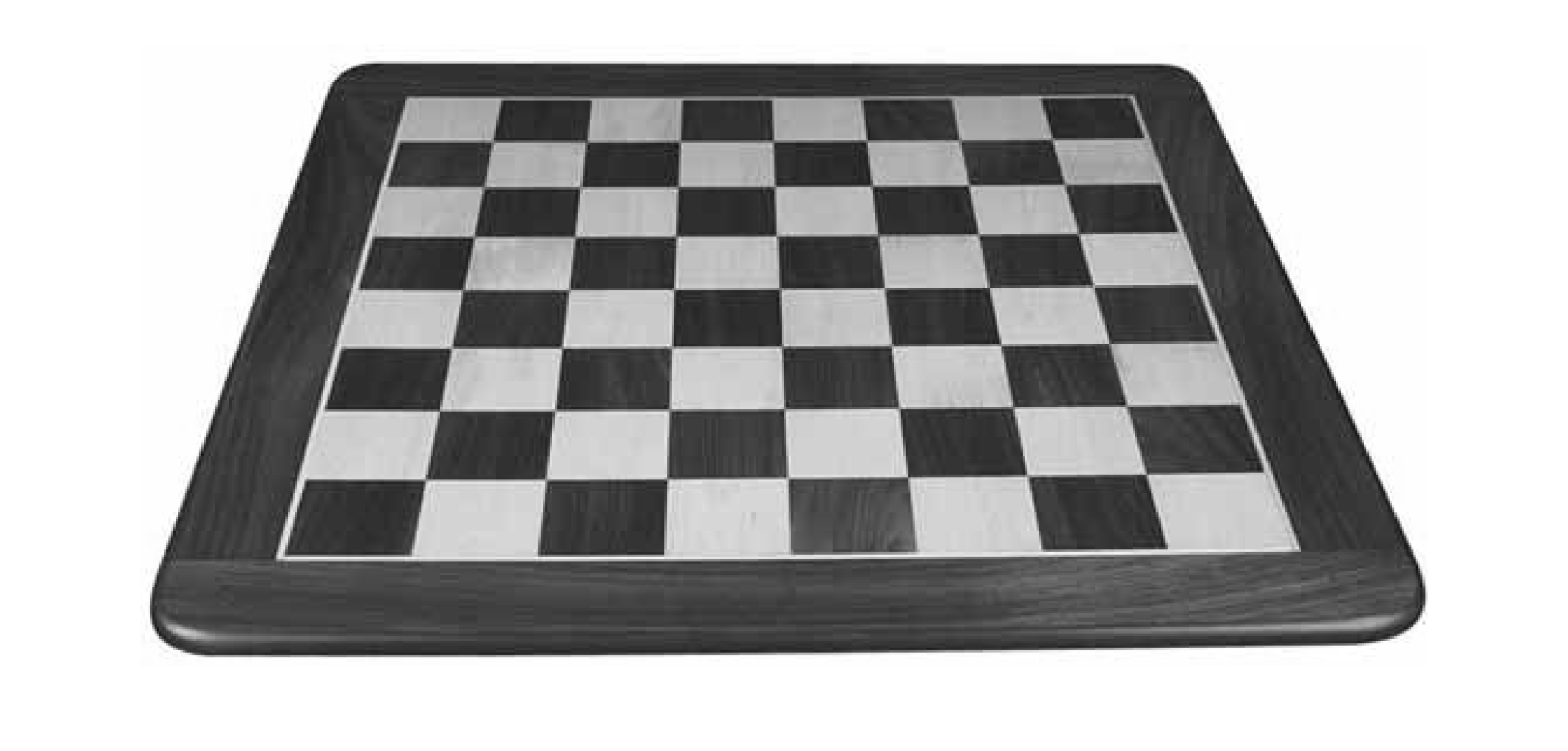
3/15/2015



HW4 ELEC345

# “Cornerness” measure

1. Convert the input image to gray scale.



1. Compute Ix and Iy using central difference x-gradient and y-gradient filters respectively. You had implemented these filters in Assignment 1.

