Nima Aghli

PA2

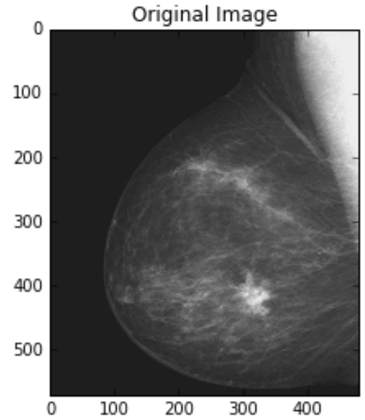
In [175]:

```
%matplotlib inline
from pylab import *
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import cv2
import matplotlib.image as mpimg
from pylab import *
import pandas as pd
```

Figure 3.4.b

In [176]:

```
img=mpimg.imread('Fig0304(a).tif')
img_p = cv2.imread('Fig0304(a).tif')
img_negative=255-img_p#substract the max intensity value from each pixel to rever
se values

figure()
plt.imshow(img_p)
title('Original Image')
show()

figure()
plt.imshow(img_negative)
title('Negative Image')
show()
```



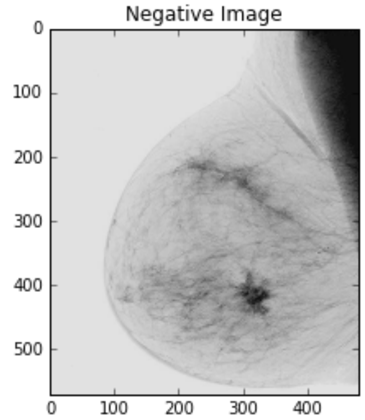


Figure 3.16

In [177]:

```
img_1 = cv2.imread('Fig0316(1).tif')
img_2 = cv2.imread('Fig0316(2).tif')
img_3 = cv2.imread('Fig0316(3).tif')
img_4 = cv2.imread('Fig0316(4).tif')



nrows = 4
ncols = 2
fig = plt.figure(figsize=(10, 15))
ax = fig.add_subplot(nrows, ncols, 1)
ax.imshow(img_4)
ax = fig.add_subplot(nrows, ncols, 2)
ax.hist(img_4.ravel(),256,[0,256]);

ax = fig.add_subplot(nrows, ncols, 3)
ax.imshow(img_1)
ax = fig.add_subplot(nrows, ncols, 4)
ax.hist(img_1.ravel(),256,[0,256]);

ax = fig.add_subplot(nrows, ncols, 5)
ax.imshow(img_2)
ax = fig.add_subplot(nrows, ncols, 6)
ax.hist(img_2.ravel(),256,[0,256]);

ax = fig.add_subplot(nrows, ncols, 7)
ax.imshow(img_3)
ax = fig.add_subplot(nrows, ncols, 8)
ax.hist(img_3.ravel(),256,[0,256]);
```

