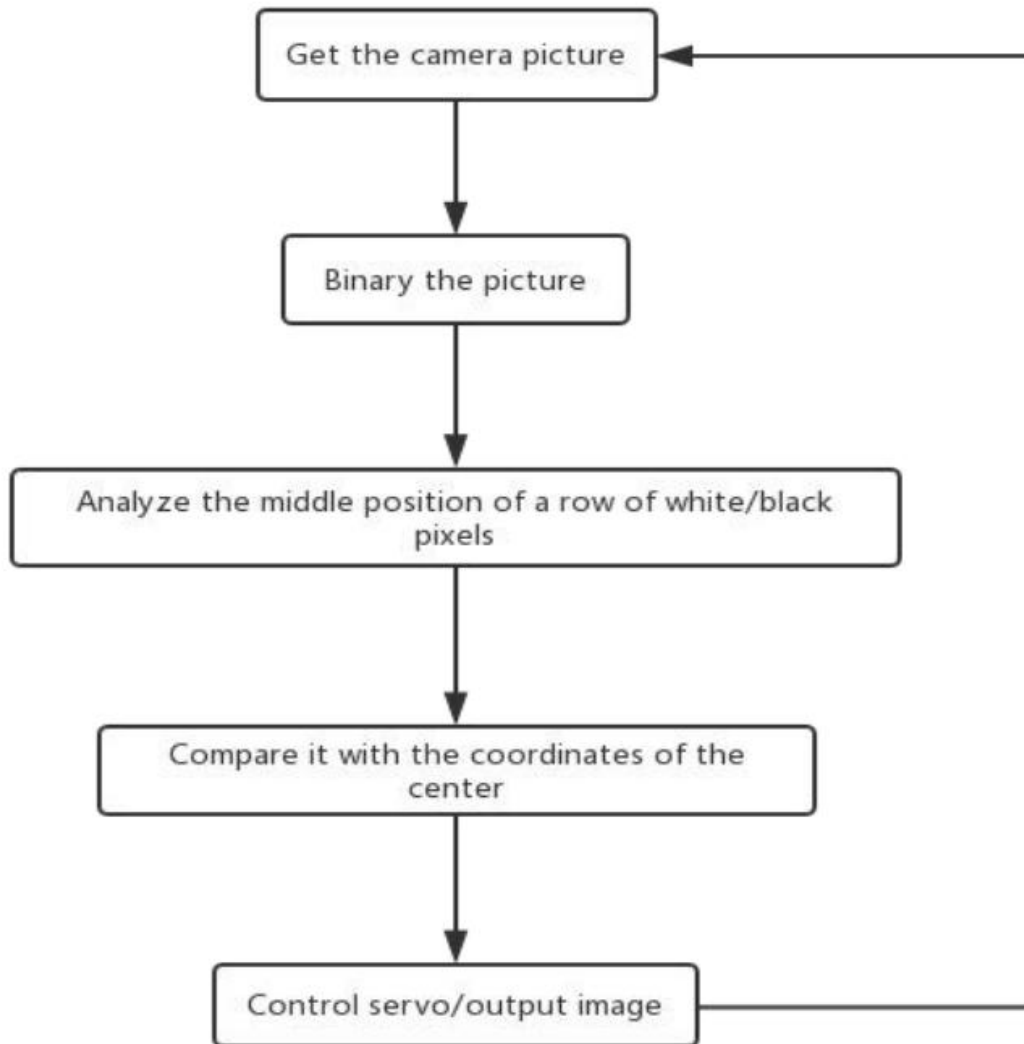# Lesson 5 How to Use the Video Tracking line Function

The WEB interface includes control buttons for video Tracking Line inspection. This article introduces how to use the function.

5.1 Introduction the Video Tracking line

1. Video line inspection is to capture the picture in real time through the camera, and Raspberry Pi performs binarization processing on each frame of the video picture. At this time, all video images have only two colors, black and white. In the video frame, the pixels that exceed the set threshold are turned into black pixels, and those that are lower than the set threshold are turned into white pixels.

