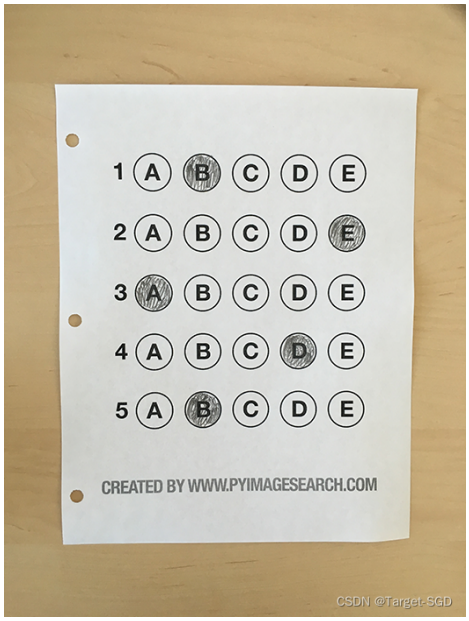
實戰分享！用Python 實現答題卡識別！

AI算法與圖像處理 2022-03-28 16:58

作者| 棒子胡豆

來源丨CSDN博客編輯：AI科技大本營

答題卡素材圖片：



思路

- 讀入圖片，做一些預處理工作。
- 進行輪廓檢測，然後找到該圖片最大的輪廓，就是答題卡部分。
- 進行透視變換，以去除除答題卡外的多餘部分，並且可以對答題卡進行校正。

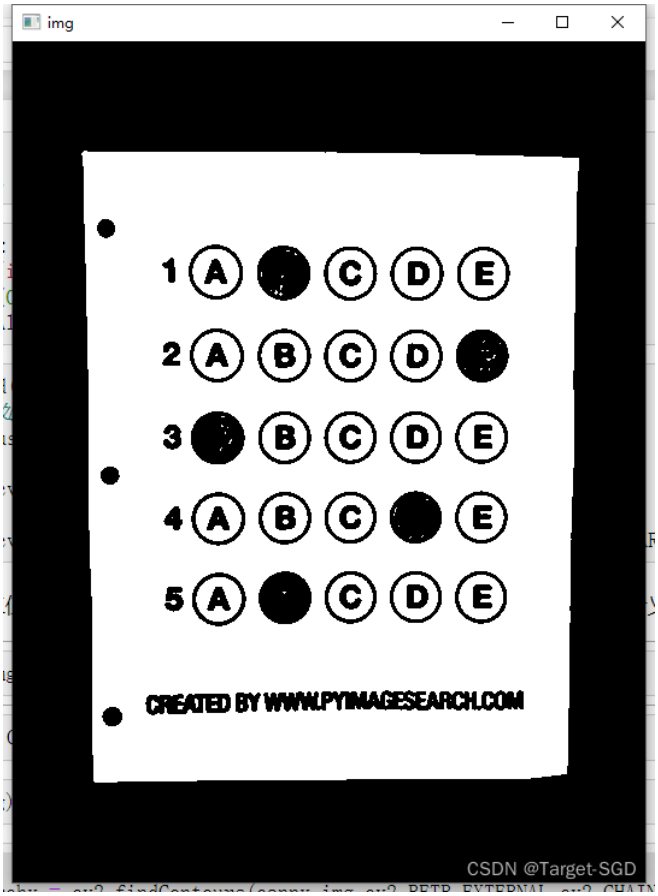
- 再次檢測輪廓，定位每個選項。
- 對選項圓圈先按照豎坐標排序，再按照行坐標排序，這樣就從左到右從上到下的獲得了每個選項輪廓。
- 對每個選項輪廓進行檢查，如果某個選項輪廓中的白色點多，說明該選項被選中，否則就是沒被選上。

細節部分看過程：

1、預處理（去噪，灰度，二值化）

```
img = cv2.imread( "1.png" ,1)
#高斯去噪
img_gs = cv2.GaussianBlur(img,[5,5],0)
# 轉灰度
img_gray = cv2.cvtColor(img_gs,cv2.COLOR_BGR2GRAY)
# 自適應二值化
_,binary_img = cv2.threshold(img_gray,0,255,cv2.THRESH_OTSU|cv2.THRESH_BINARY)
```