I used 0.5\*[-1,0,1] as central difference operater.

diffx = 0.5\* [-1, 0, 1];

Ix = imfilter(image, diffx, 'symmetric');

1. Compute Ix2, Iy2 and IxIy.

Codes:

Ix2 = Ix.^2;

Iy2 = Iy.^2;

Ixy = Ix.\*Iy;

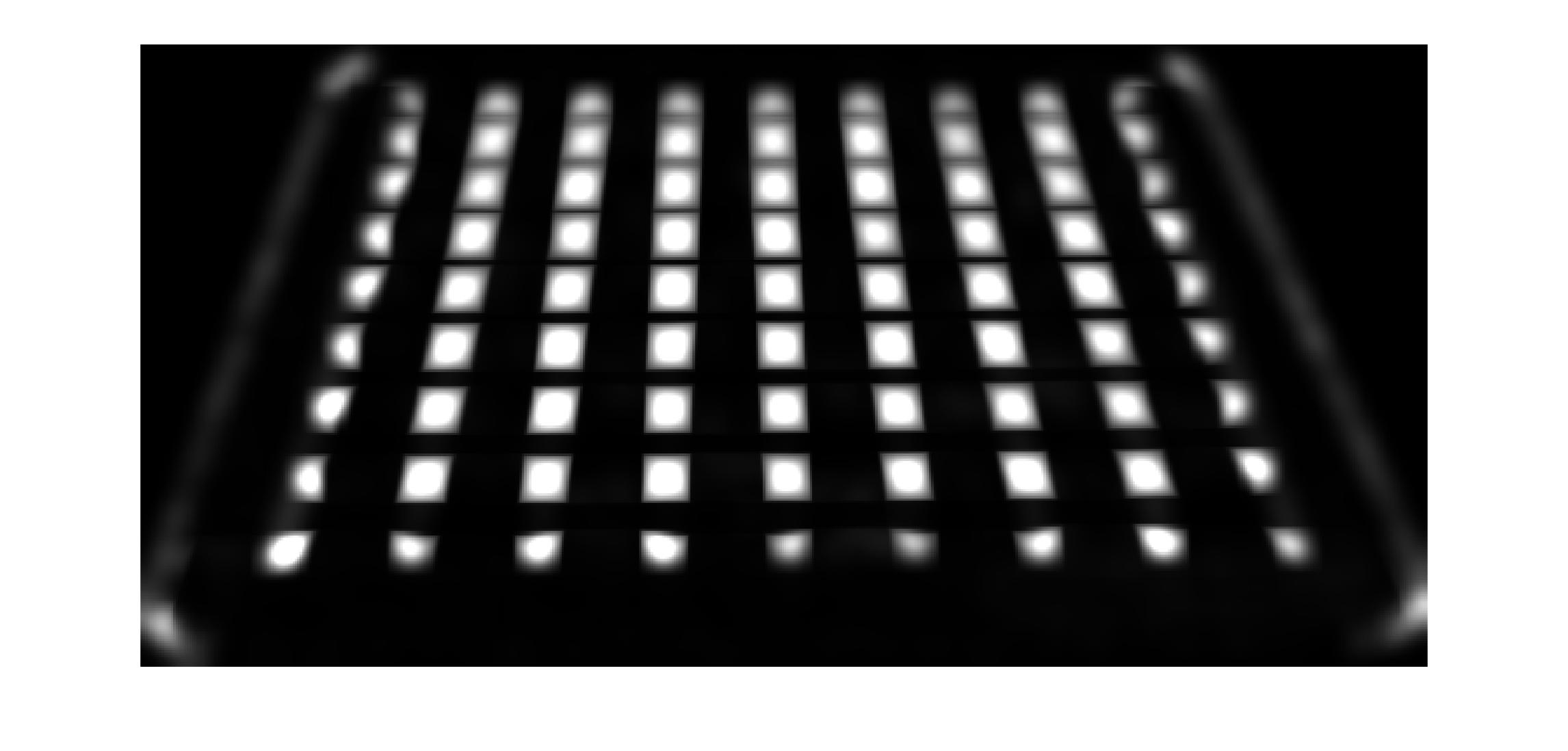
1. Create a Gaussian smoothing filter (w in Equation (1)) using fspecial with a chosen standard deviation (σ) and size 4σ.

gauss = fspecial('gaussian', 4\*sigma, sigma);

1. Apply the Gaussian filter to Ix2, Iy2 and IxIy using imfilter.

convIx2 = imfilter(Ix2, gauss, 'symmetric');

1. Compute the cornerness measure M. Recall that determinant([a, b; b, c]) = ac−b2 and trace([a, b; b, c]) = a + c. M should be map of cornerness, the same size as the image.



