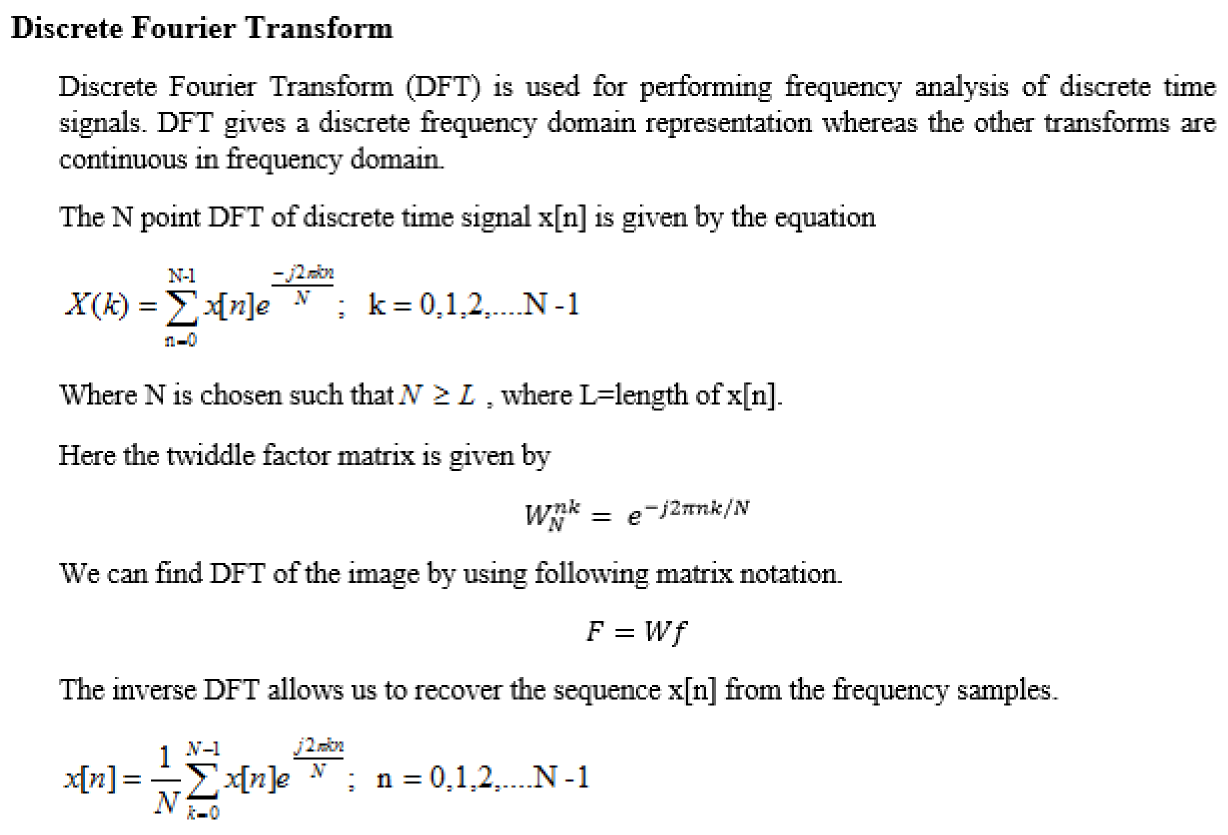
Experiment No : 8 Discrete Fourier Transform

- Name: Sachi Shah
- Roll No.: C094
- Batch: EC1
- Sap Id: 70321018081

Aim:

1. Compute the DFT of the image and plot its Magnitude and Phase Spectrum
2. Implement low pass and high pass filtering in frequency domain using DFT and comment on the subjective quality of the reconstructed image.

```
import cv2
from sympy import fft, ifft
import numpy as np
import matplotlib.pyplot as plt
```



Discrete Fourier Transform

Discrete Fourier Transform (DFT) is used for performing frequency analysis of discrete time signals. DFT gives a discrete frequency domain representation whereas the other transforms are continuous in frequency domain.

The N point DFT of discrete time signal x[n] is given by the equation

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}} ; \quad k = 0,1,2,....N-1$$

Where N is chosen such that $N \geq L$, where L=length of x[n].

Here the twiddle factor matrix is given by

$$W_N^{nk} = e^{-j2\pi nk/N}$$

We can find DFT of the image by using following matrix notation.

$$F = Wf$$

The inverse DFT allows us to recover the sequence x[n] from the frequency samples.

