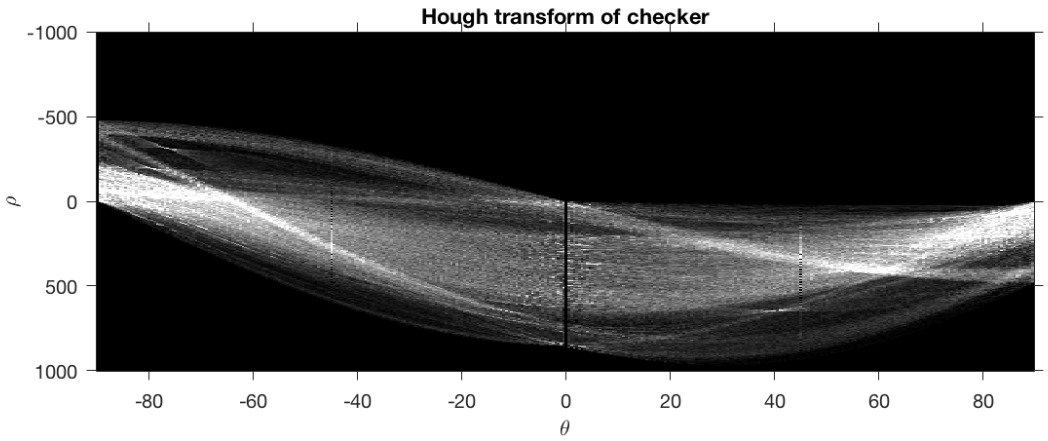
Aynur Süne - 21041

EE417 Post-Lab #4

Line Detection (Hough Lines)

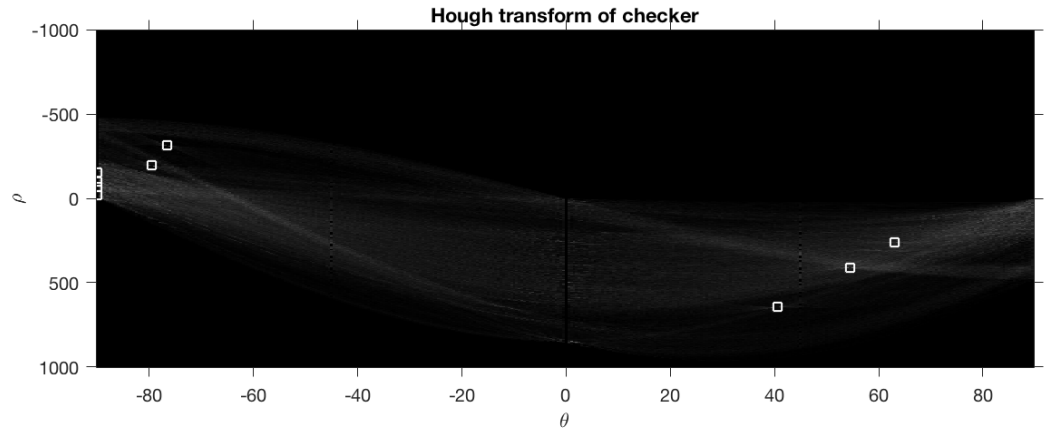
Hough Transform is a method for detecting straight lines (and curves) in images. Main idea is to map a difficult pattern problem into a simple peak detection problem.

```
function lab4houghlines(img)
tic;
[r,c,color] = size(img);
if(color==3)
    img = rgb2gray(img);
end

    BW = edge(img,'canny'); % extract edges
    [H,T,R] = hough(BW,'RhoResolution',0.5,'Theta',-90:0.5:89.5);

    subplot(2,1,1);
    imshow(BW);

       % hough matrix
      subplot(2,1,2);
      imshow(imadjust(rescale(H)),'XData',T,'YData',R,'InitialMagnification','fit')
;
      title('Hough transform of checker');
      xlabel('\theta'), ylabel('\rho');
      axis on, axis normal, hold on;
      colormap(gca,hot);

       P  = houghpeaks(H,10);
      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');


       lines = houghlines(BW,T,R,P,'FillGap',10,'MinLength',7);
       figure, imshow(img), hold on
       max_len = 0;
       min_len = 1000;
       for k = 1:length(lines)
         xy = [lines(k).point1; lines(k).point2];
         plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

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

         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');%longest line
   plot(xy_short(:,1),xy_short(:,2),'LineWidth',2,'Color','red');  % shortest line

    toc;
end
```

