Moments of image, contour

각 섹션 별 내용 차례

2024년 1학기

서경대학교 김진헌

Structural Analysis and Shape Descriptors

차례

1_center_of_graphic_img.py

2_center_of_gray_or_bin_image.py

3_center_of_contour.py

- 모멘트의 여러 특징 중 주로 모멘트를 이용해 중심점을 찾는 것에 초점을 맞추어 분석함.
- OpenCV 함수는 영상 모멘트와 컨투어 모멘트를 지원한다.
- 영상 모멘트는 이진 영상과 그레이 영상 모멘트를 구분하여 계산한다

1. Moments 개요

무게중심이 영상의 특징이 될 수 있다

- 객체 - 영상, 윤곽선(점들의 집합)- 의 특징을 화소값과 좌표의 지수승과의 곱으로 계량화
- 3차 지수승까지 연산. 그중 영상의 특징으로 활용 가치 있는 것만 선택.
- 객체의 특징을 수치화 → 이후 분석 작업: 예) 두 객체의 비교. 분류/인식

(1) The spatial moments $\mathtt{Moments::m}_{ji}$ are computed as:

M00는 모든 영상의 합?

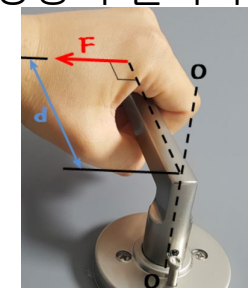
i, j를 모멘트의 차수라고 한다.

$$\mathtt{m}_{ji} = \sum_{x,y} \left(\mathtt{array}(x,y) \cdot x^j \cdot y^i \right)$$

화소값 거리

- m00: 영상에 있는 모든 계조값의 합 혹은 컨투어의 경우는 길이
  - 만약 계조 값이 0(검은 색), 1(흰 색)로 이진화가 되어있다면 흰 색의 면적
- m10: 이진화가 되어 있다면 x 축 방향의 길이(length, x좌표)의 모든 합.
  - x축을 지렛대로 사용하는 회전력. 세로 방향으로 모두 더해야 한다.
- m01: 이진화가 되어 있다면 y 축 방향의 길이의 모든 합
  - y축을 지렛대로 사용하는 회전력.
- x축 중심 = m10/m00
- y축 중심 = m01/m00

X 축의 픽셀값의 중심점 찾을 수 있다(?)



원점에서의 좌표값

거리하고 픽셀값을 곱한다(?)

Moment = 화소 값 x 거리

회전력(torque, moment)=힘 x 거리

$M[N \cdot m] = F[N] \cdot d[m]$

객체가 이동을 하면 특성이 달라짐 -> 중심점을 다시 맞춰서 다시 계산함

(2) The central moments $\texttt{Moments::mu}_{ji}$ are computed as:

Translation에 견실

$$\texttt{mu}_{ji} = \sum_{x,y} \left(\texttt{array}(x,y) \cdot (x - \bar{x})^j \cdot (y - \bar{y})^i\right)$$

where $(\bar{x}, \bar{y})$ is the mass center: $\bar{x} = \frac{\texttt{m}_{10}}{\texttt{m}_{00}}, \ \bar{y} = \frac{\texttt{m}_{01}}{\texttt{m}_{00}}$

로테이션은 HU MOMENT

(3) The normalized central moments $\texttt{Moments::nu}_{ij}$ are computed as: $\texttt{nu}_{ji} = \frac{\texttt{mu}_{ji}}{\texttt{m}_{00}^{(i+j)/2+1}}$.

$\texttt{mu}_{00} = \texttt{m}_{00}, \texttt{nu}_{00} = 1 \ \texttt{nu}_{10} = \texttt{mu}_{10} = \texttt{mu}_{01} = \texttt{mu}_{10} = 0,$

hence the values are not stored.

