OpenCV 中的轮廓特征

轮廓特征

学习找到轮廓的不同特征，如区域，周长，边界矩形等。

目标

在本文中，我们将学习

- 查找轮廓的不同特征，例如面积，周长，质心，边界框等
- 您将看到大量与轮廓有关的功能。

矩

图像矩可帮助您计算某些特征，例如物体的重心，物体的面积等。请查看"图像矩"上的Wikipedia页面

函数 **cv.moments()** 提供了所有计算出的矩值的列表。见下文：

```
import numpy as np
import cv2 as cv
img = cv.imread('star.jpg',0)
ret,thresh = cv.threshold(img,127,255,0)
contours,hierarchy = cv.findContours(thresh, 1, 2)
cnt = contours[0]
M = cv.moments(cnt)
print( M )
```

在图像矩中，您可以提取有用的数据，例如面积，质心等。质心由关系C给出 C*x* = *frac{M*{10}}{M{*00}}* 和 C_*y* = *frac{M*{01}}{M_{00}}。可以按照以下步骤进行：

```
cx = int(M['m10']/M['m00'])
cy = int(M['m01']/M['m00'])
```

轮廓面积

轮廓区域由函数 **cv.contourArea()** 或从力矩 M_{00}中给出。

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

轮廓周长

也称为弧长。可以使用 **cv.arcLength()** 函数找到它。第二个参数指定形状是闭合轮廓(如果通过True)还是曲线。

```
perimeter = cv.arcLength(cnt,True)
```

轮廓近似

根据我们指定的精度，它可以将轮廓形状近似为顶点数量较少的其他形状。它是Douglas-Peucker算法的实现。检查维基百科页面上的算法和演示。

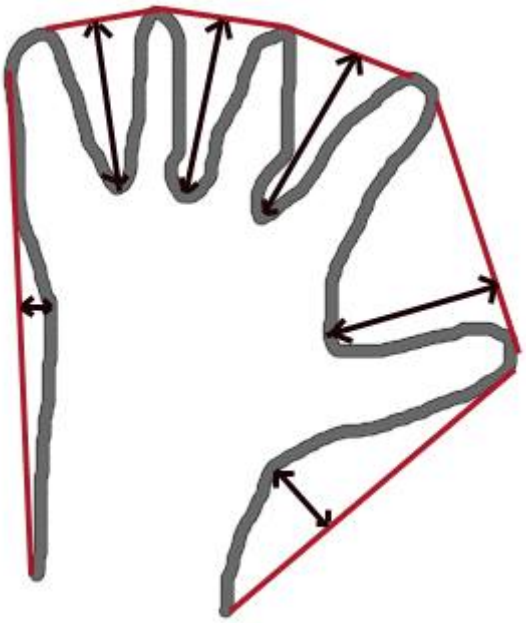
为了理解这一点，假设您试图在图像中找到一个正方形，但是由于图像中的某些问题，您没有得到一个完美的正方形，而是一个“坏形状”(如下图所示)。现在，您可以使用此功能来近似形状。在这种情况下，第二个参数称为epsilon，它是从轮廓到近似轮廓的最大距离。它是一个精度参数。需要正确选择epsilon 才能获得正确的输出。

```
epsilon = 0.1*cv.arcLength(cnt,True)
approx = cv.approxPolyDP(cnt,epsilon,True)
```

下面，在第二张图片中，绿线显示了 精度 epsilon = 10% 时的近似曲线。第三幅图显示了精度 epsilon = 1% 时的情况。第三个参数指定曲线是否闭合。

凸包

凸包外观看起来与轮廓逼近相似，但并非如此(在某些情况下两者可能提供相同的结果)。在这里，**cv.convexHull()** 函数检查曲线是否存在凸凹缺陷并对其进行校正。一般而言，凸曲线是始终凸出或至少平坦的曲线。如果在内部凸出，则称为凸度缺陷。例如，检查下面的手的图像。红线显示手的凸包。双向箭头标记显示凸度缺陷，这是船体与轮廓线之间的局部最大偏差。



关于它的语法，有一些事情需要讨论：

```
hull = cv.convexHull(points[, hull[, clockwise[, returnPoints]]])
```

参数详细信息：

- points: 就是我们传入的轮廓。
- hull: 是输出，通常我们避免它。
- clockwise：方向标记。如果为True，则输出凸包为顺时针方向。否则，其方向为逆时针方向。
- returnPoints：默认情况下为True。然后返回船体点的坐标。如果为False，则返回与船体点相对应的轮廓点的索引。

因此，要获得如上图所示的凸包，以下内容就足够了：

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

检查凸度

cv.isContourConvex() 是一个函数用来检查曲线是否为凸多边形。它只是返回True还是False。

```
k = cv.isContourConvex(cnt)
```

