Report: Local Features Matching

Rishabh Singh

October 11, 2019

1 Detection

1.1 Image Gradients

The image gradients in both X and Y directions were computed using the $conv2$ function with $0.5 * [1, 0, -1]$ as the convolution vector. The results on the template images are shown in Fig.1. As evident through the results, the vertical edges are more prominent in the gradient along X-axis while the horizontal edges are highlighted in the gradient calculated along Y-direction.

Figure 1: Gradient calculation along X(left) and Y(right) direction in both images.

1.2 Harris Response Function

In order to compute the Harris response function efficiently, instead of looping through every single pixel in each image, the matrix structure of the inputs and the linearity involved in the formulation were utilised. The local autocorrelation matrix M_p at pixel p is defined as:

$$M_p = \sum_{p' \epsilon N(p)} w_{p'} \begin{bmatrix} I_x(p')^2 & I_x(p')I_y(p') \\ I_x(p')I_y(p') & I_y(p')^2 \end{bmatrix}$$

for all p' in the neighbourhood $N(p)$. This can be re-structured as:

$$M_p = \begin{bmatrix} \sum_{p' \epsilon N(p)} w_{p'} I_x(p')^2 & \sum_{p' \epsilon N(p)} w_{p'} I_x(p') I_y(p') \\ \sum_{p' \epsilon N(p)} w_{p'} I_x(p') I_y(p') & \sum_{p' \epsilon N(p)} w_{p'} I_y(p')^2 \end{bmatrix} = \begin{bmatrix} GI_x^2 & GI_{xy} \\ GI_{xy} & GI_y^2 \end{bmatrix}$$

The elements are similar to applying Gaussian filter to the values of gradient products at each pixel location. Thus, for more efficient computation, matrices I_x^2, I_y^2 and $I_x I_y$ were computed. These were filtered with a Gaussian kernel having standard deviation σ using $imgaussfilt()$. Let the results be denoted as GI_x^2, GI_y^2 and GI_{xy}. The Harris response function at pixel p becomes:

$$C_p = GI_x^2 * GI_y^2 - (GI_{xy})^2 - k * (GI_x^2 + GI_y^2)^2$$

The pixel p is considered to be an interest point if the magnitude of C_p is above certain threshold, $thresh$ and if it is maximum among the 8-connectivity surrounding it.(see *Appendix 3.1* for implementation details)

1.3 Effect of parameters

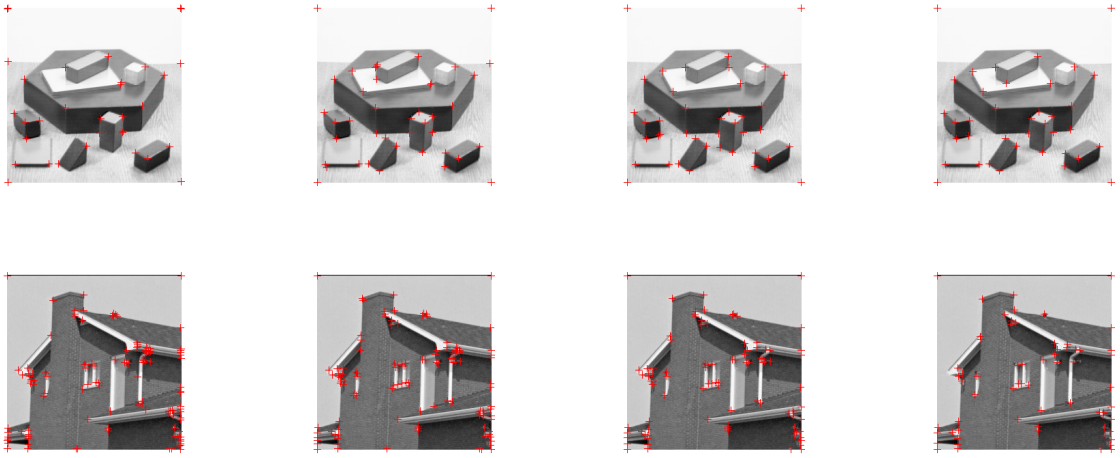

Figure 2: The variation in key points detected for values of $\sigma = [0.5, 1, 1.5, 2]$ from left to right(k=0.05, thresh=1e-5)

The standard deviation of Gaussian filter was varied in the set of values $[0.5, 1, 1.5, 2]$. The keypoints obtained for $\sigma = 0.5$ were less in number with lots of false positives, such as beginning of edges etc. For $\sigma = 1, 1.5$, the count of points detected was higher but the multiplicity of a point was higher. Given a point is detected, three or more points were detected in its neighborhood. The results were better for $\sigma = 2$, with lesser false positives and no cluttering of detected points.(Fig.2)
The empirical constant k in the Harris response function was varied for the set of values= $[0.04, 0.05, 0.06]$. On the increasing the value of k, fewer interest points were detected. Some

