Computer Vision in HealthCare Application Lab-3

Topic: Edge Detection

Name: Sagnik Chatterjee

Reg.No: 19BAI1153

Dataset used: Skin Cancer HAM10000

Libraries

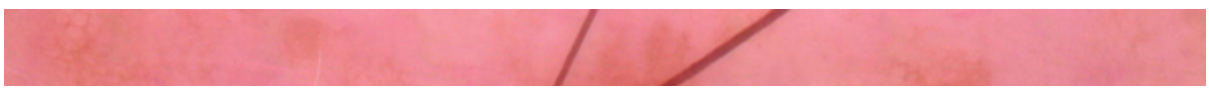
```
import cv2
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
import numpy as np
from skimage import io
plt.rcParams["figure.figsize"] = [8, 8]
plt.rcParams["figure.autolayout"] = True
```

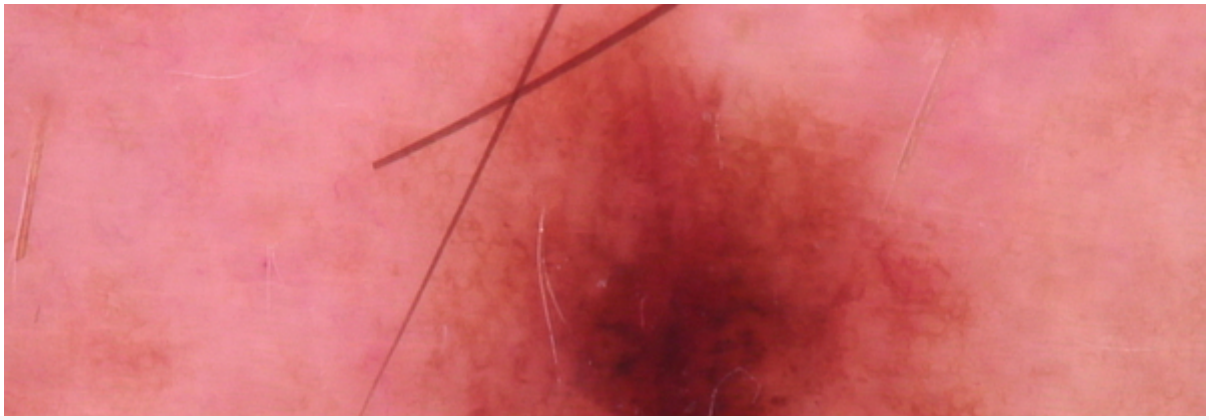
First we read Image and convert it to grayscale

Then we add various filter techniques to detect edges of the image

```
img = cv2.imread('/content/ISIC_0024306.jpg')
print("Shape of the image: ",img.shape)
print("Original Image: \n")
cv2_imshow(img)
```

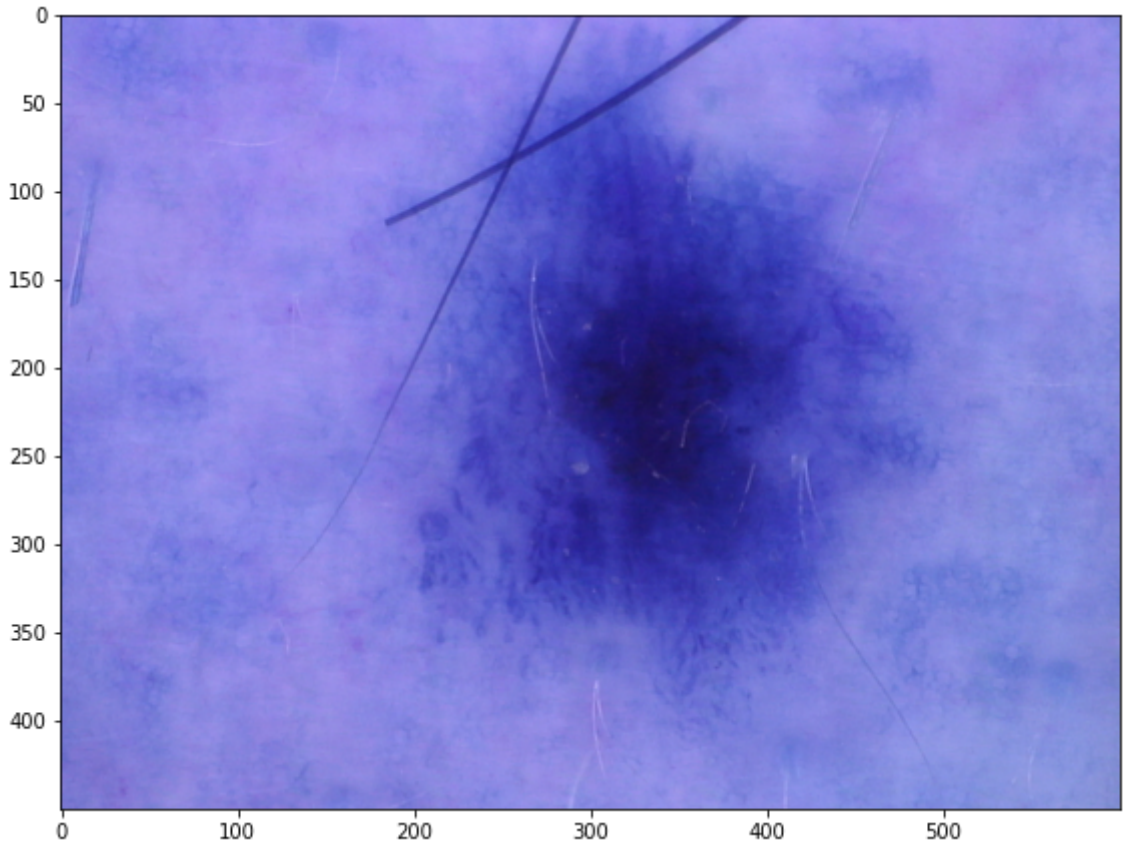
```
Shape of the image:  (450, 600, 3)
Original Image:
```





