

# Project: Perception Pick & Place

Brett Gleason

August 26, 2017

The rubric for this project can be found at the following URL:  
<https://review.udacity.com/#!/rubrics/1067/view>  
I will consider the rubric points individually and describe how I addressed each point in my implementation.

## Required Steps for a Passing Submission:

1. Extract features and train an SVM model on new objects (see 'pick\_list\_\*.yaml' in '/pr2\_robot/config/' for the list of models you'll be trying to identify).
2. Write a ROS node and subscribe to '/pr2/world/points' topic. This topic contains noisy point cloud data that you must work with.
3. Use filtering and RANSAC plane fitting to isolate the objects of interest from the rest of the scene.
4. Apply Euclidean clustering to create separate clusters for individual items.
5. Perform object recognition on these objects and assign them labels (markers in RViz).
6. Calculate the centroid (average in x, y and z) of the set of points belonging to that each object.
7. Create ROS messages containing the details of each object (name, pick\_pose, etc.) and write these messages out to '.yaml' files, one for each of the 3 scenarios ('test1-3.world' in '/pr2\_robot/worlds/'). See the example 'output.yaml' for details on what the output should look like.
8. Submit a link to your GitHub repo for the project or the Python code for your perception pipeline and your output '.yaml' files (3 '.yaml' files, one for each test world). You must have correctly identified 100% of objects from 'pick\_list\_1.yaml' for 'test1.world', 80% of items from 'pick\_list\_2.yaml' for 'test2.world' and 75% of items from 'pick\_list\_3.yaml' in 'test3.world'.
9. Congratulations! You're done!

—

## Writeup / README

---

1. 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.

You're reading it!

## Exercise 1, 2 and 3 pipeline implemented

---

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

The objective of exercise 1 was to take a simulated scene with several objects on a table and create two separate point cloud files: the first containing the table surface and the second containing the objects on the table. To accomplish this, several filters were used as well as the random sample consensus (or RANSAC) algorithm.

### Voxel Grid Downsampling

The starting point for this exercise is a point cloud file from a simulated RGB-D camera. The point cloud from the RGB-D camera is very dense, meaning any operations on it will be very computationally intense.

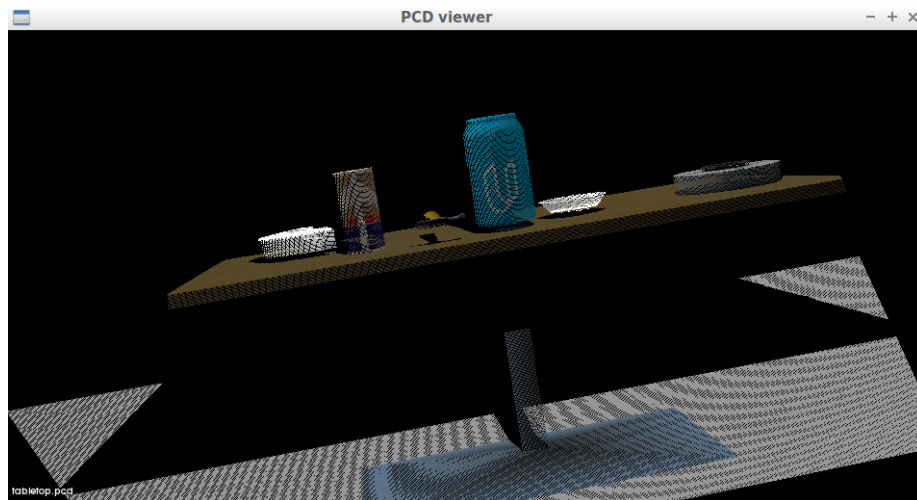


Figure 1: Point cloud file from RGB-D camera

Voxel grid downsampling reduces the density of the point cloud, allowing for faster computations without any loss in the ability for the point cloud data

to be used for object recognition later in the perception pipeline. A voxel can be thought of as a three dimensional analog to a pixel. The input point cloud can be divided into a voxel grid by dividing it into cubic regions. To perform the voxel grid downsampling, the points within each cubic region (voxel) are replaced with a single point that represents the average of the points contained within the voxel. By choosing the size of the voxel appropriately, the downsampled point cloud will provide a good approximation of the original point cloud while allowing faster computation in later steps. For this project, voxel grid downsampling is accomplished using the python PCL library. A VoxelGrid filter object is created from the input point cloud and a filter() function can be run on this object after the size of the voxel is set.

