Project: Perception Pick & Place

Writeup / README

Exercise 1, 2 and 3 pipeline implemented

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

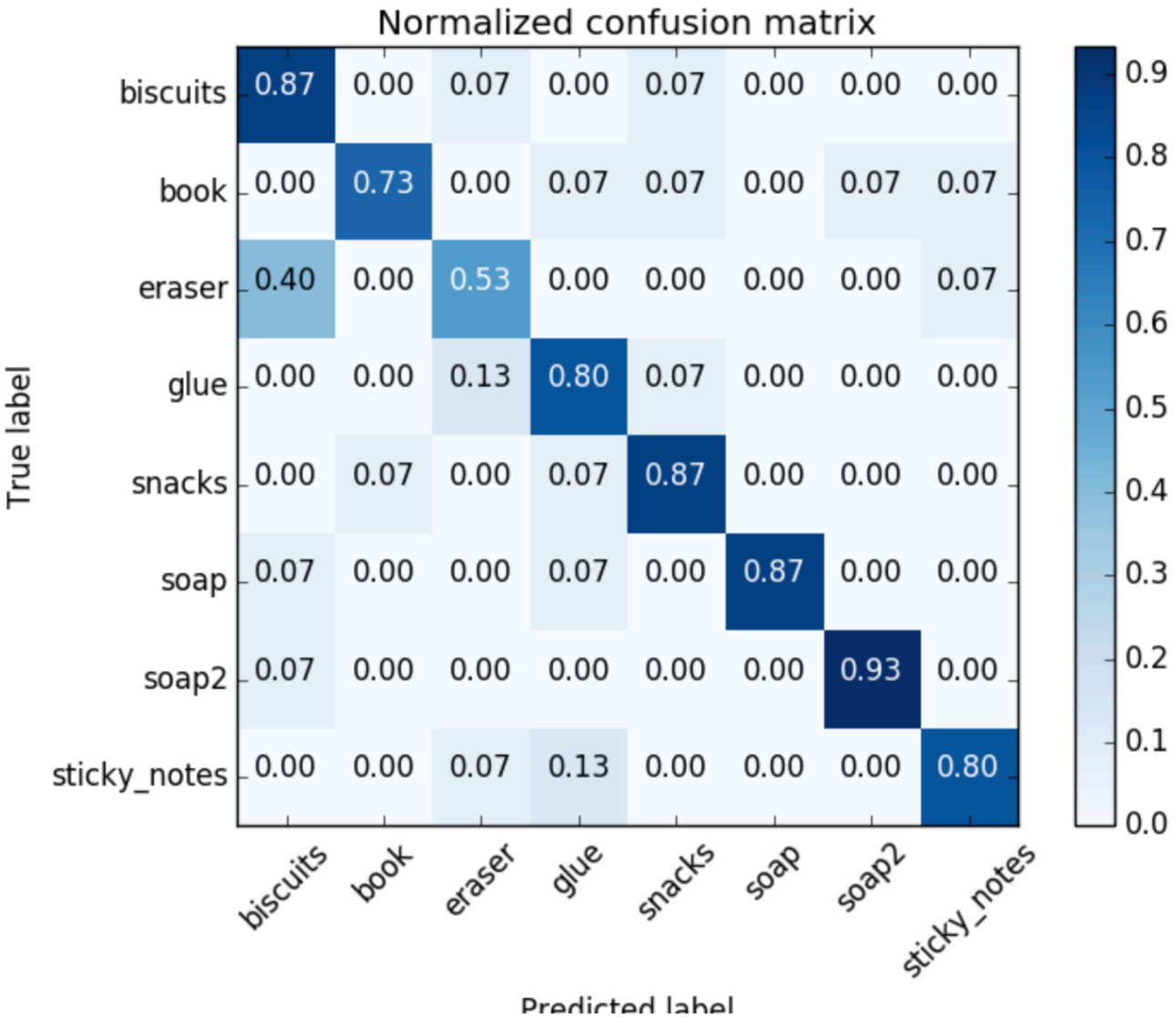
`RACSAC.py` implements this excercise. Most parts of the code from it are also re-used in the rest of the project. During the tests in the project, since the dropboxes were in the range of the RGB-D sensor, that caused them to be considered as objects requiring classification. A workaround for this was to restrict the y-axis using the passthrough filter so only what needs to be classified is used in the RANSAC plane fitting.

