

# Computer vision homework2

wuzhuangzhe

October 2024

## 1 Harris Corner Detector

### 1.1 Calculate spatial derivative

输入为已转换成灰度图的图像，之后再经过一次高斯滤波，最后通过 cv2.sobel 计算梯度。

### 1.2 Calculate Harris response

尝试过使用 for 循环依次计算，结果显示计算效率低，时间长。采用的方法是计算出 grad\_x 和 grad\_y 矩阵，然后求出  $I_x * I_x, I_x * I_y, I_y * I_y$  矩阵。函数输入了窗口大小 win\_size，根据 Harris response 的计算方法，使用大小为  $win\_size * win\_size$  的高斯核和  $I_x * I_x, I_x * I_y, I_y * I_y$  矩阵做卷积，得到 R 矩阵。

### 1.3 Select candidate corners and non-maximal suppression

使用 ndimage.maximum\_filter() 对响应矩阵 R 进行最大滤波。这个滤波器将每个像素替换为其邻域（由 min\_dist 定义的窗口大小）中的最大值。然后将非局部最大值的响应值设置为 0，从而仅保留角点候选位置。最后将响应值 R 中低于阈值 thresh 的像素值设置为 0。使用 np.argwhere(R) 找出 R 中非零像素的索引构成 pix 即可。

