Color recognition

Note: The virtual machine and ROS wifi image transmission module need to have consistent ROS-DOMAIN-ID, which must be set to 20

Before running the experiment, please ensure that the ROS wifi image transmission module has correctly enabled the proxy on the virtual machine (Linux with humbleROS2 system)

1、Program Function Description

Identify multiple colors at any time and autonomously store the currently recognized colors. This feature can recognize the colors we need in complex environments.

2、Program code reference path

```
~/yahboomcar_ws/src/yahboom_esp32ai_car/yahboom_esp32ai_car/colorHSV.py
```

- colorHSV.py

  The main task is to complete image processing and calculate the center coordinates of the tracked object.

3、Program startup

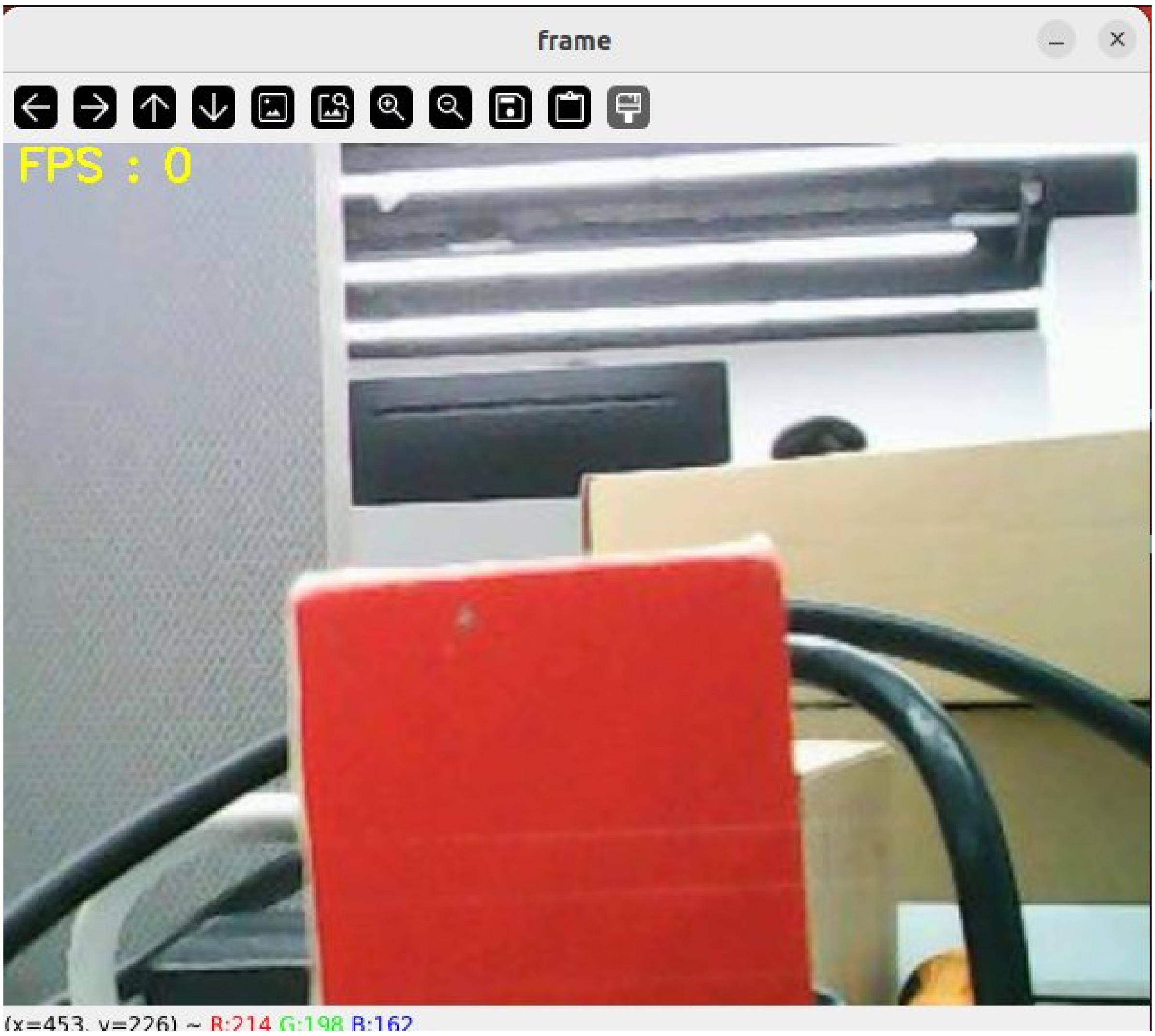
3.1、Start command

Terminal input,

```
#Start color recognition program
ros2 run yahboom_esp32ai_car colorHSV
```

