Upgrading from STIR version 1.4 to 2.0

Kris Thielemans

Contents

1 Overview

This document summarises what the differences are between STIR version 2 and 1.4.

The main change in STIR 2.0 is in the reconstruction code. All iterative reconstructions use now an objective function hierarchy. See the doxygen documentation for `GeneralisedObjectiveFunction`. This will make it far easier to add different types of reconstruction algorithms, and in particular applicable to something else than PET emission data. For instance, in STIR 2.0 there is now a facility to reconstruct list mode data as well. In addition, reconstruction classes have now been templated in the type of the "target" data, i.e. the parameters that need to be estimated. In STIR 1.4 this was fixed to images (or in fact `DiscretisedDensity<3,float>` objects. The following describes what the consequences are for you.

Another change is that the ECAT6 code now relies on the LLN Matrix library, to avoid problems with copyright issues. However, as the Matrix library is somewhat out-dated, this is likely to generate problems (there are known aliasing problems on 64-bit systems). As a result, ECAT6 support in STIR is now problematic.

2 For Users

Generally, all utilities work in the same way as before. There are changes in some parameters used for parsing.

2.1 Bugs fixed

Various small bugs were fixed, but I am not aware of bugs that caused major problems, except things listed below, such as a bug when using Euler angles when constructing shapes.

Bing Bai found a work-around for the problem with the incremental interpolating backprojector. This seems now to work on all system where we tested it. The code is still inherently unstable though.

2.2 Incompatible behaviour

- Old OSMAPOSL parameter files no longer work. See the section below on OSMAPOSL, and sample files in the `samples` directory. Also, output is not necessarily identical. See section 2.6 for details.
- The parsing keyword `output file format` in the reconstruction hierarchy and `generate_image` has been changed to `output file format type` to be consistent with other parameters (it was already called that way in `stir_math`.
- The format for the parameter files for `generate_image` and `list_ROI_values` changes somewhat. The only difference is the specification of the origin of the shape as a single keyword specifying {z,y,x}.
- Image-origin is now read from and saved to file for those file formats that support it (at time of writing: Interfile, ECAT7 and ECAT6 for x,y info). As this was ignored before (origin was set to 0,0,0), this might mean that the images are now forward projected/reconstructed differently.
  WARNING: At present, origin info is not set/preserved for projection data. This is a major deficiency and will be fixed at a later stage. At present, the first sinogram of segment 0 is supposed to be at `z=0` and the axis of the scanner at `x=y=0`.

2.3 New utilities

conv_AVW If you have the AVW^{TM} library [1] installed on your system, the make process should have built `utilities/conv_AVW`. 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 flip some images depending on the file format. For instance, it reads *ECAT7* files using a z-flip compared to STIR. In general, STIR ignores any image orientation fields. This is dangerous of course.

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

list_image_values, find_maxima_in_image Utilities for getting data from images from the command line

attenuation_coefficients_to_projections, calculate_attenuation_coefficients Utilities for processing of attenuation coefficients.

find_maxima_in_image, list_image_values Utilities to list voxel values and coordinates of maxima.

write_proj_matrix_by_bin Write the elements of a projection matrix to a file (in a STIR-specific format). This can later be used to re-read the same data as opposed to having to recalculate all elements.

copy_ecat7_header Allows copying header info between ECAT7 files. Note that all ECAT utilities are now in the `utilities/ecat` sub-directory.

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

2.4 Filters *aka* data processors

The `ImageProcessor` hierarchy is now generalised to `DataProcessor<DataT>` that can work on different types than images. This is reflected in the name of the parameters of some data processors.

Chained Image Processor is now renamed to `Chained Data Processor`. This data processor is used to allow you to specify two data processors that will run in sequence. So, all its keywords have been changed accordingly.

```
Chained Data Processor Parameters:=
Data Processor to apply first:=
Data Processor to apply second:=
END Chained Data Processor Parameters:=
```

Threshold Min To Small Positive Value used to truncate to a cylindrical FOV. This facility no longer exists. There is a new filter for this `Truncate To Cylindrical FOV`. See also section 2.6.5.

2.5 Shapes

The code to specify Euler angles is disabled as it was buggy (get after set was not consistent). You now have to use a matrix to define direction vectors. (Note that in previous version the direction vectors were ignored during parsing (this was a bug).)

The origin is now also specified with a single keyword.

```
; next keyword can be used for non-default axes
; 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}}
; origin w.r.t. to standard STIR coordinate system (middle of first plane)
origin (in mm):= {z,y,x}
```

The following 2 entries were contributed by C. Ross Schmidtlein and Assen S. Kirov. See also the `contrib` sub-directory available in the documentation zip file.

2.5.1 Box3D

Allows to specify 3D boxes (or cuboids).

2.5.2 EllipsoidalCylinder

Allows to specify specify angle parameters to specify a wedge of the cylinder.

2.6 Reconstruction programs

Analytic reconstruction algorithms have not changed, except that there is now a new keyword

```
post filter type:=
```

However, as mentioned above, iterative reconstruction code has been changed dramatically. Because of this, the parameter format for `OSMAPOSL` has changed. Previous .par files are incompatible. However, this is mostly a question of reordering the keywords. Generally speaking, everything related to the input data, sensitivity and the prior (except the `MAP model` keyword) has now to be between

```
objective function type:= \
  PoissonLogLikelihoodWithLinearModelForMeanAndProjData
PoissonLogLikelihoodWithLinearModelForMeanAndProjData Parameters:=
; input, sensitivity and prior parameters here
end PoissonLogLikelihoodWithLinearModelForMeanAndProjData Parameters:=
```

Also, the `sensitivity` program no longer exists. Its functionality is now fully integrated into `OSMAPOSL` (and other forthcoming algorithms). See below for details. .

2.6.1 Keyword name changes

Some keywords changed because the `OSMAPOSL` code can now work on different "target types", i.e. not only images.

sensitivity image is now `sensitivity filename`.

save image at subiteration intervals is now `save estimate at subiteration intervals`.

initial image is now `initial estimate`.

2.6.2 Keywords (and functionality) that are no longer present

do rim truncation is now effectively always set to 0. See also section 2.6.5.

2.6.3 Changes for computing the sensitivity image

In STIR 1.4, a separate executable `sensitivity` was used to compute the sensitivity image. This is now a part of `OSMAPOSL` with relevant keywords (with defaults indicated)

```
time frame definition filename:=
time frame number:= 1
Bin Normalisation type:= None
sensitivity filename:=
recompute sensitivity := 0
```

The first three are used to specify normalisation and attenuation factors as in `correct_projdata`. See the Userś Guide for more information.

2.6.4 New functionality in OSMAPOSL

There is a new keyword

```
use_subset_sensitivities
```

which defaults to 0, which is the old behaviour. If set to 1, the denominator term in OSMAPOSL uses sub-sensitivities as opposed to the total sensitivity image (divided by the number of subsets). This avoids artefacts when using a larger number of subsets and attenuation and/or normalisation in the model, at the expense of using more memory. At time of writing, this has to be combined with

```
recompute sensitivity := 1
```

The sub-sensitivities are not written to file.

2.6.5 Changes in reconstructed images when using OSMAPOSL

In STIR 1.4, `OSMAPOSL` used "rim truncation" by default. This set all voxels outside a cylindrical FOV to zero. In version 2.x, this facility no longer exists. The main reason for this is that the `OSMAPOSLReconstruction` class now works for target types which are not images. In addition, you no longer have to jump through a lot of hoops if you want to use a square FOV.

Also, a bug was removed that prevented
`enforce initial positivity condition:=1`
to work.

As a consequence, reconstructed images might not be identical. In the first few iterations, the difference is only in the rim, but this then induces differences in the whole image during subsequent iterations. Of course, if you do see appreciable differences, it shows that there probably is something wrong with your reconstructions, as most likely there should be no activity in those rim voxels in any case. One potential reason is that you are using randoms or scatter pre-corrected data, or do not use this corrections at all. The theoretically recommended, and most numerically stable, way is to use the `additive sinogram` parameter.

Recommendations

- No longer use rim truncation. If you need a cylindrical FOV, let the projectors do this for you (they do this by default).
- If you see pixels at the edge of the FOV which are very high, you probably can solve this by using `additive sinogram` appropriately.

Guidelines on how to obtain identical reconstructions

Read this section if you really need compatibility. This is a bit messy though, as there were some problems with the rim truncation in STIR 1.4. So, we need to explain first what happened.

Rim truncation was applied in 3 different processing steps:

- After reading the initial image, if `enforce initial positivity condition` was set to 1 (the default).
- After dividing the updated image with the sensitivity image, if `do rim truncation` was set to 1 (the default).
- After applying the inter-iteration or inter-update filter (but only if they were set).

This first and third step were a (potentially unexpected) consequence of using the image processor `ThresholdMinToSmallPositiveValueImageProcessor`.

Unfortunately, the first and third step used a FOV that could be a few pixels larger than the second step. This is because they [2] used a test $x^2 + y^2 <= R^2$, while the explicit rim truncation [3] used $x^2 + y^2 < R^2$. Finally, the origin used for this computation was shifted half a pixel in x and y for even-sized images w.r.t. the standard STIR origin.

The final piece of the puzzle is that the interpolating backprojector has a FOV which is slightly smaller than the rim truncation routines used in STIR 1.4.

Conclusions

- Reconstructions should be identical if `OSMAPOSL` in STIR 1.4 used
  `do rim truncation:=0`
  `enforce initial positivity condition:=0`
  and there was no inter-update nor inter-iteration filtering.
- Most likely, differences will be negligible if you use the interpolating backprojector.

[2] Routines `threshold_min_to_small_positive_value` and `min_positive_value`.
[3] Routines `divide_and_truncate` and `truncate_rim`

- Otherwise, you can use the `Truncate To Cylindrical FOV` data processor in STIR 2.0. Use it with

  ```
  strictly_less_than_radius:=0
  ```

  to reproduce the first and third step in STIR 1.4 processing, and with

  ```
  strictly_less_than_radius:=1
  ```

  for the second step. Note however, that if `do_rim_truncation` was switched on, the `Truncate To Cylindrical FOV` filter would have to be used as inter-iteration filter at every sub-iteration, which might not be possible if you are using another inter-iteration filter.
  See the `OSMAPOSL_test_PM_MRP.par` and `postfilter_truncate_circular_FOV.par` files in the **recon_test_pack** for examples.

2.6.6 New projection matrices

A new "Using Interpolation" matrix attempts to reproduce results of the interpolating backprojector. It is currently very slow though.

A new "From File" matrix allows you to read a matrix (saved with `write_proj_matrix_by_bin`).

2.6.7 Parallelisation of OSMAPOSL using MPI

Contribution by Tobias Beisel. Allows running OSMAPOSL in master-slave mode. See some more info in the User's Guide.

2.6.8 Parallelisation of FBP2D using OPENMP

Contribution by Tobias Beisel. Allows running FBP2D using threads. However, this does not really speed-up the reconstruction. This code could serve as a basis for FBP3DRP however.

3 For Developers

Before reading this, it is recommended that you read the section for users first.

As mentioned before, the main changes are in the `Reconstruction` hierarchy. This is now templated in `TargetT`. Moreover, `IterativeReconstruction` now use an objective function hierarchy. See the doxygen documentation for `GeneralisedObjectiveFunction`. This reorganisation has had repercusions in various other places. Examples are

- Class `OutputFileFormat` is now also templated to allow other types than images (or in fact `DiscretisedDensity<3,float>`).
- Class `ImageProcessor` is now replaced by the templated class `DataProcessor<DataT>`.
- Most functions in `buildblock/recon_array_functions.cxx` were specific to images samples on cartesian grids (or even (`VoxelsOnCartesianGrid<float>` objects), and hence could no longer be used. In addition, they had some ugly quircks (see section 2.6.5). So most of these functions have now been removed.
- Removed replace_extension and add_extension functions that work on char * as they were unsafe (in case the array did not have enough memory). Use std::string versions instead.

Other relatively important changes are listed below

- On request, the reconstruction hierarchy has now `get/set` functions for most parameters. Please email if some are missing.

- There is now a new class `AnalyticReconstruction`.
- `VectorWithOffset`: added constructors and 1 member to be able to have vectors that use existing data. This is useful when for example a C array with data needs to be manipulated.
- `KeyParser`:
  - use the 'parameter' member (of type boost::any) for storing all values, making par_int, par_double etc obsolete and enabling some nice use of macros to shorten the switch statements.
  - made IntVect and DoubleVect private typedefs (and removed UlongVect)
  - fix parameter_info() for types that output to more than 1 line (such as 2D arrays). We now add the usual continuation character (a backslash), such that the output of parameter_info can be parsed without problems.
  - in parameter_info(), deleted an extra newline after calling parameter_info() for a parsing object.
- `distributable`: no longer use global variables (for projectors etc) in distributable_computation. This means (among other things) a better chance of thread-safety, although there's still work to do there.
- in Reconstruction hierarchy: moved most checks from post_processing() to set_up() and clarified in doxygen which function is for what purpose
- `DiscretisedDensity`
  - renamed get_empty_discretised_density() to get_empty_copy(), using covariant return types
  - added has_same_characteristics()
  - added functions to convert between *physical* coordinates (in mm) and index coordinates.
- New global function `read_from_file()` using an input file format registry. This makes it much easier to add support for a new file format. See class `InputFileFormatRegistry`. This is currently not yet used for ProjData though.
- New variables defined for `make` for *AVW* support: `AVW_INCLUDE_DIR`, `AVW_LIBS` and `HAVE_AVW`. They are set in `config.mk` according to recommend *AVW* values, i.e. using the environment variables `AVW` and `TARGET`. See the Users Guide.

In addition, I have taken the opportunity to do some other bug fixing. The main affected area is the `Shape3D` hierarchy, where the Euler code had to be disabled and a different mechanism is now used to spcify direction vectors. This means that most of this code is now ready for templating in the number of dimensions. For other smaller changes, please refer to the ChangeLog.

3.1 Additional new functionality

This list is incomplete. Please refer to the ChangeLog.

- added `has_same_characteristics()` to `Sinogram`, `Viewgram`, `RelatedViewgrams`, `Segment*`
- added `operator==` to `Sinogram`, `Viewgram`, `RelatedViewgrams`, `Segment*`, `DiscretisedDensity` and symmetry classes. Updated `operator==` for `ProjDataInfo` classes to check also fields specific to the leaf classes.
- added `find_centre_of_gravity_in_mm` for `VoxelsOnCartesianGrid` objects.

3.2 Additional incompatibilities

This list is incomplete. Please refer to the ChangeLog.

- The `zoom_image` functions have now much better defined behaviour w.r.t the new index ranges and the origin (see the doxygen). In particular, the z-range now starts from index 0, and the origin is updated such that it remains physically correct. Another incompatibility is that previously, the multiple argument versions of `zoom_image` (i.e. specifying zoom etc explicitly) modified the image in place. Now, the function with the same name returns a new image instead. The previous functionality is now provided by `zoom_image_in_place`.
- `find_centre_of_gravity_in_mm_per_plane` now returns a result that takes `image.get_origin()` into account properly. It no longer shifts the z-direction to the centre of the image. There is currently no utility that depends on this behaviour though.