Lesson 6 How to Adjust the Video Tracking Line

Since the video tracking line is easily disturbed, the line tracking function cannot work normally, so the program needs to be fine-tuned according to the usage scenario. It will be mainly affected by the following factors:

- 1. The threshold value set by the program in the process of binarizing the video frame into a black and white picture is too close to the color value in the actual environment, and it is impossible to accurately distinguish the two colors. This problem can be improved by modifying the threshold in the program.
- 2. Due to the reflection of the ground (paper, black lines) and other reasons in the arranged video line inspection scene, the video cannot correctly judge the color of the reflective area, thus affecting the video line inspection. This problem can be avoided with improved line following scenes (making scenes with non-reflective materials). After turning on the tracking function, you can pick up the car and approach the ground, walk once according to the line tracking track, and observe whether there are incomplete black lines in the video screen on the WEB interface.
- 3. The trolley turns left and right through the speed difference of the two rear wheels. When turning, the turning radius is too large, causing the camera recognition area to easily exceed the line patrol range.

6.1 Preparation Before Adjustment

Before modifying, it is recommended to back up the original file.

1. To stop the automatic running program, enter in the Raspberry Pi command line:

```
sudo killall python3
```

```
pi@raspberrypi:~ $ sudo killall python3
python3: no process found
pi@raspberrypi:~ $
```

2. Enter the server.py folder.

```
cd adeept_rasptank2/web
```

```
pi@raspberrypi:~ $ cd adeept_picar-b2/web
pi@raspberrypi:~/adeept_picar-b2/web $
```

3. Backup camera_opencv.py.

```
sudo cp camera_opencv.py camera_opencv_bak.py
```

```
pi@raspberrypi:~/adeept_picar-b2/web $ sudo cp camera_opencv.py camera_opencv_bak.py
pi@raspberrypi:~/adeept_picar-b2/web $
```

4. Check the backup file.

```
ls
```

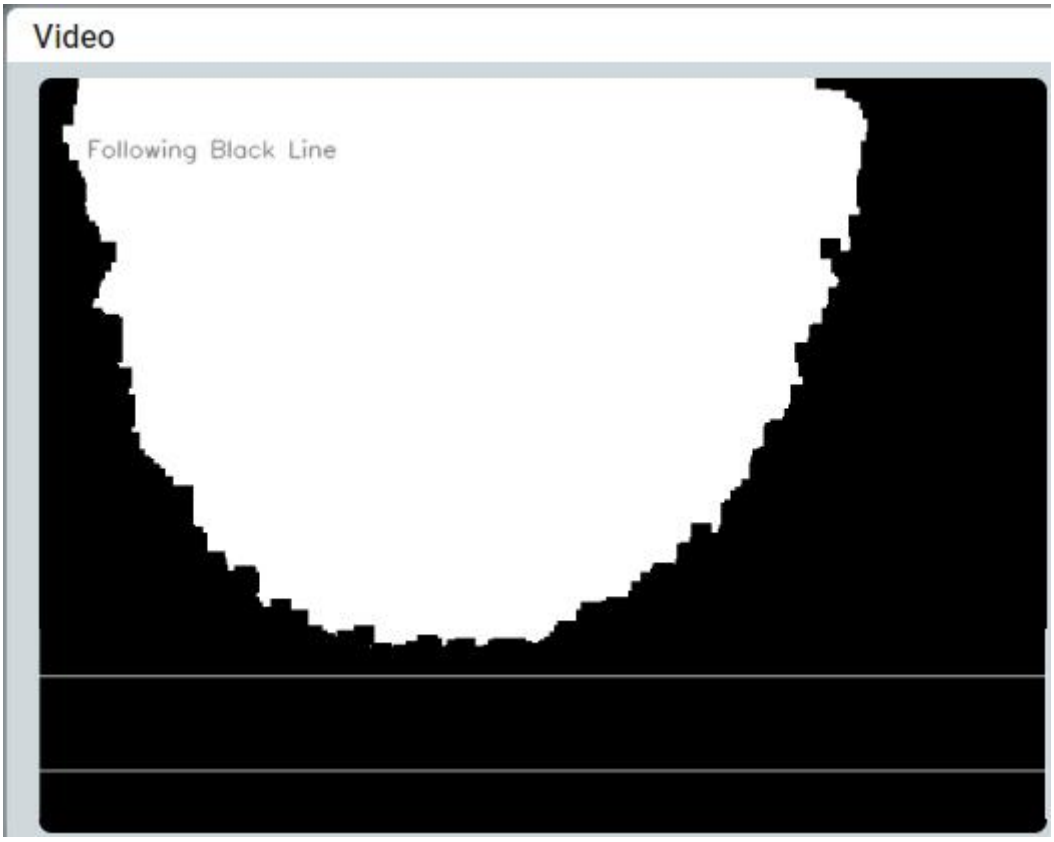
```
pi@raspberrypi:~/adeept_picar-b2/web $ ls
app.py              camera_opencv.py  info.py          PID.py        RPIMotor.py  ultra.py
base_camera.py      dist              Kalman_filter.py __pycache__   RPIservo.py  webServer.py
camera_opencv_bak.py functions.py     move.py          robotLight.py switch.py
pi@raspberrypi:~/adeept_picar-b2/web $
```

