Project Report: Perception Pick & Place

Target: The challenge is that of identifying objects in a 3D environment by using as a sensor an RGBD camera. Once the object have been identified, control a PR2 robot to place them in appropriate boxes. There are 3 different test-case with different objects.

Prerequisites for the project are 3 exercises which together form a perception pipeline. The different scripts developed/modified on the exercises are part of the project submission.

The figure below captures all of the steps used in the perception project:

```
29  # Helper function to get surface normals
30  def get_normals(cloud):
33
34  # Helper function to create a yaml friendly dictionary from ROS messages
35  def make_yaml_dict(test_scene_num, arm_name, object_name, pick_pose, place_pose):
43
44  # Helper function to output to yaml file
45  def send_to_yaml(yaml_filename, dict_list):
49
50  # prefiltering operation of the cloud
51  def pcl_filter(pcl_msg):
109
110 # segmentation & clustering
111 def segmentation(pcl_msg):
189
190 # classification and detection function
191 def classification(pcl_msg):
230
231 # callback function for the subscriber
232 def pcl_callback(pcl_msg):
249
250 # function to load parameters and request PickPlace service
251 def pr2_mover(detected_objects_list):
380
381 if __name__ == '__main__':
425
```

Exercise 1 has been used in writing the **pcl_filter()** function. Exercise 2 used to implement the **segmentation()** function and exercise 3 for the **classification()** function.

Before moving to the execution of the project, a classification model has been trained by using the models of the project within the stick_sensor environment.

Capturing Point Cloud Features

In order to capture the point cloud features, I used the sensor_stick model to analyze the different object of the PR2 project. The models folder from the PR2 project were copied into the models folder of the sensor stick folder.
For this task, the capture_feature.py has been modified in order to reflect the new list of models. Please refer to the file **ex3_capture_features_new_list.py**. The next step is to launch the Gazebo environment:

roslaunch sensor_stick training.launch

Then run the ex3_capture_features_new_list.py as:

rosrun sensor_stick ex3_capture_features_new_list.py

This script will cycle through each model, orienting it in random directions, and capturing point cloud features for it. The final features will be saved in training_set.sav.

Training the model

For training a classifier, the support vector machine algorithm has been used with linear kernel. To start the training, run:

python ex3_train_svm.py

This will train the SVM and save the final model to model.sav. I am able to achieve an accuracy of 85% (+/- 27%), which is not perfect but good.

```
Features in Training Set: 450
Invalid Features in Training set: 3
Scores: [ 0.84444444  0.84444444  0.84444444  0.84444444  0.88888889  0.8
  0.82222222  0.95454545  0.88636364  0.79545455]
Accuracy: 0.85 (+/- 0.09)
accuracy score: 0.852348993289
```

This used a linear kernel in the SVM with a C value of 0.1. Please refer to the following confusion matrices:

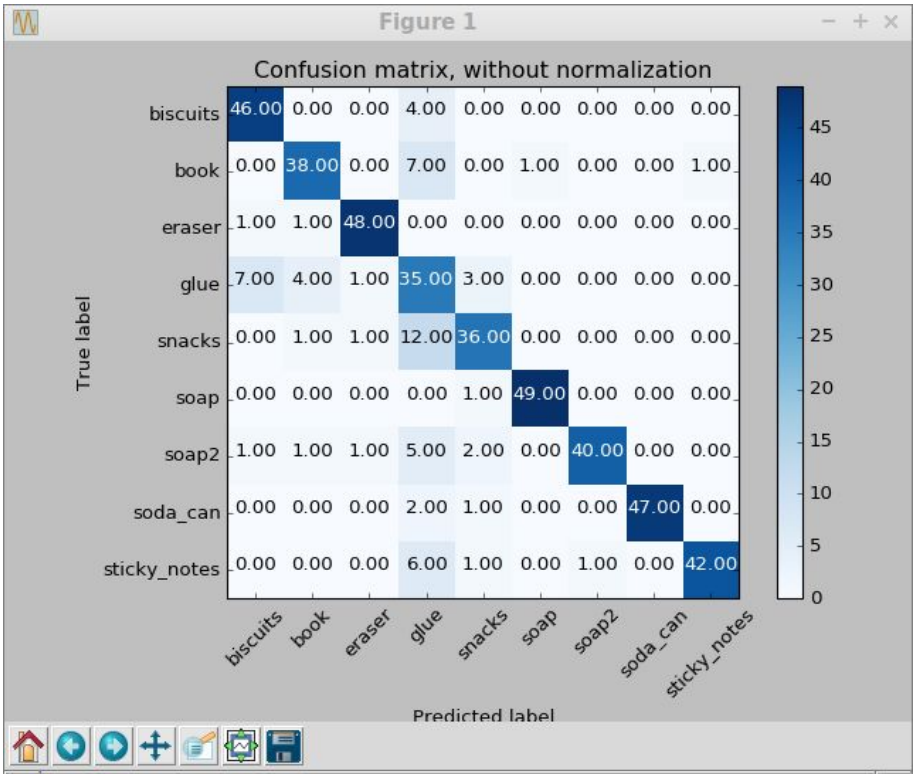


Figure 1
Confusion matrix, without normalization
biscuits 46.00 0.00 0.00 4.00 0.00 0.00 0.00 0.00 0.00
book 0.00 38.00 0.00 7.00 0.00 1.00 0.00 0.00 1.00
eraser 1.00 1.00 48.00 0.00 0.00 0.00 0.00 0.00 0.00
glue 7.00 4.00 1.00 35.00 3.00 0.00 0.00 0.00 0.00
snacks 0.00 1.00 1.00 12.00 36.00 0.00 0.00 0.00 0.00
soap 0.00 0.00 0.00 0.00 1.00 49.00 0.00 0.00 0.00
soap2 1.00 1.00 1.00 5.00 2.00 0.00 40.00 0.00 0.00
soda_can 0.00 0.00 0.00 2.00 1.00 0.00 0.00 47.00 0.00
sticky_notes 0.00 0.00 0.00 6.00 1.00 0.00 1.00 0.00 42.00
True label
biscuits book eraser glue snacks soap soap2 soda_can sticky_notes
Predicted label
45 40 35 30 25 20 15 10 5 0