边界矩形

有两种类型的边界矩形。

直角矩形

它是一个直角矩形，不考虑对象的旋转。因此，边界矩形的面积将不会最小。它可以通过函数 cv.boundingRect() 找到。

令(x，y)为矩形的左上角坐标，而(w，h)为矩形的宽度和高度。

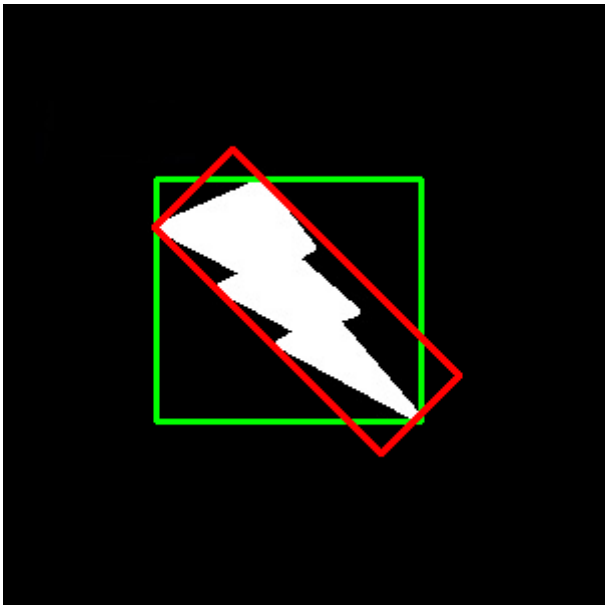
```
x,y,w,h = cv.boundingRect(cnt)
cv.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)
```

旋转矩形

在这里，边界矩形是用最小面积绘制的，因此它也考虑了旋转。使用的函数是 cv.minAreaRect() 。它返回一个Box2D结构，其中包含以下细节-(中心(x，y)，(宽度，高度)，旋转角度)。但是要绘制此矩形，我们需要矩形的4个角。它是通过函数 cv.boxPoints() 获得的

```
rect = cv.minAreaRect(cnt)
box = cv.boxPoints(rect)
box = np.int0(box)
cv.drawContours(img,[box],0,(0,0,255),2)
```

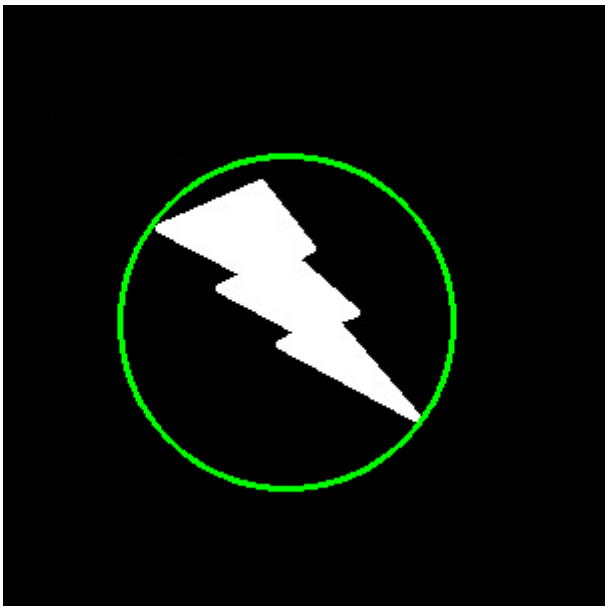
两个矩形都显示在单个图像中。绿色矩形显示法线边界矩形。红色矩形是旋转的矩形。



最小外圆

接下来，我们使用函数 cv.minEnclosingCircle() 找到对象的外接圆。它是一个以最小面积完全覆盖对象的圆圈。

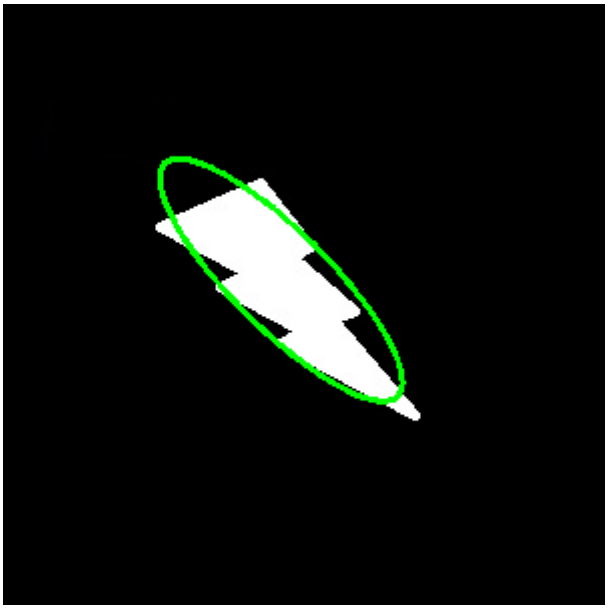
```
(x,y),radius = cv.minEnclosingCircle(cnt)
center = (int(x),int(y))
radius = int(radius)
cv.circle(img,center,radius,(0,255,0),2)
```



拟合椭圆

下一步是使椭圆适合对象。它返回椭圆所在的旋转矩形。

```
ellipse = cv.fitEllipse(cnt)
cv.ellipse(img,ellipse,(0,255,0),2)
```



拟合线

同样，我们可以将一条直线拟合到一组点。下图包含一组白点。我们可以近似一条直线。

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