OpenCV 인터페이스

담당교수: 김민기

목 차

윈도우 제어

❖영상처리

- 2차원 행렬에 대한 연산
- 연산과정에서 행렬 원소 변경
- 전체 영상에 대한 변화 인지하기 어려움

❖윈도우 영상 표시

- 영상처리로 적용된 행렬 연산의 의미를 쉽게 이해할 수 있음
- OpenCV에서는 윈도우(window, 창)가 활성화된 상태에서만 마우스나 키보드 이벤트 감지

윈도우 제어

- 윈도우 생성
  - namedWindow(winname, flags)
- 영상을 윈도우에 디스플레이
  - imshow(winname, mat)
- 윈도우 제거
  - destroyWindow(winname)
  - destroyAllWindows()
- 윈도우 이동
  - moveWindow(winname, x, y)
- 윈도우 크기 변경
  - resizeWindow(winname, width, height)

예제 4.1.1 윈도우 이동 - 01.move_window.py

```
01  import numpy as np                                  # 넘파이 라이브러리 임포트
02  import cv2                                          # OpenCV 라리브러리 임포트
03
04  image = np.zeros((200, 400), np.uint8)              # 행렬 생성
05  image[:] = 200                                      # 밝은 회색(200) 바탕 영상 생성
06
07  title1, title2 = 'Position1', 'Position2'           # 윈도우 이름
08  cv2.namedWindow(title1, cv2.WINDOW_AUTOSIZE)        # 윈도우 생성 및 크기 조정 옵션
09  cv2.namedWindow(title2)
10  cv2.moveWindow(title1, 150, 150)                    # 윈도우 이동 - 위치 지정
11  cv2.moveWindow(title2, 400, 50)
12
13  cv2.imshow(title1, image)                           # 행렬 원소를 영상으로 표시
14  cv2.imshow(title2, image)
15  cv2.waitKey(0)                                      # 키 이벤트(key event) 대기
16  cv2.destroyAllWindows()                             # 열린 모든 윈도우 파괴
```

예제 4.1.2 윈도우의 크기 변경 – 02.window_resize.py

```
01  import numpy as np
02  import cv2
03
04  image = np.zeros((200, 300), np.uint8)              # ndarray 행렬 생성
05  image.fill(255)                                     # 모든 원소에 255(흰색) 지정
06
07  title1, title2 = 'AUTOSIZE', 'NORMAL'               # 윈도우 이름 변수
08  cv2.namedWindow(title1, cv2.WINDOW_AUTOSIZE)        # 윈도우 생성 – 크기변경 불가
09  cv2.namedWindow(title2, cv2.WINDOW_NORMAL)          # 크기 변경 가능
10
11  cv2.imshow(title1, image)                           # 행렬 원소를 영상으로 표시
12  cv2.imshow(title2, image)
13  cv2.resizeWindow(title1, 400, 300)                  # 윈도우 크기 변경
14  cv2.resizeWindow(title2, 400, 300)
15  cv2.waitKey(0)                                      # 키 이벤트(key event) 대기
16  cv2.destroyAllWindows()                             # 열린 모든 윈도우 제거
```

이벤트 처리 함수

❖ 이벤트(event)
- 프로그램에 의해 감지되고 처리될 수 있는 동작이나 사건
- 이벤트의 예
  - 사용자가 키보드의 키를 누르는 이벤트
  - 마우스를 움직인다거나 마우스 버튼을 누르는 이벤트
  - 타이머(timer)와 같은 하드웨어 장치가 발생시키는 이벤트
  - 사용자가 자체적으로 정의하는 이벤트

이벤트 처리 함수

❖콜백 함수
- 이벤트를 처리하기 위해 정의된 함수
- 이벤트가 발생하거나 특정 시점에 도달했을 때 시스템이 개발자가 등록한 함수 호출

❖OpenCV에서도 기본적인 이벤트 처리 함수 지원
- 키보드 이벤트
- 마우스 이벤트
- 트랙바(trackbar) 이벤트

키보드 이벤트 제어

❖키보드 이벤트를 처리하기 위해 콜백함수 정의 불필요

❖OpenCV에서 제공하는 아래 함수를 사용

- waitKey(delay)
  - Delay(ms)시간만큼 키 입력 대기, 해당 키 값을 반환
- waitKeyEx(delay)
  - 전체 키코드를 반환, 화살표 키 등 확장키 입력을 처리할 때 사용