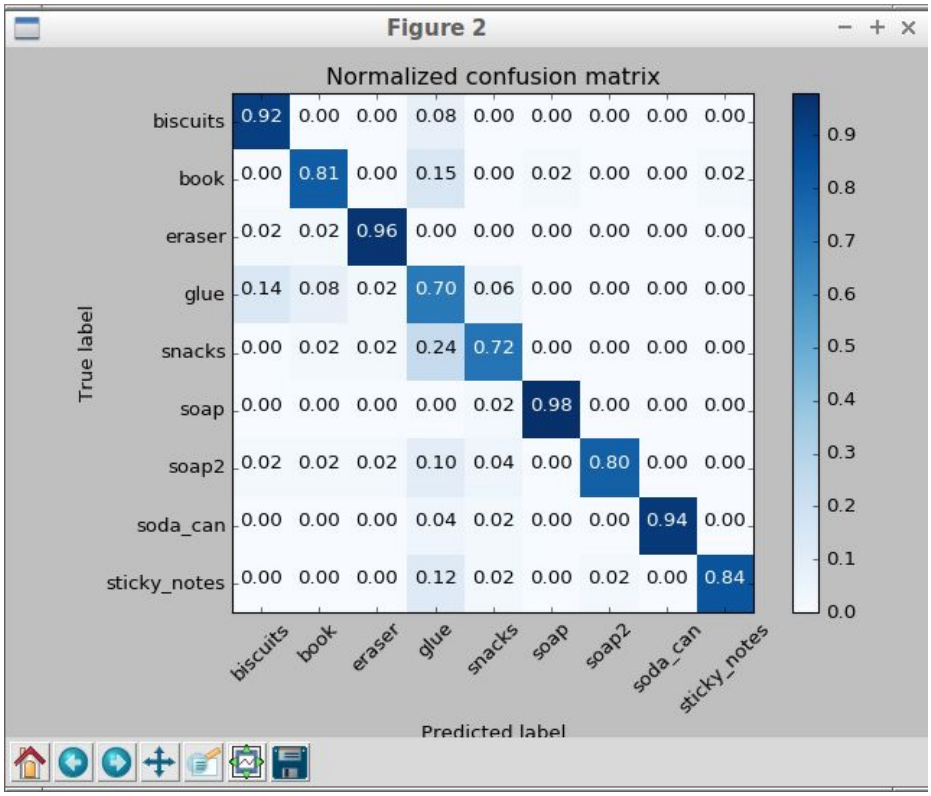


Figure 2
Normalized confusion matrix
biscuits 0.92 0.00 0.00 0.08 0.00 0.00 0.00 0.00 0.00
book 0.00 0.81 0.00 0.15 0.00 0.02 0.00 0.00 0.02
eraser 0.02 0.02 0.96 0.00 0.00 0.00 0.00 0.00 0.00
glue 0.14 0.08 0.02 0.70 0.06 0.00 0.00 0.00 0.00
snacks 0.00 0.02 0.02 0.24 0.72 0.00 0.00 0.00 0.00
soap 0.00 0.00 0.00 0.00 0.02 0.98 0.00 0.00 0.00
soap2 0.02 0.02 0.02 0.10 0.04 0.00 0.80 0.00 0.00
soda_can 0.00 0.00 0.00 0.04 0.02 0.00 0.00 0.94 0.00
sticky_notes 0.00 0.00 0.00 0.12 0.02 0.00 0.02 0.00 0.84
True label
biscuits book eraser glue snacks soap soap2 soda_can sticky_notes
Predicted label
0.9 0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0.0

PR2 Processing

The PR2 robot is able to operate with the model.sav file. To test the PR2 object recognition script, I have launch the Gazebo environment:

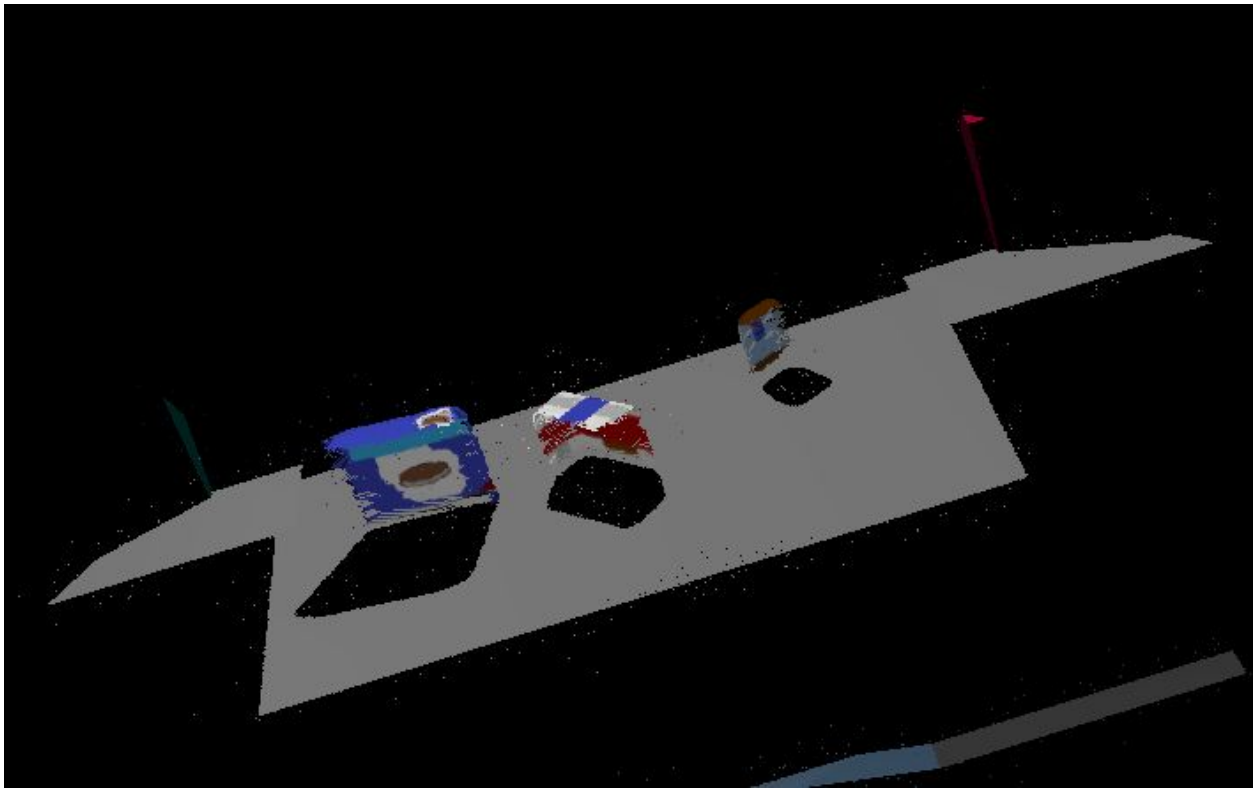
roslaunch pr2_robot pick_place_project.launch

