Assignment #2 Report

pciftcioglu@sabanciuniv.edu

November 2020

1 Introduction

This paper consists of experiments related to edge detection with different filters and their comparison. Edge detection is in simple words is the process of identifying the objects in the images and illustrate their borders to utilize an information about their shapes. Edge detection can be done with the use of gradients since its main goal is to detect discontinuities (sudden changes) of different patterns in pixel values. Since in computer we don't have the knowledge of what the objects are corresponding to in the reality, the main goal is to get as close as we can to real life object's borders with mapping the edges and producing a sketch-like object forms from the images.

2 Filters

2.1 Prewitt Edge Detector

Prewitt filter is a first derivative edge detector which first smoohtens the image to reduce the possible noises present and calculates the gradients for both of the directions x and y. It uses smoothing vectors with derivative masks -different for each direction- that results in kernel filters and they have been used also to calculate the gradient magnitudes. Gradient magnitudes are used as pixel values to produce the new image and it also been applied a threshold method where it replaces the pixel values as 255 (white) that are greater than a specific threshold value (**T**).

To implement this filter I used the built-in function of MATLAB which is called as edge(I,'Prewitt) where **I** is the gray scale version of the original image. **T** is determined automatically with a suitable value for that certain image since I haven't inserted any input for the function. However, since the function also returns the threshold used, I will be inserting the value of it.

2.2 Roberts Edge Detector

Roberts edge detector is pretty similar to Prewitt edge detector in a sense that it also calculates the gradients in both directions and finds the gradient magnitude. However, Roberts uses different masks as filters for approximation of the derivative. I again used a built-in function for the method called as edge(I, "roberts") where **I** is the original image and according to MATLAB documentation "The Roberts method can detect edges at angles of 45° from horizontal, 135° from horizontal, or both" and since I didn't input an orientation of direction, they will be used as default with automatic assigned threshold value as well.

2.3 Canny Edge Detector

Canny Edge detector is another first order derivative filter where it calculates the gradient of the Gaussian which mostly aims to detect edges with also determining the qualities of it. Similar to the Prewitt filter does, it smoothens the image first and applies derivative filters to find the gradient of the pixels values of the original image. Additionally, it calculates the gradient magnitude and the direction of the edge which corresponds to the orientation of each pixel and uses it for the decision of whether the edge found should be suppressed or not. It uses one threshold value to determine whether the gradient in a pixel is an edge-pixel or not and another threshold value to determine whether it is a strong or a weak edge and it is expected to perform a better against the noise since it takes into account what the inclination of the neighboring pixels do.

I again used a built-in functiın called as edge(I,'canny') and let the threshold values assigned automatically while σ value for Gaussian filter is by default used as 2 where **I** is the gray scale version of the original image.

2.4 Laplacian of Gaussian Edge Detector

Laplacian of Gaussian is another filter for edge detection but as a difference than others, it uses second order derivative for its implementation and I used a built-in function called as edge('I', "log") where **I** is the original image in gray-scale format. First, it applies a Gaussian filter to smoothen the image and after that it finds the Laplacian which corresponds to calculating the sum of second derivates of the pixel values instead of first derivative. Additionally, instead of calculating the extremum (values) in the gradient magnitude, it finds the zero crossings of the second order derivatives (Laplacian) since the extremum values in the first derivative corresponds to the '0' values in the second derivative. Since I again didn't specify neither a threshold value or a sigma (as default, $\sigma = 2$), it assigns those automatically. It uses the threshold value to detect whether an edge is a strong or a weak one after it calculates the slopes of the zero-crossings.

3 Experiments

There are two different types of edge detection methods classified as 1st and 2nd derivative filters given in the assignment document and throughout the experiments I will be comparing their performances as well as indicating the best performing ones. For all the functions used, the original images are transformed to gray-scale as it mentioned in the previous section and the threshold values implemented by the algorithms have been recorded. To provide a better comparison I will use 4 different types of images where they all have different features in terms of complexity, quality and context.

3.1 Experiment:

Figure 1: Organized Table

For the first trial I wanted to use an image where the color changes are pretty distinctive and the shape of the objects are easy to recognize.

