Name: Muhammad Haroon Khan Bangash

Roll no: 303643

Degree: DE 41 EE (A)

Subject: Dip

Assignment: 02

Disclaimer:

Some of the tasks are not done due to the lack of coding knowledge and limited resources on the internet. Although I have used dozens of websites but some codes weren't on internet as demanded by professor. I could have switched to Matlab for obtaining grades but it would have killed my curiosiy for assignment and learning objective. Incomplete tasks are compensated with materials that were available on different websites.

Importing the picture

In [1]:

```python
import numpy as np
import cv2
import os
from matplotlib import pyplot as plt
from PIL import Image, ImageFilter
%matplotlib inline
```

In [1]:

```python
!pip install pyppeteer
```

```
Collecting pyppeteer
  Downloading pyppeteer-1.0.2-py3-none-any.whl (83 kB)
Requirement already satisfied: tqdm<5.0.0,>=4.42.1 in c:\users\hp\anaconda3\lib\site-pac
kages (from pyppeteer) (4.64.0)
Collecting pyee<9.0.0,>=8.1.0
  Downloading pyee-8.2.2-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: certifi>=2021 in c:\users\hp\anaconda3\lib\site-packages
(from pyppeteer) (2022.12.7)
Requirement already satisfied: urllib3<2.0.0,>=1.25.8 in c:\users\hp\anaconda3\lib\site-
packages (from pyppeteer) (1.26.9)
Requirement already satisfied: importlib-metadata>=1.4 in c:\users\hp\anaconda3\lib\site
-packages (from pyppeteer) (4.11.3)
Requirement already satisfied: appdirs<2.0.0,>=1.4.3 in c:\users\hp\anaconda3\lib\site-p
ackages (from pyppeteer) (1.4.4)
Collecting websockets<11.0,>=10.0
  Downloading websockets-10.4-cp39-cp39-win_amd64.whl (101 kB)
Requirement already satisfied: zipp>=0.5 in c:\users\hp\anaconda3\lib\site-packages (fro
m importlib-metadata>=1.4->pyppeteer) (3.7.0)
Requirement already satisfied: colorama in c:\users\hp\anaconda3\lib\site-packages (from
tqdm<5.0.0,>=4.42.1->pyppeteer) (0.4.4)
Installing collected packages: websockets, pyee, pyppeteer
Successfully installed pyee-8.2.2 pyppeteer-1.0.2 websockets-10.4
```

In [2]:

```python
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
```

In [3]:

```
image = cv2.imread('haroon.jpeg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
plt.figure(figsize=(11,6))
plt.imshow(image)
plt.title('Image')
plt.xticks([])
plt.yticks([])
plt.show()
```

In [4]:

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

In [6]:

```
img=cv2.imread("haroon.jpeg")

print(img.shape)
```

```
(1600, 1200, 3)
```

Adding Gaussian Noise to image

In [35]:

```
gauss_noise=np.zeros((640,480),dtype=np.uint8)
cv2.randn(gauss_noise,128,20)
gauss_noise=(gauss_noise*0.5).astype(np.uint8)
```

In [36]:

```
gn_img=cv2.add(img,gauss_noise)
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
Input In [36], in <cell line: 1>()
----> 1 gn_img=cv2.add(img,gauss_noise)

error: OpenCV(4.6.0) D:\a\opencv-python\opencv-python\opencv\modules\core\src\arithm.cp
p:650: error: (-209:Sizes of input arguments do not match) The operation is neither 'arr
ay op array' (where arrays have the same size and the same number of channels), nor 'arr
ay op scalar', nor 'scalar op array' in function 'cv::arithm_op'
```