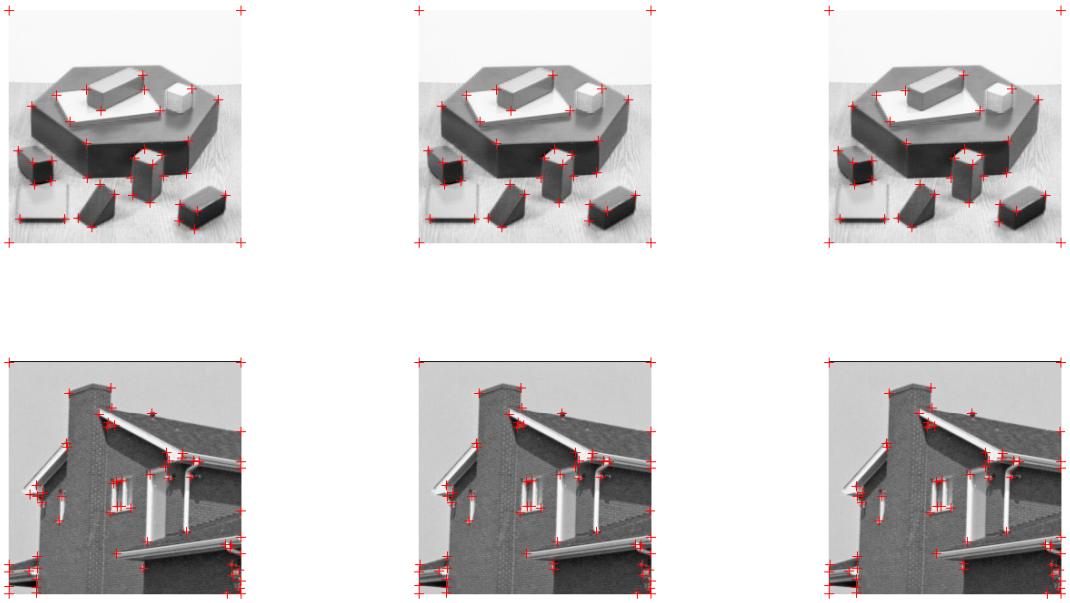

Figure 3: The variation in key points detected for values of $k = [0.04, 0.05, 0.06]$ from left to right($\sigma = 2$, thresh=1e-5)

corners which were correctly detected for $k = 0.04$ were missing in the set of points detected for higher values.(Fig.3)
The value of the threshold parameter $thresh$ was varied for the set $[1e-6, 5.5e-6, 1e-5, 5.5e-5, 1e-4]$. The points detected for $thresh = 1e-4, 5.5e-5$ miss out on a lot of interest points, so the accuracy in detection was lesser. $thresh = 1e-6$ produced a lot of false positives which were not actually interest points. Among $thresh = 5.5e-6$ and $1e-5$, the accuracy of detection was slightly higher in the former case.(Fig.4)
The final results were generated with the parameters $\sigma = 2$, $k = 0.04$, $thresh = 5.5e-6$ as shown in Fig.5.

2 Description and Matching

In order to match the interest points in two given images, the first step is to locate the interest points. This was done using Harris detection with the parameters mentioned above.

In order to avoid out-of-bound issues, the interest points which were detected close to the boundary of the image in a 10-pixel range were removed.

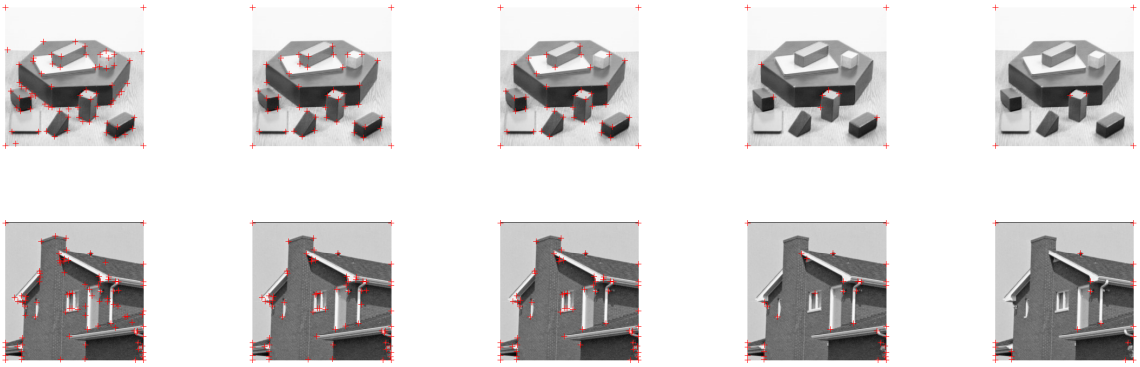

Figure 4: The variation in key points detected for values of $threshold = [1e-6, 5.5e-6, 1e-5, 5.5e-5, 1e-4]$ from left to right(k=0.04, $\sigma = 2$)