예제 4.2.1 키 이벤트 사용 – 03.event_key.py

```
01  import numpy as np
02  import cv2
03
04  ## switch case문을 사전(dictionary)으로 구현
05  switch_case = {
06      ord('a'): "a키 입력",                    # ord() 함수: 문자 → 아스키코드 변환
07      ord('b'): "b키 입력",
08      0x41: "A키 입력",
09      int('0x42', 16): "B키 입력",             # 0x42(16진수) → 10진수 변환
10      2424832: "왼쪽 화살표키 입력",           # 0x250000
11      2490368: "윗쪽 화살표키 입력",           # 0x260000
12      2555904: "오른쪽 화살표키 입력",         # 0x270000
13      2621440: "아래쪽 화살표키 입력"          # 0x280000
14  }
15
16  image = np.ones((200, 300), np.float)        # 원소값 1인 행렬 생성
17  cv2.namedWindow('Keyboard Event')            # 윈도우 이름
18  cv2.imshow("Keyboard Event", image)
19
20  while True:                                  # 무한 반복
21      key = cv2.waitKeyEx(100)                 # 100ms 동안 키 이벤트 대기
22      if key == 27: break                      # ESC 키 누르면 종료
23
24      try:
25          result = switch_case[key]
26          print(result)
27      except KeyError:
28          result = -1
29
30  cv2.destroyAllWindows()                      # 열린 모든 윈도우 제거
```

마우스 이벤트 제어

❖마우스 이벤트 처리를 위한 콜백함수 등록

- setMouseCallback(winname, onMouse, param)
  - onMouse: 프로그래머가 정의한 콜백함수

❖콜백함수 정의

- onMouse(event, x, y, flags, param)
  - Event: 마우스 L/R 클릭, 누르기, 떼기, 더블클릭 …
  - Flags: 특수키 Shift, Alt, Ctrl를 눌렀는지 여부

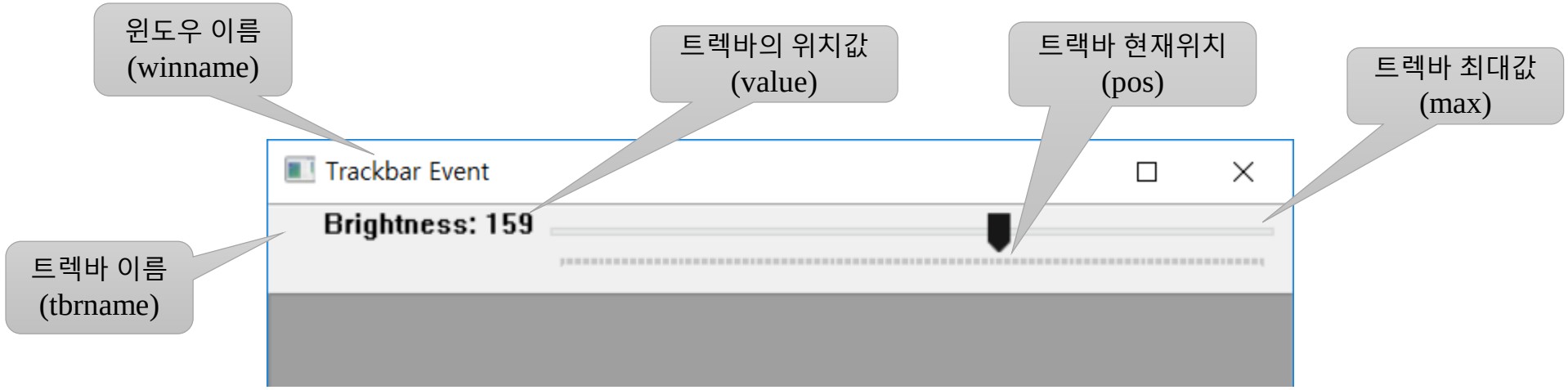
예제 4.2.2 마우스 이벤트 사용 – 04.event_mouse.py

```
01  import numpy as np
02  import cv2
03
04  def onMouse(event, x, y, flags, param):                    # 콜백 함수 – 이벤트 내용 출력
05      if event == cv2.EVENT_LBUTTONDOWN:
06          print("마우스 왼쪽 버튼 누르기")
07      elif event == cv2.EVENT_RBUTTONDOWN:
08          print("마우스 오른쪽 버튼 누르기")
09      elif event == cv2.EVENT_RBUTTONUP:
10          print("마우스 오른쪽 버튼 떼기")
11      elif event == cv2.EVENT_LBUTTONDBLCLK:
12          print("마우스 왼쪽 버튼 더블클릭")
13
14  image = np.full((200, 300), 255, np.uint8)                 # 초기 영상 생성
15
16  title1, title2 = "Mouse Event1", "Mouse Event2"            # 윈도우 이름
17  cv2.imshow(title1, image)                                  # 윈도우 보기
18  cv2.imshow(title2, image)
19
20  cv2.setMouseCallback(title1, onMouse)                      # 마우스 콜백 함수
21  cv2.waitKey(0)                                             # 키 이벤트 대기
22  cv2.destroyAllWindows()                                    # 열린 모든 윈도우 제거
```