2. Analyze all the black/white pixels in the selected area of the video frame to find the center of the black area/white area.

3. Calculate the distance between the center position of the black area/white area and the border.

4. Set the motion state of RaspTank according to the distance, and output the processed video frames.

Get the camera picture

Binary the picture

Analyze the middle position of a row of white/black pixels

Compare it with the coordinates of the center

Control servo/output image

5.2 Introduction the CVFL Control Interface

Control the switch of visual tracking line function.

CVFL Control

L1 ——————————————————●——————   START

L2 ——————————————————●——————   COLOR

SP —●——————————————————————————
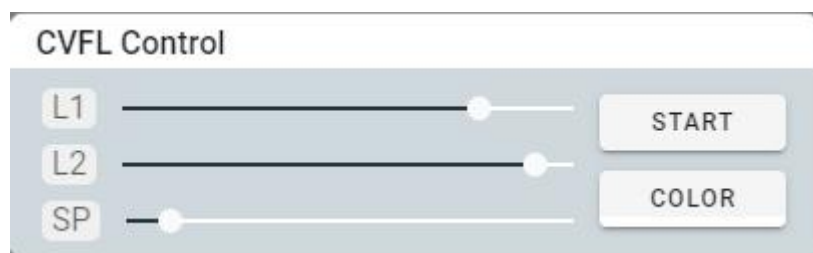
- **L1, L2:** The two parameters of L1 and L2 are used to set the height of the two auxiliary lines. Robot products will only recognize the areas in the screen surrounded by two auxiliary lines.

- **SP:** Invalid value. No adjustment is required.

- **COLOR:** Switch to search for white lines on black or black lines on white.

- **START:** Switch video tracking line function. The first click starts the tracking line function, and the second click closes the tracking line function.

5.3 Start using the video Tracking line function

Make a simple video line inspection environment for testing the video line inspection function. Example: Paste a piece of black tape on A4 paper. We used tape about 1.5cm wide.

Please follow the previous tutorial to ensure that the WEB interface can communicate with Banana Pi normally. The real-time images captured by RaspTank can be seen in the WEB interface.

1. On the WEB interface, you need to click the "COLOR" button once to change the background color of the button to black. (If the background color is black by default, click twice)
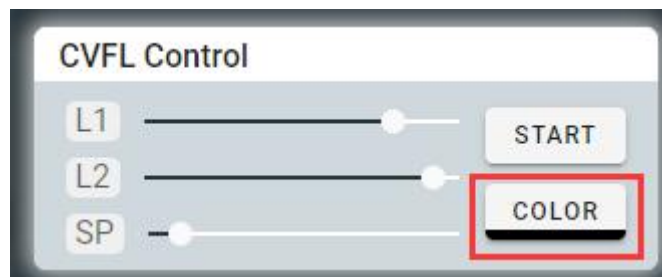
2. Please hang the wheels of the RaspTank car in the air during the test to prevent the car from moving. And place the A4 paper in front of the trolley.

