ECSE 4530 Image Processing

HW 5

Mar 5, 2018
Daniel Southwick
661542908

```
%% ==================================5.1====================================
```

Main Function

```
%% 1a
market = double(imread('market.png'));
x = 205 ; y = 696;
gx = market(x+1,y) - market(x,y);
gy = market(x,y+1) - market(x,y);
mag = sqrt((gx)^2+(gy)^2);
alp = atand(gy/gx);
```

Matlab Result

alp	30.9638
gx	-20
gy	-12
mag	23.3238
market	*533x800 double*
x	205
y	696

Using direct method, the gradient, magnitude and its angle at the point (205,696) is shown above. Gradient of x is -20, gradient of y is -12, magnitude of the gradient is 23.3238, angle is 30.9628 degrees.

Main Function

```
%% 1b
k = [-1 -2 -1; 0 0 0; 1 2 1]/8;
gx2 = imfilter(market, k);
gy2 = imfilter(market, k');
gx2b = gx2(x,y);
gy2b = gy2(x,y);
mag2 = sqrt(gx2b^2 + gy2b^2);
alp2 = atand(gy2b/gx2b);
```

Matlab Result

alp2	-16.0259
gx2	-23.5000
gy2	6.7500
k	[-0.1250,-0.250...
mag2	24.4502
market	*533x800 double*

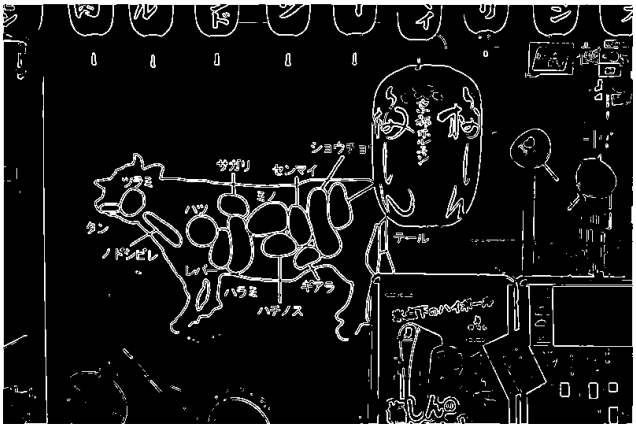
Using the sobel operators, the gradient, magnitude and its angle at the point (205,696) is shown above. Gradient of x is -23.5, gradient of y is 6.75, magnitude of the gradient is 24.4502, angle is -16.0259 degrees.

```
%% ==================================5.2=====================================
```

Main Function

```
%% 2a
magsob = sqrt(gx2.^2+gy2.^2);
angsob = atand(gy2./gx2);
imshow(magsob>50,[]);
```

Matlab Result



Above is the image which magnitude is greater than 50. White pixels in the magnitude greater than 50 means that pixel in the original picture has large gradients, which implies a sudden change of color, thus this magnitude picture detected the edges from the original picture.

Main Function

```
%% 2b
imshow(abs(angsob-45) < 20,[]);
```

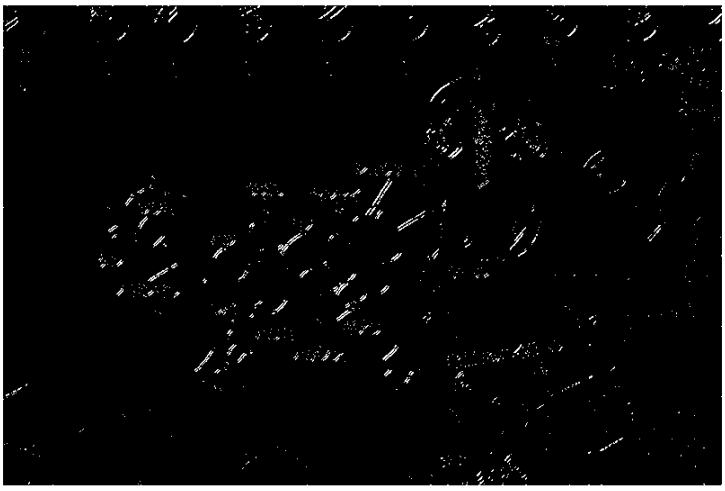
Matlab Result

Above is the image which angle is in the range of 25 degrees and 60 degrees. Thus we can see lots of white spot that has an angle in between that range. All pixels from the original picture that has an angle between 25 and 60 will be shown here. The cow in the picture has straight horizontal lines thus the angle should be 90 or 0 degrees thus it's dark.

