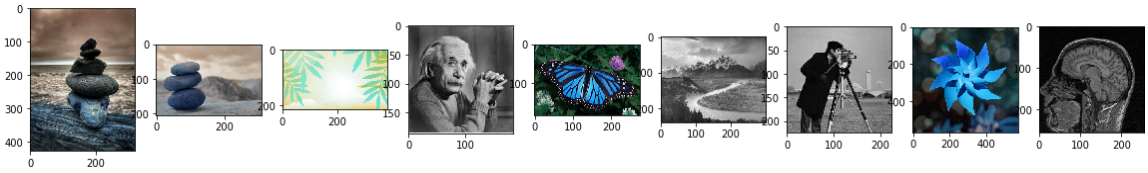
In [1]:

```
# -*- coding: utf-8 -*-
"""
@author: UzairAzhar
"""
#helper functions in helper_functions.py file
from helper_functions import load_images_from_folder
from helper_functions import plot_image
from helper_functions import hist_equalization
from helper_functions import rgbExclusion
from helper_functions import convolution
from helper_functions import box_filter_conv
from helper_functions import gaussian_filter_conv
from helper_functions import add_noise
from helper_functions import gaussianFilter_and_medianFilters
from helper_functions import mesh_plots
from helper_functions import sobel
from helper_functions import laplacian
from helper_functions import canny
```

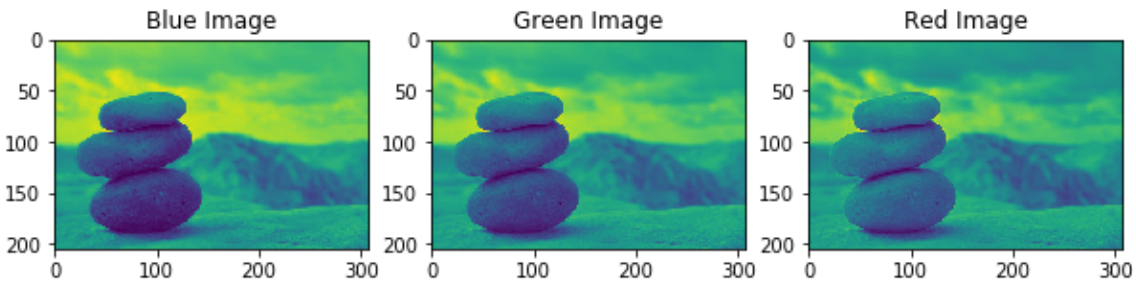
In [2]:

```
#2.1
#Add path where images are and plot them
path_of_img_dir=(r"images")
images=load_images_from_folder(path_of_img_dir)
plot_image(images)
```



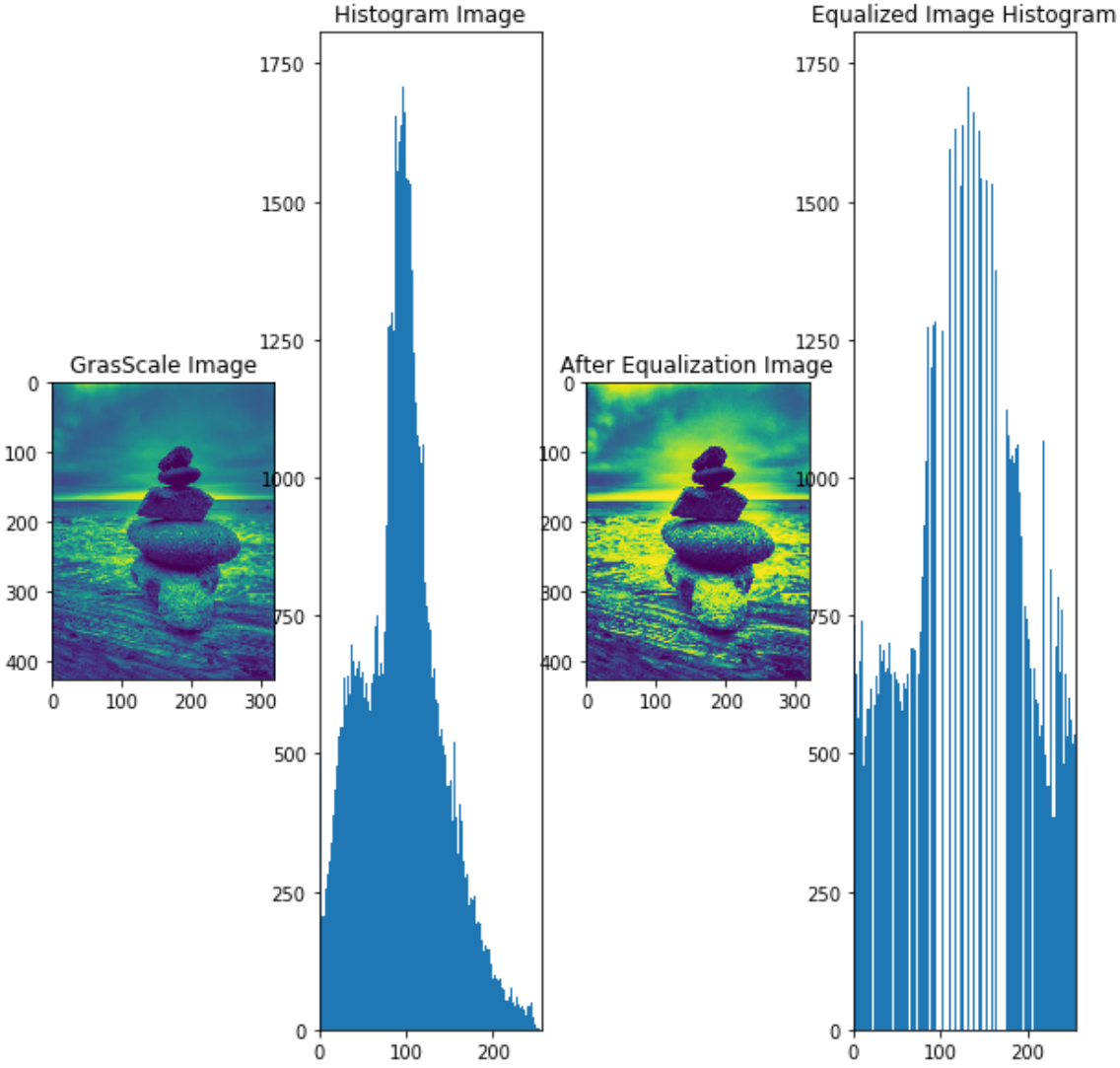
In [3]:

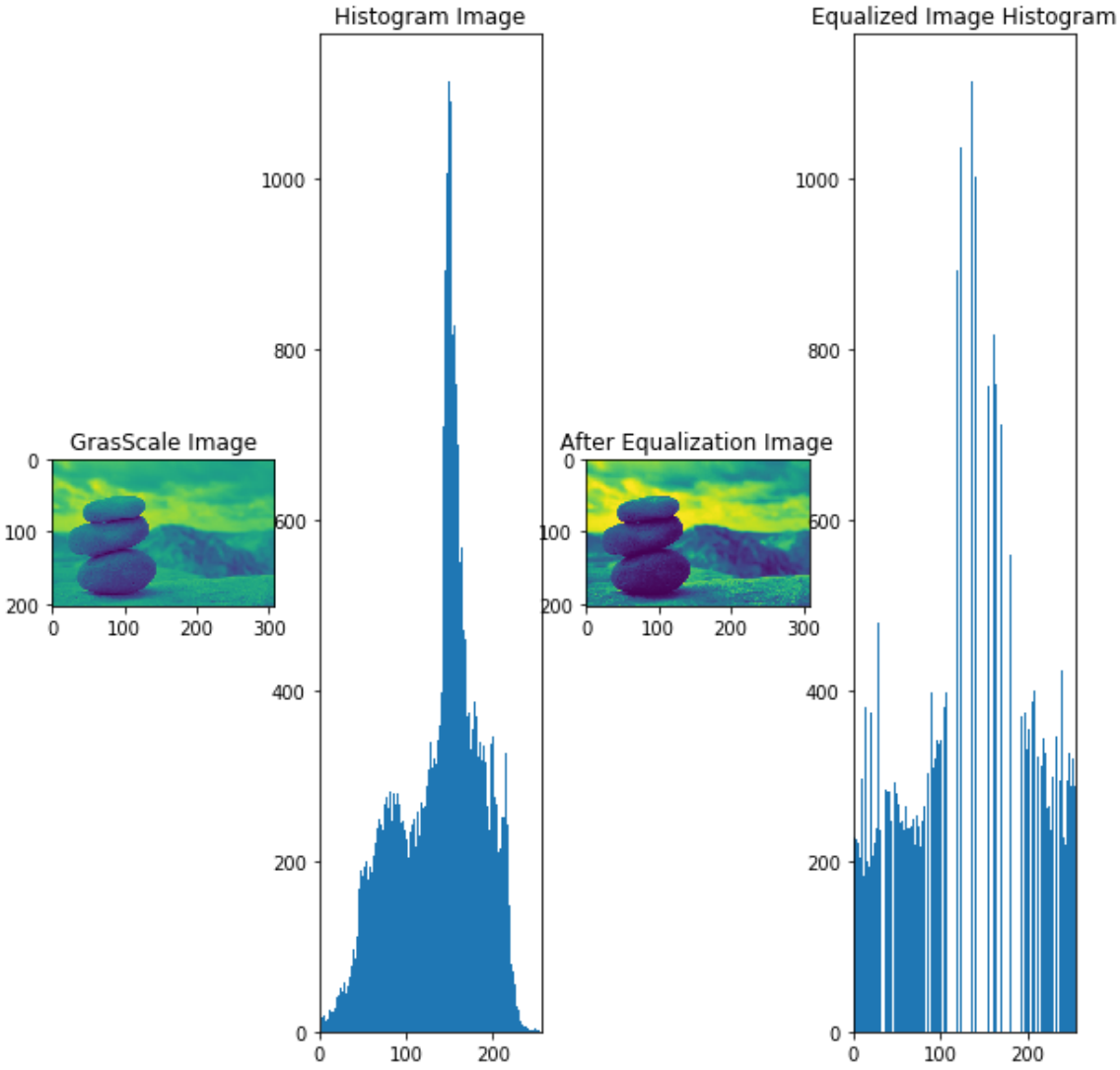
```
# In[] : 2.2 #rgbExclusion
# RGB Exclusion done for any particular Image
image_number=1 # 2nd image in dir has been selected for rgbExclusion
[B,G,R]=rgbExclusion(images[image_number])
```

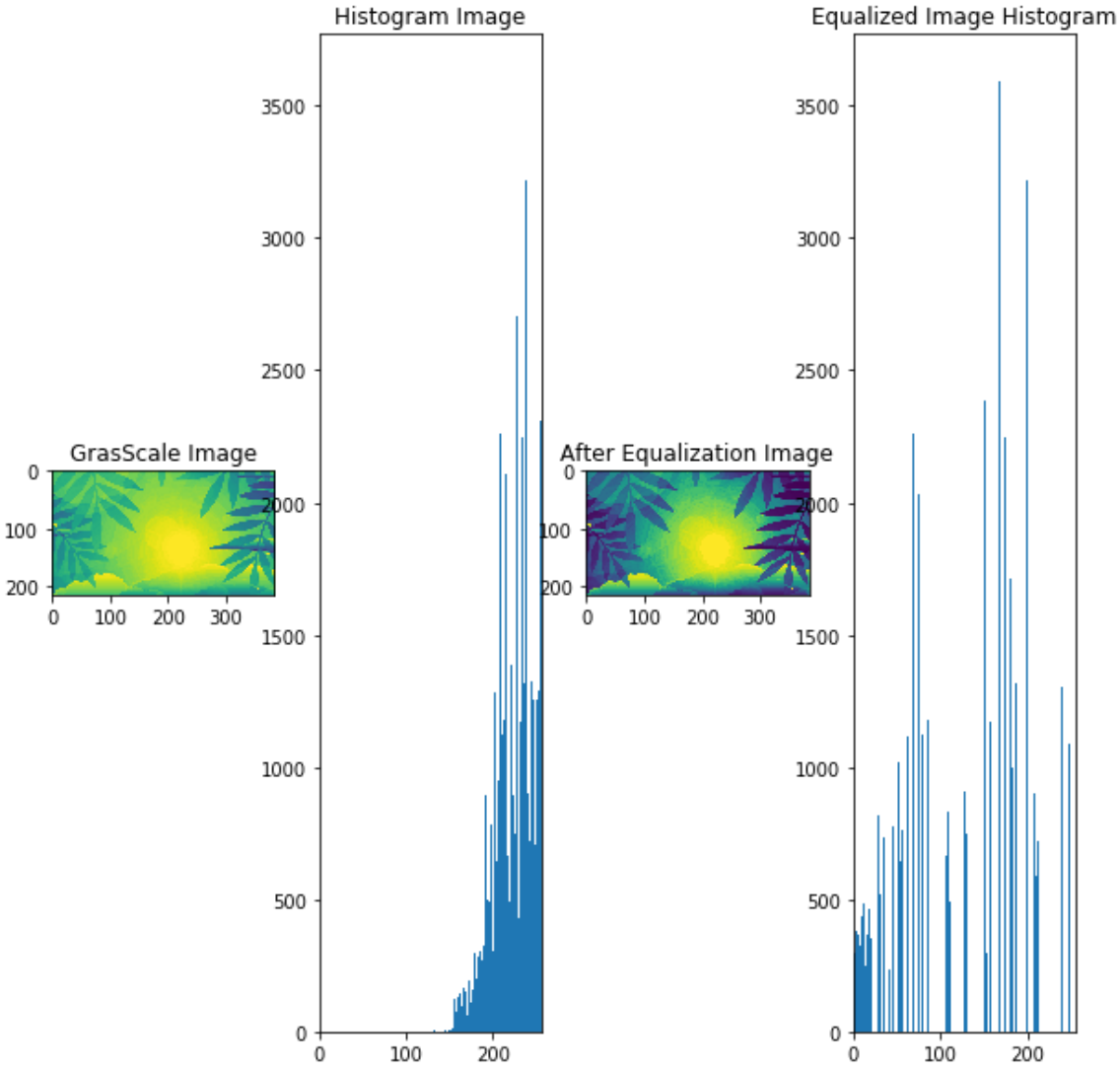


In [4]:

```
# In[] : 2.3 # Equalization
#In this function first histogram of orignal image has been drawn and then after equali
zation, both image and histogram is drawn
hist_equalization(images)
```


GrasScale Image
Histogram Image
After Equalization Image
Equalized Image Histogram

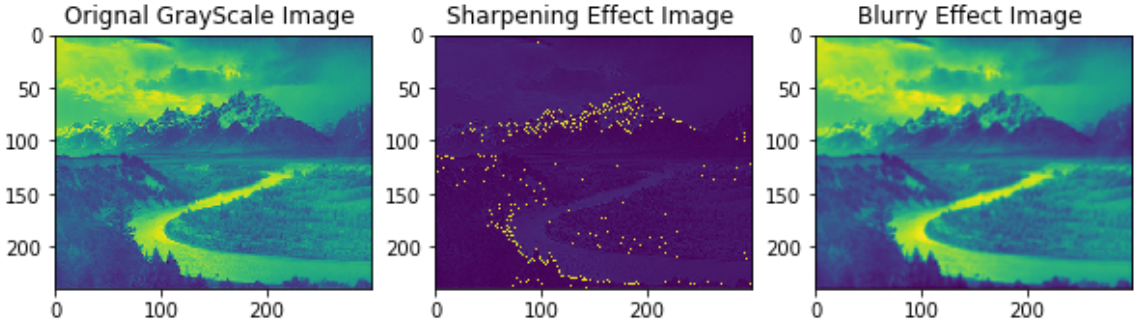

GrasScale Image
Histogram Image
After Equalization Image
Equalized Image Histogram



In [5]:

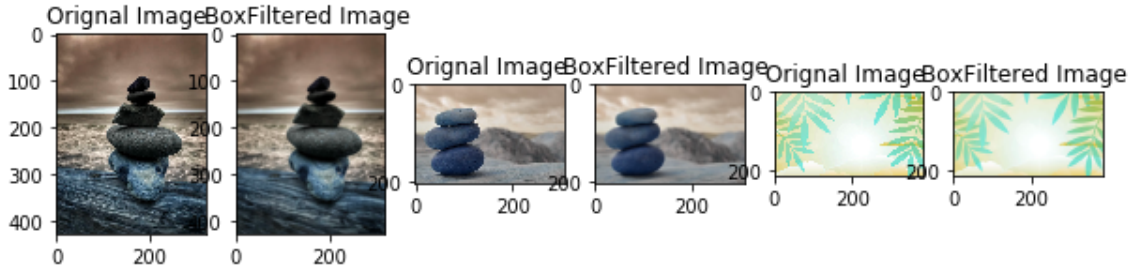
```
# In[] : 2.4 # Convolution
# here we have done convolution with sharpening filter and blurry filter(box) using own implementation of
#conv and compared it with built in implementation
image=images[5] # image 6 in dir has been selected for convolution implementation purpose
conv_img_diff=convolution(image)
print('Diff in Images = %d\nHence both Manual and Libs Convolution results are same'%conv_img_diff)
```

```
Diff in Images = 0
Hence both Manual and Libs Convolution results are same
```



