Visual Perception

Lab 1 Report: Corner Detection

Mohammad Rami Koujan

M.Sc. VIBOT

University of Girona

1 Introduction:

During this lab sessions, an implementation of image corners detection method based on Harris algorithm was performed using Matlab program. The way of implementation was on stages, step by step, until getting the final results with some optimizations.

2 First part

In this section the main idea is to calculate the matrix E which contains for every point the value of the smaller eigenvalue of M. Starting from the available code, which calculates the first and second order derivatives of the read image, the only required thing is to compute the eigenvalues of the matrix M for each pixel and then taking the minimum one. In order to compute the matrix M in an effective way, a kernel with size 3*3 and value of 1 can be convolved with the images of first and second order derivative so that the individual elements of M for every pixel are ready to be stored in separate variables. Then, a loop should iterate on every pixel to calculate the eigenvalues of the matrix M and then extract the minimum value for each pixel and store it in the E matrix. Figure 1 shows matrix E image for different chessboard input images.

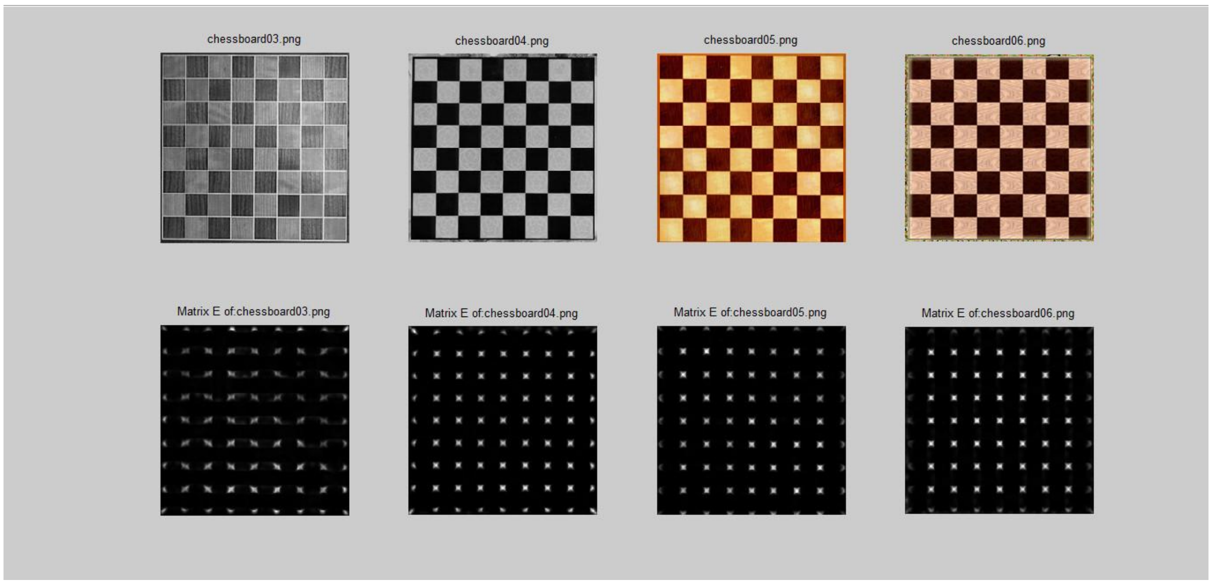

chessboard03.png
chessboard04.png
chessboard05.png
chessboard06.png
Matrix E of:chessboard03.png
Matrix E of:chessboard04.png
Matrix E of:chessboard05.png
Matrix E of:chessboard06.png

Figure 1: E matrices for different input images

The previous images demonstrate that the pixels at corners in the original images are highlighted, given higher intensity value, in the E images while homogenous regions have dark colors, low intensity values.

3 Second part

The next step is to calculate the matrix R from the following equation:

$$R = \det(M) - k \cdot tr(M)^2$$

This can be done by simply starting from the computed matrix M in the previous step and then calculating the determinant and the trace of this matrix. Basically, there are two main differences between R and E matrices. The first one is related to the computation time. If matrix R is calculated based on the built-in det() and trace() matlab fucntions, calculating matrix E will be faster than R by about 35 % of the time required for R. However, if R is computed manually, based on multiplication and addition operations, the time response of R will be better by 40 % of the time used for E. Secondly, the results of both matrices are somewhat the same, even though the intensity values of corners in matrix E are higher than that in R, which means more accurate detector. Figure 2 shows R matrix images for the corresponding input images.

