The writeup / README should include a statement and supporting figures / images that explain how each rubric item was addressed,

###Exercise 1, 2 and 3 Pipeline Implemented

##The pcl_callback() function

Step:

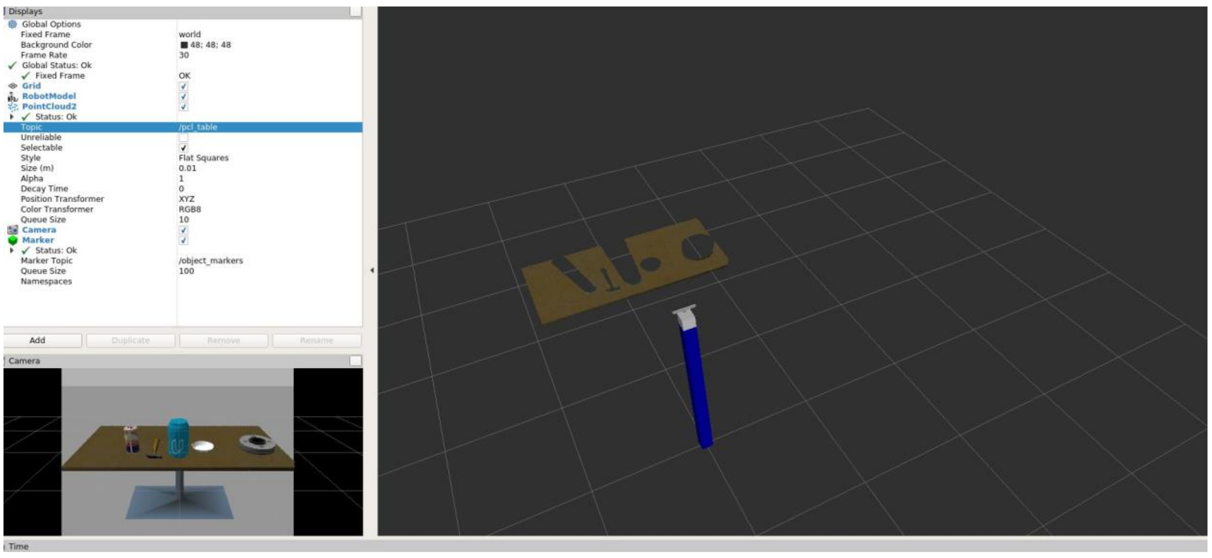
I. Convert the incoming ROS cloud to PCL format
II. Change a voxel grid filter to down sample the image, I set LEAF_SIZE on 0.01 which is good enough for exercise.
III. Complete the RANSAC plan segmented to find the table image and you can publish table and object cloud separately. See below Code and for result image.

#Code

```
# Voxel Grid Downsampling

   vox = cloud.make_voxel_grid_filter()

   LEAF_SIZE = 0.01

   vox.set_leaf_size(LEAF_SIZE, LEAF_SIZE, LEAF_SIZE)

   cloud_filtered = vox.filter()


   # PassThrough Filter to isolate the table and objects

   passthrough = cloud_filtered.make_passthrough_filter()

   passthrough.set_filter_field_name('z')

   axis_min = 0.76

   axis_max = 1.1

   passthrough.set_filter_limits(axis_min, axis_max)

   cloud_filtered = passthrough.filter()
```

```
# RANSAC Plane Segmentation to identify the table from the objects
seg = cloud_filtered.make_segmenter()
seg.set_model_type(pcl.SACMODEL_PLANE)
seg.set_method_type(pcl.SAC_RANSAC)
max_distance = 0.01
seg.set_distance_threshold(max_distance)
inliers, coefficients = seg.segment()


# Extract inliers (table surface) and outliers (objects)
cloud_table = cloud_filtered.extract(inliers, negative=False)
cloud_objects = cloud_filtered.extract(inliers, negative=True)
```

