윤곽선 검출하기

+ 근사화해서 검출하기/윤곽선 그리기

contours를 어떤 방식으로 뽑아낼 것이냐가 중점

- findContours()
- drawContours()

2024년 1학기

서경대학교 김진헌

차례

중요

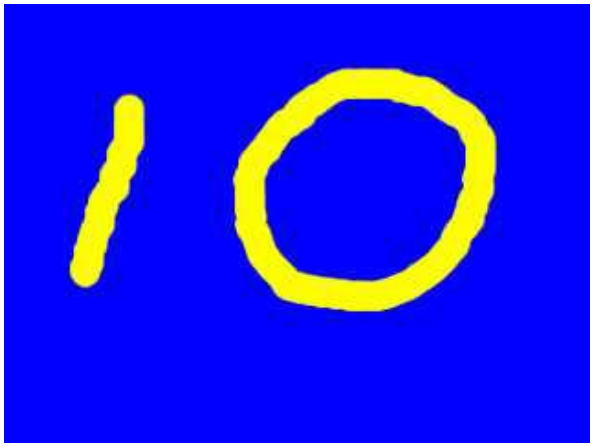
3차원 numpy 배열로 되어있는 (좌표 개수, 1 , 2)

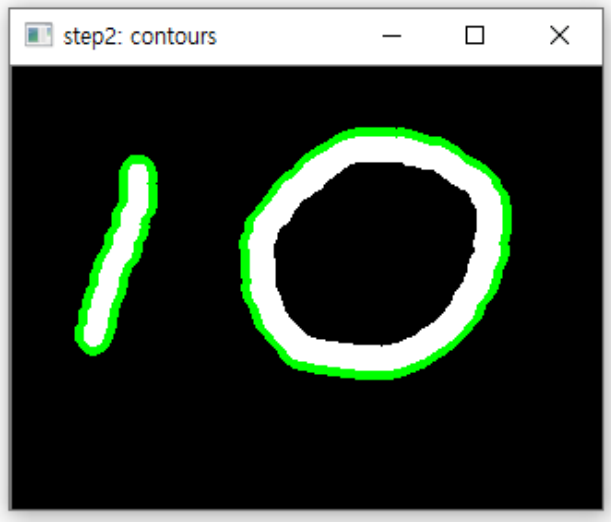
- find_drawContours.py

1. 들어가며..

find_drawContours.py

- 미션: 영상을 입력 받아 이진화를 시행한 후 윤곽선을 추출하여 그 정보를 바탕으로 윤곽선을 그린다.
- 프로그램: find_drawContours.py
- 사용한 주요 함수
  - findContours() 함수
    - 주요 입력: 이진 영상, 검색모드(retrieval mode), 윤곽선 근사화 방법(method)
    - 반환 값: contours, hierarchy
  - drawContours() 함수
    - 주요입력: contour, 혹은 contours 정보, 색상, 굵기





절차 및 주요 주제

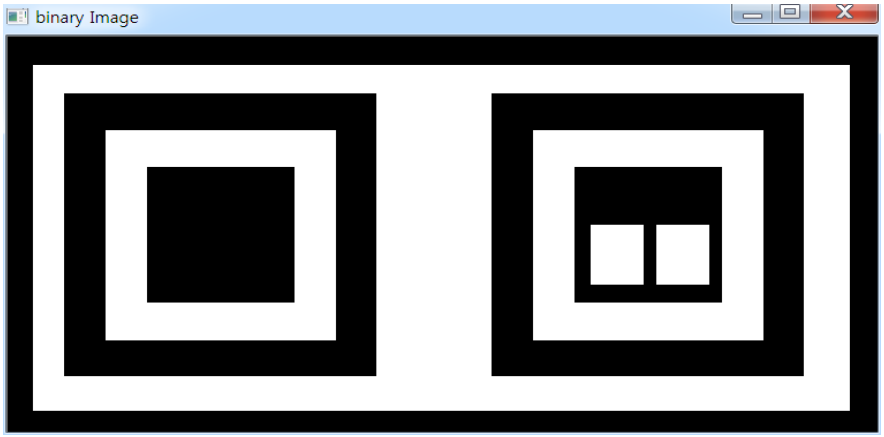
find_drawContours.py

1. 영상의 이진화
   - edge detection(Canny) or thresholding(Otsu)
     비추천 추천
2. findContours() 함수로 윤곽선(contours), 계층구조(hierarchy) 정보 반환받는다.
   - 윤곽선 - 이진 윤곽선의 좌표정보(x,y)를 저장
     - 검출되는 contours 정보는 여러 개의 contour들고 구성된 list 구조체로 되어있다.
     - contour는 임의의 contour(1개)를 구성하기 위한 점들의 ndarray 구조체이다. shape=(점의 개수, 1, 2). <= 1은 고정. 2는 (x,y)를 의미.
   - **계층구조(hierarchy)** - 한 화면에 존재하는 여러 윤곽선들의 관계를 기술
     - ndarray로 반환된다. 사례) hierarchy.shape= (1, 2, 4) ← 2개의 contour인 경우. 1, 4는 고정.
     - 4개의 필드로 이루어진 계층구조로 한 화면에 존재하는 여러 contour의 관계를 규명한다.
     - 이전 contour/ 다음 contour는 동격의 contour 번호를 지정하고, 첫째 자식/부모 contour 번호는 상하관계를 기술한다.
3. drawContours() 함수로 윤곽선을 그리기
   - 이 함수는 리스트 자료형의 Contours를 입력으로 받는다. 1개만 그리려고 해도 리스트로 만들어야 한다.

