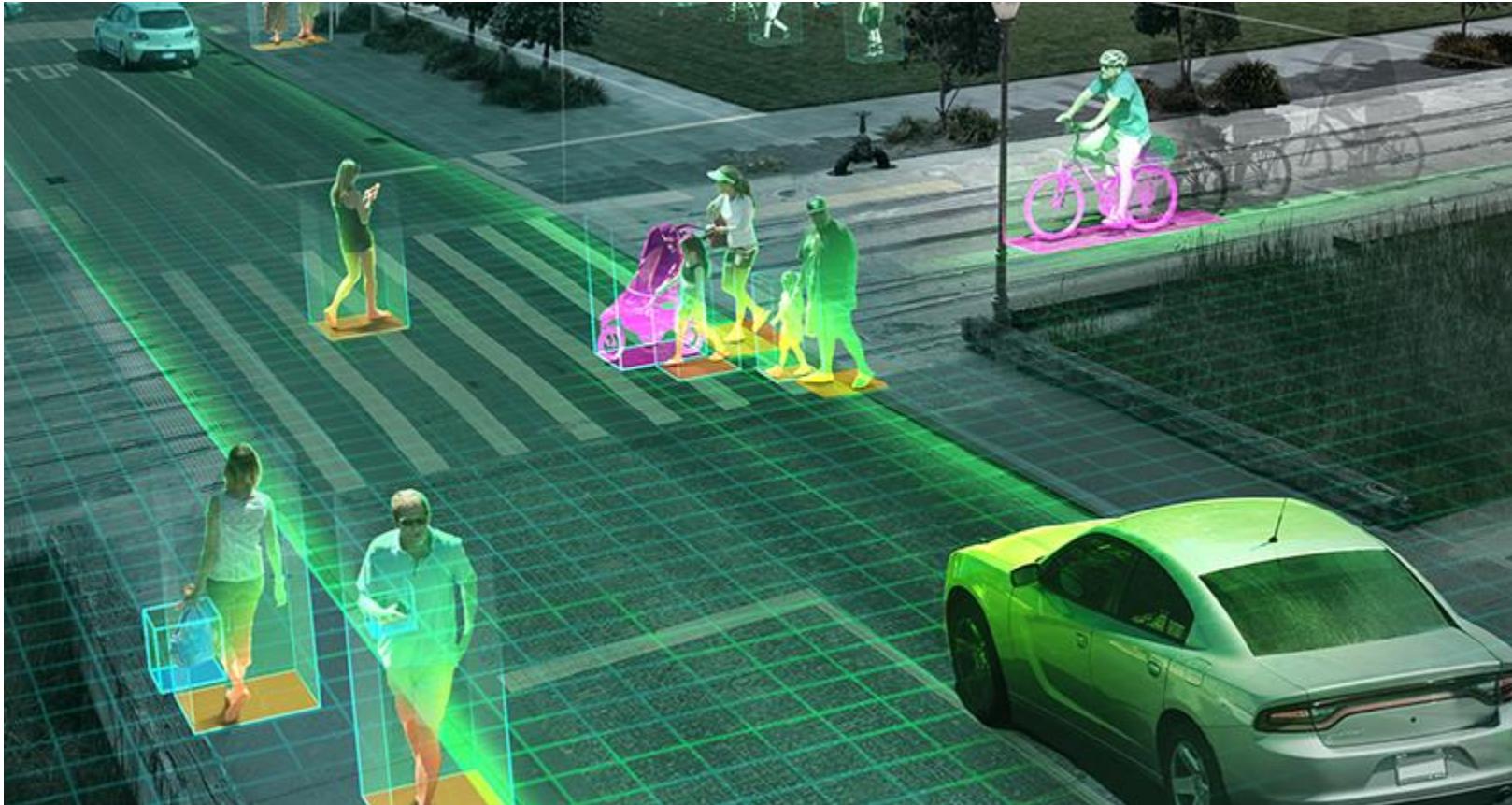


OpenCV

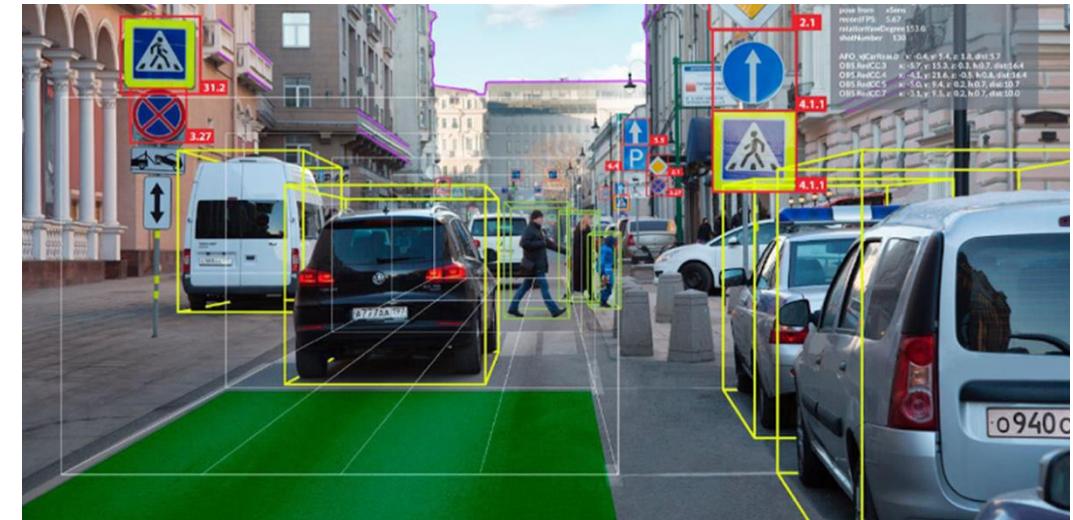
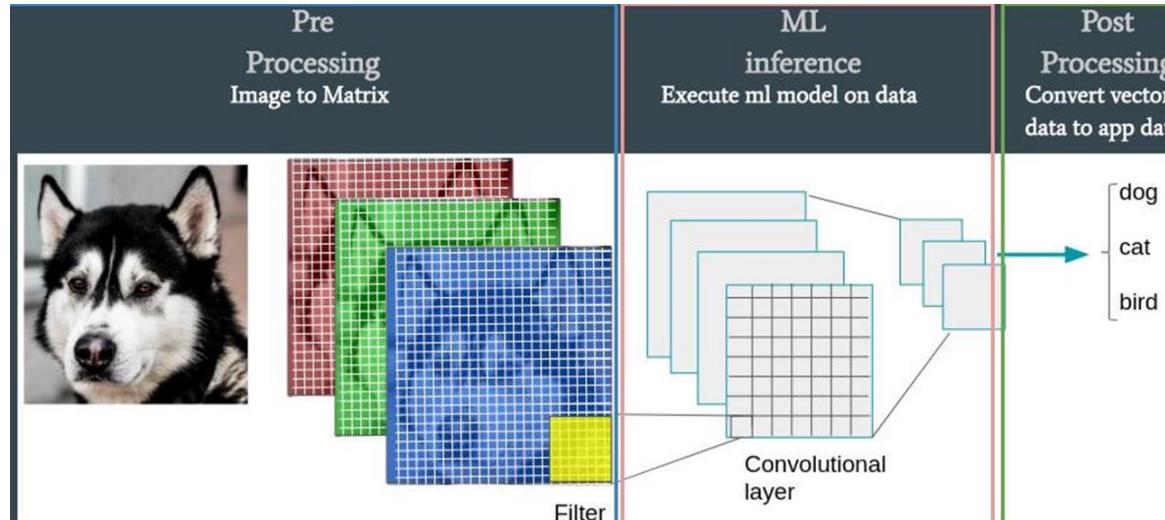


박경규

컴퓨터 비전

Computer vision 은 인공지능의 한 분야로서, 이미지 또는 영상에서 특징(Feature, 의미있는 정보)들을 추출하는 방법을 연구하는 분야입니다.

