ELE510 Image Processing with robot vision: LAB, Exercise 5, Frequency-domain processing.

Problem 1

The Fourier Transform is separable, that means that the two-dimensional transform is a sequence of two one-dimensional transforms. For images this can be considered as a transform along rows followed by a transform along columns (note that the input to the second step is the result from the first step, i.e. an image where the rows represents frequency and the columns space, $F(f_x, y)$). To get a better understanding of the **DFT** it is therefore convenient to study the one-dimensional transform:

$$G(k) = \sum_{x=0}^{w-1} g(x) e^{-j2\pi \frac{kx}{w}}, \qquad k = 0, 1, 2, \cdots, (w-1), \tag{1}$$

and its inverse, **IDFT**:

$$g(x) = \frac{1}{w} \sum_{k=0}^{w-1} G(k) e^{j2\pi \frac{kx}{w}}, \qquad x = 0, 1, 2, \cdots, (w-1). \tag{2}$$

One period of the signal is $g(x),\ x = 0, 1, 2, \cdots, (w-1)$ and in the frequency domain $F(k),\ k = 0, 1, 2, \cdots, (w-1)$.

a) Find the DC-component, $G(0)$. What does $\frac{G(0)}{w}$ represent?

To find the DC-component, $G(0)$, we can simply plug in $k = 0$ into the equation for the one-dimensional Discrete Fourier Transform (DFT):

$$G(0) = \sum_{x=0}^{w-1} g(x) e^{-j2\pi \frac{0x}{w}} = \sum_{x=0}^{w-1} g(x) e^{0} = \sum_{x=0}^{w-1} g(x). \tag{3}$$

So, $G(0)$ is equal to the sum of all values in the signal $g(x)$ for x ranging from 0 to $w-1$.

Now, let's consider the term $\frac{G(0)}{w}$. This term represents the average value of the signal $g(x)$ over the range of values from 0 to $w-1$. In other words, it's the mean value of the signal. So, $\frac{G(0)}{w}$ represents the DC (Direct Current) component of the signal, which is the average or mean value of the signal. This is often referred to as the "baseline" or the average intensity of the signal and can be thought of as the signal's component at zero frequency, which does not oscillate like the other frequency components in the DFT.

b) Show that the DFT is periodic, i.e. $G(k) = G(k + l \cdot w)$, where l is an arbitrary integer.

To show that the Discrete Fourier Transform (DFT) is periodic, we'll substitute $k + l \cdot w$ into the DFT equation and demonstrate that it yields the same result as k for any integer value of l.

The DFT equation is given as:

$$G(k) = \sum_{x=0}^{w-1} g(x) e^{-j2\pi \frac{kx}{w}}, \quad k = 0, 1, 2, \cdots, (w-1). \tag{4}$$

Now, let's substitute $k + l \cdot w$ into the equation:

$$\begin{aligned} G(k + l \cdot w) &= \sum_{x=0}^{w-1} g(x) e^{-j2\pi \frac{(k+l\cdot w)x}{w}} \\ &= \sum_{x=0}^{w-1} g(x) e^{-j2\pi \frac{kx}{w}} e^{-j2\pi lx} \quad \text{using properties of exponents} \\ &= \sum_{x=0}^{w-1} g(x) e^{-j2\pi \frac{kx}{w}} \cdot 1 \end{aligned}$$

since $e^{-j2\pi lx} = cos(-2\pi lx) + jsin(-2\pi lx) = 1 + j \cdot 0$.

So:

$$G(k + l \cdot w) = G(k) \quad \text{for any integer value of } l. \tag{5}$$

This demonstrates that the DFT is periodic with a period of w. In other words, the DFT values repeat every w samples in the frequency domain.

c) Find $G(k)$ for the centered box-function with 5 non-zero samples, $w = 16$.

$$g(x) = \begin{cases} 1 & \text{for} \quad x = 0, 1, 2 \text{ and } 14, 15. \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

To find the Discrete Fourier Transform (DFT) of the centered box function with 5 non-zero samples, we can use the formula for the DFT:

For $k = 0$:

$$\begin{aligned} G(0) &= \sum_{x=0}^{15} g(x) e^{-j2\pi \frac{0x}{16}} \\ &= \sum_{x=0}^{15} g(x) e^{0} \\ &= g(0) + g(1) + g(2) + g(14) + g(15) \\ &= 1 + 1 + 1 + 1 + 1 \\ &= 5 \end{aligned}$$

For $k = 1$:

$$\begin{aligned} G(1) &= \sum_{x=0}^{15} g(x) e^{-j2\pi \frac{1x}{16}} \\ &= \sum_{x=0}^{15} g(x) e^{-j\frac{\pi}{8}x} \\ &= g(0)e^{-j\frac{\pi}{8}0} + g(1)e^{-j\frac{\pi}{8}} + g(2)e^{-j\frac{\pi}{8}\cdot 2} + g(14)e^{-j\frac{\pi}{8}\cdot 14} + g(15)e^{-j\frac{\pi}{8}\cdot 15} \\ &= e^{-j\frac{\pi}{8}} + e^{-j\frac{\pi}{4}} + e^{-j\frac{7\pi}{8}} + e^{-j\frac{15\pi}{8}} \\ &= cos\left(-\frac{\pi}{8}\right) + jsin\left(-\frac{\pi}{8}\right) + cos\left(-\frac{\pi}{4}\right) + jsin\left(-\frac{\pi}{4}\right) + cos\left(-\frac{7\pi}{8}\right) + jsin\left(-\frac{7\pi}{8}\right) \end{aligned}$$

The others can be calculated similarly. With x different from 0, 1, 2, 14, 15, the result is 0.

Problem 2

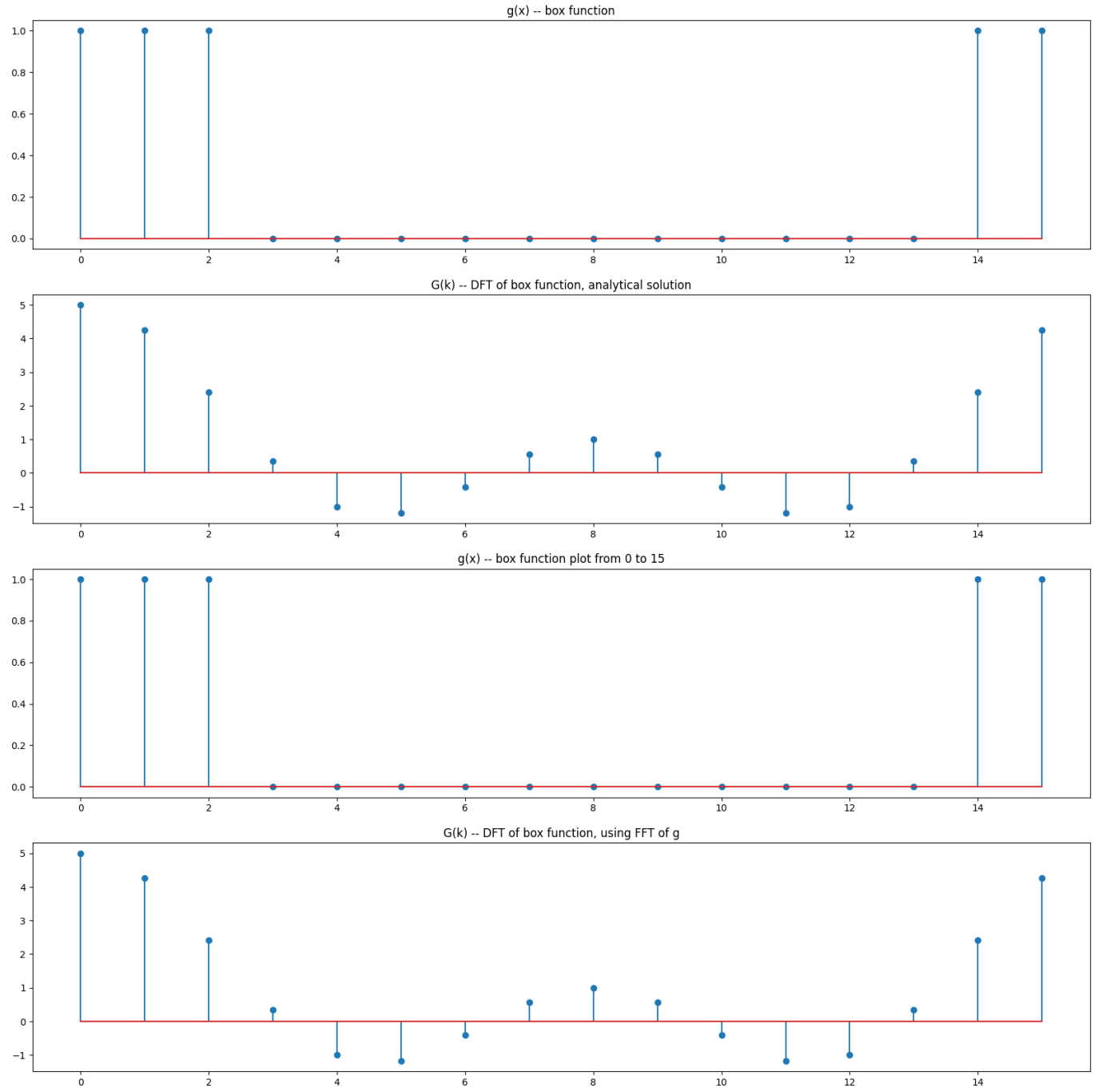
a) Use $g(x)$ as defined in **1 c)**, for $x \in [0, 15]$. Use **numpy.fft.fft** for finding the dft, $G(k)$. Plot both $g(x)$ and $G(k)$. Also plot the mathematical solution from probelm c) and see if / how they correspond.

You can also try to sketch $g(x)$ and $G(k)$ in the index range $-8, -7, \cdots, -1, 0, 1, 2, \cdots, 7$ (note the periodic property of both functions).

In [ ]:
```
import os
import matplotlib.pyplot as plt
import math
import numpy as np
import cv2
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
n = 16
f = np.arange(n)
F = np.zeros(n)
F[[0, 1, 2, 14, 15]]=1
print(F)


k = np.arange(-8, 8)
A = np.zeros(n, dtype=complex)
for k_val in k:
    A[k_val] = np.sum(F * np.exp(-1j * 2 * np.pi * k_val * f / n))


# Visualization of the result from the calculations, with zeros in the midle :
plt.figure(figsize=(20,20))
plt.subplot(411)
plt.stem(f,F)
plt.title('g(x) -- box function')

plt.subplot(412)
plt.stem(f,A)
plt.title('G(k) -- DFT of box function, analytical solution')


g = np.zeros(n)
g[[0, 1, 2, 14, 15]]=1

G = np.fft.fft(g)


plt.subplot(413)
plt.stem(g)
plt.title('g(x) -- box function plot from 0 to 15')
plt.subplot(414)
plt.stem(G)
plt.title('G(k) -- DFT of box function, using FFT of g')

plt.show()
```

```
[1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1.]
```



Problem 3

a)

In **Problem 2** the DFT (fft) was real - by coincidence. In general it is complex. In this part we will take in an image and look at it in space-domain (the image itself) and in frequency domain looking at just **the magnitude** of the DFT.

Useful functions : **numpy.fft.fft2** , **numpy.fft.fftshift** .

Import an image as a grayscale image. It can be your own image for fun (our just do soapbubbles.png). Fill inn the cell below, finding a zero mean version of the image and the DFT of both the iage and the zero mean image.

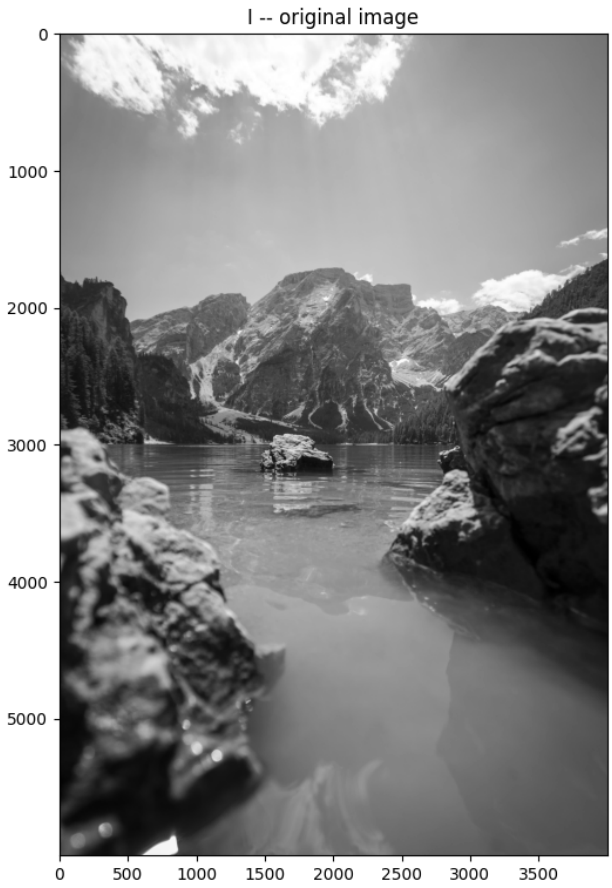
In [ ]:
```
# Import an image I as grayscale
imagepath = 'images/IMG_7944.jpg'
I = cv2.imread(imagepath, cv2.IMREAD_GRAYSCALE)

Iz = I- np.mean(I)

DFT_I = np.fft.fft2(I)
```

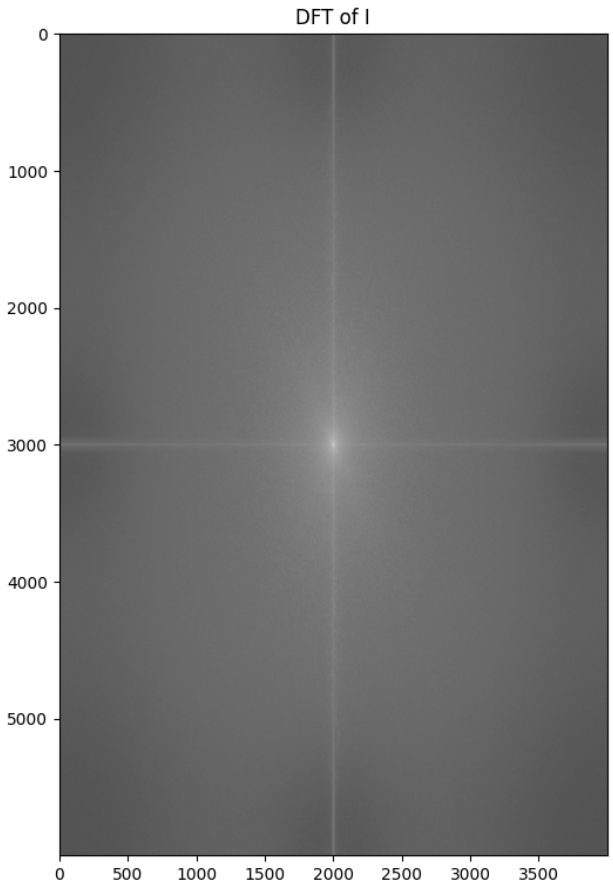
```
DFT_SHIFT_I = np.fft.fftshift(DFT_I)
DFT_Iz = np.fft.fft2(Iz)
DFT_SHIFT_Iz = np.fft.fftshift(DFT_Iz)
```

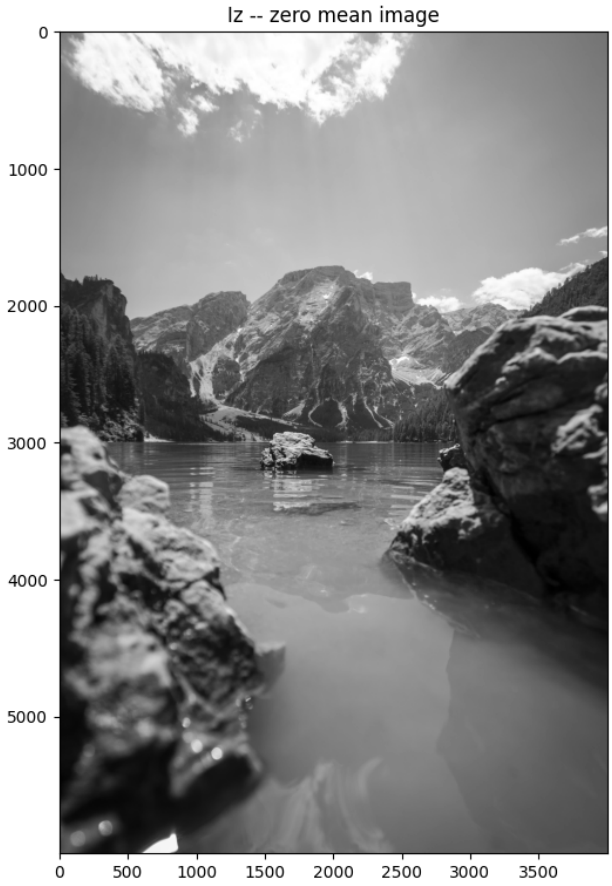
b)

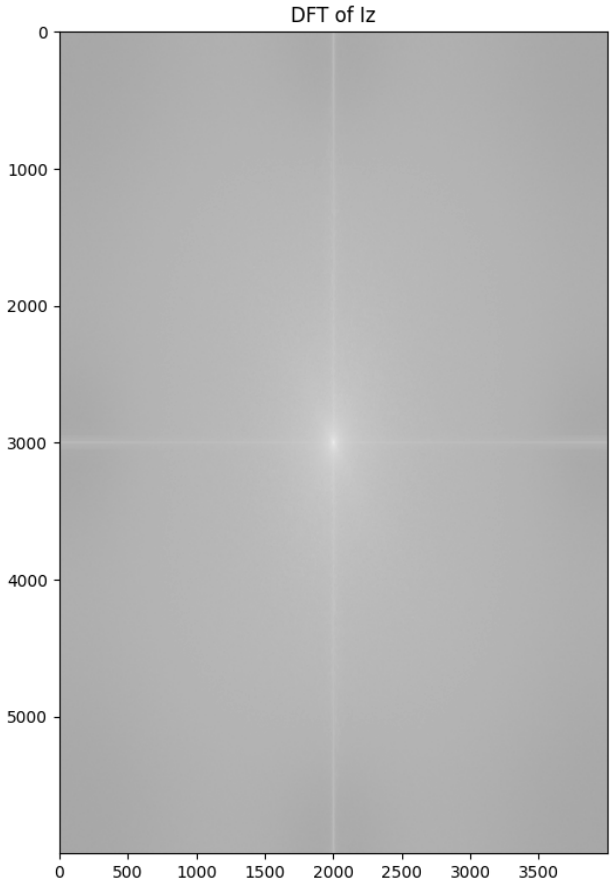
Now you will find the magnitude of the DFT of I and Iz. You are going to make a plot where you plot the image at the top, one row with the magnitude directly of I and Iz, and another where we do the scaling we do for display purposes as we talked about in class using the logarithm of the logarithm(1+magnitude(DFT)) .

In [ ]:
```
# plot here
plt.figure(figsize=(15,20))
plt.subplot(221)
plt.imshow(I, cmap='gray')
plt.title('I -- original image')

plt.subplot(222)
plt.imshow(np.log(np.abs(DFT_SHIFT_I)), cmap='gray')
plt.title('DFT of I')

plt.subplot(223)
plt.imshow(Iz, cmap='gray')
plt.title('Iz -- zero mean image')

plt.subplot(224)
plt.imshow(np.log(np.abs(DFT_SHIFT_Iz)), cmap='gray')
plt.title('DFT of Iz')

plt.show()
```









c) Comment on both the difference we see on the magnitude for I and Iz as well as with and without the scaling

The magnitude of the DFT of the zero-meaned image (Iz) has a shifted center frequency compared to the original image (I). This shift is due to the subtraction of the mean from the image, resulting in a DC component at the center of the DFT for Iz.

Scaling the magnitude of the DFT for display purposes (using log(1 + magnitude)) enhances the visibility of details in the frequency domain, especially for low-intensity components. Without scaling, the visualization might not effectively reveal the spectral content of the image. Scaling is commonly used to display DFT magnitudes because it compresses the dynamic range, making it easier to distinguish details in both high and low-frequency components. This can be particularly useful in applications like image processing and analysis.

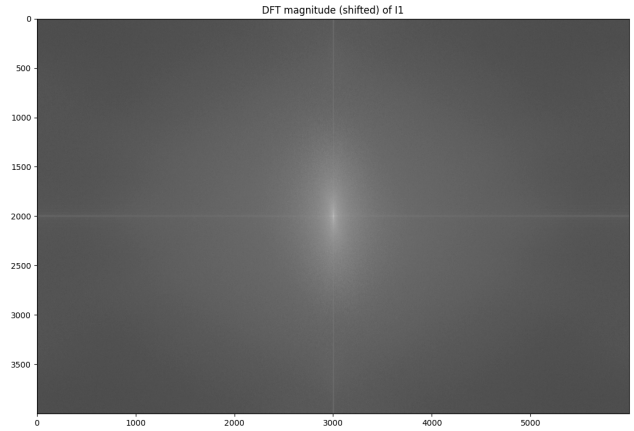
The image used its very high resolution so it mignt not be the perfect one to show the results.

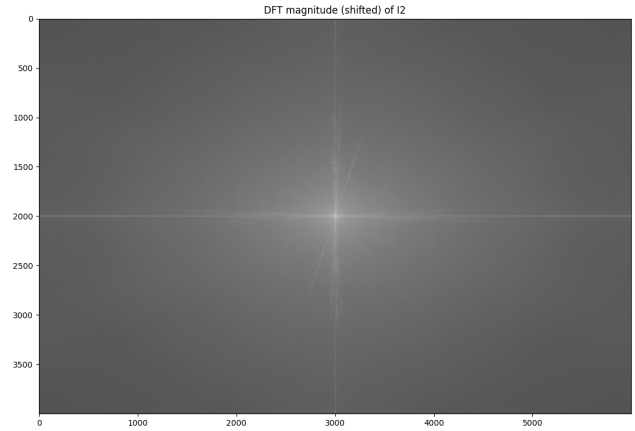
Problem 4

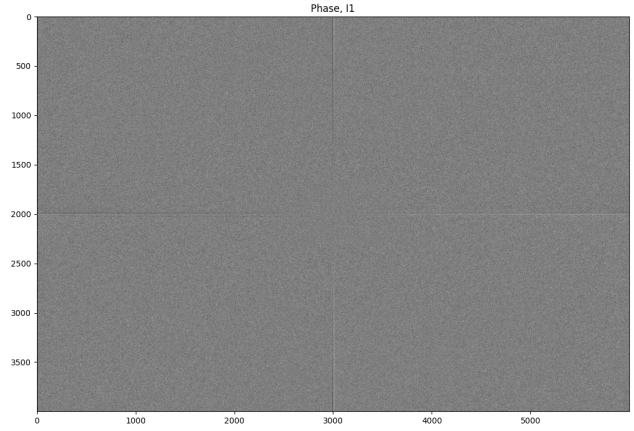
a)

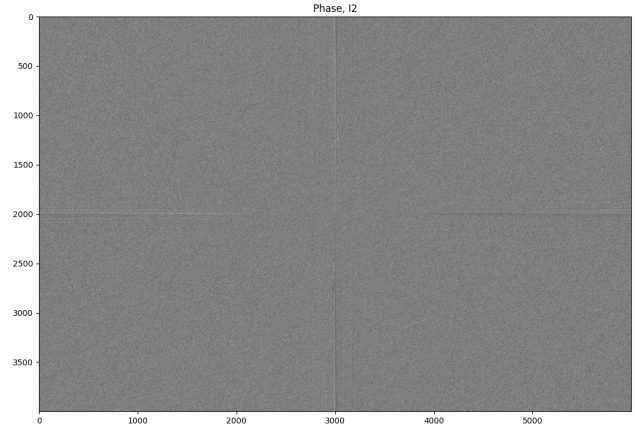
Now we will consider both the magnitude and the phase. Lets input two images convert to DFT and see we get the image back doing IDFT. Also convert to DFT and switch the phase between the images before IDFT and look at the results.
Use two images of choice. Just rescale or crop so you have to images the same size before doing the DFT. (alternative use BorderCollie.jpeg and zebra.jpeg , they are the same size) Plot the DFT (scaled magnitude and phase) and the reconstructed images with the right and wrong phase.

In [ ]:
```
I1 = cv2.imread('images/IMG_8907.JPG', cv2.IMREAD_GRAYSCALE)
I2 = cv2.imread('images/IMG_8922.JPG', cv2.IMREAD_GRAYSCALE)


DFT_I1 = np.fft.fftshift(np.fft.fft2(I1))
DFT_I2 = np.fft.fftshift(np.fft.fft2(I2))

# Compute the phase of DFT
phase_I1 = np.angle(DFT_I1)
phase_I2 = np.angle(DFT_I2)

plt.figure(figsize=(30,30))
plt.subplot(321),plt.imshow(I1, cmap="gray")
plt.title('original image I1')
plt.subplot(322),plt.imshow(I2, cmap="gray")
plt.title('original image I2')
plt.subplot(323),
plt.imshow(np.log(np.abs(DFT_I1) + 1),cmap="gray")
plt.title('DFT magnitude (shifted) of I1')
plt.subplot(324),
plt.imshow(np.log(np.abs(DFT_I2) + 1),cmap="gray")
plt.title('DFT magnitude (shifted) of I2')
plt.subplot(325)
plt.imshow(phase_I1,cmap="gray")
plt.title('Phase, I1')
plt.subplot(326),
plt.imshow(phase_I2 ,cmap="gray")
plt.title('Phase, I2')
plt.show()
```



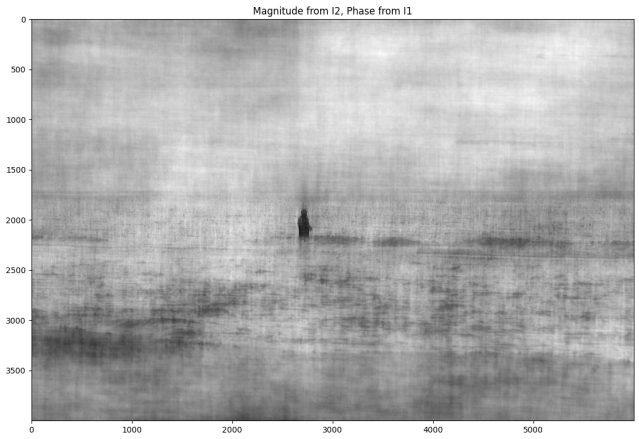


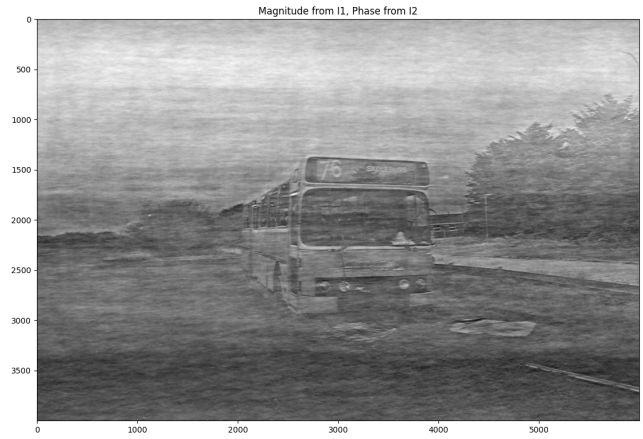




In [ ]:
```
#  Splitting up into real and imaginary, putting back together and do IDFT for one im
rec1 = abs(DFT_I1)
rec2 = abs(DFT_I2)

rec1 = np.real(np.fft.ifft2(np.fft.ifftshift(np.abs(DFT_I1) * np.cos(phase_I1)+1j * n

rec2 = np.real(np.fft.ifft2(np.fft.ifftshift(np.abs(DFT_I2) * np.cos(phase_I2) +1j *

# Splitting up into real and imaginary using wrong phase, putting back together␣ ,→an
rec_j1 = np.real(np.fft.ifft2(np.fft.ifftshift(np.abs(DFT_I1) * np.cos(phase_I2)+1j *
rec_j2 = np.real(np.fft.ifft2(np.fft.ifftshift(np.abs(DFT_I2) * np.cos(phase_I1)+1j *

plt.figure(figsize=(30,20))
plt.subplot(221)
plt.imshow(rec1,cmap="gray")
plt.title('Reconstructed I1, (magnitude from I1, Phase from I1)')
plt.subplot(222),
plt.imshow(rec2,cmap="gray")
plt.title('Reconstructed I2, (magnitude from I2, Phase from I2)')
plt.subplot(223)
plt.imshow(rec_j2,cmap="gray")
plt.title('Magnitude from I2, Phase from I1')
```

```
plt.subplot(224),
plt.imshow(rec_j1,cmap="gray")
plt.title('Magnitude from I1, Phase from I2')

plt.show()
```





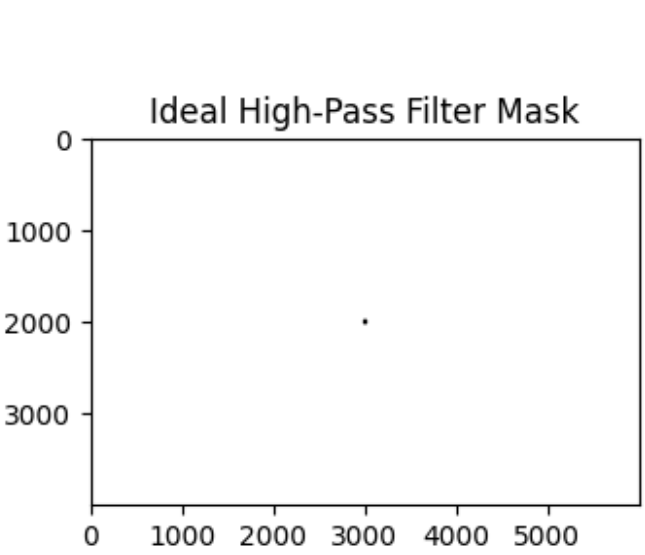
b)

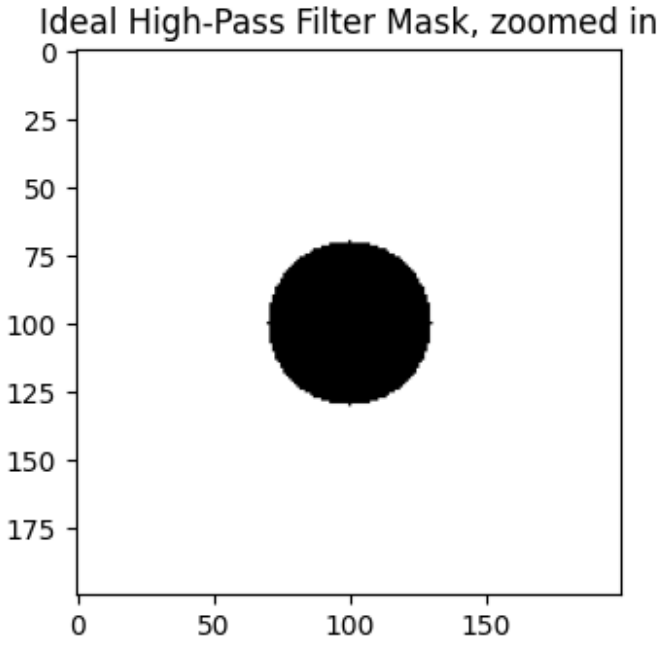
Take one of the images from the previous section , do the shift so the low frequencies are in the midle of the DFT images. Thereafter remove low frequencies (high pass filter) or remove high frequencies (loww pass filter) by an ideal filter in the frequency domain. How can you do that? Make and image the same size with just ones and zeros. Let there be zeros in a circle in the midle (for high pass). How big circle? That defines the **cutoff-frequency** . Try different sizes and see. Use this image as a filter. Remember convolution (filtering) in space domain is multiplication in the frequency domain.

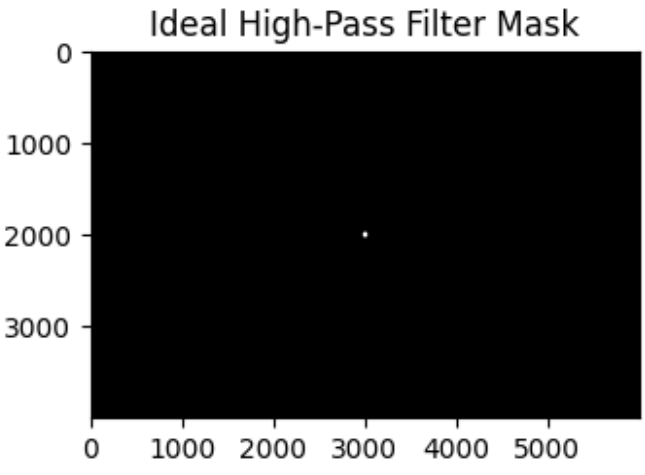
OBS when transforming back to space domain we might get a complex image since we have tampered with the complex DFT image. Plot the **magnitude** of the reconstructed image in the space-domain

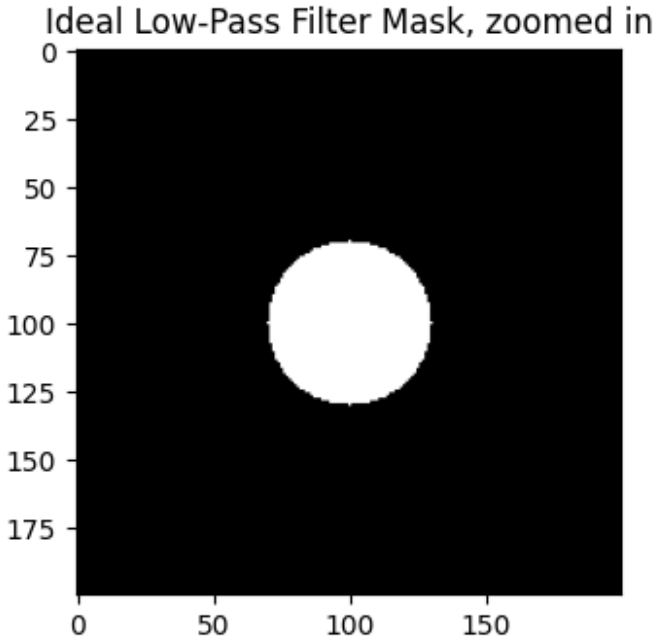
In [ ]:
```
# initialization:
high_pass_filter=np.ones_like(I1)
low_pass_filter=np.zeros_like(I1)
rows,cols = I1.shape

center_row, center_col = rows // 2, cols // 2

cutoff_frequency_hp = 30

#Create and plot the ideal high-pass filter mask here:
for i in range(rows):
    for j in range(cols):
```

```
        distance = np.sqrt((i - center_row) ** 2 + (j - center_col) ** 2)
        if distance <= cutoff_frequency_hp:
            high_pass_filter[i, j] = 0
        if distance <= cutoff_frequency_hp:
            low_pass_filter[i, j] = 1

plt.figure(figsize=(8, 8))
plt.subplot(221)
plt.imshow(high_pass_filter, cmap="gray")
plt.title('Ideal High-Pass Filter Mask')
plt.subplot(222)
#zoomed in version
plt.imshow(high_pass_filter[center_row-100:center_row+100, center_col-100:center_col+
plt.title('Ideal High-Pass Filter Mask, zoomed in')

plt.subplot(223)
plt.imshow(low_pass_filter, cmap="gray")
plt.title('Ideal High-Pass Filter Mask')

plt.subplot(224)
plt.imshow(low_pass_filter[center_row-100:center_row+100, center_col-100:center_col+1
plt.title('Ideal Low-Pass Filter Mask, zoomed in')
plt.show()
```









In [ ]:
```
I1_HP = DFT_I1 * high_pass_filter
I2_HP = DFT_I2 * high_pass_filter
I1_LP = DFT_I1 * low_pass_filter
I2_LP = DFT_I2 * low_pass_filter

# Shift the filtered images back to the original position
I1_HP = np.fft.ifft2(np.fft.ifftshift(I1_HP)).real
I2_HP = np.fft.ifft2(np.fft.ifftshift(I2_HP)).real
I1_LP = np.fft.ifft2(np.fft.ifftshift(I1_LP)).real
I2_LP = np.fft.ifft2(np.fft.ifftshift(I2_LP)).real

#plotting
plt.figure(figsize=(30,20))
plt.subplot(221)
plt.imshow(I1_HP,cmap="gray")
plt.title('Frequency domain, High-pass filtered (ideal filter), I1')
plt.subplot(222),
plt.imshow(I2_HP,cmap="gray")
plt.title('Frequency domain, High-pass filtered (ideal filter), I2')
plt.subplot(223)
plt.imshow(I1_LP,cmap="gray")
plt.title('Frequency domain, Low-pass filtered (ideal filter), I1')
plt.subplot(224),
plt.imshow(I2_LP,cmap="gray")
plt.title('Frequency domain, Low-pass filtered (ideal filter), I2')
plt.show()
```