ANTS2 toolkit:
Detector optimization and experimental data processing for position sensitive scintillation detectors

Andrey Morozov (LIP-Coimbra)

Coimbra, November 2018

Content

- Team, history and target applications
- Position sensitive scintillation detectors (PSSD)
- Reconstruction module
- Simulation module
- Semi-automatic detector optimization
- Implementation details
- Future?

ANTS2 development team

Developers:

Andrey Morozov
Vladimir Solovov
Raimundo Martins (not active anymore)

Neural network module:

Francisco Neves

Consultant:

Vitaly Chepel

Work on ANTS2 has started in 2013 based on

- ANTS package developed at the neutron detector group of LIP
- B-spline LRF parameterization algorithms developed by V.Solovov for LUX collaboration.

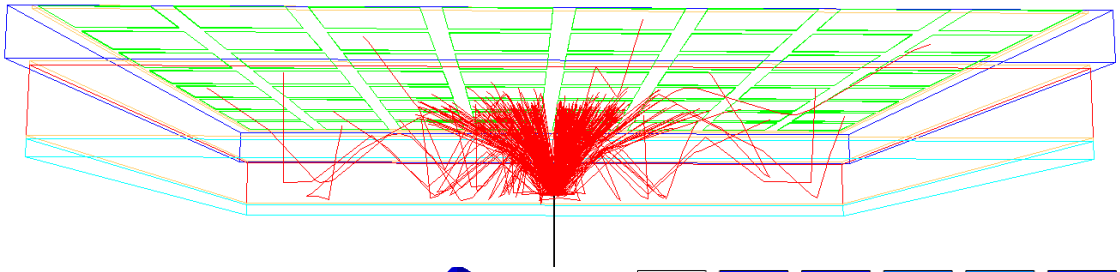
ANTS2 applications

- Development and optimization of PSSDs
- Optimization of event reconstruction techniques
- Reconstruction of the response of light sensors
- Reconstruction of scintillation events
- Evaluation of optical photon and neutron transport

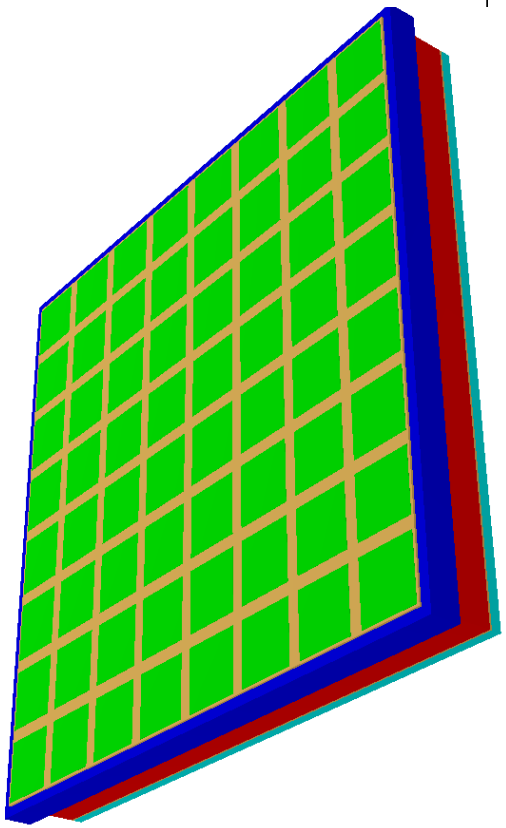
Position sensitive scintillation detectors (PSSD)

PSSD examples

Medical gamma camera



Gamma ray interaction resulting in emission of optical photons

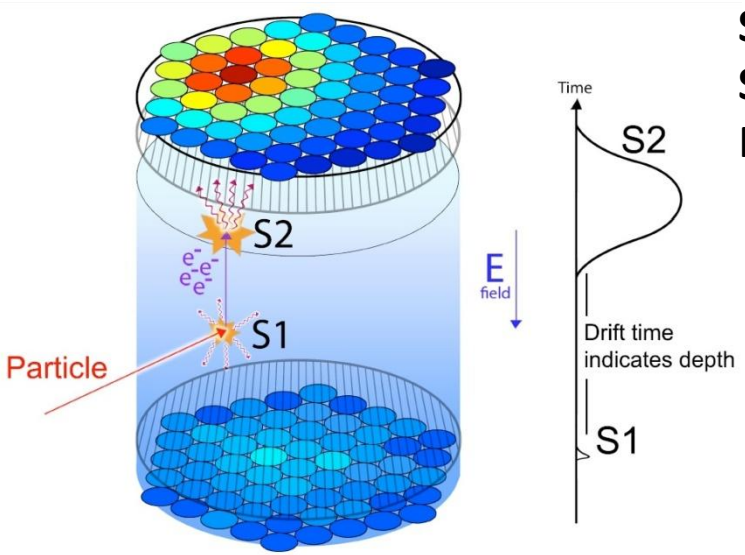


0	1	1	11	13	2	0	0
0	1	12	58	78	9	1	1
0	0	12	68	97	6	1	0
1	0	1	13	10	3	1	0
0	1	0	1	0	1	0	0
0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1
0	0	0	0	1	0	0	0

Distribution of the sensor signals is used to reconstruct **XY position and energy** of the scintillation event.

Another example with the same geometry:
thermal neutron detector with ^{6}Li glass scintillator

Dual-phase dark matter detectors



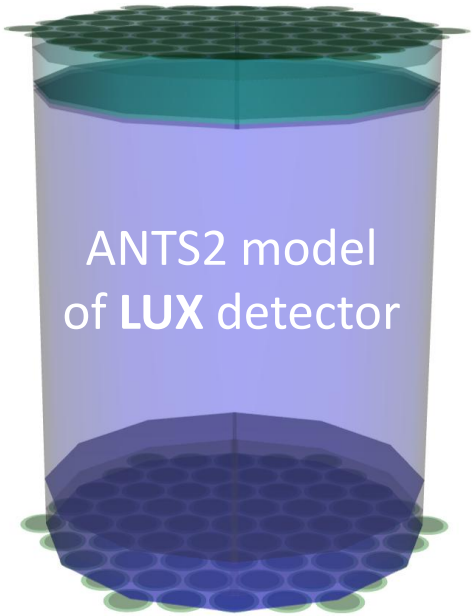
S1 - Primary scintillation

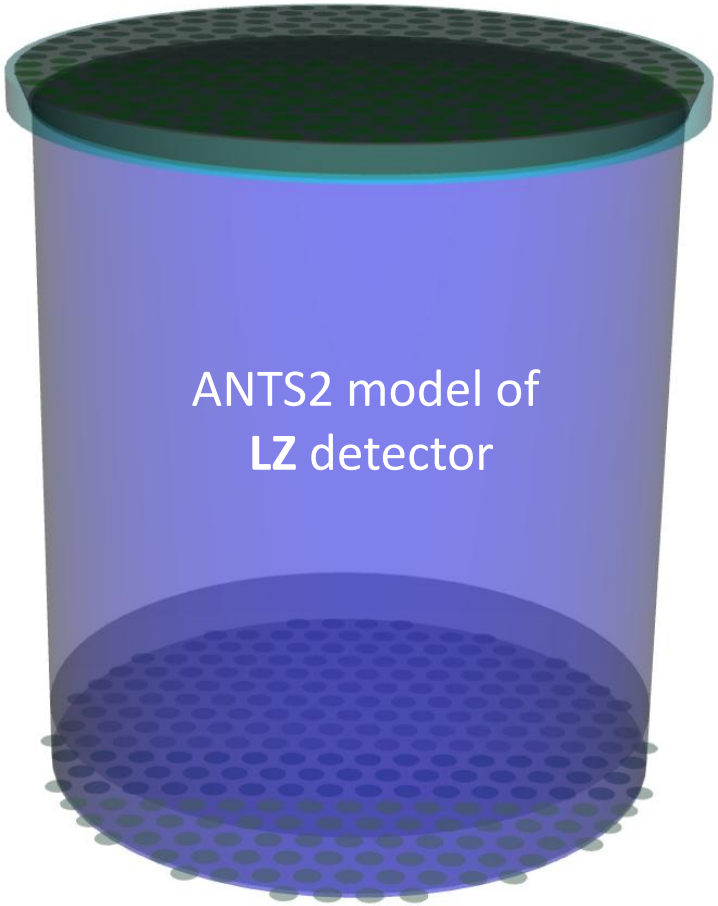
S2 - Secondary scintillation (proportional to charge)

Position sensitivity:

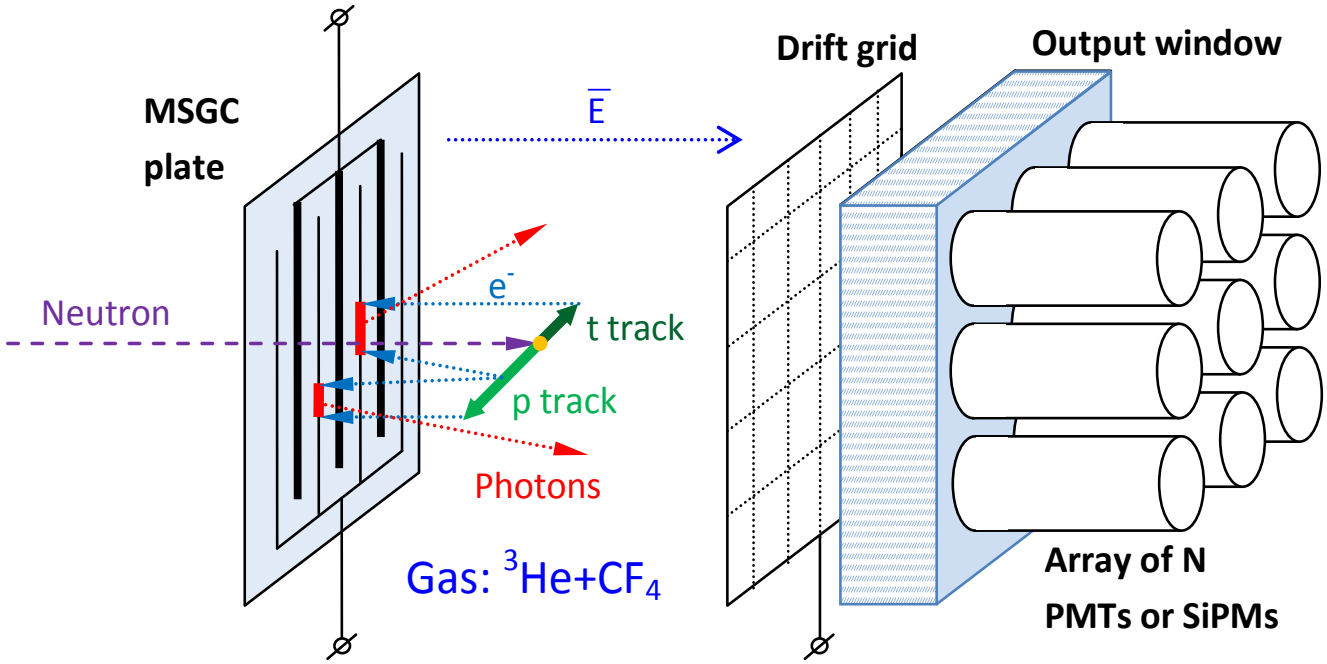
X and **Y** from the S2 light distribution

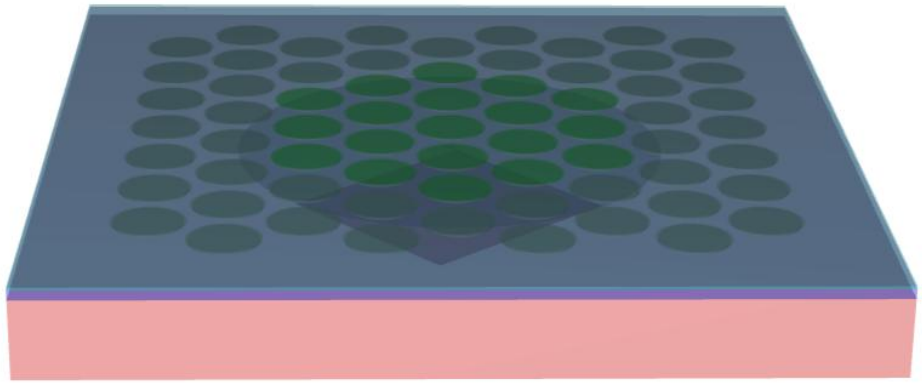
Z from the drift time (delay between S1 and S2)





Thermal neutron scintillation detectors



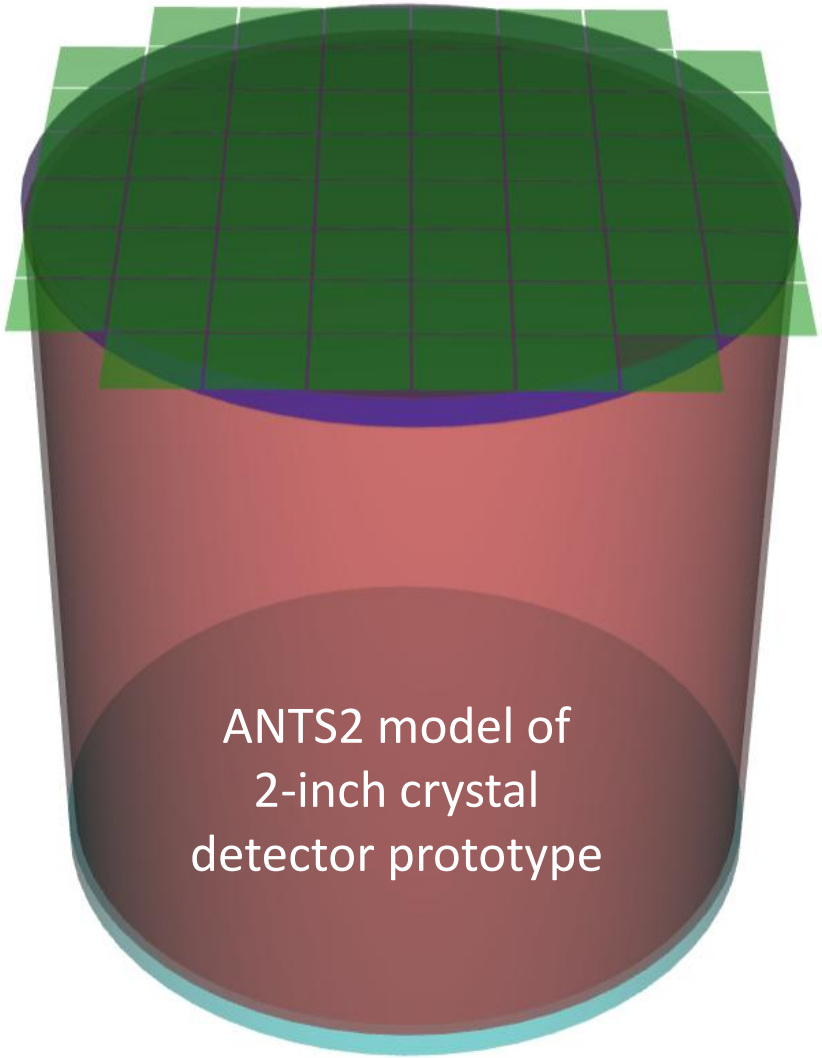


ANTS2 model of GSPC-19 detector

$LaBr_3$ scintillation detector

The detector is intended for:

- Studies of short-lived radioisotopes at relativistic velocities.
- Absorber of the Compton camera for radiotherapy monitoring.



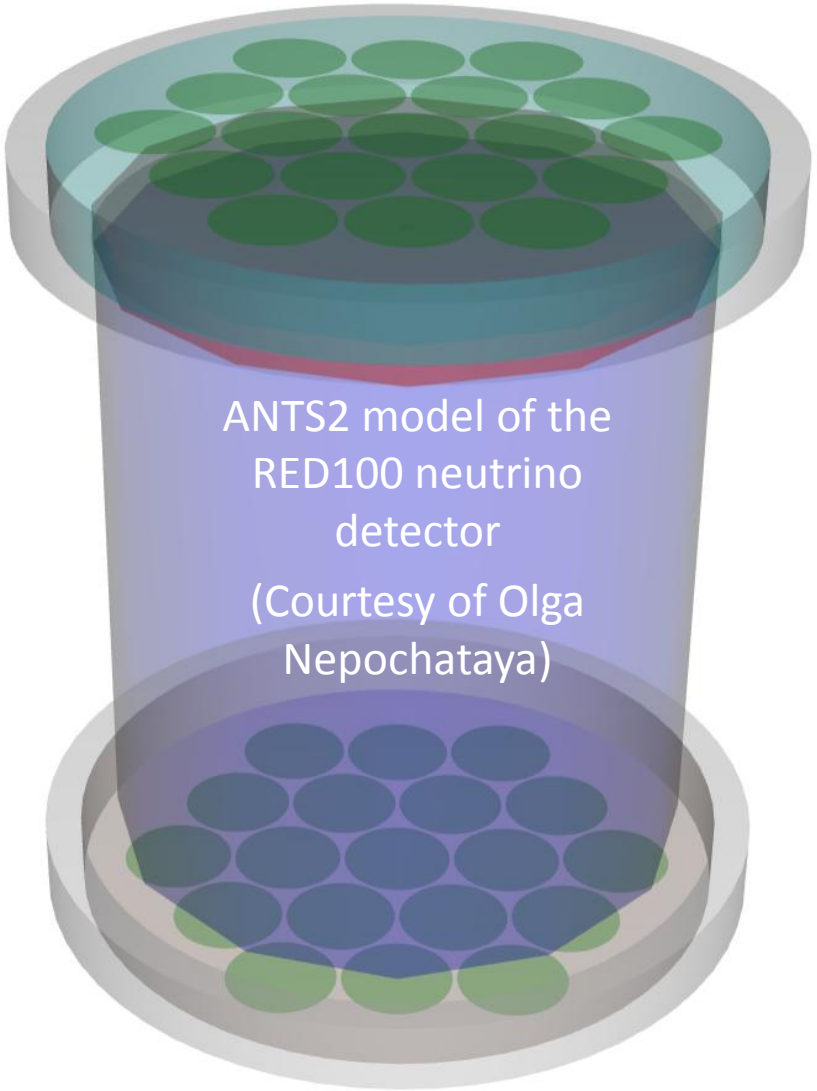
Cooperation with **Politecnico di Milano**

Neutrino detectors

Design is “inherited” from the dual-phase dark matter detectors.

Coherent scattering of a neutrino on a Xe nucleus results in recoil of the nucleus, leading to generation of ionization electrons in liquid xenon.

