UNIVERSITY OF CAMBRIDGE

Data Science in Medical Imaging

MPhil in Data Intensive Science

Submission: 23:59 on Friday the 4th of April 2025

Coursework Minor A2: Data Science Applications to Medical Imaging
Dr. A. Biguri, Dr. J. Kaggie, Dr. L. Escudero Sánchez

You need to submit your answers for the three modules below (all of them). The datasets needed to complete this coursework can be found in this link: https://shorturl.at/qtgJB

The coursework will be submitted via a GitLab repository which will be created for you. You should write a report of up to 3000 words to accompany the software you write to solve the problems. You should place all your code and your report in this repository. The report should be in PDF format in a folder called "report". You will be provided access to the repository until the deadline above. After this you will lose access which will constitute submission of your work.

The code associated with the coursework should be written in Python and follow best software development practice as defined by the Research Computing module. This should include:

- *Writing clear readable code that is compliant with a common style guide and uses suitable build management tools.*
- *Providing appropriate documentation that is compatible with auto-documentation tools like Doxygen/Sphinx.*
- *The project must be well structured (sensible folder structure, README.md, licence etc.) following standard best practice.*
- *Appropriate and robust unit tests for automatic validation.*

- *Uses appropriate version control best practice, including branching for development and testing, and commit hooks to protect "main".*
- *Appropriate containerisation to ensure portability of the project to other computers and operating systems if necessary.*

MODULE 1: PET-CT Image Reconstruction

You are given control of a PET-CT scanner, as the computer that does the reconstruction is broken. We have some patient that was scanned in the machine, and we need the to give the reconstructions to the doctors, its your job to clean up the data (sinograms) and produce reconstructions.
The following code will load and display the measured sinograms.

```
ct_sino=np.load("ct_sinogram.npy")
pet_sino=np.load("pet_sinogram.npy")
```

- The CT sinogram is exactly what has been measured, with no data changing. Lucky you! All CT scanners in the world are fan-beam geometry, but this one is the only one that is parallel beam (i.e. X-ray source is a line parallel to the detector). This is only important because you can now use `radon` and `iradon` from `skimage`, which you would not be able to use if it was fan-beam shaped. It has been acquired around 180 degrees, one measurement per degree.
- The PET sinogram has been minimally processed. In particular, the listmode data has been made into a singoram for you, and the singles/random events and the background noise have been corrected, so they are not present in the singorams. It has been binned to 180 degrees, on 100 different angles.


```
# Display
plt.figure()
plt.subplot(121)
plt.imshow(pet_sino,cmap="gray_r")  # PET scientist like to reverse the colorbar
plt.axis("equal")
plt.subplot(122)
plt.imshow(ct_sino, cmap="gray")
plt.axis("equal")
```

Exercise 1.1: Your first task is thus to clean up the sinograms before reconstruction. The scanner provides several measurements to help you.
For CT:

- Dark fields "ct_dark.npy"
- Flat fields "ct_flat.npy"

For PET:

- Detector gain calibration sinogram for PET. These scintillator are very complex so a calibration step was done to compute the gain of each of them given the same input signal. "pet_calibration.npy"

(TURN OVER

Don't attenuation correct the PET sinogram yet. We need a CT reconstruction for that, as PET and CT sinograms are not the same size, and interpolating in sinogram space is not desirable. Similarly, to estimate the scatter we need the CT recon, so lets wait until we have it.

[3]

[Hint: The flat fields already contain I_0, so if they are appropriately used, you don't need this value in the Beer-Lambert equation.]

Exercise 1.2: The next steps is to obtain both a scatter correction for PET, and also an attenuation field. However, you noticed that the sinograms are different shape, and interpolating in sinogram space is not trivial (and generally undesired). We need to reconstruct the CT image, and rescale that.

- Reconstruct the CT image with FBP, and OS-SART. [5]

SART was the first iterative algorithm proposed in CT. OS-SART is a "stochastic" version of it. In essence, OS-SART is:

$$x^{k+1} = x^k + \gamma * A_i^T (A_i x^k - b)$$

Where k is the iteration number and A_i corresponds to a subset of A (i.e. you only want to forward/backproject *some* angles). Divide A_i into 10 equal sized parts (i.e. you divide the sinogram into parts with equal number of angles). Find the appropriate values of γ and $K = max(k)$ that produces a better image than FBP.

[Hint: don't expect a significantly better image for this case and this algorithm.]
Extra: How does it to compare to SIRT (i.e. gradient descent as we did in the CT practical)? Does it reach a good result faster or slower? Does it need more or less iterations?

Exercise 1.3: From the CT image, we want to obtain both scatter estimation to clean up the PET data and attenuation sinogram of the same size as the PET sinogram, for attenuation correction.
Given scatter correction is quite a complex tasks, we are going to ignore it in this exercise, and assume we can correct it. Attenuation correction though is necessary. To do so, resize the CT image to the size of the PET image, produce a sinogram, and attenuation correct the PET sinogram.
The PET image, at reconstruction, has pixels of 4.24 mm in size, and the CT image of 1.06 mm

[3]

Exercise 1.4: We are now ready to reconstruct the PET image. [5]
Reconstruct with FBP and with OSEM. OSEM is defined as:

$$x^{k+1} = x^k A_i^T \left(\frac{b}{A_i x^k} \right)$$

Extra: How does OSEM compare to MLEM (i.e. using A and A^T full, without subsets)? Does it reach a good result faster or slower? Does it need more or less iterations?

Exercise 1.5: Theory. Write a few sentences/code to answer these questions.

1. 1.5.1 Display the PET and CT scans overlayed [1]
2. 1.5.2 If we had a TOF-PET scanner, how accurate would we need to be at measuring time to know the location of the emission so precisely that reconstruction would not be needed? [1]
3. 1.5.3 Why is OSEM the most used algorithm in PET, but not in CT? What makes OSEM different (mathematically) to gradient descent? [1]
4. 1.5.4 PET-MR scanners exist. What is the extra data processing you need to take in a PET-MR that does not exist in a PET-CT? How is it done? [1]

MODULE 2: MRI Image Denoising

For this module you will work with k-space data (Fourier space data) from MRI – and deal with a few common artifacts. The data contains a 3D knee image with six coil receivers.

Exercise 2.1 Visualisation and identifying noise

- Load the complex data "kspace.npy" using the `np.load()` routine. Determine which array dimension is the coil dimension. [1]
- Create an image of the magnitude of k-space for each coil. You can use `np.log1p()` to scale the kspace data for easier visualisation. [1]
- Transpose the data into image space using the Fourier transform. Create and show a magnitude and phase image from one coil. [1]
- Show the magnitude images from all coils. [1]
- Choose and appropriate way to combine the data from all of the coils into a single image (in image space). Do this by squaring the image from each coil, summing the magnitudes, and then taking the square root of the final result. Show this image. [2]
- Write about your observations. [2]

Exercise 2.2 Removing noise

- Choose three denoising methods using the image space data, and show the effects of the denoising for all coils. Denoising methods can include mean, bilateral, wavelet, or any other filter. A number of filters can be found in scikit-image. "gaussian_filter" can be found in scipy.ndimage. Comment on the different denoising methods. [10]
- Start from the original noisy k-space. For the first coil, apply a low-pass Butterworth filter within k-space and show the new image, showing both magnitude and phase images. How does this denoising method compare to the image-based methods? [6]
- Using one of denoising methods on the images, recreate a new combined image. [4]
- Comment on methods that might be able to further improve the combined image quality. [2]

The Butterworth filter can be defined as such:

```python
def butterworth_lowpass_filter(shape, D0=30, n=2):
    P, Q = shape[0], shape[1]
    u = np.arange(P) - P // 2
    v = np.arange(Q) - Q // 2
    U, V = np.meshgrid(u, v, indexing='ij')
    D = np.sqrt(U**2 + V**2)
    H = 1 / (1 + (D / D0) ** (2 * n))
    return H
```

MODULE 3: CT Image segmentation and classification

For this module you will use the images provided in the link on page 1, Module3. The folder contains CT scans for a selection of 40 patients of lung cancer from the public dataset LIDC-IDR (https://www.cancerimagingarchive.net/collection/lidc-idri/). The images are provided in NIfTI format and there is one NIfTI (.nii) file for the scan (one per patient) and one file for the segmentation of a lung nodule.

Exercise 3.1: Image segmentation

- Open the NIfTI files and create one Numpy array per patient scan and one per segmentation mask. [Hint: you can use existing Python libraries such as SimpleITK or nibabel.]. Find the range of voxels (3D pixels) in which the segmentation exists for each patient. Create a numpy array with a subvolume of the images by increasing 30 voxels in the x and y directions in both senses (min and max) and 5 in the z direction. [7]
- Create your own processing-based segmentation function using a simple thresholding algorithm (do not use supervised machine learning). Apply your algorithm to the subvolumes created in the previous step. To do so, find the min/max of the intensities corresponding to voxels inside the segmentation to set a min/max threshold. [4]
- Compare your segmentations obtained with the thresholding algorithm and the ground truth ones. Is your method working as expected? Describe how you could improve it. [4]

Exercise 3.2: Image feature extraction and classification

For this exercise you should only use the ground truth segmentations from Exercise 3.1 and the clinical labels, which define which nodules were malignant vs benign, provided in labels.csv.

You will work with the following three histogram-based *radiomic features*. Let V be the set of intensities or pixel values of the N voxels (3D pixels) in the Region of Interest (ROI), i.e. the image voxels inside the segmentation. We can construct a histogram with those pixel values using M equally-spaced bins and calculate:

1 - Energy
It is a measurement of the intensity of the voxels, that we can define as:

$$E = \sum_{i=1}^{N} (V_i)^2$$

2 - Mean Absolute Deviation (MAD)
MAD is the mean difference between all intensity values and the Mean Value ($\bar{V}$) of the set of voxel intensities:

$$MAD = \frac{1}{N}\sum_{i=1}^{N}|V_i - \bar{V}|$$

3 - Uniformity
Uniformity is a measurement of the homogeneity of the image, defined as:

$$U = \sum_{i=1}^{M} p_i^2$$

where p is the normalised histogram, i.e.

$$p_i = P(i)/N$$

being $P(i)$ the number of entries in the ith bin of the histogram.

You will use these *radiomic features* as follows:

- Write your own code to calculate each of the three features described above. In order to calculate the Uniformity, first you have to find the ranges of min-max intensities of the voxels and decide a sensible number of equally-spaced bins that works for all cases. [6]
- Compute the value of each feature on each patient's nodule. [3]
- Which of those features would you use to classify between the benign and malignant lesions and why? [6]

END OF PAPER