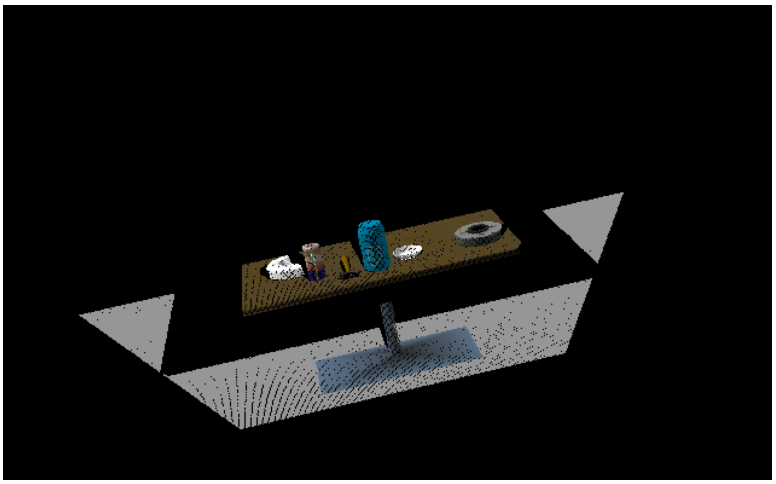
3D Perception Report

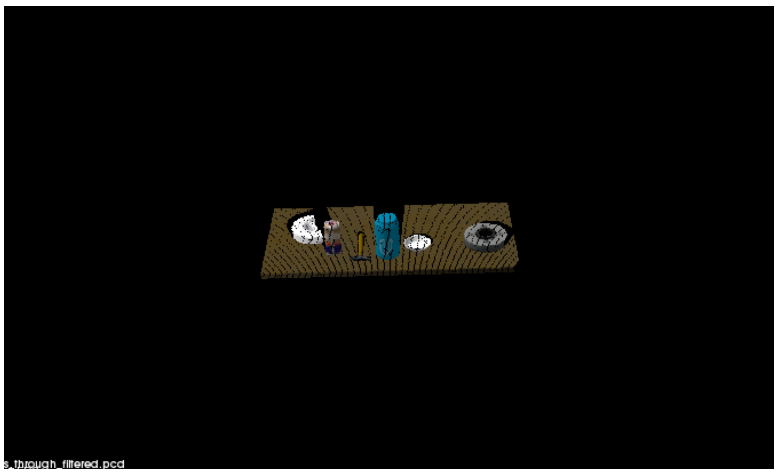
by zhanghanwei

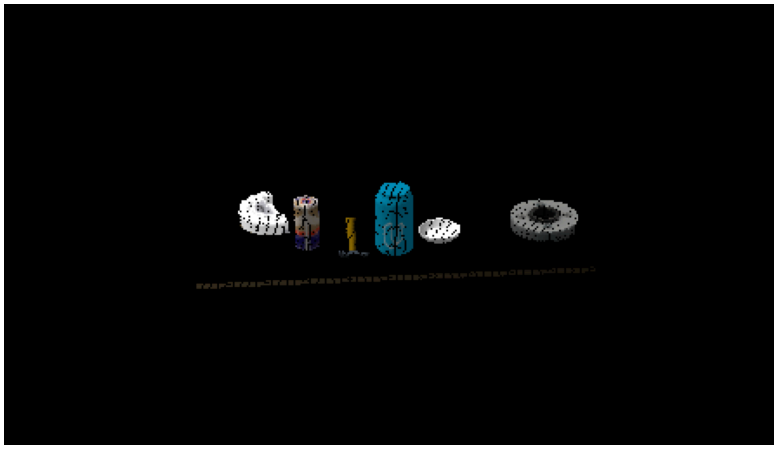
Exercise 1, 2 and 3 pipeline implemented