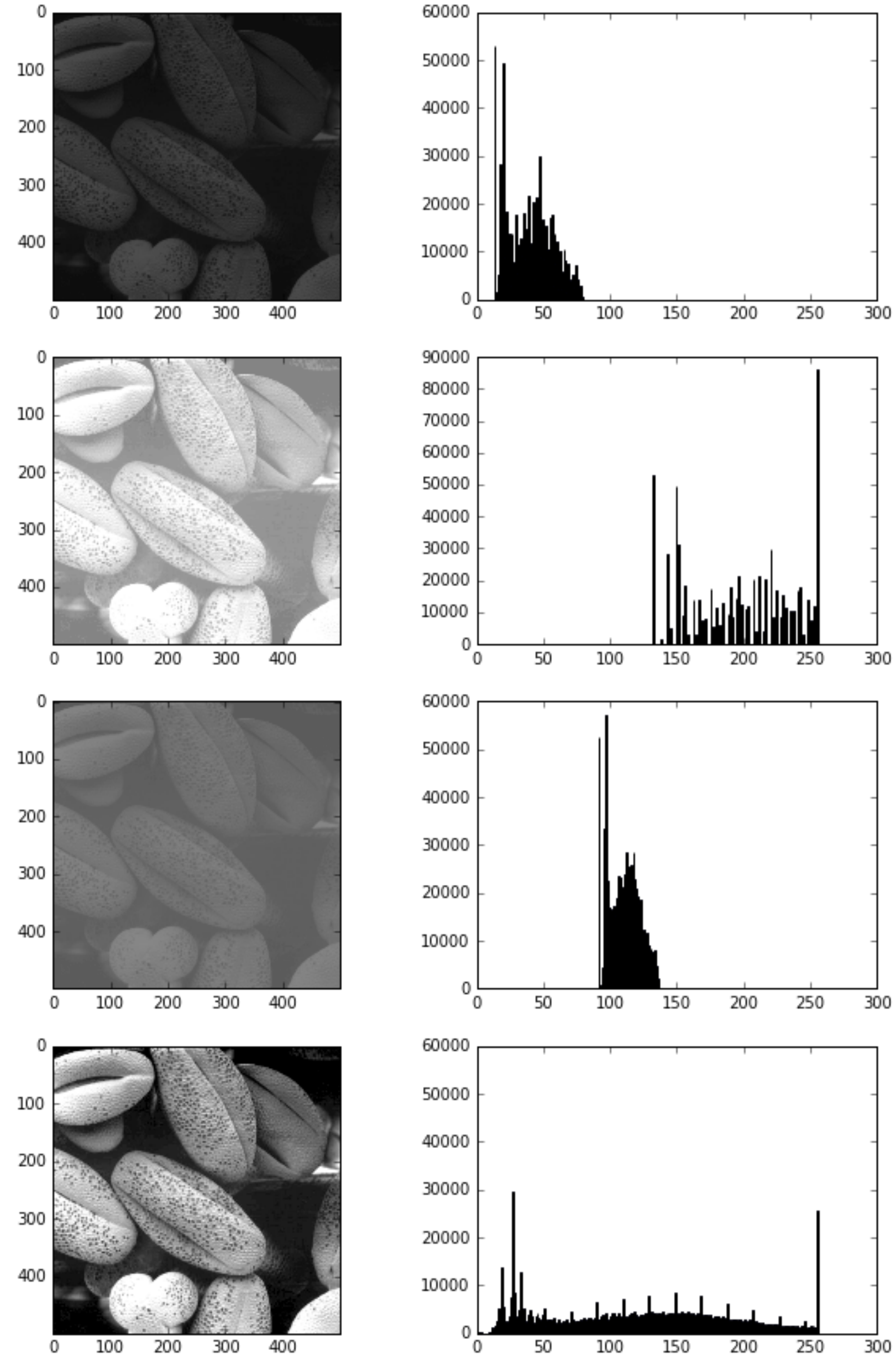


Figure 3.20 (Histogram Equalizetion)

In [178]:

```
## Histogram Equalizetion Example
img_1 = cv2.imread('Fig0316(1).tif',0)
img_2 = cv2.imread('Fig0316(2).tif',0)
img_3 = cv2.imread('Fig0316(3).tif',0)
img_4 = cv2.imread('Fig0316(4).tif',0)

equ_1 = cv2.equalizeHist(img_1)
equ_2 = cv2.equalizeHist(img_2)
equ_3 = cv2.equalizeHist(img_3)
equ_4 = cv2.equalizeHist(img_4)

#strat showing
nrows = 4
ncols = 3
fig = plt.figure(figsize=(10, 15))
ax = fig.add_subplot(nrows, ncols, 1)
ax.imshow(img_4,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
ax = fig.add_subplot(nrows, ncols, 2)
ax.imshow(equ_4,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
ax = fig.add_subplot(nrows, ncols, 3)
ax.hist(equ_4.ravel(),256,[0,256]);

fig = plt.figure(figsize=(10, 15))
ax = fig.add_subplot(nrows, ncols, 4)
ax.imshow(img_1,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
ax = fig.add_subplot(nrows, ncols, 5)
ax.imshow(equ_1,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
ax = fig.add_subplot(nrows, ncols, 6)
ax.hist(equ_1.ravel(),256,[0,256]);

fig = plt.figure(figsize=(10, 15))
ax = fig.add_subplot(nrows, ncols, 7)
ax.imshow(img_2,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
ax = fig.add_subplot(nrows, ncols, 8)
ax.imshow(equ_2,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
ax = fig.add_subplot(nrows, ncols, 9)
ax.hist(equ_2.ravel(),256,[0,256]);


fig = plt.figure(figsize=(10, 15))
ax = fig.add_subplot(nrows, ncols, 7)
ax.imshow(img_3,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
ax = fig.add_subplot(nrows, ncols, 8)
ax.imshow(equ_3,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
ax = fig.add_subplot(nrows, ncols, 9)
ax.hist(equ_3.ravel(),256,[0,256]);
```

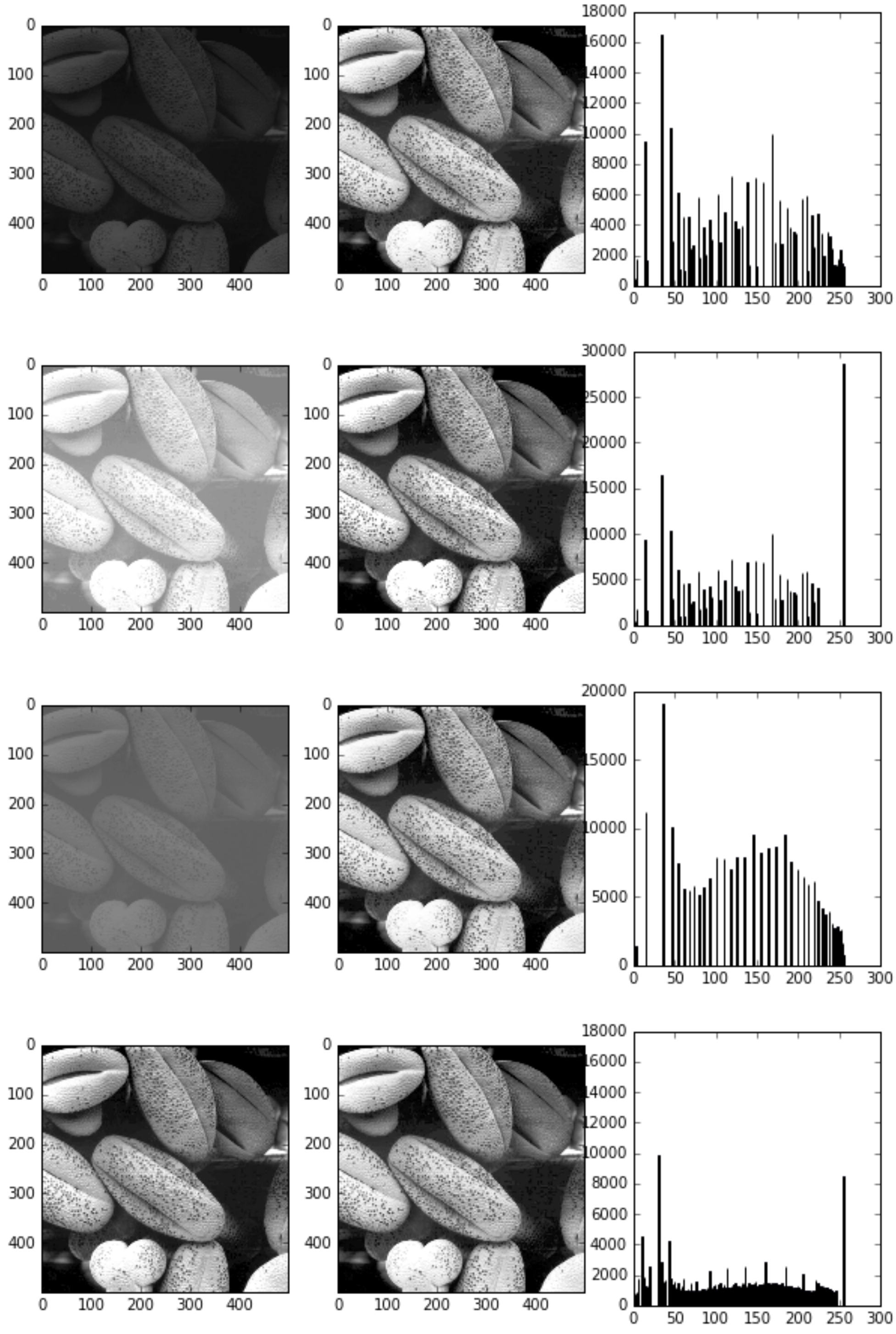


Figure 3.35(Median And Averaging Special filter With OpenCV)

```
In [179]: #median special filter averaging special filter using opencv
          img_5 = cv2.imread('Fig0335(a).tif')

          blur_5 = cv2.blur(img_5,(3,3))
          median_5 = cv2.medianBlur(img_5,3)
          nrows = 1
          ncols = 3
          fig = plt.figure(figsize=(10, 15))
          ax = fig.add_subplot(nrows, ncols, 1)
          ax.imshow(img_5)

          ax = fig.add_subplot(nrows, ncols, 2)
          ax.imshow(blur_5)
          ax = fig.add_subplot(nrows, ncols, 3)
          ax.imshow(median_5)
```

```
Out[179]: <matplotlib.image.AxesImage at 0x116dc69e8>
```

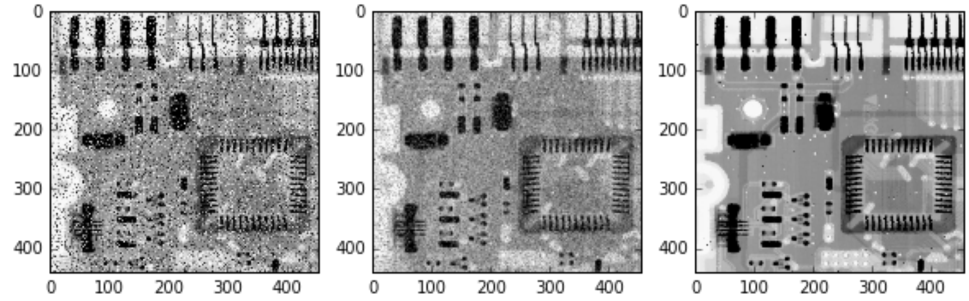


Figure 3.35(Median Special Filter Implementation)

In [180]:

```
    #Implementation of Median_filter
    #pseudo code from https://en.wikipedia.org/wiki/Median_filter
    source = cv2.imread('Fig0335(a).tif', 0)
    nrows = 1
    ncols = 2
    fig = plt.figure(figsize=(10, 15))
    ax = fig.add_subplot(nrows, ncols, 1)
    ax.imshow(source,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
    final = source
    window=[1,1,1]*3#create windows of size 9 here to save neghbors intensity val
ues
    height = np.size(source, 0)#generate height and
    width = np.size(source, 1)#width to loop over them
    edgex = math.floor((3/ 2))#used to take care of "not processing boundaries"
    edgey = math.floor((3/ 2))#used to take care of "not processing boundaries"
    for y in range(edgex,width-edgex):
        for x in range(edgey,height-edgey):
            i=0
            for fx in range(0,3):
               for fy in range(0,3):
                   window[i] = source[x + fx - edgex,y + fy - edgey]#pick one of
3*3 neighbor in each itreation
                   i = i + 1
            window.sort()#sort values in the array sized 9 and pick the middle on
e as new pixel value
            final[x,y]=window[4]


ax = fig.add_subplot(nrows, ncols, 2)
ax.imshow(final,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
```

Out[180]:

```
<matplotlib.image.AxesImage at 0x19b2a4438>
```

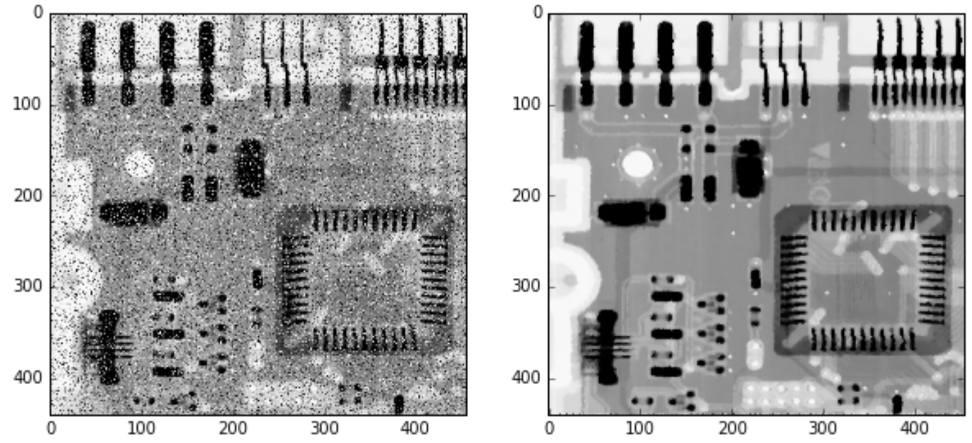


Figure 3.35(Averaging Special Filter Implementation)

In [181]:
```
    #Implementation of averaging special filter

    source = cv2.imread('Fig0335(a).tif', 0)
    nrows = 1
    ncols = 2
    ##create Figure to show raw and filtered image
    fig = plt.figure(figsize=(10, 15))
    ax = fig.add_subplot(nrows, ncols, 1)
    ax.imshow(source,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)

    final = source #matrix final will hold calculated new pixel values
    window=[1,1,1]*3 # we create kernal here
    height = np.size(source, 0)
    width = np.size(source, 1)
    edgex = math.floor((3/ 2))#used to take care of "not processing boundaries"
    edgey = math.floor((3/ 2))#used to take care of "not processing boundaries"
    #members-np.matrix([[1, 1,1],[1, 1,1],[1, 1,1]])
    for y in range(edgex,width-edgex):
        for x in range(edgey,height-edgey):
            i=0
            temp=0
            for fx in range(0,3):
               for fy in range(0,3):
                   temp = source[x + fx - edgex,y + fy - edgey]+temp #sum values
of neighbors in each itreation
                   i = i + 1
            temp=math.floor((temp/ 9)) # get the floor value of avraged 9 negighb
or values
            final[x,y]=temp # save new averaged value as new pixel value
#plot the results
ax = fig.add_subplot(nrows, ncols, 2)
ax.imshow(final,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
```

Out[181]:
```
<matplotlib.image.AxesImage at 0x18e1f5390>
```

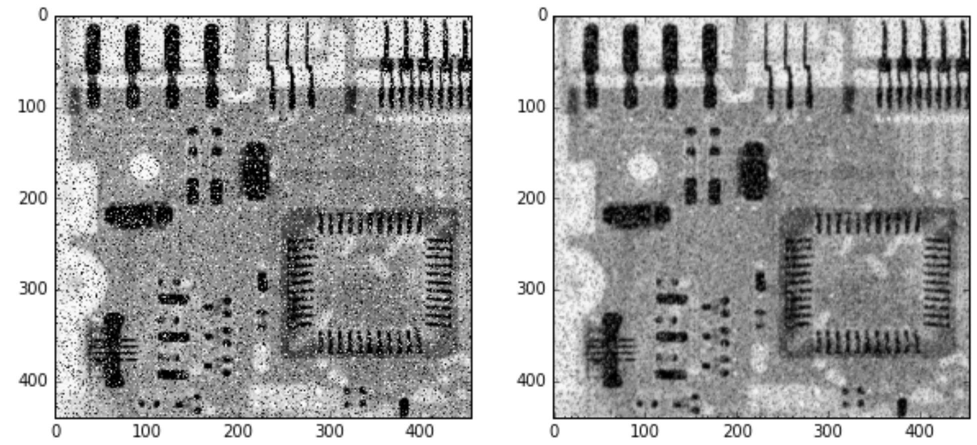
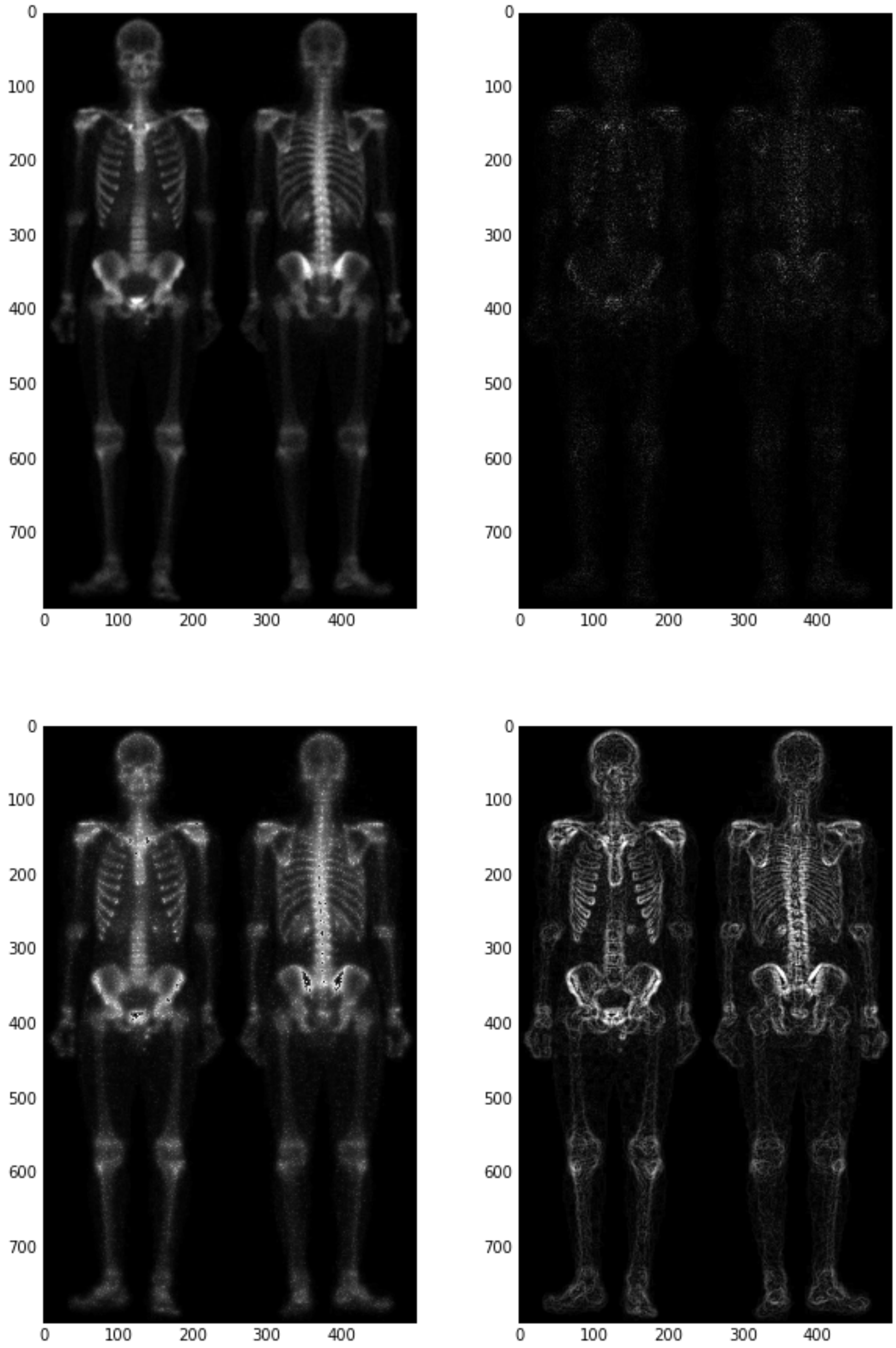


Figure 3.43(Laplacian And Sobel Of Image)

In [182]:

```
#Laplacian and Sobel Filter Example
#Sobel Filter Config Learned From :
#https://github.com/abidrahmank/OpenCV2-Python/blob/master/Official_Tutorial_Pyth
on_Codes/3_imgproc/sobel.py


img_6 = cv2.imread('Fig0343(a).tif',0)
kernel_size = 3
scale = 1
delta = 0
ddepth = cv2.CV_16S

#Generate Laplacian Image (First apply median averaging as said in book)
img_6_blur = cv2.GaussianBlur(img_6,(3,3),0)
gray_lap = cv2.Laplacian(img_6_blur,ddepth,ksize = kernel_size,scale = scale,delt
a = delta)
dst = cv2.convertScaleAbs(gray_lap)

#add Original Image to Laplacian Image
img_7=dst+img_6

#generate sobel
sobelx = cv2.Sobel(img_6_blur,cv2.CV_64F,1,0,ksize=3,scale = 2, delta = 0, border
Type = cv2.BORDER_DEFAULT)
sobely = cv2.Sobel(img_6_blur,cv2.CV_64F,0,1,ksize=3,scale = 2, delta = 0, border
Type = cv2.BORDER_DEFAULT)
abs_grad_x = cv2.convertScaleAbs(sobelx)   # converting back to uint8
abs_grad_y = cv2.convertScaleAbs(sobely)
sobel = cv2.addWeighted(abs_grad_x,0.5,abs_grad_y,0.5,0)

#Show Result
nrows = 2
ncols = 2
fig = plt.figure(figsize=(10, 15))
ax = fig.add_subplot(nrows, ncols, 1)
ax.imshow(img_6,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)

ax = fig.add_subplot(nrows, ncols, 2)
ax.imshow(dst,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)

ax = fig.add_subplot(nrows, ncols, 3)
ax.imshow(img_7,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)

ax = fig.add_subplot(nrows, ncols, 4)
ax.imshow(sobel,cmap = plt.get_cmap('gray'), vmin = 0, vmax = 255)
```

```
Out[182]: <matplotlib.image.AxesImage at 0x19bae9358>
```


0
100
200
300
400
500
600
700
0
100
200
300
400
0
100
200
300
400
500
600
700
0
100
200
300
400
0
100
200
300
400
500
600
700
0
100
200
300
400
0
100
200
300
400
500
600
700
0
100
200
300
400