5.Patrol game

5.1 How to use

Note: [R2] of the remote controller has the function of [pause/on] for this gameplay. This section takes the X3 Wheat Wheel as an example.

Different environments have different parameters; this function needs to be debugged patiently to get good results.

5.1.1 Start

One-click start(robot side)

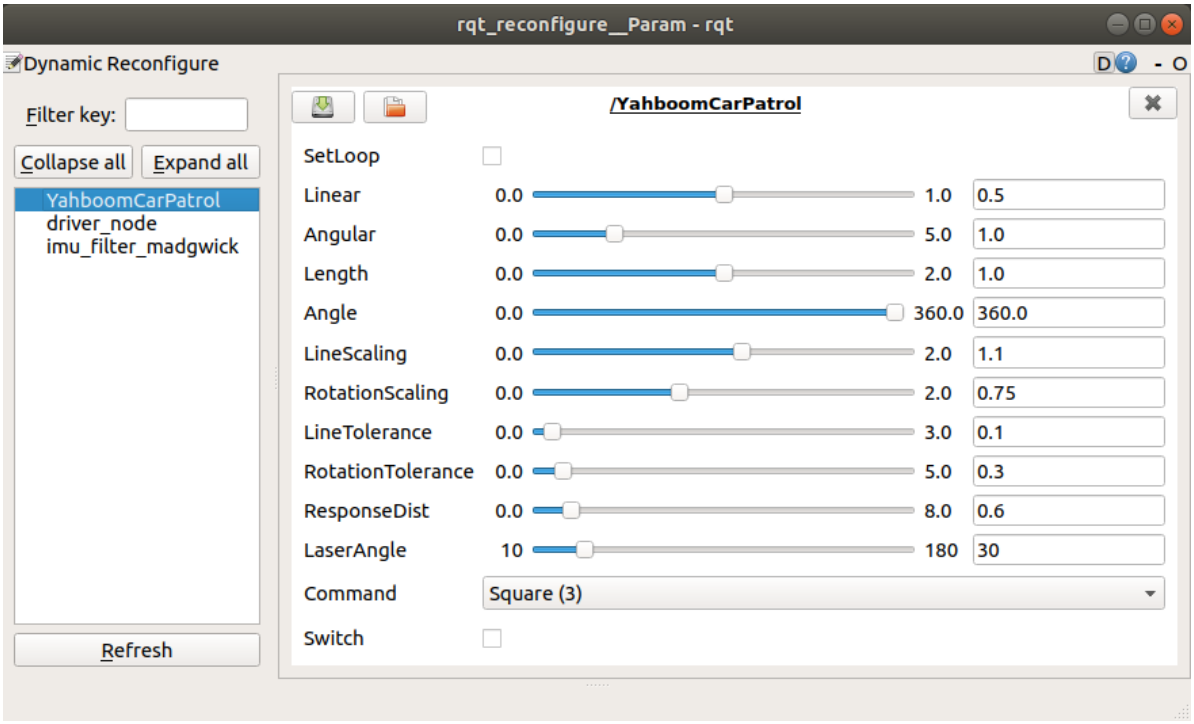
```
#You need to enter docker first, perform this step more
#If running the script to enter docker fails, please refer to 07.Docker-orin/05,
Enter the robot's docker container
~/run_docker.sh
roslaunch  yahboomcar_bringup  patrol.launch
```

Start successfully, print log

```
# The log of the successful start of the patrol function
Bring  up  rqt_reconfigure  to  control  the  Robot.
# The log of the successful startup of the underlying driver
First  IMU  message  received.
# Lidar startup successful log
RPLIDAR  S/N : 6 A97EDF9C7E29BD1A7E39EF2FA44431B
[  INFO ] [ 1631677752.206795121 ]:  Firmware  Ver :  1.29
[  INFO ] [ 1631677752.208026726 ]:  Hardware  Rev :  7
[  INFO ] [ 1631677752.210976099 ]:  RPLidar  health  status  :  0
[  INFO ] [ 1631677752.808115075 ]:  current  scan  mode :  Sensitivity,
max_distance :  12.0  m, Point  number :  7.9 K , angle_compensate :  2
```

At this point, after the above three parts are successfully started, the dynamic parameter debugging tool is started on the virtual machine side.

```
rosrun  rqt_reconfigure  rqt_reconfigure
```



Parameter parsing:

parameter	scope	Parse
【SetLoop】	【False,True】	Loop patrol, default False.
【Linear】	【0.0, 1.0】	The running speed of the trolley
【Angular】	【0.0, 5.0】	The running angular velocity of the trolley
【Length】	【0.0, 2.0】	The straight-line running distance of the trolley
【Angle】	【0.0, 360.0】	Rotation angle of the cart
【LineScaling】	【0.0, 2.0】	Straight line distance scaling, default 0.9
【RotationScaling】	【0.0, 2.0】	Rotation angle scaling, default 1.0
【LineTolerance】	【0.0, 1.0】	Allowable straight-line distance error
【RotationTolerance】	【0.0, 5.0】	Permissible rotation angle error
【ResponseDist】	【0.0, 8.0】	If there is an obstacle within the response distance, the car stops moving; Remove the obstacles, and the car continues to complete the patrol mission.
【LaserAngle】	【10, 180】	Lidar detection angle(left and right side angle)
【Command】	Default【Square】	Patrol mode: [LengthTest, AngleTest,Triangle, Square, Parallelogram, Circle]
【Switch】	【False,True】	Patrol function [Start/Pause]

When debugging parameters, mainly debug [RotationScaling], which can be debugged as [1.0, 1.1, 1.2, 1.3, 1.4], etc. according to the actual situation, and observe the effect, the circle is generally 0.9.

