DAIICT

Assignment 5

Advanced Image Processing

Kishan Vaishnani - 202011004
Dhyanil Mehta - 202011032
4-7-2021

Table of Contents

Task 1

Problem Statement

Find out and show the Magnitude and Phase information of the cameraman image (Fig.1). Perform log transformation on the magnitude spectrum and show the result. Discuss the reason of difference between the observations (with log and without log transformation). Observe the significance of DFT coefficients of the image by reconstructing the image with higher frequency coefficients and lower frequency coefficients, separately without changing the phase information. [Tips: Use FFT for finding out the DFT coefficients.]

Figure 1

Library Used: SciPy, NumPy

Method

1. Getting 2D Discrete Fourier Transform of image

   `scipy.fft.fft2(image)`

2. Shift fourier transform

   `scipy.fft.fftshift(DFT)`

3. Get magnitude from DFT

   `numpy.abs(DFT of image)`

4. Get phase of frequency domain image

   `numpy.angle(DFT of image)`

5. Inverse 2D Discrete Fourier Transform

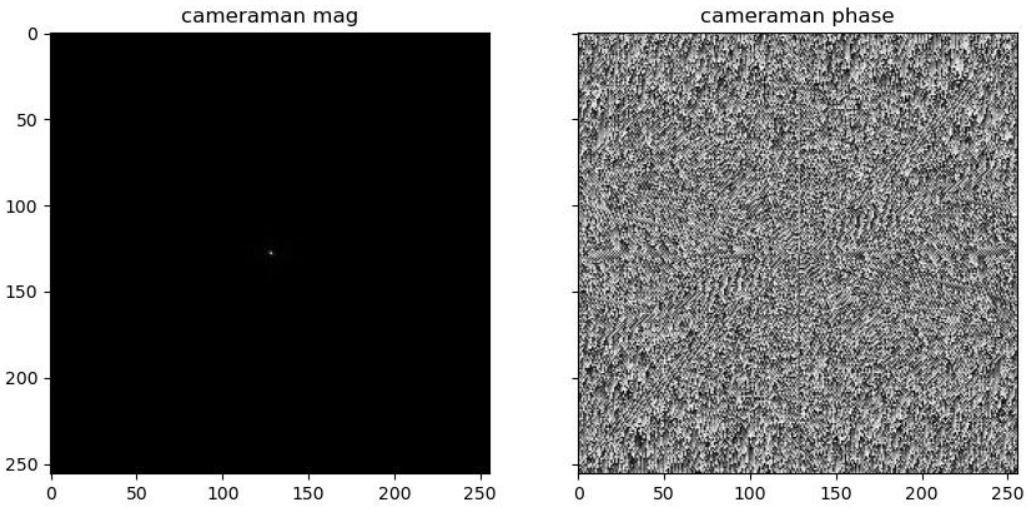
   `scipy.fft.ifft2(DFT)`

6. Log transformation

   $\boldsymbol{s = c * log(1 + r)}$ **where r = magnitude for this task**

