# # 1. Day1\_영상입출력

1. 컴퓨터 비전(Computer Vision)

- 인공지능의 한 분야

- 컴퓨터와 시스템을 통해 디지털 이미지, 비디오 및 기타 시각적 입력에서 의미 있는 정보를

추출한 다음 이러한 정보를 바탕으로 작업을 실행

- 영상 처리, 동영상 처리, 객체 탐지(사람인식), 패턴 인식, 광학 문자 인식 등에서 활용

ex) cctv 화재감지해서 119에 연락(연기, 불)하는 AI, 자율주행, 자율비행

2. OpenCV(Open Source Computer Vision library)

- 컴퓨터 비전과 이미지 처리를 위한 오픈소스 라이브러리

- 1999년 Intel에서 영상처리 기술을 개발하기 위한 목적

- 2000년 BSD 라이선스로 배포

- 2011년 이후 OpenCV2로 개발 시작(파이썬)

3. OpenCV 설치

- pip install opencv-python

- 파이참 메뉴 -> settings -> project 이름 -> python interpreter -> '+' 클릭 후

-> opencv-python 검색 -> install package 클릭

4. 영상(Image)

픽셀(pixel) : 영상을 구성하는 가장 작은 단위

영상(Image) : 픽셀이 바둑판 모양의 격자에 나열되어 있는 형태. 2차원 행렬 형태

1) 그레이스케일 영상

- 흑백 사진처럼 색상 정보가 없는 영상

- 밝기 정보만으로 구성된 영상

- 밝기 정보는 256단계로 표현(0~255)

- 저장형태 : numpy.uint8 = 8bit = 1byte(256개의 밝기는 2의 8승이므로 8개의 비트필요)

- 가로크기 \* 세로크기 = 28 \* 28 = 784 bytes

2) 트루컬러 영상

- 컬러 사진처럼 색상 정보를 가지고 있기 때문에 다양한 색상을 표현할 수 있는 영상

- red, green, blue 색 성분을 사용하고 각 256단계로 표현

- 튜플 형태로 표현 -> (255, 255, 255)

- 저장형태 : numpy.ndarray = 3byte

- 가로크기 \* 세로크기 \* 3 = 28 \* 28 \* 3 = 2352 bytes

영상 파일 형식

1) bmp

- 픽셀 데이터를 압축하지 않고 그대로 저장

- 용량이 매우 큼(픽셀의 일반적인 저장 크기)

- 파일 구조가 단순해서 별도의 라이브러리 없이 프로그래밍이 가능

2) jpg, jpeg

- 압축률이 좋아서 파일 용량이 크게 감소

- 사진과 같은 컬러영상을 저장

- 손실 압축(잦은 복사 시 알고리즘이 깨져 화질낮아짐)

3) gif

- 움직이는 영상 지원

- 256색 이하의 영상으로 저장

- 무손실 압축(잦은 복사에도 화질유지)

4) png

- 웹 이미지용으로 권장

- 무손실, 손실 압축 둘다 가능

- 알파 채널(투명도)를 지원, (255, 255, 255, 0.5:투명도)

5. 영상의 입출력

영상 읽기

cv2.imread(영상, 방법)

영상: 파일 이름 또는 경로

방법

- IMREAD\_GRAYSCALE: 영상을 그레이스케일로 읽기

- IMREAD\_COLOR: 트루컬러로 읽기(기본값, 불러온 영상의 색상 정보가 BGR순서)

채널 순서 변환

cv2.cvtColor(영상, 변환방법)

영상: 변수 또는 파일경로

변환방법

- cv2.COLOR\_BGR2RGB: BGR을 RGB로 변환

- cv2.COLOR\_RGB2BGR: RGB를 BGR로 변환

6. 영상의 차원

변수명 = cv2.imread(영상, 방법)

type(변수명): <class 'numpy.ndarray'>

변수명.shape: (h,w) 또는 (h, w, 3)

변수명.dtype: uint8

7. 이미지 생성

# 

# # 2. Day2\_도형그리기+카메라+동영상처리

1. 도형 그리기 - 영상에 객체 구분해서 박스치는 용도!