Main Function

```
%% 2c
imshow(((magsob>50)&(abs(angsob-45) < 20)),[]);
```

Matlab Result



Above is the image which we combined part a and part b using logical AND. This will combine the two feature that the previous two picture has. Which means this image detects the edges that has an angle between 20 to 65 degrees in the original picture.

```
%% ==================================5.3=================================
```

Main Function

```
%% 3a
deck = double(imread('deck.png'));
BW = edge(deck,'Canny');
imshow(BW,[]);
```

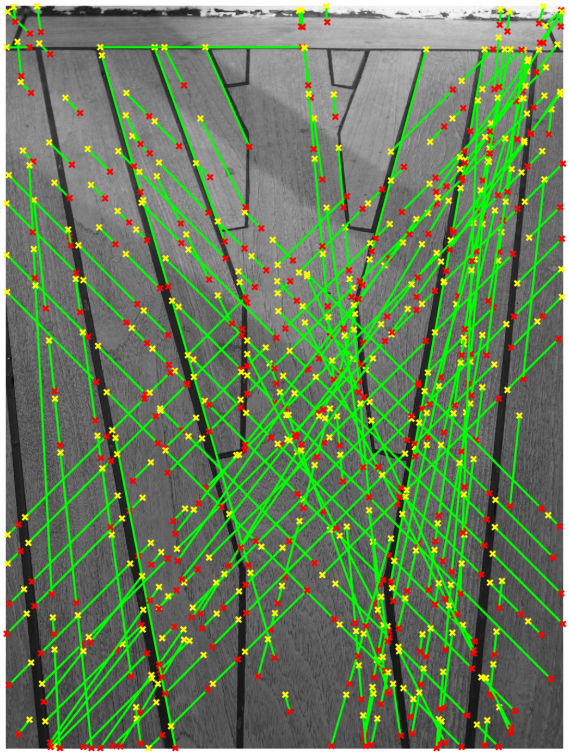
Matlab Result

Above is the Canny edges of the picture deck using default Matlab thresholds.

Main Function

```
%% 3b
[H,theta,rho] = hough(BW);
peaks = houghpeaks(H,50);
lines = houghlines(BW,theta,rho,peaks);
figure, imshow('deck.png'), hold on
max_len2 = 0;
for k = 1:length(lines)
   xy2 = [lines(k).point1; lines(k).point2];
   plot(xy2(:,1),xy2(:,2),'LineWidth',2,'Color','green');
   % Plot beginnings and ends of lines
   plot(xy2(1,1),xy2(1,2),'x','LineWidth',2,'Color','yellow');
   plot(xy2(2,1),xy2(2,2),'x','LineWidth',2,'Color','red');
   % Determine the endpoints of the longest line segment
   len2 = norm(lines(k).point1 - lines(k).point2);
   if ( len2 > max_len2)
      max_len2 = len2;
      xy_long = xy2;
   end
end
```

Matlab Result



Above is the super imposed deck image with the original pictures. We took the 50 highest peaks of the hough function.

%% 3c

The result indeed makes sense, there are many lines detected at + - 45 degrees because hough transform are used to find the long, straight lines in the clutter. And as you can see from the original deck picture, the major lines are mostly in that angle, along with the wood pattern.

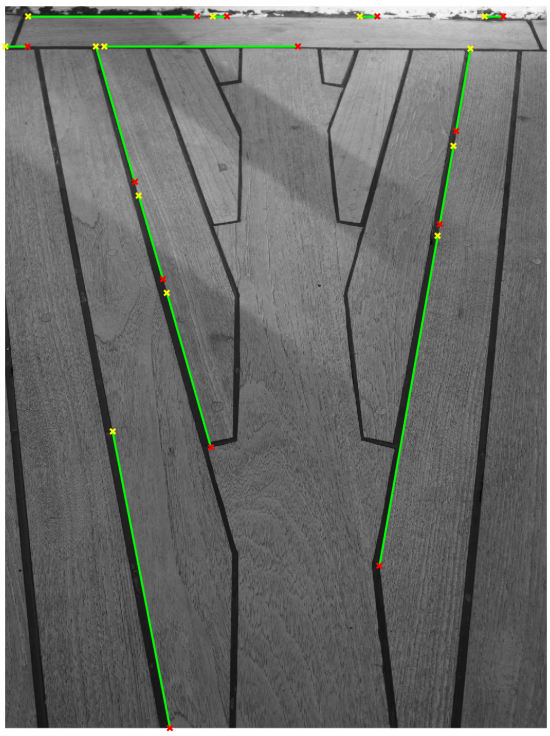
Main Function

```
%% 3d
BW2 = edge(deck,'Sobel');
[H2,theta2,rho2] = hough(BW2);
peaks2 = houghpeaks(H2,50);
lines2 = houghlines(BW2,theta2,rho2,peaks2);
figure, imshow('deck.png'), hold on
max_len2 = 0;
for k2 = 1:length(lines2)
   xy2 = [lines2(k2).point1; lines2(k2).point2];
   plot(xy2(:,1),xy2(:,2),'LineWidth',2,'Color','green');

   % Plot beginnings and ends of lines
   plot(xy2(1,1),xy2(1,2),'x','LineWidth',2,'Color','yellow');
   plot(xy2(2,1),xy2(2,2),'x','LineWidth',2,'Color','red');

   % Determine the endpoints of the longest line segment
   len2 = norm(lines2(k2).point1 - lines2(k2).point2);
   if ( len2 > max_len2)
      max_len2 = len2;
      xy_long = xy2;
   end
end
```

Matlab Result

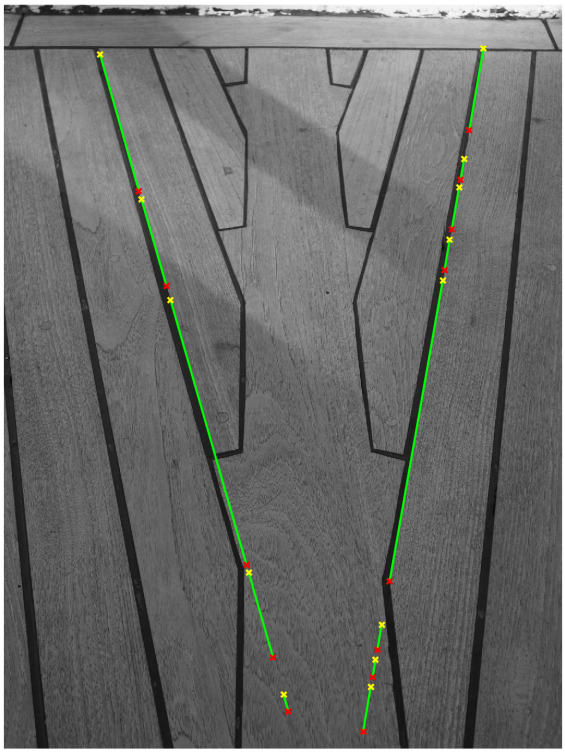


The above picture applied the same procedure in part b with the Sobel edges. It detects way less edges than the Canny edges but these detections are on the major edges without detecting any spurious lines caused by the wood pattern.

Main Function

```
%% 3e
[H,theta,rho] = hough(BW);
peaks = houghpeaks(H,2);
lines = houghlines(BW,theta,rho,peaks);
figure, imshow('deck.png'), hold on
max_len2 = 0;
for k = 1:length(lines)
   xy2 = [lines(k).point1; lines(k).point2];
   plot(xy2(:,1),xy2(:,2),'LineWidth',2,'Color','green');
   % Plot beginnings and ends of lines
   plot(xy2(1,1),xy2(1,2),'x','LineWidth',2,'Color','yellow');
   plot(xy2(2,1),xy2(2,2),'x','LineWidth',2,'Color','red');
   % Determine the endpoints of the longest line segment
   len2 = norm(lines(k).point1 - lines(k).point2);
   if ( len2 > max_len2)
      max_len2 = len2;
      xy_long = xy2;
   end
end
```

Matlab Result



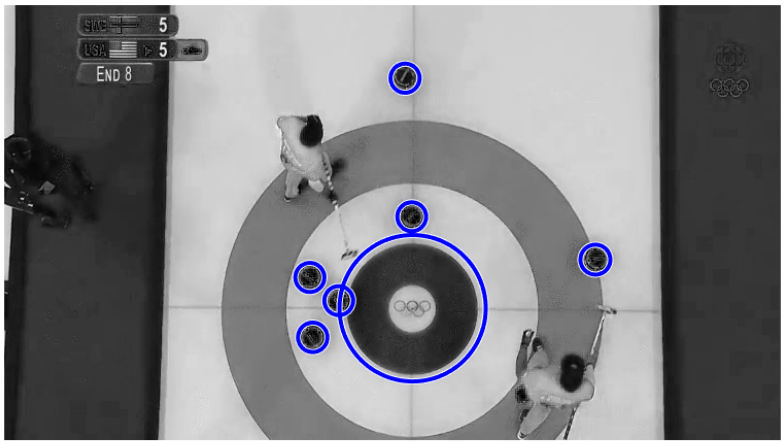
The threshold on the hough function was adjusted really low to get rid of all the spurious lines. In this case, we only took the hihest 2 peaks, if we changed to 3, another apurious lines will be added.

```
%% ==================================5.4=================================
```

Main Function

```
%% 4
curling = imread('curling.jpg');
curling = curling(:,:,3);
imshow(curling);
[centers, radii] = imfindcircles(curling,[13 350],'ObjectPolarity','dark');
viscircles(centers, radii,'EdgeColor','b');
```

Matlab Result



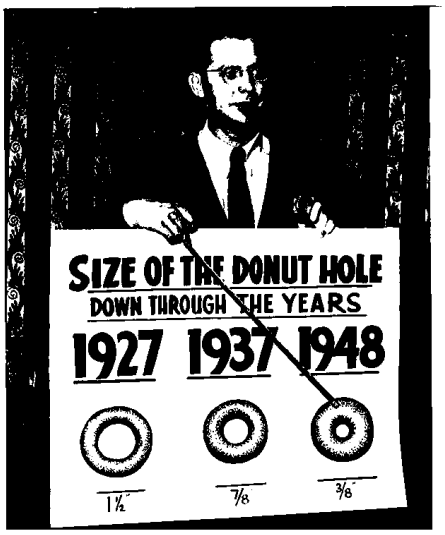
After playing around with different object polarity and edge threshold settings, imfindcircles came out to produce the best result when the color channel is in blue, the polarity is dark and the range of the radius is between 13 to 350. We changed the picture into blue channel since red stands out (turns into a darker color) when the channel is blue. And since all the actual curling balls and the center of the score mark are red, they were shown in a really dark color. Thus we want to detect the darker circles in object polarity. Lastly, we played around with the range of the radius to get rid of all spurious circles. This setting successfully detected all curling balls and the center of goal.

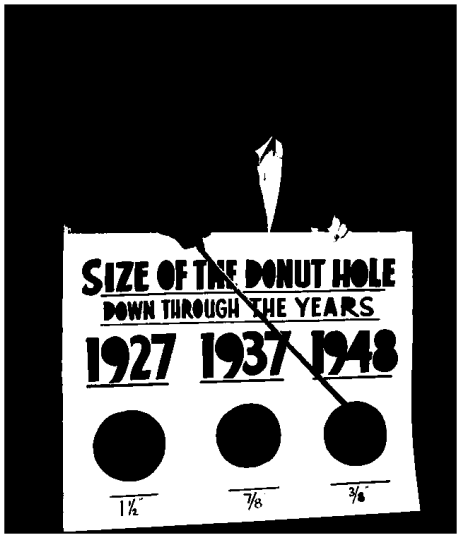
```
%% ==================================5.5=================================
```

Main Function

```
%% 5
important = imread('important.png');
important = important>180;
imshow(important,[]);
Ilabel = bwlabel(important,8);
stat = regionprops(Ilabel,'Area');
area=[stat.Area];
[stat_area,stat_index]=max(area);
important2 = ismember(Ilabel,stat_index);
imshow(important2);
```

Matlab Result





(Above Greyscale 180) *(Isolate largest white region)*

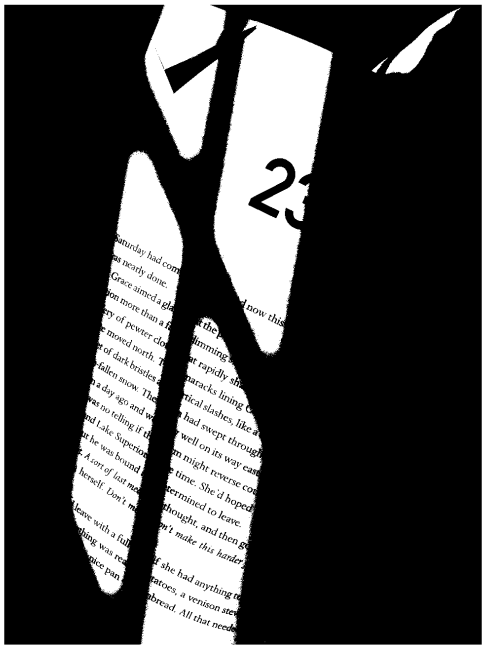
The picture on the left is what we find above a grayscale level of 180. The picture on thr right is used bwlabel and regionprops to isolate the largest white region.

```
%% ===================================5.6=====================================
```

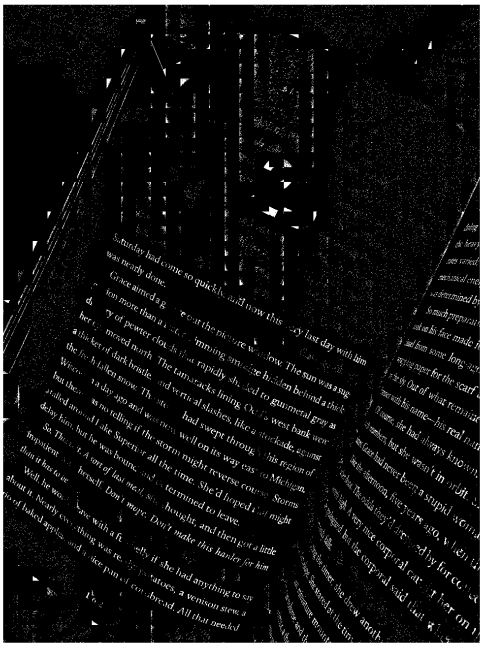
Main Function

```
%% 6
page = imread('page.png');
level = graythresh(page);
BW6 = imbinarize(page,level);
imshow(BW6);
fun = @(block_struct)...
block_struct.data<mean2(block_struct.data)-2*std2(block_struct.data);
page2 = blockproc(page,[60 60],fun);
imshow(page2,[]);
```

Matlab Result



(Otsu's global threshold)



(Blockproc local threshold)

By using the graythresh, we found the otsu's threshold, which is 0.4784*255 = 122 from the matlab and applied it to the image, the result is shown above on the left, it looks ok where the light has shined on the book, but the rest of the texts were wiped out since they were in the shadow. Thus the global threshold is suboptimal.

The blockproc processed 60x60 blocks of the image and separated the pixel intensity by the formula provided in the question. The result has been shown on the above right. It is a lot better compare to global threshold. Local threshold works better for the images that has no strong peaks in histogram and objects are small with respect to the image size. This picture has both qualities.