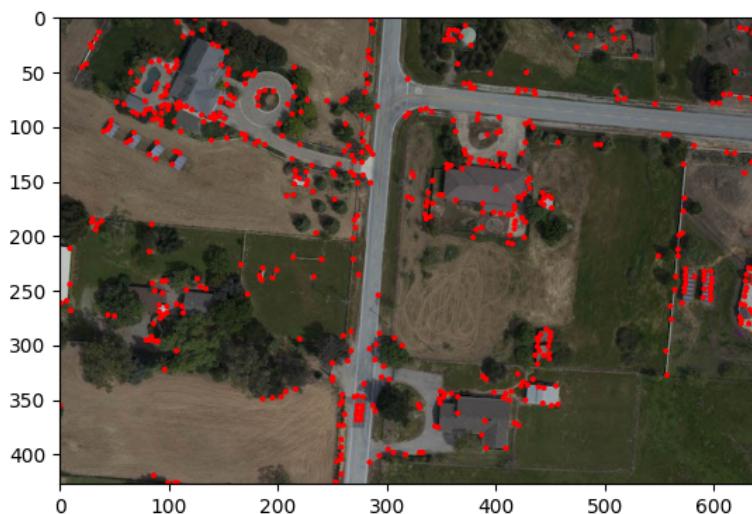


图 1: corners 1\_1

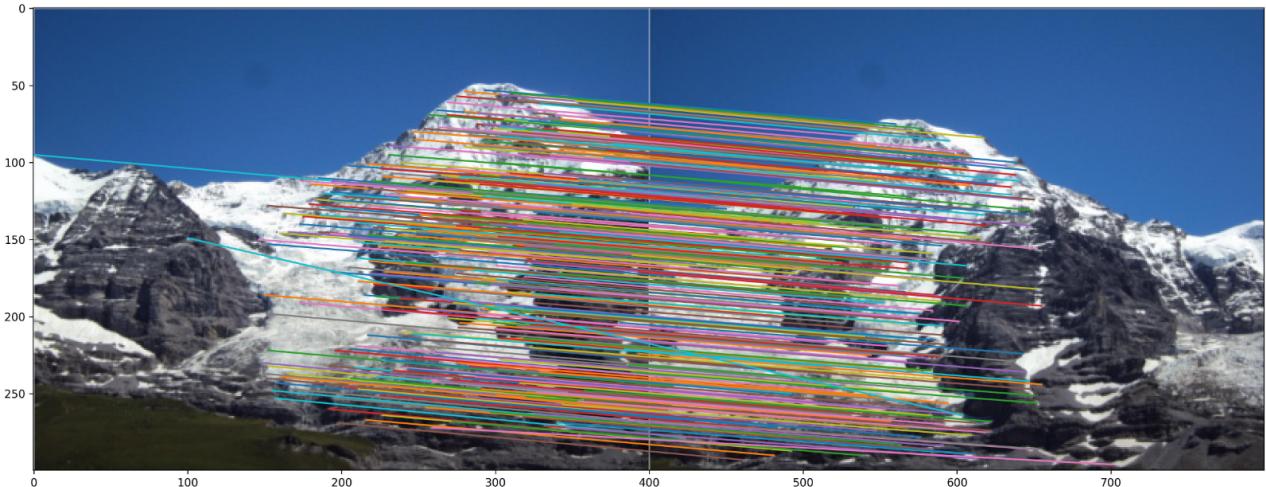
可以运行 test 文件夹中 corners.py 检查角点检测算法。

## 2 Implement Histogram of Gradients

先计算 grad\_x 和 grad\_y 矩阵，然后计算出梯度的长度和方向。将 360 度划分为 8 个 bin 进行统计，对兴趣点的每个 cell 进行统计。并通过旋转使最突出的梯度方向为主方向。这样就实现了该点处的特征直方图。

## 3 Local feature matching





为了方便可视化和调试，`test_matching` 函数改为传入两张图片作为参数。

## 4 Image stitching and Blending

### 4.1 compute\_homography

输入两个长度为  $n$  的位置坐标数组，先构建  $2n \times 9$  的矩阵，并进行 svd 分解，找到最小的奇异值对应的向量，转换成  $3 \times 3$  的矩阵，最后进行归一化。

### 4.2 Align the image with RANSAC

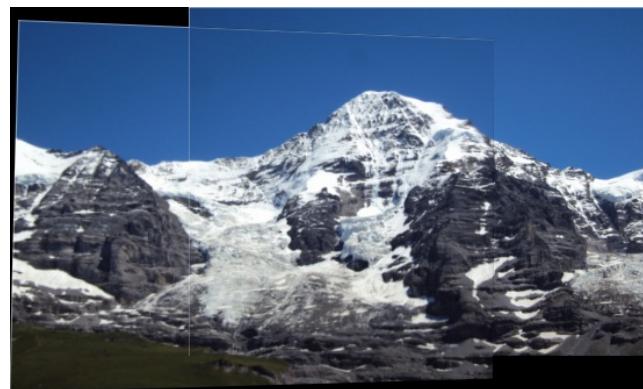
设置迭代次数和阈值，用于判断是否为内点。每次随机选取四组对应点，计算单应性矩阵，寻找该矩阵下的内点并计数，记录含最多内点的集合。根据记录结果，返回单应性矩阵。

### 4.3 Stitch and blend the image

已经提供的函数代码存在问题，在  $x$  和  $y$  方向上的变换存在相反的问题。修改了问题后，可以成功实现  $\alpha$ -blending。为避免黑色像素的影响，创建了 `mask` 数组，标记 RGB 值为 0（黑色）的像素位置。效果对比如下：



