3. Calibrate Autopilot data

Note: This course is only applicable to the R2L car and Yahboom's autopilot map.

If you use other models or other maps, you need to develop and debug the code by yourself, and the code provided in this chapter cannot be used directly.

Before turning on automatic driving, we need to calibrate some data, which includes the following two parts:

- Running outer circle: **Yellow Line HSV** and **Turning Coefficient**
- Motion decision-making: **sign recognition area**, **turning distance coefficient**, **turning time**, **side parking warehousing data package.**

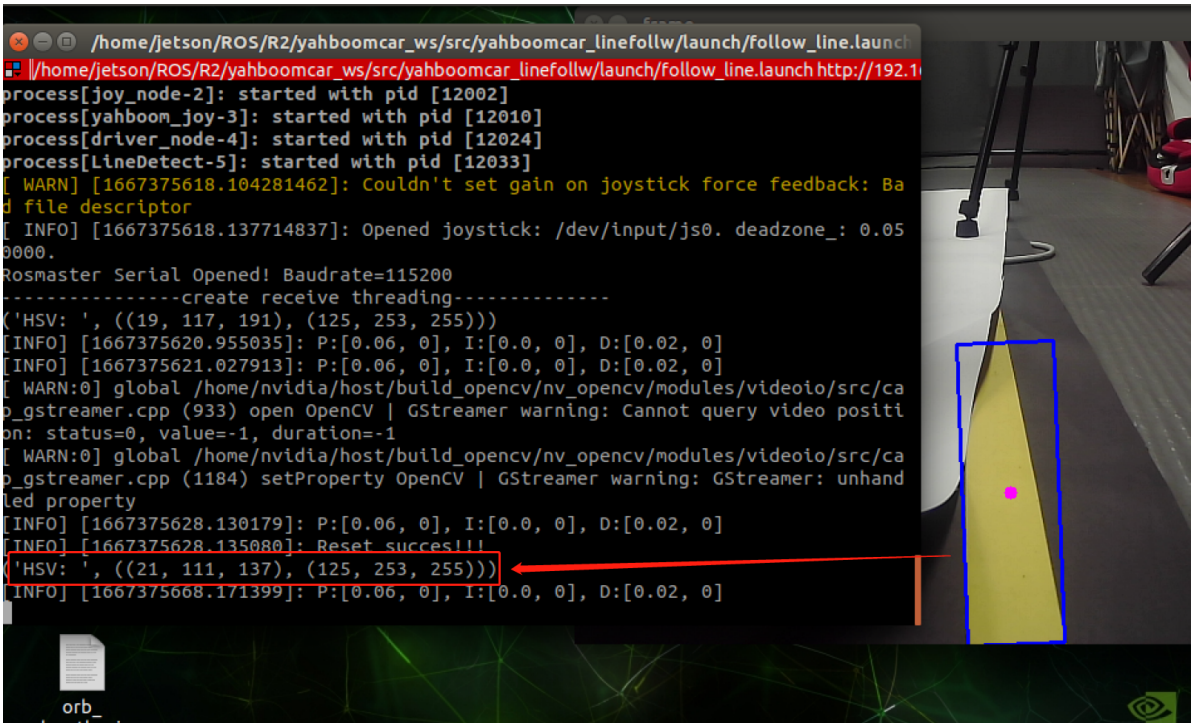
3.1. Data of running outer circle

3.1.1.Calibration yellow line hsv

(1) Start

```
roslaunch yahboomcar_linefollw follow_line.launch
```

After the program is started, we press the R key to enter the color picking mode, select a part of the area on the yellow line with the mouse, and after releasing the mouse, the HSV value of this part of the area will be printed out on the terminal.