1) [LengthTest]: straight test command, adjust the parameters of [LineScaling] and [LineTolerance], so that the actual running distance of the car is close to the value [Length].

【LineScaling】The smaller the parameter, the larger the straight distance. 【LineTolerance】The smaller the parameter, the greater the front and rear vibration. After debugging many times, you can go to the best data, and the error always exists.

2) [AngleTest]: rotate the test command, adjust the parameters of [RotationScaling] and [RotationTolerance], so that the actual rotation distance of the car is close to the value [Angle].

【RotationScaling】The smaller the parameter, the larger the rotation angle.
【RotationTolerance】The smaller the parameter, the greater the left and right vibration. After debugging many times, you can go to the best data, and the error always exists.

After debugging [1)] and [2)], [LineScaling] and [LineTolerance], [RotationScaling] and [RotationTolerance] generally do not need to be adjusted.

5.1.2 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_bringup] function package, and modify the parameters corresponding to the [patrol.py] file, as shown below

```
class  YahboomCarPatrol():
     def __init__(self):
        self.SetLoop  =  False
        self.Linear  =  0.5
        self.Angular  =  1.0
        self.Length  =  1.0
        self.Angle  =  360.0
        self.LineScaling  =  1.1
        self.RotationScaling  =  0.75
        self.LineTolerance  =  0.1
        self.RotationTolerance  =  0.3
        self.ResponseDist  =  0.6
        self.LaserAngle  =  20
        self.Command  =  "finish"
        self.circle_adjust  =  rospy.get_param('~circle_adjust', 2.0)
```

[rqt_reconfigure] Modify the initial value of the debugging tool

```
gen.add("SetLoop", bool_t, 0, "SetLoop", False)
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("Length", double_t, 0, "Length in limit", 1.0, 0, 2.0)
gen.add("Angle", double_t, 0, "Angle in limit", 360.0, 0, 360.0)
gen.add("LineScaling", double_t, 0, "Line Scaling", 1.1, 0, 2.0)
gen.add("RotationScaling", double_t, 0, "Rotation Scaling", 0.75, 0, 2.0)
gen.add("LineTolerance", double_t, 0, "Line Tolerance", 0.1, 0, 3.0)
gen.add("RotationTolerance", double_t, 0, "Rotation Tolerance", 0.3, 0, 5.0)
gen.add("ResponseDist", double_t, 0, "ResponseDist in limit", 0.6, 0.0, 8.0)
gen.add("LaserAngle", int_t, 0, "LaserAngle in limit", 30, 10, 180)
```

Enter the [cfg] folder of the [yahboomcar_bringup] function package, and modify the initial values of the parameters corresponding to the [PatrolParam.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
```

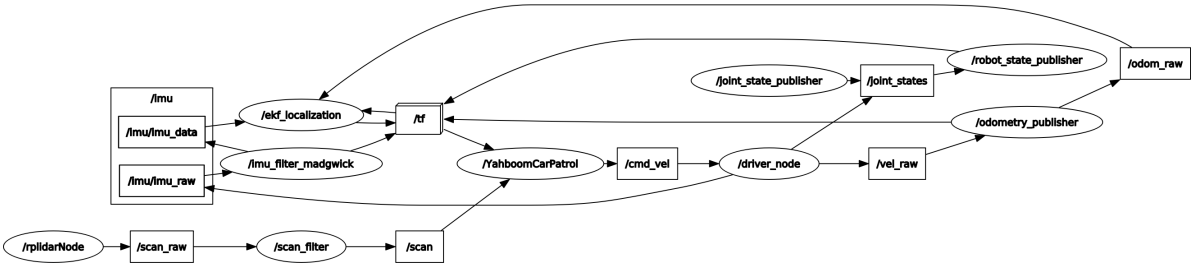
5.1.3 Patrol function

- After the startup is successful, select the trajectory to be executed in the [Command] [Triangle, Square, Parallelogram, Circle].
- The parameter [Length] can be adjusted according to requirements. For example, the default value is 1.0, the command is Square, and the trajectory of the car is a square with a side length of 1.0.
- When adjusting the parameter [Linear], it should be noted that the greater the speed, the greater the inertia and the smaller the precision.
- Parameter [LaserAngle]: For example: the angle is 30°, at this time, the system only analyzes 30° on the left and right sides of the car(the front is 0°)
- Parameter [Switch]: After setting, click the box behind [Switch] to start patrol. It is executed once by default. After the execution is completed, the check mark in the box will disappear automatically.

- If cyclic patrol is required, click the box behind [SetLoop] to continuously patrol, and the error will accumulate more and more.

Node view