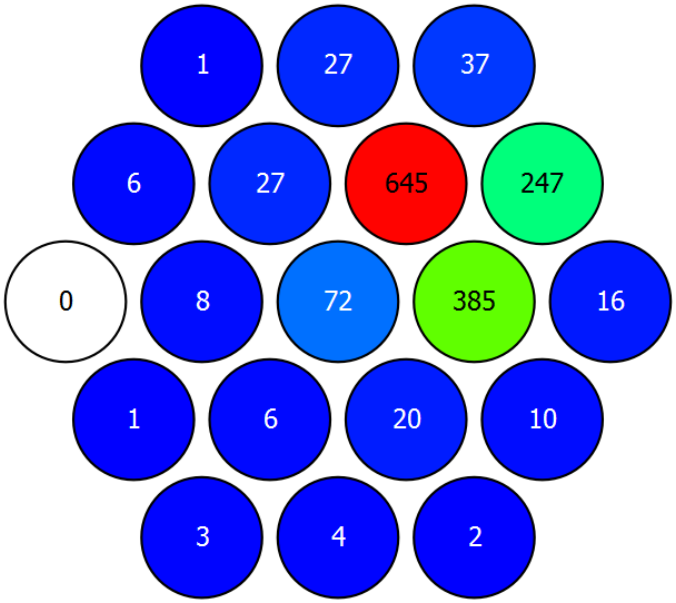
With dual-phase design, it is possible to observe single ionization electrons extracted from the liquid.



Cooperation with **MEPhI** (Moscow)

Reconstruction module

Center of Gravity method



Light sensor signal distribution for a scintillation event

Traditional approach since 1957 in PSSDs:

Center of Gravity (CoG) method

$$x = \frac{\sum X_i S_i}{\sum S_i} \qquad y = \frac{\sum Y_i S_i}{\sum S_i}$$

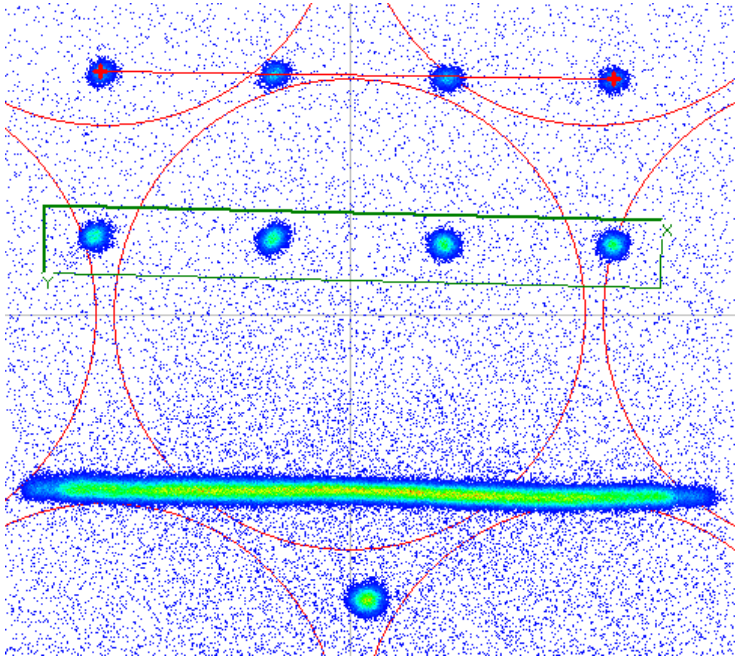
Robust and simple to implement

However, has serious drawbacks:

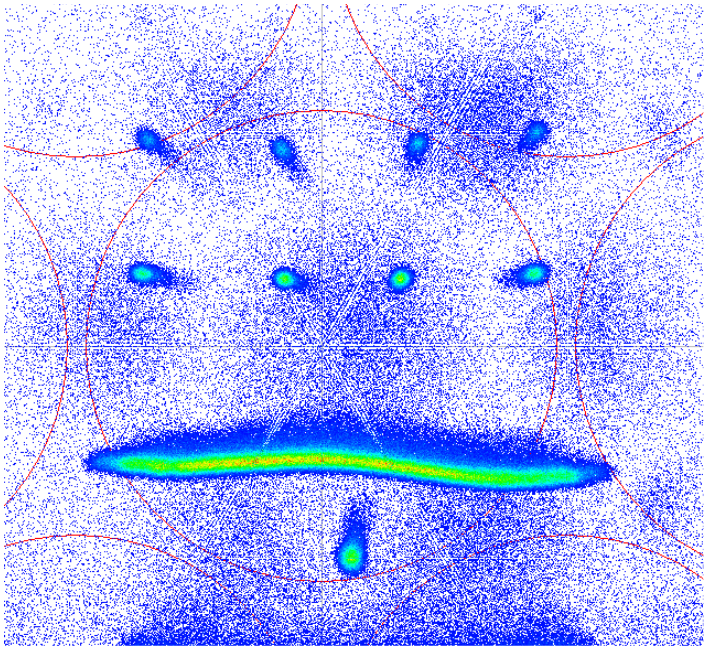
- Systematic distortions in position and energy
- Sensitive to gain drift
- Resolution typically below the limit given by the photon statistics
- Noise event discrimination can be difficult

Center of gravity is sufficient?

Expectation:



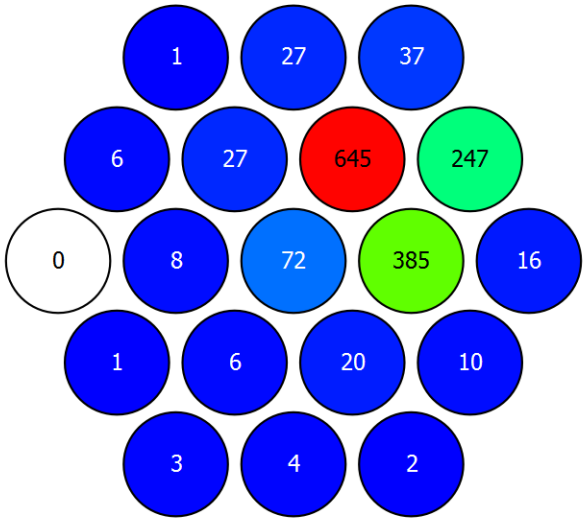
Center of Gravity reconstruction
(PMT gains are already equalized!):



Could be a long journey to meet the expectations!

Statistical reconstruction

Measured signals

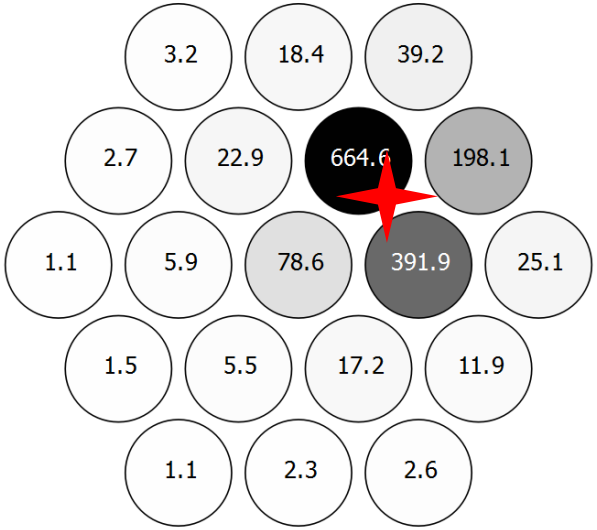


Statistical methods reconstruct events by finding the best match between the

- measured sensor signals and
- expected signals $S_i(x, y, energy)$

Potentially distortion-free and can give the best possible resolution

Expected signals



Requires detailed knowledge of the detector response!

Response parameterization

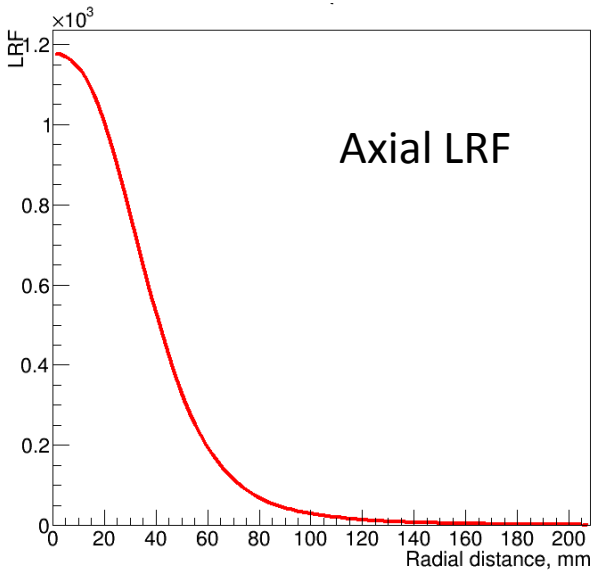
ANTS2 uses **Light Response Functions** (LRF) to define detector response.

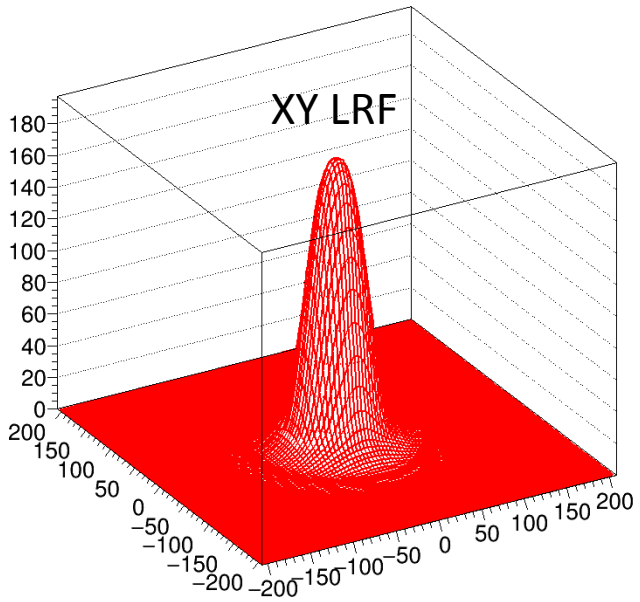
LRF gives the dependence of the average signal of a sensor on the light source position.

ANTS2 offers several approaches to parameterize LRFs:

1. Cubic B-splines

LRF types can cover 1D (Axial symmetry), 2D (XY or Axial + Z) and 3D cases





LRF parameterization

2. Analytic functions

A script can be used to define the parameterization function, e.g.:

```
function eval(r)
{
    return A * Math.exp( -0.5 * r * r / S2) + C
}
```

3. Custom (plug-ins)

Custom LRF types can be provided using the plug-in interface of ANTS2.

4. Composite

The LRF of each sensor can be configured as a sum of several LRF components, each of which is parameterized with one of the schemes described above.

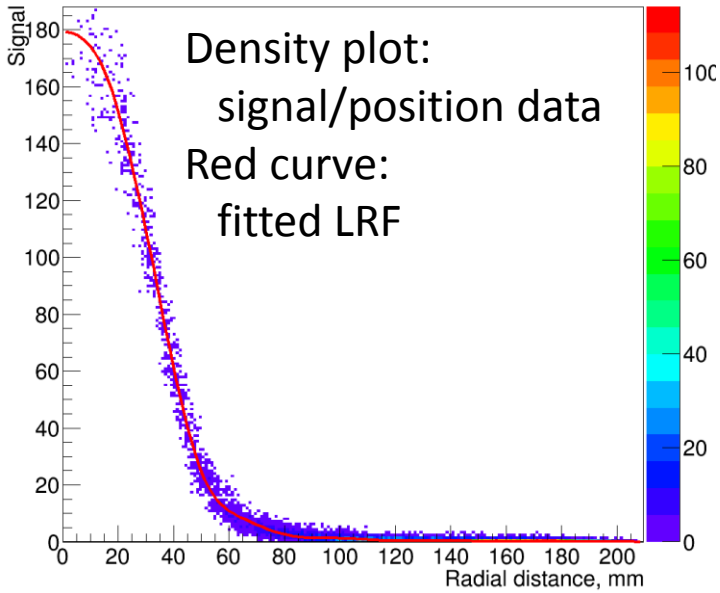
LRF fitting

LRF fitting requires two datasets:

- **signals** of the sensors
- corresponding source **positions**

The position data can be

- **true** positions of the light source
  (simulations or scan with a pencil beam)
- **reconstructed** positions



ANTS2 can take into account the symmetry of the sensor arrays
to group light sensors and use a common LRF profile for several sensors.

B-spline library allows to set constraints on the LRF profile, e.g.:

- non-increasing function
- always positive
- flat-top

Statistical reconstruction in ANTS2

The **difference** between the signals and the model prediction:

- Least Squares
- Maximum Likelihood

Implementation of the **search** of the best match:

- Simplex or Migrad minimizers
- Contracting grids
  - On CPU using multithreading
  - On GPU using CUDA (custom kernerls):
    reached rates of ~10^6 events per second for 37 PMT detector!

Scripting tools (with multithreading capability) are provided to implement custom solutions

ANN and kNN-based reconstruction

Artificial neural networks (ANN)

The ANN module of ANTS2 can

- Configure and train the network using a calibration dataset
- Use the trained network for event reconstruction

The cascade algorithm allows to automatically find the best ANN topology.

kNN search

- Operates in a multidimensional space, where each dimension represents signal value of one of the sensors (sensor space)
- Detector response model is represented by a set of calibration events with known positions
- To reconstruct an event:
  - find ***k*** nearest calibration events
  - calculate centroid of their positions

PSSD calibration

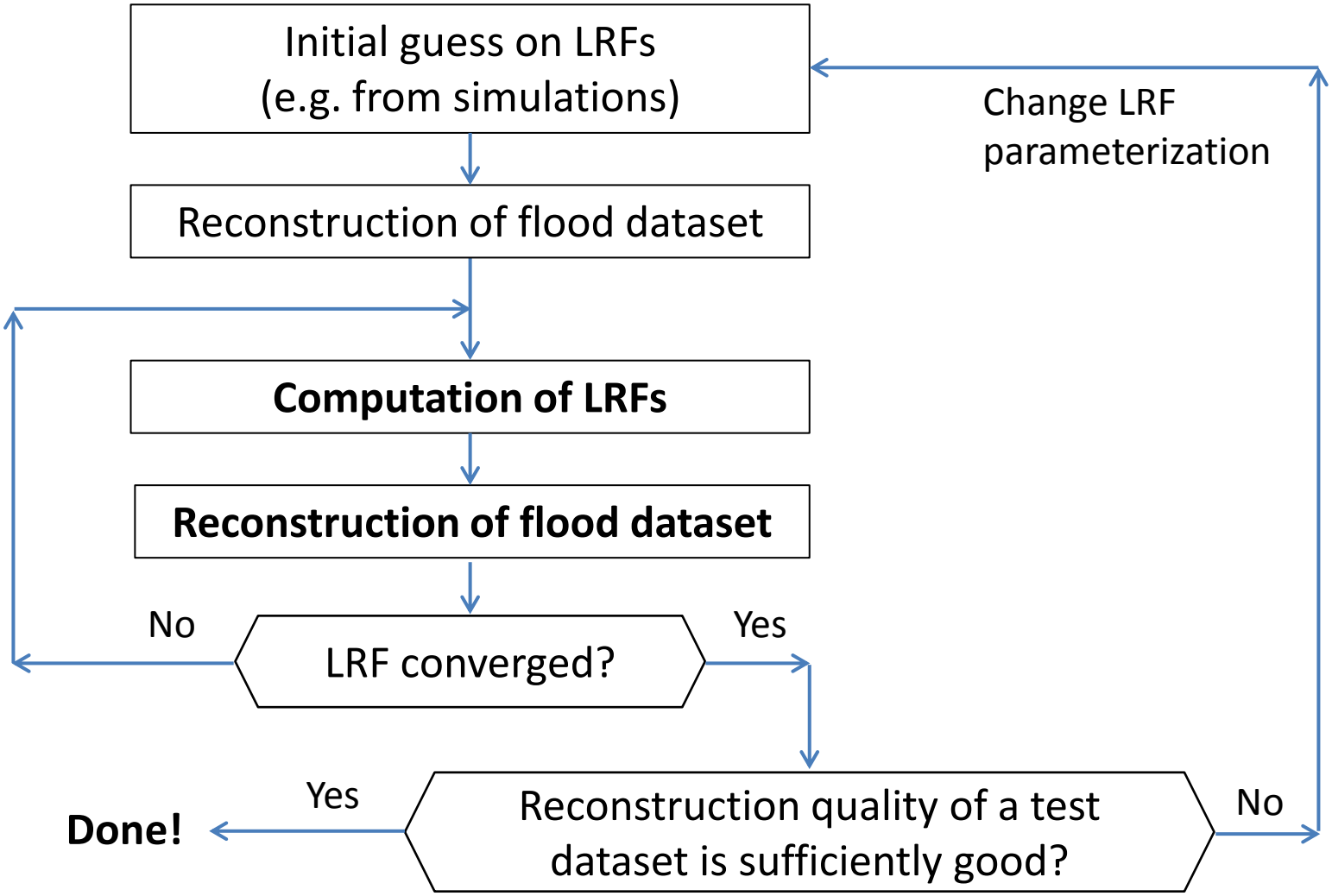
All methods discussed above require knowledge of the detector response model, which can be obtained from:

- **Simulations**
  It is usually difficult to make accurate enough predictions!
- From **calibration** datasets with **known source positions**
  Can be very impractical / impossible to provide!

