IMU Fusion Radar Mapping and Navigation

1. Source Code Description

We provide two sets of IMU function source code: one is the source code for the IMU driver only, and the other is the source code for the IMU fusion radar mapping and navigation example.

imu_ros2_device: This is the source code for the IMU driver only.

yahboomcar_ws: This is the source code for the IMU fusion radar mapping and navigation example.

Generally, the imu_ros2_device package is sufficient for installation on your own motherboard. The other packages shown in the examples require some dependencies, and errors may occur during installation and compilation. Users need to resolve environment dependency issues themselves. The **source code here is for reference only.** It will not run if the corresponding vehicle and radar equipment are not provided. The source code is implemented using our company's Rosmaster A1 vehicle product. Please be aware of this.

2. Radar Mapping

2.1 Pre-Use Configuration

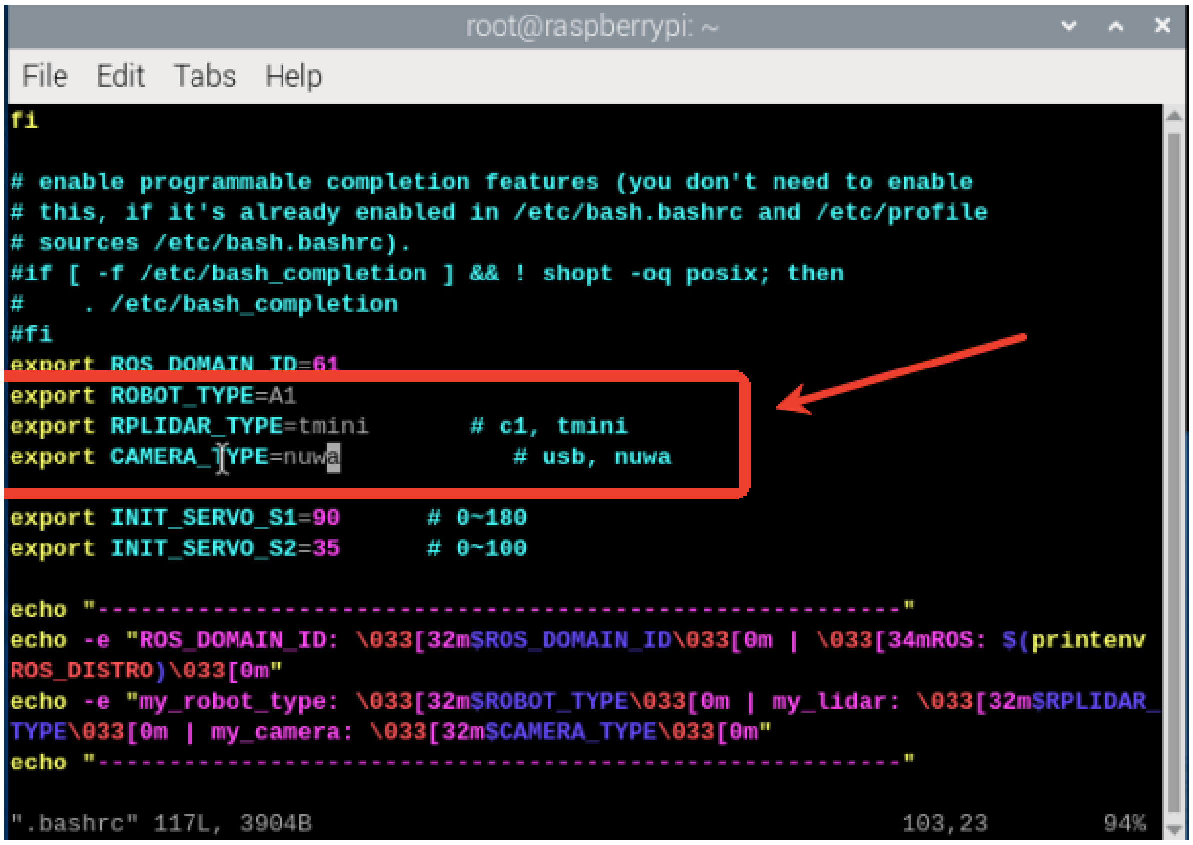
This vehicle model is equipped with a USB camera, a depth camera, and two different models of LiDAR. However, since the products cannot be automatically recognized, the machine type and radar model need to be manually set.

