Igor Martinelli: Assignment 1 Report

maigor@ethz.ch, 19.916.048.

1 extract_harris.py

The provided Python function, extract_harris.py, implements the Harris Corner Detector algorithm for detecting corners or key points in grayscale images. It begins by converting the input image to a floating-point format, followed by computing the image gradients in the x and y directions. Gaussian smoothing is applied to the squared gradients and the product of gradients to reduce noise. Next, the local auto-correlation matrix elements are computed, and the Harris response function (corner strength) is calculated using these elements. Corners are then detected by applying a threshold to the corner strength and performing non-maximum suppression. The function returns a set of corner coordinates and a corresponding corner strength map.

2 extract_descriptors.py

The extract_descriptors.py script contains two key functions for local image descriptor extraction. The first function, filter_keypoints(img, keypoints, patch_size=9), takes a grayscale image img, a numpy array of keypoints keypoints, and an optional patch size parameter, and it filters the keypoints to retain only those sufficiently distant from the image edges. The second function, extract_patches(img, keypoints, patch_size=9), extracts local image patches centered around the keypoints, converting the image to a floating-point format, calculating patch offsets, and assembling the patch descriptors into a numpy array, which is returned as the output. These functions collectively enable the extraction of local descriptors from keypoints in a grayscale image for further analysis or feature matching tasks.

3 match_descriptors.py

The provided Python script contains two functions for matching image descriptors. The ‘ssd(desc1, desc2)‘ function calculates the Sum of Squared Differences (SSD) between two sets of descriptors, where ‘desc1‘ and ‘desc2‘ are numpy arrays representing descriptor vectors for the first and second images, respectively. The result is a numpy array ‘distances‘ storing the squared distances between all pairs of descriptors. The ‘match_descriptors(desc1,

desc2, method="one_way", ratio_thresh=0.5)‘ function matches descriptors based on various methods specified by the ‘method‘ parameter. It first computes the SSD distances using the ‘ssd‘ function and then matches descriptors using one of three methods: "one_way," "mutual," or "ratio." The function returns a numpy array ‘matches‘ containing indices that represent the matches between descriptors, with the format depending on the chosen matching method.

4 Results

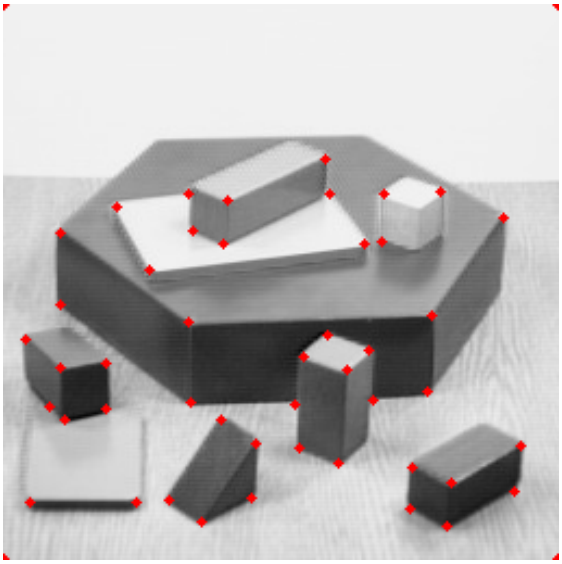

Figure 1: Harris corner detection for first image

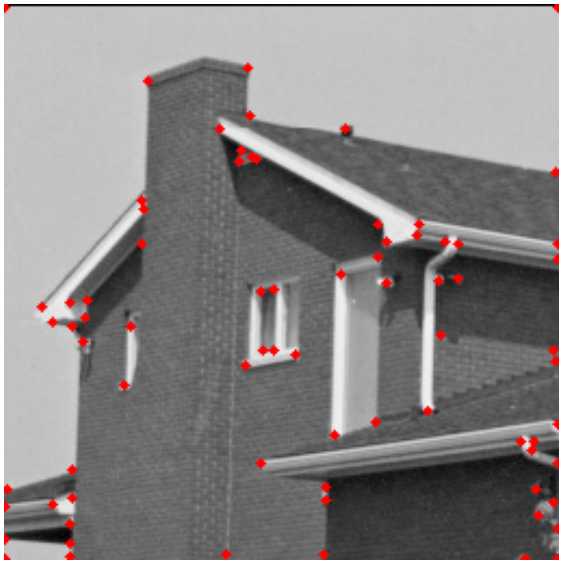

Figure 2: Harris corner detection for second image

We see in the above figures the result of our implementation of extract_harris.py on the two images. We notice that some corners are ignored and that the corners of the image are also detected, this will be later resolved in extract_descriptors.py. Overall it looks like a pretty good corner detection, although not perfect.