# Image result



Displays

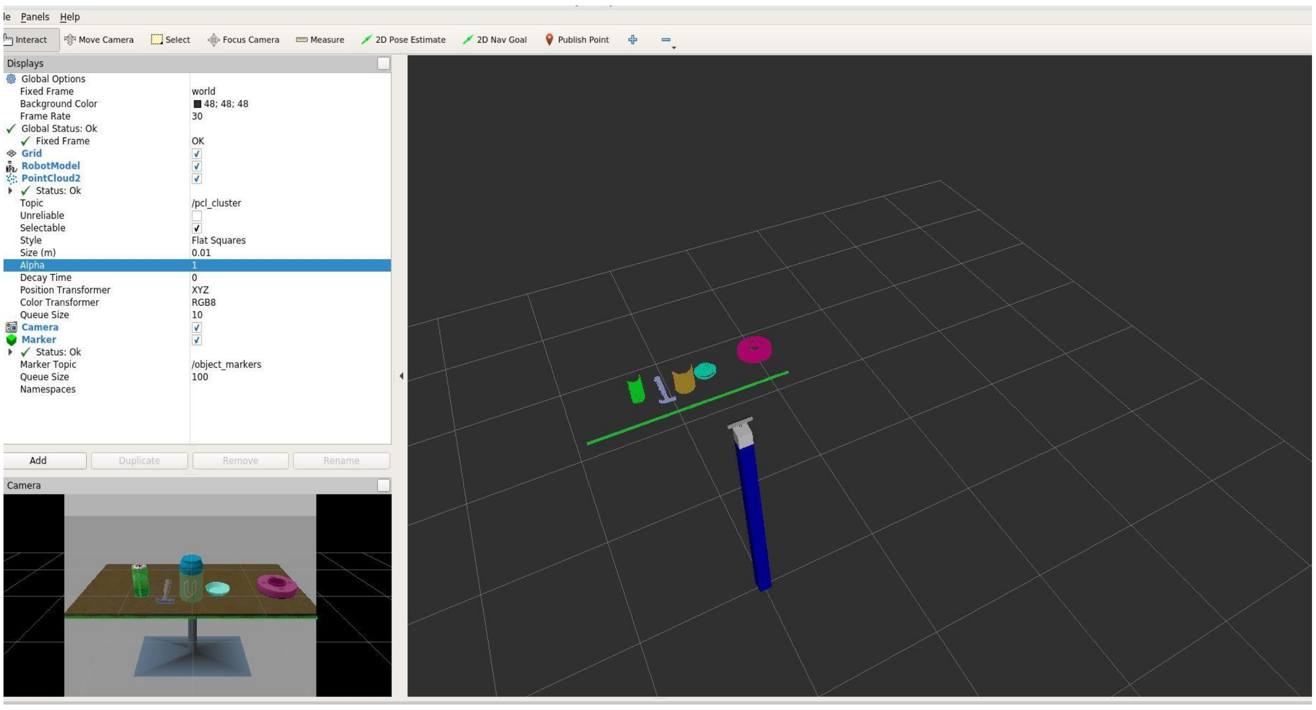
Global Options	
Fixed Frame	world
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
RobotModel	✓
PointCloud2	✓
✓ Status: Ok	
Topic	/pcl_table
Unreliable	
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	RGB8
Queue Size	10
Camera	✓
Marker	✓
✓ Status: Ok	
Marker Topic	/object_markers
Queue Size	100
Namespaces	

## Exercise 2 steps: Pipeline including clustering for segmentation

Steps:

I. In this process we use Euclidean clustering to construct point cloud using object. first, I convert the XYZ RGB cloud to xyz because K-d tree only needs xyz to create white cloud.

II. After that, I applied Euclidean clustering, setting tolerance, minimum cluster size and maximum cluster size 0.05, 100, 3000 respectively. Next step followed as mention in class. Result seen below in picture.



