Project: Perception Pick & Place

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! Your Done!

Writeup / README

You're reading it!

Hi. I am Miguel Colmenares, engineer in electronics and i want to tell you about my experience in this project.

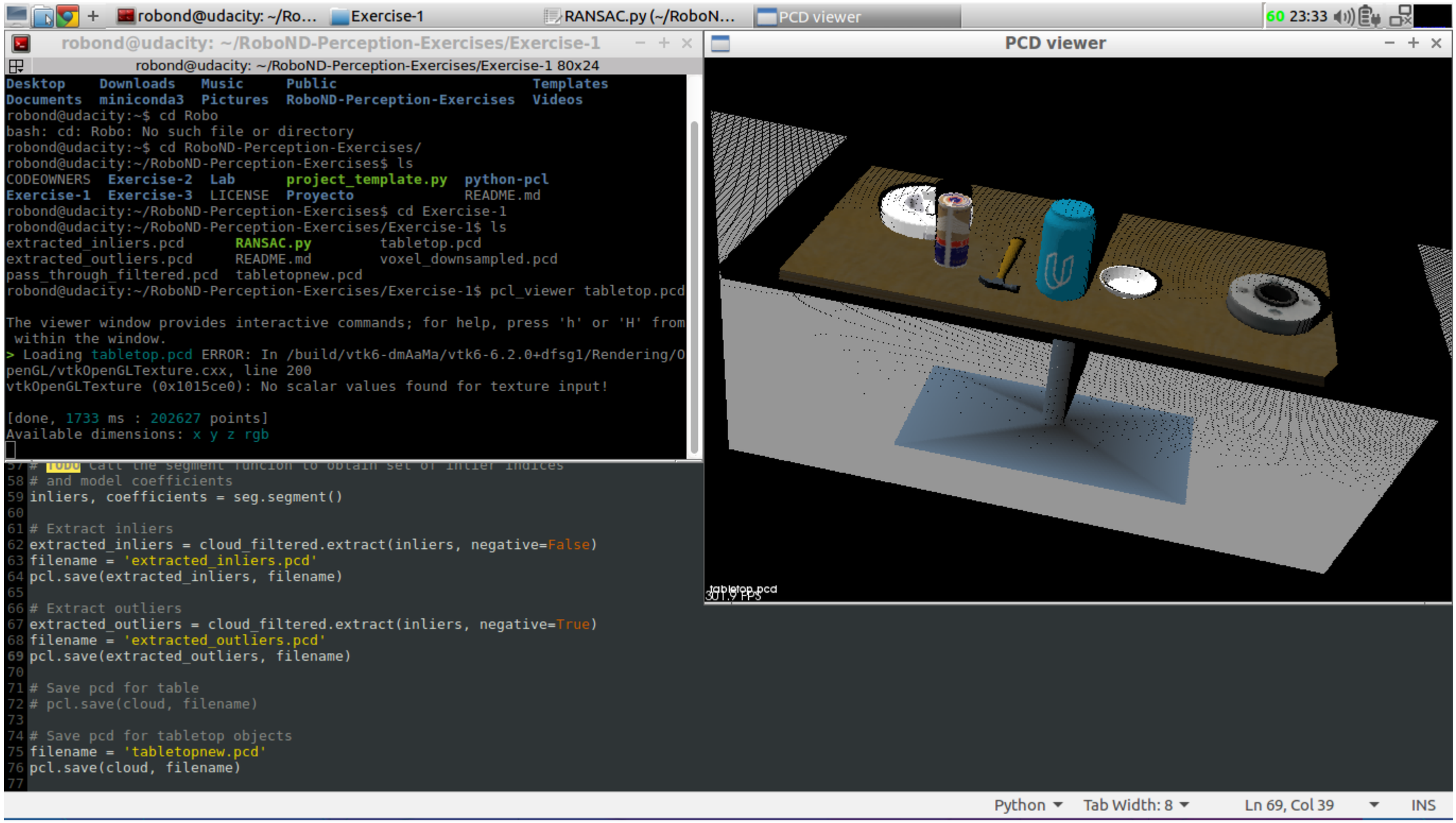
And part of this project i shared it in my instagram: @uchobogens

Kinematic Analysis

Exercise 1, 2 and 3 pipeline implemented

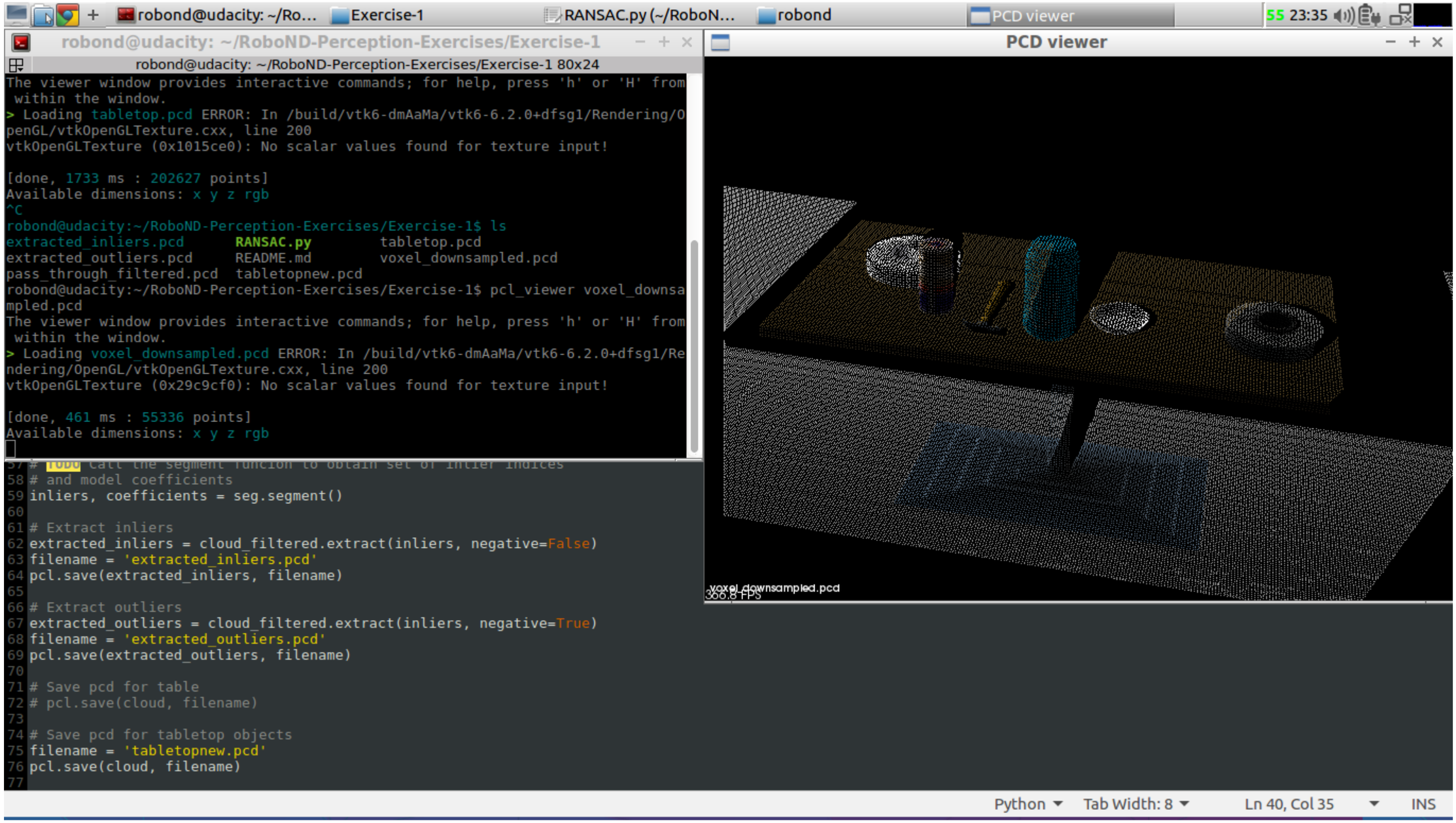
1. Completed Exercise 1 steps:

[Imagen: tabletop]



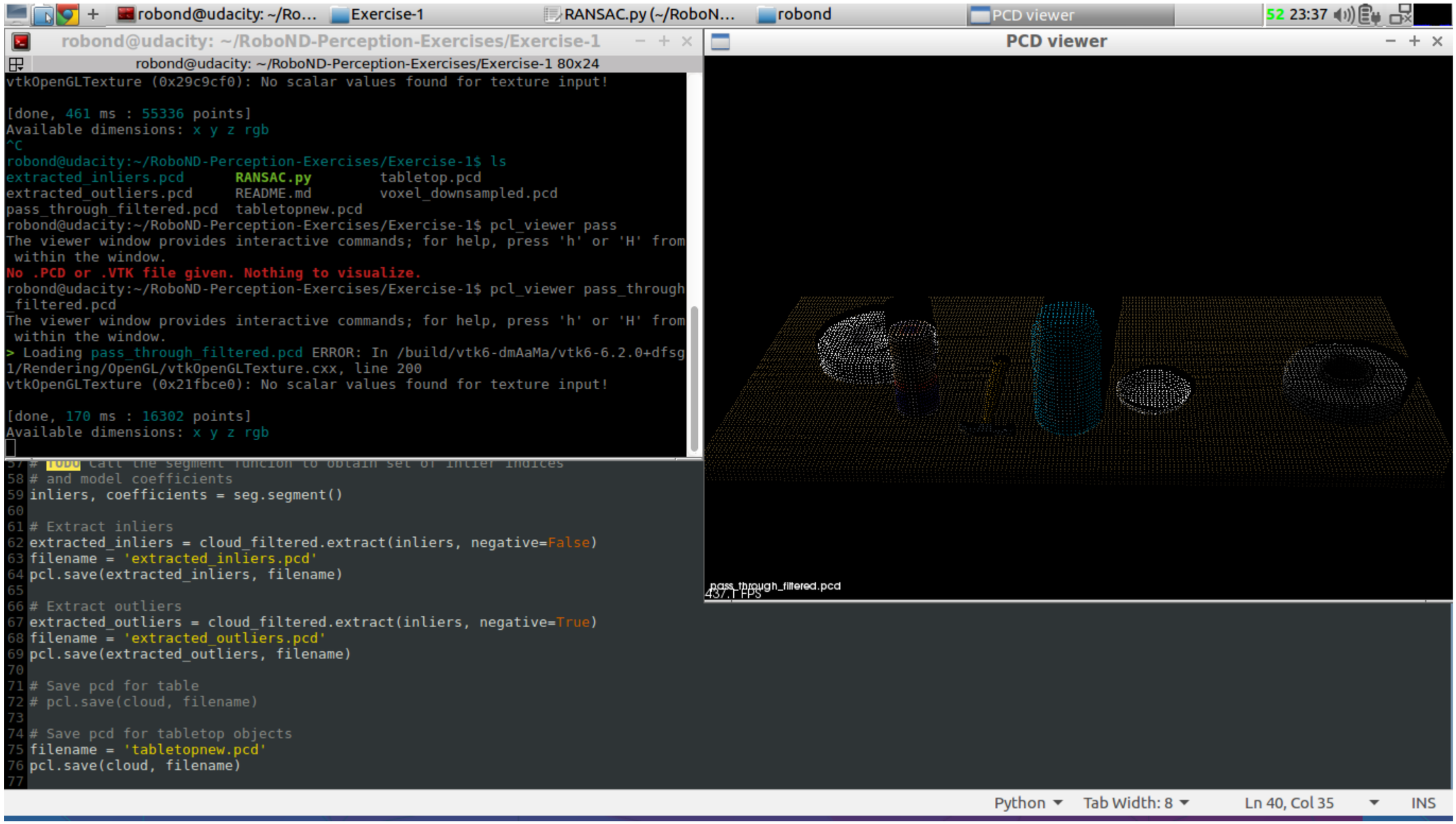
In this image I have the environment, the table and the objects in high resolution.

[Imagen: voxeldown]



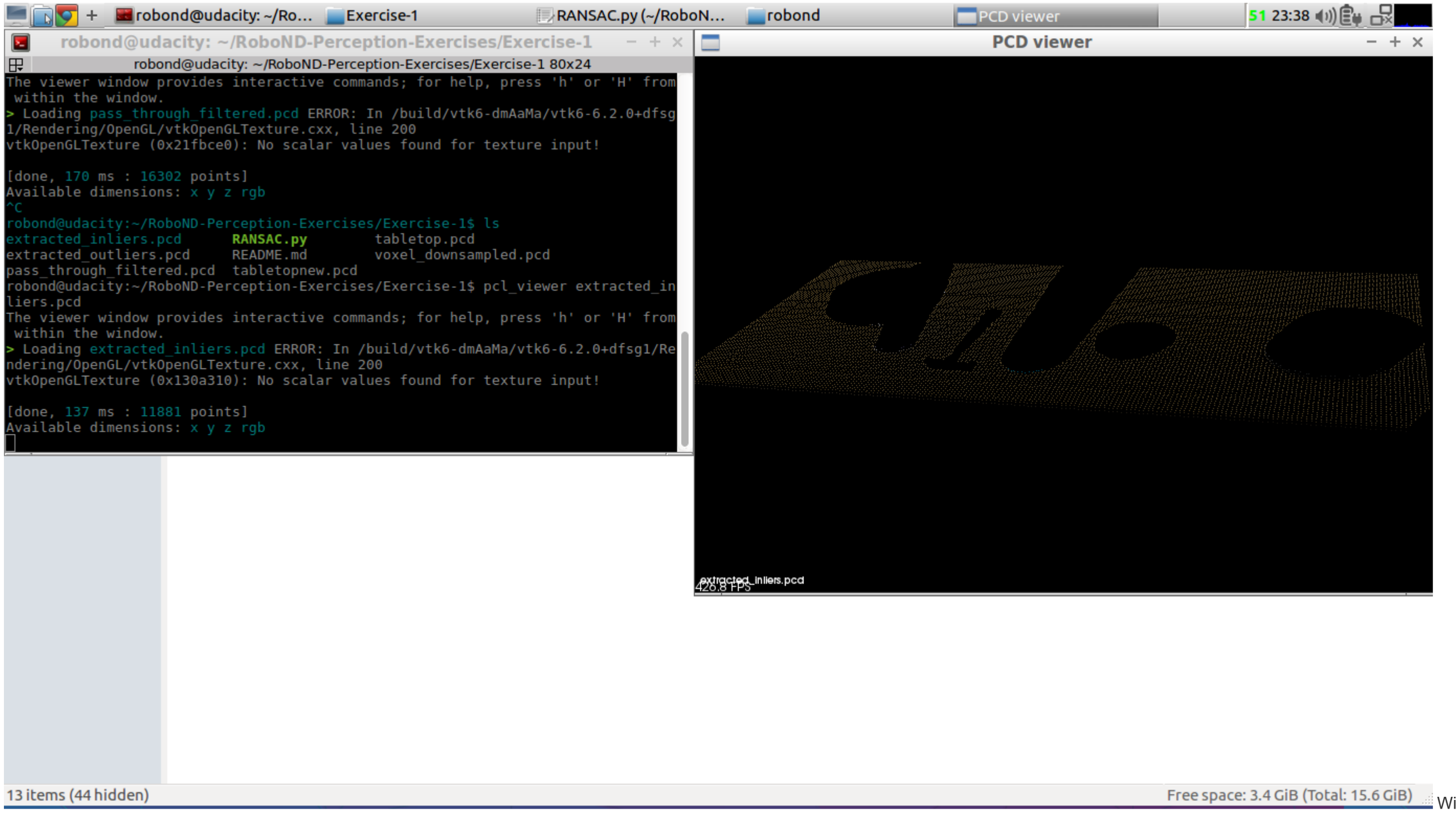
A voxel filter was applied to lower the resolution of the captured image.