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} x[n] e^{\frac{j2\pi kn}{N}} ; \quad n = 0,1,2,....N-1$$

Example :

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-j(2\pi/N)} & e^{-j(4\pi/N)} & e^{-j(6\pi/N)} \\ 1 & e^{-j(4\pi/N)} & e^{-j(8\pi/N)} & e^{-j(12\pi/N)} \\ 1 & e^{-j(6\pi/N)} & e^{-j(12\pi/N)} & e^{-j(18\pi/N)} \end{bmatrix}}_{\text{DFT matrix: } F} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & e^{-j(2\pi/4)} & e^{-j(4\pi/4)} & e^{-j(6\pi/4)} \\ 1 & e^{-j(4\pi/4)} & e^{-j(8\pi/4)} & e^{-j(12\pi/4)} \\ 1 & e^{-j(6\pi/4)} & e^{-j(12\pi/4)} & e^{-j(18\pi/4)} \end{bmatrix}}_{\text{DFT matrix: } F} \begin{bmatrix} 2 \\ 3 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 3-j2 \\ -3 \\ 3+j2 \end{bmatrix}.$$

```python
"""Function to calculate the discrete Fourier Transform
of a 1D real-valued signal x
and Recover it"""

x = [2,3,-1,1]
print("Original Data = ",x)

X = np.fft.fft(x)
print("DFT of x(n) = ",X)

xr = np.fft.ifft(X)
print("Recovered x(n) = ",xr)

```

```
Original Data =  [2, 3, -1, 1]
DFT of x(n) =  [ 5.+0.j  3.-2.j -3.+0.j  3.+2.j]
Recovered x(n) =  [ 2.+0.j  3.+0.j -1.+0.j  1.+0.j]
```

compute FFT using built in function

$$F(u\ ,\ v)=A\ f(x\ ,y)\ A^T$$

$$A=\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -j & -1 & j\\ 1 & -1 & 1 & -1\\ 1 & j & -1 & -j\end{bmatrix} \qquad f(x,y)=\begin{bmatrix}2 & 4 & 1 & 6\\ 3 & 2 & 1 & 4\\ 6 & 5 & 3 & 2\\ 1 & 2 & 4 & 3\end{bmatrix}$$

$$F(u,v)=\begin{bmatrix}49 & 3+2j & -7 & 3-2j\\ -3 & -1 & -9+2j & -3-10j\\ 9 & 5-4j & -3 & 5+4j\\ -3 & -3+10j & -9-2j & -1\end{bmatrix}$$

```
x = [[2,4,1,6],[3,2,1,4],[6,5,3,2],[1,2,4,3]]
X = np.fft.fft2(x)        #fft2 means for 2D array
print("DFT of x(n) = \n",X)

print("\n")

xr = np.fft.ifft2(X)       #ifft2 means for 2D array
print("Recovered x(n) = \n",xr)
```

```
DFT of x(n) =
 [[49. +0.j  3. +2.j -7. +0.j  3. -2.j]
 [-3. +0.j -1. +0.j -9. +2.j -3.-10.j]
 [ 9. +0.j  5. -4.j -3. +0.j  5. +4.j]
 [-3. +0.j -3.+10.j -9. -2.j -1. +0.j]]


Recovered x(n) =
 [[2.+0.j 4.+0.j 1.+0.j 6.+0.j]
 [3.+0.j 2.+0.j 1.+0.j 4.+0.j]
 [6.+0.j 5.+0.j 3.+0.j 2.+0.j]
 [1.+0.j 2.+0.j 4.+0.j 3.+0.j]]
```

