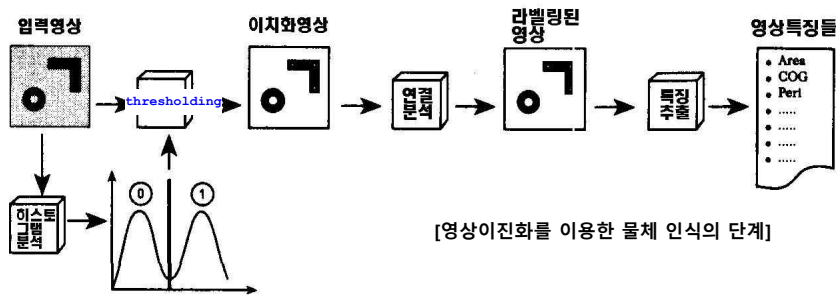
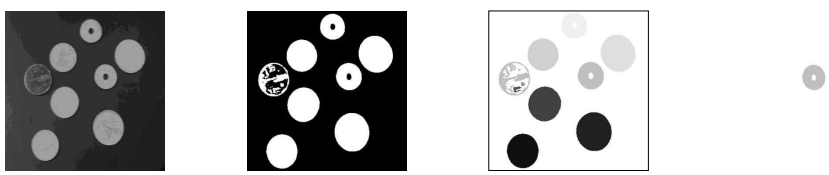
A-1. 윤곽선 검출

1

이진영상을 이용한 물체인식



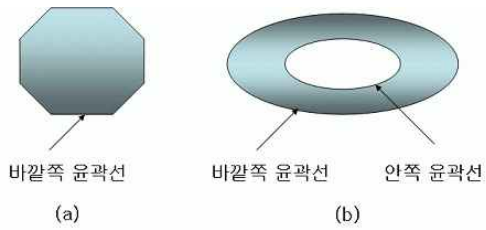
[영상이진화를 이용한 물체 인식의 단계]

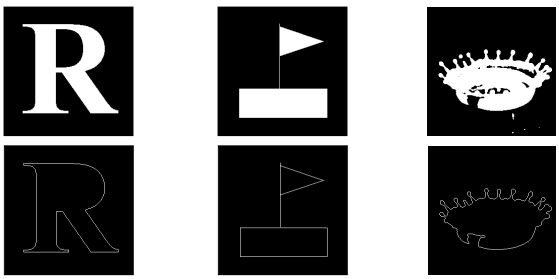


2

윤곽선 검출

- 바깥 윤곽선과 안쪽 윤곽선

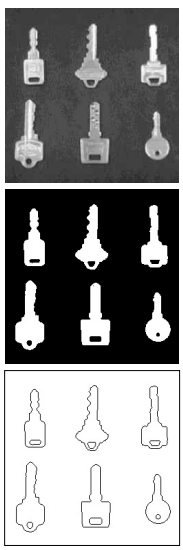




3

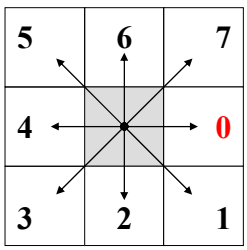
윤곽선 추적(Contour tracing)

- **영역의 윤곽선(경계) 추적**이란
  - 이진화된 영상에서 일정한 밝기값을 가지는 영역의 경계를 추적하여 경계픽셀의 순서화된 정보를 얻어내는 것



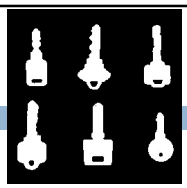
- 윤곽선 추적(Contour tracing)
  - 객체의 외곽선을 따라 이동하는 기법
  - 객체의 경계선 추적(boundary tracing)이라고도 함
- 8방향 연결성을 고려한 윤곽선 추적 진행 방향

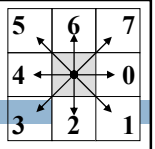
f[y-1][x-1]	f[y-1][x]	f[y-1][x+1]
f[y][x-1]	f[y][x]	f[y][x+1]
f[y+1][x-1]	f[y+1][x]	f[y+1][x+1]



4

윤곽선 추적





- 윤곽선 추적 알고리즘

1) 영상을 위에서 아래로, 왼쪽에서 오른쪽으로 스캔하면서 객체 픽셀을 찾는다.
   If 객체 픽셀을 찾으면, 이 좌표를 시작으로 *외곽선 추적을 시작*한다.
   초기 외곽선 추적 진행 방향은 d = 0 으로 설정한다.