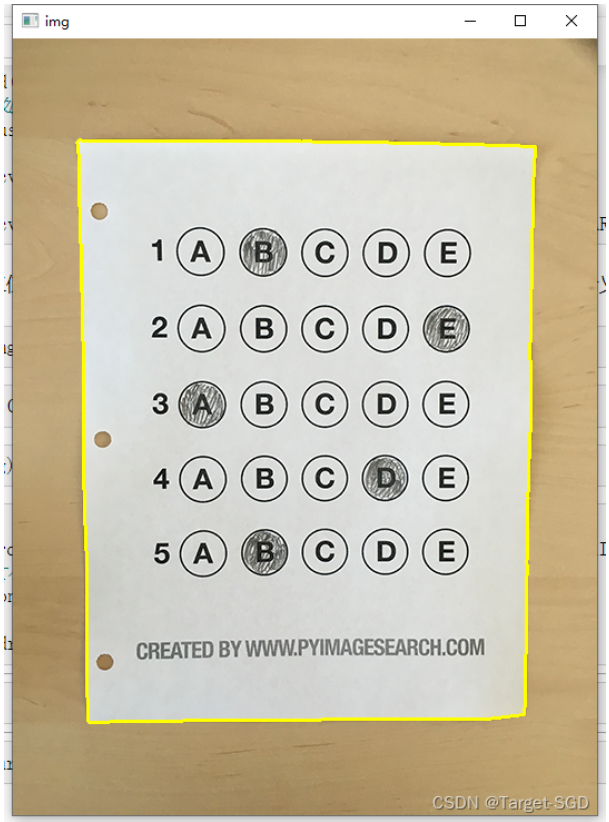
> 注：cv2.THRESH_OTSU|cv2.THRESH_BINARY，該參數指的是自適應閾值+反二值化，做自適應閾值的時候閾值要設置為0



2、輪廓檢測

```
# 找輪廓
contours, hierarchy = cv2.findContours(binary_img,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
# 按照輪廓的面積從大到小排序
cnts = sorted(contours,key = cv2.contourArea,reverse=True)
# 畫輪廓
draw_img = cv2.drawContours(img.copy(),cnts[0],-1,(0,255,255),2)
```

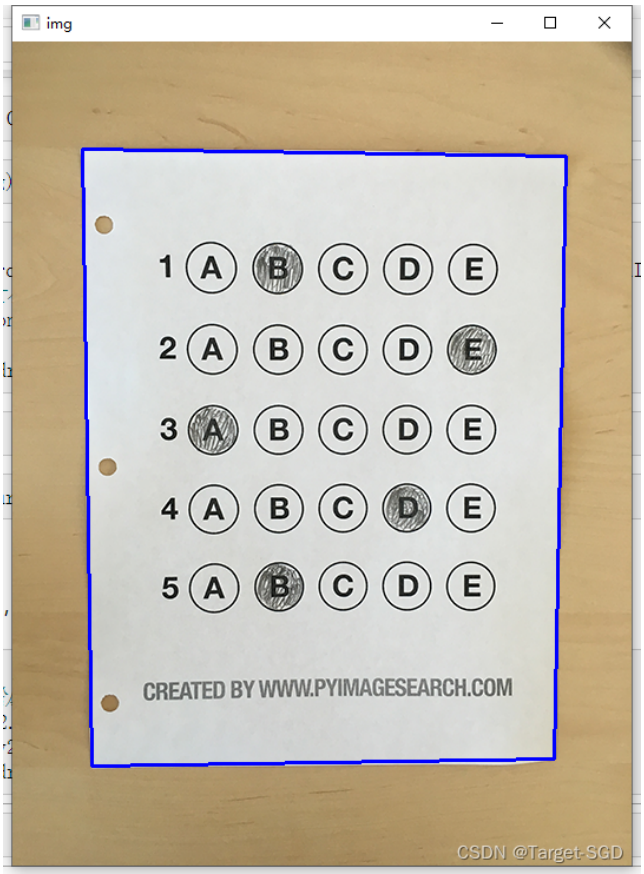
> 注：findContours函數，傳入的圖像應該是二值圖像，cv2.RETR_EXTERNAL指的是只檢測外部輪廓，cv2.CHAIN_APPROX_NONE指的返回輪廓上的所有點。




```
# 輪廓近似
# 閾值，一般為輪廓長度的2%
 alpha = 0.02*cv2.arcLength(cnts[0],True)
approxCurve = cv2.approxPolyDP(cnts[0],alpha,True)
draw_img = cv2.drawContours(img.copy(),[approxCurve],-1,(255,0,0),2)
```