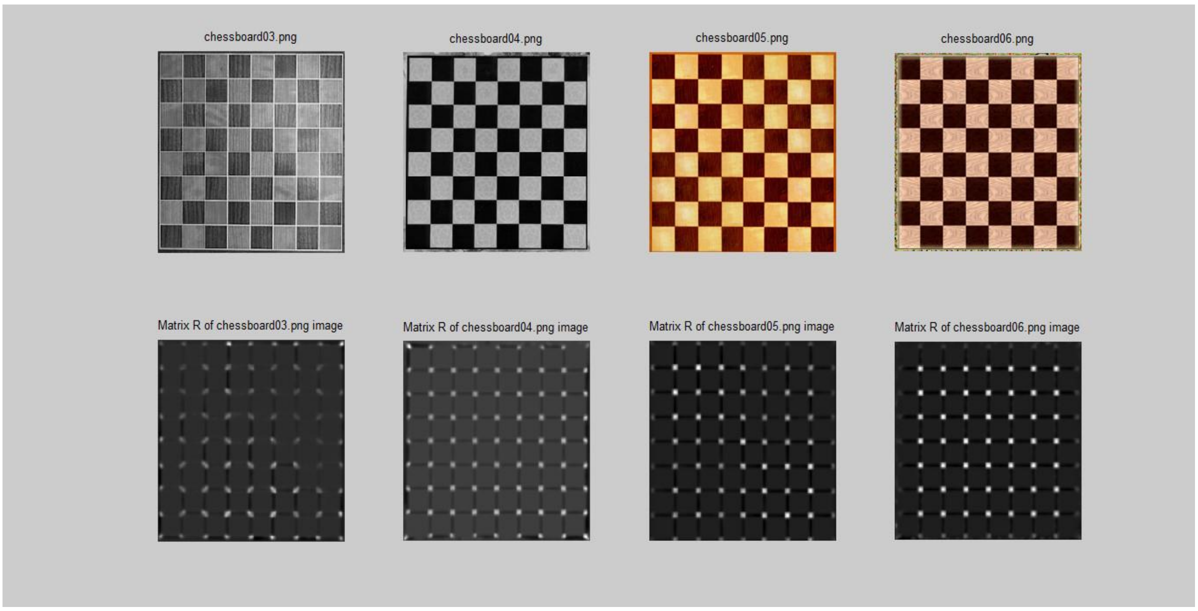

chessboard03.png
chessboard04.png
chessboard05.png
chessboard06.png
Matrix R of chessboard03.png image
Matrix R of chessboard04.png image
Matrix R of chessboard05.png image
Matrix R of chessboard06.png image

Figure 2: R matrices for different input images

4 Third Part

Selecting the 81 most salient points of matrix E and R, as demanded in this part, will give the images in figure 3 and 4 respectively.

