segmentation (objects)



2. Complete Exercise 2 steps: Pipeline including clustering for segmentation implemented.

Euclidean Clustering was used to segment objects from each other. The tolerances for distance threshold as well as minimum and maximum cluster size (in points) were set as the following:

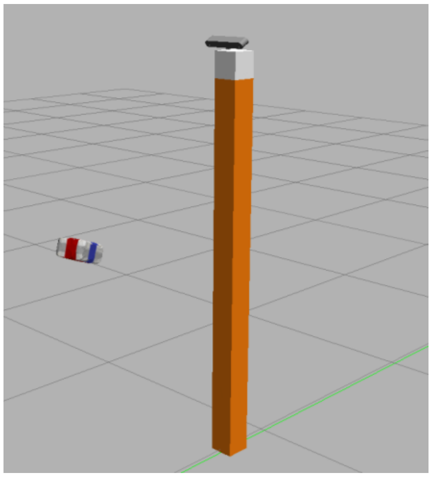
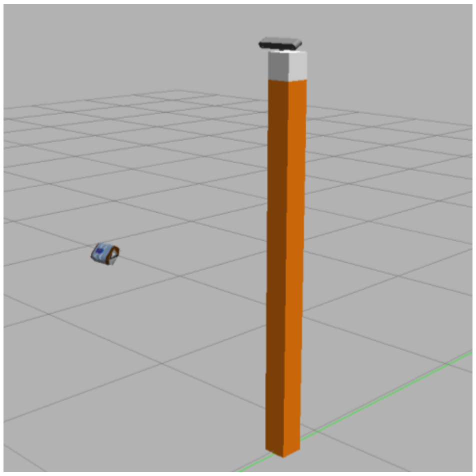
```
ec.set_ClusterTolerance(0.01)
ec.set_MinClusterSize(30)
ec.set_MaxClusterSize(1500)
```

These are the segmented objects from test one, each object is represented by a different color.

3. Complete Exercise 3 Steps. Features extracted and SVM trained. Object recognition implemented.

Both compute_color_histograms() and compute_normal_histograms() functions have been filled out and SVM has been trained using train_svm.py.

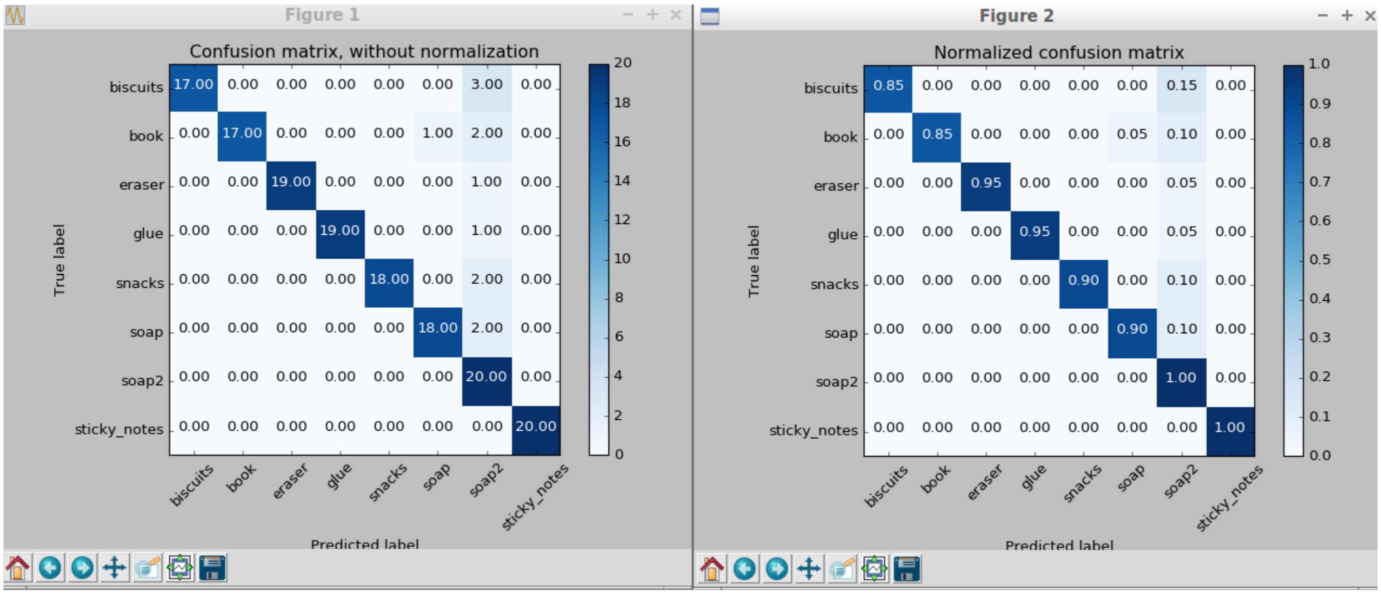
First features were captured using capture_features.py. Features were captured in the sensor_stick simulator for ['biscuits', 'soap', 'soap2', 'book', 'glue', 'sticky_notes', 'snacks', 'eraser'] model names with 20 sample of each captures.

