Project: Perception Pick & Place

Brett Gleason

August 27, 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. 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. 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. 3. Use filtering and RANSAC plane fitting to isolate the objects of interest
   from the rest of the scene.

4. 4. Apply Euclidean clustering to create separate clusters for individual
   items.

5. 5. Perform object recognition on these objects and assign them labels
   (markers in RViz).

6. 6. Calculate the centroid (average in x, y and z) of the set of points
   belonging to that each object.

7. 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. 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. 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.
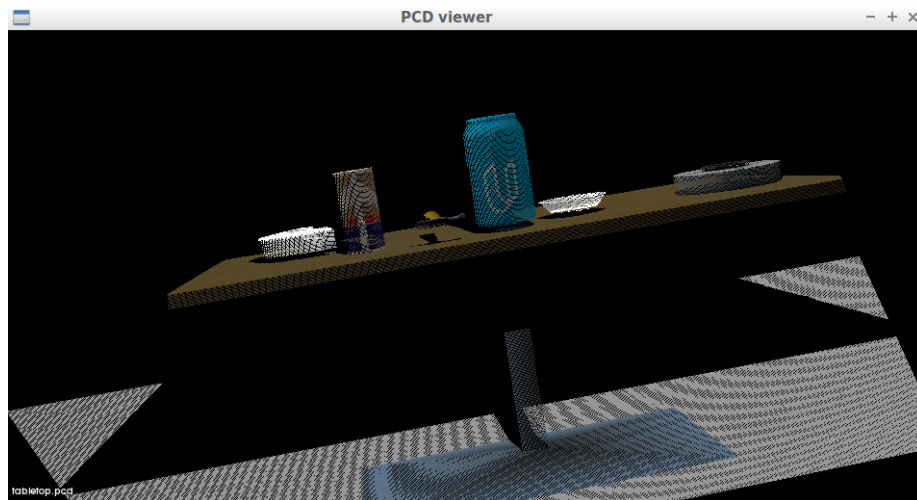

Figure 1: Point cloud file from RGB-D camera

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 down-sampled 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.
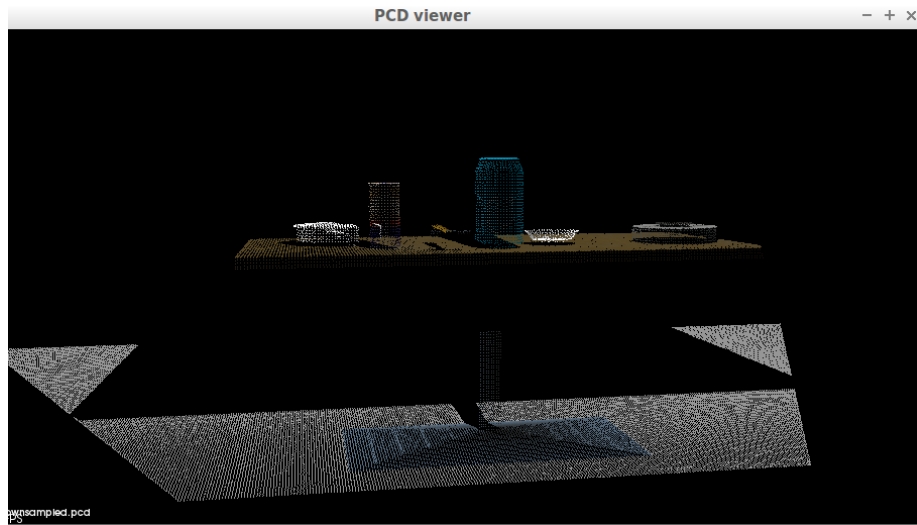

