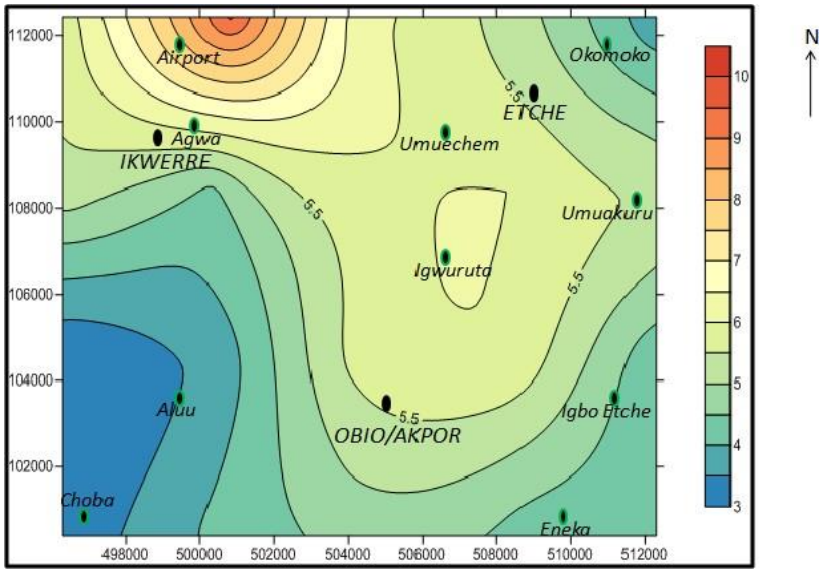
7장. 영상 분할

세부목차

Contours

❖ Contours

- 같은 색상이나 밝기의 연속된 점을 잇는 곡선
- 모양 분석, 객체 인식에 효과적
- 정확도를 높이기 위해서 Binarization 하는것이 유리
  - Threshold, Canny Edge
  - 검정 배경에서 하얀 객체 찾는 방식



[그림 7-1] 지도에서 사용하는 등고선

Contours

❖ Contours

- dst, contours, hierarchy = cv2.findContours(img, mode, method)
  - img : 입력 영상
  - mode : 결과 contour 모드
    - cv2.RETR_EXTERNAL : 가장 바깥쪽 라인만
    - cv2.RETR_LIST : 모든 라인을 계층 없이
    - cv2.RETR_CCOMP : 모든 라인을 2계층으로
    - cv2.RETR_TREE : 모든 라인의 모든 계층 정보를 트리 구조로
  - method : approximation method
    - cv2.CHAIN_APPROX_NONE : 모든 좌표 저장
    - cv2.CHAIN_APPROX_SIMPLE : 꼭지점 좌표만 저장
    - cv2.CHAIN_APPROX_TC89_L1 : Teh-Chin 근사 알고리즘
    - cv2.CHAIN_APPROX_TC89_KCOS : Teh-Chin 근사 알고리즘
  - dst : 수정된 이미지, OpenCV3.2+ (그 이전 버전엔 원본 훼손)
  - contours : 검출한 컨투어 좌표, Python List
  - hierachy : 계층 정보
    - Next, Prev, FirstChild, Parent
    - -1 : 해당 사항 없음

Contours

❖ Contours

- 주어진 contour에 따라 선을 그린다(갯수와 상관없이 모든 선 표시)
- cv2.drawContours(img, contours, contourIdx, color, thickness)
  - img : 입력 영상
  - contours : Python List
  - contourIdx : draw 대상 index, -1: all
  - color : 색상
  - thickness : 선 두께, 0 : 채우기

Contours

❖ Contours

- Example < 1/2 >

```
import cv2
import numpy as np
img = cv2.imread('../img/shapes.png')
img2 = img.copy()
# 그레이 스케일로 변환 ---①
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# 스레시홀드로 바이너리 이미지로 만들어서 검은배경에 흰색전경으로 반전 ---②
ret, imthres = cv2.threshold(imgray, 127, 255, cv2.THRESH_BINARY_INV)
# 가장 바깥쪽 컨투어에 대해 모든 좌표 반환 ---③
im2, contour, hierarchy = cv2.findContours(imthres, cv2.RETR_EXTERNAL, \
cv2.CHAIN_APPROX_NONE)
# 가장 바깥쪽 컨투어에 대해 꼭지점 좌표만 반환 ---④
im2, contour2, hierarchy = cv2.findContours(imthres, cv2.RETR_EXTERNAL, \
cv2.CHAIN_APPROX_SIMPLE)
# 각각의 컨투의 갯수 출력 ---⑤
print('도형의 갯수: %d(%d)'% (len(contour), len(contour2)))
```

[예제 7-1] 칸투어 찾기와 그리기(cntr_find.py)

Contours

❖ Contours

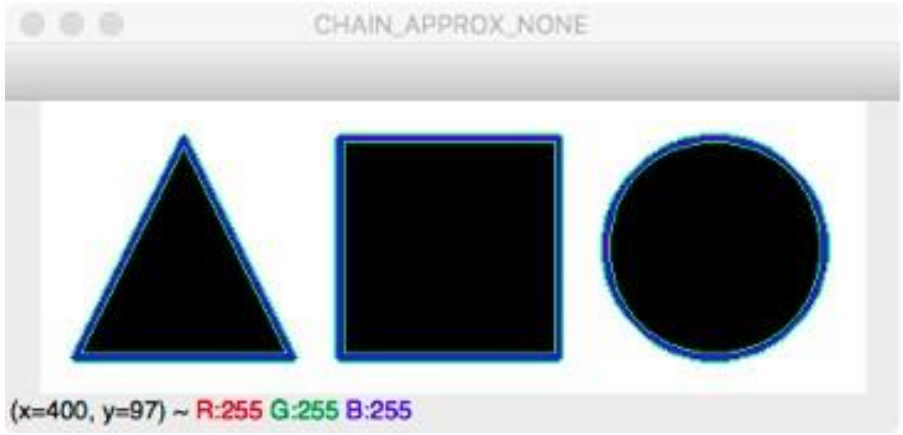
- Example < 2/2 >

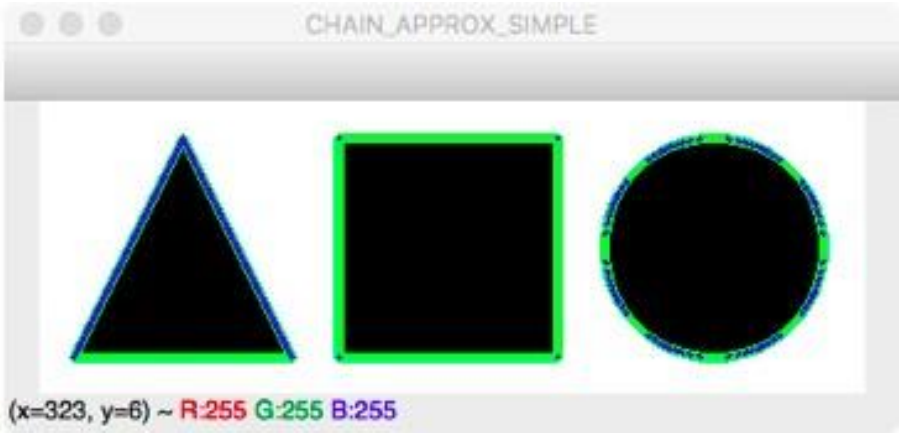
```
# 모든 좌표를 갖는 컨투어 그리기, 초록색 ---⑥
cv2.drawContours(img, contour, -1, (0,255,0), 4)
# 꼭지점 좌표만을 갖는 컨투어 그리기, 초록색 ---⑦
cv2.drawContours(img2, contour2, -1, (0,255,0), 4)
# 컨투어 모든 좌표를 작은 파랑색 점(원)으로 표시 ---⑧
for i in contour:
    for j in i:
        cv2.circle(img, tuple(j[0]), 1, (255,0,0), -1)
# 컨투어 꼭지점 좌표를 작은 파랑색 점(원)으로 표시 ---⑨
for i in contour2:
    for j in i:
        cv2.circle(img2, tuple(j[0]), 1, (255,0,0), -1)
# 결과 출력 ---⑩
cv2.imshow('CHAIN_APPROX_NONE', img)
cv2.imshow('CHAIN_APPROX_SIMPLE', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Contours

❖ Contours

- Example <결과>





[그림 7-2] [예제 7-1]의 실행 결과

Contours

❖ Contour Hierarchy

- Example < 1/2 >

```
import cv2
import numpy as np
# 영상 읽기
img = cv2.imread('../img/shapes_donut.png')
img2 = img.copy()
# 바이너리 이미지로 변환
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, imthres = cv2.threshold(imgray, 127, 255, cv2.THRESH_BINARY_INV)
# 가장 바깥 컨투어만 수집 --- ①
im2, contour, hierarchy = cv2.findContours(imthres, cv2.RETR_EXTERNAL, \
cv2.CHAIN_APPROX_NONE)
# 컨투어 갯수와 계층 트리 출력 --- ②
print(len(contour), hierarchy)
# 모든 컨투어를 트리 계층 으로 수집 ---③
im2, contour2, hierarchy = cv2.findContours(imthres, cv2.RETR_TREE, \
cv2.CHAIN_APPROX_SIMPLE)
```

[예제 7-2] 컨투어 계층 트리(cntr_hierarcy.py)

Contours

❖ Contour Hierarchy

- Example < 2/2 >

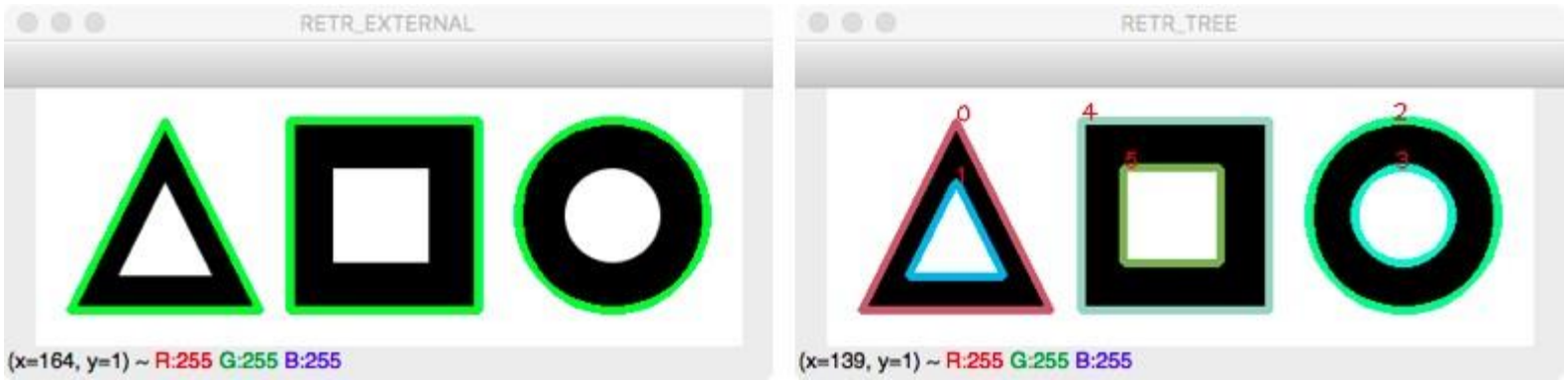
```
# 컨투어 갯수와 계층 트리 출력 ---④
print(len(contour2), hierarchy)
# 가장 바깥 컨투어만 그리기 ---⑤
cv2.drawContours(img, contour, -1, (0,255,0), 3)
# 모든 컨투어 그리기 ---⑥
for idx, cont in enumerate(contour2):
    # 랜덤한 컬러 추출 ---⑦
    color = [int(i) for i in np.random.randint(0,255, 3)]
    # 컨투어 인덱스 마다 랜덤한 색상으로 그리기 ---⑧
    cv2.drawContours(img2, contour2, idx, color, 3)
    # 컨투어 첫 좌표에 인덱스 숫자 표시 ---⑨
    cv2.putText(img2, str(idx), tuple(cont[0][0]), cv2.FONT_HERSHEY_PLAIN, \
                                                        1, (0,0,255))
# 화면 출력
cv2.imshow('RETR_EXTERNAL', img)
cv2.imshow('RETR_TREE', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Contours

❖ Contour Hierarchy

- Example <결과>

```
3 [[[ 1 -1 -1 -1]
   [ 2  0 -1 -1]
   [-1  1 -1 -1]]]
 6 [[[ 2 -1  1 -1]
   [-1 -1 -1  0]
   [ 4  0  3 -1]
   [-1 -1 -1  2]
   [-1  2  5 -1]
   [-1 -1 -1  4]]]
```



