3 Multi-machine formation

3.1 Introduction

For the problem of how to configure multi-machine communication and synchronization time, please refer to the lesson [Multi-machine handle control] for details; if there is a network, the network system time can be directly synchronized without setting.

When using multi-machine handle control, it is first necessary to ensure that the robot is under the same local area network and configured with the same [ROS_MASTER_URI]; for multiple robots to control motion, there can only be one host. The example in this section sets the virtual machine as the host, and other robots as the slaves. There are several slaves. Of course, you can also set one robot as the master and others as the slaves.

According to different models, you only need to set the purchased model in [.bashrc], X1(ordinary four-wheel drive) X3(Mike wheel) X3plus(Mike wheel mechanical arm) R2(Ackerman differential) and so on. Section takes X3 as an example

```
#Raspberry Pi 5 master needs to enter docker first, please perform this step
#If running the script into docker fails, please refer to ROS/07, Docker tutorial
~/run_docker.sh
```

Open the [.bashrc] file

```
sudo vim .bashrc
```

Find the [ROBOT_TYPE] parameter and modify the corresponding model

```
export  ROBOT_TYPE=X3    # ROBOT_TYPE: X1 X3 X3plus R2 X7
```

3.2 Use

Take the virtual machine as the host and the three robots as slaves as an example; a map must be available before use. The three slaves are [robot1], [robot2], and [robot3], and [robot1] is set as the leader, and [robot2] and [robot3] are set as the follower. Make sure the field is large enough to avoid collisions when playing this feature. And no obstacle avoidance function.

3.2.1 Start the robot

virtual machine side

```
roscore
```

Start the command(robot1 side), for the convenience of operation, this section takes [mono + laser + yahboomcar] as an example.

```
roslaunch  yahboomcar_multi  laser_bringup_multi.launch  ns:=robot1          # laser + yahboomcar
roslaunch  yahboomcar_multi  laser_usb_bringup_multi.launch  ns:=robot1        # mono + laser + yahboomcar
roslaunch  yahboomcar_multi  laser_astrapro_bringup_multi.launch  ns:=robot1    # Astra + laser + yahboomcar
```

Start command(robot2 side), for the convenience of operation, this section takes [mono + laser + yahboomcar] as an example.

```
roslaunch  yahboomcar_multi  laser_bringup_multi.launch  ns:=robot2          # laser + yahboomcar
roslaunch  yahboomcar_multi  laser_usb_bringup_multi.launch  ns:=robot2        # mono + laser + yahboomcar
roslaunch  yahboomcar_multi  laser_astrapro_bringup_multi.launch  ns:=robot2    # Astra + laser + yahboomcar
```

More bots and so on.

