chap 12 레이블링과 외곽선 검출

레이블링 -> 객체가 여러개 있을때 라벨을 붙여주는 것.

레이블링의 이해

- 배경과 객체를 구분하였다면 이제 다시 각각의 객체를 구분하고, 분석하는 작업이 필요. -> 이때 사용할 수 있는 기법이 레이블링(labeling)
- 레이블링은 영상내에 존재하는 객체 픽셀 집합에 고유 번호를 매기는 작업으로 연결된 구성 요소 레이블링 이라고 함.
- 레이블링 기법을 이용하여 각 객체의 위치와 크기, 정보를 추출하는 작업은 객체 인식을 위한 전처리 과정으로 자주 사용.

-- 영상의 레이블링은 일반적으로 이진화된 영상에서 수행.

- 검은색 픽셀은 배경, 흰식 픽셀은 객체로 간주. 정확하게 픽셀값이 0이면 배경, 0이 아니면 객체 픽셀로 인식. -> 하나의 객체는 1개 이상의 인접한 픽셀로 이루어 짐. 하나의 객체를 구성하는 모든 픽셀에는 같은 레이블 번호가 지정.

특정 픽셀과 이웃한 픽셀의 연결관계

1. 4-방향 연결성 : 특정픽셀의 상하좌우로 붙어있는 픽셀끼리 연결되어 있다고 정의
2. 8-방향 연결성 : 상하좌우 뿐만아니라 대각성 방향으로 인접한 픽셀도 연결되어 있다고 간주.

- 이진영상에 레이블링을 수행하면 각각의 객체 영역에 고유의 번호가 매겨진 2차원 정수 행렬이 만들어짐. -> 이러한 2차원 정수 행렬을 레이블 맵이라고 부름. 이를 수행하는 알고리즘은 매우 다양하게 존재하지만 모두 같은 형태의 레이블 맵을 생성

레이블링 실습 예제

- opencv에서는 connectedComponents()함수로 레이블맵 기능 제공

```
def labeling_basic():
    src = np.array([
    [0,0,1,1,0,0,0,0],
    [1,1,1,1,0,0,1,0],
    [1,1,1,1,0,0,0,0],
    [0,0,0,0,0,1,1,0],
    [0,0,0,1,1,1,1,0],
    [0,0,0,1,0,0,1,0],
    [0,0,1,1,1,1,1,0],
    [0,0,0,0,0,0,0,0]].astype(np.uint8))

    src = src * 255

    cnt, labels = cv.connectedComponents(src)

    # cnt에는 라벨의 개수가 담김. 이때 라벨이 3개라면 4가 출력됨. 배경이 포함되기에 실제 라벨
```

```
수는 -1을 해줘야 한다.
    # labels에는 라벨링 결과 행렬이 담긴다.
    print(cnt)
    print(labels)
```

- 기본적인 레이블링 동작은 입력 영상으로부터 레이블 맵을 생성하는 것.
- 보통 레이블링을 수행하고, 각각의 객체영역이 어느 위치에 어느정도 크기로 존재하는지 확인할 필요 있음
- Opencv에서는 레이블 맵과 각 객체 영역의 통계정보를 한번에 제공하는 connectedComponentsWithSats() 함수 제공

---

connectedComponentsWithSats() 이해하기 및 예제

- connectedComponents() 함수에 stats와 centroids가 추가된 형태.

- stats

0	**0**	**8**	**8**	**38**
0	0	4	3	10
6	1	1	1	1
2	3	5	4	14

행 기준 배경, 1번, 2번, 3번객체 각 열 요소는 바운딩 박스정보 (x,y,width,height,면적) 예를 들어 배경은 (0,0,width = 8, height = 8, area = 38)

- centroids

3.615	**3.615**
1.7	1.2
6	1
4.285	4.285

행은 각 순서대로 배경, 1,2,3번객체 각 정보는 무게중심의 x좌표, y좌표

- stats행렬에서 두번째 행 원소값은 [0,0,4,3,10] -> 1번 객체를 감싸는 바운딩 박스가 (0,0)좌표에서 시작하여 가로크기가 4, 세로크기가 3인 사각형이고 픽셀의 개수는 10
- centroids 행렬에서 두번째 행 원소값이 1.7,1.2로 저장되어 있음. 무게줌심 좌표가 (1.7,1.2)

```
def labeling_stats():
    src = cv.imread()
```

```
    _, src_bin = cv.threshold(src, 0,255,cv.THRESH_BINARY |
cv.THRESH_OTSU)

    cnt, labels, stats, centroids =
cv.connectedComponentsWidhStats(src_bin)

    dst = cv.cvtColor(src, cv.COLOR_GRAY2BGR)

    for i in range(1,cnt):
        (x,y,w,h,area) = stats[i]

        if area < 20:
            continue
        # 검출 객체가 20크기 미만이면 무시하겠다.

        pt1 = (x,y)
        pt2 = (x + w, y + h)
        cv.rectangle(dst, pt1, pt2, (0,255,255))
```

