Project: 3D Perception

## Exercise 1, 2 and 3 Pipeline Implemented

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

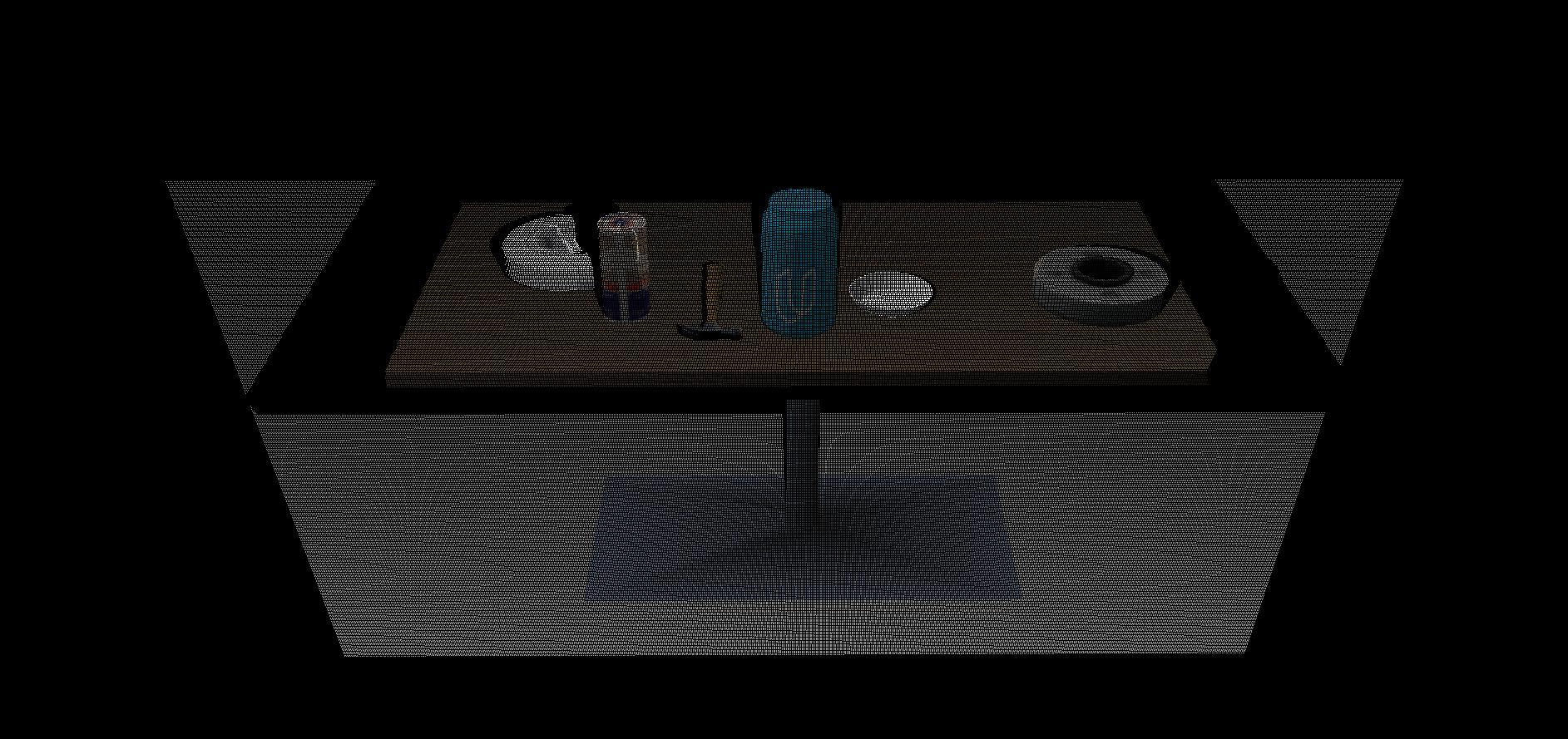
The source code is

https://github.com/nakata0705/RoboND-Perception-Exercises/blob/master/Exercise-1/RANSAC.py