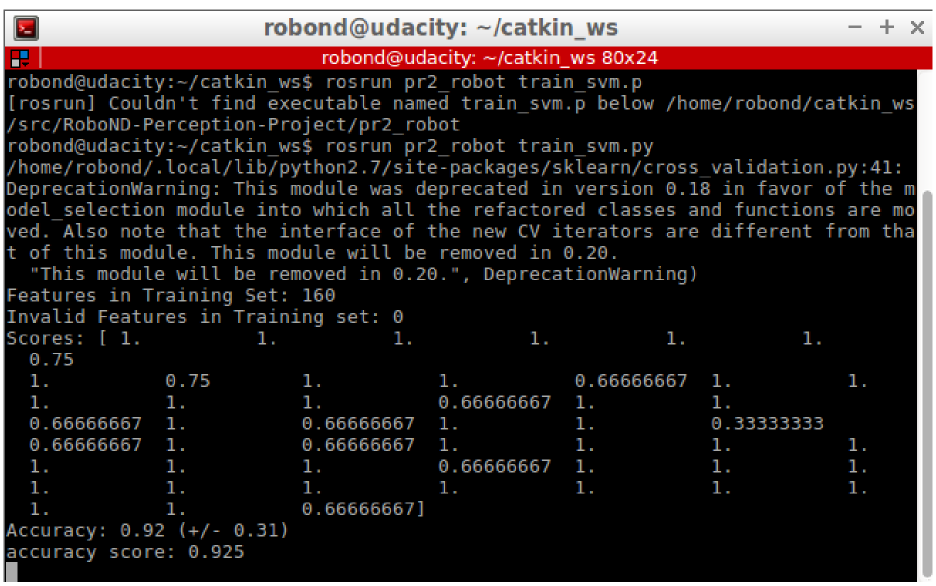
Then train_svm.py was used using the following classifier:

clf = svm.SVC(kernel='linear', C=0.1)

This is the generated confusion matrix from the svm.



The svm gave accuracy of 0.92 (+/- 0.31) which is very great.



