**opencv学习笔记十八：模板匹配(cv2.matchTemplate、cv2.minMaxLoc)**

**1、目标匹配函数：cv2.matchTemplate()**

res=cv2.matchTemplate(image, templ, method, result=None, mask=None)

image：待搜索图像  
templ：模板图像  
result：匹配结果  
method：计算匹配程度的方法,主要有以下几种

| **method** | **含义** |
| --- | --- |
| CV\_TM\_SQDIFF | 平方差匹配法：该方法采用平方差来进行匹配；最好的匹配值为0；匹配越差，匹配值越大。 |
| CV\_TM\_CCORR | 相关匹配法：该方法采用乘法操作；数值越大表明匹配程度越好。 |
| CV\_TM\_CCOEFF | 相关系数匹配法：1表示完美的匹配；-1表示最差的匹配。 |
| CV\_TM\_SQDIFF\_NORMED | 计算归一化平方差，计算出来的值越接近0，越相关 |
| CV\_TM\_CCORR\_NORMED | 计算归一化相关性，计算出来的值越接近1，越相关 |
| CV\_TM\_CCOEFF\_NORMED | 计算归一化相关系数，计算出来的值越接近1，越相关 |

**2、获取匹配结果函数：cv2.minMaxLoc()**

min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(ret)

参数说明：min\_val， max\_val, min\_loc, max\_loc 分别表示最小值，最大值，以及最小值和最大值对应的图像中的位置， ret就是cv2.matchTemplate()函数返回的矩阵

# 模板匹配

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

template = cv2.imread('face.jpg', 0)

h, w = template.shape[:2]

methods = ['cv2.TM\_CCOEFF', 'cv2.TM\_CCOEFF\_NORMED', 'cv2.TM\_CCORR',

'cv2.TM\_CCORR\_NORMED', 'cv2.TM\_SQDIFF', 'cv2.TM\_SQDIFF\_NORMED']

res = cv2.matchTemplate(img, template, cv2.TM\_SQDIFF)

min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(res)

for meth in methods:

img2 = img.copy()

# 匹配方法的真值

method = eval(meth)

res = cv2.matchTemplate(img, template, method)

min\_val, max\_val, min\_loc, max\_loc = cv2.minMaxLoc(res)

# 如果是平方差匹配TM\_SQDIFF或归一化平方差匹配TM\_SQDIFF\_NORMED，取最小值

if method in [cv2.TM\_SQDIFF, cv2.TM\_SQDIFF\_NORMED]:

top\_left = min\_loc

else:

top\_left = max\_loc

bottom\_right = (top\_left[0] + w, top\_left[1] + h)

# 画矩形

cv2.rectangle(img2, top\_left, bottom\_right, 255, 2)

plt.subplot(121), plt.imshow(res, cmap='gray')

plt.xticks([]), plt.yticks([]) # 隐藏坐标轴

plt.subplot(122), plt.imshow(img2, cmap='gray')

plt.xticks([]), plt.yticks([])

plt.suptitle(meth)

plt.show()

  
  
  
  
  


多对象匹配：我们匹配的是图中的金币，读入的两张图分别是原图和金币模板

img\_rgb = cv2.imread('mario.jpg')

img\_gray = cv2.cvtColor(img\_rgb, cv2.COLOR\_BGR2GRAY)

template = cv2.imread('mario\_coin.jpg', 0)

h, w = template.shape[:2]

res = cv2.matchTemplate(img\_gray, template, cv2.TM\_CCOEFF\_NORMED)

threshold = 0.8

# 取匹配程度大于%80的坐标

loc = np.where(res >= threshold)

#np.where返回的坐标值(x,y)是(h,w)，注意h,w的顺序

for pt in zip(\*loc[::-1]):

bottom\_right = (pt[0] + w, pt[1] + h)

cv2.rectangle(img\_rgb, pt, bottom\_right, (0, 0, 255), 2)

cv2.imshow('img\_rgb', img\_rgb)

cv2.waitKey(0)

