IPPR Lab 5

Name : Arya Shah

Roll No. E071

Class : BTech CSBS

Aim: To Detect the Edges Of The Image Uisng The Sobel And The Laplacian Operators

```
#Importing Libraries
from skimage import io
import matplotlib.pyplot as plt
from skimage.color import rgb2gray
import numpy as np
from scipy import signal
from random import randint
```

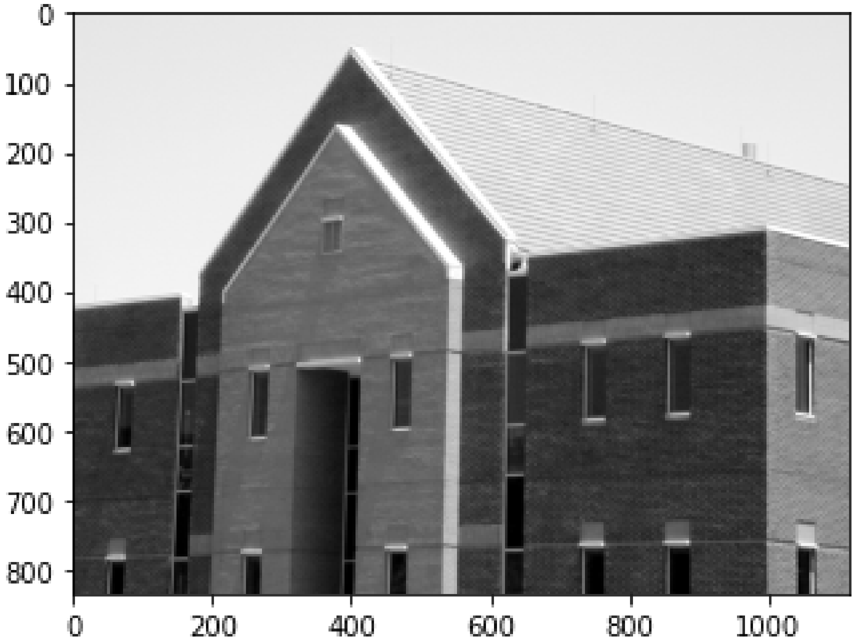
```
#Import Image
image=io.imread('roof.tif')
```

```
image.shape
```

```
(834, 1114)
```

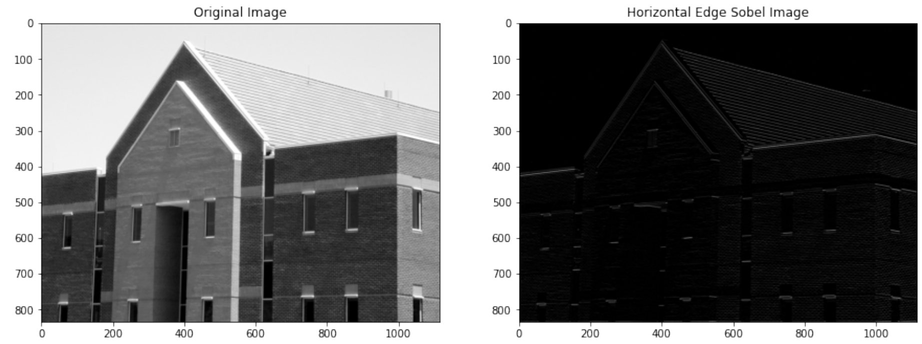
```
plt.imshow(image, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x1d2ae227fc8>
```



```
sh = image.shape

rows = sh[0]
cols = sh[1]

image_horz = image.copy()

sobel_horz = [[-1,-2,-1],[0,0,0],[1,2,1]]

image_horz = signal.convolve2d(image,sobel_horz,mode='same')

for r in range(rows):
    for c in range(cols):
        if image_horz[r][c]<0:
            image_horz[r][c]=0

#Display original and sobel image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(image, cmap = 'gray')
plt.title('Original Image')

plt.subplot(1,2,2)
plt.imshow(image_horz, cmap = 'gray')
plt.title('Horizontal Edge Sobel Image')
```

```
Text(0.5, 1.0, 'Horizontal Edge Sobel Image')
```



```
image_vert = image.copy()
```

