1) Compute and display the Harris pixel-wise cornerness function R values for the image checker.jpg using a) Gaussian window/weighting function with a standard deviation of $\sigma_I = 1$ (use 3σ mask size), b) Gaussian Gx,Gy gradients with a standard deviation of $\sigma_D = 0.7$ (use 3σ mask size), and c) trace weighting factor of $\alpha = 0.05$. (Please use the Gaussian smoothing and derivative formulas given earlier in class, and normalize the sum of the smoothing mask to 1 and the sum of the abs derivative masks to 1.) Give the values of R(17:23, 17:23) in your report (these coordinates are for Matlab indices, so subtract 1 if using Python).

Next remove the smaller and negative values in R (anything < 1,000,000). Display the thresholded R using imagesc (scales values to min/max display graylevel). (Note: use double() and not im2double() in your code [as it scales values to 0-1] on checker.jpg.)

Lastly, do non-maximum suppression on R (for this version, keep a location only if a unique maximum is found in its 3x3 region) to identify the actual corner points and display them on the original image. [5 pts]

1.1 R(17:23, 17:23):

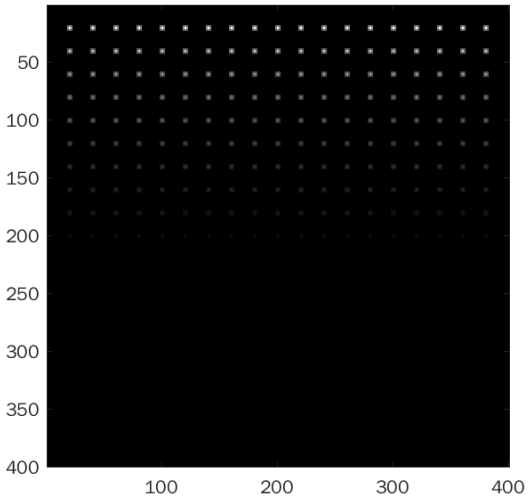
ans =

1.0e+07 *

0.0004	0.0015	-0.0765	-0.4244	-0.4247	-0.0769	0.0014
0.0015	0.0578	0.1736	-0.0115	-0.0136	0.1708	0.0572
-0.0745	0.1733	0.9564	1.3360	1.3285	0.9479	0.1725
-0.4133	-0.0084	1.3329	2.2220	2.2104	1.3229	-0.0045
-0.4132	-0.0113	1.3205	2.2040	2.1925	1.3107	-0.0074
-0.0747	0.1687	0.9388	1.3149	1.3077	0.9306	0.1680
0.0014	0.0561	0.1704	-0.0023	-0.0041	0.1680	0.0555

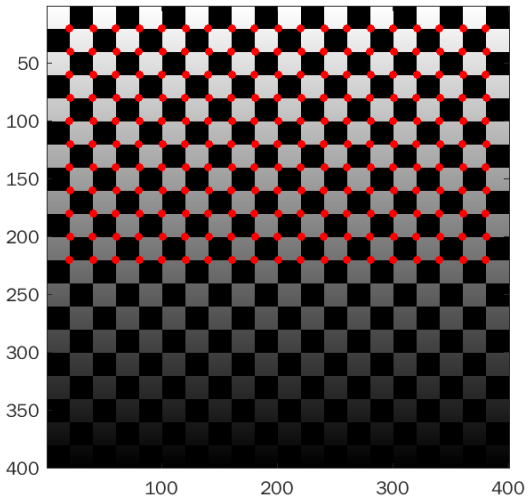
This is R(17:23, 17:23). For gaussian derivatives, I normalize them so that the ABS values of each mask add up to 1. Because I used fspecial, gaussian smoothing has been normalized to 1.

1.2 Thresholded R



These are the points that pass the threshold (1,000,000).

1.3 Non-maximum Suppression



The red dot is the only maximum position that shows "corner". As the bottom of the image darkens, there is no single maximum position.

2) Implement the FAST feature point detector using a radius of $r = 3$ (you can hardcode the particular circle border locations), intensity threshold of $T = 10$, and a consecutive number of points threshold of n*

= 9. Run the detector on the image tower.png. Display the image and overlay the FAST feature points. Repeat with T = {20, 30, 50} and compare all four results. [6 pts]

The first image is at T = 10, the second image is at T = 20, the third image is at T = 30, and the fourth image is at T = 50.

We can see that as the threshold increases, the number of points decreases. In addition, the remaining points are more likely to be "corner" points. This is because as T increases, so do the values in the interval [I(x) + T), I(x) -t]. So, as T gets smaller, there are more points, and the more likely those points are to be “edge”.

