

CH. 5. 영상처리와 컴퓨터 비전

Section 01 라즈베리파이 카메라 모듈

□ 라즈베리파이 카메라 모듈 V3 특징

- Raspberry Pi는 6개의 새로운 카메라 모듈을 출시했다.
- 주요 특징은 이제 표준 카메라 모듈에 자동 초점이 제공되고 1200만 화소로 업그레이드되었으며 새로운 120도 광각도 출시된다는 것이다.
- Raspberry Pi는 업데이트된 카메라인 Camera Module 3(일명 Camera v3 또는 Camera Module v3)을 출시했으며 MSRP는 표준 버전이 \$25, 광각 버전이 \$35이다.
- 새로운 모듈은 더 작은 센서 온 보드 폼 팩터를 유지하면서 고품질 카메라의 12MP에 필적하는 더 많은 픽셀을 제공한다.
- 이 카메라의 새로운 특징은 자동 초점이다.
- 라즈베리파이 카메라는 라즈베리파이의 첫 번째 공식 액세서리였다.
- 원래 5MP 모델은 2016년에 v2로 업데이트되어 8MP로 향상되었다.
- 그런 다음 카메라는 2020년에 12MP 고품질 카메라로 좀 더 향상되었다.

Section 01 라즈베리파이 카메라 모듈

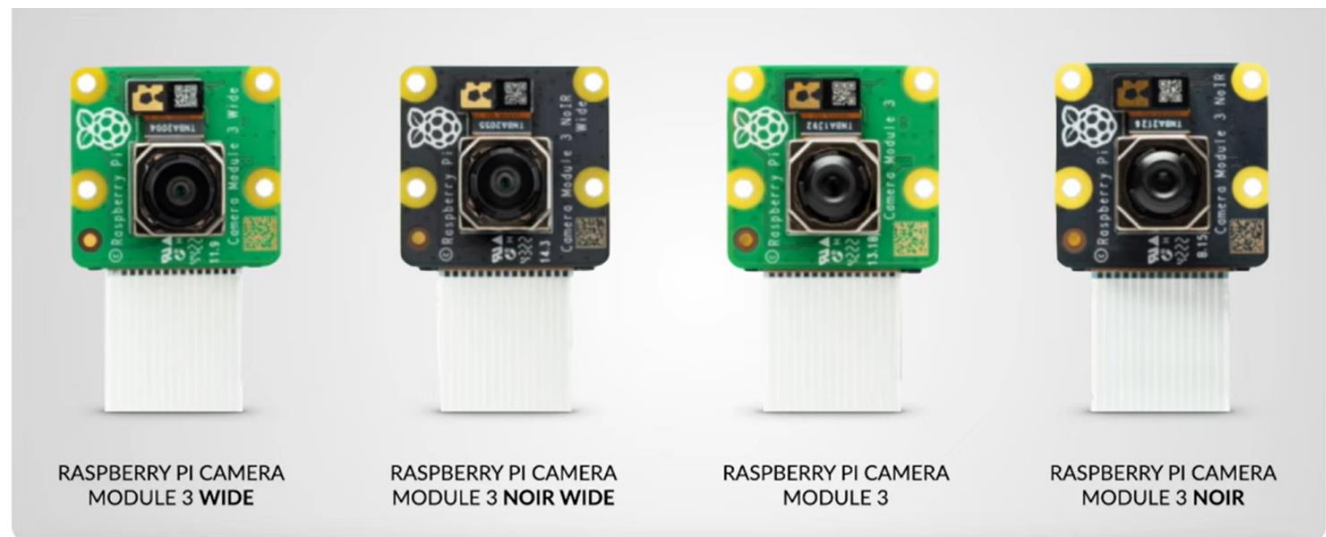
□ Raspberry Pi 카메라 v3 사양

감지기	소니 IMX708
광학 크기	7.4mm 센서 대각선
센서 해상도	1190만 화소
스틸 해상도	4608 x 2592픽셀
비디오 모드	1080p50, 720p100, 480p120
자동 초점 시스템	위상 검출 자동 초점
초점 범위	무한대에 10CM
초점 거리	4.74mm
치수	25 x 23.9 x 11.5mm(75도)
	25 x 23.9 x 12.4mm(120도 "광각")

Section 01 라즈베리파이 카메라 모듈

□ Raspberry Pi 카메라 v3 사용

- 카메라를 제어하는 두 가지 주요 방법이 있다.
- libcamera 라이브러리(raspistill 및 raspivid를 대체함) 및 Picamera2(Picamera를 생성한 장기 실행 커뮤니티에 대한 사내 업데이트).
- Raspberry Pi 4의 전원이 꺼진 상태에서 카메라 케이블을 삽입하고 제자리에 고정했다.
- 카메라는 이전 모델과 동일한 커넥터를 사용하므로 어댑터 케이블을 통해 Raspberry Pi Zero 2 W를 포함한 모든 범위의 Raspberry Pi 보드와 함께 사용할 수 있다.



Section 02 영상 처리와 컴퓨터 비전의 개요

□ 영상 처리

- 영상 처리는 말 그대로 영상을 처리한다는 의미이다.
- 카메라로 찍은 사진 또는 영상에 여러 가지 연산을 가해서 원하는 결과를 새롭게 얻어내는 과정을 영상 처리 또는 이미지 프로세싱(image processing) 이라고 한다.
- 대부분 영상 처리의 목적은 더 좋은 품질의 영상을 얻으려는 것이다.
- 몇 가지 예를 들면 다음과 같다.
 - 영상(화질) 개선: 사진이나 동영상이 너무 어둡거나 밝아서 화질을 개선하는 과정
 - 영상 복원: 오래되어 빛바랜 옛날 사진이나 영상을 현대적인 품질로 복원하는 과정
 - 영상 분할: 사진이나 영상에서 원하는 부분만 오려내는 과정

Section 02 영상 처리와 컴퓨터 비전의 개요

□ 컴퓨터 비전

- 컴퓨터 비전은 영상처리 개념을 포함하는 좀 더 큰 포괄적인 의미이다.
- 영상처리가 원본 영상을 사용자가 원하는 새로운 영상으로 바꿔 주는 기술이라면 컴퓨터 비전은 영상에서 의미있는 정보를 추출해 주는 기술을 의미한다.
- 예를 들면 다음과 같다.
 - 객체 검출(object detection): 영상속에 원하는 대상이 어디에 있는지 검출
 - 객체 추적 (object tracking): 영상 속 관심 있는 피사체가 어디로 움직이는지 추적
 - 객체 인식 (object recognition): 영상 속 피사체가 무엇인지 인식

Section 02 영상 처리와 컴퓨터 비전의 개요

□ 컴퓨터 비전

- 컴퓨터 비전 작업을 하기 전에 영상처리 작업을 하는 경우가 많다.
- 만약 영상에서 객체를 인식하려고 하는데, 화질이 나쁘면 당연히 인식이 잘 되지 않을 것이다.
- 그래서 먼저 화질 개선 작업을 해야 할 수도 있다.
- 하지만, 컴퓨터 비전의 전처리 작업에 화질 개선 작업만 있는 것은 아니다.
- 오히려 고화질 영상은 물체를 인식하는 데 불필요하게 연산이 많이 필요하므로 영상을 단순화하는 작업이 필요할 수도 있다.