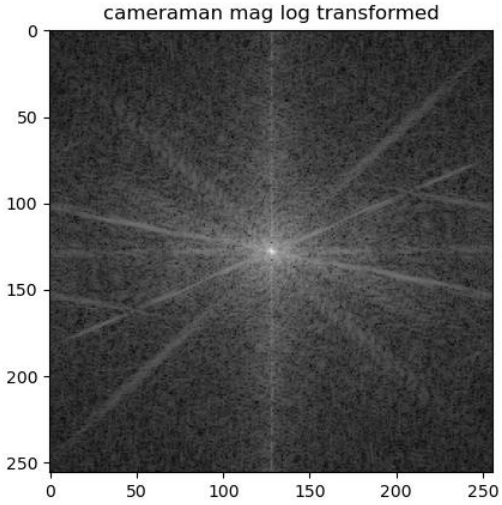
Observation

1. Cameraman's magnitude and phase

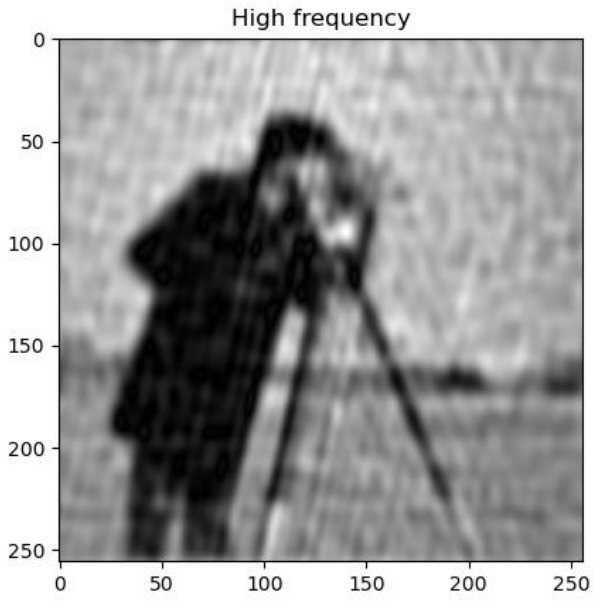


2. Log transformed cameraman's magnitude

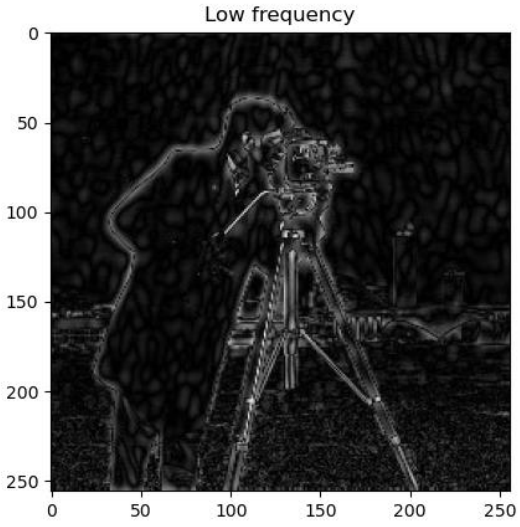
   The result shows that the image contains components of all frequencies, but that their magnitude gets smaller for higher frequencies. Hence, low frequencies contain more image information than the higher ones.



3. Reconstructed image with higher frequency coefficient



4. Reconstructed image with lower frequency coefficient



Conclusion

- By removing the lower frequency coefficients, we get a distorted version of the original image from reconstruction.
- By removing the higher frequency coefficients, the edges are sharpened and the image spectrum gets dark.

Task 2

Problem Statement

Demonstrate the significance of magnitude and phase information by reconstructing an image with the magnitude of Fig.2(a) and phase of Fig. 2(b). Repeat the experiment with the magnitude of Fig.2 (b) and phase of Fig.2 (a). [Tips: First of all, resize the images to create same dimensional images]

Figure 2

Library Used: SciPy, cv2(For resizing image)

