

# Lab 05

Basic Image Processing Algorithms  
Fall 2022

# Discrete Fourier Transform

- ◉ We sample one period of the Fourier transform in evenly spaced frequencies:

$$X(\omega_1, \omega_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) e^{-j\omega_1 n_1} e^{-j\omega_2 n_2}$$

The size of the image in the spatial domain is  $N_1 \times N_2$

The size of the image in the frequency domain will be the same:  $N_1 \times N_2$

$$X(k_1, k_2) = X(\omega_1, \omega_2) \Big|_{\omega_1 = \frac{2\pi}{N_1} k_1, \omega_2 = \frac{2\pi}{N_2} k_2}$$

$$k_1 = 0, 1, \dots, N_1 - 1$$

$$k_2 = 0, 1, \dots, N_2 - 1$$

$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) e^{-j \frac{2\pi}{N_1} k_1 n_1} e^{-j \frac{2\pi}{N_2} k_2 n_2}$$

Only one period is kept

# Discrete Fourier Transform

---

- ◉ Forward formula: gives the description of the image in the discrete frequency domain

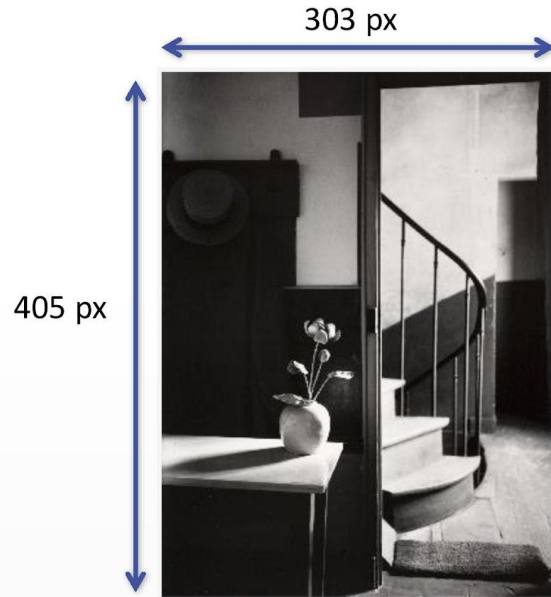
$$X(k_1, k_2) = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x(n_1, n_2) e^{-j\frac{2\pi}{N_1}k_1n_1} e^{-j\frac{2\pi}{N_2}k_2n_2}$$

- ◉ Inverse Fourier transform: maps from the discrete frequency domain back to the discrete spatial domain

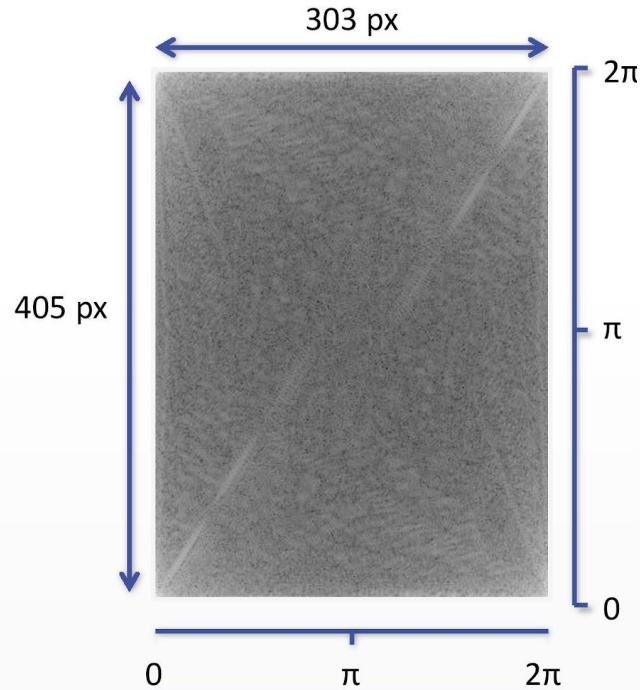
$$x(n_1, n_2) = \frac{1}{N_1 N_2} \sum_{k_1=0}^{N_1-1} \sum_{k_2=0}^{N_2-1} X(k_1, k_2) e^{j\frac{2\pi}{N_1}k_1n_1} e^{j\frac{2\pi}{N_2}k_2n_2}$$

- algorithmically it has the same structure as the forward transform,

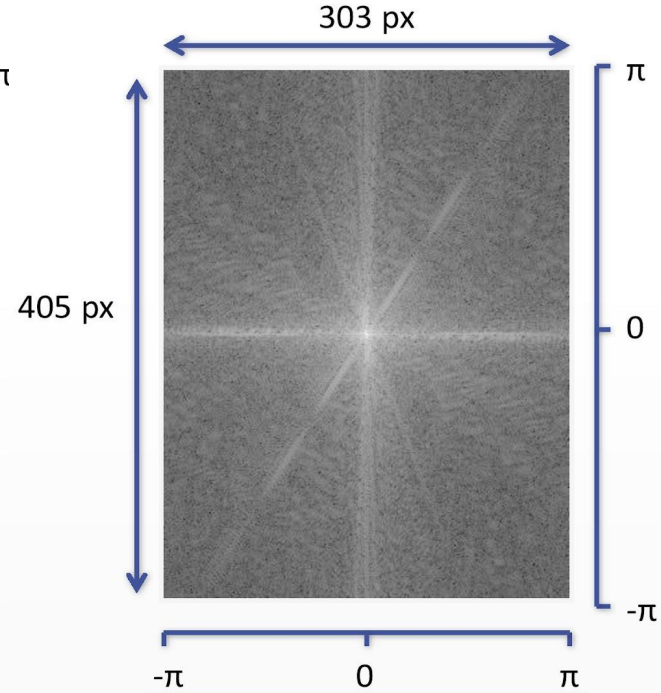
# Discrete Fourier Transform



Original Image\*



Magnitude of the  
Discrete Fourier Transform



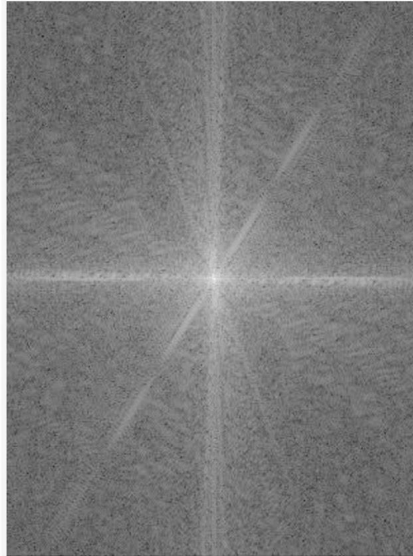
Magnitude of the **centered**  
Discrete Fourier Transform

# Discrete Fourier Transform

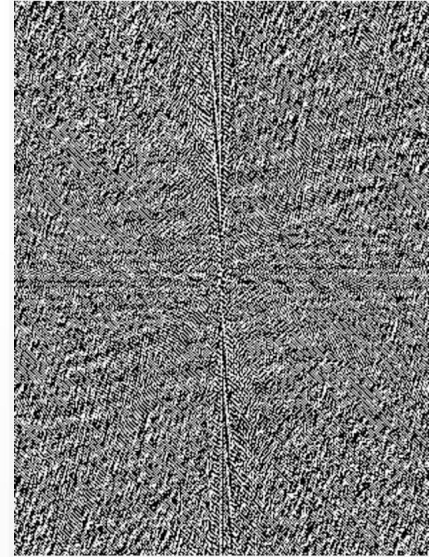
---



Original Image\*



Magnitude of the **DFT**



Phase of the **DFT**

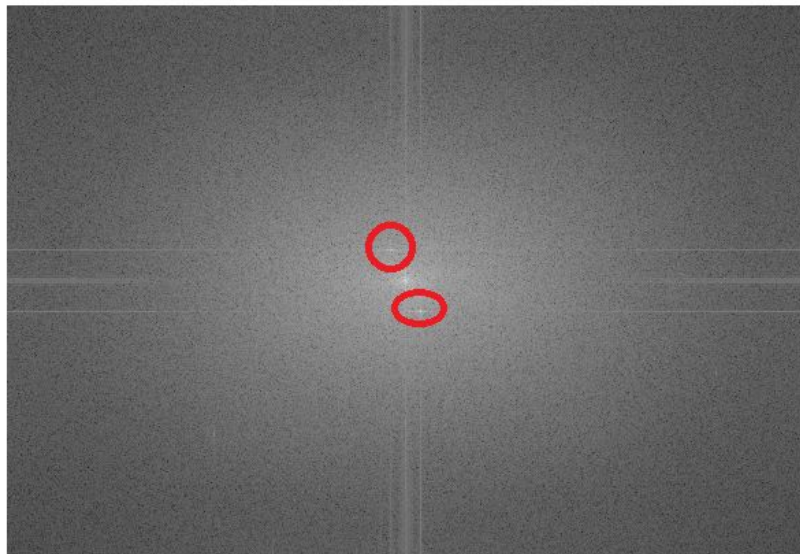


# Example application I. – Noise filtering

original input with noise



original magnitude of DFT



**On the left:** original image (Robert Capa: Lovers Parting Near Nicosia, Sicily) + some sinusoidal noise

**On the right:** magnitude part of the frequency domain

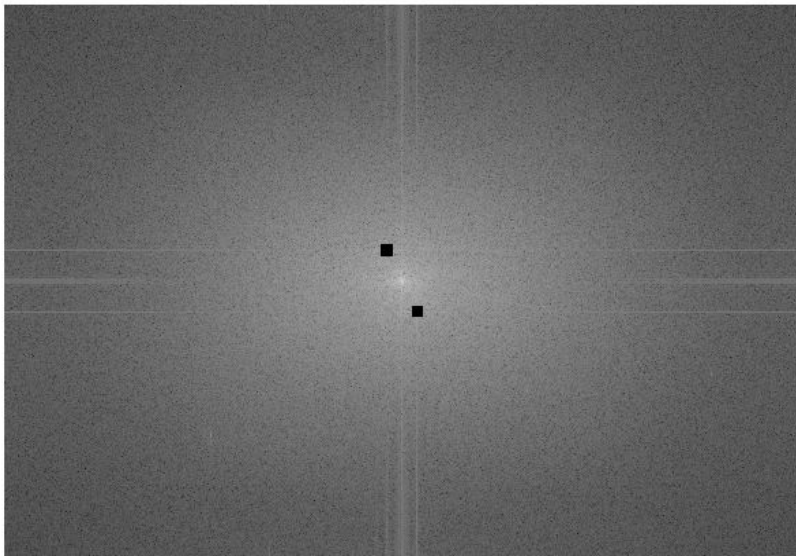
Thanks to the regular form of the noise, we can see it as two concentrated frequency-points on the DFT-image (circled with red).

# Example application I. – Noise filtering

What if we hide these two, intensive-regions from the frequency-domain?

(Hide := decrease their significance, actually I set their value to complex zero,  $0+0i$ .) The noise disappears!

**modified magnitude of DFT**



**filtered output**



# Example application I. – Noise filtering

**part of orig. input**



**part of filt. output**

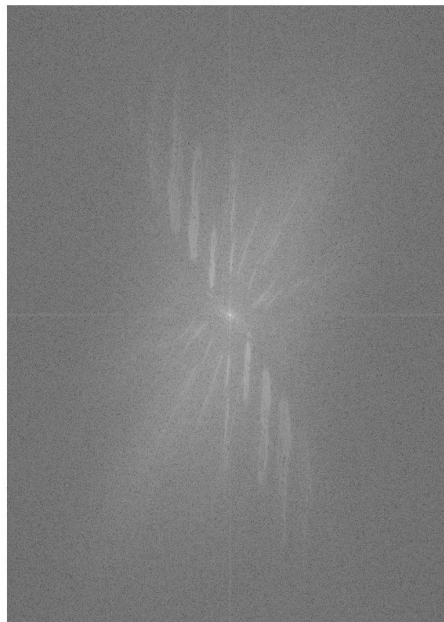




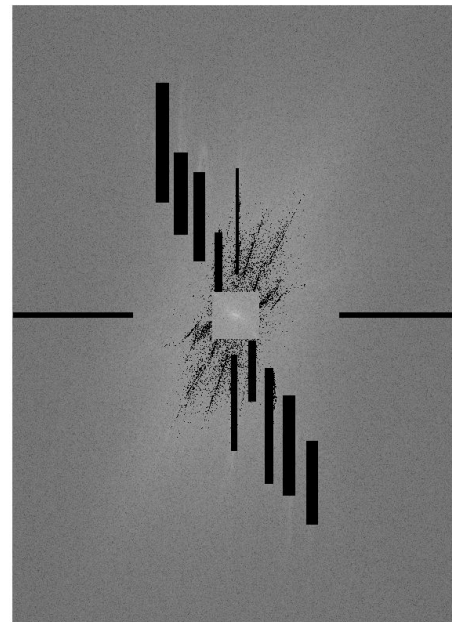
# Example application II. – Noise filtering



original magnitude of DFT



modified magnitude of DFT



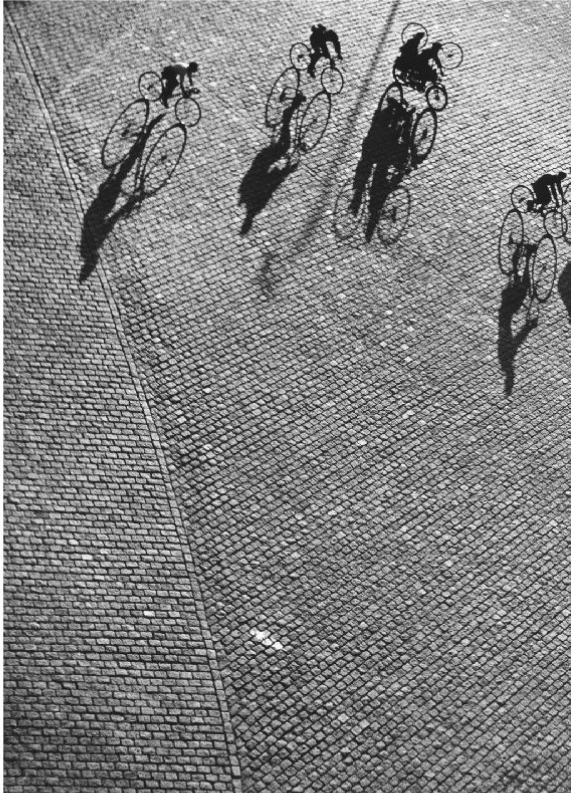
**On the left:** Lucien Herve: Paris Sans Quitter Ma Fenetre (Les Cyclistes)

**On the center:** magnitude part of the frequency domain

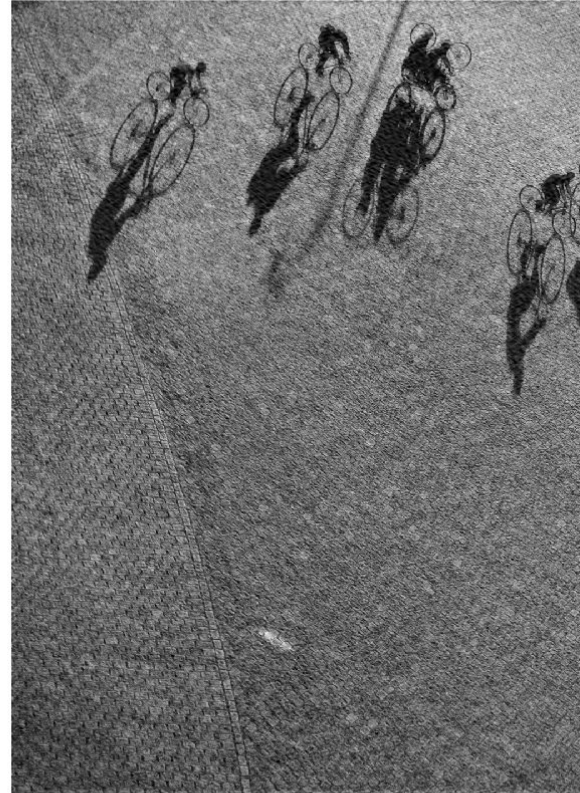
**On the right:** modified magnitude (again, the spec. values are replaced with complex zeros)

# Example application II. – Noise filtering

original input

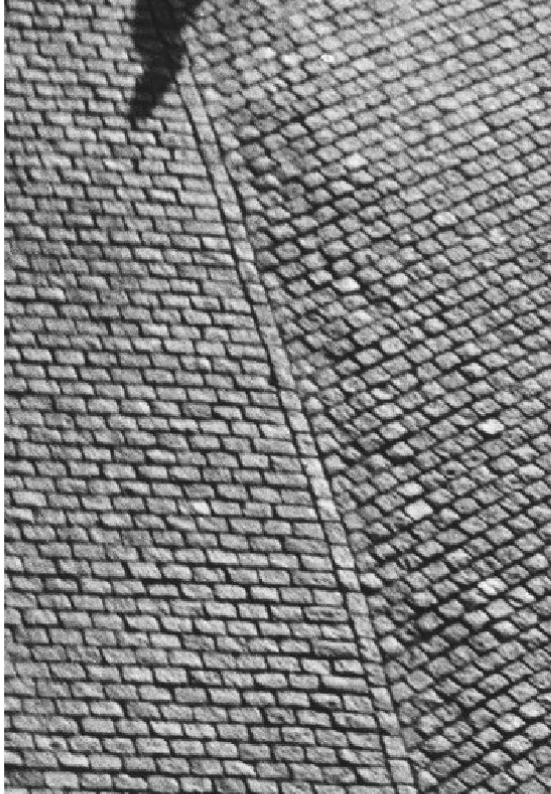


filtered output

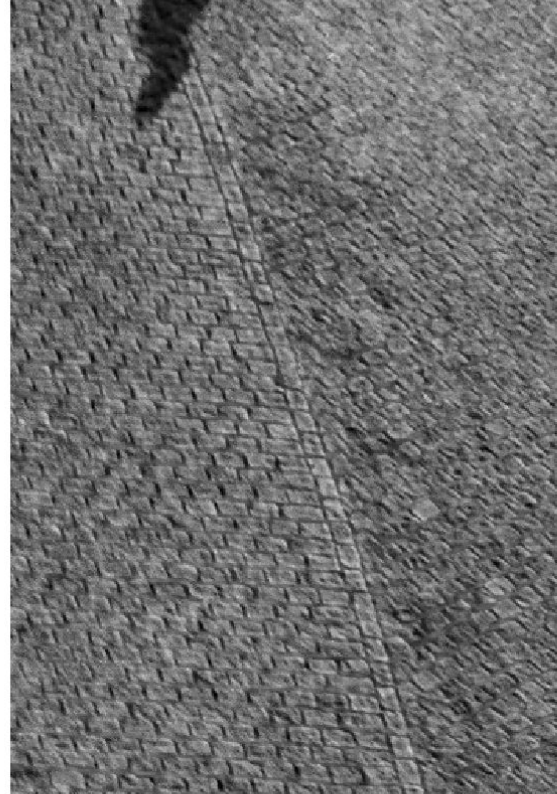


# Example application II. – Noise filtering

part of orig. input



part of filtered output



Now please  
**download the 'Lab 05' code package**  
from the  
**[moodle system](#)**

# Exercise 1

Implement the **function** `my_fourier` in which:

- Create the empty `F` as the complex discrete Fourier space. This should be the same size as the input image (`I`).
- With `k1` and `k2` iterate through the Fourier-space (two (nested) `for` loops).
- Compute `F(k1, k2)` which is denoted by  $X(k_1, k_2)$  on Slide 3. For this you'll need to use `n1` and `n2` to iterate through the input image (another two (nested) `for` loops inside the previous ones  $\rightarrow$  4 nested loops)

You can assume that the input of this function is a 2D double matrix. You should return a 2D complex double matrix. **Remember: Matlab has 1-based indexing.**

Run `script1.m` which will test your implementation. **Diff <  $10^{-8}$  is OK.**

Original image

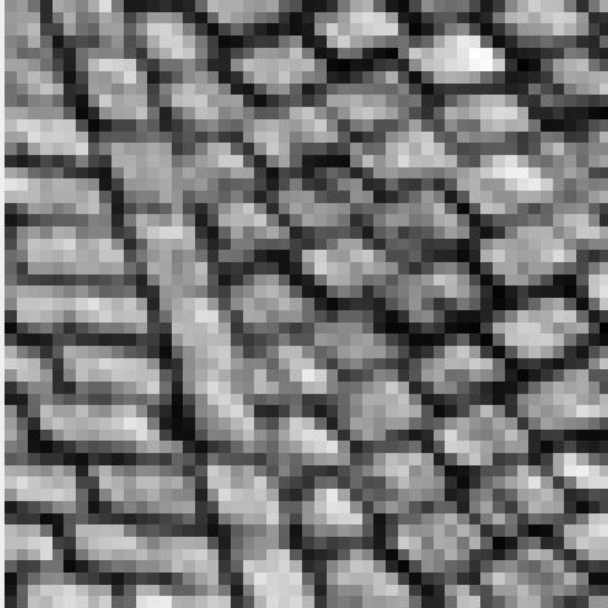
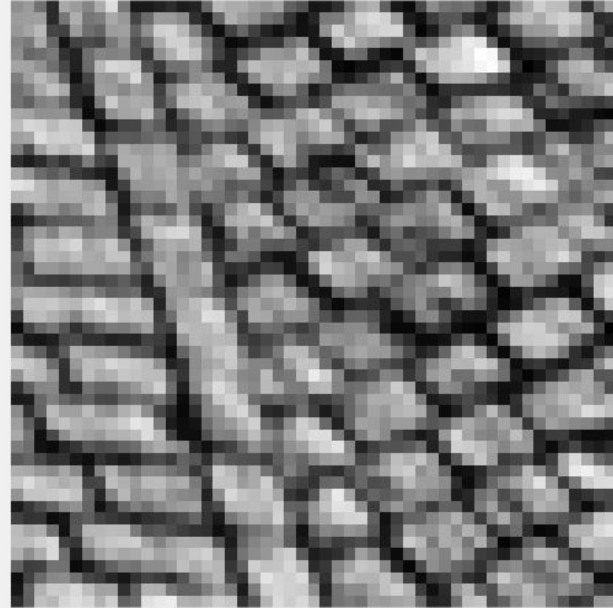


Image recovered from your DFT



Check the console as well:

```
Runtime: 3.654 s
Sum of absolute difference (in the frequency domain): 3.156e-10
Sum of absolute difference (in the spatial domain): 7.165e-12
>>
```



## Exercise 2

Implement the **function** `fourier_parts` in which:

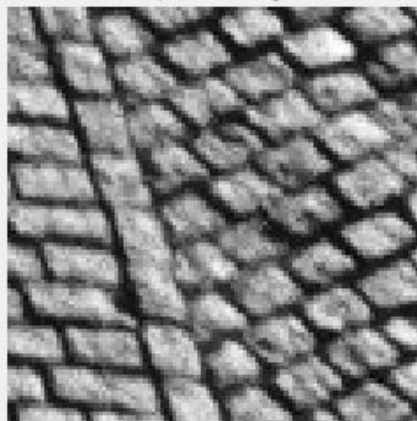
- Shift the input F-space matrix to the center (`fftshift`).
- Compute the phase (P) of the F matrix (`angle`)
- Compute the magnitude (M) of the F matrix (`abs`)

You should return two double matrices (they are NOT *complex* double)!

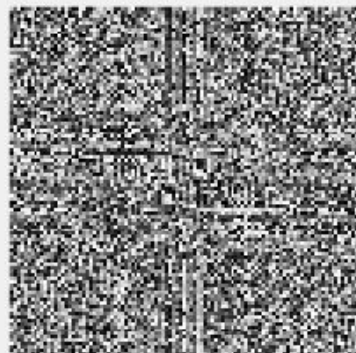
You can assume that the input is a 2D complex double type matrix.

Run `script2.m` to check your implementation and plot the DFT of an image.

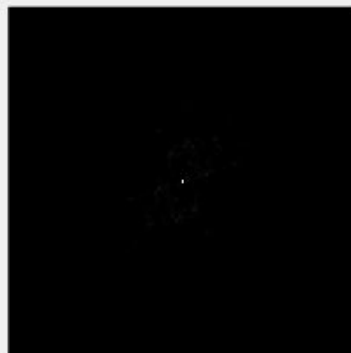
**Input image**



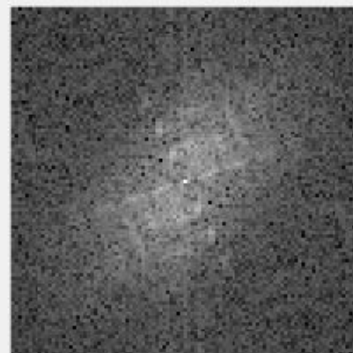
**Phase**



**Magnitude**



**Log magnitude**



## Exercise 3

Implement the **function** `mask_fourier` in which we replace the neighborhoods around the  $(x,y)$  points with complex zeros (frequency filtering).

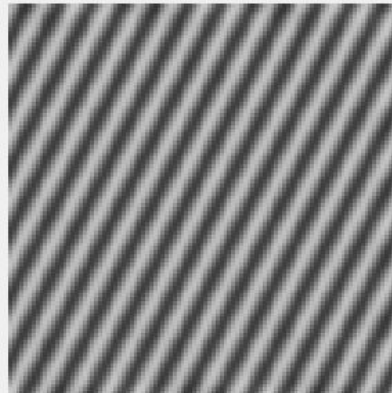
- Shift the F space to the center (`fftshift`).
- Round the input coordinate vectors (`x` and `y`).
- For each point, set the `r`-radius neighborhood of the point to `0+0i`. This step is exactly the same as in the non maximum suppression function of the previous Lab, please interpret `y` as the first coordinate, `x` as the second one.
- Undo the shift of the F space (`ifftshift`).

Run `script3.m` and examine the results.

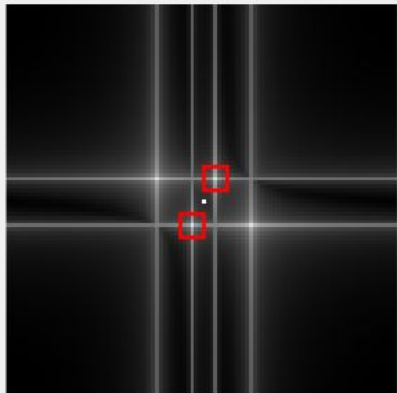
Original image



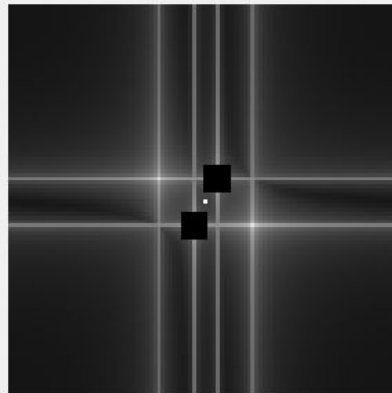
Filtered image



F-space with areas to suppress



Modified F-space



# Exercise 4

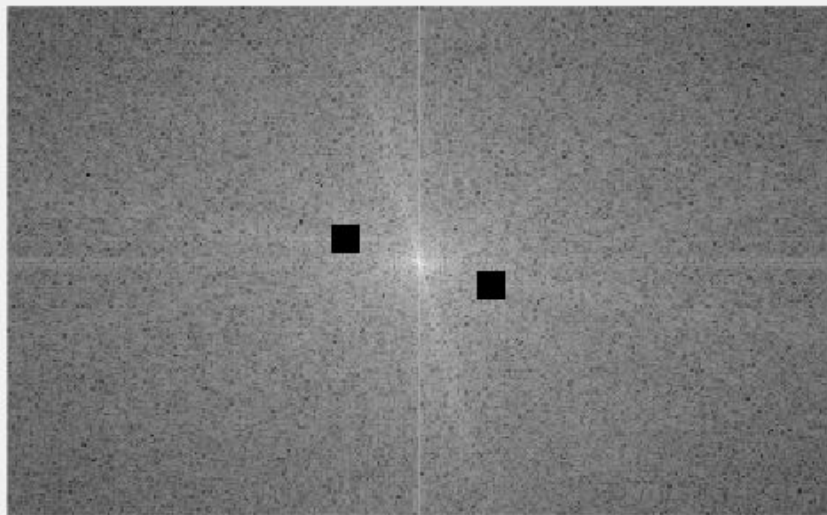
Read and understand the **script script4.m**. This script:

- Reads in an image and computes its DFT.
- Plots the log magnitude of the F-space.
- Using `ginput()` it asks the user to select some points on the magnitude plot. If the user is done with selecting points, *Return* (ENTER) key is pressed.
- The selected coordinates (float type) and the F-space is passed to the `mask_fourier` function together with a pre-defined radius value.
- The `mask_fourier` function should set the values in the neighborhood of the selected pixels to complex zero (`0 + 0i`).
- The new F-space (returned by the masking function) is transformed back to the spatial domain and the image is shown to allow visual comparison.

**Click on the center of the unwanted points.  
Press ENTER when finished.**



**Modified F-space**





**Original image**



**IFFT of the modified F-space**



**THE END**