Object 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 function description

After the program is started, select the object to be tracked with the mouse, press the space bar, and the car enters the tracking mode. The car will follow the tracked object left and right, and always ensure that the tracked object remains in the center of the screen.

2. Operation steps

2.1. Start

Input in terminal,

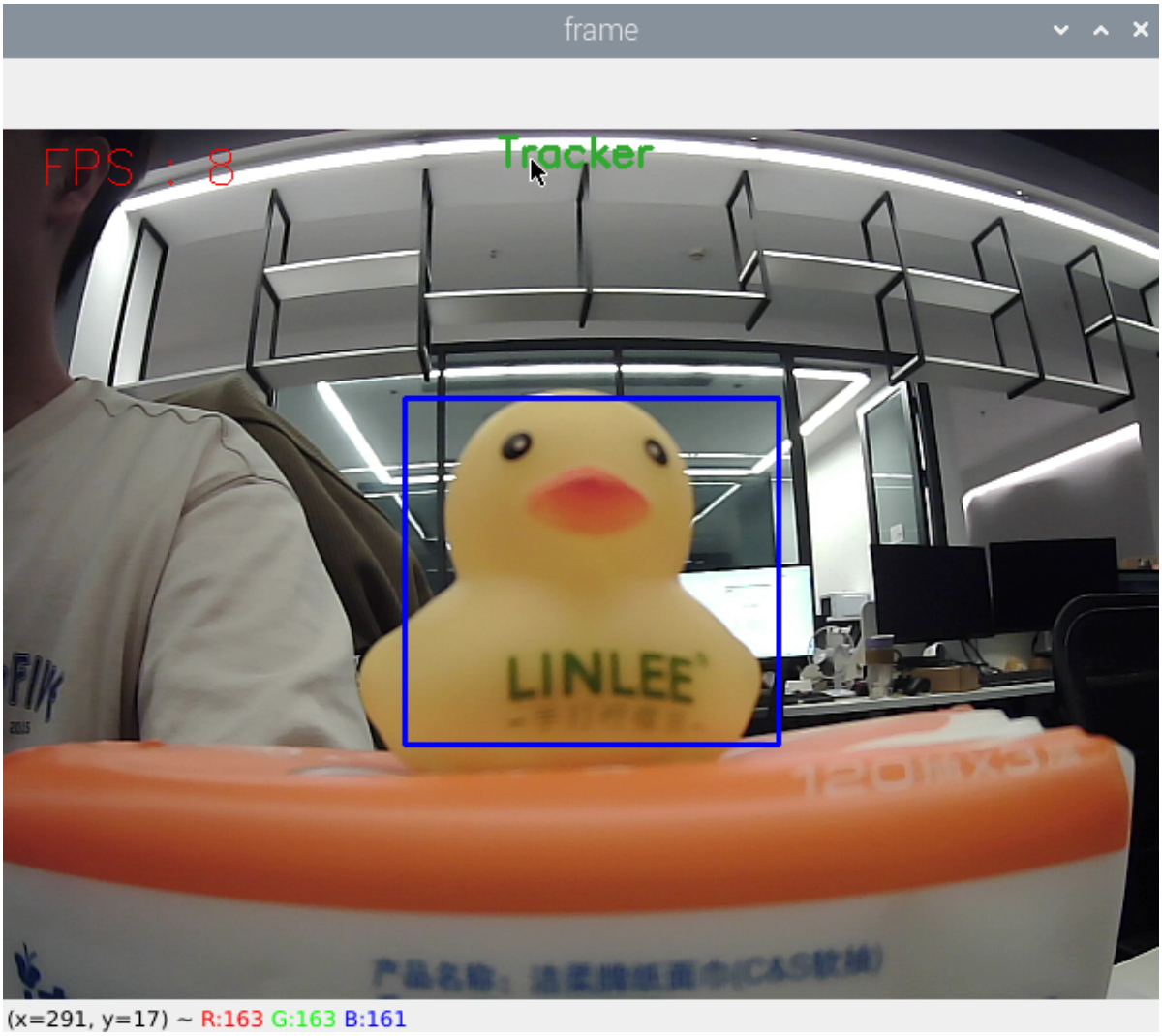
```
#Start chassis driver
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
```

```
#Start object tracking program
ros2 run yahboom_esp32ai_car mono_Tracker
```

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.

2.2. Recognition

After starting, enter the selection mode, select the target location with the mouse, as shown in the figure below, and release it to start recognition.



Keyboard control:

[r]: Select mode, you can use the mouse to select the area to identify the target, as shown above.