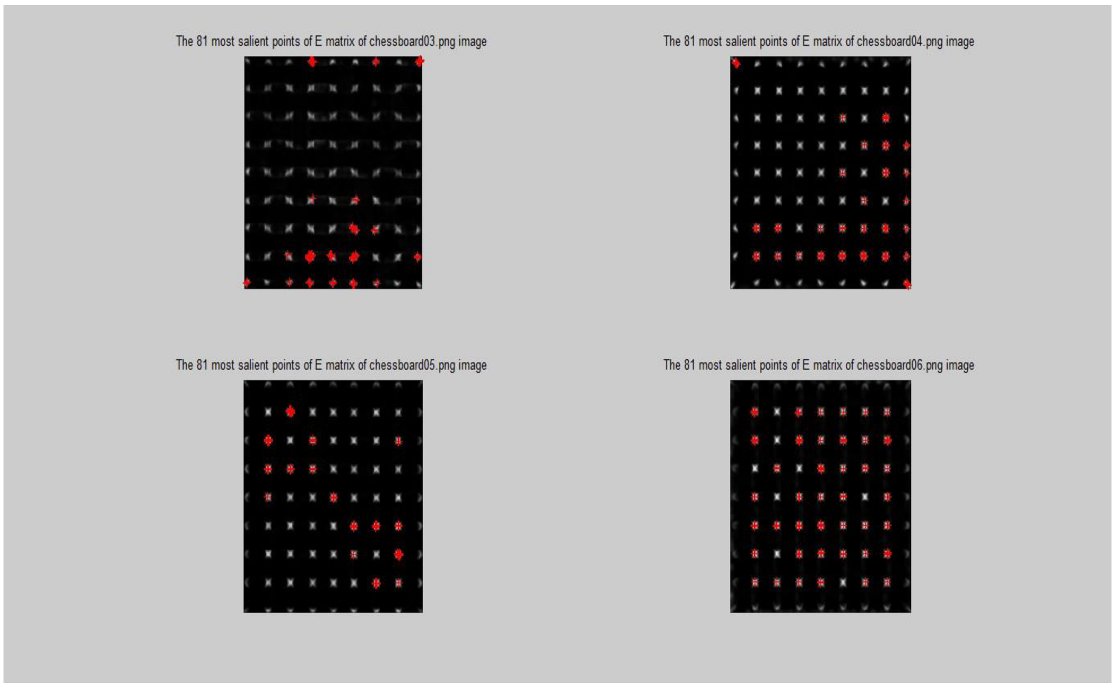


Figure 3: The 81 most salient points in matrix E for different input images

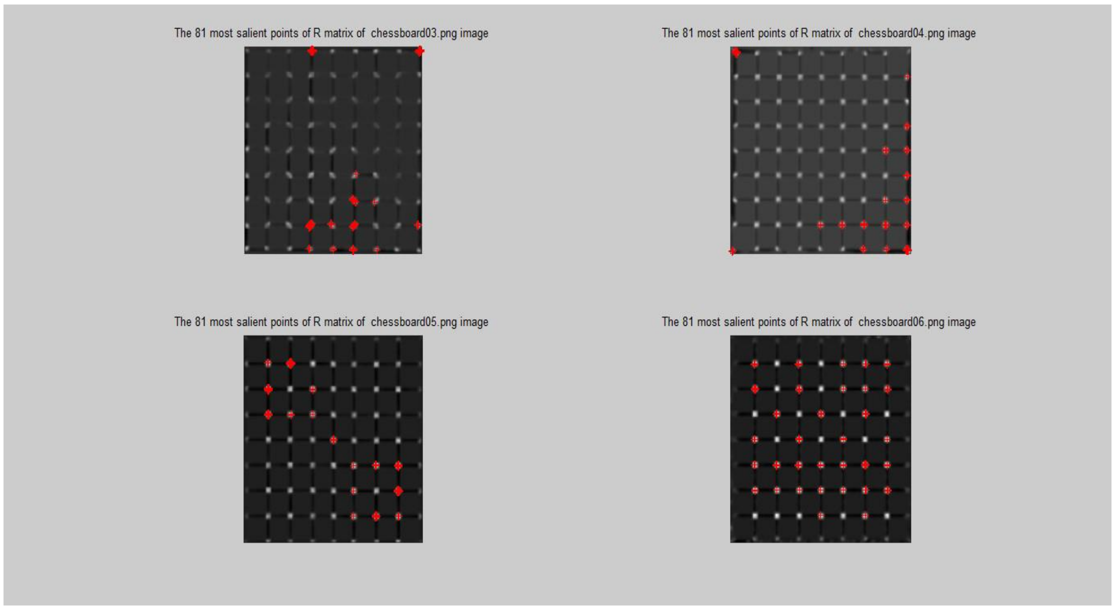
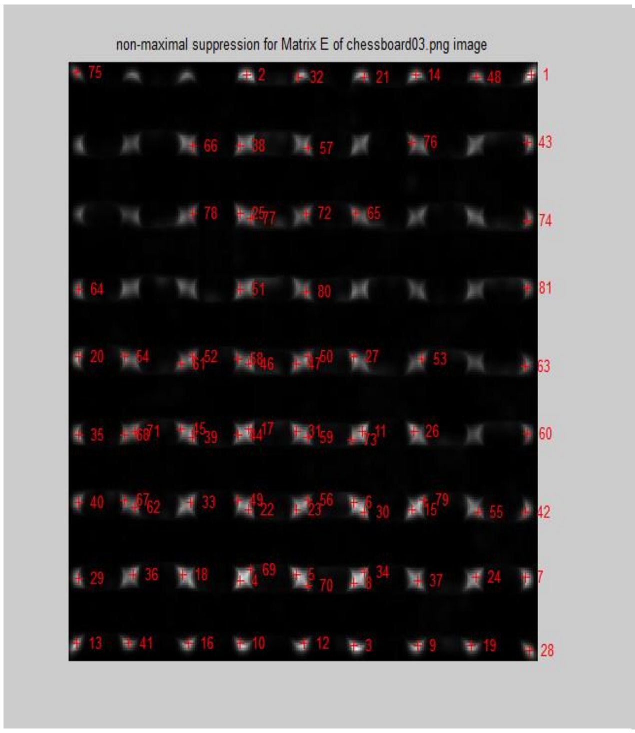


Figure 4: The 81 most salient points in matrix R for different input images

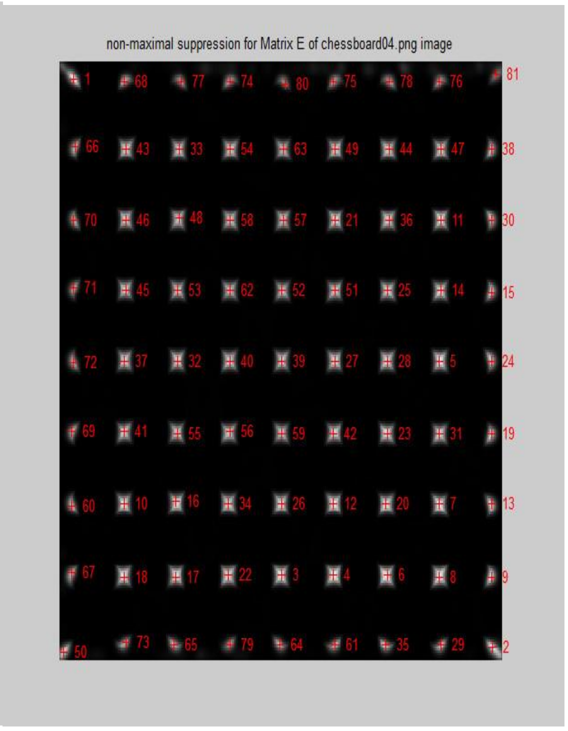
The images clearly show that the 81 selected points are not on the top of the corresponding corners in the original images. This is because some small areas around some corners have a higher intensity value than other corners, which explains why some corners points in the shown results are gathered in small clouds. It is noteworthy that this effect is higher in R images than in E images.

