6. ORB_SLAM2 PCL build map

pcl official website address:: http://pointclouds.org

pcl_ros wiki: https://wiki.ros.org/pcl_ros

pcl_ros github: https://github.com/ros-perception/perception_pcl

The operating environment and hardware and software reference configuration are as follows:

- Reference model: ROSMASTER X3
- Robot hardware configuration: Arm series main control, Silan A1 LiDAR, AstraPro Plus depth camera.
- Robot system: Ubuntu (version not required) + docker (version 20.10.21 and above)
- PC virtual machine: Ubuntu (20.04) + ROS2 (Foxy)
- Usage scenario: use on a relatively clean 2D plane

6.1, Introduction

PCL (The Point Cloud Library) is a large open source project for 2D/3D image and point cloud processing.The PCL framework consists of a number of state-of-the-art algorithms for filtering, feature estimation, surface reconstruction, alignment, model collocation and segmentation. These algorithms have many applications, such as filtering outliers in noisy data, assembling multiple 3D point clouds, segmenting relevant parts of a scene, extracting key points and computing descriptors of geometric shapes for object recognition, creating and visualizing surfaces of objects using point clouds, etc. PCL has been compiled and configured for Linux, MacOS, Windows, and Android/iOS. iOS platforms. To simplify development, PCL has been split into a series of small, individually compilable code libraries.

Basic concepts of the Point Cloud Library and the PCL interface for ROS:

Point Cloud Library: provides a set of data structures and algorithms for processing 3D data;

ROS's PCL interface: provides a set of messages and conversion functions between messages and PCL data structures.

From a C++ point of view, the PCL contains a very important data structure, which is the point cloud. This data structure is designed as a template class that takes the type of the point as a parameter to the template class. The point cloud class is actually a container that contains all the public information needed for the point cloud, regardless of the type of the point. Here are the most important public fields in the point cloud:

header: this field is of type pcl:PCLHeader. It specifies when the point cloud was acquired.

points: this field is of type std::vector<PointT...> and is a container for all the points. pointT in the vector definition corresponds to the template parameter of the class, i.e. the type of the point.

width: this field specifies the width of the point cloud when it is organized into an image, otherwise it contains the number of points in the cloud.

height: this field specifies the height of the point cloud when organized as an image, otherwise it is always 1.

is_dense: this field specifies if there are invalid values (infinity or NaN values) in the point cloud.

sensor origin: this field is of type Eigen::Vector4f and defines the bitmap of the sensor according to the translation with respect to the origin.

sensororientation: this field is of type Eigen::Ouaternionf and defines the orientation of the sensor according to its rotation.

6.2. Using

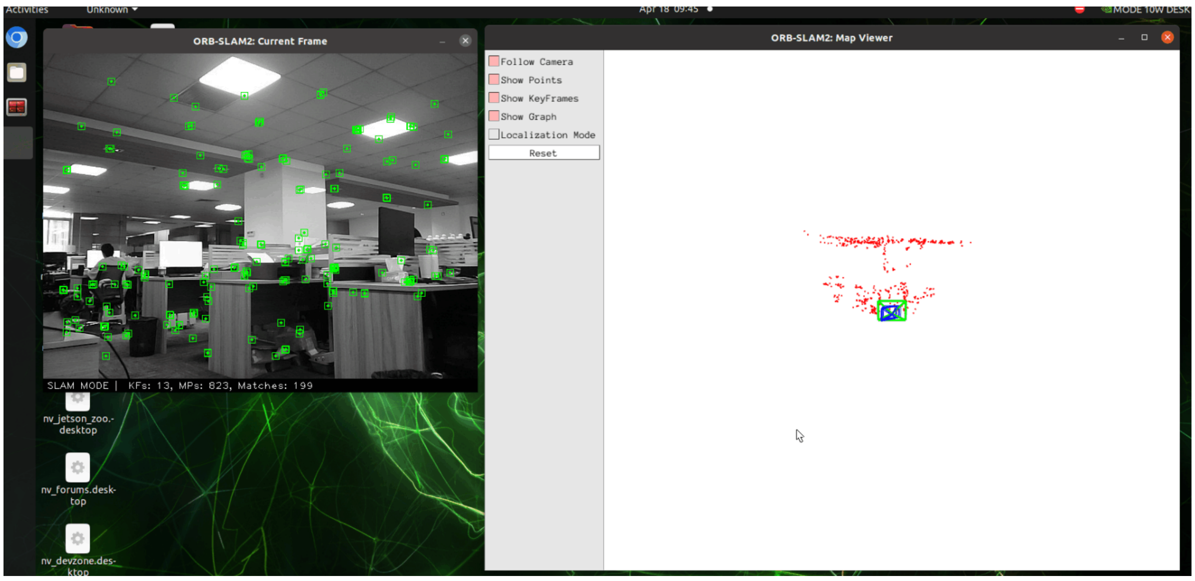
Enter the docker container (for steps, see [docker course chapter ----- 5. Enter the docker container of the robot]), and execute the following LAUNCH file in a sub-terminal:

