

스마트홈 IoT가전기기 제어를 위한 객체인식 모델구현

사전

딥러닝을 위한 파이썬 프로그래밍과
영상 처리 개념 이해(9)

학습내용

- OpenCV 인터페이스
 - 직선/사각형 그리기
 - 글자쓰기, 원 그리기
- 영상파일 처리하기

학습목표

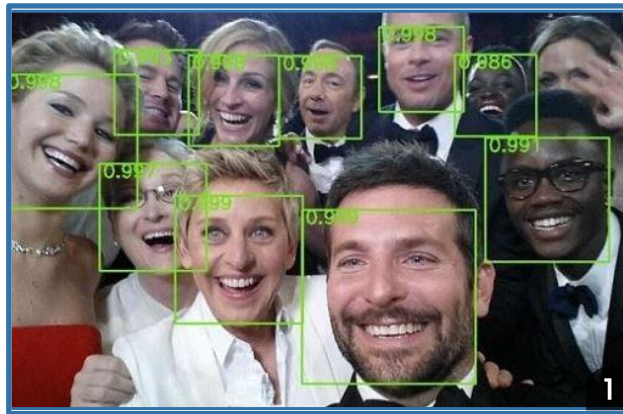
- OpenCV 라이브러리를 이용해서 파이썬 프로그램 코딩을 이해할 수 있다.

OpenCV 인터페이스

OpenCV 인터페이스

◆ 그리기 함수

- 화소(Pixel) 영상 처리 프로그래밍 과정
 - 입력으로 사용한 영상 위에 다양한 그리기 함수를 적용
 - 영상 처리 결과를 직접 확인 가능
 - 얼굴 검출 알고리즘을 적용했을 때
 - 전체 영상 위에 검출한 얼굴 영역을 사각형이나 원으로 표시



- 차선을 확인하고자 직선 검출 알고리즘을 적용했을 때
- 차선을 정확하게 검출했는지 확인하기 위해 도로 영상 위에 선으로 표시



OpenCV 인터페이스

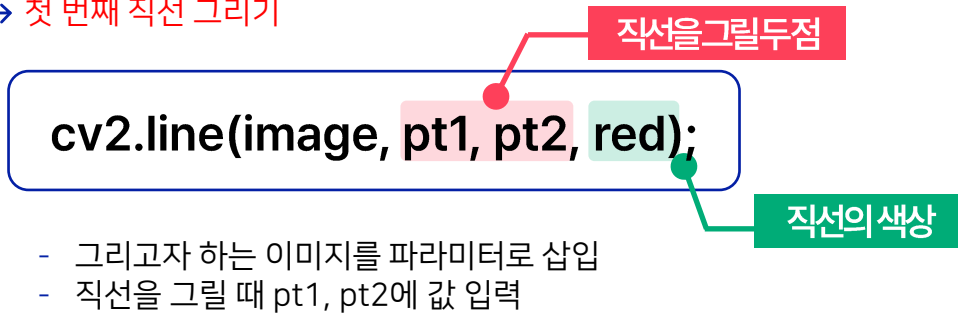
◆ 직선 그리기 함수

• cv2.line()

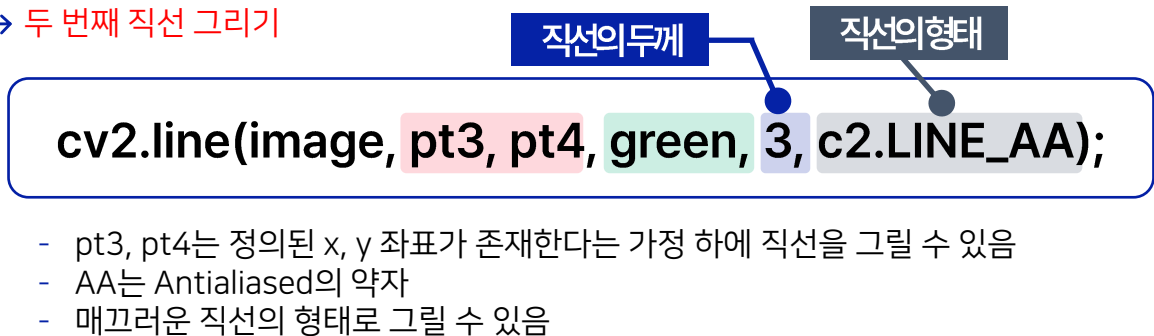
→ 두 점을 이용하여 직선을 그리고 싶을 때 사용

- 그리고자 하는 다양한 점의 정보와 선의 정보를 입력하여 직선을 그릴 수 있음

→ 첫 번째 직선 그리기



→ 두 번째 직선 그리기



OpenCV 인터페이스


◆ 직선 그리기 함수

- cv2.rectangle()

- 사각형을 그리고 싶을 때 사용
- OpenCV 라이브러리에서는 두 개의 점을 이용하여 사각형을 그릴 수 있음
 - 두 번의 클릭으로 사각형을 만드는 것과 유사함
- 첫 번째 사각형 그리기