트랙바 이벤트 제어

❖트랙바(trackbar) 생성 및 콜백함수 등록

- createTrackbar(tbname, winname, value, max, onChange)
  - value: 슬라이더 위치에 따른 현재 값 (정수)
  - max: 슬라이더 최대값 (최소값은 0)
  - onChange: 프로그래머가 정의한 콜백함수



트랙바 이벤트 제어

❖콜백함수 정의

- onChange(pos): 여기에 처리할 내용 정의

❖트랙바 위치값 가져오기 & 설정하기

- getTrackbarPos(tbname, winname)
- setTrackbarPos(tbname, winname)

예제 4.2.3 트랙바 이벤트 사용 – 05.event_trackbar.py

```
01  import numpy as np
02  import cv2
03
04  def onChange(value):                                    # 트랙바 콜백 함수
05      global image, title                                 # 전역 변수 참조
06
07      add_value = value - int(image[0][0])                # 트랙바 값과 영상 화소값 차분
08      print("추가 화소값:", add_value)
09      image = image + add_value                           # 행렬과 스칼라 덧셈 수행
10      cv2.imshow(title, image)
11
12  image = np.zeros((300, 500), np.uint8)                  # 영상 생성
13
14  title = 'Trackbar Event'
15  cv2.imshow(title, image)
16
17  cv2.createTrackbar('Brightness', title, image[0][0], 255, onChange)  # 트랙바 콜백 함수 등록
18  cv2.waitKey(0)
19  cv2.destroyAllWindows()                                 # 열린 모든 윈도우 제거
```

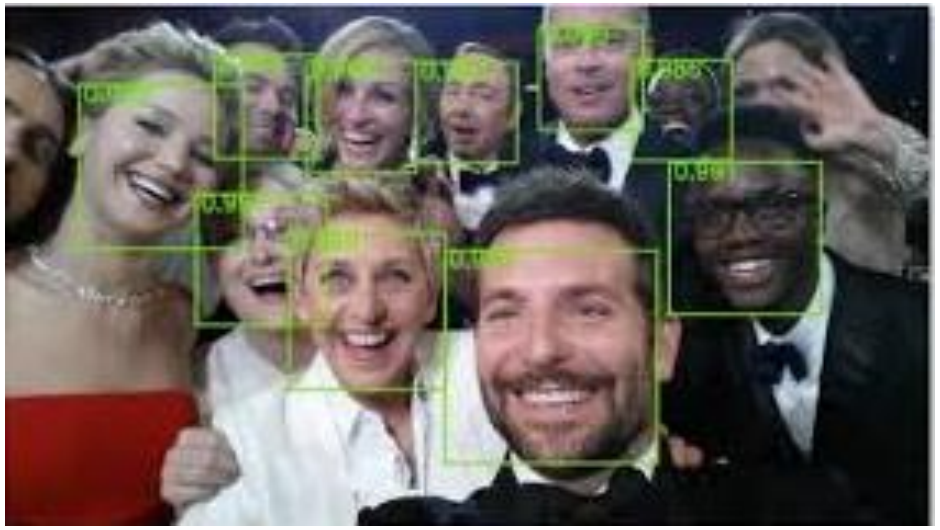
심화예제 4.2.4 마우스 및 트랙바 이벤트 사용 – 06.event_mouse_trackbar.py

```
01  import numpy as np
02  import cv2
03
04  def onChange(value): ···                                    # 트랙바 콜백 함수 – 소스 표시 생략
10
11  def onMouse(event, x, y, flags, param):                     # 마우스 콜백 함수
12       global image, bar_name                                 # 전역 변수 참조
13
14       if event == cv2.EVENT_RBUTTONDOWN:                     # 마우스 우버튼
15            if (image[0][0] < 246): image = image + 10
16            cv2.setTrackbarPos(bar_name, title, image[0][0])  # 트랙바 위치 변경
17            cv2.imshow(title, image)
18
19       elif event == cv2.EVENT_LBUTTONDOWN:
20            if (image[0][0] >= 10):
21                 image = image - 10
22            cv2.setTrackbarPos(bar_name, title, image[0][0])  # 트랙바 위치 변경
23            cv2.imshow(title, image)
24
25  image = np.zeros((300, 500), np.uint8)
26  title = "Trackbar & Mouse Event"                            # 윈도우 이름
27  bar_name = 'Brightness'                                     # 트랙바 이름
28  cv2.imshow(title, image)
29
30  cv2.createTrackbar(bar_name, title, image[0][0], 255, onChange)  # 트랙바 콜백 함수
31  cv2.setMouseCallback(title, onMouse)                        # 마우스 콜백 함수 등록
32  cv2.waitKey(0)                                              # 키 입력 대기
33  cv2.destroyAllWindows()                                     # 모든 윈도우 닫기
```

