

Color Following

This lesson uses the Raspberry Pi as an example. For Raspberry Pi and Jetson-Nano boards, you need to open a terminal on the host computer and enter the command to enter the Docker container. Once inside the Docker container, enter the commands mentioned in this lesson in the terminal. For instructions on entering the Docker container from the host computer, refer to [\[01. Robot Configuration and Operation Guide\] -- \[4.Enter Docker \(For JETSON Nano and RPi 5\)\]](#). For RDKX5 and Orin boards, simply open a terminal and enter the commands mentioned in this lesson.

1. Program Functionality

After the program starts, the default tracking color is red. Press the `[r/R]` key to enter color selection mode. Select a color with the mouse, and the screen will lock to that color. Press the `[Spacebar]` to enter follow mode. The robot will always keep the object it is following centered in the screen and maintain a distance of 0.4 meters from the object. Press the `[q/Q]` key to exit the program.

2. Program Code Reference Path

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_depth/yahboomcar_depth/Advanced/colorTracker.py  
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_depth/yahboomcar_depth/astra_common.py
```

- `colorTracker.py`

Mainly performs image processing. Based on the center coordinates of the color object being tracked and the depth distance information, it calculates the vehicle's desired travel distance and turning angle, and sends control direction and travel data to the vehicle.

- `astra_common.py`

- Color tracking (`color_follow` class): Identifies and tracks objects of a specific color using the HSV color space.
- Image processing tools: Includes multi-image stitching (`manyimgs` function) and shape detection (`is_regular_square` function).
- File read/write functionality: Used to save and read HSV color ranges.

3. Program Startup

3.1. Startup Commands

For Raspberry Pi and Jetson-Nano boards, you need to enter the Docker container first. For RDKX5 and Orin controllers, this is not necessary.

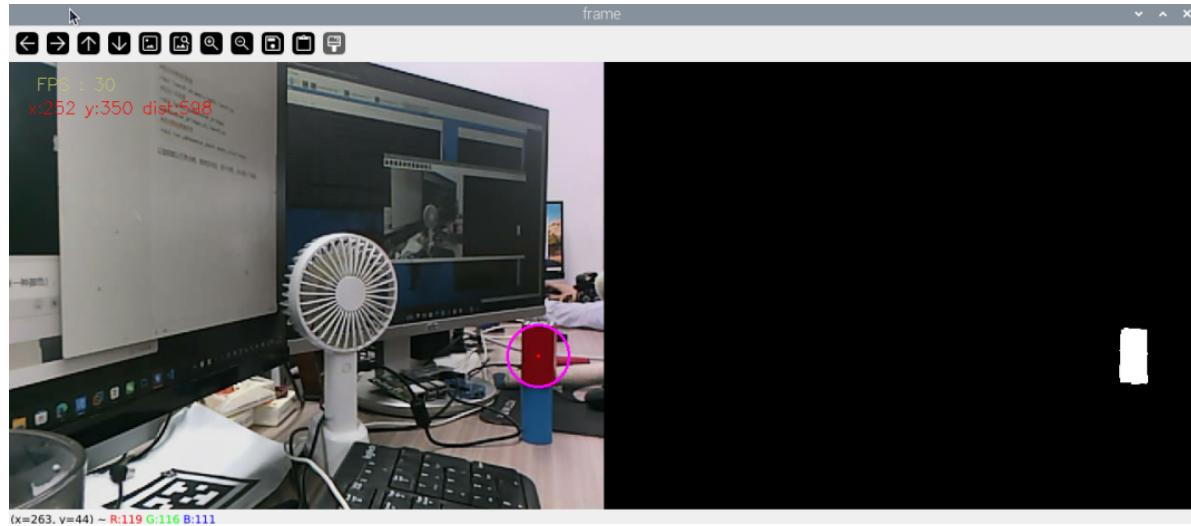
Enter the Docker container (see [\[Docker course\] --- \[4. Docker Startup Script\]](#) for steps).

All the following commands must be executed within the same Docker container (**see [\[Docker course\] --- \[3. Docker Submission and Multi-Terminal Access\]](#) for steps**).

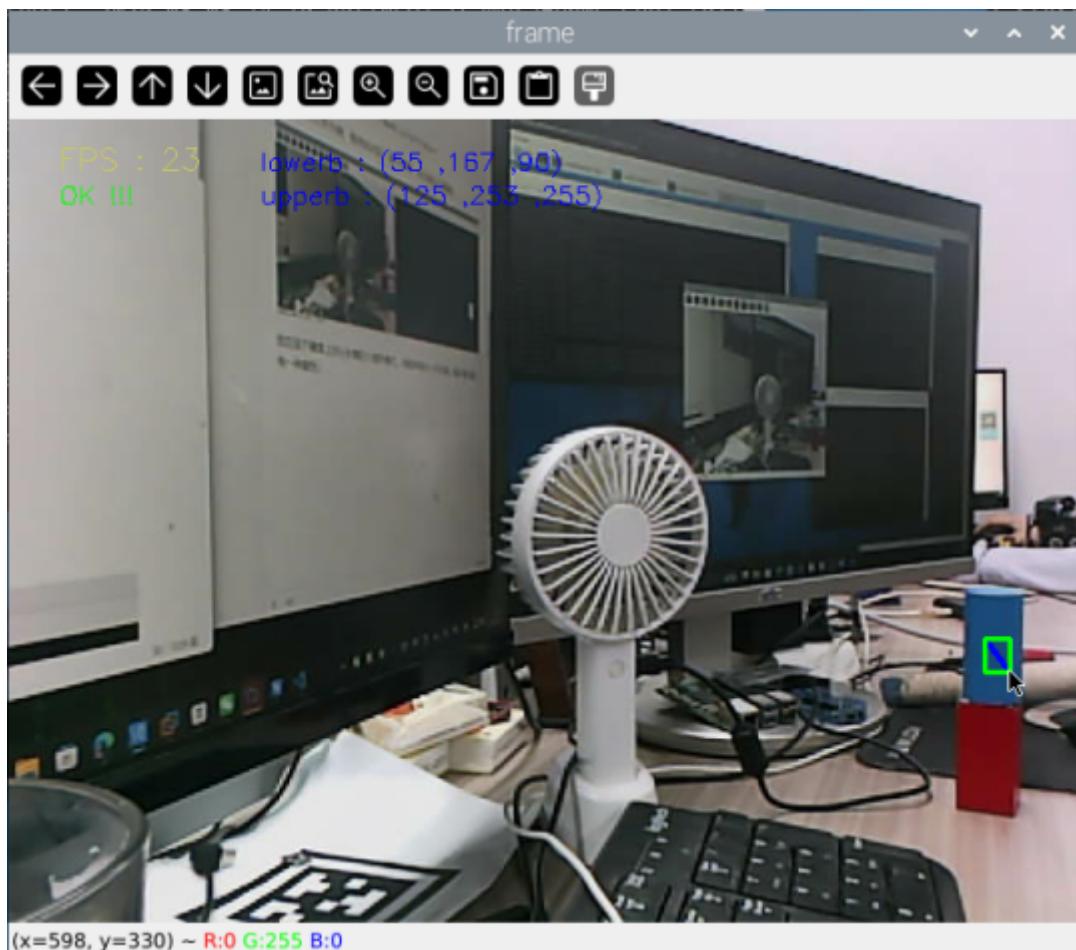
Enter the terminal:

