Lab 02: Local Features

Harris Corner Detection and Patch Description & Matching

Orion Junkins

263-5902-00L Computer Vision

Marc Pollefeys, Siyu Tang & Fisher Yu

October 13, 2023

Introduction

This lab consists of two stages. First, well-localizable key points are found using a Harris detector. Second, local patches for all detected key points are isolated, and corresponding pairs of key points are found across two images using three different matching algorithms. All functionality is implemented in Python using select utilities from OpenCV, Scipy, and Numpy.

Part 1: Harris Corner Detection

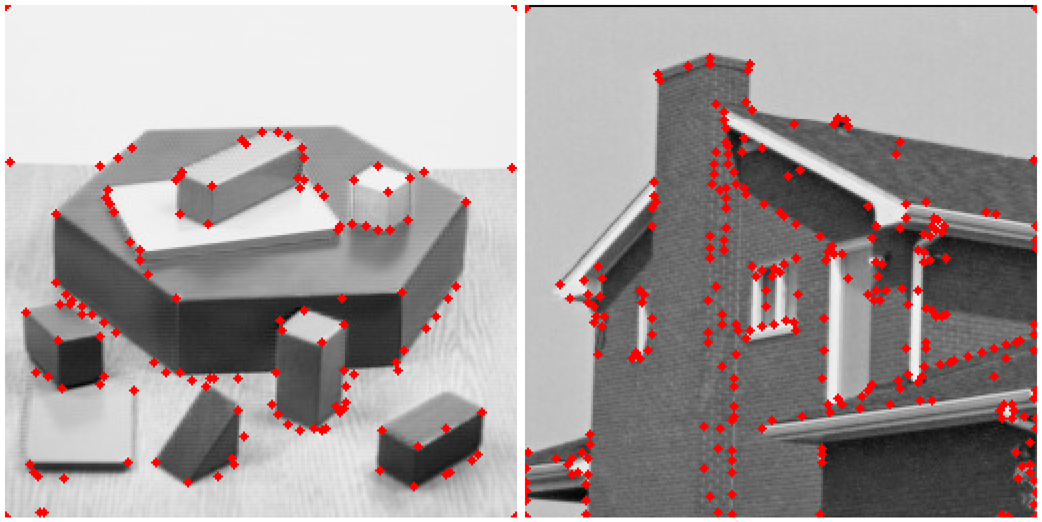
Procedure

Given a grayscale image, Harris detection is performed in the following steps.

