模板匹配

电子信息工程系

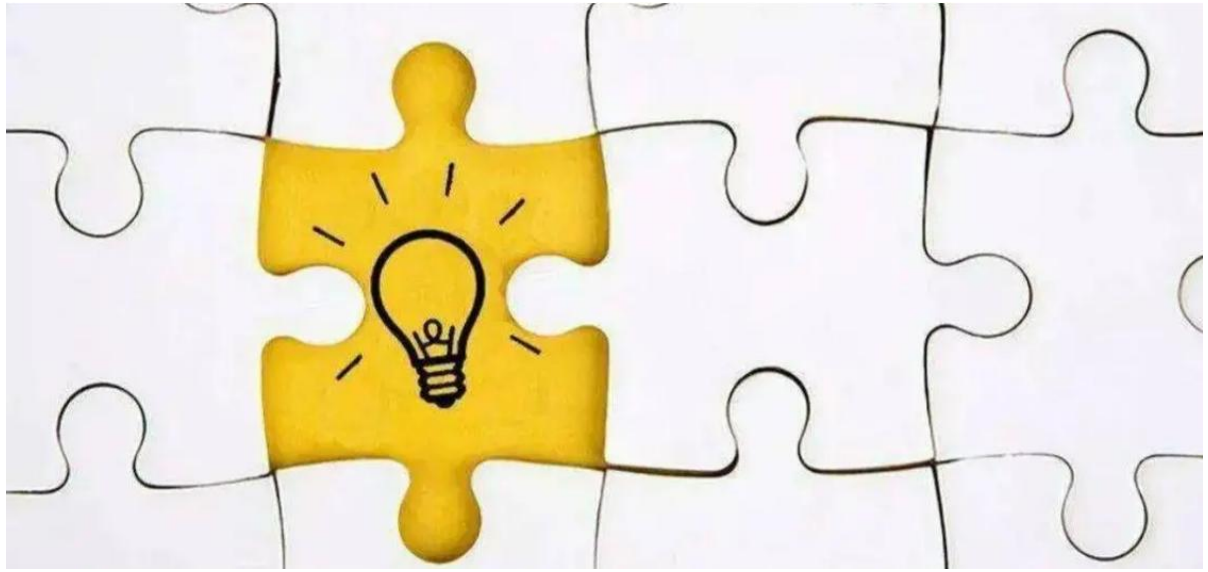
袁羽

目录

CONTENTS

一 模板匹配

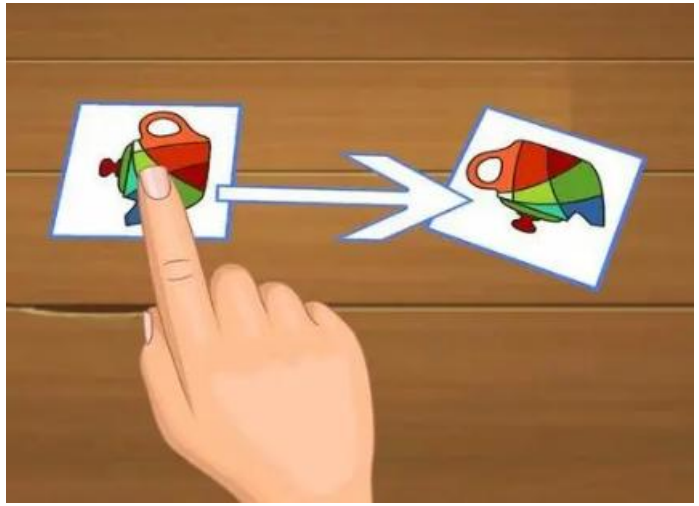
模板就是一副已知的小图像，而模板匹配就是在一副大图像中搜寻目标，已知该图中有要找的目标，且该目标同模板有相同的尺寸、方向和图像元素，通过一定的算法可以在图中找到目标，确定其坐标位置。



一 模板匹配

模板匹配是一种最原始、最基本的模式识别方法，研究某一特定对象物的图案位于图像的什么地方，进而识别对象，这就是一个匹配问题。它是图像处理中最基本、最常用的匹配方法。

