영상정보처리 13주차 과제 템플리트

이름: 김경민
학번: 32200327 입력 이미지: 자유

구글 드라이브 마우팅 및 작업 경로로 이동

- 다음 쉘에 필요한 작업을 하시오.

```
from google.colab import drive
drive.mount('/gdrive')
%cd /gdrive/MyW Drive/Classroom/[22-1 영상정보처리] 2000004793-2022-1/ImageProcClass/Notebook-week1
!pwd
```

```
Drive already mounted at /gdrive; to attempt to forcibly remount, call drive.mount("/gdrive",
/gdrive/My Drive/Classroom/[22-1 영상정보처리] 2000004793-2022-1/ImageProcClass/Notebook-week
/gdrive/My Drive/Classroom/[22-1 영상정보처리] 2000004793-2022-1/ImageProcClass/Notebook-week
```

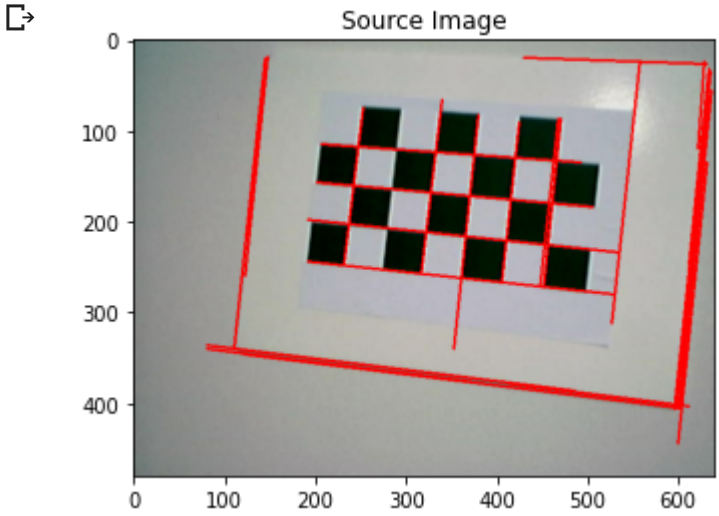
문제 1

수업 시간에 배운 다음의 방법들을 함수화 하고, 자유롭게 선택된 이미지를 이용하여 테스트 하시오.

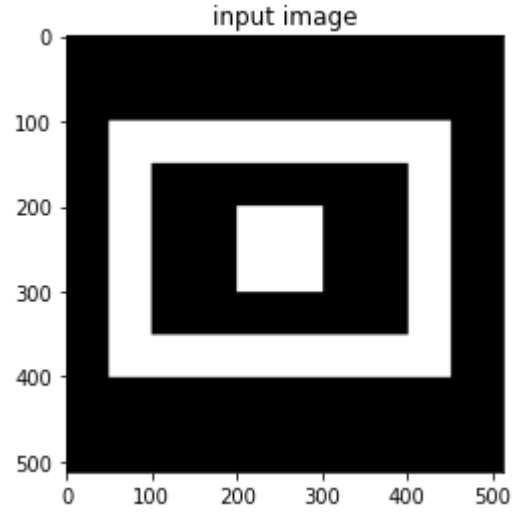
1. HoughLinesP 를 이용한 직선 성분 찾기
2. contour 관련 함수들
    - 하나의 세그먼트에 대한 외곽선 그리기
    - 하나의 이미지에 있는 모든 세그먼트에 대한 외곽선 그리기
    - 하나의 세그먼트에 대한 외곽선 그리기 영역 계산하기
    - 하나의 컨투어에 대한 centroid 찾기
    - 하나의 컨투어에 대한 bounding rectangle 찾기
    - 하나의 컨투어에 대한 rotated rectangle 찾기
    - 하나의 컨투어에 대한 convex hull 찾기
    - 하나의 컨투어에 대한 convexity defects 찾기

```
def show_with_matplotlib_jh(img, title):
    if img is None:
      print("show_with_matplotlib: Could not read the image.")
      return

    if img.shape[2] != 3:
      print()
      print("show_with_matplotlib: given image does not contains 3 channels")
```