1) 직선

cv2.line(영상, 직선의 시작점과 끝점 튜플, 선 색상, 선 두께, 선 타입)

2) 사각형

cv2.rectangle(영상, 사각형 꼭지점 좌표 튜플, 선 색상, 선 두께)

✔ 선 두께를 -1로 설정하면 사각형 내부를 색상으로 채움

3) 원

cv2.circle(영상, 원의 중앙좌표 튜플, 반지름, 선 색상, 선 두께)

✔ 선 두께를 -1로 설정하면 원 내부를 색상으로 채움

4) 문자열

cv2.putText(영상, 문자열, 영상에서 문자열을 출력할 위치 좌표 튜플, 글꼴, 폰트크기, 색상)

2. 동영상 처리하기

cv2.VideoCapture 클래스

카메라와 동영상으로부터 프레임(frame)을 받아오는 작업을 처리함

1) 카메라 영상 입력

cv2.VideoCapture(index)

index: 시스템의 기본 카메라를 open하려면 0, 별도의 카메라를 open하려면 고유의 index 값

cv2.VideoCapture.isOpended()

카메라에 연결이 되었는지 여부(True: 성공, False: 실패)

cv2.CAP\_PROP\_FRAME\_WIDTH: 카메라로 읽어들인 영상의 가로 사이즈

cv2.CAP\_PROP\_FRAME\_HEIGHT: 카메라로 읽어들인 영상의 세로 사이즈

cv2.VideoCapture.read()

ret: 영상이 정상적으로 리턴되었는지 여부(True, False)

frame: 영상(그레이스케일 영상 또는 컬러 영상)

2) 동영상 입력

cv2.VideoCapture(파일명)

cv2.CAP\_PROP\_FRAME\_COUNT: 동영상에서 읽어들인 영상의 총 프레임 수

cv2.CAP\_PROP\_FPS: 동영상에서 1초당 프레임 수

+) 총 프레임수와 1초당 프레임 수로 동영상의 속도와 재생 시간을 계산할 수 있음

3) 동영상 출력

FourCC(Four Charactor Code): 코드포맷

- 4바이트로 된 문자열이며 데이터 형식을 구분하는 고유 글자

- 주로 AVI 파일의 영상 코덱을 구분할 때 사용

cv2.VideoWriter.fourcc(\*'DIVX') // Divx

cv2.VideoWriter.fourcc(\*'XVID') // Xvid

cv2.VideoWriter.fourcc(\*'MP4V') // mp4

파일 저장

cv2.VideoWriter(파일명, FourCC객체, fps, 프레임사이즈 튜플, 컬러영상 여부)

프레임사이트 튜플: (w, h)의 튜플

컬러영상 여부: True(컬러), False(그레이스케일)

3. 키보드 이벤트

cv2.waitKey(delay)

delay : 밀리초 단위 대기(0보다 작거나 같으면 무한정 기다림. 기본값은 0)

return : 눌려진 키의 아스키 코드(ESC:27, ENTER:13, TAB: 9 ...)

✔ ord(): 문자열을 입력하면 해당 문자열의 아스키 코드값을 반환

4. 마우스 이벤트

cv2.setMouseCallback(윈도우이름, 콜백함수, 파라미터)

윈도우이름: 마우스 이벤트를 처리할 윈도우 창의 이름

콜백함수: 마우스 이벤트가 발생할 때 호출될 콜백 함수를 설정

def 함수명(event, x, y, flags, param):

pass

event: 이벤트 객체

x, y: 마우스 x,y 좌표

flags: 마우스 버튼이 눌리고 있는지, 떼졌는지 여부

param: 추가적인 정보가 전달되었다면 저장

파라미터: 콜백함수에 전달할 추가적인 정보

# # 3. Day3\_영상의화소처리(밝기+색조절)

1. 영상의 화소처리

영상의 특정 좌표 픽셀값을 변경하여 출력 영상의 좌표 픽셀을 설정하는 모든 연산

1) 밝기 조절

- 영상을 전체적으로 밝게하거나 어둡게 하는 연산

cv2.add(첫번째 영상, 두번째 영상)

cv2.subtract(첫번째 영상, 두번째 영상)

cv2.multiply(첫번째 영상, 두번째 영상)

cv2.divide(첫번째 영상, 두번째 영상)

문제1)

영상 + 영상, cv2.add(영상, 영상)의 차이점 설명하기

2) 가중치 합(weighted sum)

두 영상의 같은 위치에 존재하는 픽셀값에 대하여 가중합을 계산해서 결과 영상의 픽셀값으로 설정

cv2.addWeighted(첫 번째 영상, 알파값(가중치), 두번째 영상, 1-알파값, 상수)

알파값: 첫 번째 영상에 설정할 가중치. 가중치의 합은 1이 됨

상수: 연산 결과에 가감할 상수. 기본값은 0

3) 두 영상의 차이(absdiff)

두 영상의 픽셀 값만 빼면 음수가 나올 수 있는데, 해당 값에 절대값을 취한 값

cv2.absdiff(첫번째 영상, 두번째 영상)

문제2)

add, addWeightedm subtract, absdiff

dog.jpg와 squared.bmp 위 연산을 시도

matplotlib의 subplot을 이용하여 이미지를 비교

(단, addWeighted는 50%, 50%로 설정)

2. 컬러 영상과 색

- 컬러 영상은 3차원 배열

- numpy.ndarray

- img.shape: (h, w, 3) -> OpenCV에서는 BGR

- 색상 채널 분리: cv2.split(영상)

- 색상 채널 결합: cv2.merge(입력 영상 리스트 또는 튜플)

문제3

candies.png를 불러와 각 채널 b, g, r로 분리하여 화면에 출력

(단, cv2.split() 함수를 사용하지 않고, 슬라이싱을 활용)

3. 히스토그램(histogram)

영상의 픽셀값 분포를 그래프 형태로 표현

예) 그레이스케일 영상에서 밝기 정보에 해당하는 픽셀의 개수를 구하고 막대 그래프로 표현

cv2.calcHist(영상(리스트구조), 히스토그램을 구할 채널, None, 빈의 개수를 나타내는 리스트, 히스토그램 각 차원의 최소값과 최대값으로 구성된 리스트)

+) 빈의갯수 : 데이터구조를 몇개로 분할하는지

✔ 영상과 히스토그램의 관계

- 밝은 영상이면 히스토그램이 전체적으로 오른쪽으로 치우처져 있음

- 어두운 영상이면 히스토그램이 전체적으로 왼쪽으로 치우처져 있음

- 명암비가 확실한 영상이면 히스토그램이 양쪽으로 분포해 있음

문제4

dog.bmp 영상을 사용하여 3채널(BGR)로 계산해 히스토그램 그리기

(단, 하나의 plot에서 BGR 그래프를 그리기)

1. 균등화, 평탄화(Equalization)

- 히스토그램을 활용하여 이미지의 품질을 개선하기 위한 방법

- 화소값을 0 ~ 255 사이에 고루게 분포하도록 개선

cv2.equalizeHist(영상)

2. 색공간

YCbCr

- 색공간을 밝기 정보로 표현하는 방식

- Y: 밝기 정보

- YCbCr을 줄여서 YCC라고 부르기도 함

# # 4. Day4\_균등화/평탄화/정규화+ROI

1. 균등화, 평탄화(Equalization)

- 히스토그램을 활용하여 이미지의 품질을 개선하기 위한 방법

- 화소값을 0 ~ 255 사이에 고루게 분포하도록 개선

cv2.equalizeHist(영상)

2. 색공간

YCbCr

- 색공간을 밝기 정보로 표현하는 방식

- Y: 밝기 정보

- YCbCr을 줄여서 YCC라고 부르기도 함

HSV

- 색을 표현하는 방법이자 색을 배치하는 방식

- 색상(빨강, 녹생, 파랑 -> H), 채도(선명도 -> S), 명도(빛의 밝기 -> V)의 좌표를 써서 특정한 색을 지정

3. CLAHE(Contrast Limited Adaptive Histogram Equalization)

- CLAHE는 이미지의 전체적인 대비를 증가시키는 기존의 평탄화 기술을 개선한 것

- 이미지의 일정한 영역으로 나누어 평탄화를 적용(기존은 전체 이미지의 대비를 활용 / 특정부분이 밝으면 이상해짐)

변수 = cv2.createCLAHE(대비, 영역 크기)

변수.apply(영상) # 여러 영상 평탄화 가능

대비: CLAHE 알고리즘에서 각 픽셀의 대비를 얼마나 증가시킬지 제한

4. 정규화

- 특정 영역에 값이 몰려 있는 경우 화질을 개선하고, 이미지 간의 연산 시 서로 조건이 다른 경우 같은 조건으로

cv2.normalize(정규화 이전 영상, 정규화 이후 영상, 정규화 구간1, 정규화 구간2, 정규화 알고리즘)

cv2.NORM\_MINMAX: 정규화 구간1 ~ 정규화 구간2

cv2.NORM\_L1: 전체 합으로 나눔

cv2.NORM\_L2: 단위 벡터로 정규화

cv2.NORM\_INF: 최대값으로 나눔

5. inRange()

영상에서 지정된 범위 안에 픽셀을 선택(색 조건)

cv2.inRange(영상, min값, max값)

BGR에서의 녹색계열

0 <= B <= 100

128 <= G <= 255

0 <= R <= 100

HSV에서의 녹색계열

50 <= H <= 80

150 <= S <= 255

0 <= V <= 255

6. copyTo()

마스크 연산을 지원하는 픽셀 값 복사 함수

cv2.copyTo(영상, 마스크, 출력영상)

문제

무료 동영상 사이트에서 영상을 다운받아 woman.mp4를 합성하여 출력하는 영상을 만들어보자

(단, 동영상의 해상도는 woman.mp4와 같아야 하며 esc를 누르면 언제든지 프로그램을 종료해야함)

7. 관심 영역(ROI, Region of Interest)

영상 내에서 관심이 있는 영역

cv2.selectionROI(창이름, 영상, 중앙 좌표 여부=False)

중앙 좌표 여부: True일 경우, 선택한 ROI를 중앙좌표로 계산, False(왼쪽 상단)

# # 5. Day5\_영상의 이진화 및 변환

1. 영상의 이진화(Binarization)

- 픽셀을 검은색 또는 흰색과 같이 두 개의 값으로 나누는 작업

- 영상에서 의미있는 관심영역(ROI)과 비관심영역으로 구분할 때 사용(마스크와 같은 개념)

- 배경과 객체를 나눌 때

- 영상의 이진화 연산을 할 때 나누는 특정한 값을 임계값이라고 함

cv2.threshold(영상, 임계값, 최대값, 플래그)

플래그

cv2.THRESH\_BINARY: 픽셀값이 임계값을 넘으면 최대값으로 지정하고 넘지 못하면 0으로 지정

cv2.THRESH\_BINARY\_INV: THRESH\_BINARY의 반대(주로 사용)

cv2.THRESH\_TRUNC: 픽셀값이 임계값을 넘으면 최대값으로 지정하고, 넘지 못하면 원래값을 유지

cv2.THRESH\_TOZERO: 픽셀값이 임계값을 넘으면 원래 값을 유지하고, 넘지 못하면 0으로 지정

cv2.THRESH\_TOZERO\_INV: THRESH\_TOZERO의 반대

2. 오츠의 이진화 알고리즘(임계값을 자동으로 잡아줌)

- 자동 이진화

- 자동으로 임계값을 구하는 알고리즘. 임계값을 구분하는 가장 좋은 방법으로 사용

cv2.thresold(영상, 임계값, 최대값, 플래그 | cv2.THRESH\_OTSU) # 임계값 0 세팅

- 임계값을 임의로 정해 픽셀을 두 부류로 나누고 두 부류의 명암 분포를 구하는 작업을 반복하여 모든 경우의

수 중에서 두 부류의 명암 분류가 가장 균일할 때 임계값을 선택

3. 지역 이진화

- 균일하지 않은 조명 환경에서 사용하는 이진화 방법

- 전체 구역을 N등분하고 각각의 구역에 이진화를 한 뒤에 이어붙이는 방법

- 여러 개의 임계값을 이용할 수 있음

문제1

rice.png를 이용하여 가로, 세로 4등분하고 자동 이진화를 해보자

4. 적응형 이진화

- 노이즈를 제거한 뒤에 Otsu 이진화를 적용

- 영상을 여러 영역으로 나눈 뒤, 그 주변 픽셀 값만 활용하여 임계값을 구함

cv2.adaptiveThreshold(영상, 임계값을 만족하는 픽셀에 적용할 값, 임계값 결정방법, Threshold 적용방법, 블록사이즈, 가감할 상수)