2) 외곽선 추적 진행 방향에 객체 픽셀이 존재하는지를 판단한다.
3) If 진행 방향에 객체 픽셀이 존재하면, *해당 픽셀로 이동*한다.
   ① 외곽선 추적 방향을 d = d – 2 로 변화시키고 2)번으로 간다.
4) Else 진행 방향에 객체 픽셀이 존재하지 않으면, 외곽선 추적 방향을 d = d + 1 로 변화시키고, 2)번으로 간다.
   ① 모든 방향에 대하여 객체 픽셀이 존재하지 않으면, 1 픽셀짜리 객체이므로, *외곽선 추적을 종료*한다.

5) 외곽선 추적 중, 현재 픽셀 위치가 외곽선 추적 시작 좌표와 같고, 진행 방향이 d = 0 인 경우, *외곽선 추적을 종료*한다.

5

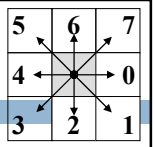
윤곽선 추적

- 윤곽선 추적 알고리즘 테스트
  - 작은 크기의 영상에 대하여 알고리즘을 검증
  - 테스트 영상 : 8x6 크기의 이진 영상

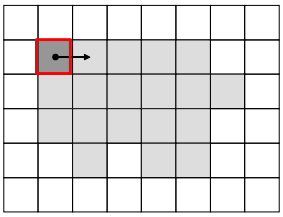
	1	1	1	1	1		
	1	1	1	1	1	1	
	1	1	1	1	1		
		1		1	1		

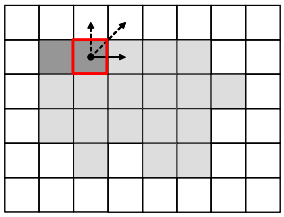
6

윤곽선 추적



- 예제 영상을 이용한 윤곽선 추적 알고리즘 테스트





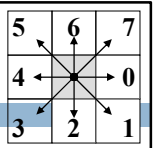
첫 객체 픽셀 -> *외곽선 추적 시작*
초기 추적 진행 방향은 d = 0

진행 방향에 객체 픽셀이 존재하면
해당 픽셀로 이동
추적 진행 방향은 d = d - 2 = -2 = 6

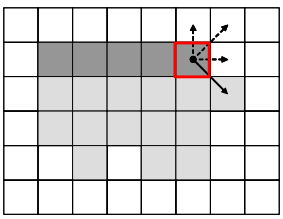
객체 픽셀이 존재하지 않으면
추적 방향을 d = d + 1 = 7 로 변화
객체 픽셀이 존재하지 않으면
추적 방향을 d = d + 1 = 8 = 0 로 변화