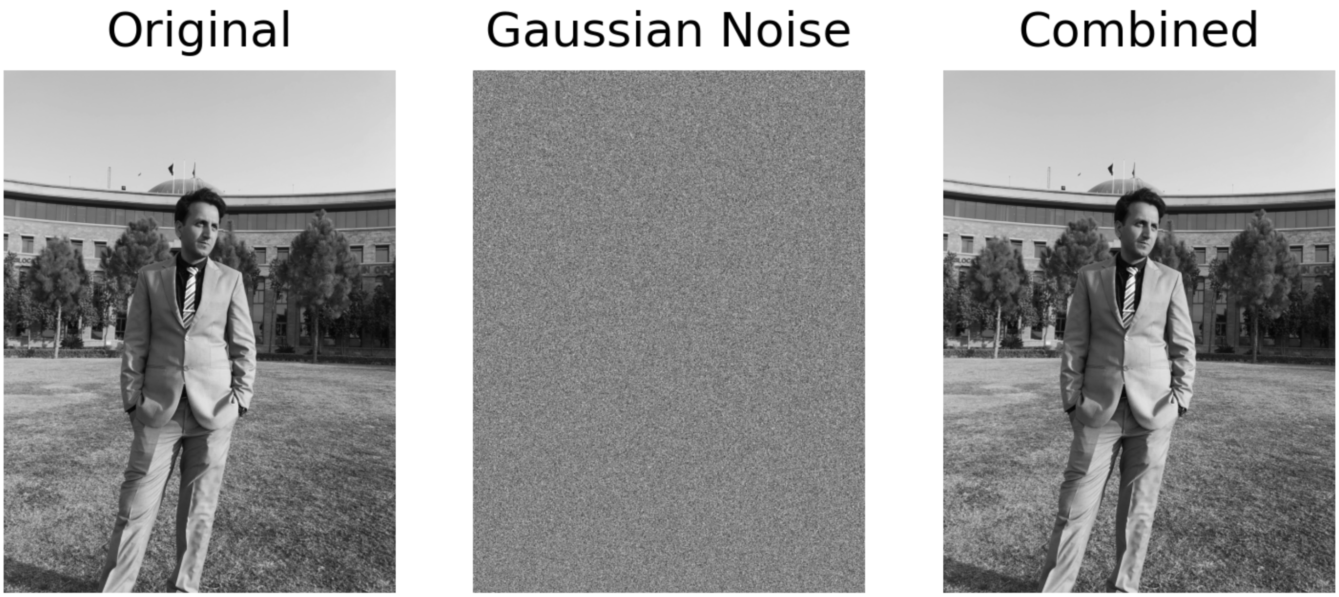
In [37]:

```
fig=plt.figure(dpi=300)
```

```
fig.add_subplot(1,3,1)
plt.imshow(img,cmap='gray')
plt.axis("off")
plt.title("Original")

fig.add_subplot(1,3,2)
plt.imshow(gauss_noise,cmap='gray')
plt.axis("off")
plt.title("Gaussian Noise")

fig.add_subplot(1,3,3)
plt.imshow(img,cmap='gray')
plt.axis("off")
plt.title("Combined")
```

