

OPEN CV 활용

충북대학교 산업인공지능학과 석사 과정

- 학번 : 2020254008 / 최원희
- 학번 : 2020254004 / 손의걸



OPEN CV 란?

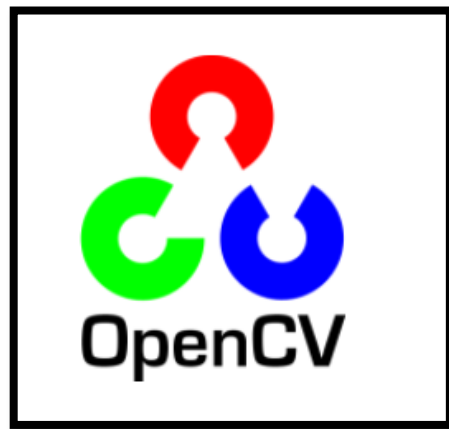
OpenCV는 **Open Source Computer Vision Library**의 약어로 오픈소스 컴퓨터 비전 라이브러리입니다.

실시간 영상 처리에 중점을 둔 영상 처리 라이브러리로서,

Apache 2.0 라이선스하에 배포되어 학술적 용도 외에도 상업적 용도로도 사용할 수 있습니다.

OpenCV는 계산 효율성과 실시간 처리에 중점을 두고 설계되었습니다.

500가지가 넘는 알고리즘이 최적화돼 있으며 이 알고리즘을 구성하거나 지원하는 함수는 알고리즘 수의 10배가 넘습니다.



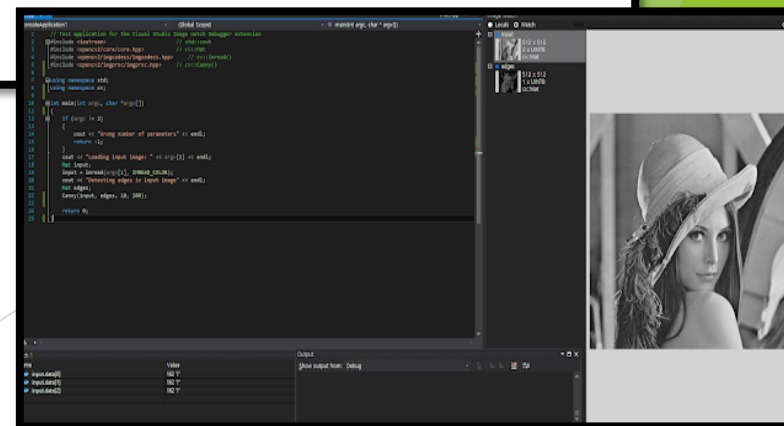
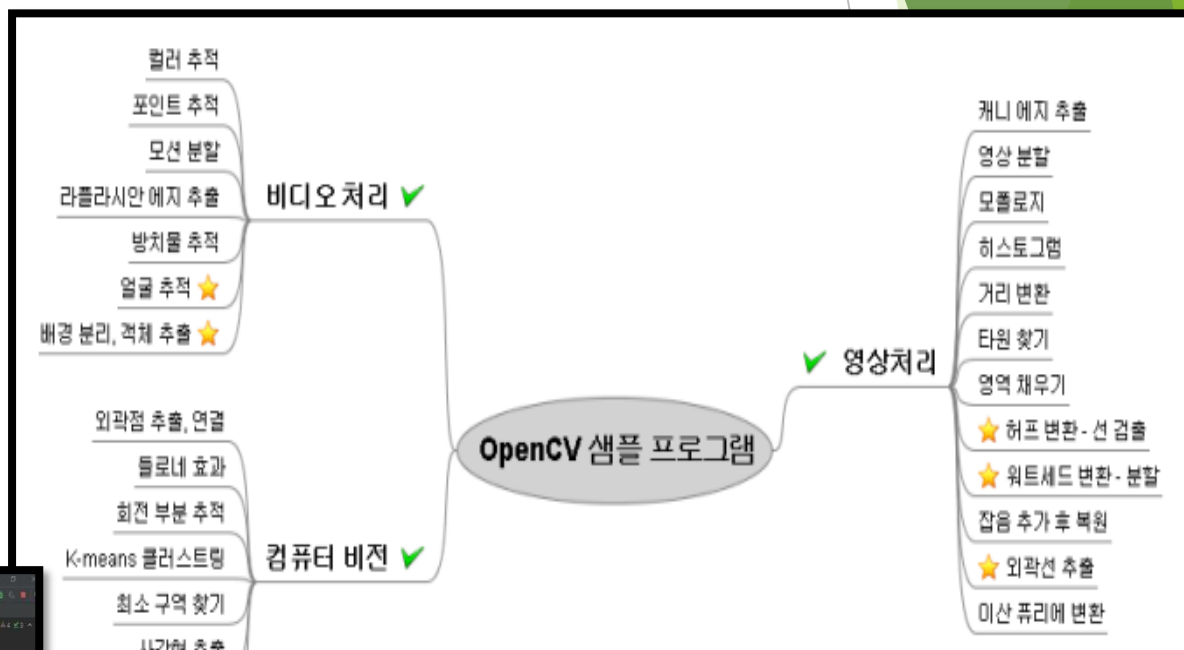
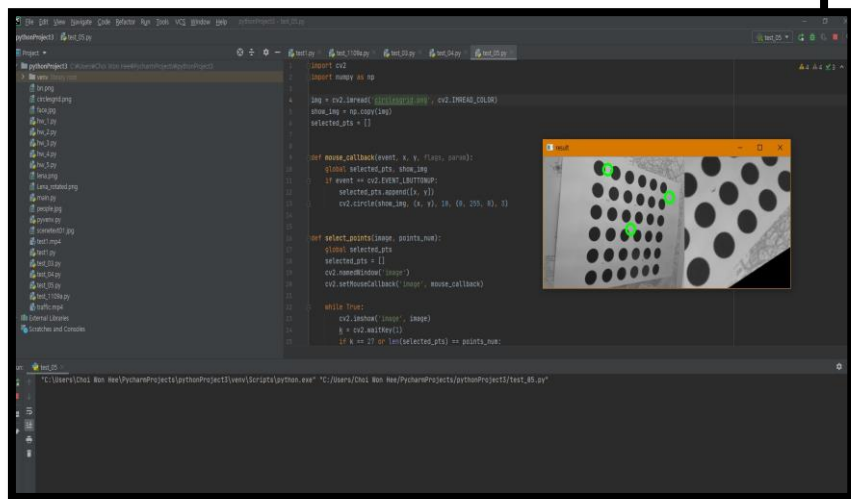
물체 인식, 얼굴 인식, 제스처 인식을 비롯해 자율주행 자동차, OCR 판독기, 불량 검사기 등에 활용할 수 있습니다.