這裡做輪廓近似的目的是，之前檢測到的輪廓看似是一個多邊形，其實本質上是只是點集。

> cv2.approxPolyDP(contour,epsilon,True)，多邊形逼近，第一個參數是點集，第二個參數是精度（原始輪廓的邊界點與擬合多邊形之間的最大距離），第三個參數指新產生的輪廓是否需要閉合，返回值approxCurve為多邊形的點集（按照逆時針排序）。與該函數類似的函數還有cv2.boundingRect（矩形包圍框）cv2.minAreaRect（最小包圍矩形框），cv2.minEnclosingCircle（最小包圍圓形）cv2.filtEllipse（最優擬合橢圓）cv2.filtLine（最優擬合直線），cv2.minEnclosingTriangle（最小外包三角形）

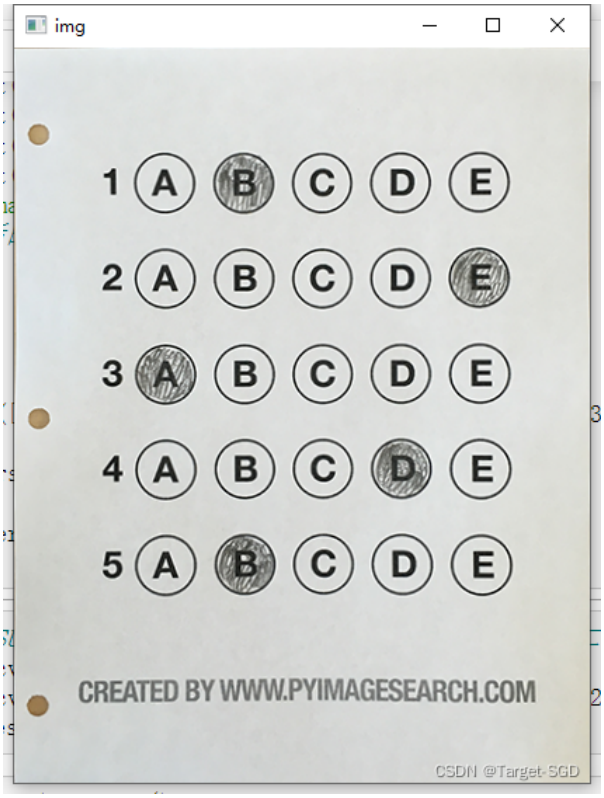


3、透視變換

```
#透視變換
# 矩形的四個頂點為approxCurve[0][0],approxCurve[1][0],approxCurve[2][0],approxCurve[3][0]
# 分別表示矩形的TL,BL,BR,TR四個點
a1 = list(approxCurve[0][0])
a2 = list(approxCurve[1][0])
a3 = list(approxCurve[2][0])
a4 = list(approxCurve[3][0])
# 原始矩陣
mat1 = np.array([a1,a2,a3,a4],dtype = np.float32)

# 計算矩形的w和h
 w1 = int(np.sqrt((a1[0]-a4[0])**2+(a1[1]-a4[1])**2))
w2 = int(np.sqrt((a2[0]-a3[0])**2+(a2[1]-a3[1])**2))
```

```
h1 = int(np.sqrt((a1[0]-a2[0])**2+(a1[1]-a2[1])**2))
h2 = int(np.sqrt((a3[0]-a4[0])**2+(a3[1]-a4[1])**2))
w,h=max(w1,w2),max(h1,h2)
# 計算透視變換後的坐標
new_a1 = [0,0]
new_a2 = [0,h]
new_a3 = [w,h]
new_a4 = [w,0]
# 目標矩陣
mat2 = np.array([new_a1,new_a2,new_a3,new_a4],dtype = np.float32)
# 透視變換矩陣
mat = cv2.getPerspectiveTransform(mat1,mat2)
# 進行透視變換
res = cv2.warpPerspective(img,mat,(w,h))
imshow((res))
```



