Indian Institute of Science Education and Research Bhopal

Computer Vision(DSE/EECS-312)

Assignment-1: Answer Sheet

Name: Adheesh Trivedi
Roll No.: 22016
Time of submission: Marks Obtained:

Please follow the instructions given in the assignment carefully.

Please provide your detailed answers and any explanations or diagrams directly below each question in the 'Answers' section.

1. Apply the filters mentioned below on the image attached and analyze their impact. Describe what you found after applying each filter and why certain phenomena are happening. ***(Marks:)***

1.

-1	0	1
-1	0	1
-1	0	1

2.

1	1	1
0	0	0
-1	-1	-1

Since the above filters are of dimension 3×3. Construct the same filters of dimension 5×5 and do the above experiments.

Answer:

Prewitt filter (3 by 3)

I used the 3×3 Prewitt filter to produce image dipicting the edges in the original image:

$$\delta_x = \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

and,

$$\delta_y = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

These filters when convolved with the original image gives the following images:

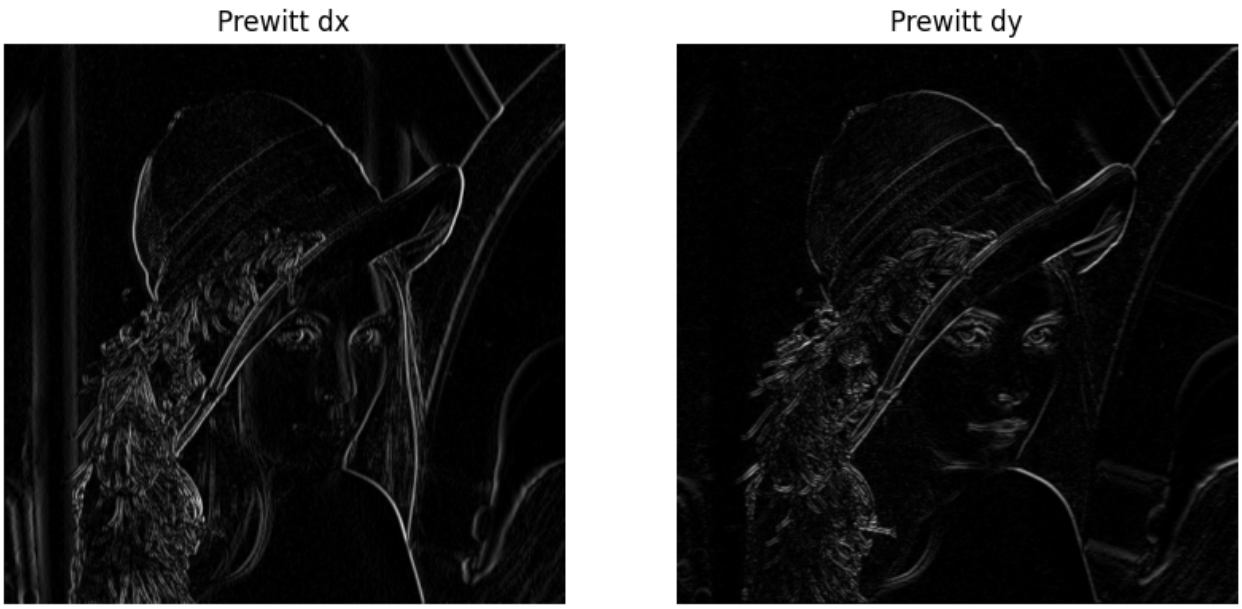

Figure 1: Prewitt filter applied to the original image

Here we notice that the images is highlighting the edges that were in the orignial image.

Comparision with larger filter (5 by 5)

I used the following 5×5 filter to produce image dipicting the edges in the original image:

$$\delta_x = \frac{1}{15}\begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -1 & -2 & 0 & 2 & 1 \\ -1 & -2 & 0 & 2 & 1 \\ -1 & -2 & 0 & 2 & 1 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$$

and,

$$\delta_y = \frac{1}{15}\begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -2 & -2 & -2 & -2 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Here the 2nd and 2nd last enteries have greater value than the first and the last enteries. This is because the center pixel has more weightage than the pixels at the corners. This is done to increase localization in the gradient.