Displays

Global Options	
Fixed Frame	world
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
RobotModel	✓
PointCloud2	✓
✓ Status: Ok	
Topic	/pcl_cluster
Unreliable	☐
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	RGB8
Queue Size	10
Camera	✓
Marker	✓
✓ Status: Ok	
Marker Topic	/object_markers
Queue Size	100
Namespaces	

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

Here, I added code on features.py, function called ‘compute_color_histograms’ and ‘compute_normal_histograms’. where the range of colours is [0,255] and bin value is 32. Which works fine for this project.

In the ‘capture_features’ script, for test1.world, and test2.world I set range i=100, which give above 90% accuracy, but when I run test3.world accuracy was 74%. So, I change i=1000, after running script it accuracy is 94%. it took me more than 2 hours to generate ‘tranning_set.svm’ script. Result for world1, world2 and world3 can see as below.

<u>Test1.world</u>

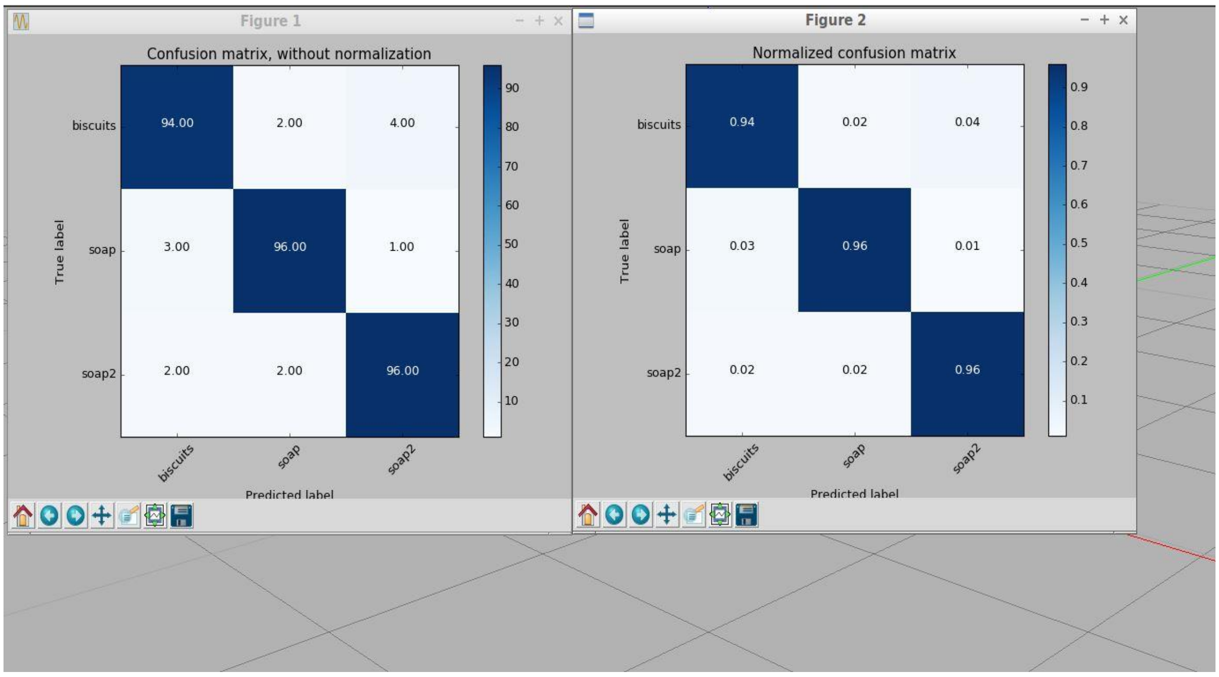


Figure 1
Confusion matrix, without normalization
True label
Predicted label
Figure 2
Normalized confusion matrix
True label
Predicted label

	biscuits	soap	soap2
biscuits	94.00	2.00	4.00
soap	3.00	96.00	1.00
soap2	2.00	2.00	96.00

	biscuits	soap	soap2
biscuits	0.94	0.02	0.04
soap	0.03	0.96	0.01
soap2	0.02	0.02	0.96

