# Face recognition tracking

Note: The virtual machine, ROS-wifi image transmission module and ESP32 communication board ROS_DOMAIN_ID need to be consistent, and both must be set to 20. You can view [ESP32 communication board parameter configuration] to set the ESP32 communication board ROS_DOMAIN_ID, and view the tutorial [Connecting MicroROS Agent] to determine whether the ID is consistent.

**Before running the experiment, please make sure that the microros balance car and ROS-wifi image transmission module have correctly enabled the agent on the virtual machine (linux with humbleROS2 system)**

## 1. Program description

After running the program, when the face is displayed on the screen, when a box appears around the face, the car will move left and right following the movement of the face.

## 2. Operation steps

Program code reference path:

```
/home/yahboom/yahboomcar_ws/src/yahboom_esp32ai_car/yahboom_esp32ai_car/face_fllo
w.py
```

### 2.1. Start command

Terminal input,

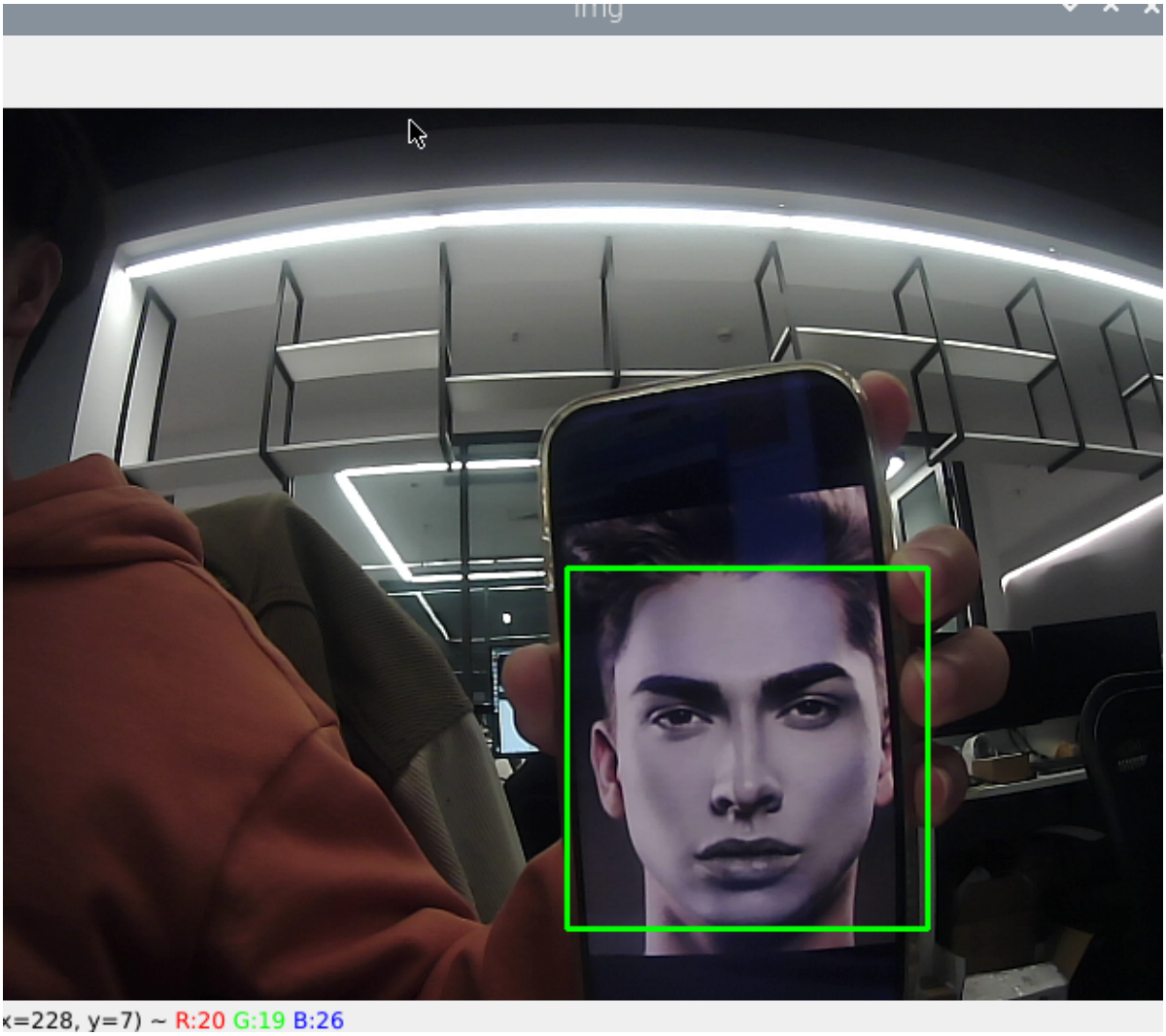
```
#Start chassis driver
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
```

```
#Start face recognition tracking program
ros2 run yahboom_esp32ai_car face_fllow
```

**If the camera image is inverted**, you need to see **3. Camera image correction (select)** document to correct it yourself, this experiment will not be described.

After the program is started, the following camera screen will appear.

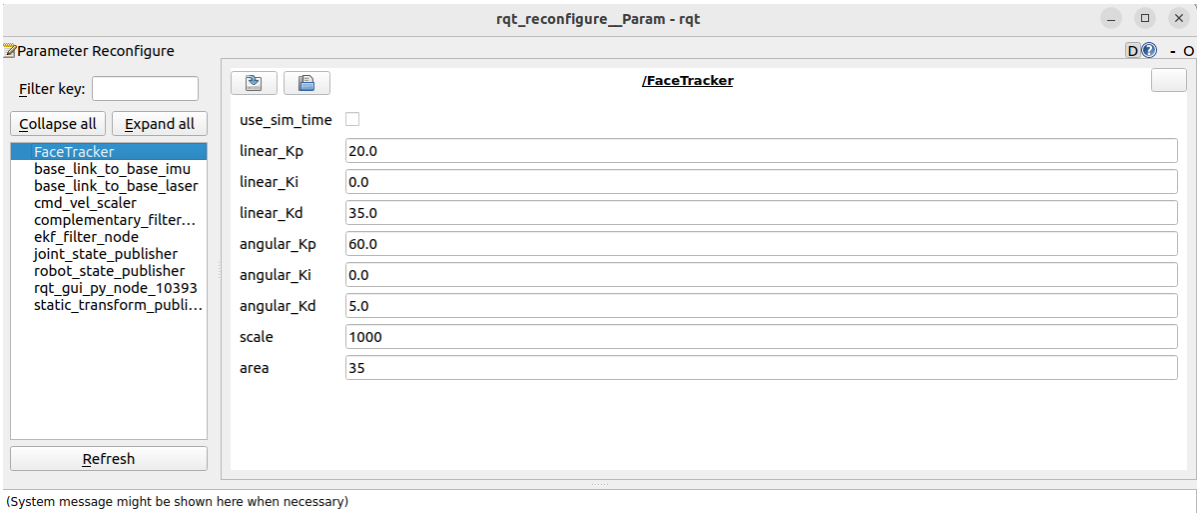

When the face is recognized, it will be selected, and the car will move left and right following the face, and the terminal will print the speed calculated by pid.




## 2.2, Dynamic parameter adjustment

Terminal input,

```
ros2 run rqt_reconfigure rqt_reconfigure
```




After modifying the parameters, click the blank space in the GUI to write the parameter value. Or you can modify the source code directly, and then recompile and update. As shown in the figure above,

- faceTracker mainly adjusts the three parameters of angularPID to make the car more sensitive

# 3. Core code

## 3.1. face_fllow.py

The principle of function implementation is similar to object tracking. Both calculate the rotation speed of the car based on the center coordinates of the target and then publish it to the chassis. Some codes are listed below.

```
#Face Recognition
face_patterns =
cv2.CascadeClassifier('/home/yahboom/yahboomcar_ws/src/yahboom_esp32ai_car/yahboo
m_esp32ai_car/haarcascade_frontalface_default.xml')
        faces = face_patterns.detectMultiScale(frame , scaleFactor=1.1,
minNeighbors=5, minSize=(100, 100))
        if len(faces)>0:
            for (x, y, w, h) in faces:
                m=x
                n=y
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
            self.execute(m,w*h/1000)
        else:
            self.pub_cmdVel.publish(Twist())

#Calculate the car's rotation speed based on the center point
[z_Pid, x_Pid] = self.PID_controller.update([(point_x - 320)/10, (area -
self.area)/2])
if area == 0:
    print("Not Found Face")
    self.pub_cmdVel.publish(Twist())

if self.img_flip == True:
    if area >self.area and x_Pid >0:
        self.twist.linear.x = +x_Pid
    else:
        self.twist.linear.x = -x_Pid
    if point_x > 320 and z_Pid > 0:
        self.twist.angular.z = +z_Pid
    else:
        self.twist.angular.z = -z_Pid
else:
    if area >self.area and x_Pid >0:
        self.twist.linear.x = -x_Pid
    else:
        self.twist.linear.x = +x_Pid
    if point_x > 320 and z_Pid > 0:
        self.twist.angular.z = -z_Pid
    else:
        self.twist.angular.z = +z_Pid
```