Reconstruction

Where does your science data come from?

Karl Kosack
CEA Saclay

Atelier CTA France Analyse, Observatiore de Meudon, 2 Oct 2017

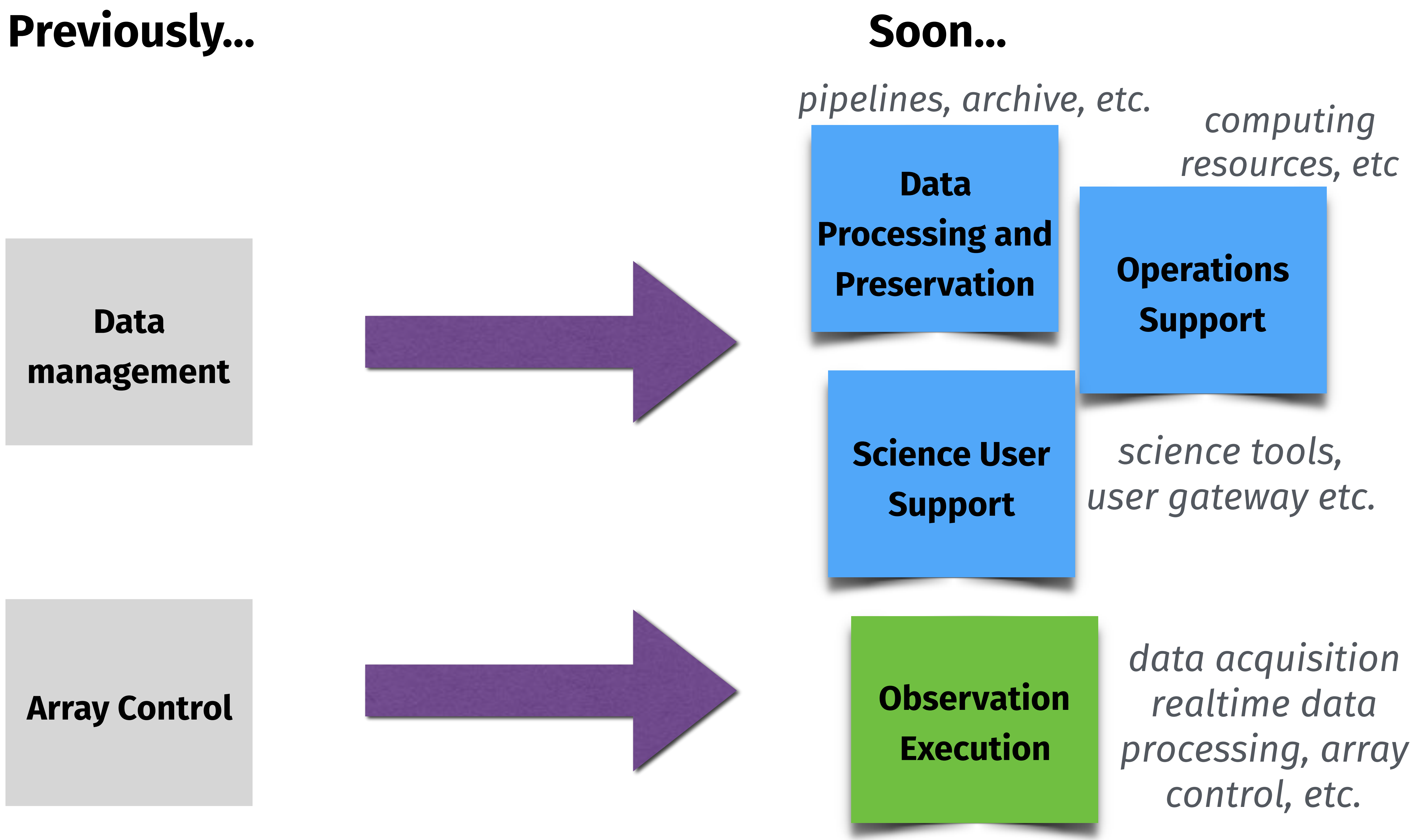
Goals of CTA Data Processing

- Record and maintain all CTA data and related software **over 30+ years**
- Make efficient use of use **modern computing systems** (grids, clouds, GPUs, etc.)
- Maintain ability to **re-process old data**
  - apply newer techniques, better calibration
  - check old results
  - versioned software and related support files
- Generate **common data products** for science users
  - Event Lists
  - Instrumental Response Functions (IRFs)
  - Associated Technical data (pointing, weather, etc.)
- provide **robust, well-understood science results**

Goals of CTA Data Processing

- Record and maintain all CTA data and related software **over 30+ years**
- Make efficient use of use **modern computing systems** (grids, clouds, GPUs, etc.)
- Maintain ability to **re-process old data**
  - apply newer techniques, better calibration
  - check old results
  - versioned software and related support files
- Generate **common data products** for science users
  - Event Lists
  - Instrumental Response Functions (IRFs)
  - Associated Technical data (pointing, weather, etc.)
- provide **robust, well-understood science results**

Which Working Packages?



gamma-ray detection in context

keV ***X-ray***	*MeV* ***γ-ray***	*GeV* ***γ-ray***	*TeV* ***γ-ray***

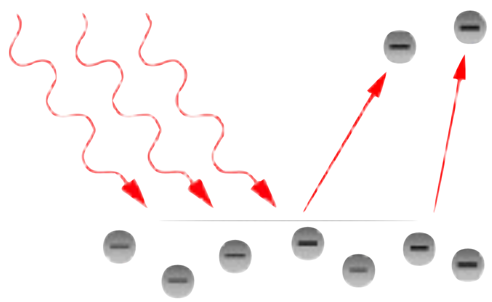
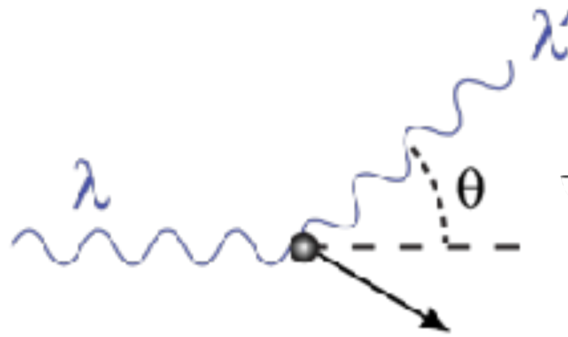


