计算机视觉实践报告（一）

目录

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 图像拼接

图像拼接技术的基本流程图如下：图像采集、图像处理、图像配准、图像融合。

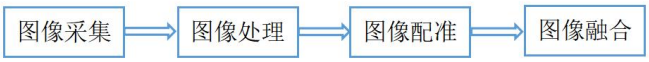
（1）图像获取：使用图像采集设备进行图像采集和获取，有平移式、旋转式、手持式等获取方式；

（2）图像处理：图像配准前的预备工作，消除影响图像配准精度的无关信息，提升图像配准效率。图像预处理包括图像去噪、几何校正、均匀颜色等。目的是提高配准精度、降低配准难度，调整灰度差异、去噪、几何修正以及将两幅图像投影到同一坐标系等；

（3）图像配准：将多幅图像进行匹配、叠加。算法既要保证配准精度，又不能计算量过大，应当能够估计待拼接图像之间可能存在的缩放、旋转、仿射变换、投影变换以及亮度和颜色等变化；

（4）图像融合：图像配准可能由于算法误差累积、色彩差异等原因，导致图像出现拼

接缝隙，以及整幅图像的颜色亮度差异。因此在图像配准处理后需要进行图像融合处理，矫正差异，消除缝隙，才能使拼接的图像更加自然。图像融合的目的是得到无缝的高质量图像。在不损失原始图像信息的前提下，消除接缝与亮度差异，实现拼接边界的平滑过渡。



2.2 图像配准

图像配准计算出两幅图像间的空间变换模型并进行空间变换，使两幅图像的重叠部分在空间上对准。图像之间的空间变换关系包括: 平移、旋转、尺度缩放、仿射变换、投影变换，其中投影变换更具有普遍性。

图像拼接的关键是精确找出相邻两张图像中重叠部分的位置，然后确定两张图像的变换关系，即图像配准。由于视角、拍摄时间、分辨率、光照强度、传感器类型等差异，待拼接图像往往存在平移、旋转、尺度变化、透视形变、色差、扭曲、运动目标遮挡等差别，配准目的是找出一种最能描述待拼接 图像之间映射关系的变换模型。目前常用的一些空间变换模型有平移变换、刚性变换、仿射变换以及投影变换等。

假设图像 $f_1(x,y)$、$f_2(x,y)$存在投影变换关系，则用以下齐次方程表示：

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

m_0、m_1、m_3 和 m_4 表示旋转角度和缩放尺度；m_2 和 m_5 分别表示 x 与 y 方向上的平移量；m_6 和 m_7 分别表示 x 和 y 方向上的变形量。图像配准用上式确定空间变换模型 M 的参数，根据各参数的意义及不同变换模型的特点，对矩阵 M 作相应简化就可以得到各变换模型的参数矩阵。

图像配准可以分为基于频域和基于空间域的，本次实验采用基于空间域的 SIFT 特征的匹配方法。

2.3 基于特征匹配

基于特征的图像拼接是利用图像的明显特征来估计图像之间的变换，如轮廓、角点等，而不是利用图像全部的信息。

图像的配准问题可以归结为求解对应点集。在待配准的图像中选取一些特征点，对准了这些特征点，两幅图像也就配准了。控制点法往往要借助人工选取初始匹配点，影响算法的速度和适用范围。因此要采用一些数学方法自动实现图像间对应控制点的选取。

基于特征的配准算法主要流程图如下，包含三个步骤：特征提取、特征匹配和变换矩阵求解。通过对图像的像素点归纳抽象出高级语义的图像特征，以特征作为待配准图像的匹配依据，对重叠区域的图像进行特征搜索确定图像之间的匹配关系。图像的特征可以是一个区域、一条线或者是一个点，由于特征点具有几何变换的鲁棒性，故在图像拼接中，一般采用特征点作为配准的特征。

（1）特征提取：提取特征是为了对图像进行匹配，根据待配准图像的特点考虑选择何种特征，在特征的选择上主要考虑三个方面：第一，选择待配准的两幅图像中都存在的特征；第二，能够同时在两幅图像中提取出足够数量的特征；第三，选取的特征必须具有较好的独特性且便于下一步的特征匹配工作。

