LINE AND CIRCLE DETECTION USING HOUGH TRANSFORM

NOVEMBER 16, 2021

TA's: Muhammed Zemzemoğlu and Mehmet Emin Mumcuoğlu

Name of Student: Moses Chuka Ebere
Student Number: 31491
Course: Computer Vision (EE 417)

TABLE OF CONTENTS

INTRODUCTION

HOUGH TRANSFORM

Let’s say that using an edge detector, we found the edges in an image. Now, we would like to group these edges to perform higher level analysis. How can this be done?

Canny’s edge detection algorithm partially does the above where strong edges reinforce weak ones to form a chain of edges; however, we need a more robust way. This is where Hough Transform comes in.

Hough Transform is a technique primarily used to detect straight lines and curves in images. It does this by mapping a tasking pattern problem into a case of relatively simple peak detection.

This lab focuses on applying Hough Transform for line and circle detection.

EXPLANATION OF METHODS

Line Detection Using Hough Transform

This is underpinned by the notion that using the parametric equation of a line:

$$\rho = x\cos\theta + y\sin\theta$$

A point in the image space translates to a sinusoid in the parameter/accumulator space.

ρ is the length of the normal from a line to the origin.
θ is the orientation of the line.
The Hough Transform for line detection assumes a parameter space where the rho and theta are fixed and a point in the image space represents a sinusoid in the parameter space.
The function looks at collinear points in the image space and finds the number of times their corresponding sinusoids in the parameter space intersect. With the help of a threshold, any cumulative intersection that exceeds the threshold qualifies to be a line.

Note: This function utilizes edge images.

In MATLAB, a function was written that accepts an image, converts it to grayscale, finds its edges and detects its lines using MATLAB's inbuilt hough function. The function is included in the appendix.

Also, the function is called in the main script (also in the appendix).

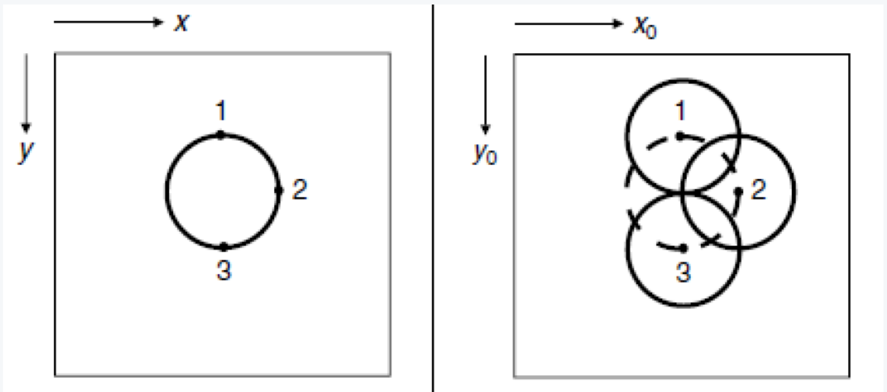
Circle Detection Using Hough Transform

This is underpinned by the notion that using the implicit form of the equation of a circle:

$$(x - x_0)^2 + (y - y_0)^2 = r^2$$

A circle centered at (x_0,y_0) in the image space translates to a new circle centered at (x,y) in the parameter/accumulator space.

r = radius of the circle.
Here, several points on the circle in the image space (x_i,y_i) are fixed and using the same radius or a different one, they are used to create new circles in the accumulator space. It is found that since the points belong to the same circle, they all intersect at the same point (x_0,y_0), which corresponds to the center of the original circle in the image space.



(a) Image containing a circle (b) Accumulator space

The function uses the following parametric equations to locate the position of the intersections. Plotting the accumulator space, we notice peaks at regions with myriads of intersections. These peaks are used to identify the circles in the original image.

Parametric Form: $x = x_0 + r\cos(\theta) \quad y = y_0 + r\sin(\theta)$

Parameters: $x_0 = x - r\cos(\theta) \quad y_0 = y - r\sin(\theta)$

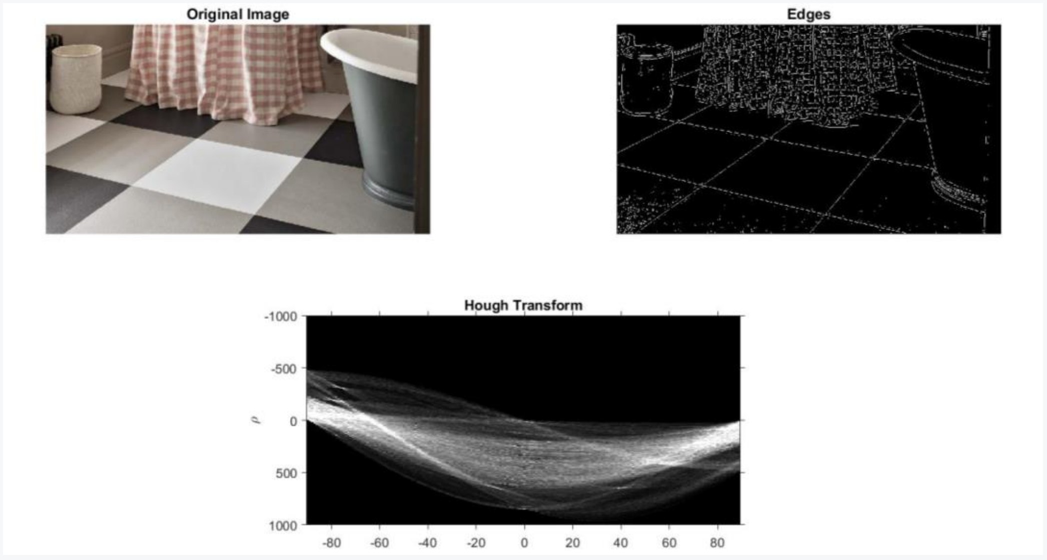
θ is the angle between the radius and the horizontal.

