



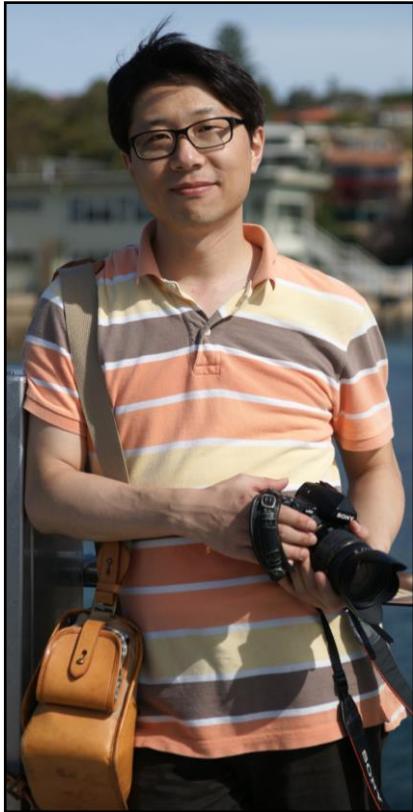
파이썬 OpenCV 프로그래밍 입문과 활용

1. OpenCV 개요

『OpenCV 4로 배우는 컴퓨터 비전과 머신 러닝』 저자

황 선 규 / 공학박사

강사 소개



□ 소개

- Name: 황선규
- E-mail: sunkyoo.hwang@gmail.com
- Blog: <https://sunkyoo.github.io/>

□ 약력

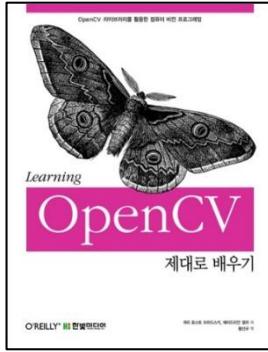
- ~ 2006년 8월: 한양대학교 공학박사
- ~ 2007년 11월: 뉴질랜드 캔터베리 대학교 HIT Lab NZ Postdoctoral Fellow
- ~ 2009년 3월: 한양대학교 연구교수
- ~ 2016년 3월: LG전자 MC연구소 전략 스마트폰 카메라 기능 & 카메라 프레임워크 개발
- 2016년 9월 ~: 패스트캠퍼스 『OpenCV로 배우는 컴퓨터 비전 프로그래밍 캠프』 강의 진행 중

강사 소개

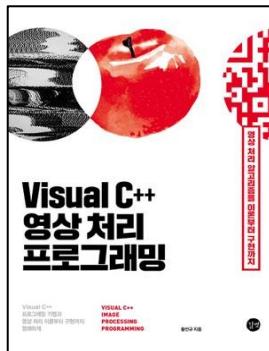
□ 저서/역서



2007



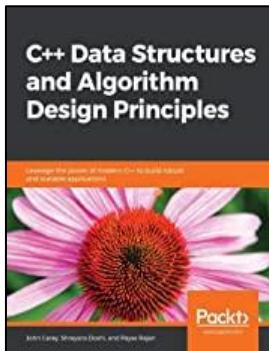
2009



2015



2019



2020

□ 강의 경력

- (주) 고영테크놀로지 (2017.05)
- 국민대학교 특강 (2017.08)
- (주) 기가비스 (2017.10)
- (주) 삼성전자 해외법인 로봇 비전 초청 (2018.03)
- (주) 골프존 (2018.06)
- 한국미래기술교육연구원 (2019.01) [[관련기사](#)]
- (주) 삼성전자 비전 알고리즘 (2019.05, 10, 11)
- (주) 삼성 디스플레이 특강 (2019.10)
- 서원대학교 특강 (2019.11)
- (주) 현대모비스 (알고리즘, 코딩테스트) (2019.12)



컴퓨터 비전이란?



<http://szeliski.org/Book/>



WIKIPEDIA
The Free Encyclopedia

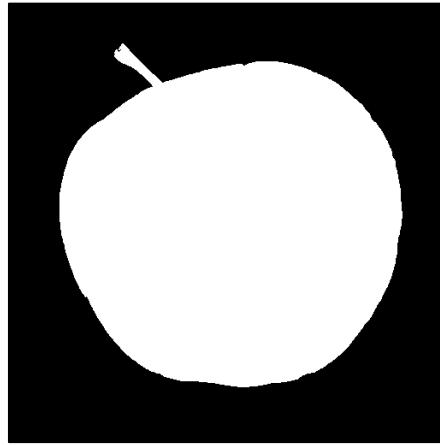
Computer vision is an interdisciplinary field that deals with how computers can be made to gain *high-level understanding* from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do.

https://en.wikipedia.org/wiki/Computer_vision

컴퓨터 비전이란?



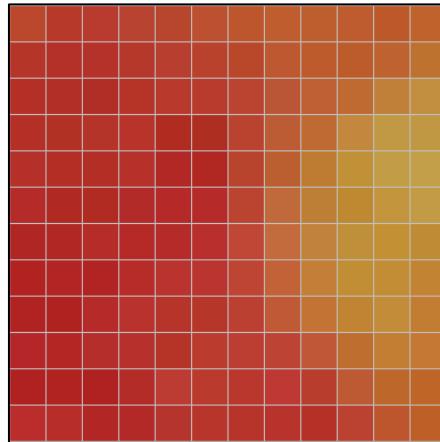
사과?



둥글다?



꼭지 모양으로 구분?



빨간색이다?



사과가 몇 개??

컴퓨터 비전 vs. 영상 처리

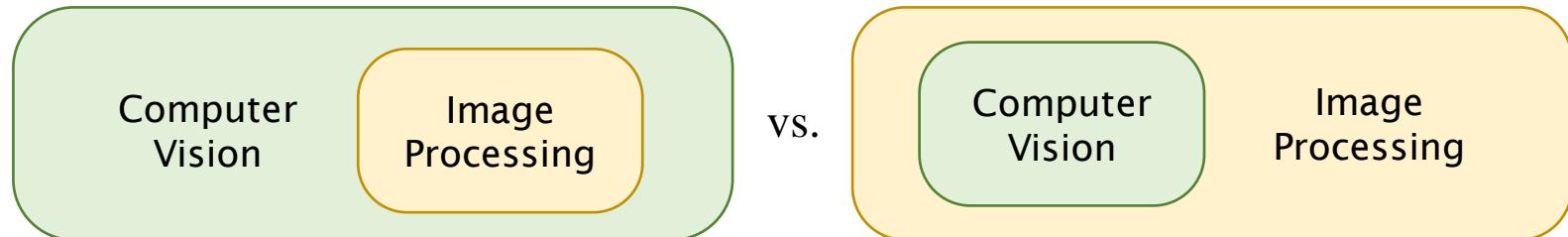
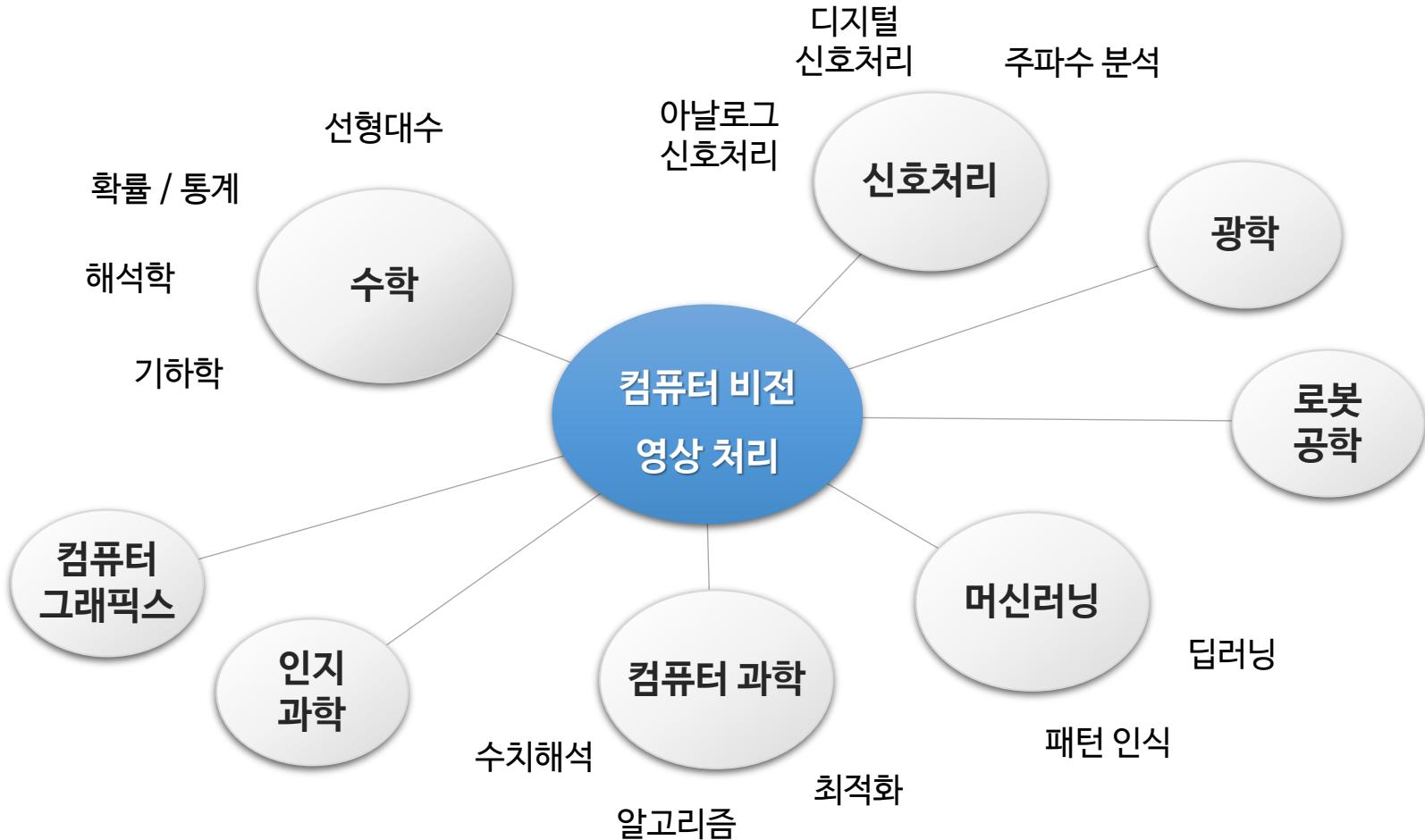


Image processing is processing of images using mathematical operations by using any form of signal processing for which the input is an image, a series of images, or a video, such as a photograph or video frame; the output of image processing may be either an image or a set of characteristics or parameters related to the image.

Computer vision, on the other hand, is often considered *high-level image processing* out of which a machine/computer/software intends to decipher the physical contents of an image or a sequence of images.

https://en.wikipedia.org/wiki/Image_processing

컴퓨터 비전 관련 분야



컴퓨터 비전 연구 분야

□ 영상의 화질 개선



Filtering App



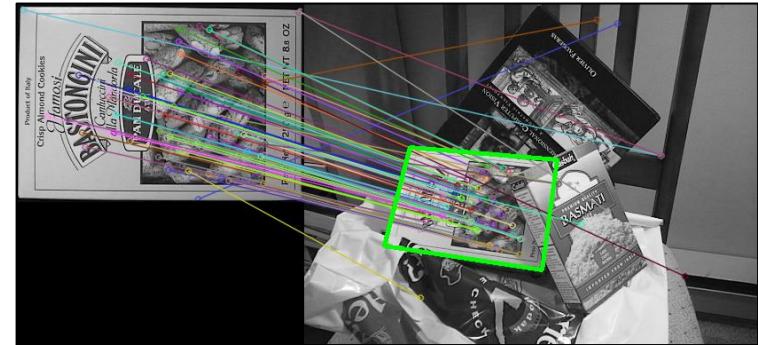
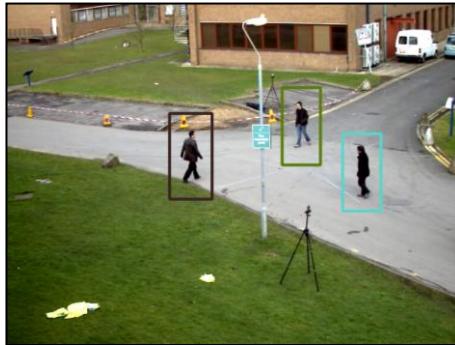
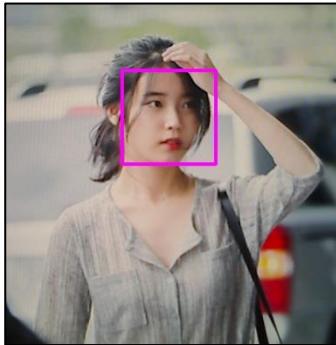
HDR



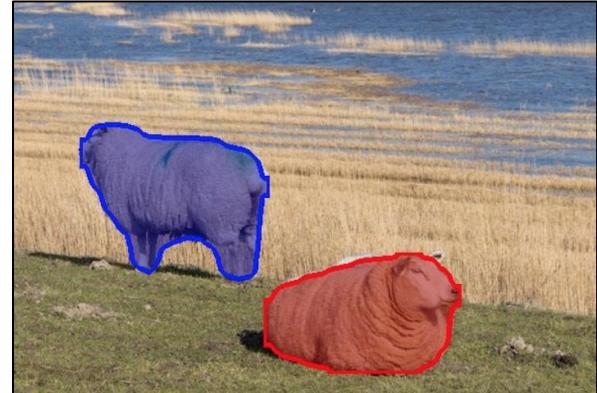
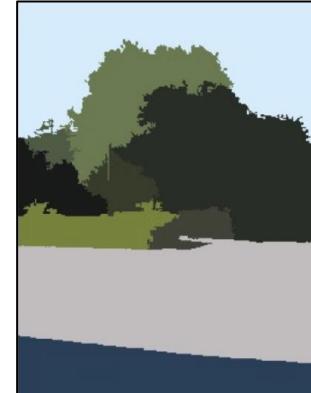
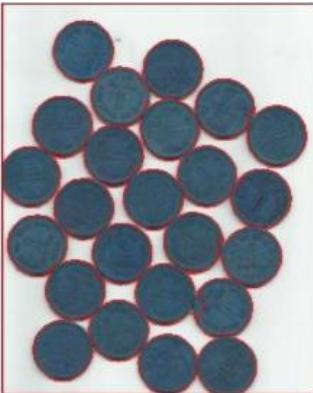
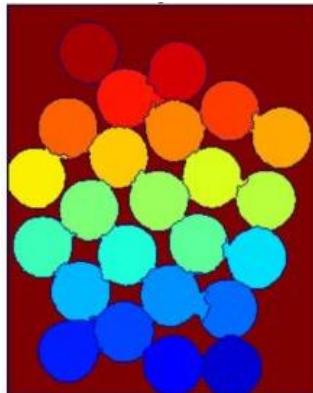
Super Resolution

컴퓨터 비전 연구 분야

□ 객체 검출(Object detection)

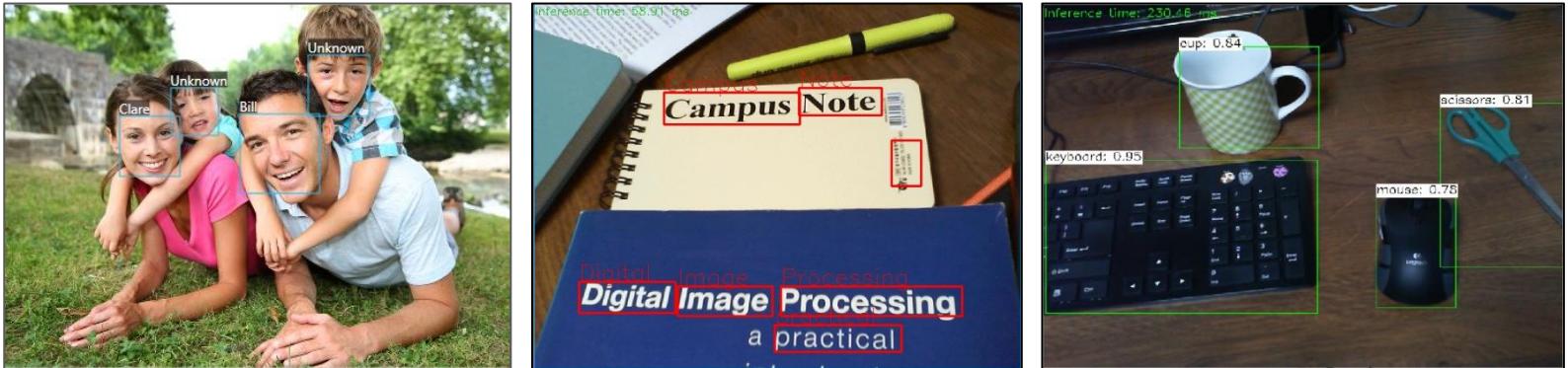


□ 분할(Segmentation)



컴퓨터 비전 연구 분야

□ 인식(Recognition)



□ 움직임 정보 추출 및 추적



컴퓨터 비전 응용 분야

□ 머신 비전(machine vision)

- 공장 자동화: 제품의 불량 검사, 위치 확인, 측정 등
- 높은 정확도와 빠른 처리 시간 요구
- 조명, 렌즈, 필터, 실시간 (Real-time) 처리

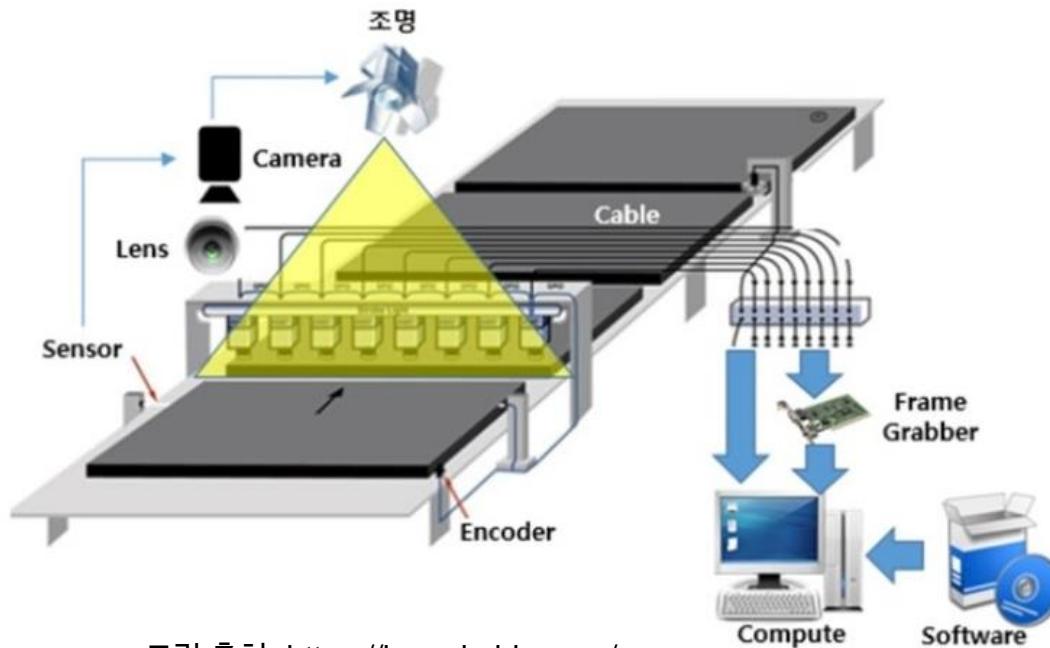


그림 출처: <https://laonple.blog.me/>

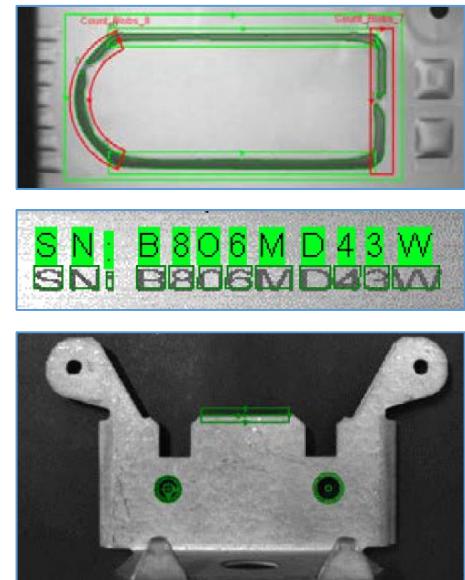


그림 출처: <http://www.cognex.com>

컴퓨터 비전 응용 분야

□ 인공지능(AI) 서비스

- 인공지능 로봇과 자율 주행 자동차
- 입력 영상을 객체와 배경으로 분할 → 객체와 배경 인식 → 상황 인식
→ 로봇과 자동차의 행동 지시
- Computer Vision + Sensor Fusion + Deep Learning
- Amazon Go / 구글, 테슬라의 자율 주행 자동차



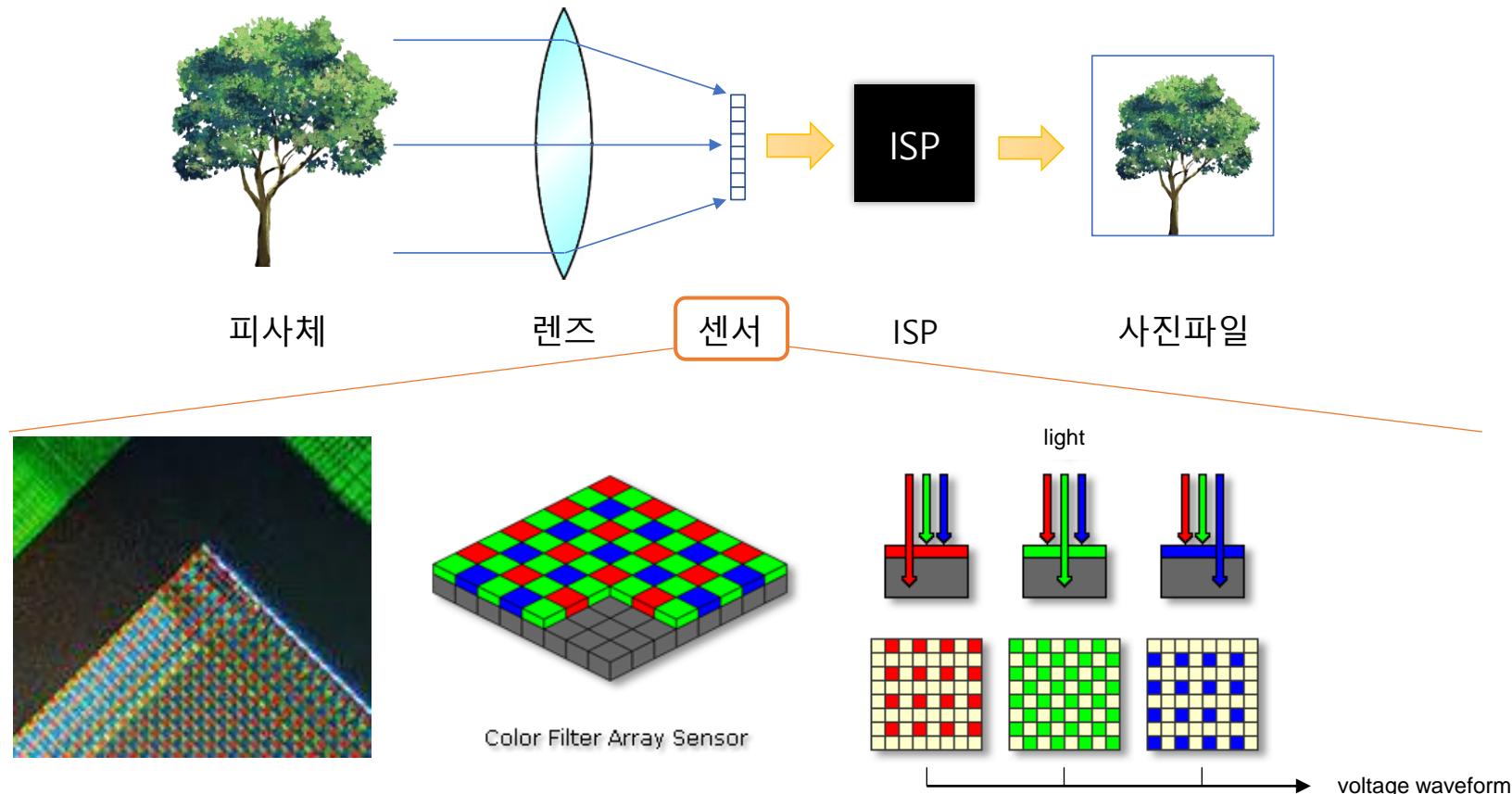
<https://www.youtube.com/watch?v=NrmMk1Myrxc>



<https://www.youtube.com/watch?v=wuhbqcMzOaw>

영상의 획득

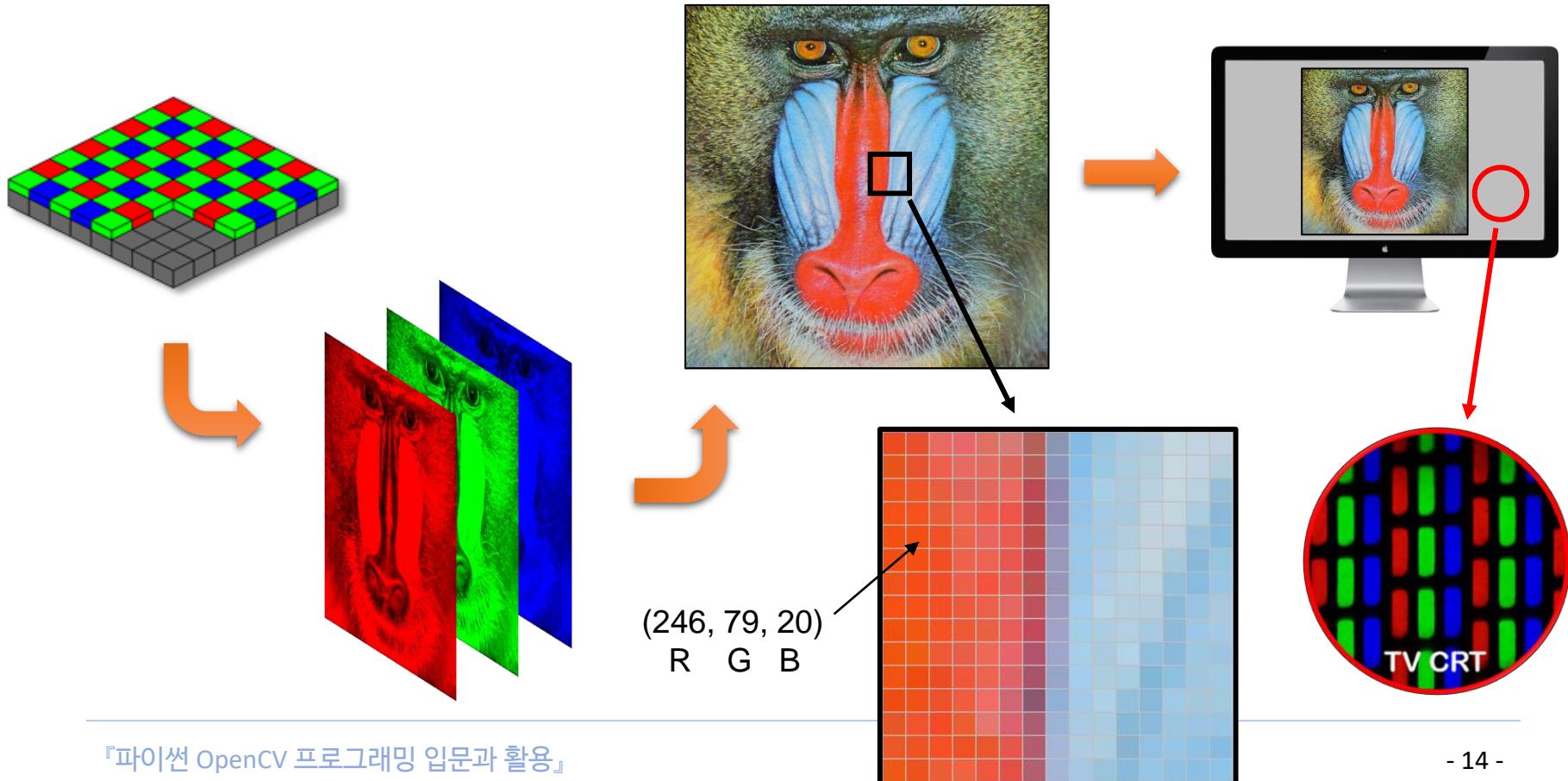
□ 디지털 카메라에서 영상의 획득 과정



영상의 표현 방법

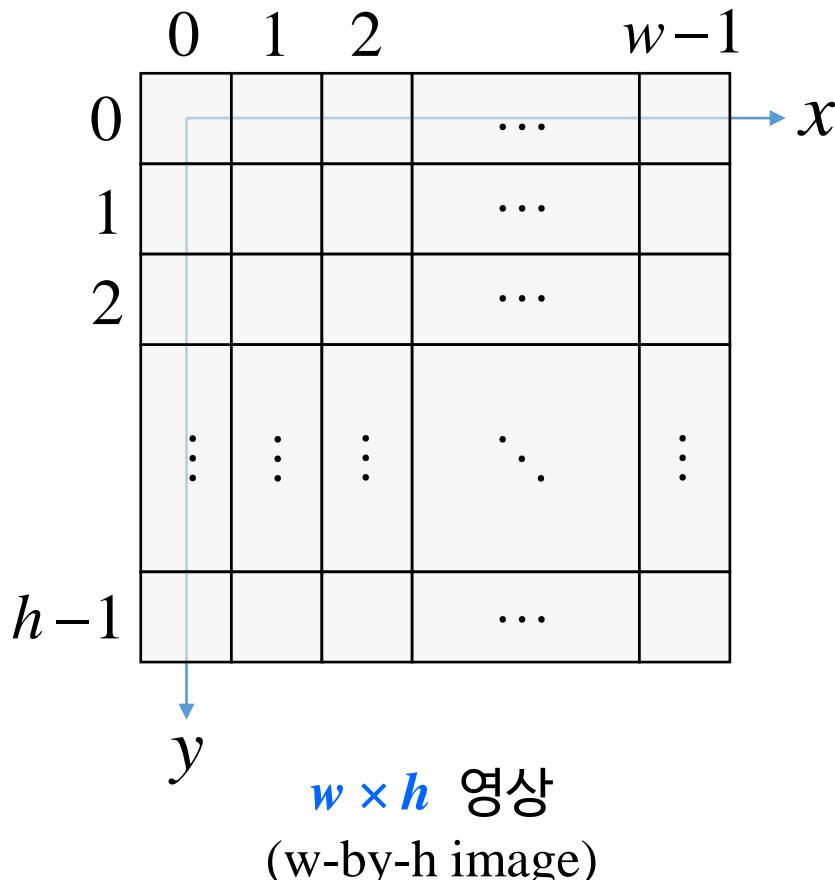
□ 영상(image)이란?

