EE 417 - Computer Vision
Post Lab Report 3

Edge Detection

Name: Çiğdem Ceyda Düzgeç

Student ID: 23928

Date: 19.11.2020

Çiğdem Ceyda Düzgeç
23928

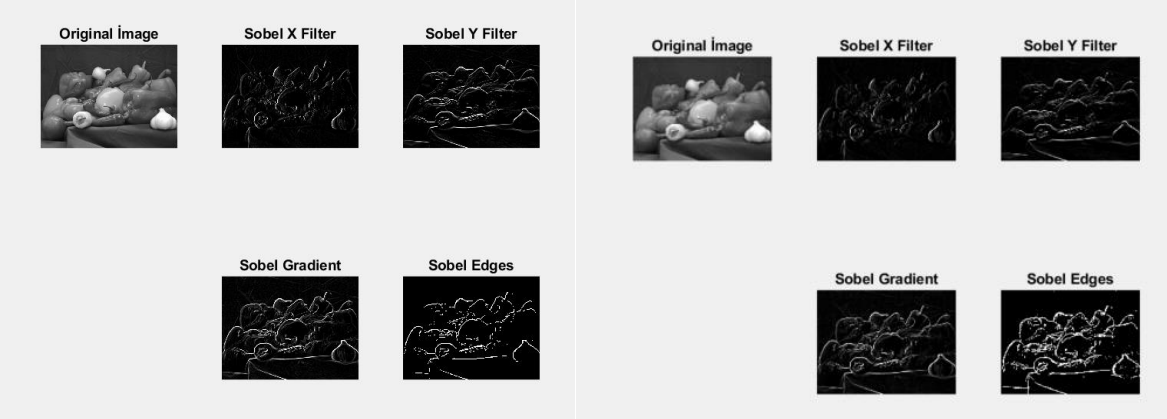
Introduction About Edge Detection:

Edge detection is a technique for finding the boundaries of objects within images by detecting the discontinuities in intensity between pixels. Also, it can be used for extracting important features such as corners lines and curves that can be used in higher-level computer vision algorithms such as recognition. Detection of discontinuities is done by using derivatives because the difference is large at these areas.

1. Sobel Operator: lab3sobel.m

Sobel operator is a type of edge detection which uses 2D derivative operation. It has two masks with 3x3 pixels as Gx and Gy which are [-1 0 1; -2 0 2 ; -1 0 1] and [-1 -2 -1; 0 0 0 ; 1 2 1] respectively. These masks can be applied together or separately. Also, they can be applied from left to right, right to left, top to bottom or bottom to top by simply changing the direction of kernel with – sign. For Gx it uses first and third column to multiply these values with actual pixels to sum up a total value which approximates the magnitude of the gradient and see if this reaches the threshold or not. If it exceeds threshold the pixels are changed into white. As for Gy it is pretty much the same except it uses rows. Sobel gives more weight to the center that is why we have 2 on the center of rows and columns.

We first checked our image and turned it into grey scale image and into double so we can use it to make calculations. We used our Sobel filter from lab 2 to find Gx and Gy filters. Then since we want to use them both and do not want to get negative values, we used L2 norm to get our gradient image. Then we created a new matrix so we can put our processed pixels here. Lastly, we find every pixel which is greater than our threshold and change them into white. The smaller threshold gives us whiter images because it detects smaller details.



