**目标：**

* 我们将要学习在图像间进行特征匹配
* 使用 OpenCV 中的蛮力（Brute-Force）匹配和 FLANN 匹配

## api：

### # Initiate ORB detector

### orb = cv.ORB\_create()

### # find the keypoints and descriptors with ORB

### kp1, des1 = orb.detectAndCompute(img1,None)

### bf = cv.BFMatcher(cv.NORM\_HAMMING, crossCheck=True)

### # Match descriptors.

### matches = bf.match(des1,des2)

### # Sort them in the order of their distance.

### matches = sorted(matches, key = lambda x:x.distance)

### cv2.drawMatches(img1,kp1,img2,kp2,matches[:10], flags=2)

### cv.FlannBasedMatcher(index\_params,search\_params)

### flann = cv.FlannBasedMatcher(index\_params,search\_params)

**蛮力（Brute-Force）匹配基础**

蛮力匹配器很简单。首先在第一幅图像中选取一个关键点然后依次与第二幅图像的每个关键点进行（描述符）距离测试，最后返回距离最近的关键点。

对于BF匹配器，首先我们必须使用cv.BFMatcher()创建一个BFMatcher对象。它需要两个可选的参数，第一个是normType，它指定要使用的距离测量，默认情况下，它是cv.NORM\_L2，它适用于SIFT，SURF等（cv.NORM\_L1也在那里）。对于基于二进制字符串的描述符，如ORB，BRIEF，BRISK等，应使用cv.NORM\_HAMMING，它使用汉明距离作为度量。如果ORB使用WTA\_K==3或4，则应使用cv.NORM\_HAMMING2。

第二个参数是布尔变量crossCheck，默认为false。如果为真，则Matcher仅返回具有值(i，j)的那些匹配，使得集合A中的第i个描述符具有集合B中的第j个描述符作为最佳匹配，反之亦然。也就是说，两组中的两个特征应该相互匹配。它提供了一致的结果，是D.Lowe在SIFT论文中提出的比率测试的一个很好的替代方案。

一旦创建，两个重要的方法是BFMatcher.match()和BFMatcher.knnMatch()。第一个返回最佳匹配。第二种方法返回k个最佳匹配，其中k由用户指定。当我们需要做更多的工作时，它可能是有用的。

就像我们使用cv.drawKeypoints()来绘制关键点一样，cv.drawMatches()帮助我们绘制匹配项。它水平堆叠两个图像，并从第一个图像到第二个图像绘制线条，显示最佳匹配。还有cv.drawMatchesKnn，它绘制了所有k个最佳匹配。如果k = 2，它将为每个关键点绘制两条匹配线。因此，如果我们想要有选择地绘制它，我们必须传递一个掩码。

让我们看一下每个SURF和ORB的一个例子（两者都使用不同的距离测量）。

**与ORB描述符的强力匹配**

在这里，我们将看到一个关于如何匹配两个图像之间的特征的简单示例。在这种情况下，我有一个查询图像和一个目标图像。我们将尝试使用特征匹配在目标图像中查找查询图像。（图片为/samples/c/box.png和/samples/c/box\_in\_scene.png）

我们使用ORB描述符来匹配功能。所以让我们从加载图像，查找描述符等开始。

接下来，我们使用距离测量cv.NORM\_HAMMING创建一个BFMatcher对象（因为我们使用的是ORB）,并且启用了crossCheck以获得更好的结果。然后我们使用Matcher.match()方法在两个图像中获得最佳匹配。我们按照距离的升序对它们进行排序，以便最佳匹配（低距离）出现在前面。然后我们只绘制前10场比赛（太多了看不清，如果愿意的话你可以多画几条）

## 实例

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

### 这个Matcher对象是什么？

matches = bf.match(des1，des2)行的结果是DMatch对象的列表。此DMatch对象具有以下属性：

* DMatch.distance - 描述符之间的距离。越低越好。
* DMatch.trainIdx - 列车描述符中描述符的索引
* DMatch.queryIdx - 查询描述符中描述符的索引
* DMatch.imgIdx - 火车图像的索引。

### 对 SIFT 描述符进行蛮力匹配和比值测试

这一次，我们将使用BFMatcher.knnMatch()来获得最佳匹配。在这个例子中，我们将采用k = 2，以便我们可以在他的论文中应用D.Lowe解释的比率测试。

## 实例2

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

## FLANN匹配

FLANN 是快速最近邻搜索包（Fast\_Library\_for\_Approximate\_Nearest\_Neighbors）的简称。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。我们将看到基于FLANN的匹配器的第二个示例。

对于基于FLANN的匹配器，我们需要传递两个字典，指定要使用的算法和其他相关参数等。首先是IndexParams。对于各种算法，要传递的信息在FLANN文档中进行了解。总而言之，对于像SIFT，SURF等算法，你可以传递以下内容：

FLANN\_INDEX\_KDTREE = 1

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

但使用 ORB 时，我们要传入的参数如下。注释掉的值是文献中推荐使用的，但是它们并不适合所有情况，其他值的效果可能会更好。

FLANN\_INDEX\_LSH = 6

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

table\_number = 6, # 12

key\_size = 12, # 20

multi\_probe\_level = 1) #2

第二个字典是SearchParams。它指定应递归遍历索引中的树的次数。值越高，精度越高，但也需要更多时间。如果要更改该值，请传递search\_params = dict(checks = 100)。

有了这些信息，我们就可以开始工作了。

## 实例3

|  |  |
| --- | --- |
|  | **import cv2 import numpy as np  img1 = cv2.imread('../pic\_datas/box.png',0) *# 填写图片名称* img2 = cv2.imread('../pic\_datas/box\_in\_scene.png',0) *# 填写图片名称  # Initiate SIFT detector* sift = cv2.xfeatures2d.SIFT\_create() *# find the keypoints and descriptors with ORB* kp1,des1 = sift.detectAndCompute(img1,None) kp2,des2 = sift.detectAndCompute(img2,None) *# FLANN parameters* FLANN\_INDEX\_KDTREE = 1 index\_params = dict(algorithm=FLANN\_INDEX\_KDTREE,trees=5) search\_params = dict(checks=50) *# 也可以用空字典dict()* flann = cv2.FlannBasedMatcher(index\_params,search\_params) *# Match descriptors.* matches = flann.knnMatch(des1,des2,k=2) *# Need to draw only good matches, so create a mask* matchesMask = [[0,0] for i in range(len(matches))] for i,(m,n) in enumerate(matches):  if m.distance < 0.7\*n.distance :  matchesMask[i] = [1, 0] *# 定义应该绘制参数字典* draw\_params = dict(  matchColor=(0,255,0),  singlePointColor=(255,0,0),  matchesMask=matchesMask,  flags=0 ) img3 = cv2.drawMatchesKnn(img1,kp1,img2,kp2,matches,None,\*\*draw\_params) cv2.imshow('Result',img3) cv2.waitKey(0) cv2.destroyAllWindows()** |

### 效果

|  |
| --- |
|  |