EE417 - Assignment 4 - Post-Lab Report

Baris Sevilmis
Lab Date: 06/11/2018
Due Date: 12/11/2018

Overview

Fourth lab assignment of EE417 focuses on Hough Transform Techniques, namely Hough Line Detection and Hough Circle Detection. Details of these techniques will be provided in further sections of the report as well as their results on multiple images.

Part 1: Hough Line Detection

Hough line detection is an unique algorithm in order to detect straight lines and also curves in given image. Two important concepts of this algorithm are the image space and parameters space. Transforming between these spaces provides us features such as detection of collinear edges in an image. Given formulas below ensure the formulas used for transferring between these two spaces:

Image space line equation: $y = mx + n$
Parameter space line equation: $n = (-x)m + y$

These equations are actually same, however second equation provides us to depict lines within the parameter space. As known already, lines in image space are points in parameter space, points in image space are lines in parameter space. Therefore, collinear points in image space are intersecting lines in parameter space. Hough transform technique starts giving algorithm a set of collinear edge points from image space meaning that intersecting lines in parameter space. Given that these lines intersect at point (m,n) in parameter space, this point intersection point refers to a line in image space. Therefore, by giving some collinear points(certain edge points) in image space, and by intersection of corresponding lines in parameter space, at the end we get corresponding lines in image space. For each point in parameter space, array of counters are used in terms of counting lines that are passing through this point. As a result, more lines passing indicate more edges are collinear in image space. Finding a peak point through a threshold in these counter arrays ensures bright points in parameter space. In terms of solving infinite parameter space issue and representation of vertical lines, normal equation of line turns into following sinusoid form in which ρ is distance from origin and $\theta \in [0, \pi]$:

$$\rho = xcos(\theta) + ysin(\theta)$$

With help of these formula straight lines can be detected in image with no issues remaining. In Listing 1 and 2, Matlab Code for Hough Line Detection are provided with corresponding plot functions.

Listing 1: MatLab Code for Hough Line Detection- Part 1

```
function [J, theta, rho] = lab4houghlines(img)
    [row, col, ch] = size(img);
    if(ch == 3)
        rgbimg = rgb2gray(img);
    end
    %Canny edge detection
    J = edge(rgbimg,'Canny');

    tic
    %Hough Transform
    [H,theta,rho] = hough(J,'RhoResolution',0.5,'Theta'
        ,-90:0.5:89);
    toc

    tic
    %Hough Peaks
    J_peaks = houghpeaks(H,20);
    toc
    figure(1);
    subplot(3,2,1);imshow(img);title('Original Image');
    subplot(3,2,2);imshow(J);title('Canny Detected Edges');

    subplot(3,2,[3 4]);imshow(imadjust(rescale(H)),'XData',theta
        ,'YData',rho, 'InitialMagnification','fit');
    title('Hough Transform');
    xlabel('\theta'), ylabel('\rho');
    axis on, axis normal, hold on;

    subplot(3,2,[5 6]);imshow(H,[],'XData',theta,'YData',rho,'
        InitialMagnification','fit');
    title('20 Peaks Hough Transform with Default (0.5 of maximum
        Hough Points Threshold)');
    xlabel('\theta'), ylabel('\rho');
    axis on, axis normal, hold on;
    plot(theta(J_peaks(:,2)),rho(J_peaks(:,1)),'s','color','
        white');
```

Listing 2: MatLab Code for Hough Line Detection- Part 2

```
    tic
    %Hough Lines
    lines = houghlines(J,theta,rho,J_peaks,'FillGap',10, '
       MinLength',40);
    figure(2); imshow(rgbimg); hold on;
    toc

    %Hough Lines Plot
    max_len = 0;
    min_len = 200;
    for k = 1:length(lines)
        xy = [lines(k).point1; lines(k).point2];
        plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

        % Plot beginnings and ends of lines
        plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow')
           ;
        plot(xy(2,1),xy(2,2),'x','LineWidth',2,'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 ( len < min_len)
            min_len = len;
            xy_short = xy;
        end
    end
    plot(xy_long(:,1),xy_long(:,2),'LineWidth',2,'Color','cyan')
       ;
    plot(xy_short(:,1),xy_short(:,2),'LineWidth',2,'Color','red'
       );

end
```

Before discussing results, a more compact description of Hough Line Detection Algorithm is depicted in Figure 1.

