2.Lidar avoiding

Lidar obstacle avoidance gameplay introduction:

- Set lidar detection angle and response distance
- After the trolley is turned on, the trolley drives in a straight line when there are no obstacles
- Determine the orientation of obstacles appearing in the car(front left, front right, straight ahead)
- According to the orientation of the obstacle appearing in the car, make a reaction(turn left, turn right, turn left in a big circle, turn right in a big circle)

2.1 How to use

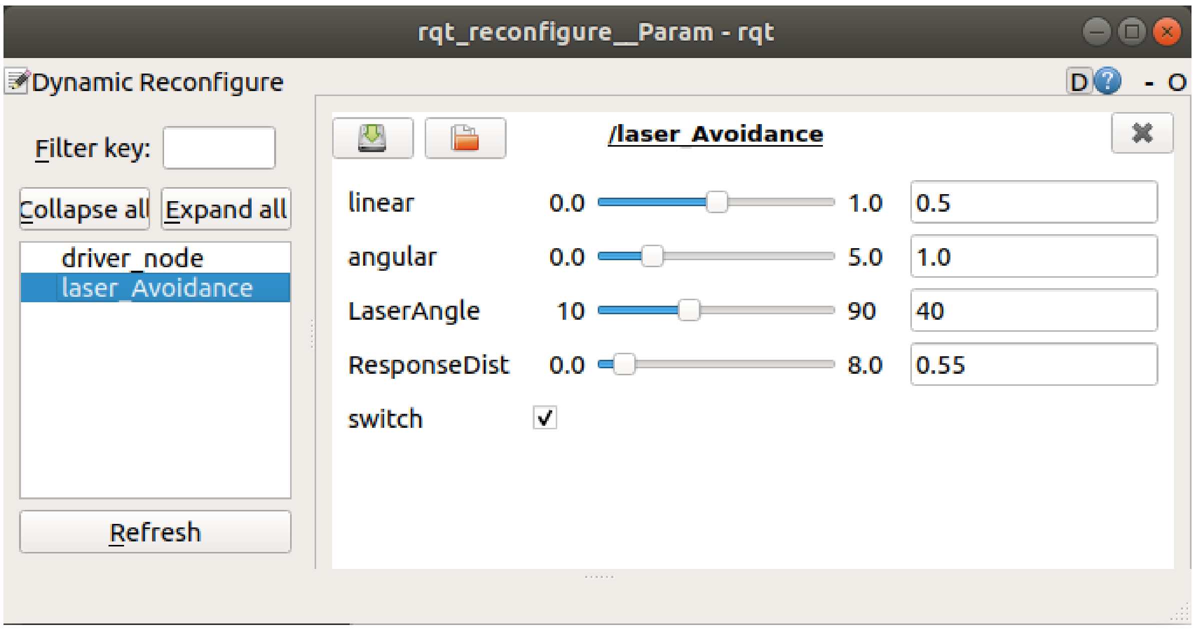
Note: The [R2] of the remote control handle has the function of [pause/open] for this gameplay. Due to the problem of the moving method, the case in this section does not support the Ackerman model. Different models, the parameter range will change, the principle is the same; take the X3 wheat wheel car as an example.

One-click start(robot side), after executing the command, the car starts to move.

```
#You need to enter docker first, perform this step more
#If running the script to enter docker fails, please refer to 07.Docker-orin/05、Enter the robot's docker container
~/run_docker.sh
roslaunch yahboomcar_laser laser_Avoidance.launch
```

Dynamic debugging parameters(VM side)

```
rosrun rqt_reconfigure rqt_reconfigure
```



linear	0.0	1.0	0.5
angular	0.0	5.0	1.0
LaserAngle	10	90	40
ResponseDist	0.0	8.0	0.55
switch	✓		

Parameter parsing:

parameter	scope	Parse
【linear】	【0.0, 1.0】	Trolley line speed
【angular】	【0.0, 5.0】	car angular velocity
【LaserAngle】	【10, 90】	Lidar detection angle(left and right side angle)
【ResponseDist】	【0.0, 8.0】	car response distance
【switch】	【False,True】	Car movement [start/pause]

In the box in front of [switch], click the value of [switch] to be True, and the car stops. [switch] The default is False, the car moves.

- parameter modification

When the parameters are adjusted to the optimal state, modify the corresponding parameters to the file, and there is no need to adjust them when using them again.

According to the optimal parameters of the [rqt_reconfigure] debugging tool, enter the [scripts] folder of the [yahboomcar_laser] function package, and modify the parameters corresponding to the [laser_Avoidance.py] file, as shown below

```
class  laserAvoid :
     def __init__(self):
        rospy.on_shutdown(self.cancel)
 …
        self.linear  =  0.5
        self.angular  =  1.0
        self.LaserAngle  =  40
        self.ResponseDist  =  0.55
```

[rqt_reconfigure] Modify the initial value of the debugging tool

```
gen.add("linear", double_t, 0, "linear in robot", 0.5, 0, 1.0)
gen.add("angular", double_t, 0, "angular in robot", 1.0, 0, 5.0)
gen.add("LaserAngle", int_t, 0, "LaserAngle", 40, 10, 90)
gen.add("ResponseDist", double_t, 0, "ResponseDist", 0.55, 0, 8)
gen.add("switch", bool_t, 0, "switch in rosbot", False)
```