- 픽셀(pixel)이 바둑판 모양의 격자에 나열되어 있는 형태 (2차원 행렬)



영상의 표현 방법

□ 영상에서 사용되는 좌표계



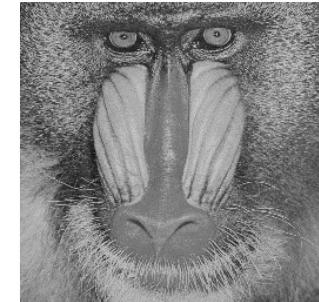
$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & a_{M,2} & \cdots & a_{M,N} \end{bmatrix}$$

$M \times N$ 행렬
(m-by-n matrix)

영상의 표현 방법

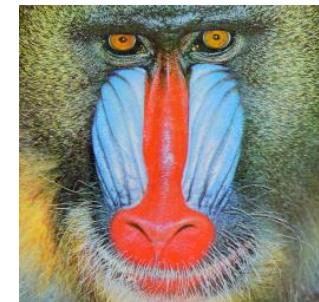
□ 그레이스케일(grayscale) 영상

- 색상 정보가 없이 오직 밝기 정보만으로 구성된 영상
- 밝기 정보를 256 단계로 표현



□ 트루컬러(truecolor) 영상

- 컬러 사진처럼 색상 정보를 가지고 있어서 다양한 색상을 표현할 수 있는 영상
- Red, Green, Blue 색 성분을 256 단계로 표현
 $\rightarrow 256^3 = 16,777,216$ 색상 표현 가능



□ 픽셀(pixel)

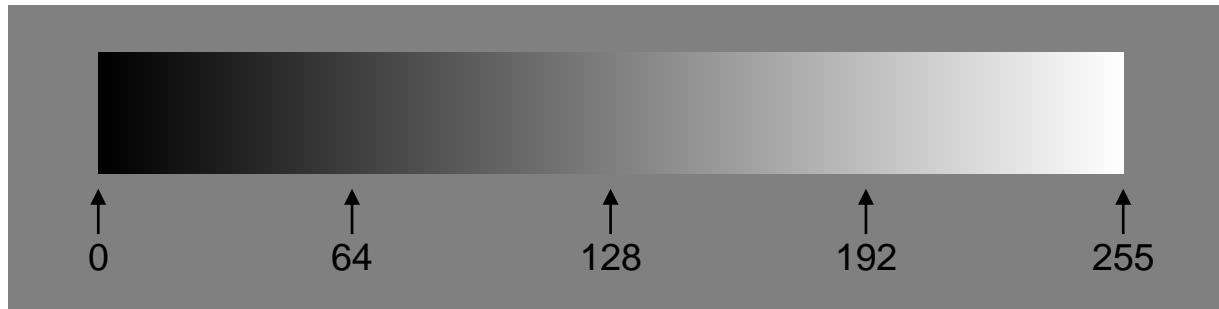
- 영상의 기본 단위, picture element, 화소(畫素)

그레이스케일 영상과 컬러 영상

□ 그레이스케일 값의 범위(Grayscale level)

- 그레이스케일 영상에서 하나의 픽셀은 0부터 255 사이의 정수 값을 가짐.

- 0 : 가장 어두운 밝기(검정색)
 - 255 : 가장 밝은 밝기(흰색)



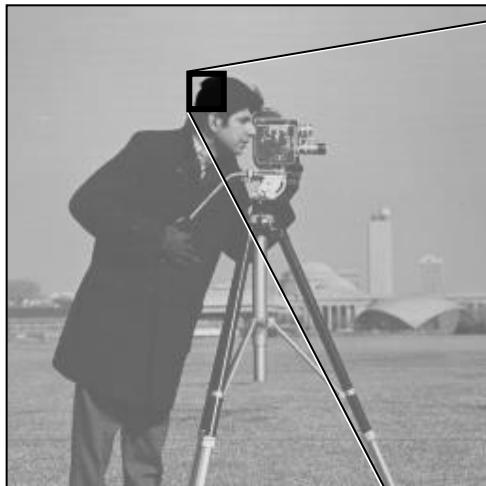
- C/C++에서 unsigned char로 표현 (1 byte)

```
typedef unsigned char BYTE;           // Windows
typedef unsigned char uint8_t;        // Linux
typedef unsigned char uchar;         // OpenCV
```

- Python에서는? → numpy.uint8

그레이스케일 영상과 컬러 영상

- 그레이스케일 영상에서 픽셀 값 분포의 예

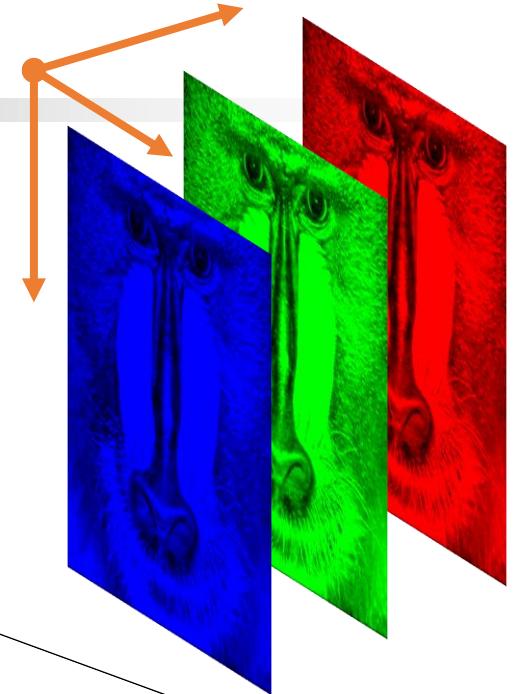
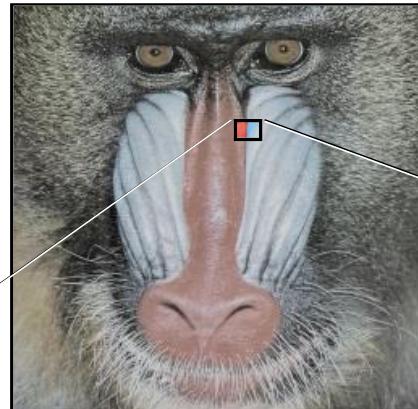


187	187	187	194	197	173	77	25	19	19
190	187	190	191	158	37	15	14	20	20
187	182	180	127	32	16	13	16	14	12
184	186	172	100	20	13	15	18	13	18
186	190	187	127	18	14	15	14	12	10
189	192	192	148	16	15	11	10	10	9
192	195	181	37	13	10	10	10	10	10
189	194	54	14	11	10	10	10	9	8
189	194	19	16	11	11	10	10	9	9
192	88	12	11	11	10	10	10	9	9

픽셀

그레이스케일 영상과 컬러 영상

- 트루컬러 영상에서 픽셀 값 분포의 예



231	103	92	228	118	86	220	122	122	213	92	59	166	136	164	131	185	228	142	192	227	145	195	228
229	104	92	221	114	72	226	109	109	208	101	51	150	146	189	137	189	232	132	189	229	143	195	229
228	109	103	229	110	90	228	107	107	206	86	64	144	165	206	138	187	230	137	188	227	142	195	230
229	110	104	231	108	102	226	117	102	206	81	45	134	155	199	135	187	230	139	193	231	156	199	229
224	106	93	219	132	114	208	118	137	189	88	70	133	166	211	127	184	229	145	192	229	165	201	228
231	111	98	227	102	70	209	110	103	178	84	65	121	157	210	131	183	229	145	192	228	161	199	228

영상의 표현 방법

□ 영상 데이터 크기 분석

- 그레이스케일 영상: (가로 크기) × (세로 크기) Bytes
- 트루컬러 영상: (가로 크기) × (세로 크기) × 3 Bytes



$$512 \times 512 = 262144 \text{ Bytes}$$



$$\begin{aligned}1920 \times 1080 \times 3 &= 6220800 \text{ Bytes} \\&\approx 6 \text{ MBytes}\end{aligned}$$

영상 파일 형식 특징

BMP

- 픽셀 데이터를 압축하지 않고 그대로 저장 → 파일 용량이 큰 편
- 파일 구조가 단순해서 별도의 라이브러리 도움 없이 직접 파일 입출력 프로그래밍 가능

JPG

- 주로 사진과 같은 컬러 영상을 저장하기 위해 사용
- 손실 압축(lossy compression)
- 압축률이 좋아서 파일 용량이 크게 감소 → 디지털 카메라 사진 포맷으로 주로 사용됨

GIF

- 256 색상 이하의 영상을 저장
→ 일반 사진을 저장 시 화질 열화가 심함
- 무손실 압축(lossless compression)
- 움직이는 GIF 지원

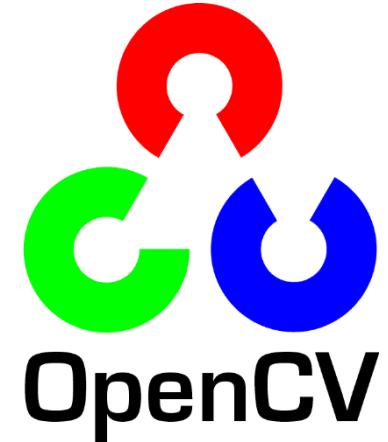
PNG

- Portable Network Graphics
- 무손실 압축 (트루컬러 영상도 무손실 압축)
- 알파 채널(투명도)을 지원

OpenCV 개요

□ What is OpenCV?

- Open source
- Computer vision & machine learning
- Software library



<http://www.opencv.org/>

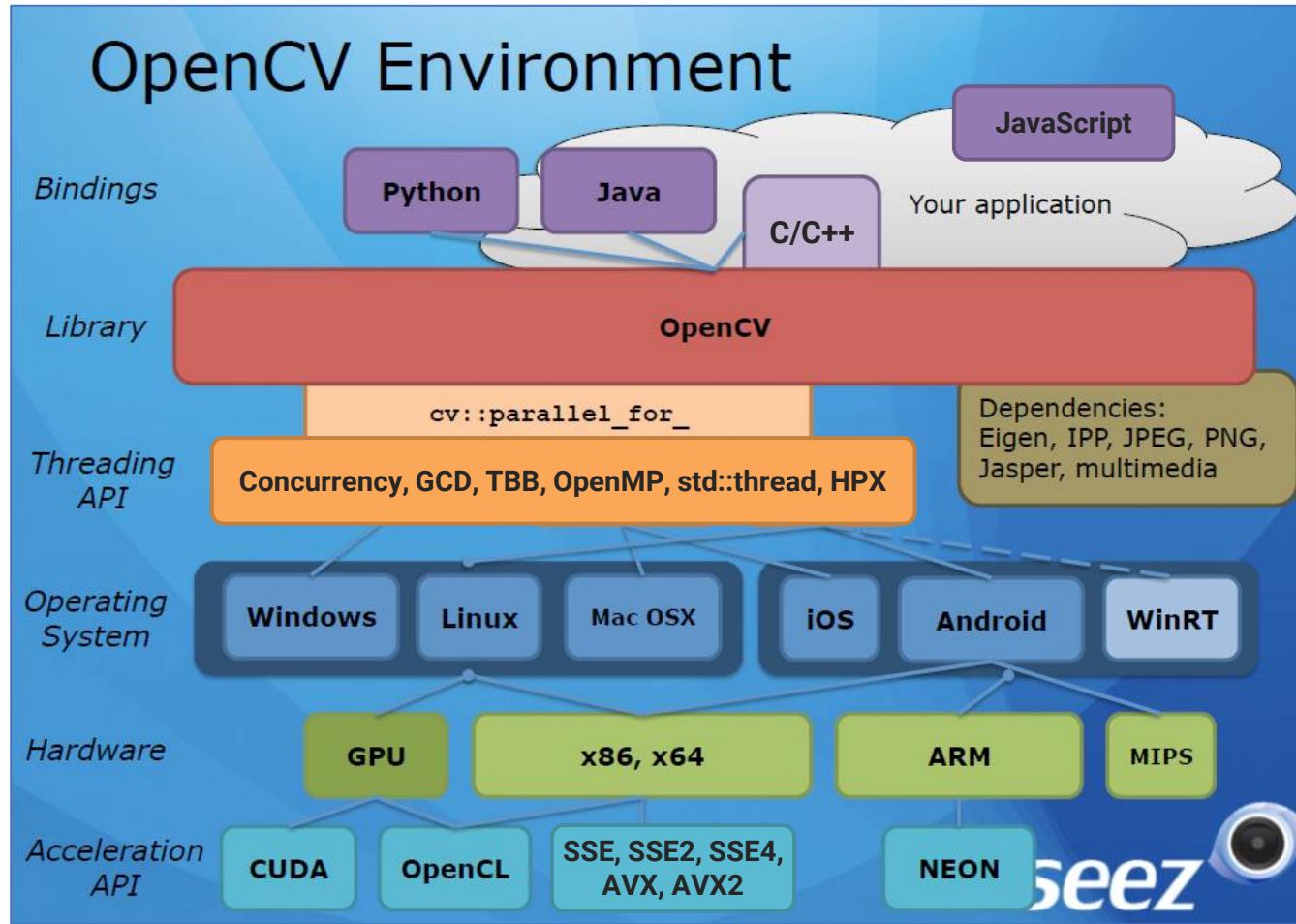
□ Why OpenCV?

- BSD license ... **Free for academic & commercial use**
- Multiple interface ... C, C++, Python, Java, JavaScript, MATLAB, etc.
- Multiple platform ... Windows, Linux, Mac OS, iOS, Android
- Optimized ... **CPU instructions, Multi-core processing, OpenCL, CUDA**
- Popular ... More than 15 million downloads
- Usage ... Stitching streetview images, detecting intrusions, monitoring mine equipment, helping robots navigate and pick up objects, Interactive art, etc.

OpenCV 역사



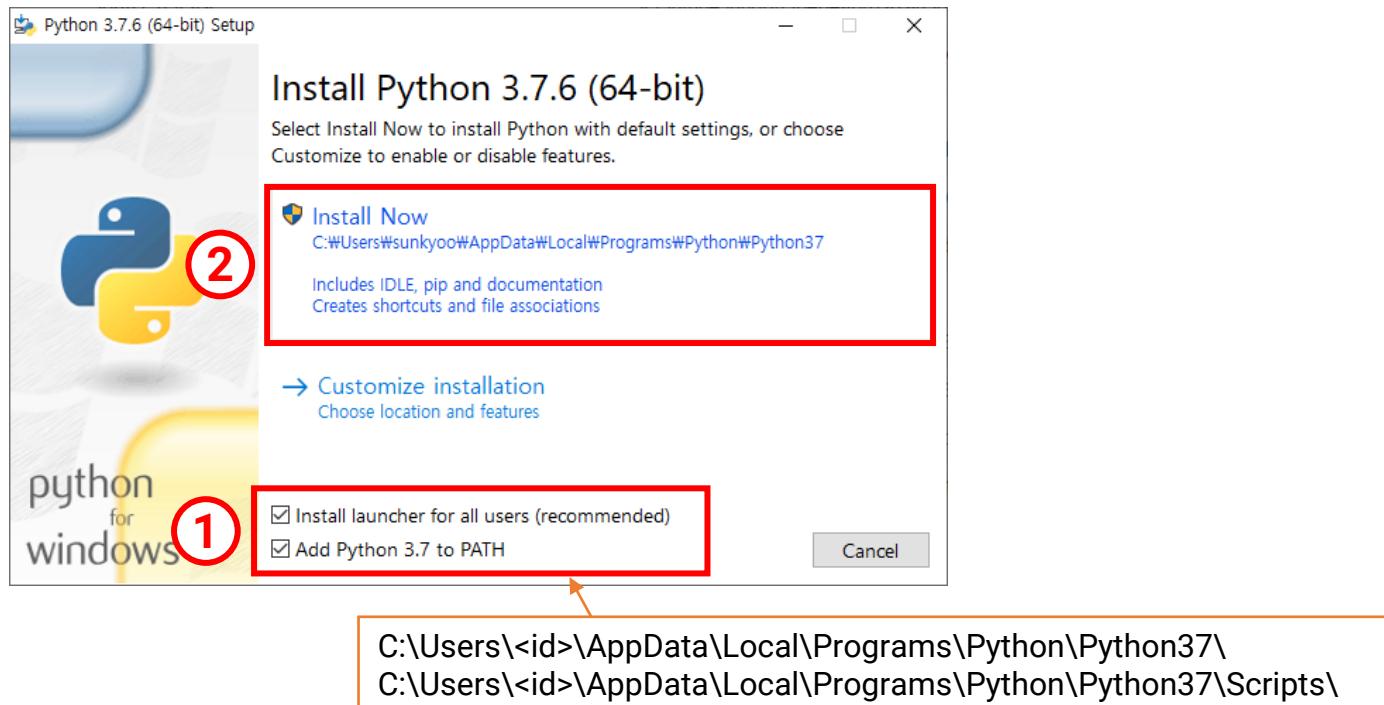
OpenCV 구성



OpenCV-Python 설치

□ Python 설치하기

- Python 설치 프로그램 다운로드: <https://www.python.org/>
 - Python 3.7.x 64-bit 권장
 - [python-3.7.6-amd64.exe](#) 파일 다운로드 & 실행



OpenCV-Python 설치

□ pip 명령으로 OpenCV-Python 설치하기

■ 설치 명령

```
> pip install opencv-python
```

- <PYTHON_PATH>\Lib\site-packages 아래에 cv, opencv_python-4.2.0.34.dist-info 폴더가 생성되고, 그 아래에 pyd 파일이 저장됨.
(cv2.cp37-win_amd64.pyd)
- 설치되는 OpenCV 버전은 <https://pypi.org/>에서 확인 가능
- numpy 패키지도 함께 설치됨
- OpenCV 추가 모듈도 함께 사용하려면 opencv-contrib-python 패키지를 설치

- OpenCV 4.2.0 버전에서 cv2.imread() 함수를 사용하여 컬러 영상을 그레이스케일 형식으로 불러올 때 픽셀 값이 잘못 설정되는 문제가 있음
- cv2.imread() 함수가 정상 동작하는 OpenCV를 설치하려면 아래 명령 사용
> pip install opencv-python==4.1.0.25

OpenCV-Python 설치

- OpenCV 설치 배포 프로그램을 이용하여 설치하기 (Optional)
 - <https://opencv.org/releases/>에서 OpenCV 설치 프로그램 다운로드 (e.g. opencv-4.3.0-vc14_vc15.exe)
 - 다운로드 받은 EXE 파일을 실행하여 압축 해제
 - "Extract to" 입력창에 C:\ 폴더 지정
→ C:\opencv 폴더가 새로 생성되면서 그 아래에 OpenCV 관련 파일이 저장됨
 - C:\opencv\build\python\cv2\python-3.7\cv2.cp37-win_amd64.pyd 파일을 <PYTHON_PATH>\Lib\site-packages 폴더로 복사
 - C:\opencv\build\x64\vc15\bin 폴더를 시스템 환경 변수 PATH에 추가

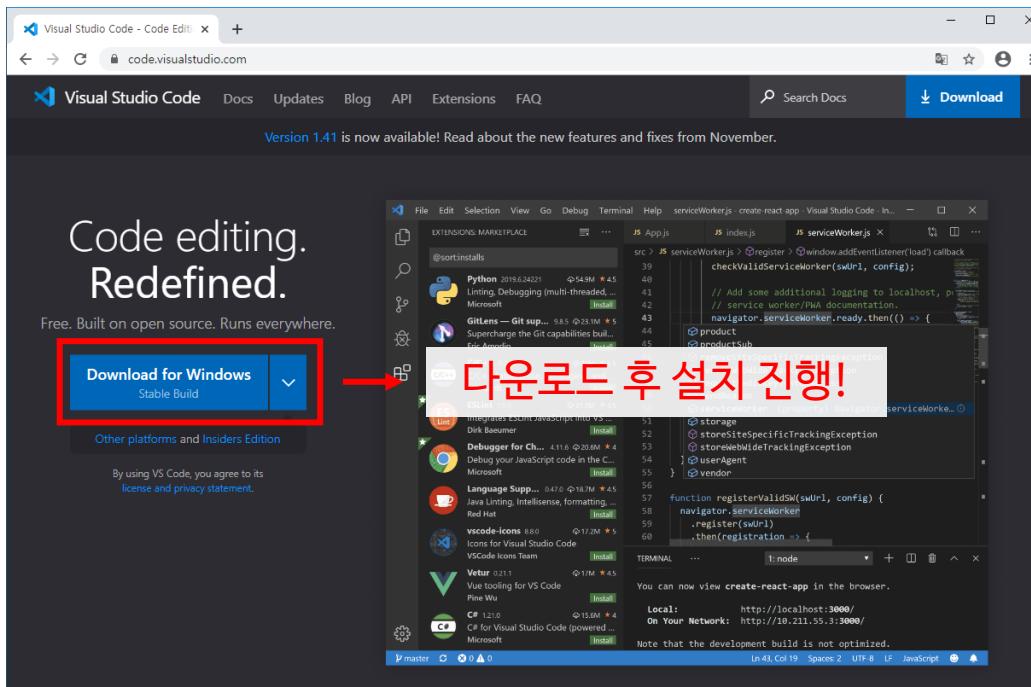
OpenCV-Python 코딩 작업 환경

- 메모장 + 명령프롬프트
- 주피터 노트북(Jupyter Notebook)
 - 웹 브라우저에서 파이썬 코드를 작성 & 실행
 - 블록 단위 코딩
 - 마크업 언어와 그림 등을 활용한 설명 추가가 쉬움
 - matplotlib 패키지를 이용하여 영상을 웹 브라우저에 출력 가능
- 파이썬 IDE
 - Visual Studio Code, PyCharm, Spider 등
 - OpenCV에서 제공하는 GUI 기능 사용 가능
 - 편리한 **디버깅**

VS Code 설치

□ Visual Studio Code

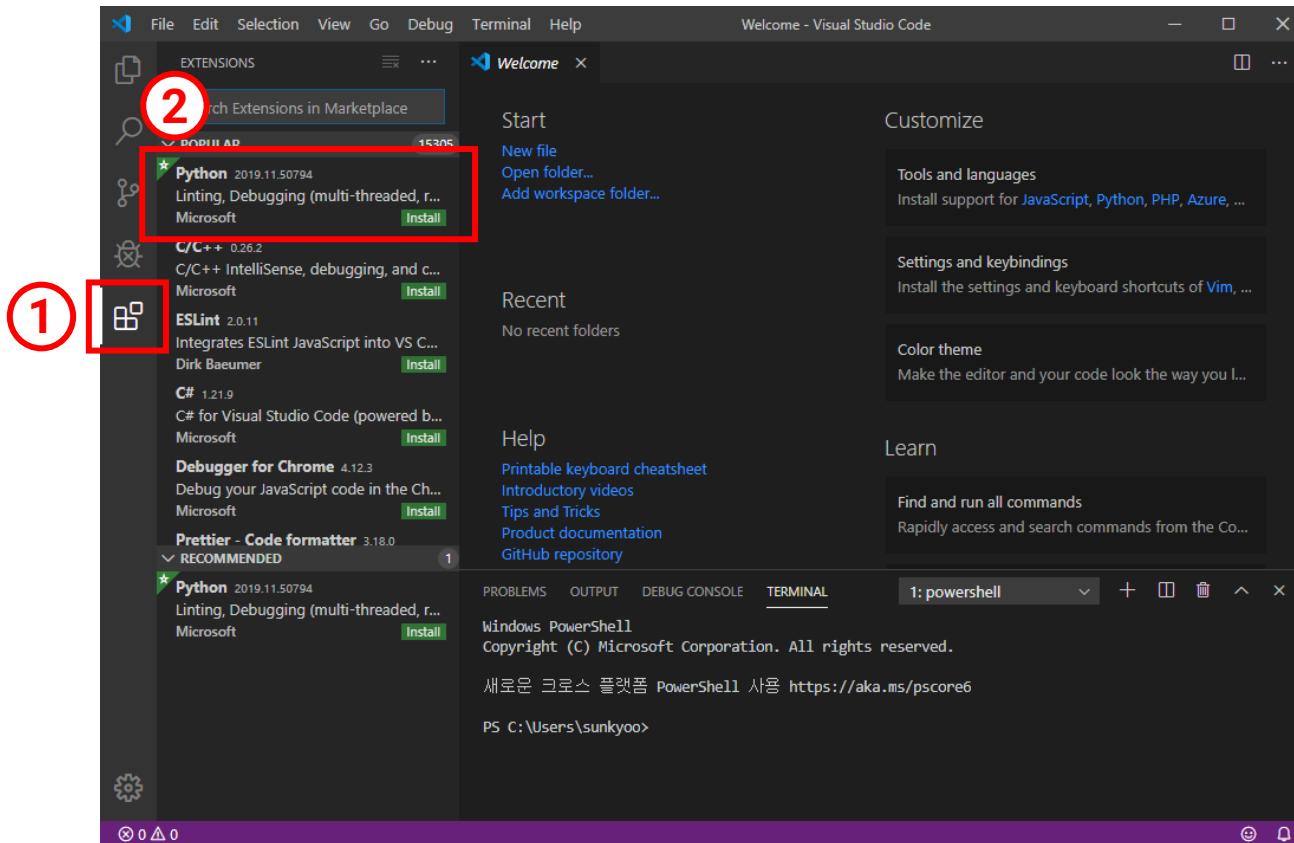
- Microsoft에서 개발한 범용 소스 코드 편집기 (Windows, MacOS, Linux)
- 설치 프로그램 다운로드: <https://code.visualstudio.com/>
- 튜토리얼: <https://code.visualstudio.com/docs>



