QR code motion control

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. Introduction to gameplay

This course mainly uses the robot's camera to obtain the camera's image, identify the QR code information, and control the robot's movement according to the QR code information.

2. Program code reference path

The source code of this function is located at,

```
/home/yahboom/yahboomcar_ws/src/yahboom_esp32ai_car/yahboom_esp32ai_car/qrTracker
.py
```

3. Program startup

3.1. Start command

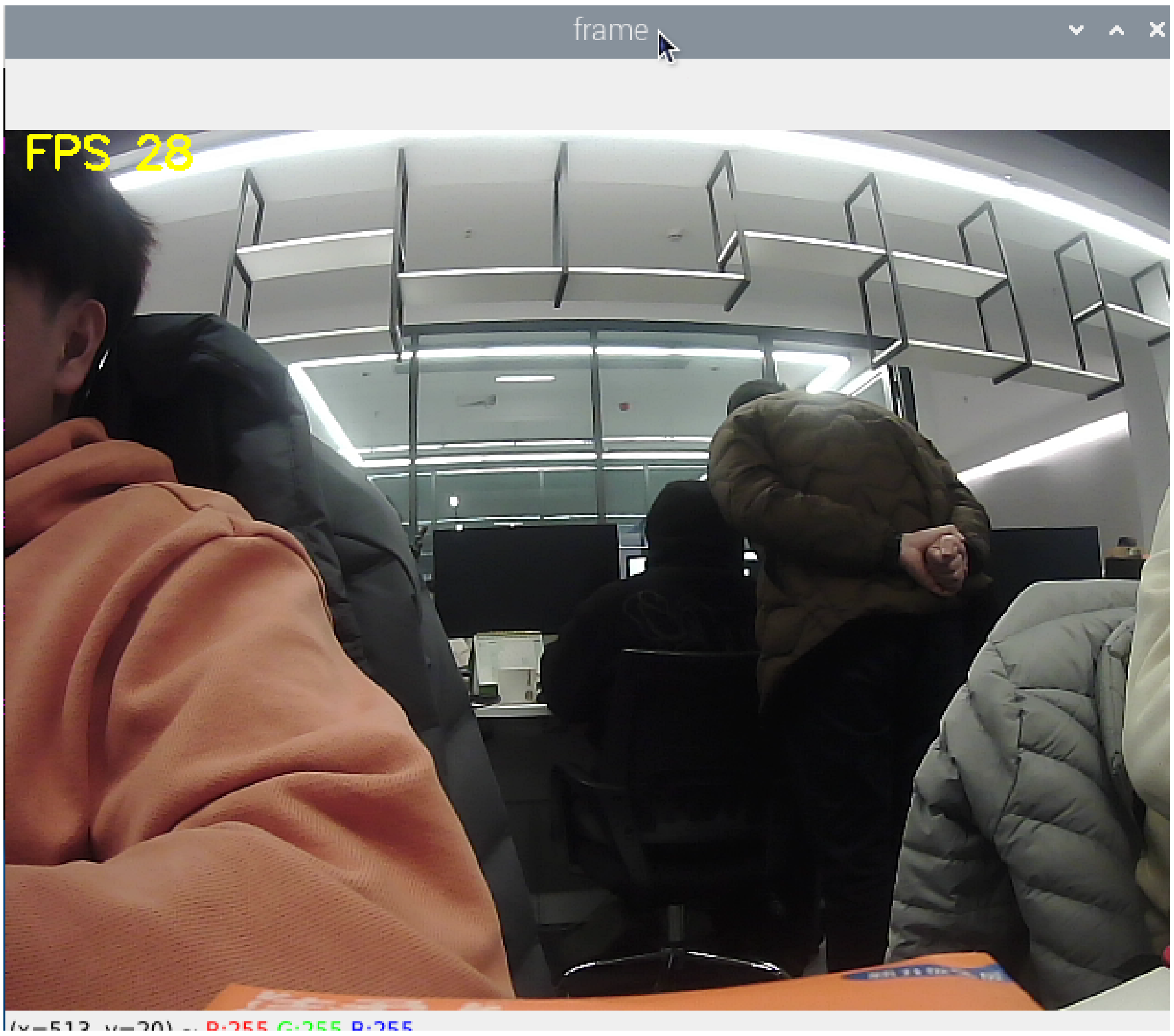
Input in the terminal,

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

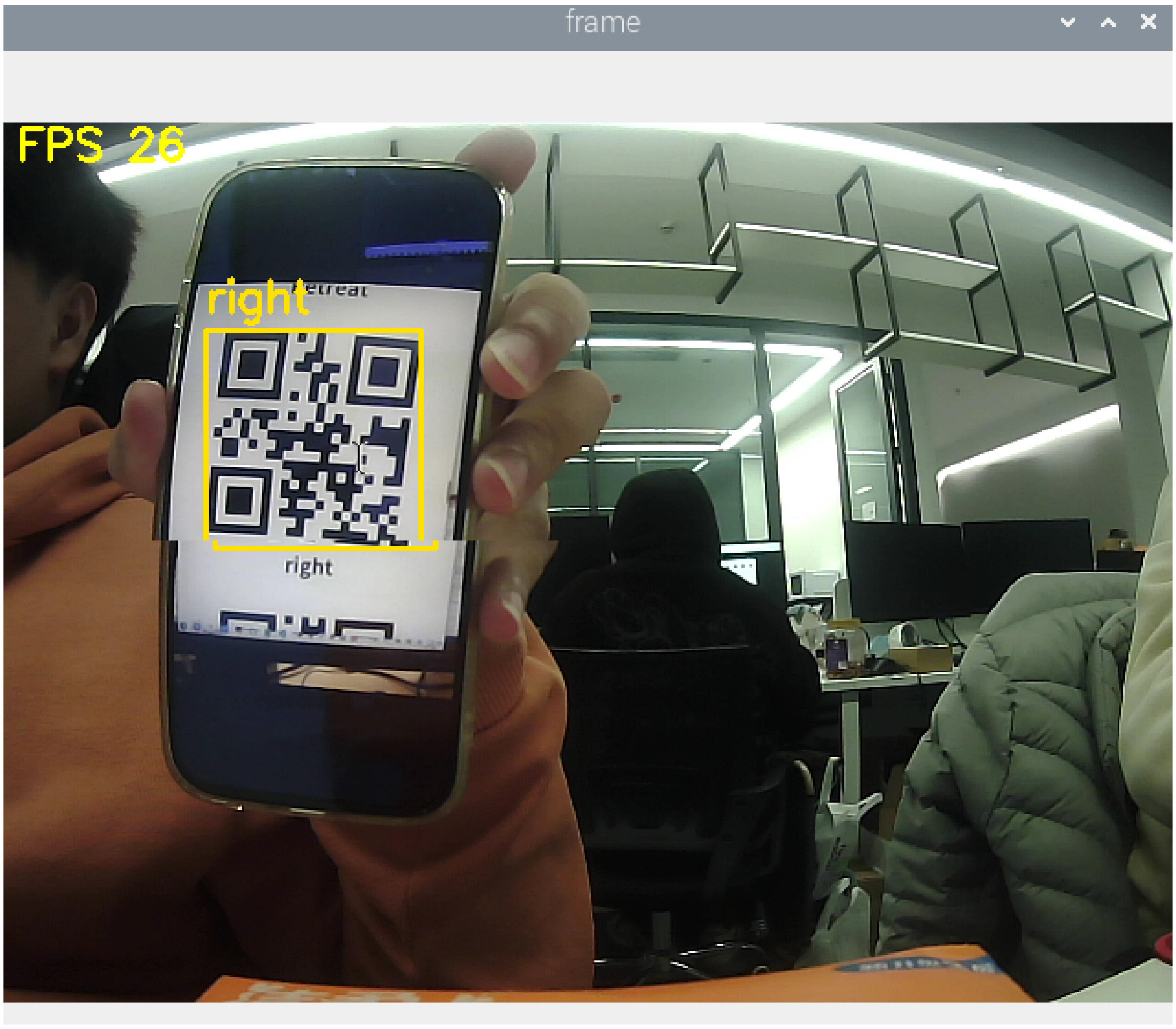
```
#Start QR code motion control program
ros2 run yahboom_esp32ai_car qrTracker
```

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

The camera screen is successfully displayed.



Start recognizing QR codes and executing instructions. The QR code that can be recognized in the current routine is QRCode, and the information is "forward" for forward, "back" for backward, "left" for left translation, "right" for right translation, "turnleft" for left rotation, "turnright" for right rotation, and "stop" for stop.



Press q to turn off the camera.

4. Core code

Import the QR code parsing library pybar

```
import pyzbar.pyzbar as pyzbar
from PIL import Image
```

If pybar is not installed on your system, open the terminal and run the following command to install it.

The environment has been configured at the factory, so this step is suitable for your own development.

```
pip3 install pyzbar
sudo apt install libzbar-dev
```