Know-how of LIP:

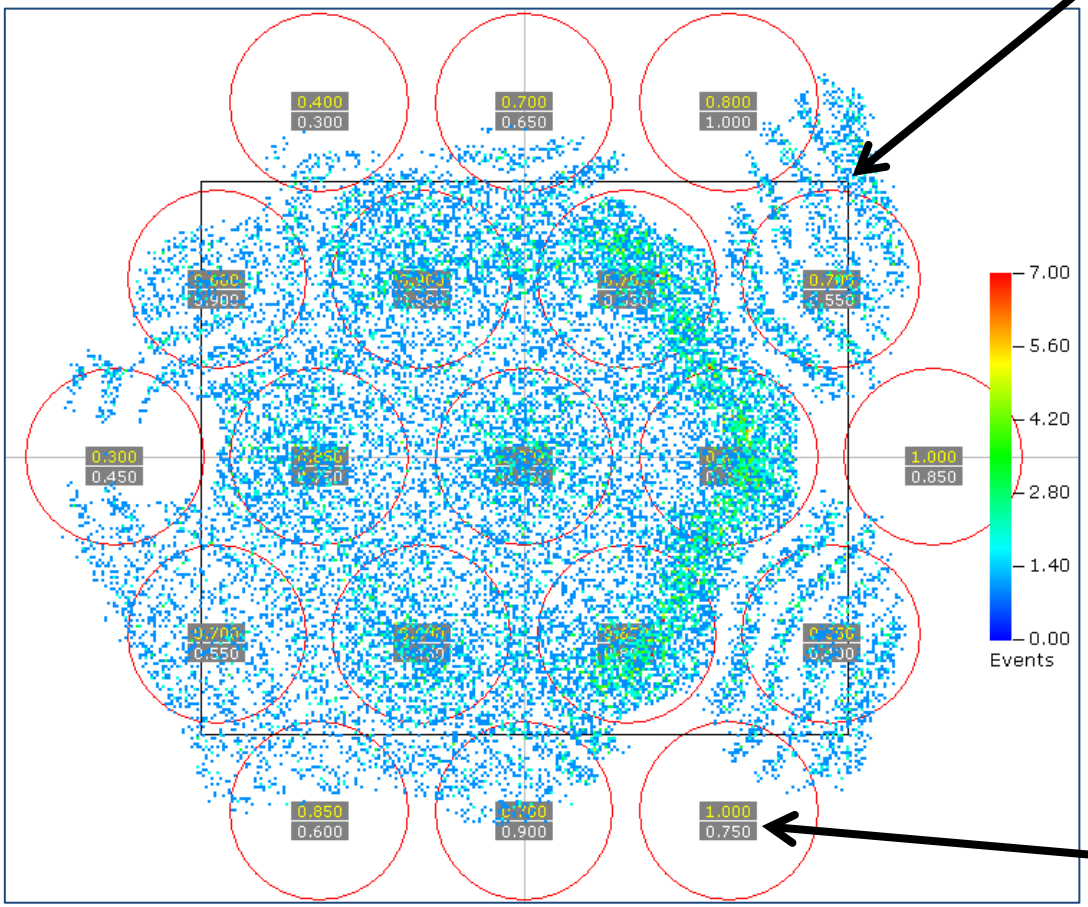
- Iterative (adaptive) response reconstruction:
  Is it possible to soften the requirements and use **only flood field calibration** datasets.
  No information on the true event positions is needed

Iterative response reconstruction

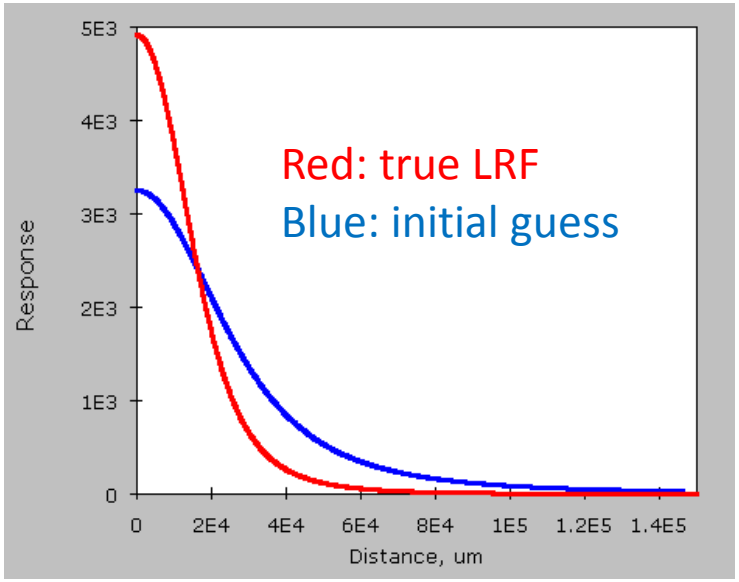


Iterative method

Initial guess on LRF



Flood field: events uniformly cover the outlined area (black rectangle)
Energy spread: 20% FWHM

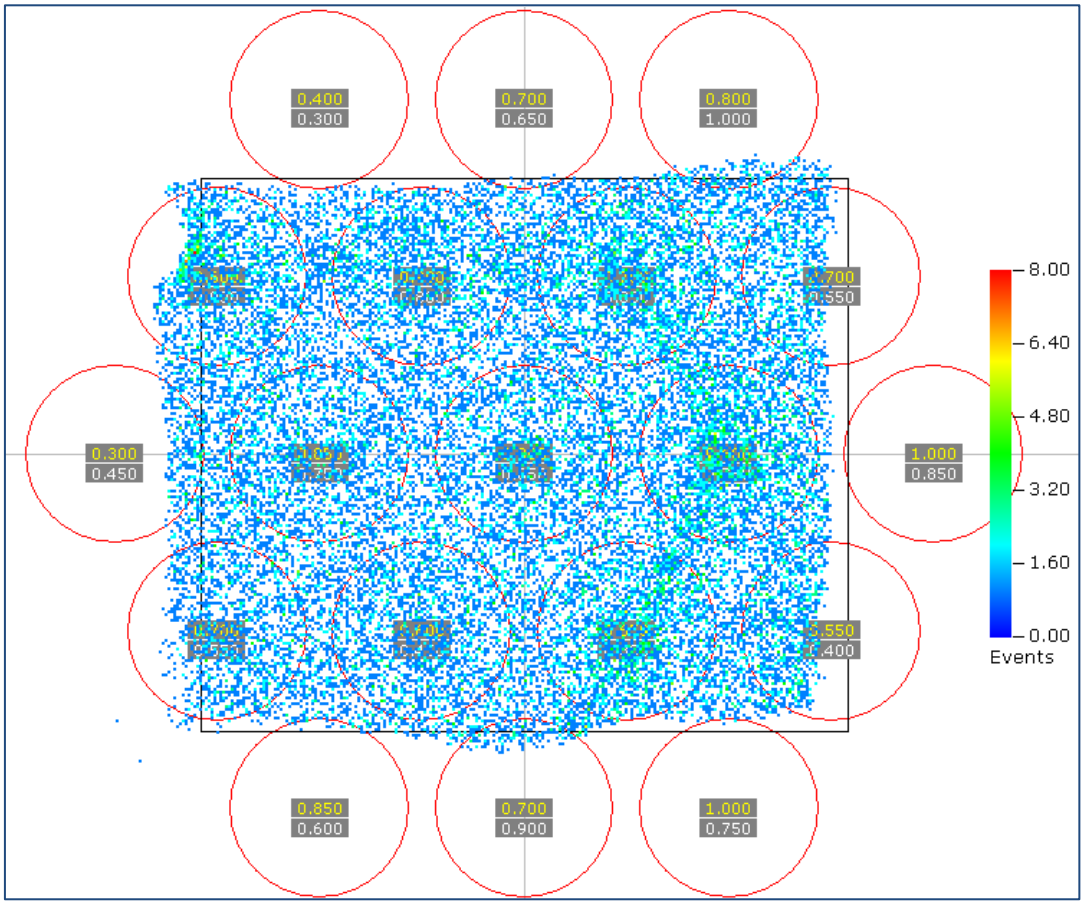


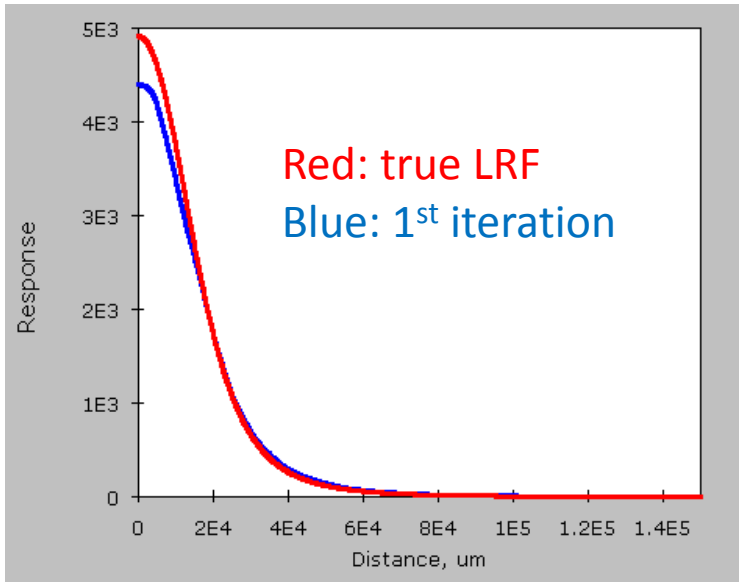
Initial gains: <u>random</u>, up to 30% off the true values

Color scale: density of reconstructed events

Iterative method

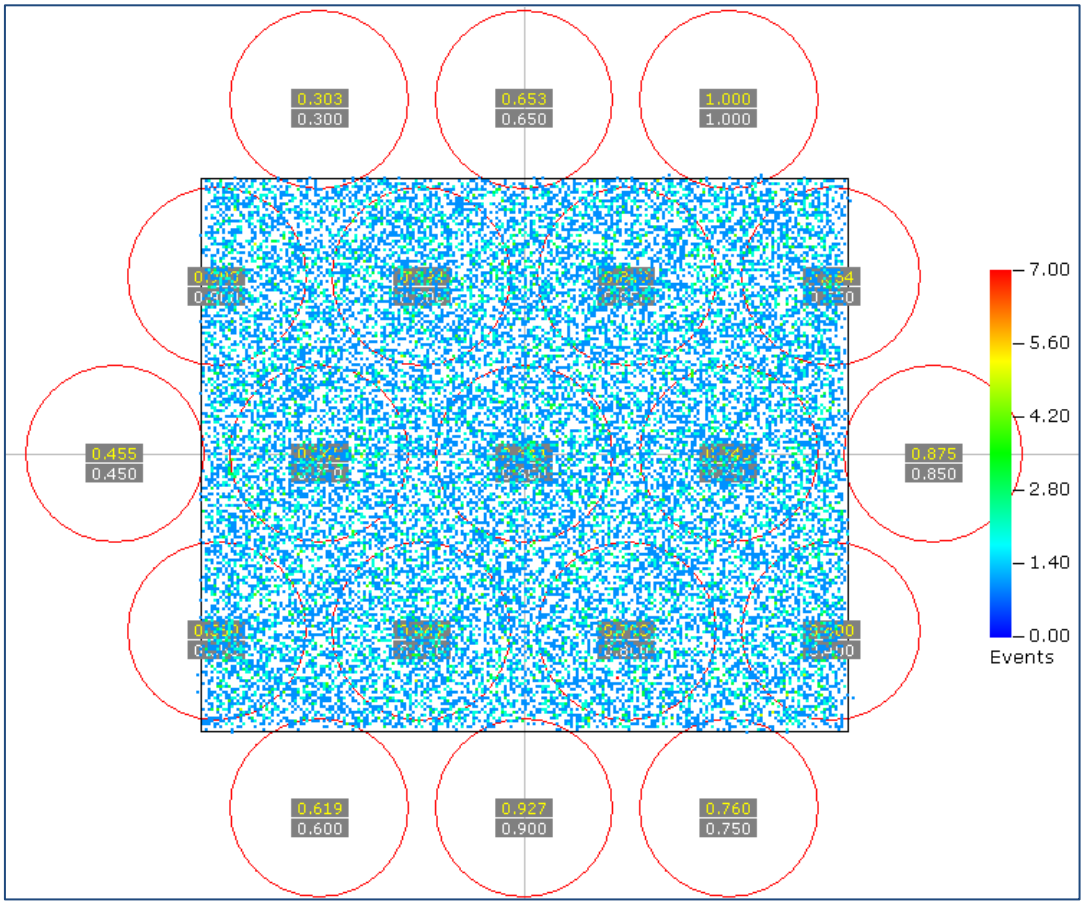
After one iteration



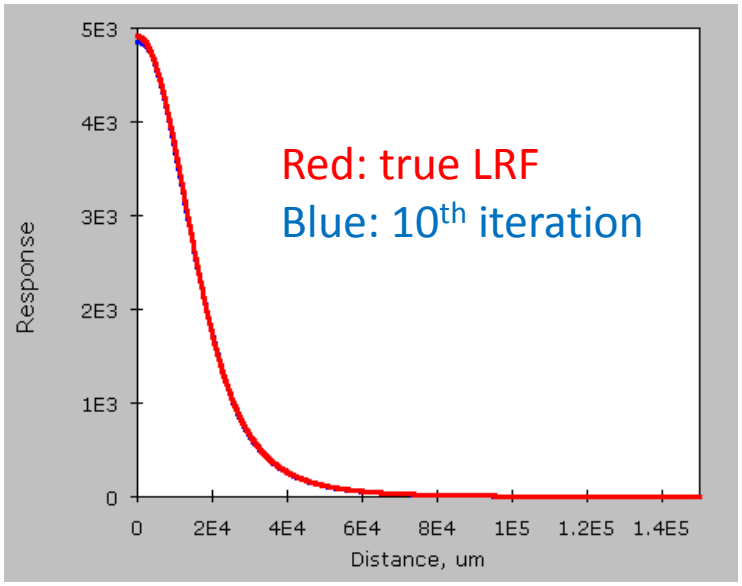


Iterative method

After 10 iterations



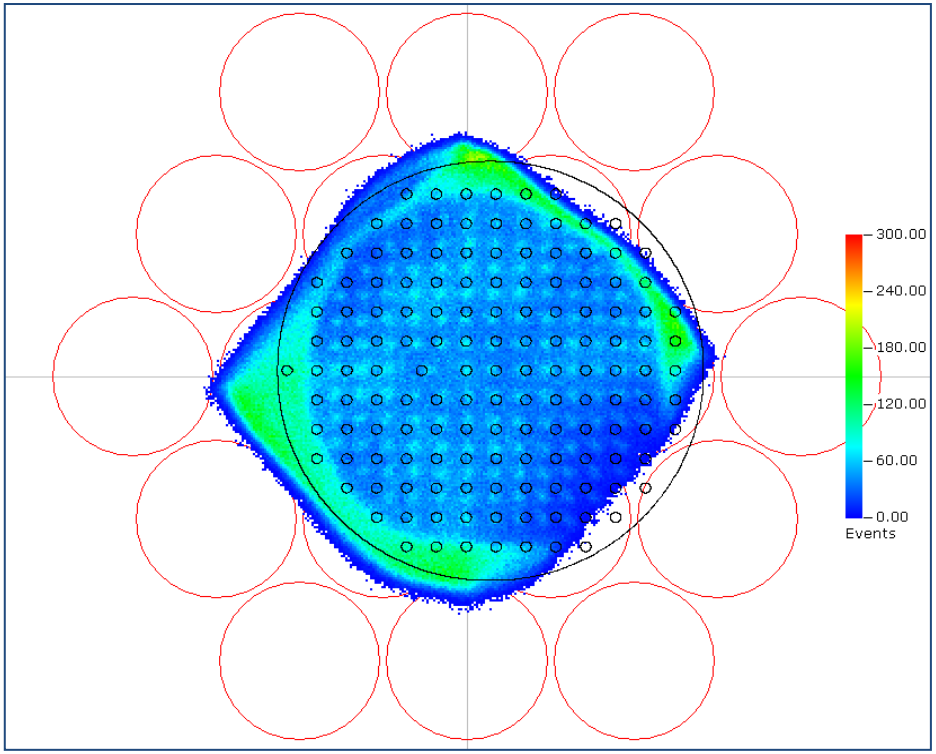
Nearly perfect match!



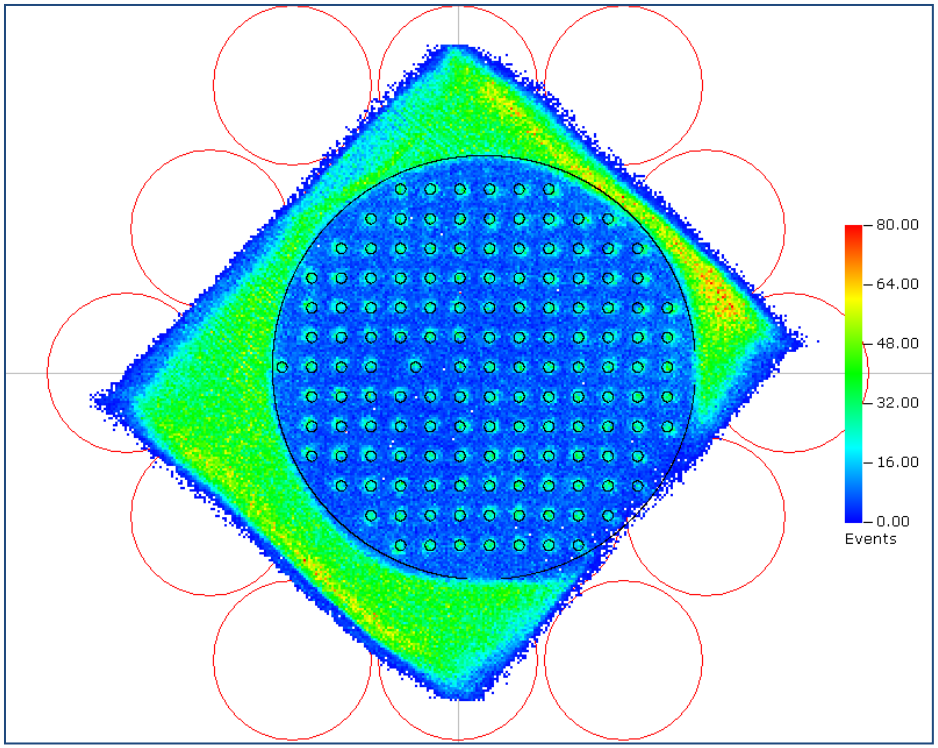
Reconstructed gains are within 5% from the true values.

