4、Voice controlled car color tracking

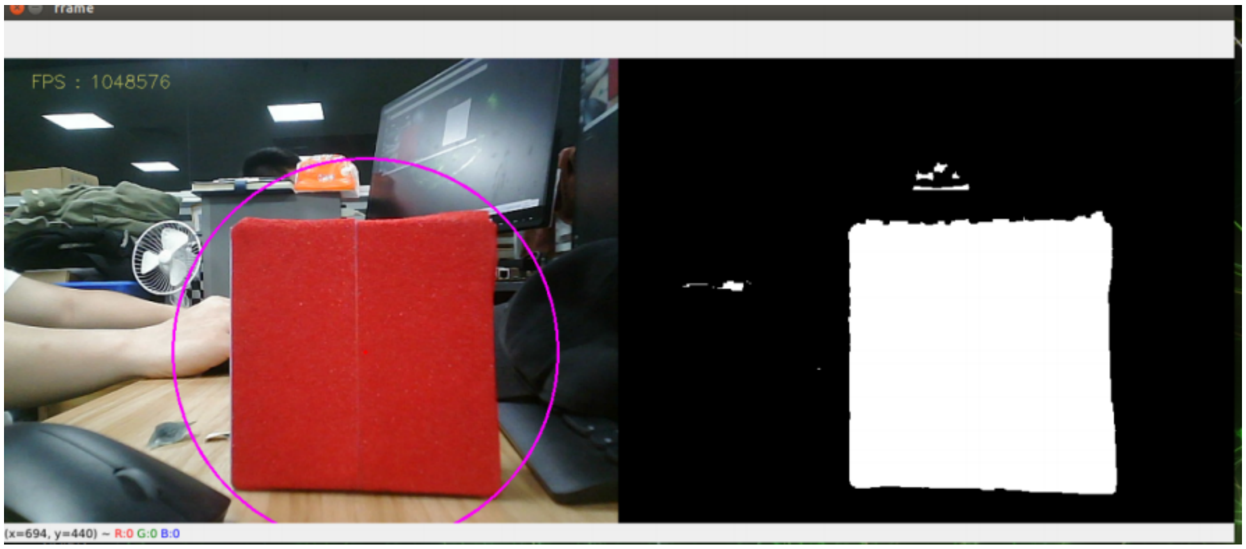
This course needs to be combined with the hardware of the Rosmaster-X3 car, and only code analysis will be done here. Firstly, let's take a look at the built-in voice commands，

functional word	Speech recognition results	Voice broadcast content
yellow following	72	OK, I found the yellow
red following	73	OK, I found the red
green following	74	OK, I found the green
follow this color	75	OK, I found this color
stop following	76	OK, it has been stoped

1、Run sample program

Terminal input，

```
#Start handle control node
ros2 run yahboomcar_ctrl yahboom_joy_X3
ros2 run joy joy_node
#Activate voice control color tracking node
ros2 run yahboomcar_voice_ctrl Voice_Ctrl_colorHSV
ros2 run yahboomcar_voice_ctrl Voice_Ctrl_colorTracker
#Start the trolley chassis
ros2 run yahboomcar_voice_ctrl Voice_Ctrl_Mcnamu_driver_X3
#Start the depth camera to obtain depth images
ros2 launch astra_camera astra.launch.xml
```



Taking tracking red as an example, after waking up the module, say "red following" to it, press the R2 button on the remote control, and the program receives the command to start processing the image. Then, calculate the center coordinates of the red object and publish the center coordinates of the object. Combined with the depth information provided by the depth camera, calculate the speed, and finally publish it to drive the car

2、Core code

code path：~/driver_ws/src/yahboomcar_voice_ctrl/yahboomcar_voice_ctrl/Voice_Ctrl_colorHSV.py

This section mainly analyzes voice commands, image processing, and finally releases the center coordinates,

```
#Define a publisher to publish the center coordinates of the detected object
self.pub_position = self.create_publisher(Position,"/Current_point", 10）
#Import voice driver library
from Speech_Lib import Speech
#Create voice control objects
self.spe = Speech()
#The following is to obtain instructions, then judge the recognition results, and
load the corresponding HSV values
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)]
#Process the image, calculate the center coordinates of the detected object, enter
the execute function, and publish the center coordinates
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()
```

code path：~/driver_ws/src/yahboomcar_voice_ctrl/yahboomcar_voice_ctrl/Voice_Ctrl_colorTracker.py

This section receives topic data and depth data from the center coordinate, calculates the speed, and publishes it to the chassis,

```
#Define a subscriber to subscribe to deep information
self.sub_depth = self.create_subscription(Image,"/camera/depth/image_raw",
self.depth_img_Callback, 1)
#Define a subscriber to subscribe to central coordinate information
self.sub_position =
self.create_subscription(Position,"/Current_point",self.positionCallback,1)
#Callback function
def positionCallback(self, msg)#Obtain the center coordinate value
def depth_img_Callback(self, msg) #Obtain deep information
#Pass in the X value and depth information of the center coordinate, calculate the
speed, and publish it to the chassis
def execute(self, point_x, dist)
```