Name: Jay Goyal

Roll no.: C017

Semester: VI

Program: B.Tech

Branch: EXTC

Date of performance: 19th February

Date of Submission: 26th February

Experiment Number: 5

Aim:

a. To write a program in PYTHON to implement FFT and IFFT on an image

b. To write a program in PYTHON implement LPF and HPF in frequency domain on an image

Conclusion:

The concepts learnt in theis practical where

Concept of filtering in frequency domain in image processing

Discrete Fourier Transform

The inverse DFT

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

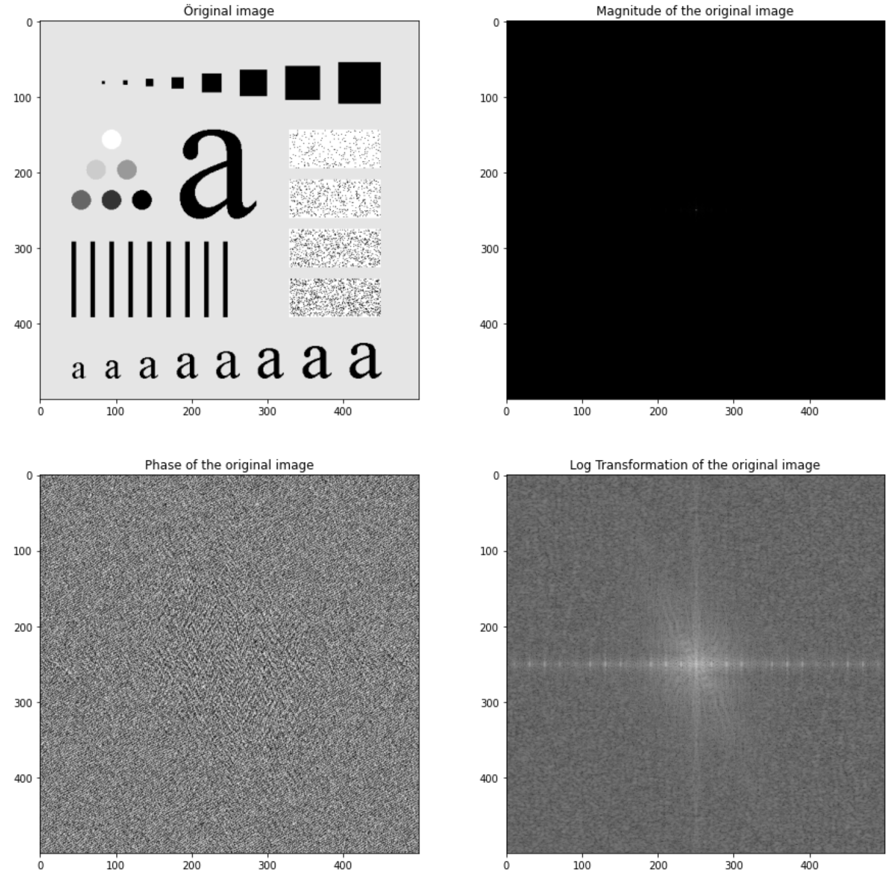
```
img= cv2.imread("/content/Fig0333(a)(test_pattern_blurring_orig).tif",0) # Read the image
```

```
img_fft= np.fft.fft2(img, s=None, axes=(-2, -1), norm=None) # Apply FFT on origianl image
```

```
img_fft= np.fft.fftshift(img_fft) #Shifting the origin. Equvalent to multiplying by (-1)^x+y
img_abs= np.abs(img_fft) #Obtaining magintude of the transformed image
img_phase= np.angle(img_fft) #Obtaining the Phase of the trasnformed image
img_log= np.log10(1+img_abs) # Performing Log transformation
```

```
fig = plt.figure(figsize=(15,15),facecolor='w')
plt.subplot(2,2,1)
plt.imshow(img, cmap="gray")
plt.title("Öriginal image")
plt.subplot(2,2,2)
plt.imshow(img_abs, "gray")
plt.title("Magnitude of the original image")
plt.subplot(2,2,3)
plt.imshow(img_phase, "gray")
plt.title("Phase of the original image")
plt.subplot(2,2,4)
plt.imshow(img_log, "gray")
plt.title("Log Transformation of the original image")
```

```
    Text(0.5, 1.0, 'Log Transformation of the original image')
```