Then run the object recognition script:

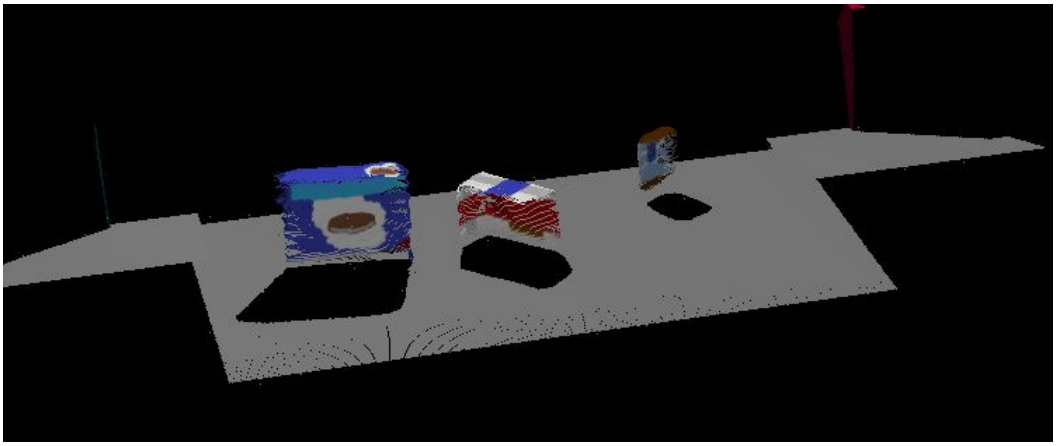
rosrun pr2_robot pr2_perception_project.py

