# 计算机视觉实践实验报告（一）

## 一．实验目的

* 理解关键点检测算法DOG原理。
* 理解尺度变化不变特征SIFT。
* 采集一系列局部图像，自行设计拼接算法。
* 使用Python实现图像拼接算法。

## 二．实验原理

主要通过关键点检测算法DOG，变化不变特征SIFT，将提取到的不同图片的特征点通过尺度空间进行拼接起来

## 2.1 DOG

DoG (Difference of Gaussian)是灰度图像增强和角点检测的方法,Difference of Gaussian(DOG)是高斯函数的差分。我们已经知道可以通过将图像与高斯函数进行卷积得到一幅图像的低通滤波结果，即去噪过程，这里的Gaussian和高斯低通滤波器的高斯一样，是一个函数，即为正态分布函数。

## 2.2 SIFT

SIFT，即尺度不变特征变换，是用于图像处理领域的一种描述。这种描述具有尺度不变性，可在图像中检测出关键点，是一种局部特征描述子。SIFT特征对旋转、尺度缩放、亮度变化等保持不变性，是一种非常稳定的局部特征。

### 2.2.1 SIFT算法特征

* 稳定性：SIFT特征是图像的局部特征，其对旋转、尺度缩放、亮度变化保持不变性，对视角变化、仿射变换、噪声也保持一定程度的稳定性；
* 区分性：信息量丰富，适用于在海量特征数据库中进行快速、准确的匹配；
* 多量性：即使少数的几个物体也可以产生大量的SIFT特征向量；
* 高速性：经优化的SIFT匹配算法甚至可以达到实时的要求；
* 可拓展性：可以很方便的与其他形式的特征向量进行联合。

### 2.2.2 SIFT算法检测步骤

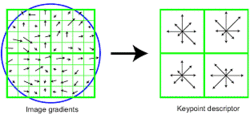
SIFT特征检测主要包括以下4个基本步骤：尺度空间极值检测、 关键点定位、方向确定和关键点描述。

### 2.2.3 SIFT算法特征的生成

即从多幅图像中提取对尺度缩放、旋转、亮度变化无关的特征向量。主要包括：

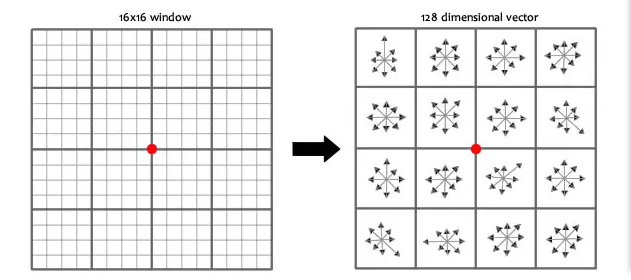
1. 构建尺度空间，检测极值点，获得尺度不变性
2. 特征点过滤并进行精确定位
3. 为特征点分配方向值
4. 生成特征描述子

旋转后以主方向为中心取 8×8的窗口。如下图所示，左图的中央为当前关键点的位置，每个小格代表为关键点邻域所在尺度空间的一个像素，求取每个像素的梯度幅值与梯度方向，箭头方向代表该像素的梯度方向，长度代表梯度幅值，然后利用高斯窗口对其进行加权运算。最后在每4×4的小块上绘制8个方向的梯度直方图，计算每个梯度方向的累加值，即可形成一个种子点，如右图所示。每个特征点由4个种子点组成，每个种子点有8个方向的向量信息。



与求主方向不同，此时每个种子区域的梯度直方图在0-360之间划分为8个方向区间，每个区间为45度，即每个种子点有8个方向的梯度强度信息。

在实际的计算过程中，为了增强匹配的稳健性，Lowe建议对每个关键点使用4×4共16个种子点来描述，这样一个关键点就可以产生128维的SIFT特征向量。



### 2.2.4 特征向量的匹配

当两幅图像的SIFT特征向量生成以后，下一步就可以采用关键点特征向量的欧式距离来作为两幅图像中关键点的相似性判定度量。取图1的某个关键点，通过遍历找到图像2中的距离最近的两个关键点。在这两个关键点中，如果最近距离除以次近距离小于某个阈值，则判定为一对匹配点。

## 2.3尺度空间

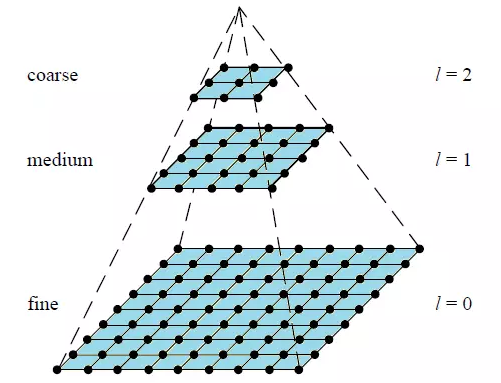
在未知的场景中，计算机视觉并不能提供物体的尺度大小，其中的一种方法是把物体不同尺度下的图像都提供给机器，让机器能够对物体在不同的尺度下有一个统一的认知。在建立统一认知的过程中，要考虑的就是在图像在不同的尺度下都存在的特征点。

