# Implementing Edge Detection Using Various Algorithms

Parth Parashar

## IMAGE PREPROCESSING

- Loading the image using opencv
- Converting the image into grayscale
- Applying Gaussian Blur on the image
- In case of various methods discussed below, before displaying the image , all gradient values were converted in to positive(absolute value function) since we need to focus only on magnitude of intensity change , not the direction.

## DISPLAYING THE OUTPUT

We use matplotlib and divide it into subplots in order to compare the original and changed images . Moreover, in order to display the original image we need to convert it into RGB format since opencv by default stores any image in BGR format.

## SOBEL EDGE DETECTION

Sobel edge detection involves using a 3*3 matrix(kernel) and calculating the gradient magnitude in x and y directions ($G_x$ , $G_y$). Both are combined using $\sqrt{G_x^2 + G_y^2}$. A threshold of 50 was used since a value around 50 strikes a good balance: it is high enough to suppress minor variations and noise, yet low enough to retain important edge details. Setting the threshold too low results in detecting too many false edges caused by noise, while setting it too high may cause the loss of significant edges.

## SCHARR EDGE DETECTION

Just like Sobel , Scharr also uses a 3*3 kernel in similar manner (uses gradient) and a similar threshold came out to be good enough. Though the exact entries

in the kernel are different from that in case of Sobel.

## LAPLACIAN EDGE DETECTION

Unlike Sobel and Scharr , this method involves application of second order derivative which higlights regions of rapid intensity. For the given image any threshold above 40 was giving nearly completely black image , the best edge detection was observed in case of threshold = 19.

## CANNY EDGE DETECTION

It involves five steps , and gives the cleanest edges though bit faded. After experimentation with upper and lower thresholds , the best suited pair was found to be (50,20)

## COMPUTATIONAL AND VISUAL DIFFERENCES

- **Sobel and Scharr:** Both are computationally efficient, involving simple convolutions with small kernels. Scharr is slightly more intensive but provides sharper and more accurate edges compared to Sobel.
- **Laplacian:** Computationally efficient as well, but being a second-order derivative, it is more sensitive to noise and often requires prior smoothing for clean edge detection.
- **Canny:** The most computationally demanding due to its multi-stage process including Gaussian blurring, gradient calculation, non-maximum suppression, and double thresholding. This complexity results in superior, continuous, and noise-resistant edges.
- **Visual Quality:** Sobel and Laplacian may produce noisy or broken edges; Scharr offers clearer and more precise edges, while Canny yields the cleanest and most complete edge maps, robust to noise.

## CONCLUSION

These experiments showed that hyperparameter tuning is key to good edge detection. Sobel and Scharr are fast but sensitive to noise, while Laplacian needs careful preprocessing. Canny offers the best edge quality and noise resistance but requires more computation. Choosing the right method depends on the balance between accuracy and efficiency for the task.