Section 02 영상 처리와 컴퓨터 비전의 개요

□ OpenCV의 개요

- OpenCV는 오픈 소스 컴퓨터 비전 라이브러리 (Open Source Computer Vision Library)를 줄여 쓴 말이다.
- OpenCV는 영상 처리와 컴퓨터 비전 프로그래밍 분야의 가장 대표적인 라이브러리이다.
- 예전에는 프로그래밍 영역 중에서도 영상 처리나 컴퓨터 비전 분야는 대학원에서도 본격적으로 접하게 될 만큼 비교적 전문적인 분야였다.
- 매우 복잡한 수학적 알고리즘을 C/C++ 언어로 구현해야 했으므로 체계적인 수학 지식과 뛰어난 프로그래밍 실력을 갖추고 있지 않으면 이해하기 어려웠기 때문이다.
- 하지만, 이제는 OpenCV와 같은 훌륭한 라이브러리가 있어 기초적인 수학 지식만으로도 이미 구현된 알고리즘을 손쉽게 사용할 수 있게 된 데다가, 메모리 주소 하나 하나를 직접 다뤄야 했던 어렵고 복잡한 C 언어가 아닌 비교적 배우기 쉽고 빠르게 구현할 수 있는 파이썬 언어로 OpenCV를 사용할 수 있게 되면서 영상 처리와 컴퓨터 비전분야의 문턱이 무척 낮아졌다.

Section 03 기본 입출력

□ 이미지 읽기

- OpenCV를 사용해서 이미지를 읽고 화면에 표시하는 가장 간단한 코드는 아래와 같다.

```
1  import cv2
2
3  img_file = “./img/lena.jpg”
4  img = cv2.imread(img_file)
5
6  if img is not None:
7      cv2.imshow('IMG', img)
8      cv2.waitKey()
9      cv2.destroyAllWindows()
10 else:
11     print(“No image file.”)
```

Section 03 기본 입출력

□ 이미지 읽기

- 예제에서 사용한 함수는 다음과 같다.
 - `img = cv2.imread(file_name [, mode_flag])` : 파일로부터 이미지 읽기
 - file name: **이미지 경로, 문자열**
 - mode_flag=cv2.IMREAD_COLOR: **읽기 모드 지정**
 - cv2.IMREAD_COLOR: **컬러(BGR) 스케일로 읽기, 기본 값**
 - cv2.IMREAD_UNCHANGED: **파일 그대로 읽기**
 - cv2.IMREAD_GRAYSCALE: **그레이(흑백) 스케일로 읽기**
 - img : **읽은 이미지, NumPy 배열**
 - `cv2.imshow(title, img)` : **이미지를 화면에 표시**
 - title: **창 제목, 문자열**
 - img: **표시할 이미지, Numpy 배열**

Section 03 기본 입출력

□ 이미지 읽기

- 예제에서 사용한 함수는 다음과 같다.
 - `key = cv2.waitKey([delay])`: 키보드 입력 대기
 - `delay=0`: 키보드 입력을 대기할 시간(ms), 0: 무한대 (기본값)
 - `key`: 사용자가 입력한 키 값, 정수
 - -1: 대기시간 동안 키 입력 없음
- `cv2.imread()` 함수는 파일로부터 이미지를 읽을 때 모드를 지정할 수 있다.
- 별도로 모드를 지정하지 않으면 3개 채널 (B, G, R)로 구성된 컬러 스케일로 읽어들이지만, 필요에 따라 그레이 스케일 또는 파일에 저장된 스케일 그대로 읽을 수 있다.
 - `img = cv2.imread(file_name, cv2.IMREAD_GRAYSCALE)`

Section 03 기본 입출력

□ 이미지 읽기

- 읽기 모드를 그레이 스케일로 지정하면 원래의 파일이 컬러 이미지일지라도 그레이 스케일로 읽는다.

```
1  import cv2
2
3  img_file = "./img/lena.jpg"
4  img = cv2.imread(img_file, cv2.IMREAD_GRAYSCALE)
5
6  if img is not None:
7      cv2.imshow('IMG', img)
8      cv2.waitKey()
9      cv2.destroyAllWindows()
10 else:
11     print("No image file.")
```

Section 03 기본 입출력

□ 이미지 저장하기

- OpenCV로 읽어들이는 이미지를 다시 파일로 저장하는 함수는 `cv2.imwrite()` 이다.
 - `cv2.imwrite(file_path, img)` : 이미지를 파일에 저장
 - `file_path`: 저장할 파일 경로 이름, 문자열
 - `img`: 저장할 영상, NumPy 배열

Section 03 기본 입출력

□ 이미지 저장하기

- 다음 예제는 컬러 이미지 파일을 그레이 스케일로 읽어 들어서 파일로 저장하는 예제이다.

```
1 import cv2
2
3 img_file = "./img/model3.jpg"
4 save_file = "./img/model3_gray.jpg"
5
6 img = cv2.imread(img_file, cv2.IMREAD_GRAYSCALE)
7 cv2.imshow(img_file, img)
8 cv2.imwrite(save_file, img)
9 cv2.waitKey()
10 cv2.destroyAllWindows()
```

Section 03 기본 입출력

□ 동영상 및 카메라 프레임 읽기

- OpenCV는 동영상 파일이나 컴퓨터에 연결한 카메라 장치로부터 연속된 이미지 프레임을 읽을 수 있는 API를 제공한다.
- 다음은 동영상 파일이나 연속된 이미지 프레임을 읽을 수 있는 API의 주요내용이다.
 - `cap = cv2.VideoCapture(file_path 또는 index)` : 비디오 캡처 객체 생성자
 - `file_path`: 동영상 파일 경로
 - `index`: 카메라 장치 번호, 0부터 순차적으로 증가(0, 1, 2, ...)
 - `cap`: VideoCapture 객체
 - `ret = cap.isOpened()` : 객체 초기화 확인
 - `ret`: 초기화 여부, True/ False
 - `ret, img = cap.read()` : 영상 프레임 읽기
 - `ret`: 프레임 읽기 성공 또는 실패 여부, True/False
 - `img`: 프레임 이미지, NumPy 배열 또는 None

Section 03 기본 입출력

□ 동영상 및 카메라 프레임 읽기

- OpenCV는 동영상 파일이나 컴퓨터에 연결한 카메라 장치로부터 연속된 이미지 프레임을 읽을 수 있는 API를 제공한다.
- 다음은 동영상 파일이나 연속된 이미지 프레임을 읽을 수 있는 API의 주요내용이다.
 - `cap.set(id, value)` : 프로퍼티 변경
 - `cap.get(id)`: 프로퍼티 확인
 - `cap.release()` : 캡처 자원 반납
- 동영상 파일이나 컴퓨터에 연결한 카메라 장치로부터 영상 프레임을 읽기 위해서는 `cv2.VideoCapture()` 생성자 함수를 사용하여 객체를 생성해야 한다.
- 비디오 캡처 객체의 `set()`, `get()` 함수를 이용하면 여러 가지 속성을 얻거나 지정할 수 있으며, 프로그램을 종료하기 전에 `release()` 함수를 호출해서 자원을 반납해야 한다.

Section 03 기본 입출력

□ 동영상 파일 읽기

- 다음 예제는 동영상 파일을 읽기 위한 간단한 코드이다.

```
1  import cv2
2
3  video_file = "./img/big_buck_bunny.avi"
4
5  cap = cv2.VideoCapture(video_file)
6  if cap.isOpened():
7      while True:
8          ret, img = cap.read()
9          if ret:
10             cv2.imshow(video_file, img)
11             cv2.waitKey(25)
12         else:
13             break
14
```

Section 03 기본 입출력

□ 동영상 파일 읽기

- 다음 예제는 동영상 파일을 읽기 위한 간단한 코드이다.

```
14  
15     else:  
16         print("can't open video.")  
17     cap.release()  
18     cv2.destroyAllWindows()
```

Section 03 기본 입출력

□ 카메라 프레임 읽기

- 카메라로 프레임을 읽기 위해서는 `cv2.VideoCapture()` 함수에 동영상 파일 경로 대신에 카메라 장치 인덱스 번호를 정수로 지정해 주면 된다.
- 카메라 장치 인덱스 번호는 0부터 시작해서 1씩 증가한다. 만약 카메라가 하나만 연결되어 있으면 당연히 0번 인덱스를 사용하면 된다.
- 이 부분을 제외하고는 나머지 코드는 동영상 파일을 읽는 것과 거의 똑같다.