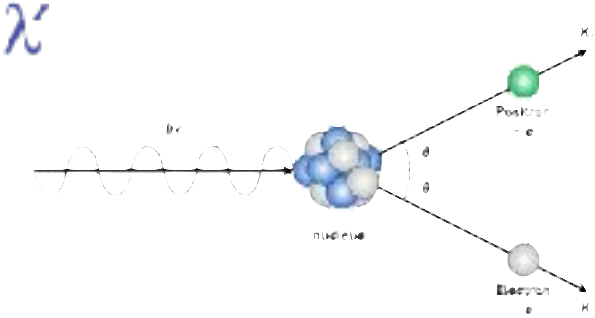
photo-electric effect

400 cm² @ 5keV



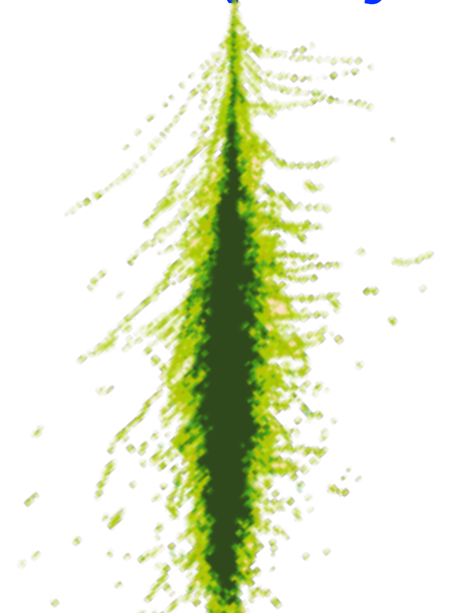
Compton Effect

50 cm² @ 5 MeV



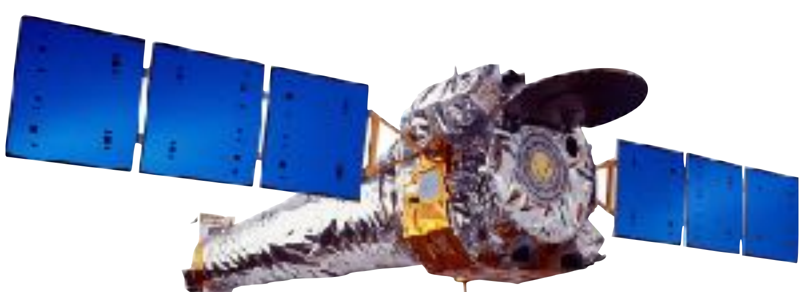
Pair Conversion

1m² @ 1 GeV

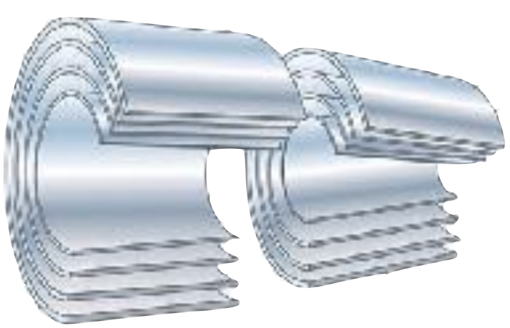


Extensive Air Shower

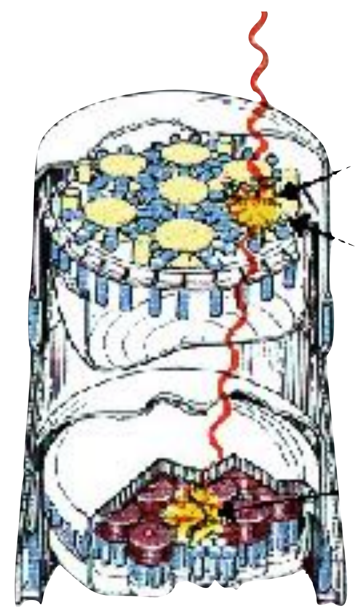
> 10⁵ - 10⁶ m² @ 1 TeV



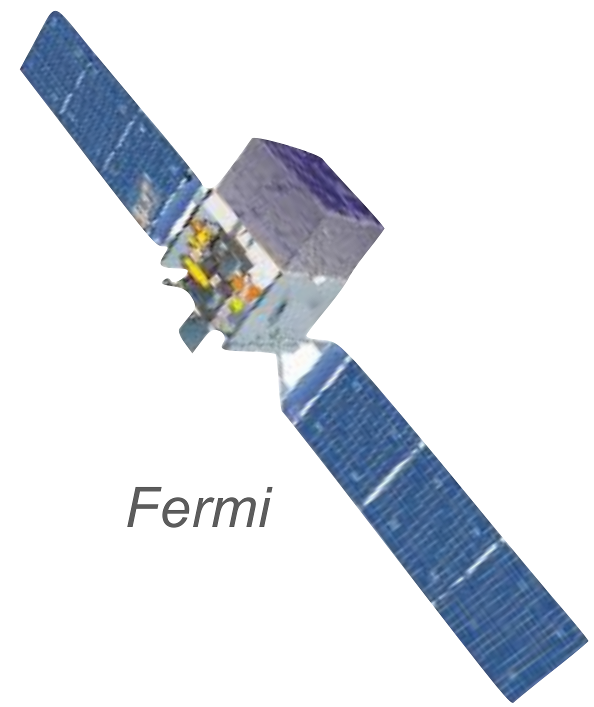