Test2.world

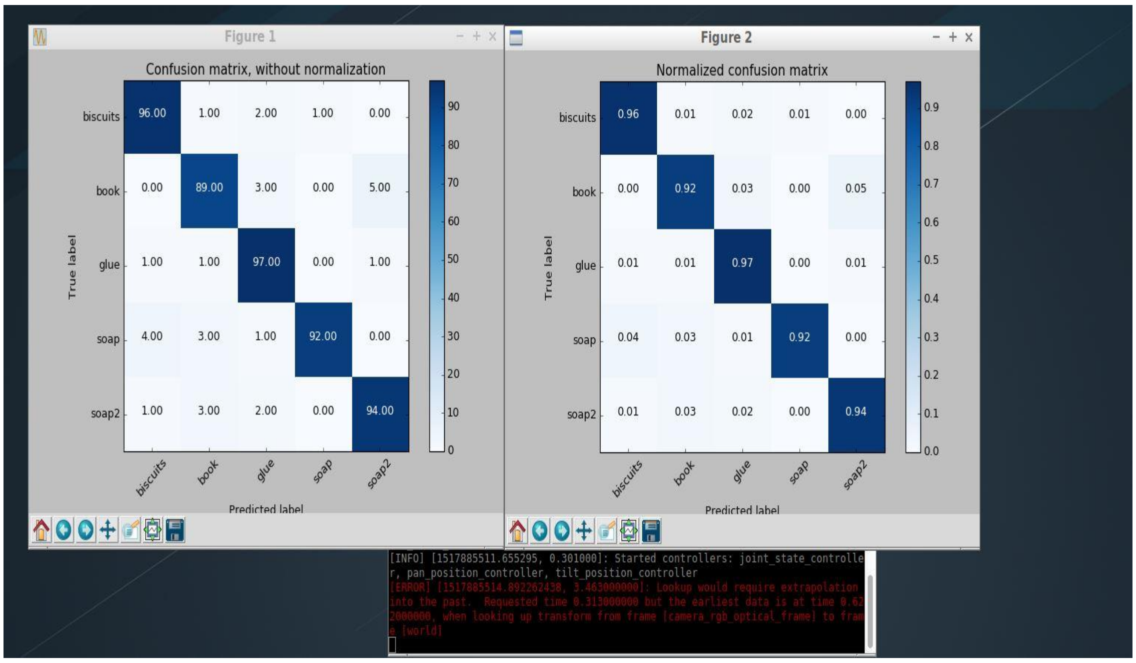


Figure 1
Confusion matrix, without normalization
True label
Predicted label
Figure 2
Normalized confusion matrix
True label
Predicted label

	biscuits	book	glue	soap	soap2
biscuits	96.00	1.00	2.00	1.00	0.00
book	0.00	89.00	3.00	0.00	5.00
glue	1.00	1.00	97.00	0.00	1.00
soap	4.00	3.00	1.00	92.00	0.00
soap2	1.00	3.00	2.00	0.00	94.00

	biscuits	book	glue	soap	soap2
biscuits	0.96	0.01	0.02	0.01	0.00
book	0.00	0.92	0.03	0.00	0.05
glue	0.01	0.01	0.97	0.00	0.01
soap	0.04	0.03	0.01	0.92	0.00
soap2	0.01	0.03	0.02	0.00	0.94

```
[INFO] [1517885511.655295, 0.301000]: Started controllers: joint_state_controlle
r, pan_position_controller, tilt_position_controller
[ERROR] [1517885514.892262438, 3.463000000]: Lookup would require extrapolation
into the past.  Requested time 0.313000000 but the earliest data is at time 0.62
2000000, when looking up transform from frame [camera_rgb_optical_frame] to fram
e [world]
```