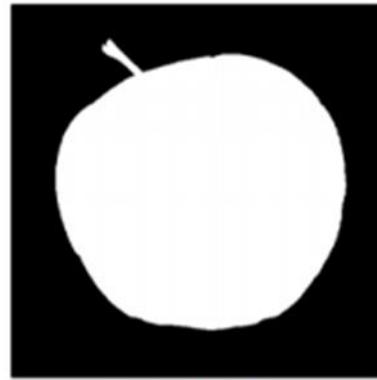
- 이미지 분류, 객체 검출, 영상 분할, 화질 개선
- 연속 영상에서 물체를 추적
- 어떤 장면을 3차원 모델로 맵핑
- 인간의 자세와 팔다리 움직임을 3차원으로 추정
- 콘텐츠 기반 이미지 검색



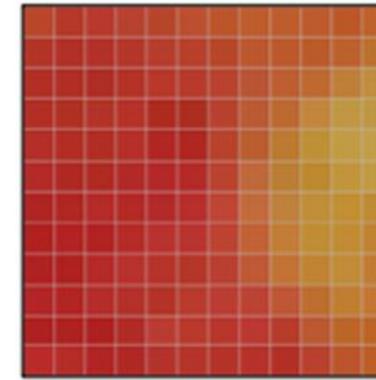
컴퓨터 비전



사과?



둥글다?



빨간색이다?



꼭지 모양?



사과가 몇 개??

이미지 데이터

컴퓨터에서 이미지 데이터는 숫자로 관리하며, 이미지를 구성하는 가장 기본 단위는 픽셀(pixel)입니다.
컬러 이미지는 Red, Green, Blue(RGB)의 3색으로 표현하고, 농도를 0~255의 수치로 나타냅니다.
RGB 이미지는 채널수가 3이고, 흑백이미지는 채널수가 1입니다.

003366 R - 000 G - 051 B - 102	336699 R - 051 G - 102 B - 153	6699CC R - 102 G - 153 B - 204	99CCFF R - 153 G - 204 B - 255	CCFF00 R - 204 G - 255 B - 000	FF0033 R - 255 G - 000 B - 051
003399 R - 000 G - 051 B - 153	3366CC R - 051 G - 102 B - 204	6699FF R - 102 G - 153 B - 255	99CC00 R - 153 G - 204 B - 000	CCFF33 R - 204 G - 255 B - 051	FF0066 R - 255 G - 000 B - 102
0033CC R - 000 G - 051 B - 204	3366FF R - 051 G - 102 B - 255	669900 R - 102 G - 153 B - 000	99CC33 R - 153 G - 204 B - 051	CCFF66 R - 204 G - 255 B - 102	FF0099 R - 255 G - 000 B - 153

이미지 포맷 종류 및 특징

포맷	압축률	특징	단점
BMP	무압축	압축하지 않는 저장 방식으로 원본 이미지의 보존률이 높음	압축하지 않기 때문에 용량이 큼
GIF	상	이미지 손실이 적으면서 높은 압축률 애니메이션(다중 프레임) 저장 가능	최대 해상도 65,536 x 65,536 에 256 컬러라는 제한이 있음
JPG	상	높은 압축률로 용량이 작아 현재 가장 널리 사용됨 저장 시 품질(압축률)을 설정 가능	손실 압축방식으로 압축률을 높일수록 이미지 손상 증가
PNG	상	JPG에 비해 적은 손실에 상대적으로 높은 압축률 투명한 부분을 보존하여 저장 가능	JPG보다 손실은 적지만 압축률은 약간 낮음
TIF	무압축 또는 낮은 압축률	입출력 속도와 전송률이 높음	용량이 크고 하위 버전 호환이 되지 않음
PSD	상	포토샵 전용 포맷	포토샵 툴에서만 사용할 수 있는 전용 포맷

OpenCV

실시간 이미지 프로세싱에 중점을 둔 오픈 소스 컴퓨터 비전 라이브러리입니다.
딥러닝 관련 연구가 파이썬으로 진행되면서 파이썬 랩핑 라이브러리를 주로 사용하고 있는 추세입니다.