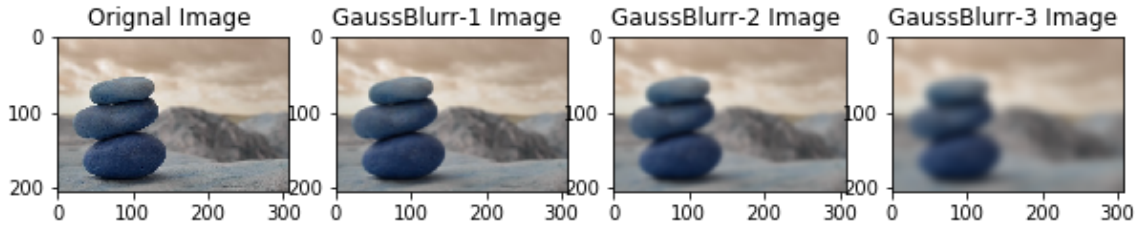
In [7]:

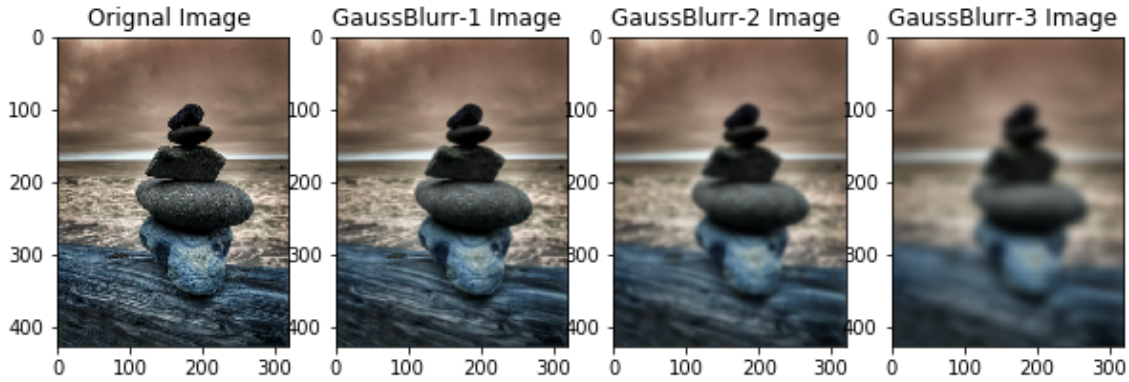
```
# In[] : 2.5 # Image Processing Operations
#2.51
# here box filter has been applied on any 3 images
box_filter_conv(images)
```



In [8]:

```
#2.52
# here gaussian filter has been applied on two images with varying kernel size to vary
 sigma
gaussian_filter_conv(images)
```

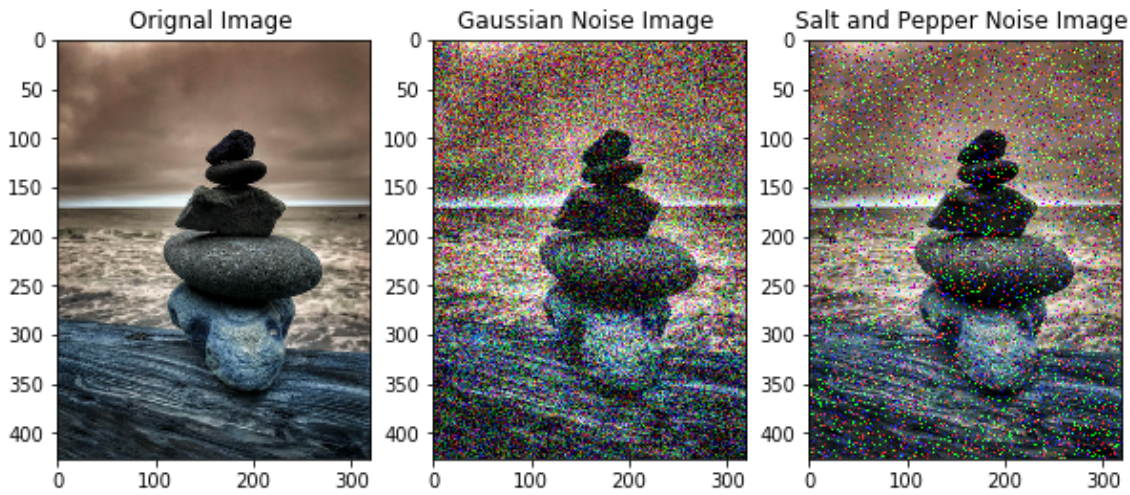




In [9]:

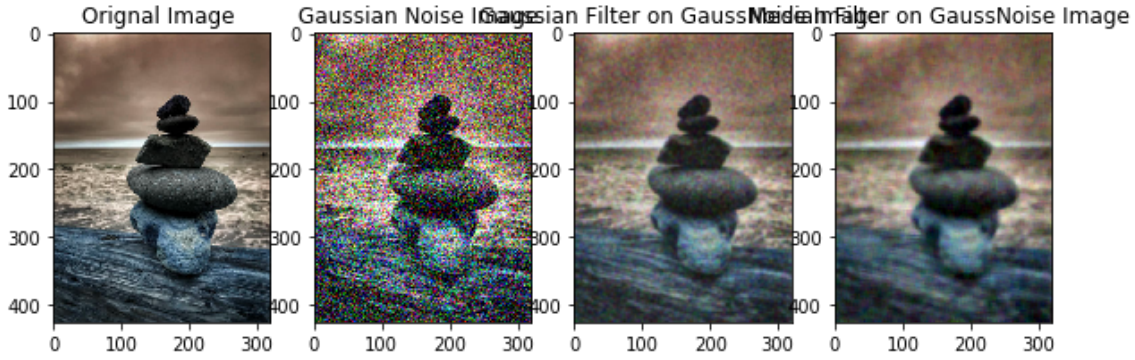
```
#2.53
# Here on all image s
# 1) Gaussian Noise
# 2) Salt and Pepper
# has been added, but only one image has been displayed, rest are just output for furth
er use
[sp_noise_images,gauss_noise_images]=add_noise(images)
```

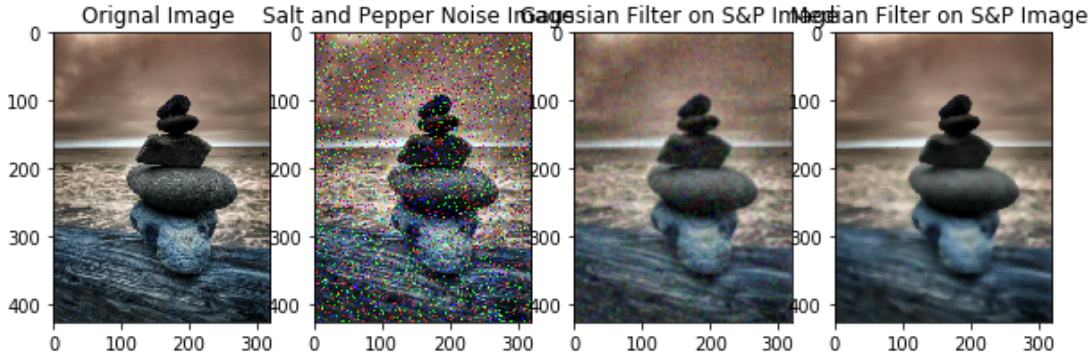
```
C:\Users\uzair\Documents\MuhammadUzair-Azhar\helper_functions.py:231: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as a pre
vious axes currently reuses the earlier instance.  In a future version, a
new instance will always be created and returned.  Meanwhile, this warning
can be suppressed, and the future behavior ensured, by passing a unique la
bel to each axes instance.
  ax=plt.subplot(131)
C:\Users\uzair\Documents\MuhammadUzair-Azhar\helper_functions.py:235: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as a pre
vious axes currently reuses the earlier instance.  In a future version, a
new instance will always be created and returned.  Meanwhile, this warning
can be suppressed, and the future behavior ensured, by passing a unique la
bel to each axes instance.
  ax=plt.subplot(132)
C:\Users\uzair\Documents\MuhammadUzair-Azhar\helper_functions.py:239: Matp
lotlibDeprecationWarning: Adding an axes using the same arguments as a pre
vious axes currently reuses the earlier instance.  In a future version, a
new instance will always be created and returned.  Meanwhile, this warning
can be suppressed, and the future behavior ensured, by passing a unique la
bel to each axes instance.
  ax=plt.subplot(133)
```



In [48]:

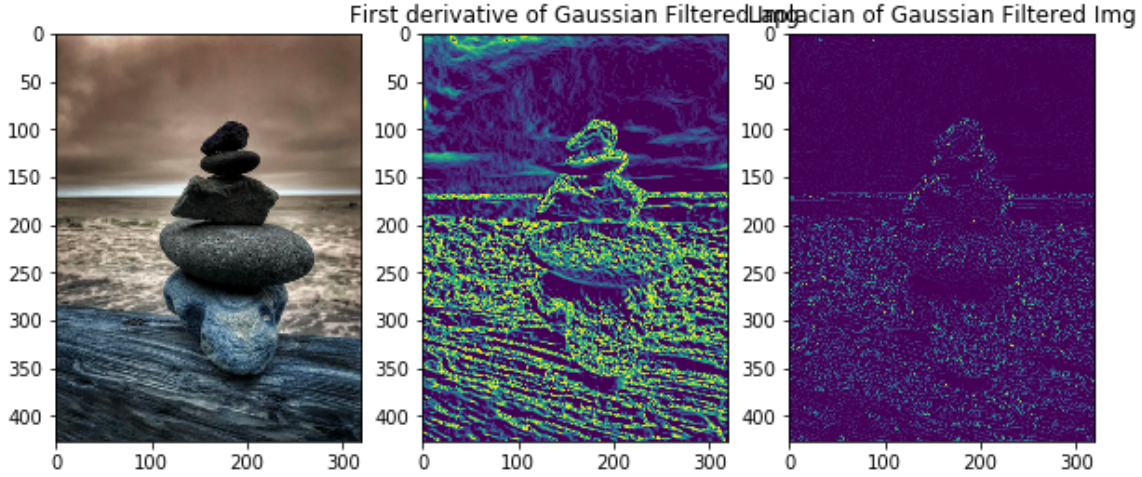
```
#2.54
gaussianFilter_and_medianFilters(images,sp_noise_images,gauss_noise_images)
```

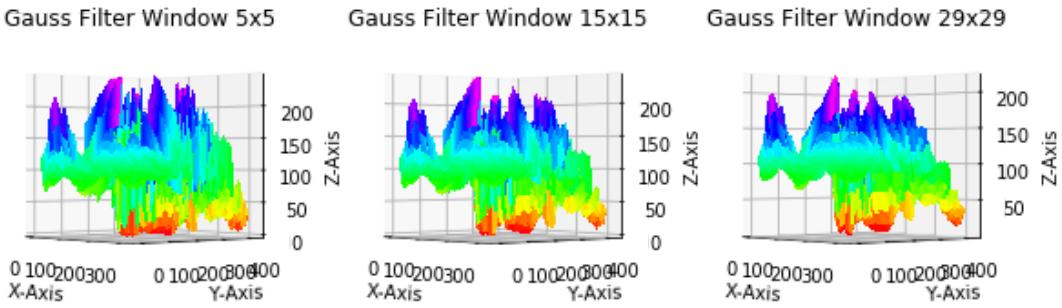


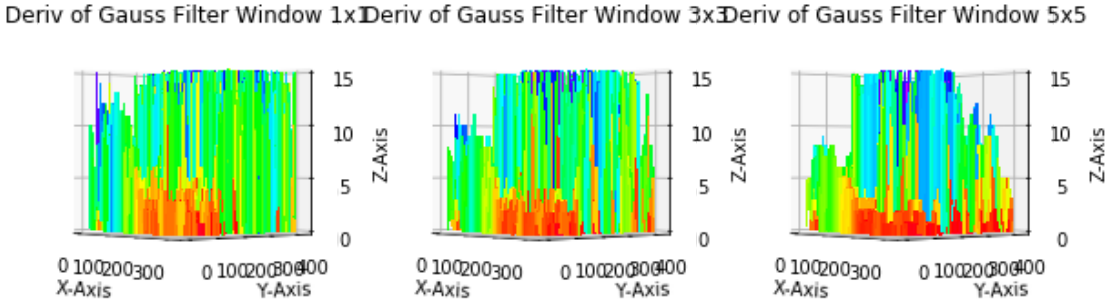


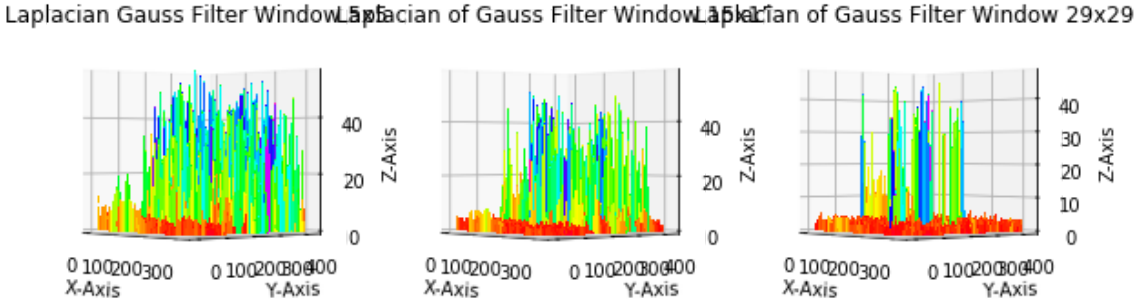
In [10]:

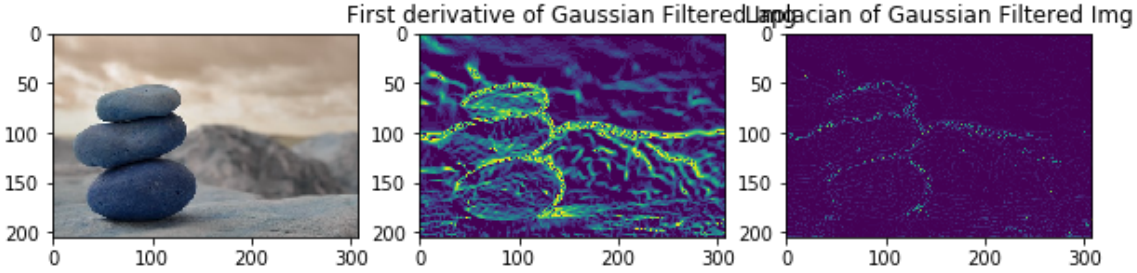
```
#2.55
# here mesh plots of all images has been generated and plotted that includes
#1)Gaussian Filter
#2) First deriv of Gaussian Filter
#3)Second deriv of Gaussian Filter
mesh_plots(images)
```

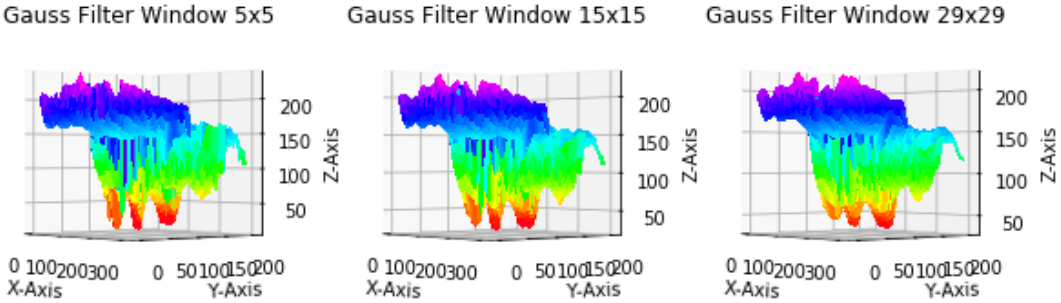

First derivative of Gaussian Filtered Img
Laplacian of Gaussian Filtered Img

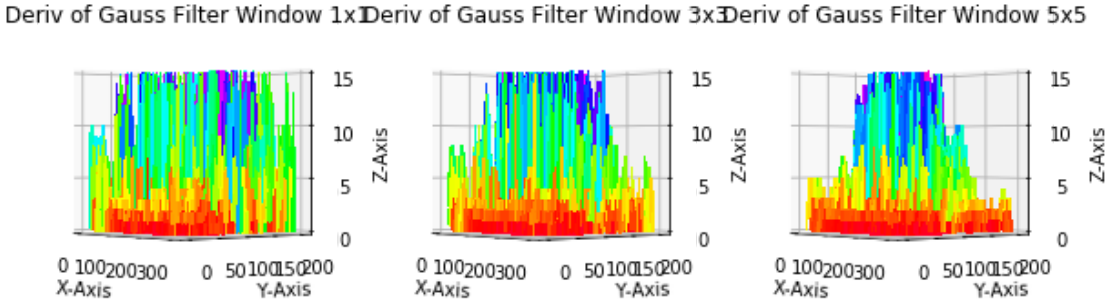

Gauss Filter Window 5x5
Gauss Filter Window 15x15
Gauss Filter Window 29x29
X-Axis
Y-Axis
Z-Axis