（2）特征匹配：在提取出待配准图像的特征集后，要确定特征集中特征点的匹配关系，即进行特征匹配。首先进行高层次的语义描述，对每个特征点抽象出一个特征描述符，通过

特征描述符来寻找匹配的特征点。由于特征点集数据量的庞大，可以采用一定的搜索策略来进行匹配对的确定，通常采用的搜索算法有 k-d 树、BBF 算法等。

（3）变换矩阵的求解：经过特征点匹配确定相邻两幅图像的对应特征点匹配对集，然后根据对应点匹配集求出透视变换矩阵。通过透视变换矩阵将处于不同像素坐标系下的待配准图像变换到统一的坐标系下，完成图像配准。透视变换矩阵的求解关系到配准的质量好坏，仅仅从待配准图像的初始匹配对中求解代表摄像机的复杂运动模型的变换矩阵很难，通常应先进行匹配对的提纯，然后采用迭代求精的方法确定精确的变换矩阵参数，常用方法有 M 估计法、随机抽样一致性(RANSAC)算法、MLESAC 算法等。

基于特征的配准方法有如下优点：1）适用范围广，位移较大的待配准图像也能够通过提取特征进行配准；2）配准的依据是图像的特征而不是图像的所有像素点，只利用的图像部分信息可以降低运算量，提高配准效率；3）由于特征匹配是针对高层次语义上的特征描述符，故算法对于待配准图像之间存在噪声干扰和亮度差异具有很好的鲁棒性。

基于特征的配准方法缺点：只利用图像的部分信息，如果特征提取的数量过少，在特征匹配阶段即使是很小的错误都会导致最终配准的失败。故基于特征的配准算法其核心关键在于选取鲁棒性高的特征并且精确地进行特征匹配。

2.4 基于 SIFT 的特征描述

Harris 角点检测可以很好的检测到角点，这与角点本身的特性也有关系，就算在图像旋转的情况下，也能将角点检测出来。然而，如果减小（或增加）图像的大小，那么图像就会丢失某些部分，甚至有可能增加角点的质量。

所以要有一个能够无视图片尺度变换，也能准确检测到图像特征（比如角点）的算法。然后 David Lowe 就提出尺度不变特征变换（Scale-Invariant Feature Transform, SIFT）。SIFT 算法的实质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。

2.4.1 图像尺度空间

在一定的范围内，无论物体是大还是小，人眼都可以分辨出来，要让计算机能够对物体在不同尺度下有一个统一的认识，就要考虑图像在不同的尺度下都存在的特点，对图像进行不同尺度的改变。

在不同的尺度空间不能使用相同的窗口检测极值点，对小的关键点使用小的窗口，对大的关键点使用大的窗口，为了达到上述目的，使用尺度空间滤波器。

高斯核是唯一可以产生多尺度空间的核函数。

一个图像的尺度空间 $L(x,y,\sigma)$，定义为原始图像 $I(x,y)$与一个可变尺度的二维高斯函数 $G(x,y,\sigma)$卷积运算，即：$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y)$。G 为高斯核函数，σ为尺度空间因子，决定了图像的模糊的程度。大尺度下（σ值大）表现图像的概貌信息，小尺度下（σ值小）表现图像的细节信息。

2.4.2 多分辨率高斯图像金字塔

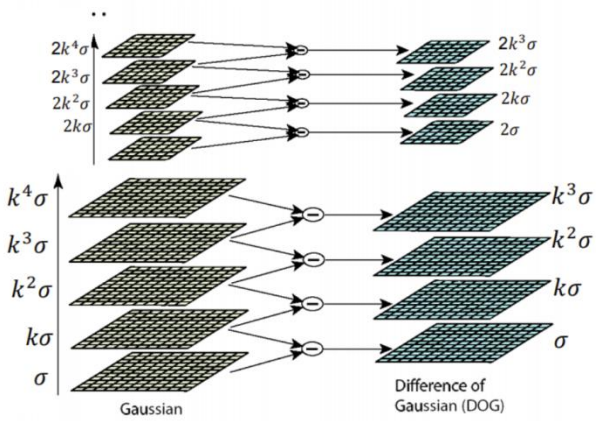
为解决分辨率不同的问题，以 6 幅不同高斯核处理的模糊图像为一组，进行图像金字塔的下采样处理，得到不同分辨率的图像，然后每个分辨率都有 6 张不同模糊程度的图像，以达到多尺度的目的。

