```
1
2  # 기본적 영상처리 기법들
3  #
4  # 실행 방법 (아나콘다 설치된 컴퓨터에서 CMD 명령어 창에서 아래와 같이 실행)
5  # conda activate ip                    ip 가상환경 활성화
6  # conda install -n ip matplotlib
7  # python imageprocessing_basic.py              프로그램 실행
8
9  import cv2, sys
10 import numpy as np
11 from matplotlib import pyplot as plt
12
13 #=============================================================================
14 # [1] OpenCV형 img 객체로 전달받은 영상데이터를
15 #     windowName이름의 창(window)을 만든 후, 그 안에 img의 내용을 출력
16 #=============================================================================
17 def show_image_window(img, windowName='IMAGE', position=[0,0]) :
18     cv2.namedWindow(windowName,cv2.WINDOW_AUTOSIZE)                 # 새로운 창 생성
19     cv2.moveWindow(windowName, position[0], position[1])           # x:position[0], y 위치로 창 이동
20     cv2.imshow(windowName, img)                                    # 화면에 창 출력
21     # end of the fn.
22
23 #=============================================================================
24 # [2] 영상파일 읽기- 영상파일 읽어서 openCV형 칼라/GRAY img 객체(인스턴스)를 리턴함
25 #=============================================================================
26 def read_image(showImage = True):
27     print('파일 크기가 크면 실행시 시간이 걸릴 수 있습니다. [강제종료는 컨트롤키와 c키를 동시에 누르세요]')
28     print('파일명에 공백문자가 안들어가도록 하고, 파일확장자도 반드시 붙이세요₩n')
29     fileName = input('영상파일명 입력:')                            # 그림 파일명 입력
30     print('입력한 파일명은 [{}] 입니다'.format(fileName))
31     #fileName = 'sample1.png'
32
33     inputImg = cv2.imread(fileName,cv2.IMREAD_UNCHANGED)               # 영상데이터를 읽어 img 객체에 저장
34
35     if inputImg is None:                                               # 제대로 불러오지 못했으면
36         sys.exit("영상 파일 열기 오류")                               # 프로그램 종료
37
38         # 칼라 채널 수에 따라 GRAY 영상으로 변환
39     heightSize, widthSize, channelSize = inputImg.shape
40     if channelSize == 3:
41         grayImg = cv2.cvtColor(inputImg, cv2.COLOR_BGR2GRAY)
42     elif channelSize == 4:
43         grayImg = cv2.cvtColor(inputImg, cv2.COLOR_BGRA2GRAY)
44         inputImg = cv2.cvtColor(inputImg, cv2.COLOR_BGRA2BGR)
45     else:
46         grayImg = inputImg.copy()
47         inputImg = cv2.cvtColor(inputImg, cv2.COLOR_GRAY2BGR)
48
49     if showImage :
50         nameString = 'INPUT IMG' + str(inputImg.shape)                          # 창이름을 파일명과 img구조로 문자열로
설정
51         show_image_window(inputImg, windowName=nameString, position=[0,0])   # 영상 출력을 위해 위에서 만든 출력용 함
수 호출
52         show_image_window(grayImg, windowName='GRAY', position=[0,500])   # 영상 출력을 위해 위에서 만든 출력용 함수
호출
53
54     return inputImg, grayImg
55
56 #=============================================================================
57 # [3] Convolution
58 #=============================================================================
59 def filtering_by_convolution(colorImg):
60     global grayImg
61     #-----------------------------------------
62     # [3-1] 평균값 커널에 의한 블러링
63     #-----------------------------------------
64     kernel = np.ones((7, 5), np.float32) / 25.0
65
66         # 입력영상에 대해 블러링 수행
67     blurredImg = cv2.filter2D(colorImg, -1, kernel)
68     show_image_window(blurredImg, windowName='BLURRED', position=[500,0])
69
70     #-----------------------------------------
71     # [3-2] 엠보싱
72     #-----------------------------------------
```

```
    kernel2 = np.array([[-1, 0, 0],
                        [0, 0, 0],
                        [0, 0, 1]])

        # Gray영상에 대해 블러링 수행
    embossingImg = cv2.filter2D(grayImg, -1, kernel2)
    show_image_window(embossingImg, windowName='EMBOSSING', position=[500,500])

    #----------------------------------------
    # [3-3] 샤프닝
    #----------------------------------------
    kernel3 = sharpening_2 = np.array([[-1, -1, -1, -1, -1],
                        [-1, 2, 2, 2, -1],
