15. Automatic driving to clear obstacles

15.1. Using

This function has the function of line patrol, which can automatically clear the obstacles above the ground line, and continue to patrol the line after the cleaning is completed. At the same time, it has the function of radar obstacle avoidance.

Note: [R2] of the remote controller has the function of [pause/on] for this gameplay. The first time you use it, you first perform color calibration.

HSV (Hue, Saturation, Value) is a color space created by A. R. Smith in 1978 based on the intuitive characteristics of color, also known as the Hexcone Model.

The parameters of the color in this model are: Hue (H), Saturation (S), Lightness (V).

H: 0 — 180

S: 0 — 255

V: 0 — 255

Part of the red is classified as purple here:

	Black	Grey	White	Red		Orange	Yellow	Green	Cyan	Blue	Purple
H_min	0	0	0	0	156	11	26	35	78	100	125
H_max	180	180	180	10	180	25	34	77	99	124	155
S_min	0	0	0	43		43	43	43	43	43	43
S_max	255	43	30	255		255	255	255	255	255	255
V_min	0	46	221	46		46	46	46	46	46	46
V_max	46	220	255	255		255	255	255	255	255	255

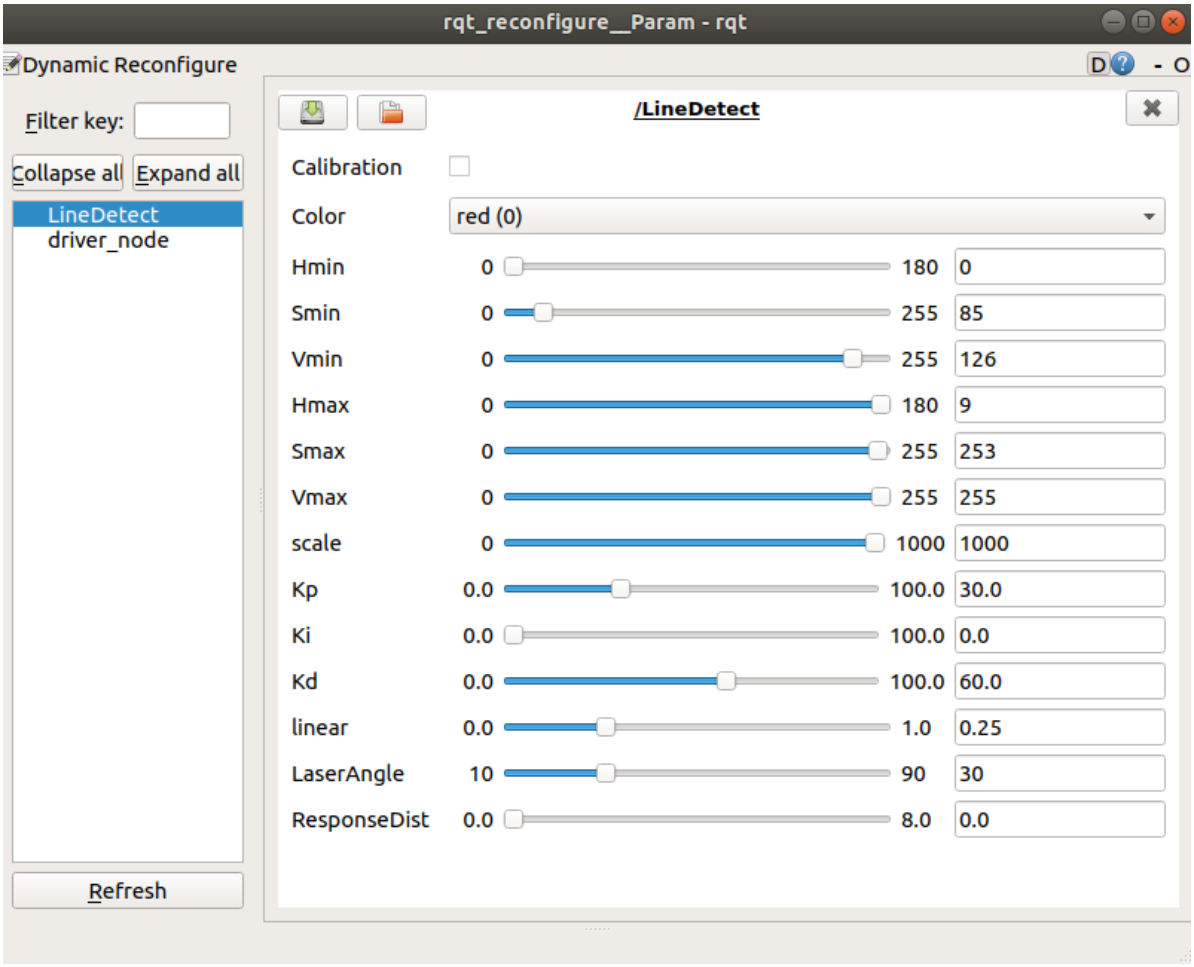
Start command (robot side)

```
roslaunch arm_autopilot arm_autopilot.launch
```