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

I tested the Voxel Grid Downsampling, the Pass Through Filtering and the RANSAC Plane Fitting with code RANSAC.py and get the result. Then I add the code in pcl_callback() function and modified the parameters to fit the test world.(54~111 lines in project_template.py)

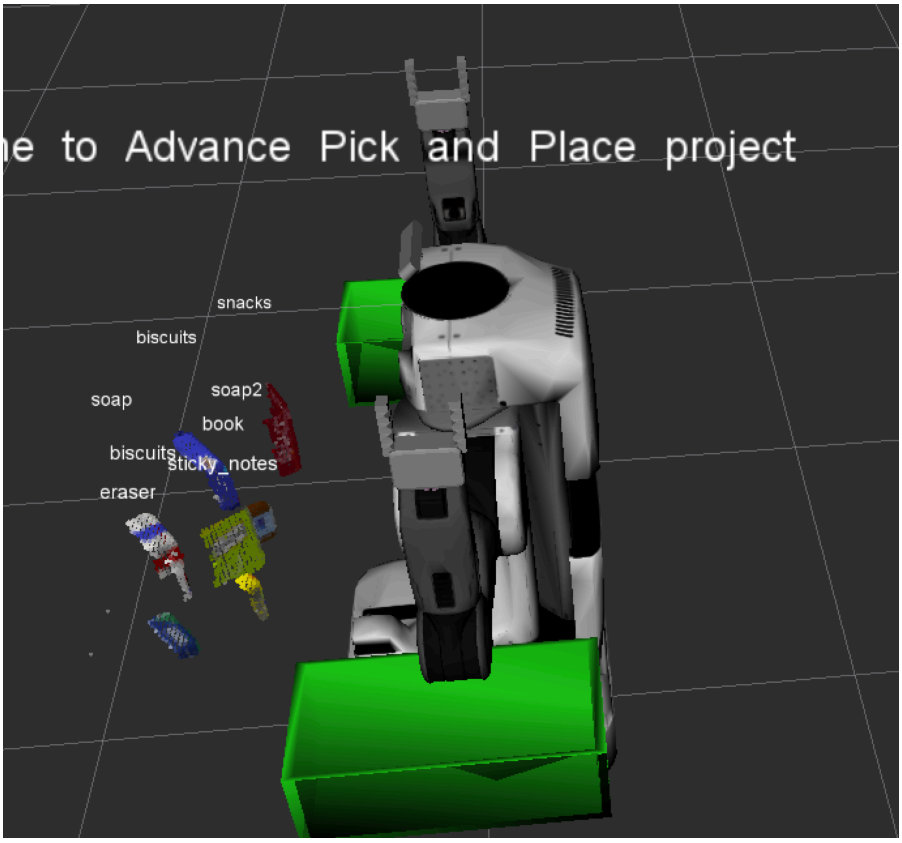






2. **Complete Exercise 2 steps: Pipeline including clustering for segmentation implemented.**

   I add the code in segmentation.py and get cluster_indices=6 for the demo class data. Then I add the code to project_template.py in 113~157 lines. The result of object cluster indices in test3:

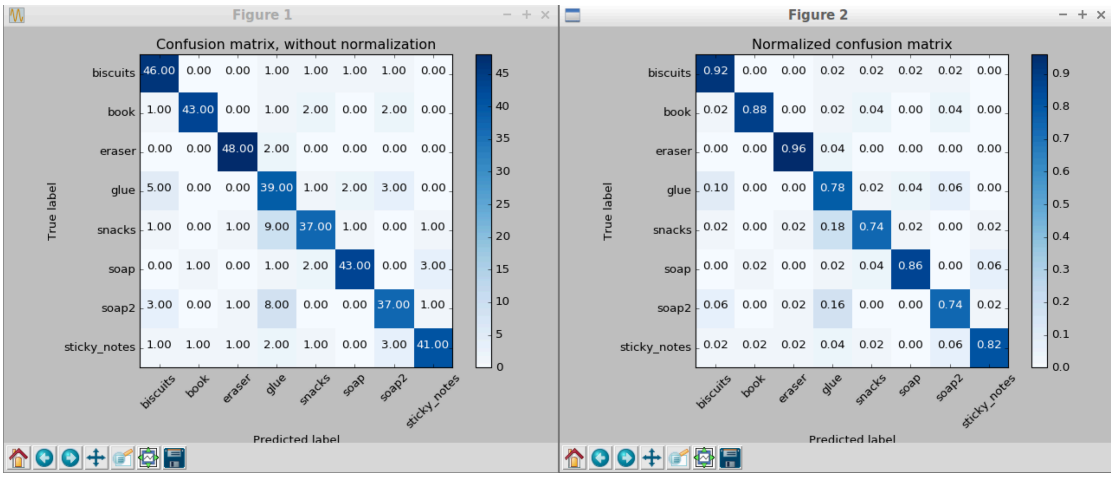



```
cluster_indices= 8
[INFO] [1525512428.005263, 1446.597000]: Detected 8 objects: ['biscuits', 'snacks', 'soap', 'book', 'eraser', 'soap2', 'sticky_notes', 'glue']
```

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

I completed the functions in features.py, and add the code in pcl_callback() function in 168~197 lines. With testN.world, I modified capture_feature.py to provide training_setN.sav and train_svm.py to build modelN.sav.

```
-rw-rw-r--  1 robond robond  194257 May  4 06:46 model1.sav
-rw-rw-r--  1 robond robond  440663 May  4 07:10 model2.sav
-rw-rw-r--  1 robond robond  800815 May  4 20:06 model3.sav
-rw-rw-r--  1 robond robond    1149 May  4 20:46 output_1.yaml
-rw-rw-r--  1 robond robond    1898 May  4 21:28 output_2.yaml
-rw-rw-r--  1 robond robond    3049 May  5 02:27 output_3.yaml
drwxrwxr-x  6 robond robond    4096 Apr 25 06:32 src/
-rw-rw-r--  1 robond robond  806662 May  4 06:43 training_set1.sav
-rw-rw-r--  1 robond robond 1364220 May  4 06:54 training_set2.sav
-rw-rw-r--  1 robond robond 2190160 May  4 19:52 training_set3.sav
```

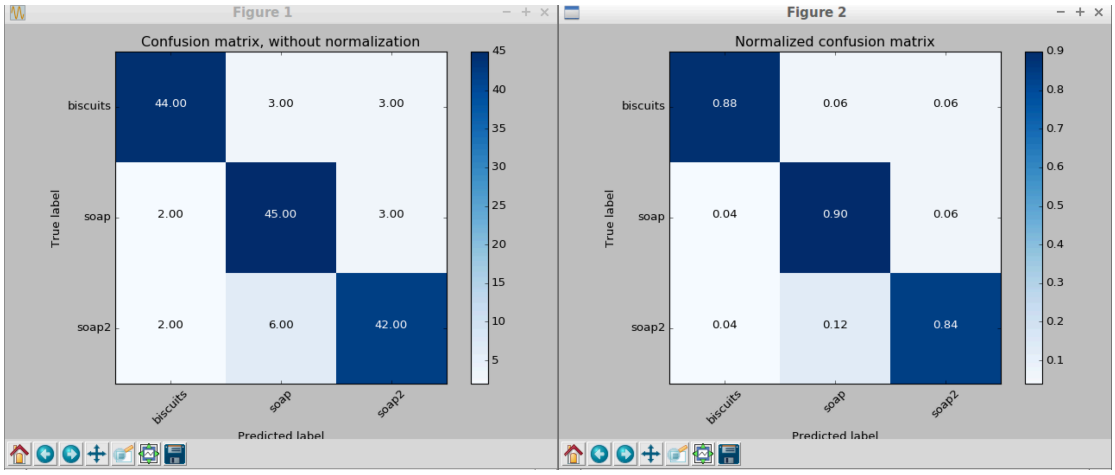


Pick and Place Setup

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.