3. Click "START", now there are only black and white in the video screen. And "Following Block Line" is displayed.
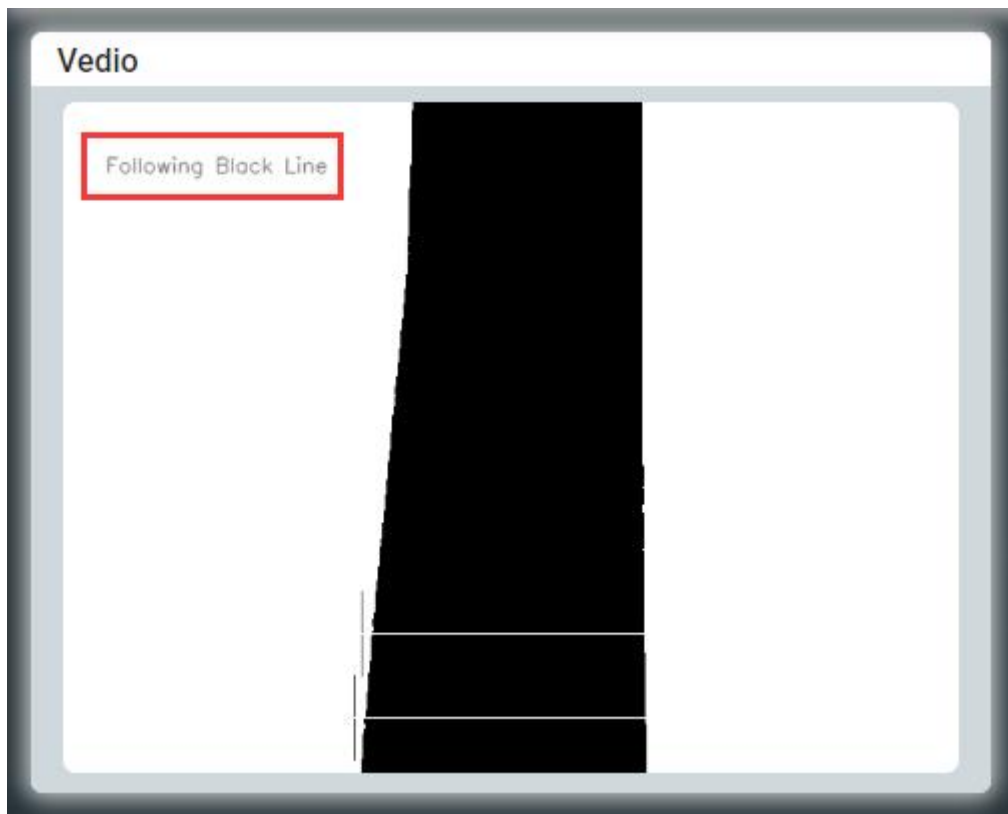
4. You can see that there are two "T"-shaped lines on the screen. These two T-lines can be adjusted by the sliders L1 and L2. The area between the two lines is used as the video line tracking area. For example: slide L1, the T-line will move to another position.
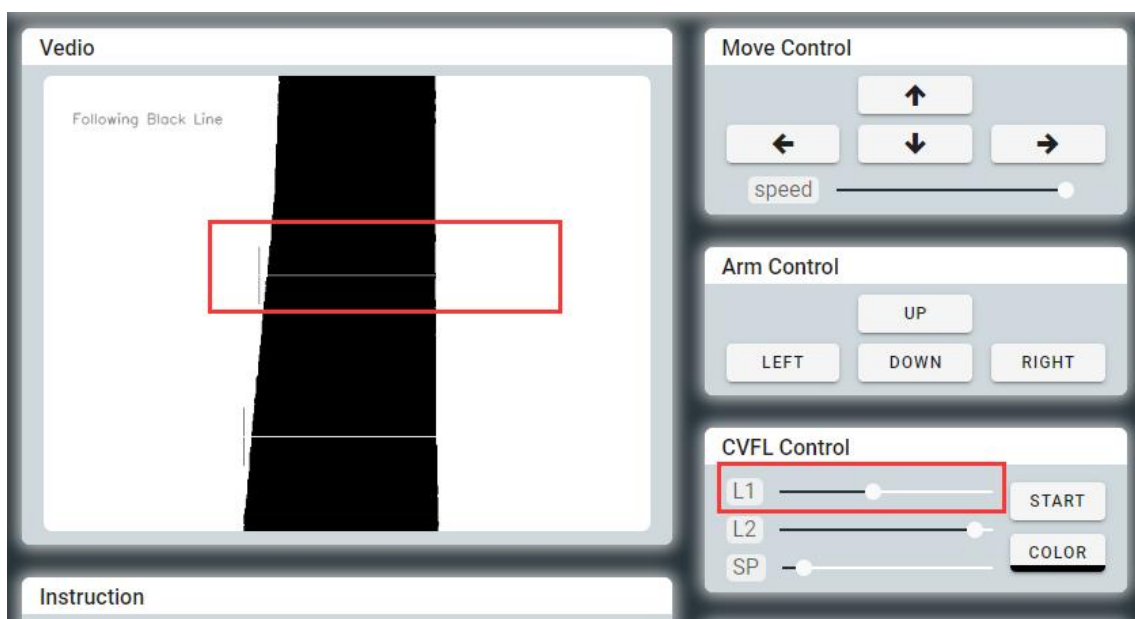
5. Move the A4 paper and observe the movement state of the left and right wheels of the trolley.

