**Contour Features**

เป้าหมาย

ในบทความนี้เราจะเรียนรู้

* เพื่อหาคุณสมบัติที่แตกต่างกันของรูปทรงเช่นปริมณฑล, centroid, bounding box ฯลฯ
* คุณจะเห็นมากมายของฟังก์ชันที่เกี่ยวข้องกับรูปทรง

1. Moments

ช่วงเวลาของภาพช่วยให้คุณสามารถคำนวณคุณสมบัติบางอย่างเช่นศูนย์กลางของมวลของวัตถุพื้นที่ของวัตถุ ฯลฯ ดูหน้าวิกิพีเดียใน[Image Moments](http://en.wikipedia.org/wiki/Image_moment)

cv2.momentsฟังก์ชัน()ให้พจนานุกรมของค่าขณะทั้งหมดที่คำนวณ ดูด้านล่าง:

**import** **cv2**

**import** **numpy** **as** **np**

img = cv2.imread('star.jpg',0)

ret,thresh = cv2.threshold(img,127,255,0)

contours,hierarchy = cv2.findContours(thresh, 1, 2)

cnt = contours[0]

M = cv2.moments(cnt)

**print** M

จากช่วงเวลานี้คุณจะสามารถดึงข้อมูลที่มีประโยชน์เช่นพื้นที่เซนทรอยด์ ฯลฯ เซนทรอยด์จะได้รับจากความสัมพันธ์, และC_x = \ frac {M_ {10}} {M_ {00}} C_y = \ frac {M_ {01}} {M_ {00}}ซึ่งสามารถทำได้ดังนี้

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

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

2. Contour Area

พื้นที่ Contour จะได้รับจากการทำงาน**cv2.contourArea ()**หรือจากช่วงเวลา**M [**'M00']

area = cv2.contourArea(cnt)

3. Contour Perimeter

เรียกอีกอย่างว่าความยาวของส่วนโค้ง มันสามารถพบได้ออกมาใช้**cv2.arcLength ()**ฟังก์ชั่น อาร์กิวเมนต์ที่สองระบุว่ารูปร่างเป็นรูปร่างที่ปิดหรือไม่ (ถ้าผ่านจริง ) หรือเป็นเส้นโค้ง

perimeter = cv2.arcLength(cnt,True)

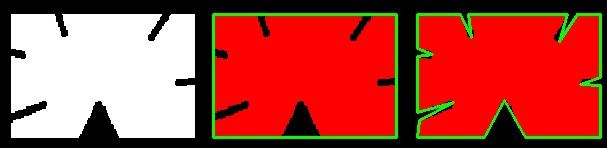
4. Contour Approximation

ประมาณรูปรูปร่างเป็นรูปร่างอื่นที่มีจำนวนจุดภาพน้อยขึ้นอยู่กับความแม่นยำที่เราระบุ มันคือการดำเนินการขั้นตอนวิธีการดักลาส[Peucker](http://en.wikipedia.org/wiki/Ramer-Douglas-Peucker_algorithm)ตรวจสอบหน้าวิกิพีเดียสำหรับอัลกอริทึมและการสาธิต

ในการทำความเข้าใจเรื่องนี้สมมติว่าคุณกำลังพยายามหารูปสี่เหลี่ยมจัตุรัสในภาพ แต่เนื่องจากปัญหาในภาพคุณไม่ได้รับสแควร์ที่สมบูรณ์แบบ แต่เป็น "รูปร่างที่ไม่ดี" (ดังรูปที่ 1) ตอนนี้คุณสามารถใช้ฟังก์ชันนี้เพื่อประมาณรูปร่างได้ ในข้อนี้อาร์กิวเมนต์ที่สองเรียกว่าepsilonซึ่งเป็นระยะห่างสูงสุดจากรูปทรงเป็นเส้นประมาณ เป็นพารามิเตอร์ที่ถูกต้อง ควรเลือกepsilon ที่ชาญฉลาดเพื่อให้ได้ผลลัพธ์ที่ถูกต้อง

epsilon = 0.1\*cv2.arcLength(cnt,True)

approx = cv2.approxPolyDP(cnt,epsilon,True)