[그림 7-3] [예제 7-2]의 실행 결과

Contours

❖ Image Moment

- Moment
  - 물리학, 힘의 양(물리량 X 거리)을 기술하는 용어
- Image Moment
  - 대상 물체의 양적인 속성을 표현
    - $m_{p,q} = \sum_x \sum_y f(x,y) x^p y^q$
  - 컨투어가 둘러 싸는 영역 x,y 좌표의 픽셀 값과 좌표 인덱스의 p,q 차수 곱의 합
  - 바이너리 이미지 : 0이 아닌 모든 값은 1로 계산
  - p, q 차수 : 0 ~3으로 바꿔가며 계산, 의미있는 정보 획득
- m00 : 컨투어의 면적
  - 모든 수의 차수 0은 1, 영역의 갯수 만큼 1로 곱하여 합
- 중심 모멘트
    - $\mu_{p,q} = \sum_x \sum_y f(x,y)(x-\bar{x})^p (y-\bar{y})^q$
    - $\bar{x} = \frac{m_{10}}{m_{00}}$
    - $\bar{y} = \frac{m_{01}}{m_{00}}$

Contours

❖ Image Moment

- moment = cv2.moments(contour)
  - contour : 모멘트 계산 대상 컨투어 좌표
  - moment : 결과 모멘트, 파이썬 딕셔너리
    - m00, m01, m10, m11, m02, m12, m20, m21, m03, m30 : 공간 모멘트
    - mu20, mu11, mu02, mu30, mu21, mu12, mu03 : 중심 모멘트
    - nu20, nu11, nu02, nu30, nu21, nu03 : 정규화 중심 모멘트
- retval = cv.contourArea(contour[, oriented=False]) : 컨투어로 넓이 계산
  - contour : 넓이를 계산할 컨투어
  - oriented : 컨투어 방향 플래그
    - True : 컨투어 방향에 따라 음수 반환
    - False : 절대값 반환
  - retval : 컨투어 영역의 넓이 값
- retval = cv.arcLength(curve, closed) : 컨투어로 둘레의 길이 계산
  - curve : 둘레 길이를 계산할 컨투어
  - closed : 닫힌 호인지 여부 플래
  - retval : 컨투어의 둘레 길이 값

Contours

❖ Image Moment

- 중심점, 넓이, 둘레길이 Example < 1/2 >

```
mport cv2
import numpy as np
img = cv2.imread("../img/shapes.png")
# 그레이 스케일 변환
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# 바이너리 스케일 변환
ret, th = cv2.threshold(imgray, 127, 255, cv2.THRESH_BINARY_INV)
# 컨투어 찾기
img2, contours, hierachy = cv2.findContours(th, cv2.RETR_EXTERNAL, \
                                                cv2.CHAIN_APPROX_SIMPLE)
# 각 도형의 컨투어에 대한 루프
for c in contours:
    # 모멘트 계산
    mmt = cv2.moments(c)
    # m10/m00, m01/m00 중심점 계산
    cx = int(mmt['m10']/mmt['m00'])
    cy = int(mmt['m01']/mmt['m00'])
```

[예제 7-3] 모멘트를 이용한 중심점, 넓이, 둘레길이(contr_moment.py)

Contours

❖ Image Moment

- 중심점, 넓이, 둘레길이 Example < 2/2 >

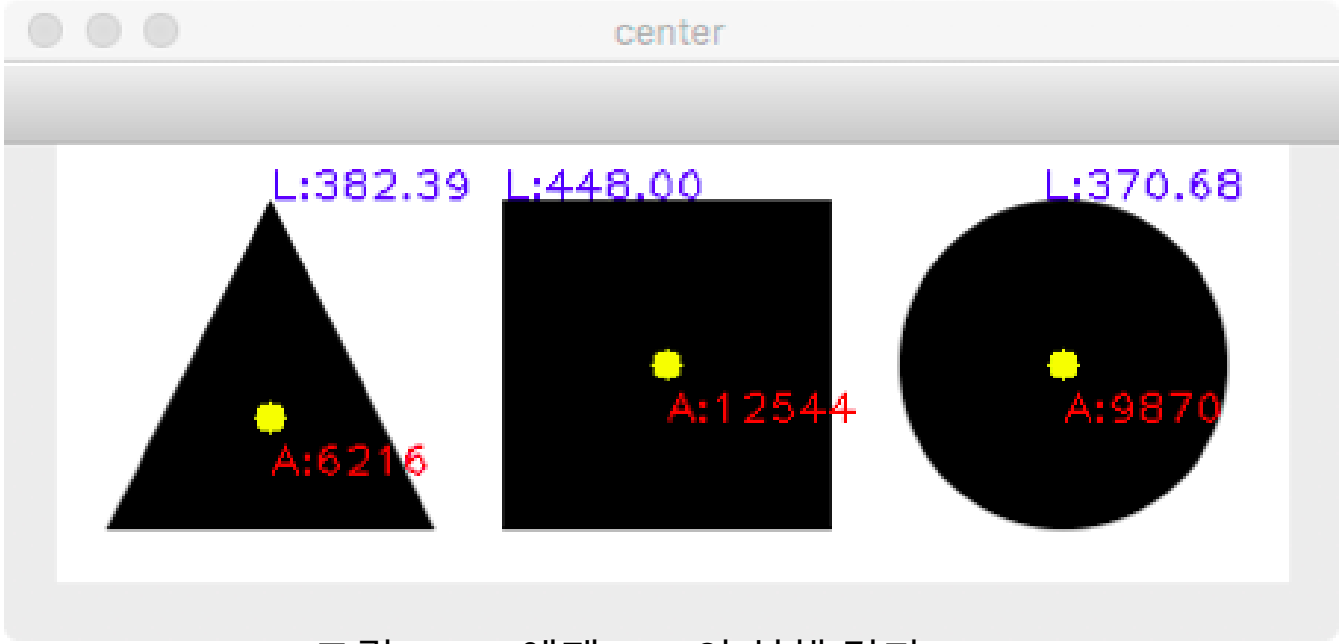
```
        # 영역 넓이
        a = mmt['m00']
        # 영역 외곽선 길이
        l = cv2.arcLength(c, True)
        # 중심점에 노란색 점 그리기
        cv2.circle(img, (cx, cy), 5, (0, 255, 255), -1)
        # 중심점 근처에 넓이 그리기
        cv2.putText(img, "A:%.0f"%a, (cx, cy+20) , cv2.FONT_HERSHEY_PLAIN, \
                                                        1, (0,0,255))
        # 컨투어 시작점에 길이 그리기
        cv2.putText(img, "L:%.2f"%l, tuple(c[0][0]), cv2.FONT_HERSHEY_PLAIN, \
                                                        1, (255,0,0))
        # 함수로 컨투어 넓이 계산해서 출력
        print("area:%.2f"%cv2.contourArea(c, False))
# 결과 출력
cv2.imshow('center', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Contours

❖ Image Moment

- 중심점, 넓이, 둘레길이 Example <결과>

```
area:9870.00
area:12544.00
area:6216.00
```



[그림 7-4] [예제 7-3]의 실행 결과

Contours

❖ Bounding Box

- x,y,w,h = cv2.boundingRect(contour) : 좌표를 감싸는 사각형 구하기
  - x, y : 사각형 좌 상단 좌표
  - w, h : 폭, 높이
- rotateRect = cv2.minAreaRect(contour) : 좌표를 감싸는 최소한의 사각형
  - rotateRect : 회전한 사각형 좌표
    - center : 중심점 (x,y)
    - size : 크기 (w, h)
    - angle : 회전 각, 양수:시계 방향, 음수:반시계 방향
- vertex = cv2.boxPoints(rotateRect) : rotateRect로 부터 꼭지점 좌표
  - vertex : 4개의 꼭지점 좌표, 소수점 포함, 정수 변환 필요
- center, radius = cv2.minEnclosingCircle(contour) :
  - 좌표를 감싸는 최소한의 동그라미
  - center : 원점 좌표(x, y), 튜플
  - radius : 반지름
- area, triangle = cv.minEnclosingTriangle(points) : 좌표를 감싸는 최소한의 삼각형
  - area : 넓이
  - triangle : 3개의 꼭지점 좌표

Contours

❖ Bounding Box

- ellipse = cv.fitEllipse(points) : 좌표를 감싸는 최소한의 타원 계산
  - ellipse
    - center : 원점 좌표(x,y),튜플
    - axes : 축의 길이(x축, y축), 튜플
    - angle : 회전 각도
- line = cv.fitLine(points, distType, param, reps, aeps[, line]) : 중심점을 통과하는 직선 계산
  - distType : 거리 계산 방식
    - cv2.DIST_L2, cv2.DIST_L1, cv2.DIST_L12, cv2.DIST_FAIR, cv2.DIST_WELSCH, cv2.DIST_HUBER
  - param : distType에 전달할 인자, 0=최적값 선택
  - reps : 반지름 정확도, 선과 원본 좌표의 거리, 0.01 권장
  - aeps : 각도 정확도, 0.01 권장
  - line
    - vx, vy : 정규화된 단위 벡터, 직선의 기울기, 튜플
    - x0, y0 : 중심점 좌표, 튜플

Contours

❖ Bounding Shapes

- Example < 1/2 >

```
import cv2
import numpy as np
# 이미지 읽어서 그레이스케일 변환, 바이너리 스케일 변환
img = cv2.imread("../img/lightning.png")
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, th = cv2.threshold(imgray, 127,255,cv2.THRESH_BINARY_INV)
# 컨튜어 찾기
im, contours, hr = cv2.findContours(th, cv2.RETR_EXTERNAL, \
                                    cv2.CHAIN_APPROX_SIMPLE)
contr = contours[0]
# 감싸는 사각형 표시(검정색)
x,y,w,h = cv2.boundingRect(contr)
cv2.rectangle(img, (x,y), (x+w, y+h), (0,0,0), 3)
# 최소한의 사각형 표시(초록색)
rect = cv2.minAreaRect(contr)
box = cv2.boxPoints(rect) # 중심점과 각도를 4개의 꼭지점 좌표로 변환
box = np.int0(box) # 정수로 변환
cv2.drawContours(img, [box], -1, (0,255,0), 3)
```

[예제 7-4] 컨투어를 감싸는 도형 그리기(cntr_bound_fit.py)

Contours

❖ Bounding Shapes

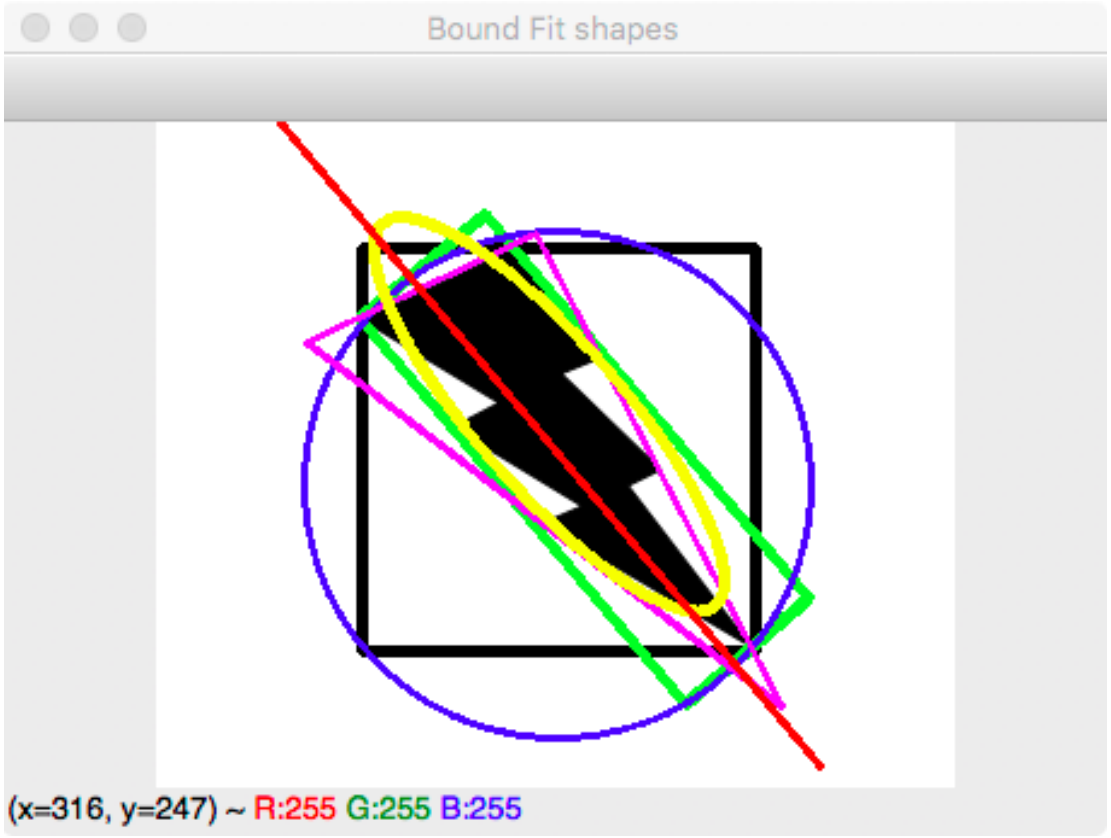
- Example < 2/2 >

