Gmapping mapping

This chapter needs to be used with a car chassis and other sensors to operate. This is just an explanation of the implementation method. In fact, it cannot be run. It needs to be used with the company's rosmaster-X3 car to achieve this function.

If you need to transplant it to your own motherboard, you need to install other dependency files.

1. Function description

After the program is started, control the car through the handle or keyboard.

The car will use radar scanning data during movement, use the gmapping algorithm to build a map, and save the map after the construction is completed.

The process is visualized in rviz.

2. Code path

gmapping code:

The source code of this function is located in the supporting virtual machine system.

```
~/rplidar_ws/src/slam_gmapping
~/rplidar_ws/src/openslam_gmapping
```

launch code:

The source code of this function is located in the supporting virtual machine system.

```
~/ydlidar_ws/src/yahboomcar_nav/launch
```

3. Start up

3.1 Mapping

Input following command:

```
ros2 launch yahboomcar_nav map_gmapping_launch.py
```

3.2 Save map

Input following command:

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

The map saving path is as follows:

```
~/ydlidar_ws/src/yahboomcar_nav/maps
```

There is a .pgm image and a .yaml file.

The contents of the yaml file are as follows:

```
image: /home/yahboom/ydlidar_ws/src/yahboomcar_nav/maps/yahboomcar.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: The path of the map file, which can be an absolute path or a relative path.
- mode: This attribute can be one of trinary, scale or raw, depending on the selected mode. trinary mode is the default mode
- resolution: resolution of the map, meters/pixel
- Origin: 2D pose (x, y, yaw) in the lower left corner of the map. The yaw here is rotated counterclockwise (yaw=0 means no rotation). Many parts of the current system ignore the yaw value.
- negate: whether to reverse the meaning of white/black and free/occupied (the interpretation of the threshold is not affected)
- occupied_thresh: Pixels with an occupation probability greater than this threshold will be considered fully occupied.
- free_thresh: Pixels with occupancy probability less than this threshold will be considered completely free.

4. Code analysis

map_gmapping_launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node
import os
from launch.actions import IncludeLaunchDescription
from launch.launch_description_sources import PythonLaunchDescriptionSource,
from ament_index_python.packages import get_package_share_directory
from launch.conditions import
LaunchConfigurationEquals,LaunchConfigurationNotEquals
from launch.actions import DeclareLaunchArgument


def generate_launch_description():
    RPLIDAR_TYPE = os.getenv('LIDAR_TYPE')
    rplidar_type_arg = DeclareLaunchArgument(name='rplidar_type',
default_value=RPLIDAR_TYPE,
                                             description='The type of robot')


    gmapping_s2_launch = IncludeLaunchDescription(PythonLaunchDescriptionSource(
        [os.path.join(get_package_share_directory('yahboomcar_nav'), 'launch'),
        '/map_gmapping_4ros_s2_launch.py']),
        condition=LaunchConfigurationEquals('rplidar_type', 's2')
    )
```

```
    gmapping_a1_launch = IncludeLaunchDescription(PythonLaunchDescriptionSource(
        [os.path.join(get_package_share_directory('yahboomcar_nav'), 'launch'),
        '/map_gmapping_a1_launch.py']),
        condition=LaunchConfigurationNotEquals('rplidar_type', 's2')
    )

    return LaunchDescription([
        rplidar_type_arg,
        gmapping_s2_launch,
        gmapping_a1_launch
    ])
```

laser_bringup_gmapping_launch.py starts the radar chassis,

slam_gmapping.launch.py starts gmapping mapping related nodes.

save_map_launch.py

```
from ament_index_python.packages import get_package_share_path

from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument
from launch.substitutions import LaunchConfiguration

from launch_ros.actions import Node
import os


def generate_launch_description():
    package_share_path = str(get_package_share_path('yahboomcar_nav'))
    package_path = os.path.abspath(os.path.join(
        package_share_path, "../../../../src/yahboomcar_nav"))
    map_name = "yahboom_map"
    default_map_path = os.path.join(package_path, 'maps', map_name)

    map_arg = DeclareLaunchArgument(name='map_path',
default_value=str(default_map_path),
                                    description='The path of the map')

    map_saver_node = Node(
        package='nav2_map_server',
        executable='map_saver_cli',
        arguments=[
            '-f', LaunchConfiguration('map_path'), '--ros-args', '-p',
'save_map_timeout:=60000'],
    )

    return LaunchDescription([
        map_arg,
        map_saver_node
    ])
```

We need to use a nav2_map_server function package.

Taking the ros2-foxy version as an example, enter the following command in the terminal to install the nav2_map_server function package

```
sudo apt install ros-foxy-nav2-* -y
```