Please make sure that the car can judge the position of the black line according to the content of the previous lesson and that the car can make a correct steering judgment for the offset black line.

6.2 Threshold for Video Frame Conversion Judgment

There are usually two methods for the threshold required when a video frame is converted into a black and white image: automatically adjust the threshold and manually set a fixed threshold.

The advantage of automatically adjusting the threshold is that as long as the traced line and the background color are different, they can be distinguished by automatically adjusting the threshold. The disadvantage is that when the camera recognition area exceeds the tracking range, the program will also distinguish the background of the same color in more detail and convert it into a black and white picture, which may cause the car to move irregularly when it leaves the tracking area. As shown below. Therefore, it is not recommended to use the method of automatically adjusting the threshold.



Line 165 (or line 165) in the server/camera_opencv.py program is the automatic adjustment threshold (OTSU algorithm), which has been commented.

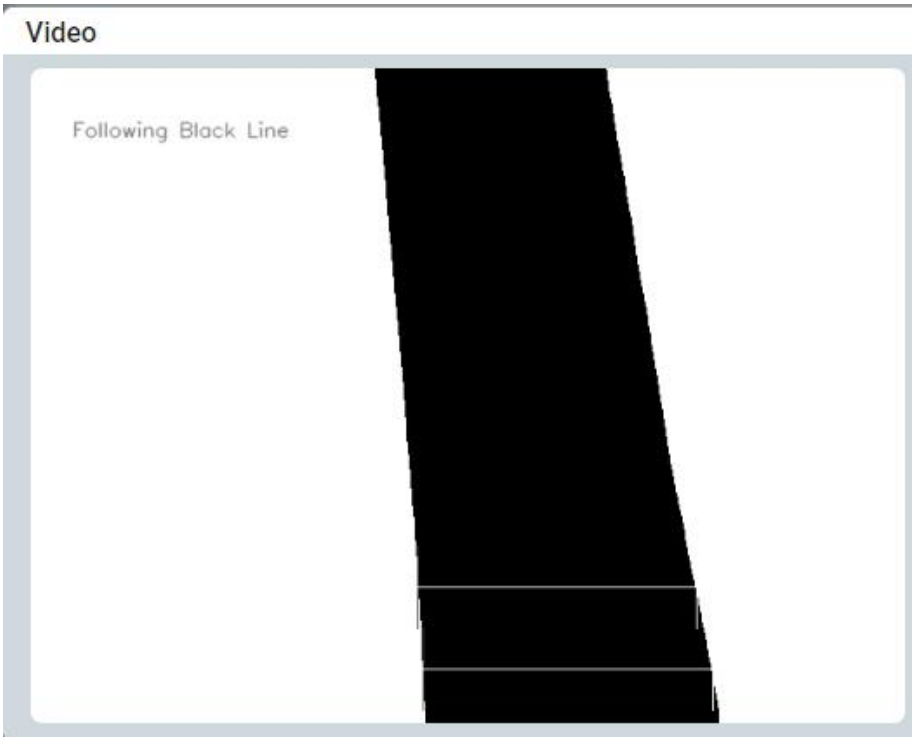
```
164     # retval_bw, imgInput =  cv2.threshold(imgInput, 0, 255, cv2.THRESH_OTSU) # THRESH_OTSU:Adaptive Threshold (Dynamic Threshold).flag, use Otsu algorithm to choose
165     retval_bw, imgInput =  cv2.threshold(imgInput, Threshold, 255, cv2.THRESH_BINARY) # Set the threshold manually and set it to 80.
166     # imgInput = cv2.erode(imgInput, None, iterations=6)
```

Line 165 (or line 165) is to manually set the judgment threshold for video frame binarization.

retval_bw, imgInput = cv2.threshold(imgInput, 80, 255, cv2.THRESH_BINARY)

Among them, "80" is the threshold value set manually. When the arranged scene cannot be correctly converted into a black and white picture (as shown in the figure below), you can try to adjust the threshold value set. The range of values is 0-255.

