FPT Education
FPT UNIVERSITY

Fourier Transform in Image Processing

OpenCV-Python Tutorial

https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_transforms/py_fourier_transform/py_fourier_transform.html#additional-resources

GOAL

- To find the **Fourier Transform** of images using *OpenCV*
- To utilize the **Fast Fourier Transform** (FFT) functions available in *Numpy*
- To see some applications of Fourier Transform
- To see following functions: *cv2.dft()*, *cv2.idft()*, …

FPT Education
FPT UNIVERSITY

INTRODUCTION

- **Fourier Transform** is used to analyze the frequency characteristics of various filters.
- For images, **2D Discrete Fourier Transform (DFT)** is used to find the frequency domain.
- A fast algorithm called **Fast Fourier Transform (FFT)** is used for calculation of DFT.

INTRODUCTION

- For a sinusoidal signal $x(t) = A\sin(2\pi ft)$, we can say f if the frequency of signal, and if its frequency domain is taken, we can see a spike at f.
- If signal is sampled to form a discrete signal, we get the same frequency domain, but is periodic in the range $[-\pi, \pi]$ or $[0, 2\pi]$ (or $[0, N]$ for N-point DFT).
- You can consider an image as a signal which *is sampled in two directions* → **taking Fourier transform in both X and Y directions** gives the frequency representation of image.

FPT Education
FPT UNIVERSITY

INTRODUCTION

- For the sinusoidal signal, if the **amplitude** varies so *fast* in short time.

→ a **high frequency** signal.

- If it varies *slowly*, it is a **low frequency** signal.

→ extend the same idea to **images**.

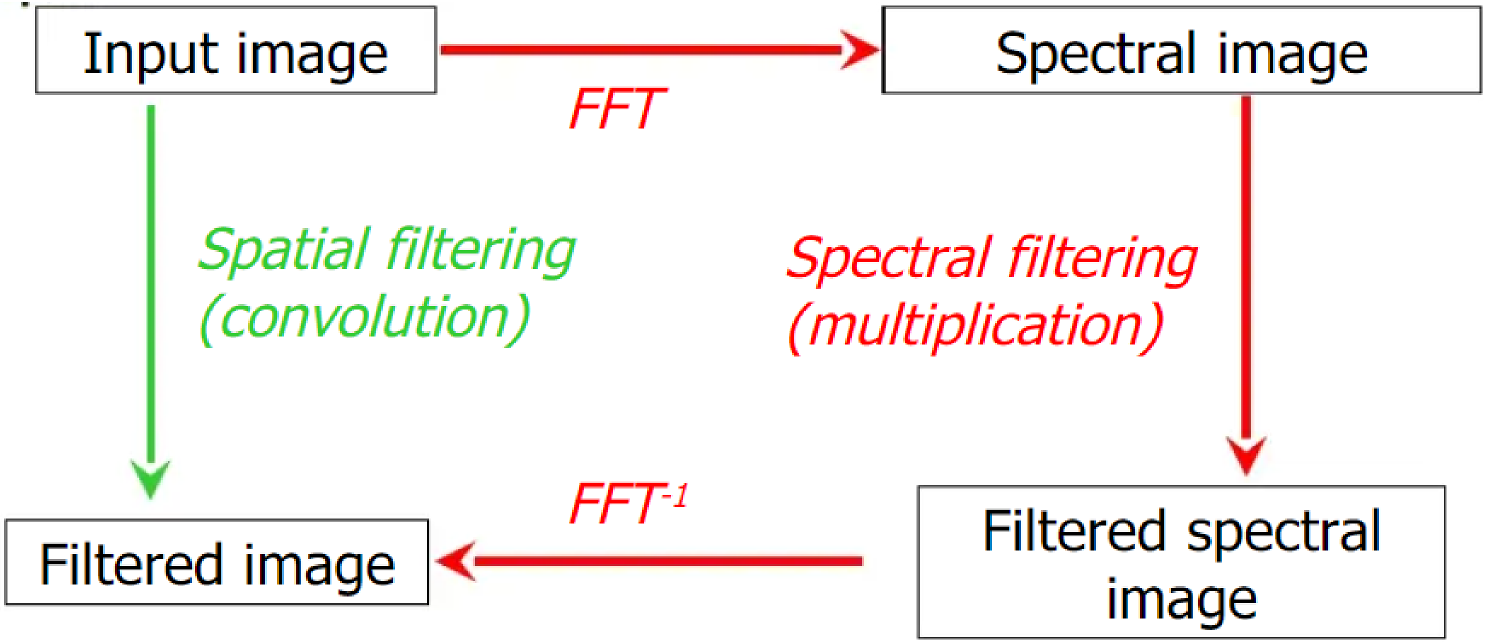
- **Where does the amplitude varies drastically in images?**

→ At the **edge** points, or **noises**.