Chandra, XMM



grazing-incidence mirror



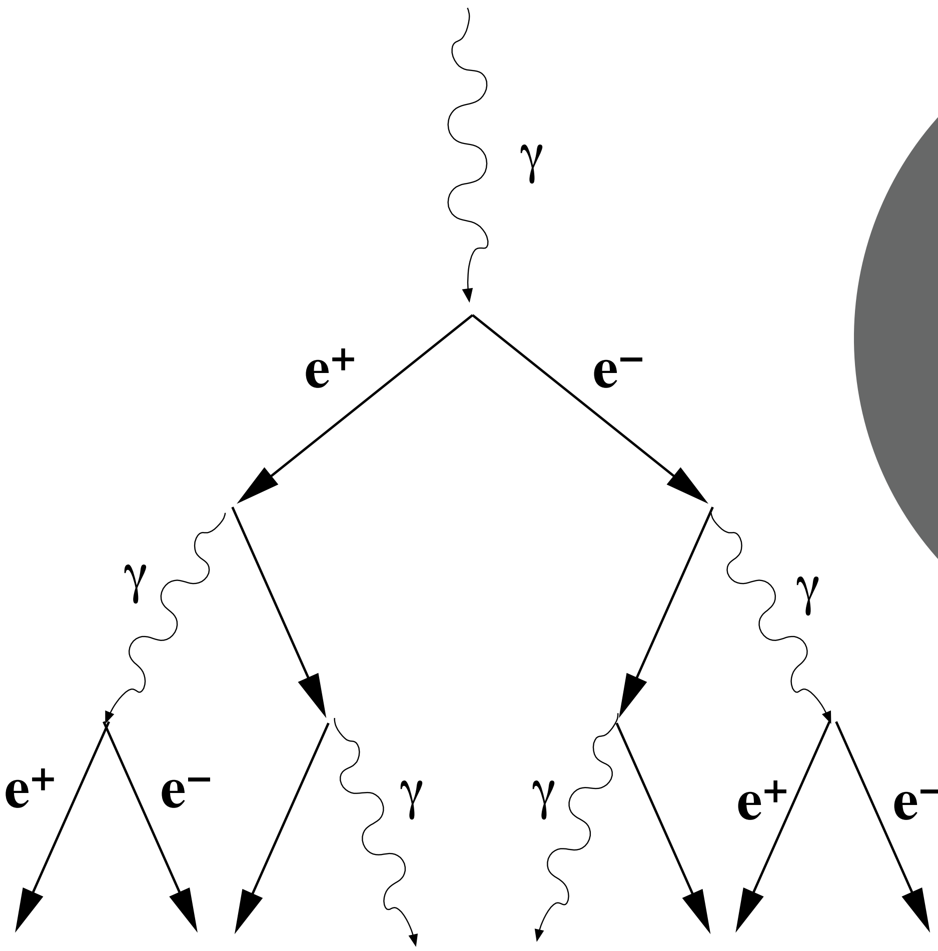
Comptel

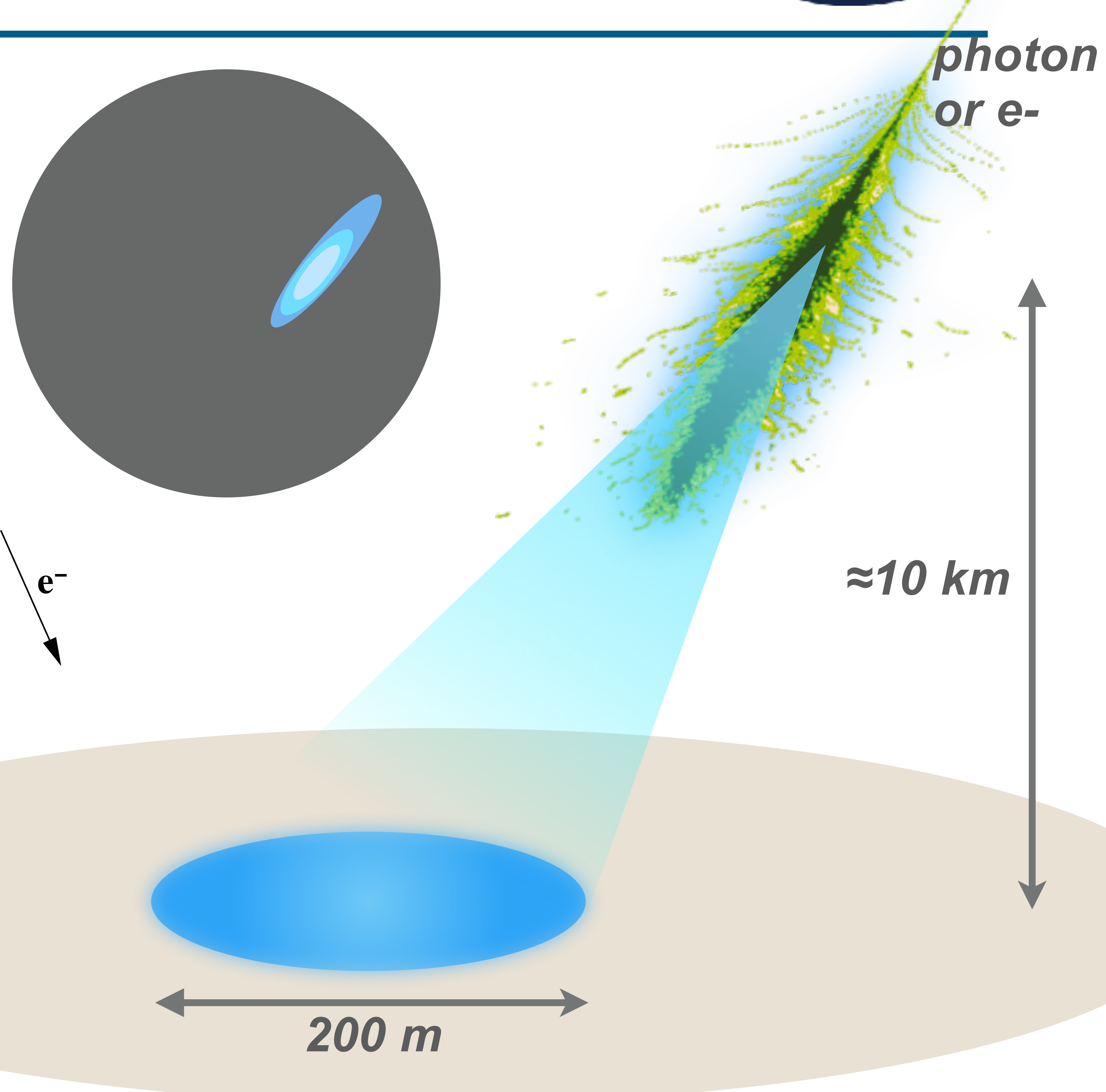


Fermi

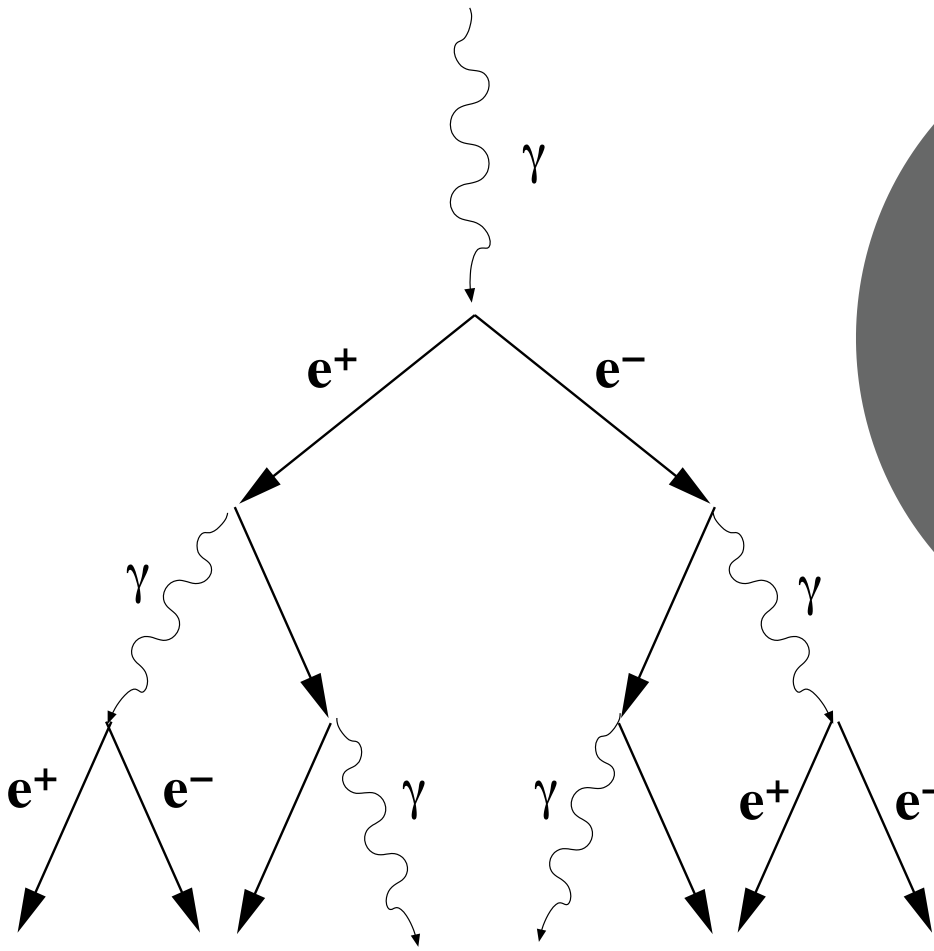
Whipple 10m

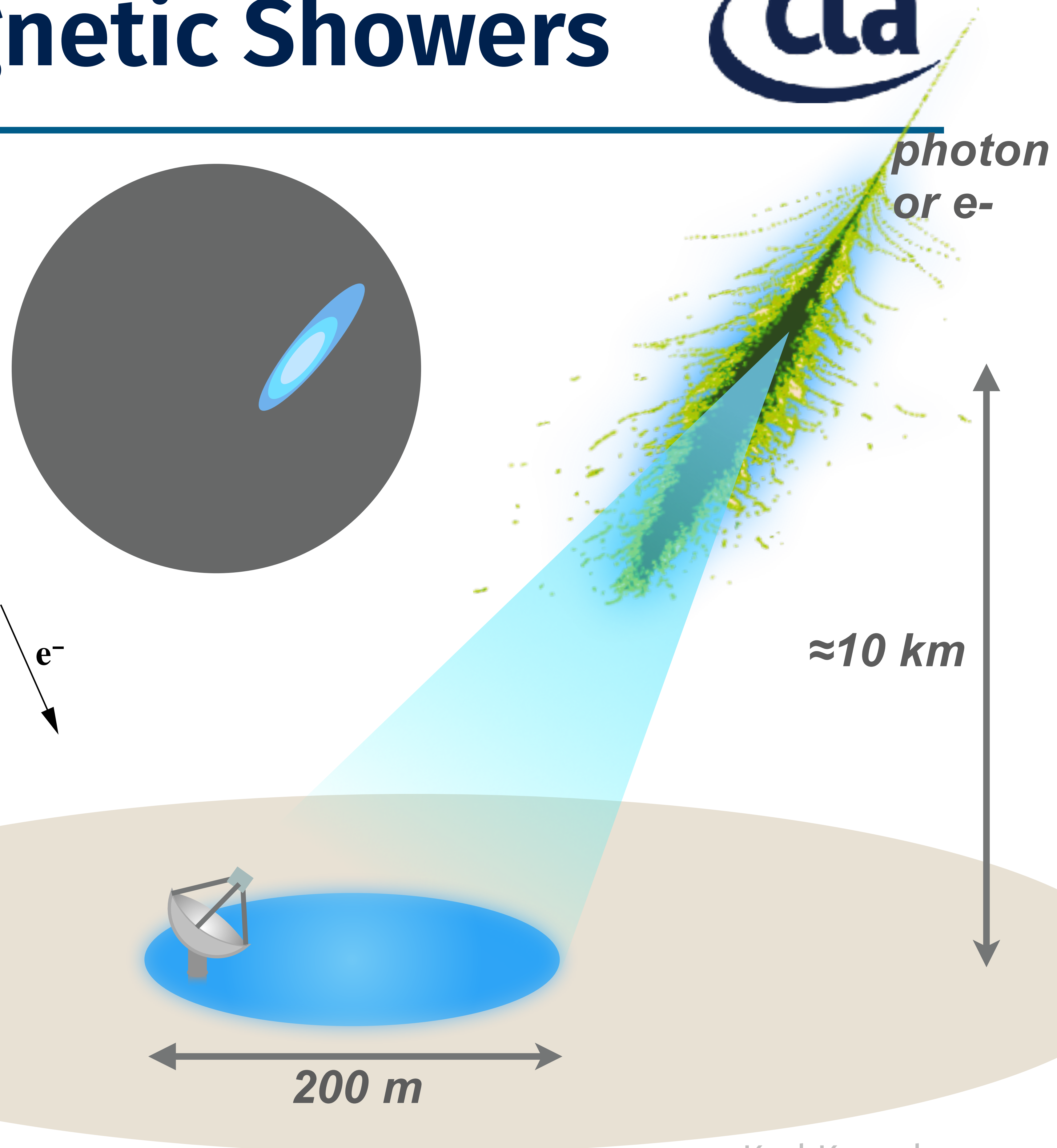
Electromagnetic Showers



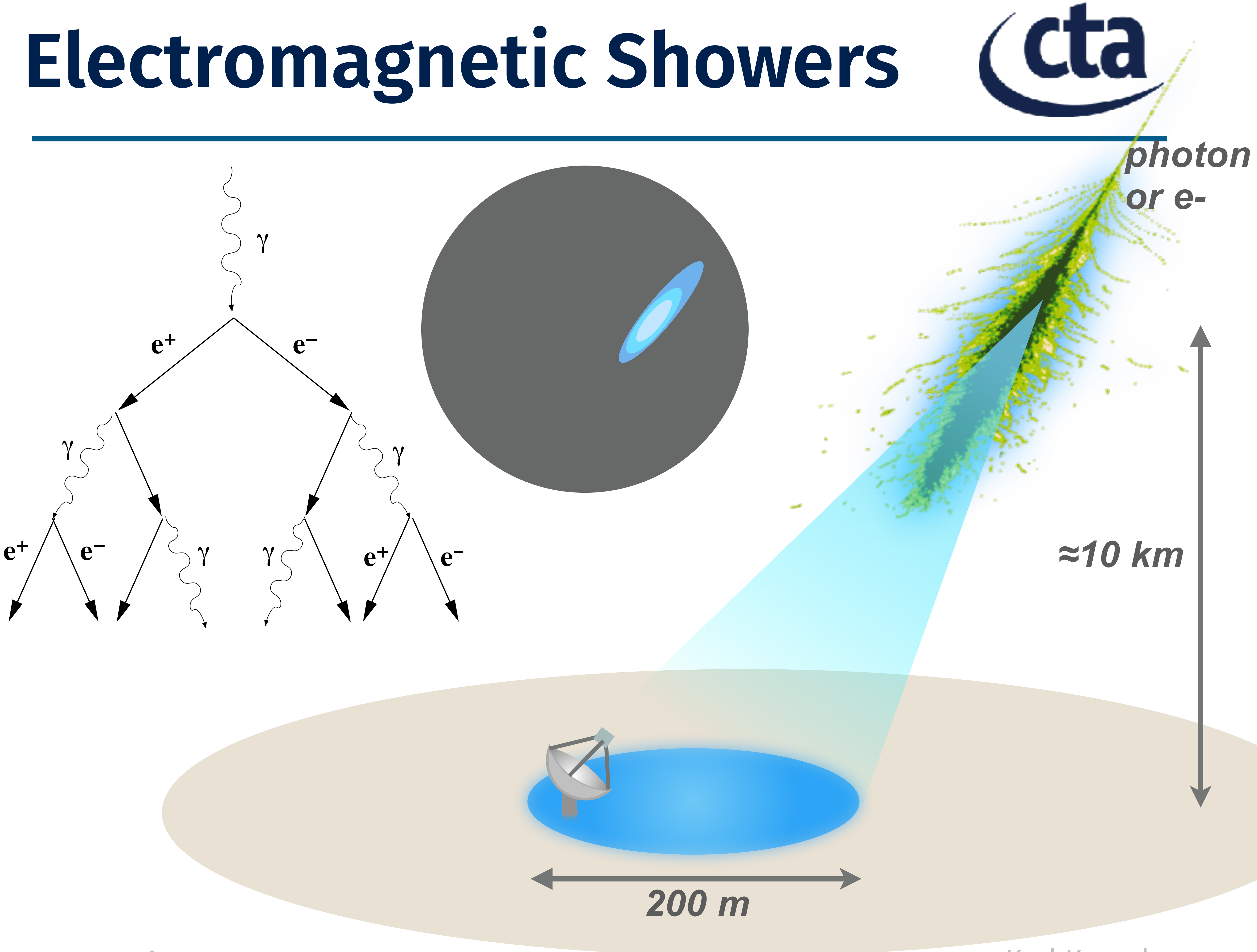


Electromagnetic Showers

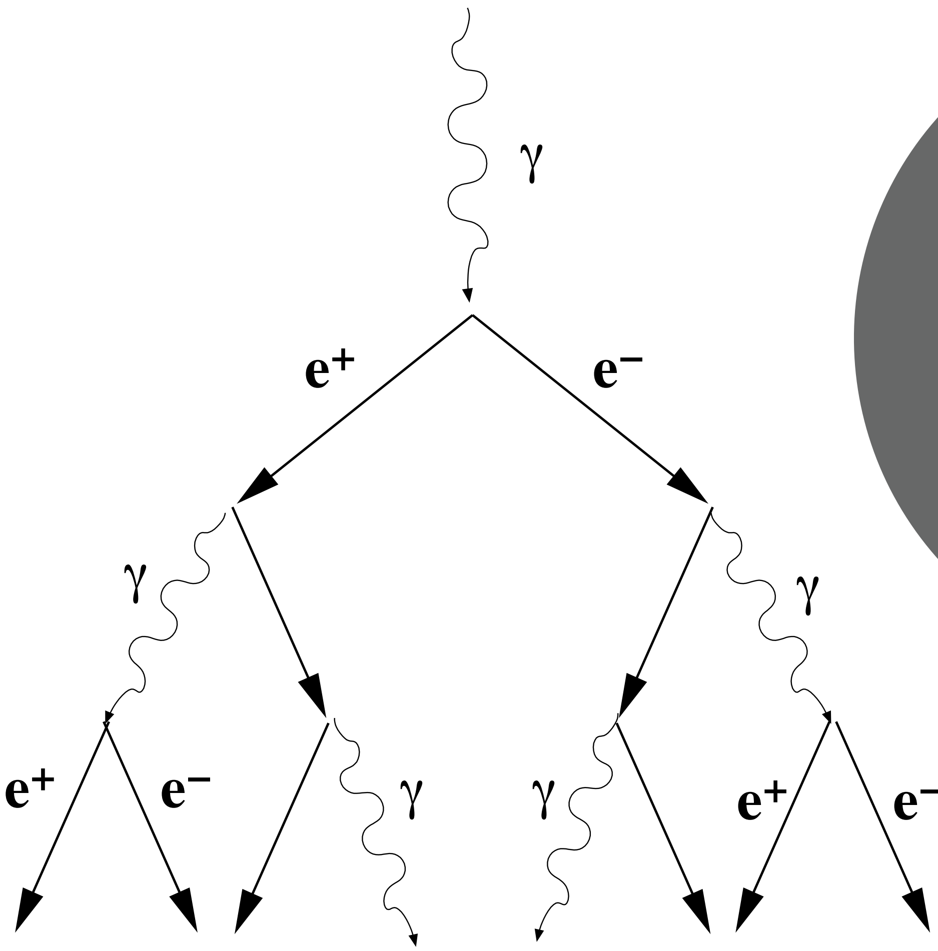