또한 멀티 코어 프로세싱을 지원하기 때문에 다양한 상황에 응용이 가능합니다.

OPEN CV가 등장하면서 비전 전문가가 아니어도 컴퓨터 비전을 활용할 수 있게 되어 학생, 연구원, 프로그래머 등 많은 사람들이 사용하고 있습니다.

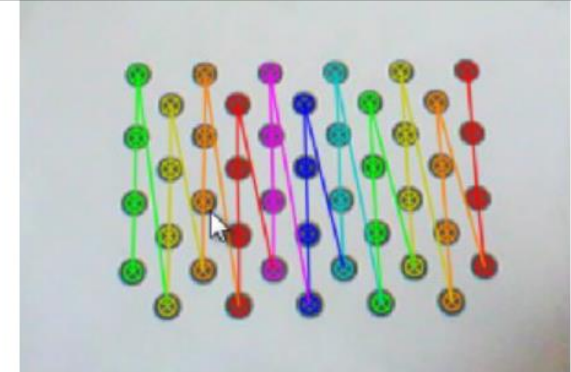
장점 및 사용처

- OpenCV를 사용하면 대단한 기능도 매우 짧게 자석이 가능하기 때문에 좋음
- 다양한 언어를 지원
- 크로스 플랫폼
- 웬만한 영상 처리 기법은 다 지원 (+수학연산)
- 머신러닝
- CUDA 같은 거 활용해서 개발 가능
- 완전 무료 사용
- HCI
- 물체인식
- 안면인식
- 로봇틱스
- 제스처 인식

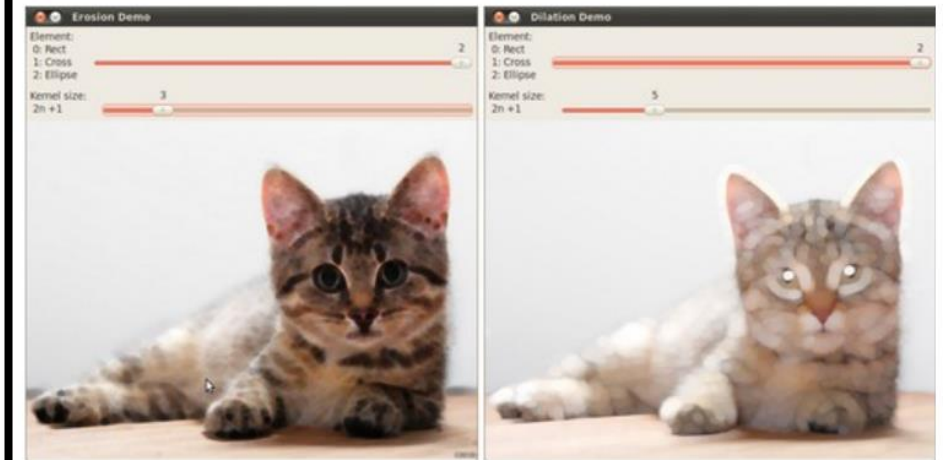


제공하는 기능 (1)

- Main modules:
 - core. Core functionality
 - imgproc. Image processing
 - imgcodecs. Image file reading and writing
 - videoio. Media I/O
 - highgui. High-level GUI
 - video. Video Analysis
 - calib3d. Camera Calibration and 3D Reconstruction
 - features2d. 2D Features Framework
 - objdetect. Object Detection
 - ml. Machine Learning
 - flann. Clustering and Search in Multi-Dimensional Spaces
 - photo. Computational Photography
 - stitching. Images stitching
 - cudaarithm. Operations on Matrices
 - cudabgsegm. Background Segmentation
 - cudacodec. Video Encoding/Decoding
 - cudafeatures2d. Feature Detection and Description
 - cudafilters. Image Filtering
 - cudaimgproc. Image Processing
 - cudalegacy. Legacy support
 - cudaobjdetect. Object Detection
 - cudaoptflow. Optical Flow
 - cudastereo. Stereo Correspondence
 - cudawarping. Image Warping
 - cudev. Device layer
 - shape. Shape Distance and Matching
 - superres. Super Resolution
 - videostab. Video Stabilization
 - viz. 3D Visualizer



이런 느낌의 일을 할 수 있다.

