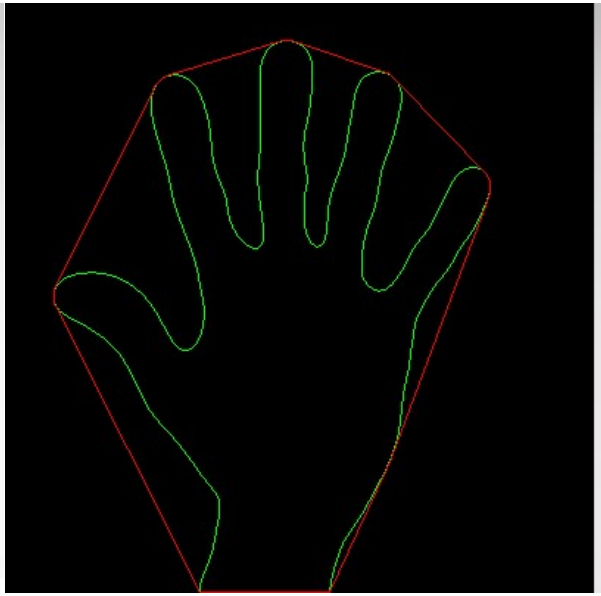
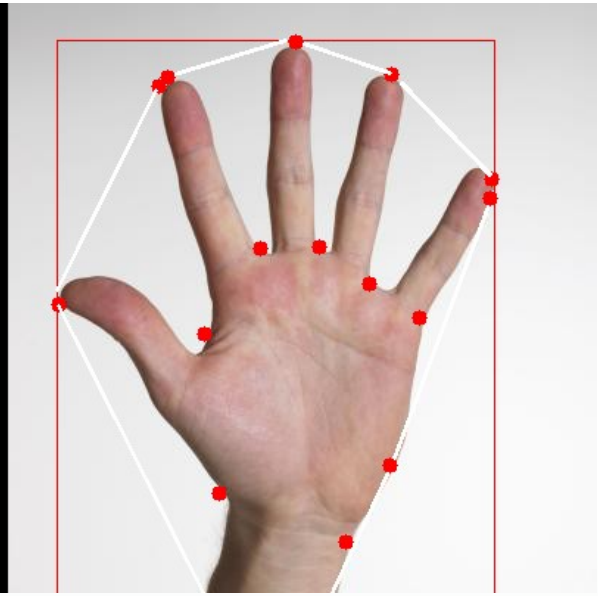
Contagem de Dedos

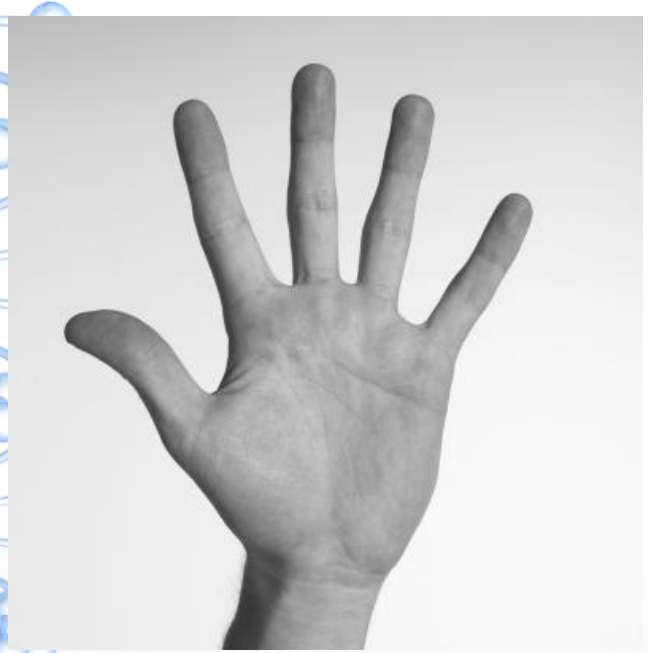




Pré-processamento

- Faça a importação do matplotlib, numpy e opencv
- Leia a imagem e converte-a em escala de cinza
-
- Seria possível, você aplicar a binarização (Otsu) direta:
- _, thresh1 = cv2.threshold(grey, 0, 255, cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
- cv2.imwrite('thresh1.png', thresh1)

