LABexc5-ELE510-2021

October 10, 2021

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

Purpose: *To learn about the Fourier Transform and its use for computation of the image Frequency Spectrum. The emphasis is on the fundamentals of digital images.*

The theory for this exercise can be found in chapter 6 of the text book [1] and in appendix A.1.3 in the compendium [2]. See also the following documentations for help: - OpenCV - numpy - matplotlib - scipy

IMPORTANT: Read the text carefully before starting the work. In many cases it is necessary to do some preparations before you start the work on the computer. Read necessary theory and answer the theoretical part first. The theoretical and experimental part should be solved individually. The notebook must be approved by the lecturer or his assistant.

Approval:

The current notebook should be submitted on CANVAS as a single pdf file.

```
To export the notebook in a pdf format, goes to File -> Download as -> PDF via LaTeX (.pdf).
```

Note regarding the notebook: The theoretical questions can be answered directly on the notebook using a *Markdown* cell and LaTex commands (if relevant). In alternative, you can attach a scan (or an image) of the answer directly in the cell.

Possible ways to insert an image in the markdown cell:

```
![image name]("image_path")
```

```
<img src="image_path" alt="Alt text" title="Title text" />
```

Under you will find parts of the solution that is already programmed.

```
<p>You have to fill out code everywhere it is indicated with `...`</p>
<p>The code section under `######## a)` is answering subproblem a) etc.</p>
```

1.1 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:

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

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?

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

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

$$g(x) = \begin{cases} 1 & \text{for} \quad x = 0, 1, 2 \text{ and } 14, 15. \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

Section a)

Replacing in the equation $k = 0$

$$G(0) = \sum_{x=0}^{w-1} g(x) e^{-j2\pi \frac{0x}{w}} \tag{4}$$

Notice the exponent of e is equal to 0

$$-j2\pi \frac{0x}{w} = 0 \implies e^{-j2\pi \frac{0x}{w}} = 1 \tag{5}$$

This means we can omit $e^{-j2\pi \frac{kx}{w}}$ and the DC-component would be just the sum of $g(x)$

$$G(0) = \sum_{x=0}^{w-1} g(x) \tag{6}$$

Section b)

$$G(k) = G(k + l \cdot w) \implies \sum_{x=0}^{w-1} g(x) e^{-j2\pi \frac{kx}{w}} = \sum_{x=0}^{w-1} g(x) e^{-j2\pi \frac{(k+lw)x}{w}} \tag{7}$$

Simplifying e's exponent…

$$-j2\pi \frac{k + lwx}{w} = -j2\pi \left(\frac{kx}{w} + \frac{lwx}{w}\right) \tag{8}$$

We can eliminate w in the second fraction and we will end up with the following exponent

$$-j2\pi\frac{kx}{w} + -j2\pi lx \tag{9}$$

We can split the addition of the exponent into a product of the same base

$$e^{-j2\pi\frac{kx}{w}+-j2\pi lx} = e^{-j2\pi\frac{kx}{w}} \cdot e^{-j2\pi lx} \tag{10}$$

l and x are integers, hence we can say $e^{-j2\pi lx} = 1$ (See euler's formula)

$$G(k+l\cdot w) = \sum_{x=0}^{w-1} g(x)e^{-j2\pi\frac{kx}{w}} \implies G(k+l\cdot w) = G(k) \tag{11}$$

Section c)

We can split the function into two summatories for the non-zero samples (notice we ignore $g(x)$ because it's equal to 1):

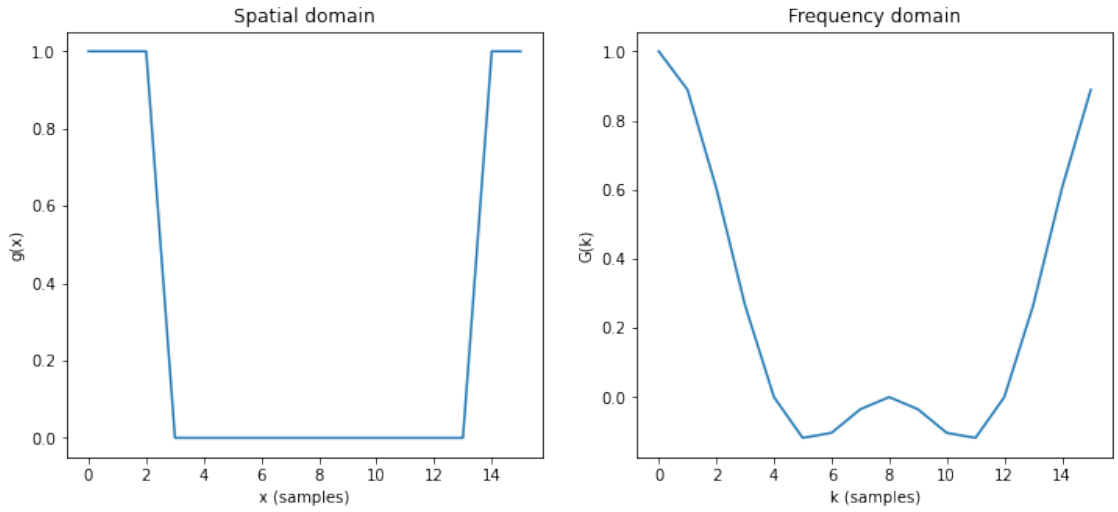
$$G(k) = \sum_{x=0}^{2} e^{-j2\pi\frac{kx}{16}} + \sum_{x=14}^{15} e^{-j2\pi\frac{kx}{16}} \tag{12}$$

Sketch both (using Python) $g(x)$ and $G(k)$ in the index range $-8, -7, \cdots, -1, 0, 1, 2, \cdots, 7$ (note the periodic property of both functions).

