Visual tracking autonomous driving

This lesson uses a Raspberry Pi 5 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] --[5.Enter Docker (For JETSON Nano and RPi 5)]**.

For Orin boards, simply open a terminal and enter the commands mentioned in this lesson.

1. Program Function Description

> After starting the program, adjust the camera's pitch angle and tilt the camera downward so that the line can be seen. Then, click the image window and press the R key to enter color selection mode. Then, select the color you want to track within the line area in the image. Release the mouse button and the processed image will automatically load. Finally, press the spacebar to enable the line tracking function. During operation, the car will stop and a buzzer will sound if it encounters an obstacle. After enabling the controller control program, press the R2 key on the controller to pause the car's movement.

2. Code Reference Path

For the Raspberry Pi 5 controller, you must first enter the Docker container. For the Orin board, this is not necessary.

```
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_depth/yahboomcar_depth/Advanced/follow_line.py
~/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_depth/yahboomcar_depth/follow_common.py
```

- follow_line.py

  Mainly performs image processing, obtains the center coordinates based on the calibrated color, and subscribes to radar topic data. Calculates the car's turning angle, calculates the distance to obstacles, stops, and warns the car, then publishes forward, turning, and buzzer data to the car.

- follow_common.py
  - File read/write functions: used to save/read the HSV color range
  - Image processing tools: including multi-image stitching and display (ManyImgs function)
  - Color line follower (line_follow class): extracts the center coordinates and radius of a color using the HSV color space
  - Get HSV from ROI: obtains the HSV value within the ROI range by specifying the ROI range

3. Program Startup

3.1. Startup Command

Enter the terminal:

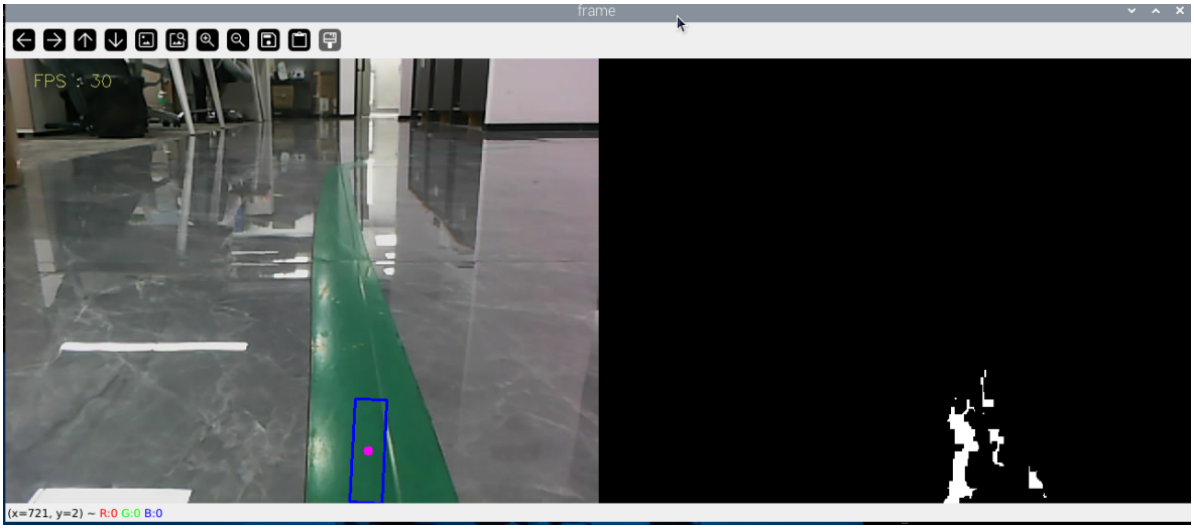
For the Raspberry Pi 5 controller, you must first enter the Docker container; the Orin controller does not need to.