- **Edges** and *noises* are **high frequency contents** in an image.
- If there is ***no much changes*** *in amplitude*, it is a **low frequency component**.

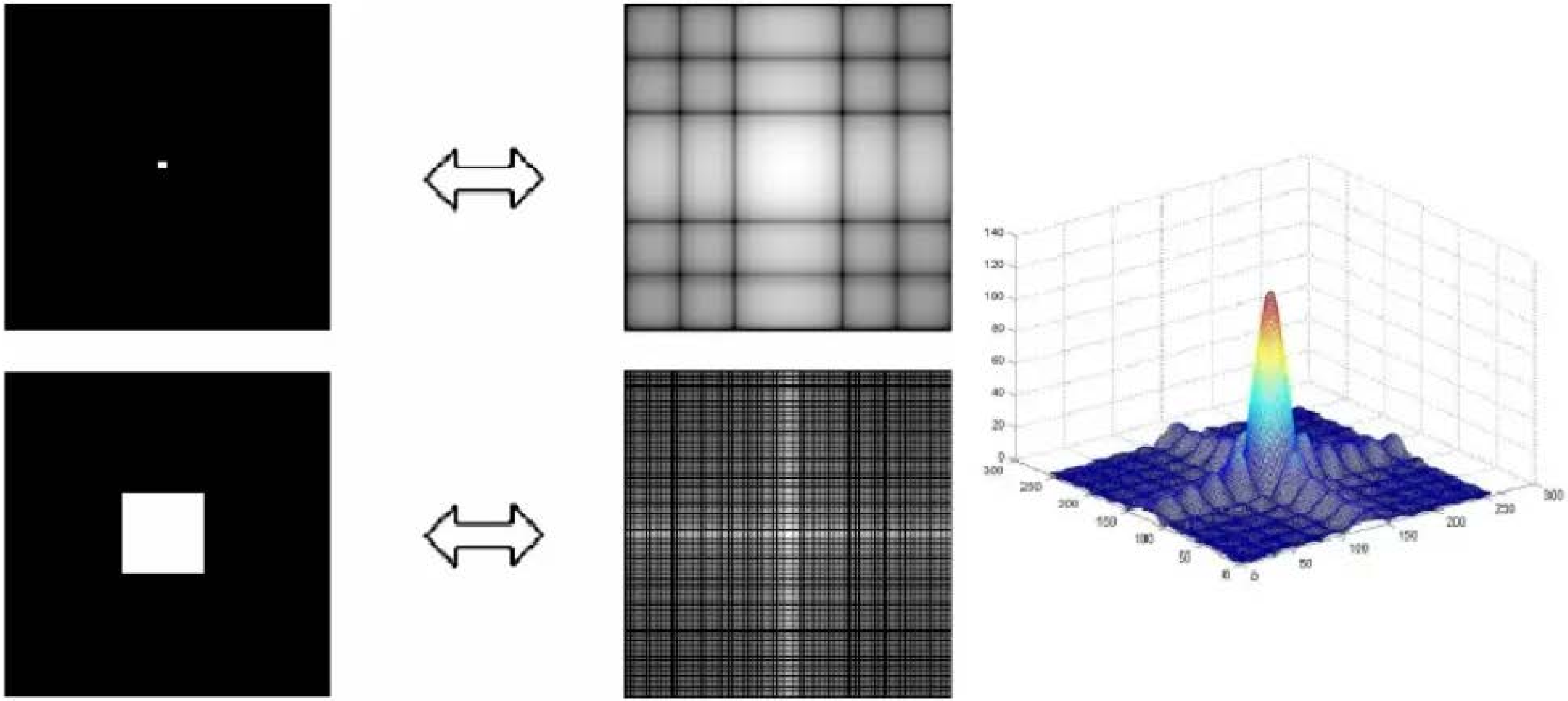
Low frequency

High frequency



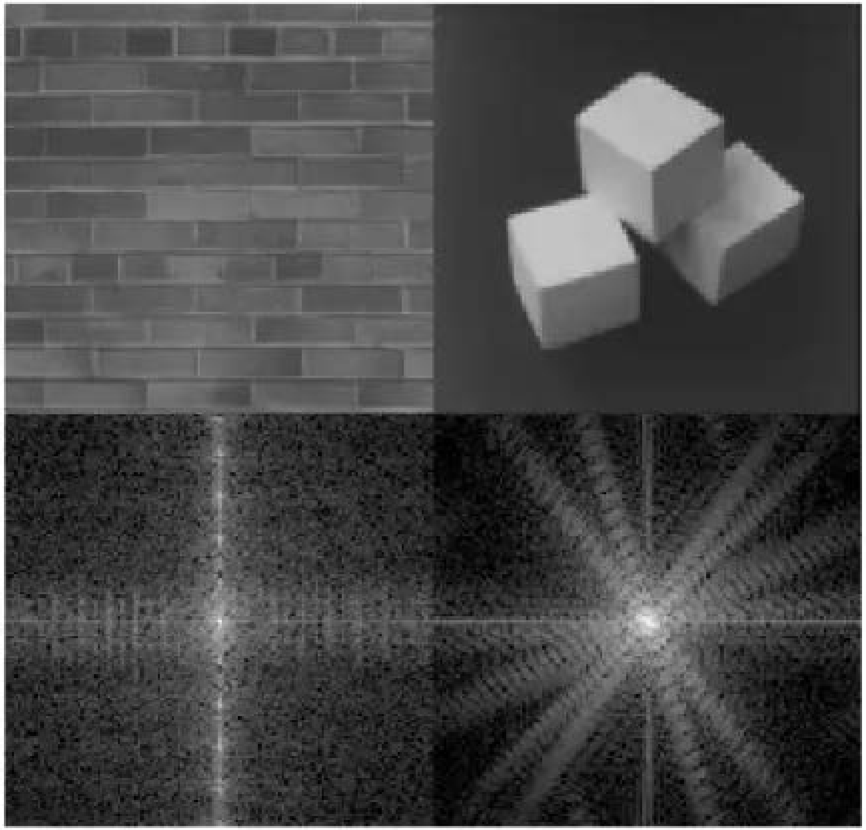
*In the **spatial domain**, filtering is done using **convolution**. In the **spectral domain (or frequential)**, it is done using **multiplication** (or image **masking**).*

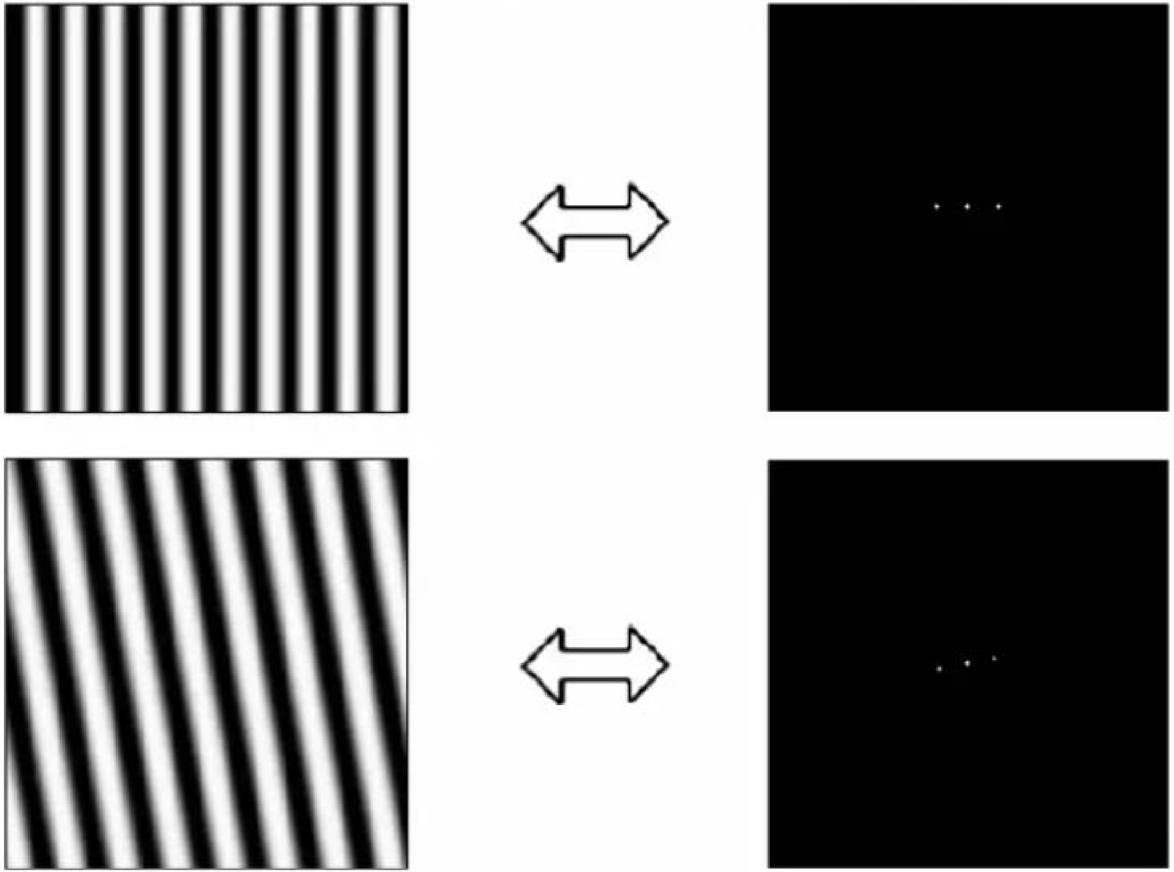
In the case of non-multiplicative filter in the spectral domaine, we cannot abtain the same result in the spatial domain. For non-linear spatial filters, we cannot also obtain the same result in the spectral domain.