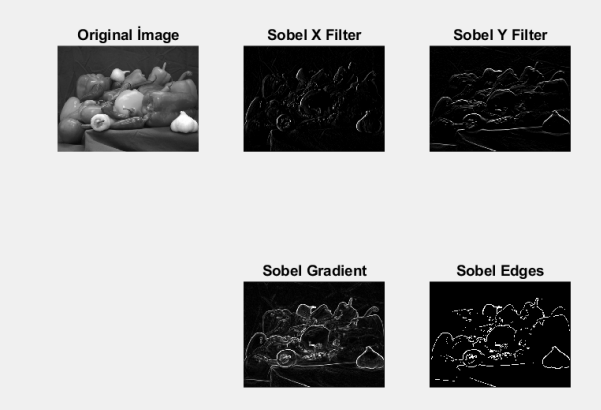
Threshold: 100 Threshold: 50

Çiğdem Ceyda Düzgeç
23928

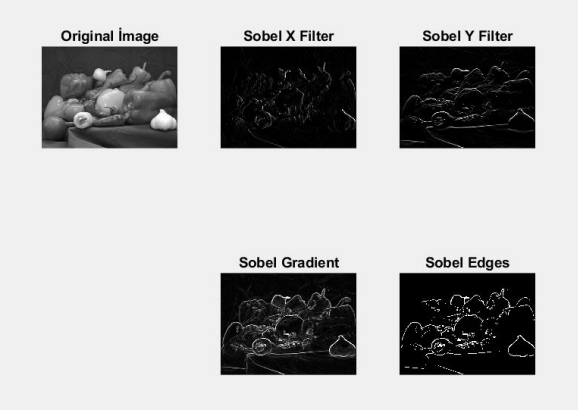
2. Prewitt Operator: lab3prewitt.m

Prewitt operator is also a discrete 2D derivative operation. It is pretty much the same with Sobel operator. except that Sobel gives more weight to the center pixels whereas Prewitt does not differentiate closer pixels to the center. That is why it's Gx and Gy filters are [-1 0 1;-1 0 1 ; -1 0 1] and [-1 -1 -1; 0 0 0; 1 1 1] respectively.

In our code we did similar steps to Sobel except we used different kernels. We can also apply Prewitt in different directions by changing the sign of the kernel.



Kernel with negative x filter



Kernels with positive filters

Çiğdem Ceyda Düzgeç
23928

3. Laplacian of Gaussian: lab3log.m

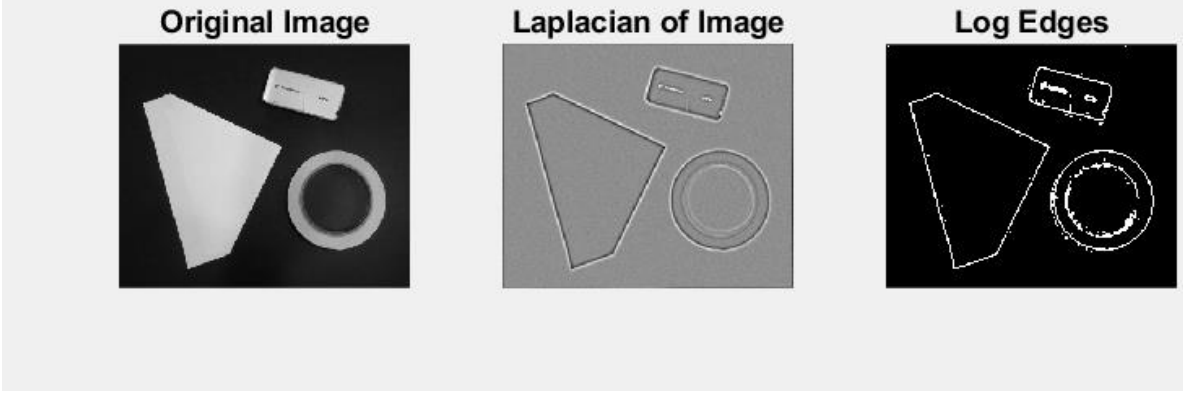
Laplacian filter is a second derivative filter which uses zero crossing logic by checking the sign change between pixels and if it can find a change that means there should be a zero between these by using a threshold. It is cheaper to implement than the gradient because it is one mask, but gradient is two masks. Also, it gives equal weights in all directions and it can no provide information about edge direction.

It is sensitive to noise in the image such as salt and pepper. So, a threshold should be used in order to filter the noise. Before applying this filter, the image should be smoothened by a linear filter such as Box filter or Gaussian filter. Instead of applying the Gaussian and Laplacian separately it can be done in one step of convolution with a kernel which is [0 1 0 ; 1 -4 1; 0 1 0]

