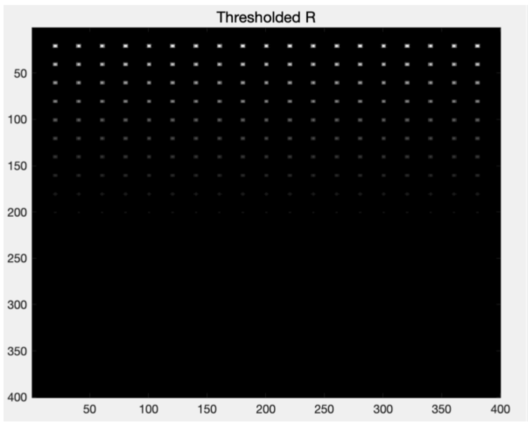
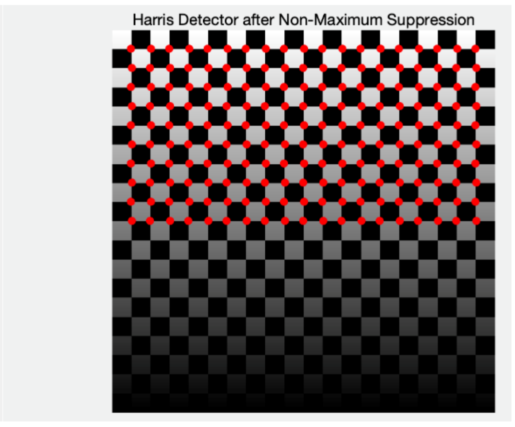
Yukun Duan
Computer Vision for HCI AU'22
Homework Assignment #8

1.
Here is the value of R (17:23, 17:23):

```
    0.0003    0.0011   -0.0543   -0.3012   -0.3014   -0.0546    0.0010
    0.0011    0.0411    0.1232   -0.0082   -0.0096    0.1212    0.0406
   -0.0529    0.1230    0.6787    0.9481    0.9428    0.6727    0.1224
   -0.2933   -0.0060    0.9459    1.5769    1.5686    0.9389   -0.0032
   -0.2932   -0.0080    0.9371    1.5642    1.5559    0.9302   -0.0053
   -0.0530    0.1198    0.6663    0.9332    0.9281    0.6604    0.1192
    0.0010    0.0398    0.1209   -0.0016   -0.0029    0.1192    0.0394
```



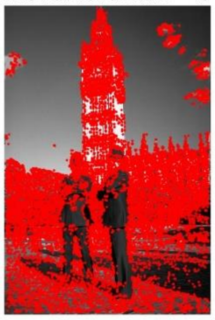


We can see there the points at the center are significantly larger than those outside. For example, there are 4 points over $1e^7$ around the center, which will be kept after thresholding. Other values will be set to 0 after thresholding. The first image shows the R value after thresholding. We can see little white dots printed in the crossing, and the intensity is decreasing from the top to the bottom. The second image is the result of the harries detector after applying the threshold and non-maximum suppression. The result is printed on the original image as red dots. At each crossing, there is a red dot around the center closely (sometimes left, sometimes right, depending on the pixel value). There is no dot at the bottom because the pixel value is decreasing from top to bottom, and we only keep values that are larger than 1000000. Starting from line 11, there is no value over 1000000 thus no dots are shown.

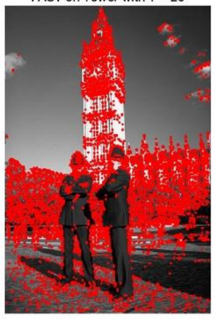
2.
The FAST that I implemented has T = [10, 20, 30, 50] and n* = 9. The results are shown below. When the threshold is at 10, FAST picks up the most interest points because it's easy to find a circle with more than 9 successive pixels above or below the upper/baseline. As the threshold increases, there will always be fewer points being picked up compared to fewer thresholds. We can see the red dots on the ground are becoming less and less, and most of

the dots are fading as the threshold goes to 50. There are some dots around the shadow of two people as there is an obvious difference between shades and areas out of shades. Some lines and corners on the tower are recognized by FAST, and there are some dots in the background (trees) because leaves turn out to be in different intensities.