Section 03 기본 입출력

□ 카메라 프레임

```
1 import cv2
2
3 cap = cv2.VideoCapture(0)
4 if cap.isOpened():
5     while True:
6         ret, img = cap.read()
7         if ret:
8             cv2.imshow('camera', img)
9             if cv2.waitKey(1) != -1:
10                 break
11             else:
12                 print('no frame')
13                 Break
14 else:
15     print("can't open camer.")
16 cap.release()
17 cv2.destroyAllWindows()
```

Section 03 기본 입출력

□ 카메라 비디오 속성 제어

- 캡처 객체에는 영상 또는 카메라의 여러 가지 속성을 확인하고 설정할 수 있는 `get(id)`, `set(id, value)` 함수를 제공한다.
- 속성을 나타내는 아이디는 `cv2.CAP_PROP_FRAME_`으로 시작하는 상수로 정의되어 있으며, 그 수가 너무 많아서 이 책에서 모두 다룰 수는 없다.
- 여기서는 주요 속성만을 다루므로 나머지 속성에 대한 명세는 API 문서를 참조하기 바란다.
 - 속성 ID: '`cv2.CAP_PROP_`' 로 시작하는 상수
 - `cv2.CAP_PROP_FRAME_WIDTH` 프레임 폭
 - `cv2.CAP_PROP_FRAME_HEIGHT`: 프레임 높이
 - `cv2.CAP_PROP_FPS`: 초당 프레임 수
 - `cv2.CAP_PROP_POS_MSEC`: 동영상 파일의 프레임 위치 (ms)
 - `cv2.CAP_PROP_POS_AVI_RATIO`: 동영상 파일의 상대 위치 (0 : 시작, 1 : 끝)
 - `cv2.CAP_PROP_FOURCC`: 동영상 파일 코덱 문자
 - `cv2.CAP_PROP_AUTOFOCUS`: 카메라 자동 초점 조절
 - `cv2.CAP_PROP_ZOOM`: 카메라 줌

Section 03 기본 입출력

□ 카메라 비디오 속성 제어

- 각 속성 아이디를 `get()`에 전달하면 해당 속성의 값을 구할 수 있고, `set()` 함수에 아이디와 값을 함께 전달하면 값을 지정할 수 있다.
- 앞서 동영상 파일을 재생하는 실습을 할 때 적절한 FPS에 따라 지연 시간을 설정해야 하지만, FPS를 대충 짐작하거나 별도의 플레이어를 활용해서 알아내야 했다.
- 비디오 속성 중에 FPS를 구하는 상수는 `cv2.CAP_PROP_FPS`이고 이것으로 동영상의 FPS를 구하고 다음과 같이 적절한 지연 시간을 계산해서 지정할 수 있다.
 - `fps = cap.get (cv2.CAP_PROP_FPS)` # 초당 프레임 수 구하기
 - `delay = int (1000/fps)` # 지연 시간 구하기
- `cv2.waitKey()` 함수에 전달하는 지연 시간은 밀리초($1/1000$) 단위이고 정수만 전달할 수 있으므로 1 초를 1000으로 환산해서 계산한 뒤 정수형으로 바꾼다.

Section 03 기본 입출력

□ 카메라 비디오 속성 제어

- FPS에 맞는 지연 시간을 지정해서 완성한 코드는 다음 예제와 같다.

```
1 import cv2
2
3 video_file = "./img/big_buck_bunny.avi"
4
5 cap = cv2.VideoCapture(video_file)
6 if cap.isOpened():
7     fps = cap.get(cv2.CAP_PROP_FPS)
8     delay = int(1000/fps)
9     print(f"FPS: {fps}, Delay: {delay}ms")
10
```

Section 03 기본 입출력

□ 카메라 비디오 속성 제어

- FPS에 맞는 지연 시간을 지정해서 완성한 코드는 다음 예제와 같다.

```
10
11     while True :
12         ret, img = cap.read()
13         if ret:
14             cv2.imshow(video_file, img)
15             cv2.waitKey(delay)
16         else:
17             break
18     else:
19         print("can't open video")
20     cap.release()
21     cv2.destroyAllWindows()
```


Section 03 기본 입출력

□ 카메라 비디오 속성 제어

- 아쉽게도 FPS 속성을 카메라 장치로부터 읽을 때는 대부분 정상적인 값을 가져오지 못한다.
- 이번엔 다른 속성을 하나 더 살펴보자.
- 카메라로부터 읽은 영상이 너무 고화질인 경우 픽셀 수가 많아 연산하는 데 시간이 많이 걸리는 경우가 있다.
- 이 때 프레임의 폭과 높이를 제어해서 픽셀 수를 줄일 수 있다.
- 프레임의 폭과 높이 속성 아이디 상수는 `cv2.CAP_PROP_FRAME_WIDTH`와 `cv2.CAP_PROP_FRAME_HEIGHT` 이다.

Section 03 기본 입출력

□ 카메라 비디오 속성 제어

- 기본 영상 프레임의 폭과 높이를 구해서 출력하고 새로운 크기를 지정하는 코드는 다음 예제와 같다.

```
1 import cv2
2
3 cap = cv2.VideoCapture(0)
4 width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
5 height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
6 print("Original width: %d, height: %d" %(width, height))
7
8 cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
9 cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)
10 width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
11 height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
12 print("Resized width: %d, height: %d" %(width, height))
13
```

Section 03 기본 입출력

□ 카메라 비디오 속성 제어

- 기본 영상 프레임의 폭과 높이를 구해서 출력하고 새로운 크기를 지정하는 코드는 다음 예제와 같다.

```
13
14 if cap.isOpened() :
15     while True:
16         ret, img = cap.read()
17         if ret :
18             cv2.imshow("camera", img)
19             if cv2.waitKey(1) != -1 :
20                 break
21         else:
22             print("no frame!")
23             break
24     else:
25         print("can't open camera!")
26 cap.release()
27 cv2.destroyAllWindows()
```

Section 03 기본 입출력

□ 카메라 파일 저장하기

- 카메라나 동영상 파일을 재생하는 도중 특정한 프레임만 이미지로 저장하거나 특정 구간을 동영상 파일로 저장할 수도 있다.
- 한 개의 특정 프레임만 파일로 저장하는 방법은 `cv2.imwrite()` 함수를 그대로 사용하면 된다.
- 다음 예제는 카메라로부터 프레임을 표시하다가 아무 키나 누르면 해당 프레임을 파일로 저장하는 코드이다.

```
1 import cv2
2
3 cap = cv2.VideoCapture(0)
4 if cap.isOpened() :
5     while True:
6         ret, frame = cap.read()
7         if ret :
8             cv2.imwrite('camera', frame)
```

Section 03 기본 입출력

□ 카메라 파일 저장하기

- 카메라나 동영상 파일을 재생하는 도중 특정한 프레임만 이미지로 저장하거나 특정 구간을 동영상 파일로 저장할 수도 있다.
- 한 개의 특정 프레임만 파일로 저장하는 방법은 `cv2.imwrite()` 함수를 그대로 사용하면 된다.
- 다음 예제는 카메라로부터 프레임을 표시하다가 아무 키나 누르면 해당 프레임을 파일로 저장하는 코드이다.

```
9         if cv2.waitKey(1) != -1 :
10             cv2.imwrite('photo.jpg', frame)
11         else:
12             print('no frame!')
13             break
14     else:
15         print("no camera!")
16     cap.release()
17     cv2.destroyAllWindows()
```

