Problem Definition

In this assignment, we are asked to implement "Harris Corner Detection" algorithm to detect the corners of the objects exist in the images. It's also asked to convert thick corners into thin corners by "Nonmax Suppression" method.

Problem Solution

To tackle corner detection problem, as asked in the assignment, I implement a detection program based on “Harris Corner Detection” algorithm. The steps of the algorithm are as follows:

1) Compute x and y derivatives of the image:
   For both of finding the gradients on each pixel (Ix, Iy) and applying the smoothing to reduce the noise in the image, I use “Gaussian Smoothing Filter”. Because 2D Gaussian equation is separable, I take the first derivatives of that equation w.r.t. x and y and convolve them with the image to get the gradient values. Sigma value is 1 by default and I utilize from meshgrid function of Matlab to be able to generate a filter as Gaussian distributed.
2) Compute the products of derivatives at every pixel:
   This is an intermediate step to construct the matrix H that will be then used to determine the response of the pixels.
3) Compute the sums of the products of derivatives at each pixel:
   This is another intermediate step for matrix H and yields indices of that matrix. Here, 2D Gaussian filter is applied to the results got in step 2 (Ix2, Iy2, Ixy) by convolution.
4) Define at each pixel (x, y) the matrix:
   Here, the matrix is constructed by the results of step 3 (Sx2, Sy2, Sxy). This matrix shows the orientation of eigenvalues $\lambda 1$ and $\lambda 2$ and thus what the type of pixel is (corner, edge or flat region).
5) Compute the response of the detector at each pixel:
   By differing square of trace of the matrix H times the coefficient k from the determinant of the matrix H, I get the response of the pixel. To make calculation efficient, I use function "trace" of Matlab to determine the sum of the diagonals of the matrix (Calculating trace manually gave bad result in terms of "computation in time").
6) Check whether or not the response of a pixel exceeds the pre-defined threshold value.
   Response value determined in step 5 is compared with a predefined threshold value, which is 50000 by default, to classify the pixel as corner (if it's greater than threshold) or others we aren't interested in such as edge or flat region (if it's less than or equal to threshold). The value of the output image in given pixel is set as response value where the threshold is exceeded and is set as 0 (black) where the response fails exceeding the threshold.
7) Compute nonmax suppression
   To thin the corners as one pixel per corner, I apply the "Dilation Filter". This filter finds the local maximum of each pixel group marked as corner.

Experiments

To test the implementation, a grayscale image whose size 300x300 is used (Figure 1). The name of the image file is "corner_gray.png" and locates in the doc directory. This image file contains four geometric objects, three of them are the shapes that have corners (rectangle, square and triangle). Forth shape is a circle that doesn't have any corner at all.

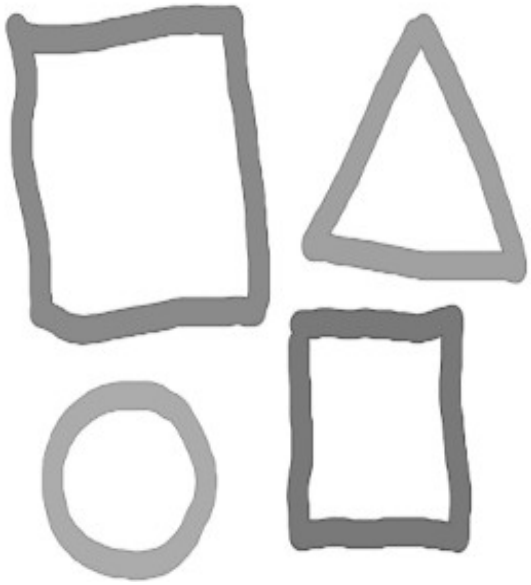

Figure 1 - The input image

The implementation is executed with different values of parameters "sigma" and "threshold". The values used are as follows:

Sigma	Threshold
1	10000
1	20000
2	10000
2	20000

Results

First of all, I executed the code with values of sigma = 1 and threshold = 10000. In this case, the output image was generated with some noise (4 mis-classified pixel) (Figure 2). When the threshold value was increased to 50000 with same sigma value, the number of mis-classified pixel decreased from 4 to 3 (Figure 3). In both case, the corners of the objects were recognized correctly though.