Start dynamic tuning parameters

```
rosrun rqt_reconfigure rqt_reconfigure
```

Dynamic adjustment parameter window



Select the [LineDetect] node, generally only need to adjust [Hmin], [Smin], [Vmin], [Hmax], these four parameters can be well identified. The sliding bar is always in the dragging state, and the data will not be transferred to the system, only after releasing it; you can also select a row, and then slide the mouse wheel.

Parameter parsing:

Parameter	Range	Parse
【laserAngle】	【10，90】	Lidar detection angle (left and right angles)
【ResponseDist】	【0.0，8.0】	Obstacle detection distance
【Calibration】	【False，True】	Color calibration switch
【linear】	【0.0，1.0】	The linear speed of the robot forward.

【Kp】、【Ki】、【Kd】：Car speed PID debugging.

【scale】：PID control scaling factor.

Note: If the linear speed [linear] is too fast, the robot will not be able to see obstacles during the movement. The flatter the ground, the better.

- 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 [arm_autopilot] function package, and modify the parameters corresponding to the [autopilot_main.py] file, as shown below

```
class LineDetect:
    def __init__(self):
        ... ...
        self.scale = 1000.0
        self.FollowLinePID = (30.0, 0.0, 60.0)
        self.linear = 0.25  # Too fast speed will cause the robot to not see
obstacles during the movement
```

[rqt_reconfigure] Initial value modification of debugging tools

```
gen.add("Calibration", bool_t, 0, "color calibration", False)
command_enum = gen.enum([gen.const("red", int_t, 0, "first data"),
                         gen.const("green", int_t, 1, "two data"),
                         gen.const("blue", int_t, 2, "three datat"),
                         gen.const("yellow", int_t, 3, "four data")], "An enum
to set size")
gen.add("Color", int_t, 0, "A size parameter which is edited via an enum", 0, 0,
3, edit_method=command_enum)
gen.add("Hmin", int_t, 0, "Hmin in HSV", 0, 0, 180)
gen.add("Smin", int_t, 0, "Smin in HSV", 85, 0, 255)
gen.add("Vmin", int_t, 0, "Vmin in HSV", 126, 0, 255)
gen.add("Hmax", int_t, 0, "Hmax in HSV", 9, 0, 180)
gen.add("Smax", int_t, 0, "Smax in HSV", 253, 0, 255)
gen.add("Vmax", int_t, 0, "Vmax in HSV", 255, 0, 255)
gen.add("scale", int_t, 0, "scale", 1000, 0, 1000)
gen.add("Kp", double_t, 0, "Kp in PID", 30.0, 0, 100)
gen.add("Ki", double_t, 0, "Ki in PID", 0.0, 0, 100)
gen.add("Kd", double_t, 0, "Kd in PID", 60.0, 0, 100)
gen.add("linear", double_t, 0, "linear", 0.25, 0, 1.0)
gen.add("LaserAngle", int_t, 0, "LaserAngle", 30, 10, 90)
gen.add("ResponseDist", double_t, 0, "ResponseDist", 0.55, 0, 8)
```