Input is an edge image (E(i,j)=1 for edgels)

1. Discretize θ and ρ in increments of dθ and dρ.
   Let A(R,T) be an array of integer accumulators,
   initialized to 0.
2. For each pixel E(i,j)=1 and h=1,2,...T do
   1. ρ = i cos(h * dθ) + j sin(h * dθ)
   2. Find closest integer k corresponding to ρ
   3. Increment counter A(k,h) by one
3. Find local maxima in A(R,T)

Figure 1: Hough Transform Technique

Figure 2 depicts results of Hough Line detection on checker.png including every necessary step. Figure 3 shows the line detections on the image space with longest and shortest lines. On the other hand, Figure 4 shows the running time of following steps:

- Counting lines in parameter space
- Picking peak points
- Line Detection

Threshold value that is used in algorithm by default is computed by $0.5 * max(H(:))$. This threshold value corresponds to 128 in given results. In addition, Canny edge detector was used for preprocessing before Hough Transform as it performs quite well in terms of edge detection.

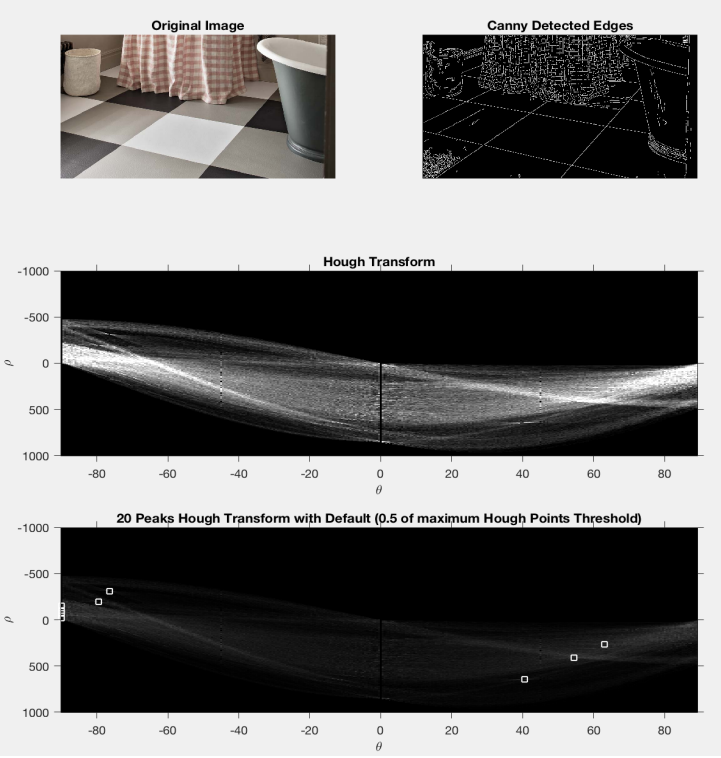

Figure 2: Parameter Space results with Threshold = 128

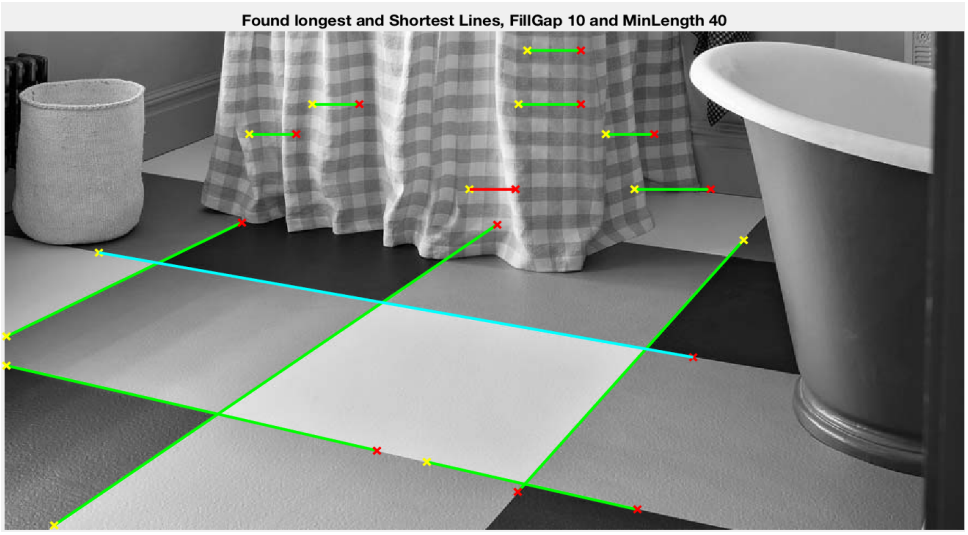

Figure 3: Image Space Result with Threshold = 128

```
Elapsed time is 0.041657 seconds.
Elapsed time is 0.040895 seconds.
Elapsed time is 0.237807 seconds.
```

Figure 4: Running Time of Hough functions with Threshold = 128

Figure 2,3 and 4 are the exact same results as in the slides.

- Maximum distance value is 545 in x-axis direction
- Maximum distance value is 101 in y-axis direction
- Minimum distance value is 70 in x-axis direction
- Minimum distance value is 0 in x-axis direction

In order to understand functioning of algorithm better, it would be reasonable to see results with different threshold values. Therefore, a set of results with lower threshold value and higher threshold value do need to be tested as well. Figure 5 and Figure 6 depicts same image with threshold value of 50. Figure 7 depicts the running time for same set of functions such as in Figure 4. Figure 8 and 9 demonstrates Hough Line Detection with Threshold = 200. Lastly, Figure 10 depicts running time of corresponding functions for Threshold = 200. It is clear that, more the threshold increases, less the lines are detected. Only highest peak values can pass the threshold such that the result can be confirmed. Same happens when peak point amount is changed such as decreasing 20 peak points to 10 peak points will result same as increasing threshold. As the results of this operation is obvious, Figure 8 and 9 can be checked to see similar results of peak amount decrease. To conclude, there are 2 parameters that are used within the houghlines function as well, however they are not responsible of algorithms performance. These parameters are FillGap and MinLength, that are responsible of visualization of lines such that they decide whether to merge lines given a length and minimum length of lines to be displayed on the original image. To show their effects, their values will be also altered by stabilizing Threshold to 128.

