4 Lidar follow

Lidar follow gameplay introduction:

- Set the lidar detection angle and distance.
- After the car is turned on, the car follows the target closest to the car and keeps a certain distance.
- When there is an obstacle in front of the car, the buzzer keeps sounding and stops moving back until there is no obstacle.
- The PID of the linear speed and angular speed of the car can be adjusted to make the car follow the best effect.

4.1 How to use

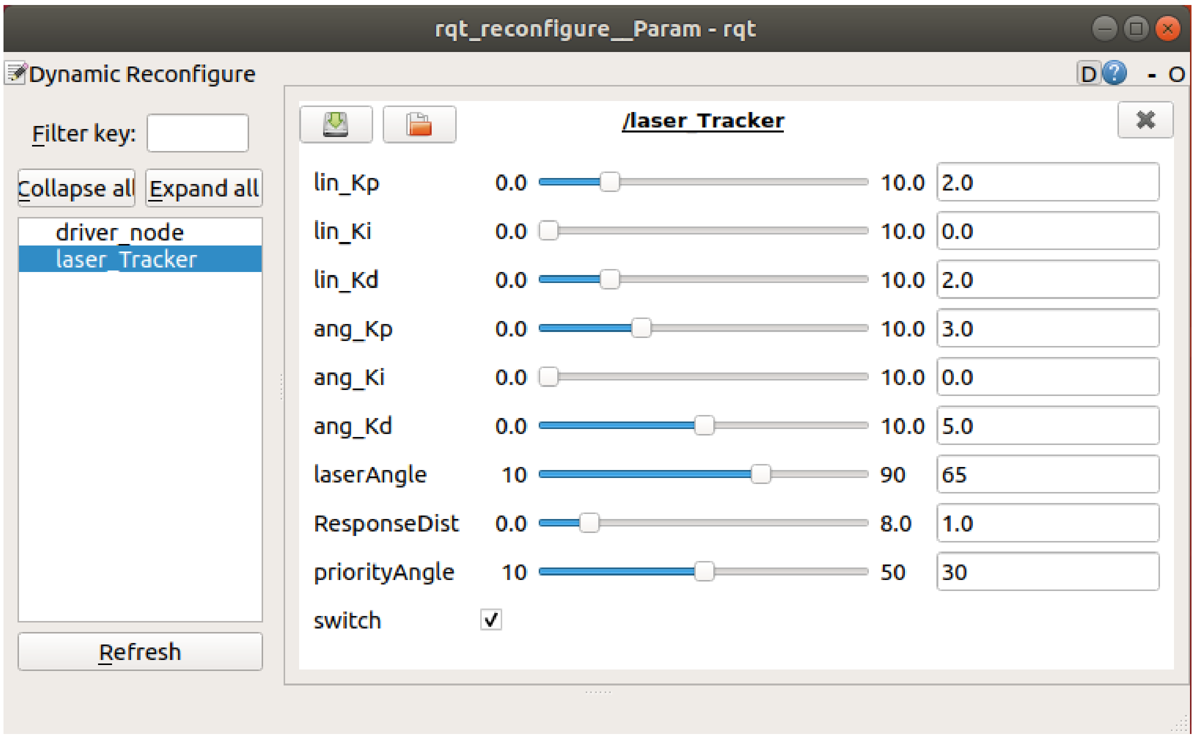
Note: [R2] of the remote controller has the function of [pause/on] for this gameplay. Different models will have slight differences, but the principle is the same; take the X3 Wheat Wheel as an example.

One key to start, 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_Tracker.launch
```

Dynamic debugging parameters

```
rosrun  rqt_reconfigure  rqt_reconfigure
```


rqt_reconfigure__Param - rqt

Dynamic Reconfigure

/laser_Tracker

lin_Kp	0.0	10.0	2.0
lin_Ki	0.0	10.0	0.0
lin_Kd	0.0	10.0	2.0
ang_Kp	0.0	10.0	3.0
ang_Ki	0.0	10.0	0.0
ang_Kd	0.0	10.0	5.0
laserAngle	10	90	65
ResponseDist	0.0	8.0	1.0
priorityAngle	10	50	30
switch	✓		

Parameter parsing:

parameter	scope	Parse
[laserAngle]	[10, 90]	Lidar detection angle(left and right side angle)
[ResponseDist]	[0.0, 8.0]	car following distance
[priorityAngle]	[10, 50]	The car gives priority to the following range(left and right angles)
[switch]	[False,True]	The car starts to stop

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

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

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.

The parameter [priorityAngle] cannot be larger than [laserAngle], otherwise it is meaningless.

- 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_Tracker.py] file, as shown below

```
class  laserTracker :
     def __init__(self):
         rospy.on_shutdown(self.cancel)
 …
         self.lin_pid  =  SinglePID(2.0, 0.0, 2.0)
         self.ang_pid = SinglePID(3.0, 0.0, 5.0)
         self.ResponseDist  =  rospy.get_param('~targetDist', 1.0)
         Server(laserTrackerPIDConfig, self.dynamic_reconfigure_callback)
         self.laserAngle  =  65
         self.priorityAngle  =  30   # 40
```

[rqt_reconfigure] Modify the initial value of the debugging tool