3.2.2 Open multi-machine formation

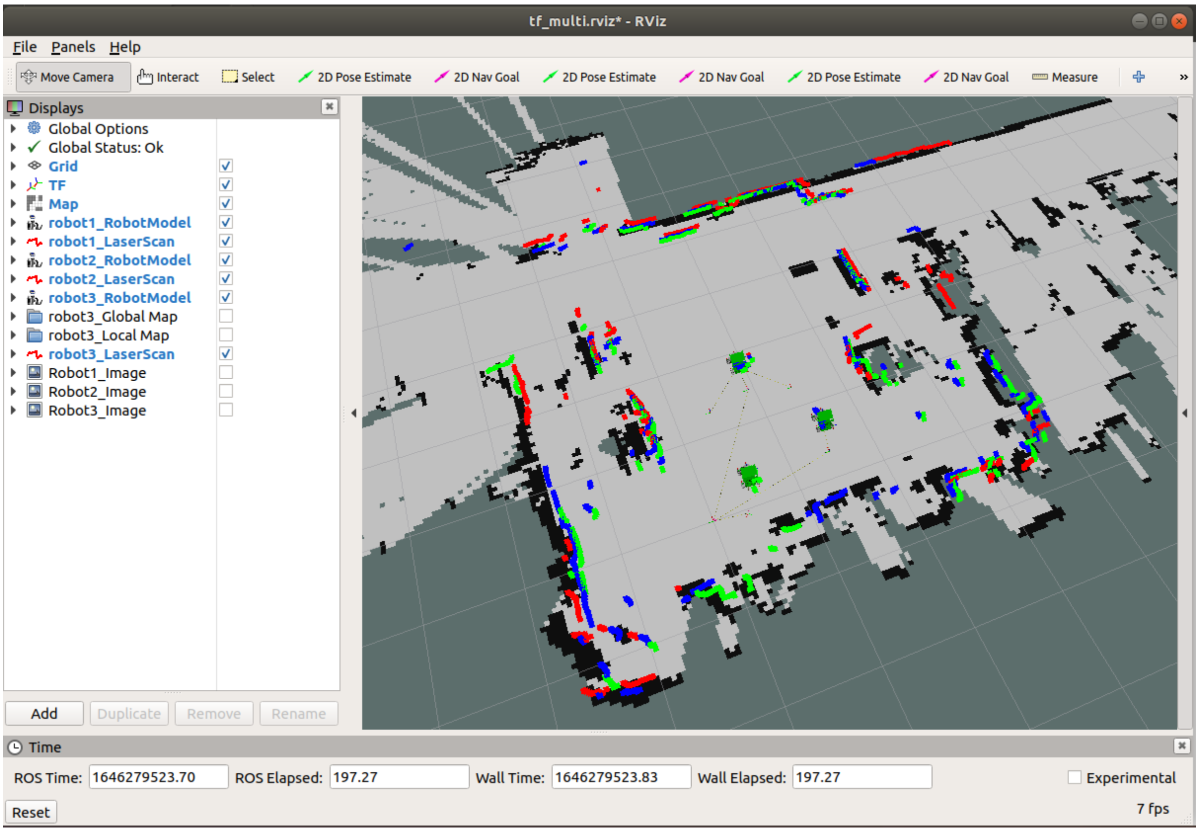
For the process of opening the handle control, please refer to the lesson [Multi-machine handle control].

virtual machine side

```
roslaunch yahboomcar_multi tf_queuebroad.launch use_rviz:=true map:=my_map
```

- [use_rviz] parameter: whether to open rviz.
- [map] Parameters: map name, the map to be loaded.

