Contour Area

contourArea() 함수

2024년 1학기

서경대학교 김진헌

1_contourArea.py
2_contourArea_compare.py
3_contours_sort_size.py

Structural Analysis and Shape Descriptors

차례

- 1. 면적 함수: contourArea()
- 2. 그래픽 도형 면적-Rectangle
- 3. 그래픽 도형 면적 - Circle
- 4. 최대/최소 크기의 객체 구하기
- 5. 검출되는 객체를 크기 순으로 소팅

1_contourArea.py

2_contourArea_compare.py

3_contours_sort_size.py

1. 면적 함수: contourArea()

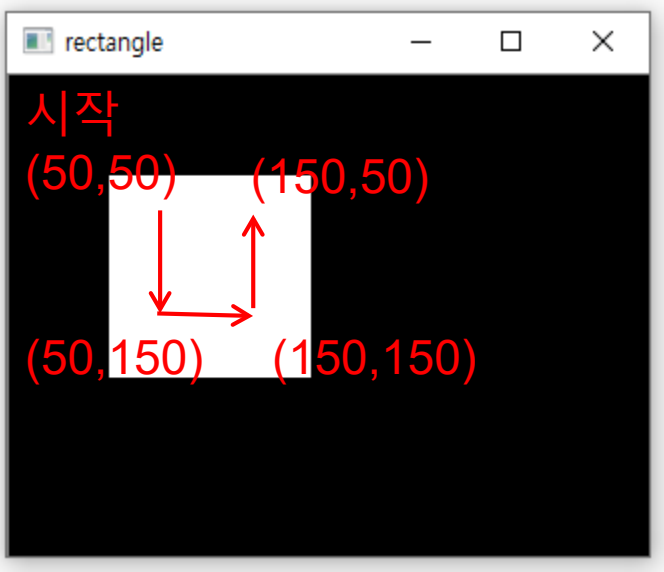
https://docs.opencv.org/4.1.0/d3/dc0/group__imgproc__shape.html#ga8ce13c24081bbc7151e9326f412190f1

- double contourArea(InputArray **contour**, bool **oriented**=false) True로 하면 부호 있는 상태로 반환
  - **사례 : double a=contourArea(contours[i], false);**
    - **i번째 contour의 면적을 계산할 경우**
- **주어진 contour의 면적을 계산하여 반환한다.**
  - **contour** - Input vector of 2D points (contour vertices), stored in std::vector or Mat.
  - **oriented** - Oriented area flag.
    - False(=default)로 지정하면 절대값 면적을 반환한다.
    - True로 지정하면 + 혹은 - 부호를 가진 면적을 반환한다.
      - **Hole이면 +면적 반환**: 시계방향으로 배열된 윤곽선, 즉 hole이면 양의 부호(+)를 면적을 반환하고,
      - **Contour이면 -면적 반환**: 컨투어가 반시계방향으로 배열된 윤곽선이라면 음의 부호(-)의 면적을 반환한다. → contour

검은 배경, 하얀 객체

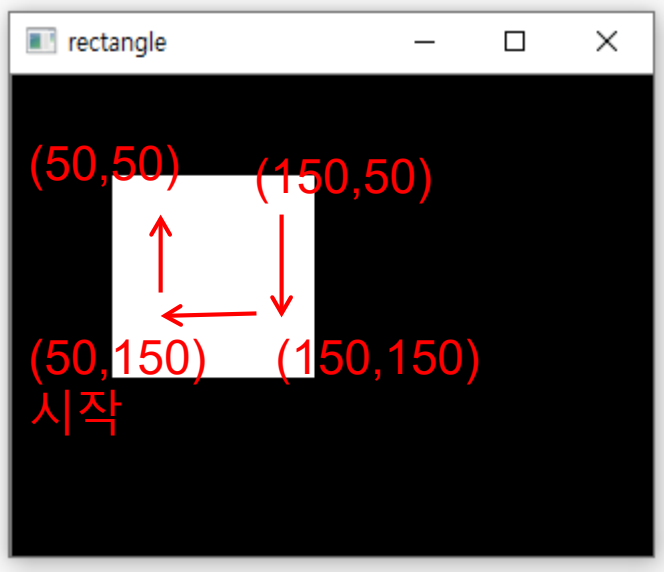
2. 그래픽 도형 면적-Rectangle

1_contourArea.py



← 좌측 그림에 보인 바와 같이 contour를 분석해 보면 같이 자료가 반시계 방향으로 구성되었음을 확인할 수 있었다. 따라서 True 옵션으로 면적을 구하면 -부호의 면적이 반환된다.