```
# 최소한의 원 표시(파랑색)
(x,y), radius = cv2.minEnclosingCircle(contr)
cv2.circle(img, (int(x), int(y)), int(radius), (255,0,0), 2)
# 최소한의 삼각형 표시(분홍색)
ret, tri = cv2.minEnclosingTriangle(contr)
cv2.polylines(img, [np.int32(tri)], True, (255,0,255), 2)
# 최소한의 타원 표시(노랑색)
ellipse = cv2.fitEllipse(contr)
cv2.ellipse(img, ellipse, (0,255,255), 3)
# 중심점 통과하는 직선 표시(빨강색)
[vx,vy,x,y] = cv2.fitLine(contr, cv2.DIST_L2,0,0.01,0.01)
cols,rows = img.shape[:2]
cv2.line(img,(0, 0-x*(vy/vx) + y), (cols-1, (cols-x)*(vy/vx) + y), \
(0,0,255),2)
# 결과 출력
cv2.imshow('Bound Fit shapes', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Contours

❖ Bounding Box

- Example <결과>

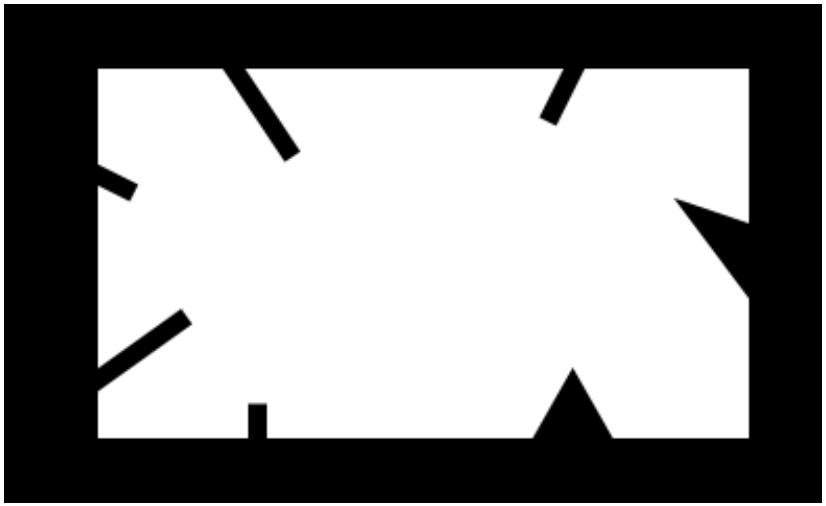


[그림 7-5] [예제 7-4]의 실행 결과

Contours

❖ Contour Approximation

- 정확한 Contour가 오히려 부정확 한 경우
  - Noise, 침식
- Douglas-Peucker 알고리즘
  - epsilon 값 지정: contour에서 근사 contour 까지의 최대 거리
- approx = cv2.approxPolyDP(contour, epsilon, closed)
  - contour : 대상 컨투어 좌표
  - epsilon : 근사 값 정확도, 오차범위
  - closed : 컨투어가 닫힘 여부
  - approx : 근사 계산한 컨투어 좌표



Contours

❖ Contour Approximation

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/bad_rect.png')
img2 = img.copy()

# 그레이스케일과 바이너리 스케일 변환
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, th = cv2.threshold(imgray, 127, 255, cv2.THRESH_BINARY)

# 컨투어 찾기 ---①
temp, contours, hierachy = cv2.findContours(th, cv2.RETR_EXTERNAL, \
                                                cv2.CHAIN_APPROX_SIMPLE)
contour = contours[0]
# 전체 둘레의 0.05로 오차 범위 지정 ---②
epsilon = 0.05 * cv2.arcLength(contour, True)
```

[예제 7-5] 근사 컨투어(cntr_approximate.py)

Contours

❖ Contour Approximation

- Example

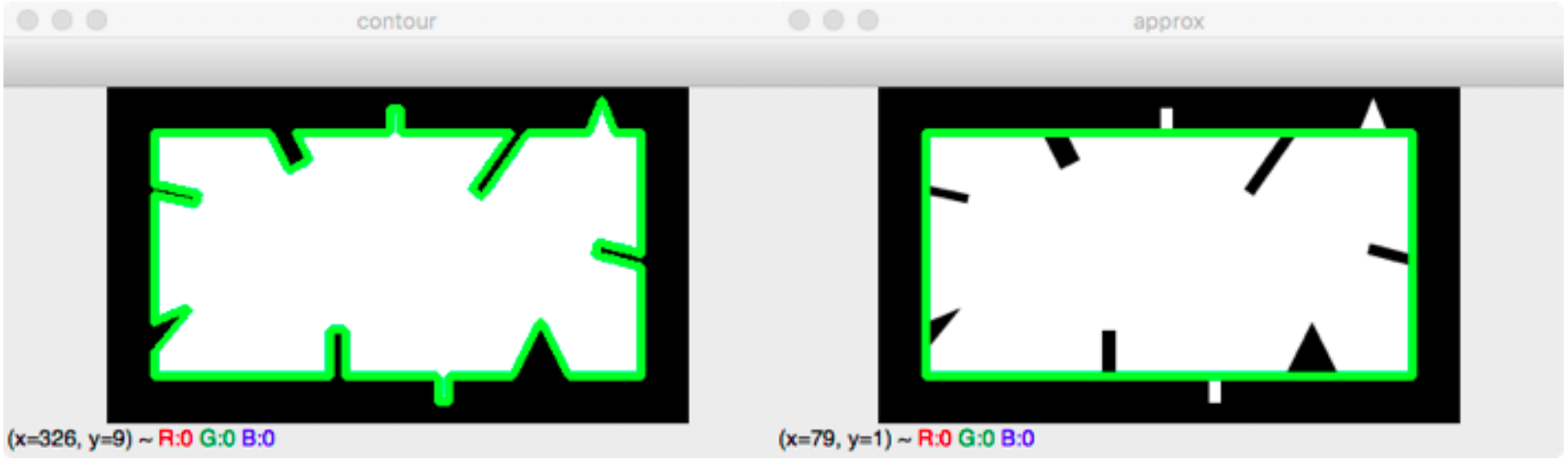
```
# 근사 컨투어 계산 ---③
approx = cv2.approxPolyDP(contour, epsilon, True)

# 각각 컨투어 선 그리기 ---④
cv2.drawContours(img, [contour], -1, (0,255,0), 3)
cv2.drawContours(img2, [approx], -1, (0,255,0), 3)

# 결과 출력
cv2.imshow('contour', img)
cv2.imshow('approx', img2)
cv2.waitKey()
cv2.destroyAllWindows()
```

Contours

- **Contour Approximation**
  - Example <결과>



[그림 7-6] [예제 7-5]의 실행 결과

Contours

❖ Convex Hull

- 볼록선체 : 오목한 부분이 없고 볼록 곡선으로만 구성된 도형(물체)
- 물체의 외곽 영역 좌표
- 물체의 실제 차지하는 영역 및 중심점 찾기에 효과적
- hull = cv2.convexHull(points, hull, clockwise, returnPoints)
  - points : contours
  - hull : output, avoid
  - clockwise : 방향지정, True/False
  - returnPoints :
    - True*: hull points 좌표 반환
    - False : contour 중 hull 해당 인덱스 반환
- ret = cv2.isContourConvex(contour) : True/False
  - ret : Convex 여부 확인

Contours

❖ Convex Hull

- defects = cv2.convexityDefects(contour, convexhull) : 볼록 선체 결함 찾기
  - contour : 입력 컨투어
  - convexhull : 볼록 선체에 해당하는 컨투어의 인덱스
  - defects : 볼록 선체 결함이 있는 컨투어의 배열 인덱스, N x 1 x 4 배열
    - [start, end, farthest, distance]
      - start : 오목한 각이 시작되는 컨투어의 인덱스
      - end : 오목한 각이 끝나는 컨투어의 인덱스
      - farthest : 볼록 선체에서 가장 먼 오목한 지점의 컨투어 인덱스
      - distance : farthest와 볼록 선체와의 거리
        - 8비트 고정 소수점(distance/256.0)

Contours

❖ Convex Hull

- Example <1/2>

```
import cv2
import numpy as np
img = cv2.imread('../img/hand.jpg')
img2 = img.copy()
# 그레이 스케일 및 바이너리 스케일 변환 ---①
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, th = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY_INV)
# 컨투어 찾기와 그리기 ---②
temp, contours, heiarchy = cv2.findContours(th, cv2.RETR_EXTERNAL, \
                                                cv2.CHAIN_APPROX_SIMPLE)
cntr = contours[0]
cv2.drawContours(img, [cntr], -1, (0, 255,0), 1)
# 볼록 선체 찾기(좌표 기준)와 그리기 ---③
hull = cv2.convexHull(cntr)
cv2.drawContours(img2, [hull], -1, (0,255,0), 1)
# 볼록 선체 만족 여부 확인 ---④
print(cv2.isContourConvex(cntr), cv2.isContourConvex(hull))
# 볼록 선체 찾기(인덱스 기준) ---⑤
hull2 = cv2.convexHull(cntr, returnPoints=False)
```

[예제 7-6] 블록 선체(cntr_convexhull.py)

Contours

❖ Convex Hull

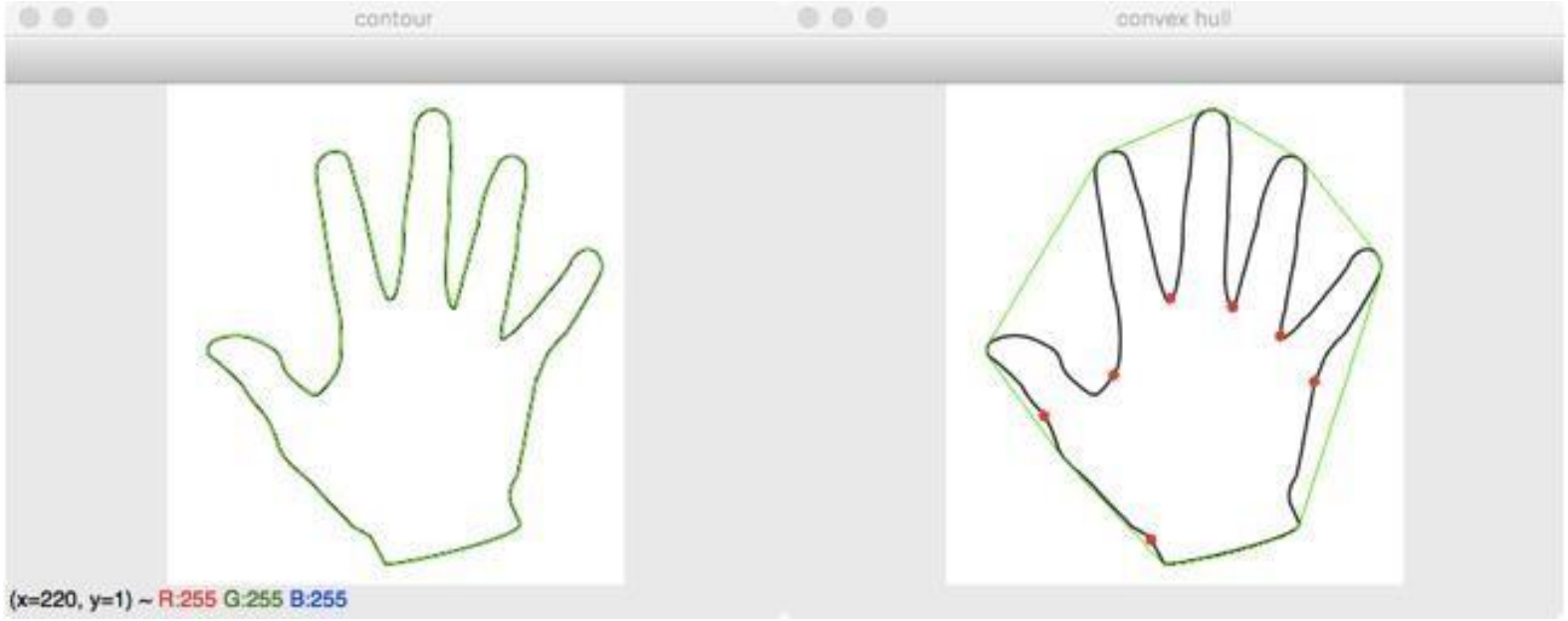
- Example <2/2>

