7. cartographer Mapping algorithm

7.1 Description of program functionality

After the programs on the VM and the car are started, the car is controlled by the controller or keyboard. The car uses the radar scan data to build a map and saves the map after the map is built. This process is visualized in rviz.

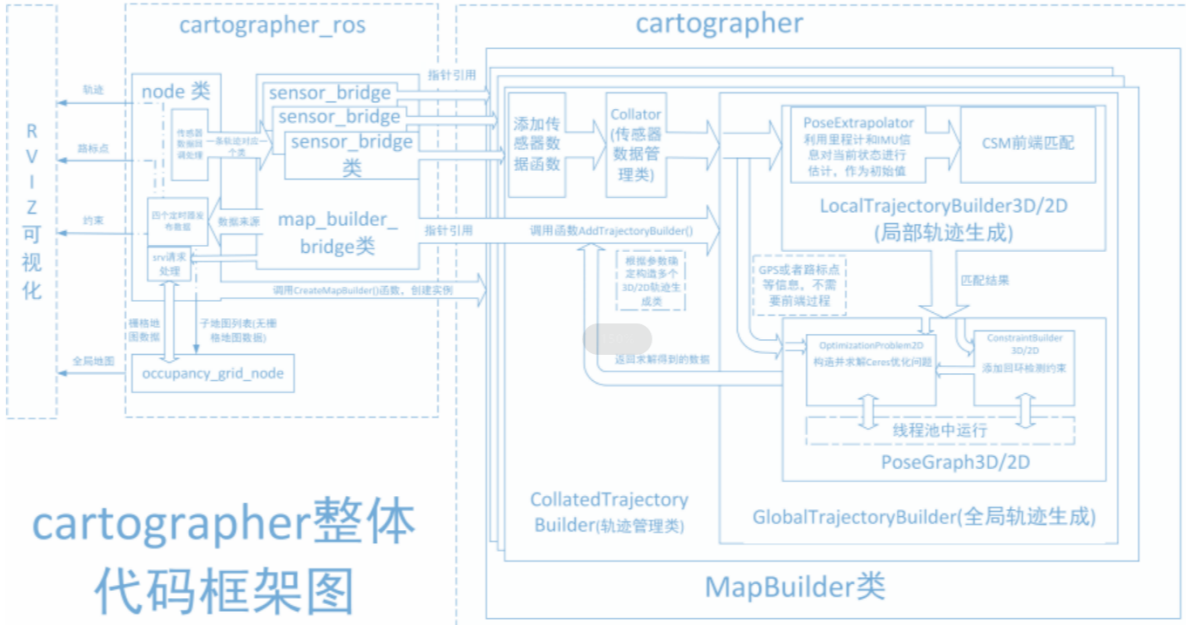
7.2 Introduction to the Cartographer

> Cartographer：https://google-cartographer.readthedocs.io/en/latest/
>
> Cartographer ROS2：https://github.com/ros2/cartographer_ros

- **Cartographer**

Cartographer is an open-source 2D and 3D SLAM (simultaneous localization and mapping) library supported by ROS systems. Method building algorithm based on graph optimization (multi-threaded backend optimization, cere constructed problem optimization). Data from multiple sensors (e.g., LIDAR, IMU, and camera) can be combined to simultaneously calculate the position of the sensor and map the environment around the sensor.

The cartographer source code consists of three main parts: cartographer, cartographer_ros, and ceres-solver (back-end optimization).



cartographer采用的是主流的SLAM框架，也就是特征提取、闭环检测、后端优化的三段式。由一定数量的LaserScan组成一个submap子图，一系列的submap子图构成了全局地图。用LaserScan构建submap的短时间过程累计误差不大，但是用submap构建全局地图的长时间过程就会存在很大的累计误差，所以需要利用闭环检测来修正这些submap的位置，闭环检测的基本单元是submap，闭环检测采用scan_match策略。cartographer的重点内容就是融合多传感器数据（odometry、IMU、LaserScan等）的submap子图创建以及用于闭环检测的scan_match策略的实现。

- **cartographer_ros**

cartographer_ros is run under ROS and can accept various sensor data in the form of ROS messages

After processing, it is published as a message for debugging and visualization.

