**opencv+python实现图像匹配----模板匹配、特征点匹配**

**模板匹配与特征匹配**

* **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，进而识别对象物，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。
* **特征匹配**：所谓特征匹配(FBM)，就是指将从影像中提取的特征作为共轭实体,而将所提特征属性或描述参数(实际上是特征的特征，也可以认为是影像的特征)作为匹配实体，通过计算匹配实体之间的相似性测度以实现共轭实体配准的影像匹配方法。在匹配目标发生旋转或大小变化时，该算法依旧有效。

**python的版本及依赖的库的安装**

#版本python 3.7.1

pip install numpy==1.15.3

pip install matplotlib==3.0.1

pip install opencv-python==3.4.2.16

pip install opencv-contrib-python==3.4.2.16

**opencv+python模板匹配**[1](https://blog.csdn.net/zhuisui_woxin/article/details/84400439#fn1)

**匹配材料**

* 目标图片：target.jpg  
  
* 模板图片：template.jpg  
  template

**模板匹配Template Matching----单目标匹配**

#opencv模板匹配----单目标匹配

import cv2

#读取目标图片

target = cv2.imread("target.jpg")

#读取模板图片

template = cv2.imread("template.jpg")

#获得模板图片的高宽尺寸

theight, twidth = template.shape[:2]

#执行模板匹配，采用的匹配方式cv2.TM\_SQDIFF\_NORMED

result = cv2.matchTemplate(target,template,cv2.TM\_SQDIFF\_NORMED)

#归一化处理

cv2.normalize( result, result, 0, 1, cv2.NORM\_MINMAX, -1 )

#寻找矩阵（一维数组当做向量，用Mat定义）中的最大值和最小值的匹配结果及其位置

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

#匹配值转换为字符串

#对于cv2.TM\_SQDIFF及cv2.TM\_SQDIFF\_NORMED方法min\_val越趋近与0匹配度越好，匹配位置取min\_loc

#对于其他方法max\_val越趋近于1匹配度越好，匹配位置取max\_loc

strmin\_val = str(min\_val)

#绘制矩形边框，将匹配区域标注出来

#min\_loc：矩形定点

#(min\_loc[0]+twidth,min\_loc[1]+theight)：矩形的宽高

#(0,0,225)：矩形的边框颜色；2：矩形边框宽度

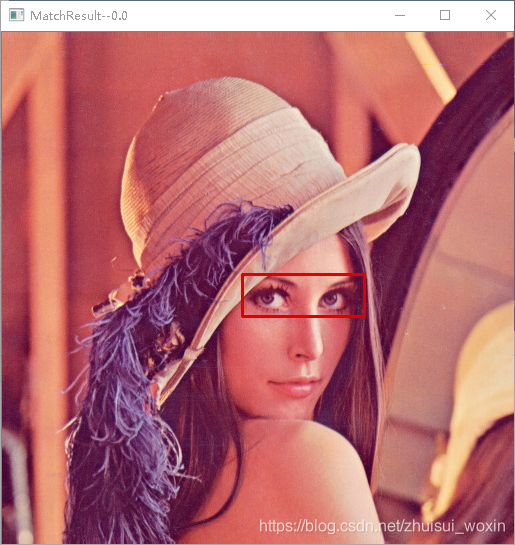
cv2.rectangle(target,min\_loc,(min\_loc[0]+twidth,min\_loc[1]+theight),(0,0,225),2)

#显示结果,并将匹配值显示在标题栏上

cv2.imshow("MatchResult----MatchingValue="+strmin\_val,target)

cv2.waitKey()

cv2.destroyAllWindows()

运行结果：  
  
可以看到显示的min\_val的值为0.0，说明完全匹配。

**模板匹配Template Matching----多目标匹配**

* 目标图片：target.jpg  
  

#opencv模板匹配----多目标匹配

import cv2

import numpy

#读取目标图片

target = cv2.imread("target.jpg")

#读取模板图片

template = cv2.imread("template.jpg")

#获得模板图片的高宽尺寸

theight, twidth = template.shape[:2]

#执行模板匹配，采用的匹配方式cv2.TM\_SQDIFF\_NORMED

result = cv2.matchTemplate(target,template,cv2.TM\_SQDIFF\_NORMED)

#归一化处理

#cv2.normalize( result, result, 0, 1, cv2.NORM\_MINMAX, -1 )

#寻找矩阵（一维数组当做向量，用Mat定义）中的最大值和最小值的匹配结果及其位置

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

#绘制矩形边框，将匹配区域标注出来

#min\_loc：矩形定点

#(min\_loc[0]+twidth,min\_loc[1]+theight)：矩形的宽高

#(0,0,225)：矩形的边框颜色；2：矩形边框宽度

cv2.rectangle(target,min\_loc,(min\_loc[0]+twidth,min\_loc[1]+theight),(0,0,225),2)