5 Fourth part

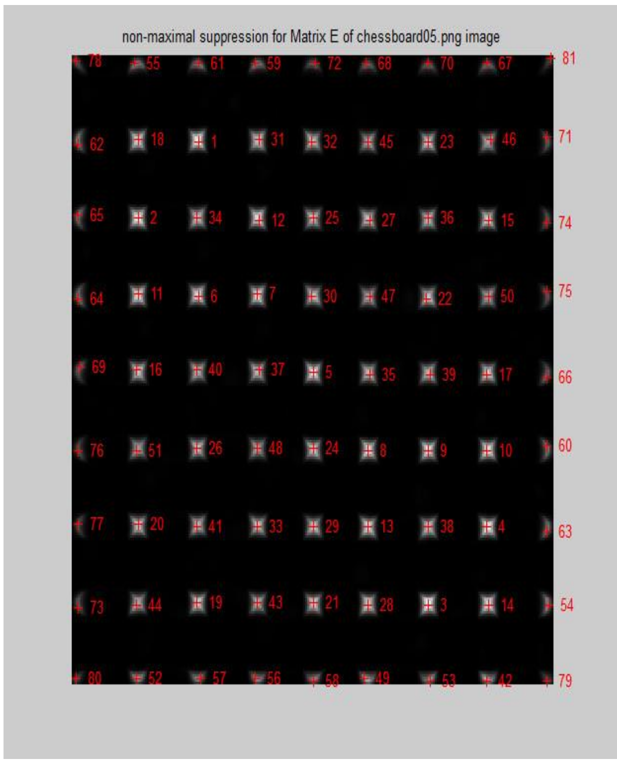
In this step, an attempt to compensate for the undesired results obtained in the previous step was made. The main idea is to perform a non-maximal suppression for E and R. This was done by opening a window of a size 11*11 around the maximum point and setting all the neighbors to the minimum value in the processed matrix. Thereafter, this procedure should be repeated for the next maximum point and so on until scanning the 81 most salient point. The results for both R and E are given below.


non-maximal suppression for Matrix E of chessboard03.png image

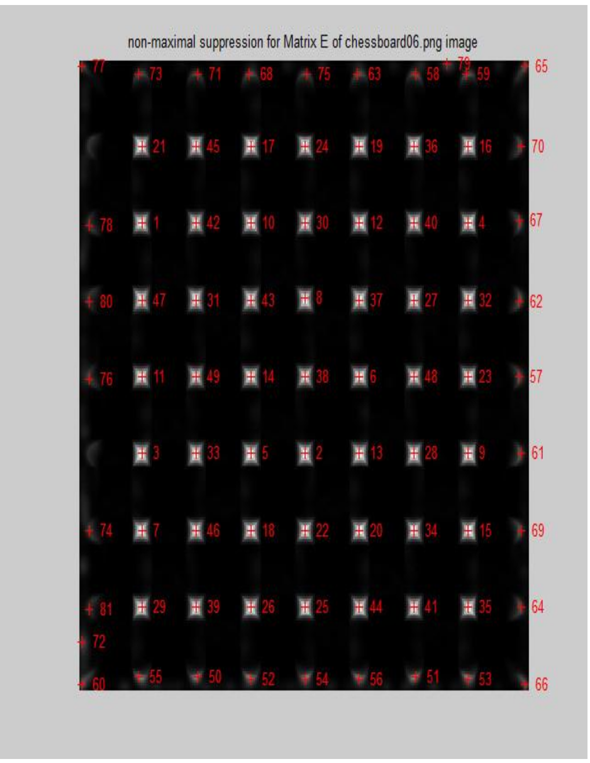
(5.1)


non-maximal suppression for Matrix E of chessboard04.png image

(5.2)

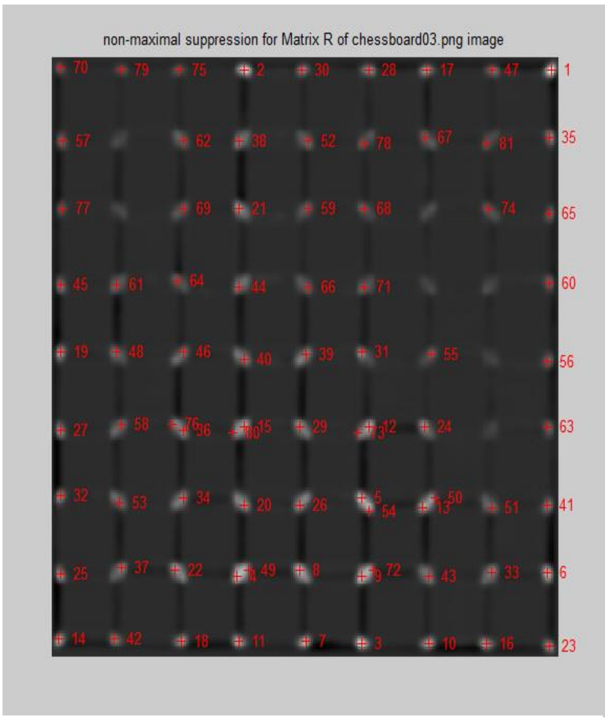

non-maximal suppression for Matrix E of chessboard05.png image

(5.3)