Based on the vehicle model, the radar type and camera type should be modified as follows:

```
root@ubuntu:/# cd
root@ubuntu:~# vim .bashrc
```

```
root@raspberrypi:/# cd
root@raspberrypi:~# vim .bashrc
```

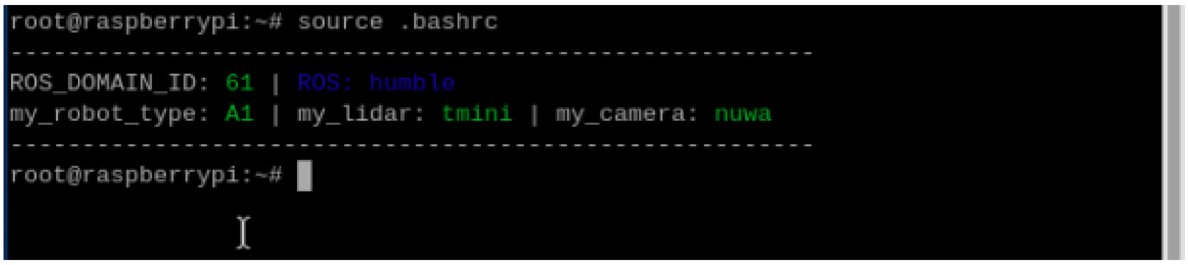
Find this location, press the 'i' key on the keyboard, and change it to the corresponding camera and radar model. Here, the default is tmini and nuwa.



```
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#    . /etc/bash_completion
#fi
export ROS_DOMAIN_ID=61
export ROBOT_TYPE=A1
export RPLIDAR_TYPE=tmini          # c1, tmini
export CAMERA_TYPE=nuwa               # usb, nuwa

export INIT_SERVO_S1=90      # 0~180
export INIT_SERVO_S2=35      # 0~100

echo "-------------------------------------------------------"
echo -e "ROS_DOMAIN_ID: \033[32m$ROS_DOMAIN_ID\033[0m | \033[34mROS: $(printenv ROS_DISTRO)\033[0m"
echo -e "my_robot_type: \033[32m$ROBOT_TYPE\033[0m | my_lidar: \033[32m$RPLIDAR_TYPE\033[0m | my_camera: \033[32m$CAMERA_TYPE\033[0m"
echo "-------------------------------------------------------"

".bashrc" 117L, 3904B                                      103,23        94%
```

After making the changes, save and exit Vim, then execute:

```
root@raspberrypi:~# source .bashrc
-------------------------------------------------------
ROS_DOMAIN_ID: 61 | ROS: humble
my_robot_type: A1 | my_lidar: tmini | my_camera: nuwa
-------------------------------------------------------
root@raspberrypi:~#
```



```
root@raspberrypi:~# source .bashrc
-------------------------------------------------------
ROS_DOMAIN_ID: 61 | ROS: humble
my_robot_type: A1 | my_lidar: tmini | my_camera: nuwa
-------------------------------------------------------
root@raspberrypi:~#
```

2.2 Program Startup

Start Map Building

The following map building algorithm can only be selected from two options; they cannot be started simultaneously.

Gamapping

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

cartographer

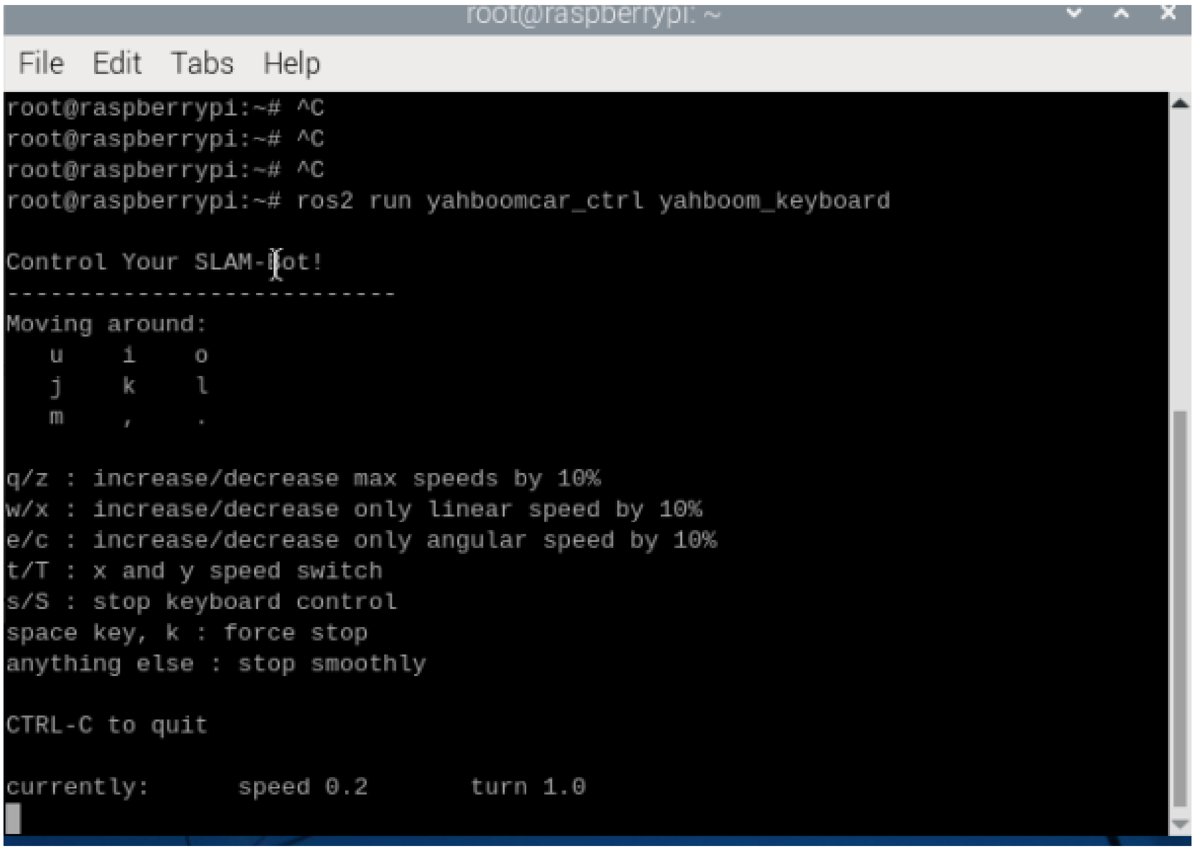
```
ros2 launch yahboomcar_nav map_cartographer_launch.py
```