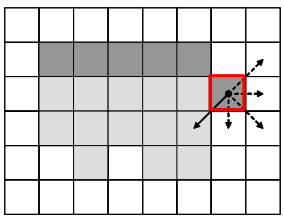
7

윤곽선 추적



- 예제 영상을 이용한 윤곽선 추적 알고리즘 테스트





진행 방향에 객체 픽셀이 존재하면
해당 픽셀로 이동
추적 진행 방향은 d = d - 2 = -2 = 6

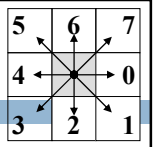
6,7,0 방향으로 객체 픽셀 존재 안하므로
d = 1로 변화

진행 방향에 객체 픽셀이 존재하면
해당 픽셀로 이동
추적 진행 방향은 d = d - 2 = -1 = 7

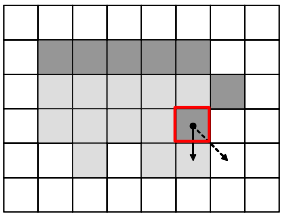
7,0,1,2 방향으로 객체 픽셀 존재 안하므로
d = 3로 변화

8

윤곽선 추적

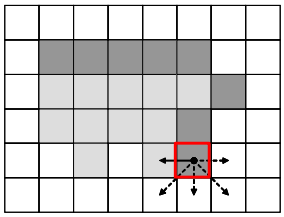


- 예제 영상을 이용한 윤곽선 추적 알고리즘 테스트



진행 방향에 객체 픽셀이 존재하면
해당 픽셀로 이동
추적 진행 방향은 d = d - 2 = 1

1 방향으로 객체 픽셀 존재 안하므로
d = 2로 변화

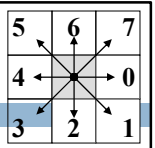


진행 방향에 객체 픽셀이 존재하면
해당 픽셀로 이동
추적 진행 방향은 d = d - 2 = 0

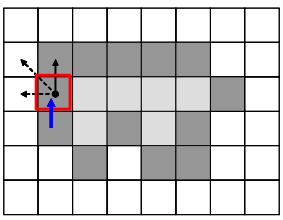
0,1,2,3 방향으로 객체 픽셀 존재 안하므로
d = 4로 변화

9

윤곽선 추적

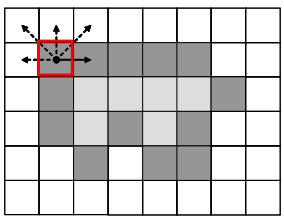


- 예제 영상을 이용한 윤곽선 추적 알고리즘 테스트



6번 방향으로 이동
추적 진행 방향은 d = d - 2 = 4

4,5 방향으로 객체 픽셀 존재 안하므로
d = 6로 변화

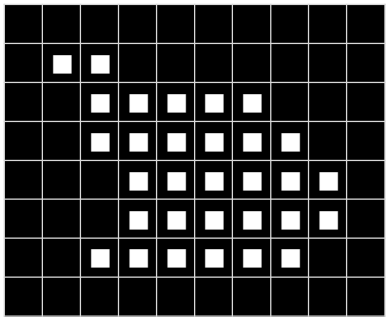


진행 방향에 객체 픽셀이 존재하면
해당 픽셀로 이동
추적 진행 방향은 d = d - 2 = 4

4,5,6,7 방향으로 객체 픽셀 존재 안하므로
d = 0로 변화

현재 픽셀 위치가 외곽선 추적 시작 좌표와 같고,
진행 방향이 d = 0 인 경우 추적 종료

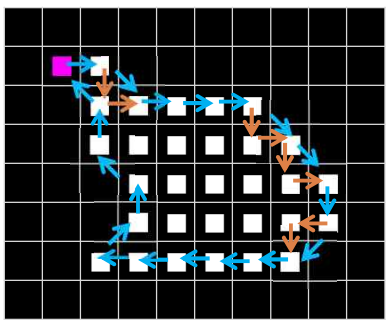
10

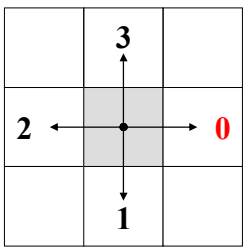


11

- **4방향 이웃**

1) If 진행 방향에 객체 픽셀이 존재하면, *해당 픽셀로 이동*한다.
   ① 외곽선 추적 방향을 d = d – 1 로 변화시키고 2)번으로 간다.





