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

File : src\RoboND-Perception-Project\pr2_robot\scripts\project_template.py

Lines : 65 – 136

- → Voxel grid filter – leaf size 0.01
- → Passthrough filter along z axis - axis_min = 0.65 axis_max = 1.1
- → Passthrough filter along y axis - axis_min = -0.5 axis_max = 0.5
- → RANSAC Plane segmentation - max_distance = 0.01

+

- → Outlier removal – threshold = 0.1

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

File : src\RoboND-Perception-Project\pr2_robot\scripts\project_template.py

Lines : 142 – 179

- → Euclidean Clustering – cluster tolerance = 0.02, min size = 30, max size = 500
- → Identify number of objects
- → Convert to ROS type of message
- → Publish

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

File : src\sensor_stick\scripts\capture_features.py

Lines : all

- → List models that we want to capture for training
- → Generate 100 examples for each model
- → Compute histogram
- → Normalize
- → Stack histogram with normalized one together
- → Save on disc

File : src\sensor_stick\scripts\train_svm.py

Lines : all

- → Unmodified file. Saved information is getting picked up, SVM is trained with a data, output is saved model and 2 graphs

File : src\RoboND-Perception-Project\pr2_robot\scripts\project_template.py

Lines : 187-228

- → Prepare features in the same way as during training of the model
- → Predict class
- → Convert to ROS message
- → Publish

## 4. 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.

File : src\RoboND-Perception-Project\pr2_robot\scripts\project_template.py

Lines : 308-378

- → Set scene number
- → Set object name
- → Select proper arm based on object
- → Select proper dropbox for an arm
- → Set the target pose (x,y,z coordinates)
- → Reformat data to write log
- → Write the yaml

Files for world 1:

- → model-1.sav – model for world 1 trained on world 1 objects only
- → output-1.yaml – output of the pipeline 1

Files for world 2:

- → model-2.sav – model for world 1 trained on world 2 objects only
- → output-2.yaml – output of the pipeline 2

Files for world 3:

- → model-3.sav – model for world 1 trained on world 3 objects only
- → output-3.yaml – output of the pipeline 3