Section 03 기본 입출력

□ 카메라 파일 저장하기

- 예제를 실행하면 카메라로부터 촬영한 영상이 화면에 나오는데, 카메라를 보고 자세를 취하면서 키보드의 아무 키나 누르면 코드를 실행한 디렉터리에 photo.jpg로 사진이 저장된다.
- 하나의 프레임이 아닌 여러 프레임을 동영상으로 저장하려고 할 때는 `cv2.VideoWriter()` 라는 새로운 API가 필요하다.
 - `writer = cv2.VideoWriter(file_path, fourcc, fps, (width, height))` : 비디오 저장 클래스 생성자 함수
 - `file_path`: 비디오 파일 저장 경로
 - `fourcc`: 비디오 인코딩 형식 4글자
 - `fps`: 초당프레임 수
 - `(width, height)` : 프레임 폭과 프레임 높이
 - `Writer`: 생성된 비디오 저장 객체
 - `writer.write(frame)`: 프레임 저장
 - `frame`: 저장할 프레임, NumPy 배열

Section 03 기본 입출력

□ 카메라 파일 저장하기

- 하나의 프레임이 아닌 여러 프레임을 동영상으로 저장하려고 할 때는 `cv2.VideoWriter()` 라는 새로운 API가 필요하다.
 - `writer.set(id, value)`: 프로퍼티 변경
 - `writer.get(id)`: 프로퍼티 확인
 - `ret = writer.fourcc(c1, c2, c3, c4)`: fourcc 코드 생성
 - `c1, c2, c3, c4`: 인코딩 형식 4글자, 'MJPG', 'DIVX' 등
 - `ret`: fourcc 코드
 - `cv2.VideoWriter_fourcc(c1, c2, c3, c4)`: `cv2.VideoWriter.fourcc()`와 동일

Section 03 기본 입출력

□ 카메라 파일 저장하기

- `cv2.VideoWriter()` 생성자 함수에 저장할 파일 이름과 인코딩 포맷 문자, fps, 프레임 크기를 지정해서 객체를 생성하고 `write()` 함수로 프레임을 파일에 저장하면 된다.

```
1 import cv2
2
3 cap = cv2.VideoCapture(0)
4 if cap.isOpened() :
5     file_path = './record.avi'
6     fps = 25.40
7     fourcc = cv2.VideoWriter_fourcc(*'DIVX')
8     width = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
9     height = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
10    size = (int(width), int(height))
11    out = cv2.VideoWriter(file_path, fourcc, fps, size)
```


Section 03 기본 입출력

□ 카메라 파일 저장하기

- cv2.VideoWriter() 생성자 함수에 저장할 파일 이름과 인코딩 포맷 문자, fps, 프레임 크기를 지정해서 객체를 생성하고 write() 함수로 프레임을 파일에 저장하면 된다.

```
12 while True :
13     ret, frame = cap.read()
14     if ret :
15         cv2.imshow('camera-recording', frame)
16         out.write(frame)
17         if cv2.waitKey(int(1000/fps)1) != -1 :
18             break
19         else :
20             print("no frame!")
21             break
22     out.release()
23 else :
24     print("can't open camera!")
25 cap.release()
26 cv2.destroyAllWindows()
```

Section 03 기본 입출력

□ 창 관리

- 한 개 이상의 이미지를 여러 창에 띄우거나 각 창에 키보드와 마우스 이벤트를 처리하려면 창을 관리하는 기능이 필요하다.
- 다음은 OpenCV가 제공하는 창 관리 관련 API들을 요약한 것이다.
- `cv2.namedWindow(title [, option])` : 이름을 갖는 창 열기
 - title: 창 이름, 제목 줄에 표시
 - option: 창 옵션, 'cv2.WINDOW_' 로 시작
 - `cv2.WINDOW_NORMAL`: 임의의 크기, 사용자 창 크기 조정 가능
 - `cv2.WINDOW_AUTOSIZE` : 이미지와 같은 크기, 창 크기 재조정 불가능
- `cv2.moveWindow(title, x, y)`: 창 위치 이동
 - title: 위치를 변경할 창의 이름
 - x, y: 이동할 창의 위치

Section 03 기본 입출력

□ 창 관리

- 한 개 이상의 이미지를 여러 창에 띄우거나 각 창에 키보드와 마우스 이벤트를 처리하려면 창을 관리하는 기능이 필요하다.
- 다음은 OpenCV가 제공하는 창 관리 관련 API들을 요약한 것이다.
- `cv2.resizeWindow (title, width, height)`: 창 크기 변경
 - title: 크기를 변경할 창의 이름
 - width, height: 크기를 변경할 창의 폭과 높이
- `cv2.destroyWindow(title)`: 창 닫기
 - title: 닫을 대상 창 이름
- `cv2.destroyAllWindows()`: 열린 모든 창 닫기

Section 03 기본 입출력

□ 창 관리

- 다음 예제는 창 관리 함수를 이용하는 예제이다.

```
1  import cv2
2
3  file_path = './img/model3.jpg'
4  img = cv2.imread(file_path)
5  img_gray = cv2.imread(file_path, cv2.IMREAD_GRAYSCALE)
6
7  cv2.namedWindow('origin', cv2.WINDOW_AUTOSIZE)
8  cv2.namedWindow('gray', cv2.WINDOW_NORMAL)
9
10 cv2.imshow('origin', img)
11 cv2.imshow('gray', img_gray)
12
```

Section 03 기본 입출력

□ 창 관리

- 다음 예제는 창 관리 함수를 이용하는 예제이다.

```
13 cv2.moveWindow('origin', 0, 0)
14 cv2.moveWindow('gray', 100, 100)
15
16 cv2.waitKey(0)
17 cv2.resizeWindow('origin', 200, 200)
18 cv2.resizeWindow('gray', 100, 100)
19
20 cv2.waitKey(0)
21 cv2.destroyWindow('gray')
22
23 cv2.waitKey(0)
24 cv2.destroyAllWindows()
```

Section 04 그림 그리기

□ 직선 그리기

- 이미지에 직선을 그리는 함수는 `cv2.line()` 이다.
- `cv2.line(img, start, end, color [, thickness, lineType])`
 - `img`: 그림 그릴 대상 이미지, NumPy 배열
 - `start`: 선 시작 지점 좌표(x, y)
 - `end`: 선 끝 지점 좌표(x, y)
 - `color`: 선 색상, (Blue, Green, Red), 0-255
 - `thickness=1`: 선 두께
 - `lineType`: 선 그리기 형식
 - `cv2.LINE_4`: 4 연결 선 알고리즘
 - `cv2.LINE_8`: 8 연결 선 알고리즘
 - `cv2.LINE_AA`: 안티에일리어싱 (antialiasing, 계단 현상 없는 선)

Section 04 그림 그리기

□ 직선 그리기

- `img` 이미지에 `start` 지점에서 `end` 까지 선을 그린다.
- `color`는 선의 색상을 표현하는 것으로 0~255 사이의 값 3개로 구성해서 표현한다.
- 각 숫자는 파랑, 초록, 빨강(BGR) 순서이며, 이 색상을 섞어서 다양한 색상을 표현한다.
- 일반적으로 웹에서 사용하는 RGB 순서와 반대라는 것이 특징이다.
- `thickness`는 선의 두께를 픽셀 단위로 지시하는데, 생략하면 픽셀이 적용된다.
- `lineType`은 선을 표현하는 방식을 나타내는 것으로 사선을 표현하거나 두꺼운 선의 끝을 표현할 때 픽셀에 따른 계단 현상을 최소화하기 위한 알고리즘을 선택한다.
- `cv2.LINE`으로 시작하는 3개의 상수를 선택할 수 있다.
 - `cv2.LINE_4`와 `cv2.LINE_8`은 각각 브레젠햄(Bresenham) 알고리즘의 4연결, 8연결을 의미하고 `cv2.LINE_AA`는 가우시안 필터를 이용한다.