Input command to start rviz visual mapping

```
ros2 launch yahboomcar_nav display_map_launch.py
```

The program has gamepad control enabled by default. If you are using a gamepad, you can directly connect to the receiver for control. If you want to use keyboard control, enter the following in the terminal:

```
ros2 run yahboomcar_ctrl yahboom_keyboard
```



```
root@raspberrypi:~# ^C
root@raspberrypi:~# ^C
root@raspberrypi:~# ^C
root@raspberrypi:~# ros2 run yahboomcar_ctrl yahboom_keyboard

Control Your SLAM-Bot!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
t/T : x and y speed switch
s/S : stop keyboard control
space key, k : force stop
anything else : stop smoothly

CTRL-C to quit

currently:      speed 0.2       turn 1.0
```

Then, control the car to slowly traverse the area requiring mapping. After mapping is complete, enter the following command to save the map:

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

This will save a map named yahboomcar in:

~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_nav/maps/yahboomcar.yaml

Two files will be generated: yahboomcar.pgm and yahboomcar.yaml.

Let's look at the contents of yahboom_map.yaml:

```
image: yahboomcar.pgm
mode: trinary
resolution: 0.05
origin: [-9.02, -15.5, 0]
negate: 0
occupied_thresh: 0.65
free_thresh: 0.25
```

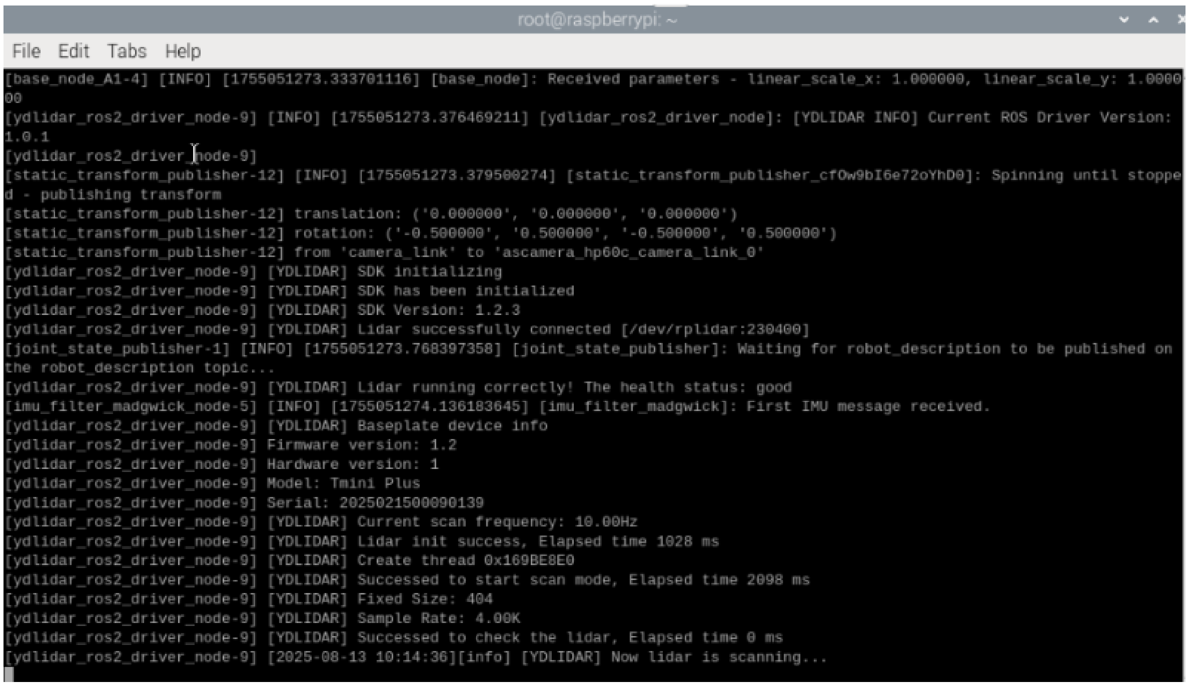
- image: Represents the map image, i.e., yahboomcar.pgm