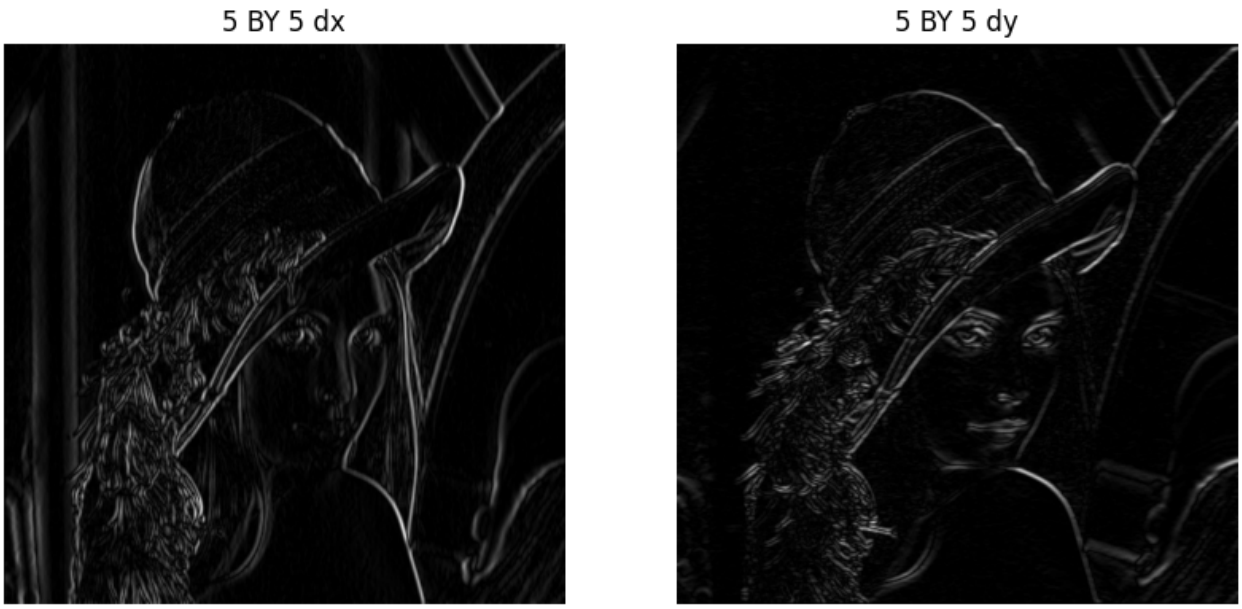

Figure 2: 5 by 5 filter applied to the original image

Results

The δ_x filter detects vertical edges and the δ_y filter detects horizontal edges. This is happening because the filter computes gradient about the direction mentioned in its subscript. So when there is sudden change in the pixel values in the direction of the filter, the filter gives a high value. Since edges are nothing but sudden changes in a function. This is why the edges are highlighted in the images.

Few differences observed between different size of filters:

- The edges are thicker in 5×5 than in 3×3.
- 3×3 does a lot better in preserving the hair details.

2. Sobel Edge Detection Implementation. ***(Marks:)***

   (a) Write a Python function to apply the Sobel filter given below to detect edges along the x-direction and y-direction. Combine these to compute the gradient magnitude image.

   1.

-1	0	1
-2	0	2
-1	0	1

   2.

1	2	1
0	0	0
-1	-2	-1

   3.

1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

   4.

-1	-2	0	2	1
-2	-3	0	3	2
-3	-5	0	5	3
-2	-3	0	3	2
-1	-2	0	2	1

   (b) Apply your function to an image and display the gradient magnitude image alongside the original. Vary the size of the kernel and document the effects.

   (c) Manually implement thresholding on the gradient magnitude to create a binary edge image. Experiment with different thresholds and show the results.

   (d) Apply the Sobel edge detector on a noisy image (you may add synthetic noise to an attached clean image). Discuss how noise affects edge detection and the visual quality of the output images.

Answer:

(a)

Sobel filter (3 by 3)

We will use the 3×3 Sobel filter to produce image dipicting the edges in the original image:

$$\delta_x = \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

and,

$$\delta_y = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Thus we get the gradients in different directions.