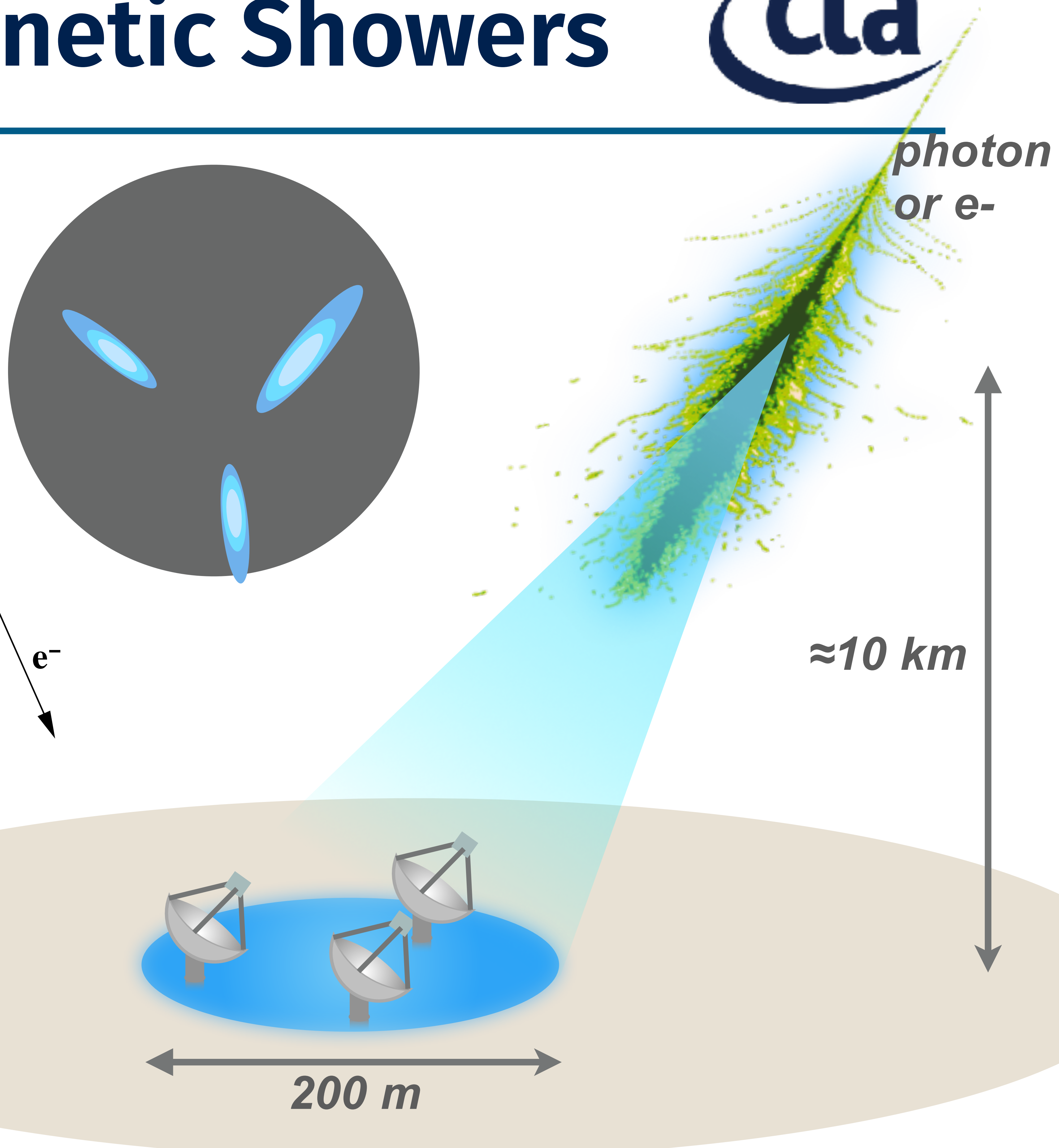


Electromagnetic Showers

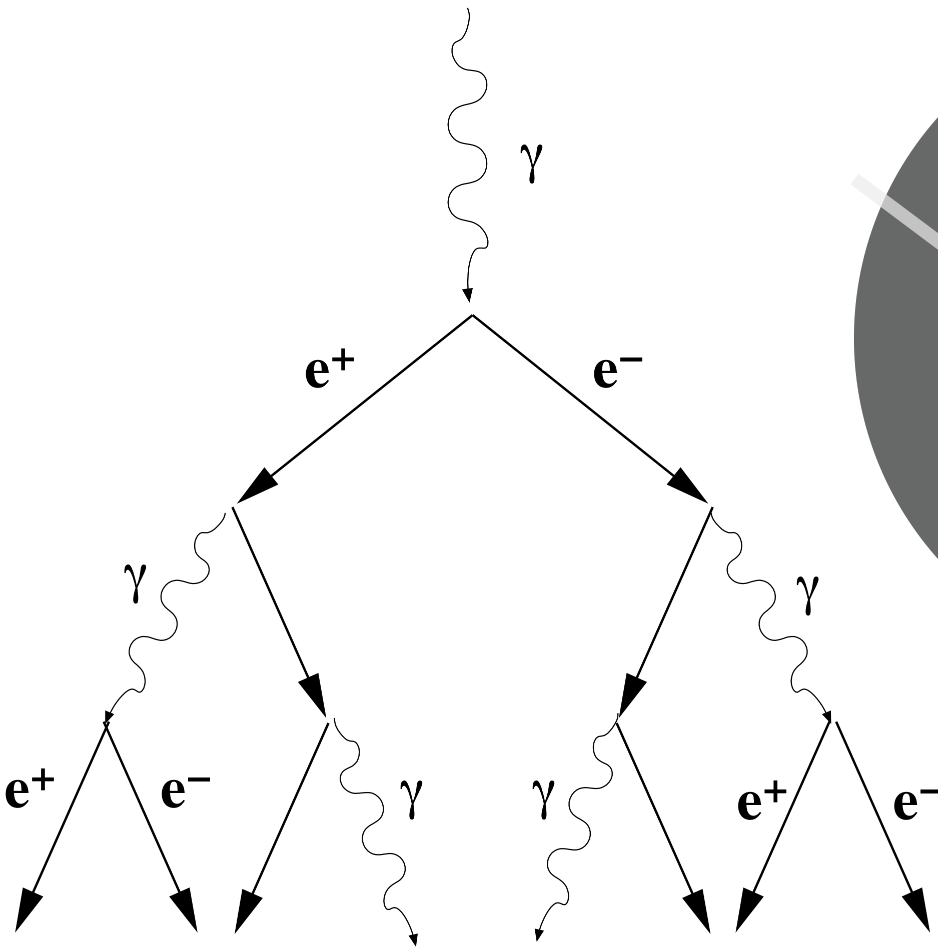


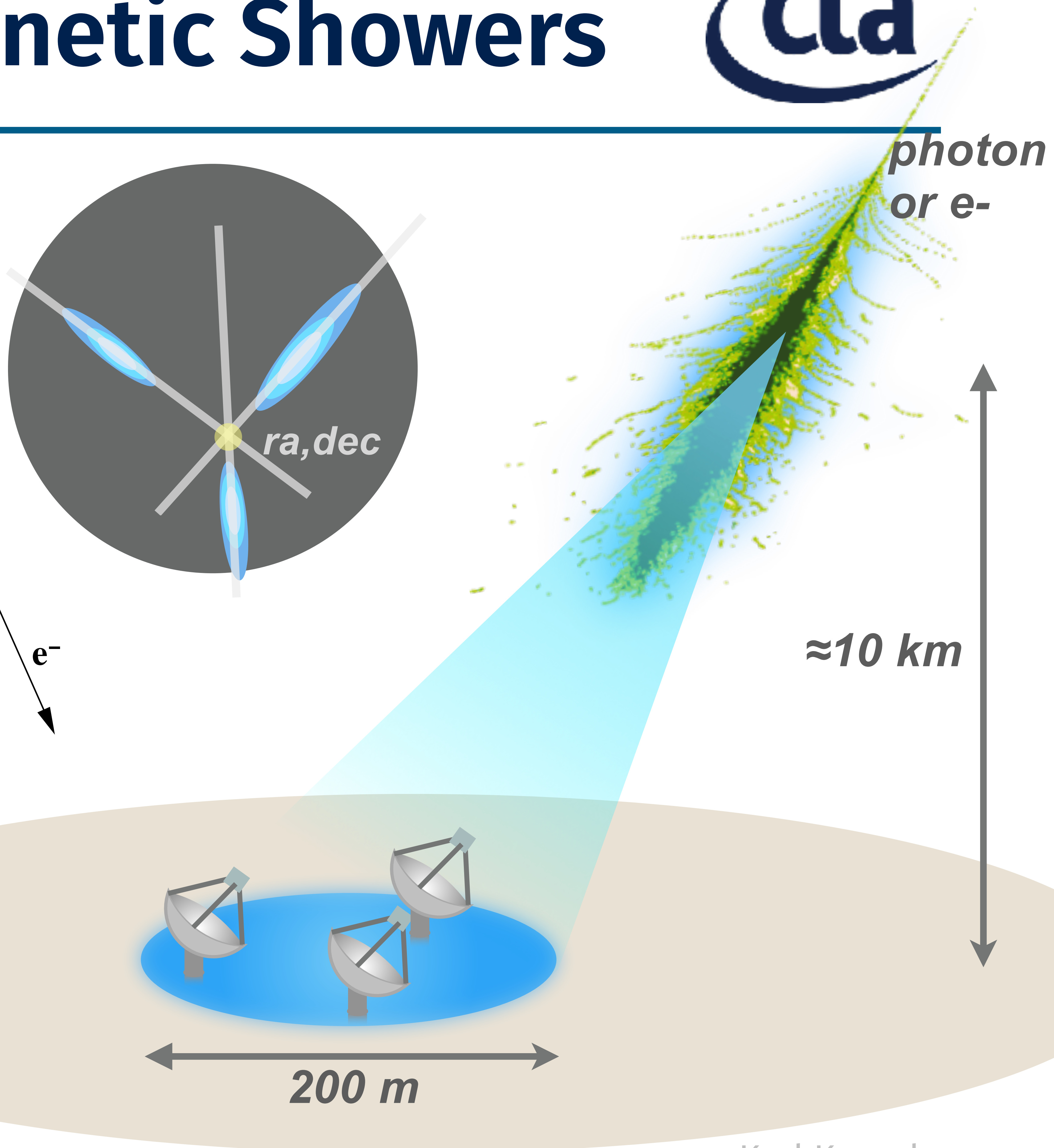
Electromagnetic Showers



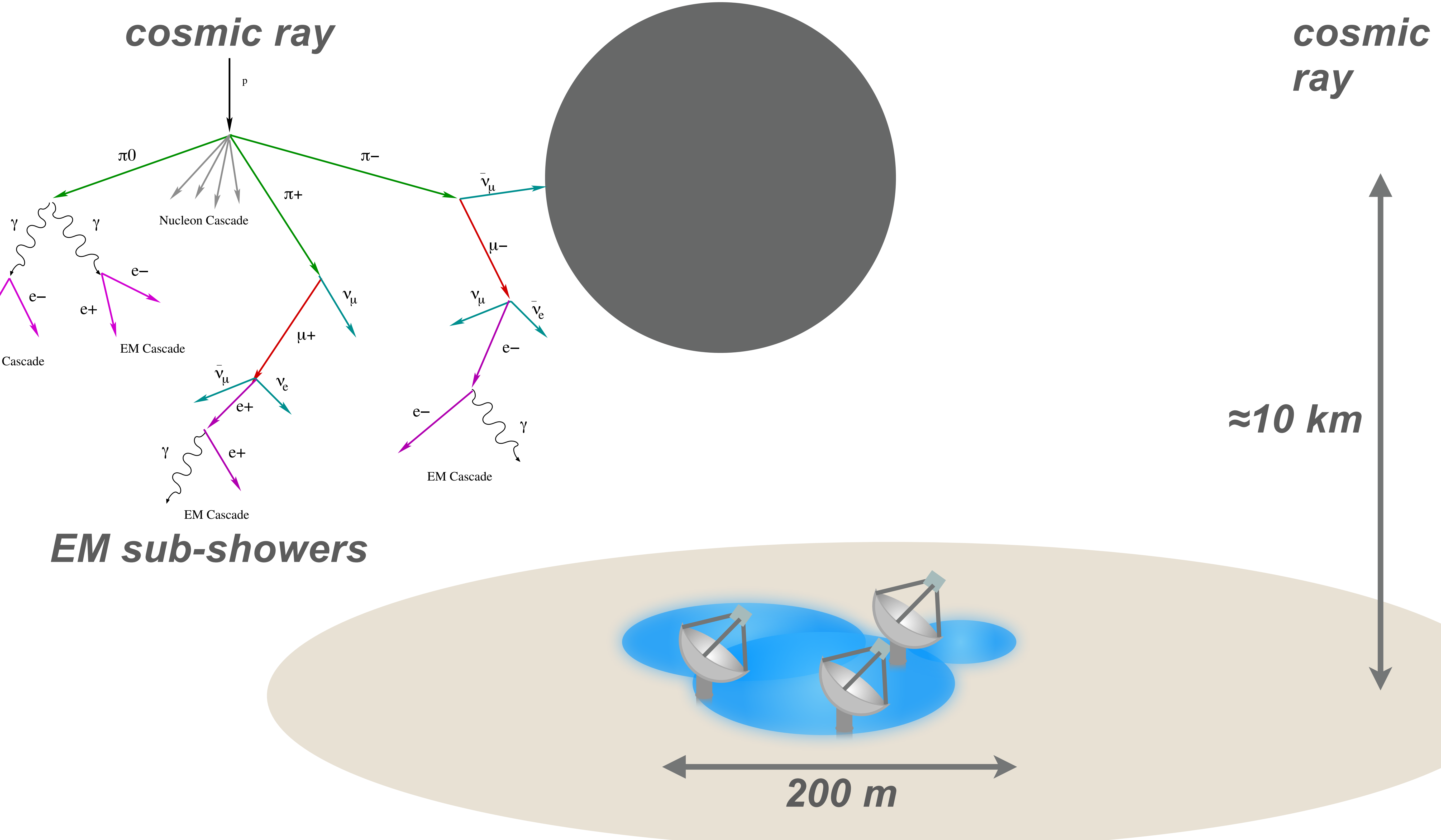


Electromagnetic Showers



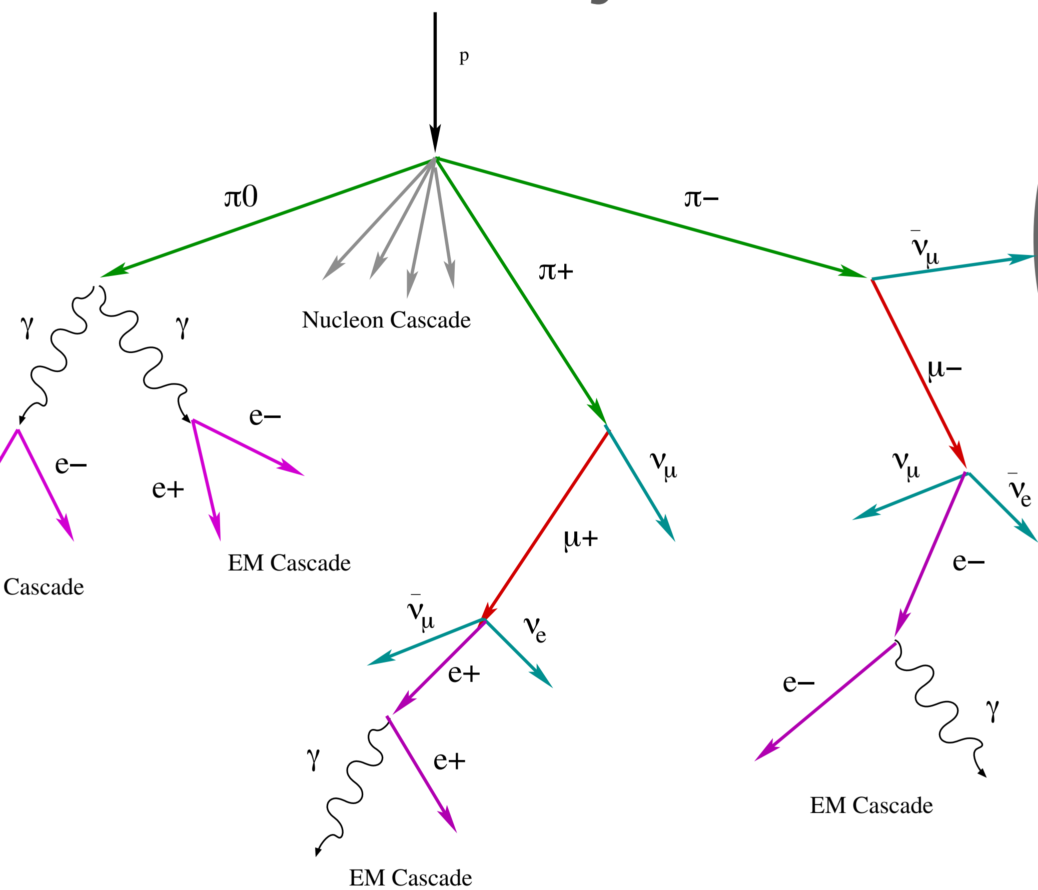


Hadronic Showers

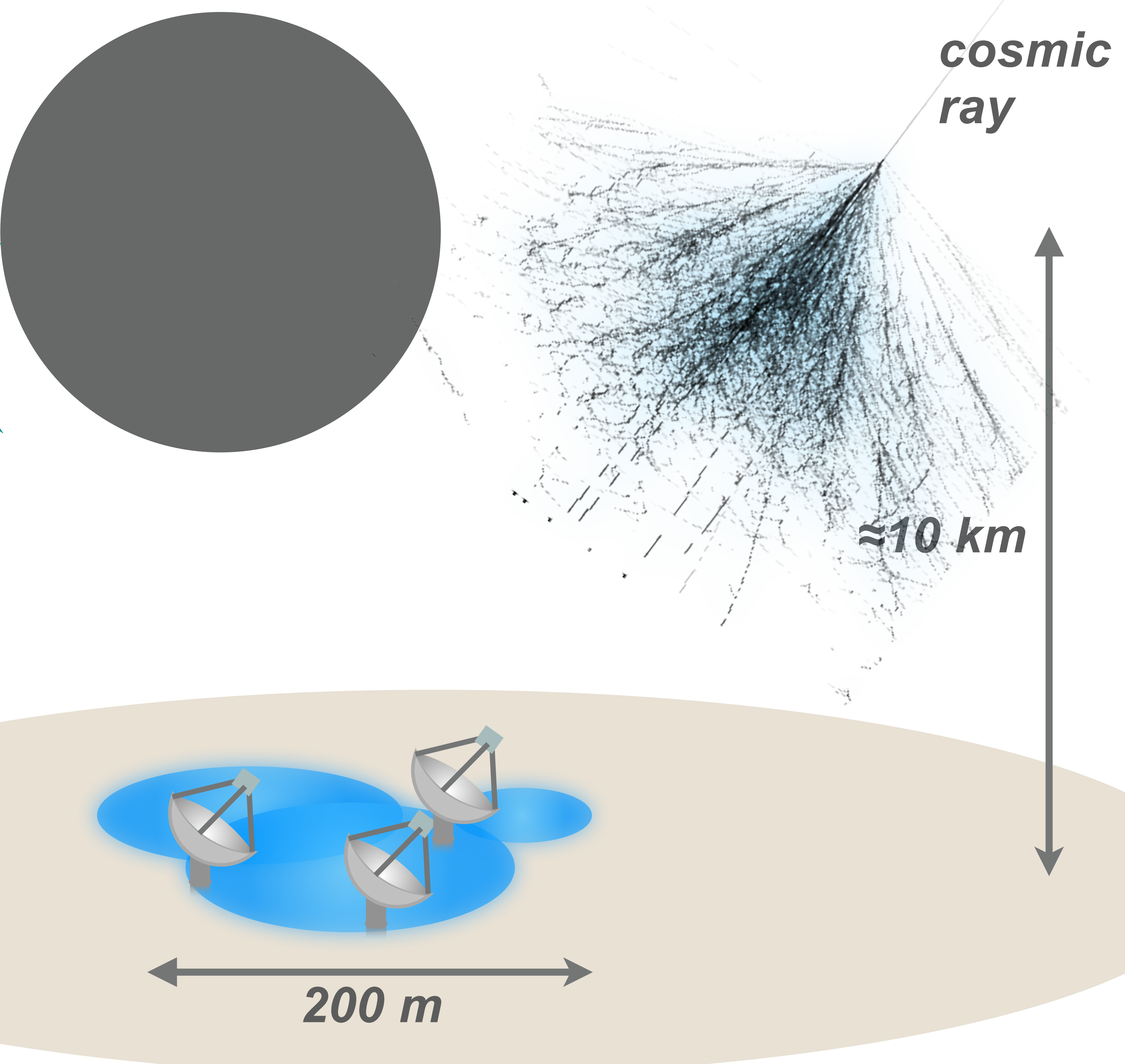


Hadronic Showers

