姓名学号

- 张文浩 20190061140

实验日期

- 2021.11.12

实验题目

- 特征检测与匹配

实验要求

- 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。

实验原理

sift

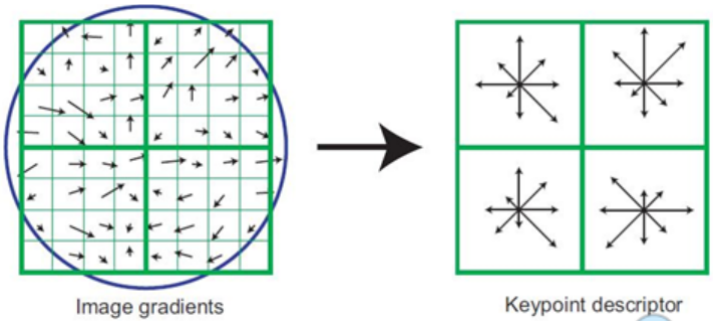
sift特征简介

SIFT(Scale-Invariant Feature Transform)特征，即尺度不变特征变换，是一种计算机视觉的特征提取算法，用来侦测与描述图像中的局部性特征。

实质上，它是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。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。

sift特征提取步骤

1. **尺度空间的极值检测：** 尺度空间指一个变化尺度(σ)的二维高斯函数$G(x,y,\sigma)$与原图像$I(x,y)$卷积（即高斯模糊）后形成的空间,尺度不变特征应该既是空间域上又是尺度域上的局部极值。极值检测的大致原理是根据不同尺度下的高斯模糊化图像差异（Difference of Gaussians,DoG）寻找局部极值，这些找到的极值所对应的点被称为关键点或特征点。
2. **键点定位：** 在不同尺寸空间下可能找出过多的关键点，有些关键点可能相对不易辨识或易受噪声干扰。该步借由关键点附近像素的信息、关键点的尺寸、关键点的主曲率来定位各个关键点，借此消除位于边上或是易受噪声干扰的关键点。
3. **方向定位：** 为了使描述符具有旋转不变性，需要利用图像的局部特征为给每一个关键点分配一个基准方向。通过计算关键点局部邻域的方向直方图，寻找直方图中最大值的方向作为关键点的主方向。
4. **关键点描述子：** 找到关键点的位置、尺寸并赋予关键点方向后，将可确保其移动、缩放、旋转的不变性。此外还需要为关键点建立一个描述子向量，使其在不同光线与视角下皆能保持其不变性。SIFT描述子是关键点邻域高斯图像梯度统计结果的一种表示，见下图。通过对关键点周围图像区域分块，计算块内梯度直方图，生成具有独特性的向量，这个向量是该区域图像信息的一种抽象，具有唯一性。Lowe在原论文中建议描述子使用在关键点尺度空间内4X4的窗口中计算的8个方向的梯度信息，共4X4X8=128维向量表征。(opencv中实现的也是128维)



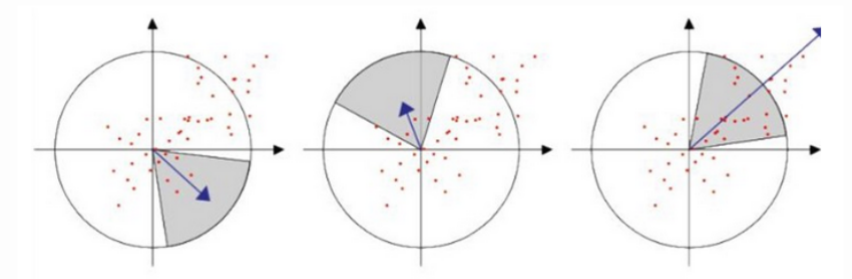
surf

surf特征简介

SURF(Speeded Up Robust Features, 加速稳健特征) 是一种稳健的图像识别和描述算法。它是SIFT的高效变种，也是提取尺度不变特征，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（正如其名）。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；SURF 的描述子基于 2D 离散小波变换响应并且有效地利用了积分图。