Section 04 그림 그리기

□ 직선 그리기

- 다음 예제에서 다양한 선을 그려보면서 cv2.line() 함수의 매개 변수의 의미를 알아보자.

```
1 import cv2
2
3 img = cv2.imread('./img/blank_500.jpg')
4
5 cv2.line(img, (50, 50), (150, 50), (255, 0, 0))
6 cv2.line(img, (200, 50), (300, 50), (0, 255, 0))
7 cv2.line(img, (350, 50), (450, 50), (0, 0, 255))
8
9 cv2.line(img, (100, 100), (400, 100), (255, 255, 0), 10)
10 cv2.line(img, (100, 150), (400, 150), (255, 0, 255), 10)
11 cv2.line(img, (100, 200), (400, 200), (0, 255, 255), 10)
12 cv2.line(img, (100, 250), (400, 250), (200, 200, 200), 10)
13 cv2.line(img, (100, 300), (400, 300), (0, 0, 0), 10)
```


Section 04 그림 그리기

□ 직선 그리기

- 다음 예제에서 다양한 선을 그려보면서 `cv2.line()` 함수의 매개 변수의 의미를 알아보자.

```
14
15 cv2.line(img, (100, 350), (400, 400), (0, 0, 255), 20, cv2.LINE_4)
16 cv2.line(img, (100, 400), (400, 450), (0, 0, 255), 20, cv2.LINE_8)
17 cv2.line(img, (100, 450), (400, 500), (0, 0, 255), 20, cv2.LINE_AA)
18 cv2.line(img, (0, 0), (500, 500), (0, 0, 255))
19
20 cv2.imshow('lines', img)
21 cv2.waitKey(0)
22 cv2.destroyAllWindows()
```

Section 04 그림 그리기

□ 사각형 그리기

- 사각형을 그리는 함수는 `cv2.rectangle()` 이다.
- `cv2.rectangle(img, start, end, color[, thickness, lineType])`
 - img: 그림 그릴 대상 이미지, NumPy 배열
 - start: 사각형 시작 꼭짓점 (x, y)
 - end: 사각형 끝 꼭짓점 (x, y)
 - color: 색상(Blue, Green, Red)
 - thickness: 선 두께
 - - 1: 채우기
 - lineType: 선 타입, `cv2.LINE()` 과 동일

Section 04 그림 그리기

□ 사각형 그리기

- 사각형을 그릴 때 사용하는 `cv2.rectangle()` 함수는 앞서 설명한 `cv2.line()` 함수와 사용법이 거의 비슷하다.
- 다만, 선이 아닌 면을 그리는 것이므로 선의 두께를 지시하는 `thickness` 에 -1을 지정하면 사각형 면 전체를 `color`로 채우기를 한다.
- 사각형을 그리기 위한 좌표는 시작 점의 좌표 두 쌍과 그 반대 지점의 좌표 두 쌍으로 표현한다.

Section 04 그림 그리기

□ 사각형 그리기

- 다음 예제는 사각형을 그리는 예제이다.

```
1  import cv2
2
3  img = cv2.imread('./img/blank_500.jpg')
4
5  cv2.rectangle(img, (50, 50), (150, 150), (255, 0, 0))
6  cv2.rectangle(img, (300, 300), (100, 100), (0, 255, 0), 10)
7  cv2.rectangle(img, (450, 200), (200, 450), (0, 0, 255), -1)
8
9  cv2.imshow('rectangle', img)
10 cv2.waitKey(0)
11 cv2.destroyAllWindows()
```

Section 04 그림 그리기

□ 다각형 그리기

- 다각형을 그리는 함수는 `cv2.polylines()` 이다.
- `cv2.polylines(img, [points], isClosed, color [, thickness, lineType])`
 - `img`: 그림 그릴 대상 이미지
 - `points`: 꼭짓점 좌표, NumPy 배열 리스트
 - `isClosed`: 닫힌 도형 여부, `True/ False`
 - `color`: 색상(`Blue, Green, Red`)
 - `thickness`: 선 두께
 - `lineType`: 선 타입, `cv2.LINE()` 과 동일
- 이 함수의 `points` 인자는 각형을 그리기 위한 여러 개의 꼭짓점 좌표를 전달한다.
- 이때 좌표를 전달하는 형식이 지금까지와는 달리 NumPy 배열 형식이다.
- `isClosed` 인자는 Boolean 타입인데, `True`는 첫 꼭짓점과 마지막 꼭짓점을 연결해서 닫힌 도형(면)을 그리게 하고, `False`는 단순히 여러 꼭짓점을 잇는 선을 그리게 한다.

Section 04 그림 그리기

□ 다각형 그리기

- 다음 예제는 다각형을 그리는 예제이다.

```
1  import cv2
2  import numpy as np
3
4  img = cv2.imread('./img/blank_500.jpg')
5
6  pts1 = np.array([[50, 50], [150, 150], [100, 140], [200, 240]], dtype=np.int32)
7  pts2 = np.array([[350, 50], [250, 200], [450, 200]], dtype=np.int32)
8  pts3 = np.array([[150, 300], [50, 450], [250, 450]], dtype=np.int32)
9  pts4 = np.array([[350, 250], [450, 350], [400, 450], [300, 450], [250, 350]], dtype=np.int32)
10
```

Section 04 그림 그리기

□ 다각형 그리기

- 다음 예제는 다각형을 그리는 예제이다.

```
11 cv2.polylines(img, [pts1], False, (250, 0, 0))
12 cv2.polylines(img, [pts2], False, (0, 0, 0), 10)
13 cv2.polylines(img, [pts3], True, (0, 0, 255), 10)
14 cv2.polylines(img, [pts4], True, (0, 0, 0))
15
16 cv2.imshow('polyline', img)
17 cv2.waitKey(0)
18 cv2.destroyAllWindows()
```

Section 04 그림 그리기

□ 원, 타원, 호 그리기

- 원을 그리기 위한 함수는 다음과 같다.

- `cv2.circle(img, center, radius, color [, thickness, lineType])`

- `img` : 그림 상 이미지
- `center`: 원점 좌표(x, y)
- `radius`: 원의 반지름
- `color` 색상(Blue, Green, Red)
- `thickness`: 선 두께(-1 : 채우기)
- `lineType`: 선 타입, `cv2.LINE()` 과 동일

Section 04 그림 그리기

□ 원, 타원, 호 그리기

- 호나 타원을 그리기 위한 함수는 다음과 같다.
- `cv2.ellipse(img, center, axes, angle, from, to, color[, thickness, lineType])`
 - `img` : 그림 대상 이미지
 - `center`: 원점 좌표(x, y)
 - `axes`: 기준 축 길이
 - `angle`: 기준 축 회전 각도
 - `from, to`: 호를 그릴 시작 각도와 끝 각도
- 완전한 동그라미를 그릴 때 가장 좋은 함수는 `cv2.circle()` 이다.
- 하지만, 이 함수로는 동그라미의 일부분, 즉 호를 그리거나 찌그러진 동그라미인 타원을 그리는 것은 불가능하며, 이런 호나 타원을 그리려면 `cv2.ellipse()` 함수를 써야 한다.