- mode: This attribute can be one of trinary, scale, or raw, depending on the selected mode. Trinary mode is the default mode.
- resolution: Map resolution, meters per pixel
- origin: 2D pose (x, y, yaw) of the bottom left corner of the map. Here, yaw is a counter-clockwise rotation (yaw=0 means no rotation). Currently, many parts of the system ignore the yaw value.
- negate: Whether to reverse the meaning of white/black, free/occupied (the interpretation of the threshold is not affected)
- occupied_thresh: Pixels with an occupation probability greater than this threshold are considered fully occupied.
- free_thresh: Pixels with an occupation probability less than this threshold are considered completely free.

3. Navigation and Obstacle Avoidance

Robot vehicle terminal launches underlying sensor commands:

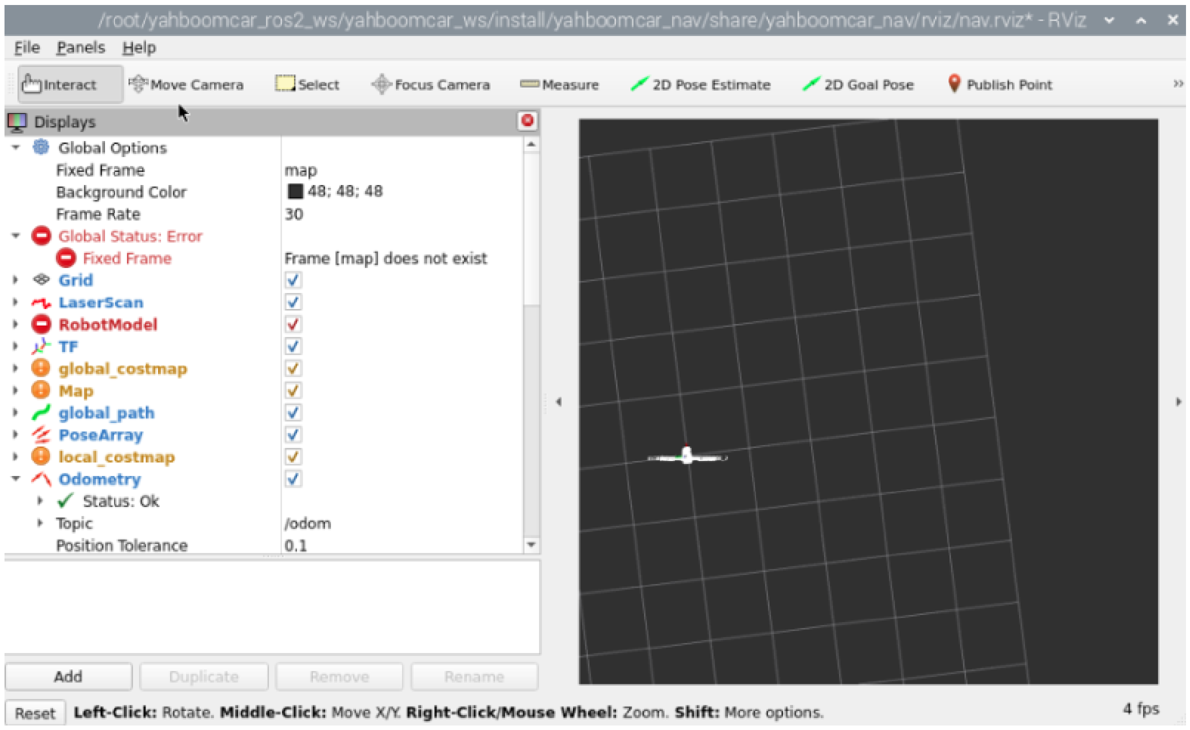
```
ros2 launch yahboomcar_nav laser_bringup_launch.py
```



```
[base_node_A1-4] [INFO] [1755051273.333701116] [base_node]: Received parameters - linear_scale_x: 1.000000, linear_scale_y: 1.0000
00
[ydlidar_ros2_driver_node-9] [INFO] [1755051273.376469211] [ydlidar_ros2_driver_node]: [YDLIDAR INFO] Current ROS Driver Version:
1.0.1
[ydlidar_ros2_driver_node-9]
[static_transform_publisher-12] [INFO] [1755051273.379500274] [static_transform_publisher_cfOw9bI6e72oYhD0]: Spinning until stoppe
d - publishing transform
[static_transform_publisher-12] translation: ('0.000000', '0.000000', '0.000000')
[static_transform_publisher-12] rotation: ('-0.500000', '0.500000', '-0.500000', '0.500000')
[static_transform_publisher-12] from 'camera_link' to 'ascamera_hp60c_camera_link_0'
[ydlidar_ros2_driver_node-9] [YDLIDAR] SDK initializing
[ydlidar_ros2_driver_node-9] [YDLIDAR] SDK has been initialized
[ydlidar_ros2_driver_node-9] [YDLIDAR] SDK Version: 1.2.3
[ydlidar_ros2_driver_node-9] [YDLIDAR] Lidar successfully connected [/dev/rplidar:230400]
[joint_state_publisher-1] [INFO] [1755051273.768397358] [joint_state_publisher]: Waiting for robot_description to be published on
the robot_description topic...
[ydlidar_ros2_driver_node-9] [YDLIDAR] Lidar running correctly! The health status: good
[imu_filter_madgwick_node-5] [INFO] [1755051274.136183645] [imu_filter_madgwick]: First IMU message received.
[ydlidar_ros2_driver_node-9] [YDLIDAR] Baseplate device info
[ydlidar_ros2_driver_node-9] Firmware version: 1.2
[ydlidar_ros2_driver_node-9] Hardware version: 1
[ydlidar_ros2_driver_node-9] Model: Tmini Plus
[ydlidar_ros2_driver_node-9] Serial: 2025021500090139
[ydlidar_ros2_driver_node-9] [YDLIDAR] Current scan frequency: 10.00Hz
[ydlidar_ros2_driver_node-9] [YDLIDAR] Lidar init success, Elapsed time 1028 ms
[ydlidar_ros2_driver_node-9] [YDLIDAR] Create thread 0x169BE8E0
[ydlidar_ros2_driver_node-9] [YDLIDAR] Successed to start scan mode, Elapsed time 2098 ms
[ydlidar_ros2_driver_node-9] [YDLIDAR] Fixed Size: 404
[ydlidar_ros2_driver_node-9] [YDLIDAR] Sample Rate: 4.00K
[ydlidar_ros2_driver_node-9] [YDLIDAR] Successed to check the lidar, Elapsed time 0 ms
[ydlidar_ros2_driver_node-9] [2025-08-13 10:14:36][info] [YDLIDAR] Now lidar is scanning...
```