To center the DFT

$$f(x,y) \rightarrow f'(x,y) = (-1)^{x+y}\ f(x,y) \rightarrow F(u,v)$$

Log transform

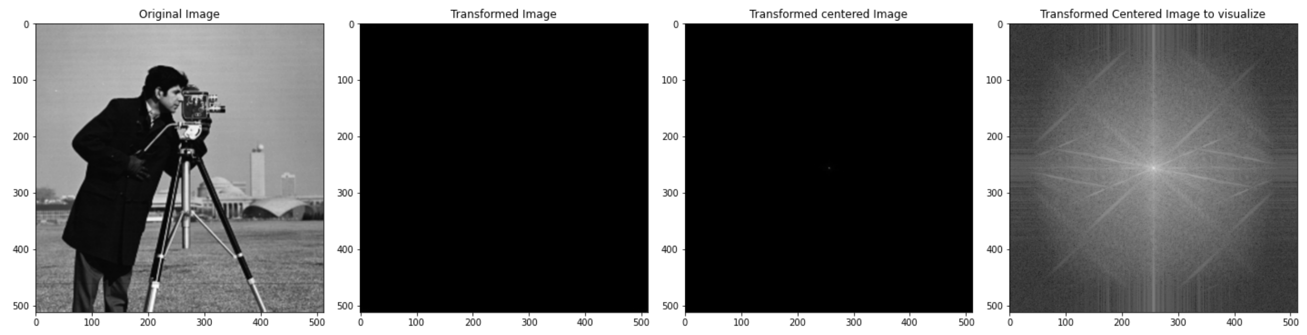
$$|F(u,v)| \rightarrow \log(1+ |F(u,v)|)$$

```
#Shift the origin. Equvalent to multiplying by (-1)^x+y
x = [[2,4,1,6],[3,2,1,4],[6,5,3,2],[1,2,4,3]]
X = np.fft.fft2(x)
Xcentered = np.fft.fftshift(X)
print('Centered FFT = \n',Xcentered)

#compute the log

```

```
Centered FFT =
 [[-3. +0.j  5. +4.j  9. +0.j  5. -4.j]
 [-9. -2.j -1. +0.j -3. +0.j -3.+10.j]
 [-7. +0.j  3. -2.j 49. +0.j  3. +2.j]
 [-9. +2.j -3.-10.j -3. +0.j -1. +0.j]]
```

To compute the DFT of a image

```
# Read the image
img=cv2.imread('cameraman.tif', 0)

# Compute the FFT
imgfft=np.fft.fft2(img)

plt.figure(figsize = (20,20))
plt.subplot(141)
plt.imshow(img, cmap='gray')
plt.title('Original Image')
plt.subplot(142)
plt.imshow(np.abs(imgfft), cmap='gray')
plt.title('Transformed Image')

#centering
fftcen = np.fft.fftshift(imgfft)
plt.subplot(143)
plt.imshow(np.abs(fftcen),cmap = 'gray')
plt.title('Transformed centered Image')

#visualise(magnitude spectrum)
fftlog = np.log10(1+fftcen)
plt.subplot(144)
plt.imshow(np.abs(fftlog),cmap = 'gray')
plt.title('Transformed Centered Image to visualize')

plt.tight_layout()
```