                        [-1, 2, 9, 2, -1],
                        [-1, 2, 2, 2, -1],
                        [-1, -1, -1, -1, -1]]) / 9.0

        # Gray영상에 대해 블러링 수행
    sharpeningImg = cv2.filter2D(grayImg, -1, kernel3)
    show_image_window(sharpeningImg, windowName='SHARPENING', position=[1000,500])

#=============================================================================
# [4] 트랙바 생성하고, 트랙바에서 설정하는 두 경계값에 대한 에지 영상 출력
#=============================================================================
def find_edge():
    global grayImg, edgeImg

    show_image_window(grayImg, windowName='GRAY', position=[0,500])     # Gray 영상 출력

        # 트랙바 생성
    cv2.namedWindow('CANNY EDGE',cv2.WINDOW_AUTOSIZE)
    cv2.moveWindow('CANNY EDGE', 1000,0)
    cv2.createTrackbar('lowThreshold', 'CANNY EDGE', 0, 255, onPosChange)
    cv2.createTrackbar('highThreshold', 'CANNY EDGE', 0, 255, onPosChange)
    cv2.createTrackbar('gaussianBlurring', 'CANNY EDGE', 0, 1, onPosChange)
        # 트랙바의 초기 변수 값 설정
    cv2.setTrackbarPos('lowThreshold', 'CANNY EDGE', 50)
    cv2.setTrackbarPos('highThreshold', 'CANNY EDGE', 200)
    cv2.setTrackbarPos('gaussianBlurring', 'CANNY EDGE', 0)

    while True:
        cv2.imshow('CANNY EDGE', edgeImg)

        keyInChar = cv2.waitKey(10)
        if  keyInChar == ord("q") :
            cv2.destroyWindow('GRAY')
            cv2.destroyWindow('CANNY EDGE')
            break

#=============================================================================
# [4-2] 트랙바 콜백함수 - 어떤 이벤트가 있을 때 자동 호출되는 함수
#=============================================================================
def onPosChange(posValue):
    global grayImg, edgeImg
        # 트랙바에서
    lowVal = cv2.getTrackbarPos('lowThreshold', 'CANNY EDGE')
    highVal = cv2.getTrackbarPos('highThreshold', 'CANNY EDGE')
    blurringVal = cv2.getTrackbarPos('gaussianBlurring', 'CANNY EDGE')
        # CANNY 에지 영상 구하기
    if blurringVal == 1:
        blurredGrayImg = cv2.GaussianBlur(grayImg, (5,5), 0)              # 5x5 크기의 gaussian blurr 필터만 적용
        edgeImg = cv2.Canny(blurredGrayImg, lowVal, highVal)
    else :
        edgeImg = cv2.Canny(grayImg, lowVal, highVal)
    print('lowVal:{}, highVal:{}, blurring여부:{}, 에지 영상 정보: {}'.W
        format(lowVal, highVal, blurringVal, edgeImg.shape))

#=============================================================================
# [5] 이진화
#=============================================================================
def binarization():
    global grayImg

    ret, binImg = cv2.threshold(grayImg, 127, 255, cv2.THRESH_BINARY)
    gauBinImg = cv2.adaptiveThreshold(grayImg,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,W
```

```
            cv2.THRESH_BINARY,15,2)
    show_image_window(binImg, windowName='BINARIZED', position=[300,0])
    #show_image_window(gauBinImg, windowName='GAUS-BINARIZED', position=[300,500])

    return binImg
#=============================================================================
# MAIN: 영상 파일 입출력과 기본 처리
#=============================================================================
if __name__ == '__main__' :                                     # 직접 실행한 경우이면
    global grayImg, edgeImg
    while True:
            # 영상파일 읽기
        colorImg, grayImg  = read_image(True)

        '''
            # [1] Convolution을 이용한 다양한 영상 처리
        filtering_by_convolution(colorImg)
        cv2.waitKey(0)

            # [2] TrackBar를 이용하여 경계값에 따른 Canny 에지 구하기
        find_edge()
        cv2.waitKey(0)
        '''

            # [3] 이진화
        binaryImg = binarization()

        kernel = np.ones((3,3),np.uint8)                # 모폴로지 연산의 대상영역을 커널로 설정

            # [4-1] Morphology - Erosion
        erodedImg = cv2.erode(binaryImg, kernel, iterations = 1)