Input command to launch RViz visualization mapping (RViz visualization can be launched on either the vehicle terminal or the virtual machine terminal; **choose one**, but do not launch it repeatedly on both the virtual machine and the vehicle terminal :), the image below shows the input on the vehicle terminal:

```
ros2 launch yahboomcar_nav display_nav_launch.py
```



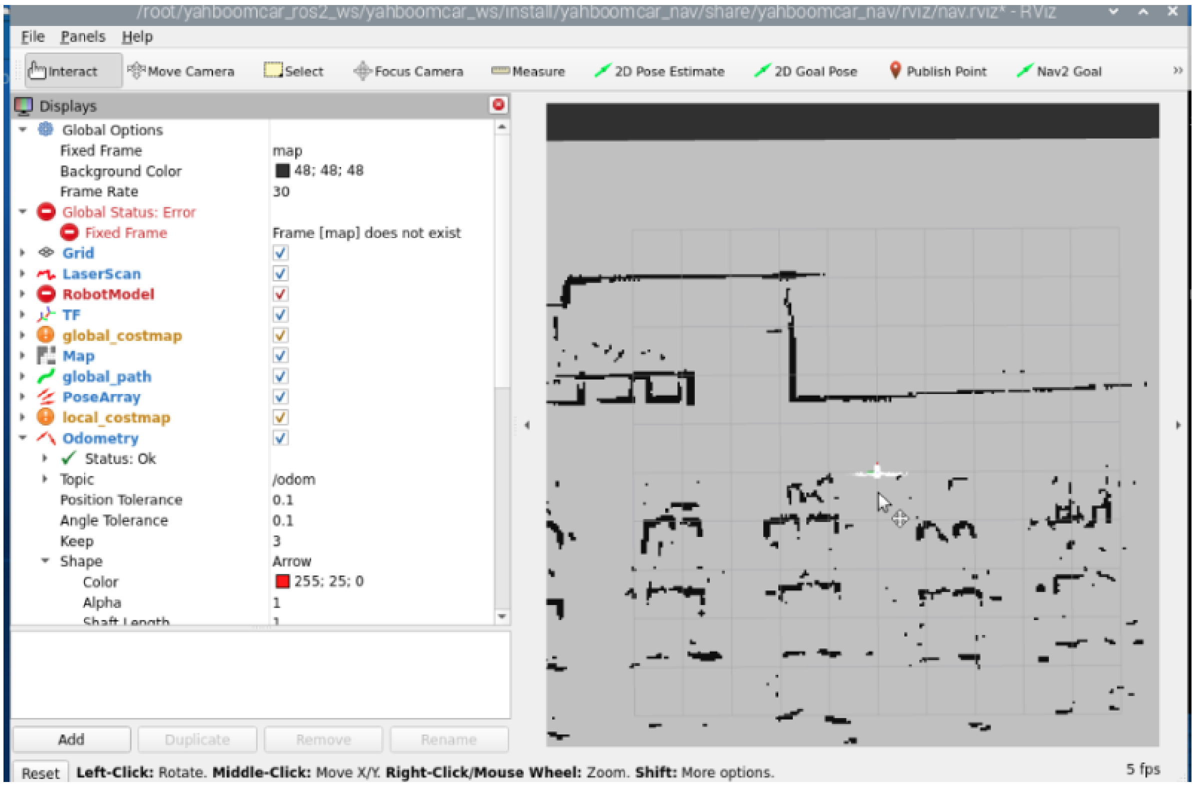
/root/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_nav/share/yahboomcar_nav/rviz/nav.rviz* - RViz

```
Global Options
  Fixed Frame          map
  Background Color     48; 48; 48
  Frame Rate           30
Global Status: Error
  Fixed Frame          Frame [map] does not exist
Grid                   ✓
LaserScan              ✓
RobotModel             ✓
TF                     ✓
global_costmap         ✓
Map                    ✓
global_path            ✓
PoseArray              ✓
local_costmap          ✓
Odometry               ✓
  Status: Ok
  Topic                /odom
  Position Tolerance   0.1
```

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.

At this point, the map is not yet loaded because the navigation program has not been started. Next, run the navigation node by entering the following in the terminal:

```
ros2 launch yahboomcar_nav navigation_teb_launch.py
```