Test3.World

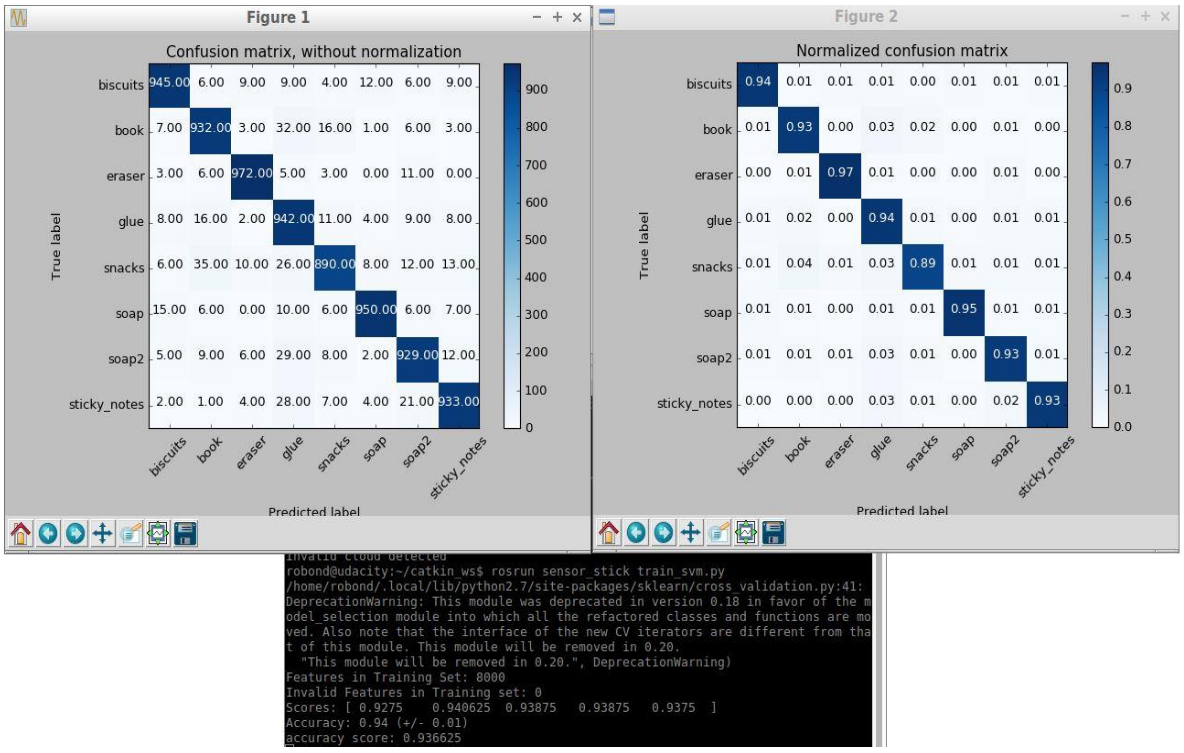


Figure 1
Confusion matrix, without normalization
True label
Predicted label
Figure 2
Normalized confusion matrix
True label
Predicted label

	biscuits	book	eraser	glue	snacks	soap	soap2	sticky_notes
biscuits	945.00	6.00	9.00	9.00	4.00	12.00	6.00	9.00
book	7.00	932.00	3.00	32.00	16.00	1.00	6.00	3.00
eraser	3.00	6.00	972.00	5.00	3.00	0.00	11.00	0.00
glue	8.00	16.00	2.00	942.00	11.00	4.00	9.00	8.00
snacks	6.00	35.00	10.00	26.00	890.00	8.00	12.00	13.00
soap	15.00	6.00	0.00	10.00	6.00	950.00	6.00	7.00
soap2	5.00	9.00	6.00	29.00	8.00	2.00	929.00	12.00
sticky_notes	2.00	1.00	4.00	28.00	7.00	4.00	21.00	933.00

	biscuits	book	eraser	glue	snacks	soap	soap2	sticky_notes
biscuits	0.94	0.01	0.01	0.01	0.00	0.01	0.01	0.01
book	0.01	0.93	0.00	0.03	0.02	0.00	0.01	0.00
eraser	0.00	0.01	0.97	0.01	0.00	0.00	0.01	0.00
glue	0.01	0.02	0.00	0.94	0.01	0.00	0.01	0.01
snacks	0.01	0.04	0.01	0.03	0.89	0.01	0.01	0.01
soap	0.01	0.01	0.00	0.01	0.01	0.95	0.01	0.01
soap2	0.01	0.01	0.01	0.03	0.01	0.00	0.93	0.01
sticky_notes	0.00	0.00	0.00	0.03	0.01	0.00	0.02	0.93