크기 변화에 견실

m_{00}는 영상의 면적과 유사한 개념
혹은, 윤곽선의 경우에는, 길이

3종의 모멘트의 특징

1. spatial moments: 거리와 화소값을 곱하는 동작을 3차승에 대해서까지 시행

2. central moments: 1차원 무게 중심을 뺀 좌표를 연산에 사용.=> translation에 견실.
⇒ 1, 2는 모두 크기에 영향을 받는다.

spatial moments
double m00
double m10
double m01
double m20
double m11
double m02
double m30
double m21
double m12
double m03

3. central normalized moments: 영상의 화소 합계(m_{00})에 적당한 지수승을 취한 값으로 나누어 크기 변화에 대한 정규화를 꾀한다. 크기 변화에 대한 견실성이 있다.

central moments
double mu20
double mu11
double mu02
double mu30
double mu21
double mu12
double mu03

central normalized moments
double nu20
double nu11
double nu02
double nu30
double nu21
double nu12
double nu03

OpenCV 함수, moment()에서 반환하는 모멘트 값 (사전형 자료)

key

2. moments() 함수

영상(1채널) OR CONTOUR 자료

- retval = cv.moments(array[, BinaryImage=False])
  - Calculates all of the moments up to the third order of a polygon or rasterized shape.
  - The function computes moments, up to the 3rd order, of a vector shape or a rasterized shape.
  - 영상이나, findContours()에서 반환받은 컨투어 중 1개를 넣을 수 있다.
  - 영상을 넣으면 전체 모멘트가 반환되고, 컨투어를 1개 넣으면 그 컨투어의 모멘트를 반환 받을 수 있다.

- array : 다음 둘 중의 하나
  - **1) Raster image (1 channel, uint8 or float) ➔ 반환 값: image moment**
    - Note that the numpy type for the input array should be either np.int32 or np.float32. → cv.findContours() 함수에서 반환 받은 contour는 이미 np.int32 형을 사용하고 있다. → 조정없이 사용 가능
    - 2D array: 1채널 영상 데이터(8비트). 8비트 계조영상이거나 이진영상. 이진 영상도 8비트 영상(0 혹은 255)으로 표현되어야 함.
      - *영상을 입력으로 사용하면 객체 별로 모멘트를 구하는 것이 아니다. 전체 영상에 대한 모멘트를 반환한다.* 그레이 영상은 의미가 없다(?)
  - **2) Contour 자료(1개) ➔ 반환 값: contour moment**
    - ~~**an array (1×N or N×1) of 2D points** (Point or Point2f)~~ - opencv 문서
    - 컨투어 1개는 (**Nx1x2**)형으로 구성된다. N은 점의 개수, 2는 (x, y)를 나타낸다.
    - N개로 이루어진 화소에 대해 모멘트 연산을 행한다. 사실상 점에 대한 연산이다.
    - m_{00}는 컨투어의 길이를 나타낸다.

- binaryImage: If it is true, all non-zero image pixels are treated as 1's. The parameter is used for images only. 2D array일 때만 의미를 가짐.
  - False: 영상 데이터는 그레이 계조 정보이다. 입력 영상을 그레이 영상으로 제공하여야 한다.
    0이 아니면 모두 1이다 - 엉터리
  - True: 0이 아닌 화소는 모두 1로 간주한다. 입력 영상은 0, 255로 구성된 이진 영상이어야 한다.
- retval : returned in the structure **cv::Moments**.

3. 영상 모멘트

- 영상 모멘트는 이진 영상과 그레이 영상 모멘트를 구분하여 계산한다
- moments()를 구할 때 영상을 입력한다.
- 이때 입력 파라미터(binaryImage)로 binary 영상(True)으로 모멘트를 구할 것인지, gray 영상(Fault, default)으로 모멘트를 구할 것인지 결정한다.
- 영상은 둘 중 어떤 것인지 넣을 수 있으나, 실제로는 이들 플래그에 맞는 binary 혹은 gray 입력 영상을 제공하는 것이 합리적이라 판단된다.
- **이중 현실적으로 가장 쓸모가 있다고 여겨지는 것은 이진영상을 넣고, binary 영상(True)으로 모멘트를 구하는 것이라고 판단된다. ➔ 흰색 객체의 특징을 분석하는 용도로 적합.**
- 그레이 영상을 넣고 입력 파라미터(binaryImage)로 binary 영상(True)으로 설정하면 해석하기 곤란한 상황이 전개되니 주의 하자.
  - 그레이 영상의 1보다 큰 화소는 모두 객체로 취급하기 때문이다.

INPUT = 그레이영상, TRUE

사례 1: 단순 영상 모멘트 계산

- 이진(0과 1) 영상의 경우

실험 1_center_of_graphic_img.py

m = cv2.moments(imgBin)

binaryImage=False(default) 이므로 계조 영상으로 연산, imgBin은 uint8 계조영상이다.