외곽선 검출

- 객체의 외곽선은 객체 영역 픽셀중에서 배경 영역과 인접한 일련의 픽셀을 의미 -> 보통 검은색 배경 안에 있는 흰색 객체 영역에서 가장 최외각에 있는 픽셀을 찾아 외곽선으로 정의함.
- 만약 흰색 객체 영역 안에 검은색 배경 영역인 홀이 존재한다면 홀을 둘러싸고 있는 개체 픽셀들도 외곽선으로 검출 -> 객체의 외곽선은 객체 바깥쪽 외곽선과 안쪽 홀 외곽선으로 구분할 수 있음. (도넛모양을 생각하면 도넛 안쪽, 도넛 바깥쪽!)
- 객체 하나의 외곽선은 여러 개의 점으로 구성. -> 또한 하나의 영상에 여러개의 객체가 존재할 수 있음으로 영상 하나에서 추출된 전체 객제의 외곽선 정보가 list형태로 반환됨.

findeCoutours() 함수

- openCv에서 제공하는 영상 내부 객체들의 외곽선을 검출하는 함수.

mode 인자로 외곽선을 어떤 방식으로 검출할 것인지 나타낼 수 있음.

1. RETR_EXTERNAL - 객체 바깥쪽 외곽선만 검색. 계층구조는 만들지 않는다.
2. RETR_LIST - 객체 바깥쪽과 안쪽 외곽선을 모두 검색. 계층구조는 만들지 않는다.
3. RETR_CCOMP - 모든 외곽선을 검색하고 2단계 계층 구조를 구성
4. RETR_TREE - 모든 외곽선을 검색하고, 전체 계층구조를 구성

method인자로 검출된 외곽선 좌표들을 근사화 하는 방법을 지정

1. CHAIN_APPROX_NONE - 모든 외곽선 점들의 좌표를 저장
2. CHAIN_APPROX_SIMPLE - 외곽선 중에서 수평선, 수직선, 대각선 성분은 끝점만 저장
3. CHAIN_APPROX_TC89_L1 - L1근사화 적용
4. CHAIN_APPROX_TC89_KCOS - cos근사화 적용

- TC89방식은 점의 개수는 줄이지만 모양이 달라질수 있으므로 주의

외곽선 계층 구조

- 객체 안에 다시 객체가 있을 수 있음 -> 외곽선의 계층구조는 외곽선의 포함관계에 의해 결정
- 포함관계가 있다면 부모-자식외곽선, 없다면 대등한 이전 혹은 다음 외곽선

외곽선 검출 모드

1. RETR_EXTERNAL

- 가장 바깥쪽 외곽선만 검출. 내부의 홀 외곽선은 검출되지 않음.
- 또한 계층정보를 저장하지 않음

2. RETR_LIST

- 바깥쪽과 안쪽 홀 외곽선 모두를 검출함.
- 그러나 외곽선의 부모/자식 계층 정보는 생성되지 않음.

3. RETR_CCOMP

- 먼저 바깥쪽 외곽선을 모두 검출하고, 안쪽 홀 외곽선이 있다면 검출하면서 자식 외곽선으로 설정함.
- 이때 계층정보를 저장하는데 상하 계층이 최대 2개로만 구성됨.
- 즉, 객체 안에 여러개의 홀이 존재할 경우, 그중 하나만 자식 외곽선으로 설정됨.

4. RETR_TREE

- 외곽선 전체의 계층구조를 생성

findeContours() 함수 예제

```
# 예제를 간단하게 작성
def contours_basic():
    src = cv.imread()

    contours, _ = cv.findContours(src, cv.RETR_LIST, cv.CHAIN_APPROX_NONE)
    # 외곽선 정보를 뽑아내기. 이때 NONE이므로 계층정보는 없어서 안받아옴.

    dst = cv.cvtColor(src, cv.COLOR_GRAY2BGR)


    # len(contours) -> 검출된 전체 외곽선 개수
    for i in range(len(contours)):
        c = (0,0,255) # 빨간색으로 고정
        cv.drawContours(dst, contours, i, c, 2)

    cv.imshow()
```

- RETR_CCOMP모드로 검출하는 예제

```
def contours_hier():
    src = cv.imread()

    contours, hierarchy = cv.findContours(src, cv.RETR_CCOMP,
cv.CHAIN_APPROX_NONE)

    dst = cv.cvtColor(src, cv.COLOR_GRAY2BGR)

    idx = 0
    while idx >= 0:
        c = (0,0,255)
        # 컬러는 R로 고정
        cv.drawContours(dst, contours, idx, c, -1, cv.LINE_8, hierarchy)
        idx = hierarchy[0,idx,0]

    cv.imshow()
```

- RETR_CCOMP로 2단계 계층 정보가 구성됨.

