Lab09 PCL and ICP

By Michael Su, Lily Hsu last modified on 04/19 2020

The objective of this tutorial is to introduce Point Cloud Library(PCL) and Iterative Closest Point(ICP).

[**Overview**](#_e7vue6301h0) **1**

[**Topics and Activities**](#_lg1g6up84yjy) **2**

[Topic 1: PCLexample](#_7t0w87d2e32w) 2

[Discussion and Checkpoint](#_j25ic61clf18) 2

[Topic 2: Surface normal](#_nkxhd3dzloq4) 3

[Discussion.](#_hkh0bbgwsnz5) 3

[Topic 3: Iterative Closest Point](#_b4ev1dk8dvsf) 3

[**Reference**](#_sxcm0uqez0at) **3**

# Overview

Estimated Time to Finish: 1.3 hours

After completing this tutorial you should

* be able to
  + run RealSense Camera
  + use ROS topics from RealSense Camera
  + use OpenCV
* know
  + how pointcloud is generated.
  + what is Morphological Image Processing
  + what is camera matrix
  + what is HSV color space

# Topics and Activities

## **Topic 1: PCLexample**

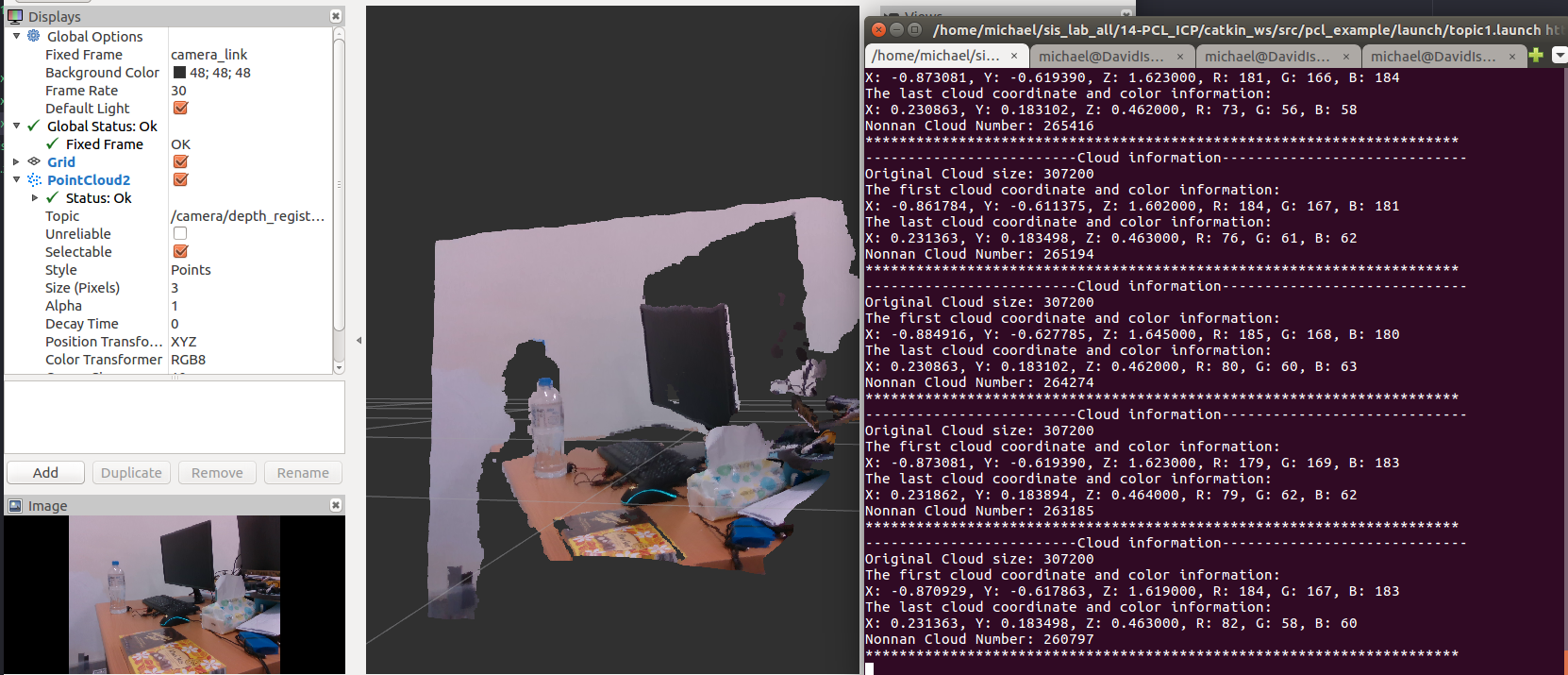
**laptop$ cd 09-PCL\_and\_ICP/catkin\_ws**