- When the black line is on the left side, the right wheel of the trolley turns, and the left wheel does not move or turns slowly, so that the trolley moves forward to the left.

- When the black line is on the right side, the left wheel of the trolley turns, and the right wheel does not move or turns slowly, so that the trolley moves forward to the right.

- When the black line is in the middle, the car moves forward.

6. Click the "START" button again, and the video line inspection function will stop.

5.4 The main code program

Please see the complete code in camera_opencv.py.

Binarize video frames

```python
1.    def elementDraw(self,imgInput):
2.
3.      if self.CVMode == 'findlineCV':
4.
5.        CVThread.scGear.moveAngle(1, -45) # The camera looks down.
6.        if frameRender:
7.          imgInput = cv2.cvtColor(imgInput, cv2.COLOR_BGR2GRAY)
8.          """"
9.          Image binarization, the method of processing functions can be searched for "threshold" in the link: http://docs.opencv.org/3.0.0/examples.html
10.         '''
11.         # retval_bw, imgInput =  cv2.threshold(imgInput, 0, 255, cv2.THRESH_OTSU) # THRESH_OTSU:Adaptive Threshold (Dynamic Threshold).flag, use Otsu algorithm to choose the threshold value.
12.         retval_bw, imgInput =  cv2.threshold(imgInput, 80, 255, cv2.THRESH_BINARY) # Set the threshold manually and set it to 80.
13.         imgInput = cv2.dilate(imgInput, None, iterations=2) # dilate
14.         imgInput = cv2.erode(imgInput, None, iterations=2) #  erode
15.       try:
16.         if lineColorSet == 255:
```

```
17.        cv2.putText(imgInput,('Following White Line'),(30,50), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(128,255,128),1,c
    v2.LINE_AA)
18.    else:
19.        cv2.putText(imgInput,('Following Black Line'),(30,50), cv2.FONT_HERSHEY_SIMPLEX, 0.5,(128,255,128),1,cv
    2.LINE_AA)
20.
21.    cv2.line(imgInput,(self.left_Pos1,(linePos_1+30)),(self.left_Pos1,(linePos_1-30)),(255,128,64),1)
22.    cv2.line(imgInput,(self.right_Pos1,(linePos_1+30)),(self.right_Pos1,(linePos_1-30)),(64,128,255),)
23.    cv2.line(imgInput,(0,linePos_1),(640,linePos_1),(255,255,64),1)
24.
25.    cv2.line(imgInput,(self.left_Pos2,(linePos_2+30)),(self.left_Pos2,(linePos_2-30)),(255,128,64),1)
26.    cv2.line(imgInput,(self.right_Pos2,(linePos_2+30)),(self.right_Pos2,(linePos_2-30)),(64,128,255),1)
27.    cv2.line(imgInput,(0,linePos_2),(640,linePos_2),(255,255,64),1)
28.
29.    cv2.line(imgInput,((self.center-20),int((linePos_1+linePos_2)/2)),((self.center+20),int((linePos_1+linePos_2)/2
    )),(0,0,0),1)
30.    cv2.line(imgInput,((self.center),int((linePos_1+linePos_2)/2+20)),((self.center),int((linePos_1+linePos_2)/2-20
    )),(0,0,0),1)
31.    except:
32.        pass
33.    return imgInput
```