외곽선 처리 함수

1. boundingRect()

- 주어진 외곽선 점들을 감싸는 가장 작은 크기의 사각형 즉, 바운딩 박스를 구하고 싶다면 boundingRect()함수 사용
- 특정 객체의 바운딩 박스는 connectComponentsWithStats()로 구할 수 있지만, 이미 외곽선 정보가 있다면 boundingRect()가 효율적

2. minAreaRect()

- 외곽선 또는 점들을 감싸는 최소 크기의 회전된 사각형을 구하고 싶을때 사용
- 특정 외곽선을 감싸는 가장 작은 면적의 사각형 정보를 반환.

3. minEnclosingCircle()

- 외곽선 또는 점들을 감싸는 최소 크기의 원을 구함.

4. arcLength()

- 임의의 곡선을 형성하는 점들의 집합을 가지고 있을때, 해당 곡선의 길이를 구하고 싶을때 사용

4. contourArea()

- 임의의 외곽선 정보를 갖고 있을때 해당 외곽선이 감싸는 영역의 면적을 알고 싶을때 사용

외곽선 근사화

- 외곽선 또는 곡선을 근사화 하는 approxPolyDP()함수 제공
- 주어진 곡선의 형태를 단순화 하여 작은 개수의 점으로 구성된 곡선을 생성

-approxPolyDP() - 더글라스 포이커 알고리즘을 사용하여 곡선 또는 다각형을 근사화

- 입력 외곽선으로부터 가장 멀리 떨어져 있는 두점을 찾아 직선으로 연결하고, 또 해당 직선에서 가장 멀리 떨어진 외곽선상의 점을 찾아 근사화 점으로 추가.
- 해당 작업을 반복하다 새로운 점과 직선의 거리가 epsilon인자보다 작으면 근사화가 완료됨.
- epsilon인자는 보통 입력 외곽선 또는 곡선 길이의 일정 비율

정리

1. 가장 멀리 떨어진 두개의 점을 찾아 잇는다.
2. 이후 이 선에서 가장 먼점을 찾는다.
3. 찾고나서 입실론 인자보다 작은지 비교. 만약 작다면 끝.
4. 크다면 해당 점을 선으로 잇고 반복

외곽선 처리 함수 예제

```
def setLabel(img, pts, label):
    (x,y,w,h) = cv.boundingRect(pts)

    pt1 = (x,y)
    pt2 = (x + w , y + h)
    cv.rectangle(img, pt1, pt2, (0,0,255), 1)
    cv.putText(img, label, pt1, pt2, cv.FONT_HERSHEY_PLAIN, 1, (0,0,255))

def main():
    img = cv.imread()

    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

    _, img_bin = cv.threshold(gray, 0, 255, cv.THRESH_BINARY_INV |
cv.THRESY_OTSU)
    contours, _ = cv.findContours(img_bin, cv.RETR_EXTERNAL,
cv.CHAIN_APPROX_NONE)

    for pts in contours:
        if cv.contours(pts) < 400:
            continue
        # 400이하는 pass
        approx = cv.approxPolyDP(pts, cv.arcLength(pts, True) * 0.02,
True)

        vtc = len(approx)
        # 외곽선 점의 개수를 세준다.

        if vtc == 3:
            setLabe(img, pts, "TRI")
        else if vtc == 4:
            setLabe(img, pts, "RECT")
        else:
            lenth = cv.arcLength(pts,True)
```

```
            area = cv.contourArea(pts)
            ratio = 4. * math.pi * area / (lenth * lenth)
            # 길이, 반지름, 면적을 통해 원인지 아닌지 판별한다.

            if ratio > 0.85:
                #원이라고 판별
                setLabel(img, pts, "CIR")
        cv.imshow()
```

- 입력영상에 모든 도형의 객체의 바깥 외곽선을 찾고, 각 외곽선을 근사화.
- 근사화된 외곽선의 점의 개수로 도형을 판단.
- 삼각형도 아니고, 사각형도 아니라면 수식을 이용해 외곽선 길이와 도형의 면적비율을 통해 조사
- 비율이 1에 가까울수록 원에 가까운 모양.