2. Multi-vehicle navigation

2.1. Program function description

After the virtual machine and the car-side program are started, after calibrating the positions of car 1 and car 2 on the map in the virtual machine rviz, and setting the target points respectively, the car will automatically navigate to the specified location.

2.2. Program reference path

Raspberry Pi PI5 master needs to enter the docker container first, Orin master does not need to enter,

the source code of this function is located at

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_multi/launch
```

The virtual machine source code is located at

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_rviz/launch/display_multi_nav_launch.py
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_multi/launch/map_server_launch.py
```

2.3. Program startup

Note: Multi-machine communication needs to be set up in advance here, that is, the virtual machine and the car (taking two cars as an example) need to be on the same LAN and the ROS_DOMAIN_ID must be the same before they can be distributed communication. Setting method reference: [06] Development environment setup -> [4], multi-machine communication configuration, set up and verify.

The virtual machine is running rviz2, visual navigation and loading the navigation map, so you need to first place the navigation map in the following location in the virtual machine,

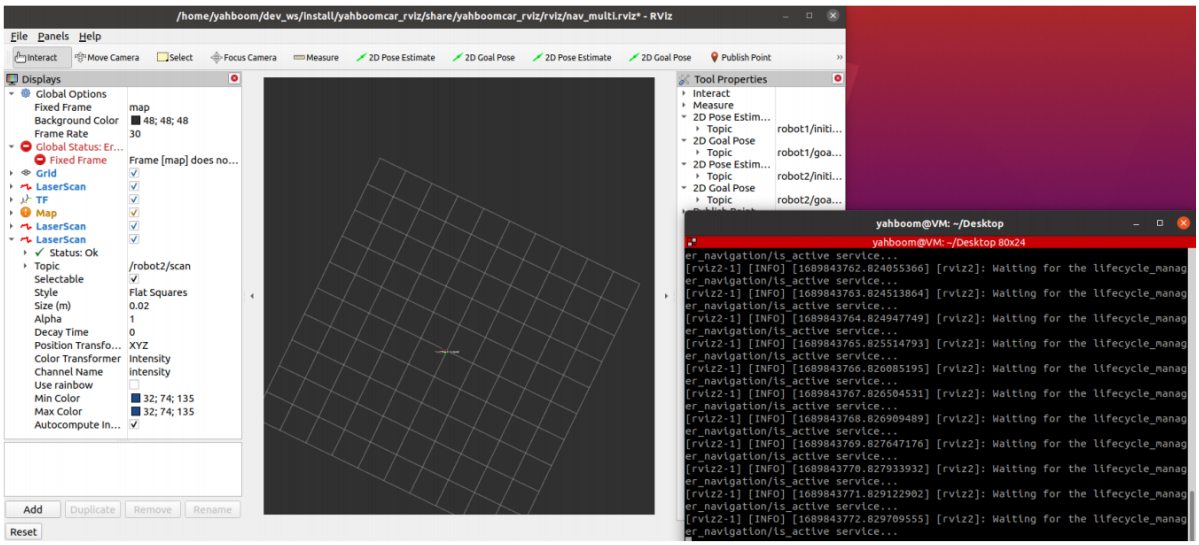
```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_multi/maps
```

yahboom_map.yaml is the startup map parameter file loaded by the program, and yahboom_map.pgm is the map image loaded by the map parameter file.

2.3.1. Start RVIZ on the virtual machine

Virtual machine terminal input,

```
ros2 launch yahboomcar_rviz display_multi_nav_launch.py
```



2.3.2. Start the two car chassis

After entering the docker containers of Car 1 and Car 2 respectively, enter in the terminal according to the actual car model and radar type,

