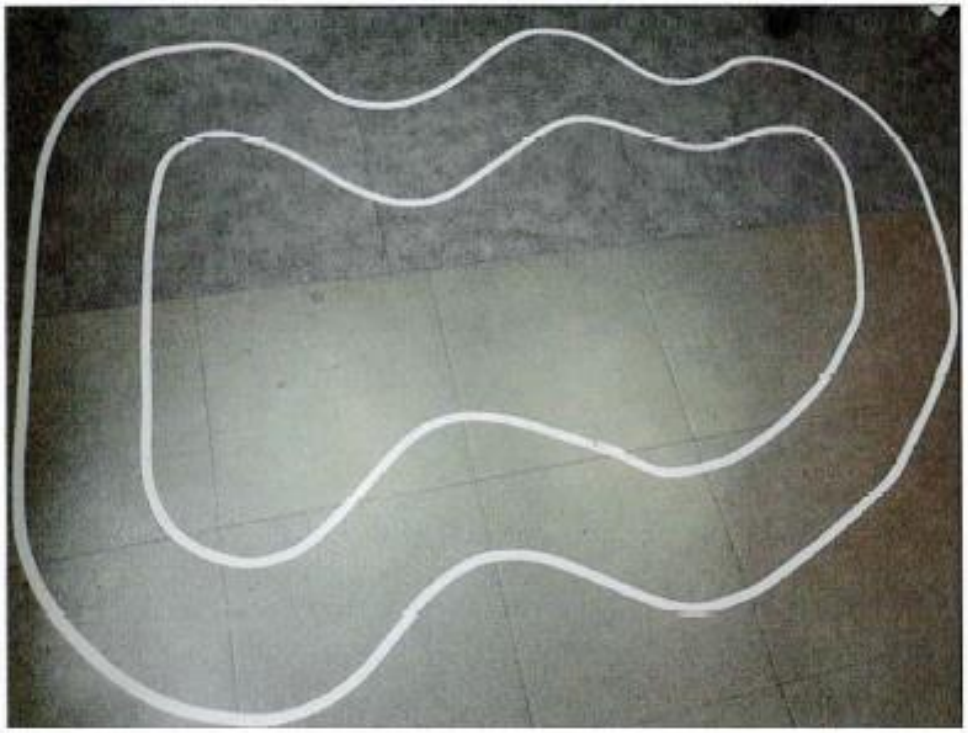
CH. 10. OpenCV 라인트레이서

Section 01 카메라 설정

❑ 회색 및 갈색 계열의 바닥에 흰색 테이프로 차선을 그려 사용하였다.



Section 01 카메라 설정

❑ 자율주행 자동차를 다음과 같이 차선 중앙에 위치시킨다.

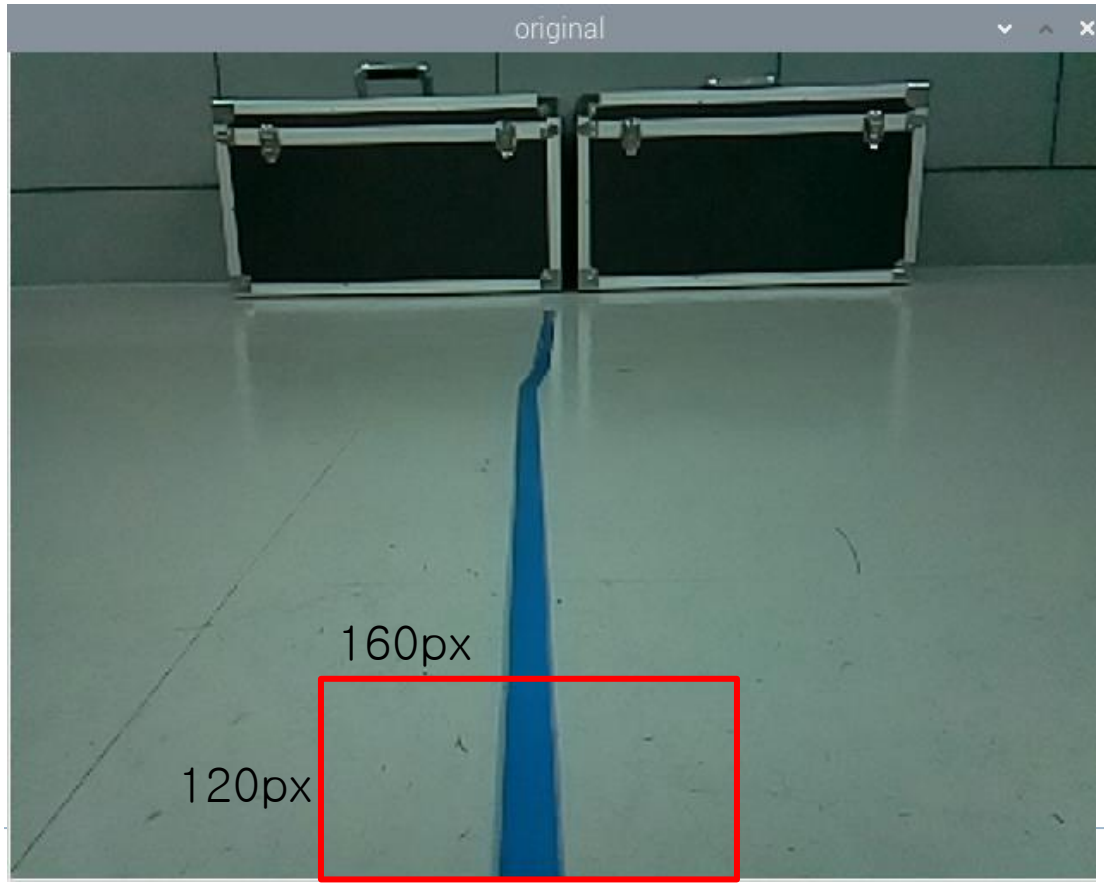
Section 01 카메라 설정

- 카메라로 보여지는 데이터를 확인해보자.
- 파이썬 코드를 작성하여 확인한다.

```
1   import cv2
2
3   cam = cv2.VideoCapture(0)
4   cam.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
5   cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
6
7   while( cam.isOpened() ):
8       ret, img = cam.read()
9       cv2.imshow('original', img)
10
11      if cv2.waitKey(1) == ord('q'):
12          break
13
14  cv2.destroyAllWindows()
```

Section 01 카메라 설정

- ❑ 카메라의 위쪽 부분에 라인과 상관없는 데이터가 있을 확률이 높다.
- ❑ 라인트레이서에 필요한 라인 정보만 얻기 위해 아래 부분의 데이터만 잘라서 확인해보자.