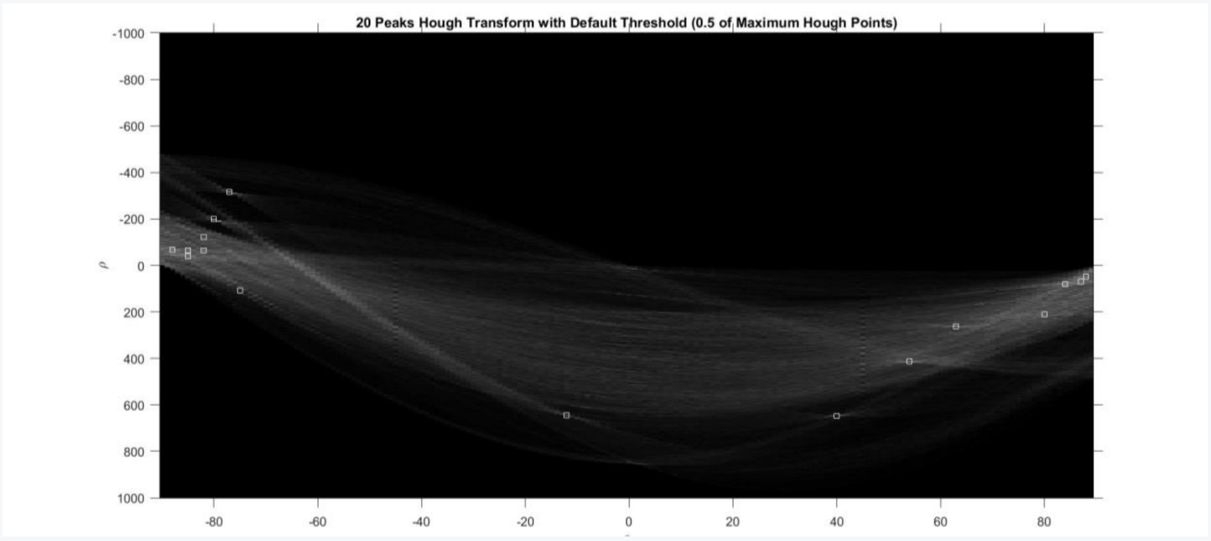
In MATLAB, a function was written that accepts an image and detects the circles (using different sensitivities) using MATLAB's inbuilt imfindcircles. The function is included in the appendix.

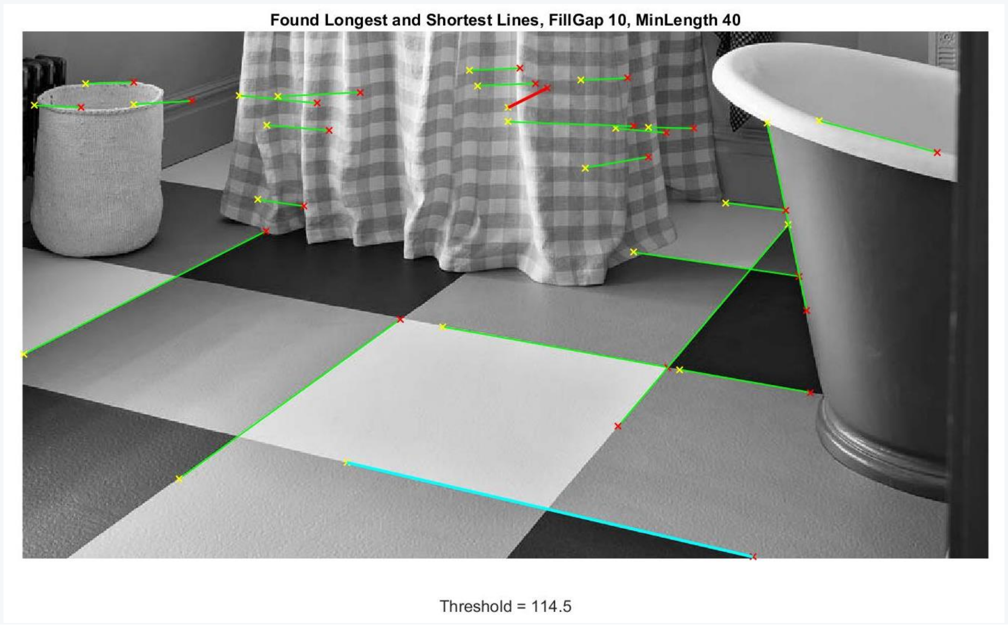
Also, the function is called in the main script (also in the appendix).

RESULTS

Line Detection

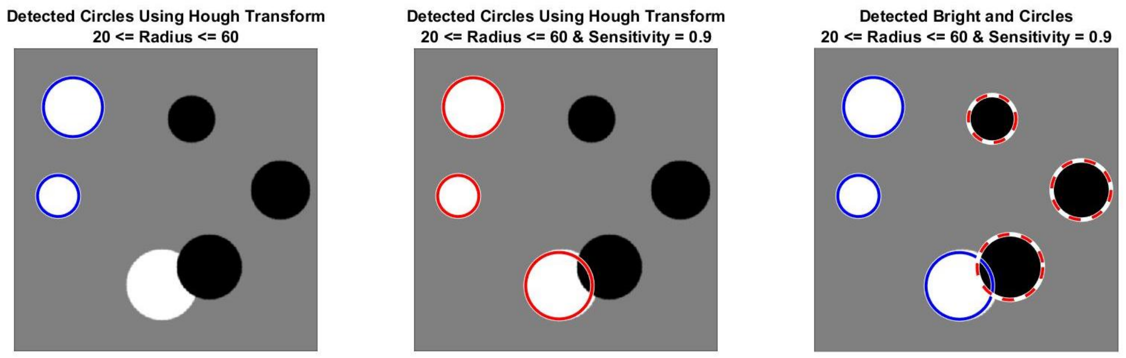






In the above, the edge detector used was zero crossing and the threshold was t = 0.2*max(H(:)). The function adequately detects the lines, and it is good at identifying the longest and shortest line. If we change the edge detector and/or the threshold, we'll definitely end up with varying results all round.

Circle Detection

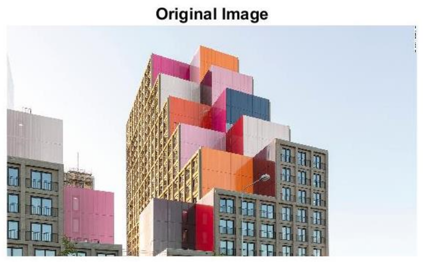


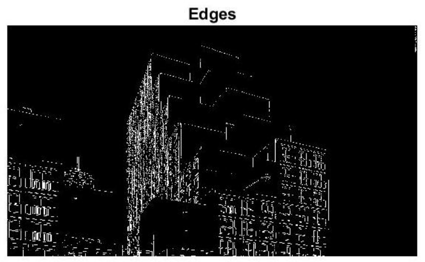
The above result shows that altering the sensitivity of our algorithm directly affects the results that will be obrained.
Notice: Hough Transform is so robust that it functions well even in the presence of occlusion. In the central image, the black circle is clearly partially obstructing the white circle; however, due to the high sensitivity, the circle is still detected.

