University of Balamand

Faculty of Engineering

Departments of Electrical & Computer Engineering

Course: CPEN528 Machine Vision

Assignment 1: Edge Detection

Submitted to: **Dr. Issam Dagher**

Prepared by: **Hala Msalem** (A2020119)

Thursday, March 7, 2024

Table of Contents

Objective:

The objective of this assignment is to perform edge detection on an image by applying different masks: Gaussian Filter and derivation, Prewitt, Robert, Sobel, and first derivative of the gaussian mask (Canny). Different thresholds will be used for detection.

Gray scale Image:

Gaussian Filter and differentiation:

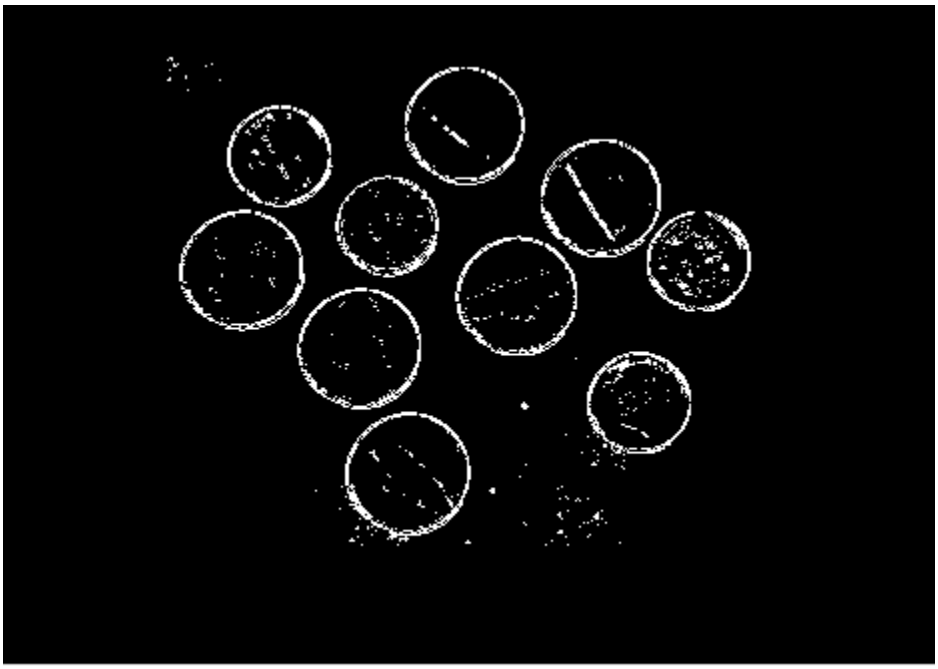
Gaussian Filter is used for the smoothing stage and the derivation is for the differentiation, and finally comparing the threshold value to the magnitude to get the real edges.

MATLAB code:

