STIR

Software for Tomographic Image Reconstruction

User's Guide Version 3.0

K. Thielemans
Hammersmith Imanet Ltd; Algorithms and Software Consulting Ltd; University College London
Ch. Tsoumpas
Hammersmith Imanet Ltd; Imperial College London; King's College London
D. Sauge, C. Labbé, C. Morel
Hopital Cantonal Genève
M. Jacobson
Technion University
A. Zverovich
Brunel University
T. Beisel
Univ. of Paderborn
C. Falcón
Univ. of Barcelona

Contents

1 Introduction

The objective of this document is to give practical information about the use of the object-oriented library for 3D Reconstruction of PET and SPECT data, called *STIR*. The most recent version of this document (and the library) can be found on
`http://stir.sourceforge.net`.

This library was originally developed by the PARAPET project (funded by the European Union during 1997-2000), extended by Hammersmith Imanet and made into an Open Source project. The current library has different license restrictions than the original PARAPET distribution. Details on licensing are to be found during the registration process, and the file **STIR/LICENSE.txt** that comes with the distribution.

See the publications section of the STIR website for information on which reference to use for STIR. You will want to read [Thi12], see [Fus13] for SPECT additions since STIR 3.0 (for *STIR* 1.x, use [Lab99a], [Lab99b]).

This guide attempts to give end users an overview of all the functionality in *STIR*. It mainly provides procedures for downloading and installing the necessary gnu g++ compiler tools and the *STIR* reconstruction building blocks. Then a brief explanation is given about how to run reconstruction algorithms as well as additional utilities. A description of the reconstruction building block library can be found on our web site (section *documentation*). A short (but out-dated) review of analytic and iterative reconstruction methods is available in [PAR1.3] (available on the STIR web-site).

2 A general note on documentation in STIR

Although we attempt to keep all documentation up-to-date, we recommend to read documentation in the following order:

- general overview documents, such as this User's Guide
- the `STIR Wiki`
- online generated documentation (produced by doxygen). This is produced from comments and (partly) code in the source files.
- check the source itself if in any doubt.

Many questions have already been answered via the mailing lists. See the web-site on how to search these.

Any comments on documentation, and especially contributions are always welcome. Please use the stir-users mailing list.

3 Installation

3.1 Installing source files

Download the source at the *STIR* site:

`http://stir.sourceforge.net`

(Section **Registered Users**).
Download the source files in the zip format. You can then use unzip from
`http://www.info-zip.org/pub/infozip/`. Extract using
> `unzip -a file.zip`

The `-a` option makes sure that files are extracted using the end-of-line convention appropriate for your system. Note that other programs that can handle zip files (such as WinZip) should work as well, although you might have problems with the EOL convention.

Note that you can put the distribution in any location (it does not have to be your home directory).

The result is a **STIR** directory and subdirectories, described in the annex section.

3.2 Installing external software

STIR relies on some external software and a few external libraries which enable certain functionality. On most systems, you should be able to get these using your package manager. Please check the Wiki for most up-to-date information.

3.2.1 BOOST

The only required external library is the well-respected *boost* library. If it is not installed on your system, you can download it from `http://www.boost.org`. Currently *STIR* only uses the include files from *boost* (so you do not need to build the boost libraries). However, you will need to tell your compiler where to find the `boost/` directory with all the include files (see section 3.3.1).

3.2.2 C++ Compiler

In order to compile and run the *STIR* programs, you will need a compiler and associated tools. These days, any compiler should work. See the `STIR Wiki` for more specific information of which compilers we have tried. We would love to hear from any attempts of using another compiler.

3.2.3 Enabling ECAT 7 support

Older CTI (Siemens) scanners use a file format called ECAT™. At present, *STIR* uses parts of the Louvain la Neuve ecat library (called LLN in the rest of this document).[1] The library might still be available on `ftp://ftp.topo.ucl.ac.be/pub/ecat`. It is also available for GATE users.

[1] *STIR* versions 1.? came with specific files for ECAT6 support without the need for the LLN library. However, due to license restrictions this is now no longer the case.

You have to download that library and issue 'make –f Makefile.unix' (or 'make –f Makefile.cygwin' if you are using CYGWIN on Windows) in the new ecat directory first. Please get a recent version of this library (dated 20 July 2004 or later) as Merence Sibomana has introduced various bug fixes, some of which solve problems that you would otherwise experience when using *STIR* and ECAT7 files.

3.2.4 Enabling GE RDF and VOLPET support

We have software that reads VOLPET and RDF data for all GE scanners, but this can currently not be distributed due to licensing restrictions. If you have the GE PET Toolbox proprietary package you can ask Kris Thielemans for VOLPET and RDF support in *STIR*.

3.3 Compilation

3.3.1 Using CMake

Since version 2.2, *STIR* contains files to build using the platform-independent `CMake`. This is the recommended way to build STIR as it is easier for configuration and finding system dependencies.

Normal build procedure on Linux/Unix would be something like[2]

```
> cd /somewhere/nice/STIRbuild/Release
> cmake-gui /somewhere/else/STIR&
```

This will bring up a basic interface where you might to have to choose your prefered build system (i.e. `make` on Linux/Unix) and do an initial configuration. Then you specify build variables. You can press `c` to do additional configurations, `g` for generating your build files, `q` to quit. After this step, you will have to run your build system (e.g. on Linux etc, type `make`, then `make test`, then `make install`). More information on using CMake is on the STIR Wiki.

For example, if CMake cannot find *boost*, you will have to set `BOOST_ROOT`.

GRAPHICS	Possible values: X, PGM, MATHLINK, NONE. X (default) uses basic X windows graphics, PGM writes the graphics to a .pgm file, MATHLINK uses the (external) MathLink library which could be used to send data directly to Mathematica, NONE switches off all graphics. See also section 4.14.1.
STIR_MPI	Toggles between ON, OFF. Enable parallel processing using MPI. *STIR* contains code for running `OSMAPOSL` and `OSSPS` in parallel-mode. See section 4.3 for information on how to run programs using MPI. [3].[4]
STIR_OPENMP	Toggles between ON, OFF. Enable threaded processing using OPENMP. [5] See section 4.4 for information on how to run programs using OPENMP.
STIR_LOCAL	Specify location of directy with your own extensions to STIR. See the developer's guide.

You can also (optionally) specify locations of some external libraries for additional IO capabilities of STIR.

- LLN files for ECAT support via `LLN_INCLUDE_DIRS` and `LLN_LIBRARIES`.

[2] Point `CMake` to the main **STIR** directory, not **STIR/src**.

[3] In fact, any algorithm that uses `PoissonLogLikelihoodWithLinearModelForMeanAndProjData`. This functionality was contributed mainly by Tobias Beisel, Univ. of Paderborn). You need a version of the MPI library to get this to work. We have tested this using `OpenMPI` 1.4.2 and 1.4.5

[4] When using MPI, you can set a compiler define `STIR_MPI_TIMINGS=1` to enable additional timings for all send/receive pairs (see **distributed_functions.h** in **include/stir/recon_buildblock**. This should only be used for testing purposes as it can slow down STIR dramatically.

[5] Since version 2.0, *STIR* contains preliminary code for running `FBP2D` in threaded mode (contribution by Tobias Beisel, Univ. of Paderborn). You need a compiler that supports `OPENMP`. Note however, that this code doesn't presently result in a decent speed-up of `FBP2D`.

- AVW™ (a commercial library[6] via `AVW_ROOT_DIR`. See section 4.12.2.4 for usage.
- ITK_DIR, use IO from `ITK`, a large open source library. Specifying this enables NRRD, MetaIO and Nifti IO.
- GE RDF™ support via `RDF_INCLUDE_DIRS` and `RDF_LIBRARIES` (requires the GE proprietary PET Toolbox).

Note that you can use e.g. *DISABLE_LLN_MATRIX* to *not* use the LLN ECAT library, even if cmake found it.

Finally, do not forget to set

- CMAKE_BUILD_TYPE to `Release`, `Debug` or `RelWithDebug`,
- CMAKE_INSTALL_PREFIX to specify where the final installation has to occur. CMake provides normally a default in a system location, but you could set it for instance to `./install` to put the files inside your build directory.

On Windows or MacOSX, the procedure is essentially the same, except that you will likely have to specify more locations When using Visual Studio, XCode or other IDEs as your build system, you do not need to specify the Build Type, as you will be able to select that in your IDE.

3.3.2 Operating system specifics

3.3.2.1 Mac OS

The lln library does not compile on Mac OS.

3.3.2.2 All Unix/Linux flavours

If you want to use the X windows display routines, CMake should work out-of-the-box if you have suitable libraries installed, see the wiki for required packages.

X development libraries

If you experience compilation or linking problems mentioning X11, you have to check if the X development libraries are installed on your system. You can check this by doing
> `find /usr -name Xlib.h -print`
If this file is not found, you'll have to install these libraries somehow.

X display depth

STIR version 1.0 required that your Xserver worked in 8bit mode (or Pseudocolor mode in X terminology). This is no longer necessary. However, if you are experiencing problems with the display, you could try 8bit mode anyway.

3.3.2.3 Cygwin on Windows

Cygwin is a must-have if you are using Windows but would like to have nearly everything that Linux/Unix has to offer (at least from a user's point of view). Check out http://cygwin.com.

[6] See `www.mayo.edu/bir/Software/AVW/AVW1.html`.

Using X windows on cygwin

- Install the Xorg-x11 packages using the cygwin setup utility (you need the devel package).
- install ncurses-devel if you don't have it yet (use the cygwin Net setup).
- make sure that /usr/X11R6/bin is in your PATH (necessary for DLLs)

You can start the Xorg X server by executing the startxwin.sh script (you might want to modify this a bit to suit your taste).

Other decent X servers should work as well (Kris Thielemans used Exceed at some point).

3.4 Running tests

We **highly recommend** running test programs after building. There are currently two sets of tests. The first set tests various components of *STIR*. This process is described on the Wiki. The second set of tests is available via the *STIR* web-site, where we we provide test reconstructions as part of the **recon_test_pack**, with an automated procedure to see if you can reproduce the correct results. You really should download this and run it.

Note that if you are using a non-standard scanner, you might want to run one reconstruction on your data with the debug version of the reconstruction program, to see if everything works as intended. You'll probably want to choose a small value for the 'maximum absolute segment number to process' parameter in your reconstruction (see Section 4.8.1), as this will be ***very*** slow.

4 Running *STIR* programs

Here we describe:

- documentation and program conventions
- supported file formats
- rebinning algorithms
- reconstruction programs
- utility programs
- user-selectable components
- display properties

STIR is highly configurable and modular. It is therefore difficult to give a complete (and readable) description of all the options in this guide. Moreover, many options are common to many programs. For instance, a forward projector will be needed in many places, and there are different forward projectors implemented in *STIR*, each with their own set of parameters. Your best bet might be to start with some of the sample parameter files, and then come back to this guide. In particular, section 4.13 on user-selectable components gives more detail on what options are available and what their parameters are.

4.1 Conventions

When discussing command line parameters of the *STIR* executables, the following format is used in this documentation (and the usage messages):

```
> executable_name parameter1 parameter2 \
[optional_parameter3 [optional_parameter4 etc]]
```

This means that the first 2 parameters are mandatory, but a 3^{rd} parameter can be given, or 4 parameters. A single parameter is usually given as one word, but sometimes as a string between `< >`.

All parameters have to be on a single command line. This is indicated by the backslash \, as in Unix this is the standard way of continuing a command line on the next line.

All executables are supposed to be located in the path of the command shell used.

Most *STIR* programs accept a single parameter on the command line, which is sometimes optional

```
> executable_name [parameter_filename]
```

The parameter file is a text file which uses an Interfile-like syntax. It is composed of keywords, corresponding to the names of the various parameters, with the values entered next to them. Spaces and tabs are normally irrelevant. Parameters omitted from the parameter file are assigned a default value. Comments are indicated via a semi-colon (;) at the start of the line. Do not put a comment before the first keyword in the file.

If a parameter file is not passed to the executable, some programs will prompt the user for the required information. For questions that ask for a number, the format is as follows:

What number do you want to enter today
[minimum, maximum, D: default]:

If a simple Carriage Return is entered, the default value is selected.

Sample parameter files for particular programs (or part of a parameter file relevant to a particular common component) are presented in subsequent sections. See also the files in **STIR/samples**.

4.2 Error handling etc

STIR executables tend to write a lot of information to the terminal on what is happening. Check for lines starting with `WARNING` and `ERROR`. After an error, the executable will stop[7].Information on some of the common causes for errors is on the Wiki.

All *STIR* executables return a status value to signify success or failure. What value this is depends on your Operating System. On Unix, Linux and Windows variations, success is indicated by a status of 0, and failure by anything else. There is currently no differentiation between the reasons for failure.

4.3 Running programs using MPI

If you have compiled with STIR_MPI, you need run the executables in such a way that they know about the availables nodes. How to do this is system/MPI version specific. The following should work with OpenMPI on Linux:

```
> mpirun -np 12 --hostfile  /myconf.txt OSMAPOSL mypars.par
```

where the host file describes your set-up, e.g. if you have 3 nodes with differing number of cores, the host file could look like this:

[7] On some systems there is at the end a somewhat confusing error about exceptions thrown, but ignore that and check the ERROR line.

```
# The Hostfile for Open MPI
beo-09 slots=4
beo-10 slots=8
beo-11 slots=8
```

Without host file, the executables will normally be run on the host where you executed the command, which is useful for multi-processor/core machines.

MPICH would use a similar line using `mpiexec`.

If you have a queueing system that supports MPI such as `Torque`, it is better to use that system for sorting out available resources etc. For example, to use 2 nodes with 4 processes each, the following should work

> `qsub -I -l nodes=2:ppn=4`

This will open a interactive prompt on one node where you just type

> `mpirun OSMAPOSL mypars.par`

Of course, you can use `qsub` to submit a job as opposed to getting an interactive prompt.

4.4 Running programs using OPENMP

If you have compiled with OPENMP support, the executables should run as normal. The environment variable `OMP_NUM_THREADS` can be set to the number of processes to be used for OpenMP. Without using this, the number of threads is defined as the number of cores on your system. It is probably useful to reduce the number of threads as currently performance is limited by the available cache in your processor.

4.5 File formats

The STIR utility and reconstruction programs frequently need to read and write files of image and projection data. Files formats are encountered in which data and header information are maintained in separate files (e.g. interfile). In other formats, data files carry header information (e.g. the native GE Advance sinogram format).

When reading a file, *STIR* will automatically discover its file format (independent of its name). See section 4.13.1 for supported output file formats. In addition, utilities are available for converting ECAT6 and ECAT7 to/from interfile (see Section 4.12.2).

4.5.1 Interfile

The most comprehensively supported file format in the library is a newly proposed version of interfile. More details about this type can be found in the "Other info" section of the *STIR* website.

- For PET projection data, we use the proposed Interfile version. You can find sample Interfile headers in the **recon_test_pack** or by using `create_projdata_template`.
- For SPECT projection data, we use Interfile version 3.3 (with a few small changes). You can find a sample SPECT Interfile header in the **samples** directory.
- For images, we use the proposed Interfile version. Currently, images have to be written as PET data, even for SPECT.

Interfile is currently the only format supported for writing projection data (except by the *conv_to_ecat?* utilities*)*. Projection data files are written in pairs:
projdata_filename.hs, projdata_filename.s

where *projdata_filename.hs* is the header text file and *projdata_filename.s* is the data file. Please read section 4.13.1 for info regarding images written by *STIR* in Interfile.

4.5.2 VOLPET sinograms

The GE VOLPET sinogram format is supported for reading, but for the GE Advance only and data files must have a single data set, maximum ring difference 11 and contain 281 bins * 256 segments * 336 views. See section 3.2.4 for enabling support for other scanners and data in the GE RDF format.

4.5.3 ECAT6 and ECAT7 data

See section 3.2.3 for enabling support for ECAT data.

ECAT6 images can be read without conversion. However, only the first frame (ECAT matrix 1,1,1,0,0) will be read.

ECAT7 sinograms, attenuation files and images can be read without conversion. However, only the first frame (ECAT matrix 1,1,1,0,0) will be read for static reconstruction.

Warning The calibration factor field in the main header of ECAT7 images is currently ignored.

For dynamic or gated data, conversion can be used, see 4.12.2.

4.5.4 Image IO using the AVW library

See the `conv_AVW` utility (section 4.12.2.4).

4.5.5 Image IO using the ITK library

If you compiled *STIR* such that it could find the *ITK* library, you will be read all image formats supported by *ITK*, see `the ITK Wiki` for some information. However, many file formats do not specify geometric information (e.g jpg). For those that do, *STIR* currently ignores orientation, offset and timing. Note that as DICOM files often store only a single slice, *STIR* attempts to find other slices belonging to the same series/time frame/gate.

4.5.6 SimSET files

There are some preliminary files to make it easier to use SimSET and *STIR* together in the **SimSET** directory. See the README.txt in that directory for more info.

4.6 List mode processing

Some scanners produce list mode data, which is essentially a list of events. *STIR* provides utilities to use the list mode files, for example to convert them to sinograms. It is also possible to reconstruct images directly from list mode data, although this has not been tested very well yet as of *STIR 2.1*.

Currently supported list mode formats are specific to the ECAT HR+ and ECAT EXACT 3D scanners. There are some unfinished classes available on the *STIR* web-site to read *LMF* format files, in conjunction with the *LMF* library. However, these might be obsolete as the OpenGATE project might be distributing files to enable STIR to read *LMF* format files.

See the *STIR* web-site for an independent project that allows reading *ROOT* files and convert it to *STIR*-compatible Interfile projection data.

4.6.1 lm_to_projdata

This utility can be used to bin (or "sort") the list mode data into 'projection data' (also known as **3D sinograms**. This can then be processed further by other *STIR* utilities. It needs to be run as follows:

> `lm_to_projdata par_filename`

See **STIR/samples/lm_to_projdata.par** for an example file. Please check out the online documentation (as generated by doxygen) for more info.

Currently, each time frame is written in a different file (as Interfile). Gating will be supported in a future version.

4.6.2 lm_to_projdata_bootstrap

As before, but using bootstrapping (with replication). This is useful to generate multiple realisations from a single list mode file to study variance.

You can use it as follows:

> `lm_to_projdata_bootstrap par_filename seed`

The seed is a positive integer (do not use 0). Specifying different seeds will give you different noise realisations. You can use the same parameter file as for `lm_to_projdata`.

4.6.3 lm_fansums

Computes fan-sums, i.e. for each detector, sum all data that are in coincidence with that detector. Results are written as a simple text file. This could be used to check if all detectors are working properly for instance.

4.6.4 list_lm_events

Allows inspecting the events in a list mode file. Run without arguments to see a usage message.

4.7 Rebinning algorithms

In 3D PET, the name *rebinning* is used for the process of manipulating the 3D projection data set to the equivalent of 2D projection data. This reduces the number of segments (see *STIR* Glossary) to 1. Popular rebinning algorithms are SSRB, FORE and variations such as FOREX, and FOREJ. Rebinning is often used before reconstruction to decrease total reconstruction time.