Method

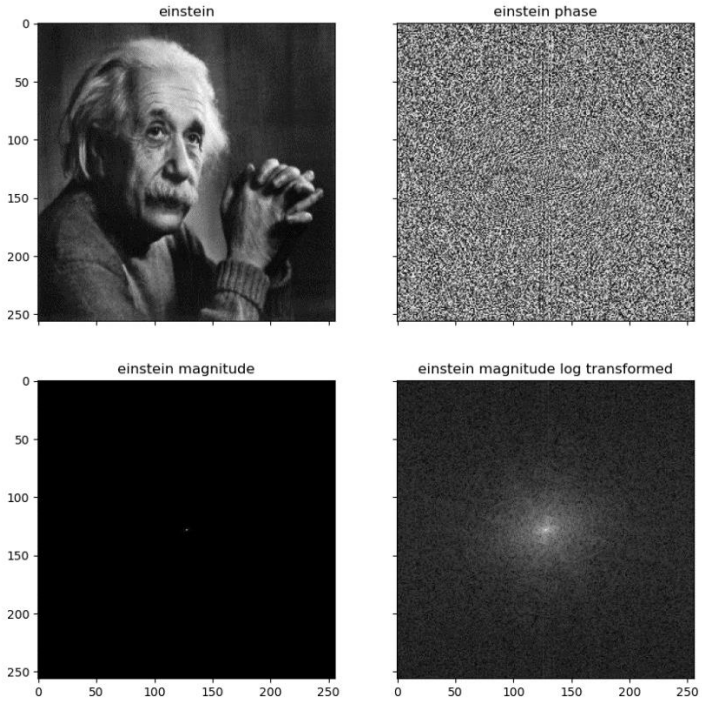
1. Fourier transform expression in terms of magnitude and phase

   $\boldsymbol{F(u,v) = |F(u,v)| * e^{i\emptyset(u,v)}}$

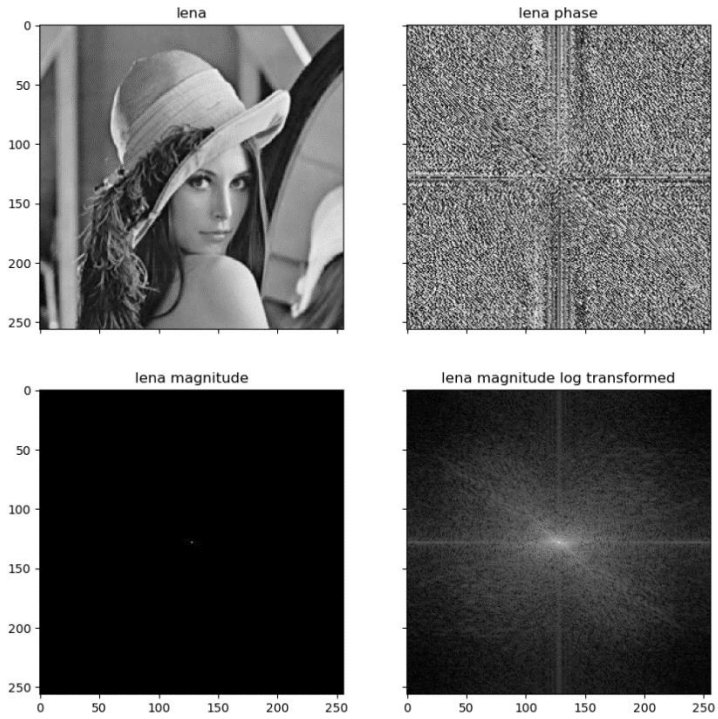
   Fourier transform expression can be obtained if we have magnitude $\boldsymbol{|F(u,v)|}$ and phase $\boldsymbol{\emptyset(u,v)}$**.** It can then be used to reconstruct the image using inverse discrete Fourier~~fourier~~ transform.

