Simulation of artificial light sources

for improving and verifying the understanding how your IACTs work

K .Bernlöhr

MC as an approximation to the real world

- MC simulations can, at best, only be an approximation for the real instruments, either
  - because deliberately simplified, e.g. for computing efficiency reasons,
  - because of lack of knowledge about the real instrument properties,
  - or even due to misunderstandings or bugs.
- Before you can claim with simulations that an instrument performs as required, you need to trust that simulations are *accurate enough*.

Simulation of test setups

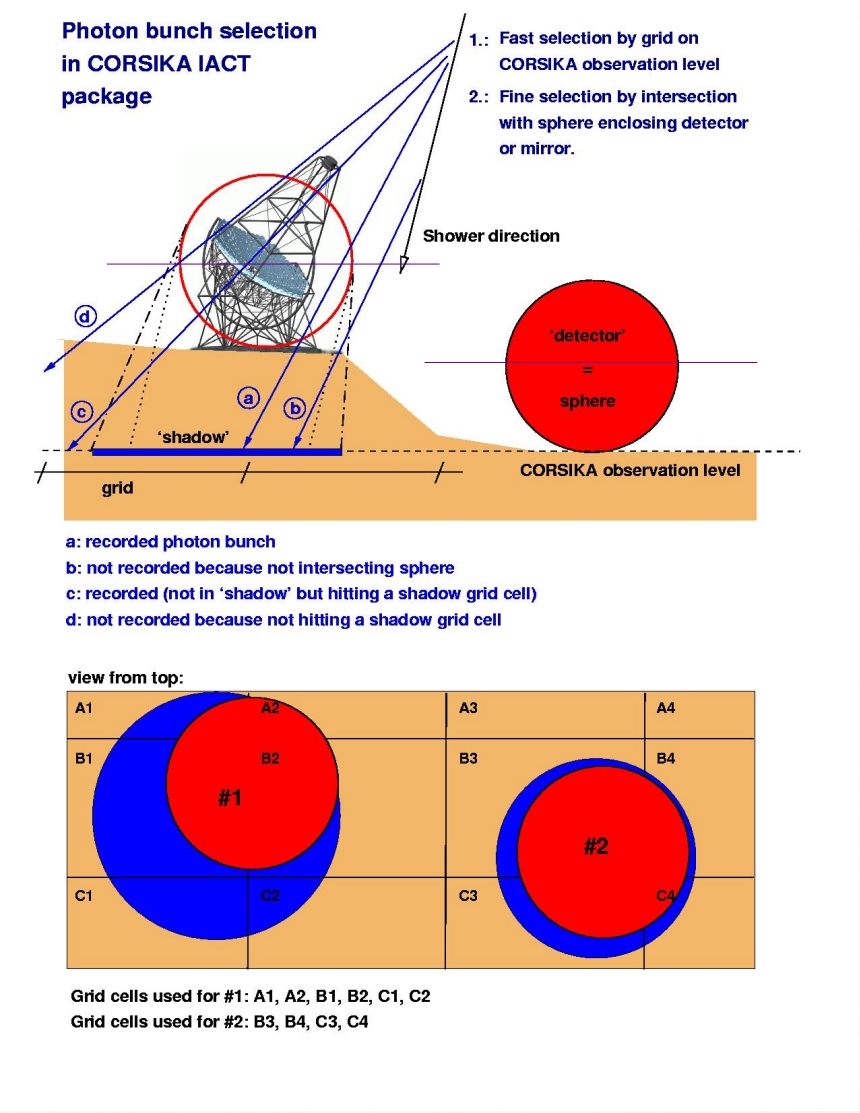
- Verifying test setups with the same simulation code & configs as in shower simulations is an effective way to re-assure that MC corresponds to reality.
- Camera tests may include:
  - flat-fielding-style light flashers + “NSB” with illumination levels from sub-p.e. to 1000s of p.e.
    - may allow trigger tests by enabling parts of camera but not able to emulate time gradients,
  - camera test facility (?), either based on optical fibers (from single or few LEDs/laser) or on LED arrays.
- Later: understand calibration light sources.

A variety of camera test and calibration equipment is planned

- Camera test and evaluation devices
  - Light sources far from (partial) camera, like F-F U.
  - Close-by light sources, e.g. optical fibres.
- Illuminator shining onto the whole telescope
  - variant: octocopter light source
- Calibration devices in the telescopes
  - Flat-fielding unit in dish center, secondary center, or from next to camera over secondary into camera.
- Central laser facility (?)
- Want to be able to simulate all of them!

