Digital Image Processing Assignment 3

MTH Junaidi AP17110010074 of CSE B

Lab 03: Intensity transformation (Spatial Filtering)

In [26]:

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
import cv2
```

In [27]:

```
lena = mpimg.imread("lena.png")
```

In [28]:

```
lena.shape
```

Out[28]:

```
(256, 256, 3)
```

Averaging

In [29]:

```
temp1 =[]
temp2 = []
for i in range(256):
    for j in range(256):
        temp2.append(sum(lena[i][j])/3)
    temp1.append(temp2)
    temp2 =[]
```

In [30]:

```
image = np.array(temp1)
```

In [31]:

```
image.shape
```

Out[31]:

```
(256, 256)
```

In [32]:

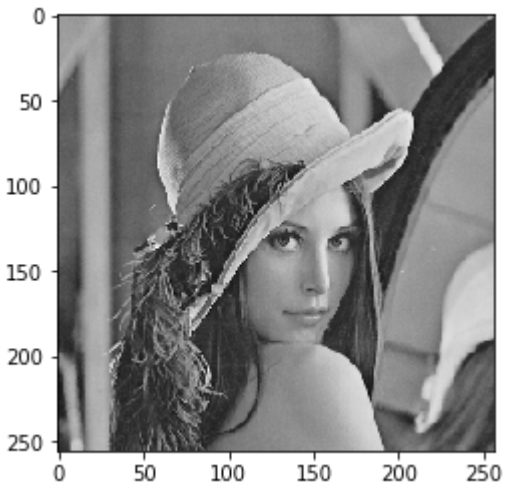
```
image = image*255
```

In [33]:

```
plt.imshow(image, cmap='gray', vmin=0, vmax=255)
```

Out[33]:

```
<matplotlib.image.AxesImage at 0x122f4ebd0>
```



In [34]:

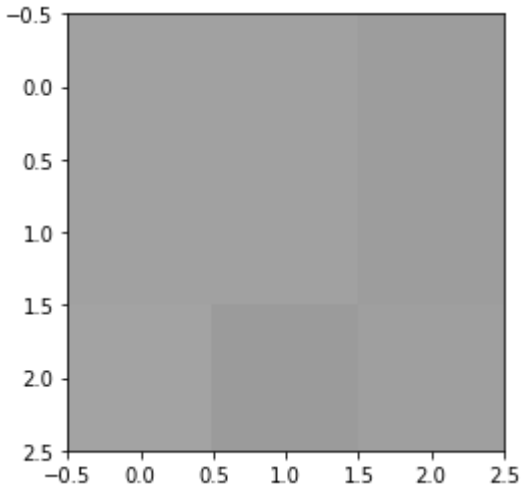
```
smallimg = image[:3,:3]
```

In [35]:

```
plt.imshow(smallimg, cmap='gray', vmin=0, vmax=255)
```

Out[35]:

```
<matplotlib.image.AxesImage at 0x12322f950>
```



In [36]:

```
smallimg
```

Out[36]:

```
array([[161.0000056 , 161.0000056 , 157.00000584],
       [161.0000056 , 161.0000056 , 157.00000584],
       [163.00000548, 155.00000596, 159.00000572]])
```

In [37]:

```
smallimg.shape
```

Out[37]:

```
(3, 3)
```

In [85]:

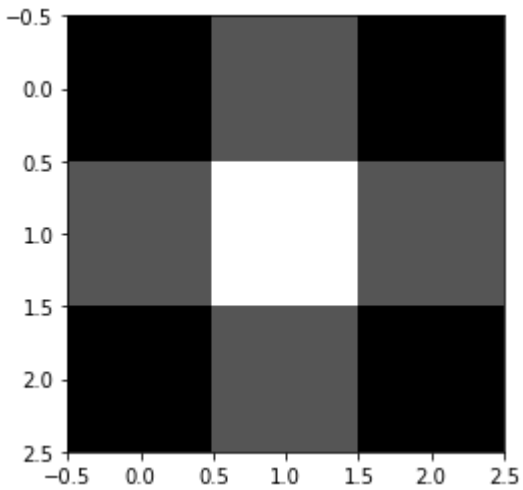
```
filter_window = [[1/4, 2/4, 1/4],
                 [2/4, 4/4, 2/4],
                 [1/4, 2/4, 1/4]]
```

In [87]:

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

Out[87]:

```
<matplotlib.image.AxesImage at 0x1c26f4d9d0>
```



In [88]:

```
#This function takes the filter matrix, image and pixel location as input and re
turns the output pixel intensity
def pixel_filter_Application(filter_window, image,x,y):
    temp = 0
    for i in range(-(Filter_dim-2), (Filter_dim-1)):
        for j in range (-(Filter_dim-2), (Filter_dim-1)):
            temp += image[x+j][y+i]*filter_window[i+1][j+1]
    temp=temp/sum(sum(filter_window))
    return temp
```

In [92]:

```
filter_window = np.array(filter_window)
```

In [93]:

```
temp2 =[]
for i in range(1,image.shape[0]-1):
    temp1 = []
    for j in range(1,image.shape[1]-1):
        temp1.append(pixel_filter_Application(filter_window , image, i , j ))
    temp2.append(temp1)
```

In [94]:

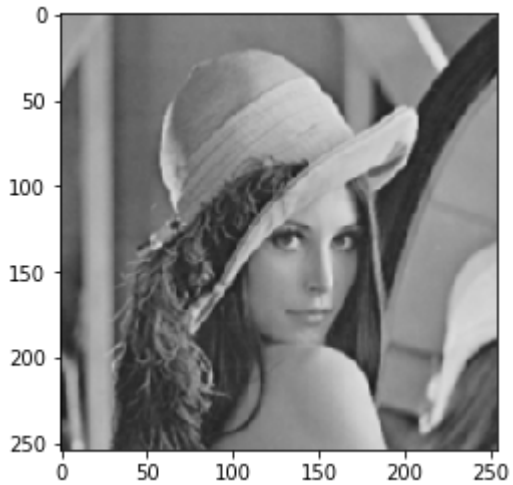
```
Averaged_image = np.array(temp2)
```

In [95]:

```
plt.imshow(Averaged_image, cmap='gray', vmin=0, vmax=255)
```

Out[95]:

```
<matplotlib.image.AxesImage at 0x1c26f2ac50>
```



In [38]:

```
Filter_dim =3
```

In [98]:

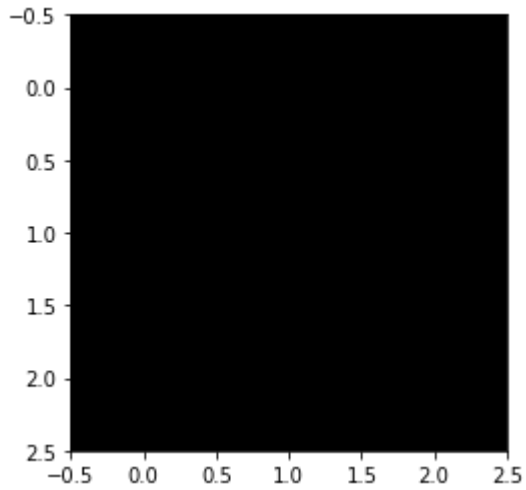
```
filter_window = []
for i in range(Filter_dim):
    temp =[ ]
    for j in range (Filter_dim):
        temp.append(1)
    filter_window.append(temp)
```

In [99]:

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

Out[99]:

```
<matplotlib.image.AxesImage at 0x1c27680590>
```



In [100]:

```
filter_window = np.array(filter_window)
```

In [101]:

```
sum(sum(smallimg))/9
```

Out[101]:

```
159.4444501399994
```

In [103]:

```
#This function takes the filter matrix, image and pixel location as input and re
turns the output pixel intensity
def pixel_filter_Application(filter_window, image,x,y):
    temp = 0
    for i in range(-(Filter_dim-2), (Filter_dim-1)):
        for j in range (-(Filter_dim-2), (Filter_dim-1)):
            temp += image[x+j][y+i]*filter_window[i+1][j+1]
    temp=temp/sum(sum(filter_window))
    return temp
```

In [104]:

```
pixel_filter_Application(filter_window , smallimg, 1,1)
```

Out[104]:

```
159.4444501399994
```

In [105]:

```
image[1][1]
```

Out[105]:

```
161.0000056028366
```

In [106]:

```
temp2 =[]
for i in range(1,image.shape[0]-1):
    temp1 = []
    for j in range(1,image.shape[1]-1):
        temp1.append(pixel_filter_Application(filter_window , image, i , j ))
    temp2.append(temp1)
```

In [107]:

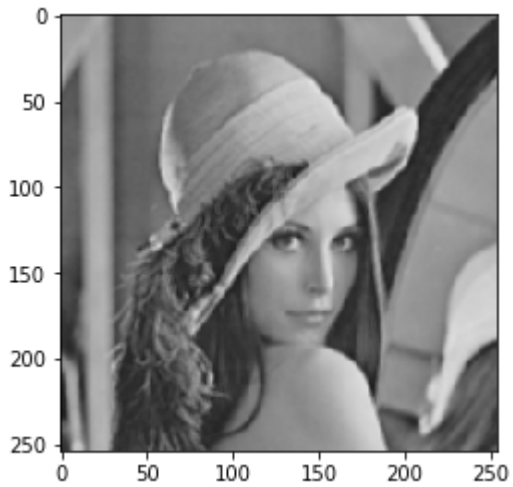
```
Averaged_image = np.array(temp2)
```

In [108]:

```
plt.imshow(Averaged_image, cmap='gray', vmin=0, vmax=255)
```

Out[108]:

```
<matplotlib.image.AxesImage at 0x1c27478190>
```



In [62]:

```
filter_window = []
for i in range(Filter_dim):
    temp =[ ]
    for j in range (Filter_dim):
        a = min( i+1 , Filter_dim - i)
        b = min(j+1 , Filter_dim - j)
        m =  a+b
        temp.append(m)

    filter_window.append(temp)
```

In [63]:

```
filter_window = np.array(filter_window)
```

In [64]:

```
temp2 =[]
for i in range(1,image.shape[0]-1):
    temp1 = []
    for j in range(1,image.shape[1]-1):
        temp1.append(pixel_filter_Application(filter_window , image, i , j ))
    temp2.append(temp1)
```

In [65]:

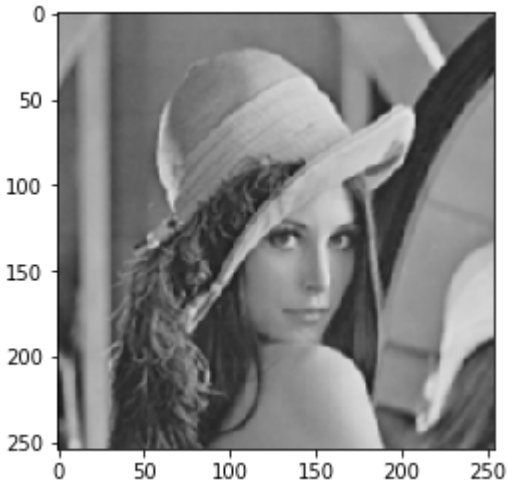
```
Averaged_image = np.array(temp2)
```

In [66]:

```
plt.imshow(Averaged_image, cmap='gray', vmin=0, vmax=255)
```

Out[66]:

```
<matplotlib.image.AxesImage at 0x1229e1e90>
```



Median Filtering

In [67]:

```
#This function takes the filter matrix, image and pixel location as input and re
turns the output pixel intensity
def pixel_filter_Application_median(filter_window, image,x,y):
    temp = []
    for i in range(-(Filter_dim-2), (Filter_dim-1)):
        for j in range (-(Filter_dim-2), (Filter_dim-1)):
            temp.append(image[x+j][y+i]*filter_window[i+1][j+1])
    temp=np.median(temp)
    return temp
```

In [68]:

```
filter_window = []
for i in range(Filter_dim):
    temp =[ ]
    for j in range (Filter_dim):
        temp.append(1)
    filter_window.append(temp)
```

In [69]:

```
filter_window
```

Out[69]:

```
[[1, 1, 1], [1, 1, 1], [1, 1, 1]]
```

In [70]:

```
temp2 =[]
for i in range(1,image.shape[0]-1):
    temp1 = []
    for j in range(1,image.shape[1]-1):
        temp1.append(pixel_filter_Application_median(filter_window , image, i ,
j ))
    temp2.append(temp1)
```

In [71]:

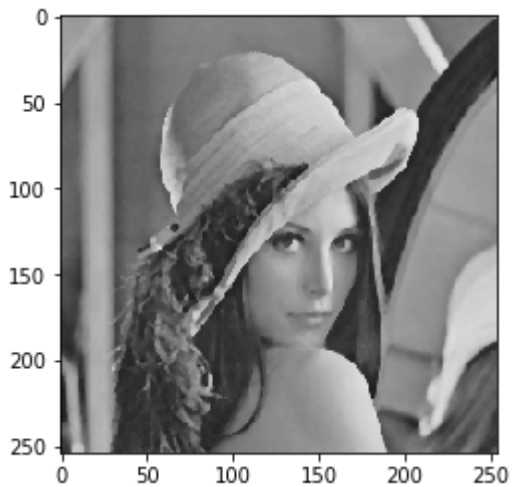
```
Averaged_image = np.array(temp2)
```

In [72]:

```
plt.imshow(Averaged_image, cmap='gray', vmin=0, vmax=255)
```

Out[72]:

```
<matplotlib.image.AxesImage at 0x123867710>
```



Max Filtering

In [73]:

```
#This function takes the filter matrix, image and pixel location as input and re
turns the output pixel intensity
def pixel_filter_Application_max(filter_window, image,x,y):
    temp = []
    for i in range(-(Filter_dim-2), (Filter_dim-1)):
        for j in range (-(Filter_dim-2), (Filter_dim-1)):
            temp.append(image[x+j][y+i]*filter_window[i+1][j+1])
    temp=max(temp)
    return temp
```

In [74]:

```
temp2 =[]
for i in range(1,image.shape[0]-1):
    temp1 = []
    for j in range(1,image.shape[1]-1):
        temp1.append(pixel_filter_Application_max(filter_window , image, i , j
))
    temp2.append(temp1)
```

In [75]:

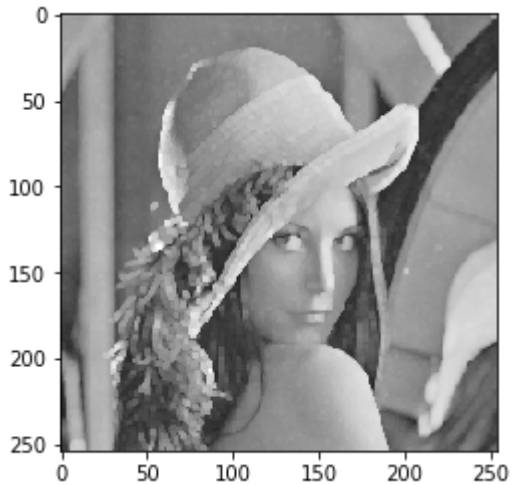
```
max_image = np.array(temp2)
```

In [76]:

```
plt.imshow(max_image, cmap='gray', vmin=0, vmax=255)
```

Out[76]:

```
<matplotlib.image.AxesImage at 0x108f4ba10>
```



Min FIltering

In [77]:

```
#This function takes the filter matrix, image and pixel location as input and re
turns the output pixel intensity
def pixel_filter_Application_min(filter_window, image,x,y):
    temp = []
    for i in range(-(Filter_dim-2), (Filter_dim-1)):
        for j in range (-(Filter_dim-2), (Filter_dim-1)):
            temp.append(image[x+j][y+i]*filter_window[i+1][j+1])
    temp=min(temp)
    return temp
```

In [78]:

```
temp2 =[]
for i in range(1,image.shape[0]-1):
    temp1 = []
    for j in range(1,image.shape[1]-1):
        temp1.append(pixel_filter_Application_min(filter_window , image, i , j
))
    temp2.append(temp1)
```

In [79]:

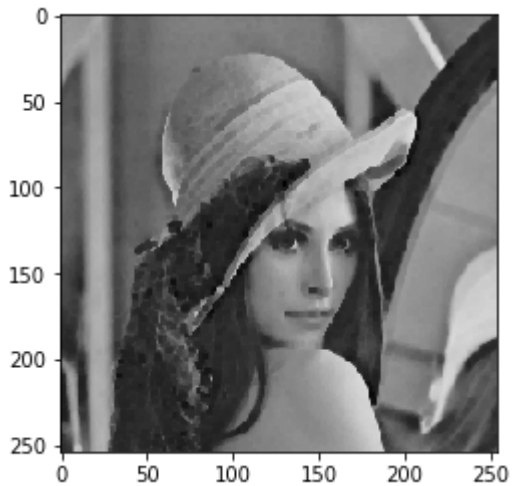
```
min_image = np.array(temp2)
```

In [80]:

```
plt.imshow(min_image, cmap='gray', vmin=0, vmax=255)
```

Out[80]:

```
<matplotlib.image.AxesImage at 0x123d49ad0>
```



Question 2

Image Operations on Images with Salt and pepper noise

In [148]:

```
gauss = np.random.normal(0,1,image.size)
```

In [150]:

```
gauss = gauss.reshape(img.shape[0],img.shape[1]).astype('uint8')
```

In [154]:

```
noise_image = img + img * gauss
```

In [178]:

```
np.amax(noise_image)
```

Out[178]:

```
239.93725490196078
```

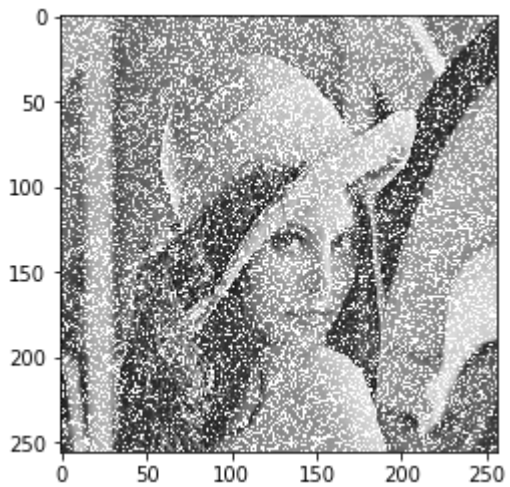
This is the Image we will operate on (With salt and pepper noise)

In [156]:

```
plt.imshow(noise_image, cmap='gray', vmin=0, vmax=1)
```

Out[156]:

```
<matplotlib.image.AxesImage at 0x1c2c417b10>
```



Noraml Average

In [218]:

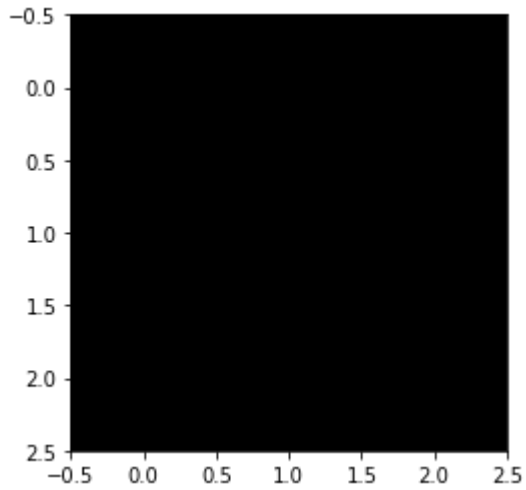
```
filter_window = []
for i in range(Filter_dim):
    temp =[ ]
    for j in range (Filter_dim):
        temp.append(1)
    filter_window.append(temp)
```

In [219]:

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

Out[219]:

```
<matplotlib.image.AxesImage at 0x1c2e534950>
```



In [220]:

```
filter_window = np.array(filter_window)
```

In [226]:

```
#This function takes the filter matrix, image and pixel location as input and re
turns the output pixel intensity
def pixel_filter_Application(filter_window, image,x,y):
    temp = 0
    for i in range(-(Filter_dim-2), (Filter_dim-1)):
        for j in range (-(Filter_dim-2), (Filter_dim-1)):
            temp += image[x+j][y+i]*filter_window[i+1][j+1]
    temp=temp/sum(sum(filter_window))
    return temp
```

In [227]:

```
sum(sum(filter_window))
```

Out[227]:

```
9
```

In [228]:

```
temp2 =[]
for i in range(1,noise_image.shape[0]-1):
    temp1 = []
    for j in range(1,noise_image.shape[1]-1):
        temp1.append(pixel_filter_Application(filter_window , noise_image, i , j
))
    temp2.append(temp1)
```

In [229]:

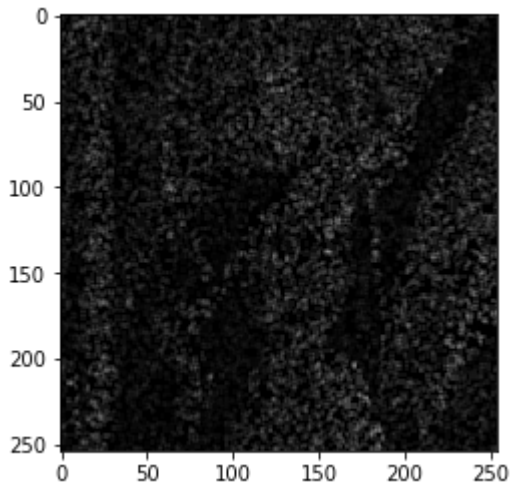
```
averaged_noise_image = np.array(temp2)
```

In [230]:

```
plt.imshow(averaged_noise_image, cmap='gray', vmin=0, vmax=255)
```

Out[230]:

```
<matplotlib.image.AxesImage at 0x1c2ed91b50>
```



In [231]:

```
np.amax(averaged_noise_image)
```

Out[231]:

```
139.7111111111111
```

Weighted Average

In [202]:

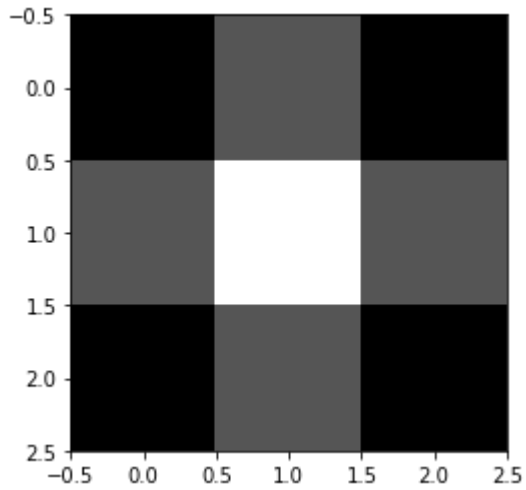
```
filter_window = [[1/4, 2/4, 1/4],
                 [2/4, 4/4, 2/4],
                 [1/4, 2/4, 1/4]]
```

In [203]:

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

Out[203]:

```
<matplotlib.image.AxesImage at 0x1c2e454810>
```



In [204]:

```
filter_window = np.array(filter_window)
```

In [205]:

```
temp2 =[]
for i in range(1,noise_image.shape[0]-1):
    temp1 = []
    for j in range(1,noise_image.shape[1]-1):
        temp1.append(pixel_filter_Application(filter_window , noise_image, i , j
))
    temp2.append(temp1)
```

In [206]:

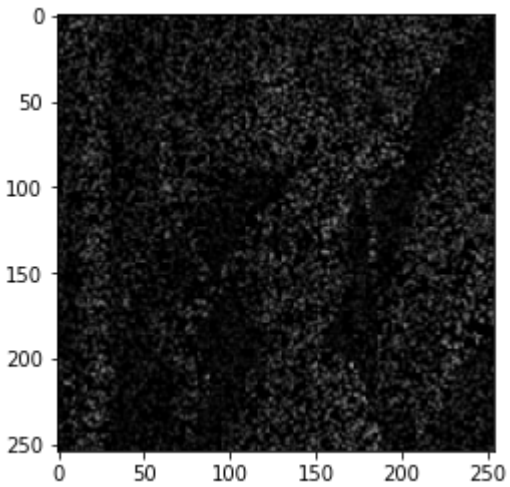
```
Weighted_averaged_noise_image = np.array(temp2)
```

In [207]:

```
plt.imshow(Weighted_averaged_noise_image, cmap='gray', vmin=0, vmax=255)
```

Out[207]:

```
<matplotlib.image.AxesImage at 0x1c2de1f3d0>
```



Median Filtering

In [232]:

```
temp2 =[]
for i in range(1,image.shape[0]-1):
    temp1 = []
    for j in range(1,image.shape[1]-1):
        temp1.append(pixel_filter_Application_median(filter_window , noise_image
, i , j ))
    temp2.append(temp1)
```

In [233]:

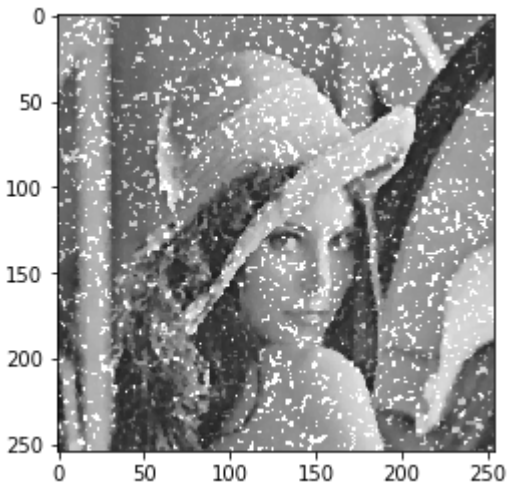
```
Median_noise_image = np.array(temp2)
```

In [235]:

```
plt.imshow(Median_noise_image, cmap='gray', vmin=0, vmax=1)
```

Out[235]:

```
<matplotlib.image.AxesImage at 0x1c2f76b210>
```



WE see a Immese Improvement from averaging to Median filtering

In [238]:

```
temp2 =[]
for i in range(1,image.shape[0]-1):
    temp1 = []
    for j in range(1,image.shape[1]-1):
        temp1.append(pixel_filter_Application_max(filter_window , noise_image, i
, j ))
    temp2.append(temp1)
```

In [239]:

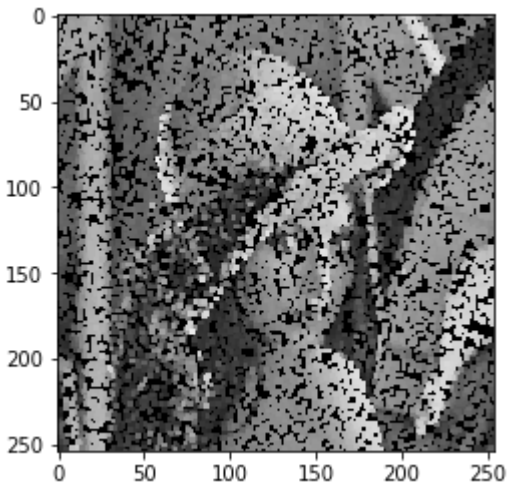
```
Max_noise_image = np.array(temp2)
```

In [242]:

```
plt.imshow(Max_noise_image, cmap='gray', vmin=0, vmax=255)
```

Out[242]:

```
<matplotlib.image.AxesImage at 0x1c300a0190>
```



In [245]:

```
temp2 =[]
for i in range(1,image.shape[0]-1):
    temp1 = []
    for j in range(1,image.shape[1]-1):
        temp1.append(pixel_filter_Application_min(filter_window , noise_image, i
, j ))
    temp2.append(temp1)
```

In [246]:

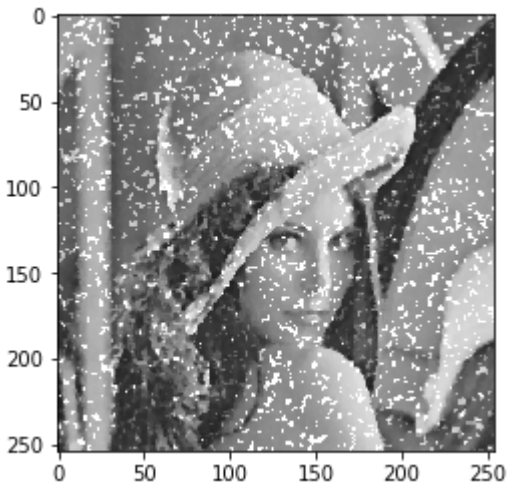
```
Min_noise_image = np.array(temp2)
```

In [247]:

```
plt.imshow(Median_noise_image, cmap='gray', vmin=0, vmax=1)
```

Out[247]:

```
<matplotlib.image.AxesImage at 0x1c302328d0>
```



Question 3

Laplation Filter

In [248]:

```
laplation_filter = [[0, -1, 0],
                    [-1, 4, -1],
                    [0, -1, 0]]
```

In [250]:

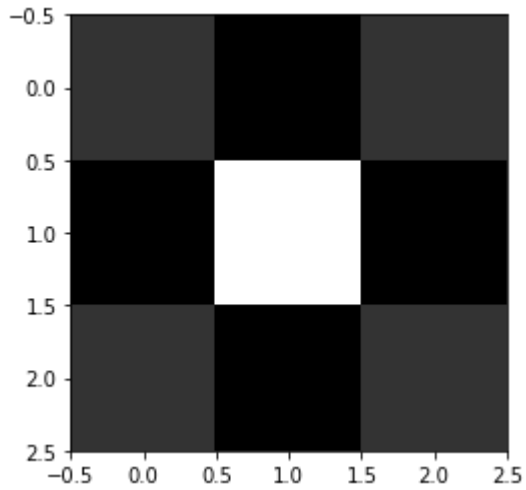
```
laplation_filter = np.array(laplation_filter)
```

In [253]:

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

Out[253]:

```
<matplotlib.image.AxesImage at 0x1c30549910>
```



In [251]:

```
sum(sum(laplation_filter))
```

Out[251]:

```
0
```

In [264]:

```
#This function takes the filter matrix, image and pixel location as input and re
turns the output pixel intensity
def Laplation_pixel_filter_Application(laplation_filter, image,x,y):
    temp = 0
    for i in range(-(Filter_dim-2), (Filter_dim-1)):
        for j in range (-(Filter_dim-2), (Filter_dim-1)):
            temp += image[x+j][y+i]*laplation_filter[i+1][j+1]
    return temp
```

In [265]:

```
temp2 =[]
for i in range(1,noise_image.shape[0]-1):
    temp1 = []
    for j in range(1,noise_image.shape[1]-1):
        temp1.append(Laplation_pixel_filter_Application(laplation_filter , image
, i , j ))
    temp2.append(temp1)
```

In [266]:

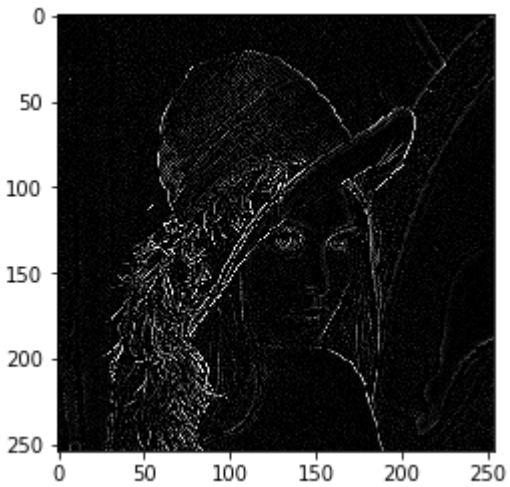
```
Laplation_image = np.array(temp2)
```

In [269]:

```
plt.imshow(Laplation_image, cmap='gray', vmin=0, vmax=255)
```

Out[269]:

```
<matplotlib.image.AxesImage at 0x1c2ec729d0>
```



Unsharp Filtering

In [338]:

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

In [339]:

```
unsharp_filter = np.array(unsharp_filter)
```

In [340]:

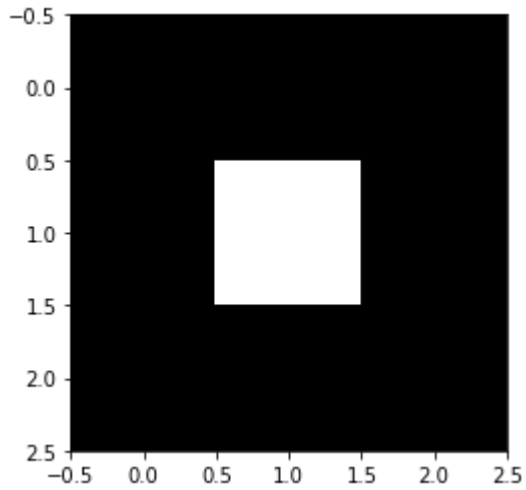
```
unsharp_filter = unsharp_filter/8
```

In [341]:

```
plt.imshow(unsharp_filter, cmap="gray")
```

Out[341]:

```
<matplotlib.image.AxesImage at 0x1c32ddc550>
```



In [342]:

```
#This function takes the filter matrix, image and pixel location as input and re
turns the output pixel intensity
def unsharp_pixel_filter_Application(unsharp_filter, image,x,y):
    temp = 0
    for i in range(-(Filter_dim-2), (Filter_dim-1)):
        for j in range (-(Filter_dim-2), (Filter_dim-1)):
            temp += image[x+j][y+i]*unsharp_filter[i+1][j+1]
    return temp
```

In [343]:

```
temp2 =[]
for i in range(1,noise_image.shape[0]-1):
    temp1 = []
    for j in range(1,noise_image.shape[1]-1):
        temp1.append(unsharp_pixel_filter_Application(unsharp_filter , image, i
, j ))
    temp2.append(temp1)
```

In [344]:

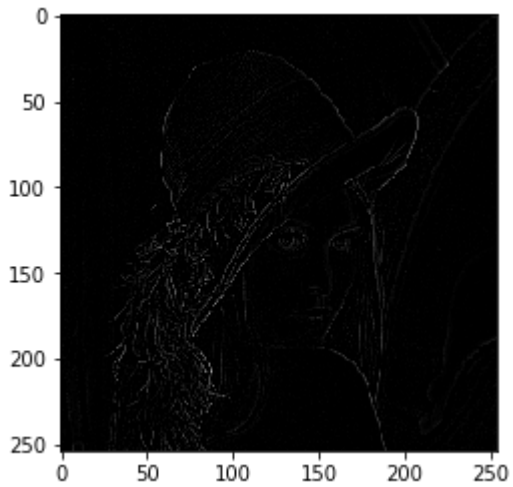
```
unsharp_image = np.array(temp2)
```

In [347]:

```
plt.imshow(unsharp_image, cmap='gray', vmin=0, vmax=255)
```

Out[347]:

```
<matplotlib.image.AxesImage at 0x1c32ff4990>
```



High Boost Filtering

In [348]:

```
# for K = 2 let us use this filter

HB_filter = [[-1, -1, -1],
                    [-1, 15, -1],
                    [-1, -1, -1]]
```

In [349]:

```
HB_filter = np.array(HB_filter)
```

In [350]:

```
HB_filter = HB_filter/9
```

In [351]:

```
#This function takes the filter matrix, image and pixel location as input and re
turns the output pixel intensity
def highboost_pixel_filter_Application(HB_filter, image,x,y):
    temp = 0
    for i in range(-(Filter_dim-2), (Filter_dim-1)):
        for j in range (-(Filter_dim-2), (Filter_dim-1)):
            temp += image[x+j][y+i]*HB_filter[i+1][j+1]
    return temp
```

In [352]:

```
temp2 =[]
for i in range(1,noise_image.shape[0]-1):
    temp1 = []
    for j in range(1,noise_image.shape[1]-1):
        temp1.append(highboost_pixel_filter_Application(HB_filter , image, i , j
))
    temp2.append(temp1)
```

In [353]:

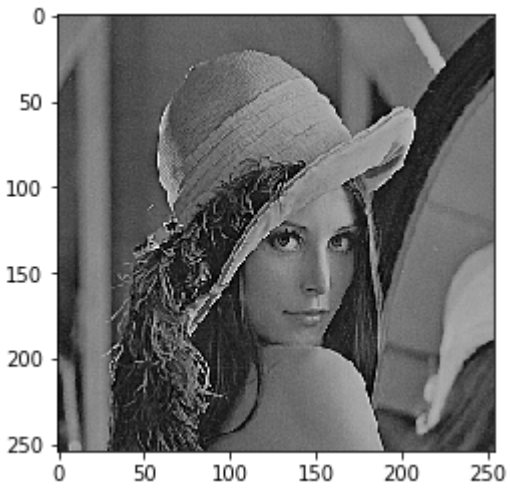
```
HB_image = np.array(temp2)
```

In [354]:

```
plt.imshow(HB_image, cmap='gray', vmin=0, vmax=255)
```

Out[354]:

```
<matplotlib.image.AxesImage at 0x1c31e8c4d0>
```



Sobel Operator

In [355]:

```
# Vertical Filter
Sobel_filter = [[-1, 0, 1],
                [-2, 0, 2],
                [-1, 0, 1]]
```

In [356]:

```
Sobel_filter = np.array(Sobel_filter)
```

In [357]:

```
#This function takes the filter matrix, image and pixel location as input and re
turns the output pixel intensity
def Sobel_pixel_filter_Application(Sobel_filter, image,x,y):
    temp = 0
    for i in range(-(Filter_dim-2), (Filter_dim-1)):
        for j in range (-(Filter_dim-2), (Filter_dim-1)):
            temp += image[x+j][y+i]*Sobel_filter[i+1][j+1]
    return temp
```

In [358]:

```
temp2 =[]
for i in range(1,noise_image.shape[0]-1):
    temp1 = []
    for j in range(1,noise_image.shape[1]-1):
        temp1.append(Sobel_pixel_filter_Application(Sobel_filter , image, i , j
))
    temp2.append(temp1)
```

In [359]:

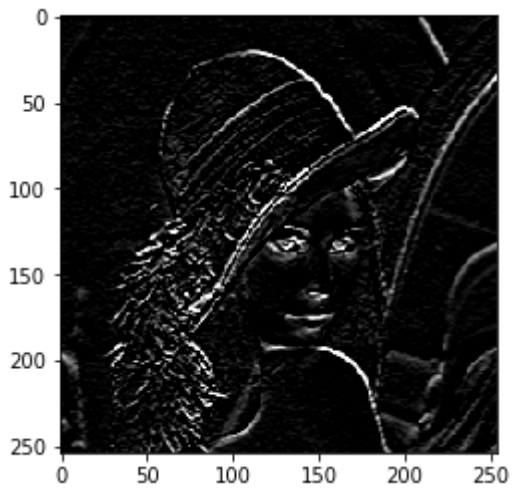
```
Sobel_image = np.array(temp2)
```

In [360]:

```
plt.imshow(Sobel_image, cmap='gray', vmin=0, vmax=255)
```

Out[360]:

```
<matplotlib.image.AxesImage at 0x1c3167f190>
```



Prewitt Operator

In [362]:

```
Perwitt_filter = [[-1, 0, 1],
                  [-1, 0, 1],
                  [-1, 0, 1]]
```

In [365]:

```
Perwitt_filter = np.array(Perwitt_filter)
```

In [366]:

```
#This function takes the filter matrix, image and pixel location as input and re
turns the output pixel intensity
def Prewitt_pixel_filter_Application(Perwitt_filter, image,x,y):
    temp = 0
    for i in range(-(Filter_dim-2), (Filter_dim-1)):
        for j in range (-(Filter_dim-2), (Filter_dim-1)):
            temp += image[x+j][y+i]*Perwitt_filter[i+1][j+1]
    return temp
```

In [367]:

```
temp2 =[]
for i in range(1,noise_image.shape[0]-1):
    temp1 = []
    for j in range(1,noise_image.shape[1]-1):
        temp1.append(Prewitt_pixel_filter_Application(Perwitt_filter , image, i
, j ))
    temp2.append(temp1)
```

In [368]:

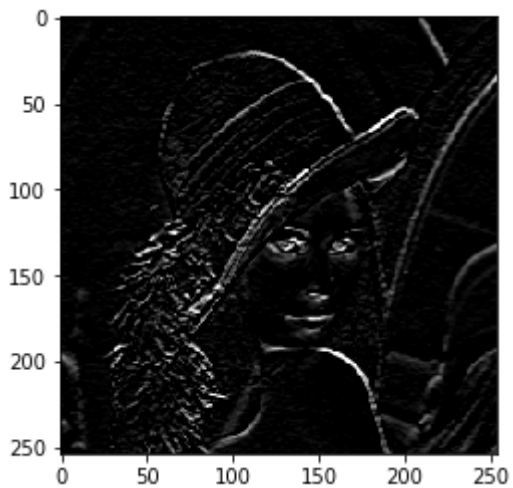
```
Perwitt_image = np.array(temp2)
```

In [369]:

```
plt.imshow(Perwitt_image, cmap='gray', vmin=0, vmax=255)
```

Out[369]:

```
<matplotlib.image.AxesImage at 0x1c2eebba90>
```



In [ ]:

```

```