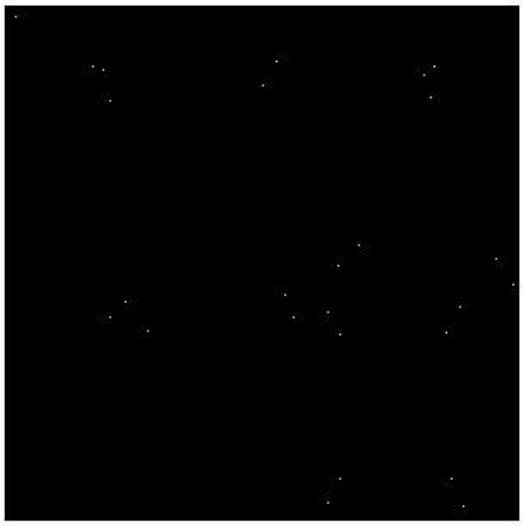

Figure 2 - Sigma = 1, Threshold = 10000

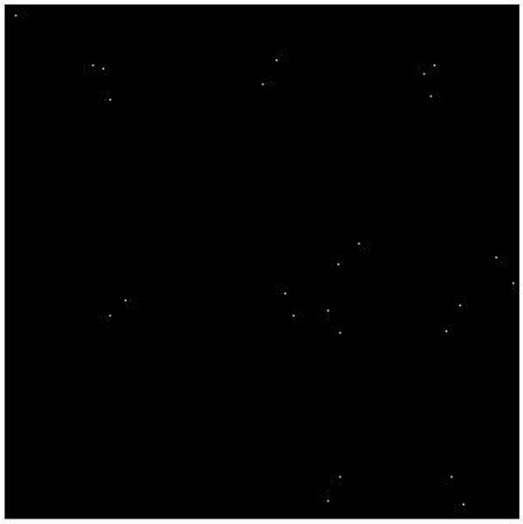

Figure 3 - Sigma = 1, Threshold = 50000

As second phase, I executed the code with values of sigma = 2. Firstly, I run the code with threshold = 10000 and the number of mis-classified pixel increased to 7 (Figure 4). And finally, I run the code with

threshold = 50000 and the noise was the same as in previous case (Figure 5). Again, the corners of the objects were recognized correctly as well.

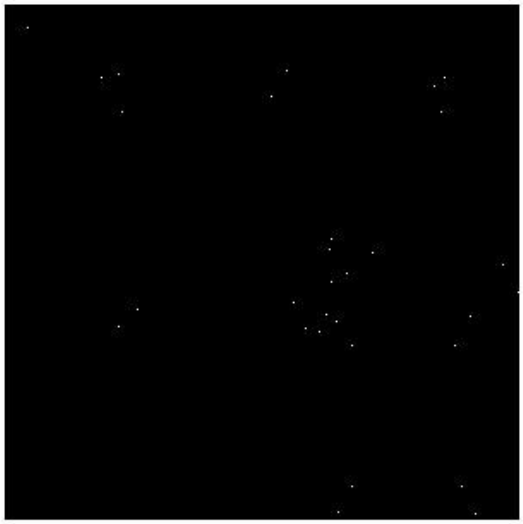

Figure 4 - Sigma = 2, Threshold = 10000

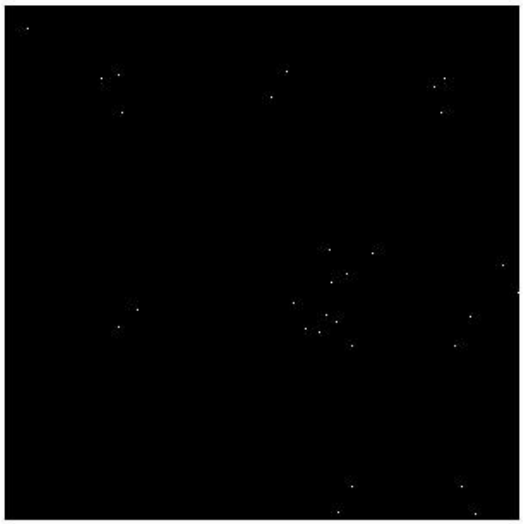

Figure 5 - Sigma = 2, Threshold = 50000

Based on the experiments done above, the optimum value for sigma and threshold were selected as 1 and 50000 respectively.

Conclusion

In this assignment, I implemented the "Harris Corner Detection" algorithm from scratch and also modified algorithm by adding thinness property of corners detected by "Dilation Filter". I used different sigma and threshold values to obtain the optimum results from the implementation. As a final statement, the implementation runs a little bit slower than expected and there may need to be optimized to run more efficiently.