2.4.3 高斯差分图像金字塔（DOG）

上面用高斯图像金字塔处理了图像，让 6 个不同高斯模糊程度不同的图片都可以缩小成其他不同的尺寸，而高斯差分图像金字塔（DOG）就是将图像金字塔每一层的图片（6 张不同模糊程度，相同尺寸）进行差分运算。差分运算后得出的特征图，就代表这两张图片的差异性结果。

每一层的第一与第二幅图片进行差运算、第二与第三幅进行差运算。以此类推，逐组逐层生成每一个差分图像，所有差分图像构成差分金字塔。因此，DOG 金字塔的第 o 组第 l 层图像是有高斯金字塔的第 o 组第 l+1 层减第 o 组第 l 层得到的。后续 Sift 特征点的提取都是在 DOG 金字塔上进行。

$k = 2^{1/s}$



DOG定义公式：

$$D(x,y,\sigma) = [G(x,y,k\sigma) - G(x,y,\sigma)] * I(x,y) = L(x,y,k\sigma) - L(x,y,\sigma)$$

2.4.4 DOG 空间关键点检测

DoG 可以在不同的尺度空间中搜索局部最大值。图像中的每个像素点要和其图像域（同一尺度空间的 8 个点）和尺度域（相邻的尺度空间——上下各 9 个点，一共 18 个）的所有相邻点（一共 26 个）进行比较，当其大于或者小于所有相邻点时，就会被认为是关键点(局部极值点)。关键点是图像在相应尺度空间中的最好代表。

由于 DOG 图像金字塔每一层的最上面和最下面的图片无法进行比较，所以 5 张 DOG 特征图中只有中间 3 张才能检测到关键点。

但这一步检测的极值点是离散的，不一定是最终的关键点，所以要进行下一步——极值点的精确定位。

2.4.5 关键点（极值点）的精确定位

检测出的关键点是 DOG 空间的局部极值点，而且这些极值点均为离散的点，精确定位极值点的一种方法是：对尺度空间 DOG 函数进行曲线拟合，计算极值点，从而实现关键点的精确定位。

第一步就得进行曲线拟合，就是将这些离散的点变成一条连续的线。方法就是用泰勒展开公式。使用尺度空间的泰勒级数展开来获得极值的准确位置，如果极值点的灰度值小于阈值（一般为 0.03 或 0.04）就会被忽略掉。 在 OpenCV 中这种阈值称为 contrastThreshold。

2.4.6 消除边界响应

DoG 算法对边界非常敏感，在关键点定位后，关键点中很容易掺杂着边界，所以需要将

边界响应去除。利用 Hessian 矩阵，模仿 Harris 算法中的 M 矩阵，找到边界并消去。

2.4.7 特征点的主方向

经过上述两个步骤，图像的关键点就完全找到了，这些关键点具有尺度不变性。为了实现旋转不变性，还需要为每个关键点分配一个方向角度，也就是根据检测到的关键点所在高斯尺度图像的邻域结构中求得一个方向基准。

对于任一关键点，采集其所在高斯金字塔图像以 r 为半径的区域内所有像素的梯度特征（幅值和幅角），半径 r = 3 x 1.5σ，σ是关键点所在 octave 的图像的尺度，可以得到对应的尺度图像。每个点 L(x,y)的梯度的模 m(x,y)和方向θ(x,y)的计算公式如下：

$$m(x,y) = \sqrt{[L(x+1,y) - L(x-1,y)]^2 + [L(x,y+1) - L(x,y-1)]^2}$$

$$\theta(x,y) = \arctan \frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}$$

每个特征点可以得到三个信息(x,y,σ,θ)，即位置、尺度和方向。具有多个方向的关键点复制成多份，然后将方向值分别赋予复制后的特征点，一个特征点产生多个坐标、尺度相等，但方向不同的特征点。

2.4.8 生成特征描述

完成关键点梯度计算后，使用直方图统计关键点邻域内像素的梯度幅值和方向。将 360° 分为 36 柱，每 10° 为一柱，然后在以 r 为半径的区域内，将梯度方向在某一个柱内的像素找出来，然后将他们的幅值相加在一起作为柱的高度。