**laptop$ catkin\_make**

**laptop$ source devel/setup.bash**

**(Make sure your RealSense Camera is connected to CP)**

**laptop$ roslaunch pcl\_example topic1.launch**

****

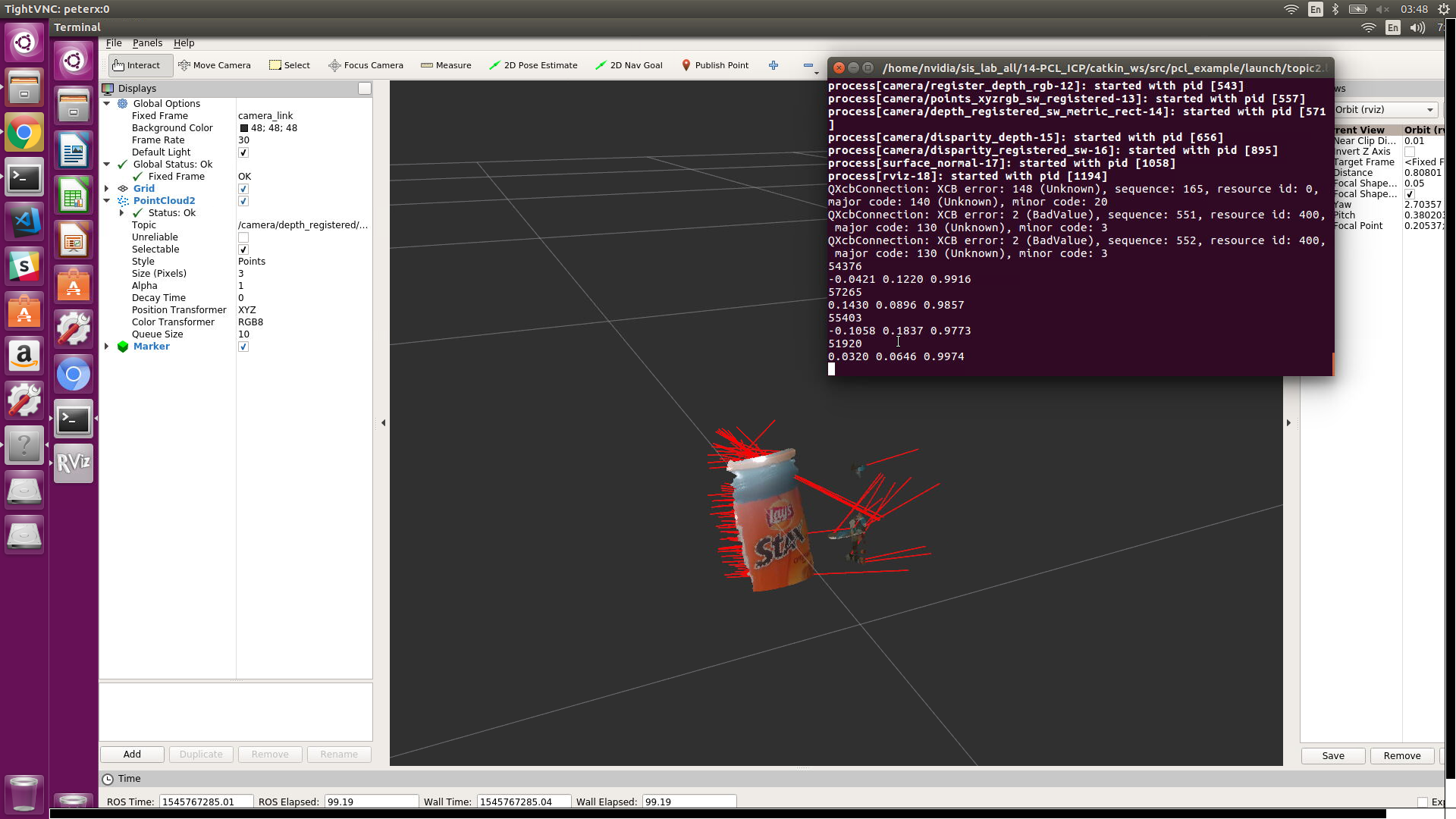
* Convert point cloud type from sensor\_msgs::PointCloud2(ros) to pcl::PointCloud
* Information about pcl::PointCloudXYZRGB information
* How to remove the nan points from the original point cloud

