Writeup. Project 3. 3D perception

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

The pcl_callback() has three steps. The first steop: Filtering & RANSAC is explained in Exercise 1 as below:

1. Read data: pcl_data = ros_to_pcl(pcl_msg)
   // ros input to pcl for image processing
2. Voxel Grid down sampling to reduce # points to process
   vox = pcl_data.make_voxel_grid_filter() with LEAF_SIZE = 0.01 (1% from X, Y, Z dim), then it will create cloud of points from the object
3. Statistical filter: to reduce noise from the image, stat_filter is applied based on the tutorial. There is no variable should be determined empirically.
4. Passthrough filter: the most critical filter in this project. It should limit the region of interest from the image captured in camera.
   From the exercise, we create Z-axis filter (0.6~0.8). Min value is set to avoid capture edge of table (0.6) and Max value is set to avoid look at the floor and consider as "BOOK"
   In actual project, Y axis should also be cropped (-0.4 ~ 0.4), otherwise robo-arms and two bins are captured and considered as "BOOK". I guess all of solid plate & rectangular object could be considered as "BOOK". So we need to crop the image in both Z and Y axis just look at the objects on the table
5. RANSAC filter: Same as exercise 1, RANSAC place filter is applied and max_distance is set as 0.01 to remove the table. After segmentation, Outlier become objects

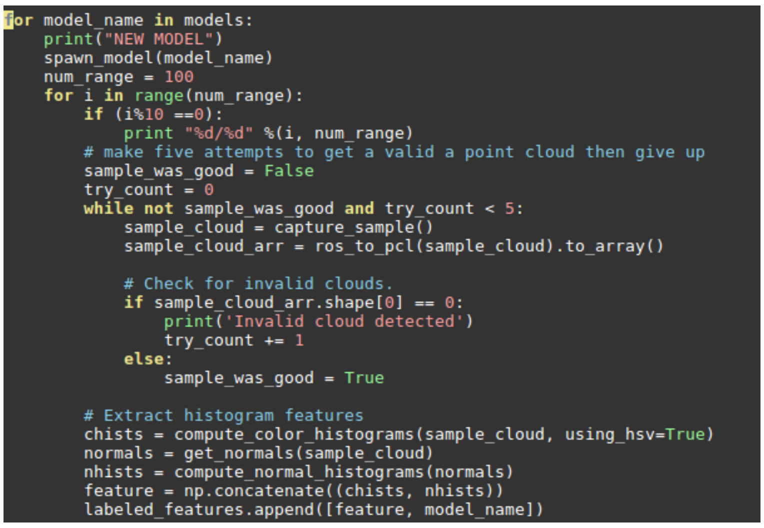
2. Complete Exercise 2 steps: Pipeline including clustering for segmentation implemented

In pcl_callback(), the clustering is implemented as what we did in Exercise 2.

1. Euclidian Clustering to separate clouds into different objects. We don't know what this object is yet. I followed the procedure from Exercise 2 but need to tune the parameters such as Cluster tolerance, Min ~ Max cluster. I found these values from the Exercise 2.
   ec.set_ClusterTolerance(0.02), ec.set_MinClusterSize(10), ec.set_MaxClusterSize(25000)