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

```
gen.add("Kp", double_t, 0, "Kp in PID", 30.0, 0, 100)
```

Analysis of the above one as an example

Parameter	Parse	Corresponding parameters
name	Parameter name	"Kp"
type	Parameter data type	double_t
level	A bitmask to pass to the callback	
description	A description parameter	"Kp in PID"
default	Initial value for node startup	30.0
min	Parameter minimum	0
max	Parameter maximum	100

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

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

15.1.1.The first step

Color scale

- Click the box on the right side of [Calibration] to enter the calibration mode; you can also select the image window and click the [C] key on the keyboard.
- Select the color [red, green, blue, yellow] in [Color] and place the color block in the lower half of the camera's field of view.
- Adjust the [Hmin, Smin, Vmin, Hmax, Smax, Vmax] parameters in the dynamic control parameter window.
- After the parameter adjustment is completed, the system will automatically write it into the [HSV.yaml] file and save it, and it will be read next time the system is started.

In addition：

In the calibration mode, you can also use the mouse to select the color block for color calibration, and then fine-tune the [Hmin, Smin, Vmin, Hmax, Smax, Vmax] parameters in the dynamic control parameter window.

15.1.2.The second step

Notice:

1）、Here it is best to adjust the runway in the middle of the camera screen and the direction of the squares to be parallel to the runway, so that it is easier to clamp the squares

2）、At the turning point, the effect of clamping the claw and the block may not be very good (maybe because the angle of rotation is a bit large, or it may be too fast to see the obstacle), so at this time, you need to reduce the speed and try again.

Method 1: Continuously click the handle [R1] key within 0.5 seconds to start line inspection.

Method 2: Autopilot (select the image window)

- Make sure that the box to the right of [Calibration] is not selected, as shown above.
- Click [Spacebar] to run the line patrol.
- Click【f 】to switch the color of the patrol line.
- Press【i】to enter recognition mode.
- Click【c】to switch calibration mode.
- Press [q] to exit the program.

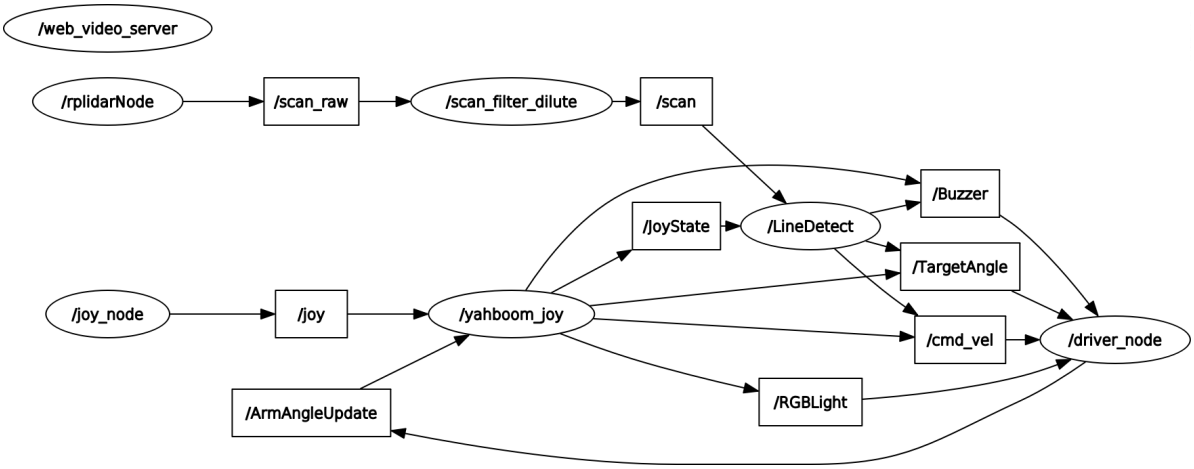
15.2.Launch file

```
<launch>
    <arg name="img_flip" default="false"/>
    <include file="$(find ydlidar_ros_driver)/launch/TG.launch"/>
    <include file="$(find yahboomcar_ctrl)/launch/yahboom_joy.launch"/>
    <include file="$(find yahboomcar_bringup)/launch/yahboomcar.launch"/>
    <!-- web_video_server -->
    <node pkg="web_video_server" type="web_video_server" name="web_video_server"
output="screen"/>
    <node name="LineDetect" pkg="arm_autopilot" type="autopilot_main.py"
output="screen" required="true">
        <param name="img_flip" type="bool" value="$(arg img_flip)"/>
    </node>
</launch>
```