직선의형태

```
cv2.rectangle
(image, pt1, pt2, blue, 3, cv2.LINE_4);
```



- pt1과 pt2라는 점을 이용
- 두께가 3인 파란색 사각형 그리기
- 4개의 영역을 참조하는 파라미터를 사용한 경우

- 두 번째 사각형 그리기

```
cv2.rectangle
(image, roi, red, 3, cv2.LINE_8);
```

- 8개의 영역을 사용하는 파라미터 옵션 사용
- roi는 4개의 점을 하나의 리스트로 묶어서 정의한 표현

OpenCV 인터페이스

◆ 직선 그리기 함수

```

01 import numpy as np
02 import cv2
03
04 blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)
05 image = np.zeros((400, 600, 3), np.uint8)
06 image[:] = (255, 255, 255)
07
08 pt1, pt2 = (50, 50), (250, 350)
09 pt3, pt4 = (400, 150), (500, 50)
10 roi = (50, 200, 200, 100)
11
12 ## 직선 그리기
13 cv2.line(image, pt1, pt2, red)
14 cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)
15
16 ## 사각형 그리기
17 cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)
18 cv2.rectangle(image, roi, red, 3, cv2.LINE_8)
19 cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)
20
21 cv2.imshow("Line & Rectangle", image)
22 cv2.waitKey(0)
23 cv2.destroyAllWindows()

```

OpenCV 인터페이스

◆ 직선 및 사각형 그리기

```

01 import numpy as np
02 import cv2
03
04 blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)
05 image = np.zeros((400, 600, 3), np.uint8)
06 image[:] = (255, 255, 255)
07
08 pt1, pt2 = (50, 50), (250, 350)
09 pt3, pt4 = (400, 150), (500, 50)
10 roi = (50, 200, 200, 100)

```

numpy를 np로 정의

OpenCV 사용 선언

blue 라인으로 직선을 그리고자 할 때

→ 색상 선언

- blue, green, red 순서로 정의 가능

- pt1, pt2, pt3, pt4, roi를 이용하여 직선과 사각형 그리기

```

11
12 ## 직선 그리기
13 cv2.line(image, pt1, pt2, red)
14 cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)
15
16 ## 사각형 그리기
17 cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)
18 cv2.rectangle(image, roi, red, 3, cv2.LINE_8)
19 cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)
20
21 cv2.imshow("Line & Rectangle", image)
22 cv2.waitKey(0)
23 cv2.destroyAllWindows()

```

pt1, pt2를 이용하여 두께로 빨간색 직선 설정

pt3, pt4를 이용하여 두께 3으로 매끄러운 녹색 직선 설정

OpenCV 인터페이스

◆ 직선 및 사각형 그리기

- 사각형 그리기 함수 rectangle()

```

11
12 ## 직선 그리기
13 cv2.line(image, pt1, pt2, red)
14 cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)
15
16 ## 사각형 그리기
17 cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)
18 cv2.rectangle(image, roi, red, 3, cv2.LINE_8)
19 cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)
20
21 cv2.imshow("Line & Rectangle", image)
22 cv2.waitKey(0)
23 cv2.destroyAllWindows()