模板匹配具有自身的局限性，主要表现在它只能进行平行移动，若原图像中的匹配目标发生旋转或大小变化，该算法无效。



一 模板匹配

模板是被查找目标的图像，查找模板在原始图像中的哪个位置的过程就叫模板匹配。OpenCV提供的matchTemplate()方法就是模板匹配方法，其语法如下：

result = cv2.matchTemplate(image, templ, method, mask)

参数说明：

image：原始图像。

templ：模板图像，尺寸必须小于或等于原始图像。

method：匹配的方法。

mask：可选参数。掩模，只有cv2.TM_SQDIFF和cv2.TM_CCORR_NORMED支持此参数，建议采用默认值。

返回值说明：

result：计算得出的匹配结果。如果原始图像的宽、高分别为W、H，模板图像的宽、高分别为w、h，result就是一个W-w+1列、H-h+1行的32位浮点型数组。数组中每一个浮点数都是原始图像中对应像素位置的匹配结果，其含义需要根据method参数来解读。

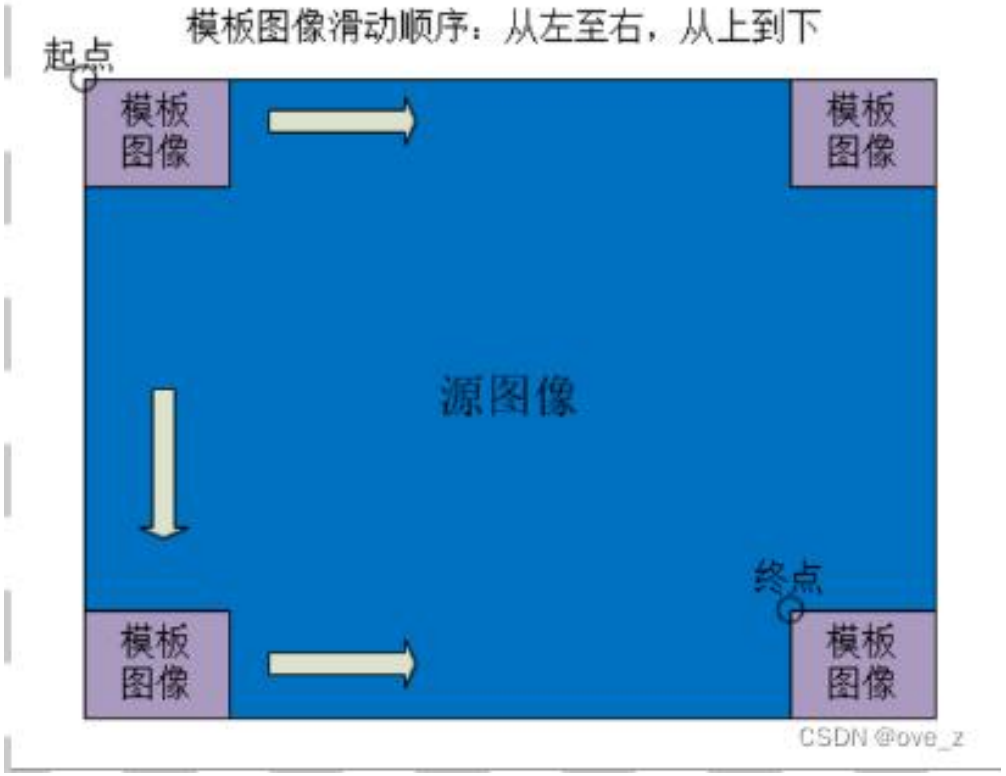
一 模板匹配

匹配的方法：

匹配方法	含义
cv2.TM_SQDIFF	该方法使用平方差进行匹配，因此最佳的匹配结果在结果为0处，值越大匹配结果越差。
cv2.TM_SQDIFF_NORMED	该方法使用归一化的平方差进行匹配，最佳匹配也在结果为0处。
cv2.TM_CCORR	相关性匹配方法，该方法使用源图像与模板图像的卷积结果进行匹配，因此，最佳匹配位置在值最大处，值越小匹配结果越好。
cv2.TM_CCORR_NORMED	归一化的相关性匹配方法，与相关性匹配方法类似，最佳匹配位置也是在值最大处。
cv2.TM_CCOEFF	相关性系数匹配方法，该方法使用源图像与其均值的差、模板与其均值的差二者之间的相关性进行匹配，最佳匹配结果在值等于1处，最差匹配结果在值等于-1处，值等于0直接表示二者不相关。
cv2.TM_CCOEFF_NORMED	归一化的相关性系数匹配方法，正值表示匹配的结果较好，负值则表示匹配的效果较差，也是值越大，匹配效果也好