Figure 2: Point cloud file after voxel grid downsampling

Pass Through Filtering

Because the robot and the table are both stationary for this project it is easy to filter out parts of the point cloud that are not relevant. For this exercise a region on the Z-axis was selected such that only the table and the objects on it would be able to "pass through" the filter as shown in figure 3.
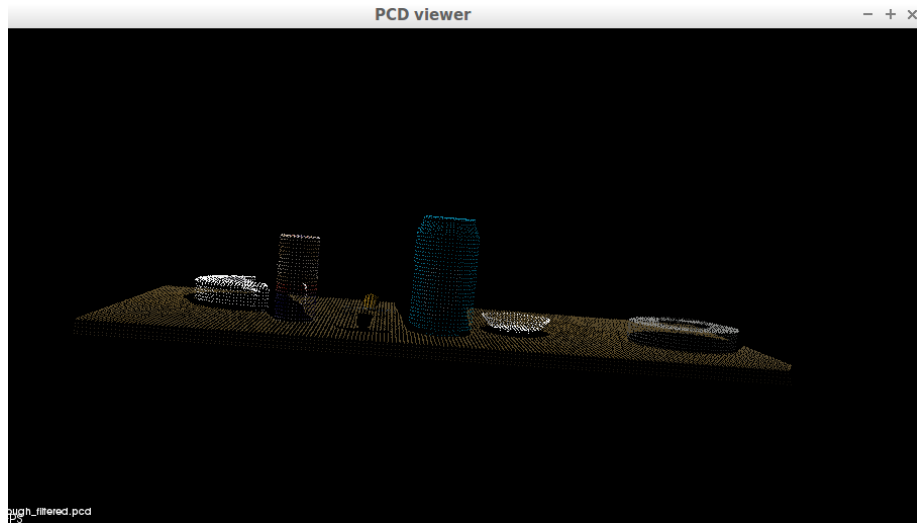
3

Figure 3: Point cloud file after pass through filtering

RANSAC

Finally, random sample consensus (or RANSAC) is an algorithm that can be used to remove the table from the point cloud, leaving only the objects of interest behind. RANSAC determines whether each point in the cloud belongs to a particular model. If the point fits the model it is classified as an inlier, if it doesn't fit it is an outlier. For this exercise a plane was used as the model, resulting in the points belonging to the table being classified as inliers to the plane, while the points belonging to the objects on the table were classified as outliers. At this point the objective of the exercise is accomplished, the input point cloud from the RGB-D camera has been separated into a point cloud containing the table surface and a point cloud containing the objects of interest.
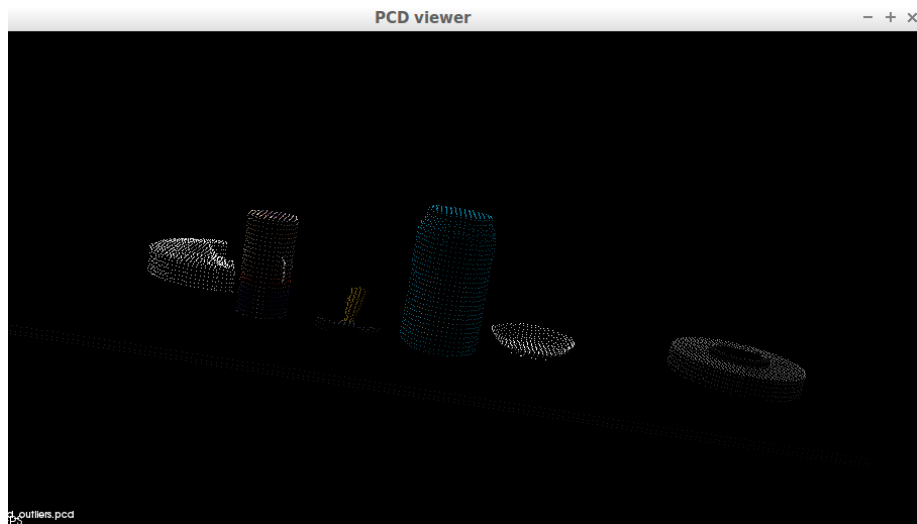
Figure 4: RANSAC inliers

Figure 5: RANSAC outliers

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

For the second exercise the objective was to take the filtered point cloud containing the objects as shown in figure 5 and

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.