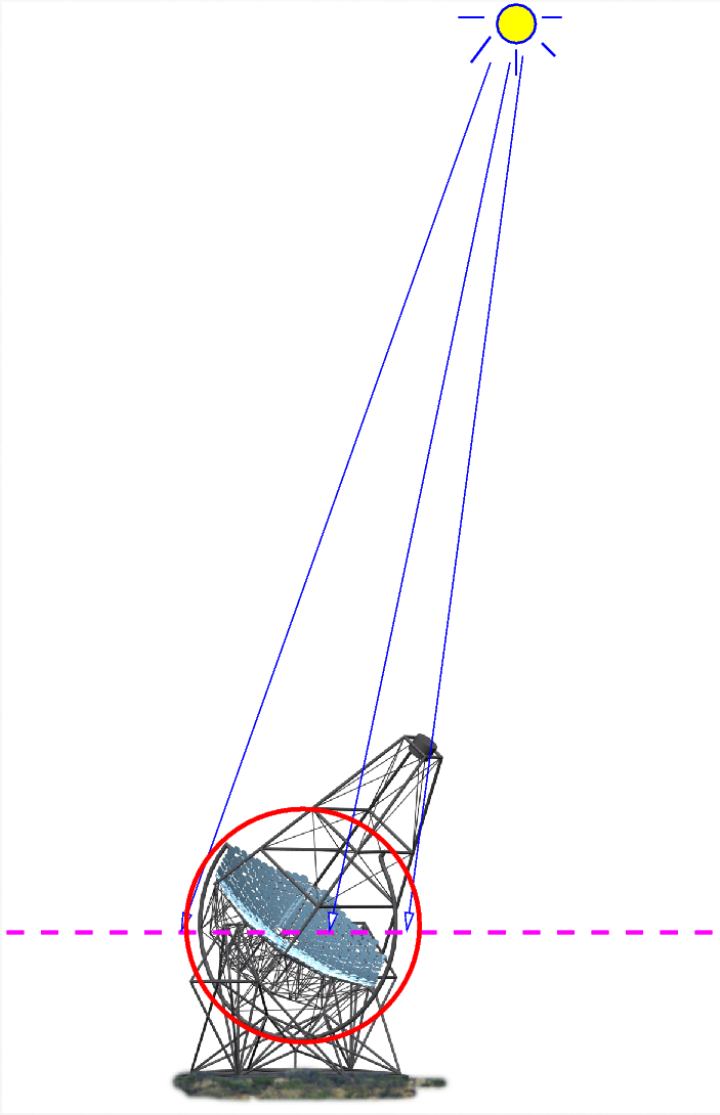
CORSIKA/IACT interface

- Store photon bunches of unspecified or of a specific wavelength at arrival in mid-plane of fiducial sphere.
- Grid with assigned telescopes good for showers and many telescopes but can be disabled during compilation (v1.50).



Replace air showers (CORSIKA) with an artificial light source

- The CORSIKA IACT interface can be easily linked with other C/C++ programs, making it easy to simulate artificial light sources in the same data format.
- Prefer fiducial sphere mid-plane = detection level (no offset of sphere completely above d.l.).



One limitation of the format

- With the CORSIKA IACT interface, the arrival position of photon bunches is stored as the intersection point with the horizontal mid-plane of the fiducial sphere.
- Thus exactly horizontal light is not supported.
- The grid-cell ray-tracing acceleration scheme is better by-passed with artificial light sources, avoiding loss of photon bunches.
  - Need IACT/atmo package version ≥ 1.50 and compile with -DIACT_NO_GRID; also allowing mid-plane at “detection level”.

New IactLightEmission code

- Mimics CORSIKA IACT output.
- Using IACT/atmo code for raytracing to fiducial sphere and output (just like CORSIKA does).
- Using random number code from sim_telarray.
- Implements a few basic light source types (extensible as C++ classes):
  - Isotropic, monochromatic, simple pulse shape
  - Isotropic, user-defined spectrum and pulse-shape
  - Non-isotropic, user-defined spectrum and p-shape

Efficiency trick for distant sources

- For light sources well outside the “fiducial sphere”, ray-tracing all generated photons to intersection with sphere would be inefficient.
- Some applications could be sped up by a factor of a thousand by generating only those photon bunches going in the direction to the sphere:
  - Easier to solve for isotropic than for anisotropic light sources;
  - solution is internal to light source class and application code does not care about it.

