파이썬 활용 (#4/5)

2023. 02. 23

초빙교수 김현용
충북대학교 산업인공지능연구센터

CHUNGBUK NATIONAL UNIVERSITY

INDUSTRIAL AI RESEARCH CENTER

강의 일정

INDUSTRIAL AI RESEARCH CENTER

	1일차	2일차	3일차	4일차	5일차
주제	파이참과 엑셀파일	데이터 분석	웹 자동화	컴퓨터 비전	AI와 딥러닝
세부 내용	• 강사/강의 소개 • **PyCharm** • [실습] script 작성 • 가상환경 • 파일 경로 • 파일 입출력 • [실습] 파일 입출력 • **OpenPyXL** - 엑셀파일 입출력 - 셀 편집 - 차트 그리기 • [실습] 셀서식 지정 • [실습] 차트 그리기	• 데이터분석 • **NumPy** • 회귀분석 • [실습] 선형회귀 • **Pandas** • [실습] 데이터프레임 다루기 • **Matplotlib** • [실습] 시각화 • [실습] 회귀분석	• **웹스크래핑** • [실습] 기온 가져오기 • **Open API** (파파고) • **Selenium** • [실습] 구글 검색 • **PyAutoGui** • [실습] 구글 검색	• 컴퓨터 비전 개요 • **OpenCV** - 영상 입출력 - 영상 데이터 조작 • Matplotlib 영상표시 • OpenCV - 그리기 함수 - 동영상 입출력 • OpenCV - 히스토그램 - 이진화 - 에지 검출	• 인공신경망 • 딥러닝 • **PyTorch** • 객체검출 • **YOLOv8-객체검출** • [실습] 객체검출 • YOLOv8-객체분할 • chatGPT

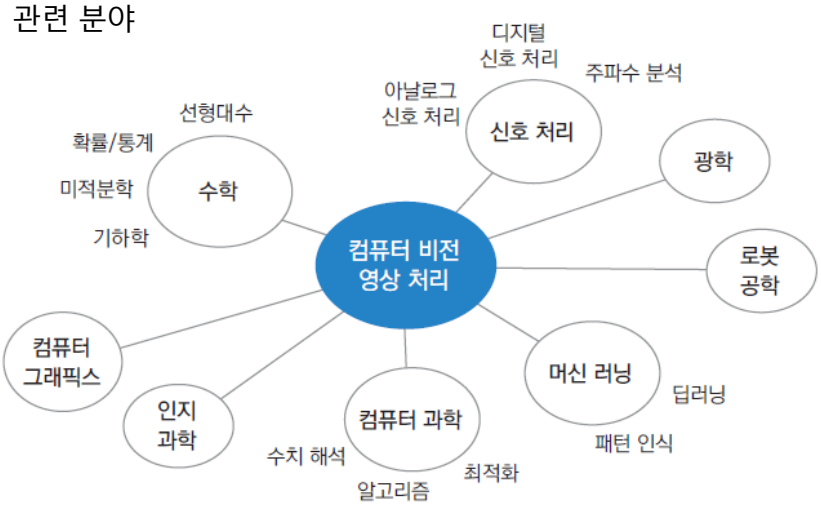
컴퓨터 비전 개요

INDUSTRIAL AI RESEARCH CENTER

▣ 컴퓨터/머신 비전(computer/machine vision) vs. 영상처리(image processing)

- 컴퓨터를 이용하여 정지 영상 또는 동영상으로부터 정보를 추출하는 학문
- 사람이 눈으로 사물을 보고 인지하는 작업을 컴퓨터가 수행하게 만드는 기술
  - 영상 획득 : 눈 → 카메라
  - 정보 추출(인지) : 뇌 → 알고리즘을 통해 컴퓨터가 수행

▣ 관련 분야



OpenCV

https://opencv.org/

참고문헌 (저자: 황선규 박사)

컴퓨터 비전 시스템

INDUSTRIAL AI RESEARCH CENTER

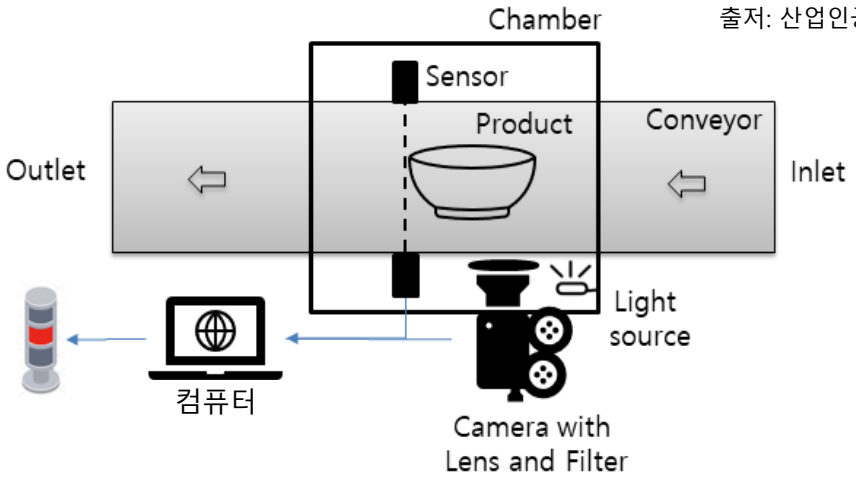
▣ 컴퓨터 비전 시스템

- 공장 자동화 : 제품의 불량 검사, 위치 확인, 치수 측정 등
- 높은 정확도와 빠른 처리 시간 요구 (C++)

▣ 딥러닝을 이용한 화장품 용기의 라벨 검사 시스템 개발

- HW : 저렴한 임베디드 디바이스 사용
- SW : 파이썬 → 처리속도 : 초당 10개 정도

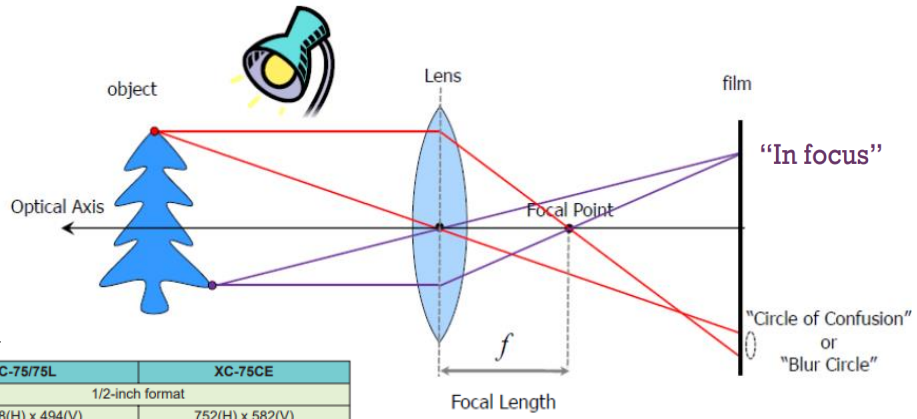
출저: 산업인공지능연구센터 홈페이지



카메라를 통한 영상 형성

▣ 카메라를 통한 영상의 형성 과정

- 조명에서 조사된 빛 중에서 반사된 빛이 센서를 자극하여 신호를 발생시킴
- 렌즈를 통해 빛을 집중시킴
- 2차원으로 배열된 이미지 센서를 통해 전기적 신호로 변환 [analog]
- 전기적 신호는 아날로그-디지털 변환기(ADC), 신호처리장치(ISP)를 통해 디지털 영상을 생성한 후 컴퓨터로 전송. (ISP : 화이트밸런스 조정, 색 보정, 잡음 제거 등 전처리 수행)