Section 04 그림 그리기

□ 원, 타원, 호 그리기

- 다음 예제는 원, 호, 타원을 그리는 코드이다.

```
1 import cv2
2
3 img = cv2.imread('./img/blank_500.jpg')
4
5 cv2.circle(img, (150, 150), 100, (255, 0, 0))
6 cv2.circle(img, (300, 150), 70, (0, 255, 0), 5)
7 cv2.circle(img, (400, 150), 50, (0, 0, 255), -1)
8
9 cv2.ellipse(img, (50, 300), (50, 50), 0, 0, 360, (0, 0, 255))
10 cv2.ellipse(img, (150, 300), (50, 50), 0, 0, 180, (255, 0, 0))
11 cv2.ellipse(img, (200, 300), (50, 50), 0, 181, 360, (0, 0, 255))
12
13 cv2.ellipse(img, (325, 300), (75, 50), 0, 0, 360, (0, 0, 0))
14 cv2.ellipse(img, (450, 300), (50, 75), 0, 0, 360, (255, 0, 255))
```

Section 04 그림 그리기

□ 원, 타원, 호 그리기

- 다음 예제는 원, 호, 타원을 그리는 코드이다.

```
15
16 cv2.ellipse(img, (50, 425), (50, 75), 15, 0, 360, (0, 0, 0))
17 cv2.ellipse(img, (225, 425), (50, 75), 45, 0, 360, (0, 0, 0))
18
19 cv2.ellipse(img, (350, 425), (50, 75), 45, 0, 180, (0, 0, 255))
20 cv2.ellipse(img, (400, 425), (50, 75), 45, 181, 360, (255, 0, 0))
21
22 cv2.imshow('circle', img)
23 cv2.waitKey(0)
24 cv2.destroyAllWindows()
```

Section 04 그림 그리기

□ 글씨 그리기

- 문자열을 이미지에 표시하는 함수는 `cv2.putText()` 이다.
- `cv2.putText(img, text, point, fontFace, fontSize, color [, thickness, lineType])`
 - `img` : 글씨를 표시할 이미지
 - `text` : 표시할 문자열
 - `point` : 글씨를 표시할 좌표(좌측 하단 기준)(`x, y`)
 - `fontFace` : 글꼴
 - `cv2.FONT_HERSHEY_PLAIN` : 산세리프체 작은 글꼴
 - `cv2.FONT_HERSHEY_SIMPLEX` : 산세리프체 일반 글꼴
 - `cv2.FONT_HERSHEY_DUPLEX` : 산세리프체 진한 글꼴
 - `cv2.FONT_HERSHEY_COMPLEX_SMALL` : 세리프체 작은 글꼴
 - `cv2.FONT_HERSHEY_COMPLEX` : 세리프체 일반 글꼴
 - `cv2.FONT_HERSHEY_TRIPLEX` : 세리프체 진한 글꼴
 - `cv2.FONT_HERSHEY_SCRIPT_SIMPLEX` : 필기체 산세리프 글꼴
 - `cv2.FONT_HERSHEY_SCRIPT_COMPLEX` : 필기체 세리프 글꼴
 - `cv2.FONT_ITALIC` : 이탤릭체 플래그

Section 04 그림 그리기

□ 글씨 그리기

- 문자열을 이미지에 표시하는 함수는 `cv2.putText()` 이다.
- `cv2.putText(img, text, point, fontFace, fontSize, color [, thickness, lineType])`
 - `fontSize` : 글꼴 크기
 - `color, thickness, lineType`: `cv2.rectangle()` 과 동일
- `point` 좌표는 문자열의 좌측 하단을 기준으로 지정해야 한다.
- 선택할 수 있는 글꼴의 종류는 위의 설명처럼 `cv2.FONT_HERSHEY_`로 시작하는 상수로 정해져 있다.
- 크게 세리 (serif)체와 산세리프(sans-serif)체 그리고 필기체로 나뉘는데, 세리프체는 한글 글꼴의 명조체처럼 글자에 장식을 붙여 모양을 낸 글꼴을 통틀어 말하며, 산세리프체는 고딕체처럼 획에 특별히 모양을 낸 것이 없는 글꼴을 말한다.
- OpenCV 상수에서는 상대적으로 단순한 모양인 산세리프체에 `SIMPLEX`라는 이름을 붙였고, 상대적으로 복잡한 모양인 세리프체에 `COMPLEX`라는 이름을 붙인 것을 볼 수 있다.

Section 04 그림 그리기

□ 글씨 그리기

- 다음 예제는 각각의 글꼴을 보여주고 있다.

```
1  import cv2
2
3  img = cv2.imread('./img/blank_500.jpg')
4
5  cv2.putText(img, "Plain", (50, 30), cv2.FONT_HERSHEY_PLAIN, 1, (0, 0, 0))
6  cv2.putText(img, "Simplex", (50, 70), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0))
7  cv2.putText(img, "Duplex", (50, 110), cv2.FONT_HERSHEY_DUPLEX, 1, (0, 0, 0))
8  cv2.putText(img, "Simplex", (200, 110), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 250))
9
10 cv2.putText(img, "Complex Small", (50, 180), cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0,
    0, 0))
11
12 cv2.putText(img, "Complex", (50, 220), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 0, 0))
13 cv2.putText(img, "Triplex", (50, 260), cv2.FONT_HERSHEY_TRIPLEX, 1, (0, 0, 0))
14
```

Section 04 그림 그리기

□ 글씨 그리기

- 다음 예제는 각각의 글꼴을 보여주고 있다.

```
15 cv2.putText(img, "Complex", (200, 260), cv2.FONT_HERSHEY_COMPLEX, 2, (0, 0, 255))
16
17 cv2.putText(img, "Script Simplex", (50, 330), cv2.FONT_HERSHEY_SCRIPT_SIMPLEX, 1, (0, 0, 0))
18 cv2.putText(img, "Script Complex", (50, 370), cv2.FONT_HERSHEY_SCRIPT_COMPLEX, 1, (0, 0, 0))
19
20 cv2.putText(img, "Plain Italic", (50, 430), cv2.FONT_HERSHEY_PLAIN | cv2.FONT_ITALIC, 1, (0, 0, 0))
21 cv2.putText(img, "Complex Italic", (50, 470), cv2.FONT_HERSHEY_COMPLEX | cv2.FONT_ITALIC, 1, (0, 0, 0))
22
22 cv2.imshow('draw text', img)
23 cv2.waitKey(0)
24 cv2.destroyAllWindows()
```

Section 05 이벤트 처리

□ 키보드 이벤트

- 이미 앞서 여러 번 사용한 적이 있는 `cv2.waitKey(delay)` 함수를 쓰면 키보드의 입력을 알아낼 수 있다.
- 이 함수는 `delay` 인자에 밀리초(ms, 0.001초) 단위로 숫자를 전달하면 해당 시간 동안 프로그램을 멈추고 대기하다가 키보드의 눌린 키에 대응하는 코드 값을 정수로 반환한다.
- 지정한 시간까지 키보드 입력이 없으면 -1을 반환한다. `delay` 인자에 0을 전달하면 대기 시간을 무한대로 하겠다는 의미이므로 키를 누를 때까지 프로그램은 멈추고 이 때는 -1을 반환할 일은 없다.
- 키보드에서 어떤 키를 눌렀는 지를 알아내려면 `cv2.waitKey()` 함수의 반환 값을 출력해 보면 된다.
 - `key = cv2.waitKey(0)`
 - `print(key)`

Section 05 이벤트 처리

□ 키보드 이벤트

- 출력되는 키 값을 확인해 보면 ASCII 코드와 같다는 것을 알 수 있다.
- 환경에 따라 한글 모드에서 키를 입력하면 오류가 발생할 수 있으니 키를 입력할 때 한글은 사용하지 않는 것이 좋다.
- 입력된 키를 특정 문자와 비교할 때는 파이썬 기본 함수인 `ord()` 함수를 사용하면 편리하다.
- 예를 들어 키보드의 'a' 키를 눌렀는지 확인하기 위한 코드는 다음과 같다.
 - `if cv2.waitKey(0) == ord(' a'):`

Section 05 이벤트 처리

□ 키보드 이벤트

- 그런데 몇몇 64 비트 환경에서 `cv2.waitKey()` 함수는 8비트(ASCII 코드 크기)보다 큰 32비트 정수를 반환해서 그 값을 `ord()` 함수를 통해 비교하면 서로 다른 값으로 판단할 때가 있다.
- 그래서 하위 8비트를 제외한 비트를 지워야 하는 경우가 있다.
- `0xFF`는 하위 8비트가 모두 1로 채워진 숫자이므로 이것과 `&` 연산을 수행하면 하위 8비트보다 높은 비트는 모두 0으로 채울 수 있다.
 - `key = cv2.waitKey (0) & 0xFF`
 - `if key == ord ('a') :`