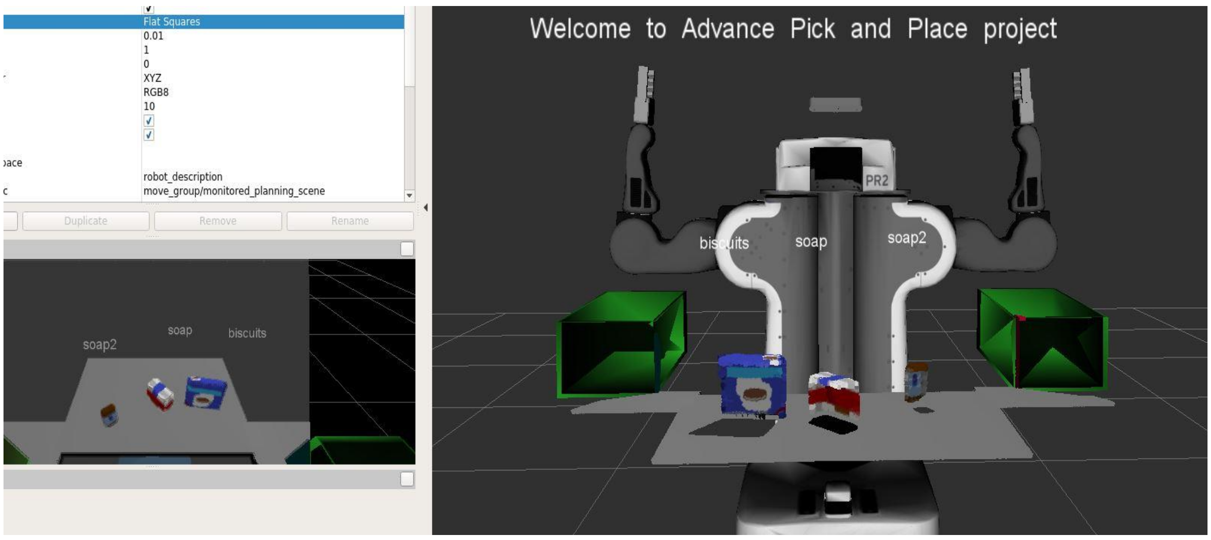
```
Invalid cloud detected
robond@udacity:~/catkin_ws$ rosrun sensor_stick train_svm.py
/home/robond/.local/lib/python2.7/site-packages/sklearn/cross_validation.py:41:
DeprecationWarning: This module was deprecated in version 0.18 in favor of the m
odel_selection module into which all the refactored classes and functions are mo
ved. Also note that the interface of the new CV iterators are different from tha
t of this module. This module will be removed in 0.20.
  "This module will be removed in 0.20.", DeprecationWarning)
Features in Training Set: 8000
Invalid Features in Training set: 0
Scores: [ 0.9275    0.940625  0.93875   0.93875   0.9375  ]
Accuracy: 0.94 (+/- 0.01)
accuracy score: 0.936625
```

### **Pick and Place Setup**

Here, to identify object of three test list, we have to edit the ‘pick_place_project.launch’ file form folder ‘RoboND-Perception-Project/pr2_robot/launch’, and the object recognition can be performed base on the train model.sav published by SVM tanning.