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

This is done in `segmentation.py` and also used in the later code. Pretty straightforward. Values in the project were obtained by "trial-and-error".

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

Exercise 3 is coded in `object_recognition.py`. In the project, a total of 8 different items are used: soap, soap2, book, biscuits, eraser, glue, sticky_notes and snacks. 15 random poses of each item were used in generating the training set. Training an SVM model on this training set, and then testing it resulted in approximately 80% accuracy:

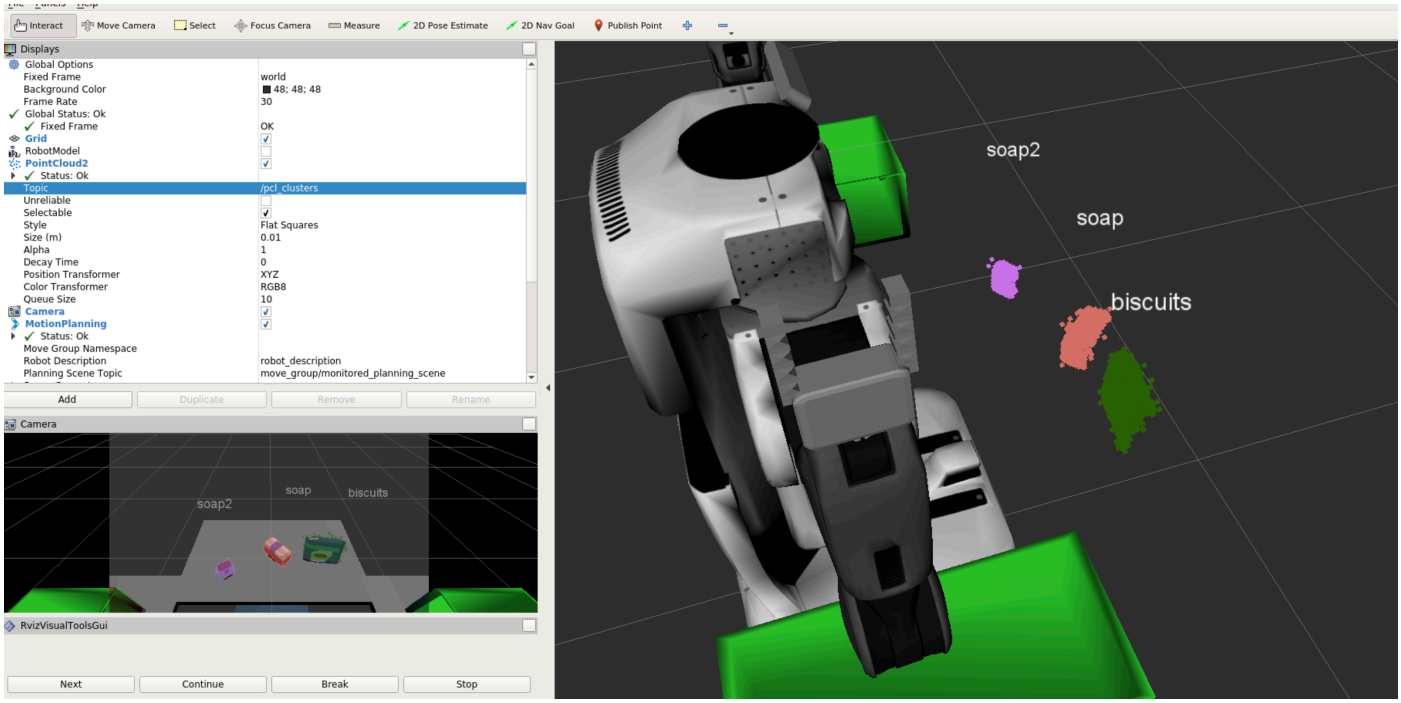


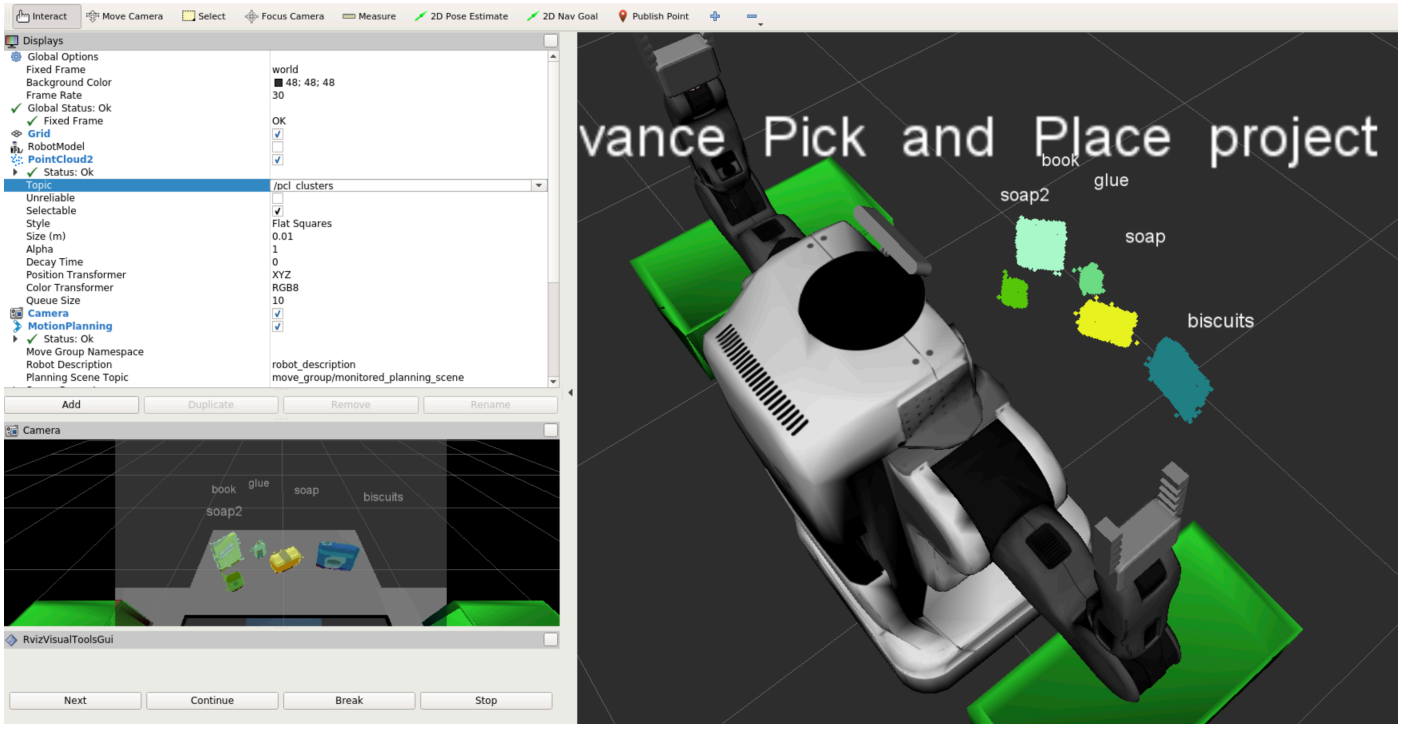
The model was trained specifically for each of the given worlds (world1's model only uses 3 categories, world2's model - 5 categories and world3's model - all 8 categories). While this demonstratres better results for the individual worlds (since there are less false positives), I found that the use of a model with all categories worked decently well and used this model version for the final submission.