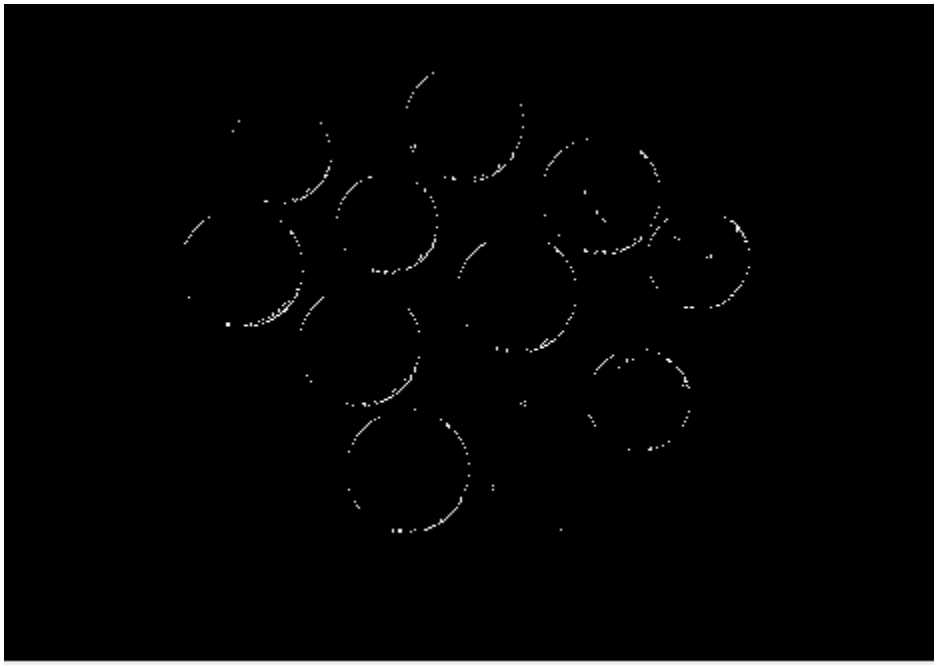
```
sigma_squared=1;
g = zeros(3);
k1=1;
for i = -1:1
    k2=1;
    for j=-1:1
        value = exp(-(i^2+j^2) / (2 * sigma_squared));
        g(k1,k2)=value;
        k2=k2+1;
    end
    k1=k1+1;
end

A = imread('coins.png');
A1 = conv2(A, g, 'same');

g1 = [-1 +1];
I1 = conv2(A1, g1, 'same'); %fx
I2=abs(I1); %|fx|
g2= [-1 ;+1];
I3 = conv2(I, g2, 'same'); %fy
I4=abs(I3); %|fy|
mag=I2+I4; % magniuted
angle=atan(I4/I2);

th =200;
n=size(mag,1)
m=size(mag,2)
im = zeros(n,m);
for i = 1: n
    for j= 1: m
        if mag(i,j)>th
            im(i,j)=1;
        else im(i,j)=0;
        end
    end
end

imshow(im,[]);
```

Results:

T=200:



T=300:



We can see that a higher threshold results into less edges.

Prewitt:

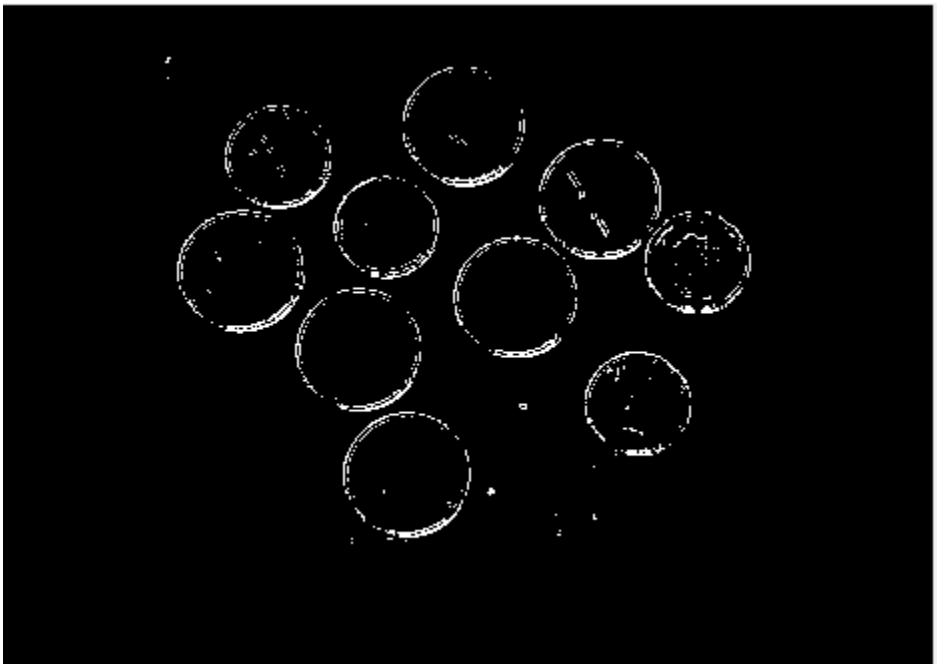
Prewitt mask does the smoothing and differentiation stage then we compare the threshold value to the magnitude to get the real edges.