Pipeline Description

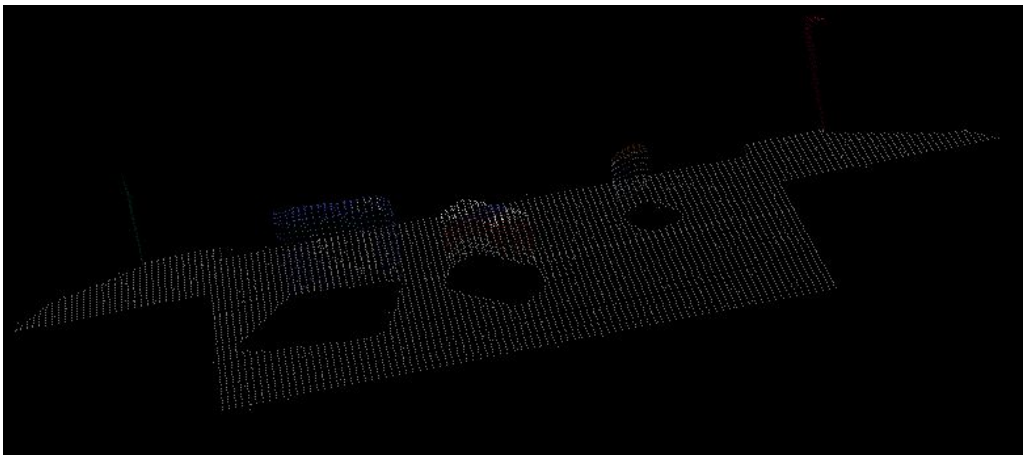
In order to appreciate the different steps, different pcl data have been saved. I started by saving the original point cloud received from the camera sensor. This looks like:



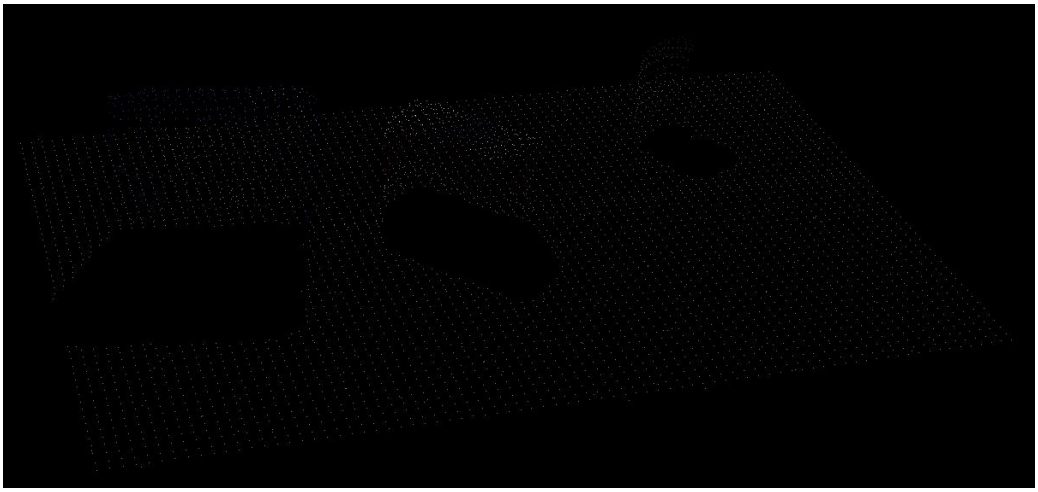
Statistical Outlier Filtering: computes the distance to each point's neighbors, then calculates a mean distance. Any points whose mean distances are outside a defined interval are removed. A mean k value of *20* and a standard deviation threshold of *0.1* provided good filtering. Here is the cloud after performing the outlier removal filter.



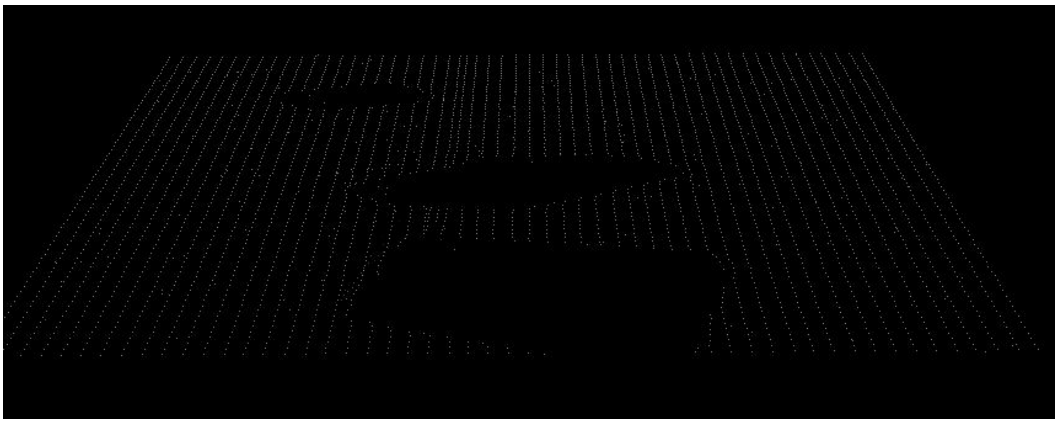
Voxel Grid Filter: The raw point cloud data have more details than required, requiring more processing power when analyzing. A voxel grid filter down samples the data by taking spatial average of the points in the cloud. I have used an X, Y, and Z voxel grid filter leaf size of *0.01* so to leave enough detail while minimizing processing time.

