Sim_telarray and helpers: auxiliary simulations and tests

K. Bernlöhr
MPIK Heidelberg

CTA Analysis & Simulation WG Bootcamp
DESY, Zeuthen
2017-06-20

Auxiliary simulation types

- Sim_telarray internal:
  - Ray-tracing: PSF, optical area, ...
  - Shadowing (needs special compilation, only 1M)
  - Calibration-type event data:
    - Pedestals dark or with NSB
    - Flat-fielding data
    - Single-p.e. measurements
- With CORSIKA:
  - muon rings
- With artificial light sources:
  - Any calibration-type events
  - Laser beams
  - Simplified muon rings

Quick view into the data

The hessioxxx package provides the tools to look directly into the simtel files:

- listio: data blocks structure
- statio: data blocks statistics
- read_hess = read_cta = read_simtel
  - Show data contents (-s | -S | -h)
  - PS plot of camera images (-p ...)
- Histogram files (x.hdata.gz)
  - hdata2root
  - list_histograms

Ray-tracing with PSF

- For PSF simulation we need
  - dummy1.corsika.gz (one event header needed!) as input file to sim_telarray
  - -C stars=... # One-line file, e.g.: 0. 70 1.0 10.0
  - -C star_photons=...
  - -C imaging_list=xt.lis
- In xt.lis we get then all photons which make it to the focal surface, even if no pixel is hit.
- See 2013 Tutorial page (also here) for details on manual procedure.

Ray-tracing for PSF made simple

- Create sub-directory (e.g. ‘PSF’) under sim_telarray directory.
- Get xpsf_z_loop_prod3.sh script from Prod3/Config/PSF into your PSF sub-directory.
- Make sure to have rx, whistogram, selop, and selxm tools installed (see 2013 Tutorial site) and in PATH (rx gets installed under sim_telarray).
- Then it is as simple as

  ./xpsf_z_loop_prod3.sh [-p] type [za [dist [rnda]]]
  e.g.: ./xpsf_z_loop_prod3.sh 1 20 12 # For LST at 20°, 12 km.

- See Prod3/Config/PSF location for results.

Resulting PSF table

- From input:
  - Flen [cm], distance [km], theta (z.a.) [deg] off-axis angle [deg], lin.off [cm] (with nom. flen)
- Computed from ray-tracing:
  - sig(x) [cm] (r.m.s.), sig(y) [cm], sig(t) [ns],
    r50 [cm], r68 [cm], r80 [cm], r95 [cm],
    xmean [cm] (c.o.g. w.r.t. expected lin.off),
    mirr.area [m²],
    xc [cm] (c.o.g.), x10 [cm], x50 [cm], x68, x80, x95
- Fit center-of-gravity of spot versus off-axis angle to derive the effective focal length.

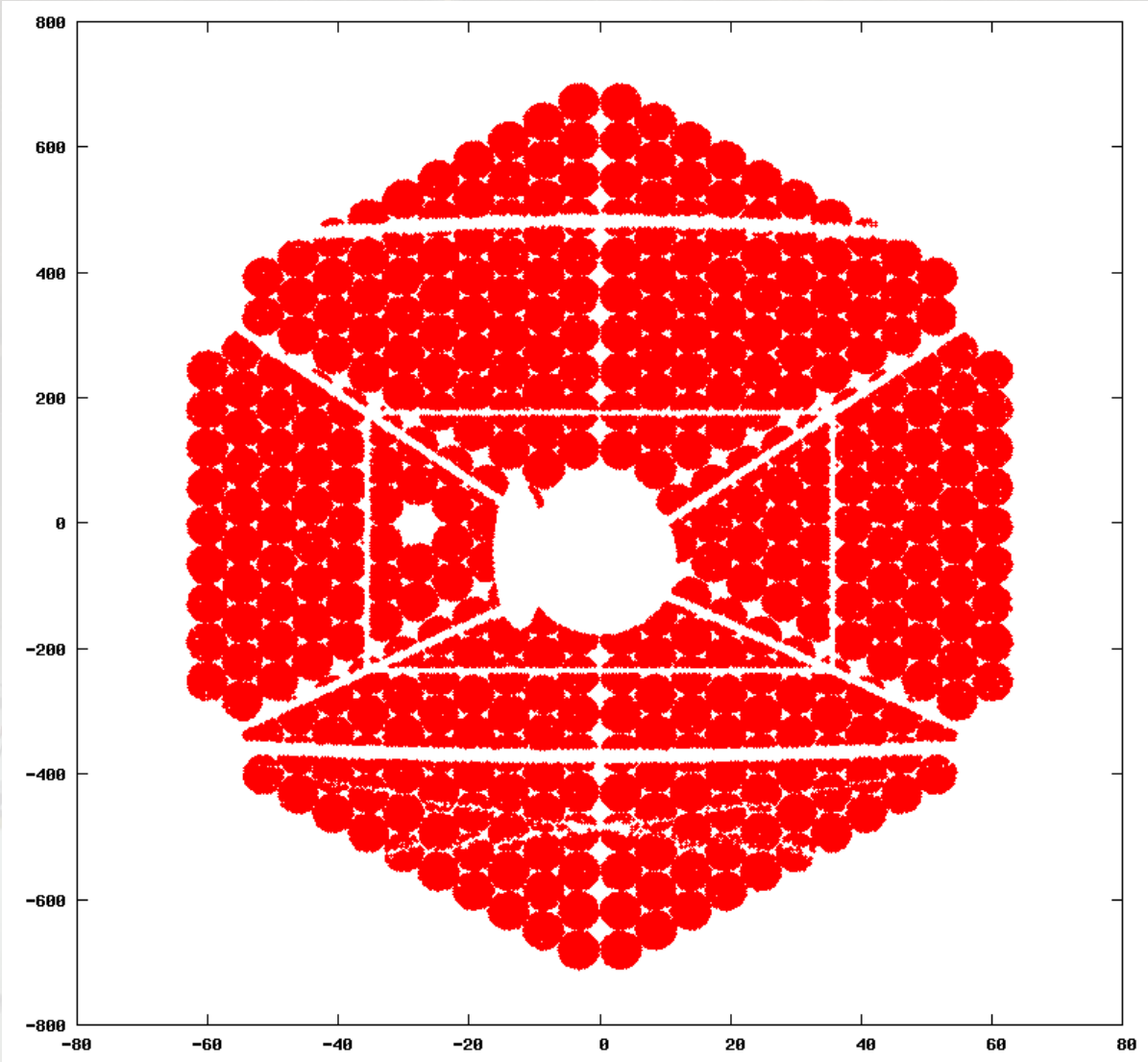
Ray-tracing for shadowing

- The ray-tracing in normal simulations only includes explicit shadowing of the camera body on incoming light:
  - camera_body_diameter = … % [cm]; now also shape
- For more explicit shadowing compile with

  -DRAYTRACING_INTERSECT_RODS

  and include file with shadowing elements (cyl.)
  - masts_file = CTA-MST_masts.dat % not latest design!
  - camera_body_diameter = 0. % with explicit element
- See impact points of lost photons etc.