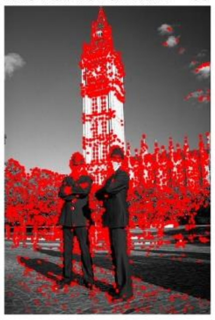
FAST on Tower with T = 10



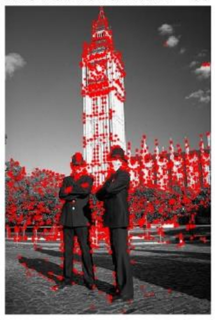
FAST on Tower with T = 20



FAST on Tower with T = 30



FAST on Tower with T = 50



Attached Code:

```
% Yukun Duan
% CSE5524 - HW8
% 10/22/2022


%% Problem 1
image = double(imread("checker.png"));
sigma_window = 1; sigma_grad = 0.7; alpha = 0.05; T = 1000000;

% gaussian window with sigma = 1, s * sigma mask
G = fspecial('gaussian', 2*ceil(3*sigma_window)+1, sigma_window);
[Gx, Gy] = gaussDeriv2D(sigma_grad);

Ix = imfilter(image, Gx, 'replicate');
Iy = imfilter(image, Gy, 'replicate');
Ixy = Ix .* Iy;
```

```matlab
gIx2 = imfilter(Ix.^2, G, 'replicate');
gIy2 = imfilter(Iy.^2, G, 'replicate');
gIxy = imfilter(Ixy, G, 'replicate');

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

disp(R(17:23, 17:23))

R(R<T) = 0;
imagesc(R)
colormap('gray')
title('Thresholded R','FontSize', 14)
pause;
saveas(gcf,'ThresholdedR.jpg')

duplicate = R;
for r = 1:size(R, 1)
    for c = 1:size(R, 2)
        w = getWindow(3, duplicate, r, c);
        if duplicate(r, c) ~= max(w, [], 'all')
            R(r, c) = 0;
        end
    end
end

imshow(image/255, 'InitialMagnification','fit')
hold on
[y, x] = find(R);
plot(x, y, 'r.', 'MarkerSize', 20)
title('Harris Detector after Non-Maximum Suppression', 'FontSize',14)
saveas(gcf,'Detector.jpg')
hold off
pause;

%% Problem 2
img = double(imread("tower.png"));
T_list = [10 20 30 50]; n = 9;

for T = T_list
    fast = getFAST(img, T, n);
    imshow(img/255, 'InitialMagnification','fit')
    hold on
    [y, x] = find(fast);
    plot(x, y, 'r.')
    title(sprintf('FAST on Tower with T = %i', T), 'FontSize',14)
    saveas(gcf, sprintf('FAST_T%i.jpg', T))
    hold off
end


%% Gaussian derivative function from HW2
function [Gx, Gy] = gaussDeriv2D(sigma)
    length = 2 * ceil(sigma * 3) + 1;
    for r = 1:length
        for c = 1:length
            y = -r + ceil(3*sigma) + 1;
            x = c - ceil(3*sigma) - 1;
            Gx(r,c) = -x * exp(-1 * (x^2 + y^2)/(2 * sigma.^2)) / (2 * pi *
sigma^4);
            Gy(r,c) = -y * exp(-1 * (x^2 + y^2)/(2 * sigma.^2)) / (2 * pi *
sigma^4);
        end
    end
end


%% Get the square surrounded img(r, c) with length = len
function window = getWindow(len, img, r, c)
    left = max(1, c - floor(len/2));
    right = min(size(img, 2), c + floor(len/2));
    top = max(1, r - floor(len/2));
    bottom = min(size(img, 1), r + floor(len/2));
    window = img(top:bottom, left:right);
end


%% Hardcode the border with r = 3.
function border = getBoarder(img, r, c)
    top = img(r-3, c-1:c+1);
    bottom = img(r+3, c-1:c+1);
    left = img(r-1:r+1, c-3);
    right = img(r-1:r+1, c+3);
    lt = img(r-2, c-2);
    rt = img(r-2, c+2);
    lb = img(r+2, c-2);
    rb = img(r+2, c+2);
    border = [top, rt, right', rb, flip(bottom), lb, flip(left'), lt];
end
```

```matlab
%% Check if the array satisfy n >= n*.
function res = verifyList(category, n)
    list = [category, category];
    l = 1; temp = 1;
    for i = 2:size(list, 2)
        if list(i-1) == list(i) && list(i) ~= 0
            temp = temp + 1;
        else
            if temp > l
                l = temp;
            end
            temp = 1;
        end
    end
    res = l >= n;
end

%% Get FAST feature points matrix
function fast = getFAST(img, T, n)
    fast = zeros(size(img));
    for r = 4:size(img,1)-3
        for c = 4:size(img,2)-3
            b = getBoarder(img, r, c);
            category = zeros(size(b));
            category(b > img(r,c)+T) = 1;
            category(b < img(r,c)-T) = -1;
            fast(r,c) = verifyList(category, n);
        end
    end
end
```