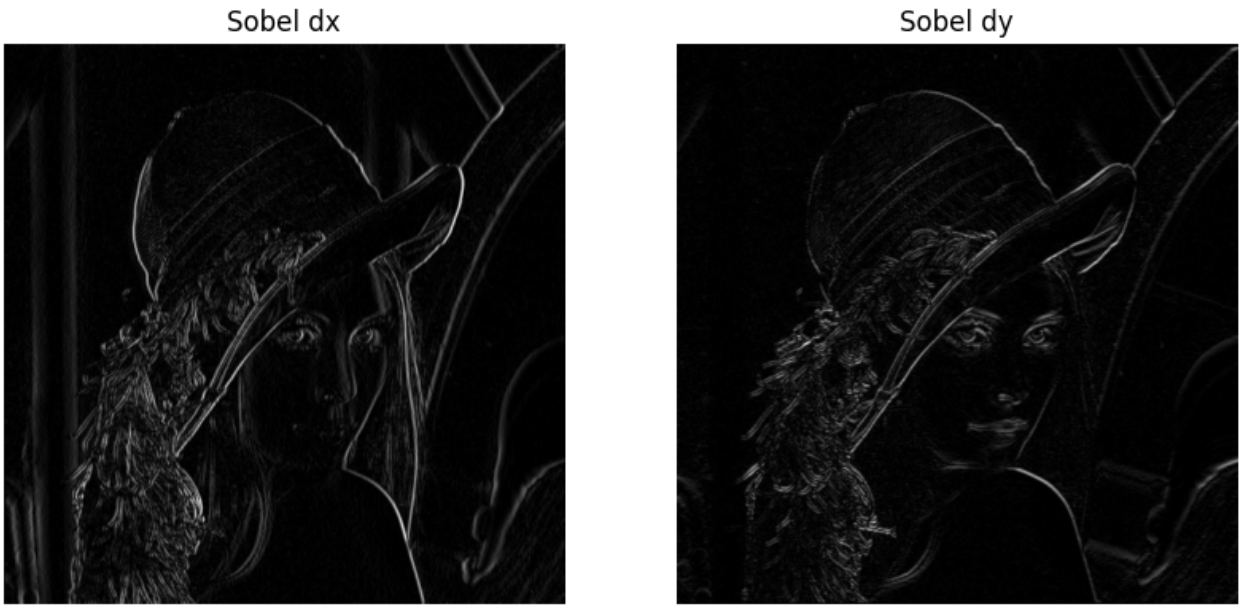

Figure 3: Sobel filter applied to the original image

Comparision with 5 by 5 Sobel filter

We will use following 5×5 filter to produce image dipicting the edges in the original image:

$$\delta_x = \frac{1}{24} \begin{bmatrix} -1 & -2 & 0 & 2 & 1 \\ -2 & -3 & 0 & 3 & 2 \\ -3 & -5 & 0 & 5 & 3 \\ -2 & -3 & 0 & 3 & 2 \\ -1 & -2 & 0 & 2 & 1 \end{bmatrix}$$

and,

$$\delta_y = \frac{1}{24} \begin{bmatrix} -1 & -2 & -3 & -2 & -1 \\ -2 & -3 & -5 & -3 & -2 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 5 & 3 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{bmatrix}$$

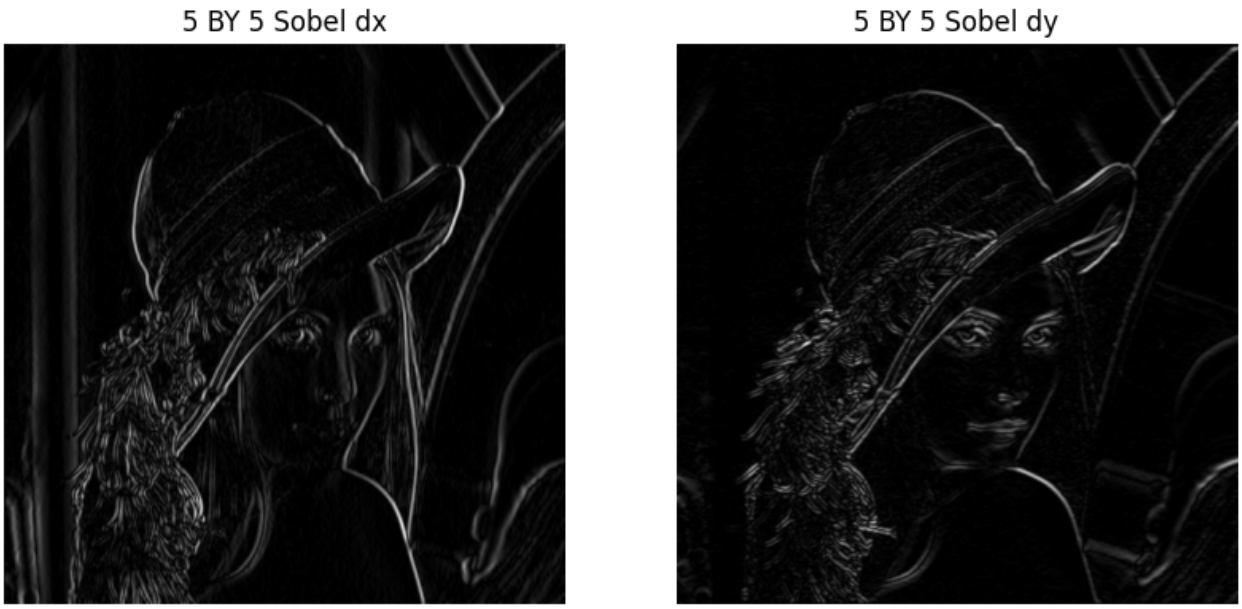

Figure 4: Sobel filter (5 by 5) applied to the original image

Combining the gradients

We can compute the final gradient using the following formula:

$$||\delta(I)|| = \sqrt{\delta_x^2 + \delta_y^2}$$

Here I am basically making a 2d vector out of the two gradients and then taking the magnitude of the vector. This will give us the final gradient image. We have to do this for each corresponding pixels.