그리기 함수

❖영상처리 프로그래밍의 결과 확인 필요

- 얼굴 검출 알고리즘을 적용했을 때,
  - 전체 영상 위에 검출한 얼굴 영역을 사각형이나 원으로 표시
- 차선 확인하고자 직선 검출 알고리즘을 적용했을 때,
  - 차선을 정확하게 검출했는지 확인하기 위해 도로 영상 위에 선으로 표시



직선 or 사각형 그리기

❖직선 그리기

- line(img, pt1, pt2, color, thickness, lineType, shift)
  - pt1, pt2: 포인트 좌표(x, y)를 나타내는 2원소 튜플(정수)
  - shift: 입력 좌표에 대한 오른쪽 비트 시프트, 즉 좌표 값 축소비율

❖사각형 그리기

- rectangle(img, pt1, pt2, color, thickness, lineType, shift)
- rectangle(img, rec, color, thickness, linetype, shift)
  - rec: (x, y, w, h)를 나타내는 4원소 튜플(정수)

예제 4.3.1 직선 & 사각형 그리기 – 07.draw_line_rect.py

```
01  import numpy as np
02  import cv2
03
04  blue, green, red = (255, 0, 0), (0, 255, 0), (0, 0, 255)          # 색상 선언
05  image = np.zeros((400, 600, 3), np.uint8)                         # 3채널 컬러 영상 생성
06  image[:] = (255, 255, 255)                                        # 3채널 흰색
07
08  pt1, pt2 = (50, 50),  (250, 350)                                  # 좌표 선언 – 정수형 튜플
09  pt3, pt4 = (400, 150), (500, 50)
10  roi = (50, 200, 200, 100)                                         # 사각형 영역 – 4원소 튜플
11
12  ## 직선 그리기
13  cv2.line(image, pt1, pt2, red)
14  cv2.line(image, pt3, pt4, green, 3, cv2.LINE_AA)                  # 계단 현상 감소선
15
16  ## 사각형 그리기
17  cv2.rectangle(image, pt1, pt2, blue, 3, cv2.LINE_4)               # 4방향 연결선
18  cv2.rectangle(image, roi, red, 3, cv2.LINE_8 )                    # 8방향 연결선
19  cv2.rectangle(image, (400, 200, 100, 100), green, cv2.FILLED)     # 내부 채움
20
21  cv2.imshow("Line & Rectangle", image)                             # 윈도우에 영상 표시
22  cv2.waitKey(0)
23  cv2.destroyAllWindows()                                           # 모든 열린 윈도우 닫기
```