```
gen.add("lin_Kp", double_t, 0, "Kp in PID", 2.0, 0, 10)
gen.add("lin_Ki", double_t, 0, "Ki in PID", 0.0, 0, 10)
gen.add("lin_Kd", double_t, 0, "Kd in PID", 2.0, 0, 10)
gen.add("ang_Kp", double_t, 0, "Kp in PID", 3.0, 0, 10)
gen.add("ang_Ki", double_t, 0, "Ki in PID", 0.0, 0, 10)
gen.add("ang_Kd", double_t, 0, "Kd in PID", 5.0, 0, 10)
gen.add("laserAngle", int_t, 0, "laserAngle", 65, 10, 90)
gen.add("ResponseDist", double_t, 0, "ResponseDist", 1.0, 0, 8)
gen.add("priorityAngle", int_t, 0, "priorityAngle", 30, 10, 50)
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 [laserTrackerPID.cfg] file.

```
gen.add("lin_Kp", double_t, 0, "Kp in PID", 2.0, 0, 10)
```

Analysis of the above one as an example

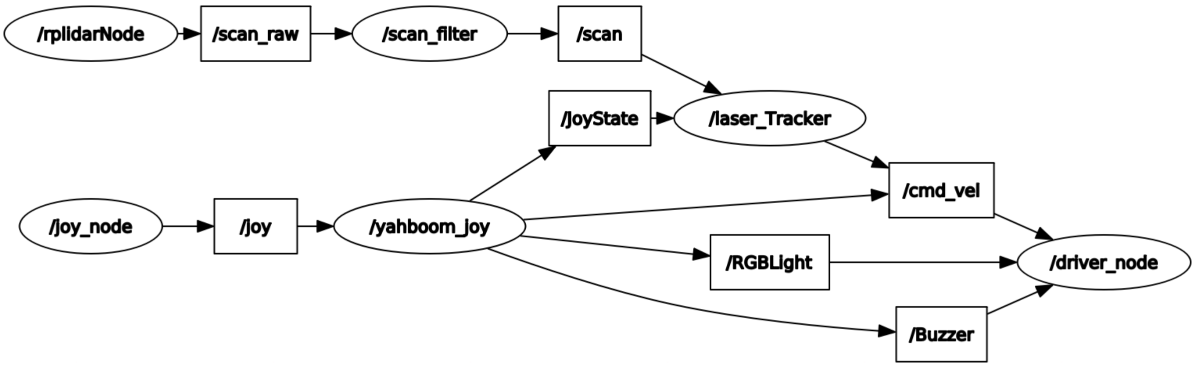
parameter	Parse	Corresponding parameters
name	the name of the parameter	"lin_Kp"
type	parameter data type	double_t
level	a bitmask passed to the callback	0
description	a description parameter	"Kp in PID"
default	Initial value for node startup	2.0
min	parameter minimum	0
max	parameter maximum	10

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
```



4.2 source code analysis

launch file

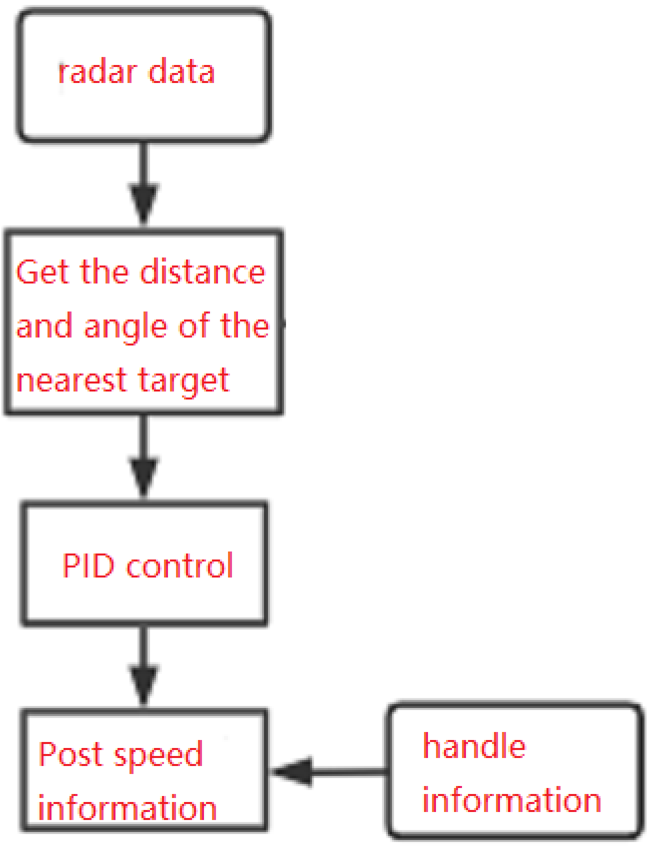
- base.launch

```
< 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 follower node-->
    <!-- Activate lidar follow node -->
    < node  name = 'laser_Tracker'  pkg = "yahboomcar_laser"  type =
"laser_Tracker.py"  required = "true"  output = "screen" />
</ launch >
```

laser_Tracker.py source code flow chart:



According to the position of the target, the trolley moves to face the target autonomously, and always maintains a certain distance.