SONY社 CCD 카메라 모듈

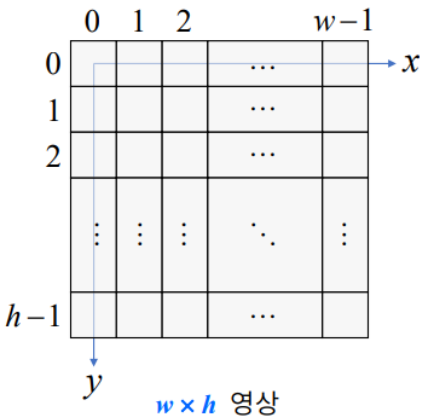
	XC-75/75L	XC-75CE
Optical size	1/2-inch format	
Effective picture elements	768(H) x 494(V)	752(H) x 582(V)
Total number of pixels	811(H) x 508(V)	795(H) x 596(V)
Chip size	7.95mm(H) x 6.45mm(V)	
Unit cell size	8.4μm(H) x 9.8μm(V)	8.6μm(H) x 8.3μm(V)

영상의 구조

▣ 영상(image)의 구성

- 화소(pixel, picture element) : 영상을 구성하는 최소 단위
- 영상(image) : 픽셀이 모인 2차원 행렬로 표현
- 채널(channel) : 컬러 영상인 경우 R, G, B 등의 복수 개의 채널로 구성
- 해상도(resolution) : ' 500만 화소(5MP) ' 는 영상이 500만 개의 화소로 구성

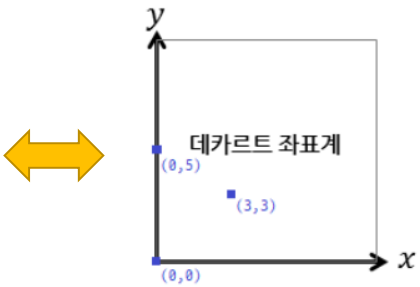
▣ 영상의 표현과 좌표계



$w \times h$ 영상
(w-by-h image)

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{bmatrix}$$

$M \times N$ 행렬
(m-by-n matrix)



수학의 (x, y) 좌표계

색좌표(공간)

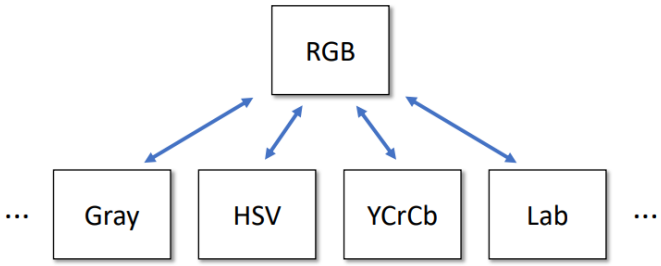
INDUSTRIAL AI RESEARCH CENTER

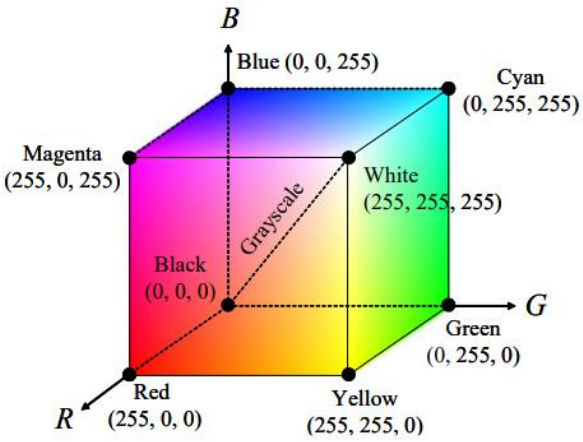
▣ 색좌표

- 관능적인 색상을 정량화하기 위한 수학적인 표현법
- RGB 색 공간 : 빛의 삼원색인 Red, Gree, Blue를 혼합하여 색상을 표현 (가산 혼합)
- Grayscale 색 공간: 색상 정보는 없고 밝기 정보만 표현

Intensity = 0.299R + 0.587G + 0.114B

- HSV 색 공간 : Hue(색상), Saturation(채도), Value(명도)
- YCrCb 색 공간 : Y(밝기) + Cr, Cb 색상 평면 *예) 컬러영상의 밝기 조절*
- OpenCV에서는 기본적으로 **BGR 컬러 영상**을 사용





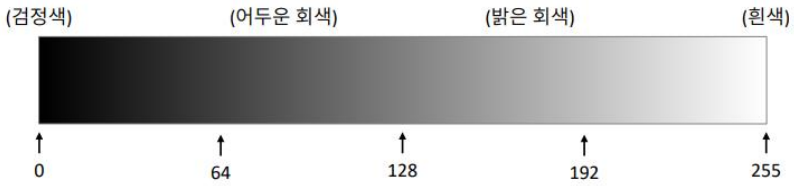
영상의 종류 (#1/2)

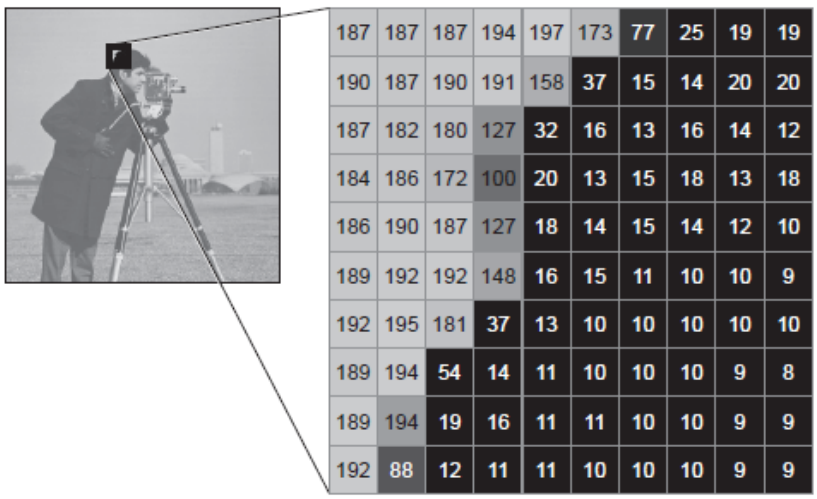
INDUSTRIAL AI RESEARCH CENTER

▣ 그레이스케일(grayscale, 명암) 영상

- 흑백 사진처럼 색상 정보가 없이 오직 밝기 정보만으로 구성된 영상
- 채널 : 1개

  → 밝기 정보를 256 단계로 표현
- 영상의 크기 : 가로 x 세로 Bytes



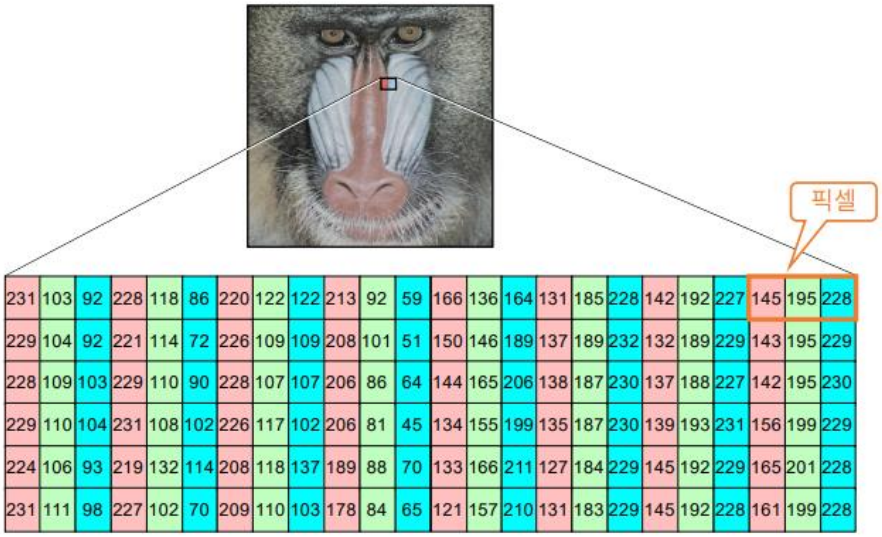


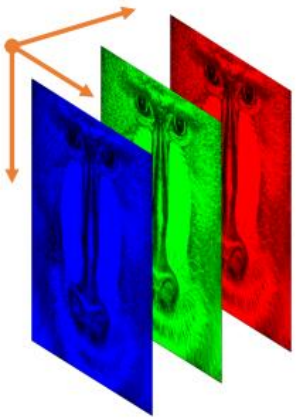
영상의 종류 (#2/2)