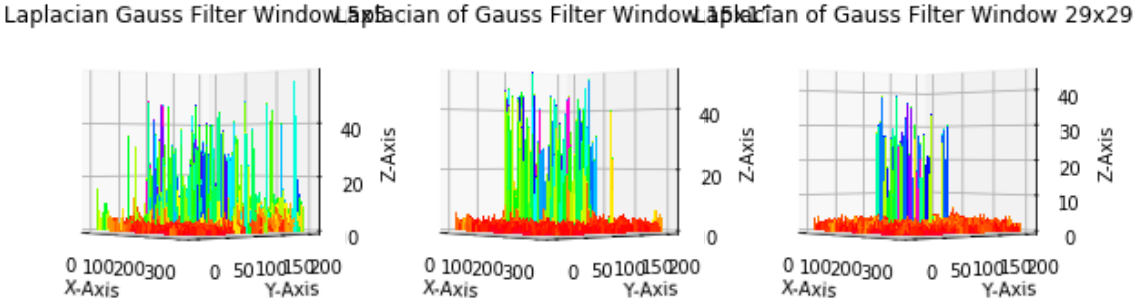

Deriv of Gauss Filter Window 1x1
Deriv of Gauss Filter Window 3x3
Deriv of Gauss Filter Window 5x5
X-Axis
Y-Axis
Z-Axis

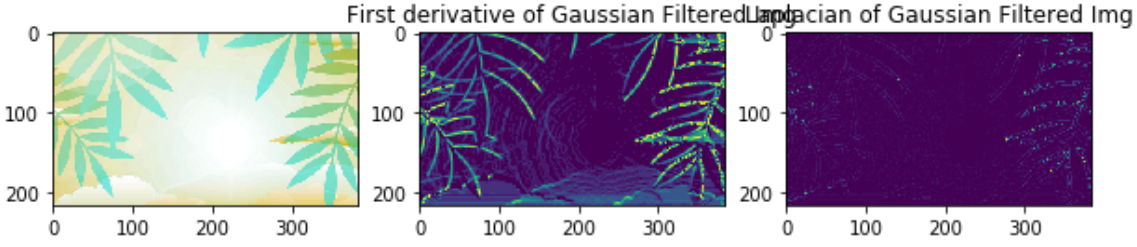

Laplacian Gauss Filter Window 5x5
Laplacian of Gauss Filter Window 15x15
Laplacian of Gauss Filter Window 29x29
X-Axis
Y-Axis
Z-Axis

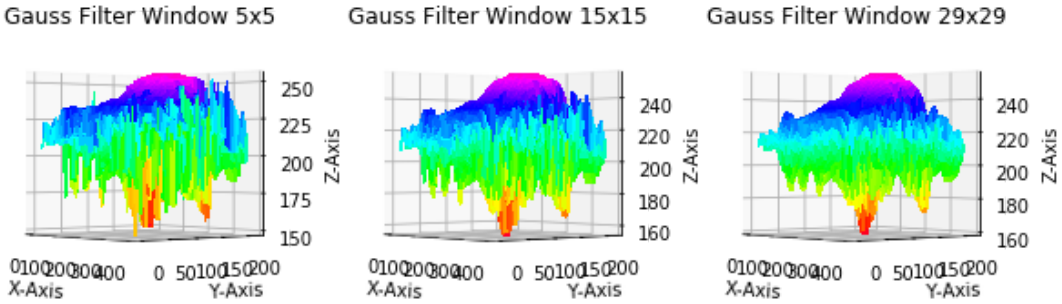

First derivative of Gaussian Filtered Img
Laplacian of Gaussian Filtered Img

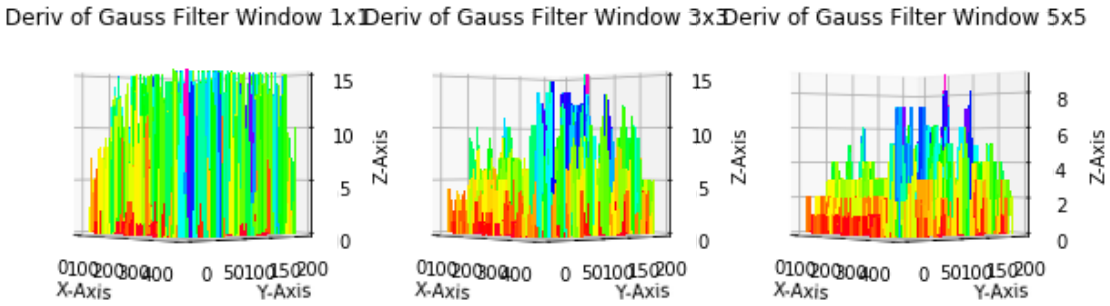

Gauss Filter Window 5x5
Gauss Filter Window 15x15
Gauss Filter Window 29x29
Z-Axis
X-Axis
Y-Axis

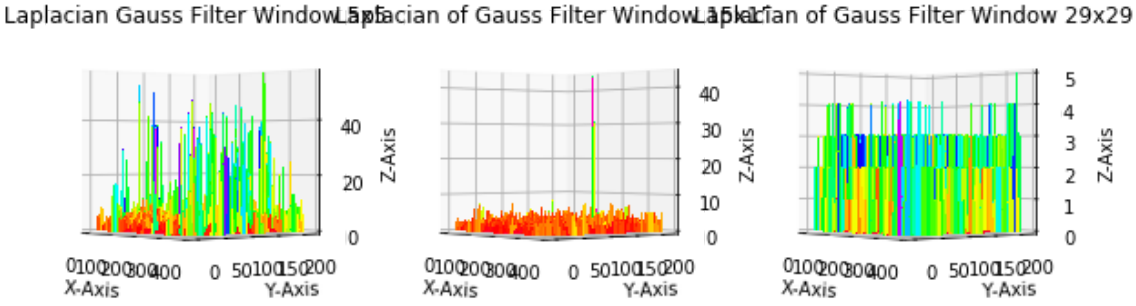

Deriv of Gauss Filter Window 1x1
Deriv of Gauss Filter Window 3x3
Deriv of Gauss Filter Window 5x5
Z-Axis
X-Axis
Y-Axis

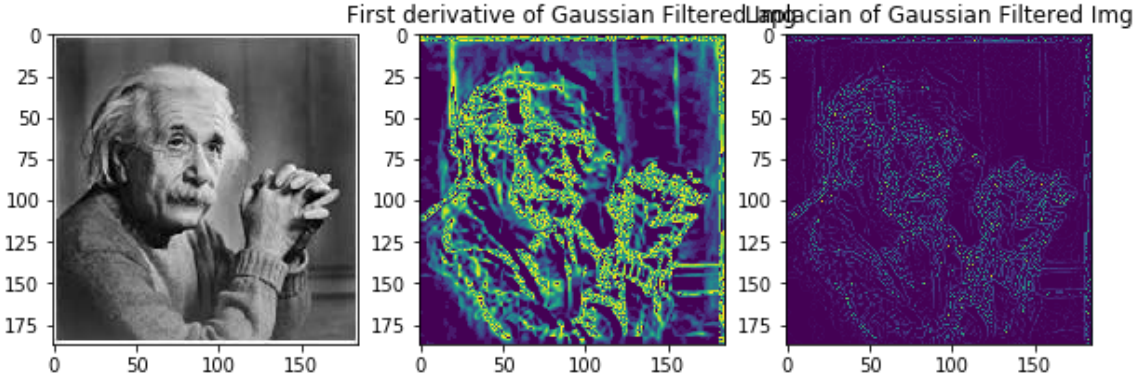

Laplacian Gauss Filter Window 5x5
Laplacian of Gauss Filter Window 15x15
Laplacian of Gauss Filter Window 29x29
Z-Axis
X-Axis
Y-Axis

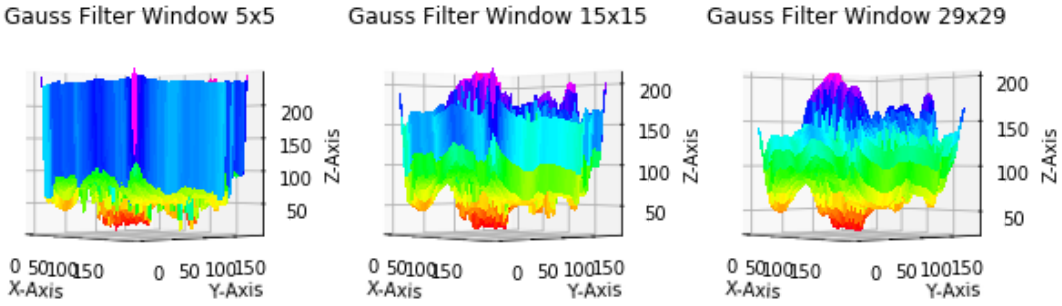

First derivative of Gaussian Filtered Img
Laplacian of Gaussian Filtered Img

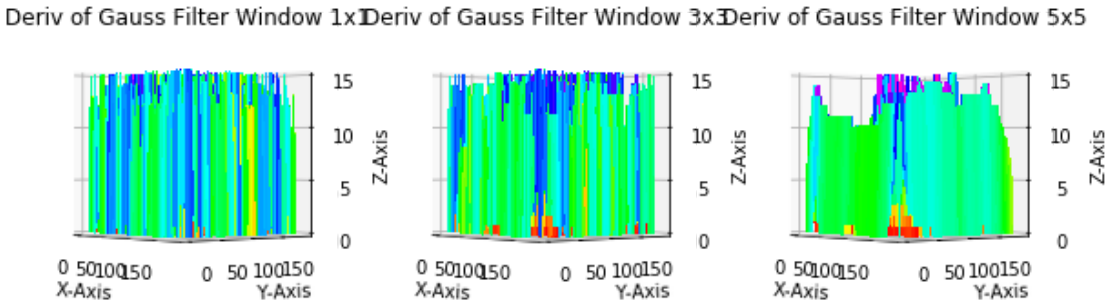

Gauss Filter Window 5x5
Gauss Filter Window 15x15
Gauss Filter Window 29x29
X-Axis
Y-Axis
Z-Axis

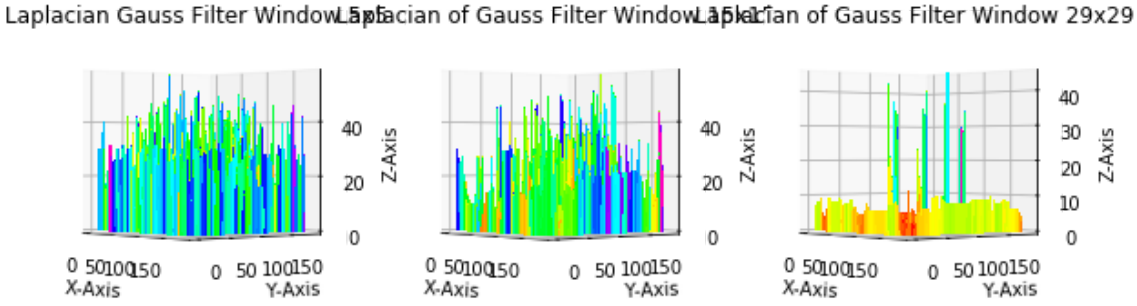

Deriv of Gauss Filter Window 1x1
Deriv of Gauss Filter Window 3x3
Deriv of Gauss Filter Window 5x5
X-Axis
Y-Axis
Z-Axis

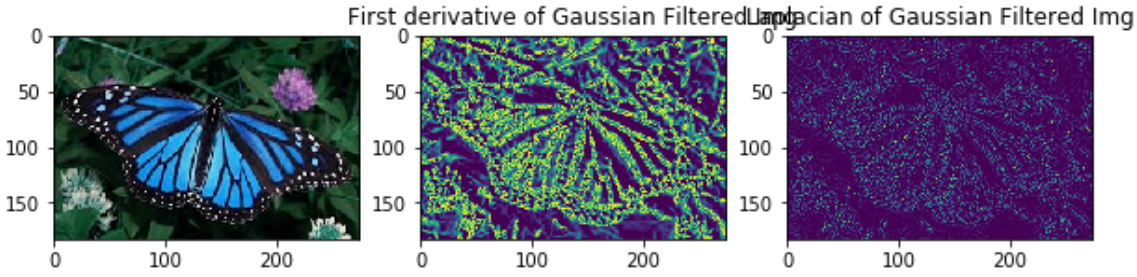

Laplacian Gauss Filter Window 5x5
Laplacian of Gauss Filter Window 15x15
Laplacian of Gauss Filter Window 29x29
X-Axis
Y-Axis
Z-Axis

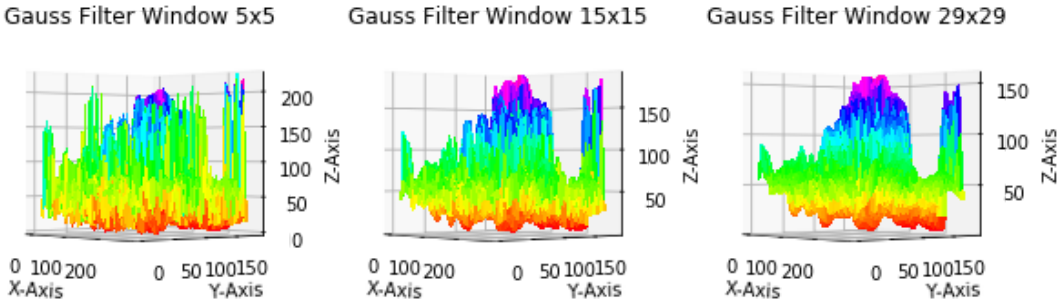

First derivative of Gaussian Filtered Img
Laplacian of Gaussian Filtered Img

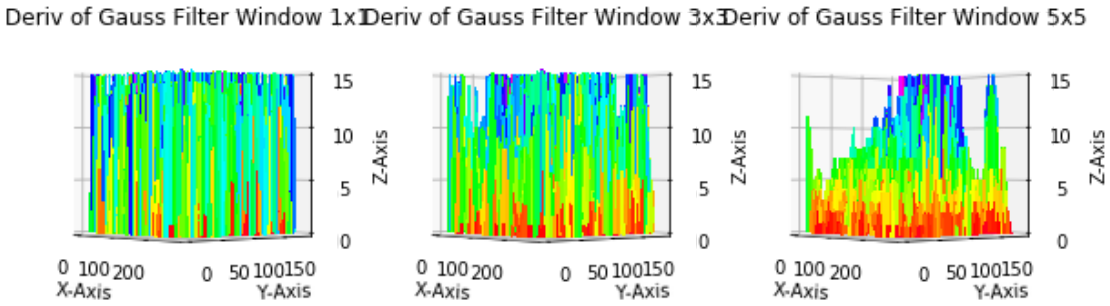

Gauss Filter Window 5x5
Gauss Filter Window 15x15
Gauss Filter Window 29x29
Z-Axis
X-Axis
Y-Axis

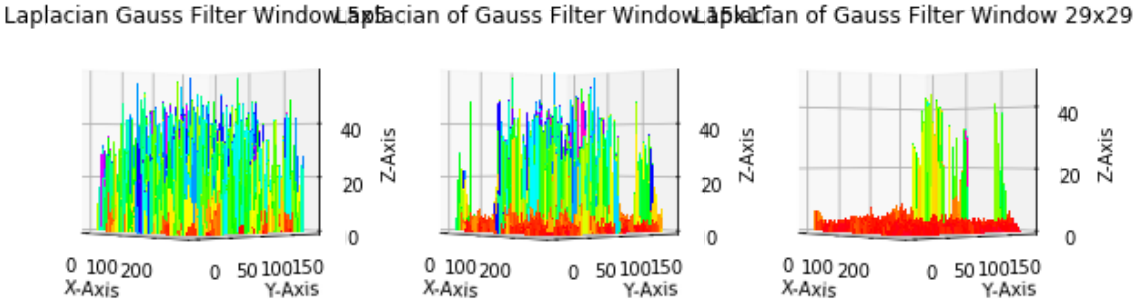

Deriv of Gauss Filter Window 1x1
Deriv of Gauss Filter Window 3x3
Deriv of Gauss Filter Window 5x5
Z-Axis
X-Axis
Y-Axis

