## 1.1 Harris Corner Detector

## Aim

To understand the Harris corner detector through the implementation of the cornermetric Matlab command. Corners show a significant gradient change in all directions and this makes them easy to identify. The R value is an indicator of the strength of the gradient change in particular direction which is due to a corner.

## Method

maxR = max(max(R(:,:)))

minR = min(min(R(:,:)))

threshold = 0.005

idx = find(R1 < threshold);

R1(idx) = 0;

The code above finds the maximum and minimum R values received from the “cornermetric” function. It then applies a user set threshold to threshold the image.

## Results



Figure Image A



Figure Image B

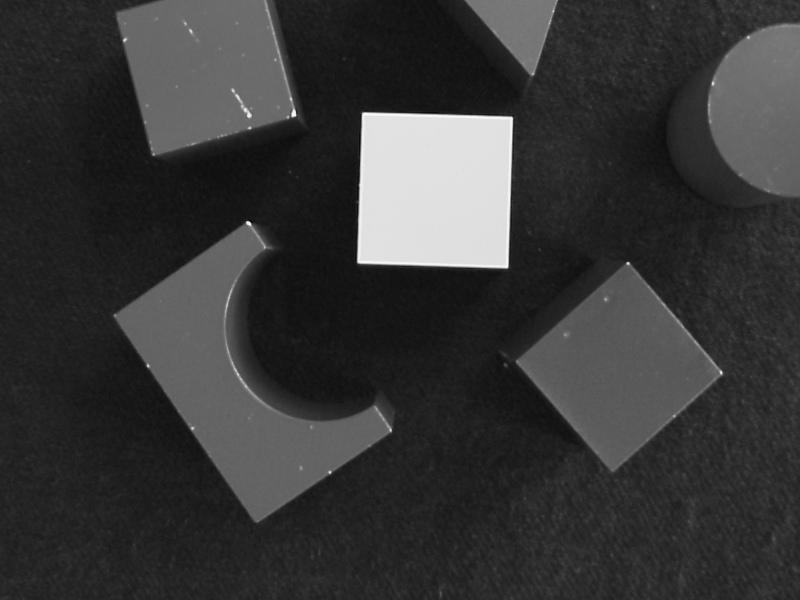
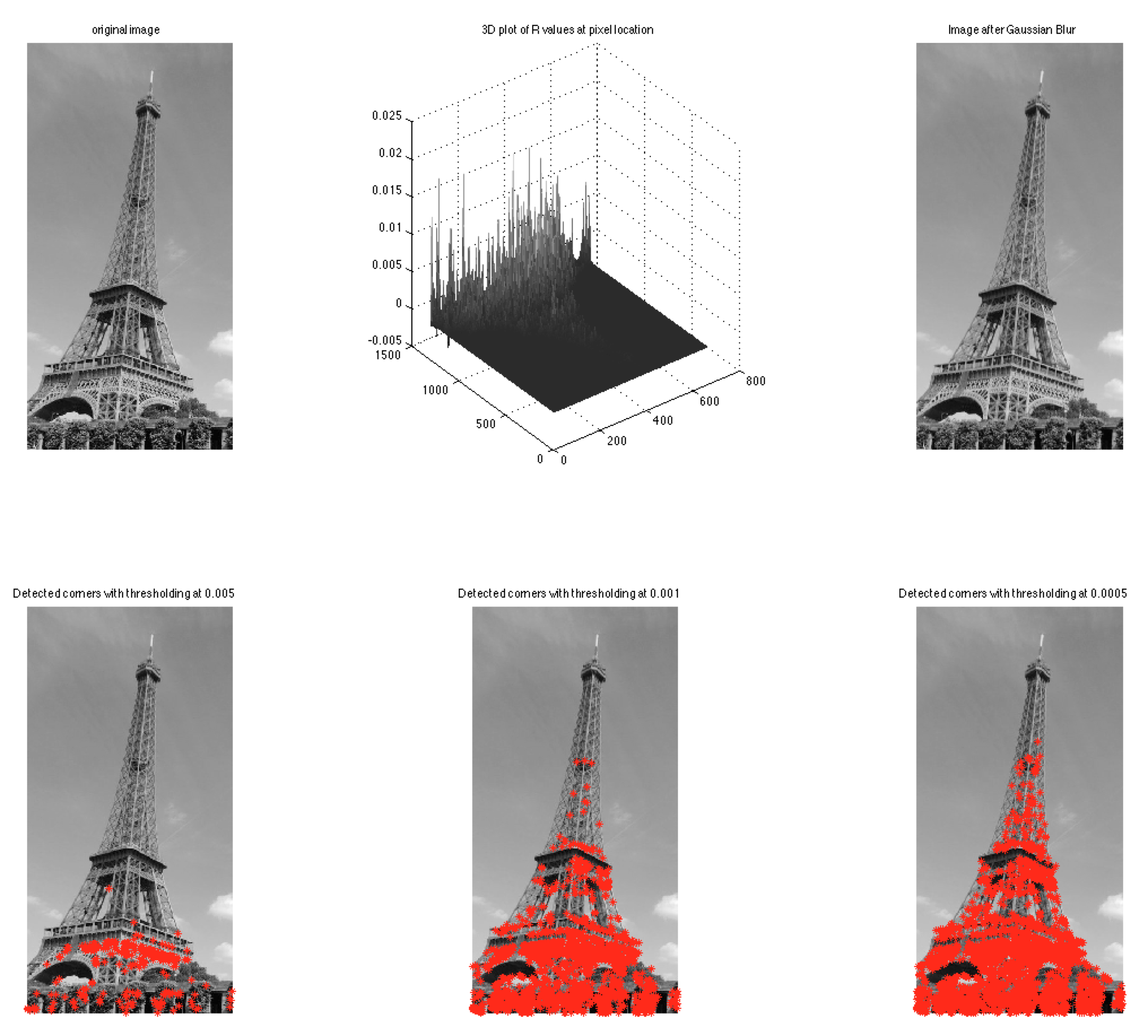