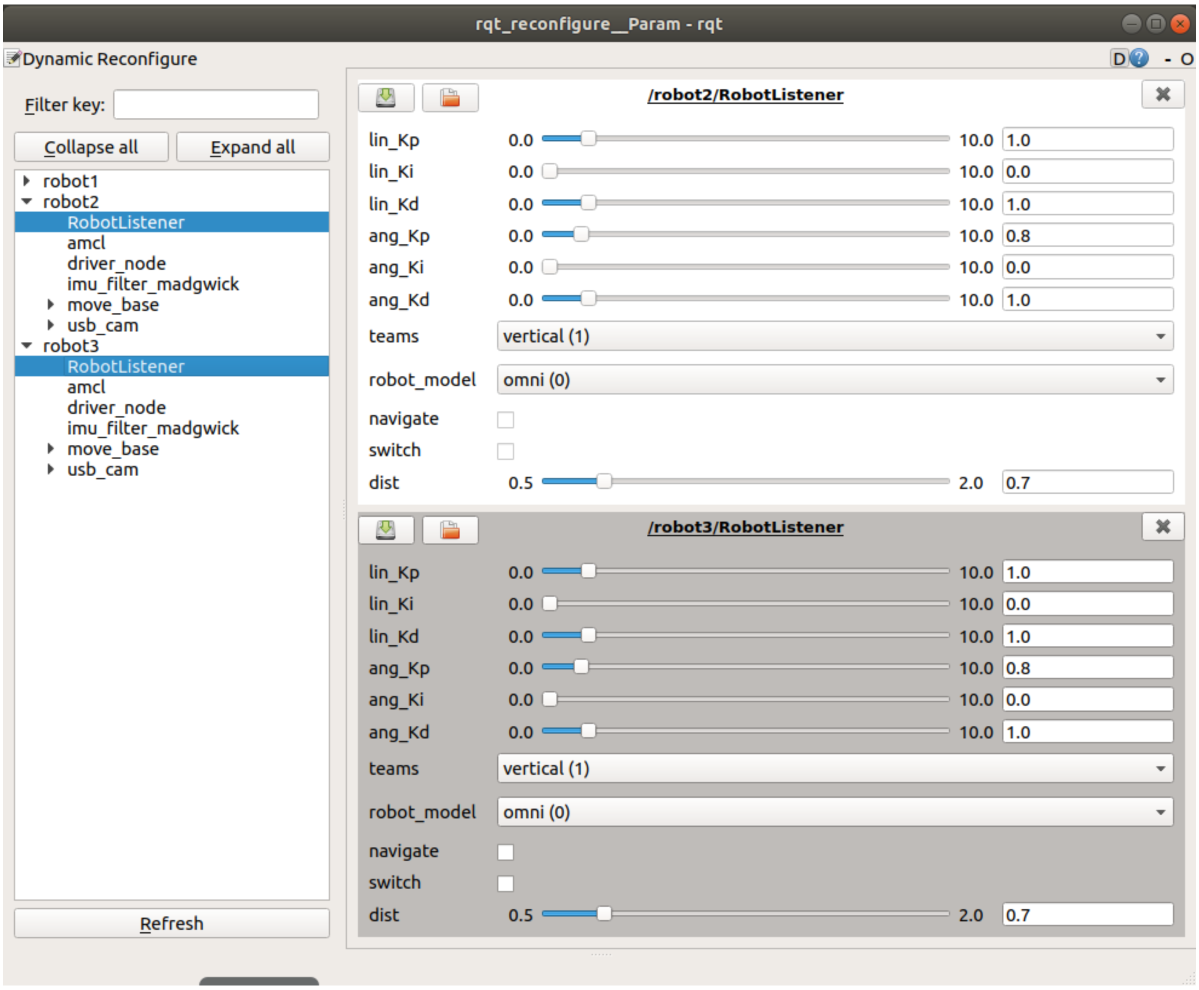
After startup, you need to initialize the pose setting of the robot. For the specific setting method, please refer to [Multi-machine Navigation This Lesson]. After setting, the following figure is shown.



3.2.3 Formation control

Open the dynamic parameter adjustment tool

```
rosrun rqt_reconfigure rqt_reconfigure
```



The tool can be set individually for each robot.