1. Start the camera

```
ros2 launch astra_camera astro_pro_plus.launch.xml
```

2. Start orbslam to publish the camera position as well as the color and depth maps, depending on the performance of different masters, the waiting time here is about 10s

```
ros2 launch yahboomcar_slam orbslam_base_launch.py
```

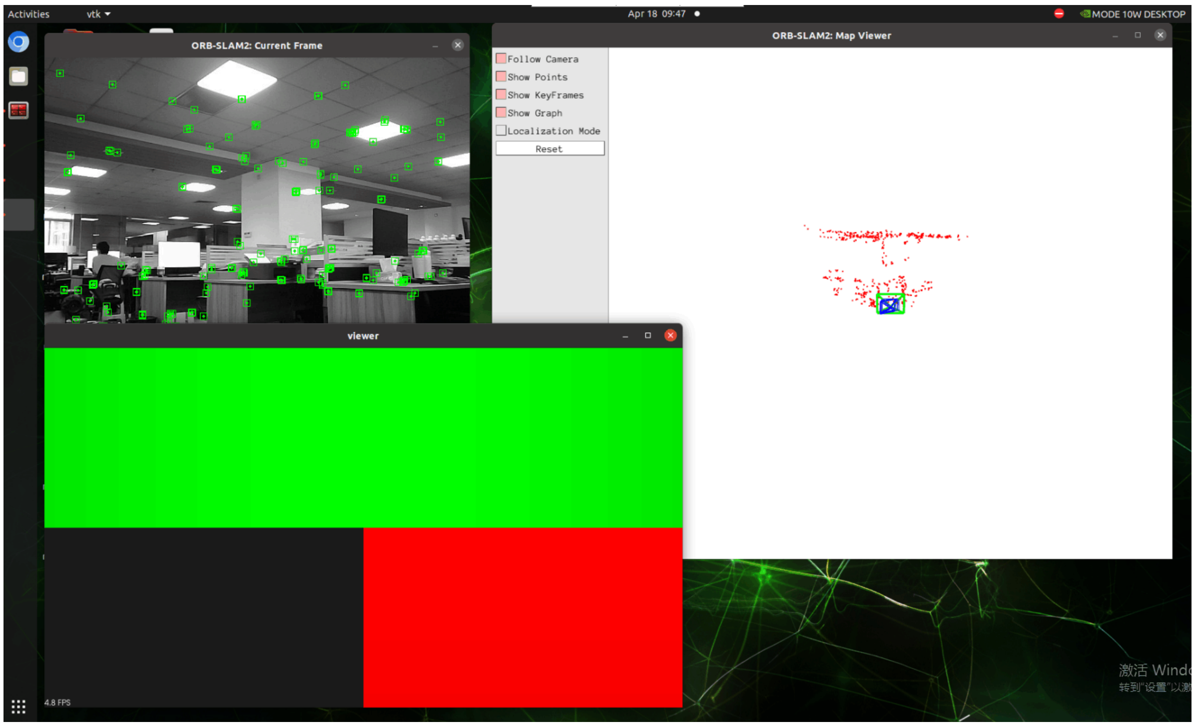


```
SLAM MODE |  KFs: 13, MPs: 823, Matches: 199
```

3. Start point cloud construction

```
ros2 launch yahboomcar_slam orbslam_pcl_map_launch.py
```

When the viewer pops up, move the camera slowly, and when a picture appears, look at the following picture.