3. Complete Exercise 3 steps. Feature extracted and SVM trained. Object recognition implemented.

1. It's separate file/flow from project. Not included in pcl_callback() function. It is part of sensor_stick. capture_features.py and train_svm.py
2. Features extracted and SVM trained
   To guessing well, I captured features 100 times (a single object is rotated randomly 100 times) and image is captured in HSV format. In SVM training, it achieve 0.98 score. Below are capture_features.py



```
for model_name in models:
    print("NEW MODEL")
    spawn_model(model_name)
    num_range = 100
    for i in range(num_range):
        if (i%10 ==0):
            print "%d/%d" %(i, num_range)
        # make five attempts to get a valid a point cloud then give up
        sample_was_good = False
        try_count = 0
        while not sample_was_good and try_count < 5:
            sample_cloud = capture_sample()
            sample_cloud_arr = ros_to_pcl(sample_cloud).to_array()

            # Check for invalid clouds.
            if sample_cloud_arr.shape[0] == 0:
                print('Invalid cloud detected')
                try_count += 1
            else:
                sample_was_good = True

        # Extract histogram features
        chists = compute_color_histograms(sample_cloud, using_hsv=True)
        normals = get_normals(sample_cloud)
        nhists = compute_normal_histograms(normals)
        feature = np.concatenate((chists, nhists))
        labeled_features.append([feature, model_name])
```

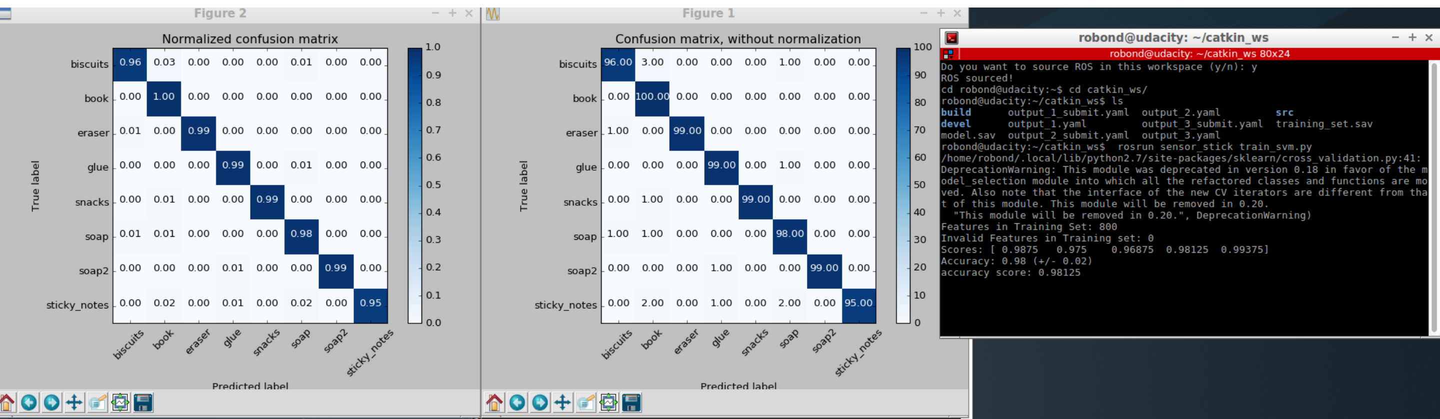


Figure 2

Normalized confusion matrix

True label \ Predicted label	biscuits	book	eraser	glue	snacks	soap	soap2	sticky_notes
biscuits	0.96	0.03	0.00	0.00	0.00	0.01	0.00	0.00
book	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
eraser	0.01	0.00	0.99	0.00	0.00	0.00	0.00	0.00
glue	0.00	0.00	0.00	0.99	0.00	0.01	0.00	0.00
snacks	0.00	0.01	0.00	0.00	0.99	0.00	0.00	0.00
soap	0.01	0.01	0.00	0.00	0.00	0.98	0.00	0.00
soap2	0.00	0.00	0.00	0.01	0.00	0.00	0.99	0.00
sticky_notes	0.00	0.02	0.00	0.01	0.00	0.02	0.00	0.95