Observation

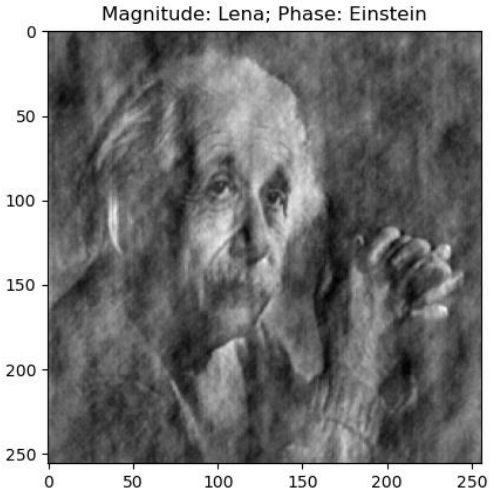
1. Einstein's phase and magnitude



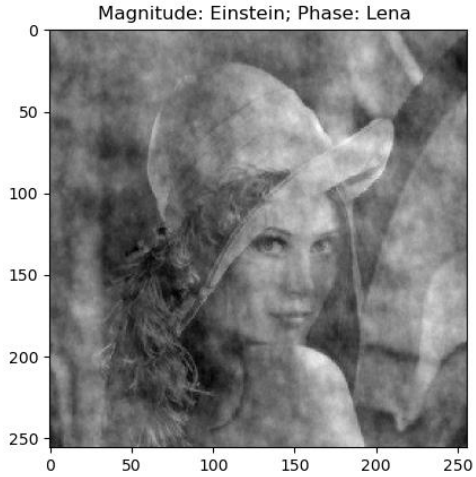
2. Lena's phase and magnitude



3. Einstein’s phase and Lena’s magnitude



4. Lena’s phase and Einstein’s magnitude



Conclusion

- By reconstructing both the images using interchanged phases and magnitudes, the resultant images appear to have a filter superimposed but with filter rotated 90 degrees in another image.

Task 3

Problem Statement

Implement low pass and high pass filters in frequency domain for Ideal, Butterworth (order 2), and Gaussian kernels and apply those on Fig.1. Discuss about the results.

Figure 1

Library used: OpenCV

Method

1. Create Butterworth filter

   High pass filter: $1 - \frac{1}{\left(1+\frac{D_{(r,c)}}{radius}\right)^{2n}}$

   Low pass filter: $\frac{1}{\left(1+\frac{D_{(r,c)}}{radius}\right)^{2n}}$

2. Create Gaussian Filter

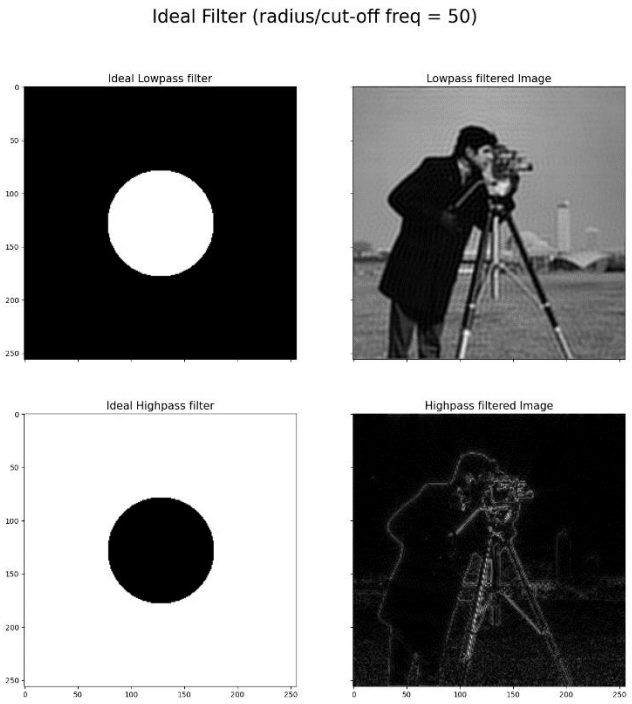
   High pass filter: $1 - e^{-D^2_{(r,c)}/(2*radius^2)}$