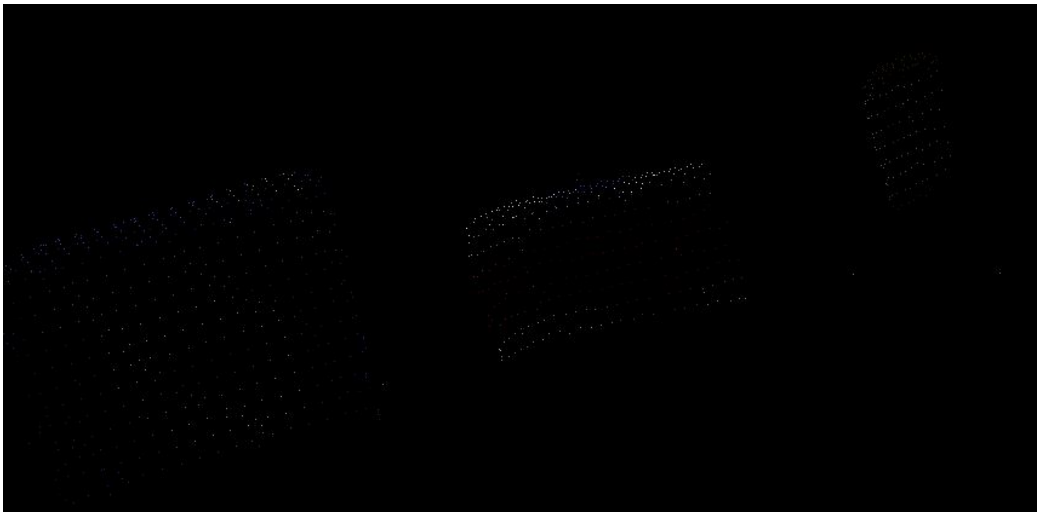


Passthrough Filter: The passthrough filter allows a 3D point cloud to be cropped by specifying an axis with cut-off values along them. I have used passthrough filters for both the X, Y and Z axis. This prevented processing values outside the area immediately in front of the robot. For the used values the following cloud has been obtained:



RANSAC Plane Segmentation: Random Sample Consensus (RANSAC) is used to identify points in the dataset that belong to a particular model. It assumes that all of the data in a dataset is composed of both inliers and outliers, where inliers can be defined by a particular model with a specific set of parameters, and outliers don't. I used a RANSAC max distance value of *0.01*.
The extracted inliers includes the table. The outliers the objects:



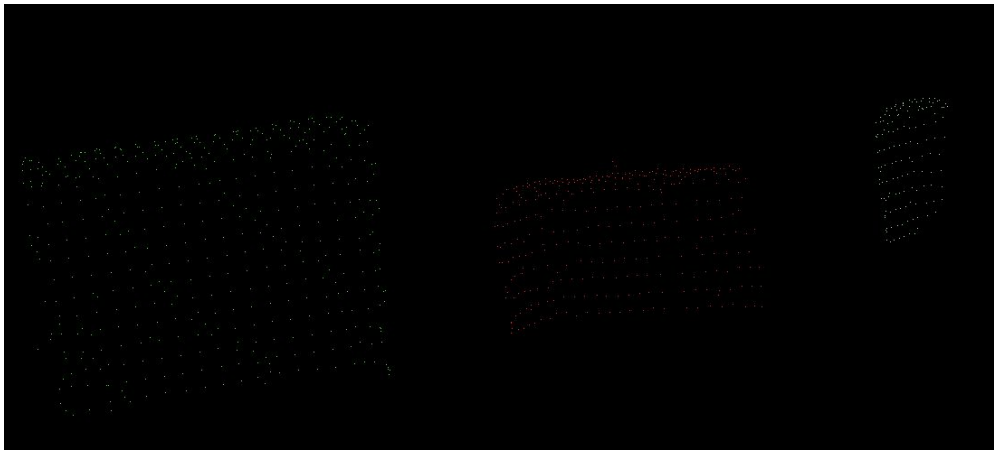


Clustering: It is possible to identify individual objects within a processed cloud. One approach is to use Euclidean clustering to perform this calculation. Two possible algorithms include:

- K-means
- DBSCAN

K-means: K-means clustering algorithm is able to group data points into n groups based on their distance to randomly chosen centroids.

DBSCAN: Is a clustering algorithm that creates clusters by grouping data points that are within some threshold distance from their nearest neighbor. Performing a DBSCAN within the PR2 simulation required converting the XYZRGB point cloud to a XYZ point cloud. By assigning random colors to the isolated objects I could save the following cloud which is related to test case 1.



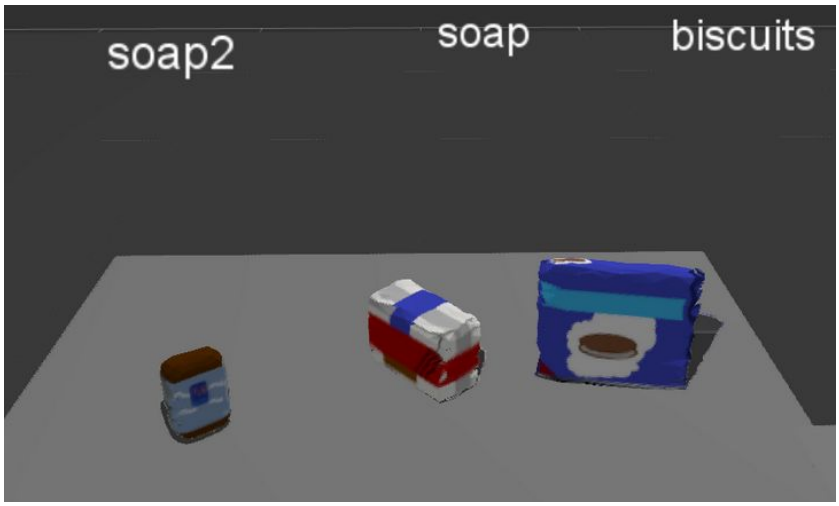
Change the Simulation Environment

To change between test cases, in the file **pick_place_project.launch** in the launch folder, I have changed:

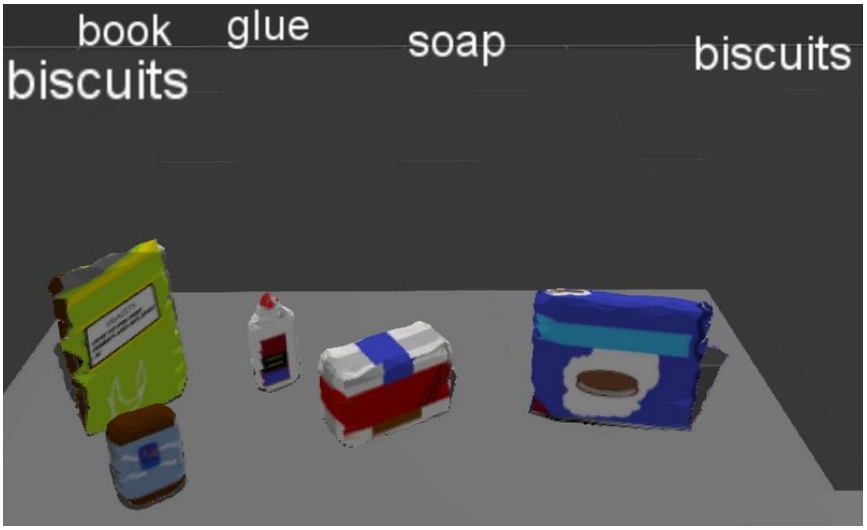
- test1 to test2 or test3 on line 13
- pick_list_1 to pick_list_2 or pick_list_3 on line 39

