

# Homework 2

林宇辰 2200013211

October 2023

## 1 Harris Corner Detector

### 1.1 Calculate spatial derivative

这一问实现了一个函数，来求图片的梯度。由于这是最基础的问题，在这里我实现了一个将RGB图片转成灰度图的工具函数。然后使用 *Sobel* 算子作为卷积核对图片做卷积来获得图片的水平梯度和数值梯度。

### 1.2 Calculate Harris response

这一小问的目标是算出角点的相应值。首先依照讲义中的泰勒展开将角点附近的窗口转化为  $M$  矩阵

$$M = \begin{bmatrix} \sum_{x,y \in W} I_x^2 & \sum_{x,y \in W} I_x I_y \\ \sum_{x,y \in W} I_x I_y & \sum_{x,y \in W} I_y^2 \end{bmatrix}$$

于是首先计算出水平梯度、数值梯度对应的逐元素相乘、逐元素平方的矩阵，再遍历矩阵取滑动窗口，并运用高斯卷积核对取出的窗口矩阵做卷积来减少噪声，增加鲁棒性。接着运用响应值的计算公式

$$R = \det(M) - \alpha \text{trace}(M)^2$$

来计算每一点的响应值。

### 1.3 Select candidate corners

这一问的目标是选择角点。对输入的矩阵  $R$ ，首先使用 *threshold* 参数来选择候选角点，如果  $R_{ij} > threshold * R_{max}$ ，就认为这是响应值较大的候选角点，这时选出的是全局较大值。然后使用非极大值抑制算法，针对每个候选角点，在角点周围取一个窗口，检测候选角点是否是窗口中响应值最大的。如果是，那么就选择他作为角点。可视化结果如图 1 所示（对比了 cv2 库的相同算法结果）

## 2 Implement Histogram of Gradients

这一问结合了角点检测和梯度直方图，通过 SIFT 的思想来保证角点特征的旋转不变性，为之后的特征检测打下基础。我实现这个函数的过程可以分为如下几步：

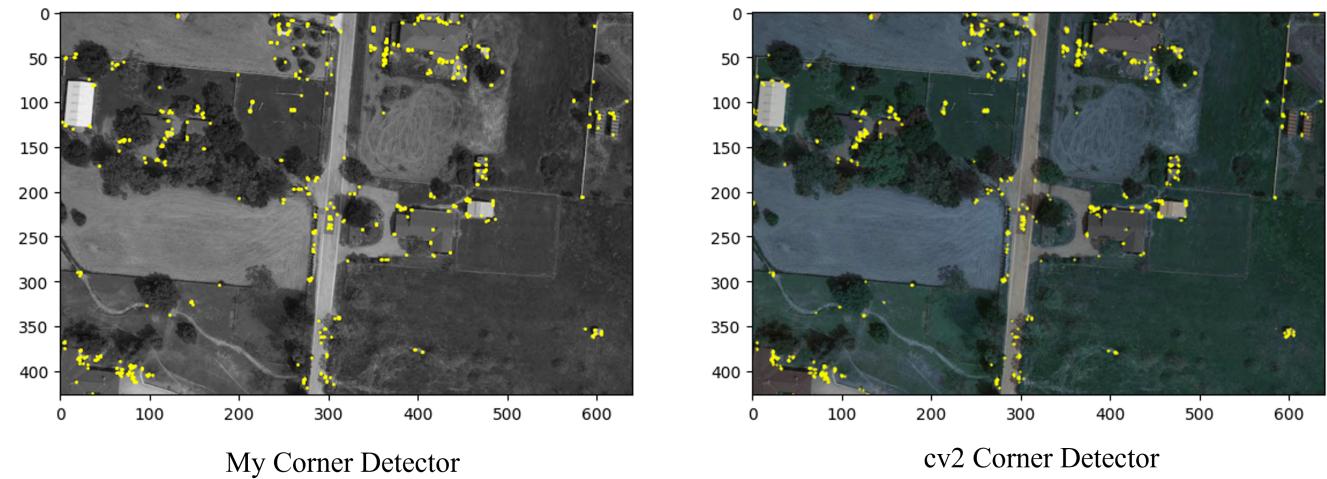


图 1: Harris Corner Detector

## 2.1 数据准备

首先用 task1 中的求梯度算子计算出图片的水平、竖直梯度。然后算出每一点梯度的模长和方向。然后来选择超参，块大小  $blocksize = 4$ ，方向数  $numbins = 8$ 。也就是在每个候选角点的周围取 16 个 block，然后把梯度平面平分成八个方向进行计算。

