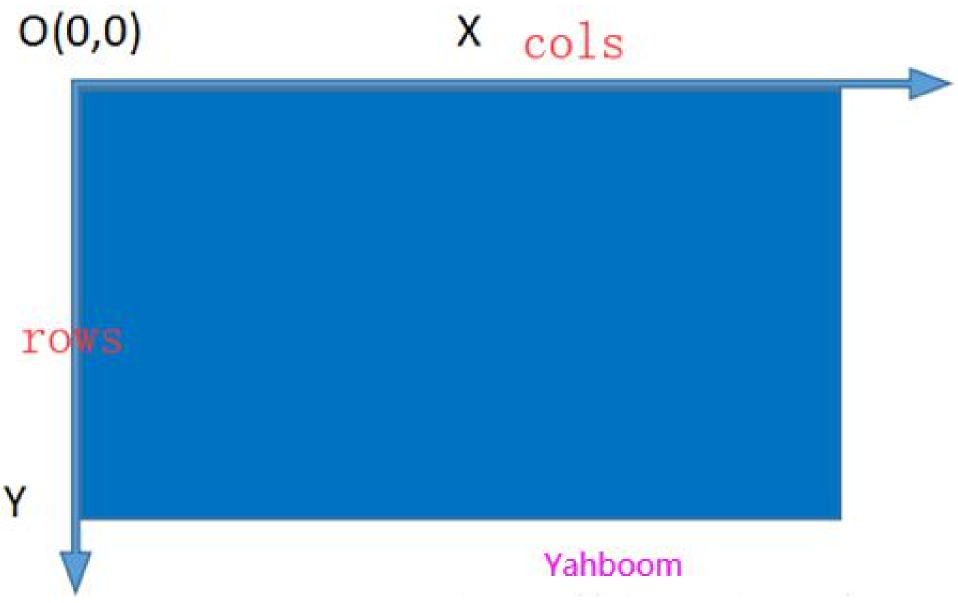
## 3.1.10 Autopilot

**1. OpenCV coordinate system**

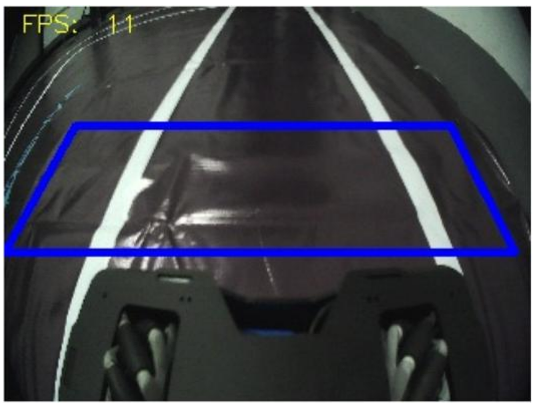
row == heigh == Point.y
col == width == Point.x
Mat::at(Point(x, y)) == Mat::at(y,x)

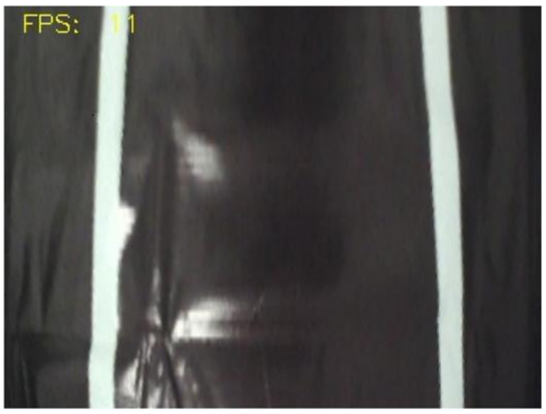


Obtain image: ret, frame = image.read()
Image resize: frame = cv2.resize(frame,(320,240))

**2.Perspective transformation**

Because the picture seen by the camera is distorted, we need to transform the area in front of the body into a top view through perspective transformation. Then zoom in to the entire picture area.
As shown in the figure below, the left is the original camera picture, and the right is the picture after perspective transformation.





