CSE 5524 HW8

**Problem 1**

For grading purpose, here is the value of R(17:23,17:23)

ans =

1.0e+07 \*

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

After remove smaller and negative values

A picture containing computer

Description automatically generated

Actual corner points after performing non-maximum suppression:

Background pattern

Description automatically generated

**Problem 2**

A picture containing room, hydrant

Description automatically generated A picture containing shape

Description automatically generated

T = 10 T = 20

A picture containing hydrant, standing

Description automatically generated A picture containing photo, standing, person, different

Description automatically generated

T = 30 T = 50

The result looks reasonable since when we increase the T, we get fewer interest points.

Code:

%% Problem 1

checker = double(imread('checker.png'));

% image derivatives

sigmaD = 0.7;

[Gx,Gy] = gaussDeriv2D(sigmaD);

Ix = imfilter(checker,Gx,'replicate');

Iy = imfilter(checker,Gy,'replicate');

% multiply derivatives

Ix2 = Ix.\*Ix;

Iy2 = Iy.\*Iy;

IxIy = Ix.\*Iy;

% gaussian blur

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;

%%

for i = 1:400

for j = 1:400

if R(i,j)<1000000

R(i,j) = 0;

end

end

end

imagesc(R);

%% perform 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)));

b = size(x); % if maximum is unique, size should be [1,1]

if sum(b)~= 2

R(i,j)=0;

end

end

%% display the corner in the original image

C = imfuse(checker,R);

imagesc(C);

%% 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 = border(x,y); % index of border points

bordervalues = wrap(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;

%% helper functions

% hard code the border index of home pixel

function result = border(x,y)

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

% wrapped arrary of pixel values of the border

function result = wrap(points,tower)

result = zeros(1,16);

for i = 1: 16

index = points{i};

result(i) = tower(index(1),index(2));

end

end

% check the maximum number of contiguous points

function result = check(bordersymbols)

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)

% this function samples 3 std from the 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