## 2.2 计算主方向

对每个候选角点，首先统计周围所有 block 的梯度直方图，来选出梯度最大的方向。这里我使用的方法是，遍历所有点，然后把每个点的梯度模长加到对应的方向上，最后选出最大的方向，也就是“principle orientation”。

## 2.3 旋转后计算 feature

在步骤 2 得到主方向之后，我们将每个区域内梯度的角度减去主方向的角度，然后再统计直方图得到特征。特征的维度是  $4 * 4 * 8 = 128$  的，在返回之前对每个 feature 做归一化。

## 3 Local feature matching

这一问基本上就是基于前两问，我的方法是，分别计算出两张图片的角点以及角点对应的旋转不变性特征向量。然后计算两组当中每一个特征向量的欧氏距离（也就是计算  $n^2$  次）。如果两个向量之间的欧氏距离小于  $threshold$ ，就认为他们是匹配的角点。可视化结果如图 2

## 4 Image stitching and Blending

### 4.1 Compute the alignment of image pairs

遍历每一对对映点，构成  $A$  矩阵，针对每一对，将  $A$  扩展两行

$$A_{unit} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1 * x_2 & -y_1 * x_2 & -x_2 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1 * y_2 & -y_1 * y_2 & -y_2 \end{bmatrix}$$

然后对  $A$  做奇异值分解，选取特征值最小的向量，reshape 得到所求的矩阵。

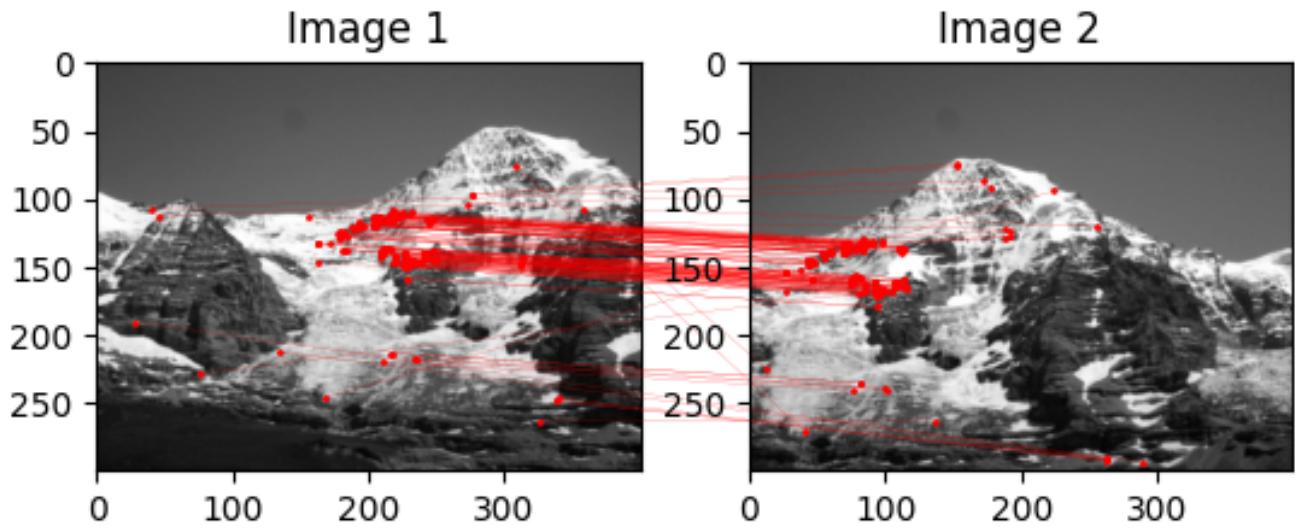


图 2: Local feature matching

#### 4.2 Align the image with RANSAC

进行迭代，每一次迭代，从两组对应点中选出若干组作为取样结果。对取样结果，求出他们的单应矩阵，然后将单应矩阵作用到 pix1 上，计算变换之后 pix1 中的每个元素与 pix2 中对应元素的欧式距离，作为评估拟合程度的标准。