Ray-tracing for shadowing



(H.E.S.S. telescope)

Ray-tracing for shadowing

- Sorry, I do not have the data for any current CTA telescope design – but you are welcome to play with shadowing in the H.E.S.S. telescopes ;)
- If YOU have the CAD data (or know the one who has it) and can extract start pos., end pos., diam. of obscuring elements (as cylinders), I want to hear from you.
- Implementation for dual-mirror telescopes was started but not completed yet. Coming soon?

Calibration-type event data

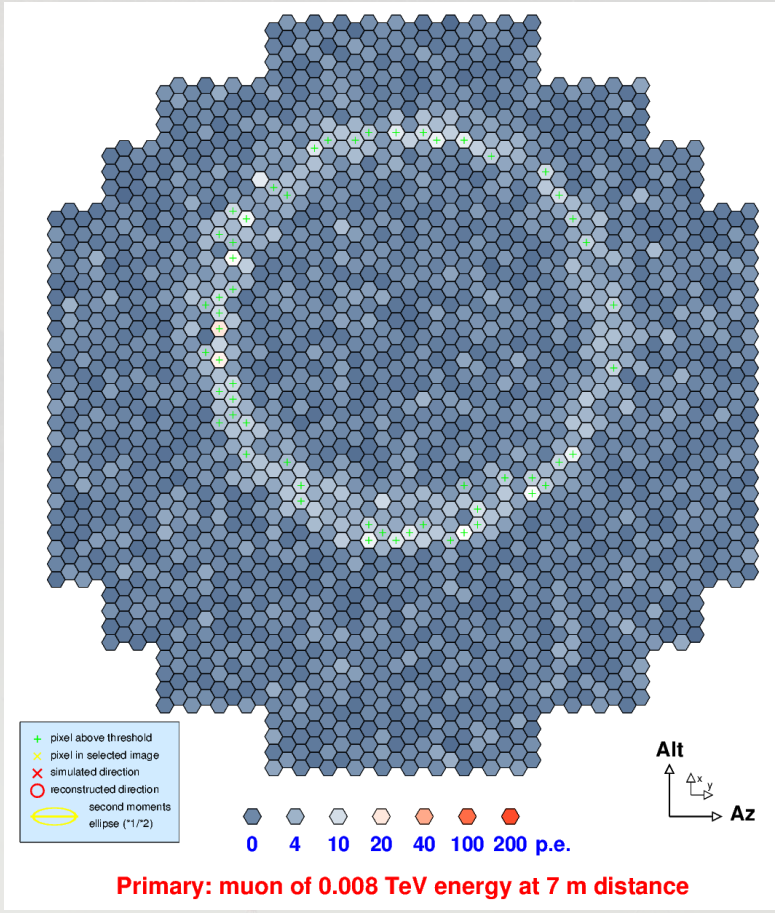
- In sim_telarray alone, without artificial light sources:
  - DARK_EVENTS, PEDESTAL_EVENTS, LED_EVENTS (one per pixel), LASER_EVENTS (flat-field style illumination assumed).
  - Various other parameters for NSB, light pulse shape and level, ...
- Resulting events precede normal events but:
  - wrapped in another eventio block (envelope),
  - needs extract_calibevent to extract to new file.

Calibration-type event data

- Exact match for each pixel only valid for this one run; the next run will see a new random assignment of QE, gain, pedestals, ...
- Including these events for each run is too much overhead.
- But you can still apply your own (single-p.e., pedestal, gain, ...) calibration procedures and compare them with the MC-provided calibration parameters on a statistical basis.
  - Remember calib_scale (default=0.92) in read_hess.

Muon rings with CORSIKA

- Need to start high enough to include all Cherenkov light which may be seen in selected muon rings.
- One problem is multiple scattering – so you don't know the actual impact position on the dish.



- Could use IACTEXT and start muon particle output – but never done that.
- Easier: use idealized artificial "muons".

Artificial light sources

- The IACT interface can also be used to write your own CORSIKA-IACT-style data, without involving CORSIKA.
- Writing such data manually involves not much more than a hundred lines of code.
- With the LightEmission package (included with sim_telarray but also available separately) the repetitive work is in a C++ library and a very simple program can fit into 10-15 lines.
- Example applications included.

Artificial light sources

Artificial light sources available:

- SimpleLightSource:
  - Really simple: point-like, isotropic, monochromatic, simple light pulse shapes (Gaussian or flat-top).
- LightSource:
  - Much more flexible,
  - Can define spectrum, pulse shape, and angular distribution either via user table or with keyword
    - e.g. “angular_dist.dat” or “Cone:1.23”
  - Light source may be extended and have some propagation speed.

Artificial light sources

- In older presentations (Bologna meeting, telcons) and on web site you can find animations with some unrealistic (but nice) applications, e.g. the “superluminal optocopter”.
- More realistic/useful applications provided:
  - ff-1m.cc: Flat-fielding of single-reflector telescopes,
  - ff-gct.cc: Flat-fielding GCT style via secondary,
  - ls-beam.cc: A laser beam crossing the FoV,
  - fake-muon.cc: An idealized muon emitting the right amount of Cherenkov light on a cone defined by index of refraction of air and its propagation speed.