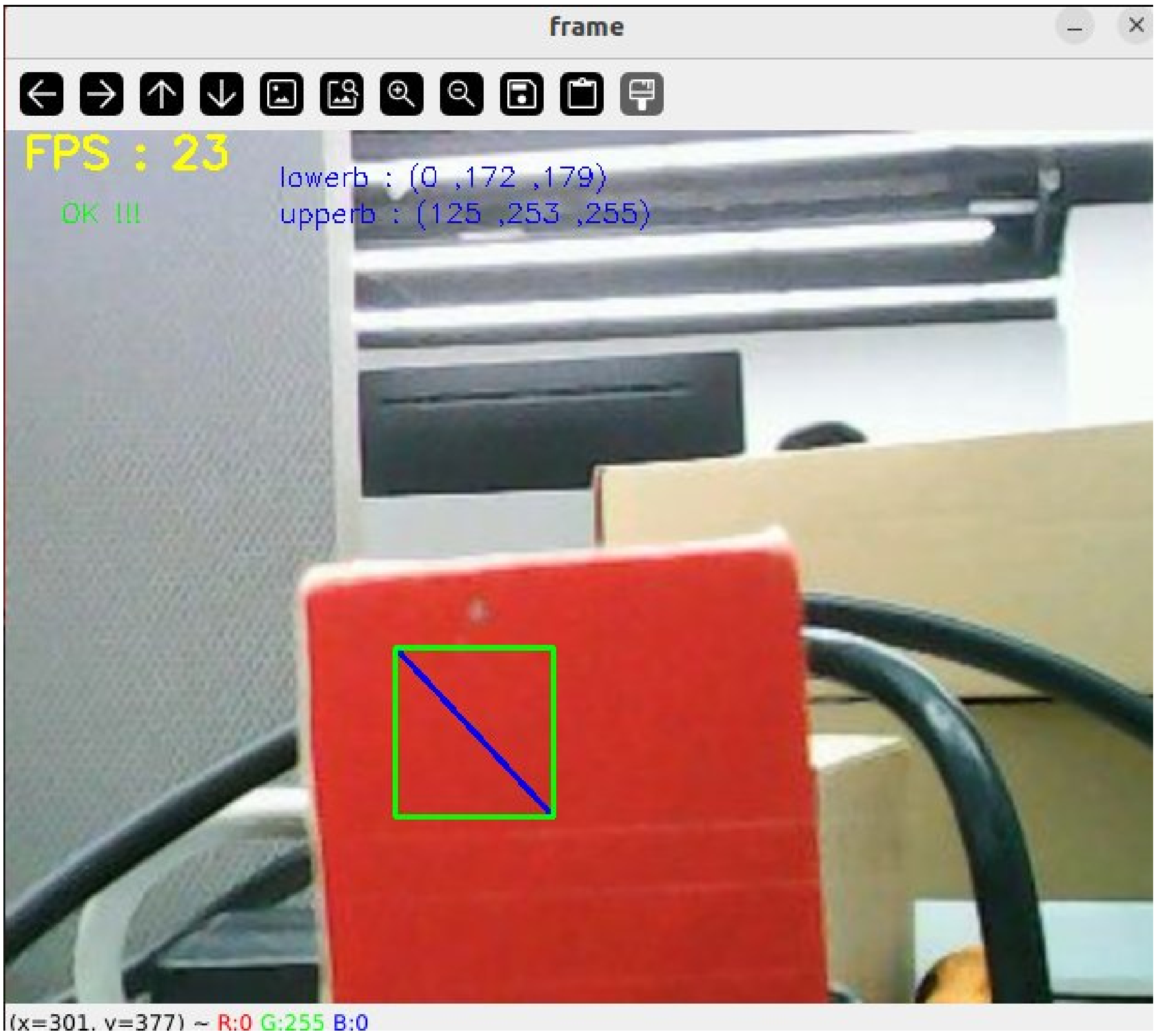
If the image of the camera is inverted, you need to see 3. Camera image change** and correct it yourself in the document. This experiment will not be explained further.

Taking tracking red as an example, after the program starts, the following screen will appear,