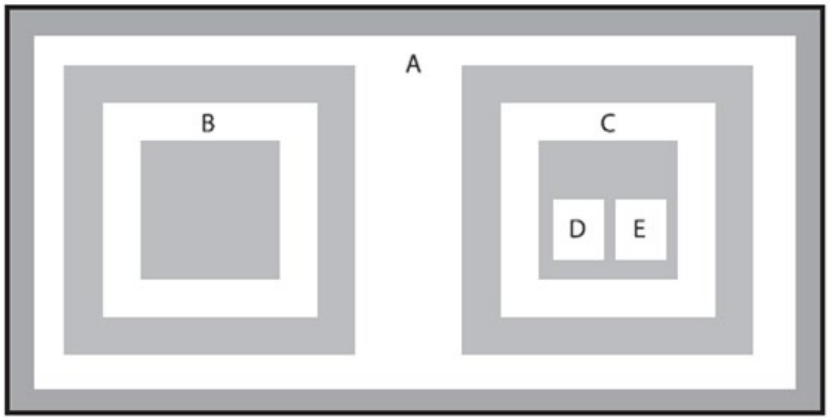
객체와 배경의 정의

FindContours.cpp

find_drawContours.py



CONTOUR_TEST_ORG.JPG



A~E가 검출하고자 하는 객체(흰색)이다.

- 윤곽선 추출 알고리즘에서는 검은 바탕에 흰 색의 물체가 있는 것으로 보고 이 흰색의 물체에 대한 윤곽선을 추출하는 것으로 이해한다.
- 위 그림에서는 흰색으로 되어 있는 A~E가 검출할 객체이다.
- **Contour** : 검은 색(배경) -> 흰 색(물체)으로 변하는 연결 선
- **Hole** : 흰 색(물체) -> 검은 색(배경) 으로 변하는 연결 선

2. findContours() 함수

2.1 호출 방법

findContours() 함수의 입력 파라미터

- The function retrieves contours from the binary image using the algorithm [Suzuki85]. The contours are a useful tool for shape analysis and object detection and recognition.
- C++: void **findContours**(InputOutputArray **image**, OutputArrayOfArrays **contours**, OutputArray **hierarchy**, int **mode**, int **method**, Point **offset**=Point())
- python: ~~**image,**~~ **contours**, **hierarchy** =**cv.findContours**(image, mode, method[, contours[, hierarchy[, offset]]])
- ~~**image**~~ - Source, an 8-bit single-channel image. Non-zero pixels are treated as 1's. Zero pixels remain 0's, so the image is treated as binary .
  - 입력으로 들어간 source 영상과 같은 영상이 1번째 반환 값(image)로 나온다. 반환 값을 사용할 용의가 없으면 underscore(_)를 많이 사용한다.
  - → 2020년 4월 현재. image 반환 값이 없어졌음

8 비트 싱글 채널 이미지
1~255 라면 255로 취급

2.2 mode: Contour retrieval mode

findContours() 함수의 입력 파라미터

검출될 contour들의 관계를 지정하는 입력.
검출된 것인 Contour인지/hole인지, 몇 단계에서 검출된 것인지 등의 정보를
기록하는 hierarchy 반환 값에 지정한 관계에 따라 정보가 기록되어 반환된다.

mode: Contour retrieval mode, see **RetrievalModes**

* 자세한 것은 실습 분석에서 다룸

```
MODE = cv2.RETR_TREE        # 모든 윤곽선을 수직적 관계로 계층 구조화
MODE = cv2.RETR_CCOMP       # 2개 레벨로 구조화: contour/hole
MODE = cv2.RETR_LIST        # 계층구조는 없고 모두 추출
MODE = cv2.RETR_EXTERNAL    # 맨 바깥의 윤곽선만 추출
```

contour/hole 둘 중에 하나

위의 10 숫자 사진

2.3 method, offset

findContours() 함수의 입력 파라미터

- **method** – Contour approximation method (if you use Python see also a note below). ⇨ contours_approximation_methods.py에서 기술
  - **CV_CHAIN_APPROX_NONE** stores absolutely all the contour points. That is, any 2 subsequent points (x1,y1) and (x2,y2) of the contour will be either horizontal, vertical or diagonal neighbors, that is, max(abs(x1-x2),abs(y2-y1))==1. 점이 많아
  - **CV_CHAIN_APPROX_SIMPLE** compresses horizontal, vertical, and diagonal segments and leaves only their end points. For example, an up-right rectangular contour is encoded with 4 points. horizontal, vertical, diagonal 로 단순화
  - **CV_CHAIN_APPROX_TC89_L1,CV_CHAIN_APPROX_TC89_KCOS** applies one of the flavors of the Teh-Chin chain approximation algorithm. See [TehChin89] for details. 좀 더 세련됨
- **offset** – Optional offset by which every contour point is shifted. 넘어감
  - This is useful if the contours are extracted from the image ROI and then they should be analyzed in the whole image context.

3. Contours 정보

findContours() 함수의 반환 값

- **contours** - Detected contours.
- C++ 설명: Each contour is stored as a vector of points. 생략
  - vector < vector<Point> > **contours**;
  - 윤곽선 한 개는 vector<Point>로 충분. 한 화면에 윤곽선이 많기 때문에 이들의 벡터 표현으로 윤곽선 집합이 표현된다.

- **Python 설명**: 1개의 윤곽선은 포인트들의 리스트 연결로 볼 수 있다.
  - contours is a Python list of all the contours in the image.
  - Each individual contour is a Numpy array of (x, y) coordinates of boundary points of the object.