<https://opencv.org/>



■ 주요 기능

- 이진화(binarization)
- 노이즈 제거
- 외곽선 검출(edge detection)
- 패턴인식
- 기계학습(machine learning)
- ROI(Region Of Interest) 설정
- 이미지 변환(image warping)
- 하드웨어 가속

OpenCV 기초

OpenCV 설치 : pip install opencv-python

이미지파일 업로드 : car.jpg, mask.png, starwars.png

```
[1] from google.colab import files  
upload_file = files.upload()
```

라이브러리 임포트

```
[2] import numpy as np  
import cv2
```

```
[3] print("OpenCV version: ", cv2.__version__)
```



OpenCV 기초

```
[4] img = cv2.imread("car.jpg")
```

```
# 이미지 shape 확인 : (높이, 넓이, 채널수)
print(f"height: {img.shape} pixels")
print(f"height: {img.shape[0]} pixels")
print(f"width: {img.shape[1]} pixels")
print(f"channels: {img.shape[2]}")
```

height: (549, 976, 3) pixels

height: 549 pixels

width: 976 pixels

channels: 3

```
[5] X, Y = 100, 200
```

```
(b,g,r) = img[100, 100]
```

```
print(f"Pixel at ({X}, {Y}) - Red: {r}, Green: {g}, Blue: {b}")
```

Pixel at (100, 200) - Red: 185, Green: 178, Blue: 162

OpenCV 기초

이미지 출력 : cv2.imshow() 함수는 Colab에서 동작 안함

```
[6] # cv2.imshow('My Car', img)
```

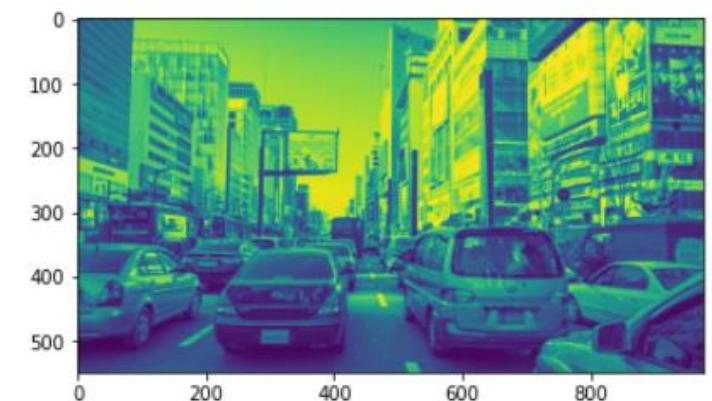
이미지 출력 : Matplotlib

- cv2에서는 이미지를 RGB 순서가 아닌 BGR 순서로 처리합니다.
- cv2.cvtColor(img, cv2.COLOR_BGR2RGB) 함수로 색상 위치를 변경합니다.

```
[7] import matplotlib.pyplot as plt

rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

img = cv2.imread("car.jpg")
plt.imshow(img)
plt.show()
plt.imshow(rgb_img)
plt.show()
plt.imshow(gray)
plt.show()
```



OpenCV 기초

이미지 출력 : PIL 사용

```
[8] import PIL.Image as Image  
  
img = cv2.imread("car.jpg")  
rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  
Image.fromarray(rgb_img)
```

이미지 출력 : Colab cv2_imshow() 사용

```
[9] from google.colab.patches import cv2_imshow  
  
img = cv2.imread("car.jpg")  
cv2_imshow(img)
```



OpenCV 기초

이미지 생성 및 저장

cv2에서는 이미지를 RGB 순서가 아닌 BGR 순서로 처리합니다.

```
[10] # 이미지의 크기를 결정합니다  
img_size = (200, 400)  
  
# 이미지 정보를 가지는 행렬을 만듭니다  
# 파란색 이미지이므로, 각 요소가 [255, 0, 0]인 200x400의 행렬을 만듭니다  
new_img = np.array([[[255, 0, 0] for _ in range(img_size[1])] for _ in range(img_size[0])], dtype="uint8")  
  
# 이미지를 표시합니다  
cv2_imshow(new_img)  
  
# 이미지를 파일로 저장합니다  
cv2.imwrite("new_img.jpg", new_img)
```

