ETH ZURICH, D-INFK

Computer Vision Exercise 1: Feature Extraction

Xingchang Huang

October 2, 2017

1 HARRIS CORNER DETECTION

The objective of exercise 1 is to implement our own feature detector(e.g., Harris Corner Detector) and descriptor in MATLAB, in order to find the matching of features or points between two images.

1.1 RESIZING, PADDING, BLURRING

In the assignment slide, the image blurring part is behind the intensity gradient computing part but I found that it would be useless after calculating the gradients. Therefore, I put it in the first step with 3x3 gaussian kernel ($\sigma = 0.5$) and filtered both images. Also, I have resized the image to 224x224 size to avoid intensive computation. This should be a more reasonable size used in the input of many deep neural networks and still contain enough information for feature extraction. Finally, I have done zero padding for both images in order to avoid exceeding the boundary while calculating the intensity gradients in the next step.

1.2 INTENSITY GRADIENTS

Next, I calculate the intensity gradients for every pixel using the simplest formula (e.g., x-direction):

$$I_x = \frac{p_{x+1,y} - p_{x-1,y}}{2} \tag{1.1}$$

Since the images have been gray images, p mean the grey-scale value of pixels.

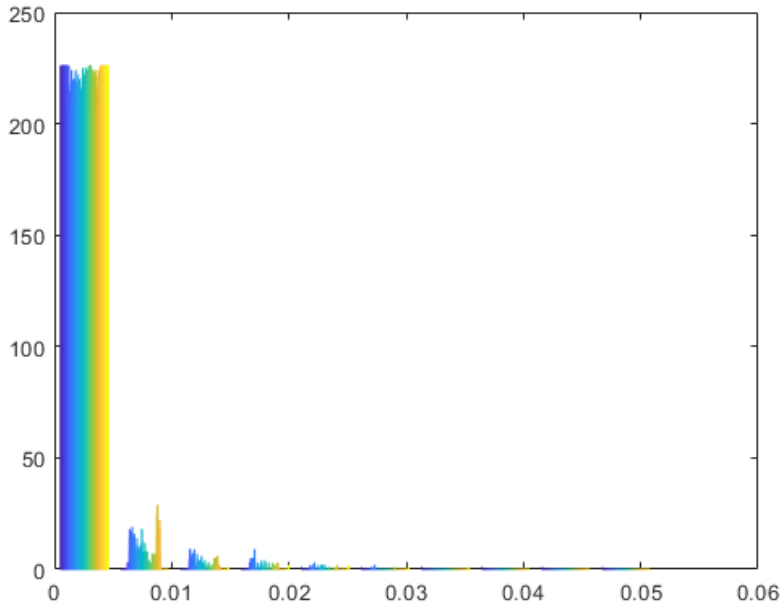

Figure 1.1: Histogram of Harris Matrix

1.3 Harris Response

After calculating the gradients for each pixel, I can therefore calculate the Harris Matrix of each pixel by summing up the matrix of its adjacent 8 pixels and then calculate the Harris Response K by determinant and trace following the equations in slides.

1.4 Threshold and Non-maximum Suppression

Next is to find a suitable threshold to choose the candidate Harris Corner Points. In the same time, I consider the adjacent points to see whether this point has the maximum value. If not, it would not be considered to be a candidate point. For implementation, I calculate the maximum value in a 3x3 matrix and find whether that point is equal to the maximum value. More importantly, the value of threshold can be determined by the histogram of Harris Matrix as shown in Figure 1.1. Though the histogram may become fluctuated, it is still preferable to resize the image to a smaller size due to effective computation.

Therefore, the threshold here is set to be around 0.001. And the result of detected corners are shown in Figure 1.2. If the Non-maximum Suppression is not used, the example result will be in Figure 1.3, which seems to detect much more points than the corner points.

Figure 1.2: Detected Harris Corners in both images.