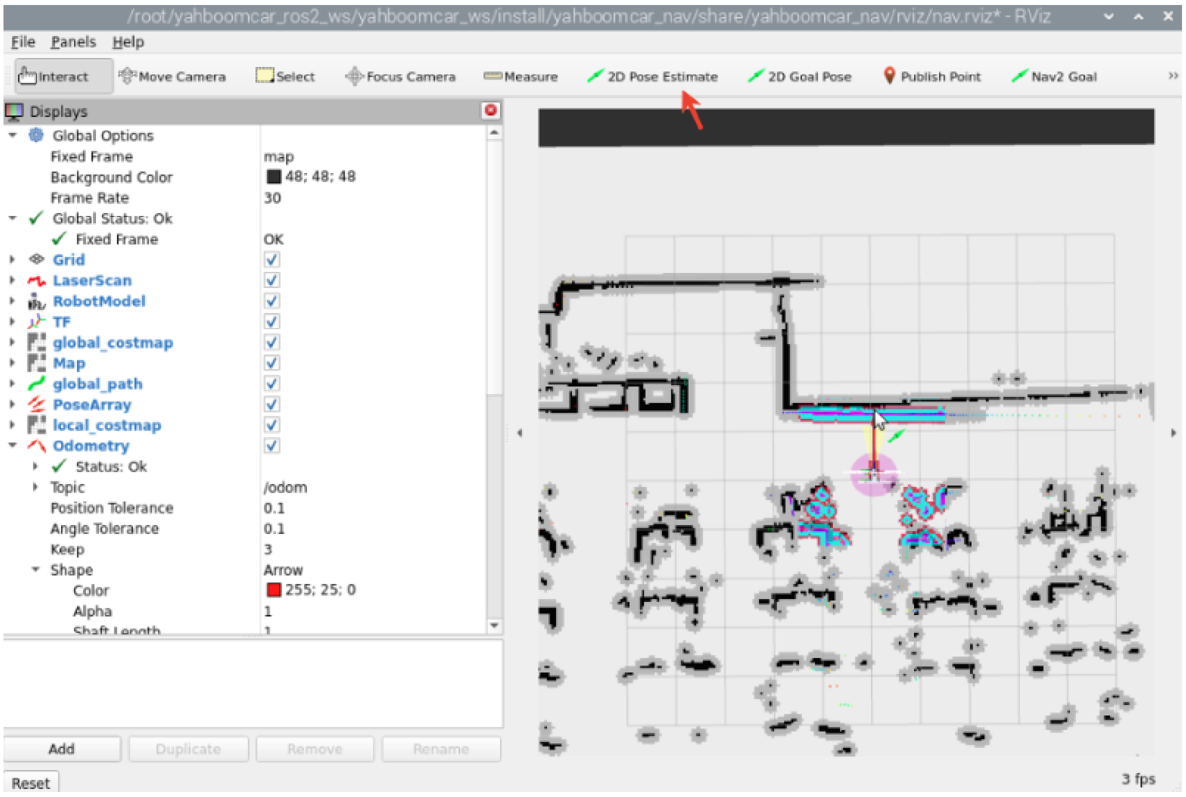


/root/yahboomcar_ros2_ws/yahboomcar_ws/install/yahboomcar_nav/share/yahboomcar_nav/rviz/nav.rviz* - RViz

```
Global Options
  Fixed Frame          map
  Background Color     48; 48; 48
  Frame Rate           30
Global Status: Error
  Fixed Frame          Frame [map] does not exist
Grid                   ✓
LaserScan              ✓
RobotModel             ✓
TF                     ✓
global_costmap         ✓
Map                    ✓
global_path            ✓
PoseArray              ✓
local_costmap          ✓
Odometry               ✓
  Status: Ok
  Topic                /odom
  Position Tolerance   0.1
  Angle Tolerance      0.1
  Keep                 3