INDUSTRIAL AI RESEARCH CENTER

▣ 컬러(truecolor) 영상

- 채널 : 3개 (Red, Green, Blue) + 1개 (알파, 투명도)
- 각 채널로 RGB 색 성분을 256 단계로 표현 → 256^3 = 16,777,216 색상 표현 가능
- 영상의 크기 : 가로 x 세로 x 3채널 Bytes



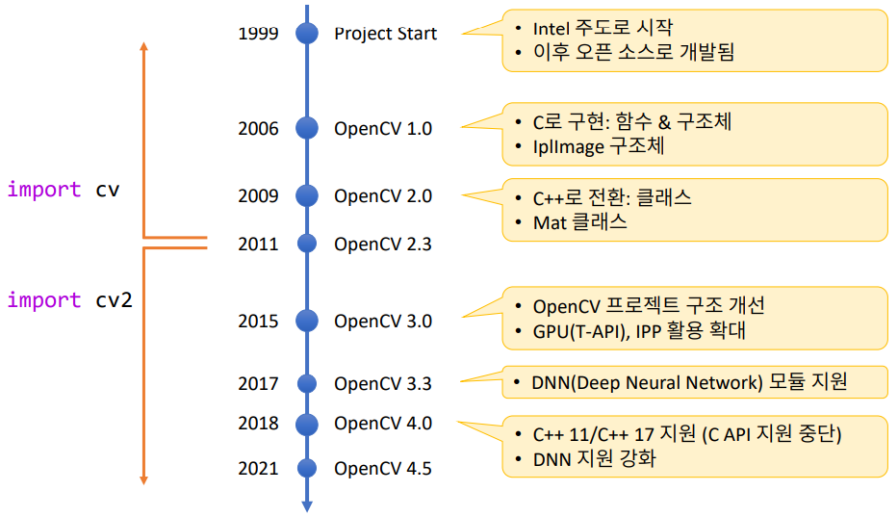


OpenCV - 개요

INDUSTRIAL AI RESEARCH CENTER

▣ OpenCV 개요

- 1999년부터 인텔(Intel)에서 개발하여 공개한 컴퓨터 비전 라이브러리
- 영상을 처리하려면 많은 계산량이 필요하므로 인텔 칩의 성능을 테스트할 목적으로 개발
- Open source : 학계 연구용이나 상업적인 용도로 자유롭게 사용 가능
- Computer vision : 컴퓨터 비전 및 머신 러닝 라이브러리임
- 2017년부터 심층 신경망(DNN, Deep Neural Network) 모델을 추론하는 기능도 제공
- C/C++로 작성되었지만 Python, Java, Matlab, JavaScript 등의 인터페이스도 제공



OpenCV - 설치

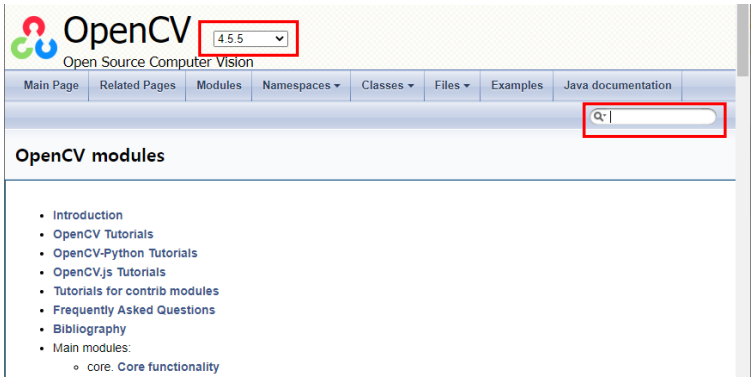
INDUSTRIAL AI RESEARCH CENTER

▣ OpenCV 구성 및 설치
- 기본(main) 모듈
- 추가(extra) 모듈 : 안정화되지 않은 최신 알고리즘, CUDA 관련 기능, 저작권 문제 등

▣ OpenCV 설치
- 기본 모듈 : **pip install opencv-python → import cv2**
- 추가 모듈 : **pip install opencv-contrib-python**
- 명령 프롬프트에서 설치 확인

```
C:₩Users₩sunkyoo>python
Python 3.7.7 (tags/v3.7.7:d7c567b08f, Mar 10 2
Type "help", "copyright", "credits" or "licens
>>> import cv2
>>> cv2.__version__
'4.1.0'
>>> exit()
```

▣ OpenCV 도움말 찾기
- http://docs.opencv.org/master/
- 버전 지정 & 검색 활용



▣ 클래스 객체 확인
- type(a) : 객체 종류(클래스)?
- dir(a) : 멤버 변수/함수 목록?
- help(a[.sort]) : 사용법?

OpenCV – 영상 읽고 쓰기

INDUSTRIAL AI RESEARCH CENTER

▣ 영상 불러오기

```
cv2.imread(filename, flags=None) -> retval
```

- filename: 불러올 영상 파일 이름 (문자열)
- flags: 영상 파일 불러오기 옵션 플래그

cv2.IMREAD_COLOR	BGR 컬러 영상으로 읽기 (기본값) shape = (rows, cols, 3)
cv2.IMREAD_GRAYSCALE	그레이스케일 영상으로 읽기 shape = (rows, cols)
cv2.IMREAD_UNCHANGED	영상 파일 속성 그대로 읽기 (e.g.) 투명한 PNG 파일: shape = (rows, cols, 4)

- retval: 불러온 영상 데이터 (numpy.ndarray)

- ndim: 차원 수. len(img.shape)과 같음.
- shape: 각 차원의 크기. (h, w) 또는 (h, w, 3)
  - (h, w) → 그레이스케일 영상
  - (h, w, 3) → 컬러 영상
- size: 전체 원소 개수
- dtype: 원소의 데이터 타입. 영상 데이터는 uint8.

▣ 영상 저장하기

```
cv2.imwrite(filename, img, params=None) -> retval
```

- filename: 저장할 영상 파일 이름 (문자열)
- img: 저장할 영상 데이터 (numpy.ndarray)
- params: 파일 저장 옵션 지정 (속성 & 값의 정수 쌍)
  e.g) [cv2.IMWRITE_JPEG_QUALITY, 90] : JPG 파일 압축률을 90%로 지정
- retval: 정상적으로 저장하면 True, 실패하면 False.