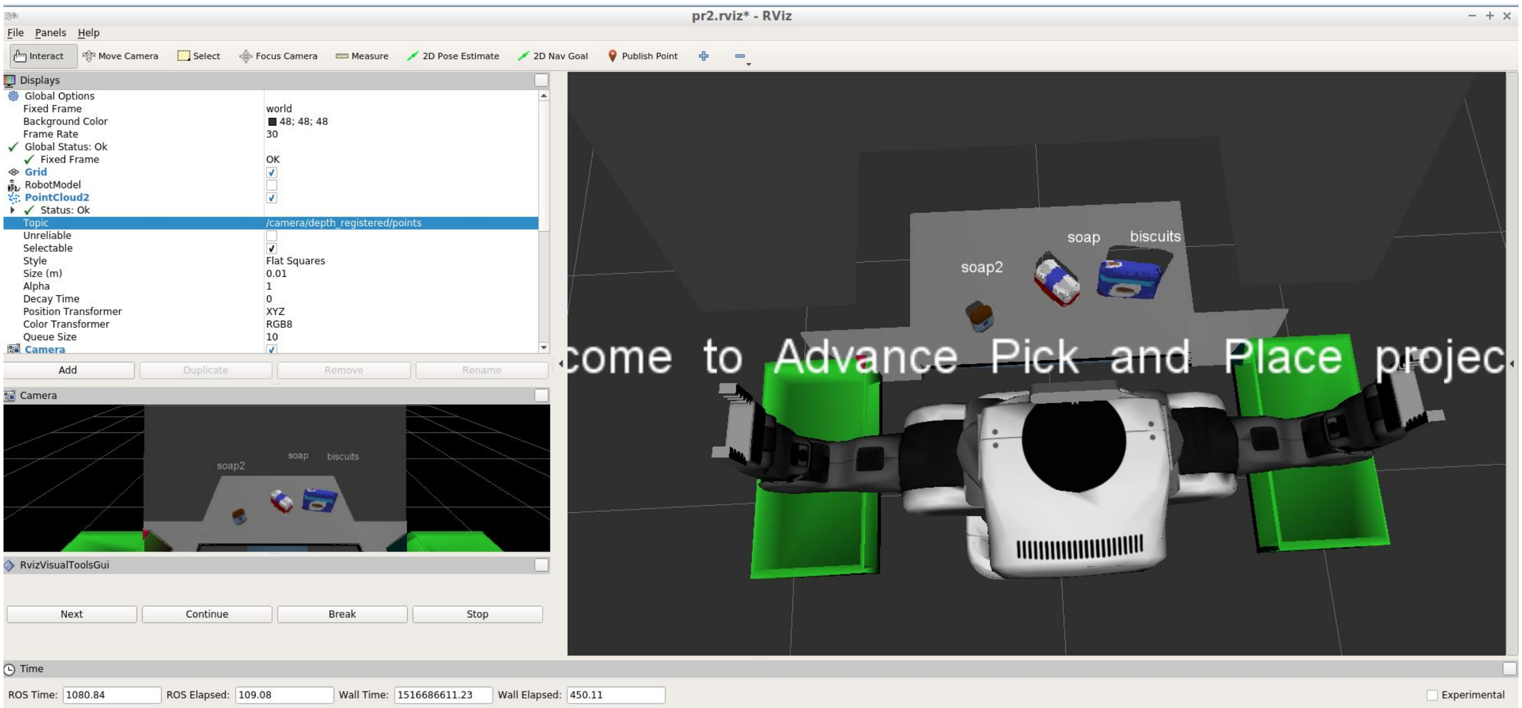
Pick and Place Setup

1. For all three tabletop setups (`test*.world`), perform object recognition, then read in respective pick list (`pick_list_*.yaml`). Next construct the messages that would comprise a valid `PickPlace` request output them to `.yaml` format.

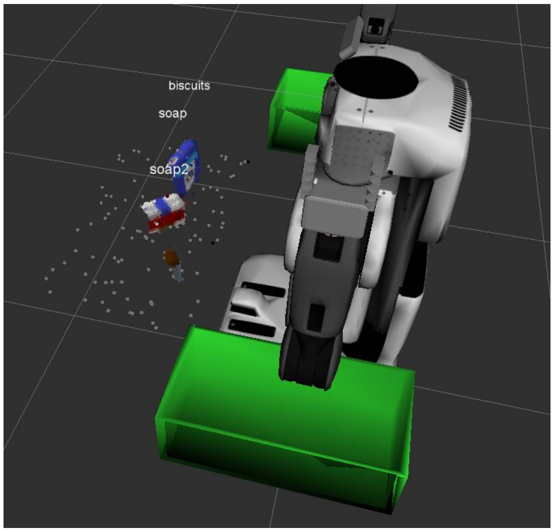
The added functionality was added to the already existing ros node that communicates with your perception pipeline to perform sequential object recognition. Also the PickPlace requests are saved into output_1.yaml, output_2.yaml, and output_3.yaml for each scene respectively. All these files are attached in this project submission.

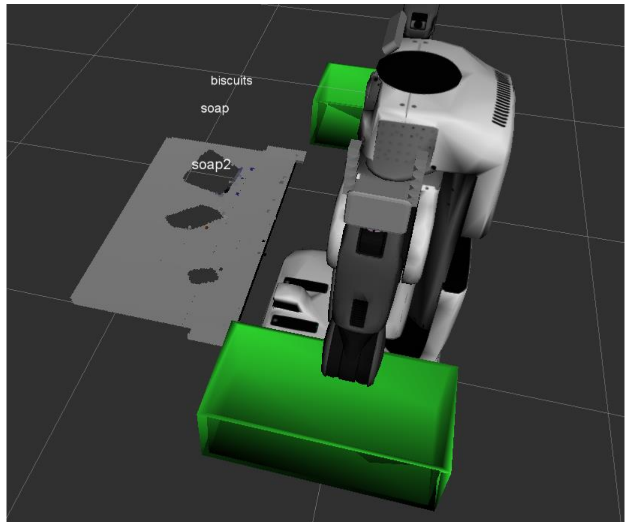
These are the test results showing label markers in RViz that demonstrate the recognition success rate in each of the three scenarios:

- Test 1 Results:

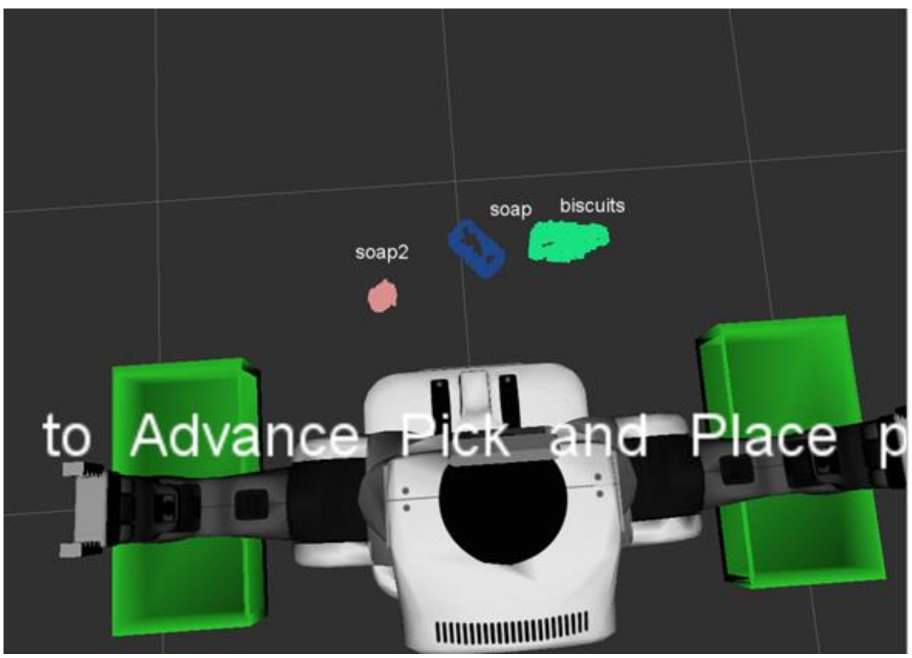


Test one results give a succession rate of 100% (3/3) this shows the segmented objects from the table.