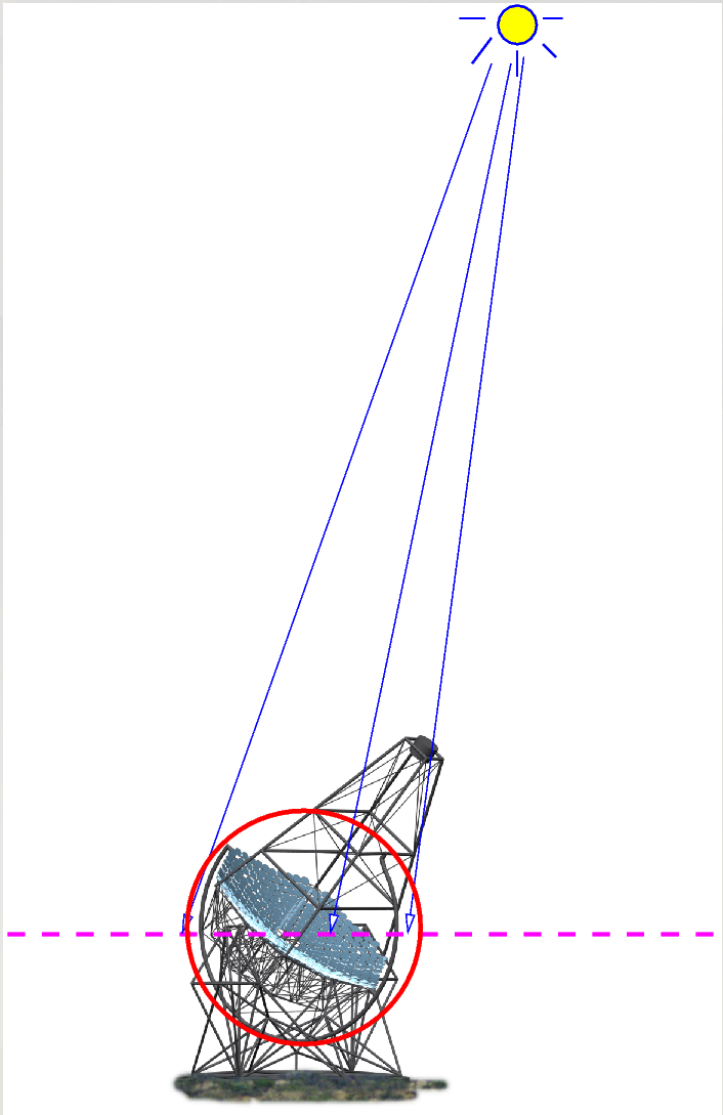
Using artificial light source

with sim_telarray:

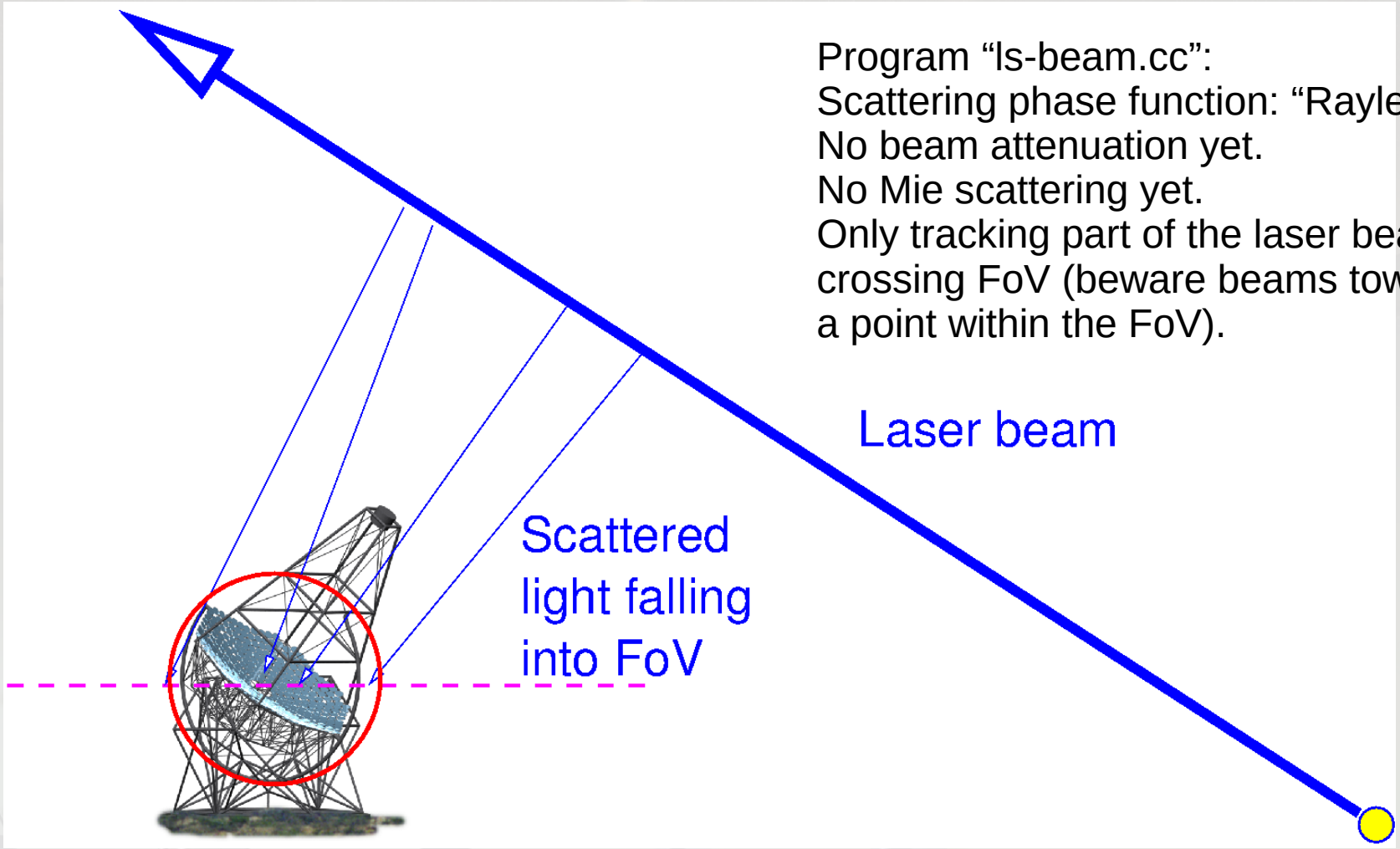
- Whole telescope involved: -C bypass_optics=0
- Single-reflector telescope and light goes directly into camera (flat-fielding unit or test stand): -C bypass_optics=1
- Dual-reflector telescope and light goes via secondary into camera (e.g. GCT flat-fielding): -C bypass_optics=1
- Dual-reflect tel., light directly into camera: -C bypass_optics=2

Artificial light into full telescope

- Use sim_telarray just like with CORSIKA Cherenkov light.
- Normally one telescope (with mid-plane = obs.lev.)
- Multiple telescopes are possible and see the same light source(s).