3)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Problem 1
myIm = double(imread('checker.png'));

[Gx, Gy] = gaussDeriv2D(0.7);
Gx = Gx ./ sum(abs(Gx), 'all');
Gy = Gy ./ sum(abs(Gy), 'all');
Ix = imfilter(myIm, Gx, 'replicate');
Iy = imfilter(myIm, Gy, 'replicate');

Ix2 = Ix.^2;
Iy2 = Iy.^2;
IxIy = Ix .* Iy;

sigma=1;
G = fspecial('gaussian', 2*ceil(3*sigma)+1, sigma);
gIx2 = imfilter(Ix2, G, 'replicate');
gIy2 = imfilter(Iy2, G, 'replicate');
gIxIy = imfilter(IxIy, G, 'replicate');

R = gIx2 .* gIy2 - gIxIy .^ 2 - 0.05 .* ((gIx2 + gIy2) .^2);

R(17:23, 17:23)

imagesc(R(17:23, 17:23));
colormap('gray');

sum(abs(Gy),'all');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Remove
%Threshold



[x,y] = size(R);
for i = 1: x
    for j = 1 : y
        if R(i,j) < 1000000
            RemovedR(i,j) = 0;
        else
            RemovedR(i,j) = R(i,j);
```

```
        end
    end
end

imagesc(RemovedR);
colormap('gray');
axis('image');
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%unique max
nonMaxX = [];
nonMaxY = [];
for i = 1+1: x-1
    for j = 1+1: y-1
        neibor = RemovedR(i-1:i+1, j-1:j+1);
        count = 0;
        for rN = 1:3
            for cN = 1:3
                if neibor(rN, cN) == RemovedR(i,j)
                    count = count+1;
                end
            end
        end

        if RemovedR(i,j) == max(neibor, [],'all') & count == 1
            nonMax(i,j) = RemovedR(i,j);
            nonMaxX(end+1) = i;
            nonMaxY(end+1) = j;
        else
            nonMax(i,j) = 0;
        end
    end
end

figure;

imagesc(myIm);
colormap('gray');
axis('image');
hold on;
plot(nonMaxY,nonMaxX,'r.','MarkerSize',15);
hold off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Problem 2

myIm = imread('tower.png');

[fastX10, fastY10] = fast(myIm, 10);
[fastX20, fastY20] = fast(myIm, 20);
[fastX30, fastY30] = fast(myIm, 30);
[fastX50, fastY50] = fast(myIm, 50);



figure;
imshow(myIm);
```

```
hold on;
plot(fastY10,fastX10,'r.');
hold off;

figure;
imshow(myIm);
hold on;
plot(fastY20,fastX20,'r.');
hold off;

figure;
imshow(myIm);
hold on;
plot(fastY30,fastX30,'r.');
hold off;


figure;

imshow(myIm);
hold on;
plot(fastY50,fastX50,'r.');
hold off;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Problem 2 function
function [fastX, fastY] = fast(myIm, T)
[x,y] = size(myIm);
r = 3;
n = 9;
fastX = [];
fastY = [];
for i = 4:x-3
    for j = 4:y-3
        upper = myIm(i,j) + T;
        lower = myIm(i,j) - T;
        grayScales = [myIm(i-3,j), myIm(i-3,j+1), myIm(i-2,j+2),
myIm(i-1,j+3),...
                      myIm(i,j+3), myIm(i+1,j+3), myIm(i+2,j+2),
myIm(i+3,j+1),...
                      myIm(i+3,j), myIm(i+3,j-1), myIm(i+2,j-2),
myIm(i+1,j-3),...
                      myIm(i,j-3), myIm(i-1,j-3), myIm(i-2,j-2),
myIm(i-3,j-1)];
        nu = 0;
        nl = 0;
        added = 0;
        for k = 1:16
            if grayScales(k) > upper
                nl = 0;
                nu = nu + 1;
                if nu >= n
                    fastX(end+1) = i;
                    fastY(end+1) = j;
                    added = 1;
                    break
                end
```

```
            elseif grayScales(k) < lower
                    nu = 0;
                    nl = nl + 1;
                    if nl >= n
                        fastX(end+1) = i;
                        fastY(end+1) = j;
                        added = 1;
                        break
                    end
            else
                nu = 0;
                nl = 0;
            end
        end


        if grayScales(1) > upper & grayScales(16) > upper & added ==0
            for k = 1:16
                if grayScales(k) > upper
                    nu = nu + 1;
                    if nu >= n
                        fastX(end+1) = i;
                        fastY(end+1) = j;
                        added = 1;
                        break
                    end
                else
                     nu = 0;
                     break
                end
            end
        elseif grayScales(1) < lower & grayScales(16) < lower & added ==0
            for k = 1:16
                if grayScales(k) < lower
                    nl = nl + 1;
                    if nl >= n
                        fastX(end+1) = i;
                        fastY(end+1) = j;
                        added = 1;
                    break
                    end
                else
                    nl = 0;
                    break
                end
            end
        end



    end
end
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Problem 1 Functions
```

```
function [Gx, Gy] = gaussDeriv2D(sigma)


 mask_size = 2*ceil(3*sigma)+1;
 Gx = zeros(mask_size, mask_size);
 Gy = zeros(mask_size, mask_size);
 y = -ceil(3*sigma);
for i = 1: mask_size
     x = -ceil(3*sigma);
    for j = 1:mask_size
        Gx(i,j) = (exp((-1)*(x^2 + y^2)/ (2*sigma^2))) * (((1) *x)/(2*
pi*sigma^4));
       Gy(i,j) = (exp((-1)*(x^2 + y^2)/ (2*sigma^2))) * (((1) *y)/(2*
pi*sigma^4));
        x = x + 1;
    end
    y = y + 1;
end
end
```