10. Data conversion and point cloud

10.1, scan to image

```
#Raspberry Pi 5 master needs to enter docker first, please perform this step
#If running the script into docker fails, please refer to ROS/07, Docker tutorial
~/run_docker.sh
#Multiple ros commands require multiple terminals to be executed in the same
docker container. Please refer to the tutorials in Sections 07/5 and 5.8.
```

- start up

```
roslaunch yahboomcar_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
the converted image information
```

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

```
def laserCallback(self, data):
     # Extract the received lidar 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 into image data
     img = self.pointcloud_to_laserImage(points)
     # Convert image data into ROS image information and publish it
     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

Introducing several tools run by several ROS nodes. Their function is to convert between format point clouds or packages and point cloud data (PCD file format).

Launch Astra Camera

```
roslaunch astra_camera astraproplus.launch
```

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

```
roslaunch yahboomcar_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 ROS point cloud messages 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, convert_pointcloud_to_image

```
rosrun pcl_ros convert_pointcloud_to_image input:=/camera/depth_registered/points
output:=/my_image
View image: rosrun image_view image_view image:=/my_image
```

Subscribe to a ROS point cloud topic and publish it as image information.

10.2.4, pcd_to_pointcloud

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

Load a PCD file and publish it one or more times as a ROS point cloud message.

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

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

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

10.3, PCL three-dimensional 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 point cloud-related general 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 authorized method and can be used for free commercial and academic applications.

PCL was originally developed under ROS (Robot Operating System) from Technische Universität München (TUM - Technische Universität München) and Stanford University) is an open source project maintained and developed by Dr. Radu and others. 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 powerful team with peers in global 3D information acquisition and processing. The development and maintenance teams are mainly from many well-known universities, research institutes and related hardware and software companies. It is developing very rapidly, and new research institutions are 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 research solutions are constantly being proposed. The development plan and code updates are very active, and it has been released from version 1.0 to version 1.7.0 in less than a year.

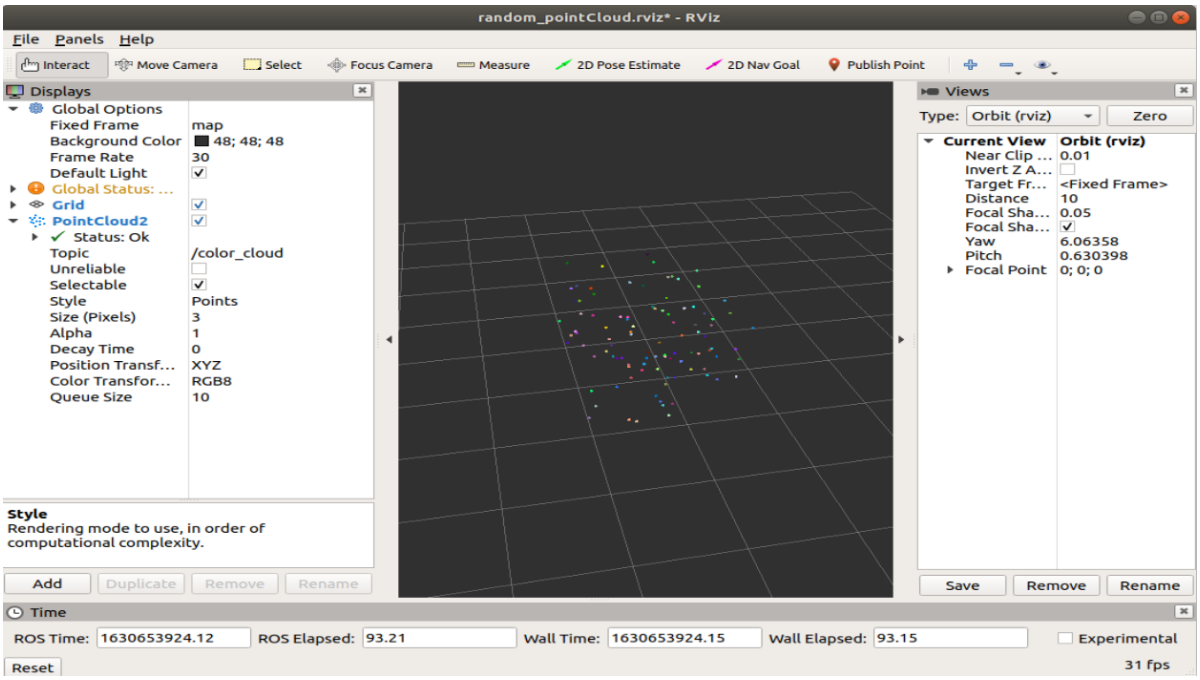
This section mainly explains random point cloud publishing and point cloud visualization.

10.3.1. Point cloud release

To publish a point cloud, the launch file contains the startup of rviz. So I can clearly see a point cloud flashing in the middle of rviz.

```
roslaunch yahboomcar_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.
~/yahboomcar_ws/src/yahboomcar_visual/src/pub_pointCloud.cpp

10.3.2. Point cloud visualization

-rviz

```
rviz
```

- pcl_visualization

The PCL visualization library was created to quickly restore and visualize the results obtained after calculating three-dimensional point cloud data through algorithms. A highgui program similar to OpenCV is used to display two-dimensional images or two-dimensional shapes on the screen.

Start command

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

- cloud_topic parameter: the topic name of the point cloud subscription.

- shortcut key

[Ctrl]+[-]: Zoom out.

[Shift]+[+]: Zoom in.

[Alt]+[-]: Zoom out.

[Alt]+[+]: Pull in.

The mouse wheel and left and right buttons are also controllable.

- Code analysis

The source code comments are very clear, please check the source code directly.
~/yahboomcar_ws/src/yahboomcar_visual/src/pcl_visualize.cpp