**The original point cloud**

Exercise-1/tabletop.pcd: x y z rgb data. 202,627 points

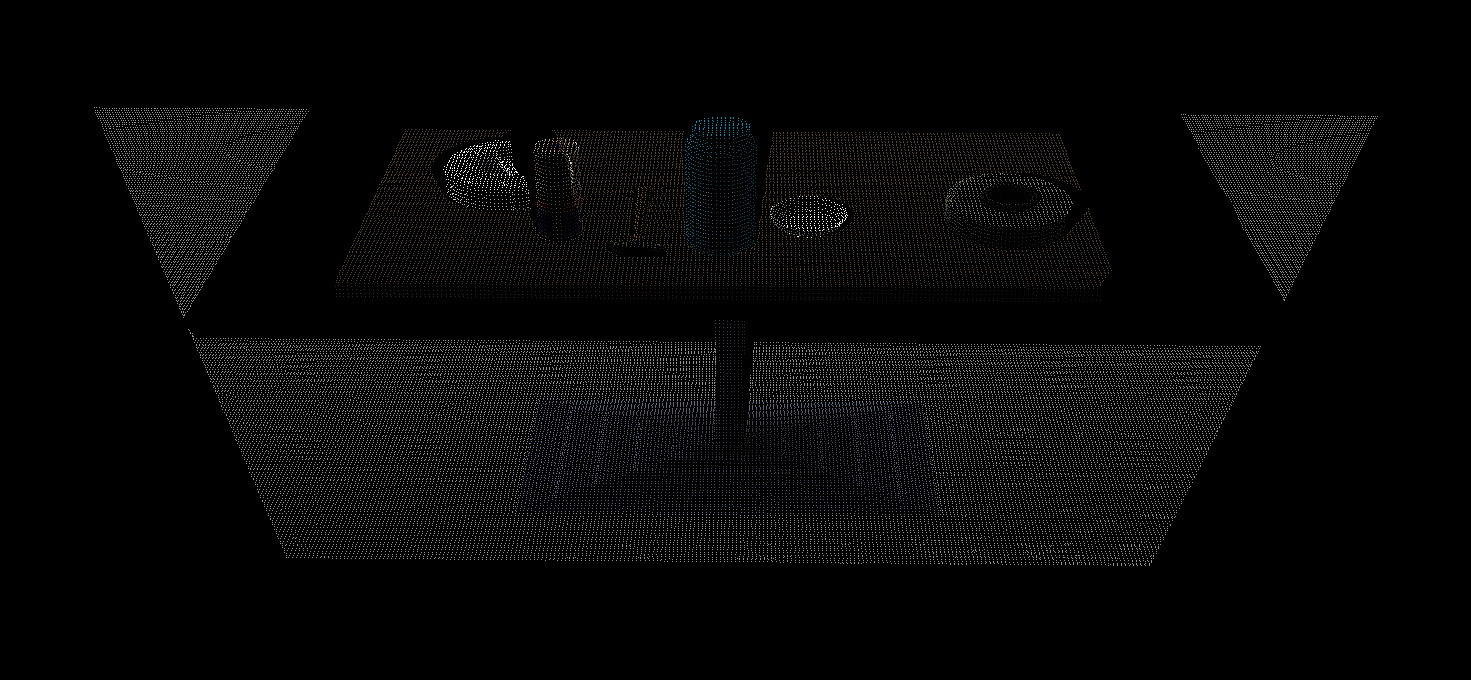
The original point cloud. There are more than necessary points.



