姓名学号

- 张进华 201900150221

实验日期

- 2021.11.12

实验题目

- 图像匹配2
    - 特征检测与匹配

实验内容

- 测试OpenCV中的SIFT, SURF, ORB等特征检测与匹配的方法。将检测到的特征点和匹配关系进行可视化输出，比较不同方法的效率、效果等。

实验步骤

步骤一 了解实验原理

sift特征简介

- SIFT(Scale-Invariant Feature Transform)特征，即尺度不变特征变换,在不同的尺度空间上查找关键点(特征点)，并计算出关键点的方向。

surf特征简介

- SURF(Speeded Up Robust Features, 加速稳健特征) 是是SIFT的高效变种，算法步骤与SIFT算法大致相同，但采用的方法不一样，要比SIFT算法更高效。SURF使用海森(Hesseian)矩阵的行列式值作特征点检测并用积分图加速运算

orb特征简介

- ORB（Oriented FAST and Rotated BRIEF）该特征检测算法是在著名的FAST特征检测和BRIEF特征描述子的基础上提出来的，其运行时间远远优于SIFT和SURF，可应用于实时性特征检测。ORB特征检测具有尺度和旋转不变性，对于噪声及其透视变换也具有不变性，良好的性能是的利用ORB在进行特征描述时的应用场景十分广泛。ORB特征检测主要分为以下两个步骤:(1)方向FAST特征点检测(2)BRIEF特征描述。

步骤二 代码实现

- 本次实验的代码采用的库如下图所示，版本使用3.4.2.16的，而不是最新的，否则在特征提取的时候会报错

Package	Version	Latest version
numpy	1.17.0	1.17.0
opencv-contrib-python	3.4.2.16	▲ 4.1.0.25
opencv-python	3.4.2.16	▲ 4.1.0.25
pip	19.2.1	19.2.1
setuptools	39.1.0	▲ 41.0.1

特征提取

- **1.sift**

```
def sift(filename):
    img = cv2.imread(filename) # 读取文件
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 转化为灰度图
    sift = cv2.xfeatures2d_SIFT.create()
    keyPoint, descriptor = sift.detectAndCompute(img, None) # 特征提取得到关键点以及对应的描述符（特征向量）
    return img,keyPoint, descriptor
```

- **2.surf**

```
def surf(filename):
    img = cv2.imread(filename) # 读取文件
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 转化为灰度图
    sift = cv2.xfeatures2d_SURF.create()
    keyPoint, descriptor = sift.detectAndCompute(img, None) # 特征提取得到关键点以及对应的描述符（特征向量）
    return img, keyPoint, descriptor
```

- **3.orb**

```
def orb(filename):
    img = cv2.imread(filename) # 读取文件
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 转化为灰度图
    sift = cv2.ORB_create()
    keyPoint, descriptor = sift.detectAndCompute(img, None) # 特征提取得到关键点以及对应的描述符（特征向量）
    return img, keyPoint, descriptor
```

- 这里解释一下为什么要进行转化为灰度图?
    - 识别物体，最关键的因素是梯度（SIFT/HOG），梯度意味着边缘，这是最本质的部分，而计算梯度，自然就用到灰度图像了，可以把灰度理解为图像的强度。颜色，易受光照影响，难以提供关键信息，故将图像进行灰度化，同时也可以加快特征提取的速度。
- 这里提取出的keyPoint返回的是特征点所包含的信息，是一个Dmatch数据类型
    - angle：角度，表示关键点的方向，通过Lowe的论文可以知道，为了保证方向不变形，SIFT算法通过对关键点周围邻域进行梯度运算，求得该点方向。-1为初值。
    - class_id——当要对图片进行分类时，可以用class_id对每个特征点进行区分，未设定时为-1，需要靠自己设定
    - octave——代表是从金字塔哪一层提取的得到的数据。
    - pt——关键点点的坐标
    - response——响应程度，代表该点角点的程度。

