10. Data conversion and point cloud

10.1. Scan into Image

- Start

```
roslaunch yahboom_navrobo_visual laser_to_image.launch
```

- View node graph

```
rqt_graph
```

- py code analysis

Create subscribers and publishers

```
self.laserSub = rospy.Subscriber("/scan", LaserScan, self.laserCallback) #
Receive scan node
self.image_pub = rospy.Publisher('/laserImage', Image, queue_size=1) # Publish
converted image information
```

Process data in callback function [self.laserCallback()] and publish

```
def laserCallback(self, data):
    # Extract received laser radar data and convert it into point cloud data
    cloud_out = self.laserProj.projectLaser(data)
    lidar = point_cloud2.read_points(cloud_out)
    points = np.array(list(lidar))
    # Convert point cloud data to image data
    img = self.pointcloud_to_laserImage(points)
    # Convert image data to ROS image information and publish
    self.image_pub.publish(self.bridge.cv2_to_imgmsg(img))
    img=cv.resize(img,(640,480))
    cv.imshow("img", img)
    cv.waitKey(10)
    ROS_INFO("Published ... ");
```

10.2, ROS and PCD

Introduce several tools running on several ROS nodes. Their function is to convert between format point cloud or package and point cloud data (PCD file format).

Start Astra camera/Gemin335 camera

```
sudo supervisorctl stop ChassisServer
#Start Astra camera
roslaunch orbbec_camera astra_pro2.launch
#Gemin335 camera
roslaunch orbbec_camera gemini_330_series.launch
```

Point cloud display: rviz (start rviz command, select corresponding topic, modify parameters, present different effects); pcl_visualization tool.

```
roslaunch yahboom_navrobo_visual pointCloud_visualize.launch
cloud_topic:=/camera/depth_registered/points
```

10.2.1, pointcloud_to_pcd

```
rosrun pcl_ros pointcloud_to_pcd input:=/camera/depth/points # x y z
rosrun pcl_ros pointcloud_to_pcd input:=/camera/depth_registered/points # x y z
rgb
```

Save the ROS point cloud message in the specified PCD file.

10.2.2, convert_pcd_to_image

```
rosrun pcl_ros convert_pcd_to_image <cloud.pcd>
```

Load a PCD file (must have x y z rgb) and publish it as a ROS image message five times per second.

10.2.3, pcd_to_pointcloud

```
rosrun pcl_ros pcd_to_pointcloud <file.pcd> [ <interval> ]
```

Load a PCD file and publish it once or multiple times as a ROS point cloud message.

- file.pcd: The (required) file name to read.
- interval: (optional) The number of seconds to sleep between messages. If the parameter [interval] is zero or not specified, the message is published once.

```
roslaunch yahboom_navrobo_visual pointCloud_visualize.launch
cloud_topic:=/cloud_pcd
```

10.2.4, bag_to_pcd

rosbag recording

Command: rosbag record topic1 [topic2 topic3 ...]

```
rosbag record /camera/depth_registered/points
```

bag_to_pcd

```
rosrun pcl_ros bag_to_pcd <input_file.bag> <topic> <output_directory>
# For example:
rosrun pcl_ros bag_to_pcd 2021-09-09-11-41-56.bag
/camera/depth_registered/points my_pcd
```

Read a bag file and save the ROS point cloud message in the specified PCD file. This requires a bag file.

10.3, PCL 3D point cloud

PCL (Point Cloud Library) is a large cross-platform open source C++ programming library built on the basis of previous point cloud related research. It implements a large number of general point cloud related algorithms and efficient data structures, involving point cloud acquisition, filtering, segmentation, registration, retrieval, feature extraction, recognition, tracking, surface reconstruction, visualization, etc. It supports multiple operating system platforms and can run on Windows, Linux, Android, Mac OS X, and some embedded real-time systems. If OpenCV is the crystallization of 2D information acquisition and processing, then PCL has the same status in 3D information acquisition and processing. PCL is a BSD license method and can be used for commercial and academic applications for free.

PCL was originally an open source project maintained and developed by Dr. Radu from Technische Universität München (TUM - Technische Universität München) and Stanford University under ROS (Robot Operating System). It is mainly used in the field of robot research and application. With the accumulation of various algorithm modules, it became independent in 2011 and officially formed a strong development and maintenance team with global 3D information acquisition and processing peers, mainly from many well-known universities, research institutes and related hardware and software companies. It has developed very rapidly, with new research institutions constantly joining. With the financial support of many world-renowned companies such as Willow Garage, NVidia, Google (GSOC 2011), Toyota, Trimble, Urban Robotics, Honda

Research Institute, etc., new development plans are constantly proposed, and code updates are very active. So far, it has been released from version 1.0 to version 1.7.0 in less than a year.

This section mainly explains random point cloud publishing and point cloud visualization.

10.3.1, point cloud publishing

Publish point cloud, launch file, including rviz startup. So I can clearly see a point cloud flashing in the middle of rviz.

```
roslaunch yahboom_navrobo_visual pointCloud_pub.launch use_rviz:=true
```

- use_rviz parameter: whether to enable rviz visualization
- Code analysis

The source code comments are very clear, please check the source code directly.
/home/yahboom/YBAMR-COBOT-EDU-00001/src/yahboom_navrobo_other/yahboom_navrobo_visual/src/pub_pointCloud.cpp

10.3.2, Point Cloud Visualization

- rviz

```
rviz
```

- pcl_visualization

PCL visualization library, created to quickly restore and visualize the results obtained after calculating 3D point cloud data through algorithms. Similar to OpenCV's highgui program, it is used to display 2D images or 2D shapes on the screen.

Startup command

```
roslaunch yahboom_navrobo_visual pointCloud_visualize.launch
cloud_topic:=color_cloud
```

- cloud_topic parameter: the topic name of the subscribed point cloud.
- Shortcut keys

[Ctrl]+[-]: Zoom out.

[Shift]+[+]: Zoom in.

[Alt]+[-]: Zoom out.

[Alt]+[+]: Zoom in.

Mouse wheel and left and right buttons can also be used for control.

- Code analysis

The source code comments are very clear, please check the source code directly.
/home/yahboom/YBAMR-COBOT-EDU-00001/src/yahboom_navrobo_other/yahboom_navrobo_visual/src/pcl_visualize.cpp