```
>> demo_Hough_Circle
clc; close all; clear all;

alphaBetaSpacing = 1;
radius = 23;
thetaSpacing= pi/8;

hFig = figure;
set (hFig, 'Units', 'normalized', 'Position', [0,0,1,1]);

im= imread('circle1.png');
im= rgb2gray(im);
%im= imresize(im, 0.5);

gradientImY= imfilter(double(im), fspecial('sobel'),
'replicate');
gradientImX= imfilter(double(im), fspecial('sobel')',
'replicate');

[H, betas, alphas,
expected_average_H_for_uniform_edge_distribution] =
Hough_circles(gradientImY, gradientImX, alphaBetaSpacing, radius,
thetaSpacing);

max_y= NaN;
max_x= NaN;
max_H= -inf;
for i = 1:length(betas)
    for j = 1:length(alphas)
        if (H(i,j)>max_H)
            max_H = H(i,j);
            max_y = i;
            max_x = j;
        end
    end
end
max_x=max_x*alphaBetaSpacing;
max_y=max_y*alphaBetaSpacing;
%Orginal
subplot(2,3,1);
imagesc(im);
colormap('gray'); colorbar; impixelinfo;
title('Original Image');


line([max_x, max_x],[max_y, max_y-
radius],'Color','g','LineWidth',2)
hold on
plot(max_x,max_y,'r.','MarkerSize',10,'Marker','x');
hold off

%gradientImY
```

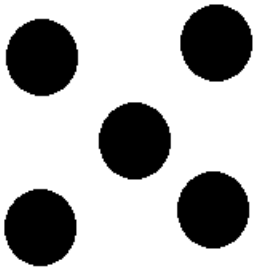
```
subplot(2,3,2);
imagesc(gradientImY);
colormap('gray'); colorbar; impixelinfo;
title('Y Gradient');

%gradientImX
subplot(2,3,3);
imagesc(gradientImX);
colormap('gray'); colorbar; impixelinfo;
title('X gradient');


%Hough Transform
subplot(2,3,4);
pcolor(H);
shading flat;
title('Hough Transform');
colormap('gray'); colorbar; impixelinfo;
set(gca,'YDir','reverse');

%Imaginary
subplot(2,3,5);
pcolor(expected_average_H_for_uniform_edge_distribution);
shading flat;
title('Expected average Hough Transform for uniform edge
distribution');
colormap('gray'); colorbar; impixelinfo;




% finds lb_i and ub_i in the sorted array, arr such that:
%   arr(lb_i) <= val <= arr(ub_i) and
(lb_i+1==ub_i)||(lb_i==ub_i)
% See asserts for pre-conditions
function [lb_i, ub_i] = find_lb_ub_in_sorted_arr(val, arr)

assert(isvector(arr));
assert(issorted(arr));
assert(val>=arr(1));
assert(val<=arr(end));

ub_i= find(arr>=val, 1);
lb_i= max([ 1, ub_i-1 ]);

assert((lb_i+1==ub_i)||(lb_i==ub_i))
assert( ((arr(lb_i)<=val)&&(arr(ub_i)>=val)) || (val>=arr(end))
);
```

```
function [H, betas, alphas,
expected_average_H_for_uniform_edge_distribution] = ...
    Hough_circles(gradientImY, gradientImX, alphaBetaSpacing,
radius, thetaSpacing)

assert(radius > 0);
assert(thetaSpacing>0);
assert(thetaSpacing<pi/2);

Y= size(gradientImY,1);
assert(Y==size(gradientImX,1));
X= size(gradientImY,2);
assert(X==size(gradientImX,2));


betas = unique([1:alphaBetaSpacing:Y Y]);
alphas = unique([1:alphaBetaSpacing:X X]);

H = zeros(length(betas),length(alphas));
expected_average_H_for_uniform_edge_distribution=H;

thetasArray= unique( [-pi :thetaSpacing:pi pi] );
thetasArray= thetasArray(1:end-1); % remove pi/2 to be in cyclic
mode

%Imaginary image
for i=1:Y
    for j=1:X
        for k= thetasArray
            %if (gradientImX(i,j) || gradientImY(i,j))
            y = i + sin(k)*radius;
            x = j + cos(k)*radius;

             if ( x <= X && y <= Y && x >= 1 && y >= 1)
               [y_lb_i,y_ub_i] =
find_lb_ub_in_sorted_arr(y,betas);
               [x_lb_i,x_ub_i] =
find_lb_ub_in_sorted_arr(x,alphas);

               y_l = betas(y_lb_i);
               y_u = betas(y_ub_i);
               x_l = alphas(x_lb_i);
               x_u = alphas(x_ub_i);

               %blinear

expected_average_H_for_uniform_edge_distribution(y_ub_i,x_lb_i)
=...
```

```
expected_average_H_for_uniform_edge_distribution(y_ub_i,x_lb_i)
+abs((x_u-x)*(y_l-y));


expected_average_H_for_uniform_edge_distribution(y_ub_i,x_ub_i)
=...

expected_average_H_for_uniform_edge_distribution(y_ub_i,x_ub_i) +
abs((x_l-x)*(y_l-y));


expected_average_H_for_uniform_edge_distribution(y_lb_i,x_lb_i)
=...

expected_average_H_for_uniform_edge_distribution(y_lb_i,x_lb_i) +
abs((x_u-x)*(y_u-y));


expected_average_H_for_uniform_edge_distribution(y_lb_i,x_ub_i)
=...

expected_average_H_for_uniform_edge_distribution(y_lb_i,x_ub_i) +
abs((x_l-x)*(y_u-y));
             end
           %end
        end
    end
end
%Real image
for i = 1:Y
    for j = 1:X
        if (gradientImX(i,j) || gradientImY(i,j))
            t = atan2(gradientImY(i,j), gradientImX(i,j));
            grad_magnitude =
norm([gradientImX(i,j),gradientImY(i,j)],2);
            y = i + sin(t)*radius;
            x = j + cos(t)*radius;
            if ( x <= X && y <= Y && x >= 1 && y >= 1)
                [y_lb_i,y_ub_i] =
find_lb_ub_in_sorted_arr(y,betas);
                [x_lb_i,x_ub_i] =
find_lb_ub_in_sorted_arr(x,alphas);

                y_l = betas(y_lb_i);
                y_u = betas(y_ub_i);
                x_l = alphas(x_lb_i);
                x_u = alphas(x_ub_i);

                %blinear
                H(y_ub_i,x_lb_i) = H(y_ub_i,x_lb_i) +
grad_magnitude*abs((x_u-x)*(y_l-y));
```

```
                H(y_ub_i,x_ub_i) = H(y_ub_i,x_ub_i) +
grad_magnitude*abs((x_l-x)*(y_l-y));
                H(y_lb_i,x_lb_i) = H(y_lb_i,x_lb_i) +
grad_magnitude*abs((x_u-x)*(y_u-y));
                H(y_lb_i,x_ub_i) = H(y_lb_i,x_ub_i) +
grad_magnitude*abs((x_l-x)*(y_u-y));
            end
        end
    end
        %y_lb_i
        %         a              b
        %    y         *
        %
        % y_ub_i    c             d
        %                   x
        %       x_lb_i          x_ub_i


end
```