14

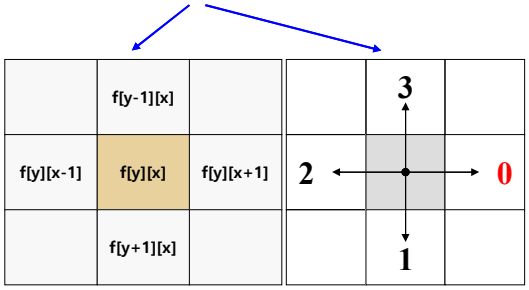
윤곽선 추적 in OpenCV

- **cv2.findContours(image, mode, method) ➔ contours, hierarchy**
  - **mode** : contours를 찾는 방법
    - cv2.RETR_EXTERNAL : contours line중 가장 바깥쪽 Line만 찾음.
    - cv2.RETR_LIST : 모든 contours line을 찾지만, hierarchy 관계를 구성하지 않음.
    - cv2.RETR_CCOMP : 모든 contours line을 찾으며, hierarchy 관계는 2-level로 구성함.
    - cv2.RETR_TREE : 모든 contours line을 찾으며, 모든 hierarchy 관계를 구성함.
  - **method** : contours를 찾을 때 사용하는 근사치 방법
    - cv2.CHAIN_APPROX_NONE : 모든 contours point를 저장.
    - cv2.CHAIN_APPROX_SIMPLE : contours line을 그릴 수 있는 point 만 저장. (ex; 사각형이면 4개 point)
    - cv2.CHAIN_APPROX_TC89_L1 : contours point를 찾는 algorithm
    - cv2.CHAIN_APPROX_TC89_KCOS : contours point를 찾는 algorithm

16

윤곽선 그리기 in OpenCV

- **cv2.drawContours(image, contours, contourIdx, color[, thickness[, lineType]]) ➔ image**
  - **contours** – contours 정보.
  - **contourIdx** – contours list type에서 몇 번째 contours line을 그릴 것인지. -1 이면 전체
  - **color** – contours line color
  - **thickness** – contours line의 두께. 음수이면 contours line의 내부를 채움
  - **lineType** – 선 연결성. 8은 8방향 연결, 4는 4방향 연결



17

예제 7.6 : 윤곽선 추적1

```
# Python으로 배우는 OpenCV 프로그래밍

src = np.zeros(shape=(512,512,3), dtype=np.uint8)
cv2.rectangle(src, (50, 100), (450, 400), (255, 255, 255), -1)
cv2.rectangle(src, (100, 150), (400, 350), (0, 0, 0), -1)
cv2.rectangle(src, (200, 200), (300, 300), (255, 255, 255), -1)
gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)

mode = cv2.RETR_EXTERNAL
method = cv2.CHAIN_APPROX_SIMPLE   ##method =cv2.CHAIN_APPROX_NONE
contours, hierarchy = cv2.findContours(gray, mode, method)
print('type(contours)=', type(contours))
print('type(contours[0])=', type(contours[0]))
print('len(contours)=', len(contours))
print('contours[0].shape=', contours[0].shape)
print('contours[0]=', contours[0])

cv2.drawContours(src, contours, -1, (255,0,0), 3) # 모든 윤곽선

for pt in contours[0][:]: # 윤곽선 좌표
    cv2.circle(src, (pt[0][0], pt[0][1]), 5, (0,0,255), -1)
```

18

예제 7.7 : 윤곽선 추적2

```
# Python으로 배우는 OpenCV 프로그래밍

src = np.zeros(shape=(512,512,3), dtype=np.uint8)
cv2.rectangle(src, (50, 100), (450, 400), (255, 255, 255), -1)
cv2.rectangle(src, (100, 150), (400, 350), (0, 0, 0), -1)
cv2.rectangle(src, (200, 200), (300, 300), (255, 255, 255), -1)
gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)

mode = cv2.RETR_LIST
method = cv2.CHAIN_APPROX_SIMPLE;
contours, hierarchy = cv2.findContours(gray, mode, method)
##cv2.drawContours(src, contours, -1, (255,0,0), 3) # 모든 윤곽선

print('len(contours)=', len(contours))
print('contours[0].shape=', contours[0].shape)
print('contours=', contours)

for cnt in contours:
    cv2.drawContours(src, [cnt], 0, (255,0,0), 3)

    for pt in cnt: # 윤곽선 좌표
        cv2.circle(src, (pt[0][0], pt[0][1]), 5, (0,0,255), -1)
```

19

A-2. 모양 특징 검출

20

윤곽선에 의한 모양 관련 특징 검출

- Contour의 둘레 길이
  - **cv2.arcLength(curve, closed) → retval**
    - closed가 true이면 폐곡선 도형을 만들어 둘레길이를, false이면 시작점과 끝점을 연결하지 않고 둘레 길이를 구함
- Contour 면적
  - **cv2.contourArea(contour[, oriented]) → retval**
- Contour 최소 면적 도형
  - 사각형
    - **cv2.boundingRect(points) → rect**
  - 회전 사각형
    - **cv2.minAreaRect(points) → box**
  - 삼각형
    - **cv2.minEnclosingTriangles(points[, triangle]) → retval, triangles**
  - 원
    - **cv2.minEnclosingCircle(points) → center, radius**