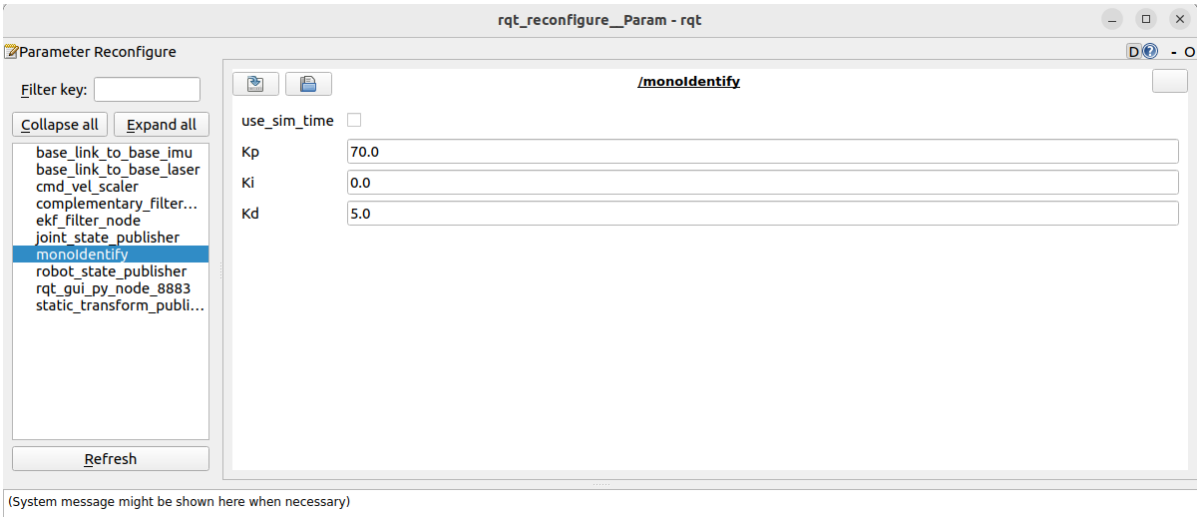
[q]: Exit the program.

[Spacebar]: Target tracking; just move the target slowly when following, moving too fast will lose the target.

Press the spacebar to see the speed calculated by pid.

You can also use the dynamic parameter regulator to debug the PID parameters, terminal input,

```
ros2 run rqt_reconfigure rqt_reconfigure
```



As can be seen from the above figure, the adjustable parameters are the three PID parameters of the car. After modifying the parameters, press the R key to take effect and reset. Or modify the source code directly, and then recompile and update.

2.3. View node topic communication

You can view the topic communication between nodes through the following command,

```
ros2 run rqt_graph rqt_graph
```

3. Core code

The principle of function implementation is similar to color 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.

```
#This part is to get the center coordinates after selecting the object, which is
used to calculate the rotation speed of the car
if self.tracker_type != "color":
    if self.gTracker_state == True:
        Roi = (self.Roi_init[0], self.Roi_init[1], self.Roi_init[2] -
self.Roi_init[0], self.Roi_init[3] - self.Roi_init[1])
        self.gTracker = Tracker(tracker_type=self.tracker_type)
        self.gTracker.initWorking(rgb_img, Roi)
        self.gTracker_state = False
    rgb_img, (targBegin_x, targBegin_y), (targEnd_x, targEnd_y) =
self.gTracker.track(rgb_img)
    center_x = targEnd_x / 2 + targBegin_x / 2
    center_y = targEnd_y / 2 + targBegin_y / 2
    width = targEnd_x - targBegin_x
    high = targEnd_y - targBegin_y
    area = width * high
    self.point_pose = (center_x, center_y, min(width, high))

if self.Track_state == 'tracking':
   if self.circle[2] != 0: threading.Thread(target=self.execute, args=
(self.circle[0], self.circle[1])).start()

# Calculate the speed of the car through pid
```

```
[z_Pid, x_Pid] = self.PID_controller.update([(point_x - 320) / 8, (area - 40) /
8])
if self.img_flip == True:
    if area > 40 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 > 40 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
if z_Pid > 1.0:z_Pid=1.0
if z_Pid < -1.0:z_Pid=-1.0
if abs(point_x - 320) < 40:
    self.twist.angular.z = 0.0
if 40 - 20 < area < 40 or abs(x_Pid) > 0.8:
    self.twist.linear.x = 0.0
self.twist.linear.x = 0.0
print("z_Pid:{:.2f}, x_Pid:{:.2f},point_x:{:.2f},area:{:.2f}".format(z_Pid,
x_Pid, point_x, area))
self.pub_cmdVel.publish(self.twist)
```