GSPC-19 neutron detector

Good match between the mask contour and the reconstructed image for **experimental data:**



CoG reconstruction

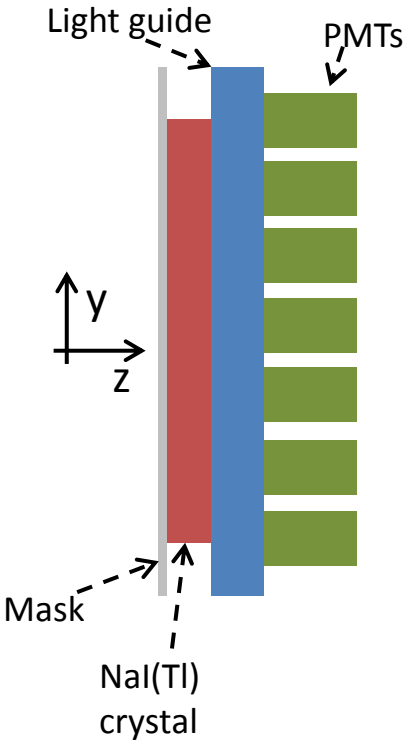


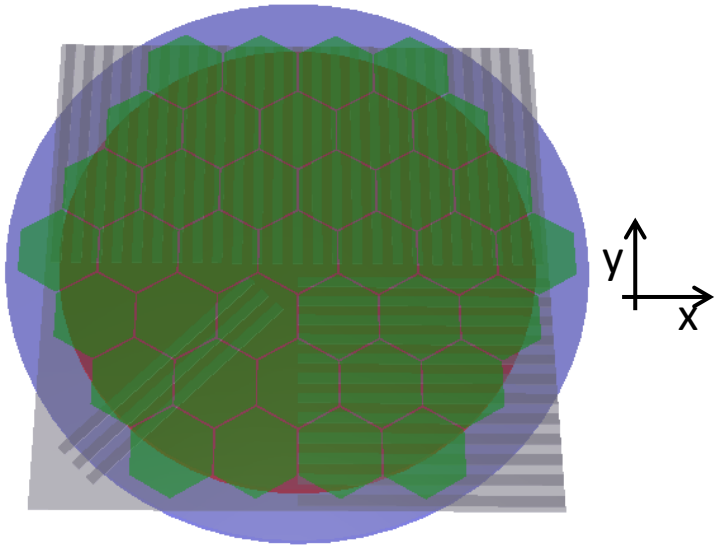
Reconstruction using LRFs obtained with the **iterative method**

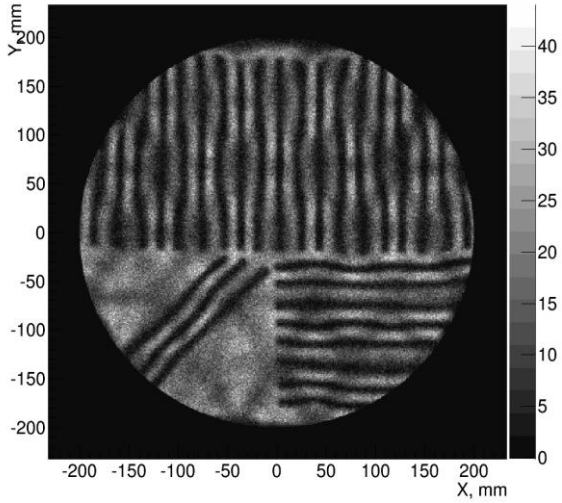
2012 IEEE NSS/MIC N21-6: A 2D gas scintillation detector for thermal neutrons

Clinical gamma camera

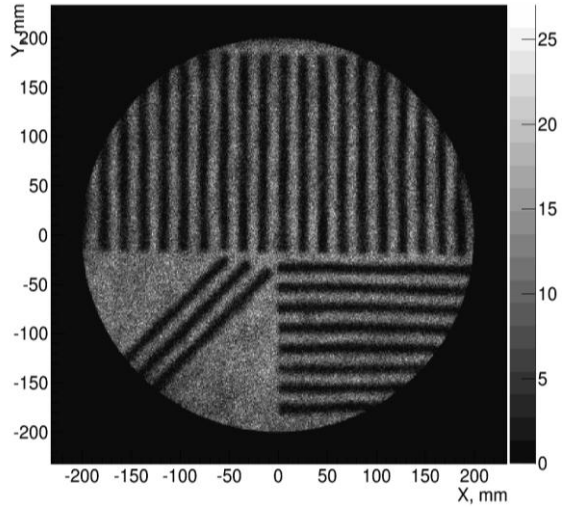
Reconstruction using
the initial guess on the LRFs



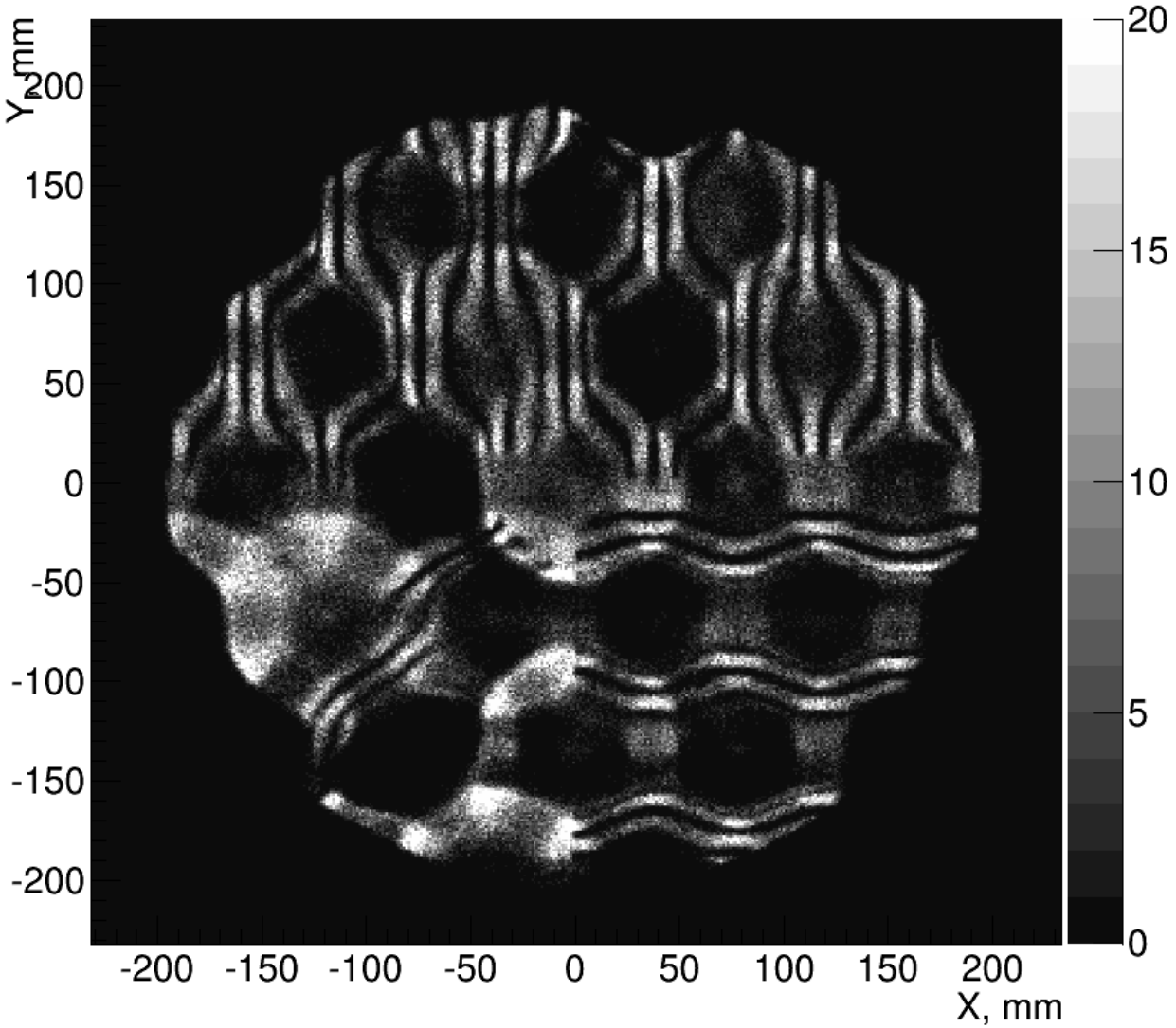




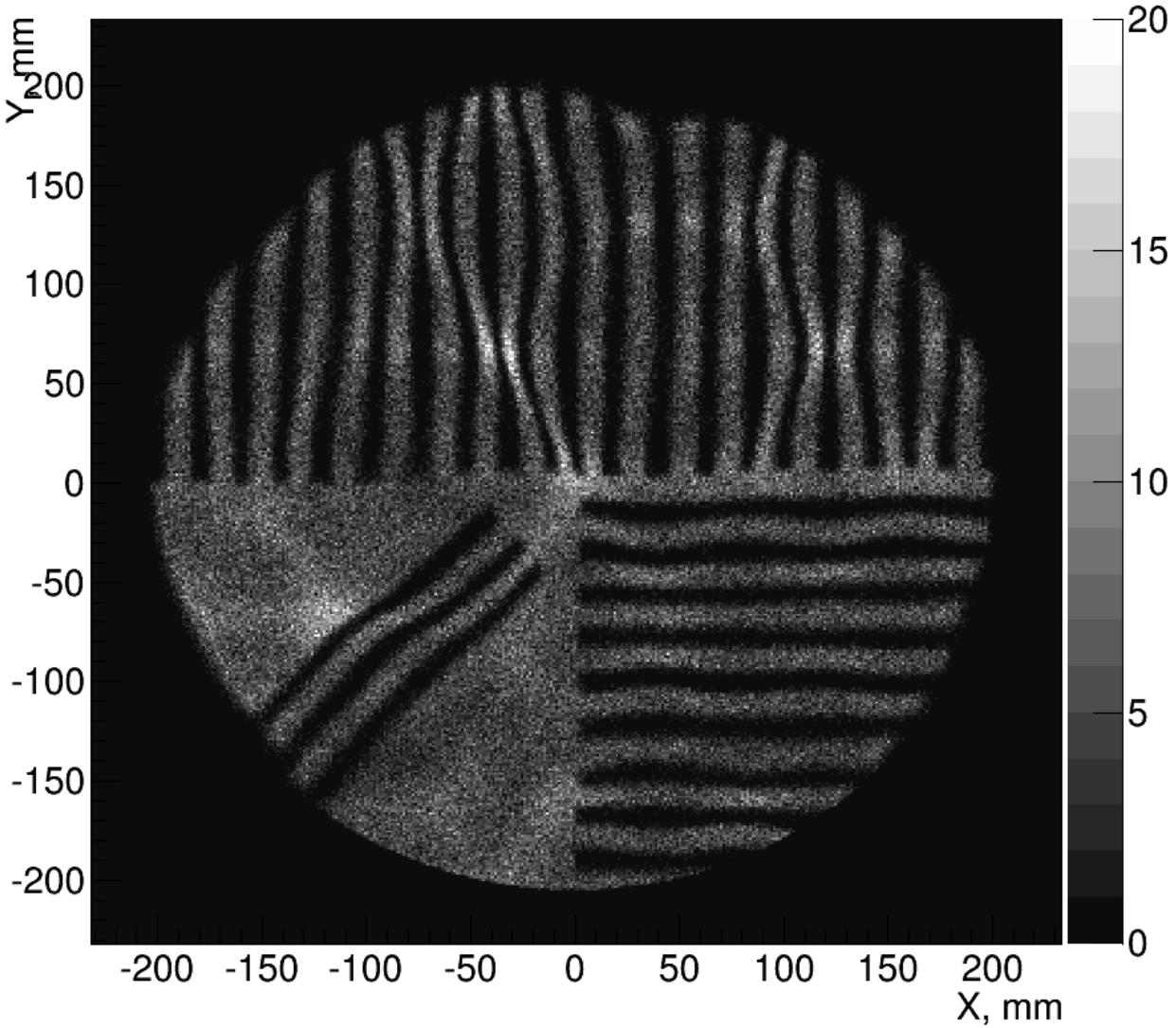
... and after 30 iterations.



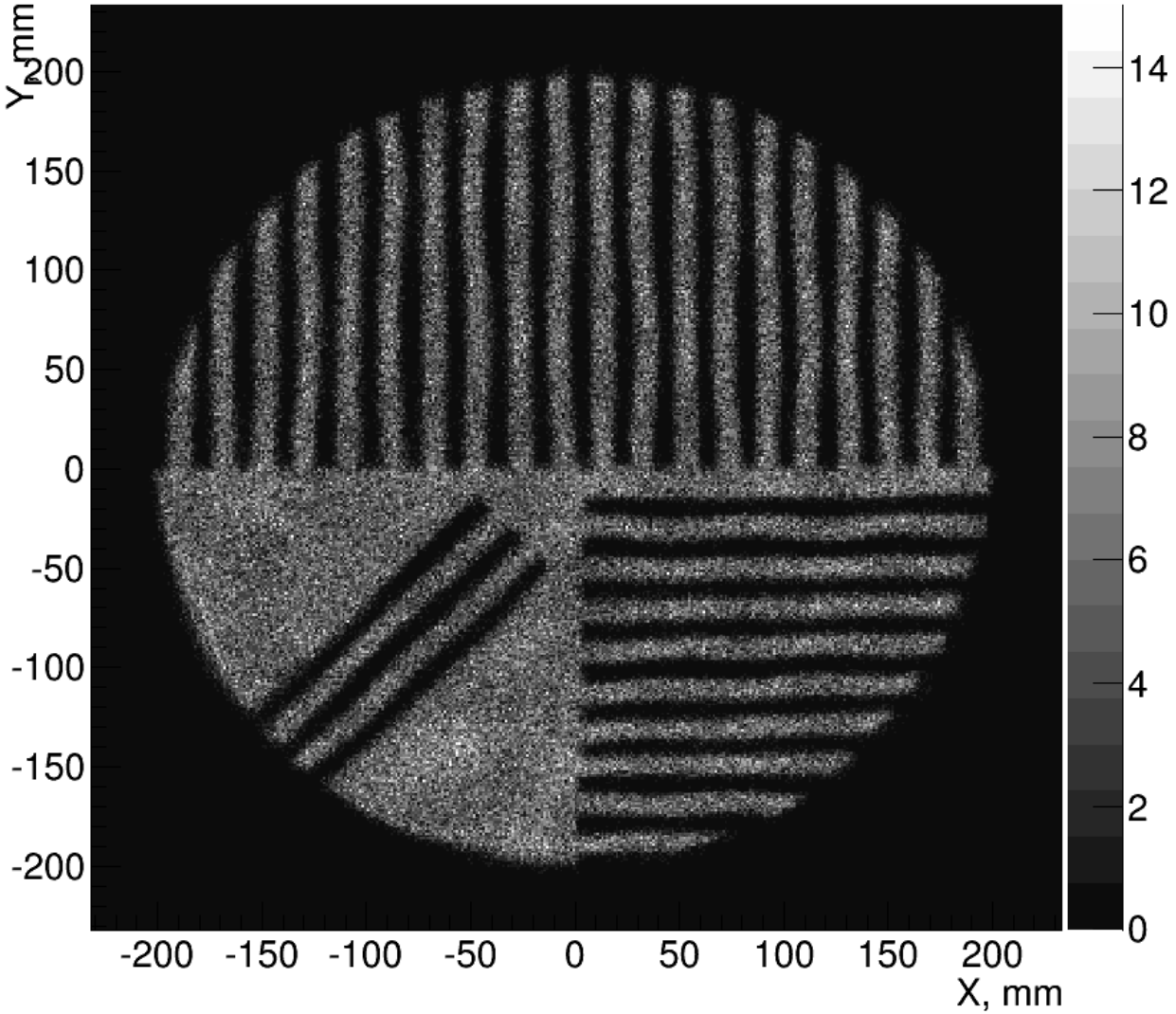
Phys. Med. Biol. 60 (2015) 4169



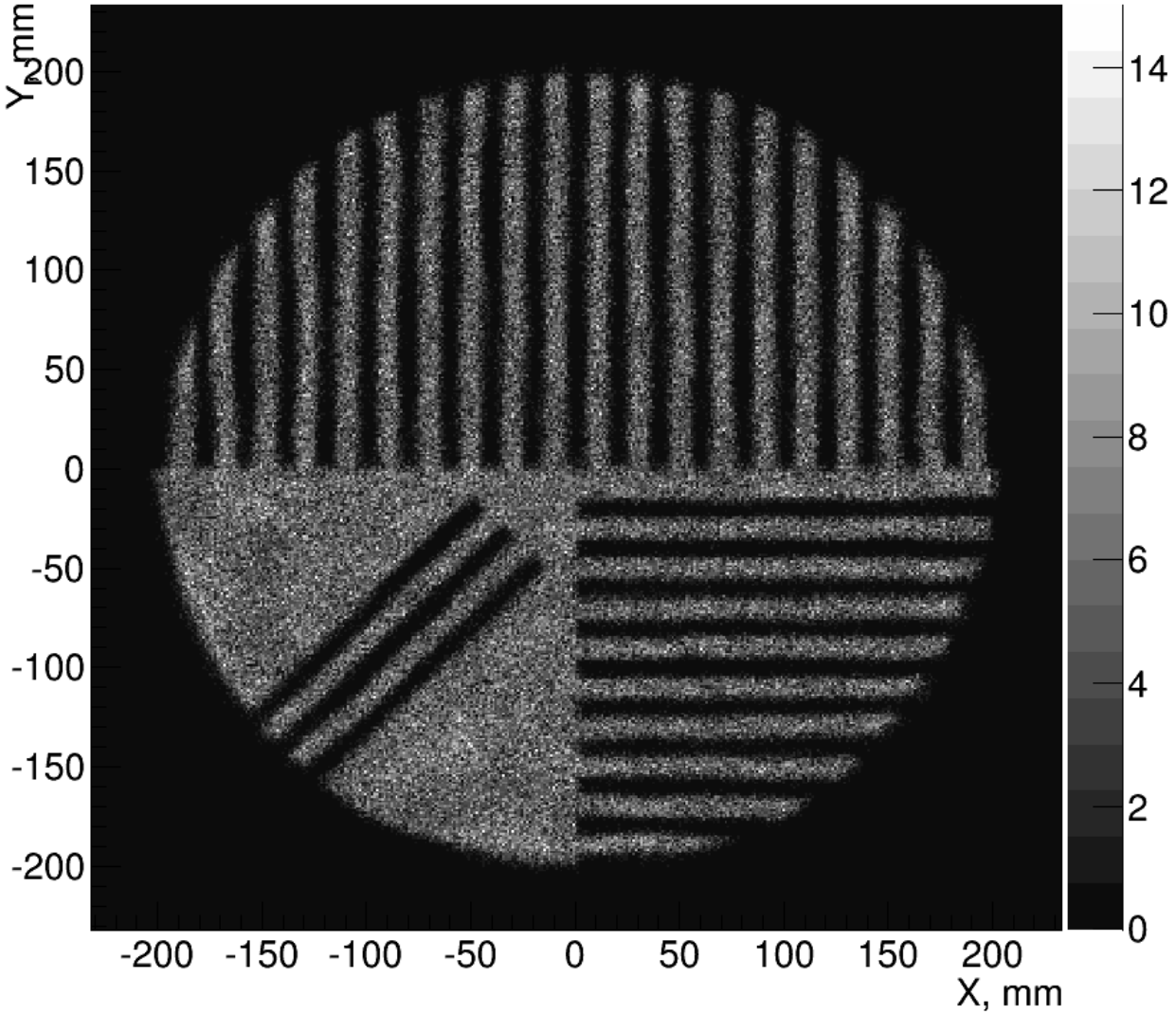
Simulation, iteration 0



Simulation, iteration 1

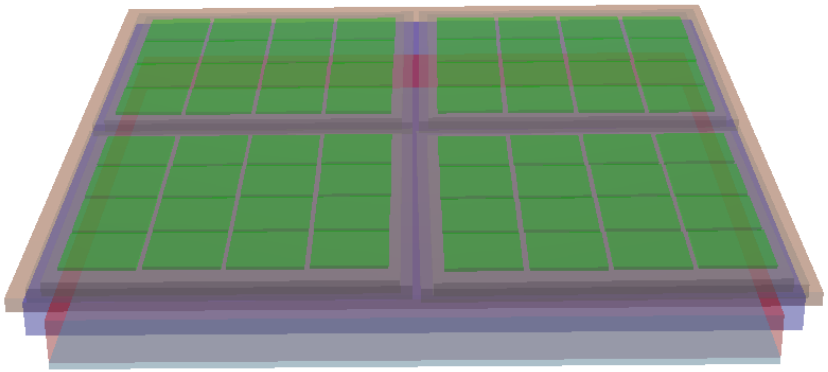


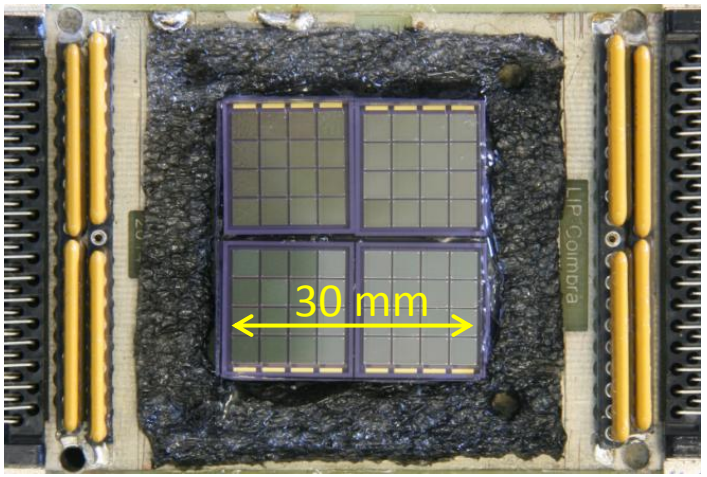
Simulation, iteration 12

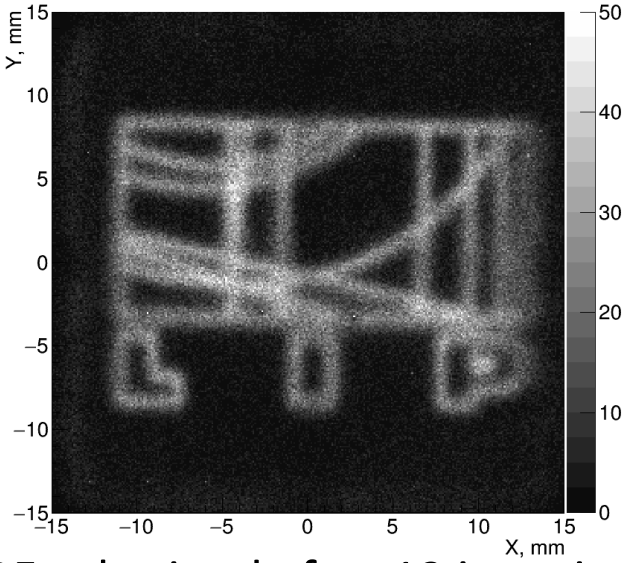


Simulation, iteration 40

Compact gamma camera







Reconstruction using the LRFs obtained after 12 iterations

Phys. Med. Biol. 62 (2017) 3619

Simulation module

Simulation module

What do we expect from the simulation module?