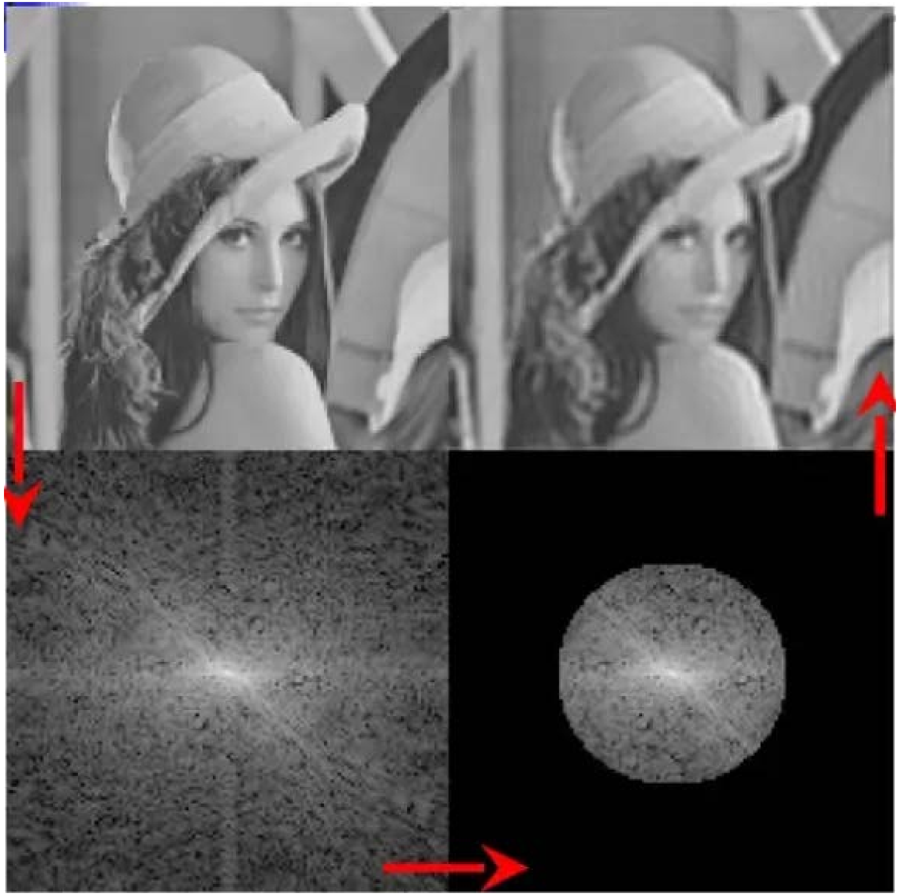
Input images FFT 3D view of the FFT

Source : *Thomas Guyet. Images numériques. IUT Sérécom (France).*

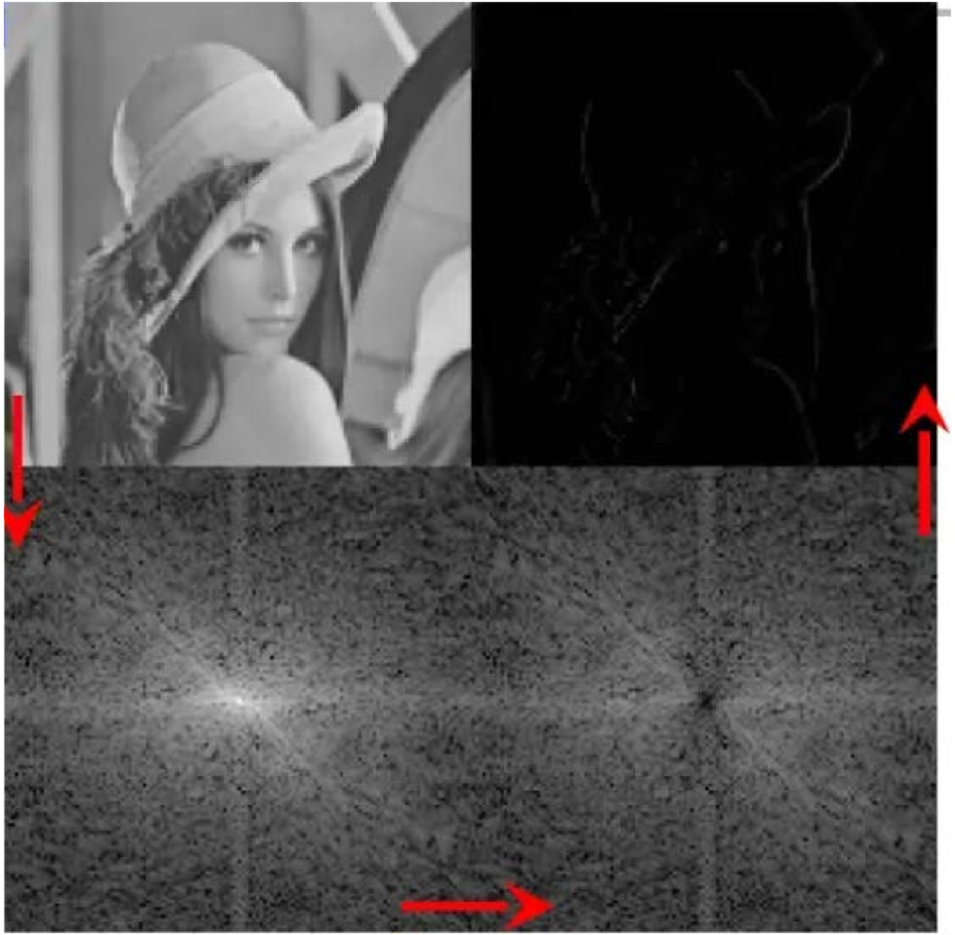




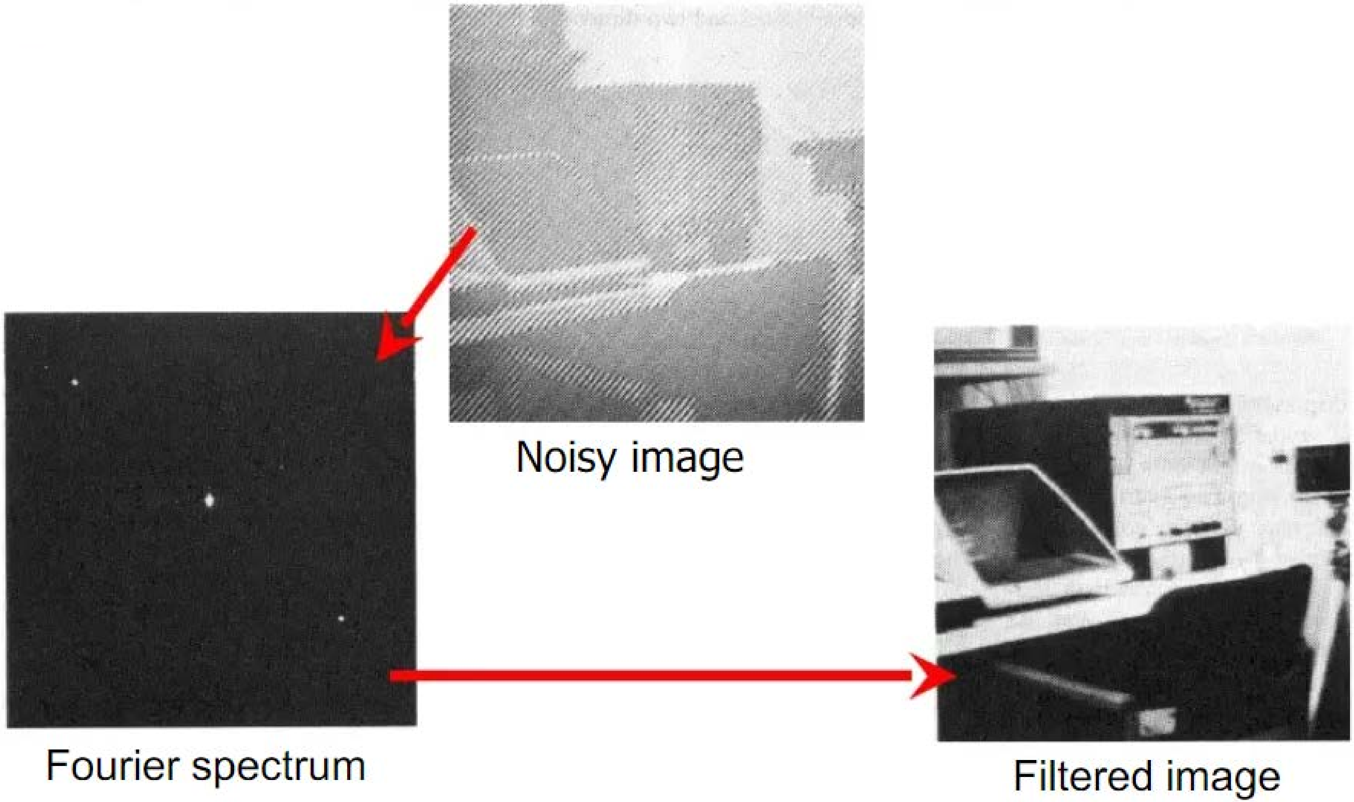
Rotate an angle



Clear the high frequencies in FFT by setting the far pixels (from center) to zero.



Clear the low frequencies in FFT by setting the near pixels (from center) to zero.


Noisy image
Fourier spectrum
Filtered image

FOURIER TRANSFORM IN NUMPY