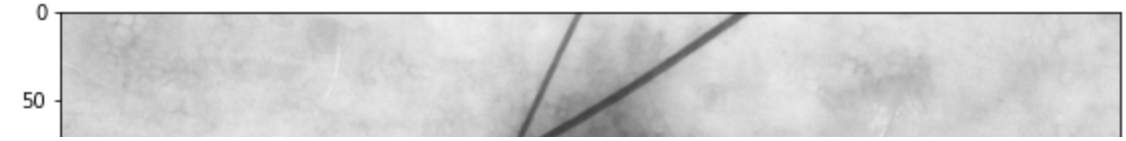
```
imgc = cv2.imread('/content/ISIC_0024306.jpg')
print(imgc.shape)
plt.imshow(imgc,cmap='gray')
```

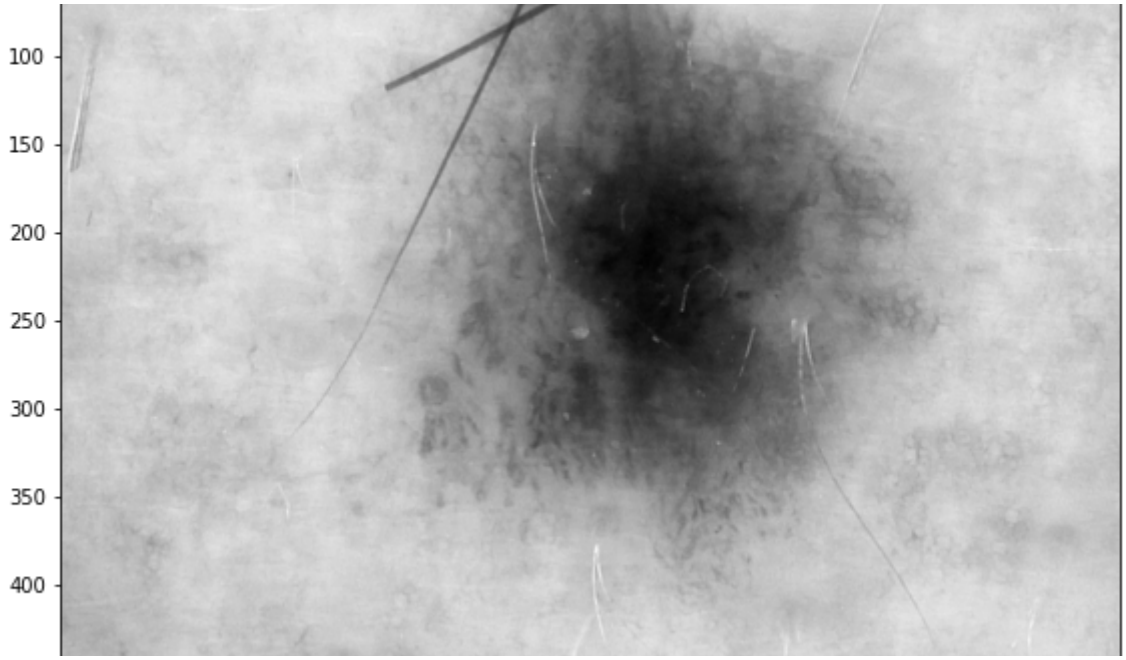
```
(450, 600, 3)
<matplotlib.image.AxesImage at 0x7f5f8934d250>
```




```
img_path = "/content/ISIC_0024306.jpg"

img = cv2.imread(img_path)
img_gs = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(img_gs, cmap = "gray")
```

```
<matplotlib.image.AxesImage at 0x7f5f8927f110>
```





Laplacian Filter

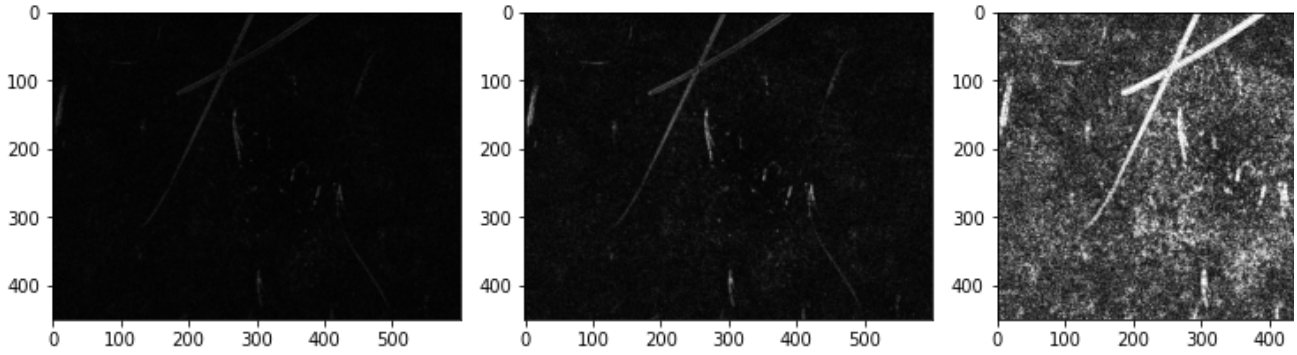
For color

```
laplacian = cv2.Laplacian(imgc,cv2.CV_64F)
io.imshow(laplacian)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
```