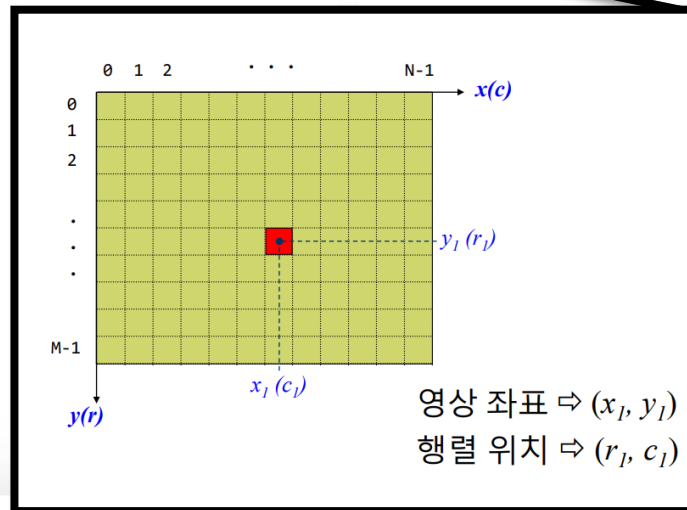


사진을 보정

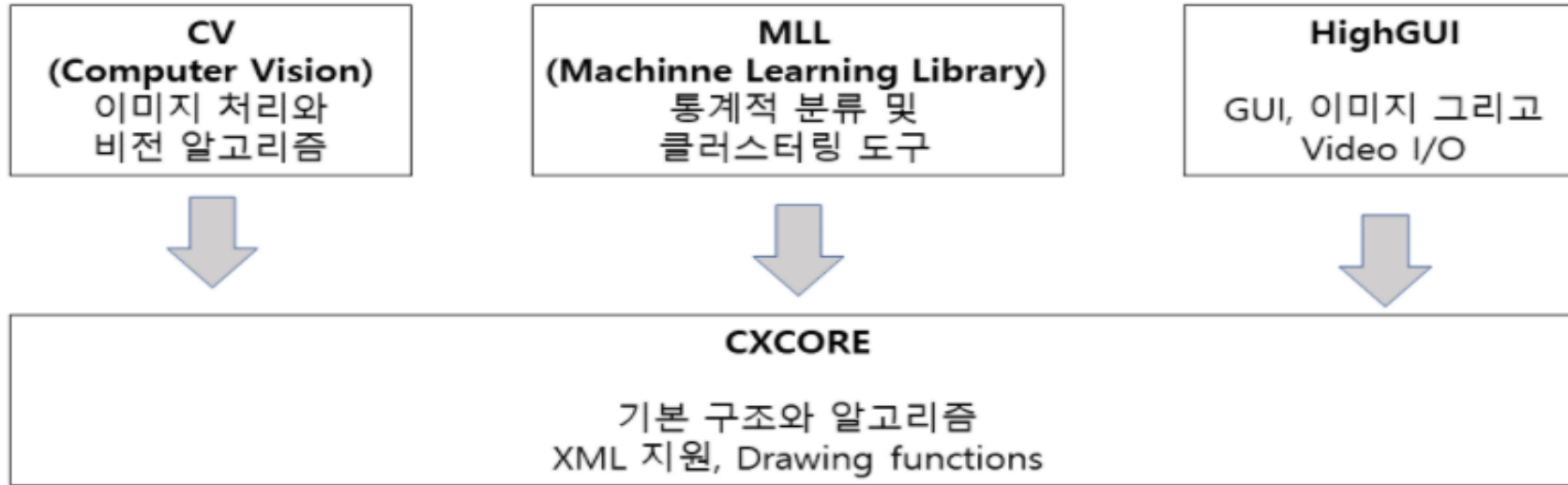
제공하는 기능 (2)

- Extra modules:

- aruco. ArUco Marker Detection
- bgsegm. Improved Background-Foreground Segmentation Methods
- bioinspired. Biologically inspired vision models and derived tools
- ccalib. Custom Calibration Pattern for 3D reconstruction
- cvv. GUI for Interactive Visual Debugging of Computer Vision Programs
- datasets. Framework for working with different datasets
- dnn. Deep Neural Network module
- dpm. Deformable Part-based Models
- face. Face Recognition
- fuzzy. Image processing based on fuzzy mathematics
- hdf. Hierarchical Data Format I/O routines
- line_descriptor. Binary descriptors for lines extracted from an image
- matlab. MATLAB Bridge
- optflow. Optical Flow Algorithms
- plot. Plot function for Mat data
- reg. Image Registration
- rgbd. RGB-Depth Processing
- saliency. Saliency API
- sfm. Structure From Motion
- stereo. Stereo Correspondance Algorithms
- structured_light. Structured Light API
- surface_matching. Surface Matching
- text. Scene Text Detection and Recognition
- tracking. Tracking API
- xfeatures2d. Extra 2D Features Framework
- ximgproc. Extended Image Processing
- xobjdetect. Extended object detection
- xphoto. Additional photo processing algorithms



기본 구조 (아키텍처)



- CV : Computer Vision으로 기본 구성 요소에서는 기본 이미지 처리와 고급 컴퓨터 비전 알고리즘을 포함
- ML (Machine Learning) : 머신 러닝이 주를 이루므로 이를 활용할 수 있는 기본적인 통계 분류기와 클러스터링 도구를 포함
- HighGUI : OpenCV에 기본으로 포함되어 있는 GUI이다. 여기에는 비디오와 이미지 저장 그리고 이를 로드하기 위한 I/O 루틴을 포함
- CXCORE : 이곳에는 기본 데이터 구조와 여러 content가 구성

설치 방법 (1)

1.openCV를 다운받고 설치

<http://sourceforge.net/projects/opencvlibrary/>

2.자신의 운영체제에 맞는 cmake다운 (windows는 zip,exe파일이 있는데 zip파일로 다운)