surf特征提取提取步骤

1. **特征点检测：** SURF使用Hessian矩阵来检测特征点，该矩阵是x,y方向的二阶导数矩阵，可测量一个函数的局部曲率，其行列式值代表像素点周围的变化量，特征点需取行列式值的极值点。用方型滤波器取代SIFT中的高斯滤波器，利用积分图（计算位于滤波器方型的四个角落值）大幅提高运算速度。
2. **特征点定位：** 与SIFT类似，通过特征点邻近信息插补来定位特征点。
3. **方向定位：** 通过计算特征点周围像素点x,y方向的哈尔小波变换，并将x,y方向的变换值在xy平面某一角度区间内相加组成一个向量，在所有的向量当中最长的(即x、y分量最大的)即为此特征点的方向。即在特征点的圆形邻域内，统计60度扇形内所有点的水平、垂直Harr小波特征总和，然后扇形以0.2弧度大小的间隔进行旋转并再次统计该区域内Harr小波特征值之后，最后将值最大的那个扇形的方向作为该特征点的主方向。该过程示意图如下：



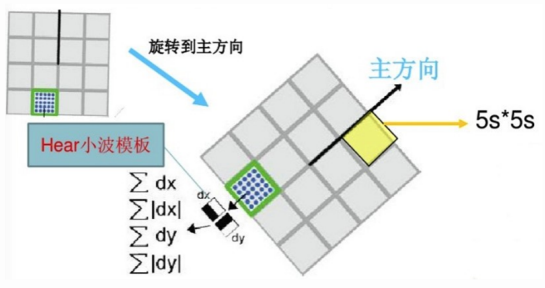
4. **特征描述子：** 选定了特征点的方向后，其周围相素点需要以此方向为基准来建立描述子。此时以5X5个像素点为一个子区域，取特征点周围20*20个像素点的范围共16个子区域，计算子区域内的x、y方向(此时以平行特征点方向为x、垂直特征点方向为y)的哈尔小波转换总和

$$\Sigma dx、\Sigma dy$$

与其向量长度总和

$$\Sigma|dx|、\Sigma|dy|$$

共四个量值，共可产生一个64维的描述子。



orb

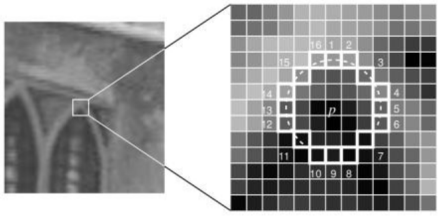
orb特征简介

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，良好的性能使得利用ORB在进行特征描述时的应用场景十分广泛。

orb特征提取算法

- **FAST特征点检测**

1. 从图片中选取一个像素P，下面我们将判断它是否是一个特征点。我们首先把它的亮度值设为Ip
2. 设定一个合适的阈值t。
3. 考虑以该像素点为中心的一个半径等于3像素的离散化的Bresenham圆，这个圆的边界上有16个像素（如图所示）



4. 现在，如果在这个大小为16个像素的圆上有nn个连续的像素点，它们的像素值要么都比Ip+t大，要么都比Ip−t小，那么它就是一个角点。（如图1中的白色虚线所示）。n的值可以设置为12或者9，实验证明选择9可能会有更好的效果。

- **BRIEF特征描述子**

BRIEF提供了一种计算二值串的捷径，而并不需要去计算一个类似于SIFT的特征描述子。它需要先平滑图像，然后在特征点周围选择一个Patch，在这个Patch内通过一种选定的方法来挑选出来nd个点对。然后对于每一个点对(p,q)，我们来比较这两个点的亮度值，如果I(p)>I(q)则这个点对生成了二值串中一个的值为1，如果I(p)<I(q)，则对应在二值串中的值为-1，否则为0。所有nd个点对，都进行比较之间，我们就生成了一个nd长的二进制串。

总体来说，BRIEF是一个效率很高的提取特征描述子的方法，同时，它有着很好的识别率，但当图像发生很大的平面内的旋转。

特征匹配

BruteForce匹配和**FLANN匹配**是opencv二维特征点匹配常见的两种办法，分别对应BFMatcher（BruteForceMatcher）和FlannBasedMatcher。

二者的区别在于BFMatcher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，这也是Brute Force（暴力法）的原始含义。而FlannBasedMatcher中FLANN的含义是Fast Library forApproximate Nearest Neighbors，从字面意思可知它是一种近似法，算法更快但是找到的是最近邻近似匹配，所以当我们需要找到一个相对好的匹配但是不需要最佳匹配的时候往往使用FlannBasedMatcher。当然也可以通过调整FlannBasedMatcher的参数来提高匹配的精度或者提高算法速度，但是相应地算法速度或者算法精度会受到影响。

实验步骤

sift

1. 读取图像，转化为灰度图
2. 新建一个sift算子
3. 特征提取得到关键点及对应的描述符（特征向量）
4. 绘制出特征点（关键点）

```
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()  # 新建一个sift算子
    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

img = cv2.drawKeypoints(img,kp,None)
```

surf

1. 读取图像，转化为灰度图
2. 新建一个surf算子
3. 特征提取得到关键点及对应的描述符（特征向量）
4. 绘制出特征点（关键点）

```
def surf(filename):
    img = cv2.imread(filename) #  读取文件
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 转化为灰度图
    surf = cv2.xfeatures2d_SURF.create() # 新建surf算子
    keyPoint, descriptor = surf.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
```

orb

1. 读取图像，转化为灰度图
2. 新建一个orb算子
3. 特征提取得到关键点及对应的描述符（特征向量）
4. 绘制出特征点（关键点）

```
def orb(filename):
    img = cv2.imread(filename) # 读取文件
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 转化为灰度图
    orb = cv2.ORB_create() # 新建orb算子
    keyPoint, descriptor = orb.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
```

function output

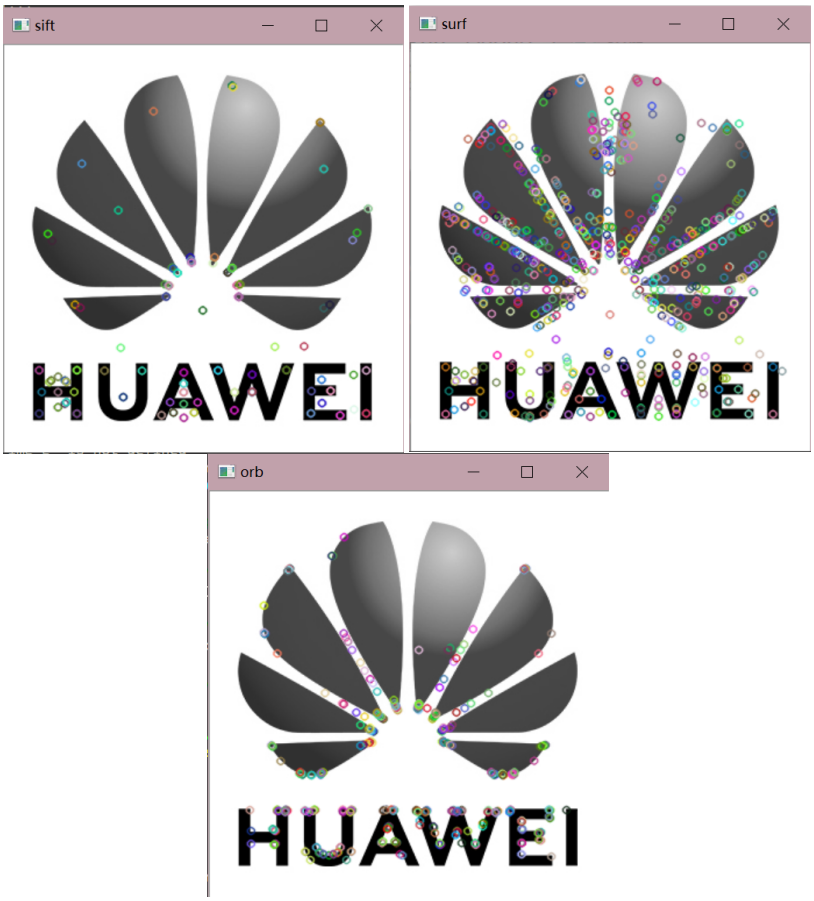
sift、surf、orb函数都返回了三个东西，分别是img、keypoint和descriptor。