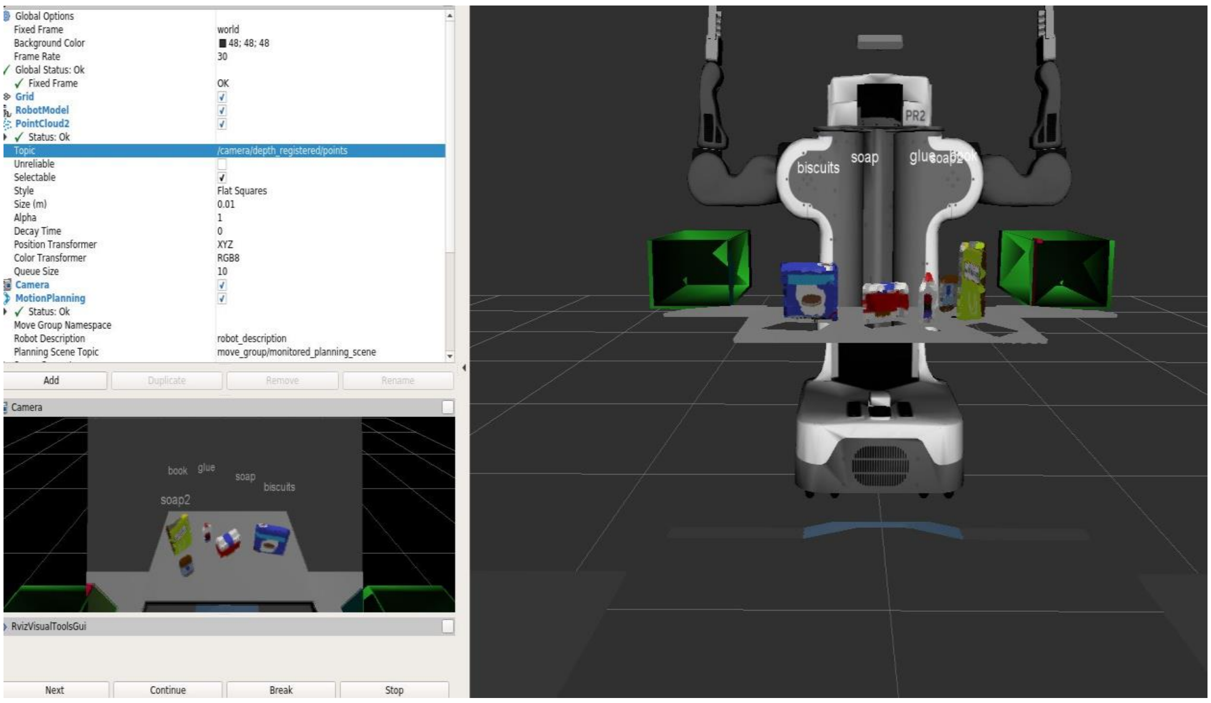
After above, the final process is to publish yaml file for three different object list. Where I create the pr2_mover function, then need to detect object, compute centroid and obtain the test_seen_num from world file and convert it readable datatype by ROS. I follow other process discus as in class room. Below we can see final object recognition success rate in each of the three scenarios.

test1.world (100%)