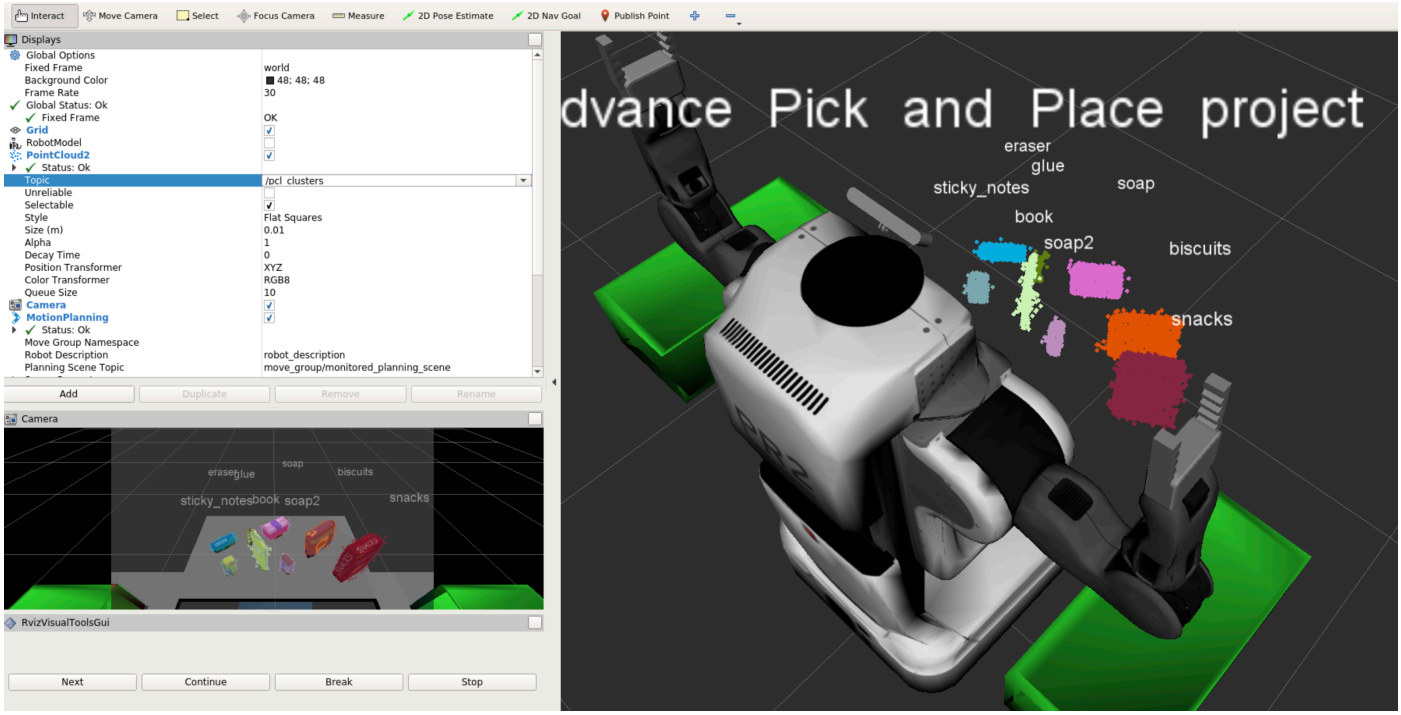
Pick and Place Setup

1. 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.

The initial part of the project was mostly just re-use of the code from the exercises and tweaking of the values in the filters accordingly. After the tweaking, the following RViz screenshots were taken for each of the given worlds.


Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame
world
Background Color
48; 48; 48
Frame Rate
30
Global Status: Ok
Fixed Frame
OK
Grid
RobotModel
PointCloud2
Status: Ok
Topic
/pcl_clusters
Unreliable
Selectable
Style
Flat Squares
Size (m)
0.01
Alpha
1
Decay Time
0
Position Transformer
XYZ
Color Transformer
RGB8
Queue Size
10
Camera
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planning_scene
Add
Duplicate
Remove
Rename
Camera
soap2
soap
biscuits
RvizVisualToolsGui
Next
Continue
Break
Stop
soap2
soap
biscuits


Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame
world
Background Color
48; 48; 48
Frame Rate
30
Global Status: Ok
Fixed Frame
OK
Grid
RobotModel
PointCloud2
Status: Ok
Topic
/pcl_clusters
Unreliable
Selectable
Style
Flat Squares
Size (m)
0.01
Alpha
1
Decay Time
0
Position Transformer
XYZ
Color Transformer
RGB8
Queue Size
10
Camera
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_planning_scene
Add
Duplicate
Remove
Rename
Camera
book
glue
soap
biscuits
soap2
RvizVisualToolsGui
Next
Continue
Break
Stop
vance Pick and Place project
book
glue
soap2
soap
biscuits



While the results on thsse screenshots seem to demonstrate the 100% accurate recognition, world 3 occasionally had the `sticky_notes` being recognised as `book` or `eraser`. One way this could be improved is to simply make the training data set bigger.