Pré-processamento



- Só que ficam essas falhas na binarização da mão

Pré-processamento

- Aplica-se um filtro gaussiano para fazer a suavização da imagem
- value = (35, 35)
- blurred = cv2.GaussianBlur(grey, value, 0)
-

Pré-processamento

- Aí então, faz-se a binarização

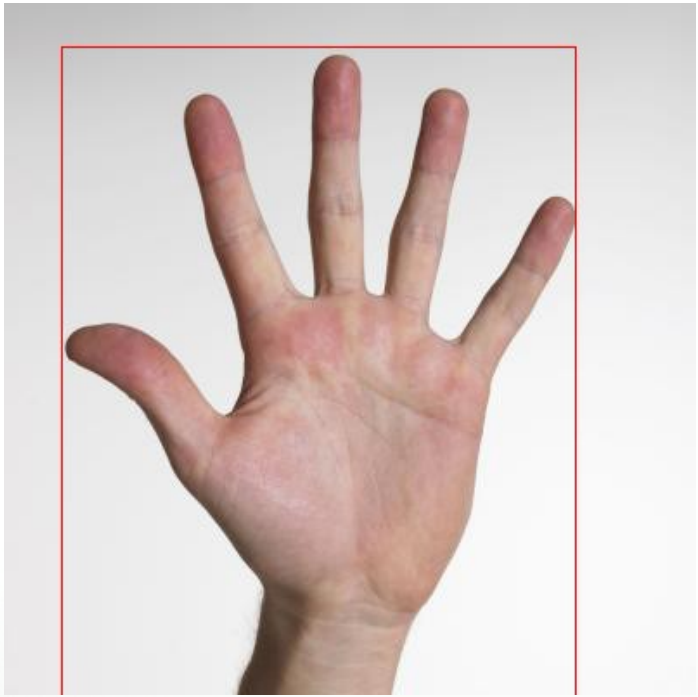
Contorno

- (version, _, _) = cv2.__version__.split('.')
- if version == '3':
- image, contours, hierarchy = cv2.findContours(thresh1.copy(), \
- cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
- elif version == '2':
- contours, hierarchy = cv2.findContours(thresh1.copy(),cv2.RETR_TREE, \
- cv2.CHAIN_APPROX_NONE)
-

Contorno

- # encontra o maior contorno
- cnt = max(contours, key = lambda x: cv2.contourArea(x))
- cv2.drawContours(res, [cnt], 0, (0, 255, 0), 0)