VS Code 설치

□ Python 확장 설치

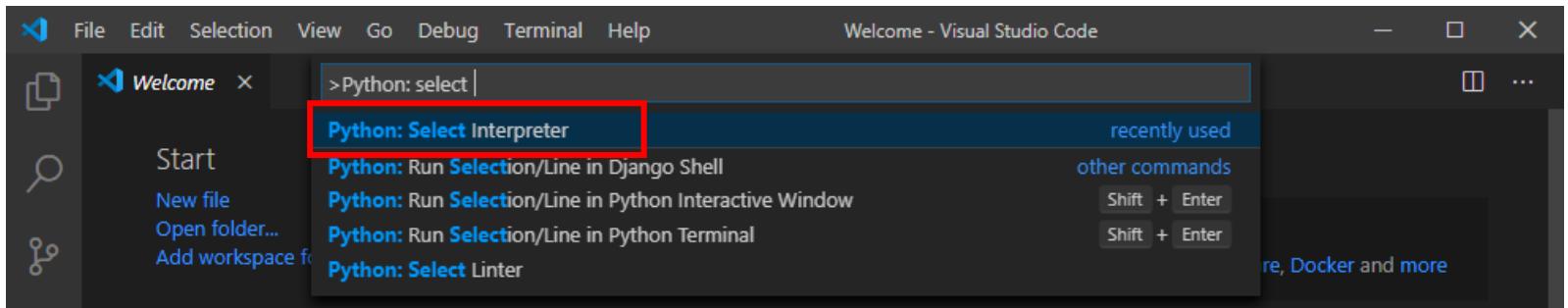
■ Extensions 뷰에서 "Python (by Microsoft)" 설치



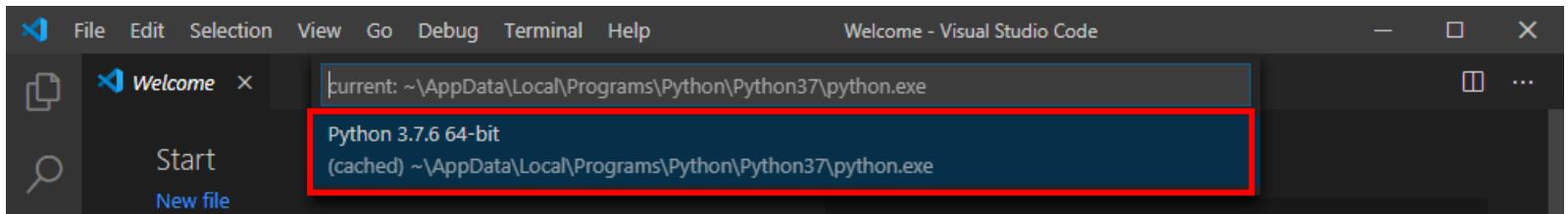
VS Code 설치

□ Python 인터프리터 선택

- [View] → [Command Palette...] (Ctrl + Shift + P) 선택 후, "Python: Select Interpreter" 입력 & 선택



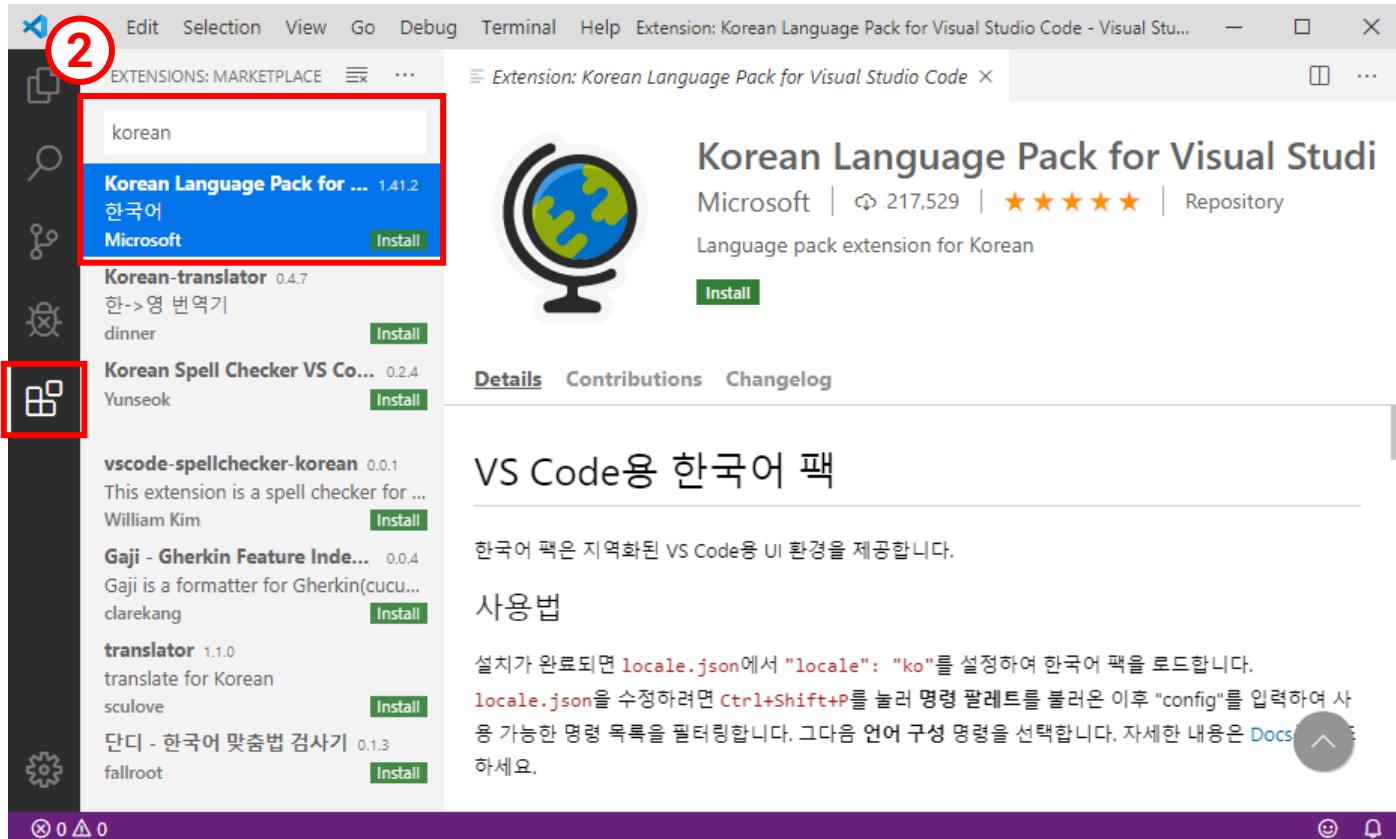
- 설치된 Python 프로그램을 선택



VS Code 설치

□ 한글 언어팩 설치 (Optional)

- Extensions 뷰에서 "Korean Language Pack" 검색하여 설치



HelloCV.py 프로그램 만들기

- VS Code에서 새 Python 프로그램 만들기
 - VS Code에서 [파일] → [폴더 열기] 메뉴 선택한 후, 임의의 폴더 선택
 - (e.g.) C:\coding\python\opencv\t-academy
 - 아래 그림처럼 [새 파일] 버튼 클릭 후, HelloCV.py 파일 이름 입력



- 편집창에 소스 코드 입력

```
import cv2

print('Hello OpenCV', cv2.__version__)
```

HelloCV.py 프로그램 만들기

- VS Code에서 새 Python 프로그램 실행하기
 - 편집창 우측 상단 "Run Python File in Terminal" 삼각형 버튼 클릭!

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a folder structure with '열려 있는 편집기' containing 'hellocv.py'. A red box highlights the 'hellocv.py > ...' item.
- Code Editor:** Displays the Python code:

```
1 import cv2
2
3
4 print('Hello OpenCV', cv2.__version__)
5
```
- Terminal:** Shows the command-line output:

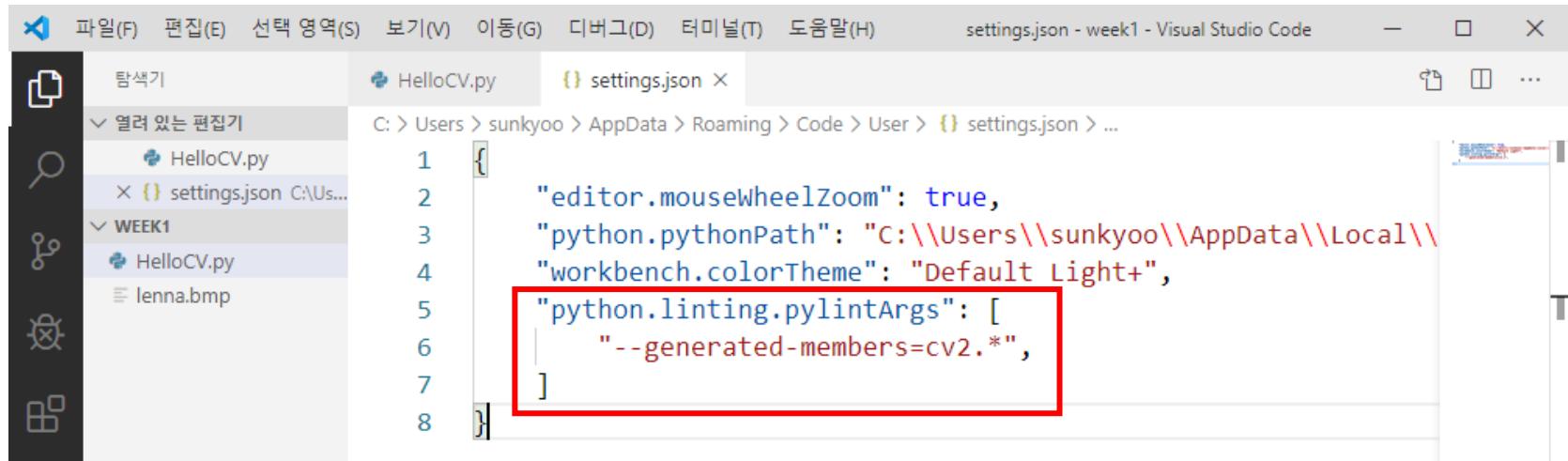
```
PS C:\coding\python\opencv\t-academy\1교시> & C:/Users/sunkyoo/AppData/Local/Programs/Python/Python37/python.exe c:/coding/python/opencv/t-academy/1교시/hellocv.py
Hello OpenCV 4.1.0
PS C:\coding\python\opencv\t-academy\1교시>
```

A red box highlights the terminal window.
- Status Bar:** Shows 'Python 3.7.6 64-bit' and other status indicators.

HelloCV.py 프로그램 만들기

□ VS Code 편집창에서 cv2 에러 없애기

- VS Code pylint에서 cv2 모듈의 멤버를 제대로 인식하지 못하는 문제가 있음 (실행은 정상적으로 동작함)
- [보기] → [명령 팔레트] (Ctrl + Shift + P)
- "Preferences: Open Settings (JSON)" 선택
- "python.linting.pylintArgs": ["--generated-members=cv2.*"] 문장 추가



The screenshot shows the Visual Studio Code interface with the title bar "settings.json - week1 - Visual Studio Code". The left sidebar shows a file tree with "HelloCV.py" and "settings.json" selected. The main editor area displays the contents of "settings.json". A red box highlights the line of code: "python.linting.pylintArgs": ["--generated-members=cv2.*"]. The rest of the JSON configuration includes:

```
1 {  
2     "editor.mouseWheelZoom": true,  
3     "python.pythonPath": "C:\\\\Users\\\\sunkyoo\\\\AppData\\\\Local\\\\Programs\\\\Python\\\\Python37\\\\python.exe",  
4     "workbench.colorTheme": "Default Light+",  
5     "python.linting.pylintArgs": [  
6         "--generated-members=cv2.*",  
7     ]  
8 }
```



break



파이썬 OpenCV 프로그래밍 입문과 활용

2. OpenCV 프로그래밍 기초

『OpenCV 4로 배우는 컴퓨터 비전과 머신 러닝』 저자

황 선 규 / 공학박사

목차

- OpenCV 프로그래밍 기초
 - 영상 불러와서 출력하기
 - 영상 데이터 다루기 (생성, 복사, 크롭, 합성)
 - 컬러 영상 처리: 색 공간 변환
 - 기하학적 변환 (크기 변환, 회전 변환)
 - OpenCV 그리기 함수

- 카메라 & 동영상 처리
 - 카메라 & 동영상 프레임 처리
 - 동영상 파일 저장하기

영상 불러와서 출력하기

□ BMP 파일을 불러와서 출력하는 소스 코드 추가 입력

```
import sys
import cv2
```

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

cat.bmp 파일을 불러와 img 변수에 저장.
type(img): <class 'numpy.ndarray'>

```
if img is None:
    print('Image load failed!')
    sys.exit()
```

영상 파일 불러오기가 실패하면 img가 None
이고, 이 경우 메시지를 출력하고 종료.

```
cv2.namedWindow('image')
cv2.imshow('image', img)
cv2.waitKey()
```

image라는 이름의 새 창을 만들고,
이 창에 img 영상을 출력.
그리고 키보드 입력이 있을 때까지 대기.

```
cv2.destroyAllWindows()
```

생성된 모든 창을 닫음

영상 불러와서 출력하기

- VS Code에서 img_show.py 프로그램 실행하기
 - cat.bmp 파일을 해당 폴더로 복사한 후, 프로그램 실행

The screenshot shows a Visual Studio Code window with the following details:

- File Explorer:** Shows a file tree with several Python files and a video file.
- Code Editor:** Displays the `img_show.py` script:

```
1 import sys
2 import cv2
3
4 # 영상 불러오기
5 img = cv2.imread('cat.bmp')
6
7 if img is None:
8     print('Image load failed!')
9     sys.exit()
10
11 # 영상 화면 출력
12 cv2.namedWindow('image')
13 cv2.imshow('image', img)
14 cv2.waitKey()
15
16 # 창 닫기
17 cv2.destroyAllWindows()
```
- Terminal:** Shows a Windows PowerShell window with the command `PS C:\coding\python\opencv\t-academy\2교시> & C:/Users/sunkkyoo/AppData/Local/Programs/Python/Python37/python.exe c:/coding/python/opencv/t-academy/2교시/img_show.py`.
- Output:** Shows the output of the program, which is a window titled "image" displaying a close-up of a brown tabby cat's face.

OpenCV API

□ 영상 파일 불러오기

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

- **filename:** 불러올 영상 파일 이름 (문자열)
 - 상대 경로: 'cat.bmp', '../data/cat.bmp'
 - 절대 경로: 'c:\cat.bmp', '/home/id/cat.bmp'
- **flags:** 영상 파일 불러오기 옵션 플래그

cv2.IMREAD_COLOR	BGR 컬러 영상으로 읽기 (기본값) shape = (rows, cols, 3), dtype= uint8
cv2.IMREAD_GRAYSCALE	그레이스케일 영상으로 읽기 shape = (rows, cols), dtype= uint8
cv2.IMREAD_UNCHANGED	영상 파일 속성 그대로 읽기 (e.g.) 투명한 PNG 파일: shape = (rows, cols, 4)

- **retval:** 불러온 영상 데이터 (**numpy.ndarray**)

OpenCV API

□ 영상 파일 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) JPG 파일 압축률을 90%로 지정하고 싶다면
[cv2.IMWRITE_JPEG_QUALITY, 90] 지정
- retval: 정상적으로 저장하면 True, 실패하면 False.

OpenCV API

□ 새 창 띄우기 & 창 닫기

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

- **winname:** 창 고유 이름. 이 이름으로 창을 구분함.
- **flags:** 창 속성 지정 플래그

cv2.WINDOW_NORMAL	영상 크기가 창 크기에 맞게 지정됨
cv2.WINDOW_AUTOSIZE	창 크기가 영상 크기에 맞게 자동으로 변경됨 (기본값)

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

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

- **winname:** 닫고자 하는 창 이름
- 일반적인 경우 프로그램 종료 시 운영 체제에 의해 열려 있는 모든 창이 자동으로 닫힘

OpenCV API

□ 창 위치 & 크기 지정

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

- **winname:** 창 이름
- **x, y:** 이동할 위치 좌표

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

- **winname:** 창 이름
- **width, height:** 변경할 창 크기
- 참고 사항
 - 창 생성 시 cv2.WINDOW_NORMAL 속성으로 생성되어야 동작함.
 - 영상 출력 부분의 크기만을 고려함 (제목 표시줄, 창 경계는 고려되지 않음)

OpenCV API

□ 영상 출력하기

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

- **winname:** 영상을 출력할 대상 창 이름
- **mat:** 출력할 영상 데이터 ([numpy.ndarray](#))
- 데이터 타입에 따른 출력 방식
 - **uint8:** 픽셀 값을 그대로 출력
 - **uint16, int16:** 픽셀 값을 255로 나눠서 출력
 - **float32, float64:** 픽셀 값에 255를 곱해서 출력
- 참고 사항
 - 만약 **winname**에 해당하는 창이 없으면 자동으로 **cv2.WINDOW_AUTOSIZE** 속성의 창을 새로 만들어서 영상을 출력함
 - Windows 운영체제에서는 Ctrl + C (복사), Ctrl + S (저장) 지원
 - 실제로는 **cv2.waitKey()** 함수를 호출해야 화면에 영상이 나타남.

OpenCV API

□ 키보드 입력 대기

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

- delay: 밀리초 단위 대기 시간. $\text{delay} \leq 0$ 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

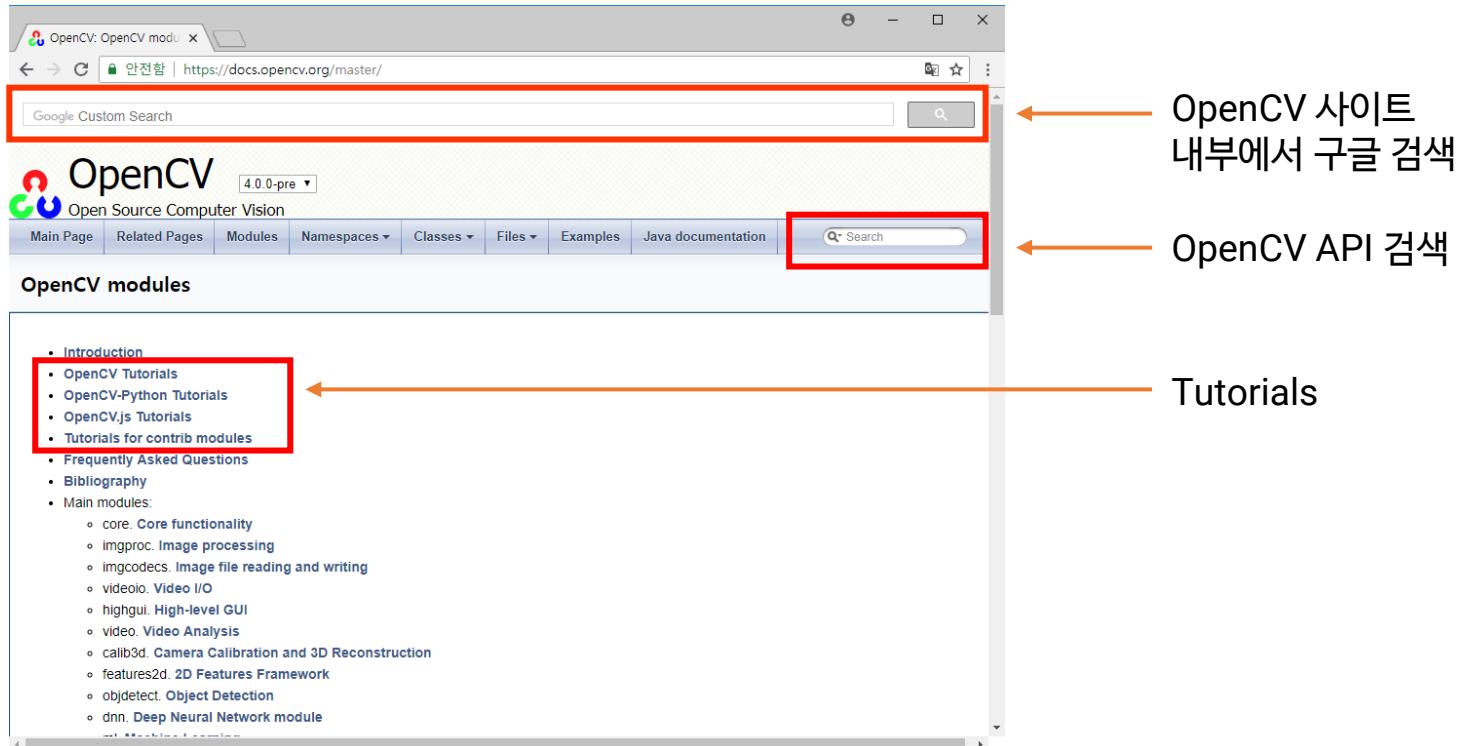
```
while True:  
    if cv2.waitKey() == ord('q'):  
        break
```

- 주요 특수키 코드: ESC → 27, ENTER → 13, TAB → 9

OpenCV API 도움말 찾기

□ OpenCV API 도움말 찾기

- OpenCV 최신 도움말: <http://docs.opencv.org/master/>
- OpenCV 도움말 웹페이지에서 우측 상단 검색창 활용



Matplotlib을 이용한 영상 출력

□ Matplotlib 패키지

- 함수 그래프, 차트(chart), 히스토그램(histogram) 등의 다양한 그리기 기능을 제공하는 Python 패키지
- 보통 `matplotlib.pyplot` 모듈을 `plt` 이름으로 불러와서 사용
- Jupyter Notebook에서 사용할 때 유용

```
> pip install matplotlib
```

□ 컬러 영상 출력

- 컬러 영상의 색상 정보가 **RGB 순서**이어야 함
- `cv2.imread()` 함수로 불러온 영상의 색상 정보는 BGR 순서이므로 이를 RGB 순서로 변경해야 함 → `cv2.cvtColor()` 함수 이용

□ 그레이스케일 영상 출력

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

Matplotlib을 이용한 영상 출력

```
import matplotlib.pyplot as plt
import cv2

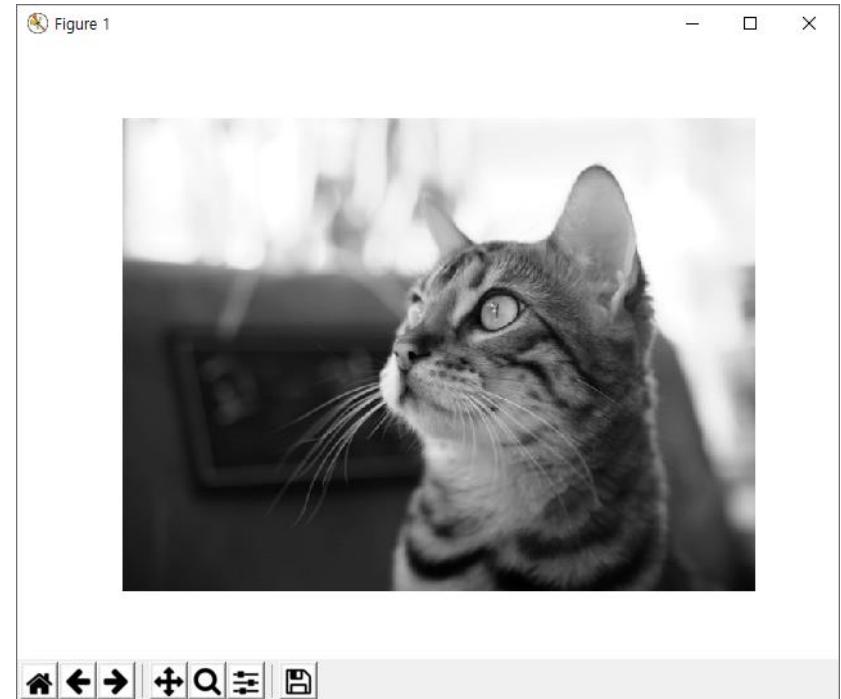
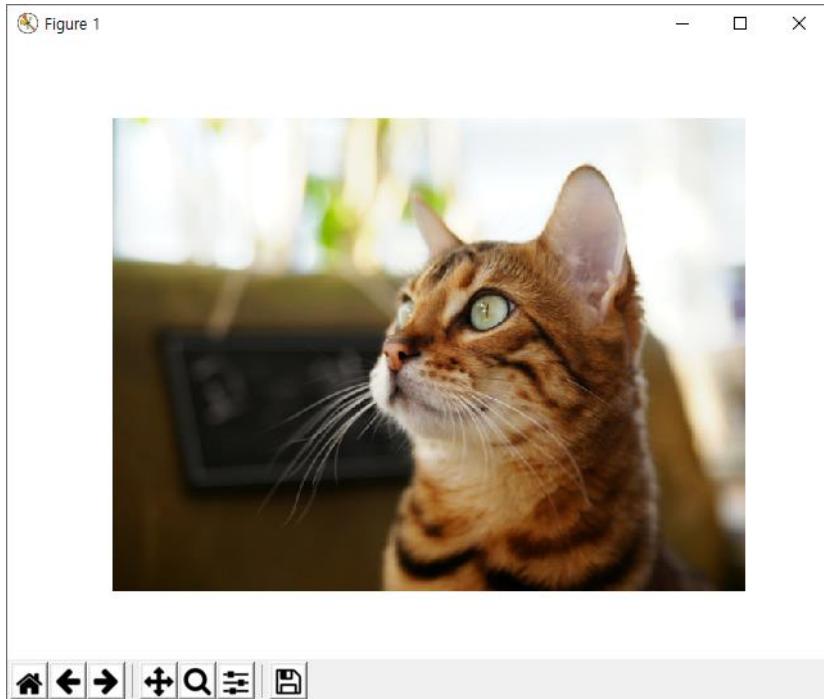
# 컬러 영상 출력
imgBGR = cv2.imread('cat.bmp')
imgRGB = cv2.cvtColor(imgBGR, cv2.COLOR_BGR2RGB)
plt.axis('off')
plt.imshow(imgRGB)
plt.show()

# 그레이스케일 영상 출력
imgGray = cv2.imread('cat.bmp', cv2.IMREAD_GRAYSCALE)
plt.axis('off')
plt.imshow(imgGray, cmap='gray')
plt.show()

# 두 개의 영상을 함께 출력
plt.subplot(121), plt.axis('off'), plt.imshow(imgRGB)
plt.subplot(122), plt.axis('off'), plt.imshow(imgGray, cmap='gray')
plt.show()
```

