1. Defectos de convexidad Vimos qué es el casco convexo en el segundo capítulo sobre contornos. Cualquier desviación del objeto de esta carcasa puede considerarse un defecto de convexidad.

OpenCV viene con una función lista para encontrar esto, cv.convexityDefects() . Una llamada de función básica se vería como a continuación:

import cv2 as cv  
hull = cv.convexHull(cnt,returnPoints = False)  
defects = cv.convexityDefects(cnt,hull)

import cv2 as cv  
import numpy as np  
img = cv.imread('resources/star.png')  
assert img is not None, "file could not be read, check with os.path.exists()"  
img\_gray = cv.cvtColor(img,cv.COLOR\_BGR2GRAY)  
ret,thresh = cv.threshold(img\_gray, 127, 255,0)  
contours,hierarchy = cv.findContours(thresh,2,1)  
cnt = contours[0]  
hull = cv.convexHull(cnt,returnPoints = False)  
defects = cv.convexityDefects(cnt,hull)  
for i in range(defects.shape[0]):  
 s,e,f,d = defects[i,0]  
 start = tuple(cnt[s][0])  
 end = tuple(cnt[e][0])  
 far = tuple(cnt[f][0])  
 cv.line(img,start,end,[0,255,0],2)  
 cv.circle(img,far,5,[0,0,255],-1)  
cv.imshow('img',img)  
cv.waitKey(0)  
cv.destroyAllWindows()

1. Prueba de polígono puntual Esta función encuentra la distancia más corta entre un punto de la imagen y un contorno. Devuelve la distancia que es negativa cuando el punto está fuera del contorno, positiva cuando el punto está dentro y cero si el punto está en el contorno.

Por ejemplo, podemos comprobar el punto (50,50) de la siguiente manera:

dist = cv.pointPolygonTest(cnt,(50,50),True)

1. Combina formas OpenCV viene con una función cv.matchShapes() que nos permite comparar dos formas o dos contornos y devuelve una métrica que muestra la similitud. Cuanto menor sea el resultado, mejor será la coincidencia. Se calcula en base a los valores del momento hu. En los documentos se explican diferentes métodos de medición.

import cv2 as cv  
import numpy as np  
img1 = cv.imread('resources/star.jpg', cv.IMREAD\_GRAYSCALE)  
img2 = cv.imread('resources/star2.jpg', cv.IMREAD\_GRAYSCALE)  
assert img1 is not None, "file could not be read, check with os.path.exists()"  
assert img2 is not None, "file could not be read, check with os.path.exists()"  
ret, thresh = cv.threshold(img1, 127, 255,0)  
ret, thresh2 = cv.threshold(img2, 127, 255,0)  
contours,hierarchy = cv.findContours(thresh,2,1)  
cnt1 = contours[0]  
contours,hierarchy = cv.findContours(thresh2,2,1)  
cnt2 = contours[0]  
ret = cv.matchShapes(cnt1,cnt2,1,0.0)  
print( ret )