- Fully custom (but easy to define) 3D detector geometry
- Fresnel / Snell's + custom properties of light scattering on material interfaces
- Simulate primary/secondary scintillation and PMT/SiPM signal generation
- Fast photon tracing
- Optional feature: simulate propagation and interactions of gamma rays, neutrons and positive ions with detector materials

Decided against using Geant4 and have chosen to build a custom simulator based on **TGeoManager** 3D navigator from **CERN ROOT**

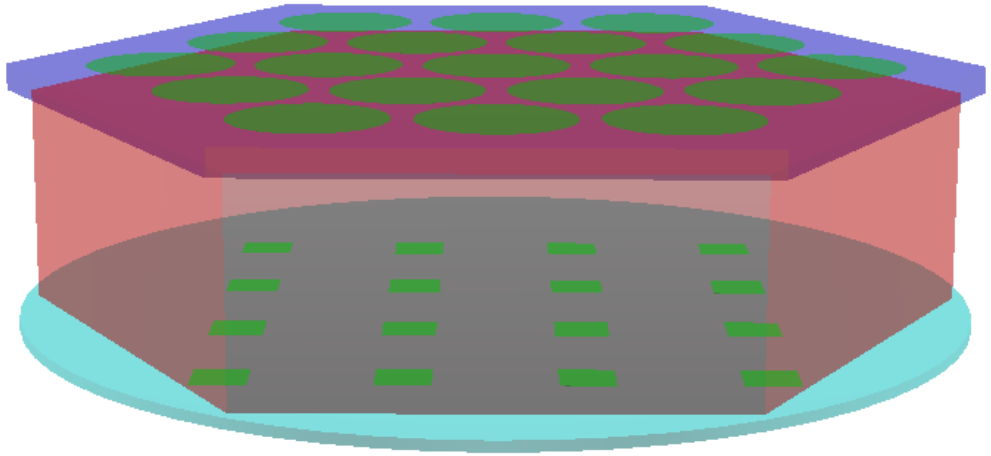
Attempt to follow the "controlled complexity" approach

Detector geometry

ANTS2 offers three methods to define detector geometry:

1. Detector as a stack of *slabs*

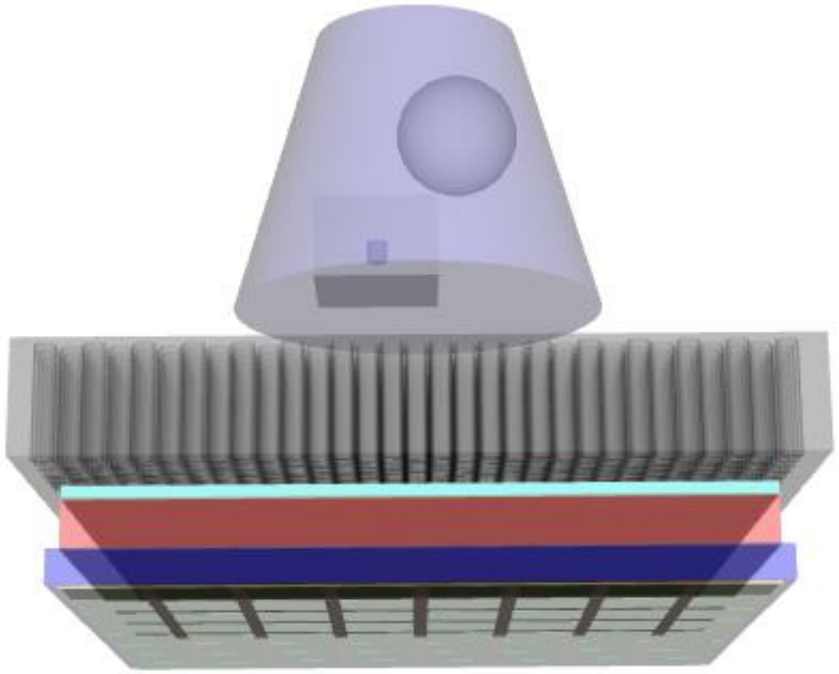
Detector is considered to be a stack of *slabs* “sandwiched” between two arrays of sensors. Each *slab* can be a cylinder or a polygon of arbitrary number of edges.



Very quick to configure – well suited for detector prototyping phase!

Detector geometry

2. Custom TGeoShape-based objects in a hierarchical tree
   - World object is the root of the tree
   - A new object can be placed inside any other at an arbitrary position and with an arbitrary orientation.
   - Extremely flexible: Tens of shapes supported by the TGeoManager, including composite shapes (union, subtraction and intersection)



Custom tools are provided for handling of:

- Arrays of objects
- Stacks of objects
- Compound lightguides
- Grids/meshes of wires

Geometry can be build staring from *slabs* defined with the first method.

Detector geometry

3. Import from a GDML file

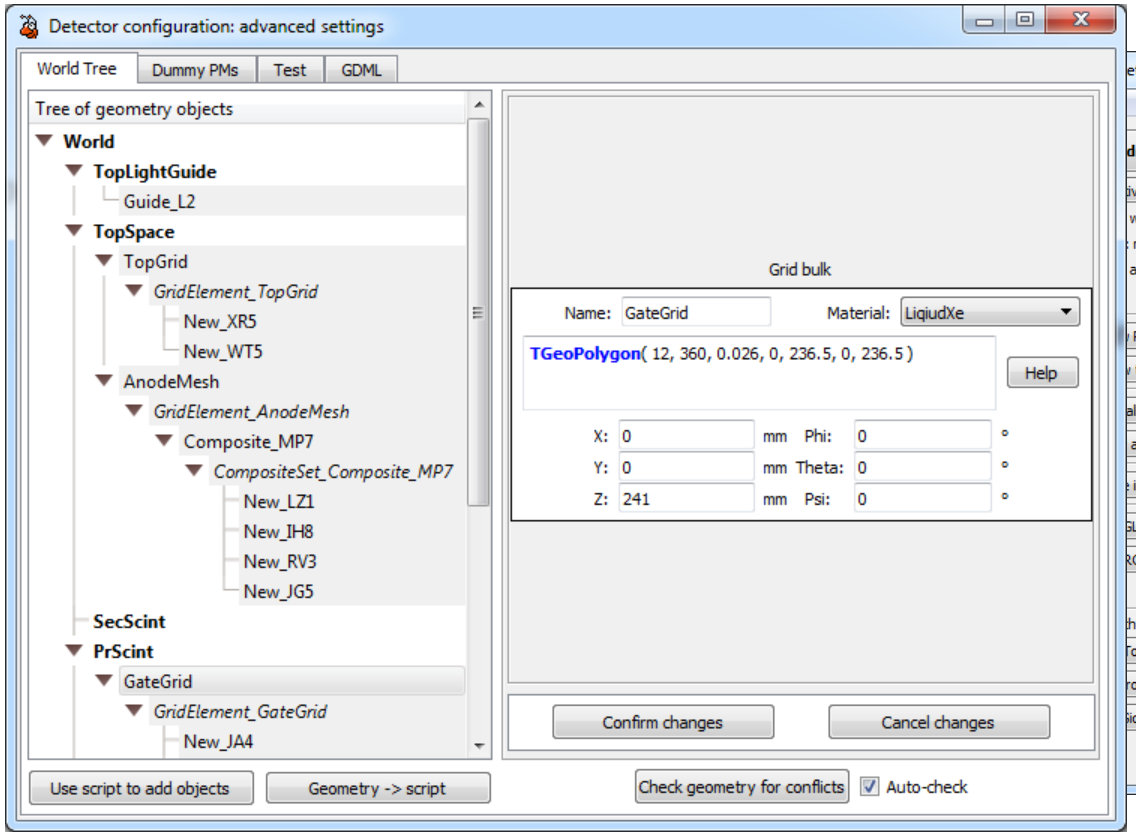
   Geometry defined in a GDML file (XML-based format) can be directly imported to ANTS2 or converted to ANTS2 system.

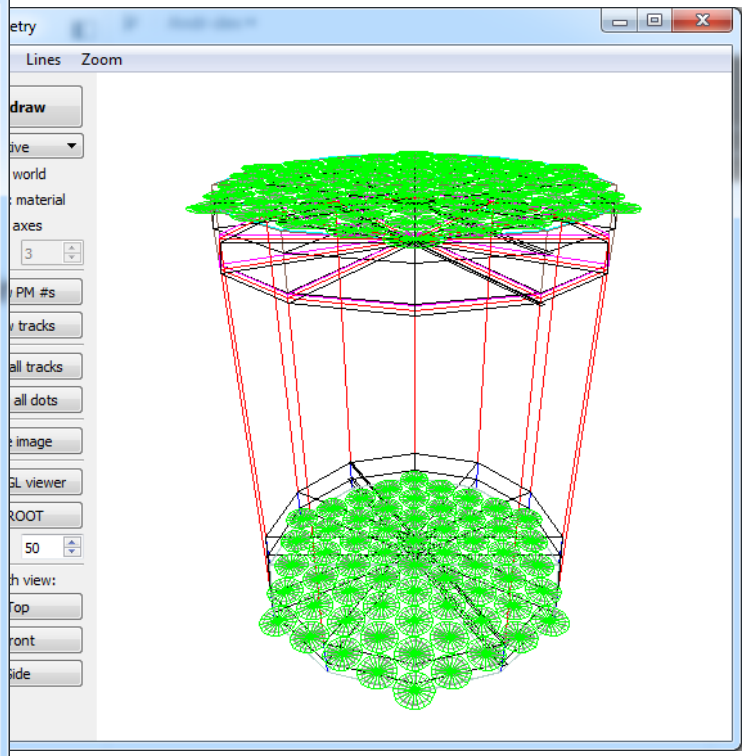
   GDML standard is supported by both Geant4 and ROOT TGeo.

Detector geometry

Geometry can be defined or modified:

- in an interactive GUI
- using scripting system (JavaScript or Python)
- by editing the configuration files (JSON format: attribute + value pairs)





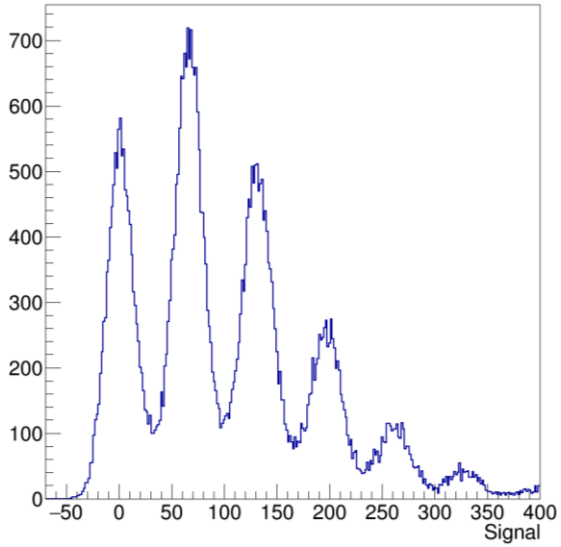
Optical sensors

Photomultiplier tubes (**PMT**) and silicon photomultipliers (**SiPM**):

The general properties of a sensor *model* are shared by all individual sensors of that type: the **shape**, **size** and the **material** of the optical interface.

If required, the following parameters can also be configured:

- Photon detection efficiency
  (optionally *vs* wavelength, angle and position)
- For SiPMs: the number of microcells,
  dark count and optical crosstalk
- Signal readout properties
  - single photoelectron spectrum
  - electronic noise
  - Analog-to-digital conversion properties



ANTS2 simulation: distribution of signals from a SiPM sensor

The parameter values defined for a sensor *model* can be overridden for each individual sensor.

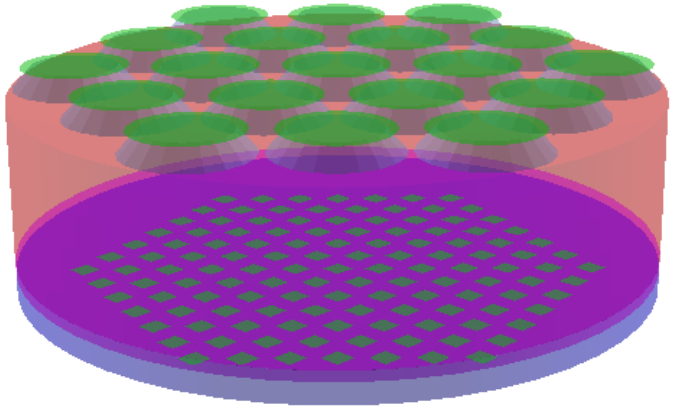
Arrays of sensors

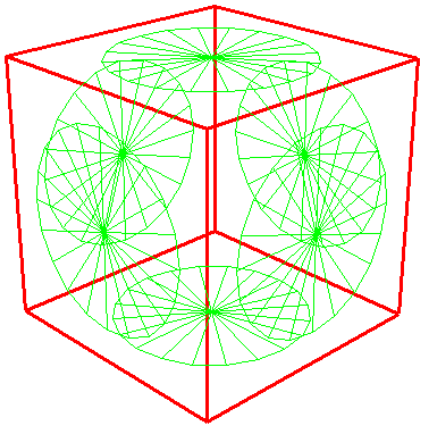
Regular mode:

- All sensors are automatically positioned in square or hexagonal packing in the XY plane parallel to the *slabs*
- The Z position is automatically updated on all changes in the *slabs*

Custom mode:

- The XYZ position, orientation and sensor *model* are configured individually.





Simulation modes

ANTS2 offers two main simulation modes:

Optical photons only

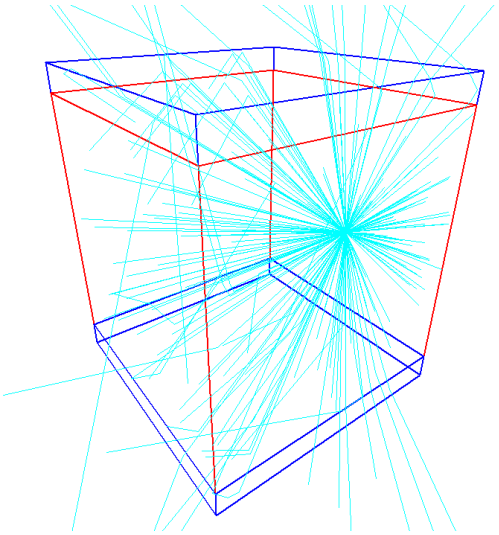
- To be used when it is sufficient to simulate each event as a "*photon bomb*" – isotropic emission of a photon from a given source
- Allows to assess the "ultimate" detector performance under ideal conditions
- Well suited during detector prototyping phase

Particles + optical photons

- Used to simulate interaction of particles with the detector and generate optical photons in primary/secondary scintillation.
- These simulations produce more accurate results but require additional configuration effort.

Photon sources

Individual events are simulated starting from generation of photons from a point source (or along a line for secondary scintillation)



The number of photons per event is constant or sampled from a user-defined distribution.

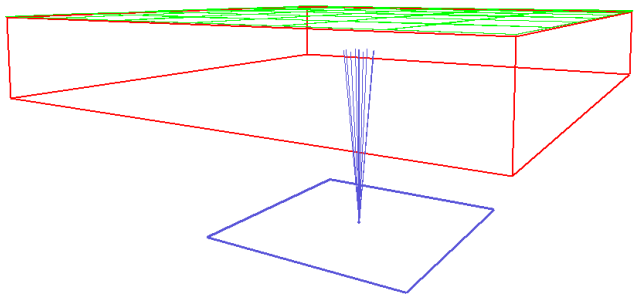
GUI offers a simple way to generate an event dataset, with the source positions being:

- a fixed position
- a regular array of positions (scan)
- random positions in an area (flood)

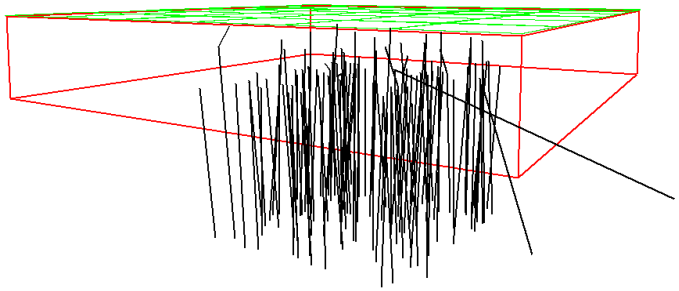
Particle sources

Individual events are simulated starting from emission of particles from a *particle source(s)*.

- A source emits particles with the initial energy which is fixed or sampled from a given distribution.



- A source shape can be: a point, line, square, round, box or cylinder. Emission is uniform over the shape.