【original camera picture】 【picture after perspective transformation】

**Function:**

matSrc = np.float32([[0, 149],     [310, 149], [271, 72], [43, 72]])
matDst = np.float32([[0,240], [310,240], [310,0], [0,0]])

```
        matAffine = cv2.getPerspectiveTransform(matSrc,matDst)# mat 1 src 2 dst
        dst = cv2.warpPerspective(frame,matAffine,(320,240))
```

Then, we can draw the rectangle based on the perspective coordinates.

```
        # Draw the rectangle box
        pts = np.array([[0, 149],    [310, 149], [271, 72], [43, 72]], np.int32)
        pts = pts.reshape((-1, 1, 2))
        cv2.polylines(frame, [pts],True, (255, 0, 0), 3)
```

**3.Binary image**

The core idea of binarization is to set a threshold, and the value greater than the threshold is 0 (black) or 255 (white), making the image called black and white. The threshold can be fixed or adaptive.

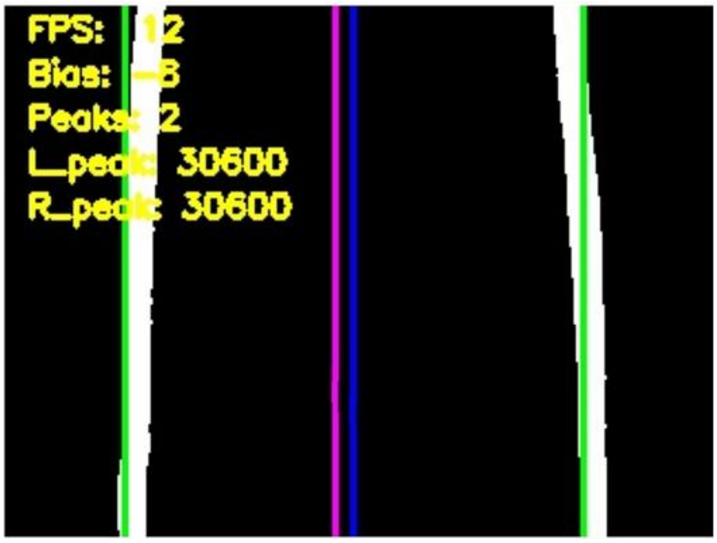
Threshold function is provided in Python-OpenCV:

```
    cv2.threshold（src, threshold, maxValue, method）
```

eg:

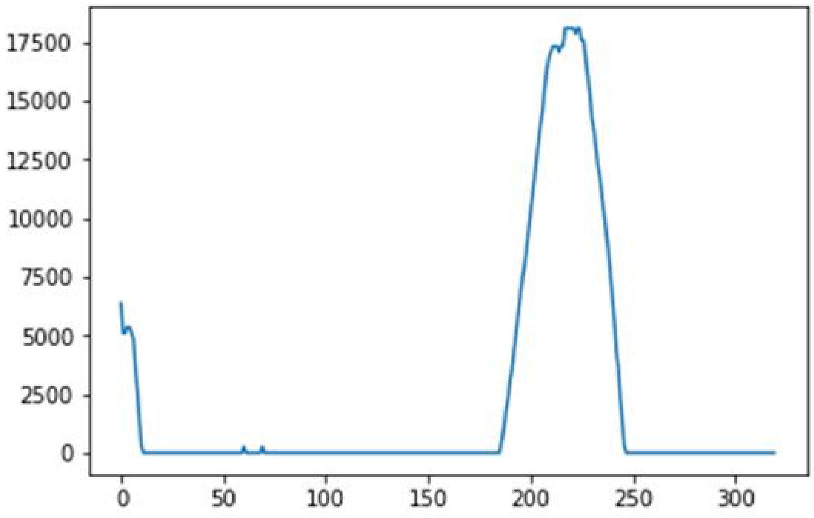
```
     dst_gray = cv2.cvtColor(dst, cv2.COLOR_RGB2GRAY)
     dst_retval, dst_binaryzation = cv2.threshold(dst_gray, 110, 255,
cv2.THRESH_BINARY)
```

After the perspective transformation of the picture, the converted image is shown below.

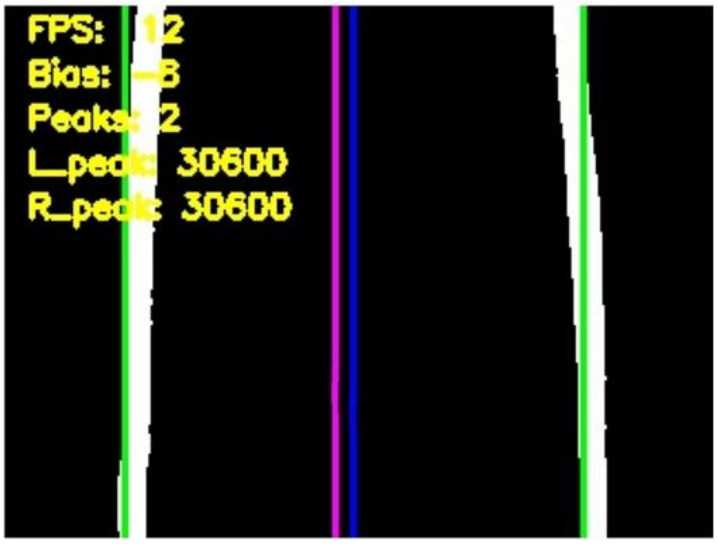


**4.Draw histogram**

!Note: During test, we need to open this code. During Autopilot, we need to close this code)