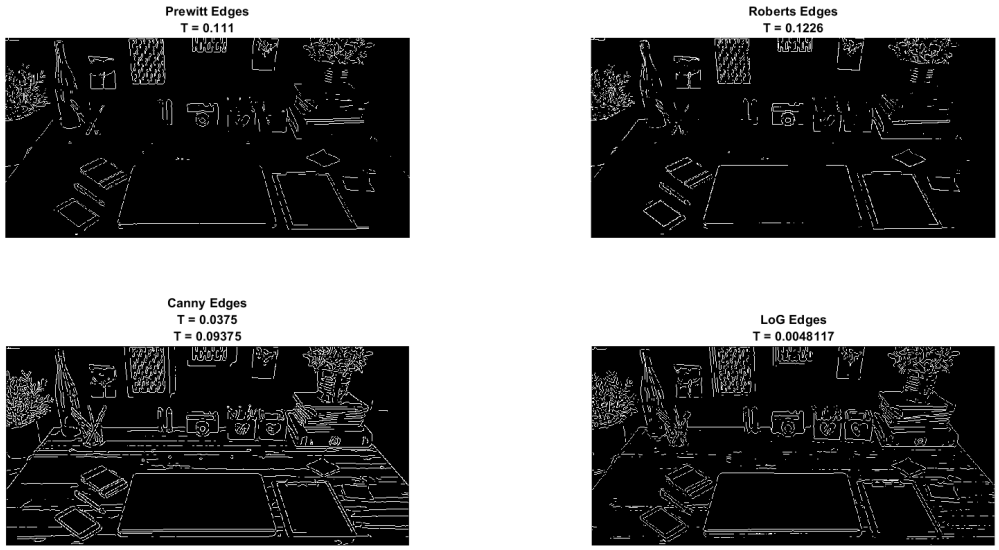

Figure 2: Organized Table Image Edges

As it can be seen from the 2, Prewitt and Roberts filters failed to identify all the borders with the given threshold values. However, canny filter and LoG edges which are the Gradient of Gaussian and Laplacian of Gaussian, achieved to detect all the lines even though they detected some extra lines on the table which are not actually object borders.

3.2 Experiment:

Figure 3: Beatles

For the second experiment, I used a bit more complex image to see whether they will be performing the same with a context with humans and relatively a bit more complex background.

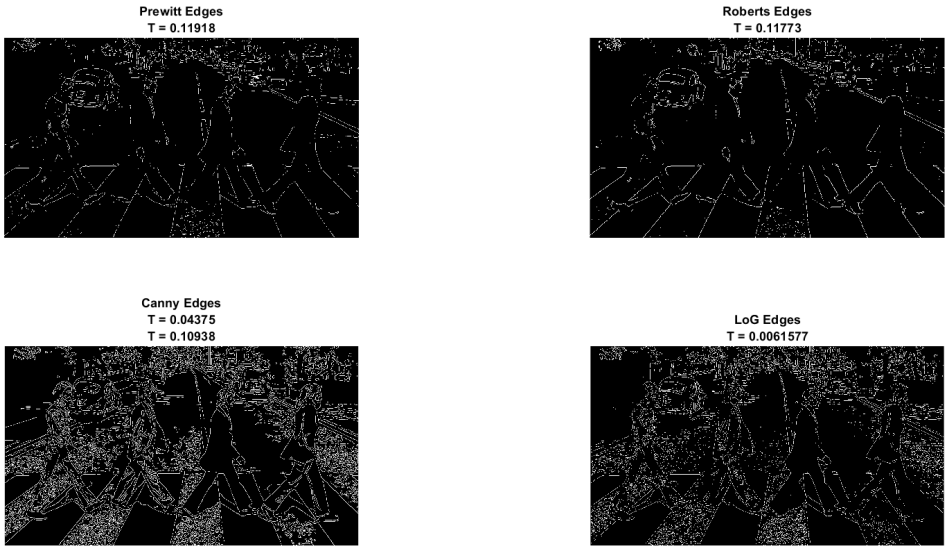

Figure 4: Beatles Image Edges

Looking at the results on the Figure 4, while Prewitt and Roberts filters failed to draw all the edges, Canny and LoG filters detected too many edges that it became hard to see where the object borders are. Depending on the context that the images will be used for processing, it is possible to comment on which one performed better. Since it is easier to understand the human-like shapes in the first two, it is possible to say that they performed better. But further trial with different threshold values might result in better drawn edges of objects.