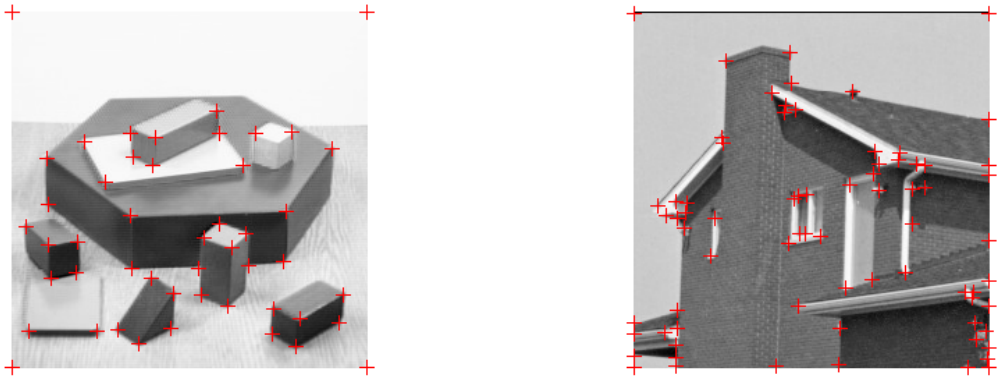

Figure 5: Interest points detected in template images for $\sigma = 2$, $k = 0.04$, $thresh = 5.5e-6$

After filtering the points near the boundary, 183 points in $image1$ and 199 points in $image2$ were detected. A $9X9$ patch around each interest point was considered as descriptor. In order to define the distance metric for matching, *squaredeuclidean* was used as the argument to the $pdist2$ function in MATLAB.(see *Appendix 3.2*)

In case of one-way matching, 183 points got matched to each other in both images. In case of mutual nearest neighbors, 128 points got matched to each other. In case of ratio test, 99 points got matched to each other.(see *Appendix 3.3*)

