Object recognition and tracking

Note: The ROS_DOMAIN_ID of the Raspberry Pi and the microROS control board need to be consistent. You can check [MicroROS Control Board Parameter Configuration] to set the microROS control board ROS_DOMAIN_ID. Check the tutorial [Connect MicroROS Agent] to determine whether the IDs are consistent.

1、Program function description

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

2、Steps

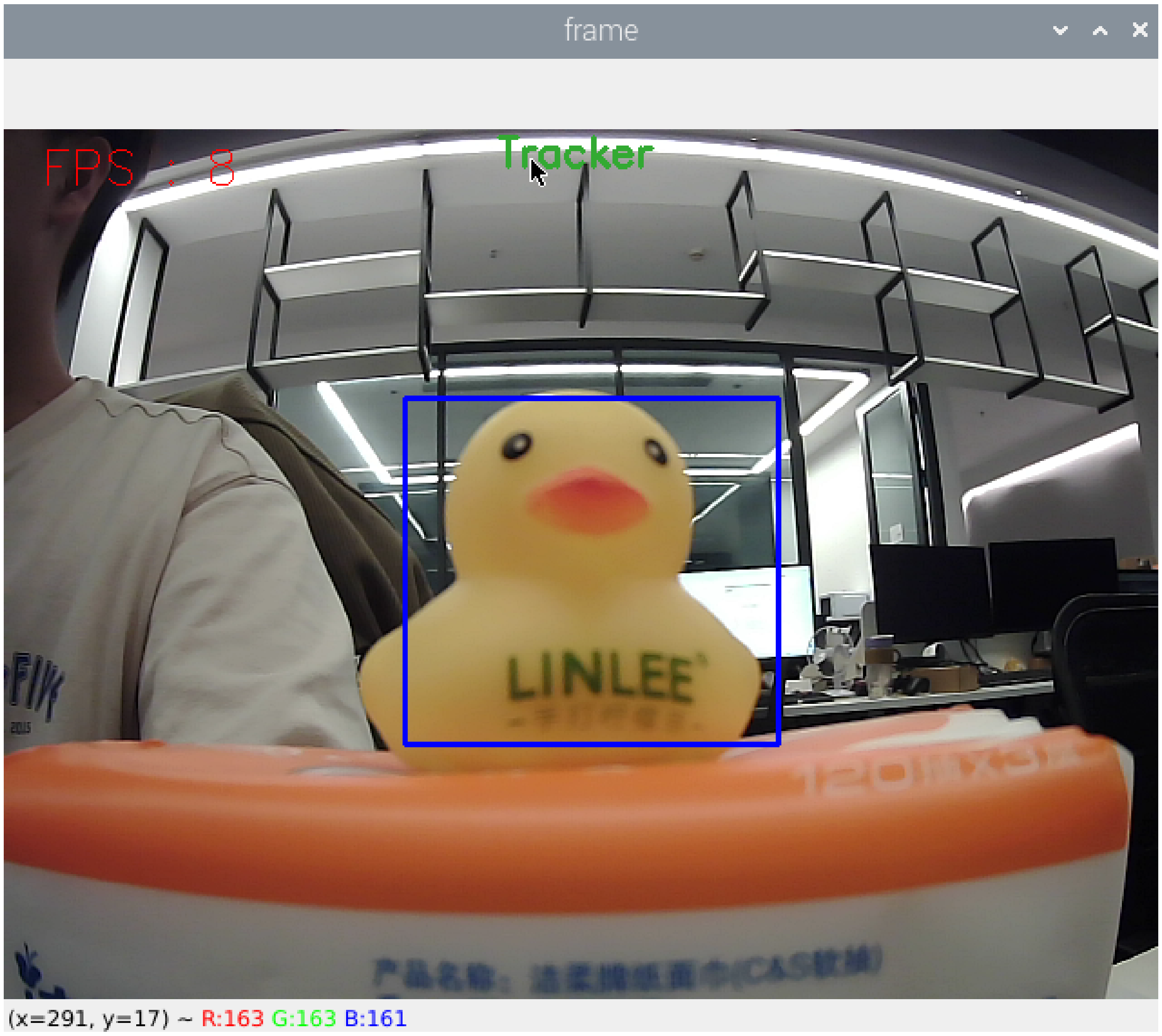
2.1、start up

Enter docker and enter the following command in the terminal.

```
ros2 run yahboomcar_astra mono_Tracker
```