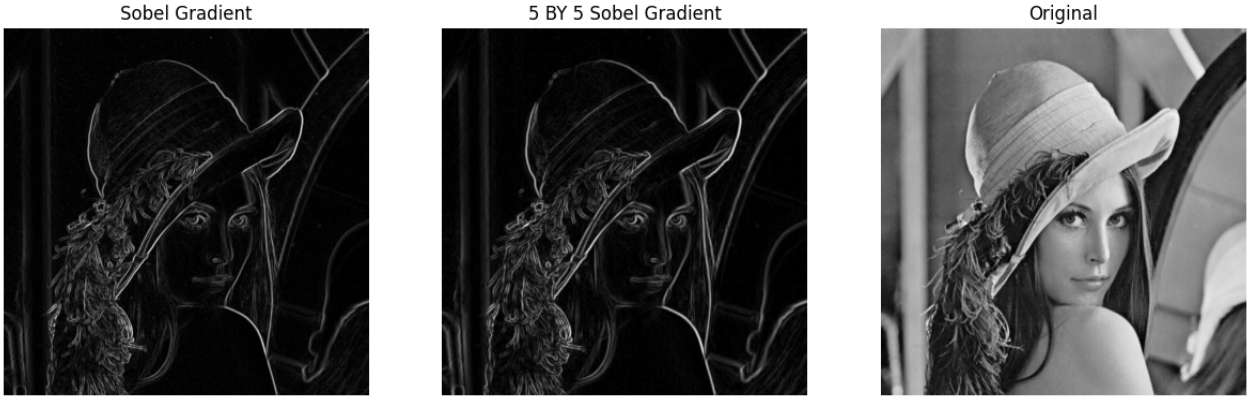

Figure 5: Gradients calculated for different kernel size filters

(b)

Results from different kernel sizes

- The details on hair are well preserved in the 3×3 filter compared to that of the 5×5 filter.
- Overall, the result from 5×5 filter is around a pixel or two width thicker than that of 3×3 filter.

- 5×5 takes mroe time to compute than that of 3×3.

(c)

Thresholding

Thresholding is a technique used to convert a grayscale image to a binary image. If the pixel value is greater than the threshold value, we set it to 255 (white) else we set it to 0 (black). This will give us a binary image. This is useful in edge detection because we can remove the noise in the image, which may have caused subtle disruptions in the gradient image and only keep the edges.