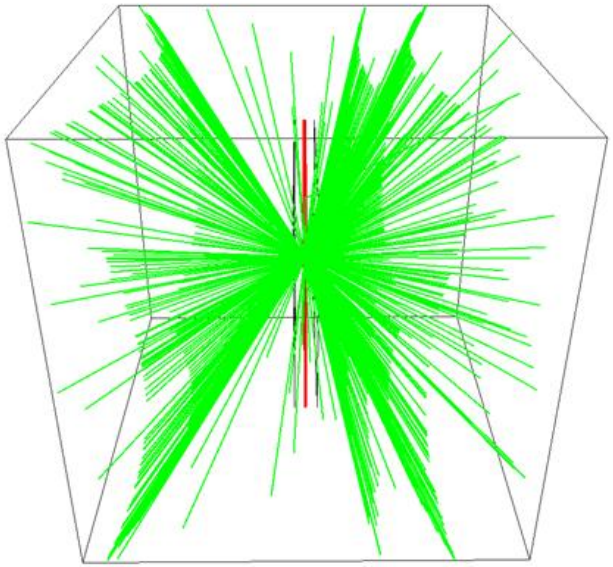


- Emission from a source can be collimated: the particles are generated with initial direction in the user-defined cone.

Alternative mode: particle generation from a file with particle records or a script-based particle generator

Interaction processes

ANTS2 can simulate:



- **Gamma rays**:
  - Photoelectric absorption
  - Compton scattering
  - Pair production
- **Neutrons**:
  - Absorption
    - Capture + emission of fission fragment
  - Elastic scattering
    - Coherent scattering on polycrystalline materials using NCrystal library
- **Positive ions**:
  - Continuous energy deposition

All particles are considered to move in straight lines between the interactions.

Materials

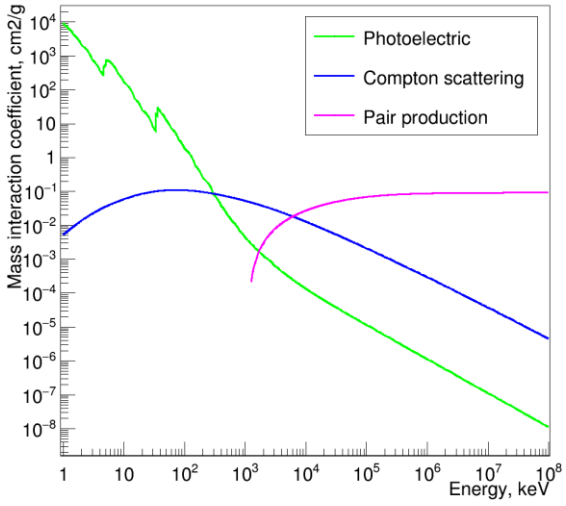
Every volume of the detector geometry have to be associated with one of the user-defined materials.

Materials have three groups of user-defined properties:

- Optical
- Primary / secondary scintillation
- Particle interaction

ANTS2 have a set of tools to import data from

- XCOM database (gamma rays)
- TRIM/SRIM output (positive ions)
- IAEA database aggregator (neutrons)

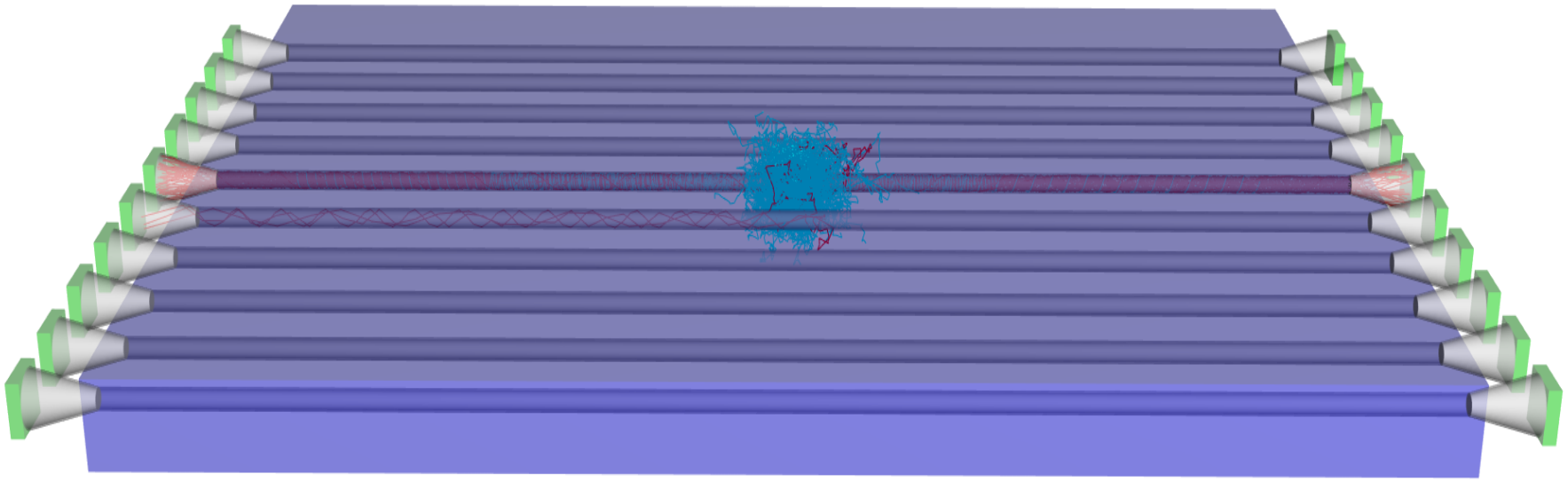


Photon tracking

Photon tracing includes the following processes:

- Reflection and transmission (Fresnel equations, Snell's law)
- Absorption (+ possible reemission with wavelength shifting)
- Rayleigh scattering
- Custom rules for optical interfaces (next slide)

3D navigation and calculation of the normal to the surfaces (volume boundaries) are performed using TGeo library of ROOT.

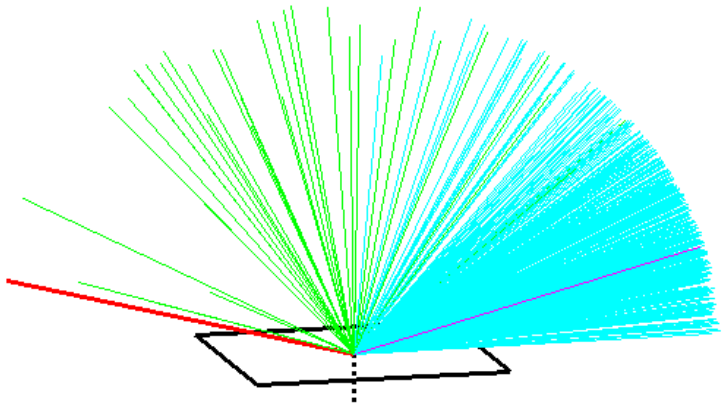


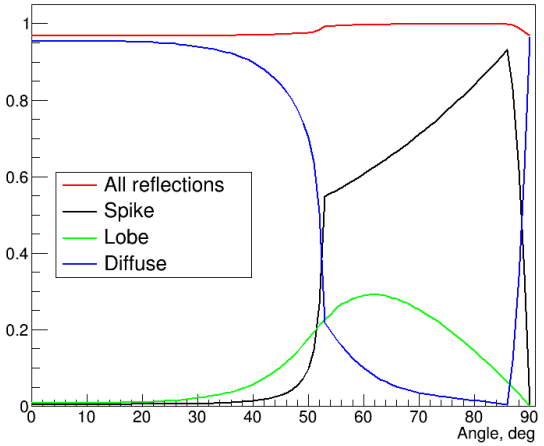
Photon "tracks" in the neutron detector with wavelength shifting fibers

Custom rules for optical interfaces

In ANTS2 it is possible to define custom rules for behavior of optical photons at the borders of geometrical volumes.

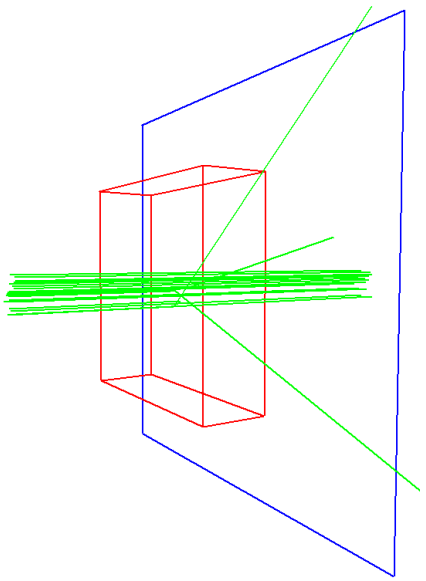
- The rules are defined for a pair of materials (*from* and *to*).
- In the simplest case, the rule includes the probabilities of absorption, specular reflection and Lambertian scattering.
- More advanced models describe light scattering on rough surfaces (e.g. the model of Cláudio Silva), dielectric-metal interface and wavelength shifters.
- A custom model can also be defined with a script.

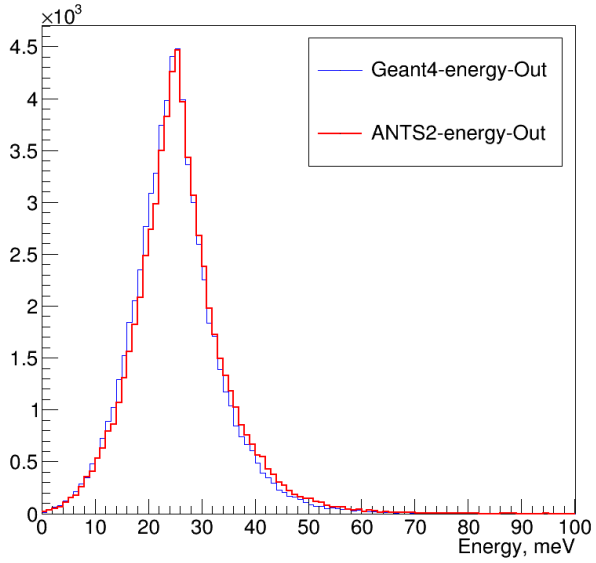


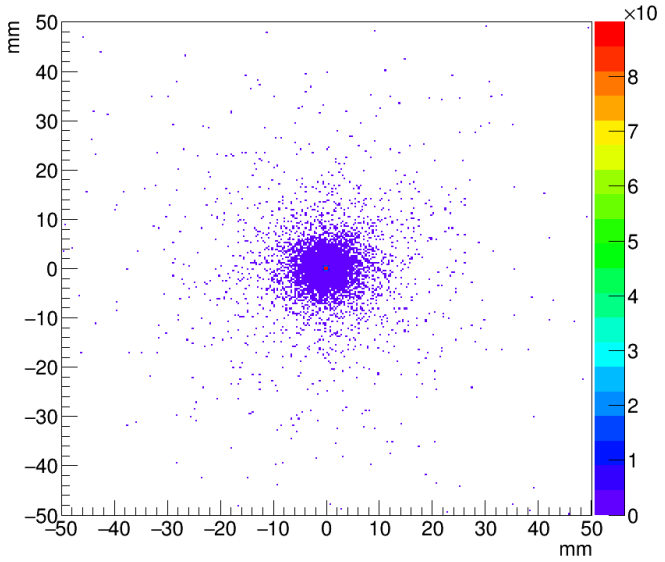


Simulation tools

- *Monitor* objects:
  Record statistics of the passing optical photons or particles.
  Gives access to the distributions of
  - time
  - position
  - angle of incidence,
  - wavelength (energy)

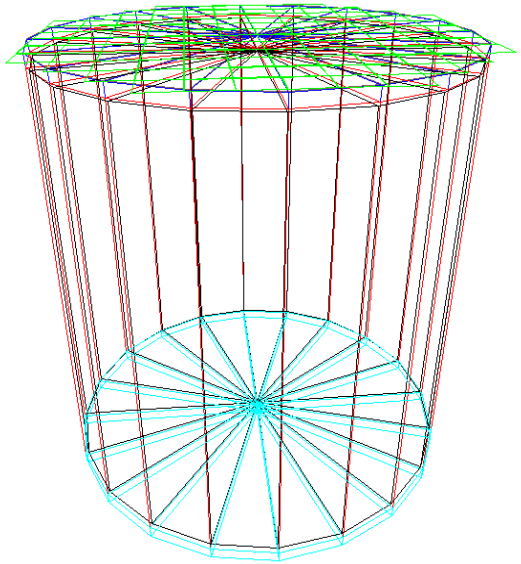


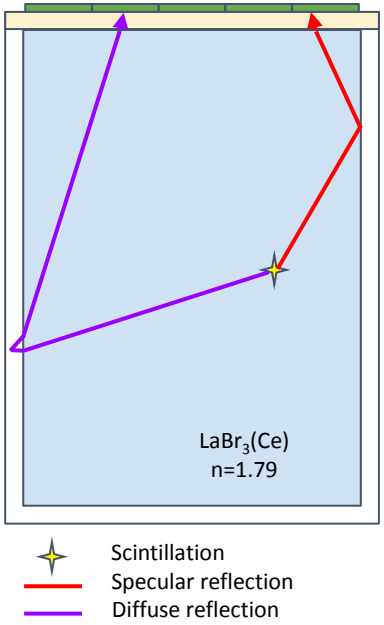




Simulation tools

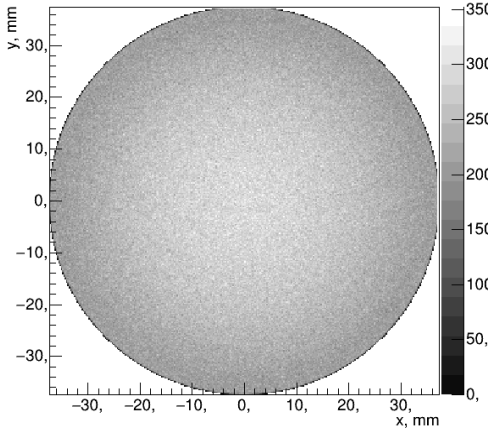
Monitors helped to show the feasibility of 3D position reconstruction in detectors with $LaBr_3$ cylindrical scintillators and one plane of SiPM sensors.



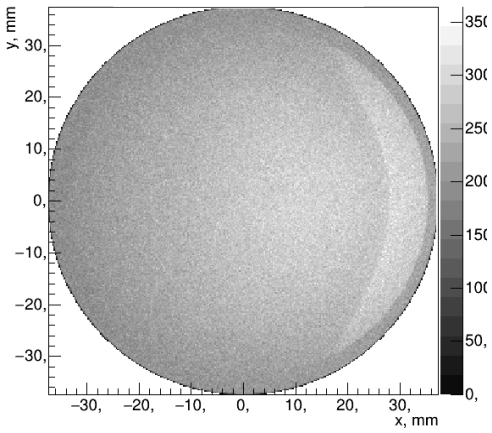


This is how the light pattern on the output window depends on the position of the light source inside the crystal:

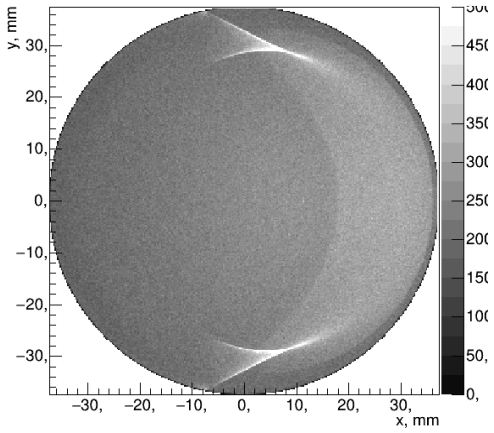
On axis



Half radius



Close to the border



Simulation tools

- “Photon” script unit:

  Detailed history of photon tracing can be collected and accessed for individual photons selected according to two configurable lists of processes: “must have” and “should not have”.
- “Depo” script unit:

  A detailed history of tracking, energy deposition and termination can be accessed for each particle.

Validations

- Comparison with analytical models
- Cross-comparison with Geant4
  - Measurements of PTFE reflectivity by F. Neves
  - Neutron absorption/scattering for materials of ^{10}B-RPCs detectors (cooperation with ESS)
- Comparison with experimental data
  - Gamma cameras: R. Martins (Master thesis)
  - Evaluations on the workbench of L. Pereira (PhD thesis)
  - Measurements of ^{10}B-RPC detection efficiency by L. Margato
  - Gamma camera + collimator: J. Marcos (ongoing PhD)
  - $LaBr_3$ 3D detectors (ongoing cooperation with Milan Polytechnic)

Semi-automatic detector optimization

Semi-automatic detector optimization

A multi-parameter optimization is often required during detector development.

The brute-force approach:

- Define a grid of parameter values
- Perform simulation + analysis for each grid node
- Find the best node.

Can be very time consuming!