- **Numpy** has an **FFT package** to do Fourier Transform.
- **np.fft.fft2()** provides us the *frequency transform* which will be a complex array.

np.fft.fft2(a, s=None)

- The first argument **a** is the input image, which is grayscale.
- Second argument **s** is optional which decides the size of output array. If it is greater than size of input image, input image is padded with zeros before calculation of FFT. If *no arguments passed*, **output array size will be same as input**.

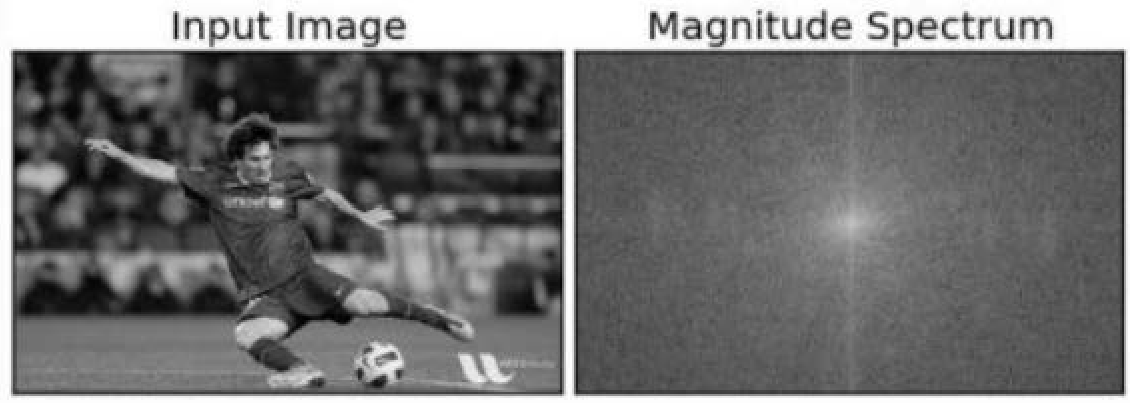
FOURIER TRANSFORM IN NUMPY

- When you got the result, **zero frequency** component (DC component) will be at top left corner.
- If you want to **bring it to center**, you need to **shift the result by** $\frac{N}{2}$ in both directions.
- This is simply done by the function, **np.fft.fftshift()**. (It is more easier to analyze).
- Once you found the frequency transform, you can find the **magnitude spectrum**.

FPT Education
FPT UNIVERSITY

FOURIER TRANSFORM IN NUMPY

- You can see more **whiter region at the center** showing **low frequency content** is more.



FOURIER TRANSFORM IN NUMPY

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('messi5.jpg',0)
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20*np.log(np.abs(fshift))

plt.subplot(121),plt.imshow(img, 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()
```

FOURIER TRANSFORM IN NUMPY

- So we found the **frequency transform**, we can do **some operations** in **frequency domain**, like **high pass filtering** and **reconstruct the image**, i.e. find **inverse DFT**.
- For that we simply **remove the low frequencies** by **masking** with a rectangular window of size **60x60**.
- Next, we apply the **inverse shift** using **np.fft.ifftshift()** so that **zero frequency component** again come at the **top-left corner**.
- Then we find inverse FFT using **np.ifft2()** function.
- The result will be a **complex number** → take its **absolute value**.

```
rows, cols = img.shape
crow,ccol = rows/2 , cols/2
fshift[crow-30:crow+30, ccol-30:ccol+30] = 0
f_ishift = np.fft.ifftshift(fshift)
img_back = np.fft.ifft2(f_ishift)
img_back = np.abs(img_back)

plt.subplot(131),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(132),plt.imshow(img_back, cmap = 'gray')
plt.title('Image after HPF'), plt.xticks([]), plt.yticks([])
plt.subplot(133),plt.imshow(img_back)
plt.title('Result in JET'), plt.xticks([]), plt.yticks([])

plt.show()
```

This also shows that **most of the image data** is present in **the low frequency region of the spectrum**.

The result shows **High Pass Filtering** is an edge detection operation.

FOURIER TRANSFORM IN OPENCV

- **OpenCV** provides the functions **cv2.dft()** and **cv2.idft()** for this. It returns the same result as previous, but with **two channels**.
- **First channel** will have the **real part** of the result and **second channel** will have the **imaginary part** of the result.
- The input image should be converted to np.float32 first.

FOURIER TRANSFORM IN OPENCV

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

img = cv2.imread('messi5.jpg',0)

dft = cv2.dft(np.float32(img),flags = cv2.DFT_COMPLEX_OUTPUT)
dft_shift = np.fft.fftshift(dft)

magnitude_spectrum = 20*np.log(cv2.magnitude(dft_shift[:,:,0],dft_shift[:,:,1]))

plt.subplot(121),plt.imshow(img, 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()
```

FOURIER TRANSFORM IN OPENCV

- This time we will see how to **remove high frequency contents** in the image, i.e. we apply **Low Pass Filtering** to image. It actually **blurs the image**.
- For this, we create a mask first **with high value (1) at low frequencies**, i.e. we pass the low frequency content, and **0 at high frequency region**.

```
rows, cols = img.shape
crow,ccol = rows/2 , cols/2

# create a mask first, center square is 1, remaining all zeros
mask = np.zeros((rows,cols,2),np.uint8)
mask[crow-30:crow+30, ccol-30:ccol+30] = 1

# apply mask and inverse DFT
fshift = dft_shift*mask
f_ishift = np.fft.ifftshift(fshift)
img_back = cv2.idft(f_ishift)
img_back = cv2.magnitude(img_back[:,:,0],img_back[:,:,1])

plt.subplot(121),plt.imshow(img, cmap = 'gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(img_back, cmap = 'gray')
plt.title('Magnitude Spectrum'), plt.xticks([]), plt.yticks([])
plt.show()
```