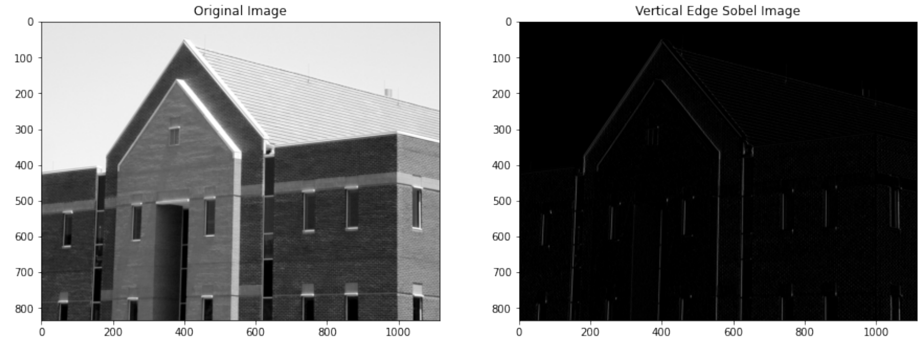
```
sobel_vert = [[-1,0,1],[-2,0,2],[-1,0,1]]
```

```
image_vert = signal.convolve2d(image,sobel_vert,mode='same')
```

```
for r in range(rows):
    for c in range(cols):
        if image_vert[r][c]<0:
            image_vert[r][c]=0
```

```
#Display original and sobel image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(image, cmap = 'gray')
plt.title('Original Image')

plt.subplot(1,2,2)
plt.imshow(image_vert, cmap = 'gray')
plt.title('Vertical Edge Sobel Image')
```

```
Text(0.5, 1.0, 'Vertical Edge Sobel Image')
```

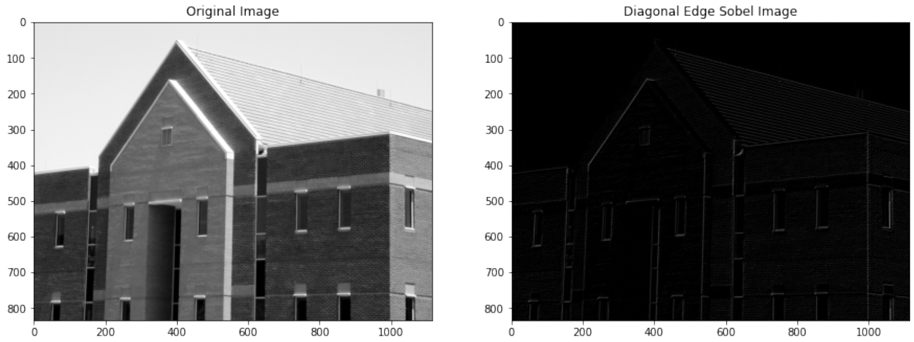


```
image_diag = image.copy()
```

```
sobel_diag = [[2,1,0],[1,0,-1],[0,-1,-2]]
```

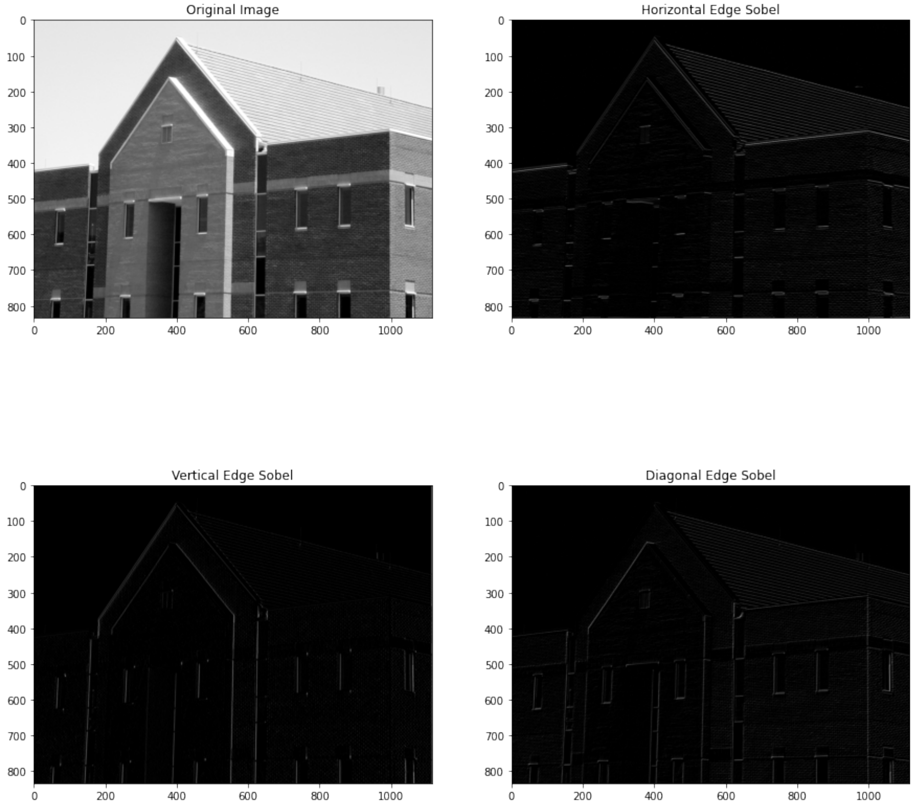
```
image_diag = signal.convolve2d(image,sobel_diag,mode='same')

for r in range(rows):
    for c in range(cols):
        if image_diag[r][c]<0:
            image_diag[r][c]=0

#Display original and sobel image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(image, cmap = 'gray')
plt.title('Original Image')

plt.subplot(1,2,2)
plt.imshow(image_diag, cmap = 'gray')
plt.title('Diagonal Edge Sobel Image')
```

```
Text(0.5, 1.0, 'Diagonal Edge Sobel Image')
```



```
plt.figure(figsize=(15,15))
plt.subplot(2,2,1)
plt.imshow(image,cmap='gray')
plt.title('Original Image')

plt.subplot(2,2,2)
plt.imshow(image_horz,cmap='gray')
plt.title('Horizontal Edge Sobel')

plt.subplot(2,2,3)
plt.imshow(image_vert, cmap = 'gray')
plt.title('Vertical Edge Sobel')

plt.subplot(2,2,4)
plt.imshow(image_diag, cmap = 'gray')
plt.title('Diagonal Edge Sobel')
```

```
Text(0.5, 1.0, 'Diagonal Edge Sobel')
```