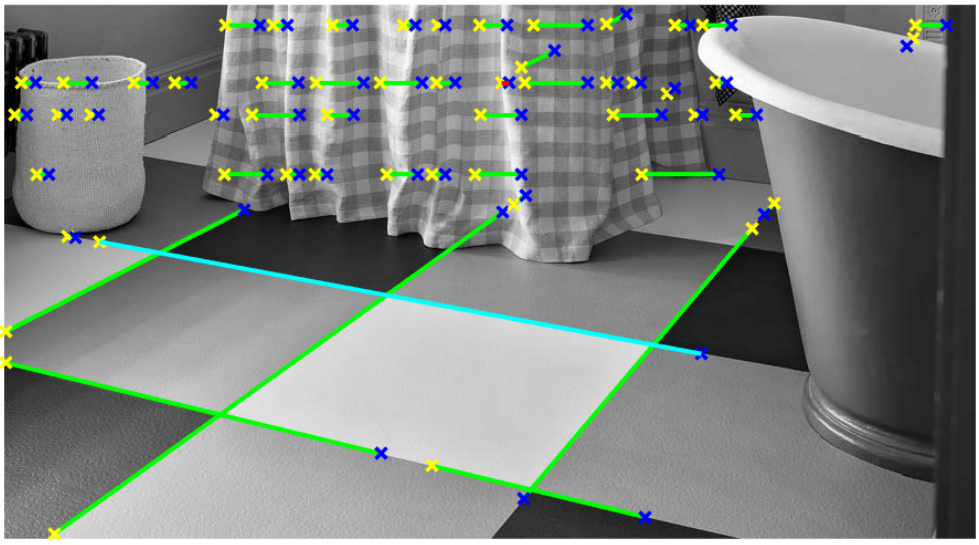


First, image is converted to grey-scale, then Canny Edge Detection Algorithm and Hough Transform function are applied with appropriate built-in matlab functions. One curve is generated for each edge point, as can be observed in the Hough Matrix, the second figure (rescale(H) in code). Curves generated by collinear points in the gradient image intersect in peaks in the Hough transform space. These intersection points indicate line segments of the original image, as shown below.

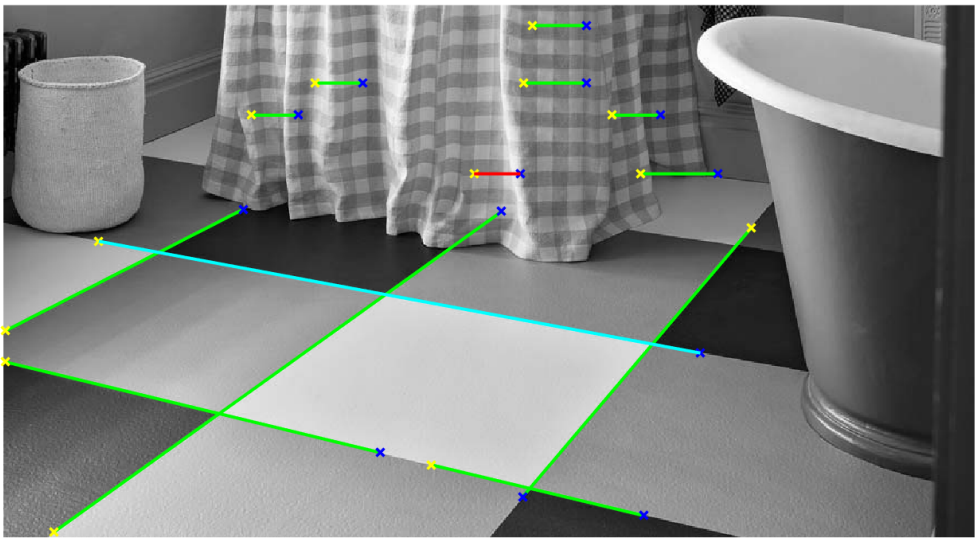
Number of Peaks = 10



Minimum Length = 7



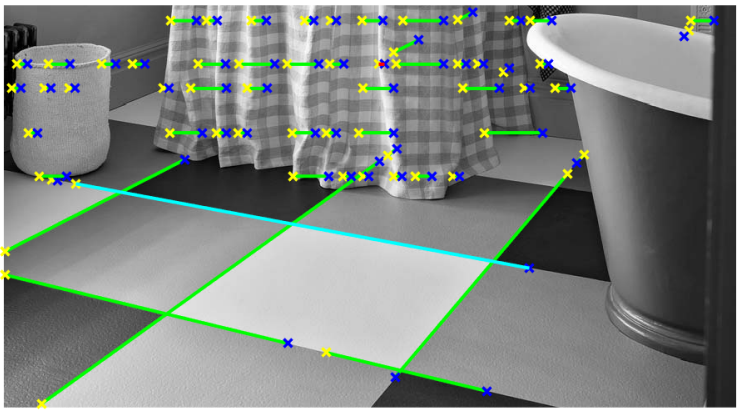
Minimum Length = 40



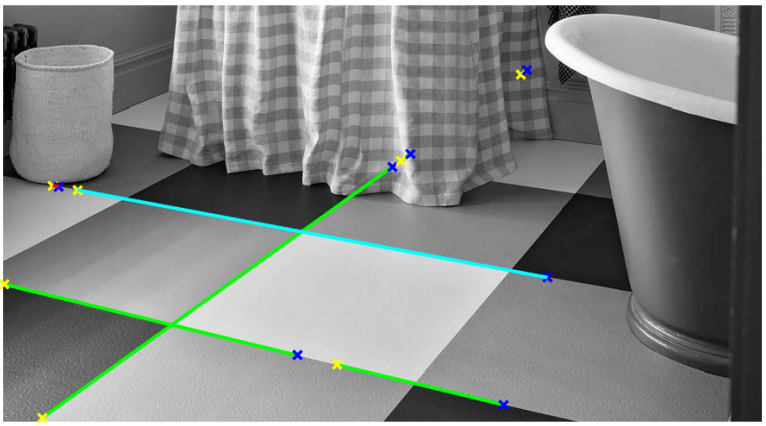
When threshold for number of peaks is 10 and minimum length is 7, algorithm is able to show more edges than when it is equal to 40, but longest edge is the same for both.

