Assignment 1 : LoG Method

Team Name : The Salvator Brothers
School of Engineering and Applied Science, Ahmedabad University

Computer Vision - CSP502, Winter 2021

1st Kirtan Kalaria
AUL202005
Ahmedabad University
Ahmedabad, India
kirtan.k@ahduni.edu.in

2nd Manav Vagrecha
AU1841022
Ahmedabad University
Ahmedabad, India
manavkumar.v@ahduni.edu.in

***Abstract*—To reduce the sensitivity of the Filters to noise, Marr and Hildreth proposed an edge detection method known as Laplacian of Gaussian operator.**

***Index Terms*—Marr-Hildreth Edge Detector, LoG Operator,**

I. Introduction

Marr and Hildreth proposed a Gaussian Filter combined with the Laplacian to filter out the noise before enhancement for edge detection. Thus, it generally known as the Laplacian of Gaussian (LoG) operator or Marr-Hildreth Edge Detector. The fundamental characteristics of LoG edge detector are:

- The detection criteria is the presence of the zero crossing in the 2^{nd} derivative, combined with a corresponding large peak in the 1st derivative.
- The edge location can be estimated using sub-pixel resolution by interpolation.

II. Motivation and Background

The derivative operators such as Laplacian Operator and Local Edge Operator are not very useful because they are very sensitive to noise. Here, in order to remove high frequency noise, in the LoG operator, the smooth filter is Gaussian and to enhance this filter, Laplacian was used.

III. Detailed Mathematical analysis of the algorithm

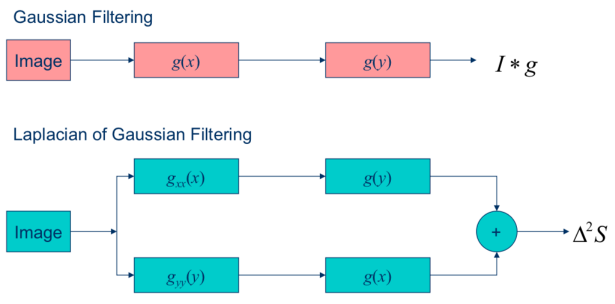

Fig. 1. Gaussian vs Laplacian of Gaussian Filtering method

As shown in the Fig: 1, the Gaussian Filtering simply uses gaussian over the image while Marr-Hildreth method uses Laplacian over Gaussian operator. Thus, the mathematical expression will be:

$$\nabla^2(f * g) = f * \nabla^2 g$$

$$\therefore \nabla^2(f * g) = f * \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) g$$

$$\therefore \nabla^2(f * g) = f * \frac{\partial^2 g}{\partial x^2} + f * \frac{\partial^2 g}{\partial y^2}$$

$$\text{Also, } \nabla^2 g(x,y) = \left[\frac{(x^2 + y^2 - 2\sigma^2)}{\sigma^4}\right] exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

The LoG operator takes the second derivative of the image. Where the image is basically uniform, the LoG will give zero. Wherever a change occurs, the LoG will give a positive response on the darker side and a negative response on the lighter side.

At a sharp edge between two regions, the response will be:

- zero away from the edge
- positive just to one side
- negative just to the other side
- zero at some point in between on the edge itself

The 2 well known 3x3 LoG operators are 4-neighbours : $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$, 8-neighbours : $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$ is the most common 5x5 Laplacian of Gaussian Operator

IV. Marr-Hildreth Algorithm

1) Compute LoG
   - Using 1 2-D filter { $\nabla^2 g(x,y)$ }
   - Using 4 1-D filters { $g(x).g_{xx}(x).g(y).g_{yy}(y)$ }
2) Find zero-crossings from each row and column
3) Find slope of zero-crossings
4) Apply threshold to slope and mark edges

V. Comparison with other Algorithms

A. Time Complexity

- Unlike the Sobel's and Prewitt's edge detectors, the Laplacian of Gaussian (LoG) edge detector uses only one kernel instead of one each for different axes.
- LoG combines the Laplacian and Gaussian into one single kernel, saving number of operations / computations.
- LoG can be executed by 4 1-D convolutions, saving time massively.
- Moreover, LoG can be fairly estimated by difference of two Gaussians.

B. Space complexity

- It's space complexity is the least among itself and Sobel, Canny, Robert, Prewit and plain Zero Crossing.
- 4 1-D and DoG approximation techniques have even lesser space complexity.

C. Sensitivity to noise

- Unlike Laplacian and Local Edge Operators, it less sensitive to noise.
- The Gaussian element reduces possible noise sensitivity of Laplacian, thereby reducing overall sensitivity.
- Higher σ leads to reduced noise sensitivity at the cost of missing out finer edges.
- Involving 2^{nd} derivatives, it is still fairly sensitive.

