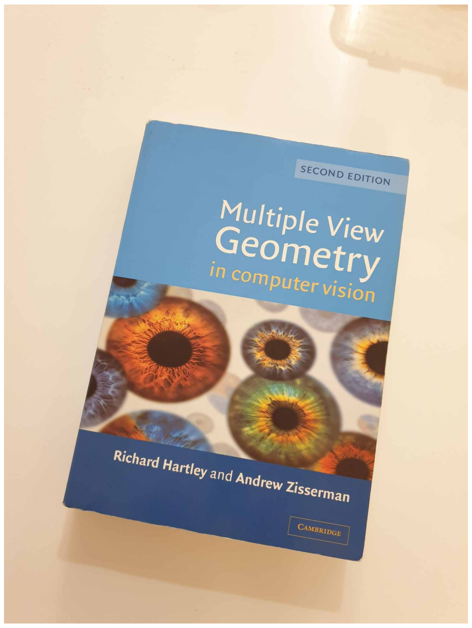
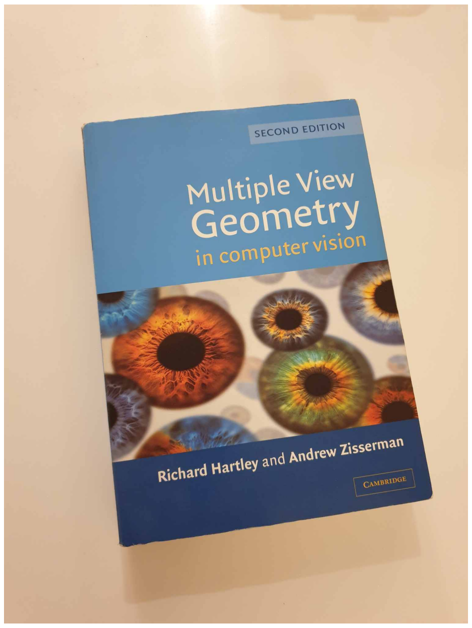
Harris Corner Detection

In this report we illustate harris corner detection algorithm and corresponding point matching using images in Figure ref



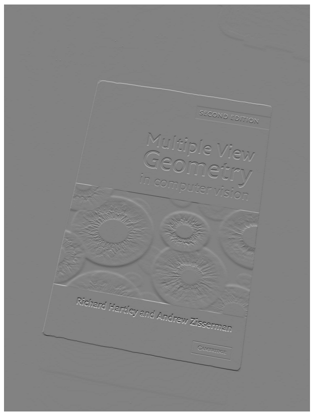
(a) Image 1

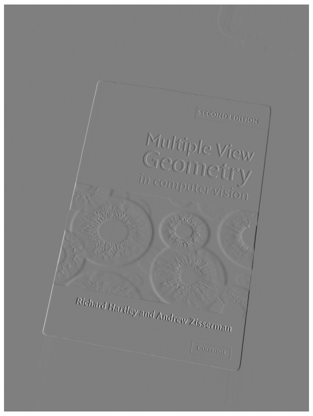


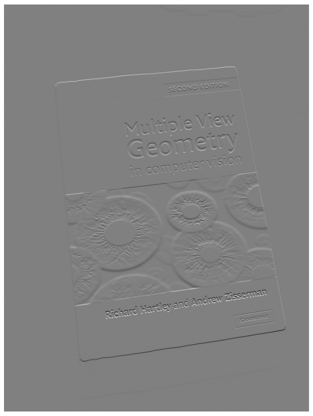
(b) Image 2

Figure 1: Images used for harris corner detection

Now we need to find the edges of images in order to find harris score for them. To do so, we filter images with the derivative of gaussian filter in order to get smooth edges.