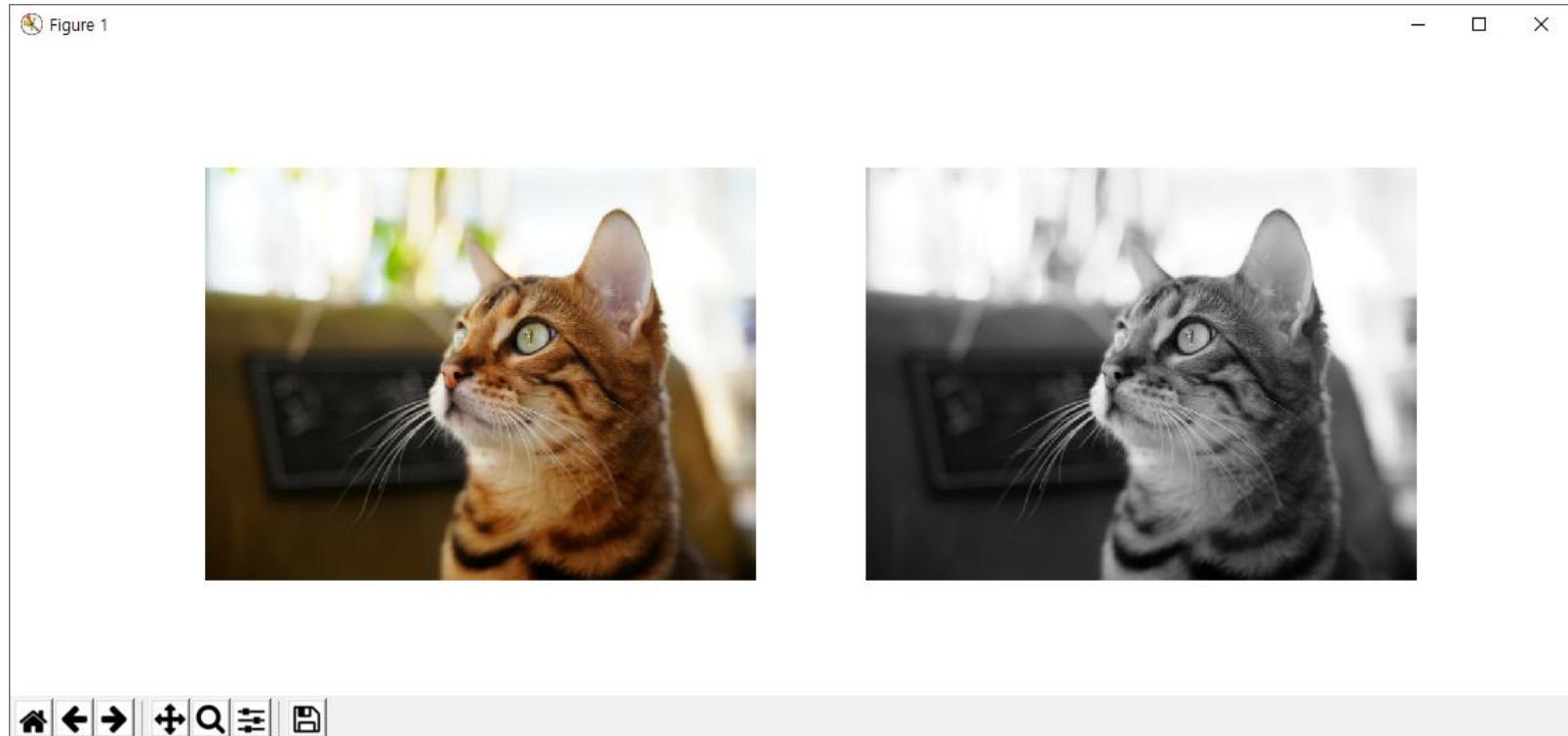
Matplotlib을 이용한 영상 출력

- Matplotlib.pyplot을 이용한 영상 출력 결과



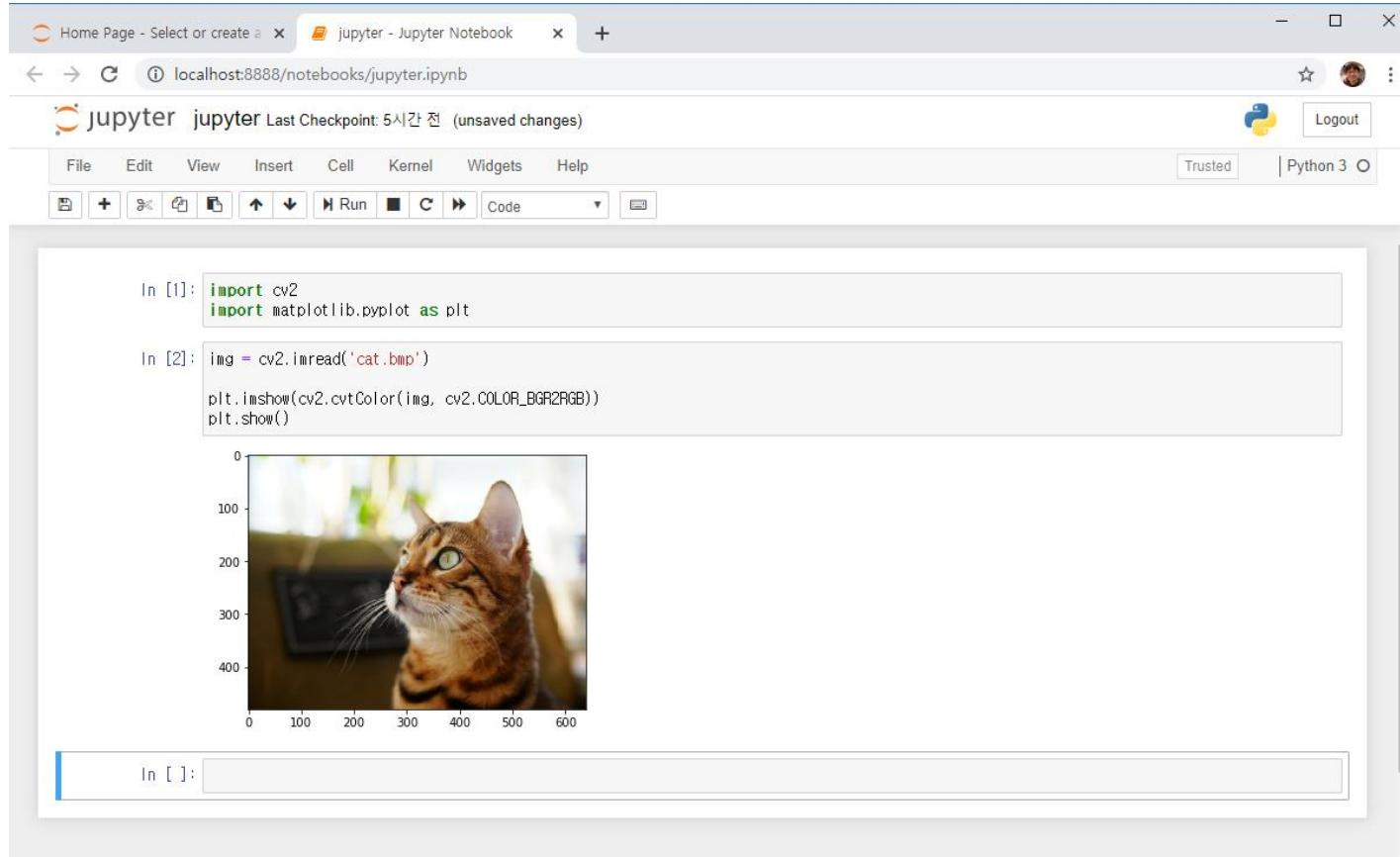
Matplotlib을 이용한 영상 출력

- Matplotlib.pyplot을 이용한 영상 출력 결과



Matplotlib을 이용한 영상 출력

□ Jupyter Notebook에서 영상 출력



The screenshot shows a Jupyter Notebook window with the following details:

- Title Bar:** Home Page - Select or create a × jupyter - Jupyter Notebook × +
- URL:** localhost:8888/notebooks/jupyter.ipynb
- User Information:** jupyter jupyter Last Checkpoint: 5시간 전 (unsaved changes) Python 3 Trusted Logout
- Toolbar:** File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Run, Cell, Code, Cell
- Code Cells:**
 - In [1]:

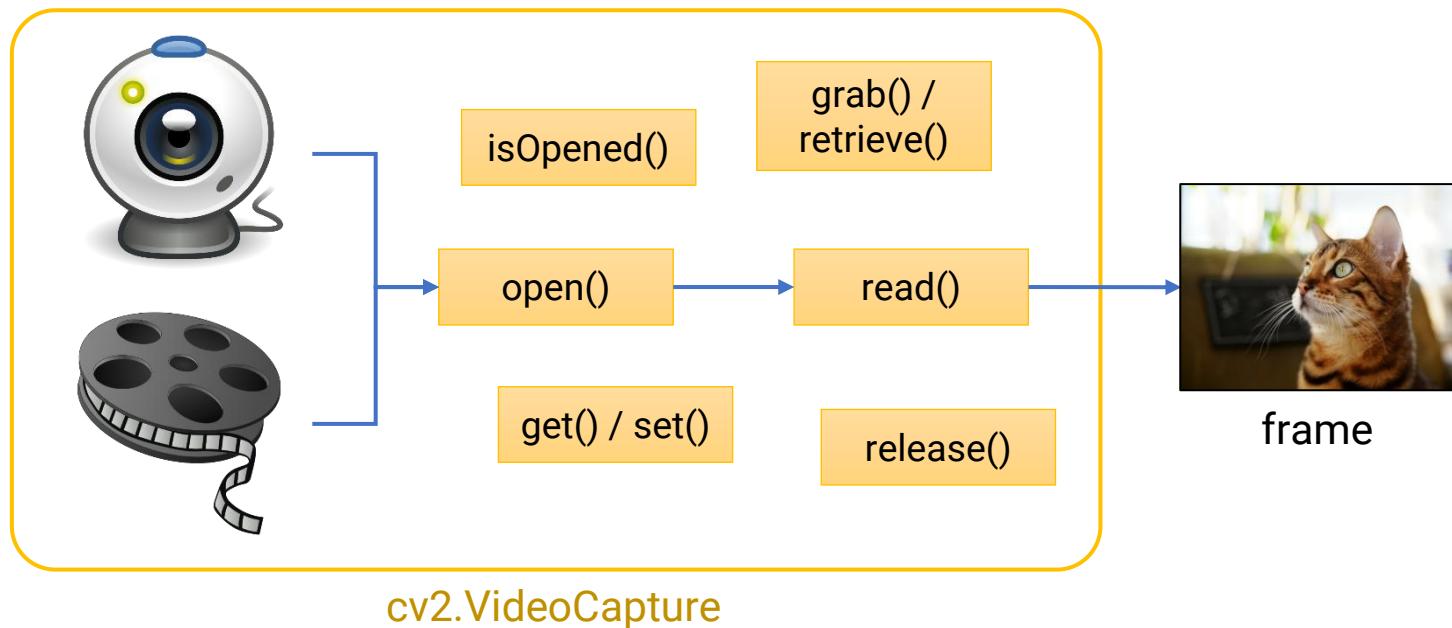
```
import cv2
import matplotlib.pyplot as plt
```
 - In [2]:

```
img = cv2.imread('cat.bmp')
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```
- Output:** An image of a Bengal cat's head, facing right, with its name and coordinates labeled.
- Input Cell:** In []:

카메라 & 동영상 처리

□ cv2.VideoCapture 클래스

- OpenCV에서는 카메라와 동영상으로부터 프레임(frame)을 받아오는 작업을 cv2.VideoCapture 클래스 하나로 처리함



OpenCV API

□ 동영상, 정지 영상 시퀀스, 비디오 스트림 열기

```
cv2.VideoCapture(filename, apiPreference=None) -> retval
```

- **filename:**
 - 비디오 파일 이름 (e.g. 'video.avi')
 - 정지 영상 시퀀스 (e.g. 'img_%02d.jpg')
 - 비디오 스트림 URL (e.g. 'protocol://host:port/script?params|auth')
- **apiPreference:** 선호하는 동영상 처리 방법을 지정
- **retval:** cv2.VideoCapture 객체

```
cv2.VideoCapture.open(filename, apiPreference=None) -> retval
```

- **retval:** 성공하면 True, 실패하면 False.

OpenCV API

□ 카메라 열기

```
cv2.VideoCapture(index, apiPreference=None) -> retval
```

- index: camera_id + domain_offset (CAP_*) id
 - camera_id == 0 이면 시스템 기본 카메라
 - domain_offset == 0 이면 auto detect.
 - 기본 카메라를 기본 방법으로 열려면 index에 0을 전달.
- apiPreference: 선호하는 카메라 처리 방법을 지정
- retval: cv2.VideoCapture 객체

```
cv2.VideoCapture.open(index, apiPreference=None) -> retval
```

- retval: 성공하면 True, 실패하면 False.

OpenCV API

□ 비디오 캡쳐가 준비되었는지 확인

```
cv2.VideoCapture.isOpened() -> retval
```

- retval: 성공하면 True, 실패하면 False.

□ 프레임 받아오기

```
cv2.VideoCapture.read(image=None) -> retval, image
```

- retval: 성공하면 True, 실패하면 False.
- image: 현재 프레임 (numpy.ndarray)

OpenCV API

□ 카메라, 비디오 장치 속성 값 참조

```
cv2.VideoCapture.get(propId) -> retval
```

■ propId: 속성 상수

cv2.CAP_PROP_FRAME_WIDTH	프레임 가로 크기
cv2.CAP_PROP_FRAME_HEIGHT	프레임 세로 크기
cv2.CAP_PROP_FPS	초당 프레임 수
cv2.CAP_PROP_FRAME_COUNT	비디오 파일의 총 프레임 수
cv2.CAP_PROP_POS_MSEC	밀리초 단위로 현재 위치
cv2.CAP_PROP_POS_FRAMES	현재 프레임 번호
cv2.CAP_PROP_EXPOSURE	노출
cv2.CAP_PROP_ZOOM	줌(확대/축소 비율)

■ retval: 성공하면 해당 속성 값, 실패하면 0.

OpenCV API

□ 카메라, 비디오 장치 속성 값 참조

```
cv2.VideoCapture.set(propId, value) -> retval
```

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

카메라 처리 예제

```

import cv2

cap = cv2.VideoCapture()
cap.open(0)                                     ← VideoCapture 객체 생성 후 0번 카메라 장치 열기.
                                                ← cap = cv2.VideoCapture(0) 과 동일

print('Frame width:', round(cap.get(cv2.CAP_PROP_FRAME_WIDTH)))
print('Frame height:', round(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))

while True:
    ret, frame = cap.read()                     ← 카메라로부터 한 프레임을 정상적으로 받아오면
                                                ← ret에는 True, frame에는 해당 프레임이 저장됨.

    edge = cv2.Canny(frame, 50, 150)

    cv2.imshow('frame', frame)                  ← 현재 프레임과 에지 검출 영상을 출력
    cv2.imshow('edge', edge)

    if cv2.waitKey(10) == 27:                   ← 일정 시간(e.g. 10ms) 기다린 후 다음 프레임 처리.
                                                ← 만약 ESC 키를 누르면 while 루프 종료.

        break

cap.release()                                 ← 사용한 자원을 해제. (생략 가능)
cv2.destroyAllWindows()

```

카메라 처리 예제

```
import cv2

cap = cv2.VideoCapture()
cap.open(0)
print('Frame width:', round(cap.get(cv2.CAP_PROP_FRAME_WIDTH)))
print('Frame height:', round(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)))

while True:
    ret, frame = cap.read()
    edge = cv2.Canny(frame, 50, 150)

    cv2.imshow('frame', frame)
    cv2.imshow('edge', edge)

    if cv2.waitKey(10) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

```
if not cap.isOpened():
    print("Camera open failed!")
    exit()
```

```
if not ret:
    break
```

동영상 처리 예제

```
import cv2

cap = cv2.VideoCapture('vtest.avi')

fps = round(cap.get(cv2.CAP_PROP_FPS))
delay = round(1000 / fps)

while True:
    ret, frame = cap.read()

    edge = cv2.Canny(frame, 50, 150)

    cv2.imshow('frame', frame)
    cv2.imshow('edge', edge)

    if cv2.waitKey(delay) == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

동영상 처리 예제



동영상 파일 저장하기

- cv2.VideoWriter 클래스
 - OpenCV에서는 cv2.VideoWriter 클래스를 이용하여 일련의 프레임을 동영상 파일로 저장할 수 있음
 - 일련의 프레임은 모두 크기와 데이터 타입이 같아야 함
- Fourcc (4-문자 코드, four character code)
 - 동영상 파일의 코덱, 압축 방식, 색상 혹은 픽셀 포맷 등을 정의하는 정수 값
 - <http://www.fourcc.org/codecs.php>

cv2.VideoWriter_fourcc(*'DIVX')	DIVX 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

OpenCV API

□ 저장을 위한 동영상 파일 열기

```
cv2.VideoWriter(filename, fourcc, fps, frameSize, isColor=None)  
-> retval
```

- filename: 비디오 파일 이름 (e.g. 'video.mp4')
- fourcc: fourcc (e.g. cv2.VideoWriter_fourcc(*'DIVX'))
- fps: 초당 프레임 수 (e.g. 30)
- frameSize: 프레임 크기 (e.g. [640, 480])
- isColor: 컬러 영상이면 True, 그렇지 않으면 False.
- retval: cv2.VideoWriter 객체

```
cv2.VideoWriter.open(filename, fourcc, fps, frameSize,  
isColor=None) -> retval
```

- retval: 성공하면 True, 실패하면 False.

OpenCV API

□ 비디오 파일이 준비되었는지 확인

```
cv2.VideoCapture.isOpened() -> retval
```

- retval: 성공하면 True, 실패하면 False.

□ 프레임 받아오기

```
cv2.VideoCapture.read(image) -> None
```

- image: 저장할 프레임 (numpy.ndarray)

동영상 파일 저장하기

- 카메라 입력 프레임에서 에지 영상을 구하여 동영상으로 저장하기

```
import sys
import cv2

# 카메라로부터 cv2.VideoCapture 객체 생성
cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("Camera open failed!")
    sys.exit()

# 동영상 저장을 위한 cv2.VideoWriter 객체 생성
w = round(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
h = round(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = 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))
```

동영상 파일 저장하기

- 카메라 입력 프레임에서 에지 영상을 구하여 동영상으로 저장하기 (Con't)

```
# 매 프레임 처리 및 화면 출력
while True:
    ret, frame = cap.read()
    if not ret:
        break

    edge = cv2.Canny(frame, 50, 150)
    edge = cv2.cvtColor(edge, cv2.COLOR_GRAY2BGR)
    out.write(edge)

    cv2.imshow('frame', frame)
    cv2.imshow('edge', edge)

    if cv2.waitKey(10) == 27:
        break

# 자원 해제
cap.release()
out.release()
cv2.destroyAllWindows()
```



break



파이썬 OpenCV 프로그래밍 입문과 활용

3. 명함 검출과 인식

『OpenCV 4로 배우는 컴퓨터 비전과 머신 러닝』 저자

황 선 규 / 공학박사

명함 검출과 인식

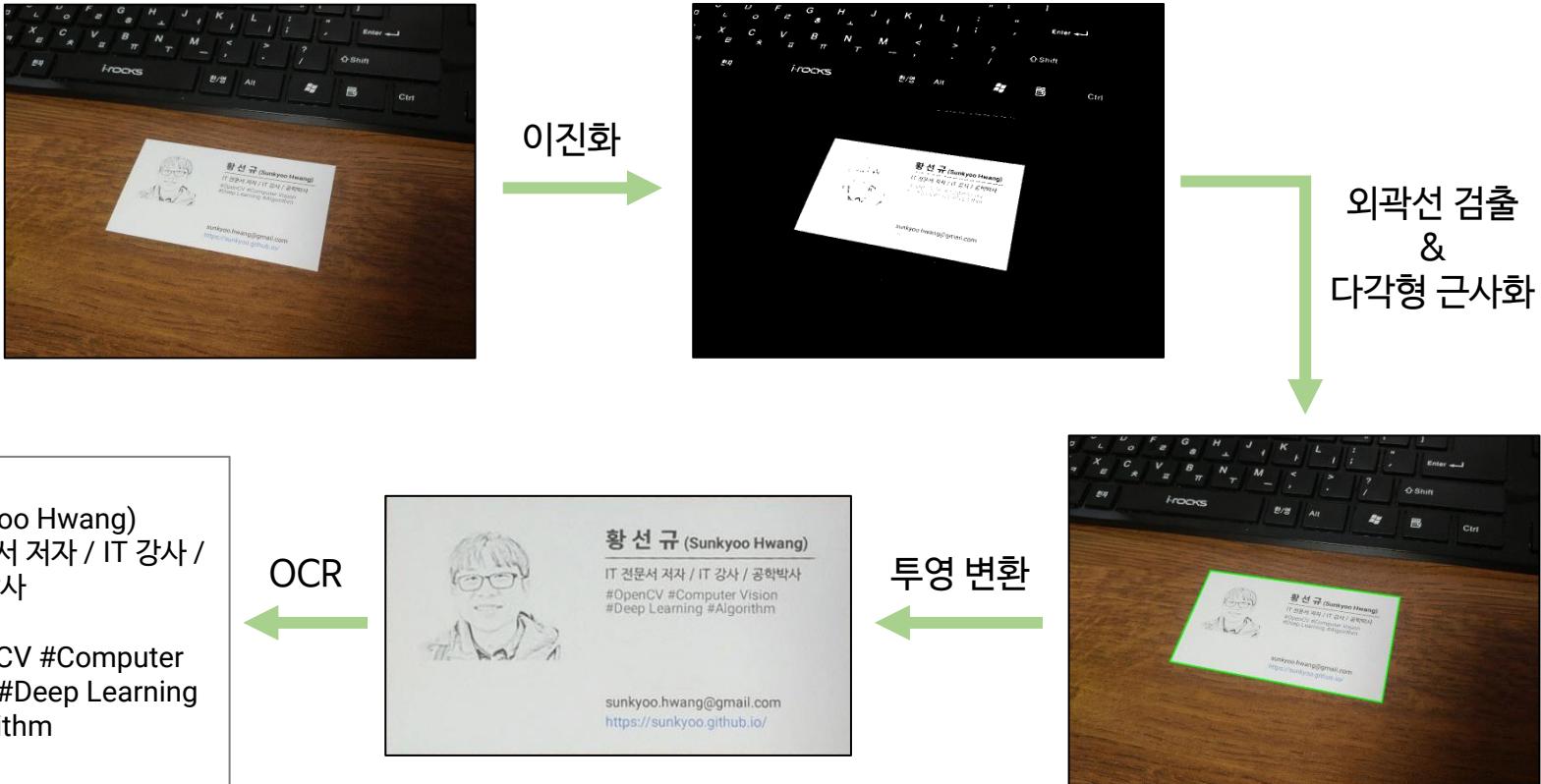
□ 일반적인 명함 사진의 조건

- 명함은 흰색 배경에 검정색 글씨이다.
- 명함은 충분히 크게 촬영되었다.
- 명함은 각진 사각형 모양이다.



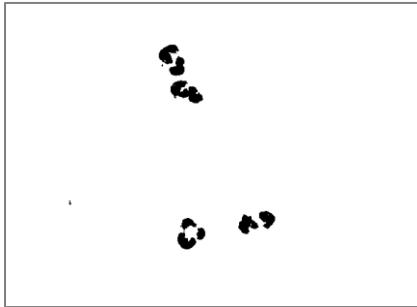
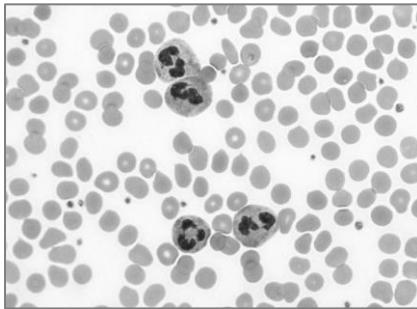
명함 검출과 인식

□ 명함 검출 및 인식 진행 과정



이진화

- 영상의 이진화(Binarization)란?
 - 영상의 픽셀 값을 0 또는 1(255)로 만드는 연산
 - 배경(background) vs. 객체(object)
 - 관심 영역 vs. 비관심 영역



motivation for this come segmentation and regioic circumstances. By reoating methods are appvariety of sources, occut to the majority of datmostly structured da

motivation for this come segmentation and regioic circumstances. By reoating methods are appvariety of sources, occut to the majority of datmostly structured da

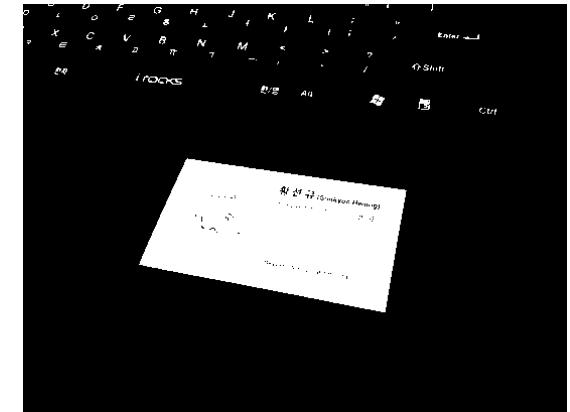
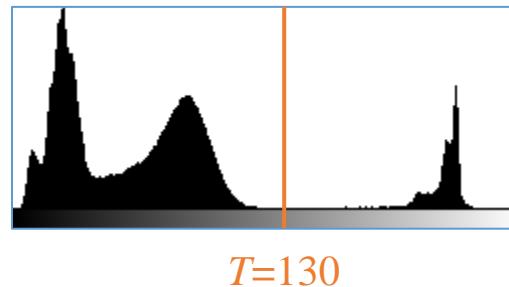


이진화

□ 그레이스케일 영상의 이진화

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

- T : 임계값, 문턱치, threshold



이진화: OpenCV API

□ 임계값 함수

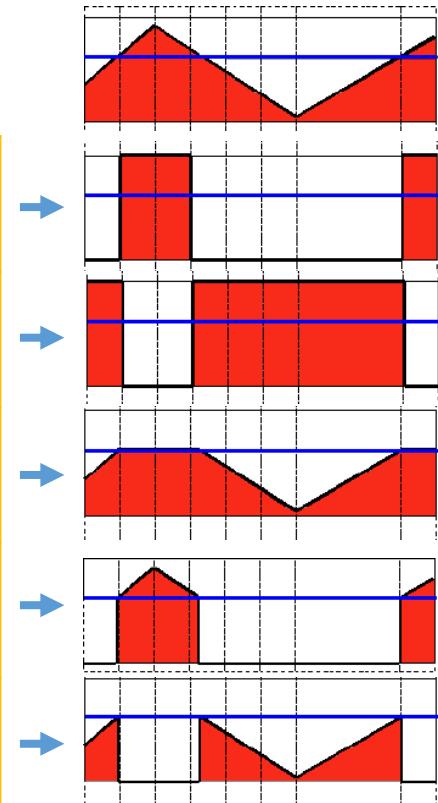
```
cv2.threshold(src, thresh, maxval, type, dst=None)
-> retval, dst
```

- src: 입력 영상. (다채널, 8비트 또는 32비트 실수형)
- thresh: 임계값
- maxval: THRESH_BINARY 또는 THRESH_BINARY_INV 방법을 사용할 때의 최댓값 지정
- type: 임계값에 의한 변환 함수 지정 또는 자동 임계값 설정 방법 지정 (cv.ThresholdTypes)
- retval: 사용된 임계값
- dst: (출력) 임계값 영상 (src와 동일 크기, 동일 타입)

이진화: OpenCV API

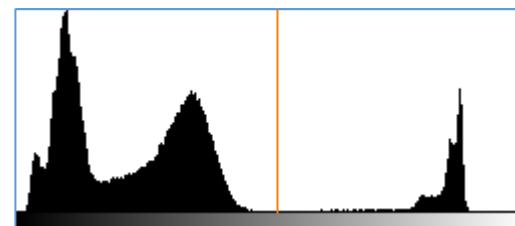
□ cv2.ThresholdTypes

cv2.THRESH_BINARY	$dst(x,y) = \begin{cases} \text{maxval} & \text{if } \text{src}(x,y) > \text{thresh} \\ 0 & \text{otherwise} \end{cases}$
cv2.THRESH_BINARY_INV	$dst(x,y) = \begin{cases} 0 & \text{if } \text{src}(x,y) > \text{thresh} \\ \text{maxval} & \text{otherwise} \end{cases}$
cv2.THRESH_TRUNC	$dst(x,y) = \begin{cases} \text{threshold} & \text{if } \text{src}(x,y) > \text{thresh} \\ \text{src}(x,y) & \text{otherwise} \end{cases}$
cv2.THRESH_TOZERO	$dst(x,y) = \begin{cases} \text{src}(x,y) & \text{if } \text{src}(x,y) > \text{thresh} \\ 0 & \text{otherwise} \end{cases}$
cv2.THRESH_TOZERO_INV	$dst(x,y) = \begin{cases} 0 & \text{if } \text{src}(x,y) > \text{thresh} \\ \text{src}(x,y) & \text{otherwise} \end{cases}$
cv2.THRESH_MASK	
cv2.THRESH_OTSU	Otsu 알고리즘으로 임계값 결정
cv2.THRESH_TRIANGLE	삼각 알고리즘으로 임계값 결정

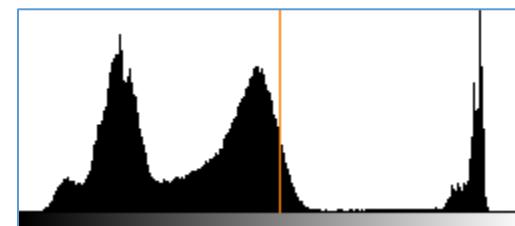
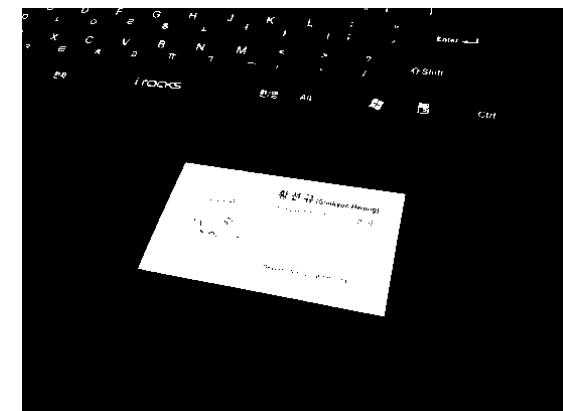


이진화: 임계값 결정 방법

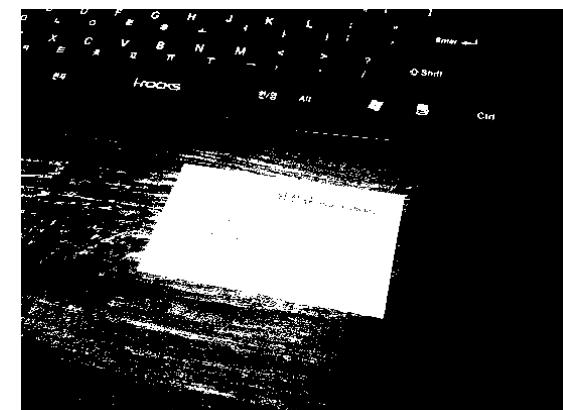
- 입력 영상의 밝기가 다른 경우



$T=130$



$T=130$

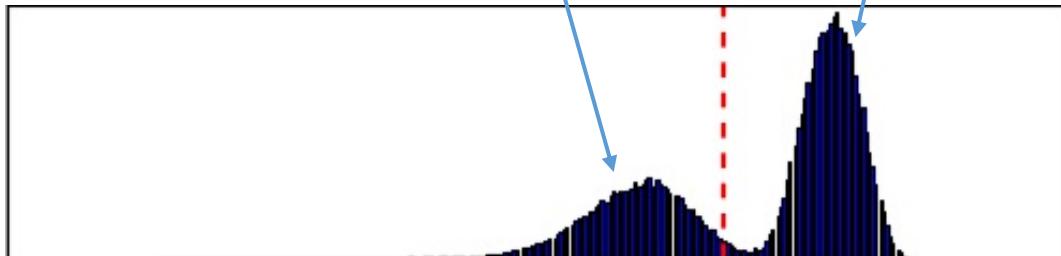


이진화: 임계값 결정 방법

□ 자동 임계값 결정 방법: Otsu 방법

- 입력 영상이 배경(background)과 객체(object) 두 개로 구성되어 있다고 가정 → bimodal histogram
- 두 픽셀 분포의 분산의 합이 최소가 되는 임계값을 선택 (Minimize within-class variance)
- 효과적인 수식 전개와 재귀식을 이용하여 빠르게 임계값을 결정

$$\sigma_{Within}^2(T) = \omega_1(T)\sigma_1^2(T) + \omega_2(T)\sigma_2^2(T)$$



T

자동 임계값 결정 방법 – Otsu 방법

□ Otsu 방법을 이용한 자동 이진화

