**Name**: Soundarya

**Reg_number**: 2048057

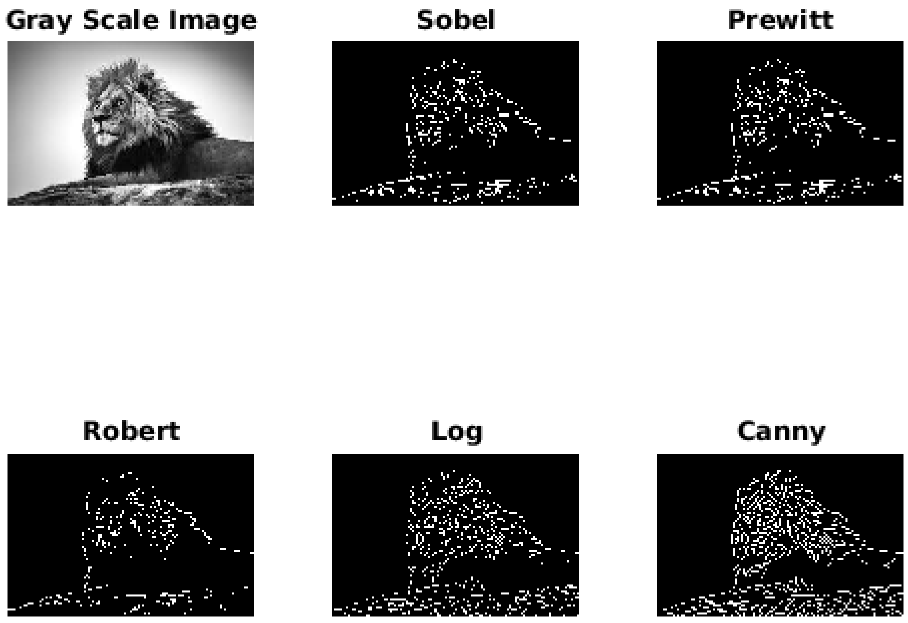
# Write a program to demonstrate edge detection.

## Note:

- Robert, Prewit, Sobel
- Horizontal, Vertical (X and Y axis direction masks)
- Different Kernel size (Smaller and larger masks)
- Different threshold.

## DIFFERENT METHODS OF EDGE DETECTION

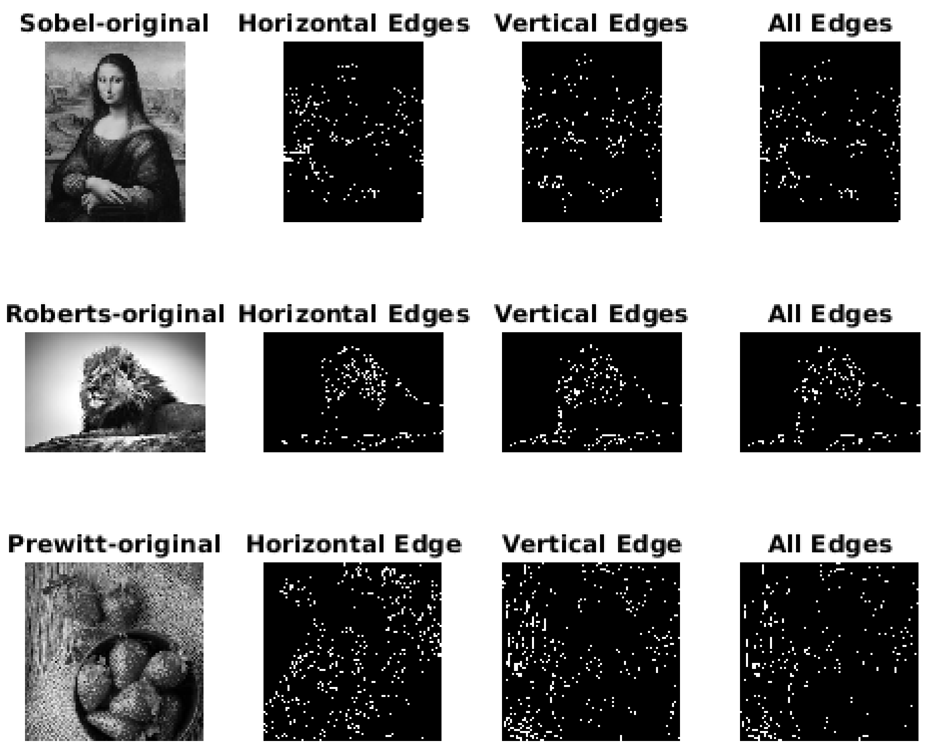
```
% importing the imgae
I = rgb2gray(imread("lion.jpg"));
subplot(2, 3, 1), imshow(I); title("Gray Scale Image");

% Sobel Edge Detection
J = edge(I, 'Sobel'); subplot(2, 3, 2), imshow(J); title("Sobel");

% Prewitt Edge detection
K = edge(I, 'Prewitt'); subplot(2, 3, 3), imshow(K); title("Prewitt");

% Robert Edge Detection
L = edge(I, 'Roberts'); subplot(2, 3, 4), imshow(L); title("Robert");

% Log Edge Detection
M = edge(I, 'log'); subplot(2, 3, 5), imshow(M); title("Log");

% Canny Edge Detection
N = edge(I, 'Canny'); subplot(2, 3, 6), imshow(N); title("Canny");
```