(a) $\frac{\partial I_1}{\partial x}$


(b) $\frac{\partial I_1}{\partial y}$


(c) $\frac{\partial I_2}{\partial x}$

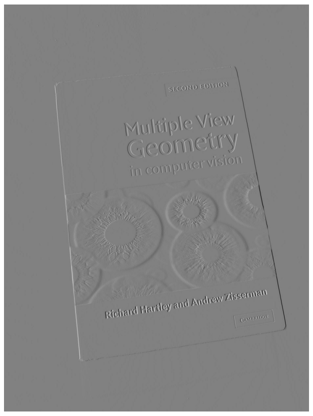

(d) $\frac{\partial I_2}{\partial y}$

Figure 2: Derivatives of images

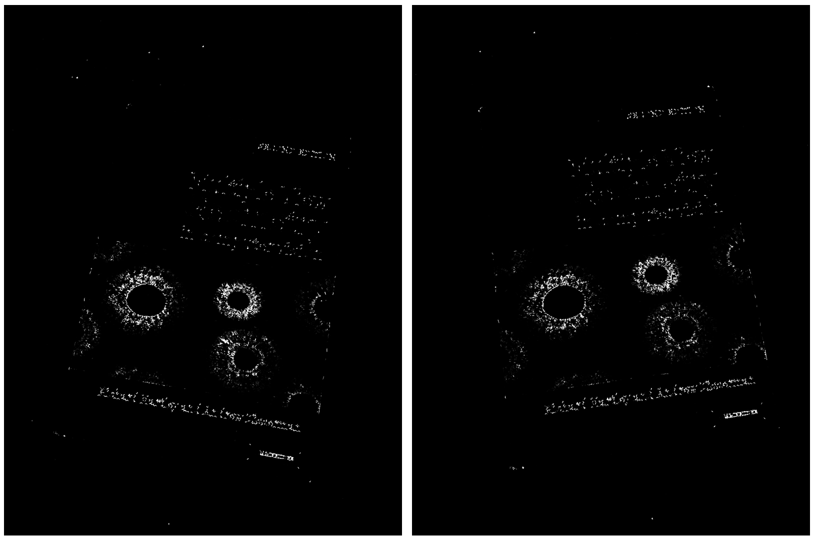
Now we have to compute the matrix M defined below in order to find the harris score:

$$M = \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix}$$

and harris score R is computed as below:

$$R = det(M) - k \times trace(M)^2$$

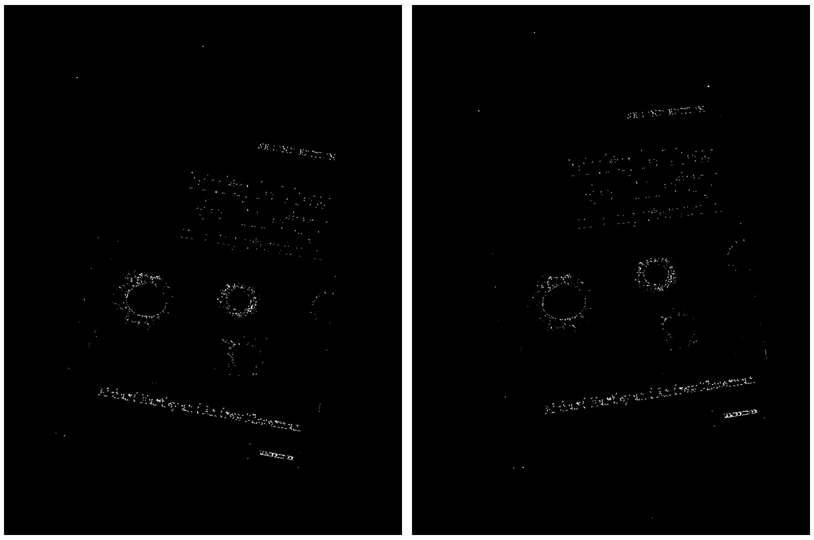
Where k is a hyperparameter needed to be tuned based on the defined problem. In Figure 3 the harris score is computed for both images with $k = 0.1$:



(a) R_1 (b) R_2

Figure 3: Harris Score ($k = 0.1$)