Out[37]:

```
Text(0.5, 1.0, 'Combined')
```



Original

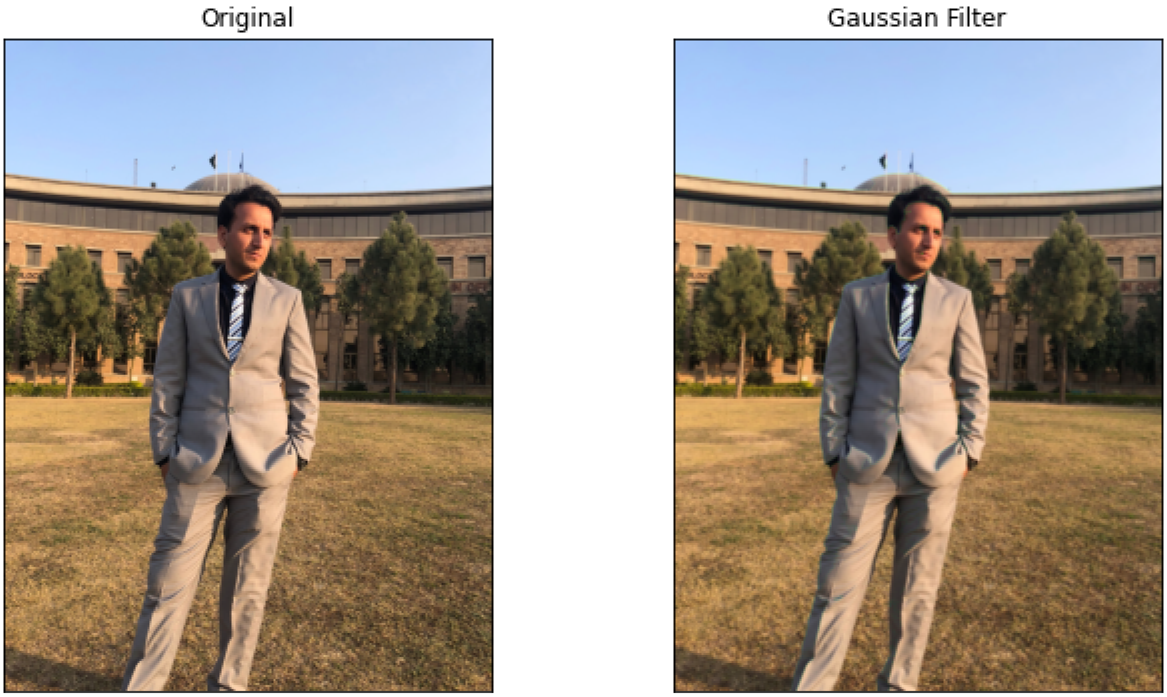
Gaussian Noise

Combined

gaussian filter to original image

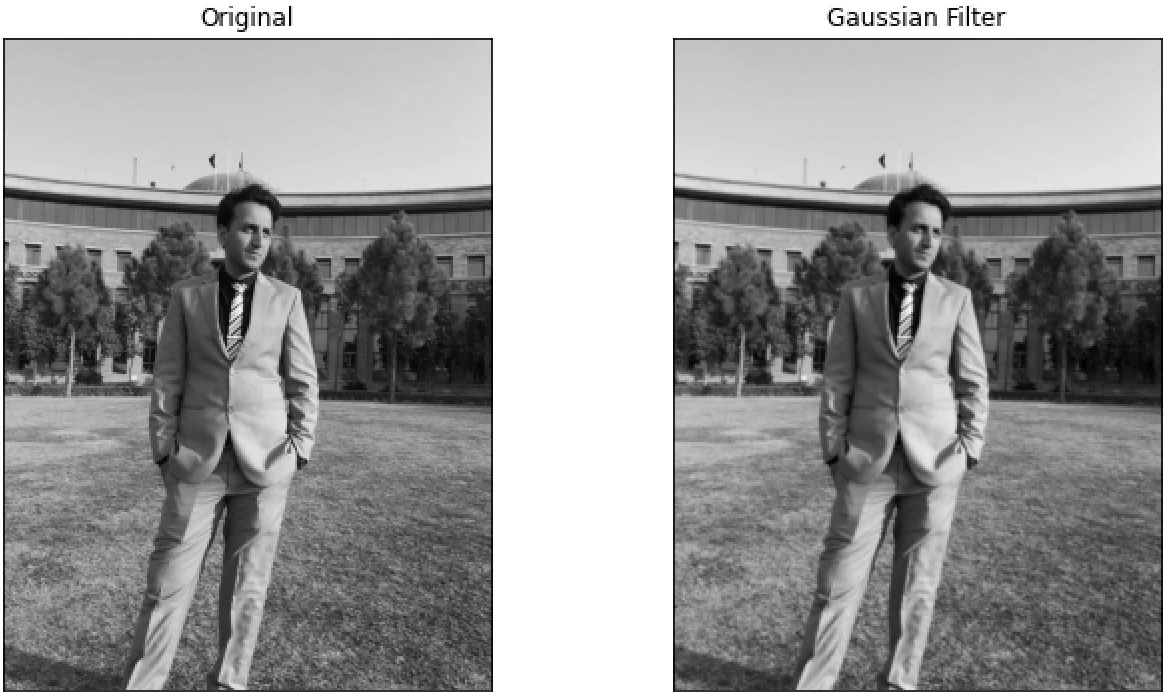
In [8]:

```
new_image = cv2.GaussianBlur(image, (figure_size, figure_size),0)

plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_HSV2RGB)),plt.title('Original
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_HSV2RGB)),plt.title('Gaus
plt.xticks([]), plt.yticks([])
plt.show()
```



In [9]:

```
new_image_gauss = cv2.GaussianBlur(image2, (figure_size, figure_size),0)

plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(new_image_gauss, cmap='gray'),plt.title('Gaussian Filter')
plt.xticks([]), plt.yticks([])
plt.show()
```



