8.Contours

8.1Intro

- 需要输入二值化图像：先灰度图，再黑白（threshhold）
- cv2.findContours(); cv2.drawContours()

```
findContours(image, mode, method[, contours[, hierarchy[, offset]]]) ->
contours, hierarchy
```

```
drawContours(image, contours, contourIdx, color[, thickness[, lineType[,
hierarchy[, maxLevel[, offset]]]]]) -> image
#其中，contourIdx = -1: all
```

```
import numpy as np
import cv2
im = cv2.imread('data/star.jpg')
imgray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(src=imgray, thresh=127, maxval=255,
type=cv2.THRESH_BINARY)#src, thresh, maxval, type

contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
img = cv2.drawContours(im, contours, -1, (0, 255, 0), 2)

cv2.imshow("thresh", thresh)
cv2.imshow("imgray", imgray)
cv2.imshow("contours", img)
cv2.waitKey(0)
cv2.destroyAllWindows
```

8.2轮廓特征

8.2.1Moment

[wiki]

- 具有不变性：平移、缩放、旋转不变性； 和唯一性：M(p,q)由f(x,y)唯一确定
- 公式(连续)

$$M_{pd} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dxdy$$

二值图像面积：M_{00}

几何中心：

$$\{\bar{x}\,\bar{y}\} = \{\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\}$$

```
import cv2
import numpy as np
from pprint import pprint #pprint打印dictionary更好看一点，与print功能相同

img = cv2.imread('data/star.jpg')
gray_img=cv2.cvtColor(img,cv2.COLOR_BGRA2GRAY)
ret, thresh = cv2.threshold(gray_img, 127, 255, 0)
contours,hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

cnt = contours[0] #first contour
M = cv2.moments(cnt)#返回dictionary

pprint(M)
```

8.2.2面积

- cv2.contourArea(), or M['m00']

```
area=cv2.contourArea(cnt)
```

8.2.3周长

- cv2.arcLength()

```
perimeter=cv2.arcLength(cnt,True) #True fro closed contour
```

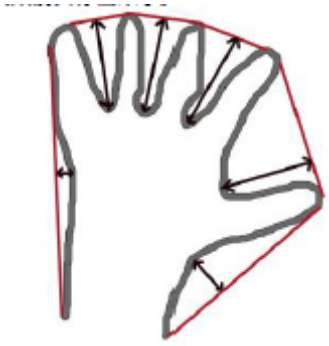
8.2.4轮廓近似

- Douglas-Peucker algorithm：是将曲线近似表示为一系列点，并减少点的数量的一种算法。
- cv2.approxPolyDP(): output approx contour rather than exact one

```
epsilon = 0.1*perimeter
approx = cv2.approxPolyDP(cnt,epsilon,True)
image=cv2.drawContours(img,[approx],-1,(255,0,0),2)

cv2.imshow('approxPolyDP',image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

8.2.5Convex hull



- to get the convex

```
hull = cv2.convexHull(cnt)
```

- convexity_defects

```
cv2.isContourConvex(cnt)-->True/False
or
cv2.defects = cv2.convexityDefects(cnt, hull)-->4-element integer vector,
(start_index, end_index, farthest_pt_index, fixpt_depth)
```

8.2.6边界矩形

- 直边界矩形-cv2.boundingRect(cnt) --> (w,y,w,h), (x,y)左上角坐标,(w,h)宽和高
  - 面积不是最小的

```
w,y,w,h = cv2.boudingRect()
img_1 = cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),255)
```

- 旋转的边界矩形-cv2.minAreaRect() -->(x,y),(w,h),(ϕ)-->Box2D
  - 面积最小矩阵

```
rect = cv2.minAreaRect(cnt)
box = cv2.boxPoints(rect)
box = np.int0(box)
img_2 = cv2.drawContours(img,[box],0,(0,255,0)
```

8.2.7最小外接圆

- cv2.minEnclosingCircle()-->(x,y),radius

```
(x,y),r = cv2.minEnclosingCircle(cnt)
img = cv2.circle(img,(int(x),int(y)),int(r),(255.0,0),2)
```

8.2.8椭圆拟合

- cv2.fitEllipse()-->旋转""边界矩形""的内切圆