MATLAB code:

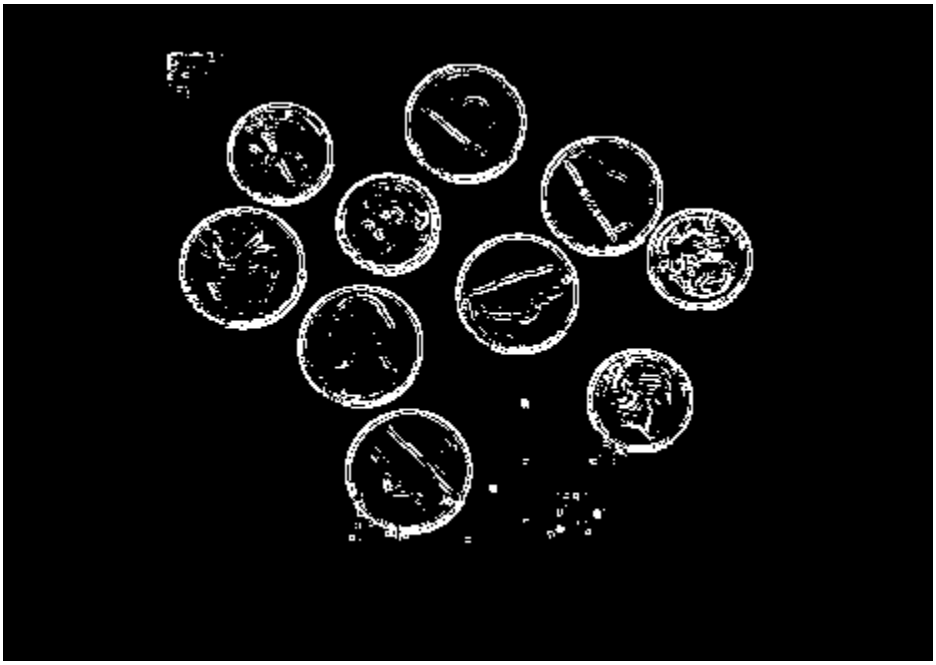
```
%prewitt mask
I = imread('coins.png');

p1 = [-1 0 +1;
    -1 0 +1;
    -1 0 +1];
I1 = conv2(I, p1, 'same'); %fx
I2=abs(I1); %|fx|

p2= [-1 -1 -1;
    0 0 0;
    1 1 1];
I3 = conv2(I, p2, 'same'); %fy
I4=abs(I3); %|fy|
mag=I2+I4; % magniuted

angle=atan(I4/I2);

th =400;
n=size(mag,1)
m=size(mag,2)
im = zeros(n,m);
for i = 1: n
    for j= 1: m
        if mag(i,j)>th
            im(i,j)=1;
        else im(i,j)=0;
        end
    end
end

imshow(im,[]);
```

Results:

T=400:



T= 200:



We can see that a lower threshold results into more edges.

Robert:

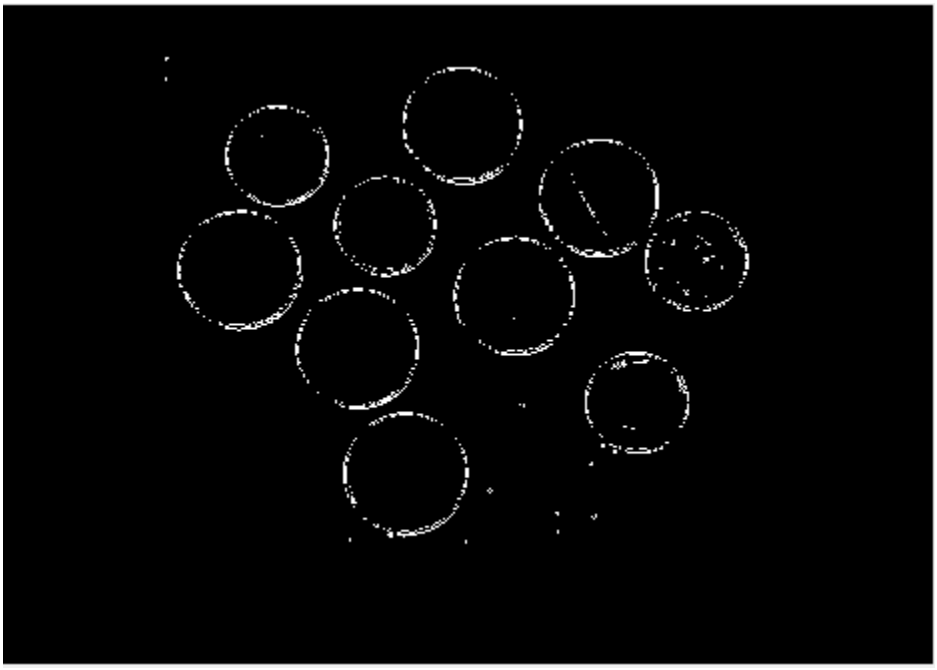
Robert mask does the smoothing and differentiation stage then we compare the threshold value to the magnitude to get the real edges.

MATLAB code:

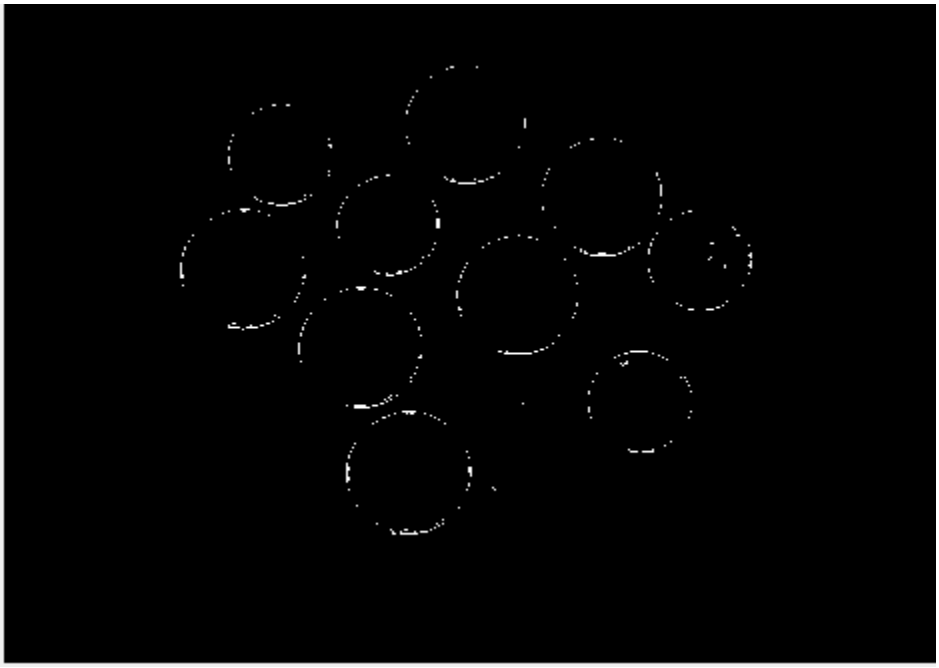
```
%robert mask
I = imread('coins.png');

g1 = [ 0 +1;
    -1 0 ];
I1 = conv2(I, g1, 'same'); %fx
I2=abs(I1); %|fx|

g2= [1 0;
    0 -1];
I3 = conv2(I, g2, 'same'); %fy
I4=abs(I3); %|fy|
mag=I2+I4; % magniuted

angle=atan(I4/I2);

th =200;
n=size(mag,1)
m=size(mag,2)
im = zeros(n,m);

for i = 1: n
    for j= 1: m
        if mag(i,j)>th
            im(i,j)=1;
        else im(i,j)=0;
        end
    end
end

imshow(im,[]);
```