To zoom and rotate the coordinate system, see the following chart

Slide the wheel: zoom in and out.

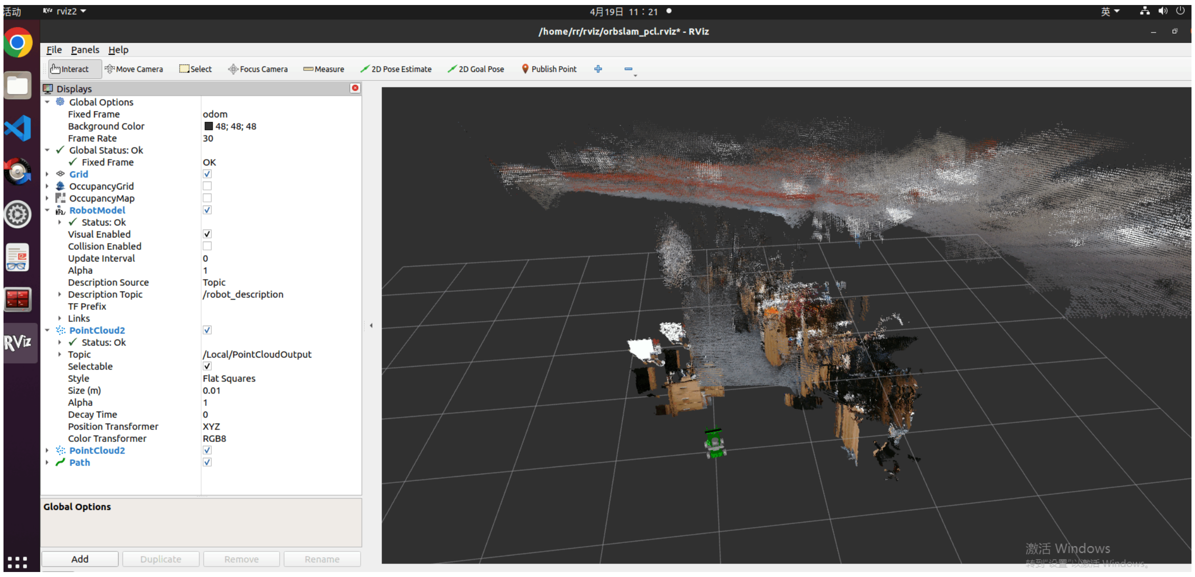
Press and hold the wheel: Pan

Left click on the cursor: Rotate.