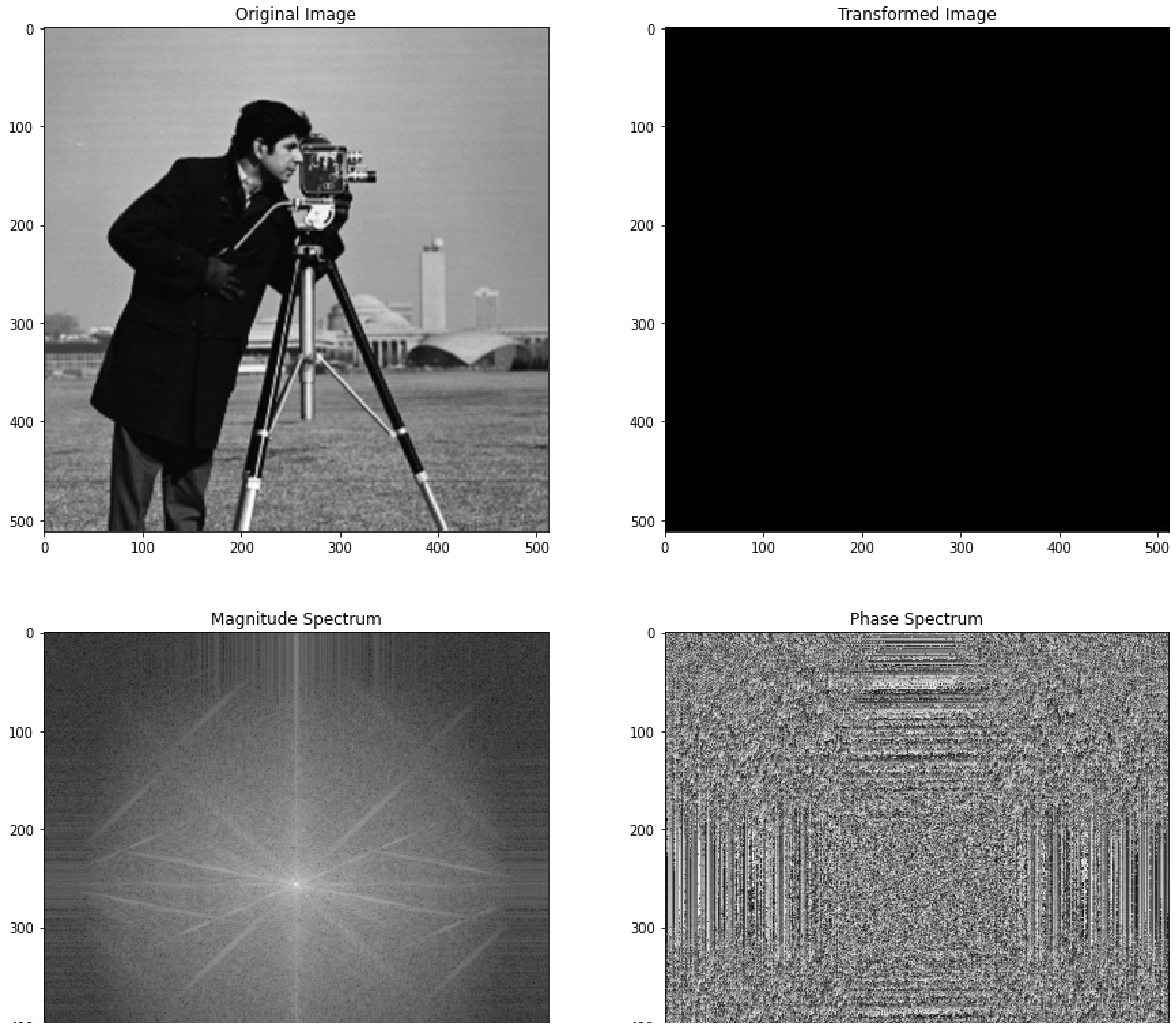


```
# Compute the Phase spectrum of the transformed image
fftphase = np.angle(imgfft)
```

Plot the image, magnitude spectrun, phase spectrum, log transformed magnitude spectrum

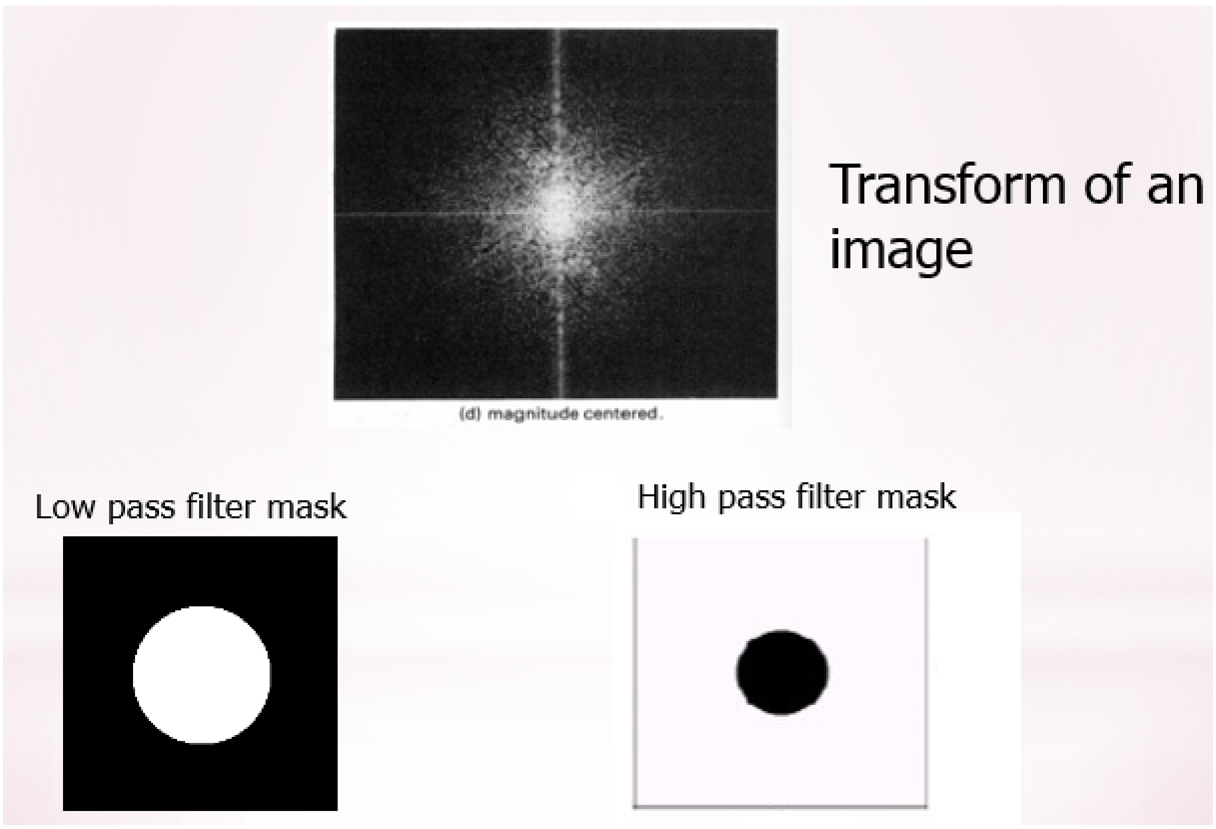
```
plt.figure(figsize = (15,15))
plt.subplot(221)
plt.imshow(img, cmap='gray')
plt.title('Original Image')

plt.subplot(222)
plt.imshow(np.abs(imgfft), cmap='gray')
plt.title('Transformed Image')

plt.subplot(223)
plt.imshow(np.abs(fftlog),cmap = 'gray')
plt.title('Magnitude Spectrum')

plt.subplot(224)
plt.imshow(fftphase,cmap = 'gray')
plt.title('Phase Spectrum')
```

```
Text(0.5, 1.0, 'Phase Spectrum')
```



