Report of Assignment 1

Sinuo Liu

Task1: Harris Corner Detection

1. Implementation

First, the gradients of the image in the x and y directions are calculated. Since the image is already gray, we don't need to grayscale it. I create a Sobel filter and then use scipy.signal.concolve2d to calculate the gradient in the x and y axes on the image. Then I choose to blur the computed gradients using a Gaussian filter with 7x7 kernel size to get rid of the noise. To compute the matrix M, I compute Ix^2, Iy^2, $IxIy$ and perform the weighted sum on each of them. Next I compute Harris response function C directly.

Finally, potential corners are selected using a threshold on the response value. Non-maximum suppression is applied to identify the local maxima in the response map, which are retained as keypoints.

2. Parameters

HARRIS_SIGMA: Low sigma values may cause noise to affect corner detection, resulting in more corners being detected, while high sigma values may smooth out important details, resulting in true corners being missed. Here I choose simga =2.

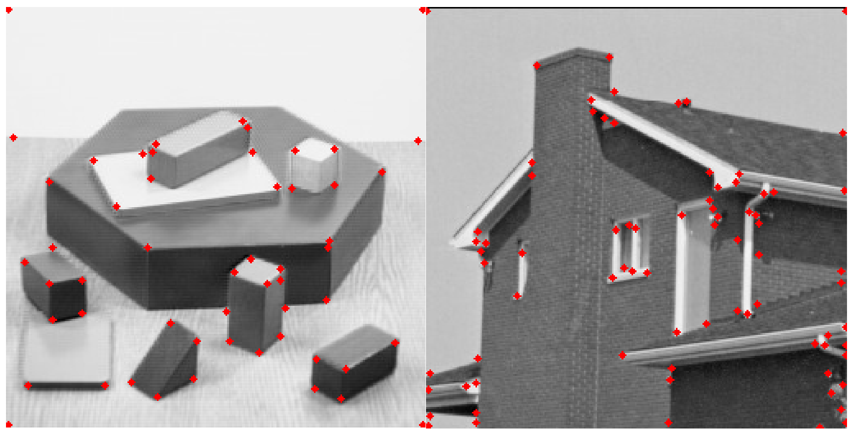
HARRIS_K: Harris response function constant. This parameter influences the sensitivity of the corner detection. Here I set k = 0.04.

HARRIS_THRESH: A high threshold can result in missed corners, while a low threshold might detect false corners. Here I choose threshold = 6*1e-4.

Gaussian kernel size: A larger kernel size averages a larger neighborhood of pixels, leading to a more significant smoothing effect. While this can effectively reduce noise corners detected, it may also blur important details and edges in the image. Here I choose 7x7.

3. Results

The number of keypoints for “blocks_harris”, “house_harris”, “I1_harris”, and “I2_harris” is respectively 54, 77, 361, and 373. Below are the results:



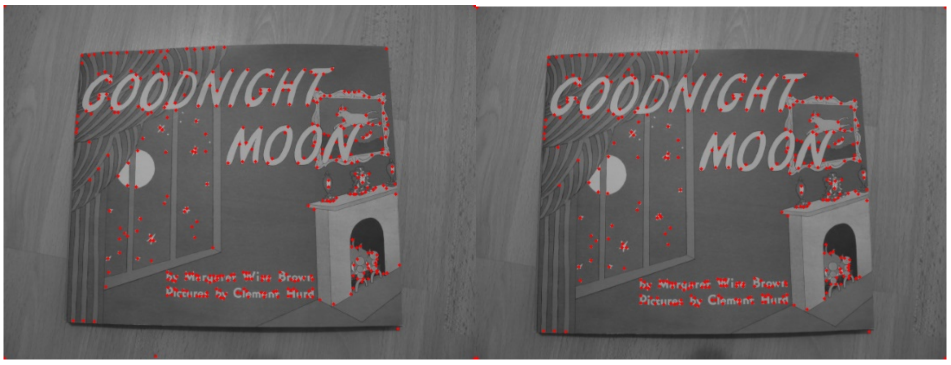


Fig. 1 Harris Corner Detection Results

4. Difficulties

Adjusting the parameters is a bit difficult, the initial parameters do not work well and there are a lot of points that are not corners detected. After increasing the Gaussian sigma and threshold there are fewer points but there are a lot of corners that are not detected. It’s hard to find the balance. Also for some reason, the four corners of the image are also being detected as corner points, which confuses me.

Task2: Description & Matching

1. Implementation

(i) Filtering Keypoints

The filter_keypoints function filters out keypoints that are too close to the edges of the image. The margin is determined based on the patch_size, which represents the size of the region around each keypoint.