3.3 Experiment:

Figure 5: Traffic

For this experiment, I wanted to use another image where the complexity level is relatively higher in terms of the objects to detect and draw their edges. And I wanted to test whether all the filters will succeed in detecting the lines to at least understand what is in the image from the resultant edgels.

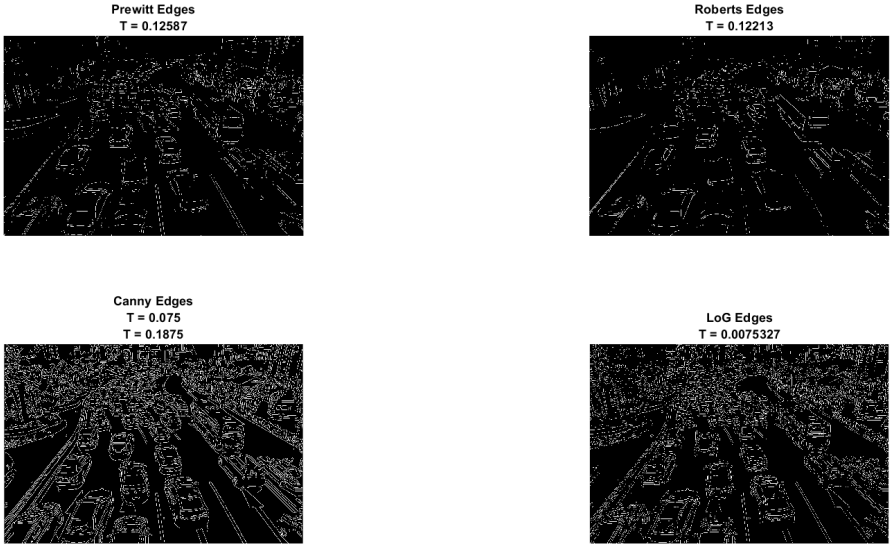

Figure 6: Traffic Image Edges

As it can be seen in the Figure 6, this time it is relatively easier to understand what is going on in the image and the shapes of the objects with all of the filters. However, it is still possible to indicate that the Canny and LoG filters detected more edges. Additionally, it might be important to point out that the threshold values assigned automatically for Prewitt and Roberts are nearly the same and also for Canny's first threshold value and LoG edges are very close to each other.

3.4 Experiment:

Figure 7: Sunset View

Another example that I used as can be seen in the Figure 7 had more smooth transitions between colors and the objects are not that clear to distinct. I wondered what the filters will detect as edges and whether the gradients will achieve to detect objects.

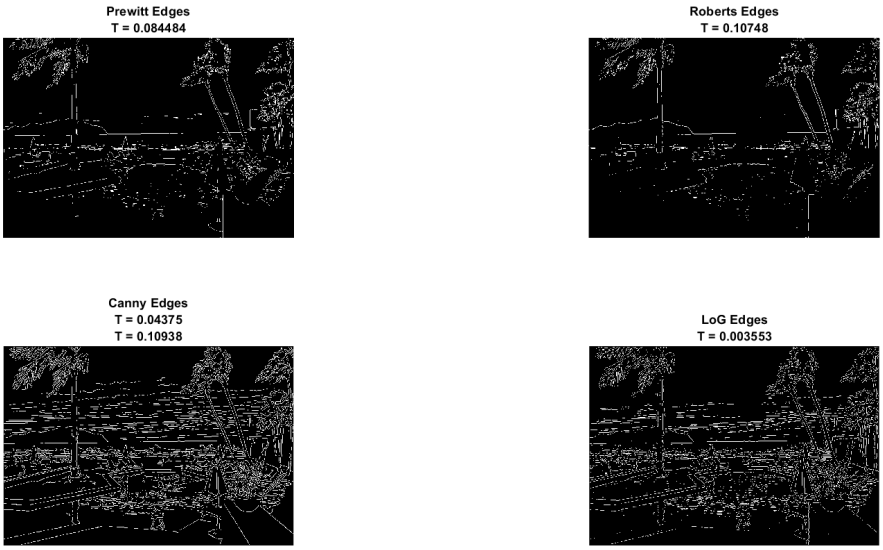

Figure 8: Sunset View Image Edges

As it can be seen from the Figure 8, even though Prewitt and Roberts edges failed to detect some of the borders in the image, it is still possible to say that they performed better since the other filters detected way more edges than it is desired. And this time, it is clear to observe that the threshold values are very distinctive than each other.