### Discussion and Check point

1. Please show the results
2. How is the PCL pointcloud organized? Is there any relation between RGB image and pointcloud?
3. What information can we get from a single point?
4. Why does filtered point cloud have less points than original point cloud?

## Topic 2: Surface normal

**laptop$ roslaunch pcl\_example topic2.launch**



* Using pcl to find point cloud surface normal
* pcl::search::KdTree - find the nearest point, pcl::NormalEstimation - find the normal
* visualization\_msgs::Marker - ros tool helping you to visualize your work. There are many kind of markers. In this tutorial, we use visualization\_msgs::Marker::LINE\_LIST as the marker type.
* Method : use approximations to infer the surface normals from the point cloud.
  + for each point p in cloud P

1. get the nearest neighbors of p

2. compute the surface normal n of p（compute）

3. check if n is consistently oriented towards the viewpoint and flip otherwise

### Discussion and Check point

1. Show the results of the topic 2
2. Do you find any pointcloud pre-processing before calculating surface normal?
3. If you increase the search range for surface normal, What will happen?

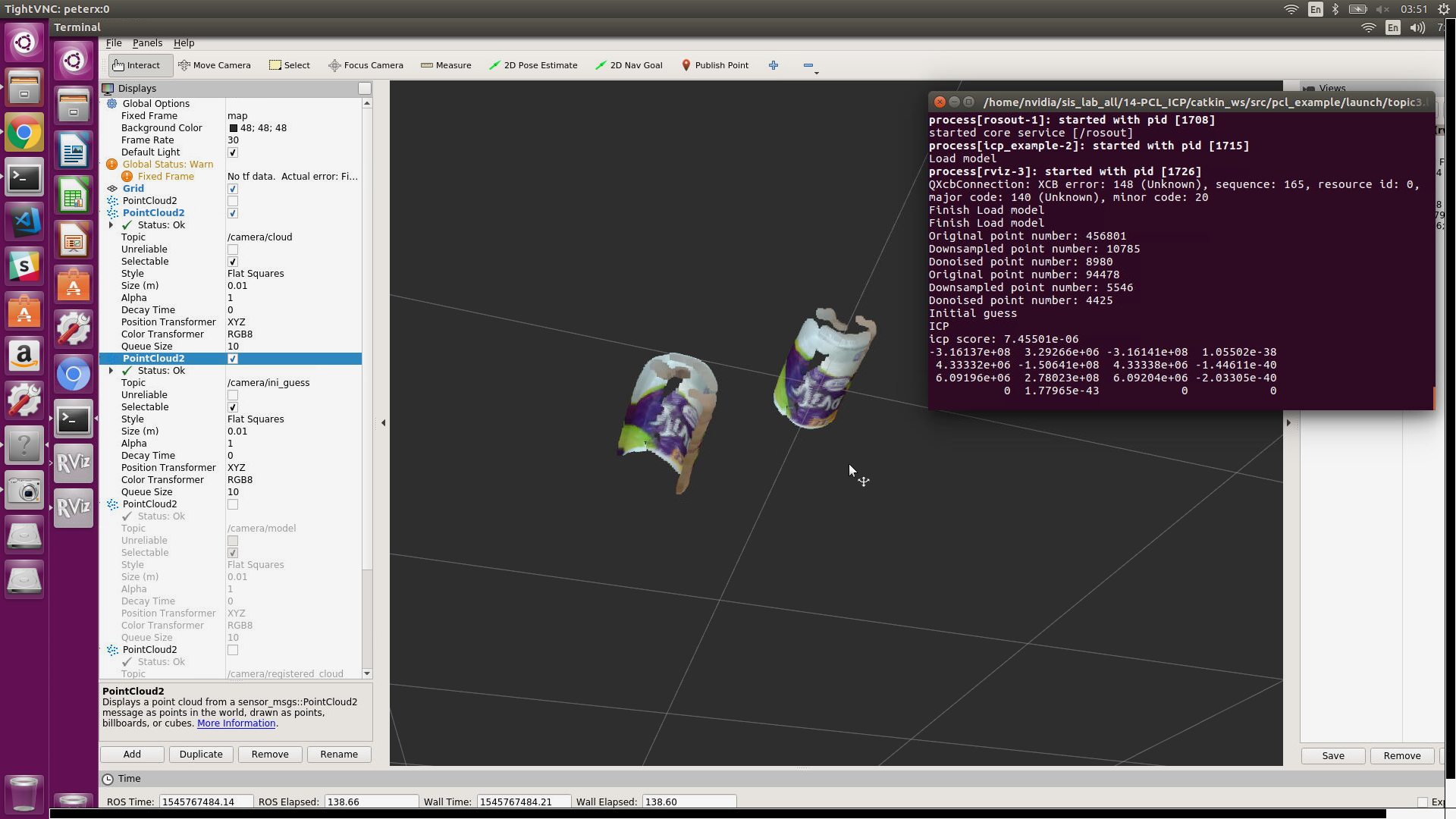
## Topic 3: Iterative Closet Point

**laptop$ roslaunch pcl\_example topic3.launch model:=[VIVA\_model.ply file path]**

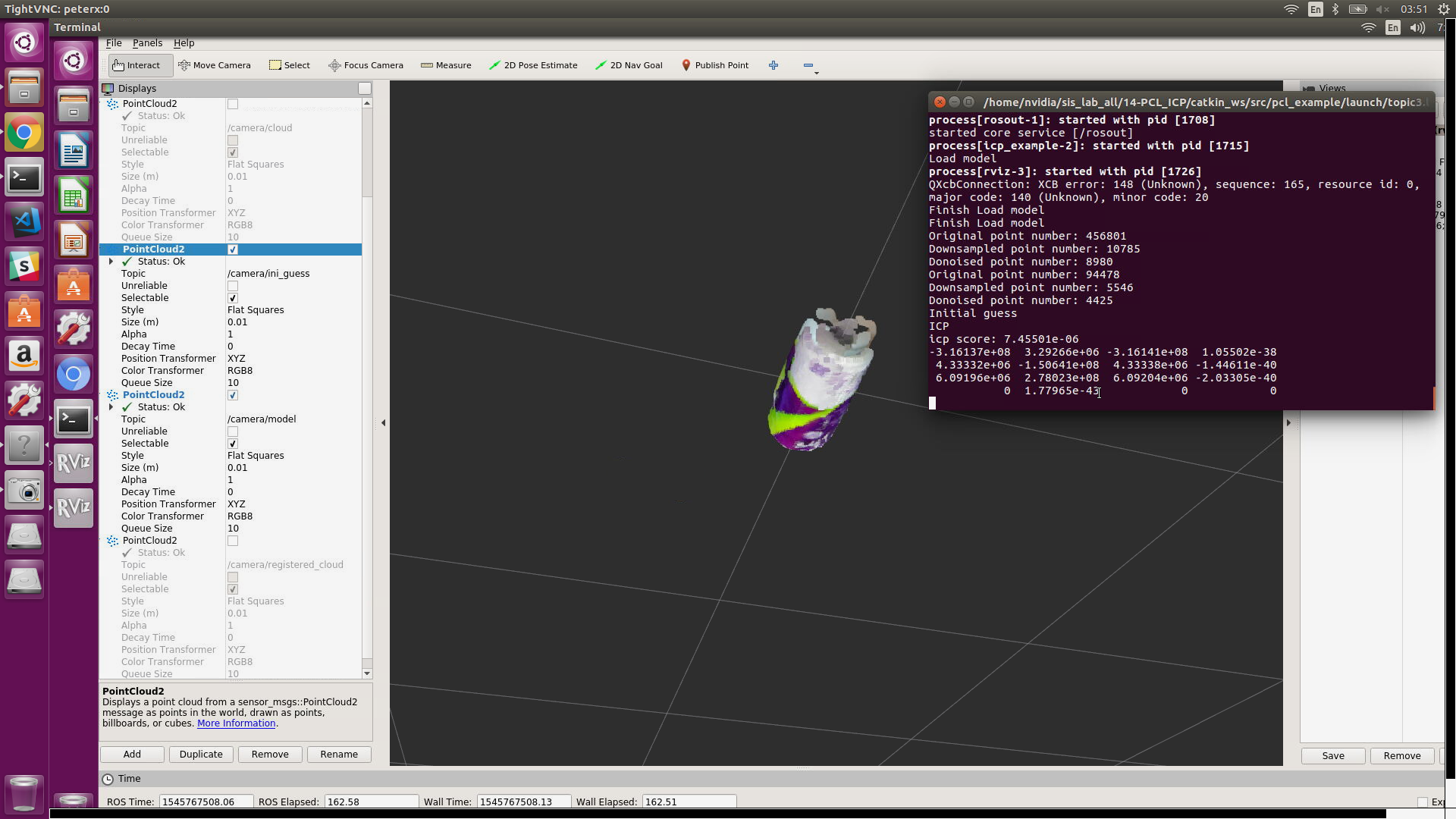
**cloud:=[VIVA\_cloud.ply file path]**

You can find this two files in **/cpl\_example/model/** folder.

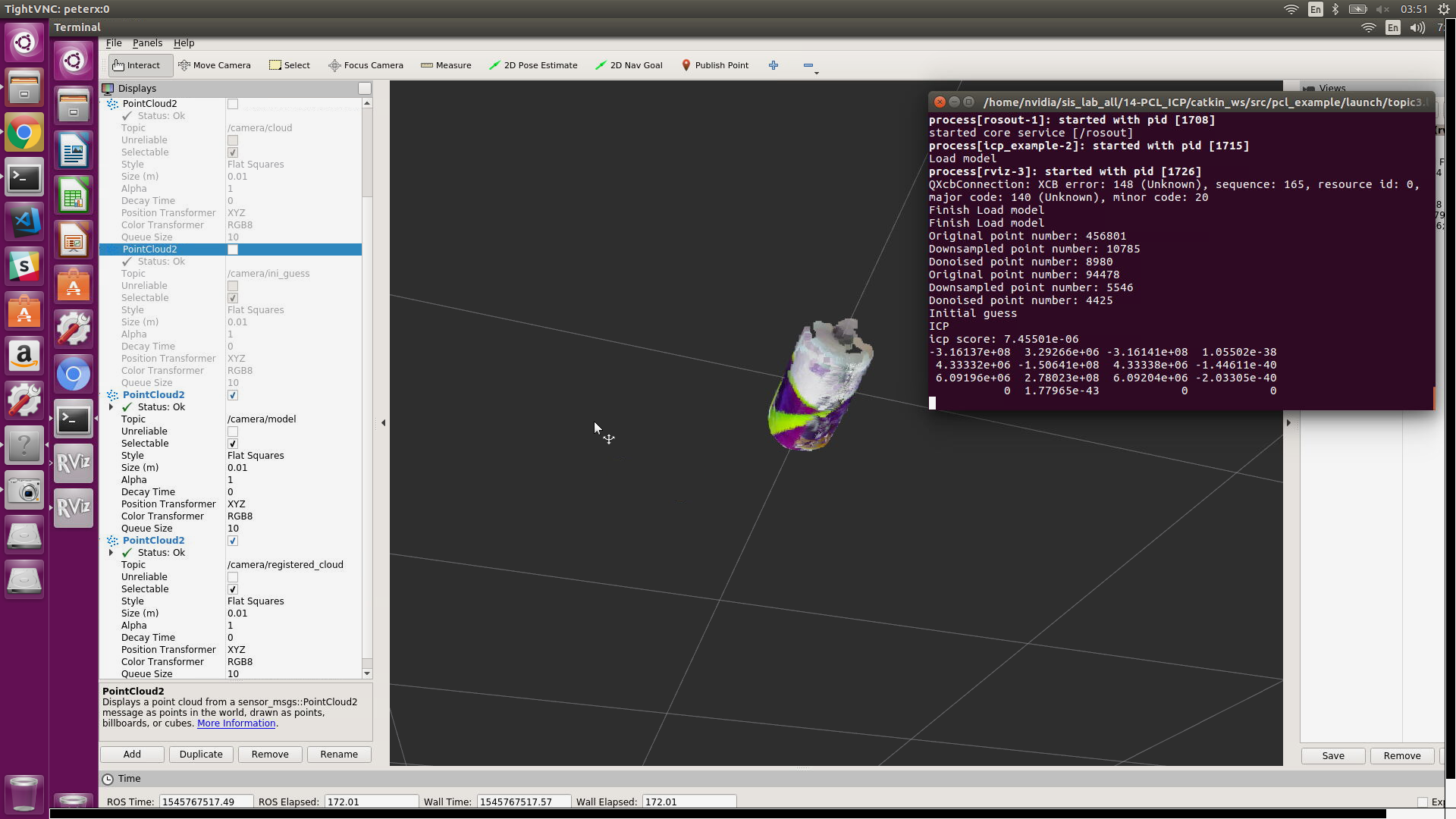
Tip: click the file press ctrl+c, then move to terminal press shift+ctrl+v. it will paste the path of the file.

****

**/camera/ini\_guess & /camera/cloud**

****

**/camera/ini\_guess & /camera/model**

****

**/camera/model & /camera/registered\_cloud**

* There are four functions in icp\_example.cpp, addNormal, point\_preprocess, initial\_guess, point\_2\_plane\_icp. Read the code to know how to do icp(with normal).
  + Preprocess pointcloud
    - Downsample
    - Denoise
  + Start from initial guess
    - Position: Centroid
    - Orientation: PCA
  + ICP

Method

* + - For each point on M, find closest point on P
    - Find best transform for this corresponding
    - Transform M

Iter （criterion）

* + - Number of iterations has reached the maximum user imposed number of iterations (setMaximumIterations)
    - The epsilon (difference) between the previous transformation and the current estimated transformation is smaller than an user imposed value (setTransformationEpsilon)
    - The sum of Euclidean squared errors is smaller than a user defined threshold (setEuclideanFitnessEpsilon)

Different kind of icp

* + - pcl::GeneralizedIterativeClosestPoint< PointSource, PointTarget > Class Template Reference
    - pcl::IterativeClosestPoint< PointSource, PointTarget, Scalar > Class Template Reference
    - pcl::IterativeClosestPointWithNormals< PointSource, PointTarget, Scalar > Class Template Reference
    - pcl::IterativeClosestPointNonLinear< PointSource, PointTarget, Scalar > Class Template Reference
    - pcl::JointIterativeClosestPoint< PointSource, PointTarget, Scalar > Class Template Reference
    - pcl::registration::IncrementalICP< PointT, Scalar > Class Template Reference

### Discussion and Check point

There are other object models and corresponding object point cloud. Please change the model and corresponding cloud and observe the result.

1. Show the result of topic 3
2. Is it essential to do initial guess for ICP? If not, what will happen? Please show me the registration result that doesn’t execute initial guess before ICP.
3. What do we use for ICP termination criterion?
4. What’s the difference between point-to-plane ICP and point-to-point ICP? (optional)

# Reference

* <http://wiki.ros.org/rviz/Tutorials/Markers%3A%20Points%20and%20Lines#Using_Points.2C_Line_Strips.2C_and_Line_Lists>
* <http://eigen.tuxfamily.org/dox/group__TutorialBlockOperations.html>
* <https://eigen.tuxfamily.org/dox/classEigen_1_1Quaternion.html>
* <http://mediatum.ub.tum.de/doc/800632/941254.pdf>