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

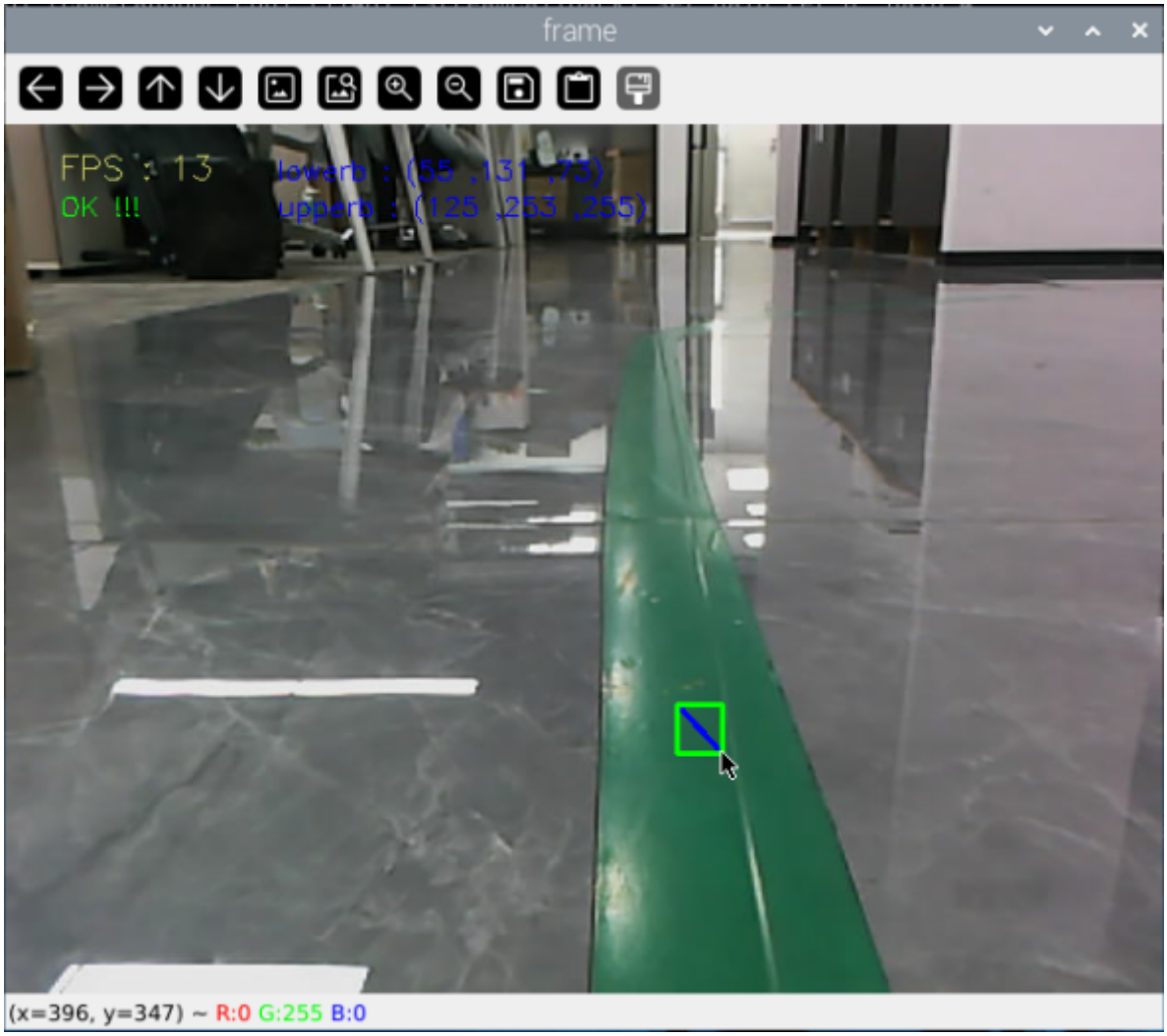
All the following commands must be executed from the Docker terminal within the same Docker container **(see [Docker Course] for steps). [3. Docker Submission and Multi-Terminal Access]**

```
# Start the depth camera data
ros2 launch ascamera hp60c.launch.py
# Start the car chassis and radar
ros2 launch yahboomcar_bringup laser_bringup_launch.py
# Start the color tracking program
ros2 run yahboomcar_depth depth_follow_line
```

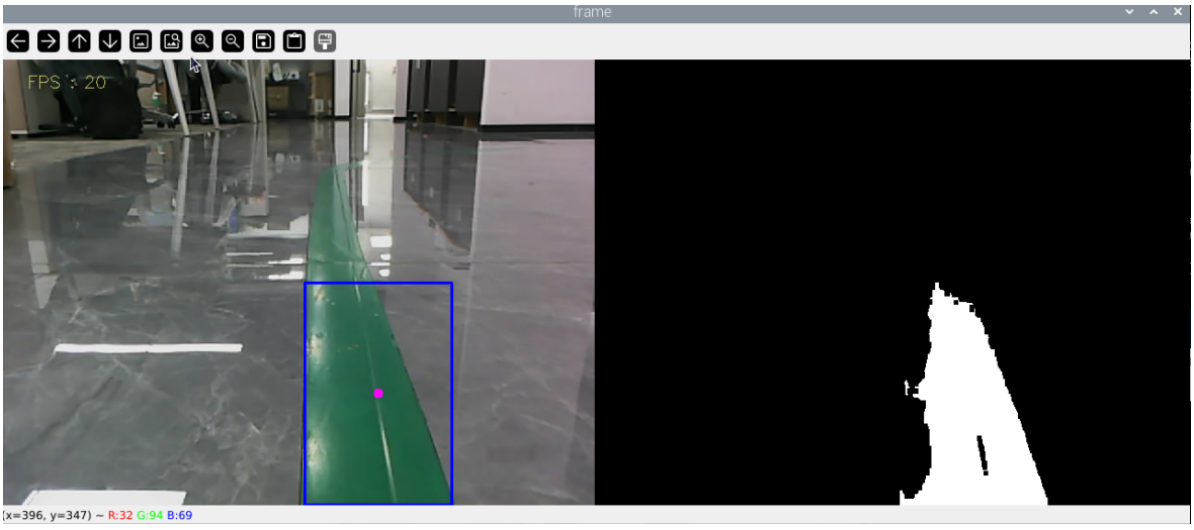
After running the program, the following screen will appear. Taking the green line patrol as an example,



The default HSV value will be used upon startup. You will need to select a new color. 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 selecting, the result will be as shown below.

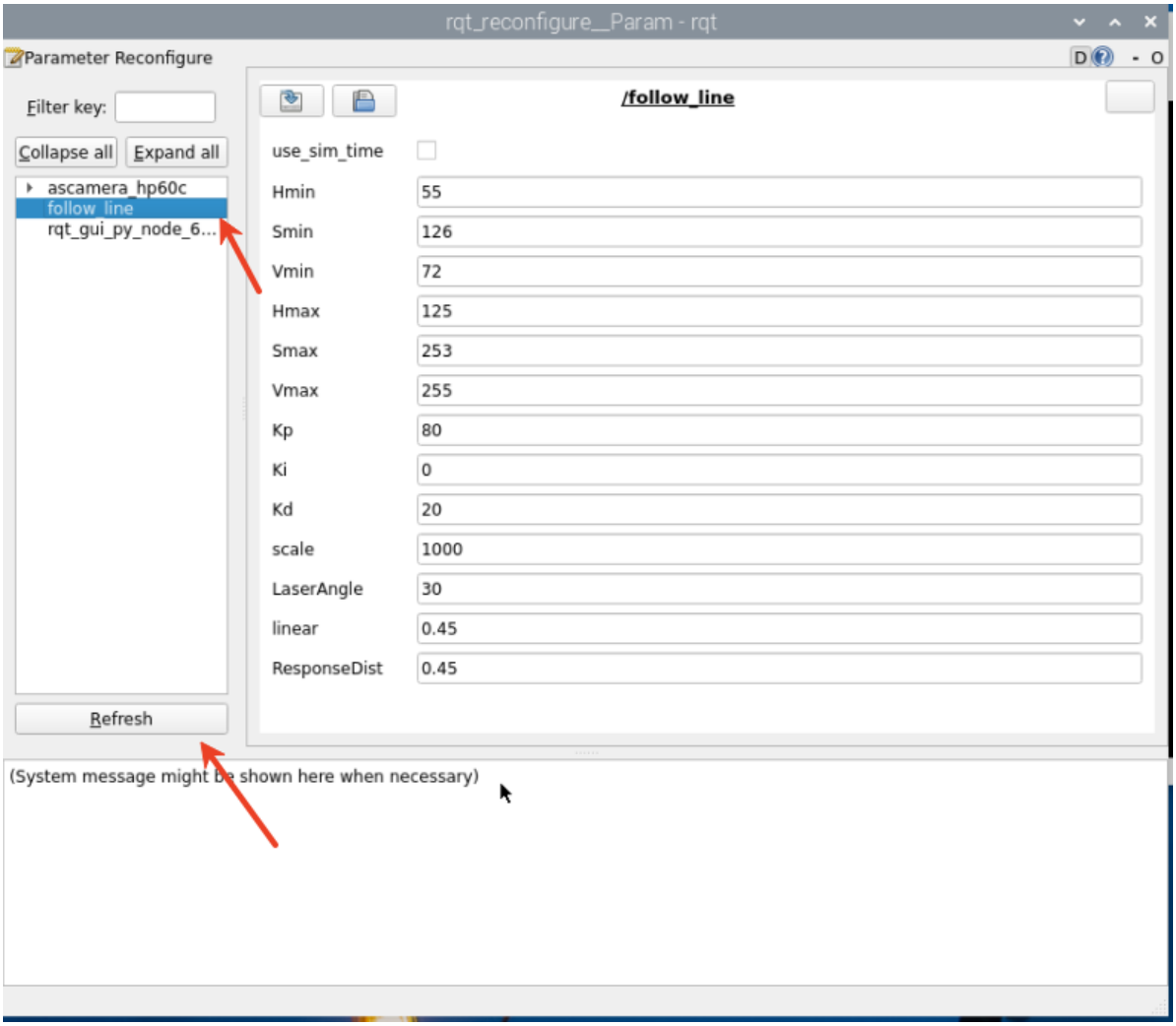


Then, press the `Spacebar` key to start calculating the speed, and the car will automatically patrol the line. If an obstacle is encountered during line patrol, the system will stop and beep.

3.2 Dynamic Parameter Adjustment

Enter in the terminal:

```
ros2 run rqt_reconfigure rqt_reconfigure
```



☑ After modifying the parameters, click a blank area in the GUI to write the parameter value. Note that this will only take effect for the current boot. To permanently effect the parameter, you need to modify it in the source code.

Parameter Analysis:

[Hmin], [Smin], [Vmin]: Indicates the lowest HSV value

[Hmax], [Smax], [Vmax]: Indicates the highest HSV value

[Kp], [KI], [KD]: Three parameters of the PID that adjust the turning angle

[scale]: PID control scale factor

[LaserAngle]: Radar detection angle

[linear]: Linear velocity

[ResponseDist]: Obstacle avoidance detection distance

4. Core Code

4.1. follow_line.py

This program has the following main functions:

- Calculates the offset between the center coordinates of the patrol line and the center of the image;
- Calculates the angular velocity based on the coordinate offset;
- Publishes the speed to drive the car;

- Subscribes to radar topic data and calculates obstacle distances

This program has the following main functions:

Calculates the center coordinates;

```
#计算hsv值 #Calculate HSV value
rgb_img, self.hsv_range = self.color.Roi_hsv(rgb_img, self.Roi_init)
#计算self.circle，计算出X的坐标、半径值。半径值为0说明没有检测到线，则发布停车信息
#Calculate self.circle and determine the coordinates and radius value of X. A
radius value of 0 indicates that no line was detected, and parking information
will be published
rgb_img, binary, self.circle = self.color.line_follow(rgb_img, self.hsv_range)
```

计算出角速度的值，

```
#320是中心点的X坐标的值，通过得到的图像的X值与320的偏差，可以计算出“我现在距离中心有多远”，然后
计算角速度的值
#320 is the value of the X coordinate of the center point. By calculating the
deviation between the X value of the obtained image and 320, we can determine
"how far am I from the center now" and then calculate the value of angular
velocity
[z_Pid, _] = self.PID_controller.update([(point_x - 320)*1.0/16, 0])
```