7.3 Application Code Reference path

After SSH connection car, the location of the function source code is located at,

```
/userdata/yahboomcar_ws/src/yahboomcar_nav/launch/map_gmapping_launch.py
```

The virtual machine side source code is located at,

```
/home/yahboom/dev_ws/src/yahboomcar_rviz/launch/yahboomcar_mapping_launch.py
```

7.4. Program Startup

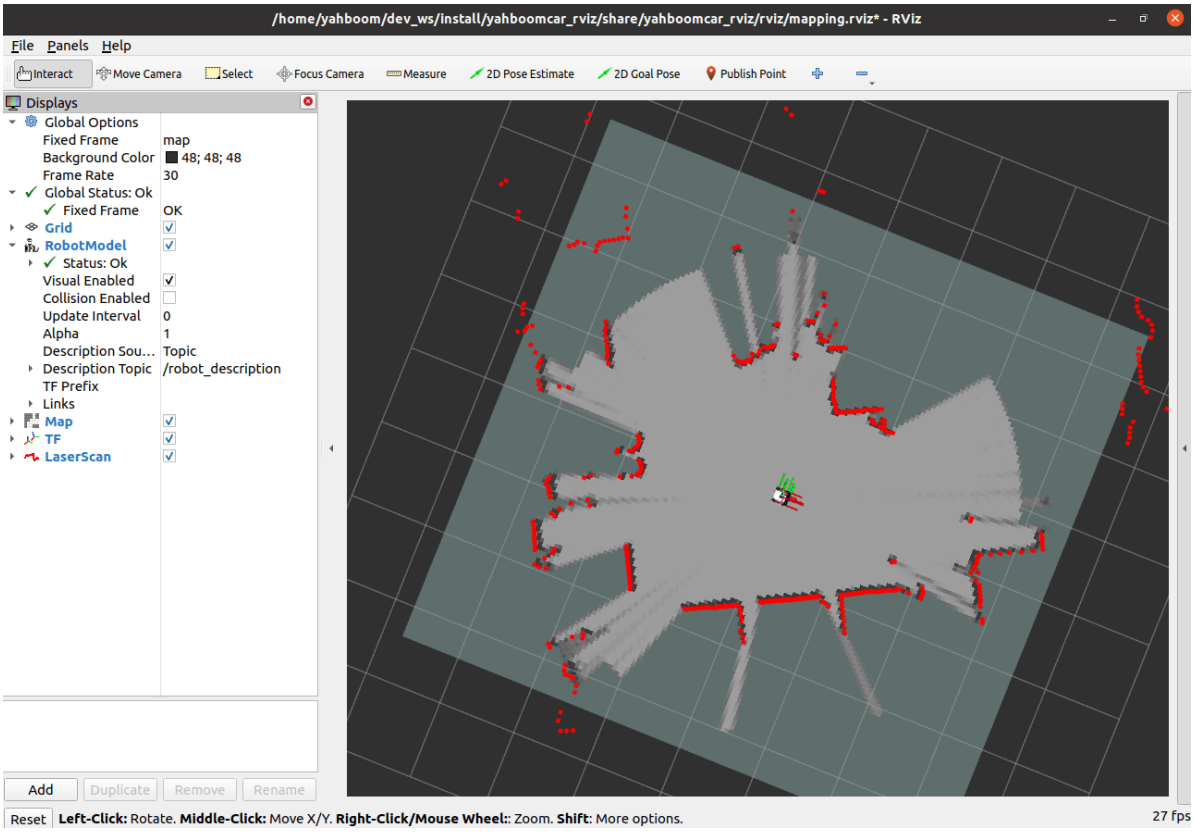
7.4.1. Build the diagram

Open the virtual machine terminal and type,

```
ros2 launch yahboomcar_rviz yahboomcar_mapping_launch.py
```

After SSH connects to the car, terminal input,

```
ros2 launch yahboomcar_nav map_cartographer_launch.py
```



Handle [L1] key unlock, control car movement. If you do not use the handle, you can use the keyboard control, the car terminal input,

```
ros2 run yahboomcar_ctrl yahboom_keyboard
```

【 Slowly move the cart 】 Start to build the map until the complete map is built.

Note: When building the drawing, the slower the better the effect (especially the rotation speed should be slower), the faster the speed, the effect will be very poor.

In addition, the path of cartographer configuration parameters is,

```
/userdata/yahboomcar_ws/src/yahboomcar_nav/params/cartographer_config.lua
```

You can modify it according to your needs, compile and run it.

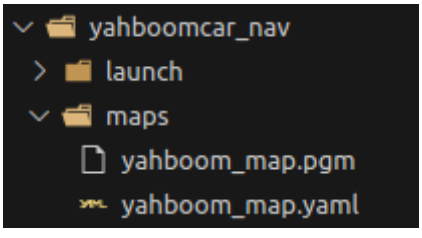
7.4.2. Map saving

Car terminal input:

```
ros2 launch yahboomcar_nav save_map_launch.py
```

The map can be saved as follows:

```
/userdata/yahboomcar_ws/src/yahboomcar_nav/maps
```