Results:

T= 200:



T=300:



Sobel:

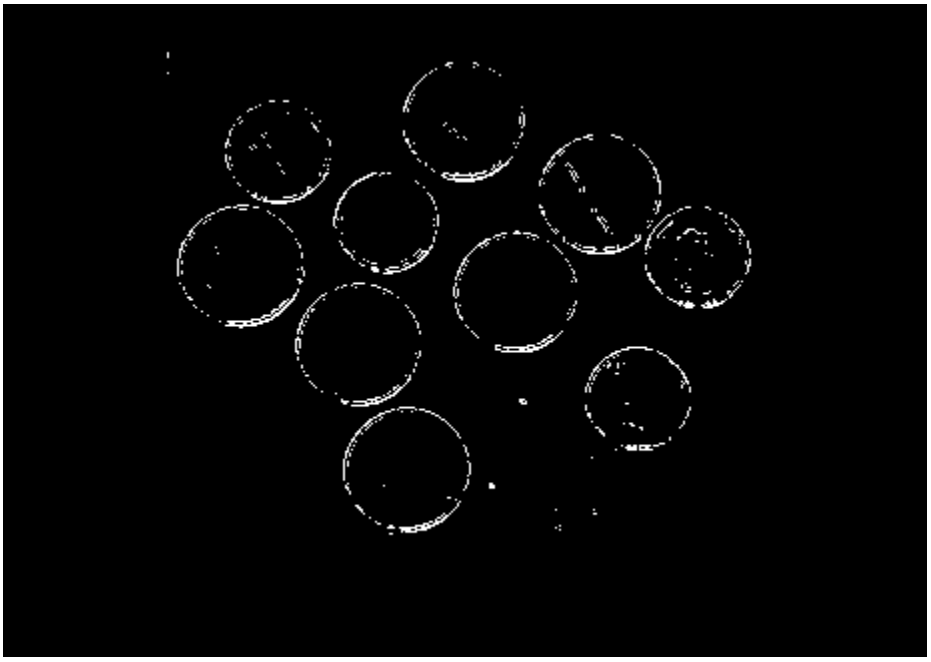
Sobel mask does the smoothing and differentiation stage then we compare the threshold value to the magnitude to get the real edges.

MATLAB code:

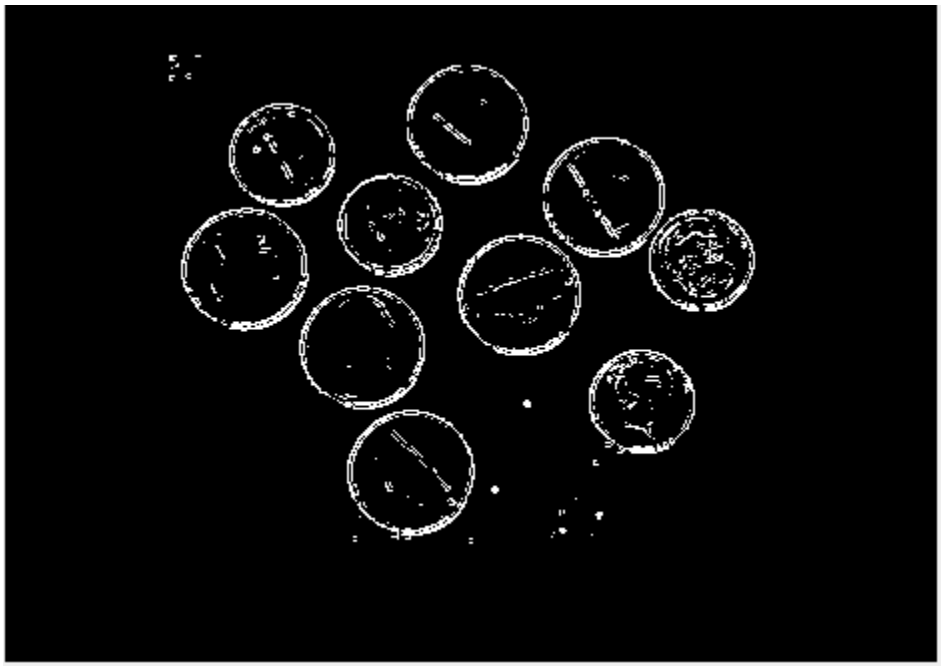
```
%sobel mask
I = imread('coins.png');

g1 = [-1 0 +1;
    -2 0 +2;
    -1 0 +1];
I1 = conv2(I, g1, 'same'); %fx
I2=abs(I1); %|fx|

g2= [-1 -2 -1;
    0 0 0;
    1 2 1];
I3 = conv2(I, g2, 'same'); %fy
I4=abs(I3); %|fy|
mag=I2+I4; % magniuted

angle=atan(I4/I2);

th =600;
n=size(mag,1)
m=size(mag,2)
im = zeros(n,m);
for i = 1: n
    for j= 1: m
        if mag(i,j)>th
            im(i,j)=1;
        else im(i,j)=0;
        end
    end
end

imshow(im,[]);
```

Results:

T=600:



T=400:



First derivative of the gaussian mask (Canny):

Canny's mask does the smoothing and differentiation stage. Then we normalize the magnitude and apply non-maximum suppression to thin the edges. Finally, we apply double thresholding to the new magnitude matrix to get the real edges.

