Color Tracking

5.1、Program function description

After the program is started, the image released by the camera is displayed on the web page. Use the mouse to select a color, the car will lock this color, and press the [Space] key to enter the tracking mode. The car will keep a distance of 1 meter from the tracked object and always ensure that the tracked object remains in the center of the screen.

In addition, the [L1] key of the handle can lock/open the motion control of the car. When the motion control is turned on, the function will be locked; when the motion control is locked, the function can be turned on.

5.2、Program code reference path

```
#launch file
/home/sunrise/yahboomcar_ws/src/yahboomcar_colortrack/launch/colortrack_launch.xml
```

The node or launch file that is launched is explained as follows,

- color_identify: mainly completes image processing and calculates the center coordinates of the tracked object. The code path is,

```
/home/sunrise/yahboomcar_ws/src/yahboomcar_colortrack/yahboomcar_colortrack/color_identify.py
```

- color_track: mainly calculates the speed based on the center coordinates and depth information of the tracked object, and publishes the speed data to the chassis. The code path is,

```
/home/sunrise/yahboomcar_ws/src/yahboomcar_colortrack/yahboomcar_colortrack/color_track.py
```

- astra.launch.xml: Start Astra camera
- yahboomcar_bringup_launch.py: Start chassis

- rosbridge_websocket_launch.xml: Start websocket to realize the interaction between ROS and Web
- rosboard_node: Start ROSboard

5.3, Program startup

5.3.1, Program startup and web page display

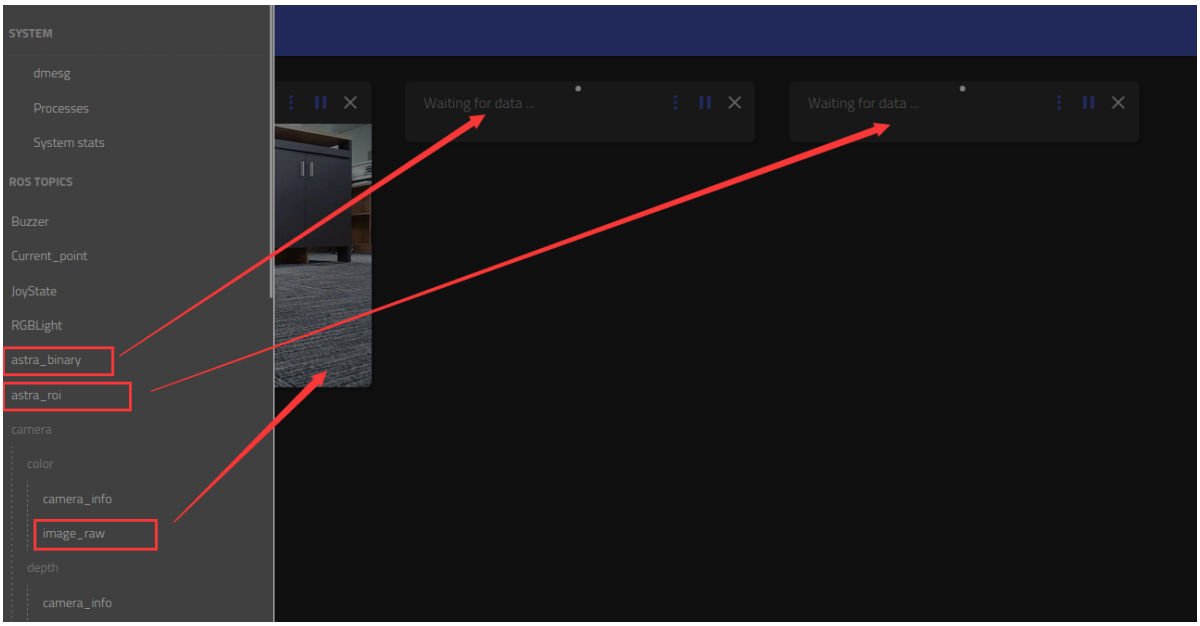
After SSH connects to the car, enter in the terminal,

```
ros2 launch yahboomcar_colortrack colortrack_launch.xml
```

Open the browser on the PC (note that the computer and the Sunrise Network must be in the same LAN), enter the URL: car IP:9999, for example, my car IP is 192.168.2.84, enter the URL in the browser on the virtual machine to open the ROSboard web page:

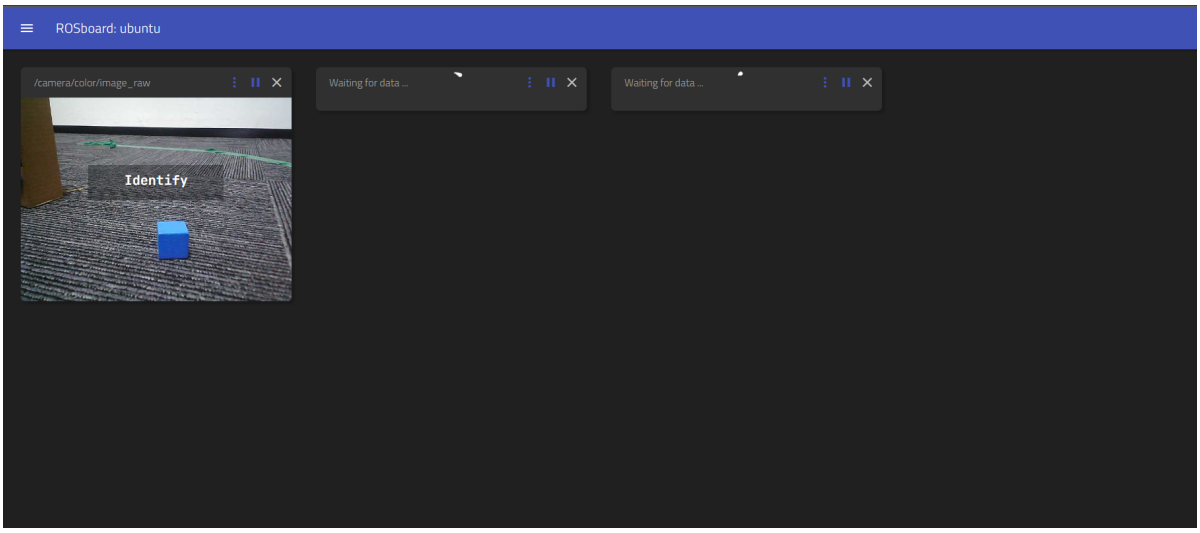
```
192.168.2.84:9999
```

Select the three topics /camera/color/image_raw, /astra_binary, /astra_roi respectively. At this time, only the RGB image released by the camera is displayed, and no data is released in the /astra_binary and /astra_roi topics.