Figure 1

Confusion matrix, without normalization

True label \ Predicted label	biscuits	book	eraser	glue	snacks	soap	soap2	sticky_notes
biscuits	96.00	3.00	0.00	0.00	0.00	1.00	0.00	0.00
book	0.00	100.00	0.00	0.00	0.00	0.00	0.00	0.00
eraser	1.00	0.00	99.00	0.00	0.00	0.00	0.00	0.00
glue	0.00	0.00	0.00	99.00	0.00	1.00	0.00	0.00
snacks	0.00	1.00	0.00	0.00	99.00	0.00	0.00	0.00
soap	1.00	1.00	0.00	0.00	0.00	98.00	0.00	0.00
soap2	0.00	0.00	0.00	1.00	0.00	0.00	99.00	0.00
sticky_notes	0.00	2.00	0.00	1.00	0.00	2.00	0.00	95.00

robond@udacity: ~/catkin_ws

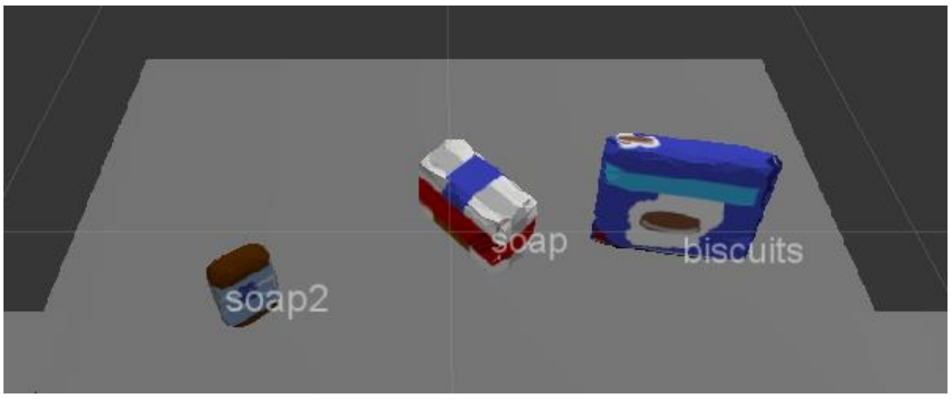
```
Do you want to source ROS in this workspace (y/n): y
ROS sourced!
cd robond@udacity:~$ cd catkin_ws/
robond@udacity:~/catkin_ws$ ls
build      output_1_submit.yaml  output_2.yaml         src
devel      output_1.yaml         output_3_submit.yaml  training_set.sav
model.sav  output_2_submit.yaml  output_3.yaml
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: 800
Invalid Features in Training set: 0
Scores: [ 0.9875    0.975     0.96875   0.98125   0.99375]
Accuracy: 0.98 (+/- 0.02)
accuracy score: 0.98125
```

Writeup. Project 3. 3D perception

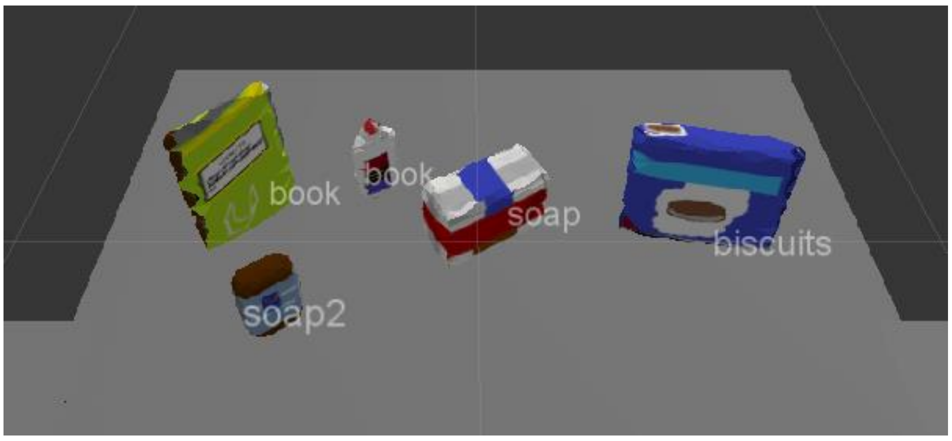
2. **For all three tabletop setups (test*.world), perform object recognition, then read in respective pick list (pick_list_*.yaml). Next construct the messages that would comprise a valid PickPlace request output them to .yaml format.**

   Result: for three different scenarios.