OpenCV - 자료형

INDUSTRIAL AI RESEARCH CENTER

▣ 데이터 자료형

OpenCV 자료형 (1채널)	NumPy 자료형	구분
cv2.CV_8U	numpy.uint8	8비트 부호없는 정수
cv2.CV_8S	numpy.int8	8비트 부호있는 정수
cv2.CV_16U	numpy.uint16	16비트 부호없는 정수
cv2.CV_16S	numpy.int16	16비트 부호있는 정수
cv2.CV_32S	numpy.int32	32비트 부호있는 정수
cv2.CV_32F	numpy.float32	32비트 부동소수형
cv2.CV_64F	numpy.float64	64비트 부동소수형
cv2.CV_16F	numpy.float16	16비트 부동소수형

- 그레이스케일 영상: cv2.CV_8UC1 → numpy.uint8, shape = (h, w)
- 컬러 영상: cv2.CV_8UC3 → numpy.uint8, shape = (h, w, 3)

▣ 실습/예제

```
# Color image
img_bgr = cv2.imread('cat.bmp')    # cv2.IMREAD_COLOR (default)
type(img_ bgr)                     # <class 'numpy.ndarray'>
img_bgr.dtype                      # dtype('uint8')
img_bgr.shape                      # (480, 640, 3)
len(img_bgr.shape)                 # 3

# Gray image
img_gray = cv2.imread('cat.bmp', cv2.IMREAD_GRAYSCALE)
img_gray.shape                     # (480, 640)
len(img_gray.shape)                # 2
```

```
h, w = img_bgr.shape[:2]       # h = 480, w = 640

t0 = time.time()
for j in range(h):
    for i in range(w):
        img_gray[j, i] = 255           # intensity
        img_bgr[j, i] = [0, 0, 255]    # [B, G, R]

# Vectorization : 140배 더 빠름 (1ms)
img_gray[:, :] = 255
img_bgr[:, :] = [0, 0, 255]
```


OpenCV – 윈도우 함수

INDUSTRIAL AI RESEARCH CENTER

▣ 새 창 생성

```
cv2.namedWindow(winname, flags=None) -> None
```

- winname: 창 고유 이름 (문자열)
- flags: 창 속성 지정 플래그

cv2.WINDOW_NORMAL	영상 크기를 창 크기에 맞게 지정
cv2.WINDOW_AUTOSIZE	창 크기를 영상 크기에 맞게 변경 (기본값)

▣ 창 위치 이동

```
cv2.moveWindow(winname, x, y) -> None
```

▣ 창 크기 변경

```
cv2.resizeWindow(winname, width, height) -> None
```

▣ 창 닫기

```
cv2.destroyWindow(winname) -> None
cv2.destroyAllWindows() -> None
```

▣ 영상 크기 변경

```
cv2.resize(src, dsize, dst=None, fx=None, fy=None, interpolation=None) -> img
```

- src/dst : 입력/출력 영상
- dsize: 결과영상 크기, (w, h) 튜플, (0, 0)이면 fx와 fy값을 이용하여 결정
- fx, fy: x와 y 방향 scale factor
- interpolation: 보간법 지정. 기본값은 cv2.INTER_LINEAR

OpenCV – 영상 표시

INDUSTRIAL AI RESEARCH CENTER

▣ 영상 표시하기

```
cv2.imshow(winname, mat) -> None
```

- winname: 영상을 출력할 대상 창 이름
- mat: 출력할 영상 데이터 (numpy.ndarray)
- 참고 사항
  - uint16, int32 자료형 행렬의 경우, 행렬 원소 값을 255로 나눠서 출력
  - float32, float64 자료형 행렬의 경우, 행렬 원소 값에 255를 곱해서 출력
  - 만약 winname에 해당하는 창이 없으면 창을 새로 만들어서 영상을 출력함
  - Windows 운영체제에서는 Ctrl + C (복사), Ctrl + S (저장) 지원
  - 실제로는 cv2.waitKey() 함수를 호출해야 화면에 영상이 나타남

▣ 키보드 입력

```
cv2.waitKey(delay=None) -> retval
```

- delay: 밀리초 단위 대기 시간. delay ≤ 0 이면 무한히 기다림. 기본값은 0.
- retval: 눌린 키 값(ASCII code). 키가 눌리지 않으면 -1.
- 참고 사항
  - cv2.waitKey() 함수는 OpenCV 창이 하나라도 있을 때 동작함
  - 특정 키 입력을 확인하려면 ord() 함수를 이용 : 27(ESC), 13(ENTER), 9(TAB)

```
while True:
    if cv2.waitKey() == ord('q'):
        break
```

```
if cv2.waitKey(1000) >= 0:
    break
```


OpenCV – 색 변환

INDUSTRIAL AI RESEARCH CENTER

▣ 색 공간 변환

```
cv2.cvtColor(src, code, dst=None, dstCn=None) -> dst
```

- src: 입력 영상
- code: 색 변환 코드

cv2.COLOR_BGR2GRAY / cv2.COLOR_GRAY2BGR	BGR ↔ GRAY
cv2.COLOR_BGR2RGB / cv2.COLOR_RGB2BGR	BGR ↔ RGB
cv2.COLOR_BGR2HSV / cv2.COLOR_HSV2BGR	BGR ↔ HSV
cv2.COLOR_BGR2YCrCb / cv2.COLOR_YCrCb2BGR	BGR ↔ YCrCb

- dstCn: 결과 영상의 채널 수. 0이면 자동 결정.
- dst: 출력 영상

▣ 실습/예제

b_plane = src[:, :, 0] g_plane = src[:, :, 1] r_plane = src[:, :, 2]

B 평면

G 평면

R 평면

Matplotlib - 영상 표시 (#1/2)

INDUSTRIAL AI RESEARCH CENTER

▣ Matplotlib

- 함수 그래프, 차트(chart), 히스토그램(histogram) 등의 다양한 그리기 기능을 제공하는 모듈

```
> pip install matplotlib
```

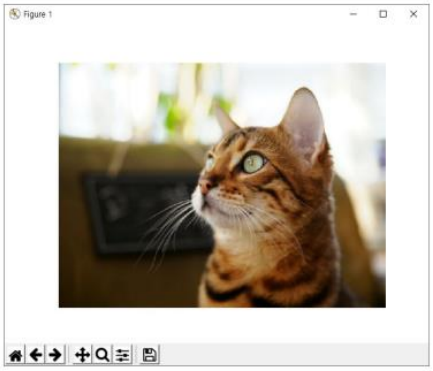
▣ 컬러 영상 출력

- cv2.imread() 함수로 불러온 영상의 색상 정보는 BGR 순서
- Matplotlib의 색상 정보는 RGB 순서

→ cv2.cvtColor(img, cv2.COLOR_BGR2RGB) 함수로 변경

▣ 그레이스케일 영상 출력

- plt.imshow() 함수에서 컬러맵을 cmap='gray' 으로 지정




```
# Color 영상 출력
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
plt.imshow(img_rgb)
plt.axis('off')
plt.show()
```

```
# Gray 영상 출력
plt.imshow(img_gray, cmap='gray')
plt.axis('off')
plt.show()
```


Matplotlib – 영상 표시 (#2/2)

INDUSTRIAL AI RESEARCH CENTER

▣ 실습 예제

```
img_bgr = cv2.imread('cat.bmp')
img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
img_gray = cv2.imread('cat.bmp', cv2.IMREAD_GRAYSCALE)

fig, ax = plt.subplots(3, 1, figsize=(4, 7))

# BGR image 출력
ax[0].imshow(img_bgr)
ax[0].axis('off')                    # 축의 눈금 생략
ax[0].set_title('BGR image')

# RGB image 출력
ax[1].imshow(img_rgb)
ax[1].axis('off')
ax[1].set_title('RGB image')

# Grayscale image 출력
ax[2].imshow(img_gray, cmap='gray')
ax[2].axis('off')
ax[2].set_title('Gray image')

plt.tight_layout()
plt.show()
```

OpenCV - 영상 생성

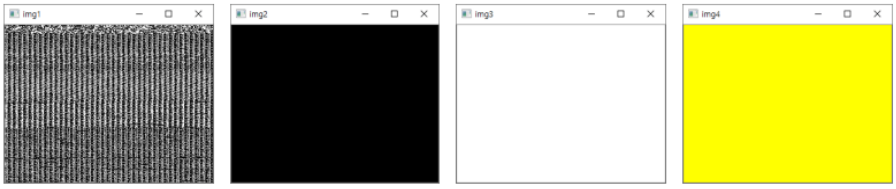
▣ 영상 초기화

```
numpy.empty(shape, dtype=float, ...) -> arr
numpy.zeros(shape, dtype=float, ...) -> arr
numpy.ones(shape, dtype=None, ...) -> arr
numpy.full(shape, fill_value, dtype=None, ...) -> arr
```