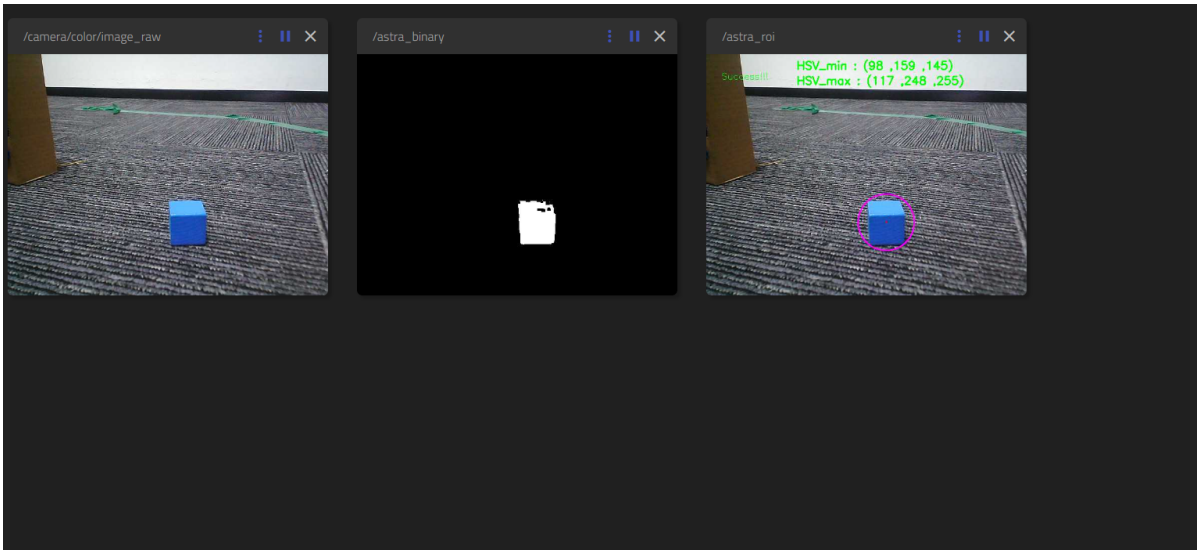


5.3.2, Color Identification and Tracking

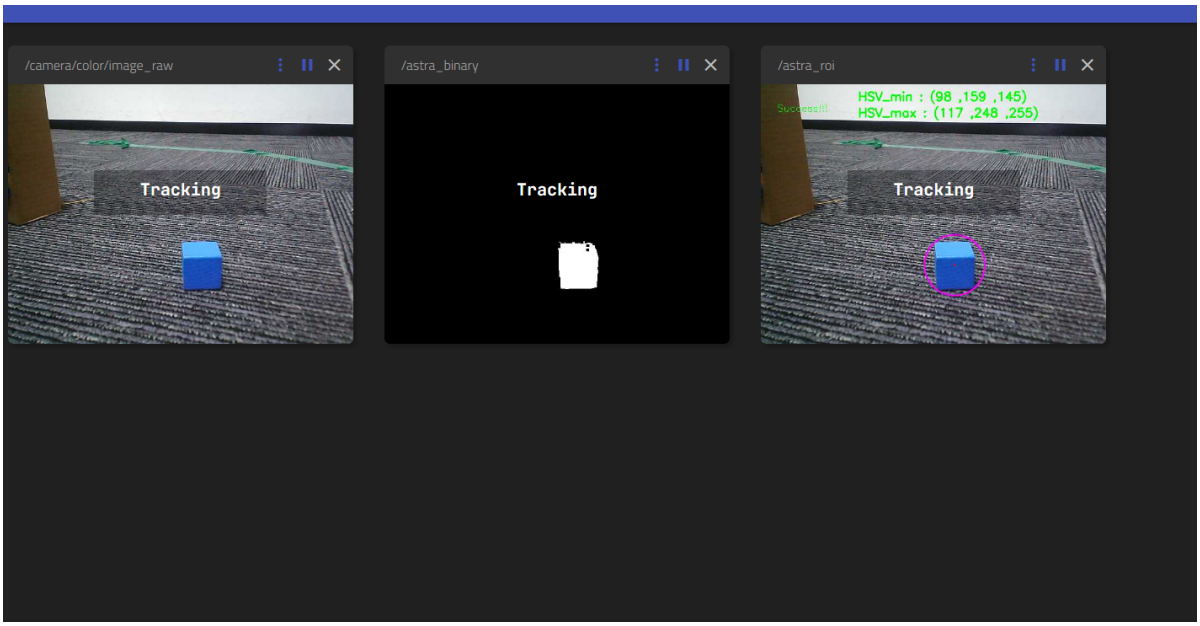
Press the [i] or [I] key in the page to enter the Identify mode, click the mouse twice in the RGB image to select an area (the area can only have one color),



After selecting, the following screen will appear, /astra_binary displays the binary image after color selection, /ast ra_roi displays the HSV value range of the color and the center point position of the color area.



Press the [Space] key on the page to enter the tracking mode, and the car will move to a position 1 meter away from the target and stop; press the [q] key on the page to cancel the tracking; press the [r] key on the page to clear the color selection, and then press [i] to reselect the color.



Move the object slowly, and the car will follow and keep a distance of 1 meter.

