机器视觉实验报告（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

2.1 DOG

高斯差分图像金字塔（Difference of Gaussian，DOG）是一种常用的图像尺度空间表示方法，常用于计算机视觉领域中的目标检测、图像匹配等任务中。图像高斯金字塔(Gaussian Pyramid)是采用高斯函数对图像进行模糊以及降采样处理得到，差分金字塔的是在高斯金字塔的基础上操作的，其建立过程是：在高斯金子塔中的每组中相邻两层相减（下一层减上一层）就生成高斯差分金字塔。使用 DOG 对图像进行归一化处理，从而提取出一些特征在不同模糊程度、不同尺度下都存在的特征。

2.2 SIFT

SIFT 算法是一种图像处理领域中的一种局部特征描述算法。SIFT 算法的主要目的是寻找和描述图像中的关键点，即具有尺度不变性和旋转不变性的特征点。SIFT 算法的主要步骤有四个：（1）尺度空间的极值检测，通过高斯金字塔和高斯差分金字塔来寻找候选的特征点；（2）关键点的精确定位，通过对比测试和边缘测试来剔除不稳定的特征点；（3）关键点的主方向分配，通过计算关键点周围区域的梯度方向直方图来确定关键点的主方向和辅方向；（4）关键点描述子的生成，通过对关键点周围区域进行划分和统计来生成一个向

量来描述关键点的信息。

2.3 RANSAC

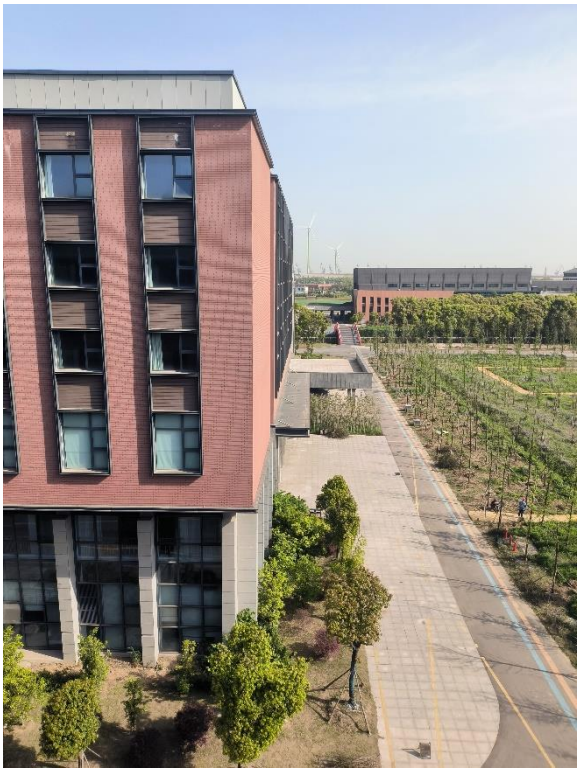
RANSAC（Random Sample Consensus，随机抽样一致性）是一种基于随机采样的迭代算法，用于解决存在噪声和异常值的数据拟合问题。RANSAC 算法的基本思想是：从所有数据中随机选择一小组数据（称为样本），并用这些样本计算模型参数。然后，计算所有其他数据与该模型之间的误差，并将误差小于某个阈值的数据视为该模型的内点。重复此过程多次，选择具有最多内点的模型作为最终的拟合结果。

由于 RANSAC 算法基于随机采样，所以它对于含有异常值和噪声的数据集也能有效地工作。它的主要优点是简单易懂，易于实现，并且能够处理多种类型的模型拟合问题。但是，RANSAC 算法的缺点是计算成本高，迭代次数不易确定，并且结果可能不稳定。