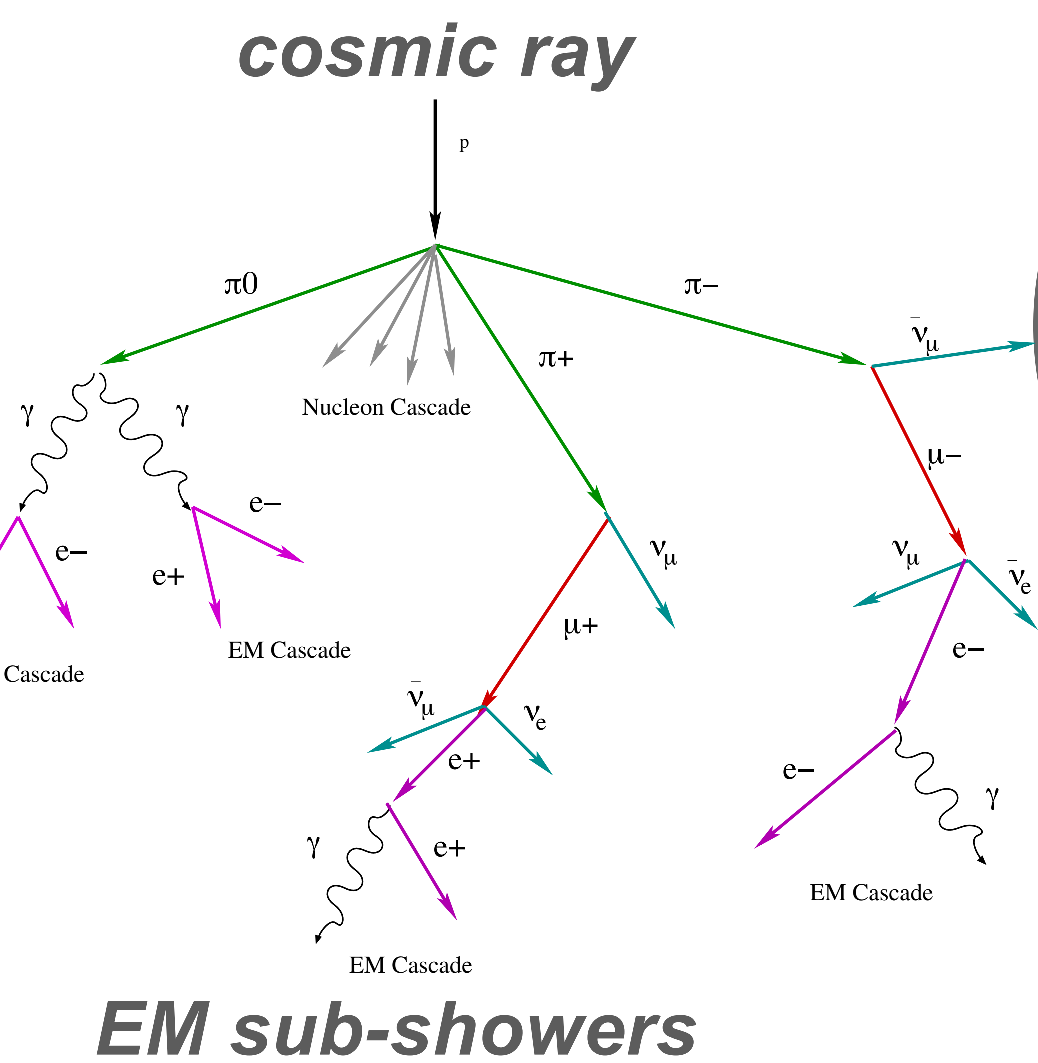
cosmic ray

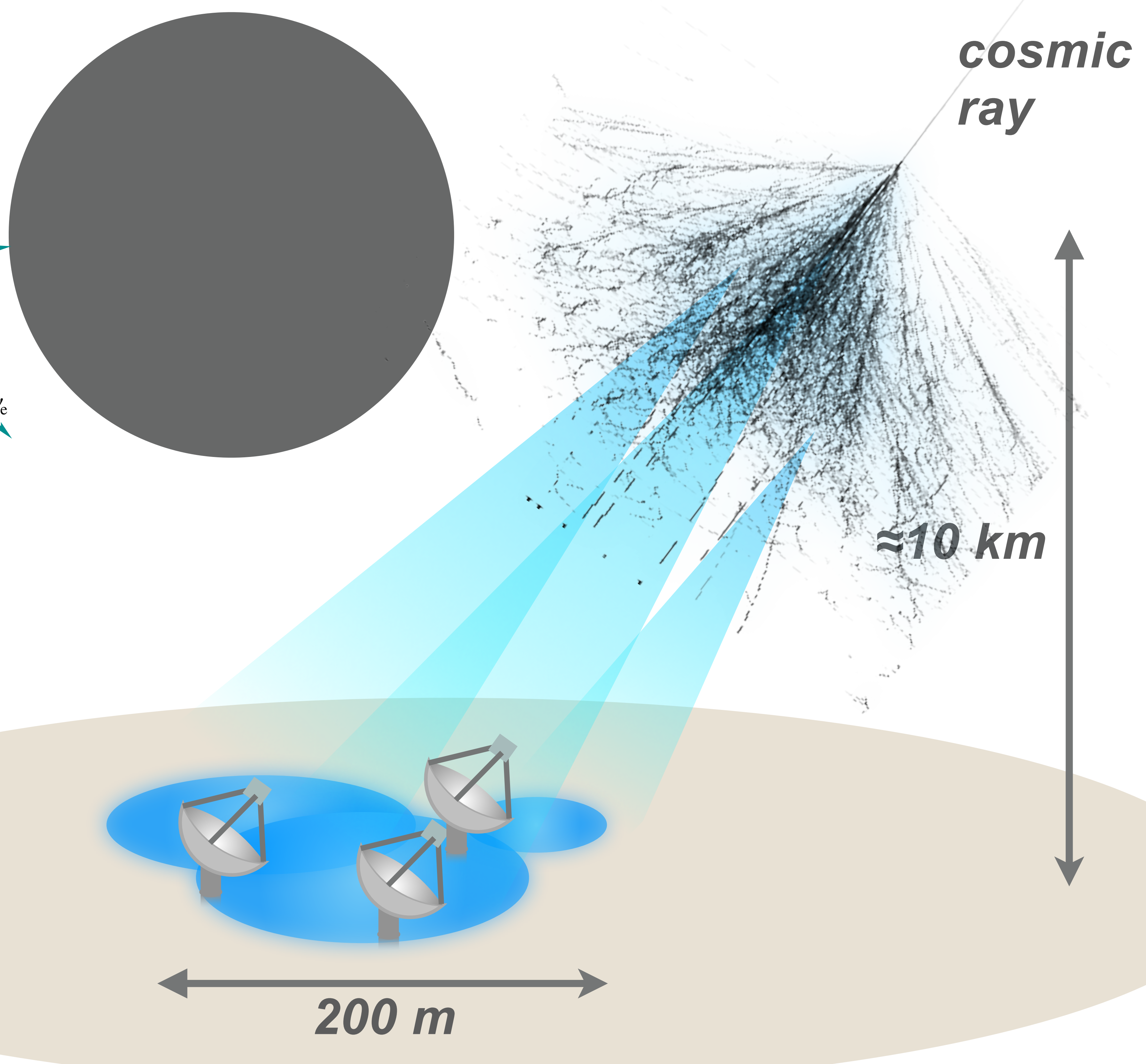


EM sub-showers

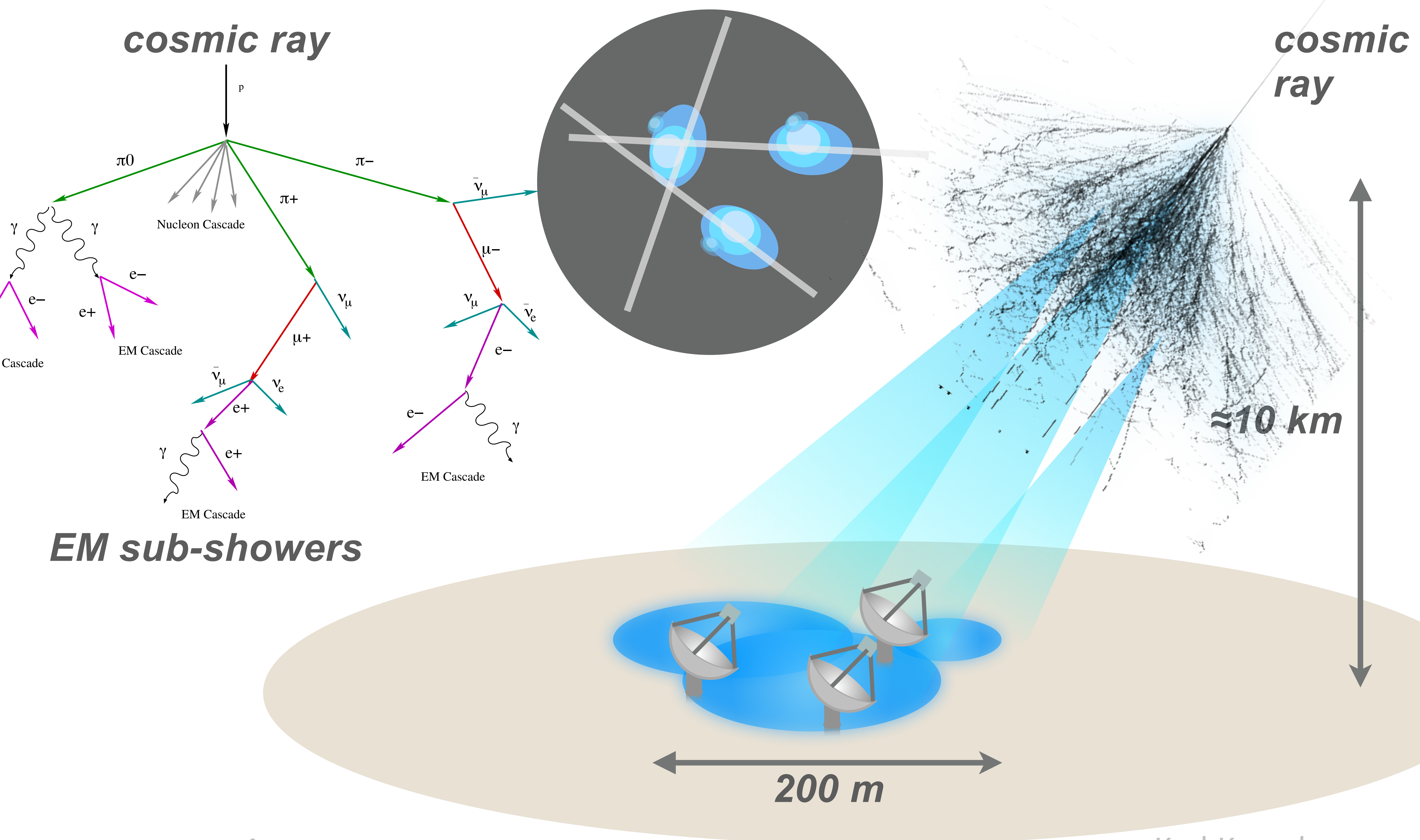


Hadronic Showers

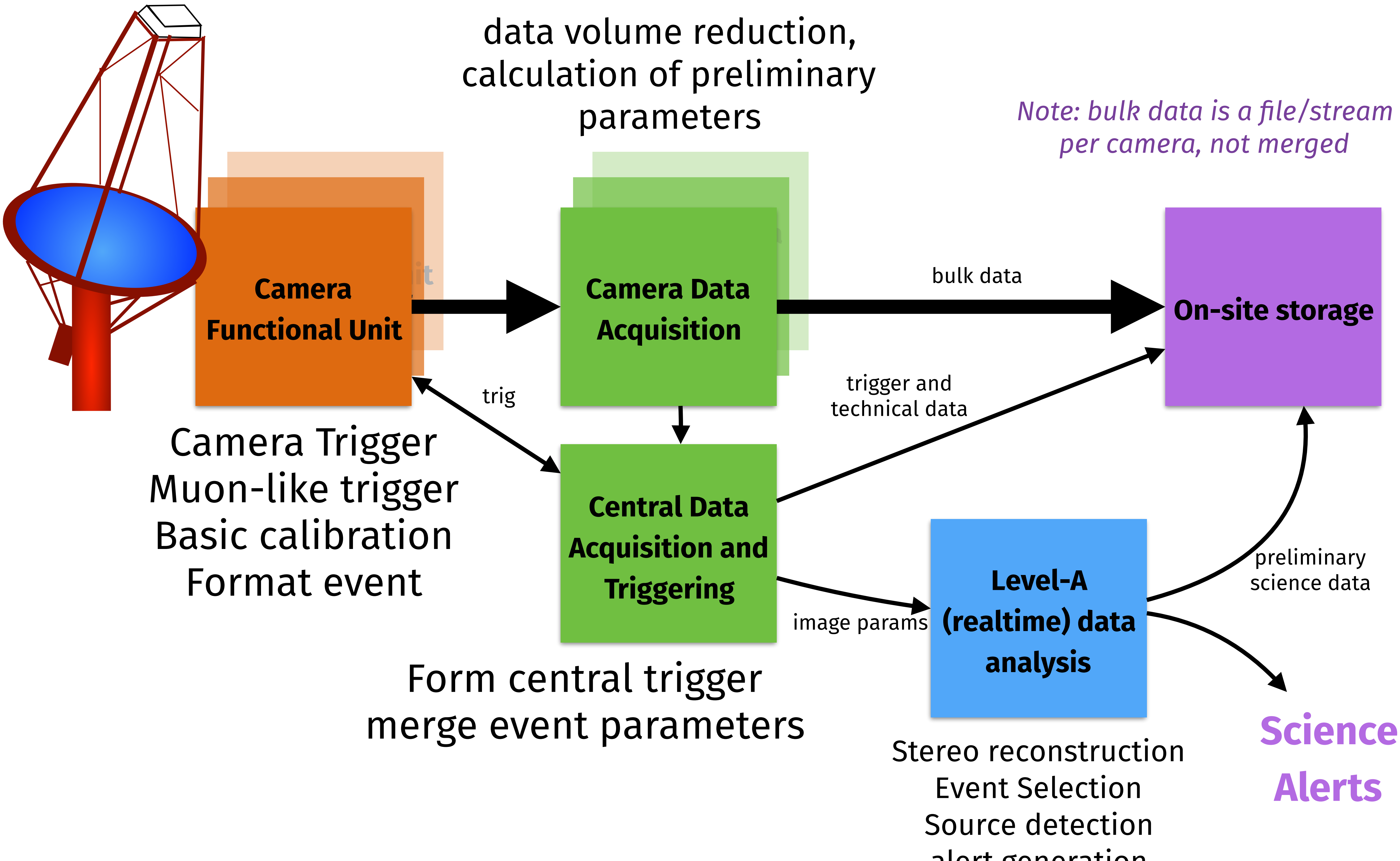




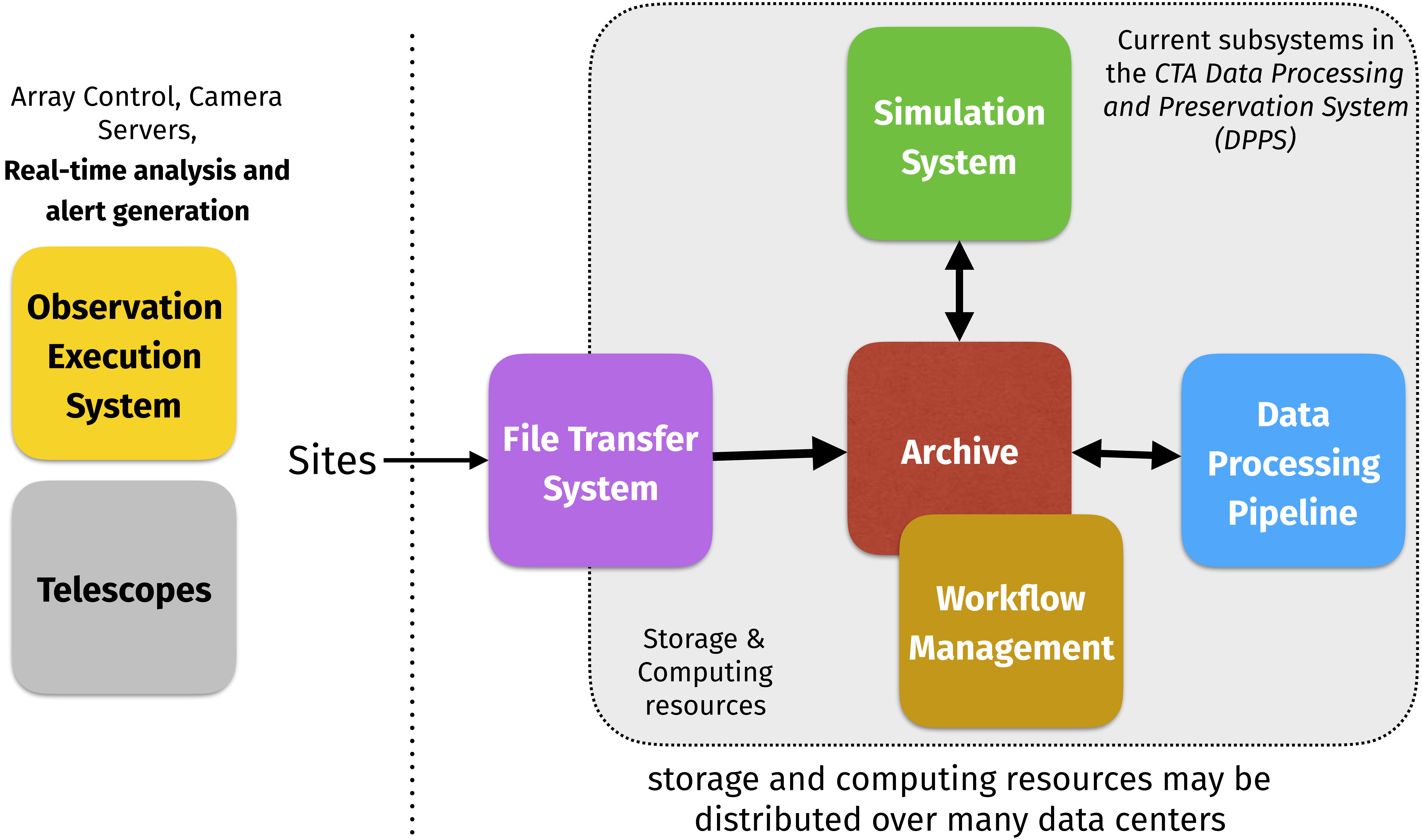
Hadronic Showers



On-site Data Path (simplified)



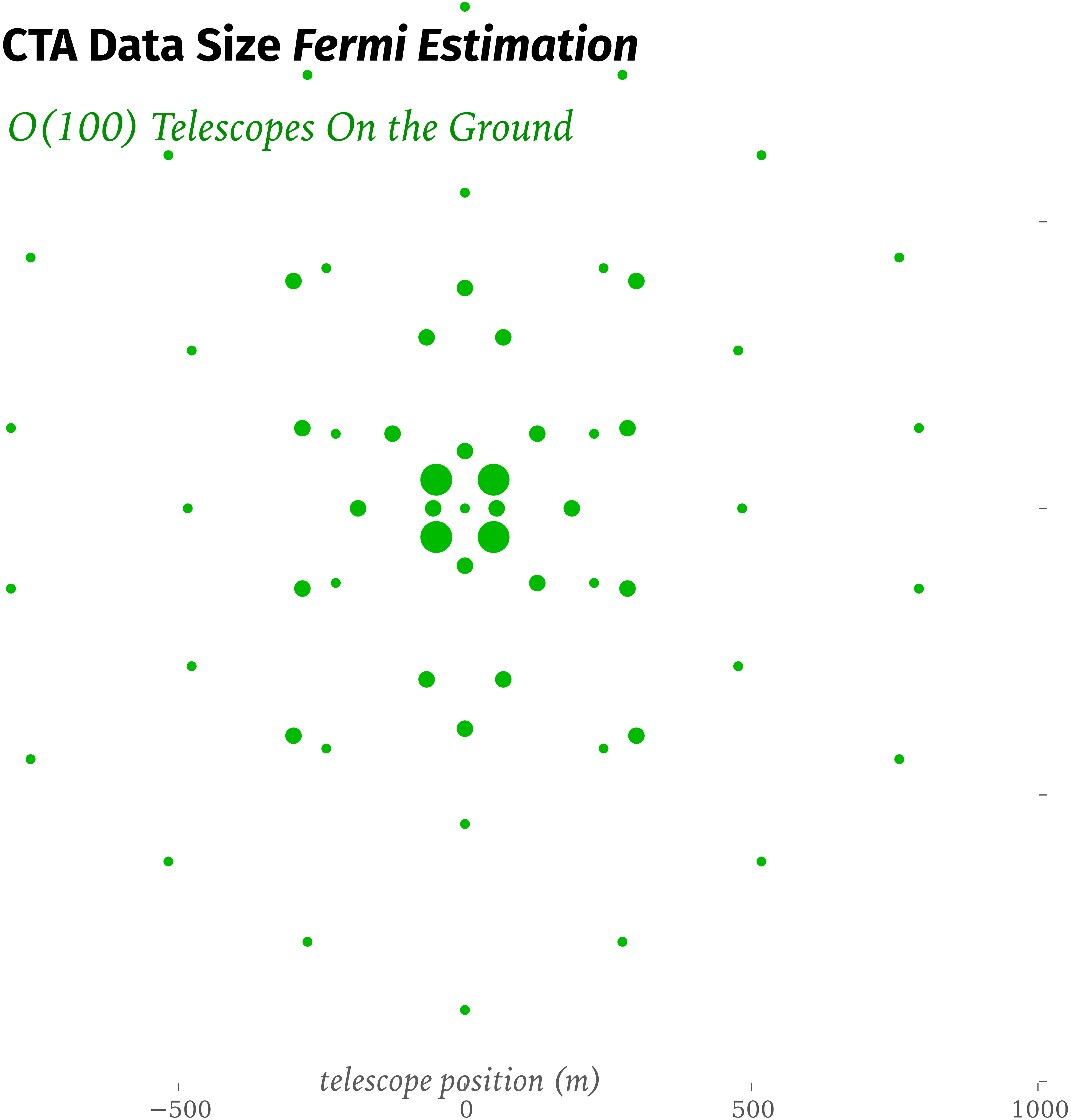
Off-site data processing (simplified)