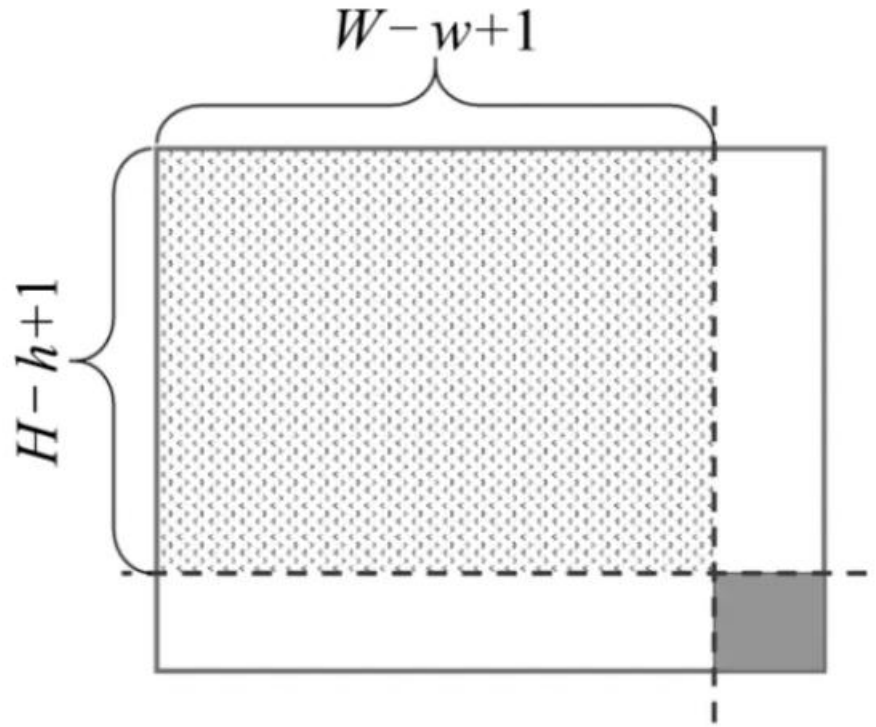
一 模板匹配

在模板匹配的计算过程中，模板会在原始图像中移动。模板与重叠区域内的像素逐个对比，最后将对比的结果保存在模板左上角像素点索引位置对应的数组位置中。



一 模板匹配

在模板匹配的计算过程中，模板会在原始图像中移动。模板与重叠区域内的像素逐个对比，最后将对比的结果保存在模板左上角像素点索引位置对应的数组位置中。



模板将原始图像中每一块区域都覆盖一遍，但结果数组的行、列数并不等于原始图像的像素的行、列数。假设模板的宽为w，高为h，原始图像的宽为W，高为H。模板移动到原始图像的边缘之后就不会继续移动了，所以模板的移动区域该区域的边长为“原始图像边长-模板边长+1”，最后加1是因为移动区域内的上下、左右的2个边都被模板覆盖到了，如果不加1会丢失数据。

一 模板匹配

在模板匹配的计算过程中，模板会在原始图像中移动。模板与重叠区域内的像素逐个对比，最后将对比的结果保存在模板左上角像素点索引位置对应的数组位置中。

使用cv2.TM_SQDIFF（平方差匹配）方法计算出的数组格式如下（其他方法计算出的数组格式相同，仅数值不同）：

```
[[0.10165964 0.10123613 0.1008469  ... 0.10471864 0.10471849 0.10471849]
 [0.10131165 0.10087635 0.10047968 ... 0.10471849 0.10471834 0.10471849]
 [0.10089004 0.10045089 0.10006084 ... 0.10471849 0.10471819 0.10471849]
 ...
 [0.16168603 0.16291814 0.16366465 ... 0.12178455 0.12198001 0.12187888]
 [0.15859096 0.16000605 0.16096526 ... 0.12245651 0.12261643 0.12248362]
 [0.15512456 0.15672517 0.15791312 ... 0.12315679 0.1232616  0.12308815]]
```