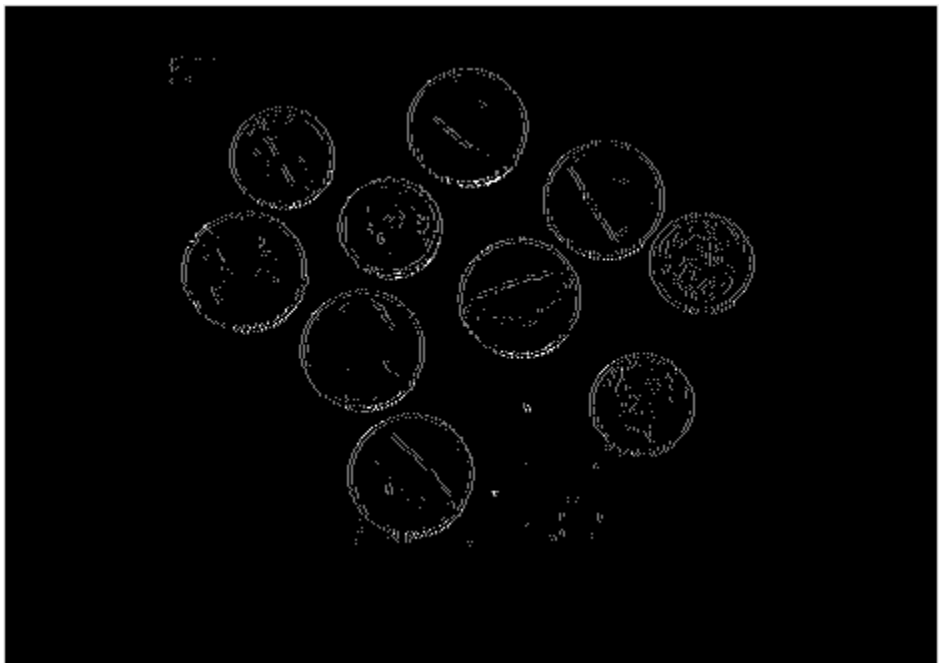
MATLAB code:

```
sigma_squared=1;

gx = zeros(3);
gy = zeros(3);
k1=1;

for i = -1:1
    k2=1;
    for j=-1:1

        valuex = (-i/sigma_squared)* exp(-(i^2+j^2) / (2 * sigma_squared));
        gx(k1,k2)=valuex;

        valuey = (-j/sigma_squared)*exp(-(i^2+j^2) / (2 * sigma_squared));
        gy(k1,k2)=valuey;
        k2=k2+1;
    end
    k1=k1+1;
end

A = imread('coins.png');

Ax = conv2(A, gx, 'same');
Ix=abs(Ax);

Ay= conv2(A, gy, 'same');
Iy=abs(Ay);

mag=Ix+Iy;
angle=atand(Iy/Ix);

%normalize the mag
% maxValue = max(mag, [], 'all');
% Nmag = (mag / maxValue)*100;

% Quantize gradient angles to four orientations: horizontal, vertical, and two
diagonals
angle_quantized = zeros(size(A));
angle_quantized((angle >= -22.5 & angle < 22.5) | (angle >= 157.5 & angle <= 180) |
(angle >= -180 & angle < -157.5)) = 0;  % Horizontal
angle_quantized((angle >= 22.5 & angle < 67.5) | (angle >= -157.5 & angle < -112.5))
= 45;  % Diagonal 1
angle_quantized((angle >= 67.5 & angle < 112.5) | (angle >= -112.5 & angle < -67.5))
= 90;  % Vertical
```

```
angle_quantized((angle >= 112.5 & angle < 157.5) | (angle >= -67.5 & angle < -22.5))
= 135;  % Diagonal 2


% Non-maximum suppression
for i = 2:size(A, 1) -1
    for j = 2:size(A, 2) -1
        switch angle_quantized(i, j)
            case 0  % Horizontal
                if mag(i, j) <= mag(i, j - 1) || mag(i, j) <= mag(i, j + 1)
                    mag(i, j) = 0;
                end
            case 45  % Diagonal 1
                if mag(i, j) <= mag(i - 1, j + 1) || mag(i, j) <= mag(i + 1, j - 1)
                    mag(i, j) = 0;
                end
            case 90  % Vertical
                if mag(i, j) <= mag(i - 1, j) || mag(i, j) <= mag(i + 1, j)
                    mag(i, j) = 0;
                end
            case 135  % Diagonal 2
                if mag(i, j) <= mag(i - 1, j - 1) || mag(i, j) <= mag(i + 1, j + 1)
                    mag(i, j) = 0;
                end
        end
    end
end


% Double thresholding
high_threshold_ratio = 0.7;
low_threshold_ratio = 0.4;
high_threshold = max(mag(:)) * high_threshold_ratio;
low_threshold = high_threshold * low_threshold_ratio;

edge_map = zeros(size(A));
edge_map(mag >= high_threshold) = 1;  % Strong edges
edge_map((mag >= low_threshold) & (mag < high_threshold)) = 0.5;  % Weak edges

% Edge tracking by hysteresis
for i = 2:size(A, 1) - 1
    for j = 2:size(A, 2) - 1
        if edge_map(i, j) == 0.5
            if any(edge_map(i-1:i+1, j-1:j+1) == 1)
                edge_map(i, j) = 1;
            end
        end
    end
end

% Display the resulting edge map
imshow(edge_map);
```

Results:

```
high_threshold_ratio = 0.7; (of the max value in magnitude)
low_threshold_ratio = 0.4; (of high threshold)
```



Conclusion:

To conclude, I was able to perform edge detection in different ways using various masks, by extracting fx and fy to get the gradient's magnitude and its angle. Then we applied thresholding to get the binary image. I also concluded that a high threshold gives less edges than a lower threshold.