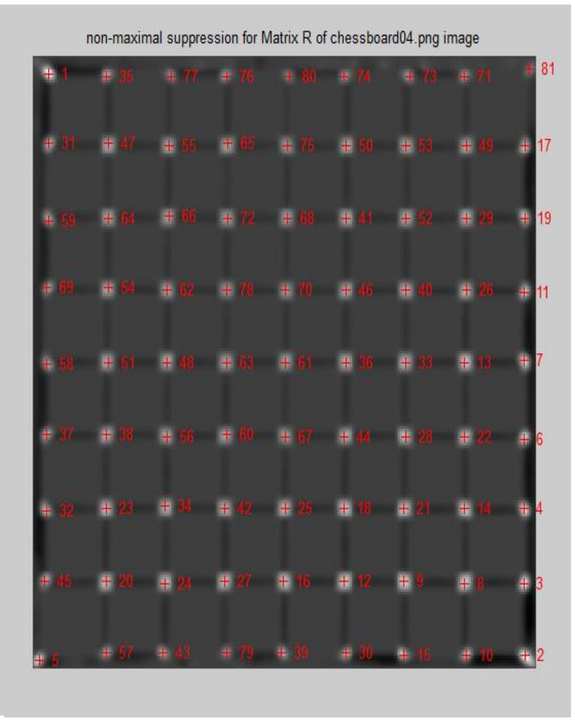

non-maximal suppression for Matrix E of chessboard06.png image

(5.4)

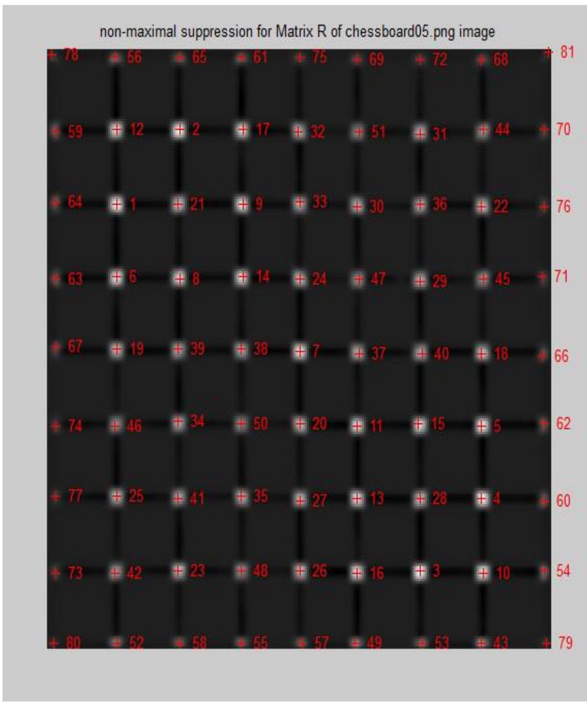
Figure 5: Results of applying non-maximal suppression on E images



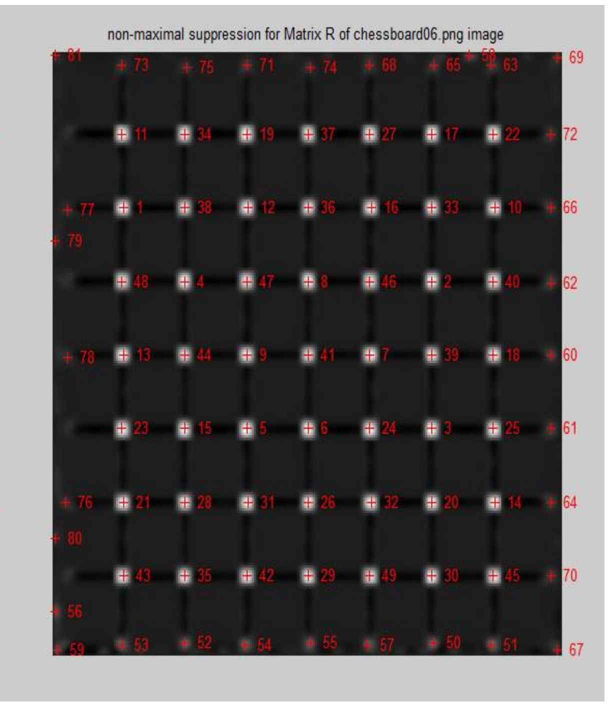
(6.1)



(6.2)



(6.3)



(6.4)

Figure 6: Results of applying non-maximal suppression on R images

As it is shown in figure 5 and 6, the selected points are better distributed on the corresponding corners. Nevertheless, there are still some corners that are not associated with any point. This problem can be solved by increasing the size of the window and it also depends on the image itself. That is to say, since image "chessboard03" has wider edges and consequently more spread corners on both sides of the connection of these edges, it is better to choose much higher window size for this image. The following figure illustrate this by choosing the size to be 25*25