二 单模板匹配

匹配过程中只用到一个模板场景叫单模板匹配。原始图像中可能只有一个和模板相似的图像，也可能有多个。如果只获取匹配程度最高的那一个结果，这种操作叫作单目标匹配。如果需要同时获取所有匹配程度较高的结果，这种操作叫作多目标匹配。

二 单模板匹配

1.单目标匹配

单目标匹配只获取一个结果即可，就是匹配程度最高的结果。
matchTemplate()方法的计算结果是一个二维数组。如果使用平方差匹配，则最小值为最佳匹配结果；如果使用相关匹配或相关系数匹配，则最大结为匹配结果。

如何确定二维数组中的最大值或最小值?

二 单模板匹配

1.单目标匹配

OpenCV提供了一个minMaxLoc()方法专门用来解析这个二维数组中的最大值、最小值以及这2个值对应的坐标，minMaxLoc()方法的语法如下：

minValue, maxValue, minLoc, maxLoc = cv2.minMaxLoc(src, mask)

参数说明：

src：matchTemplate()方法计算得出的数组。

mask：可选参数，掩模，建议使用默认值。

返回值说明：

minValue：数组中的最小值。

maxValue：数组中的最大值。

minLoc：最小值的坐标，格式为(x, y)。

maxLoc：最大值的坐标，格式为(x, y)。

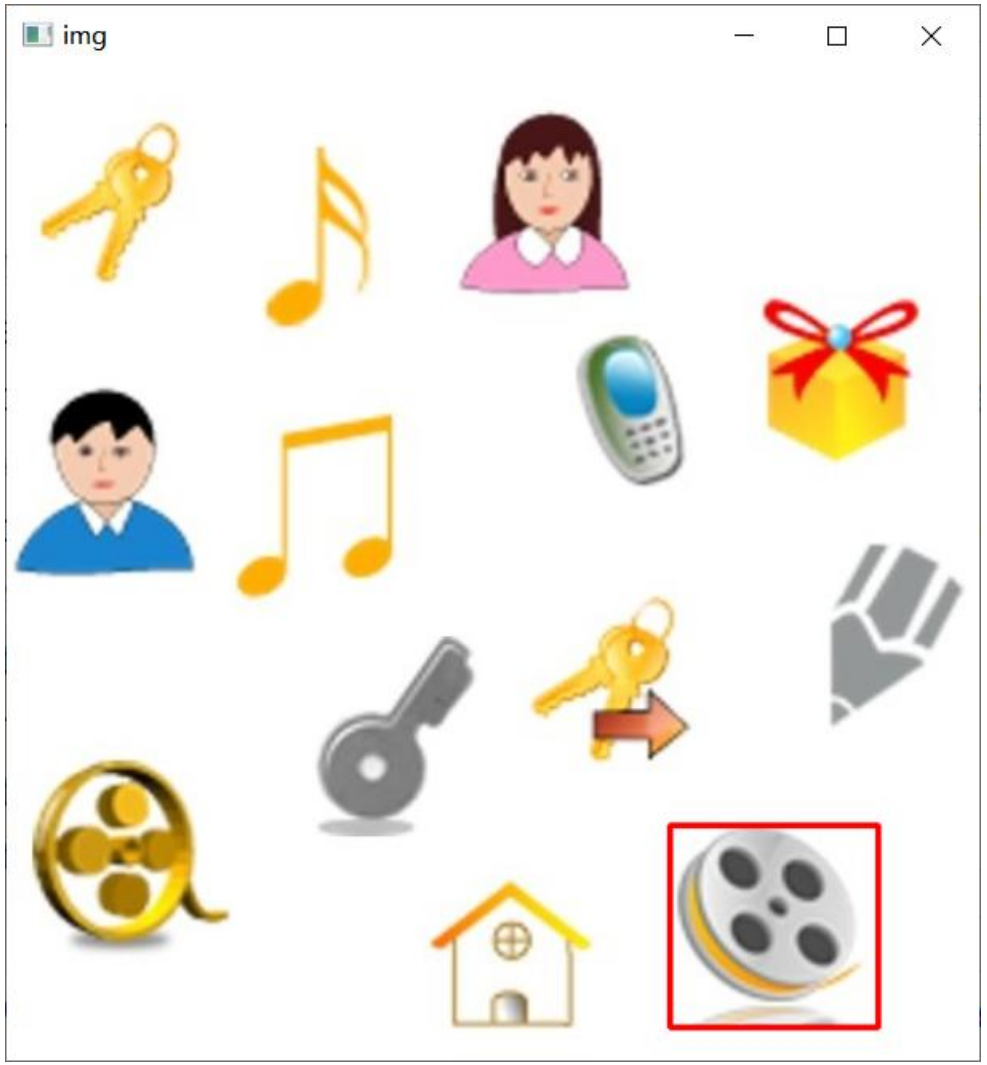
平方差匹配的计算结果越小，匹配程度越高。minMaxLoc()方法返回的minValue值就是模板匹配的最优结果，minLoc就是最优结果区域左上角的点坐标，区域大小与模板大小一致。

