

**19CSE413**  
**DIGITAL IMAGE PROCESSING**

**CB.EN.U4CSE20309**  
**AVANTIKA BALAJI**

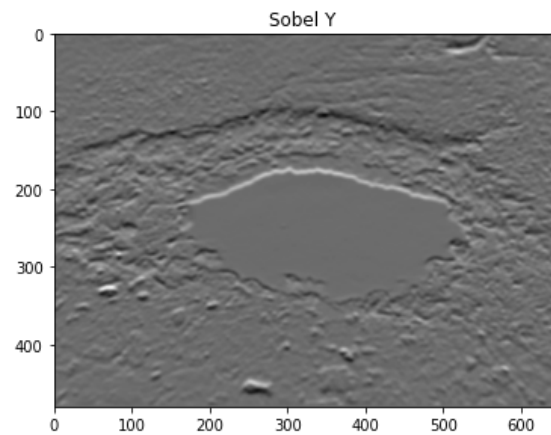
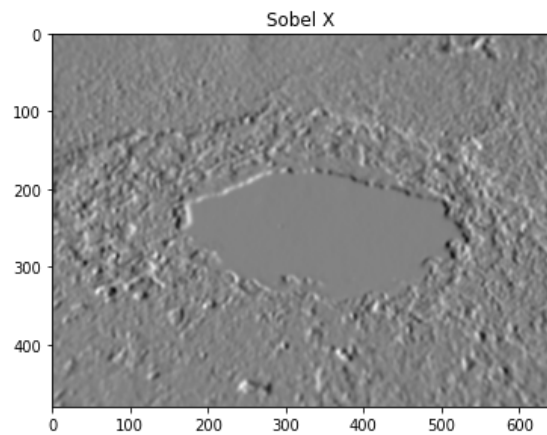
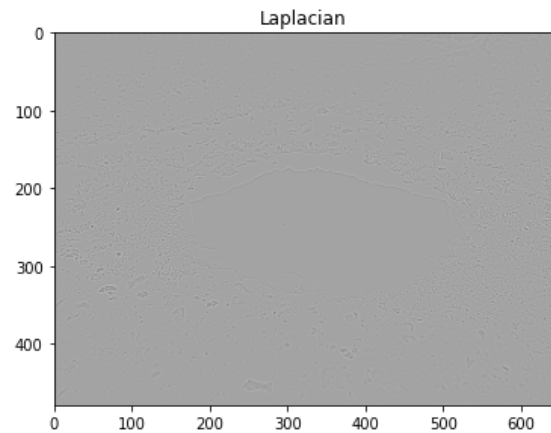
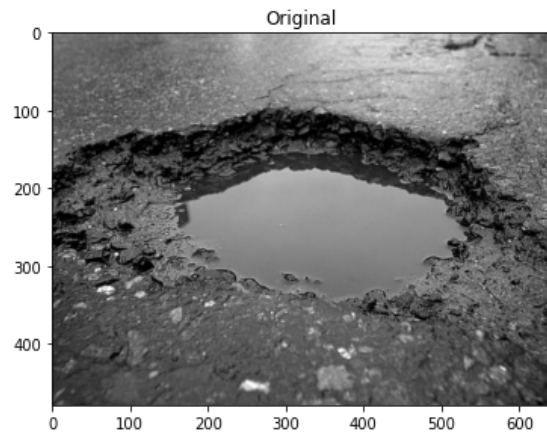
**EVALUATION - 2**