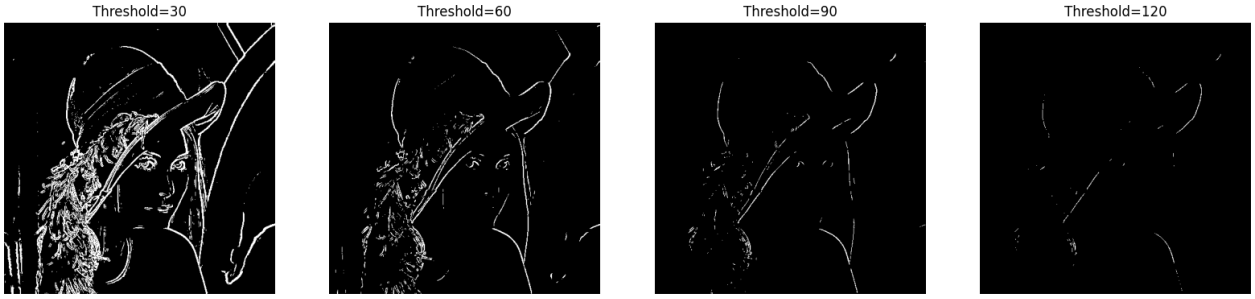

Figure 6: Thresholding applied to the gradient image for Sobel 3×3

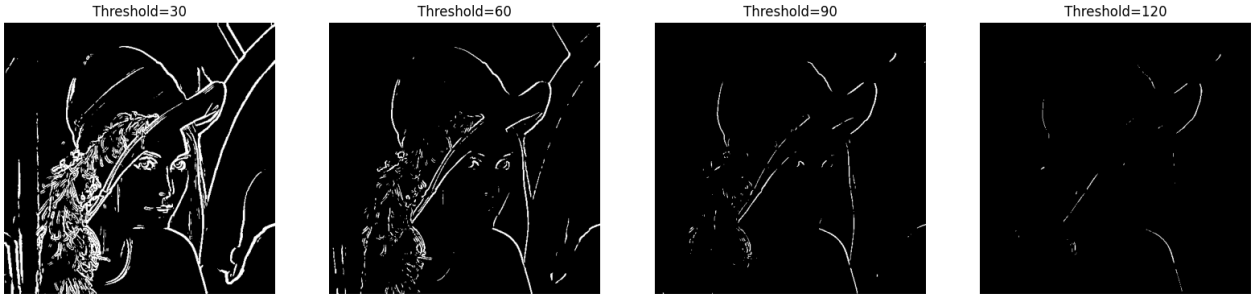

Figure 7: Thresholding applied to the gradient image for Sobel 5×5

Both the filter gives more or less same results here, appart from the edges are still everso slightly thicker in the 5×5 filter.

(d)

Applying noise to the image.

I added gaussian noise to the image. Here I have experimented with different values of sigma for the noise.

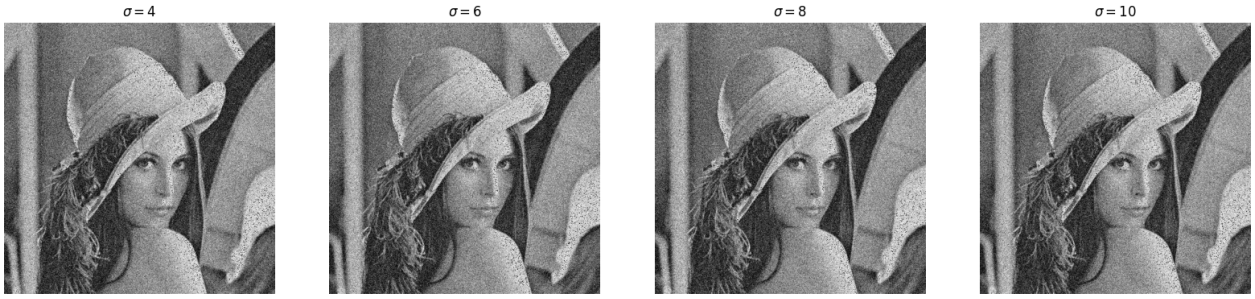

Figure 8: Gaussian noise applied different values of sigma

How noise affects edge detection

Noise in the image heavily impacts the edge detection algorithms we are considering here. The noise in the image causes the gradient image to have a lot of noise. This noise can be seen in the gradient image. This noise can be removed by thresholding, but this will also remove the subtle edges in the image. This is why it is important to have a clean image before applying edge detection algorithms.