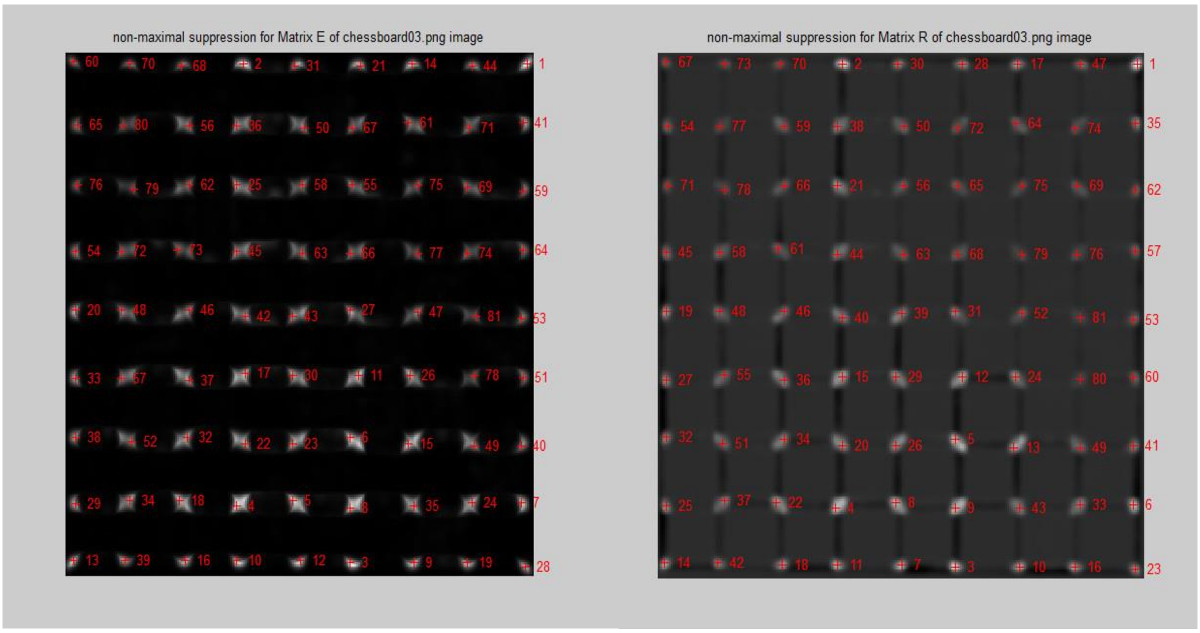


Figure 7: non-maximal suppression for E and R with window size=25*25

6 Fifth Part

The last part is even one step forward to optimize the previous results and make them more precise. This could be done by doing what is known as subpixel accuracy. The concept is to fit a parabola on the top of the previously chosen maximum point. After that, the location of the local maximum of this parabola will be recognized as a corner. In order to do so, a window of size 3*3 was opened around each maximum point in E or R matrix, obtained from the fourth part, and then the fitting procedure was based on the following equation:

I(x,y)=a*x.^2+b*y.^2+c*x+d*y+e*x*y+f

Where I(x,y) is the intensity of the pixel found in row y and column x of the matrix E or R. After calculating the corresponding equations for every point in the window, we get an over-determined system of 9 equations and sixth unknowns (a,b,c,d,e,f). This system can be solved by using the pseudo inverse function in Matlab. The result was an equation of a parabola in 2D. This parabola may have its global maximum, which can be calculated by setting the first order partial derivatives to zero and then solving for the corresponding point, outside our window of interest. Hence, the maximum should be localized within the searching window, not by using the partial derivatives which may give global extremum point depending on the fitted parabola.

One possible approach to obtain this local point is by using a brute force method which tends to do high density sampling of the interval of interest and then finding the maximum point. The results for subpixel accuracy method are shown below.