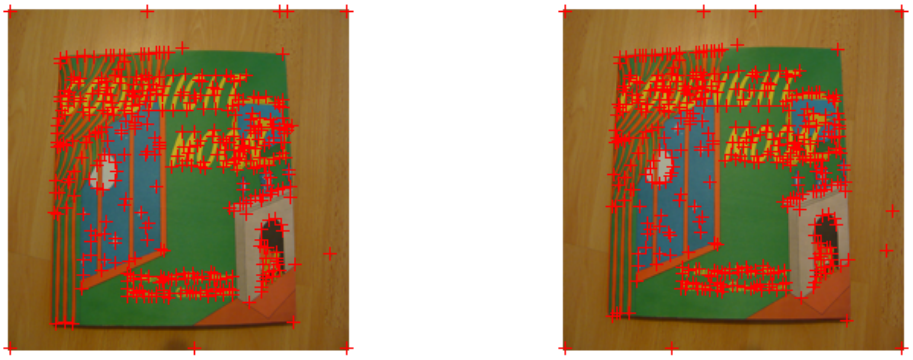

Figure 1.3: Detected Harris Corners without using Non-maximum Suppression.

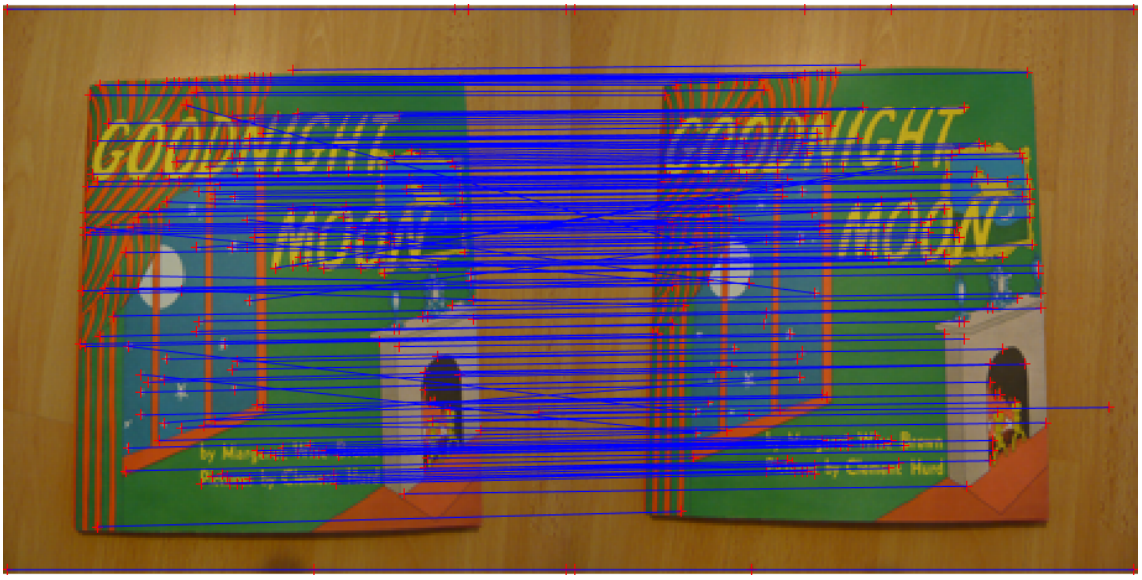

Figure 3.1: My Feature Matching

2 Patch Descriptor Generation

Here, I just need to focus on the detected corner points. For each point, I found a 9x9 matrix for this point where it located at the center. This 9x9 matrix is then reshaped into a feature vector with 81 length, which is considered to be patch descriptor.

3 Descriptors Matching

When it comes to descriptors matching, SSD (sum of squared differences) is calculated between descriptors. I need to scan through all descriptors (also known as feature vector) in both images and calculate the SSD using element-wise subtraction and square easily in MATLAB. After that, another threshold should be set here to find the similar descriptors and the most similar one under those similar descriptors. The threshold here is set to be 0.1. After matching the indices of points, I can draw the result in MATLAB as Figure 3.1. As shown in the figure, some corner points do not fit correctly shown as slashes. And some points may have more than one corresponding point.