parameter parsing

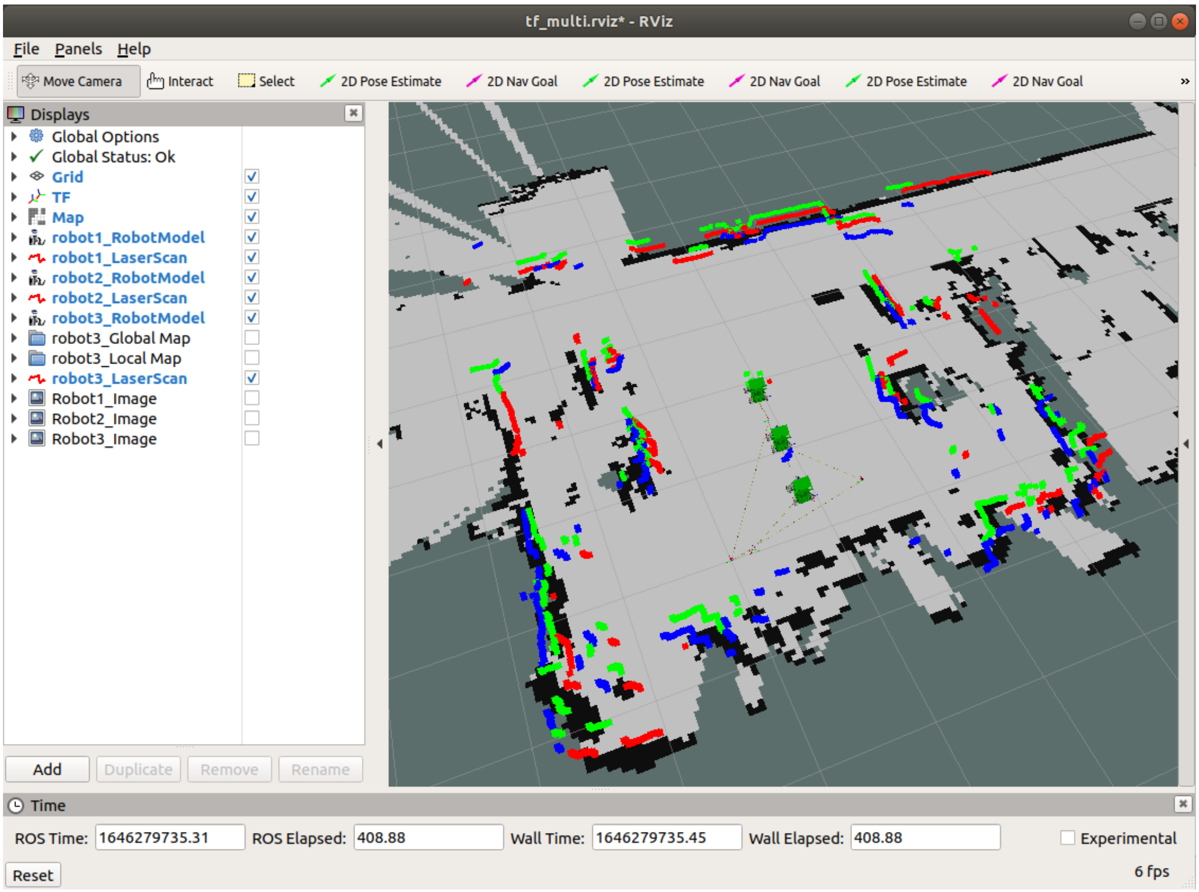
[lin_Kp], [lin_Ki], [lin_Kd]: PID debugging of trolley linear speed.

[ang_Kp], [ang_Ki], [ang_Kd]: PID debugging of car angular velocity.

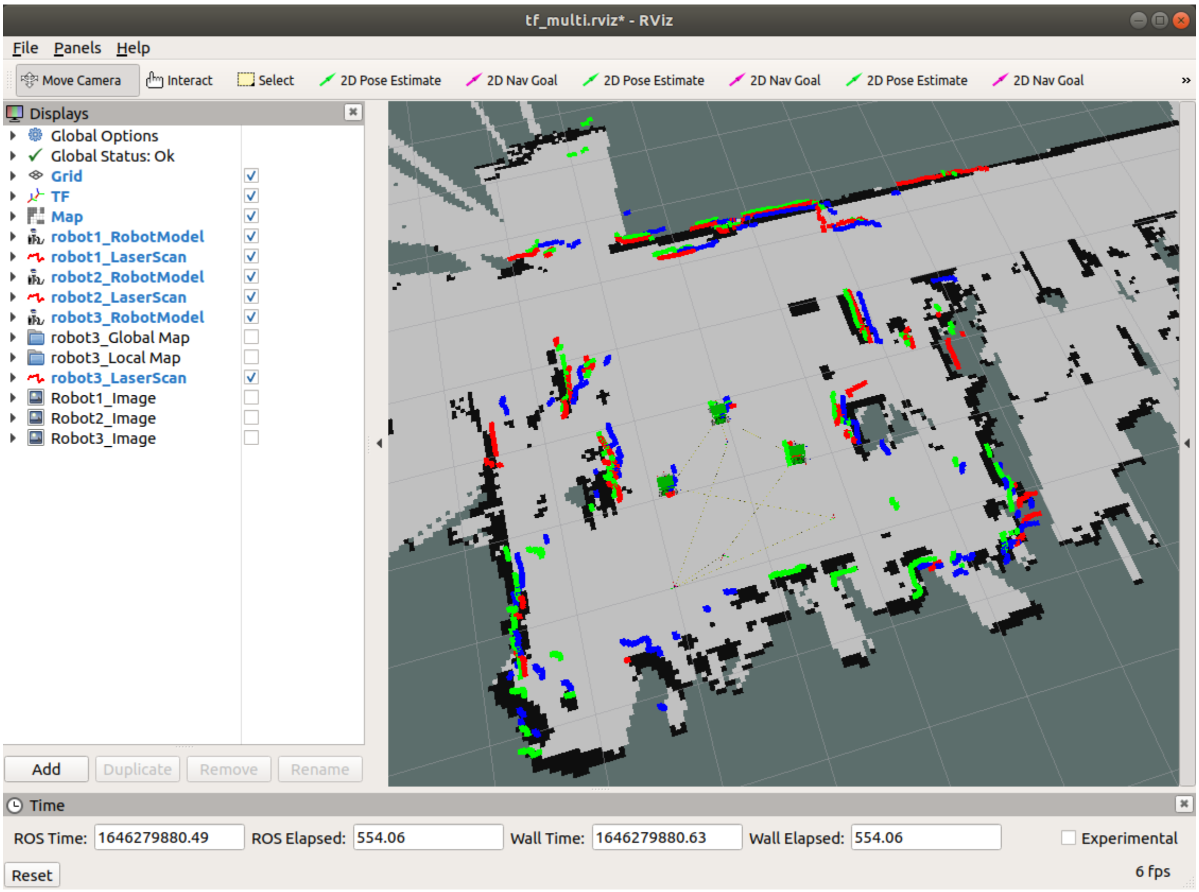
parameter	scope	Parse
[teams]	Default[vertical]	Formation: [convoy, vertical, horizontal]
[robot_model]	Default [omni]	Model: [omni, diff]
[navigate]	[False,True]	whether to run in navigation mode
[Switch]	[False,True]	Function switch [Start/Pause]
[dist]	[0.5, 2.0]	distance between queues