<http://www.cmake.org/cmake/resources/software.html>

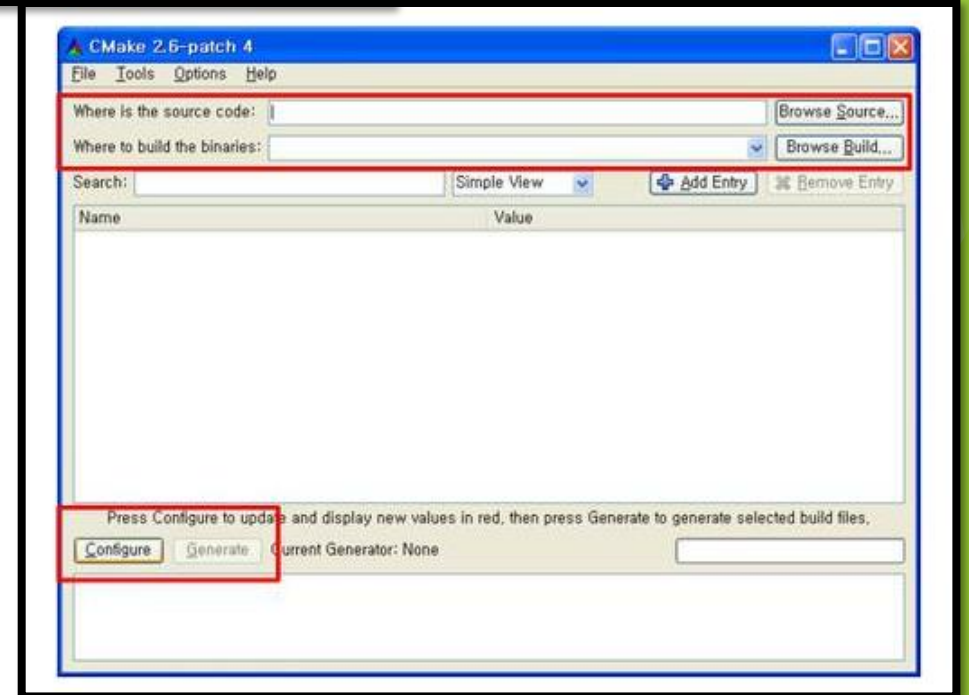
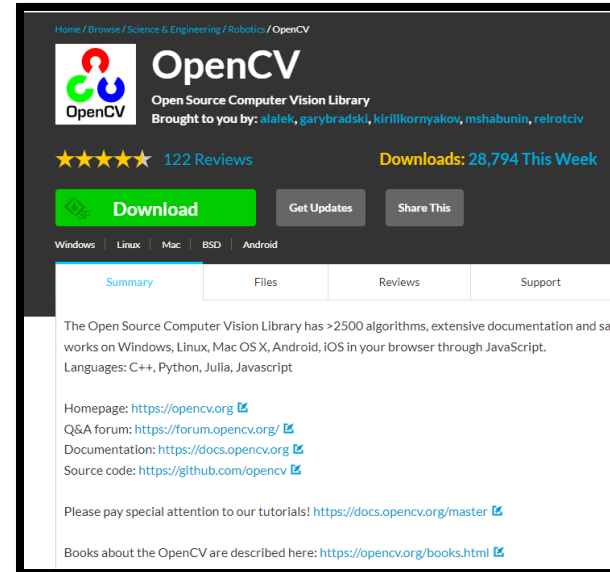
3.C드라이브에 cmake_binary_dir폴더 생성

4. 다운받은 cmake파일을 생성한 폴더에 압축풀기

5. 압축풀 폴더에 들어가보면 cmake-gui.exe 파일 실행

6. 실행해보면 그림과같은 프로그램이 실행되는데 Browse Source를 눌러 openCV를 설치한폴더로 지정한다

7.Browse Build를 눌러 아까 생성한 폴더로 지정



설치 방법 (2)

8. configure를 누르면 다른 작은 화면이 나타나는데 자신이 쓰고있는 컴파일러를 선택하고 finish를 눌러준다. 그 밑의 화면에서 어떤동작이 실행되고 가운데의 화면에 파일이 나타나는데 그때 파일마다 빨갈게 음영이 되었을 것이다.. 그러면 다시 configure버튼을 눌러주면 빨간색음영이 사라진다. 그리고 그옆의 Generate를 누르면 완료!

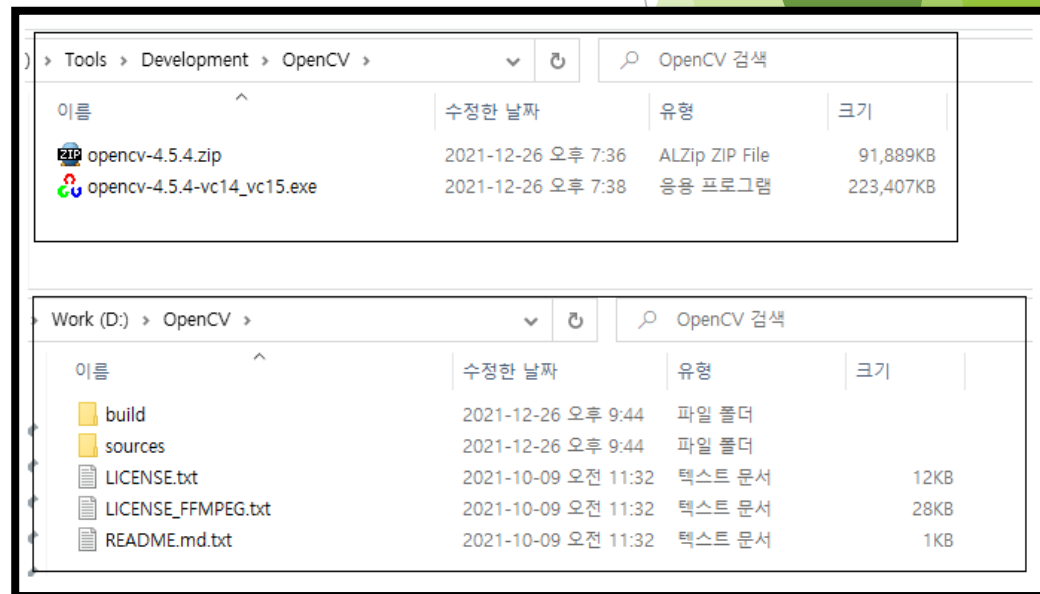
9. 아까 만든 폴더에 들어가면 openCV라는 파일이있는데 Visual Studio2005기준으로하면 들어가서 release로 해놓고 빌드를 한다.

10. 도구 - 옵션을 들어가면 프로젝트및 솔루션에서 VC++ 디렉토리에서 다음파일의 디렉토리의 표시에서 포함 파일로 설정을 한다음 "C:\OpenCV2.0\include\opencv"이경로로 추가한다.

11. 다음파일의 디렉토리의 표시에서 라이브러리 파일로 바꾼다음 C:\cmake_binary_dir\lib\release로 설정한다.

