pcl_recognition

A package for object recognition, build for EECS 476 Mobile Robotics Individual Project

Collaborators

Ran Hao (rxh349@case.edu)

Shipei Tian(sxt437@case.edu)

Things to do before running code

Plug in your Kinect to your machine

Start Kinect driver: `roslaunch freenect_launch freenect.launch`

Open rqt_reconfigure: `rosrun rqt_reconfigure rqt_reconfigure`

In **camera** / **driver**, check the box **depth_registration**.

Object Recognition Kitchen

Since now, the most fast and accurate way to recognize object is using ORK which is a mesh based recognition, see (*http://wg-perception.github.io/object_recognition_core/*).

Run it by

```
rosrun object_recognition_core object_add.py -n "coke " -d "A empty coke can" --commit
```

```
rosrun object_recognition_core mesh_add.py [the object id that previous command returned] `rospack find pc
```

```
roslaunch pcl_recognition ORK.launch
```

Video demonstration: *https://youtu.be/lbvA26GHWz0*

PCL Approaches

This packages contains multi approaches from PCL to recognize objects in a Kinect scene, including:

Correspondence Grouping

(*http://pointclouds.org/documentation/tutorials/correspondence_grouping.php*)

Run it by

```
roscd pcl_recognition/pcd
```

For test 1:

```
rosrun pcl_recognition correspondence_grouping coke_bad.pcd coke_scene.pcd -k -c --model_ss 0.01 --scene_s
```

For test 2:

```
rosrun pcl_recognition correspondence_grouping new_coke.pcd coke_2.pcd -k -c --model_ss 0.02 --scene_ss 0.
```

Implicit Shape Model

(*http://pointclouds.org/documentation/tutorials/implicit_shape_model.php*)

Run it by

```
roscd pcl_recognition
```

```
./ism_command.sh
```

Hypothesis Verification

(*http://pointclouds.org/documentation/tutorials/global_hypothesis_verification.php*)

Run it by

```
roscd pcl_recognition/pcd
```

```
rosrun pcl_recognition global_hypothesis_verification milk.pcd milk_cartoon_all_small_clorox.pcd -k
```

Iterative Closest Point

(*http://pointclouds.org/documentation/tutorials/interactive_icp.php*)

```
roscd pcl_recognition/pcd
```

```
rosrun pcl_recognition icp coke.ply 20
```

Library for recognition

From all the PCL Algorithms, the best one is Correspondence Grouping using Hough, a library is build with that algorithm, see **object_recognizer.cpp**, for usage see **object_recognize_main.cpp**

Run it by

For test 1:

```
roslaunch pcl_recognition object_recognizer_test1.launch
```

For test 2:

```
roslaunch pcl_recognition object_recognizer_test2.launch
```

For test 3:

```
rosrun pcl_recognition object_recognize_main
```

PCD Edit Tool

For all approaches in PCL library, it all requires pcd files as original input, so we write an useful tool to make a pcd file from Kinect cloud so that you can use it as input for object_recognizer.

Run it by

```
rosrun pcl_recognition pcd_edit_tool name_you_want_to_save.pcd
```

Usage demonstration: *https://youtu.be/GV69MXoV2kg*

Calculate normal, centroid of a plane

Run it by

```
roslaunch pcl_recognition find_stool_coke.launch
```

Select the object you want to find using Publish Selected Points.

Or, you can this node separately by:

```
rosrun pcl_recognition find_stool (your pcd file name)
```

It will let you select the type of object you want to find in order to load the best filter range, if other selection, it will let you manually input the filter range