例 为原始图片中匹配成功的区域绘制红框:使用cv2.TM_SQDIFF_NORMED方式进行模板匹配，在原始图像中找到与模板一样的图案，并在该图案上绘制红色方框。

模板

原始图像

例 为原始图片中匹配成功的区域绘制红框:使用cv2.TM_SQDIFF_NORMED方式进行模板匹配，在原始图像中找到与模板一样的图案，并在该图案上绘制红色方框。



例 为原始图片中匹配成功的区域绘制红框：使用cv2.TM_SQDIFF_NORMED方式进行模板匹配，在原始图像中找到与模板一样的图案，并在该图案上绘制红色方框。

```
import cv2
img = cv2.imread("background.jpg")  # 读取原始图像
templ = cv2.imread("template.png")  # 读取模板图像
height, width,c = templ.shape  # 获取模板图像的宽度、高度和通道数
results = cv2.matchTemplate(img, templ, cv2.TM_SQDIFF_NORMED)  # 按照标准平方差方式匹配
# 获取匹配结果中的最小值、最大值、最小值坐标和最大值坐标
minValue, maxValue, minLoc, maxLoc = cv2.minMaxLoc(results)
resultPoint1 = minLoc  # 将最小值坐标当做最佳匹配区域的左上角点坐标
# 计算出最佳匹配区域的右下角点坐标
resultPoint2 = (resultPoint1[0] + width, resultPoint1[1] + height)
# 在最佳匹配区域位置绘制红色方框，线宽为2像素
cv2.rectangle(img, resultPoint1, resultPoint2, (0, 0, 255), 2)
cv2.imshow("img", img)  # 显示匹配的结果
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

练习 在集体合照中找到某个人的位置

练习 寻找车牌


贵D·B2888
贵D·GV807
浙J·OUX65