Contours

findContours() 함수의 반환 값

- Contours 정보 출력 결과
  - print('**type(contours)=**', type(contours))
    - => *<class 'list'>*
  - print('**Number of total contours = **', len(contours))
    - => 검출된 윤곽선의 개수
  - print('**type(contour[0])=**', type(contours[0]))
    - => *<class 'numpy.ndarray'>. 첫 번째 윤곽선의 타입*
  - print('**len(contours[0])= **', len(contours[0]))
    - => *0번째 윤곽선을 구성하기 위한 모서리 점들의 개수*
  - print('contours[0].shape= ', contours[0].shape)
    - (len(contours[0]), 1, 2). 여기서 2는 (x,y)를 의미.
      점 개수
  - print('**contours[0]= **', contours[0])
    - => *0번째 윤곽선을 구성하기 위한 점들의 좌표를 출력한다.*
    - *사례: contours[0]= [[[22 27]] [[22 355]] [[730 355]] [[730 27]]]*
    - *4각형을 표현하는 윤곽선의 경우 4개 모서리의 좌표 정보를 담고 있다.->ndarray 타입*
    - → 복습: 여기서 point 좌표 (x, y)를 tuple로 산출하는 방법은? -> contours_generation.py의 array_to_tuple() 함수 정의 참조

4. Hierarchy 정보

findContours() 함수의 반환 값 2개 리턴함

Python 기반의 설명

- Hierarchy 정보:
  - print('**type(hierarchy)=**', type(hierarchy))
    - <class 'numpy.ndarray'>
  - print('**hierarchy.shape=**', hierarchy.shape)
    - hierarchy.shape= (1, **윤곽선의_개수**, 4)

[Next, Previous, First_Child, Parent]

1로 고정

동격의 이전 혹은 이후 컨투어의 번호
-1이면 더 이상의 윤곽선 없음
(처음이거나, 마지막 윤곽선)

-1이면 부/모 없음

  - print('**len(hierarchy)=**', len(hierarchy))
    - len(hierarchy)=1, 항상 고정. 이때문에 hierarchy[0] 만 사용됨.
  - **type(hierarchy[0])** → <class 'numpy.ndarray'>
  - **len(hierarchy[0])** → 윤곽선의 개수와 동일함.

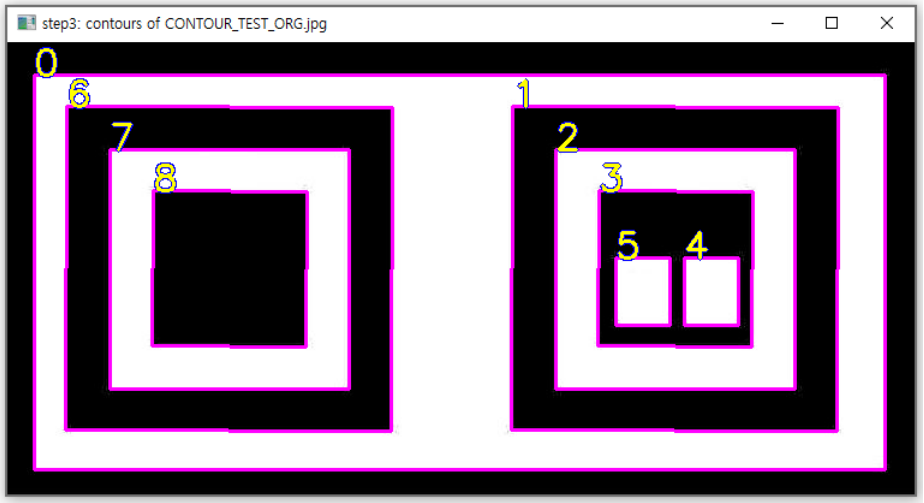
Hierarchy 상세 설명

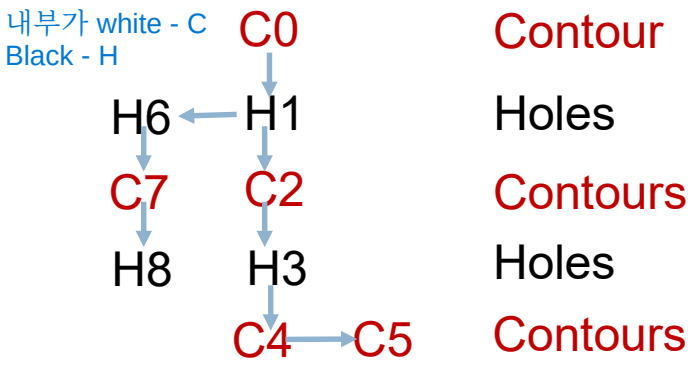
see **RetrievalModes**

RETR_EXTERNAL Python: cv.RETR_EXTERNAL	retrieves only the extreme outer contours. It sets `hierarchy[i][2]=hierarchy[i][3]=-1` for all the contours.
RETR_LIST Python: cv.RETR_LIST	retrieves all of the contours without establishing any hierarchical relationships.
RETR_CCOMP Python: cv.RETR_CCOMP	retrieves all of the contours and organizes them into a two-level hierarchy. At the top level, there are external boundaries of the components. At the second level, there are boundaries of the holes. If there is another contour inside a hole of a connected component, it is still put at the top level.
RETR_TREE Python: cv.RETR_TREE	retrieves all of the contours and reconstructs a full hierarchy of nested contours.
RETR_FLOODFILL Python: cv.RETR_FLOODFILL	

