Harris Corner Detector: Non-maximum Suppression

Image features such as Harris Corners can serve as a compact image representation useful for task such as image matching, computing image statistics, 3D model estimation and video tracking. In this lab you will build a Harris Corner Detector via the implementation sketch in video Robo_2_4.Otherfeatures_pt2_v1_Good.mp4.

In the last section you wrote a function to compute a corner score for each pixel in the image. In this section you will perform non-maximum suppression and thresholding on your previous solution to isolate the image locations with the strongest corner scores.

Your Script

Save Reset MATLAB Documentation (https://www.mathworks.com/help/)

```
img = imread('peppers.png');
img_gray = double(rgb2gray(img));

img_gray_smooth = gauss_blur(img_gray);
[I_x,I_y] = grad2d(img_gray_smooth);

I_xx = gauss_blur(I_x.*I_x);
I_yy = gauss_blur(I_y.*I_y);
I_xy = gauss_blur(I_x.*I_y);

k = 0.06;
R = (I_xx.*I_yy - I_xy.*I_xy) - k*(I_xx+I_yy).^2;

r = 5;
thresh = 10000;
hc = nmsup(R,r,thresh);

figure()
imshow(img)
hold on;
plot(hc(:,1), hc(:,2), 'rx')
hold off;

function loc = nmsup(R,r,thresh)
    %% Step 1-2 must be performed in a way that allows you to
    %% preserve location information for each corner.
    [sy,sx] = size(R);
    [x,y] = meshgrid(1:sx,1:sy);

    %% Step 1: eliminate values below the specified threshold.



    %% Step 2: Sort the remaining values in decreasing order.


    %% Step 3: Starting with the highest scoring corner value, if
    %% there are corners within its r neighborhood remove
    %% them since their scores are lower than that of the corner currently
    %% considered. This is true since the corners are sorted
    %% according to their score and in decreasing order.


    %% The variable loc should contain the sorted corner locations which
    %% survive thresholding and non-maximum suppression with
    %% size(loc): nx2
    %% loc(:,1): x location
    %% loc(:,2): y location
    maxima_matrix = ordfilt2(R,25,ones(5));
    mask = (R==maxima_matrix) & (R>thresh);
    maxima_matrix = R .* mask;
    [sort_zero, i] = sort(maxima_matrix(:),'descend');
```

```
    %% The variable loc should contain the sorted corner locations which
    %% survive thresholding and non-maximum suppression with
    %% size(loc): nx2
    %% loc(:,1): x location
    %% loc(:,2): y location

    [a, b] = ind2sub([sy sx], i);

    loc = [];
    val = sort_zero(1);
    sort_ind = 1;
    while val > 0
      col = b(sort_ind);
      row = a(sort_ind);

      % enter in loc
      loc = [loc; col row];

      % for next iteration
      sort_ind = sort_ind + 1;
      val = sort_zero(sort_ind);
    end
    display(length(loc));
end

function [I_x,I_y] = grad2d(img)
        %% compute image gradients in the x direction
        %% convolve the image with the derivative filter from the lecture
        %% using the conv2 function and the 'same' option
        dx_filter = [1/2 0 -1/2];
        I_x = conv2(img, dx_filter, 'same');

        %% compute image gradients in the y direction
        %% convolve the image with the derivative filter from the lecture
        %% using the conv2 function and the 'same' option
        dy_filter = dx_filter';
        I_y = conv2(img, dy_filter, 'same');
end

function smooth = gauss_blur(img)
    %% Since the Gaussian filter is separable in x and y we can perform Gaussian smoothing by
    %% convolving the input image with a 1D Gaussian filter in the x direction then
    %% convolving the output of this operation with the same 1D Gaussian filter in the y direction.

    %% Gaussian filter of size 5
    %% the Gaussian function is defined f(x) = 1/(sqrt(2*pi)*sigma)*exp(-x.^2/(2*sigma))
    x =  -2:2;
    sigma = 1;

    % my soln: I use fspecial('gaussian', hsize = [1 5], sigma)
    %gauss_filter_x = fspecial('gaussian', gauss_size, sigma);
    %gauss_filter_y = fspecial('gaussian', gauss_size', sigma);
    %smooth_x = imfilter(img, gauss_filter_x);
    %% convolve smooth_x with the transpose of the Gaussian filter
    %smooth = imfilter(smooth_x, gauss_filter_y);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % for edX class:
    gauss_filter = 1/(sqrt(2*pi)*sigma)*exp(-x.^2/(2*sigma^2));

    %% using the conv2 function and the 'same' option
    %% convolve the input image with the Gaussian filter in the x
    smooth_x = conv2(img, gauss_filter, 'same');
```

```
    %% convolve smooth_x with the transpose of the Gaussian filter
    smooth = conv2(smooth_x, gauss_filter', 'same');
end

```

Run Script

Previous Assessment: Incorrect

Submit

Is the estimated corner score R correct?

Are the estimated corner locations correct?

Variable correct has an incorrect value.

Output

37