글자 쓰기

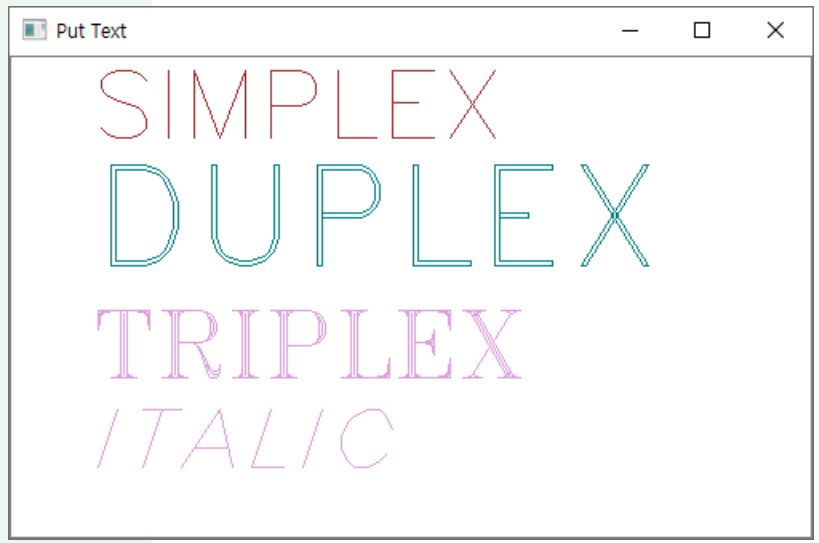
❖영상에 문자열 출력하기

- putText(img, text, pt, fontFace, fontScale,
  color, thinkness, lineType, bottomLeftOrigin)
  - text: 출력할 문자열
  - pt: 문자열의 시작 위치 좌표 (문자열 가장 왼쪽 하단을 의미)
  - fontFace & fontScale: 글꼴 & 글꼴의 크기
  - bottomLeftOrigin: 영상의 원점 좌표 설정(True: 좌하단, False: 좌상단)

```
putText(image, "DUPLEX", pt1, FONT_HERSHEY_DUPLEX, 2, Scalar(128, 128, 0));
putText(image, "TRIPLEX", pt2, FONT_HERSHEY_TRIPLEX, 3, Scalar(221, 160, 221));
```

예제 4.3.2 글자 쓰기 – 08.put_text.py

```
01  import numpy as np
02  import cv2
03
04  olive, violet, brown = (128, 128, 0), (221, 160, 221), (42, 42, 165)       # 색상 지정
05  pt1, pt2 =  (50, 230), (50, 310)                                           # 문자열 위치 좌표
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    # 글자체 상수
14  cv2.putText(image, 'ITALIC', pt2, fontFace, 4, violet)
15
16  cv2.imshow('Put Text', image)                                              # 윈도우 이름 지정 및 영상 표시
17  cv2.waitKey(0)                                                             # 키이벤트 대기
```



원 or 타원 그리기

❖원 그리기

- Circle(img, center, radius, color, thickness, lineType, shift)

❖타원 그리기

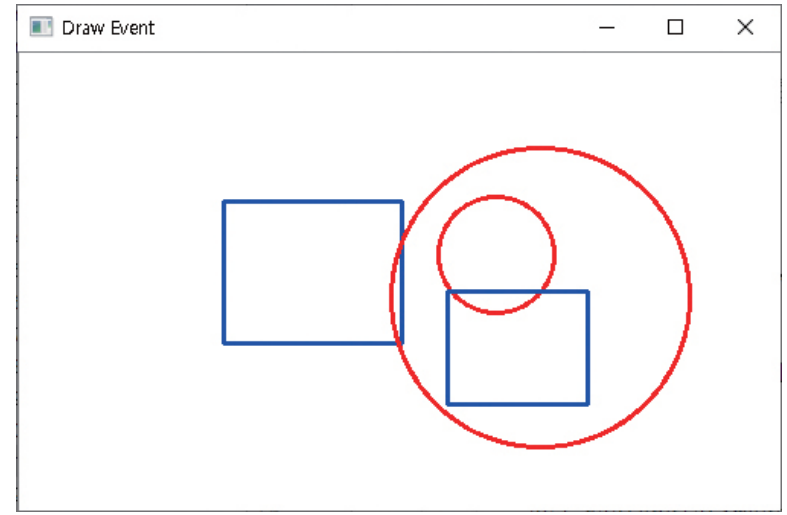
- ellipse(img, center, axes_size, angle, startAngle, endAngle, color, thickness, lineType shift)
  - axes_size: 타원의 절반 크기(x축 반지름, y축 반지름)
  - angle: 타원의 각도 (3시 방향이 0도, 시계방향 회전)
  - startAngle and endEngle: 호의 시작각도, 호의 종료각도