mode=CV_RETR_TREE

* 계층구조, tree

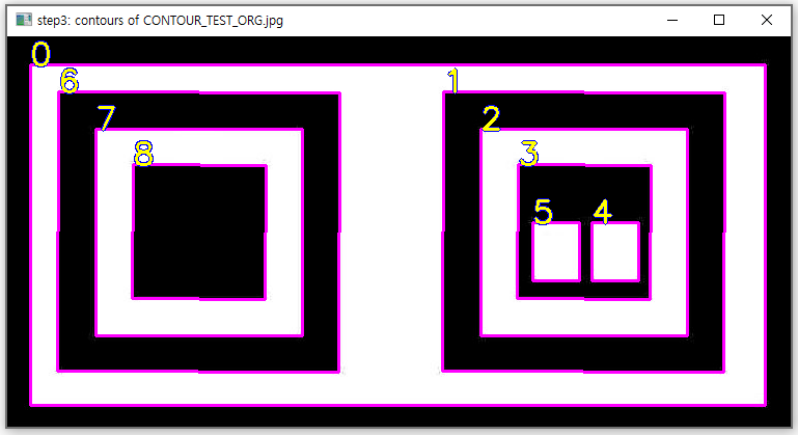




- TREE - 부모와 자식관계로 contour와 hole의 윤곽선을 기술한다.
  - contour : 어두운 면에서 밝은 면으로 넘어오면서 생기는 윤곽선,
  - hole : 밝은 면에서 어두운 면으로 넘어오면서 생기는 윤곽선
- 외곽의 contour를 추출하고 안에 있는 holes 이후 그 안에 있는 contour → hole → contour 순으로 추출된다. 부모와 자식 관계로 묘사된다.
- 그러나 contour과 hole을 구분하는 정보 필드는 없다. contour와 hole이 명시적으로 구분되지 않는다. → 모두 contour로 기술된다.
- Next/Previous는 연결되는 contour끼리 동격의 관계를 형성한다. Child/Parent는 수직 관계를 의미한다.
- 수직관계는 contour→hole 혹은 hole→contour 변화에서 만들어진다.

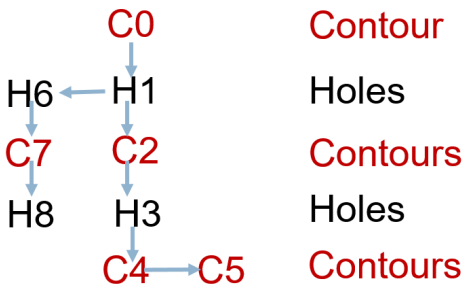
mode=CV_RETR_TREE

* 계층구조, tree



번호: hierarchy

0 : [-1 -1 1 -1]
1 : [6 -1 2 0]
2 : [-1 -1 3 1]
3 : [-1 -1 4 2]
4 : [5 -1 -1 3]
5 : [-1 4 -1 3]
6 : [-1 1 7 0]
7 : [-1 -1 8 6]
8 : [-1 -1 -1 7]

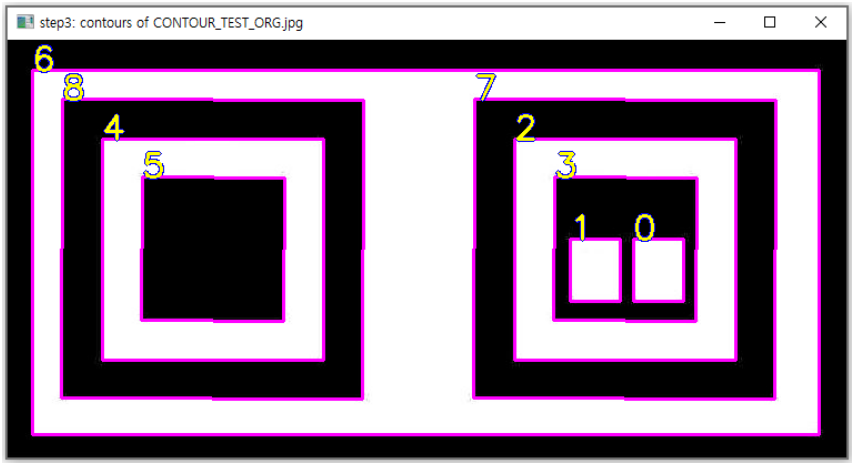


Contour
Holes
Contours
Holes
Contours

```
cntr= 0: Next=-1, Previous=-1, First_Child= 1, Parent=-1
cntr= 1: Next= 6, Previous=-1, First_Child= 2, Parent= 0
cntr= 2: Next=-1, Previous=-1, First_Child= 3, Parent= 1
cntr= 3: Next=-1, Previous=-1, First_Child= 4, Parent= 2
cntr= 4: Next= 5, Previous=-1, First_Child=-1, Parent= 3
cntr= 5: Next=-1, Previous= 4, First_Child=-1, Parent= 3
cntr= 6: Next=-1, Previous= 1, First_Child= 7, Parent= 0
cntr= 7: Next=-1, Previous=-1, First_Child= 8, Parent= 6
cntr= 8: Next=-1, Previous=-1, First_Child=-1, Parent= 7
```

mode=CV_RETR_CCOMP

* 계층구조, ccomp



- 2개의 층으로 추출하되, top level은 contour, 그 하위 층은 hole을 검출한다.
- Contour는 부모/자식이 모두 없거나 자식이 없다.
- Holes는 부모가 있다.→ 맨 바깥의 배경에서는 hole을 검출하지 않는다.