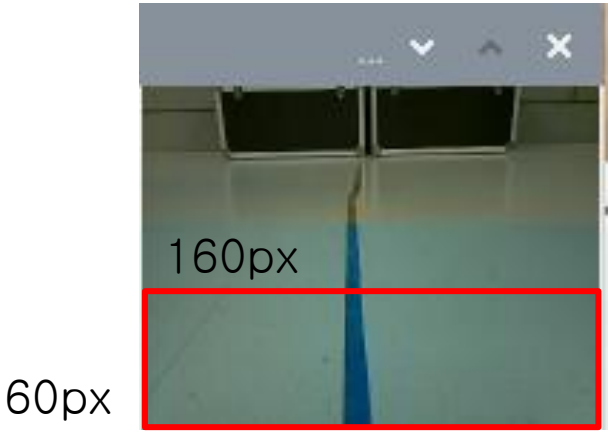
Section 01 카메라 설정

- 카메라의 위쪽 부분에 라인과 상관없는 데이터가 있을 확률이 높다.
- 라인트레이서에 필요한 라인 정보만 얻기 위해 아래 부분의 데이터만 잘라서 확인해보자.

```
1   import cv2
2
3   cam = cv2.VideoCapture(0)
4   cam.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
5   cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
6
7   while( cam.isOpened() ):
8       ret, img = cam.read()
9       cv2.imshow('original', img)
10
11      if cv2.waitKey(1) == ord('q'):
12          break
13
14  cv2.destroyAllWindows()
```

Section 01 카메라 설정

❑ 여기서 다시 불필요한 부분을 잘라서 정보 수집을 하도록 하자.



Section 01 카메라 설정

❑ 영상 데이터에서 내가 원하는 데이터를 자르는 코드를 만들어 보자.

```
1   import cv2
2
3   cam = cv2.VideoCapture(0)
4   cam.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
5   cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
6
7   while( cam.isOpened() ):
8       ret, img = cam.read()
9       cv2.imshow('original', img)
10
```

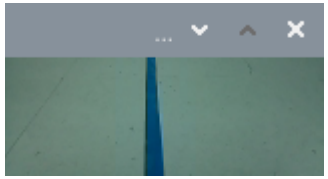
Section 01 카메라 설정

❑ 영상 데이터에서 내가 원하는 데이터를 자르는 코드를 만들어 보자.

```
12      crop_img = img[60:120, 0:160]
13      cv2.imshow('crop', crop_img)
14
15      if cv2.waitKey(1) == ord('q'):
16          break
17
18  cv2.destroyAllWindows()
```

Section 01 카메라 설정

❑ 영상 데이터에서 내가 원하는 데이터를 자르는 코드를 만들어 보자.



Section 02 영상 처리

- 이미지의 회색 처리와 블러 처리를 해보도록 하자.
- 다음과 같은 코드를 작성해보자.

```
1   import cv2
2
3   cam = cv2.VideoCapture(0)
4   cam.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
5   cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
6
7   while( cam.isOpened() ):
8       ret, img = cam.read()
9       cv2.imshow('original', img)
10
```

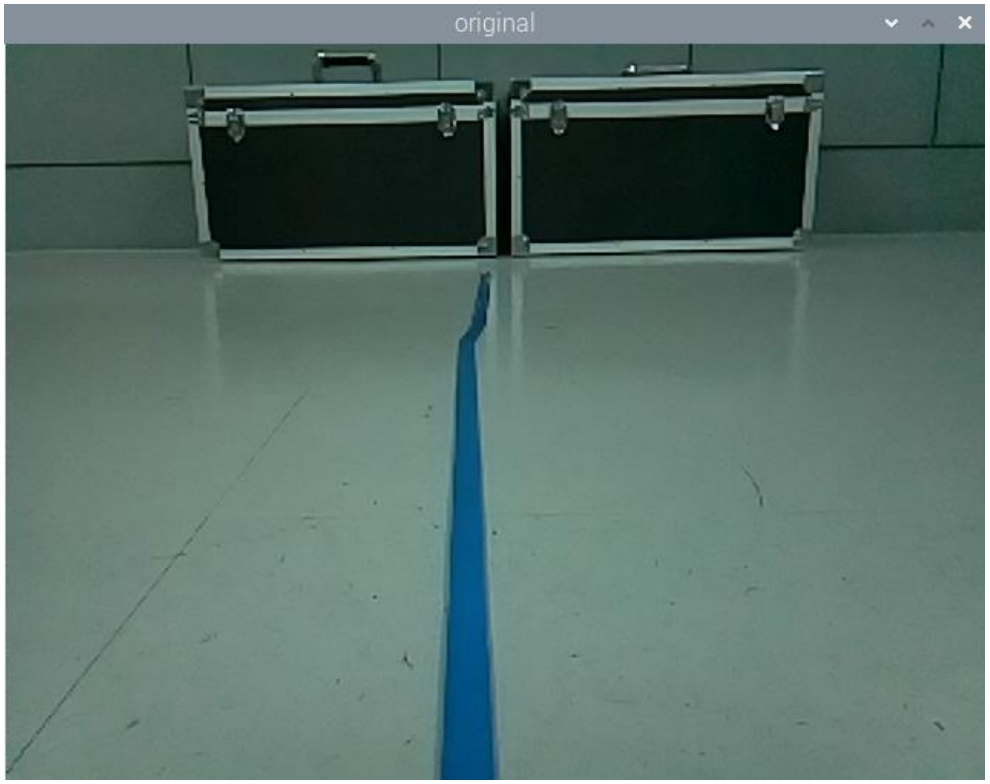
Section 02 영상 처리

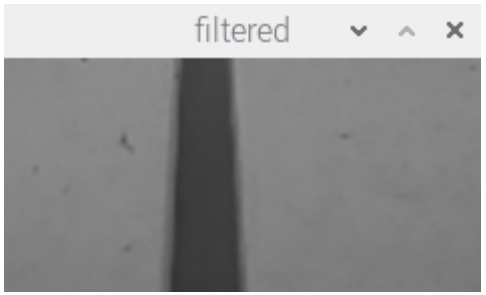
- 이미지의 회색 처리와 블러 처리를 해보도록 하자.
- 다음과 같은 코드를 작성해보자.

```
12      crop_img = img[60:120, 0:160]
13
14      gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
15
16      blur = cv2.GaussianBlur(gray, (5, 5), 0)
17
18      cv2.imshow('filtered', blur)
19
20      if cv2.waitKey(1) == ord('q'):
21          break
22
23  cv2.destroyAllWindows()
```

Section 02 영상 처리

- ❑ 예제 실행 결과는 회색조와 블러링 처리를 한 영상이다.
- ❑ 블러링 처리는 영상을 뽀얗게 만들어 준다.





Section 02 영상 처리

- 가우시안 블러링은 노이즈 제거에 효과적이지만 경계를 흐릿하게 만드는 단점도 존재한다.
- 바이레터럴 필터는 노이즈는 제거하면서 비교적 또렷한 영상을 만들 수 있지만 속도가 느리다.
- 이 두가지 필터를 비교해보자.

```
1   import cv2
2
3   cam = cv2.VideoCapture(0)
4   cam.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
5   cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
6
7   while( cam.isOpened() ):
8       ret, img = cam.read()
9       cv2.imshow('original', img)
10
```

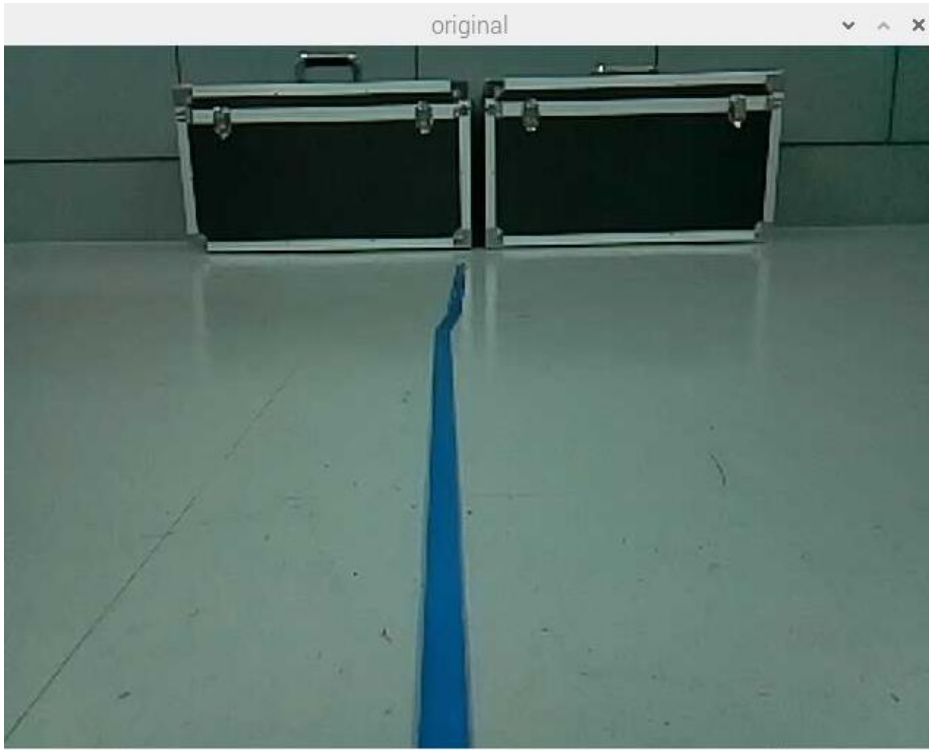
Section 02 영상 처리

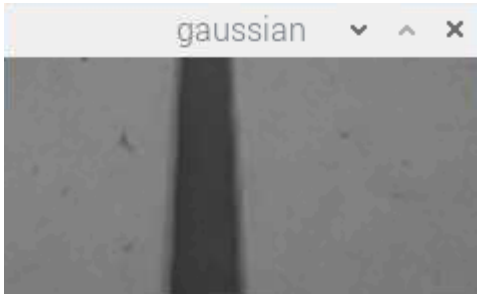
- 가우시안 블러링은 노이즈 제거에 효과적이지만 경계를 흐릿하게 만드는 단점도 존재한다.
- 바이레터럴 필터는 노이즈는 제거하면서 비교적 뚜렷한 영상을 만들 수
-

```
12      crop_img = img[60:120, 0:160]
13
14      gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
15
16      blur1 = cv2.GaussianBlur(gray, (5, 5), 0)
17      blur2 = cv2.bilateralFilter(gray, 5, 75, 75)
18
19      cv2.imshow('gaussian', blur1)
20      cv2.imshow('bilateral', blur2)
21
22      if cv2.waitKey(1) == ord('q'):
23          break
24
25  cv2.destroyAllWindows()
```

Section 02 영상 처리

❑ 두가지 필터를 비교한 실행 결과이다.





Section 02 영상 처리

- 이제 임계점의 값을 찾는 단계로 중요한 단계이다.
- 이미지에서 임계점을 설정하고 임계점 이상의 값을 최대값으로 바꾸어 라인을 찾기 쉽게 변환해보자.

```
1   import cv2
2
3   cam = cv2.VideoCapture(0)
4   cam.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
5   cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
6
7   while( cam.isOpened() ):
8       _, img = cam.read()
9       cv2.imshow('original', img)
10
11      crop_img = img[60:120, 0:160]
12
```

Section 02 영상 처리

- 이제 임계점의 값을 찾는 단계로 중요한 단계이다.
- 이미지에서 임계점을 설정하고 임계점 이상의 값을 최대값으로 바꾸어 라인을 찾기 쉽게 변환해보자.

```
14      gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
15
16      blur = cv2.GaussianBlur(gray, (5, 5), 0)
17
18      ret, thresh1 = cv2.threshold(blur, 100, 255, cv2.THRESH_BINARY)
19
20      cv2.imshow('thresh1',thresh1)
21
22      if cv2.waitKey(1) == ord('q'):
23          break
24
25  cv2.destroyAllWindows()
```

Section 02 영상 처리

- 예제 실행 결과에서 왼쪽 그림의 경우 임계점이 100을 넘었을 때 255(검정)으로 바뀌어서 라인이 검정으로 잘 처리되었다.
- 오른쪽 그림은 임계점이 100이상일 경우로 노이즈가 심하다.
- 이처럼 임계점 설정에 따라서 선을 인식하는 범위가 틀려진다.
- 빛의 상태에 따라 틀리고 라인의 색상에 따라 틀리다.

Section 02 영상 처리

- 이처럼 여러 요인으로 인해 틀려질 수 있으므로 상황에 맞춰 임계점을 잘 설정할 필요가 있는데 값을 수정하면서 임계점을 설정하는 것은 비효율적이다.
- 따라서 오츠의 알고리즘을 적용하도록 하겠다.

```
1   import cv2
2
3   cam = cv2.VideoCapture(0)
4    cam.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
5    cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
6
7   while( cam.isOpened() ):
8       _, img = cam.read()
9
10      cv2.imshow('original', img)
11
12      crop_img = img[60:120, 0:160]
13
14      gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
```

Section 02 영상 처리

- 따라서 오츠의 알고리즘을 적용하도록 하겠다.

```
15
16      blur1 = cv2.GaussianBlur(gray, (5, 5), 0)
17      blur2 = cv2.bilateralFilter(gray, 5, 75, 75)
18
19      _, thresh1 = cv2.threshold(blur1, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
20      _, thresh2 = cv2.threshold(blur2, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
21
22      cv2.imshow('thresh1',thresh1)
23      cv2.imshow('thresh2',thresh2)
24
25      if cv2.waitKey(1) == ord('q'):
26          break
26
27  cv2.destroyAllWindows()
```

Section 02 영상 처리

- 다음은 오츠의 알고리즘과 두가지 필터를 적용한 결과이다.

Section 02 영상 처리

❑ 자 여기에서 이미지의 주변 노이즈를 한번 더 제거하기 위해 열림 연산과 닫힘 연산을 적용해 보자.

```
1   import cv2
2
3   cam = cv2.VideoCapture(0)
4    cam.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
5    cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
6
7   while( cam.isOpened() ):
8       _, img = camera.read()
9       cv2.imshow('original', img)
10
11      crop_img = img[60:120, 0:160]
12
13      gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
14
15      blur = cv2.bilateralFilter(gray, 5, 75, 75)
16
```

Section 02 영상 처리

❑ 자 이제 차선을 배경과 분리하도록 하자. 이를 위해서 먼저 차선의 윤곽선을 그려야 한다.

```
17      _, thresh1 = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
18
19      k = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
20      opening = cv2.morphologyEx(thresh1, cv2.MORPH_OPEN, k)
21      closing = cv2.morphologyEx(thresh1, cv2.MORPH_CLOSE, k)
22      cv2.imshow('opening', opening)
23      cv2.imshow('closing', closing)
24
25      contours1, _ = cv2.findContours(opening.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_NONE)
26      contours2, _ = cv2.findContours(closing.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_NONE)
27
28      cv2.drawContours(crop_img, contours1, -1, (0, 0, 255), 4)
29      cv2.drawContours(crop_img, contours2, -1, (255, 0, 0), 4)
30      cv2.imshow('contour', crop_img)
31
```

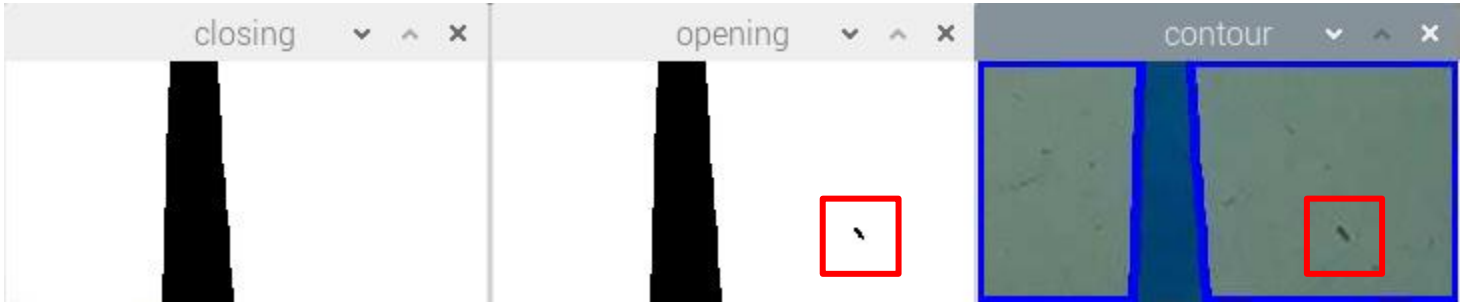
Section 02 영상 처리

❑ 자 이제 차선을 배경과 분리하도록 하자. 이를 위해서 먼저 차선의 윤곽선을 그려야 한다.

```
32      if cv2.waitKey(1) == ord('q'):
33          break
34
35  cv2.destroyAllWindows()
```

Section 02 영상 처리

- 실행 결과에서 좌측 그림은 닫힘 연산, 가운데 그림은 열림 연산을 적용한 결과이다.
- 닫힘 연산의 경우 주변의 커다란 노이즈 하나를 제거했다.
- 우측 그림은 윤곽선을 그린 결과인데, 차선을 따라서 그린 것이 아니라 차선을 제외한 나머지 영역의 윤곽선을 그렸다.
- 따라서 이를 반전시킬 연산이 필요하다.
- 이를 위한 가장 단순한 방법은 배경은 검정색, 전경은 흰색이 되도록 반전시키는 것이다.



Section 02 영상 처리

❑ 차선 영상의 배경과 전경을 반전시켜보자.

```
1   import cv2
2
3   cam = cv2.VideoCapture(0)
4    cam.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
5    cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
6
7   while( cam.isOpened() ):
8       _, img = camera.read()
9       cv2.imshow('original', img)
10
11      crop_img = img[60:120, 0:160]
12
13      gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
14
15      blur = cv2.bilateralFilter(gray, 5, 75, 75)
16
```

Section 02 영상 처리

❑ 차선 영상의 배경과 전경을 반전시켜보자.

```
17    _, thresh1 = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)
18
19    k = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
20    opening = cv2.morphologyEx(thresh1, cv2.MORPH_OPEN, k)
21    closing = cv2.morphologyEx(thresh1, cv2.MORPH_CLOSE, k)
22    cv2.imshow('opening', opening)
23    cv2.imshow('closing', closing)
24
25    contours1, _ = cv2.findContours(opening.copy(), cv2.RETR_EXTERNAL,
      cv2.CHAIN_APPROX_NONE)
26    contours2, _ = cv2.findContours(closing.copy(), cv2.RETR_EXTERNAL,
      cv2.CHAIN_APPROX_NONE)
27
28    #cv2.drawContours(crop_img, contours1, -1, (0, 0, 255), 4)
29    cv2.drawContours(crop_img, contours2, -1, (255, 0, 0), 4)
30    cv2.imshow('contour', crop_img)
31
```

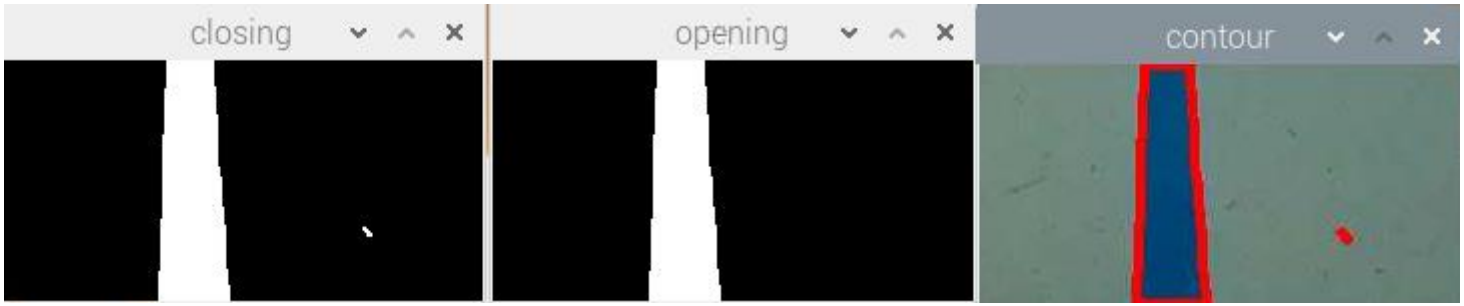
Section 02 영상 처리

❑ 차선 영상의 배경과 전경을 반전시켜보자.

```
32      if cv2.waitKey(1) == ord('q'):
33          break
34
35  cv2.destroyAllWindows()
```

Section 02 영상 처리

- 실행 결과에서 좌측 그림은 닫힘 연산, 가운데 그림은 열림 연산을 적용한 결과이다.
- 이번 예제에서는 흑, 백이 반전되었기 때문에 연산 결과 또한 반전된다.
- 따라서, 이번에는 열림 연산을 적용해야 한다.
- 우측 그림은 윤곽선을 그린 결과인데, 차선을 따라서 윤곽선이 그려졌음을 확인할 수 있다.



Section 02 영상 처리

- 이제 차선의 중심을 찾는 코드가 필요하다.
- 차가 이동하게 되면 카메라 위치에 따라서 차선의 위치가 바뀌게 된다.
- 따라서 차량의 위치에 관계없이 차선의 중심을 찾아서 차선의 중심을 따라서 이동하도록 해야 한다.
- 이를 위해 필요한 것이 중심 모멘트이다.

Section 02 영상 처리

❑ 이제 중심 모멘트를 이용하여 차선의 중심을 찾는 코드를 작성해보자.

```
1   import cv2
2
3   cam = cv2.VideoCapture(0)
4    cam.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
5    cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
6
7   while( cam.isOpened() ):
8       _, img = camera.read()
9       cv2.imshow('original', img)
10
11      crop_img = img[60:120, 0:160]
12
13      gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
14
15      blur = cv2.bilateralFilter(gray, 5, 75, 75)
16
```

Section 02 영상 처리

❑ 여러 요인으로 인해 틀려질 수 있으므로 상황에 맞춰 임계점을 잘 설정하고 다음 단계로 진행한다.

```
17    _, thresh1 = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)
18
19    k = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
20    opening = cv2.morphologyEx(thresh1, cv2.MORPH_OPEN, k)
21    cv2.imshow('opening', opening)
22
23    contours, _ = cv2.findContours(opening.copy(), cv2.RETR_EXTERNAL,
      cv2.CHAIN_APPROX_NONE)
24
25    M = cv2.moments(contours[0])
26
27    cx = int(M['m10']/M['m00'])
28    cy = int(M['m01']/M['m00'])
29
30    cv2.drawContours(crop_img, contours2, -1, (0, 0, 255), 4)
31    cv2.circle(crop_img, (cx, cy), 3, (0, 0, 255), -1)
```

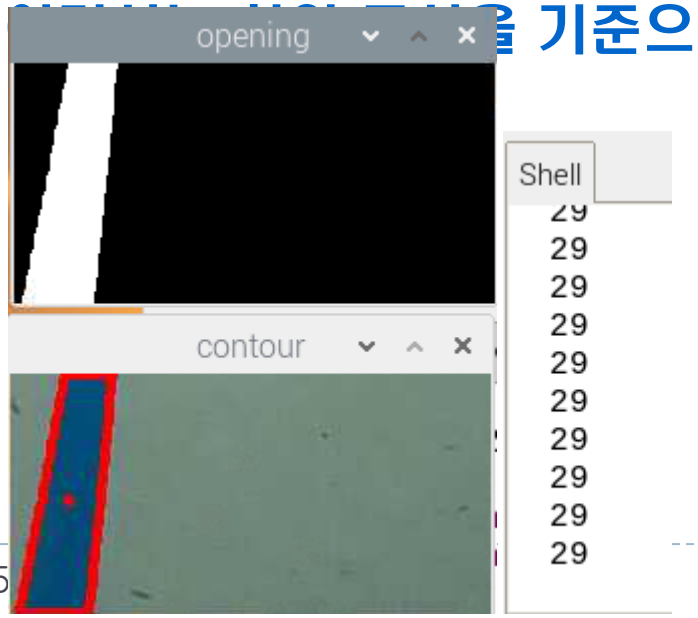
Section 02 영상 처리

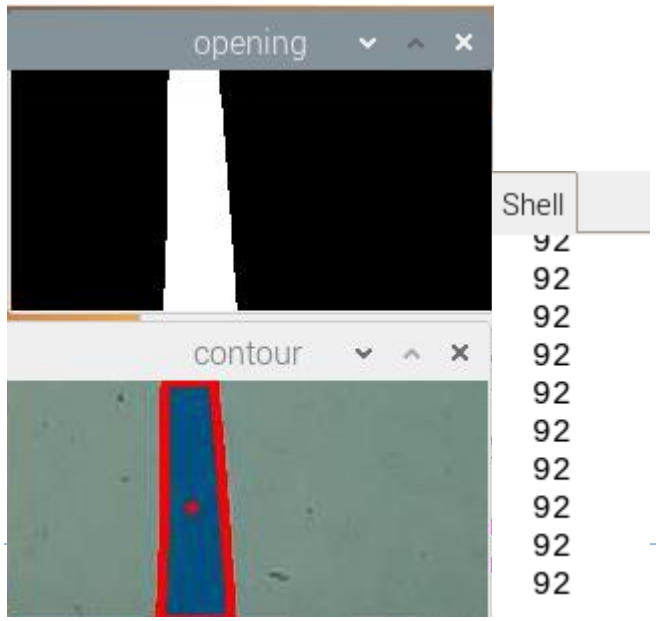
❑ 여러 요인으로 인해 틀려질 수 있으므로 상황에 맞춰 임계점을 잘 설정하고 다음 단계로 진행한다.

```
31
32          cv2.imshow(‘contour’, crop_img)
33
34          print(cx)
35
36      if cv2.waitKey(1) == ord('q'):
37          break
38
39  cv2.destroyAllWindows()
```

Section 02 영상 처리

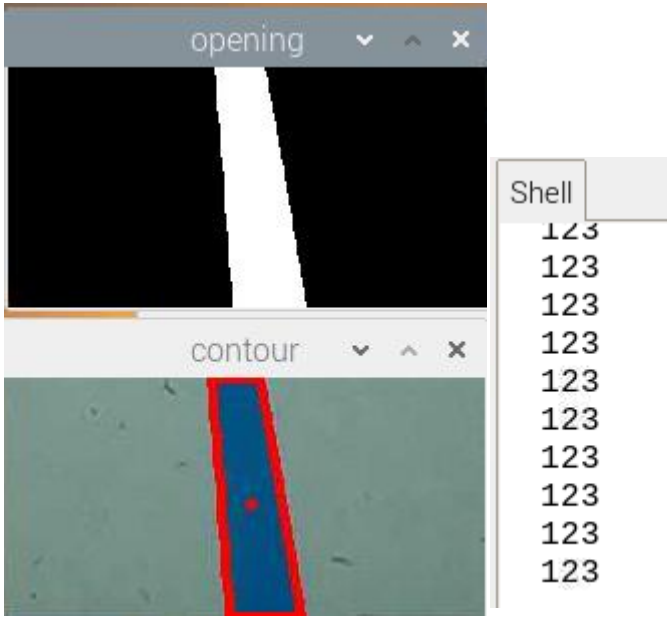
- 예제 실행 결과를 보면, 선이 중심보다 왼쪽이 있을 때는 30 ~ 95의 무게 중심을 갖는다.
- 이 때 자동차의 카메라의 위치가 차의 중심에서 왼쪽으로 치우쳐있기 때문에 기준점 설정에 주의해야 한다.
- 차의 중심을 차선의 중심에 놓았을 때 무게 중심을 기준으로 하던지 아니면, 카메라의 중심을 차선의 중심에 놓았을 때의 무게 중심을 기준으로 하던지 결정해야 한다.
- [illegible]을 기준으로

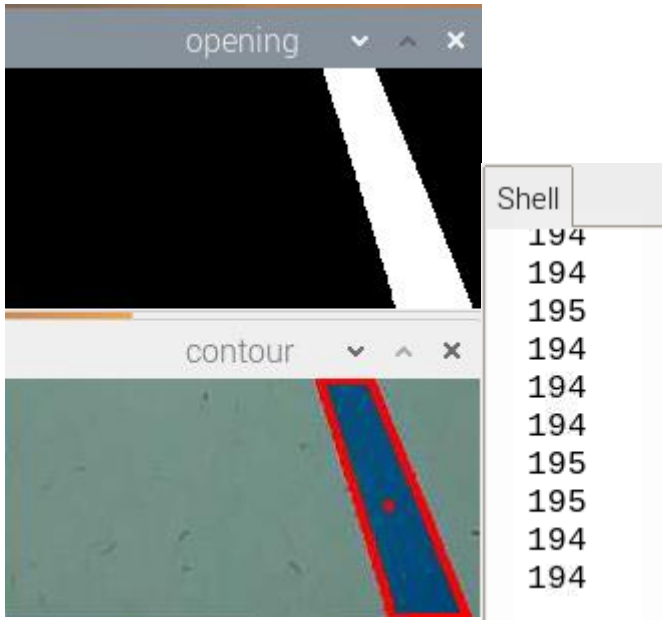




Section 02 영상 처리

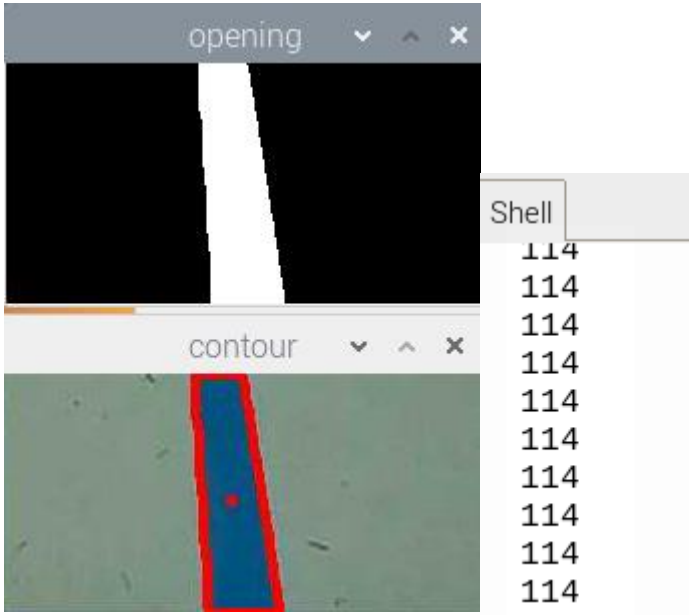
❑ 또한, 선이 중심보다 오른쪽에 있을 때는 125~195 의 무게중심을 갖는다.





Section 02 영상 처리

❑선이 중심에 있을 때는 95~125 의 무게중심을 갖는다.



❑ 정리하면 다음과 같다.

- 선이 중심보다 왼쪽이 있을 때 즉, 무게 중심이 30~95 일 때는 자동차가 선의 오른쪽으로 치우쳐서 이동중에 있으므로 차량을 왼쪽으로 이동시킨다.
- 선이 중심보다 오른쪽에 있을 때 즉, 무게 중심이 125~195 일 때는 자동차가 선의 왼쪽으로 치우쳐서 이동중에 있으므로 차량을 오른쪽으로 이동시킨다.
- 그 외의 조건은 자동차를 직진시킨다.

Section 03 OpenCV 라인트레이서

❑ 앞 절에서 정리한 내용을 코드에 적용해보자.

```
1   import cv2
2
3   cam = cv2.VideoCapture(0)
4    cam.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
5    cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
6
7   while( cam.isOpened() ):
8       _, img = camera.read()
9       cv2.imshow('original', img)
10
11      crop_img = img[60:120, 0:160]
12
13      gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
14
15      blur = cv2.bilateralFilter(gray, 5, 75, 75)
16
```

Section 03 OpenCV 라인트레이서

❑ 앞 절에서 정리한 내용을 코드에 적용해보자.

```
17    _, thresh1 = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)
18
19    k = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
20    opening = cv2.morphologyEx(thresh1, cv2.MORPH_OPEN, k)
21    cv2.imshow('opening', opening)
22
23    contours, _ = cv2.findContours(opening.copy(), cv2.RETR_EXTERNAL,
      cv2.CHAIN_APPROX_NONE)
24
25    M = cv2.moments(contours[0])
26
27    cx = int(M['m10']/M['m00'])
28    cy = int(M['m01']/M['m00'])
29
```

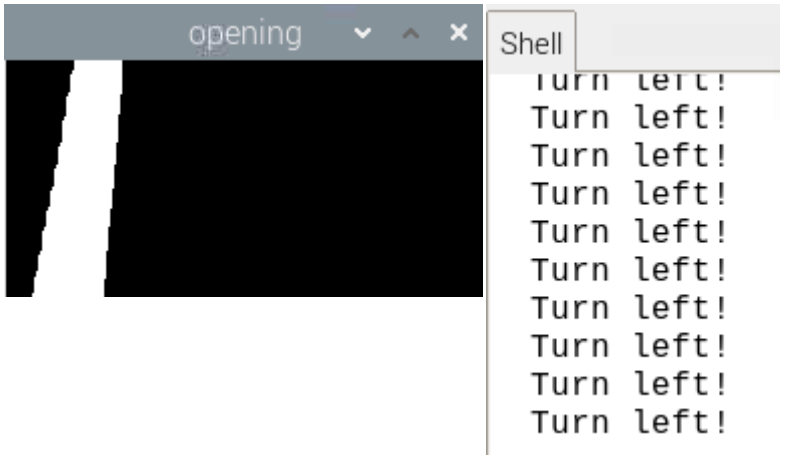
Section 03 OpenCV 라인트레이서

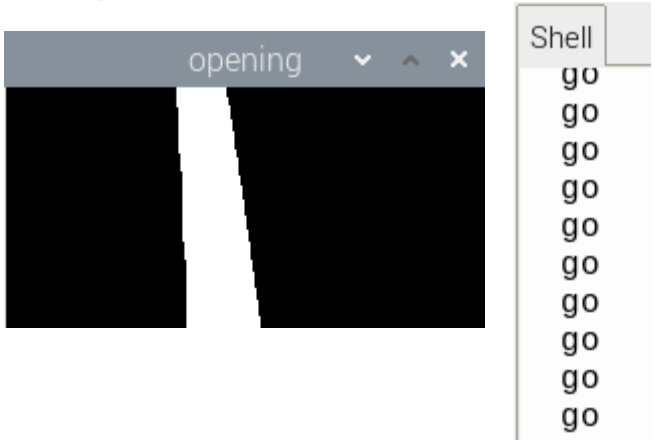
❑ 앞 절에서 정리한 내용을 코드에 적용해보자.

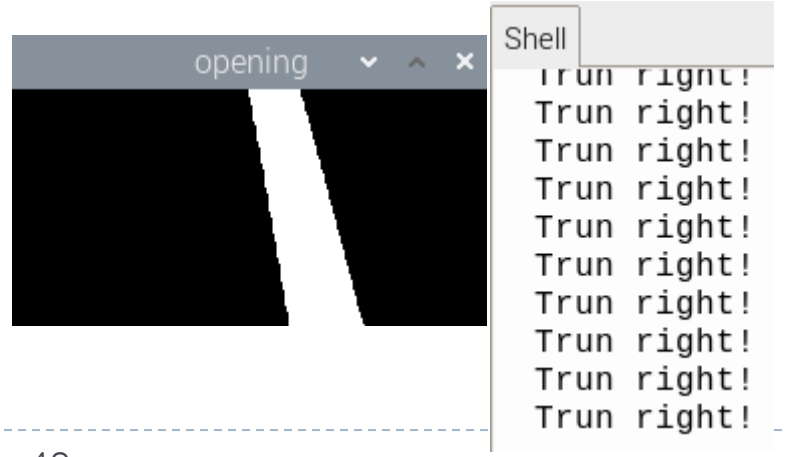
```
34          if cx >= 125 and cx <= 150:
35              print("Turn Left!")
36          elif cx >= 15 and cx <= 70:
37              print("Turn Right")
38          elif cx > 70 and cx <= 125:
39              print("go")
40
41      if cv2.waitKey(1) == ord('q'):
42          break
43
44  cv2.destroyAllWindows()
```

Section 03 OpenCV 라인트레이서

❑ 앞 절에서 정리한 내용을 코드에 적용해보자.







Section 03 OpenCV 라인트레이서

❑ 이제 실제로 모터가 동작하는 코드를 넣어 완성한다.

```
1   import cv2
2   import time
3   from ctypes import *
4   import os
5
6   WiringPi = CDLL(“/home/pi/WiringPi/wiringPi/libwiringPi.so.2.70”, mode=RTLD_GLOBAL)
7   swcar = cdll.LoadLibrary(‘/home/pi/swcar/libswcar.so’)
8
9    motor_status = "STOP"
10   servo_status = "CENTER"
11
12   def motor_forward(speed):
13       global motor_status
14       if (motor_status == "REVERSE") :
15           swcar.SIO_WriteMotor(100, 0)
16           time.sleep(0.1)
17       swcar.SIO_WriteMotor(100, speed)
18       motor_status = "FORWARD"
```

Section 03 OpenCV 라인트레이서

❑ 이제 실제로 모터가 동작하는 코드를 넣어 완성한다.

```
19
20  def motor_reverse(speed):
21      global motor_status
22      if (motor_status == "FORWARD") :
23          swcar.SIO_WriteMotor(100, 0)
24          time.sleep(0.1)
25      swcar.SIO_WriteMotor(100, speed)
26      motor_status = "REVERSE"
27
28  def motor_stop():
29      global motor_status
30      swcar.SIO_WriteMotor(100, 0)
31      motor_status = "STOP"
32
33  def servo_left():
34      global servo_status
35      swcar.SIO_WriteServo(100, 90)
36      servo_status = "LEFT"
```

Section 03 OpenCV 라인트레이서

❑ 이제 실제로 모터가 동작하는 코드를 넣어 완성한다.

```
38
39  def servo_right():
40      global servo_status
41      swcar.SIO_WriteServo(100, 10)
42      servo_status = "RIGHT"
43
44  def servo_center():
45      global servo_status
46      swcar.SIO_WriteServo(100, 50)
47      servo_status = "CENTER"
48
49  swcar.SIO_Init(0)
50
51  motor_stop()
52  servo_center()
53  time.sleep(2)
54
```

Section 03 OpenCV 라인트레이서

❑ 이제 실제로 모터가 동작하는 코드를 넣어 완성한다.

```
55  print('Press ctrl + c to terminate program')
56  print('Running Driving Motor.')
57
58  cam = cv2.VideoCapture(0)
59  cam.set(cv2.CAP_PROP_FRAME_WIDTH, 160)
60  cam.set(cv2.CAP_PROP_FRAME_HEIGHT, 120)
61
62  while( cam.isOpened() ):
63      _, img = camera.read()
64      cv2.imshow('original', img)
65
66          dist1 = swcar.SIO_ReadDistLS()
67          dist2 = swcar.SIO_ReadDistUS(1)
68          dist3 = swcar.SIO_ReadDistUS(0)
69
70          if (dist1 < 200) or (dist2 < 150) or (dist3 < 150) :
71              motor_stop()
73
```

Section 03 OpenCV 라인트레이서

❑ 이제 실제로 모터가 동작하는 코드를 넣어 완성한다.

```
74      crop_img = img[60:120, 0:160]
75
76      gray = cv2.cvtColor(crop_img, cv2.COLOR_BGR2GRAY)
77
78      blur = cv2.bilateralFilter(gray, 5, 75, 75)
79
80      k = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
81      opening = cv2.morphologyEx(thresh1, cv2.MORPH_OPEN, k)
82      cv2.imshow('opening', opening)
83
84      contours, _ = cv2.findContours(opening.copy(), cv2.RETR_EXTERNAL,
    cv2.CHAIN_APPROX_NONE)
85
86      M = cv2.moments(contours[0])
87
88      cx = int(M['m10']/M['m00'])
89      cy = int(M['m01']/M['m00'])
90
```

Section 03 OpenCV 라인트레이서

❑ 이제 실제로 모터가 동작하는 코드를 넣어 완성한다.

```
91      if cx >= 25 and cx <= 95:
92          print("Turn Left!")
93          servo_left()
94          motor_forward(fspeed)
95          time.sleep(0.3)
96          motor_stop()
97      elif cx >= 125 and cx <= 195:
98          print("Turn Right")
99          servo_right()
100         motor_forward(fspeed)
101         time.sleep(0.3)
102         motor_stop()
103     elif cx > 70 and cx <= 125:
104         print("go")
105         motor_forward(fspeed)
106         time.sleep(0.3)
107         motor_stop()
```

Section 03 OpenCV 라인트레이서

❑ 이제 실제로 모터가 동작하는 코드를 넣어 완성한다.

```
107     else:
108         print(“back”)
109         servo_center()
110         motor_reverse(bspeed)
111         time.sleep(0.3)
112         motor_stop()
113
114     if cv2.waitKey(1) == ord('q'):
115         motor_stop()
116         servo_center()
117         break
118
119 cv2.destroyAllWindows()
```

Q&A