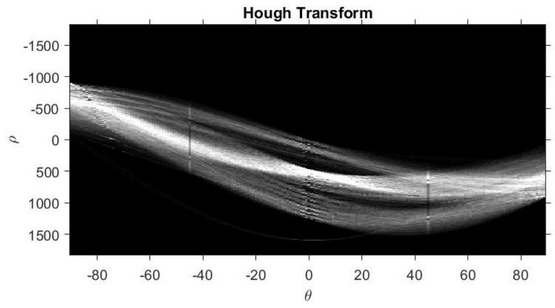
MULTIPLE RESULTS

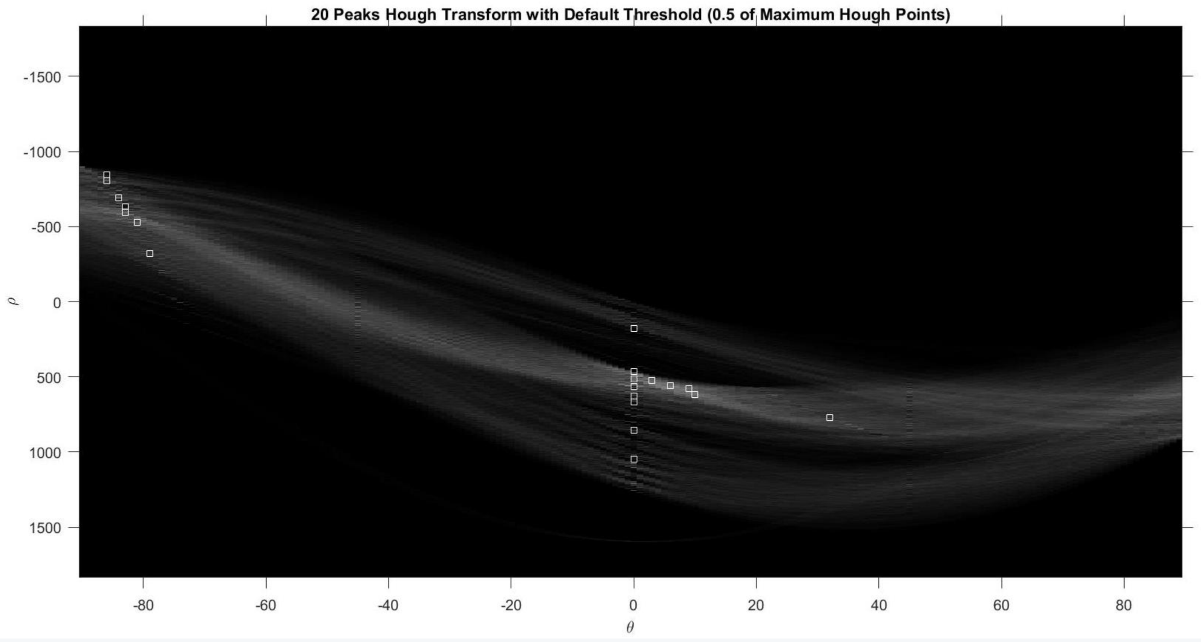
Line Detection

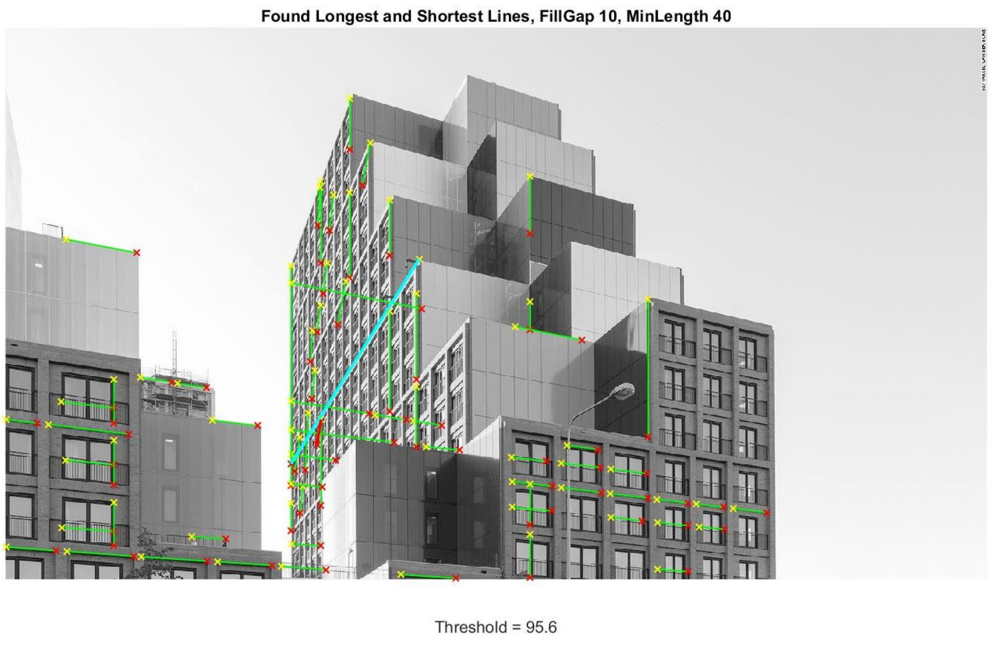
Let's apply the line detection algorithm on RGB images:



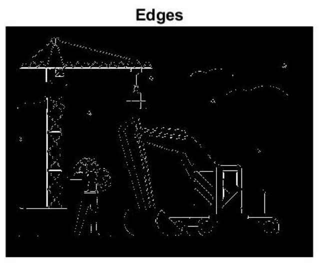


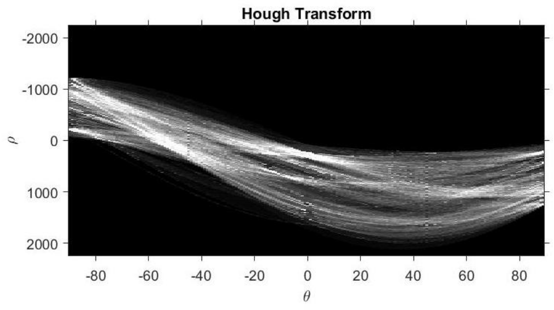


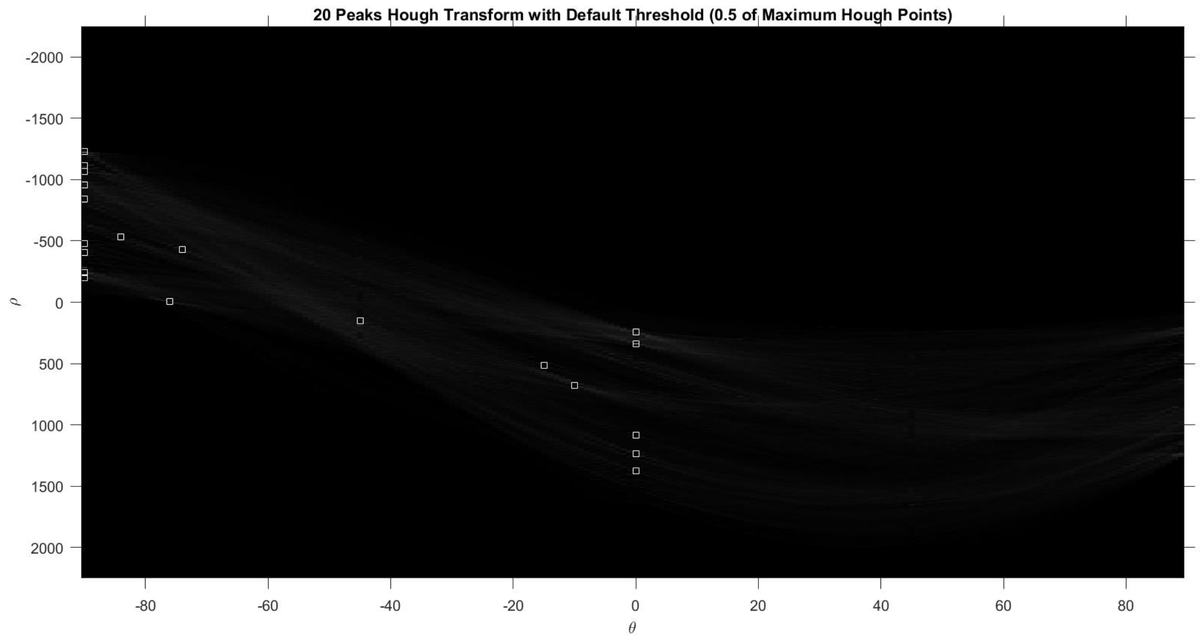


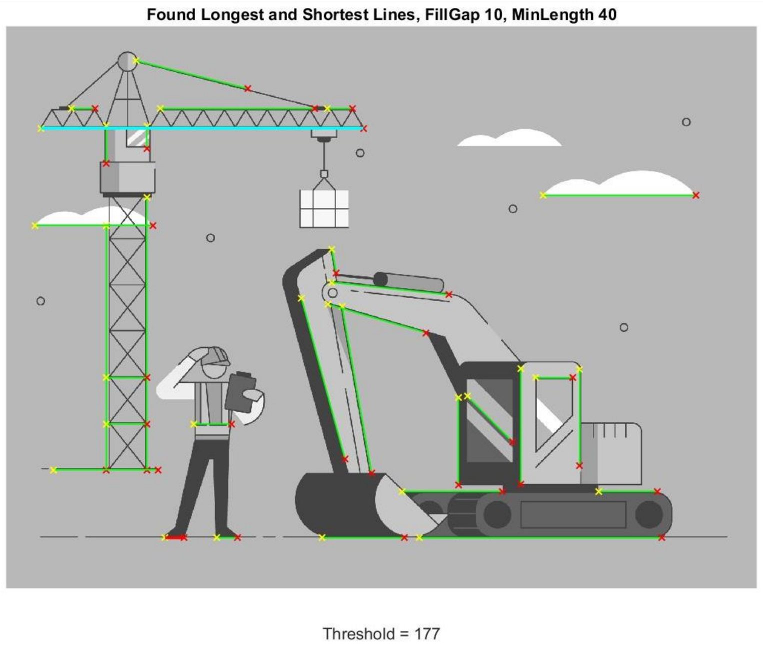


In the above, prewitt was used to detect the edges before the line detection steps. Also, t = 0.2*max(H(:)).







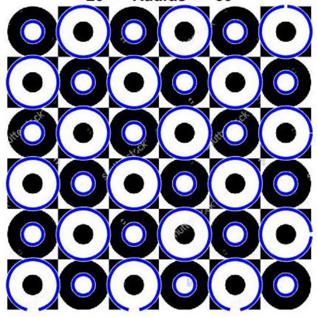


In the above, Laplacian of Gaussian edge detection was used first.

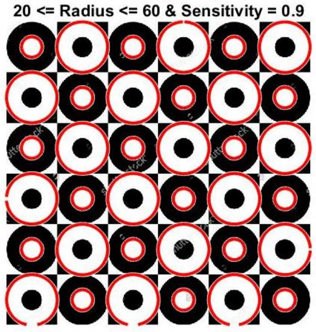
Circle Detection

Applying this on some color images:

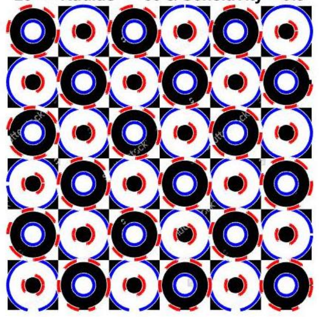
Detected Circles Using Hough Transform
20 <= Radius <= 60



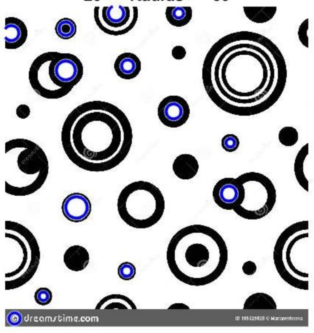
Detected Circles Using Hough Transform
20 <= Radius <= 60 & Sensitivity = 0.9