## Laplacian, Sobel Filtering Technique

```
laplacian = cv2.Laplacian(sharpened_h,cv2.CV_64F)
sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=21)
sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=21)

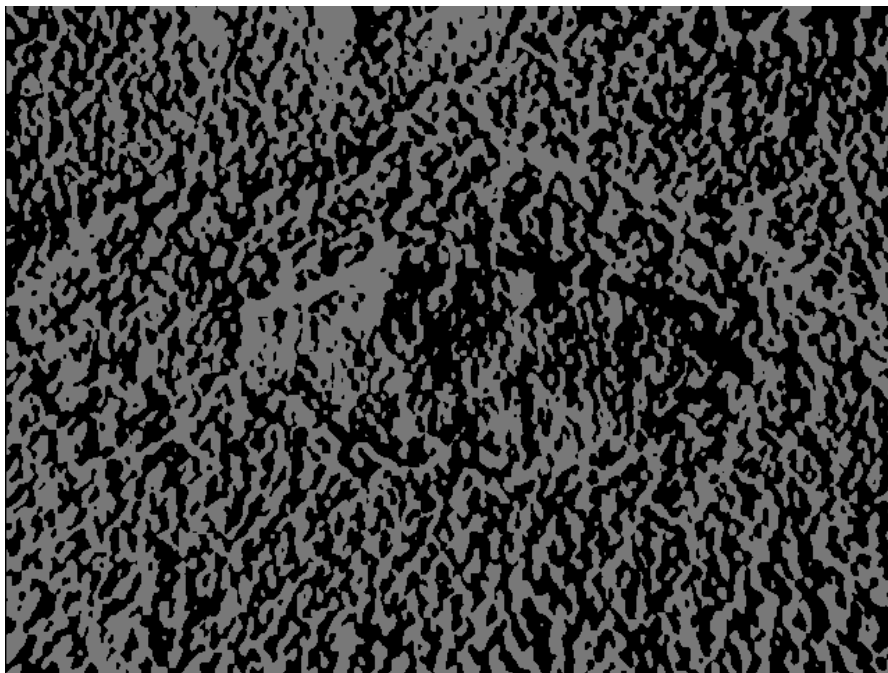
plt.figure(figsize=(15, 10))
plt.subplot(2,2,1)
plt.imshow(img, cmap = 'gray')
plt.title('Original')
plt.subplot(2,2,2),plt.imshow(laplacian,cmap = 'gray')
plt.title('Laplacian')
plt.subplot(2,2,3)
plt.imshow(sobelx,cmap = 'gray')
plt.title('Sobel X')
plt.subplot(2,2,4)
plt.imshow(sobely,cmap = 'gray')
plt.title('Sobel Y')

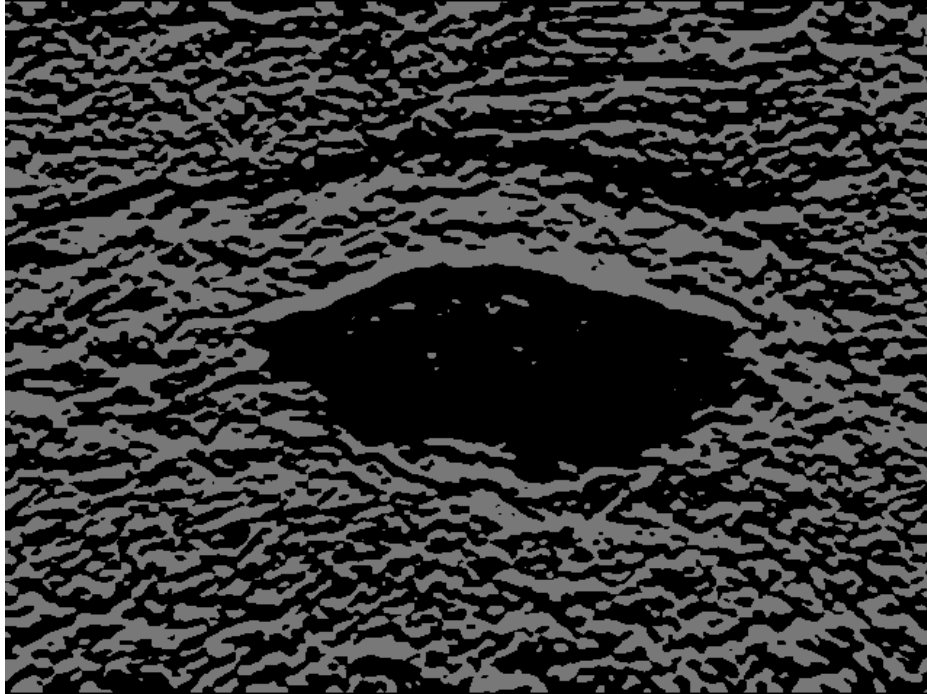
plt.show()
```



## Thresholding of Laplacian and Sobel

```
from google.colab.patches import cv2_imshow
ret, thresh1= cv2.threshold(laplacian, 120, 255, cv2.THRESH_TRUNC)
ret, thresh2= cv2.threshold(sobelx, 120, 255, cv2.THRESH_TRUNC)
ret, thresh3= cv2.threshold(sobely, 120, 255, cv2.THRESH_TRUNC)
cv2_imshow(thresh1)
cv2_imshow(thresh2)
cv2_imshow(thresh3)
# De-allocate any associated memory usage
if cv2.waitKey(0) & 0xff == 27:
    cv2.destroyAllWindows()
```