D. Drawback of algorithm

- It responds to non-edge points ie. false edges are detected.
- There exists much better edge detection methods, such as Canny Edge Detector based on the search for local directional maxima in the gradient magnitude or the differential approach based on the search for zero crossings of the differential expression that corresponds to the second-order derivative in the gradient direction
- Small local noisy edges can derail proper edge detection. Also false positive of edges is higher than many other operators.
- It is isotropic and hence fails to capture the direction of edges.

VI. Results

A. Plots

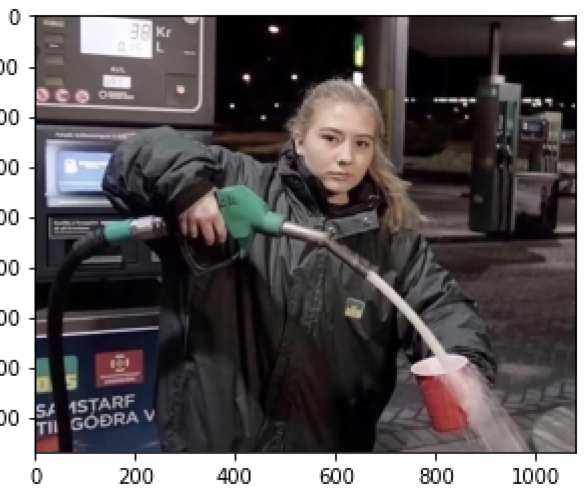
Fig. 2. Input Image

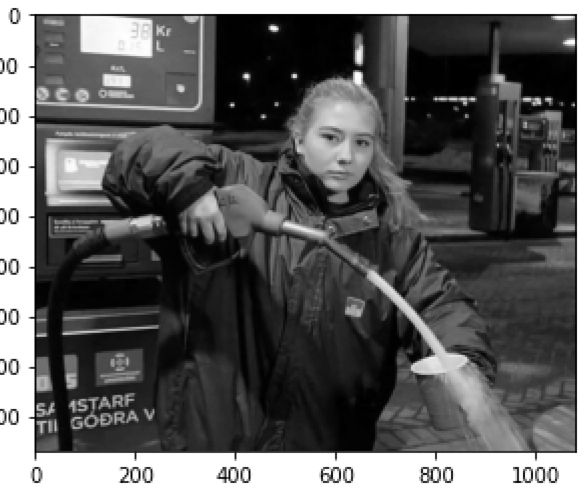
Fig. 3. GreyScaled Image

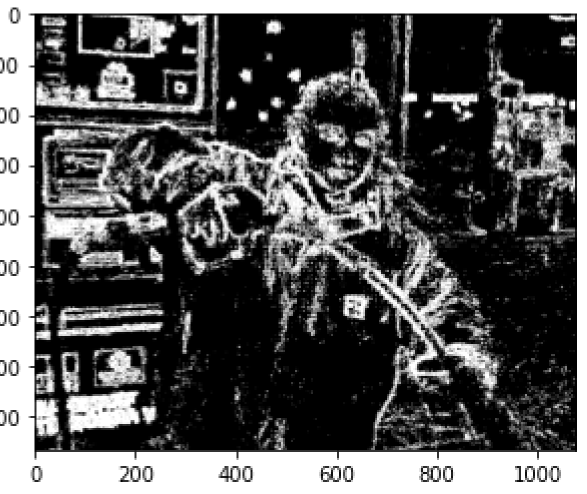
Fig. 4. Output Image after applying LoG over Fig:3 with τ=15 and type-1 kernel

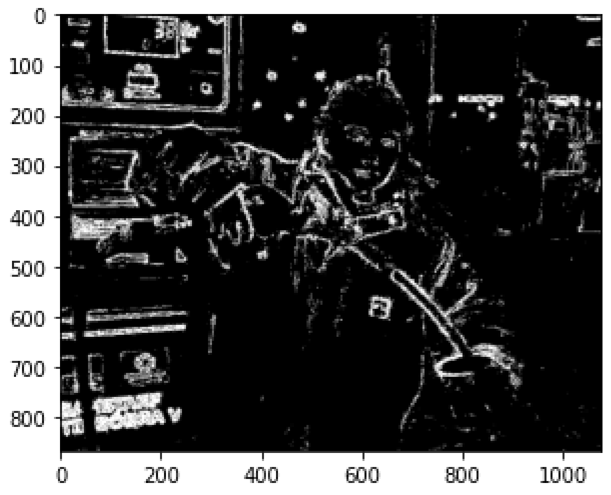
Fig. 5. Output Image after applying LoG over Fig:3 with τ=25 and type-1 kernel