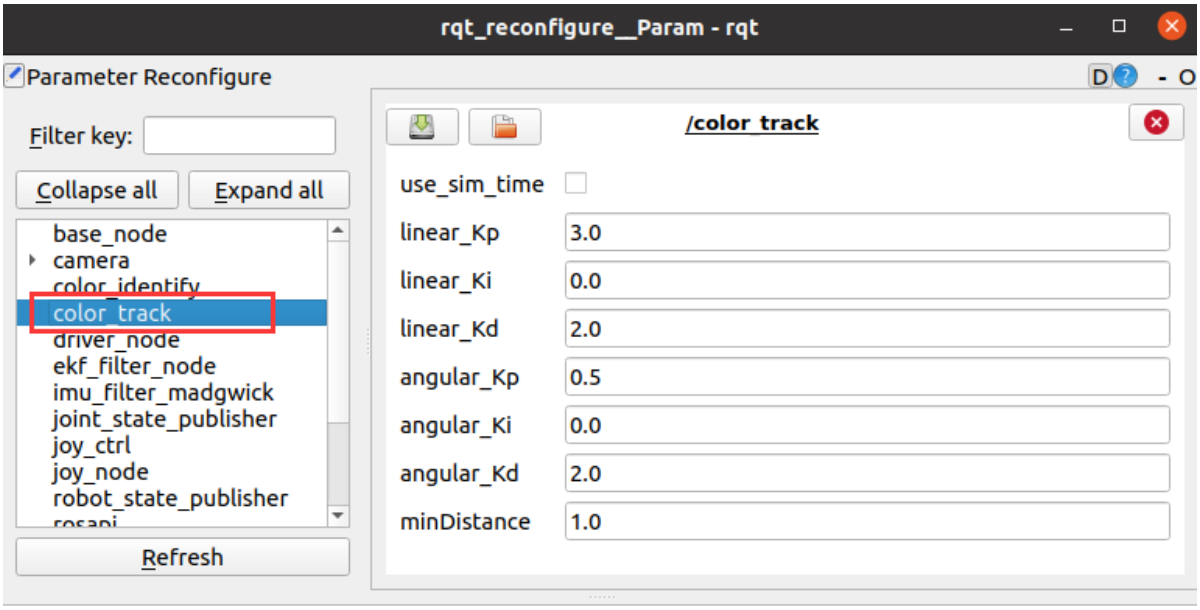
The interaction between this function and the keyboard is as follows:

Button	Function	Remarks
【r】or 【R】	Color recognition	Color selection mode, the program enters this mode by default after startup
【Space】	Color tracking	The car tracks to a position 1 meter away from the target
【q】or 【Q】	Stop movement	Stop movement and retain the HSV range of color
【i】or 【I】	Reset	Stop movement and clear the HSV range of color

5.3.3, Dynamic parameter adjustment

Virtual machine terminal input,

```
ros2 run rqt_reconfigure rqt_reconfigure
```



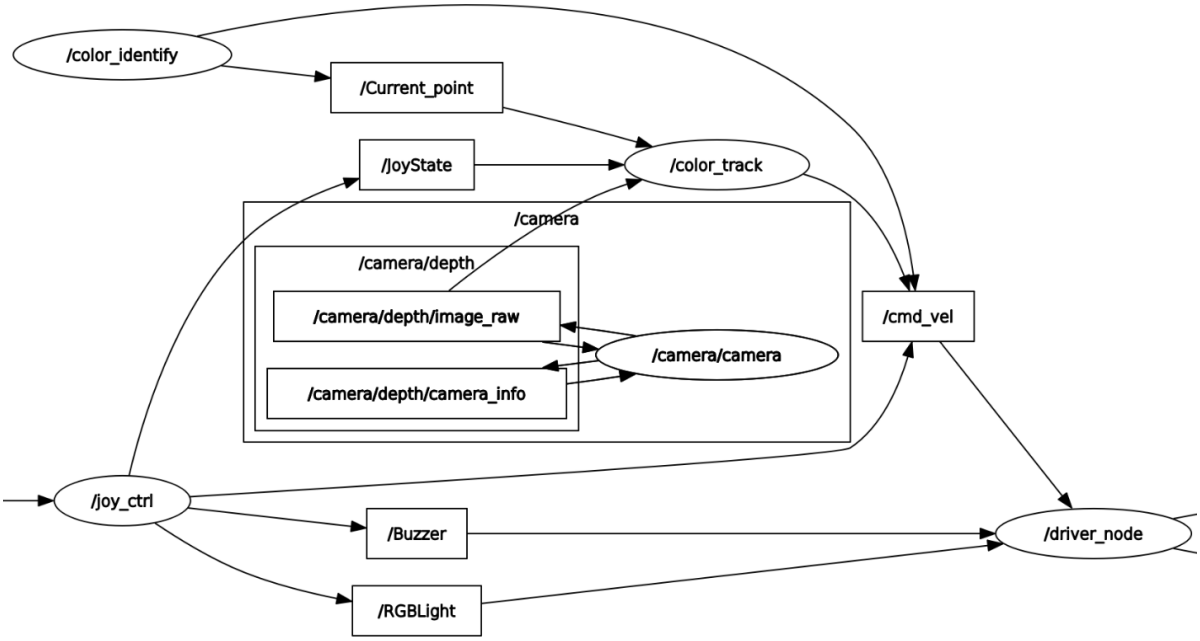
The adjustable parameters include the car's linear velocity, angular velocity PID and tracking distance.