```

pt1, pt2 두개의 점을 이용

roi를 이용

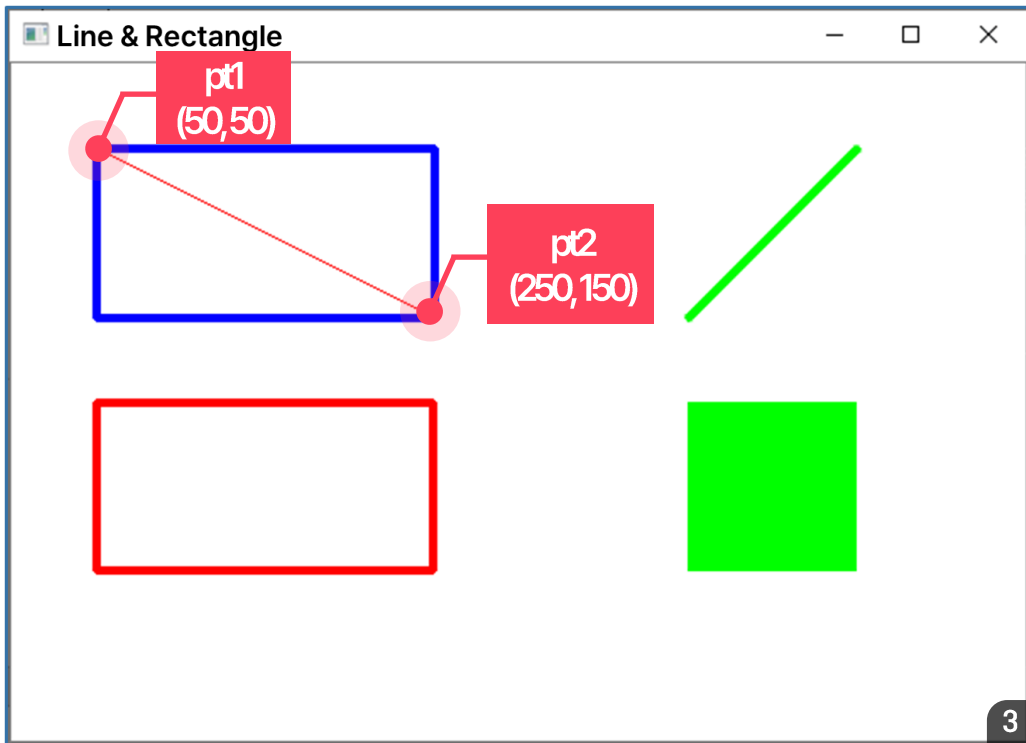
함수안에 직접값을 입력하여 사각형 그리기

→ cv2.FILLED

- 현재 정의된 사각형 안을 채워서 그리고 싶다는 의미

OpenCV 인터페이스

◆ 직선 및 사각형 그리기



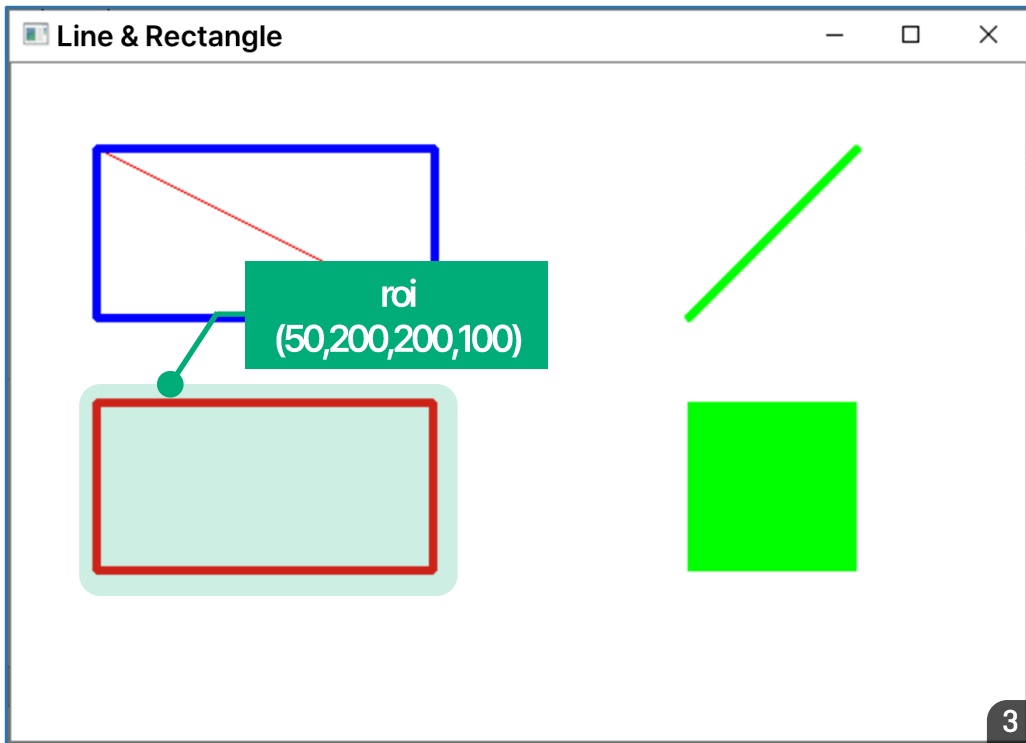
```

01 import numpy as np
02 import cv2
03
04 blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)
05 image = np.zeros((400, 600, 3), np.uint8)
06 image[:] = (255, 255, 255)
07
08 pt1, pt2 = (50, 50), (250, 350)
09 pt3, pt4 = (400, 150), (500, 50)
10 roi = (50, 200, 200, 100)
11
12 ## 직선 그리기
13 cv2.line(image, pt1, pt2, red)
14 cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)
15
16 ## 사각형 그리기
17 cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)
18 cv2.rectangle(image, roi, red, 3, cv2.LINE_8)
19 cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)
20
21 cv2.imshow("Line & Rectangle", image)
22 cv2.waitKey(0)
23 cv2.destroyAllWindows()