## DIFFERENT DIRECTION MASKS

```
mycolourimage = imread('mona.jpg');
myimage = rgb2gray(mycolourimage);
subplot(3,4,1);imshow(myimage); title('Sobel-original');

% Apply Sobel Operator
% Display only the horizontal Edges

sobelhz = edge(myimage,'sobel','horizontal');
subplot(3,4,2);imshow(sobelhz,[]); title('Horizontal Edges');

% Apply Sobel Operator
% Display only the vertical Edges

sobelvrt = edge(myimage,'sobel','vertical');
subplot(3,4,3);imshow(sobelvrt,[]); title('Vertical Edges');


% Apply Sobel Operator
% Display both horizontal and vertical Edges
sobelvrthz = edge(myimage,'sobel','both');
subplot(3,4,4);imshow(sobelvrthz,[]); title('All Edges');

mycolourimage1 = imread('lion.jpg');
myimage1 = rgb2gray(mycolourimage1);
subplot(3,4,5);imshow(myimage1); title('Roberts-original');

% Apply Roberts Operator
% Display both horizontal Edges

robertshz = edge(myimage1,'roberts','horizontal');
```

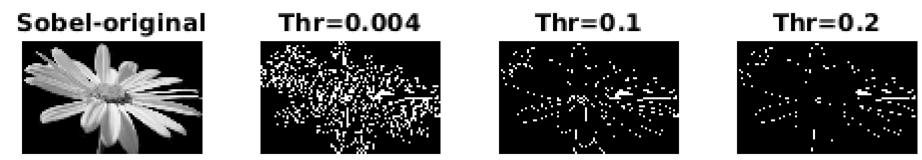
```
subplot(3,4,6);imshow(robertshz,[]); title('Horizontal Edges');

% Apply Roberts Operator
% Display both vertical Edges

robertsvr = edge(myimage1,'roberts','vertical');
subplot(3,4,7);imshow(robertsvr,[]); title('Vertical Edges');

% Apply Roberts Operator
% Display both horizontal and vertical Edges
robertvrthz = edge(myimage1,'roberts','both');
subplot(3,4,8);imshow(robertvrthz,[]); title('All Edges');

mycolourimage2 = imread('strawberry.jpg');
myimage2 = rgb2gray(mycolourimage2);
subplot(3,4,9);imshow(myimage2); title('Prewitt-original');

% Apply Prewitt Operator
% Display both horizontal Edges

prewitthr = edge(myimage2,'prewitt','horizontal');
subplot(3,4,10);imshow(prewitthr,[]); title('Horizontal Edge');

% Apply Prewitt Operator
% Display both vertical Edges

prewittvr = edge(myimage2,'prewitt','vertical');
subplot(3,4,11);imshow(prewittvr,[]); title('Vertical Edge');

% Apply Prewitt Operator
% Display both horizontal and vertical Edges

prewittvrthr = edge(myimage2,'prewitt','both');
subplot(3,4,12);imshow(prewittvrthr,[]); title('All Edges');
```