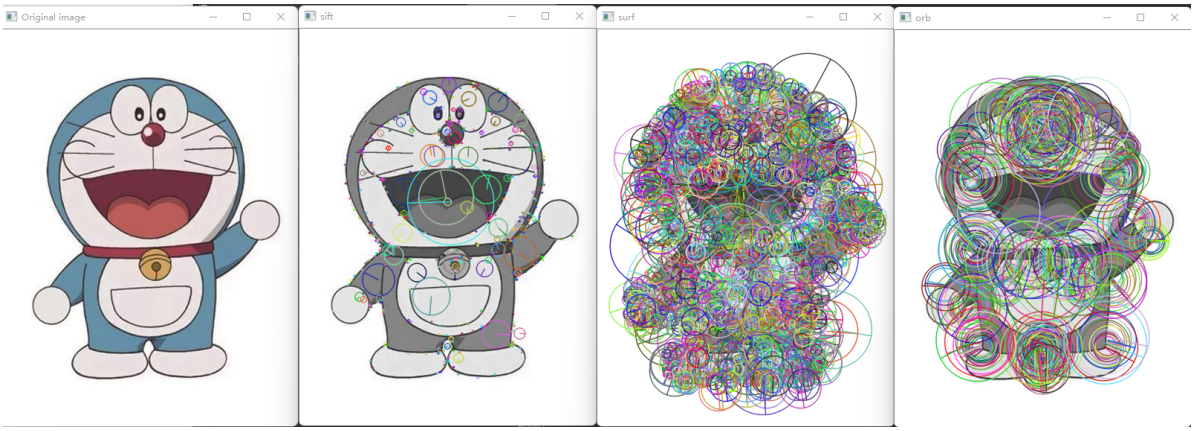
- size——该点直径的大小
- des:返回特征点的特征描述符，是一个二维列表

```
F:\anaconda3\envs\openCV\python.exe F:/computer-vision/Exp8
[<KeyPoint 00000224189165D0>, <KeyPoint 0000022418776600>,
-----------
[[ 1.  1.  1. ...  0.  0.  6.]
 [18.  1.  0. ...  0.  0.  0.]
 [26.  7.  0. ...  0.  0.  0.]
 ...
 [ 0.  0.  0. ...  1.  0.  2.]
 [18.  3.  0. ...  0.  2.  8.]
 [ 0.  0.  0. ...  0.  0.  5.]]
```

- 比较一下提取的结果看看,代码如下，surf和orb都差不多

```
 img, key, des = sift('images/doraemon1.jpg')
    img = cv2.drawKeypoints(img, key, None,
flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
    print("Number of points extracted by sift: %d, size of des: %d" % (len(des),
len(des[0])))
    cv2.imshow("sift", img)
```

- 下图是提取的结果，从左到右分别是原图、sift、surf、orb




```
Number of points extracted by sift: 458, size of des: 128
Number of points extracted by surf: 1783, size of des: 64
Number of points extracted by orb: 500, size of des: 32
```

可以看出:

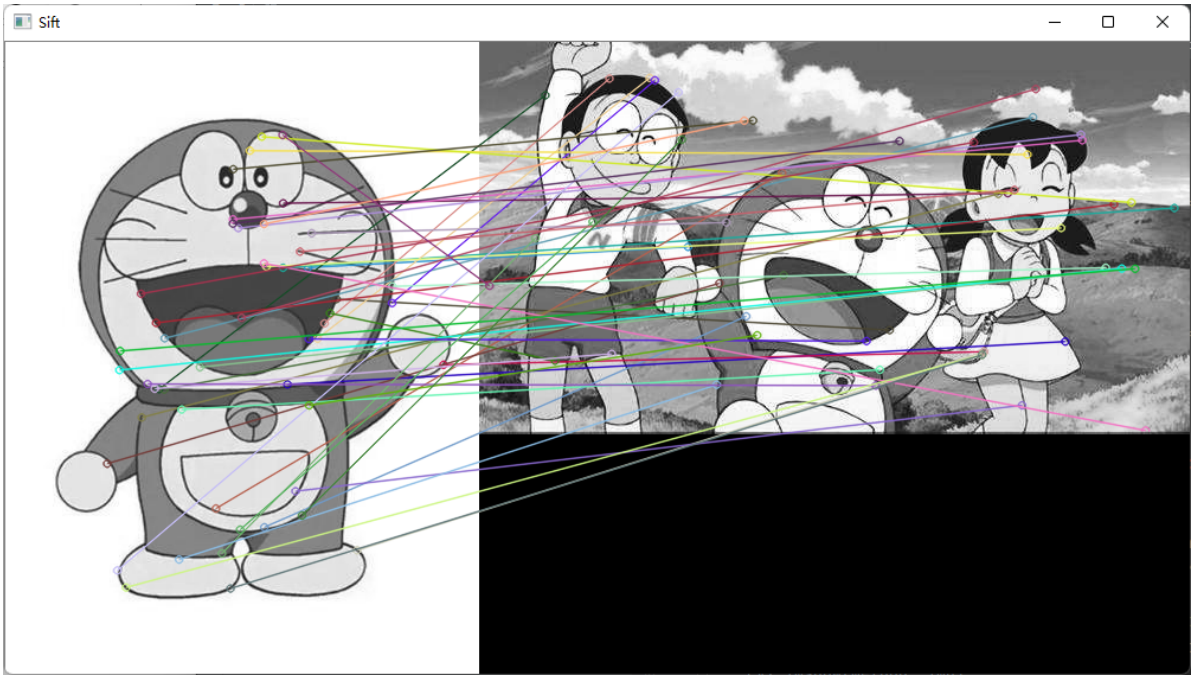
- sift虽然提取的特征点最少，但是效果最好。
- sift提取的特征点维度是128维，surf是64维，orb是32维。