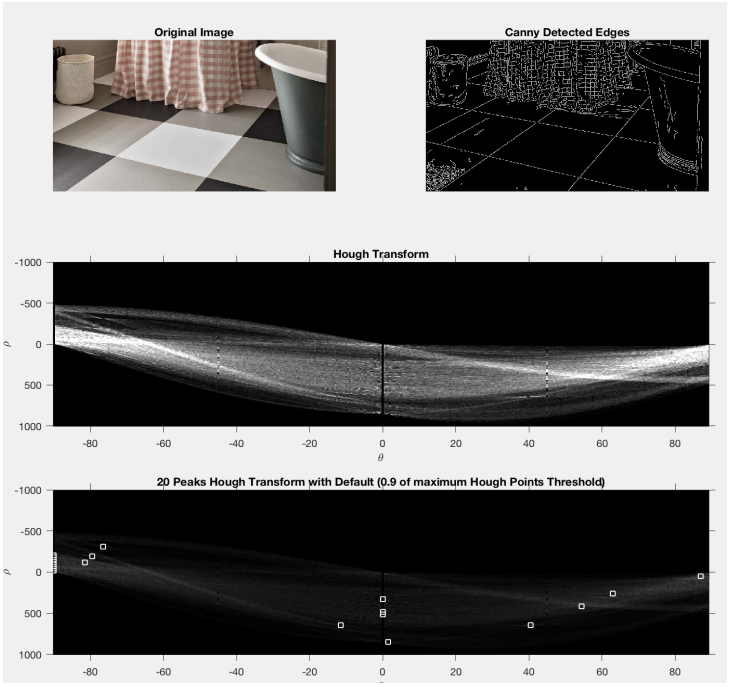

Figure 5: Parameter Space results with Threshold = 50

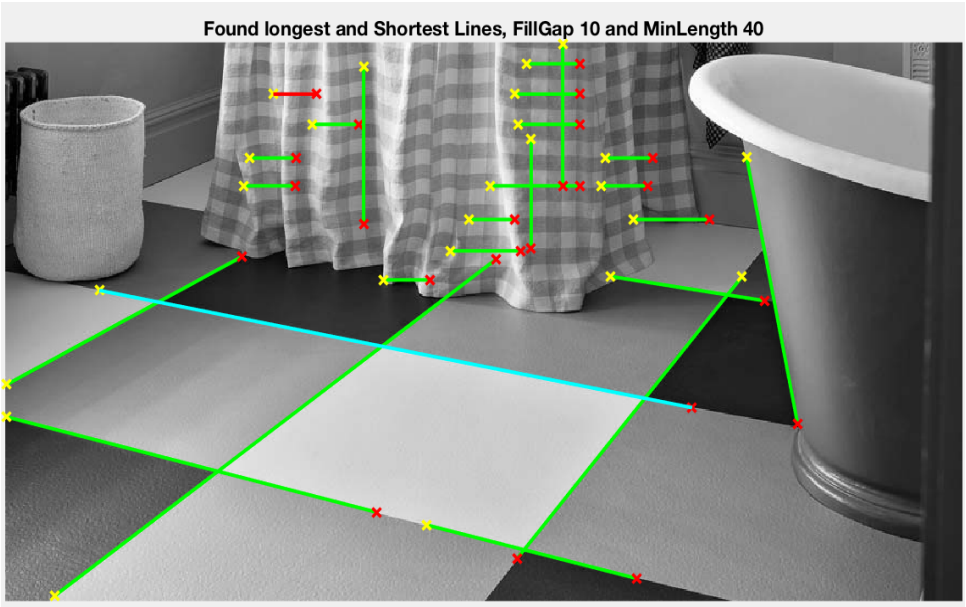

Figure 6: Image Space Result with Threshold = 50

```
Elapsed time is 0.039768 seconds.
Elapsed time is 0.025632 seconds.
Elapsed time is 0.535292 seconds.
```