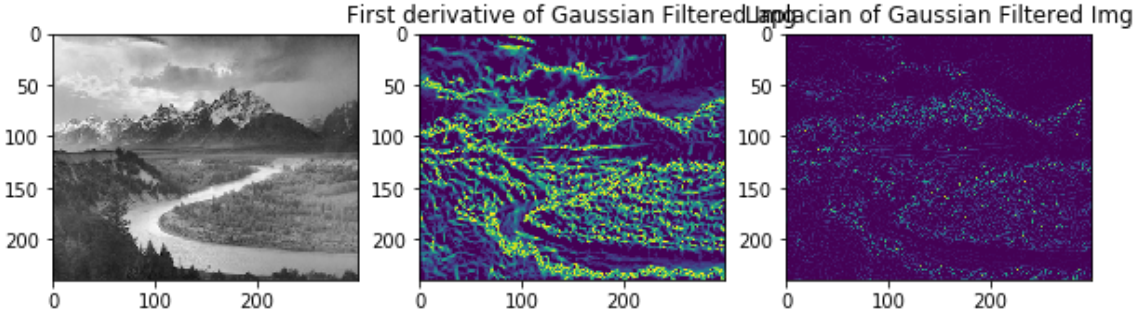

Laplacian Gauss Filter Window 5x5
Laplacian of Gauss Filter Window 15x15
Laplacian of Gauss Filter Window 29x29
Z-Axis
X-Axis
Y-Axis

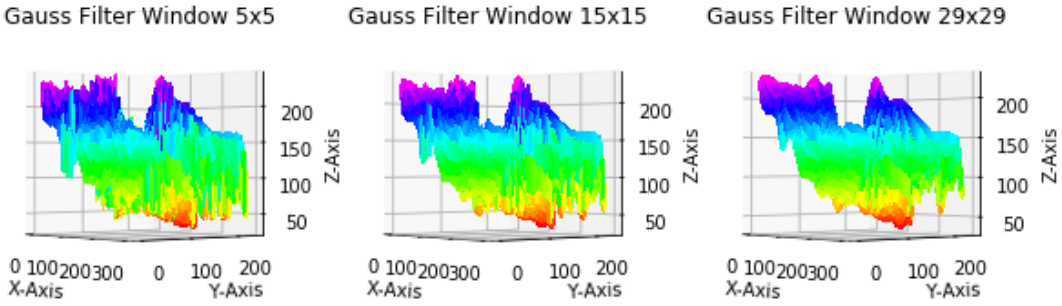

First derivative of Gaussian Filtered Img
Laplacian of Gaussian Filtered Img

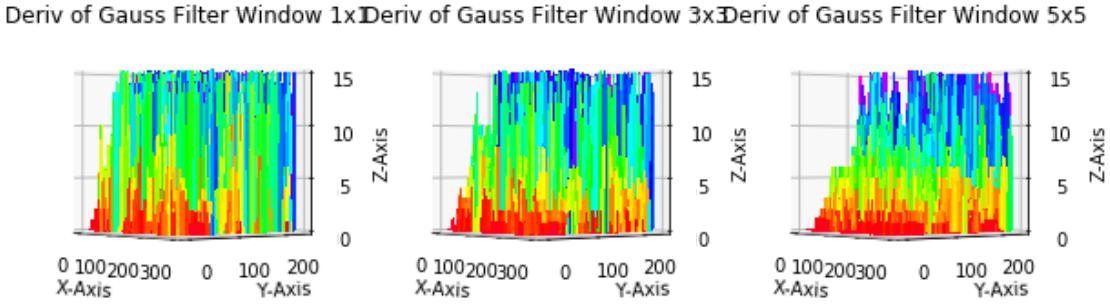

Gauss Filter Window 5x5
Gauss Filter Window 15x15
Gauss Filter Window 29x29
X-Axis
Y-Axis
Z-Axis

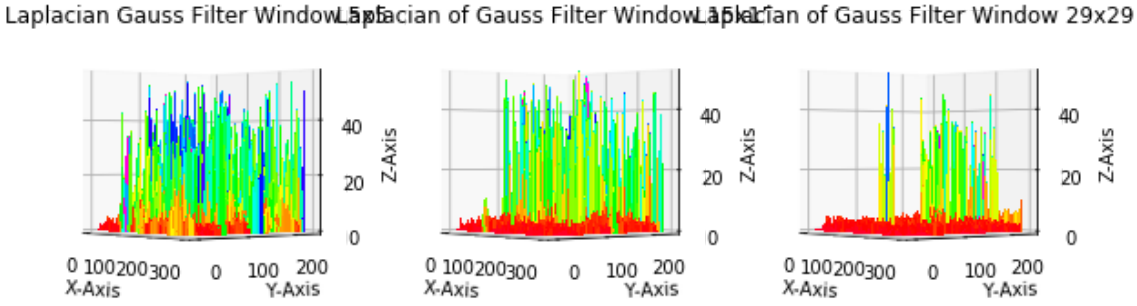

Deriv of Gauss Filter Window 1x1
Deriv of Gauss Filter Window 3x3
Deriv of Gauss Filter Window 5x5
X-Axis
Y-Axis
Z-Axis

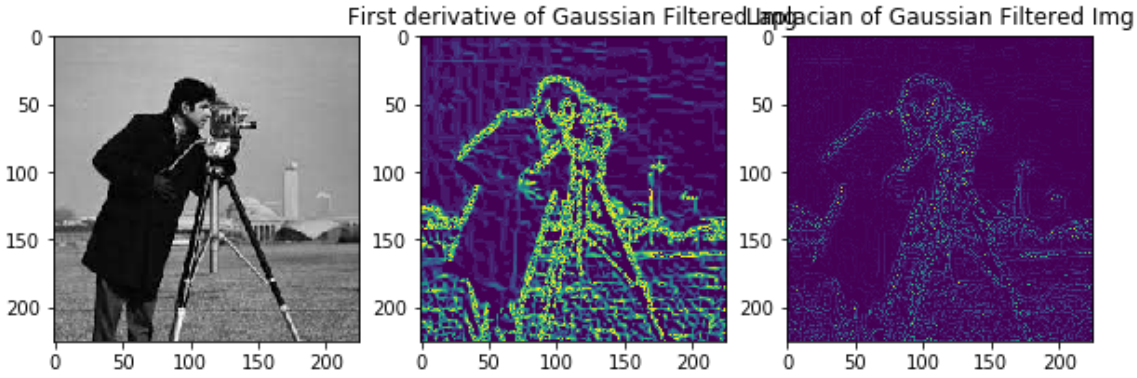

Laplacian Gauss Filter Window 5x5
Laplacian of Gauss Filter Window 15x15
Laplacian of Gauss Filter Window 29x29
X-Axis
Y-Axis
Z-Axis

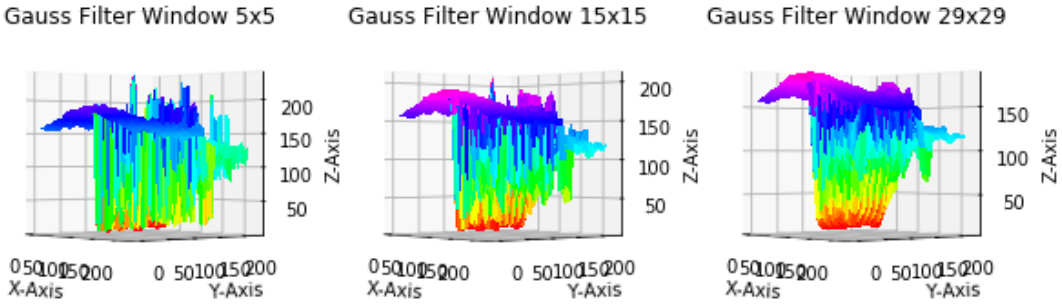

First derivative of Gaussian Filtered Img
Laplacian of Gaussian Filtered Img

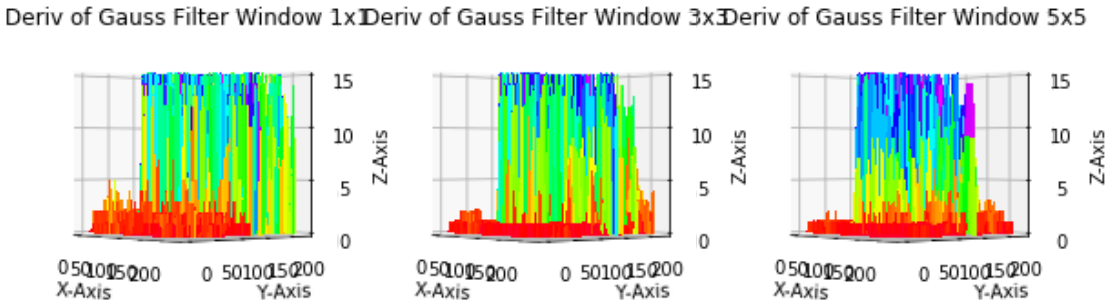

Gauss Filter Window 5x5
Gauss Filter Window 15x15
Gauss Filter Window 29x29
Z-Axis
X-Axis
Y-Axis

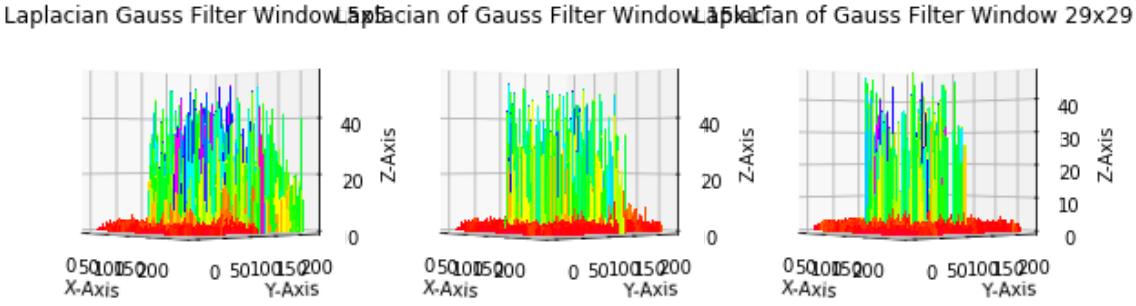

Deriv of Gauss Filter Window 1x1
Deriv of Gauss Filter Window 3x3
Deriv of Gauss Filter Window 5x5
Z-Axis
X-Axis
Y-Axis

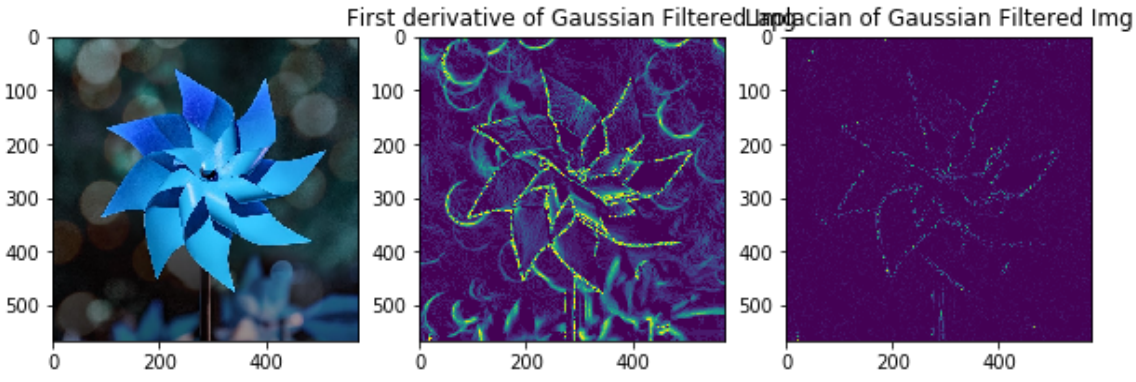

Laplacian Gauss Filter Window 5x5
Laplacian of Gauss Filter Window 15x15
Laplacian of Gauss Filter Window 29x29
Z-Axis
X-Axis
Y-Axis

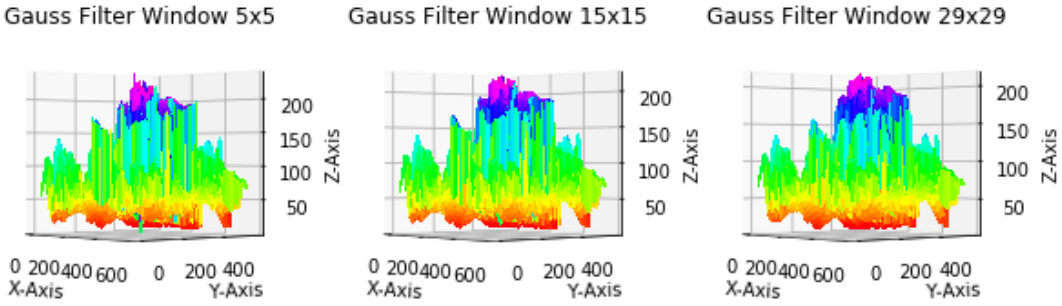

First derivative of Gaussian Filtered Img
Laplacian of Gaussian Filtered Img

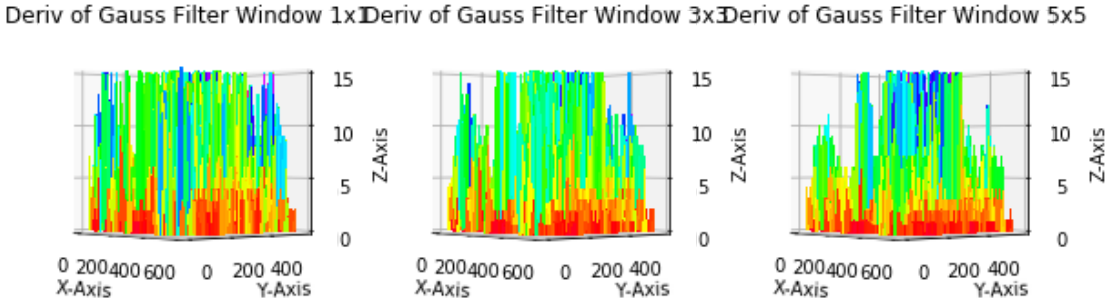

Gauss Filter Window 5x5
Gauss Filter Window 15x15
Gauss Filter Window 29x29
X-Axis
Y-Axis
Z-Axis

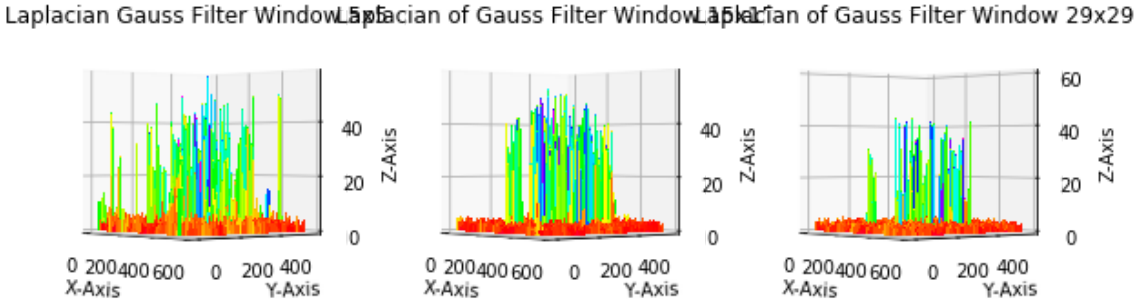

Deriv of Gauss Filter Window 1x1
Deriv of Gauss Filter Window 3x3
Deriv of Gauss Filter Window 5x5
X-Axis
Y-Axis
Z-Axis