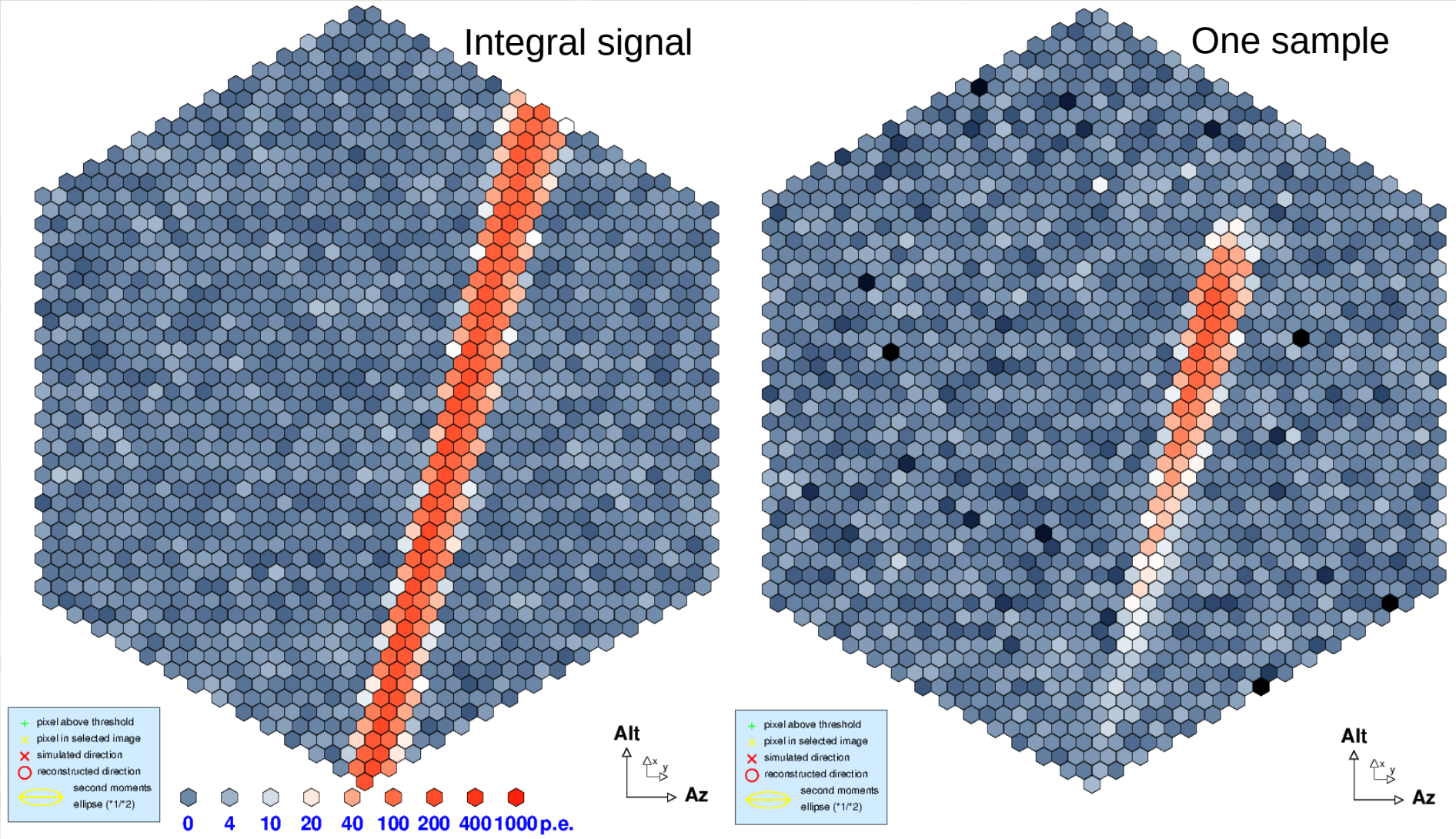


Laser beam crossing FoV



Program “ls-beam.cc”:
Scattering phase function: “Rayleigh”.
No beam attenuation yet.
No Mie scattering yet.
Only tracking part of the laser beam crossing FoV (beware beams towards a point within the FoV).