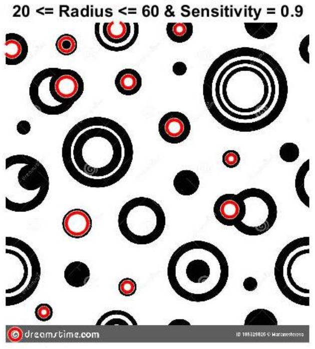
Detected Bright and Circles
20 <= Radius <= 60 & Sensitivity = 0.9



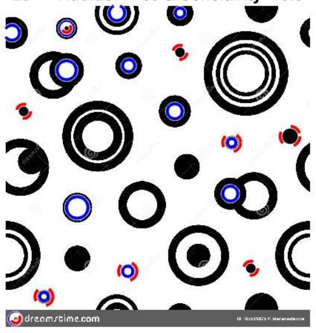
Detected Circles Using Hough Transform
20 <= Radius <= 60



Detected Circles Using Hough Transform
20 <= Radius <= 60 & Sensitivity = 0.9



Detected Bright and Circles
20 <= Radius <= 60 & Sensitivity = 0.9



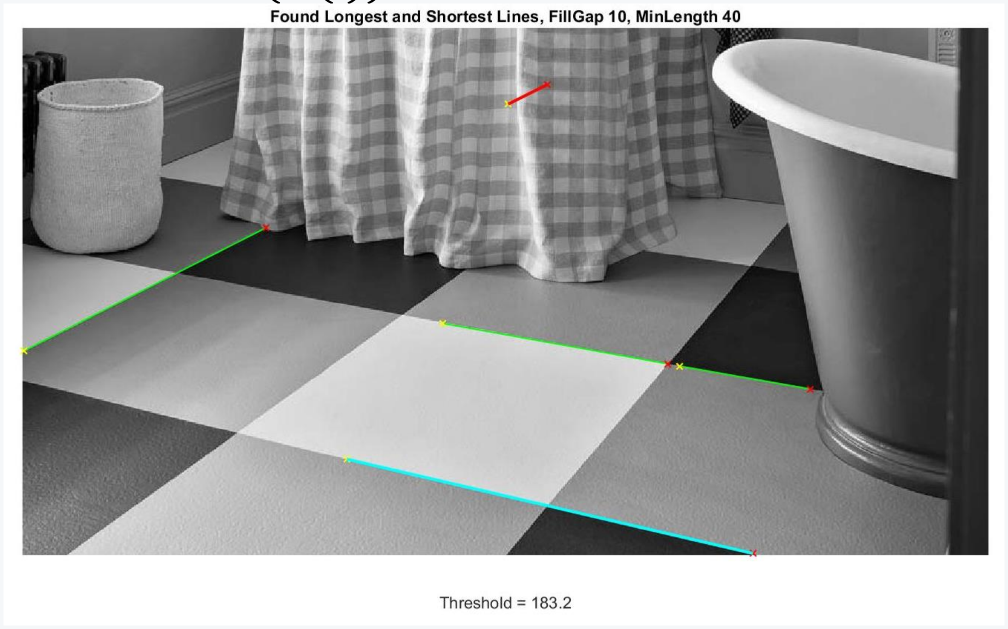
POST-LAB QUESTIONS

Effect of Changing Threshold and/or Sensitivity on Line and Circle Detection

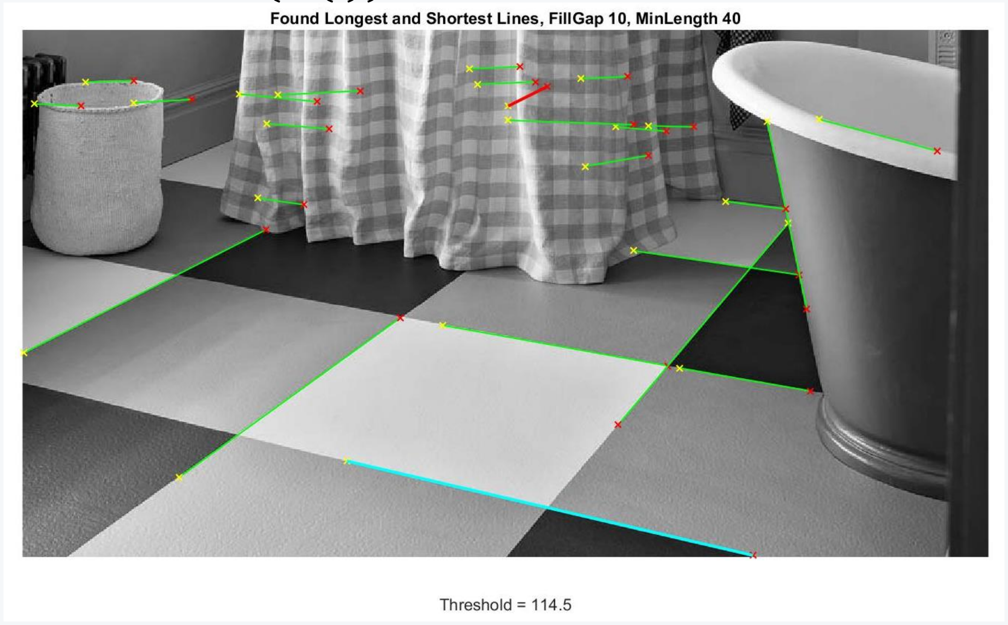
Line Detection

Using different thresholds, t, on the result, H, of a Standard Hough Transform:

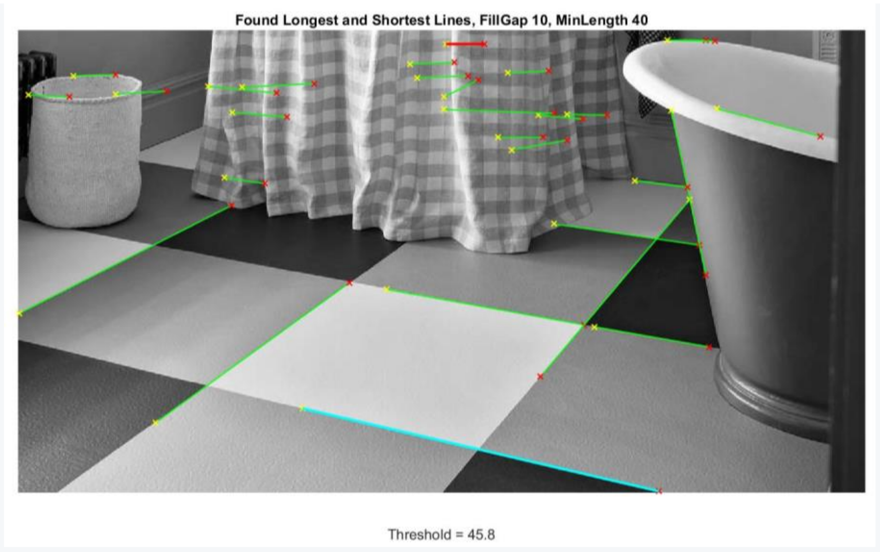
1. When t = 0.8*max(H(:))