12. 새로운 프로젝트를 생성하고 프로젝트-프로젝트 속성-링커-입력-추가종속성에서 cv200.lib highgui200.lib cvaux200.lib cxcore200.lib 이 파일을 입력한다.

13. C:\cmake_binary_dir\bin\Debug 폴더에 있는 cv200.dll highgui200.dll cvaux200.dll cxcore200.dll 파일들을 복사하여 현재 코딩중인 프로젝트의 폴더에 붙여 넣습니다.



기본 사용법

<code>cv2.imread(file_name, flag)</code>	이미지를 읽어 Numpy 객체로 만드는 함수
--	--------------------------

- file_name: 읽고자 하는 이미지 파일

- flag: 이미지를 읽는 방법 설정

IMREAD_COLOR: 이미지를 Color로 읽고, 투명한 부분은 무시

IMREAD_GRAYSCALE: 이미지를 Grayscale로 읽기

IMREAD_UNCHANGED: 이미지를 Color로 읽고, 투명한 부분도 읽기(Alpha)

<code>cv2.imshow(title, image)</code>	특정한 이미지를 화면에 출력합니다.
---------------------------------------	---------------------

- title: 윈도우 창의 제목

- image: 출력할 이미지 객체

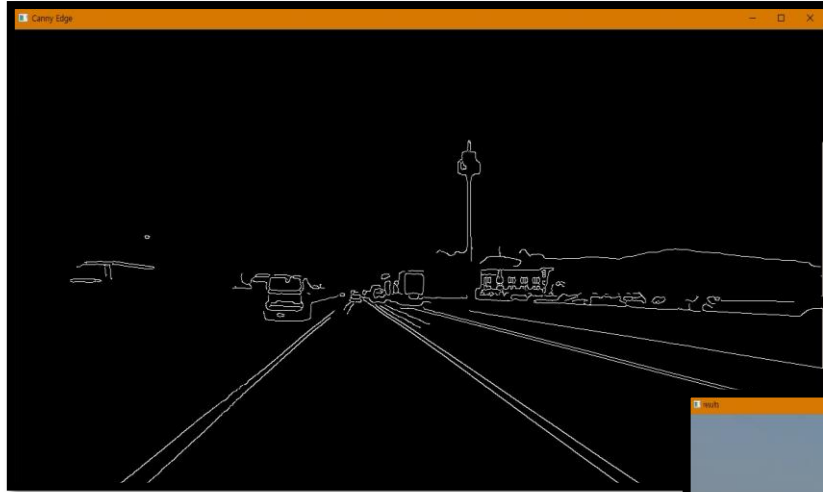


cat.jpg



기법 활용 (1)

Canny Edge



원본
Image

Input Grey Image



Histogram equilization



Gaussian Smoothing



기법 활용 (2)

메인 코드

```
import cv2

src = cv2.imread("Image/fruits.jpg", cv2.IMREAD_COLOR)
height, width, channel = src.shape

dst = cv2.pyrUp(src, dstsize=(width * 2, height * 2), borderType=cv2.BORDER_DEFAULT)
dst2 = cv2.pyrDown(src)

cv2.imshow("src", src)
cv2.imshow("dst", dst)
cv2.imshow("dst2", dst2)
cv2.waitKey()
cv2.destroyAllWindows()
```

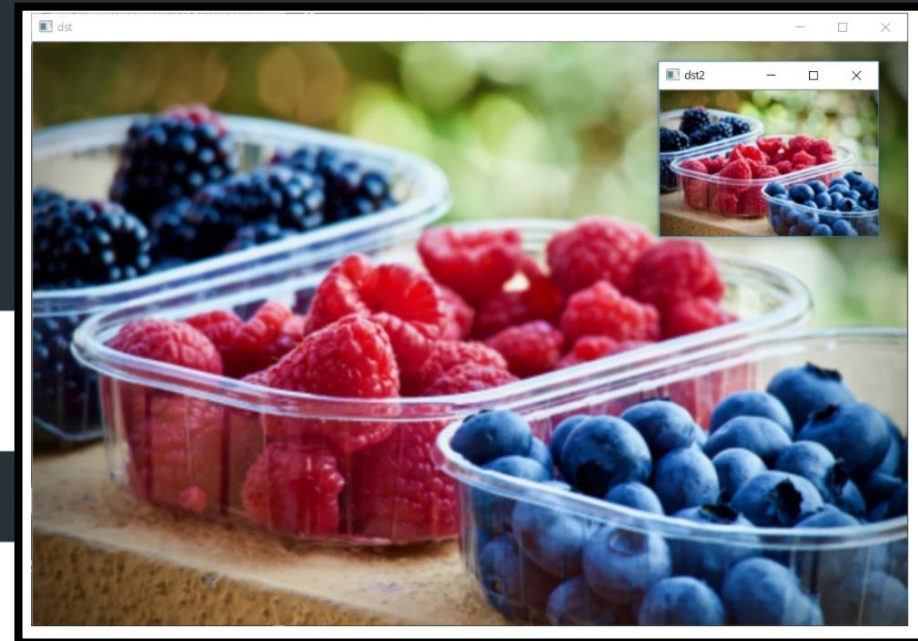
세부 코드

```
height, width, channel = src.shape
```

height, width, channel = src.shape 를 이용하여 해당 이미지의 높이, 너비, 채널의 값을 저장합니다.

너비와 높이를 이용하여 출력 이미지 크기를 설정할 수 있습니다.