- shape: 각 차원의 크기. (h, w) 또는 (h, w, 3)
- dtype: 원소의 데이터 타입. 일반적인 영상이면 numpy.uint8 지정
- arr: 생성된 영상(numpy.ndarray)
- 참고사항:
  - numpy.empty() 함수는 임의의 값으로 초기화된 배열을 생성
  - numpy.zeros() 함수는 0으로 초기화된 배열을 생성
  - numpy.ones() 함수는 1로 초기화된 배열을 생성
  - numpy.full() 함수는 fill_value로 초기화된 배열을 생성

▣ 실습/예제

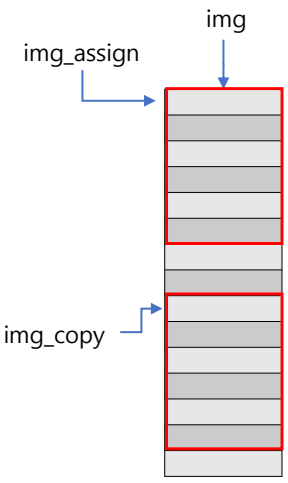
```
img1 = np.empty((480, 640, 3), np.uint8)                # random image
img2 = np.zeros((480, 640, 3), np.uint8)                # black image
img3 = np.ones((480, 640, 3), np.uint8) * 255           # white image
img4 = np.full((480, 640, 3), (255, 255, 0), np.uint8)  # color image
```



OpenCV - 얕은 복사와 깊은 복사

▣ 참조(얕은 복사) vs (깊은) 복사

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('cat.bmp')

img_assign = img          # shallow copy (reference) : 이름만 다를 뿐, 같은 변수
img_copy = img.copy()     # deep copy : 완전히 다른 변수

img[100:200, 200:300] = 0                # black
img_assign[200:300, 300:400] = 255       # white
img_copy[300:400, 400:500] = [0, 0, 255] # red

id(img)          # 1557621973392 (원본)
id(img_assign)   # 1557621973392 (원본과 동일)
id(img_copy)     # 1557606882192
```



원본 얕은 복사 깊은 복사

OpenCV - 그리기 함수 (#1/3)

INDUSTRIAL AI RESEARCH CENTER

▣ 직선 그리기

```
cv2.line(img, pt1, pt2, color, thickness=None, lineType=None,
         shift=None) -> img
```

- img: 그림을 그릴 영상
- pt1, pt2: 직선의 시작점과 끝점. (x, y) 튜플.
- color: 선 색상 또는 밝기. (B, G, R) 튜플 또는 정수값.
- thickness: 선 두께. 기본값은 1.
- lineType: 선 타입. cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA 중 선택. 기본값은 cv2.LINE_8
- shift: 그리기 좌표 값의 축소 비율. 기본값은 0.

▣ 사각형 그리기

```
cv2.rectangle(img, pt1, pt2, color, thickness=None, lineType=None,
              shift=None) -> img
cv2.rectangle(img, rec, color, thickness=None, lineType=None,
              shift=None) -> img
```

- img: 그림을 그릴 영상
- pt1, pt2: 사각형의 두 꼭지점 좌표. (x, y) 튜플.
- rec: 사각형 위치 정보. (x, y, w, h) 튜플.
- color: 선 색상 또는 밝기. (B, G, R) 튜플 또는 정수값.
- thickness: 선 두께. 기본값은 1. 음수(-1)를 지정하면 내부를 채움.

OpenCV - 그리기 함수 (#2/3)

INDUSTRIAL AI RESEARCH CENTER

▣ 원 그리기

```
cv2.circle(img, center, radius, color, thickness=None, lineType=None,
           shift=None) -> img
```

- img: 그림을 그릴 영상
- center: 원의 중심 좌표. (x, y) 튜플.
- radius: 원의 반지름

▣ 타원 그리기

```
cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color,
thickness=None, lineTpye=None, shift=None) -> img
```

- axes: 타원의 절반 크기 (x축 반지름, y축 반지름)
- angle: 타원의 기울기 (3시 방향이 0, 시계방향으로 증가)
- start/endAngle: 기울기(angle)을 기준으로 타원을 그리는 시작/종료 각도

▣ 다각형 그리기

```
cv2.polylines(img, pts, isClosed, color, thickness=None, lineType=None,
              shift=None) -> img
```

- img: 그림을 그릴 영상
- pts: 다각형 외곽 점들의 좌표 배열. numpy.ndarray의 리스트.
  (e.g.) [np.array([[10, 10], [50, 50], [10, 50]], dtype=np.int32)]
- isClosed: 폐곡선 여부. True 또는 False 지정.

OpenCV - 그리기 함수 (#3/3)

INDUSTRIAL AI RESEARCH CENTER

▣ 문자열 출력

```
cv2.putText(img, text, org, fontFace, fontScale, color, thickness=None,
            lineType=None, bottomLeftOrigin=None) -> img
```

- img: 그림을 그릴 영상
- text: 출력할 문자열
- org: 영상에서 문자열을 출력할 위치의 좌측 하단 좌표. (x, y) 튜플.
- fontFace: 폰트 종류. cv2.FONT_HERSHEY_ 로 시작하는 상수 중 선택
- fontScale: 폰트 크기 확대/축소 비율
- color: 선 색상 또는 밝기. (B, G, R) 튜플 또는 정수값.
- thickness: 선 두께. 기본값은 1. 음수(-1)를 지정하면 내부를 채움.
- lineType: 선 타입. cv2.LINE_4, cv2.LINE_8, cv2.LINE_AA 중 선택.
- bottomLeftOrigin: True이면 영상의 좌측 하단을 원점으로 간주. 기본값은 False.

▣ fontFace 상수

FONT_HERSHEY_SIMPLEX

FONT_HERSHEY_PLAIN

FONT_HERSHEY_DUPLEX

FONT_HERSHEY_COMPLEX

FONT_HERSHEY_TRIPLEX

FONT_HERSHEY_COMPLEX_SMALL

FONT_HERSHEY_SCRIPT_SIMPLEX

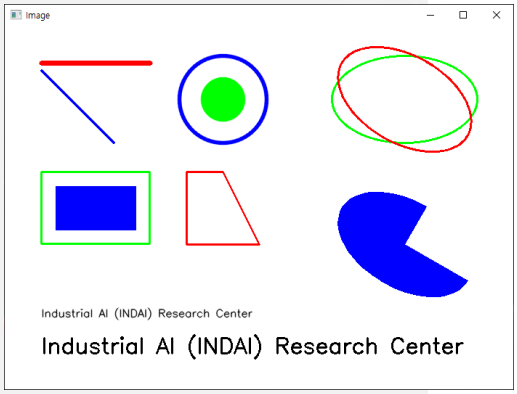
FONT_HERSHEY_SCRIPT_COMPLEX

FONT_HERSHEY_COMPLEX | FONT_ITALIC

[실습] 그리기

INDUSTRIAL AI RESEARCH CENTER

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

BLUE = (255, 0, 0);       GREEN = (0, 255, 0);       RED = (0, 0, 255)      # color 설정

img = np.full((500, 700, 3), 255, np.uint8)

cv2.line(img, (50, 50), (200, 50), RED, 5)
cv2.line(img, (50, 60), (150, 160), BLUE, 2)
cv2.rectangle(img, (50, 200, 150, 100), GREEN, 2)
cv2.rectangle(img, (70, 220), (180, 280), BLUE, -1)
cv2.circle(img, (300, 100), 60, BLUE, 3, cv2.LINE_AA)
cv2.circle(img, (300, 100), 30, GREEN, -1, cv2.LINE_AA)
cv2.ellipse(img, (550, 100), (100, 60), 0, 0, 360, GREEN, 2)
cv2.ellipse(img, (550, 100), (100, 60), 30, 0, 360, RED, 2)
cv2.ellipse(img, (550, 300), (100, 60), 30, 0, 270, BLUE, -1)
pts = np.array([[250, 200], [300, 200], [350, 300], [250, 300]])
cv2.polylines(img, [pts], True, RED, 2)
text = 'Industrial AI (INDAI) Research Center'                    # 문자 출력
cv2.putText(img, text, (50, 400), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1, cv2.LINE_AA)
cv2.putText(img, text, (50, 450), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2, cv2.LINE_4)

cv2.imshow('Image', img);              cv2.waitKey()
cv2.destroyAllWindows()
```