```
# 볼록 선체 결함 찾기 ---⑥
defects = cv2.convexityDefects(cntr, hull2)
# 볼록 선체 결함 순회
for i in range(defects.shape[0]):
    # 시작, 종료, 가장 먼 지점, 거리 ---⑦
    startP, endP, farthestP, distance = defects[i, 0]
    # 가장 먼 지점의 좌표 구하기 ---⑧
    farthest = tuple(cntr[farthestP][0])
    # 거리를 부동 소수점으로 변환 ---⑨
    dist = distance/256.0
    # 거리가 1보다 큰 경우 ---⑩
    if dist > 1 :
        # 빨강색 점 표시
        cv2.circle(img2, farthest, 3, (0,0,255), -1)
# 결과 이미지 표시
cv2.imshow('contour', img)
cv2.imshow('convex hull', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Contours

❖ Convex Hull

- Example



[그림 7-7] [예제 7-6]의 실행 결과

Contours

❖ Point Polygon Test

- 이미지의 어느점에서 contour 까지의 거리 계산
  - cv2.pointPolygonTest(img, pt, measureDist)
    - img : 입력 영상
    - pt : (x,y) , 측정한 지점
    - measureDist : True/False
      - True : 거리 계산
      - False : 내/외 부만 표현(-1,0,1)
    - 반환
      - 양수 : 내부
      - 음수 : 외부

Contours

❖ Point Polygon Test

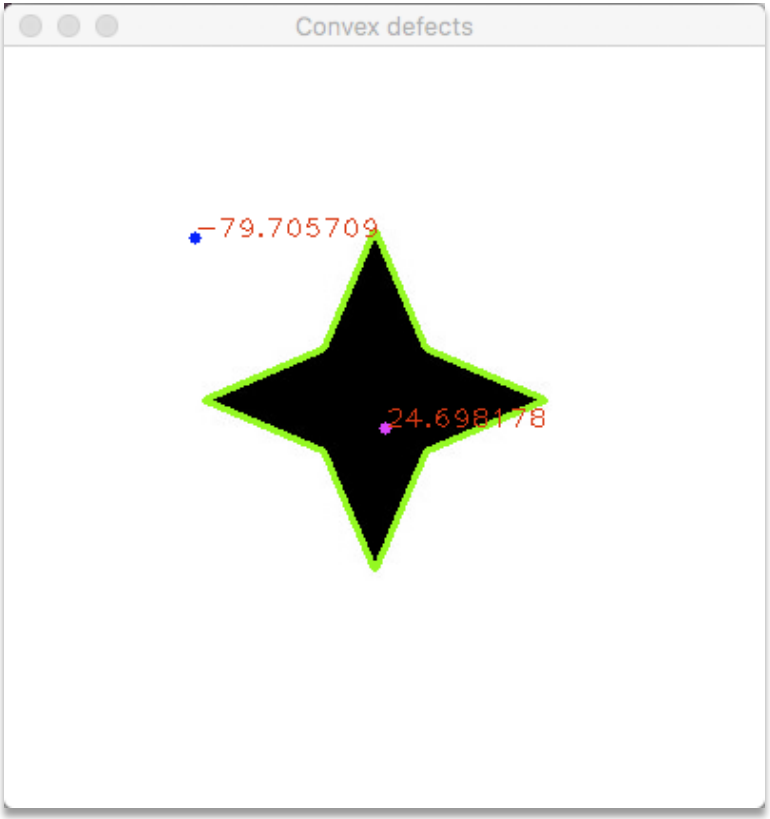
- Example

```
img = cv2.imread('../img/4star.jpg')
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, th = cv2.threshold(imgray, 127,255, cv2.THRESH_BINARY_INV)
_, contours, _ = cv2.findContours(th, cv2.RETR_EXTERNAL, ₩
                                  cv2.CHAIN_APPROX_SIMPL
                                  E)
contour = contours[0]
cv2.drawContours(img, [contour], 0, (0,255,0),2)
p1 = (100,100)
p2 = (200,200)
cv2.circle(img, p1, 3, (255,0,0), -1)
cv2.circle(img, p2, 3, (255,0,255), -1)
dist1 = cv2.pointPolygonTest(contour, p1, True)
dist2 = cv2.pointPolygonTest(contour, p2, True)
cv2.putText(img, '%f'%dist1,   p1,                          1, (0,0,255),   1)
                               cv2.FONT_HERSHEY_PLAIN,
cv2.putText(img, '%f'%dist2,   p2,                          1, (0,0,255),   1)
                               cv2.FONT_HERSHEY_PLAIN,
cv2.imshow('Convex defects', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Contours

❖ Point Polygon Test

- Example <결과>



Contours

❖ Match Shape

- 서로 다른 2개의 contour 비교
- ret = cv2.matchShape(cntr1, cntr2, method, param)
  - cntr1, cntr2 : 비교 대상 contour
  - method : Hu Moment 비교 알고리즘
    - cv2.CONTOURS_MATCH_I1
    - cv2.CONTOURS_MATCH_I2
    - cv2.CONTOURS_MATCH_I3
  - param : 알고리즘 전달 인수, 0.0 (not supported)
  - ret:
    - 닮음 정도 : 작은 수 일수록 높은 닮음, 0=동일한 모양

Contours

❖ Match Shape

- Example <1/2>

```
import cv2
import numpy as np
# 매칭을 위한 이미지 읽기
target = cv2.imread('../img/4star.jpg') # 매칭 대상
shapes = cv2.imread('../img/shapestomatch.jpg') # 여러 도형
# 그레이 스케일 변환
targetGray = cv2.cvtColor(target, cv2.COLOR_BGR2GRAY)
shapesGray = cv2.cvtColor(shapes, cv2.COLOR_BGR2GRAY)
# 바이너리 스케일 변환
ret, targetTh = cv2.threshold(targetGray, 127, 255, cv2.THRESH_BINARY_INV)
ret, shapesTh = cv2.threshold(shapesGray, 127, 255, cv2.THRESH_BINARY_INV)
# 컨투어 찾기
_, cntrs_target, _ = cv2.findContours(targetTh, cv2.RETR_EXTERNAL, \
                                      cv2.CHAIN_APPROX_SIMPLE)
_, cntrs_shapes, _ = cv2.findContours(shapesTh, cv2.RETR_EXTERNAL, \
                                      cv2.CHAIN_APPROX_SIMPLE)
```

[예제 7-7] 도형 매칭으로 비슷한 도형 찾기(contr_matchShape.py)

Contours

❖ Match Shape

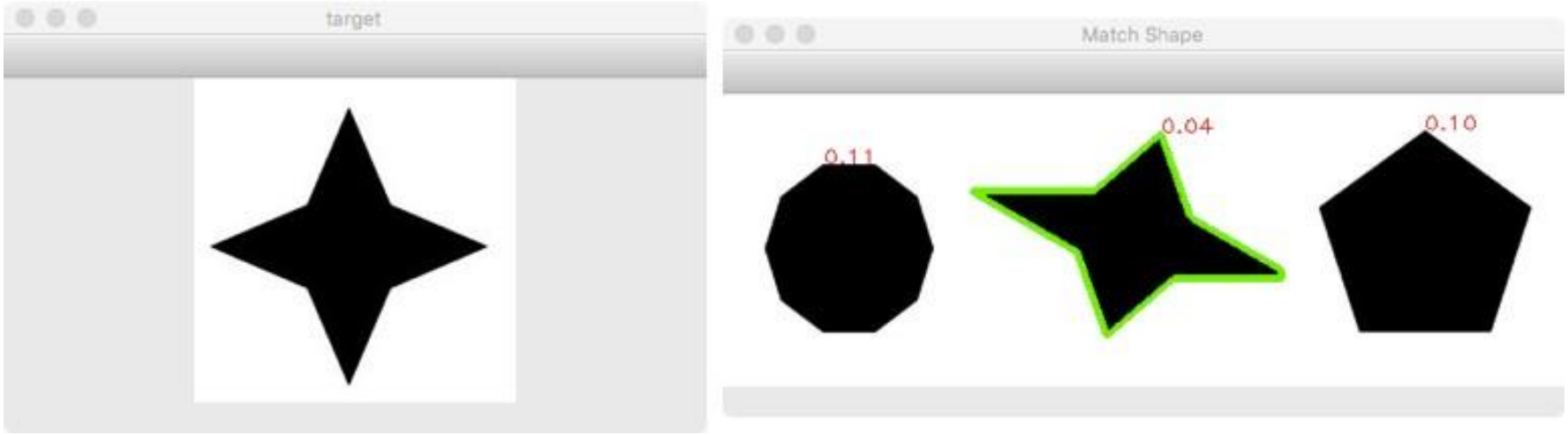
- Example <2/2>

```
# 각 도형과 매칭을 위한 반복문
matchs = [] # 컨투어와 매칭 점수를 보관할 리스트
for contr in cntrs_shapes:
    # 대상 도형과 여러 도형 중 하나와 매칭 실행 ---①
    match = cv2.matchShapes(cntrs_target[0], contr, cv2.CONTOURS_MATCH_I2, 0.0)
    # 해당 도형의 매칭 점수와 컨투어를 쌍으로 저장 ---②
    matchs.append( (match, contr) )
    # 해당 도형의 컨투어 시작지점에 매칭 점수 표시 ---③
    cv2.putText(shapes, '%.2f'%match, tuple(contr[0][0]),\
                cv2.FONT_HERSHEY_PLAIN, 1,(0,0,255),1 )
# 매칭 점수로 정렬 ---④
matchs.sort(key=lambda x : x[0])
# 가장 적은 매칭 점수를 얻는 도형의 컨투어에 선 그리기 ---⑤
cv2.drawContours(shapes, [matchs[0][1]], -1, (0,255,0), 3)
cv2.imshow('target', target)
cv2.imshow('Match Shape', shapes)
cv2.waitKey()
cv2.destroyAllWindows()
```

Contours

❖ Match Shape

- Example <결과>



[그림 7-8] [예제 7-7]의 실행 결과

세부목차

1. Contour
2. **Hough Transform**
3. Connected Component
4. Workshop

Hough Transform

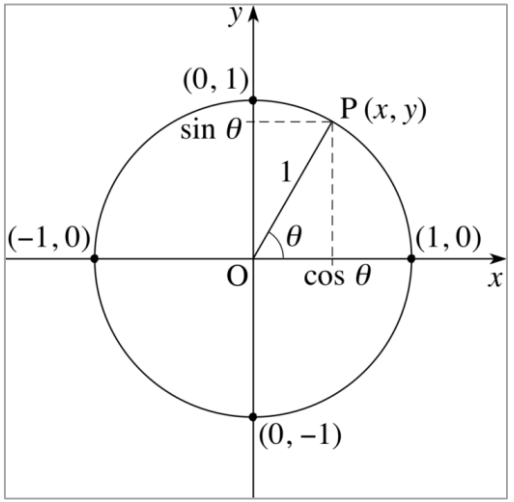
❖ Hough Transform

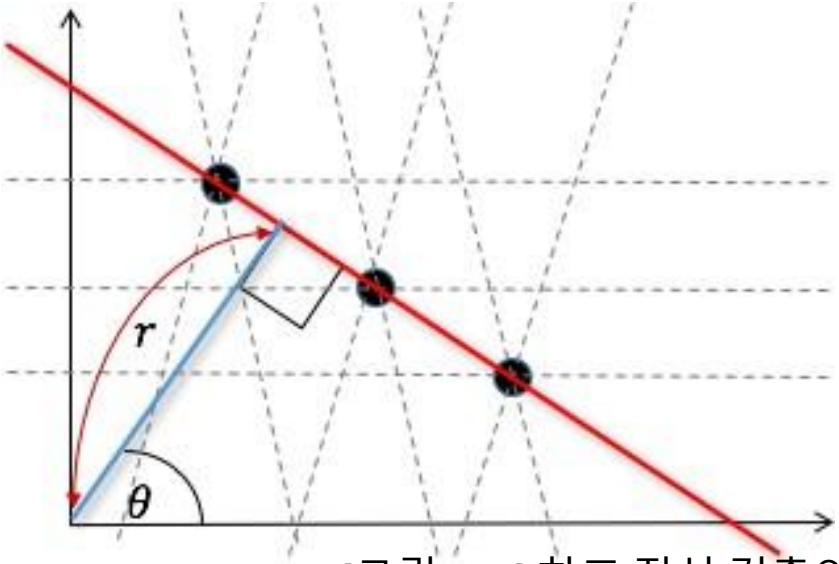
- 허프 변환
- 직선, 원, 간단한 모양 식별
- 원래 직선 찾는 방법으로 시작해서 다양한 모양 인식에 확장
- Paul Hough 최초 발견(bubble chamber), 1962
- Richard Duda and Peter Hart 일반화, 1972
- Dana H. Ballard에 의해 Computer Vision 영역에 대중화, 1981
- Hough Line Transform
- Hough Circle Transform

Hough Transform

❖ Hough Line Transform

- 영상의 수 많은 점들 중 직선 검출
- $r = x\cos\theta + y\sin\theta$
- 한 점에 대한 (r,θ)
- 모든 점들 중 동일한 (r,θ) 을 갖는 점들이 직선





[그림 7-9] 하프 직선 검출의 예

Hough Transform

❖ Hough Line Transform

- lines = cv2.HoughLines(img, rho, theta, threshold[, lines, srn=0, stn=0, min_theta, max_theta])
  - img : 입력 영상, 1채널 바이너리 스케일
  - rho : 거리 측정 해상도, 0~1
  - theta : 각도 측정 해상도, 라디언 단위(np.pi/0 ~180)
  - threshold : 직선으로 판단할 최소한의 동일 갯수
    - 작은 값 : 정확도 감소, 검출 갯수 증가
    - 큰 값 : 정확도 증가, 검출 갯수 감소
  - lines : 검출 결과, N x 1 x 2 배열(r,θ)
  - srn, stn : 멀티 스케일 허프 변환에 사용, 선 검출에서는 사용 안함
  - min_theta, max_theta : 검출을 위해 사용할 최대 최소 각도

Hough Transform

❖ Hough Line Transform

- Example