Figure 7: Running Time of Hough functions with Threshold = 50

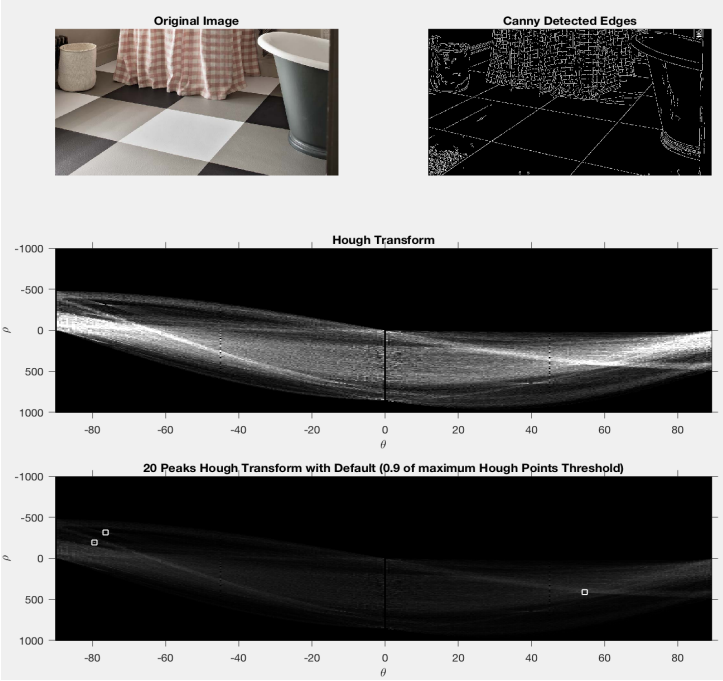

Figure 8: Parameter Space results with Threshold = 200

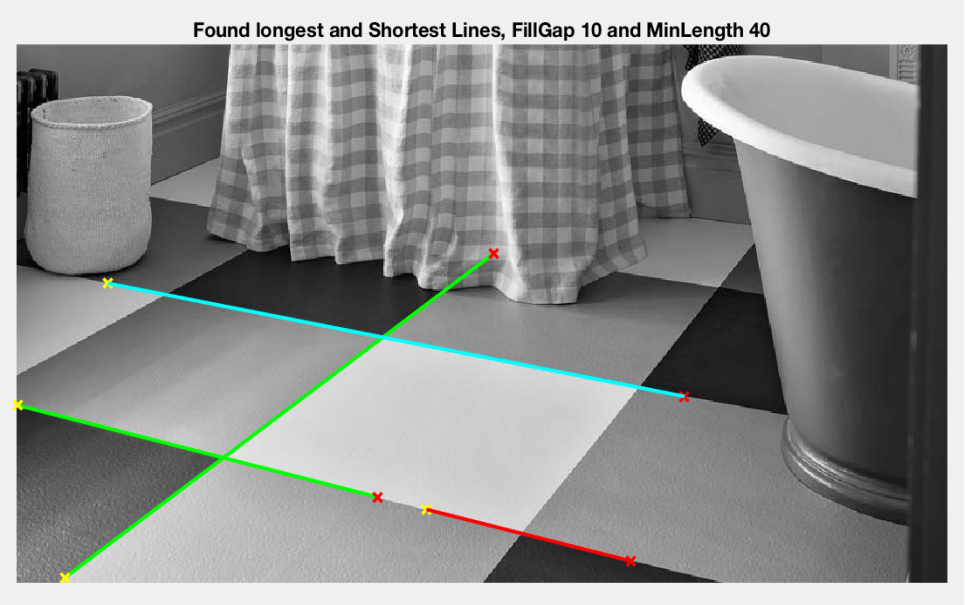

Figure 9: Image Space Result with Threshold = 200

```
Elapsed time is 0.040060 seconds.
Elapsed time is 0.017270 seconds.
Elapsed time is 0.305433 seconds.
```

Figure 10: Running Time of Hough functions with Threshold = 200

Figure 11 depicts same results with Figure 3, only difference is FillGap value is increased to 30. Threshold is 128 and MinLength is 40 in order to see full effects of FillGap, and it is clear that increasing FillGap resulted into merging some of the lines as the gap size for merging lines are increased. On the other hand, Figure 12 depicts same results with Figure 3, however MinLenght is increased to 70. FillGap is 10 and Threshold is 128. As a result, it can be shown that as MinLength increase, minimum length to display values has also increased such that lines with length less than 40 are not shown.