[Imagen: passthrough]



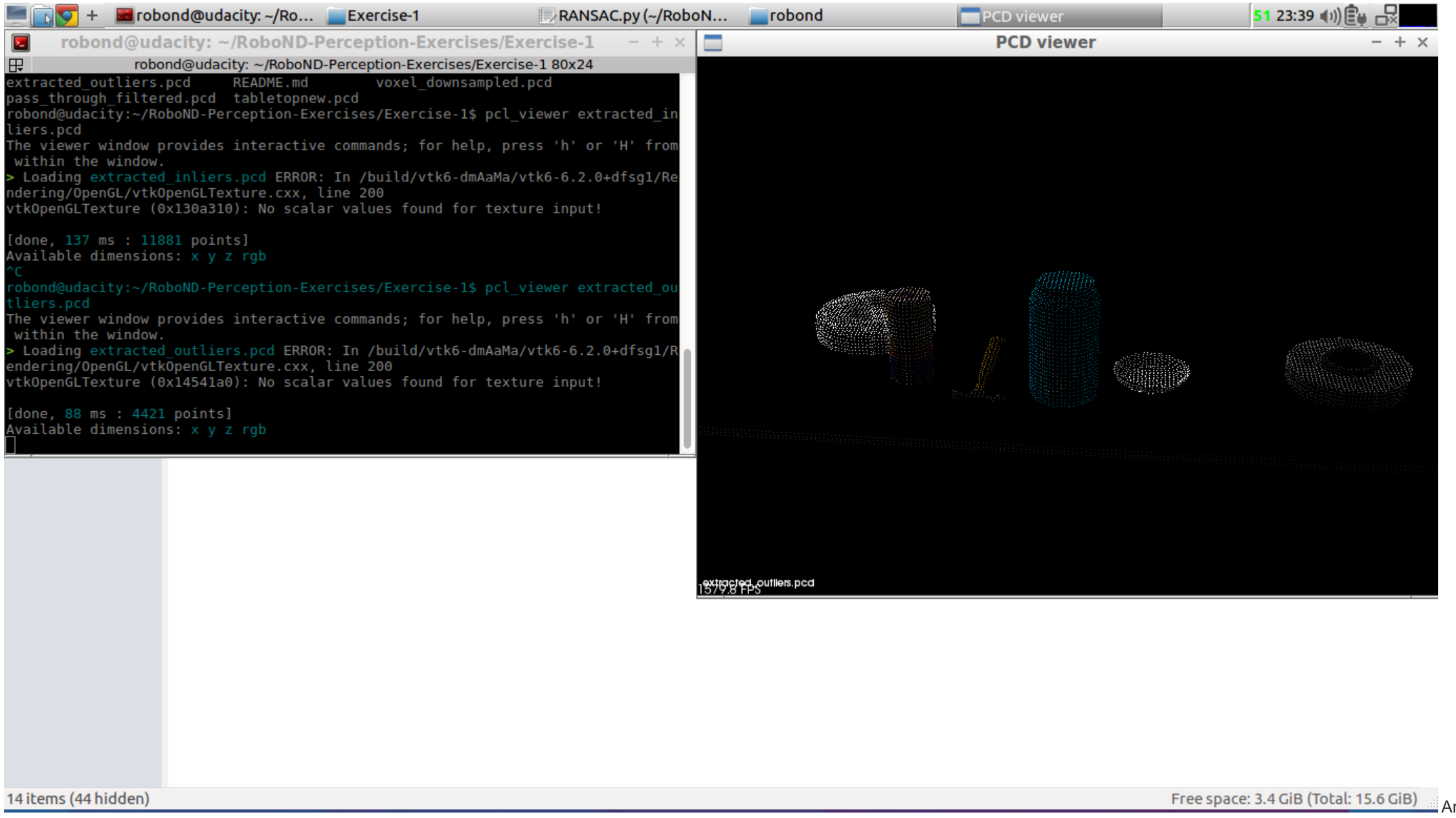
With the pass_throsgh filter i only get the table with the objects.

[Imagen: tableinliers]



With the extracted_inliers filter i only get the table.

[Imagen: objecetliers]