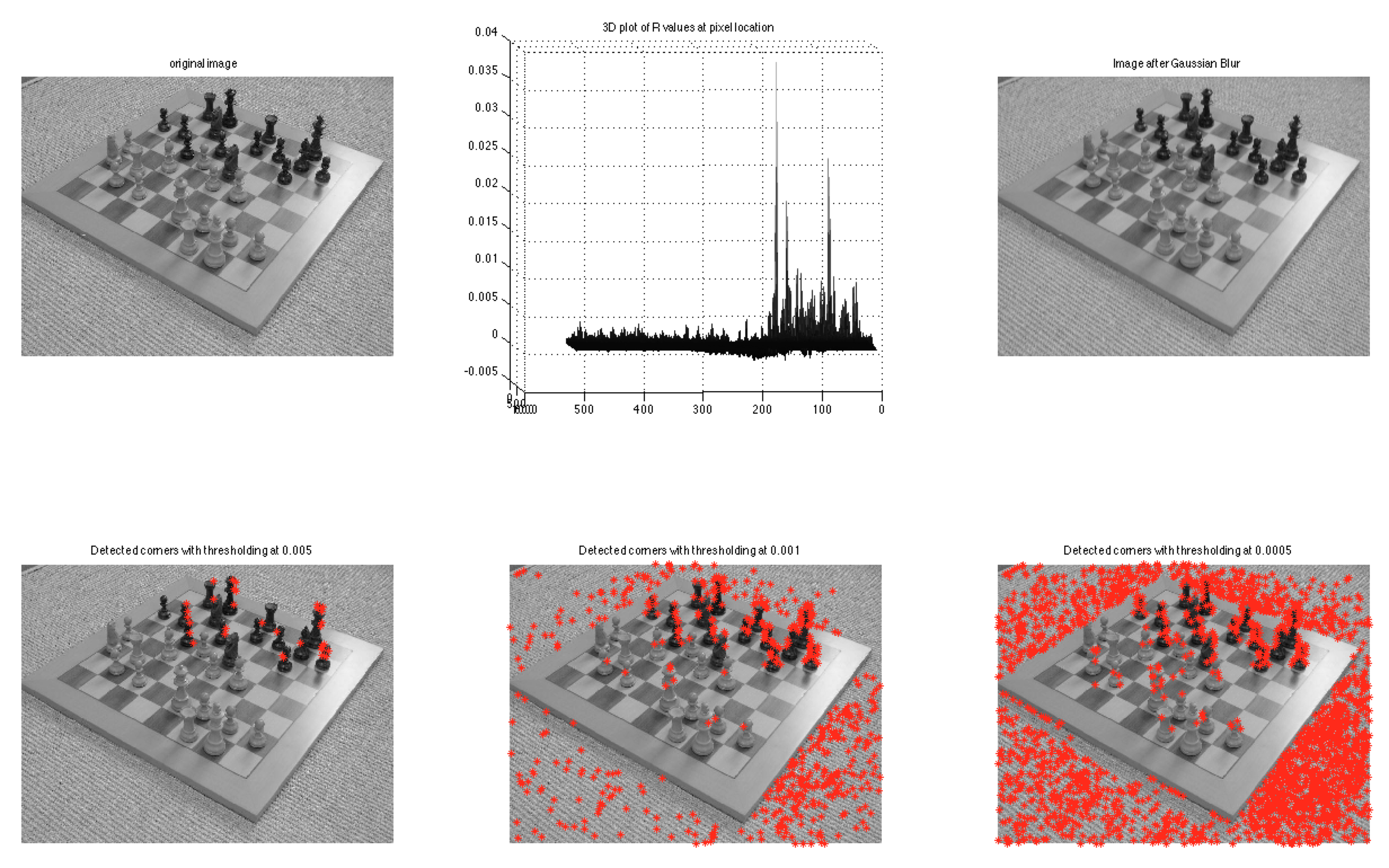
Figure Image C

The following images were used to investigate the effectiveness the Harris corner detector. Image A was selected to study the effect of how noisy backgrounds can be picked up as corners. Image B was selected to determine the how a similar intensity background affects the detection of corners around objects.