  Shape                Arrow
    Color              255; 25; 0
    Alpha              1
    Shaft Length       1
```

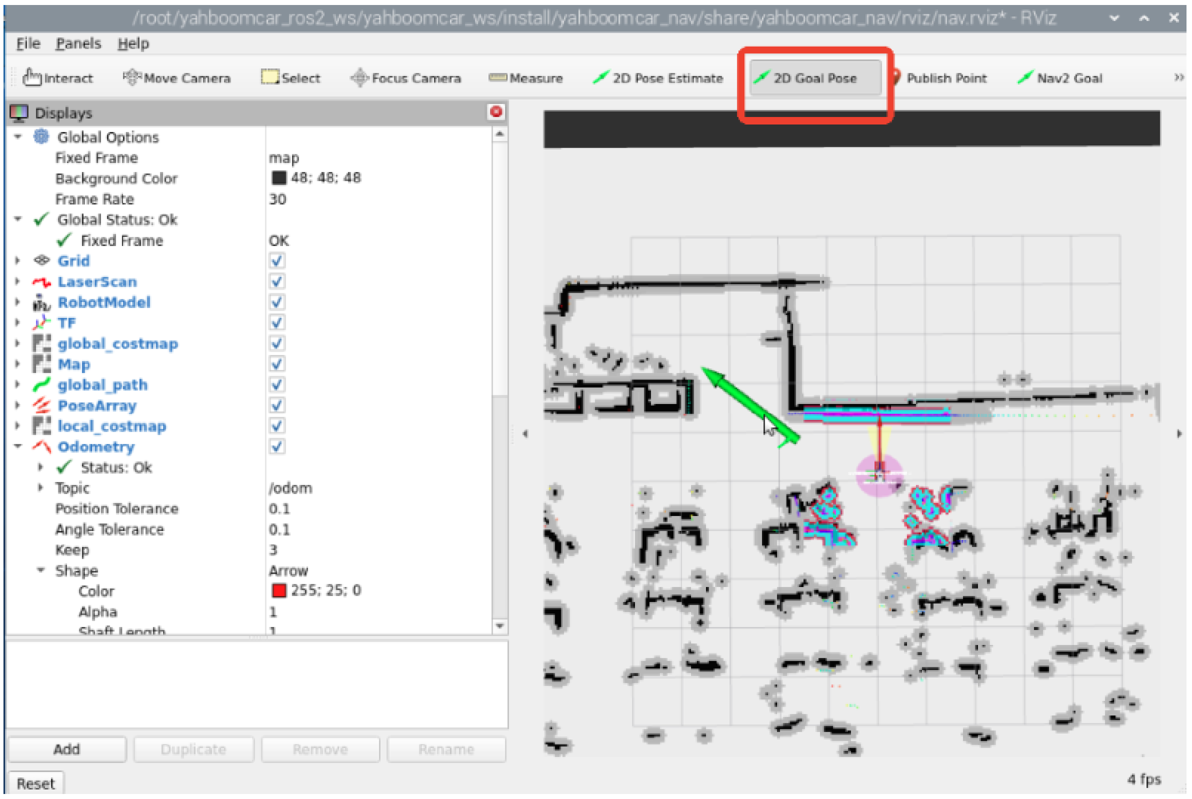
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.

Now you can see the map has loaded. Next, click on "2D Pose Estimate" to set the initial pose for the car. Based on the car's position in the actual environment, use the mouse to click and drag in RViz to move the car model to the position we set. As shown in the image below, if the area scanned by the radar roughly overlaps with the actual obstacles, the pose is accurate. After pose initialization, the robot model and the expanded region will appear in the RViz interface.