```
import cv2
import numpy as np

img = cv2.imread('../img/sudoku.jpg')
img2 = img.copy()
h, w = img.shape[:2]
# 그레이 스케일 변환 및 엣지 검출 ---①
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(imgray, 100, 200 )
# 허프 선 검출 ---②
lines = cv2.HoughLines(edges, 1, np.pi/180, 130)
for line in lines: # 검출된 모든 선 순회
    r,theta = line[0] # 거리와 각도wh
    tx, ty = np.cos(theta), np.sin(theta) # x, y축에 대한 삼각비
    x0, y0 = tx*r, ty*r #x, y 기준(절편) 좌표
```

[예제 7-8] 허프 선 검출(hough_line.py)

Hough Transform

❖ Hough Line Transform

- Example

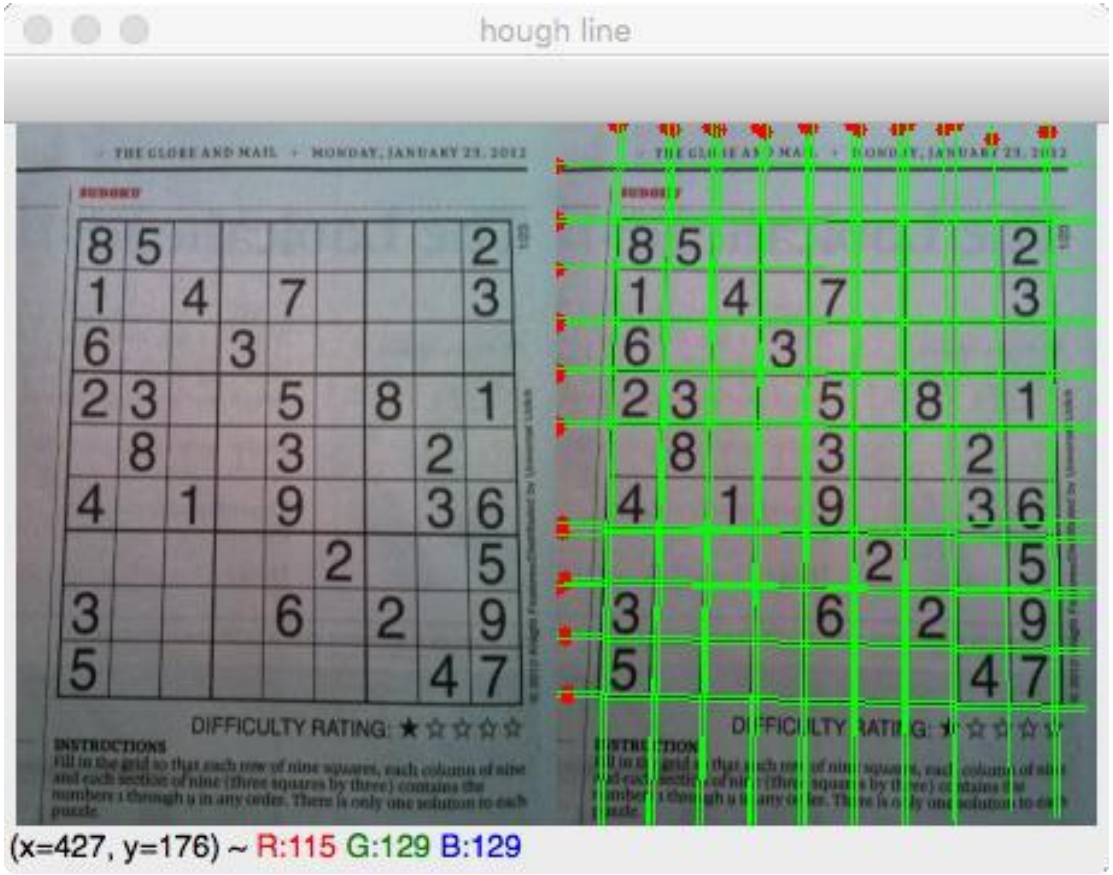
```
# 기준 좌표에 빨강색 점 그리기
    cv2.circle(img2, (abs(x0), abs(y0)), 3, (0,0,255), -1)
    # 직선 방정식으로 그리기 위한 시작점, 끝점 계산
    x1, y1 = int(x0 + w*(-ty)), int(y0 + h * tx)
    x2, y2 = int(x0 - w*(-ty)), int(y0 - h * tx)
    # 선그리기
    cv2.line(img2, (x1, y1), (x2, y2), (0,255,0), 1)

#결과 출력
merged = np.hstack((img, img2))
cv2.imshow('hough line', merged)
cv2.waitKey()
cv2.destroyAllWindows()
```

Hough Transform

❖ Hough Line Transform

- Example <결과>



[그림 7-10] [예제 7-8]의 실행 결과

Hough Transform

❖ Probabilistic Hough Line Transform

- 확율적 허프 직선 변환
- 모든 픽셀에 대해서 연산하는 비효율성 회피
- 무작위 픽셀 선택
- lines = cv2.HoughLineP(src, rho, theta, threshold, minLineLen, maxLineGap)
  - src : 입력 이미지
  - rho : 거리 정확도 (0~1)
  - theta : 각도 정확도 , radian (np.pi/0~180)
  - threshold : 직선으로 판단한 최소한의 동일 값 갯수
  - minLineLen : 길이가 이 보다 작으면 탈락
  - maxLineGap : 거리가 이 보다 크면 서로 다른 직선
  - lines : 검출된 선 좌표
    - 직선의 2점에 대한 nx1x 4 배열(x1, y1, x2, y2)

Hough Transform

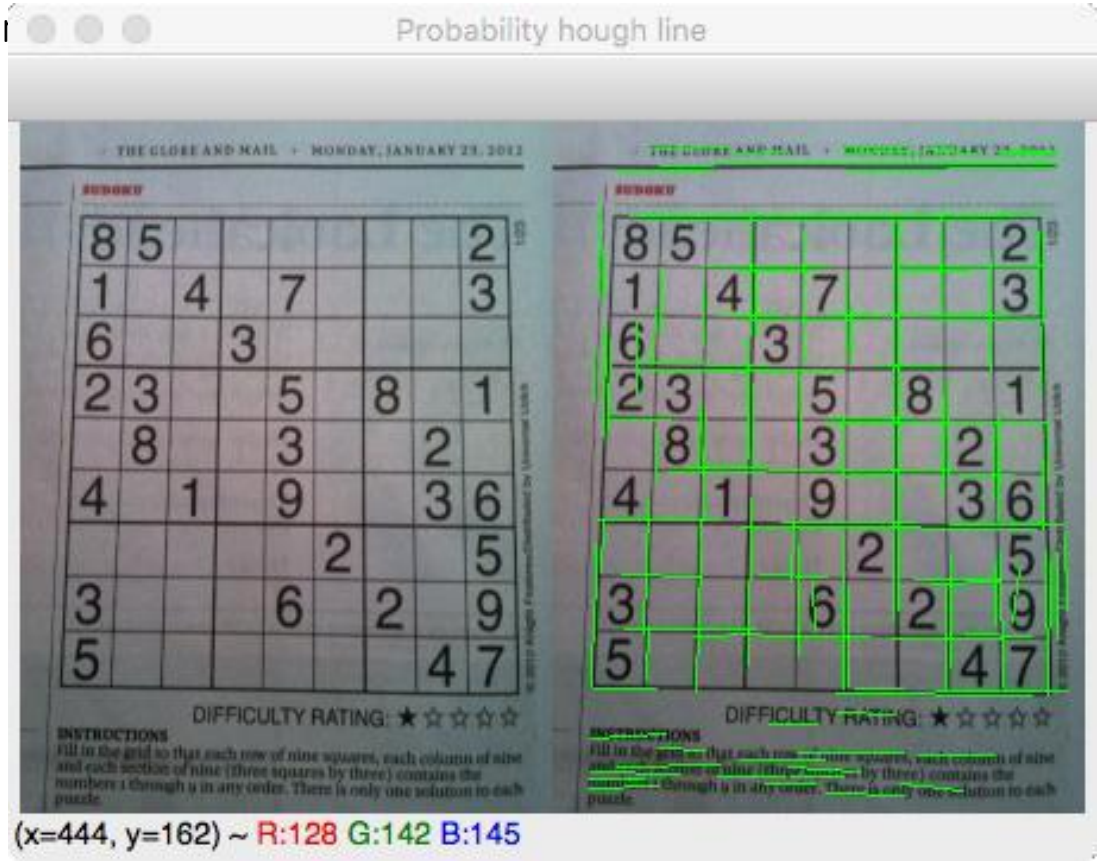
- **Probabilistic Hough Line Transform**