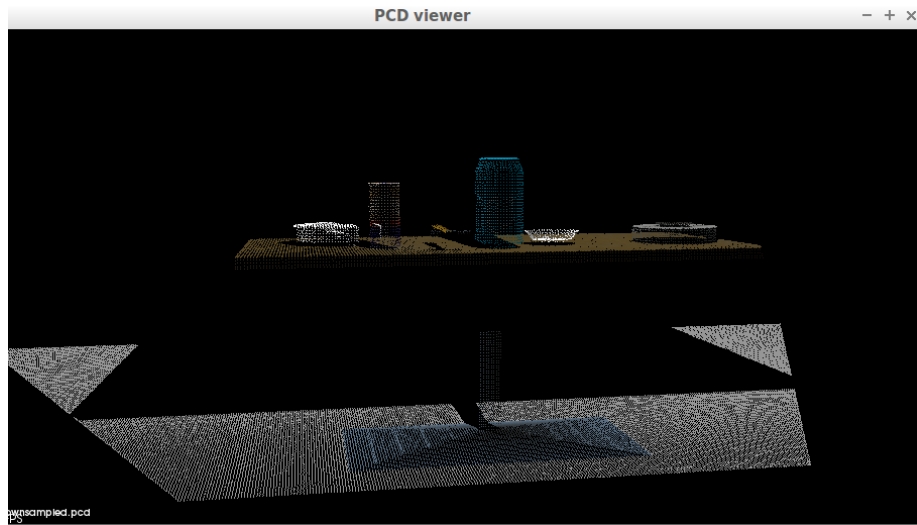


Figure 2: Point cloud file after voxel grid downsampling

### Pass Through Filtering

Because the robot and the table are both stationary for this project,

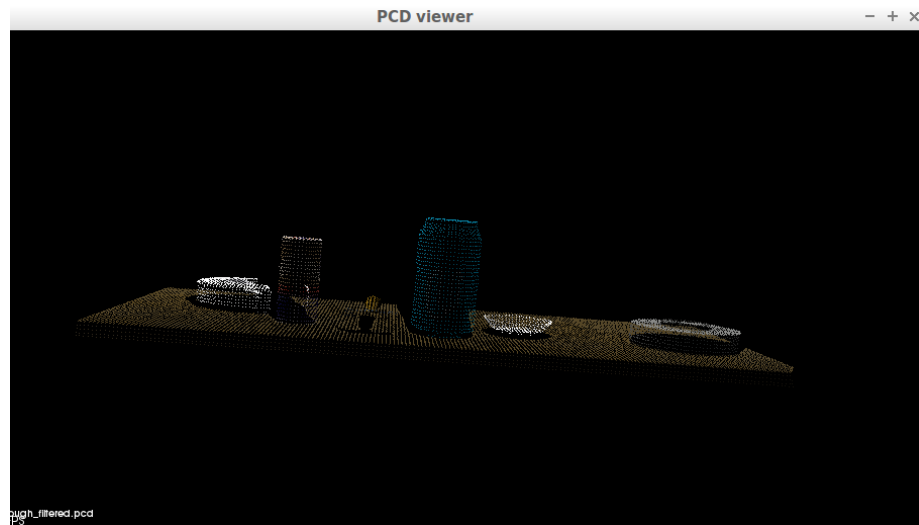


Figure 3: Point cloud file after passthrough filtering

## RANSAC



Figure 4: RANSAC inliers

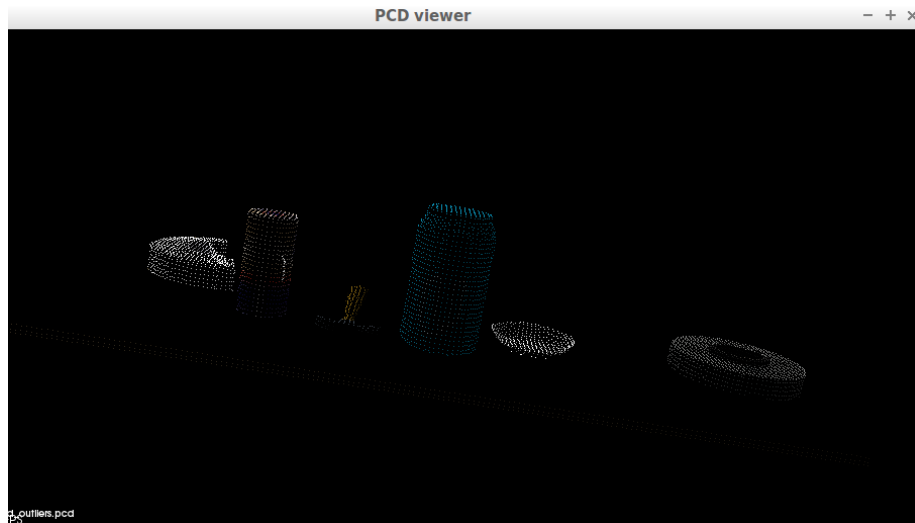


Figure 5: RANSAC outliers

2. Complete Exercise 2 steps: Pipeline including clustering for segmentation implemented.
3. Complete Exercise 3 Steps. Features extracted and SVM trained. Object recognition implemented.

## 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.