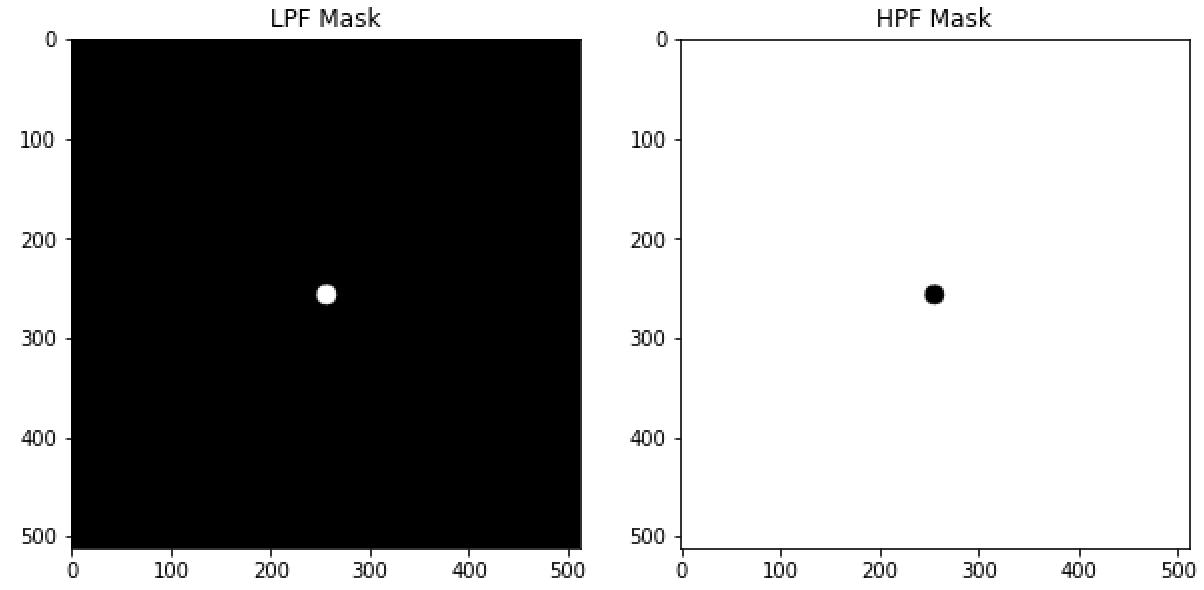
Filtering the low frequency and the high frequency components of an image