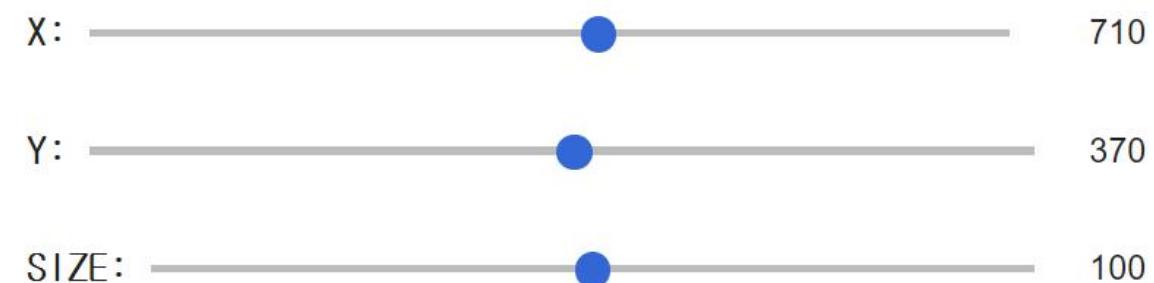


OpenCV 기초

이미지 트리밍

이미지를 나타내는 행렬의 일부를 꺼내면 트리밍이 됩니다.

```
[11] X = 710 #@param {type:"slider", min:0, max:1280, step:1}  
Y = 370 #@param {type:"slider", min:0, max:720, step:1}  
SIZE = 100 #@param {type:"slider", min:0, max:200, step:1}
```



```
[12] img = cv2.imread("car.jpg")  
  
# Crop cordination = image[y: y+h, x:x+w]  
cropped = img[Y:Y+SIZE, X:X+SIZE]  
cv2_imshow(cropped)
```



OpenCV 기초

이미지 리사이즈

크기를 지정할 때는 (폭, 높이)의 순서

```
[13] img = cv2.imread("car.jpg")  
  
my_img = cv2.resize(img, (img.shape[1] * 3, img.shape[0] * 2))  
cv2.imshow(my_img)
```

OpenCV 기초

이미지 회전 : warpAffine()

- 첫번째 인수 : 회전 중심
- 두번째 인수 : 회전 각도
- 세번째 인수 : 배율

```
[14] img = cv2.imread("car.jpg")
    mat = cv2.getRotationMatrix2D(tuple(np.array(img.shape[:2]) / 2), 180, 2.0)
    my_img = cv2.warpAffine(img, mat, img.shape[:2])
```

색상(색공간) 변환

```
[15] img = cv2.imread("car.jpg")
    my_img = cv2.cvtColor(img, cv2.COLOR_RGB2LAB)

    cv2.imshow(my_img)
```

OpenCV 기초

임계값 처리(이진화)

이미지 용량을 줄이기 위해 일정값 이상을 모두 같은 값으로 처리

- 첫번째 인수 : 처리하는 이미지
- 두번째 인수 : 임계값
- 세번째 인수 : 최대값(maxvalue)
- 네번째 인수 : THRESH_BINARY, THRESH_BINARY_INV, THRESH_TOZERO, THRESH_TRUNC, THRESH_TOZERO_INV 중 선택
- THRESH_BINARY: 픽셀값이 임계값을 초과하는 경우 해당 픽셀을 maxValue로 하고, 그 이외의 경우 0(검은색)
- THRESH_BINARY_INV: 픽셀값이 임계값을 초과하는 경우 0으로 설정하고, 그 이외의 경우 maxValue
- THRESH_TRUNC: 픽셀값이 임계값을 초과하는 경우 임계값으로 설정하고, 그 이외의 픽셀은 변경하지 않음
- THRESH_TOZERO: 픽셀값이 임계값을 초과하는 경우 변경하지 않고, 그 이외의 경우 0으로 설정
- THRESH_TOZERO_INV: 픽셀값이 임계값을 초과하는 경우 0으로 설정하고, 그 이외의 경우 변경하지 않음

```
[16] img = cv2.imread("car.jpg")
    retval, my_img = cv2.threshold(img, 75, 255, cv2.THRESH_TOZERO)

    cv2.imshow(my_img)
```

