Problem Statement:

The goal of this assignment is three-fold:

Enhance edges of the following image using:
a) Roberts cross operator (Please read Roberts cross operator from the book)
b) Sobel's operator
c) Apply Laplacian operator in 3x3 window and show the results.

Code

The goal of this assignment is three-fold:

a) Roberts cross operator (Please read Roberts cross operator from the book)

b) Sobel's operator

c) Apply Laplacian operator in 3x3 window and show the results.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
def padpix(x):
    m,n = x.shape
    x1 = np.zeros((m,n+2),dtype = 'uint8')
    x2 = np.zeros((m+2,n+2),dtype = 'uint8')
    x1[:,1:n+1] = x
    x1[:,0] = x[:,1]
    x1[:,n+1] = x[:,n-2]
    x2[1:m+1,:] = x1
    x2[0,:] = x1[1,:]
    x2[m+1,:] = x1[m-2,:]
    return x2
```

In [3]:
```
def convol0(x,h):
    m,n = x.shape
    x1 = np.zeros(x.shape, dtype = 'float')
    for i in range(1,m-1):
        for j in range(1,n-1):
            for ii in range(-1,2):
                for jj in range(-1,2):
                    x1[i,j] += x[i+ii,j+jj]*h[ii+1,jj+1]
    return x1[1:m+1,1:n+1]
```

In [4]:
```
def thresh(x,thr):
    xout = np.zeros(x.shape, dtype = 'uint8')
    xout[x>thr] = 255
    return xout
```

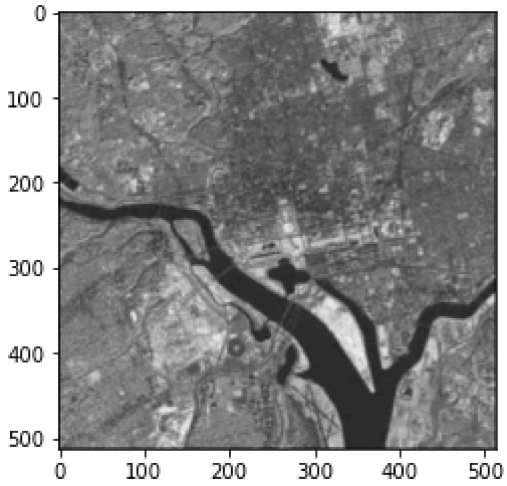
In [5]:
```
def Gfilter(filter_sz,sig):
    w = np.zeros((filter_sz,filter_sz), dtype = 'float')
    fs = filter_sz//2
    for i in range(-fs,fs):
```

```
        for j in range(-fs,fs):
            w[i,j] = 1.0*np.exp(-0.5*(i*i+j*j))/np.sqrt(2*np.pi)/sig
    return w/np.sum(w)
```

Filtered Image

In [6]:
```
x = plt.imread('washdc512.jpg')
h = Gfilter(15,1.0)
x = convol0(padpix(x),h)
plt.imshow(x,cmap = 'gray')
print(np.max(x),np.min(x))
print(x.shape)
```

```
43.10907221378113 0.0
(513, 513)
```



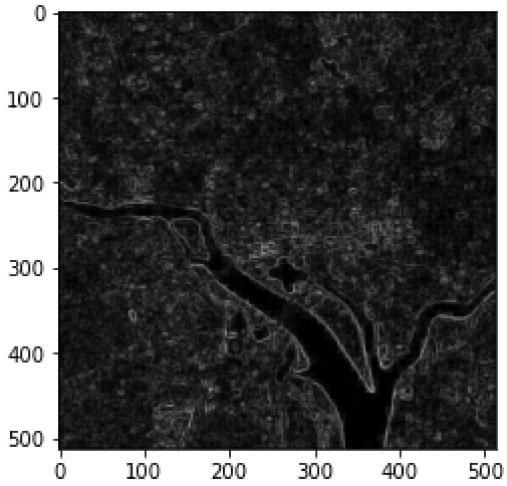
a) Robert's Operator

In [7]:
```
def hpfRoberts():
    hv = np.asarray([[0,0,0],
                     [0,1,0],
                     [0,0,-1]])

    hh = np.asarray([[0,0,0],
                     [0,0,1],
                     [0,-1,0]])
    return hh,hv
```

In [8]:
```
x = plt.imread('washdc512.jpg')
hh,hv = hpfRoberts()
yh = convol0(padpix(x),hh)
yv = convol0(padpix(x),hv)
y = np.abs(yh) + np.abs(yv)
plt.imshow(y, cmap = 'gray')
print(y.shape)
```

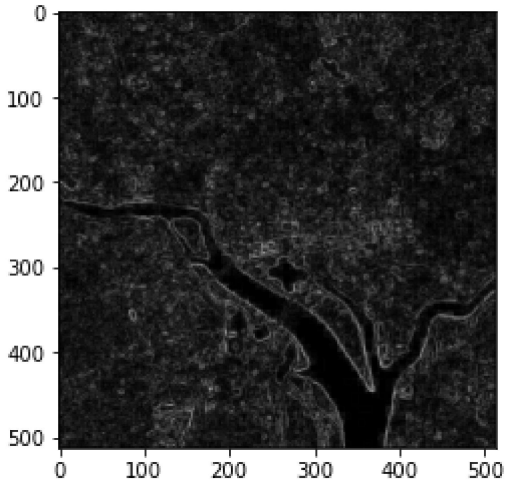
```
(513, 513)
```



In [9]:
```
y = y + np.min(y)
print(np.max(y))
y = np.uint8(y*255/np.max(y))
print(np.min(y),np.max(y),np.mean(y))
plt.imshow(y, cmap = 'gray')
```

```
76.0
0 255 28.121070490825286
```

Out[9]: <matplotlib.image.AxesImage at 0x1e55c0bfb50>

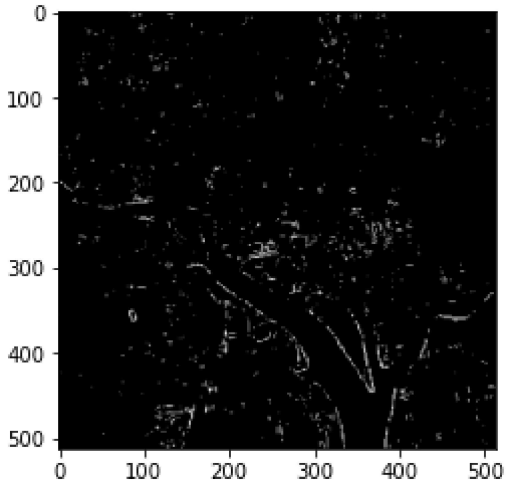


Implementing Threshold

In [10]:
```
print(np.mean(y),np.max(y),np.min(y))
xx = thresh(y,100)
plt.imshow(xx, cmap = 'gray')
```

```
28.121070490825286 255 0
```

Out[10]: <matplotlib.image.AxesImage at 0x1e55c113e80>



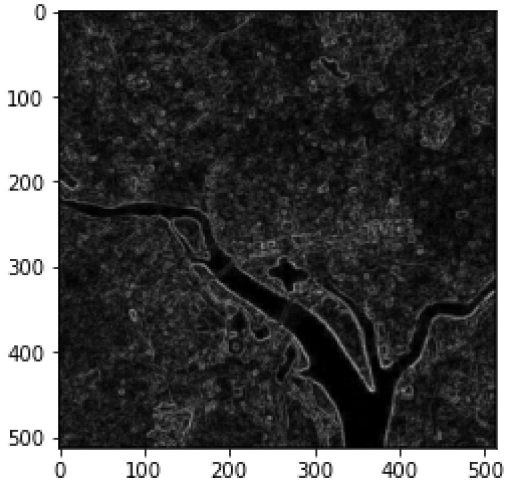
b) Sobel's Operator

In [11]:
```
def hpfSobels():
    hv = np.asarray([[1,2,1],
                     [0,0,0],
                     [-1,-2,-1]])

    hh = np.asarray([[1,0,-1],
                     [2,0,-2],
                     [1,0,-1]])
    return hh,hv
```

In [12]:
```
x = plt.imread('washdc512.jpg')
hh,hv = hpfSobels()
yh = convol0(padpix(x),hh)
yv = convol0(padpix(x),hv)
y = np.abs(yh) + np.abs(yv)
plt.imshow(y, cmap = 'gray')
print(y.shape)
```

```
(513, 513)
```

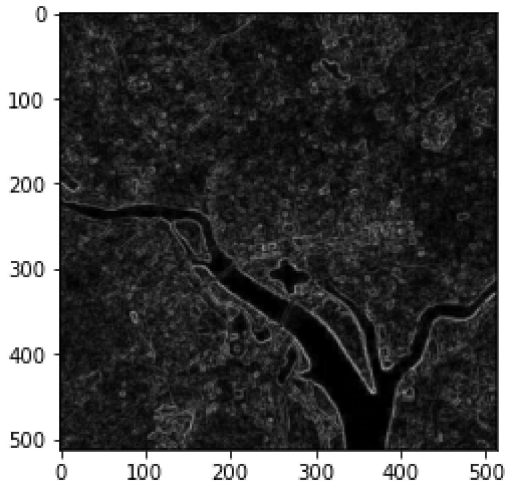


In [13]:
```
y = y + np.min(y)
print(np.max(y))
```

```
y = np.uint8(y*255/np.max(y))
print(np.min(y),np.max(y),np.mean(y))
plt.imshow(y, cmap = 'gray')
```

```
318.0
0 255 30.043470165559015
```

Out[13]: <matplotlib.image.AxesImage at 0x1e55b12e5b0>



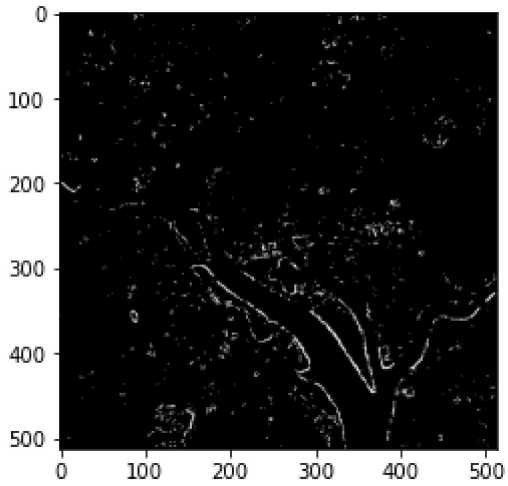
Implementing Threshold

In [14]:

```
print(np.mean(y),np.max(y),np.min(y))
xx = thresh(y,100)
plt.imshow(xx, cmap = 'gray')
```

```
30.043470165559015 255 0
```

Out[14]: <matplotlib.image.AxesImage at 0x1e55b180910>



c) Apply Laplacian operator in 3x3 window and show the results.

In [15]:

```
def hpfLaplacian():
    hv = np.asarray([[0,1,0],
```

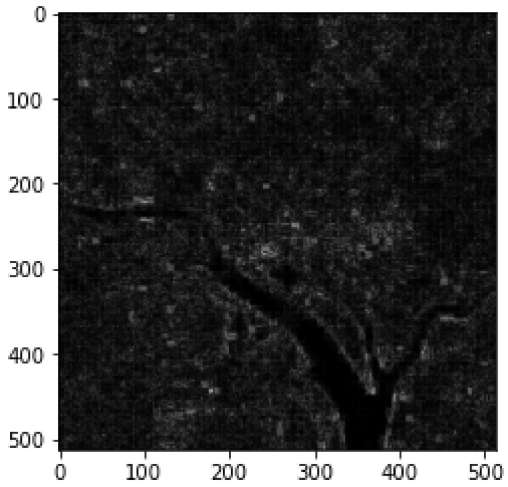
```
                    [1,-4,1],
                    [0,1,0]])

    hh = np.asarray([[-1,-1,-1],
                    [-1,8,-1],
                    [-1,-1,-1]])
    return hh,hv
```

In [16]:

```
x = plt.imread('washdc512.jpg')
hh,hv = hpfLaplacian()
yh = convol0(padpix(x),hh)
yv = convol0(padpix(x),hv)
y = np.abs(yh) + np.abs(yv)
plt.imshow(y, cmap = 'gray')
print(y.shape)
```

```
(513, 513)
```

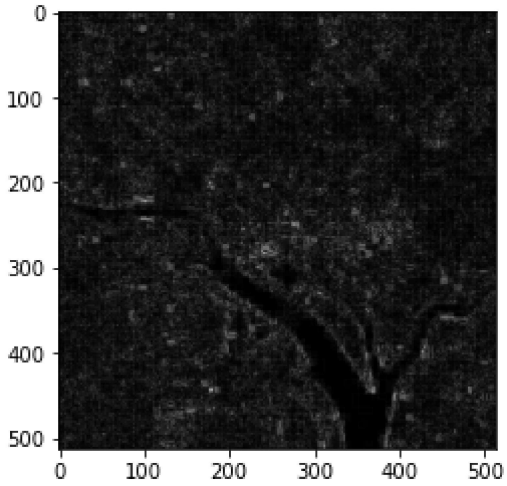


In [17]:

```
y = y + np.min(y)
print(np.max(y))
y = np.uint8(y*255/np.max(y))
print(np.min(y),np.max(y),np.mean(y))
plt.imshow(y, cmap = 'gray')
```

```
252.0
0 255 21.310754686152244
```

Out[17]: <matplotlib.image.AxesImage at 0x1e55b23c100>

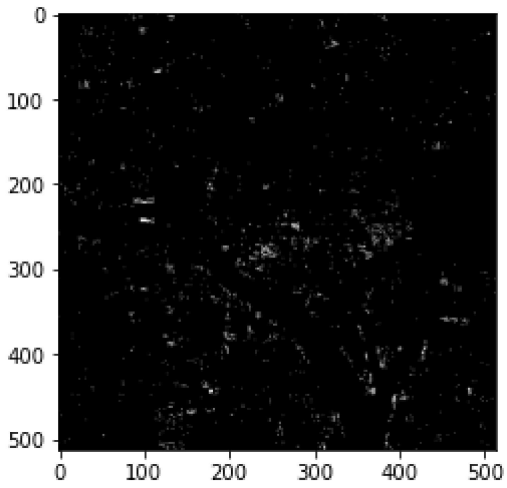


Implementing Threshold

In [20]:
```
print(np.mean(y),np.max(y),np.min(y))
xx = thresh(y,100)
plt.imshow(xx, cmap = 'gray')
```

```
21.310754686152244 255 0
```

Out[20]:
```
<matplotlib.image.AxesImage at 0x1e55b265c40>
```



Conclusion:

Implementing the three methods, which are Robert's, Sobel's and Laplacian, all can generate images with edge detection, and after implementing the threshold value of 100, the edges of the image become like a "line drawing" and are able to define the edges very well. We are also able to find where the gradient values are high at a threshold value of 100. At lower threshold values the edges become spurious or noisy. If we compare the edges generated using Sobel's method vs Robert's cross method, we notice that with Sobel's operator, generating edges tends to be slower than Robert's cross operator. However, due to its larger convolution kernel, it can smooth the input image to a greater extent and thus making the operator less sensitive to noise. We also notice that the edges produced using Sobel's method produces considerably higher output values for similar edges, compared to Robert Cross. If we compare Laplacian operator to Sobel's method, we notice that the edges produced using Laplacian tend to be a lot less smooth. We also notice that with a threshold value of 100, the edges are a lot more noisy when compared with Sobel's and Robert's operator, thus making it difficult to define the edges and finding where the gradient values are high.