우측 그림의 경우는 contour 원본의 좌표 정보를 거꾸로 인덱싱하여 contourArea() 함수에 제공한 것이다. 따라서, True 옵션으로 면적을 구하면 시계 방향이므로 +부호의 면적이 반환된다.



contourArea(contour, **True**)

```
    x   y
[[[ 50  50]]
 [[ 50 150]]
 [[150 150]]
 [[150  50]]]
```

원본은 반시계 방향으로 배열되어 있음. - 부호

```
For this contour:
area with sign=-10000.00
    orientation: counter-clockwise.
```

contourArea(contour , **True**)

contourArea() 함수를 호출할 때 역방향으로 배열(시계 방향) 하여 입력하였음. + 부호

```
    x   y
[[[150  50]]
 [[150 150]]
 [[ 50 150]]
 [[ 50  50]]]
```

```
For this reverse contour:
area with sign=10000.00
    orientation: clockwise
```

3. 그래픽 도형 면적 - Circle

1_contourArea.py

회전 방향에 따라 3개의 점을 그리는 루틴

```
contour = contours[0]
x,y = contour[0][0]; point = (x,y);
cv.circle(imgC, point, 10, (0, 0, 255), -1)  # starting location

x,y = contour[int(len(contour)/3)][0]; point = (x,y)
cv.circle(imgC, point, 5, (0, 255, 255), -1) # 2/3 location

x,y = contour[int(len(contour)*2/3)][0]; point = (x,y)
 cv.circle(imgC, point, 3, (0, 255, 0), -1)    # 2/3 location
```

```
2) Computational area:
    area_exp=31415.93
4) cv2.contourArea(contour) 함수로 반환받은 면적
oriented=False, default → 절댓값 면적 반환
contourArea()=31134.00
5) Error: expectaion-contourArea=281.93,
    percentage=0.90%  차이남
                                  부호 있는 면적 반환
6) 부호있는 면적을 반환: contourArea(contour, oriented=True)
area with sign=-31134.00
    orientation: counter-clockwise. → 부호=음수

7) 취득한 컨투어를 역방향으로 컨투어를 제공합니다.
부호있는 면적: contourArea(contour[::-1], oriented=True):
area with sign=31134.00
    orientation: clockwise → 부호=양수
```

4. 최대/최소 크기의 객체 구하기

2_contourArea_compare.py

```
contours, _ = cv2.findContours(imgBin, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

인수들 공부하기

```
area = []
for i in range(len(contours)):
    a = cv2.contourArea(contours[i])
    print(f'area of contour[{i}]:{a}')
    area.append(a)
i_min = np.argmin(area)  배열에서 제일 작은 요소
i_max = np.argmax(area)  배열에서 제일 큰 요소
print('index of min=', i_min)
print('index of max=', i_max)
```

```
Number of total contours =  5
area of contour[0]:8678.5
area of contour[1]:11772.0
area of contour[2]:34333.0
area of contour[3]:7697.0
area of contour[4]:21034.0
index of min= 3
index of max= 2
```

컨투어를 선 색깔로 채우기(255, 0, 0)

```
# 1) 가장 면적이 작은 컨투어의 중심에 Min으로 표기한다.
cv2.drawContours(img2, contours, i_min, (255, 0, 0), -1)    # 특정 콘투어 지정
M = cv2.moments(contours[i_min])    # 영상 모멘트를 계산한다.
x, y = centroid(M)    # 중점
cv2.putText(img2, 'Min', (x,y), cv.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)
```

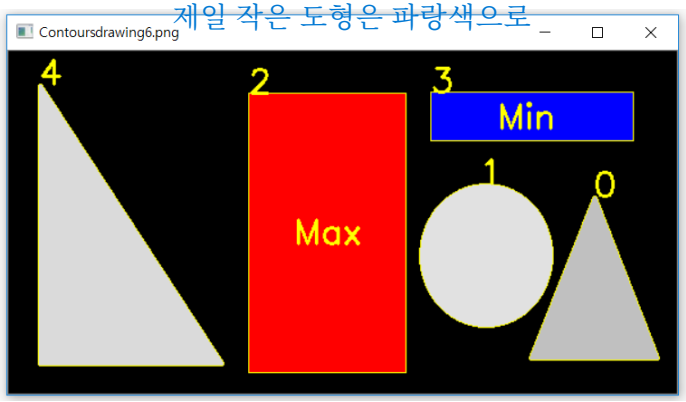
제일 작은 컨투어 참조

B G R

면적 제일 큰 도형은 빨간색으로,
제일 작은 도형은 파랑색으로