三张 blending 图片如下：



#### 4.4 Generate a panorama

采用了已实现的拼接函数。在 grail, library, parrington, XueMountain 等几个数据集上进行了测试。  
效果如下：



图 2: Xue-Mountain-Entrance 0185-0187 3 images

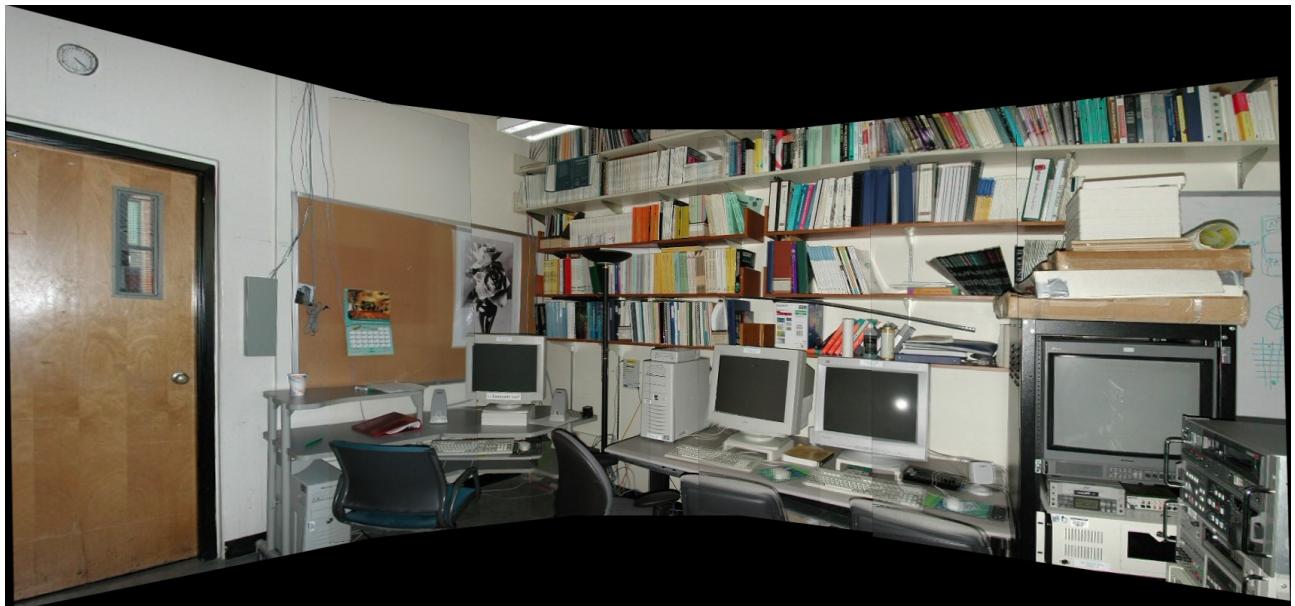


图 3: grail 02-06 5 images

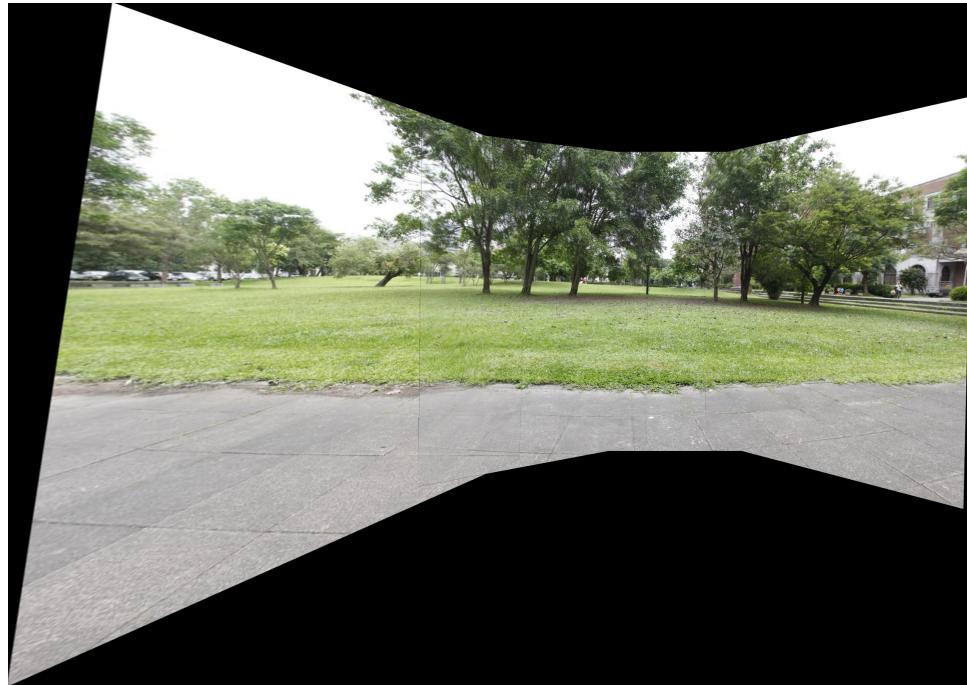


图 4: library 1-4 4 images



图 5: parrington 01-05 5 images

## 5 Analyze

根据实验，有多种因素可能影响图片拼接质量。

### 5.1 超参数的选择

#### 5.1.1 corner\_selection: min\_dist

在进行 maximum\_filter 时，min\_dist 代表所取邻域的大小。若 min\_dist 更小，会提取出更多的角点（兴趣点），后面 matching 的时候也会有更多的匹配点，使单应性矩阵计算更准确，但同时会使计算效率降低。matching 比较效果如下：

