Filter in the spatial domain

Build your own function called **SpatialFilt** that filters an image $f(x,y)$ with a Gaussian filter.

- Input parameters:
  - *myimage*: Image we want to filter $\big(f(x,y)\big)$.
  - *im_pixel_size:* Pixel size in the original image.
  - *k_size*: Size of the kernel.
  - *k_sigma*: Sigma value (σ) for the Gaussian filter in millimeters.
  - *edge_fill*: option for filtering pixels at the edges - 0: zeros, 1: repeat last pixel value.
- Output:
  - *filt_image:* Filtered image

Instructions:

- IMPORTANT: You must implement the whole process (**you cannot use Matlab functions: *filter*, or *conv functions***).
- You can assume an odd size of filter in the spatial domain (and maybe give an error message if the input value for "k_size" is even).
- Gaussian filter will be implemented using the following equation:

$$f(x,y) = e^{\frac{-(|r|-\mu)^2}{2\sigma^2}}$$

  where $|r|$ is the distance of position (x,y) to the center and with μ, the mean value of the Gaussian, which should be zero and σ, the value of sigma in mm.
- Make sure you normalize the filter, so the range of values in the filtered image is the same as the range of values in the original image.
- To optimize the computational time, you can implement the 2D filter in two 1D filtering steps, in the horizontal and vertical directions.

Filter in the Fourier domain

Build your own function called **FourierFilt** that filters an image $f(x,y)$ in the Fourier domain with a Gaussian low pass filter.

- Input parameters:
  - *myimage*: Image we want to filter $\big(f(x,y)\big)$
  - *im_pixel_size:* Pixel size in the original image.
  - *k_sigma_spat*: Sigma value (σ) **for the Gaussian filter in the spatial domain** in millimeters.
  - *zero_pad*: number of zeros to use for zero padding **in the smallest dimension** of the image. The zero padding in the other direction will be done accordingly, so we have the same resolution on both direction in the Fourier domain **if possible**. **It has to work for any value of zero_pad, including zero_pad=0, which means no zero padding**.
- Output:
  - *filt_image: Filtered image*

Instructions:

- Use ***fft***, ***ifft*** and ***fftshift*** Matlab functions for the Fourier transform.
- Be careful, the image is not square, so if you take the FFT with a different number of pixels in the horizontal and vertical directions, you will end up with a different pixel size in the Fourier domain. Think about the consequences of that and make sure you applied the Gaussian filter isotropically.

- Make sure you normalize the filter, so the range of values in the filtered image is the same as the range of values in the original image.
- The sigma value is given for the Gaussian in the spatial domain. You will have to implement the equivalent Gaussian in the Fourier domain, taking into account that the Fourier transform of $f(x)$ is $F(w)$, given as:

$$f(x) = e^{-ax^2} \rightarrow F(w) = \sqrt{\frac{\pi}{a}} e^{\frac{-\pi^2 w^2}{a}}$$

Gaussian filter in the Fourier domain must be created directly in the Fourier domain taking into account the given equation, **explicitly calculating the value of sigma of the gaussian in the Fourier domain**. You can check if it is ok by comparing it with the Fourier Transform of the Gaussian filter in the spatial domain.

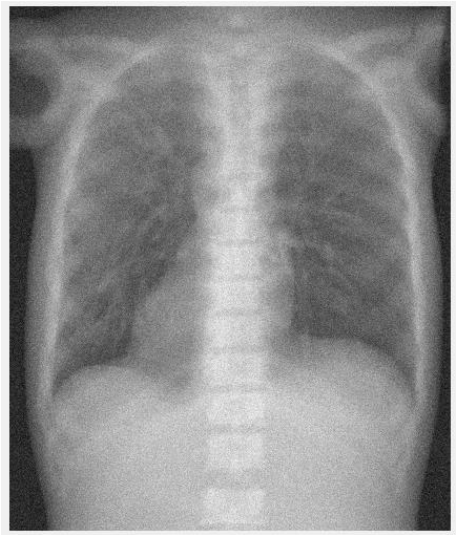
Test function

Write a script called **Main** that performs two filtering operations on example image 'ThoraxXray_500x420_noise':

a. One calling function calling function **SpatialFilt**, with Gaussian filter, *k_sigma*=1.5 mm, mask size 21×21 and *edge_fill*=0.

b. One calling **FourierFilt**, with *zero_pad*=420 and the same sigma in the spatial domain, that is, *k_sigma_spat*=1.5 mm.

Image 'ThoraxXray_500x420_noise' has a matrix size of 500×420, pixel size of 0.8 mm, and data type is float (32 bit real in ImageJ), **endianess: big endian**.



The script must finish with the display of two figures:

- One with the original image, *myimage*, the two filtered images, *filt_image*, generated in 2 (using **subplot**).
- One with three "difference" images:
  a. original image - image filtered in the spatial domain;
  b. original image - image filtered in the Fourier domain;
  c. image filtered in the spatial domain - image filtered in the Fourier domain.

To evaluate these results, the images in each figure need to be comparable, so **you should use the same LUT for the three images** in each figure and add the colorbar to see the range of values. **Show your images in the right position** (use transpose when needed).

General tips:

- You are expected to avoid loops by using Matlab function ***meshgrid***.
- You need to implement both the edge filling in the spatial domain and the zero padding in the Fourier domain manually (**do not use *padarray* or the zero padding option in the *fft* function**).
- Kernel normalization is not the same in spatial and Fourier domains, think what you need to do to the filter in each domain so the range of values of the image is not changed during the filtering.

Submission

Upload a zip folder with everything you need (functions and images) so that Main code runs directly after uncompressing the file.

Check it before uploading it, **if it does not run showing the two figures at the end, the grade will be zero**.