   Low pass filter: $e^{-D^2_{(r,c)}/(2*radius^2)}$

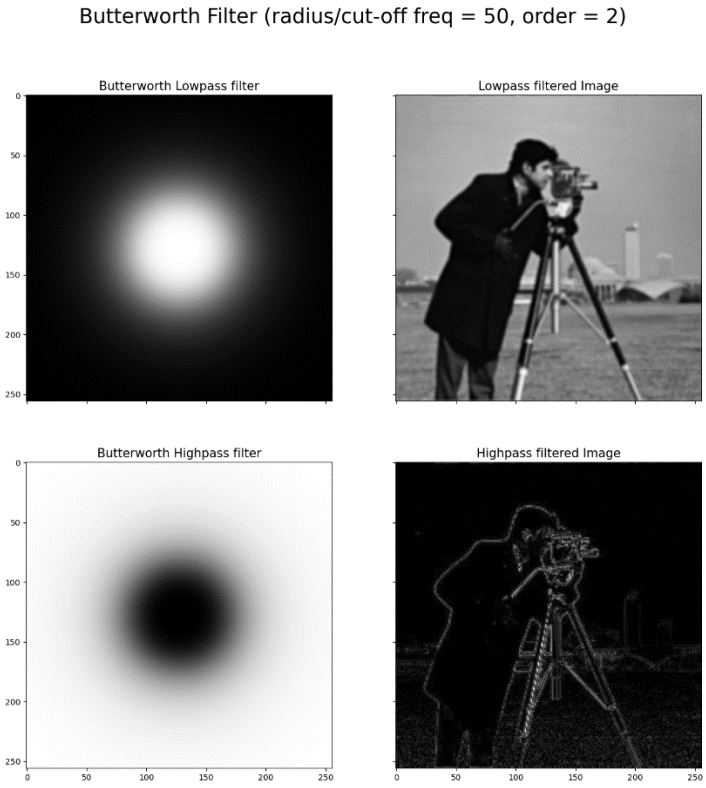
   $where, D_{(r,c)} := Distance\ matrix\ of\ image\ size$

Observation

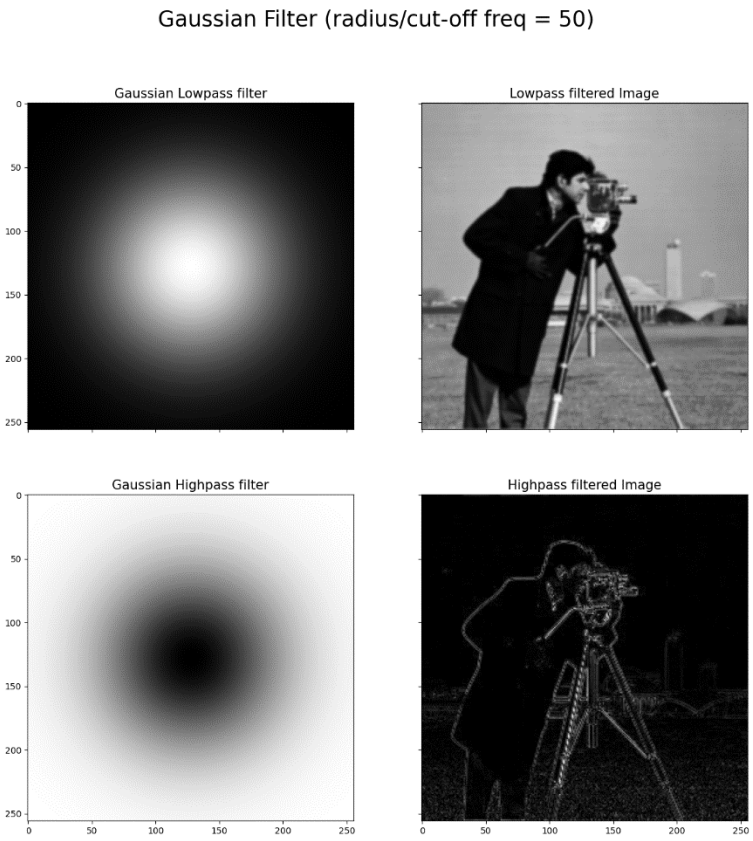
1. Ideal Low pass and High pass filter



2. Butterworth Low pass and High pass filter



3. Gaussian Low pass and High pass filter



Conclusion

- Low-pass filtering smoothens and blurs the image by averaging out rapid changes in the intensities while high pass filter tends to retain the high frequency information within an image while reducing the low frequency information.
- Ideal low-pass filtering introduces ringing artifacts in the image. Using Butterworth or Gaussian low-pass filtering removes it by smoothing the ideal filter.