User program for light sources

- Few lines of codes for basic simulation:
  - Run parameters
    - altitude, atmospheric profile, radius of fiducial sphere, ...
  - Any number of light sources, firing requested number of photons from given location around given time.
  - Repeat this for the requested number of events.
- Laser-beam light source (simple version available) may also need aerosol distribution in height, scattering phase function etc.

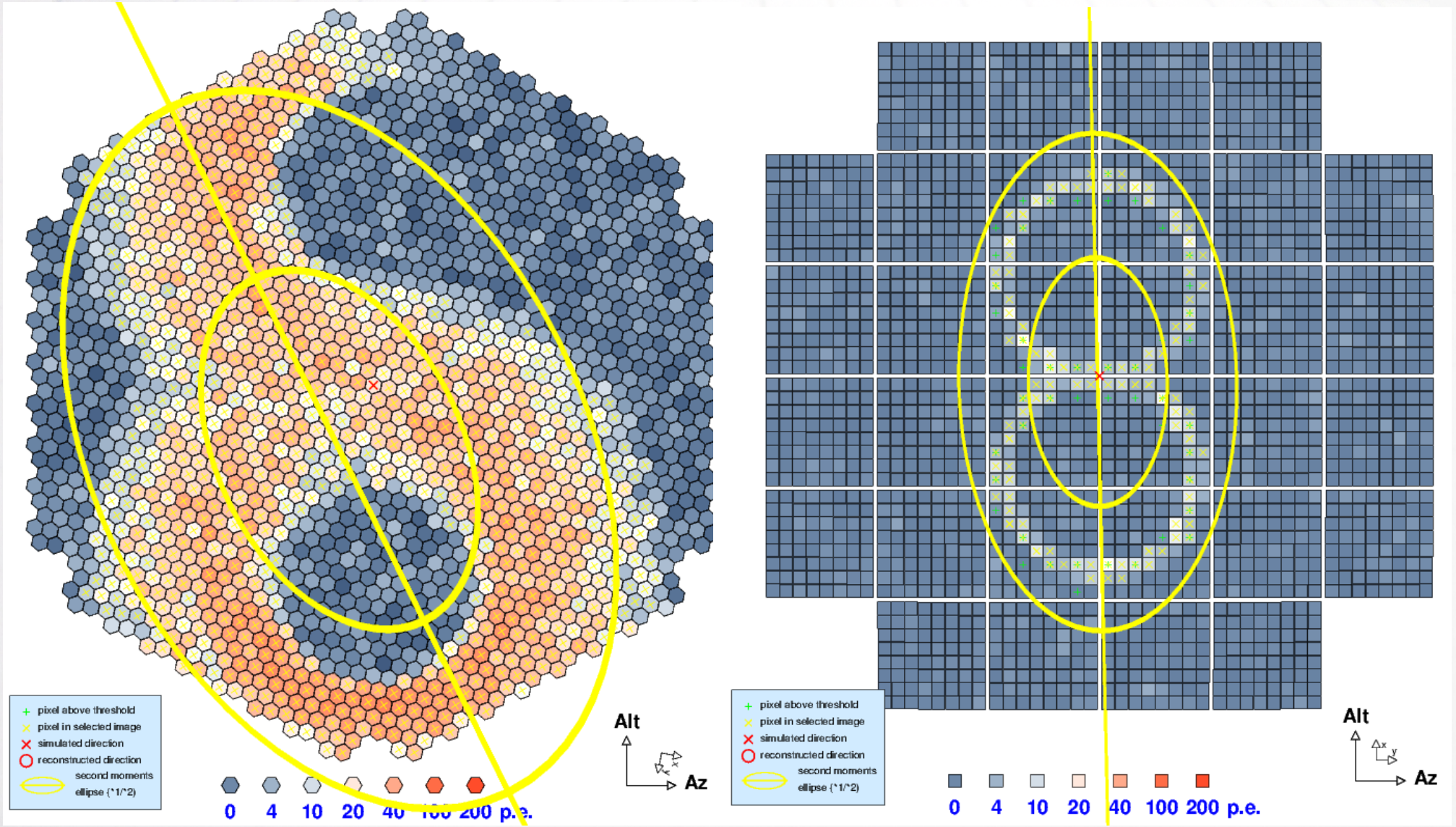
The superluminal octocopter

as a very unrealistic example lightsource

```
int main()
{
   Run run(1,0.,1,2.5e2);
   double rc = 20e2;

   run.NextEvent();  // Actually just one event here