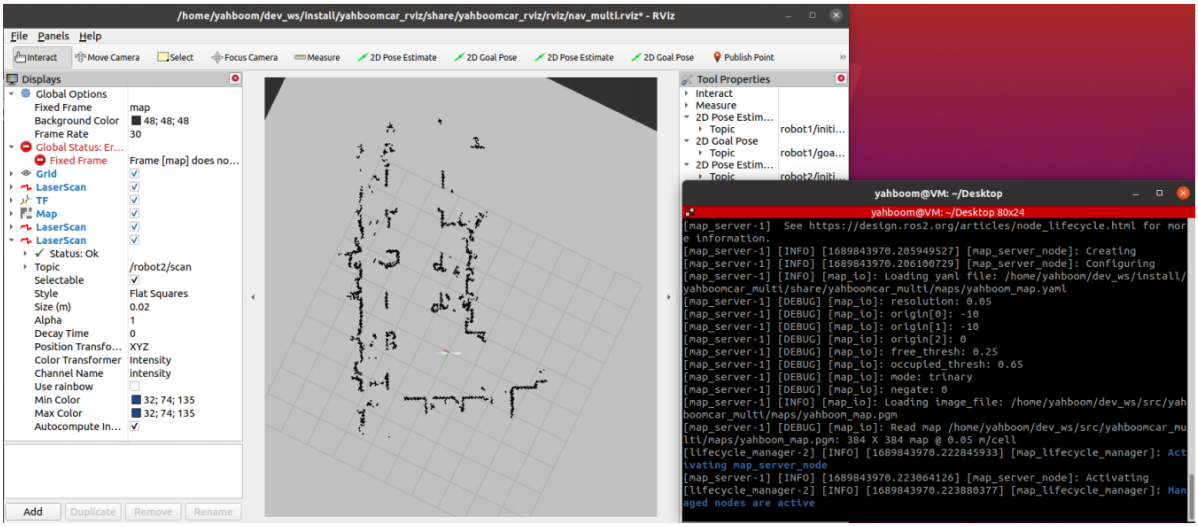
```
#X3 model, A1 radar launched
#smallcar1
ros2 launch yahboomcar_multi X3_A1_bringup_multi.launch.xml robot_name:=robot1
#littlecar2
ros2 launch yahboomcar_multi X3_A1_bringup_multi.launch.xml robot_name:=robot2

#X3 model, s2 radar started
#smallcar1
ros2 launch yahboomcar_multi X3_S2_bringup_multi.launch.xml robot_name:=robot1
#littlecar2
ros2 launch yahboomcar_multi X3_S2_bringup_multi.launch.xml robot_name:=robot2

#R2 model, A1 radar launched
#smallcar1
ros2 launch yahboomcar_multi R2_A1_bringup_multi.launch.xml robot_name:=robot1
#littlecar2
ros2 launch yahboomcar_multi R2_A1_bringup_multi.launch.xml robot_name:=robot2

#R2 model, 4ROS radar launched
#smallcar1
ros2 launch yahboomcar_multi R2_4ROS_bringup_multi.launch.xml robot_name:=robot1
#littlecar2
ros2 launch yahboomcar_multi R2_4ROS_bringup_multi.launch.xml robot_name:=robot2
```

[robot_name] indicates the serial number of the vehicle to start. Currently, the program can choose to start robot1 and robot2.

2.3.3. Load the map on the virtual machine side

```
ros2 launch yahboomcar_multi map_server_launch.py
```

The map loaded here may not be loaded in one go. If rviz does not display the image, then ctrl c closes the node and starts it several times to try.