자식없음

0-> 1-> 2-> 4-> 6 Top level(Contours)

3 5 7-> 8 Lower level(Holes)

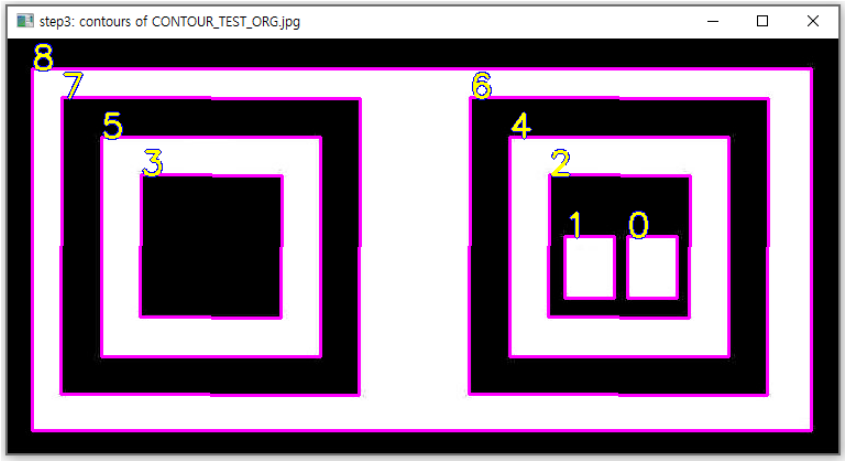
Parent 가 -1이 아니라면 Hole

```
cntr= 0: Next= 1, Previous=-1, First_Child=-1, Parent=-1
cntr= 1: Next= 2, Previous= 0, First_Child=-1, Parent=-1
cntr= 2: Next= 4, Previous= 1, First_Child= 3, Parent=-1
cntr= 3: Next=-1, Previous=-1, First_Child=-1, Parent= 2
cntr= 4: Next= 6, Previous= 2, First_Child= 5, Parent=-1
cntr= 5: Next=-1, Previous=-1, First_Child=-1, Parent= 4
cntr= 6: Next=-1, Previous= 4, First_Child= 7, Parent=-1
cntr= 7: Next= 8, Previous=-1, First_Child=-1, Parent= 6
cntr= 8: Next=-1, Previous= 7, First_Child=-1, Parent= 6
```

```
if Parent == -1:
    Contours
else Holes
```

mode=CV_RETR_LIST

* 계층구조, list



- 계층적 구조에 대한 관계를 고려하지 않고 모든 윤곽선을 도출해 낸다.
- 사실상 C와 H의 구분이 없다.

모두 부모, 자식없음. 대등관계

윤곽선 번호: hierarchy

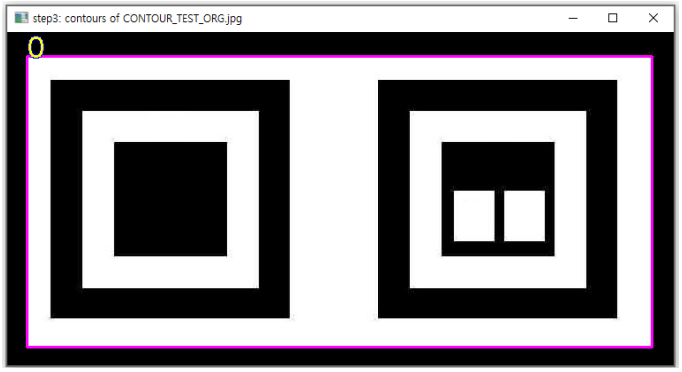
```
0 : [ 1 -1 -1 -1]
1 : [ 2  0 -1 -1]
2 : [ 3  1 -1 -1]
3 : [ 4  2 -1 -1]
4 : [ 5  3 -1 -1]
5 : [ 6  4 -1 -1]
6 : [ 7  5 -1 -1]
7 : [ 8  6 -1 -1]
8 : [-1  7 -1 -1]
```

```
0: Next= 1, Previous=-1, First_Child=-1, Parent=-1
1: Next= 2, Previous= 0, First_Child=-1, Parent=-1
cntr= 2: Next= 3, Previous= 1, First_Child=-1, Parent=-1
cntr= 3: Next= 4, Previous= 2, First_Child=-1, Parent=-1
cntr= 4: Next= 5, Previous= 3, First_Child=-1, Parent=-1
cntr= 5: Next= 6, Previous= 4, First_Child=-1, Parent=-1
cntr= 6: Next= 7, Previous= 5, First_Child=-1, Parent=-1
cntr= 7: Next= 8, Previous= 6, First_Child=-1, Parent=-1
cntr= 8: Next=-1, Previous= 7, First_Child=-1, Parent=-1
```

mode=CV_RETR_EXTERNAL

*계층구조, external

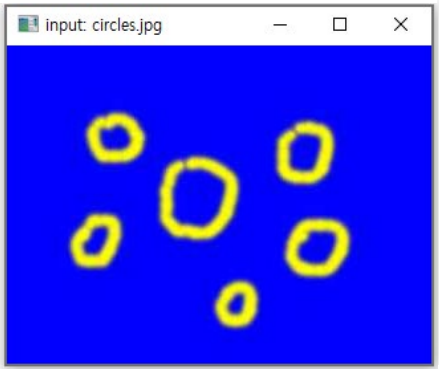
- retrieves only the extreme outer contours.

