小马机器人有限公司

# 1.表计识别中的opencv算法

## 1．1.cv2.FlannBasedMatcher(dict(algorithm=1, tree=5), dict(checks=10))

在上篇文章中介绍了暴力匹配BFMatcher，相对暴力匹配BFMatcher来讲，FLANN匹配算法比较准确、快速和使用方便。FLANN具有一种内部机制，可以根据数据本身选择最合适的算法来处理数据集。值得注意的是，FLANN匹配器只能使用SURF和SIFT算法来检测角点。

### 1.1 FLANN介绍

FLANN (Fast\_Library\_for\_Approximate\_Nearest\_Neighbors)快速最近邻搜索包。

它是一个对大数据集和高维特征进行最近邻搜索的算法的集合,而且这些算法都已经被优化过了。在面对大数据集时它的效果要好于 BFMatcher。

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

searchparams = dict(checks = 100)

指定递归遍历的次数checks 。值越高结果越准确, 但是消耗的时间也越多。

首先我们定义一个需要查询的图像，在其中找到一些特征点(关键点)，然后再在另一幅图像中也找到了一些特征点，最后对这两幅图像之间的特征点进行匹配。

#### 1.1.1 2.2 FLANN的单应性匹配

单应性是一个条件，该条件表面当两幅图像中的一副出像投影畸变时，两幅图像还可以进行匹配。

bf = cv2.FlannBasedMatcher(dict(algorithm=1, tree=5), dict(checks=10))  
if cv2.\_\_version\_\_ == '4.4.0':  
 # 调用SIFT算法进行图像匹配  
 sift = cv2.SIFT\_create()  
else:  
 sift = cv2.xfeatures2d.SIFT\_create()  
# sift匹配算法找到特征点  
kp1, des1 = sift.detectAndCompute(template\_image, None)  
kp2, des2 = sift.detectAndCompute(current\_image, None)  
# k最近邻匹配  
matches = bf.knnMatch(des1, des2, k=2)  
# print('特征点坐标：',matches)  
# print('匹配到的特征点数量：',len(matches))  
# 根据matches生成相同长度的matchesMask列表，列表元素为[0,0]  
matchesMask = [[0, 0] for i in range(len(matches))]  
for i, (m1, m2) in enumerate(matches):  
 if m1.distance < 0.7 \* m2.distance:  
 matchesMask[i] = [1, 0]  
 pt1 = kp1[m1.queryIdx].pt  
 pt2 = kp2[m1.trainIdx].pt  
good = []  
# 找到匹配的符合的特征点  
for m, n in matches:  
 if m.distance < 0.7 \* n.distance:  
 good.append(m)  
# 单应性操作：图像发生畸变的情况也能匹配  
if len(good) > 10:  
 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)  
 # findHomography 函数是计算变换矩阵  
 # 参数cv2.RANSAC是使用RANSAC算法寻找一个最佳单应性矩阵H，即返回值M  
 # 返回值：M 为变换矩阵，mask是掩模  
 M, mask = cv2.findHomography(src\_pts, dst\_pts, cv2.RANSAC, 5.0)  
else:  
 M = None  
return M