Visual tracking

Visual tracking
1. Introduction
2. Code analysis
1.3 Steps

1. Introduction

Dogzilla's visual line patrol function can switch and recognize multiple colors, draw the recognized line as the smallest square, and then calculate the center point of the square. According to the coordinates of the center point, if the center point deviates from the visual center, it will combine with the PID algorithm to calculate whether the robot dog should go left or right, so as to achieve the function of visual line patrol.

2. Code analysis

Press the R key to initialize the posture of the robot dog, set the footstep frequency as slow, the body height as 90, and the camera looks down and forward.

```
# 初始化机器狗  init DOGZILLA
def dog_init(self):
    self.dog.stop()
    time.sleep(.01)
    self.dog.pace("slow")
    time.sleep(.01)
    self.dog.translation('z', 90)
    time.sleep(.01)
    self.dog.attitude('p', 10)
```

Use the mouse to draw a rectangle in the color to be recognized. The drawn rectangle must be an object of the same color. The system will automatically analyze the color HSV value in the rectangle box and record the linefollowhsv.text file saved in the same directory.

```
# 鼠标操作 mouse action
def onMouse(self, event, x, y, flags, param):
    if event == 1:
        self.Track_state = 'init'
        self.select_flags = True
        self.Mouse_XY = (x,y)
    if event == 4:
        self.select_flags = False
        self.Track_state = 'mouse'
    if self.select_flags == True:
        self.cols = min(self.Mouse_XY[0], x), min(self.Mouse_XY[1], y)
        self.rows = max(self.Mouse_XY[0], x), max(self.Mouse_XY[1], y)
        self.Roi_init = (self.cols[0], self.cols[1], self.rows[0], self.rows[1])
```

Program processing content: press the space bar to start the line patrol function, the robot dog moves forward, and the recognized result calls the execute function for processing.

```python
# 程序处理 program processing
def process(self, rgb_img, action):
    binary = []
    rgb_img = cv.resize(rgb_img, (640, 480))

    # rgb_img = cv.flip(rgb_img, 1)
    if action == 32:
        self.Track_state = 'tracking'
        self.dog.forward(50)
    elif action == ord('i') or action == 105: self.Track_state = "identify"
    elif action == ord('r') or action == 114: self.Reset()

    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.dyn_update = True
            else: self.Track_state = 'init'
    elif self.Track_state == "identify":
        if os.path.exists(self.hsv_text): self.hsv_range = read_HSV(self.hsv_text)
        else: self.Track_state = 'init'
    if self.Track_state != 'init' and len(self.hsv_range) != 0:
        rgb_img, binary, self.circle = self.color.line_follow(rgb_img, self.hsv_range)
        if self.dyn_update == True :
            write_HSV(self.hsv_text, self.hsv_range)
            self.dyn_update = False
    if self.Track_state == 'tracking':
        if len(self.circle) != 0:
            threading.Thread(target=self.execute, args=(self.circle[0], self.circle[1], self.circle[2])).start()
    return rgb_img, binary
```

In the execute function, the PID algorithm is used to calculate the rotation direction of the robot dog, so that the robot dog can move along the line.

```python
# 执行命令 executive command
def execute(self, point_x, point_y, radius):
    [z_Pid, _] = self.PID_controller.update([(point_x - 320), 0])
    print("point_x:%d, point_y:%d, radius:%d, z_Pid:%d" % (point_x, point_y, radius, int(z_Pid)))
    self.dog.turn(int(z_Pid))
```