임계값을 만족하는 픽셀에 적용할 값: 총 픽셀의 개수(255)

임계값 결정 방법

cv2.ADAPTIVE\_THRESH\_MEAN\_C: 이웃 픽셀의 평균으로 결정 -> 선명하지만 잡티가 많아짐

cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C: 가우시안 분포에 따른 가중치의 합으로 결정 -> 선명도는 조금 떨어지지만 잡티가 적음

블록사이즈: 3이상의 값. 사이즈가 클수록 연산 시간이 오래 걸림

5. 이미지 유사도

- 픽셀 값의 분포가 서로 비슷하다면 유사한 이미지일 확률이 높음

cv2.compareHist(히스토그램1, 히스토그램2, 알고리즘)

cv2.HISTCMP\_CORREL: 상관관계(1: 완전 일치, -1: 완전 불일치, 0: 무관계)

cv2.HISTCMP\_CHISQR: 카이제곱(0: 완전 일치, 무한대: 완전 불일치) # 커질수록 불일치(얼마나 차이나는지)

cv2.HISTCMP\_INTERSECT: 교차(1: 완전 일치, 0: 완전 불일치)(얼마나 겹치는지)

cv2.HISTCMP\_BHATTACHARYYA: 밀도함수(0: 완전 일치, 1: 완전 불일치)

(히스토그램 간 유사성)

6. 영상의 변환

- 영상을 구성하는 픽셀의 배치 구조를 변경함으로 전체 영상의 모양을 바꾸는 작업

1) 이미지 이동(translate)

원래 있던 좌표에 이동시키려는 거리만큼 연산(shift 연산)

M = [1 0 a]

[0 1 b]

x방향으로 a만큼, y방향으로 b만큼 이동하는 행렬

cv2.warpaffine(영상, 2\*3 변환행렬, 결과영상, 보간법 알고리즘)

\* 보간법 알고리즘(interpolation)

cv2.INTER\_LINEAR: 인접한 4개의 픽셀 값에 거리 가중치 사용 -> 속도는 빠르지만 퀄리티가 좀 떨어짐

cv2.INTER\_NEAREST: 가장 가까운 픽셀 값 사용 -> 속도가 가장 빠르지만 퀄리티가 가장 많이 떨어짐

cv2.INTER\_AREA: 픽셀 영역 관계를 이용한 재샘플링 -> 영역적인 정보를 추출해서 결과 영상을 세팅하는 방법, 다운 샘플링시 효과적(축소)

cv2.INTER\_CUBIC: 인접한 16개의 픽셀 값에 가중치를 사용 -> 퀄리티는 가장 좋지만 속도가 떨어짐

2) 크기 변환(resize)

영상의 크기를 원본 영상보다 크게 또는 작게 만드는 변환

cv2.resize(영상, 결과, x와y방향 스케일 비율, 보간법)

3) 회전(rotation)

영상을 특정 각도만큼 회전시키는 변환(반시계 방향)

cv2.getRotationMatrix2D(중심좌표, 회전각도 확대비율) -> affine 행렬 생성 및 적용

회전각도: 반시계방향(기본값), 음수는 시계방향

확대비율: 0~1사이의 실수

4) 투시 변환(perspective)

- 직사각형 형태의 영상을 임의의 입체감 있는 사각형 형태로 변경할 수 있는 변환

- 원본 영상에 있는 직선은 결과 영상에서 그대로 유지되지 않고 평행 관계가 깨질 수 있음

- 투시 변환은 보통 3\*3 크기의 실수 행렬로 표현

(8개의 파라미터로 표현할 수 있지만, 좌표 계산 편의상 9개의 원소를 갖는 행렬로 표현) # 변환행렬

cv2.getPerspectiveTransform(영상, 4개의 결과 좌표점) -> 투시 변환 행렬

cv2.warpPerspective(영상, 투시변환행렬, 결과영상크기)

# # 6. Day6\_필터링(블러링, 엣지검출)

1. 필터링

커널 또는 필터라고 하는 행렬을 정의하고, 이미지 위에서 이동해가며 커널과 겹쳐진 이미지 영역과

연산을 한 후에, 그 결과값을 연산을 진행한 이미지 픽셀을 대신하여 새로운 이미지를 만드는 연산

cv2.filter2D(영상, -1, 커널사이즈, 중심점 좌표, 추가될 값, 가장자리 화소처리)

-1: 입력과 동일한 크기의 영상

커널: 3\*3, 5\*5 ...

가장자리 화소처리: 픽셀의 끝의 색상을 채우는 방식

BORDER\_CONSTANT: 000|abcdef|000 #검정색채움

BORDER\_REPLICATE: aaa|abcdef|fff #색복사채움

1) 블러링(Blurring)

초점이 맞지 않은듯 영상을 흐릿하게 하는 작업

\* 평균 블러링

cv2.blur(영상, 커널)

- 가장 일반적인 블러링 방법으로 균일한 값을 정규화된 커널을 이용한 이미지 필터링 방법

- 커널 영역 내에서 평균 값으로 해당 픽셀을 대체함

- 주변 픽셀들의 평균값을 적용하면 픽셀 간 차이가 적어져 선명도가 떨어지므로 전체적으로 흐려짐

- 필터의 크기가 클수록 평균 블러링을 적용했을 때 선명도가 더 떨어짐

\* 가우시안 블러링

cv2.GaussianBlur(영상, 출력영상, 커널)

출력영상: (0,0)일 때 입력 영상과 크기가 같은 영상 생성

커널: 3 = 3\*3

- 가우시안 분포를 갖는 커널로 블러링하는 것

- 대상 픽셀에 가까울수록 많은 영향을 주로, 멀어질수록 적은 영향을 주기 때문에

원래의 영상과 비슷하면서도 노이즈를 제거하는 효과가 있음

+) sigmaX: 가우시안 분포에 적용되는 표준편차 / 표준편차가 클수록 블러링 강해짐

1이 가장 적당함

+) 커널사이즈 (0,0)을 넣으면 자동으로 커널사이즈 계산, 가우시안분포에서는

커널 사이즈 중요

\* 미디언 블러링

cv2.medianblur(영상, 커널)

- 커널의 픽셀 값 중 중앙값을 선택

- 소금-후추 잡음을 제거하는 효과

\* 바이레터럴 필터(Bilateral Filter)

cv2.bilateralFilter(영상, 픽셀의 거리, 시그마 컬러, 시그마 스페이스)

픽셀의 거리: -1을 입력하면 자동 결정됨

시그마 컬러: 색공간의 시그마 값

시그마 스페이스: 좌표 공간의 시그마 값(값이 클수록 떨어져 있는 픽셀들에 영향을 미침)

- 기존 블러링의 문제점

잡음을 제거하는 효과는 뛰어났지만, 경계도 흐릿하게 만드는 문제

- 개선방법

경계도 뚜렷하고 노이즈도 제거되는 효과를 모두 만족(속도가 느림)

요약)가우시안 블러: 이미지를 부드럽게 만들어주는 데 사용됨.

미디언 블러: 소금-후추 노이즈 제거에 효과적.

바이레터럴 블러: 이미지를 부드럽게 만들면서 엣지를 보존함.

평균 블러: 이미지를 간단하게 부드럽게 만들어줌.