실험 목표: 단순 4각형 이진 영상을 만들어 0차, 1차 모멘트 만이라도 직접 계산해서 이것이 moments() 함수로 연산한 것과 같은지 확인해 본다..

True로 하면 예측된 결과는?

```
# 이진 영상 정의
size = (w, h) = (11, 5)
imgBin = np.ones( shape: (h, w), np.uint8)

# m00 계산
print(f'rectangle dimension(w x h)={size}')
print(f'm00(전체 계조값의 합), area={w} x {h}: {w*h}') # 11 x 5: 55

# m10 계산 -원점(0,y)에 대해 x축을 지렛대로 토크를 구한다고 생각하자.
sum_x = np.sum(range(w))    # 0+1+2+...+(w-1) -> 0+1+2+...+10 = 55
print(f'\n1개의 x축(가로 축)에 따른 좌표 값의 합={sum_x}')
print(f'가로선의 수={h}: m10={sum_x * h}')   # 가로선의 수=5: m10=275

# m10 계산-원점(x,0)에 대해 y축을 지렛대로 토크를 구한다고 생각하자.
sum_y = np.sum(range(h))    # 0+1+2+...+(h-1) -> 0+1+2+3+4 =10
print(f'\n1개의 y축(세로 축)에 따른 좌표 값의 합={sum_y}')    # 10
print(f'세로선의 수={w}: m01={sum_y * w}')   # 세로선의 수=11: m01=110

# 이제 OpenCV 함수, moments()로 구해보자.
m = cv2.moments(imgBin)
```

```
rectangle dimension(w x h)=(11, 5)
m00(전체 계조값의 합), area=11 x 5: 55

1개의 x축(가로 축)에 따른 좌표 값의 합=55
가로선의 수=5: m10=275

1개의 y축(세로 축)에 따른 좌표 값의 합=10
세로선의 수=11: m01=110

type(m) <class 'dict'>
m00:          55.000000
m10:         275.000000
m01:         110.000000
center= (5, 2)
```

사례 2: 기하 영상 모멘트

- 이진(0과 255) 영상의 경우

실험 2_center_of_graphic_img.py

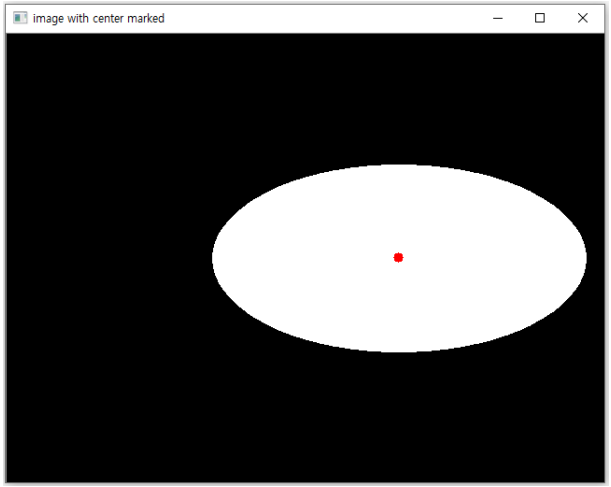
m = cv2.moments(imgBin) binaryImage=False 이므로 계조 영상으로 연산

ellipse(img, center, axes, angle, startAngle, endAngle, color)

cv2.ellipse(imgBin, (420, 240), (200, 100), 0, 0, 360, 255, -1)
cv2.ellipse(imgBin, (110, 240), (100, 200), 0, 0, 360, 255, -1)

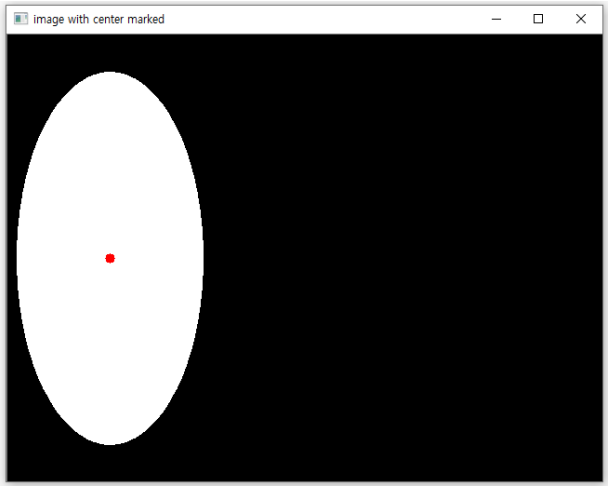
⇨

함수로 생성한 영상(A, B)의 기하학적 중심점의 평균과 이 두개를 한한 영상(C)에서 구한 모멘트의 중심점을 비교해 본다.