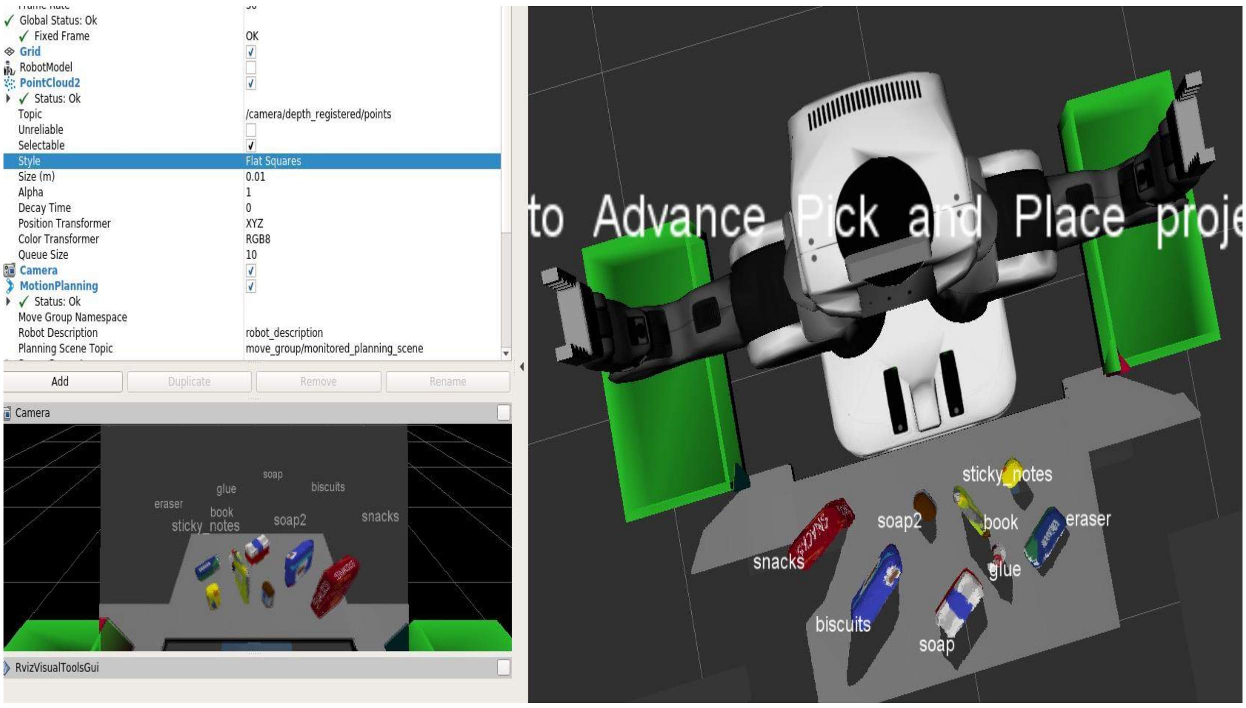
```
                      ✓
                      Flat Squares
                      0.01
                      1
                      0
 r                    XYZ
                      RGB8
                      10
                      ✓
                      ✓
 pace
                      robot_description
 c                    move_group/monitored_planning_scene
```

test2.world (100%)



```
Global Options
  Fixed Frame                 world
  Background Color            ■ 48; 48; 48
  Frame Rate                  30
✓ Global Status: Ok
  ✓ Fixed Frame               OK
Grid                          ✓
RobotModel                    ✓
PointCloud2                   ✓
  ✓ Status: Ok
  Topic                       /camera/depth_registered/points
  Unreliable                  ☐
  Selectable                  ✓
  Style                       Flat Squares
  Size (m)                    0.01
  Alpha                       1
  Decay Time                  0
  Position Transformer        XYZ
  Color Transformer           RGB8
  Queue Size                  10
Camera                        ✓
MotionPlanning                ✓
  ✓ Status: Ok
  Move Group Namespace
  Robot Description           robot_description
  Planning Scene Topic        move_group/monitored_planning_scene
```

test3.world (100%)



✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
RobotModel	☐
PointCloud2	✓
✓ Status: Ok	
Topic	/camera/depth_registered/points
Unreliable	☐
Selectable	✓
Style	Flat Squares
Size (m)	0.01
Alpha	1
Decay Time	0
Position Transformer	XYZ
Color Transformer	RGB8
Queue Size	10
Camera	✓
MotionPlanning	✓
✓ Status: Ok	
Move Group Namespace	
Robot Description	robot_description
Planning Scene Topic	move_group/monitored_planning_scene

###Conclusion

Overall my script performs well result to pass the project. I did face lot of problem in test3.world and train SVM script, but support of mentor and slack community was very helpful. I find running program on VM at windows is bit frustrated. I need to set up .bashrc every time I run scripts, otherwise it throw some error. And when I set i=1000 to train SVM , it took quit lot of time.