5.Get the black lines on both sides of the track and draw green lines



The size of image after perspective transformation is 320*240, so the center line of the image is: midpoint: 320 / 2-1 = 159
Or: midpoint = np.int (histogram.shape [0] / 2)

**Left line search method:** Take the coordinate position of the largest value in the left half of the histogram array, and implement the following function:
leftx_base = np.argmax(histogram[:midpoint], axis = 0)

**Right line search method:** Take the coordinate position of the largest value in the right half of the histogram array, and implement the following function:
rightx_base = np.argmax(histogram[midpoint:], axis = 0) + midpoint

**Actual midline position:** (left line + right line) / 2
lane_center = int ((leftx_base + rightx_base) / 2) #left and right line middle position

**Image center line**: This line is purple, cv2.line (dst_binaryzation, (159,0), (159,240), (255,0,255), 2)

**Left line:** This line is green, cv2.line (dst_binaryzation, (leftx_base, 0), (leftx_base, 240), (0,255,0), 2)

**Right line:** This line is green, cv2.line (dst_binaryzation, (rightx_base, 0), (rightx_base, 240), (0,255,0), 2)

**Actual center line:** This line is blue, cv2.line (dst_binaryzation, (lane_center, 0), (lane_center, 240), (255,0,0), 2)

**6. Calculate the offset**

After finding the white line of Autopilot in step 5, we can know the size of the body deviation according to the difference between the actual center line and the center line of the image, so that the car can be controlled to correct the deviation to achieve the purpose of Autopilot.

Calculation method:

**Bias = midpoint-lane_center**

According to the sign and magnitude of the offset value, we limit it to -20 ~ 20, which means that the maximum speed of the steering is 20, to prevent the steering from losing the white line of the track too quickly.

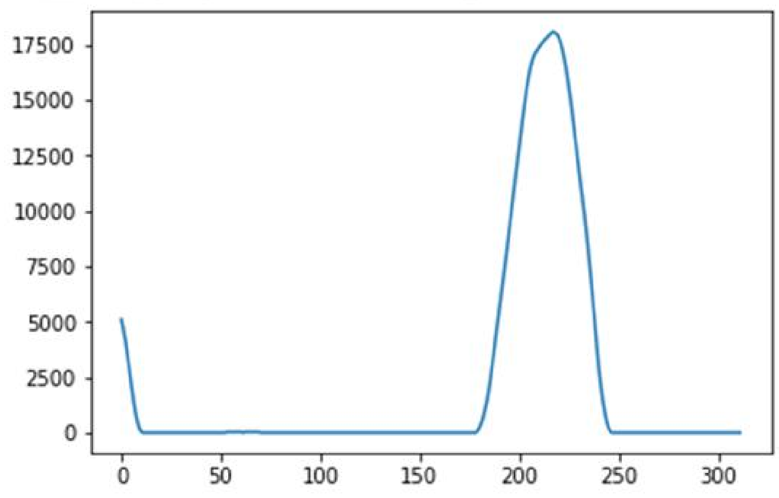
**Steering angle PID adjustment:**

```
Z_axis_pid.SystemOutput = Bias
 Z_axis_pid.SetStepSignal(0)
 Z_axis_pid.SetInertiaTime(0.5, 0.2)
```

After the PID calculation, the PID output is also limited to a maximum of ± 20.

```
if Z_axis_pid.SystemOutput > 20:
    Z_axis_pid.SystemOutput = 20
elif Z_axis_pid.SystemOutput < -20:
    Z_axis_pid.SystemOutput = -20
```