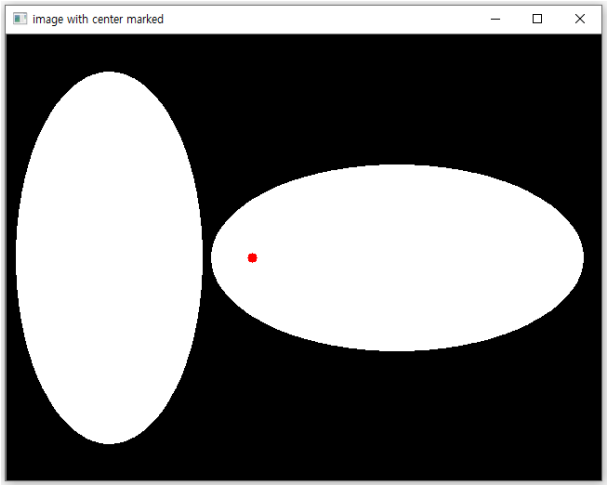
영상 A

x=419.995 y=239.999
center= (419, 239)



영상 B

x=110.001 y=240.002
center= (110, 240)



영상 C = A | B

x=264.983 y=240.001
center= (264, 240)

확인 목표: (영상 A의 중심 + 영상 B의 중심)/2 == 영상 C의 중심
A, B: 빨간 점은 타원형 함수의 중심점을 표시한 것

영상 C의 x좌표 = (419+110)/2 = 264.5
영상 C의 y좌표 = (239+240)/2 = 239.5
C: 빨간 점은 모멘트 함수에서 구한 점을 표시한 것

정리 및 미션

미션 수행을 위해서는 사례5까지 수행해 보아야 합니다.

- 실험 내용 정리 - 사례 1, 2에 대한 고찰
  - 1) A영상의 기하학적 중심을 ctr1이라 하자.
  - 2) B영상의 기하학적 중심을 ctr2이라 하자.
  - 3) A, B 두 영상의 기하학적 중심의 평균 “ctr3 = (ctr1+ctr2)/2”를 정의한다.
  - 4) C영상의 영상 모멘트로 구한 중심 ctr4라 하면, “ctr3==ctr4”가 True인 것을 확인하였다.
    - 이는 영상 모멘트의 중심이 기하학적 중심과 일치하는 것을 보이는 사례로 해석할 수 있다.
- 진도 완료 후 검토 권장 미션
  - 미션1
    - C 영상의 윤곽선을 구해 2개의 윤곽선 중심점을 각각 ctr5, ctr6라 하고, 그 평균을 ctr7이라 하자.
    - 질문: “ctr3 == ctr7”가 true일까? 즉, 여러 개별 기하학적 평균 중심과 그 영상의 개별 윤곽선 모멘트의 중심이 같을까?
  - 미션2
    - 영상 C의 영상 모멘트에 대한 중심 ctr8을 구한다.
    - 질문: “ctr8 == ctr7”이 True일까? , “ctr8 == ctr3”이 True일까?

주의!!: 편의상 “==가 true이냐”고 표기했는데 논리적 값은 같을 수 없다. 약간의 오차를 허용하면서 같은 값을 갖는 가를 질문하는 것으로 이해하기 바랍니다.