```
import cv2
import numpy as np
img = cv2.imread('../img/sudoku.jpg')
img2 = img.copy()
# 그레이 스케일로 변환 및 엣지 검출 ---①
imgray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(imgray, 50, 200 )
# 확율 허프 변환 적용 ---②
lines = cv2.HoughLinesP(edges, 1, np.pi/180, 10, None, 20, 2)
for line in lines:
    # 검출된 선 그리기 ---③
    x1, y1, x2, y2 = line[0]
    cv2.line(img2, (x1,y1), (x2, y2), (0,255,0), 1)
merged = np.hstack((img, img2))
cv2.imshow('Probability hough line', merged)
cv2.waitKey()
cv2.destroyAllWindows()
```

[예제 7-9] 확률 허프 변환으로 선 검출(hough_lineP.py)

Hough Transform

- **Probabilistic Hough Line Transform**
  - Exa



[그림 7-11] [예제 7-9]의 실행 결과

Hough Transform

❖ Hough Circle Transform

- 허프 원 검출 연산
  - 직교 좌표를 극 좌표 변환
  - 직선 변환의 알고리즘으로 원 검출 가능
  - 연산이 많아 속도 저하
- Hough Gradient Method
  - OpenCV 구현 방법
  - Canny로 edge 검출
  - Sobel로 Gradient 방향 계산
  - 원의 중심과 반지름 계산, 3차원 accumulator
  - Sobel을 이용해서 Gradient 방향을 구하기 때문에 부정확
  - 중첩된 동심원 검출 어려움

Hough Transform

❖ Hough Circle Transform

- circles = cv2.HoughCircle(src, method, dp, minDst, circles, param1, param2, minRds, maxRds)
  - src : 입력 영상
  - method : cv2.HOUGH_GRADIENT, only available
    - cv.HOUGH_STANDARD
    - cv.HOUGH_PROBABILISTIC
    - cv.HOUGH_MULTI_SCALE
    - cv.HOUGH_GRADIENT
  - dp : accumulator와 입력 이미지간의 해상도 반비례율
    - 1: 동일, 2: ½, 작은 값 : 정확도 증가, 큰값 : 검출 갯수 증가
  - minDst : 원들 중심간의 최소 거리, 0=err, 동심원 검출 불가
  - circles : 검출 원 결과, N x 1 x 3 배열(x, y, r)
  - param1 : Canny Edge에 사용할 maxValue
  - param2 : HUGH_GRADENT threshold ,
    - 작은 값 : 검출 갯수 증가, 정확도 감소
  - minRadius, maxRadius : 원의 최소 반지름, 최대 반지름(0이면 영상의 크기)

Hough Transform

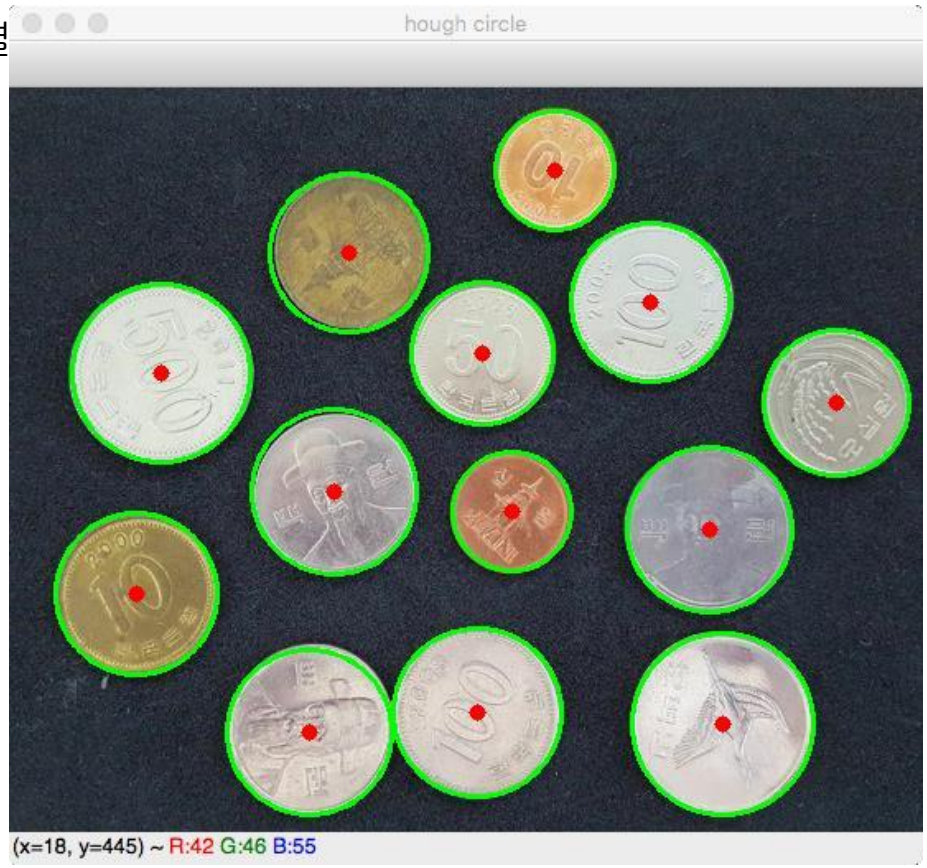
❖ Hough Circle Transform

- Example [예제 7-10] 허프 원 검출(hough_circle.py)

```
import cv2
import numpy as np
img = cv2.imread('../img/coins_spread1.jpg')
# 그레이 스케일 변환 ---①
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# 노이즈 제거를 위한 가우시안 블러 ---②
blur = cv2.GaussianBlur(gray, (3,3), 0)
# 허프 원 변환 적용( dp=1.5, minDist=30, cany_max=200 ) ---③
circles = cv2.HoughCircles(blur, cv2.HOUGH_GRADIENT, 1.5, 30, None, 200)
if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0,:]:
        # 원 둘레에 초록색 원 그리기
        cv2.circle(img,(i[0], i[1]), i[2], (0, 255, 0), 2)
        # 원 중심점에 빨강색 원 그리기
        cv2.circle(img, (i[0], i[1]), 2, (0,0,255), 5)
# 결과 출력
cv2.imshow('hough circle', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Hough Transform

- **Hough Circle Transform**
  - Example <결



[그림 7-12] [예제 7-10]의 실행 결과

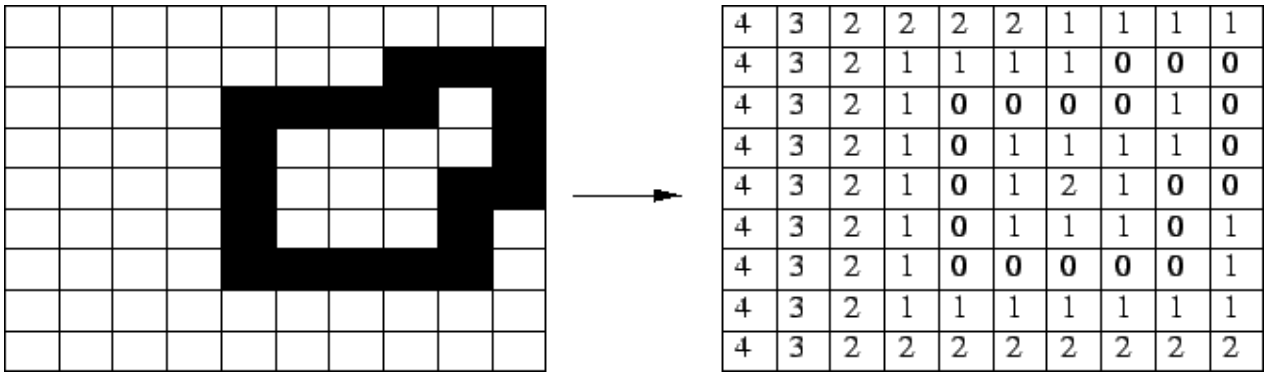
세부목차

1. Contour
2. Hough Transform
3. **Connected Component**
4. Workshop

Connected Component

❖ Distance Transform

- 물체의 최 중심점 찾기
- 경계로 부터 가장 멀리 떨어진 곳
- 바이너리 스케일 이미지 대상
- 픽셀 값이 0인 위치에서 0에서 시작해서 멀어 질때 마다 1씩 증가
- cv2.distanceTransform(src, distanceType, maskSize)
  - src : 입력 영상, 바이너리 스케일
  - distanceType : 거리 계산 방식 선택
    - cv2.DIST_L2, cv2.DIST_L1, cv2.DIST_L12, cv2.DIST_FAIR, cv2.DIST_WELSCH, cv2.DIST_HUBER
  - maskSize : 거리 변환 커널 크기



[그림 7-13] 거리 변환 알고리즘

Connected Component

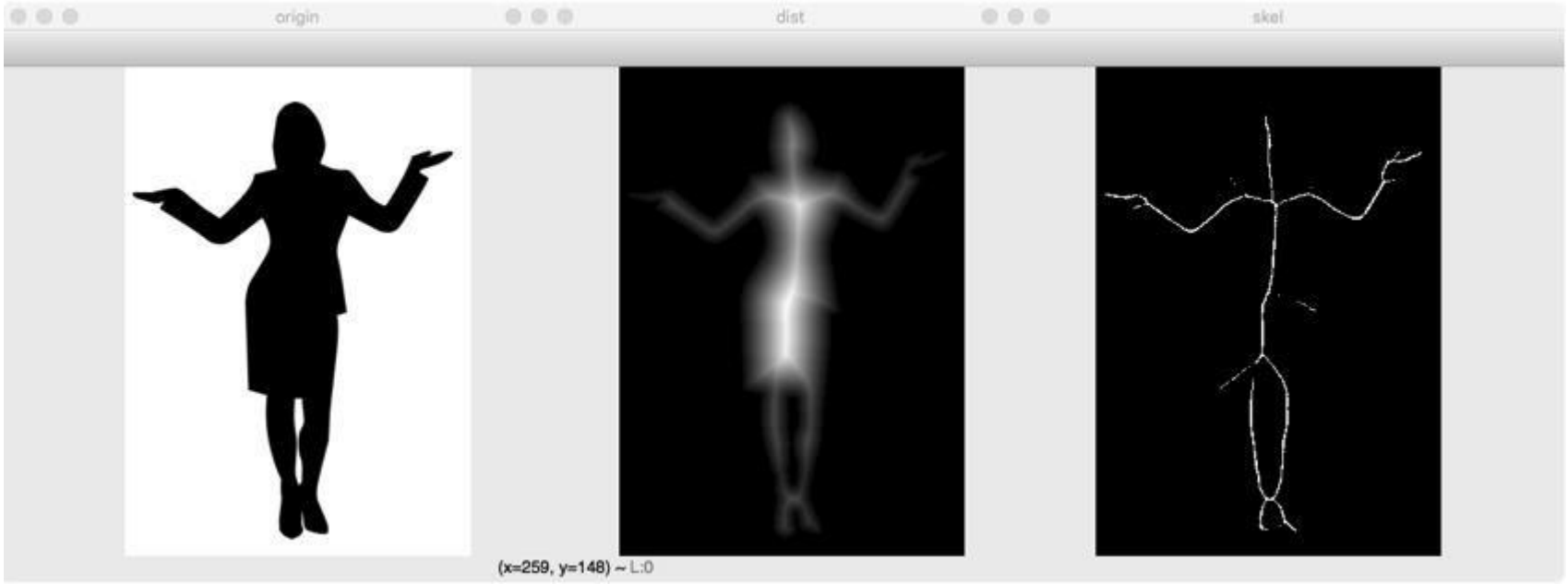
❖ Distance Transform

```
import cv2
import numpy as np
# 이미지를 읽어서 바이너리 스케일로 변환
img = cv2.imread('../img/full_body.jpg', cv2.IMREAD_GRAYSCALE)
_, biimg = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY_INV)
# 거리 변환 ---①
dst = cv2.distanceTransform(biimg, cv2.DIST_L2, 5)
# 거리 값을 0 ~ 255 범위로 정규화 ---②
dst = (dst/(dst.max()-dst.min()) * 255).astype(np.uint8)
# 거리 값에 쓰레시홀드로 완전한 뼈대 찾기 ---③
skeleton = cv2.adaptiveThreshold(dst, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, \
                                                    cv2.THRESH_BINARY, 7, -3)
# 결과 출력
cv2.imshow('origin', img)
cv2.imshow('dist', dst)
cv2.imshow('skel', skeleton)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[예제 7-11] 거리 변환으로 전신 스켈레톤 찾기(distanceTrans.py)

Connected Component

❖ Distance Transform

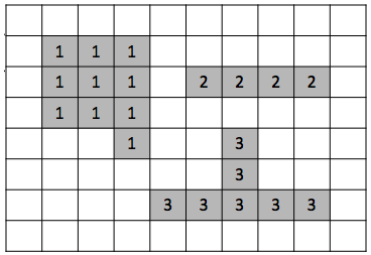


[그림 7-14] [예제 7-11]의 실행 결과

Connected Component

❖ Labeling

- 연결된 요소들 끼리 분리
- 픽셀 값이 0으로 끊어 지지 않는 여역 끼리 같은 레이블 값 부여
- retval, labels = cv.connectedComponents(src[,labels, connectivity=8, ltype]) : 연결 요소 레이블링과 갯수 반환
  - src : 입력 영상, 바이너리 스케일 이미지
  - labels : 레이블링 된 입력 영상과 같은 크기의 배열
  - connectivity : 연결성 검사할 방향 갯수, 4, 8 중 선택
  - ltype : 결과 레이블 배열 dtype
  - retval : 레이블 갯수
- retval, labels, stats, centroids = cv.connectedComponentsWithStats(src[, labels, stats, centroids, connectivity, ltype]) : 레이블링과 각종 상태 정보 반환
  - stats : N x 5 행렬(N:레이블 갯수)
    - [x좌표, y좌표, 폭, 높이, 넓이]
  - centroids : 각 레이블의 중심점 좌표, N x 2 행렬(N:레이블 갯수



[그림 7-15] 연결 요소 레이블링 개념도

Connected Component

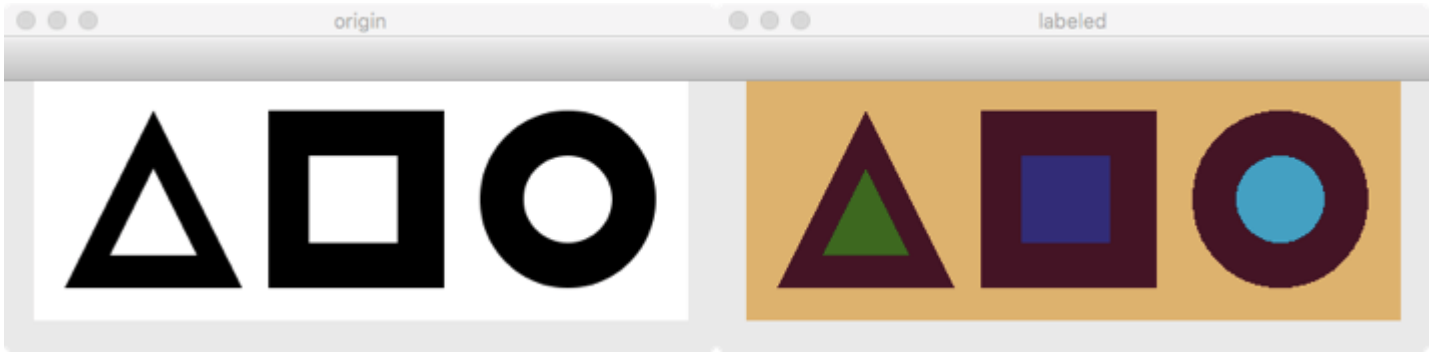
❖ Labeling