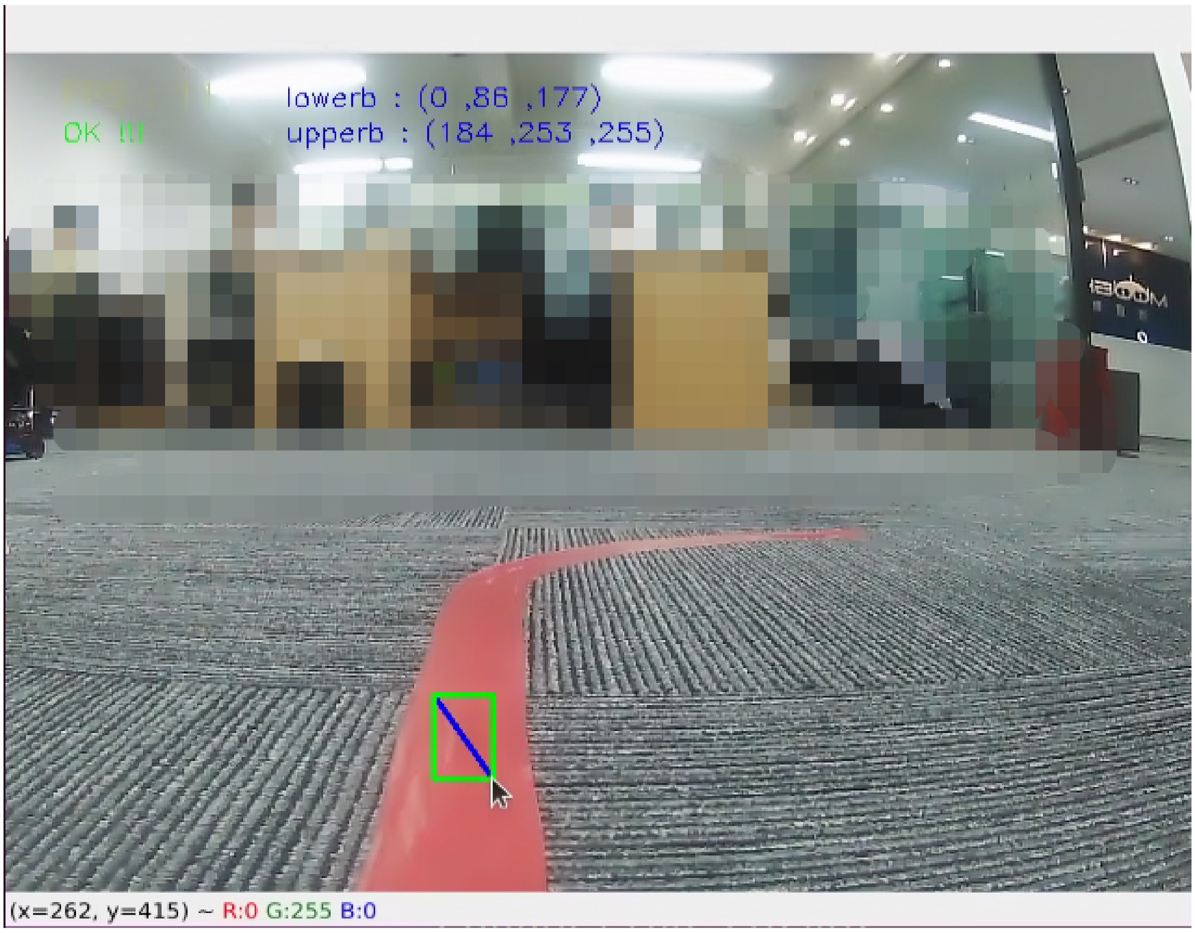
1.3 Steps

Note: since the desktop window needs to be opened for visual display, you need to log in to the raspberry pie desktop with remote VNC before performing the following:

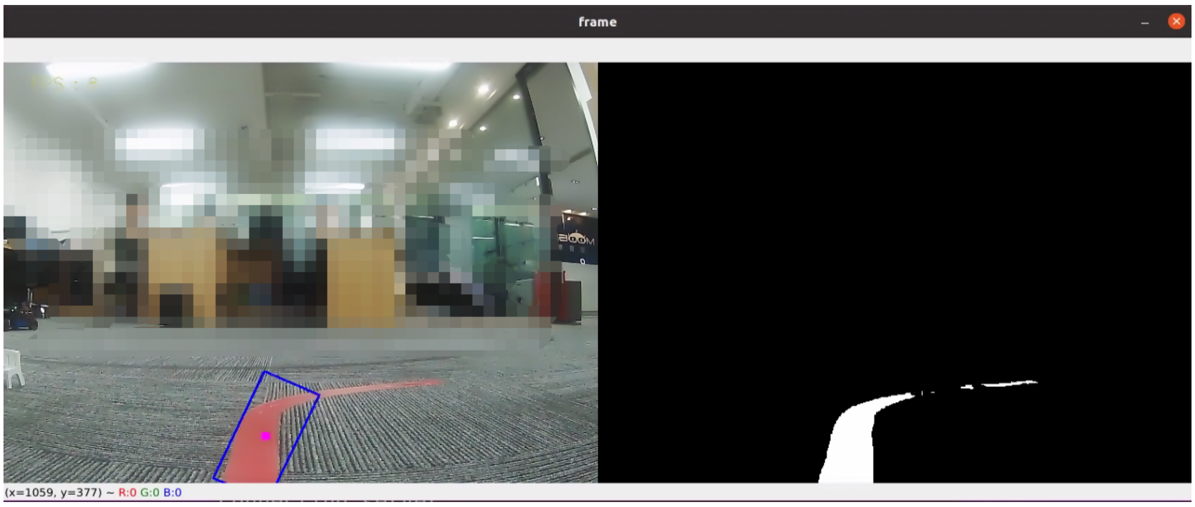
Open the system terminal, then enter the visual line patrol game directory, and then run the program.

```
cd ~/DOGZILLA/Samples/3_AI_Visual/11_12.followline/
python3 follow_line.py
```

When the system is opened for the first time, only one camera screen window will be displayed. Put the robot dog on the line of color to be inspected, press the R key of the keyboard to put the robot dog into the ready state, and then draw a small rectangle on the line to be inspected with the mouse.



After release, the system will add another window to display the processing effect, and set the recognized color to white and other colors to black.



At this time, press the space bar once, and the robot dog starts to walk along a straight line.

If it stops, press the R key to reset, and then press the I key to recall the processing result image.

If you need to leave, press Q to exit the program.

Note: all keyboard inputs can only be input when the camera display window is active.