Includes a.pGM image and a.YAML file. The.yaml file contains the following contents: :

```
image: /userdata/yahboomcar_ws/src/yahboomcar_nav/maps/yahboom_map.pgm
mode: trinary
resolution: 0.05
origin: [-5.95, -3.26, 0]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.25
```

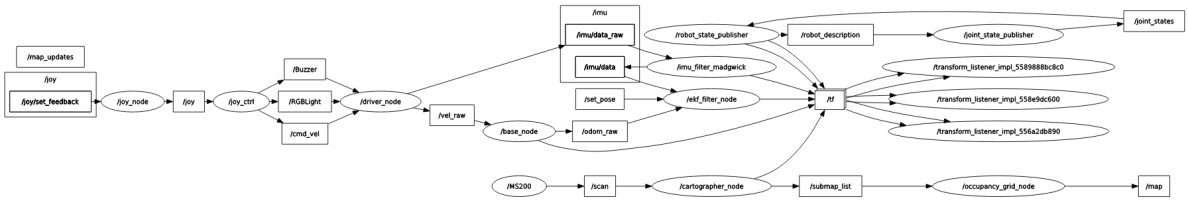
Parameter analysis:

- image: indicates the path of the map file. The path can be absolute or relative
- mode: This property can be one of trinary, scale, or raw, depending on the mode selected. trinary is the default mode
- resolution: Map resolution in meters per pixel
- origin: 2D pose (x,y,yaw) in the lower left corner of the map, where yaw is rotated counterclockwise (yaw=0 indicates no rotation). yaw values are currently ignored in many parts of the system.
- negate: Whether to reverse the meanings of white/black, free/occupied (the interpretation of the threshold is not affected)
- occupied_thresh: The occupied_thresh pixel whose occupied_probability is greater than this threshold will be considered as fully occupied.
- free_thresh: Pixels whose occupancy probability is less than this threshold are considered completely free.

7.4.3. View the topic communication graph

Virtual machine terminal input,

```
ros2 run rqt_graph rqt_graph
```



To view the cartographer node details, enter,

```
ros2 node info /cartographer_node
ros2 node info /occupancy_grid_node
```

```
yahboom@yahboom-virtual-machine:~$ ros2 node info /cartographer_node
/cartographer_node
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /scan: sensor_msgs/msg/LaserScan
  Publishers:
    /constraint_list: visualization_msgs/msg/MarkerArray
    /landmark_poses_list: visualization_msgs/msg/MarkerArray
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
    /scan_matched_points2: sensor_msgs/msg/PointCloud2
    /submap_list: cartographer_ros_msgs/msg/SubmapList
    /tf: tf2_msgs/msg/TFMessage
    /trajectory_node_list: visualization_msgs/msg/MarkerArray
  Service Servers:
    /cartographer_node/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /cartographer_node/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /cartographer_node/get_parameters: rcl_interfaces/srv/GetParameters
    /cartographer_node/list_parameters: rcl_interfaces/srv/ListParameters
    /cartographer_node/set_parameters: rcl_interfaces/srv/SetParameters
    /cartographer_node/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
    /finish_trajectory: cartographer_ros_msgs/srv/FinishTrajectory
    /start_trajectory: cartographer_ros_msgs/srv/StartTrajectory
    /submap_query: cartographer_ros_msgs/srv/SubmapQuery
    /write_state: cartographer_ros_msgs/srv/WriteState
  Service Clients:

  Action Servers:

  Action Clients:
```

```
yahboom@yahboom-virtual-machine:~$ ros2 node info /occupancy_grid_node
/occupancy_grid_node
  Subscribers:
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /submap_list: cartographer_ros_msgs/msg/SubmapList
  Publishers:
    /map: nav_msgs/msg/OccupancyGrid
    /parameter_events: rcl_interfaces/msg/ParameterEvent
    /rosout: rcl_interfaces/msg/Log
  Service Servers:
    /occupancy_grid_node/describe_parameters: rcl_interfaces/srv/DescribeParameters
    /occupancy_grid_node/get_parameter_types: rcl_interfaces/srv/GetParameterTypes
    /occupancy_grid_node/get_parameters: rcl_interfaces/srv/GetParameters
    /occupancy_grid_node/list_parameters: rcl_interfaces/srv/ListParameters
    /occupancy_grid_node/set_parameters: rcl_interfaces/srv/SetParameters
    /occupancy_grid_node/set_parameters_atomically: rcl_interfaces/srv/SetParametersAtomically
  Service Clients:
    /submap_query: cartographer_ros_msgs/srv/SubmapQuery
  Action Servers:

  Action Clients:
```

7.4.4 View the TF tree

Open the virtual machine terminal and type,

```
#保存tf树# Save tf tree
ros2 run tf2_tools view_frames.py
#查看tf树#View  tf tree
evince frames.pdf
```