OpenCV 기초

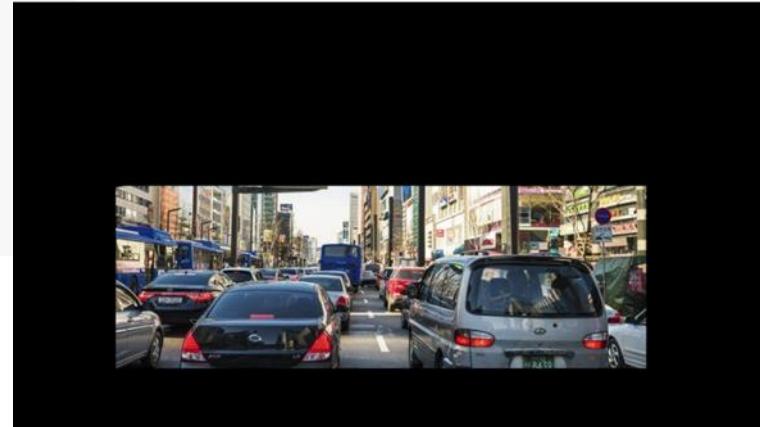
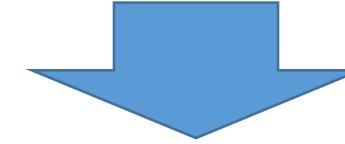
이미지 마스킹

```
[17] img = cv2.imread("car.jpg")
mask = cv2.imread("mask.png", 0)

# 마스크 이미지를 이미지와 같은 크기로 리사이즈
mask = cv2.resize(mask, (img.shape[1], img.shape[0]))

my_img = cv2.bitwise_and(img, img, mask = mask)

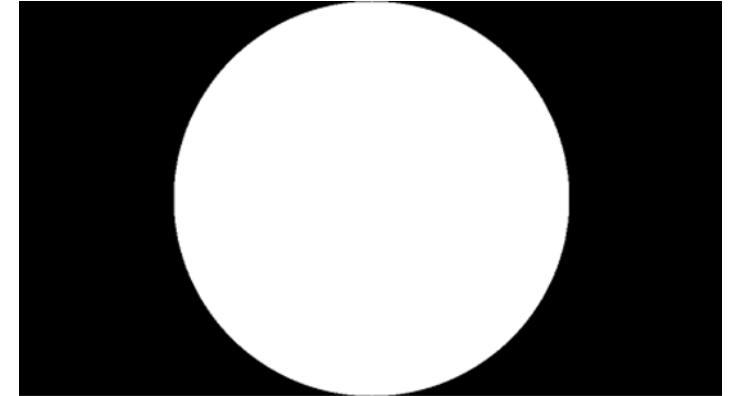
cv2.imshow(my_img)
```



OpenCV 기초

```
[18] (height, width) = img.shape[:2]
    center = (width // 2, height // 2)

    mask = np.zeros(img.shape[:2], dtype='uint8')
    cv2.circle(mask, center, int(height/2), (255, 255, 255), -1)
    cv2_imshow(mask)
```



```
[19] masked = cv2.bitwise_and(img, img, mask=mask)
    cv2_imshow(masked)
```



OpenCV 기초

흐림효과 적용

- 첫번째 인수 : 원본 이미지
- 두번째 인수 : $n \times n$ (마스크 크기)에서 n 값을 지정(n 은 홀수)
- 세번째 인수 : x축 방향의 편차(일반적으로 0 지정)

```
[20] my_img = cv2.GaussianBlur(img, (11, 11), 0)
     cv2.imshow(my_img)
```

노이즈 제거

```
[21] my_img = cv2.fastNlMeansDenoisingColored(img)
     cv2.imshow(my_img)
```

OpenCV 기초

이미지 팽창과 침식

```
[22] img = cv2.imread("starwars.png")
```

```
# 필터 정의  
filt = np.array([[0, 1, 0],  
                 [1, 0, 1],  
                 [0, 1, 0]], np.uint8)
```

```
# 팽창 처리합니다  
img_dilate = cv2.dilate(img, filt)  
img_erod = cv2.erode(img, filt)  
cv2.imshow(img)  
cv2.imshow(img_dilate)  
cv2.imshow(img_erod)
```

■ 이미지 원본



■ 이미지 팽창



■ 이미지 침식



Thank you



kgpark88@gmail.com