I trained three SVM classifiers for test1, test2 and test3.

The normalized confusion matrix of three classifiers:



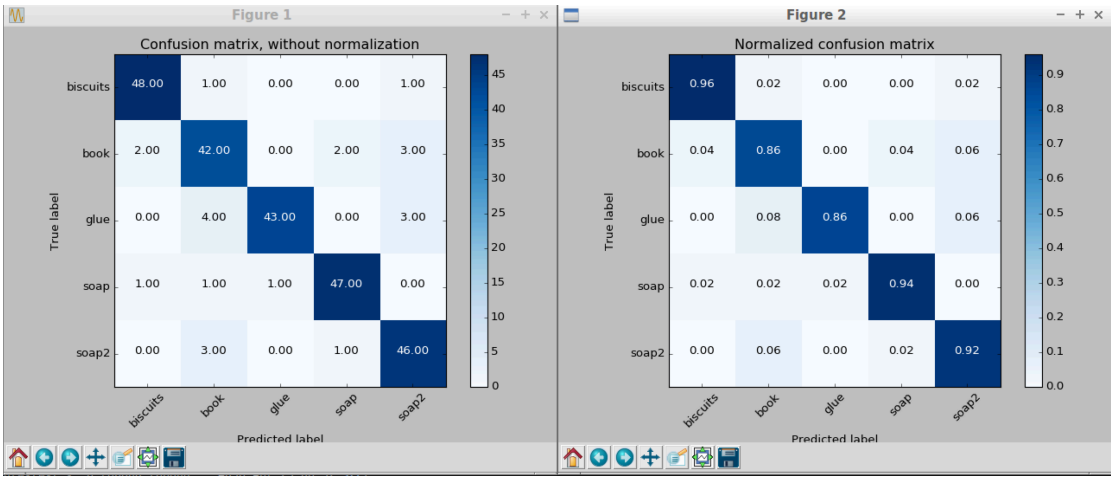


Figure 1
Confusion matrix, without normalization
biscuits 48.00 1.00 0.00 0.00 1.00
book 2.00 42.00 0.00 2.00 3.00
glue 0.00 4.00 43.00 0.00 3.00
soap 1.00 1.00 1.00 47.00 0.00
soap2 0.00 3.00 0.00 1.00 46.00
True label
biscuits book glue soap soap2
Predicted label
45 40 35 30 25 20 15 10 5 0
Figure 2
Normalized confusion matrix
biscuits 0.96 0.02 0.00 0.00 0.02
book 0.04 0.86 0.00 0.04 0.06
glue 0.00 0.08 0.86 0.00 0.06
soap 0.02 0.02 0.02 0.94 0.00
soap2 0.00 0.06 0.00 0.02 0.92
True label
biscuits book glue soap soap2
Predicted label
0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0.0