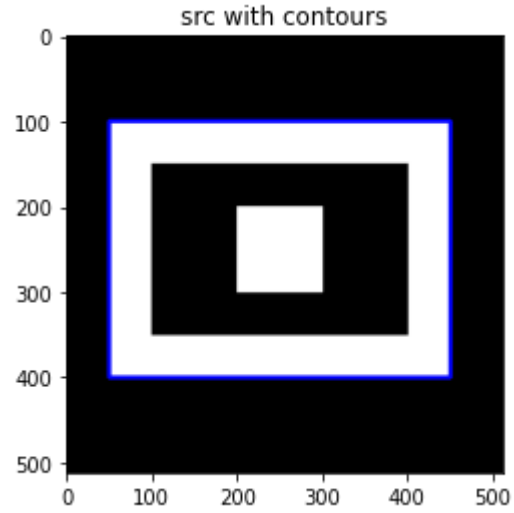
```
        return

    # Convert BGR image to RGB:
    img_RGB = img[:, :, ::-1]

    # Show the image using matplotlib:
    plt.imshow(img_RGB)
    plt.title(title)
    plt.show()

# matplot grayscale display
def show_with_matplotlib_gray_jh(img, title):
   if img is None:
      print("show_with_matplotlib_gray: Could not read the image.")
      return

   if img.ndim > 2:
      print()
      print("show_with_matplotlib: given image has more than 2 dim")
      return

   plt.imshow(img, cmap="gray")
   plt.title(title)
   plt.show()


import cv2
from google.colab.patches import cv2_imshow
import numpy as np
import matplotlib.pyplot as plt

image_path = "../Dongkeun-OpenCV-ImgData/chessBoard.jpg"
#image_path2 = '../Dongkeun-OpenCV-ImgData/SegmentTest.jpg'
image_path2 = '../Dongkeun-OpenCV-ImgData/testShapes1.jpg'
image_path3 = '../Dongkeun-OpenCV-ImgData/hand.jpg'



img = cv2.imread(image_path) # line 찾은 결과를 원본에 컬러로 표시해 나타냄
gray_img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) # HoughLinesP 찾을 땐, gray 이미지


def find_Line(src, gray_src):
  edges = cv2.Canny(gray_src, 10, 30)
  lines = cv2.HoughLinesP(edges, rho=1, theta=np.pi/180.0, threshold=100,minLineLength=80,maxLineGa
  # maxLineGap : 선과 선 사이의 최대 허용간격(이 값보다 작으면 reject) # 이 값 너무 작아지면 선 추출
  # minLineLength : 선의 최소 길이(이 값보다 작으면 reject) # 이 값 너무 커지면 짧은 선들은 추출X

  for line in lines:
    x1, y1, x2, y2   = line[0] # lines에서 각 값들 추출
    cv2.line(src,(x1,y1),(x2,y2),(0,0,255),2)


find_Line(img,gray_img)
show_with_matplotlib_jh(img, "Source Image")
```

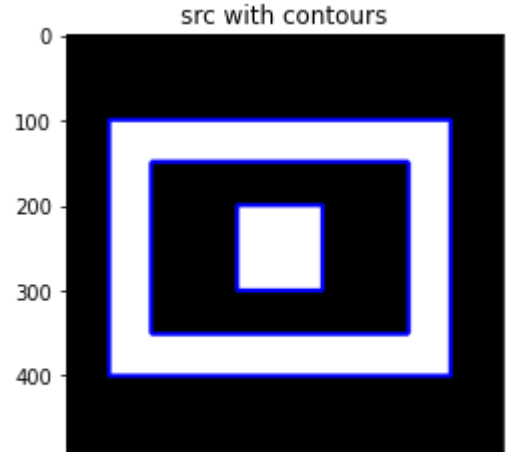



```
# 하나의 외곽선 컨투어만 찾는 함수
def findNdrawContour_external(src,gray_src):
  contours, hierarchy = cv2.findContours(gray_src, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) # 외
  print(len(contours))
  return contours
  #cv2.drawContours(src, contours, 0, (255,0,0), 3)


# 모든 컨투어 외곽선 찾는 함수
def findNdrawContour_all(src,gray_src):
  contours, hierarchy = cv2.findContours(gray_src, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE) # contou
  #print(len(contours))
  return contours

 # for cnt in contours:
  #  cv2.drawContours(src, [cnt], 0, (255, 0, 0), 3)  # 모든 외곽선 찾았기 때문에 point array list0

def draw_Contour(src,contours):
  for cnt in contours:
    cv2.drawContours(src, [cnt], 0, (255, 0, 0), 3)

image = np.zeros(shape=(512,512,3), dtype=np.uint8)
cv2.rectangle(image, (50, 100), (450, 400), (255, 255, 255), -1)
cv2.rectangle(image, (100, 150), (400, 350), (0, 0, 0), -1)
cv2.rectangle(image, (200, 200), (300, 300), (255, 255, 255), -1)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
show_with_matplotlib_gray_jh(gray_image, "input image")

contours = findNdrawContour_external(image,gray_image)
draw_Contour(image,contours)
show_with_matplotlib_jh(image, "src with contours")

contours = findNdrawContour_all(image,gray_image)
draw_Contour(image,contours)
show_with_matplotlib_jh(image, "src with contours")
```




```
1
```