透視變換的計算步驟：

1. 首先獲取原圖多邊形的四個頂點，注意頂點順序。
2. 然後構造原始頂點矩陣。
3. 計算矩形長寬，構造變換後的目標矩陣。
4. 獲取原始矩陣到目標矩陣的透視變換矩陣
5. 進行透視變換

4、輪廓檢測，檢測每個選項

```
res_gray = cv2.cvtColor(res,cv2.COLOR_BGR2GRAY)
_,binary_res = cv2.threshold(res_gray,0,255,cv2.THRESH_OTSU|cv2.THRESH_BINARY_INV)
contours = cv2.findContours(binary_res,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)[0]
dst = cv2.drawContours(res.copy(),contours,-1,(0,0,255),1)
imshow(dst)
```

篩選選項輪廓

```
# 挑選合適的輪廓
def check(contours):
    ans = []
    for  i  in  contours:
        area =  float (cv2.contourArea(i))
        length =  float (cv2.arcLength(i,True))
        if  area<=0 or length<=0:
            continue
        if  area/length >7.05 and area/length<10.5:
            ans.append(i)
    return  ans
ans_contours = check(contours)
dst_new = cv2.drawContours(res.copy(),ans_contours,-1,(0,255,255),3  )
imshow(dst_new)
```

5、畫輪廓的外接圓，排序，定位每個選項

```
# 遍歷每一個圓形輪廓，畫外接圓
circle = []
for  i  in  ans_contours:
    (x,y),r = cv2.minEnclosingCircle(i)
    center = (int(x),int(y))
    r = int(r)
    circle.append((center,r))
# 按照外接圓的水平坐標排序center[1]，也就是圓心的高度h，或者y坐標
circle.sort(key = lambda x:x[0][1])
A = []
for  i  in  range(1,6):
    now = circle[(i-1)*5:i*5]
    now.sort(key = lambda x:x[0][0])
    A.extend(now)
```

每個選項按照圓心從左到右，從上到下的順序保存在了A中

6、選項檢測

思路：對於A中的每個選項圓，計算它有所覆蓋的坐標，然後判斷這些坐標在二值圖像中對應的值，統計白色點的個數， 如果白色點所佔的比例比較大的話，說明該選項被選中。

```
def dots_distance(dot1,dot2):
```

```
    #計算二維空間中兩個點的距離
    return  ((dot1[0]-dot2[0])**2+(dot1[1]-dot2[1])**2)**0.5
def count_dots(center,radius):
    #輸入圓的中心點與半徑，返回圓內所有的坐標
    dots = []
    for  i  in  range(-radius,radius+1):
        for  j  in  range(-radius,radius+1):
            dot2 = (center[0]+i,center[1]+j)
            if  dots_distance(center,dot2) <= radius:
                dots.append(dot2)
    return  dots

da = []
for  i  in  A:
    dots = count_dots(i[0],i[1])
    all_dots = len(dots)
    whilt_dots = 0
    for  j in  dots:
        if  binary_res[j[1]][j[0]] == 255:
            whilt_dots = whilt_dots+1
    if  whilt_dots/all_dots>=0.4:
        da.append(1)
    else :
        da.append(0)
da = np.array(da)
da = np.reshape(da,(5,5))
```

```
da
```

```
array([[0,  1,  0,  0,  0],
       [0,  0,  0,  0,  1],
       [1,  0,  0,  0,  0],
       [0,  0,  0,  1,  0],
       [0,  1,  0,  0,  0]])
```

這樣每個答題卡就轉換成了一個二維數組，接下來在做一些簡單的收尾工作就可以了。

閱讀原文

喜歡此內容的人還喜歡

C函數指針別再停留在語法，得上升到軟件設計~

嵌入式資訊精選

Python 的__name__ 變量，到底是個什麼東西?

小白學視覺

IoU、GIoU、DIoU、CIoU損失函數的那點事兒

小白學視覺