Digital Image Processing: Assignment 7

Vaibhav Sharma: T23156

November 4, 2024

1 Introduction

In this report, we explore the effects of low-pass and high-pass filtering on images and analyze them in both spatial and frequency domains. We also verify that convolution in the spatial domain is equivalent to multiplication in the frequency domain.

1.1 Fourier Transform

The Fourier Transform decomposes an image into its frequency components, allowing us to analyze its spatial frequency content. The 2D Fourier Transform $F(u, v)$ of an image $f(x, y)$ is defined as:

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) e^{-2\pi i(ux+vy)} \, dx \, dy \tag{1}$$

and the inverse Fourier Transform is given by:

$$f(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u,v) e^{2\pi i(ux+vy)} \, du \, dv \tag{2}$$

1.2 Low-pass and High-pass Filters

A low-pass filter allows low frequencies to pass while attenuating high frequencies, resulting in a smoother image. Conversely, a high-pass filter allows high frequencies to pass, enhancing edges and fine details. Mathematically, these filters can be represented by functions $H(u, v)$ in the frequency domain.

2 Low-pass Filter

To apply a low-pass filter, we set up a mask $H(u, v)$ that passes low frequencies (close to the origin) and attenuates high frequencies. The process involves:

$$G_{\text{low}}(x,y) = \mathcal{F}^{-1}\{F(u,v) \cdot H_{\text{low}}(u,v)\} \tag{3}$$

The low-pass filter function transforms the image to the frequency domain, applies a mask, and returns to the spatial domain. Here's the code for the low-pass filter:

```
import numpy as np
class DFT(Matrix):
    ...
    def low_pass_filter(self, radius: int) -> 'DFT':
        dft = DFT()
        dft.data = np.zeros(self.data.shape, dtype=complex)
        for i in range(self.data.shape[0]):
            for j in range(self.data.shape[1]):
                if np.sqrt((i - self.data.shape[0]//2)**2 + (j - self.data.
                    shape[1]//2)**2) < radius:
                    dft.data[i,j] = self.data[i,j]
        return dft
```

3 High-pass Filter

The high-pass filter is applied by designing a mask $H_{\text{high}}(u, v)$ that passes high frequencies and attenuates low frequencies:

$$G_{\text{high}}(x, y) = \mathcal{F}^{-1}\{F(u, v) \cdot H_{\text{high}}(u, v)\} \tag{4}$$

The high-pass uses a mask that keeps high frequencies and removes low frequencies here is the code:

```
import numpy as np
class DFT(Matrix):
    ...
    def high_pass_filter(self, radius: int) -> 'DFT':
        dft = DFT()
        dft.data = np.zeros(self.data.shape, dtype=complex)
        for i in range(self.data.shape[0]):
            for j in range(self.data.shape[1]):
                if np.sqrt((i - self.data.shape[0]//2)**2 + (j - self.data.
                    shape[1]//2)**2) > radius:
                    dft.data[i,j] = self.data[i,j]

        return dft
```

4 Verification of Convolution and Multiplication Relationship

According to the Convolution Theorem, convolution in the spatial domain corresponds to multiplication in the frequency domain:

$$f(x, y) * h(x, y) \leftrightarrow F(u, v) \cdot H(u, v) \tag{5}$$

where $f(x, y) * h(x, y)$ is the convolution of f and h in the spatial domain, and $F(u, v) \cdot H(u, v)$ represents their product in the frequency domain.

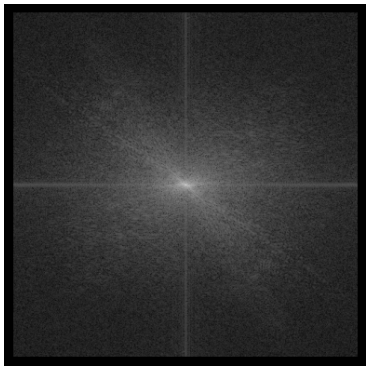
This relationship is validated by convolving the original image with a filter in the spatial domain and comparing it with the result of multiplying their Fourier transforms. To verify that convolution in the spatial domain corresponds to multiplication in the frequency domain, we use the following function:

```
import numpy as np
def verify_convolution_multiplication(image, kernel):
    # Fourier Transform of the image and kernel
    f_image = np.fft.fft2(image)
    f_kernel = np.fft.fft2(kernel, s=image.shape)

    # Convolution in spatial domain
    conv_spatial = cv2.filter2D(image, -1, kernel)

    # Multiplication in frequency domain
    f_mult = f_image * f_kernel
    conv_freq = np.fft.ifft2(f_mult)
    conv_freq = np.abs(conv_freq)

    return conv_spatial, conv_freq
```

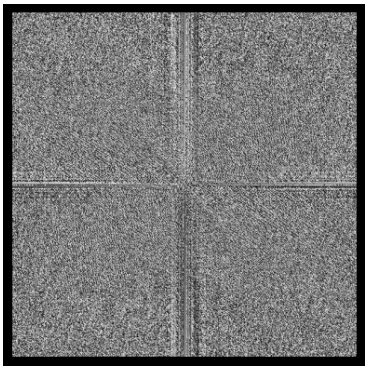
5 Results

5.1 Original Image and Fourier Transform

(a) Original Image



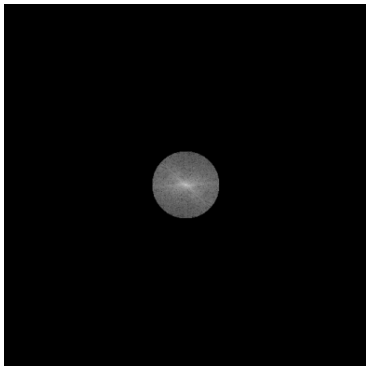
(b) Magnitude Spectrum



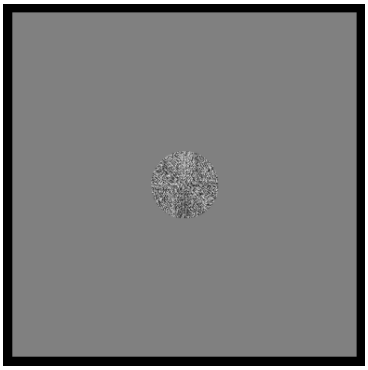
(c) Phase Spectrum

5.2 Low-pass Filter Results

(a) Filtered Image

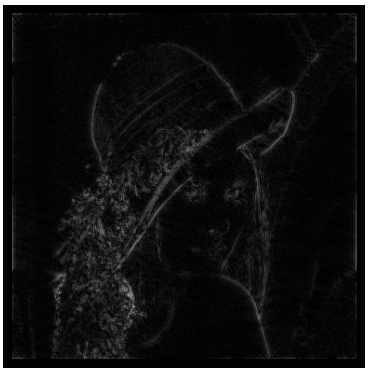


(b) Magnitude Spectrum

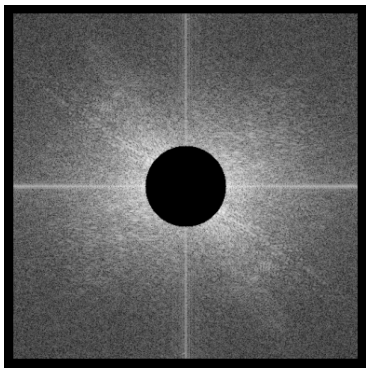


(c) Phase Spectrum

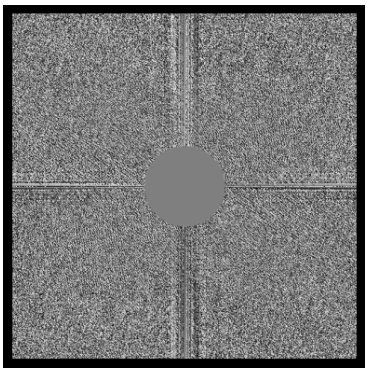
5.3 High-pass Filter Results



(a) Filtered Image



(b) Magnitude Spectrum



(c) Phase Spectrum

5.4 Verification of Convolution and Multiplication Relationship

Figure 4: Spatial Domain Convolution (Left) and Frequency Domain Multiplication (Right)

6 Conclusion

This experiment demonstrated the effects of low-pass and high-pass filtering on an image and validated the convolution-multiplication relationship between the spatial and frequency domains. The results confirm the Fourier properties and highlight the practical uses of filtering in image processing.

Code Availability

All the code used in this project is also available in a public GitHub repository. You can access it at: https://github.com/Computer-Science-Practicum/DIP-Lab-Assignment.