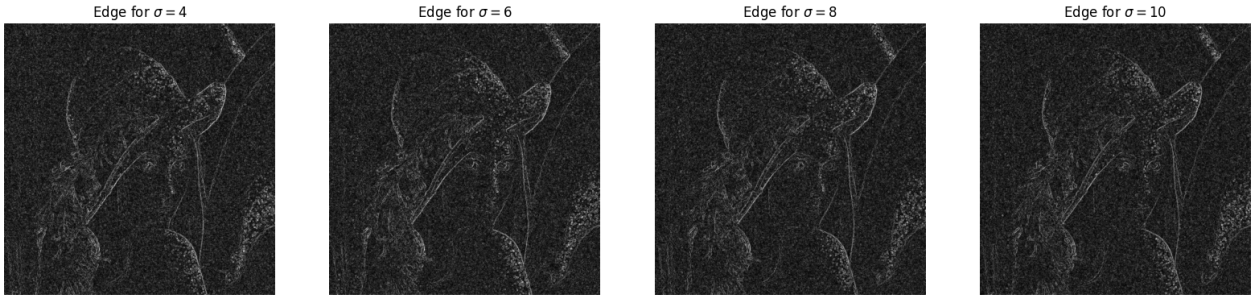

Figure 9: Sobel filter (3×3) applied to the noisy image

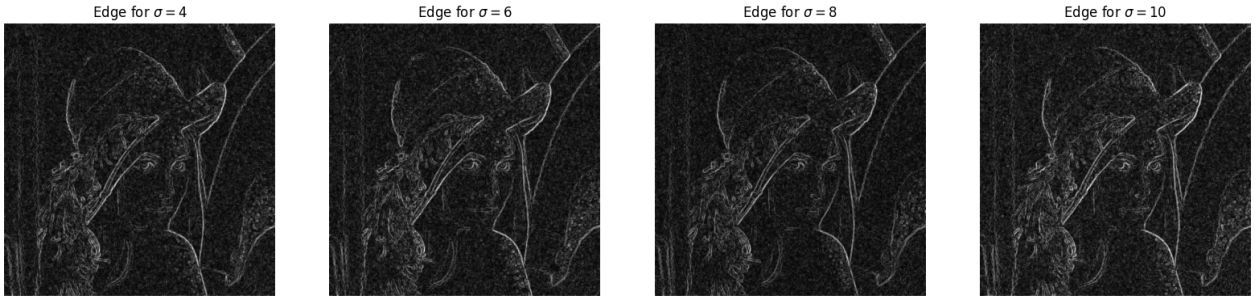

Figure 10: Sobel filter (5×5) applied to the noisy image

Here it is easily observed that the noise in the gradient image obtained by the 3×3 filter is much more than that of in 5×5. Thus impact of noise lowered with larger kernel size. However, both the images are yet noisey, and the edges are not clearly visible. This is why it is important to have a clean image before applying edge detection algorithms.

3. Laplacian of Gaussian Edge Detection (follow the class notes). ***(Marks:)***

   (a) Implement Gaussian smoothing from scratch. Apply your Gaussian filter given below to smooth an image before edge detection.

   (b) Develop the Laplacian filter and apply it to the smoothed image from 3 (a) to detect edges via zero-crossings. Describe how you detect zero-crossings in your implementation.

   (c) Display the edges detected from the smoothed image alongside the edges detected from the non-smoothed image. Discuss the differences and the impact of noise.

Answer:

(a)

Gaussian filter and Gaussian blur

I used the Gaussian filter to smoothen the image to reduce impact of noise in edge detection algorithms being used here. The Gaussian filter is used to blur the image. The filter is given by:

$$n_\sigma = \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

This is reffered as Gaussian blur and is very often used in image processing.

Figure 11: Gaussian filter applied to the original image

The result is slightly blurred as the kernel size is very small (3×3). Image will get more blurry when we increase the kernel size, as the kernel would have more pixels to interact with.

(b)

Laplacian filter

The Laplacian filter is a second-order derivative filter used in image processing for edge detection and enhancement. It highlights regions of rapid intensity change, which correspond to edges in an image. I used the following laplacian filter that was taught in class:

$$\Delta^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

This filter is convolved with the image to produce the following image:

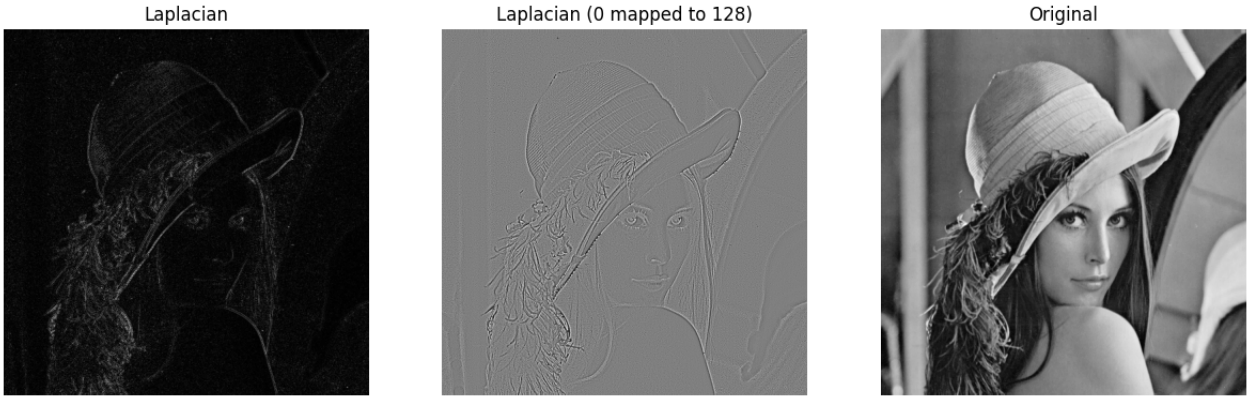

Figure 12: Laplacian filter applied to the original image

Zero crossing

The way I implemented zero crossing here is for each pixel value $P_{i,j}$ I am checking if the sign is changed from previous pixels, that is $P_{i-1,j}$ and $P_{i,j-1}$. If the sign is changed then I am marking that pixel as edge pixel if the change between the pixels is greater than a certain threshold given to the function of zero crossing I have implemented.

As observed here, no matter how much I increase the zero crossing, the results are consistently noisy. This is because the image is noisy and the zero crossing is not able to detect the edges properly due to it's small size of kernel. Also, we would want to keep the threshold as low as possible, as more the threshold, more the edges will be lost.