For Grayscale

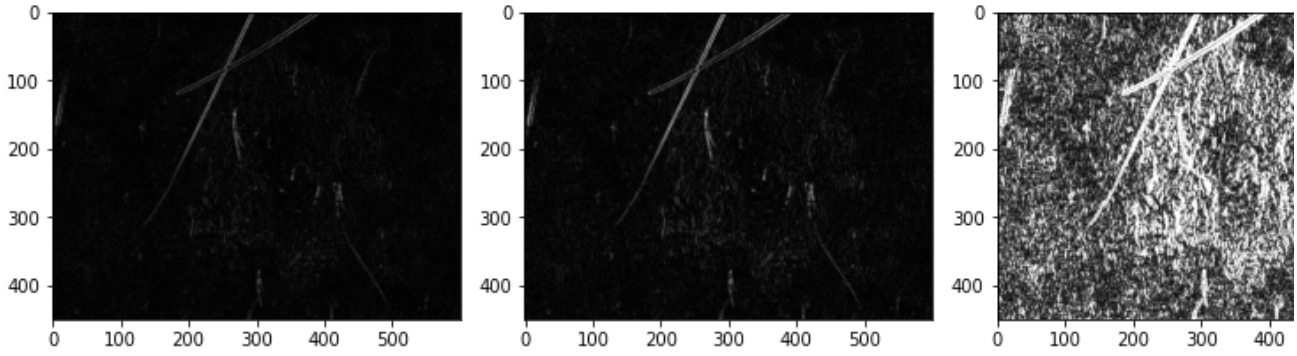
```
kernel_sizes = [1, 3, 5, 7]
plt.figure(figsize = (15, 15))
for i in range(len(kernel_sizes)):
  plt.subplot(1, 4, i+1)
  laplacian = cv2.Laplacian(img_gs, ksize=kernel_sizes[i], ddepth=cv2.CV_64F)
  laplacian = cv2.convertScaleAbs(laplacian)
  plt.imshow(laplacian,cmap='gray')
```



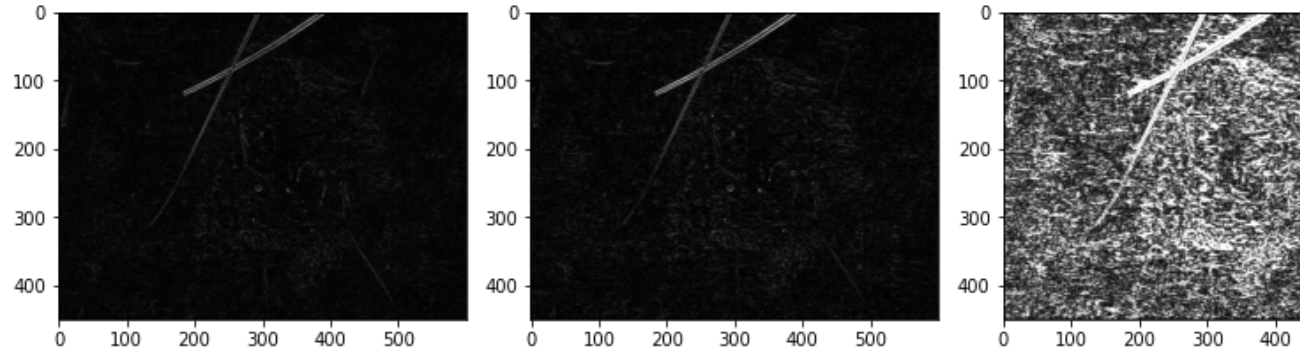
Sobel Filter

We see for different kernel sizes as well as for different dx and dy values.

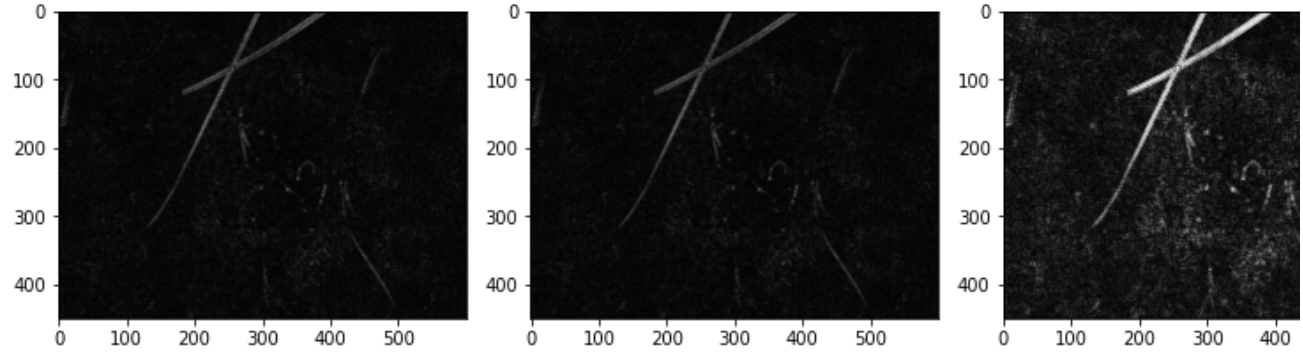
```
kernel_sizes = [1, 3, 5, 7]
plt.figure(figsize = (15, 15))
for i in range(len(kernel_sizes)):
  sobelx = cv2.Sobel(src=img_gs, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=kernel_sizes[i])
  sobelx = cv2.convertScaleAbs(sobelx)
  plt.subplot(1, 4, i+1)
  plt.imshow(sobelx, cmap = "gray")
```



```
kernel_sizes = [1, 3, 5, 7]
plt.figure(figsize = (15, 15))

for i in range(len(kernel_sizes)):
  sobelx = cv2.Sobel(src=img_gs, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=kernel_sizes[i])
  sobelx = cv2.convertScaleAbs(sobelx)
  plt.subplot(1, 4, i+1)
  plt.imshow(sobelx, cmap = "gray")
```