2. When t = 0.5*max(H(:))



3. When t = 0.2*max(H(:))



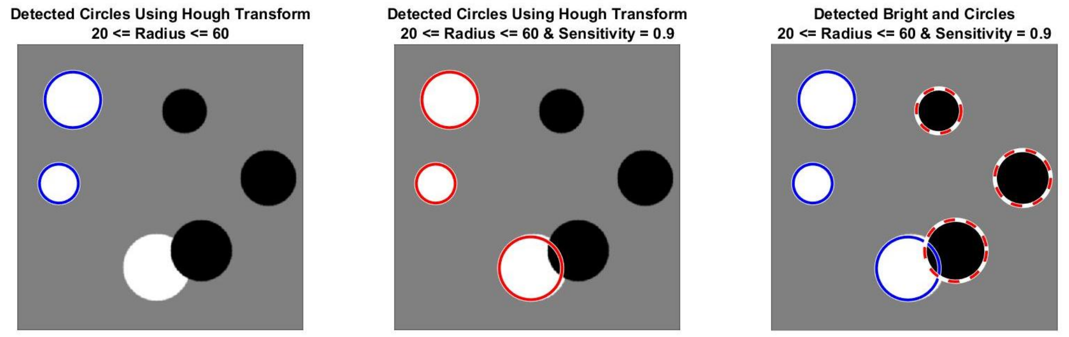
In the parameter space, the threshold defines the minimum number of sinusoid intersections to be considered as a peak.
We observe that the higher the threshold, the lower the number of possible peaks we end up with. This is evident in the above three results.

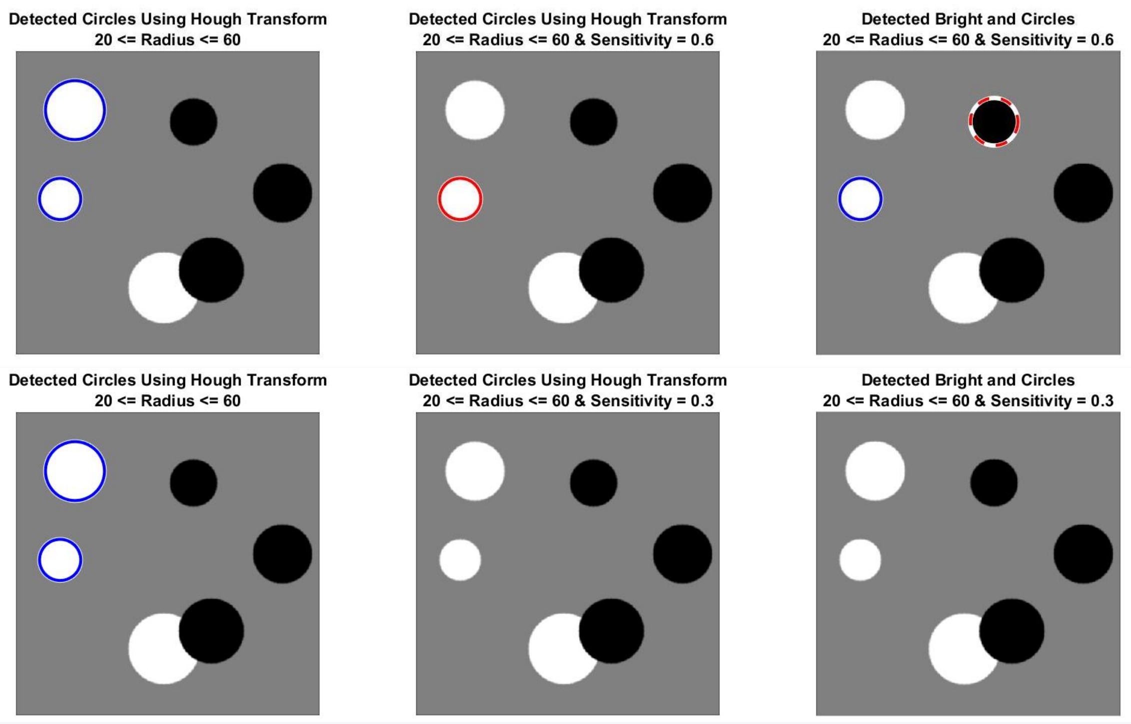
- At t = 183.2, only five lines had intersections above this.
- As t was lowered, more lines were identified.

I believe that the higher the threshold, the higher the accuracy of the line detection algorithm. This is predicated on the fact that for a line to be above a high threshold, we must have found so many collinear points in the image. Having so many collinear points convinces us that what we have is actually a line.

Circle Detection

Circle detection was performed using different sensitivities, s = 0.9, 0.6, and 0.3.


Detected Circles Using Hough Transform
20 <= Radius <= 60
Detected Circles Using Hough Transform
20 <= Radius <= 60 & Sensitivity = 0.9
Detected Bright and Circles
20 <= Radius <= 60 & Sensitivity = 0.9

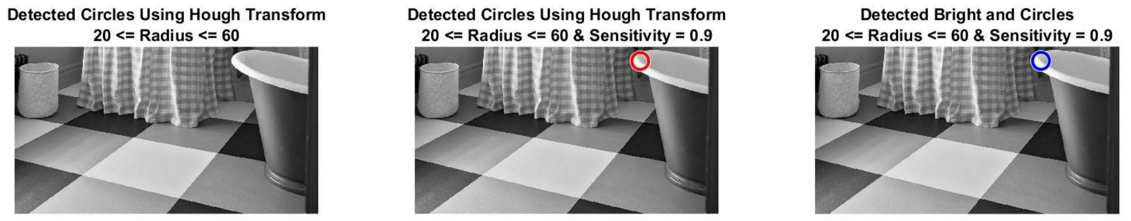


As we can observe in the above results, the lower the sensitivity, the lower the possibility of detecting circles. When the sensitivity was 0.9, three white circles were detected (central result) as opposed to two white circle when the sensitivity was the default value (0.85). At s = 0.6, only one white circle (central image) was detected. At s = 0.3, no white circle (central image) was detected.