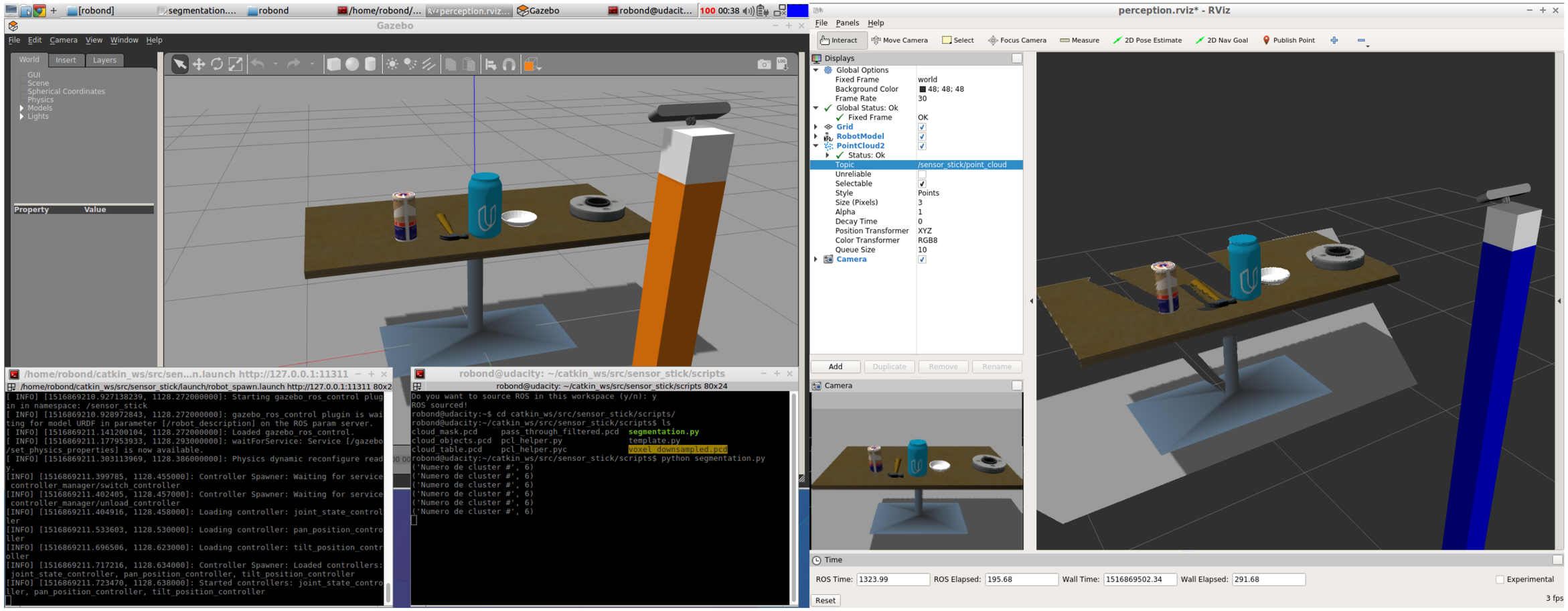


And with the extracted_outliers filter i only get the objects.

2. Completed Exercise 2 steps:

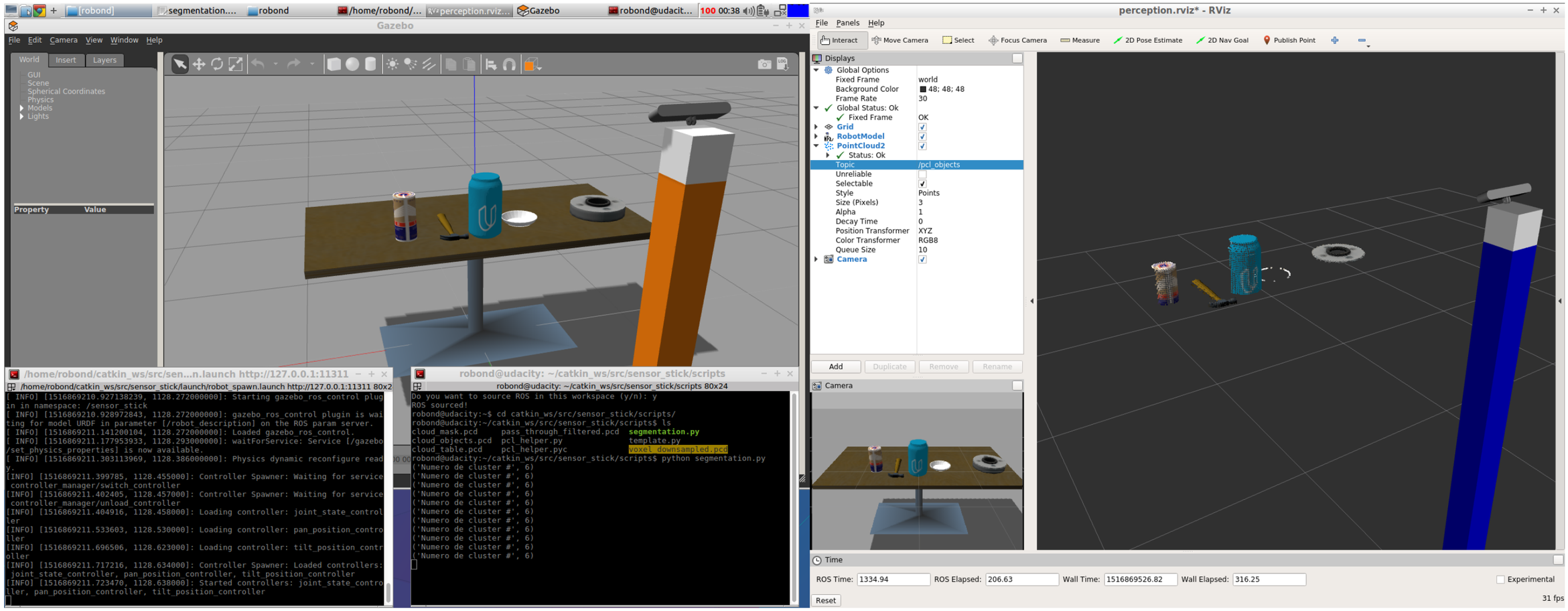
Now that I get only the objects, I can identify each object in a cluster and assign it a color.

[Imagen: pointcloud]



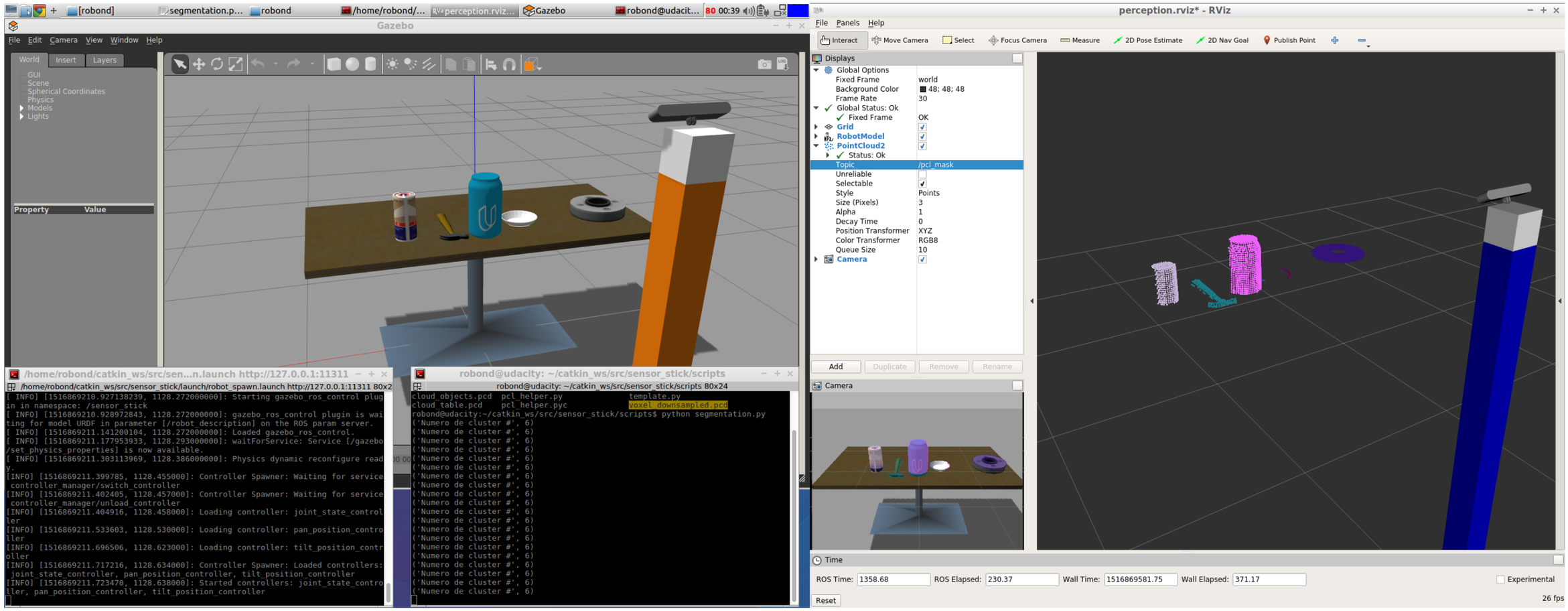
Here you can appreciate the environments in Gazebo and Rviz.