Codes:

detH = convIx2.\*convIy2 - convIxy.\*convIxy;

traceH = convIx2 + convIy2;

M = detH./traceH;

M(isnan(M)) = 0;

# Corner extraction

1. Perform non-maximal suppression on M to find local maximas. You could implement the local maxima detection by various means. A few suggested Matlab functions are: ordfilt2, imdilate, imregionalmax.

Thresholding first:

Threshidx = M < Threashold;

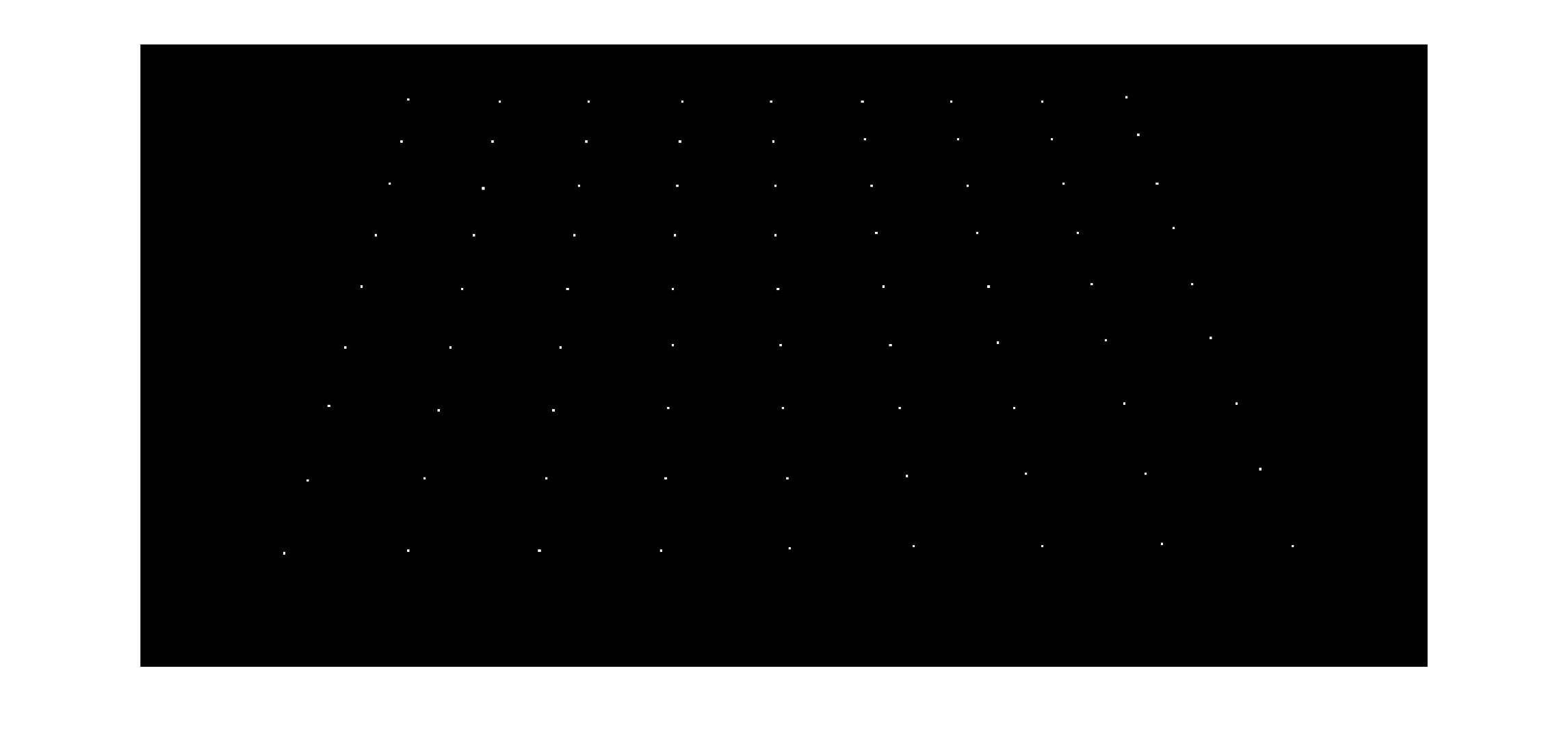
M(Threshidx)=0;