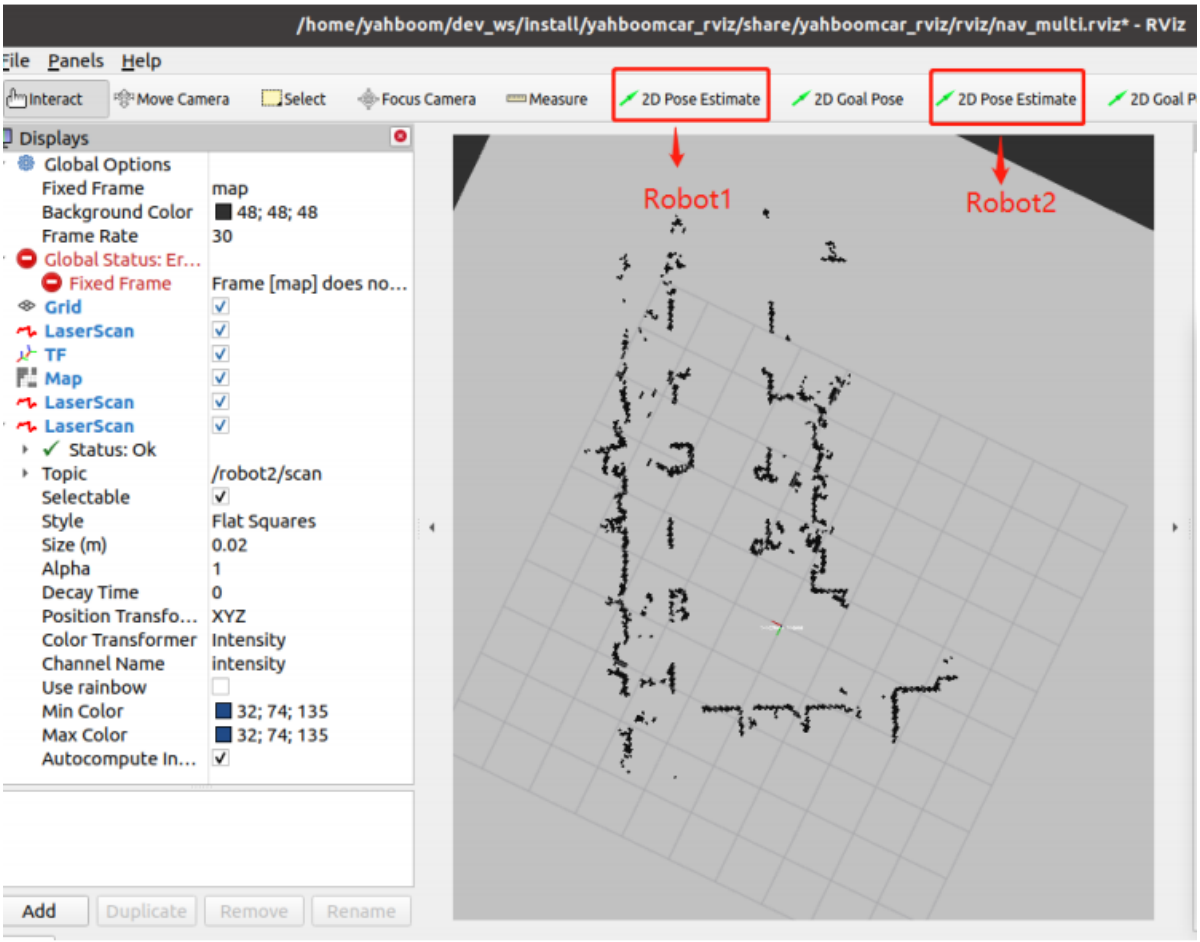
2.3.4. Two cars start AMCL positioning

Raspberry Pi 5 master enters the docker container of car 1 and car 2 respectively, **Orin master** does not need to enter docker,

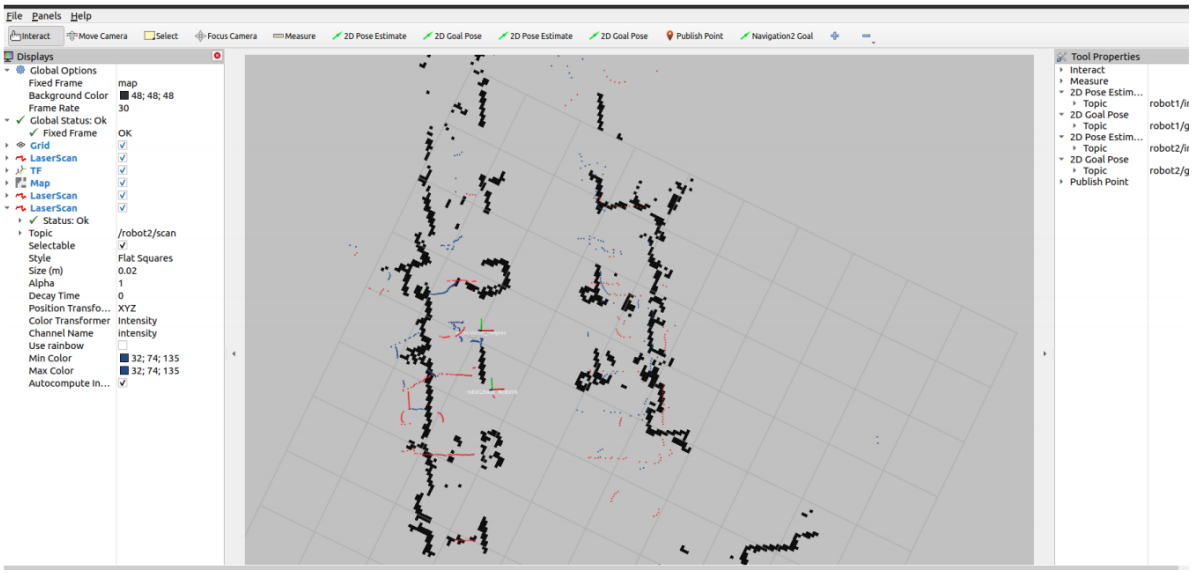
according to the actual car model,

```
#X3 models
#smallcar1
ros2 launch yahboomcar_multi X3_amcl_robot1_launch.py
#smallcar2
ros2 launch yahboomcar_multi X3_amcl_robot2_launch.py

#R2 model
#smallcar1
ros2 launch yahboomcar_multi R2_amcl_robot1_launch.py
#smallcar2
ros2 launch yahboomcar_multi R2_amcl_robot2_launch.py
```