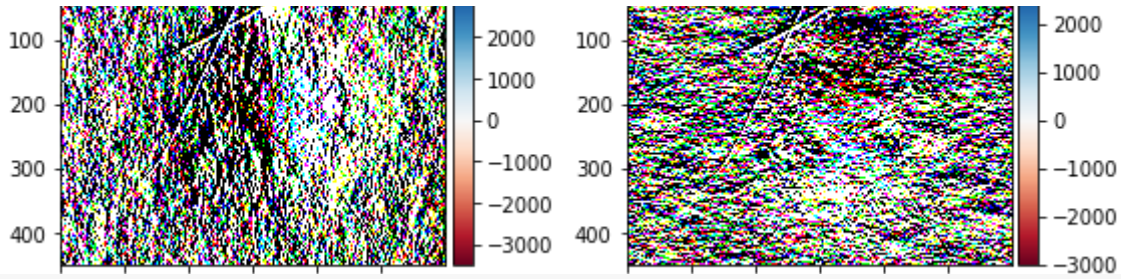
```
kernel_sizes = [1, 3, 5, 7]
plt.figure(figsize = (15, 15))
for i in range(len(kernel_sizes)):
  sobelx = cv2.Sobel(src=img_gs, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=kernel_sizes[i])
  sobelx = cv2.convertScaleAbs(sobelx)
  plt.subplot(1, 4, i+1)
  plt.imshow(sobelx, cmap = "gray")
```



For Color image

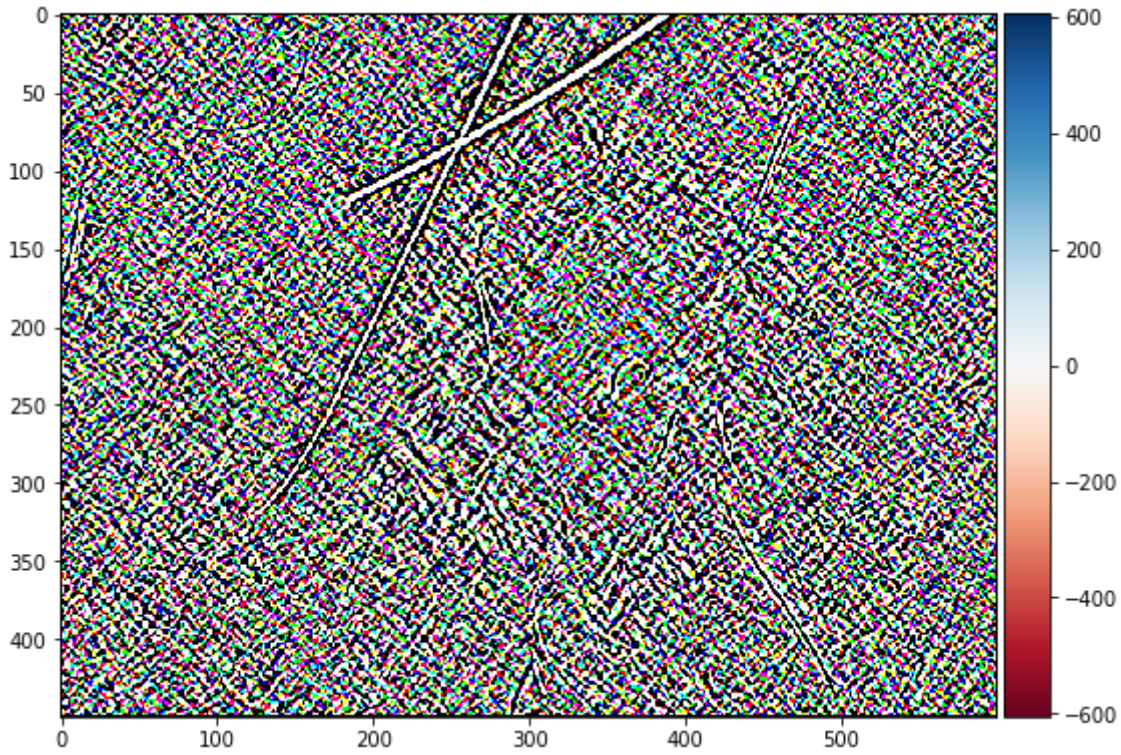
```
sobelx = cv2.Sobel(imgc, cv2.CV_64F, 1, 0, ksize = 5)
sobely = cv2.Sobel(imgc, cv2.CV_64F, 0, 1, ksize = 5)
sobelxy = cv2.Sobel(imgc, cv2.CV_64F, 1, 1, ksize = 5)

plt.subplot(1,2,1)
io.imshow(sobelx)
plt.title("Sobel x")
plt.subplot(1,2,2)
io.imshow(sobely)
plt.title("Sobel y")
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
Text(0.5, 1.0, 'Sobel y')
```



```
io.imshow(sobelxy)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or
<matplotlib.image.AxesImage at 0x7f5f88cd2490>
```