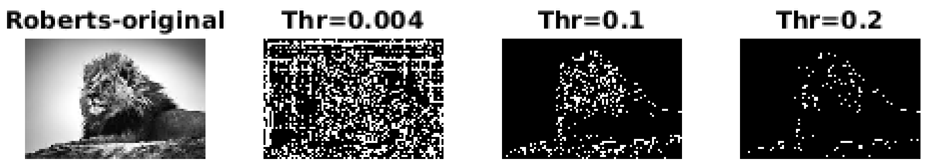


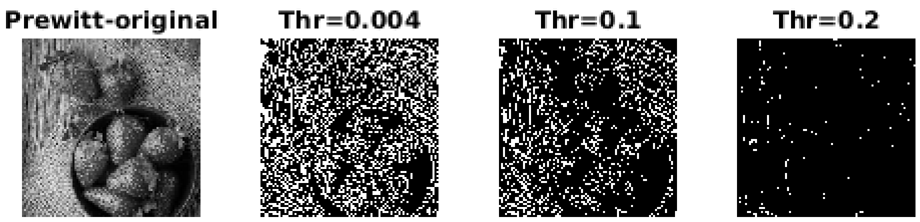
## DIFFERENT THRESHOLD VALUES

```
mycolourimage = imread('flower.jpeg');
myimage = rgb2gray(mycolourimage);
subplot(3,4,1);imshow(myimage); title('Sobel-original');

% Apply Sobel Operator
% Display only the horizontal Edges

sobel1 = edge(myimage,'sobel',0.004);
subplot(3,4,2);imshow(sobel1,[]); title('Thr=0.004');

% Apply Sobel Operator
% Display only the vertical Edges

sobel2 = edge(myimage,'sobel',0.1);
subplot(3,4,3);imshow(sobel2,[]); title('Thr=0.1');


% Apply Sobel Operator
% Display both horizontal and vertical Edges
sobel3 = edge(myimage,'sobel',0.2);
subplot(3,4,4);imshow(sobel3,[]); title('Thr=0.2');

mycolourimage1 = imread('lion.jpg');
myimage1 = rgb2gray(mycolourimage1);
subplot(3,4,5);imshow(myimage1); title('Roberts-original');

% Apply Roberts Operator
% Display both horizontal Edges

roberts1 = edge(myimage1,'roberts',0.004);
```

```
subplot(3,4,6);imshow(roberts1,[]); title('Thr=0.004');

% Apply Roberts Operator
% Display both vertical Edges

roberts2 = edge(myimage1,'roberts',0.1);
subplot(3,4,7);imshow(roberts2,[]); title('Thr=0.1');

% Apply Roberts Operator
% Display both horizontal and vertical Edges
robert3 = edge(myimage1,'roberts',0.2);
subplot(3,4,8);imshow(robert3,[]); title('Thr=0.2');

mycolourimage2 = imread('strawberry.jpg');
myimage2 = rgb2gray(mycolourimage2);
subplot(3,4,9);imshow(myimage2); title('Prewitt-original');

% Apply Prewitt Operator
% Display both horizontal Edges

prewitt1 = edge(myimage2,'prewitt',0.04);
subplot(3,4,10);imshow(prewitt1,[]); title('Thr=0.004');

% Apply Prewitt Operator
% Display both vertical Edges

prewitt2 = edge(myimage2,'prewitt',0.1);
subplot(3,4,11);imshow(prewitt2,[]); title('Thr=0.1');

% Apply Prewitt Operator
% Display both horizontal and vertical Edges

prewitt3 = edge(myimage2,'prewitt',0.2);
subplot(3,4,12);imshow(prewitt3,[]); title('Thr=0.2');
```



