University of Rajshahi

Department of Computer Science & Engineering

CSE4182 - Digital Image Processing Lab

Explaining Fourier Transformation effect on image

Prepared by: Abdur Rahim Sheikh

student id: 1810576141

Instructor: Sangeeta Biswas, associate professor

Date: August 14, 2022

Abstract

The Fourier Transform is used to work if signal as well as image in frequency domain. And a fastest version of Fourier Transform which is fft (Fast Fourier Transform) can be used to work with any signal in frequency domain with minimal time complexity and more flexibility.

Contents

1 Introduction

To understand the effect of Fourier Transform we need an image and here is the image I chose to apply fast Fourier transform.

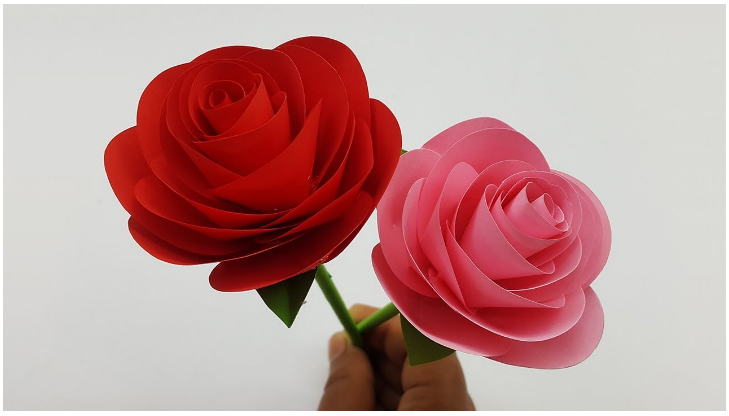

Figure 1.1: red rose

2 Methods

This is an RGB image. We will apply Fourier Transform in a gray-scale image. To convert gray-scale we will use the python module OpenCV called cv2.

2.1 Transform to Gray-scale

```
img = plt.imread('rose.jpg')
src_img = cv2.cvtColor(img,cv2.COLOR_RGB2GRAY)
plt.imshow(src_img)
```

2.2 Fourier Transform

In this report, we used Numpy's built-in function FFT2 (2 because our image is a 2D signal). Then centered the Fourier transformed image by using "fftshift" function. Then, we will calculate the magnitude spectrum.

```
fftImg = np.fft.fft2(src_img)
centerImg = np.fft.fftshift(fftImg)
centerImgLog = 100 * np.log(abs(centerImg))
```

2.3 Filters

I created 4 filter kind of arbitary but those are 3 rectengle of different size and one circle.

2.3.1 Filter1

It's a 100 by 100 rectangle in the center.

```
filter1 = np.zeros((r,c),np.uint8)
filter1= cv2.rectangle(filter1,(oy-50,ox-50),(oy+50,ox+50),(255,255,255),-1)
imageBack1 = centerImg * filter1
filterImg1 = np.abs(np.fft.ifft2(imageBack1))
```

Output:

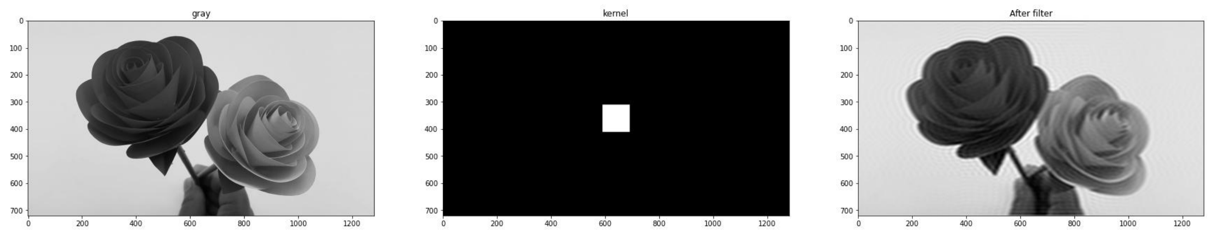

Figure 2.1: Filter 1 applied in frequency domain(Fourier Transformed)

Analysis: there is some circular glow around the image.

2.3.2 Filter2

This filter is a circle in the center.

```
filter2 = np.zeros((r,c),np.uint8)
filter2 = cv2.circle(filter2,(oy,ox),70,(255,255,255),-1)
imageBack2 = centerImg * filter2
filterImg2 = np.abs(np.fft.ifft2(imageBack2))
```

Output:

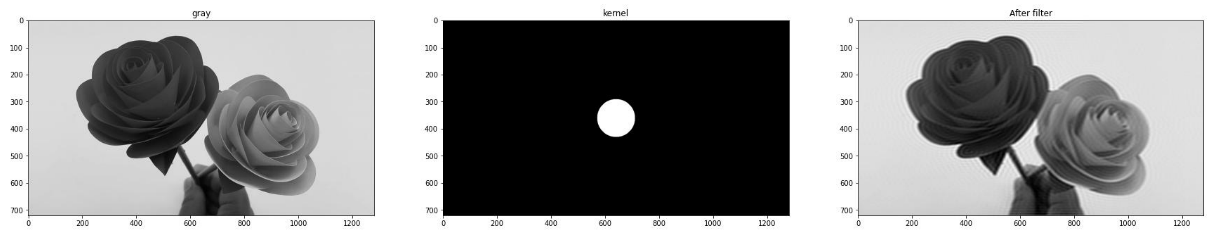

Figure 2.2: Filter 2 applied in frequency domain(Fourier Transformed)

Analysis: This also have circular glow but it's more sharp then the rectangle.

2.3.3 Filter3

The idea was to create a filter that contained half of all the values 1 and the others zero (axis=column). It is the same as above but the difference is we applied it with a column axis.

```
filter3 = np.zeros((r,c),np.uint8)
filter3[:oy,:] = 1
imageBack3 = centerImg * filter3
filterImg3 = np.abs(np.fft.ifft2(imageBack3))
```

Output:

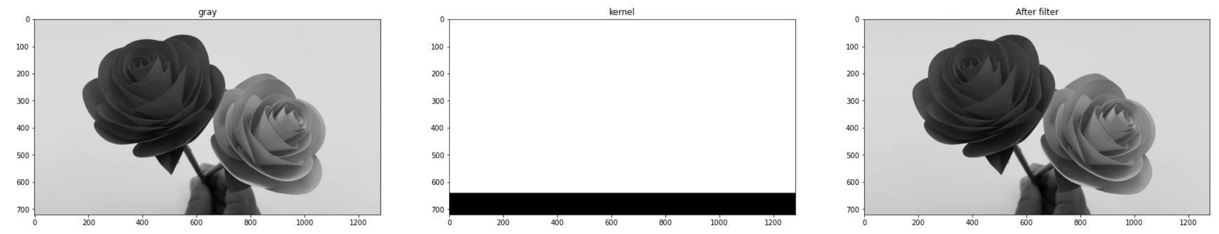

Figure 2.3: Filter 3 applied in frequency domain(Fourier Transformed)

Analysis: No effect seen.

2.3.4 Filter4

It's another rectangle.

```
filter4 = np.zeros((r,c),np.uint8)
filter4[:ox+30,:oy+90] = 1
imageBack4 = centerImg * filter4
filterImg4 = np.abs(np.fft.ifft2(imageBack4))
```

Output:

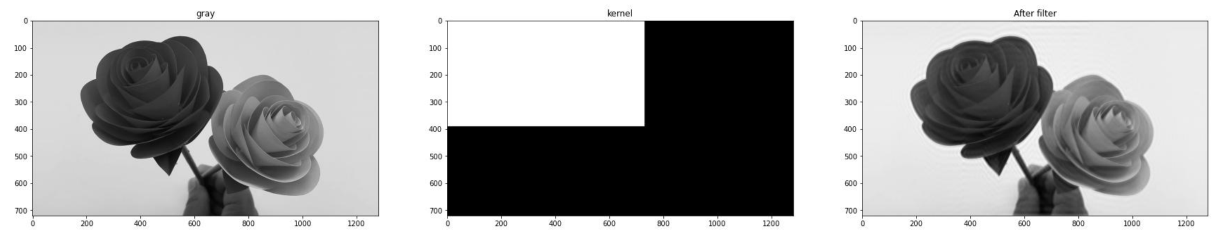

Figure 2.4: Filter 4 applied in the frequency domain(Fourier Transformed)

3 Conclusion

The main advantage of Fourier transform is that we can apply filters in the frequency domain. And it's time complexity is less. So it's a great tool to filter images and do other application with it.

4 visualization code

```
def show_transformation(gray,kernel,filtered,img_name):
plt.figure(figsize=(30,20))
def plot_it(img,title,ind):
    plt.subplot(1,3,ind)
    plt.imshow(img,cmap='gray')
    plt.title(title)

plot_it(gray,'gray',1)
plot_it(kernel,'kernel',2)
plot_it(filtered,'After filter',3)
plt.savefig(img_name,bbox_inches='tight')
plt.show()
```

List of Figures