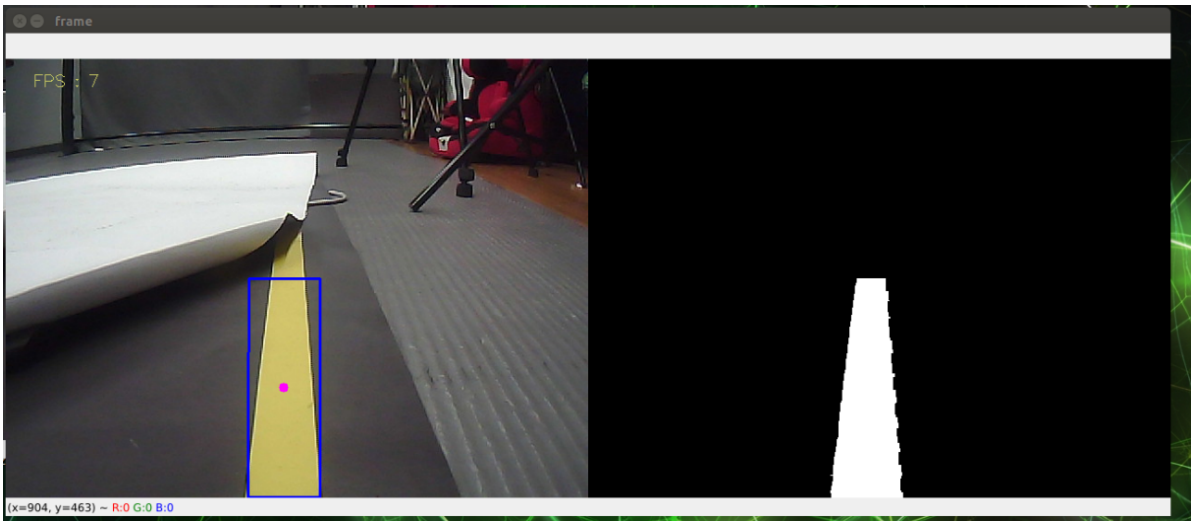
Copy the value here to the ~/yahboomcar_ws/src/yahboomcar_autodrive/scripts/LineFollowHSV.text file, save and exit.

(2)Verify

Run the program to verify whether the calibrated data is accurate, terminal input,

```
roscore
python yahboomcar_ws/src/yahboomcar_autodrive/scripts/line_detect.py
```

The following screen appears to show that the calibration data is more accurateThe following screen appears to show that the calibration data is more accurate



Because the camera is sensitive to light, the lighting environment before and after calibration, including the environment of automatic driving behind the vehicle, should be as consistent as possible.

3.1.2. Turning coefficient

After completing the calibration of the yellow line HSV value, run the program

```
roslaunch yahboomcar_bringup bringup.launch
python yahboomcar_ws/src/yahboomcar_autodrive/scripts/line_detect.py
rosrun yahboomcar_autodrive test.py
```

After the program runs, the car will start to move along the yellow line, and the **R2 of the handle** can start/pause the car movement. If the left and right swings of the car are too large, lift the camera net up a little. After the camera height of the car is fixed, let the car run to the turning point, let the car turn automatically, observe the turning situation, and modify the turning coefficient according to the actual situation. The file location of the turning coefficient file is,

```
~/yahboomcar_ws/src/yahboomcar_autodrive/scripts/line_detect.py
```

Find the following section,

```
    def execute(self, point_x, color_radius):
        if color_radius == 0: print("stop")
        else:
            twist = Twist()
            b = Bool()
            if self.flag == 1:
                center_x = 320-point_x
                #print(point_x)
                if point_x<320  :
                    point_x = point_x
```

```
                    [z_Pid, _] = self.PID_controller.update([(point_x -
30)*1.8/32,0])
                    if self.img_flip == True: self.go_angular_z = -z_Pid
                    else: self.go_angular_z = +z_Pid
                elif point_x>320 or point_x==320:
                    [z_Pid, _] = -self.PID_controller.update([(610-
point_x)*1.8/32,
```

- Among them, **[z_Pid, _] = self.PID_controller.update([(point_x - 30)*1.8/32,0])**, indicates the value of the angular velocity calculated according to the offset of the coordinates,The **1.8 and 32** here are the turning coefficients, and the one below is the same. Modify these two parameters according to the actual situation, and give priority to debugging 1.8.
- Press the yellow line of the inner ring, it means that the calculated angular velocity is too large, and the coefficient should be reduced;
- If you press the yellow line on the outer ring or fail to return to the right position in time after turning, it means that the calculated angular velocity is too small, and the coefficient should be increased.

After the above two calibrations are completed, it is necessary to fix the **camera with screws to ensure that it will not loosen, otherwise, the calibration data will be inaccurate**.

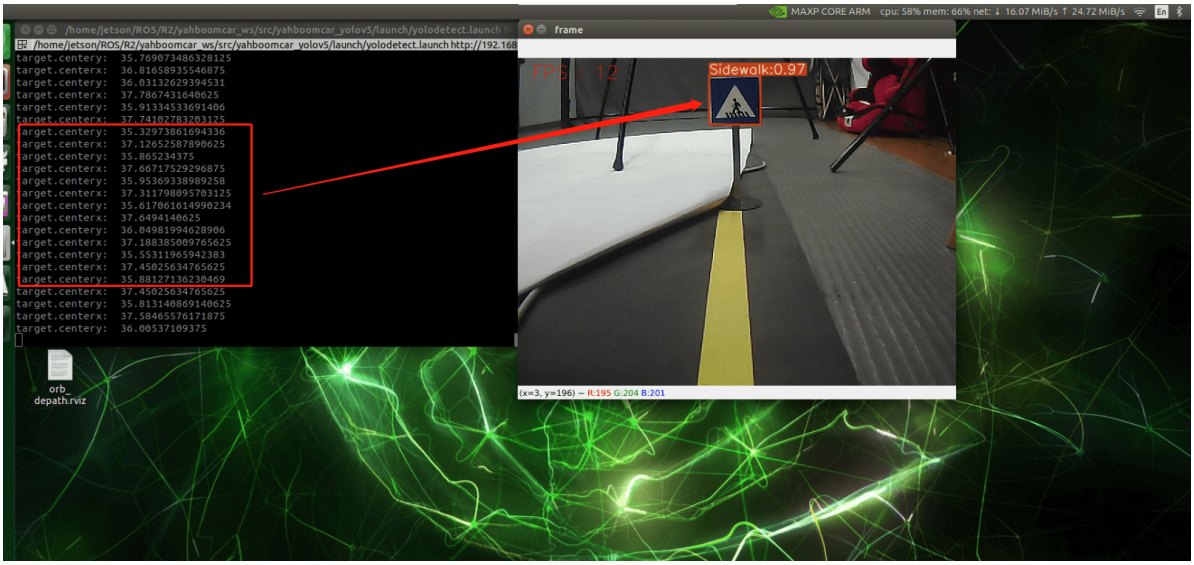
3.2. Movement decision-making

3.2.1. Sign recognition area

Put the car in the middle of the road, and then place the traffic sign at the distance that needs to be recognized. For example, I want the car to be recognized when it is 20cm away from it, so we put the sign at 20cm away from the car, and then turn on program,

```
roslaunch yahboomcar_yolov5 yolodetect.launch
```

As shown in the picture below, I hope to recognize the sign of the sidewalk here, so only this sign will appear in the picture,



The red box part represents the center coordinate xy of the recognition box, **target.centerx** and **target.centery**. Get the center coordinates to delineate the area. For example, the coordinates here are (37,35). I hope that when this sign appears, stop or slow down when the target.centery is greater than 35, then you can Modify in the file, the file path is,

```
~/yahboomcar_ws/src/yahboomcar_autodrive/scripts/test.py
```

Find the following section,

```
 def excute(self,msg):
        #print("----")
        '''if msg.data != 1:
            self.turn_right.process()
        elif msg.data ==1 or msg.data ==0 :
            self.turn_right.Go()'''
        #print(msg.data[1].frame_id)
        #print(msg.data[1].frame_id)
        #print(msg.data)
        if msg.data == [] or msg.data[0].frame_id == "Green_light":
            print("Go Now")
            self.move_status.data = 1
            self.pubFlag.publish(self.move_status)
            self.turn_right.Go(self.turn_right.go_angular_z)
            self.turn_right.finish_flag = 0
        elif msg.data[0].frame_id == "Turn_right":
            self.move_status.data = 2
            self.pubFlag.publish(self.move_status)
            if msg.data[0].centery>42 and self.turn_right.finish_flag == 0 : #If
the position height of the camera is changed, the y value here needs to be
calibrated
                self.dist = msg.data[0].centerx * 0.012
                print("self.dist: ",self.dist)
                #print("Turn Now")
                print("self.trun_angular_z: ",self.turn_right.trun_angular_z)
                self.turn_right.Stop()
                sleep(5)
```

The code is an example of detecting the right turn flag,

```
if msg.data[0].centery>42 and self.turn_right.finish_flag == 0
```

In this sentence, msg.data[0].centery is the sent message data. When it is greater than 42, it is considered that it is about to start turning the corner, so there is the following execution statement,You can add identified content in it according to your own needs.

3.2.2. Turning distance parameter

Start up,

```
roslaunch yahboomcar_bringup bringup.launch
python yahboomcar_ws/src/yahboomcar_autodrive/scripts/line_detect.py
rosrun yahboomcar_autodrive test.py
```

When the car recognizes the right turn sign, observe the turning process and make adjustments according to the pressure line situation.

This parameter is mainly for the distance that the car moves forward after recognizing the right turn sign. This distance parameter is to prevent the car from hitting the yellow line when turning. Of course, if you want to turn immediately after recognizing the sign, Then just write the coefficient here as 0. The file directory for the coefficients is,

```
~/yahboomcar_ws/src/yahboomcar_autodrive/scripts/test.py
```

