Image Processing

Digital Assignment 2

Name: Abhinav Kumar

Reg. No: 15BCE1046

Image Processing in OpenCV

Fourier Transform

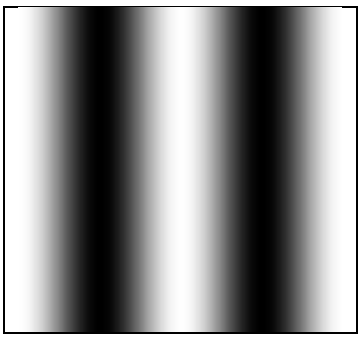
Brief Description

The Fourier Transform is an important image processing tool which is used to decompose an image into its sine and cosine components. The output of the transformation represents the image in the 'Fourier' or 'Frequency Domain', while the input image is the spatial domain equivalent. In the Fourier domain image, each point represents a particular frequency contained in the spatial domain image.

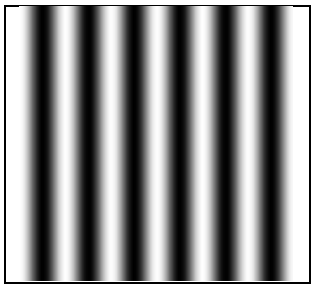
The Fourier Transform is used in a wide range of applications, such as image analysis, image filtering, image reconstruction and image compression.

Basic Principles

Fourier theory states that any signal, in our case visual images, can be expressed as a sum of a series of sinusoids. In the case of imagery, these are sine variations in brightness across the image. For example the sine pattern shown below can be captured in a single Fourier term that encodes 1. The spatial frequency, 2. The magnitude (positive of negative), and 3. The phase.



The three values capture all of the information in the sinusoidal image. The spatial frequency is the frequency across space with which the brightness modulates. For example, the image below shows another sinusoidal with a higher frequency.



For a square image of size N*N, the 2D DFT is given by:

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j)\, e^{-\iota 2\pi (\frac{ki}{N} + \frac{lj}{N})}$$

Now we will use OpenCV to use Fourier Transform:

In [219]:
```
import cv2
```

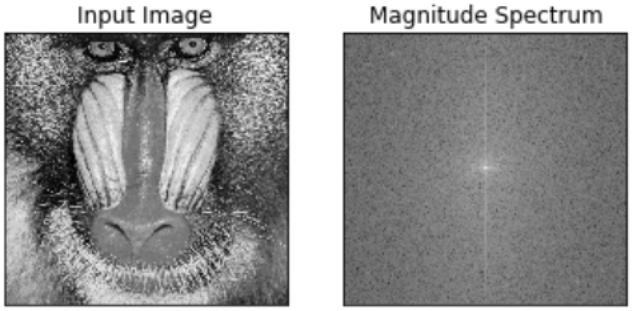
In [220]:
```
import numpy as np
```

In [221]:
```
from matplotlib import pyplot as plt
```

In [222]:
```
mandrill_img = cv2.imread('Downloads/mandrill.jpeg')
gray_1 = cv2.cvtColor(mandrill_img,cv2.COLOR_BGR2GRAY)
```

In [223]:
```
f = np.fft.fft2(gray_1)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20*np.log(np.abs(fshift))
```

In [224]:
```
plt.subplot(121),plt.imshow(gray_1, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```

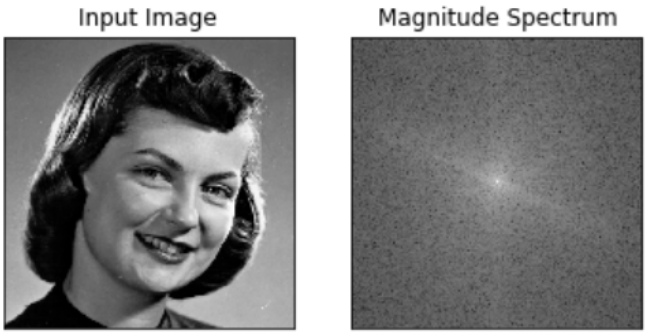


In [225]:
```
#So now we know how to convert an image into Frequency domain
```

In [230]:
```
gray_2 = cv2.imread('Downloads/girlface.bmp',0)
#gray_2 = cv2.cvtColor(sample,cv2.COLOR_BGR2GRAY)
```

In [231]:
```
f1 = np.fft.fft2(gray_2)
fshift1 = np.fft.fftshift(f1)
magnitude_spectrum1 = 20*np.log(np.abs(fshift1))
```

In [232]:
```
plt.subplot(121),plt.imshow(gray_2, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(magnitude_spectrum1, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
fshift.shape
```



Out[232]: (480, 500)

Now that we have got the frequency domain of the image we can use frequency domain filtering in the Magnitude Spectrum.

We can use HPF (High pass filtering) to detect the edges:

In [233]:
```
fshift1[200:280,210:290] = 0
```

In [234]:
```
f_ishift = np.fft.ifftshift(fshift1)
img_back = np.fft.ifft2(f_ishift)
img_back = np.abs(img_back)
```

In [236]:
```
plt.subplot(121),plt.imshow(gray_2, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Image after HPF'), plt.xticks([]), plt.yticks([])
plt.show
```

Out[236]:
```
<function matplotlib.pyplot.show>
```