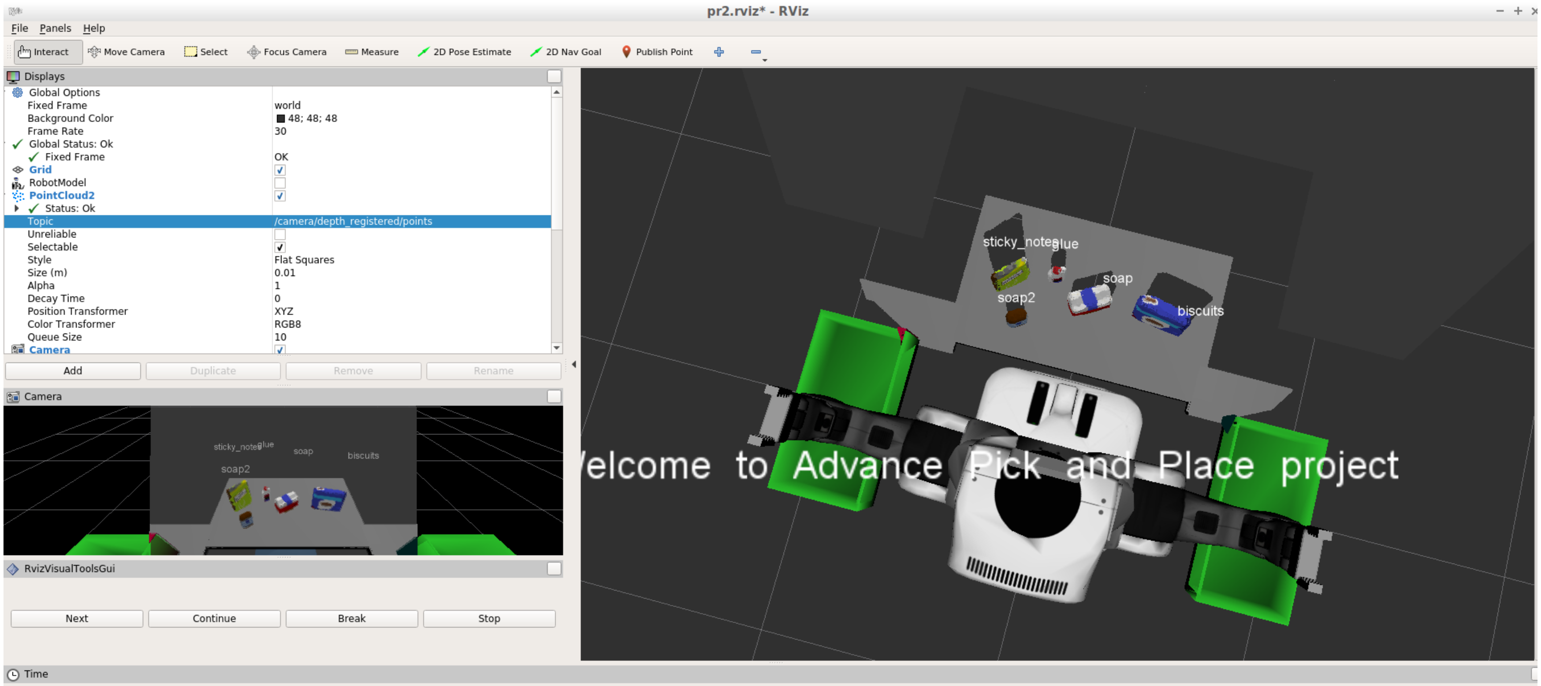




These are the segmented objects using Euclidean Clustering.