4.7.1 SSRB

The *Single Slice Rebinning* algorithm [Dau87] is the oldest and simplest rebinning algorithm. It essentially ignores the obliqueness of a Line of Response and moves data to the axial position in segment 0 such that z-resolution on the axis of the scanner is preserved.

The *STIR* implementation of SSRB is a generalisation that applies the same idea while still allowing preserving some of the obliqueness. For instance, for a dataset with 9 segments, **SSRB** can produce a new dataset with only 3 segments. This essentially increases the axial compression (or *span* in CTI terminology), see the *STIR* Glossary on axial compression. In addition, **SSRB** can introduce extra *mashing* (see the *STIR* Glossary) of the data, i.e. add views together.

Usage:

```
> SSRB output_filename input_projdata_name \
num_segments_to_combine [num_views_to_combine \
[do_normalisation [max_in_segment_num_to_process ]]]
```

num_segments_to_combine has to be odd. It is used as the number of segments in the original data to combine.

num_views_to_combine has to be at least 1 (which is the default). It is used as the number of views in the original data to combine.

do_normalisation has to be 1 (default) or 0. When it is 1, the result is normalised, i.e. divided by *num_segments_to_combine*num_views_to_combine*. This is appropriate for rebinning data where normalisation has already been applied, but inappropriate otherwise.

max_in_segment_num_to_process defaults to all segments. Can be used to ignore the most oblique segments in the input data. Note that the most oblique segments that cannot be rebinned completely will be ignored automatically. For instance, when the input data has 7 segments, and *num_segments_to_combine* is 3, only 3 segments are used from the input (and 1 output segment produced), as 9 input segments would be necessary to produce 3 'complete' output segments.

4.8 Image reconstruction programs

4.8.1 Iterative algorithms

Two statistical reconstruction algorithms are currently implemented:

OSMAPOSL an implementation of the OSEM-One Step Late algorithm with various additional refinements and capabilities. See [Jac00] for a description of many details of the implementation.

OSSPS an implementation of the Ordered Subsets Paraboloidal Surrogate algorithm by Erdogan and Fessler, and Ahn and Fessler [Ahn2003].

Since *STIR* 2.0, all iterative algorithms use a *generalised objective function*[8] which the algorithm tries to maximise. The result of the algorithm will obviously depends very much on the objective function used.

Currently, only Poisson-based objective functions are implemented in *STIR*. We have an objective function for static projection data (*aka* sinograms), list mode data and parametric images from dynamic projection data (currently only using the Patlak model).

Most parameters are shared between algorithms and objective functions. Therefore, once you understand how to run e.g. OSEM, it should be easy to understand OSSPS etc. This guide therefore describes the case of running OSMAPOSL on projection data in detail first. Other algorithms (and objective functions) are then described as variations on this case.

4.8.1.1 Running OSMAPOSL on static projection data

The program **OSMAPOSL** executes the IF-OSEM-OSL algorithm. This is a generalisation of the One Step Late algorithm [Gre90] but allowing inter-iteration filtering and/or a prior. Note that this algorithm is in general not convergent.

OSMAPOSL allows

- the use of subsets, similar to the OSEM algorithm [Hud94]. OSMAPOSL will only work when the subsets are approximatelly balanced (i.e. the number of subsets has to be a divisor of the

[8] We call them *generalised* because they do not necessarily have to correspond to a function. Most algorithms only require that we can compute some kind of gradient. As an example, the update of OSEM with the Median Root Prior (MRP) does not correspond to the gradient of a function.

number of views). In fact, by default this implementation[9] assumes that the subsensitivity image is proportional to the total sensitivity image, but this can be switched off .
- inter-update filtering (applying a filter to the image before it is multiplied with the update image) [Jac00]
- inter-iteration filtering (applying a filter after the rest of the image update is performed), sometimes called EMS [Sil90], see also [Mus04].
- post-filtering (apply a filter after the last subiteration)
- applying the prior information in an additive or multiplicative way [Mus01]
- random permutation of the order of the subsets in each iteration

See [Jac00] for more information, and also the online documentation for the class **OSMAPOSLReconstruction.**

The successive image iterates of the algorithm are saved at pre-specified subiteration intervals in a sequence of image files. The files are named with a pre-specified output file prefix with the subiteration number appended after an underscore.

As described in Section 4.1, the recommended manner of running **OSMAPOSL** is to pass the executable a parameter file argument which specifies the relevant parameters of the reconstruction. Parameters omitted from the file are assigned a default value. If a parameter file is not used, the program prompts the user for the required information.

The form of a typical parameter file[10] is as follows:

```
OSMAPOSLParameters :=
;lines starting with semicolons are comments
objective function type:= \
  PoissonLogLikelihoodWithLinearModelForMeanAndProjData
PoissonLogLikelihoodWithLinearModelForMeanAndProjData Parameters:=
; input, sensitivity and prior parameters here

input file := projection_data_filename.hs

; use -1 to use the maximum available
maximum absolute segment number to process := 4
zero end planes of segment 0 := 1

; keywords that specify the projectors to be used
Projector pair type := Matrix
 Projector Pair Using Matrix Parameters :=
   ; Use the PET Ray-tracing matrix.
   ;  This needs to be changed to SPECT UB when using SPECT data
   Matrix type := Ray Tracing
     Ray Tracing Matrix Parameters:=
   End Ray Tracing Matrix Parameters:=
 End Projector Pair Using Matrix Parameters :=

; background (e.g. randoms)
additive sinogram := 0

; sensitivity related keywords
 ; time frame info used for dead-time calculation when using ECAT7
 ;time frame definition filename:=
 ;time frame number:= 1

 ; normalisation and attenuation info
 ; Bin Normalisation type:= None
```

[9] In fact, this assumption is part of the `PoissonLogLikelihoodWithLinearModel` class, which is the base class of the objective function that OSMAPOSL uses.

[10] Parameter values used here are for illustration only and are *not* recommended.

```
 recompute sensitivity := 1
 use subset sensitivities:=1 ; recommended
 ; optional filename to store/read the sensitivity image
 ; (if use subset sensitivity is off)
 ;sensitivity filename:=
 ; optional filename to store/read the subsensitivities
 ; use %d where you want the subset-number (a la printf)
 subset sensitivity filenames:= sens_%d.hv

; keywords for specifying the prior information
prior type := None

; next keywords can be used to specify image size, but will be removed
; they are ignored when using an initial image
zoom := 1
; use --1 for default sizes that cover the whole field of view
XY output image size (in pixels) := -1
end PoissonLogLikelihoodWithLinearModelForMeanAndProjData Parameters:=


; set output file format, if omitted a default value will be used
Output file format := Interfile
Interfile Output File Format Parameters :=
; byte order := little-endian
; number format := signed integer
; number of bytes per pixel := 2
End Interfile Output File Format Parameters :=

initial estimate:= initial_image_filename.hv
enforce initial positivity condition:=1
number of subsets:= 6
start at subset:= 0
number of subiterations:= 30
start at subiteration number:=1

output filename prefix := out_file
save estimates at subiteration intervals:= 2
uniformly randomise subset order:= 1


; keywords that specify the filtering that occurs after every subiteration
; warning: do not normally use together with a prior
inter-iteration filter subiteration interval := 4
inter-iteration filter type := Separable Cartesian Metz
 ; keywords below will depend on the filter type (see text)
 separable cartesian metz filter parameters :=
   x-dir filter fwhm (in mm) := 6
   y-dir filter fwhm (in mm) := 6
   z-dir filter fwhm (in mm) := 6
   ; use some sharpening here as example (not really recommended though)
   x-dir filter metz power := 2
   y-dir filter metz power := 2
   z-dir filter metz power := 2
 end separable cartesian metz filter parameters :=


; keywords that specify the filtering that occurs at the end
; of the reconstruction
post-filter type := None

; keywords that specify the filtering that occurs before
; multiplying with the update image
```

```
inter-update filter subiteration interval := 4
; would have to be filled in.
inter-update filter type := None

map model := additive

; keywords for preventing too drastic (multiplicative) updates
; below just set to their default values
maximum relative change := 3.40282e+38
minimum relative change := 0

; enabling this will write the multiplicative update images
; every sub-iteration
write update image := 0

END :=
```

See **STIR/samples/** for example parameter files.

The following gives a brief explanation of the parameters. Where appropriate, the notation *[min, max, default]* is used to specify allowable numeric ranges. Values of ∞ indicate that the largest value possible for the numeric type of the parameter value is allowed.

Objective function related parameters (Poisson statistics)

input file The name of the file for the measured projection data.

zero end planes of segment 0 If this is set to 1, the reconstruction pretends that the projection data measured in the extreme-most rings of the scanner (the end planes of segment 0) are zero. See also the discussion in Section 4.8.1.

time frame definition filename See *Bin Normalisation type*

time frame number Defaults to 1. Currently only used for finding the frame number and hence dead-time (in ECAT7). See *Bin Normalisation type*.

Bin Normalisation type This keyword is used to set both normalisation factors and attenuation correction factors. See section 4.13.8.

use subset sensitivities Defaults to 1 (recommended). If you set this to 0, it is assumed that all subset sensitivities are equal, and therefore proportional to the total sensitivity. We can then save some memory by internally storing only the total sensitivity. However, in many cases and in particular f you have non-uniform attenuation, this assumption will generate some artefacts in the image.

recompute sensitivity Defaults to 0. If *sensitivity filename* or *subset sensitivity filenames* are specified and recomputation is switched off, the sensitivity data will be read from file, and the *Bin Normalisation type* keyword will be effectively ignored. On the other hand, if the filename is set and recomputation is switched on, the (subset) sensitivity will be written to that filename. This parameter might be useful to avoid recomputing the sensitivity if it does not change.

sensitivity filename The sensitivity image is an important ingredient of likelihood-based reconstruction algorithms of PET or SPET emission data [11] For each voxel, it represents the total

[11] In *STIR* 2.0, this is generalised to the sensitivity for any problem with Poisson statistics where the data are linear combinations of the variables of interest.

probability of detection of an event originating in that voxel (up to a global proportionality factor). It is computed by backprojecting projection data with all elements set to 1.

The name of the file containing the sensitivity image. The default is to compute it from the previous keywords. You could set it to 1, which is am image uniformly 1 over all voxels. Using this is generally a bad idea however. It is only somewhat appropriate when using fully precorrected data in 2D PET. Even in that case, you will get an image with the wrong scale factor.

Note that when no attenuation nor normalisation was used, the sensitivity will be computed based on geometrical backprojection only. This only appropriate when reconstructing from pre-corrected projection data.

subset sensitivity filenames This should be used instead of *sensitivity filename* when using subset sensitivities. It should provide a pattern á la `printf`[12] to allow constructing different filenames for each subset.

Projector pair type Specifies the back/forward projector pair to be used in the reconstruction. See Section 4.13.4 for possible values. We recommend using matching projectors (*e.g.* based on a single projection matrix).

additive_sinogram This parameter can be used to take randoms or scatter into account. It is a constant background term that will be added to the forward projected data in the denominator of the MAP-OSL algorithm. Please note that currently the forward projector produces 'geometric' data. This is because the normalisation and attenuation coefficients can be cancelled out between the forward and backprojector. They then remain only in the sensitivity image and the background term. This means that in effect, a normalised and attenuation corrected randoms sinogram would have to be passed as 'additive sinogram'. By suitably adding a term to the input projection data, the Shifted Poisson version of OSL can be run as well.

maximum absolute segment number to process Denoting the input value by *num_segments*, this indicates that the reconstruction will be carried out using segment numbers -*num_segments* through +*num_segments* of the measured projection data. Defaults to use all segments in the input data.

prior type The type of prior information. See Section 4.13.3 for the list of possible values.

zoom The zoom factor. Defaults to 1 which gives x,y voxel size equal to the bin size of the projection data[13]. Larger than 1 means smaller voxel size. This parameter is ignored when an *initial estimate* is specified. *This parameter will be removed in a future version.*

XY output image size (in pixels) Number of pixels to use in x,y direction. Default (-1) covers the whole FOV as determined from the projection data and gives an odd number of pixels. This parameter is ignored when an *initial image* is specified.
This parameter will be removed in a future version.

Algorithm settings

output filename prefix The output filename prefix. When image iterates are saved, subiteration numbers are appended to this prefix after an underscore.

[12] In fact, `boost::format` is used, so the pattern can be more flexible.
[13] This is different from the CTI convention which involves some factors like the number of elements and 128.

output file format type Set output file format, if omitted a default value will be used. Subsequent parameters will depend on the type of output file format. See section 4.13.1.

initial estimate The name of the image file with which the algorithm is initialised. The default is an image which is uniformly 1 over all voxels. The default can be specified by giving the parameter an input of 1. Using an input of 0 will use a uniformly 0 image, but this is inappropriate for OSMAPOSL.

enforce initial positivity condition If this parameter is not set to 0, the program will set all non-positive voxel values in the initial estimate to small positive ones.

number of subsets [1, num_views, 1] The number of subsets. For symmetric balancing among subsets, it is advisable to select a number which divides evenly into *num_views*, or even *num_views/4*, depending on the symmetries used by the projector.

uniformly randomise subset order When set to 1, the order of the subsets for each full iteration is randomly permuted (uniformly).

start at subset [0, num_subsets-1, 0] Specifies with which subset in the ordered subset sequence to start, where the subsets are enumerated from *0* to *num_subsets-1*.

number of subiterations [1,∞, 1] The number of subiterations to run.

start at subiteration number [1,∞, 1] Initializes the subiteration counter to the input value. Useful for resuming reconstructions (for example, the continuity of output image file numbering may be preserved).

save estimates at subiteration intervals [1, num_subiterations,0] Specifies at what intervals (in subiterations) the iterates of the algorithm are saved. The final iteration is always saved.

inter-iteration filter subiteration interval [0, num_subiterations, 0] Specifies at what intervals (in subiterations) inter-iteration filtering is carried out. A value of 0 disables inter-iteration filtering.

inter-iteration filter type The type name of the inter-iteration filter. See Section 4.13.2 for the list of possible values.

post-filter type The type name of the post filter. See Section 4.13.2 for the list of possible values.

inter-update filter subiteration interval [0, num_subiterations, 0] Specifies at what intervals (in subiterations) inter-update filtering is carried out (see [Jac00]). A value of 0 disables inter-update filtering.
(OSMAPOSL specific)

inter-update filter type The type name of the inter-update filter. See Section 4.13.2 for the list of possible values.
(OSMAPOSL specific)

map model : additive|multiplicative The default choice *additive* applies the prior on the image as is, the other choice *multiplicative* applies it essentially on the image times the sensitivity image. See [Mus01] for more details.
(OSMAPOSL specific)

maximum relative change [0, 3.40282e+38, 3.40282e+38] The multiplicative update image will be thresholded from above with this value (at every subiteration except the first) *i.e.*, before multiplying it with the old image to get the new one. The default value does not impose any thresholding (as in strict OSMAPOSL). However, we find that when subsets are used, a value of about 10 is beneficial.
(OSMAPOSL specific)

minimum relative change [0, 3.40282e+38, 0] The multiplicative update image will be thresholded from below with this value (at every subiteration except the first).
(OSMAPOSL specific)

write update image [0, 1, 0] When this is set to 1, OSMAPOSL will write the multiplicative update images every sub-iteration.
(OSMAPOSL specific)

4.8.1.2 Notes and technical issues regarding the selection of parameters

Application of filters:
In addition to the remarks in section 4.13.2 on filtering, one should note the following

(i) Inter-iteration filtering (or inter-update filtering) generally results in resolution that is object dependent and space varying. This is probably not what you want. See [Mus04,Mus02] for more details.

(ii) If a strictly positive Metz power parameter is chosen, a non-trivial Metz filter results whose frequency response possesses an amplifying middle frequency band. In this case, it is potentially hazardous to choose too small a corresponding FWHM parameter. For then, the amplifying mid-band may coincide with the high frequencies of the noise components of the image and hence strengthen these components. Empirically, we have found that good lesion detectability is obtained by IMF-OSEM with Metz powers of 1, together with FWHMs of 40%-75% the relevant dimensions of the lesion (see also [Jac00]).

(iii) All filtering operations in all iterative algorithms currently implemented in the library apply post-thresholding to ensure positivity at each iteration. Future releases may allow users to vary the thresholding rule or de-activate it.

Zeroing end planes of segment 0:

The "zero end planes of segment 0" option is made available to help users of the *STIR* reconstruction software overcome a modelling difficulty created by an awkward image discretisation convention. So that images reconstructed using the *STIR* software can be compared to those of other reconstruction software, the *STIR* library defaults to an image discretisation scheme that has become common among most commercial analytic reconstruction software packages. In this scheme, images are discretised on a voxel grid in which there are 2*num_rings-1 transaxial slices of voxels, num_rings denoting the number of rings in the scanner. The axial extent of each slice is half that of the detector rings and the extreme-most voxel slices are centered about the mid-plane of the extreme-most detector rings. A consequence of this set-up is that the voxel grid does not cover the entire length of the scanner, but rather leaves a gap at each end of the scanner whose axial extent is half the axial length of a detector ring.

We have discovered that this creates a problem for iterative MLE algorithms when reconstructing activity distributions that extend into these "end gaps". For the activity present in the gaps is surely detected by the scanner. However, the restricted extent of the voxel grid imposes a model of the projection data statistics in which the activity in the gaps is zero. As a result, the algorithm assigns an excess of activity to the end planes of the image grid, seemingly because they are the

next most likely origins of the annihilations in the end gaps. In turn, the reconstructed images exhibit excessively bright end planes.

From a theoretical point of view, there is nothing "wrong" with such reconstruction results. The image observed approximates a maximum likelihood estimate consistent with the parametric model assumed. From a practical point of view, however, it is obvious that the parametric model assumed is an inferior one, since it does not reflect the physical reality that activity is present in the end gaps. In turn, poor quantification of the end plane activity results.

We have found that using the "zero end planes of segment 0" option can often remedy this problem. This causes the reconstruction to pretend that the detector pairs in the extreme-most detector rings detected no counts. For the scanners currently supported by the library, we have verified that only the measurements of these detector pairs can be influenced by the end gap activity. By setting this option, one models the acquisition as one where the end ring detector pairs had zero efficiency. This provides a more realistic model for the truncated projection data. Note also that the acquisition software of some scanners (e.g. the Positron HZL/R) automatically discards the end plane data. For the same reasons, it is advisable to use the "zero end planes of segment 0" in these cases as well.

The method of zeroing end planes, unfortunately, does not apply well to scanners whose 2D projection data includes merged cross-planes (e.g. the GE scanners or all recent Siemens scanners). This is because the currently supported projectors yield a relatively poor model of the detection probabilities associated with the end planes of such scanners. Excluding the end detector pairs from the computation degrades the model still further. Consequently, some of the voxel values in the end planes may get very large (i.e. noisy) with increasing iterations. For such scanners, we advise that users forego the "zero end planes of segment 0" option. Doing so still yields very reasonable results in the interior part of the reconstructed image. Moreover, future development of the *STIR* library's suite of projector functions will remedy most of the problems discussed in this section.

Finally, for phantom studies in which the activity is known to lie within the space covered by the voxel grid, the end plane phenomenon does not arise.