21

윤곽선에 의한 모양 관련 특징 검출

- 근사
  - 직선
    - **cv2.fitLine(points, disType, param, reps, aeps[, line]) ➔ line**
      - disType은 직선 근사를 위해 사용하는 거리 계산 방법. CV_DIST_L2일 때 최소자승에 의한 직선 근사(least square fit)로 가장 빠르게 계산
      - param은 distType에서 사용하는 상수. 0이면 최적의 값을 계산
      - reps와 aeps는 반지름과 각도의 충분한 정확도. 0.01을 사용
      - line은 (vx,vy, x0,y0). (vx,vy)는 직선의 방향 벡터, (x0,y0)는 직선 위의 한 점

$$(y - y_0) = \frac{vy}{vx}(x - x_0)$$

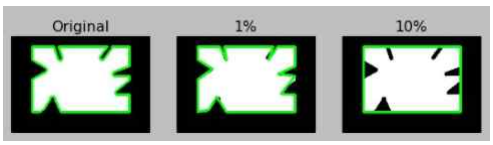
  - 타원
    - **cv2.fitEllipse(points) ➔ ellipse**

22

윤곽선에 의한 모양 관련 특징 검출

- 근사
  - 다각형
    - **cv2.approxPolyDP (curve, epsilon, closed[, approxCurve]) ➔ approxCurve**
      - epsilon은 다각형의 직선과의 허용 거리. 값이 크면 approxCurve에 저장되는 좌표점의 개수가 작아짐

```
# 적용하는 숫자가 커질 수록 Point의 갯수는 감소
epsilon1 = 0.01*cv2.arcLength(contours[i], True)
epsilon2 = 0.1*cv2.arcLength(contours[i], True)
approx1 = cv2.approxPolyDP(contours[i], epsilon1, True)
approx2 = cv2.approxPolyDP(contours[i], epsilon2, True)
```

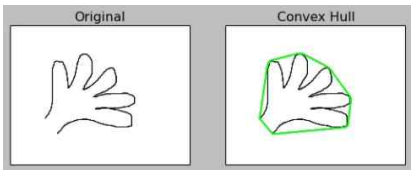


  - 다각형 내부 확인 : 한 점이 윤곽선 내부에 포함되는지 확인
    - **cv2.pointPolygonTest(contour, pt, measureDist) ➔ retval**
      - measureDist가 true면, pt에서 contour의 가장 가까운 다각형의 에지까지의 거리

23

윤곽선에 의한 모양 관련 특징 검출

- contours point를 모두 포함하는 볼록한 외곽선
  - **cv2.convexHull(points[, hull[, clockwise[, returnPoints]]])**
    ➔ hull



- Contour가 Convex인지 판단
  - **cv2.isContourConvex(contour) ➔ retval**
- Convex 결함을 계산
  - **cv2.convexityDetects(contour, hull[, detects]) ➔ detects**

볼록체가 되지 못한 부분, 즉 오목하게 들어간 부분

24

예제 8.12 : 길이

```
# Python으로 배우는 OpenCV 프로그래밍

src = cv2.imread('images/banana.jpg')

gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
ret, bImage = cv2.threshold(gray, 220, 255, cv2.THRESH_BINARY_INV)
bImage = cv2.dilate(bImage, None)  ##bImage = cv2.erode(bImage, None)
cv2.imshow('bImage',  bImage)      ##cv2.imshow('src',  src)

mode   = cv2.RETR_EXTERNAL
method = cv2.CHAIN_APPROX_SIMPLE
contours, hierarchy = cv2.findContours(bImage, mode, method)
print('len(contours)=', len(contours))

maxLength = 0
k = 0
for i, cnt in enumerate(contours):
    perimeter = cv2.arcLength(cnt, closed = True)
    if perimeter> maxLength:
        maxLength = perimeter
        k = i
print('maxLength=', maxLength)
cnt = contours[k]
dst2 = src.copy()
cv2.drawContours(dst2, [cnt], 0, (255,0,0), 3)
cv2.imshow('dst2',  dst2)
```