3.5 Experiment:

Figure 9: Birth of Venus

The last experiment that I wanted to do is about what the results will be if I use a painting with a complex context and structure rather than a real life image.

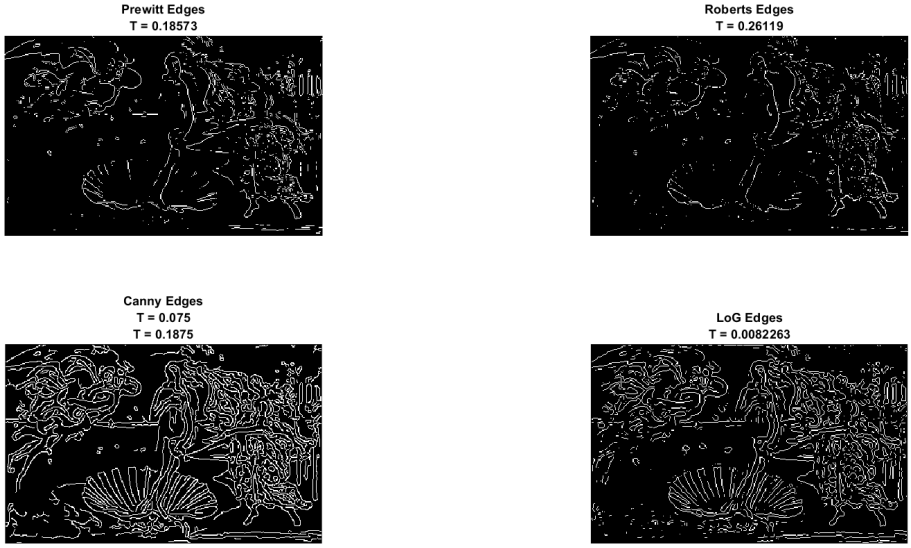

Figure 10: Birth of Venus Image Edges

From Figure 10, the resultant images were way better than I expected. Even though Prewitt and Roberts failed to detect all the edges present in the image, Canny and LoG filters actually resultant in a really nice borders of the object borders in the painting. And it is also possible to say that even though it is hard to understand what the objects are since the context is very abstract, Canny filters was really successful to identify the edgels with a slightly better performance than LoG edges.

4 Conclusion and Discussion

After trying all the filters with different types of images, it is possible to say that the best performing filter changes as the image complexity, context and the color distribution changes. And it is also important to select a filter based on what we want to observe in the resultant image. However, for most of the trials, it can be reasonable to say that Canny filter performed the best and it might be the reason that with the method it uses for determining whether an edge is strong or weak results in better performance with noises even after Gaussian smoothing and with any other possible distortion happened in the original image. Additionally, maybe it would be a better idea to use Prewitt and Roberts edges in the less complex images as I observed in the former experiments that it would be enough to detect the objects with the less detailed structures of the backgrounds and it might be the case that their simplistic approach provided those results. However, it would be still reasonable to try different threshold values or parameters to see whether there will be a better edge detection even with the filters that have been performed as the best and with which threshold values it would get close to optimum in the future.

5 Appendix

5.1 main.m

```
%% Read the image and turn it to grayscale

clear all; close all; clc;

Im = imread('venus.jpg');

I = rgb2gray(Im);

% I = double(I);

%% Prewitt Edge Detector
[I_Prewitt,threshOut_Prewitt] = edge(I, 'Prewitt');

%% Roberts Edge Detector
[I_Roberts,threshOut_Roberts] = edge(I, 'roberts');

%% Canny Edge Detector
[I_Canny, threshOut_Canny] = edge(I, 'canny');

%% Laplacian of Gaussian Edge Detector
[I_LoG, threshOut_LoG] = edge(I, 'log');

%% Visualization

figure
    imshow(uint8(I))
    title("Original Image")
figure
    subplot(2,2,1)
    imshow(I_Prewitt);
    title(["Prewitt Edges", "T = "+ threshOut_Prewitt]);

    subplot(2,2,2)
    imshow(I_Roberts);
    title(["Roberts Edges", "T = "+ threshOut_Roberts]);

    subplot(2,2,3)
    imshow(I_Canny)
    title(["Canny Edges", "T = "+ threshOut_Canny])

    subplot(2,2,4)
    imshow(I_LoG)
    title(["LoG Edges", "T = "+ threshOut_LoG])
```