Original Image

Horizontal Edge Sobel

Vertical Edge Sobel

Diagonal Edge Sobel

Laplacian Operator

```
#Import Image
image_color=io.imread('fruits.png')
```

```
image_color.shape
```

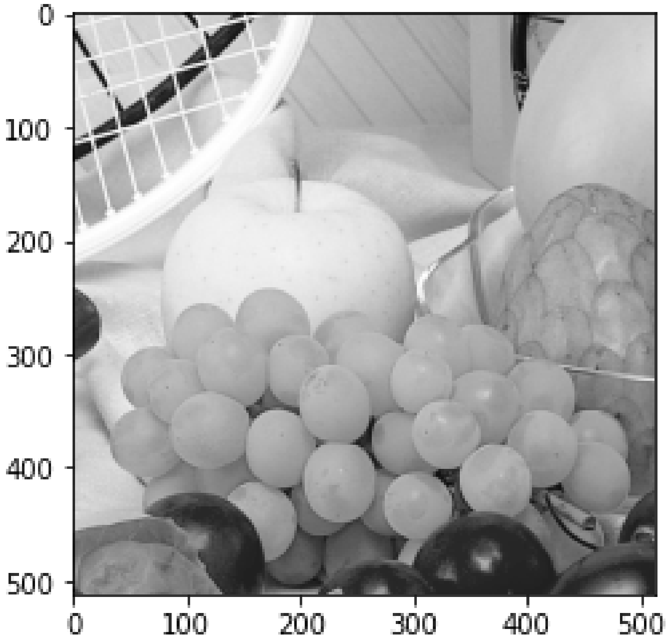
```
(512, 512, 3)
```

```
image = rgb2gray(image_color)
```

```
image=255*image
```

```
plt.imshow(image, cmap = 'gray')
```

```
<matplotlib.image.AxesImage at 0x1d2b18ed1c8>
```



```
image.shape
```

```
(512, 512)
```

```
sh = image.shape
```

```
rows = sh[0]
cols = sh[1]
```

```
image_laplace = image.copy()
```

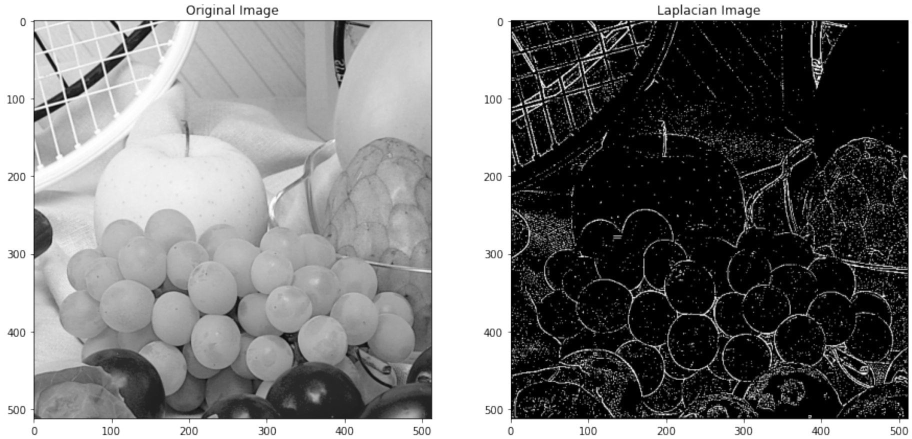
```
laplace = [[1,1,1],[1,-8,1],[1,1,1]]
```

```
image_laplace = signal.convolve2d(image,laplace,mode='same')
```

```
for r in range(rows):
    for c in range(cols):
        if image_laplace[r][c]>50:
            image_laplace[r][c] = 200
        else:
            image_laplace[r][c] = 0
```

```
#Display original and sobel image
plt.figure(figsize=(15,15))
plt.subplot(1,2,1)
plt.imshow(image, cmap = 'gray')
plt.title('Original Image')
plt.subplot(1,2,2)
plt.imshow(image_laplace, cmap = 'gray')
plt.title('Laplacian Image')
```

```
Text(0.5, 1.0, 'Laplacian Image')
```



Conclusion

- Sobel Filter is applied to detect horizontal, vertical and diagonal edges of the image
- Laplacian Operator is applied on the given image which detects all the types of edges. The threshold of 50 is used to identify the actual edges and to eliminate noisy edges