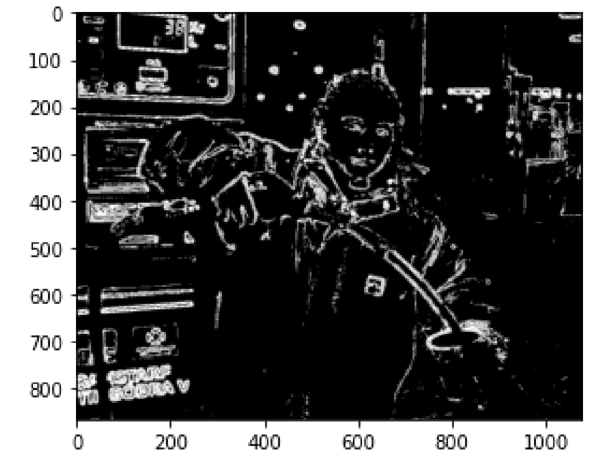
Fig. 6. Output Image after applying inbuilt LoG over Fig:3 with τ=25 and σ=1

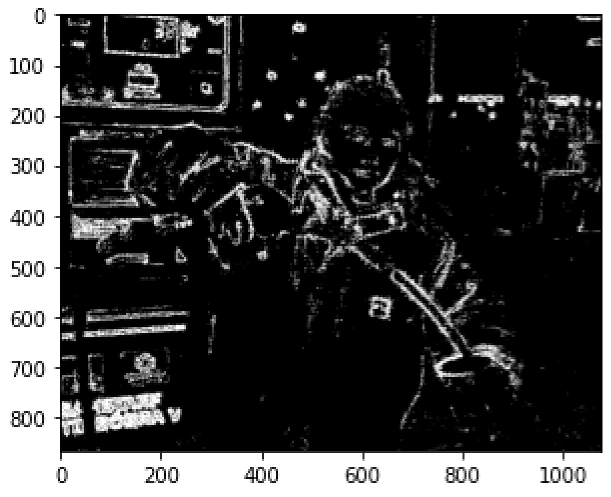
Fig. 7. Output Image after applying LoG over Fig:3 with τ=12 and type-2 kernel

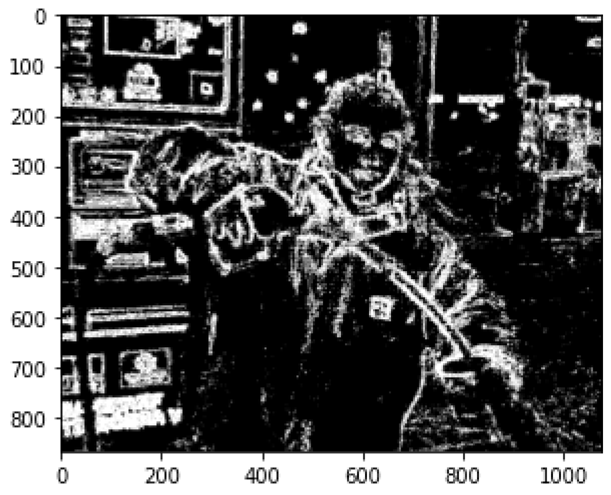
Fig. 8. Output Image after applying LoG over Fig:3 with τ=40 and type-3 (5x5) kernel

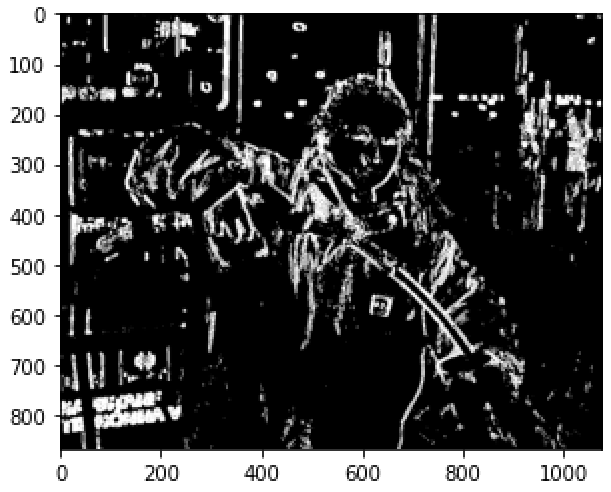
Fig. 9. Output Image after applying Sobel-X kernel over Fig:3 with τ=25