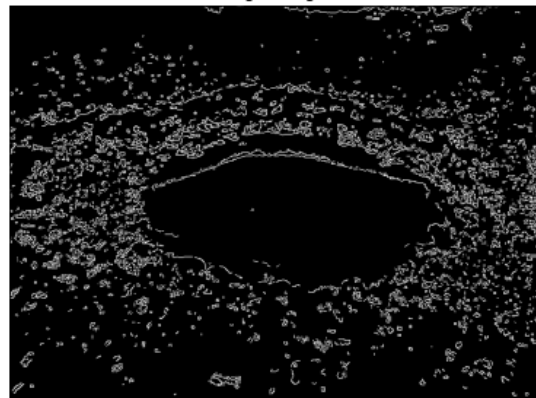
## Canny Edge Detector

```
image = np.uint8(sharpened_h)
edges = cv2.Canny(image,300,400)
plt.figure(figsize=(15,5))
plt.subplot(121),plt.imshow(img,cmap = 'gray')
plt.title('Original Image'), plt.xticks([], plt.yticks([]))
plt.subplot(122),plt.imshow(edges,cmap = 'gray')
plt.title('Edge Image'), plt.xticks([], plt.yticks([]))
plt.show()
```

Original Image



Edge Image

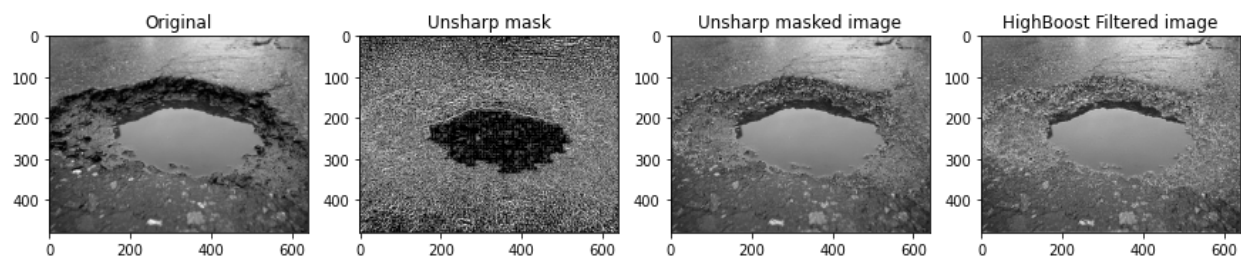


## Unsharp Masking and High Boost Filtering

```
from skimage import data
from skimage import img_as_ubyte, img_as_int
from scipy import ndimage

low_pass = (1/9)*np.array([[1, 1, 1], [1, 1, 1], [1, 1, 1]],
dtype='float64') #LP mask
blurred = ndimage.convolve(img, low_pass, mode='constant', cval=0.0)
unsharp_mask = img_as_ubyte(img - blurred)
unsharp = img_as_ubyte(img + 1*unsharp_mask)
highboost=img_as_ubyte(img + 3*unsharp_mask)

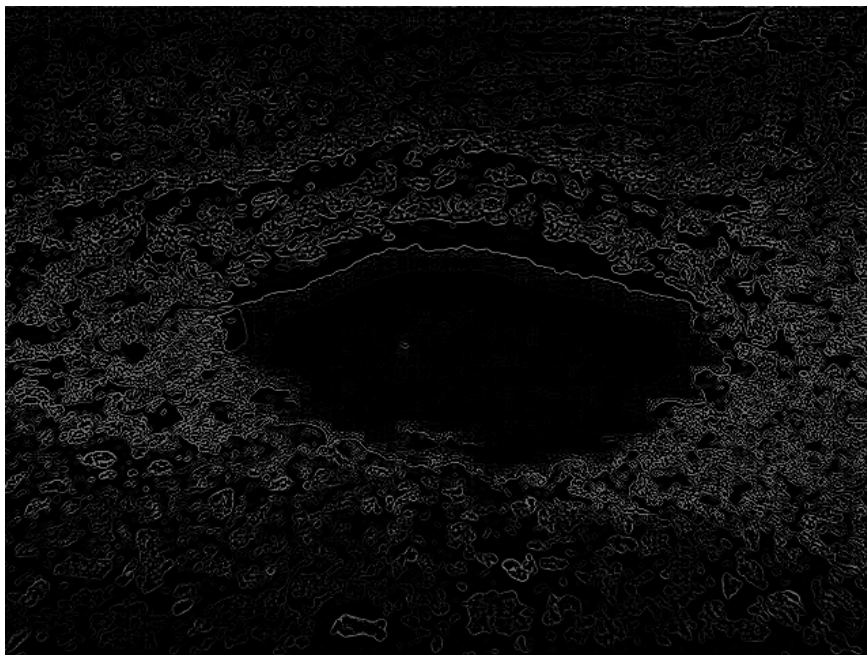
plt.figure(figsize=(15,5))
plt.subplot(1, 4, 1)
plt.title('Original')
plt.imshow(img,cmap="gray")
plt.subplot(1, 4, 2)
plt.title('Unsharp mask')
plt.imshow(unsharp_mask,cmap="gray")
plt.subplot(1, 4, 3)
plt.title('Unsharp masked image')
plt.imshow(unsharp,cmap="gray")
plt.subplot(1, 4, 4)
plt.title('HighBoost Filtered image')
plt.imshow(highboost,cmap="gray")
```



