영상정보처리 13주차 과제 템플리트

이름: 이현정
학번: 32203660

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/MyDrive/ImageProcClass/Notebook-Week13
```

```
Drive already mounted at /gdrive; to attempt to forcibly remount, call drive.mount("/gdrive",
/gdrive/MyDrive/ImageProcClass/Notebook-Week13
```

문제 1

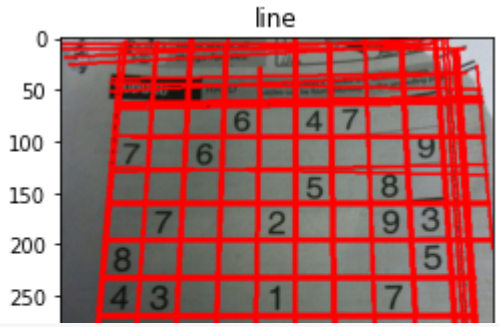
수업 시간에 배운 다음의 방법들을 함수화 하고, 자유롭게 선택된 이미지를 이용하여 테스트 하시오.

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

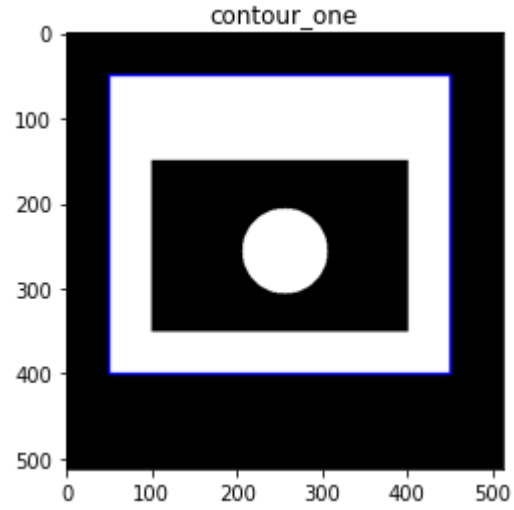
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import cm

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()
```

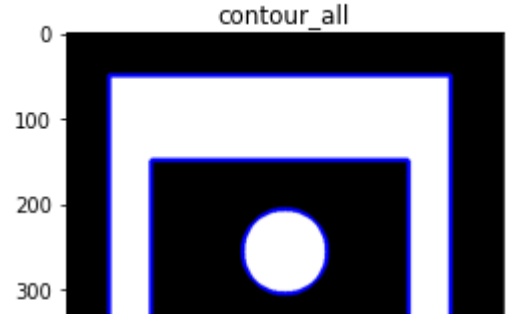
```
src = cv2.imread('../Dongkeun-OpenCV-ImgData/dave.jpg')
gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
#show_with_matplotlib_gray_jh(gray, 'gray')

# HoughLinesP 를 이용한 직선 성분 찾기
def find_line(src, min, max):
  edges = cv2.Canny(src, 50, 100) # 바이너리 이미지로 변경
  lines = cv2.HoughLinesP(edges, rho=1, theta=np.pi/180.0, threshold=100, minLineLength=min, maxLin
  for line in lines:
    x1, y1, x2, y2   = line[0]
    cv2.line(src,(x1,y1),(x2,y2),(0,0,255),2)
  if (src.shape[2] == 3) :
    show_with_matplotlib_jh(src, 'line')
  else :
    show_with_matplotlib_gray_jh(src, 'line')
# 실행
find_line(src, 10, 100) # 긴 선이 많은 이미지로 maxLineGap를 크게 줌
```




```
#src = cv2.imread('../Dongkeun-OpenCV-ImgData/hand.jpg')
def imake():
  src = np.zeros(shape=(512,512,3), dtype=np.uint8)
  cv2.rectangle(src, (50, 50), (450, 400), (255, 255, 255), -1)
  cv2.rectangle(src, (100, 150), (400, 350), (0, 0, 0), -1)
  cv2.circle(src, (256, 256), 50, (255, 255, 255), -1)
  return src
# gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY) bgr 이미지 함수 내부에서 처리


#하나의 세그먼트에 대한 외곽선 그리기
def contour_one(src, min, max):
  gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY) # bgr 이미지 -> grayscale
  ret, binary = cv2.threshold(gray, min, max, cv2.THRESH_BINARY) # 바이너리 이미지로
  contours, hierarchy = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
  cv2.drawContours(src, contours, -1, (255,0,0), 2)
  show_with_matplotlib_jh(src, "contour_one")

#하나의 이미지에 있는 모든 세그먼트에 대한 외곽선 그리기
def contour_all(src, min, max):
  gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY) # bgr 이미지 -> grayscale
  ret, binary = cv2.threshold(gray, min, max, cv2.THRESH_BINARY) # 바이너리 이미지로
  contours, hierarchy = cv2.findContours(binary, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
  for cnt in contours:
    cv2.drawContours(src, [cnt], 0, (255,0,0), 3)
  show_with_matplotlib_jh(src, "contour_all")

#하나의 세그먼트에 대한 외곽선 그리기 영역 계산하기
def area_one(src, min, max):
  gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY) # bgr 이미지 -> grayscale
  ret, binary = cv2.threshold(gray, min, max, cv2.THRESH_BINARY) # 바이너리 이미지로
  contours, hierarchy = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
  for cnt in contours:
    area = cv2.contourArea(cnt)
    print('contour_one 외곽선 영역 크기 : ', area)

# 실행
contour_one(imake(),150,255)
area_one(imake(), 150, 255)

contour_all(imake(),150,255)
```

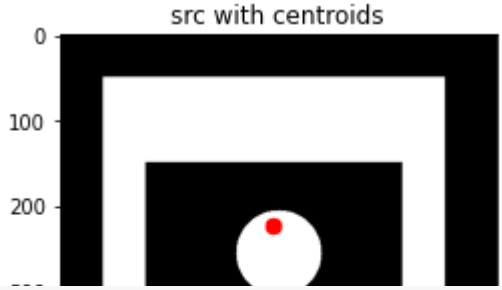


contour_one 외곽선 영역 크기 : 140000.0

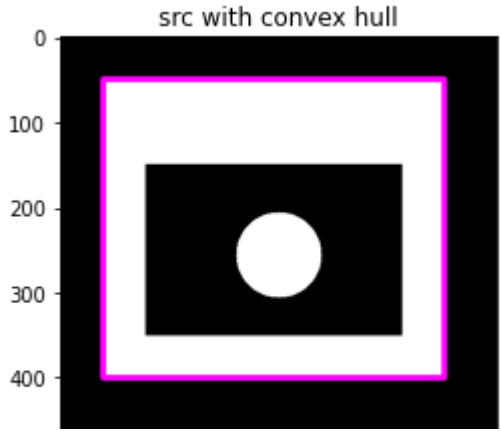



```
#하나의 컨투어에 대한 centroid 찾기
def centroid(src) :
  img_gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
  ret, img_binary = cv2.threshold(img_gray, 127, 255, 0)
  contours, hierarchy = cv2.findContours(img_binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
  # 하나 컨투어 설정 : cv2.RETR_EXTERNAL

  for cnt in contours:
    M = cv2.moments(cnt)
    # 무게중심 위치
    cx = int(M['m10']/M['m00'])
    cy = int(M['m01']/M['m00'])
    cv2.circle(src, (cx, cy), 10, (0,0,255), -1)
  show_with_matplotlib_jh(src, "src with centroids")


#하나의 컨투어에 대한 bounding rectangle 찾기
def rect(src) :
  img_gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
  ret, img_binary = cv2.threshold(img_gray, 127, 255, 0)
  contours, hierarchy = cv2.findContours(img_binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
  # 하나 컨투어 설정 : cv2.RETR_EXTERNAL

  for cnt in contours:
      x, y, w, h = cv2.boundingRect(cnt) # 박스 위치
      cv2.rectangle(src, (x, y), (x + w, y + h), (0, 255, 0), 3)
  show_with_matplotlib_jh(src, "src with B.R.")
```

```
#하나의 컨투어에 대한 rotated rectangle 찾기
def r_rect(src) :
  img_gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
  ret, img_binary = cv2.threshold(img_gray, 127, 255, 0)
  contours, hierarchy = cv2.findContours(img_binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
  # 하나 컨투어 설정 : cv2.RETR_EXTERNAL

  for cnt in contours:
      rect = cv2.minAreaRect(cnt) # 박스를 돌리면서 최적 위치
      box = cv2.boxPoints(rect)
      box = np.int0(box)
      cv2.drawContours(src,[box],0,(255,0,0),3)

  show_with_matplotlib_jh(src, "src with Min. Rect")

centroid(imake())
rect(imake())
r_rect(imake())
```

