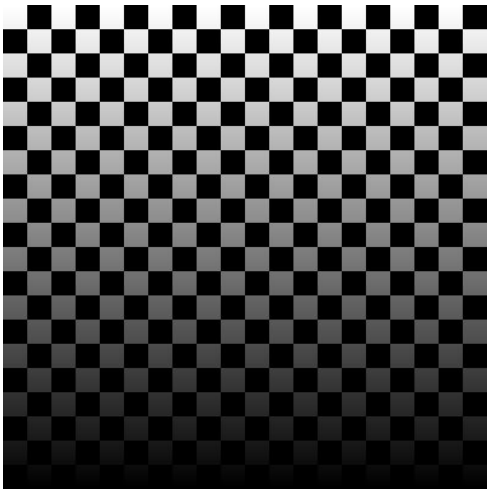
Name: __Tristan Roman (.198)__ Score: /11

CSE 5524 Computer Vision for HCI SP'22

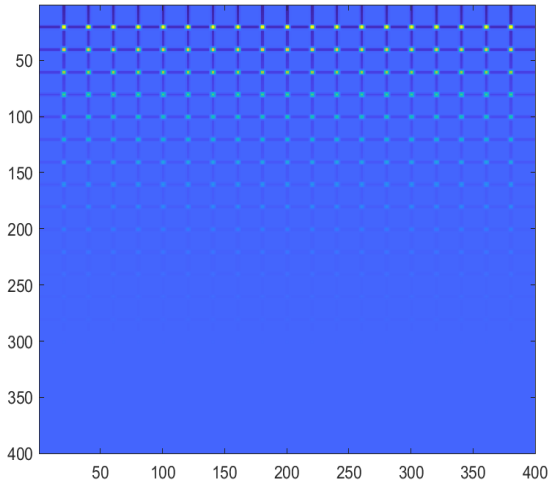
Homework Assignment #8 Due: Thursday 3/24

```
% Tristan Roman
% CSE 5524 – HW8
% 3/24/2022

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

sigmaD = 0.7;
[Gx,Gy] = gaussDeriv2D(sigmaD);
Ix = imfilter(checker,Gx,'replicate');
Iy = imfilter(checker,Gy,'replicate');

% derivative products
Ix2 = Ix.*Ix;
Iy2 = Iy.*Iy;
IxIy = Ix.*Iy;

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

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

if R(i,j)<1000000
    R(i,j) = 0;
end
imagesc(R);

% non-maximal suppression
for i = 2:399
    for j = 2:399
        region = R(i-1:i+1,j-1:j+1);
    end
    x = find(max(max(region)));
    % size 1x1 if unique
    b = size(x);
    if sum(b)~= 2
        R(i,j)=0;
    end
end

% display the corner in the original image
C = imfuse(checker,R);
imagesc(C);
```

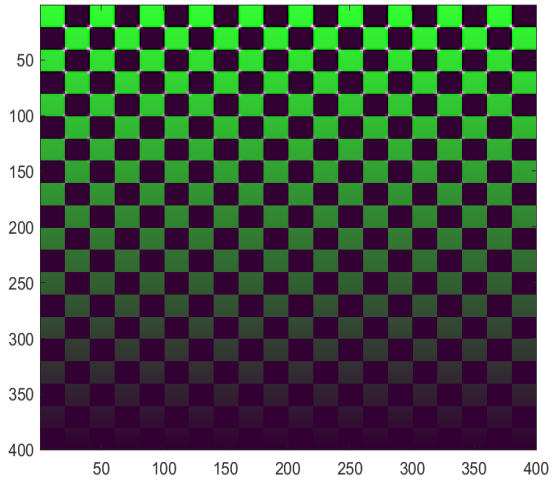
Top is the original image.

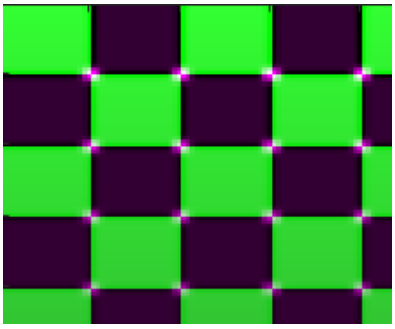


This is the Harris pixel-wise corner algorithm with 3sigma mask size. You can definitely see the distinguishing points where the corners are



This is the overlay after the smoothing mask and removal of small/negative values (better seen below).





```
%% Problem 2
tower = imread('tower.png');
T = {10,20,30,50};
n_star = 9;

[r,c] = size(tower);
% x,y index of interest point
fastX = [];
fastY = [];
for x = 4:r-3
    for y = 4:c-3

        points = edge(x,y); % index of border points
        bordervalues = border(points,tower); % value of each border points
        bordersymbols = zeros(1,16);

        % check each border value
        for i = 1:16
            if bordervalues(i) > tower(x,y)+T{4} % change this for different T value
                bordersymbols(i) = 1;
            elseif bordervalues(i) < tower(x,y)-T{4} % change this for different T
value
                bordersymbols(i) = -1;
            end
        end



        n = check([bordersymbols,bordersymbols]);
        if n >= n_star
            fastX = [fastX,y];
            fastY = [fastY,x];
        end
    end

end

figure;
imshow(tower);
hold on;
plot(fastX,fastY,'r.');
hold off;
```

This is the resulting image:

It seems most apparent corners and sharp edges are identified by the algorithm properly!

```
%% helper functions

function result = edge(x,y) % border index of home pixel
    result = {[x-3,y],[x-3,y+1],[x-2,y+2],[x-
1,y+3],[x,y+3],[x+1,y+3],[x+2,y+2],[x+3,y+1],[x+3,y],[x+3,y-1],[x+2,y-2],[x+1,y-
3],[x,y-3],[x-1,y-3],[x-2,y-2],[x-3,y-1]};
end

%%
function result = border(points,tower) % pixel values of the border
    result = zeros(1,16);
    for i = 1: 16
        index = points{i};
        result(i) = tower(index(1),index(2));
    end
end

%%
function result = check(bordersymbols) % check number of contiguous points
    num1 = 0;
    num0 = 0;
    counter = 0;
    for i = 1:32
        if bordersymbols(i) == 1
            counter = counter +1;
        else
            if counter > num1
                num1 = counter;
            end
            counter = 0;
        end
    end
    counter = 0;
    for i = 1:32
        if bordersymbols(i) == -1
            counter = counter + 1;
        else
            if counter > num0
                num0 = counter;
            end
            counter = 0;
        end
    end
    result = max(num1,num0);
end

%%
function [Gx, Gy] = gaussDeriv2D(sigma) % samples 3 standard devs from Gaussian
    Gx = ones(2*ceil(3*sigma) + 1,2*ceil(3*sigma) + 1); % mask size
    Gy = ones(2*ceil(3*sigma) + 1,2*ceil(3*sigma) + 1); % mask size
    range = [-ceil(3*sigma): ceil(3*sigma)];
    for x = 1:2*ceil(3*sigma)+1
        for y = 1:2*ceil(3*sigma)+1
            Gx(x,y) = range(y)*exp(-
(range(x)^2+range(y)^2)/(2*sigma^2))/(2*pi*sigma^4);
```

```
            Gy(x,y) = range(x)*exp(-
(range(x)^2+range(y)^2)/(2*sigma^2))/(2*pi*sigma^4);
        end
    end
end
```