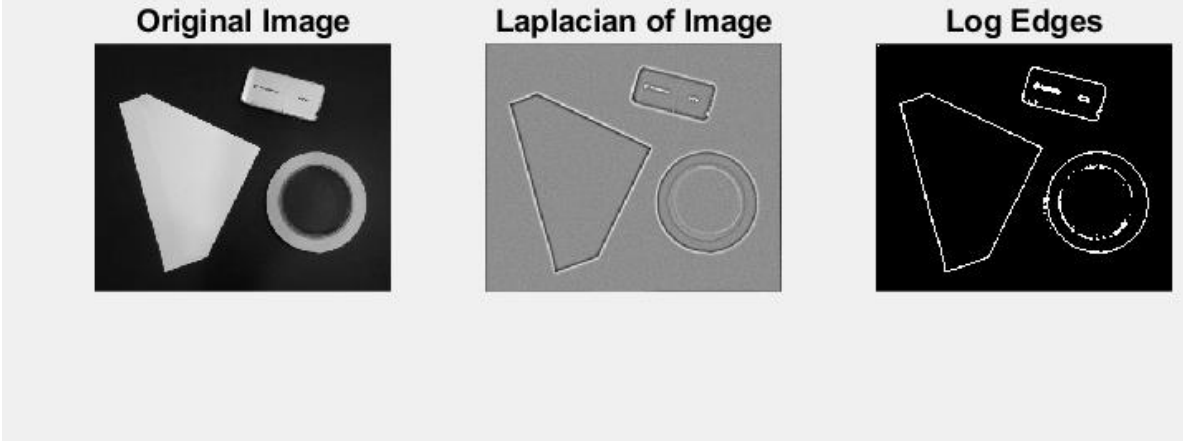
In our code, we first turned it into grey scale then we used implemented Gaussian function of Matlab Toolkit on greyscale image to get our Gaussian. Then we created kernel which is [0 1 0 ; 1 -4 1; 0 1 0] and use both of them to do convolution. Since we are not doing any calculation, we did not convert these into double.
-First, we checked if the pixel which is located on the center exceeds threshold.
- If it was, we also checked the sign change between neighbors by multiplying them. Because if they have different signs the result should be smaller than 0 and this is the most efficient way.
-And if there was a change, we also check if it exceeds slope threshold. We can divide the difference by 2 or not. But we should take this into consideration when giving a value to threshold. If all conditions are met, we declared this pixel as an edge.
- If there was a sign change but the slope is not big enough, we do not declare it as an edge. It should satisfy both conditions with both thresholds.
-But if the center pixel does not satisfy the conditions in else loop, we still checked the neighbors

At the end we used Matlab Toolkit medfilt function to get a better image.

Çiğdem Ceyda Düzgeç
23928

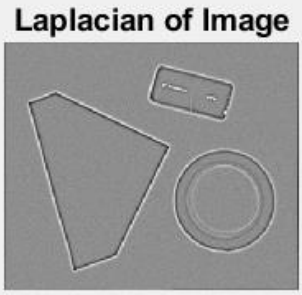


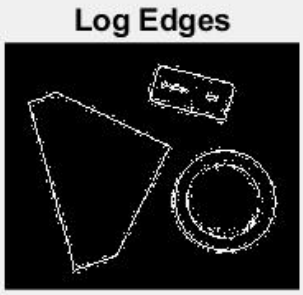
Threshold 1 = 1 Threshold = 7 with medfilt with scaling



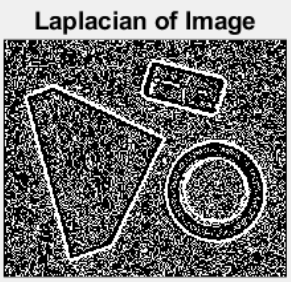
Threshold 1 = 5 Threshold = 10 with medfilt with scaling

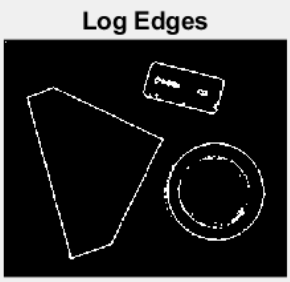
Çiğdem Ceyda Düzgeç
23928





Threshold 1 = 5 Threshold = 10 without medfilt with scaling





Threshold 1 = 5 Threshold = 10 with medfilt without scaling

Çiğdem Ceyda Düzgeç
23928

Appendix

A. lab3.main.m

```
%% Edge Detection
clear all; close all; clc;

%% Sobel
I = imread('peppers.png');
Thres = 100;
Im_sobel = lab3sobel(I,Thres);

%% Prewitt
Tresh2 = 100;
Im_perwitt = lab3prewitt(I,Tresh2);

%% Laplacian

I2 = imread('Object_contours.jpg');
T1 = 1;
T2 = 7;
G = lab3log(I2,T1,T2);
```