4 Comparison with SIFT

After running SIFT in VLFeat package on the same pair of images, I got the result as the Figure 4.1. I have used vl_sift function to extract the feature points and descriptors. And then I used the vl_ubcmatch function to the matched points. The function vl_ubcmatch also contains a parameter THRESH, whose default number is 1.5. Thus, default SIFT found less corner points than my implementation. After trying to change the THRESH to 0.1, I got another result of

Figure 4.1: SIFT Feature Matching

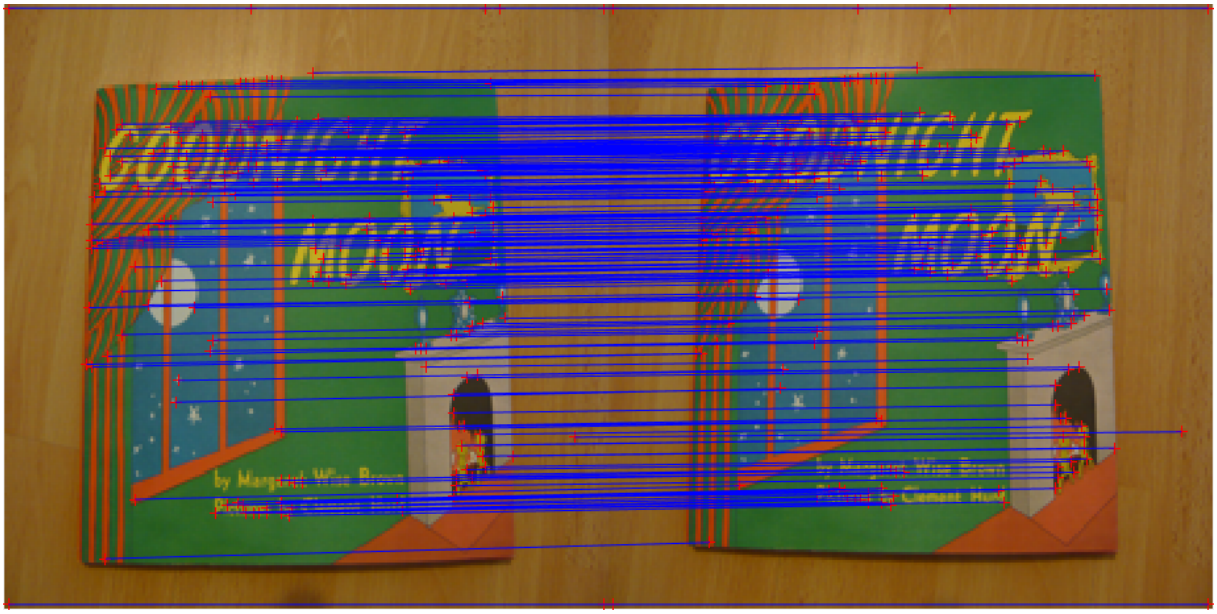

Figure 5.1: My Implementation of SIFT Feature Descriptor and Matching.

SIFT method shown in Figure 4.1.

Basically, SIFT still has some mistaken matching. I think some mismatch is difficult to avoid because of some points are quite similar shown in the images but actually they do not correspond to each other.

5 Bonus: My Implementation of SIFT Descriptor

In this section, I used detected harris corners for my implementation of SIFT descriptor. As there is no requirement for the scale and rotation invariances, I just find a gradient histogram for each corner point. More specifically, for each point, I scan through its 16x16 neighborhood. Then for each 16x16 region, I split it into 16 4x4 regions and calculated a histogram for that 4x4 region, which has 8 bins for each histogram, where each bin represents the strength of that gradient direction. Finally, I got a $4 \times 4 \times 8 = 128$-dimensional feature vector as the feature descriptor of that point. It is also noted that the histogram should be normalized and the threshold of SSD should be changed to around 0.05. Here is my result of my SIFT descriptor in Figure 5.1.

For comparison, my method seems better, which may result from a larger context while constructing the feature descriptor.