2.2、identify

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

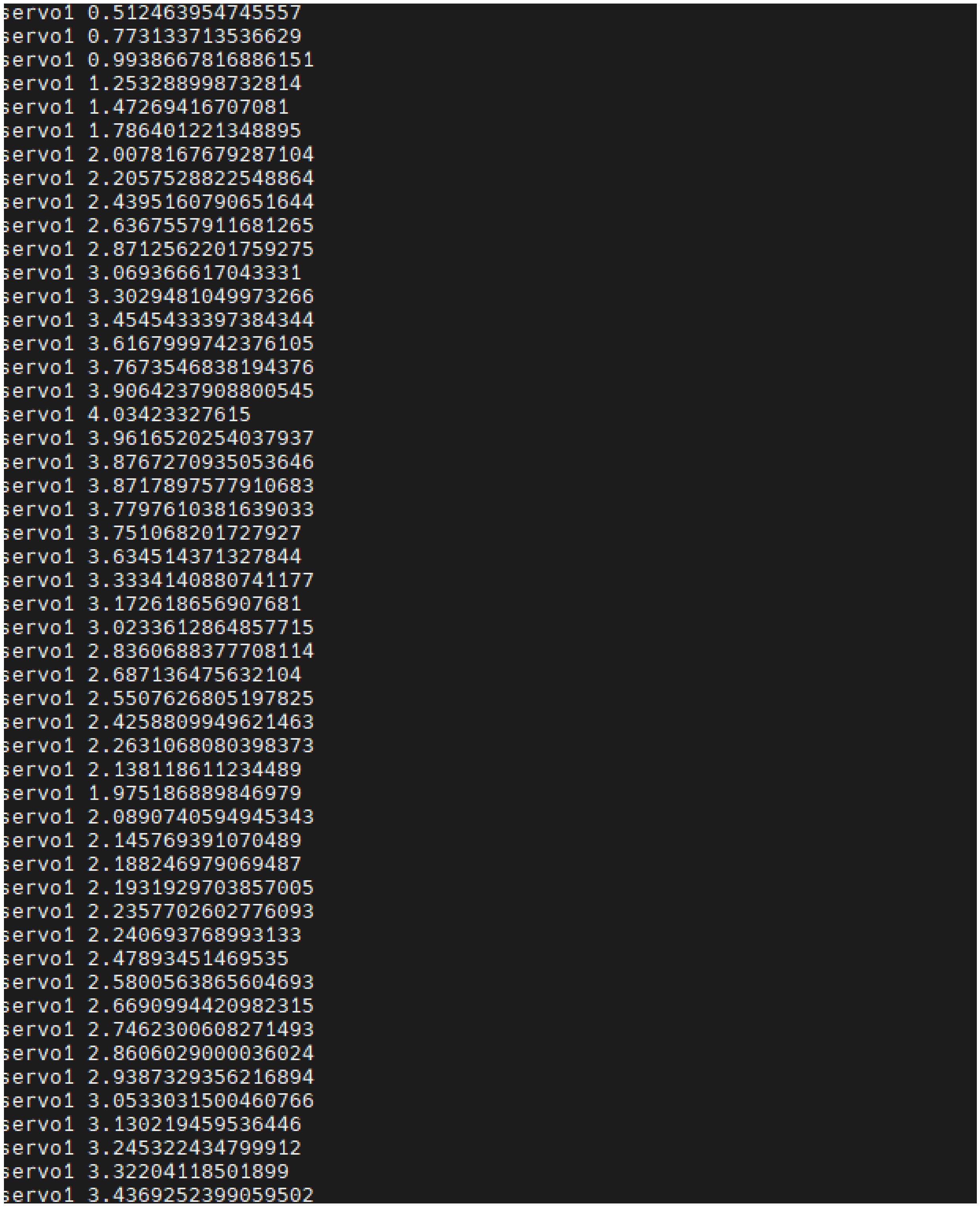


Keyboard key control:

【r】: In selection mode, you can use the mouse to select the area where you want to identify the target, as shown in the picture above.

【q】: exit the program.

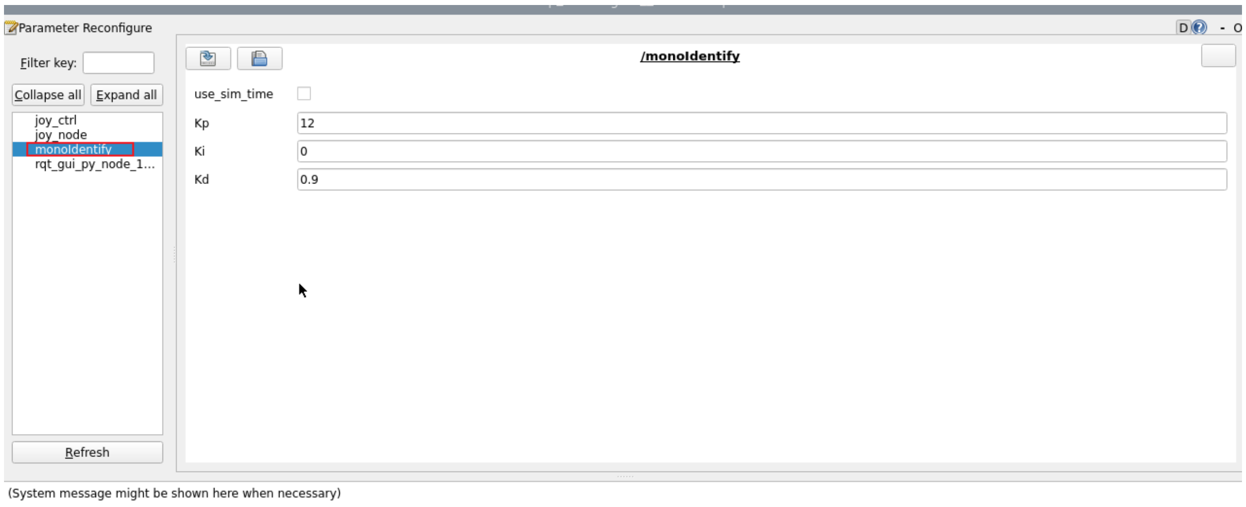
【space bar】: Target tracking; just move the target slowly while following. If you move too fast, you will lose the target.



```
servo1 0.512463954745557
servo1 0.773133713536629
servo1 0.9938667816886151
servo1 1.253288998732814
servo1 1.47269416707081
servo1 1.786401221348895
servo1 2.0078167679287104
servo1 2.2057528822548864
servo1 2.4395160790651644
servo1 2.6367557911681265
servo1 2.8712562201759275
servo1 3.069366617043331
servo1 3.3029481049973266
servo1 3.4545433397384344
servo1 3.6167999742376105
servo1 3.7673546838194376
servo1 3.9064237908800545
servo1 4.03423327615
servo1 3.9616520254037937
servo1 3.8767270935053646
servo1 3.8717897577910683
servo1 3.7797610381639033
servo1 3.751068201727927
servo1 3.634514371327844
servo1 3.3334140880741177
servo1 3.172618656907681
servo1 3.0233612864857715
servo1 2.8360688377708114
servo1 2.687136475632104
servo1 2.5507626805197825
servo1 2.4258809949621463
servo1 2.2631068080398373
servo1 2.138118611234489
servo1 1.975186889846979
servo1 2.0890740594945343
servo1 2.145769391070489
servo1 2.188246979069487
servo1 2.1931929703857005
servo1 2.2357702602776093
servo1 2.240693768993133
servo1 2.47893451469535
servo1 2.5800563865604693
servo1 2.6690994420982315
servo1 2.7462300608271493
servo1 2.8606029000036024
servo1 2.9387329356216894
servo1 3.0533031500460766
servo1 3.130219459536446
servo1 3.245322434799912
servo1 3.32204118501899
servo1 3.4369252399059502
```