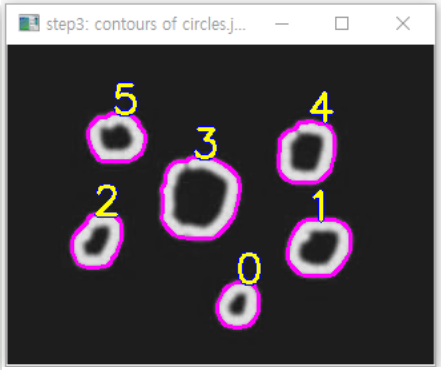



```
cntr= 0: Next=-1, Previous=-1, First_Child=-1, Parent=-1
```

맨 바깥 것 하나
내부 신경 안 씀

동료관계만 존재 / 부모, 자식관계 없음






```
cntr= 0: Next= 1, Previous=-1, First_Child=-1, Parent=-1
cntr= 1: Next= 2, Previous= 0, First_Child=-1, Parent=-1
cntr= 2: Next= 3, Previous= 1, First_Child=-1, Parent=-1
cntr= 3: Next= 4, Previous= 2, First_Child=-1, Parent=-1
cntr= 4: Next= 5, Previous= 3, First_Child=-1, Parent=-1
cntr= 5: Next=-1, Previous= 4, First_Child=-1, Parent=-1
```

5. drawContours() 함수

5. drawContours() 함수의 파라미터

- **Draws contours outlines or filled contours.**
- void **drawContours**(InputOutputArray **image**, InputArrayOfArrays **contours**, int **contourIdx**, const Scalar& **color**, int **thickness**=1, int **lineType**=8, InputArray **hierarchy**=noArray(), int **maxLevel**=INT_MAX, Point **offset**=Point())

cv.drawContours(image, contours, contourIdx, color[, thickness[, lineType[, hierarchy[, maxLevel[, offset]]]]]) -> image

- **image** - Destination image. 그림 그릴 화면
- **contours** - All the input contours. Each contour is stored as a point vector.
- **contourIdx** - Parameter indicating a contour to draw. If it is negative, all the contours are drawn. contour의 인덱스 번호. 총 개수 = contours.size() in C++, len(contours) in python

(a) contourIdx를 윤곽선 번호를 지정하여 하나씩 그린다.

```
for i in range(len(contours)):
    cv2.drawContours(img2, contours, contourIdx = i, color=(0, 0, 255), thickness=2)
```

(b) contourIdx를 음수로 지정하여 콘투어를 일괄 그린다.

```
cv2.drawContours(img3, contours, contourIdx=-1, color=(0, 255, 255), thickness=2)
```

- **color** - Color of the contours. 예: Scalar(0, 0, 255)
- **thickness** - Thickness of lines the contours are drawn with. If it is negative (for example, thickness=CV_FILLED), the contour interiors are drawn.
  - contour의 내부가 채워짐. = 이진 영상의 흰 색 부분이 채워진다.
- **lineType** - Line connectivity. See line() for details.
- **hierarchy** - Optional information about hierarchy. It is only needed if you want to draw only some of the contours (see maxLevel).
  - findContours의 hierarchy와 같음

- **maxLevel** - Maximal level for drawn contours.
  - If it is 0, only the specified contour is drawn.
  - If it is 1, the function draws the contour(s) and all the nested contours.
  - If it is 2, the function draws the contours, all the nested contours, all the nested-to-nested contours, and so on.
    - 값을 지정하지 않으면 INT_MAX(정수의 최대치). 모든 nested contour(hole 포함)을 모두 그린다.
  - This parameter is only taken into account when there is hierarchy available.
- **offset** - Optional contour shift parameter. Shift all the drawn contours by the specified .
  - ROI 영상에서 찾아낸 contour 정보를 원 영상의 좌표로 변환할 때 편리하게 사용할 수 있다.

6. Approximation Method

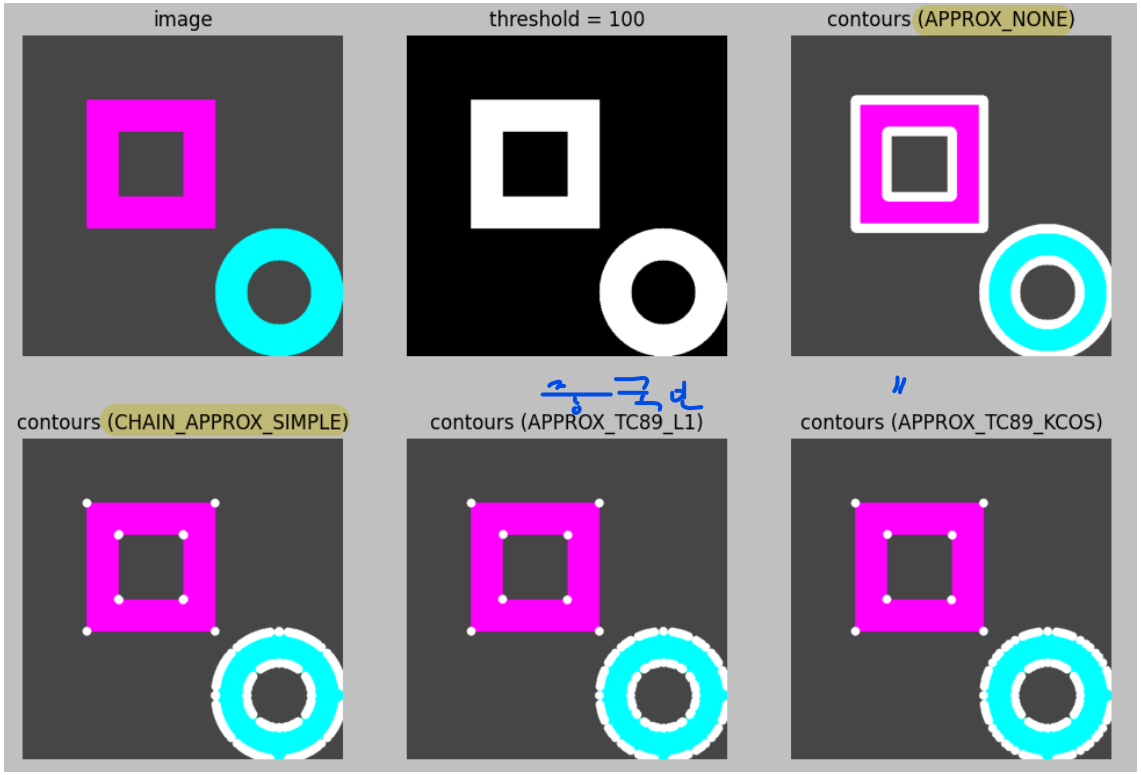
findContours()의 입력 파라미터

contours_approximation_methods.py

- **method** – Contour approximation method (if you use Python see also a note below).
- 다음과 같은 4개의 파라미터를 적용해서 추출되는 콘투어를 이루는 점들의 개수를 비교해 보기로 한다.
  - **CV_CHAIN_APPROX_NONE** stores absolutely all the contour points. That is, any 2 subsequent points (x1,y1) and (x2,y2) of the contour will be either horizontal, vertical or diagonal neighbors, that is, max(abs(x1-x2),abs(y2-y1))==1.
  - **CV_CHAIN_APPROX_SIMPLE** compresses horizontal, vertical, and diagonal segments and leaves only their end points. For example, an up-right rectangular contour is encoded with 4 points.
  - **CV_CHAIN_APPROX_TC89_L1,CV_CHAIN_APPROX_TC89_KCOS** applies one of the flavors of the Teh-Chin chain approximation algorithm. See [TehChin89] for details.

contours_approximation_methods.py

- 입력 영상에 대해 4개의 근사화 추출 기법을 적용했을 때 검출되어 나오는 컨투어를 이루는 점들의 개수를 비교해 본다.
- 이를 통해 어떤 알고리즘이 가장 많이 근사화를 이루는지 관찰 할 수 있다.



컨투어 4개(흰색선)에 대해 근사화를 이루는데 소요된 점의 개수

바깥선, 내부선 개수(?)

```
CHAIN_APPROX_NONE: points list [284, 564, 404, 800]
CHAIN_APPROX_SIMPLE: points list [148, 286, 8, 4]
CHAIN_APPROX_TC89_L1: points list [44, 106, 4, 4]
CHAIN_APPROX_TC89_KCOS: points list [52, 82, 4, 4]
```

7. 컨투어의 점을 그리기

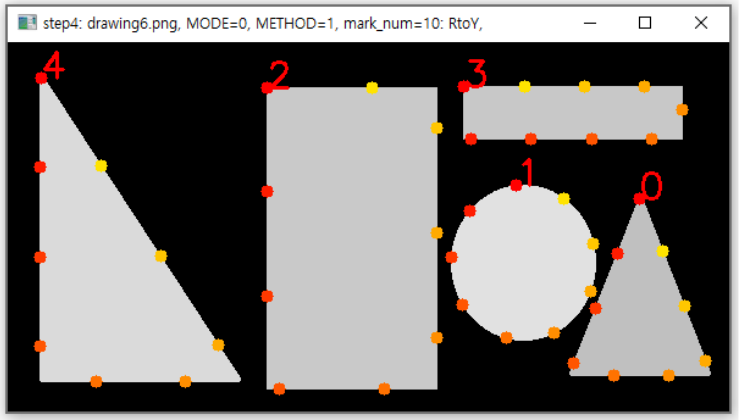
contour_points_marking.py

- 컨투어를 구성하기 위한 점의 배열을 관찰한다.
  - 컨투어의 점은 counter-clockwise로 배열되어 있다.
  - Hole의 점들은 clockwise로 배열되어 있다.