ANTS2 offers a more efficient alternative:

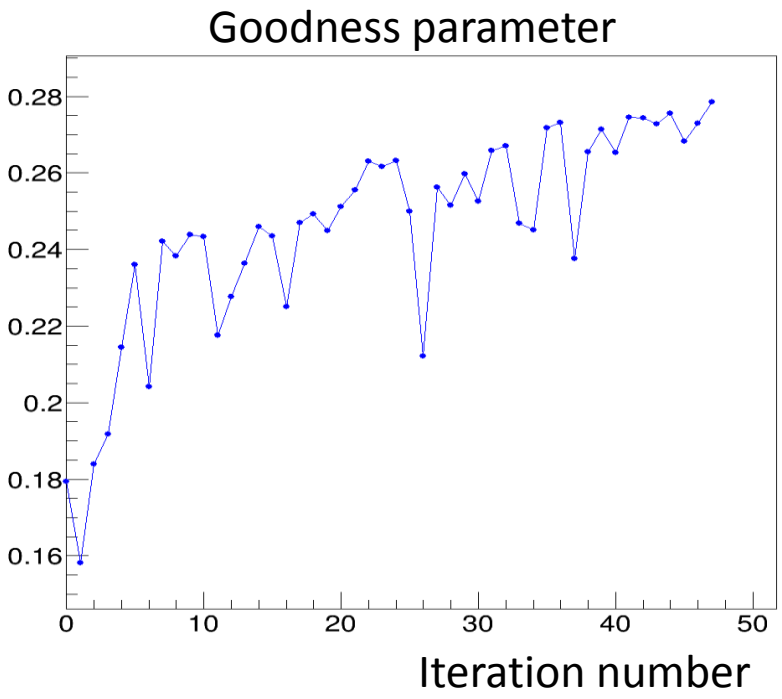
In script, define a minimization function. This function will be called by the minimizer, running, e.g. Simplex algorithm.
On each call from the minimizer the function:

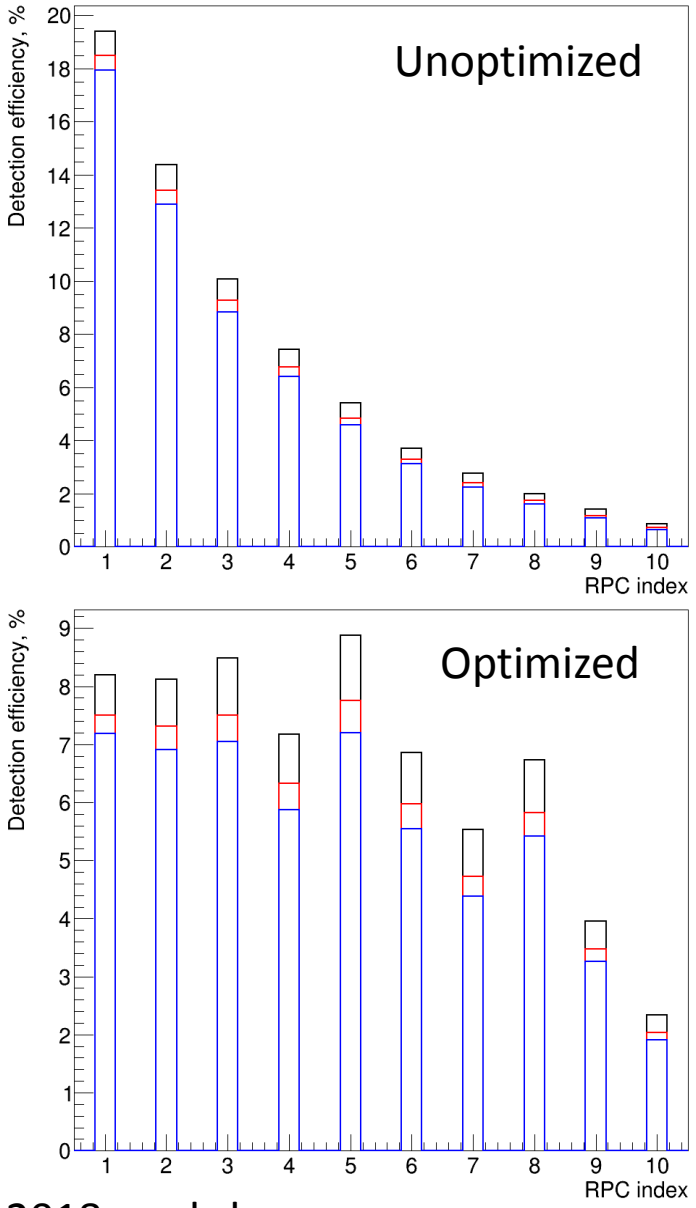
- modifies the detector geometry
- runs a simulation providing a set of events
- reconstructs the events
- performs analysis and returns the goodness value to the minimizer.

Semi-automatic detector optimization

^{10}B-RPC neutron detector:

Optimize 5 neutron converter thicknesses to equalize as much as possible the count rate of 10 RPCs but keep the total detection efficiency as high as possible.





Presented at PSND 2018 workshop

Implementation details

Implementation

ANTS2 is written in C++ using Qt framework
Requires CERN ROOT 5 or 6

Optional libraries:

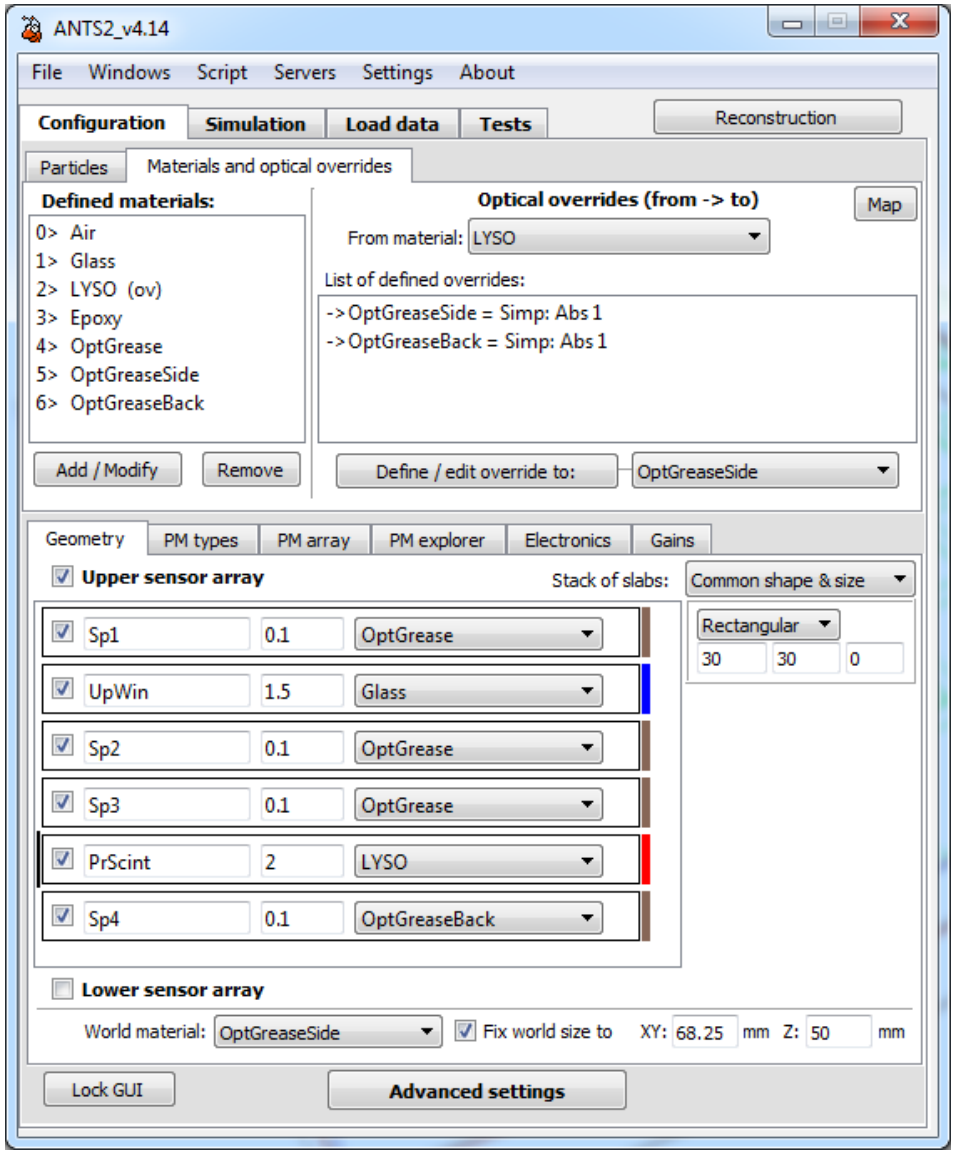
EIGEN3:	for faster LRF fitting
PythonQt + Python 2/3:	for Python scripting
CUDA toolkit:	for GPU-based statistical reconstruction
NCrystal:	for elastic neutron scattering simulation
FANN:	for ANN reconstruction
FLANN:	for kNN reconstruction and event rejection

Open source and more detailed information can be found at:

https://github.com/andrmor/ANTS2
http://coimbra.lip.pt/ants/ants2

JINST 11 (2016) P04022 JINST 11 (2016) P09014

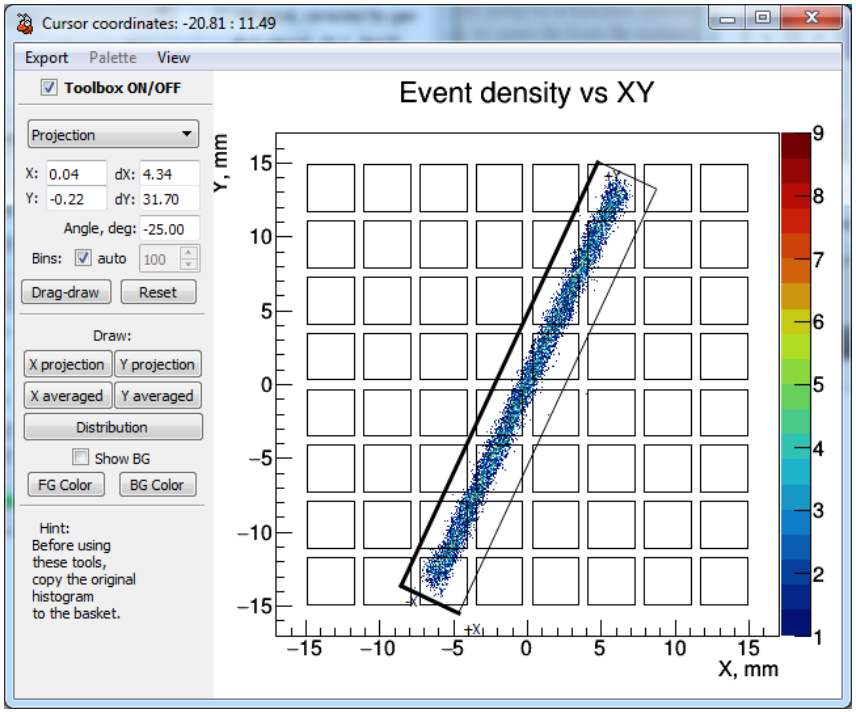
ANTS2 GUI



Main window

GUI was developed in **Qt** framework

Visualization of the detector geometry, tracks, markers, histograms and graphs is implemented using **ROOT** as a library.



Graph window

ANTS2 scripting

ANTS2 offers scripting in two programming languages:

- **JavaScript**
- **Python**

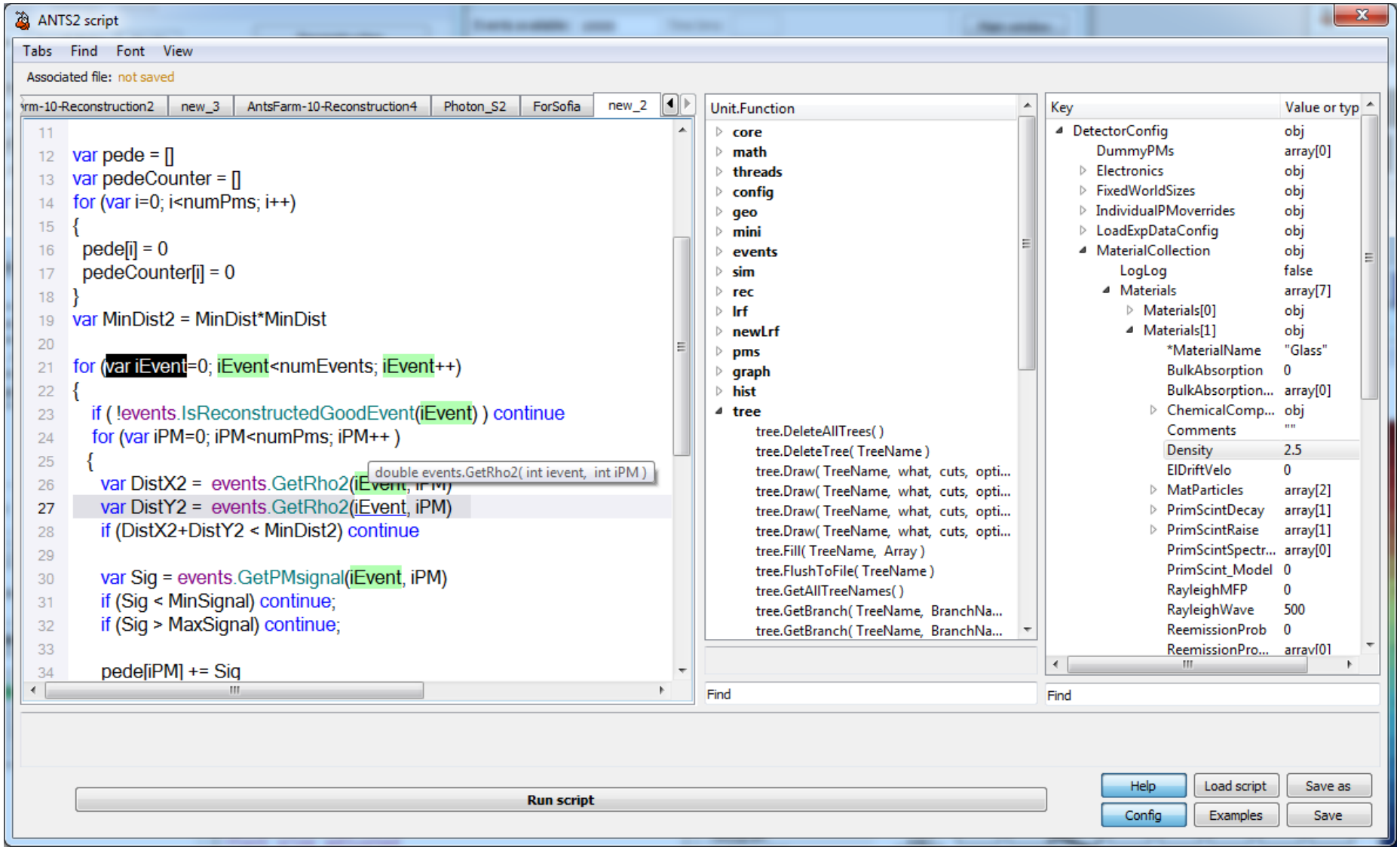
The JavaScript scripting supports multithread script evaluation.
The Python scripting uses the system Python interpreter, so all Python modules installed on the system are accessible in ANTS2 via import directive.

Scripts can be executed

- in GUI (next slide)
- in batch mode
- through the WebSocket interface of the ANTS2 server

Among many other tasks, the scripts can read and modify the detector configuration, run simulations and reconstruction, have access to the event data, can write and read ROOT trees, histograms and graphs, have access to GUI, etc.

ANTS2 scripting



Script window of ANTS2

Multiplatform ANTS2

- ANTS2 was created as a multi-platform application being based on multi-platform libraries
  - Qt: Windows, Linux, Mac
  - ROOT5: Windows, Linux, Mac
- However, life had a few surprises up its sleeve:
  - Qt+ROOT5: Windows, Linux, ~~Mac~~
    - ROOT5: only 32-bit mode on Windows => 2GB memory limit
  - Qt+ROOT6: ~~Windows~~, Linux, ~~Mac~~
- And even on Linux, each distribution requires a bit of tweaking to get ANTS2 compiled

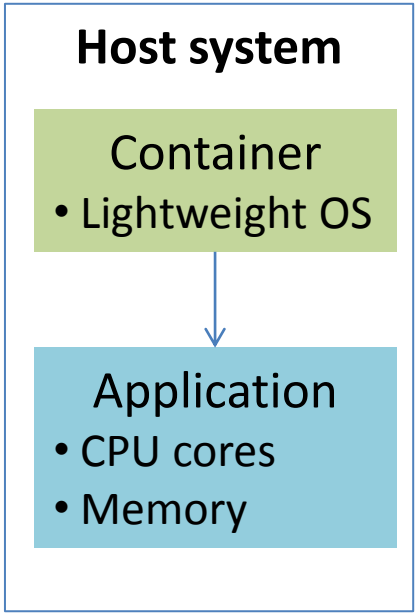
=> Need a user-friendly universal solution!

Virtualization using docker container

Docker container

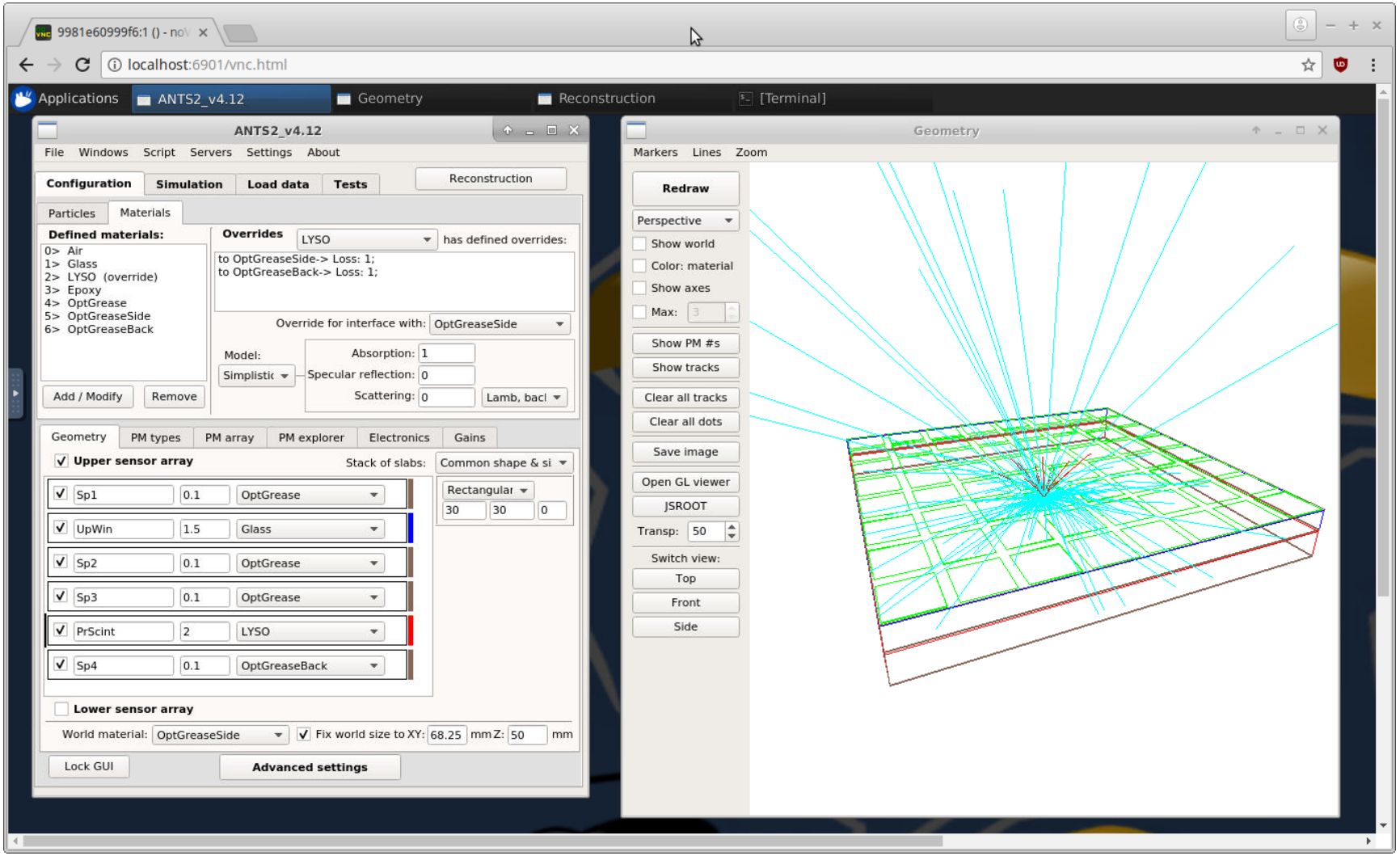
(Linux OS + ANTS2 + all required libraries)

- Runs on Windows, Linux and Mac hosts
- Nearly native code performance
- Safety (strong isolation)
- User-friendly: The already built container can be downloaded from the Docker Hub



ANTS2 on Docker: Desktop

- Docker container : ANTS2 + XFCE desktop + VNC server
- Modern (HTML5 support) browser: noVNC client