**The point cloud after voxel down sampling**

Exercise-1/voxel\_downsampled.pcd: x y z rgb data. 55,336 points

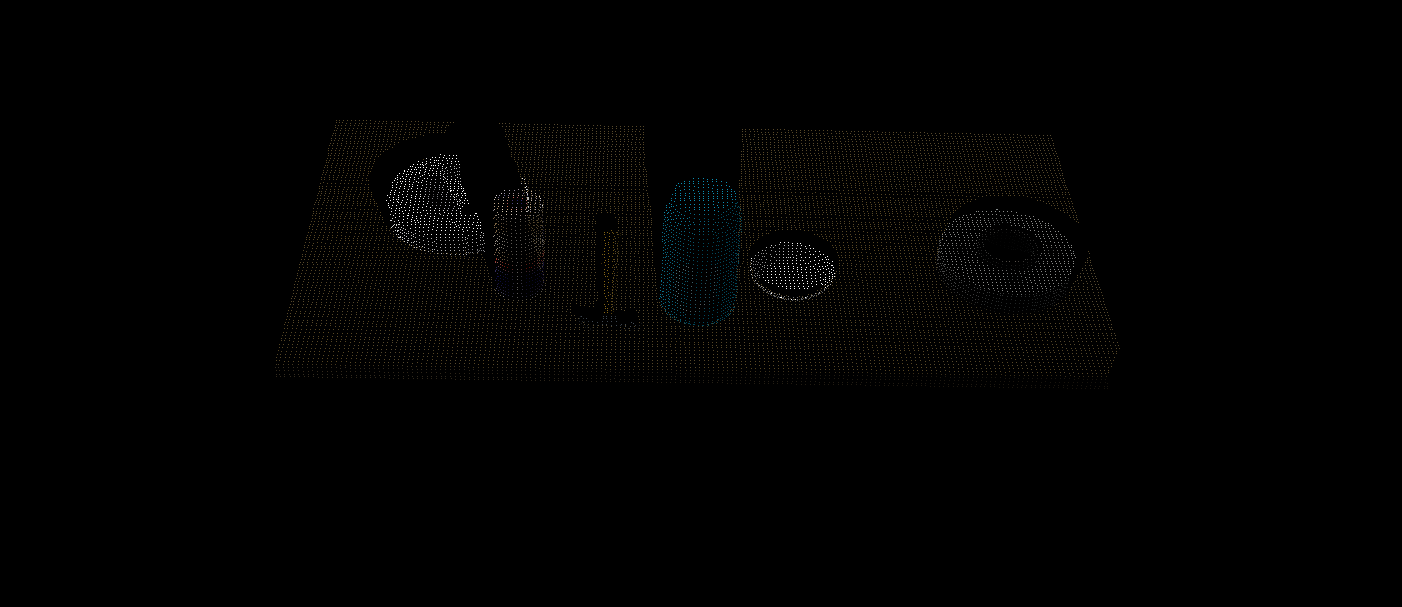
Reduced the number of points by applying voxel filter with 1cm x 1cm x 1cm resolution.



**The point cloud after pass through filter**

Exercise-1/pass\_through\_filtered.pcd: x y z rgb data. 16,302 points

Filtered further by extracting points with height between 0.6m and 1.1m.



**The point cloud after RANSAC filter (inlier for SACMODEL\_PLANE)**

Exercise-1/extracted\_inliers.pcd: x y z rgb data. 11,881 points

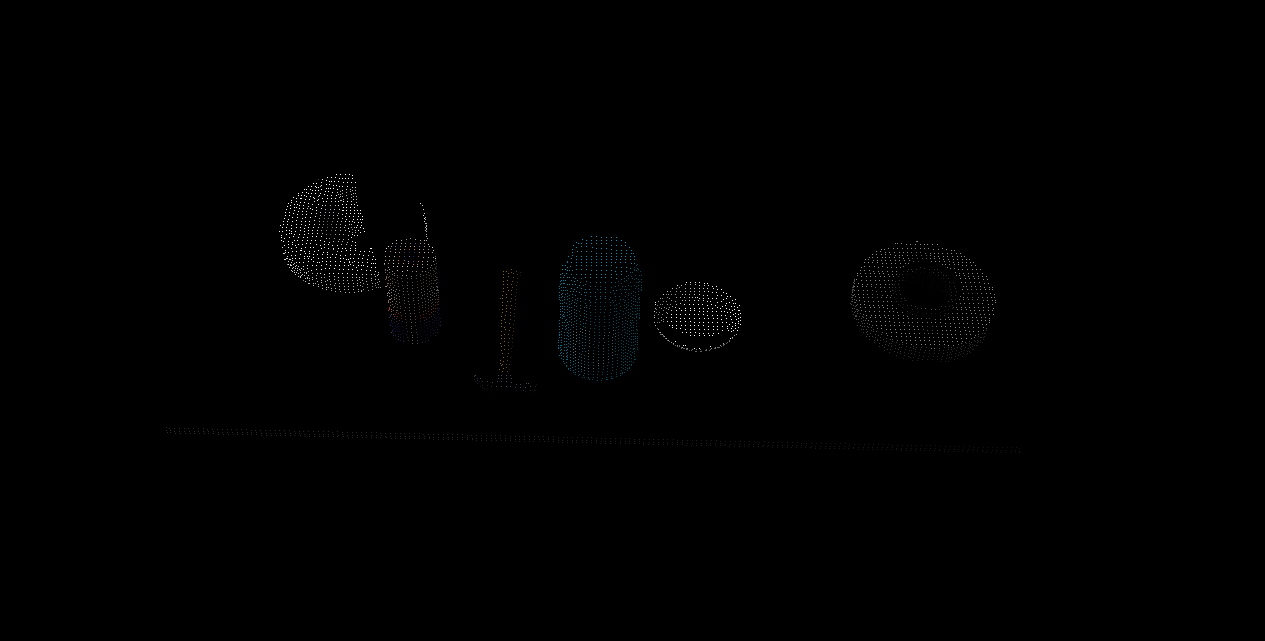
Extracted the largest plane by using RANSAC filter with SACMODEL\_PLANE parameter.