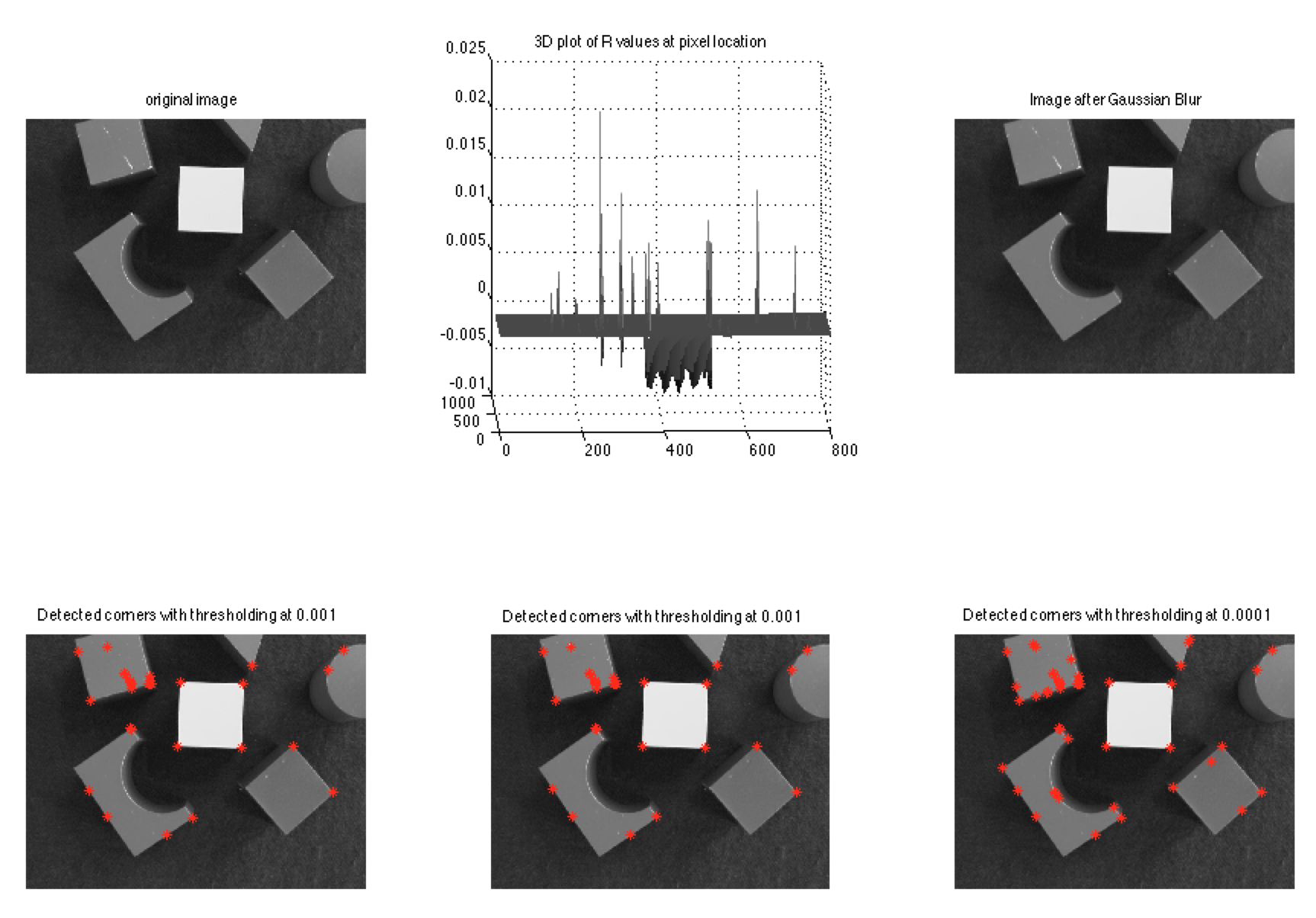
Figure

Figure 4 shows the results of using the Harris detector with threshold R values set at 0.005, o.001 and 0.0005. The top left image shows the converted gray scale image and the top right image shows the image after the Gaussian filter is applied. The top middle image is of a 3D plot of R value at pixel location. It shows a much high concentration of corners toward the bottom of the image. The bottom images show the detected corners with decreasing threshold values of 0.005, 0.001, and 0.0005 from left to right.



Figure

Figure 5 uses the same layout of images as figure 4. The bottom left image with a threshold value of 0.005 shows corners detected only on the black chess pieces. The bottom right image with a threshold value of 0.0005 shows many detected corners on the carpeted background around the chessboard.



Figure

Figure 6 shows the same layout of images as figures 4 and 5 but the bottom right image has a lower threshold value of 0.0001 to attempt to detect more corners. The corners on this image are far more pronounced as shown as lone peaks on the 3d plot.

## Discussion

## 2.1 Canny Edge Detector

## Aim

To use different edge detecting techniques and understand their differences. Edges can be formed due to depth discontinuity, material change, colour change and change in texture. A canny edge detector, a sobel edge detector and laplacian of Gaussian detectors can be very helpful in detecting edges.

## Method

The following code was used to apply the sobel and laplacian of Gaussian edge detectors. To replicate the smoothing effect of the canny edge detector, an averaging filter was applied to the images. The values for each of the images used will be show in the results section. The thresholding for the sobel and LOG function was done automatically to increase the useability of the code.

## Results

## Discussion

The canny edge detector was the easiest edge detector to use as it didn’t require a smoothing filter to be used as instead the sigma value could be changed. It finds the local maximum for the gradient where the gradient is calculated using the derivative of a Gaussian filter.

The Canny detector takes two inputs: one for the threshold value and the other to threshold the noise/weak edge. In the output only the strong edge are displaced unless the weak edges are connected to them. Therefore it is advisable to first use the canny edge detector.

The LOG edge detector is the hardest one to edge detectors to use as it is very susceptible to noise. These had to be larger averaging filters needed to be used on the LOG image. This could be due to the use of the laplacian technique which highlights noise as well as the edge. This has already been explored in last week tutorial.

If the other filters are to be used a good starting value for an averaging filter size is 5 by 5. Letting the filters automatically choose their own threshold worked well and was far more efficient than changing values for each case.

## 2.1 Hough Transform

## Aim

To use the Hough transform and associated function to detect lines in images.

## Method