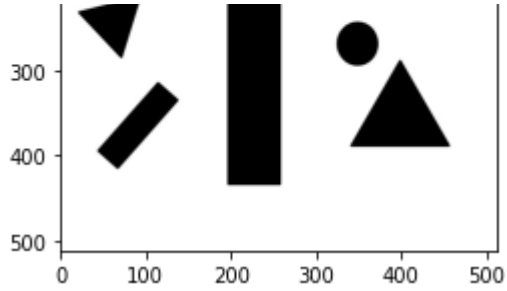


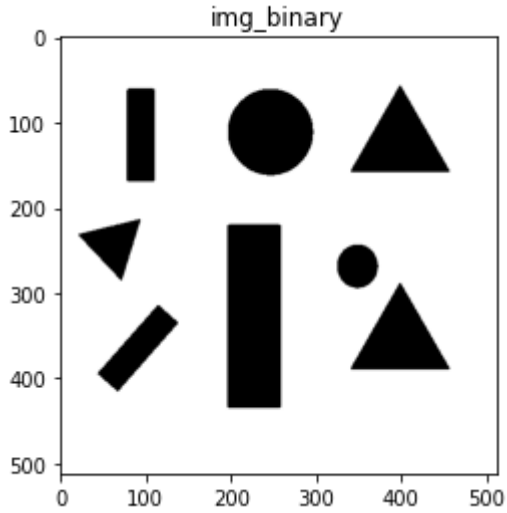



```
# 외곽선 그리기 영역 계산 함수
def calc_area(src,gray_src):
  ret, img_binary = cv2.threshold(gray_src, 20, 255, cv2.THRESH_BINARY)
  show_with_matplotlib_gray_jh(img_binary, "img_binary")

  contours = findNdrawContour_all(src,img_binary)
  print(len(contours))

  draw_Contour(img_color,contours)


  # 외곽선 그리기 영역 계산
  for cnt in contours:
```

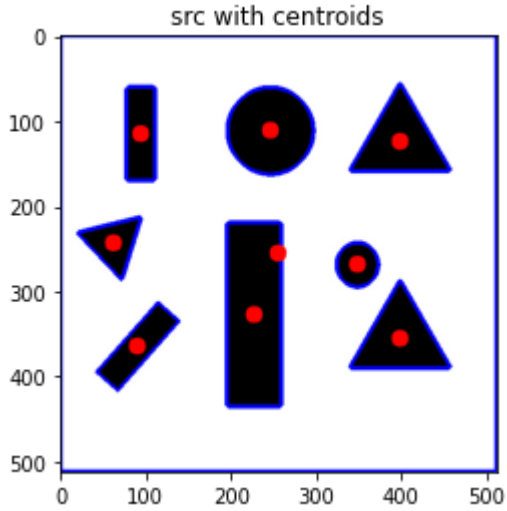
```
        area = cv2.contourArea(cnt)
        #print(area)

      return area

    # centroid 찾는 함수
    def find_centroid(src,gray_src):
      ret, img_binary = cv2.threshold(gray_src, 20, 255, cv2.THRESH_BINARY)
      #show_with_matplotlib_gray_jh(img_binary, "img_binary")

      contours = findNdrawContour_all(src,img_binary)

      for cnt in contours:
        M = cv2.moments(cnt)

        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])

        cv2.circle(img_color, (cx, cy), 10, (0,0,255), -1)
        #draw_Contour(src,contours)


    def find_boundingRect(src,gray_src):
      ret, img_binary = cv2.threshold(gray_src, 20, 255, cv2.THRESH_BINARY)
      contours = findNdrawContour_all(src,img_binary)

      for cnt in contours:
        x, y, w, h = cv2.boundingRect(cnt)
        cv2.rectangle(img_color, (x, y), (x + w, y + h), (0, 255, 0), 2)


    def find_rotatedRect(src,gray_src):
      ret, img_binary = cv2.threshold(gray_src, 20, 255, cv2.THRESH_BINARY)
      contours = findNdrawContour_all(src,img_binary)

      for cnt in contours:
        rect = cv2.minAreaRect(cnt)
        box = cv2.boxPoints(rect) # 최적화 된 사각형의 꼭짓점 값을 반환
        box = np.int0(box)

        cv2.drawContours(src,[box],0,(0,0,255),2)


    img_color = cv2.imread(image_path2)
    img_gray = cv2.imread(image_path2, cv2.IMREAD_GRAYSCALE)
    show_with_matplotlib_jh(img_color, "img_color")
    show_with_matplotlib_gray_jh(img_gray, "img_gray")


    area = calc_area(img_color,img_gray)
    print(area)
    #show_with_matplotlib_jh(img_color, "src with contours")

    find_centroid(img_color,img_gray)
    show_with_matplotlib_jh(img_color, "src with centroids")
```