Challenge: **Raw Data Volume**

CTA Data Size *Fermi Estimation*

O(100) Telescopes On the Ground

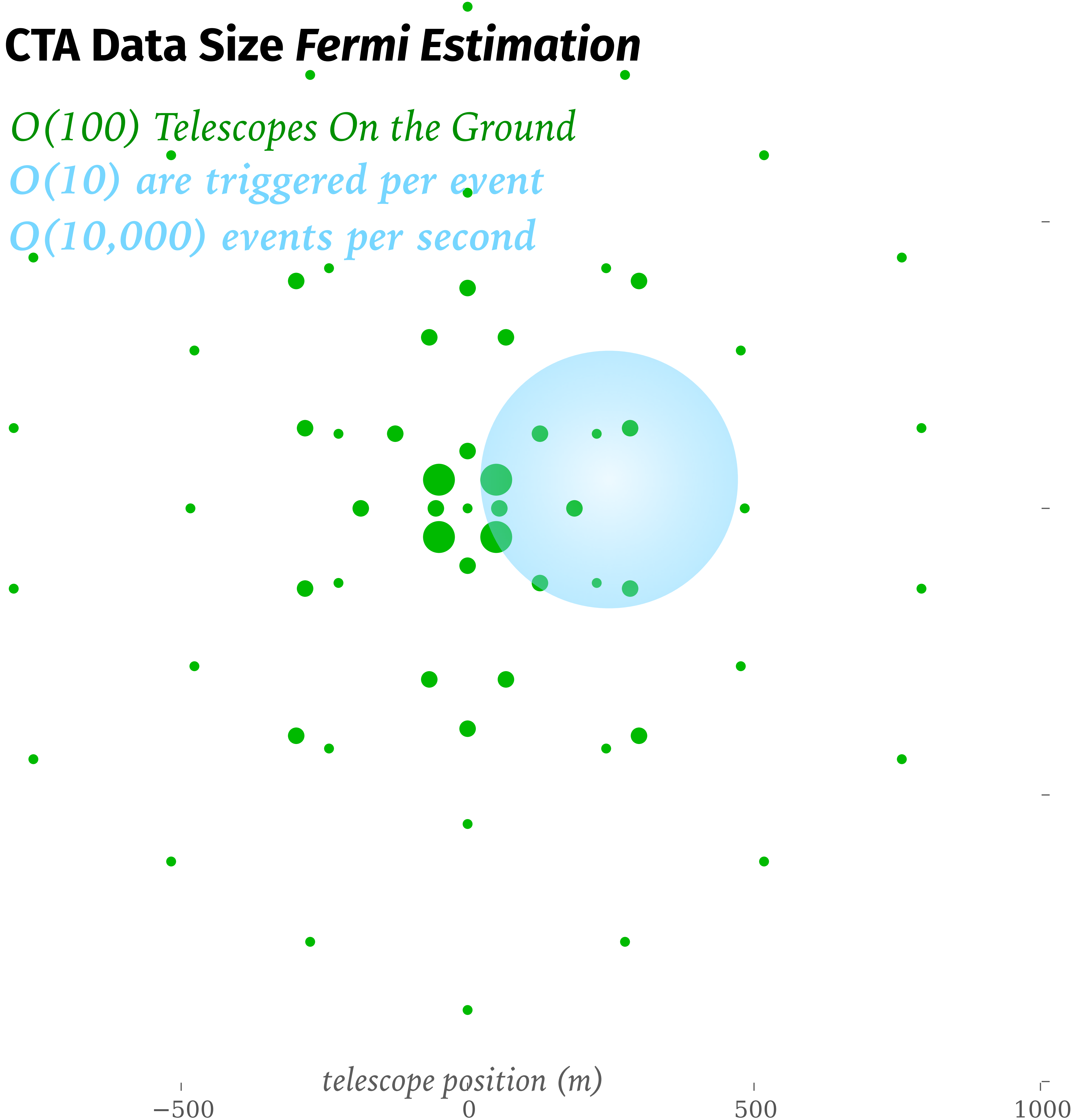


CTA Data Size *Fermi Estimation*

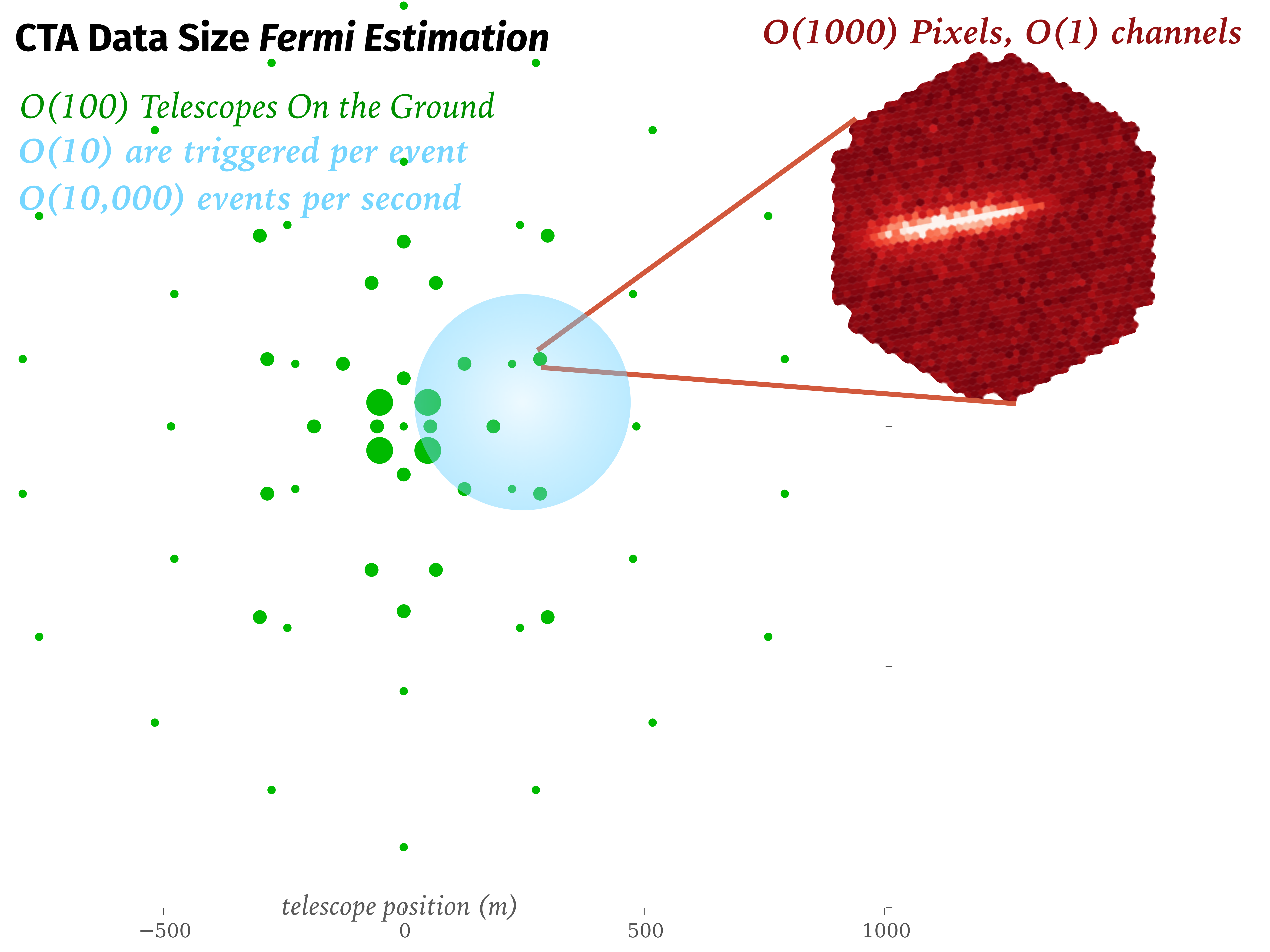
O(100) Telescopes On the Ground

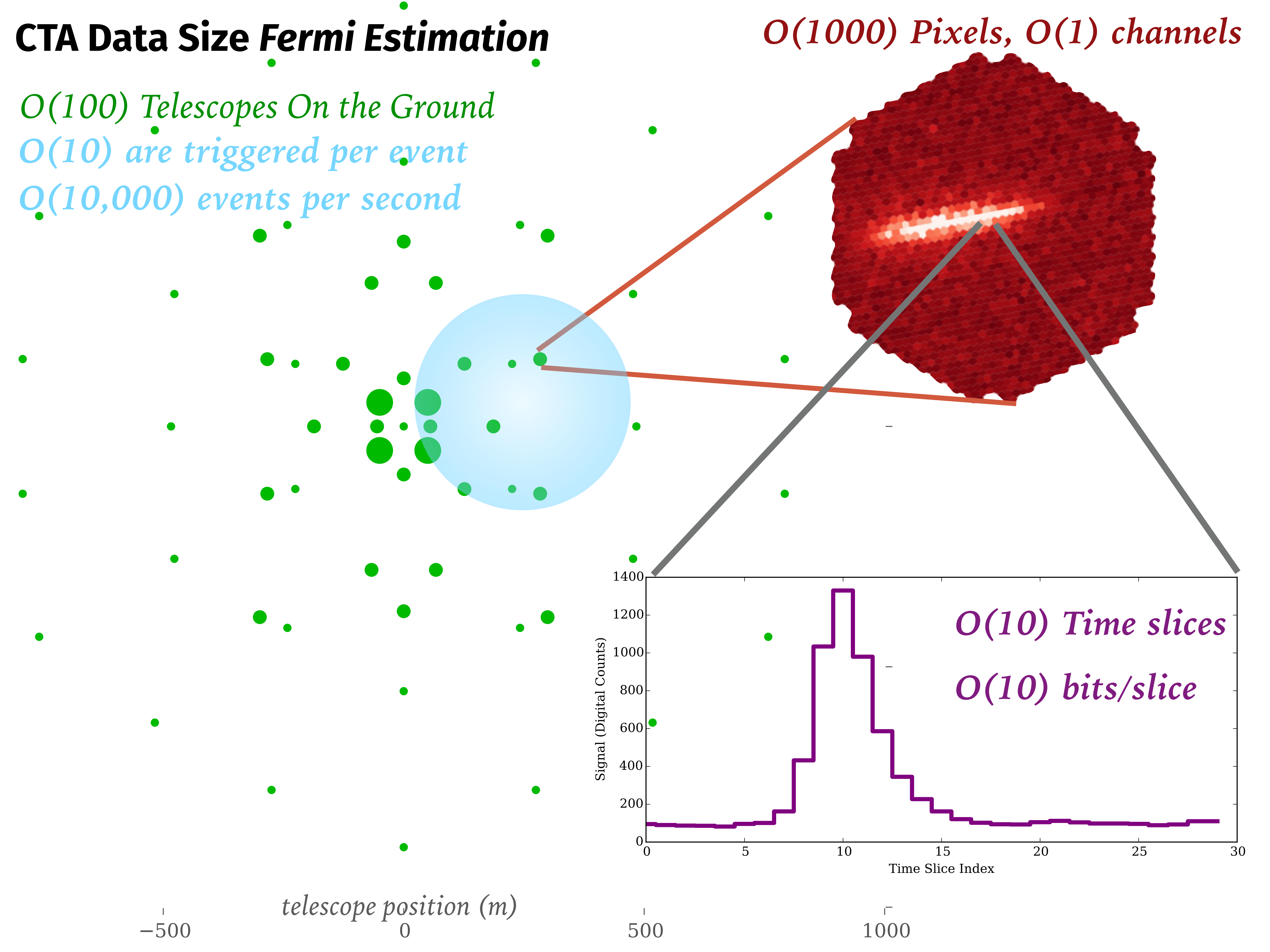
O(10) are triggered per event

O(10,000) events per second

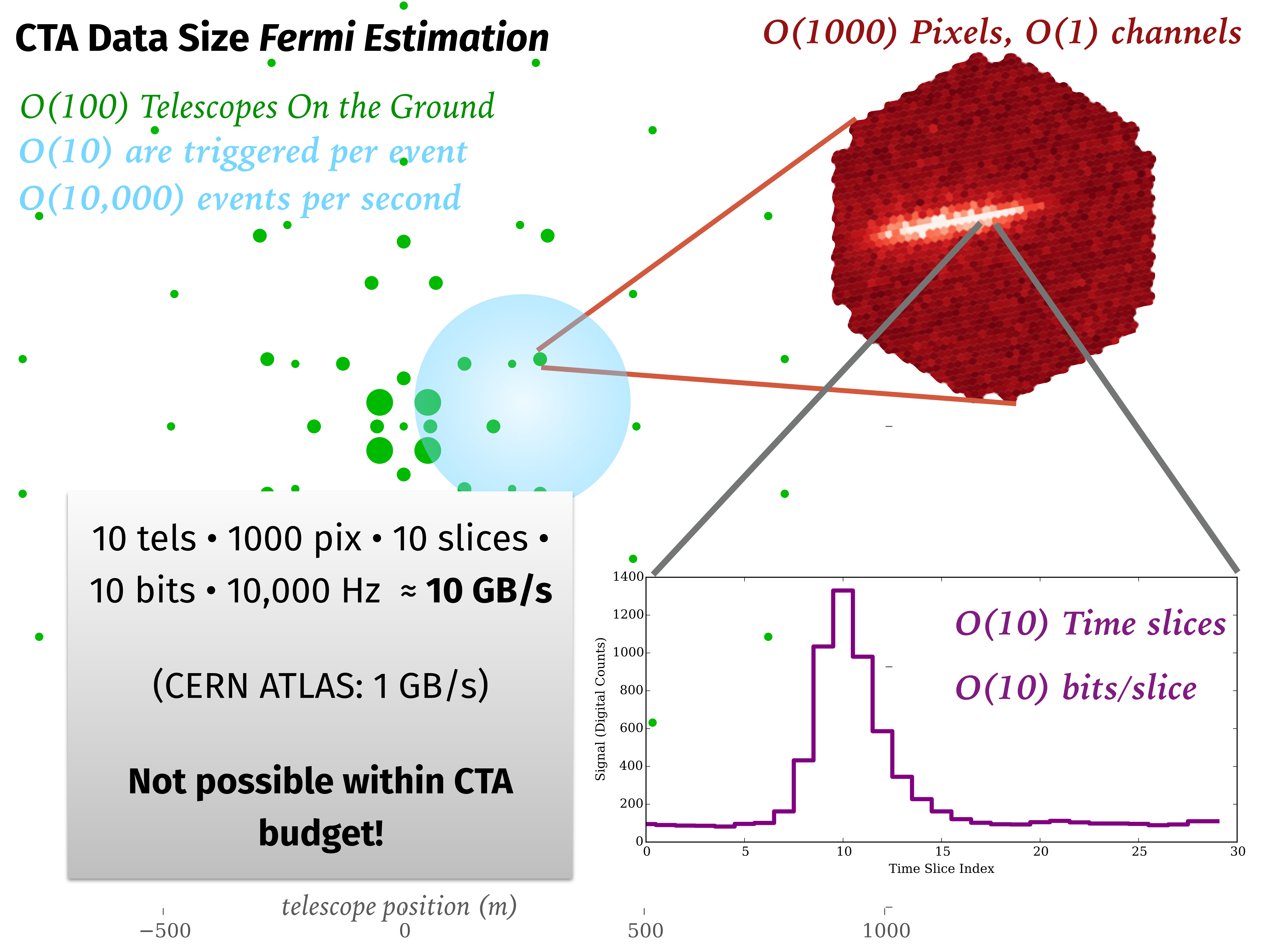


CTA Data Size *Fermi Estimation*




CTA Data Size *Fermi Estimation*
O(1000) Pixels, O(1) channels
O(100) Telescopes On the Ground
O(10) are triggered per event
O(10,000) events per second
O(10) Time slices
O(10) bits/slice
Signal (Digital Counts)
Time Slice Index
telescope position (m)
−500
0
500
1000

CTA Data Size *Fermi Estimation*



10 tels • 1000 pix • 10 slices • 10 bits • 10,000 Hz ≈ **10 GB/s**

(CERN ATLAS: 1 GB/s)

Not possible within CTA budget!

Telescope	Data rate	Data rate (Central Trigger)
LST	110Gb/s	40Gb/s
MST	450Gb/s	150Gb/s
SST	60Gb/s	30Gb/s
Total	610Gb/s	220Gb/s

≈ 30 GB/s

Central Trigger

Full waveform signal from photodetectors (Total 1314h):
130 PB/year

Pixel integration

3% full waveform signal, remaining signal integrated:
21 PB/year

Telescope	Data rate (sampled pixels)	Data rate (Integrated)	Total
LST	2.2Gb/s	8.6Gb/s	11Gb/s
MST	4.5Gb/s	15.5Gb/s	20Gb/s
SST	1Gb/s	4.1Gb/s	5.1Gb/s
Total			36 Gb/s

≈ 4.5 GB/s

Final estimate including 20% technical data:

CTA North: **5.4 GB/s**

CTA South: **3.2 GB/s**

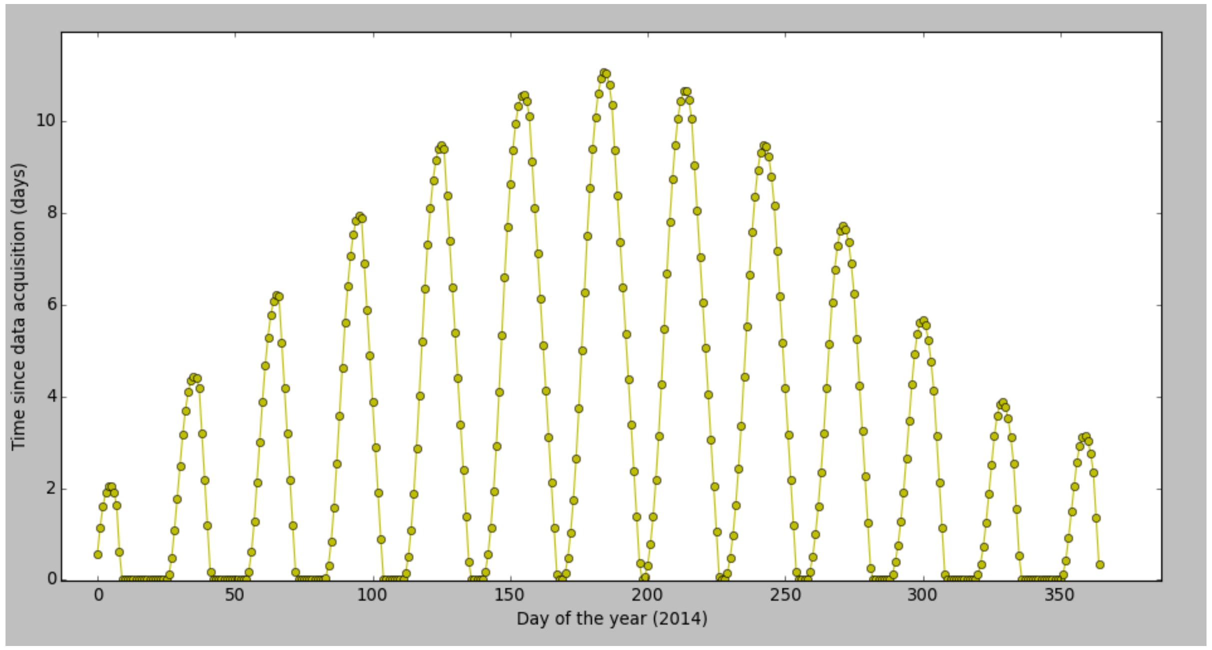
→ 40 PB/y, max 370 TB/day!

Source: Nadine Neyroud, CTA PC meeting July 2017

Constraints

Have to worry about:

- Budget (we aren't CERN)
- Off-site link: 1Gb/s
- Want to transfer data off-site in < 10 days



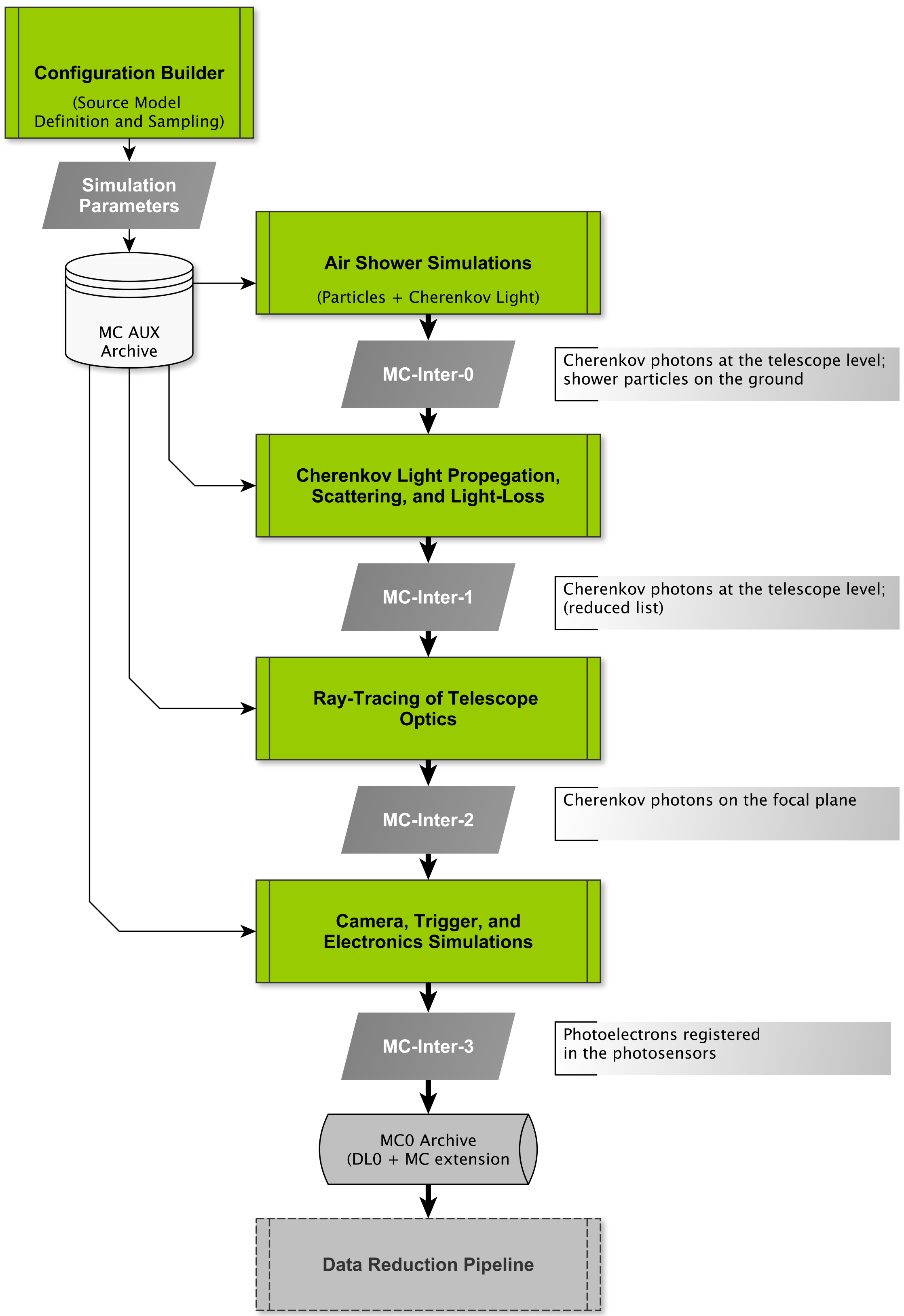
Daily data transfer duration/ Day of the year

Result: need to reduce raw data to ≈4 PB/year

- additional factor or 10x needed after trace integration (factor of 70x overall from pure raw data)

→ Some fraction of the data processing *must* occur on-site, and not all data can be retained

Note: Don't forget simulations!



Vast amount of Monte-Carlo data are needed to understand the system

- generate Instrumental Response Functions (IRFs) for science
- Train reconstruction algorithms

Simulation data size is roughly equal to the raw data volume!

- both for "parameterized grid" or "runwise" simulations
- but:
  - all versions do not need to be maintained (can delete old files roughly yearly)
  - no MC data on-site (so no limit on transfer speed, only cost of storage)

Solution?

We cannot store all data: need ≈70x reduction

- lossless data compression: factor of 2x at least, maybe closer to 3x with novel techniques, better data formats, etc.
- lossy data compression:
  - **Drop "uninteresting" waveforms** (and perhaps full pixel info), leaving only 3% of pixels on average with waveforms (typical image size)
  - **sparse techniques to compress images or waveforms**? (wavelets, curvelets, etc)
  - **drop very hadron-like events** (only gives factor of 2 or so, and adds complexity)

Need to study implementation and science impact

- Real-time? Off-line + local data cache?
- How robust is the technique? Do we lose important info? What are the risks? What can still be "**reprocessed**" later?

Data Processing In general

Data Levels: amount of processing

MCs are somewhere here right now

Pipeline starts here
Will need to eventually produce data here to be compatible with "real" CTA data

Reconstructed Events
(many reconstructions and parameters, no more telescopes)

Science Data:
Classified Events (final reconstruction), IRFs

R0 *(raw low-level)* camera data transmitted from telescope to central servers. R0 content and format is internal to each camera and is specified and coordinated between individual camera teams.

R1 *(raw common)* data output by an individual camera functional unit to the camera DAQ functional unit. This is the first level of data seen by the ACTL system and is therefore as common as possible between all cameras/hardware. Exceptionally, some R1 data may be stored for engineering purposes.

