Eidolon quick start guide

By PINLab (2005)

Frederic Schoenahl, Nucl. Med. Division of Geneva Hospital

Table of Contents

Short Contents

1 Introduction

Here are a few general statements about the software you currently are working with.

1.1 The Eidolon software

1.1.1 Current release

Eidolon and the PINLab utilities and libraries are released by the PET Instrumentation and Neuroimaging laboratory at the Geneva University Hospital at the University of Geneva (http://dmnu-pet5.hcuge.ch). Its development intends to provide the nuclear imaging research community a bunch of tools for making development process of algorithms and processings easy. The suite is composed of core libraries, to be installed first and linked applications of various kind: image processing, reconstruction, simulation, etc. its development started in 2003.

An important aspect of the toolbox is that it is kept as simple as possible and a big effort in development is done. Existing libraries are reused for specific tasks, modular implementations allow extension of the toolbox. Reliability and portability is ensured by limiting the (newest) developments to pure ANSI-C and widespread GNU tools.

1.1.2 History

Eidolon development was started with the ParaPET project in 1999. In this framework, which has comprised the development of the STIR (ex: PPHEAD), Eidolon was designed as Monte Carlo simulator for generating realistic images.

1.1.3 General information

Eidolon is a research tools. It is not an end-user polished tool : Eidolon is intended to researchers having skills in development and having an idea on how to use a debugger and understand basics of programming. Here are a few points to understand the philosophy behind the software :

- **Code access** Any user of the Eidolon software is responsible for the validation process against a particular task. The code is easy to follow by a debugger or manually tracking. By a lack of time or for debugging purposes, some items are deactivated or hardcoded. It is of your responsability to evaluate the state of variables before using the code for studies. Features which really do not work are usually explicitely mentioned in this guide or in the code comments, if a feature is not working, check first if it has not been deactivated for validations purposes. Please contact authors for more informations and read carefully both code comments and available documentation.
- **Choice of the language** As the software was developped primary on NeXT workstations, the language of choice was Objective-C. This is an Object Oriented language with dynamic typing and an elegant messaging system. Objective-C can be understood as a syntaxic improvement of C language for object oriented programming, Objective-C interfaces easily with C and C++ libraries, and all non-object features (maths, file IO) are C language features.

  Objective-C is essentially characterized by a clear syntax, and now has excellent debugging support. Objective-C still is a reference language under Apple environments. Here follows a short example to play with a Photon object :

  ```
  id photon = [[Photon alloc] init];
  [photon propagate: 10. inDirection: [otherPhoton direction];
  ```

  In the example above, **photon** is the instance of the Photon class defined elsewhere. Any message (method call) passed to the photon comes just after and arguments follow. Arguments indicators (*inDirection:* above) can be given.

  ```
  id INSTANCE = [[CLASS alloc] init];
  [INSTANCE MESSAGE: arg1 ARG2INDICATOR: arg2]
  ```

Here, INSTANCE can be any class which has a method called 'MESSAGE:ARG2INDICATOR:' with two arguments, so you can manage objects lists and make them react in chain to a message. That's a fantastic issue which is really difficult to realize in statically typed languages like C++ (at least, would make the code really obscure).

It is a short example, please check some internet pages to get more info. Starting point could be http://objective-c.info where a few links and example codes are available.

- **Physics implementation** The influence of the influence of using different particle physics libraries for Eidolon for both interactions and cross-section libraries was investigated in Zaidi, 2000. The current implementation uses the GEANT 3 physics library in a highly optimized scheme. Routines were extracted from source code, validated in C and cross-section libraries were changed to look-up-tables to improve computational burden. Other cross-section libraries and interaction codes were used (XPHOT, XCOM, EPDL97, PETSIM...) in their field of purposes, the code is not active and stored in the `ScannerModel` subdirectory.

1.2 Authors

Code is issued by various contributors at PINLab or external. Project is written basically by H. Zaidi <habib.zaidi@hcuge.ch> in the framework of the ParaPET project, with code contributions from A. K. Hermann Sheurer. Now undertaken by F. Schoenahl <schoenahl@ieee.org> at Geneva University Hospital, division of Nuclear Medicine, PET and Neuroimaging Laboratory.

1.3 Distribution and licencing

Eidolon is for now only distributed to partner organisations under the following agreement.

```
This document is a temporary licence, but fully applicable.

This is a development version of EIDOLON which is experimental. Some
directories have been modified in order to preserve patents on several
proprietary code.

It is not allowed to distribute this software, modified or not, without
permission, and for commercial purposes. Please inform us of any change
that could improve this package  so that no divergent versions exist at
the same time.

The final release will be protected by the LGPL   (see LGPL.txt in this
directory). We are more restrictive on this particular version.

The  "rayleigh.c" and "compton.c"  routines were translated from fortran
to C, and taken from the GEANT3 software Monte Carlo library,   property
of CERN (http://www.cern.ch). Authors are

    R.Brun,  L.Urban, G.Tromba (*), P.Bregant (**)

   (*) now at: Sincrotrone Trieste, Padriciano 99, Trieste (I)
  (**) U.S.L. n.1 Triestina

adapted by

    H. Zaidi  and  A. K. Herman-Sheurer.

The licence that applies for these files is the same as for GEANT3 licen-
```

cing (see http://www.cern.ch) and code was used and is distributed with authorization from the authors above.

2 Requirements

- UNIX compatible system[1]
- GCC compiler with OBJECTIVE-C support (http://gcc.gnu.org)
- PINLib package from PINLab for image formats support

[1] compatibility essentially depends on PINLib package portability, PINLib package is still experimental

3 Compilation

This section deals with the installation of Eidolon on your system.

3.1 UNIX systems

Be sure you have the PINLib libraries installed prior to installing eidolon. In particular, the `libpin.a` library and the `libiol.a` library and modules. On unix systems, you need a *bash* compatible shell. After unzipping, cd into the eidolon directory:

```
$ cd eidolon
$ ./configure --with-pin-prefix=... --prefix=...
```

In the above command line, `--with-pin-prefix=...` or `-with-iol-prefix=...` allow to set the base path where PINLib is installed. `--prefix=...` allows to set up the target path (default : '`/usr/local`') Then type

```
$ make
```

to compile and

```
$ make install
```

to copy binaries to destination pathes. If the target path is located in the system folders, you must be logged as root user (command '`su`') prior to typing `make install`.

3.2 Other systems

For CYGWIN (`http://cygwin.com`), follow the same procedure as above. (TODO)

4 Usage

This part explains about basically running Eidolon.

4.1 Launch Eidolon

If the install (`--prefix=...`) was e.g. `/my/folder`, then be sure `/my/folder/bin` is in your `PATH` environment variable.

If not add it temporarly using :

```
$ export PATH="${PATH}":/my/folder/bin
```

or permanently by editing your '`~/.bash_profile`' or '`~/.bashrc`'.

If you have `libiol` support, and did not install it in a conventional directory (`/usr/lib` or `/usr/local/lib`) you can set the following variable prior to running Eidolon :

```
$ export LD_LIBRARY_PATH="${LD_LIBRARY_PATH}":/my/libraries
```

Copy the `SBRUN`, the `VBRUN`, or `HBRUN` the from the Eidolon folder to any location :

```
$ cp -r SBRUN /any/place
$ cd /any/place/SBRUN
```

These folder contain ready to run simulations for the three different simulation types : SBRUN for shape-based simulation, VBRUN for voxel-based simulation, and HBRUN for hybrid simulations.

From this directory you can launch a simulation. First edit the '`Parameters/setup.cfg`' to change the simulation prefix (default is **mySimulation**) and launch :

```
$ eidolon
```

When finished, dumps are available in '`/any/place/SBRUN/DATA`'.

- '**prefix**`.[S|scn]`' are sinogram
- '**prefix**`_SCX.S`' are scatter sinogram (`libIOL` only) $X = 1 \ldots 5$
- '**prefix**`_SCTOT.S`' is the total scattering sinogram (`libIOL` only)
- '**prefix**`.[V|img]`' are volume images (reference source distribution)
- '**prefix**`.dat`' are simulation statistics
- '**prefix**`_scatter_`*N*`.dat`' is scatter level N energy profile
- '**prefix**`_total.dat`' is total pairs spectrum
- '**prefix**`_true.dat`' is trues detected pairs spectrum
- '**prefix**`_coincidence_true.dat`'
- '**prefix**`_coincidence_false.dat`'

prefix is the prefix which is mentioned in the '`setup.cfg`' files. See Configuration. All data are made of one unique acquisition frame.

4.2 Visualize data

Reference images are always of dimension `128 x 128 x (NRINGS * 2 - 1)` (TODO check if not hard-coded to 47).

NOTE If you compile without `libIOL` support, the suffices are `.scn` for sinogram and `.img` for volumes. These are CTI ECAT6 format outputs.

With `libIOL` you get `.S` and `.V` files which are ECAT7 datasets. In CTI ECAT7 format, to display images in CTI (c) ECAT format, you can use XMedCon. (http://xmedcon.sourceforge.net).

To display sinograms, you can first process them with the PINLib `libIOL` utility `iol-pconvert` like this :

```
$ iol-pconvert -S -i dataset.S
```

the obtained file (default name is `sinogram,f0,s0.V`) is directly viewable using `XMedCon` as 3D ECAT7 image.

4.3 Reconstructing data

4.3.1 Manufacturer software

CTI © implementation for console software is called ECAT 7.2. It is possible to import both ECAT6 and ECAT7 format sinograms into the database for image reconstruction. A few points are to be considered however :

- Eidolon uses ideal normalization (unit response for all detectors) thus only geometrical effects depending on geometry are to be used by reconstruction algorithms. You therefore must deactivate all daily scanner normalization weightings prior to reconstruction. Note that some processing, like scatter correction use indirectly daily normalization. In order to remove normalization from the reconstruction process, open the sinogram in the `dbutil` tool, and edit header, choose to edit the subheader of first frame and set the *Corrections Applied* field value by '1' to say it has been normalized. From a terminal command prompt on the PET console, you can use `hdrEdit -f dataset.S` to edit directly a non-imported sinogram.
- ...

4.3.2 STIR software

`STIR` is an open source reconstruction software maintained by Dr. K. Thielemans et Al. from Hammersmith Imanet Ltd, London, UK. The base project page is at http://stir.sourceforge.net. STIR is shipped with several utilities that can be used to get data from Eidolon.

- **ECAT6** Use directly the '`stir_convecat6_if`' conversion utility supplied by STIR to generate a compatible CTI ECAT6 file. Then follow instructions for processing the DATA. This is the historical way `STIR` (previously called `PPHEAD`) was bound to Eidolon.
- **ECAT7** `STIR` should have been compiled with ECAT7 support from the Louvain-La-Neuve (LLN) library by M. Sibomana[1]. Check either the `STIR` *User's guide* and `STIR` mailing-list archive because this topic was often discussed. Then ECAT7 data should be converted using '`ifheaders_for_ecat7`' (this utility only exists in `STIR` if it was compiled with the Louvain-La-Neuve library support, read carefully the `STIR` user's guide).

  In order to convert the eidolon sinogram, e.g. '`MySimulation.S`', and if STIR is compiled from '`/home/fred/STIR`' then you will get your `STIR` compatible file by doing

  ```
  /home/fred/STIR/opt/utilities/ifheaders_for_ecat7 MySimulation.S
  ```

  and you get

  ```
  MySimulation.S
  MySimulation_S_f1g1d0b0.hs
  ```

  '`MySimulation_S_f1g1d0b0.hs`' is an interfile header which can be edited with an ASCII editor. At this point, please check the "Arc Correction" problem below, this header must be edited manually to have `STIR` not complaining about that issue. Note that the conversion does not affect the *data encoding* of sinograms, it simply links the data block and creates an (extended) interfile header file.

  Then create an ASCII file called e.g. '`recon.par`' in which you type :

  ```
  OSMAPOSLParameters :=

  input file := MySimulation_S_f1g1d0b0.hs
  output filename prefix := output
  ```

[1] Note that the LLN code is available for download from the `STIR` project site

```
projector pair type := Separate Projectors
  Projector Pair Using Separate Projectors Parameters:=
  Forward projector type := Ray Tracing
    Forward Projector Using Ray Tracing Parameters:=
    End Forward Projector Using Ray Tracing Parameters:=
  Back projector type := Interpolation
    Back Projector Using Interpolation Parameters:=
    End Back Projector Using Interpolation Parameters:=
  End Projector Pair Using Separate Projectors Parameters:=


sensitivity image:= sens.hv
zero end planes of segment 0:= 0

number of subsets:= 1
number of subiterations:= 24
Save images at subiteration intervals:= 3

END :=
```

Tune the parameters to your needs, or use one of the sample reconstruction files shipped with STIR (following above example, they are to be found in '`/home/fred/STIR/samples`'). Then run sensitivity program to generate sensitivity matrix :

```
/home/fred/STIR/opt/iterative/sensitivity/sensitivity recon.par
```

Chose ('0') for attenuation[2], and ('1') for normalization[3]. You will get a sensitivity header `sens.hv` and image data `sens.v`. You can view the sensitivity image by running

```
xmedcon sens.hv
```

Note that the prefix of the sensitivity image (here **sens**) can be adjusted to another name, however, don't forget to update adequate fields in the above '`recon.par`' file.

Finally, run reconstruction by using

```
/home/fred/STIR/opt/iterative/OSMAPOSL/OSMAPOSL recon.par
```

Then the iterative reconstruction process starts. If you get any problems, see the below troubleshooting list that should solve your issue.

```
$ /home/fred/STIR/opt/iterative/OSMAPOSL/OSMAPOSL recon.par

WARNING: Interfile warning: 'default bin size (cm)' (2.710000) is expected to be 0.3


WARNING: Interfile warning: I have used all explicit settings for the scanner
        from the Interfile header, and remaining fields set from the
        ECAT 925 model.


WARNING: Interfile parsing ended up with the following scanner:
Scanner parameters:=
Scanner type := ECAT 925
Number of rings                          := 24
```

[2] You must create an attenuation image if you want to have realistic reconstruction

[3] this matches Eidolon's overall unit response

```
Number of detectors per ring             := 384
Inner ring diameter (cm)                 := 82.5
Average depth of interaction (cm)        := 0.007
Distance between rings (cm)              := 0.675
Default bin size (cm)                    := 2.71
View offset (degrees)                    := 15
Maximum number of non-arc-corrected bins := 192
Default number of arc-corrected bins     := 192
Number of blocks per bucket in transaxial direction         := 4
Number of blocks per bucket in axial direction              := 3
Number of crystals per block in axial direction             := 8
...
```

Check that all warnings are not problematic for the validity of the reconstruction.

TROUBLESHOOTING There are many reasons why `STIR` could complain with Eidolon files.

- **Arc correction** You may have to add manually a field in the file header `*.hs` (edit it with a text editor). You must add "arc correction" in the header text to say the file has already been corrected for arc-correction : replace

  ```
  applied corrections := {none}
  ```

  should be replaced by

  ```
  applied corrections := {arc correction}
  ```

  or likewise

  ```
  applied corrections := {arc corrected}
  ```

- **Scanner type** Eidolon does not encode the scanner code name in CTI files headers (because CTI format usually not concerns General Electrics or Phillips systems you could implement in Eidolon. `STIR` uses the scanner information to generate the sensitivity matrix. If STIR does not request the scanner type, you must hardcode it in the Eidolon source codes by editing '`libiol/src/modules/mod-img-turku.c`'[4] if you have `libIOL`, or '`eidolon/src/Sinogram.bproj/sinoCTI.c`' otherwise. Note that if `STIR` does not complain, you should make sure the system is matching the simulated one by comparing your '`scanner.cfg`' with the `System type` field of the CTI header.

  With `libIOL` support, you can edit a field using CTI software `hdrEdit -f filename.S` to append required fields ("facility"). An easier way for setting-up this issue is under development.

- **Coordinate system** Eidolon and STIR have reversed y-axis direction, if you position a source at `x,y,z` in Eidolon, it will be seen as `x,-y,z` by STIR. You can visualize this effect by generating point source sinograms in STIR with `fwtest`.

NOTE You may have to add manually the missing fields in header and know about the system that has been simulated. For this, either use the `hdrEdit` utility from CTI, or hardcode the values directly in the PINLib suite (edit e.g. '`libiol/src/modules/mod-img-turku.c`'). New release will set this up auto matically. If you modify something in the `libiol` suite, you must recompile IOL, then recompile Eidolon.

[4] Note that if you modify anything in libiol, you must recompile, reinstall IOL and then reconfigure and recompile Eidolon

5 Configuration

This part explains about basically configuring Eidolon.

Please read Conventions section before this section. In this section, '--8<---' denotes the beginning and end of the configuration file and [...] a value. There is NO BLANK ROW allowed (even at the beginning of the file).

2/3D vectors are written

```
x1 x2 x3    * 3D vector values
```

without any comma or symbols between values. Only white spaces and tabs are allowed between values. Each file contains a configuration syntax as follows :

```
--8<-----------
[DATATYPE 1]     * COMMENT
[DATATYPE 2]     * COMMENT
... etc ...
--8<-----------
```

that means the value 1 of type data type DATATYPE with comment COMMENT. The following description :

```
--8<-----------
[DOUBLE 1]       * A single value
[DVECTOR3 2]     * A double vector with 3 slots
--8<-----------
```

is a template for example the following configuration file :

```
--8<-----------
3.14             * A single value
10. 11. 12.      * A double vector with 3 slots
--8<-----------
```

DVECTORs are DOUBLE valued vectors and IVECTORs are int valued vectors. Here follow the description how-to for the two main simulation types :

5.1 Shape based simulation

Type 1 simulations files required are

```
./DATA

./Parameters
    setup.cfg
    interactions.cfg
    scanner.cfg
    [acquisition.cfg]
    source.cfg
    scatter.cfg
```

Bracketed files are not mandatory for the simulation to be running, however, a warning will be issued and default parameters will be set.

WARNING the DATA folder must exist and is case sensitive, else, you may get a segmentation fault the END of your simulation ! (UPDATE) Since revision 28, the DATA and Parameter folders are created if they are missing in the current folder, however, the simulation will break, you must fill the Parameters folder with simulation `.cfg` files.

File description follows :

- '`setup.cfg`' contains :

```
--8<-----------
[INTEGER 1]     * Pseudo random number generator seed value
[STRING  2]     * Prefix to name all files generated by simulation
[INTEGER 3]     * Use '2' for SB simulations
--8<-----------
```

- '`interactions.cfg`' contains :

```
--8<-----------
[INTEGER 1]     * Takes values {0,1} for object scattering ON/OFF
[INTEGER 2]     * Takes values {0,1} for detector scattering ON/OFF
--8<-----------
```

- '`scanner.cfg`' contains :

```
--8<-----------
[INTEGER 1]     * Number of rings for this scanner
[INTEGER 2]     * Number of detectors per rings
[DOUBLE  3]     * Scanner radius (mm), FOVcenter to detector surface
[DOUBLE  4]     * Spacing between rings (mm)
[DOUBLE  5]     * Spacing between samples (bins) (mm)
[INTEGER 6]     * Maximum acquisition ring difference (3D)
[INTEGER 7]     * Number of radial projections (bins) for acquisition
[INTEGER 8]     * Number of views for acquisition (e.g.  N_det / 2 )
[DOUBLE  9]     * Detector energy resolution (  1 / sqrt(E) )
[DOUBLE  10]    * Lower energy threshold
[DOUBLE  11]    * Upper energy threshold
[DOUBLE  12]    * Detector radial dimension (thickness) (mm)
[STRING  13]    * Detector material string (see APPENDIX C)
[DOUBLE  14]    * Fan beam range (restricts LOR's)     (*)
[DVECTOR3 15]   * Detector spatial resolution (x y z  mm)
[DOUBLE 16]     * Object cut-off energy (KeV)
[INTEGER 17]    * Performs Arc Correction while acquiring (0|1)
--8<-----------
```

(*) this feature is not active with libPIN/IOL support.

- '`acquisition.cfg`' (optional) contains :

```
--8<-----------
[INTEGER 1]     * Radial compression (trim) factor  (unsigned >= 0) (*)
[INTEGER 2]     * Tangential or angular compression (mash) factor (unsigned >= 0)  (
[INTEGER 3]     * Axial compression (span) factor (unsigned >= 0) (*)
--8<-----------
```

(*) this feature is only necessary with PINLib/IOL support.

- '`source.cfg`' contains :

  This file is as long as necessary to contain various sources. Sources are simulated one after the other. You must construct your file ordered following this scheme :

```
X SHAPES
-- SHAPE 1
----> SHAPE DESCRIPTION
----> N excluded sources from SHAPE 1
----------> EXCLUDED SHAPE 1
----------------> description
----------> EXCLUDED SHAPE 2
----------------> description
```

```
    ...
----------> EXCLUDED SHAPE N
----------------> description
-- SHAPE 2
----> SHAPE DESCRIPTION
----> N excluded sources from SHAPE 2
----------> EXCLUDED SHAPE 1
----------------> description
----------> EXCLUDED SHAPE 2
----------------> description
    ...
----------> EXCLUDED SHAPE N
----------------> description
    ...     ...
-- SHAPE X
----> SHAPE DESCRIPTION
----> N excluded sources from SHAPE 2
----------> EXCLUDED SHAPE 1
----------------> description
----------> EXCLUDED SHAPE 2
----------------> description
    ...
----------> EXCLUDED SHAPE N
----------------> description
```

A SHAPE DESCRIPTION is like this

```
[STRING   ]     * Shape description {Cylindroid, Ellipsoid, Box}
[DVECTOR2 ]     * Tilt (polar rotation) as  Azimuth then Altitude
[DVECTOR3 ]     * Size of the shape [mm], see CONVENTIONS
[DVECTOR3 ]     * Center of the shape [mm], see CONVENTIONS
[DOUBLE   ]     * Axial aperture for the source [deg]  (*)
[INTEGER  ]     * Number of pairs emitted from this source
```

(*) Warning : aperture is calculated from source center! An EXCLUDED SHAPE DESCRIPTION is like this

```
[STRING   ]     * Null shape {Cylindroid, Ellipsoid, Box}
[DVECTOR2 ]     * Tilt (polar rotation) as  Azimuth then Altitude
[DVECTOR3 ]     * Size of the shape [mm], see CONVENTIONS
[DVECTOR3 ]     * Center of the shape [mm], see CONVENTIONS
```

Finally here is shown a typical configuration file, we have here the template for 2 sources shown as follow :

```
.S1             Shape 1
... E1      Excluded shape 1 from Shape 1
... E2      Excluded shape 2 from Shape 1
... E3      Excluded shape 3 from Shape 1
S2
... E1      Excluded shape 1 from Shape 2
```

in more details :

```
    --8<-----------
[INTEGER  1]     * Number of shapes (= 2)
[STRING   2]     * S1 Shape description {Cylindroid, Ellipsoid, Box}
[DVECTOR2 3]     * S1 Tilt (polar rotation) as Azimuth then Altitude
[DVECTOR3 4]     * S1 Size of the shape [mm], see CONVENTIONS
```

```
[DVECTOR3 5]     * S1 Center of the shape [mm], see CONVENTIONS
[DOUBLE   6]     * S1 Axial aperture for the source [deg]  (*)
[INTEGER  7]     * S1 Number of pairs emitted from this source
[INTEGER  8]     * S1 Number of shapes to be excluded FROM S1
[STRING   8.1]   *  E1 Null shape {Cylindroid, Ellipsoid, Box}
[DVECTOR2 8.2]   *  E1 Tilt (polar rotation)
[DVECTOR3 8.3]   *  E1 Size of the shape [mm], see CONVENTIONS
[DVECTOR3 8.4]   *  E1 Center of the shape [mm], see CONVENTIONS
[STRING   8.1]   *  E2 Null shape {Cylindroid, Ellipsoid, Box}
[DVECTOR2 8.2]   *  E2 Tilt (polar rotation)
[DVECTOR3 8.3]   *  E2 Size of the shape [mm], see CONVENTIONS
[DVECTOR3 8.4]   *  E2 Center of the shape [mm], see CONVENTIONS
[STRING   8.1]   *  E3 Null shape {Cylindroid, Ellipsoid, Box}
[DVECTOR2 8.2]   *  E3 Tilt (polar rotation)
[DVECTOR3 8.3]   *  E3 Size of the shape [mm], see CONVENTIONS
[DVECTOR3 8.4]   *  E3 Center of the shape [mm], see CONVENTIONS
[STRING   9]     * S2 Shape description {Cylindroid, Ellipsoid, Box}
[DVECTOR2 10]    * S2 Tilt (polar rotation) as Azimuth then Altitude
[DVECTOR3 11]    * S2 Size of the shape [mm], see CONVENTIONS
[DVECTOR3 12]    * S2 Center of the shape [mm], see CONVENTIONS
[DOUBLE   13]    * S2 Axial aperture for the source [deg]  (*)
[INTEGER  14]    * S2 Number of pairs emitted from this source
[INTEGER  15]    * S2 Number of shapes to be excluded FROM S2 ( = 1)
[STRING   15.1]  *  E1 Null shape {Cylindroid, Ellipsoid, Box}
[DVECTOR2 15.2]  *  E1 Tilt (polar rotation)
[DVECTOR3 15.3]  *  E1 Size of the shape [mm], see CONVENTIONS
[DVECTOR3 15.4]  *  E1 Center of the shape [mm], see CONVENTIONS
    --8<----
```

(*) Warning : aperture is calculated from source center!

- '`scatter.cfg`' contains :

```
--8<-----------
[STRING   1]    * Scatter description {Cylindroid, Ellipsoid, Box}
[DVECTOR2 2]    * Tilt (polar rotation) as  Azimuth then Altitude
[DVECTOR3 3]    * Size of the medium [mm], see CONVENTIONS
[DVECTOR3 4]    * Center of the medium [mm], see CONVENTIONS
[STRING 10]     * Medium material string (see APPENDIX A,B)
--8<-----------
```

Note - multi shaped scatterers are not available (removed from code)

5.2 Voxel based simulation

as for VB simulation, but require an additional "scatter.cfg" file to describe the medium distribution.

```
./DATA

./Parameters
    setup.cfg
    interactions.cfg
    scanner.cfg
    [acquisition.cfg]
    voxel.cfg
```

```
./Phantoms
    <EMPHANTOM>.raw
    <TXPHANTOM>.raw
```

Bracketed files are not mandatory for the simulation to be running, however, a warning will be issued and default parameters will be set.

WARNING the DATA folder must exist and is case sensitive, else, you may get a segmentation fault the END of your simulation ! (UPDATE) Since revision 28, the DATA and Parameter folders are created if they are missing in the current folder, however, the simulation will break, you must fill the Parameters folder with simulation `.cfg` files.

File description follows :

- '`setup.cfg`' contains :

```
--8<-----------
[INTEGER 1]     * Pseudo random number generator seed value
[STRING  2]     * Prefix to name all files generated by simulation
[INTEGER 3]     * Use '1' for VB simulations
--8<-----------
```

- '`interactions.cfg`' contains :

```
--8<-----------
[INTEGER 1]     * Takes values {0,1} for object scattering ON/OFF
[INTEGER 2]     * Takes values {0,1} for detector scattering ON/OFF
--8<-----------
```

- '`scanner.cfg`' contains :

  Same as for SB simulations, see above

- '`acquisition.cfg`' (optional) contains :

  Same as for SB simulations, see above

- '`voxel.cfg`' contains : The difference with SB simulation is that there is no TX phantom path or information in this file.

```
--8<-----------
[STRING  1]     * Emission phantom path               (*)
[INTEGER 2]     * EM phantom bytes per element: takes values {1,2}
[STRING  3]     * Transmission phantom path              (*)
[INTEGER 4]     * TX phantom bytes per element: takes values {1,2}
[IVECTOR 5]     * Phantom dimensions                  (**)
[DVECTOR 6]     * Phantom voxels dimensions (mm)
[DOUBLE  7]     * Axial shift centering phantom (mm) cf CONVENTIONS
[DOUBLE  8]     * Axial aperture for emission  [0 90] cf REMARKS
[INTEGER 9]     * Total number of pairs to simulate   (***)
--8<-----------
```

  (*) These are either absolute pathes or pathes related to the location from which eidolon is launched (i.e. '.'). For example if you launch Eidolon from '`/home/fred/.`' , and "Emission phantom path" is '`Phantom/myPhantom.raw`' then the '`/home/fred/Phantom`' path should exist with a '`myPhantom.raw`' file inside. (**) X and Y dimensions should be equal. Moreover, TX and EM dimensions have to match. (***) That number is buffered and therefore the minimal number is 1000 and maximal number is ULONG_MAX (18446744073709551615).

- '`scatter.cfg`' contains :

  Same as for SB simulations, see above

5.3 Hybrid voxel/shape based simulation

The '`source.cfg`' file is replaced by a '`emvoxel.cfg`' file that links both the source distribution (EM). It is similar to the above '`voxel.cfg`' without transmission phantom information, which is given by the '`scatter.cfg`'.

Type 3 simulations files required are

```
./DATA

./Parameters
    setup.cfg
    interactions.cfg
    scanner.cfg
    [acquisition.cfg]
    scatter.cfg
    emvoxel.cfg

./Phantoms
    <EMPHANTOM>.raw
```

Bracketed files are not mandatory for the simulation to be running, however, a warning will be issued and default parameters will be set.

WARNING the DATA folder must exist and is case sensitive, else, you may get a segmentation fault the END of your simulation ! (UPDATE) Since revision 28, the DATA and Parameter folders are created if they are missing in the current folder, however, the simulation will break, you must fill the Parameters folder with simulation `.cfg` files.

Note that the `Phantom` folder only depends on if you use it in your

Details of file contents is as follows :

- '`setup.cfg`' contains :

```
--8<-----------
[INTEGER 1]     * Pseudo random number generator seed value
[STRING  2]     * Prefix to name all files generated by simulation
[INTEGER 3]     * Use '3' for HV simulations
--8<-----------
```

- '`interactions.cfg`' contains :

```
--8<-----------
[INTEGER 1]     * Takes values {0,1} for object scattering ON/OFF
[INTEGER 2]     * Takes values {0,1} for detector scattering ON/OFF
--8<-----------
```

- '`scanner.cfg`' contains :

  Same as for SB simulations, see above

- '`acquisition.cfg`' (optional) contains :

  Same as for SB simulations, see above

- '`emvoxel.cfg`' contains :

```
--8<-----------
[STRING  1]     * Emission phantom path               (*)
[INTEGER 2]     * EM phantom bytes per element: takes values {1,2}
[IVECTOR 3]     * Phantom dimensions                  (**)
[DVECTOR 4]     * Phantom voxels dimensions (mm)
[DOUBLE  5]     * Axial shift centering phantom (mm) cf CONVENTIONS
```

```
[DOUBLE  6]     * Axial aperture for emission  [0 90] cf REMARKS
[INTEGER 7]     * Total number of pairs to simulate   (***)
--8<-----------
```

(*) These are either absolute pathes or pathes related to the location from which eidolon is launched (i.e. '.'). For example if you launch Eidolon from '`/home/fred/.`' , and "Emission phantom path" is '`Phantom/myPhantom.raw`' then the '`/home/fred/Phantom`' path should exist with a '`myPhantom.raw`' file inside. (**) X and Y dimensions should be equal. Moreover, TX and EM dimensions have to match. (***) That number is buffered and therefore the minimal number is 1000 and maximal number is ULONG_MAX (18446744073709551615).

5.4 Hybrid shape/voxel based simulation

TODO

6 Documentation

- *Generate the quick start guide* You need to have a LaTeX system available for generating Adobe PDF format guide and to have the `makeinfo` command. Then type from the Eidolon directory :

  ```
  $ make -C doc pdf
  ```

  resulting in `doc/eidolon.pdf` file. Typing

  ```
  $ make -C doc html
  ```

  generates an HTML folder browsable from `./doc/eidolon.html/index.html`. Browsable documentation gets installed while installing eidolon and is callable with

  ```
  $ info eidolon
  ```

- *For developer documentation* To compile autodocumentation you need objcdoc available at

  http://www.informatik.uos.de/elmar/projects/objcdoc

  A copy of this software is available in the `./doc` directory of the `eidolon` folder.

7 Conventions

The following conventions should be taken into account prior to writing configuration files.

7.1 Units

Unit is [mm] for distances, [keV] for energies[1]. Tilting angles are in [rad] but aperture angle is expressed in [deg]. Note that internally, Eidolon uses [rad] everywhere.

7.2 Coordinate systems

The coordinate system in Eidolon is as described as follow :

- X0 is in the center of the field of view and x axis oriented towards right side of the scanner
- Y0 is in the center of the field of view and y axis oriented towards upper side of the scanner
- Z0 is at the proximal side of the field of view, at zero'th plane and oriented toward last plane (plane max)

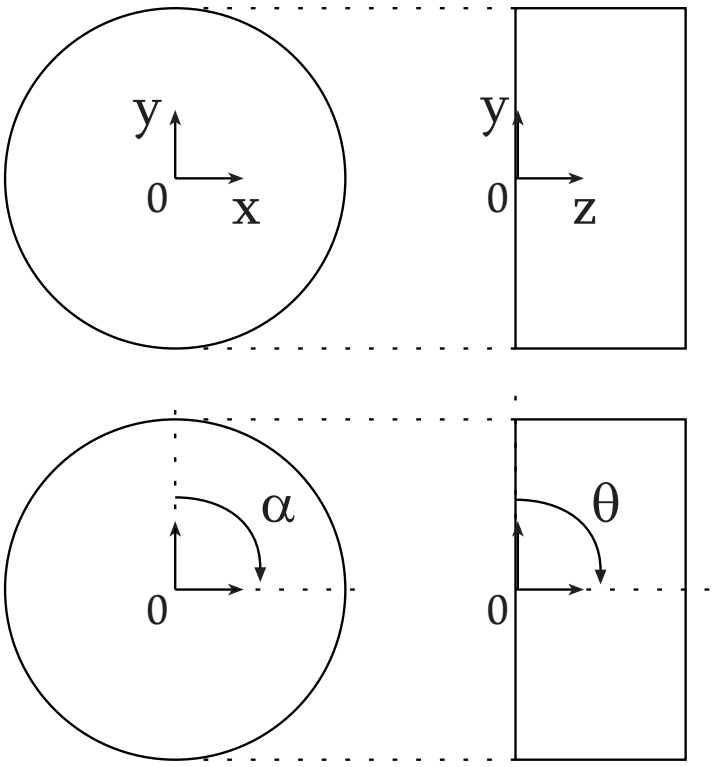

Figure 7.1: Coordinate system conventions for a transaxial (left) and axial (right) view.

Example If the scanner has radii 50.0 mm and axial FOV 80.0 mm the isocenter of the field of view is [0.0 0.0 40.0]

7.3 Objects parametrization

Dimensions of shapes object are understood as "radii". Namely, if you want to position a rod source of diameter 1mm and length 100 mm, you should give it dimensions [0.5 0.5 50.0] whatever the shape is (box, cylindroid, ellipsoid).

[1] GEANT3 requires MeV and conversion is done within Eidolon to pass values in MeV

7.4 Voxel blocks parametrization

A voxel attenuation medium is a 3D raw image. Encoding (little or big endian) should be the same as the runtime system encoding. i.e. generate your phantoms on the same system you run eidolon. For recall, PC's usually encode little endian and SPARC stations encode in big endian. If you have a mistake in encoding, eidolon returns an error message complaining about the fact that the material is unknown.

- **EMISSION phantom design** : the emission phantom contains 8 or 16 bits encoded data (signed char 8 or signed int 16) in C ordering. First element is the X = left, Y = up, Z = proximal plane of the voxel block. Internally the emission phantom is a *double* valued C-vector.
- **TRANSMISSION phantom design** same organisation. Internally, data is stored as *signed short int* (16 bits) vector.

Note that the X and Y dimensions of your phantoms should be identical (TODO).

Due to the choice of the coordinate system (Z0 = first slice), you must place the phantom emission and transmission at the center of the field of view. To calculate the location of the phantom to be centered, just compute the Z shift

$Z_s = (N_r \cdot s - (D_z/2) \cdot v_z)/2$

- Z_s is the Z shift in [mm]
- N_r is the number of rings for this scanner
- s is the interring spacing
- D_z is the dimension Z of the phantom (twice the radius)
- v_z is the voxel dimension Z of the phantom

Example for an CTI ART scanner with 24 rings and a $s = 6.75$ mm, and a 128 x 128 x 64 phantom with voxel size 2 x 2 x 4 mm we get

$Z_s = (24. \cdot 6.75 - (128./2.) \cdot 4.)/2. = -47mm$

8 Algorithms

The following algorithms were used in the software.

8.1 Random numbers

Random numbers are generated from pseudo-random sequences. (todo)

8.2 Random photon properties

8.2.1 Random location

8.2.1.1 Shape based emission

Basic objects as encoded in `Shape3d.m`

8.2.1.2 Voxel based emission

8.2.2 Random directions

From the origin (pair birth location), an event gets a direction with altitude and azimuth with the following properties :

- azimuth : $\alpha \in [-\pi : +\pi]$
- altitude : $\theta \in [-\pi/2 : +\pi/2]$

The aperture should be given in the `source.cfg` or `voxel.cfg` configurations files whatever the simulation is in degrees. This angle reduces the axial angle aperture (altitude) `A` [rad] using the following statement

$$\alpha = \sin^{-1}(\sin(A \cdot r))$$

the angle is converted into radian internally.

8.3 Physical processes

8.4 Geometry and ray-tracing

8.4.1 Siddon algorithm

Coordinates are first mapped to a simpler coordinate system where the top-left edge of the image becomes the reference zero.

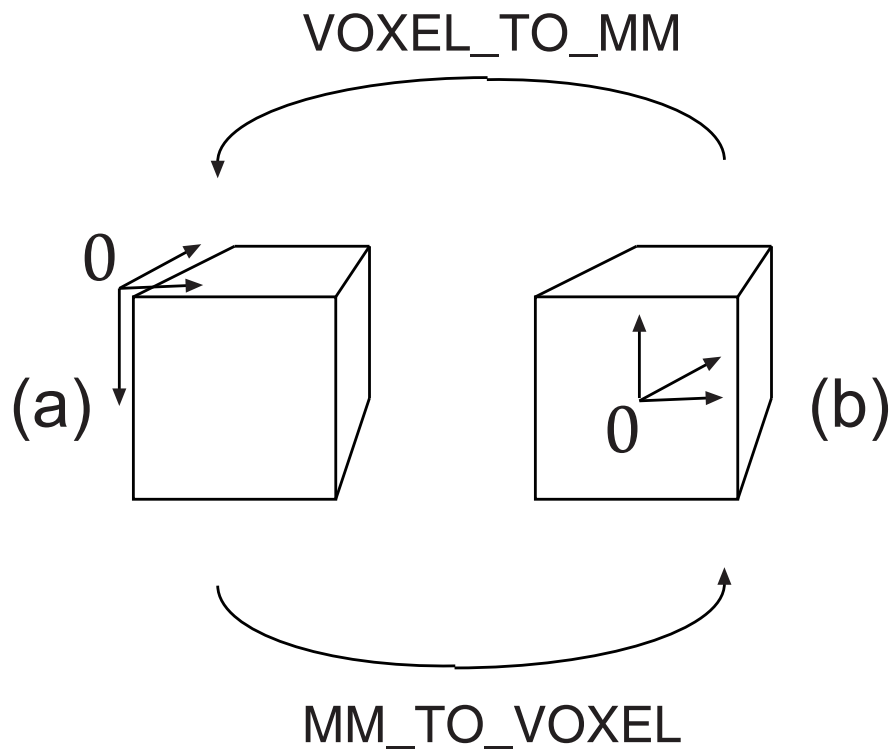

Figure 8.1: Coordinate system conventions for transforming into image coordinate systems (a) and original coordinate system (b)

The **MM_TO_VOXEL** transformation runs as follow :

- $x' = x + N_x/2$
- $y' = -y + N_y/2$
- $z' = z - S_z$

where $N_.$ is the length of the image block per given dimension and S_z the transaxial shift as given by user for adjusting the image - scanner centers.

The **VOXEL_TO_MM** transformation runs inversely :

- $x = x' - N_x/2$
- $y = y' - N_y/2$
- $z' = z + S_z$

8.5 Data storage

8.5.1 Arc Correction

Arc correction can be triggered on or off using the `scanner.cfg` configuration file present in the `Parameters/` directory. Arc-correction is performed following Townsend & Defrise, 1992 CERN technical report :

$s = R_d \sin\left(\left(2 \cdot \pi \cdot k\right)/N_d\right) \Rightarrow k = \arcsin\left(\left(2 \cdot s\right)/R_d\right) \cdot N_d/\pi$

9 Obsolete

9.1 NeXT

To compile on NeXT systems add in Makefile

```
-D _next
```

However, NeXT support is not really pursued. Preferred run-time is GNU objective-C implementation and NeXT functions use is not garanteed to work...

9.2 Interface

To compile with interface support add in Makefile

```
-D _INTERFACE
```

Note that interfacing has to change, it was released under NeXTStep. Some work has been done with integrating GNUStep (http://gnustep.org) and developing Python-TKinter (http://python.org) however the project is not mature yet and easy integration is a strong requirement for the interface system.

9.3 PVM Parallel computations

PVM was initially implemented in Eidolon and is available in defining

```
-D PVM_DELEGATE
```

however, Eidolon is fast and optimized in the physics parts, and distribution over processors is not really a priority for speed-up. PVM support is not garanteed, however, no significant changes have been done on the distributed computation system while it was implemented and working.

10 Appendices

10.1 Medium materials table

These materials are implemented in '`ScatterModel.subproj/ScattererChemistryMethods.m`' their definition is in '`ScatterModel.subproj/Scatterer.h`'. These are for simple voxel based phantoms.

Voxel value	Configuration code	Material
1	H2O	Water
2	AIR	Air
3	AL	Aluminium
4	PB	Lether
5	SOFTTISSUES	Human soft tissues
6	BRAIN	Brain tissues
7	BONE	Bone tissue
8	LUNG	Lungs tissue

10.2 Detector materials table

These materials are implemented in '`ScatterModel.subproj/ScattererChemistryMethods.m`' their definition is in '`ScatterModel.subproj/Scatterer.h`'. These are for Zubal voxel phantoms.

Configuration code	Crystal material
BGO	Bismuth Germanate
NaI	...
BaF2	...
LuAP	...
LSO	...
CsI	...

10.3 Zubal materials table

These crystal types are implemented in '`ScannerModel.subproj/ScannerModel.h`'

TODO