You can also use the dynamic parameter adjuster to debug PID parameters and input it into the Docker terminal.

```
ros2 run rqt_reconfigure rqt_reconfigure
```



use_sim_time

Kp 12

Ki 0

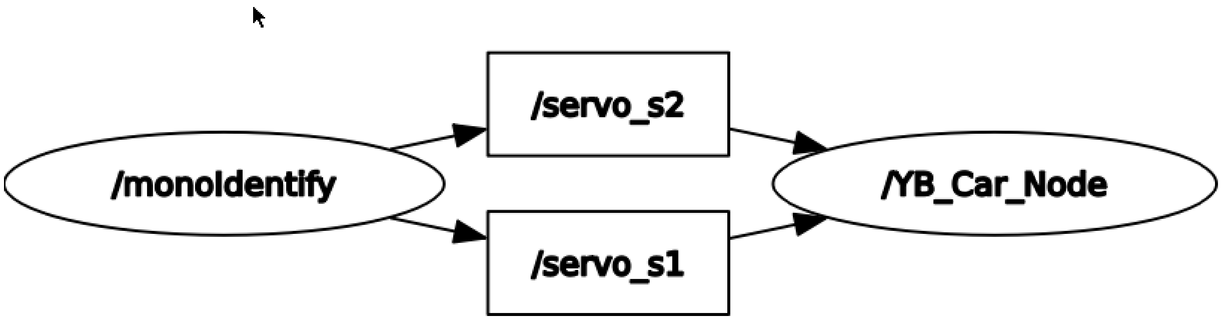
Kd 0.9

(System message might be shown here when necessary)

The adjustable parameters include the PID parameters of the car gimbal. After modifying the parameters, click "refresh" to refresh the data.

2.3、View node topic communication diagram

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



3、core code

The principle of function implementation is similar to that of color tracking. It calculates the rotation angle of s1 and s2 servos based on the sum of the center coordinates of the target, and then publishes it to the chassis. Part of the code is as follows.

```
#This part is to obtain the center coordinates after selecting the object, which is
used to calculate the steering gear angle.
if len(contours) != 0:
    areas = []
    for c in range(len(contours)): areas.append(cv.contourArea(contours[c]))
    max_id = areas.index(max(areas))
    max_rect = cv.minAreaRect(contours[max_id])
    max_box = cv.boxPoints(max_rect)
    max_box = np.int0(max_box)
    (color_x, color_y), color_radius = cv.minEnclosingCircle(max_box)
```

```
#This part is to calculate the values of center_x and distance.
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
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 steering gear movement angle through pid
[x_Pid, y_Pid] = self.PID_controller.update([point_x - 320, point_y - 240])
   if self.img_flip == True:
        self.target_servox -= x_Pid
        self.target_servoy += y_Pid
   else:
        self.target_servox -= x_Pid
        self.target_servoy += y_Pid
   if self.target_servox >= 90:
        self.target_servox = 90
   elif self.target_servox <= -90:
        self.target_servox = -90
   if self.target_servoy >= 20:
        self.target_servoy = 20
   elif self.target_servoy <= -90:
        self.target_servoy = -90
```