```
rqt_graph
```



5.2 Source code analysis

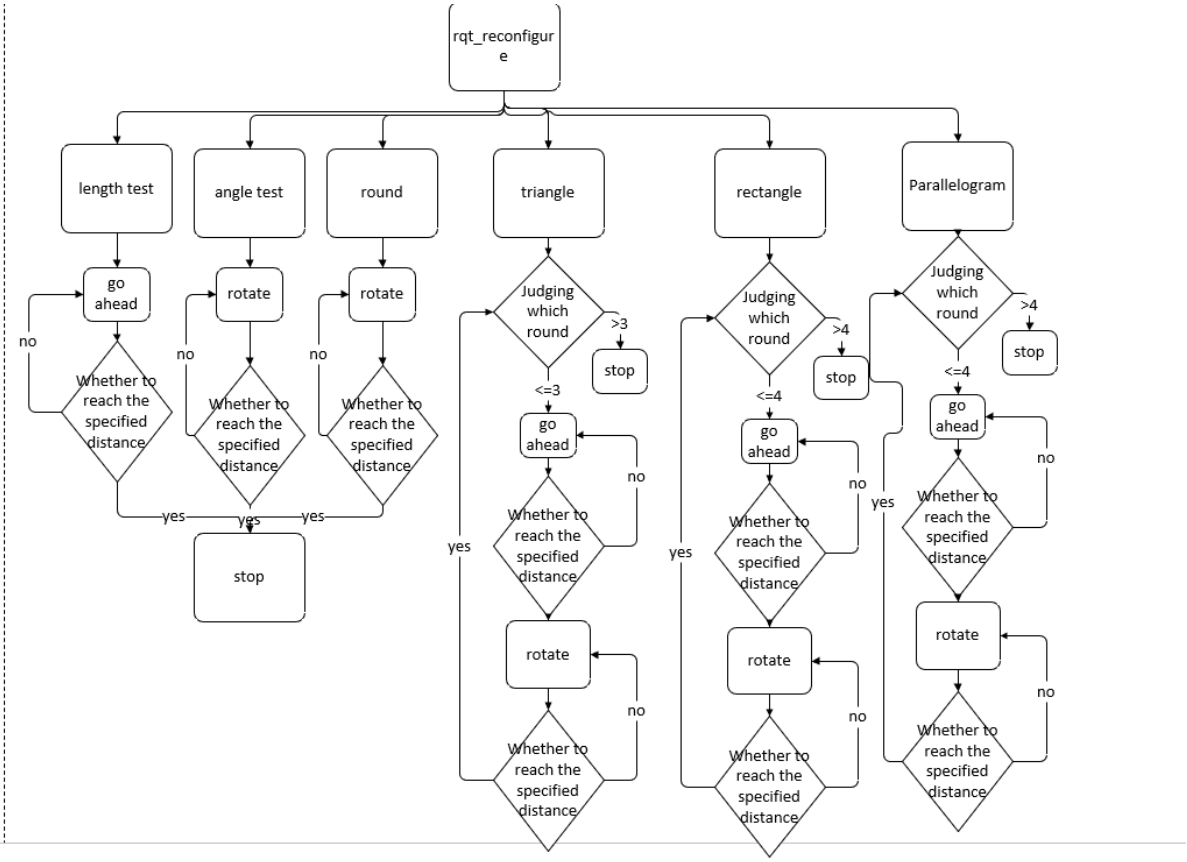
launch file

- patrol.launch

```
< launch >
    <!-- Start the underlying driver -->
    < include  file = "$(find yahboomcar_bringup)/launch/bringup.launch" />
    <!-- Start lidar Start lidar -->
    < include  file = "$(find rplidar_ros)/launch/rplidar.launch" />
    <!-- Start patrol node -->
    < node  pkg = "yahboomcar_bringup"  type = "patrol.py"  name =
"YahboomCarPatrol"  required = "true"  output = "screen" >
        < param  name = "circle_adjust"  type = "double"  value = "2.0" />
    </ node >
</ launch >
```

- circle_adjust parameter: It is a parameter for patrolling around a circle, which is a proportional factor to adjust the radius of the circle.

patrol.py source code flow chart:


rqt_reconfigur
e
length test
angle test
round
triangle
rectangle
Parallelogram
go
ahead
rotate
rotate
Judging
which
round
Judging
which
round
Judging
which
round
Whether to
reach the
specified
distance
Whether to
reach the
specified
distance
Whether to
reach the
specified
distance
Whether to
reach the
specified
distance
Whether to
reach the
specified
distance
Whether to
reach the
specified
distance
stop
stop
stop
stop
go
ahead
go
ahead
go
ahead
rotate
rotate
rotate
no
yes
>3
<=3
>4
<=4
>4
<=4