```
import cv2
import numpy as np
# 이미지 읽기
img = cv2.imread('../img/shapes_donut.png')
# 결과 이미지 생성
img2 = np.zeros_like(img)
# 그레이 스케일과 바이너리 스케일 변환
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
_, th = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
# 연결된 요소 레이블링 적용 ---①
cnt, labels = cv2.connectedComponents(th)
#retval, labels, stats, cent = cv2.connectedComponentsWithStats(th)
# 레이블 갯수 만큼 순회
for i in range(cnt):
    # 레이블이 같은 영역에 랜덤한 색상 적용 ---②
    img2[labels==i] = [int(j) for j in np.random.randint(0,255, 3)]
# 결과 출력
cv2.imshow('origin', img)
cv2.imshow('labeled', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[예제 7-12] 연결된 영역 레이블링(connected_label.py)

Connected Component

❖ Labeling



[그림 7-16] [예제 7-12]의 실행 결과

Connected Component

❖ Flood Fill

- 연속된 영역에 같은 색상 채우기
- retval, img, mask, rect = cv2.floodFill(img, mask, seed, newVal[, loDiff, upDiff, flags])
  - img : 입력 영상, 1 또는 3채널
  - mask : 입력 영상 보다 2x2픽셀이 더 큰 배열, 0이 아닌 영역을 만나면 채우기 중지
  - seed : 채우기 시작할 좌표
  - newVal : 채우기에 사용할 색상 값
  - loDiff, upDiff : 채우기 결정할 최소 최대 차이 값
  - flags : 채우기 방식 선택 플래그,
    - 4 또는 8 방향 채우기
    - cv2.FLODDFILL_MASK_ONLY : img가 아닌 mask에만 채우기 적용
      - 채우기에 사용할 값을 8~16 비트에 포함 필요
    - cv2.FLOODFILL_FIXED_RANGE : 이웃픽셀이 아닌 seed 픽셀과 비교
  - retval : 채우기한 픽셀의 갯수
  - rect : 채우기가 이뤄진 영역을 감싸는 사각형

Connected Component

❖ Flood Fill

- $src(x', y') - \text{loDiff} \leq src(x, y) \leq src(x', y') + \text{upDiff}$
  - $src(x, y)$: 현재 픽셀
  - $src(x', y')$: 이웃 픽셀

```
import cv2
import numpy as np
img = cv2.imread('../img/taekwonv1.jpg')
rows, cols = img.shape[:2]
# 마스크 생성, 원래 이미지 보다 2픽셀 크게 ---①
mask = np.zeros((rows+2, cols+2), np.uint8)
# 채우기에 사용할 색 ---②
newVal = (255,255,255)
# 최소 최대 차이 값 ---③
loDiff, upDiff = (10,10,10), (10,10,10)
# 마우스 이벤트 처리 함수
def onMouse(event, x, y, flags, param):
    global mask, img
    if event == cv2.EVENT_LBUTTONDOWN:
        seed = (x,y)
        # 색 채우기 적용 ---④
        retval = cv2.floodFill(img, mask, seed, newVal, loDiff, upDiff)
        # 채우기 변경 결과 표시 ---⑤
        cv2.imshow('img', img)
# 화면 출력
cv2.imshow('img', img)
cv2.setMouseCallback('img', onMouse)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[예제 7-13] 마우스로 색 채우기(flood_fill.py)

Connected Component

❖ Flood Fill

[그림 7-17] [예제 7-13]의 실행 결과

Connected Component

❖ Watershed

- Edge를 산등성이, 균일한 영역을 계곡으로 취급하여 객체 분할 수행
- 사용자(또는 알고리즘)에 의해 분리할 객체를 Marker 지정
- 서로 다른 마커 영역에 물을 채워서 서로 만날때 멈춘다.
- Marker
  - 서로 다른 객체마다 ID(1~)를 부여, 분리 대상을 표시
  - 0 : Unknown
  - 예) 전경=1, 배경=2
  - 예)오렌지=1, 라임=2, 배경=3
- markers = cv2.watershed(img, markers)
  - img : 입력 영상
  - markers
    - 경계가 -1로 셋팅된 마커, 입력과 같은 크기

Connected Component

❖ Watershed

- Example <1/3>

```
import cv2
import numpy as np
img = cv2.imread('../img/taekwonv1.jpg')
rows, cols = img.shape[:2]
img_draw = img.copy()
# 마커 생성, 모든 요소는 0으로 초기화 ---①
marker = np.zeros((rows, cols), np.int32)
markerId = 1 # 마커 아이디는 1에서 시작
colors = [] # 마커 선택한 영역 색상 저장할 공간
isDragging = False # 드래그 여부 확인 변수
# 마우스 이벤트 처리 함수
def onMouse(event, x, y, flags, param):
    global img_draw, marker, markerId, isDragging
    if event == cv2.EVENT_LBUTTONDOWN: # 왼쪽 마우스 버튼 다운, 드래그 시작
        isDragging = True
        # 각 마커의 아이디와 현 위치의 색상 값을 쌍으로 매핑해서 저장
        colors.append((markerId, img[y,x]))
```

[예제 7-14] 마우스와 워터셰드로 배경 분리(watershed.py)

❖ Watershed

- Example <2/3>

```
elif event == cv2.EVENT_MOUSEMOVE: # 마우스 움직임
        if isDragging: # 드래그 진행 중
            # 마우스 좌표에 해당하는 마커의 좌표에 동일한 마커 아이디로 채워 넣기 ---②
            marker[y,x] = markerId
            # 마커 표시한 곳을 빨강색점으로 표시해서 출력
            cv2.circle(img_draw, (x,y), 3, (0,0,255), -1)
            cv2.imshow('watershed', img_draw)
    elif event == cv2.EVENT_LBUTTONUP: # 왼쪽 마우스 버튼 업
        if isDragging:
            isDragging = False # 드래그 중지
            # 다음 마커 선택을 위해 마커 아이디 증가 ---③
            markerId +=1
    elif event == cv2.EVENT_RBUTTONDOWN: # 오른쪽 마우스 버튼 누름
            # 모아 놓은 마커를 이용해서 워터 쉐드 적용 ---④
            cv2.watershed(img, marker)
            # 마커에 -1로 표시된 경계를 초록색으로 표시 ---⑤
            img_draw[marker == -1] = (0,255,0)
```

❖ Watershed

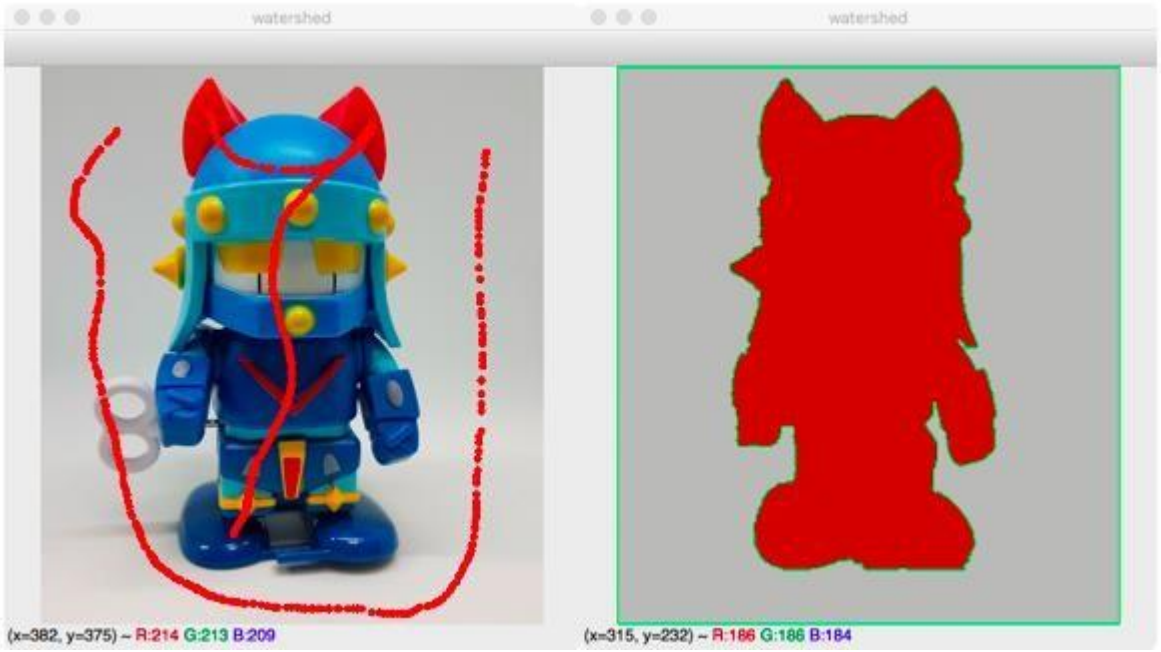
- Example <3/3>

```
            for mid, color in colors: # 선택한 마커 아이디 갯수 만큼 반복
                # 같은 마커 아이디 값을 갖는 영역을 마커 선택한 색상으로 채우기 ---⑥
                img_draw[marker==mid] = color
            cv2.imshow('atershed', img_draw) # 표시한 결과 출력
# 화면 출력
cv2.imshow('watershed', img)
cv2.setMouseCallback('watershed', onMouse)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Connected Component

❖ Watershed

- Example <결과>



[그림 7-18] [예제 7-14]의 실행 결과

Connected Component

❖ Grabcut

- Graph Cut 알고리즘을 기반으로 확장
- 전경으로 분리 할 대상 좌표로 색상 분포를 추정
- 동일한 레이블을 가진 연결된 영역에서 배경과 전경 분리

Connected Component

❖ Grabcut

- mask, bgdModel, fgdModel = cv2.grabCut(img, mask, rect, bgdModel, fgdModel, iterCount[, mode])
  - img : 입력 영상
  - mask : 입력 영상과 크기가 같은 1채널 배열, 배경과 전경을 구분하는 값 저장
    - cv2.GC_BGD : 확실한 배경(0)
    - cv2.GC_FGD : 확실한 전경(1)
    - cv2.GC_PR_BGD : 아마도 배경(2)
    - cv2.GC_PR_FGD : 아마도 배경(3)
  - rect : 전경이 있을 것으로 추측되는 영역의 사각형 좌표, 튜플 (x1, y1, x2, y2)
  - bgdModel, fgdModel : 함수내에서 사용할 임시 배열 버퍼, 재 사용할 경우 수정하지 말것
  - iterCount : 반복 횟수
  - mode : 동작 방법
    - cv2.GC_INIT_WITH_RECT : rect에 지정한 좌표를 기준으로 그랩컷 수행
    - cv2.GC_INIT_WITH_MASK : mask에 지정한 값을 기준으로 그랩컷 수행
    - cv2.GC_EVAL : 재시도

Connected Component

❖ Grabcut

- Example <1/3>

```
import cv2
import numpy as np
img = cv2.imread('../img/taekwonv1.jpg')
img_draw = img.copy()
mask = np.zeros(img.shape[:2], dtype=np.uint8) # 마스크 생성
rect = [0,0,0,0] # 사각형 영역 좌표 초기화
mode = cv2.GC_EVAL # 그랩컷 초기 모드
# 배경 및 전경 모델 버퍼
bgdmodel = np.zeros((1,65),np.float64)
fgdmodel = np.zeros((1,65),np.float64)
# 마우스 이벤트 처리 함수
def onMouse(event, x, y, flags, param):
    global mouse_mode, rect, mask, mode
    if event == cv2.EVENT_LBUTTONDOWN : # 왼쪽 마우스 누름
        if flags <= 1: # 아무 키도 안 눌렀으면
            mode = cv2.GC_INIT_WITH_RECT # 드래그 시작, 사각형 모드 ---①
            rect[:2] = x, y # 시작 좌표 저장
```

[예제 7-15] 마우스와 그랩컷으로 배경 분리(grabcut.py)