Figure 6: Harris detection results on the template images

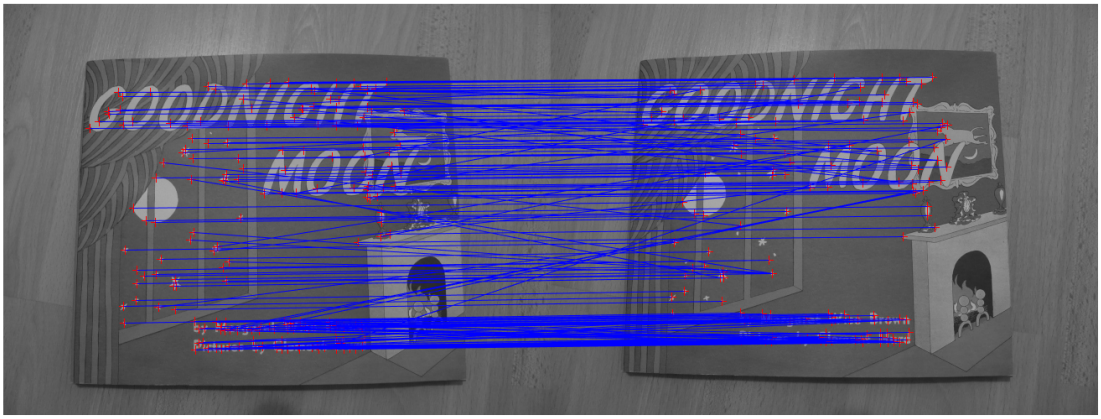

Figure 7: One-way matching from image 1 to image 2

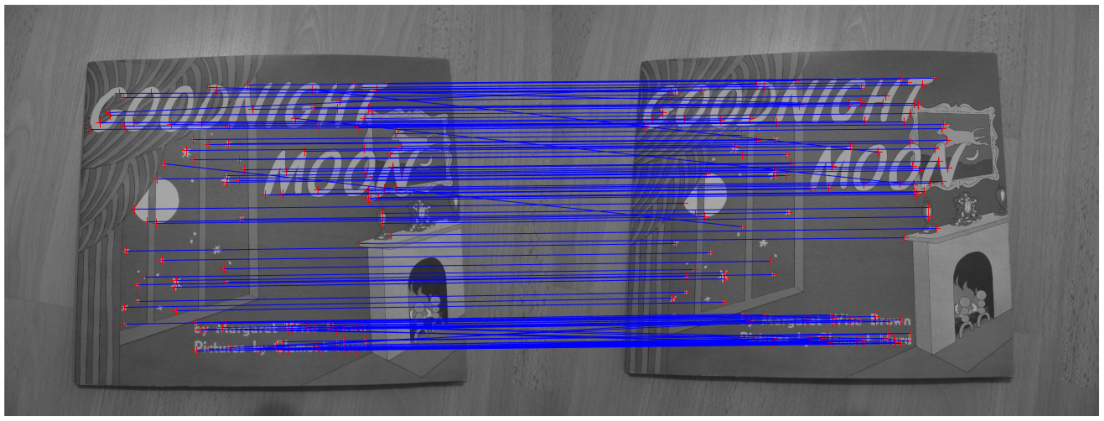

Figure 8: Mutual nearest neighbor matching between image 1 and image 2

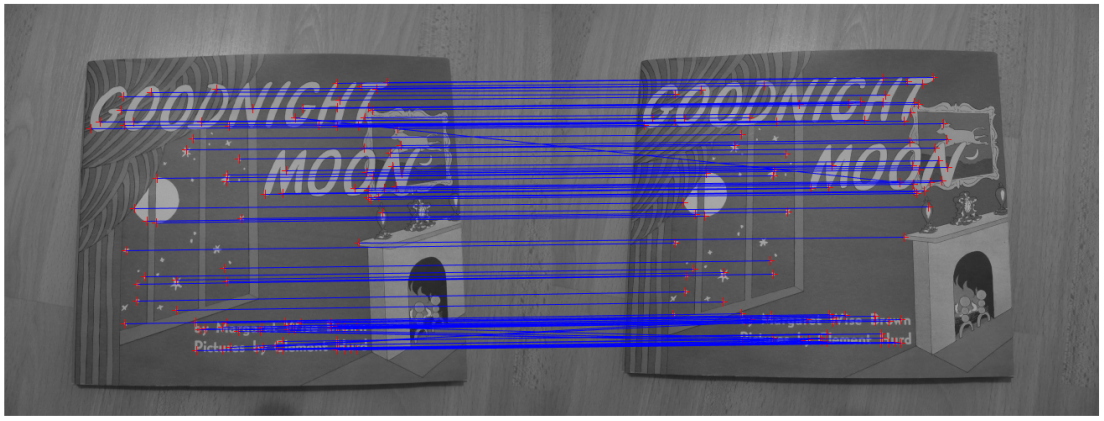

Figure 9: Ratio test matching from image 1 to image 2

3 Discussion

- As can be viewed in the results of Harris detection (Fig.5), the corners which lie at higher contrast boundaries are more probable to be detected correctly than those where change in intensity is lesser. Thus, it can be concluded from the results that Harris detection is sensitive to contrast change.

- Calculating local autocorrelation matrix requires finding the gradients of an image which is sensitive to noise and computationally expensive.

- One-way matching has some incorrect matching as can be witnessed from the matching blue lines not parallel to the image axis in Fig 7. Since it depends solely on the distance between descriptors, finding the nearest point in the feature space from a given point can result in false detection (as there can be a number of points at the same distance, say radially). A natural remedy to this problem can be a 'two-way' matching or 'mutual' matching. As seen in Fig.8, the percentage of incorrect matching has reduced, though some pairs still got wrongly matched. Ratio test matching has best results among all three methods (Fig.9). Majority of the matches were correct.

- As we moved from one-way matching to the ratio test, the number of matched points got lesser. So, the detected points were somewhat pruned by the successive tests. Ratio test pruned better matches than the mutual neighbors matching. But mutual neighbors method had more number of matches. The choice of the method depends on the application, whether we want to find more matches for a stronger global comparison or concentrate on matching of finer details.

- These matching techniques can be computationally expensive if the number of detected interest points are large in an image.