#匹配值转换为字符串

#对于cv2.TM\_SQDIFF及cv2.TM\_SQDIFF\_NORMED方法min\_val越趋近与0匹配度越好，匹配位置取min\_loc

#对于其他方法max\_val越趋近于1匹配度越好，匹配位置取max\_loc

strmin\_val = str(min\_val)

#初始化位置参数

temp\_loc = min\_loc

other\_loc = min\_loc

numOfloc = 1

#第一次筛选----规定匹配阈值，将满足阈值的从result中提取出来

#对于cv2.TM\_SQDIFF及cv2.TM\_SQDIFF\_NORMED方法设置匹配阈值为0.01

threshold = 0.01

loc = numpy.where(result<threshold)

#遍历提取出来的位置

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

#第二次筛选----将位置偏移小于5个像素的结果舍去

if (temp\_loc[0]+5<other\_loc[0])or(temp\_loc[1]+5<other\_loc[1]):

numOfloc = numOfloc + 1

temp\_loc = other\_loc

cv2.rectangle(target,other\_loc,(other\_loc[0]+twidth,other\_loc[1]+theight),(0,0,225),2)

str\_numOfloc = str(numOfloc)

#显示结果,并将匹配值显示在标题栏上

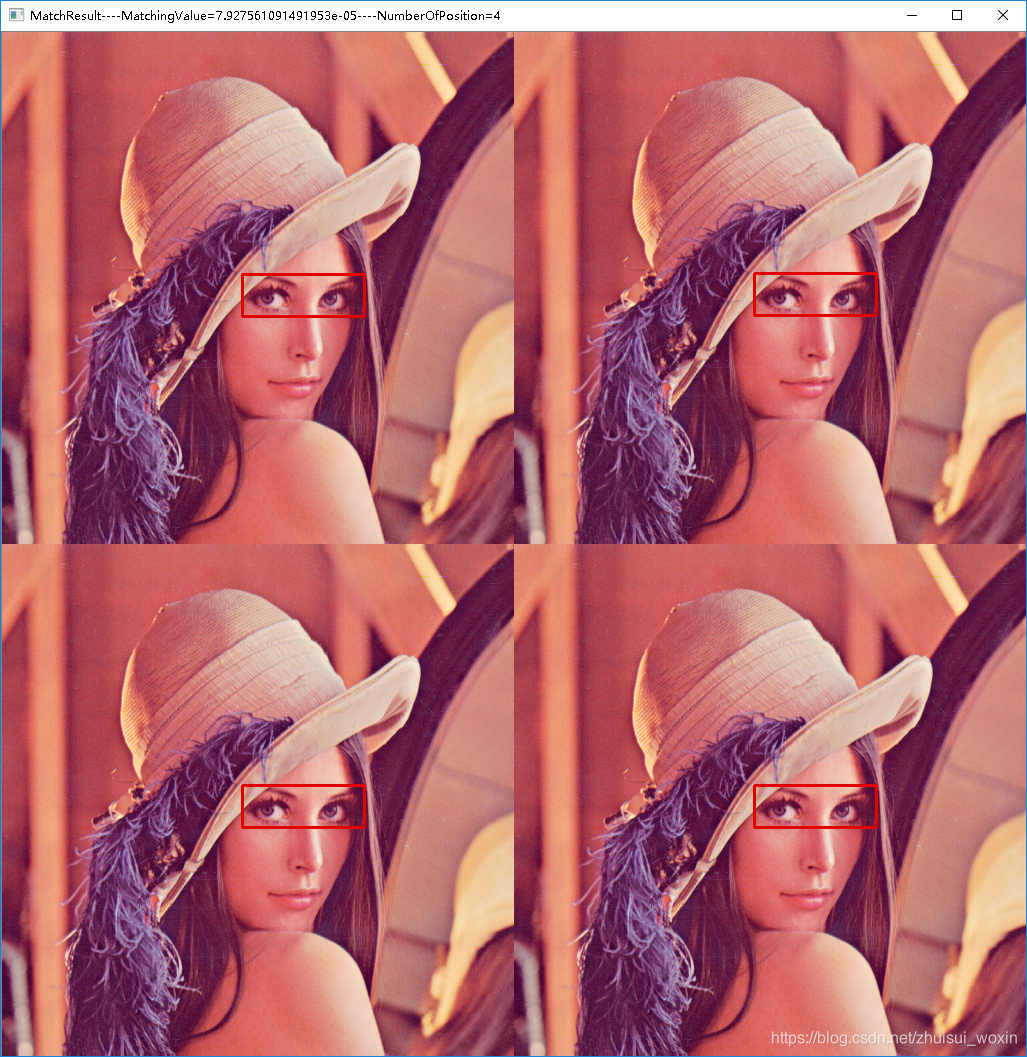
strText = "MatchResult----MatchingValue="+strmin\_val+"----NumberOfPosition="+str\_numOfloc

cv2.imshow(strText,target)

cv2.waitKey()

cv2.destroyAllWindows()

https://csdnimg.cn/release/blogv2/dist/pc/img/newCodeMoreWhite.png

运行结果：  


**opencv+python特征匹配**[2](https://blog.csdn.net/zhuisui_woxin/article/details/84400439#fn2)

**匹配材料**

* 目标图片：target.jpg  
  
* 模板图片：template\_adjst.jpg  
  

**BFMatching描述特征点–运行结果不精确**

#opencv----特征匹配----BFMatching

import cv2

from matplotlib import pyplot as plt

#读取需要特征匹配的两张照片，格式为灰度图。

template=cv2.imread("template\_adjust.jpg",0)

target=cv2.imread("target.jpg",0)

orb=cv2.ORB\_create()#建立orb特征检测器

kp1,des1=orb.detectAndCompute(template,None)#计算template中的特征点和描述符

kp2,des2=orb.detectAndCompute(target,None) #计算target中的

bf = cv2.BFMatcher(cv2.NORM\_HAMMING,crossCheck=True) #建立匹配关系

mathces=bf.match(des1,des2) #匹配描述符

mathces=sorted(mathces,key=lambda x:x.distance) #据距离来排序

result= cv2.drawMatches(template,kp1,target,kp2,mathces[:40],None,flags=2) #画出匹配关系

plt.imshow(result),plt.show() #matplotlib描绘出来

**基于FLANN的匹配器(FLANN based Matcher)描述特征点**

#

'''

基于FLANN的匹配器(FLANN based Matcher)

1.FLANN代表近似最近邻居的快速库。它代表一组经过优化的算法，用于大数据集中的快速最近邻搜索以及高维特征。

2.对于大型数据集，它的工作速度比BFMatcher快。

3.需要传递两个字典来指定要使用的算法及其相关参数等

对于SIFT或SURF等算法，可以用以下方法：

index\_params = dict(algorithm = FLANN\_INDEX\_KDTREE, trees = 5)

对于ORB，可以使用以下参数：

index\_params= dict(algorithm = FLANN\_INDEX\_LSH,

table\_number = 6, # 12 这个参数是searchParam,指定了索引中的树应该递归遍历的次数。值越高精度越高

key\_size = 12, # 20

multi\_probe\_level = 1) #2

'''

import cv2 as cv

from matplotlib import pyplot as plt

queryImage=cv.imread("template\_adjust.jpg",0)

trainingImage=cv.imread("target.jpg",0)#读取要匹配的灰度照片

sift=cv.xfeatures2d.SIFT\_create()#创建sift检测器

kp1, des1 = sift.detectAndCompute(queryImage,None)

kp2, des2 = sift.detectAndCompute(trainingImage,None)

#设置Flannde参数

FLANN\_INDEX\_KDTREE=0

indexParams=dict(algorithm=FLANN\_INDEX\_KDTREE,trees=5)

searchParams= dict(checks=50)

flann=cv.FlannBasedMatcher(indexParams,searchParams)

matches=flann.knnMatch(des1,des2,k=2)

#设置好初始匹配值

matchesMask=[[0,0] for i in range (len(matches))]

for i, (m,n) in enumerate(matches):

if m.distance< 0.5\*n.distance: #舍弃小于0.5的匹配结果

matchesMask[i]=[1,0]

drawParams=dict(matchColor=(0,0,255),singlePointColor=(255,0,0),matchesMask=matchesMask,flags=0) #给特征点和匹配的线定义颜色

resultimage=cv.drawMatchesKnn(queryImage,kp1,trainingImage,kp2,matches,None,\*\*drawParams) #画出匹配的结果

plt.imshow(resultimage,),plt.show()

运行结果：  


**基于FLANN的匹配器(FLANN based Matcher)定位图片**

#基于FLANN的匹配器(FLANN based Matcher)定位图片

import numpy as np

import cv2

from matplotlib import pyplot as plt

MIN\_MATCH\_COUNT = 10 #设置最低特征点匹配数量为10

template = cv2.imread('template\_adjust.jpg',0) # queryImage

target = cv2.imread('target.jpg',0) # trainImage

# Initiate SIFT detector创建sift检测器

sift = cv2.xfeatures2d.SIFT\_create()

# find the keypoints and descriptors with SIFT

kp1, des1 = sift.detectAndCompute(template,None)

kp2, des2 = sift.detectAndCompute(target,None)

#创建设置FLANN匹配

FLANN\_INDEX\_KDTREE = 0

index\_params = dict(algorithm = FLANN\_INDEX\_KDTREE, trees = 5)

search\_params = dict(checks = 50)

flann = cv2.FlannBasedMatcher(index\_params, search\_params)

matches = flann.knnMatch(des1,des2,k=2)

# store all the good matches as per Lowe's ratio test.

good = []

#舍弃大于0.7的匹配

for m,n in matches:

if m.distance < 0.7\*n.distance:

good.append(m)

if len(good)>MIN\_MATCH\_COUNT:

# 获取关键点的坐标

src\_pts = np.float32([ kp1[m.queryIdx].pt for m in good ]).reshape(-1,1,2)

dst\_pts = np.float32([ kp2[m.trainIdx].pt for m in good ]).reshape(-1,1,2)

#计算变换矩阵和MASK

M, mask = cv2.findHomography(src\_pts, dst\_pts, cv2.RANSAC, 5.0)

matchesMask = mask.ravel().tolist()

h,w = template.shape

# 使用得到的变换矩阵对原图像的四个角进行变换，获得在目标图像上对应的坐标

pts = np.float32([ [0,0],[0,h-1],[w-1,h-1],[w-1,0] ]).reshape(-1,1,2)

dst = cv2.perspectiveTransform(pts,M)

cv2.polylines(target,[np.int32(dst)],True,0,2, cv2.LINE\_AA)

else:

print( "Not enough matches are found - %d/%d" % (len(good),MIN\_MATCH\_COUNT))

matchesMask = None

draw\_params = dict(matchColor=(0,255,0),

singlePointColor=None,

matchesMask=matchesMask,

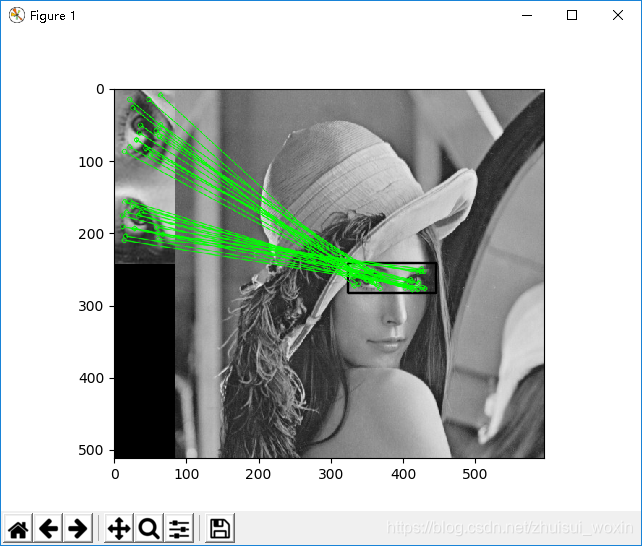
flags=2)

result = cv2.drawMatches(template,kp1,target,kp2,good,None,\*\*draw\_params)

plt.imshow(result, 'gray')

plt.show()

https://csdnimg.cn/release/blogv2/dist/pc/img/newCodeMoreWhite.png

运行结果：  


**参考资料**

1.[opencv英文版文档大全](https://docs.opencv.org/)  
2.[opencv中文网、论坛](http://www.opencv.org.cn/)  
3.[opencv的版本不对可能出现的一个错误](https://stackoverflow.com/questions/52305578/sift-cv2-xfeatures2d-sift-create-not-working-even-though-have-contrib-instal/52309540)

1. [模板匹配英文版教程–opencv–version:3.4.2](https://docs.opencv.org/3.4.2/de/da9/tutorial_template_matching.html)  
   [模板匹配中文版教程–opencv–version:2.3.2](http://www.opencv.org.cn/opencvdoc/2.3.2/html/doc/tutorials/imgproc/histograms/template_matching/template_matching.html#template-matching) [↩︎](https://blog.csdn.net/zhuisui_woxin/article/details/84400439#fnref1)
2. [FLANN特征点匹配英文版教程–opencv–version:3.4.2](https://docs.opencv.org/3.4.2/d7/dff/tutorial_feature_homography.html)  
   [FLANN特征点匹配中文版教程–opencv–version:2.3.2](http://www.opencv.org.cn/opencvdoc/2.3.2/html/doc/tutorials/features2d/feature_homography/feature_homography.html#feature-homography) [↩︎](https://blog.csdn.net/zhuisui_woxin/article/details/84400439#fnref2)