4.8.1.3 Extra parameters the MPI version

When running the MPI version of `OSMAPOSL` on distributed memory systems, the following extra parameters can be set in the .par files.

enable distributed caching (default: 0) Enables/disables the caching algorithm to save some communication overhead (according to Tobias Beisel's tests, this makes only makes a difference with very large files).

enable distributed tests (default : 0) Tests to check whether the distributed functions work. This is of no use if you're not developing new code for the parallel version. It could be thrown out of the code at some point.

enable message timings (default : 0) Prints timings of send and receive operations, to see where the time gets lost. Could be used for later optimization.

message timings threshold (default : 1.0) Defines a threshold of the above timings, so that not all messages are timed, but those which consume a serious amount of time.

enable rpc timings (default : 0) Measures the total time and the average slave time spend on RPC_process_related_viewgrams_gradient() function. Gives some indication of how much time you save by calculating in parallel.

See [Bei08] for some info on performance.

4.8.1.4 OSSPS

Algorithm description

OSSPS is a relaxed preconditioned sub-gradient descent algorithm:

$$\lambda^{\text{new}} = \lambda + \zeta D \nabla \Psi$$

with λ the parameters to be estimated, Ψ the objective function, D a dagional matrix (the preconditioner) and ζ an iteration-dependent number called the relaxation parameter (see below).

D depends on Ψ. The data-dependent term in the preconditioner suggested in Ahn and Fessler turns out to be equal to the product of the Hessian of Ψ summed over all columns (while using the data plug-in approach for approximating the Hessian close to the solution). Therefore, this OSSPS implementation uses this prescription in general and can be applied to any objective function implemented in *STIR*.

Note that the exact paraboloidal surrogate algorithm by Erdogan and Fessler is currently not implemented.

The relaxation value for the additive update follows the suggestion from Ahn and Fessler:

$$\zeta = \frac{\alpha}{1 + \gamma n} \tag{1}$$

Note that because of the nature of the preconditioner, the OSSPS algorithm does not perform very well in our experience when there is no "additive sinogram".

Running OSSPS

Most of what has been described above for OSMAPOSL applies verbatim for OSSPS. See **STIR/samples/** for example parameter file(s). The following describes the parameters that are specific to OSSPS:

precomputed denominator if you do not specify this keyword, the data-dependent part of the precomputed denominator will be computed automatically (and saved using `output filename prefix`). You can use this parameter to re-use the file from a previous run.
Note: setting the value to 1 will use an images full of ones (which is not a good idea!)

relaxation parameter α in the formula 1 above

relaxation gamma γ in the formula 1 above

upper bound you can give an upper bound on the image values (the lower bound is always zero). The upper bound defaults to a very large value.

4.8.1.5 Using list mode data as input

Since *STIR* 2.0, you can specify an objective function that takes list mode data as input and uses a projection matrix. This can be used to do a static image reconstruction without any resolution loss etc.

WARNING: The current implementation has not been thoroughly tested. Please use the mailing lists for questions and results.

The following gives the main parameters to use for OSMAPOSL.

```
OSMAPOSLParameters :=
  objective function type:=\
     PoissonLogLikelihoodWithLinearModelForMeanAndListModeDataWithProjMatrixByBin
  PoissonLogLikelihoodWithLinearModelForMeanAndListModeDataWithProjMatrixByBin Parameters:=
    list mode filename:= listmodefile.lm
    time frame definition filename:= timeframes.fdef
    ; time frame to use data for the reconstruction
    time frame number:=1
    ; maximum ring difference (defaults to all ring differences)
    ;max ring difference num to process:=
    ; matrix to use for forward and back projection (defaults to ray tracing)
    ;Matrix type:= using ray tracing
    ; background term
    additive sinogram:=
    sensitivity filename:=filename

    ; other usual objective function parameters

  End PoissonLogLikelihoodWithLinearModelForMeanAndListModeDataWithProjMatrixByBin Parameters:=

  ; normal OSMAPOSL algorithm parameters

End:=
```

Currently, the sensitivity filename needs to be specified as the code to compute it using a bin normalisation type has not yet been copied to the PoissonLogLikelihoodWithLinearModelForMeanAndListModeDataWithProjMatrixByBin class. You could run OSMAPOSL first with projection data without axial compression (and view "mashing") to create the sensitivity filename. Similarly, if you want to use OSSPS, you will need to run the projection data version first to get the "precomputed denominator".

The additive sinogram is as discussed in 4.8.1, but needs to be without axial compression (and view "mashing").

4.8.1.6 Parametric image estimation algorithms

Both OSMAPOSL and OSSPS work have implementations to estimate parametric images from dynamic data, called **POSMAPOSL** and **POSSPS** respectively. Their parameters are identical to the static versions of the algorithms, except that they need a different type of objective function. Currently, the only defined version is where the input is dynamic projection data and the Patlak linear model is used to go from dynamic images to parametric images. See section sec:KineticModels for more information.

Currently, the input and output has to be in ECAT7 format.

Below is an excerpt of a possible parameter for file POSMAPOSL. A complete parameter file can be found in **STIR/samples/**. The only change is the fact that dynamic data form the input, and that a (linear) kinetic model has to be specified.

```
OSMAPOSLParameters :=
  objective function type:=\
     PoissonLogLikelihoodWithLinearKineticModelAndDynamicProjectionData
  PoissonLogLikelihoodWithLinearKineticModelAndDynamicProjectionData Parameters:=

  ; input dynamic projection data with emission
  input file := SOME_DYNAMIC_PROJDATA.S
  ; specify additive (dynamic) projection data to handle randoms or so
  additive sinograms := SCATTER_AND_RANDOMS_ADDED_ECAT7.S
```

```
  ; kinetic model specification
  Kinetic Model type := Patlak Plot
    Patlak Plot Parameters :=
    ;; see the Patlak model description elsewhere
    end Patlak Plot Parameters :=

  ; other parameters such as projectors and sensitivity
  ; identical to the static case

  end PoissonLogLikelihoodWithLinearKineticModelAndDynamicProjectionData:=

  ; normal OSMAPOSL algorithm parameters

End:=
```

4.8.2 Filtered back projection (FBP2D)

This implements SSRB+FBP. Currently, data have to be completely precorrected before-hand (arc-correction will be performed automatically if necessary).
The implementation is careful about the implementation of the ramp-filter to avid problems with the DC component (see RampFilter.cxx for more details).

As described in Section 4.1, the recommended manner of running **FBP2D** is to pass the executable a parameter file argument which specifies the relevant parameters of the reconstruction. Parameters omitted from the file are assigned a default value. If a parameter file is not used, the program prompts the user for the required information.

A sample.par file can be found in the **samples/** directory. Below are the parameters you probably need.

```
fbp2dparameters :=

input file := input.hs
output filename prefix := output

; output image parameters
; zoom defaults to 1
zoom := 1
; image size defaults to whole FOV
xy output image size (in pixels) := 180

; can be used to call SSRB first
; default means:
; if no axial compression, use 3
; otherwise, use 1
;num segments to combine with ssrb := -1

; filter parameters, default to pure ramp
alpha parameter for ramp filter := 1
cut-off for ramp filter (in cycles) := 0.5
```

```
; keywords that specify the filtering that occurs at the end
; of the reconstruction
post-filter type := None

end :=
```

The filter parameters can be used to specify an apodising window with 2 parameters: a cut-off frequency fc and α which specifies the usual Hamming window (although I'm not so sure about the terminology here).

$$(\alpha + (1 - \alpha) * cos(\pi * f/\text{fc}))$$

The actual implementation works differently to overcome problems with defining the ramp in frequency space (with a well-known DC offset as consequence). We therefore compute the ramp*Hanning in "ordinary" space in continuous form, do the sampling there, and then DFT it.

Warning: the current version of the interpolating backprojector, the default backprojector used by FBP2D, has a central artefact on some systems (including Sparc and 64-bit AMD and Intel processors).

4.8.3 3D Reprojection Algorithm (FBP3DRP)

This implements Kinahan and Rogers FBP algorithm with reprojection of the missing data [Kin89]. The implementation is fairly generic and should be able to handle non-standard data (e.g. with less or more sinograms in a segment than you would normally have). Also, the Colsher filter is numerically computed at a finer grid in an initial stage to avoid DC problems.

A sample.par file can be found in the **samples/** directory. See separate documentation on FBP3DRP, the doxygen documentation (or the source) for more info. Similar to FBP2D, an apodizing window can be specified. FBP3DRP defaults to using the "pure" Colsher filter.

Warning: the current version of the interpolating backprojector, the default backprojector used by FBP3DRP, has a central artefact on some systems (including Sparc and 64-bit AMD and Intel processors).

4.9 Scatter Correction

Coincidence events where one or more of the photons are scattered form a major contribution to the data in 3D PET. This scatter background needs to be estimated and taken into account during reconstruction. In *STIR 2.1*, a version of the Single Scatter Simulation (*SSS*) algorithm [Wat96, Wat00, Wat97, Wat04, Wer02, Poe03] has been implemented and evaluated [Tso04]. Polycarpou et al describe in detail how to perform scatter correction using STIR [Pol11].

4.9.1 Data Initialisation

- Apply randoms correction and normalisation on emission data (this will be used to scale the scatter estimate)
- Reconstruct attenuation image (ideally use odd number of voxels for zoom_image step below).
- Use **create_tail_mask_from_ACFs** to estimate the region that will be used for scaling
- Reconstruct emission image without scatter correction

- Construct a "sub-sampled sinogram" template (i.e. same geometrical characteristics but less rings and detectors without arc-correction) as suggested in literature (e.g. in [Tso04]). This template sinogram should have no compression (number of views = half the number of detectors per ring, and span = 1 (see the STIR glossary). Use a number of tangential positions that is (Number of detectors per ring - 1) to cover all possible detector pairs. You can reduce this if your scanner has a narrower FOV, but be careful that you do not make it too small (otherwise you will get strange behaviour at the edge when upsampling the scatter estimate). Because of a limitation in the upsampling routine, it is currently recommended to have ring difference 0 only in this template.
- Construct a "subsampled" attenuation image using `zoom_image` and `stir_math`. STIR SSS will select 1 scatter point per voxel in this attenuation image. This can be used to speed up your calculation. When caching line integrals (which is the default), it also means that the scatter estimation needs less memory.
  Careful: `zoom_image` will scale the voxel values with the product of the zoom-factors in each dimension. You need to undo this scaling for the "subsampled" image. For example

```
# zoom to larger voxels
zoom_image zoomed.hv image_mu.hv 21 .25 0 0 5 .1626
# correct back to mu-values
 stir_math --accumulate  --including-first --times-scalar .1626 \
   --times-scalar .25 --times-scalar .25  zoomed.hv
```

  The exact number of voxels and zoom does not matter too much (as long as you cover the whole object), but the offset parameters to zoom_image **HAVE TO BE ZERO** and the number of voxels (in both input and output image) **HAVE TO BE ODD**[14]

```
> create_tail_mask_from_ACFs --ACF-filename <projdata> \
--ACF-threshold <number (1.01)> \
--output-filename <projdata> \
--safety-margin <0>
```

ACF-filename Attenuation Correction Factors Projection Data Filename

ACF-threshold Value used to threshold ACF and creates the mask, defaults to 1.01

safety-margin This dilates the mask for a number of safety bins in tangential direction

output-filename The file that new projection data will be stored

4.9.2 SSS

The actual scatter estimation process consists of 2 steps:

- Use **estimate_scatter** to find subsampled (single) scatter estimate from current emission image
- Use **upsample_and_fit_single_scatter**

[14] See the Wiki for more information on offset conventions.

These steps are now discussed in more detail.

```
> estimate_scatter scatter.par
```

Example `scatter.par`

```
Scatter Estimation Parameters:=
; threshold below which not to bother putting a scatter point
; in a voxel (in cm^-1)
   attenuation_threshold :=.01
; Place the scatter points in a random position of a voxel or in the centre
   random := 1
; Caching (of the line integrals) speeds up the estimation drastically.
   use_cache := 1
; The resolution of the scanner (at 511 keV) as given in the literature
; given as a ratio (example value is 22%)
   energy_resolution :=.22
; Acquisition settings
   lower_energy_threshold :=350
   upper_energy_threshold :=650
   activity_image_filename := ACTIVITY_IMAGE
   density_image_filename := DENSITY_IMAGE
   density_image_for_scatter_points_filename := LOW_RESOLUTION_DENSITY_IMAGE
   template_proj_data_filename := SUBSAMPLED_PROJECTION_DATA_TEMPLATE
   output_filename_prefix := OUTPUT_PROJECTION_DATA
End Scatter Estimation Parameters:=
```

activity_image current estimate of the activity (or emission) image

density_image mu-map or attenuation image (at 511 keV and in units cm^{-1}), used to compute the line-integrals (in the exponentials) in the Klein-Nishina formula.

density_image_for_scatter_points "subsampled" attenuation image (at 511 keV and in units cm^{-1}), used to select scatter points (and their corresponding mu), as discussed in section 4.9.1

template_proj_data "subsampled sinogram" template, as discussed in section 4.9.1

The reason we use 2 different attenuation images is to be able to investigate if the sampling used for the attenuation in those 2 situations makes any difference. Unfortunately, we didn't complete that investigation so cannot recommend what to use for the original attenuation image (for large uniform objects, it won't make any difference, for objects with small features of high attenuation, it might). As discussed in the literature however, you can safely reduce the number of scatter points and still get good estimates.

```
> upsample_and_fit_single_scatter \
--min-scale-factor <number> \
--max-scale-factor <number> \
```

```
--remove-interleaving <1|0> \
--half-filter-width <number> \
--output-filename <filename> \
--data-to-fit <filename> \
--data-to-scale <filename> \
--weights <filename>
```

min-scale-factor,max-scale-factor : Both min and max scale factors are set such that the scaling does not explode for some reason. STIR's scatter routine is scaled such that scale-factor=1 is appropriate to model single scatter only. For human patients, the presence of multiple scatter and out-of-FOV scatter means that the actual scale factor will be a bit more than 1. How much will depend on your scanner, the size of the patient etc. In most cases it shouldn't be more than 2. Therefore, limiting the scale factor between 0.9 and 2 is probably a good idea.

remove-interleaving : Remove the interleaving in the coarse scatter estimate before upscaling (it is recommended to set this to 1)

half-filter-width : width of filter to apply on estimated scale factors (in axial direction). The filter is a simple boxcar.

output-filename : Scatter projection data at the emission data settings

data-to-fit : Normalised Emission sinogram (corrected for randoms)

data-to-scale : Course scatter sinogram

weights : This is the mask sinogram, or any other weighting factors on projection space that is used to define the region of scaling.

The procedure of scaling involves B-Splines interpolation for the tangential (Cubic), axial (Cubic) and view (Linear) dimensions, while for oblique segments, currently the interpolation is achieved by applying the pseudo-inverse transform of Single Slice Rebinning as described in [Tso05].

4.9.3 Iterative loop (if necessary)

The scatter estimation is based on images that do not include scatter, therefore an iterative procedure is needed such that scatter is eliminated progressively. In the first iteration, scatter (B.2) is overestimated and thus images are overcorrected (B.3) for it. Then, in the second iteration (B.2) scatter is underestimated. The mean value of the two scatter estimates can be used for scatter correction. Two iterations are usually enough to estimate a good approximation of scatter.

- Reconstruct emission image with scatter estimate
  Two options exist:
  - FBP: Correct the scatter estimate for attenuation and subtract from the precorrected emission data. Run FBP with sufficient filtering.

$$\text{Image} = \text{FBP}\left(\text{ACF} * (N * (P - R) - S)\right)$$

  - Iterative (e.g. OSEM or OSSPS): Use the following for the additive sinogram (A).

$$A = (R * N + S) * (\text{ACF})$$

where

N Normalization Factors,

ACF Attenuation Correction Factors,

R Random Projection Data,

S Scatter Projection Data as estimated using *STIR*,

P Measured Projection Data (prompts)

4.9.4 Limitations

The three separate executables need to be merged in one together with the iterative loop. The *SSS* can estimate 3D scatter that is measured by indirect rings but no interpolation of indirect rings is currently in place. Therefore, upsampling of oblique sinograms is currently handled in a simple way i.e. (pseudo)-inverse SSRB but ideally interpolation of the 3D sinograms will be useful. In the current version, we have only *SSS* in *STIR* with scaling per sinogram of the scatter sinograms for taking consideration of the out of field of view activity and multiple scattering events. This generally seems to produce good results in most instances [Thi07].

4.10 Parametric Image Construction using kinetic modelling

A parametric image class (**ParametricDiscretisedDensity**) has been implemented that at the moment can hold two parameters[15]. In this release *Patlak plot* [Pat83, Pat85], i.e. a linear kinetic model for tracers with irreversible kinetic behavior, has been implemented. The **PatlakPlot** class is derived from the **KineticModel** class.

The user will need to provide information about:

- File of input function (e.g. plasma activity or reference tissue activity values);
- Calibration Factor (i.e. scale factor that sets the projection data to the same units with the input function)
- Starting frame to apply *Patlak Plot*
- Delay between input function and PET measurement
- Time-frame definition file.
- The boolean *In total counts* is 0 or 1 depending if the counts are total or mean over each frame
- In correct scale should be set to 1 if *STIR* has already scaled correctly the images according to *x_voxel_size*

Example Sample File: `PatlakPlot.par`

```
Patlak Plot Parameters:=

time frame definition filename :=
starting frame := UNSIGNEDINT
calibration factor := FLOAT
scale factor := FLOAT
```

[15]Most relevant *STIR* classes are templated in the number of parameters and would therefore work with numbers. There are some preprocessor defines however to shorten the code in a few places. Currently this uses `#DEFINE NUM_PARAMS 2`.

```
blood data filename := plasma.if
; In seconds
Time Shift := FLOAT
In total counts := 1
In correct scale := 1

end Patlak Plot Parameters:=
```

Current assumptions are that an *F-18* tracer is used; the input function is not decay corrected and it is given in a text file with three columns (i.e. **PlasmaSample**): time in *seconds*, plasma-activity in *kBq/cm-3*, and blood-activity in *kBq/cm-3*) and multiple the total number of **PlasmaSample** is usually given in the very first row. Currently, the parametric image has to be in **ECAT7** format with **num_frames** being equal to **num_params**.

How to Run *Patlak Plot*:

```
> apply_patlak_to_images \
output-parametric-image input-dynamic-image patlak-plot.par
```

How to Estimate Dynamic Images using *Patlak Plot* Parametric Images:

```
> get_dynamic_images_from_parametric_images \
output-dynamic-image input-parametric-image patlak-plot.par
```

Of course the reference tissue model can also be applied:

```
Patlak Plot Parameters:=

time frame definition filename :=
starting frame := UNSIGNEDINT
calibration factor := 1
scale factor := 1
blood data filename := tac.roi
; In seconds
Time Shift := 0
In total counts := 1
In correct scale := 1

end Patlak Plot Parameters:=
```

Important Note: The current version of indirect reconstruction does not consider any special weighting for the time frames. This is a good approximation only if the counts at each frame is very similar which is mostly true for frames with the same duration. If the user needs to use *weighted least-squares*, the code will need minor modification.