        show_image_window(erodedImg, windowName='ERODED', position=[800,0])
        erodedImg2 = cv2.erode(binaryImg, kernel, iterations = 2)
        show_image_window(erodedImg2, windowName='ERODED-2Times', position=[800,500])
        cv2.waitKey(0)

            # [4-2] Morphology - Dilation
        dilatedImg = cv2.dilate(binaryImg, kernel, iterations = 1)
        show_image_window(dilatedImg, windowName='DILATED', position=[800,0])
        dilatedImg3 = cv2.dilate(binaryImg, kernel, iterations = 2)
        show_image_window(dilatedImg3, windowName='DILATED-3Times', position=[800,500])
        cv2.waitKey(0)

            # [4-3] Morphology - Opening
        openingImg = cv2.morphologyEx(binaryImg, cv2.MORPH_OPEN, kernel, iterations = 3)
        show_image_window(openingImg, windowName='OPENING-3Times', position=[800,0])
        cv2.waitKey(0)

            # [4-4] Morphology - Closing
        closingImg = cv2.morphologyEx(binaryImg, cv2.MORPH_CLOSE, kernel, iterations = 3)
        show_image_window(closingImg, windowName='CLOSING-3Times', position=[800,500])
        cv2.waitKey(0)

            # [4-5] Morphology - Gradient
        gradientImg = cv2.morphologyEx(binaryImg, cv2.MORPH_GRADIENT, kernel, iterations = 2)
        show_image_window(gradientImg, windowName='GRADIENT-2Times', position=[800,0])
        cv2.waitKey(0)

        cv2.destroyAllWindows()
        show_image_window(colorImg, windowName='INPUT', position=[0,0])

            # [5-1] Geometric - Scaling
        shrinkedImg = cv2.resize(colorImg, None, fx=0.5, fy=0.5, interpolation = cv2.INTER_AREA)
        show_image_window(shrinkedImg, windowName='SHRINKED', position=[800,0])
        cv2.waitKey(0)

            # [5-2] Geometric - Translation
        rows, cols, ch = colorImg.shape
        Mask = np.float32([[1,0,100],[0,1,50]])         # 변환 전/후의 3점의 좌표 세트 입력
        translatedImg = cv2.warpAffine(colorImg, Mask, (cols,rows))  # cols, row, 영상의 크기
        show_image_window(translatedImg, windowName='TRANSLATED', position=[800,500])
        cv2.waitKey(0)

            # [5-3] Geometric - Rotation
        Mask = cv2.getRotationMatrix2D(((cols-1)/2.0,(rows-1)/2.0), 90, 1)   # 회전각도 (90도)
```

```
        rotatedImg = cv2.warpAffine(colorImg, Mask, (cols,rows))
        show_image_window(rotatedImg, windowName='ROTATEDED', position=[800,000])
        cv2.waitKey(0)

            # [5-4] Geometric - Affine Tr.
        pts1 = np.float32([[50,50],[200,50],[50,200]])     # 입력 영상의 3개 점
        pts2 = np.float32([[10,100],[200,50],[100,250]])   # 타겟 영상의 대응점
        M = cv2.getAffineTransform(pts1,pts2)
        affineImg = cv2.warpAffine(colorImg, M, (cols,rows))
        show_image_window(affineImg, windowName='AFFINE-TR', position=[800,500])
        cv2.waitKey(0)

            # [5-5-1] Geometric - Perspective Tr.
        pts1 = np.float32([[214,49],[272,57],[32,434],[475,505]])    # 4개 점
        pts2 = np.float32([[0,0],[600,0],[0,600],[600, 600]])
        M = cv2.getPerspectiveTransform(pts1,pts2)
        perspectiveImg = cv2.warpPerspective(colorImg, M, (600,600))
        show_image_window(perspectiveImg, windowName='PERSPECTIVE-TR1', position=[800,0])
            # [5-5-2] Geometric - Perspective Tr.
        pts1 = np.float32([[32,434],[475,505],[214,49],[272,57]])    # 4개 점
        pts2 = np.float32([[0,0],[600,0],[0,600],[600, 600]])
        M = cv2.getPerspectiveTransform(pts1,pts2)
        perspectiveImg = cv2.warpPerspective(colorImg, M, (600,600))
        show_image_window(perspectiveImg, windowName='PERSPECTIVE-TR2', position=[800,500])

        keyInChar = cv2.waitKey(0)                                      # 키보드 입력 무한정 기다림

        if  keyInChar == ord("q") :
            cv2.destroyAllWindows()                                     # OPENCV를 이용해서 생성한 창들 모두 종료
            break

```