Summary:

The higher the sensitivity factor, the more the circular objects detected. Note that this will include weak and partially obscured circles.

Caveat: Higher sensitivity values render the algorithm prone to false detection. This can be observed below:


Detected Circles Using Hough Transform
20 <= Radius <= 60
Detected Circles Using Hough Transform
20 <= Radius <= 60 & Sensitivity = 0.9
Detected Bright and Circles
20 <= Radius <= 60 & Sensitivity = 0.9

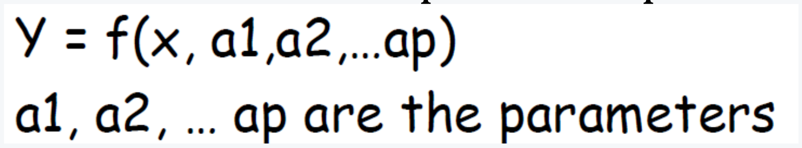
Ordinarily, there are no real circles in this image, so at s = 0.85 (default sensitivity), nothing is detected. However, at s = 0.9, the risk of false detection is clearly seen as the code falsely identifies a segment of the bathtub as a circle.

- **High and low sensitivities both affect accuracy such that high sensitivity can return false results and low sensitivity might not be able to identify viable results.**

DISCUSSION

General Comments on Hough Transform

- The main idea is that we map a difficult pattern problem into a simple peak detection problem.
- Hough Transform is a really robust algorithm as it could be used to detect any curve that can be expressed in parametric form.



$Y = f(x, a1, a2, \ldots ap)$

a1, a2, … ap are the parameters

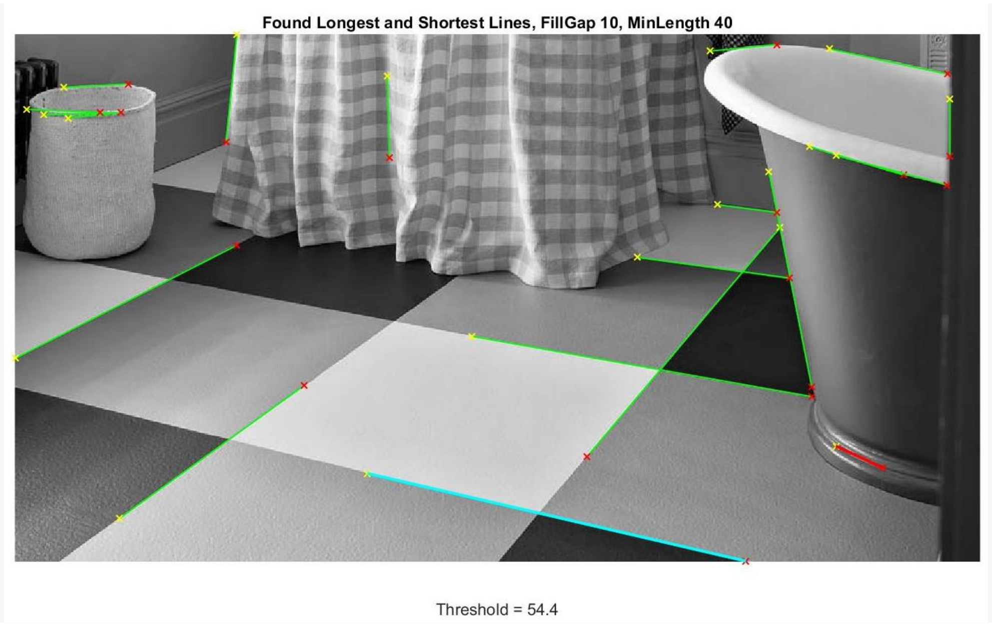
<u>**Disadvantage**</u>

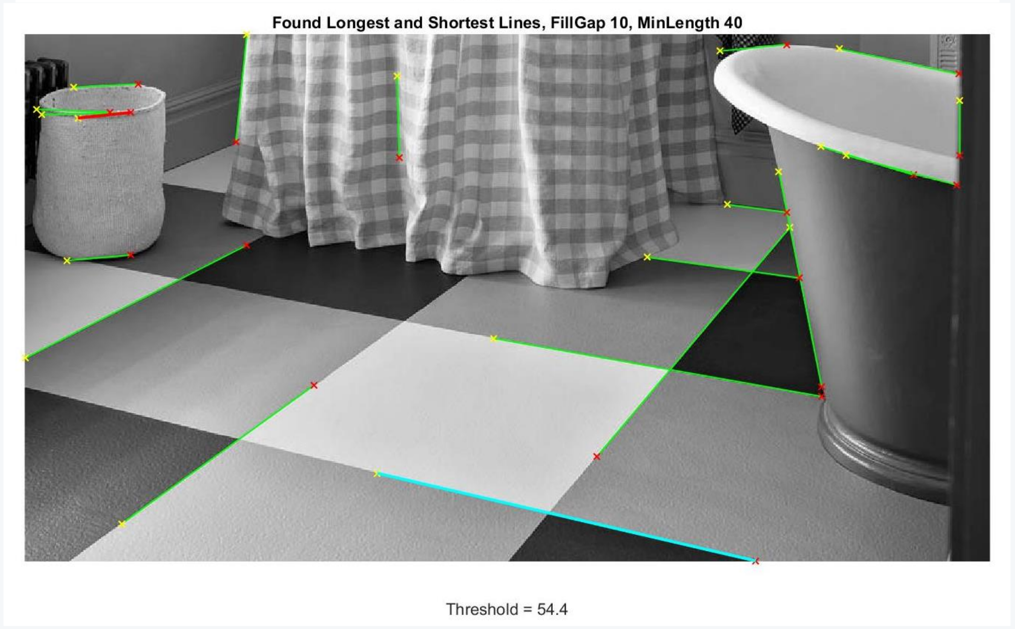
If p is large, the parameter space will be high-dimensional thereby driving up the computational cost. Therefore, we generally avoid cases of large p.

- **Hough transform is robust to undesirable features like noise, occlusion, etc.**

Line Detection

- We refrain from using the slope-intercept equation of a line because it leaves us with a parameter space that is infinite when we have slope of negative and positive infinity.
- In the above case, it is impossible to discretize an infinite dimensional space.
- **The line detection is partially dependent on the edge algorithm used to obtain the input edge image.**





In the first result, Sobel edge detection was applied; in the second result, prewitt was applied. Notice that the lines detected are quite different in some regions.

Circle Detection

- This is so robust that it could allow us to detect circles with a range of radii.
- We can speed up the process if we know the orientation of the edge, E_theta, from the edge detection process. This allows to only increment one counter.

REFERENCES

Reinhard Klette. 2014. Concise Computer Vision: An Introduction into Theory and Algorithms. Springer Publishing Company, Incorporated.

https://www.bigrentz.com/blog/how-to-start-construction-company

https://edition.cnn.com/style/article/anticipated-architecture-2021/index.html

https://www.designtrends.com/graphic-web/patterns/circle-pattern.html

https://www.dreamstime.com/seamless-abstract-pattern-black-overlapping-circles-dots-bubbles-bulbs-background-vector-illustration-calm-classic-design-image185529825

APPENDIX

Line Detection

The function for the Line Detection using Hough Transform in MATLAB is as follows:

```
function lab5houghlines (im)
tic
% The row, column, and channels of the image are obtained
along with the cardinality of the image.
[r, c, ch] = size(im);
Card = r*c;

img = im;

% This is added in case the image introduced is an RGB
image.
% It functions to convert it to a gray-scale image.
if ch == 3
    img = rgb2gray(img);
end

% Create a Black-white Edge Image
I = edge(img, 'zerocross');

% Find the space matrix, theta, and rho parameters.
[H,T,R] = hough(I);

% Threshold
t = 0.5*max(H(:));

% Find the peaks
P = houghpeaks(H, 20, 'threshold', t);

% Plot the results
figure %1
subplot(2,2,1);
imshow(im);
title('Original Image');

subplot(2,2,2)
imshow(I)
title('Edges')
```

```
subplot(2,2,3.5);
imshow(imadjust(rescale(H)),'XData',T,'YData',R,'InitialMag
nification','fit');
title('Hough Transform');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;

figure %2
imshow(H,[],'XData',T,'YData',R,'InitialMagnification','fit
');
xlabel('\theta'), ylabel('\rho');
axis on, axis normal, hold on;
plot(T(P(:,2)),R(P(:,1)),'s','color','white');
title('20 Peaks Hough Transform with Default Threshold (0.5
of Maximum Hough Points)')

% Extract the line segments based on Hough Transform
lines = houghlines(I,T,R,P,'FillGap',10,'MinLength',40);

% Plot the lines extracted and highlight the longest and
shortest lines.
figure %3
imshow(img), hold on
max_len = 0;
min_len = 0;
for k = 1:length(lines)
    % Obtain the endpoints of each line found earlier.
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',1,'Color','green');

   % Plot beginnings and ends of lines

plot(xy(1,1),xy(1,2),'x','LineWidth',1,'Color','yellow');
   plot(xy(2,1),xy(2,2),'x','LineWidth',1,'Color','red');

   % Determine the endpoints of the longest line segment
   len = norm(lines(k).point1 - lines(k).point2);
   if (len > max_len)
      max_len = len;
      xy_long = xy;
   end
   if(k==1)
       min_len = max_len;
   end
   % Use this to find the shortest line.
   if (len <= min_len)
      min_len = len;
      xy_short = xy;
   end
end

% Highlight the longest and shortest lines.
plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','cyan'
);
plot(xy_short(:,1),xy_short(:,2),'LineWidth',2,'Color','red
');
title('Found Longest and Shortest Lines, FillGap 10,
MinLength 40')

toc
end
```

In the main script, the above function is called as follows:

```
%% Line Detection
% Read the image to be preprocessed
a = imread('checker.jpg');

% Call the Hough Transform Function for Lines.
lab5houghlines (a);
```

Circle Detection

The function for the Circle Detection using Hough Transform in MATLAB is as follows:

```
function lab5houghcircles (im)
tic

% The row, column, and channels of the image are obtained
along with the cardinality of the image.
```

```
[r, c, ch] = size(im);
Card = r*c;

% This is added in case the image introduced is an RGB
image.
% It functions to convert it to a gray-scale image.
if ch == 3
    im = rgb2gray(im);
end

% Set the range of circle radii to be detected.
Rmin = 20;
Rmax = 60;

% Find all the bright circles in the image within 20 <=
raduis <= 60 pixels
[centersBright, radiiBright] = imfindcircles(im,[Rmin
Rmax],'ObjectPolarity','bright');

% Sensitivity
s = 0.9;

% Plot the results
figure
subplot(1,3,1)
imshow(im)
hold
viscircles(centersBright, radiiBright,'Color','b');
title({('Detected Circles Using Hough Transform'),
[num2str(Rmin),...
    ' <= Radius <= ', num2str(Rmax)]})

subplot(1,3,2)
imshow(im)
hold
[centersBright1, radiiBright1]...
    = imfindcircles(im,[Rmin
Rmax],'ObjectPolarity','bright', 'Sensitivity', s);
viscircles(centersBright1, radiiBright1,'Color','r');
title({('Detected Circles Using Hough Transform'),
[num2str(Rmin),...
```

```
    ' <= Radius <= ', num2str(Rmax), ' & Sensitivity = ',
num2str(s)]})

subplot(1,3,3)
imshow(im)
hold
[centersBright2, radiiBright2]...
    = imfindcircles(im,[Rmin
Rmax],'ObjectPolarity','bright', 'Sensitivity', s);

% Find all the dark circles in the image within 20 <=
raduis <= 60 pixels
[centersDark2, radiiDark2]...
    = imfindcircles(im,[Rmin Rmax],'ObjectPolarity','dark',
'Sensitivity', s);
viscircles(centersBright2, radiiBright2,'Color','b');
viscircles(centersDark2, radiiDark2,'LineStyle','--');
title({('Detected Bright and Circles'), [num2str(Rmin),...
    ' <= Radius <= ', num2str(Rmax), ' & Sensitivity = ',
num2str(s)]})

toc
end
```

In the main script, the above function is called as follows:

```
%% Circle Detection
% Read the image to be preprocessed
b = imread('circlesBrightDark.png');

% Call the Hough Transform Function for Circles.
lab5houghcircles (b);
```