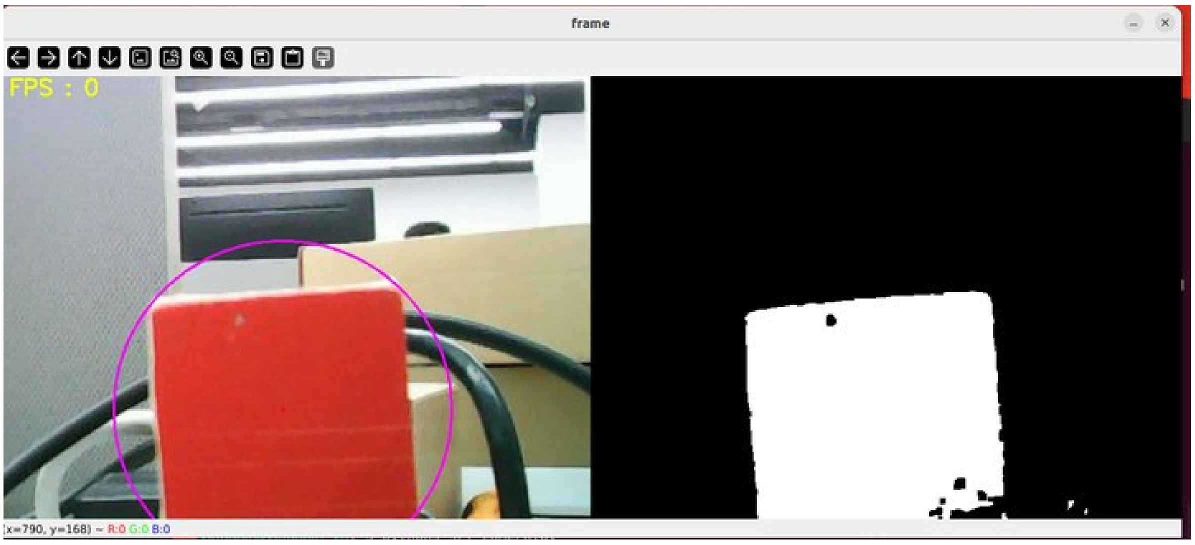


(x=453, y=226) ~ R:214 G:198 B:162

Then press the r/R key on the keyboard to enter color selection mode, and use the mouse to frame an area (which can only have one color),



After selection, the effect is shown in the following figure,



4、Core code

4.1、colorHSV.py

This program mainly has the following functions:

- Turn on the camera to capture the image;
- Obtain keyboard and mouse events for switching modes and color selection;

Part of the core code is as follows,

```
 if action == 32: self.Track_state = 'tracking'
        elif action == ord('i') or action == ord('I'): self.Track_state = 
"identify"
        elif action == ord('r') or action == ord('R'): self.Reset()
        elif action == ord('q') or action == ord('Q'): self.cancel()
        if self.Track_state == 'init':
            cv.namedWindow(self.windows_name, cv.WINDOW_AUTOSIZE)
            cv.setMouseCallback(self.windows_name, self.onMouse, 0)
            if self.select_flags == True:
                cv.line(rgb_img, self.cols, self.rows, (255, 0, 0), 2)
                cv.rectangle(rgb_img, self.cols, self.rows, (0, 255, 0), 2)
                if self.Roi_init[0] != self.Roi_init[2] and self.Roi_init[1] != 
self.Roi_init[3]:
                    rgb_img, self.hsv_range = self.color.Roi_hsv(rgb_img, 
self.Roi_init)
                    self.gTracker_state = True
                    self.dyn_update = True
                else: self.Track_state = 'init'

class MY_Picture(Node):
    def __init__(self, name):
        super().__init__(name)
        self.bridge = CvBridge()
        self.sub_img = self.create_subscription(
            CompressedImage, '/espRos/esp32camera', self.handleTopic, 1)

        self.color_identify = Color_Identify("ColorIdentify")
        self.last_stamp = None
        self.new_seconds = 0
        self.fps_seconds = 1


    def handleTopic(self, msg):
        self.last_stamp = msg.header.stamp
        if self.last_stamp:
            total_secs = Time(nanoseconds=self.last_stamp.nanosec, 
seconds=self.last_stamp.sec).nanoseconds
            delta = datetime.timedelta(seconds=total_secs * 1e-9)
            seconds = delta.total_seconds()*100

            if self.new_seconds != 0:
                self.fps_seconds = seconds - self.new_seconds

            self.new_seconds = seconds
        start = time.time()
        frame = self.bridge.compressed_imgmsg_to_cv2(msg)
        frame = cv.resize(frame, (640, 480))

        action = cv.waitKey(10) & 0xFF
        frame, binary =self.color_identify.process(frame, action)
        if len(binary) != 0: cv.imshow('frame', ManyImgs(1, ([frame, binary])))
        else:cv.imshow('frame', frame)
        msg = self.bridge.cv2_to_imgmsg(frame, "bgr8")
        self.color_identify.pub_img.publish(msg)

        end = time.time()
        fps = 1/((end - start)+self.fps_seconds)

        text = "FPS : " + str(int(fps))

        cv.putText(frame, text, (10,20), cv.FONT_HERSHEY_SIMPLEX, 0.8,
(0,255,255), 2)
        if len(binary) != 0:
            cv.imshow('frame', ManyImgs(1, ([frame, binary])))
        else:
            cv.imshow('frame', frame)

        if action == ord('q') or action == 113:
            cv.destroyAllWindows()
```