25

예제 8.12 : 면적, 최소면적 도형

```
area = cv2.contourArea(cnt)
print('area=', area)
x, y, width, height = cv2.boundingRect(cnt)
dst3 = dst2.copy()
cv2.rectangle(dst3, (x, y), (x+width, y+height), (0,0,255), 2)
cv2.imshow('dst3',  dst3)

rect = cv2.minAreaRect(cnt)
box = cv2.boxPoints(rect)
box = np.int32(box)
print('box=', box)
dst4 = dst2.copy()
cv2.drawContours(dst4,[box],0,(0,0,255),2)
cv2.imshow('dst4',  dst4)

(x,y),radius = cv2.minEnclosingCircle(cnt)
dst5 = dst2.copy()
cv2.circle(dst5,(int(x),int(y)),int(radius),(0,0,255),2)
cv2.imshow('dst5',  dst5)
```

26

예제 8.13 : 근사, 내부점 확인

```
rows,cols = dst6.shape[:2]
[vx,vy,x0,y0] = cv2.fitLine(cnt, cv2.DIST_L2, 0, 0.01, 0.01)
y1 = int((-x0*vy/vx) + y0)        # x = 0일 때, (0,y1)
y2 = int(((cols-1-x0)*vy/vx)+y0) # x = cols일 때, (cols-1,y2)
dst6 = dst2.copy()
cv2.line(dst6,(0,y1), (cols-1,y2),(0,0,255), 2)
cv2.imshow('dst6',  dst6)

ellipse = cv2.fitEllipse(cnt)
dst7 = dst2.copy()
cv2.ellipse(dst7, ellipse,(0,0,255),2)
cv2.imshow('dst7',  dst7)

poly = cv2.approxPolyDP(cnt, epsilon=20, closed=True)
dst8 = dst2.copy()
cv2.drawContours(dst8, [poly], 0, (0,0,255), 2)
cv2.imshow('dst8',  dst8)

dst9 = dst2.copy()
for y in range(rows):
    for x in range(cols):
        if cv2.pointPolygonTest(cnt, (x, y), False)>0:
            dst9[y, x] = (0, 255, 0)
cv2.imshow('dst9',  dst9)
```

$$(y - y_0) = \frac{vy}{vx}(x - x_0)$$

27

예제 8.14 : 볼록 다각형

```
# Python으로 배우는 OpenCV 프로그래밍

src = cv2.imread('images/hand.jpg')

hsv = cv2.cvtColor(src, cv2.COLOR_BGR2HSV)
lowerb = (0, 40, 0)
upperb = (20, 180, 255)
bImage = cv2.inRange(hsv, lowerb, upperb)

mode   = cv2.RETR_EXTERNAL
method = cv2.CHAIN_APPROX_SIMPLE
contours, hierarchy = cv2.findContours(bImage, mode, method)

dst = src.copy()
cnt = contours[0]
cv2.drawContours(dst, [cnt], 0, (255,0,0), 2)

hull = cv2.convexHull(cnt)
cv2.drawContours(dst, [hull], 0, (0,0,255), 2)
cv2.imshow('dst',  dst)
```

검사하여 하한값과 상한값 사이에
들어오면 흰색(255)로 표시하고
그렇지 않은 픽셀은 검은색(0)으로 표시

28

예제 8.15 : 볼록 결함

```
hull = cv2.convexHull(cnt, returnPoints=False)
dst2 = dst2.copy()

T = 5 # 10
defects = cv2.convexityDefects(cnt, hull)
print('defects.shape=', defects.shape)
for i in range(defects.shape[0]):
    s,e,f,d = defects[i,0]
    dist = d/256
    start = tuple(cnt[s][0])
    end   = tuple(cnt[e][0])
    far   = tuple(cnt[f][0])
    if dist > T:
        cv2.line(dst2, start, end, [255,255,0], 2)
        cv2.line(dst2, start, far, [0,255,0], 1)
        cv2.line(dst2, end,   far, [0,255,0], 1)

        cv2.circle(dst2, start, 5, [0,255,255], -1)
        cv2.circle(dst2, end,   5, [0,128,255], -1)
        cv2.circle(dst2, far,   5, [0,0,255], -1)
cv2.imshow('dst2',  dst2)
```

29