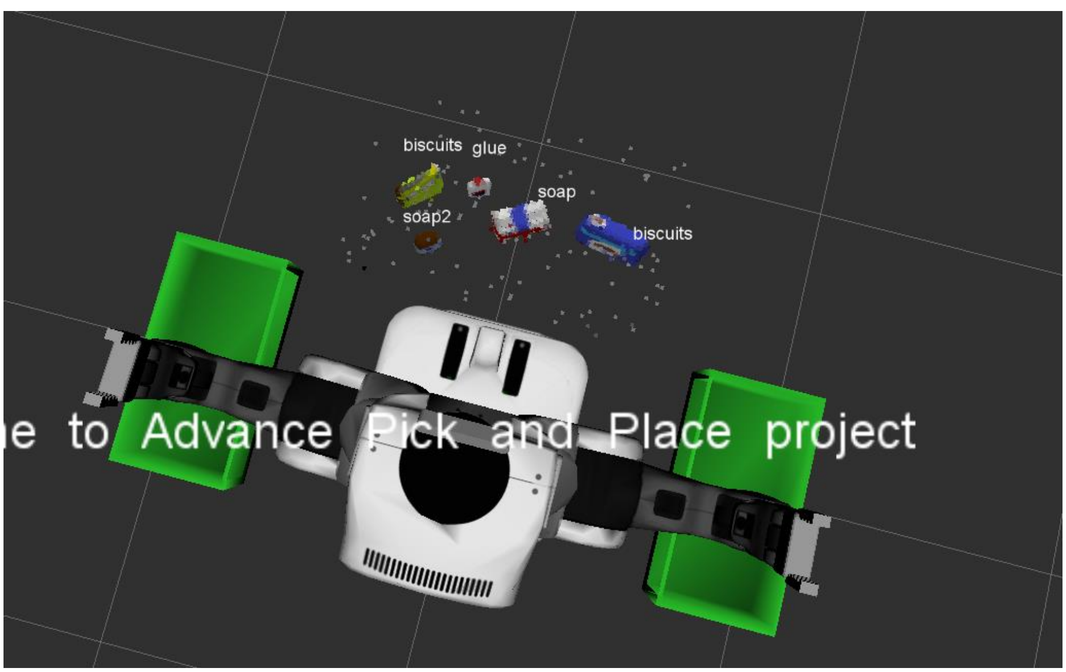


- Test 2 Results

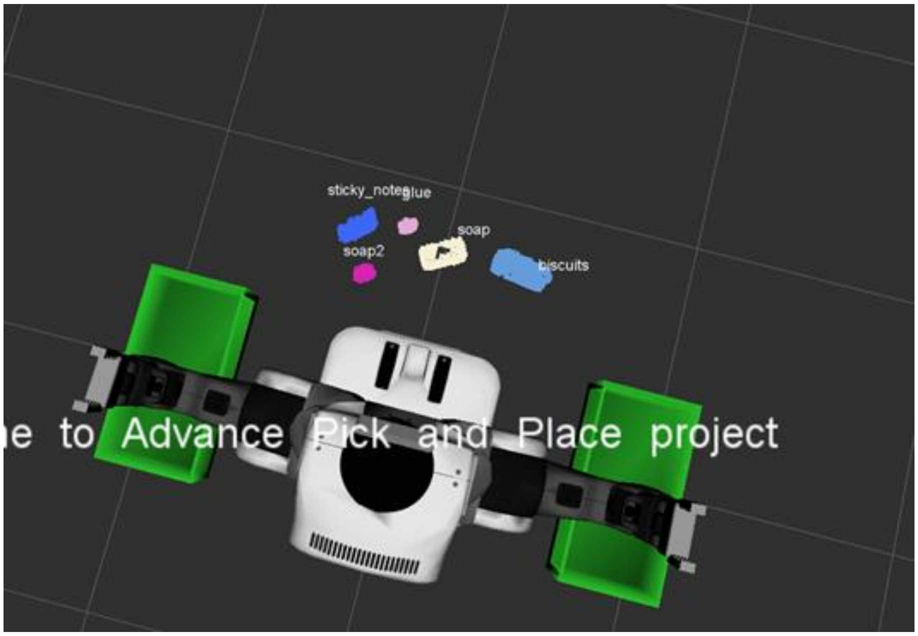


Test 2 results give a succession rate of 80% (4/5).

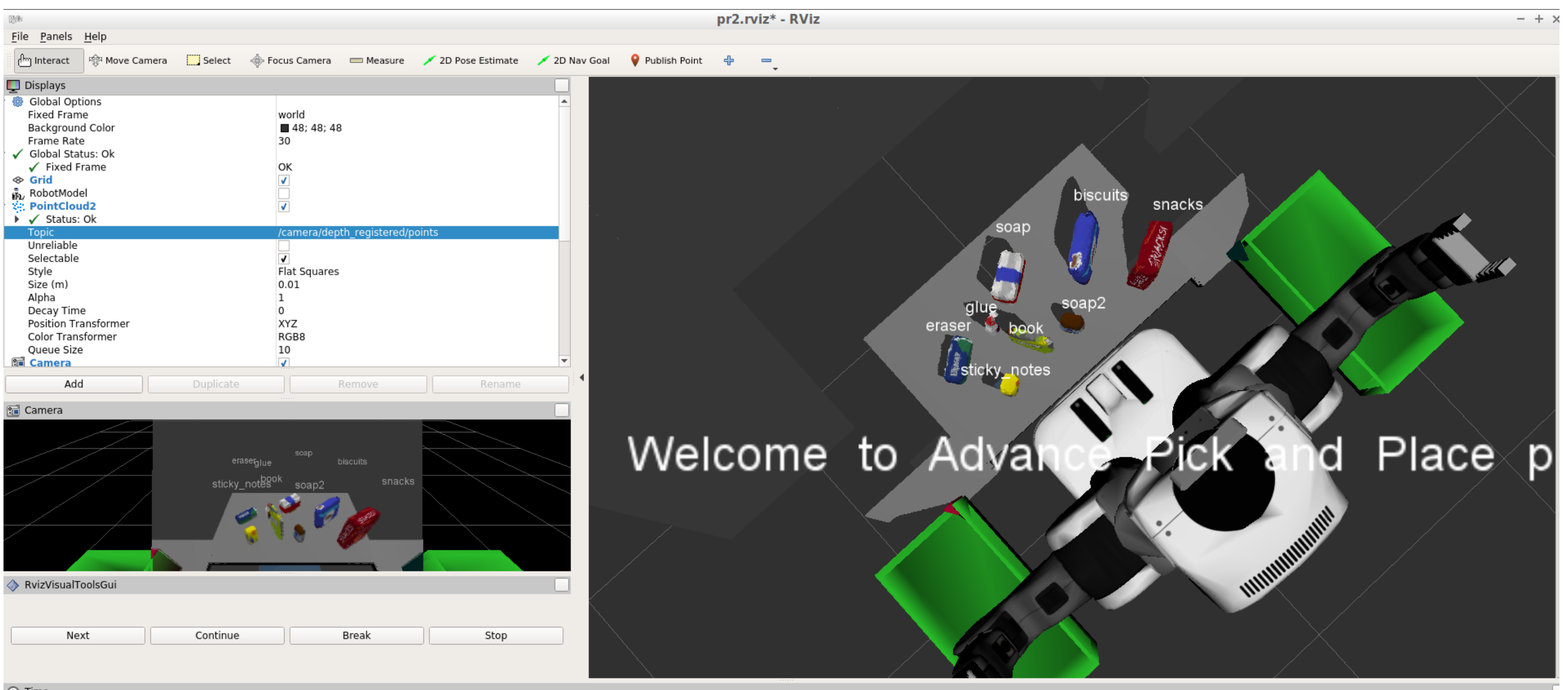
These are the extracted objects.