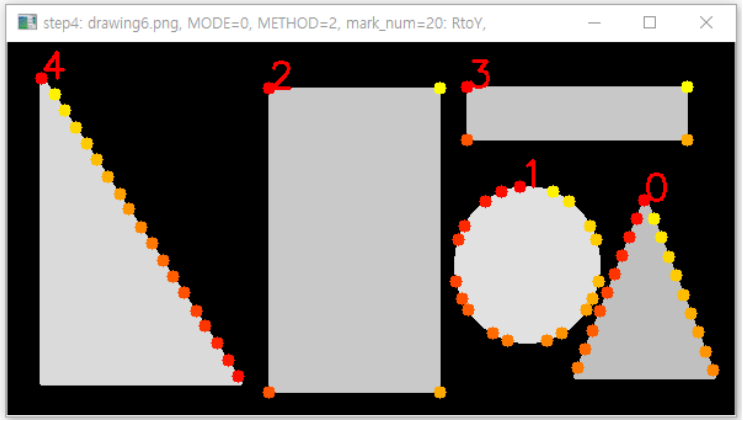
Contour의 경우

입력 영상: drawing6.png

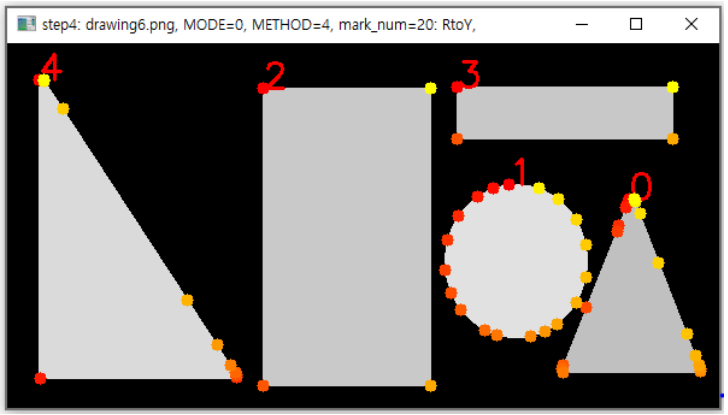
contour_points_marking.py



최대 mark_num=10 개 미만의 점으로 윤곽선 표현
MODE=RETR_EXTERNAL
METHD=CHAIN_APPROX_NONE



최대 mark_num=20 개 미만의 점으로 윤곽선 표현
MODE=RETR_EXTERNAL
METHD=CHAIN_APPROX_SIMPLE

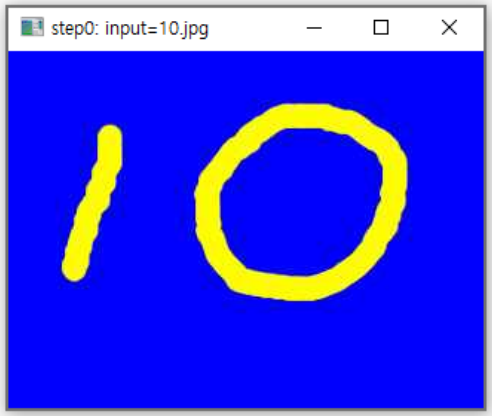


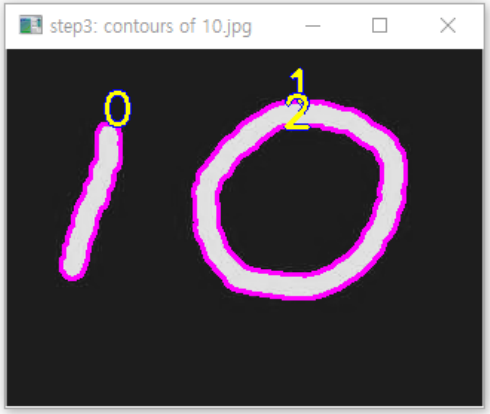
최대 mark_num=20 개 미만의 점으로 윤곽선 표현
MODE=RETR_EXTERNAL
METHD=CHAIN_APPROX_TC89_KCOS

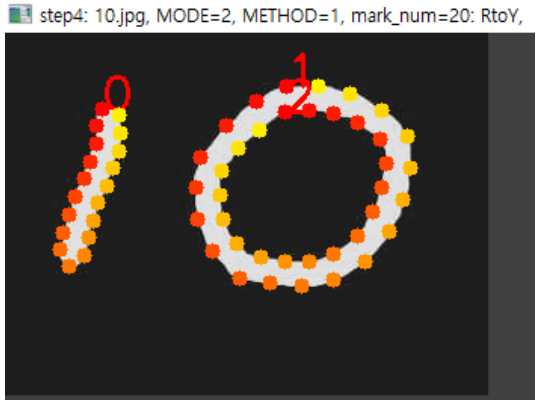
윤곽선상의 n개의 대표되는 점 정하고 그 중 시작점에 윤곽선 번호를 출력한다.
시작점은 적색에서 시작하여 끝의 점은 노란색으로 끝나도록 색상을 바꾸어 출력하였다. 실험 결과 컨투어의 점은 counter-clockwise로 배열됨을 알 수 있다.

Hole의 경우

contour_points_marking.py

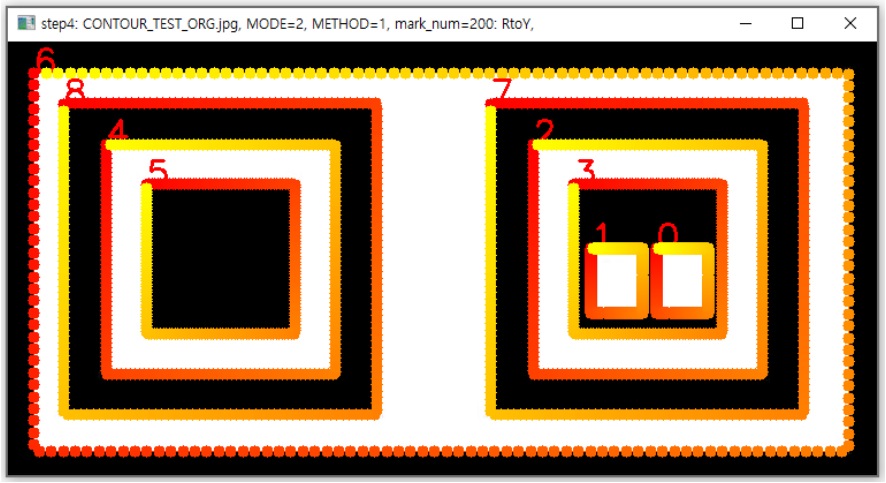






컨투어 2번은 Hole이다. 그런데 색상 배열이 0, 1번과는 반대 방향으로 시계 방향으로 Red to Yellow가 배열 되었음을 알 수 있다. Hole의 점들은 clockwise로 배열됨을 알 수 있다.

최대 mark_num=20 개 미만의 점으로 윤곽선 표현
MODE=CCOMP
METHD=CHAIN_APPROX_NONE



최대 mark_num=200 개 미만의 점으로 윤곽선 표현
MODE=RETR_CCOMP
METHD=CHAIN_APPROX_NONE

컨투어 7, 8, 5, 3번은 Hole이다. 색상 배열이 시계 방향으로 Red to Yellow가 배열 되었음을 알 수 있다. Hole의 점들은 clockwise로 배열됨을 다시 확인 할 수 있다.

검토사항

- 영상에서 윤곽선이 총 몇 개 검출되었는지 알아내는 방법에 대하여 기술하시오.
- Sequence와 Element의 차이점에 대해 기술하시오.
- 특정 윤곽선만 그리는 방법과 윤곽선 전체를 그리는 방법에 대해 기술하시오.