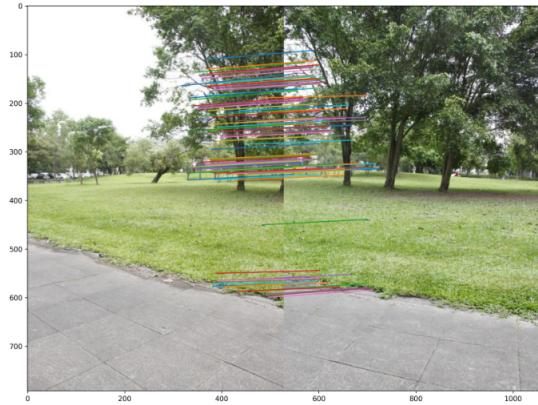


图 6: min\_dist\_3



图 7: min\_dist\_9

### 5.1.2 RANSAC 求解时的迭代次数和阈值

理论上，迭代次数越多，求解结果越接近于两张图片理论上的最优单应性矩阵，运行结果越稳定。目前使用了 300000 次的迭代，结果稳定。可以通过计算排列组合：50 个特征点若取 4 个， $C_{50}^4 \approx 2 \times 10^5$ ，在  $10^5$  数量级下，有较大的概率能找到较优的矩阵。

对于判定阈值，实验是测试了从 0.5-10 中的若干，发现如果匹配结果良好（未出现明显的瑕疵匹配），阈值对结果的影响没有很大，通过断点测试也能发现 RANSAC 计算得出的矩阵，能让大部分 ( $>95\%$ ) 点成为内点。

如果匹配结果中存在瑕疵，则较大的阈值会导致不应该被归为内点的点被此时算得的最优矩阵包容，矩阵与实际较优存在差距。但如果阈值过小，也会产生副作用，比如满足条件的点太少，算得的矩阵有误差。存在这样的 trade-off，最终选取了 threshold=3，对大多数图片较为适用。

## 5.2 图片本身拍摄

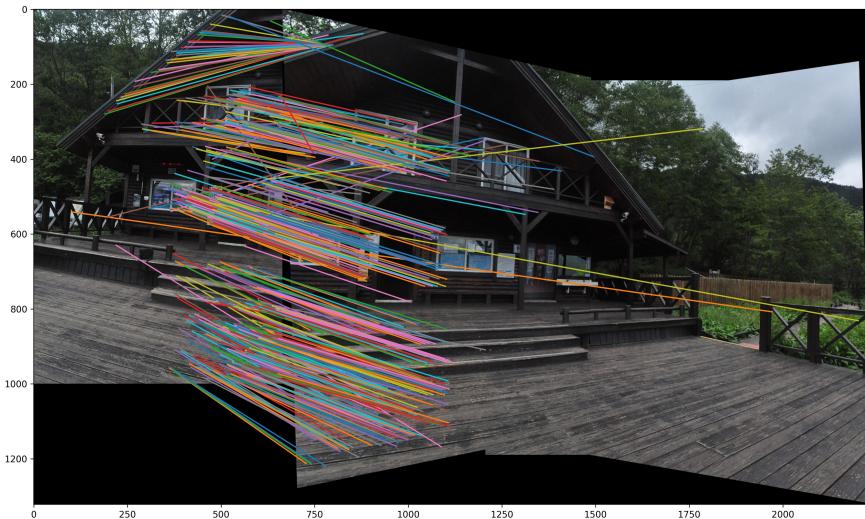
通过以上四张全景图，可以发现畸变程度有较大差异。当拍摄两张照片时，如果转动角度较大，拼接时会产生较大的畸变。如果要拼接多张图片，若先拼成的图片产生了拉伸等畸变，计算 hog 特征时，可能产生较大的误差，导致匹配失败。以雪山 entrance 数据集的 0184-0187 为例。拼接完 3 张图片后，畸变已经较大，与第四张图片的特征匹配准确度显著降低。



思考：如果我们采用面对畸变、拉伸、旋转更具鲁棒性的特征检测算法，在拼接多张图片时表现将更好。

### 5.3 特征匹配算法

以上使用了角点检测与 hog 梯度直方图来匹配特征点。如果使用 SIFT 特征算法，会发现特征点整体变多，同时对于拉伸、旋转的畸变更不敏感，可用于拼接更多图片。以雪山 Entrance 为例。5.2 中畸变的图片无法完成匹配，但如果使用 SIFT，发现特征点仍然可以较好地进行匹配，如下：



最终效果也提升较大：



可使用 SIFT 对 grail、library、parrington 做更多图片的拼接：