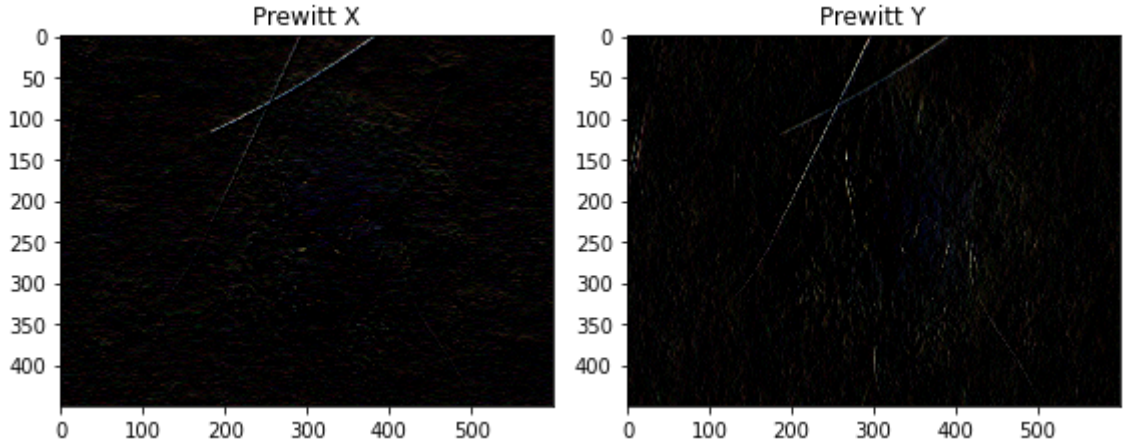


Prewitt Filter

For color

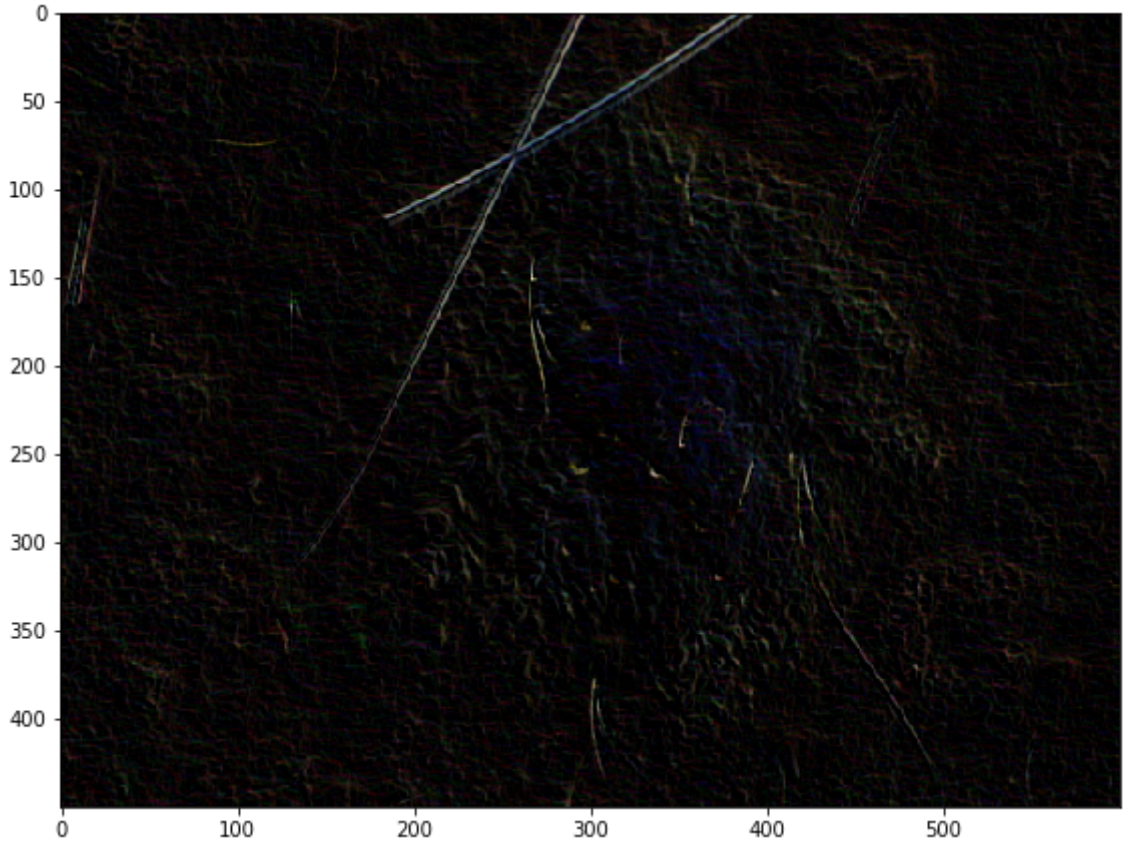
```
#prewitt
kernel_horizontal = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
kernel_vertical = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
img_prewittx = cv2.filter2D(imgc, -1, kernel_horizontal)
img_prewitty = cv2.filter2D(imgc, -1, kernel_vertical)

plt.subplot(1,2,1)
io.imshow(img_prewittx)
plt.title("Prewitt X")
plt.subplot(1,2,2)
io.imshow(img_prewitty)
plt.title("Prewitt Y")
```

```
Text(0.5, 1.0, 'Prewitt Y')
```




```
io.imshow(img_prewittx+img_prewitty)
```

```
<matplotlib.image.AxesImage at 0x7f5f88aedb10>
```