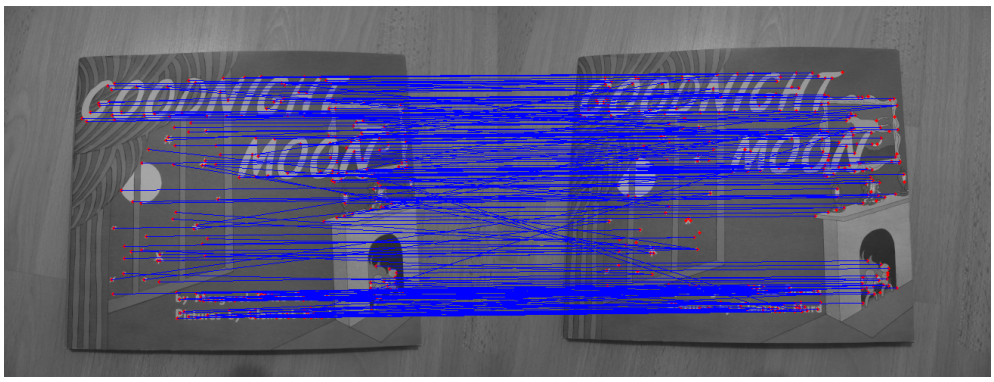

Figure 3: One-way matching of image descriptors

In Figure 3 we see the result of the descriptor matching obtained through the one-way method. With this method we notice that multiple descriptors of the left image are mapped

to the same descriptor on the right image. This is obviously inaccurate.

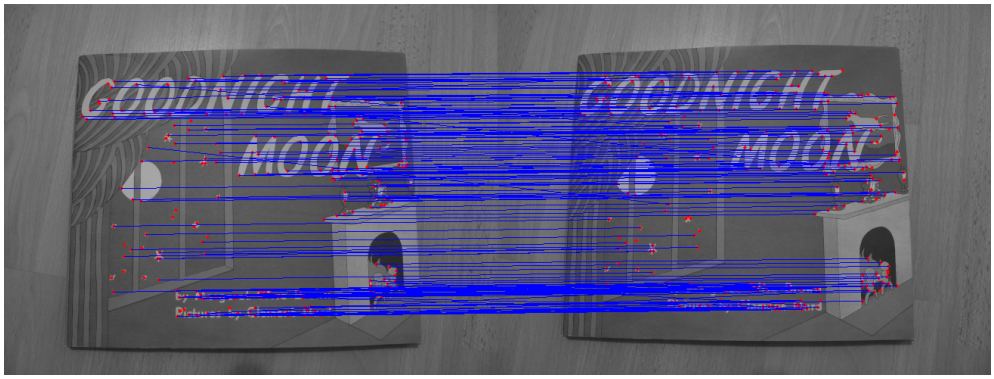

Figure 4: Mutual matching of image descriptors

In Figure 4 we don't see the problem of the one-way matching happening, but some descriptors on the left image are not matched to any descriptors on the right image. With this method there is a total of 181 matches.

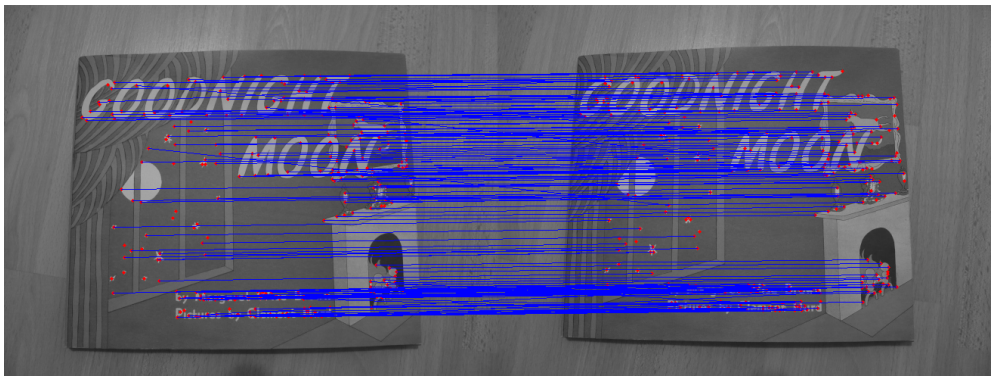

Figure 5: Mutual matching of image descriptors

In Figure 5 the total number of matches is 132, hence a lot less than the mutual method. It is hard to establish which one of the two methods is more accurate.