Task 4

Problem Statement

Consider the image of Fig.1, and implement unsharp masking and high boost filtering in frequency domain. Compare the results with the implementation of spatial domain operation. Tune the parameters (cut-off frequency for frequency domain, kernel size for spatial domain, etc.) to achieve the same results in both domains. Report the results.

Library Used

Method

1. Equation used for unsharp & high boost filtering in spatial domain

   $$g_{mask}(x,y) = f(x,y) - \bar{f}(x,y)$$

   $$g(x,y) = f(x,y) + k * g_{mask}(x,y)$$

   $$where, k = 1 : Unsharp\ masking$$
   $$k > 1 : highboost\ filtering$$

   For high boost filtering we have used k = 2

   $\bar{f}(x,y)$: Is a smooth image of actual image, we have generated by use of gaussian blur.

   Spatial Domain filtering:

   `cv2.GaussianBlur(image, kernel_size=(3, 3), sigma=0.0) -> blurred image` $\bar{f}(x,y)$

   Frequency Domain filtering:

   Gaussian low-pass filtering using the equation used in Task 3 with radius 50 to obtain blurred image $\bar{f}(x,y)$

2. Function used for applying the unsharp masking equation

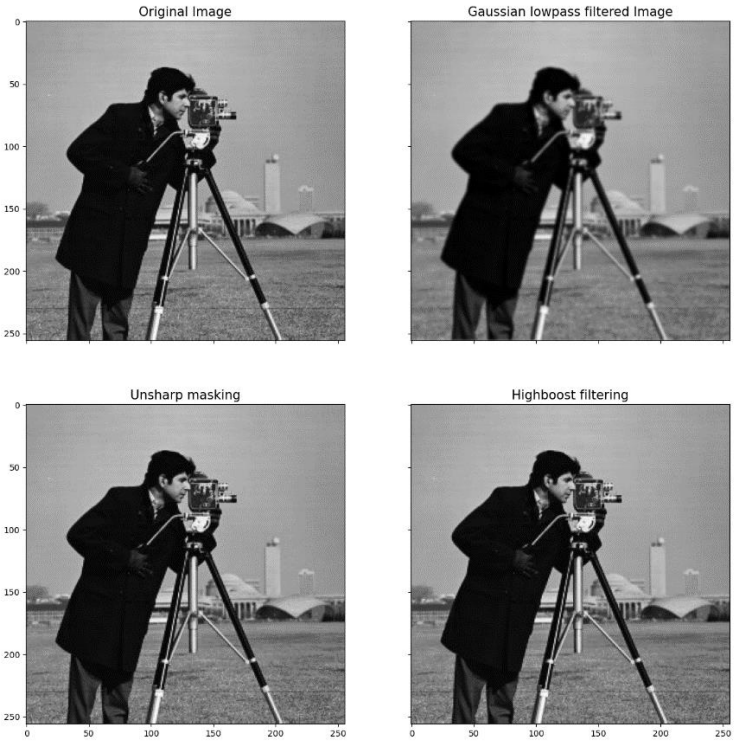
   $g(x,y)$ `= cv2.addWeighted(`$f(x,y)$`, k+1,` $\bar{f}(x,y)$`, -k)`

   $[g(x,y)$ `= (k+1)*` $f(x,y)$ `+ (-k)*` $\bar{f}(x,y) = f(x,y) + k * f(x,y) - k * \bar{f}(x,y) = f(x,y) + k * g_{mask}(x,y)]$

Observation

1. Spatial domain High boost and unsharp masking

2. Frequency domain high boost and unsharp masking



Conclusion

- Spatial domain unsharp masking and high boost filtering give better sharpening results while in frequency domain there is not much change.