Develop a ideal LPF and HPF mask

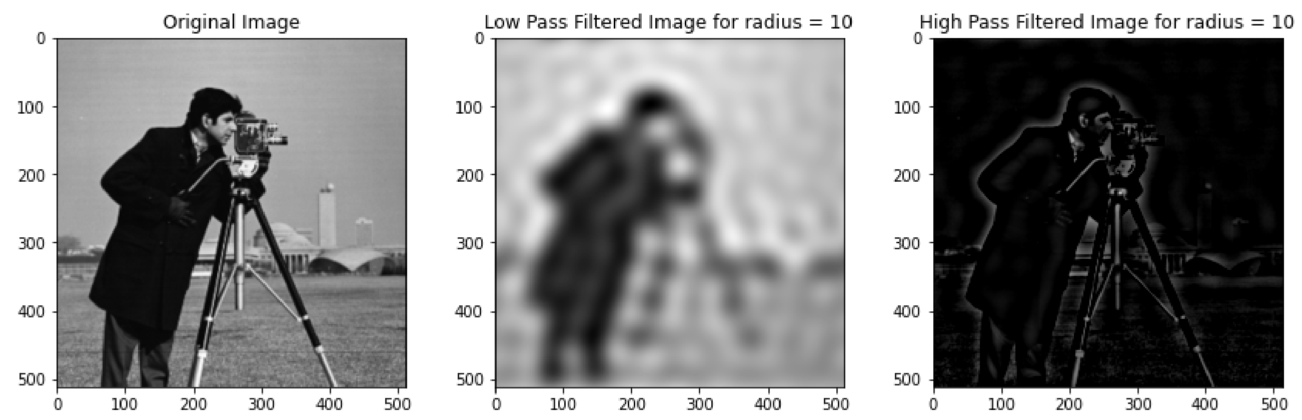


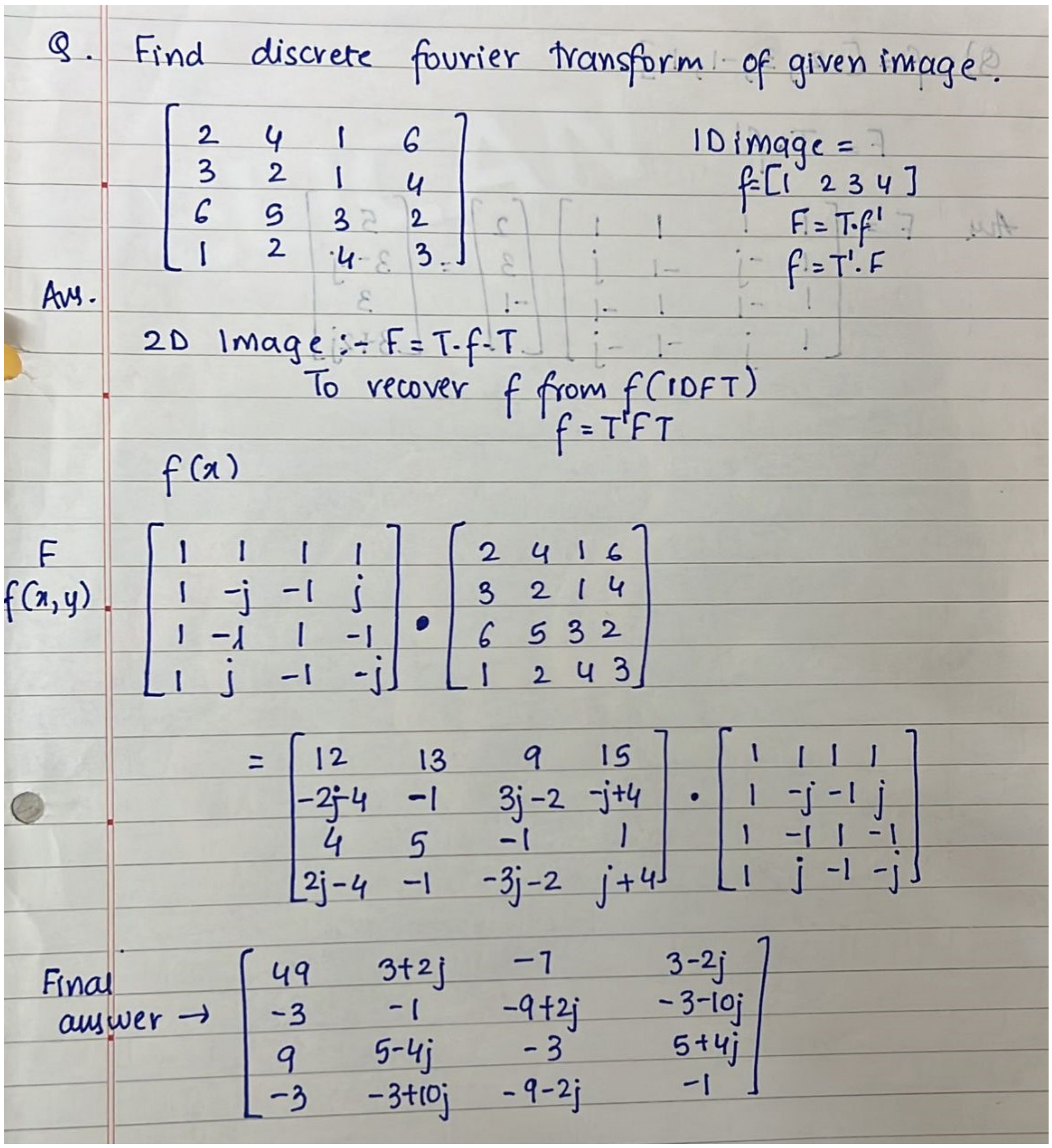
```python
# Compute the rows and columns of the image
r,c = img.shape

# copy the image as a lpf and hpf variable
lpf = img.copy()
hpf = img.copy()

#Take radii d0= 10, 30,60,160,460
d = 10 #radius of the circle

#Creat an ideal lpf and ideal hpf masks
for i in range(r):
    for j in range(c):
        d1 = np.sqrt((i-r//2)**2 + (j-c//2)**2)   #Calculated radius
        if d1>d:
            lpf[i,j] = 0
            hpf[i,j] = 1
        else:
            lpf[i,j] = 1
            hpf[i,j] = 0

#Plot orinal, low pass filtered and high pass filtered image

plt.figure(figsize=(10,10))
plt.subplot(121)
plt.imshow(lpf,cmap = 'gray')
plt.title("LPF Mask")

plt.subplot(122)
plt.imshow(hpf,cmap = 'gray')
plt.title("HPF Mask")