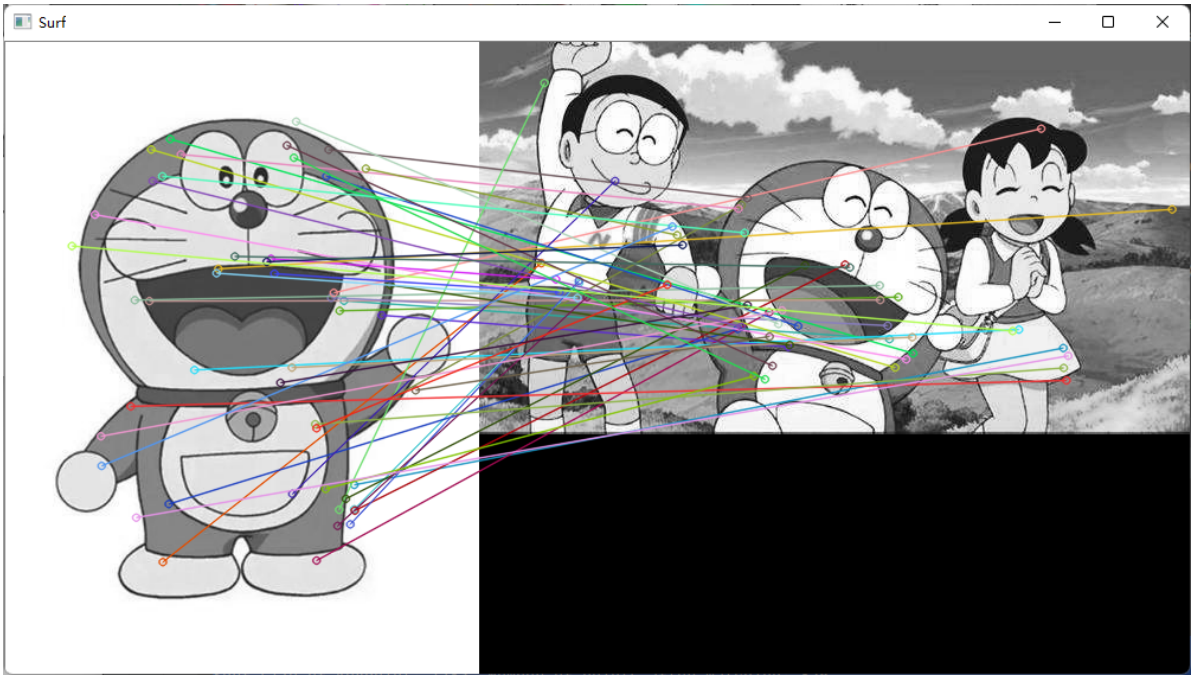
特征匹配

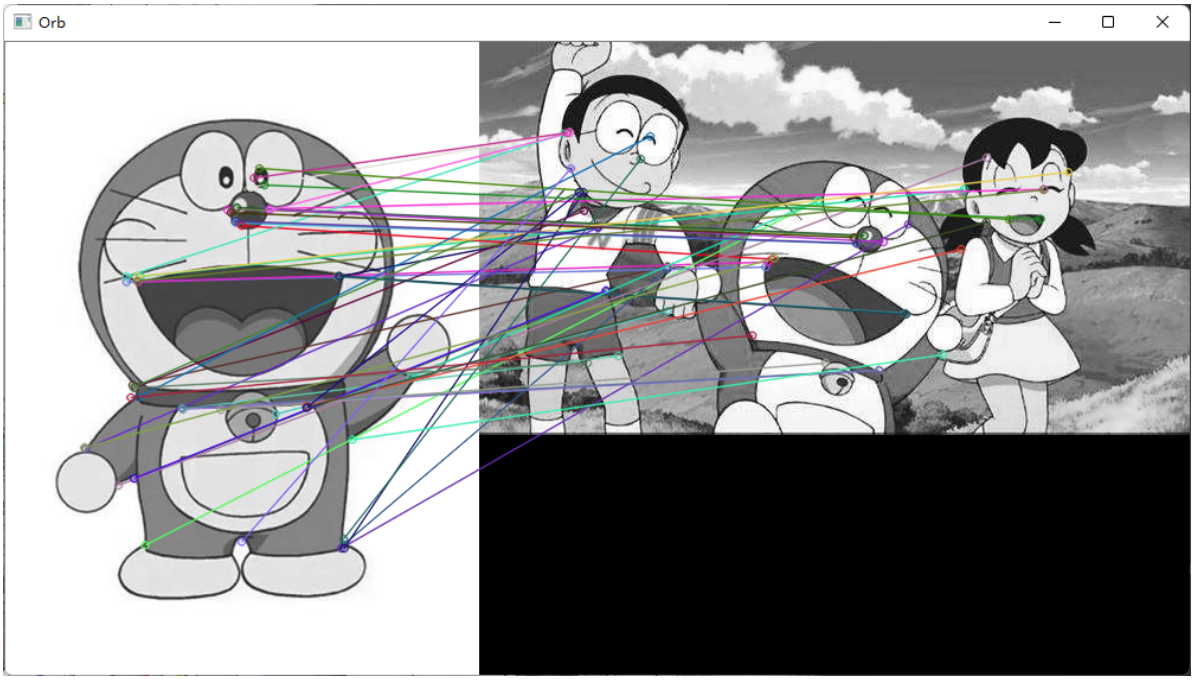
- BruteForce匹配总是尝试所有可能的匹配，从而使得它总能够找到最佳匹配，实验是进行最优特征点匹配，因此选用BruteForce Matcher。
- orb方法与其他两种的距离测量方法不同，SIFT 和 SURF使用L2距离 cv.NORM_L2， ORB使用汉明距离 cv.NORM_HAMMING

```
    if method == 'Sift':
        img1, kp1, des1 = sift(filename1)
        img2, kp2, des2 = sift(filename2)
        bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)  # sift的normType应该使用NORM_L2或者NORM_L1

    if method == 'Surf':
        img1, kp1, des1 = surf(filename1)
        img2, kp2, des2 = surf(filename2)
        bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)  # surf的normType应该使用NORM_L2或者NORM_L1

    if method == 'Orb':
        img1, kp1, des1 = orb(filename1)
        img2, kp2, des2 = orb(filename2)
        bf = cv2.BFMatcher(cv2.NORM_HAMMING, crossCheck=True)  # orb的normType应该使用NORM_HAMMING

    matches = bf.match(des1, des2)