2) 에지(edge) 검출

cv2.Canny(영상, 최소임계값, 최대임계값, 커널)

최소임계값, 최대임계값: 두 개의 경계값(Max, Min)을 지정해서 경계에 있는 영역픽셀을 찾음

- 영상에서 화소의 밝기가 급격하게 변하는 부분

- 물체의 윤곽선(경계선)이 해당

- 에지를 검출할 수 있으면 물체의 윤곽선을 알 수 있음

- "케니 엣지 검출"은 상당한 수준으로 엣지를 신뢰성있게 검출하는 방법

+) 밝기값이 급격히 변하는 부분을 엣지로 잡읍

임계값의 차이가 커질수록 선의 수가 줄어듦

문제1

웹캠 영상에서 스페이스바를 누를 때마다 필터링을 바꾸는 프로그램을 작성해보자

(단, 영상은 "일반영상", "가우시안 필터영상", "케니 필터영상" 순으로 반복 변경)

# # 7. Day7\_객체외곽선 검출

1. 모폴로지 처리

영상의 밝은 영역이나 어두운 영역을 축소 또는 확대하는 기법(+이미지의 형태를 개선함)

cv2.getStructuringElement(구조 요소의 모양, 사이즈)\_구조요소

구조 요소의 모양

cv2.MORPH\_RECT: 사각형

cv2.MORPH\_ELLIPSE: 타원형

cv2.MORPH\_CROSS: 십자형

1) 침식(erosion) 연산

cv2.erode(영상, 구조요소, 출력영상, 고정점 위치)

- 이미지를 깎아 내는 연산

- 객체 크기는 감소하고 배경은 확대(경계픽셀을 강조!)

- 작은 크기의 객체(잡음)제거 효과가 있음

- 어두운 부분의 노이즈를 제거하는 효과

2) 팽창(dilation) 연산

cv2.dilate(영상, 구조요소, 출력영상, 고정점 위치)

- 물체의 주변을 확장하는 연산

- 팽창 연산은 객체 외곽을 확대시키는 연산(경계픽셀을 제거!)

- 객체 크기가 증가되고 배경은 감소

- 객체 내부의 홀이 있다면 홀을 채울 수 있음

- 밝은 부분의 노이즈를 제거하는 효과

3) 열림(Opening)

- 팽창 연산과 침식 연산의 조합

- 침식 연산을 적용한 다음, 팽창 연산을 적용(침식 -> 팽창)

- 침식 연산으로 인해 밝은 영역이 줄어들고, 어두운 영역이 늘어남

- 객체 크기 감소를 원래대로 복구할 수 있음

- 작은 객체를 제거하고 구멍을 메우는데 사용

4) 닫힘(closing)

- 팽창 연산과 침식 연산의 조합

- 팽창 연산을 적용한 다음, 침식 연산을 적용(팽창 -> 침식)

- 어두운 영역이 줄어들고 밝은 영역이 늘어남

- 늘어난 영역을 다시 복구하기 위해 침식 연산을 적용하면 밝은 영역이 줄어들고 어두운 영역이 늘어남

- 작은 구멍을 메우고 끊어진 객체를 연결하는 데 사용

5) 그레디언트(Gradient)

- 영상에 팽창 연산을 적용한 결과에서 영상에 침식 연산을 적용한 결과를 뺌(경계가 강조된 이미지 영상을 생성)

- 엣지 검출, 객체의 윤곽 추출, 영상 분할 등에 사용

2. 레이블링

- 이진화, 모폴로지를 수행하면 객체와 배경 영역을 구분할 수 있게 됨 -> 객체 단위 분석을 통해

각 객체를 분할하여 특징을 분석하고 객체의 위치, 크기 정보, 모양 분석, ROI 추출 등이 가능

- 서로 연결되어 있는 객체 픽셀에 고유번호를 할당하여 영역 기반 모양분석, 레이블맵, 바운딩 박스

픽셀 개수, 무게 중심, 좌표 등을 반환할 수 있게 함

cv2.connectecComponents(영상, 레이블맵)

레이블맵: 픽셀 연결 관계(4방향 연결, 8방향 연결)

return: 객체 개수, 레이블맵 행렬

cv2.connectecComponentsWithStats(영상, 레이블맵)

return: 객체 개수, 레이블맵 행렬, (객체 위치, 가로세로길이, 면적 등 행렬), 무게중심정보

3. 객체의 외곽선 검출

cv2.findContours(영상, 검출모드, 외곽선 좌표 근사화 방법)