Enter the [cfg] folder of the [yahboomcar_laser] function package, and modify the initial values of the parameters corresponding to the [laserAvoidancePID.cfg] file.

```
gen.add("linear", double_t, 0, "linear in robot", 0.5, 0, 1.0)
```

Analysis of the above one as an example

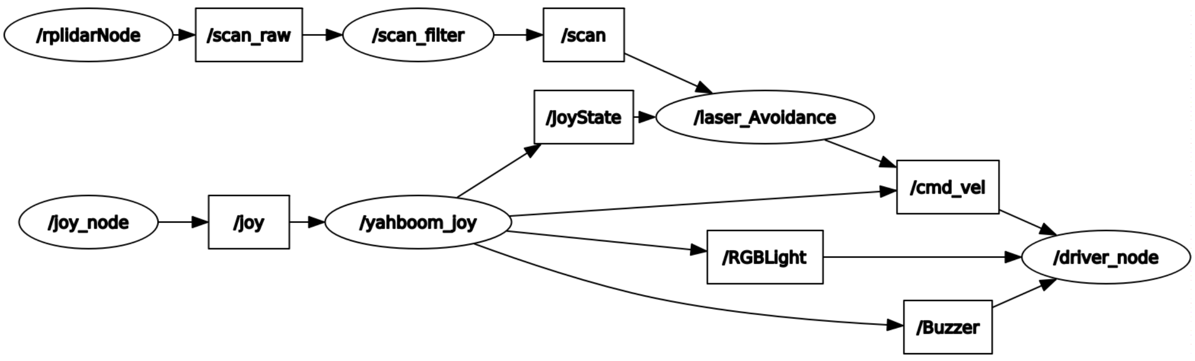
parameter	Parse	Corresponding parameters
name	the name of the parameter	"linear"
type	parameter data type	double_t
level	a bitmask passed to the callback	0
description	a description parameter	"linear in robot"
default	Initial value for node startup	0.5
min	parameter minimum	0
max	parameter maximum	1.0

Note: After modification, the update environment must be recompiled to be effective.

```
cd  ~/yahboomcar_ws
catkin_make
source  devel/setup.bash
```

Node view

```
rqt_graph
```



【laser_Aviodance】Node Analysis

- Subscribe to lidar data
- Subscribe to handle information
- Post trolley speed

2.2 source code analysis

launch file

- base.launch

```
< launch >
    <!-- start lidar node-->
    <!-- Activate the lidar node -->
    < include  file = "$(find rplidar_ros)/launch/rplidar.launch" />
    <!-- Start the car chassis drive node-->
    <!-- Start the car chassis drive node -->
    < include  file = "$(find yahboomcar_bringup)/launch/yahboomcar.launch" />
    <!-- handle control node -->
    <!-- Handle control node -->
    < include  file = "$(find yahboomcar_ctrl)/launch/yahboom_joy.launch" />
</ launch >
```

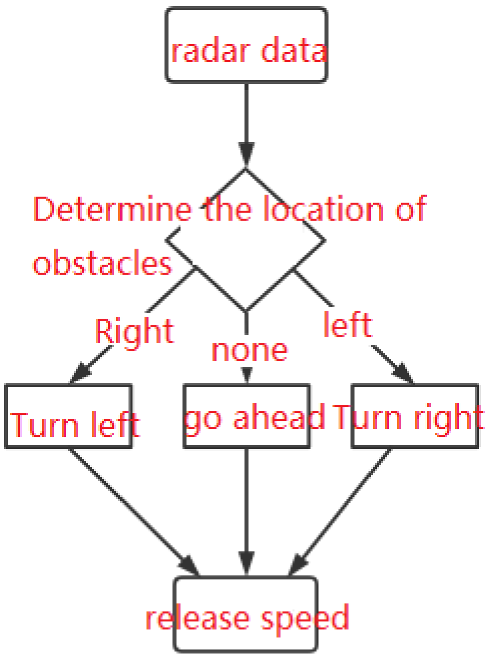
- laser_Avoidance.launch

```
< launch >
    <!-- start the base.launch file-->
    <!-- Launch the base.launch file -->
    < include  file = "$(find yahboomcar_laser)/launch/base.launch" />
    <!-- Start the lidar obstacle avoidance node-->
    <!-- Activate lidar obstacle avoidance node -->
    < node  name = 'laser_Avoidance'  pkg = "yahboomcar_laser"  type =
"laser_Avoidance.py"  required = "true"  output = "screen" />
</ launch >
```

Laser_Avoidance.py source code parameter analysis:

parameter	Defaults	judge
self.front_warning	Default is 0	When the value is greater than 10, it means there is an obstacle ahead.
self.Left_warning	Default is 0	When the value is greater than 10, it means that there is an obstacle in the front left.
self.Right_warning	Default is 0	When the value is greater than 10, it means that there is an obstacle in the front right.

Program design flow chart:



The car moves forward autonomously and avoids surrounding obstacles. For specific program design, see the source code of [laser_Avoidance.py].