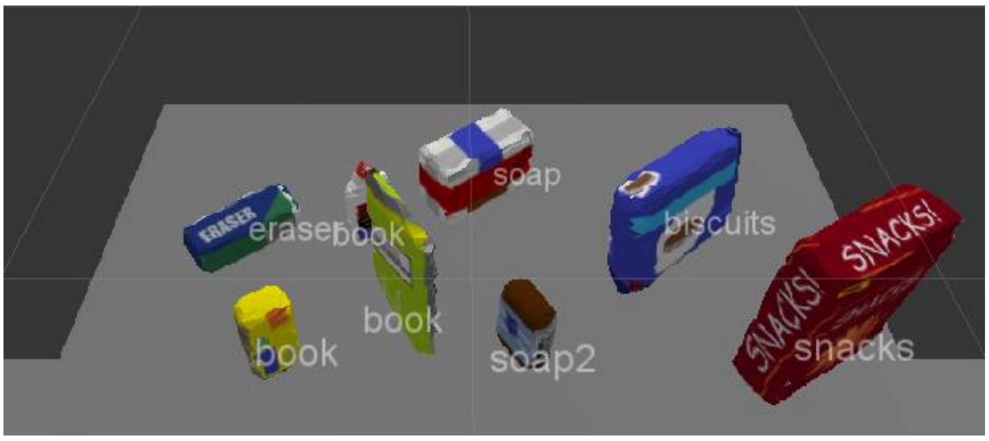
   Output 1: 100%



Output 2: 80% (Glue is captured as book)



Output 3: 75% (Glue and Sticky note as captured to book)



Based on training result from SVM (model.sav), the system can guess the label of the object. Of course, I can't guess 100% correctly as we can see from the image above. Based on the guess and the list of the object should be picked up, my robot start main mission: pick and place to correct bin (green or red)

1) Load all target_objects from pick_list using rospy.get_param (line 226 in pr2_mover)
2) For all found object (actually guessed from the image), check this object is in the target_list or not (line 240)
3) If it is in the list, check group: red or green.
   1) If green, place into right
   2) If red, place into left

Since it capture glue and sticky note to books, I need to consider multiple items can be considered to the same object repeatedly. In other words, although one object from the image is considered as book and delivered to the correct bin, it may not be actual book. In the code, two nested loop is designed as below

```
For found_object in the image
        For target_object in the target_object_list // should be found
                if target_object_name == found_object_label
                        FOUND !!
```

(continued)

Writeup. Project 3. 3D perception

2. **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.**

When one object is found, I tried to skip comparing it for next found_object to save runtime (example, if book was founded in previous step, I will skip book from target_object_list). However, since multiple objects (glue, sticky note, and book) are considered as book, I need to check new_found_object with book several times because I can't guarantee the previous found object is actually book.

3. **Discussion for improving the performance**

As we can see from the 1st page, I have very high accuracy from my SVM training, however, I can't detect glue. It always consider glue as book. In other words, if I have a single object in the image, it can capture the object well. But if two or more objects are placed and some of the portion is hidden (glue in test 3), the accuracy is dropped.

a) Since the table position & camera is fixed (I assume the distance between the objects and camera is limited), I can use the size (number of pixel) of the object. It may help distinguishing between glue and book in Example 2
b) Currently, I generated all guess and pass to pr2_mover() and robo move them based on the input. But the image can be changed when one object is actually moved to the bin. Like Example 3, if book is captured first and moved to the bin correctly, now robot can have clear view of glue. To do that, I need additional features like below:
    a) Sorting object in the image based on the distance from the robot (may be use y position in their centroid)
    b) Pick one object and move to the bin
    c) Exit pr2_mover()
    d) Since while loop is operating, camera will feed new image to pcl_callback and generate the guessing list