```
import cv2

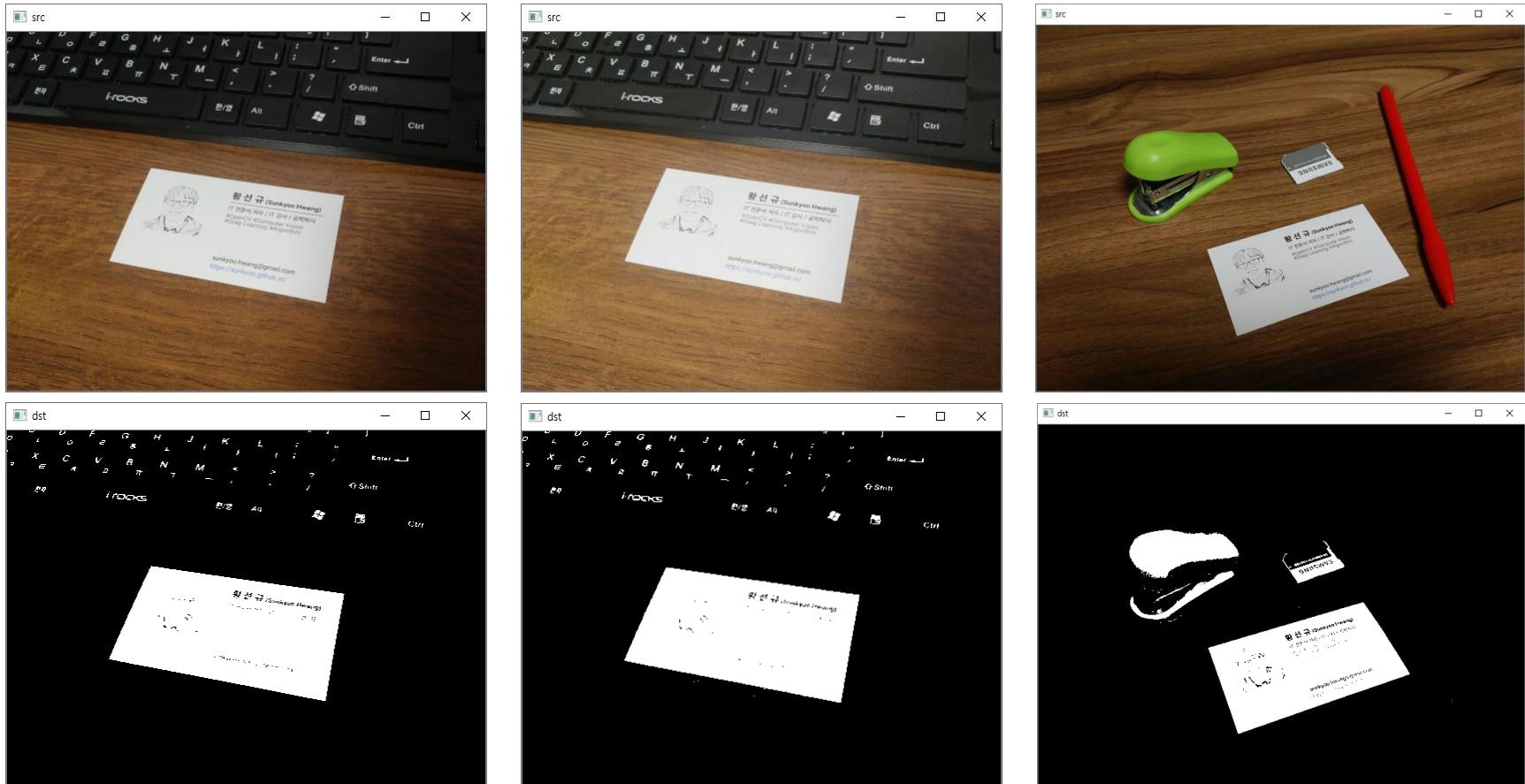
src = cv2.imread('namecard1.jpg', cv2.IMREAD_GRAYSCALE)

th, src_bin = cv2.threshold(src, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
print('threshold:', th)

cv2.imshow('src', src)
cv2.imshow('dst', dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

자동 임계값 결정 방법 – Otsu 방법

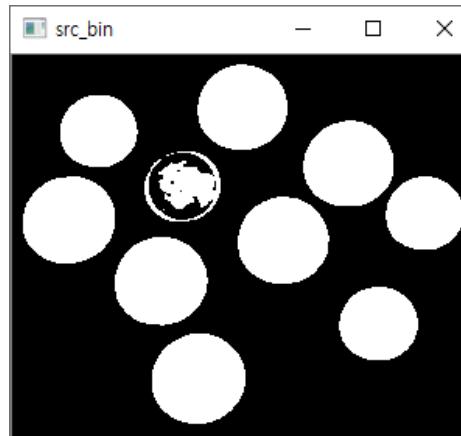
□ Otsu 방법을 이용한 자동 이진화



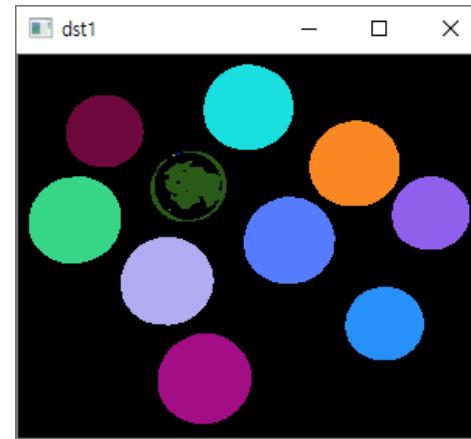
객체 단위 분석

- 객체 단위 분석
 - 흰색으로 표현된 객체를 분할하여 특징을 분석
 - 객체 위치 및 크기 정보, ROI 추출
- 레이블링 (Connected Component Labeling)
 - `cv2.connectedComponent()`, `cv2.connectedComponentWithStats()`
 - 서로 연결되어 있는 객체 픽셀에 고유한 번호를 지정
 - 각 객체의 바운딩 박스, 무게 중심 좌표로 함께 반환
- 외곽선 검출 (Contour Tracing)
 - `cv2.findContours()`
 - 각 객체의 외곽선 좌표를 모두 검출

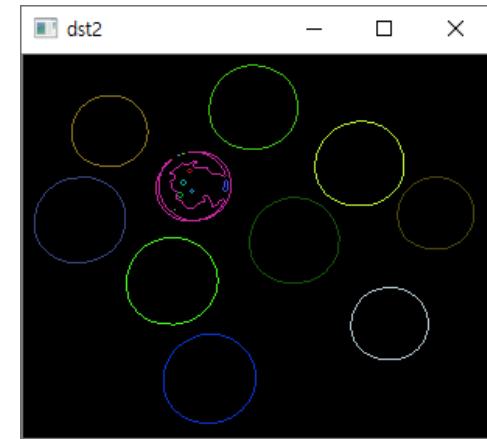
객체 단위 분석



레이블링



외곽선 검출



- 영역 기반 모양 분석
- 레이블맵, 바운딩 박스, 픽셀 개수, 무게 중심 좌표를 반환

- 외곽선 기반 모양 분석
- 외곽선 점들의 좌표와 계층 구조를 반환
- 다양한 외곽선 처리 함수에서 활용 가능
(근사화, 컨벡스헬 등)

외곽선 검출

- 외곽선 검출이란?
 - 객체의 외곽선 좌표를 모두 추출하는 작업
 - 바깥쪽 & 안쪽(홀) 외곽선
 - 외곽선의 계층 구조도 표현 가능
- 객체 하나의 외곽선 표현 방법
 - `numpy.ndarray`
 - `shape=(K, 1, 2), dtype=int32` (K는 외곽선 좌표 개수)
- 여러 객체의 외곽선 표현 방법
 - "객체 하나의 외곽선"을 원소로 갖는 리스트
 - 리스트 길이 = 외곽선 개수

외곽선 검출: OpenCV API

□ 외곽선 검출

```
cv2.findContours(image, mode, method, contours=None,  
                 hierarchy=None, offset=None) -> contours, hierarchy
```

- image: 입력 영상. non-zero 픽셀을 객체로 간주함.
- mode: 외곽선 검출 모드
- method: 외곽선 근사화 방법
- contours: 검출된 외곽선 좌표. [numpy.ndarray로 구성된 리스트](#).
`len(contours)=N,`
`contours[i].shape=(K, 1, 2)`
- hierarchy: 외곽선 계층 정보. [numpy.ndarray](#).
`shape=(1, N, 4).` `hierarchy[0, i, 0~3]`가 순서대로
`next, prev, child, parent` 외곽선 인덱스를 가리킴.
해당 외곽선이 없으면 -1을 가짐.

외곽선 검출: OpenCV API

□ 외곽선 검출 (Con't)

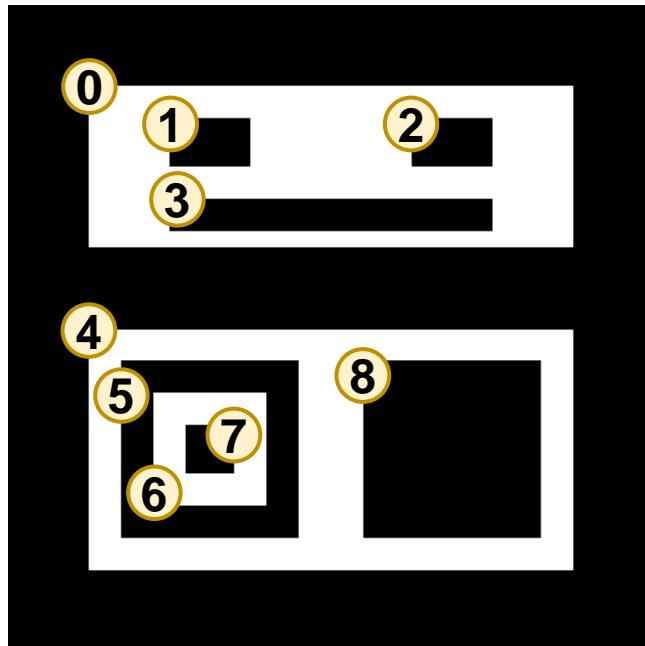
■ mode:

- | | | |
|--------------------------|--|---------|
| □ RETR_EXTERNAL : | 가장 바깥쪽 외곽선만 검출
(hierarchy[i][2]=hierarchy[i][3]=-1) | 계층 정보 X |
| □ RETR_LIST : | 계층 관계없이 모든 외곽선 검출
(hierarchy[i][2]=hierarchy[i][3]=-1) | |
| □ RETR_CCOMP : | 2레벨 계층 구조로 외곽선 검출
상위 레벨은 (흰색) 객체 외곽선,
하위 레벨은 (검정색) 구멍(hole) 외곽선. | 계층 정보 O |
| □ RETR_TREE : | 계층적 트리 구조로 모든 외곽선 검출 | |

외곽선 검출: OpenCV API

□ 외곽선 검출 (Con't)

- mode:



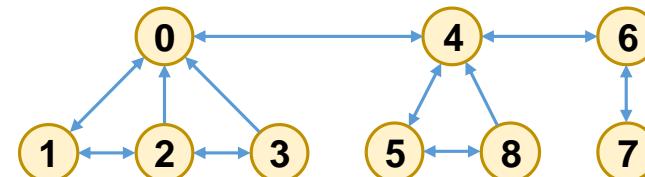
- RETR_EXTERNAL



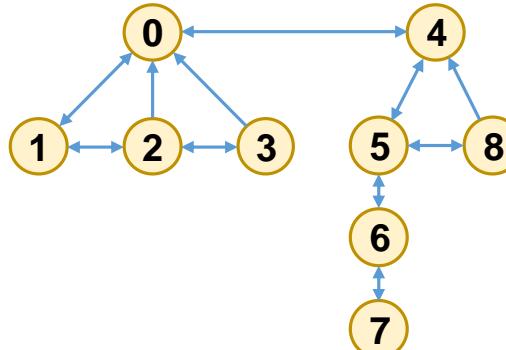
- RETR_LIST



- RETR_CCOMP



- RETR_TREE

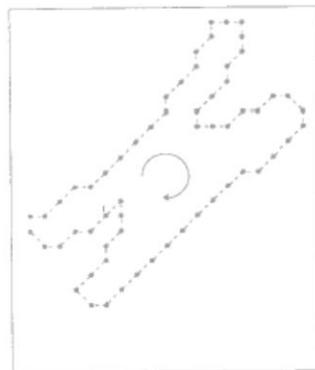


외곽선 검출: OpenCV API

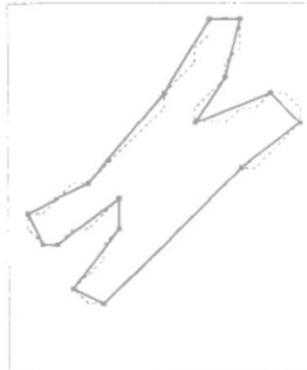
□ 외곽선 검출 (Con't)

■ method:

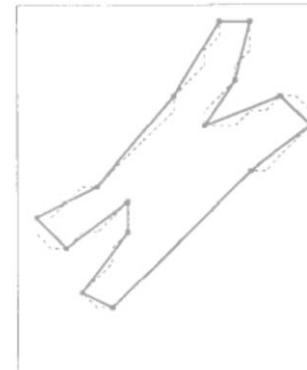
- **CHAIN_APPROX_NONE** : 근사화 없음
- **CHAIN_APPROX_SIMPLE** : 수직선, 수평선, 대각선에 대해 끝점만 사용하여 압축
- **CHAIN_APPROX_TC89_L1** : Teh & Chin L1 근사화
- **CHAIN_APPROX_TC89_KCOS** : Teh & Chin k cos 근사화



contours



L1



k cos

외곽선 검출

```
src = cv2.imread('namecard1.jpg', cv2.IMREAD_GRAYSCALE)

h, w = src.shape[:2]
dst1 = np.zeros((h, w, 3), np.uint8)
dst2 = np.zeros((h, w, 3), np.uint8)

# 이진화
_, src_bin = cv2.threshold(src, 0, 255, cv2.THRESH_OTSU)

# 외곽선 검출
contours1, _ = cv2.findContours(src_bin, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
contours2, _ = cv2.findContours(src_bin, cv2.RETR_LIST, cv2.CHAIN_APPROX_NONE)

for i in range(len(contours1)):
    c = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    cv2.drawContours(dst1, contours1, i, c, 1)

for i in range(len(contours2)):
    c = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    cv2.drawContours(dst2, contours2, i, c, 1)

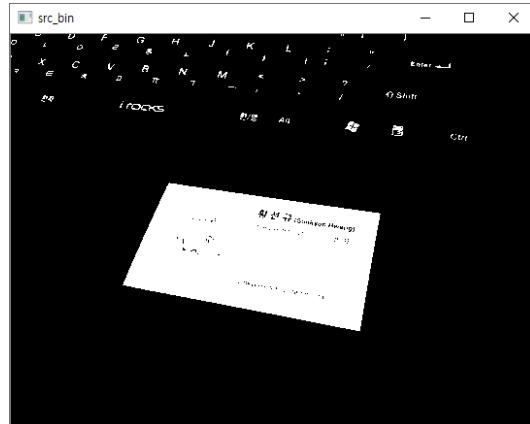
...

```

외곽선 검출



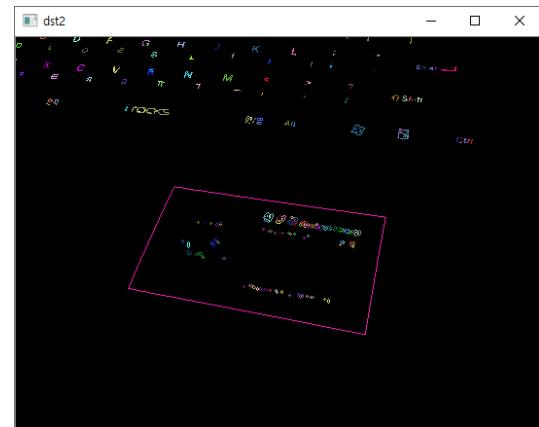
입력 영상



이진화



RETR_EXTERNAL



RETR_LIST

외곽선 관련 OpenCV API

□ 면적 구하기

```
cv2.contourArea(contour, oriented=None) -> retval
```

- contour: 외곽선 좌표. `numpy.ndarray. shape=(K, 1, 2)`
- oriented: True이면 외곽선 진행 방향에 따라 부호 있는 면적을 반환
- retval: 외곽선으로 구성된 면적

□ 외곽선 길이 구하기

```
cv2.arcLength(curve, closed) -> retval
```

- curve: 외곽선 좌표. `numpy.ndarray. shape=(K, 1, 2)`
- closed: True이면 폐곡선으로 간주
- retval: 외곽선 길이

외곽선 관련 OpenCV API

□ 바운딩 박스(외곽선을 외접하여 둘러싸는 가장 작은 사각형) 구하기

```
cv2.boundingRect(array) -> retval
```

- array: 외곽선 좌표. `numpy.ndarray. shape=(K, 1, 2)`
- retval: 사각형. (x, y, w, h)

□ 바운딩 서클(외곽선을 외접하여 둘러싸는 가장 작은 원) 구하기

```
cv2.minEnclosingCircle(points) -> center, radius
```

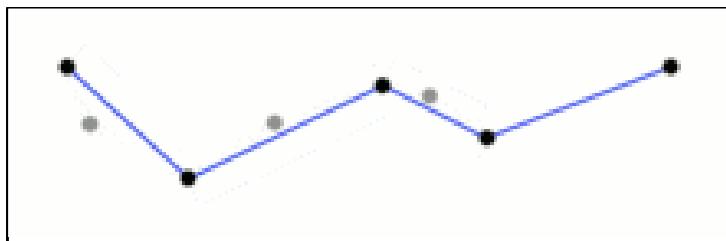
- points: 외곽선 좌표. `numpy.ndarray. shape=(K, 1, 2)`
- center: 바운딩 서클 중심 좌표 (x, y)
- radius: 바운딩 서클 반지름

외곽선 관련 OpenCV API

□ 외곽선 근사화

```
cv2.approxPolyDP(curve, epsilon, closed, approxCurve=None)  
-> approxCurve
```

- curve: 입력 곡선 좌표. `numpy.ndarray. shape=(K, 1, 2)`.
- epsilon: 근사화 정밀도 조절. 입력 곡선과 근사화 곡선 간의 최대 거리. e.g) ([외곽선 전체 길이](#)) * 0.02
- closed: `True`를 전달하면 폐곡선으로 간주
- approxCurve: 근사화된 곡선 좌표. `numpy.ndarray. shape=(K', 1, 2)`



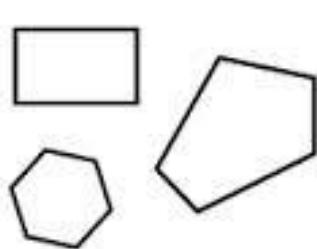
https://en.wikipedia.org/wiki/Ramer%20-%20Douglas%20-%20Peucker_algorithm

외곽선 관련 OpenCV API

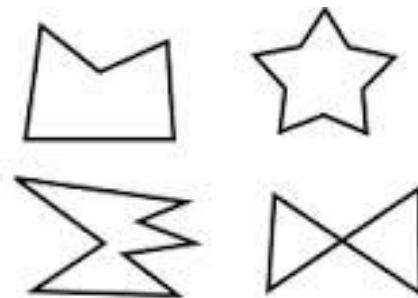
□ 컨벡스(Convex) 검사

```
cv2.isContourConvex(contour) -> retval
```

- contour: 입력 곡선 좌표. `numpy.ndarray. shape=(K, 1, 2)`.
- retval: 컨벡스이면 True, 아니면 False.



Convex Polygons



Non-convex Polygons

외곽선을 이용한 명함 검출

```
src = cv2.imread('namecard1.jpg')
src_gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)

h, w = src.shape[:2]
dst1 = np.zeros((h, w, 3), np.uint8)
dst2 = np.zeros((h, w, 3), np.uint8)

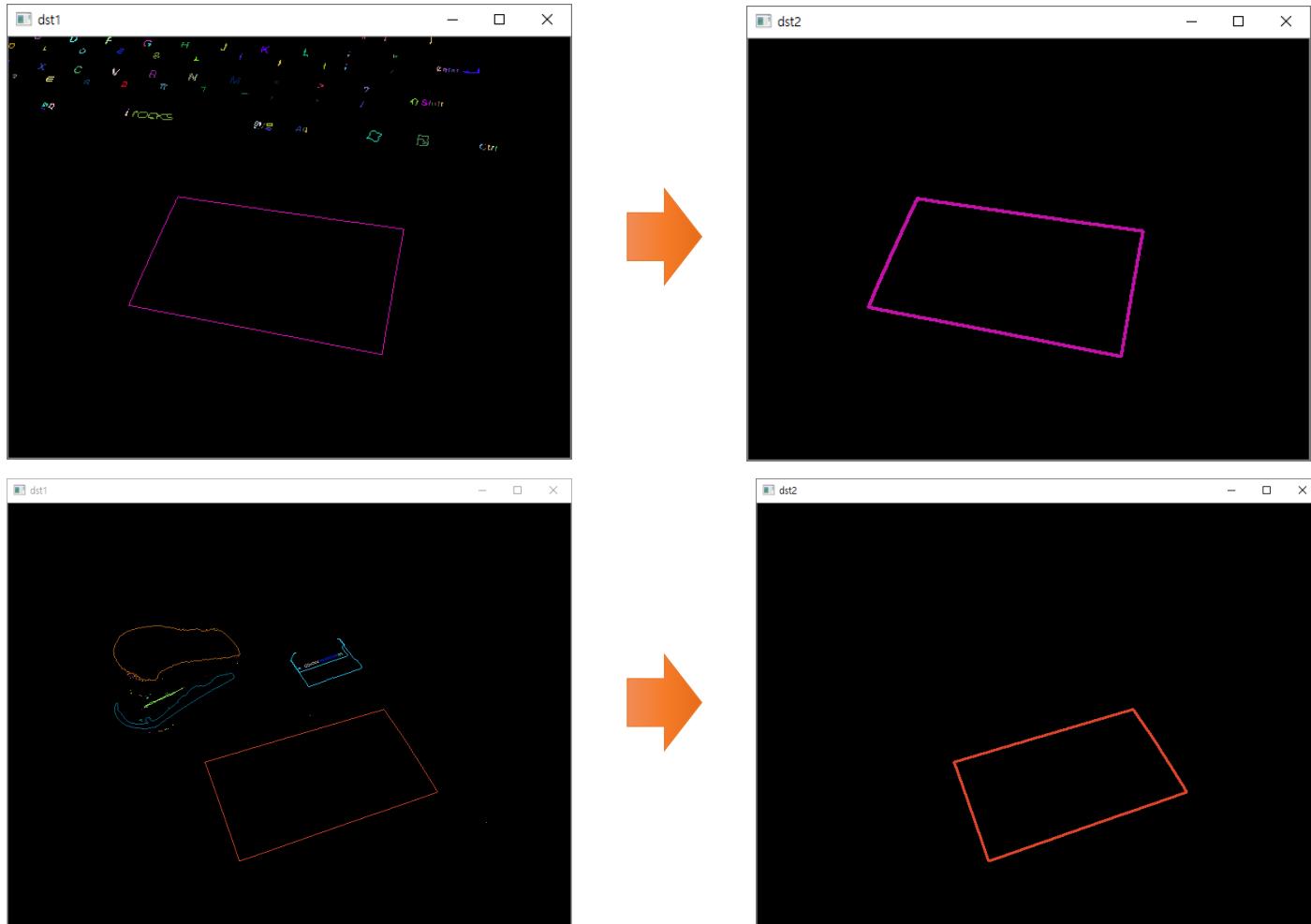
_, src_bin = cv2.threshold(src_gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
contours, _ = cv2.findContours(src_bin, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

for i in range(len(contours)):
    pts = contours[i]
    if (cv2.contourArea(pts) < 1000):
        continue

    approx = cv2.approxPolyDP(pts, cv2.arcLength(pts, True)*0.02, True)
    if not cv2.isContourConvex(approx):
        continue

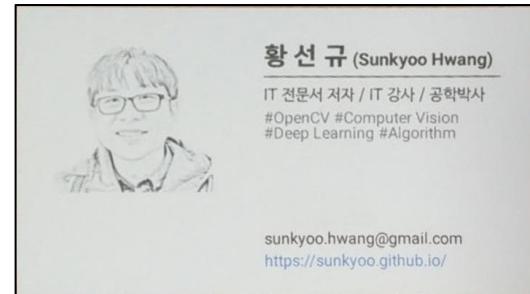
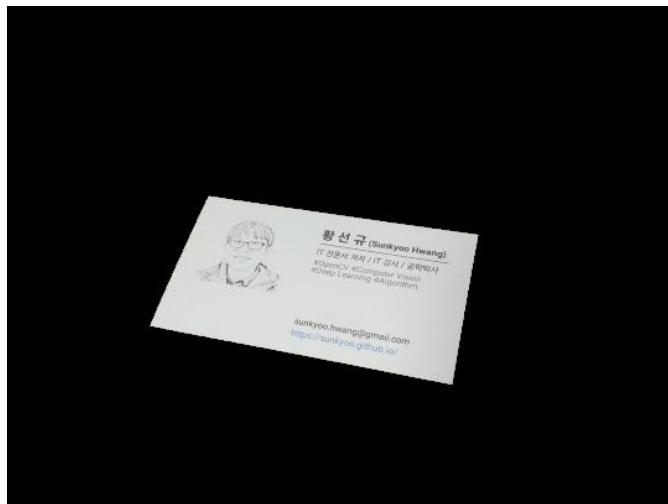
    if len(approx) == 4:
        cv2.drawContours(dst2, contours, i, c, 2)
```

외곽선을 이용한 명함 검출



영상의 기하학적 변환

- 영상의 기하학적 변환(geometric transformation)이란?
 - 영상을 구성하는 픽셀의 배치 구조를 변경함으로써 전체 영상의 모양을 바꾸는 작업
 - Image registration, removal of geometric distortion, etc.



