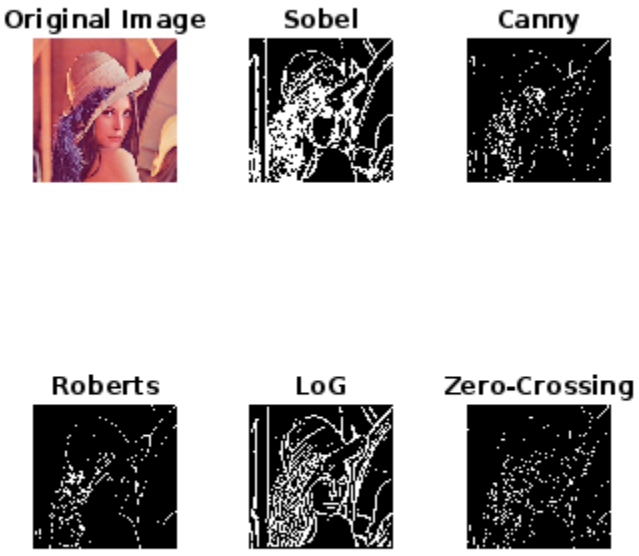
```
% Operators without functions
%Created by Jyotiraditya Bhos

clear all;
close all;
clc;

% Read an image from the URL and store it
I = imread('https://upload.wikimedia.org/wikipedia/en/thumb/7/7d/
Lenna_%28test_image%29.png/330px-Lenna_%28test_image%29.png');

% Convert the original image to grayscale for edge detection
I_gray = rgb2gray(I);

% Define the Sobel filter for horizontal and vertical gradients
sobel_x = [-1 0 1; -2 0 2; -1 0 1];
sobel_y = [-1 -2 -1; 0 0 0; 1 2 1];

% Apply Sobel filter to compute gradients in x and y directions
gx = imfilter(double(I_gray), sobel_x, 'same');
gy = imfilter(double(I_gray), sobel_y, 'same');

% Calculate the gradient magnitude and threshold to produce an edge map
BW1 = sqrt(gx.^2 + gy.^2) > 50;

% Define the Prewitt filter for horizontal and vertical gradients
prewitt_x = [-1 0 1; -1 0 1; -1 0 1];
prewitt_y = [-1 -1 -1; 0 0 0; 1 1 1];

% Apply Prewitt filter to compute gradients in x and y directions
gx = imfilter(double(I_gray), prewitt_x, 'same');
gy = imfilter(double(I_gray), prewitt_y, 'same');

% Calculate the gradient magnitude and threshold to produce an edge map
BW3 = sqrt(gx.^2 + gy.^2) > 50;

% Define the Roberts filter for horizontal and vertical gradients
roberts_x = [1 0; 0 -1];
roberts_y = [0 1; -1 0];

% Apply Roberts filter to compute gradients in x and y directions
gx = imfilter(double(I_gray), roberts_x, 'same');
gy = imfilter(double(I_gray), roberts_y, 'same');

% Calculate the gradient magnitude and threshold to produce an edge map
BW4 = sqrt(gx.^2 + gy.^2) > 50;

% Create a Laplacian of Gaussian (LoG) filter
sigma = 2;
size = 6*sigma;
x = -size/2:size/2;
y = x;
```

```
[X, Y] = meshgrid(x, y);
LoG = (X.^2 + Y.^2 - 2*sigma^2) .* exp(-(X.^2 + Y.^2) / (2*sigma^2));

% Apply the LoG filter to detect edges
BW5 = imfilter(double(I_gray), LoG, 'same') > 0;

% Use MATLAB's built-in edge detection for Zero-Crossing
BW6 = edge(I_gray, 'zerocross');

% Compute the gradient magnitude for Canny edge detection
gx = imfilter(double(I_gray), sobel_x, 'same');
gy = imfilter(double(I_gray), sobel_y, 'same');
gmag = sqrt(gx.^2 + gy.^2);

% Threshold and process edges for a Canny-like result
BW2 = gmag > 0.1 * max(gmag(:));
BW2 = bwmorph(BW2, 'thin', Inf);  % Thin edges
BW2 = bwareaopen(BW2, 30);        % Remove small objects

% Use tiled layout to display the original image and all edge detection
results
tiledlayout(2,4);
nexttile; imshow(I);
title('Original Image');
nexttile; imshow(BW1);
title('Sobel');
nexttile; imshow(BW2);
title('Canny');
nexttile; imshow(BW3);
title('Prewitt');
nexttile; imshow(BW4);
title('Roberts');
nexttile; imshow(BW5);
title('LoG');
nexttile; imshow(BW6);
title('Zero-Crossing');
```



Published with MATLAB® R2024b