Minimum length found when minimum length parameter equals to 7 is 7. Maximum length is 554.2797. When minimum length is 40, minimum length line found is 40 and maximum length line is 554.2797.

Minimum *Length* = 7 & *Threshold* = 0.1*max(H(:))

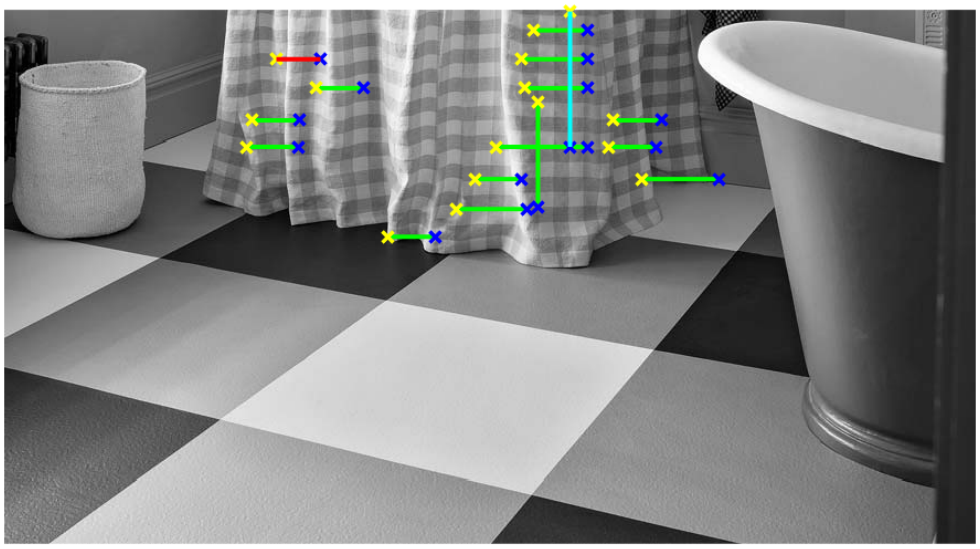


Minimum Length = 7 & *Threshold* = 0.9*max(H(:))



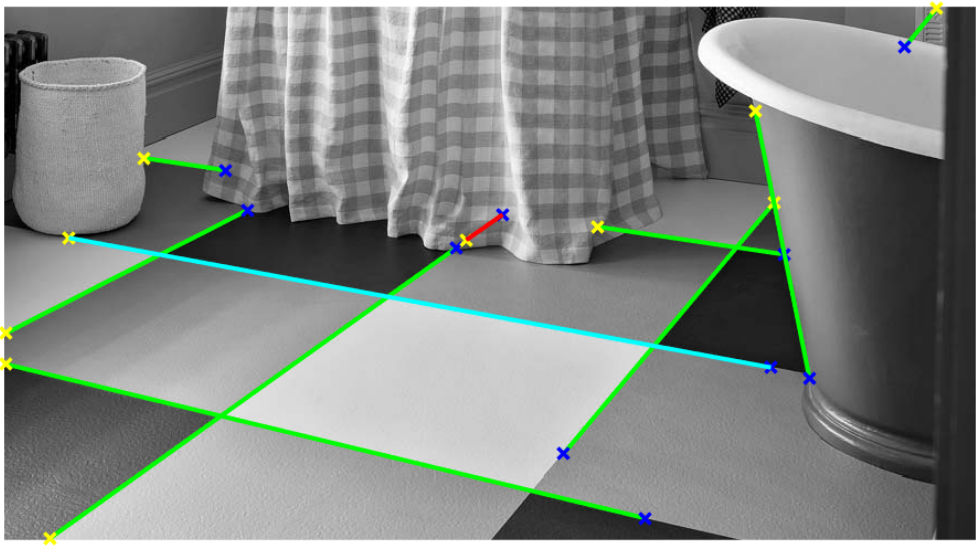
When threshold is low, result is same with default because min length is low so most of the lines are already detected. But when threshold increased, less strong lines are able to pass the threshold. Maximum and minimum lengths found are still same.