영상의 기하학적 변환

Affine Transform



Translation



Shear



Scaling



Rotation



Combinations

2x3
matrix

Perspective Transform



Original



3x3
matrix

어파인 변환과 투시 변환

□ 어파인 변환(Affine Transform)

■ 6DOF

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

□ 투시 변환(Perspective Transform)

■ 8DOF

$$\begin{pmatrix} wx' \\ wy' \\ w \end{pmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

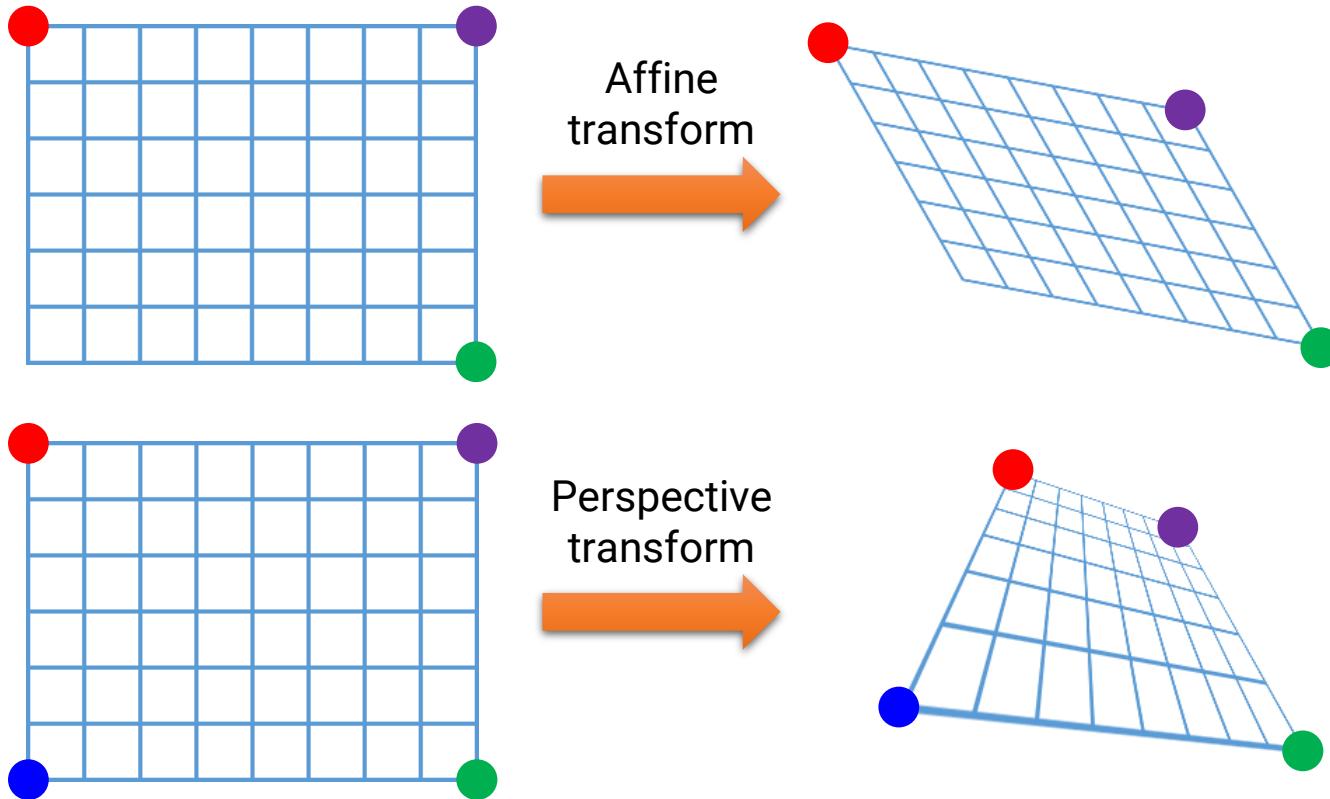
$$x' = \frac{p_{11}x + p_{12}y + p_{13}}{p_{31}x + p_{32}y + 1}$$

$$y' = \frac{p_{21}x + p_{22}y + p_{23}}{p_{31}x + p_{32}y + 1}$$

- DOF: Degree of Freedom

어파인 변환과 투시 변환

□ 어파인 변환 vs. 투시 변환



OpenCV API

□ 투시 변환 행렬 구하기

```
cv2.getPerspectiveTransform(src, dst, solveMethod=None) -> retval
```

- src: 4개의 원본 좌표점. `numpy.ndarray. shape=(4, 2)`
e.g) `np.array([[x1, y1], [x2, y2], [x3, y3], [x4, y4]], np.float32)`
- dst: 4개의 결과 좌표점. `numpy.ndarray. shape=(4, 2)`
- 반환값: 3x3 크기의 투시 변환 행렬

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \text{map_matrix} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

where $dst(i) = (x'_i, y'_i), src(i) = (x_i, y_i), i = 0, 1, 2, 3$

OpenCV API

□ 영상의 투시 변환

```
cv2.warpPerspective(src, M, dsize, dst=None, flags=None,  
borderMode=None, borderValue=None) -> dst
```

- src: 입력 영상
- M: 3x3 변환 행렬, 실수형
- dsize: 결과 영상의 크기. (0, 0)을 지정하면 src와 같은 크기.
- dst: 출력 영상
- flags: 보간법. 기본값은 cv2.INTER_LINEAR
- borderMode: 가장자리 픽셀 확장 방식.
- borderValue: cv2.BORDER_CONSTANT일 때 사용할 상수 값.
기본값은 0.

투시 변환 예제

□ 찌그러진 명함 평기

```
import sys
import numpy as np
import cv2

src = cv2.imread('namecard1.jpg')

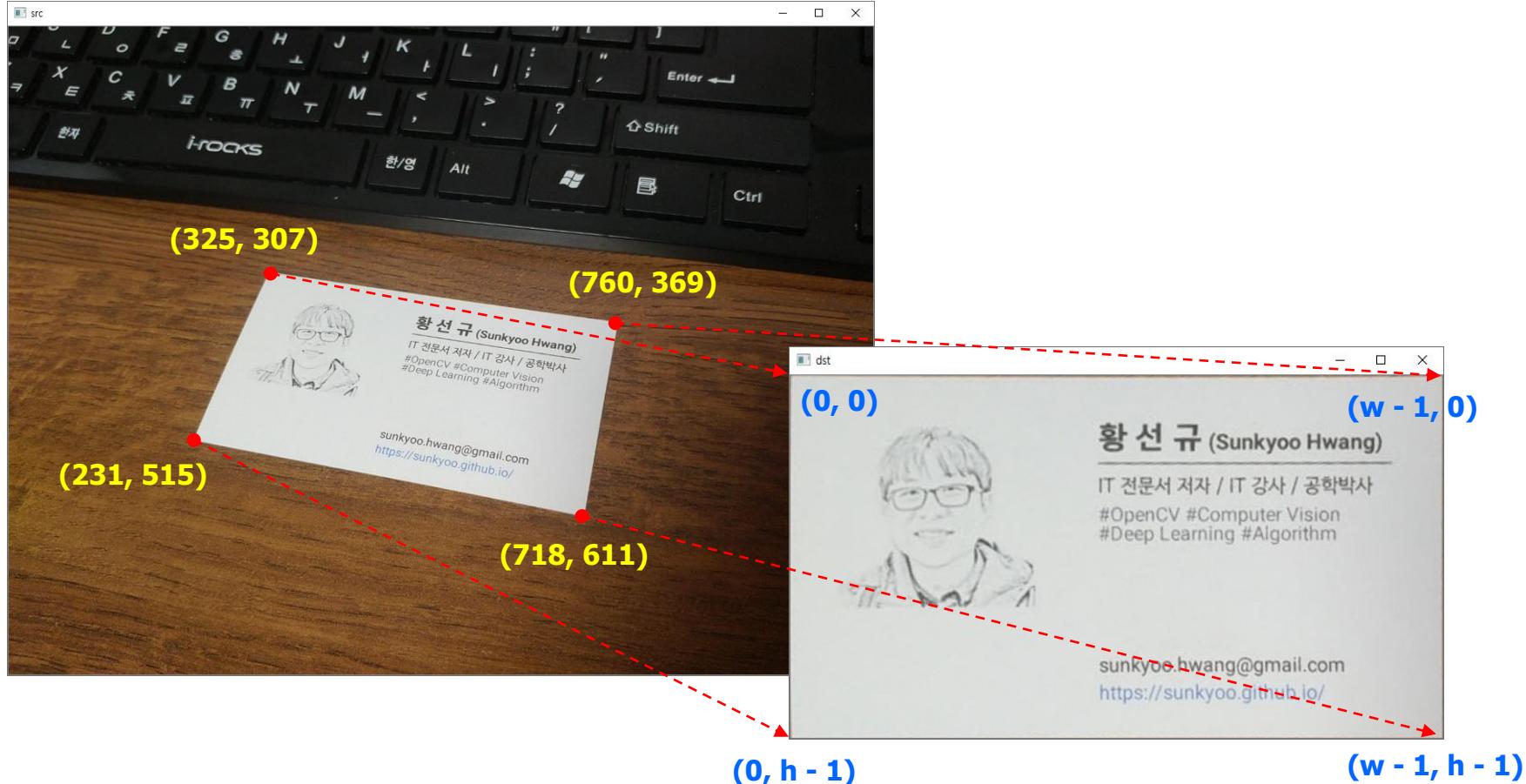
w, h = 720, 400
srcQuad = np.array([[325, 307], [760, 369], [718, 611], [231, 515]], np.float32)
dstQuad = np.array([[0, 0], [w-1, 0], [w-1, h-1], [0, h-1]], np.float32)

pers = cv2.getPerspectiveTransform(srcQuad, dstQuad)
dst = cv2.warpPerspective(src, pers, (w, h))

cv2.imshow('src', src)
cv2.imshow('dst', dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

투시 변환 예제

□ 찌그러진 명함 펴기 실행 화면



Tesseract 사용하기

- Tesseract
 - 광학 문자 인식(OCR) 라이브러리
 - <https://github.com/tesseract-ocr/tesseract>
 - 1985년~1994년 사이에 휴렛 팩커드에서 개발 → 2005년 오픈 소스
→ 2006년부터 구글에서 관리
 - 2018년 4.0이 발표되면서 LSTM(Long Short-Term Memory) 기반 OCR 엔진 및 모델이 추가
 - 총 116개의 언어가 제공
 - Apache License v2.0

- Windows Pre-built 설치 파일 다운로드
 - <https://github.com/UB-Mannheim/tesseract/wiki>
 - 독일 만하임 대학교(Mannheim University) 도서관에서 오래된 신문에 대해 OCR을 수행하기 위해 Tesseract를 사용

Tesseract 사용하기

- pytesseract 설치하기
 - <https://github.com/madmaze/pytesseract>
 - 파이썬에서 Tesseract를 사용하기 위해 필요한 패키지
 - > pip install pytesseract
- OpenCV/numpy와 사용하기
 - Tesseract는 numpy ndarray 객체를 지원
 - 다만 Tesseract는 RGB 컬러 포맷을 사용하므로 cv2.cvtColor() 함수를 이용하여 BGR 순서를 RGB 순서로 변경해야 함.

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

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
print(pytesseract.image_to_string(img_rgb))
```

명함 검출 및 인식 프로그램

```
import sys
import numpy as np
import cv2
import pytesseract

filename = 'namecard1.jpg'
if len(sys.argv) > 1:
    filename = sys.argv[1]

src = cv2.imread(filename)

dw, dh = 720, 400
srcQuad = np.array([[0, 0], [0, 0], [0, 0], [0, 0]], np.float32)
dstQuad = np.array([[0, 0], [0, dh], [dw, dh], [dw, 0]], np.float32)
dst = np.zeros((dh, dw), np.uint8)

src_gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
th, src_bin = cv2.threshold(src_gray, 0, 255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)

contours, _ = cv2.findContours(src_bin, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
```

명함 검출 및 인식 프로그램

```
for pts in contours:
    if cv2.contourArea(pts) < 1000:
        continue

    approx = cv2.approxPolyDP(pts, cv2.arcLength(pts, True)*0.02, True)

    if not cv2.isContourConvex(approx) or len(approx) != 4:
        continue

    srcQuad = reorderPts(approx.reshape(4, 2).astype(np.float32))

    pers = cv2.getPerspectiveTransform(srcQuad, dstQuad)
    dst = cv2.warpPerspective(src, pers, (dw, dh), flags=cv2.INTER_CUBIC)

    dst_rgb = cv2.cvtColor(dst, cv2.COLOR_BGR2RGB)
    print(pytesseract.image_to_string(dst_rgb, lang='Hangul+eng'))

cv2.imshow('src', src)
cv2.imshow('dst', dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

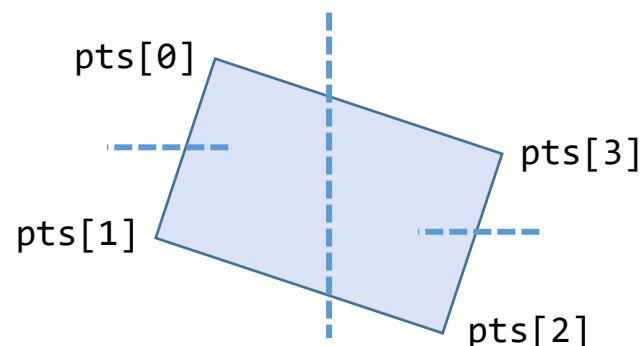
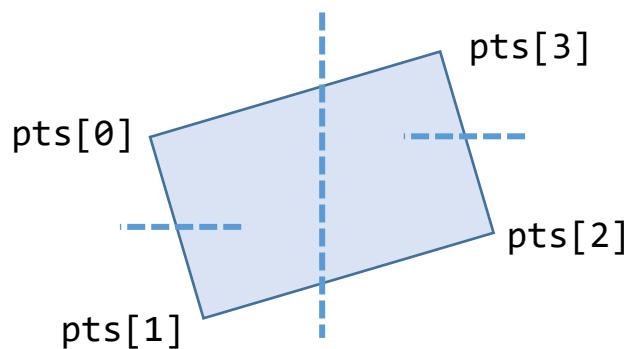
명함 검출 및 인식 프로그램

```
def reorderPts(pts):
    idx = np.lexsort((pts[:, 1], pts[:, 0])) # 칼럼0 -> 칼럼1 순으로 정렬한 인덱스를 반환
    pts = pts[idx] # x좌표로 정렬

    if pts[0, 1] > pts[1, 1]:
        pts[[0, 1]] = pts[[1, 0]]

    if pts[2, 1] < pts[3, 1]:
        pts[[2, 3]] = pts[[3, 2]]

    return pts
```



명함 검출 및 인식 프로그램

```
관리자: C:\Windows\System32\cmd.exe - python namecard.py namecard3.jpg
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. All rights reserved.

c:\#coding#\python#\opencv#\t-academy#\3교시>python namecard.py
황 선 규 (Sunkyoo Hwang)

IT 전문서 저자 / IT 강사 / 공학 박사

#OpenCV #Computer Vision
#Deep Learning #Algorithm

sunkyoo.hwang@gmail.com
https://sunkyoo.github.io/

c:\#coding#\python#\opencv#\t-academy#\3교시>
c:\#coding#\python#\opencv#\t-academy#\3교시>python namecard.py namecard3.jpg
황 선 규 (Sunkyoo Hwang)