[Imagen: pclobjects]



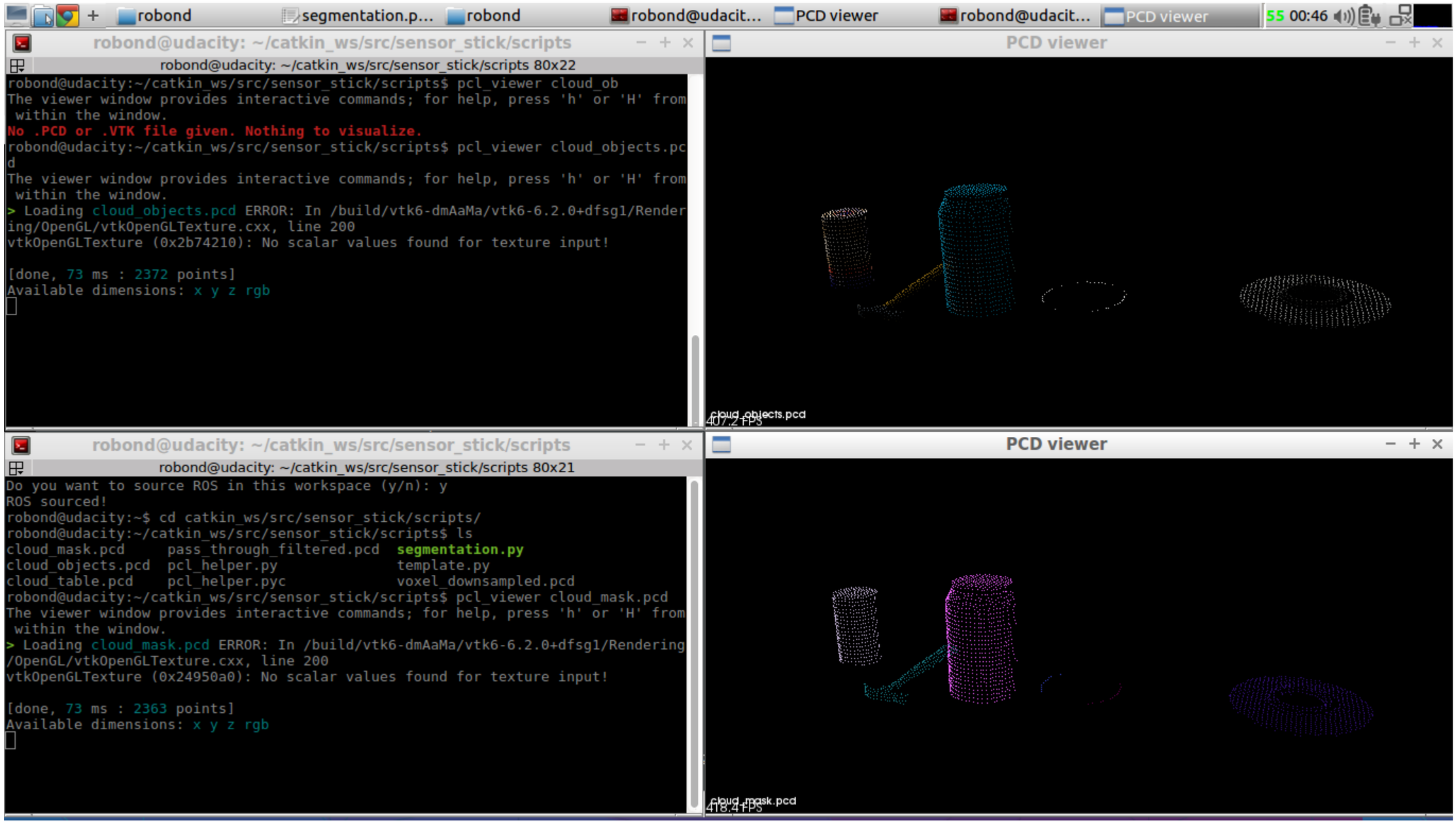
With the Topic: "/pcl_objects" you can appreciate only the objects.

[Imagen: pclmask]



With the Topic: "/pcl_mask" you can appreciate only the cloud_cluster mask.

[Imagen: objcYmask]

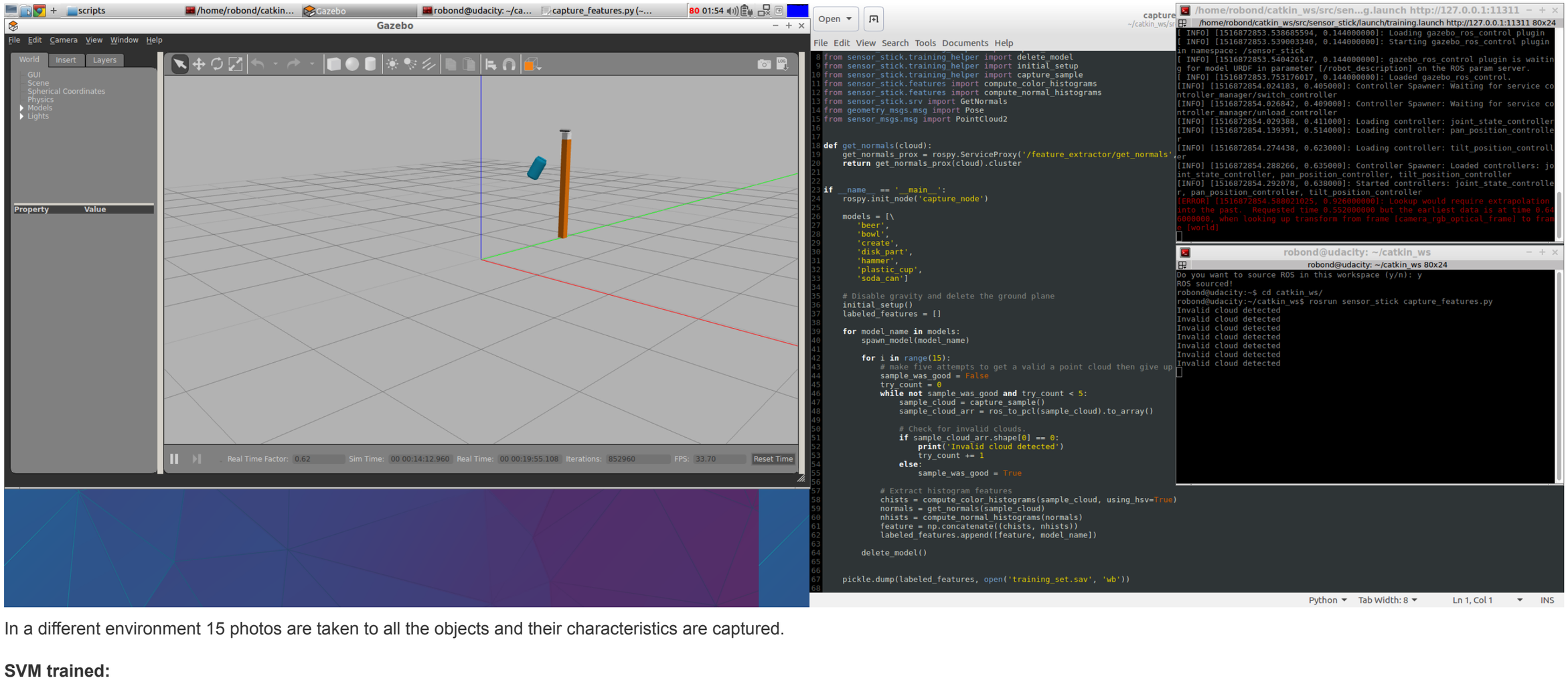


Here you can better appreciate the objects and the masks.