Rho = 0.1 & Min Length = 40



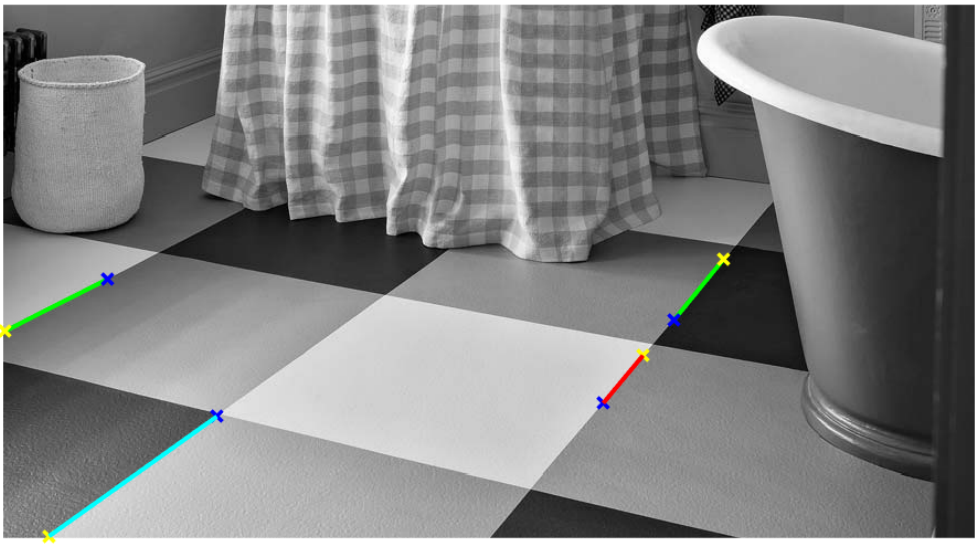
Small Rho Resolution value finds smaller lines even minimum length parameter is high. Minimum length found is 40 and maximum length is 122.

Rho = 0.9 & Min Length = 40



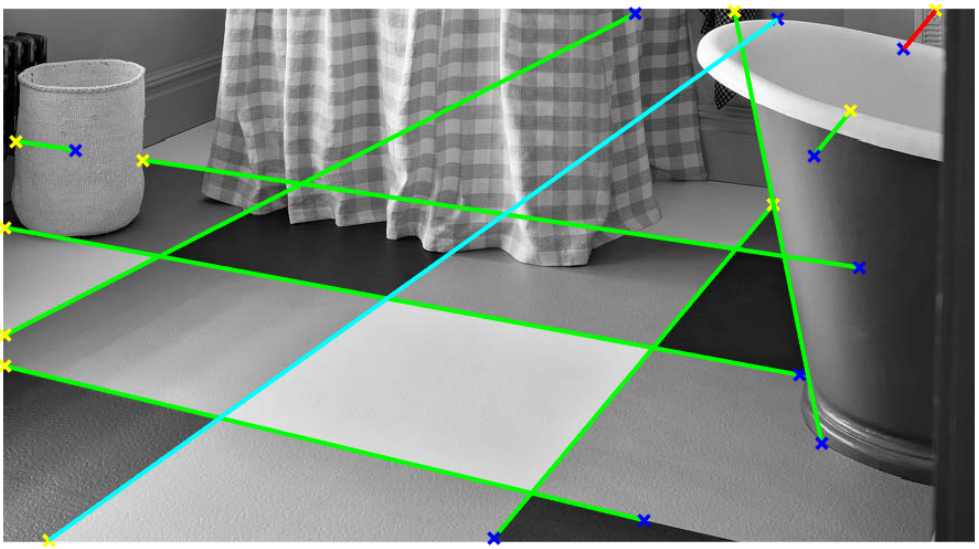
It can be observed that when rho resolution increased, longer lines tend to be detected. Minimum length found is 40.2244 and maximum is 646.6723.

Rho = 0.9 & Min Length = 40 & Fill Gap = 2



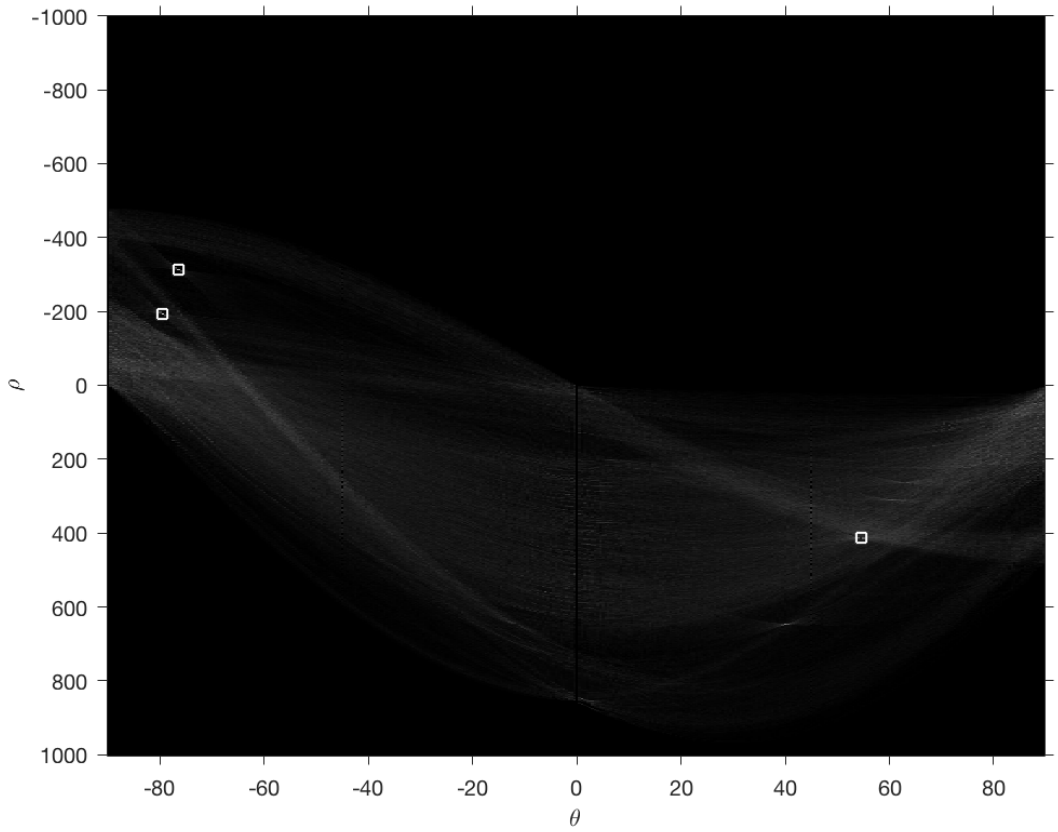
Minimum line found is 56.0803 and maximum is 186.4618.