Relaunch the Gazebo environment for the new objects to be loaded.

Test 1:

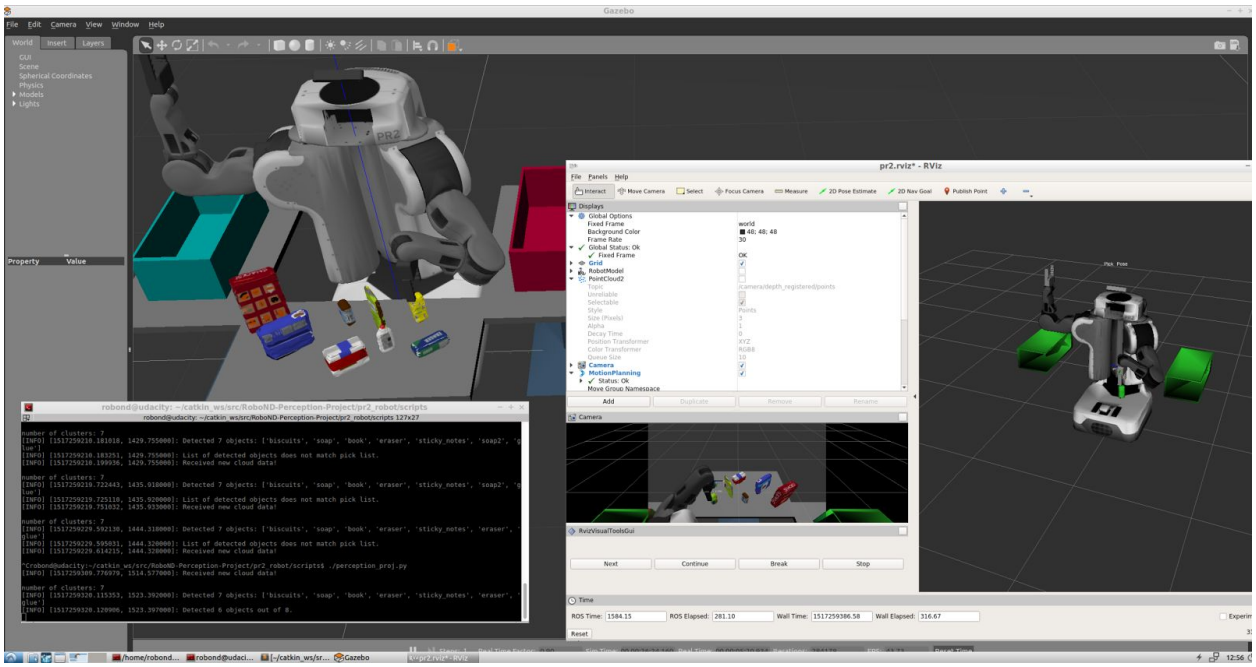


Test 2:



Test 3:

Summary



This project was very interesting and also with some challenge. A full processing pipeline including acquisition, filtering, clustering and identification have been put in place. The simulation environment allows to experiment with all the steps and to fine tune the parameters in the several steps. I have enjoyed feel ready for the next challenge.

What did not work very well is the fact that the robot is not properly grasping the object. Most of them, although the identification and the comand were OK, were left on the table.
I am sure there is room for optimization, but I would like to pass the exam and focus on the keeping the schedule and optimize as my time allows it.