17 전문서 저자 / 17 강사 / 공학 박사

#OpenCV #Computer Vision
#Deep Learning #Algorithm

sunkyoo.hwang@gmail.com
https://sunkyoo.github.io/
```

dst

황 선 규 (Sunkyoo Hwang)

IT 전문서 저자 / IT 강사 / 공학박사
#OpenCV #Computer Vision
#Deep Learning #Algorithm

sunkyoo.hwang@gmail.com
https://sunkyoo.github.io/



break



파이썬 OpenCV 프로그래밍 입문과 활용

4. 딥러닝 활용과 얼굴 검출

『OpenCV 4로 배우는 컴퓨터 비전과 머신 러닝』 저자

황 선 규 / 공학박사

OpenCV와 얼굴 검출

- OpenCV에서 지원하는 얼굴 검출 기법
 - Haar Cascade 방법
 - 2001년 Viola & Jones에 의해 제안된 방법
 - Haar-like 특징과 Adaboost 알고리즘, Cascade 구조를 사용하여 빠르고 정확한 얼굴 검출을 수행



- DNN(Deep Neural Net) 방법
 - OpenCV 3.3.1부터 DNN 모듈을 사용한 얼굴 검출을 기본 예제로 제공 (2017년)
 - ResNet-10과 SSD를 기반으로 학습된 얼굴 검출 네트워크 사용

OpenCV와 얼굴 검출

- Haar cascade 얼굴 검출 과정 시각화



<https://www.youtube.com/watch?v=hPCTwxF0qf4>

OpenCV와 얼굴 검출

□ OpenCV DNN 얼굴 검출

- 기존의 CascadeClassifier보다 대체로 더 좋은 성능을 나타냄

	Haar Cascade	DL
Size on disk	528KB	10MB(fp32), 5MB(fp16)
Efficiency @300x300	30ms	9.34ms

- 정면 얼굴, 측면 얼굴, 가려짐이 있어도 검출 가능

딥러닝

□ 딥러닝(Deep Learning)이란?

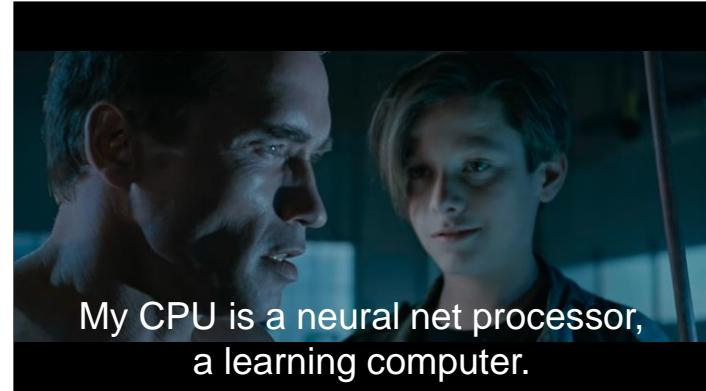
- 2000대부터 사용되고 있는 심층 신경망(deep neural network)의 또 다른 이름



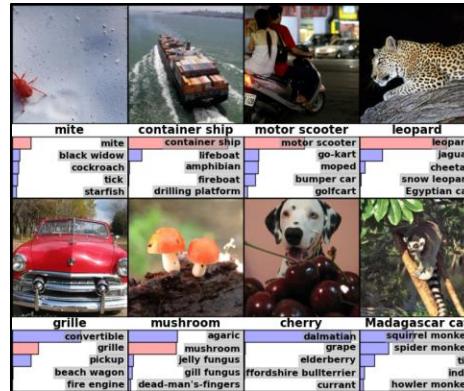
WIKIPEDIA
The Free Encyclopedia

Deep learning is part of a broader family of machine learning methods based on artificial neural networks.

https://en.wikipedia.org/wiki/Deep_learning



Terminator 2 (1991)



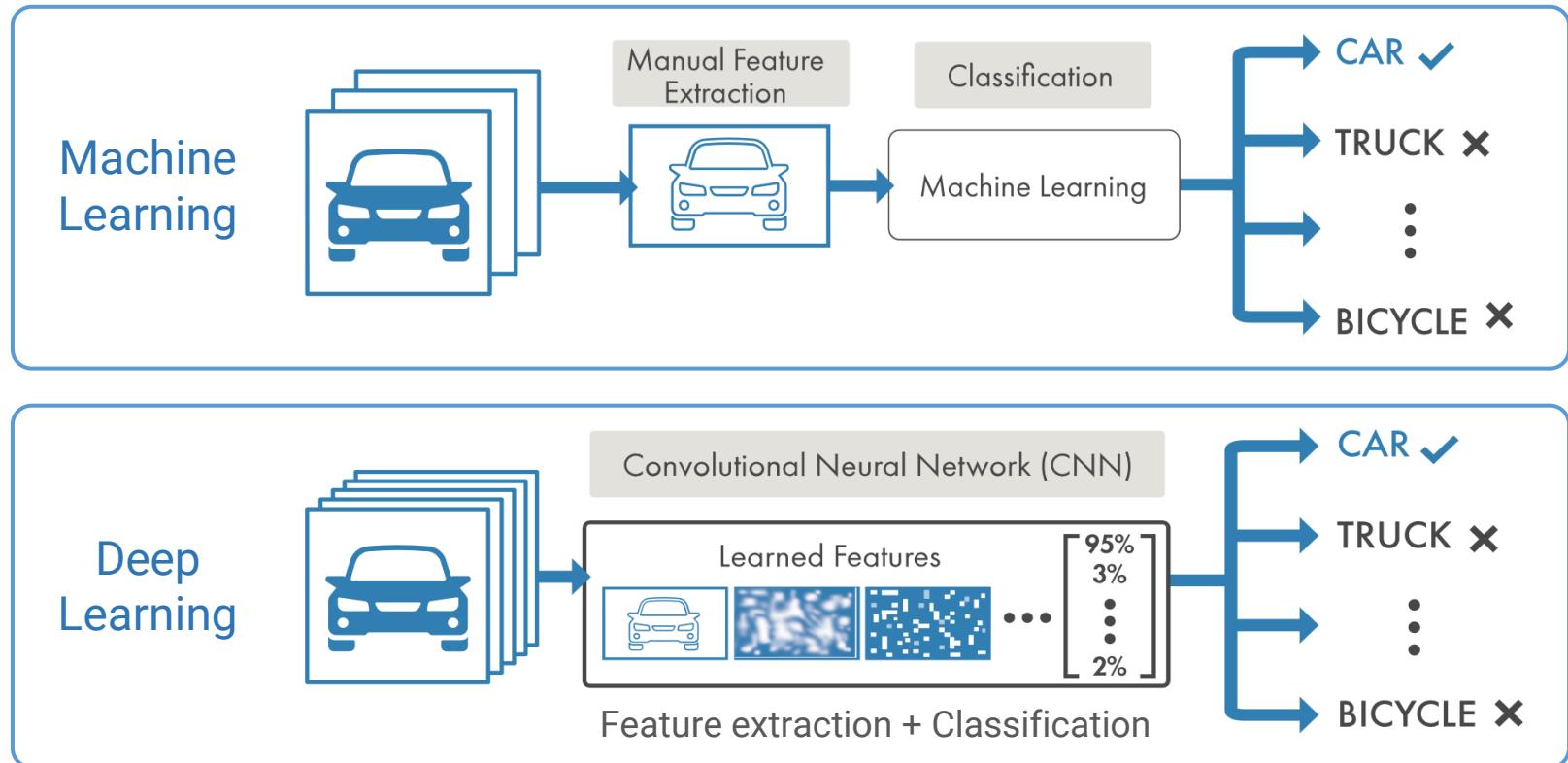
AlexNet (2012)



Alpha Go (2016)

딥러닝과 머신 러닝

□ 머신 러닝(ML) vs. 딥러닝(DL)

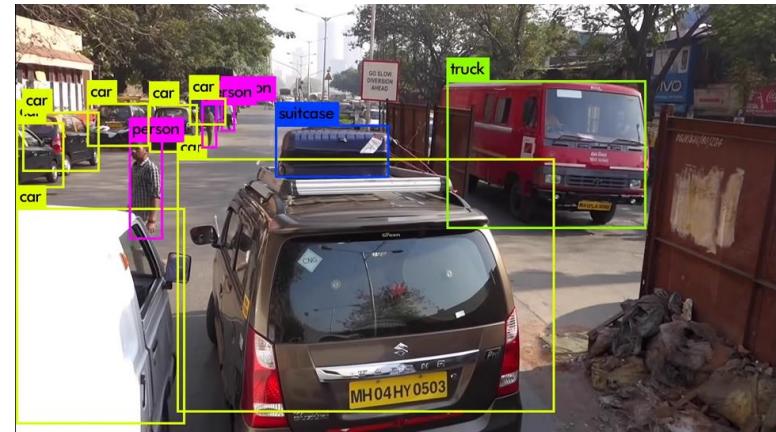


<https://www.mathworks.com/discovery/deep-learning.html> / <https://youtu.be/-SgkLEuhfbg>

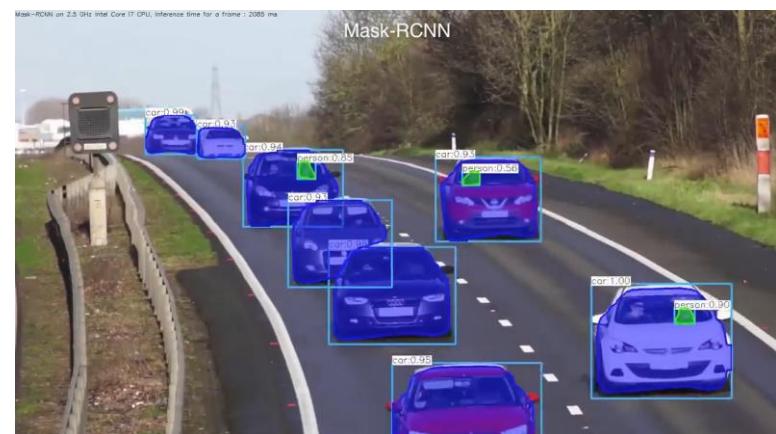
딥러닝 활용



Image Recognition

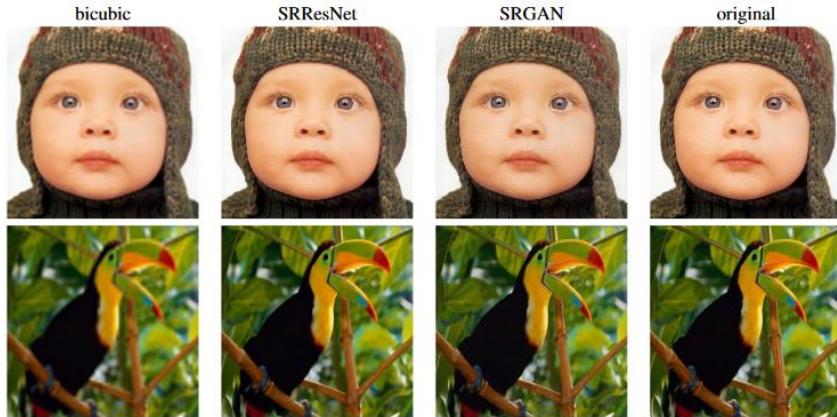


Object Detection



Object Segmentation

딥러닝 활용



Super-Resolution

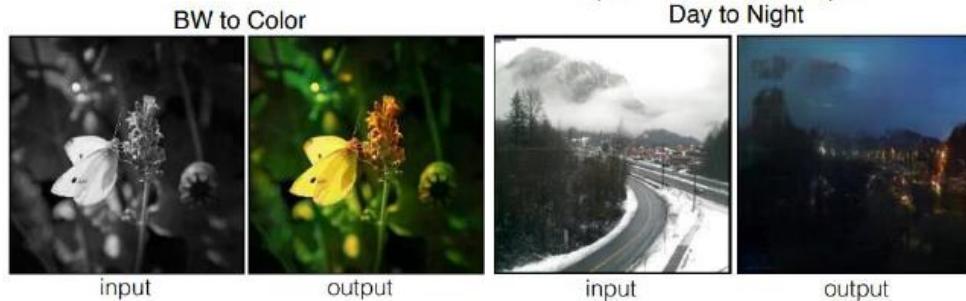


Image-to-Image Translation

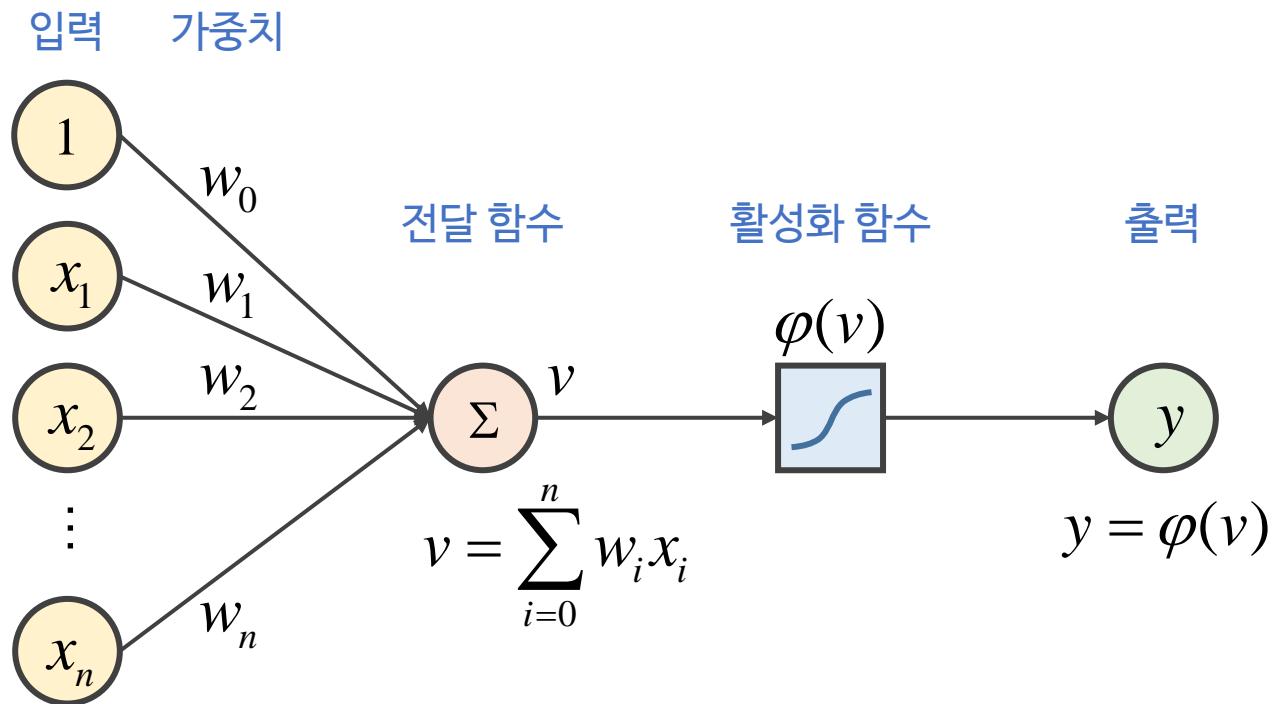


Image Inpainting

신경망 기초 이론

□ 퍼셉트론(Perceptron)

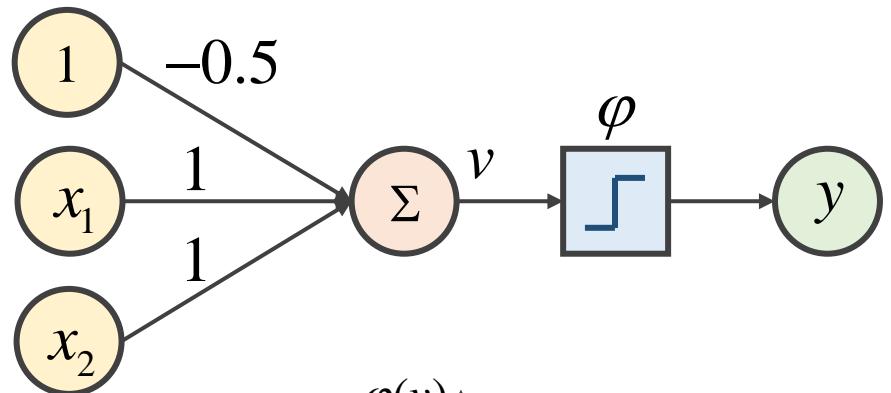
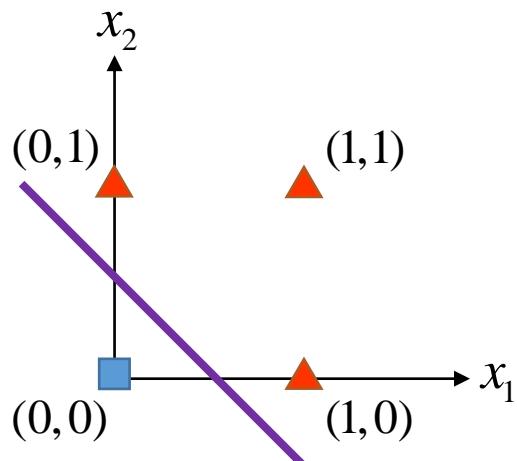
- 다수의 입력으로부터 가중합을 계산하고, 이를 이용하여 하나의 출력을 만들어내는 구조(1950년대)



신경망 기초 이론

□ 퍼셉트론에 의한 OR 연산 구현

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



$$y = \begin{cases} 1 & \text{if } x_1 + x_2 - 0.5 \geq 0 \\ 0 & \text{if } x_1 + x_2 - 0.5 < 0 \end{cases}$$

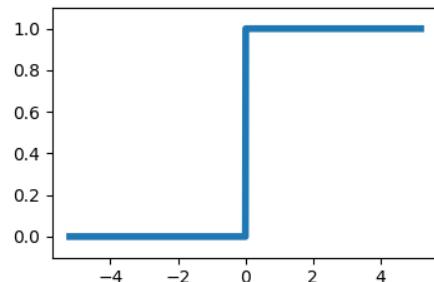
신경망 기초 이론

□ 활성화 함수(activation function)

- 생물학적 뉴런(neuron)에서 입력 신호가 일정 크기 이상일 때만 신호를 전달하는 메커니즘을 모방한 함수
- 비선형 함수를 사용

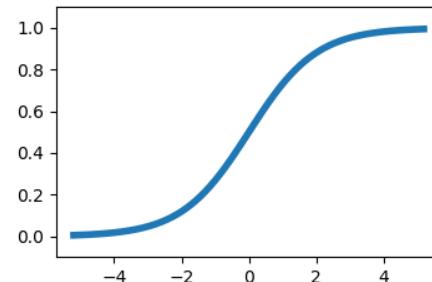
[Step function]

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$



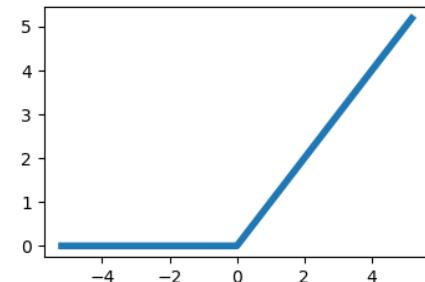
[Sigmoid]

$$f(x) = \frac{1}{1 + e^{-x}}$$



[ReLU]

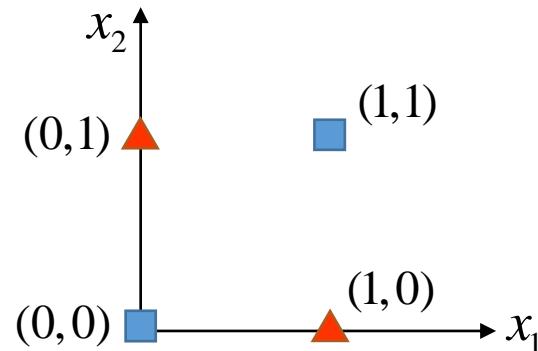
$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$



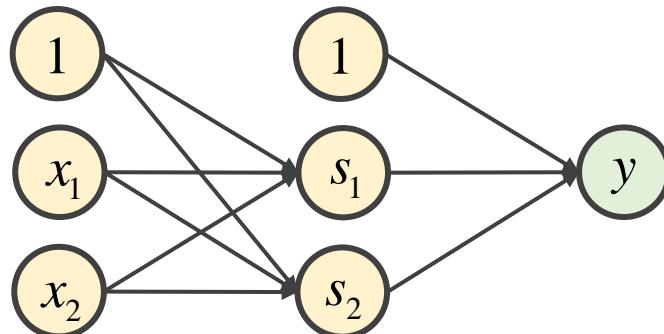
신경망 기초 이론

- 퍼셉트론에 의한 XOR 연산을 구현하려면?
 - XOR는 단순한 입력-출력 구조의 퍼셉트론으로 구현 불가 (Not linear!)

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



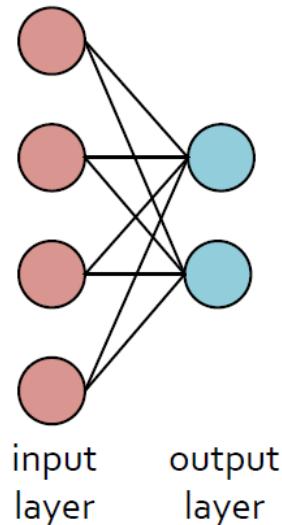
- Solution?
 - 다층 퍼셉트론
(Multi-layer perceptron)



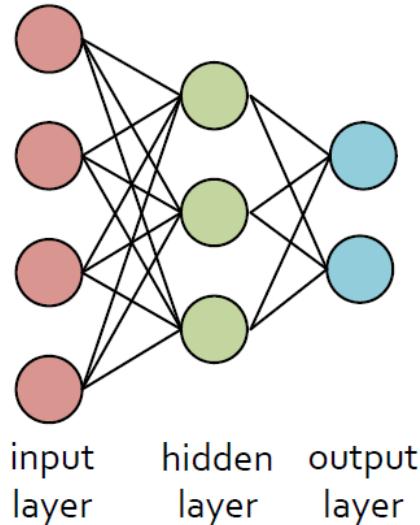
신경망 기초 이론

- 다층 퍼셉트론(MLP, Multi-Layer Perceptron)과 심층 신경망(Deep Neural Network)

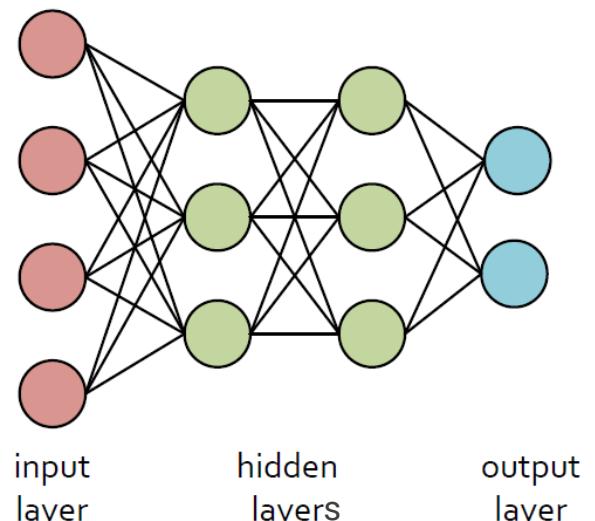
Single-Layer Perceptron



Multi-Layer Perceptron



Deep Neural Network

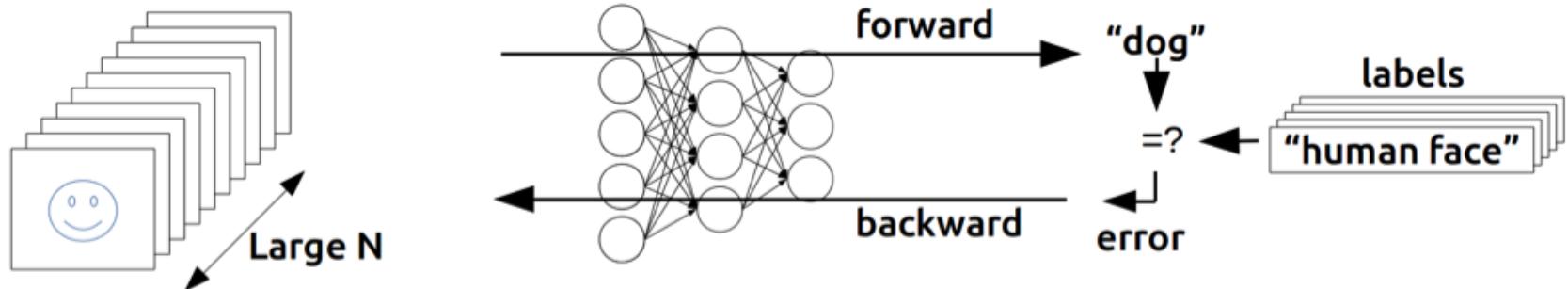


신경망 기초 이론

□ 학습 알고리즘

- Weight를 조절하는 방법
- 학습의 기준이 되는 비용(cost) 또는 손실(loss) 함수를 정의한 후, 비용을 최소화하는 방향으로 학습을 진행
- 대부분의 신경망 모델은 경사 하강법(gradient descent) 또는 오류역전파(error backpropagation) 등의 알고리즘을 사용

Training



심층 신경망의 문제점과 해결 방안

- 학습이 제대로 안 됨
 - Vanishing Gradient
 - Overfitting
 - Local minima
 - Weight initialization?

- 학습이 너무 느림
 - 하드웨어 성능이 낮음
 - Gradient Decent

심층 신경망의 문제점과 해결 방안

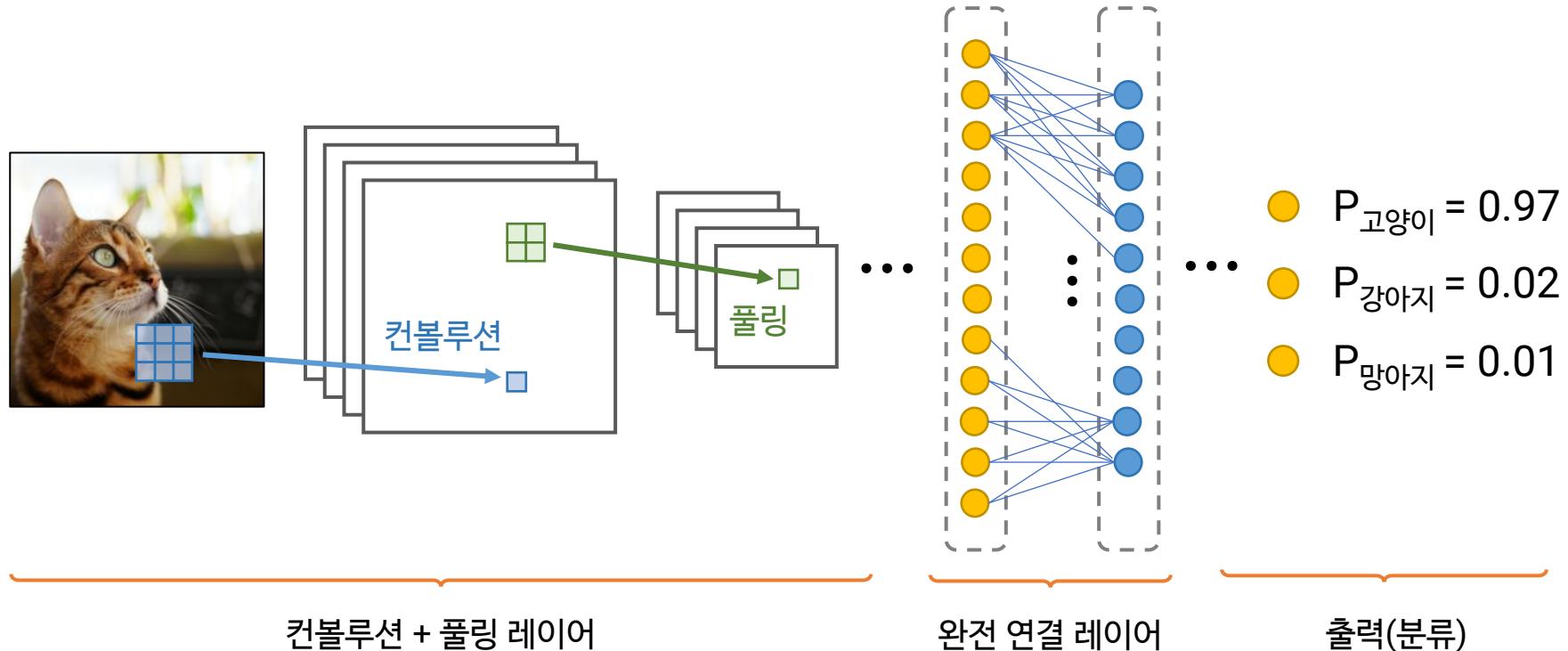
- 학습이 제대로 안 됨
 - Vanishing Gradient → ReLU(Rectified Linear Units) 사용
 - Overfitting → Regularization (Dropout, Batch Normalization)
 - Local minima → Don't worry!
 - Weight initialization? → 일정 범위 안의 임의의 값. (Xavier method)

- 학습이 너무 느림
 - 하드웨어 성능이 낮음 → CPU, GPU 발전
 - Gradient Decent → SGD, Adam method

- Large dataset (e.g. ImageNet, COCO, etc)

컨볼루션 신경망: CNN

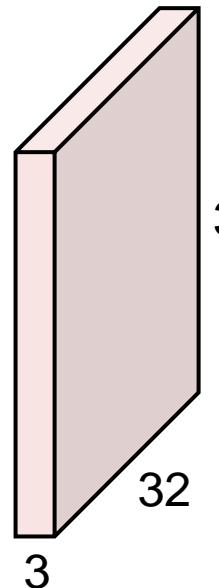
- 컨볼루션 신경망(CNN: Convolutional Neural Network)
 - 영상 인식 등을 위한 딥러닝에 특화된 네트워크 구조
 - 일반적 구성: 컨볼루션 + 풀링(pooling) + ... + 완전 연결 레이어(FCN)



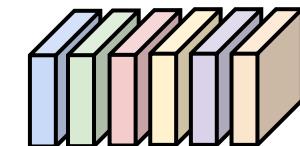
컨볼루션 신경망: CNN

- 컨볼루션 레이어(Convolution Layer)
 - 2차원 영상에서 유효한 특징(feature)를 찾아내는 역할
 - 유용한 필터 마스크가 학습에 의해 결정됨
 - 보통 ReLU 활성화 함수를 함께 사용함

32x32x3 image

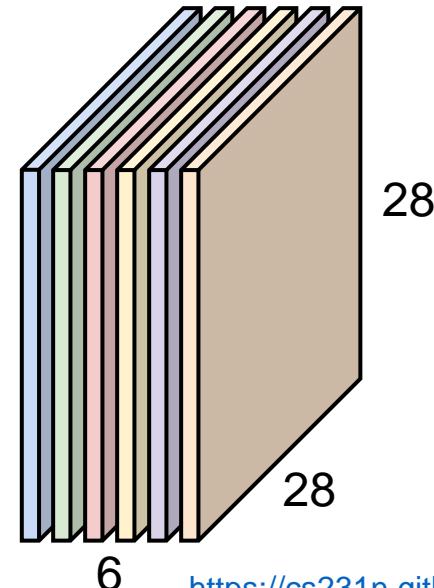


6 (5x5x3) filters



convolve (slide) over all
spatial locations

28x28x6 (activation maps)



<https://cs231n.github.io/>

컨볼루션 신경망: CNN

□ 풀링 레이어(Pooling Layer)

- 유용한 정보는 유지하면서 입력 크기를 줄임으로써 과적합(overfitting)을 예방하고 계산량을 감소시키는 효과를 가짐
- 보통 최대 풀링(max pooling) 또는 평균 풀링(average pooling) 사용
- 학습이 필요 없음

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

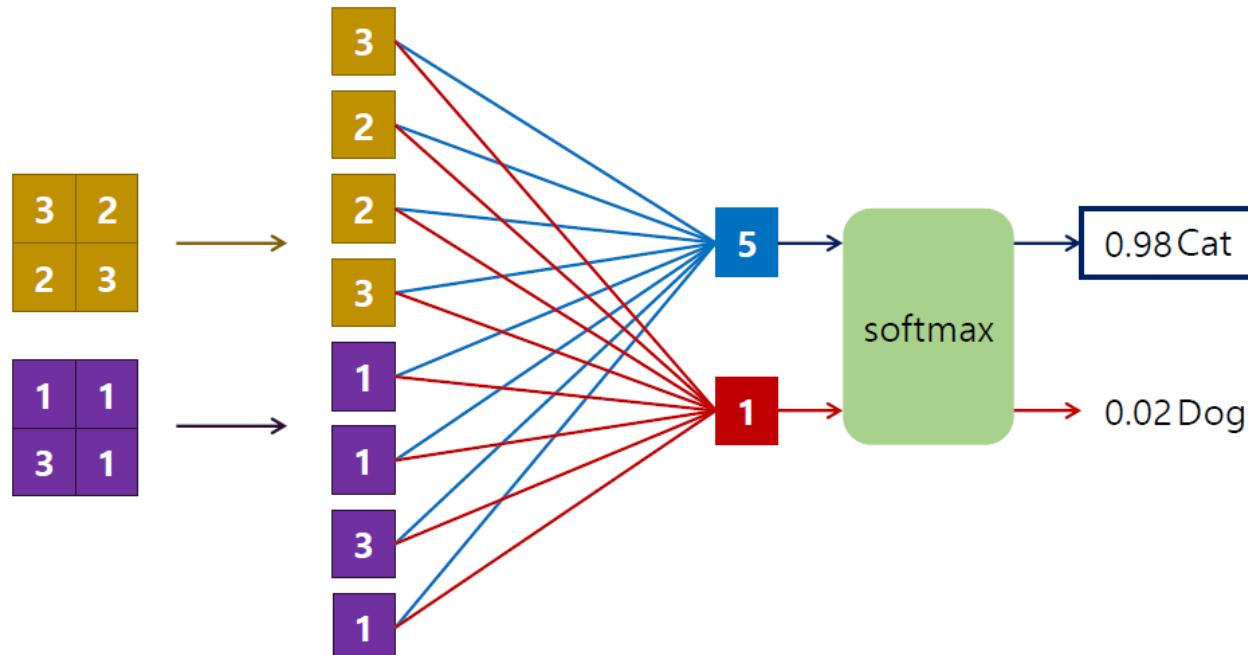
max pool with 2x2 filters
and stride 2



6	8
3	4

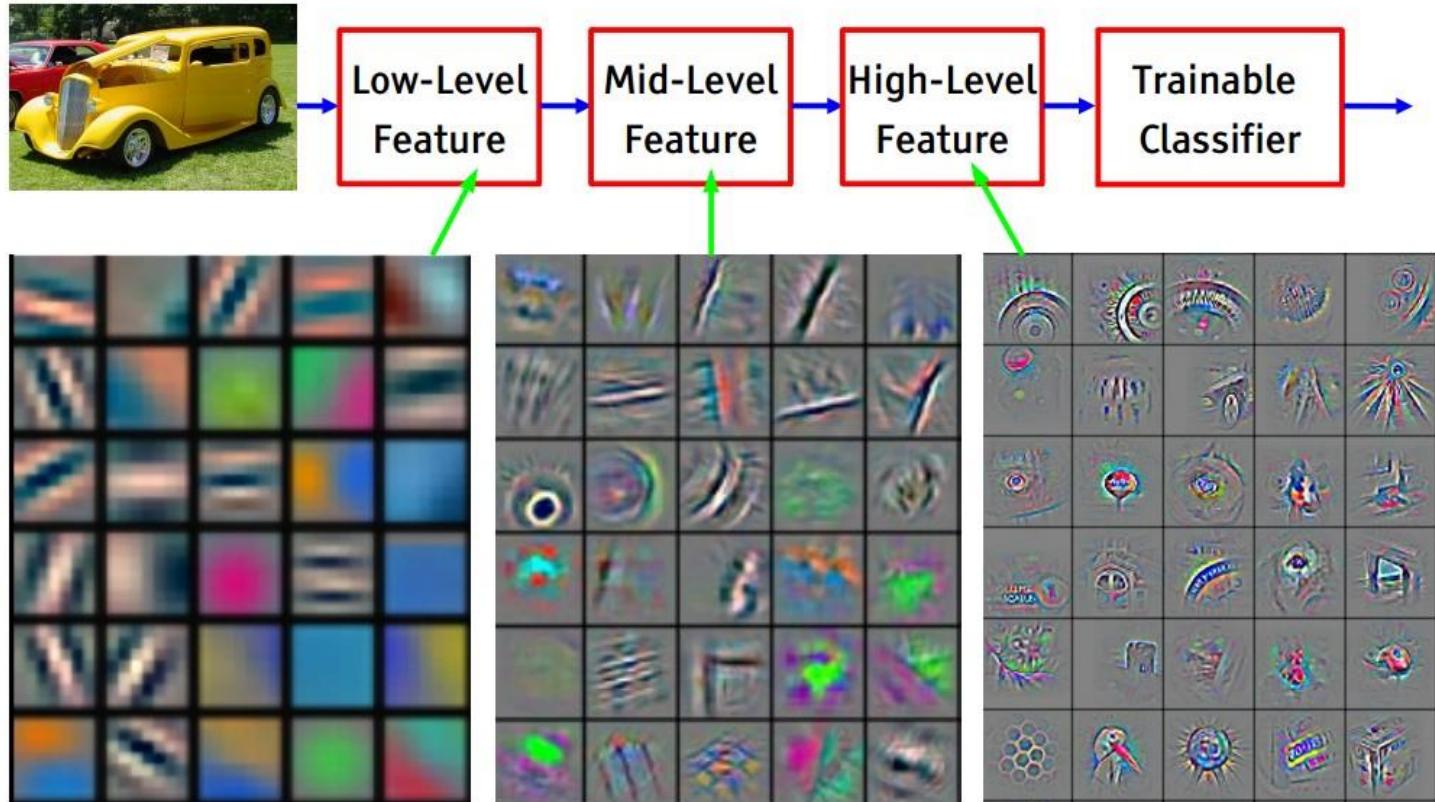
컨볼루션 신경망: CNN

- 완전 연결 레이어(Fully Connected Layer)
 - 3차원 구조의 activation map($H \times W \times C$)의 모든 값을 일렬로 이어 붙임
 - 인식의 경우, 소프트맥스(softmax) 레이어를 추가하여 각 클래스에 대한 확률 값을 결과로 얻음



컨볼루션 신경망: CNN

- 학습된 컨볼루션 레이어 필터의 예

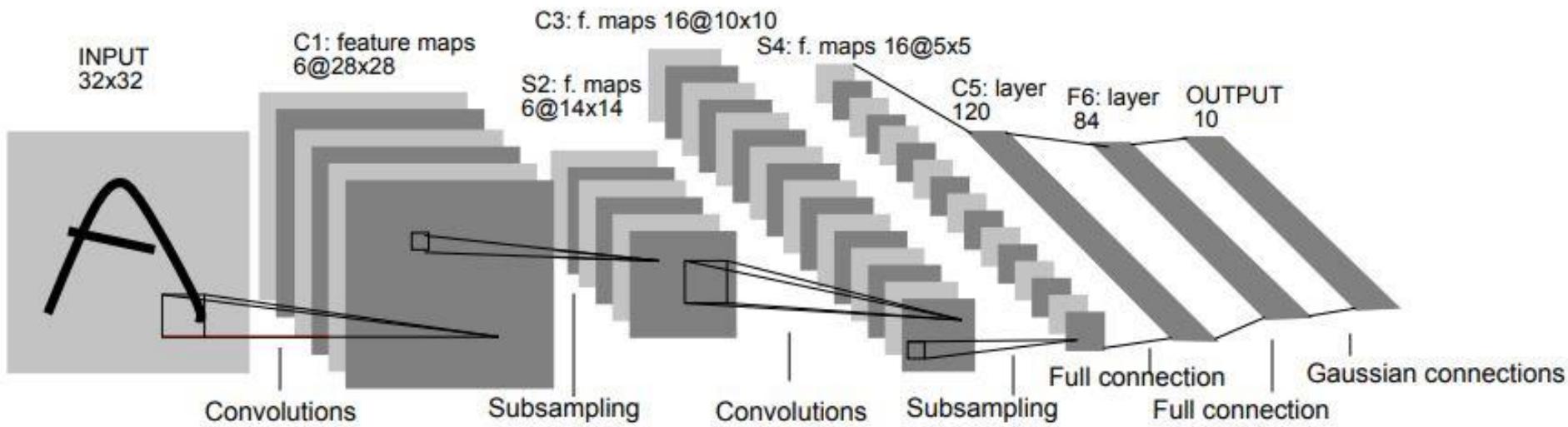


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

컨볼루션 신경망의 예

□ 필기체 숫자 인식을 위한 LeNet-5 (LeCun et al., 1998)

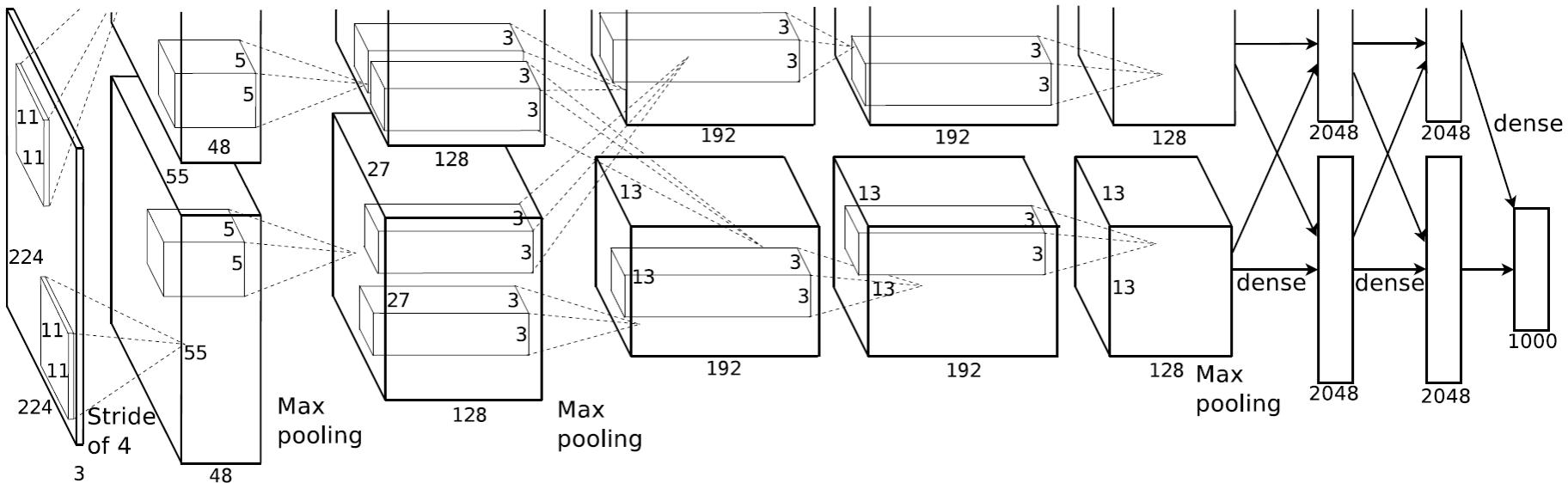
- CNN 원조
- 컨볼루션: 5×5 , Stride 1
- 풀링: 평균, 2×2 , Stride 2
- 입력: $32 \times 32 [0, 1]$ 정규화된 영상



컨볼루션 신경망의 예

□ AlexNet (Krizhevsky, 2012)

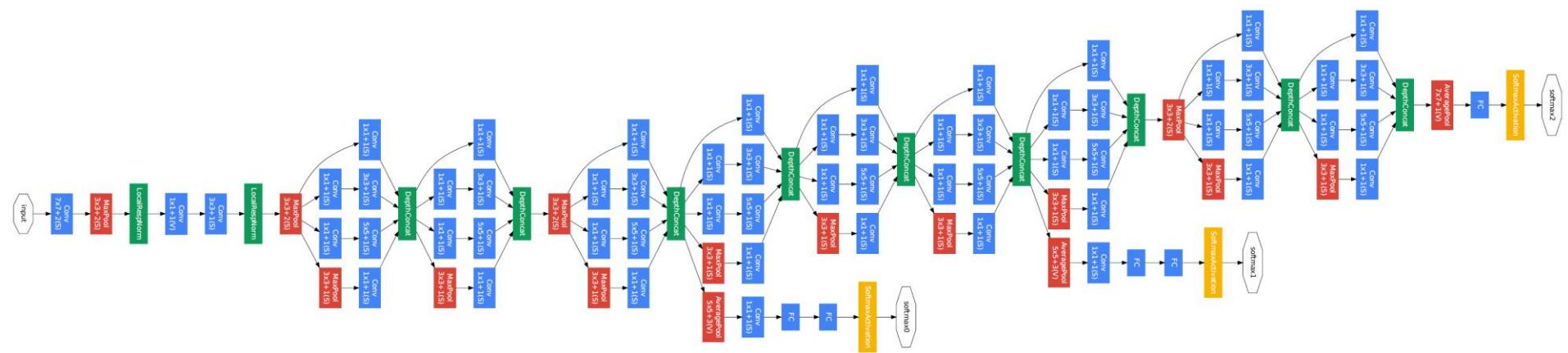
- 2012년 ILSVRC(ImageNet Large Scale Visual Recognition Challenge) 우승
 - 1000 categories / 1.2M train, 50,000 validation, and 150,000 test images
- Top-5 error: 15.4% (Other CV-based methods > 25%)
- ReLU, 2 GPU, 227x227 input, 61M parameters, ...



컨볼루션 신경망의 예

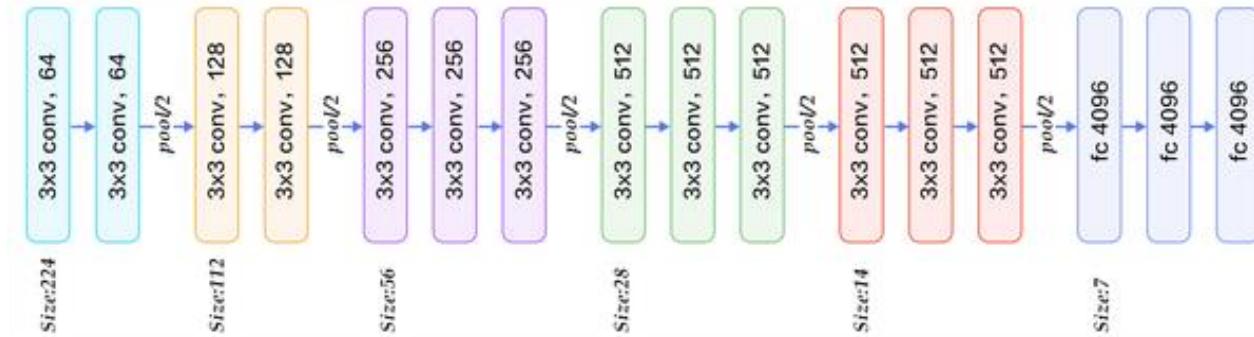
□ GoogLeNet

- 2014년 ILSVRC 영상 인식 분야에서 1위
 - 1000개 카테고리, 120만개 훈련 영상, 15만개의 테스트 영상
- Top-5 Error: 6.7% (사람: 5.1%) / 총 22개의 레이어로 구성
- 입력: 224x224, BGR, mean = (104, 117, 123)
- 출력: 1x1000, 실수형 행렬

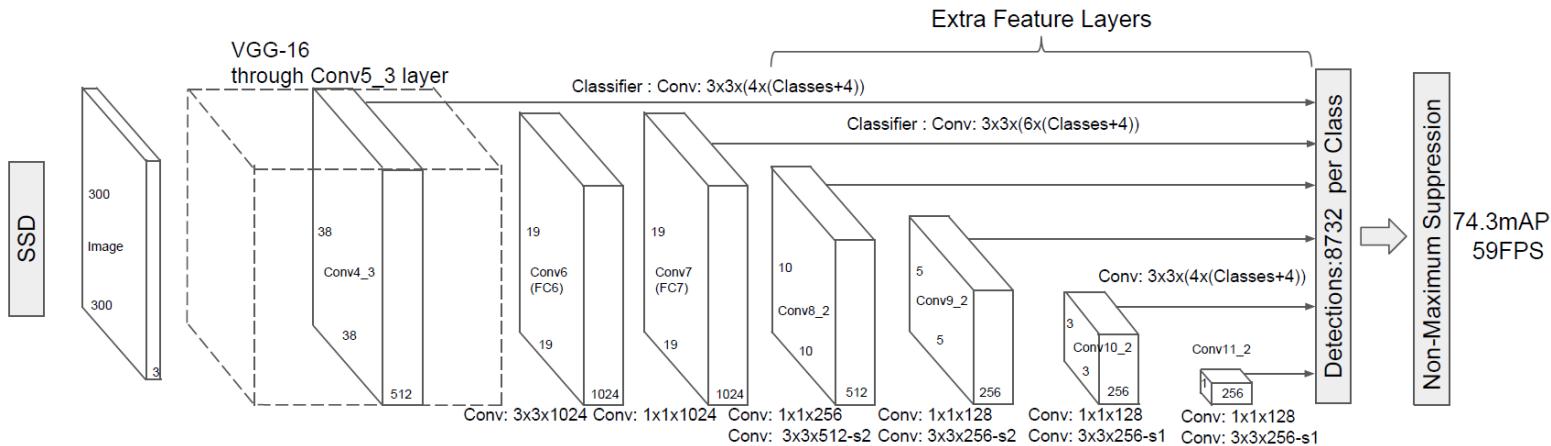


컨볼루션 신경망의 예

- VGG16 (Simonyan and Zisserman, 2014)



- 실시간 객체 검출을 위한 SSD(Single Shot Detector) (W. Liu, et. al, 2016)



OpenCV DNN

□ Tested networks

[Image classification]

- [AlexNet](#)
- [GoogLeNet](#)
- [VGG](#)
- [ResNet](#)
- [SqueezeNet](#)
- [DenseNet](#)
- [ShuffleNet](#)
- [Inception](#)
- [MobileNet](#)
- [Darknet](#)

[Object detection]

- [SSD VGG](#)
- [MobileNet-SSD](#)
- [Faster-RCNN](#)
- [R-FCN](#)
- [OpenCV face detector](#)
- [Mask-RCNN](#)
- [EAST](#)
- [YOLOv2, tiny YOLO, YOLOv3](#)

[Semantic segmentation]

- [FCN](#)
- [ENet](#)

[Pose estimation]

- [OpenPose](#)

[Image processing]

- [Colorization](#)
- [Fast-Neural-Style](#)

[Person identification]

- [OpenFace](#)

<https://github.com/opencv/opencv/wiki/Deep-Learning-in-OpenCV>

OpenCV DNN

□ Net 클래스 정의