```

OpenCV 인터페이스

◆ 직선 및 사각형 그리기



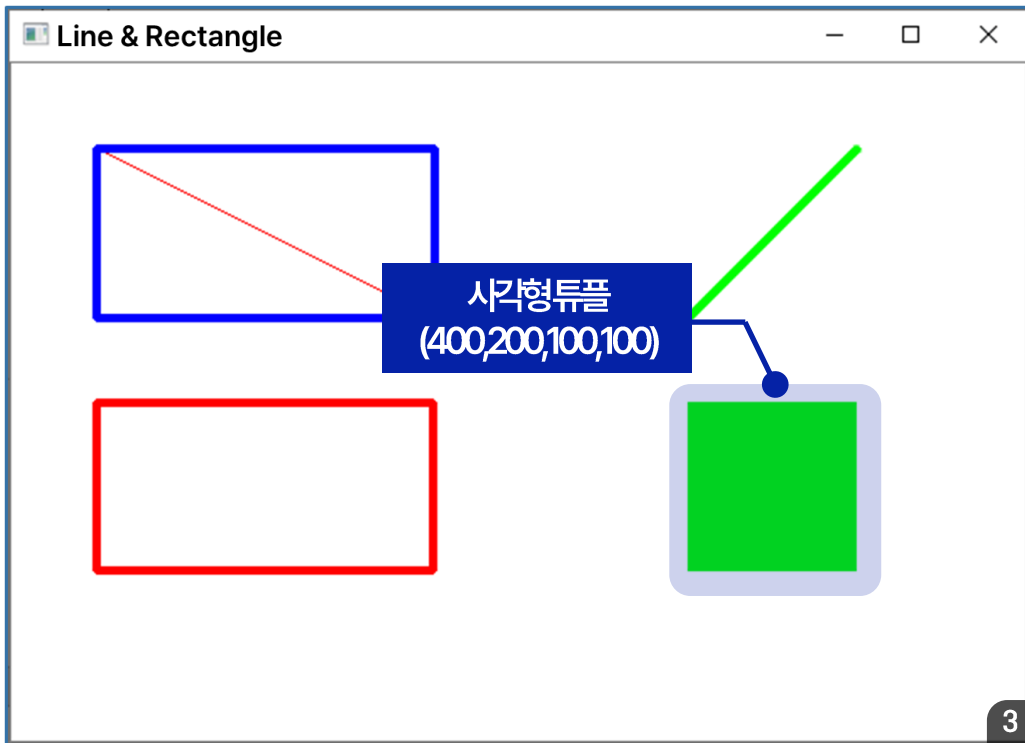
```

01 import numpy as np
02 import cv2
03
04 blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)
05 image = np.zeros((400, 600, 3), np.uint8)
06 image[:] = (255, 255, 255)
07
08 pt1, pt2 = (50, 50), (250, 350)
09 pt3, pt4 = (400, 150), (500, 50)
10 roi = (50, 200, 200, 100)
11
12 ## 직선 그리기
13 cv2.line(image, pt1, pt2, red)
14 cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)
15
16 ## 사각형 그리기
17 cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)
18 cv2.rectangle(image, roi, red, 3, cv2.LINE_8)
19 cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)
20
21 cv2.imshow("Line & Rectangle", image)
22 cv2.waitKey(0)
23 cv2.destroyAllWindows()

```

OpenCV 인터페이스

◆ 직선 및 사각형 그리기



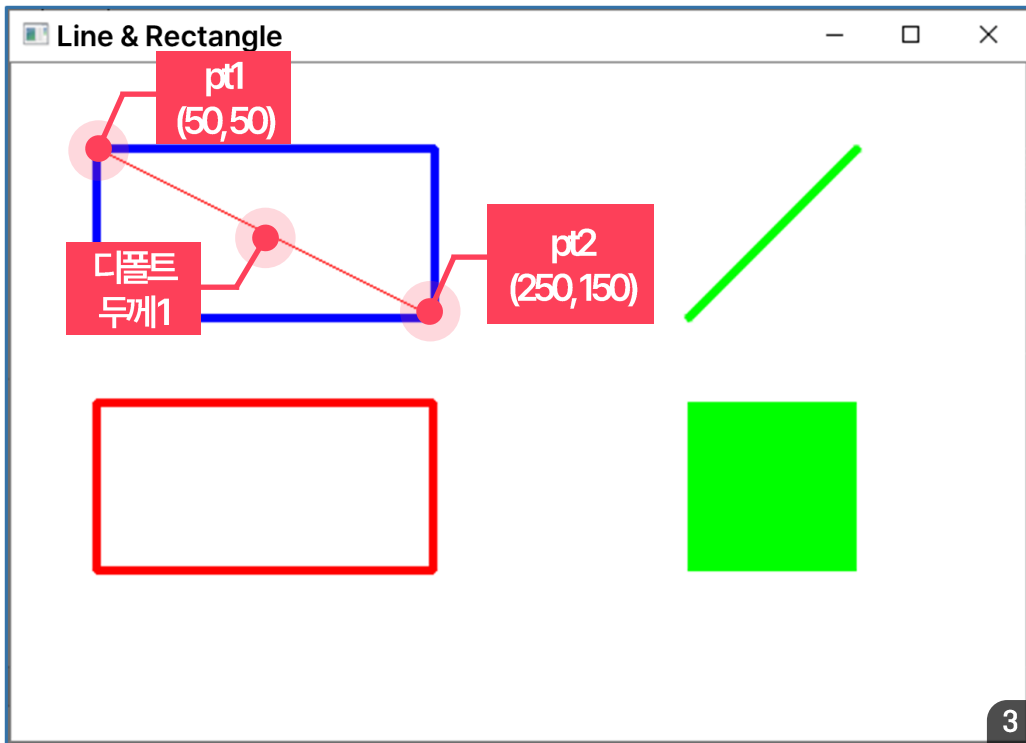
```

01 import numpy as np
02 import cv2
03
04 blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)
05 image = np.zeros((400, 600, 3), np.uint8)
06 image[:] = (255, 255, 255)
07
08 pt1, pt2 = (50, 50), (250, 350)
09 pt3, pt4 = (400, 150), (500, 50)
10 roi = (50, 200, 200, 100)
11
12 ## 직선 그리기
13 cv2.line(image, pt1, pt2, red)
14 cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)
15
16 ## 사각형 그리기
17 cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)
18 cv2.rectangle(image, roi, red, 3, cv2.LINE_8)
19 cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)
20
21 cv2.imshow("Line & Rectangle", image)
22 cv2.waitKey(0)
23 cv2.destroyAllWindows()