이미지 피라미드



기법 활용 (3)

메인 코드

```
import cv2
import numpy as np

src = np.zeros((768, 1366, 3), dtype=np.uint8)

src = cv2.line(src, (100, 100), (1200, 100), (0, 0, 255), 3, cv2.LINE_AA)
src = cv2.circle(src, (300, 300), 50, (0, 255, 0), cv2.FILLED, cv2.LINE_4)
src = cv2.rectangle(src, (500, 200), (1000, 400), (255, 0, 0), 5, cv2.LINE_8)
src = cv2.ellipse(src, (1200, 300), (100, 50), 0, 90, 180, (255, 255, 0), 2)

pts1 = np.array([[100, 500], [300, 500], [200, 600]])
pts2 = np.array([[600, 500], [800, 500], [700, 600]])
src = cv2.polylines(src, [pts1], True, (0, 255, 255), 2)
src = cv2.fillPoly(src, [pts2], (255, 0, 255), cv2.LINE_AA)

src = cv2.putText(src, "YUNDAEHEE", (900, 600), cv2.FONT_HERSHEY_COMPLEX, 2, (255, 255, 255), 3)

cv2.imshow("src", src)
cv2.waitKey()
cv2.destroyAllWindows()
```

세부 코드

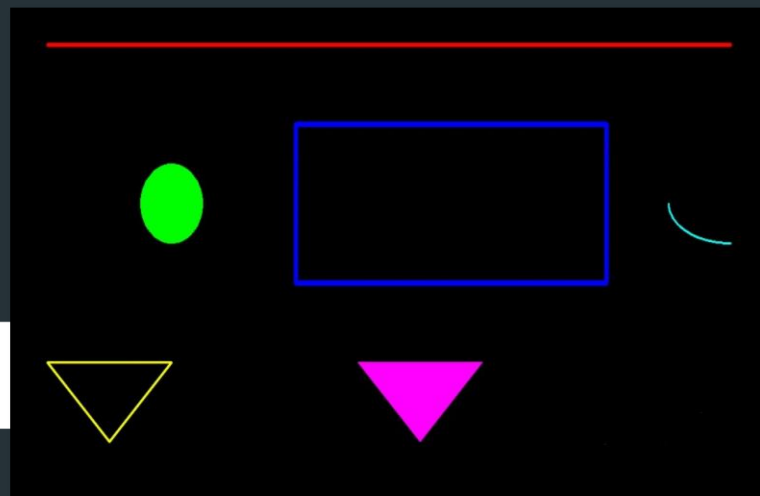
```
src = cv2.line(src, (100, 100), (1200, 100), (0, 0, 255), 3, cv2.LINE_AA)
```

직선 그리기 함수(cv2.line)로 입력 이미지에 직선을 그릴 수 있습니다.

`dst = cv2.line(src, pt1, pt2, color, thickness, lineType, shift)` 는 입력 이미지(src)에 시작 좌표(pt1)부터 도착 좌표(pt2)를 지나는 특정 색상(color)과 두께(thickness)의 직선을 그립니다. 추가로 선형 타입(lineType), 비트 시프트(shift)를 설정할 수 있습니다.

설정된 입력값으로 그려진 직선이 포함된 출력 이미지(dst)를 생성합니다.

선형 타입, 비트 시프트



기법 활용 (4)

메인 코드

```

import datetime
import cv2

capture = cv2.VideoCapture("/Image/Star.mp4")
fourcc = cv2.VideoWriter_fourcc(*'XVID')
record = False

while True:
    if(capture.get(cv2.CAP_PROP_POS_FRAMES) == capture.get(cv2.CAP_PROP_FRAME_COUNT)):
        capture.open("/Image/Star.mp4")

    ret, frame = capture.read()
    cv2.imshow("VideoFrame", frame)

    now = datetime.datetime.now().strftime("%d_%H-%M-%S")
    key = cv2.waitKey(33)






    if key == 27:
        break
    elif key == 26:
        print("캡처")
        cv2.imwrite("D://" + str(now) + ".png", frame)
    elif key == 24:
        print("녹화 시작")
        record = True
        video = cv2.VideoWriter("D://" + str(now) + ".avi", fourcc, 20.0, (frame.shape[1], frame.shape[0]))
    elif key == 3:
        print("녹화 중지")
        record = False
        video.release()

    if record == True:
        print("녹화 중..")
        video.write(frame)

capture.release()
cv2.destroyAllWindows()

```

캡처 및 녹화

 07_21-18-54.png	2018-10-07 오후 9	PNG 파일	1,017KB
 07_21-35-25.avi	2018-10-07 오후 9	AVI - Windows 기...	2,042KB
 07_21-37-02.png	2018-10-07 오후 9	PNG 파일	1,040KB
 07_21-37-04.png	2018-10-07 오후 9	PNG 파일	966KB
 07_21-37-05.avi	2018-10-07 오후 9	AVI - Windows 기...	1,920KB

기법 활용 (5)

메인 코드

```
import numpy as np
import cv2

src = cv2.imread("analysis.jpg")
gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
_, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)

_and = cv2.bitwise_and(gray, binary)
_or = cv2.bitwise_or(gray, binary)
_xor = cv2.bitwise_xor(gray, binary)
_not = cv2.bitwise_not(gray)

src = np.concatenate((np.zeros_like(gray), gray, binary, np.zeros_like(gray)), axis = 1)
dst = np.concatenate((_and, _or, _xor, _not), axis = 1)
dst = np.concatenate((src, dst), axis = 0)

cv2.imshow("dst", dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

비트 연산 표현

세부 코드

```
src = cv2.imread("analysis.jpg")
gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
_, binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
```

원본 이미지(src)와 그레이스케일(gray), 이진화(binary)을 선언합니다.

연산 이미지는 그레이스케일 이미지와 127 임계값을 갖는 이진화 이미지를 사용합니다.



기법 활용 (6)

메인 코드

```

import numpy as np
import cv2

src = cv2.imread("road.jpg")
dst = src.copy()
gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
canny = cv2.Canny(gray, 5000, 1500, apertureSize = 5, L2gradient = True)
lines = cv2.HoughLines(canny, 0.8, np.pi / 180, 150, srn = 100, stn = 200, min_theta = 0, max_theta = np.pi)

for i in lines:
    rho, theta = i[0][0], i[0][1]
    a, b = np.cos(theta), np.sin(theta)
    x0, y0 = a*rho, b*rho

    scale = src.shape[0] + src.shape[1]