OpenCV - 동영상 획득

INDUSTRIAL AI RESEARCH CENTER

▣ 카메라 객체 생성

```
cap = cv2.VideoCapture(filename/device)
```

- filename: 동영상 파일, 정지영상 시퀀스(img_%2d.jpg), 비디오 스트림 URL 등
- device: 동영상 캡처 장치의 id (카메라 한 대만 연결된 경우 0)
- cap: cv2.VideoCapture 객체

▣ 동영상 캡쳐 연결여부 확인

```
cap.isOpened() -> retval (True/False)
```

- retval: 성공하면 True, 실패하면 False

▣ 프레임 읽기

```
retval, frame = cap.read()
```

- retval: 성공하면 True, 실패하면 False
- frame: 현재 프레임 (numpy.array)

▣ 객체 해제

```
cap.release() -> None
```

- 동영상 파일이나 장치를 해제
  (클래스 소멸자에 의해 자동 호출되므로 명시적으로 수행하지 않아도 됨)

OpenCV - 동영상 속성

INDUSTRIAL AI RESEARCH CENTER

▣ 장치 속성값 참조

```
cap.get(propId) -> retval
```

- propId: 속성 상수

CAP_PROP_FRAME_WIDTH	프레임 가로 크기
CAP_PROP_FRAME_HEIGHT	프레임 세로 크기
CAP_PROP_FPS	초당 프레임 수
CAP_PROP_FRAME_COUNT	비디오 파일의 총 프레임 수
CAP_PROP_POS_MSEC	밀리초 단위로 현재 위치
CAP_PROP_POS_FRAMES	현재 프레임 번호
CAP_PROP_EXPOSURE	노출

- retval: 성공하면 해당 속성 값, 실패하면 0.

▣ 장치 속성값 설정

```
cap.set(propId, value) -> retval
```

- propId: 속성 상수
- value: 속성값
- retval: 성공하면 True, 실패하면 False

OpenCV - 동영상 저장

INDUSTRIAL AI RESEARCH CENTER

▣ 동영상 저장 객체 생성

```
out = cv2.VideoWriter(filename, fourcc, fps, frameSize, isColor=None)
```

- filename: 비디오 파일명 (e.g. 'sample.mp4')
- fourcc(four character code): 동영상의 코덱, 압축방식, 색상, 픽셀 포맷 등 정의

cv2.VideoWriter_fourcc(*'DIVX')	DIXX MPEG-4 코덱
cv2.VideoWriter_fourcc(*XVID')	XVID MPEG-4 코덱
cv2.VideoWriter_fourcc(*FMP4')	FFMPEG MPEG-4 코덱
cv2.VideoWriter_fourcc(*X264')	H.264/AVC 코덱
cv2.VideoWriter_fourcc(*MJPG')	Motion-JPEG 코덱

(코덱은 별도설치)

- fps: 초당 프레임수 (e.g. 30)
- frameSize: 프레임 크기, (width, height) 튜플
- isColor: 컬러 영상이면 True, 그렇지 않으면 False
- out: cv2.VideoWriter 객체

▣ 동영상 저장 객체 연결여부 확인

```
out.isOpened() -> retval (성공하면 True, 실패하면 False)
```

▣ 프레임 저장

```
out.Write(frame) -> None
```


[실습] 웹캠 영상 저장하기

INDUSTRIAL AI RESEARCH CENTER

```python
import sys, time, cv2

cap = cv2.VideoCapture(0)      # 기본 카메라 장치 열기
if not cap.isOpened():
    sys.exit('카메라 연결 실패')

w = round(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
h = round(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = round(cap.get(cv2.CAP_PROP_FPS))
fourcc = cv2.VideoWriter_fourcc(*'DIVX')   # *'DIVX' == 'D', 'I', 'V', 'X'
out = cv2.VideoWriter('output.avi', fourcc, fps, (w, h))

frame_count = 0
while True:
    ret, frame = cap.read()
    if not ret:
        print('프레임 획득 실패');       break

    frame_count += 1
    frame = cv2.flip(frame, 1)  # 대칭 : 0(상하), 1(좌우)
    text = f'{frame_count} frame'
    cv2.putText(frame, text, (30, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 1, cv2.LINE_AA)
    cv2.imshow('Webcam: {} x {} x {}fps'.format(w, h, fps), frame)

    inversed_img = ~frame      # 반전 -> 저장
    out.write(inversed_img)

    key = cv2.waitKey(1)
    if key == 27 or frame_count == 300:  # ESC를 누르면 while 루프 종료
        time.sleep(3);        break

cap.release()
out.release()
```

[실습] 동영상 재생하기

```
import sys, time, cv2

cap = cv2.VideoCapture('output.avi')
if not cap.isOpened():
    sys.exit('동영상 읽기 실패')

w = round(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
h = round(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = round(cap.get(cv2.CAP_PROP_FPS))
delay = int(1000 / fps)
total_frames = round(cap.get(cv2.CAP_PROP_FRAME_COUNT))

while True:
    ret, frame = cap.read()
    if not ret:
        print('프레임 획득 실패')
        break

    text = '{}/{} frames'.format(round(cap.get(cv2.CAP_PROP_POS_FRAMES)), total_frames)
    cv2.putText(frame, text, (30, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 1, cv2.LINE_AA)
    cv2.imshow('Video: {} x {} x {}fps'.format(w, h, fps), frame)

    key = cv2.waitKey(delay)
    if key == ord('s'):        # 잠시 멈춤
        cv2.waitKey()

cap.release()
time.sleep(3)
cv2.destroyAllWindows()
```


OpenCV - Trackbar

▣ 트랙바(Trackbar)

- 프로그램 동작 중 사용자가 지정한 범위 안의 값을 선택할 수 있는 컨트롤

▣ 트랙바 생성 함수

```
cv2.createTrackbar(trackbarName, windowName, value, count, onChange) -> None
```

- trackbarName: 트랙바 이름
- windowName: 트랙바를 생성할 창 이름.
- value: 트랙바 위치 초기값
- count: 트랙바 최댓값. 최솟값은 항상 0.
- onChange: 트랙바 위치가 변경될 때마다 호출할 콜백 함수 이름

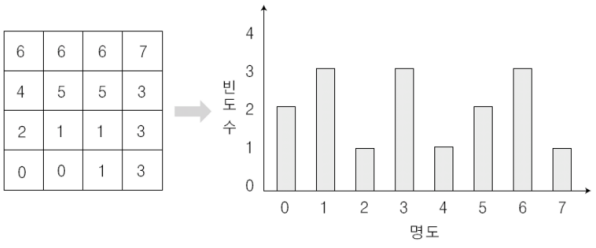
트랙바 이벤트 콜백 함수는 다음 형식을 따름.

```
onChange(pos) -> None
```

