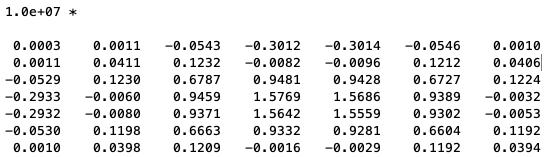
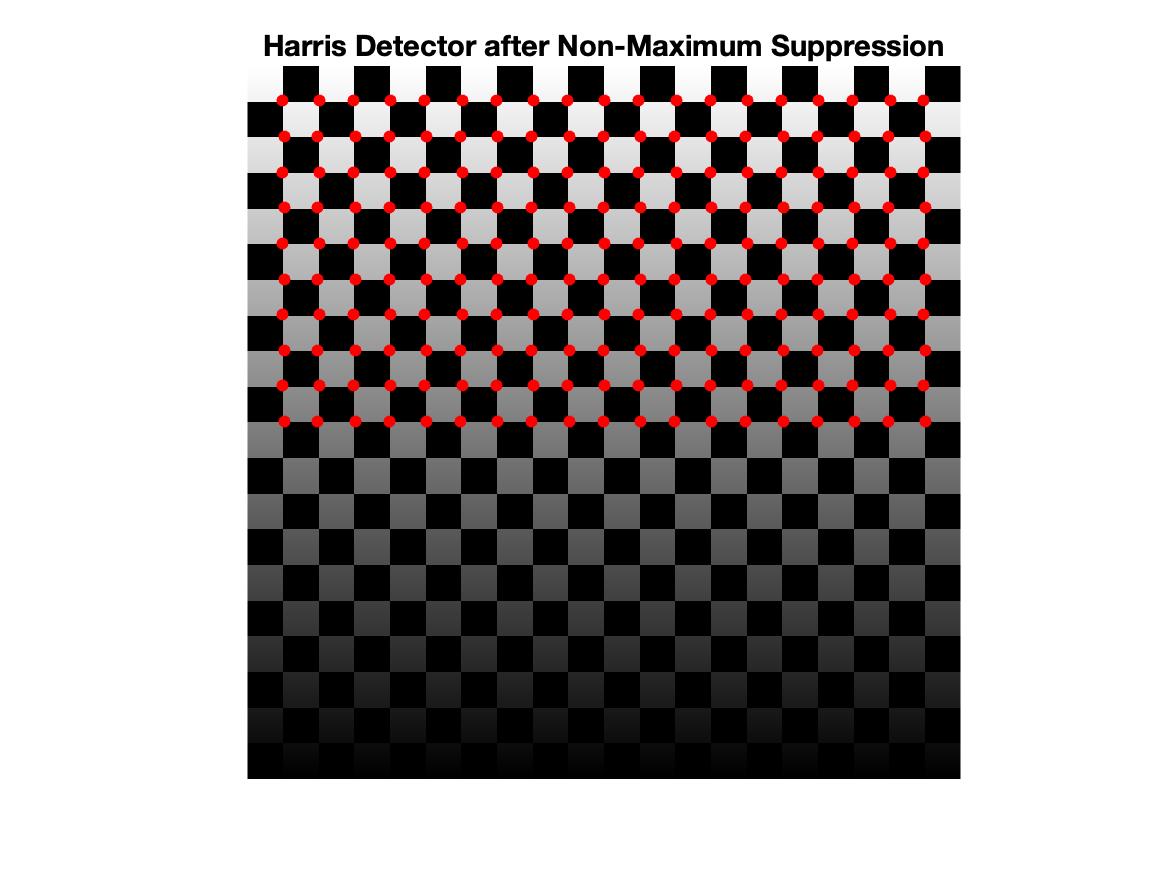
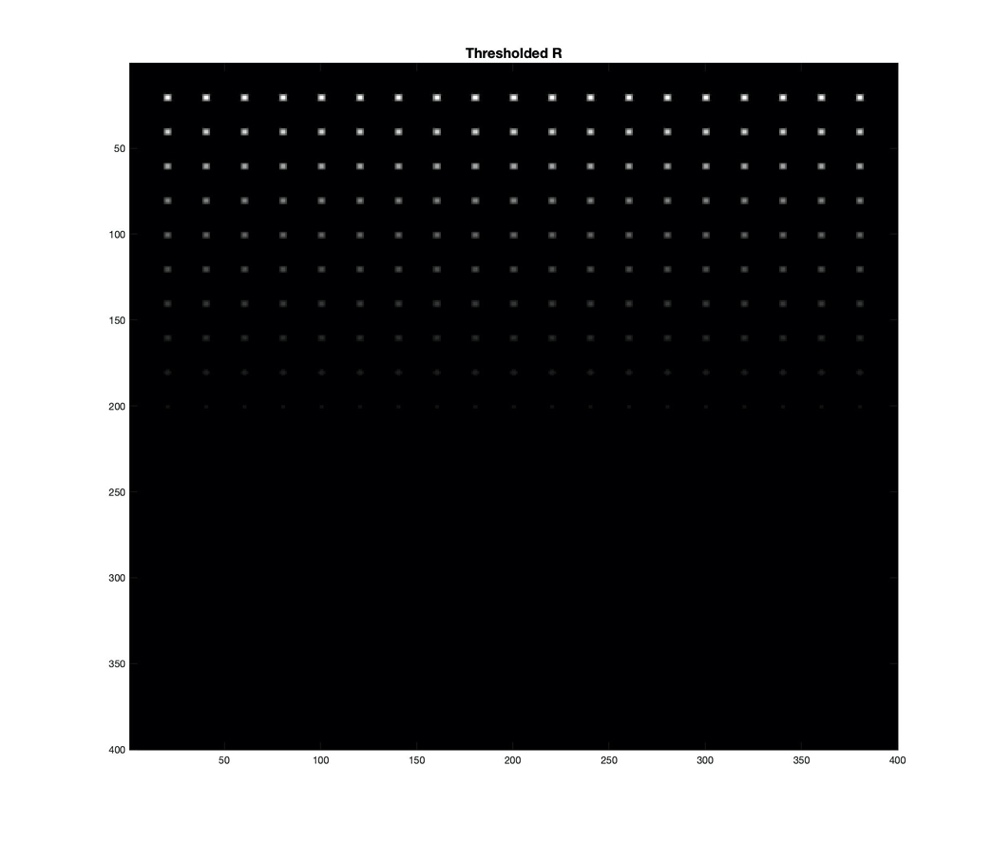
CSE 5522 HW8