   SimpleLightSource ls;

   for (double t=0; t<20.1; t+=0.5)
   {
      ls.SetPosition(SpaceVect(rc-rc*cos(2.*M_PI*t/20.),
                               rc*sin(2.*M_PI*t/20.),
                               1000e2));
      ls.Emit(run,300000,2.,t);
   }
   for (double t=0.5; t<20.1; t+=0.5)
   {
      ls.SetPosition(SpaceVect(-rc+rc*cos(2.*M_PI*t/20.),
                                rc*sin(2.*M_PI*t/20.),
                                1000e2));
      ls.Emit(run,300000,2.,t+20.);
   }
   // Run destructor takes care of the rest
}
```

The superluminal octocopter

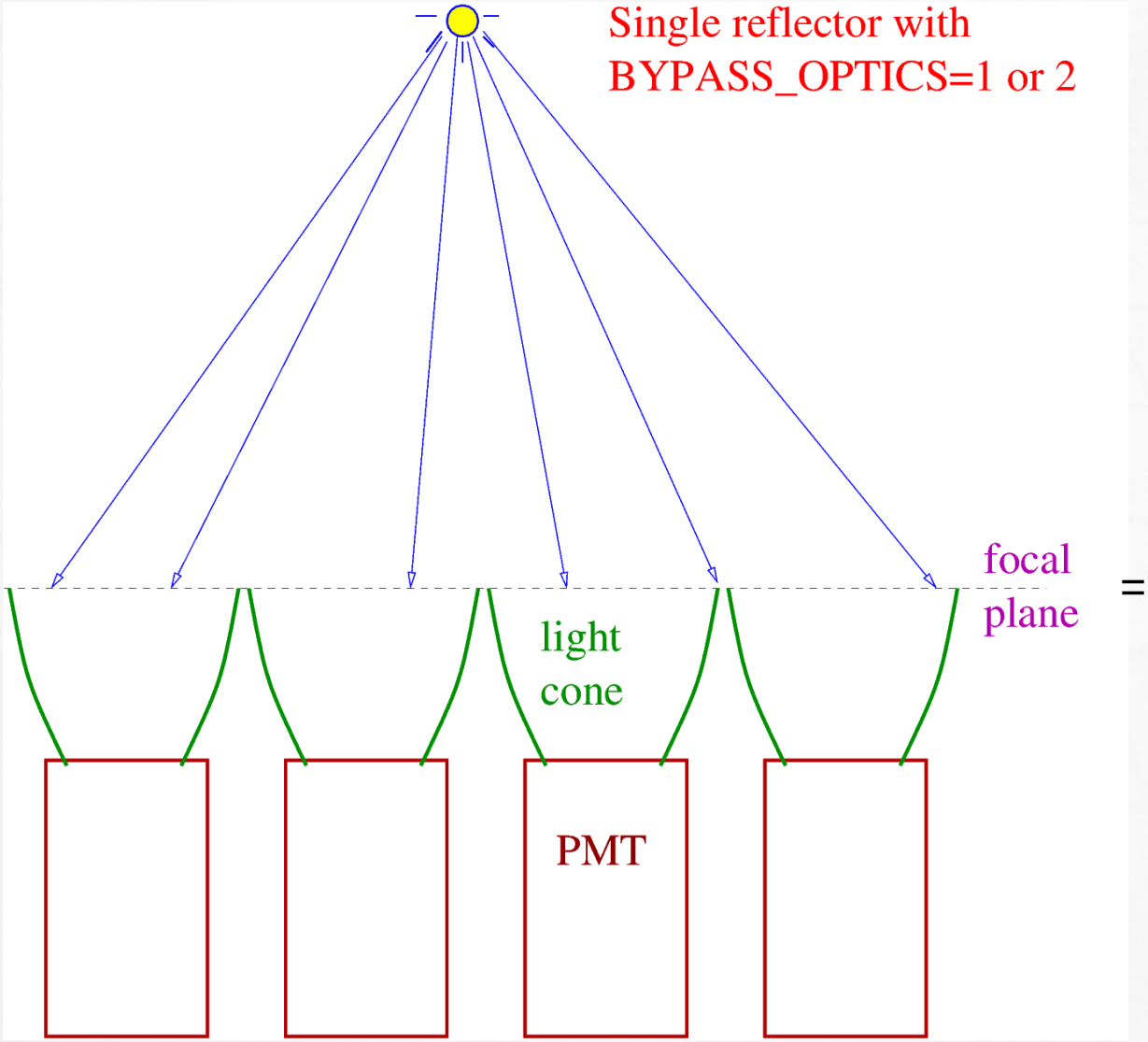
as a very unrealistic example lightsource



