Udacity Robotics Project 3 Writeup – Perception

Exercise 1. Filtering pipeline and RANSAC plane filtering

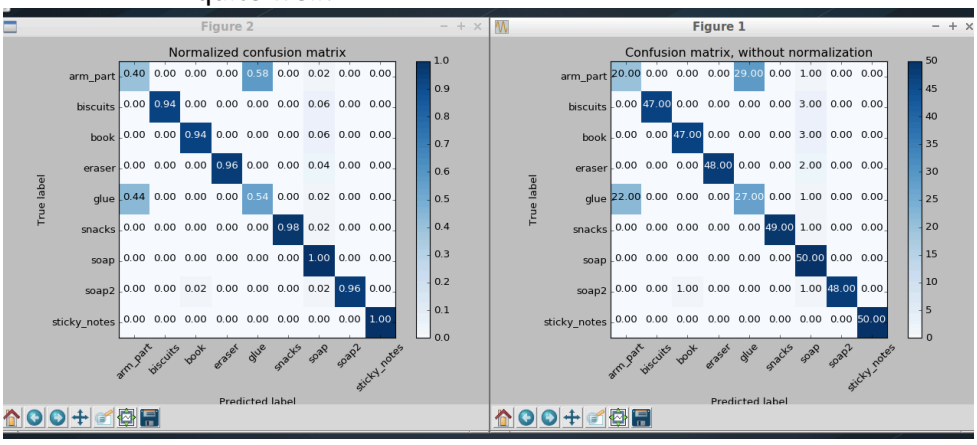
- Used filters below to take tabletop.pcd and output a filtered point cloud
    - VoxelGrid Downsampling Filter
    - PassThrough Filter
    - Extract Indices Filter
    - RANSAC Plane Fitting (Random Sample Consensus)
    - Statistical Outlier Removal Filter

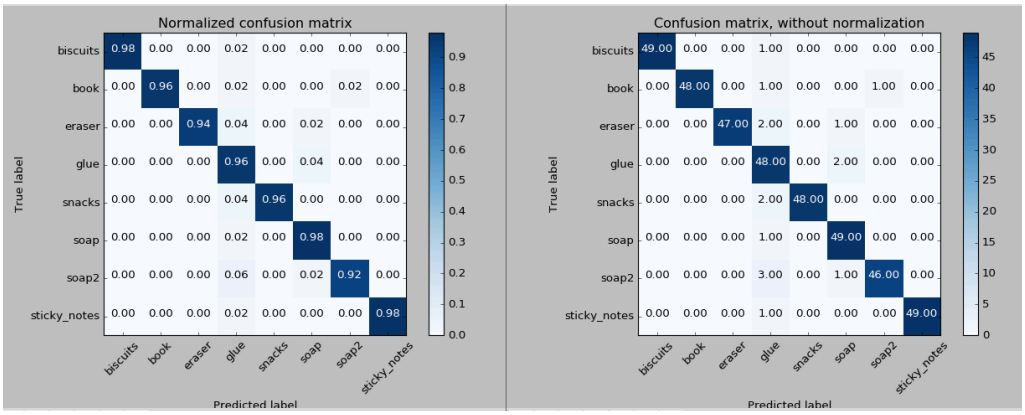
Exercise 2. Clustering for segmentation pipeline

- Subscribe to /sensor_stick/point_cloud topic
- Apply Euclidian clustering on filtered point clouds
- Publish clusters to separate pcl_table and pcl_objects topics

Exercise 3. Feature Extraction and SVM classifier training for object recognition

- Use object shape and color to classify and recognize objects, and apply a label to them
- capture_features.py
    - I tuned various parameters before settling on training my model on 50 poses per object (8 objects from 3 worlds, and 1 'arm_part' object), used a linear kernel on my SVM and achieved about 85% accuracy overall. The "arm_part' and 'glue' were frequently miscategorized as each other, so in the future I would exclude arm_part and train only on the 8 objects. The reason I used arm_part was because without it, my sensor was recognizing more objects than there existed in the world, as it was considering the robot arm to be an object.
    - In features.py, I tried different binning schemes before settling on the one I used. The range of color hist values is 0 – 256 and the range of normal hist is -1 to 1.
    - The first set of confusion matrices below is what I used initially, but it did not attain a passing submission in test_scene_3. Then, I retrained my model without arm_part, to get the 2nd set of confusion matrices, which identified the objects quite well!

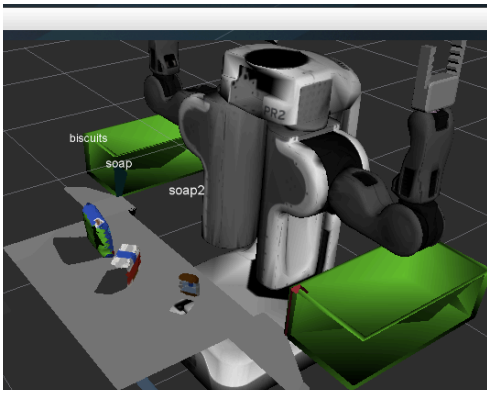


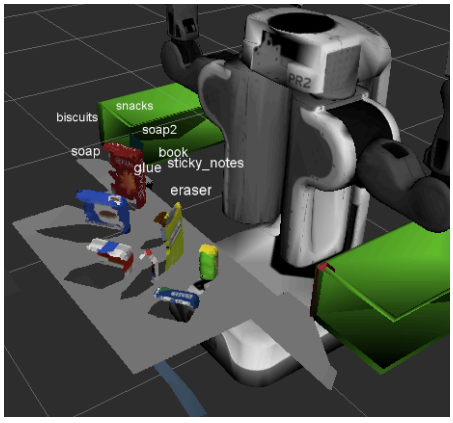


Project!