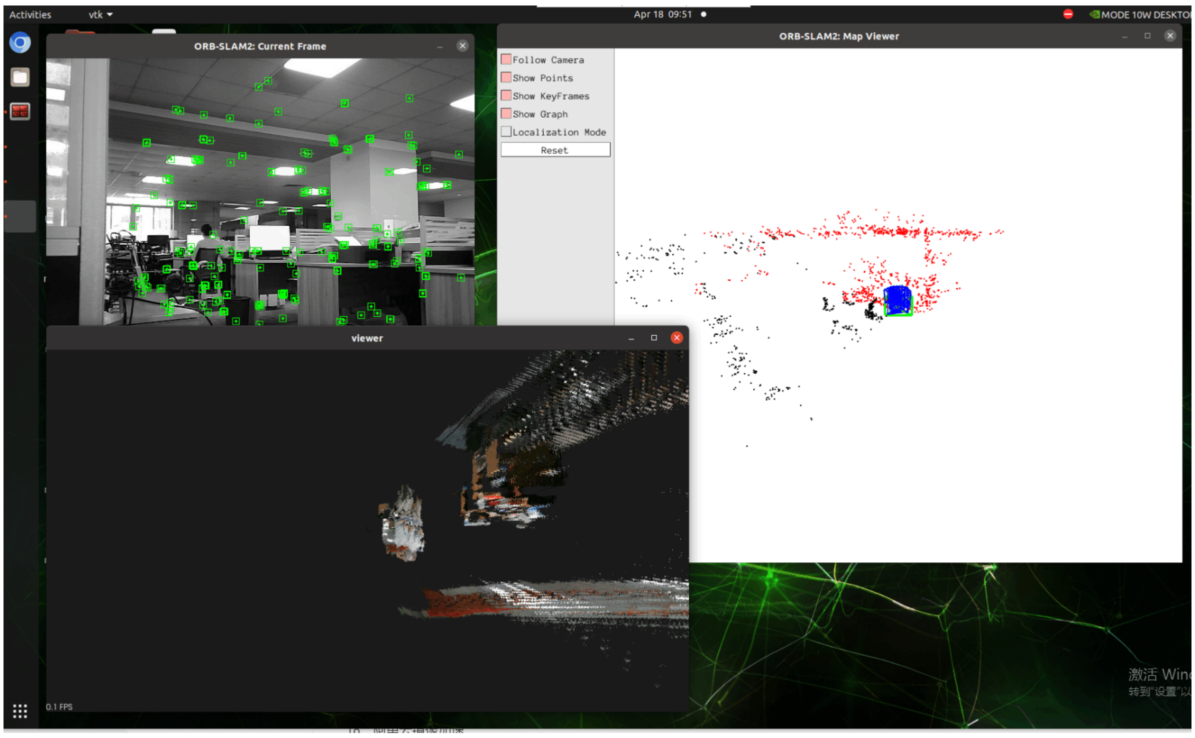
Right click on the cursor: zoom in and out.

You can open rviz at the same time to view the real-time point cloud construction effect. Considering the CPU and memory overhead during the point cloud construction process, it is recommended to open rviz on the virtual machine side to view it:

```
ros2 launch yahboomcar_slam display_pcl_launch.py
```


/home/rr/rviz/orbslam_pcl.rviz* - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Goal Pose Publish Point
Displays
Add Duplicate Remove Rename

```
Global Options
  Fixed Frame             odom
  Background Color        48; 48; 48
  Frame Rate              30
Global Status: Ok
  Fixed Frame             OK
Grid                      ✓
OccupancyGrid
OccupancyMap
RobotModel                ✓
  Status: Ok
  Visual Enabled          ✓
  Collision Enabled
  Update Interval         0
  Alpha                   1
  Description Source      Topic
  Description Topic       /robot_description
  TF Prefix
  Links
PointCloud2               ✓
  Status: Ok
  Topic                   /Local/PointCloudOutput
  Selectable              ✓
  Style                   Flat Squares
  Size (m)                0.01
  Alpha                   1
  Decay Time              0
  Position Transformer    XYZ
  Color Transformer       RGB8
PointCloud2               ✓
Path                      ✓
```



4. Slowly move the camera to build the image as shown below, finish building, close [Ctrl+c], and automatically save the pcd point cloud file resultPointCloudFile.pcd with the following path:

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_slam/pcl/resultPointCloudFile.pcd
```

5. View resultPointCloudFile.pcd file

(1) Method 1: Use the pcl_viewer tool

Install the pcl_viewer command:

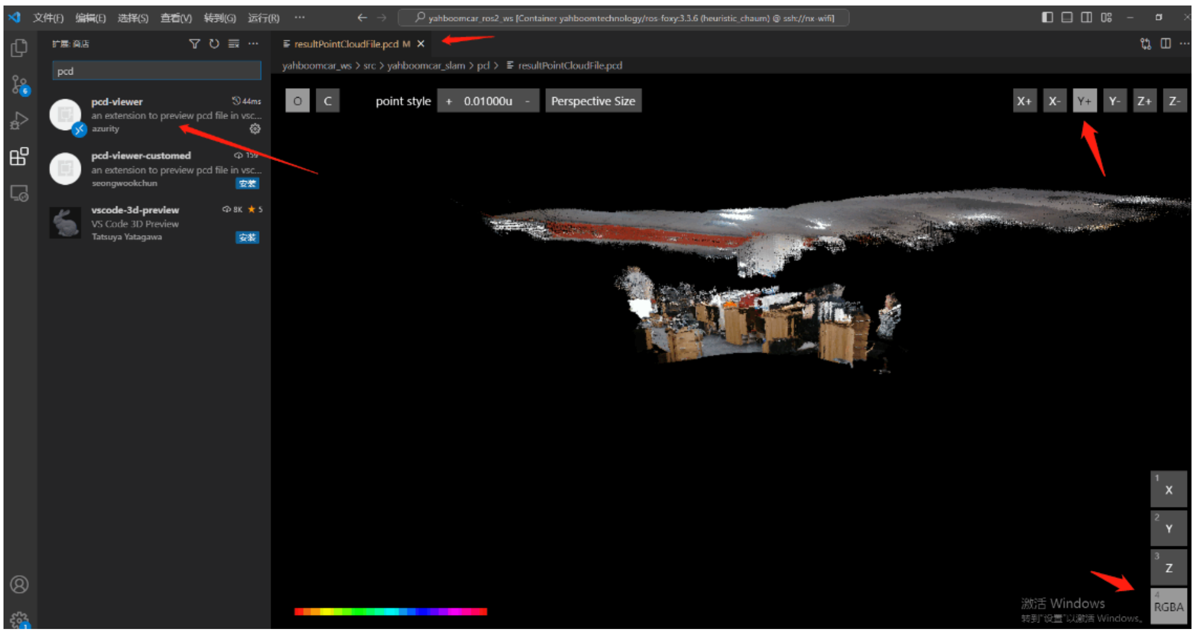
```
sudo apt-get install pcl-tools
```

Enter the directory where the pcd file is located and enter the following command to view the pcd point clouds

```
pcl_viewer resultPointCloudFile.pcd
```

(2) Method 2: Use vscode's pcd-viewer plugin [recommended].

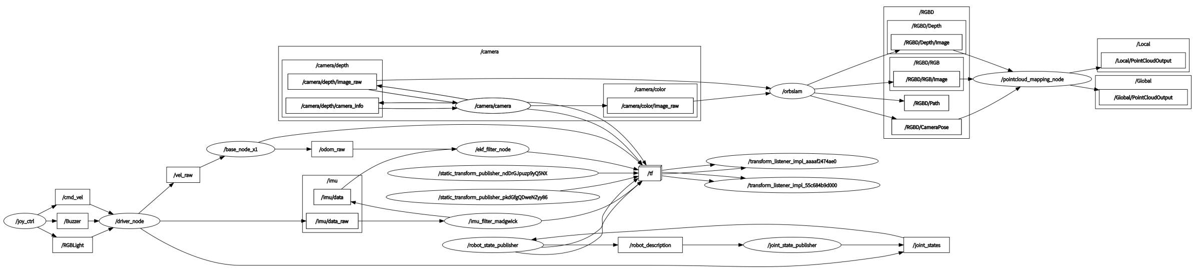
First install the pcd-viewer plugin in vscode, and then open resultPointCloudFile.pcd to view the pcd point cloud map. Here directly select the Y + direction and the bottom of the RGBA mode can be quickly rotated to the main view of the figure


文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R)
yahboomcar_ros2_ws [Container yahboomtechnology/ros-foxy:3.3.6 (heuristic_chaum) @ ssh://nx-wifi]
扩展: 商店
pcd
pcd-viewer
an extension to preview pcd file in vsc...
azurity
pcd-viewer-customed
an extension to preview pcd file in vsc...
seongwookchun
安装
vscode-3d-preview
VS Code 3D Preview
Tatsuya Yatagawa
安装
resultPointCloudFile.pcd M
yahboomcar_ws > src > yahboomcar_slam > pcl > resultPointCloudFile.pcd
O
C
point style
+ 0.01000u -
Perspective Size
X+
X-
Y+
Y-
Z+
Z-
X
Y
Z
RGBA
激活 Windows
转到"设置"以激活 Windows。

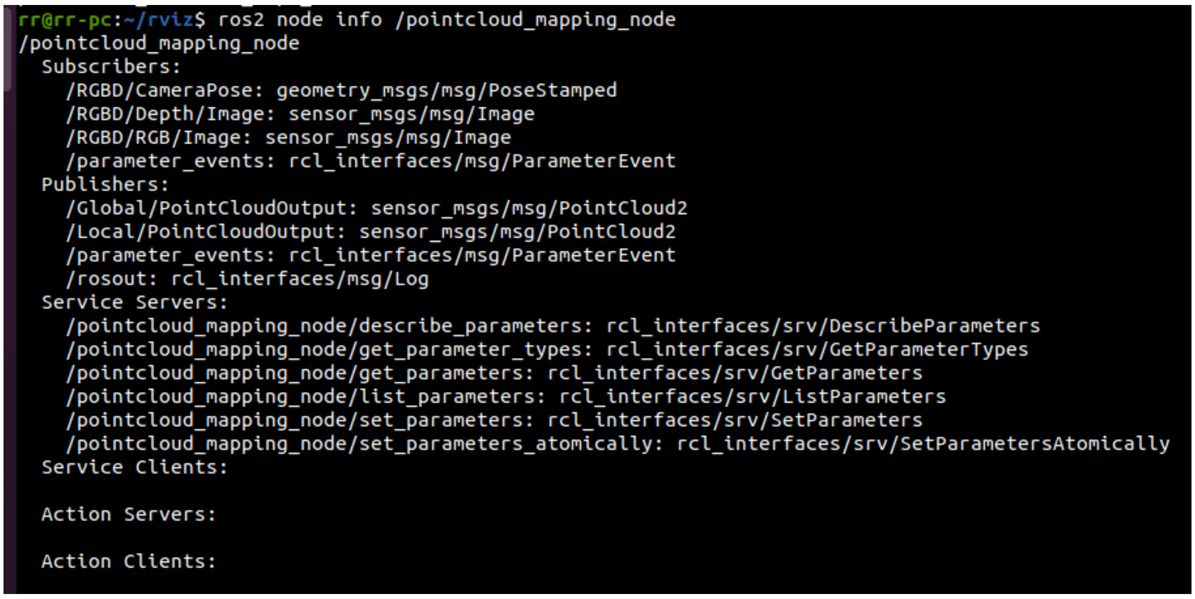
6.3. Node Resolution

6.3.1 Display of calculation diagrams