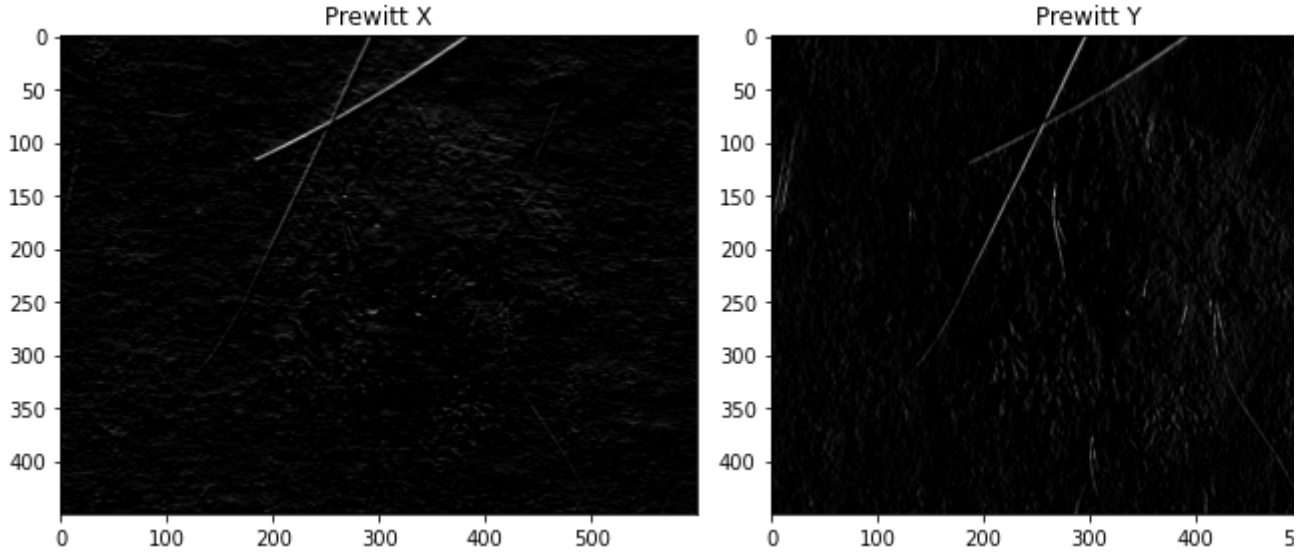


For Grayscale

```
kernel_horizontal = np.array([[1,1,1],[0,0,0],[-1,-1,-1]])
kernel_vertical = np.array([[-1,0,1],[-1,0,1],[-1,0,1]])
img_prewittx = cv2.filter2D(img_gs, -1, kernel_horizontal)
img_prewitty = cv2.filter2D(img_gs, -1, kernel_vertical)

plt.figure(figsize=(10, 7))
plt.subplot(1,2,1)
plt.imshow(img_prewittx,cmap='gray')
plt.title("Prewitt X")
plt.subplot(1,2,2)
```