他们含义分别是：

- img：标记了关键点的灰度图像
- keypoint：关键点的信息
  - angle：角度，表示关键点的方向，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，我们可以用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，代表该点强壮大小，更确切的说，是该点角点的程度。
  - size：该点直径的大小
- descriptor：表示特征点的特征描述符，是一个二维列表

特征点检测效果对比

shift:　　surf:　　orb:



时间对比

- ```
  sift: 28.91850471496582 ms          找到147个特征点
  surf: 34.93022918701172 ms          找到572个特征点
  orb: 6.47735595703125 ms            找到312个特征点
  ```

match

1. 创建BFMatcher对象
2. 匹配
3. 绘制匹配的线

```
#以sift为例
if(method == 'sift'):
    img1, kp1, des1 = sift(filename1)
    img2, kp2, des2 = sift(filename2)
    bf = cv2.BFMatcher(cv2.NORM_L2, crossCheck=True)  # 新建BFMatcher对象，sift的normType应该使用NORM_L2或者NORM_L1
    matches = bf.match(des1, des2)
    matches = sorted(matches, key=lambda x: x.distance)

img = cv2.drawMatches(img1, kp1, img2, kp2, matches[:50], img2, flags=2)
```

其中

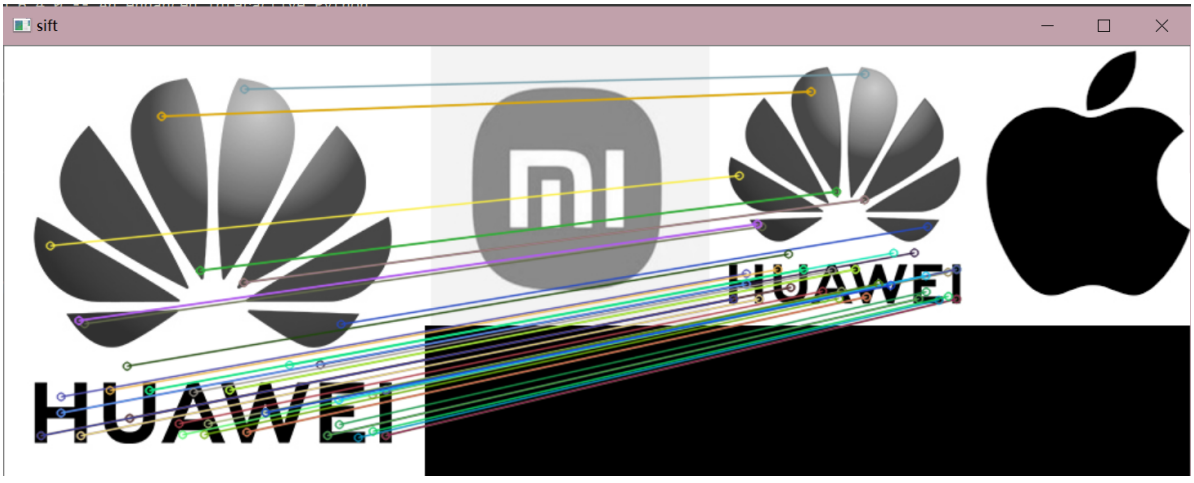
```
matches = bf.match（des1，des2）
```

行的结果是DMatch对象的列表。 此DMatch对象具有以下属性：

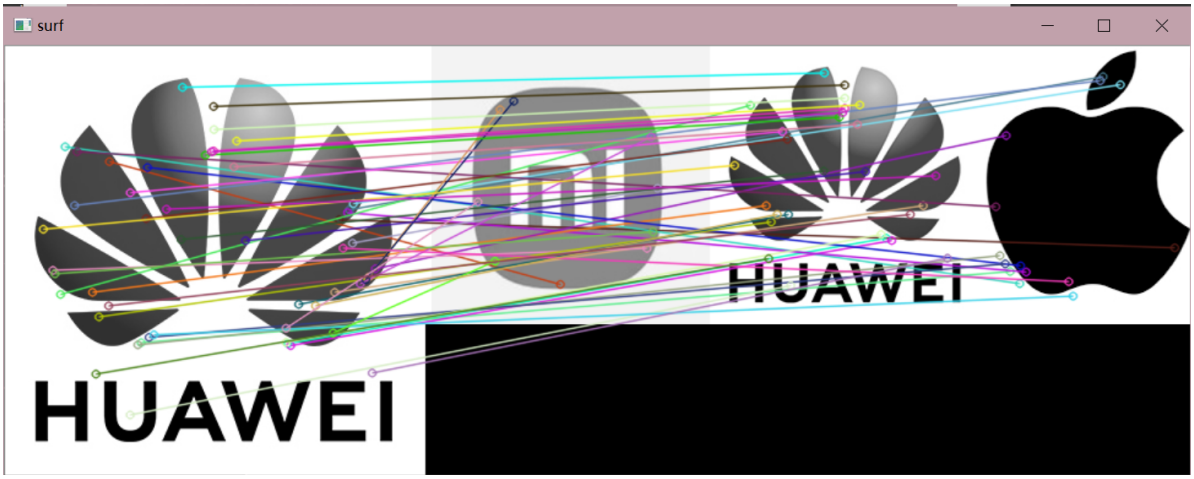
- DMatch.distance - 描述符之间的距离。 越低越好。
- DMatch.trainIdx - 训练描述符中描述符的索引
- DMatch.queryIdx - 查询描述符中描述符的索引
- DMatch.imgIdx - 目标图像的索引

匹配效果:

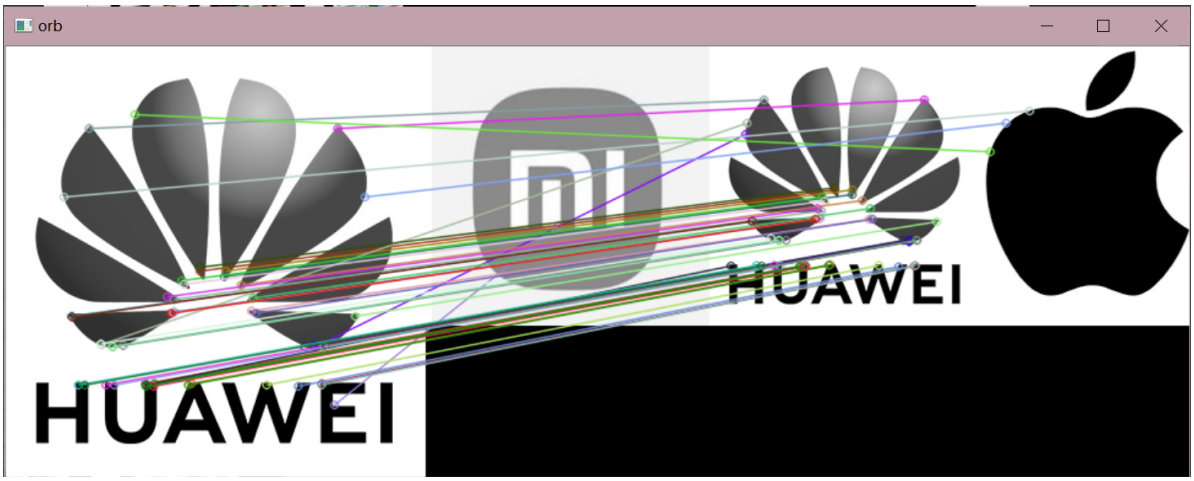
sift共匹配了 108 个点：



surf共匹配了 236 个点：
```



orb共匹配了 201 个点：



实验总结

基于特征的匹配分为特征点提取和匹配两个步骤，本篇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，以下具体比较：

- SIFT是DoG与SIFT的结合。DoG用于关键点的检测，SIFT用于描述关键点的特征（图像描述符）。
- SURF是Hessian与SURF的结合，Hessian用于关键点的检测，SURF用于描述关键点的特征。
- ORB是FAST和BRIEF的结合，FAST用于关键点的检测，BRIEF用于描述关键点的特征。

速度上ORB>SURF>SIFT，SURF的鲁棒性(抗干扰能力）更好一些。

```
计算速度：ORB>>SURF~SIFT（各差一个量级）

旋转鲁棒性：SURF>ORB~SIFT（表示差不多）

模糊鲁棒性：SURF>ORB~SIFT

尺度变换鲁棒性：SURF>SIFT>ORB（ORB并不具备尺度变换性）
```