Light sources directly illuminating the cameras (e.g. flatfield unit)

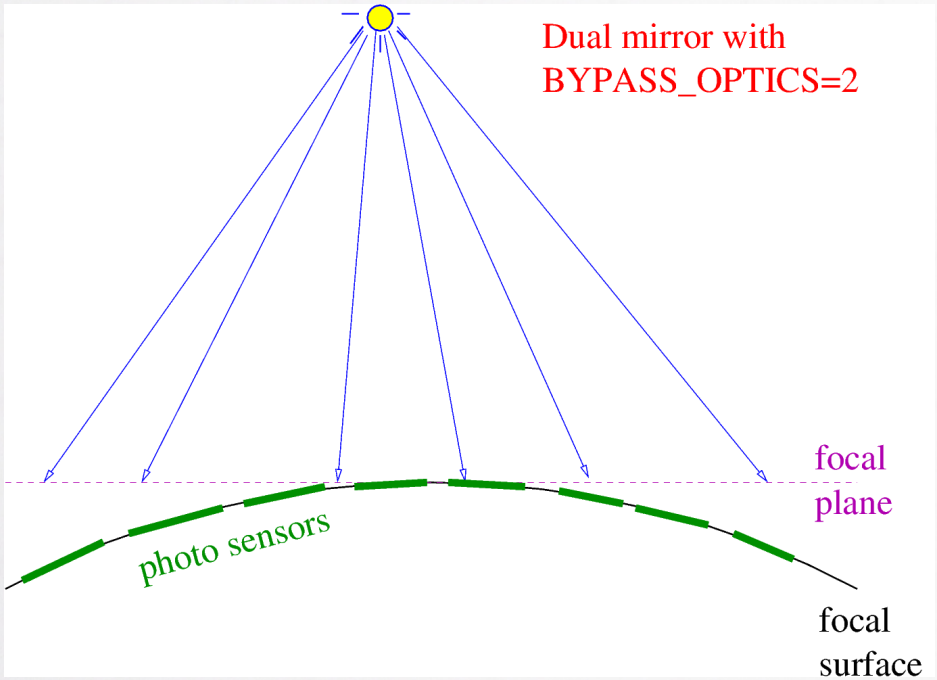
- Focal plane is at fiducial sphere mid-plane, camera looking up, sphere radius ≥ camera radius.
- Sim_telarray is instructed to bypass optics raytracing (BYPASS_OPTICS = 1 (or 2)).
- For dual mirror optics, partial ray-tracing, through secondary only, is available. Center of secondary shape will be at sphere mid-plane, secondary looking up (sphere must be big enough to include secondary completely).

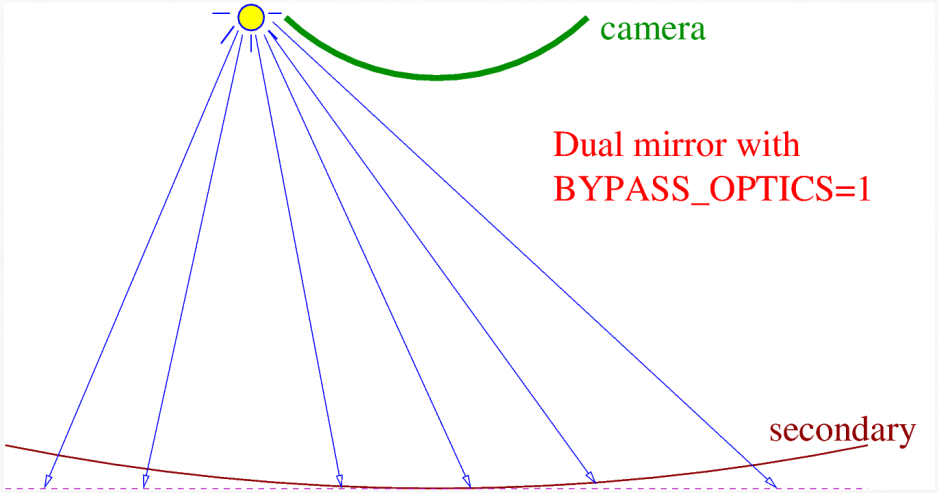
Single reflector case for bypassing optics raytracing