因为在 r 为半径的区域内像素的梯度幅值对中心像素的贡献是不同的，因此还需要对幅值进行加权处理，采用高斯加权，方差为 1.5σ。

每个特征点必须分配一个主方向，还需要一个或多个辅方向，增加辅方向是为了增强图像匹配的鲁棒性。辅方向的定义：当一个柱体的高度大于主方向柱体高度的 80%时，则该柱体所代表的的方向就是特征点的辅方向。

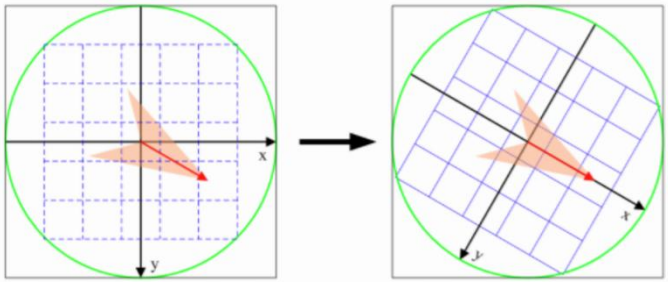
直方图的峰值，即最高的柱代表的方向是特征点邻域范围内图像梯度的主方向，但该柱体代表的角度是一个范围，所以还要对离散的直方图进行插值拟合，以得到更精确的方向角度值。利用抛物线对离散的直方图进行拟合。

获得图像关键点主方向后，每个关键点有三个信息(x,y,σ,θ)：位置、尺度、方向。由此可以确定一个 SIFT 特征区域。通常使用一个带箭头的圆或直接使用箭头表示 SIFT 区域的三个值：中心表示特征点位置，半径表示关键点尺度，箭头表示方向。

通过以上步骤，每个关键点被分配了位置、尺度和方向信息。接下来为每个关键点建立一个描述符，该描述符既具有可区分性，又具有对某些变量的不变性，如光照、视角等。而且描述符不仅仅包含关键点，也包括关键点周围对其有贡献的像素点。主要思路：通过将关键点周围图像区域分块，计算块内的梯度直方图，生成具有特征向量，对图像信息进行抽象。

描述符与特征点所在的尺度有关，所以在关键点所在的高斯尺度图像上生成对应的描述符。以特征点为中心，将其附近邻域划分为 d*d 个子区域（一般取 d=4），每个子区域都是一个正方形，边长为 3σ，考虑到实际计算时需进行三次线性插值，所以特征点邻域为 3σ(d+1) * 3σ(d+1)的范围。

为了保证特征矢量的旋转不变性，要以特征点为中心，在附近邻域内将坐标轴旋转θ角度，即将坐标轴旋转为特征点的主方向。



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

以旋转之后的主方向为中心，取一个 8×8 的窗口，求每个像素的梯度幅值和方向，箭头方向代表梯度方向，长度代表梯度幅值。然后利用高斯窗口对其进行加权运算(σ=0.5d)，最后在每个 4×4 的小窗口上绘制 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、即每个特征都由 4 个种子点组成，每个种子点有 8 个方向的向量信息。

2.4.9 综述基于 SIFT 的配准算法步骤

1）检测尺度空间极值点，初步确定关键点的位置和所在尺度。

2）精确确定关键点的位置和尺度，同时剔除低对比度的关键点和不稳定的边缘响应点。

3）分配关键点方向。利用关键点邻域像素的梯度方向分布特性为每个关键点指定方向参数，保证 SIFT 算子的旋转不变性。

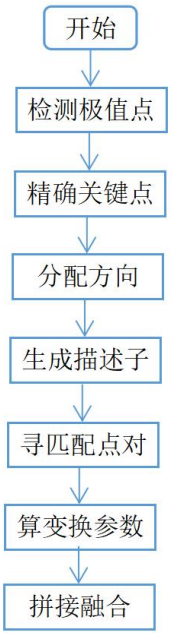
4）生成关键点描述子。将坐标轴旋转为关键点的方向，然后以关键点为中心取 8×8 的窗口，计算每个 4×4 的小块上 八个方向的梯度方向直方图，每个梯度方向的累加值形成一个种子点。实际计算过程中，为了增强匹配的稳健性。