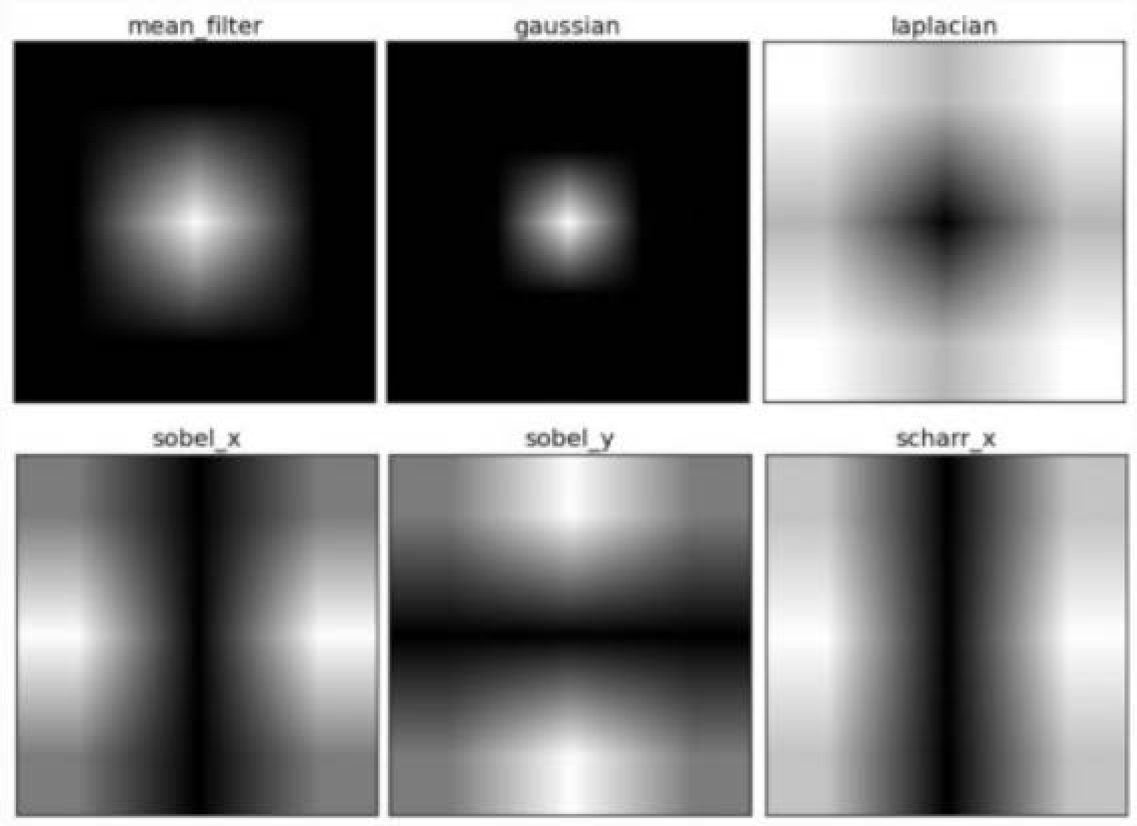
Why Laplacian is a High Pass Filter?

- **Why Laplacian is a high pass filter?**
- **Why Sobel is a high pass filter?**
- The first answer given to it was in terms of **Fourier Transform**. Just take the Fourier transform of Laplacian for some higher size of FFT.

```python
import cv2
import numpy as np
from matplotlib import pyplot as plt

# simple averaging filter without scaling parameter
mean_filter = np.ones((3,3))

# creating a guassian filter
x = cv2.getGaussianKernel(5,10)
gaussian = x*x.T

# different edge detecting filters
# scharr in x-direction
scharr = np.array([[-3, 0, 3],
                   [-10,0,10],
                   [-3, 0, 3]])
# sobel in x direction
sobel_x= np.array([[-1, 0, 1],
                   [-2, 0, 2],
                   [-1, 0, 1]])
# sobel in y direction
sobel_y= np.array([[-1,-2,-1],
                   [0, 0, 0],
                   [1, 2, 1]])
# laplacian
laplacian=np.array([[0, 1, 0],
                    [1,-4, 1],
                    [0, 1, 0]])
```

```python
filters = [mean_filter, gaussian, laplacian, sobel_x, sobel_y, scharr]
filter_name = ['mean_filter', 'gaussian','laplacian', 'sobel_x', \
                'sobel_y', 'scharr_x']
fft_filters = [np.fft.fft2(x) for x in filters]
fft_shift = [np.fft.fftshift(y) for y in fft_filters]
mag_spectrum = [np.log(np.abs(z)+1) for z in fft_shift]

for i in xrange(6):
    plt.subplot(2,3,i+1),plt.imshow(mag_spectrum[i],cmap = 'gray')
    plt.title(filter_name[i]), plt.xticks([]), plt.yticks([])

plt.show()
```

Why Laplacian is a High Pass Filter?



From image, we can see what **frequency region in each kernel blocks**, and what region it passes. From that information, we can say why **each kernel** is a **High Pass Filter** or a **Low Pass Filter**.