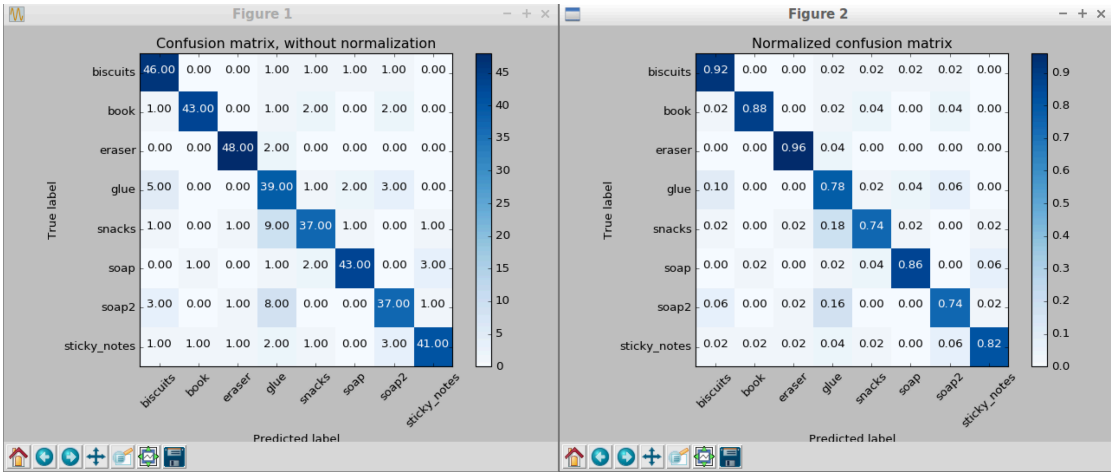
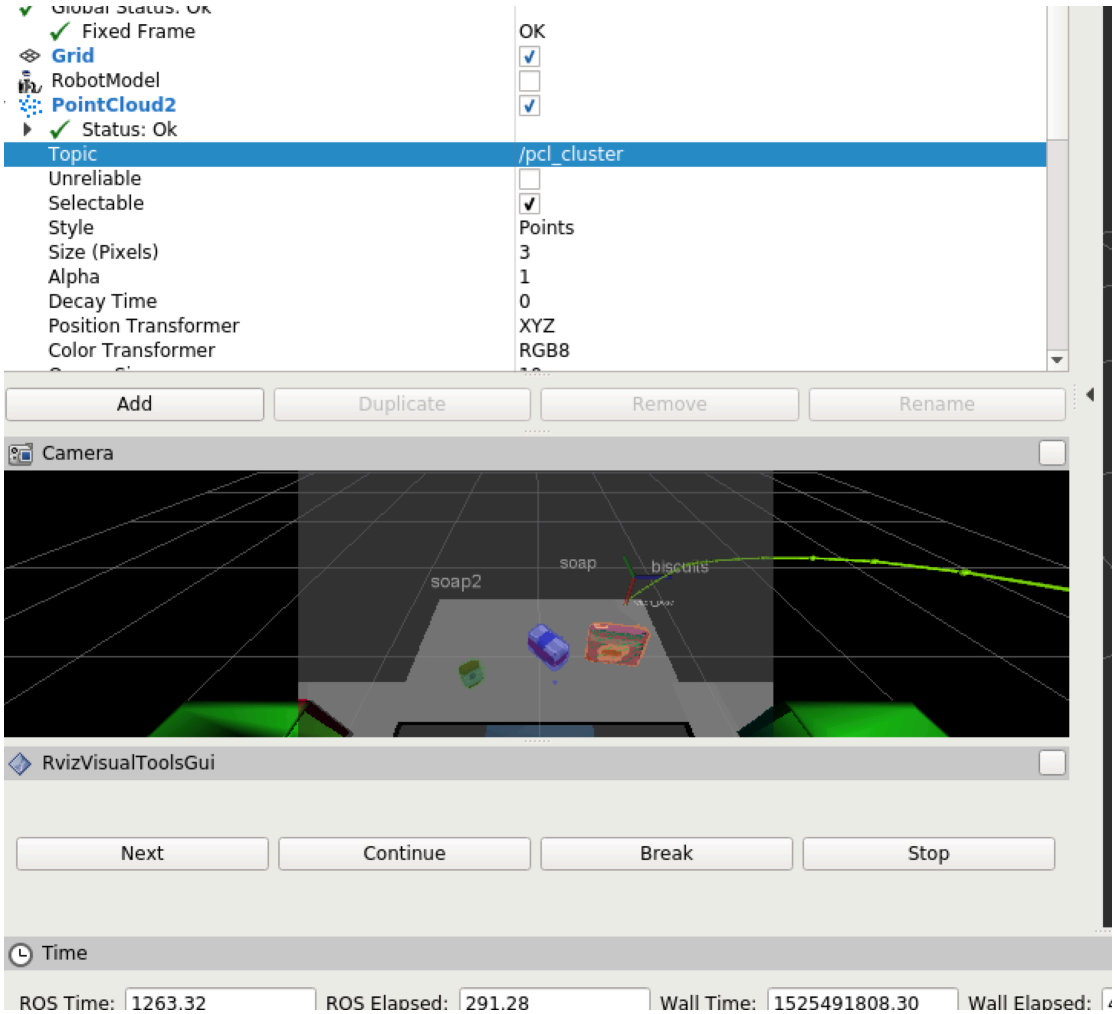