```

```
Text(0.5, 1.0, 'HPF Mask')
```



```
# multiply lpf mask and hpf mask with the fft image
LPFimg = fftcen*lpf
HPFimg = fftcen*hpf

# shift the fft
LPF_re_center = np.fft.fftshift(LPFimg)
HPF_re_center = np.fft.fftshift(HPFimg)

# Compute the inverse fft of the resultant filtered image
final_LPF = np.real(np.fft.ifft2(LPF_re_center))
final_HPF = np.real(np.fft.ifft2(HPF_re_center))

#Plot orinal, low pass filtered and high pass filtered image
plt.figure(figsize=(12,12))

plt.subplot(131)
plt.imshow(img,cmap = 'gray')
plt.title("Original Image")

plt.subplot(132)
plt.imshow(final_LPF,cmap = 'gray')
plt.title("Low Pass Filtered Image for radius = 10")


plt.subplot(133)
plt.imshow(final_HPF,cmap = 'gray',vmin=0,vmax=255)
plt.title("High Pass Filtered Image for radius = 10")

plt.tight_layout()
```





Q. Find discrete fourier transform of given image.

$$\begin{bmatrix} 2 & 4 & 1 & 6 \\ 3 & 2 & 1 & 4 \\ 6 & 5 & 3 & 2 \\ 1 & 2 & 4 & 3 \end{bmatrix}$$

1D image =
$f = [1 \ 2 \ 3 \ 4]$
$F = T \cdot f'$
$f = T' \cdot F$

Ans.