Section 05 이벤트 처리

□ 키보드 이벤트

- 예제는 키보드의 입력 값을 확인하는 코드이다.

```
1  import cv2
2
3  img = cv2.imread('./img/model3.jpg')
4  while True:
5      cv2.imshow('img', img)
6
7      keyValue = cv2.waitKey(0)
8      print(keyValue)
9
10     if keyValue == ord('q'):
11         break
12
13 cv2.destroyAllWindows()
```

Section 05 이벤트 처리

□ 키보드 이벤트

- 예제는 화면에 이미지를 표시하고 키보드의 화살표 키를 누르면 창의 위치가 상, 하, 좌, 우 방향으로 10픽셀씩 움직이고, 'esc' 키 또는 'q' 키를 누르면 종료되는 코드이다.

```
1 import cv2
2
3 img_file = './img/model3.jpg'
4 img = cv2.imread(img_file)
5 title = 'IMG'
6 x, y = 100, 100
7
8 while True:
9     cv2.imshow(title, img)
10    cv2.moveWindow(title, x, y)
11    key = cv2.waitKey(0)
12    print(key, chr(key))
```

Section 05 이벤트 처리

□ 키보드 이벤트

- 예제는 화면에 이미지를 표시하고 키보드의 화살표 키를 누르면 창의 위치가 상, 하, 좌, 우 방향으로 10픽셀씩 움직이고, 'esc' 키 또는 'q' 키를 누르면 종료되는 코드이다.

```
13     if key == 81 :
14         x -= 10
15     elif key == 84 :
16         y += 10
17     elif key == 82 :
18         y -= 10
19     elif key == 83 :
20         x += 10
21     elif key == ord('q') or key == 27 :
22         cv2.destroyAllWindows()
23         break
24     cv2.moveWindow(title, x, y)
```

Section 05 이벤트 처리

□ 키보드 이벤트

- 다음 예제 코드에서는 자동차의 카메라를 켜고 화살표로 자동차를 제어할 것이다.
- 스페이스 바를 누르면 자동차가 정지하도록 코드를 작성하였다.

```
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
```

Section 05 이벤트 처리

□ 키보드 이벤트

- 다음 예제 코드에서는 자동차의 카메라를 켜고 화살표로 자동차를 제어할 것이다.
- 스페이스 바를 누르면 자동차가 정지하도록 코드를 작성하였다.

```
14     if (motor_status == "REVERSE") :
15         swcar.SIO_WriteMotor(100, 0)
16         time.sleep(1)
17         swcar.SIO_WriteMotor(100, speed)
18         motor_status = "FORWARD"
19
20     def motor_reverse(speed):
21         global motor_status
22         if (motor_status == "FORWARD") :
23             swcar.SIO_WriteMotor(100, 0)
24             time.sleep(1)
25             swcar.SIO_WriteMotor(100, speed)
26             motor_status = "REVERSE"
27
```

Section 05 이벤트 처리

□ 키보드 이벤트

- 다음 예제 코드에서는 자동차의 카메라를 켜고 화살표로 자동차를 제어할 것이다.
- 스페이스 바를 누르면 자동차가 정지하도록 코드를 작성하였다.

```
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"
37
38 def servo_right():
39     global servo_status
40     swcar.SIO_WriteServo(100, 10)
41     servo_status = "RIGHT"
```


Section 05 이벤트 처리

□ 키보드 이벤트

- 다음 예제 코드에서는 자동차의 카메라를 켜고 화살표로 자동차를 제어할 것이다.
- 스페이스 바를 누르면 자동차가 정지하도록 코드를 작성하였다.

```
42
43     def servo_center():
44         global servo_status
45         swcar.SIO_WriteServo(100, 50)
46         servo_status = "CENTER"
47
48     swcar.SIO_Init(0)
49
50     motor_stop()
51     servo_center()
52     time.sleep(2)
53
54     print('Press ctrl + c to terminate program')
55     print('Running Driving Motor.')
```

Section 05 이벤트 처리

□ 키보드 이벤트

- 다음 예제 코드에서는 자동차의 카메라를 켜고 화살표로 자동차를 제어할 것이다.
- 스페이스 바를 누르면 자동차가 정지하도록 코드를 작성하였다.

```
56
57 cam = cv2.VideoCapture(0)
58 cam.set(3, 640)
59 cam.set(4, 480)
60
61 fspeed = 4
62 bspeed = -25
63
64 try:
65     while (cam.isOpened()):
66         _, img = cam.read()
67         cv2.imshow('Camera', img)
68
```

Section 05 이벤트 처리

□ 키보드 이벤트

- 다음 예제 코드에서는 자동차의 카메라를 켜고 화살표로 자동차를 제어할 것이다.
- 스페이스 바를 누르면 자동차가 정지하도록 코드를 작성하였다.

```
69     dist1 = mycar.sensor_get_distance_LS_FRONT()
70     dist2 = mycar.sensor_get_distance_US_FRONT()
71     dist3 = mycar.sensor_get_distance_US_REAR()
72
73     if (dist1 < 200) or (dist2 < 150) or (dist3 < 150) :
74         motor_stop()
75
76     key = cv2.waitKey(0)
77     print(key)
78
79     if key == ord('q'):
80         motor_stop()
81         time.sleep(0.02)
82         servo_center()
```

Section 05 이벤트 처리

□ 키보드 이벤트

- 다음 예제 코드에서는 자동차의 카메라를 켜고 화살표로 자동차를 제어할 것이다.
- 스페이스 바를 누르면 자동차가 정지하도록 코드를 작성하였다.

```
83     time.sleep(0.02)
84     break
85 elif key == 82:
86     print("go")
87     servo_center()
88     time.sleep(0.02)
89     motor_forward(fspeed)
90     time.sleep(0.02)
91 elif key == 84:
92     print("back")
93     servo_center()
94     time.sleep(0.02)
95     motor_reverse(bspeed)
96     time.sleep(0.02)
```

Section 05 이벤트 처리

□ 키보드 이벤트

- 다음 예제 코드에서는 자동차의 카메라를 켜고 화살표로 자동차를 제어할 것이다.

```
97         elif key == 81:
98             print("left")
99             servo_left()
100            time.sleep(0.02)
101        elif key == 83:
102            print("right")
103            servo_right()
104            time.sleep(0.02)
105        elif key == 32:
106            print("stop")
107            motor_stop()
108            time.sleep(0.02)
109
110    cam.release()
111    cv2.destroyAllWindows()
```

Section 05 이벤트 처리

□ 마우스 이벤트

- 마우스에서 입력을 받으려면 이벤트를 처리할 함수를 미리 선언해 놓고 `cv2.setMouseCallback()` 함수에 그 함수를 전달한다. 코드로 간단히 묘사하면 다음과 같다.

```
def onMouse(event, x, y, flags, param) :  
    pass
```

```
cv2.setMouseCallback('title', onMouse)
```

- 이 두 함수의 모양은 아래와 같다.
 - `cv2.setMouseCallback(win_name, onMouse [, param])` : onMouse 함수를 등록
 - win_name: 이벤트를 등록할 윈도우 이름
 - onMouse: 이벤트 처리를 위해 미리 선언해 놓은 콜백 함수
 - param: 필요에 따라 onMouse 함수에 전달할 인자

Section 05 이벤트 처리

□ 마우스 이벤트

- 이 두 함수의 모양은 아래와 같다.