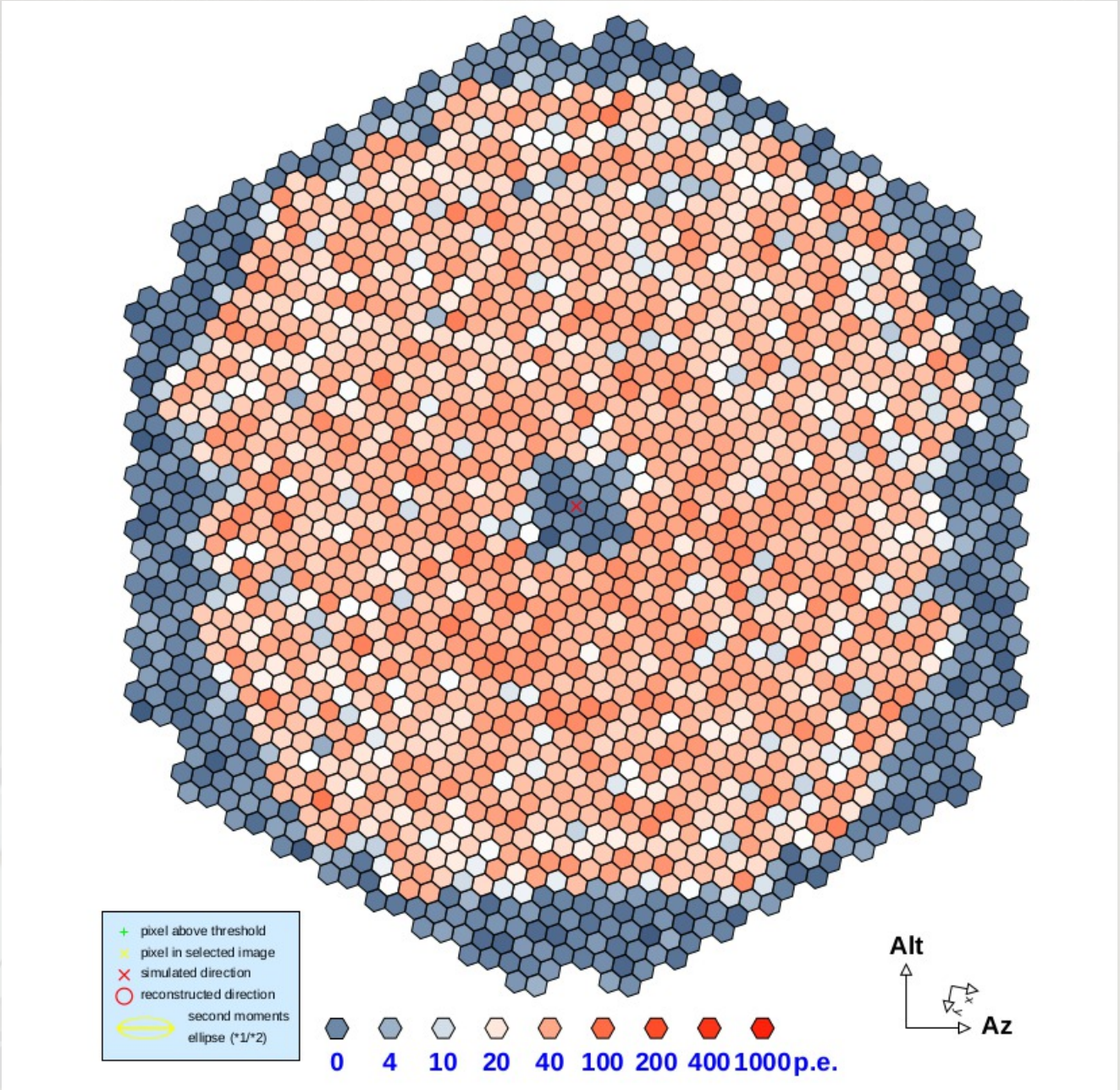
Laser beam crossing FoV



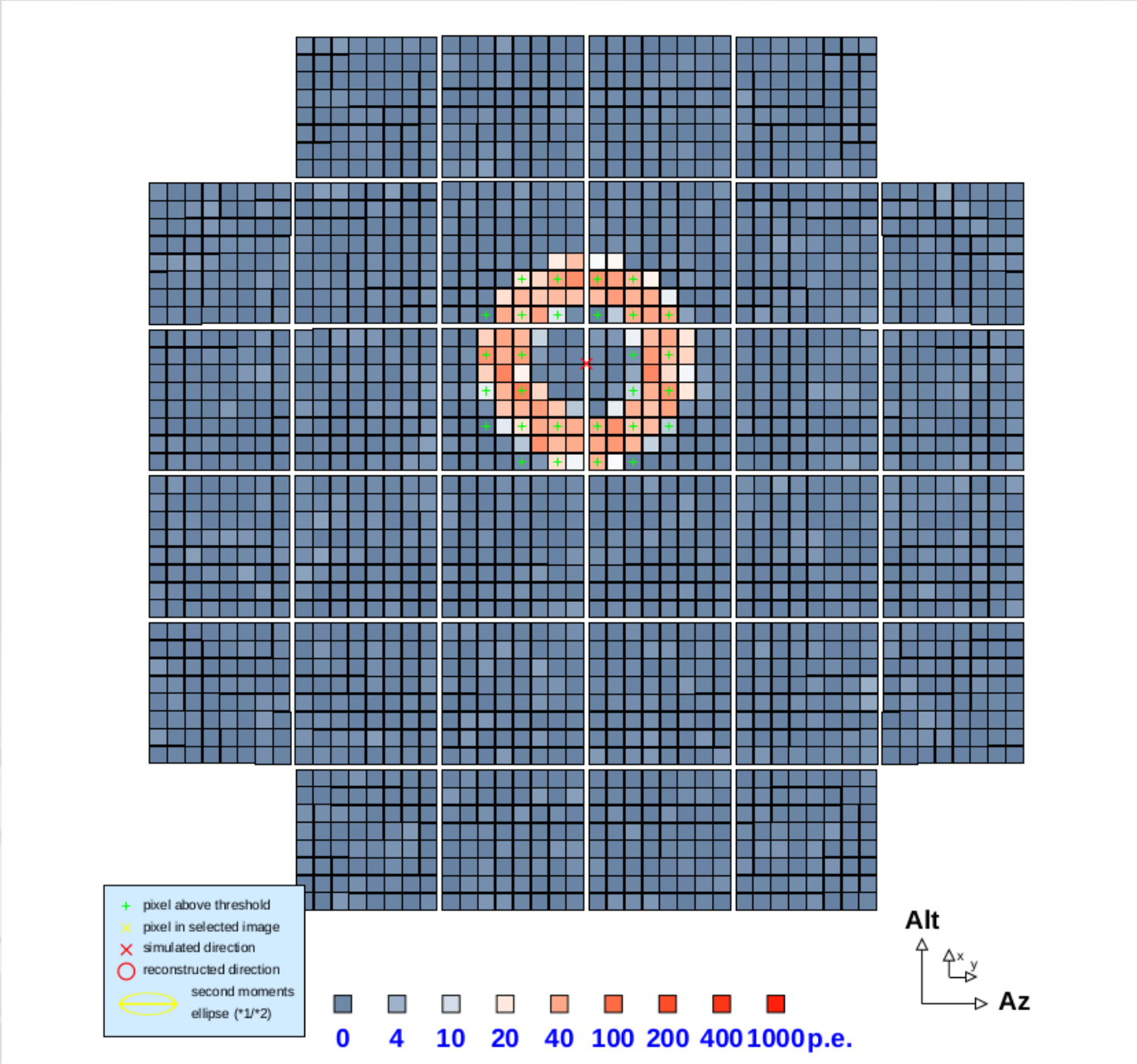
“Illuminator” light source

- Take care that light source is not at relative altitude zero (telescope mid-plane) since data format cannot represent the resulting light.
- See Illu.pdf for prepared example images.
- Try Illu_test.sh script for yourself (although that will fail for SCTs, you need a prod3-sc or prod3-demo build to run through all of it.)

LST at d = 350 m (0 deg.)