```
#Start depth camera data
ros2 launch ascamera hp60c.launch.py
# Start the car chassis
ros2 launch yahboomcar_bringup yahboomcar_bringup_M1_launch.py
# Start the color tracking program
ros2 run yahboomcar_depth depth_colorTracker
```

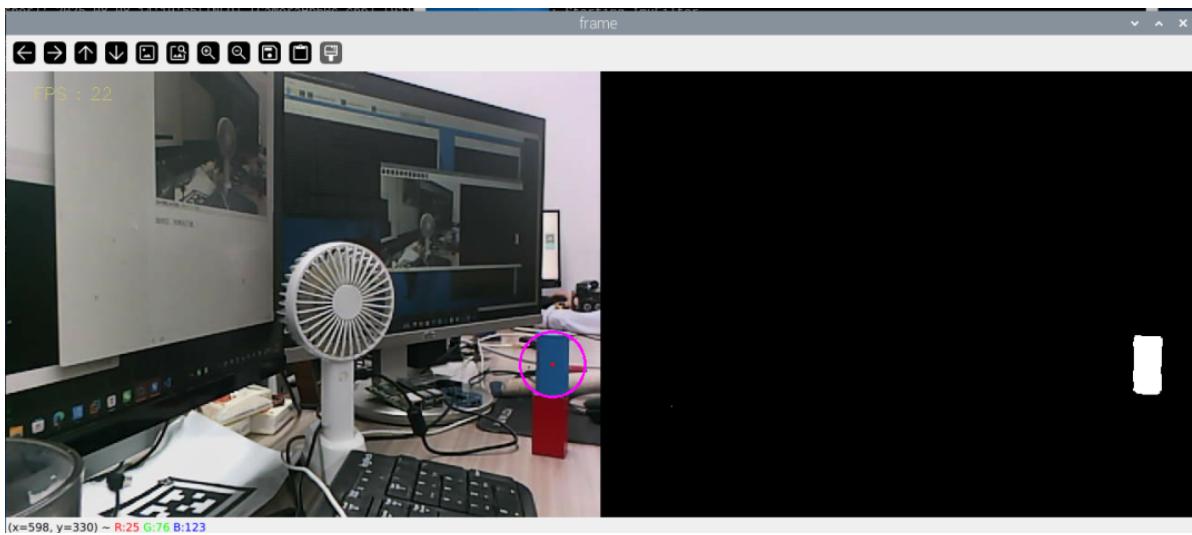
For example, after starting the program and pressing the spacebar, the following screen will appear:



Then press the `r/R` key on your keyboard to enter color selection mode. Use your mouse to select an area (this area can only have one color).



After making this selection, the effect will be as shown below.

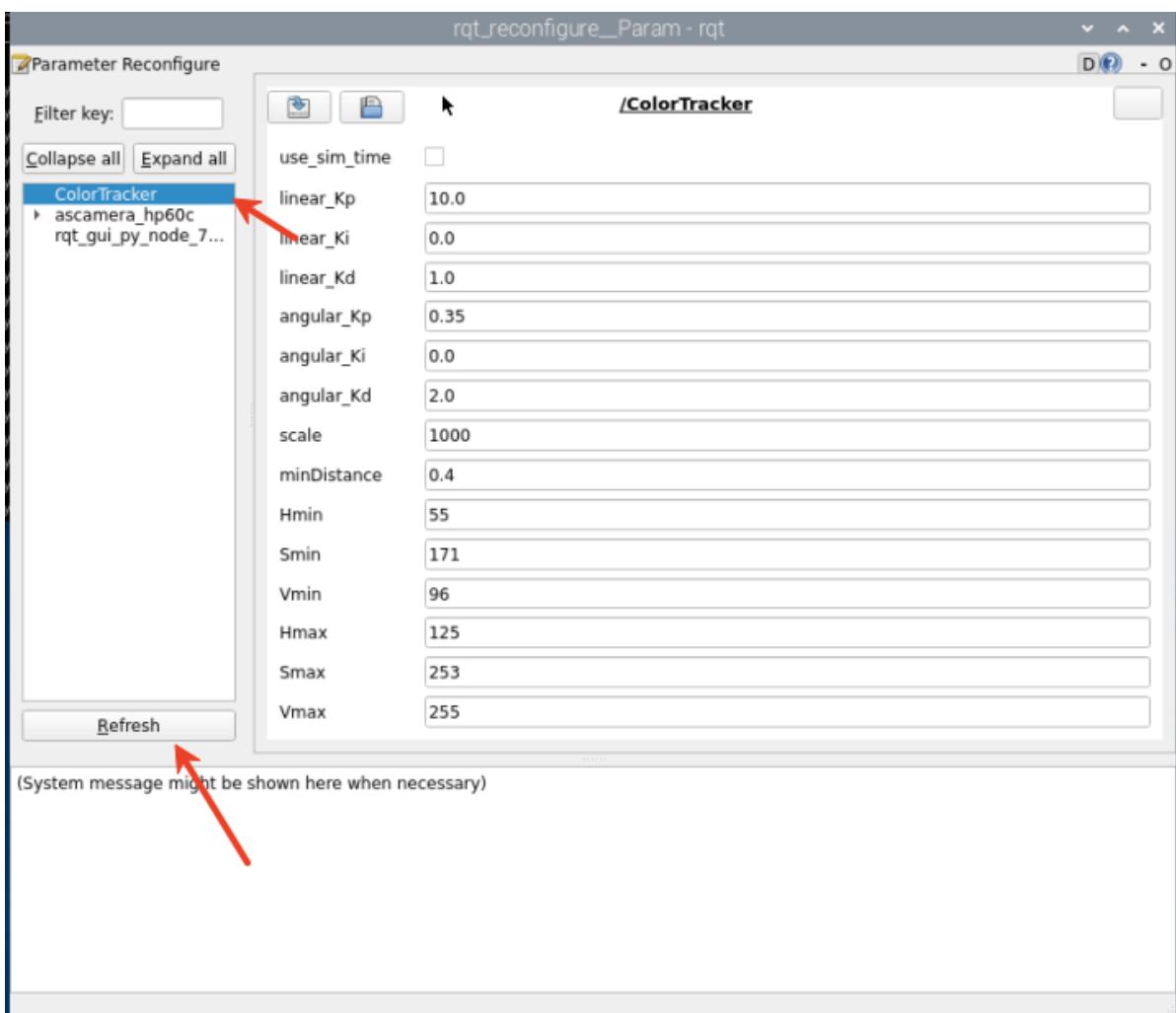


Then, press the spacebar to enter follow mode. Slowly move the object, and the robot will follow. Note that if the object's depth information is 0, the robot will stop. Therefore, it is recommended to select a larger object to obtain better depth information.

3.2. Dynamic Parameter Adjustment

Input in the terminal,