Use mregionalmax to get the corner points:

Pts = imregionalmax(M);

1. Find the coordinates of the corner points.

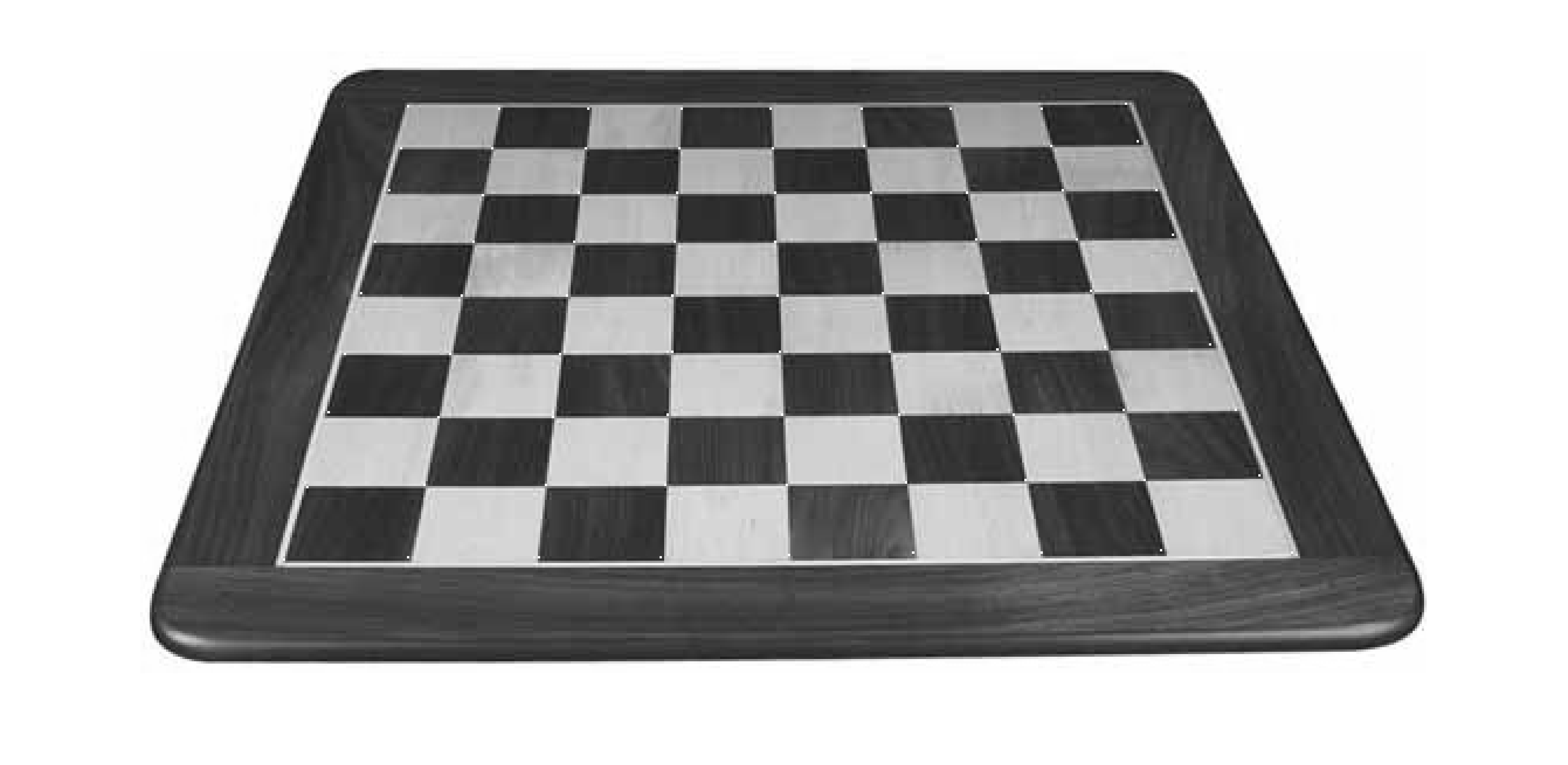
Stored in Pts



1. Display the image and superimpose the corners.

Mpt = double(imregionalmax(M));

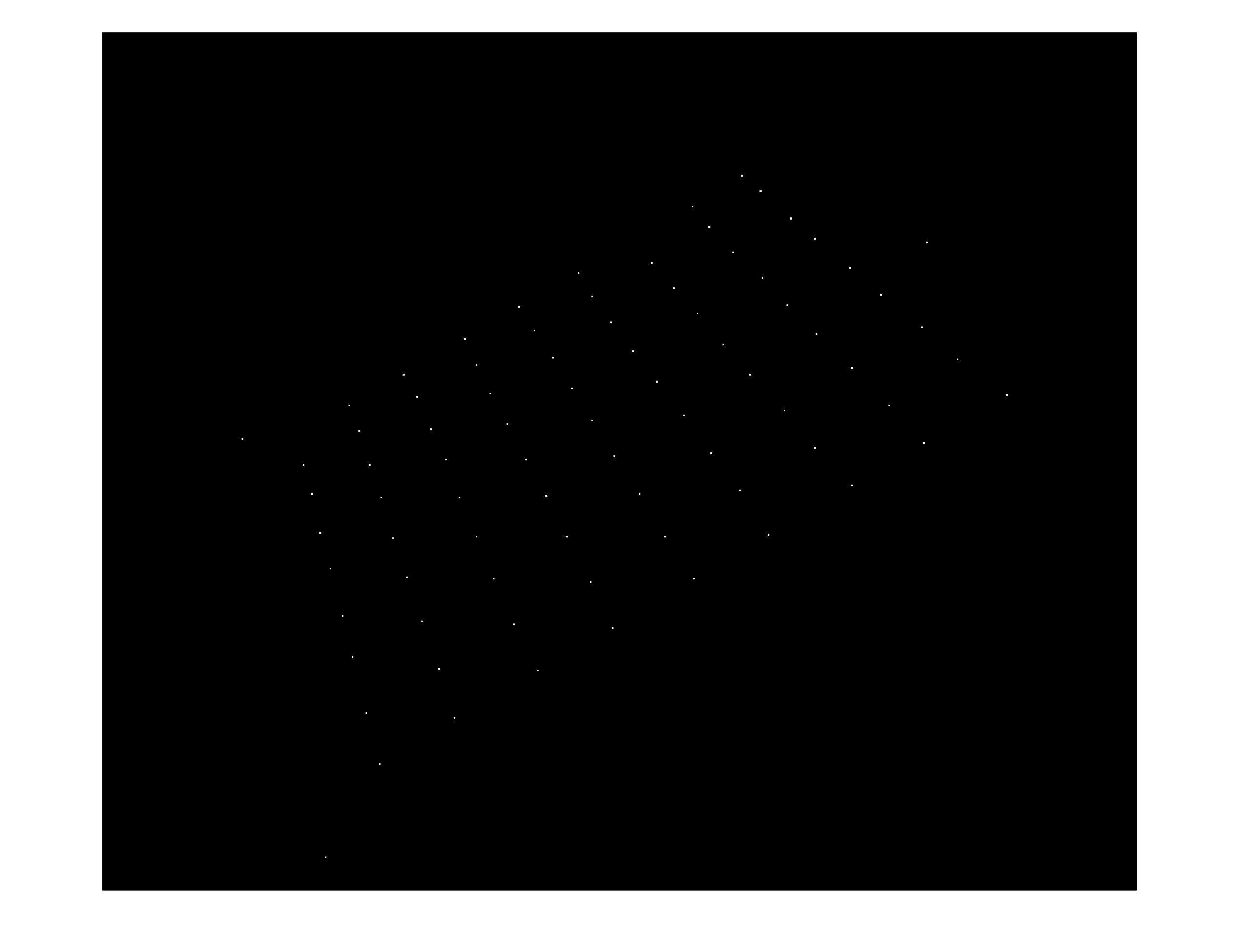
imshow(uint8(Mpt\*255+image));