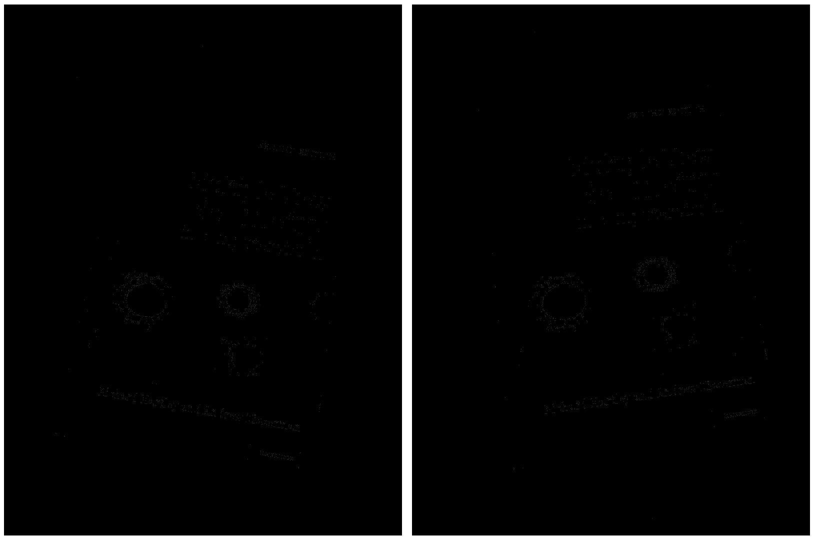
In order to igonore unwanted corners detected because of noise, we apply a hard threshold on harris score and ignore values below the threshold. In Figure 4 the thresholded harris scored are shown:



(a) R_1 (b) R_2

Figure 4: Thresholded Harris Score ($threshold = 1000$)

To sparsify the harris score, non-maximum suppression is used. In Figure 5 we apply a 5×5 window and the harris corners are shown:



(a) R_1 (b) R_2

Figure 5: 5×5 Non maximum suppression Harris Score

Corners detected based on harris algorithm is shown in Figure 6.

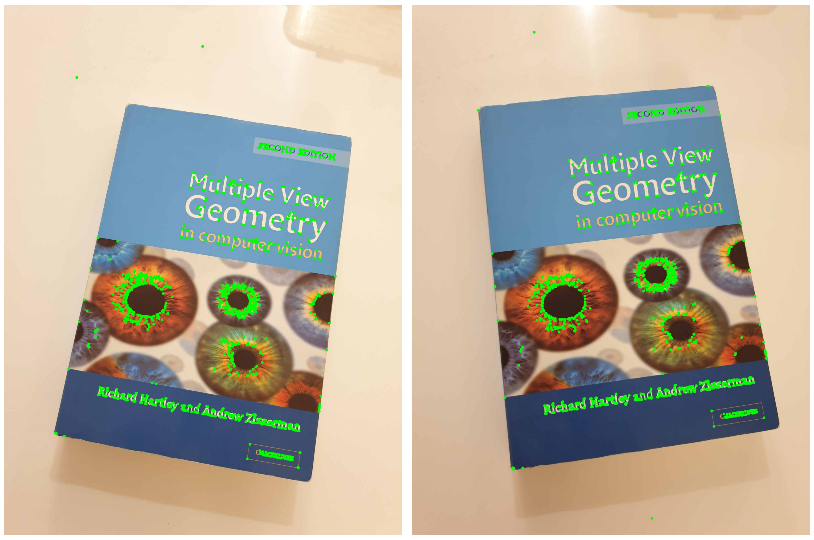

SECOND EDITION
Multiple View Geometry in computer vision
Richard Hartley and Andrew Zisserman

Figure 6: detected Corners

For describing each corner, we concatenate all of the pixel intensity in all three channles within each corner's 11×11 window and consider it as a features vector. Now we have to find corner in image2 corresponding to corner in image1. To do so we compare each corner feature vector in image1 with all of the corners in image2 by their euclidean distance. If the ratio of the euclidean distance between the nearest neighbour and the second nearest neighbor is less than a threshold, we pick that corner and if its not, we ignore it. To be more precise if C_1 denotes a corner in image1 and $nearest(C_1)$, $secondnearest(C_1)$ are the nearest and the second nearest corners to C_1

detected in Image2, then we keep C_1 if:

$$\frac{\mathbf{dist}(C_1, nearest(C_1))}{\mathbf{dist}(C_1, secondnearest(C_1))} < Threshold$$