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

Features extracted:

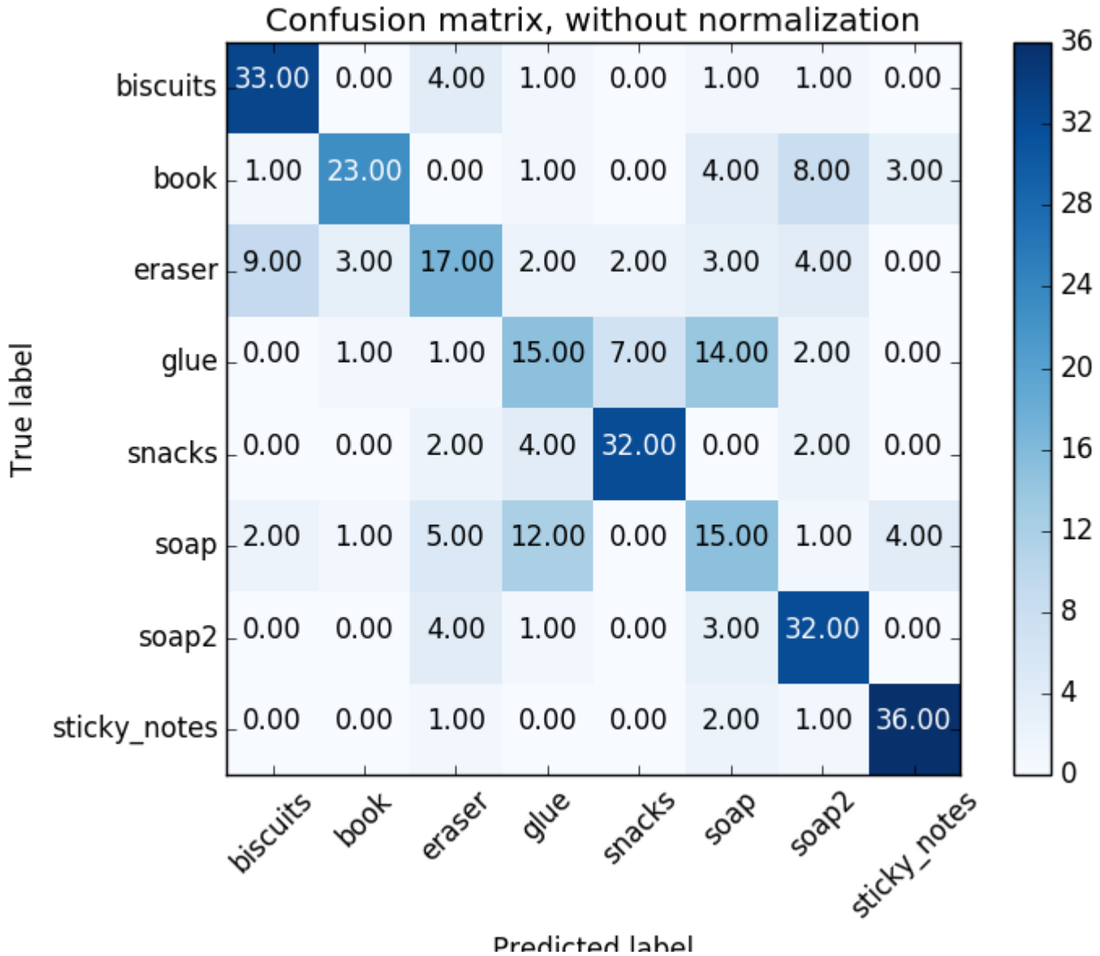
[Imagen: capturefeature]



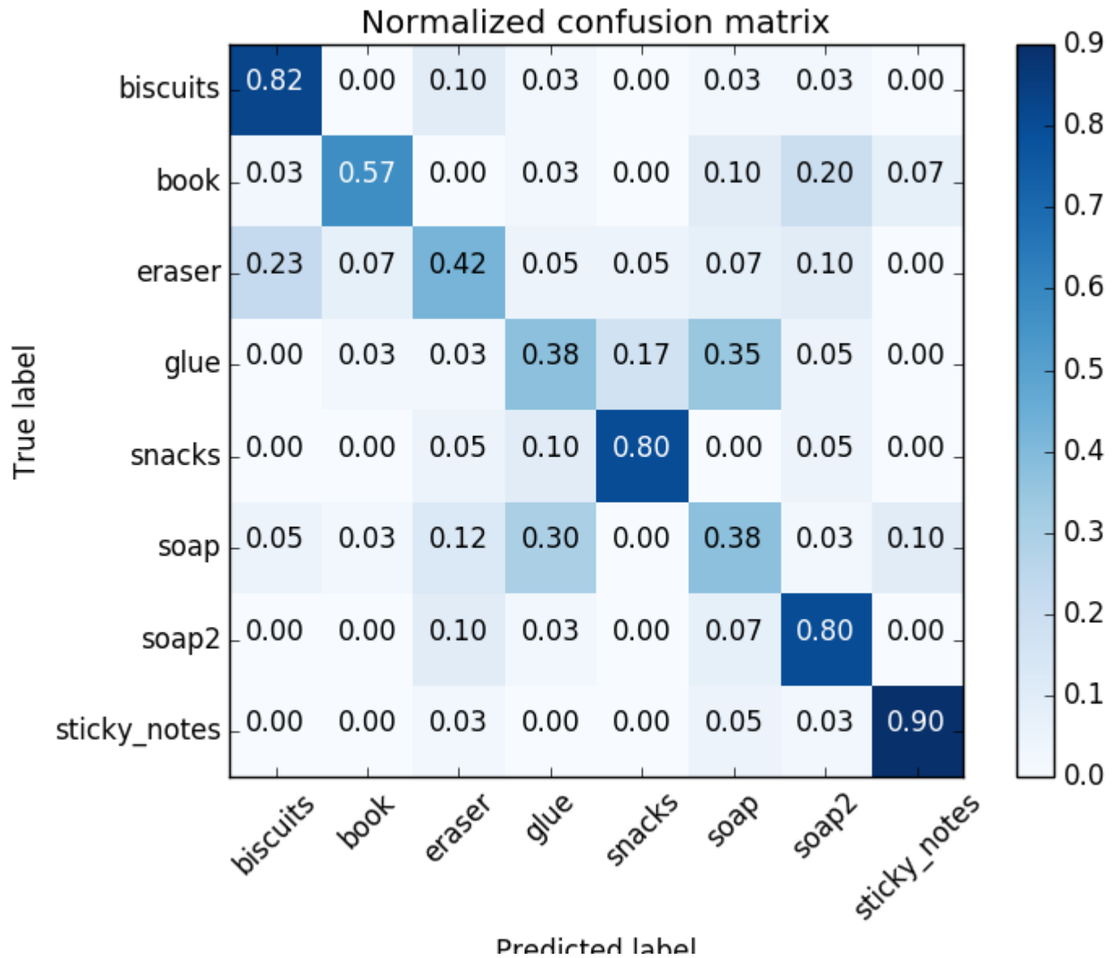
In a different environment 15 photos are taken to all the objects and their characteristics are captured.