2D Image :- $F = T \cdot f \cdot T$
To recover f from f (IDFT)
$f = T' F T$

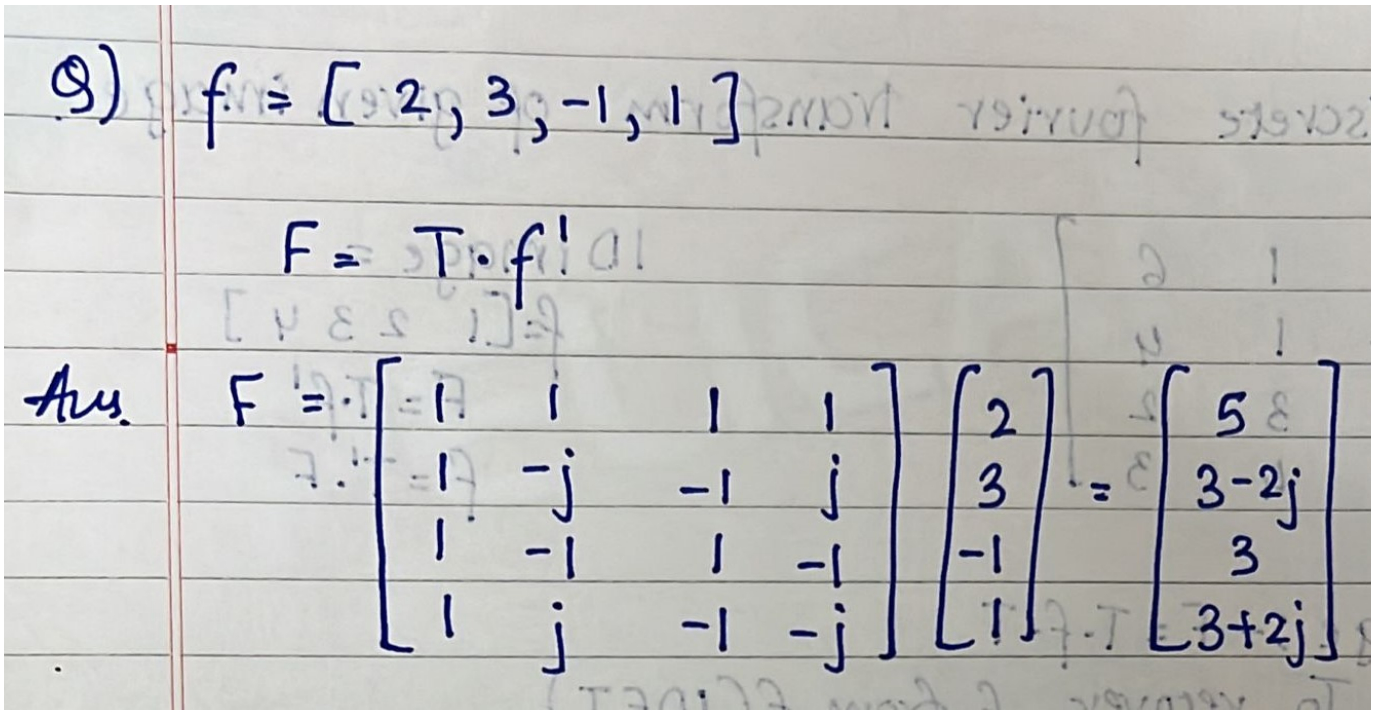
$f(x)$

F
$f(x,y)$

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \cdot \begin{bmatrix} 2 & 4 & 1 & 6 \\ 3 & 2 & 1 & 4 \\ 6 & 5 & 3 & 2 \\ 1 & 2 & 4 & 3 \end{bmatrix}$$

$$= \begin{bmatrix} 12 & 13 & 9 & 15 \\ -2j-4 & -1 & 3j-2 & -j+4 \\ 4 & 5 & -1 & 1 \\ 2j-4 & -1 & -3j-2 & j+4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$

Final answer $\rightarrow$ $$\begin{bmatrix} 49 & 3+2j & -7 & 3-2j \\ -3 & -1 & -9+2j & -3-10j \\ 9 & 5-4j & -3 & 5+4j \\ -3 & -3+10j & -9-2j & -1 \end{bmatrix}$$



Q) $f = [2, 3, -1, 1]$

$F = T \cdot f$

Ans. $$F = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 3-2j \\ 3 \\ 3+2j \end{bmatrix}$$

Conclusion :

1. We computed the discrete fourrier transsform for 1D and 2D array.
2. We implemented the code for the same and verified the results
3. We implemented the code for DFT on the image, it was obsereved that the transformed image was black in color and not visible. Therefore, we applied log transformation. Centering was done to bring low frequency pixels to center and bring high frequency pixels away from the center.
4. For filtering in frequency domain is done by multiplying transformed image and mask.
5. We implemented the lpf and hpf mask multiplied with fft. However, before computing inverse fft we have to recenter the centered transformed image. We then obsereved the output of low pass and high pass filtered images. LPF provides blurring and HPF provides sharpening of the image.