If there is a large difference between the background color and the color of the line, you can try to reduce the threshold "80", for example, change 80 to 60 or lower. The threshold can also be adjusted higher according to the situation.



6.3 Avoid Reflective Areas

If white gaps appear on black lines in a black and white image, or black areas appear on a white background, this may be due to reflective in parts of the scene. The line following scene shown in the figure below is an ideal state.

6.4 Adjust the Turning State of the Car

The picture captured by the camera during tracking is in front of the car, and the rotation center that actually controls the steering is on the car, resulting in a larger turning radius.

Due to the large turning radius, the line inspection area moves too fast when the car turns, and there is a slight delay when the video frame is binarized, which easily makes the line inspection

area exceed the line inspection range. Therefore, it is recommended that the turning radius of the black line be made larger. Making an S-shaped track with a larger radius will make it easier to achieve the tracking effect.

Adjusting the appropriate turning speed and straight-going speed can make the tracking function easier to realize.

Mainly modify the "findLineCtrl" function on line 199 in the adeept_picar-b/web/camera_opencv.py file.

```
1.  def findLineCtrl(self, posInput):
2.      global findLineMove
3.      # posInput == center
4.      if posInput != None and findLineMove == 1: # Determine whether the video tracking f
    unction can be performed
5.          if posInput > 480: # The position of the center of the black line in the screen (value r
    ange: 0-640)
6.              #turnRight
7.              if CVRun: # The default is 1, and it can be manually set to 0 under special circumst
    ances to stop the movement of the car.
8.                  move.move(turn_speed, 'no', 'right', 0.2) # 'no'/'right':turn Right, turn_speed：
    left wheel forward speed, 0.2:turn_speed*0.2 = right wheel forward speed
9.              else:
10.                 move.move(turn_speed, 'no', 'no', 0.2) # 'no'/'no': stop,The 4th parameter "0.2"
    is invalid
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', 0.2) # stop.
```

Among them, "turn_speed" is the speed when turning, and "forward_speed" is the straight speed, which can be modified in lines 31-32 in the camera_opencv.py file. Usually there is only one motor as the main driving motor when turning, so the set speed value will be higher than the straight speed value. The value range of "0.2" is 0-1, the smaller the value, the smaller the turning radius.

```
1.  # When turning, only one wheel pushes the car, so a value higher than forward_speed is required.
2.  turn_speed = 35 # Range of values: 0-100
3.  forward_speed = 20 # Avoid too fast, the video screen does not respond in time. Range of values: 0-100.
```

The larger the value, the greater the power of the motor and the faster the rotation speed. However, it will increase the turning speed of the car, which will easily cause the camera line inspection area to exceed the black line.

You can try to increase the width of the black line to reduce the probability that the video area exceeds the black line. Two black wires (about 30mm) are used in the picture below.

Note: When the battery power is low, the driving force of the motor will be weaker.

If during the video tracking process, the car will make a reverse rotation **every time**, causing the video line inspection area to exceed the line inspection range.

1. Please check whether the steering of the trolley is correct by pressing the left and right arrow keys on the WEB interface. If it is correct, please refer to step 2. If not, please exchange the circuit connections of the two motors.

2. Please try to modify the codes on lines 206 and 212 in the "findLineCtrl" function in the camera_opencv.py file.

Change:

```
206.  move.move(turn_speed, 'no', 'right', 0.2)

212.  move.move(turn_speed, 'no', 'left', 0.2)
```

To:

```
move.move(turn_speed, 'no', 'left', 0.2)

move.move(turn_speed, 'no', 'right', 0.2)
```

- If you use the nano command that comes with Linux to modify the program, the operation steps are as follows:

```
sudo nano camera_opencv.py
```

Then save the modification by "Ctrl +X", "Y", "Enter".

- If you open the program in the program editor on the computer through MobaXterm software, you need to give permission to modify the file.

sudo chmod 777 [filename]

```
sudo chmod 777 camera_opencv.py
```

See "2 Basic course/Lesson 4 How to View or Edit the Code Program in Raspberry Pi" for details.