__Frequency Domain Filtering__

Steps in frequency domain filtering

- Compute the Fourier Transform
- Multiply the result by a filter transform function
- Take the inverse transform to produce the enchanced image

In [1]:
```
from scipy import fftpack
import numpy as np
import imageio.v2 as imageio
from PIL import Image, ImageDraw
import matplotlib.pyplot as plt
```

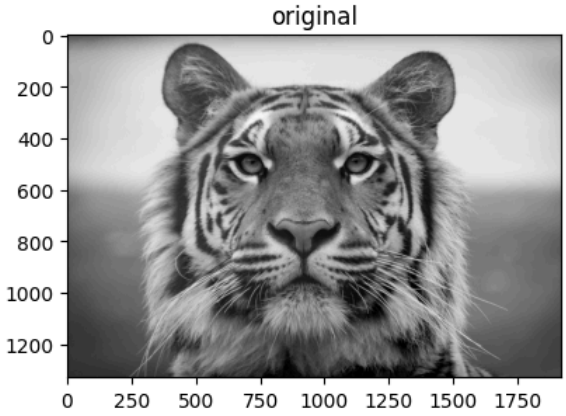
In [2]:
```
def frequency_image(image):
  image_array = np.array(image)
  fft1 = fftpack.fftshift(fftpack.fft2(image_array))
  magnitude_spectrum = 20*np.log(np.abs(fft1))
  freq_image = Image.fromarray(magnitude_spectrum)
  freq_image = freq_image.convert("L")
  return freq_image
```

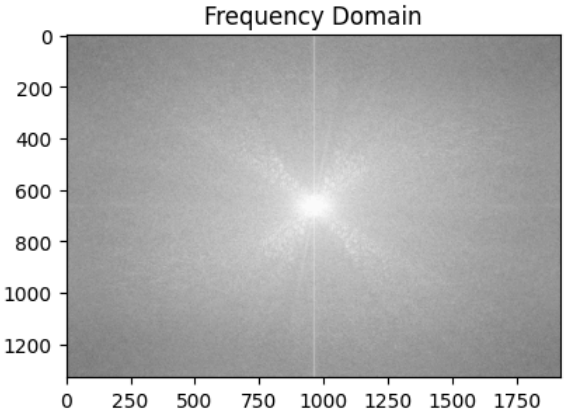
In [3]:
```
def low_pass_filter(image1):
  #convert image to numpy array
  image1_np=np.array(image1)
  #fft of image
  fft1 = fftpack.fftshift(fftpack.fft2(image1_np))
  #Create a low pass filter image
  x,y = image1_np.shape[0],image1_np.shape[1]

  #defining filter
  #size of circle
  e_x,e_y=50,50
  #create a box
  bbox=((x/2)-(e_x/2),(y/2)-(e_y/2),(x/2)+(e_x/2),(y/2)+(e_y/2))
  low_pass=Image.new("L",(image1_np.shape[0],image1_np.shape[1]),color=0)
  draw1=ImageDraw.Draw(low_pass)
  draw1.ellipse(bbox, fill=1)
  low_pass_np=np.array(low_pass)
  low_pass_np = low_pass_np.T
  #end of defining filter

  #multiply both the images
  filtered=np.multiply(fft1,low_pass_np)

  #inverse fft
  ifft2 = np.real(fftpack.ifft2(fftpack.ifftshift(filtered)))
  ifft2 = np.maximum(0, np.minimum(ifft2, 255))
  data = Image.fromarray(ifft2)
  data = data.convert("L")

  return data
```

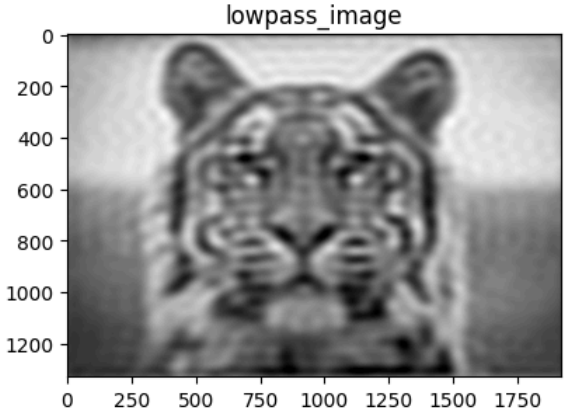
In [4]:
```
def high_pass_filter(image):
  #converting image to array
  image_array = np.array(image)

  #sending image to low pass filter
  lowpass_image = low_pass_filter(image)
  #converting image to array
  lowpass_image_array = np.array(lowpass_image)

  #subtracting lowpass image from original to obtain highpass image
  high_pass_array = image_array - lowpass_image_array

  #array to image
  high_pass_image = Image.fromarray(high_pass_array)
  high_pass_image = high_pass_image.convert("L")

  return high_pass_image
```

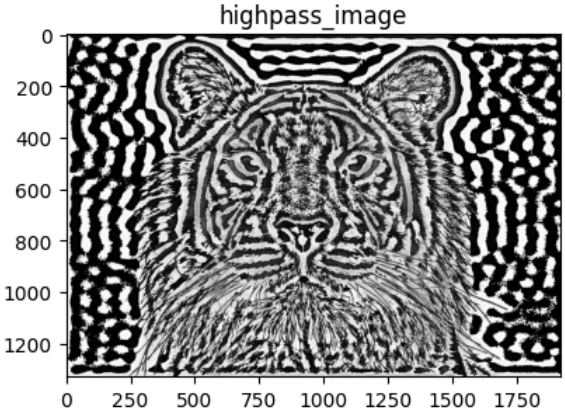
In [5]:
```
image = imageio.imread('../images/tiger.jpg', mode='L')
freq_image = frequency_image(image)
lowpass_image = low_pass_filter(image)
highpass_image = high_pass_filter(image)

fig = plt.figure()
fig.set_figheight(12)
fig.set_figwidth(10)

#plotting original image
fig.add_subplot(2,2,1)
plt.imshow(image, cmap='gray')
plt.title('original')

#plotting the image in frequency domain
fig.add_subplot(2,2,2)
plt.imshow(freq_image, cmap='gray')
plt.title('Frequency Domain')

#plotting lowpass image
fig.add_subplot(2,2,3)
plt.imshow(lowpass_image, cmap='gray')
plt.title('lowpass_image')

#plotting highpass image
fig.add_subplot(2,2,4)
plt.imshow(highpass_image, cmap='gray')
plt.title('highpass_image')
```

Out[5]:
```
Text(0.5, 1.0, 'highpass_image')
```


original
0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750


Frequency Domain
0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750


lowpass_image
0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750


highpass_image
0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750