```
ros2 run rqt_reconfigure rqt_reconfigure
```



- After modifying the parameters, click a blank area in the GUI to write the parameter values. Note that the values will only take effect for the current startup. To permanently take effect, you need to modify the parameters in the source code.

Parameter Explanation:

[linear_Kp], [linear_Ki], [linear_Kd]: PID control of linear velocity during the car following process.

[angular_Kp], [angular_Ki], [angular_Kd]: PID control of angular velocity during the car following process.

[minDistance]: Following distance, which is maintained at this distance.

[scale]: PID scaling.

[Hmin], [Smin], [Vmin]: Indicates the minimum HSV value.

[Hmax], [Smax], [Vmax]: Indicates the maximum HSV value.

4. Core Code

4.1. colorTracker.py

This program has the following main functions:

- Subscribe to the depth camera topic to obtain depth images;
- Receive keyboard and mouse events for mode switching and color extraction;
- Process the image to obtain the object's center coordinates and depth distance information.
- Calculate the distance PID and center coordinate PID to obtain the vehicle's forward speed, distance, and turning angle.

Some of the core code is as follows:

```
# 重要的函数，获取到x值、y值和distance_值Important functions to get x value, y value
and distance_ value
def process(self, rgb_img, action):

#目标中心点深度测量 Target center point depth measurement
if self.Center_r > 5:
    points = [
        (int(self.Center_y), int(self.Center_x)),           # 中心点 Center Point
        (int(self.Center_y + 1), int(self.Center_x + 1)),   # 右下偏移点 Lower right
    offset point
        (int(self.Center_y - 1), int(self.Center_x - 1))    # 左上偏移点 Upper left
    offset point
    ]
    valid_depths = []
    for y, x in points:
        depth_val = depth_image_info[y][x]
        if depth_val != 0:
            valid_depths.append(depth_val)
    if valid_depths:
        dist = int(sum(valid_depths) / len(valid_depths))
    else:
        dist = 0
    self.dist = dist
#当距离大于0.2米且启动状态为True时，调用 execute 方法控制机器人运动
#when the distance is greater than 0.2 meters and the start state is True, call
the execute method to control the robot movement
    if dist > 0.2 and self.start_state == True:
        self.execute(self.Center_x, dist)
```

```
# 根据x值、y值，使用PID算法，计算运动速度，转弯角度 According to the x value and y  
value, use the PID algorithm to calculate the movement speed and turning angle  
def execute(self, rgb_img, action):  
    #PID计算偏差 PID calculation deviation  
    linear_x = self.linear_pid.compute(dist, self.minDist)  
    angular_z = self.angular_pid.compute(point_x, 320)  
    # 小车停车区间，速度为0 The car is in the parking area and the speed is 0  
    if abs(dist - self.minDist) < 30: linear_x = 0  
    if abs(point_x - 320.0) < 30: angular_z = 0  
    twist = Twist()  
    ...  
    # 将计算后的速度信息发布 Publish the calculated speed information  
    self.pub_cmdVel.publish(twist)
```