And these are the segmented objects using Euclidean Clustering.

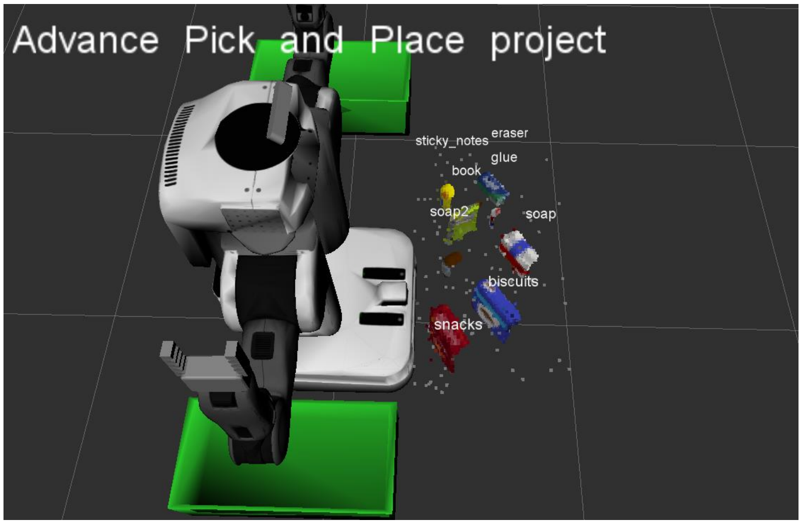


- Test 3 Results

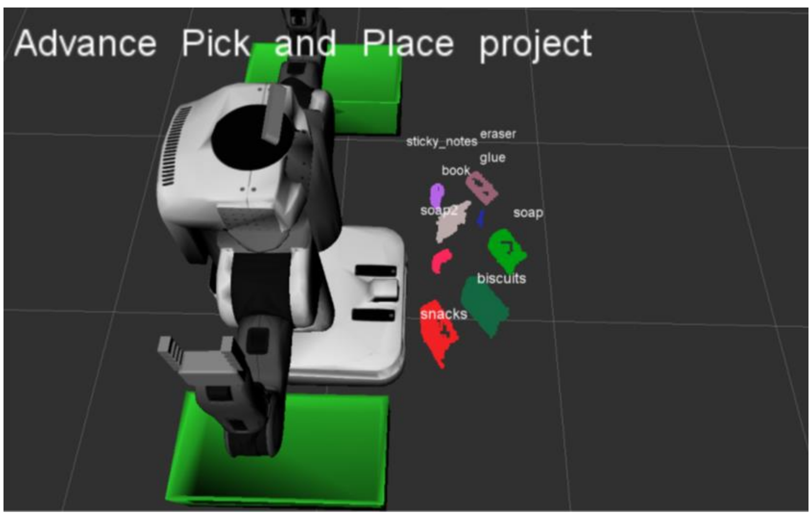


Test 3 results give a succession rate of 100% (8/8).

These are the extracted objects:



And these are the segmented objects using Euclidean Clustering.



To enhance the recognition of objects, the number of feature extraction spawns were increased to 20 times. Also HSV was used in calling compute_color_histograms() in capture_features.py. Also, a linear Kernels from sklearn.svm.SVC was used for the svm classifier which gave accurate training results of 0.92 (+/- 0.31). Furthermore, the outlier filter helped the recognition and segmentation of objects severely by removing most of the noise from the scene.

I think I could have used better clustering algorithms or could have tried testing with both the "k-means Clustering" and "DBSCAN Clustering" instead of just testing with the RANSAC algorithm. I didn't try them due to time constrains. The RANSAC algorithm may not be the best because the size ranges of the objects differ a lot and so in order for the clustering to be accurate the Min Cluster Size had to be adjusted with almost each scene. In addition, I could have worked more on the actual pick and place of the robot to give better pick and place results. Overall, it was a great project to develop my robotics perception knowledge and skills.