Exercises Instructions

Contents

Introduction..1
Start of exercises..2
Exercise 1: Data Simulation Brain 2
Exercise 2: Data Simulation Thorax with PET...2
Exercise 3: Data Simulation Thorax with SPECT...2
Exercise 4: Preparation for Image reconstruction exercise 3
Exercise 5: Image reconstruction part 1: EMML and OSEM 3
Exercise 6: Image reconstruction part 2: adding Poisson noise 4
Exercise 7: Image reconstruction part 3: MAP 4
Exercise 8: Random estimation 5
Exercise 9: Scatter Correction 5
Exercise 10: Motion Correction..6
Exercise 11: Image reconstruction part 4: PSF and MAP 7
Appendices...8
File extensions..8
Python 8
Image display outside of Python 10
STIR commands for evaluation 10

Introduction

This material is intended for practical demonstration using STIR on PET and SPECT Image Reconstruction.

Simulated data will be prepared during the exercises. These are based on 2 sets of images:

- Thorax phantom data are obtained from the recent open access article: *Tsoumpas et al 2013 Phys Med Biol*.
  We have two respiratory gated positions of a thorax FDG PET phantom along with the corresponding CTAC image.
- Brain data are obtained from BrainWeb.
  We have a segmented brain-map.

You will probably only want to run either the brain or the thorax data (except for the motion correction exercise which is currently only for the thorax).

The input data are stored in the folders called EX_*, **but you will need to run the scripts from the "main" exercises folders** (open a terminal, `cd` to where you extracted the exercises, and always `cd` back after every exercise).

We are using Python for some of the exercises. Python is an open-source interactive language, a bit like MATLAB. We provide Python scripts for the evaluation, so you should be fine. It would be best to read a Python tutorial before the course. We will use Spyder as our Python environment.

See the appendices at the end of this document for some information to get started. ***Please read this before the course.***

Authors:
Elisabetta Grecchi, Irene Polycarpou, Charalampos Tsoumpas, Kris Thielemans

Start of exercises

Open a terminal and type

```
cd ~/exercises
spyder&
```

You will then alternate between the terminal (to run shell scripts that execute STIR commands) and the spyder window (to run Python scripts that read in data and make plots).

Note:
The shell scripts (*.sh) should all write "DONE" or "Done" at the end. If they didn't, something went wrong. Check the log files (*.log) in the output directory for the script that failed.

Exercise 1: Data Simulation Brain

(Always run scripts from the exercises directory)

This is a simple simulation of a brain phantom. PSF is incorporated. Scatter is set to zero. Randoms are constant.
The aim of the exercise is get familiar with the scripts and evaluation environment, and to look at projection data in different ways (sinograms, by view etc)

Read and run script:

```
./run_simulation_brain.sh
```

Open the file /home/stir/exercises/evaluate_simulation_brain.py in Spyder. You can do this via its menus, or by typing in the terminal:

```
spyder evaluate_simulation_brain.py&
```

Exercise 2: Data Simulation Thorax with PET

(Always run scripts from the exercises directory)

This is a simple simulation of a thorax phantom (2 gates). PSF is not incorporated. Scatter is simulated using STIR. Randoms are constant.
The aim of the exercise is first to do the same things as for the brain, to see differences in sinograms etc, get a first view on scatter, but then to move on to seeing the effect of motion.

Read and run script:

```
./run_simulations_thorax.sh
```

Start spyder with the evaluation script

```
spyder evaluate_simulation_thorax.py&
```

or if spyder is running, just open the file.

Exercise 3: Data Simulation Thorax with SPECT

(Always run scripts from the exercises directory)

This is a simple simulation of single slice of a thorax phantom for SPECT. PSF is incorporated. Scatter is set to zero.
The aim of the exercise is first to see how PECT sinograms differ from PET.

Read and run script:
```
./run_simulations_SPECT.sh
```
Output is in working_folder/single_slice_SPECT.

Start spyder with the evaluation script
```
spyder evaluate_simulation_SPECT.py&
```
or if spyder is running, just open the file.