**The point cloud after RANSAC filter (outlier for SACMODEL\_PLANE)**

Exercise-1/extracted\_outliers.pcd: x y z rgb data. 4,421 points

The remaining points. Successfully extracted objects on the table.

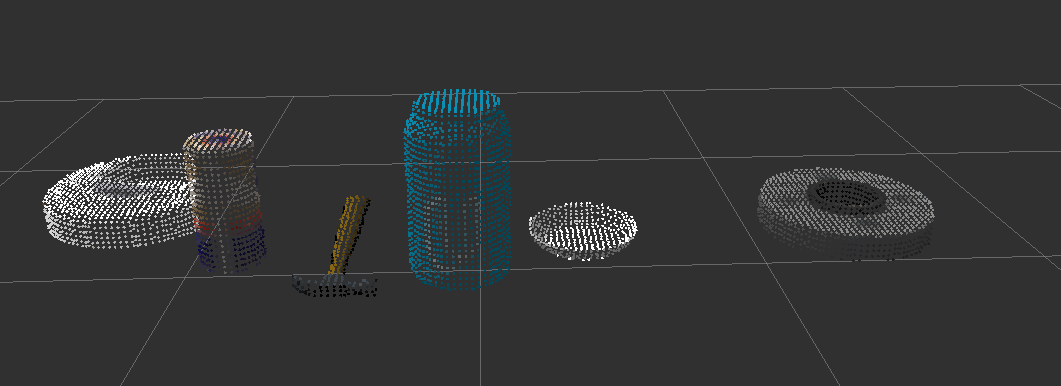


# Complete Exercise 2 steps: Pipeline including clustering for segmentation implemented.

The source code is

https://github.com/nakata0705/RoboND-Perception-Exercises/blob/master/Exercise-2/sensor\_stick/scripts/segmentation.py

**The point cloud after RANSAC filter (outlier for SACMODEL\_PLANE)**



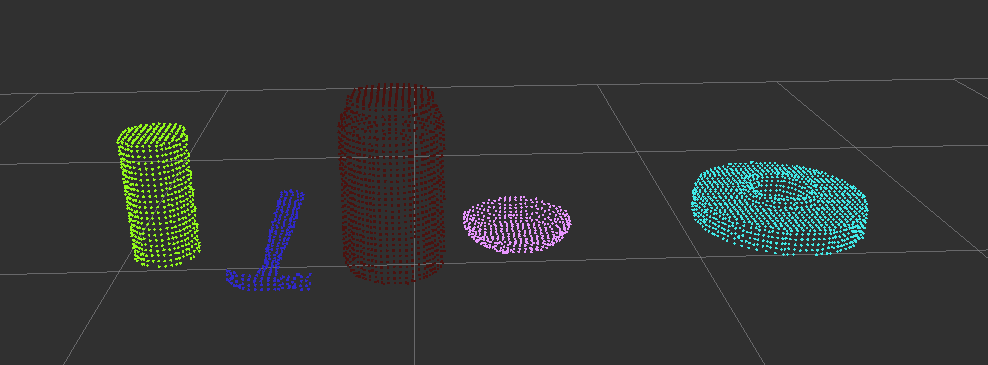
**The point cloud after clustering.**

The parameter for Euclidean clustering is as follows.