(ii) Matching

First I use the ssd function to compute the squared distances between two sets of descriptors. Three different matching techniques are implemented:

- One-way nearest neighbors matching: This method finds the closest descriptor in the second image for each feature in the first image.
- Mutual nearest neighbors matching: In this approach, a match is only considered valid if a keypoint in the first image matches to a keypoint in the second image, and vice versa. This requirement helps to ensure more reliable matches compared to the one-way method.
- Ratio test matching: Based on Lowe's ratio test, compares the distance to the best match with that of the second-best match. Only matches with a sufficiently small ratio are retained. (Here I set threshold = 0.5)

One-way nearest neighbor matching is fast but can be less accurate. On the other hand, mutual nearest neighbor matching enhances accuracy by enforcing reciprocal matches, although it doesn’t completely resolve ambiguities. Ratio test matching tends to offer the best accuracy by directly comparing the best and second-best matches.

2. Results

The number of matches using “One-way nearest neighbor matching”, “Mutual nearest neighbor matching” and “Ratio test matching” is respectively 357, 293 and 240. Below are the results:

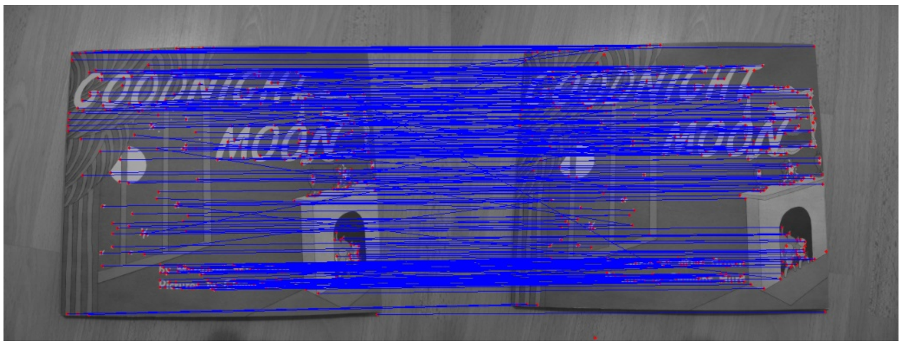


Fig. 2 One way nearest neighbors matching

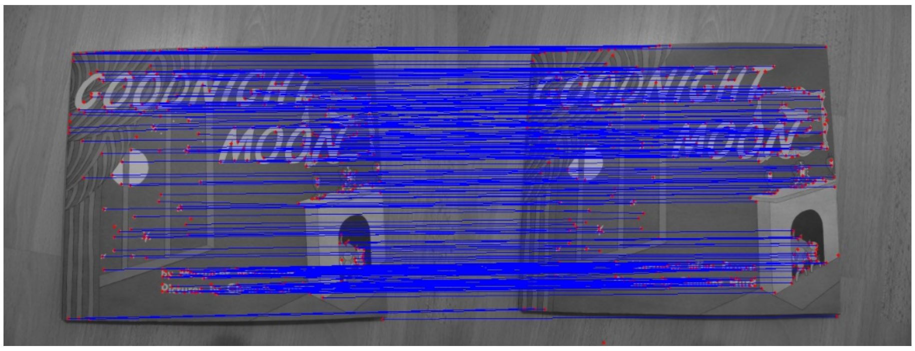


Fig. 3 Mutual nearest neighbors matching

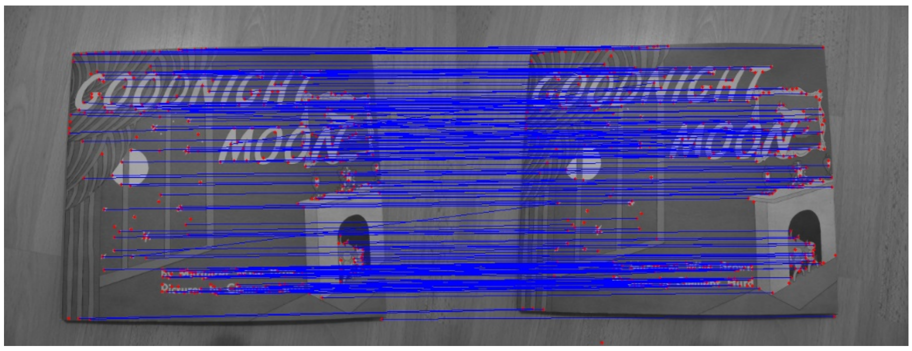


Fig. 4 Ratio test matching

It is evident that the one-way nearest neighbors matching method yields the poorest performance. This approach produces many intersecting lines, which suggests that the matches between the descriptors of the two graphs are suboptimal. In contrast, the ratio test matching method demonstrates improved results, with only one intersecting line present. Finally, the mutual nearest neighbors matching method performs best, as it shows no intersecting lines; instead, it features entirely parallel lines, indicating that each descriptor is effectively matched.