- MouseCallback (event, x, y, flags, param): 콜백 함수 선언부
 - event : **마우스 이벤트 종류**, cv2.EVENT_로 시작하는 상수(12가지)
 - cv2.EVENT_MOUSEMOVE: **마우스 움직임**
 - cv2.EVENT_LBUTTONDOWN : **왼쪽 버튼 누름**
 - cv2.EVENT_RBUTTONDOWN : **오른쪽 버튼 누름**
 - cv2.EVENT_MBUTTONDOWN : **가운데 버튼 누름**
 - cv2.EVENT_LBUTTONUP : **왼쪽 버튼 땀**
 - cv2.EVENT_RBUTTONUP : **오른쪽 버튼 땀**
 - cv2.EVENT_MBUTTONUP : **가운데 버튼 땀**
 - cv2.EVENT_LBUTTONDBLCLK : **왼쪽 버튼 더블 클릭**
 - cv2.EVENT_RBUTTONDBLCLK : **오른쪽 버튼 더블 클릭**
 - cv2.EVENT_MBUTTONDBLCLK : **가운데 버튼 더블 클릭**
 - cv2.EVENT_MOUSEWHEEL : **휠 스크롤**
 - cv2.EVENT_MOUSEHWHEEL: **휠 가로 스크롤**

Section 05 이벤트 처리

□ 마우스 이벤트

- 이 두 함수의 모양은 아래와 같다.
 - MouseCallback (event, x, y, flags, param): 콜백 함수 선언부
 - x, y: 마우스좌표
 - flags: 마우스 동작과 함께 일어난 상태, cv2.EVENT_FLAG_로 시작하는 상수(6가지)
 - cv2.EVENT_FLAG_LBUTTON(1) : 왼쪽 버튼 누름
 - cv2.EVENT_FLAG_RBUTTON(2) : 오른쪽 버튼 누름
 - cv2.EVENT_FLAG_MBUTTON(4) : 가운데 버튼 누름
 - cv2.EVENT_FLAG_CTRLKEY(8) : Ctrl 키 누름
 - cv2.EVENT_FLAG_SHIFTKEY(16) : Shift 키 누름
 - cv2.EVENT_FLAG_ALTKEY(32) : Alt 키 누름
 - param: cv2.setMouseCallback() 함수에서 전달한 인자

Section 05 이벤트 처리

□ 마우스 이벤트

- 다음 예제는 마우스를 클릭하면 지름이 30픽셀인 동그라미를 그리는 예제이다.

```
1  import cv2
2
3  title = 'mouse event'
4  img = cv2.imread('../img/blank_500.jpg')
5  cv2.imshow(title, img)
6
7  def onMouse(event, x, y, flags, param) :
8      print(event, x, y)
9      if event == cv2.EVENT_LBUTTONDOWN :
10         cv2.circle(img, (x, y), 30, (0, 0, 0), -1)
11         cv2.imshow(title, img)
12
```

Section 05 이벤트 처리

□ 마우스 이벤트

- 다음 예제는 마우스를 클릭하면 지름이 30픽셀인 동그라미를 그리는 예제이다.

```
13 cv2.setMouseCallback(title, onMouse)
14
15 while True:
16     if cv2.waitKey(0) & 0xFF == 27:
17         break
18 cv2.destroyAllWindows()
```

Section 05 이벤트 처리

□ 마우스 이벤트

- 예제는 앞서 다룬 마우스로 동그라미 그리기 예제를 컨트롤 키를 누르면 빨간색으로, 시프트 키를 누르면 파란색으로, 시프트 키와 컨트롤 키를 동시에 누르면 초록색으로 그리게 수정한 것이다.

```
1  import cv2
2
3  title = 'mouse event'
4  img = cv2.imread('../img/blank_500.jpg')
5  cv2.imshow(title, img)
6
7  colors = {'black' : (0, 0, 0),
8           'red' : (0, 0, 255),
9           'blue' : (255, 0, 0),
10          'green' : (0, 255, 0) }
11
12  def onMouse(event, x, y, flags, param) :
13      print(event, x, y, flags)
14      color = colors['black']
```

Section 05 이벤트 처리

□ 마우스 이벤트

- 예제는 앞서 다룬 마우스로 동그라미 그리기 예제를 컨트롤 키를 누르면 빨간색으로, Shift 키를 누르면 파란색으로, Shift 키와 Ctrl 키를

```

15     if event == cv2.EVENT_LBUTTONDOWN :
16         if flags & cv2.EVENT_FLAG_CTRLKEY and flags & cv2.EVENT_FLAG_SHIFT
KEY :
17             color = colors['green']
18         elif flags & cv2.EVENT_FLAG_SHIFTKEY :
19             color = colors['blue']
20         elif flags & cv2.EVENT_FLAG_CTRLKEY :
21             color = colors['red']
22         cv2.circle(img, (x, y), 30, color, -1)
23         cv2.imshow(title, img)
24
25     cv2.setMouseCallback(title, onMouse)
26
27     while True:
28         if cv2.waitKey(0) & 0xFF == 27:
29             break
30     cv2.destroyAllWindows()

```

Section 05 이벤트 처리

□ 트랙바

- **트랙 바(track-bar)**는 슬라이드 모양의 인터페이스를 마우스로 움직여서 값을 입력받는 GUI 요소이다.
- `cv2.createTrackbar()` 함수로 생성하면서 보여지기를 원하는 창의 이름을 지정한다.
- 이 때 마우스 이벤트의 방식과 마찬가지로 트랙바를 움직였을 때 동작할 함수를 미리 준비해서 함께 전달한다.
- 트랙바의 값을 얻기 위한 `cv2.getTrackbarPos()` 함수도 함께 쓰인다.
- 트랙바를 사용하기 위한 주요 코드 형식은 아래와 같다.

```
def onChange (value) :  
    v = cv2.getTrackbarPos('trackbar', 'win_name')
```

```
cv2.createTrackbar('trackbar', 'win_name', 0, 100, onChange )
```

Section 05 이벤트 처리

□ 트랙바

- 여기서 사용하는 세 함수의 형식은 다음과 같다.

- `cv2.createTrackbar(trackbar_name, win_name, value, count, onChange)` :
트랙바 생성

- `trackbar_name`: 트랙바 이름
- `win_name`: 트랙바를 표시할 창 이름
- `value` : 트랙바 초기 값, 0~count 사이의 값
- `count` : 트랙바 눈금의 개수, 트랙바가 표시할 수 있는 최대 값
- `onChange` : `TrackbarCallback`, 트랙바 이벤트 핸들러 함수

- `TrackbarCallback(value)` : **트랙바 이벤트 콜백 함수**

- `value` : 트랙바가 움직인 새 위치 값

- `pos = cv2.getTrackbarPos(trackbar_name, win_name)`

- `trackbar name`: 찾고자 하는 트랙바 이름
- `win_name`: 트랙바가 있는 창의 이름
- `pos` : 트랙바 위치 값

Section 05 이벤트 처리

□ 트랙바

- 예제는 트랙바 3개를 생성하여 각 트랙바를 움직여 이미지의 색상을 조정하는 예제이다.

```
1  import cv2
2
3
4  win_name = 'Trackbar'
5
6  img = cv2.imread('./img/blank_500.jpg')
7  cv2.imshow(win_name, img)
8
9  def onChange(x) :
10     print(x)
11     r = cv2.getTrackbarPos('R', win_name)
12     g = cv2.getTrackbarPos('G', win_name)
13     b = cv2.getTrackbarPos('B', win_name)
14     print(r, g, b)
15     img[:] = [b, g, r]
16     cv2.imshow(win_name, img)
```

Section 05 이벤트 처리

□ 트랙바

- 예제는 트랙바 3개를 생성하여 각 트랙바를 움직여 이미지의 색상을 조정하는 예제이다.

```
17
18 cv2.createTrackbar('R', win_name, 255, 255, onChange)
19 cv2.createTrackbar('G', win_name, 255, 255, onChange)
20 cv2.createTrackbar('B', win_name, 255, 255, onChange)
21
22 while True :
23     if cv2.waitKey(1) & 0xFF == 27 :
24         break
25 cv2.destroyAllWindows()
```

Q&A