ด้านล่างนี้ในภาพสองเส้นสีเขียวแสดงให้เห็นเส้นโค้งโดยประมาณสำหรับepsilon = 10% ของ ส่วนโค้ง ยาว ภาพที่สามแสดงให้เห็นเหมือนกันสำหรับepsilon = 1% ของ โค้ง ยาว อาร์กิวเมนต์ที่สามระบุว่ามีการปิดเส้นโค้งหรือไม่

5. Convex Hull

Convex Hull จะมีลักษณะคล้ายกับการประมาณรูปเส้น แต่ไม่ใช่ (ทั้งสองอย่างอาจให้ผลเหมือนกันในบางกรณี) นี่**cv2.convexHull ()**ฟังก์ชั่นตรวจสอบโค้งนูนสำหรับข้อบกพร่องและแก้ไขมัน โดยทั่วไปแล้วเส้นโค้งที่โค้งงอเป็นเส้นโค้งที่พองออกเสมอหรือแบนอย่างน้อย และถ้ามันถูก bulged ภายในเรียกว่าข้อบกพร่องนูน ตัวอย่างเช่นตรวจสอบภาพด้านล่างของมือ เส้นสีแดงแสดงให้เห็นถึงลำตัวนูนของมือ เครื่องหมายลูกศรสองด้านแสดงข้อบกพร่องของนูนซึ่งเป็นส่วนเบี่ยงเบนสูงสุดของพื้นที่ในเรือจากรูปทรง

มีเรื่องเล็ก ๆ น้อย ๆ เพื่อหารือเรื่องไวยากรณ์ของมัน:

hull = cv2.convexHull(points[, hull[, clockwise[, returnPoints]]

รายละเอียดอาร์กิวเมนต์:

* จุดคือรูปทรงที่เราผ่านเข้า
* เรือเป็นผลผลิตปกติเราหลีกเลี่ยง
* ตามเข็มนาฬิกา : ธงปฐมนิเทศ ถ้าเป็นTrueผลลัพธ์ของลำตัวนูนจะถูกปรับตามเข็มนาฬิกา มิฉะนั้นจะเป็นไปตามทิศทางทวนเข็มนาฬิกา
* returnPoints : โดยค่าเริ่มต้นทรู จากนั้นจะส่งกลับค่าพิกัดของจุดเรือ ถ้าเป็นเท็จมันจะส่งกลับค่าดัชนีของจุดรูปร่างที่สอดคล้องกับจุดฮัลล์

ดังนั้นเพื่อให้ได้ลำตัวนูนในรูปด้านบนให้ทำตามขั้นตอนต่อไปนี้:

hull = cv2.convexHull(cnt)

แต่ถ้าคุณต้องการที่จะหาข้อบกพร่องนูน, คุณต้องผ่านreturnPoints = เท็จ เพื่อให้เข้าใจเราจะใช้รูปสี่เหลี่ยมผืนผ้าเหนือ ครั้งแรกที่ผมพบว่ารูปร่างของมันเป็นCNT ตอนนี้ฉันพบลำตัวนูนของมันด้วยreturnPoints = Trueฉันได้ค่าต่อไปนี้: [[234 202], [[ 51 202]], [[51 79]], [[234 79]]ซึ่งเป็นมุมทั้งสี่ จุดสี่เหลี่ยมผืนผ้า ตอนนี้ถ้าทำแบบเดียวกันกับreturnPoints = Falseฉันจะได้รับผลดังนี้: [[129], [ 67], [ 0], [142]. นี่คือดัชนีของจุดที่สอดคล้องกันในรูปทรง สำหรับตัวอย่างเช่นตรวจสอบค่าแรก: cnt [129] = [[234,202]]ซึ่งเหมือนกับผลลัพธ์แรก (และอื่น ๆ )

คุณจะเห็นอีกครั้งเมื่อเราพูดถึงข้อบกพร่องเกี่ยวกับนูน

6. Checking Convexity

มีฟังก์ชั่นการตรวจสอบว่าเป็นเส้นโค้งนูนหรือไม่เป็น**cv2.isContourConvex**() มันก็กลับว่าจริงหรือเท็จ ไม่ใช่เรื่องใหญ่.

k = cv2.isContourConvex(cnt)

7. Bounding Rectangle

มีสี่เหลี่ยมผืนผ้าล้อมรอบสองประเภท

7.a. Straight Bounding Rectangle

เป็นรูปสี่เหลี่ยมผืนผ้าตรงไม่พิจารณาการหมุนของวัตถุ ดังนั้นพื้นที่ของสี่เหลี่ยมผืนผ้า bounding จะไม่ต่ำสุด มันถูกพบโดยฟังก์ชัน**cv2.boundingRect**()

ให้ (x, y) เป็นพิกัดซ้ายบนของสี่เหลี่ยมผืนผ้าและ (w, h) คือความกว้างและความสูง

x,y,w,h = cv2.boundingRect(cnt)

cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)

7.b. Rotated Rectangle

ที่นี่ขอบเขตรูปสี่เหลี่ยมผืนผ้าจะวาดด้วยพื้นที่ขั้นต่ำดังนั้นจึงพิจารณาการหมุนด้วย ฟังก์ชั่นที่ใช้เป็น**cv2.minAreaRect**() มันส่งกลับโครงสร้าง Box2D ซึ่งประกอบด้วย detals ต่อไปนี้ - (ศูนย์ (x, y), (ความกว้าง, ความสูง), มุมของการหมุน) แต่การวาดสี่เหลี่ยมผืนผ้านี้เราต้องใช้สี่มุมของสี่เหลี่ยมผืนผ้า ได้จากฟังก์ชัน**cv2.boxPoints ()**

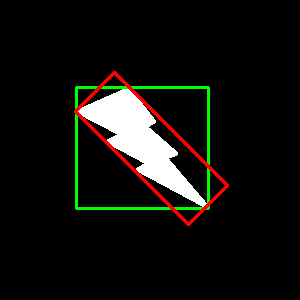
rect = cv2.minAreaRect(cnt)

box = cv2.boxPoints(rect)

box = np.int0(box)

cv2.drawContours(img,[box],0,(0,0,255),2)

ทั้งสี่เหลี่ยมจะแสดงในภาพเดียว สี่เหลี่ยมผืนผ้าสีเขียวแสดงโหนด bounding ตามปกติ สี่เหลี่ยมผืนผ้าสีแดงคือรูปสี่เหลี่ยมผืนผ้าที่หมุน



8. Minimum Enclosing Circle

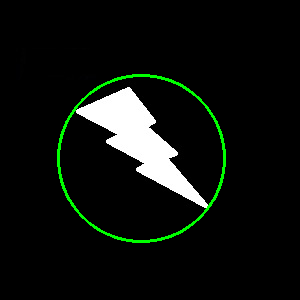
ถัดไปเราจะพบ circumcircle ของวัตถุโดยใช้ฟังก์ชั่นนั้น**cv2.minEnclosingCircle**() เป็นวงกลมที่ครอบคลุมวัตถุทั้งหมดโดยมีพื้นที่ต่ำสุด

(x,y),radius = cv2.minEnclosingCircle(cnt)

center = (int(x),int(y))

radius = int(radius)

cv2.circle(img,center,radius,(0,255,0),2)

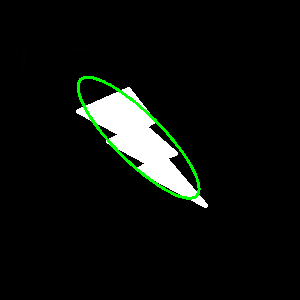


9. Fitting an Ellipse

หนึ่งถัดไปคือให้พอดีกับวงรีกับวัตถุ จะส่งกลับรูปสี่เหลี่ยมผืนผ้าที่หมุนเวียนซึ่งมีการเขียนวงรีไว้

ellipse = cv2.fitEllipse(cnt)

cv2.ellipse(img,ellipse,(0,255,0),2)



10. Fitting a Line

ในทำนองเดียวกันเราสามารถพอดีกับชุดของจุด ภาพด้านล่างมีชุดของจุดสีขาว เราสามารถประมาณเส้นตรงไปได้

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)