#### 4.3 Stitch and blend the image

先将单应矩阵作用到第一张图的四个角，然后计算出目标图像的大小。先绘制第二张图。然后在遍历整个画布，用逆单应矩阵作用在每个点上，如果作用完的点在第一张点的坐标范围内，那么就用最邻近插值绘制。同样，也用逆单应矩阵来判断是否属于重叠部分，重叠部分采用 alpha 混合算法。特别地，如果第二次绘制时（也就是绘制第一张图），在重合范围内的点为全黑，那么就直接绘制，不用 alpha 混合。这是因为新的图像可能包含了一些原图像没有的信息。结果如图 3 所示。



图 3: Stitch and blend the image

#### 4.4 Generate a panorama

把图片两张两张拼在一起。得到新的一组图片，再两张两张开始拼图，如此迭代。并且注意到只有边界上的特征是我们需要的，只匹配边界上的点。结果如图 4所示



图 4: Panorama

### 5 Analyze

#### 5.1

平移，旋转，平移加旋转其实都是对图片做矩阵变换。根据我的多次测试，拼接平移旋转图片的关键在于特征的匹配，如果图片重合处具有比较明显的角点特征，那么匹配成功的机率很大。如果特征不显著（比如头发丝），匹配效果就很差。图 5是针对旋转（包含平移）的食物的匹配，可以观察到匹配效果较好。

此外，旋转的轴对于图片拼接质量的影响很大。图 5中，相机在拍摄第一张图后，绕垂直于成像平面的轴旋转再拍摄第二张图。可以观察到图像拼接的结果非常优质。而图 7中，相机是绕平行于成像平面的轴旋转再拍摄的，图像融合的质量就比较差（这也是大部分全景图相机移动的方式）。究其原因，绕垂直轴旋转的变换矩阵是正交的，图片并没有被拉伸压缩，因此出现走样的可能性不大。而绕水平轴旋转的变换矩阵不是正交的，在拼接的时候，图片需要经过拉伸压缩然后再融合，这时算法出错的概率就比较大。