```
rqt_graph
```



6.3.2. pointcloud_mapping node details



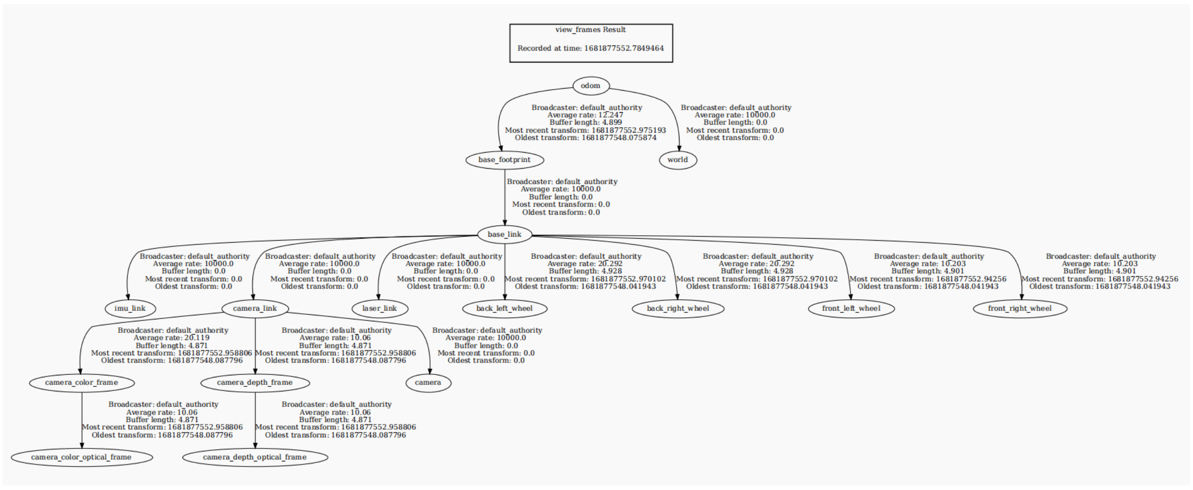
```
rr@rr-pc:~/rviz$ ros2 node info /pointcloud_mapping_node
/pointcloud_mapping_node
  Subscribers:
    /RGBD/CameraPose: geometry_msgs/msg/PoseStamped
    /RGBD/Depth/Image: sensor_msgs/msg/Image
    /RGBD/RGB/Image: sensor_msgs/msg/Image
    /parameter_events: rcl_interfaces/msg/ParameterEvent
  Publishers:
    /Global/PointCloudOutput: sensor_msgs/msg/PointCloud2
    /Local/PointCloudOutput: sensor_msgs/msg/PointCloud2
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
  Service Servers:
    /pointcloud_mapping_node/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /pointcloud_mapping_node/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /pointcloud_mapping_node/get_parameters: rcl_interfaces/srv/GetParameters
    /pointcloud_mapping_node/list_parameters: rcl_interfaces/srv/ListParameters
    /pointcloud_mapping_node/set_parameters: rcl_interfaces/srv/SetParameters
    /pointcloud_mapping_node/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:

  Action Servers:

  Action Clients:
```

6.3.3 TF transformations

```
ros2 run tf2_tools view_frames.py
```



view_frames Result
Recorded at time: 1681877552.7849464
odom
Broadcaster: default_authority Average rate: 12.247 Buffer length: 4.899 Most recent transform: 1681877552.975193 Oldest transform: 1681877548.075874
Broadcaster: default_authority Average rate: 10000.0 Buffer length: 0.0 Most recent transform: 0.0 Oldest transform: 0.0
base_footprint
world
Broadcaster: default_authority Average rate: 10000.0 Buffer length: 0.0 Most recent transform: 0.0 Oldest transform: 0.0
base_link
Broadcaster: default_authority Average rate: 10000.0 Buffer length: 0.0 Most recent transform: 0.0 Oldest transform: 0.0
Broadcaster: default_authority Average rate: 10000.0 Buffer length: 0.0 Most recent transform: 0.0 Oldest transform: 0.0
Broadcaster: default_authority Average rate: 10000.0 Buffer length: 0.0 Most recent transform: 0.0 Oldest transform: 0.0
Broadcaster: default_authority Average rate: 20.292 Buffer length: 4.928 Most recent transform: 1681877552.970102 Oldest transform: 1681877548.041943
Broadcaster: default_authority Average rate: 20.292 Buffer length: 4.928 Most recent transform: 1681877552.970102 Oldest transform: 1681877548.041943
Broadcaster: default_authority Average rate: 10.203 Buffer length: 4.901 Most recent transform: 1681877552.94256 Oldest transform: 1681877548.041943
Broadcaster: default_authority Average rate: 10.203 Buffer length: 4.901 Most recent transform: 1681877552.94256 Oldest transform: 1681877548.041943
imu_link
camera_link
laser_link
back_left_wheel
back_right_wheel
front_left_wheel
front_right_wheel
Broadcaster: default_authority Average rate: 20.119 Buffer length: 4.871 Most recent transform: 1681877552.958806 Oldest transform: 1681877548.087796
Broadcaster: default_authority Average rate: 10.06 Buffer length: 4.871 Most recent transform: 1681877552.958806 Oldest transform: 1681877548.087796
Broadcaster: default_authority Average rate: 10000.0 Buffer length: 0.0 Most recent transform: 0.0 Oldest transform: 0.0
camera_color_frame
camera_depth_frame
camera
Broadcaster: default_authority Average rate: 10.06 Buffer length: 4.871 Most recent transform: 1681877552.958806 Oldest transform: 1681877548.087796
Broadcaster: default_authority Average rate: 10.06 Buffer length: 4.871 Most recent transform: 1681877552.958806 Oldest transform: 1681877548.087796
camera_color_optical_frame
camera_depth_optical_frame