5）生成两幅图像的 SIFT 特征向量后，采用关键点特征向量的欧式距离作为两幅图像中关键点的相似性判定准则。得到满足准则的 SIFT 匹配点对。

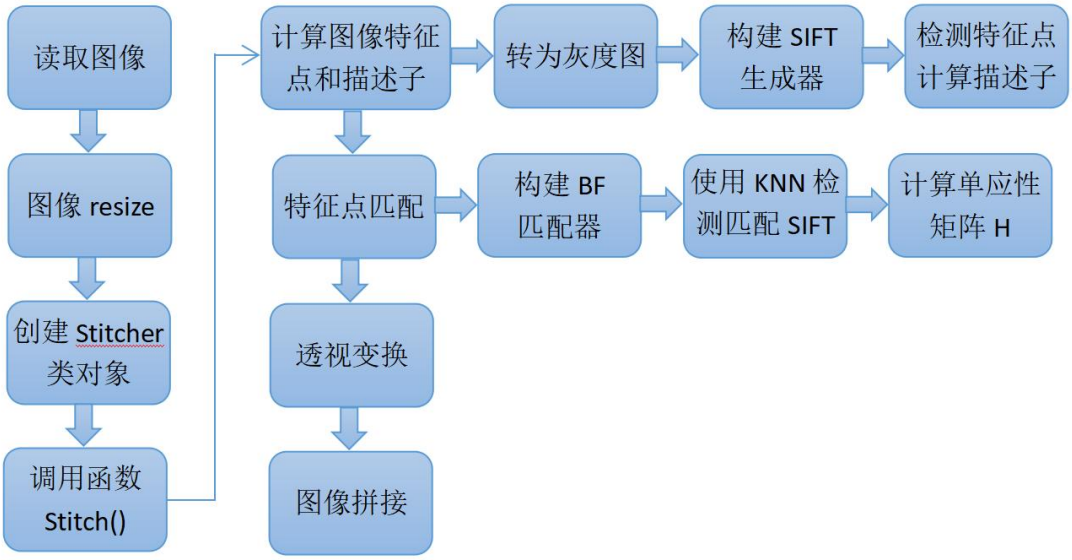
6）根据得到的 SIFT 匹配点对计算出图像的变换参数。

7）进行拼接融合得到全景图像。

SIFT 特征是图像的局部特征，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、噪声也具有一定的鲁棒性。



三．实验步骤



四．程序代码

本章给出主体函数介绍，包括图像特征点和描述子的计算、特征点的匹配、透视变换和图像拼接。

4.1 图像特征点和描述子的计算

```python
def detectAndDescribe(self, image):
    # 转换为灰度图
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # 建立SIFT生成器
    descriptor = cv2.xfeatures2d.SIFT_create()
    # 检测特征点并计算描述子
    kps, features = descriptor.detectAndCompute(gray, None)

    kps = np.float32([kp.pt for kp in kps])

    return kps, features
```

4.2 特征点匹配

Brute-Force 匹配：首先在第一幅图像中选择一个关键点然后依次与第二幅图像的每个关键点进行（改变）距离测试，最后返回距离最近的关键点。

有两种匹配方式：BFMatcher.match()和 BFMatcher.knnMatch()，第一个返回最佳匹配，第二种方法返回 top k 个最佳匹配，其中 k 由用户指定。

代码中采用第二种方法 matcher.knnMatch(featureA, featureB, 2)，featureA 和 featureB 表示两个特征向量集，K 表示按 knn 匹配规则输出的最优的 K 个结果。输出 featureA（检测图像）中每个点与被匹配对象 featureB（样本图像）中特征向量进行运算的匹配结果。最终得

到 featureA 条记录，每一条有最优 K 个匹配结果。每个结果包含三个数据：queryIdx，trainIdx，distance。queryIdx：特征向量的特征点描述符的下标（第几个特征点描述符），同时也是描述符对应特征点的下标；trainIdx：样本特征向量的特征点描述符下标,同时也是描述符对应特征点的下标；distance：代表匹配的特征点描述符的欧式距离，数值越小也就说明俩个特征点越相近。

通过近邻原则过滤：匹配上的特征点，周围的特征点，应该也是与被匹配点相似的。如果只有最近特征点满足，而次近特征点不满足，则不接受。

当筛选后的匹配对大于 4 时，计算视角变换矩阵。获取两张图中的匹配点，将其中一张图通过旋转、变换等方式将其与另一张图对齐。得到众多匹配点后，使用 RANSAC 算法，每次从中筛选四个随机的点，然后求得 H 矩阵，不断的迭代，直到求得最优的 H 矩阵为止。