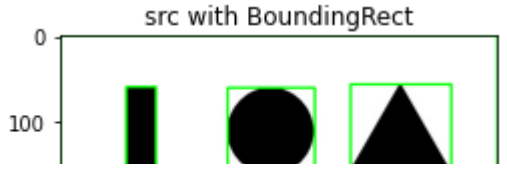
```
img_color = cv2.imread(image_path2)
img_gray = cv2.imread(image_path2, cv2.IMREAD_GRAYSCALE)

find_boundingRect(img_color,img_gray)
show_with_matplotlib_jh(img_color, "src with BoundingRect")

find_rotatedRect(img_color,img_gray)
show_with_matplotlib_jh(img_color, "src with RotatedRect")
```





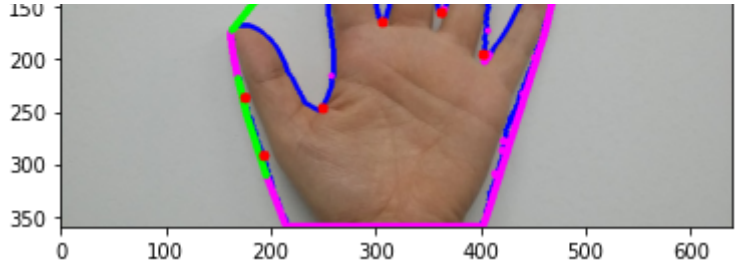

```
9
261121.0
```



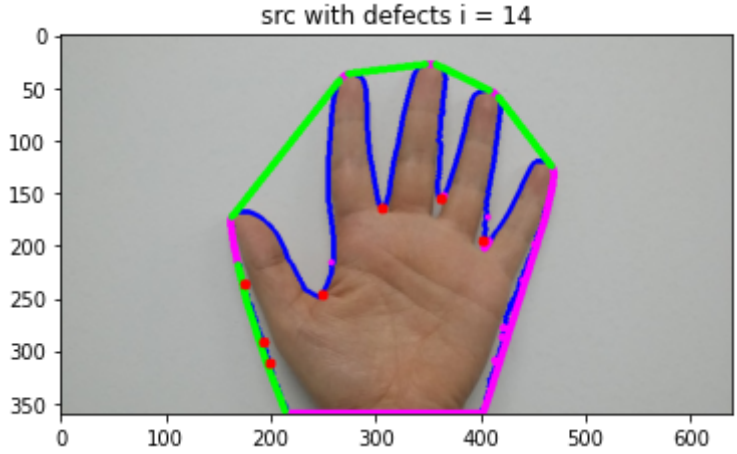



```
def find_convexHull(src,gray_src):

  ret, img_binary = cv2.threshold(gray_src, 150, 255, cv2.THRESH_BINARY_INV)

  show_with_matplotlib_gray_jh(img_binary, "img_binary")
  contours = findNdrawContour_all(src,img_binary)
  draw_Contour(src,contours)

  for cnt in contours:
    hull = cv2.convexHull(cnt) # 컨투어를 포함하는 다각형 그림
    cv2.drawContours(src, [hull], 0, (255, 0, 255), 5)
```