贵D·W7332
贵D·AB887
贵D·N8183
贵D·V3663
贵H·BV181
贵D·56777
贵D·EM297
贵D·X5050
贵D·W7250
湘K·N8789
贵D·FM865

二 单模板匹配

2.多目标匹配

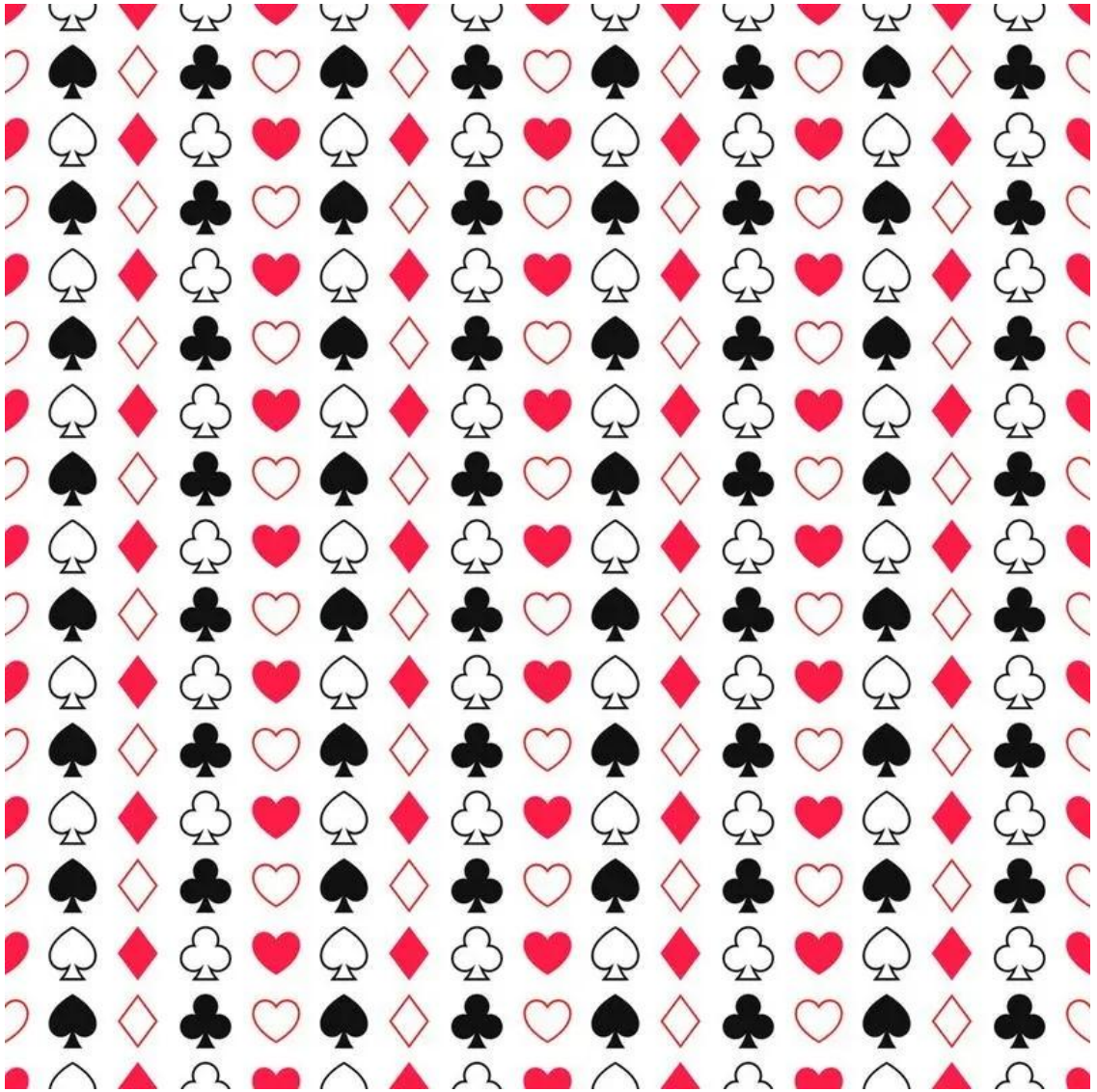
多目标匹配需要将原始图像中所有与模板相似的图像都找出来，使用相关匹配或相关系数匹配可以很好地实现这个功能。如果计算结果大于某值（例如0.999），则认为匹配区域的图案和模板是相同的。

例 为原始图片中所有匹配成功的图案绘制红框：原始图像中有很多重复的图案，每一个与模板相似的图案都需要被标记出来

例 为原始图片中所有匹配成功的图案绘制红框：原始图像中有很多重复的图案，每一个与模板相似的图案都需要被标记出来

```
import cv2
img = cv2.imread("background2.jpg")  # 读取原始图像
templ = cv2.imread("template.png")  # 读取模板图像
width, height, c = templ.shape  # 获取模板图像的宽度、高度和通道数
results = cv2.matchTemplate(img, templ, cv2.TM_CCOEFF_NORMED)  # 按照标准相关系数匹配
for y in range(len(results)):  # 遍历结果数组的行
    for x in range(len(results[y])):  # 遍历结果数组的列
        if results[y][x] > 0.99:  # 如果相关系数大于0.99则认为匹配成功
            # 在最佳匹配结果位置绘制红色方框
            cv2.rectangle(img, (x, y), (x + width, y + height), (0, 0, 255), 2)
cv2.imshow("img", img)  # 显示匹配的结果
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