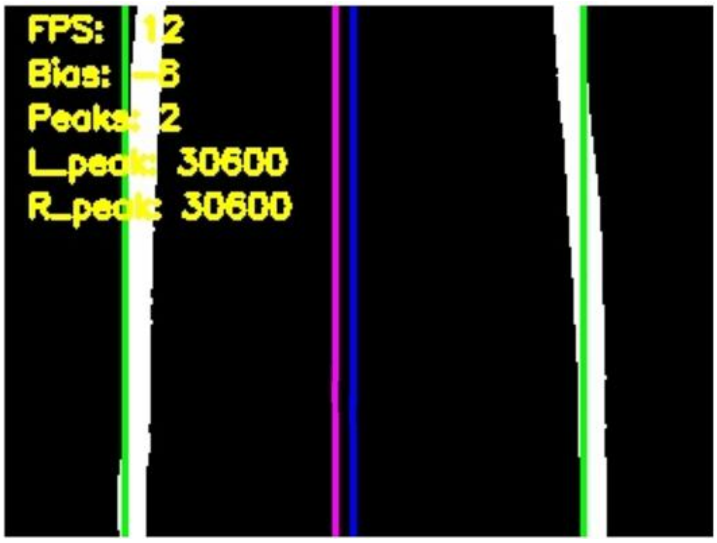
**7.Mean filtering to find the peak**



**Height of peak:**
Set a value above this value is considered a peak, otherwise it is considered to be noise fluctuations, it is recommended that this value is above the average.
Peak width: In order to avoid the appearance of several peaks in a peak, which can be judged as multiple peaks, the width of the peak is set here, and all of them are considered to belong to the same peak within this width.
Method:

```
length_data = len (filtered_signal)
thre = 1300             #peak height of peak
peak_width = 20      #Set the width of the peak
peaks_count = find_peak (filtered_signal, length_data, thre, peak_width)
#Calculate the number of peaks
```

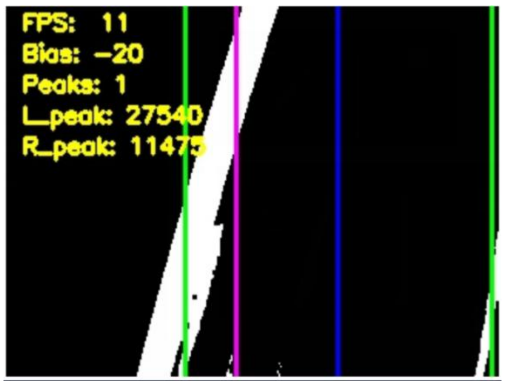
The image below shows some calculated values, as shown below:
FPS: frame rate
Bias: offset value
Peaks: number of peaks
L_peak: left front maximum
R_peak: right front maximum



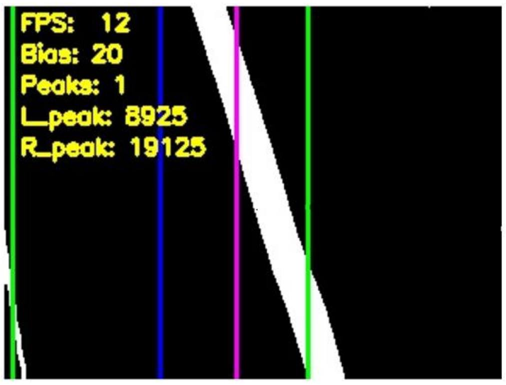
```
FPS: 12
Bias: -8
Peaks: 2
L_peak: 30600
R_peak: 30600
```

## 8.Control the robot to complete Autopilot

If there are two or four peaks: We think that the body deviation is not very large, just control the car to turn left and right.
If there is a peak: At this time, the robot will immediately run out of the track.
Two cases as shown below.



【Only see the left binarized image】



【Only see the right binarized image】

According to the example:
Left: Only one peak appears on the left, and the deviation of the blue and purple lines Bias = -20, so the robot should turn right to correct itself.
Right: Only one peak appears on the right, and the deviation of the blue and purple lines Bias = 20, so the robot should turn left to correct itself.
Other peaks: go straight.

Code as shown below.

```
#8.Start controlling robot movement
            if peaks_count == 2 or peaks_count == 4:
                  robot.Speed_axis_control(0,5,int(Z_axis_pid.SystemOutput)+3)
            elif peaks_count == 1:
                  if leftx_base_value == 0 or (rightx_base_value > leftx_base_value):
                        robot.Speed_axis_control(0,5,-20)
                  elif rightx_base_value == 0 or (leftx_base_value >
rightx_base_value):
                        robot.Speed_axis_control(0,5,20)
            elif peaks_count == 3:
                     robot.Speed_axis_control(0,5,0)
```

*Code path:*
*/home/pi/Yahboom_Project/3.AI_Visual_course/10.Autopilot.ipynb*