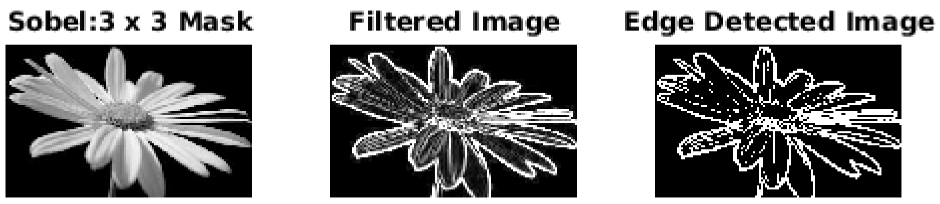


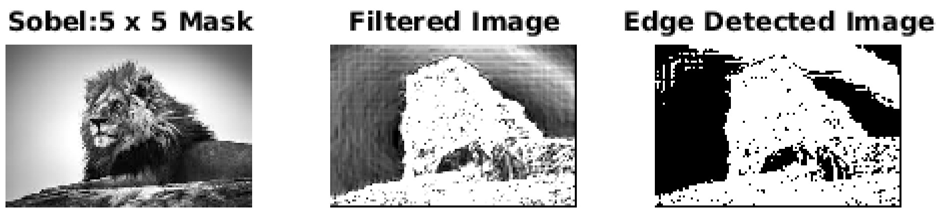


## DIFFERENT KERNEL SIZES

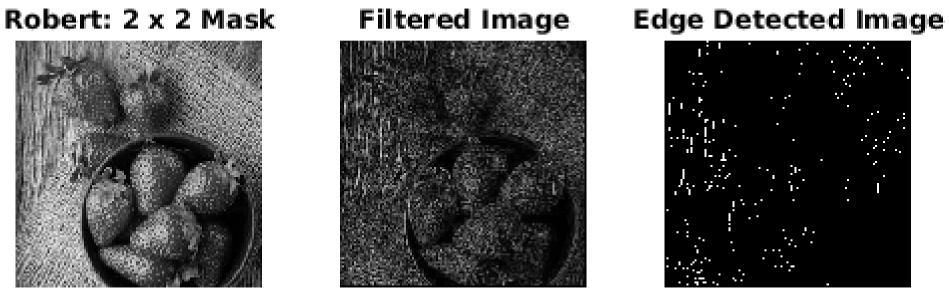
```
%  Sobel for 3 x 3 mask

% Read Input Image
input_image = imread('flower.jpeg');
input_image = uint8(input_image);
input_image = rgb2gray(input_image);
subplot(2,3,1);imshow(input_image); title('Sobel:3 x 3 Mask');

input_image = double(input_image);

% Pre-allocate the filtered_image matrix with zeros
filtered_image = zeros(size(input_image));

% Sobel Operator Mask
Mx = [-1 0 1; -2 0 2; -1 0 1];
My = [-1 -2 -1; 0 0 0; 1 2 1];

% Edge Detection Process
% When i = 1 and j = 1, then filtered_image pixel
% position will be filtered_image(2, 2)
% The mask is of 3x3, so we need to traverse
% to filtered_image(size(input_image, 1) - 2
%, size(input_image, 2) - 2)
% Thus we are not considering the borders.
for i = 1:size(input_image, 1) - 2
        for j = 1:size(input_image, 2) - 2

                % Gradient approximations
                Gx = sum(sum(Mx.*input_image(i:i+2, j:j+2)));
                Gy = sum(sum(My.*input_image(i:i+2, j:j+2)));
                % Calculate magnitude of vector
                filtered_image(i+1, j+1) = sqrt(Gx.^2 + Gy.^2);

        end
end

% Displaying Filtered Image
filtered_image = uint8(filtered_image);
subplot(2,3,2);imshow(filtered_image); title('Filtered Image');

% Define a threshold value
thresholdValue = 100; % varies between [0 255]
output_image = max(filtered_image, thresholdValue);
output_image(output_image == round(thresholdValue)) = 0;

% Displaying Output Image
output_image = im2bw(output_image);
subplot(2,3,3);imshow(output_image); title('Edge Detected Image');

%  Sobel for 5 x 5 mask
% % Read Input Image
input_image1 = imread('lion.jpg');
input_image1 = uint8(input_image1);
input_image1 = rgb2gray(input_image1);
subplot(2,3,4);imshow(input_image1); title('Sobel:5 x 5 Mask');

input_image1 = double(input_image1);

% Pre-allocate the filtered_image matrix with zeros
filtered_image1 = zeros(size(input_image1));

% Sobel Operator Mask
Mx = [2 1 0 -1 -2;2 1 0 -1 -2;4 2 0 -2 -4;2 1 0 -1 -2;2 1 0 -1 -2];
My = [2 2 4 2 2;1 1 2 1 1;0 0 0 0 0;-1 -1 -2 -1 -1;-2 -2 -4 -2 -2];

% Edge Detection Process
% When i = 1 and j = 1, then filtered_image pixel
% position will be filtered_image(2, 2)
% The mask is of 3x3, so we need to traverse
% to filtered_image(size(input_image, 1) - 2