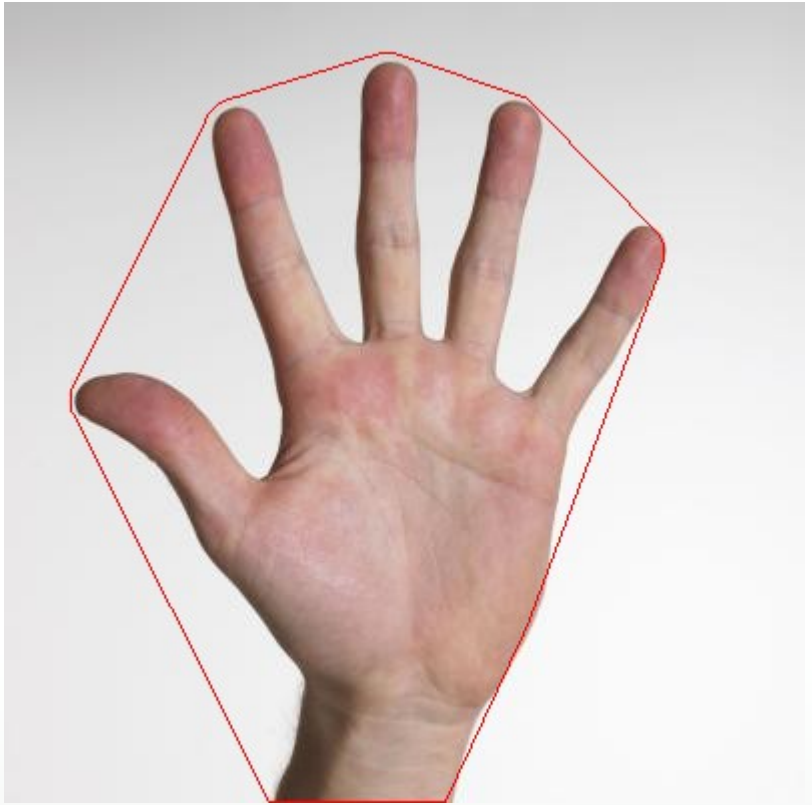
Contorno



- # cria um retângulo em torno da mão
- # variável res é uma cópia da imagem de entrada
- x, y, w, h = cv2.boundingRect(cnt)
- cv2.rectangle(res, (x, y), (x+w, y+h), (0, 0, 255), 0)

Contorno

- hull = cv2.convexHull(cnt)
- cv2.drawContours(res, [hull], 0,(0, 0, 255), 0)
- cv2.imwrite('res3.png', res)

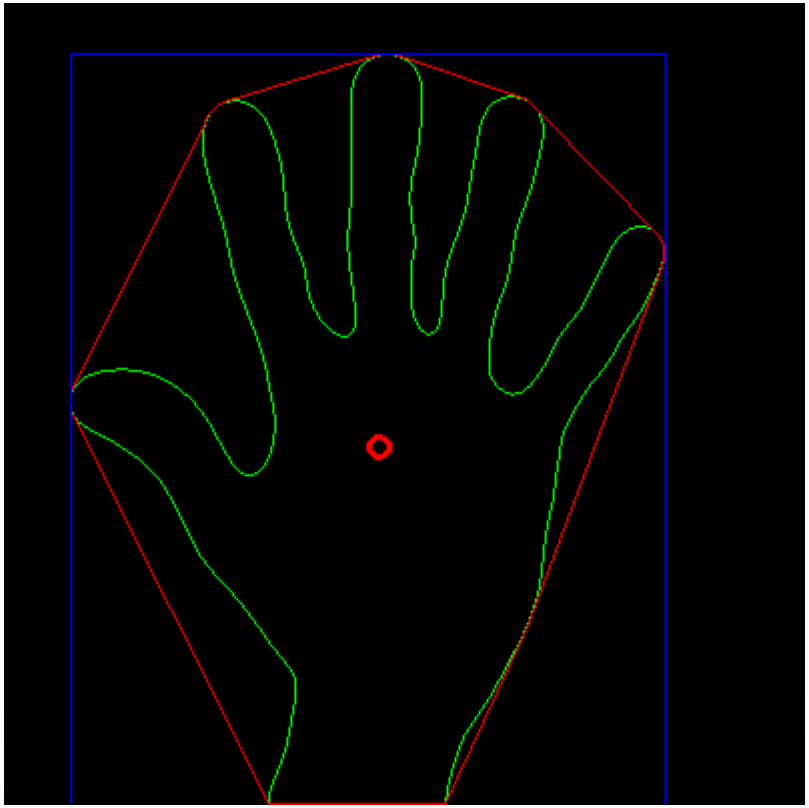


Centro

- moments = cv2.moments(cnt)
- if moments['m00']!=0:
- cx = int(moments['m10']/moments['m00'])
- cy = int(moments['m01']/moments['m00'])
-
- centr=(cx,cy)
- cv2.circle(res,centr,5,[0,0,255],2)
- cv2.imwrite('res4.png', res)

Visualizando

- drawing = np.zeros(hand.shape,np.uint8)
- cv2.circle(drawing,centr,5, [0,0,255],2)
- cv2.drawContours(drawing, [cnt], 0, (0, 255, 0), 0)
- cv2.drawContours(drawing, [hull], 0, (0, 0, 255), 0)
- cv2.rectangle(drawing, (x, y), (x+w, y+h), (255, 0, 0), 0)
- cv2.imwrite('drawing.png', drawing)



Defeitos no Convex Hull

- hull = cv2.convexHull(cnt, returnPoints=False)
-
- defects = cv2.convexityDefects(cnt, hull)
-
- http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contours_more_functions/py_contours_more_functions.html

Marcando os pontos de convexidade

- 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])
-
- cv2.line(res,start, end, [255,255,255], 2)
- cv2.circle(res,far,5,[0,0,255],-1)