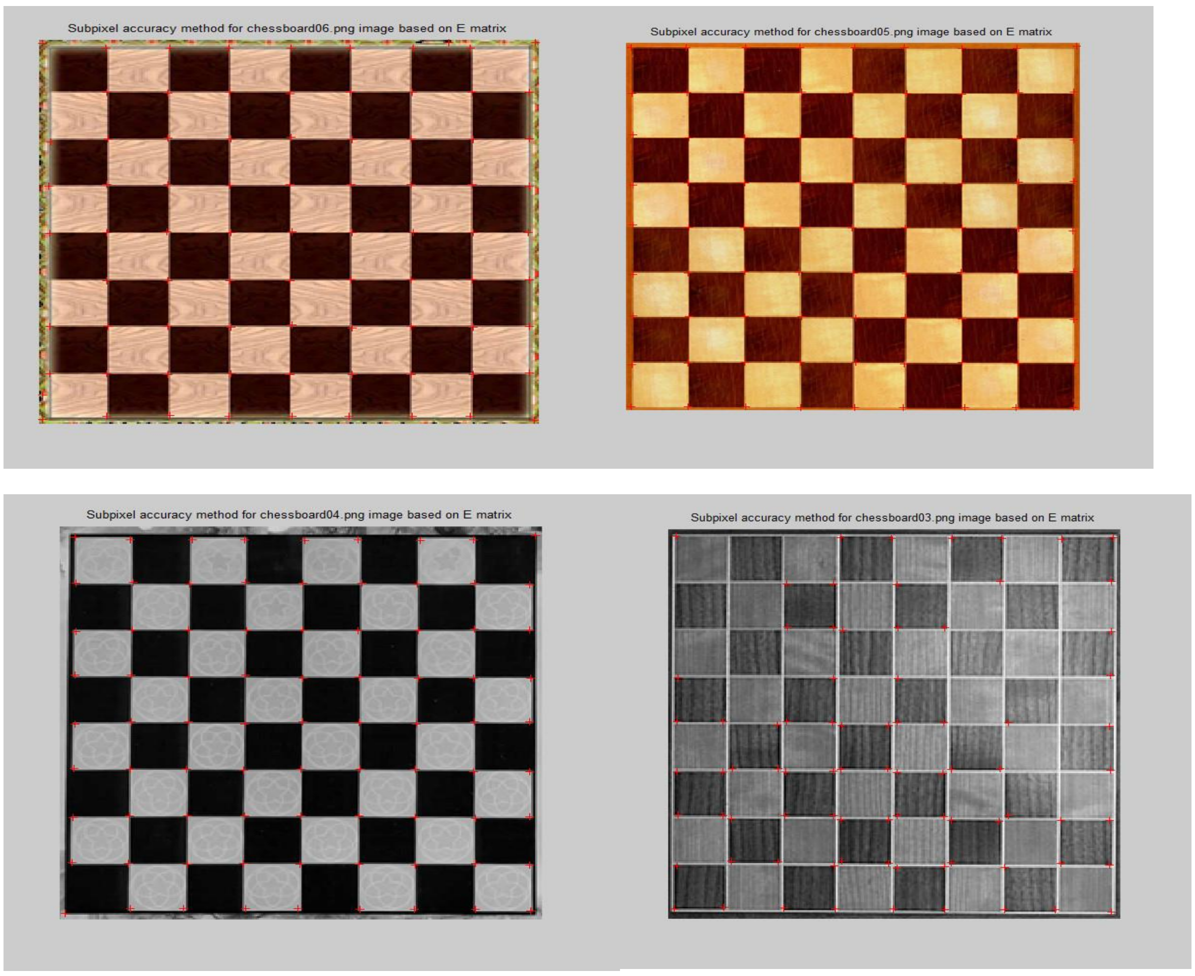
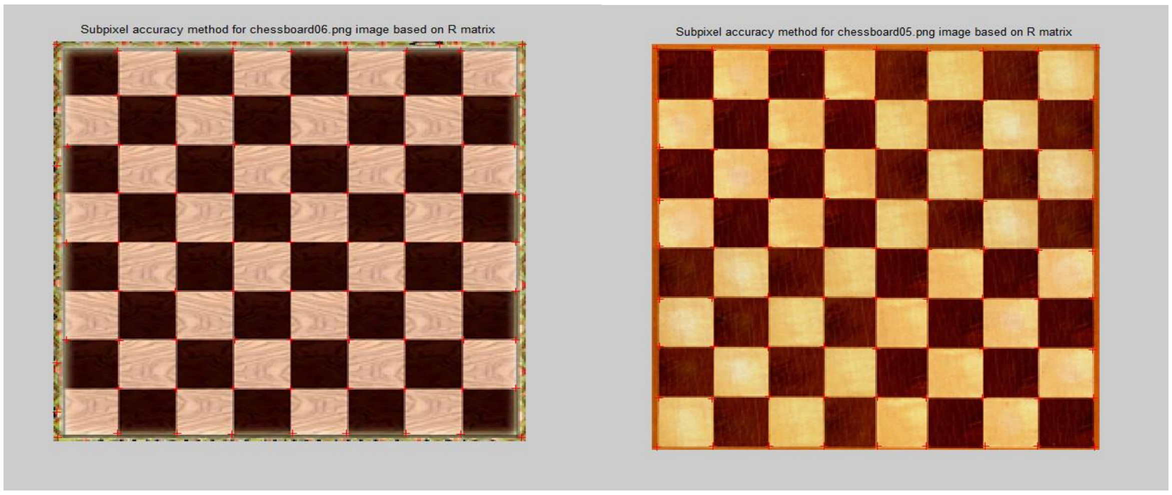
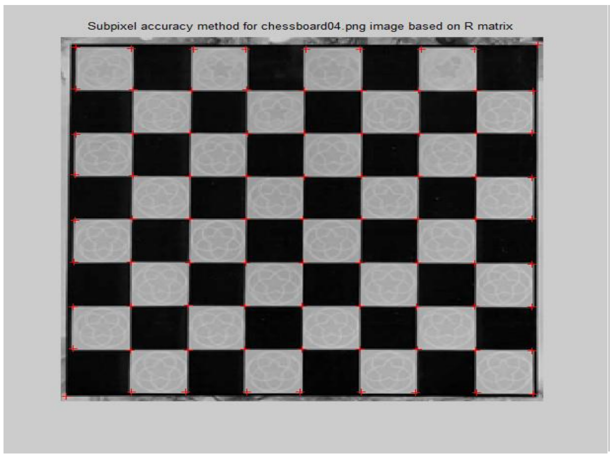

Subpixel accuracy method for chessboard06.png image based on E matrix
Subpixel accuracy method for chessboard05.png image based on E matrix
Subpixel accuracy method for chessboard04.png image based on E matrix
Subpixel accuracy method for chessboard03.png image based on E matrix