## Frequency Domain Filtering VS Spatial Domain Filtering

```
#comparison of Frequency domain filtering vs Spacial Domain Filtering
from google.colab.patches import cv2_imshow
ret, thresh= cv2.threshold(laplacian, 120, 255, cv2.THRESH_TRUNC)
print("Frequency threshold")
cv2_imshow(thresh)
#spatial image
med_image = median_filter(img,5)
hist,bins = np.histogram(med_image.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * hist.max()/ cdf.max()
cdf_m = np.ma.masked_equal(cdf,0)
cdf_m = (cdf_m - cdf_m.min())*225/(cdf_m.max()-cdf_m.min())
cdf = np.ma.filled(cdf_m,0).astype('uint8')
img2 = cdf[med_image]
lap = cv2.Laplacian(med_image,cv2.CV_64F)
sharp = img2 - 0.9*lap
ret, thresh1 = cv2.threshold(sharp, 180, 255, cv2.THRESH_BINARY)
print("Spatial threshold")
cv2_imshow(thresh1)
```

### Frequency Domain





## Spatial Domain



### LAPLACIAN FILTER:

The Laplacian of an image highlights regions of rapid intensity change and is an example of a second order or a second derivative method of enhancement.

- It is notably good at finding the fine details of an image. A Laplacian operator will improve any feature with a sharp discontinuity.
- This determines if a change in adjacent pixel values is from an edge or continuous progression.
- The Laplacian is a well-known linear differential operator that approximates the second derivative, which is given by:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

where f denotes the image.

### SOBEL EDGE DETECTION

When Sobel Edge Detection is used, the image is processed separately in the X and Y directions before being combined to form a new image that represents the sum of the X and Y edges of the image.

- It is best to first convert the image from RGB to Grayscale.
- Then we use a technique known as kernel convolution. A kernel is a 3 x 3 matrix made up of symmetrically (or differently) weighted indexes. This will be the filter that we will implement for edge detection.

X – Direction Kernel

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

Y – Direction Kernel

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

- Sobel X and Sobel Y are first-order derivatives
- Sobel X simply finds the first-order derivative in the X-direction. It means that it will detect only those edges which are changing in the X direction.
- Similarly, Sobel Y finds the first-order derivative in the Y direction. It means that it will detect only those edges which are changing in the Y direction.
- We can simply add edges in the X direction and edges in the Y direction to get the overall edges in our image

## CANNY EDGE DETECTION

The Canny edge detector is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. Steps involved include:

1. **Noise Reduction:** Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter.

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

### 2. Finding intensity gradient of the image:

- Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction ( G<sub>x</sub>) and vertical direction ( G<sub>y</sub>). From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\_Gradient (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle (\theta) = \tan^{-1} \left( \frac{G_y}{G_x} \right)$$

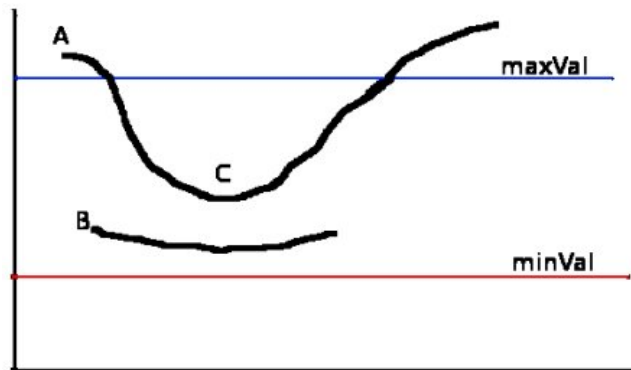


### 3. Non-maximum suppression:

- After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge.
- For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient.

### 4. Hysteresis Thresholding:

- This stage determines which edges are truly edges and which are not. We will need two threshold values for this: minVal and maxVal.
- Any edges with an intensity gradient greater than maxVal are certain to be edges, while those with an intensity gradient less than minVal are certain to be non-edges and should be discarded.
- Those with connectivity between these two thresholds are classified as edges or non-edges.



Why it is useful:

- Good Detection: The optimal detector must eliminate the possibility of getting false positives and false negatives.
- Good Localisation: The detected edges must be as close to true edges.
- Single Response Constraint: The detector must return one point only for each edge point.

### UNSHARP MASKING AND HIGH BOOST FILTERING

The unsharp filter is a simple sharpening operator which derives its name from the fact that it enhances edges (and other high frequency components in an image) via a procedure which subtracts an unsharp, or smoothed, version of an image from the original image.

- The unsharp filtering technique is commonly used in the photographic and printing industries for crispening edges.

Thus unsharp masking, first produces a mask  $m(x,y)$  as :

$$m(x,y) = f(x,y) - f_b(x,y)$$

Where  $f(x,y)$  is the original image and  $f_b(x,y)$  is the blurred version of the original image

Then, this mask is added to the original image which results in enhancing the high frequency components

$$g(x,y) = f(x,y) + k * m(x,y)$$

- Where k specifies what portion of the mask has to be added.
- When k=1, it is called Unsharp Masking, and for k>1 we call it High-Boost Filtering since we are boosting the high-frequency components by giving more weight to the masked (edge) image.

### **Observations -**

#### **Laplacian vs Sobel:**

Between Laplacian and Sobel, we see that Sobel produces a better result. Unlike the Sobel edge detector, the Laplacian edge detector uses only one kernel. The Laplacian filter detects sudden intensity transitions in the image and highlights the edges.

However, its main disadvantage is its high sensitivity to noise, which causes edges to spread with smoothing. It amplifies the noise in the image, as we observe, hence it is not the best filter when we have noisy images.


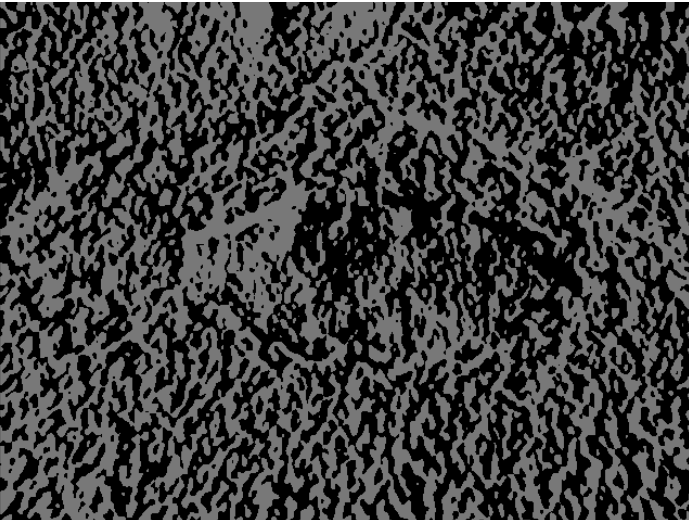
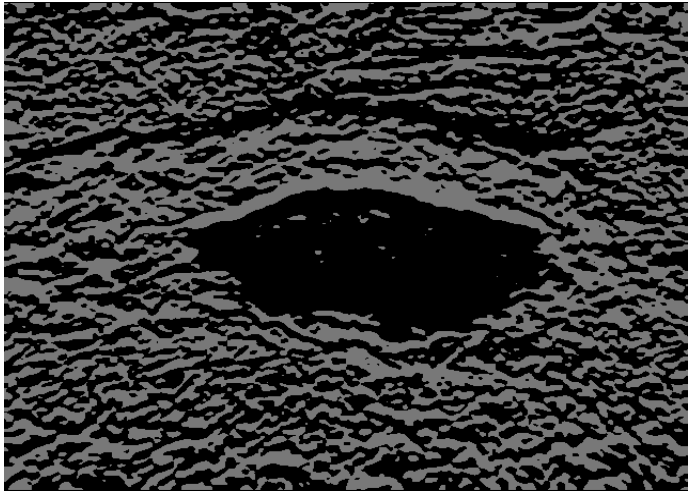
#### **Canny Edge Detector:**

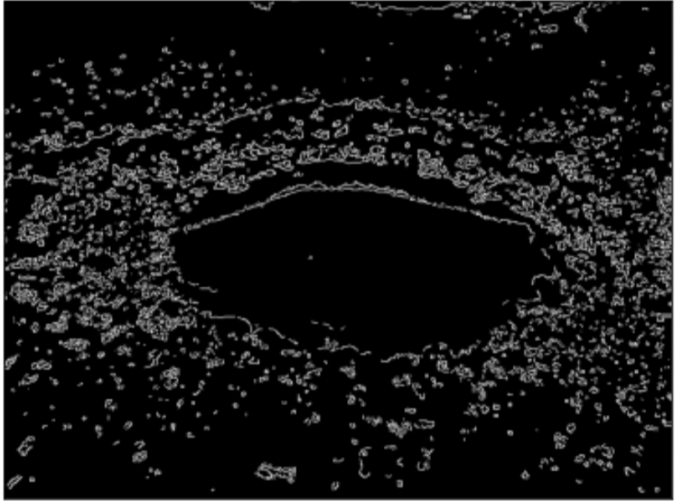
The advantage of Canny Edge Detector is that it produces very thin and clean edges.

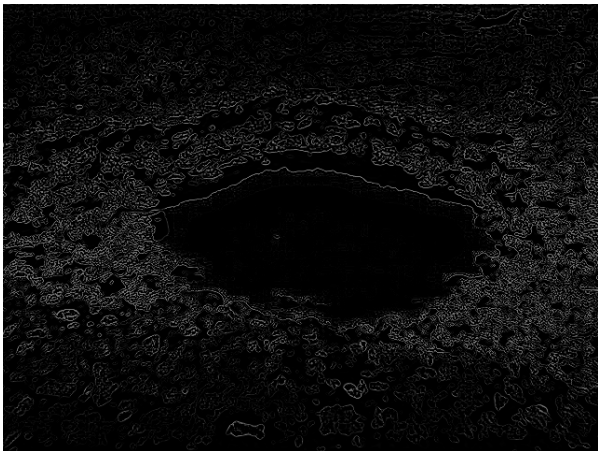

- The presence of Gaussian filter allows removing of any noise in an image.
- The signal can be enhanced with respect to the noise ratio by non-maxima suppression method which results in one pixel wide ridges as the output.
- Detects the edges in a noisy state by applying the thresholding method.
- The effectiveness can be adjusted by using parameters.
- It gives a good localization, response and is immune to a noisy environment.

#### **High Boost Filtering:**

It is simply a sharpening operator that is applied on the image to get sharp edges. The main drawback pertaining to this technique is amplification of noise pixels along with the information pixels, depending on appropriate choice of amplification factor. So a suitable noise-eliminating filter needs to be deployed in order to cancel the effect of noise amplification.

No	Filter	Edge Detection using filter
1	Laplacian	
2	Sobel - x	
3	Sobel - y	

4	Canny Edge	<p>Edge Image</p> 
---	------------	--

Frequency Domain Filtering	Spatial Domain Filtering
	

### Conclusion-

- *We observe that we get the best output images after applying low pass filters with a threshold of 50 and high pass filters with a threshold of 10.*
- *Based on our output images, we also see that Canny Edge Detection works best on analysing the edges of the potholes on the road. The edges are properly highlighted and marked without ambiguity.*
- *We see that frequency domain filtering produces a much better filtered output than spatial domain filtering.*

- The reason for doing the filtering in the frequency domain is generally because it is computationally faster to perform two 2D Fourier transforms and a filter multiply than to perform a convolution in the image (spatial) domain. This is particularly so as the filter size increases.
- While spatial filters are more commonly used for edge detection, frequency filters are more often used for high frequency emphasis. Here, the filter doesn't totally block low frequencies, but magnifies high frequencies relative to low frequencies.
- Moreover the filtering process in frequency domain is much simpler than spatial domain.