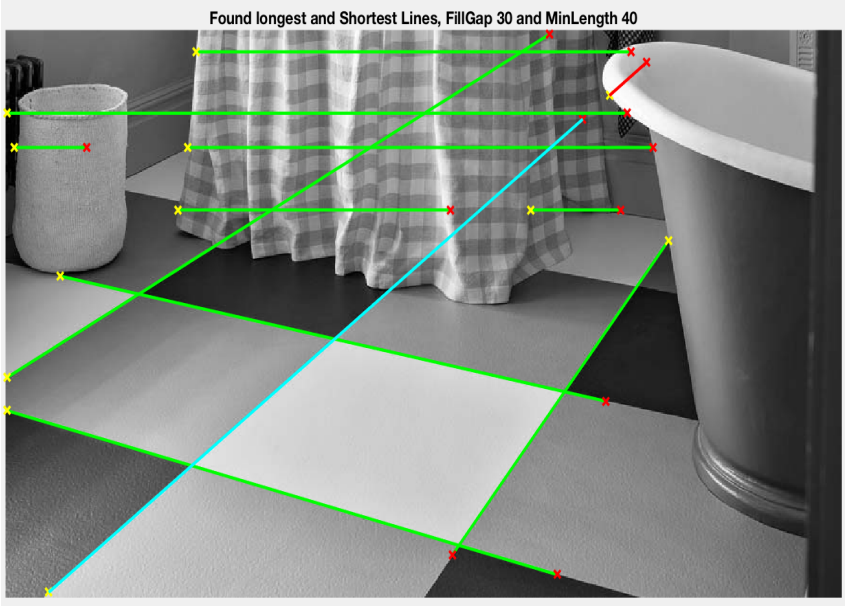

Figure 11: Image Space Result with Threshold = 128(FillGap increased from 10 to 30)

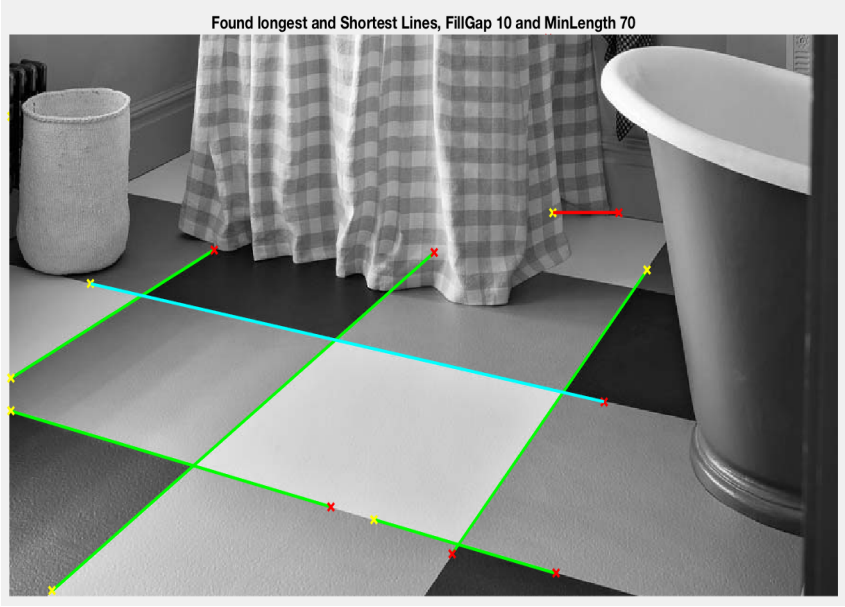

Figure 12: Image Space Result with Threshold = 128(MinLength increased from 40 to 70)

Part 2: Hough Circle Detection

Hough Circle Detection follows exact same algorithm as in hough Line Detection. For this reason, they are be considered as variations of same Hough Transform Techniques. Only difference of them resides in presentation of their natural shapes such that line equation used was the following formula:

$$\rho = xcos(\theta) + ysin(\theta)$$

To detect circles following Hough Circle Detection, used formula changes format into circle detection format that is the following:

Implicit Form: $(x - (x_0)^2 + (y - (y_0)^2) = r^2)$
Parametric Form: $x = x_0 + rcos(\theta)$ and $y = y_0 - rsin(\theta)$
Parameters: $x_0 = x - rcos(\theta)$ and $y_0 = y + rsin(\theta)$

As it can be seen, these formulas won't face problems such as infinite parameter space as all parameters are finite as in Sinusoid Line Detection formula. In addition, because of detecting a different feature(shape) functions that are used within MatLab are changed as well, because of the changes such as parameter formula change. In Listing 3, MatLab code for Hough Circles Detection is demonstrated as well as the corresponding plot functions.

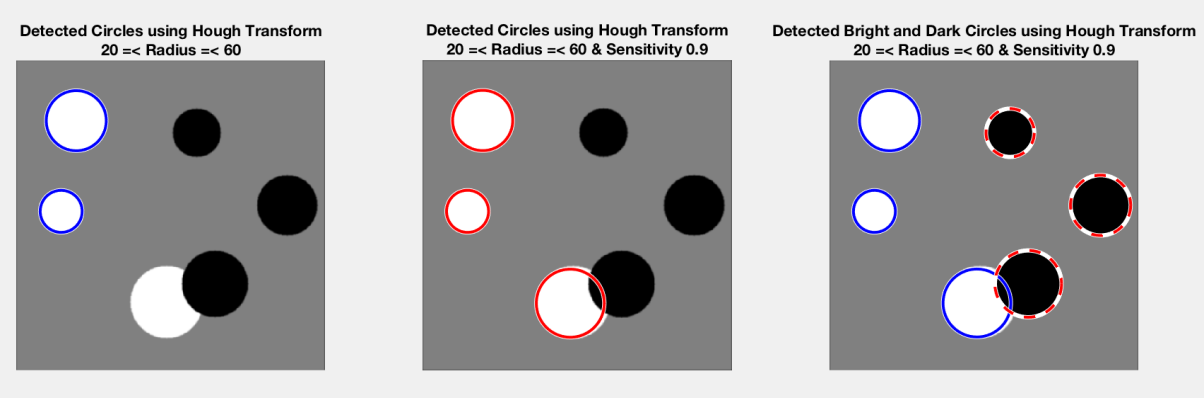

Figure 13: Hough Circles Detected with Sensitivity = 0.9)

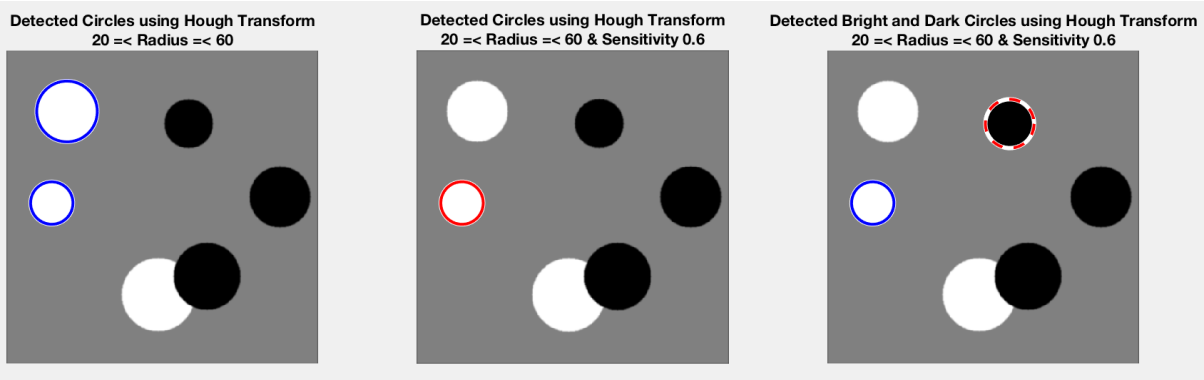

Figure 14: Hough Circles Detected with Sensitivity = 0.6)

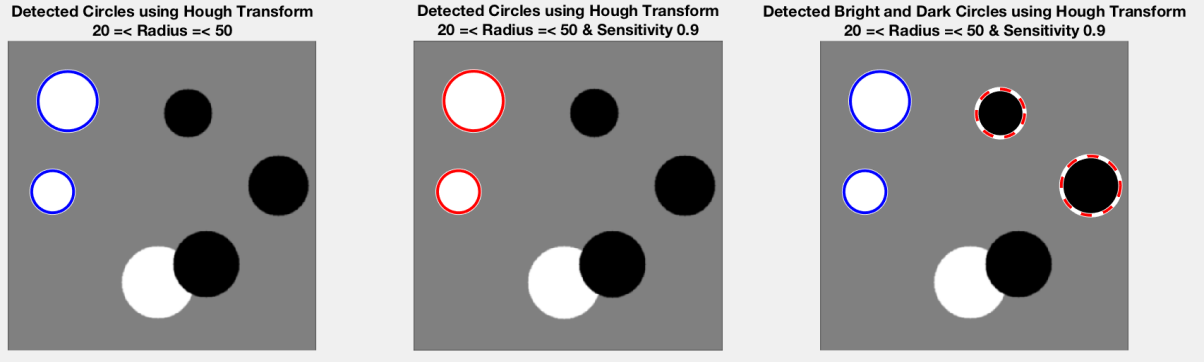

Figure 15: Hough Circles Detected with Sensitivity = 0.9 (Radius from [20 60] to [20 50]))

Discussing the results is crucial as always. Figure 13 depicts original results with $radius \in [20, 60]$ and Sensitivity = 0.9. As it can be seen all circles are detected as soon as sensitivity is raised to 0.9 in second and third sub image of Figure 13. ObjectPolarity is only in terms of detecting bright and dark circles such that it can be seen in Listing 3. Figure 14 demonstrates $radius \in [20, 60]$ and Sensitivity = 0.6. It can be seen that although radius is same, less circles are detected because of Sensitivity decrease. It can be concluded that Sensitivity decrease also decreases amount of circles detected. Figure 15 depicts $radius \in [20, 50]$ and Sensitivity = 0.9 meaning that although Sensitivity is same as in Figure 13, radius is decreased. Therefore, it can be seen that largest circles are not detected as they have larger radius than 50. Decrease in circle detection may be due to Sensitivity, non sufficient circle radius or perhaps maybe because of ObjectPolarity as incorrect color maybe detected. ObjectPolarity difference can be seen in Figure 13,14 and 15 such that until the third sub images only the bright circles are detected. Lastly, Runtime of imfindcircles function(hough circle detection) can be seen in Figure 16.

```
Elapsed time is 0.093692 seconds.
```

Figure 16: Running Time of Hough Circle Detection

To conclude, in this report Hough Transform Techniques for lines and circles are explained in detailed manner. Algorithm, working principle, parameters and results are explained, in addition functional parameters of Hough Transform in MatLab such as Sensitivity and Threshold are indicated in detail. Therefore, it would be reasonable to declare that Hough Transform Techniques achieve high level of success given that parameters are tuned perfectly.

Listing 3: MatLab Code for Hough Circle Detection

```
function [J, theta, rho] = lab4houghcircles(img)
    [row, col, ch] = size(img);
    %A = imread('circlesBrightDark.png');
    if(ch == 3)
        img = rgb2gray(img);
    end

   %Circles
    [centers, radii,metric] = imfindcircles(img,[20 60]);

    subplot(1,3,1);imshow(img);title(['Detected Circles using
       Hough Transform' newline '20 =< Radius =< 60']);
    hold on;viscircles(centers, radii,'Color','b');

    [centers2, radii2,metric2] = imfindcircles(img,[20 60], '
       Sensitivity', 0.9);

    subplot(1,3,2);imshow(img);title(['Detected Circles using
       Hough Transform' newline '20 =< Radius =< 60']);
    hold on;viscircles(centers2, radii2,'Color','r');

    [centersDark, radiiDark,metricDark] = imfindcircles(img,[20
       60], 'Sensitivity', 0.9','ObjectPolarity','dark');
    [centersBright, radiiBright,metricBright] = imfindcircles(img
       ,[20 60], 'Sensitivity', 0.9','ObjectPolarity','bright');

    subplot(1,3,3);imshow(img);title(['Detected Circles using
       Hough Transform' newline '20 =< Radius =< 60']);
    hold on;viscircles(centersBright, radiiBright,'Color','b');
       viscircles(centersDark, radiiDark,'LineStyle','--');
end
```