B. lab3sobel.m

```
function [I_edge] = lab3sobel(I,T)
    [r,c,ch] = size(I);
    if ch == 3
        Im = rgb2gray(I);
    end

 Im = double(Im);
 % Calculate gradient magnitude and threshold
 [Gx,Gy] = lab2sobelfilt(Im);
 % Find magnitude of gradient
 % Threshold Hint (find)
 Gx = double(Gx);
 Gy = double(Gy);

 G_mag = sqrt(Gx.^2 + Gy.^2);



 I_edge= zeros(size(G_mag));
 I_edge(find(G_mag > T)) = 255; % find number of every pixel
which is greater than threshold to make them white

 figure
 subplot(2,3,1), imshow(uint8(Im))
 title ('Original ?mage')
 subplot(2,3,2), imshow(uint8(Gx))
 title ('Sobel X Filter')
 subplot(2,3,3), imshow(uint8(Gy))
 title ('Sobel Y Filter')
 subplot(2,3,5), imshow(uint8(G_mag))
 title ('Sobel Gradient')
 subplot(2,3,6), imshow(uint8(I_edge))
 title ('Sobel Edges')

 % if you flip the sign of the kernel - to + you can change the
direction



end
```

C. lab3prewitt.m

```
function [I_edge] = lab3prewitt(I,T)
    [r,c,ch] = size(I);
    if ch == 3
        I = rgb2gray(I);
    end

    I = double(I);
    Sx = [-1 0 1;-1 0 1 ; -1 0 1 ];    % multiply by -1
direction will change
    Sy = [-1 -1 -1; 0 0 0; 1 1 1];     % kernels sign
determines the detection direction

    k = 1;
    Gx = zeros(size(I))
    Gy = zeros(size(I))

 %Convolution

    for i = k+1:1:r-k-1
       for j = k+1:1:c-k-1
           window = I(i-k:i+k,j-k:j+k);
           Xvalue= sum(sum(window.*Sx));
           Yvalue= sum(sum(window.*Sy));
           Gx(i,j) = Xvalue;
           Gy(i,j) = Yvalue;
        end
    end

    G_mag = sqrt(Gx.^2 + Gy.^2);
    I_edge = zeros(size(G_mag));
    I_edge(find(G_mag > T)) = 255;

     figure
     subplot(2,3,1), imshow(uint8(I))
     title ('Original ?mage')
     subplot(2,3,2), imshow(uint8(Gx))
     title ('Sobel X Filter')
     subplot(2,3,3), imshow(uint8(Gy))
     title ('Sobel Y Filter')
     subplot(2,3,5), imshow(uint8(G_mag))
     title ('Sobel Gradient')
     subplot(2,3,6), imshow(uint8(I_edge))
     title ('Sobel Edges')

end
```

Çiğdem Ceyda Düzgeç
23928

D. lab3log.m

```
function [E] = lab3log(I,T1,T2)

    [r,c,ch] =size(I);
    if ch ==3
        I = rgb2gray(I);
    end

    J = imgaussfilt(I,1.1);

    W = [0 1 0 ; 1 -4 1; 0 1 0];
    G = conv2(J,W,'same');

    E = zeros(r,c);

    % we dont need double because we are not doing any
calculations
    k=1;  % 3x3 sliding window
    for i = k+1:1:r-k-1
        for j= k+1:1:c-k-1

            if abs(G(i,j)) >= T1
                if (G(i-1,j)*G(i,j)) < 0 || (G(i+1,j)*G(i,j)) <
0 || (G(i,j-1)*G(i,j)) < 0 || (G(i,j+1)*G(i,j)) < 0 % sign
change between right left up down neighbors
                    if abs(G(i-1,j)-G(i,j)) >= T2 ||
abs(G(i+1,j)*G(i,j)) >= T2 || abs(G(i,j-1)*G(i,j)) >= T2||
abs(G(i,j+1)*G(i,j)) >= T2
                        E(i,j) = 255; % declare edge
                    end
                end

            else
                 if (G(i-1,j)*G(i+1,j)) < 0 || (G(i,j-
1)*G(i,j+1)) < 0 % sign change between right left up down
neighbors
                    if abs(G(i,j+1)-G(i,j-1)) >= T2 || abs(G(i-
1,j)*G(i+1,j)) >= T2 % check slope
                        E(i,j) = 255; % declare edge
                    end
                end
            end
        end
    end
```

```
    figure
    subplot(1,3,1), imshow(uint8(I))
    title('Original Image')
    subplot(1,3,2), imshow(G,[])
    title('Laplacian of Image')
    E = medfilt2(E);
    subplot(1,3,3), imshow(E)
    title('Log Edges')


end
```