```
def centroid(moments):
    """Returns centroid based on moments"""
    x_correction = - 30     # 출력 폰트의 추정 크기 만큼 보정한다.
    y_correction = + 10
    x_centroid = round(moments['m10'] / moments['m00']) + x_correction
    y_centroid = round(moments['m01'] / moments['m00']) + y_correction
    return x_centroid, y_centroid
```

min 글자 컨투어 가운데에 두게 하려고



복습: 특정 컨투어 인덱스를 지정해서 그릴 때

- i번째 콘투어를 그릴 때:
  - cv2.drawContours(img, contours, i, (255, 0, 0), 2)
  - cv2.drawContours(img, [contours[i]], -1, (0, 0, 255), 2)

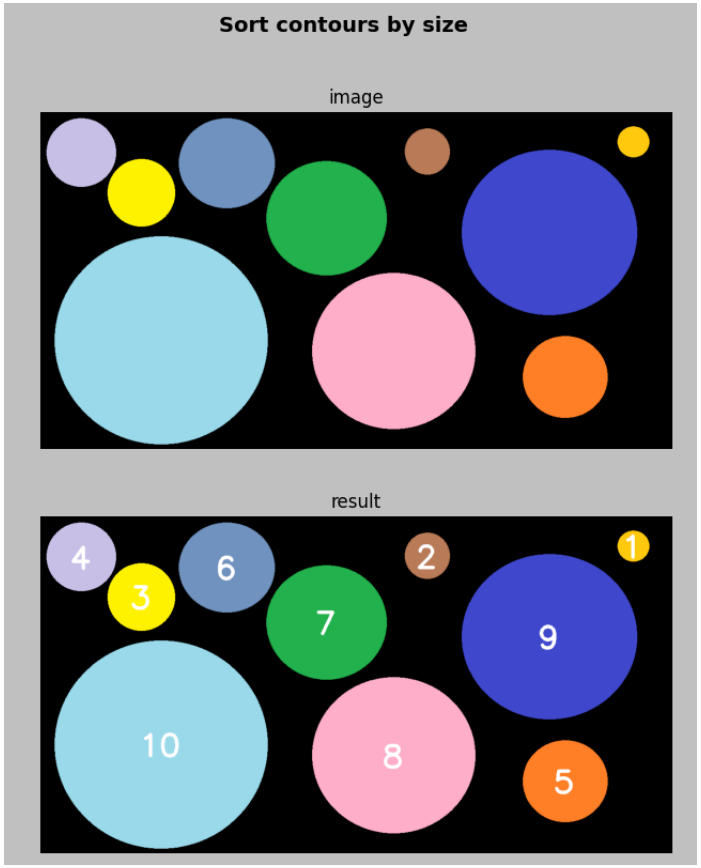
옳은 형식

- 주의:
  - 아래는 i번째 콘투어를 구성하는 점을 그린다.→ 활용 가치 없음
  - cv2.drawContours(img, contours[i], -1, (0, 0, 255), 2)

오류 형식임
리스트 형태로 넣어야 하기 때문에

5. 검출되는 객체를 크기 순으로 소팅

3_contours_sort_size.py




```
# 입력되는 contours를 전달받아 면적을 계산하여 면적의 크기순으로 소팅된
# cnt_sizes(면적 리스트)와 면적 크기로 재배열된 cnts를 반환한다.


1 usage        컨투어스를 넣으면 정렬된 컨투어스가 반환됨
def sort_contours_size(contours):
    # cnts_sizes는 면적으로 이루어진 list. 아직 소팅은 안됨.
    cnts_sizes = [cv2.contourArea(contour) for contour in contours]
    # (면적, 컨투어들) 자료를 cnts_sizes 중심으로 소팅하여 반환한다.
    (cnts_sizes, cnts) = zip(*sorted(zip(cnts_sizes, contours)))
    return cnts_sizes, cnts
```

오름차순으로 정렬된 contours 반환됨 = cnts