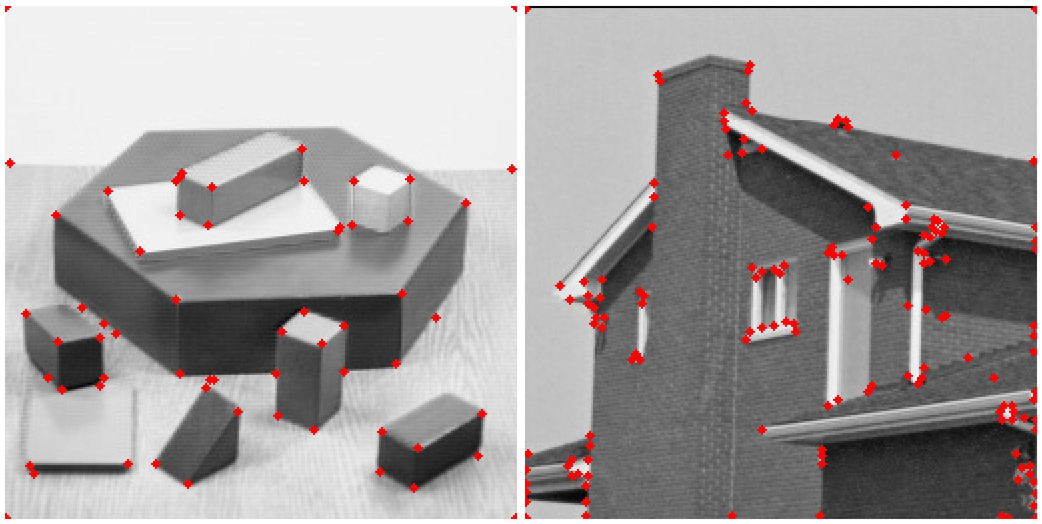
1. Define the following kernels K_x and K_y.
$$K_x = [-1 \quad 0 \quad 1]$$
$$K_y = [-1 \quad 0 \quad 1]^T$$
2. Compute image gradients. Find the gradients in the x and y directions, I_x and I_y, using a 2d convolution with K_x and K_y respectively.
$$I_x = I * K_x$$
$$I_y = I * K_y$$
3. Compute the products needed for calculating Harris Response I_x^2, I_y^2, and $I_x I_y$.
4. Blur the computed gradients using OpenCV's *GaussianBlur()* function. Let $G(I, \sigma)$ represent the Gaussian blurring of I with sigma σ. This yields $G(I_x^2, \sigma)$, $G(I_y^2, \sigma)$, $G(I_x I_y, \sigma)$. Sigma is a parameter that can be tuned, but the default value is 1.
5. Consider gradients as a 2 x 2 matrix *M* in the following form:
$$M = \begin{bmatrix} G(I_x^2, \sigma) & G(I_x I_y, \sigma) \\ G(I_x I_y, \sigma) & G(I_y^2, \sigma) \end{bmatrix}$$
6. Note that all four elements of *M* are matrices with dimensions corresponding to the original image dimensions. Calculate the determinant $det(M)$ and trace $Trace(M)$ across all points. Use these values to calculate the Harris Response for every point using the following formula:
$$R = det(M) - k[Trace(M)]^2$$
where k is a tunable parameter with a default value of 0.05. These yields a 2D array, *R*, of response values with dimensions identical to the original input image.

7. Perform non-maximum suppression by filtering out any response value not the max of its 3x3 neighborhood. This is performed using scipy.ndimage.maximum_filter().
8. Identify remaining response values above some threshold, and track an array of corresponding coordinates. This threshold is also tunable, with a default value of 1e-5.
9. Return the array of identified coordinates and the 2D array containing raw response values.

Example Outputs



Example outputs with a threshold of 1e-5



Example outputs with a threshold of 5e-5

Remarks

As visible in the sample images, this procedure is effective at identifying corners and points of interest.

Some points may be considered False Positives if compared to a human-generated ground truth of what makes a "corner", especially in the lower thresholded example. However, these false positives are generally reasonable, given what the algorithm is designed to detect. They still correspond to areas with rapid changes in pixel intensity in both the X and Y directions. For the purposes of patch matching across images, many of these false positives are still potentially useful points of interest even if they are not necessarily “corners” by a human definition.

Some unidentified points may be considered false negatives as well, but these mistakes are also generally reasonable. For example, very obtuse angled corners with relatively low contrast (such as the back two corners of the base hexagon block in the blocks image) are not identified. But, due to the relatively low gradient at these points, it is logical for them to have been missed. This limitation could potentially be addressed by introducing image preprocessing steps (i.e. boosting contrast).

Part 2: Patch Description & Matching

Procedure

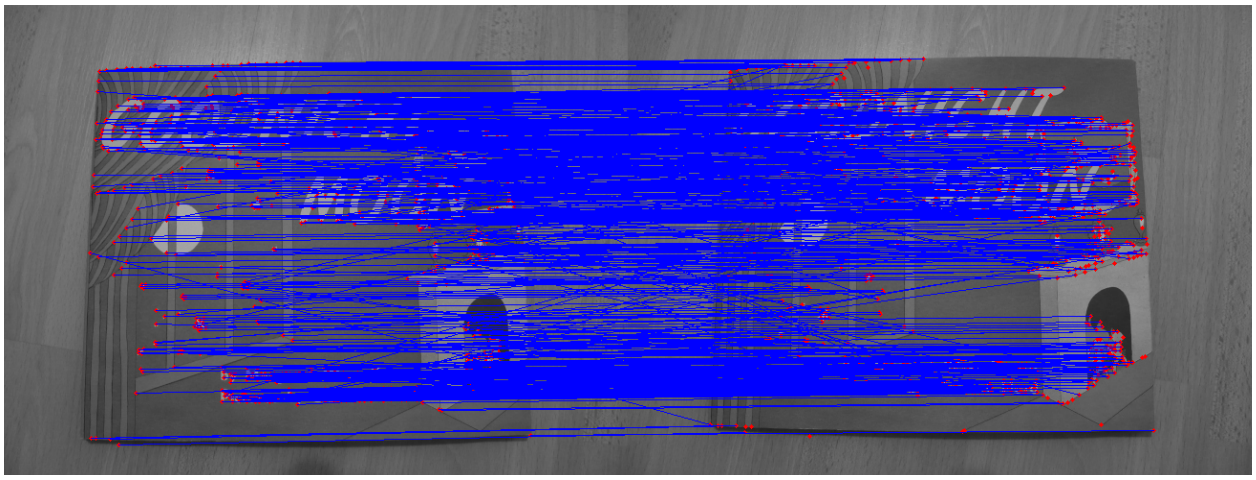
Provided two photos of the same scene, key points are identified using the procedure from Part 1 above. This yields two lists of key points. Key points are correlated across images in the following steps.

1. Extract descriptors (patches centered at each key point)
    a. Filter out key points less than the needed offset from the image's bounds. The needed offset is calculated as the floor of half the patch size. By default, the patch size is 9 x 9, so any key points with a distance to an image boundary of less than or equal to four were removed.
    b. Extract the patches centered at each remaining key point using the provided utility function. This yields two lists of patches, where each patch is a vector of pixel values.
2. Calculate the Sum of Square Differences (SSD) between every possible pair of patches. SSD between patches p and q is calculated using the following formula:

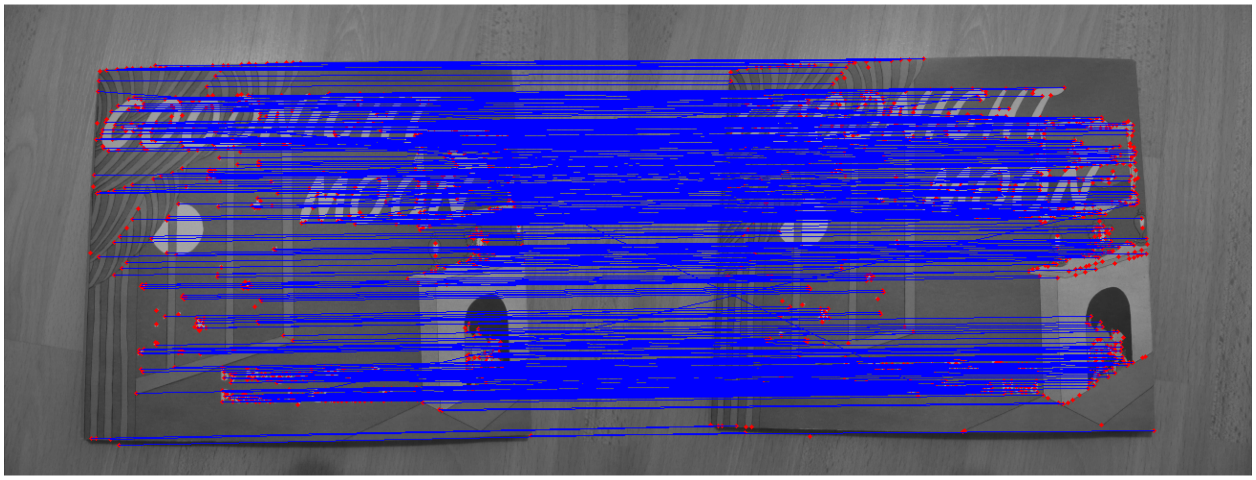
$$SSD(p, q) = \sum_i (p_i - q_i)^2$$

   where i represents the index of the individual pixel values in each patch vector. Calculating SSD for all pairs of patches yields a 2D array of similarity scores.
3. Determine which patches to label as “corresponding” using the computed SSD scores. Correspondence is determined using three different strategies depending on the mode provided by the user. These three approaches are defined as follows.
    1. Naive “one-way” matching. Find the minimum SSD value for each key point in the first list (along a single axis of the SSD array) and label those points as corresponding.
    2. “Mutual” matching. Perform naive one-way matching on both lists of key points (along both axes of the SSD matrix) and take their intersection. I.e., only label two points as corresponding if they are each other’s one-way matches.
    3. “Ratio test” matching. Perform a one-way match, but also consider the second lowest SSD for each key point. For each, compute the ratio between the first and second lowest. Only keep the key points for which this ratio is below some threshold. By default, this threshold is set to 0.5.

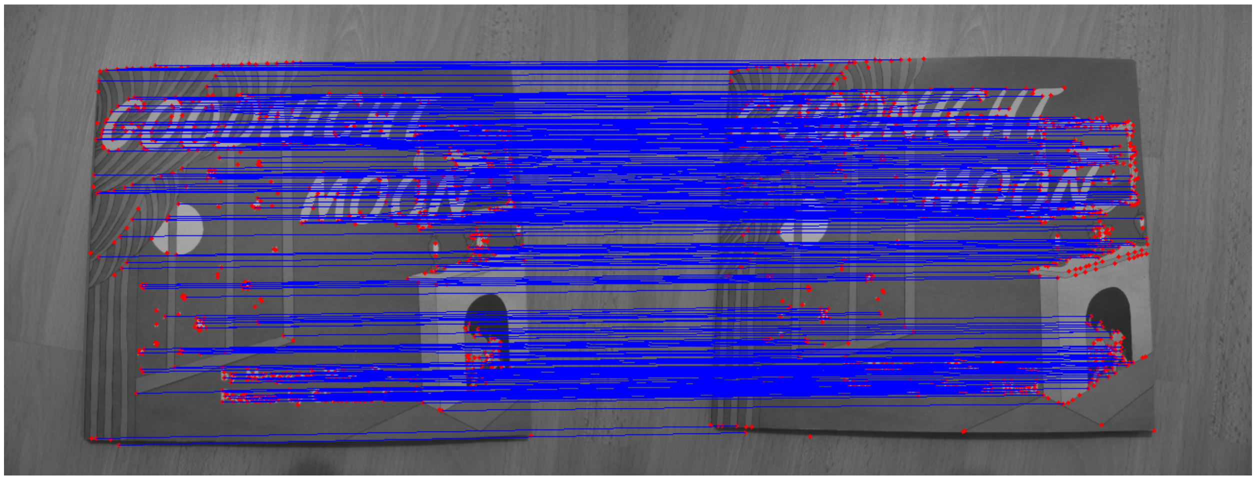
Example Outputs



Example output of “one-way” matching with 761 matches



Example output of “mutual” matching with 552 matches



Example output for “ratio test” matching with 411 matches

Remarks

While we do not have a ground truth to evaluate against, visual inspection shows that keypoint matching was largely effective. Obvious/isolated points can be seen as connected between the two images.

Also, because the two pictures are taken from the roughly same position when placing output images side by side, ‘correct’ matches should be roughly horizontal lines. While not a comprehensive evaluation metric, this observation allows a comparison of the three matching approaches.

In the “one-way” example, numerous diagonal lines indicate false positive correlations between key points. In the “mutual” example, the number of diagonal lines is drastically reduced, indicating a lower false positive count. In the final “ratio test” example, all lines are horizontal indicating the lowest false positive count of all. This indicates that the “ratio test” matching is the superior approach.

However, this reduction of false positives does come at the cost of an increase in false negatives. This increase can be inferred from the drastic decrease in the overall number of matches (761 to 411 from “one-way” to “ratio-test”). A quantified evaluation against an authoritative ground truth, as well as consideration of the constraints of the intended problem domain, would be critical before making a decision between these approaches in the real world.