%, size(input_image, 2) - 2)
% Thus we are not considering the borders.
for i = 1:size(input_image1, 1) - 4
        for j = 1:size(input_image1, 2) - 4

                % Gradient approximations
                Gx = sum(sum(Mx.*input_image1(i:i+4, j:j+4)));
                Gy = sum(sum(My.*input_image1(i:i+4, j:j+4)));

                % Calculate magnitude of vector
                filtered_image1(i+1, j+1) = sqrt(Gx.^2 + Gy.^2);

        end
end

% Displaying Filtered Image
filtered_image1 = uint8(filtered_image1);
```

```
subplot(2,3,5);imshow(filtered_image1); title('Filtered Image');

% Define a threshold value
thresholdValue1 = 100; % varies between [0 255]
output_image1 = max(filtered_image1, thresholdValue1);
output_image1(output_image1 == round(thresholdValue1)) = 0;

% Displaying Output Image
output_image1 = im2bw(output_image1);
subplot(2,3,6);imshow(output_image1); title('Edge Detected Image');
```





```
% Robert for 2 x 2 mask

% Read Input Image
input_image2 = imread('strawberry.jpg');
input_image2 = uint8(input_image2);
input_image2 = rgb2gray(input_image2);
subplot(1,3,1);imshow(input_image2); title('Robert: 2 x 2 Mask');

input_image2 = double(input_image2);

% Pre-allocate the filtered_image matrix with zeros
filtered_image2 = zeros(size(input_image2));

% Robert Operator Mask
Mx = [1 0; 0 -1];
My = [0 1; -1 0];

% Edge Detection Process
% When i = 1 and j = 1, then filtered_image pixel
% position will be filtered_image(1, 1)
% The mask is of 2x2, so we need to traverse
% to filtered_image(size(input_image, 1) - 1
%, size(input_image, 2) - 1)
for i = 1:size(input_image2, 1) - 1
    for j = 1:size(input_image2, 2) - 1

        % Gradient approximations
        Gx = sum(sum(Mx.*input_image2(i:i+1, j:j+1)));
        Gy = sum(sum(My.*input_image2(i:i+1, j:j+1)));

        % Calculate magnitude of vector
        filtered_image2(i, j) = sqrt(Gx.^2 + Gy.^2);

    end
end

% Displaying Filtered Image
filtered_image2 = uint8(filtered_image2);
subplot(1,3,2);imshow(filtered_image2); title('Filtered Image');

% Define a threshold value
thresholdValue = 100; % varies between [0 255]
output_image2 = max(filtered_image2, thresholdValue);
output_image2(output_image2 == round(thresholdValue)) = 0;

% Displaying Output Image
output_image2 = im2bw(output_image2);
subplot(1,3,3);imshow(output_image2); title('Edge Detected Image');
```