예제 4.3.4 타원 및 호 그리기 – 10.draw_ellipse.py

```
01  import numpy as np
02  import cv2
03
04  orange, blue, white = (0, 165, 255), (255, 0, 0), (255,255,255) # 색상 지정
05  image = np.full((300, 700, 3), white, np.uint8)          # 3채널 행렬 생성 및 초기화
06
07  pt1, pt2 = (180, 150), (550, 150)                        # 타원 중심점
08  size = (120, 60)                                         # 타원 크기 – 반지름 값임
09
10  cv2.circle(image, pt1, 1, 0, 2)                          # 타원의 중심점(2화소 원) 표시
11  cv2.circle(image, pt2, 1, 0, 2)
12
13  cv2.ellipse(image, pt1, size,  0, 0, 360, blue, 1)       # 타원 그리기
14  cv2.ellipse(image, pt2, size, 90, 0, 360, blue, 1)
15  cv2.ellipse(image, pt1, size,  0, 30, 270, orange, 4)    # 호 그리기
16  cv2.ellipse(image, pt2, size, 90, -45, 90, orange, 4)
17
18  cv2.imshow("문자열", image)
19  cv2.waitKey()                                            # 키입력 대기
```

심화예제 4.3.5 마우스 이벤트 및 그리기 종합 – 11.event_draw.py

```python
import numpy as np
import cv2

def onMouse(event, x, y, flags, param):
    global title, pt                                    # 전역 변수 참조

    if event == cv2.EVENT_LBUTTONDOWN:
        if pt[0] < 0: pt = (x, y)                       # 시작 좌표 지정
        else:
            cv2.rectangle(image, pt, (x, y), (255, 0, 0), 2)   # 파란색 사각형
            cv2.imshow(title, image)
            pt = (-1, -1)

    elif event == cv2.EVENT_RBUTTONDOWN:
        if pt[0] < 0: pt = (x, y)
        else:
            dx, dy = pt[0] - x, pt[1] - y               # 두 좌표 간 간격
            radius = int(np.sqrt(dx*dx + dy*dy))
            cv2.circle(image, pt, radius, (0, 0, 255), 2)      # 빨간색 원
            cv2.imshow(title, image)
            pt = (-1, -1)                               # 시작 좌표 초기화

image = np.full((300, 500, 3), (255, 255, 255), np.uint8)      # 흰색 배경 영상

pt = (-1, -1)                                           # 시작 좌표 초기화
title = "Draw Event"
cv2.imshow(title, image)
cv2.setMouseCallback(title, onMouse)                    # 마우스 콜백 함수 등록
cv2.waitKey(0)
```



영상 파일 처리

❖영상 읽어 오기

- 영상 파일을 읽어 행렬(메모리)에 저장
- imread(filename, flags)

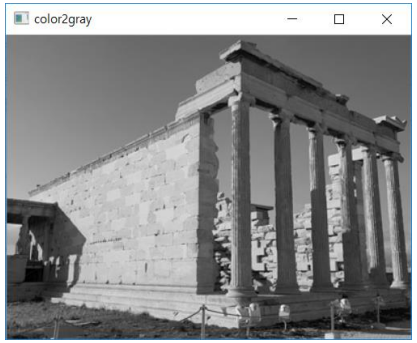
〈표 4.4.1〉 행렬의 컬러 타입 결정 상수

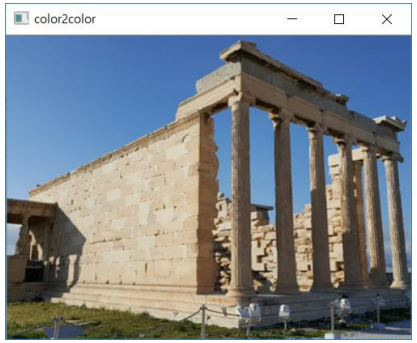
옵션	값	설명
cv2.IMREAD_UNCHANGED	-1	입력 파일에 정의된 타입의 영상을 그대로 반환(알파(alpha) 채널 포함)
cv2.IMREAD_GRAYSCALE	0	명암도(grayscale) 영상으로 변환하여 반환
cv2.IMREAD_COLOR	1	컬러 영상으로 변환하여 반환
cv2.IMREAD_ANYDEPTH	2	입력 파일에 정의된 깊이(depth)에 따라 16비트/32비트 영상으로 변환, 설정되지 않으면 8비트 영상으로 변환
cv2.IMREAD_ANYCOLOR	4	입력 파일에 정의된 타입의 영상을 반환