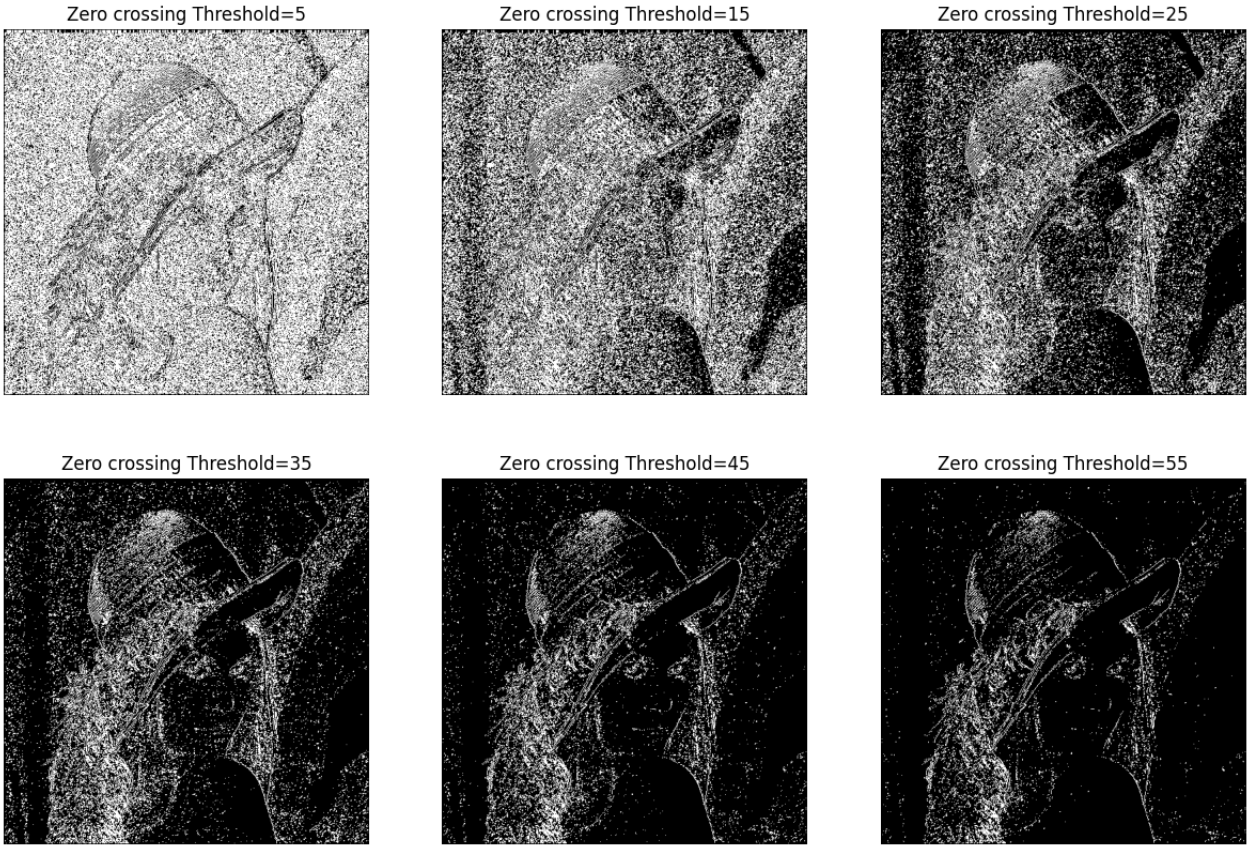

Figure 13: Zero crossing applied to the laplacian image with different threshold

Laplacian of Gaussian (LoG)

To over come the issue of noise in the laplacian filtered image, we would want our main image to be our of noise, i.e., we want the original image to be smooth. This can be done using gaussian filter. The LoG is the convolution of the laplacian filter and the gaussian filter. The LoG is given by:

$$LoG * I = \Delta^2 * n_\sigma * I$$

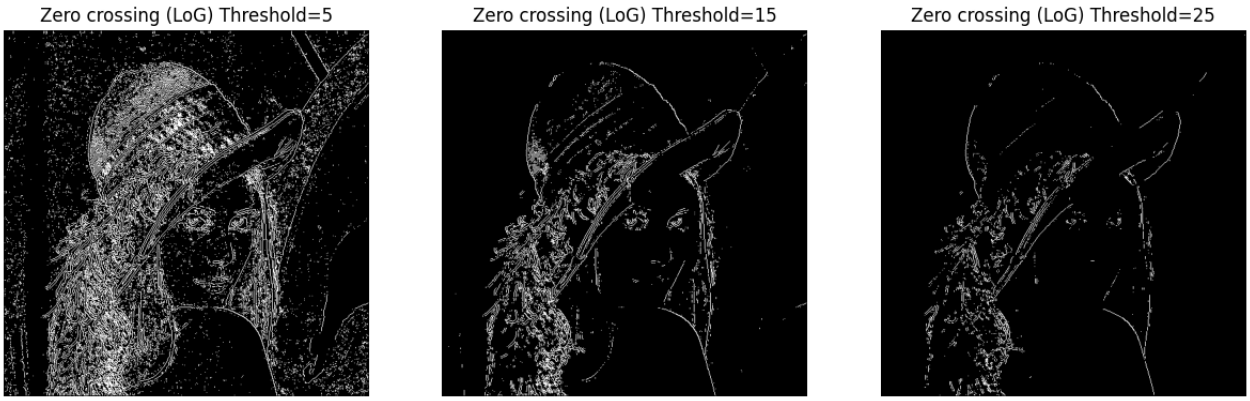

Figure 14: LoG applied to the original image

(c)

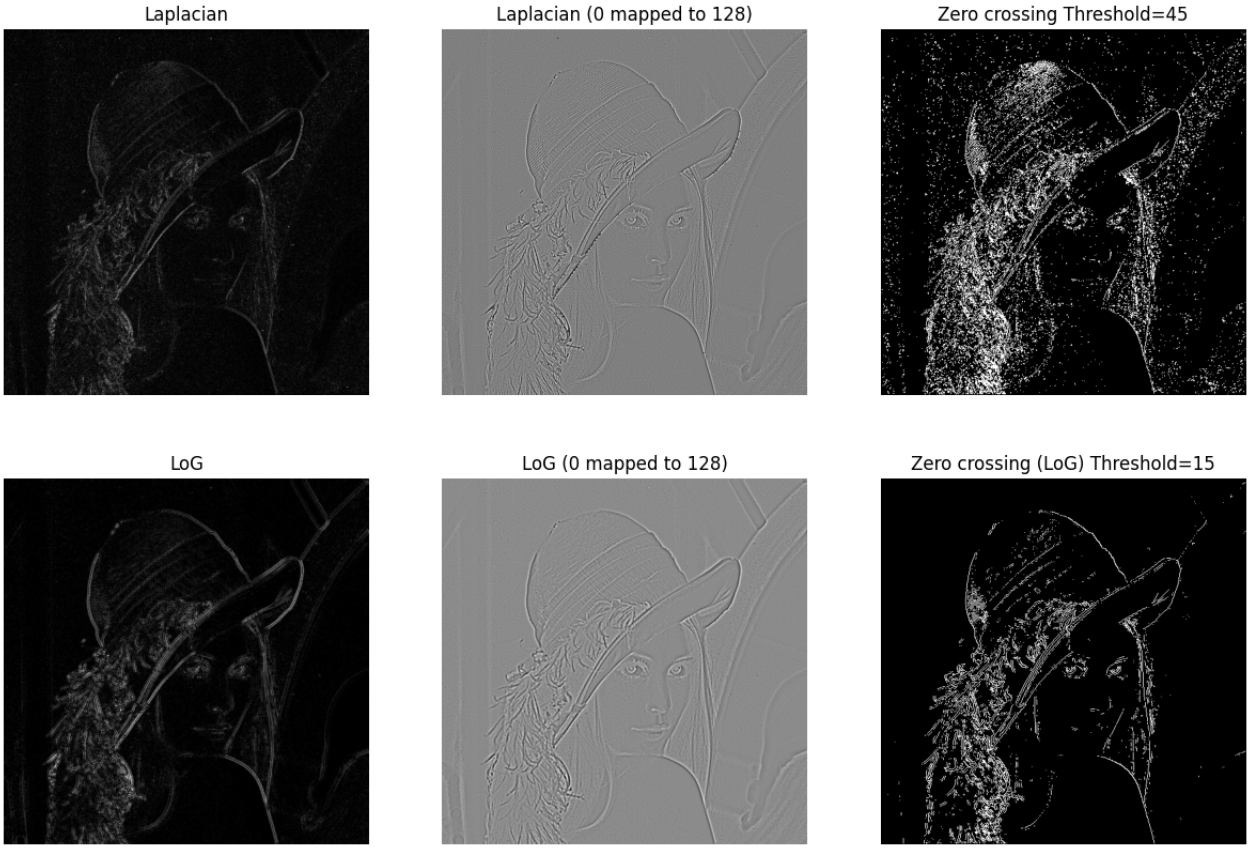

Figure 15: LoG vs Laplacian filter

As you can see that we get very good results at $threshold = 15$ only, compared to that of useless results with only laplacian filter. Here below is a side my side comparision of The original image, laplacian filter and LoG filter.