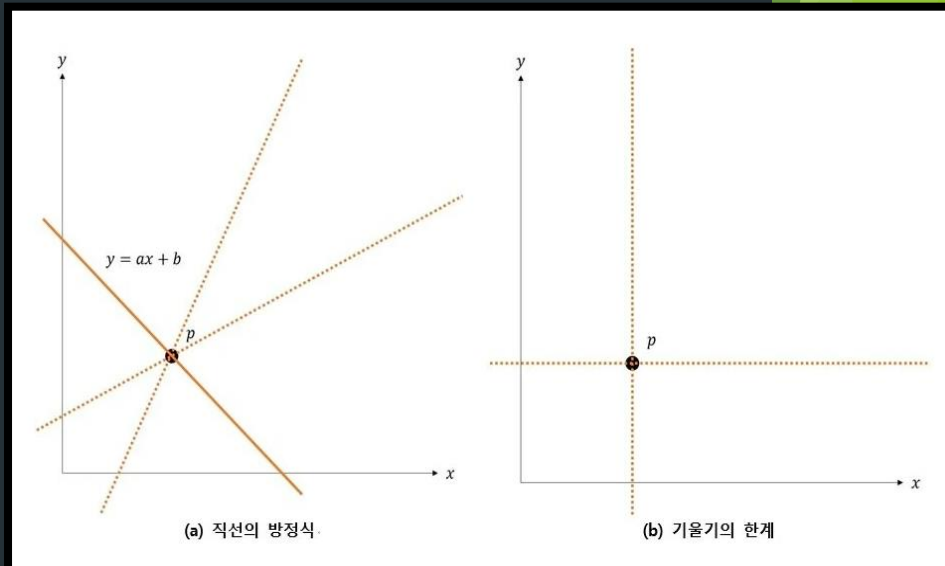
    x1 = int(x0 + scale * -b)
    y1 = int(y0 + scale * a)
    x2 = int(x0 - scale * -b)
    y2 = int(y0 - scale * a)

    cv2.line(dst, (x1, y1), (x2, y2), (0, 0, 255), 2)
    cv2.circle(dst, (x0, y0), 3, (255, 0, 0), 5, cv2.FILLED)

cv2.imshow("dst", dst)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

표준 허프 변환



기법 활용 (7)

메인 코드

```
import cv2
import numpy as np

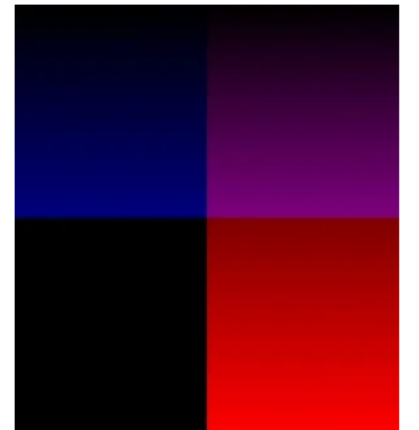
gray = np.linspace(0, 255, num=90000, endpoint=True, retstep=False, dtype=np.uint8).reshape(300, 300, 1)
color = np.zeros((300, 300, 3), np.uint8)
color[0:150, :, 0] = gray[0:150, :, 0]
color[:, 150:300, 2] = gray[:, 150:300, 0]

x, y, c = 200, 100, 0
access_gray = gray[y, x, c]
access_color_blue = color[y, x, c]
access_color = color[y, x]

print(access_gray)
print(access_color_blue)
print(access_color)

cv2.imshow("gray", gray)
cv2.imshow("color", color)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

문자열, 리스트, 튜플



세부 코드

```
gray = np.linspace(0, 255, num=90000, endpoint=True, retstep=False, dtype=np.uint8).reshape(300, 300, 1)
```

회색 그라데이션 이미지인 `gray`를 선언합니다.

그라데이션 이미지는 등간격(`numpy.linspace`)을 활용해 구현할 수 있습니다.

등간격 배열을 생성하면, 이미지 배열이 아니므로, 차원 변경(`numpy.reshape`) 함수를 활용해 단일 채널 이미지로 변경합니다.

- 등간격 사용하기 : [Python Numpy 4강 바로가기](#)
- Tip : $300 \times 300 \times 1$ 크기이 이미지를 생성할 예정이므로, `num`은 90000을 갖습니다.

기법 활용 (8)

메인 코드

```
import cv2

src = cv2.imread("Image/car.png", cv2.IMREAD_COLOR)

gray = cv2.cvtColor(src, cv2.COLOR_RGB2GRAY)
ret, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_OTSU)
binary = cv2.bitwise_not(binary)

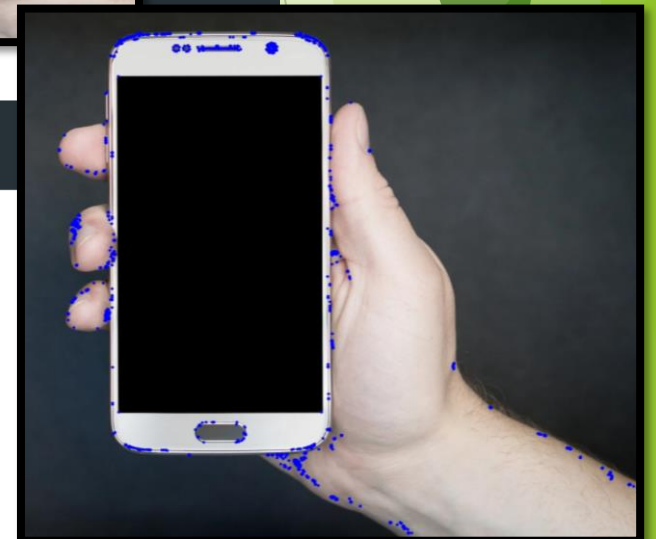
contours, hierarchy = cv2.findContours(binary, cv2.RETR_LIST, cv2.CHAIN_APPROX_TC89_KCOS)

for contour in contours:
    epsilon = cv2.arcLength(contour, True) * 0.02
    approx_poly = cv2.approxPolyDP(contour, epsilon, True)

    for approx in approx_poly:
        cv2.circle(src, tuple(approx[0]), 3, (255, 0, 0), -1)

cv2.imshow("src", src)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

윤곽선의 근사 다각형



세부 코드

```
for contour in contours:
    epsilon = cv2.arcLength(contour, True) * 0.02
```

반복문을 사용하여 색인값과 하위 윤곽선 정보로 반복합니다.