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} \mathbf{m_0} & \mathbf{m_1} & \mathbf{m_2} \\ \mathbf{m_3} & \mathbf{m_4} & \mathbf{m_5} \\ \mathbf{m_6} & \mathbf{m_7} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

计算多个二维点对之间的最优单映射变换矩阵 H（3*3），status 为可选的输出掩码，0/1 表示在变换映射中无效/有效 cv2.findHomography(srcPoints,dstPoints,method=None, ransacReprojThreshold=None,mask=None,maxIters=None,confidence=None)-->(H,status)：

method（0， RANSAC, LMEDS, RHO）

ransacReprojThreshold 一对内群点容忍的最大允许重投影错误阈值（限 RANSAC 和 RHO）

mask 可选输出掩码矩阵，通常由鲁棒算法（RANSAC 或 LMEDS）设置，是不需要设置的

maxIters 为 RANSAC 算法的最大迭代次数，默认值为 2000

confidence 可信度值，取值范围为 0 到 1

```python
def matchKeypoints(self, kpsA, kpsB, featureA, featureB, ratio, reprojThresh):
    # 建立BF匹配器
    matcher = cv2.BFMatcher()

    # 使用KNN检测来自AB图的SIFT特征匹配，返回top k个最相似的特征点，k=2
    rawMatches = matcher.knnMatch(featureA, featureB, 2)

    # 过滤
    matches = []
    for m in rawMatches:
        # 对于特征点，是否接受匹配，根据d1/d2 < 0.75(ratio)来判断，因为两个关键点中，如果最近距离/次近距离 < ratio，则接受。
        if len(m) == 2 and m[0].distance < m[1].distance * ratio:
            matches.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(matches) > 4:
        # 获取匹配对的点坐标
        ptsA = np.float32([kpsA[i] for (_, i) in matches])
        ptsB = np.float32([kpsB[i] for (i, _) in matches])

        # H为求得的单应性矩阵矩阵，status返回一个列表来表征匹配成功的特征点。参数cv2.RANSAC表明使用RANSAC算法来筛选关键点
        H, status = cv2.findHomography(ptsA, ptsB, cv2.RANSAC, reprojThresh)

        # 返回匹配的特征点对、单应矩阵、匹配成功的特征点
        return matches, H, status
```

4.3 透视变换和图像拼接

利用以获得的单应性矩阵 H 对左图进行透视变换，将透视变换后的图片与右图拼接，展示最终结果，同时根据选择展示图像匹配结果。

```python
def stitch(self, images, ratio=0.75, reprojThresh = 4.0, showMatches = True):
    # 读取图像
    imageB, imageA = images
    # 计算特征点和特征向量
    kpsA, featureA = self.detectAndDescribe(imageA)
    kpsB, featureB = self.detectAndDescribe(imageB)

    # 匹配两张图片的特征点
    M = self.matchKeypoints(kpsA, kpsB, featureA, featureB, ratio, reprojThresh)

    # 没有匹配点，退出
    if not M:
        return None

    # 匹配的特征点对、单应矩阵、匹配成功的特征点
    matches, H, status = M

    # 将图片A进行视角变换中间结果
    result = cv2.warpPerspective(imageA, H, (imageA.shape[1] + imageB.shape[1], imageA.shape[0]))
    # 将图片B传入
    result[0:imageB.shape[0], 0:imageB.shape[1]] = imageB
    self.cv_show('result', result)

    # 检测是否需要显示图片匹配
    if showMatches:
        # 生成匹配图片
        vis = self.drawMatches(imageA, imageB, kpsA, kpsB, matches, status)
        # 返回结果
        return result, vis

    # 返回匹配结果
    return result
```

调用 cv2.warpPerspective(src, M, dsize=(cols, rows))实现透视变换。src：原图；M：一个 3x3 的变换矩阵；dsize：输出图像的尺寸大小，第一个参数是 col，第二个参数是 row。

图像展示如下：

```python
# 展示图像
def cv_show(self,name, img):
    cv2.imshow(name, img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

绘制匹配特征点对函数如下：

```python
def drawMatches(self, imageA, imageB, kpsA, kpsB, matches, status):
    # 初始化可视化图片，将A、B图左右连接到一起
    hA, wA = imageA.shape[:2]
    hB, wB = imageB.shape[:2]
    vis = np.zeros((max(hA, hB), wA + wB, 3), dtype="uint8")
    vis[0:hA, 0:wA] = imageA
    vis[0:hB, wA:] = imageB