[1]:
```
# Sketch it using python.

# Both functions are discrete and periodic with period N=16. Note that the DFT␣
 ↪of the discrete box
# function approximates a truncated "sinc" function. The continuous Fourier␣
 ↪transform of the
# continuous box function is a "sinc" function with infinite duration.

import matplotlib.pyplot as plt
import math
import numpy as np

w = 16
j = complex(0,1)

# Create values for g(x)
x = list(range(w))
g_x = [0 if x_i > 2 and x_i < 14 else 1 for x_i in x]

# Create values for G(k)
# x is the same for G(k)
g_k = []
for k_i in x:
    g_ki = 0
```

```
    for x_i in range(2):
        g_ki += math.e**(-j*2*math.pi*k_i*x_i/16)
    for x_i in range(14,w):
        g_ki += math.e**(-j*2*math.pi*k_i*x_i/16)
    g_k.append(g_ki)

# Turn g_k into a numpy array to perform elementwise operation
g_k = np.array(g_k)
# Normalize the values
g_k = g_k/max(g_k).real

# Plot results
plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
plt.title('Spatial domain')
plt.xlabel('x (samples)')
plt.ylabel('g(x)')
plt.plot(x, g_x)

plt.subplot(1,2,2)
plt.title('Frequency domain')
plt.xlabel('k (samples)')
plt.ylabel('G(k)')
plt.plot(x, g_k)
plt.show()
```

```
C:\Users\aeste\anaconda3\lib\site-packages\numpy\core\_asarray.py:102:
ComplexWarning: Casting complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```



1.2 Problem 2

Introduction An image can be considered as a 2-dimensional (2D) signal. Sampling is in the domain of spatial coordinates (x_w, y_w). The continuous intensity function, $g(x_w, y_w)$, is sampled such that the image representation is a matrix, $g(x, y)$, where the indexes, x and y is connected to the coordinates by $x_w = x\,\Delta x_w$ and $y_w = y\,\Delta y_w$. The distance between each pixel is Δx_w horizontally and Δy_w vertically. We use $g(x, y)$, with indexes $y \in \{1, 2, \cdots Y\}$ and $x \in \{1, 2, \cdots X\}$. The image is rectangular, represented by a matrix of size $Y \times X$.

We can represent the image in the frequency domain by taking the discrete Fourier transform. In this problem we will use the power spectrum defined by

$$P(i, j) = |G(i, j)|^2 \tag{13}$$

where $G(i, j)$ is the 2D DFT of the signal (image) $g(i, j)$. Most images has a large mean value ("DC-component") that gives a dominating peak in the frequency origin $(0, 0)$. A typical image is also dominated by low frequency components and the power of high frequencies is low. The DFT assumes periodic signals, i.e. when an image is repeated periodically in all directions we usually get an edge at the border of the image. This will give a response in the spectrum as a horizontal and vertical line through the origin. To avoid this it is common to use a bell shaped window function that reduces the signal to zero at the borders. This method for spectral estimation is called *windowed periodogram*. The function **impowsp2** that you will find here implements this method.

We want to visualize the power spectrum by an image, i.e. we transform the image from the spatial domain to the frequency domain. To enhance the high frequency components we use the following point operation

$$B(i, j) = \log(1 + P(i, j)) \tag{14}$$

Before the point operation the DC-component must be removed [*] and the power spectrum should be centered (using **np.fftshift**), such that the frequency $(0, 0)$ is in the center of the power spectrum image. We use **plt.imshow** to display the power spectrum image and choose normalized frequencies [+] for the axis.

[*] *This can be done by setting the center pixel in the power spectrum to zero. It can also be accomplished by subtracting the mean from the input image such that we find the DFT from a zero mean signal.*

[+] *The Fourier Transform of a digital signal is computed by the DFT (Discrete Fourier Transform) which is periodic in the frequency domain. One period corresponds to* 2π *(*$[-\pi, \pi]$*) for the angular frequency. We normalize this such that the frequency is in the interval* $[-0.5, 0.5]$.

[2]:
```
import numpy as np
import cv2
from scipy import signal
```

[3]:
```
def impowsp2(Im, a, b, wtype, c, Fig):
    """
```

```
   The function finds the power spectrum of the image, Im.
   The power is estimated by using windowed periodogram, where the window type␣
↪is:
   wtype = 1    : Kaiser,  b = [1,10]   beta
   wtype = 2    : Chebyshev,  b = [20,100] sidelobe att. in dB
   wtype = n > 2    : Rectangular,  b  don't care, i.e. no window

   a : amplification constant for visualization of high frequency content (e.g.
↪ a = 10).
   The DC-component is removed in all cases. For visualization the low␣
↪frequency components in a region centered at
   zero can be removed. The size of the region is (2c-1)x(2c-1).

   Fig = [Figtype, Figure number]
       Figtype = 0 : no Figure
       Figtype = 1 : Only Frequency Spectrum
       Figtype = 2 : Both input Image and Frequency Spectrum

   Output is the normalized estimated spectrum, with no DC-component.

   """

   M,N = Im.shape
   L = max(M,N)
   if L<256: K = 256
   elif L <= 512: K = 512
   else: K = L

   A = Im - np.mean(Im[:])*np.ones_like(Im);
   # A is a zero mean signal, that means that the power for zero frequency,␣
↪the DC component, is zero.

   if wtype == 1: # Kaiser window
       w1 = np.kaiser(M,b)
       w2 = np.kaiser(N,b)
   elif wtype == 2:# Chebyshev window
       w1 = signal.chebwin(M,b)
       w2 = signal.chebwin(N,b)
   else:
       w1 = np.ones(shape=(M,1))
       w2 = np.ones(shape=(M,1))

   W = w1 * np.transpose(w2)
   B = A * W
   F = np.fft.fft2(B, s=[K,K])
   F = np.fft.fftshift(F)
   P = F * np.conj(F)
```

```
    Pmax = np.max(P)
    P = P/Pmax
    Pout = P

    # Visualization
    if Fig == 0: print("No figure") # No figure
    elif Fig == 1: # Only Frequency Spectrum
        m0 = (K/2)+1-c
        k = 2*c-1
        #P[m0+(1:k),m0+(1:k)] = np.zeros(k)
        B = np.log(np.ones_like(P) + a * Pmax * P)

        plt.figure(figsize=(10,10))
        plt.imshow(B, extent=[-0.5, 0.5, -0.5, 0.5])
        plt.title('Power spectrum')
        plt.axis([-0.5,0.5,-0.5,0.5])
        plt.show()
    elif Fig == 2: # Both input Image and Frequency Spectrum
        m0 = (K/2)+1-c
        k = 2*c-1
        #P[m0+(1:k),m0+(1:k)] = np.zeros(k)
        C = np.log(np.ones_like(P) + a * Pmax * P)

        plt.figure(figsize=(10,10))
        plt.subplot(121)
        plt.imshow(Im, cmap='gray', vmin=0, vmax=255)
        plt.title("Image")
        plt.subplot(122)
        plt.imshow(np.real(C), extent=[-0.5, 0.5, -0.5, 0.5])
        plt.title('Power spectrum')
        plt.axis([-0.5,0.5,-0.5,0.5])
        plt.show()
    else: print("'No such Figure, Fig == 0, 1 or 2'")
```

1.2.1 Problem

A function (**impowsp2**) that computes and displays the power spectrum of an image is given here. Additional information of the function can be found at the end of the exercise.

In the following experiments use the parameters: $a = 10$, $b = 3$, $wtype = 1$ and $c = 0$ for the function **impowsp2**. We want to study the frequency spectrum for a set of different images.

The first image is a test image, **qcirchirp.bmp**, containing most frequencies. It is a circular pattern where the radial intensity function is a so called chirp signal with frequencies increasing from zero to the maximum allowed frequency for the given number of samples, avoiding aliasing. The given test pattern is one quadrant from a circular test image.

a) Use **impowsp2**, as described above, and find the power spectrum for the image **qcirchirp.bmp**.

b) Explain and describe the result. Is it as expected? Why is the power zero in the first and third quadrant?

c) Why is the power highest for lower spatial frequencies?

The second image is a very popular test image, used for decades, the Camera Man, **cameraman.jpg**. This image is an example where the edges are very sharp and the power spectrum indicates **aliasing**.

d) Find the power spectrum for the image **cameraman.jpg**.

e) Explain which parts of the image gives rise to the lines in the spectrum.

f) How can you explain, from the power spectrum, why we have aliasing.

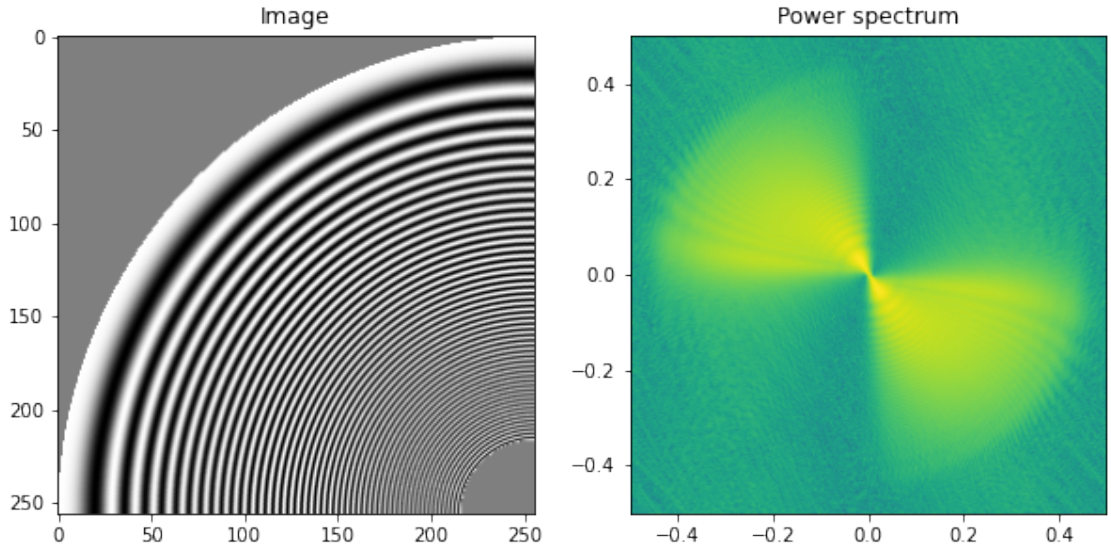
Hint: Study the lines in the spectrum that reaches the border and find where these lines continues, when we know that the spectrum repeats itself periodically.

Finally we want to study the power spectrum for some images, representing different image textures.

g) Find the power spectrum for: skin1, soap-bubbles, tex_ori, tex_scl and textil1.

h) Explain and describe the results.

[4]:
```
# Section a)
Im = cv2.imread('./images/qcirchirp.bmp', cv2.IMREAD_GRAYSCALE)
impowsp2(Im, 10, 3, 1, 0, 2)
```



Section b)

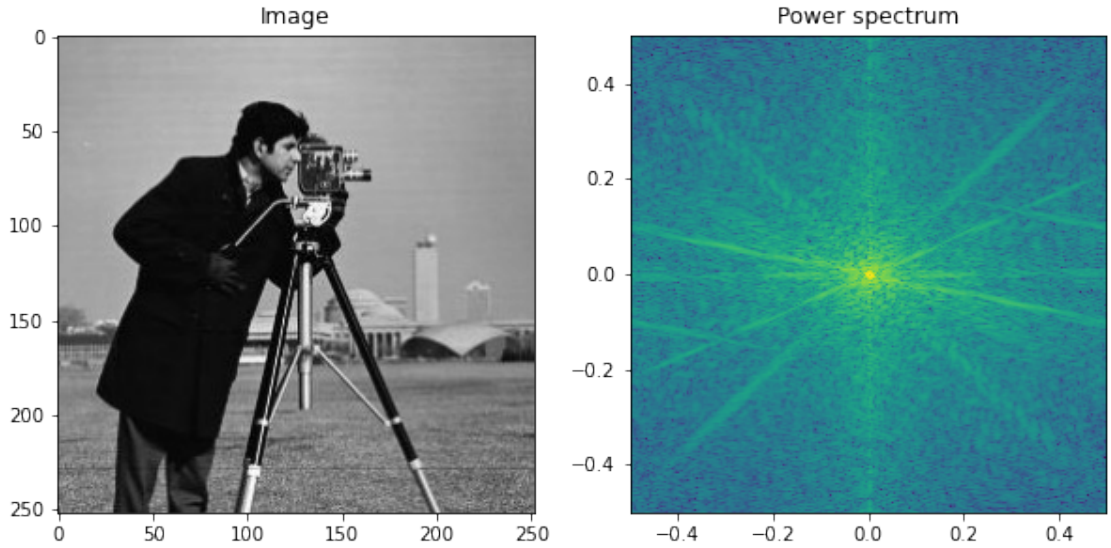
Notice the direction of the lines in the original image are concentric and similar to the second quadrant of a circle, that's why the main frequencies on the power spectrum are in the second and

fourth quadrant and the power of first and third quadrant is zero. (In a circle, the direction of the lines are the same for cross quadrants)

Section c)

The frequencies that are near the DC-value are the lowest ones, this frequencies represents low-details of the image such as the background, zones without edges, etc... That's why the frequencies has less and less power as we get further from the DC-value.

[5]:
```
# Section d)
Im = cv2.imread('./images/cameraman.jpg', cv2.IMREAD_GRAYSCALE)
impowsp2(Im, 10, 3, 1, 0, 2)
```



Section e)

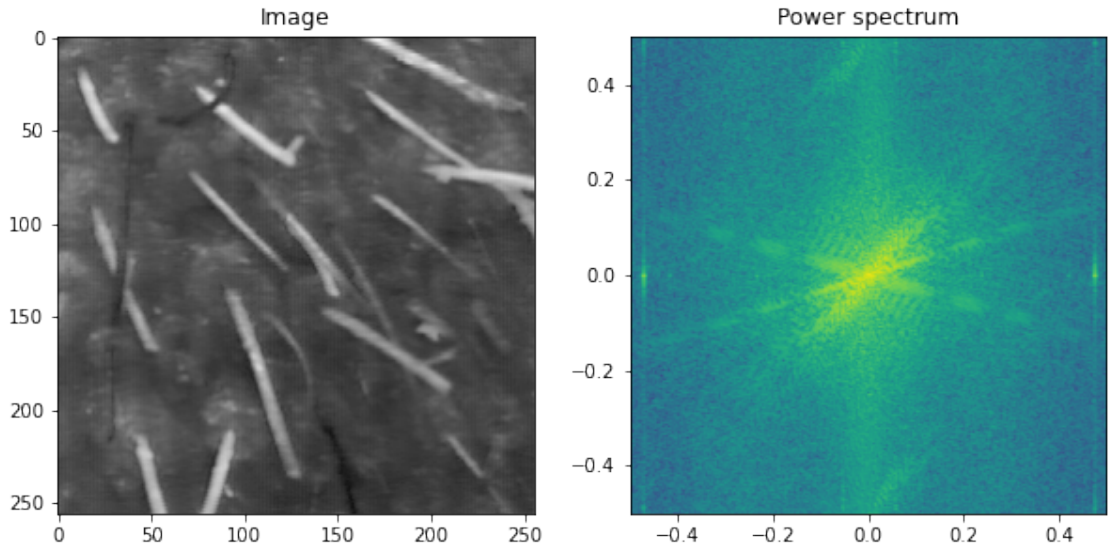
We can see a few lines in the power spectrum that goes from the center (DC-value) to the edges of the spectrum, this lines are the frequency representation of the edges contained in the image.

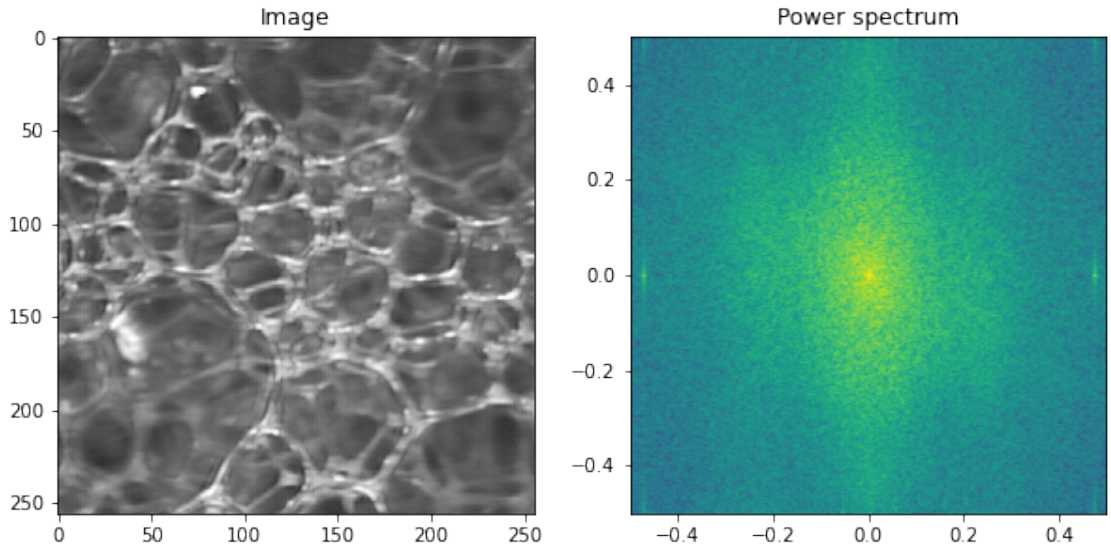
Section f)

Notice if we follow some lines along the whole spectrum we can see that some of them are being overlapped, as we know our image is a periodic signal, we can figure out this lines goes from one side of the image and appears from the opposite side. This is aliasing and it makes us impossible to differ some frequencies from others.

[6]:
```
# Section g)
Im1 = cv2.imread('./images/skin1.png', cv2.IMREAD_GRAYSCALE)
impowsp2(Im1, 10, 3, 1, 0, 2)

Im2 = cv2.imread('./images/soapbubbles.png', cv2.IMREAD_GRAYSCALE)
impowsp2(Im2, 10, 3, 1, 0, 2)
```

```
Im3 = cv2.imread('./images/tex_ori.png', cv2.IMREAD_GRAYSCALE)
impowsp2(Im3, 10, 3, 1, 0, 2)

Im4 = cv2.imread('./images/tex_scl.png', cv2.IMREAD_GRAYSCALE)
impowsp2(Im4, 10, 3, 1, 0, 2)

Im5 = cv2.imread('./images/textil1.png', cv2.IMREAD_GRAYSCALE)
impowsp2(Im5, 10, 3, 1, 0, 2)
```





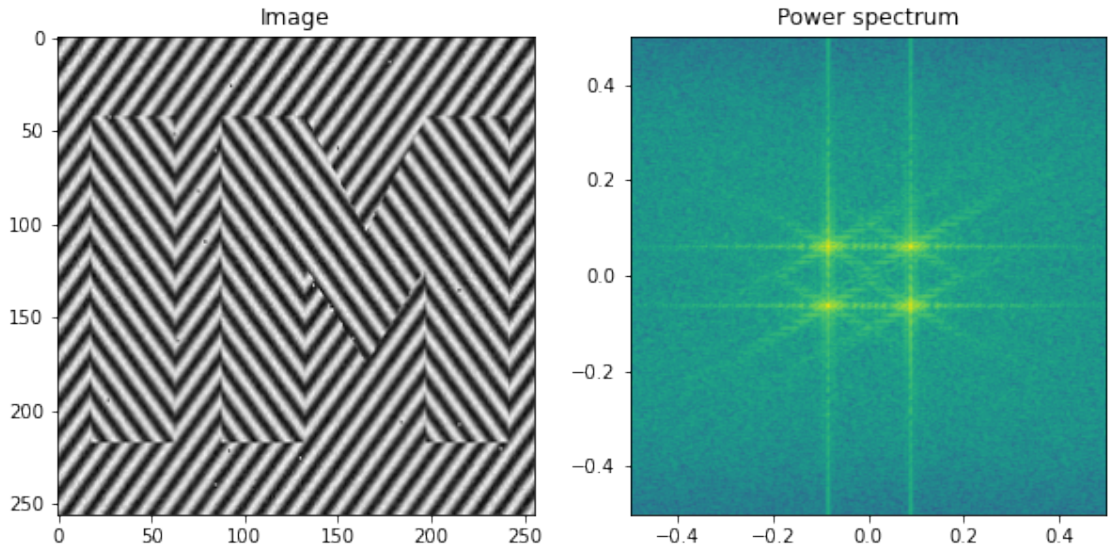


Image
Power spectrum

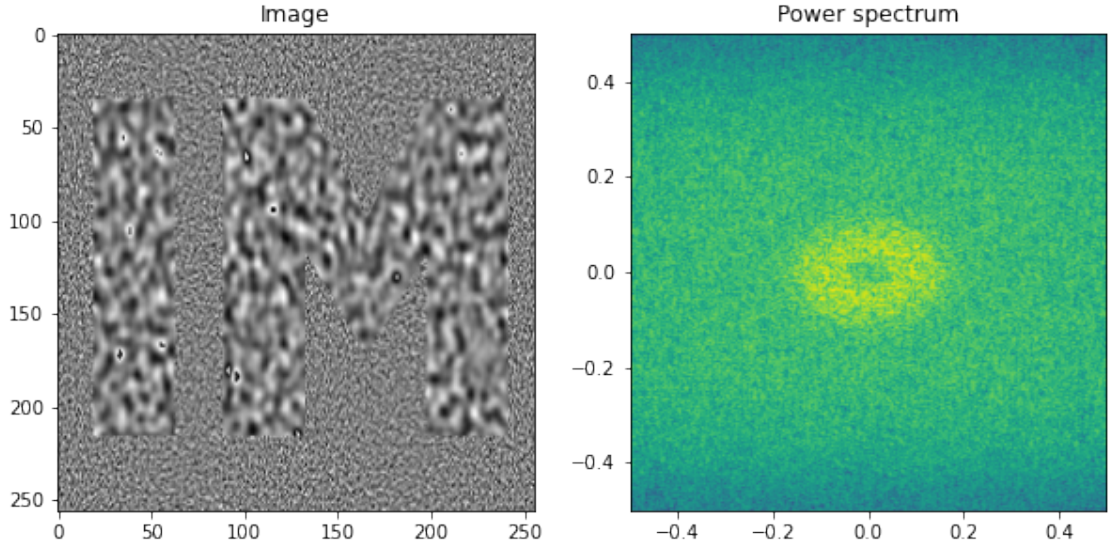
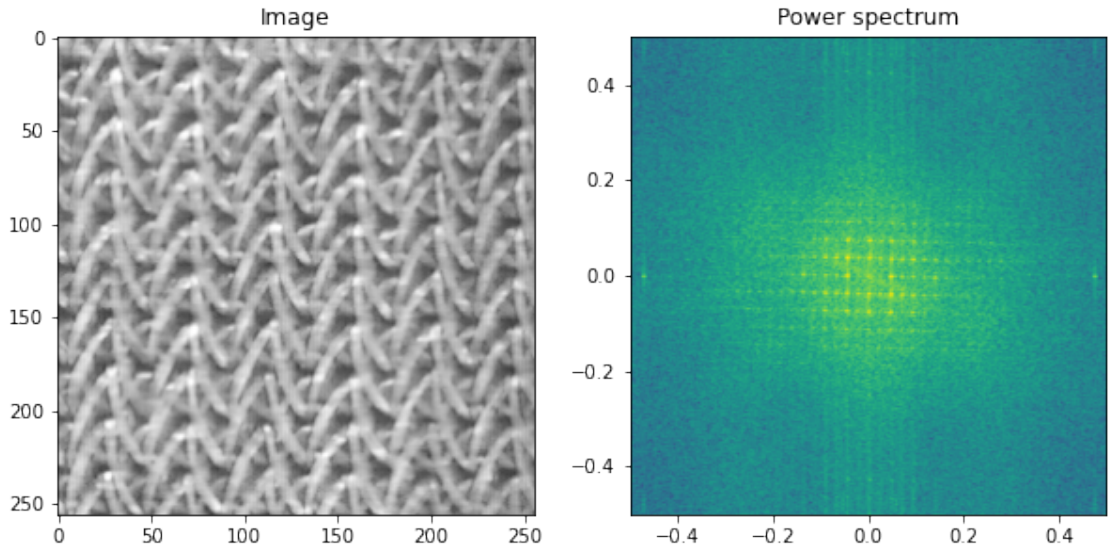


Image
Power spectrum



Section h)

Looking at this several power spectrums we may realise they have a lot of information about edges, the frequency domain of an image is a very nice place to do filtering based on frequency. We notice that first and third spectrums have very well defined lines, because their original images have also noticeable edges, while the rest doesn't seem to have such prominent edges.

1.3 Problem 3

We test the filters from exercise 4 (shown below), we use the test image **qcirchirp.bmp**. Filtering is done by using the function `cv2.filter2D()` (Documentation).

The input is the image matrix, the filter mask (in X and Y directions) and a flag parameter specifying how the boundary should be treated. Here's an example of how to call the function:

```
Imoutput = cv2.filter2D(Iminput,ddepth=-1,kernel=(filtermask*filtermask.T),borderType=borderTy
```

To study the result in the frequency domain we use the function **impowsp2** defined before.

Filter masks:

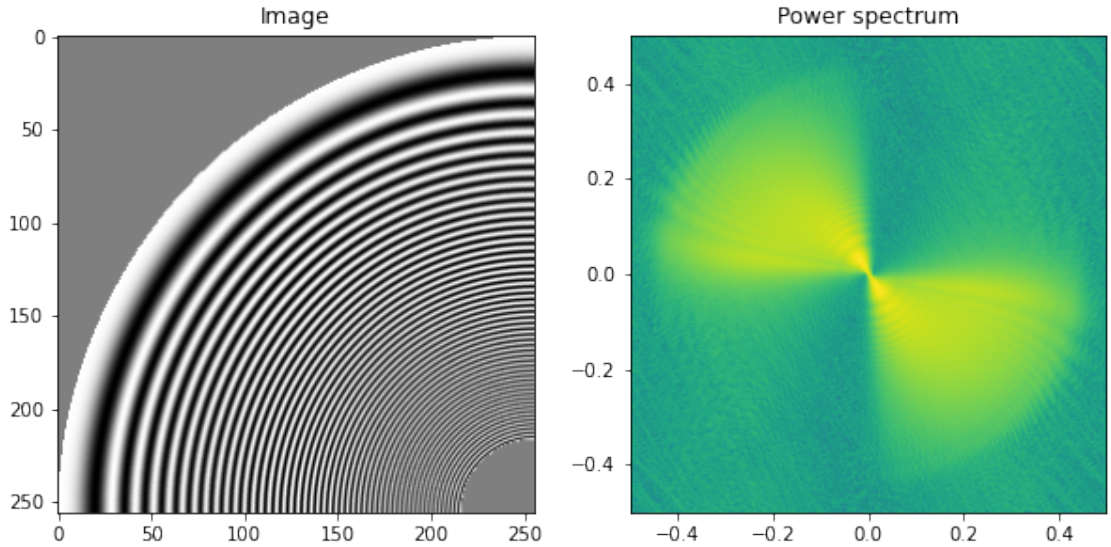
```
h1 = cv2.getGaussianKernel(9,1)
h15 = cv2.getGaussianKernel(11,1.5)
h2 = cv2.getGaussianKernel(15,2)
```

a) Use the filter masks `h1`, `h15` and `h2` with `cv2.BORDER_DEFAULT` boundary. Filter **qcirchirp.bmp** using `cv2.filter2D(...)` and find the power spectrums.

b) Which of the three filters do you think is most appropriate if the image is going to be down-sampled by a factor 2?

[7]:
```
# Read image
Im = cv2.imread('./images/qcirchirp.bmp', cv2.IMREAD_GRAYSCALE)
```

```
# Define the kernels
h1 = cv2.getGaussianKernel(9,1)
h15 = cv2.getGaussianKernel(11,1.5)
h2 = cv2.getGaussianKernel(15,2)

# Filter image
Im_h1 = cv2.filter2D(Im, -1, kernel=(h1*h1.T), borderType=cv2.BORDER_DEFAULT)
Im_h15 = cv2.filter2D(Im, -1, kernel=(h15*h15.T), borderType=cv2.BORDER_DEFAULT)
Im_h2 = cv2.filter2D(Im, -1, kernel=(h2*h2.T), borderType=cv2.BORDER_DEFAULT)

# Find power spectrum of the filtered images
impowsp2(Im, 10, 3, 1, 0, 2)
impowsp2(Im_h1, 10, 3, 1, 0, 2)
impowsp2(Im_h15, 10, 3, 1, 0, 2)
impowsp2(Im_h2, 10, 3, 1, 0, 2)
```



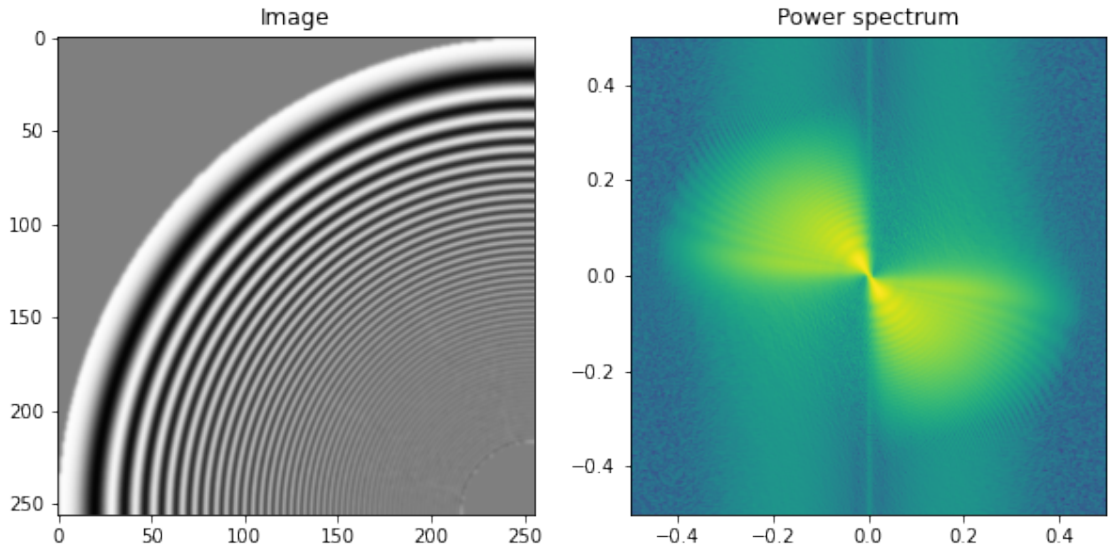


Image
Power spectrum
0
50
100
150
200
250
0.4
0.2
0.0
−0.2
−0.4

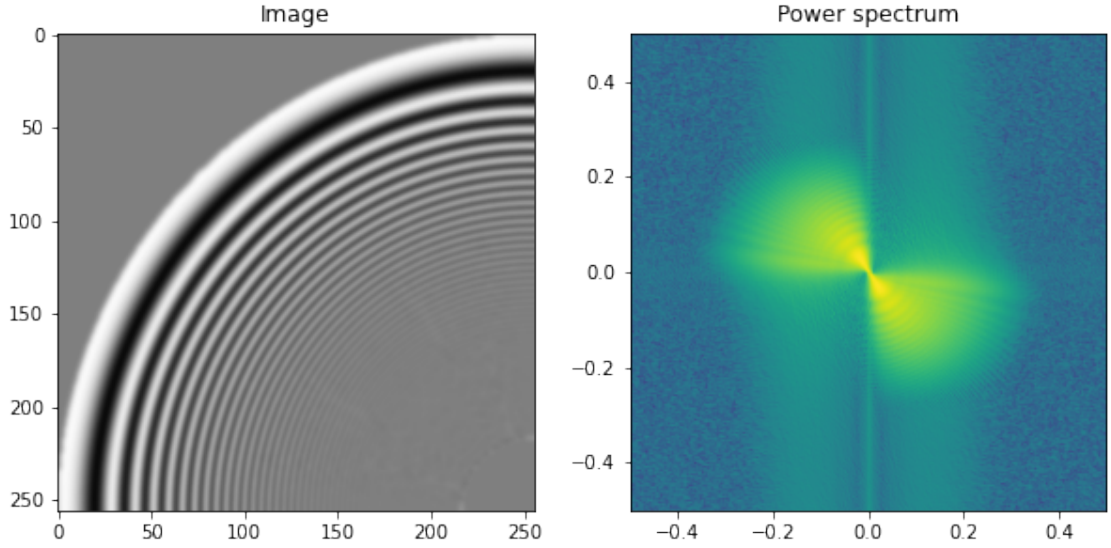
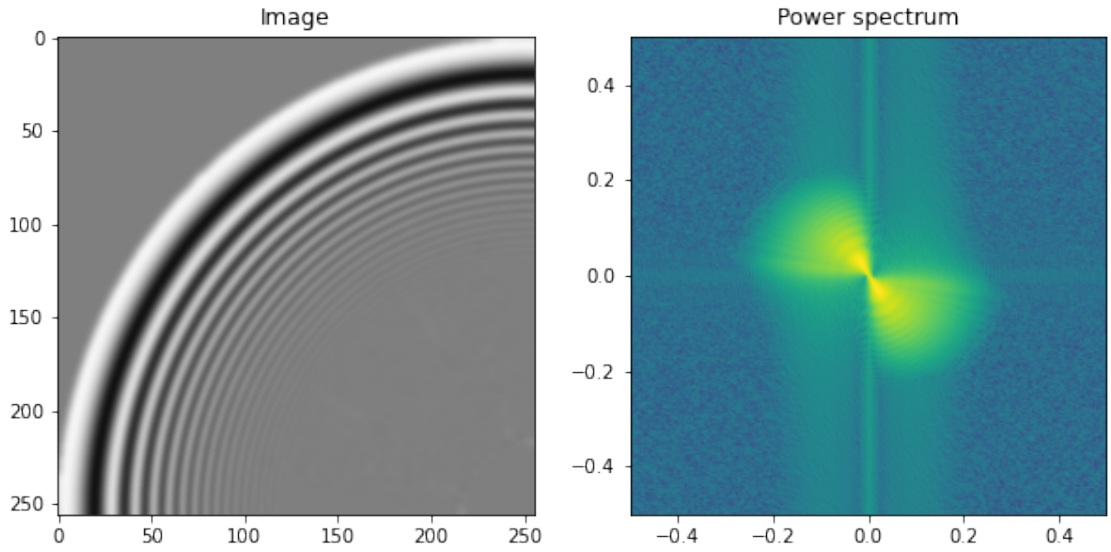


Image
Power spectrum
0
50
100
150
200
250
0.4
0.2
0.0
−0.2
−0.4



Section b)

The most appropiate one would be the second, $\sigma = 1.5$. It satisfies the equal contribution property (each pixel in image contributes equal amount to the downsampled version)

1.4 Problem 4

Create a python script to find the 2D discrete Fourier transform (DFT) of an image (use `cameraman.jpg`).

a) From the DFT, calculate and plot: **the magnitude and the phase**.

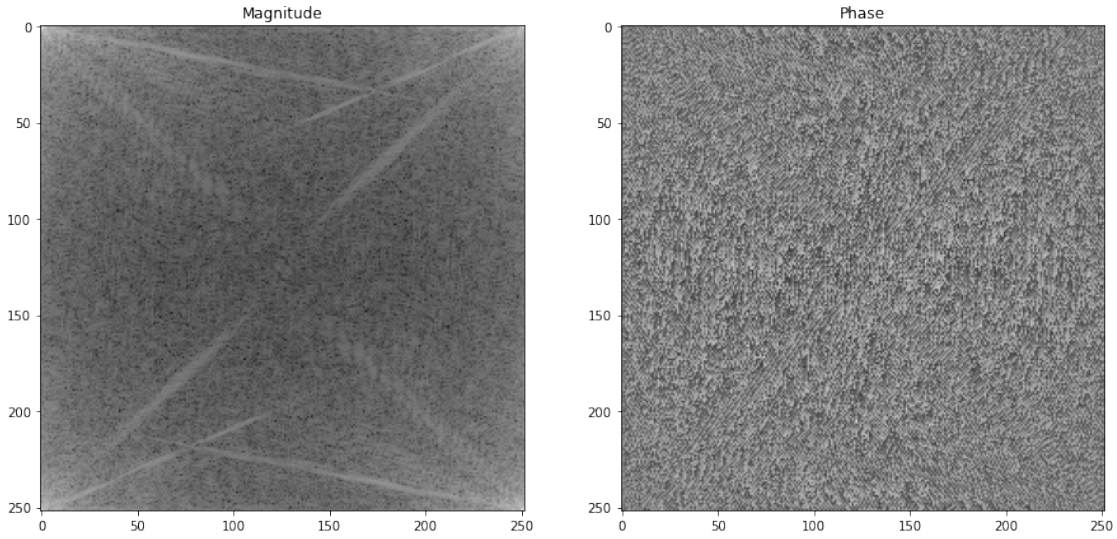
b) Shift the zero-frequency component to the center of the spectrum and plot the result.

c) Where is the DC component for the points **a)** and **b)**?

[8]:
```
# Define complex number
j = complex(0, 1)
pi = math.pi

# Read image
Im = cv2.imread('./images/cameraman.jpg', cv2.IMREAD_GRAYSCALE)

# Compute the 2-Dimensional discrete fourier transform
dft = np.fft.fft2(Im)
# Change to the log domain so we can see the high frequencies
dft = np.log(1 + dft)
```

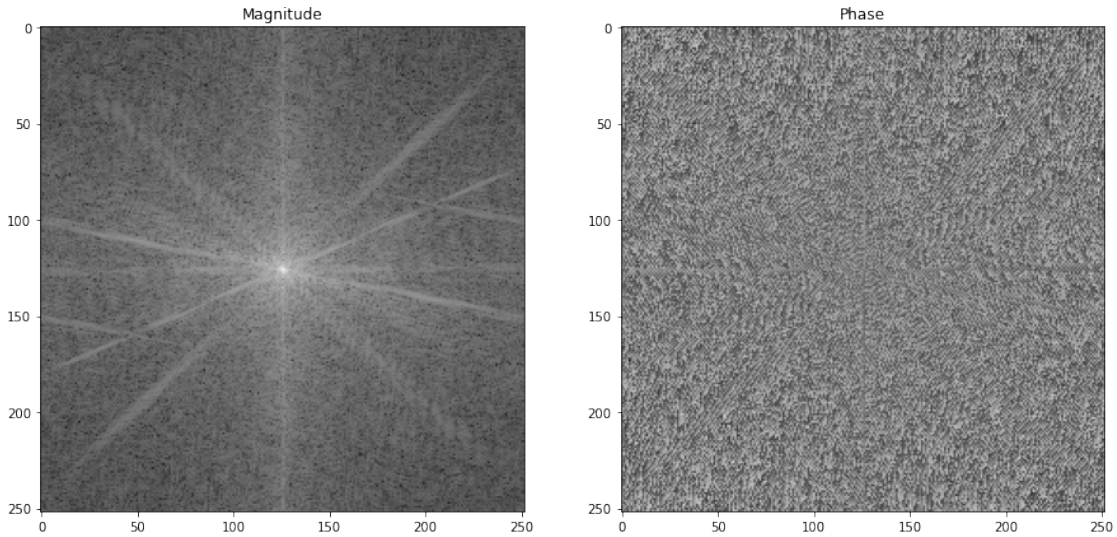
[9]:
```
# Section a)
magnitude = np.sqrt(dft.imag**2 + dft.real**2)
phase = np.arctan(dft.imag/dft.real)
```

```
# Display results
plt.figure(figsize=(15,15))

plt.subplot(1,2,1)
plt.title('Magnitude')
plt.imshow(magnitude, cmap='gray')

plt.subplot(1,2,2)
plt.title('Phase')
plt.imshow(phase, cmap='gray')

plt.show()
```



[10]:
```
# Section b)
# Shift the DC-Value to the center of the image
dft = np.fft.fftshift(dft)

# Compute magnitude and phase again
magnitude = np.sqrt(dft.imag**2 + dft.real**2)
phase = np.arctan(dft.imag/dft.real)

# Display results
plt.figure(figsize=(15,15))

plt.subplot(1,2,1)
plt.title('Magnitude')
plt.imshow(magnitude, cmap='gray')
```

```
plt.subplot(1,2,2)
plt.title('Phase')
plt.imshow(phase, cmap='gray')

plt.show()
```



Section c)

In *section a)*, we should notice the dc-value is splitted between the four corners of the spectrum, while in *section b)* it is centered on the spectrum. It is normally more useful and intuitive to have the information about the dc-value in the center of the spectrum.

1.4.1 Delivery (dead line) on CANVAS: 10.10.2021 at 23:59

1.5 Contact

1.5.1 Course teacher

Professor Kjersti Engan, room E-431,

E-mail: kjersti.engan@uis.no

1.5.2 Teaching assistant

Tomasetti Luca, room E-401

E-mail: luca.tomasetti@uis.no

1.6 References

[1] S. Birchfeld, Image Processing and Analysis. Cengage Learning, 2016.

[2] I. Austvoll, "Machine/robot vision part I," University of Stavanger, 2018. Compendium, CANVAS.