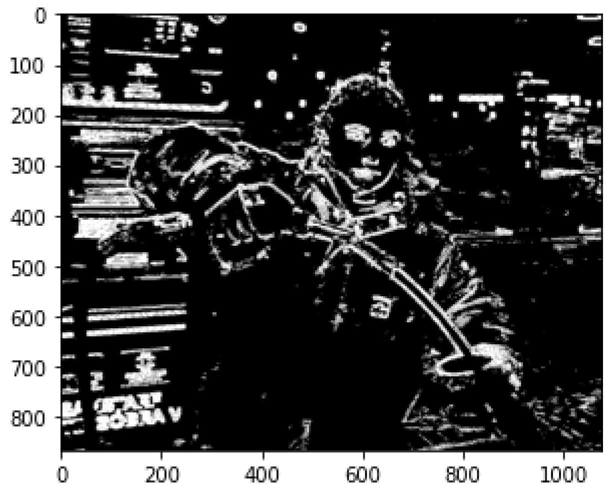
Fig. 10. Output Image after applying Sobel-Y Kernel over Fig:3 with τ=25

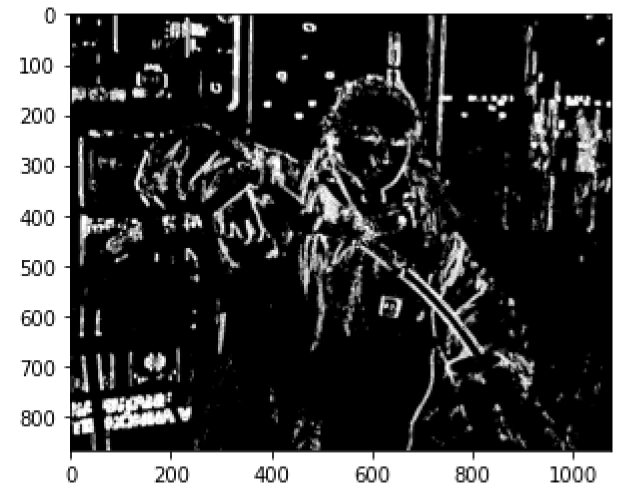

Fig. 11. Output Image after applying Priwet-X kernel over Fig:3 with τ=30

B. Explanation of Code and Results

Here, Fig: 2 is the Input image. After using the *mean()* function across RGB channels of Fig.2, we obtain a gray-scaled image Fig: 3. After passing Fig: 3 over the pipeline function *filter(mask, threshold)*, with different masks, we get all the following figures except Fig.6, which is the output obtained by using the inbuilt $gaussian\_laplace(image, sigma)$ function with $\sigma = 1$. For the filtering process, we have created functions *convolve(), en_convolve(), and mark_edges().*

convolve(image, mask)) function does simple, raw convolution of the image with the mask and gives us a convoluted image. It does not re-scale the image.

en_convolve(image, mask)) function convolutes the image with the mask like *convolve* but additionally, it re-scales all the values in the range of 0-255 i.e.8-bit.

After obtaining the convoluted image using simple *convolve* function, we need to find the zero-crossings, check slopes at those crossings with a threshold and mark the selected pixels as edge pixels for the final binary image. Thus, we have designed *mark_edges(convoluted-image, threshold)* to accomplish the task and mark all the edges.

Fig.4 and Fig.5 show the result of LoG with different thresholds. We can visually infer that $\tau = 25$ is a better suited value. In all cases, we must select the threshold by a little trial and error for best result. Ideally, we want to keep it high enough to remove unwanted noise while low enough to actually catch edges. Fig.6 shows the result of the filter function from scipy.ndimage. We can see that though the ready-made function is slightly better than ours, the performance of both is comparable. Fig.7 is done using another version of the 3×3 LoG kernel. Then Fig.8 shows the result of using a 5×5 LoG kernel. We can notice that it reduces local noise and focuses on main edges.

Fig.9,10,11 show the results of Sobel-X, Sobel-Y and Prewitt-X kernels with appropriately chosen thresholds for each. Not surprisingly, Prewitt-X and Sobel-X perform quite closely. These 3 kernels result in poorer, anisotropic edge detection which results in broken edges. We can see how LoG is a better choice in this manner.

References

[1] "Computer Vision Demonstration Website," Laplacian & Marr Hildreth Edge Detection - Computer Vision Website Header. [Online]. Available: https://www.southampton.ac.uk/~msn/book/new_demo/laplacian/. [Accessed: 05-Feb-2021].

[2] T. G. Smith, W. B. Marks, G. D. Lange, W. H. Sheriff, and E. A. Neale, "Edge detection in images USING Marr-Hildreth filtering techniques," 14-Mar-2003. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/0165027088901306. [Accessed: 05-Feb-2021].

[3] D. Marr, E. Hildreth, "Theory of Edge Detection" 22-Feb-1979. [Online]. Available: http://www.hms.harvard.edu/bss/neuro/bornlab/qmbc/beta/day4/marr-hildreth-edge-prsl1980.pdf