Applying Average filter to original image

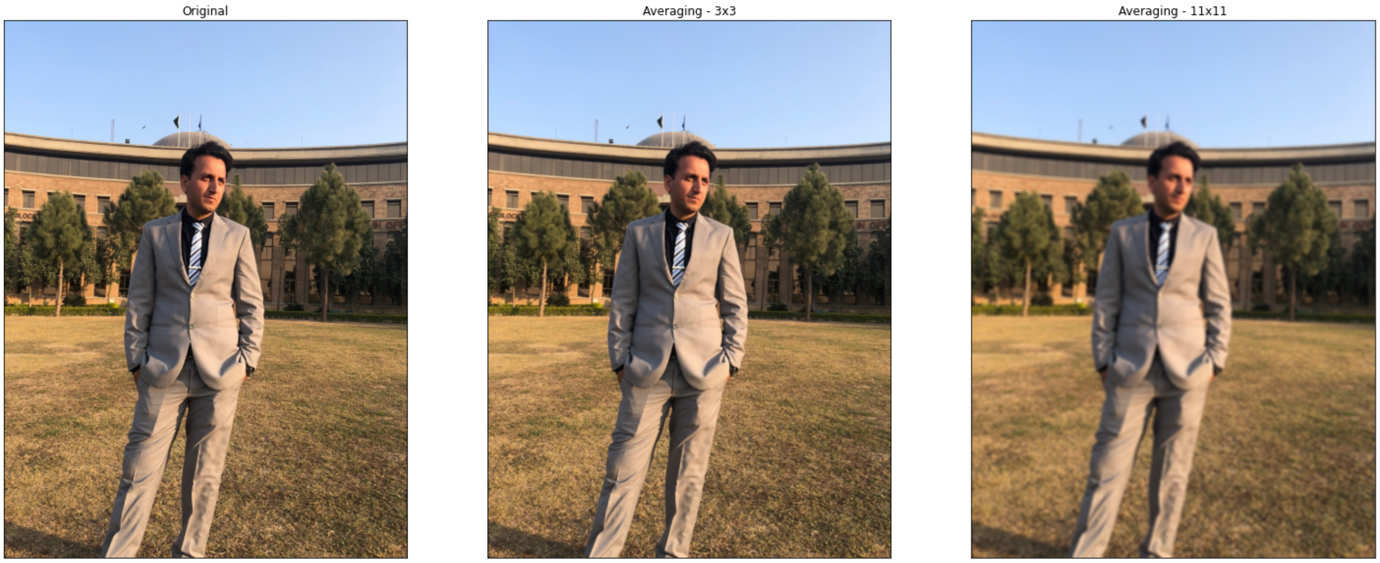
In [8]:

```
img = cv2.imread("haroon.jpeg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) # Fixes color read issue
```

In [9]:

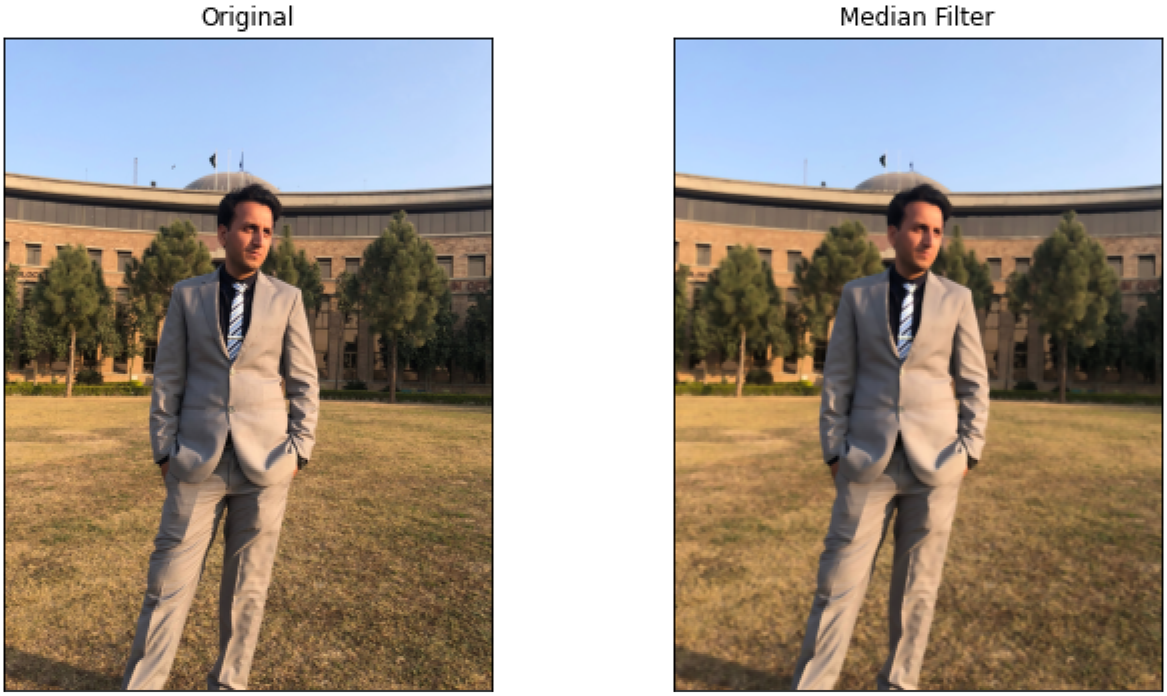
```
av3 = cv2.blur(img,(3,3))
av11 = cv2.blur(img,(11,11))
```

```
# Plot the image. This code is excluded for the rest of the article.
plt.gcf().set_size_inches(25,25)
plt.subplot(131),plt.imshow(img),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(132),plt.imshow(av3),plt.title('Averaging - 3x3')
plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(av11),plt.title('Averaging - 11x11')
plt.xticks([]), plt.yticks([])
plt.show()
```



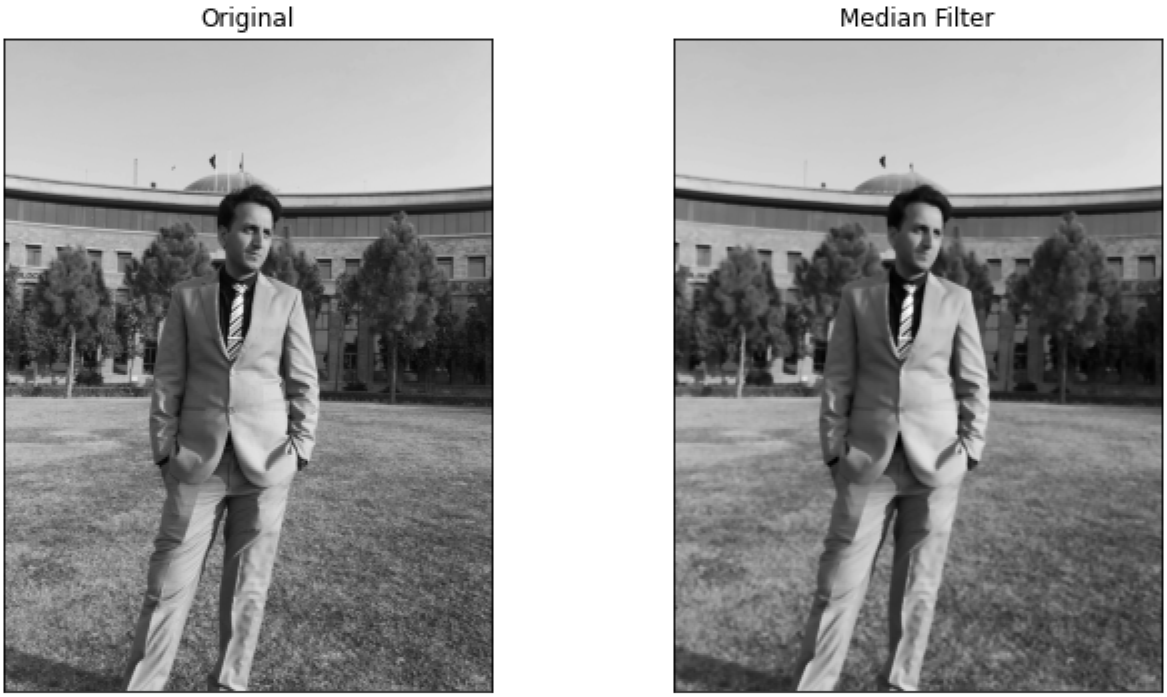
Median Filter to original image

In [10]:
```
new_image = cv2.medianBlur(image, figure_size)

plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(cv2.cvtColor(image, cv2.COLOR_HSV2RGB)),plt.title('Original
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(cv2.cvtColor(new_image, cv2.COLOR_HSV2RGB)),plt.title('Medi
plt.xticks([]), plt.yticks([])
plt.show()
```