Parse the grayscale image and extract the QR code information and image position in the image. If there is no QR code in the image, the information is None.

```
def detect_qrcode(self,image):
    # Convert to grayscale image
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    barcodes = pyzbar.decode(gray)
    for barcode in barcodes:
    # The data of the QR code and the position of the bounding box are extracted
        (x, y, w, h) = barcode.rect
        barcodeData = barcode.data.decode("utf-8")
        barcodeType = barcode.type
        return barcodeData, (x, y, w, h)
    return None, (0, 0, 0, 0)
```

Control the robot movement according to the string command of info.

```
def robot_action(self,data):
    if data == "forward":
        self.pub_vel(0.35,0.0,0.0)
    elif data == "back":
        self.pub_vel(-0.35,0.0,0.0)
    elif data == "left":
        self.pub_vel(0.0,0.0,1.5)
    elif data == "right":
        self.pub_vel(0.0,0.0,-1.5)
    elif data == "turnright":
        self.pub_vel(0.35,0.0,-1.5)
    elif data == "turnleft":
        self.pub_vel(0.35,0.0,1.5)
    elif data == "stop":
        self.pub_vel(0.0,0.0,0.0)
```

Image processing program

```
frame = self.bridge.compressed_imgmsg_to_cv2(msg)
frame = cv2.resize(frame, (640, 480))
action = cv2.waitKey(10) & 0xFF

payload, (x, y, w, h) = self.QRdetect.detect_qrcode(frame.copy())
if payload != None:
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 225, 255), 2)
    cv2.putText(frame, payload, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,
225, 255), 2)
    self.QRdetect.robot_action(payload)
else:
    self.QRdetect.pub_vel(0.0,0.0,0.0)
```

Appendix (QR code image):

forward

back

left

right

stop

turnleft

turnright