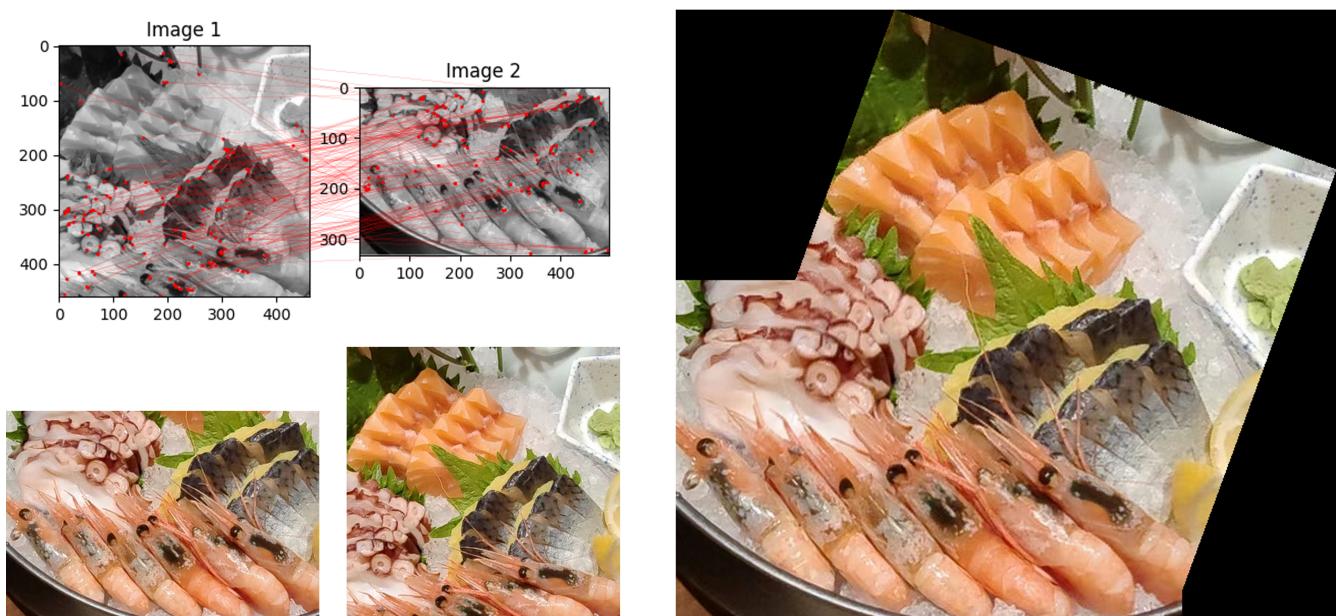


图 5: Rotate

#### 5.2

旋转、平移的幅度过大，会减少前后两张图片之间重叠的特征，大大增加了匹配的难度。容易造成走样。例如图 6



图 6: Aliasing



图 7: Ghost

### 5.3

当有物体移动的时候，会出现两种情况。第一，移动的物体位于相邻图片的重合处，这会造成鬼影。第二，移动的物体不在重合处，这时不会产生鬼影，而是会出现两个一样的物体。第二种情况需要通过增加拍摄量、调整拍摄改进。而第一种情况的产生是由于目前的 alpha 混合算法，只是简单地把前后景混合在一起。要解决鬼影问题需要使用更智能的图片混合算法，比如泊松融合。这可以提高混合的鲁棒性。鬼影现象如图 7 所示。

### 5.4

当同样的人在图片序列的不同图片中出现时，我们可以利用图像融合来制造一些很炫酷的效果。结合上一小节提到的鬼影效果，我们可以制造出很强的动感。见图 8

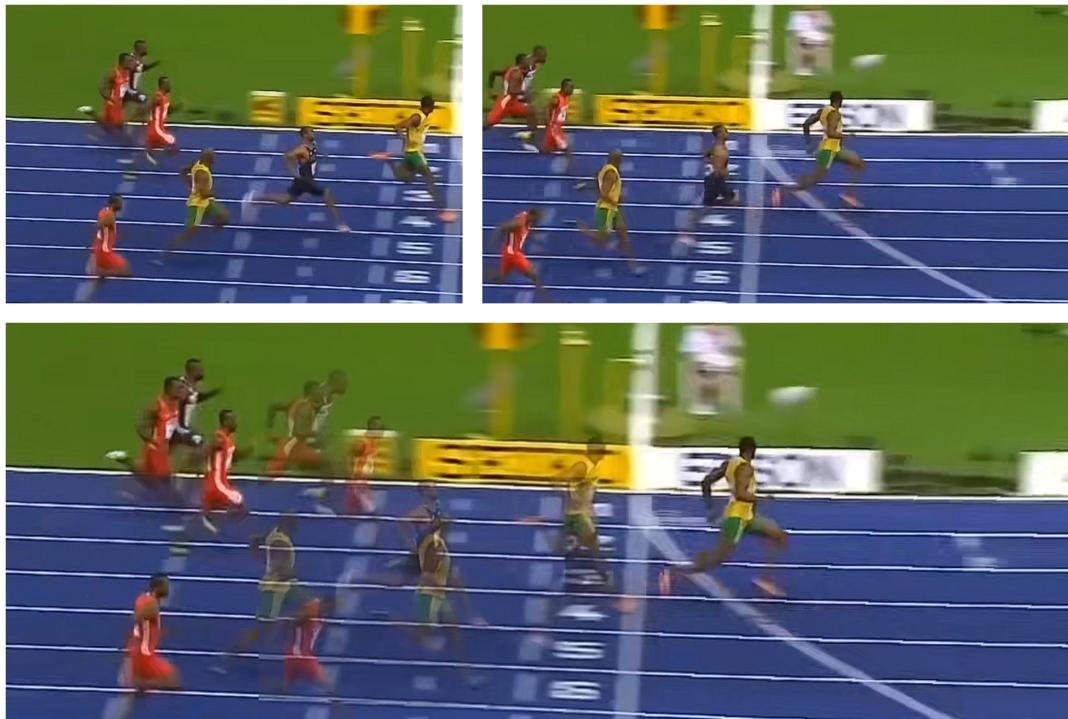


图 8: World Record