```

OpenCV 인터페이스

◆ 직선 및 사각형 그리기



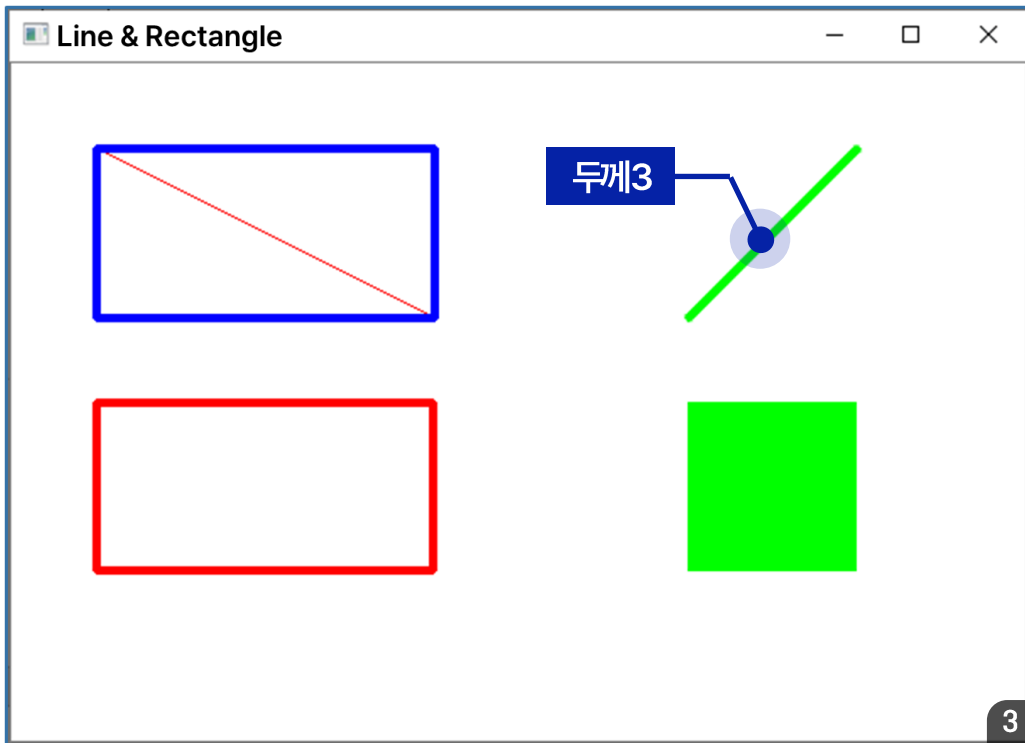
```

01 import numpy as np
02 import cv2
03
04 blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)
05 image = np.zeros((400, 600, 3), np.uint8)
06 image[:] = (255, 255, 255)
07
08 pt1, pt2 = (50, 50), (250, 350)
09 pt3, pt4 = (400, 150), (500, 50)
10 roi = (50, 200, 200, 100)
11
12 ## 직선 그리기
13 cv2.line(image, pt1, pt2, red)
14 cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)
15
16 ## 사각형 그리기
17 cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)
18 cv2.rectangle(image, roi, red, 3, cv2.LINE_8)
19 cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)
20
21 cv2.imshow("Line & Rectangle", image)
22 cv2.waitKey(0)
23 cv2.destroyAllWindows()

```

OpenCV 인터페이스

◆ 직선 및 사각형 그리기



```

01 import numpy as np
02 import cv2
03
04 blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)
05 image = np.zeros((400, 600, 3), np.uint8)
06 image[:] = (255, 255, 255)
07
08 pt1, pt2 = (50, 50), (250, 350)
09 pt3, pt4 = (400, 150), (500, 50)
10 roi = (50, 200, 200, 100)
11
12 ## 직선 그리기
13 cv2.line(image, pt1, pt2, red)
14 cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)
15
16 ## 사각형 그리기
17 cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)
18 cv2.rectangle(image, roi, red, 3, cv2.LINE_8)
19 cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)
20
21 cv2.imshow("Line & Rectangle", image)
22 cv2.waitKey(0)
23 cv2.destroyAllWindows()

```

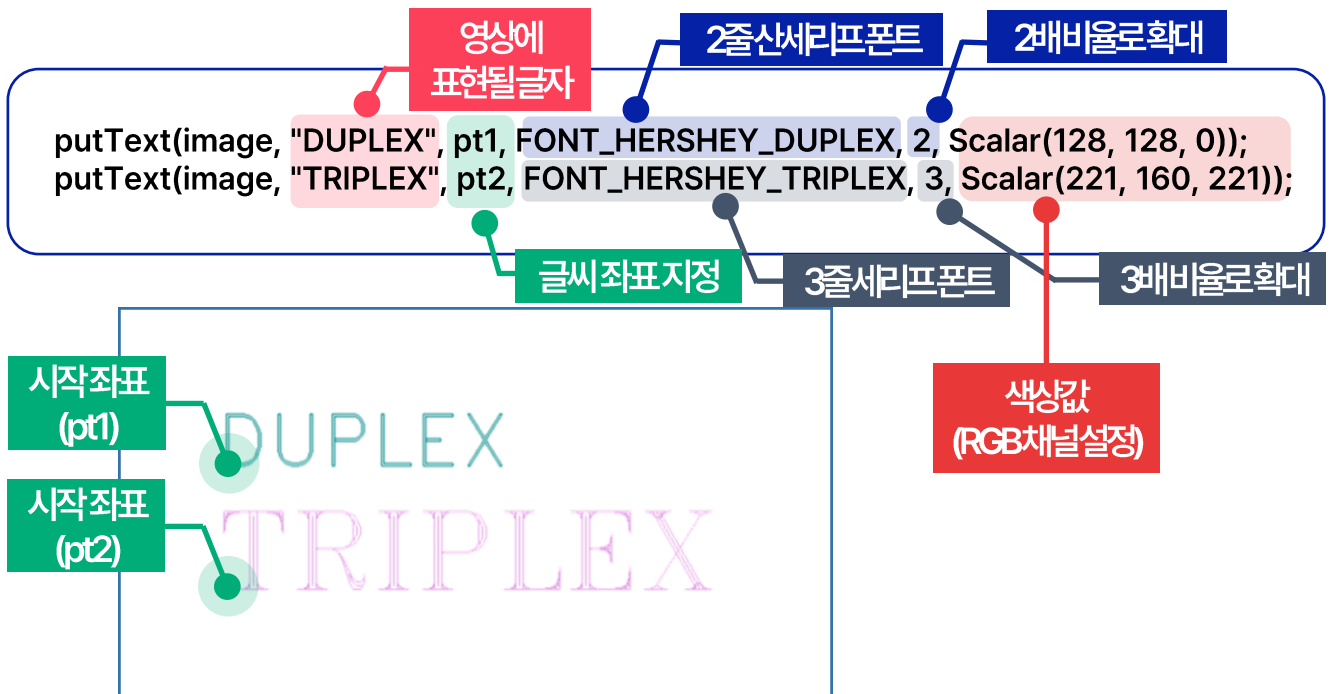
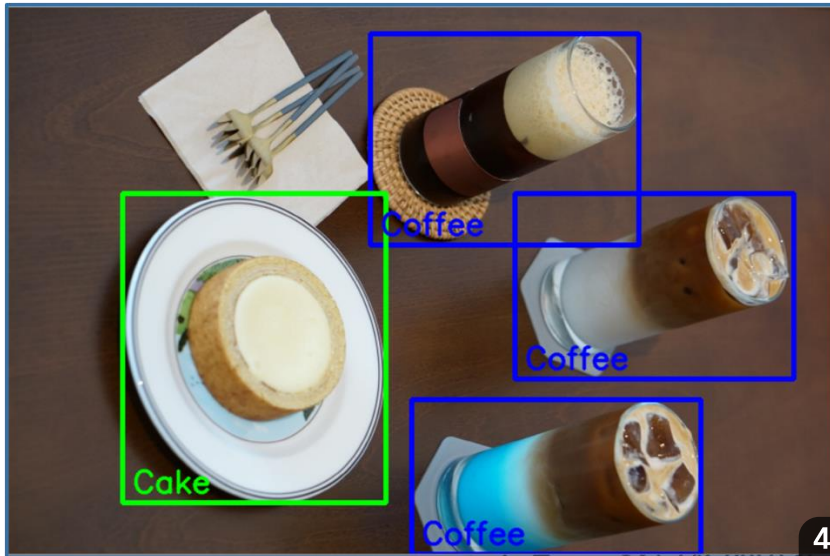
OpenCV 인터페이스

◆ 글자 쓰기 함수

- cv2.putText()

→ 행렬의 특정 위치에 원하는 글자를 써서 영상으로 표시하고 싶을 때 사용

- 영상처리 시 객체가 표시되는 의미를 글씨로 표현



OpenCV 인터페이스

◆ 글자 쓰기 함수

```

01 import numpy as np
02 import cv2
03
04 olive, violet, brown = (128, 128, 0), (221, 160, 221), (42, 42, 165)
05 pt1, pt2 = (50, 230), (50, 310)
06
07 image = np.zeros((350, 500, 3), np.uint8)
08 image.fill(255)
09
10 cv2.putText(image, 'SIMPLEX', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, brown)
11 cv2.putText(image, 'DUPLEX', (50, 130), cv2.FONT_HERSHEY_DUPLEX, 3, olive)
12 cv2.putText(image, 'TRIPLEX', pt1, cv2.FONT_HERSHEY_TRIPLEX, 2, violet)
13 fontFace = cv2.FONT_HERSHEY_PLAIN | cv2.FONT_HERSHEY_ITALIC
14 cv2.putText(image, 'ITALIC', pt2, fontFace, 4, violet)
15
16 cv2.imshow('Put Text', image)
17 cv2.waitKey(0)

```

영상에 표현될 글자

폰트 지정

확대 비율

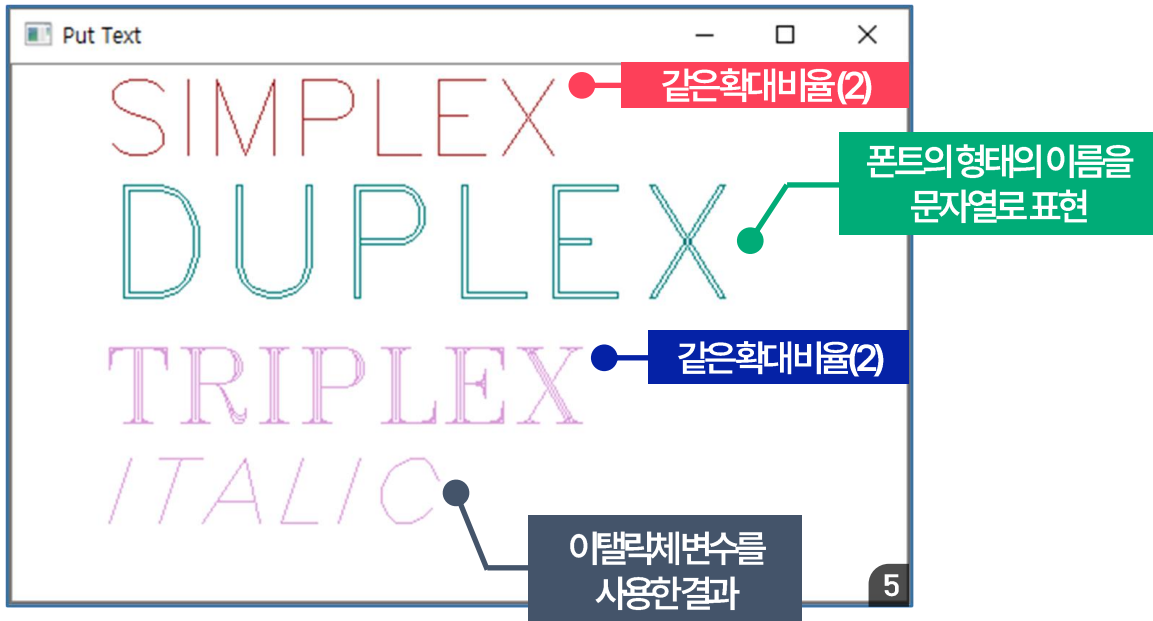
글씨 좌표 지정

색상 지정

글씨를 이탤릭체로 표현시 fontFace 변수를 이용해 폰트를 별도로 정의

OpenCV 인터페이스

◆ 글자 쓰기 함수



```

01 import numpy as np
02 import cv2
03
04 olive, violet, brown = (128, 128, 0), (221, 160, 221), (42, 42, 165)
05 pt1, pt2 = (50, 230), (50, 310)
06
07 image = np.zeros((350, 500, 3), np.uint8)
08 image.fill(255)
09
10 cv2.putText(image, 'SIMPLEX', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 2, brown)
11 cv2.putText(image, 'DUPLEX', (50, 130), cv2.FONT_HERSHEY_DUPLEX, 3, olive)
12 cv2.putText(image, 'TRIPLEX', pt1, cv2.FONT_HERSHEY_TRIPLEX, 2, violet)
13 fontFace = cv2.FONT_HERSHEY_PLAIN | cv2.FONT_ITALIC
14 cv2.putText(image, 'ITALIC', pt2, fontFace, 4, violet)
15
16 cv2.imshow('Put Text', image)
17 cv2.waitKey(0)

```

OpenCV 인터페이스

◆ 원 그리기 함수

- cv2.circle()

→ 중심 좌표를 기준으로 반지름 크기만큼 원을 그림

```
cv2.circle(image, center, 100, blue);
cv2.circle(image, pt1, 50, orange, 2);
```

원의선두께

→ 원의 중심 좌표, 반지름, 선의 색상은 반드시 지정

- center, pt1으로 원의 중심 좌표 정보 지정
- 반지름 정보 100, 50으로 지정
- 색상 정보(blue, orange) 지정

```
01 import numpy as np
02 import cv2
03
04 orange, blue, cyan = (0, 165, 255), (255, 0, 0), (255, 255, 0)
05 white, black = (255, 255, 255), (0, 0, 0)
06 image = np.full((300, 500, 3), white, np.uint8)
07
08 center = (image.shape[1]//2, image.shape[0]//2)
09 pt1, pt2 = (300, 50), (100, 220)
10 shade = (pt2[0] + 2, pt2[1], 2)
11
12 cv2.circle(image, center, 100, blue)
13 cv2.circle(image, pt1, 50, orange, 2)
14 cv2.circle(image, pt2, 70, cyan, -1)
15
16 font = cv2.FONT_HERSHEY_COMPLEX;
17 cv2.putText(image, 'center_blue', center, font, 1.0, blue)
18 cv2.putText(image, 'pt1_orange', pt1, font, 0.8, orange)
19 cv2.putText(image, 'pt2_cyan', shade, font, 1.2, black, 2)
20 cv2.putText(image, 'pt2_cyan', pt2, font, 1.2, cyan, 1)
21
22 cv2.imshow("Draw circles", image)
23 cv2.waitKey(0)
```

원의중심좌표

원의선색상

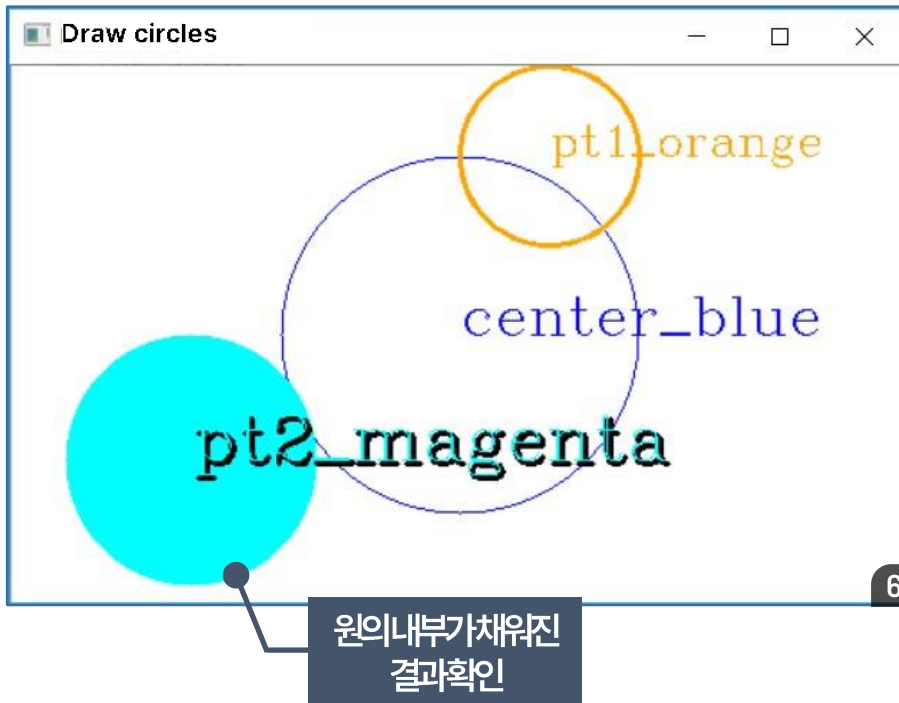
원의선두께

원의반지름

-1설정시사각형FILLED와동일

OpenCV 인터페이스

◆ 원 그리기 함수



```

01 import numpy as np
02 import cv2
03
04 orange, blue, cyan = (0, 165, 255), (255, 0, 0), (255, 255, 0)
05 white, black = (255, 255, 255), (0, 0, 0)
06 image = np.full((300, 500, 3), white, np.uint8)
07
08 center = (image.shape[1]//2, image.shape[0]//2)
09 pt1, pt2 = (300, 50), (100, 220)
10 shade = (pt2[0] + 2, pt2[1], 2)
11
12 cv2.circle(image, center, 100, blue)
13 cv2.circle(image, pt1, 50, orange, 2)
14 cv2.circle(image, pt2, 70, cyan, -1)
15
16 font = cv2.FONT_HERSHEY_COMPLEX;
17 cv2.putText(image, 'center_blue', center, font, 1.0, blue)
18 cv2.putText(image, 'pt1_orange', pt1, font, 0.8, orange)
19 cv2.putText(image, 'pt2_cyan', shade, font, 1.2, black, 2)
20 cv2.putText(image, 'pt2_cyan', pt2, font, 1.2, cyan, 1)
21
22 cv2.imshow("Draw circles", image)
23 cv2.waitKey(0)

```

영상파일 처리하기

영상파일처리하기

◆ 영상파일 처리하기

- 영상파일 → 행렬 → 확인
 - 사용되는 OpenCV 함수는 imread와 imwrite
 - imread는 이미지를 읽는 함수
 - imwrite는 이미지를 저장하는 함수
- cv2.imread()
 - 영상파일을 행렬로 저장
 - 확장자에 따라 JPG, BMP, PNG, TIF, PPM 등의 영상파일 포맷 확인 가능

◆ imread 함수

- 영상의 컬러 타입을 결정하는 옵션

| 옵션 | 값 | 설명 |
|----------------------|----|--|
| cv.IMREAD_UNCHANGED | -1 | 입력 파일에 정의된 타입의 영상을 그대로 반환 (알파(alpha) 채널 포함) |
| cv2.IMREAD_GRAYSCALE | 0 | 명암도(grayscale) 영상으로 변환하여 반환 |
| cv2.IMREAD_COLOR | 1 | 컬러 영상으로 변환하여 반환 |
| cv2.IMREAD_ANYDEPTH | 2 | 입력 파일에 정의된 깊이(depth)에 따라 16비트/32비트 영상으로 변환, 설정되지 않으면 8비트 영상으로 변환 |
| cv2.IMREAD_ANYCOLOR | 4 | 입력 파일에 정의된 타입의 영상을 반환 |

◆ imwrite 함수

- cv2.imwrite()
 - 행렬을 영상파일로 저장
 - 확장자에 따라서 JPG, BMP, PNG, TIF, PPM 등의 영상파일 포맷 저장 가능

정리하기

- OpenCV 인터페이스

- 그리기 함수
- 글자 쓰기
- 원 그리기

- 영상파일 처리하기