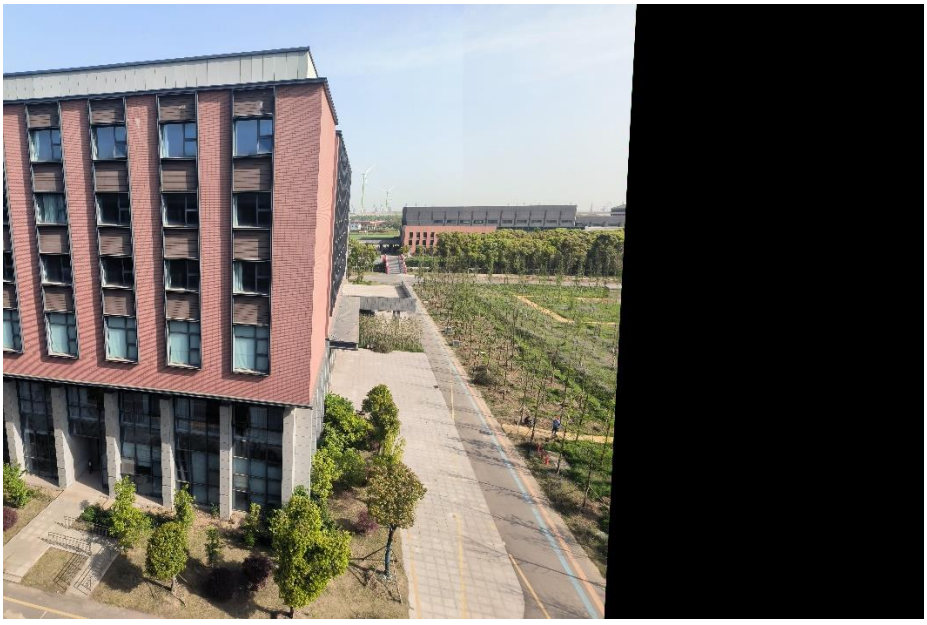
三. 实现说明

通过 SIFT 算法检测每张图片中的特征点，然后将两张图片中的特征点进行匹配以找到两张图片的视角变换以及相对位置。再通过计算透视变换矩阵，将一张图片变换到另一张图片中匹配的位置。最后将两张图片进行拼接得到最终的结果

四. 结果截图



将上面两张图进行拼接，得到拼接后的图



拼接结果

五. 运行说明

- 图片读取。对于两张想要拼接的图片，首先使用 matplotlib.pyplot 模块进

行读取。

```
img1 = plt.imread(img1)
img2 = plt.imread(img2)
```

- 特征提取。使用 opencv 库的 SIFT_create()方法构造一个 SIFT 特征提取器对象，并对两张图片进行关键点检测和特征提取。

```
sift = cv2.SIFT_create()
kp1, des1 = sift.detectAndCompute(img1, None)#
kp2, des2 = sift.detectAndCompute(img2, None)
kp1 = np.array([kp.pt for kp in kp1])
kp2 = np.array([kp.pt for kp in kp2])
```

- 关键点匹配。得到关键点及各自的特征之后，使用 cv2.BFMatcher 类的 knnMatch 方法对两张图片的关键点进行匹配。

```
ratio=0.75
matcher = cv2.BFMatcher()
# 使用 KNN 检测来自 A、B 图的 SIFT 特征匹配对，K=2
rawMatches = matcher.knnMatch(des1, des2, 2)
matches = []
for m in rawMatches:
    # 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 * ratio:
    # 存储两个点在 featuresA, featuresB 中的索引值
        matches.append((m[0].trainIdx, m[0].queryIdx))
```

遍历所有的匹配对，当遍历到的匹配对的最近距离与次近距离的比值小于设定的阈值时，选择保留此匹配对

- 计算视角变换矩阵

```
# 当筛选后的匹配对大于 4 时，计算视角变换矩阵
reprojThresh = 10
if len(matches) > 4:
    # 获取匹配对的点坐标
    ptsA = np.float32([kp1[i] for (_, i) in matches])
```

```
    ptsB = np.float32([kp2[i] for (i, _) in matches])
    # 计算视角变换矩阵
    (H, status) = cv2.findHomography(ptsA, ptsB, cv2.RANSAC,
reprojThresh)
```

当筛选到的匹配对数量大于 4 时才计算视角变换矩阵，读取筛选到的匹配对，通过 cv2.findHomography 方法计算单应性矩阵，并指定使用 RANSAC 算法计算单应性矩阵

- 图像变换

```
result = cv2.warpPerspective(img1, H, (img1.shape[1] +
img2.shape[1], img1.shape[0]))
result[0:img2.shape[0], 0:img2.shape[1]] = img2
```

将 img1 进行视角变换，通过单应性矩阵可以将一个图像中的点映射到另一个图像中的对应点。使用 cv2.warpPerspective 将 img1 进行视角变换，再将 img2 拼接到图像的右侧，从而完成图像拼接。