OpenCV Python

디지털영상처리/컴퓨터비전

Convex Hull 도형으로의 근사화

convexHull()
isContourConvex( )

contour을 집어넣어서 convexHull 이야? 라고 물어보는 거

2024년 1학기

서경대학교 김진헌

Structural Analysis and Shape Descriptors

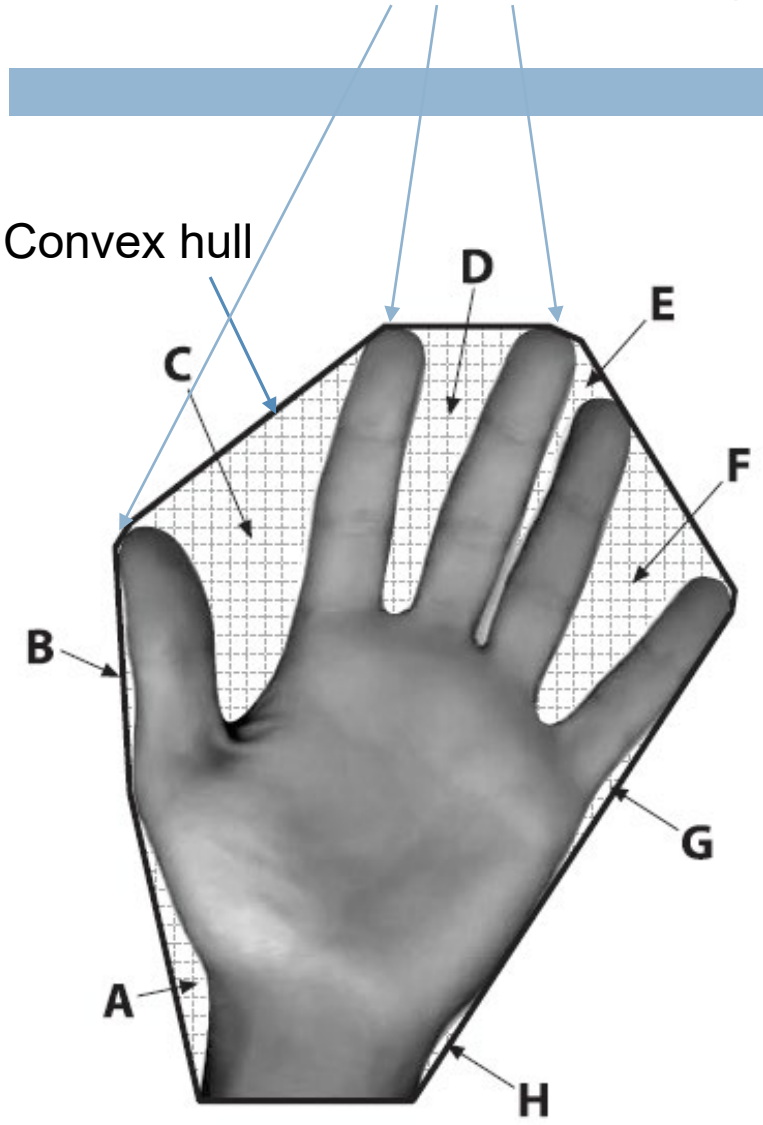
차례

1. Convex Hull이란

- A **Convex** object is one with no interior angles greater than 180 degrees.
- **Hull** means the exterior or the shape of the object.
- Therefore, the **Convex Hull** of a shape or a group of points is a tight fitting convex boundary around the points or the shape.
- 다른 표현: 여러 개의 점이 주어졌을 때, 모든 점들을 포함하는 최소 크기의 볼록 다각형
- ➔ 여러 점들을 둘러싸는 선이라 이해하면 될 듯. 각 점들에 못에 박혀 있고 이를 고무줄로 둘러싸는 모습과 유사.
- 특징
  - 모든 모서리가 볼록하다. 하나라도 있으면 탈락.
  - 연속된 3점의 내각이 180도를 넘지 않는다.
- 객체를 Convex Hull로 묘사하면 좋은 이유
  - 어떤 한 점이 콘벡스 다각형의 내부에 있는가를 알아보는 것이 매우 빠르다.
  - 어떤 점이 복잡한 다각형의 내부에 있는가를 알아볼 때는 이를 콘벡스 헐로 묘사하여 판단하는 것이 유리하다.

컨투어가 오목한 부분이 없다 -> convex hull 이다.

Contour Convexity and Convexity Defects



내부 추가된 부분?

Convexity Defect(A~H)
Convex Hull 내부에서 있는 컨투어의 오목한 부분

*Convexity defects: the dark contour line is a **convex hull** around the hand; the gridded regions (A–H) are **convexity defects** in the hand contour relative to the convex hull*

2. convexHull() 함수 개요

- Finds the convex hull of a point set.
- The function cv::convexHull finds the convex hull of a 2D point set using the Sklansky's algorithm [145] that has O(N logN) complexity in the current implementation.
  - Jack Sklansky. Finding the convex hull of a simple polygon. *Pattern Recognition Letters*, 1(2):79-83, 1982.
  - 그런데 A History of Linear-time Convex Hull Algorithms for Simple Polygons 자료를 보면 해당 알고리즘은 부정확한 것으로 발표 후 몇 년 지나지 않아 판정되었다.
  - Big O Notation: 알고리즘의 복잡도를 나타내는 표기법 중의 하나. N의 크기에 따라 증가되는 연산회수의 최악의 경우를 말한다. 본 알고리즘은 O(N logN)의 복잡도를 지니는 것으로 알려져 있다. ➔ 여러 알고리즘에 대한 소개 링크

- Useful link: 동영상 및 유사 알고리즘의 설명
  - https://www.learnopencv.com/convex-hull-using-opencv-in-python-and-c/

3. convexHull() 함수 용법

***hull**=cv.convexHull(**points**[, hull[, **clockwise=False**[, **returnPoints**=True]]])*
points를 입력받아 convex hull로 변환된 ***hull***을 반환한다.

- 입력 파라미터
  - = contour
  - *points* : Input 2D point set. findContour() 함수에서 반환한 contours 정보 중에서 하나의 콘투어를 입력으로 받는다.
  - **clockwise** : Orientation flag. 콘벡스 헐 데이터의 배열 순서를 정한다.
    - True: the output convex hull is oriented clockwise. 시계 방향. Hole의 경우…
    - False: it is oriented counter-clockwise. 반시계 방향. Contour의 경우..
    - The assumed coordinate system has its X axis pointing to the right, and its Y axis pointing upwards.
- **반환 값, hull** : convex hull. 파이썬이라면 *returnPoints*의 설정에 따라 다음 둘 중의 하나의 형태로 Hull의 윤곽정보를 전달받는다.
  - *returnPoints*=True: 변환된 헐 윤곽정보를 점의 좌표 정보로 전달 받는다.
  - *returnPoints*=False: 변환된 헐 윤곽정보를 입력시 사용되었던 컨투어의 인덱스 정보로 전달받는다. 컨투어에서 몇 번째 점이 헐의 윤곽을 구성하는지 말한다.

- 입력 파라미터 추가
  - *returnPoints* : Operation flag.
    - In case of a matrix
      - True: 컨벡스 헐의 points를 반환한다.
      - False: 컨벡스헐의 정보를 컨투어의 인덱스 번호로 대신하여 반환한다.
    - ~~When the output array is std::vector, the flag is ignored, and the output depends on the type of the vector:~~
      - ~~파이썬에서는 출력 데이터 타입을 입력 파라미터에서 지정하지 못하기 때문에 해당 사항이 없다.~~
      - ~~C++에서는 반환받을 hull의 데이터 타입을 vector로 지정하였다면<int>로 지정하였다면 인덱스 번호로 받환박고, <point>로 지정하였다면 점의 정보로 반환받는다.~~
      - ~~std::vector<int> implies returnPoints=false, std::vector<Point> implies returnPoints=true.~~

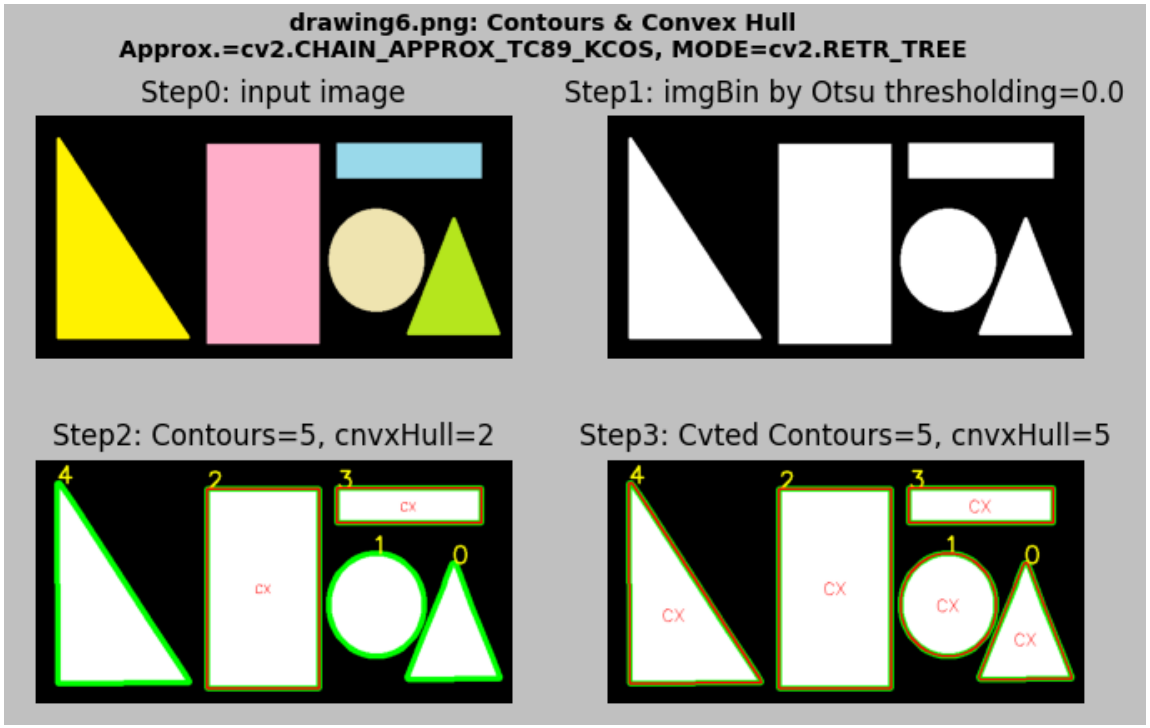
4. 실험 사례

- Convex hull로 바꾸는 함수:
  - 컨벡스 헐 = convexHull(컨투어, 어떻게 받을지 옵션)
    - (점 혹은 점에 대한 인덱스) 둘 중 한가지의 형태로 반환한다.

- Convex hull인지 확인하는 함수:
  - 논리값 = isContourConvex(컨투어, or 컨벡스 헐)

실험 1

convexHull.py



step2

```
Number of detected contours=5
0: CONVEX(X), contour.shape=(18, 1, 2)
1: CONVEX(X), contour.shape=(59, 1, 2)
2: CONVEX(O), contour.shape=(4, 1, 2)
3: CONVEX(O), contour.shape=(4, 1, 2)
4: CONVEX(X), contour.shape=(11, 1, 2)
```

step3

```
0: CONVEX(O), cnvx_hull.shape=(15, 1, 2)
1: CONVEX(O), cnvx_hull.shape=(55, 1, 2)
2: CONVEX(O), cnvx_hull.shape=(4, 1, 2)
3: CONVEX(O), cnvx_hull.shape=(4, 1, 2)
4: CONVEX(O), cnvx_hull.shape=(8, 1, 2)
```

step2

컨투어 자체를 convex hull인지 확인해 본 경우
4점으로 최적화된 컨투어로 된 도형(2번, 3번)은 통과하였다.
반면 0번, 4번처럼 다수의 점으로 컨투어화된 도형은 비통과하였다

step3

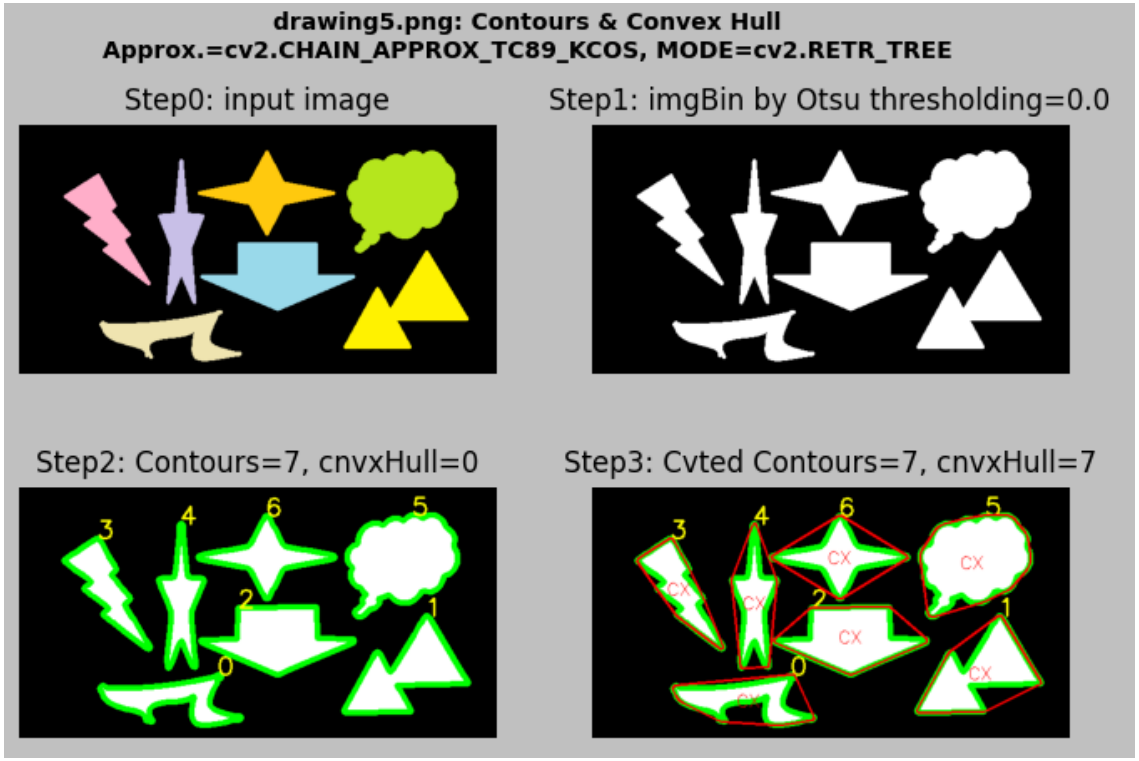
컨투어를 convexHull()함수로 convex hull로 바꾼 다음 확인해 본 경우: 모두 통과(당연)
0, 1, 4번 컨투어의 점의 수가 줄어 들면서 컨벡스로 판정되었다.

Convex hull로 바꾸는 함수:
convexHull(컨투어)

Convex hull인지 확인하는 함수:
isContourConvex(컨투어, 컨벡스홀)

실험 2

convexHull.py



step2

```
Number of detected contours=7
0: CONVEX(X), contour.shape=(84, 1, 2)
1: CONVEX(X), contour.shape=(37, 1, 2)
2: CONVEX(X), contour.shape=(21, 1, 2)
3: CONVEX(X), contour.shape=(58, 1, 2)
4: CONVEX(X), contour.shape=(51, 1, 2)
5: CONVEX(X), contour.shape=(116, 1, 2)
6: CONVEX(X), contour.shape=(47, 1, 2)
```

step3

```
0: CONVEX(O), cnvx_hull.shape=(17, 1, 2)
1: CONVEX(O), cnvx_hull.shape=(15, 1, 2)
2: CONVEX(O), cnvx_hull.shape=(12, 1, 2)
3: CONVEX(O), cnvx_hull.shape=(19, 1, 2)
4: CONVEX(O), cnvx_hull.shape=(11, 1, 2)
5: CONVEX(O), cnvx_hull.shape=(33, 1, 2)
6: CONVEX(O), cnvx_hull.shape=(8, 1, 2)
```

step2

컨투어 자체를 convex hull인지 확인해 본 경우 모두 컨벡스 헐이 아닌 것으로 평가받았다.
→ 당연함. 객체에 들어간 부분이 많음.