Öriginal image

Magnitude of the original image

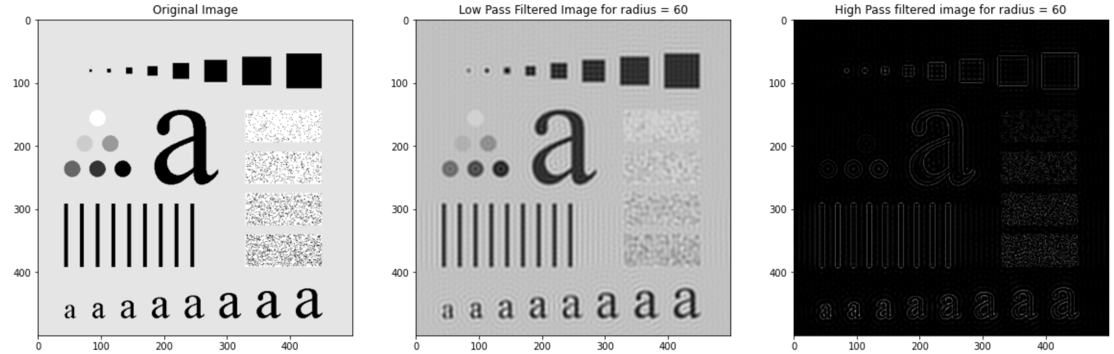
Phase of the original image

Log Transformation of the original image

```
m,n= img_fft.shape #Obtaining size (rows and cols)of the image
```

```
#Creating ideal lpf and ideal hpf masks
lpf= img.copy()
hpf= img.copy()
#Take radii d0= 60

d0= 60

for i in range(m):
  for j in range(n):
    d1= np.sqrt((i-m//2)**2 +(j-n//2)**2)
    if d1>d0:
      lpf[i,j]=0
      hpf[i,j]=1
    else:
      lpf[i,j]= 1
      hpf[i,j]=0


img_lpf= img_fft*lpf # multiply lpf mask with the fft image
img_lpf= np.fft.fftshift(img_lpf)
img_new_lpf= np.real(np.fft.ifft2(img_lpf))

img_hpf= img_fft*hpf # multiply hpf mask with the fft image
img_hpf= np.fft.fftshift(img_hpf)
img_new_hpf= np.real(np.fft.ifft2(img_hpf))

#Plot orinal, low pass filtered and high pass filtered image
plt.figure(figsize=(20,20))
plt.subplot(1,3,1)
plt.title("Original Image")
plt.imshow(img,cmap="gray")
plt.subplot(1,3,2)
plt.title("Low Pass Filtered Image for radius = " +str(d0))
plt.imshow(img_new_lpf,cmap="gray")
plt.subplot(1,3,3)
plt.title("High Pass filtered image for radius = " +str(d0))
plt.imshow(img_new_hpf,cmap="gray", vmin=0, vmax=255)
```

```
<matplotlib.image.AxesImage at 0x7f5228d4ff98>
```



Inference:

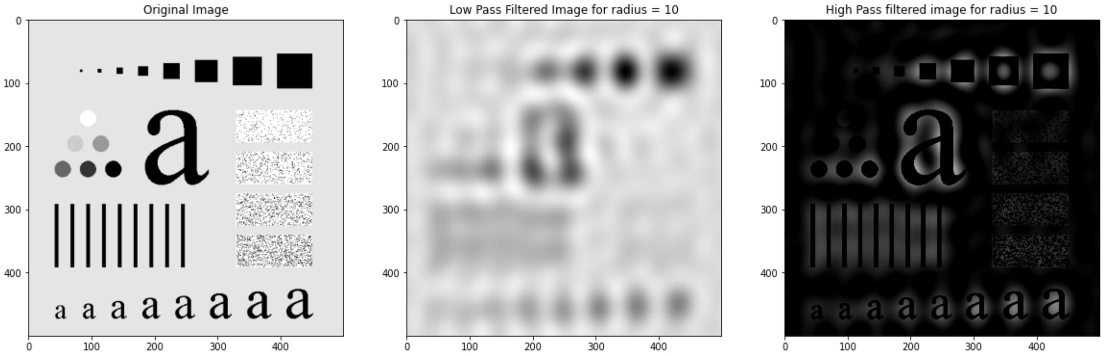
Low Pass Filtered Image for radius of 60 we get image which is greyish

High Pass Filtered Image for radius of 60 we get image which is black but with

a few letters visible.

```
#Creating ideal lpf and ideal hpf masks
lpf= img.copy()
hpf= img.copy()


d0= 10

for i in range(m):
  for j in range(n):
    d1= np.sqrt((i-m//2)**2 +(j-n//2)**2)
    if d1>d0:
      lpf[i,j]=0
      hpf[i,j]=1
    else:
      lpf[i,j]= 1
      hpf[i,j]=0


img_lpf= img_fft*lpf # multiply lpf mask with the fft image
img_lpf= np.fft.fftshift(img_lpf)
img_new_lpf= np.real(np.fft.ifft2(img_lpf))

img_hpf= img_fft*hpf # multiply hpf mask with the fft image
img_hpf= np.fft.fftshift(img_hpf)
img_new_hpf= np.real(np.fft.ifft2(img_hpf))

#Plot orinal, low pass filtered and high pass filtered image
plt.figure(figsize=(20,20))
plt.subplot(2,3,1)
plt.title("Original Image")
plt.imshow(img,cmap="gray")
plt.subplot(2,3,2)
plt.title("Low Pass Filtered Image for radius = " +str(d0))
plt.imshow(img_new_lpf,cmap="gray")
plt.subplot(2,3,3)
plt.title("High Pass filtered image for radius = " +str(d0))
plt.imshow(img_new_hpf,cmap="gray", vmin=0, vmax=255)
```

```
<matplotlib.image.AxesImage at 0x7f5228c50b00>
```


Original Image
Low Pass Filtered Image for radius = 10
High Pass filtered image for radius = 10
0
100
200
300
400

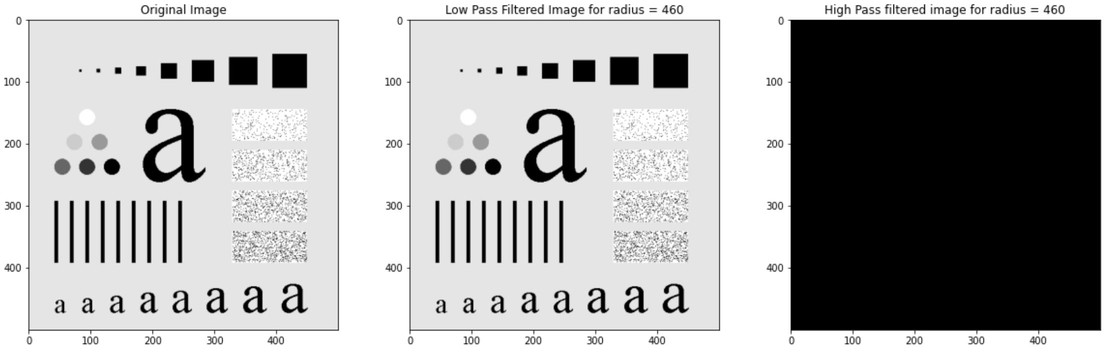
Inference:

Low Pass Filtered Image for radius of 10 we get image which is blur

High Pass Filtered Image for radius of 10 we get image which is black but with a few letters visible(but a bit clear then the radius whch is 60).

```
#Creating ideal lpf and ideal hpf masks
lpf= img.copy()
hpf= img.copy()


d0= 460

for i in range(m):
  for j in range(n):
    d1= np.sqrt((i-m//2)**2 +(j-n//2)**2)
    if d1>d0:
      lpf[i,j]=0
      hpf[i,j]=1
    else:
      lpf[i,j]= 1
      hpf[i,j]=0



img_lpf= img_fft*lpf # multiply lpf mask with the fft image
img_lpf= np.fft.fftshift(img_lpf)
img_new_lpf= np.real(np.fft.ifft2(img_lpf))

img_hpf= img_fft*hpf # multiply hpf mask with the fft image
img_hpf= np.fft.fftshift(img_hpf)
img_new_hpf= np.real(np.fft.ifft2(img_hpf))

#Plot orinal, low pass filtered and high pass filtered image
plt.figure(figsize=(20,20))
plt.subplot(3,3,1)
plt.title("Original Image")
plt.imshow(img,cmap="gray")
plt.subplot(3,3,2)
plt.title("Low Pass Filtered Image for radius = " +str(d0))
plt.imshow(img_new_lpf,cmap="gray")
plt.subplot(3,3,3)
plt.title("High Pass filtered image for radius = " +str(d0))
plt.imshow(img_new_hpf,cmap="gray", vmin=0, vmax=255)
```

```
<matplotlib.image.AxesImage at 0x7f5228b52630>
```


Original Image
Low Pass Filtered Image for radius = 460
High Pass filtered image for radius = 460
0
100
200
300
400

Inference:

Low Pass Filtered Image for radius of 460 we get image which is similar to the original

High Pass Filtered Image for radius of 460 we get image which is completely black.