Professor: Jim Davis

Yi Zhao

1. Here is the value of R (17:23, 17:23). You can see there the points at the center are significantly larger than those outside. For example, there are 4 points over 1e7 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. You can see little white dots are printed in the crossing, and the intensity are decreasing from top to the bottom. The second image is the result of harries detector after applying the threshold and non-maximum suppression. The result is printed on origin image as red dots. At each crossing, there is a red dot around the center closely (sometimes left, sometimes right, depends 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. Start from line 11, there is no value over 1000000 thus no dots are shown.





1. The FAST that I implemented has T = [10, 20, 30, 50, 100, 150] and n\* = 9. The results are shown below. When threshold is at 10, FAST pick up most interest points because it’s easy to find a circle with more than 9 successive pixels above or below the upper/baseline. As threshold increases, there will always be less points being picked up compared to less threshold. You can see the red dots on the ground is becoming less and less, and most of dots are faded 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 area out of shades. Some lines and corner on the tower are recognized by FAST, and there are some dots in the background (trees) because leaves turn to be in different intensity.



% Author: Yi Zhao

% CSE 5524, HW8

% 10/26/2019

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Problem 1

image = double(imread("./data/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;

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)

title('Thresholded R','FontSize', 14)

pause;

saveas(gcf,'./output/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,'./output/Detector.jpg')

hold off

pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Problem 2

img = double(imread("./data/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('./output/FAST\_T%i.jpg', T))

hold off

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% Helper Function

% 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

% 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