After the [Switch] function switch is turned on, the robot automatically maintains the formation set by the [teams] parameter, and the [teams] parameter only recognizes the last set value. As shown below

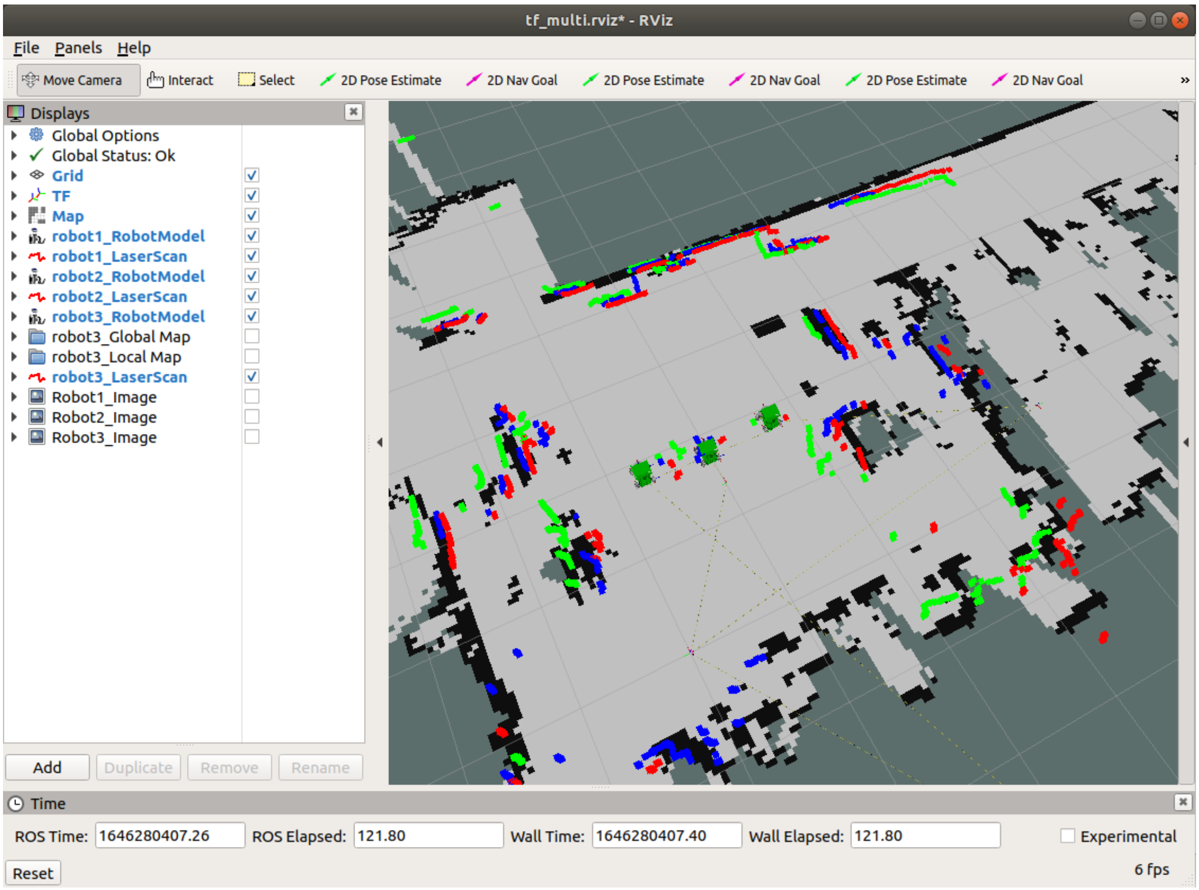
- a column(vertical)