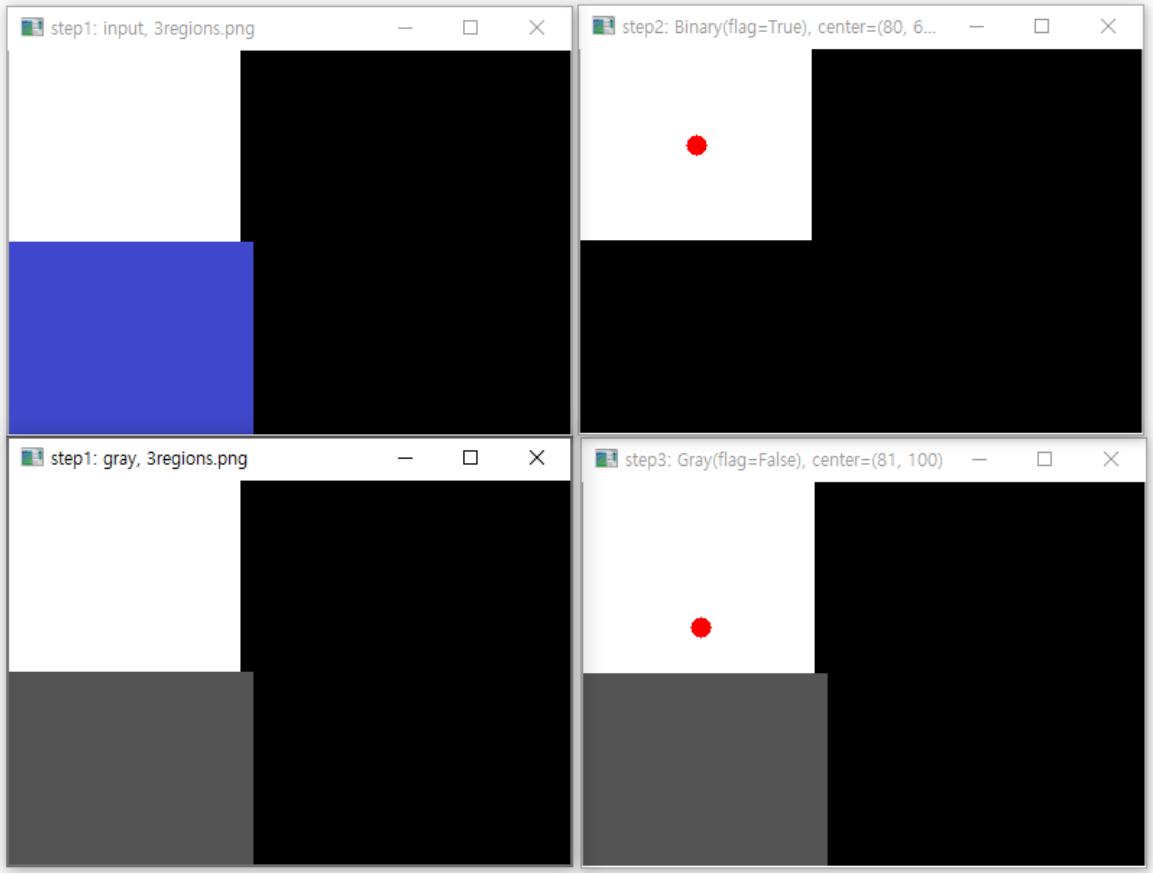
사례 3: 계조/이진 영상의 영상 모멘트

-binaryImage 옵션을 다르게 적용

적용 파일: 3regions.png: 2_center_of_gray_or_bin_image.py

1. 원본 영상

2. binaryImage=True



1. 모멘트 함수에 전달된 계조 영상

3. binaryImage=False

2. True옵션에서는 영상을 이진화시켜 사용해야 의미가 있다. 흰 오브젝트는 좌측 상단의 작은 4각형이므로 이것의 무게 중심을 구한다.

붉은 점은 무게 중심을 의미한다.

```
flag=True: image's center= (80, 66)
flag=False: image's center= (81, 100)
```

