工具：Jupyter，内核为Python3.12，opencv版本：4.8.1

题目：使用openCV读取一张图片作为图片1。结合1，将图片1进行处理得到图片2。结合2，使用FLANN（快速最近邻搜索）匹配器，将图片1与图片2，通过knnMatch函数进行特征匹配

像往常一样，我们先导入我们的opencv库和Numpy函数库，并且导入图片

1. **import** cv2
2. **import** numpy as np
3. img=cv2.imread("D:/111111.jpg")

这里定义的ImageProcessing函数实际上就是我们在图片操作中定义的三个函数的结合体，这里直接上代码，不多赘述（这里return的图片作为题目中的图片2）

1. **def** ImageProcessing(img):
2. ret1=cv2.resize(img,None,fx=0.75,fy=0.75)
3. h,w=ret1.shape[:2]
4. M=cv2.getRotationMatrix2D((h/2,w/2),135,1)
5. ret2=cv2.warpAffine(ret1,M,(h,w))
6. M=np.float32([[1,0,-0.75\*w],[0,1,-0.8\*h]])
7. ret3=cv2.warpAffine(ret2,M,(h,w))
8. **return** ret3

同样的CyShow函数也直接贴代码，不多赘述

1. **def** CvShow(name,img):
2. cv2.imshow(name,img)
3. cv2.waitKey(0)
4. cv2.destroyAllWindows()

最后一个函数是我们用来特征匹配的函数，我们把它叫做FLANNmatch（注意两个参数）

1. **def** FLANNmatch(img1,img2):

这部分与特征匹配的函数是一样的，就不多赘述了

1. gray1=cv2.cvtColor(img1,cv2.COLOR\_BGR2GRAY)
2. gray2=cv2.cvtColor(img2,cv2.COLOR\_BGR2GRAY)
3. sift=cv2.xfeatures2d.SIFT\_create(）
4. kp1,des1=sift.detectAndCompute(gray1,None)
5. kp2,des2=sift.detectAndCompute(gray2,None)

在这里我们指定FLANN匹配器的算法

1. FLANN\_INDEX\_KDTREE=1

在创建实例前，我们需要准备两个字典作为参数，所以接着我们定义一个字典，指定使用KDTREE算法，并且建立5棵树

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

然后我们定义另外一个字典，指定检查50次

1. search\_params=dict(checks=50)

接着我们就可以用两个字典通过FlannBasedMatcher函数来创建实例

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

我们把之前获取的des1和des2放进knnMatch函数，每个描述子返回最近的两个匹配

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

然后我们创建一个二维数组，来标记每个匹配的状态，初始值为[0,0]

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

紧接着我们走进了一个循环，来遍历所有匹配，并通过阈值0.7来筛选精确的匹配

1. **for** i,(m,n) **in** enumerate(matches):
2. **if** m.distance < 0.7\*n.distance:
3. matchesMask[i]=[1,0]

在这里我们建立了一个字典，匹配关键点颜色为绿色，单独的关键点颜色为红色，然后放入我们之前获得的matchesMask，flags设为默认值

1. draw\_params=dict(matchColor=(0,255,0),singlePointColor=(0,0,255),matchesMask=matchesMask,flags=0)

最后，我们只需将，图像、关键点、关键点的匹配、字典放进去，画出图像即可

1. img3=cv2.drawMatchesKnn(img1,kp1,img2,kp2,matches,None,\*\*draw\_params)
2. **return** img3

完整的函数为

1. **def** FLANNmatch(img1,img2):
2. gray1=cv2.cvtColor(img1,cv2.COLOR\_BGR2GRAY)
3. gray2=cv2.cvtColor(img2,cv2.COLOR\_BGR2GRAY)
4. sift=cv2.xfeatures2d.SIFT\_create()
5. kp1,des1=sift.detectAndCompute(gray1,None)
6. kp2,des2=sift.detectAndCompute(gray2,None)
7. FLANN\_INDEX\_KDTREE=1
8. index\_params=dict(algorithm=FLANN\_INDEX\_KDTREE,trees=5)
9. search\_params=dict(checks=50)
10. flann=cv2.FlannBasedMatcher(index\_params,search\_params)
11. matches=flann.knnMatch(des1,des2,k=2)
12. matchesMask=[[0,0] **for** i **in** range(len(matches))]
13. **for** i,(m,n) **in** enumerate(matches):
14. **if** m.distance < 0.7\*n.distance:
15. matchesMask[i]=[1,0]
16. draw\_params=dict(matchColor=(0,255,0),singlePointColor=(0,0,255),matchesMask=matchesMask,flags=0)
17. img3=cv2.drawMatchesKnn(img1,kp1,img2,kp2,matches,None,\*\*draw\_params)
18. **return** img3

最后的最后，我们只需代入即可

1. CvShow("img",FLANNmatch(ImageProcessing(img),img))

完整的代码为

1. **import** cv2
2. **import** numpy as np
3. img=cv2.imread("D:/111111.jpg")
4. **def** ImageProcessing(img):
5. ret1=cv2.resize(img,None,fx=0.75,fy=0.75)
6. h,w=ret1.shape[:2]
7. M=cv2.getRotationMatrix2D((h/2,w/2),135,1)
8. ret2=cv2.warpAffine(ret1,M,(h,w))
9. M=np.float32([[1,0,-0.75\*w],[0,1,-0.8\*h]])
10. ret3=cv2.warpAffine(ret2,M,(h,w))
11. **return** ret3
12. **def** CvShow(name,img):
13. cv2.imshow(name,img)
14. cv2.waitKey(0)
15. cv2.destroyAllWindows()
16. **def** FLANNmatch(img1,img2):
17. gray1=cv2.cvtColor(img1,cv2.COLOR\_BGR2GRAY)
18. gray2=cv2.cvtColor(img2,cv2.COLOR\_BGR2GRAY)
19. sift=cv2.xfeatures2d.SIFT\_create()
20. kp1,des1=sift.detectAndCompute(gray1,None)
21. kp2,des2=sift.detectAndCompute(gray2,None)
22. FLANN\_INDEX\_KDTREE=1
23. index\_params=dict(algorithm=FLANN\_INDEX\_KDTREE,trees=5)
24. search\_params=dict(checks=50)
25. flann=cv2.FlannBasedMatcher(index\_params,search\_params)
26. matches=flann.knnMatch(des1,des2,k=2)
27. matchesMask=[[0,0] **for** i **in** range(len(matches))]
28. **for** i,(m,n) **in** enumerate(matches):
29. **if** m.distance < 0.7\*n.distance:
30. matchesMask[i]=[1,0]
31. draw\_params=dict(matchColor=(0,255,0),singlePointColor=(0,0,255),matchesMask=matchesMask,flags=0)
32. img3=cv2.drawMatchesKnn(img1,kp1,img2,kp2,matches,None,\*\*draw\_params)
33. **return** img3
34. CvShow("img",FLANNmatch(ImageProcessing(img),img))