PROJECT SPECIFICATION

3D Perception

Writeup

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. The project repository contains a template writeup for this project that you can use as a guide and a starting point.	The writeup / README should include a statement and supporting figures / images that explain how each rubric item was addressed, and specifically where in the code each step was handled. The writeup should include a discussion of what worked, what didn't and how the project implementation could be improved going forward.	This document addresses all the rubric points. I'll point out where all the code is and where are the files are located.

Exercise 1, 2 and 3 Pipeline Implemented

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
Complete Exercise 1 steps. Pipeline for filtering and RANSAC plane fitting implemented.	The `pcl_callback()` function within the template Python script has been filled out to include filtering and RANSAC plane fitting. Not required, but to help your reviewer consider adding screenshots of output at different steps in your writeup with brief explanations.	RANSAC.py is at RoboND-Perception-Exercises/Exercise-1/RANSAC.py in the repo.
Complete Exercise 2 steps: Pipeline including clustering for segmentation implemented.	Steps for cluster segmentation have been added to the `pcl_callback()` function in the template Python script. Not required, but to help your reviewer consider adding screenshots of output at different steps in your writeup with brief explanations.	The pcl_callback() function is filled in segmentation.py at RoboND-Perception-Exercises/Exercise-2/sensor_stick/scripts/segmentation.py in repo as part of Excercise-2
Complete Exercise 3 Steps. Features extracted and SVM trained. Object recognition implemented.	Both `compute_color_histograms()` and `compute_normal_histograms()` functions have been filled out and SVM has been trained using `train_svm.py`. Please provide a snapshot of your normalized confusion matrix (output from `train_svm.py` in your writeup / README. Object recognition steps have been implemented in the `pcl_callback()` function within template Python script. Not required, but to help your reviewer consider adding screenshots of output at different steps in your writeup with brief explanations.	train_svm.py has been used to train the SVM and is under sensor_stick/scripts. For the final project I have used pr2_train_svm_world1.py and pr2_capture_features_world1.py from this directory.

Pick and Place Setup

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
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.	You can add this functionality to your already existing ros node or create a new node that communicates with your perception pipeline to perform sequential object recognition. Save your PickPlace requests into `output_1.yaml`, `output_2.yaml`, and `output_3.yaml` for each scene respectively. Add screenshots in your writeup of output showing label markers in RViz to demonstrate your object recognition success rate in each of the three scenarios. **Note: for a passing submission, your pipeline must correctly identify 100% of objects in `test1.world`, 80% (4/5) in `test2.world` and 75% (6/8) in `test3.world`.**	The recognition for all three tasks is done successfully. World1: 3/3 (100%) world2: 5/5(100%) world3: 7/8(87.5%) The output_(1/2/3).yamml files are present in the main repository As mentioned earlier, for capturing the features and training I have used pr2_train_svm_world1.py and pr2_capture_features_world1.py from sensorstick/scripts from the repo. The recognition task is done using project_template.py under pr2_robot/scripts.