Rho = 0.9 & Min Length = 40 & Fill Gap = 50



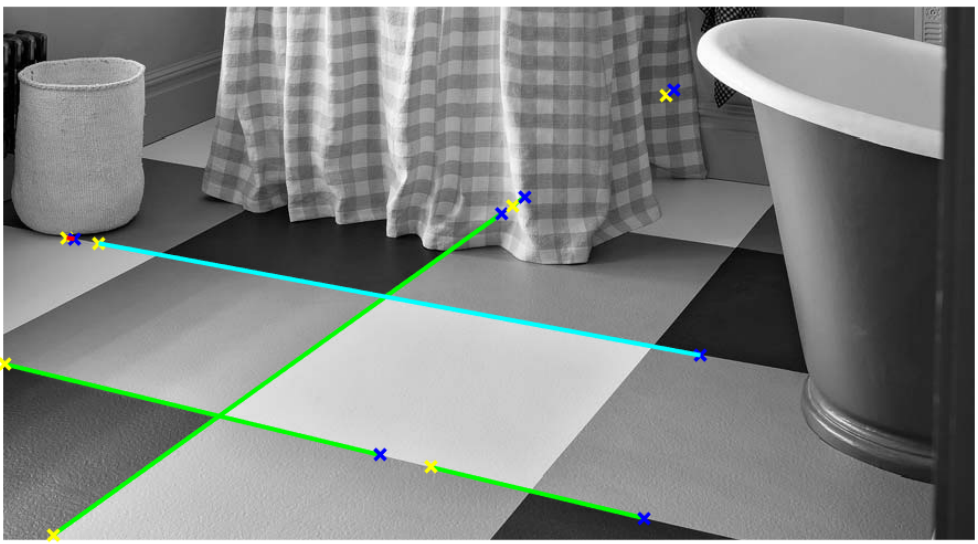
Minimum line found is 45.4533 and maximum is 810.2469. Maximum length is achieved with this example. Other lines are very large as well. Fill Gap parameter changes the range of lengths of lines found. We observed longer lines than when fill gap is 2.

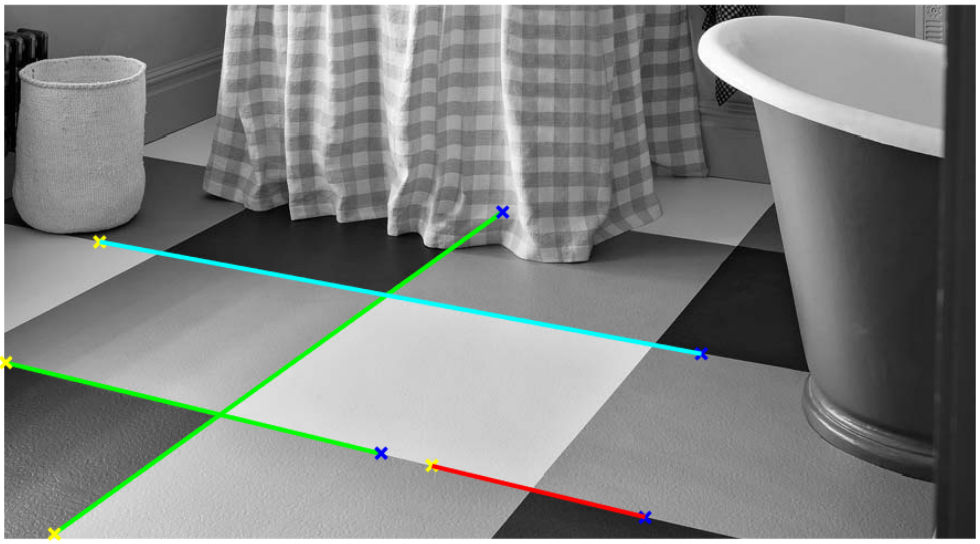
Number of Peaks = 3



Minimum Length Threshold = 7

Minimum Length = 40





When number of peaks equals to 10, it was able to detect more edges than when it is 3 because strong edges result in a higher peak value than weak edges. So when number of peaks is 3, output only contains some of the strongest edges. When this number increase, algorithm is able to find more peaks which can include weak edges with smaller peak values, as in image results where number of peaks is 10.

Number of Peaks = 3 & Minimum Length Threshold = 7: *Elapsed time is 1.193464 seconds.* Maximum length is 554.2797 and minimum is 7.0711

Number of Peaks = 3 & Minimum Length Threshold = 40: *Elapsed time is 1.096486 seconds.* Maximum length is 554.2797 and minimum is 198.4061.

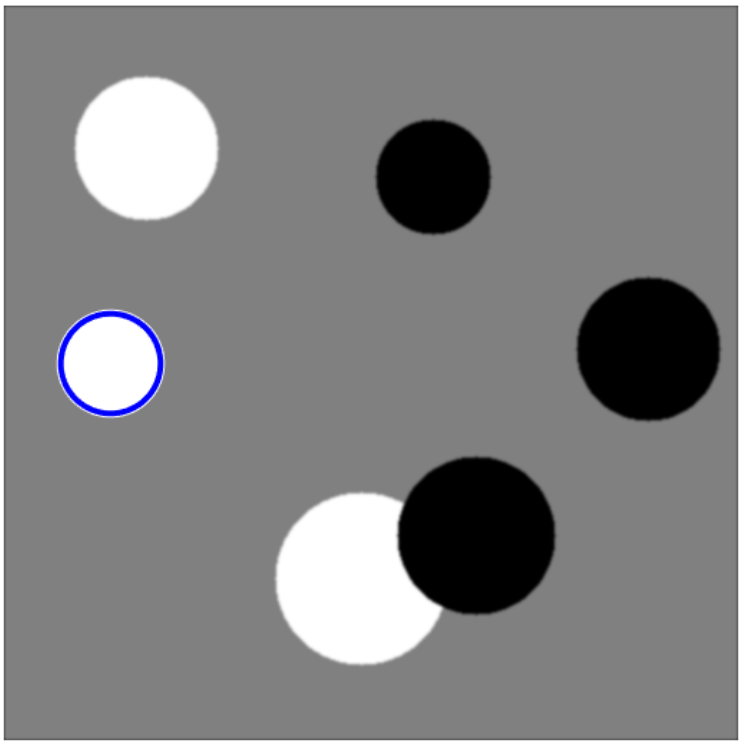
It is observed that when number of peaks is 3 which is lower than first examples, minimum length found is very large, others were around 40. This indicates that other lines are not strong enough to appear when minimum length parameter is 40.

Note that *Elapsed Time* obtained with tic-toc is included in this page as an example, but no other example is shown because there were no significant relationship observed when changing parameters.

Circle Detection (Hough Circles)

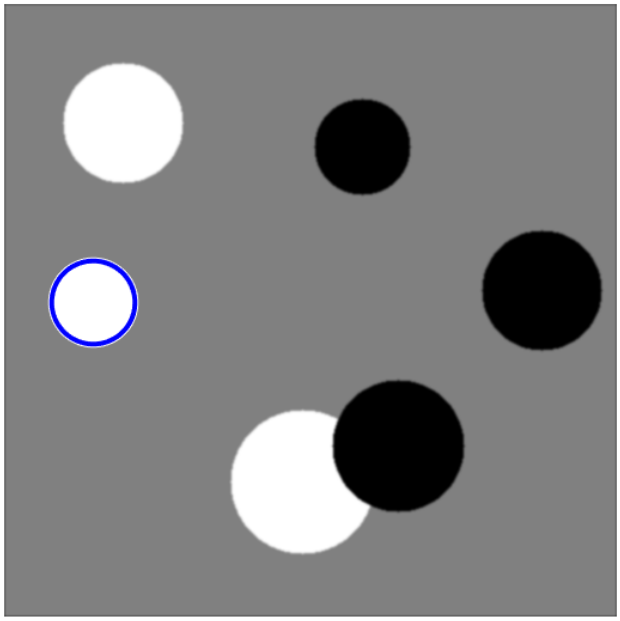
```
function lab4houghcircles(img)

tic;
    [r,c,color] = size(img);
        if(color==3)
             img = rgb2gray(img);
        end

         Rmin = 20;
         Rmax = 60;

          [centers, radii] = imfindcircles(img,[Rmin Rmax]);
          imshow(img);
            %hold on;
          viscircles(centers, radii,'Color','b');

          [centersSensLow, radiiSensLow] = imfindcircles(img,[Rmin
Rmax],'Sensitivity',0.5);
          [centersSensHigh, radiiSensHigh] = imfindcircles(img,[Rmin
Rmax],'Sensitivity',0.9);

          [centersDark, radiiDark] = imfindcircles(img, [Rmin
Rmax],'ObjectPolarity','dark');
          [centersBright, radiiBright] = imfindcircles(img, [Rmin
Rmax],'ObjectPolarity','bright');

          viscircles(centersSensHigh, radiiSensHigh,'Color','b')
          viscircles(centersSensLow, radiiSensLow,'LineStyle','--');

          viscircles(centersBright, radiiBright,'Color','b');
          viscircles(centersDark, radiiDark,'LineStyle','--');
   toc;

end
```

20 <= Radius <= 40 with Default Settings

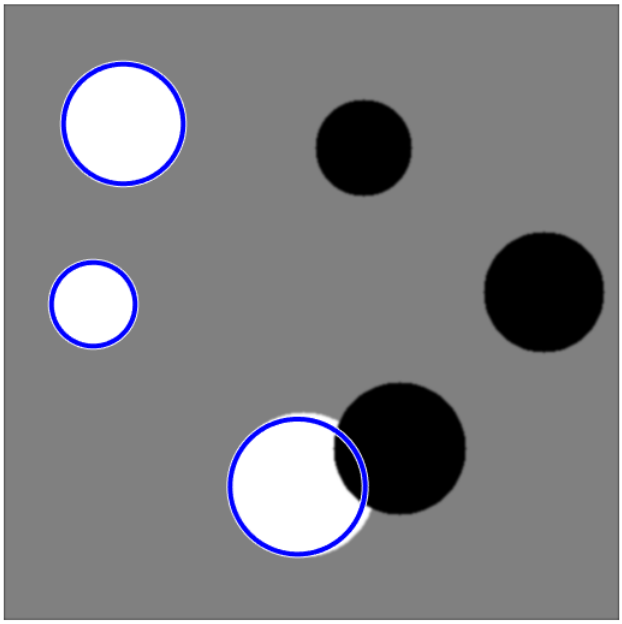


Aynur Süne - 21041

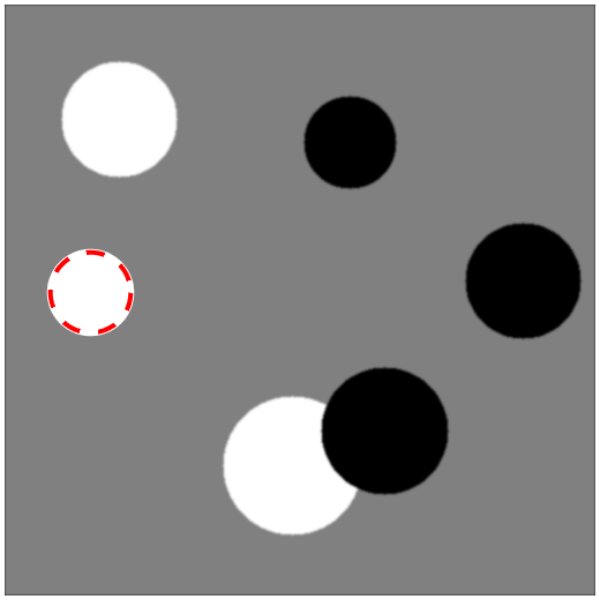
20 <= Radius <= 40 & Sensitivity = 0.9



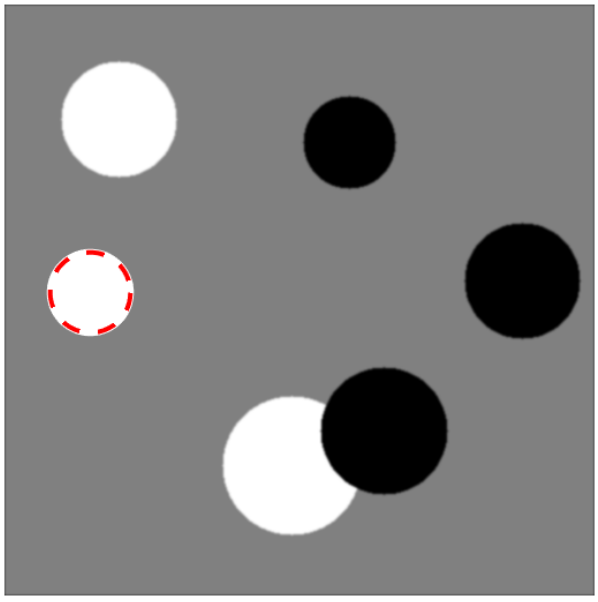
20 <= Radius <= 60 & Sensitivity = 0.9



20 <= Radius <= 40 & Sensitivity = 0.6

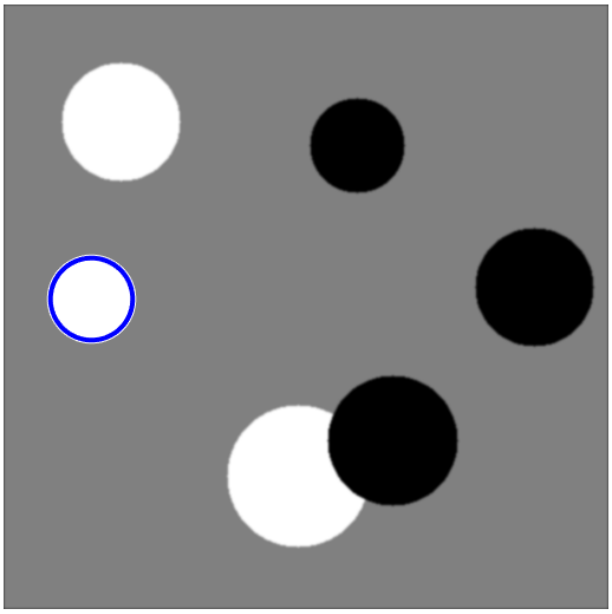


20 <= Radius <= 60 & Sensitivity = 0.6

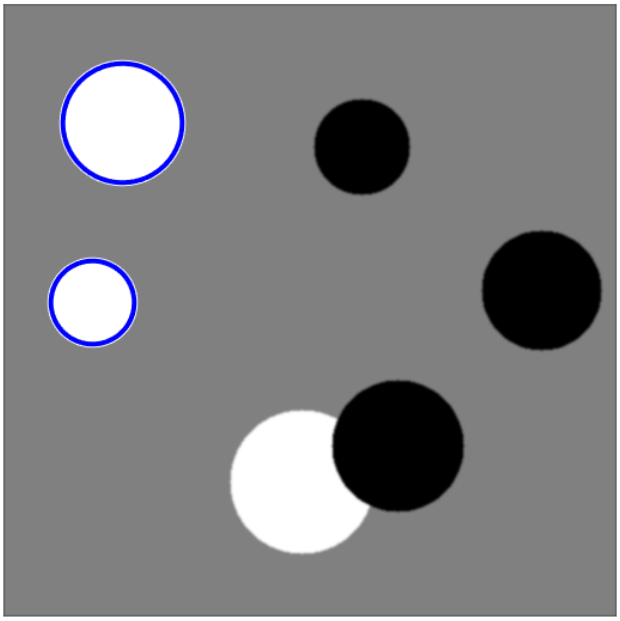


Small circle that is in the given radius range 20-60 is detected in both trial. Big circles that do not in the given radius range can be detected when sensitivity is high, otherwise they are not detected even they have the appropriate radius. We can observe that even a circle is in between radius range given, it cannot be detected with *relatively* low sensitivity value, sensitivity should be high to detect circles correctly. Note that under 0.6 sensitivity, none of the circles were detected therefore trials under 0.6 are not included. Sensitivity seems to be important for functions reliability to detect circles.

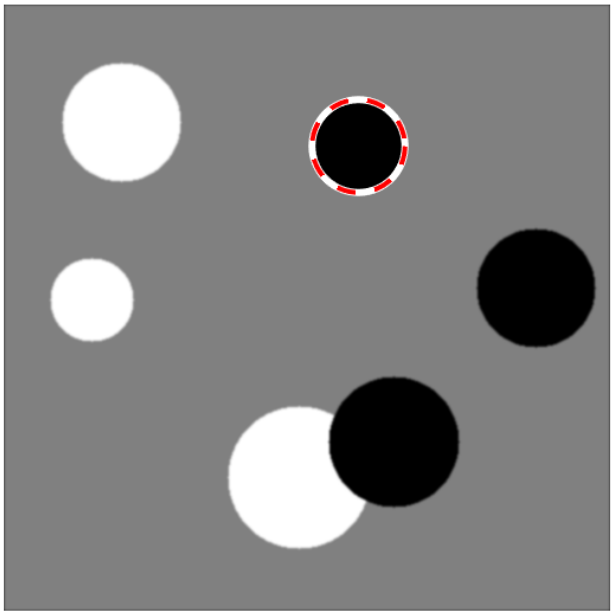
20 <= Radius <= 40 & Object Polarity = Bright



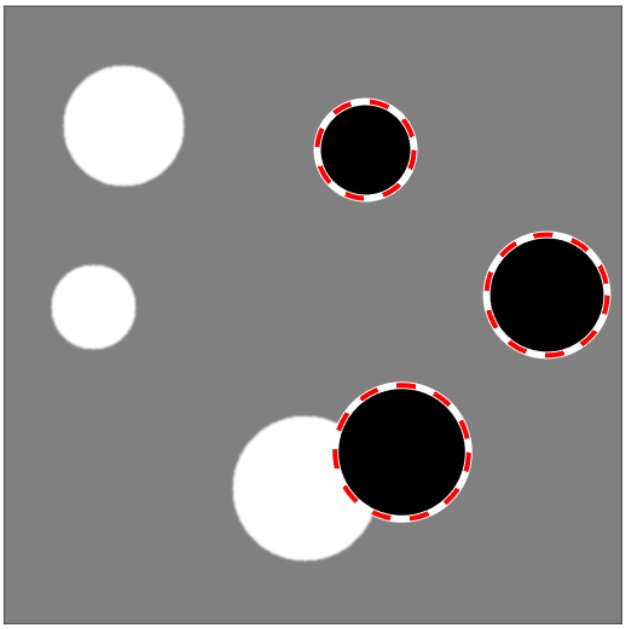
20 <= Radius <= 60 & Object Polarity = Bright



20 <= Radius <= 40 & Object Polarity = Dark



20 <= Radius <= 60 & Object Polarity = Dark



When object polarity parameter equals to Bright, it detects circles brighter than background, in this case white ones; and when parameter equals to Dark, it detects circles darker than background, like black ones. In these trials, black circles were successfully found when appropriate radius range given, but biggest bright circle could not be detected unless upper bound of the edge is set to 65. This may be related to the fact that biggest bright circle shadowed by a black circle, therefore it is harder to detect than other circles.