# Rotation and Scaling

1. Rotate the chessboard image by 30 deg and apply your function.

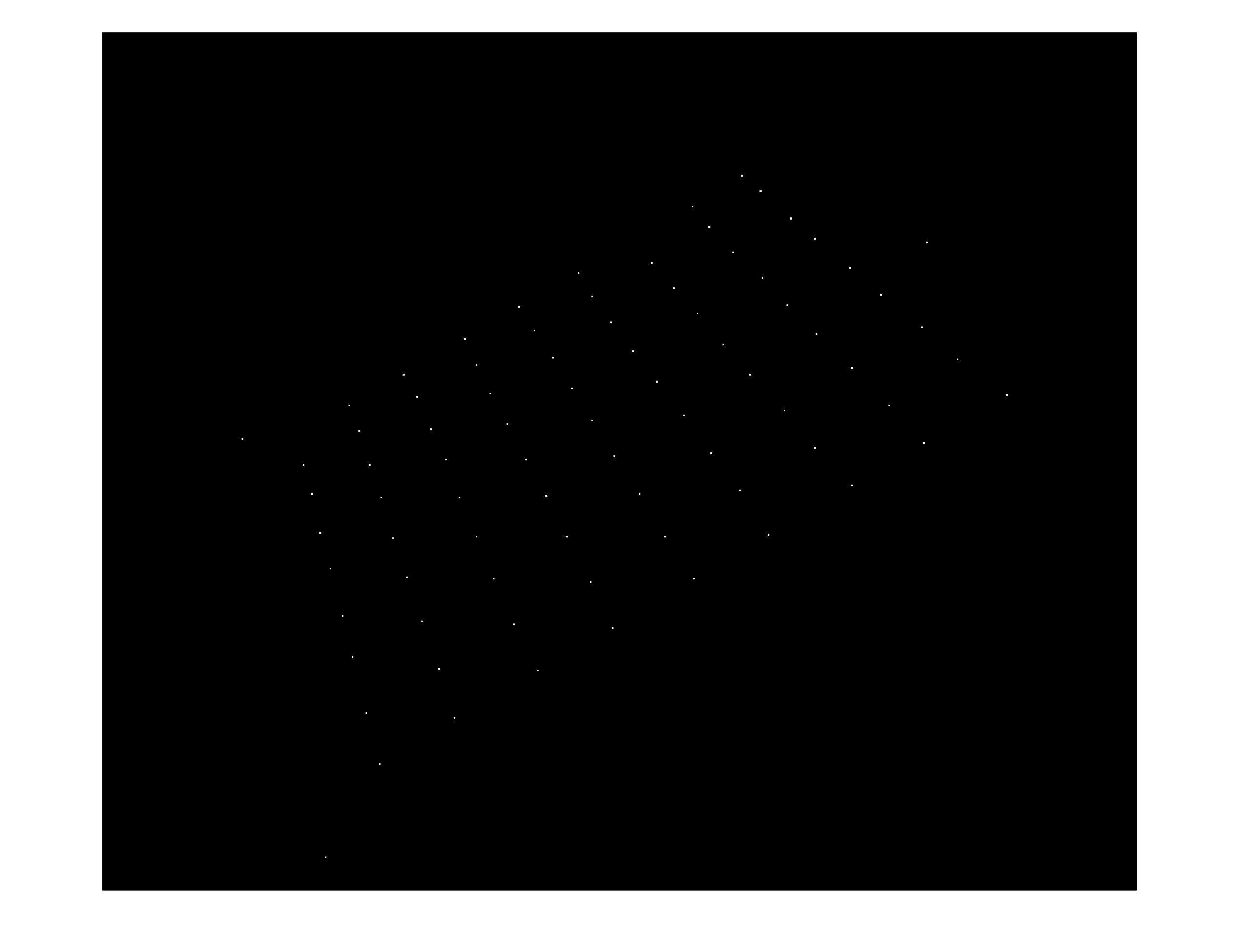
I changed the extra black produced by rotation to white and applied the function. I used a larger Gaussian filter and a higher threshold. The rotated one failed to detect all the corners. Using a better defined derivative might help to improve the result.



2. Resize the chessboard image by 4 times on both axes and apply your function.

I used a much larger Gaussian filter and a really low threshold.

The size change will make Ix^2 16 times smaller than the original figure. Also, the resize smooth things out a bit, so the corner detection is still very good.





Due to resolution of screen, it is hard to see the dots, but all the dots are detected.

# PS: For codes, run hw4.m Function is in Harris.m