- Left and right guards(convoy)



- Horizontal team(horizontal)



After the formation is set, when we control [robot1], other robots automatically maintain the formation without control.

3.3 launch file

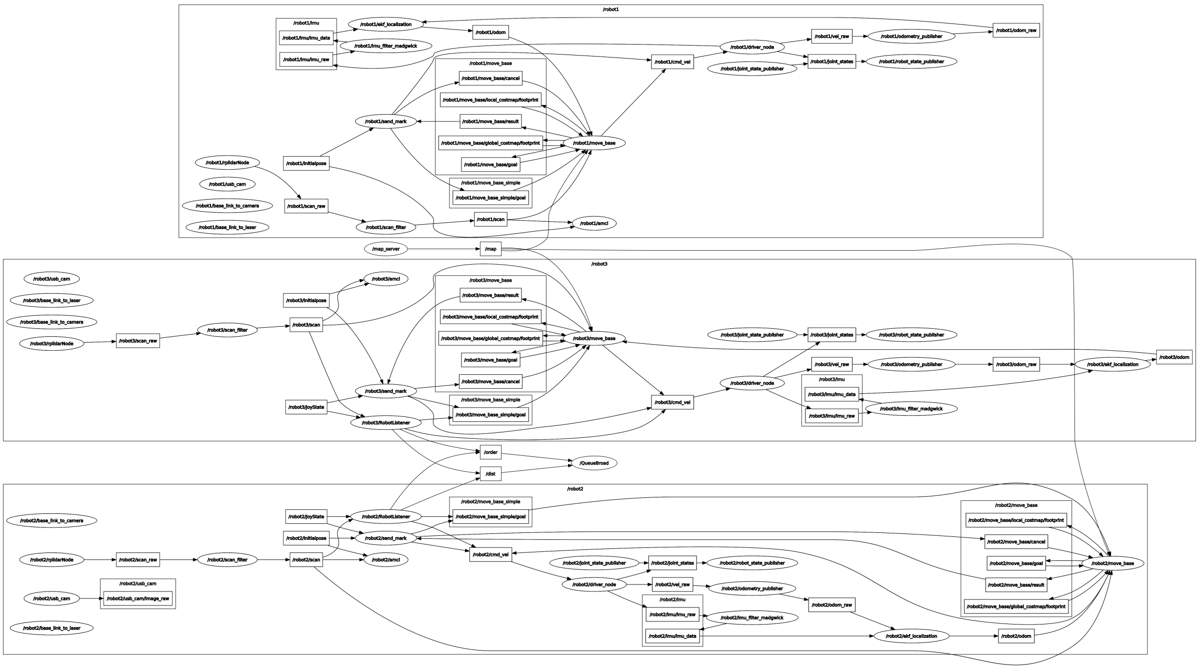
tf_queuebroad.launch

```
< launch >
    < arg  name = "first_robot1"  default = "robot1" />
    < arg  name = "second_robot2"  default = "robot2" />
    < arg  name = "third_robot3"  default = "robot3" />
    <!--  rviz  ||  Whether  to  open  rviz  -->
    < arg  name = "use_rviz"  default = "true" />
    <!--  Map name  ||  Map  name  -->
    < arg  name = "map"  default = "my_map" />
    <!--  Load map  ||  Load  map  -->
    < node  name = "map_server"  pkg = "map_server"  type = "map_server"  args =
"$(find yahboomcar_nav)/maps/$(arg map).yaml" />
    < node  pkg = "rviz"  type = "rviz"  name = "rviz"  required = "true"  args =
"-d $(find yahboomcar_multi)/rviz/tf_multi.rviz"  if = "$(arg use_rviz)" />
    <!--  ############################# first_robot1
############################# -->
    < node  pkg = "yahboomcar_multi"  type = "broad_queue.py"  name =
"QueueBroad"  output = "screen"  args = "$(arg first_robot1)" />
    < include  file = "$(find
yahboomcar_multi)/launch/library/move_base_multi.launch" >
        < arg  name = "ns"  value = "$(arg first_robot1)" />
    </ include >
    <!--  ############################# second_robot2
############################# -->
    < node  pkg = "yahboomcar_multi"  type = "listener.py"  name =
"RobotListener"  output = "screen"
          args = "$(arg second_robot2) point1"  ns = "$(arg second_robot2)/" >
        < rosparam  param = "linPIDparam" > [ 1.0, 0, 1.0 ] </ rosparam >
        < rosparam  param = "angPIDparam" > [ 0.8, 0, 1.0 ] </ rosparam >
    </ node >
    < include  file = "$(find
yahboomcar_multi)/launch/library/move_base_multi.launch" >
        < arg  name = "ns"  value = "$(arg second_robot2)" />
    </ include >
    <!--  ############################# third_robot3
############################# -->
    < node  pkg = "yahboomcar_multi"  type = "listener.py"  name =
"RobotListener"  output = "screen"
          args = "$(arg third_robot3) point2"  ns = "$(arg third_robot3)/" >
        < rosparam  param = "linPIDparam" > [ 1.0, 0, 1.0 ] </ rosparam >
        < rosparam  param = "angPIDparam" > [ 0.8, 0, 1.0 ] </ rosparam >
    </ node >
    < include  file = "$(find
yahboomcar_multi)/launch/library/move_base_multi.launch" >
        < arg  name = "ns"  value = "$(arg third_robot3)" />
    </ include >
</ launch >
```

3.4 frame analysis

Node view

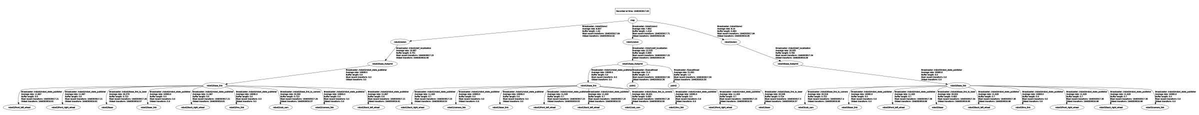
```
rqt_graph
```



[QueueBroad] The node subscribes to the queue instructions from each robot, and the node will only recognize the latest formation setting.

View tf tree

```
rosrun rqt_tf_tree rqt_tf_tree
```



It can be seen from the above figure that [robot1] will send out two coordinate systems [point1] and [point2], [robot2] and [robot3] monitor the relationship between itself and the coordinate system in real time, and make its own coordinate system coincide with the coordinate system. In the [tf_queuebroad.launch] file, we can see that [robot2] follows the [point1] coordinate system; [robot3] follows the [point2] coordinate system.