Hough transform function code

MAIN-

**%% Load image**

close all;

clc;

img = imread('building.jpg'); % already grayscale

**%% Find edges**

img = img(:,:,1);

figure();

imshow(img);

title('Original Image');

img\_filtered = imgaussfilt(img,9); %Applying gaussian filter for removing noise and smoothing

figure();

imshow(img\_filtered);

title('Filtered Image');

img\_edges = edge(img, 'Canny'); %Edge detection of original image using Canny filter

figure();

imshow(img\_edges);

title('Edges found in original image');

filtered\_edges = edge(img\_filtered, 'Canny'); %Edge detection of filtered image using Canny filter

figure();

imshow(filtered\_edges);

title('Edges found in filtered image');

**%% Performing Hough Transform**

[H, theta, rho] = hough\_lines(filtered\_edges);

**%% Plot accumulator array H/Hough transform**

figure();

imshow(imadjust(mat2gray(H)),'XData',theta,'YData',rho,...

'InitialMagnification','fit');

title('Hough transform');

xlabel('\theta'), ylabel('\rho');

axis on, axis normal, hold on;

colormap(hot); %displays hough transform

FUNCTION(hough\_lines)-

function [H, theta, rho] = hough\_lines(BW, varargin)

% Compute Hough accumulator array for finding lines.

% BW: Binary (black and white) image containing edge pixels

p = inputParser();

addParameter(p, 'RhoResolution', 1); % RhoResolution: Difference between successive rho values, in pixel

addParameter(p, 'Theta', linspace(-90, 89, 180)); % Theta: Vector of theta values to use, in degrees

parse(p, varargin{:});

rhoStep = p.Results.RhoResolution;

theta = p.Results.Theta;

MaxRho = sqrt((size(BW,1) - 1) ^ 2 + (size(BW,2) - 1) ^ 2); %Maximum value of rho

numRho = 2 \* (ceil(MaxRho / rhoStep)) + 1; %increment in rho between successive entries

diagonal = rhoStep \* ceil(MaxRho / rhoStep); % rho ranges from -diagonal to diagonal

numTheta = length(theta);

H = zeros(numRho, numTheta); %Matrix of size numRho by numTheta .Rows of H should correspond to values of rho,and columns to those of theta.

rho = -diagonal : diagonal;

for i = 1 : size(BW,1)

for j = 1 : size(BW,2)

if (BW(i, j)) %for each non-zero point

for k = 1 : numTheta

temp = j \* cos(theta(k) \* pi / 180) + i \* sin(theta(k) \* pi / 180);

rohIndex = round((temp + diagonal) / rhoStep) + 1;

H(rohIndex, k) = H(rohIndex, k) + 1;

end

end

end

end

end