ec.set\_ClusterTolerance(0.02)

ec.set\_MinClusterSize(100)

ec.set\_MaxClusterSize(10000)



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

The source code is

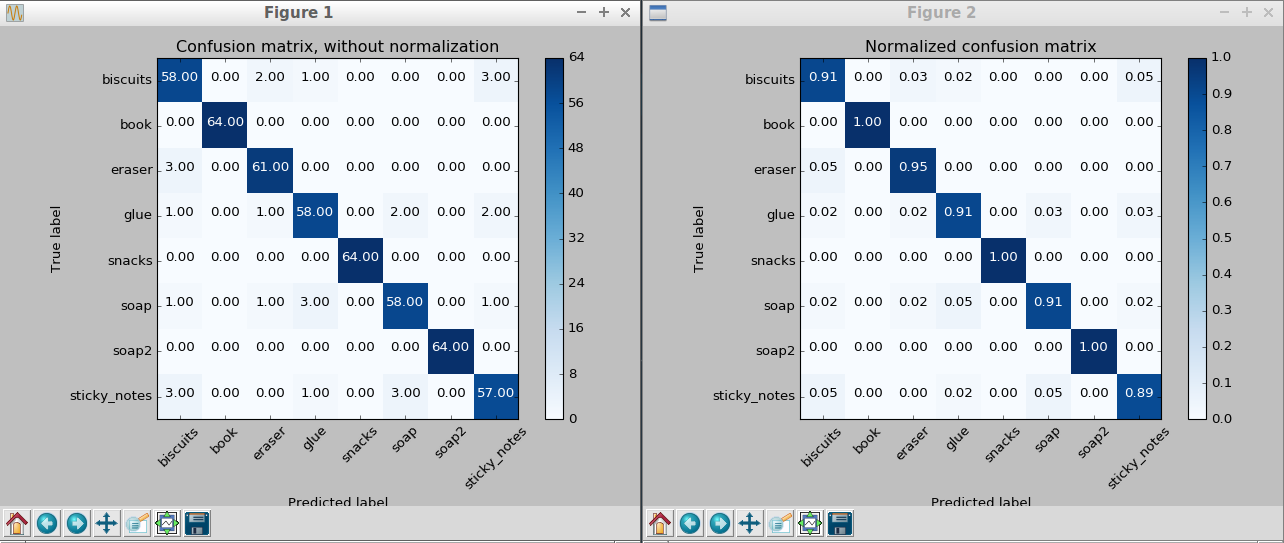
https://github.com/nakata0705/RoboND-Perception-Exercises/blob/master/Exercise-3/sensor\_stick/scripts/capture\_features.py

https://github.com/nakata0705/RoboND-Perception-Exercises/blob/master/Exercise-3/sensor\_stick/scripts/train\_svm.py

https://github.com/nakata0705/RoboND-Perception-Exercises/blob/master/Exercise-3/sensor\_stick/scripts/object\_recognition.py

https://github.com/nakata0705/RoboND-Perception-Exercises/blob/master/Exercise-3/sensor\_stick/src/sensor\_stick/features.py

The confusing matrix is as follows.



Calculation of the feature is very simple. It’s from line 146 to line 149 of object\_recognition.py.

First the program calculates HSV color histograms. The programs uses bins=6 for color histogram and bins=8 for normal histogram. Then the program concatenate the two histograms and use it as feature vector.

# 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 source code is as follows.

https://github.com/nakata0705/RoboND-Perception-Project/blob/master/pr2\_robot/scripts/project\_template.py

The output yaml files are as follows.

https://github.com/nakata0705/RoboND-Perception-Project/blob/master/pr2\_robot/scripts/output\_1.yaml

https://github.com/nakata0705/RoboND-Perception-Project/blob/master/pr2\_robot/scripts/output\_2.yaml

https://github.com/nakata0705/RoboND-Perception-Project/blob/master/pr2\_robot/scripts/output\_3.yaml

The successful recognition rate is 100% for world 1, 100% for world 2, and 87.5% (7/8). The success rate wasn’t stable. It seems those three results are impacted by the noise even after statistical outlier filtering.