Find the following section,

```
def excute(self,msg):
        #print("----")
        '''if msg.data != 1:
            self.turn_right.process()
        elif msg.data ==1 or msg.data ==0 :
            self.turn_right.Go()'''
        #print(msg.data[1].frame_id)
        #print(msg.data[1].frame_id)
        #print(msg.data)
        if msg.data == [] or msg.data[0].frame_id == "Green_light":
            print("Go Now")
            self.move_status.data = 1
            self.pubFlag.publish(self.move_status)
            self.turn_right.Go(self.turn_right.go_angular_z)
            self.turn_right.finish_flag = 0
        elif msg.data[0].frame_id == "Turn_right":
            self.move_status.data = 2
            self.pubFlag.publish(self.move_status)
            if msg.data[0].centery>42 and self.turn_right.finish_flag == 0 : #If
the position height of the camera is changed, the y value here needs to be
calibrated
                self.dist = msg.data[0].centerx * 0.012
                print("self.dist: ",self.dist)
                #print("Turn Now")
                print("self.trun_angular_z: ",self.turn_right.trun_angular_z)
                self.turn_right.Stop()
                sleep(5)
```

In this part of the code, self.dist is to calculate the distance that needs to be driven forward. It is obtained according to msg.data[0].centerx, which is the center coordinate of the sign frame multiplied by a coefficient. What we need to modify is this coefficient, which is 0.012 in the code. If you want to turn when you recognize the right-turn sign, and adjust the pose without walking forward for a while, just write this coefficient as 0. Modify as follows,

- If you press the yellow line on the inner circle, it means the calculated distance is too small, so increase the coefficient;
- If the yellow line on the outer ring is pressed, it means that the calculated distance is too large, so reduce the coefficient;

3.2.3 Turning time

Start up,

```
roslaunch yahboomcar_bringup bringup.launch
python yahboomcar_ws/src/yahboomcar_autodrive/scripts/line_detect.py
rosrun yahboomcar_autodrive test.py
```

When the car recognizes the right turn sign, observe the turning process and make adjustments according to the situation in front of the car.

This parameter is mainly for the execution time of the turning process after the right turn is recognized and the distance is adjusted forward.What we expect is that after the car turns a corner, the rear wheels or the front of the car can be kept in line with the lane line, so that it will not press the line when going forward.

The file directory where this turn time coefficient exists is,

```
~/yahboomcar_ws/src/yahboomcar_autodrive/scripts/Turn_Right.py
```

Find the following part of the code,

```
def Trun_Right(self,turn_z):
    print("Turn")
    self.move_cmd.linear.x = 0.2
    self.move_cmd.linear.y = 0.0
    self.move_cmd.angular.z = -turn_z
    self.pub_cmdVel.publish(self.move_cmd)
    sleep(5)
    self.move_cmd.linear.x = 0.1
    self.move_cmd.linear.y = 0.0
    self.move_cmd.angular.z = 0.0
    self.pub_cmdVel.publish(self.move_cmd)
```

The sleep(5) in the code is the parameter we need to modify, and the time unit is seconds. Modify according to the actual situation,

- After the car turns, the front of the car turns to the left, and the turning time is not long enough, so increase the time;
- After the car turns, the front of the car turns to the right, and the turning time is too long, so reduce the time;

Note: **After the turning distance parameters and turning time parameters are calibrated, the position of the right turn sign cannot be changed.**

3.2.4. Side parking data package

Before starting this section, you need to place the car at the position where you want to start side parking, and then start,

```
roslaunch yahboomcar_bringup bringup.launch
rosbag record /cmd_vel
```

After the program runs, the handle controls the car to fall into the warehouse. The program will record the data of the topic cmd_vel in this process. After successfully reversing the car, press ctrl+c to end the recording, and a bag file named after the recording start time will be generated in the current terminal directory.When we need to start reversing into the garage, let the program open the terminal and play this package to complete the side parking operation. The file location where the package needs to be replaced is,

```
~/yahboomcar_ws/src/yahboomcar_autodrive/scripts/test.py
```

Find the following partial code,

```
elif msg.data[0].frame_id == "Parking_lotB":
    print("reversing")
    os.system("rosbag play 2022-11-01-21-29-01.bag")
    os._exit(0)
```

Replace the name of the recording package with 2022-11-01-21-29-01 above. It should be noted that the directory where the terminal starts must be consistent with the directory where the package exists, otherwise the package will not be found and an error will occur . For example, if I start the package recording program rosbag record /cmd_vel under home, then the location where the package is stored is in the home directory. Therefore, to start the command rosrun yahboomcar_autodrive test.py, it should also be started by opening the terminal in the home directory. After parking, the program will exit automatically.

Here is a trick, start parking on the side, use the remote control to control the car until the rear wheel is 2-3cm away from the parking space, then turn to the right and start reversing; wait until the right rear wheel enters the parking space, then return to the forward direction,reverse the car; when we need to start reversing into the garage, let the program open the terminal and play this package to complete the side parking operation. The file location where the package needs to be replaced is,