step3

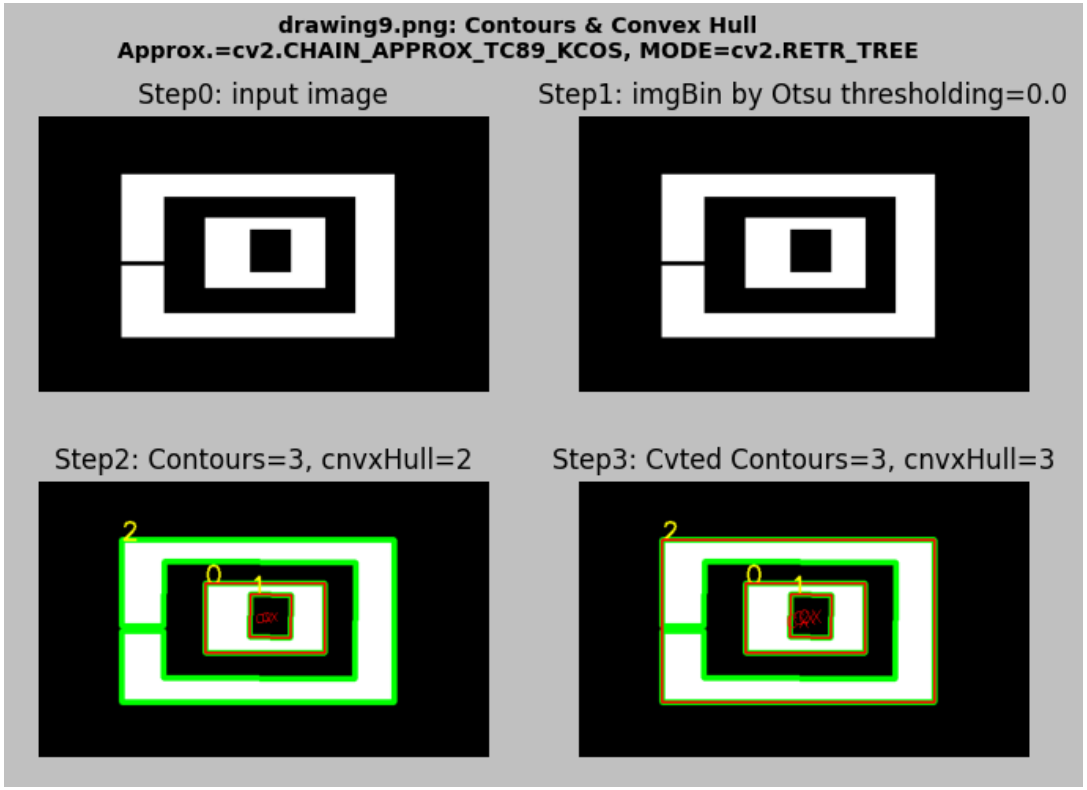
컨투어를 convexHull()함수로 convex hull로 바꾼다음 확인해 본 경우: 모두 통과(당연)
오목한 부분없이 원래의 객체를 모두 둘러싸고 있다.

Convex hull로 바꾸는 함수:
convexHull(컨투어)

Convex hull인지 확인하는 함수:
isContourConvex(컨투어, 컨벡스홀)

실험 3

convexHull.py



step2

```
Number of detected contours=3
0: CONVEX(O), contour.shape=(4, 1, 2)
1: CONVEX(O), contour.shape=(4, 1, 2)
2: CONVEX(X), contour.shape=(12, 1, 2)
```

step3

```
0: CONVEX(O), cnvx_hull.shape=(4, 1, 2)
1: CONVEX(O), cnvx_hull.shape=(4, 1, 2)
2: CONVEX(O), cnvx_hull.shape=(4, 1, 2)
```

step2

컨투어 자체를 convex hull인지 확인해 본 경우 4점으로 최적화된 컨투어로 된 도형(0번, 1번)은 통과하였다.
반면 2번은 오목한 모양을 갖고 있어 비통과하였다

step3

컨투어를 convexHull()함수로 convex hull로 바꾼다음 확인해 본 경우: 모두 통과(당연)
그런데 2번은 변환과정에서 내부 모양을 모두 잃어 버렸다. 내부의 4각형 정보가 모두 손실되었다.

Convex hull로 바꾸는 함수:
convexHull(컨투어)
Convex hull인지 확인하는 함수:
isContourConvex(컨투어, 컨벡스홀)

5. 참고: Convex Hull의 자료형에 대한 고찰

- 특정 contour(1개)에 대하여 다음과 같이 convex hull로 변환하였다고 가정하자.
  - hull = cv2.convexHull(contour)

contour하고 같은 자격을 갖는다?

- 이때 반환값 hull의 자료를 관찰하면 다음과 같다.
  - type(hull)= <class 'numpy.ndarray'>
  - hull.shape= (35, 1, 2) # 35는 hull로 모델링된 점의 개수
- drawContours() 함수로 hull을 그리려면 list 자료형으로 만들어야 한다.
  - a = [hull]
- 만약 여러 개의 hull을 contours(여러 개의 contour)처럼 만들고자 한다면 다음 2가지 방법이 있을 수 있다.
  - Hull_contours = [hull1, hull2, hull3]
  - Hull_contours =[ ]; Hull_contours.append(hull1); Hull_contours.append(hull2); Hull_contours.append(hull3);

6. 미션

소스 없음

- 임의의 도형을 convexHull() 함수로 convex하게 만들어 convex hull 자료형으로 반환 받는다.
- 이를 contour처럼 만들어 pointPolygonTest() 함수로 hull 곡선 안의 점이 과연 convex hull(붉은 색 도형) 안의 점으로 인정받을 수 있는지 확인하는 프로그램을 제작하시오.
  - 마우스로 점을 찍으면 해당 지점이 convex 안에있는지의 여부에 따라 그림의 두가지 색상으로 채운다.
- 또한 이를 일반화하여 여러 개의 hull에 대하여 트랙 바로 선택된 contour에 대하여 이 같은 동작을 수행하게 하시오.