5. Voice control trolley color tracking

Before running the program, you need to bind the port number of the voice board and the port number of the ROS expansion board in the host computer, you can refer to the previous chapter for binding; when you enter the docker container, you need to mount this voice board to be able to recognize the voice board in the docker container.

1. Program function description

After the program starts, say "Hello, Xiaoya" to the module, and the module replies "Yes" to wake up the voice board, and then you can say to it to start tracking any one of the red/green/blue/yellow colors. After receiving the command, the program recognizes the color and loads the processed image screen, then press the R2 key of the joystick to start the tracking program, the car will track the recognized color, and when the recognized color moves slowly, the car will also track the movement.

2. Program code reference path

After entering the docker container, the source code of this function is located at.

```
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl/yahboomcar_voic
e_ctrl/Voice_Ctrl_colorTracker.py
/root/yahboomcar_ros2_ws/yahboomcar_ws/src/yahboomcar_voice_ctrl/yahboomcar_voic
e_ctrl/Voice_Ctrl_colorHSV.py
```

3. Program startup

3.1. Startup command

After entering the docker container, depending on the actual model and camera, terminal input, the

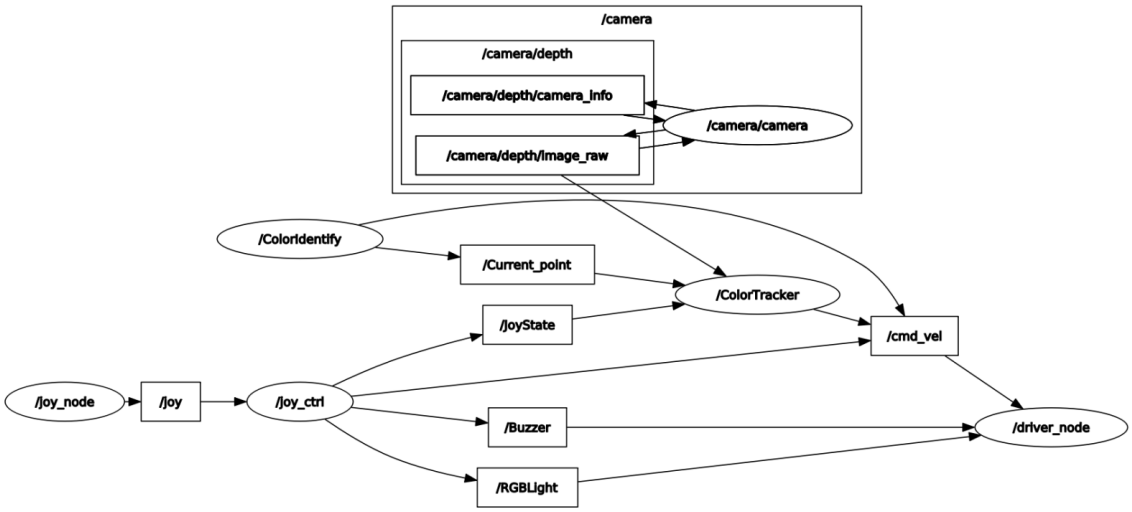
```
#启动手柄控制节点 #Start the joystick control node
ros2 run yahboomcar_ctrl yahboom_joy_R2
ros2 run joy joy_node
#启动语音控制颜色追踪节点
#Start voice control color tracking nodes
ros2 run yahboomcar_voice_ctrl Voice_Ctrl_colorHSV
ros2 run yahboomcar_voice_ctrl Voice_Ctrl_colorTracker
#启动小车底盘 #Start the trolley chassis
ros2 run yahboomcar_bringup Ackman_driver_R2
#启动深度相机，获取深度图像
#Start the depth camera and get a depth image
ros2 launch astra_camera astra.launch.xml
```

Take tracking red as an example, after waking up the module, say "start tracking red" to it, press the R2 key on the remote control, the program receives the instruction, starts to process the image, then calculates the center coordinates of the red object, releases the center coordinates of the object; combined with the depth information provided by the depth camera, calculates the speed, and finally releases the speed to drive the trolley.