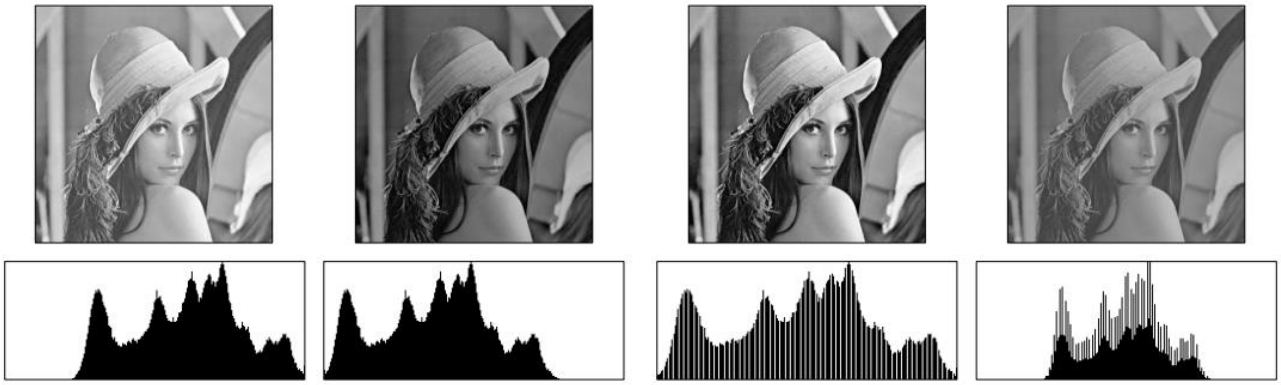
OpenCV – Histogram (#1/2)

▣ 히스토그램(histogram)

- 영상의 픽셀값의 분포, 즉 (빈)도수(frequency) 분포(표)를 알기 쉽게 그래프로 나타낸 것
- 화질 분석/개선
- 이진화에 활용



(a) 입력 영상 (b) 히스토그램



밝은 영상 어두운 영상 높은 명암비(contrast) 낮은 명암비(contrast)

OpenCV – Histogram (#2/2)

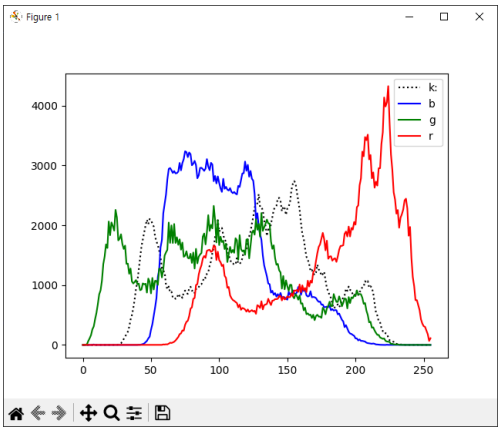
▣ 히스토그램 그리기

```
cv2.calcHist(images, channels, mask, histSize, ranges, hist=None,
             accumulate=None) -> hist
```

- images: 입력 영상 목록. 리스트, 튜플 형식 → [img1]
- channels: 히스토그램을 계산할 차원 목록. 리스트, 튜플 형식 → BG는 [0, 1]
- mask: 특정 영역만 계산할 경우 사용. 전체 영상에 대해서는 None 지정
- histSize: 각 히스토그램 계급(bin)의 개수 → [256]
- ranges: count할 히스토그램의 범위 → [0, 128]이면 128 이상은 세지 않음
- hist: 계산된 히스토그램 (numpy.ndarray)
- accumulate: 누적 플래그. 누적하려면 True, 새로 만드려면 False

▣ 실습/예제

```
img = cv2.imread('lenna_color.png')

colors = ['k', 'b', 'g', 'r']
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
imgs = [img_gray, img[:, :, 0], img[:, :, 1], img[:, :, 2]]

for p, c in zip(imgs, colors):
    hist = cv2.calcHist([p], [0], None, [256], [0, 256])
    plt.plot(hist, color=c, label=c)
plt.legend()
plt.show()
```



OpenCV - 영상 이진화 (#1/4)

INDUSTRIAL AI RESEARCH CENTER

▣ 영상의 이진화(binarization)

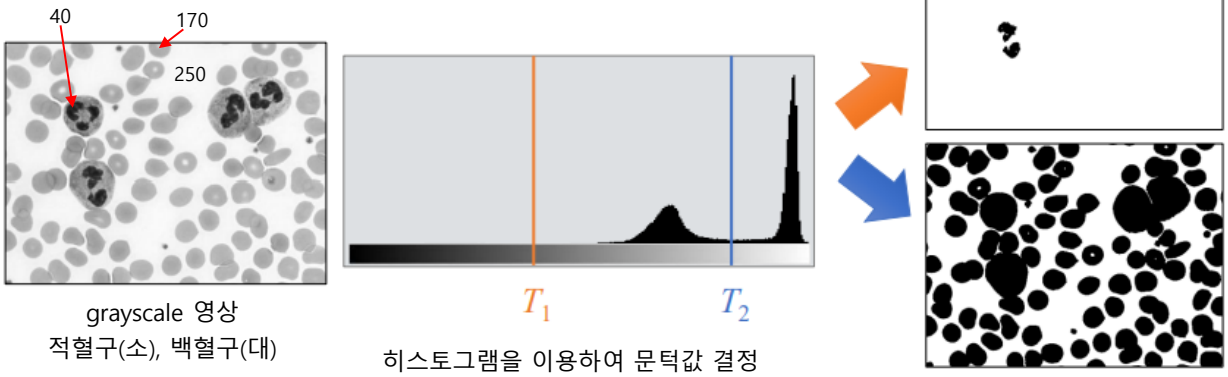
- 영상의 화소값을 0 또는 255(1)로 만드는 것 → 이진 영상(binary image)
- **grayscale 영상**의 이진화

$$g(x,y)=\begin{cases}0 & \text{if } f(x,y)\le T\\ 255 & \text{if } f(x,y)>T\end{cases}$$

- T: 임계값, 문턱치, threshold

- 배경(background) vs. 객체(object) → 관심영역(ROI, region of interest) 추출

▣ 히스토그램을 이용한 이진화



grayscale 영상
적혈구(소), 백혈구(대)

히스토그램을 이용하여 문턱값 결정

이진 영상

OpenCV - 영상 이진화 (#2/4)

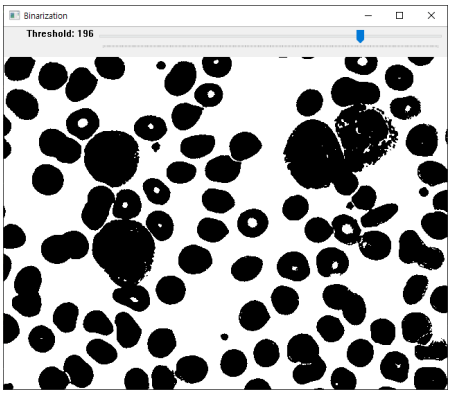
INDUSTRIAL AI RESEARCH CENTER

▣ 문턱값 함수

```
cv2.threshold(src, thresh, maxval, type, dst=None) -> retval, dst
```

- src: 입력 영상. 다채널, 8비트 또는 32비트 실수형.
- thresh: 사용자 지정 임계값 (Ostu의 경우 0)
- maxval: cv2.THRESH_BINARY 또는 cv2.THRESH_BINARY_INV 방법 사용 시 최댓값. 보통 255로 지정.
- type: cv2.THRESH_ 로 시작하는 플래그. 임계값 함수 동작 지정 또는 자동 임계값 결정 방법 지정
- retval: 사용된 임계값
- dst: 출력 영상. src와 동일 크기, 동일 타입, 같은 채널 수.