### 2.3.1 多分辨率图像金字塔

图像金字塔是同一图像在不同的分辨率下得到的一组结果，其生成过程一般包括两个步骤：

1. 对原始图像进行平滑
2. 对处理后的图像进行降采样（通常是水平、垂直方向的1/2）

降采样后得到一系列不断尺寸缩小的图像。多分辨率的图像金字塔的本质是降采样，图像的局部特征则难以保持，也就是无法保持特征的尺度不变性。

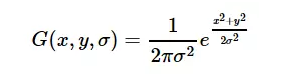


### 2.3.2 高斯尺度空间

我们知道图像和高斯函数进行卷积运算能够对图像进行模糊，使用不同的“高斯核”可得到不同模糊程度的图像。一副图像其高斯尺度空间可由其和不同的高斯卷积得到：

在这里插入图片描述

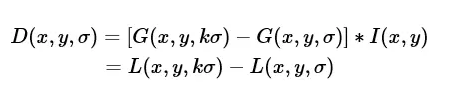
其中，G(x,y,σ)是高斯核函数。



σ称为尺度空间因子，它是高斯正态分布的标准差，反映了图像被模糊的程度，其值越大图像越模糊，对应的尺度也就越大。L(x,y,σ)代表着图像的高斯尺度空间。

构建尺度空间的目的是为了检测出在不同的尺度下都存在的特征点，而检测特征点较好的算子是高斯拉普拉斯,LoG。

使用LoG虽然能较好的检测到图像中的特征点，但是其运算量过大，通常使用DoG（差分高斯）来近似计算LoG。设k为相邻两个高斯尺度空间的比例因子，则DoG的定义为：

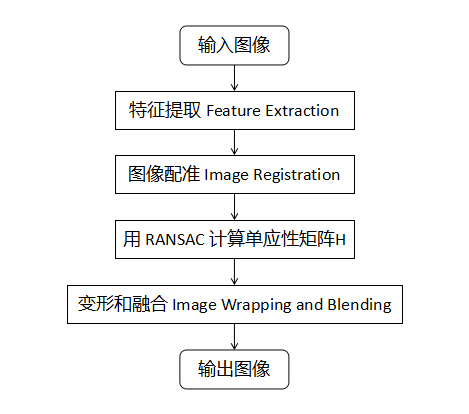


其中，L(x,y,σ)是图像的高斯尺度空间。

从上式可以知道，将相邻的两个高斯空间的图像相减就得到了DoG的响应图像。为了得到DoG图像，先要构建高斯尺度空间，而高斯的尺度空间可以在图像金字塔降采样的基础上加上高斯滤波得到，也就是对图像金字塔的每层图像使用不同的参数σ进行高斯模糊，使每层金字塔有多张高斯模糊过的图像。降采样时，金字塔上边一组图像的第一张是由其下面一组图像倒数第三张降采样得到。

高斯金字塔有多组，每组又有多层。一组中的多个层之间的尺度是不一样的（也就是使用的高斯参数σ是不同的），相邻两层之间的尺度相差一个比例因子k。

## 三．实验步骤



## 四．程序代码

#图像显示函数  
def show(name,img):  
 cv2.imshow(name, img)  
 cv2.waitKey(0)  
 cv2.destroyAllWindows()  
#读取输入图片  
ima = cv2.imread("you.jpg")  
imb = cv2.imread("zuo.jpg")  
A = ima.copy()  
B = imb.copy()  
imageA = cv2.resize(A,(0,0),fx=0.2,fy=0.2)  
imageB = cv2.resize(B,(0,0),fx=0.2,fy=0.2)  
#检测A、B图片的SIFT关键特征点，并计算特征描述子  
def detectAndDescribe(image):  
 # 建立SIFT生成器  
 sift = cv2.SIFT\_create()  
 # 检测SIFT特征点，并计算描述子  
 (kps, features) = sift.detectAndCompute(image, None)  
 # 将结果转换成NumPy数组  
 kps = np.float32([kp.pt for kp in kps])  
 # 返回特征点集，及对应的描述特征  
 return (kps, features)  
  
#检测A、B图片的SIFT关键特征点，并计算特征描述子  
kpsA, featuresA = detectAndDescribe(imageA)  
kpsB, featuresB = detectAndDescribe(imageB)  
# 建立暴力匹配器  
bf = cv2.BFMatcher()  
# 使用KNN检测来自A、B图的SIFT特征匹配对，K=2  
matches = bf.knnMatch(featuresA, featuresB, 2)  
good = []  
for m in matches:  
 # 当最近距离跟次近距离的比值小于ratio值时，保留此匹配对  
 if len(m) == 2 and m[0].distance < m[1].distance \* 0.75:  
 # 存储两个点在featuresA, featuresB中的索引值  
 good.append((m[0].trainIdx, m[0].queryIdx))  
  
# 当筛选后的匹配对大于4时，计算视角变换矩阵  
if len(good) > 4:  
 # 获取匹配对的点坐标  
 ptsA = np.float32([kpsA[i] for (\_, i) in good])  
 ptsB = np.float32([kpsB[i] for (i, \_) in good])  
 # 计算视角变换矩阵  
 H, status = cv2.findHomography(ptsA, ptsB, cv2.RANSAC,4.0)  
  
# 匹配两张图片的所有特征点，返回匹配结果  
M = (matches, H, status)  
# 如果返回结果为空，没有匹配成功的特征点，退出程序  
if M is None:  
 print("无匹配结果")  
 sys.exit()  
# 否则，提取匹配结果  
# H是3x3视角变换矩阵  
(matches, H, status) = M  
# 将图片A进行视角变换，result是变换后图片  
result = cv2.warpPerspective(imageA, H, (imageA.shape[1] + imageB.shape[1], imageA.shape[0]))  
# 将图片B传入result图片最左端  
result[0:imageB.shape[0], 0:imageB.shape[1]] = imageB  
show('res',result)  
print(result.shape)

## 五．效果展示

### 原图：



### 拼接后：



## 六．实践总结

1.选取景深较大景物去拍摄，景深较小的景物会产生大量的形变。由于是水平拼接，一定要保证拍摄时的垂直高度保持一致，否则会发生图像垂直错位的情况；

2.可以发现直接拼接虽然可以拼接但是在拼接的地方会有一条很明显的缝隙，不过直接拼接的

速度比较快；