3.2. Node Topic Communication Map

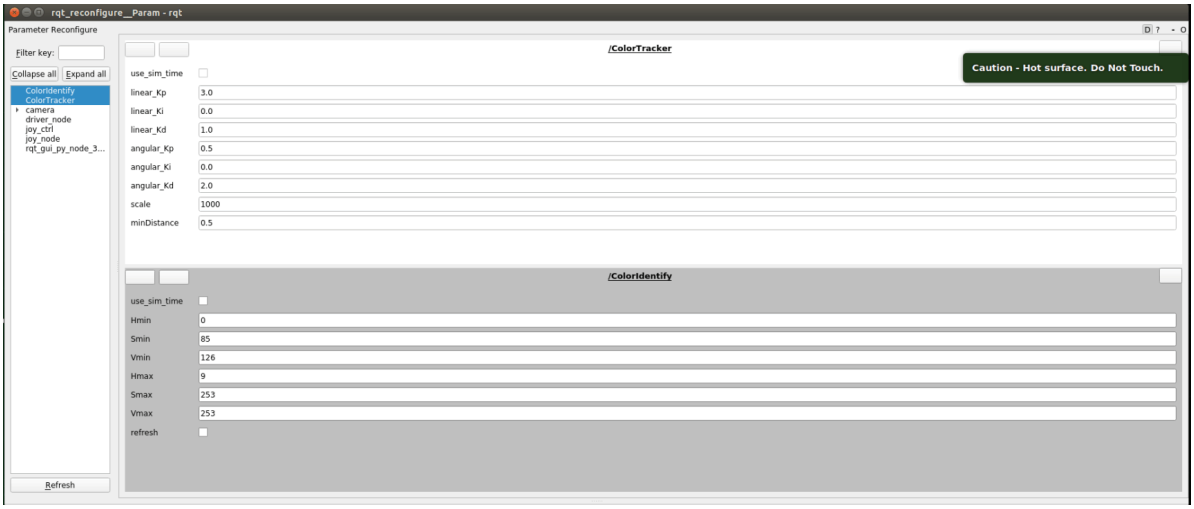
docker terminal input.

```
ros2 run rqt_graph rqt_graph
```



There are also dynamic parameter conditioners, modification of parameters, docker terminal input, the

```
ros2 run rqt_reconfigure rqt_reconfigure
```



After modifying the parameters, click on the GUI blank to write the parameter values. From the above figure, we can see that

- colorHSV is mainly responsible for image processing and can adjust the value of HSV;
- colorTracker is mainly responsible for calculating the speed, and the speed and distance related parameters can be adjusted.

The meaning of each parameter is shown in the following table.

colorHSV

Parameter name	Parameter Meaning
Hmin	H minimum
Smin	S minimum
Vmin	V minimum
Hmax	H maximum
Smax	S maximum
Vmax	V maximum
refresh	Refresh data into the program

colorTracker

Parameter name	Parameter Meaning
linear_Kp	Linear velocity P value
linear_Ki	Linear velocity i value
linear_Kd	Linear velocity d value
angular_Kp	Angular velocity P value
angular_Ki	Angular velocity i value
angular_Kd	Angular velocity d value
scale	scale factor
minDistance	tracking distance

4. Core code

4.1. colorHSV

This part mainly analyzes voice commands, as well as image processing, and finally releases the center coordinates,

```
#定义一个发布者，发布检测物体的中心坐标
# Define a publisher that publishes the center coordinates of the detected
object
self.pub_position = self.create_publisher(Position,"/Current_point", 10)
#导入语音驱动库 # Import the voice driver library
from Speech_Lib import Speech
#创建语音控制对象 # Create voice control objects
self.spe = Speech()
#以下是获取指令，然后对识别结果进行判断，加载相对应的HSV的值
# The following is to get the instruction, and then judge the identification
result, load the corresponding HSV value
command_result = self.spe.speech_read()
self.spe.void_write(command_result)
if command_result == 73 :
self.model = "color_follow_line"
print("tracker red")
self.hsv_range =  [(0, 175, 149), (180, 253, 255)]
#处理图像，计算检测物体的中心坐标，进入execute函数，发布中心坐标
# Process the image, calculate the central coordinate of the detected object,
enter the execute function, and publish the central coordinate
rgb_img, binary, self.circle = self.color.object_follow(rgb_img, self.hsv_range)
if self.circle[2] != 0: threading.Thread(
                target=self.execute, args=(self.circle[0], self.circle[1],
self.circle[2])).start()
if self.point_pose[0] != 0 and self.point_pose[1] != 0: threading.Thread(
                target=self.execute, args=(self.point_pose[0],
self.point_pose[1], self.point_pose[2])).start()
```

4.2. colorTracker

This part receives the topic data and depth data of the center coordinates, and then calculates the speed and publishes it to the chassis,

```
#定义一个订阅者，订阅深度信息
# Define a subscriber and subscribe to in-depth information
self.sub_depth = self.create_subscription(Image,"/camera/depth/image_raw",
self.depth_img_Callback, 1)
#定义一个订阅者，订阅中心坐标信息
# Define a subscriber, subscribe center coordinates information
self.sub_position =
self.create_subscription(Position,"/Current_point",self.positionCallback,1)
#回调者函数 # Callback function
def positionCallback(self, msg)#获取中心坐标值          # Get the center coordinate
value
def depth_img_Callback(self, msg) #获取深度信息
# Get in-depth information
#传入中心坐标的X值以及深度信息，计算出速度发布给底盘
# Pass in the X-value of the center coordinate and the depth information,
calculate the speed and publish to the chassis
def execute(self, point_x, dist)
```

Combined with the node communication diagram in 3.2, understanding the source code will be clearer and clearer.