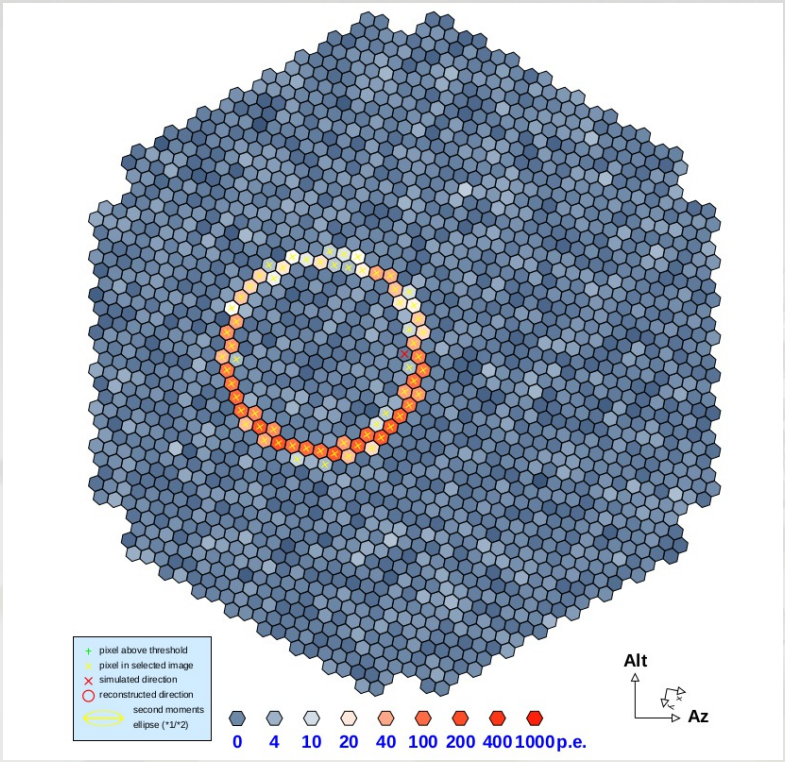


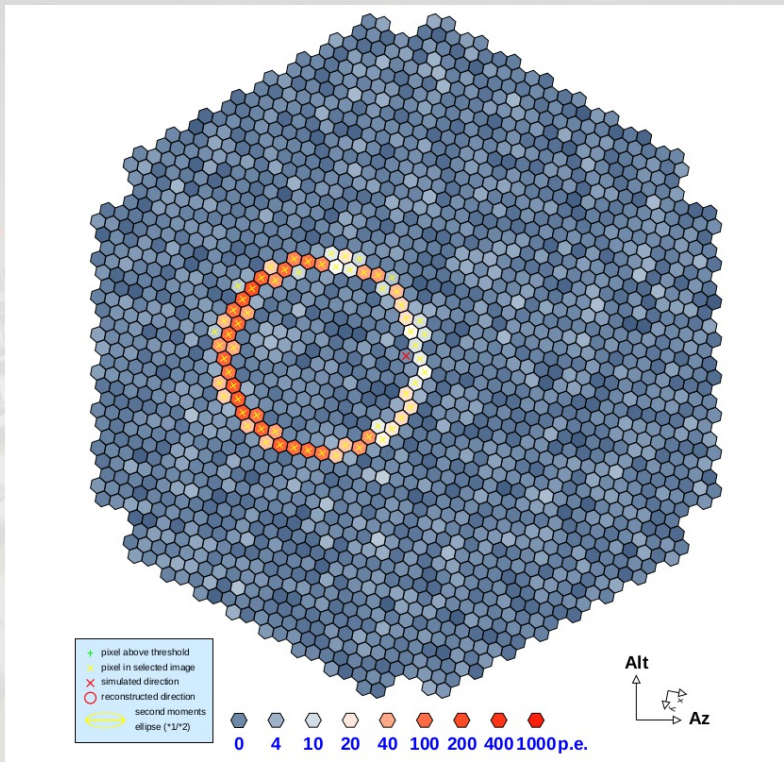
SST−GCT−S at d = 120 m (1 deg.)



Fake muons

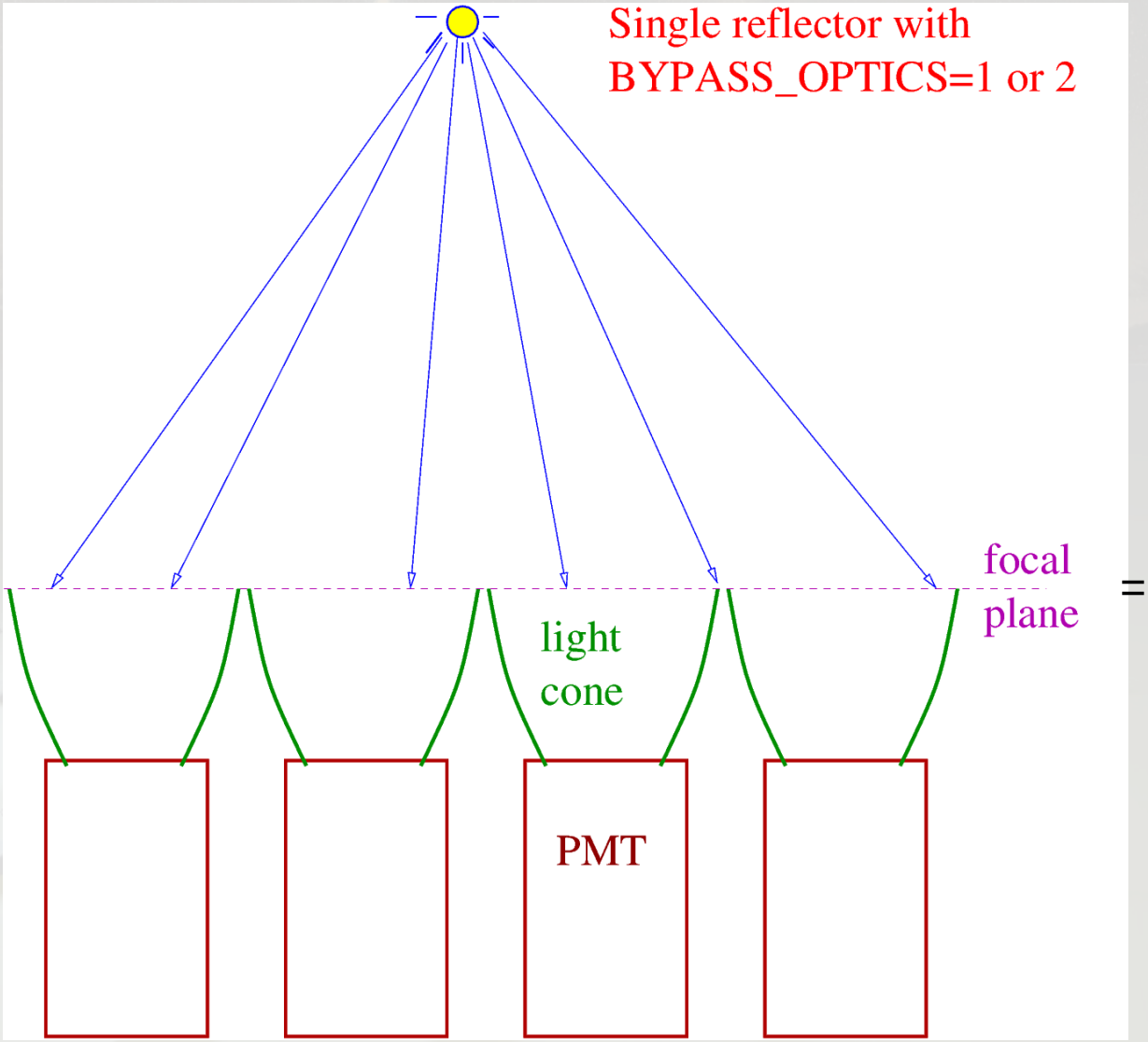
- You can directly set the actual impact position (in telescope sphere mid-plane), the starting altitude and direction, the index of refraction (constant so far), and the charge and speed of the particle.