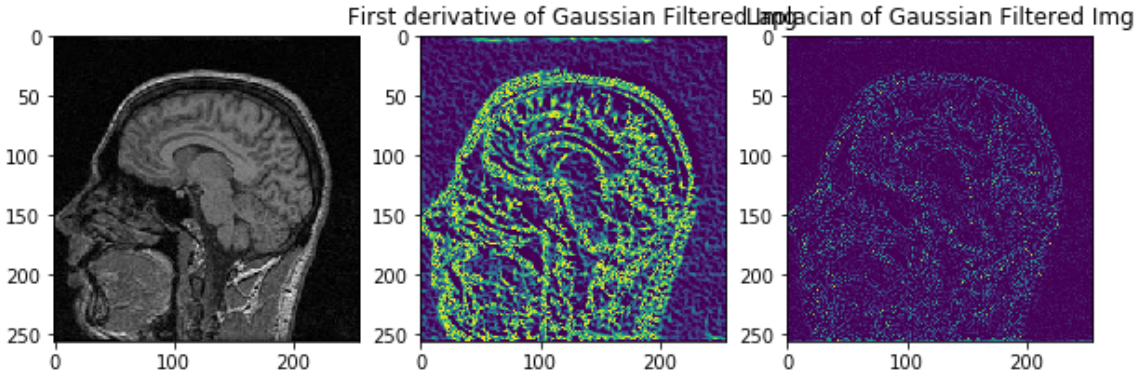

Laplacian Gauss Filter Window 5x5
Laplacian of Gauss Filter Window 15x15
Laplacian of Gauss Filter Window 29x29
X-Axis
Y-Axis
Z-Axis

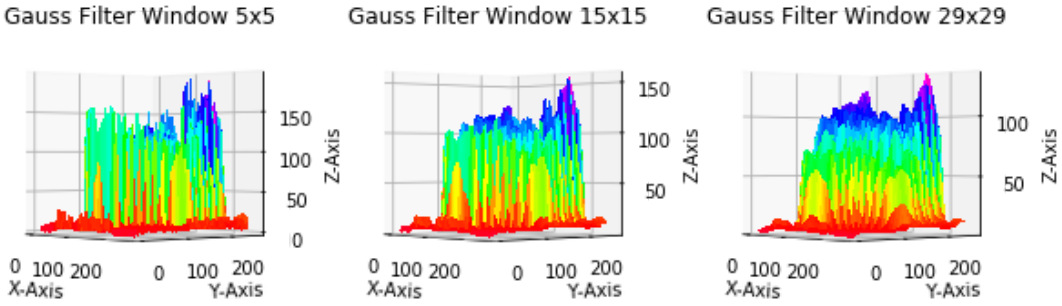

First derivative of Gaussian Filtered Img
Laplacian of Gaussian Filtered Img

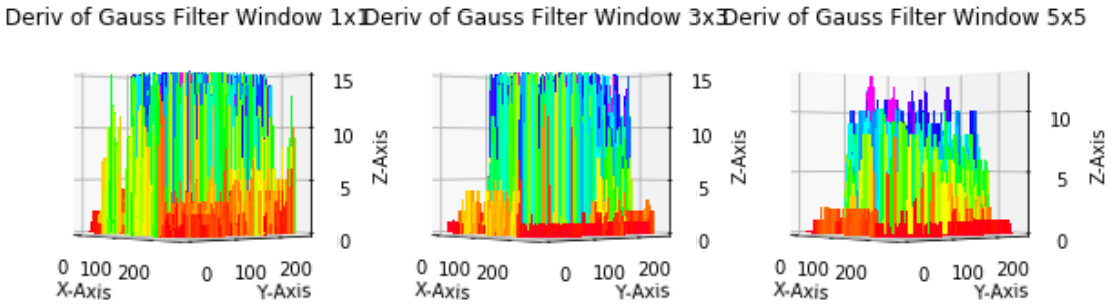

Gauss Filter Window 5x5
Gauss Filter Window 15x15
Gauss Filter Window 29x29
Z-Axis
X-Axis
Y-Axis

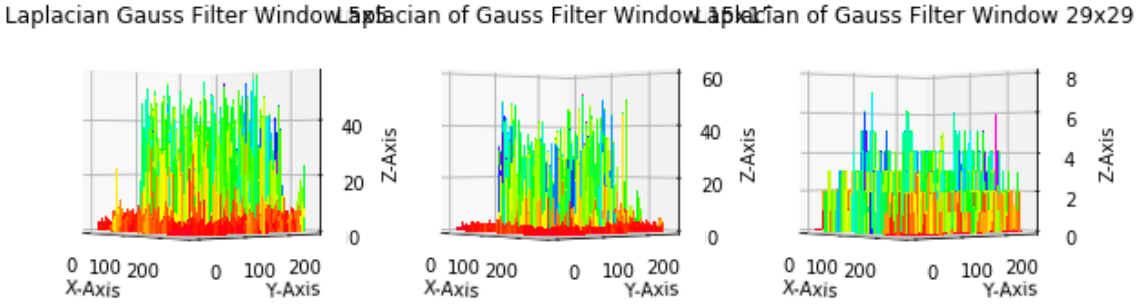

Deriv of Gauss Filter Window 1x1
Deriv of Gauss Filter Window 3x3
Deriv of Gauss Filter Window 5x5
Z-Axis
X-Axis
Y-Axis


Laplacian Gauss Filter Window 5x5
Laplacian of Gauss Filter Window 15x15
Laplacian of Gauss Filter Window 29x29
Z-Axis
X-Axis
Y-Axis


First derivative of Gaussian Filtered Img
Laplacian of Gaussian Filtered Img

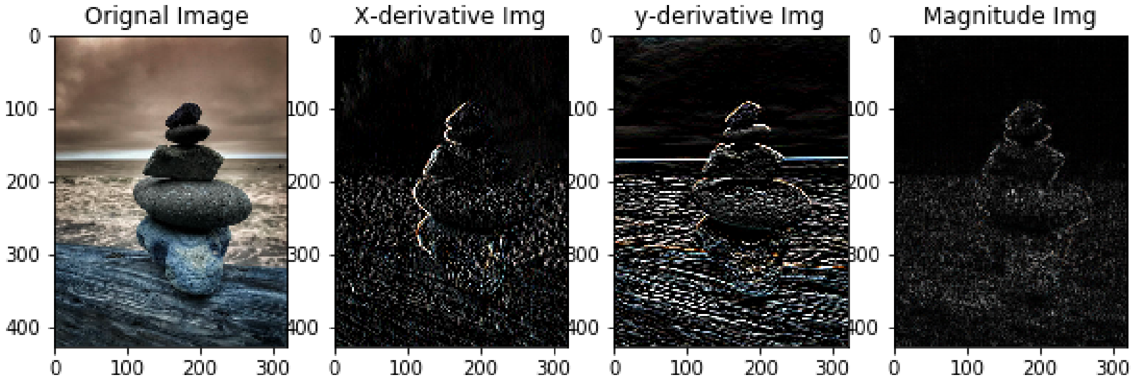

Gauss Filter Window 5x5
Gauss Filter Window 15x15
Gauss Filter Window 29x29
X-Axis
Y-Axis
Z-Axis

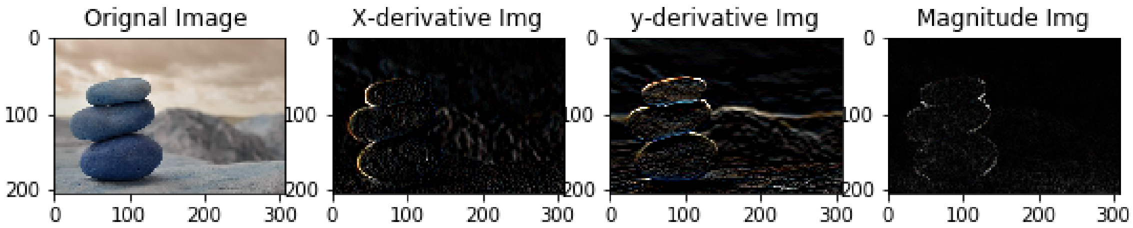

Deriv of Gauss Filter Window 1x1
Deriv of Gauss Filter Window 3x3
Deriv of Gauss Filter Window 5x5
X-Axis
Y-Axis
Z-Axis

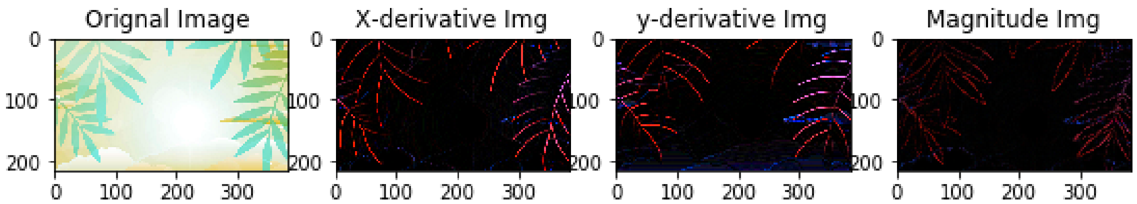

Laplacian Gauss Filter Window 5x5
Laplacian of Gauss Filter Window 15x15
Laplacian of Gauss Filter Window 29x29
X-Axis
Y-Axis
Z-Axis

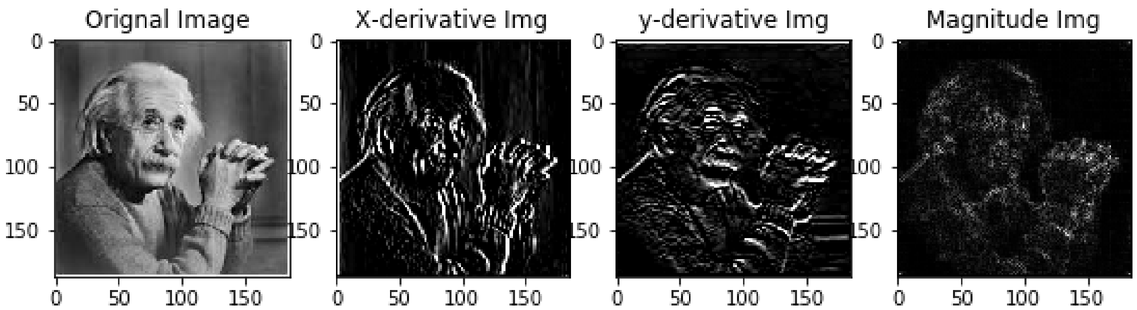
In [11]:

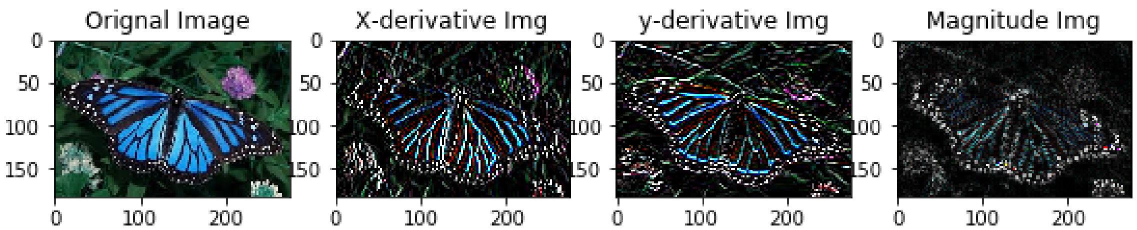
```
#2.61
# Here sobel filter has been applied on all images and plotted
sobel(images)
```

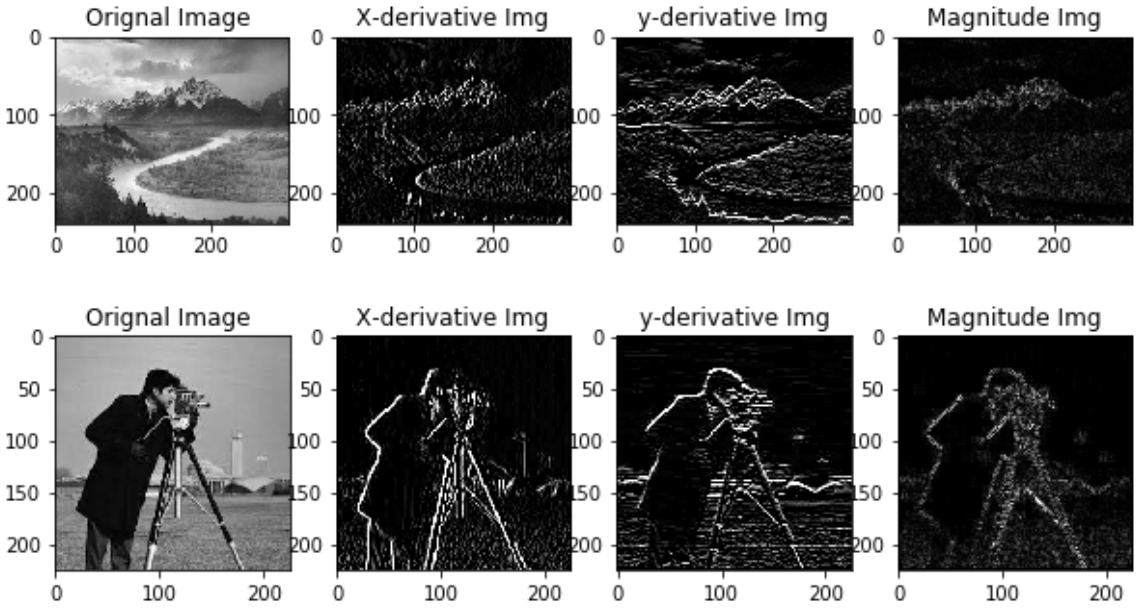

Orignal Image
X-derivative Img
y-derivative Img
Magnitude Img

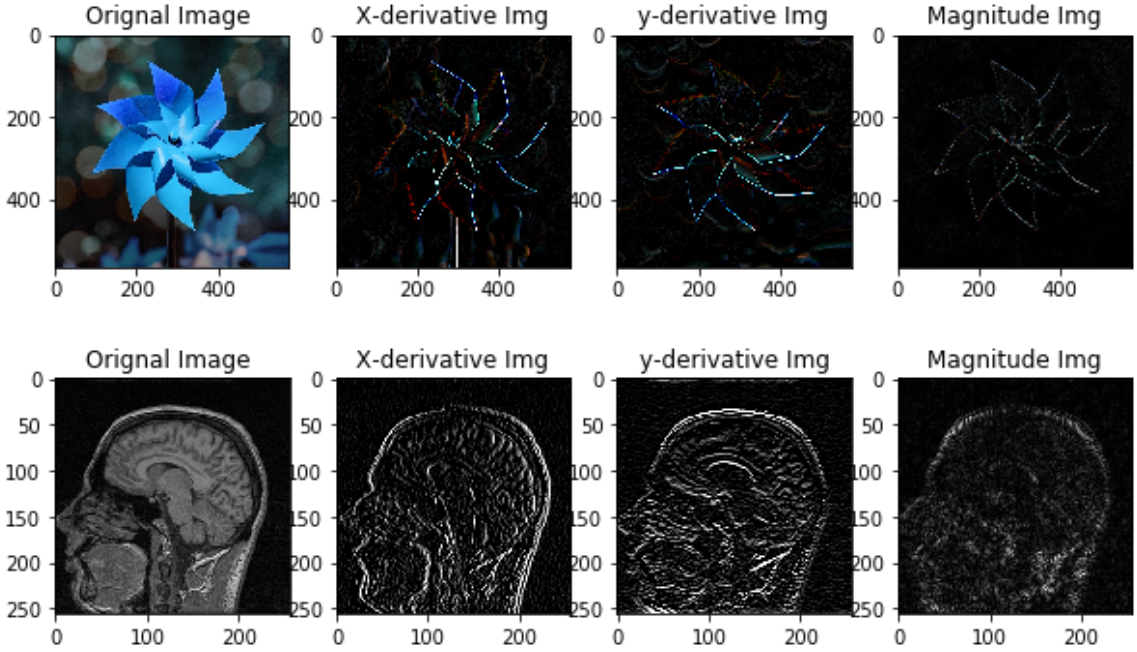

Orignal Image
X-derivative Img
y-derivative Img
Magnitude Img


Orignal Image
X-derivative Img
y-derivative Img
Magnitude Img


Orignal Image
X-derivative Img
y-derivative Img
Magnitude Img

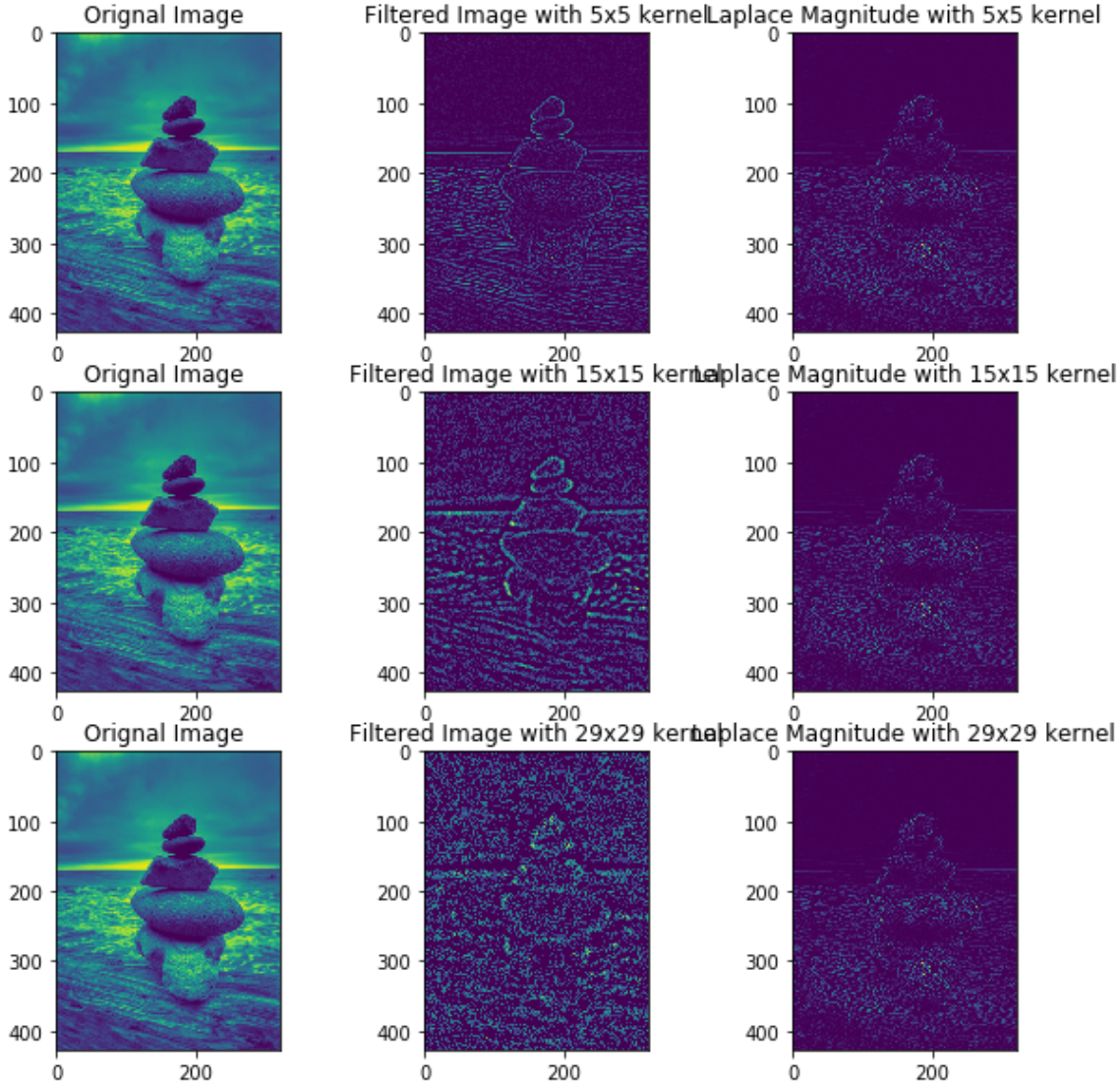

Orignal Image
X-derivative Img
y-derivative Img
Magnitude Img

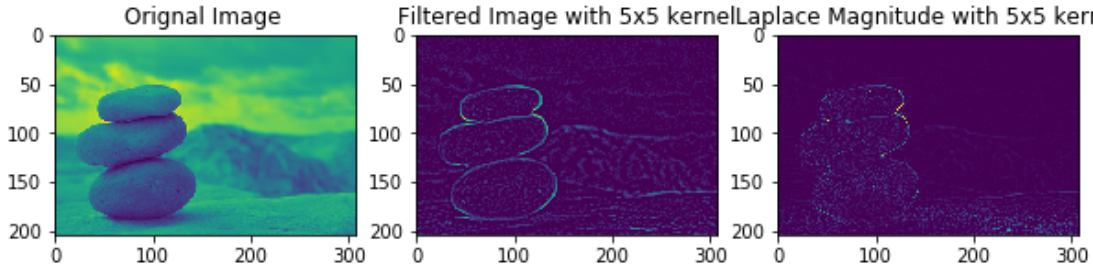

Orignal Image
X-derivative Img
y-derivative Img
Magnitude Img
Orignal Image
X-derivative Img
y-derivative Img
Magnitude Img

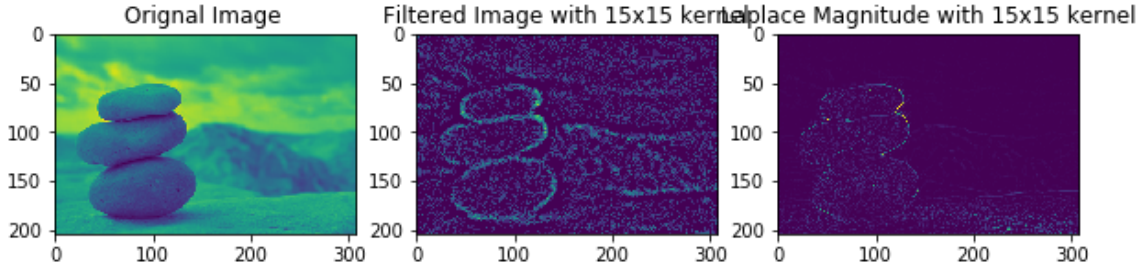

Orignal Image
X-derivative Img
y-derivative Img
Magnitude Img
Orignal Image
X-derivative Img
y-derivative Img
Magnitude Img

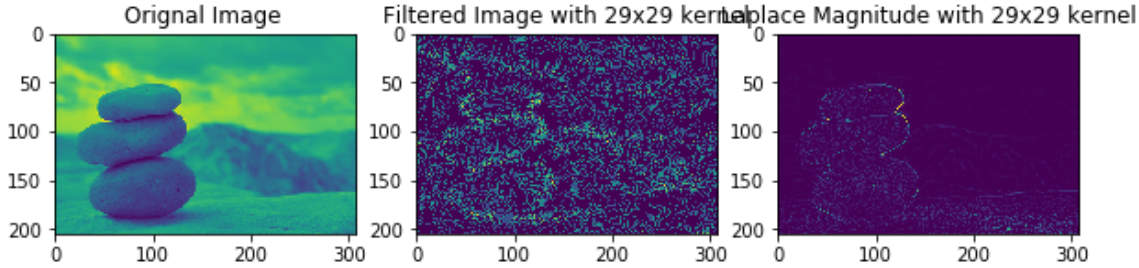
In [12]:

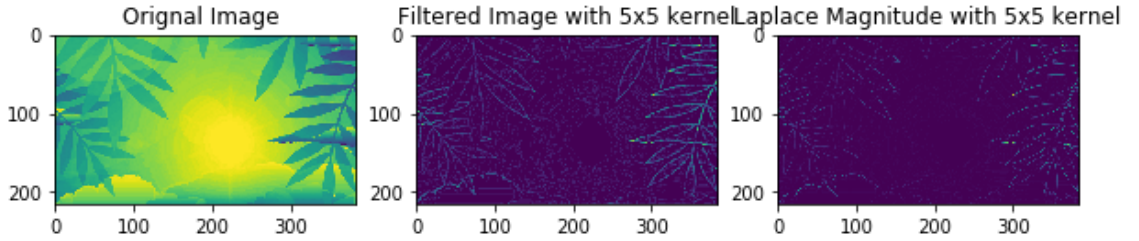
```
#2.62
# Here laplacian filter has been applied on all images and plotted
laplacian(images)
```

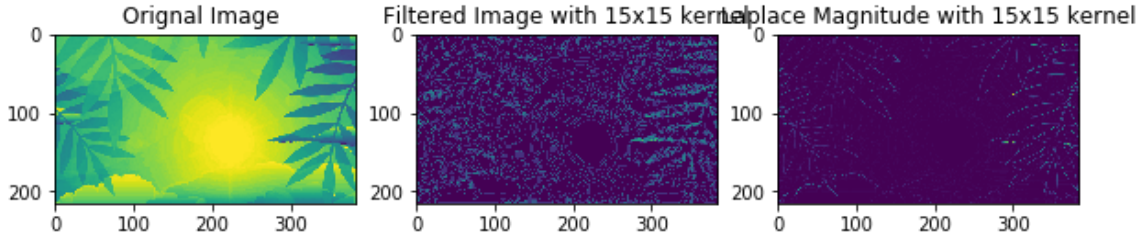

Orignal Image
Filtered Image with 5x5 kernel
Laplace Magnitude with 5x5 kernel
Orignal Image
Filtered Image with 15x15 kernel
Laplace Magnitude with 15x15 kernel
Orignal Image
Filtered Image with 29x29 kernel
Laplace Magnitude with 29x29 kernel

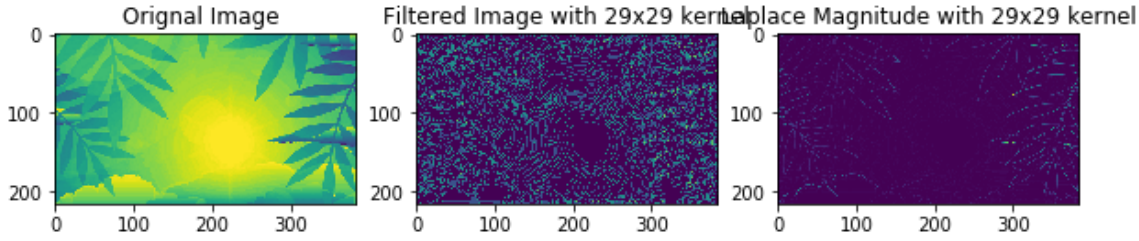

Orignal Image
Filtered Image with 5x5 kernel
Laplace Magnitude with 5x5 kernel

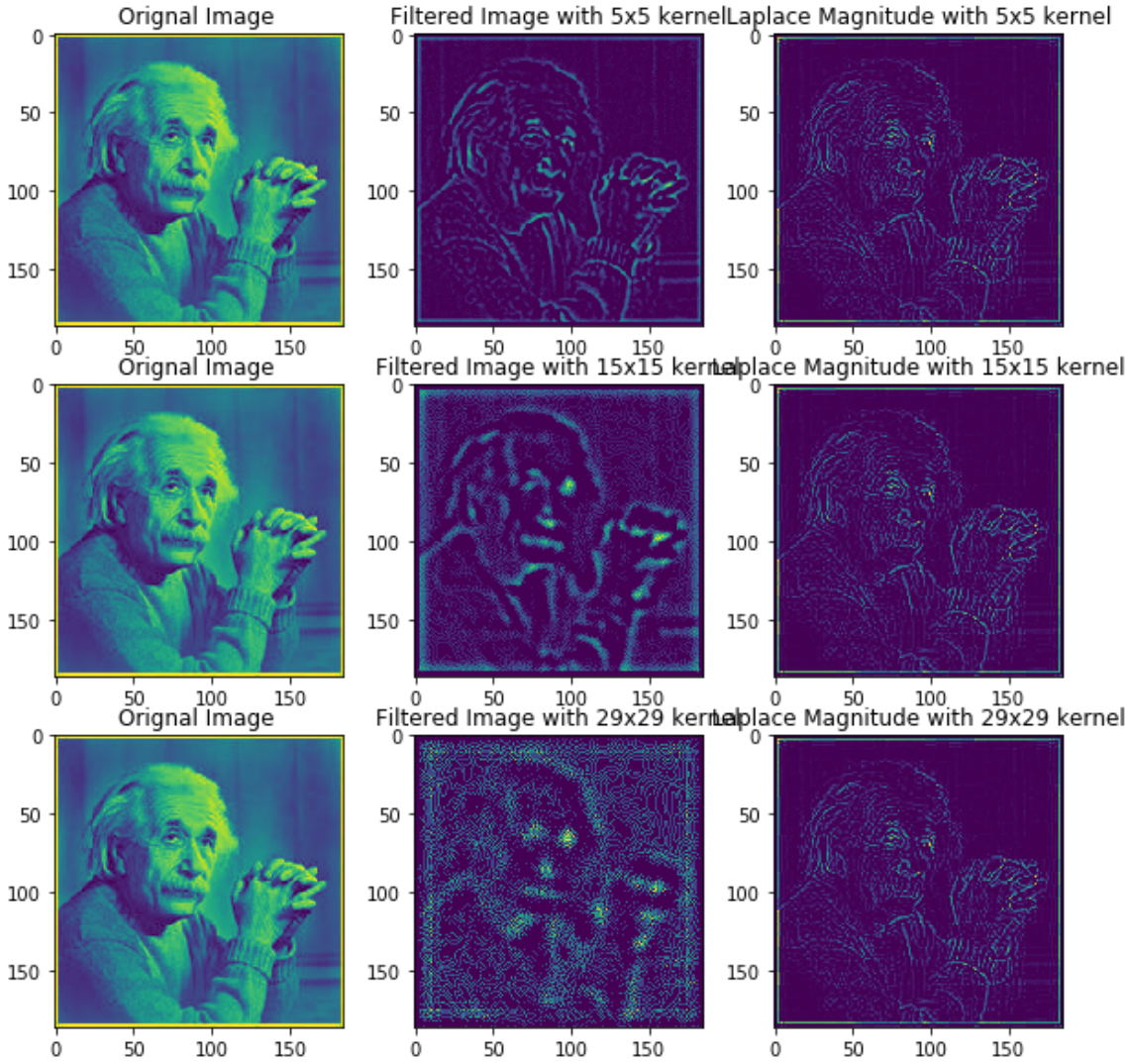

Orignal Image
Filtered Image with 15x15 kernel
Laplace Magnitude with 15x15 kernel

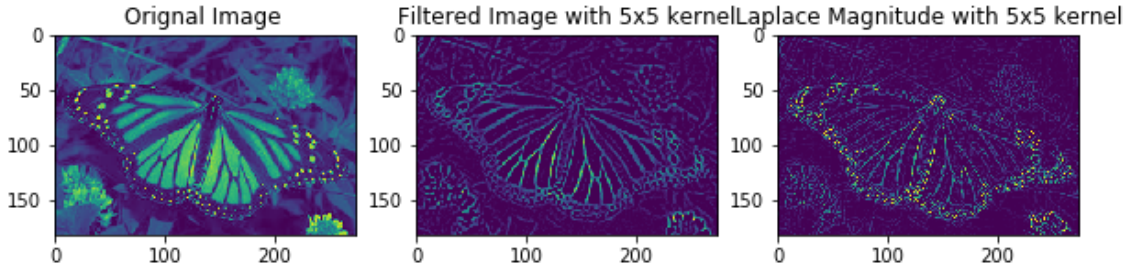

Orignal Image
Filtered Image with 29x29 kernel
Laplace Magnitude with 29x29 kernel

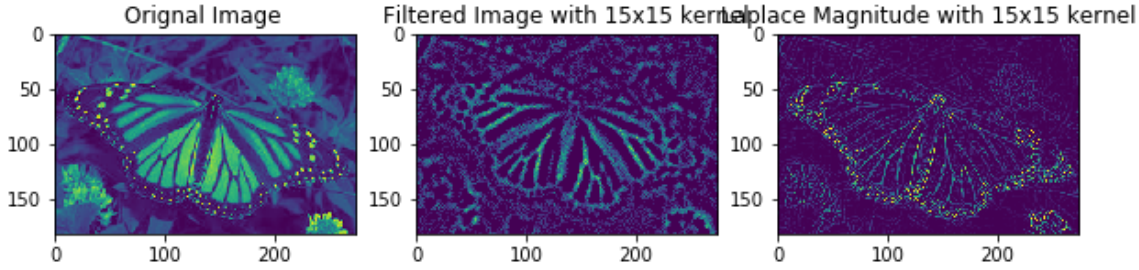

Orignal Image
Filtered Image with 5x5 kernel
Laplace Magnitude with 5x5 kernel

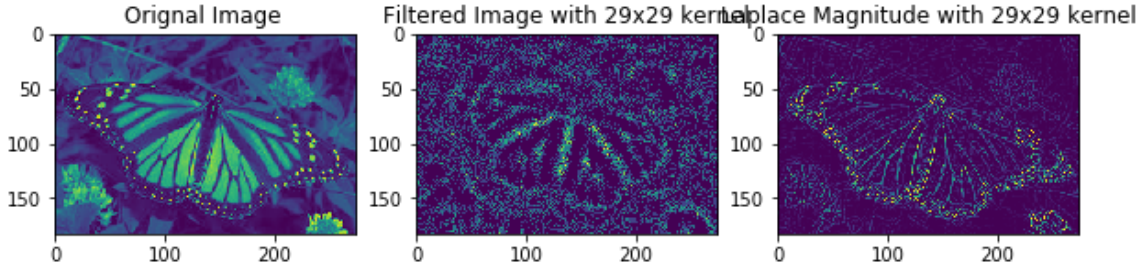

Orignal Image
Filtered Image with 15x15 kernel
Laplace Magnitude with 15x15 kernel

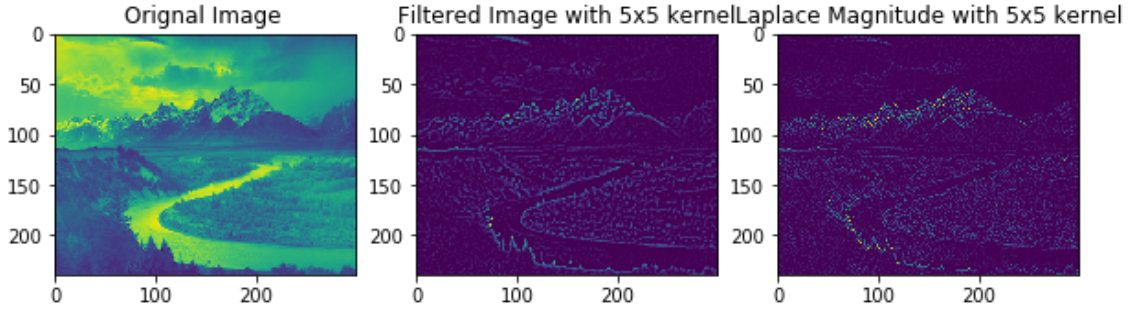

Orignal Image
Filtered Image with 29x29 kernel
Laplace Magnitude with 29x29 kernel

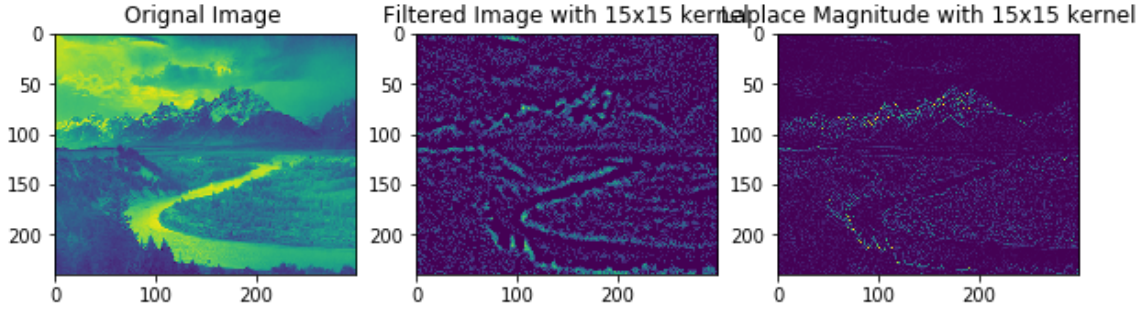

Orignal Image
Filtered Image with 5x5 kernel
Laplace Magnitude with 5x5 kernel
Orignal Image
Filtered Image with 15x15 kernel
Laplace Magnitude with 15x15 kernel
Orignal Image
Filtered Image with 29x29 kernel
Laplace Magnitude with 29x29 kernel

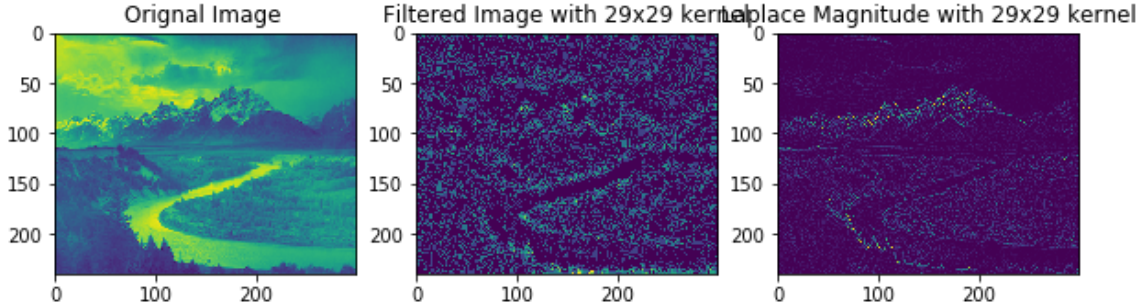

Orignal Image
Filtered Image with 5x5 kernel
Laplace Magnitude with 5x5 kernel

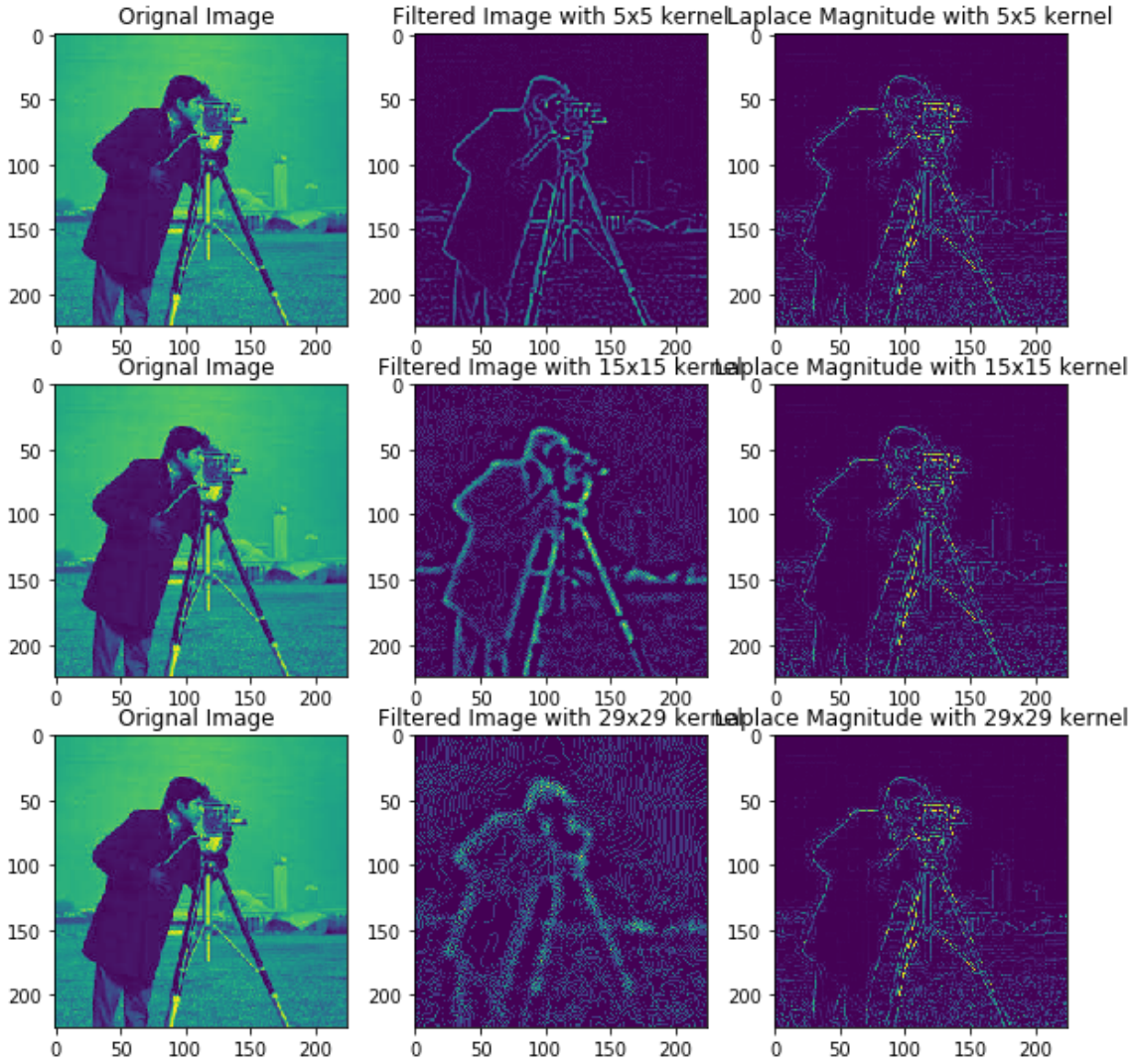

Orignal Image
Filtered Image with 15x15 kernel
Laplace Magnitude with 15x15 kernel

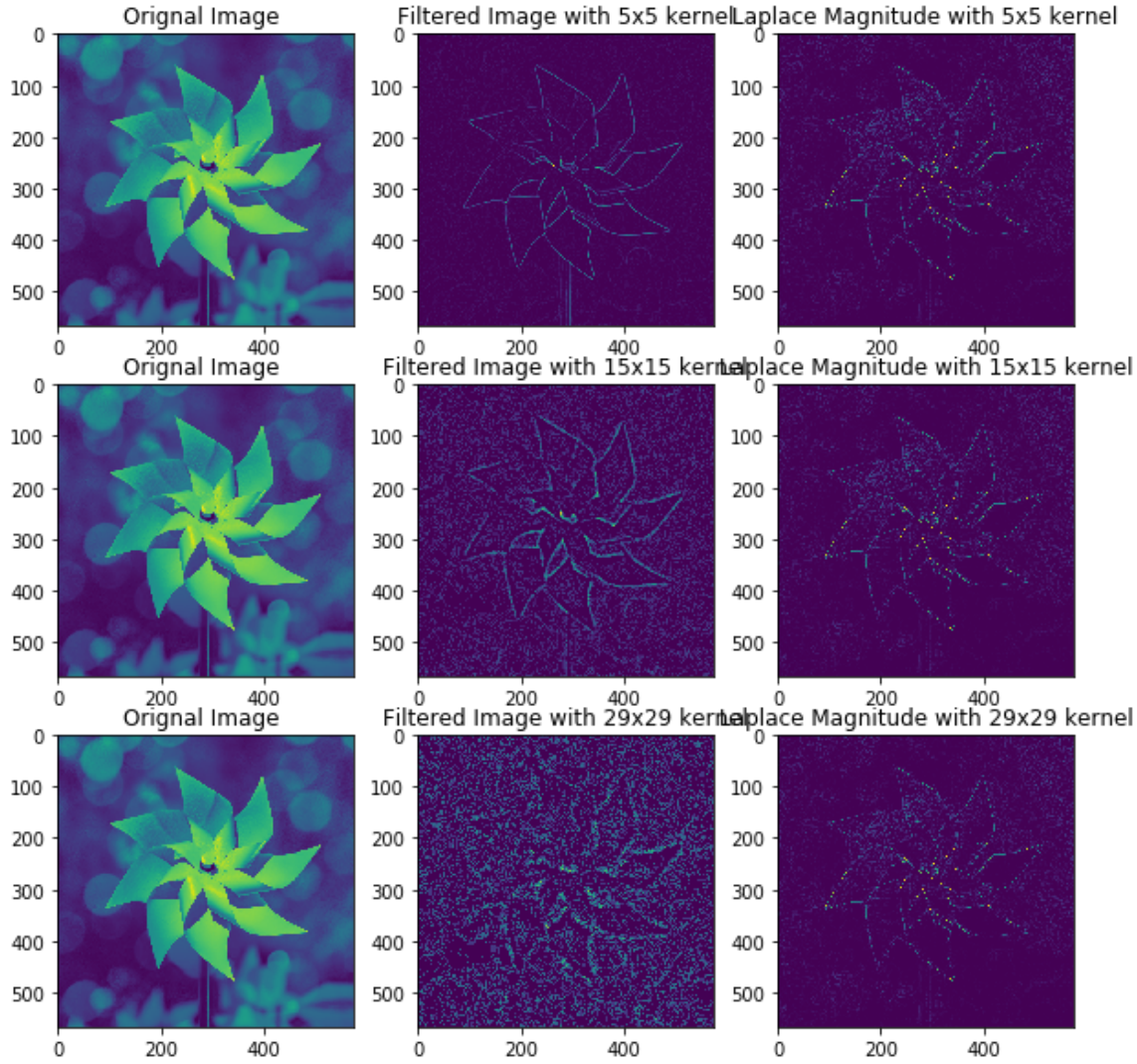

Orignal Image
Filtered Image with 29x29 kernel
Laplace Magnitude with 29x29 kernel

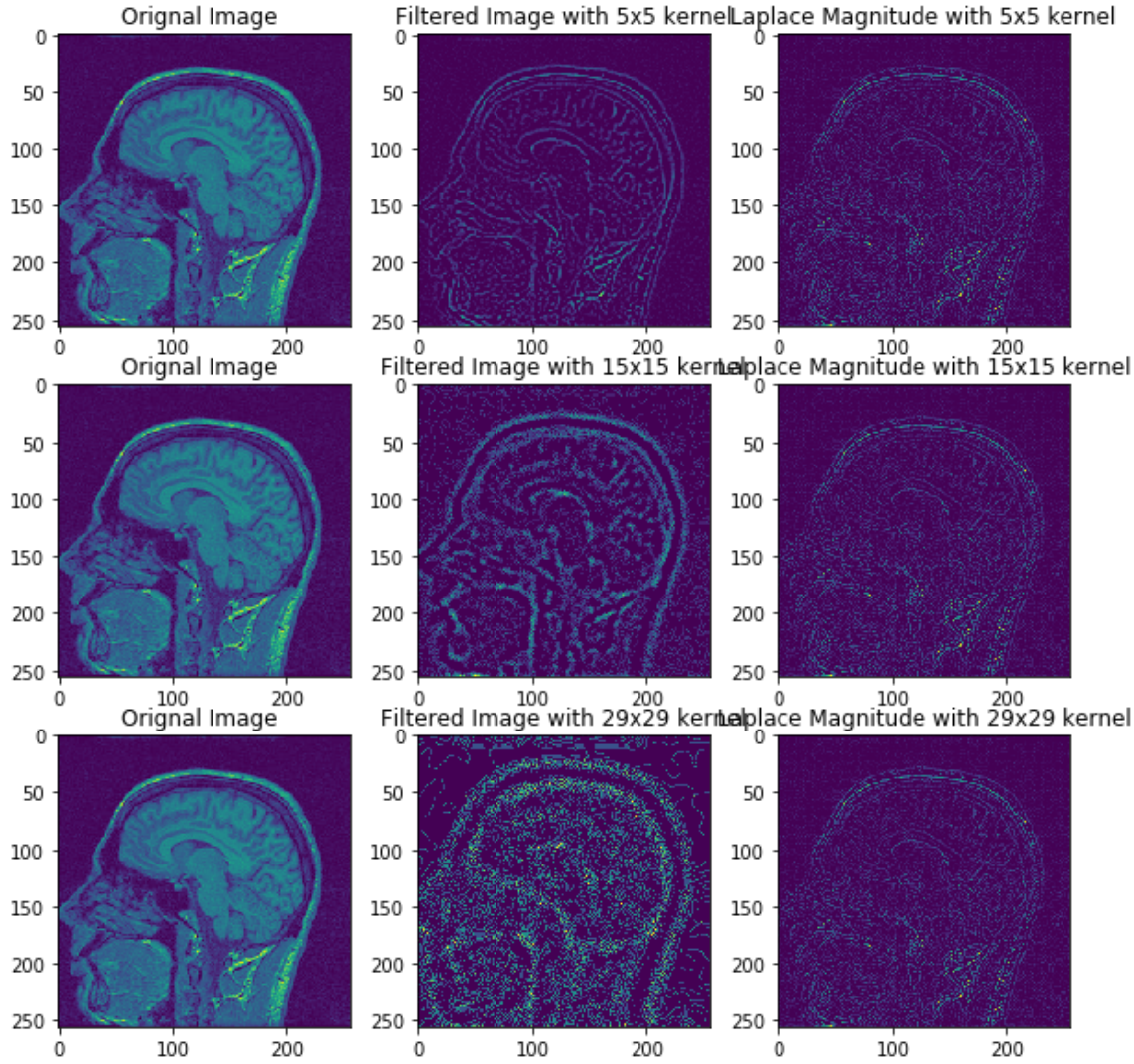

Orignal Image
Filtered Image with 5x5 kernel
Laplace Magnitude with 5x5 kernel


Orignal Image
Filtered Image with 15x15 kernel
Laplace Magnitude with 15x15 kernel


Orignal Image
Filtered Image with 29x29 kernel
Laplace Magnitude with 29x29 kernel


Orignal Image
Filtered Image with 5x5 kernel
Laplace Magnitude with 5x5 kernel
Orignal Image
Filtered Image with 15x15 kernel
Laplace Magnitude with 15x15 kernel
Orignal Image
Filtered Image with 29x29 kernel
Laplace Magnitude with 29x29 kernel

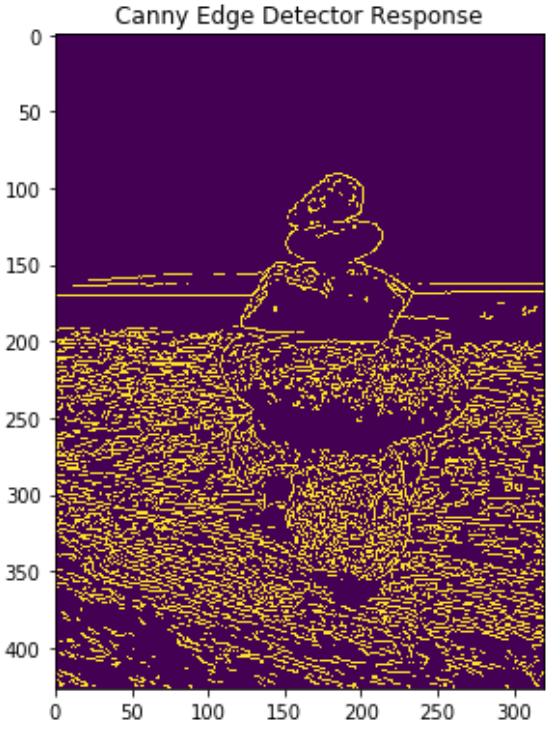

Orignal Image
Filtered Image with 5x5 kernel
Laplace Magnitude with 5x5 kernel
Orignal Image
Filtered Image with 15x15 kernel
Laplace Magnitude with 15x15 kernel
Orignal Image
Filtered Image with 29x29 kernel
Laplace Magnitude with 29x29 kernel

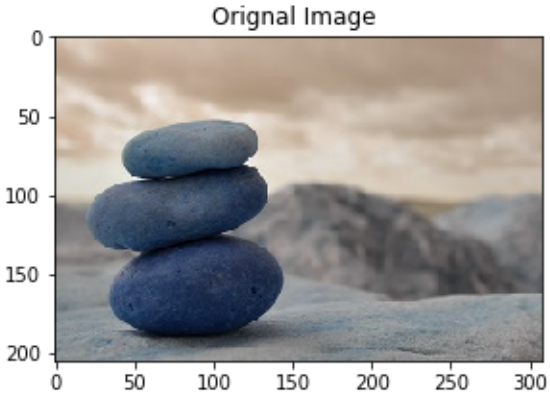

Orignal Image
Filtered Image with 5x5 kernel
Laplace Magnitude with 5x5 kernel
Orignal Image
Filtered Image with 15x15 kernel
Laplace Magnitude with 15x15 kernel
Orignal Image
Filtered Image with 29x29 kernel
Laplace Magnitude with 29x29 kernel

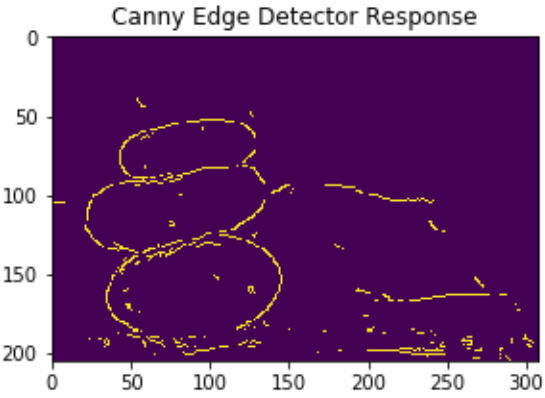
In [13]:

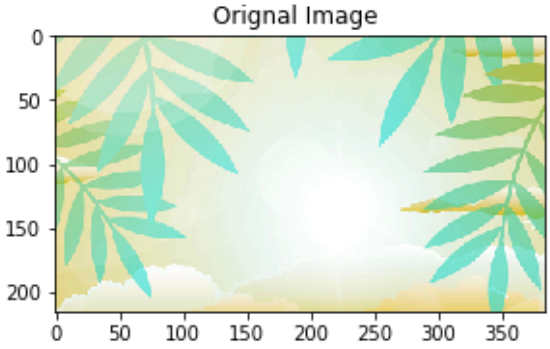
```
#2.63
## Here canny edge detector has been applied using built in lib on all images and plott
ed
canny(images)
```

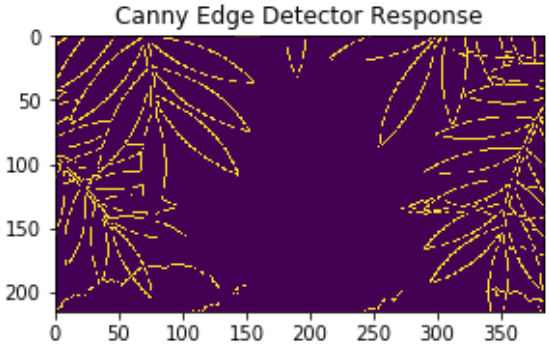

Orignal Image
0
50
100
150
200
250
300
350
400
0
50
100
150
200
250
300

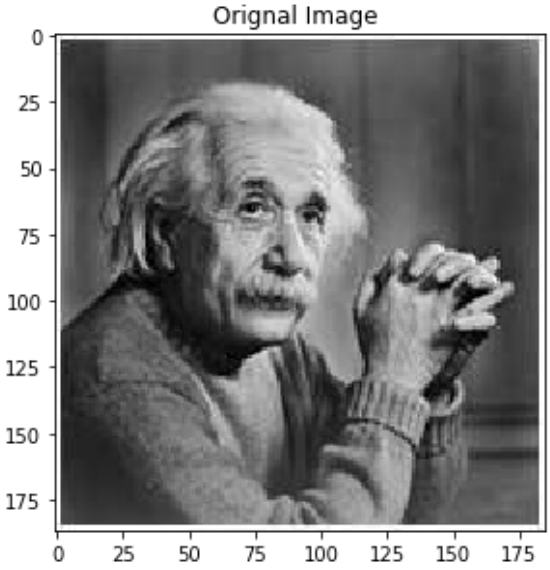

Canny Edge Detector Response
0
50
100
150
200
250
300
350
400
0
50
100
150
200
250
300

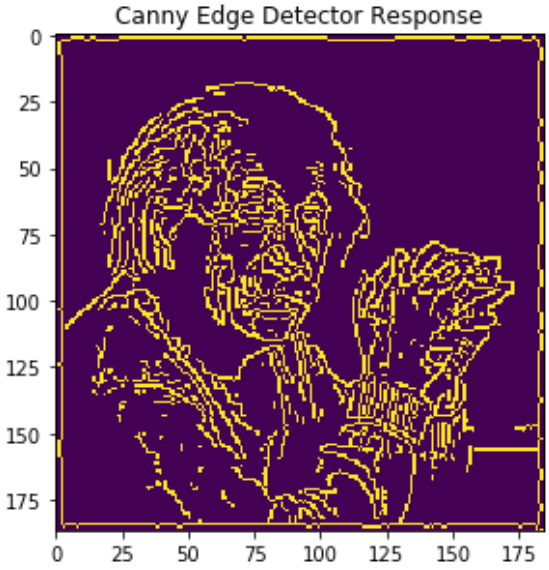

Orignal Image
0
50
100
150
200
0
50
100
150
200
250
300


Canny Edge Detector Response
0
50
100
150
200
0
50
100
150
200
250
300

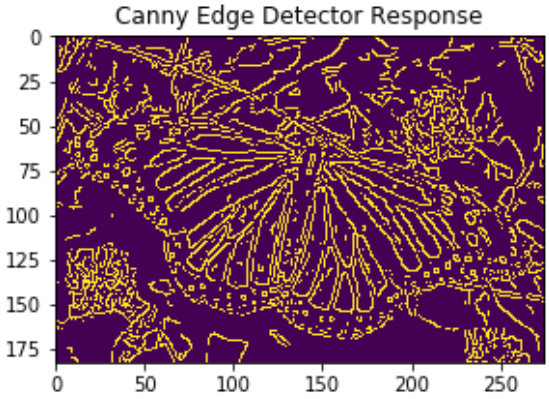

Orignal Image

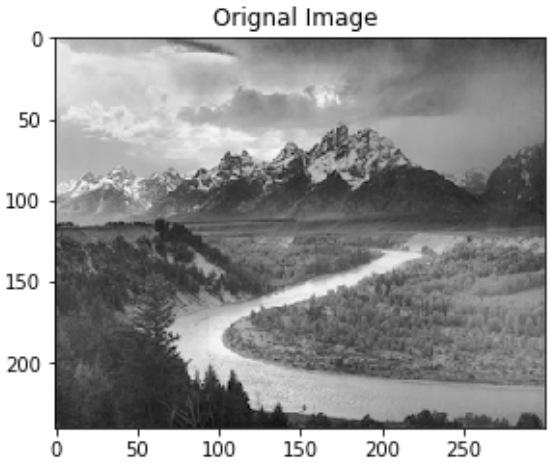

Canny Edge Detector Response

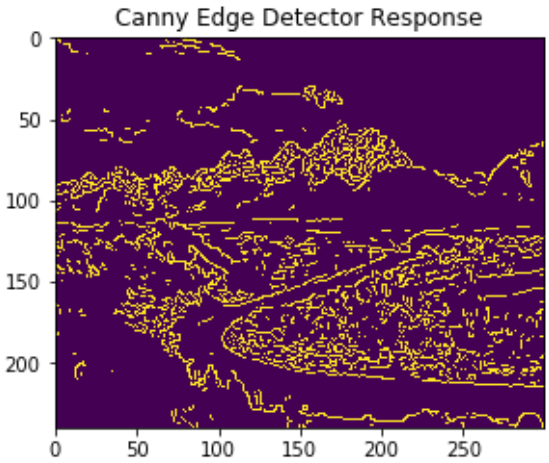

Orignal Image

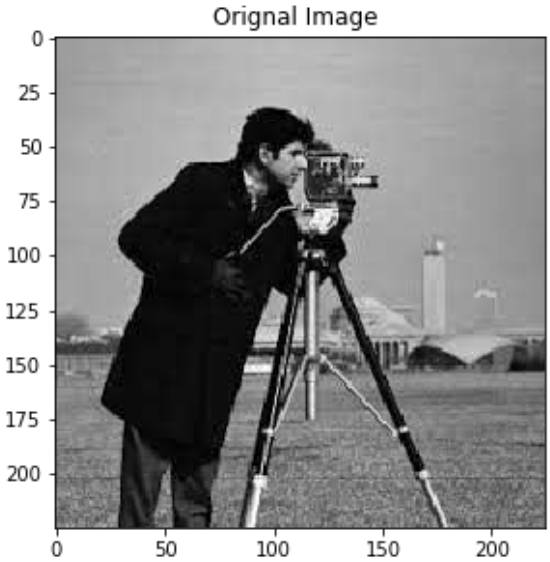

Canny Edge Detector Response

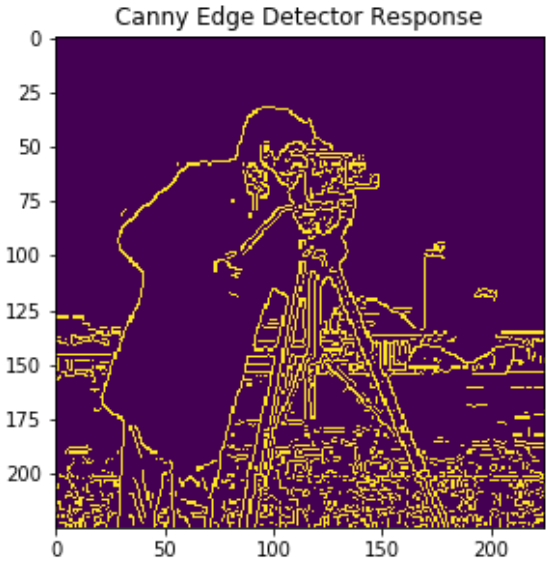

Orignal Image

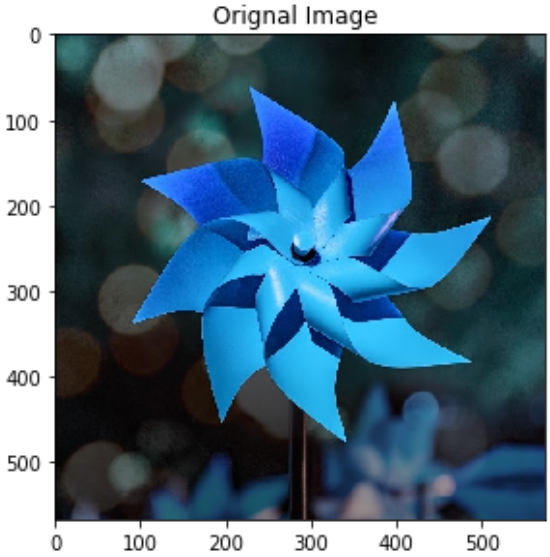

Canny Edge Detector Response

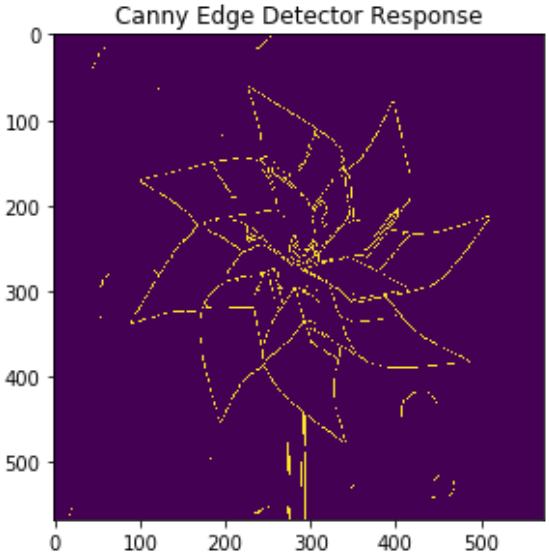

Orignal Image

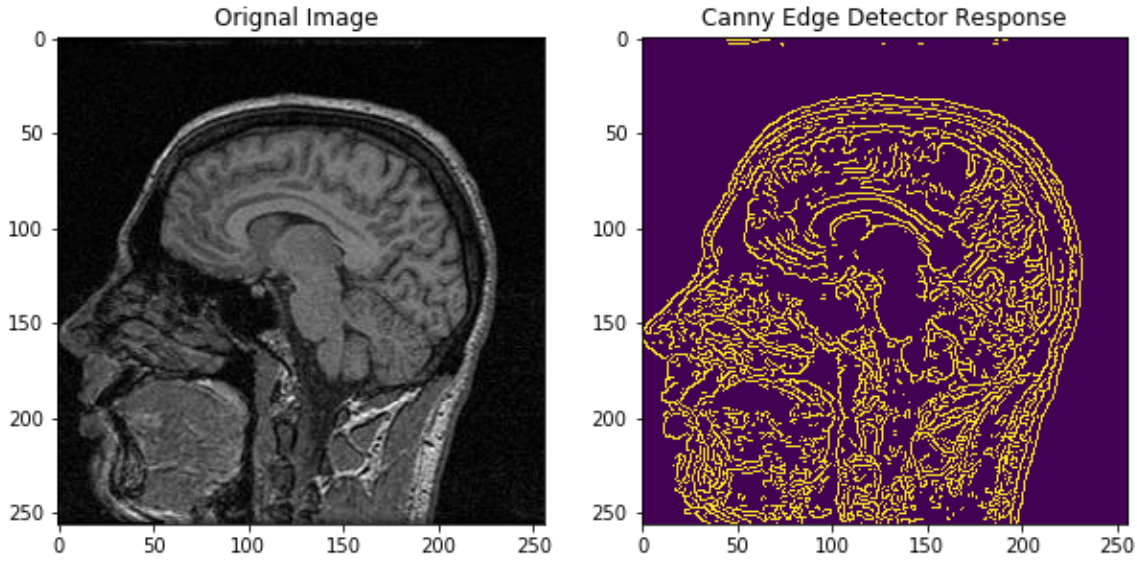

Canny Edge Detector Response


Orignal Image


Canny Edge Detector Response


Orignal Image


Canny Edge Detector Response



In [ ]:

```
#Bonus Task
# here live webcam video will open and on it, canny edge will apply with min threshold
 one 1 and max threshold of 90
import cv2
cap = cv2.VideoCapture(0)
while True:
    ret, img = cap.read()

    grayimg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurimg = cv2.GaussianBlur(grayimg, (5, 5), 0)
    canny_det = cv2.Canny(blurimg, 1, 90)
    ret, mask = cv2.threshold(canny_det, 90, 255, cv2.THRESH_BINARY)
    cv2.imshow('Live Webcam feed', mask)

    if cv2.waitKey(1) == 3:
        break
cap.release()
cv2.destroyAllWindows()
```

In [ ]: