Programming exercise 5: PCL

Registration with ICP

Registration of two point clouds means recovering a coordinate transformation (rotation and translation), applying which a source point cloud is aligned with a target point cloud. In this exercise, we will write an application that reads two point clouds from disk, performs registration and aligns the two clouds in the same display.

Write the application following these guidelines.

1. Read two user-given input clouds, a target and source cloud, from disk. Print an error and exit the application if there is a failure reading the inputs.

2. Filter the input clouds to remove `nan` entries and downsample the data to make next steps faster to compute.

3. Use the iterative closest points (ICP) method to align the two clouds. Output convergence information, fitness score and the final transformation to terminal. Try two cases: one without an initial guess, and with an initial guess generated by

```
Eigen::AngleAxisf init_rotation(0.7, Eigen::Vector3f::UnitZ());
Eigen::Translation3f init_translation(0.5, -0.3, -0.3);
Eigen::Matrix4f init_guess = (init_translation * init_rotation).matrix();
```

4. Use the transformation to align the full resolution source cloud to the full resolution target cloud.

5. Visualize the target cloud and the transformed source cloud together: transformed target cloud with red, and source cloud with green color.

Hints:

- To test your application, you may use the provided point clouds `cloud001.pcd` and `cloud002.pcd`.
- Use the inputs `argc` and `argv` of the `main` function to get user input from the console. Use `pcl::io::loadPCDFile` to read the clouds.
- Use `pcl::visualization::PCLVisualizer` to help you visualize and inspect the status at each intermediate step to make sure your application works as expected.
- For filtering, apply `pcl::removeNaNFromPointCloud`, `pcl::VoxelGrid`, and `pcl::PassThrough` appropriately.
- For alignment, downsample the source point cloud, but leave the target cloud at full resolution (or only downsample slightly). This helps to achieve a more robust alignment.

- To visualize, use `pcl::visualization::PCLVisualizer`. Define colors with the help of `pcl::visualization::PointCloudColorHandlerCustom`.

Registration with NDT

Repeat the steps of the previous task, but instead of ICP apply the Normal Distributions Transform (NDT) algorithm. Use the `pcl::NormalDistributionsTransform` object.

Hints:

- Note that NDT uses the same style of registration API as ICP.
- Remember to set the scale dependent parameters appropriately! Use the member functions `setTransformationEpsilon`, `setStepSize`, and `setResolution`.

Useful links

- Registration API: `http://pointclouds.org/documentation/tutorials/registration_api.php`
- ICP Class reference: `http://docs.pointclouds.org/trunk/classpcl_1_1_iterative_closest_point.html`
- NDT Class reference: `http://docs.pointclouds.org/trunk/classpcl_1_1_normal_distributions_transform.html`
- PCLVisualizer Class Reference: `http://docs.pointclouds.org/1.0.0/classpcl_1_1visualization_1_1_p_c_l_visualizer.html`