Control RaspTank movement according to the calculated structure

```
1.    def findLineCtrl(self, posInput):
2.      global findLineMove
3.      # posInput == center
4.      if posInput != None and findLineMove == 1:
5.        if posInput > 480: # The position of the center of the black line in the screen (value range: 0-640)
6.          #turnRight
7.          if CVRun:
8.                      move.move(turn_speed, 'no', 'right', 0.2) # 'no'/'right':turn  Right,  turn_speed ：
    left wheel speed, 0.2:turn_speed*0.2 = right wheel speed
9.          else:
10.            move.move(turn_speed, 'no', 'no', 0) # stop
11.
12.        elif posInput < 180: # turnLeft.
13.          if CVRun:
14.            move.move(turn_speed, 'no', 'left', 0.2) # 'no'/'left':turn left.
15.          else:
16.            move.move(turn_speed, 'no', 'no', 0) # 'no'/'no'： stop.
```

```
17.
18.        else:
19.          if CVRun:
20.            move.move(forward_speed, 'forward', 'no', 0.2)# 'forward'/'no': forward.
21.          else:
22.            move.move(forward_speed, 'no', 'no', 0.2) # stop
23.            pass
24.      else: # Tracking color not found.
25.        move.move(80, 'no', 'no', 0.5) # stop.
```

Calculate the center position of the black area/black area

```
1.    def findlineCV(self, frame_image):
2.      global findLineMove
3.      frame_findline = cv2.cvtColor(frame_image, cv2.COLOR_BGR2GRAY)
4.      retval, frame_findline =  cv2.threshold(frame_findline, 0, 255, cv2.THRESH_OTSU)
5.      frame_findline = cv2.erode(frame_findline, None, iterations=6)
6.      colorPos_1 = frame_findline[linePos_1]
7.      colorPos_2 = frame_findline[linePos_2]
8.      try:
9.        lineColorCount_Pos1 = np.sum(colorPos_1 == lineColorSet)
10.       lineColorCount_Pos2 = np.sum(colorPos_2 == lineColorSet)
11.
12.       lineIndex_Pos1 = np.where(colorPos_1 == lineColorSet)
13.       lineIndex_Pos2 = np.where(colorPos_2 == lineColorSet)
14.       # print("lineIndex_Pos1: %s" %lineIndex_Pos1[0])
15.       # print("lineIndex_Pos2: %s" %lineIndex_Pos2[0])
16.
17.       # Roughly judge whether there is a color to track.
18.       if lineIndex_Pos1 !=[]:
19.         if abs(lineIndex_Pos1[0][-1] - lineIndex_Pos1[0][0]) > 500:
20.           print("Tracking color not found")
21.           findLineMove = 0    # No tracking color, stop moving
22.         else:
23.           findLineMove = 1
24.       elif lineIndex_Pos2!= []:
25.         if abs(lineIndex_Pos2[0][-1] - lineIndex_Pos2[0][0]) > 500:
26.           print("Tracking color not found")
27.           findLineMove = 0
28.         else:
29.           findLineMove = 1
30.
31.       if lineColorCount_Pos1 == 0:
```

```
32.        lineColorCount_Pos1 = 1
33.     if lineColorCount_Pos2 == 0:
34.        lineColorCount_Pos2 = 1
35.
36.     self.left_Pos1 = lineIndex_Pos1[0][lineColorCount_Pos1-1]
37.     self.right_Pos1 = lineIndex_Pos1[0][0]
38.     self.center_Pos1 = int((self.left_Pos1+self.right_Pos1)/2)
39.     # print("center_Pos1: %s" %self.center_Pos1)
40.
41.     self.left_Pos2 = lineIndex_Pos2[0][lineColorCount_Pos2-1]
42.     self.right_Pos2 = lineIndex_Pos2[0][0]
43.     self.center_Pos2 = int((self.left_Pos2+self.right_Pos2)/2)
44.     # print("center_Pos2: %s" %self.center_Pos2)
45.
46.     self.center = int((self.center_Pos1+self.center_Pos2)/2)
47.  except:
48.     self.center = None
49.     pass
50.  print("center: %s" %self.center)
51.  # self.findLineCtrl(self.center, 320)
52.  self.findLineCtrl(self.center)
53.  self.pause()
```