After modifying the parameters, press the Enter key or click the blank space in the GUI to write the parameter value.

5.3.4, View the node topic communication graph

Open the virtual machine terminal and use the following command to view the topic communication between nodes,

```
ros2 run rqt_graph rqt_graph
```



5.4, Core source code analysis

5.4.1, color_identify.py

This program has the following functions:

- Open the camera and get the RGB image; in order to ensure the speed, the opencv method of reading the camera is adopted **(note that the device number is video8)**, instead of receiving data through the topic. However, after reading, the image data will be published through the topic for visualization on the web page.
- Receive keyboard and mouse topic data for switching modes and picking colors.
- Process the image to calculate the center coordinates of the tracking target and publish it.

Some core codes are as follows,

```
#Create a publisher to publish the center coordinates of the tracked object
self.pub_position = self.create_publisher(Position,"/Current_point", 10)
#Create a publisher to publish image data
self.pub_binary = self.create_publisher(Image, '/astra_binary', 10)
self.pub_roi = self.create_publisher(Image, '/astra_roi', 10)
self.pub_image = self.create_publisher(Image, '/camera/color/image_raw', 10)
#Create a subscriber to receive mouse and keyboard data
self.sub_mouse = self.create_subscription(Polygon, '/mouse_pos',
self.MouseCallback, 1)
self.sub_keyboard = self.create_subscription(Char, '/key_value',
self.KeyCallback, 1)
#Mouse event callback function, determine the selection position
def MouseCallback(self, msg):
start_cols = min(int(msg.points[0].x), int(msg.points[1].x))
start_rows = min(int(msg.points[0].y), int(msg.points[1].y))
end_cols = max(int(msg.points[0].x), int(msg.points[1].x))
end_rows = max(int(msg.points[0].y), int(msg.points[1].y))
self.Roi_init = (start_cols, start_rows, end_cols, end_rows)
#Keyboard event callback function, switch mode
def KeyCallback(self, key):
data = key.data
```

```
if data == 32: self.Track_state = 'tracking'
elif data == ord('i') or data == ord('I'): self.Track_state = 'identify'
elif data == ord('r') or data == ord('R'): self.Track_state = 'reset'
elif data == ord('q') or data == ord('Q'): self.Track_state = 'cancle'
#Calculate the value of the center coordinates, self.circle stores the xy values
rgbFrame, binary, self.circle = self.color.object_follow(rgbFrame,
self.hsv_range)
#Publish the message of the center coordinates
def execute(self, x, y, z):
position = Position()
position.anglex = x * 1.0
position.angley = y * 1.0
position.distance = z * 1.0
self.pub_position.publish(position)
```

5.4.2, color_track.py

This program has the following functions: receiving /Current_point and depth image topic data, calculating speed, and then publishing speed data.

Some codes are as follows,

```
#Define subscribers to receive required topic data
self.sub_depth =
self.create_subscription(Image,"/camera/depth/image_raw",self.depth_img_Callback,
1)
self.sub_position =
self.create_subscription(Position,"/Current_point",self.positionCallback,1)
#Define velocity publisher
self.pub_cmdVel = self.create_publisher(Twist,'/cmd_vel',10)
#Two important callback functions, get point_x value and dist value
def PositionCallback(self, msg):
def DepthCallback(self, msg):
#Calculate linear velocity and angular velocity according to point_x and dist and
publish
linear_x = self.linear_pid.compute(dist, self.minDist) / 1000.0
angular_z = self.angular_pid.compute(320, point_x) / 1000.0 if abs(dist -
self.minDist) < 30: linear_x = 0 if abs(point_x - 320.0) < 30: angular_z = 0
twist = Twist() twist.linear.x = linear_x * 1.0 twist.angular.z = angular_z * 1.0
self.pub_cmdVel.publish(twist) ```
```