검출모드

RETR\_EXTERNAL: 객체 외부 외곽선만 검출

RETR\_LIST: 객체 외부, 내부 외곽선을 모두 검출

RETR\_CCOMP: 모든 외곽선 검출, 2단계 계층 구조를 구성

RETR\_TREE: 모든 외곽선 검출, 전체 계층 구조를 구성

외곽선 좌표 근사화 방법

CHAIN\_APPROX\_NONE: 모든 외곽선 좌표를 저장

CHAIN\_APPROX\_SIMPLE: 외곽선 중에서 수평, 수직, 대각선 성분의 끝점만 저장

- 레이블링과 함께 영상에서 객체의 정보를 검출하는 방법 중 하나

- 이진화된 영상에서 검출되며, 배경 영역과 닿아있는 픽셀을 찾아 외곽선으로 인식

- 외곽선은 객체 외부 뿐 아니라, 내부에서도 생길 수 있음

4. 외곽선 그리기

cv2.drawContours(영상, 외곽선 좌표 정보, 외곽선 인덱스, 색상, 두께)

외곽선 인덱스: -1을 지정하면 모든 외곽선을 그림

cv2.arcLength(외곽선 좌표, 폐곡선 여부) # 폐곡선: 닫힌 곡선(True)

외곽선의 길이 구하기

cv2.contourArea(외곽선 좌표, False)

외곽선의 면적 구하기(False:폐곡선이므로 면적계산실행)

cv2.boundingRect(외곽선 좌표)

바운딩 박스 구하기

문제1

milkdrop.bmp 영상을 이용하여 외곽선만 검출 후 동일한 영상을 만들기

5. 외곽선의 근사화

검출한 외곽선 정보를 분석하여 정점수가 적은 외곽선 또는 다각형으로 표현할 수 있게 만드는 것

cv2.approxPolyDP(외곽선 좌표, 근사화 정밀도 조절, 폐곡선 여부)

근사화 정밀도 조절: 입력 외곽선과 근사화된 외곽선 사이의 최대 길이. 값이 작을수록 다각형이

정확해지고, 꼭지점 수가 늘어남 (값을 계산할 거리)

cv2.isContourConvex()

contour에 볼록한 부분이 있는지 체크(있으면 True, 없으면 False)

cv2.convexHull()

볼록 외피를 계산. 주어진 점 집합을 둘러싸는 가장 작은 볼록 다각형을 반환

6. OCR(Optical Character Recognition)

- 광학 문자 인식

- 영상이나 문서에서 텍스트를 자동으로 인식하고 컴퓨터가 이해할 수 있는 텍스트 데이터로 변화하는 프로세스

- Tesseract, EasyOCR, PaddleOCR, CLOVA OCR(네이버 API), 구글 API...등(클로바가 한글에 유리)

테서렉트(Tesseract)

- 오픈 소스 OCR 라이브러리로 구글에서 개발하고 현재는 여러 커뮤니티에 의해 유지보수

- 윈도우

https://github.com/UB-Mannheim/tesseract/wiki에서

tesseract-ocr-w64-setup-5.3.3.20231005.exe를 다운로드 -> 설치중 -> Choose Components에서

Additional Script Data(download) 트리를 내림 -> Hangul Script와 Hangul Vertical Script 체크

Additional Language Data(download) 트리를 내림 -> korean 체크 -> Next 버튼

- 윈도우 환경설정

탐색기 -> "내 PC" 마우스 오른쪽 버튼 클릭 "속성" -> 창을 최대화 한 후 우측 메뉴 "고급 시스템 설정"

-> "환경 변수" 버튼 클릭 -> "시스템 변수"에서 "path"를 선택하고 "편집" 버튼을 클릭

-> "새로 만들기" 버튼 클릭 -> 테서렉트 설치 경로를 추가(C:\Program Files\Tesseract-OCR)

-> 파이참 껐다키기(윈도우 경로 적용)

문제

주어진 영상에서 명함을 추출하고 테서렉트를 사용하여 명함에 있는 문자를 인식 후 출력해보자