SVM trained:

[Imagen: Confusion matrix]



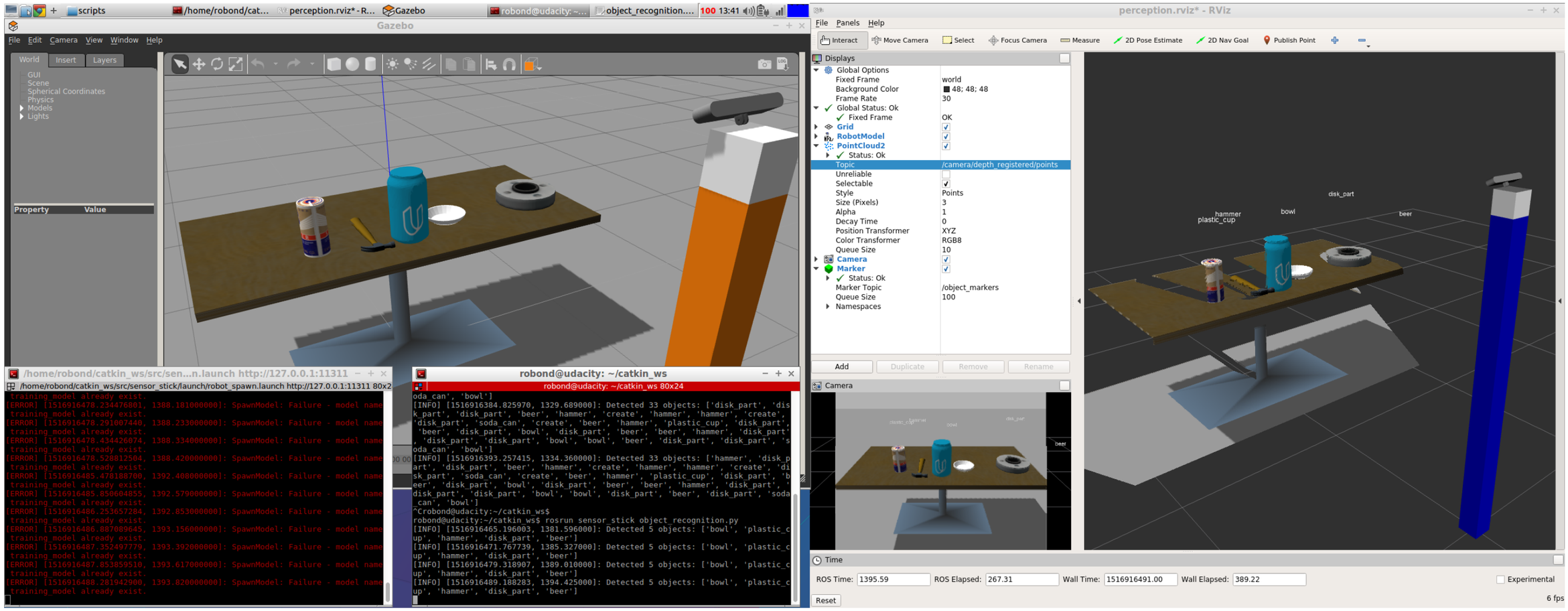
[Imagen: Normalized matrix]



Confusion matrix to SVM trined of project.

Object recognition:

[Imagen: objectsname]

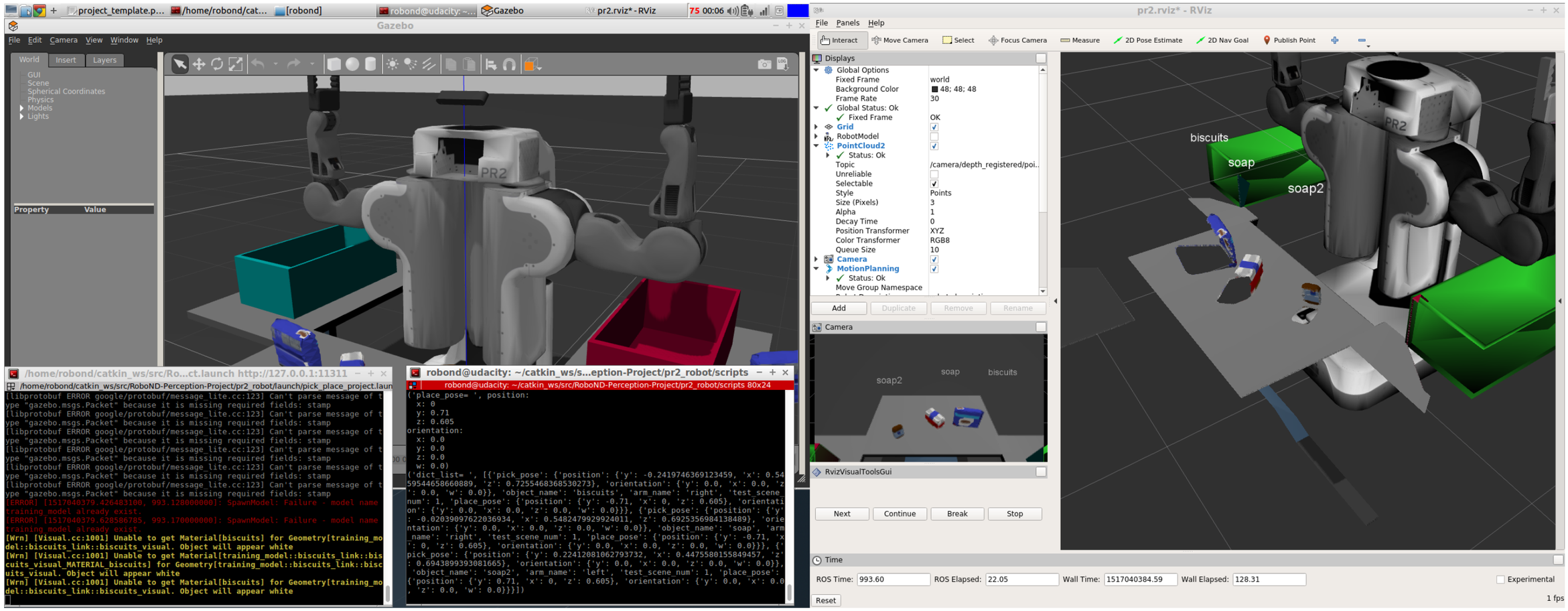


Recognized and labeled objects.