3. False에서는 회색 영상을 사용해야 의미가 있다. 아래로 내려온 이유는 좌측하단부의 청색 성분 때문이다. x 축 중심이 우측으로 1 화소 치우친 이유는 그 블록이 오른 쪽으로 조금 더 크기 때문이다.

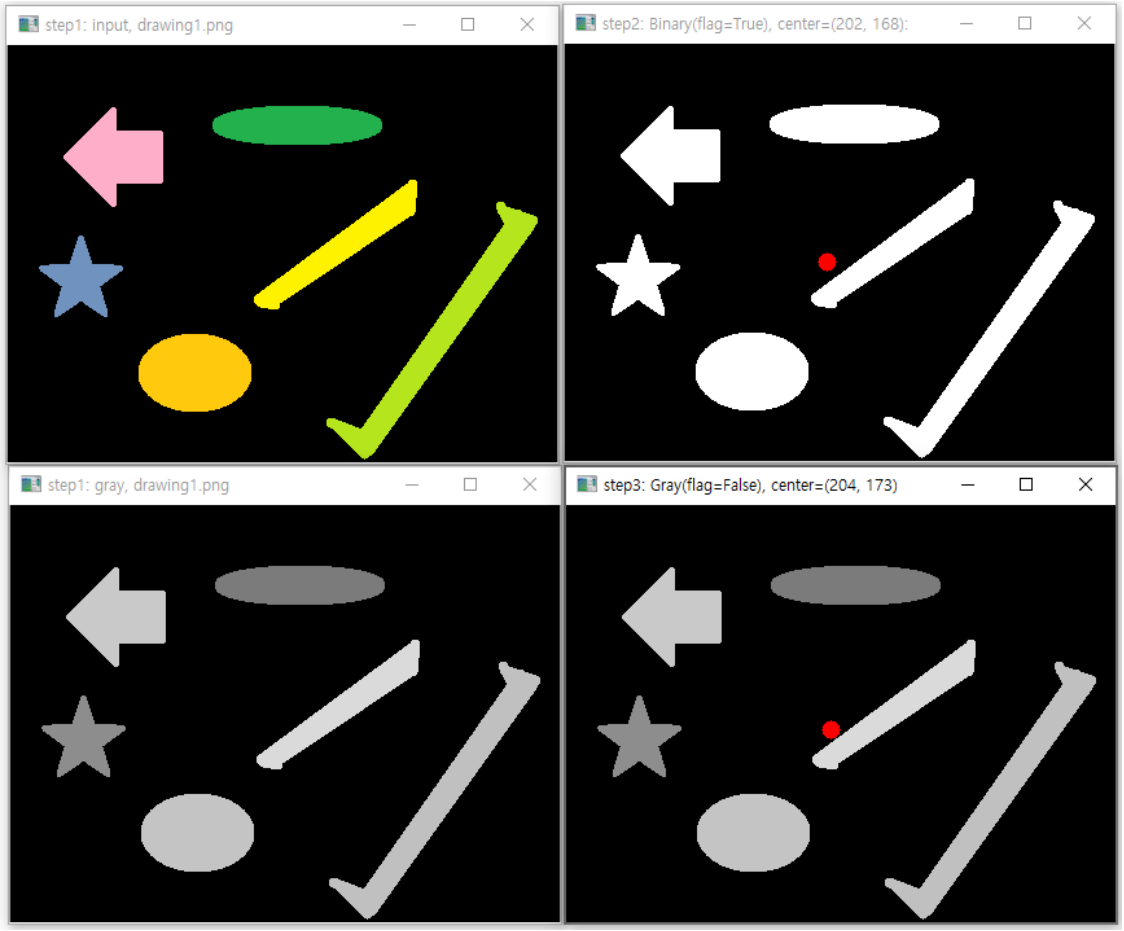
사례 4: 계조/이진 영상의 영상 모멘트

-binaryImage 옵션을 다르게 적용

적용 파일: drawing1.png: 2_center_of_gray_or_bin_image.py

1. 원본 영상

2. binaryImage=True



1. 모멘트 함수에 전달된 계조 영상

3. binaryImage=False

두 옵션에 대해 거의 같은 위치처럼 보이지만 실제로는 아래와 같이 그 중심이 같지 않다. 이는 회색의 값이 미치는 영향 때문이다.

붉은 점은 무게 중심을 의미한다.

```
flag=True: image's center= (202, 168)
flag=False: image's center= (204, 173)
```

그레이로 하는 것은 별로 의미가 없다

4. 윤곽선 모멘트

- 윤곽선을 findContours() 함수를 통해 구한 후 contours 중에서 1개의 contour를 moments() 함수에 입력하여 해당 객체의 모멘트를 반환받는다.