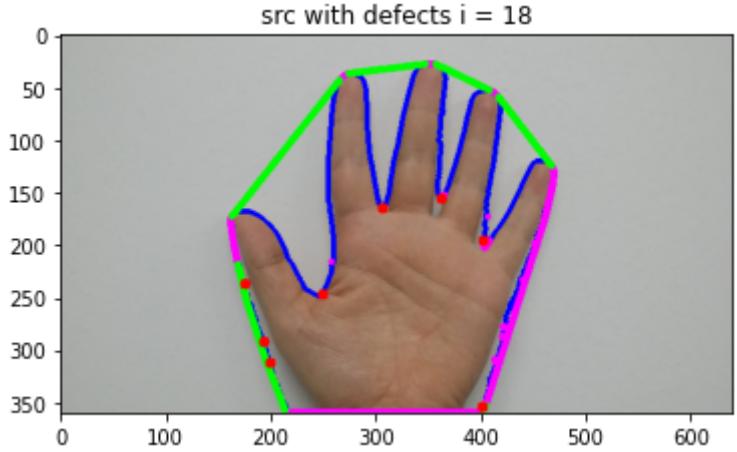
```
def find_convexityDefects (src,gray_src):

  ret, img_binary = cv2.threshold(gray_src, 150, 255, cv2.THRESH_BINARY_INV)
  contours = findNdrawContour_all(src,img_binary)

  for cnt in contours:
    hull = cv2.convexHull(cnt, returnPoints=False)
    defects = cv2.convexityDefects(cnt, hull) # Hull-contour 사이 거리가 가장 먼 부분을 찾음 # None
    #print(defects)

    if not defects is None: # None 값이 있으면 오류가 나기 때문에 중요!
      for i in range(defects.shape[0]):
        s,e,f,d = defects[i,0] # 값 추출해 그리는 과정

        start = tuple(cnt[s][0])
        end = tuple(cnt[e][0])
        far = tuple(cnt[f][0])
        print("distance =", d)

        if d > 500:
            cv2.line(src, start, end, [0, 255, 0], 5)
            cv2.circle(src, far, 5, [0,0,255], -1)
            title = "src with defects i = " + str(i)
            show_with_matplotlib_jh(src, title)


img_color = cv2.imread(image_path3)
img_gray = cv2.imread(image_path3, cv2.IMREAD_GRAYSCALE)
show_with_matplotlib_jh(img_color, "img_color")
show_with_matplotlib_gray_jh(img_gray, "img_gray")

find_convexHull(img_color,img_gray)
show_with_matplotlib_jh(img_color, "src with convex hull")

find_convexityDefects(img_color,img_gray)
```



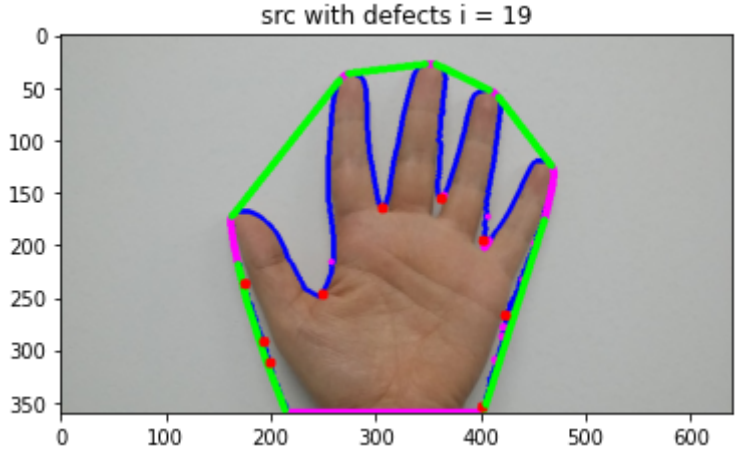
```
distance = 860
```




```
distance = 181
distance = 256
distance = 114
distance = 523
```




```
distance = 1963
```




```
distance = 538
```