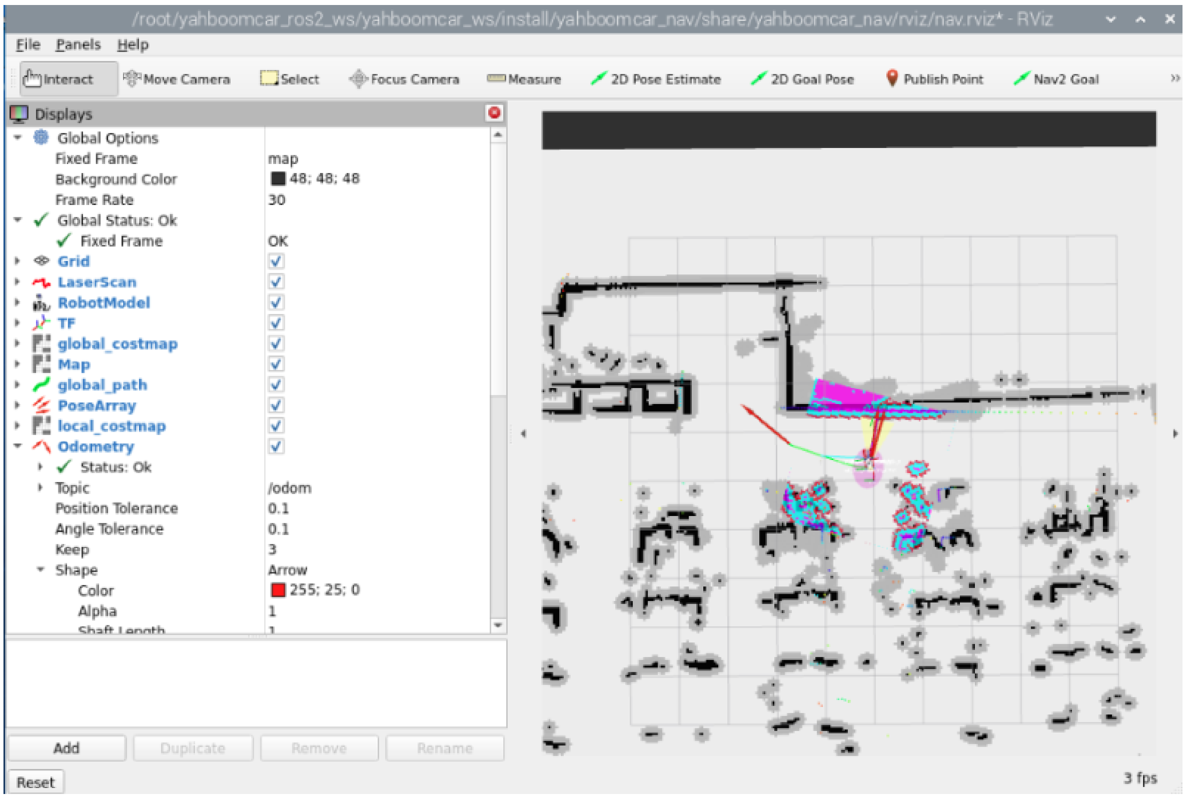
Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
Global Status: Ok	
Fixed Frame	OK
Grid	✓
LaserScan	✓
RobotModel	✓
TF	✓
global_costmap	✓
Map	✓
global_path	✓
PoseArray	✓
local_costmap	✓
Odometry	✓
Status: Ok	
Topic	/odom
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	3
Shape	Arrow
Color	🟥 255; 25; 0
Alpha	1
Shaft Length	1

For single-point navigation, click the "2D Goal Pose" tool, then use the mouse in RViz to select a target point and target orientation, and then release.



Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
Global Status: Ok	
Fixed Frame	OK
Grid	✓
LaserScan	✓
RobotModel	✓
TF	✓
global_costmap	✓
Map	✓
global_path	✓
PoseArray	✓
local_costmap	✓
Odometry	✓
Status: Ok	
Topic	/odom
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	3
Shape	Arrow
Color	🟥 255; 25; 0
Alpha	1
Shaft Length	1

The robot, considering its surroundings, plans a path and moves along it to the target point.



Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
LaserScan	✓
RobotModel	✓
TF	✓
global_costmap	✓
Map	✓
global_path	✓
PoseArray	✓
local_costmap	✓
Odometry	✓
✓ Status: Ok	
Topic	/odom
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	3
Shape	Arrow
Color	🟥 255; 25; 0
Alpha	1
Shaft Length	1

Once the robot successfully reaches the target point, the vehicle terminal will display "Goal succeeded," indicating successful navigation.