```
<opencv-src>\modules\dnn\include\opencv2\dnn\dnn.hpp
```

```
class Net
{
public:
    Net();
    ~Net();

    bool empty() const;

    Mat forward(const String& outputName = String());
    void setInput(InputArray blob, const String& name = "",
                  double scalefactor = 1.0, const Scalar& mean = Scalar());

    void setPreferableBackend(int backendId);
    void setPreferableTarget(int targetId);

    int64 getPerfProfile(std::vector<double>& timings);
    ...
};
```

OpenCV DNN API

□ 네트워크 불러오기

```
cv2.dnn.readNet(model, config=None, framework=None) -> retval
```

- model: 훈련된 가중치를 저장하고 있는 이진 파일 이름
- config: 네트워크 구성을 저장하고 있는 텍스트 파일 이름
- framework: 명시적인 딥러닝 프레임워크 이름
- retval: cv2.dnn_Net 클래스 객체

딥러닝 프레임워크	model 파일 확장자	config 파일 확장자	framework 문자열
카페	*.caffemodel	*.prototxt	“caffe”
텐서플로우	*.pb	*.pbtxt	“tensorflow”
토치	*.t7 또는 *.net		“torch”
다크넷	*.weights	*.cfg	“darknet”
DLDT	*.bin	*.xml	“dldt”
ONNX	*.onnx		“onnx”

OpenCV DNN API

□ 네트워크 입력 블롭(blob) 만들기

```
cv2.dnn.blobFromImage(image, scalefactor=None, size=None,  
                      mean=None, swapRB=None, crop=None,  
                      ddepth=None) -> retval
```

- image: 입력 영상
- scalefactor: 입력 영상 픽셀 값에 곱할 값. 기본값은 1.
- size: 출력 영상의 크기. 기본값은 (0, 0).
- mean: 입력 영상 각 채널에서 뺄 평균 값. 기본값은 (0, 0, 0, 0).
- swapRB: R과 B 채널을 서로 바꿀 것인지를 결정하는 플래그.
기본값은 False
- crop: 크롭(crop) 수행 여부. 기본값은 False.
- ddepth: 출력 블롭의 깊이. CV_32F 또는 CV_8U. 기본값은 CV_32F.
- retval: 영상으로부터 구한 블롭 객체. [numpy.ndarray](#).
[shape=\(N,C,H,W\)](#), [dtype=float32](#)

OpenCV DNN API

□ 네트워크 입력 설정하기

```
cv2.dnn_Net.setInput(blob, name=None, scalefactor=None,  
                      mean=None) -> None
```

- blob: 블롭 객체
- name: 입력 레이어 이름
- scalefactor: 추가적으로 픽셀 값에 곱할 값
- mean: 추가적으로 픽셀 값에서 뺄 평균 값

OpenCV DNN API

□ 네트워크 순방향 실행 (추론)

```
cv2.dnn_Net.forward(outputName=None) -> retval
```

- **outputName:** 출력 레이어 이름
- **retval:** 지정한 레이어의 출력 블롭. 네트워크마다 다르게 결정됨.

GoogLeNet 영상 인식

- 미리 학습된 GoogLeNet 학습 모델 및 구성 파일 다운로드
 - 모델 파일:
 - http://dl.caffe.berkeleyvision.org/bvlc_googlenet.caffemodel
 - 구성 파일:
 - https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet에서 deploy.prototxt 파일 다운로드
 - 클래스 이름 파일:
 - 1~1000번 클래스에 대한 설명을 저장한 텍스트 파일
 - https://github.com/opencv/opencv/blob/4.1.0/samples/data/dnn/classification_classes_ILSVRC2012.txt
- 입출력 형식
 - 입력: 224x224, BGR, mean=(104, 117, 123)
 - 출력: 1000개의 노드

GoogLeNet 영상 인식

```
import sys
import numpy as np
import cv2

filename = 'space_shuttle.jpg'

if len(sys.argv) > 1:
    filename = sys.argv[1]

img = cv2.imread(filename)

# Load network

net = cv2.dnn.readNet('bvlc_googlenet.caffemodel', 'deploy.prototxt')

# Load class names

classNames = None
with open('classification_classes_ILSVRC2012.txt', 'rt') as f:
    classNames = f.read().rstrip('\n').split('\n')
```

명령행 인자 지원.

GoogLeNet 영상 인식

```
# Inference
```

```
inputBlob = cv2.dnn.blobFromImage(img, 1, (224, 224), (104, 117, 123))
net.setInput(inputBlob, 'data')
prob = net.forward()
```

```
# Check results & Display
```

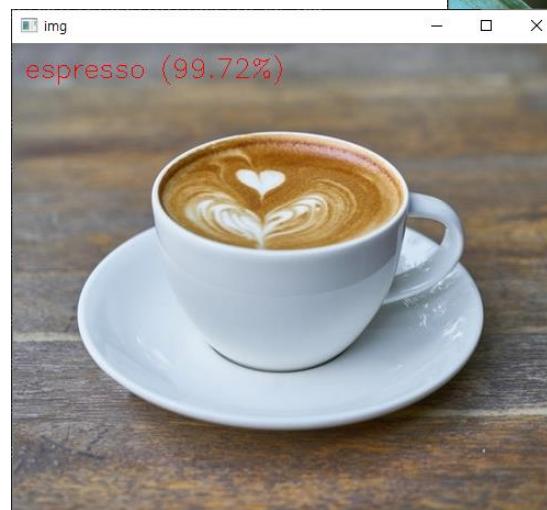
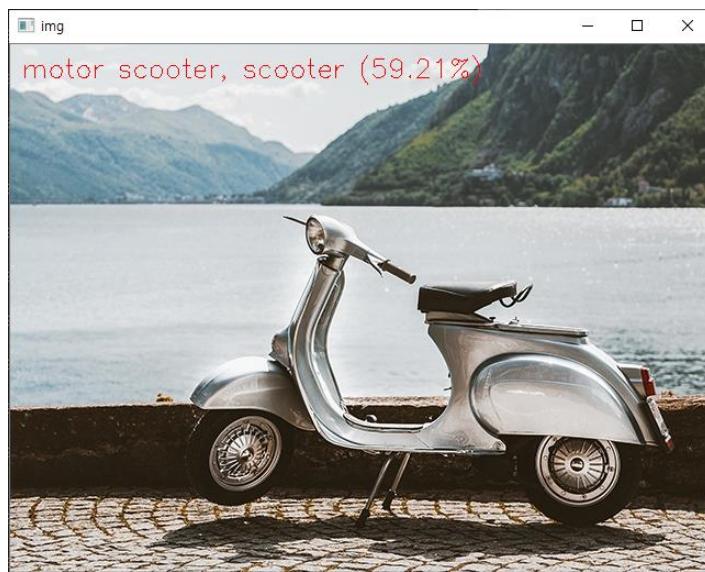
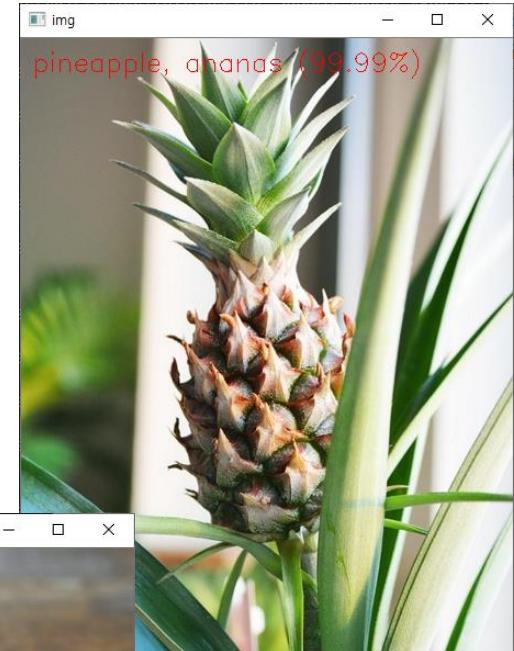
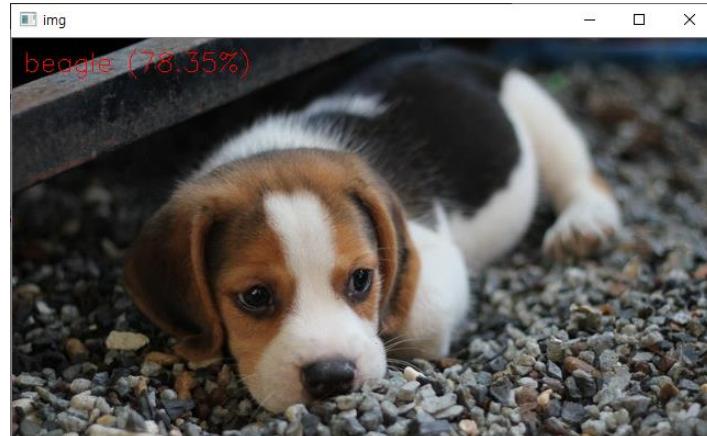
구글넷 출력 행렬 prob은 numpy.ndarray이고,
shape=(1, 1000), dtype=float32.

```
out = prob.flatten()
classId = np.argmax(out)
confidence = out[classId]
```

```
text = '%s (%4.2f%)' % (classNames[classId], confidence * 100)
cv2.putText(img, text, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255),
           1, cv2.LINE_AA)
```

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

GoogLeNet 영상 인식



OpenCV와 얼굴 검출

□ 모델 & 구성 파일 다운로드

- https://github.com/opencv/opencv/tree/master/samples/dnn/face_detector에서 파일 다운로드 후 아래 명령 실행

```
> python download_weights.py
```

□ 입출력 형식

- 입력: 300x300, BGR, mean=(104, 177, 123)
- 출력: class, confidence, coordinate 등의 정보를 담고 있는 4차원 행렬
shape=(1, 1, N, 7)

0	1	c	x1	y1	x2	y2
0	1	c	x1	y1	x2	y2
0	1	c	x1	y1	x2	y2
:	:	:	:	:	:	:

OpenCV SSD 얼굴 검출

```

import cv2

model = 'res10_300x300_ssd_iter_140000_fp16.caffemodel'
config = 'deploy.prototxt'

cap = cv2.VideoCapture(0)

net = cv2.dnn.readNet(model, config)

while True:
    _, frame = cap.read()
    if frame is None:
        break

    blob = cv2.dnn.blobFromImage(frame, 1, (300, 300), (104, 177, 123))
    net.setInput(blob)

    detect = net.forward()
    detect = detect[0, 0, :, :]

```

detect = net.forward()
detect = detect[0, 0, :, :]

detect.shape=(1, 1, N, 7)이고 이중 뒤쪽
두 개의 차원에 검출 정보가 저장됨.
그러므로 편의상 2차원 행렬로 변환하여 사용.

OpenCV SSD 얼굴 검출

```
(h, w) = frame.shape[:2]

for i in range(detect.shape[0]):
    confidence = detect[i, 2]
    if confidence < 0.5:
        break

    x1 = int(detect[i, 3] * w)
    y1 = int(detect[i, 4] * h)
    x2 = int(detect[i, 5] * w)
    y2 = int(detect[i, 6] * h)

    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0))
    label = 'Face: %4.3f' % confidence
    cv2.putText(frame, label, (x1, y1 - 1), cv2.FONT_HERSHEY_SIMPLEX,
                0.8, (0, 255, 0), 1, cv2.LINE_AA)

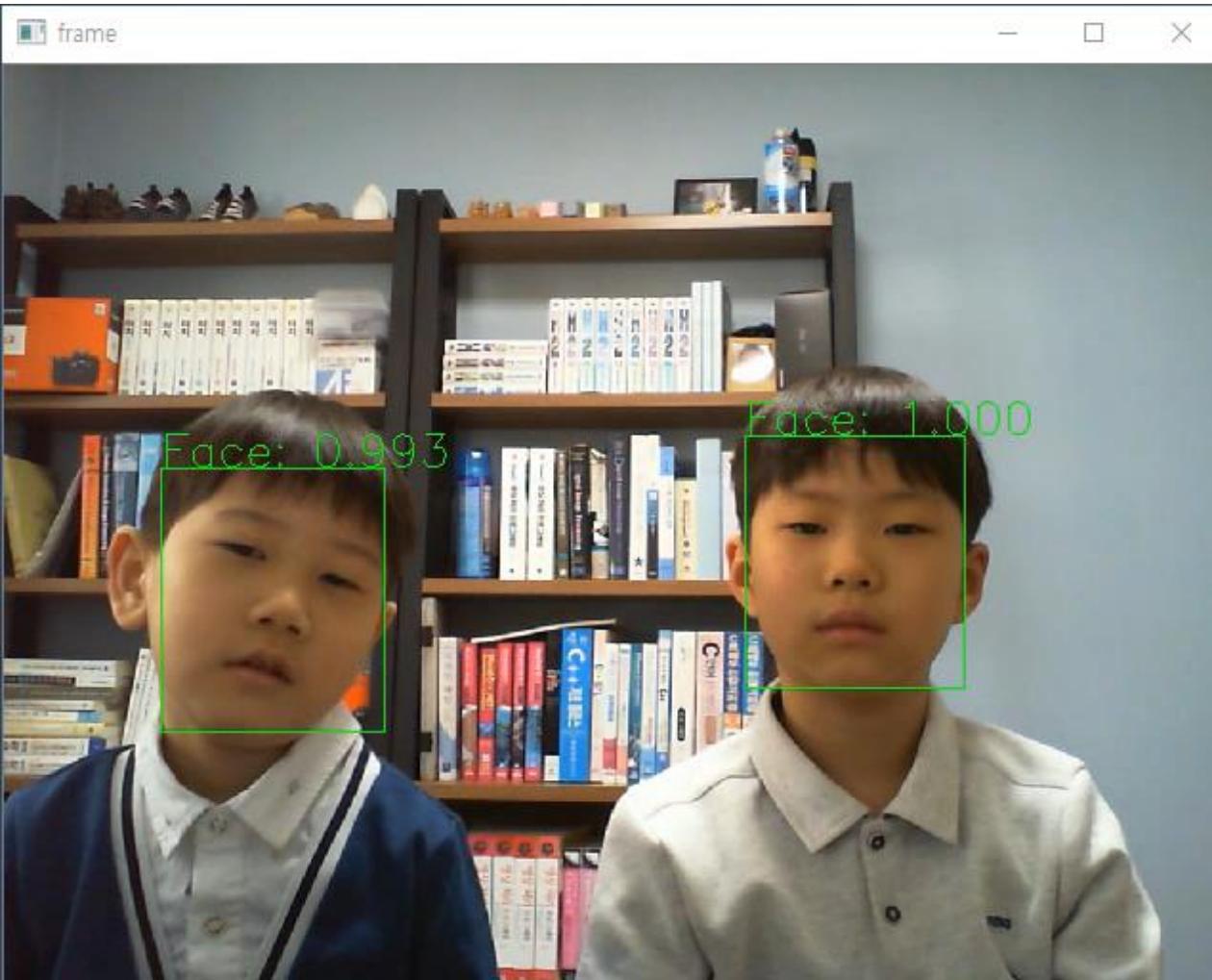
cv2.imshow('frame', frame)
if cv2.waitKey(1) == 27:
    break

cv2.destroyAllWindows()
```

detect:

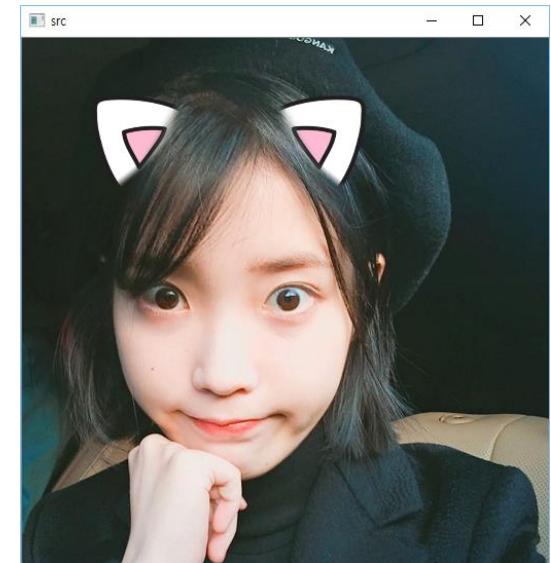
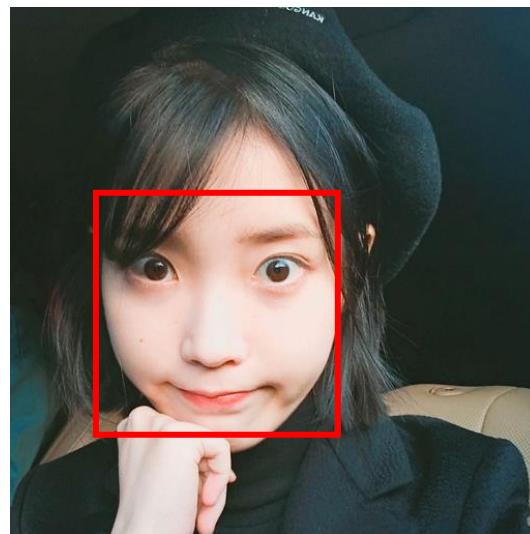
0	1	c	x1	y1	x2	y2
0	1	c	x1	y1	x2	y2
0	1	c	x1	y1	x2	y2
:	:	:	:	:	:	:

OpenCV SSD 얼굴 검출



얼굴 스티커

- 얼굴을 검출한 후 재미있는 그래픽을 합성
 - 투명한 PNG 파일 불러오기 → 4채널 (b, g, r, alpha)
 - 카메라 입력에서 얼굴 검출 → 얼굴 위치 & 크기 정보 추출
 - 두 장의 영상 합성



얼굴 스티커

```
import sys
import numpy as np
import cv2

model = 'opencv_face_detector_uint8.pb'
config = 'opencv_face_detector.pbtxt'

cap = cv2.VideoCapture(0)
net = cv2.dnn.readNet(model, config)
cat = cv2.imread('cat.png', cv2.IMREAD_UNCHANGED)

while True:
    ret, frame = cap.read()

    blob = cv2.dnn.blobFromImage(frame, 1, (300, 300), (104, 177, 123))
    net.setInput(blob)
    detect = net.forward()

    (h, w) = frame.shape[:2]
    detect = detect[0, 0, :, :]
```

얼굴 스티커

```
for i in range(detect.shape[0]):
    confidence = detect[i, 2]
    if confidence < 0.5:
        break

    x1 = int(detect[i, 3] * w + 0.5)
    y1 = int(detect[i, 4] * h + 0.5)
    x2 = int(detect[i, 5] * w + 0.5)
    y2 = int(detect[i, 6] * h + 0.5)

    fx = (x2 - x1) / cat.shape[1]
    cat2 = cv2.resize(cat, (0, 0), fx=fx, fy=fy)
    pos = (x1, y1 - (y2 - y1) // 4)

    overlay(frame, cat2, pos)

cv2.imshow('frame', frame)

if cv2.waitKey(1) == 27:
    break

cv2.destroyAllWindows()
```

얼굴 스티커

```
def overlay(frame, cat, pos):
    if pos[0] < 0 or pos[1] < 0:
        return

    if pos[0] + cat.shape[1] > frame.shape[1] or pos[1] + cat.shape[0] > frame.shape[0]:
        return

    sx = pos[0]
    ex = pos[0] + cat.shape[1]
    sy = pos[1]
    ey = pos[1] + cat.shape[0]

    img1 = frame[sy:ey, sx:ex] # shape=(h, w, 3)
    img2 = cat[:, :, 0:3]       # shape=(h, w, 3)
    alpha = 1. - (cat[:, :, 3] / 255.) # shape=(h, w)

    img1[:, :, 0] = (img1[:, :, 0] * alpha + img2[:, :, 0] * (1. - alpha)).astype(np.uint8)
    img1[:, :, 1] = (img1[:, :, 1] * alpha + img2[:, :, 1] * (1. - alpha)).astype(np.uint8)
    img1[:, :, 2] = (img1[:, :, 2] * alpha + img2[:, :, 2] * (1. - alpha)).astype(np.uint8)
```



break