例 为原始图片中所有匹配成功的图案绘制红框并统计数量。



例 为原始图片中所有匹配成功的图案绘制红框并统计数量。

```
import cv2
image = cv2.imread("pk.jpg")  # 读取原始图像
templ = cv2.imread("ht.png")  # 读取模板图像
height, width, c = templ.shape  # 获取模板图像的高度、宽度和通道数
results = cv2.matchTemplate(image, templ, cv2.TM_CCOEFF_NORMED)  # 按照标准相关系数匹配
station_Num = 0  # 初始化快轨的站台个数为0
for y in range(len(results)):  # 遍历结果数组的行
    for x in range(len(results[y])):  # 遍历结果数组的列
        if results[y][x] > 0.99:  # 如果相关系数大于0.99则认为匹配成功
            # 在最佳匹配结果位置绘制蓝色矩形边框
            cv2.rectangle(image, (x, y), (x + width, y + height), (255, 0, 0), 2)
            station_Num += 1  # 快轨的站台个数加1
cv2.putText(image, "the numbers is: " + str(station_Num), (0, 30),
        cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 1)  # 在原始图像绘制总数
cv2.imshow("result", image)  # 显示匹配的结果
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

三 多模板匹配

匹配过程中同时查找多个模板的操作叫多模板匹配。多模板匹配实际上就是进行了n次“单模板多目标匹配”操作，n的数量为模板总数。

例 多模板匹配。

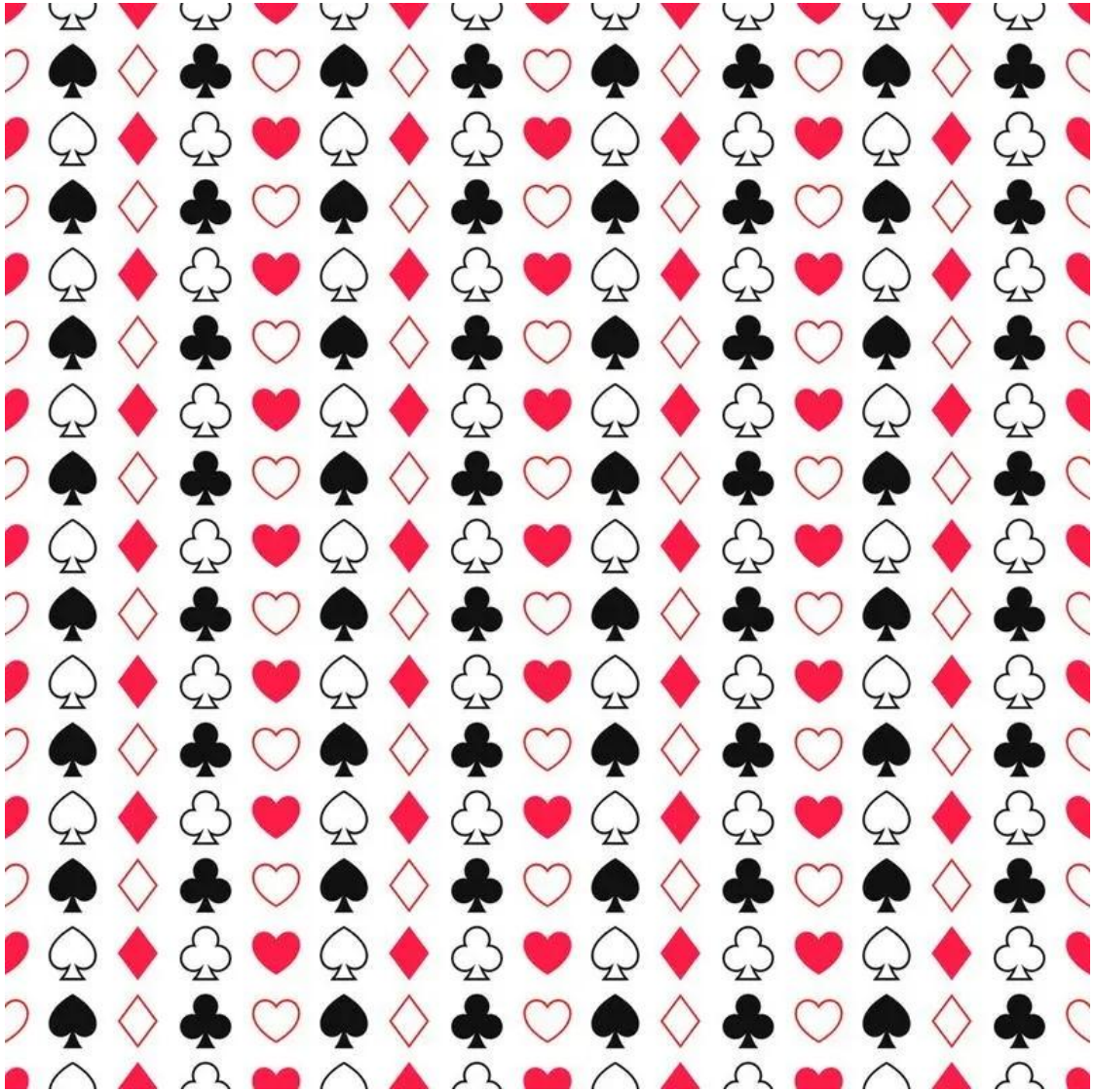
每一个模板都要做一次“单模板多目标匹配”，最后把所有模板的匹配结果汇总到一起。“单模板多目标匹配”的过程可以封装成一个方法，方法参数为模板和原始图像，方法内部将计算结果再加工一下，直接返回所有红框左上角和右下角两点横纵坐标的列表。

例 多模板匹配。