Figure 8: detected corners from E matrices by subpixel accuracy method


Subpixel accuracy method for chessboard06.png image based on R matrix
Subpixel accuracy method for chessboard05.png image based on R matrix


Subpixel accuracy method for chessboard04.png image based on R matrix

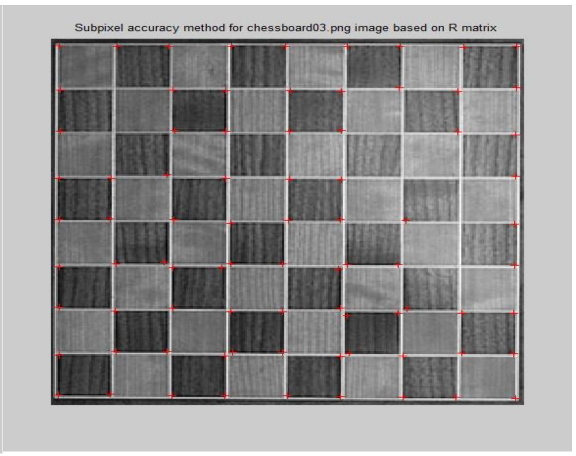
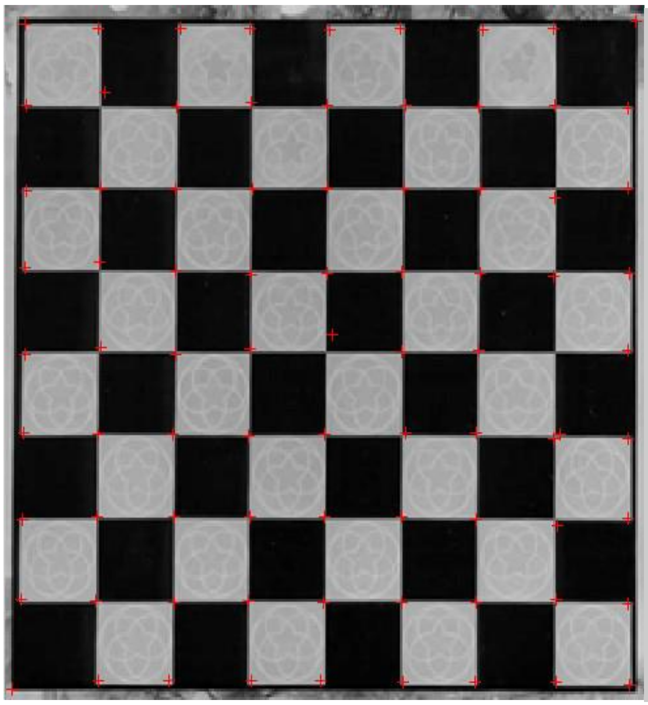

Subpixel accuracy method for chessboard03.png image based on R matrix

Figure 9: detected corners from R matrices by subpixel accuracy method

For the sake of comparison, figure 10 presents two images. In the first one, the maximum points, for chessboard04 image, were calculated again but this time based on the derivative approach mentioned previously. The second one shows one of the fitted parabolas that has a global maximum outside the 3*3 window, specifically the maximum is at point (1.6, 11.72).



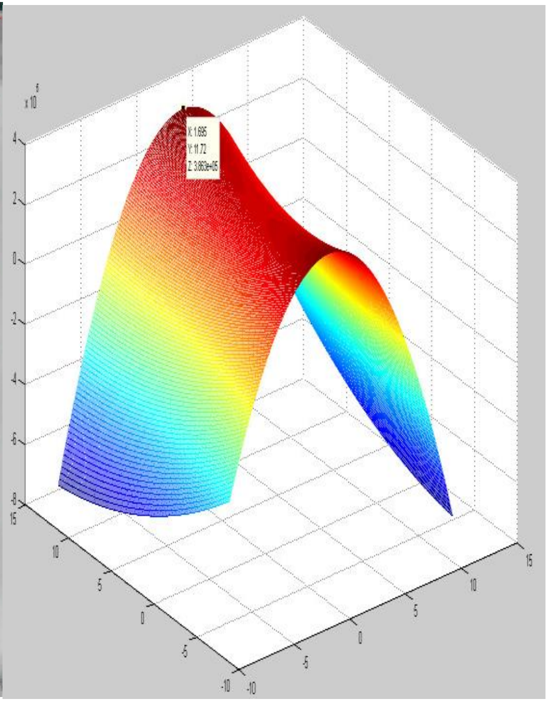


Figure 11: The result of applying subpixel accuracy with global maximum points

Figure 11 clearly shows how some points are outside the window of interest.

7 Conclusion

In this lab, Harris algorithm for image corners detection was successfully implemented in Matlab program. This method could be built based on R or E matrix. These two approaches, in fact, are a trade-off between accuracy and speed. Detecting corners based on only finding the first N maximum points in R or M matrix is not enough. Therefore, another method should be used like the non-maximal suppression which gives better results. However, it is highly dependent on the window size to which some attention should be paid. An even more accurate means of solving the previous problem is to choose the subpixel accuracy approach which does some sort of fitting by a suitable 2d function, e.g. parabola.