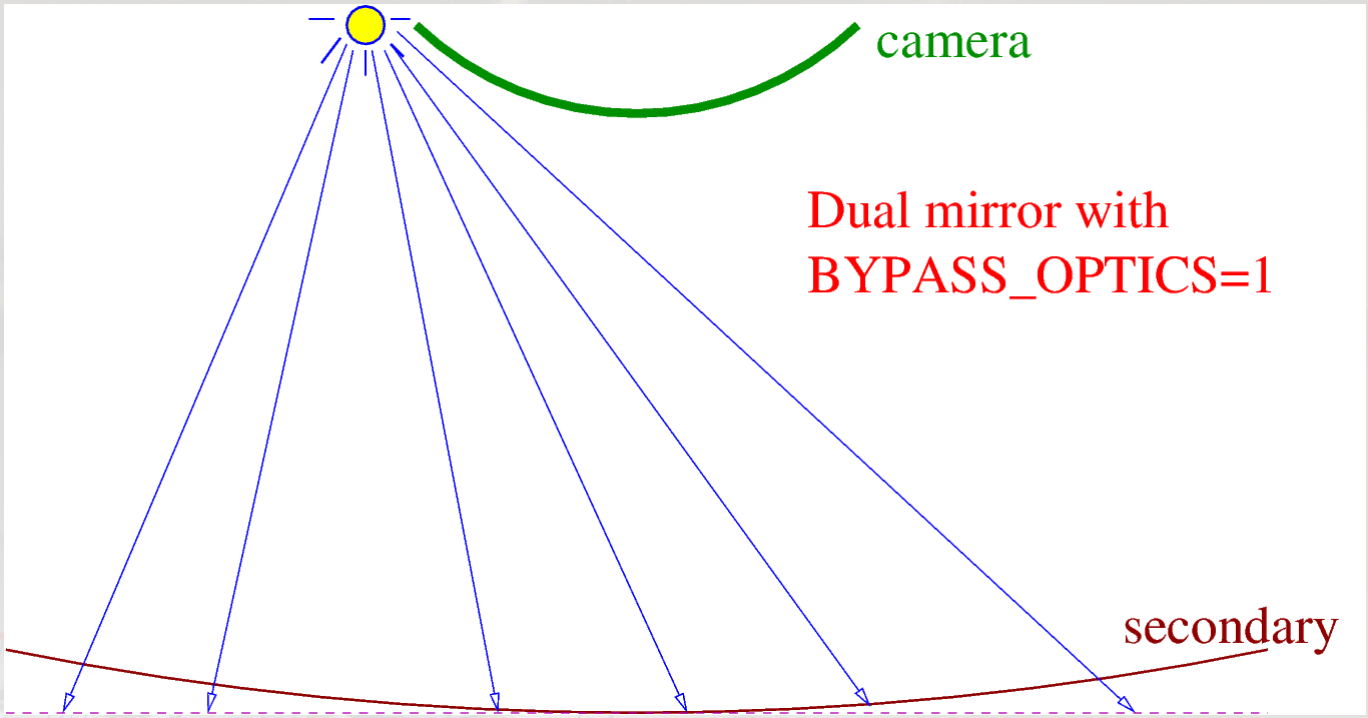
MST-NC, different impact positions

Artificial light into 1M camera



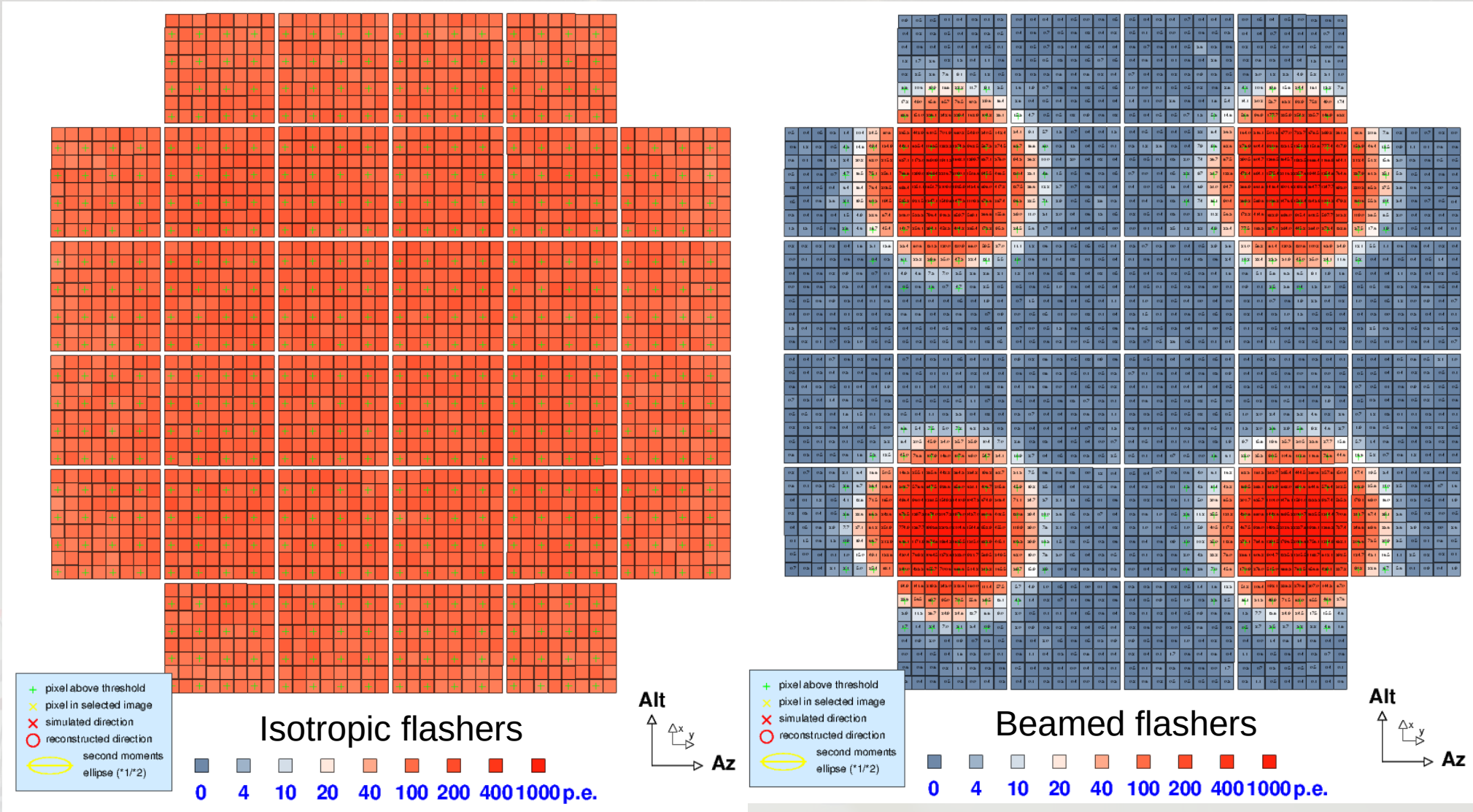
= sphere mid-plane

Dual mirror case bypassing primary

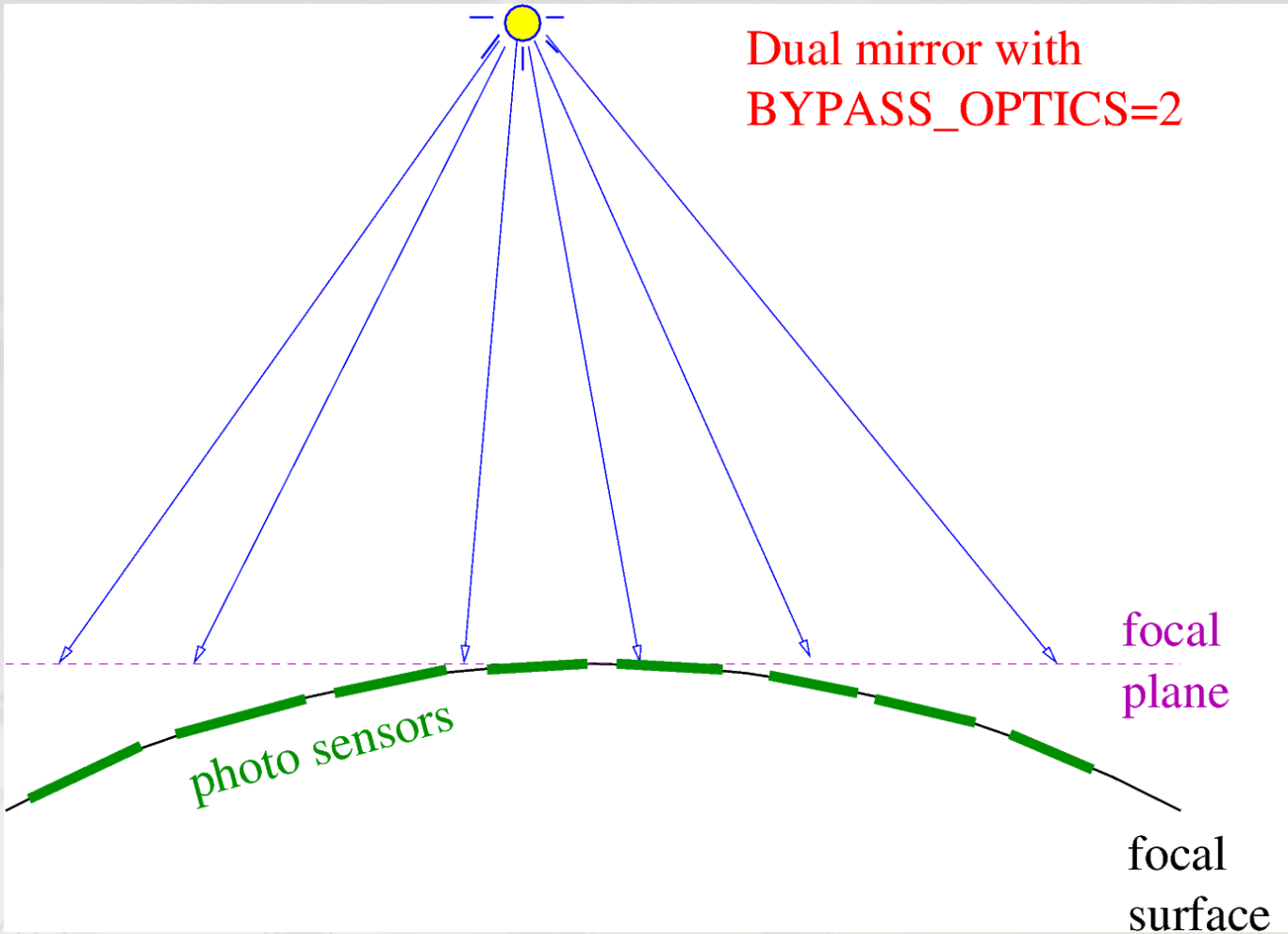


Typical application: GCT-style flat-fielding, with four light pulsars in the corners of the camera, illuminating the camera via reflection on secondary.

GCT flat-fielding unit



Dual mirror case bypassing both reflectors



Possible application: SCT-style flat-fielding

Look into data files

- Before starting:
  - cp hessioxxx/EventioRegisteredNames.dat \
    ~/.EventioRegisteredNames
  - Use ‘--help’ option first on each program.
  - If you want to see more than the first 20 pixels with ‘read_cta -S’, set the MAX_PRINT_ARRAY env.var.
- Data block structure with listio:
  - listio -s -n x.corsika.gz | less
  - listio [-s] -n x.simtel.gz | less
- Statistics of data blocks:
  - statio -t x.simtel.gz; statio -t x.corsika.gz

read_hess / read_cta / read_simtel

- Can show and explain the data contents:
  - read_cta -s [-h] x.simtel.gz | less
- Can produce Postscript file with camera images (integrated as well as samples, optionally with true p.e. number, pixel IDs, ...)
  - read_cta -u -r 4 –integration-scheme 4 \
    –integration-window 7,3 \
    -p x.ps –plot-with-true-pe –plot-with-sum-only \
    x.simtel.gz
  - gv x.ps

Histograms from sim_telarray

- What can you do with the ‘hdata.gz’ files from sim_telarray (or read_cta)?
- Convert them into a format that you prefer, e.g.
  hdata2root x.hdata.gz x.root
- Use the list_histograms tool, e.g.
  list_histograms x.hdata.gz
  list_histograms x.simtel.gz
  list_histograms -h 12 x.simtel.gz
  list_histograms -H 12 x.simtel.gz

Conclusions

- Sim_telarray has auxiliary modes and output data for PSF modeling, shadowing calculations, calibration-style events without the need for actual such events.
- Even better (and often more realistic) is to simulate artificial light sources to do that.
- Muon rings can be generated with Cherenkov light from CORSIKA or artificially.
- There are useful tools (coming with the hessioxxx package) to understand simtel data.