input



Output

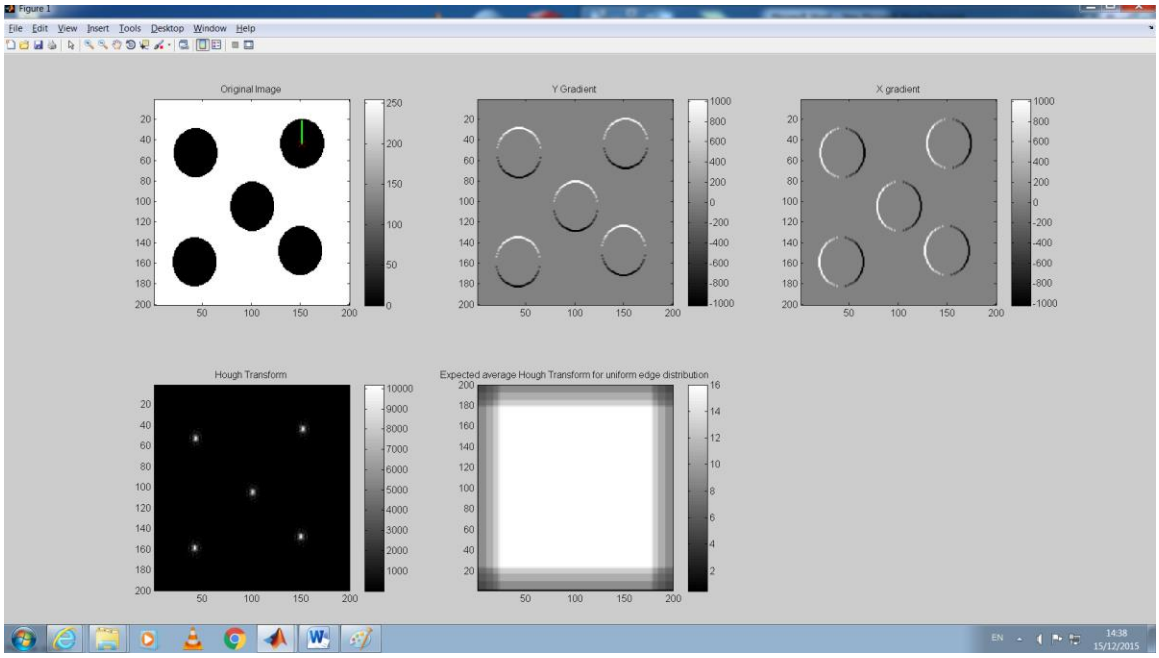


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image
Y Gradient
X gradient
Hough Transform
Expected average Hough Transform for uniform edge distribution
EN
14:38
15/12/2015