예제 4.4.1 영상파일 읽기1 – 12.read_image1.py

```python
import cv2

def print_matInfo(name, image):                    # 행렬 정보 출
    if image.dtype == 'uint8':      mat_type = 'CV_8U'
    elif image.dtype == 'int8':     mat_type = 'CV_8S'
    elif image.dtype == 'uint16':   mat_type = 'CV_16U'
    elif image.dtype == 'int16':    mat_type = 'CV_16S'
    elif image.dtype == 'float32':  mat_type = 'CV_32F'
    elif image.dtype == 'float64':  mat_type = 'CV_64F'
    nchannel = 3 if image.ndim == 3 else 1

    ## depth, channel 출력
    print("%12s: depth(%s), channels(%s) -> mat_type(%sC%d)"
          % (name, image.dtype, nchannel, mat_type,  nchannel))

title1, title2 = 'gray2gray', 'gray2color'                    # 윈도우 이름
gray2gray = cv2.imread("images/read_gray.jpg", cv2.IMREAD_GRAYSCALE)    # 명암도
gray2color = cv2.imread("images/read_gray.jpg", cv2.IMREAD_COLOR)       # 컬러 영상

## 예외처리 –영상파일 읽기 여부 조사
if gray2gray is None or gray2color is None :
    raise Exception("영상파일 읽기 에러")
```

```python
print("행렬 좌표 (100, 100) 화소값")
print("%s %s" % (title1, gray2gray[100, 100]))          # 행렬 내 한 화소값 표시
print("%s %s\n" % (title2, gray2color[100, 100]))

print_matInfo(title1, gray2gray)                        # 행렬 정보 출력 함수 호출
print_matInfo(title2, gray2color)

cv2.imshow(title1, gray2gray)                           # 행렬 정보를 영상으로 띄
cv2.imshow(title2, gray2color)
cv2.waitKey(0)
```





영상 파일 처리

❖영상을 파일에 쓰기

- 행렬(메모리) 영상을 파일에 저장
- imwrite(filename, img, params)
  - params: 압축방식에 사용되는 인수 쌍(paramid, paramValue)

〈표 4.4.3〉 압축 방식에 사용되는 params 인수 튜플(paramId, paramValue)의 예시

paramId	paramValue (기본값)	설명
cv2.IMWRITE_JPEG_QUALITY	0~100 (95)	JPG 파일 화질, 높은 값일수록 화질 좋음
cv2.IMWRITE_PNG_COMPRESSION	0~9 (3)	PNG 파일 압축 레벨, 높은 값일수록 용량은 적어지고, 압축 시간이 길어짐
cv2.IMWRITE_PXM_BINARY	0 or 1 (1)	PPM, PGM 파일의 이진 포맷 설정

예제 4.4.3 행렬 영상 저장1 – 15.write_image1.py

```
01  import cv2
02
03  image = cv2.imread("images/read_color.jpg", cv2.IMREAD_COLOR)
04  if image is None: raise Exception("영상파일 읽기 에러")          # 예외처리
05
06  params_jpg = (cv2.IMWRITE_JPEG_QUALITY, 10)                    # JPEG 화질 설정
07  params_png = [cv2.IMWRITE_PNG_COMPRESSION, 9]                  # PNG 압축 레벨 설정
08
09  ## 행렬을 영상파일로 저장
10  cv2.imwrite("images/write_test1.jpg", image)                   # 디폴트는 95
11  cv2.imwrite("images/write_test2.jpg", image, params_jpg)       # 지정한 화질로 저장
12  cv2.imwrite("images/write_test3.png", image, params_png)
13  cv2.imwrite("iamges/write_test4.bmp", image)                   # BMP 파일로 저장
14  print("저장 완료")
```

Summary

- ❖윈도우 제어
  - 윈도우 생성, 크기 변환, 이동, 삭제
- ❖이벤트 제어
  - 키보드, 마우스, 트랙바 이벤트 처리
    - cv2.waitKey(), cv2.waitKeyEx()
    - cv2.setMouseCallback(), cv2.createTrackbar()
- ❖그리기
  - 직선, 사각형, 원, 타원 그리기
    - cv2.line(), cv2.rectangle(), cv2.circle(), cv2.ellipse()
- ❖문자열 출력 cv2.put_text()