The same procedure is done for Image2. Now two sets of candidate corners are detected in both images. The intersection of these two sets is the final set consisting of corners and their corresponding ones in the other image. In Figure 7 the corners are matched in both images with $Threshold = 0.6$:

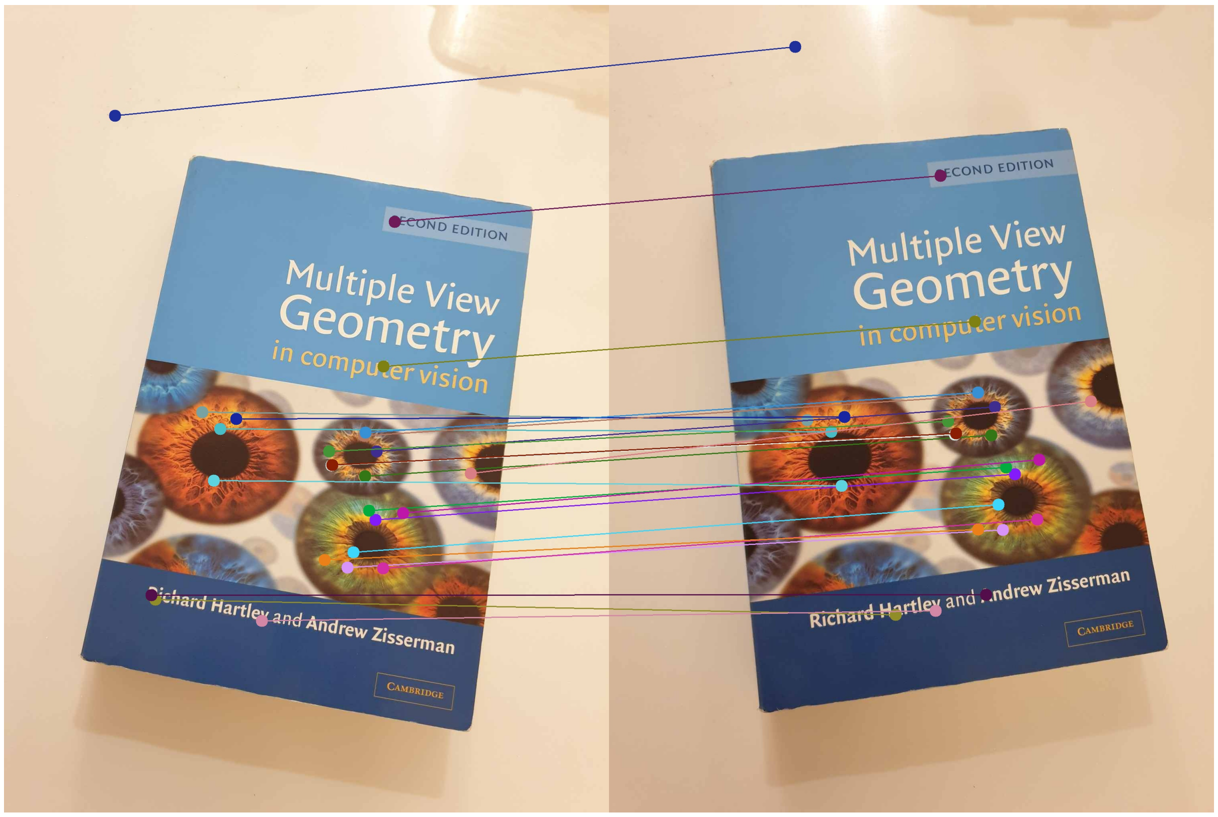


Figure 7: Matched Corners

As we can see most of the corners are matched properly but there are some false matching too which is inevitable and also negligable.