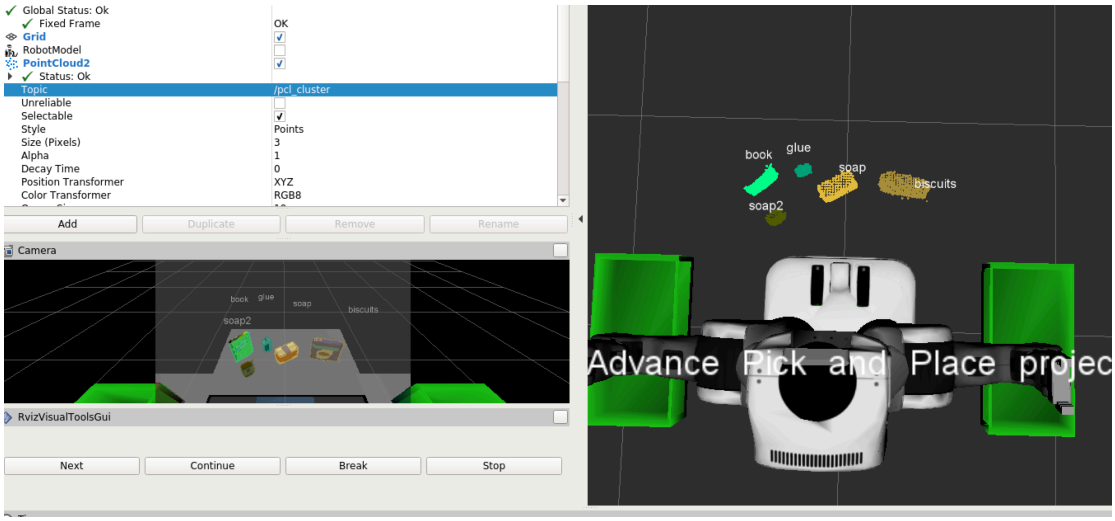