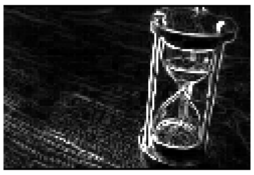
```
% Prewitt for 3 X 3 mask

% Read Input Image
input_image3 = imread('clock.jpg');
```

```
input_image3 = uint8(input_image3);
input_image3 = rgb2gray(input_image3);
subplot(2,3,1);imshow(input_image3); title('Prewitt: 3 x 3 Mask');

input_image3 = double(input_image3);

% Pre-allocate the filtered_image matrix with zeros
filtered_image3 = zeros(size(input_image3));

% Prewitt Operator Mask
Mx = [-1 0 1; -1 0 1; -1 0 1];
My = [-1 -1 -1; 0 0 0; 1 1 1];

% Edge Detection Process
% When i = 1 and j = 1, then filtered_image pixel
% position will be filtered_image(2, 2)
% The mask is of 3x3, so we need to traverse
% to filtered_image(size(input_image, 1) - 2
%, size(input_image, 2) - 2)
% Thus we are not considering the borders.
for i = 1:size(input_image3, 1) - 2
        for j = 1:size(input_image3, 2) - 2

                % Gradient approximations
                Gx = sum(sum(Mx.*input_image3(i:i+2, j:j+2)));
                Gy = sum(sum(My.*input_image3(i:i+2, j:j+2)));

                % Calculate magnitude of vector
                filtered_image3(i+1, j+1) = sqrt(Gx.^2 + Gy.^2);

        end
end

% Displaying Filtered Image
filtered_image3 = uint8(filtered_image3);
subplot(2,3,2); imshow(filtered_image3); title('Filtered Image');

% Define a threshold value
thresholdValue = 100; % varies between [0 255]
output_image3= max(filtered_image3, thresholdValue);
output_image3(output_image3 == round(thresholdValue)) = 0;

% Displaying Output Image
output_image3 = im2bw(output_image3);
subplot(2,3,3); imshow(output_image3); title('Edge Detected Image');

% Prewitt for 5 X 5 mask

% Read Input Image
input_image4 = imread('mona.jpg');
input_image4 = uint8(input_image4);
input_image4 = rgb2gray(input_image4);
subplot(2,3,4);imshow(input_image4); title('Prewitt: 5 x 5 Mask');

input_image4 = double(input_image4);

% Pre-allocate the filtered_image matrix with zeros
filtered_image4 = zeros(size(input_image4));
```

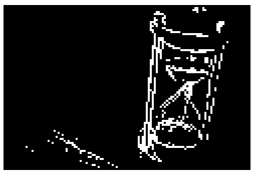
```matlab
% Prewitt Operator Mask

Mx = [2 2 2 2 2;1 1 1 1 1;0 0 0 0 0;-1 -1 -1 -1 -1;-2 -2 -2 -2 -2];
My = [-2 -1 0 2 1;-2 -1 0 2 1;-2 -1 0 2 1;-2 -1 0 2 1;-2 -1 0 2 1];

% Edge Detection Process
% When i = 1 and j = 1, then filtered_image pixel
% position will be filtered_image(2, 2)
% The mask is of 5x5, so we need to traverse
% to filtered_image(size(input_image, 1) - 2
%, size(input_image, 2) - 2)
% Thus we are not considering the borders.
for i = 1:size(input_image4, 1) - 4
        for j = 1:size(input_image4, 2) - 4

                % Gradient approximations
                Gx = sum(sum(Mx.*input_image4(i:i+4, j:j+4)));
                Gy = sum(sum(My.*input_image4(i:i+4, j:j+4)));

                % Calculate magnitude of vector
                filtered_image4(i+1, j+1) = sqrt(Gx.^2 + Gy.^2);

        end
end

% Displaying Filtered Image
filtered_image4 = uint8(filtered_image4);
subplot(2,3,5); imshow(filtered_image4); title('Filtered Image');

% Define a threshold value
thresholdValue = 100; % varies between [0 255]
output_image4= max(filtered_image4, thresholdValue);
output_image4(output_image4 == round(thresholdValue)) = 0;

% Displaying Output Image
output_image4 = im2bw(output_image4);
subplot(2,3,6); imshow(output_image4); title('Edge Detected Image');
```

**Prewitt: 3 x 3 Mask**

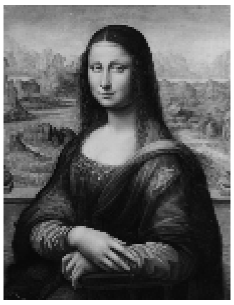
**Filtered Image**



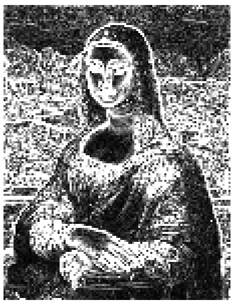
**Edge Detected Image**



**Prewitt: 5 x 5 Mask**



**Filtered Image**



**Edge Detected Image**