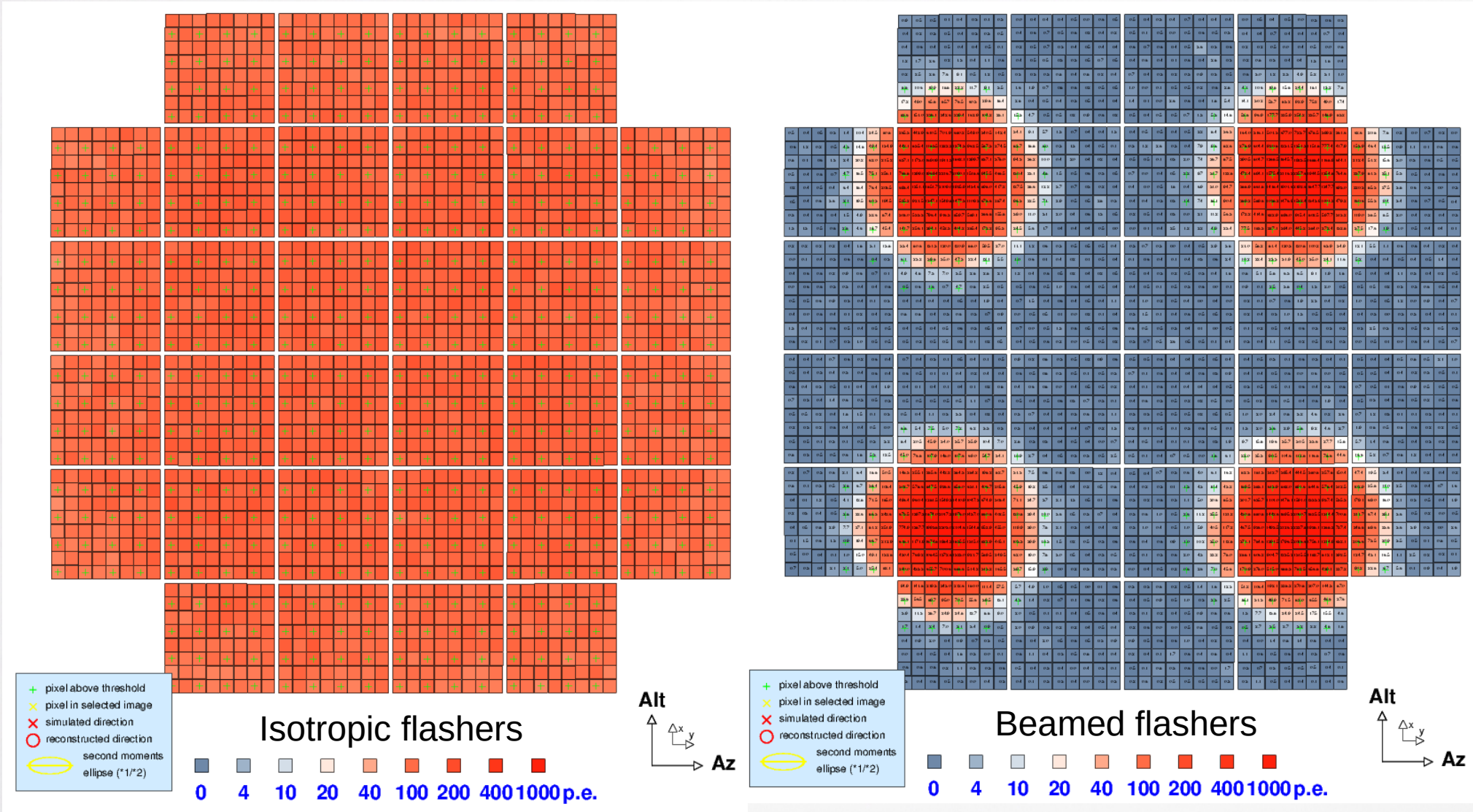
= sphere mid-plane

Dual mirror cases for bypassing optics raytracing

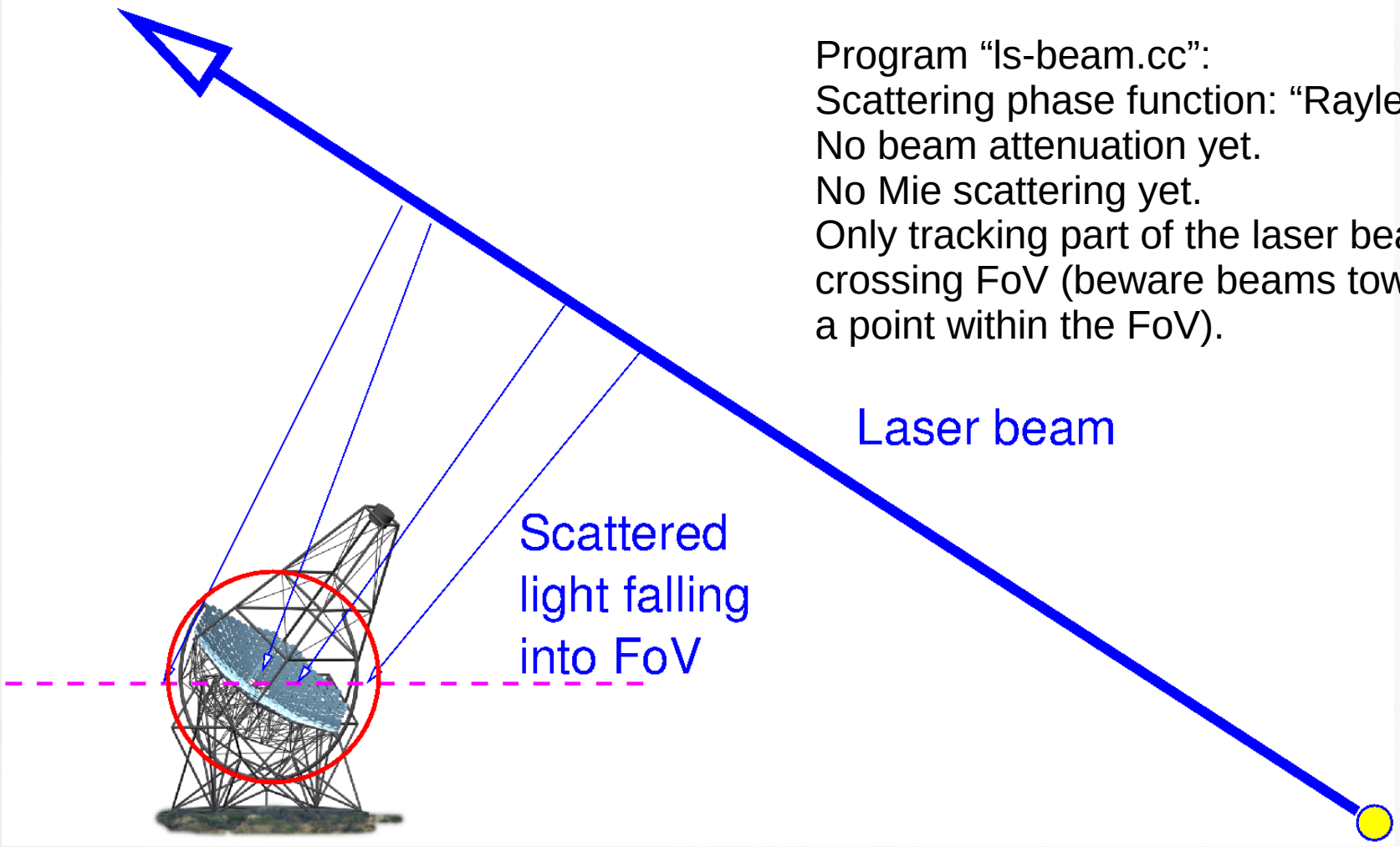




GCT flat-fielding unit

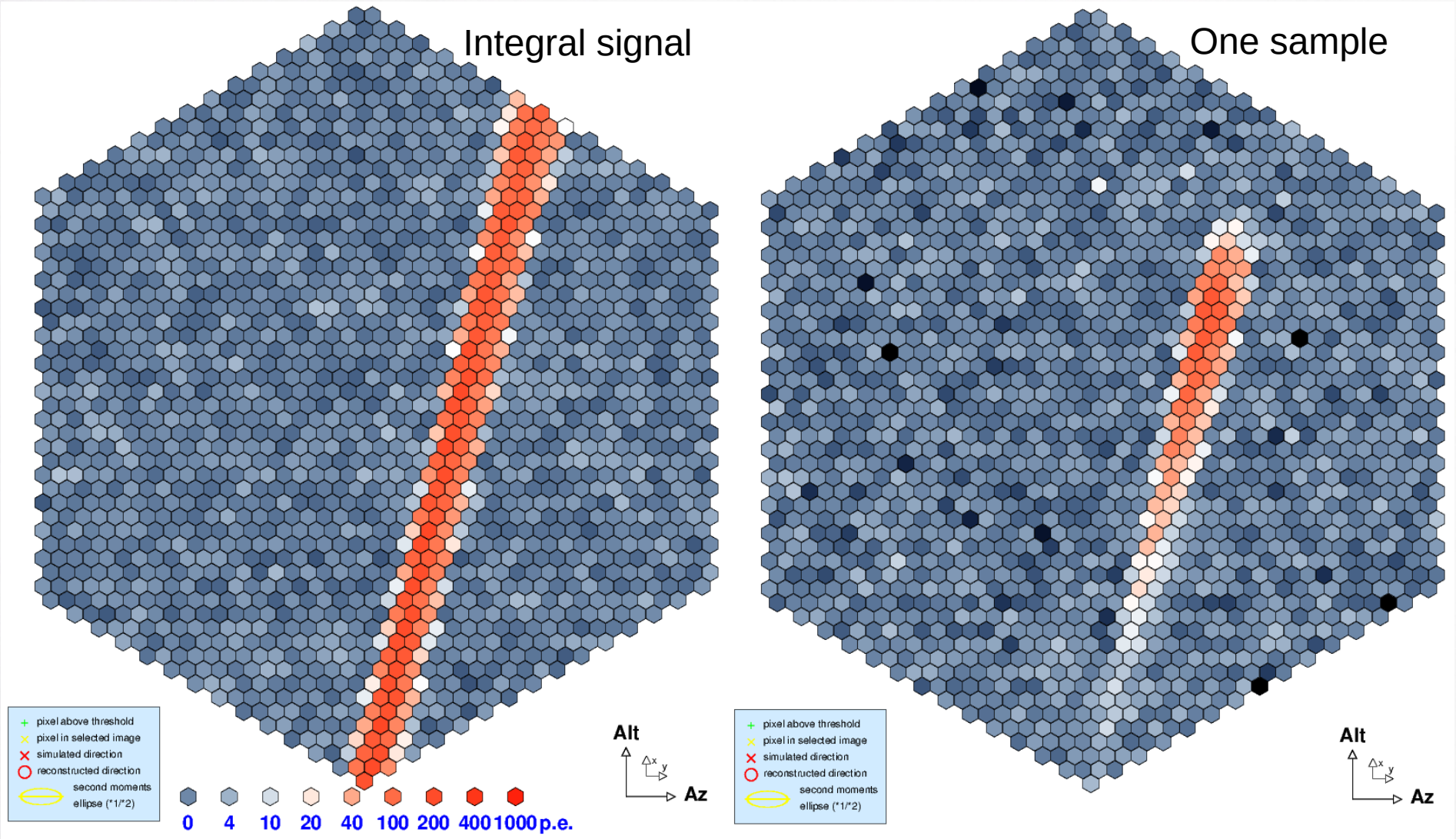


Laser beam crossing FoV



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



Where and how ?

- The LightEmission code is available as part of sim_telarray (sub-directory LightEmission) as well as a stand-alone package (LightEmission.tar.gz), including relevant IACT and hessio code.
  - Stand-alone: from the IACT/ATMO package site https://www.mpi-hd.mpg.de/hfm/~bernlohr/iact-atmo/
  - Includes unrealistic demos (octo.cc, fpls.cc) and useful applications (ff-dc.cc, ff-gct.cc, ls-beam.cc).
- Building them all (either way): make
- Feedback (and bug reports) welcome!

Status and outlook

- Implementation of artificial light sources writing photon bunches through IACT/atmo package like CORSIKA is working and in good shape.
- Bypassing optics raytracing for single reflector (and flat camera) was trivial.
- Bypassing optics with dual mirror telescope needed a bit more time but is working well now.
- Useful also for other telescope/camera simulation codes? (Feedback, please!)