Distributed simulation/reconstruction

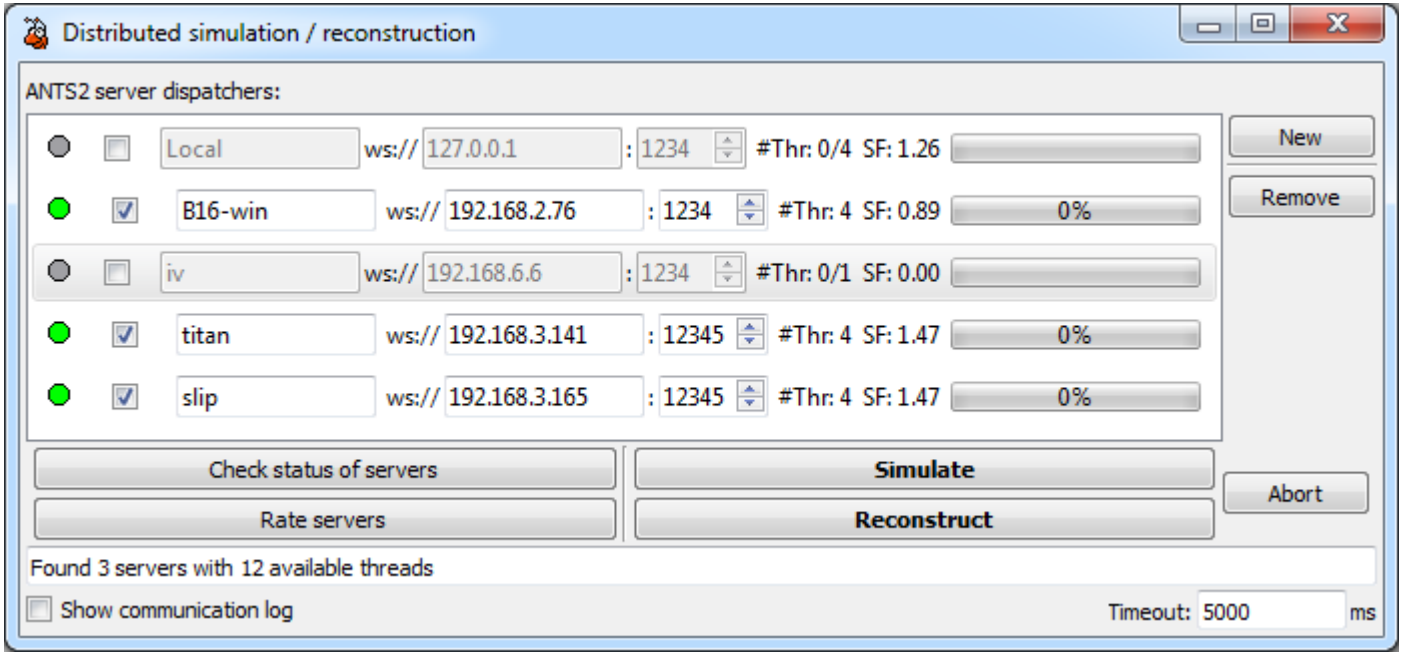
Simulation and event reconstructions can be very time-consuming.

- Server mode was added to ANTS2 in which it can (WebSocket protocol):
  - Receive binary and text data (including the detector configuration)
  - Receive and execute scripts
  - Return the results
- A dispatcher application, on a request from the master, starts a single-use ANTS2 server and gives the master the contact info (port and password).
- The master keeps the list of available server nodes and distributes the workload between them.
- The returned results are imported to the master's data hub.

Docker container with a lightweight ANTS2 server and a very small footprint dispatcher are very well suitable for running distributed ANTS2.

Distributed simulation/reconstruction

ANTS2 window for distributed simulation / reconstruction:



Another approach is to start ANTS2 on a cluster using Docker container and the batch scripting mode of ANTS2.
This approach has been tested by our colleagues from LZ collaboration, running 1000 ANTS2 jobs in parallel on a cluster.

Future?

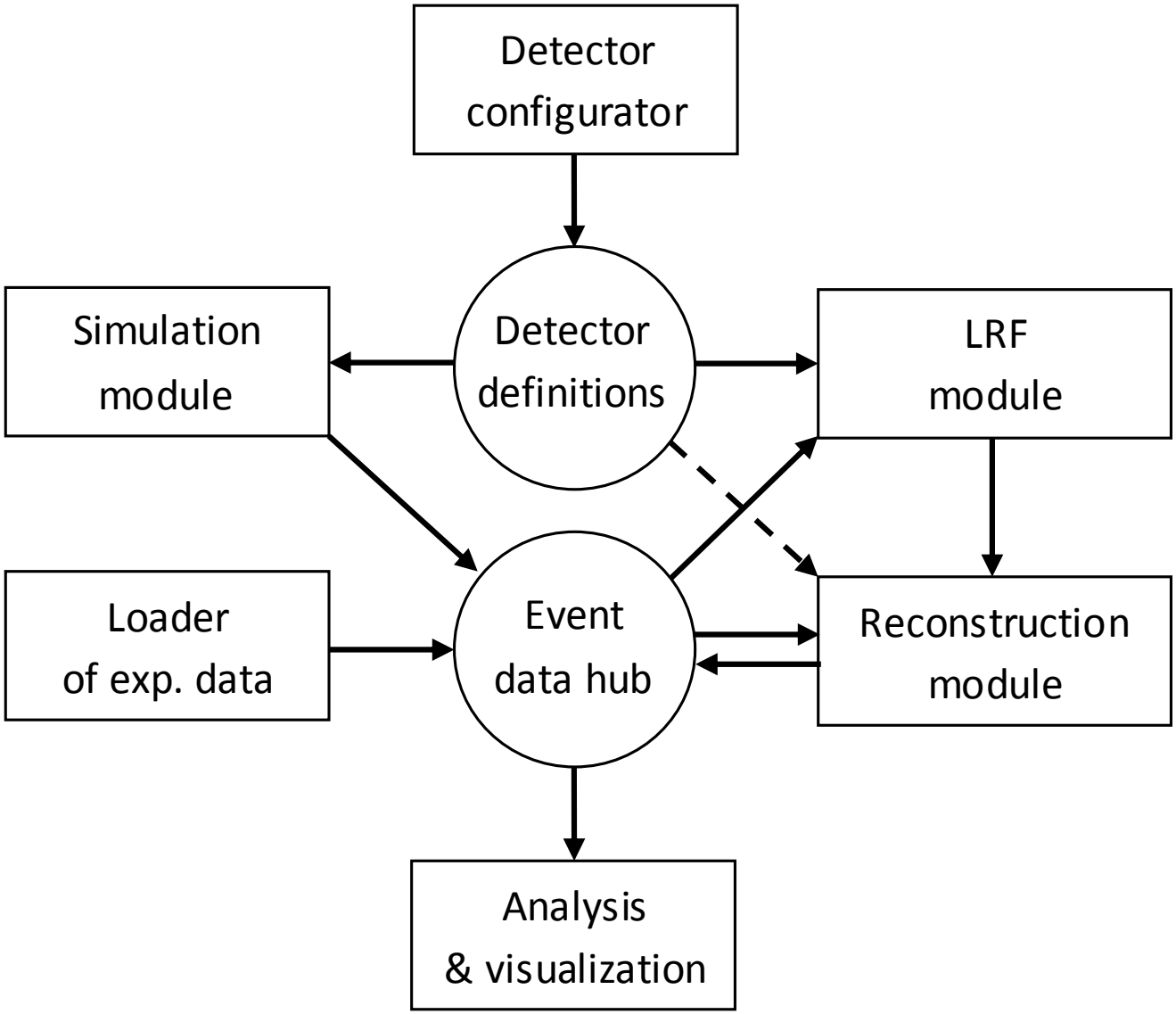
Near future:

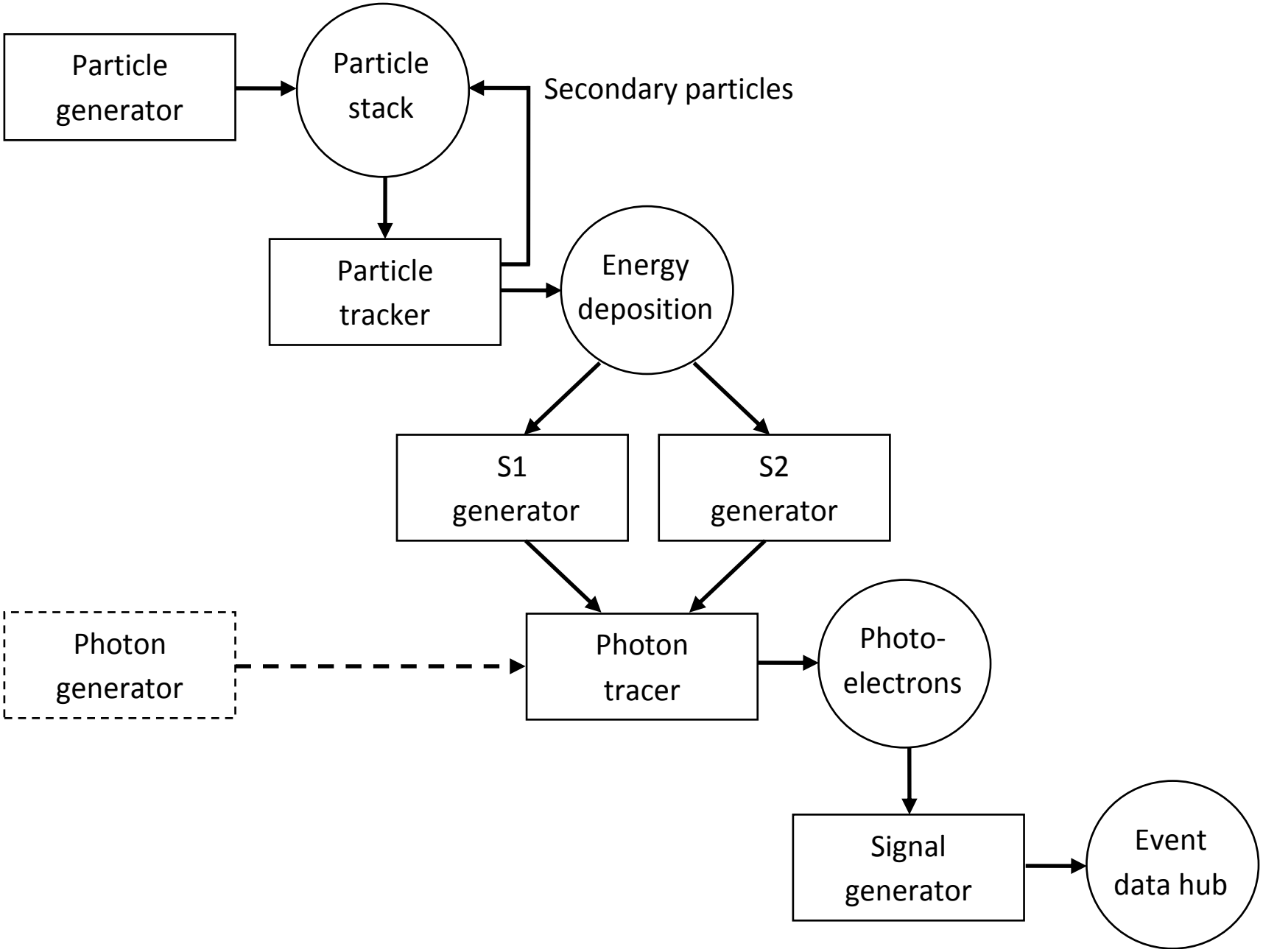
- We need a campaign of writing / updating the toolkit documentation
- Make CUDA available on Docker

Large-scale projects:

- Replace TGeoManager with VecGeom (the navigator of GeantV)
  Expecting up to x4 speed-up: SIMD + new navigation algorithms
- ANTS2-Geant4 interface: to be able to delegate tracking of particles to Geant4 and get back the energy deposition data
- Tracking of optical photons on GPUs: an attempt to dramatically speed up photon tracing. Requires a lot of custom CUDA code, but can "inherit" the navigator from VecGeom
- Reconstruction and discrimination tools for multi-vertex events
- Web interface for outreach and didactics. Already have the key components: ANTS2 server + JSROOT for in-browser detector / graph visualization!

Backup slides

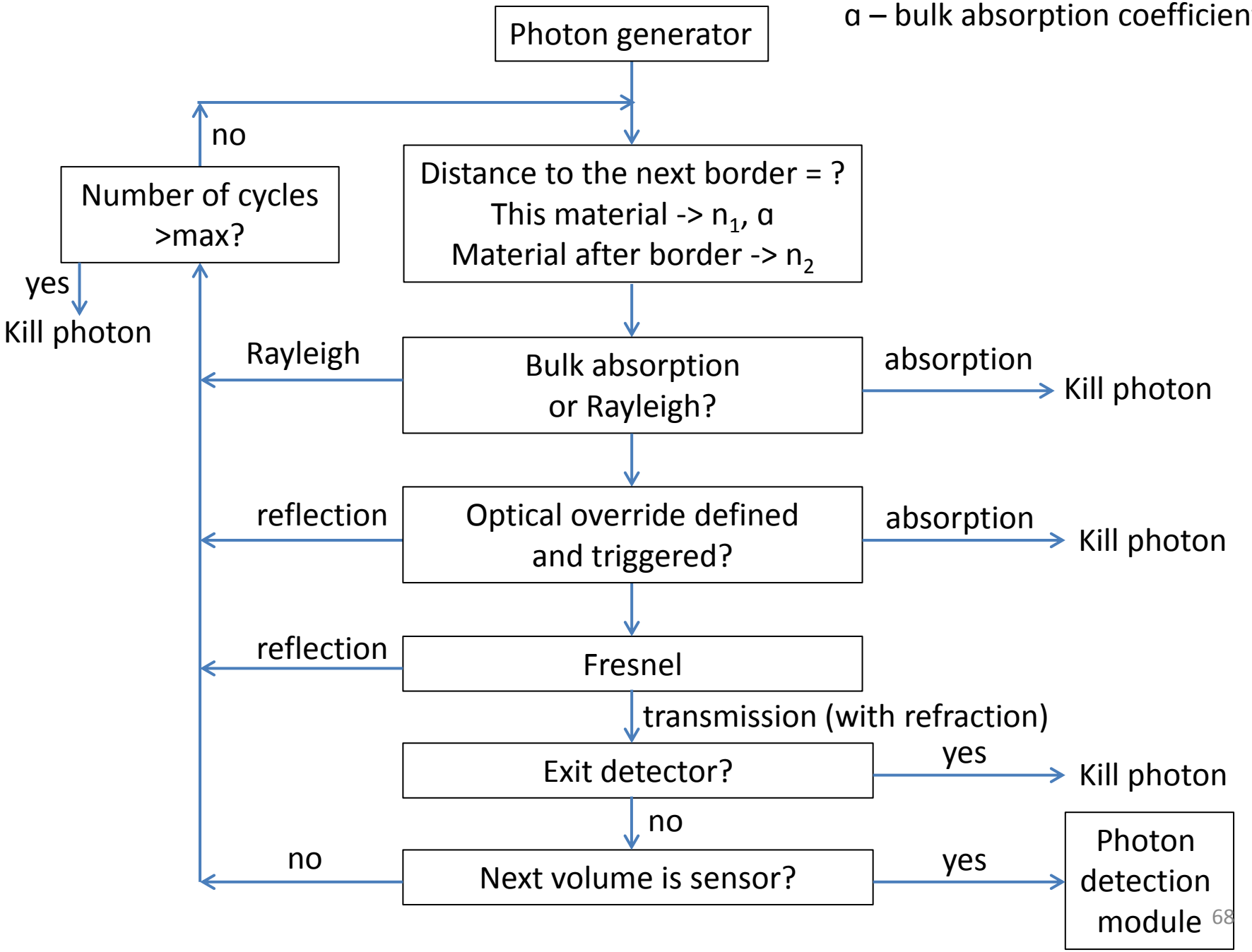

Detector configurator
Simulation module
Detector definitions
LRF module
Loader of exp. data
Event data hub
Reconstruction module
Analysis & visualization


Particle generator
Particle stack
Secondary particles
Particle tracker
Energy deposition
S1 generator
S2 generator
Photon generator
Photon tracer
Photo-electrons
Signal generator
Event data hub

Photon polarization is NOT considered!

n – refractive index

α – bulk absorption coefficient



Experimental data import

Event data import

In ANTS2 event data can be loaded from text file(s):
every line of the event file has to list the signal values of all sensors.

The line can also contain information on the position of the event
(X, Y and Z coordinates) and its energy.

The imported data can be preprocessed in ANTS2:

- Pedestal subtraction
  (a tool is provided to calculate the pedestals)
- Scaling to equalize sensor gains or to convert signals to photoelectrons
  (a tool is provided to evaluate the relative gains and estimate the signal per single photoelectron)
- Suppression of events with sensor saturation

Event discrimination tools

Event discrimination can be performed based on:

- Signal values of individual sensors or sum signal of all sensors
- Reconstructed and/or loaded event energy
- Chi2 of the event reconstruction
- Reconstructed or loaded position of the event
- Average distance to the neighbors calculated with the kNN method

Scripting tools allow to perform custom cuts using multiple criteria

ANTS2 highlights

Simulations

- 3D, custom detector geometry
- Generation and tracking of gamma rays, neutrons and positive ions
- Primary and secondary scintillation
- Time- and wavelength-resolved tracking of optical photons
- Photon scattering, wavelength shifters
- Signal generation for PMTs and SiPMs

Scintillation event reconstruction (XYZ + energy):

- Statistical algorithms
  - Contracting grids on GPUs (real-time operation)
- Artificial neural networks
- kNN-based reconstruction

ANTS2 highlights

Experimental data processing

- Import and preprocessing of experimental data
- Event discrimination tools (noise and multiple event rejection)

Detector response reconstruction

- B-spline and analytic parameterization of sensor response
- Grouping of the sensor to take advantage of the array symmetry
- Iterative reconstruction of the detector response
- A framework of tools for determination of the sensor gains

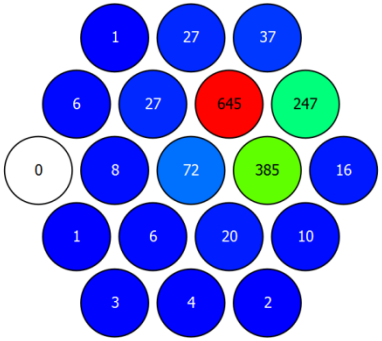
Collection of tools for characterization of PSSD performance

Iterative reconstruction of LRFs

For an array of **N** sensors, one event results in **N equations**
Event reconstruction finds X + Y + Energy → **3 variables**

If N is much larger than 3 we have an over-determined system

The distribution of the signals also “encodes” information on the detector spatial response.



Using many events, distributed over the entire detector active area (**flood field**), it may be possible to **reconstruct LRFs** for individual sensors.