Appendix:

3.1 Harris Response Function

```
% Extract Harris corners.
%
% Input:
%   img           - n x m gray scale image
%   sigma         - smoothing Gaussian sigma
%                   suggested values: .5, 1, 2
%   k             - Harris response function constant
%                   suggested interval: [4e-2, 6e-2]
%   thresh        - scalar value to threshold corner strength
%                   suggested interval: [1e-6, 1e-4]
%
```

```
% Output:
%   corners         - 2 x q matrix storing the keypoint positions
%   C               - n x m gray scale image storing the corner
    strength
function [corners, C] = extractHarris(img, sigma, k, thresh)

% Compute gradient in x and y direction

Ix = conv2(img,0.5*[1  , 0,-1], 'same');
Iy = conv2(img,0.5*[1  ;0; -1], 'same');

% Elements of the local autocorrelation matrix
Ix2 = Ix.^2;
Iy2 = Iy.^2;
Ixy = Ix.*Iy;

% Gaussian weighting of elements of local autocorrelation matrix
Ix2g=imgaussfilt(Ix2,sigma);
Iy2g=imgaussfilt(Iy2,sigma);
Ixyg=imgaussfilt(Ixy,sigma);

% Calculation of Harris Response function C
% (A+B) is the determinant and sqrt(D) is the trace
% of the autocorrelation matrix at each pixel
A= Ix2g.*Iy2g;
B = Ixyg.^2;
D= (Ix2g+Iy2g).^2;
C= (A-B)-k*D;

% Non-maximal supression
con1= imregionalmax(C);
% Harris response function value strength
con2=C>thresh;
% Select points satisfying both condition
con=con1&con2;

[r,w]=find(con);
corners=[r,w]';

end
```

3.2 Descriptors Extraction

```
% Extract descriptors.
```

```
2 %
3 % Input:
4 %   img           - the gray scale image
5 %   keypoints     - detected keypoints in a 2 x q matrix
6 %
7 % Output:
8 %   keypoints     - 2 x q' matrix
9 %   descriptors   - w x q' matrix, stores for each keypoint a
10 %                  descriptor. w is the size of the image patch,
11 %                  represented as vector
12 function [keypoints, descriptors] = extractDescriptors(img,
       keypoints)
13
14 % filtering points close to image boundaries
15 [r,c]=size(img);
16 a=keypoints(1,:);
17
18 % heuristics: 10 pixels close to horizontal boundary
19 a1=find(10<a & a<c-10);
20
21 % same heuristics for vertical boundary
22 b=keypoints(2,:);
23 b1=find(10<b & b<r-10);
24
25 ab=intersect(a1,b1);
26
27 % keypoints satisfying both constraints
28 keypoints=[a(ab);b(ab)];
29
30 % extract discriptors
31
32 patch_size=9;
33 descriptors = extractPatches(img, keypoints, patch_size);
34
35 end
```

3.3 Feature Matching

```
1 % Match descriptors.
2 %
3 % Input:
4 %   descr1        - k x q descriptor of first image
5 %   descr2        - k x q' descriptor of second image
6 %   matching      - matching type ('one-way', 'mutual', 'ratio')
```

```
7  %
8  % Output:
9  %   matches       - 2 x m matrix storing the indices of the
       matching
10 %                   descriptors
11 function matches = matchDescriptors(descr1, descr2, matching)
12 distances = ssd(descr1, descr2);
13
14 if strcmp(matching, 'one-way')
15     [~,I]=min(distances,[],2); % minimum distance from a point in
          img1 to points in img2
16     [~,c]=size(descr1);
17     matches=[1:c;I'];
18     %error('Not implemented.');
19 elseif strcmp(matching, 'mutual')
20     [~,I]=min(distances,[],2); % minimum distance from a point in
          img1 to points in img2
21     [~,c]=size(descr1);
22     m1=[1:c;I'];
23     [~,I]=min(distances,[],1); % minimum distance from a point in
          img2 to points in img1
24     [~,c]=size(descr2);
25     m2=[1:c;I];
26
27     t1=table(m1(1,:)',m1(2,:)');
28     t2=table(m2(2,:)',m2(1,:)');
29     t3=ismember(t1,t2);          % common matches in both cases
30     t4=t1(t3,:);
31     f=table2array(t4);
32     matches=f';
33
34     %error('Not implemented.');
35 elseif strcmp(matching, 'ratio')
36     [~,I]=min(distances,[],2); % minimum distance from a point in
          img1 to points in img2
37     [~,c]=size(descr1);
38     m1=[1:c;I'];
39     a=mink(distances,2,2);       % second nearest neighbour
40     a(:,3)=a(:,1)./a(:,2);       % Ratio test
41     idx=a(:,3)<0.5;              % Threshold= 0.5
42     matches=m1(:,idx');
43     %error('Not implemented.');
44 else
45     error('Unknown matching type.');
```

```
46 end
47 end
48
49 function distances = ssd(descr1, descr2)
50
51 distances=pdist2(descr1',descr2','squaredeuclidean');
52 %error('Not implemented.');
53 end
```