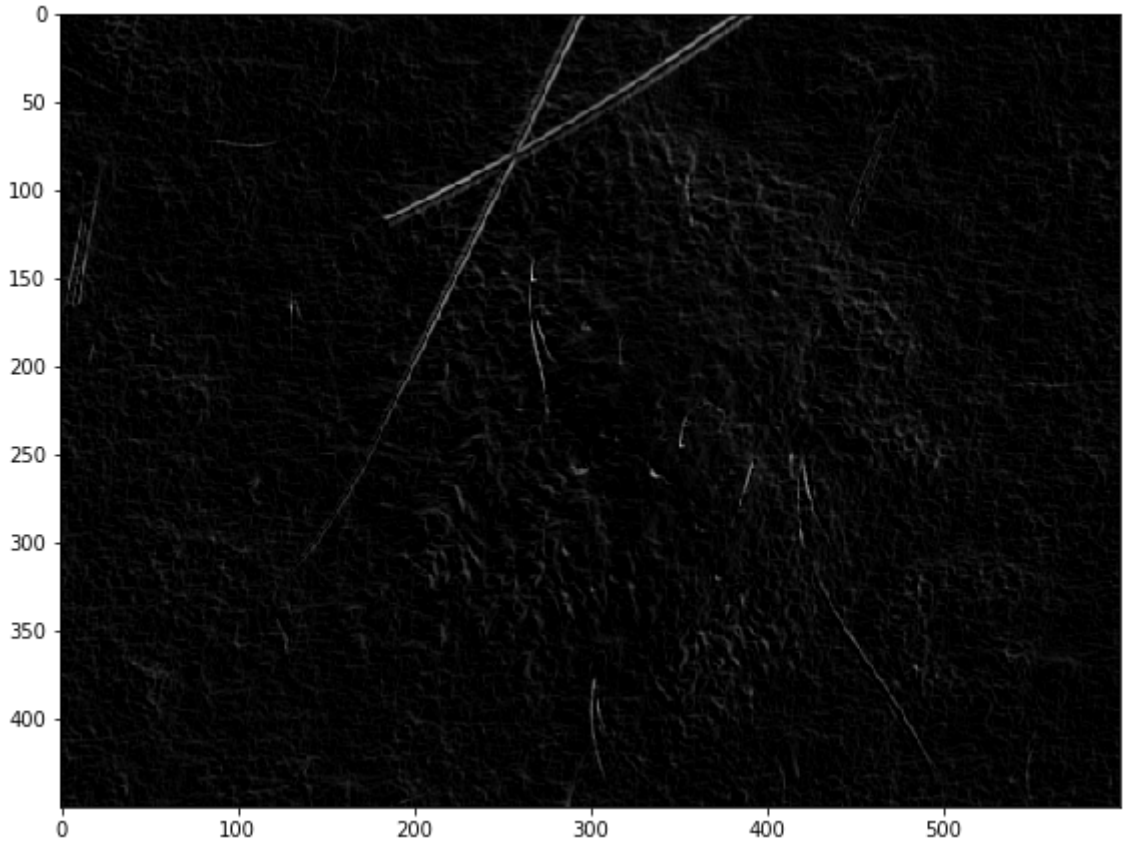
```
plt.imshow(img_prewitty,cmap='gray')
plt.title("Prewitt Y")
```

```
Text(0.5, 1.0, 'Prewitt Y')
```




```
io.imshow(img_prewittx+img_prewitty)
```

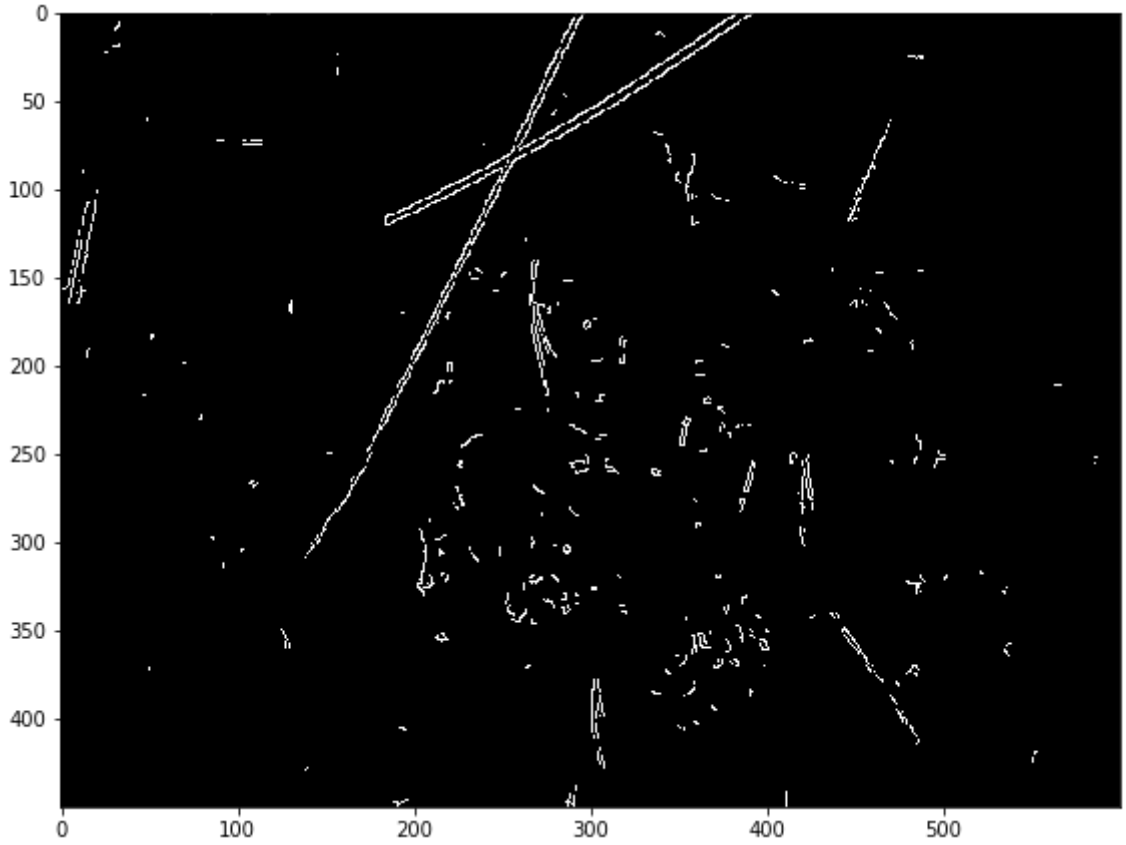
```
<matplotlib.image.AxesImage at 0x7f5f88a5e150>
```



Canny Edge Filter

```
canny = cv2.Canny(imgc,75,100)
io.imshow(canny)
```

```
<matplotlib.image.AxesImage at 0x7f5f8892a250>
```