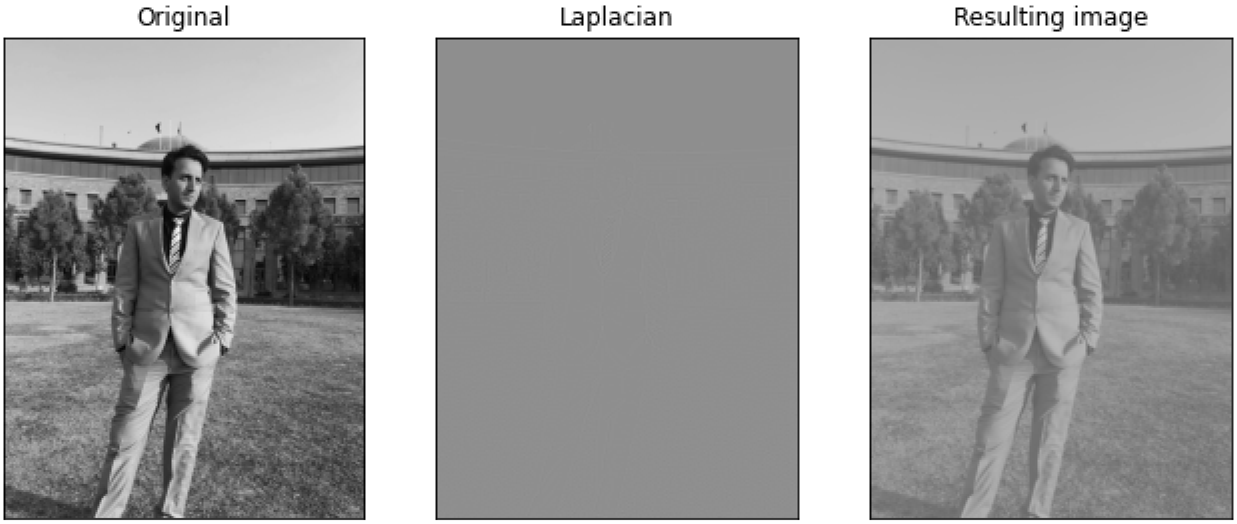
In [11]:

```
new_image = cv2.medianBlur(image2, figure_size)

plt.figure(figsize=(11,6))
plt.subplot(121), plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(new_image, cmap='gray'),plt.title('Median Filter')
plt.xticks([]), plt.yticks([])
plt.show()
```


Original
Median Filter

In [12]:

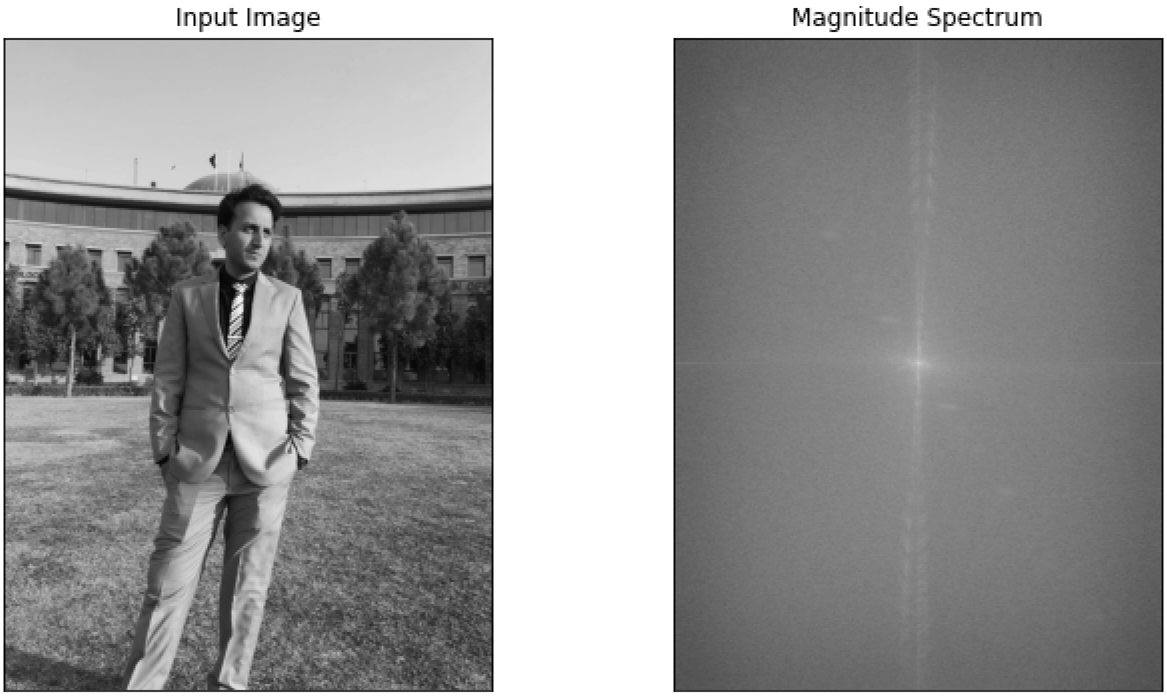
```
new_image = cv2.Laplacian(image2,cv2.CV_64F)

plt.figure(figsize=(11,6))
plt.subplot(131), plt.imshow(image2, cmap='gray'),plt.title('Original')
plt.xticks([]), plt.yticks([])
plt.subplot(132), plt.imshow(new_image, cmap='gray'),plt.title('Laplacian')
plt.xticks([]), plt.yticks([])
plt.subplot(133), plt.imshow(image2 + new_image, cmap='gray'),plt.title('Resulting image
plt.xticks([]), plt.yticks([])
plt.show()
```


Original
Laplacian
Resulting image

In [13]:

```
dft = cv2.dft(np.float32(image2),flags = cv2.DFT_COMPLEX_OUTPUT)

# shift the zero-frequncy component to the center of the spectrum
dft_shift = np.fft.fftshift(dft)
```

```
# save image of the image in the fourier domain.
magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

# plot both images
plt.figure(figsize=(11,6))
plt.subplot(121),plt.imshow(image2, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```

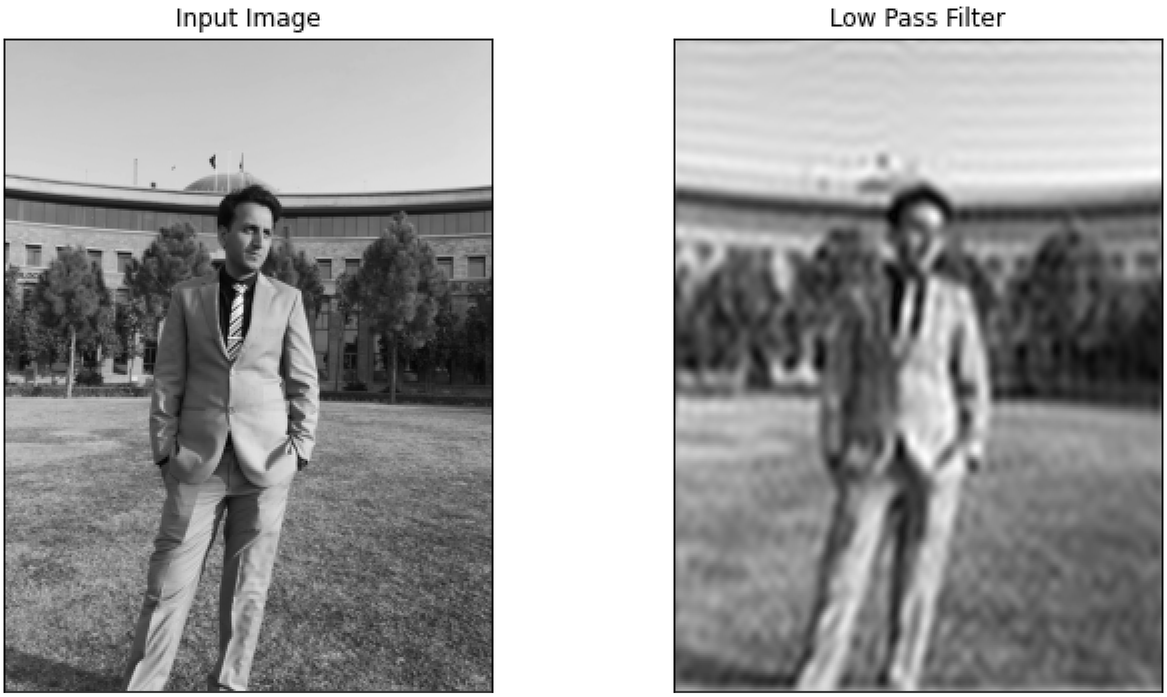


Input Image

Magnitude Spectrum

In [14]:

```
rows, cols = image2.shape
crow,ccol = rows//2 , cols//2

# create a mask first, center square is 1, remaining all zeros
mask = np.zeros((rows,cols,2),np.uint8)
mask[crow-30:crow+30, ccol-30:ccol+30] = 1

# apply mask and inverse DFT
fshift = dft_shift*mask
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1])

# plot both images
plt.figure(figsize=(11,6))
plt.subplot(121),plt.imshow(image2, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Low Pass Filter'), plt.xticks([]), plt.yticks([])
plt.show()
```

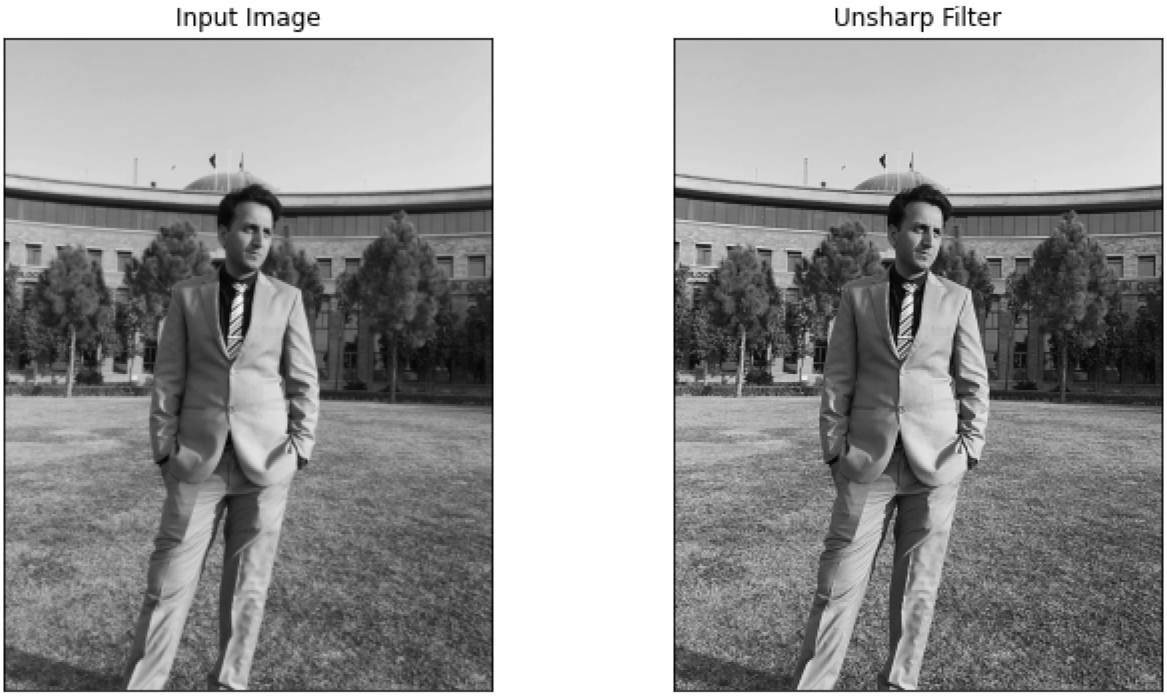


Input Image

Low Pass Filter

In [15]:

```
image2 = Image.fromarray(image2.astype('uint8'))
new_image = image2.filter(ImageFilter.UnsharpMask(radius=2, percent=150))

plt.figure(figsize=(11,6))
plt.subplot(121),plt.imshow(image2, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(new_image, cmap = 'gray')
plt.title('Unsharp Filter'), plt.xticks([]), plt.yticks([])
plt.show()
```



Input Image

Unsharp Filter

In [ ]: