# Assignment 1 – python viewer and MRI modalities, FFT. *Due Friday June 5 at 11:55 pm*

**CS463/516 Medical Imaging**

**Submission format:** one partner should submit a .zip file to moodle with:

1) pdf containing images and descriptions showing you have completed all parts of the assignment/bonuses.
2) Your python source code
3) A README file describing any incomplete parts of the assignment and your group member names.

The datasets are available here: https://drive.google.com/open?id=1RqM4qSh4L6YKrNOjdCYJJy8Aabe6n5IQ
Use the nibabel libraries to load your images from disk: https://nipy.org/nibabel/
You can use any IDE you want, but you must use python and the numpy libraries in this course

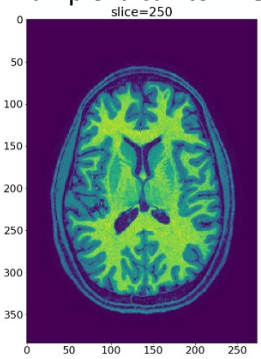
You have been given 3d MRI images from 5 separate modalities as follows:

1) T1-weighted (t1.nii)
2) T2-weighted (t2.nii)
3) Susceptibility-weighted (swi.nii)
4) Time of flight angiogram (tof.nii)
5) Blood oxygen level dependent (bold.nii)
   (all these images were taken from https://openneuro.org/ )

For part 1 just use t1.nii, for part 2 you will need all the modalities. nii is a medical imaging format (nifti format)

## Part 1: Python viewer

**Part 1a (50%):** Create a `viewer` function in python that displays a slice from a 3d image and allows for scrolling through the image slices using keypad or mouse wheel.
Example: a call to `viewer(brain,slice=250,view='axial')` should produce the following image:

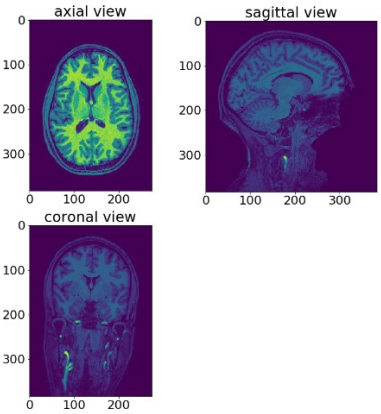



Where 'brain' is a 3d numpy array with the z-slices in the 3rd dimension (most images are structured this way). If the user rotates the mouse wheel, or presses the 'up' or 'down' arrow keys, the image should change to a different slice.
You should make heavy use of the matplotlib libraries for this question.

*Bonus +2.5%:* add histogram equalization option 'histeq', so if the user calls:
`viewer(brain,slice=250,view='axial',histeq=True)`
this will produce a histogram-equalized image in the viewer, instead of the raw image.

*Bonus +2.5%:* add an option to display all views simultaneously in the same plot using sub-plots. When the user places the mouse over one of the subplots and rotates the mouse wheel, only the selected subplot slice will change.

Sub-plot example below: You can also see the need for histogram equalization in sagittal/coronal view (blood vessels too bright relative to the rest of the tissue)

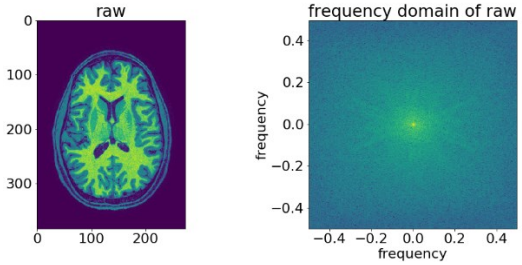



You can also add other options as you see fit (different colormaps, intensity limits, etc.). Part of your grade will be based on how creative you are in implementing the viewer.

## Part 2 modalities and frequency-domain filtering

Smoothing and edge-detection are fundamental image processing operations. In part 2, you will use numpy's fft functions to perform edge detection and smoothing on 3d images in the frequency domain.

**Part 2a (10%):** do a simple 2d FFT on one of the z-slices (axial slice) and display the result. You will need to use fftshift and the logarithm to correctly visualize the fft. It should look similar to below:



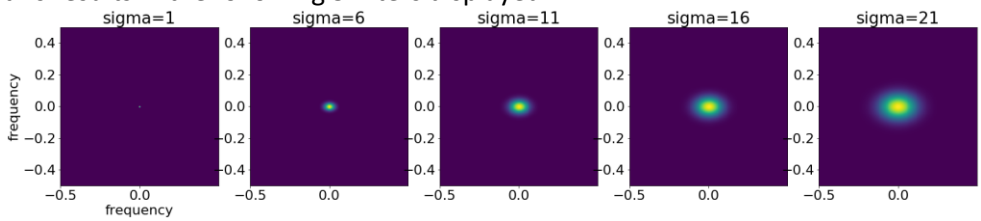

Reproduce this figure using 't2.nii' from the data download.

**Part 2b (20%):** Generate frequency-domain gaussian filter using meshgrid and use to do frequency space filtering

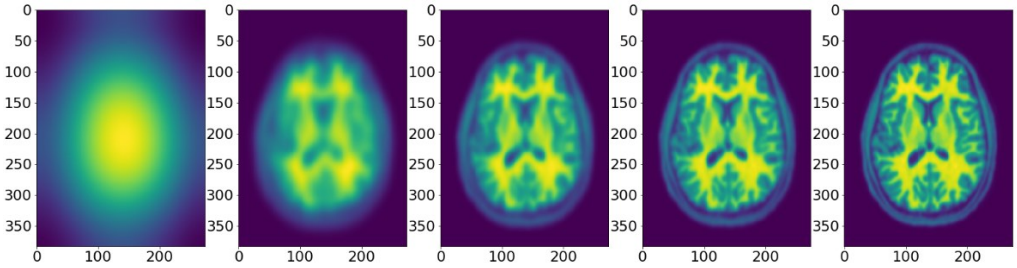
Example: generating filters for 5 different sigmas using the following code:
Where 'rotim' is the 2d FFT of the image we want to filter:

```
sz_x = rotim.shape[0]
sz_y = rotim.shape[1]
[X, Y] = np.mgrid[0:sz_x, 0:sz_y]
xpr = X - int(sz_x) // 2
ypr = Y - int(sz_y) // 2
count=1
for sigma in range(1,25,5):
    gaussfilt = np.exp(-((xpr**2+ypr**2)/(2*sigma**2)))/(2*np.pi*sigma**2)
    plt.subplot(1,5,count); plt.imshow(gaussfilt); plt.title('sigma='+str(sigma))
    count =count + 1
```

this results in the following 5 filters displayed:




Below is the corresponding image after multiplying the above filters with the Fourier transform of raw and then inverting the Fourier transform



Note how filters which highlight more the low frequencies (higher values near center) also cause more blurring. Filters that are more spread out (like sigma=21) cause less blurring.

Reproduce the above figure on the 'swi.nii' imaging modality.

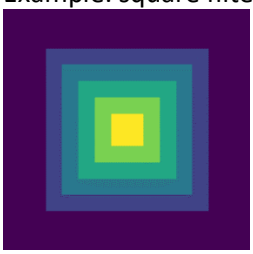
**Part 2c (20%):**

Load and display each modality in your viewer.
Perform smoothing (As above) and also edge detection using frequency domain filtering on each modality. you may need to use `fftn`, or if you prefer just do a 2d version and show a single slice.

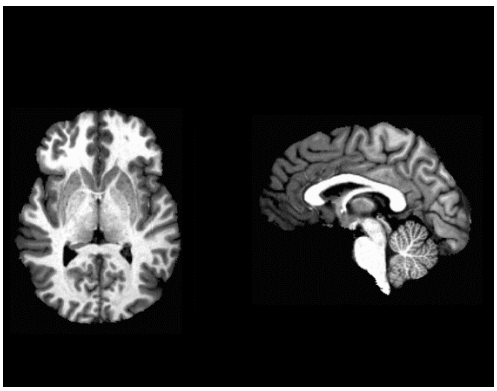
**Show the result as 5 separate plots (1 for each modality), each plot has 4 sub-plots:**
Subplot1: raw image (single slice)
Subplot2: fft(raw image) – shown as above in part 2a
Subplot3: raw image after edge enhancement using frequency domain filtering (find an edge detection filter)
Subplot4: raw image after smoothing using frequency domain filtering (can use filters defined as above)
Do NOT use other python functions to do smoothing/edge detection – must use your own filters, and `fftn`

Be sure to show these 5 separate plots in your pdf.

**Bonus question +2.5%:** Experiment with different filter shapes. What happens if you use a bar or a square, instead of a gaussian smooth?
Example: square filter (use this to multiply fourier transform of the image)



**Bonus question: +2.5%:** all the images so far have skulls. Here is an image with no skull. Can you explain how this image was created? Give details on how the skull was removed from the image.



**Bonus question:** +10% how did I create Mystery image (below)? Explain what the contrast is based on.

Mystery image