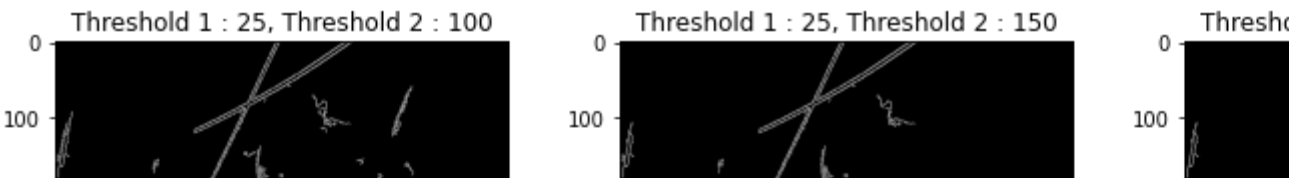


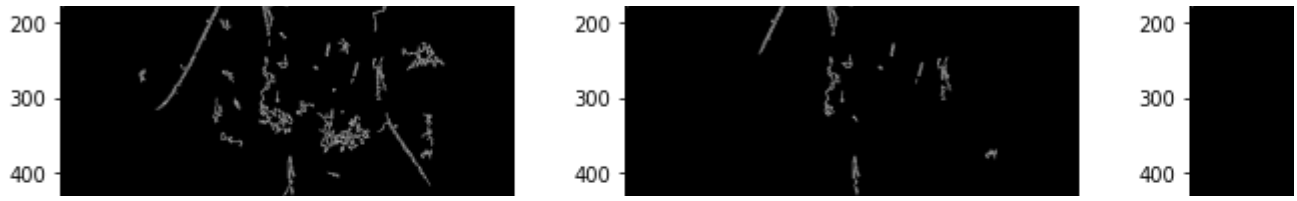

```
canny = cv2.Canny(img_gs,75,100)
io.imshow(canny)
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150:
  lo, hi, cmap = _get_display_range(image)
<matplotlib.image.AxesImage at 0x7f5f8890ad50>
```

Now let's see how it works with two diffrent thresholds.

```
l1 = [25, 50, 75, 100]
l2 = [100, 150, 200]
plt.figure(figsize = (12, 12))
ctr = 1
for i in range(len(l1)):
  for j in range(len(l2)):
    plt.subplot(len(l1), len(l2), ctr)
    canny_img = cv2.Canny(image=img_gs, threshold1 = l1[i],threshold2 = l2[j])
    plt.imshow(canny_img, cmap = "gray")
    plt.title(f"Threshold 1 : {l1[i]}, Threshold 2 : {l2[j]}")
    ctr += 1
```

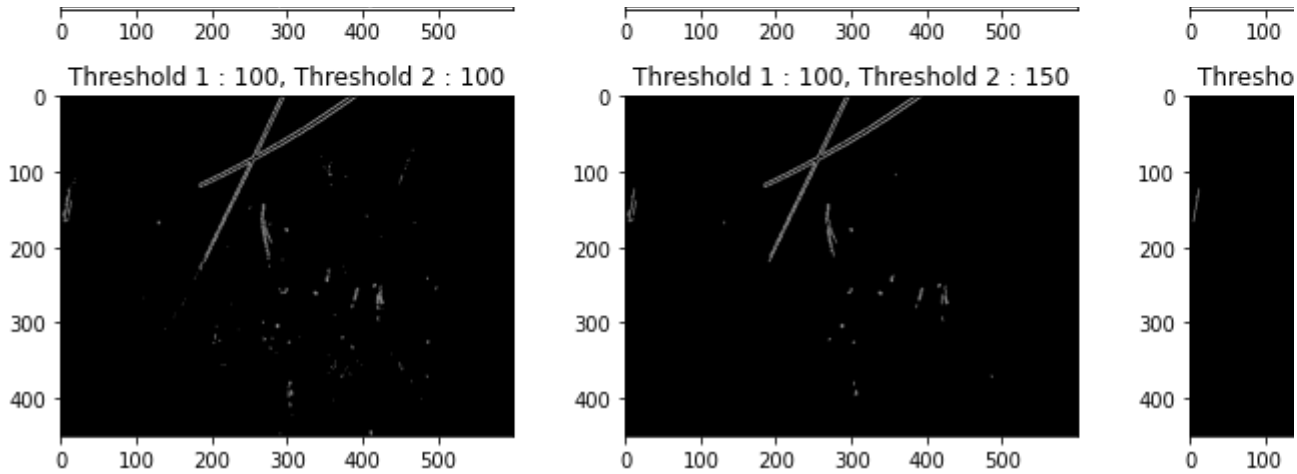




Conclusion:

- **Laplacian Edge Detection** gave pixel like output for color image and very light output for grayscale image.The kernel size 5 gave the best edges compared to others.
- **Sobel Edge Detection** technique detects edges better for color image than grayscale image.In grayscale 5 and 7 gave good results for dx=dy=1.
- **Prewitt Edge Detection** technique gives good result with both color and grayscale image but comparing it with naked eye, canny can be said to perform better. It gives off a glossy shade/elevation over the edges and its interior in our image.But it's way darker than others.
- **Canny Edge Detection** technique detects edges better for colour image than grayscale image (comparing results with naked eye).It gives more details for the colored ones. We also saw the results by passing different thresholds to the canny filter.

So for Grayscaled image, sobel (x-axis, y-axis kernels), canny, Prewitt performs better. For color image, sobel (x-axis, y-axis kernels), canny, Prewitt performs better.



✓ 2s completed at 10:50 PM