Connected Component

❖ Grabcut

- Example <2/3>

```
        # 마우스가 움직이고 왼쪽 버튼이 눌러진 상태
        elif event == cv2.EVENT_MOUSEMOVE and flags & cv2.EVENT_FLAG_LBUTTON :
            if mode == cv2.GC_INIT_WITH_RECT: # 드래그 진행 중 ---②
                img_temp = img.copy()
                # 드래그 사각형 화면에 표시
                cv2.rectangle(img_temp, (rect[0], rect[1]), (x, y), (0,255,0), 2)
                cv2.imshow('img', img_temp)
            elif flags > 1: # 키가 눌러진 상태
                mode = cv2.GC_INIT_WITH_MASK # 마스크 모드 ---③
                if flags & cv2.EVENT_FLAG_CTRLKEY :# 컨트롤 키, 분명한 전경
                    # 흰색 점 화면에 표시
                    cv2.circle(img_draw,(x,y),3, (255,255,255),-1)
                    # 마스크에 GC_FGD로 채우기 ---④
                    cv2.circle(mask,(x,y),3, cv2.GC_FGD,-1)
                if flags & cv2.EVENT_FLAG_SHIFTKEY : # 쉬프트키, 분명한 배경
                    # 검정색 점 화면에 표시
                    cv2.circle(img_draw,(x,y),3, (0,0,0),-1)
                    # 마스크에 GC_BGD로 채우기 ---⑤
                    cv2.circle(mask,(x,y),3, cv2.GC_BGD,-1)
                cv2.imshow('img', img_draw) # 그려진 모습 화면에 출력
```

Connected Component

❖ Grabcut

- Example <3/3>

```
        elif event == cv2.EVENT_LBUTTONUP: # 마우스 왼쪽 버튼 뗀 상태 ---⑥
            if mode == cv2.GC_INIT_WITH_RECT : # 사각형 그리기 종료
                rect[2:] =x, y # 사각형 마지막 좌표 수집
                # 사각형 그려서 화면에 출력 ---⑦
                cv2.rectangle(img_draw, (rect[0], rect[1]), (x, y), (255,0,0), 2)
                cv2.imshow('img', img_draw)
            # 그랩컷 적용 ---⑧
            cv2.grabCut(img, mask, tuple(rect), bgdmodel, fgdmodel, 1, mode)
            img2 = img.copy()
            # 마스크에 확실한 배경, 아마도 배경으로 표시된 영역을 0으로 채우기
            img2[(mask==cv2.GC_BGD) | (mask==cv2.GC_PR_BGD)] = 0
            cv2.imshow('grabcut', img2) # 최종 결과 출력
            mode = cv2.GC_EVAL # 그랩컷 모드 리셋
    # 초기 화면 출력 및 마우스 이벤트 등록
    cv2.imshow('img', img)
    cv2.setMouseCallback('img', onMouse)
    while True:
        if cv2.waitKey(0) & 0xFF == 27 : # esc
            break
    cv2.destroyAllWindows()
```

Connected Component

❖ Grabcut

- Example <2/3>

```
elif   event == cv2.EVENT_MOUSEMOVE and flags
       &
           if mode == cv2.GC_INIT_WITH_RECT:
                mg_temp = img.copy()
                gle(img_temp, (rect[0],
                w('img', img_temp)
           elif flags > 1:
                mode = cv2.GC_INIT_WITH_MASK
                if flags
                :
                if flags & cv2.EVENT_FLAG_SHIFTKEY :
                     cv2.circle(img_draw,(x,y),3, (0,0,0),-1)
                     cv2.circle(mask,(x,y),3, cv2.GC_BGD,-1)
                cv2.imshow('img', img_draw)
```

```
cv2.EVENT_FLAG_LBUTT
ON :
 cv2.rectan  rect[1]), (x, y), (0,255,0), 2)
 cv2.imsho
& cv2.EVENT_FLAG_CTRLKEY
     cv2.circle(img_draw,(x,y),3,(255,255,255),-1)
     cv2.circle(mask,(x,y),3, cv2.GC_FGD,-1)
```

Connected Component

❖ Grabcut

- Example <결과>

[그림 7-19] [예제 7-15]의 실행 결과

Connected Component

❖ MeanShift Filter

- 일정한 반경 크기의 커널 윈도우로 픽셀 평균 값으로 중심 이동
- 이동을 시작한 지점에서 중지한 지점까지 하나로 묶어 연결 영역 찾기
- 빈도가 가장 높은 색상으로 연결 영역 색상값 변경

- dst = cv.pyrMeanShiftFiltering(src, sp, sr[, dst, maxLevel, termcrit])
  - src : 입력 영상
  - sp : 공간 윈도우 반지름 크기
  - sr : 색상 윈도우 반지름 크기
  - maxLevel : 이미지 피라미드 최대 레벨
  - termcrit : 평균이동 중지 기준

Connected Component

❖ MeanShift Filter

- Example

```
import cv2
import numpy as np
img = cv2.imread('../img/taekwonv1.jpg')
# 트랙바 이벤트 처리 함수
def onChange(x):
    #sp, sr, level 선택 값 수집
    sp = cv2.getTrackbarPos('sp', 'img')
    sr = cv2.getTrackbarPos('sr', 'img')
    lv = cv2.getTrackbarPos('lv', 'img')
    # 평균 이동 필터 적용 ---①
    mean = cv2.pyrMeanShiftFiltering(img, sp, sr, None, lv)
    # 변환 이미지 출력
    cv2.imshow('img', np.hstack((img, mean)))
# 초기 화면 출력
cv2.imshow('img', np.hstack((img, img)))
# 트랙바 이벤트 함수 연결
cv2.createTrackbar('sp', 'img', 0,100, onChange)
cv2.createTrackbar('sr', 'img', 0,100, onChange)
cv2.createTrackbar('lv', 'img', 0,5, onChange)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

[예제 7-16] 평균 이동 세그멘테이션 필터(mean_shift.py)

❖ MeanShift Filter

- Example <결과>

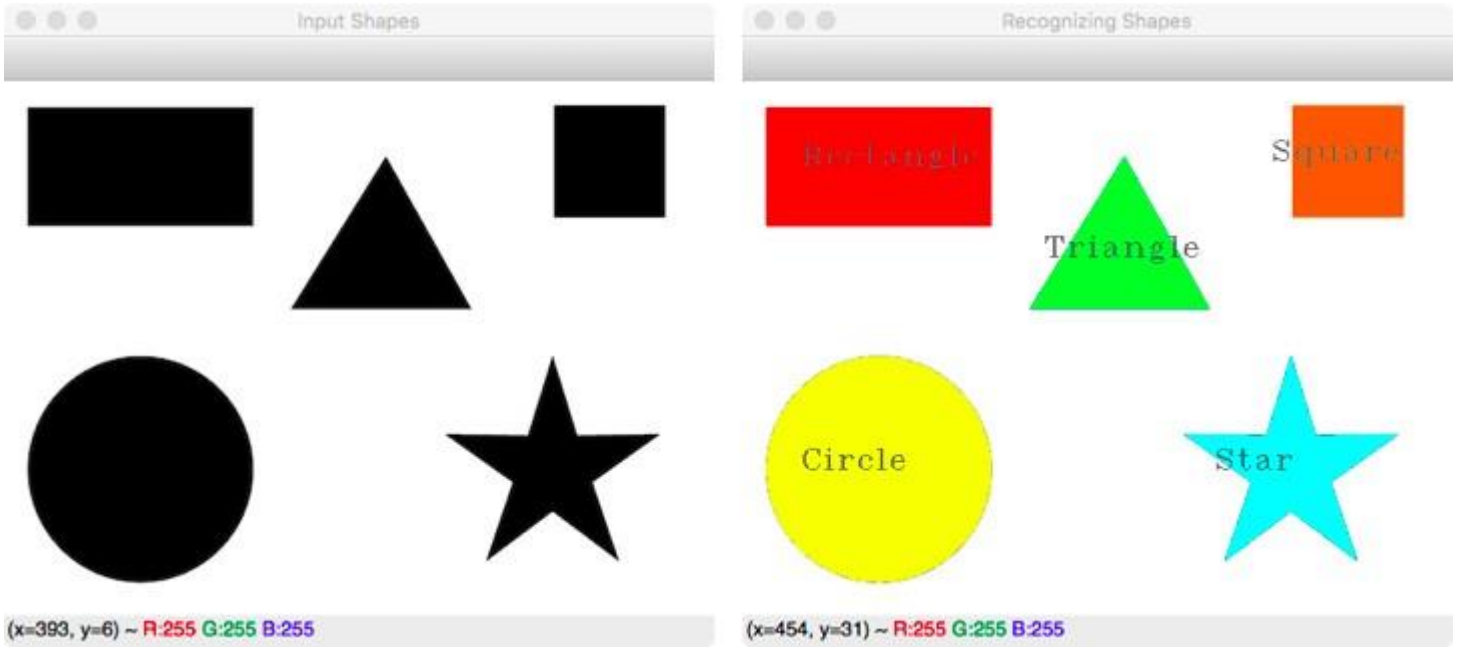
[그림 7-20] [예제 7-16]의 실행 결과

세부목차

1. Contour
2. Hough Transform
3. Connected Component
4. **Workshop**

❖ Recognizing Shapes

- 5개의 도형이 들어 있는 영상에서 가각 도형의 이름을 알아 마추는 프로그램을 만드세요.
  - 결과 예시

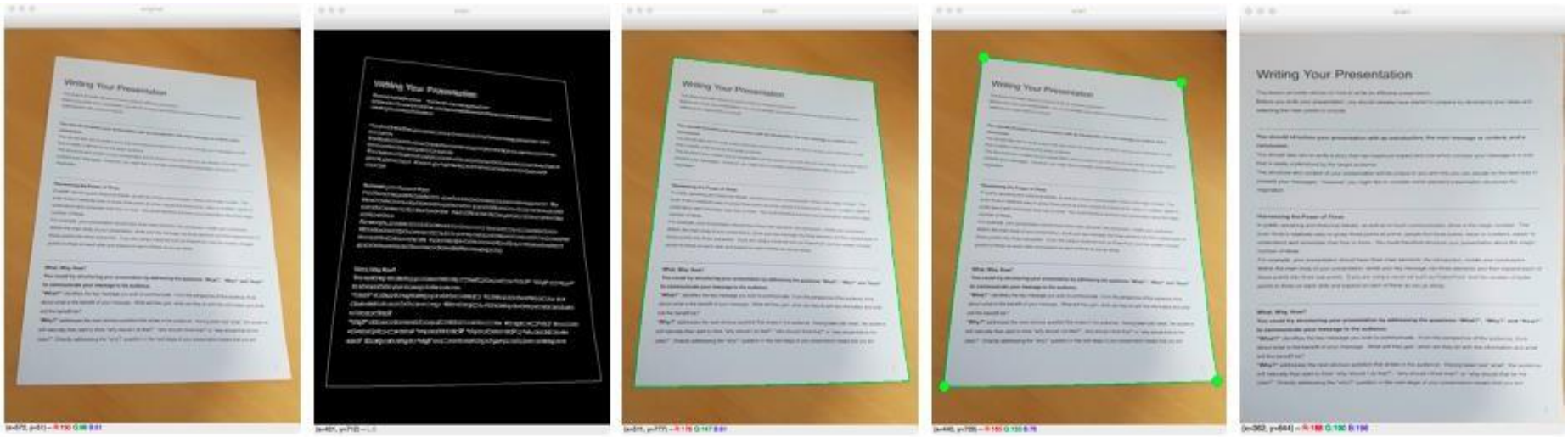


[그림 7-21] 도형 알아맞히기 사례

- 힌트
  - 컨투어를 찾아서 근사 값으로 단순화해서 꼭지점 갯수를 세어 봅니다.

❖ Document Scanner

- 마우스 입력 없이 자동으로 문서 스캔 효과내는 프로그램을 작성하세요.
- 결과 예시



[그림 7-22] 문서 스캐너 실행 예시

- 힌트
  - 4점 꼭지점 얻기
    - 케니 엣지로 경계 검출
    - findCotour()로 가장 큰 컨투어 골라서 approxPolyDP()로 단순화

Workshop

❖ Coin Counter

- 영상에서 동전을 분리하면서 갯수를 세는 프로그램을 만드세요
  .
- 결과 예시

[그림 7-23] 동전 개수 세기 사례

Workshop

❖ Coin Counter

- 힌트
  - pyMeanShiftFiltering() 동전 표면 뭉게기
  - 오츠 스레시 홀드
  - distanceTransform() 동전 중앙지점 표시
  - 로컬 최대 값 찾기
  - 지역 최대 값을 seed로 floodfill()
  - 거리 변환으로 확실한 동전 전경 영역 확보
  - 거리 변환으로 확실한 배경 영역 확보
  - 확실한 배경 – 확실한 전경 = 알 수 없는 영역
  - connectComponents() 레이블링
  - watershed()
  - findCountour(), boundingRect()으로 동전 영역 분리