    matches = sorted(matches, key=lambda x: x.distance)
    knnMatches = bf.knnMatch(des1, des2, k=1)  # drawMatchesKnn

    print('%s size of keyPoint: %d, Number of points after matching: %d' %(method, len(des1), len(matches)))
    img = cv2.drawMatches(img1, kp1, img2, kp2, matches[:50], img2, flags=2)
    cv2.imshow(method, img)
```

- 接下来看一下上面的代码运行的结果。从上到底依次是原图、sift、surf、orb








```
Sift size of keyPoint: 458, Number of points after matching: 319
Surf size of keyPoint: 1783, Number of points after matching: 825
Orb size of keyPoint: 500, Number of points after matching: 249
```

- 从输出的结果来看，orb的效果最好。

实验分析

SIFT所查找到的关键点是一些十分突出、不会因光照、仿射变换和噪音等因素而变化的点，如角点、边缘点、暗区的亮点及亮区的暗点等。
基于特征的匹配分为特征点提取和匹配两个步骤，实验主要针对特征点提取三种方法进行比较，分别是SIFT，SURF以及ORB三种方法，这三种方法在OpenCV里面都已实现。SURF基本就是SIFT的全面升级版，有 SURF基本就不用考虑SIFT，而ORB的强点在于计算时间，所以结论就是，如果对计算实时性要求非常高，可选用ORB算法，但基本要保证正对拍摄；如果对实行性要求稍高，可以选择SURF；基本不用SIFT。