근사치 정확도를 계산하기 위해 윤곽선 전체 길이의 2%로 활용합니다

윤곽선의 전체 길이를 계산하기 위해 `cv2.arcLength()` 을 이용해 검출된 윤곽선의 전체 길이를 계산합니다.

`cv2.arcLength(윤곽선, 폐곡선)` 을 의미합니다.

윤곽선 은 검출된 윤곽선들이 저장된 Numpy 배열입니다.

폐곡선 은 검출된 윤곽선이 닫혀있는지, 열려있는지 설정합니다.

- Tip : 폐곡선을 True로 사용할 경우, 윤곽선이 닫혀 최종 길이가 더 길어집니다. (끝점 연결 여부를 의미)

기법 활용 (9)

메인 코드

```
import numpy as np
import cv2

src = cv2.imread("flower.jpg")

data = src.reshape(-1, 3).astype(np.float32)
criteria = (cv2.TERM_CRITERIA_MAX_ITER + cv2.TERM_CRITERIA_EPS, 10, 0.001)
retval, bestLabels, centers = cv2.kmeans(data, 5, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)

centers = centers.astype(np.uint8)
dst = centers[bestLabels].reshape(src.shape)

cv2.imshow("dst", dst)
cv2.waitKey()
cv2.destroyAllWindows()
```

같은 중심 할당된 데이터



세부 코드

```
data = src.reshape(-1, 3).astype(np.float32)
criteria = (cv2.TERM_CRITERIA_MAX_ITER + cv2.TERM_CRITERIA_EPS, 10, 0.001)
```

K-평균 군집화 알고리즘 함수의 입력 데이터 조건을 맞추기 위해, reshape() 메서드와 astype() 메서드를 활용해 차원과 데이터 형식을 변경합니다.

K-평균 군집화 알고리즘은 [N, 3]의 차원과 float32의 데이터 형식을 입력 조건으로 사용합니다.

또한, 알고리즘의 종료 기준(TermCriteria)을 설정합니다.

종료 기준은 알고리즘의 반복 횟수가 10 회가 되거나, 정확도가 0.001 이하일 때 종료됩니다.

[종료 기준 함수 자세히 알아보기](#)

기법 활용 (10)

메인 코드

```
import cv2

src = cv2.imread("Image/convex.png")
dst = src.copy()

gray = cv2.cvtColor(src, cv2.COLOR_RGB2GRAY)
ret, binary = cv2.threshold(gray, 150, 255, cv2.THRESH_BINARY_INV)

contours, hierarchy = cv2.findContours(binary, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_NONE)

for i in contours:
    M = cv2.moments(i)
    cX = int(M['m10'] / M['m00'])
    cY = int(M['m01'] / M['m00'])

    cv2.circle(dst, (cX, cY), 3, (255, 0, 0), -1)
    cv2.drawContours(dst, [i], 0, (0, 0, 255), 2)

cv2.imshow("dst", dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

세부 코드

```
for i in contours:
    M = cv2.moments(i, False)
    cX = int(M['m10'] / M['m00'])
    cY = int(M['m01'] / M['m00'])
```

cv2.moments() 를 활용해 윤곽선에서 모멘트를 계산합니다.

cv2.moments(배열, 이진화 이미지) 을 의미합니다.

윤곽선이나 이미지 (0차~3차)

공간 모멘트(spatial moments)

$$m_{ij} = \sum_{x,y} (array(x,y) \times x^i y^j)$$

중심 모멘트(central moments)

$$\mu_{ij} = \sum_{x,y} (array(x,y) \times (x - \bar{x})^i (y - \bar{y})^j)$$

정규화된 중심 모멘트(normalized central moments)

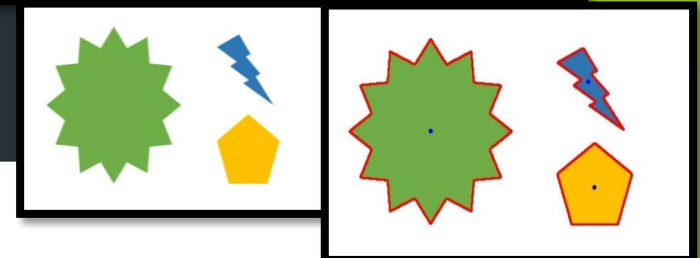
$$nu_{ij} = \frac{\mu_{ij}}{m_{00}^{i+j+1}}$$

모멘트 구조

$$M = \begin{cases} \text{0차 모멘트:} & m_{00} \\ \text{1차 모멘트:} & m_{10}, m_{01} \\ \text{2차 모멘트:} & m_{11}, m_{20}, m_{02} \\ \text{3차 모멘트:} & m_{u11}, m_{u20}, m_{u02} \\ \text{2차 중심 모멘트:} & \mu_{u11}, \mu_{u20}, \mu_{u02} \\ \text{3차 중심 모멘트:} & \mu_{u21}, \mu_{u12}, \mu_{u30}, \mu_{u03} \\ \text{2차 정규화된 중심 모멘트:} & nu_{u11}, nu_{u20}, nu_{u02} \\ \text{3차 정규화된 중심 모멘트:} & nu_{u21}, nu_{u12}, nu_{u30}, nu_{u03} \end{cases}$$

반환되지 않는 값

$$\begin{cases} nu_{00} = m_{00} \\ nu_{00} = 1 \\ nu_{01} = mu_{10} = nu_{01} = nu_{10} = 0 \end{cases}$$



감사합니다.