▣ 실습/예제



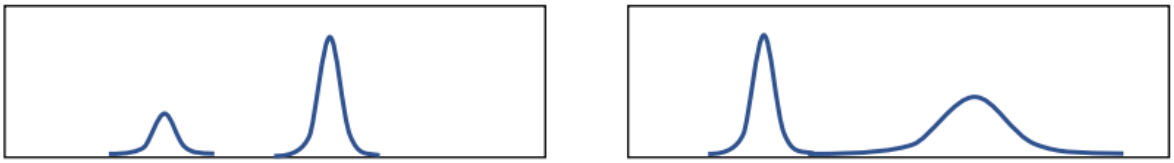

```
img = cv2.imread('cells.png', cv2.IMREAD_GRAYSCALE)

def on_change(pos):       # 트랙바 콜백함수
    _, dst = cv2.threshold(img, pos, 255, cv2.THRESH_BINARY)
    cv2.imshow('Binarization', dst)

cv2.namedWindow('Binarization')
cv2.createTrackbar('Threshold', 'Binarization', 0, 255, on_change)
cv2.setTrackbarPos('Threshold', 'Binarization', 128)  # 트랙바 초기값 설정

cv2.waitKey()
cv2.destroyAllWindows()
```

OpenCV - 영상 이진화 (#3/4)

INDUSTRIAL AI RESEARCH CENTER

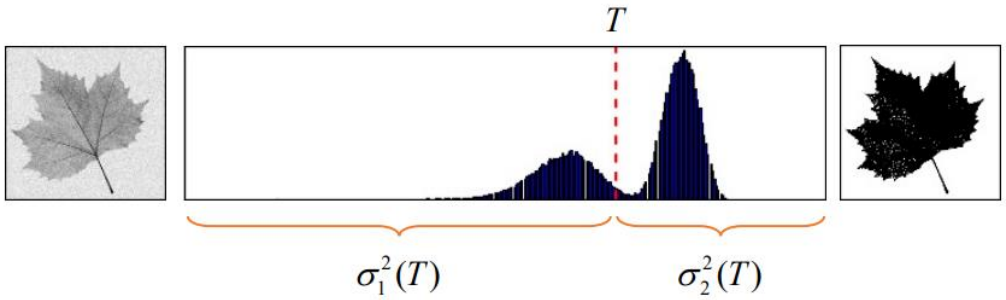
▣ 오츄(Otsu)의 이진화 (1979)

- 입력 영상이 배경과 객체 두 class로 구성되어 있다고 가정 → bimodal histogram



- 최적화 알고리즘 적용

→ 임의의 임계값 T에 의해 구분되는 두 픽셀 분포의 분산이 최소가 되는 최적의 T를 결정



- 점화식(recursion)을 이용하여 빠른 계산 알고리즘을 제시

OpenCV - 영상 이진화 (#4/4)

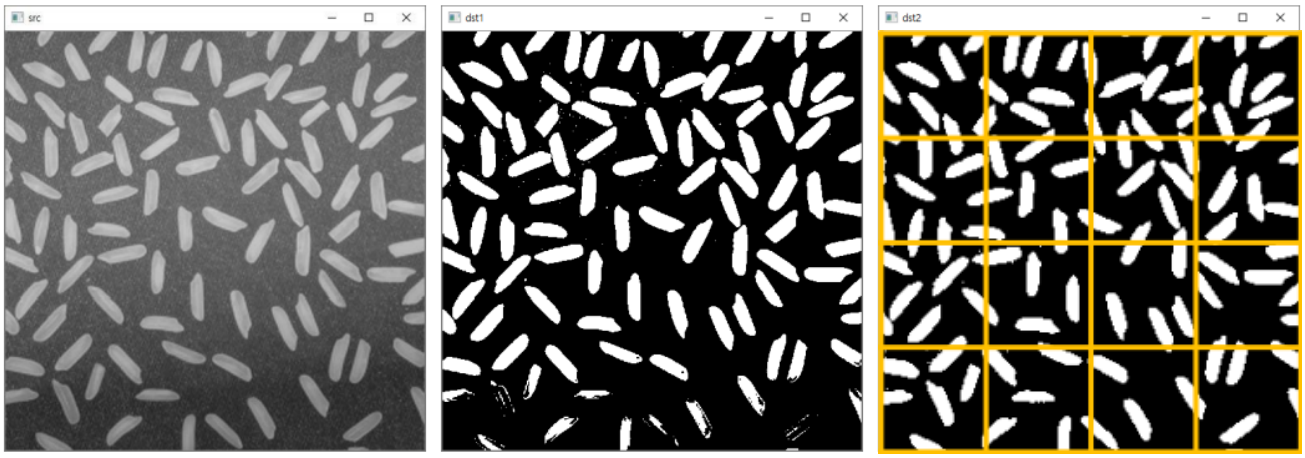
INDUSTRIAL AI RESEARCH CENTER

▣ 지역 이진화 (vs. 전역 이진화)

- 조명이 불균일한 경우 영상을 몇 개로 분할하여 각 구역에 대해 이진화 수행

▣ 실습/예제

```
img = cv2.imread('rice.png', cv2.IMREAD_GRAYSCALE)
th, dst = cv2.threshold(img, 0, 255, cv2.THRESH_OTSU)
print("Otsu's threshold:", th)          # 131.0
```



원본 영상

전역 이진화
cv2.threshold

지역 이진화
cv2.adaptiveThreshold

OpenCV - 에지 검출

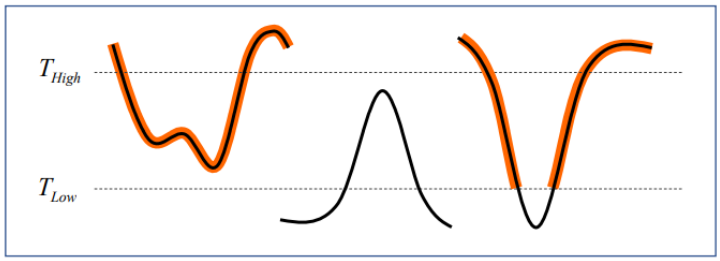
INDUSTRIAL AI RESEARCH CENTER

▣ 에지(edge) 검출

- 에지: 배경과 객체, 객체와 객체의 경계 → 특징 : 화소값이 급변하는 부분
- 1차 미분을 이용 : Prewitt, Sobel 필터 등
- 2차 미분을 이용 : Laplacian, LoG(Laplacian of Gaussian), DoG(Difference of Gaussian) 등
- **Canny 에지 검출**(1986)

캐니 에지 검출 과정

- 히스테리시스 에지 트래킹 (Hysteresis edge tracking)
  - 두 개의 임계값을 사용: T_{Low}, T_{High}
  - 강한 에지: $\|f\| \geq T_{High}$ → 최종 에지로 선정
  - 약한 에지: $T_{Low} \leq \|f\| < T_{High}$
    → 강한 에지와 연결되어 있는 픽셀만 최종 에지로 선정



[실습] 에지 검출

INDUSTRIAL AI RESEARCH CENTER

▣ 캐니 에지 검출

```
cv2.Canny(image, threshold1, threshold2, edges=None, apertureSize=None,
          L2gradient=None) -> edges
```

- image: 입력 영상
- threshold1: 하단 임계값
- threshold2: 상단 임계값

} threshold1:threshold2 = 1:2 또는 1:3

- edges: 에지 영상
- apertureSize: 소벨 연산을 위한 커널 크기. 기본값은 3.
- L2gradient: True이면 L2 norm 사용, False이면 L1 norm 사용. 기본값은 False.

L_2 norm $= \sqrt{(dI/dx)^2 + (dI/dy)^2}$, L_1 norm $= |dI/dx| + |dI/dy|$

▣ 실습/예제

```
img = cv2.imread('lenna_color.png',
                 cv2.IMREAD_GRAYSCALE)

dst = cv2.Canny(img, 50, 150)

cv2.imshow('Canny', dst)
cv2.waitKey()

cv2.destroyAllWindows()
```

원본 영상

Canny 에지 검출


감사합니다
Q & A