    # 联合遍历，画出匹配对
    for ((trainIdx, queryIdx), s) in zip(matches, status):
        # 当点对匹配成功时，画到可视化图上
        if s == 1:
            # 画出匹配对
            ptA = (int(kpsA[queryIdx][0]), int(kpsA[queryIdx][1]))
            ptB = (int(kpsB[trainIdx][0]) + wA, int(kpsB[trainIdx][1]))
            cv2.line(vis, ptA, ptB, (0, 255, 0), 1)

    # 返回可视化结果
    return vis
```

五．实验结果和分析

代码包含两个文件，一是直接运行的 demo，一是用于图像拼接函数实现的类 Stitcher。

一共进行四组图像的实验测试。第一、四组是网上获取的图像展示效果图，第二组是自己最近拍的风景图的拼接效果，第三组是在同一地点手动拍摄的两张图。

图 1 和图 2 是待拼接的两张图像，图 3 则是拼接后的效果图。

可以看出，第二组的效果最好，主要是因为第二组图像是从同一张图像中分割而来，可以看出图 1 和图 2 中间有一个岛屿是重合的，拼接过程中就根据提取出的 SIFT 特征点进行左右两图的特征匹配，进而进行特征融合，实现图像拼接，最终展示的图像中间部分只包含一个岛屿，去除了重叠部分。相对于另外三组图像，第二组图的拍摄角度、光线一致，故拼接完成后的图像更完整、更平滑，无明显区分界限，和原始的全景图（图 4）基本一致。

图 1

图 2

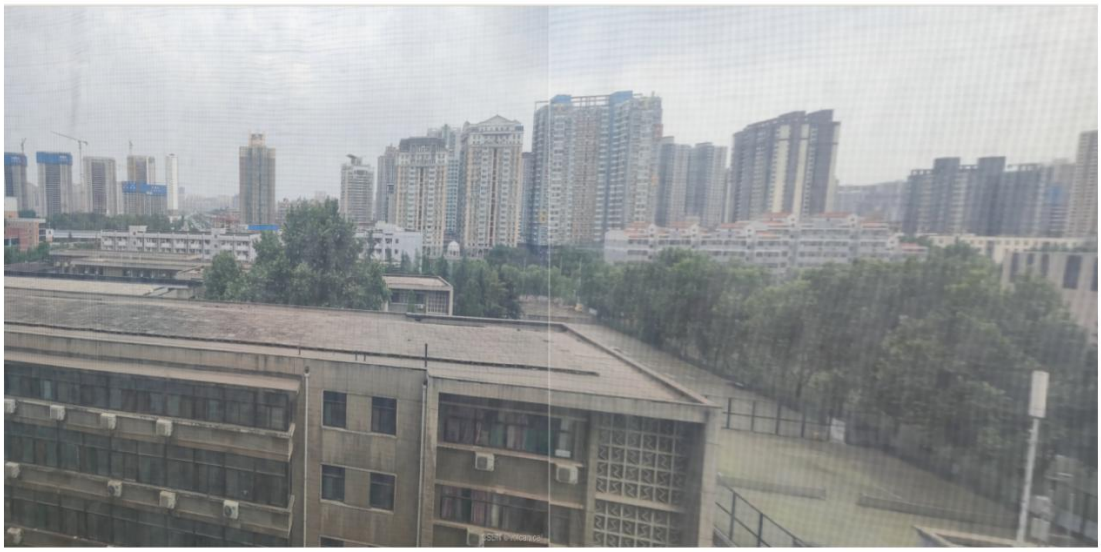


图 3

图 1

图 2

图 3

图 4

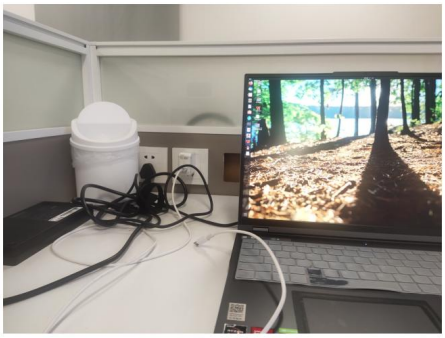


图 1

图 2

图 3

图 1

图 2

图 3