findContours() 함수는 여러 객체를 반환함

모멘트는 각 객체별로 받기 때문에 한꺼번에 할 수 없다.
객체 하나하나 구해서

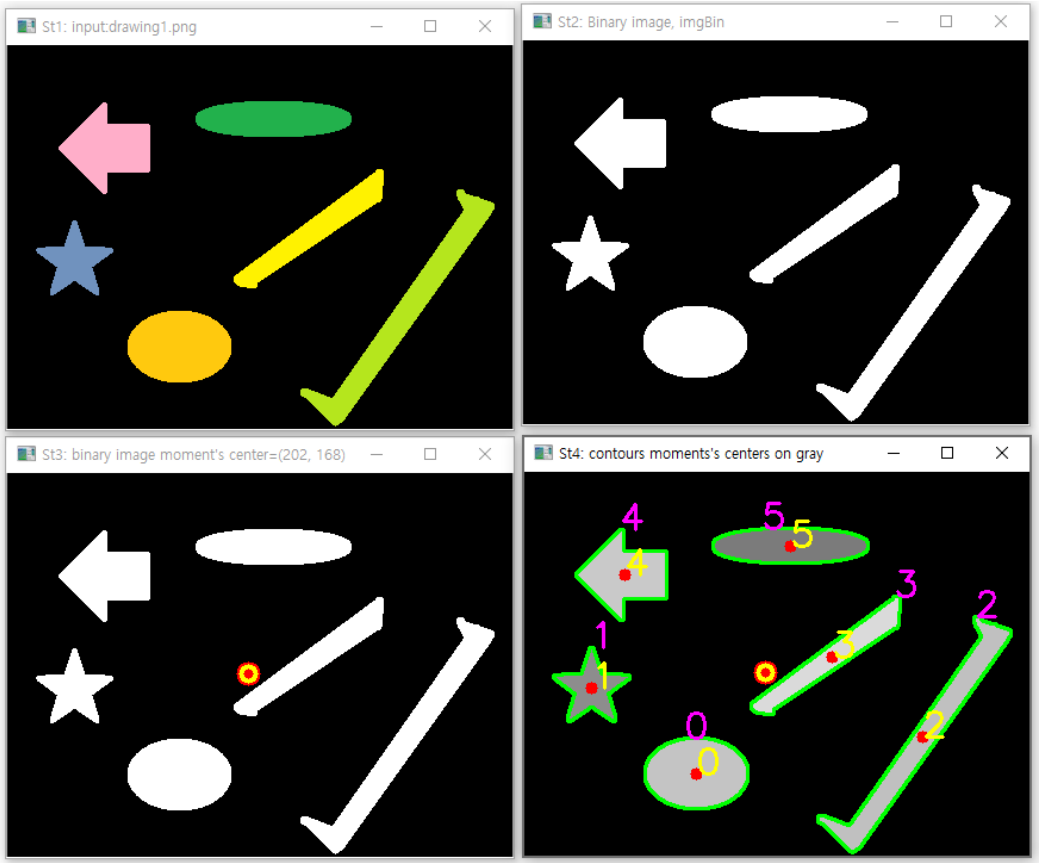
- 현재 윤곽선 모멘트의 중심과 영상 모멘트의 중심이 일치하지 않고 있다. 아직 그 이유가 분석이 되지 않은 상황이다.

사례 1: 윤곽선 모멘트

적용 파일: drawing1.png: 3_center_of_contour.py

Step 1: 원본 영상

Step 2: 이진화 영상



Step 3: 영상 모멘트 중심

Step 4: 컨투어 모멘트의 중심과 번호

이진 영상을 findContours() 함수에 제공하여 추출한 컨투에 대해 다음과 같이 각각의 모멘트를 구한다.

```
for c in contours:
    m = cv2.moments(c)
```

컨투어 기반의 센터 모멘트로 연산한 center

```
Number of total contours =  6
cn= 0  center= (144, 253)
cn= 1  center= (56, 181)
cn= 2  center= (334, 222)
cn= 3  center= (258, 155)
cn= 4  center= (84, 86)
cn= 5  center= (223, 62)
```

불일치한 경우가 정상
컨투어의 크기에 따라서 무게 중심이 달라짐

```
contour moments's centers=
 [183.16666667 159.83333333]
image moment center using binary image  =(202, 168)
```

검토 요망: 불일치

검토

- 다음 조건에서 주어진 객체의 중심점을 서로 비교하고 분석하시오.
  - 1) Hole이 있는 어떤 객체(1개)의 중심점을 영상 모멘트와 컨투어를 구하여 비교.
    - obj_with_hole.jpg 파일 사용.
    - 참고: hole이 존재하는 객체의 컨투어 모멘트의 중심은 연산 방법을 아직 제시한 적이 없습니다. 영상 모멘트가 정상이라고 생각하고 컨투어 모멘트로 그 값을 산출하는 방안에 대해서 검토해 보세요.
  - 2)Hole이 있는 어떤 객체가 2개 있는 경우 영상 모멘트와 Contour 모멘트를 구하여 그 중심점을 비교.
    - objs_with_holes.jpg 파일 사용.