Pick and Place Setup

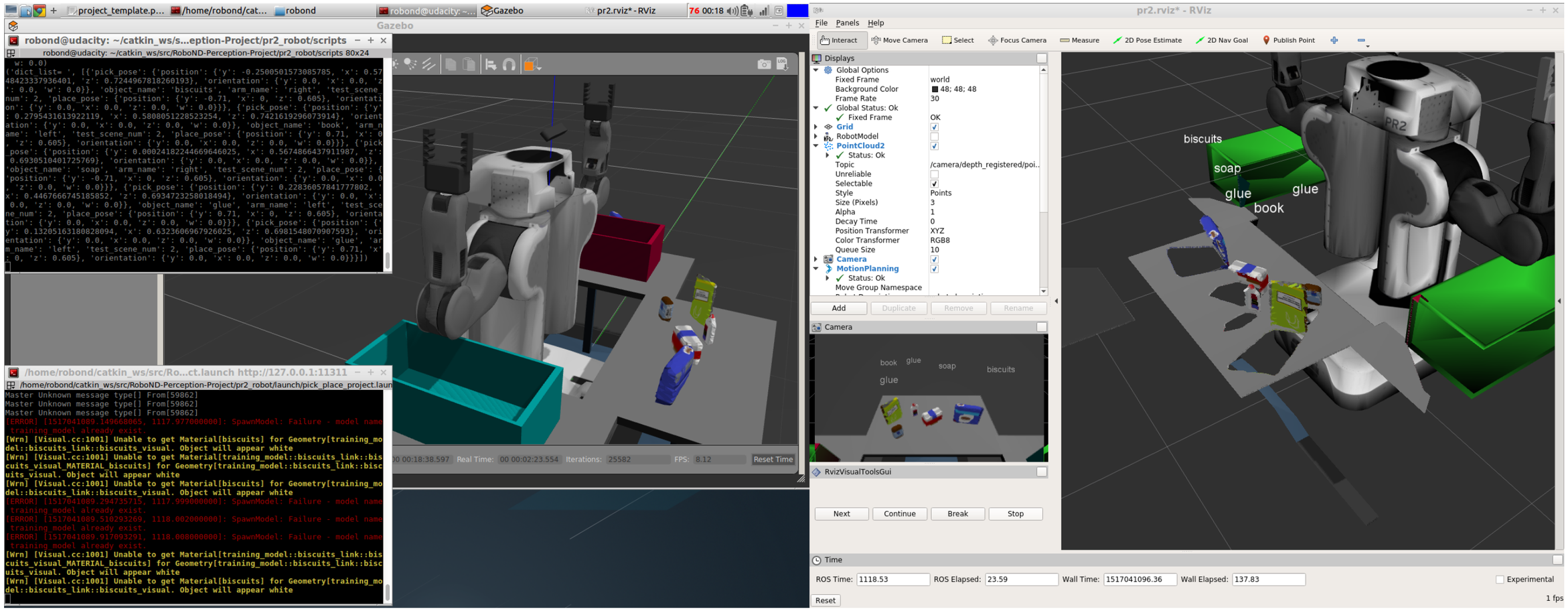
For all three tabletop setups (`test*.world`), perform object recognition, then read in respective pick list (`pick_list_*.yaml`). Next construct the messages that would comprise a valid `PickPlace` request output them to `.yaml` format.

[Imagen: world1]



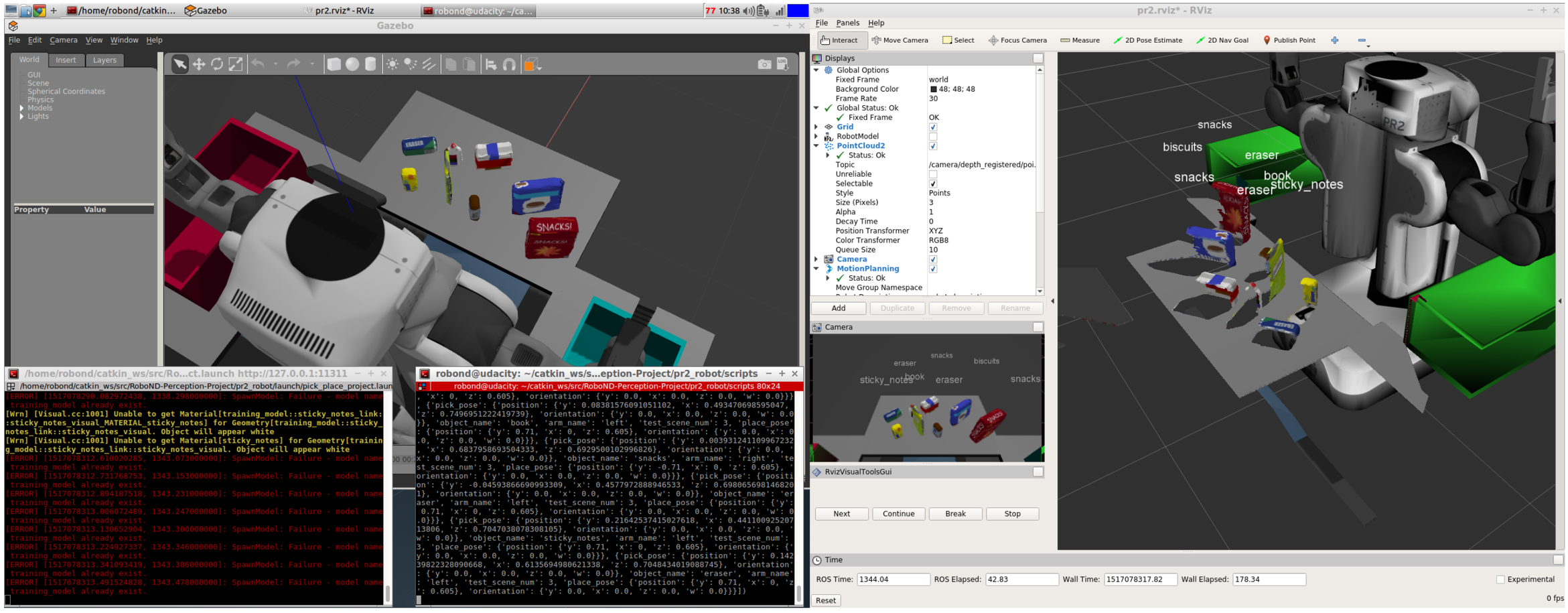
pick_place_project.launch: test1.world - pick_list_1.yaml project_template.py: models = 3 Objects - output file: output1.yaml

[Imagen: world2]



pick_place_project.launch: test2.world - pick_list_2.yaml - project_template.py: models = 5 Objects - output file: output2.yaml

[Imagen: world3]



pick_place_project.launch: test3.world - pick_list_3.yaml - project_template.py: models = 8 Objects - output file: output3.yaml

I loved this project. generate a cloud of points with color and depth and from there create a cluster, filter noise, capture features, train and most importantly recognize objects and all this by adding ROS to implement it in a pr2_robot. Fantastic.

Points to improve: improve the way to apply voxel to the environment. Improve the prediction technique to improve the confusion matrix. Improve the ways of the arm and avoid collisions.

Thank you very much enjoy it.

PD: the outputX.yaml files are in the following route RoboND-Perception-Project-Miguel\RoboND-Perception-Project-master\pr2_robot\scripts