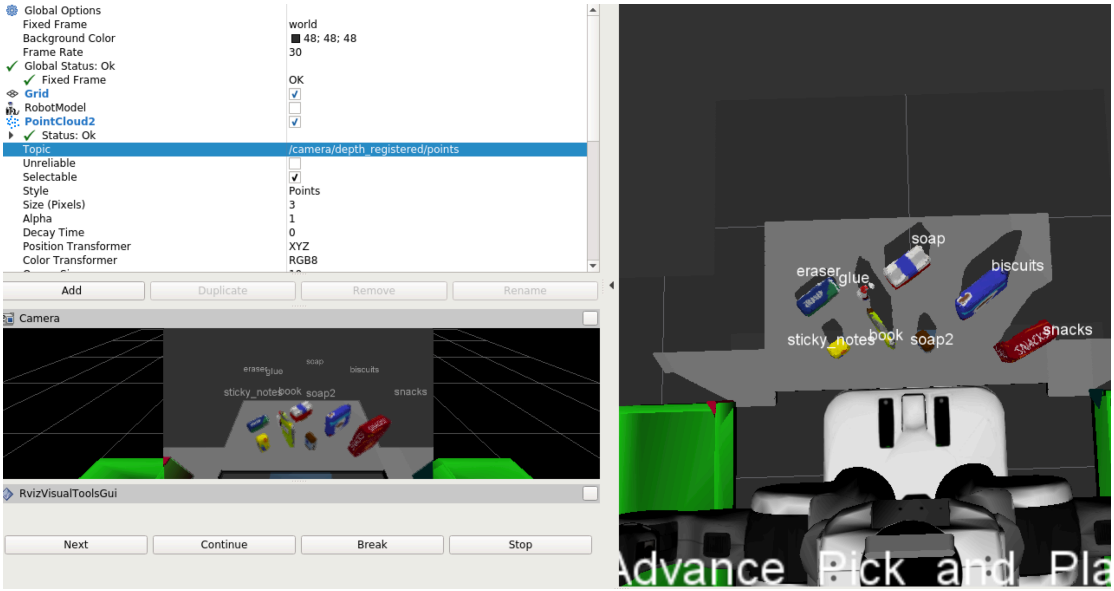
Figure 1
Confusion matrix, without normalization
biscuits 46.00 0.00 0.00 1.00 1.00 1.00 1.00 0.00
book 1.00 43.00 0.00 1.00 2.00 0.00 2.00 0.00
eraser 0.00 0.00 48.00 2.00 0.00 0.00 0.00 0.00
glue 5.00 0.00 0.00 39.00 1.00 2.00 3.00 0.00
snacks 1.00 0.00 1.00 9.00 37.00 1.00 0.00 1.00
soap 0.00 1.00 0.00 1.00 2.00 43.00 0.00 3.00
soap2 3.00 0.00 1.00 8.00 0.00 0.00 37.00 1.00
sticky_notes 1.00 1.00 1.00 2.00 1.00 0.00 3.00 41.00
True label
biscuits book eraser glue snacks soap soap2 sticky_notes
Predicted label
45 40 35 30 25 20 15 10 5 0
Figure 2
Normalized confusion matrix
biscuits 0.92 0.00 0.00 0.02 0.02 0.02 0.02 0.00
book 0.02 0.88 0.00 0.02 0.04 0.00 0.04 0.00
eraser 0.00 0.00 0.96 0.04 0.00 0.00 0.00 0.00
glue 0.10 0.00 0.00 0.78 0.02 0.04 0.06 0.00
snacks 0.02 0.00 0.02 0.18 0.74 0.02 0.00 0.02
soap 0.00 0.02 0.00 0.02 0.04 0.86 0.00 0.06
soap2 0.06 0.00 0.02 0.16 0.00 0.00 0.74 0.02
sticky_notes 0.02 0.02 0.02 0.04 0.02 0.00 0.06 0.82
True label
biscuits book eraser glue snacks soap soap2 sticky_notes
Predicted label
0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0.0



Correctly identify 100% of objects 3/3 in test1.world



Correctly identify 100% of objects 5/5 in test2.world



Correctly identify 100% of objects 8/8 in test3.world

Spend some time at the end to discuss your code, what techniques you used, what worked and why, where the implementation might fail and how you might improve it if you were going to pursue this project further.

At first, I use the filters that introduced in class and I run in the Exercise 1, 2 and 3. But in three environments it can't be clustered successfully. Then I add the Outlier Removal Filter to avoid noisy point cloud data and tuning parameters for each filter.

When I trained SVM classifier, I got low AUC result less then 50% in test2 and test3. I think the train data is not enough. I modified the capture_feature.py code from 5 samples to 50 samples for each objects. It's a good way to improve AUC and I get more than 80% for each classifier.

If I'm going to pursue this project further, I think I will continue to improve the classifier AUC at first, by adding more train data, tuning model parameters and trying to add more useful features. After successful object recognition, I need publish a point cloud which the motion planning pipeline can subscribe to and use it for a 3D collision map, and determine what point cloud should consist of for optimum collision avoidance and finally make the PR2 robot to successfully pick up all the objects.