DL0 *(raw archived)* all archival data from the data acquisition hardware/software. This is the first level of data that are stored in the bulk archive. This includes both camera event data and technical data from other subsystems, such as non-camera devices or software.

DL1 *(processed)* processed DL0 data that may still include some TEL data and parameters derived from them. For example this includes calibrated image charge, Hillas parameters, and a usable telescope pattern. This is only optionally stored in the archive.

DL2 *(reconstructed)* reconstructed shower parameters such as energy, direction, particle ID, and related signal discrimination parameters. At this point, no TEL information is stored. For each event this information may be repeated for multiple reconstruction and discrimination methods. This is only optionally stored in the archive. At this point, telescope-wise info is generally dropped.

DL3 *(reduced)* Sets of selected (e.g. gamma-ray candidates, electron candidates, selected hadron candidates, etc.) events with a single final set of reconstruction and discrimination parameters, along with associated instrumental response characterizations and any technical data needed for science analysis.

DL4 *(science)* binned data products like spectra, sky maps, or light curves, along with associated data (source models, fit results, etc).

DL5 *(high-level)* high-level or "legacy" observatory data, such as CTA survey sky maps or the CTA source catalog.

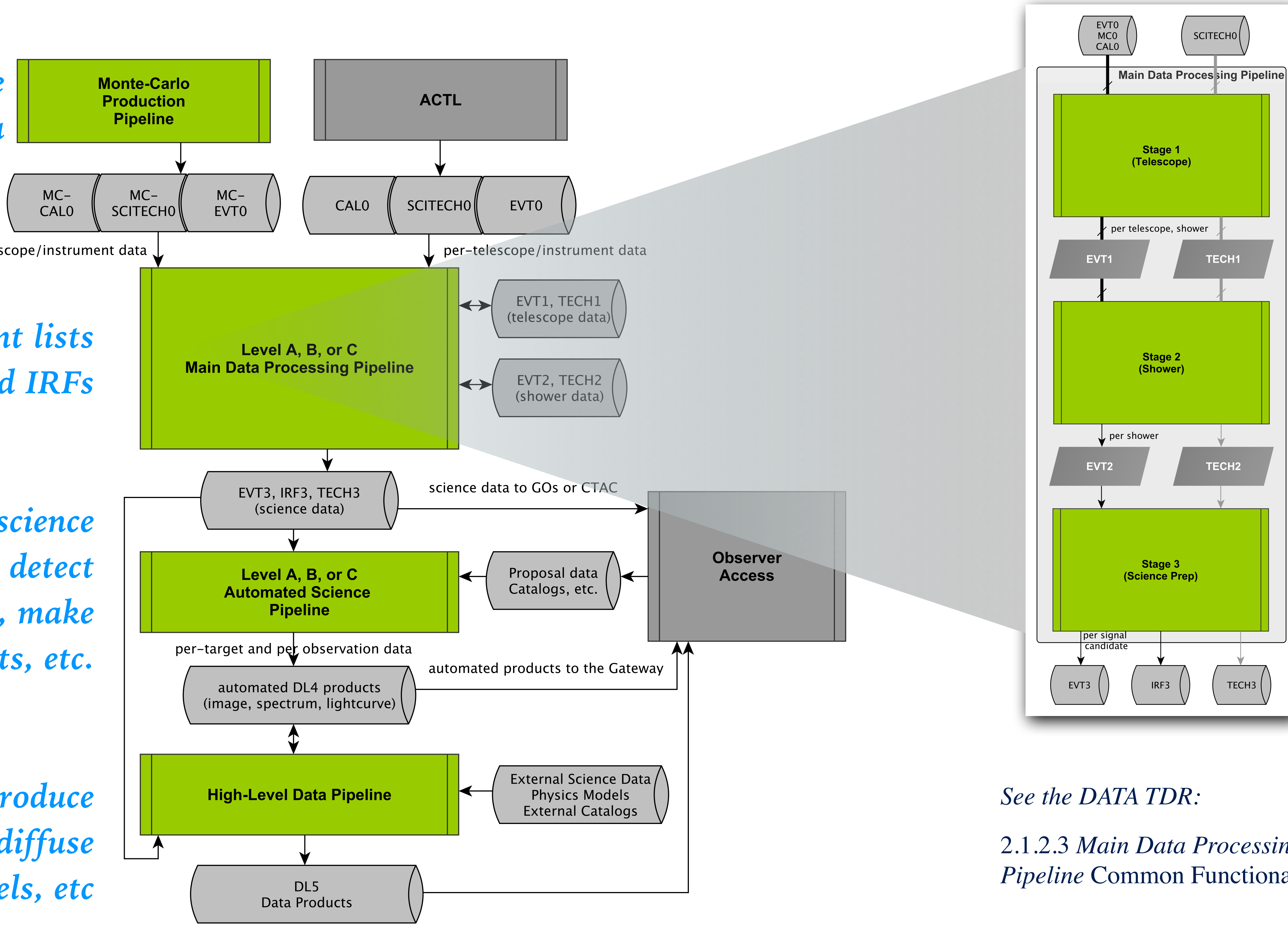
Data Processing Pipeline

generate MC data

make event lists and IRFs

run science tools, detect sources, make sky plots, etc.

produce catalogs, diffuse models, etc



See the DATA TDR:

2.1.2.3 *Main Data Processing Pipeline* Common Functional Desig

CEA, MPIK, LAPP, LUTH, INAF, IFEA, others

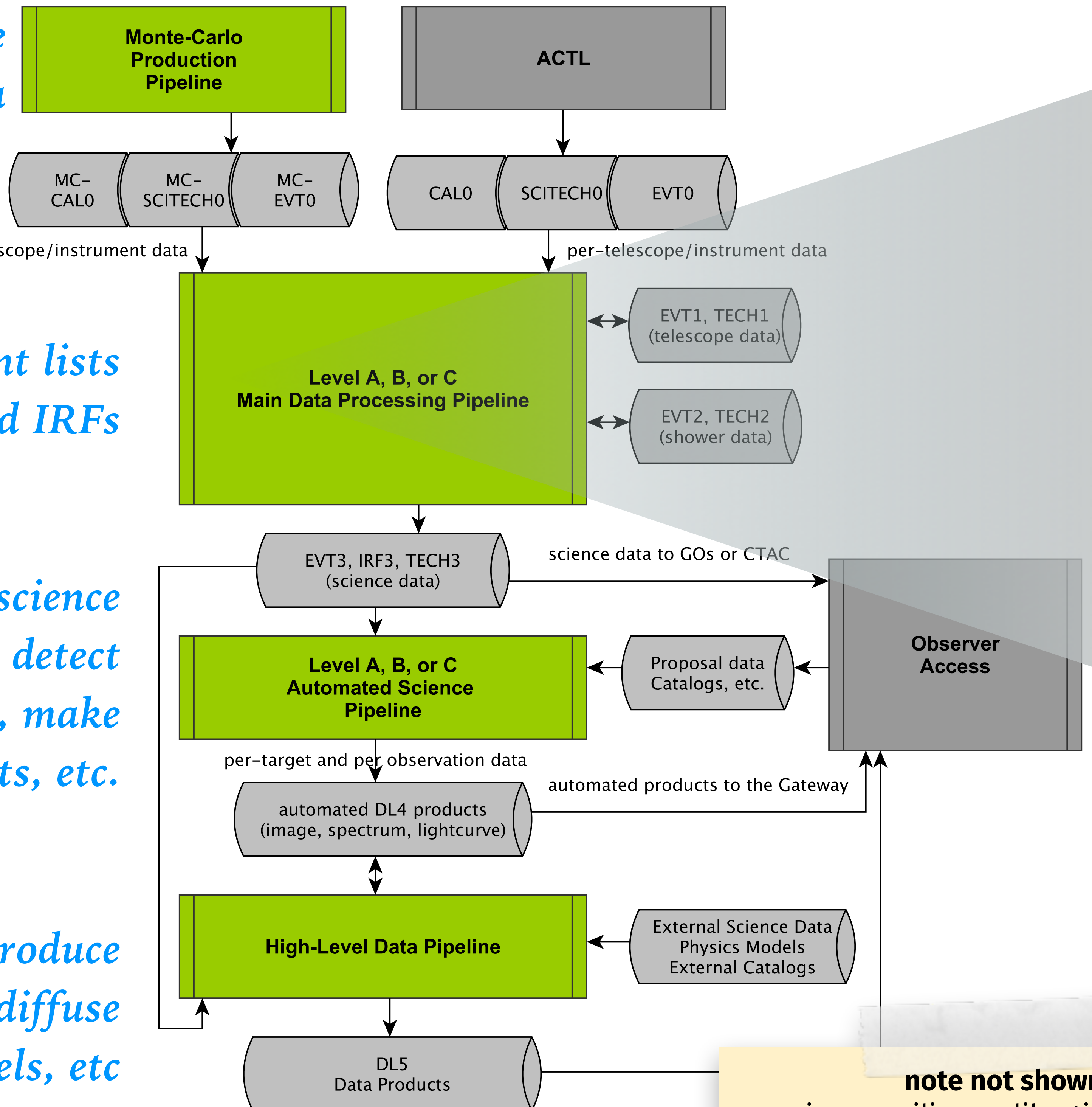
Data Processing Pipeline

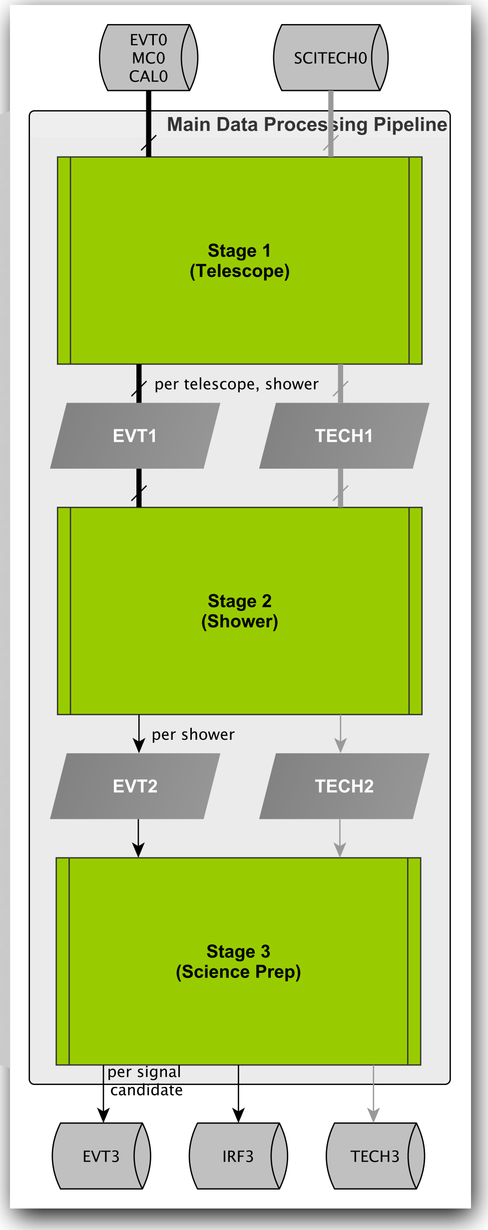
generate MC data

make event lists and IRFs

run science tools, detect sources, make sky plots, etc.

produce catalogs, diffuse models, etc





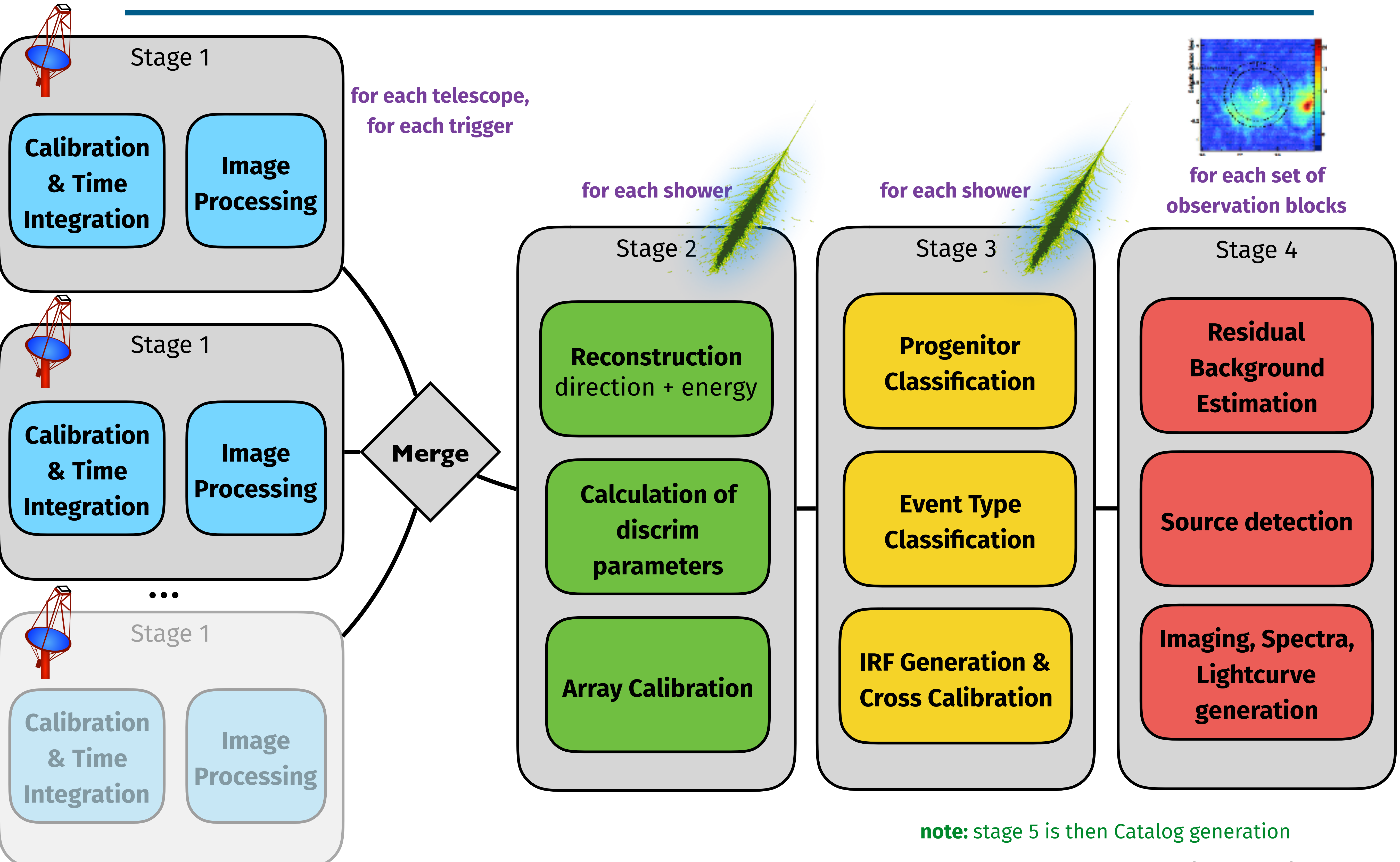
See the DATA TDR:

2.1.2.3 *Main Data Processing Pipeline* Common Functional Desig

note not shown:
various auxiliary calibration pipelines
(run separately from main data processing)