Exercise 4: Preparation for Image reconstruction exercise

For the thorax reconstruction exercise, we first need to generate a new simulation data set. This time just a single slice to speed things up. Scatter is also set to zero for simplicity here.

```
./run_simulation_single_slice.sh
```

Output is in working_folder/single_slice. There is no real need to look at the generated results as they are a single sinogram of the thorax simulation (but without scatter and with PSF).

Exercise 5: Image reconstruction part 1: EMML and OSEM

(Always run scripts from the exercises directory)
You will need to have run the corresponding simulation script from the previous section. Output is in working_folder/single_slice or working_folder/brain or working_folder/single_slice_SPECT

We will now look at EMML and OSEM. A sample script is provided to generate results

```
./run_reconstruction_thorax.sh
```
Or
```
./run_reconstruction_brain.sh
```
Or
```
./run_reconstruction_SPECT.sh
```

This will first run FBP, EMML (also known as MLEM) for 240 iterations, then OSEM (with 8 subsets) for 240 sub-iterations. For both of these, the images obtained after every 24 updates will be saved. It will also continue EMML and OSEM from 240 sub-iterations and continue for 8 more, writing images at every subiterations. In addition, the EMML and OSEM images at 240 iterations are also post-filtered with a Gaussian filter.
Output is in working_folder/single_slice or working_folder/brain or working_folder/single_slice_SPECT *resp*.

The script runs the reconstruction on the noiseless simulations. See the next exercise for adding noise.

Sample questions to address:
- Is it worth running EMML? Why not simply use OSEM?
- At late iterations, is there a difference between OSEM and MLEM convergence behaviour?

Start spyder with the evaluation script
```
spyder evaluate_reconstruction_brain.py&
```
or if spyder is running, just open the file. If you have run the thorax or SPECT simulation, just adjust the path in Line 23 (or there abouts)

```
  os.chdir('working_folder/brain')
```
to the appropriate location.

Exercise 6: Image reconstruction part 2: adding Poisson noise

(You will have to run the simulation and reconstruction scripts first.)

We can make the simulation more realistic by adding noise to the data. An example would be (please *adjust the directory name to your case, e.g.* `working_folder/`*`brain`* or `working_folder/single_slice_SPECT`)

```
cd working_folder/single_slice
# save noiseless data results in a subfolder
mkdir noiseless
cp *.* noiseless
poisson_noise -p my_noisy_data.hs  my_prompts.hs 0.1 1
# overwrite the noiseless projection data with the noisy ones
cp my_noisy_data.hs my_prompts.hs
cd ../..
```

For the SPECT exercise, replace `my_prompts.hs` in the text above with `my_sim.hs`.

Run `poisson_noise` to understand what these arguments mean.

As we overwrite the data[1], we can just use the same reconstruction and evaluation scripts as before (Exercise 5). An alternative would be to adjust the reconstruction parameter files to use your new noisy data (input) and change the filename used for the output, and change your evaluation scripts.
If you want to "reset" to the noiseless case, you can of course copy the data in `noiseless` back.

```
cd working_folder/single_slice
# save noisy data
mkdir noisy
cp *.* noisy
cp noiseless/* .
cd ../..
```

Sample questions to address:

- Are the answers to the questions in Exercise 5 the same now that we added noise?

Exercise 7: Image reconstruction part 3: MAP

This exercise needs results from the previous step (as the MAP reconstruction starts from an OSEM image in this exercise). Output is in working_folder/single_slice or working_folder/brain or working_folder/single_slice_SPECT *resp.*

We will now look at OSL and OSSPS with a Quadratic Prior. A sample script is provided to generate results

```
  ./run_reconstruction_thorax_MAP.sh
```
or
```
  ./run_reconstruction_brain_MAP.sh
```
or
```
  ./run_reconstruction_SPECT_MAP.sh
```

This will run OSL and OSSPS (continuing from a previous OSEM image after 24 subiterations).

Warning:

[1] Note that we only copy the header (.hs) file as this header contains a field that "points" to the raw float data (.s).

If you added noise to the data (exercise 6), the MAP reconstructions will also use the noisy data of course. However, because of the initialization with OSEM, you want to run the normal reconstruction script (Exercise 5) first whenever you change the noise level.

You could for instance first run this exercise with the noisy data, then move the noiseless data back in place, then re-run the reconstruction script for the current exercise.

Sample questions to address:

- Do OSL and OSSPS generate the same results?
- Does this depend on the penalty factor? Noise level? Iteration number? Initialisation (try to remove the initial estimate for instance).

Start spyder with the evaluation script

```
spyder evaluate_reconstruction_brain_MAP.py&
```

or if spyder is running, just open the file. If you have run the thorax or SPECT simulation, just adjust the path in Line 23 (or there abouts)

```
os.chdir('working_folder/brain')
```

to the appropriate location.

Exercise 8: Random estimation

This exercise uses Maximum Likelihood estimation to find singles from a sinogram of ‘delayed coincidences’. These estimated singles are then used to construct the randoms estimate.
Output is in `working_folder/randoms`.

First run

```
./run_randoms.sh
```

This will construct a noiseless randoms sinogram (by using ground-truth singles), add noise, and run the Maximum Likelihood estimation.

Sample questions to address:

- How do the estimated singles differ from the fansums? In which situations does this matter?
- Is there still noise in the estimated random sinogram? (You could do this by changing the seed for the Poisson noise generator and re-running the script).

Start spyder with the evaluation script to help you

```
spyder evaluate_randoms.py&
```

or if spyder is running, just open the file.

Exercise 9: Scatter Correction

(Always run scripts from the exercises directory)

There are 4 example scatter estimation scripts which can be used to investigate different questions about scatter.

Run 0
Ideal (i.e. correct attenuation map, scatter simulation matches with how the data was generated) scatter correction (using 3 scatter correction loops)

Run 1
Calculate scatter by using a smaller energy window than that simulated. This will demonstrate if the scaling technique works. We have selected 425keV for the lower energy window (original is 350keV).

~~Run 2~~ (this is currently not available)
~~Scatter correction using the scatter estimation from the first gate for both gates. This will demonstrate how sensitive is scatter in choosing different but adjacent gates.~~

Run 3
Perform reconstruction & attenuation correction by using wrong attenuation map. In the particular exercise we have assigned bone attenuation value to lung attenuation value for the first gate. Then we use this wrong attenuation map located at the first gate to correct for attenuation and scatter for each gates.

You should run the scripts as folllows (you can try reading it but these scripts are relatively complicated):

```
  ./run_scatter_0.sh
```

The scripts make the following files in `working_folder/GATE1` (and similar in `working_folder/GATE2`)

- `input_g1.hs`: "measured" sinogram after randoms correction
- `my_scatter.hs`: sinogram output of the simulation (i.e. ground truth)
- `scatter_estimate_run0.hs` etc: sinogram output of the iterative scatter estimation
- `FDG_g1.hv`: input of the simulation (i.e. ground truth image)
- `FBP_recon_with_scatter_correction_run0.hv`: FBP reconstruction of the scatter corrected data

Example questions to answer:
- How close is the scatter estimate in the ideal case of a simulation and how does this affect the image reconstruction? (run0)
- How different is the scatter (and its estimate) between gate 1 and gate 2? (run0)
- How different is the scatter (and its estimate) if you have a wrong estimate of the energy dependence of the detection efficiency? (run1)
- What happens to the scatter estimate and the reconstructed image if you use the wrong attenuation image? (run3)

2 Python scripts are provided as a starting point for investigating the results.
- `evaluate_scatter_run0.py` which reads results from run_scatter0.sh and displays them comparing with the truth (i.e. simulation input and simulation scatter output) for GATE1
- `evaluate_scatter_run3.py` reads results from run0 and run3 and displays them (also for GATE1). You should be able to use this script with small modification to look at results from run 2. (make a copy, don't overwrite the existing one).

Exercise 10: Motion Correction

(Always run scripts from the exercises directory. This exercise depends on the output of `run_simulations_thorax.sh`)

There are 2 scripts for Part I (MCIR vs noMC):

`run_MCIR_0.sh`
  Correct for motion using valid motion vectors and the previously calculated scatter background
  output folder: `working_folder/MCIR`

`run_MCIR_1.sh`
  Do not correct for motion
  output folder: `working_folder/noMC`

and 2 scripts for Part II (mismatched AC):

`run_MCIR_2.sh`
  Correct for motion using valid motion vectors for emission and the previously calculated scatter background but use the same average attenuation map for both gates

output folder: `working_folder/MCIR/avAC`

`run_MCIR_3.sh`

Correct for motion using valid motion vectors for emission and the previously calculated scatter background but use the same attenuation map (based on CTAC_g1) for both gates

output folder: `working_folder/MCIR/g1AC`

run scripts, e.g.:

```
./run_MCIR_0.sh
./run_MCIR_1.sh
```

It takes about 30seconds to complete each reconstruction. If there are problems, you can confirm the scripts ran OK:

```
less working_folder/MCIR/MCIR.log
less working_folder/noMC/noMC.log
```

(quit `less` by pressing q)

To save some time, you can run Part II n the terminal while already evaluating Part I.

2 Python scripts are provided as a starting point for investigating the results: `evaluate_MCIR_Part_I.py` and `evaluate_MCIR_Part_II.py`.

Part I compares motion correction with no motion correction result and shows the forward motion & backward motion vector images.

Part II compares motion correction using the three different attenuation correction files (one for each gate, the one obtained from the average, the one obtained from gate 1)

Example questions to answer:

- How close is motion correction in the ideal case of a simulation and how does this affect image reconstruction? (Part I)
- What type of motion is simulated? (Part I)
- How important is the use of the correct attenuation map? (Part II)
- Any comment on scatter correction? (Part II)

Exercise 11: Image reconstruction part 4: PSF and MAP

This exercise needs results from exercises 5 and 7. So, you should already have done the following steps:

```
./run_simulation_SPECT.sh
# oaptionally add noise in Exercise 6
./run_reconstruction_SPECT.sh
./run_reconstruction_SPECT_MAP.sh
```

We will now look at OSEM and OSSPS (with a Quadratic Prior) when PSF modelling is included in the reconstruction. We will only do this for SPECT as at present, STIR PSF modelling is PET is hard to modify. A script is provided to generate results

```
./run_reconstruction_SPECT_PSF.sh
```

This will run OSEM and OSSPS (continuing from a previous OSEM image after 24 subiterations) with PSF model (check e.g. `OSEMPSF.par`).
Output is in working_folder/single_slice_SPECT.

Warning:

If you added noise to the data (exercise 6), the MAP reconstructions will also use the noisy data of course. See the MAP exercise for more info.

Sample questions to address:

- Does PSF-modelling increase resolution?
- Do you see edge-effects or overshoots in the reconstructed images? For both OSEM and OSSPS or only OSEM? Why?
- What happens to the noise "texture" of the images if you reconstruct with noisy data. Are overshoots worse?
- Extension: you could try to underestimate the PSF (edit `OSEMPSF.par` and change the collimator modelling and re-run the reconstruction script).

Start spyder with the evaluation script

```
spyder evaluate_reconstruction_SPECT_PSF.py&
```

or if spyder is running, just open the file.

Appendices

File extensions

.hv: Interfile header for image (volume)
.ahv: (ignore) old-style Interfile header for image
.v: raw data of image (in floats)

.hs: Interfile header projection data (sinograms)
.s: raw data of projection data (in floats)

.par: STIR parameter file

.sh: Shell script (sequence of commands)

.bat: Windows batch file

.log: log file (used to record output of command)

.py: Python file

Linux Terminal

If you have never used a Linux/Unix terminal before, have a look at a tutorial at https://help.ubuntu.com/community/UsingTheTerminal.

You can use UPARROW to go to previous commands, and use copy-paste shortcuts `Left-CTRL-SHIFT-C` and `Left-CTRL-SHIFT-V`.[2]

Python

We use Spyder as a nice Integrated Development Environment (IDE) for Python (actually iPython which is a slightly friendlier version of Python). You need only minimal knowledge of Python for this course, but it would be good to read-up a bit (see below).

[2] On Windows and Linux, VirtualBox sets the "host-key" by default to Right-CTRL, so unless you changes this, you have to use Left-CTRL to "send" the CTRL-keystroke to the Virtual Machine.

You will normally work by loading an example script in Spyder in the editor, executing it bit by bit, and then editing it to do some more work. Useful shortcuts for in the editor (these are in Windows-style, including the usual copy-paste shortcuts `Left-CTRL-C` and `Left-CTRL-V`).[3]

- F9 executes the currently highlighted code.
- `LEFT-CTRL + <RETURN>` executes the current cell (menu entry `Run -> Run cell`). A cell is defined as the code between two lines which start with the agreed tag `#%%`.
- `SHIFT + <RETURN>` executes the current cell and advances the cursor to the next cell (menu entry `Run -> Run cell and advance`).
- `TAB` tries to complete the word/command you have just typed

And here are some useful ipython "magic" commands that you can use in the ipython console on the right (but not in the scripts). Most of these are identical to what you would use in the terminal.

- change to a new directory
  `cd some_dir/another_subdir`
- change back 2 levels up
  `cd ../..`
- print current working directory
  `pwd`
- edit a file
  `edit FBP.par`
- list files in current directory
  `ls *.hs`
- Running system commands from the ipython prompt can be done via an exclamation mark
  `!FBP2D FBP.par`
- Get rid of everything in memory
  `%reset`

One thing which might surprise you that in Python *indentation is important*[4]. You would write for instance

```
for z in range(0,image.shape[0]):
   plt.figure()
   plt.imshow(image[z,:,:])
# now do something else
```

The Spyder Integrated Development Environment (IDE) has of course lots of parameters which you can tune to your liking. The main setting that you might want to change is if the graphics are generated “inline” in the iPython console, or as separate windows. Go to Tools > Preferences > iPython console > Graphics > Graphics backend. Change from “inline” to “automatic” if you prefer the separate windows.

Here is some suggested material on Python (ordered from easy to quite time-consuming).

- The official Python tutorial. Just read Section 1, 3, a bit of 4 and a tiny bit of 6.
  https://docs.python.org/2/tutorial/
- Examples for matplotlib, the python module that allows you to make plots almost like in MATLAB
  https://github.com/patvarilly/dihub-python-for-data-scientists-2015/blob/master/notebooks/02_Matplotlib.ipynb
- You could read bits and pieces of Python the Hard Way
  http://learnpythonthehardway.org/book/index.html

[3] Text from http://www.southampton.ac.uk/~fangohr/blog/spyder-the-python-ide.html

[4] The amount of indentation is not important, as long as you are consistent (i.e. it doesn’t matter if you use 2 or 4 spaces, but you cannot mix them).

- Google has an online class on Python for those who know some programming. This goes quite in depth and covers 2 days. https://developers.google.com/edu/python/?csw=1

Image display outside of Python

Several display programs can be used. AMIDE reads the interfile volumes directly. ImageJ and others can use import of raw floats (i.e. the .v file).Settings are for instance.
Image type: 32-bit Real
    Width ?
    Height: ?
    Offset: 0
    Number of images ?
    Gap between images: 0
    White is 0: Ticked
    Little endian: Ticked
You will have to find the data sizes from the header (the .hv file), or by using list_image_info.

STIR commands for evaluation

These are a few STIR commands that can be used on the command prompt (not Python). You normally don't need to if you use the Python scripts though.
For nearly all of these, you can just type the command name without arguments for a usage message.

Basic information about geometry

```
list_projdata_info projdata.hs
list_image_info image.hv
```

Image reconstruction

```
OSMAPOSL someParameterFile.par
FBP2D somePparameterFile.par
OSSPS someParameterFile.par
```

These are the STIR executables used for reconstruction. In the text above, they are called by the shells scripts.

Profile extraction

```
list_image_values prof.txt input_image \
     min_plane max_plane  min_row max_row min_col max_col
```

(note: the backslash "\" is used in shell scripts for "line continuation", i.e. when everything does not fit on one line)

list_image_values writes values to a text file (for import in Excel et al).
Indices need to be in the STIR convention (plane starts from 0, col,row are centred around 0). Use list_image_info to find ranges.

Note: there is currently a bug in list_image_values that row (x) and column (y) have to be given in that order (i.e. it's z,x,y while should have been z,y,x)

Conversion of projection data to an image for display

```
extract_segments projdata.hs
```

convert projection data into an (Interfile) image e.g. for display via AMIDE (as no standard display program reads in sinogram data).