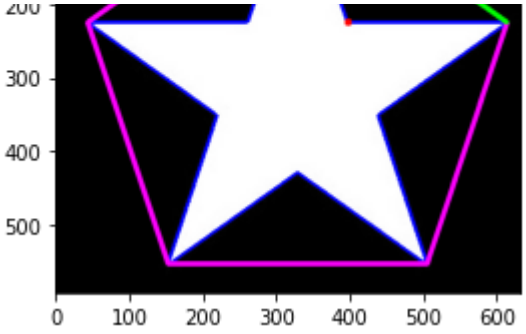



```
#하나의 컨투어에 대한 convex hull 찾기
def hull(src):
  gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
  ret, img_binary = cv2.threshold(gray, 127, 255, 0)
  contours, hierarchy = cv2.findContours(img_binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

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

  for cnt in contours:
      hull = cv2.convexHull(cnt)
      cv2.drawContours(src, [hull], 0, (255, 0, 255), 5)

  show_with_matplotlib_jh(src, "src with convex hull")


# imake() : 이미지 생성 함수
hull(imake())
```

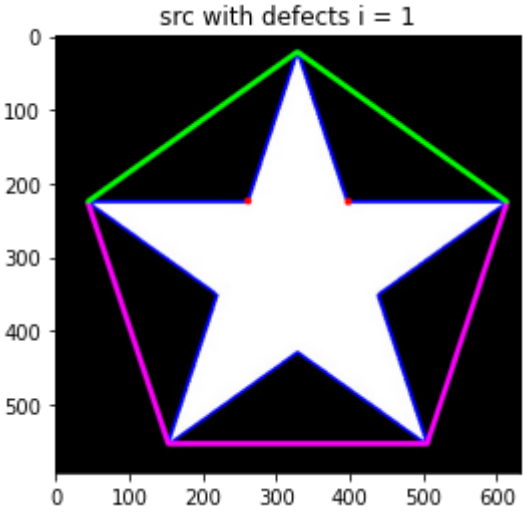



```
#하나의 컨투어에 대한 convexity defects 찾기
def defects(src):
  gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
  ret, img_binary = cv2.threshold(gray, 127, 255, 0)
  contours, hierarchy = cv2.findContours(img_binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

  for cnt in contours:
      cv2.drawContours(src, [cnt], 0, (255, 0, 0), 3)  # 외곽선 : 파란색

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

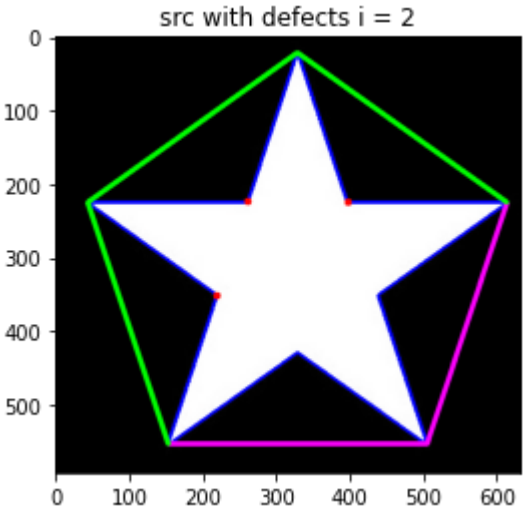
```
  icnt = 0
  # for each contour, get hull and defects
  print("no of contour = ", len(contours))
  for cnt in contours:
      hull = cv2.convexHull(cnt, returnPoints = False)
      defects = cv2.convexityDefects(cnt, hull)
      if icnt == 0:
        print("cnt.shape =", cnt.shape, " hull.shape = ", hull.shape)
        thull = hull;
        print("max value in hull = ", max(thull.flatten()), "min value = ", min(thull.flatten()))
        print("defects[0].shape = ", defects[0].shape, "defects[0] = ", defects[0])

      icnt = icnt + 1

      print("for new defects, defects.shape = ", defects.shape, " defects.shape[0] = ", defects.sha
      for i in range(defects.shape[0]):
          s,e,f,d = defects[i,0]

          start = tuple(cnt[s][0])
          end = tuple(cnt[e][0])
          far = tuple(cnt[f][0])
          if i == 0:
            print("start  ", start, " end = ", end, " far = ", far)

          print("s = ", s, " e = ", e, " f = ", f, " d = ", d)


          print("distance =", d)

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


image_path = '../Dongkeun-OpenCV-ImgData/star.png'
src = cv2.imread(image_path)
defects(src)
```



s = 0 e = 135 f = 132 d = 32096
distance = 32096



s = 136 e = 371 f = 240 d = 32301
distance = 32301



s = 371 e = 577 f = 471 d = 31744
distance = 31744

✓ 1초 오전 12:14에 완료됨