```
ellipse = cv2.fitEllipse(cnt)
im = cv2.ellipse(im,ellipse,(0,255,0),2)
```

8.2.9直线拟合

- cv2.fitLine()

```
rows, cols = img.shape[:2]
[vx, vy, x, y] = cv2.fitLine(cnt, cv2.DIST_L2, 0, 0.01, 0.01)
lefty = int((-x * vy / vx) + y)
righty = int(((cols - x) * vy / vx) + y)
cv2.line(img, (cols - 1, righty), (0, lefty), (0, 255, 0), 2)
```

8.3轮廓的性质（8.2 extended）

8.3.1宽高比

$$wh\_ratio = \frac{Width}{Height}$$

```
w,y,w,h = cv2.boudingRect()
aspect_ratio = float(w/h)
```

8.3.2 extent

$$extent = \frac{object\_area}{bounding\_rec\_area}$$

```
area = cv2.contourArea(cnt)
w,y,w,h = cv2.boudingRect()
extent = float(area/(w*y))
```

8.3.3 Solidity

$$solidity = \frac{cnt\_area}{hull\_area}$$

```
area = cv2.contourArea(cnt)
hull = cv2.convexHull(cnt)
hull_area = cv2.contourArea(hull)
solidity = float(area/hull_area)
```

8.3.4 Equivalent diameter

$$ed = \sqrt{\frac{4 * cnt\_area}{\pi}}$$

```
area = cv2.contourArea(cnt)
ed = np.sqrt(4*area/np.pi)
```

8.3.5 方向、长短轴

```
(x,y),(l,s),angle = cv2.fitEllipse(cnt)
```

8.3.6 Mask -- 像素点、最值和平均值

- mask & all pixel points

```
import cv2
import numpy as np

img = cv2.imread('data/star.jpg',0)
ret, thresh = cv2.threshold(img, 127, 255, 0)
contours,hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
cnt = contours[0] #first contour
mask = np.zeros(img.shape,np.uint8)
mask = cv2.drawContours(mask,[cnt],0,255,-1) #must be '-1' -绘制填充的轮廓

pixel_points = np.transpose(np.nonzero(mask))#np计算出(row,col)-->(y,x)
print(pixel_points)
```

- max&min&their position

```
min_val,max_val,min_loc,mac_loc = cv2.minMaxLoc(imgrey,mask=mask)
```

- 平均颜色|平均灰度

```
mean_val = cv2.mean(imgrey|img, mask=mask)
```

8.3.7极点

```
left = tuple(cnt[cnt[:,:,0].argmin()][0])
right = tuple(cnt[cnt[:,:,0].argmax()][0])
top = tuple(cnt[cnt[:,:,1].argmin()][0])
bottom = tuple(cnt[cnt[:,:,1].argmax()][0])
```

8.4 Other functions

8.4.1 凸缺陷

cv2.convexHull(points[, hull[, clockwise[, returnPoints]]]) -> hull

hull = (起点，终点，最远的点，到最远点的近似距离)

```
hull = cv2.convexHull(cnt, returnPoints=False)
defects = cv2.convexityDefects(cnt, hull)
```

ps: returnPoints的值一定是False

8.4.2 Point ploygon test

点到contour的最短距离（在轮廓外，为负值）

```
dist = cv2.pointPolygonTest(cnt,(30,50), True)
```

8.4.3 形状匹配

匹配轮廓形状，返回值越小，匹配越好（根据hu矩计算）

```
ret = cv2.matchShapes(cnt1,cnt2,1,0,0)
```

8.5 hierarchy

contours, hierarchy = cv2.findContours(thresh, **cv2.RETR_TREE**, cv2.CHAIN_APPROX_NONE)

2nd args - Flags：

- cv2.RETR_LIST: 提取所有contour，不建立父子关系
- cv2.RETR_TREE: 所有信息
- cv2.RETR_CCOMP：所有轮廓分为两级结构，外部为1，内部为2
- cv2.RETR_EXTERNAL: 只返回最外面contour

PS：return value - (Next(same level), previous, First_child, parent)

if no parent, parent arg = -1