Moreover, two direct reconstruction version of linear kinetic models have been also implemented, evaluated [Tso08, Tso07] and optimised [Ang11]. These are the parametric versions of **OSEM** and **OSSPS** therefore called **POSEM** and **POSSPS**. P

```
> POSMAPOSL POSEM.par
> POSSPS POSSPS.par
```

The parameter files include the information of the kinetic model as described for the indirect case and the declaration of the new objective function type `PoissonLogLikelihoodWithLinearKineticModelAndDynamicProjectionData`.

Important Note: *Patlak plot* Parameters are highly correlated thus **POSEM** and **POSSPS** will struggle to compute the parameters [Tso07]. However, for (non-negative) kinetic models that correlation is minimum it is expected that they will work fine.

4.11 Motion Correction

Motion correction has become an important task in PET imaging. Amongst several approaches to motion-correct PET data, two are the most common: reconstruct-transform-average (RTA) [Kle96] and motion-compensated image reconstruction (MCIR) [Li06]. In RTA, separate images are reconstructed for each motion "state" (or "frame"), which are then transformed to one reference frame and averaged to produce a motion-corrected image. In MCIR, the projection data from all frames are reconstructed together by including the motion information into the reconstruction system matrix, so that a motion-corrected image is produced directly. In both cases it is assumed that an accurate description of the motion is available.

In STIR, the motion compensation can happen either before or during reconstruction. Two research papers have validated the implementation of RTA and MCIR in STIR [Pol12], [Tso13]. As these are based on OSMAPOSL their names are RTA-OSMAPOSL and MCIR-OSMAPOSL. The implementation has been validated only for the additive median-root-prior (MRP), but it is compatible with any MAP reconstruction of STIR. Furthermore, the implementation can also work for OSSPS **but the current version of MCIR-OSSPS needs further debugging**.

During RTA, motion correction is performed by first reconstructing independently the raw data of each respiratory position using a conventional iterative algorithm, such as the ordered subsets expectation maximisation (OSEM).Then the reconstructed image of each gate is transformed to the reference position using known motion fields. The transformed gates are then averaged to produce a motion-corrected image. For example, when using MRP:

$$\Lambda_{\nu g}^{(s+1)} = \Lambda_{\nu g}^{(s)} \frac{1}{\sum\limits_{b \in S_l} P_{\nu b} A_{bg} + \beta_g \nabla_{\Lambda_\nu} E_\nu^{(s)}} \sum_{b \in S_l} P_{\nu b} \frac{Y_{bg}}{\sum\limits_{\tilde{\nu}} P_{b\tilde{\nu}} \Lambda_{\tilde{\nu} g}^{(s)} + \frac{B_{bg}}{A_{bg}}} \tag{2}$$

where

$$\beta_g \nabla_{\Lambda_\nu} E_\nu^{(s)} \equiv \beta_g \frac{\Lambda_{\nu g}^{(s)} - M_\nu^{(s)}}{M_\nu^{(s)}} \tag{3}$$

$$\Lambda_\nu = \frac{1}{G} \sum_g \sum_{\nu'} \hat{W}_{\nu' g \to \nu}^{-1} \Lambda_{\nu' g} \tag{4}$$

The transform-and-average operation is performed by the `warp_and_accumulate_gated_images` utility.

For MCIR the motion transformations are incorporated directly into reconstruction. The motion corrected image is reconstructed using the following iterative formula which is based on conventional OSEM including motion via the forward / backward transformation operators, as well. Again using the Median Root Prior, we get

$$\begin{aligned} \Lambda_\nu^{(s+1)} \quad &= \Lambda_\nu^{(s)} \frac{1}{\sum\limits_{b \in S_l, g} \sum\limits_{\nu'} \hat{W}_{\nu' g \to \nu}^{-1} P_{\nu' b} A_{bg} + \beta \nabla_{\Lambda_\nu} E_\nu^{(s)}} \\ &\times \sum_{b \in S_l, g} \sum_{\nu'} \left(\hat{W}_{\nu' g \to \nu}^{-1} P_{\nu' b} \frac{Y_{bg}}{\sum\limits_{\tilde{\nu}} P_{b\tilde{\nu}} \sum\limits_{\tilde{\nu}'} \hat{W}_{\tilde{\nu}' \to \tilde{\nu} g} \Lambda_{\tilde{\nu}'}^{(s)} + \frac{B_{bg}}{A_{bg}}} \right) \end{aligned} \tag{5}$$

Notation:

- $\Lambda_\nu^{(s)}$ is the radioactivity distribution at voxel ν and subiteration number s;
- Y_{bg} is the number of coincident photons of each detector pair (bin) b that belongs to the lth subset S and gate g;
- S_l corresponds to the lth subset of the projection space, which is divided into L total subsets;

- s is the subiteration number and $l = s \bmod L$. A set of L subiterations comprises a full iteration;
- $P_{b\nu}$ is the system projection matrix;
- $\hat{W}$, $\hat{W}^{-1}$ represent the forward / backward warping operations of the image that move the activity from voxel ν' to the voxel ν using the motion fields and linear interpolation;
- E is the "potential" function;
- M_ν corresponds to the median $3 \times 3 \times 3$ mask width of neighbourhood voxels centred at voxel ν;
- G is the total number of gates;
- β, β_g are the penalisation factors for MCIR and RTA, respectively. Note that $\beta = G \times \beta_g$, but for simplicity all cases are displayed with respect to β_g;
- A_{bg} and B_{bg} are the attenuation coeffecient and background term (e.g. scatter) for each bin and gate, respectively.

Both RTA-OS-MAP-OSL and MCIR-OS-MAP-OSL are using the same routines for warping each respiratory gate to the reference gate. However, MCIR requires two motion fields: the forward ($\hat{W}$) motion fields and the backward ($\hat{W}^{-1}$) motion fields (i.e. the same as used in RTA). The forward operator practically warps all frames to the reference frame, while the backward operator 'unwarps' the reference frame to each frame so that they can be compared with the corresponding projection gates in the numerator. It is important that the forward and backward projectors are consistent with each other, if not the result might not converge to a solution. In future we plan to include an alternative option for the backward motion fields to be the transpose matrix of the forward motion fields.

Warning:
The current implementation has been tested for respiratory gating, where each gate has the same number of counts. If this is not the case, or if there is for instance radioactive decay between frames in dynamic PET, the implementation likely needs minor modifications. For instance, the different gate duration could be accounted for by normalising for the time duration [Rah08]. A simple temporary solution is to rescale the normalisation sinogram to take into account time duration, e.g. for two gates with the first gate having 80% of the total duration and the second gate 20%, the normalisation sinogram for the first gate should be divided by a factor 0.8 and the second by 0.2.

4.11.1 Data Preparation

Multiple Files (one for each position): Emission sinogram, Multiplicative corrections (attenuation, normalisation), additive corrections (scatter, randoms), motion vectors, and gate definitions filename.

- Sinograms for each position: These have a standard suffix _g#, e.g. *sinogram_g1.hs* is the header of the position 1. To read the sinogram you will need also have set the corresponding definition file for the positions, e.g. *sinogram.gdef*
- Images: These have a similar suffix _g#, e.g. *image_g1.hv* is the header of the position 1. To read the image you will need also to set the corresponding definition file for the positions, e.g. *image.gdef*

- Motion Vectors: These have a standard suffix _g#d%, e.g. *motion_g1d1.hv* is the header of the motion corresponding to the position 1 and the first direction (i.e. axial according to STIR coordinate system); *d2* corresponds to the vertical axis direction and *d3* to the horizontal axis direction. To read the files you will need also to have set the corresponding definition file for the positions, e.g. *motion.gdef*. The image has exactly the same characteristics as the reconstructed PET image. This currently creates a minor burden as the final voxel sizes and the number of voxels have to predefined on the motion vector images. We hope to change this in future releases.

4.11.1.1 Coordinate system for motion vectors:

Although motion is designed for general motion information such as rigid motion, affine etc, currently only MotionVectors on a Cartesian grid are implemented. Information is stored in millimeter using the normal STIR coordinate system and axes (see the developer's guide).

Important STIR (as many other programs) uses a *pull interpolation* warping, i.e. for each voxel in the target (i.e. motion-corrected) image, the motion vector at that voxel is added to its coordinates, the corresponding location (i.e. before motion correction) in the original image is found and the new value is obtained by interpolating between the values of the surrounding voxels of the original image.

One way to understand this is as follows: if a point-source is in a voxel at location $\vec{r}$ in the motion-corrected image and the motion field at that voxel is $\vec{m}_1$ for gate 1, then this means that the point source was at location $\vec{r} + \vec{m}_1$ in gate 1.

A nice illustration can be found (at the time of writing) at
`http://www.bioen.utah.edu/wiki/index.php?title=Geometric_Transformation_and_Interpolation`.

4.11.2 Motion Correction

4.11.2.1 RTA

See section 4.12.7.

4.11.2.2 MCIR

The procedure involves warping operation by the use of linear interpolation (based on B-Splines) in each forward/backward step. Note that the current release assumes no weighting over the respiratory positions but this could be manually included in the multiplicative sinogram.

As the main reconstruction algorithm is exactly the same as for ordinary image reconstruction, we can use the normal OSMAPOSL program, but with a different objective function.

```
> OSMAPOSL OSMAPOSL_with_motion_correction.par
```

An example parameter file would be as follows

```
OSMAPOSLParameters :=
objective function type:= PoissonLogLikelihoodWithLinearModelForMeanAndGatedProjDataWithMotion

PoissonLogLikelihoodWithLinearModelForMeanAndGatedProjDataWithMotion Parameters:=
input filename := INPUT
projector pair type := Matrix
Projector Pair Using Matrix Parameters :=
Matrix type := Ray Tracing
Ray Tracing Matrix Parameters:=
 ; use a slightly better approximation than simple ray tracing
 number of rays in tangential direction to trace for each bin := 10
End Ray Tracing Matrix Parameters:=
```

```
End Projector Pair Using Matrix Parameters :=

use subset sensitivities := 1

; This input is to read the multiplicative factors
; (normalisation*attenuation). The suffix of each file is _g#
normalisation sinograms := ATTENNORMFACTORS

; The input is to read the additive term
; (randoms + scatter). The suffix of each file is _g#
additive sinograms := added_sinos

Gate Definitions filename := MOTION.gdef

; The Motion Vectors are in image file format and their suffix is _g#d%
; where % corresponds to the dimension (1, 2 or 3)
Motion Vectors filename prefix := MOTION
Reverse Motion Vectors filename prefix := INVERTEDMOTION

; here comes the MRP stuff
prior type := FilterRootPrior
FilterRootPrior Parameters :=
 penalisation factor := 1
 ; you can use any image processor here
 ; the next parameters specify a 3x3x3 median
 Filter type := Median
 Median Filter Parameters :=
     mask radius x := 1
     mask radius y := 1
     mask radius z := 1
  End Median Filter Parameters:=
END FilterRootPrior Parameters :=

end PoissonLogLikelihoodWithLinearModelForMeanAndGatedProjectionDataWithMotion Parameters:=

number of subsets:= 23
number of subiterations:= 460
save estimates at subiteration intervals:= 23
output filename prefix := MOTIONCORRECTEDIMAGE

END:=
```

4.11.3 Regularisation and Noise

According to [Tso13], regularisation is generally advised for either RTA or MCIR. Currently, MRP is validated. Further tests on quadratic prior and OSSPS implementation are recommended to the researchers using STIR. Otherwise, two iterations are usually enough to obtain a relatively good image if followed by postfiltering.

4.11.4 Further Extensions for the Future

More robust testing: Currently the tests are performed based on basic tests.

OSSPS: Needs further debugging as it seems the current settings do not reconstruct the motion compensated image.

Scatter Estimation: Assumed to have it already estimated prior to reconstruction.

4.11.5 Realistic Datasets and other info

An extensive database of realistic simulated PET data with motion is available at `http://www.isd.kcl.ac.uk/pet-mri/simulated-data`.

Motion fields of these data have been estimated using a local hierarchical affine registration algorithm developed by Christian Buerger [Bue11]. This independent library is available at `http://www.isd.kcl.ac.uk/internal/hyperimage`. The library is working with GIPL (Guy's Image Processing Lab) File Format and we provide in STIR two utilities two convert them to/from Interfile format (conv_gipl_to_interfile and conv_interfile_to_gipl). Note that special care need to be taken with respect to the orientation the original files have been stored in gipl format.

4.12 Utilities

Programs are given in the **STIR/utilities** directory, that allow the user to display, manipulate and convert **interfile** data, either image or projection data.

Utilities specific to files in ECAT 6 or 7 format or in the **ecat** sub-directory.

4.12.1 Displaying and performing operations on data

The programs are **manip_image**, **stir_write_pgm**, **list_image_info**, **list_image_values** and **find_maxima_in_image** for image data and **manip_projdata**, **display_projdata**, **extract_segments**, **list_projdata_info** for projection data and **stir_math** for both. Run them with the name of the image or projection data file as an argument. For example, for **manip_image**:

> `manip_image file_name.hv`

See also section 4.7.1 for the use of **SSRB** to manipulate projection data.

The program **get_time_frame_info** is *STIR*'s first (small!) step into supporting dynamic data.

Note: The displaying functionality is *really* basic, and only intended for a quick check how your data looks like. For any serious work, use a decent viewer. For example, *AMIDE* is a free viewer that can read *STIR* image data [16]. Alternatively, convert your *STIR* data to another format using (X)medcon [17].

4.12.1.1 list_image_info

A utility that lists information about an image on stdout. This includes number of voxels, physical sizes, bounding boxes, etc. It also prints min/max/sum of the data.

4.12.1.2 stir_write_pgm

A utility that writes a PGM file for a single slice of an image. You can specify min and max threshold, orientation and slice number. PGM files can be displayed for instance via ImageMagick. This utility is mainly useful for shell/batch scripts.

4.12.1.3 manip_image

This program works on two modes. Additionally to the display possibility, the main mode allows to retrieve information about the image:

- the minimum and maximum values; either for a plane or for an entire 3D image.

[16] `http://amide.sourceforge.net`
[17] `http://xmedcon.sourceforge.net`

- the number of counts.
- the voxel value profile; either for an image slice or for any row 1D row through the image.

Within this mode, a truncation (here called trim) of the data can be performed; it results in the circular resetting to zero of pixels at the image edges.

The other mode (math mode) lets one perform arithmetic operations between two images, or between an image and a scalar. The result of each calculation is kept in a math mode buffer which becomes the input to the next calculation. Hence, a sequence of mathematical operations can be carried out on the input image within math mode. The math buffer contents can also be displayed inside the mode.

4.12.1.4 manip_projdata

This utility is the counterpart of manip_image for projection data. A menu is displayed with the following options:

- viewgram-wise / sinogram-wise display.
- computation of minimum and maximum values and total counts.
- arithmetic operations between two projection data arrays or between a projection data array and a scalar.
- binarisation of the sinogram, positive values are set to 1, negative values to 0.
- truncation of the negative values.
- application of a tangential truncating window (e.g. all data for bins greater than a specified distance from the scanner axis may be set to 0).
- application of an axial truncating window to segment 0 (e.g. end plane data may be set to zero).

Each time an operation results in a new projection data array, the user is prompted for the name of an output file, to which the result is written. This output file then automatically becomes the input file to the next selected operation. Similar to manip_image, this allows new projection data files to be generated from a sequence of operations.

Warning: the user must ensure that all input and output projection data arrays in a given operation are read from / written to separate files.

4.12.1.5 display_projdata

This utility should be preferred to **manip_projdata** when the goal is to display the data by view or by segment for a defined segment number (ring difference).

4.12.1.6 list_projdata_info

A utility that lists size info of the projection data on stdout. Optionally it also prints min/max/sum of the data. Use this if you think there is something wrong with how *STIR* reads your projection data.

4.12.1.7 create_projdata_template

This utility is mainly useful to create a template that can then be used for other *STIR* utilities (such as forward_project, lm_to_projdata etc.).
> `create_projdata_template output_filename`

This will ask questions to the user about the scanner, the data size, etc. It will then output new projection data (in Interfile format). However, the binary file will not contain any data.

It currently only supports PET data. For SPECT, you will have to copy the sample Interfile header from the **samples** directory and edit it by hand. You will also need to provide a file with the binary data (but this can be empty if you are only going to use this as a template).

4.12.1.8 extract_segments

This utility extracts projection data by segment into a sequence of 3d image files. It is mainly useful to import segments into external image display/manipulation programs which do not understand 3D-PET projection data, but can read Interfile images.

The user will be asked if the images should correspond to SegmentByView or SegmentBySinogram data. In the first, data are stored as a stack of viewgrams (one for each view), in the second as a stack of sinograms (one for each axial position). (See the *STIR* glossary).

4.12.1.9 stir_math

This is a command line utility for adding or multiplying data and other numerical operations, with a somewhat awkward syntax. Just execute `stir_math` for a usage message.

Since *STIR* 2.1, this utility also supports parametric or dynamic data using the command line switches `--parametric` and `--dynamic`.

Examples

- Adding 3 images

  > `stir_math output in1 in2 in3`

  Sets *output=in1+in2+in3*

- Subtracting 2 images

  > `stir_math --times-scalar -1 output in1 in2`

  Sets *output=in1-in2*

- Sum the square of each images

  > `stir_math --power 2 --including-first output in1 in2`

  Sets $output = in1^2 + in2^2$

- Dividing 2 projection data files

  > `stir_math -s --mult --power -1 output in1 in2`

  Sets *output=in1/in2*

- Dividing 2 projection data files avoiding division by 0 by thresholding the 2^{nd} data-set.

  > `stir_math -s --mult --min-threshold .1 --power -1 \`
  `output in1 in2`

  Sets *output=in1/max(in2,.1)*

- Dividing 2 files, with first file set to the quotient

  > `stir_math --accumulate --mult --power -1 in1 in2`

  Sets *in1=in1/in2*

- Linear combination of 3 files

  > `stir_math --times-scalar 5 --divide-scalar 2.5 output \`
  `in1 in2 in3`

  Sets *output=in1+2*in2+2*in3*

Warning There is no check that the data sizes and other info are compatible and the output will have the largest data size in the input, and the characteristics (like voxel-size or so) are taken from the first input data. Hence, lots of funny effects can happen if data are not compatible.

Warning When `'--accumulate'` is not used, the output file HAS to be different from all the input files.

Warning The result of using non-integral powers on negative numbers is probably system-dependent.

4.12.1.10 generate_image

This program can be used to generate images containing geometric shapes such as cylinders, spheres etc. See the online documentation generated by doxygen for more info.

4.12.1.11 zoom_image

This can be used to reinterpolate images to different voxel sizes and/or dimensions. See the online documentation generated by doxygen for more info.

4.12.1.12 get_time_frame_info

This simple program allows display of time frame info for a given file. The specified file can be an ECAT6 or an ECAT7 file, or a simple text file specifying the number of time frames and their durations. See the class documentation for *TimeFrameDefinitions* for the format of this text file.
Basic Usage:
> `get_time_frame_info filename frame_number`

Using no arguments will give a more extensive usage message showing some options to select which data to print.

4.12.1.13 list_ROI_values

This is a test-release of a program that can be used to find ROI values for an image. See the online documentation generated by doxygen for more info.

4.12.2 Converting data

This program is used to convert CTI ECAT 6 data (either image or projection data) into interfile data. It normally should be run as follows

```
> convecat6_if output_file_name_without_extension \
cti_data_file_name [scanner_name]
```

The optional *scanner_name* can be used to force to a particular scanner (ignoring the system_type in the main header). *scanner_name* has to be recognised by the Scanner class (see STIR/buildblock/Scanner.cxx). Examples are: ECAT 953, ART, ECAT HR+, Advance etc. If the *scanner_name* contains a space, the scanner name has to be surrounded by double quotes when used as a command line argument.

If there are no command line parameters, the user is asked for these parameters instead.

The program asks if all frames should be written or not. If so, all sinograms/images are converted for a fixed 'data' number. For each data set, a suffix is added to the output_filename of the form `_f#g#b#d#` where the # are replaced by the corresponding number of the frame, gate, bed, data.

Warning CTI ECAT 6 files seem to have a peculiarity that frames and gates are numbered from 1, while bed positions are numbered from 0. Similarly, the number of bed positions in the main header seems to be 1 less than the actual number present. This is at least the case for single bed studies. If this is not true for multi-bed studies, the code would have to be adapted.

Warning Most of the data in the ECAT 6 headers is ignored (except dimensions)

Warning Data are multiplied with the subheader.scale_factor, In addition, for emission sinograms, the data are multiplied with subheader.loss_correction_fctr (unless the loss correction factor is < 0, in which case it is assumed to be 1).

Warning Currently, the decay correction factor is ignored

Warning Note that sinogram data have to be 'corner-swapped', see section 4.12.2.4.

4.12.2.1 conv_to_ecat6

This program is used to convert image or projection data into CTI ECAT 6 data (input can be any format currently supported by the library). It normally should be run as follows:

for images:

```
> conv_to_ecat6 [-k] [-i] outputfilename.img \
input_filename1 [input_filename2 ...]  scanner
```

for projection data:

```
> conv_to_ecat6 -s[2] [-k] [-i] outputfilename.scn \
input_filename1 [input_filename2 ...]
```

If there are no command line parameters, the user is asked for the filenames and options instead. Unless the $-i$ option is used, the data will be assigned a frame number in the order that they occur on the command line.

See **STIR/buildblock/Scanner.cxx** for supported scanner names, but examples are "ECAT 953", "ART", "Advance".

Command line options

-s2 This option forces output to 2D sinograms (ignoring higher segments).

-k the existing ECAT6 file will NOT be overwritten, but added to. Any existing data in the ECAT6 file with the same <frame,gate,data,bed> specification will be overwritten.

-i ask for < frame,gate,data,bed> for each dataset

Note that to store projection data in ECAT6, a 3D sinogram cannot be axially compressed (CTI *span*=1).
Warning This utility does *not* corner-swap 3D projection data back to the 'raw' convention, see section 4.12.2.4.

4.12.2.2 conv_to_ecat7

This program is used to convert image or projection data into CTI ECAT7 data (input can be any format currently supported by the library). **conv_to_ecat7** uses the Louvain la Neuve ecat library, see section 3.2.3. This means it will only work on those systems supported by that library. It normally should be run as follows

for images:

```
> conv_to_ecat7 output_ECAT7_name input_filename1 \
[input_filename2 ...]  scanner
```

for emission projection data

```
> conv_to_ecat7 --s output_ECAT7_name input_filename1 \
[input_filename2 ...]
```

for sinogram-attenuation data

```
> conv_to_ecat7 -a output_ECAT7_name orig_filename1 \
[orig_filename2...]
```

If there are no command line parameters, the user is asked for the filenames and options instead. The data will be assigned a frame number in the order that they occur on the command line.

See **buildblock/Scanner.cxx** for supported scanner names, but examples are "ECAT 953", "ART", "Advance".

4.12.2.3 ifheaders_for_ecat7: ECAT7 support for reading

The current release includes some support for making Interfile headers that point towards and ECAT 7 file. This is possible because ECAT7 normally stores the data with single subheaders per frame/gate/bed/data. **ifheaders_for_ecat** uses the Louvain la Neuve ecat library, see section 3.2.3. This means it will only work on those systems supported by that library.
This program writes Interfile headers that 'point into' an ECAT 7 file. That is, the binary data are NOT rewritten. So, the result of this program is a collection of Interfile headers for every data set in the ECAT 7 file. They are called *ecat7_filename_extension_f1g1d0b0.* etc*, indicating which frame, gate, bed, data number the dataset corresponds to. Run the program as follows

```
> ifheaders_for_ecat7 ecat7_filename.extension
```

This only works with some CTI file_types. In particular, it does NOT work with the ECAT6-like file_types, as then there are subheaders 'in' the datasets.
Warning This utility does *not* take corner-swapping of 3D projection data into account, see section 4.12.2.4.
Note that you do not have to use this utility if you want to read only "frame 1, gate 1, data 0, bed 0". In this case, you can pass the ECAT7 file directly to any *STIR* program.
Warning The calibration factor field in the main header of ECAT7 images is currently ignored.

4.12.2.4 ecat_swap_corners

Usage

```
> ecat_swap_corners out_name in_name
```

What does it do?

For some historical reason, CTI scanners store 3D sinograms sometimes in a 'corner-swapped' mode. What happens is that some corners of the positive and negative segments are interchanged. (As a consequence, segment 0 is never affected).

Below is a summary of what Kris Thielemans understood about corner-swapping from various emails with CTI people. However, he might have totally misunderstood this, so beware!

For ECAT6 data, corner-swapped mode occurs ***always*** for data straight from the scanner. However, data which have been normalised using the import_3dscan utility from CTI are already corner-swapped correctly. Unfortunately, there is no field in the ECAT6 header that allows you to find out which mode it is in.

For ECAT7 data, the situation is even more confusing. Data acquired directly in projection data have to be corner-swapped when the acquisition was in 'volume-mode' (i.e. stored by sinograms), but NOT when acquired in 'view-mode' (i.e. stored by view). It seems that bkproj_3D_sun follows this convention by assuming that any ECAT7 projection data stored in 'volume-mode' has to be corner swapped, and when it writes projection data in 'view-mode', it does the corner swapping for you. So, although there is strictly speaking no field in the ECAT7 header concerning corner swapping, it seems that the 'storage mode' field determines the corner swapping as well.

When the data is acquired in listmode, this changes somewhat. Apparently, there is a parameter in the set-up of listmode scans that allows you to put the ACS in 'volume-mode' or 'view-mode'. The resulting listmode files encode the sinogram coordinates then with corner-swapping or without. After the acquisition, the listmode data has then to be binned into projection data. It is then up to the binning program to take this corner-swapping into account. This is easiest to do by generating 'volume-mode' projection data when a 'volume-mode' when the listmode setup was in 'volume-mode', and similar for 'view-mode'.

If this sounds confusing to you, KT would agree. Here seems to be the best thing to do:

Do all acquisitions in 'view-mode', set-up your listmode scan in 'view-mode', bin the data in 'view-mode'. Forget about corner-swapping.

If you cannot do this, then this utility will corner-swap the projection data for you.

Who implemented this and how was it tested?

The actual corner swapping code was supplied by Christian Michel, based on code by Larry Byars. KT has tested it by performing a very long cylinder scan in 'volume-mode' on the ECAT 966, and looking at the delayeds. The oblique segments had obvious discontinuities in the efficiency patterns. After applying this utility, these discontinuities appeared.

Warning This utility does not (and cannot) check for you if the data has to be corner-swapped or not. So, it can do the wrong thing.

4.12.2.5 copy_ecat7_header

Allows copying header info between ECAT7 files. Check doxygen, or run without parameters for usage info.

4.12.2.6 conv_AVW

If you have the AVW^{TM}library [18] installed on your system, the build process should have built

[18] See `www.mayo.edu/bir/Software/AVW/AVW1.html`.

`utilities/conv_AVW`, see also 3.3.1. This utility allows to use the *AVW* library to read an image, and then write it out using *STIR* as Interfile.
Warning: the *AVW* library seems to do flip some images depending on the file format. For instance, it reads *ECAT7* files using a z-flip compared to *STIR*.

It normally should be run as follows:

```
> conv_AVW [ --flip_z ] imagefile
```

It will require access to a run-time license for *AVW*.

4.12.2.7 conv_GATE_projdata_to_interfile

This program converts GATE raw sinogram output (.ima) into STIR interfile format.

4.12.2.8 conv_gipl_to_interfile and conv_interfile_to_gipl

These programs convert from/to GIPL format. The latter program has a somewhat wrong name. It will actually read an image in any format supported by STIR.

4.12.3 Filtering image data

The **postfilter.cxx** program allows one to apply any available image processor on an input image. [19] Review Section 4.13.2 for info on which filters you can apply.

The program is run as follows:

```
> postfilter [<output filename> [<input file name> \
[<postfilter_par_filename>]]]
```

where the square brackets denote optional parameters (their value will be asked interactively). Example postfilter.par files can be found in the **STIR/samples** directory.

4.12.4 Comparing files

The two utilities **compare_image** and **compare_projdata** can be used to see if 2 files are identical up to rounding errors. (Note that running reconstructions on a different architecture, or even when using different compilers will almost certainly give rounding error differences.) They should be run with 2 command line arguments, specifying the 2 filenames. Optional arguments are as follows:

```
> compare_projdata file1 file1 [maximum_segment_number_to_process]
> compare_image [-r rim_truncation_in_pixels] file1 file1
```

where the `rim_truncation` argument to **compare_image** says how many pixels it should ignore at the radial rim of the image.

4.12.5 Precorrecting (or uncorrecting) projection data

The **correct_projdata** utility located in STIR/**utilities** is useful to perform precorrections such as randoms and/or scatter subtraction, normalisation and attenuation correction. It can also be used to 'uncorrect' the data which might be useful if you get completely precorrected data out from the scanner (or FORE) and want to reverse some of the corrections.

It is run as

```
> correct_projdata correct_projdata_par_filename
```

A sample parameter file is given in **STIR/samples** and is more or less as follows

[19] The current version does no longer allow to compute the impulse response (i.e. point spread function) of a discretised filter. This is largely because the image processors can be non-linear, in which case the PSF concept does not apply. For linear filters, it would still be possible to obtain the PSF by using as input image an image with all 0s except a single pixel in the middle of the image.

```
correct_projdata Parameters :=
input file := trues.hs

; Current way of specifying time frames, pending modifications to
; STIR to read time info directly from the headers

; The specified file can be an ECAT6 or an ECAT7 file, or a simple
; text file. See also section \ref{sec:get_time_frame_info}.
time frame definition filename := frames.fdef

; if a frame definition file is specified, you can say that
; the input data corresponds to a specific time frame
; the number should be between 1 and num\_frames and defaults to 1
; this is currently only used to pass the relevant time
; to the normalisation
time frame number := 1

; output file
; for future compatibility, do not use an extension in the name of
; the output file. It will be added automatically
output filename := precorrected

; default value for next is -1, meaning 'all segments'
; maximum absolute segment number to process :=

; use data (1) or set to one (0) :=

; apply (1) or undo (0) correction :=

; parameters specifying correction factors
; if no value is given, the corresponding correction will
; not be performed

; random coincidences estimate, subtracted before anything else
; is done
; randoms projdata filename := random.hs

; normalisation (or binwise multiplication)
Bin Normalisation type := from projdata
Bin Normalisation From ProjData :=
normalisation projdata filename:= norm.hs
End Bin Normalisation From ProjData:=

; scatter term to be subtracted AFTER norm(+atten correction)
; WARNING This is not the same as the output of the scatter estimation
; defaults to 0
;scatter projdata filename := scatter.hs
END:=
```

Time frame definition is only necessary when the normalisation type uses this time info for dead-time correction (which is not supported yet in the current (public) version of *STIR*).

The following gives a brief explanation of the non-obvious parameters.

use data (1) or set to one (0) Use the data in the input file, or substitute data with all 1's. This is useful to get correction factors only. Its value defaults to 1.

apply (1) or undo (0) correction Precorrect data, or undo precorrection. Its value defaults to 1.

Bin Normalisation type Normalisation (or binwise multiplication, so can contain attenuation factors as well). See Section 4.13.8.

attenuation image filename obsolete Specify the attenuation image, which will be forward projected to get attenuation factors. Has to be in units cm^{-1}.

This parameter will be removed. Use instead a *chained* bin normalisation (section 4.13.8) with a bin normalisation *from attenuation image* (section 4.13.8).

forward_projector type obsolete Forward projector used to estimate attenuation factors, defaults to Ray Tracing. See Section 4.13.5.

This parameter will be removed.

The **calculate_attenuation_coefficients** utility (located in STIR/**utilities**) can also convert an attenuation image to attenuation (correction) factors. It is less flexible than **correct_projdata** though. For example, using the `--ACF` switch, it will compute the attenuation correction factors (ACFs)[20], which can then be used in other STIR programs.

The **attenuation_coefficients_to_projections** utility (located in STIR/**utilities**) is a (simplistic) example to convert attenuation (correction) coefficients to projections by taking the logarithm. For real-world use, it is recommended to pre-filter the data first, to reduce noise and remove negatives.

4.12.6 Generating Poisson noise

For simulation purposes, it is often useful to be able to generate multiple noise realisations given the 'true' mean projection data. For PET and SPECT, the appropriate statistics is very closely Poisson[21], at least for uncorrected counts. *STIR* includes the following utility to find a noise realisation given the mean projection data.

Usage:

```
> poisson_noise [-p | --preserve-mean] output_filename \
mean_projdata_filename scaling_factor seed-unsigned-int
```

The *scaling_factor* is used to multiply the input data before generating the Poisson random number. This means that a *scaling_factor* larger than 1 will result in less noisy data.

The seed value for the random number generator has to be strictly positive. Passing different seeds will result in different noise realisations.

Without the -p option, the mean of the output data will be equal to *scaling_factor*mean_of_input*, otherwise it will be equal to *mean_of_input*.

The options -p and –preserve-mean are identical.

[20] roughly `log(-ray_tracing(image))`

[21] For a given scan, the actual distribution of the detected raw counts is binomial. However, for acquisition times that are shorter than the half life of the radio-isotope, the Poisson approximation is very good.

4.12.7 Motion related utilities

4.12.7.1 warp_image

This program warps an image to another position using as input given motion vectors

4.12.7.2 warp_and_accumulate_gated_images

```
> warp_and_accumulate_gated_images \
output_filename filename_prefix motion_vectors_prefix
```

filename prefix The images that need to be placed in the same reference position.

motion vectors prefix motion vector images

The procedure involves a warping operation by the use of linear interpolation (based on B-Splines). Note that the current release assumes no weighting over the respiratory positions, but this could be manually included if the images are scaled according to duration of each position prior to the correction.

4.12.7.3 zeropad_planes

This program zero pads the start and end planes of an image.

4.12.7.4 shift_image_origin

This utility can be used to simply change the origin in the interfile header of an image.

4.12.7.5 shift_image

This utility can be used to apply translations to an image. The translations are applied with no interpolation, but the entire image moves for an given integer number of voxels.

4.12.8 Using projectors

4.12.8.1 Utilities for forward or back-projection

forward_project and **back_project** will perform a single projection given input data and a "template" for the output. The template is used to know what data sizes and characteristics to use. Both utilities optionally allow passing a parameter file with settings which projector to use (see 4.13.5 for forward projection and 4.13.6 for back-projection). Run these utilities for a usage message.

4.12.8.2 Utilities for testing

The **fwdtest** and **bcktest** programs located in **STIR/recon_test/** can be used to examine forward or back projectors. They allow to project only subregions of the data, but can also do the whole projection in one go. These programs are really only intended for testing though. Normally, you would use **forward_project** and **back_project**.

The **fwdtest** and **bcktest** programs located in **STIR/recon_test/** can be used to examine forward or back projectors. They allow to project only subregions of the data, but can also do the whole projection in one go. These programs are really only intended for testing though. Normally, you would use **forward_project** and **back_project**.

4.12.9 Interfacing with SimSET

STIR contains some routines to make it easier to reconstruct data from SimSET [Lew98], a popular Monte Carlo simulator for PET and SPECT. This is all preliminary. Check out the `README.txt` in the **STIR/SimSET** directory.

4.13 User-selectable components

The software design of *STIR* has a heavy emphasis on object-oriented programming. For the user, the main benefit of this is that it is possible to select at *run-time* what particular type of e.g. forward projector to use. The following is a list of all available components where this type of run-time selection is currently available. In the doxygen documentation, look at the class documentation for **RegisteredObject**.

For each component, there are different types, each with its own unique name. Each type has its specific set of parameters.

Conventions used in this section

Each component has its own section, and each type has its subsection. The name of the subsection is the unique identifier of the type.

In a parameter file, the selection of the type would look for instance as

```
This Program Parameters:=
some parameter:=
filter type := my preferred filter type
my preferred filter type parameters:= ; REQUIRED keyword
par 1:=
...
end my preferred filter type parameters := ; REQUIRED keyword
another program parameter :=
;etc
end :=
```

Please note that the first and last keywords of a particular type *have* to be included in the parameter file, even if no additional parameters are given.

4.13.1 Available output file formats

STIR can write images in a number of different formats. Currently, a single 3D image is written per file, *i.e.* no multi-frame or gate files yet. The only exceptions to this rule are the **conv_to_ecat6** and **conv_to_ecat7** utilities.

Currently, the default output file format is Interfile (see below), although this could be changed (for most *STIR* programs) by editing the file
include/stir/IO/DefaultOutputFileFormat.h
(not recommended).

4.13.1.1 Common parameters

The following parameters are common to all file formats. However, the implementation for a particular file format could ignore the value of these parameters if it does not support it. These parameters follow the Interfile 3.3 syntax, *except* that the "short float" and "long float" values for "number format" keyword are not supported. Use "float" instead.

byte order <string>
values: littleendian | bigendian
default: native byte order

number format <string>
values: bit | ascii | signed integer | unsigned integer | float;
default: float

number of bytes per pixel <integer>
default: 4

If "byte order" is omitted, the default value corresponds to the native byte order of the computer the program runs on.
Summarising: if none of these keywords is specified, data are written as 4-byte floats in the native byte order, unless the specific file format has another default.

Warning: Note that only essential information is written in the headers. In particular, frame duration etc are currently *not* filled in.

4.13.1.2 Interfile

The most comprehensively supported file format in the library is a newly proposed version of interfile. More details about this type can be found on the *STIR* website. Interfile image files are written as a pair of files *image_filename.hv, image_filename.v*
where *image_filename.hv* is the header text file and *image_filename.v* is the data file. In addition, we currently write a *.ahv* file which uses Interfile 3.3 conventions, with a tweak for the `slice thickness` keyword to work-around an Analyze™ bug[22]. The *.ahv* file is probably also readable by other programs capable of reading Interfile 3.3.

If this output file format is used, and a filename without extension is specified for output, or when the filename has an extension *.hv*, the above naming conventions hold. If a filename with another extension is specified, this name is used for the name of the binary file.

When a file must be specified for reading as a parameter for a *STIR* utility or reconstruction program, the name of the *.hv* header file should be given.

Warning The interfile 3.3 standard does not allow to specify scale factors for the data. Hence, the *.ahv* file has no scale factors. This means that any program that reads the *.ahv* file **will have improperly scaled images**, unless the scale factor is 1. However, when using float output, *STIR* automatically writes data with scale factor equal to 1, so as long as your non-*STIR* program knows about float data, everything will be all right. The newly proposed Interfile standard does use a scale factor, and the *.hv* file follows this convention. However, currently probably only *STIR* programs know about this convention.[23]

Parameters

This file format currently has no extra parameters, except for the start and stop keywords.

```
Interfile Output File Format Parameters :=
; any parameters common to all file formats
End Interfile Output File Format Parameters :=
```

[22] See the comments for write_basic_interfile_image_header() in **STIR/IO/interfile.cxx**

[23] The utilities distributed by the UCL, Louvain-la-Neuve, Belgium on `ftp://ftp.topo.ucl.ac.be/pub/ecat` follow a different convention of reading the scale factor. Currently, *STIR* writes *.hv* files that can be read correctly by those utilities, including the *mediman* image viewer and (x)medcon. Note however that *mediman* cannot read float data.

Currently, data is written always in the native byte order, and only signed short, unsigned short or float data are supported (although this could easily be extended to 1 or 4 byte integers or doubles).

4.13.1.3 ITK

If this output file format is used, *ITK* IO is used, see also 4.5.5. In this case, the actual file format will be determined by the extension of the filename. Therefore, this can be used to write Nifti (.nii), Teem (.nhdr), MetaIO (.mhdr), etc.

A default extension can be set that will be appended if a filename without extension is used.

```
ITK Output File Format Parameters:=
  default extension:= .nhdr ; current default value
End ITK Output File Format Parameters:=
```

4.13.1.4 ECAT6

If this output file format is used, and a filename without extension is specified for output, *.img* will be added to the filename.

Parameters

This file format currently has only 1 extra parameter, except for the start and stop keywords.

```
ECAT6 Output File Format Parameters :=
; any parameters common to all file formats
default scanner name := <string>
End ECAT6 Output File Format Parameters :=
```

The scanner name has to be one of the values listed in **buildblock/Scanner.cxx**, but generally follows the format *ECAT 953* (this is the default). Currently, the value of this keyword is *always* used, even for other images reconstructed from data from other scanners.

Currently, data is written always in the little-endian byte order, and only signed short data are supported.

4.13.1.5 ECAT7

The file format is only available when the ECAT7 support is enabled during compilation, see section 3.2.3.

If this output file format is used, and a filename without extension is specified for output, *.img* will be added to the filename.

Parameters

This file format currently has only 1 extra parameter, except for the start and stop keywords.

```
ECAT7 Output File Format Parameters :=
; any parameters common to all file formats
default scanner name := <string>
End ECAT7 Output File Format Parameters :=
```

The scanner name has to be one of the values listed in **buildblock/Scanner.cxx**, but generally follows the format *ECAT 962* (this is the default). Currently, the value of this keyword is *always* used, even for other images reconstructed from data from other scanners.

Currently, data is written always in the big-endian byte order, and only signed short data are supported.

4.13.2 Available filters or data processors

Reconstruction algorithms and some utilities use filters, or in general data processing algorithm. In *STIR* 1.x the data-type was restricted to images, but now other data-types can in principle be used. Some of the available data processors can work on any data type, but most only work on images.

The type of data processors available to programs is fully extendable (at compile time). How to do this is beyond the scope of this document. Here we only discuss how to specify a particular data processor at run-time, and list the currently available ones. Samples of their parameters should be available in **STIR/samples**.

Each data processor has a unique name associated to it (given as the head of its subsection below). This name has to be used as value of a filter type keyword (or interactive question). Depending on the filter type, different parameters have to be given in the.par file (or will be asked interactively).

In addition, it is possible to specify the name *None*. This is the default for all keywords that ask for the type of the data processor .

See also the online documentation for the class **DataProcessor**.

4.13.2.1 Separable Convolution

This implements spatial non-periodic convolution with a 3D separable filter. The kernel is given in voxel units (*not* in Fourier space).

This filter applies a 1D convolution in all directions (z,y,x) with potentially a different filter kernel for every direction.

When parsing, the filter coefficients are read as a list of numbers for each direction. The following conventions is used:

- A list of 0 length (which is the default) corresponds to no filtering.
- When the list contains an even number of data, a 0 is appended (at the end).
- After this, the central element of the list corresponds to the 0-th element in the kernel, see below.

Convolution is non-periodic. In each direction, the following is applied:

$$\mathrm{out}_i = \sum_j \mathrm{kernelforthisdirection}_j \mathrm{in}_{i-j}$$

Note that for most kernels, the above convention means that the zero-index of the kernel corresponds to the peak in the kernel.

Elements of the input array that are outside its index range are considered to be 0.

Warning There is NO check if the kernel coefficients add up to 1. This is because not all filters need this (e.g. edge enhancing filters).

Parameters

```
Separable Convolution Filter Parameters:=
x-dir filter coefficients:= list_of_numbers
y-dir filter coefficients:= list_of_numbers
z-dir filter coefficients:= list_of_numbers
END Separable Convolution Filter Parameters:=
```

Example input for a low-pass filter in x,y, no filtering in z

```
Separable Convolution Parameters :=
x-dir filter coefficients := {0.25,.5,.25}
y-dir filter coefficients := {0.25,.5,.25}
;z-dir filter coefficients :=
END Separable Convolution Parameters :=
```

4.13.2.2 Separable Cartesian Metz

This is a separable 3D Metz filter for images discretised on a Cartesian voxel grid. These filters are composed of 3 1D filter kernels, one for the x,y and z directions each. Each 1D filter is specified by a Full Width at Half Maximum (FWHM) parameter (in millimetres), Metz power parameter and maximum spatial kernel width.

With each triple of parameters, one can associate a 1D continuous Metz filter kernel whose frequency response *M(f)* is given by the formula

$$M(f) = \frac{1 - (1 - G(f)^2)^{N+1}}{G(f)}$$

Where *G(f)* is the frequency response (*G(0)* =1) of a zero-mean Gaussian distribution function with the associated FWHM and N is the associated Metz power parameter. Note that in the special case N=0, the filter reduces to the Gaussian filter *G(f)*. If a strictly positive Metz power parameter is chosen, a non-trivial Metz filter results whose frequency response possesses an amplifying middle frequency band (see also [Jac00]).

The filtering routines apply the filter by computing discretised versions of the 1D continuous filter kernels in the space domain and convolving with the image in the appropriate direction. The following rules apply to the construction and application of filters:

(i) The value 0.0 mm is permissible for the filter FWHM parameter. Selecting a FWHM of 0.0 mm produces trivial filter kernels, i.e. impulse functions, enabling one to disable the filter in any direction.

(ii) In accordance with sampling theory, the routine that constructs the 1D filter kernels bandlimits the continuous filter to the hypothesized Nyquist frequency of the image (i.e. 1/(2*voxel_size) of the image in the associated direction). Note that this is equivalent to convolving with a sinc function in the space domain and may yield negative kernel values where unexpected (e.g. in a Gaussian filter kernel). In general, when reasonably large FWHMs are selected (a few multiples of the voxel dimension), the bandlimiting has no effect on the filter construction.

(iii) The spatial kernel width can be limited, which will set any other values to 0. If this is not done, the kernel width is determined by the first value which is smaller in absolute value than 10^-6 times the central kernel value.

(iv) Filter kernels computed by the software may contain non-positive values. This is partly because, in general, a 1D continuous Metz filter response function may contain non-positive value and so also will its samples. In addition, non-positive kernel values may arise due to the bandlimiting operations described in (ii).

Consequently, filtered positive images may likewise contain non-positive values.

Parameters

Sample parameters are given below

```
separable cartesian metz filter parameters :=
x-dir filter fwhm (in mm) := 6
y-dir filter fwhm (in mm) := 6
z-dir filter fwhm (in mm) := 6
x-dir filter metz power := 2
y-dir filter metz power := 2
z-dir filter metz power := 2
x-dir maximum kernel size := 129
y-dir maximum kernel size := 129
z-dir maximum kernel size := 31
end separable cartesian metz filter parameters :=
```

An explanation of these parameters is given here for the x-direction (others are obvious extensions)

x-dir filter FWHM (in mm) The Full Width at Half Maximum of the Gaussian filter kernel from which the x direction Metz filter kernel are derived.

x-dir filter Metz power The exponent parameter for the x direction Metz filter kernel.

x-dir maximum kernel size The maximum width of the kernel (in pixels). Prior to version 0.93 of the PARAPET library this was fixed to the number of pixels in the input image (in that direction).

4.13.2.3 Median

This applies a straightforward 3D (or 2D) median filter on the image.

Parameters

Sample parameters are given below

```
Median Filter Parameters :=
mask radius x := 1
mask radius y := 1
mask radius z := 1
End Median Filter Parameters:=
```

A radius of 0 means no filtering in that direction, a radius of 1 means the median will be computed over a mask of 3 pixels, and so on.

4.13.2.4 Truncate To Cylindrical FOV

This image processor will set all voxels to 0 outside a certain radius. TODO this needs updating.

Parameters

Sample parameters are given below

```
Truncate To Cylindrical FOV Parameters:=
; default use x^2 + y^2 < R^2
; if set to 0, will use <=
strictly_less_than_radius:=1
End Truncate To Cylindrical FOV Parameters:=
```

4.13.2.5 Threshold Min To Small Positive Value

This is a generic data processor that will work on any type of data.

Since strict positivity is a preferred property of images in many circumstances, a post-thresholding of the filtered data is sometimes applied to truncate non-positive values. The thresholding rule currently used is:
(1) If the entire filtered data is non-positive, the data is uniformly set to a hard-coded strictly positive parameter SMALL_NUM<<1.
(2) Otherwise, all non-positive entries in the data are set to SMALL_NUM times the minimum strictly positive value.

4.13.2.6 Chained Data Processor

This is a generic data processor that will work on any type of data.

This data processor allows subsequent application of 2 other data processors on the image. It can for example be used to first Metz filter an image and then threshold it.

Parameters

Example parameters are given by

```
Chained Data Processor Parameters :=
Data Processor to apply first := None
Data Processor to apply second := None
END Chained Data Processor Parameters :=
```

Obviously, in normal practice the Data Processor keywords will have values given by any of the listed data processors in this section (including Chained Data Processor again).

4.13.3 Incorporating prior information

Some iterative reconstruction algorithms allow the incorporation of *a priori* information, for instance the One Step Late algorithm (implemented as OSMAPOSL). At the moment, 'generalised' priors are used, where we mean that we need to know only the gradient of the actual (log of the) prior function.
Which priors are available to programs is fully extendable (at compile time). How to do this is beyond the scope of this document. Here we only discuss how to specify a particular prior, and list the currently available ones. Samples of their parameters should be available in **STIR/samples**.

Each prior has a unique name associated to it (given as the head of its subsection below). This name has to be used as value of a prior type keyword (or interactive question). Depending on the prior type, different parameters have to be given in the.par file (or will be asked interactively).

In addition, it is possible to specify the name *None*. This is the default for all prior keywords.

See also the online documentation for the class **GeneralisedPrior**.

Parameters

These are the keywords that can be used for all priors.

```
penalisation factor := <float>
```

where the `penalisation factor` is usually called β in the literature, and is just a global scale factor for the prior.

4.13.3.1 FilterRootPrior

This prior is an extension of the idea first developed for the Median Root Prior [Ale97]. The prior takes any Data Processor (i.e. a filter), and computes the prior 'gradient' as

$$G_v = \lambda_v / F_v - 1$$

where λ_v is the image where to compute the gradient, and F_v is the image obtained by filtering λ.

Note that for nearly all filters, this is not really a prior, as this 'gradient' is *not* the gradient of a function. This can be checked by computing the 'Hessian' (i.e. the partial derivatives of the components of the gradient). For most (interesting) filters, the Hessian will not be symmetric.

The Median Root Prior is obtained by using Median (see 4.13.2) as Data Processor.

Parameters

These are the keywords that can be used in addition to the ones listed in 4.13.3..

```
FilterRootPrior Parameters :=
penalisation factor := 1
; you can use any data processor here
; the next parameters specify a 3x3x3 median
Filter type := Median
  Median Filter Parameters :=
  mask radius x := 1
  mask radius y := 1
  mask radius z := 1
  End Median Filter Parameters:=

END FilterRootPrior Parameters :=
```

4.13.3.2 Quadratic

This implements a quadratic Gibbs prior. The gradient of the prior is computed as follows:

$$g_r = \sum_d r w_{dr} (\lambda_r - \lambda_{r+dr}) * \kappa_r * \kappa_{r+dr}$$

where λ is the image where the gradient is computed and r and dr are indices and the sum is over the neighbourhood where the weights w_{dr} are non-zero.

The κ image can be used to have spatially-varying penalties such as in Jeff Fessler's papers. It should have identical dimensions to the image for which the penalty is computed. If κ is not set, this class will effectively use 1 for all κ's.

By default, a 3x3 or 3x3x3 neigbourhood is used where the weights are set to x-voxel_size divided by the Euclidean distance between the points.

Parameters

These are the keywords that can be used in addition to the ones listed in 4.13.3..

```
Quadratic Prior Parameters:=
; next defaults to 0, set to 1 for 2D inverse Euclidean weights, 0 for 3D
only 2D:= 0
; next can be used to set weights explicitly. Needs to be a 3D array (of floats).
' value of only_2D is ignored
```

```
; following example uses 2D 'nearest neighbour' penalty
; weights:={{{0,1,0},{1,0,1},{0,1,0}}}
; use next parameter to specify an image with penalisation factors (a la Fessler)
; see class documentation for more info
; kappa filename:=
; use next parameter to get gradient images at every subiteration
; see class documentation
gradient filename prefix:=
END Quadratic Prior Parameters:=
```

4.13.4 Selecting different projector pairs

Many algorithms use both a forward and a back projector. The first step to select which one will be used is to say what kind of projector pair you want to use, of which there are only 2 candidates as given below.

Each projector pair has a unique name associated to it (given as the head of its subsection below). This name has to be used as value of a 'projector pair type' keyword (or interactive question). Depending on the projector pair type, different parameters have to be given in the.par file (or will be asked interactively).

Samples of OSMAPOSL reconstruction parameter files selecting different types of projectors should be available in **STIR/samples**.

Warning: for most iterative algorithms, it is recommended to use matching forward and back-projector. This is unfortunately not the default due to historical reasons. It is recommended to replace the default by using a projection matrix as detailed below.

4.13.4.1 Matrix

Both projectors are based on a single projection matrix, as given in Section 4.13.7.

Parameters

```
Projector Pair Using Matrix Parameters:=
Matrix type := some value
; parameters relevant to this type of matrix
End Projector Pair Using Matrix Parameters:=
```

See Section 4.13.7 for possible values for the 'matrix type' keyword.

4.13.4.2 Separate Projectors

Forward and back projectors are completely independent of each other. In some programs this is necessary to handle images or projection data of different sizes.
Even if the projectors are used that both use the same type of projection matrix, that matrix will not share the same cache or memory.

Parameters

```
Projector Pair Using Separate Projectors Parameters:=
Forward projector type:= some value
Back projector type:= some value
End Projector Pair Using Separate Projectors Parameters:=
```

See Section 4.13.5 for possible values for the 'forward projector type' keyword and Section 4.13.6 for the 'back projector type' keyword.

4.13.5 Selecting a forward projector

It is possible to select the forward projector used at run-time, and extend the available ones at compile time. The mechanism is exactly the same as for the ImageProcessor hierarchy.

Each projector has a unique name associated to it (given as the head of its subsection below). This name has to be used as value of a 'forward projector type' keyword (or interactive question). Depending on the projector type, different parameters have to be given in the.par file (or will be asked interactively).

4.13.5.1 Matrix

This forward projector uses a projection matrix to compute its result.

Parameters

```
Forward Projector Using Matrix Parameters:=
Matrix type := some value
End Forward Projector Pair Using Matrix Parameters:=
```

See Section 4.13.7 for possible values for the 'matrix type' keyword.

4.13.5.2 Ray Tracing

This forward projector uses an optimisation of Siddon's algorithm to compute its result. That is, it uses Length of Intersection. As it avoids storing the matrix elements, it is currently faster than using a forward projector using a ray tracing matrix (see Section 4.13.7). The result is identical though (up to rounding errors and possibly the voxels at the edge of the FOV).

See also online documentation for class ForwardProjectorUsingRayTracing.

Parameters

```
Forward Projector Using Ray Tracing Parameters:=
End Forward Projector Using Ray Tracing Parameters:=
```

This projector currently has no user-selectable parameters. Nevertheless, the 2 keywords given above have to follow the 'forward projector type' keyword in a parameter file.

4.13.6 Selecting a back projector

It is possible to select the type of back projector. The mechanism is exactly the same as for the ForwardProjector hierarchy.

4.13.6.1 Matrix

This back projector uses a projection matrix to compute its result.

Parameters

```
Back Projector Using Matrix Parameters:=
Matrix type := some value
Back Forward Projector Pair Using Matrix Parameters:=
```

See Section 4.13.7 for possible values for the 'matrix type' keyword.

4.13.6.2 Interpolation

This back projector uses incremental (piecewise)-linear interpolation to compute its result. See [Egg98] for details. It can *only* handle arc-corrected data.

Warning: The current implementation has problems (noticeable sometimes at 45 and 135 degrees but mostly at the centre of the image) on Sun Sparc, HP and 64-bit AMD and Intel processors (for certain optimisation settings of the C++ compiler).

Parameters

```
Back Projector Using Interpolation Parameters:=
Use piecewise linear interpolation := 1
Use exact Jacobian := 1
End Back Projector Using Interpolation Parameters:=
```

The 'piecewise linear' keyword allows the user to choose between ordinary linear interpolation or piecewise linear interpolation [Thi99] in axial direction. The latter approximates Volume of Intersection for axially uncompressed data. The piecewise linear interpolation is only used when the axial voxel size is half the axial_sampling of the projection data (for the segment in question), otherwise linear interpolation is used anyway.

The 'exact Jacobian' keyword selects if the exact or an approximate version of the Jacobian is used. The approximation consists in taking the value for the central tangential position and is often used in analytic algorithms in the literature.

Even when all default values are used, the start and end keywords given above have to follow the 'back projector type' keyword in a parameter file.

See also online documentation for class BackProjectorByBinUsingInterpolation.

4.13.7 Selecting a projection matrix

It is possible to select these independently at run-time, and extend the available ones at compile time. The mechanism is exactly the same as for the ForwardProjector hierarchy.

Warning: Currently, most projection matrices are implemented for PET. Using them for SPECT data could lead to surprising results.

4.13.7.1 Common parameters to all projection matrices

The following parameters can be used for all types of projection matrices. There default values are indicated below.

```
disable caching:= 0
store only basic bins in cache:=1
```

Here is an explanation of these parameters.

disable caching [0,1,0] Normally the elements of the matrix are stored in memory (or at least parts of it, see next keyword) after the first use. This can be disabled, but this should normally only be done if not enough RAM memory is available such that heavy swapping occurs. The current procedure is an all-or-nothing cache. In the future, it might become possible that an upper memory limit can be given.

store only basic bins in cache [0,1,1] Most projectors use symmetries to reduce the number of elements that need to be computed. For example, if the ring spacing of the scanner is an integer multiple of the voxel size in z-direction, it is only necessary to compute the matrix elements only for 1 ring(pair). When caching is enabled, by default only the independent elements are cached. If you have plenty of RAM memory, you can store all (non-zero) elements. If your system does not start swapping, this will speed-up the computation.

4.13.7.2 Ray Tracing

This projection matrix uses an optimisation of Siddon's algorithm to compute its result. That is, it uses Length of Intersection.
Currently, the LOIs are divided by voxel_size.x(), unless NEWSCALE is #defined during compilation time of ProjMatrixByBinUsingRayTracing.cxx.

It is possible to use multiple LORs in tangential direction. The result will then be the average of the various contributions. Currently all these LORs are parallel. For a very high number of LORs, the result approximates a strip integral (in tangential direction).

If the z voxel size is exactly twice the sampling in axial direction, 2 or 3 LORs are used, to avoid missing voxels.

It is possible to use a cylindrical or cuboid FOV (in the latter case it is going to be square in transaxial direction). In both cases, the FOV is slightly 'inside' the image (i.e. it is about 1 voxel at each side smaller than the maximum possible).

It is possible to reduce the number of symmetries used by this matrix. This might be useful if you have plenty of RAM and want to speed up the calculations. See also the discussion on caching in section 4.13.7.

Warning Care should be taken to select the number of rays in tangential direction such that the sampling is not greater than the x,y voxel sizes.

Warning The current implementation assumes that z voxel size is either smaller than or exactly twice the sampling in axial direction of the segments.

Parameters

The following parameters can be set (default values are indicated):

```
Ray Tracing Matrix Parameters :=
; any parameters appropriate for all matrices
restrict to cylindrical FOV := 1

number of rays in tangential direction to trace for each bin := 1

do symmetry 90degrees min phi := 1
do symmetry 180degrees min phi := 1
End Ray Tracing Matrix Parameters :=
```

It is recommended to use a number of rays that is larger than 1 (e.g. 10) to avoid some discretisation artefacts in the final image.

For the azimuthal angle ϕ, the following angles are symmetry related for a square grid: $\{\phi, 180^o - \phi, 90^o - \phi, 90^o + \phi\}$.
Two boolean parameters allow to select which angles should be considered as related:

all 4 (do_symmetry_90degrees_min_phi=true,
do_symmetry_180degrees_min_phi = true)

only $\{\phi, 180 - \phi\}$ (do_symmetry_90degrees_min_phi=false,
do_symmetry_180degrees_min_phi = true)

none (do_symmetry_90degrees_min_phi=false,
do_symmetry_180degrees_min_phi = false)

Note that when `do_symmetry_90degrees_min_phi=true`, it is irrelevant what the value is of `do_symmetry_180degrees_min_phi`. This is because otherwise a non-consecutive range in ϕ would have to be used.

The symmetry in ϕ is automatically reduced for non-square grids or when the number of views is not a multiple of 4.

In addition, there is a keyword `use actual detector boundaries`. However, changing its default value is currently not recommended.

Note that even when all default values are used, the start and stop keywords given above have to follow the 'matrix type' keyword in a parameter file.

See also online documentation for class ProjMatrixByBinUsingRayTracing.

4.13.7.3 Interpolation

This projection matrix uses interpolation in projection space as model. When using this matrix for back-projection, the centre of a voxel is projected into a "viewgram" (*i.e.* roughly a projection plane but taking scanner geometry into account), and its contribution is computed by linearly interpolating between the surrounding values in the viewgram.

It is intended to give essentially the same results as the incremental-interpolation backprojector 4.13.6. This matrix version allows using the same model for forward projection.

Note The implementation does not use the incremental computations of 4.13.6 and is therefore not affected by rounding error.

Warning The current implementation is quite slow in constructing the matrix. If you need to use this multiple times, it might be worth writing it to disk and then reading it (see 4.13.7).

Parameters

The following parameters can be set (default values are indicated):

```
Interpolation Matrix Parameters:=
  ; any parameters appropriate for all matrices

  use_piecewise_linear_interpolation:=1
  use_exact_Jacobian:=1

  do_symmetry_90degrees_min_phi:=1
  do_symmetry_180degrees_min_phi:=1
  do_symmetry_swap_segment:=1
  do_symmetry_swap_s:=1
```

```
  do_symmetry_shift_z:=1
End Interpolation Matrix Parameters:=
```

See 4.13.6 for an explanation of the first two parameters.

4.13.7.4 SPECT UB

This is a flexible implementation specific to SPECT data. See [Fus13] and [Fus14] for some information. A sample is given below

```
Projection Matrix By Bin SPECT UB Parameters:=

   ;PSF type of correction { 2D // 3D // Geometrical }
   psf type:= 2D
   ; next 2 parameters define the PSF.
   ; They are ignored if psf_type is "Geometrical"
   ; the PSF is modelled as a Gaussian with sigma dependent on the
   ; distance from the collimator
   ; sigma_at_depth = collimator_slope * depth_in_cm + collimator_sigma_0
   collimator slope := 0.0163
   collimator sigma 0(cm) := 0.1466

   ;Attenuation correction { Simple // Full // No }
   attenuation type := Simple
   ;Values in attenuation map in cm-1
   attenuation map := attMapRec.hv

   ;Mask properties { Cylinder // Attenuation Map // Explicit Mask // No}
   mask type := Explicit Mask
   mask file := mask.hv

   ; if next variable is set to 0, only a single view is kept in memory
   keep all views in cache:=1

End Projection Matrix By Bin SPECT UB Parameters:=
```

The weight matrix or system matrix in tomographic reconstruction in SPECT is the matrix that stores the percentual contribution of each voxel of the object into each bin in the projections. In the MLE framework, weight ij can be though as the probability of one photon being emitted in the voxel i reached the bin j. Each weight can be factorized in three components: geometry (PSF), attenuation and scattering. This software can calculate the geometrical and the attenuation component. The correction for scattering should be done using alternative strategies. The geometrical part is calculated as the integral of the PSF (area 1) within the bin. The geometrical part is multiplied by the attenuation factor calculated using a modified version of the Siddon algorithm.

Parameters

maximum number of sigmas [0,∞,2]

This is the number of sigmas to consider in PSF correction (float, typically 1.5 to 2.5). PSF are modelled by Gaussian functions whose extension is infinite. To increase unnecessarily

the number of sigmas would produce an increase of the weight matrix size that would not result in a better correction for PSF. It does not make sense to take into account very small contributions. A balance between the precision of the correction for PSF and the size of the matrix and the time of the reconstruction process should be done.

psf type The possible values are { 2D, 3D, Geometrical (default) } to indicate, respectively, a planar reconstruction (no information mixed between planes, planes reconstructed sequentially, 1d -PSF), volumetric (mixed information between planes, volume reconstructed at once, 2d-PSF), or no correction for PSF (in this case PSF is the trapezoid obtained by the projection of the square voxel).

collimator slope, collimator slope sigma 0 (cm) (default 0)

These two 2 parameters define the PSF. They are ignored if *psf type* is *Geometrical*. These values are mostly dependent on your collimator. The PSF is modelled as a Gaussian with sigma dependent on the distance from the collimator

$$\sigma(d) = \text{slope} * d + \sigma_0 \tag{6}$$

with d the distance (in cm) of the voxel to the collimator radius.

In the range of usual distances between the object and the detector, the dependence of the sigma with the distance keeps perfectly linear for parallel collimators. You can adjust this dependence experimentally or use analytical methods to determine it.

attenuation type The possible values are { Simple, Full, No (default) }. Attenuation is calculated as the negative exponential of the sum of the length of the projection ray in each crossed voxel by its attenuation coefficient. It requires an attenuation map (see next keyword). The simple option for correction for attenuation is to consider that the whole PSF suffers the same attenuation (the attenuation of the central ray). One single factor is applied to weight the contribution from one voxel to all the bins in a detection plane. It is a very good approximation for uniform attenuation maps. The full option means that a different attenuation coefficient is calculated for each voxel-bin contribution (that is obtained along the voxel-bin pathway). It could be useful for very inhomogeneous attenuation maps.

See [Fus14] for an evaluation and more explanation.

attenuation map A filename giving the attenuation map as the attenuation coefficient at each voxel (in cm^{-1}). Currently, this file has the same geometric characteristics as the image to be reconstructed (number of columns, row, slices and voxel dimensions, orientation).

mask type This is a parameter to indicate if any mask should be applied to the volume, to reduce the matrix size by removing weights from voxels that, according the prevoius information, do no contribute to the projections. Possible values are:

Cylinder the cylinder inscribed into the volume is used as a mask. The radius of the cylinder is the maximum volume dimension (x-y),

Attenuation Map the attenuation map is used as a mask. No weight is calculated where the attenuation map is zero (assiming no attenuation=no activity). NaNs values are set to zero.

Explicit Mask the mask is defined in a file. The mask should have the same geometrical prescription than the image. It could be useful for instance to remove the weight of voxels from the table (no activity but attenuation) or to reduce the matrix size when no attenuation for correction is considered.

No (default). No attenuation correction.

mask file A filename with the mask (if the mask type is set to *Explicit Mask*). Any voxels which are (exactly) zero in the mask will be ignored. Currently, this file has the same geometric characteristics as the image to be reconstructed (number of columns, row, slices and voxel dimensions, orientation).

keep all views in cache [0,1,0]

If this variable is set to 0 (default), only a single view is kept in memory. This avoids running out-of-memory but means that the matrix has to be recomputed at every iteration.

4.13.7.5 From File

You can use `write_proj_matrix_by_bin` to write a projection matrix to file. You can then read the matrix back into memory. This could be useful if it takes a long time to generate the matrix, or if you have an external program to write the matrix (see below for more information on what you will need to do).

Parameters

The necessary parameters to include in the par file are:

```
ProjMatrixByBinFromFile Parameters:=
  Version := 1.0
  symmetries type := PET_CartesianGrid
    PET_CartesianGrid symmetries parameters:=
      do_symmetry_90degrees_min_phi:= <bool>
      do_symmetry_180degrees_min_phi:=<bool>
      do_symmetry_swap_segment:= <bool>
      do_symmetry_swap_s:= <bool>
      do_symmetry_shift_z:= <bool>
    End PET_CartesianGrid symmetries parameters:=
  ; example projection data of the same dimensions as used
  ; when constructing the matrix
  template proj data filename:= <filename>
  ; example image of the same dimensions as used
  ; when constructing the matrix
  template density filename:= <filename>
  ; binary data with projection matrix elements
  data_filename:=<filename>
End ProjMatrixByBinFromFile Parameters:=
```

The symmetries all default to true, but it is best to include the values in the file in all cases.

You need to be careful that these parameters match the matrix written to file. To make this easier, `write_proj_matrix_by_bin` will write these to file for you. In the current version of STIR, you will need to copy these into the .par file (this will change in a future version of STIR).

Note that image and projection data characteristics are read from the `template` files. Their geometric characteristics have to match those of the data that you want to process with the stored matrix (the stored matrix could have more segments or tangential positions than the data). This will be checked at run-time.

Information on how to write your own projection matrix to file

The projection matrix is stored as a sparse matrix (in the file dennoted by the `data_filename` parameter). The format is reasonably simple. For each element ("bin") in the projection data, the Line of Response (LOR) is encoded as follows:

```
segment_num (int32_t)
view_num (int32_t)
axial_pos_num (int32_t)
tangential_pos_num (int32_t)
num_voxels_in_LOR (uint32_t)
for_each voxel
   z (int16_t)
   y (int16_t)
   x (int16_t)
   matrix_value (float)
end
```

The order of bins is not important, neither is the order of the voxels (although you will get better performance if these are stored such that voxels are listed consecutively).

To reduce data size, symmetries can be used to store only "basic" Lines Of Responses (LORs). The exact definition of the symmetries is unfortunately not easy and not documented fully here. If you want to use your own code to write the matrix and want to use symmetries you will need to check the `DataSymmetriesForBins` class or a derived class. In case all symmetries are enabled, the following should hold:

```
0<=segment_num
0<=view_num<=num_views/4
axial_pos_num==0
0<=tangential_pos_num
```

Check the STIR developer's guide and the Wiki for information on coordinate systems used by STIR. In particular, note the STIR convention about index numbering in section 2.2 of the developer's guide.

A final note: currently the sparse matrix is read completely into memory before it is used (but of course keeping only the "basic" part of the matrix taking the symmetries into account). This restricts the size of the matrix according to how much memory your system has available.

4.13.8 Selecting a bin normalisation procedure

In PET, a procedure called 'normalisation' is used which is essentially a calibration procedure for every detector pair. It provides a multiplicative factor for every bin, or element of the projection data.

The library can provide different types of normalisation procedures. It is possible to select these independently at run-time, and extend the available ones at compile time. The mechanism is exactly the same as for the ForwardProjector hierarchy.

In addition to the types listed below, you can also enter 'None', which means that the data won't be normalised at all.

4.13.8.1 From Projdata

This can be used when the normalisation factors are stored simply as projection data. Currently, these data have to have exactly the same characteristics (size etc.) as the projection data which are

going to be normalised. Note that the stored factors have to be the ones you'd apply to normalise the data (and not their reciprocal).

Parameters

```
Bin Normalisation From ProjData :=
normalisation projdata filename:= norm.hs
End Bin Normalisation From ProjData:=
```

See also online documentation for class BinNormalisationFromProjData.

4.13.8.2 From ECAT7

This can be used when normalising ECAT7 data. CTI/Siemens stores the normalisation data in a files normally ending on `.n` or `.N`. Dead-time correction is also supported, although awkwardly. To get dead-time correction to work, you need to specify the singles rates for the scan[24]

Parameters

```
Bin Normalisation From ECAT7:=
  normalisation filename:= STUDY.n
  singles rates := Singles From ECAT7 File
  Singles Rates From ECAT7 File:=
      ECAT7_filename := ecat7_sinogram.S
  End Singles Rates From ECAT7:=
End Bin Normalisation From ECAT7:=
```

In addition, you currently have to set the time frame information in the .par file.
See also online documentation for class BinNormalisationFromECAT7.

4.13.8.3 From Attenuation Image

This can be used for attenuation correction factors (ACFs) if you do not have the ACFs but an attenuation image (or mu-map). The ACFs are found by forward projecting the attenuation image, multiplying the result with -1, and exponentiating, i.e. using Beer's law.

Warning Attenuation image data are supposed to be in units cm^{-1}. (Reference: water has $\mu = .096 cm^{-1}$.)

Parameters

```
Bin Normalisation From Attenuation Image:=
attenuation_image_filename := <string>
forward projector type := <string>
End Bin Normalisation From Attenuation Image :=
```

Default forward projector is ForwardProjectorByBinUsingRayTracing (see section 4.13.5).
See also online documentation for class BinNormalisationFromAttenuationImage.

[24] In future, hopefully this will not be necessary. This work-around is needed because *STIR* currently does not directly read any of the meta-data in the headers of sinograms etc.

4.13.8.4 Chained

This can be used to apply two normalisation one after the other. For example, a first one could be the 'usual' normalisation factor, while a second one could be the attenuation factors.

Parameters

```
Chained Bin Normalisation Parameters:=
Bin Normalisation to apply first:= some_bin_normalisation_type
; parameters for this type

Bin Normalisation to apply second:= some_bin_normalisation_type
; parameters for this type

END Chained Bin Normalisation Parameters:=
```

See also online documentation for class ChainedBinNormalisation.

4.13.9 Available shapes

STIR can use shapes, *e.g.* in the `generate_image` utility, or for specifying ROIs. The distribution contains sample parameter files in the **samples** directory. In the following sub-sections the available shapes are listed.

Most shapes have a centre and orientation. This is specified in the parameter files by giving the "origin" and a $3x3$ matrix specifying 3 direction vectors. Note that these vectors do not necessarily have to be orthogonal nor have unit-norm. To decide if a point with coordinates `coords` is inside the shape, the coordinates are first translated to "shape-specific" coordinates using:

$$\texttt{shape\_coords} = \texttt{direction\_vectors}.(\texttt{coord} - \texttt{origin})$$

When parsing, the relevant variables are specified as follows:

```
origin (in mm):= <float> ;defaults to {0,0,0}
 ; values below are give a rotation around y for 90 degrees (swapping x and z)
 ; Warning: this uses the STIR convention {z,y,x}
 direction vectors (in mm) := { {0,0,1}, {0,1,0}, {-1,0,0}}
```

See also the online documentation for the class **Shape3D** and **Shape3DWithOrientation**.

4.13.9.1 Box3D

Three-dimensional cuboid box.

Parameters

```
Box3D Parameters:=
   length-x (in mm):= <float>
   length-y (in mm):= <float>
   length-z (in mm):= <float>
   ; any parameters of Shape3DWithOrientation
End:=
```

4.13.9.2 Ellipsoid

Three-dimensional ellipsoid. A point with coordinates "shape-coordinates" x, y, z is inside the shape if

$$\frac{x^2}{R_x^2} + \frac{y^2}{R_y^2} + \frac{z^2}{R_z^2} \leq 1$$

Parameters

```
Ellipsoid Parameters:=
  radius-x (in mm):= <float>
  radius-y (in mm):= <float>
  radius-z (in mm):= <float>
  ; any parameters of Shape3DWithOrientation
End:=
```

4.13.9.3 Ellipsoidal Cylinder

Three-dimensional ellipsoidal cylinder (oriented along the z-axis). A point with coordinates "shape-coordinates" x, y, z is inside the shape if

$$\frac{x^2}{R_x^2} + \frac{y^2}{R_y^2} \leq 1 \quad \text{and} \quad \text{abs}(z) \leq L_z/2$$

In addition, this shape can be restricted to a wedge by specifying initial and final angles (w.r.t. the x axis).

Parameters

```
Ellipsoidal Cylinder Parameters:=
  radius-x (in mm):= <float>
  radius-y (in mm):= <float>
  length-z (in mm):= <float>
  initial angle (in deg):= <float> ; (defaults to 0)
  final angle (in deg):= <float>   ; (defaults to 360)
  ; any parameters of Shape3DWithOrientation
End:=
```

4.13.9.4 Discretised Shape3D

This shape can be used to read for instance saved ROIs. Image values are supposed to be between 0 and 1. Note that centre and orientation are taken from the image data, not from the parameter file.

Parameters

```
Discretised Shape3D Parameters:=
  input filename := <filename>
END:=
```

where `filename` needs to specify a volume that can be read by STIR.

4.14 Display

Some of the programs (e.g. in utilities/, recon_tests/) use the display routines. Which version is actually used depends on the compilation settings (in particular the *GRAPHICS* cmake variable), see section 3.3.1.

4.14.1 X Windows display

This provides a (very basic) way of displaying bitmaps when using X windows. This works by creating a new window where some of the bitmaps are displayed. To proceed, you have to make this window the 'focus' (how to do this depends on your window manager but usually you have to move the cursor over it, or click the title bar) and then press any key. In your original terminal window you will then be asked if you want to continue with the next set of bitmaps until no more are left.

In addition, when your X server supports the Pseudocolor visual, you can cycle between 4 different colour scales by pressing Mouse button 2 while the 'bitmap' window is selected.

Warning In the current implementation, it seems to happen occasionally that not all colours in the colour scale have been allocated properly. This can be seen by looking at the side bar displaying the colour scale. Unfortunately, this effect might give you wrongly coloured regions (usually spots) in the image.

4.14.2 PGM display

This 'display' mode actually writes out a file in the Portable Greyscale Map format, which can be read by various graphics programs (like Paint Shop Pro or xv).

4.14.3 MathLink display

This mode is (somewhat) useful if you have *Mathematica*^{TM}, and want to pipe the data into *Mathematica*. (It is probably easier to read the binary data from file though using the *Mathematica* command `BinaryRead`). Sample *Mathematica* statements:

```
(* create link before starting the (first) display in your STIR
program*)

link=LinkCreate[‘‘STIR’’];

(* read data from link *)
data3d=LinkRead[link];


(* display the 3rd image *)
ListPlot3D[data3d[[3]], Mesh->False];
(* read data from link from next display *)
nextdata3d=LinkRead[link];
(* and so on *)

(* close link at end of STIR program *)
LinkClose[link];
```

5 B-spline interpolation in STIR

Special classes to interpolate data using B-Splines [Uns99] have been implemented in STIR. The B-Splines coefficients are estimated with a fast method that incorporates FIR and IIR filters [Uns93]. The level of B-splines that are currently available are 0, 1 (linear), 2 (quadratic), 3 (cubic), 4 (quartic) and 5 (quintic). Moreover, a synthetic spline has also been implemented based on B-splines for maximum order interpolation but minimal support, i.e. MOMS splines [Blu01]. In all cases no regularization has been incorporated during the interpolation procedures, therefore special care should be taken when higher than the first order B-splines are used, as noise may be enhanced at the interpolated data. Interpolation can be also performed in sinogram space. However this space is regular only in axial, and not in radial, angular and azimuthal directions. Therefore, the current interpolation can be cubic only in axial direction while for the rest directions a choice of linear interpolator is safer. In future with some additional classes, reconstruction could be performed to obtain B-spline coefficients directly from the projection sinograms, which can be useful for better noise properties [Nic02].

6 Directories in the STIR tree

STIR can roughly be split into a library applications and applications. [25]

See the doxygen documentation for an overview of the directory tree with brief descriptions.

7 Future developments and Support

The *STIR* library, in its current state, possesses many capabilities. The developers, however, look forward to still further increases in the flexibility and power of the software. Some of the developments being discussed are:

- expanded library of polymorphic classes (e.g. image grids, and ordered subsets)
- additional scanners, also for SPECT
- additional data formats support, without conversion to Interfile.
- point-spread function reconstruction

While support for the library is on a voluntary basis, users of the library are encouraged to subscribe to relevant *STIR* mailing lists (see the 'Mailing Lists' section of the *STIR* website `http://stir.sourceforge.net`) where they can follow developments of the software and obtain helpful information from other users. Questions will ONLY be answered (if at all) when directed to the mailing list.

Commercial support is available from `Algorithms and Software Consulting Ltd`.

Below, we list of some of the features that might make it into the next releases. However, which feature is actually finalised/implemented depends on the needs of the developers. If you want one of these features and are willing to help, let us know.

- More automatic testing programs
- More algorithms: potentially ART, OSCB [Ben99b]
- More projectors

[25] The distinction is not complete as some applications are implemented in terms of classes that actually end up in the library.

- More priors
- Extending the parallelisation of OSMAPOSL and OSSPS to FBP3DRP etc, or using OPEMMP
- Compatibility of the interpolating backprojector with recent processors.
- More kinetic models: Spectral Analysis, Logan Plot

8 References

[Ahn2003] S. Ahn and J. A. Fessler, *Globally convergent image reconstruction for emission tomography using relaxed ordered subsets algorithms,* IEEE Trans. Med. Imag., vol. 22, no. 3, pp. 613-626, May 2003.

[Ale97] Alenius S and Ruotsalainen U **(1997)** Bayesian image reconstruction for emission tomography based on median root prior. *European Journal of Nuclear Medicine* , Vol. 24 No. 3: 258-265.

[Ang11] Angelis G I, Thielemans K, Tziortzi A C, Turkheimer F E and Tsoumpas C **(2011)**. Convergence Optimization of Parametric MLEM Reconstruction for Estimation of Patlak Plot Parameters. *Computerized Medical Imaging and Graphics*, 35:407-416, *doi:10.1016/j.compmedimag.2011.01.002*

[Ben99] Ben-Tal A, Margalit T and Nemirovski A **(1999)** The ordered subsets mirror descent optimization method with application to tomography. *Research report #2/99, March 1999, MINERVA Optimization Center, Faculty of Industrial Engineering and Management, Technion – Israel Institute of Technology.*

[Ben99b] Ben-Tal A and Nemirovski A **(1999)** The conjugate barrier method for non-smooth, convex optimization. *Research report #5/99, October 1999, MINERVA Optimization Center, Faculty of Industrial Engineering and Management, Technion – Israel Institute of Technology.*

[Bei08] Beisel T, Lietsch S, and Thielemans K **(2008)** A Method for OSEM PET Reconstruction on Parallel Architectures Using STIR. *Proc. of the IEEE Medical Imaging Conference, Dresden, Germany, 2008.*

[Blu01] Blu T, Thcvenaz P, Unser M **(2001)**. MOMS: maximal-order interpolation of minimal support. *IEEE Trans Image Processing* 10(7): 1069-1080.

[Bue12] Buerger, C., T. Schaeffter, and A. P. King **(2011)**, Hierarchical adaptive local affine registration for fast and robust respiratory motion estimation. Medical Image Analysis, vol. 15, pp. 551-564, `doi 10.1016/j.media.2011.02.009`

[Bou97] Boudraa A E O **(1997)** Automated detection of the left ventricular region in magnetic resonance images by Fuzzy C-Means model. *Int J of Cardiac Imag* ; 13: 347-355.

[Dau86] Daube-Witherspoon M E and Muehllener G **(1986)** An iterative space reconstruction algorithm suitable for volume ECT. *IEEE Trans. Med. Imaging*, vol. MI-5: 61-66.

[Dau97] Daube-Witherspoon M E and Muehllehner G **(1987)** Treatment of axial data in three-dimensional PET, J. Nucl. Med. 28, 171-1724.

[Def95] Defrise M. **(1995)** "A factorization method for the 3D X-ray transform" *Inverse Problems,* **11** pp. 983-994.

[Def97] Defrise M, Kinahan P E, Townsend D W, Michel C, Sibomana M and Newport D F **(1997)** Exact and approximate rebinning algorithms for 3-D PET data. *IEEE Trans. Med. Imaging*, MI-16: 145-158.

[Egg98] Egger M L, Joseph C and Morel C **(1998)** Incremental beamwise backprojection using geometrical symmetries for 3D PET reconstruction in a cylindrical scanner geometry. *Phys Med. Biol.*, 43: 3009-3024.

[Gem84] Geman S and Geman D **(1984)** Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans PAMI*, 6: 721-741.

[Gem85] Geman S and McClure D **(1985)** Bayesian image analysis: an application to single photon emission tomography. *in Proc. American Statistical Society, Statistical Computing Section (Washington, DC)* 12-18.

[Gre90] Green P J **(1990)** Bayesian reconstruction from emission tomography data using a modified EM algorithm. *IEEE Trans. Med. Imaging*, MI-9: 84-93.

[Heb89] Hebert T J and Leahy R M **(1989)** A generalized EM algorithm for 3-D Bayesian reconstruction from Poisson data using Gibbs priors. *IEEE Trans. Med. Imaging*, MI-8: 194-202.

[Her80] Herman G T **(1980)** Image Reconstruction from Projections: The fundamentals of Computational Tomography. *Academic Press, New York*.

[Hud94] Hudson H M and Larkin R S **(1994)** Accelerated image reconstruction using ordered subsets of projection data. *IEEE Trans. Med. Imaging*, MI-13: 601-609.

[Fus13] Berta Marti Fuster, Carles Falcon, Charalampos Tsoumpas, Lefteris Livieratos, Pablo Aguiar, Albert Cot, Domenec Ros and Kris Thielemans, **(2013)** Integration of advanced 3D SPECT modeling into the open-source STIR framework, *Med. Phys. 40*, 092502; http://dx.doi.org/10.1118/1.

[Fus14] Berta Marti Fuster, Kjell Erlandsson, Carles Falcon, Charalampos Tsoumpas, Lefteris Livieratos, Domenec Ros, Kris Thielemans, **(2014)** Evaluation of the novel 3D SPECT Modelling Algorithm in the STIR Reconstruction Framework: Simple vs. full attenuation correction, *proc. IEEE MIC 2013*, Seoul, Korea.

[Jac00] Jacobson M, Levkovitz R, Ben-Tal A, Thielemans K, Spinks T, Belluzzo D, Pagani E, Bettinardi V, Gilardi M C, Zverovich A and Mitra G **(2000)** Enhanced 3D PET OSEM Reconstruction using inter-update Metz filters. *Phys. Med. Biol.* **45** No.8 (2000) 2417-2439*l.*

[Kin89] Kinahan P E and Rogers J G **(1989)** Analytic 3D image reconstruction using all detected events. *IEEE Trans. Nucl. Sci.*, 36: 964-968.

[Kle96] Klein, G. J., B. W. Reutter, and R. H. Huesman **(1996)**, Non-rigid summing of gated PET via optical flow, *1996 IEEE Nuclear Science Symposium Conference Record*, vol. 2, pp. 1339-1342, `doi 10.1109/NSSMIC.1996.591692`

[Lab97] Labbe C, Ashburner J, Koepp M, Spinks T, Richardson M and Cunningham V **(1997)** Accurate PET quantification using correction for partial volume effects within cerebral structures. *Neuroimage*, 5: B12.

[Lab99a] Labbé C, Thielemans K, Belluzzo D, Bettinardi V, Gilardi MC, Hague DS, Jacobson MW, Kaiser S, Levkovitz R, Margalit T, Mitra G, Morel C, Spinks T, Valente P, Zaidi H and Zverovich A*: An object-Oriented Library for 3D PET Reconstruction using Parallel Computing*, Proceedings of Bildverarbeitung fuer die Medizin 1999, Algorithmen-Systeme-Anwendungen, *Informatik aktuell, Springer*, Eds. H. Evers, G. Glombitza, T. Lehmann, H.-P. Meinzer, pp 268-272.

[Lab99b] Labbé C, Thielemans K, Zaidi H and Morel C: An object-oriented library incorporating efficient projection/backprojection operators for volume reconstruction in 3D PET. *Proc. of 3D99 Conference*, June 1999, Egmond aan Zee, The Netherlands

[Lal93] Lalush D S and Tsui M W **(1993)** A general Gibbs prior for Maximum a posteriori reconstruction in SPET. *Phys. Med. Biol.*, 38: 729-741.

[Lan90] Lang K **(1990)** Convergence of EM Image reconstruction algorithms with Gibbs smoothing. *IEEE Trans. Med. Imaging*, MI-9: 4.

[Lew98] Lewellen T K, Harrison R L and Vannoy S **(1998)** The SimSET program, *Monte Carlo Calculations in Nuclear Medicine: Applications in Diagnostic Imaging ed M Ljungberg, S-E Strand and M A King (Bristol: Institute of Physics Publishing)* pp 77-92.

[Li06] Li, T., B. Thorndyke, E. Schreibmann, *et al* **(2006)**, Model-based image reconstruction for four-dimensional PET. *Med Phys* vol. 2, pp. 1288-1298 `doi 10.1118/1.2192581`

[Man97] Manders Jones H **(1997)** A Computational Investigation of the solution of large scale Optimization problems. *Ph.D Thesis, Department of Mathematics and Statistics. Brunel, The*

University of West London. Oct 1997.

[Mar99] Margalit T, Gordon E, Jacobson M, Ben-Tal A, Nemirovski A and Levkovitz R **(1999)** The ordered sets mirror descent and conjugate barrier optimization algorithms adapted to the 3D PET reconstruction problem. Technion internal report and PhD thesis of T. Margalit

[Mus01] Mustafovic S and Thielemans K, **(2001)** Additive and Multiplicative versions of the Maximum A Posteriori Algorithm with the Median Root Prior. poster at *IEEE Med. Img. Conf. 2001.*

[Mus04] Mustafovic S and Thielemans K, **(2004)** *Object Dependency of Resolution in Reconstruction Algorithms with Inter-Iteration Filtering Applied to PET* , IEEE Trans. Med. Im. 23 (4): (2004) 433-446.

[Mus02] Mustafovic S and Thielemans K, *,* **(2002)** Comparison of Unconventional Inter-Filtering Methods to Penalised-likelihood for Space-invariant Tomographs, poster at *IEEE Medical Imaging Conf. 2002.*

[Nem78] Nemirovski A and Yudin D **(1978)** Problem complexity and method efficiency in optimization. *Nauka Publishers, Moscow, 1978 (in Russian); English translation: John Wiley & Sons, 1983.*

[Nic02] Nichols, T. E., J. Qi, E. Asma and R. M. Leahy **(2002)**. Spatiotemporal reconstruction of list-mode PET data. *IEEE Trans Med Imaging* 21(4): 396-404.

[Pat83] Patlak C S, Blasberg R G, Fenstermacher J D **(1985)** Graphical evaluation of blood-to-brain transfer constants from multiple-time uptake data. *J Cereb Blood Flow Metab* 3(1): p. 1-7.

[Pat85] Patlak C S, Blasberg R G **(1985)** Experimental and Graphical evaluation of blood-to-brain transfer constant from multiple-time uptake data: Generalizations. *J Cereb Blood Flow Metab* 5: p. 584-90.

[Poe03] Poenisch F, Enghardt W, and Lauckner K **(2003)**, Attenuation and scatter correction for in-beam positron emission tomography monitoring of tumour irradiations with heavy ions. *Phys Med Biol* 48(15): 2419-2436.

[PAR1.3] Levkovitz R, Zibulevsky M, Labbe C, Zaidi H and Morel C **(1997)** Determination of a Set of Existing Algorithms for Comparative Evaluation. *PARAPET documentation for D1.3, Geneva University Hospital and Technion.*
Available at `http://stir.sourceforge.net`.

[Pol 11] Polycarpou I, Thielemans K, Manjeshwar R, Aguiar P, Marsden PK, Tsoumpas C (2011) Comparative evaluation of scatter correction in 3D PET using different scatter-level approximations. Ann Nucl Med, 25(9): 643-649, doi: 10.1007/s12149-011-0514-y.

[Pol12] Polycarpou, I., C. Tsoumpas and P. K. Marsden **(2012)**, Analysis and comparison of two methods for motion correction in PET imaging *Med Phys* vol. 39, 3586-3590, `doi 10.1118/1.4754586`.

[Rah08] Rahmim, A., K. Dinelle, *et al* **(2008)**, Accurate event-driven motion compensation in high-resolution PET incorporating scattered and random events. *IEEE Trans Med Imaging*, vol. 27, 1018-1033, `doi 10.1109/TMI.2008.917248`.

[Rea98] Reader A J, Visvikis A, Erlandsson K, Ott R J, and Flower M A **(1998)** Intercomparison of four reconstruction techniques for positron volume imaging with rotating planar detectors. *Phys. Med. Biol.*, 43: 823-34.

[She82] Shepp L A and Vardi Y **(1982)** Maximum likelihood reconstruction for emission tomography. *IEEE Trans. Med. Imaging*, 1: 113-122.

[Sil90] Silverman B W, Jones M C, Wilson J D, and Nychka D W **(1990)** A smoothed EM approach to indirect estimation problems, with particular reference to stereology and emission tomography. *J. Roy. Stat. Soc.*, 52: 271-324.

[Thi07] Thielemans K, Manjeshwar R, Tsoumpas C and Jansen F**(2007)**, A new algorithm for

scaling of PET scatter estimates using all coincidence events. *IEEE Nucl Sci Symp Med Imaging Conf* 3586 - 3590.

[Thi12] Kris Thielemans, Charalampos Tsoumpas, Sanida Mustafovic, Tobias Beisel, Pablo Aguiar, Nikolaos Dikaios, and Matthew W Jacobson, STIR: Software for Tomographic Image Reconstruction Release 2, *Physics in Medicine and Biology, 57 (4), 2012 pp.867-883.*

[Tso04] Tsoumpas C, Aguiar P, Nikita K S, Ros D and Thielemans K**(2004)**, Evaluation of the single scatter simulation algorithm implemented in the STIR library. *IEEE Nucl Sci Symp Med Imaging Conf* 3361-3365.

[Tso07] Tsoumpas C, Turkheimer F E and Thielemans K **(2007)**, Convergence properties of algorithms for direct parametric estimation of linear models in dynamic PET. *IEEE Nucl Sci Symp Med Imaging Conf* 3034-3037

[Tso08] Tsoumpas C, Turkheimer F E and Thielemans K **(2008)**, Study of direct and indirect parametric estimation methods of linear models in dynamic positron emission tomography. *Med. Phys.* 35(4): p. 1299-1309.

[Tso13] Tsoumpas, C., I. Polycarpou, K. Thielemans, *et al* **(2013)**, The effect of regularisation in motion compensated PET image reconstruction: A realistic numerical 4D simulation study. *Phys Med Biol*, vol. 58, 1759-1773, `doi 10.1088/0031-9155/58/6/1759`.

[Uns93] Unser M, Aldroubi A, Eden M **(1993)**. B-Spline Signal Proccessing: Part II - Efficient Design and Applications. *IEEE Trans Signal Processing* 41(2): 834-848.

[Uns99] Unser M. **(1999)**. Splines: A Perfect Fit for Signal and Image Processing. *IEEE Signal Processing Magazine*: 22-38.

[Wat96] Watson C C, Newport D, and Casey M E**(1996)**, A single-scatter simulation technique for scatter correction in 3D PET. *Fully 3D Image Recon Radiol Nucl Med* Kluwer Academic: 255-268.

[Wat97a] Watson C C, Newport D, Casey M E, deKemp R A, Beanlands R S, and Schmand M **(1997)**, Evaluation of Simulation-Based Scatter Correction for 3-D PET Cardiac Imaging. *IEEE Trans Nucl Sci* , 44(1): 90-97.

[Wat97b] Watson C C, Newport D, Casey M E, Beanlands R S, Schmand M, and deKemp R A **(1997)**, A technique for measuring the energy response of a PET tomograph using a compact scattering source. *IEEE Trans Nucl Sci* , 44(6): 2500-2508.

[Wat00] Watson C C **(2000)**, New, faster, image-based scatter correction for 3D PET. *IEEE Trans Nucl Sci* 47(4): 1587-1594.

[Wat04] Watson C C, Casey M E, Michel C, Bendriem B**(2004)**, Advances in scatter correction for 3D PET/CT. *IEEE Nucl Sci Symp Med Imaging Conf* 3008 - 3012.

[Wer02] Werling A, Bublitz O, Doll J, Adam L E, and Brix G**(2002)**, Fast implementation of the single scatter simulation algorithm and its use in iterative image reconstruction of PET data. *Phys Med Biol* 47(16): 2947-2960.