```
import cv2
def myMatchTemplate(img, templ):  # 自定义方法：获取模板匹配成功后所有红框位置的坐标
    width, height, c = templ.shape  # 获取模板图像的宽度、高度和通道数
    results = cv2.matchTemplate(img, templ, cv2.TM_CCOEFF_NORMED)  # 按照标准相关系数匹配
    loc = list()  # 红框的坐标列表
    for i in range(len(results)):  # 遍历结果数组的行
        for j in range(len(results[i])):  # 遍历结果数组的列
            if results[i][j] > 0.99:  # 如果相关系数大于0.99则认为匹配成功
                # 在列表中添加匹配成功的红框对角线两点坐标
                loc.append((j, i, j + width, i + height))
    return loc
```

```
img = cv2.imread("background2.jpg")  # 读取原始图像
templs = list()  # 模板列表
templs.append(cv2.imread("template.png"))  # 添加模板1
templs.append(cv2.imread("template2.png"))  # 添加模板2
templs.append(cv2.imread("template3.png"))  # 添加模板3
loc = list()  # 所有模板匹配成功位置的红框坐标列表
for t in templs:  # 遍历所有模板
    loc += myMatchTemplate(img, t)  # 记录该模板匹配得出的
for i in loc:  # 遍历所有红框的坐标
    cv2.rectangle(img, (i[0], i[1]), (i[2], i[3]), (0, 0, 255), 2)  # 在图片中绘制红框
cv2.imshow("img", img)  # 显示匹配的结果
cv2.waitKey()  # 按下任何键盘按键后
cv2.destroyAllWindows()  # 释放所有窗体
```

练　多模板匹配并统计数量。



小结

模板匹配包括单模板匹配和多模板匹配，单模板匹配又包括单目标匹配和多目标匹配。实现这些内容的基础方法就是模板匹配方法，即matchTemplate()方法。其中，重点掌握模板匹配方法的6个参数值。

此外，为了实现单目标匹配，除了需要使用模板匹配方法matchTemplate()外，还要使用minMaxLoc()方法，这个方法返回的就是单目标匹配的最优结果。对于多目标匹配，要将它和多模板匹配区分开：多目标匹配只有一个模板，而多模板匹配则有多个模板。

THANKS