- The 3 exercises are collected in pcl_callback()
- In pr2_mover, I loop through the list of possible objects, dropbox parameters, and the pick list for each individual world. When an object matches that which is in the pick list, I assign the pick_pose and place_pose for it, and send that request through mack_yaml_dict. If it is not detected, I use default values of (0,0,0) for position and (0,0,0,0) for orientation.
- My terminal outputs the objects that were detected, along with “True” or “False” if it was correctly identified or not. The centroid position gets passed to a new topic which handles the pick and place operation.

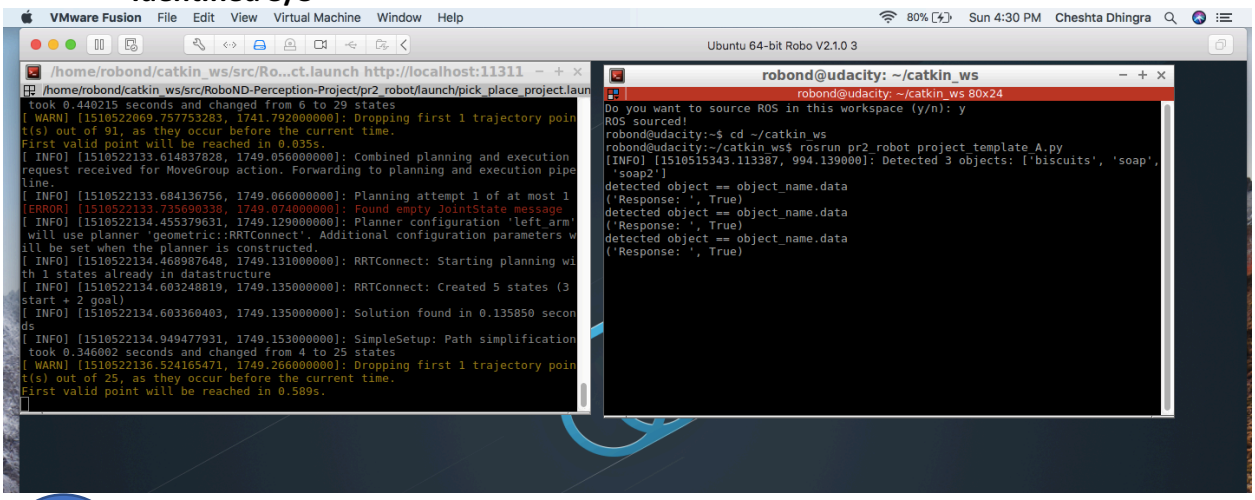
Success rate in scenes 1 and 3 (Sorry, forgot to take screenshot of scene 2!) :





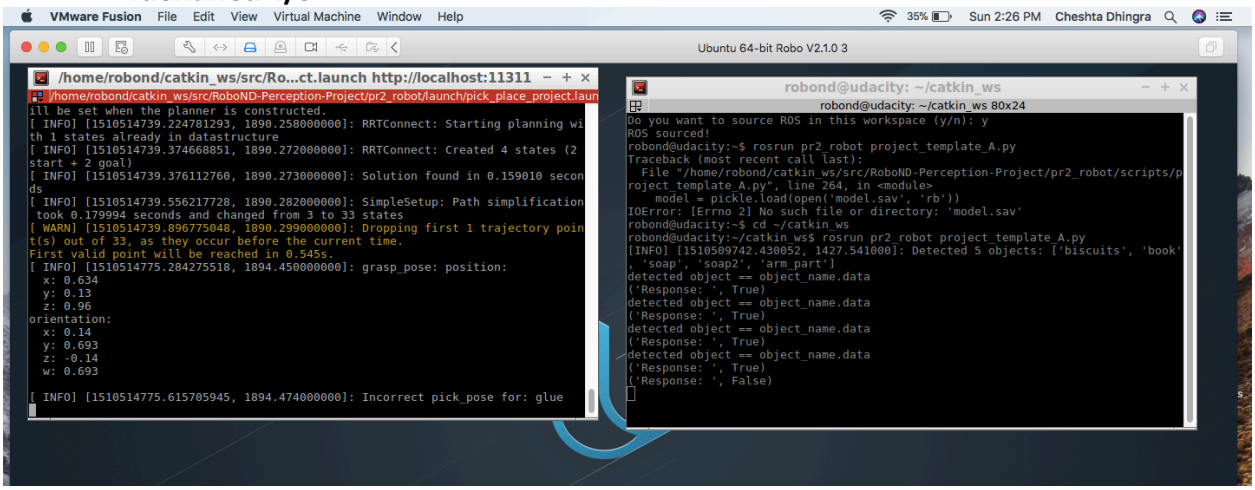
1

Identified 3/3



2

Identified 4/5



3

Identified 8/8 … Sorry, I forgot to take screenshot of terminal output ☹

Roadblocks

- Major hurdles in updating the pcl library, for which I had to download a new VM
- Found a silly error in my for loop which was causing me to recognize the first object and none of the rest, due to which I wasted a lot of time and energy…

Improvements for the future

- Collision avoidance during pick and place task
- Build an even better classifier by trying different kernels, training sizes, binning criteria