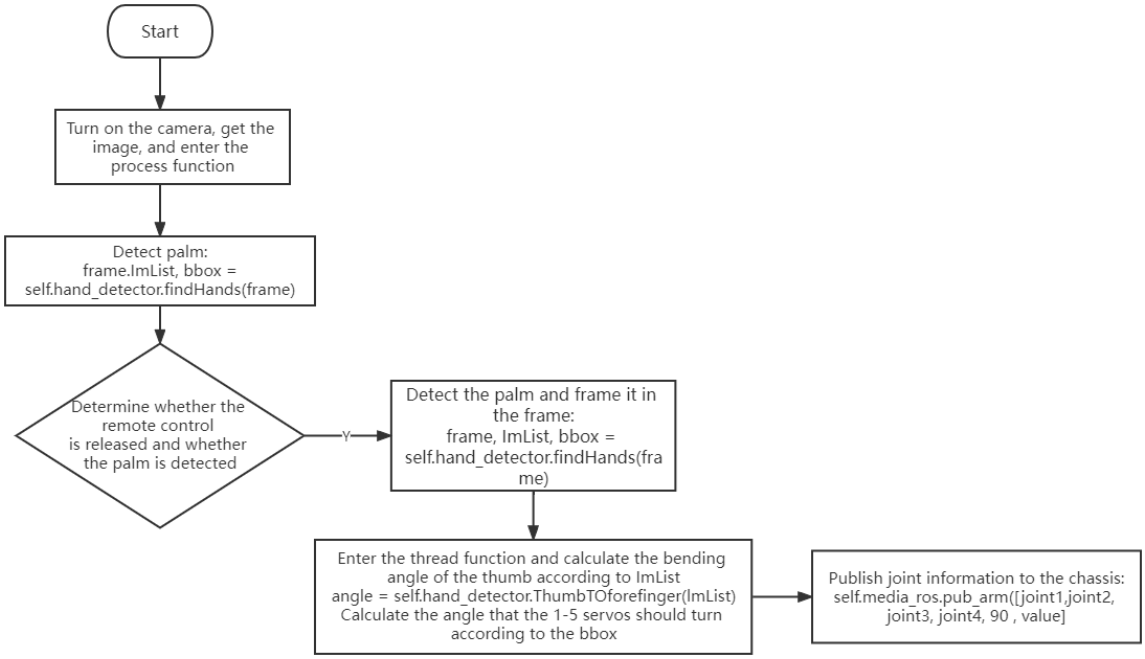
Generally, do not set the [img_flip] parameter. For the image flip parameter, use the default [false].

15.3. Node graph

```
rqt_graph
```



15.4. Flowchart


Start
Turn on the camera, get the image, and enter the process function
Detect palm:
frame.ImList, bbox = self.hand_detector.findHands(frame)
Determine whether the remote control is released and whether the palm is detected
Y
Detect the palm and frame it in the frame:
frame, ImList, bbox = self.hand_detector.findHands(frame)
Enter the thread function and calculate the bending angle of the thumb according to ImList
angle = self.hand_detector.ThumbTOforefinger(lmList)
Calculate the angle that the 1-5 servos should turn according to the bbox
Publish joint information to the chassis:
self.media_ros.pub_arm([joint1,joint2, joint3, joint4, 90 , value]