When starting the two cars amcl terminal prints ACML cannot publish a pose or update the transform. Please set the initial pose..., then use the following figure to mark the position of the two cars,



After calibration, the color of robot1 radar is red and the color of robot2 radar is blue.



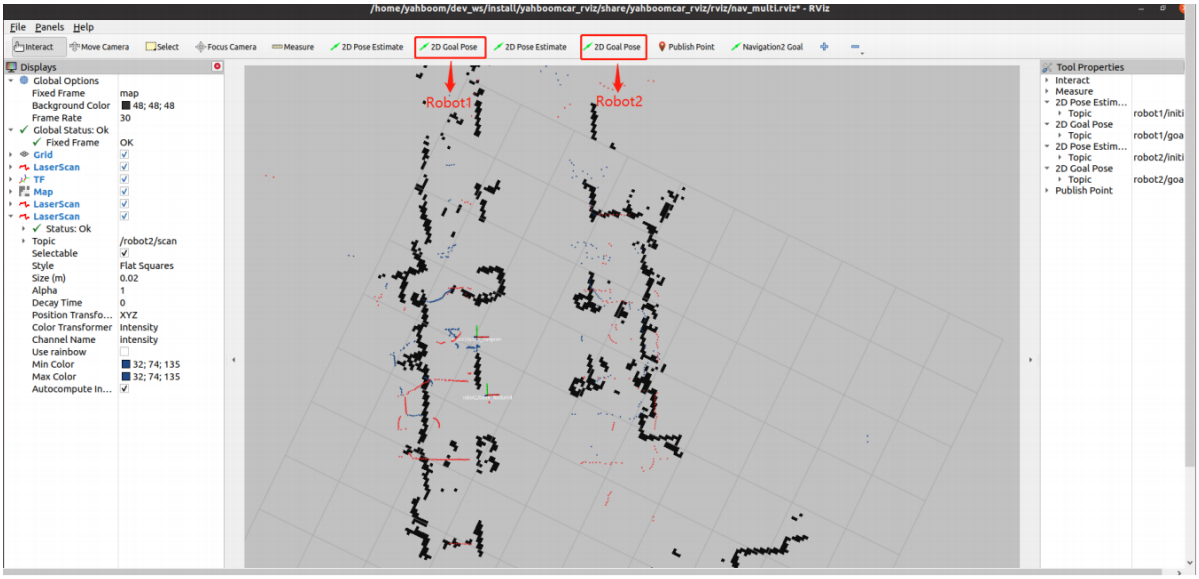
2.3.5. Start Nav_DWB navigation on two cars

Raspberry Pi 5 master enters the docker container of car 1 and car 2 respectively, **Orin master** does not need to enter docker,

according to the actual car model,

```
#X3 models
#smallcar1
ros2 launch yahboomcar_multi X3_nav_robot1_launch.py
#smallcar2
ros2 launch yahboomcar_multi X3_nav_robot2_launch.py

#R2 model
#smallcar1
ros2 launch yahboomcar_multi R2_nav_robot1_launch.py
#smallcar2
ros2 launch yahboomcar_multi R2_nav_robot2_launch.py
```

After the program is started, use the tools shown in the figure below to specify the target point.



After the target point is given, the cars will navigate to their respective destinations.

2.4. View node communication diagram

Virtual machine terminal input,

```
ros2 run rqt_graph rqt_graph
```