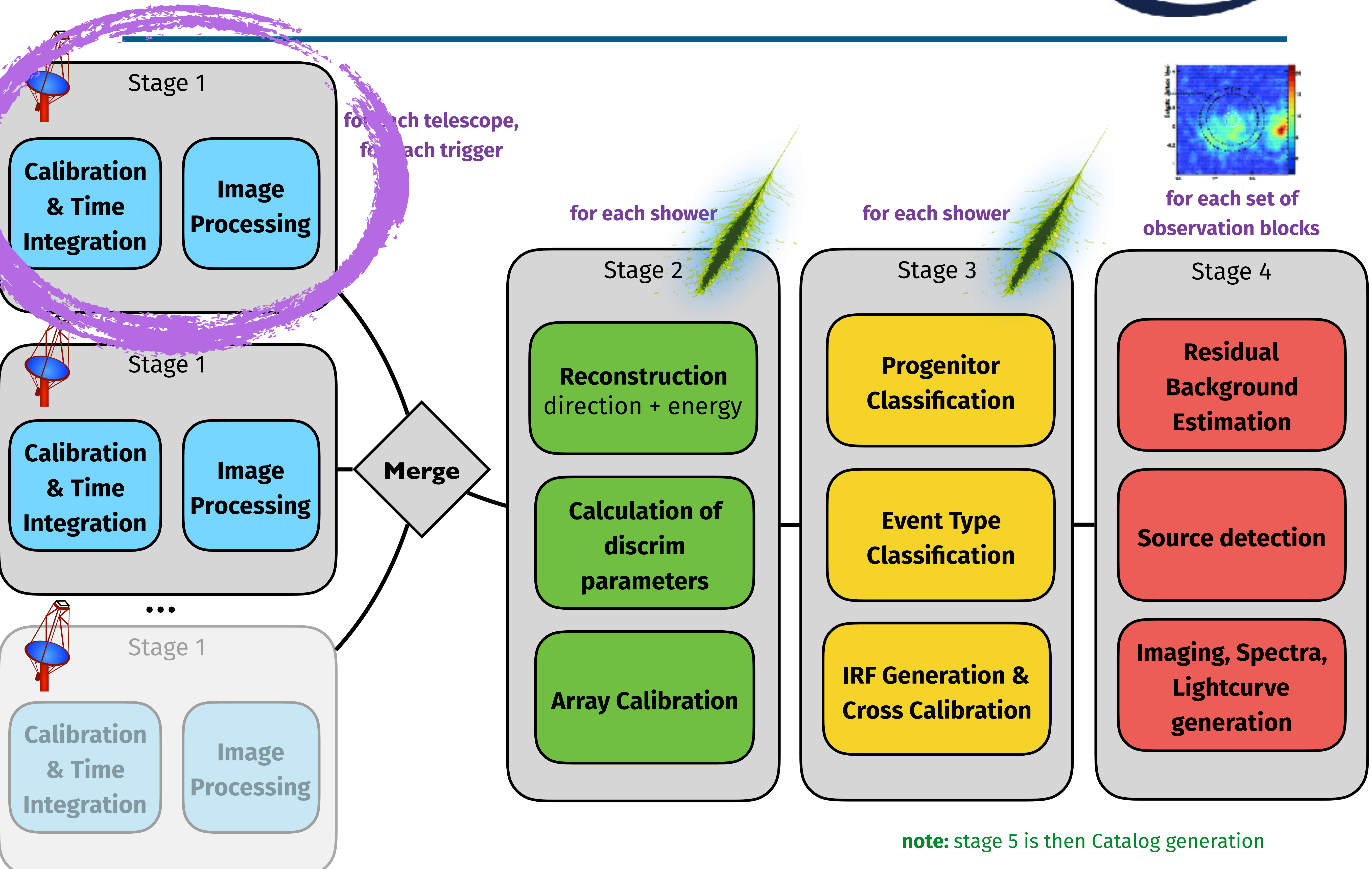
CEA, MPIK, LAPP, LUTH, INAF, IFEA, others

Main Data Processing Pipeline

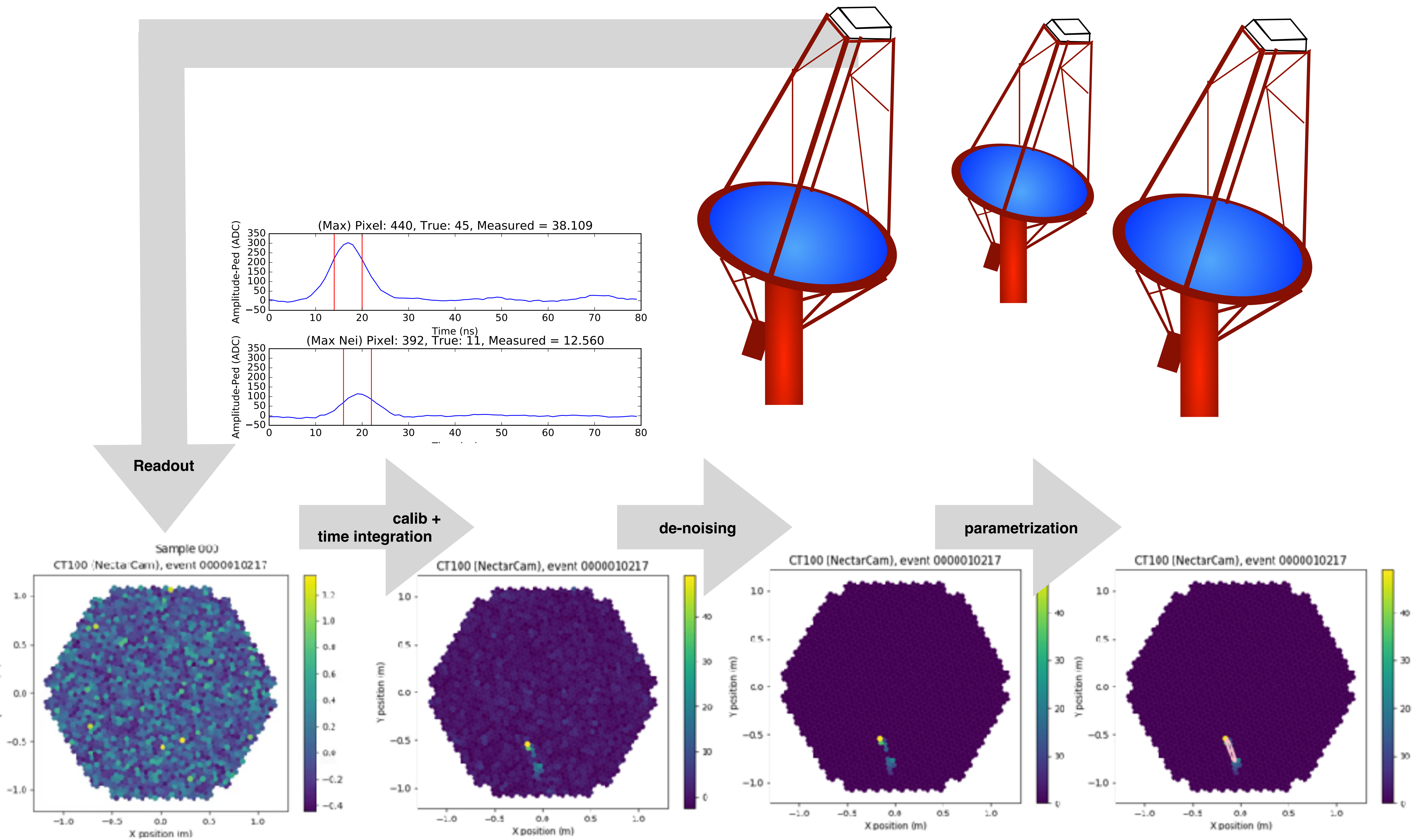


note: stage 5 is then Catalog generation

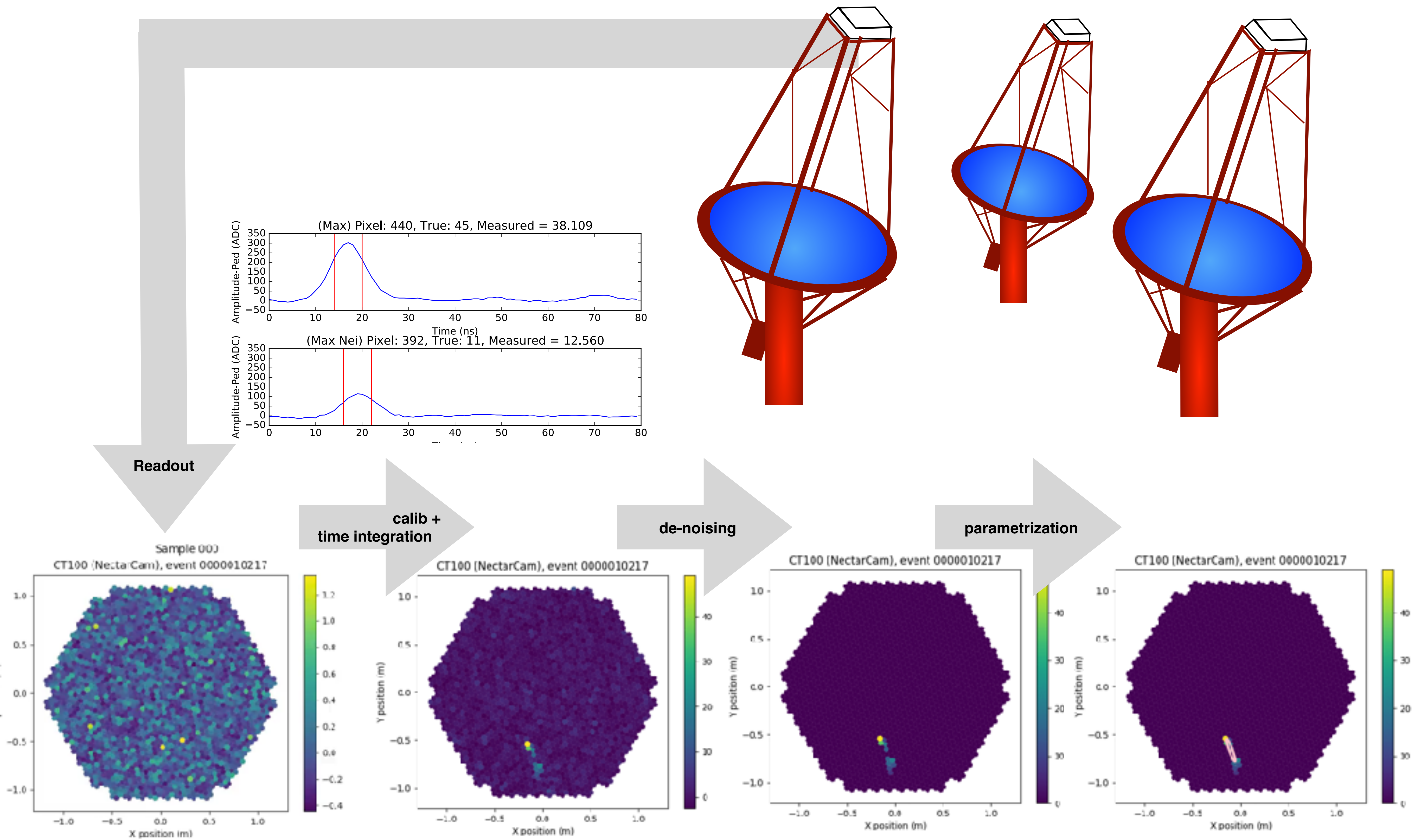
Main Data Processing Pipeline



Stage 1: Per-telescope

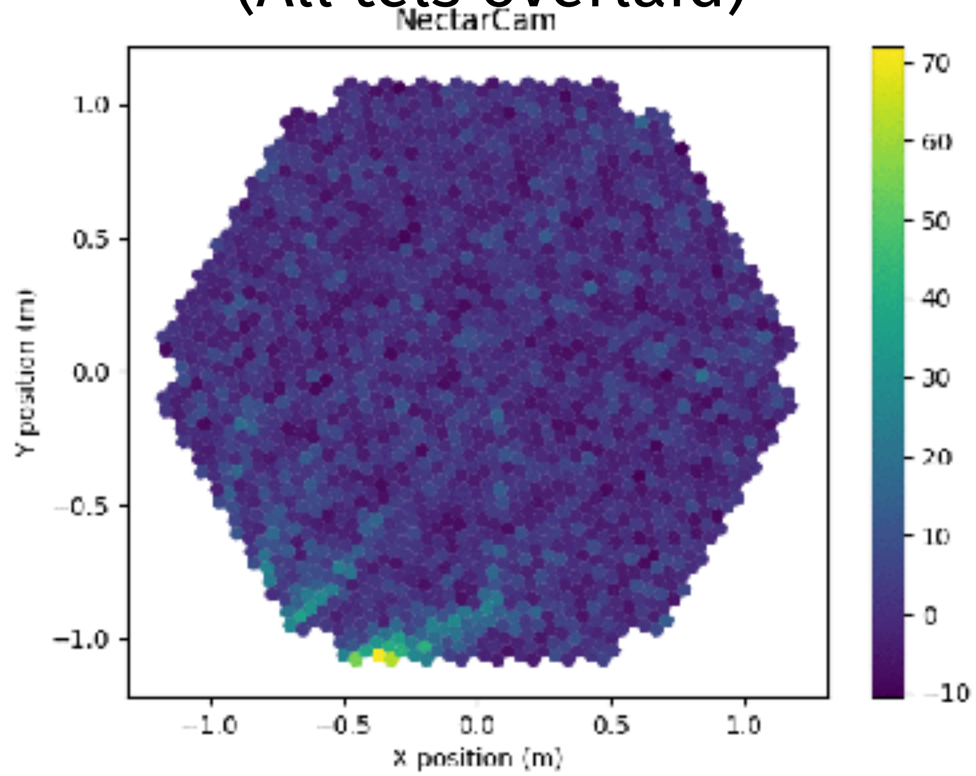


Stage 1: Per-telescope

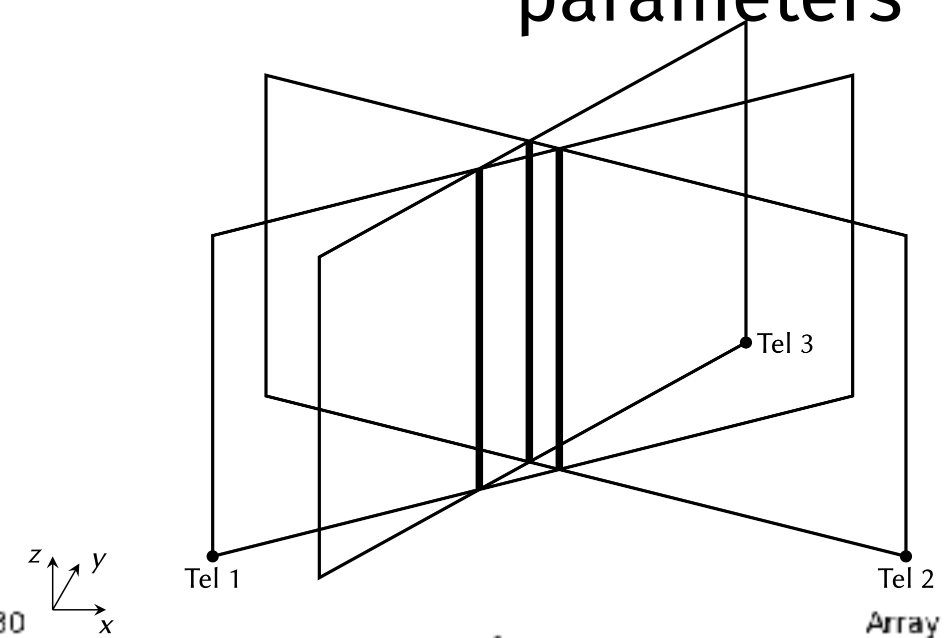


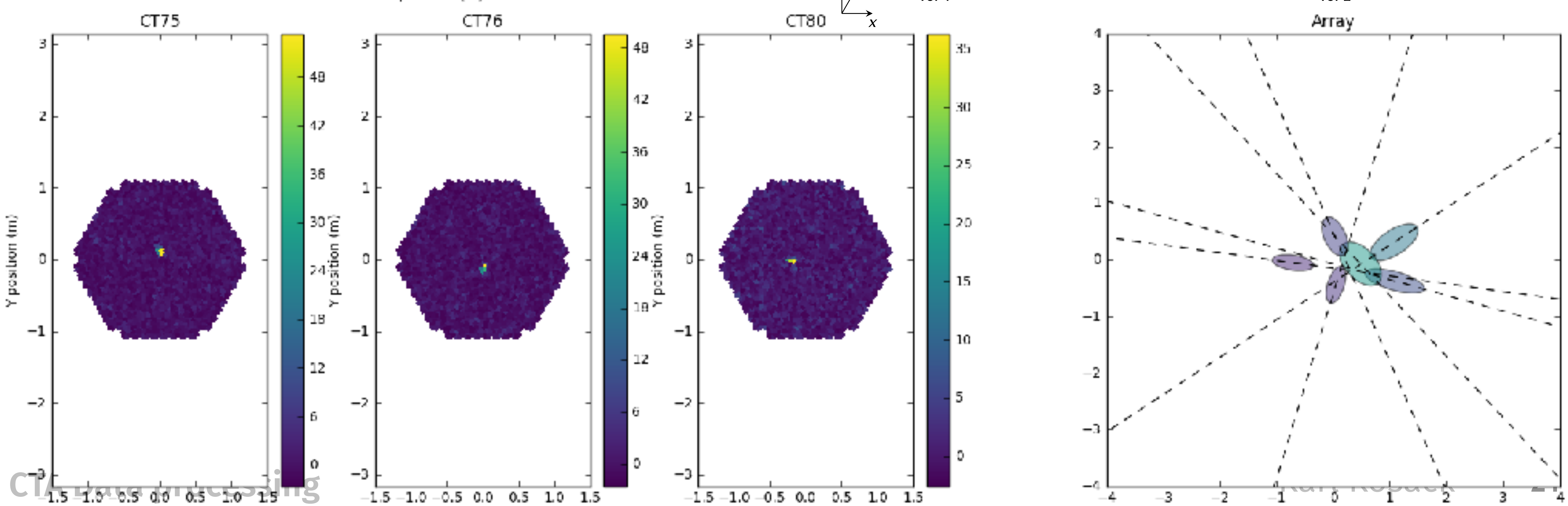
Stage 2: Reconstruction

(All tels overlaid)



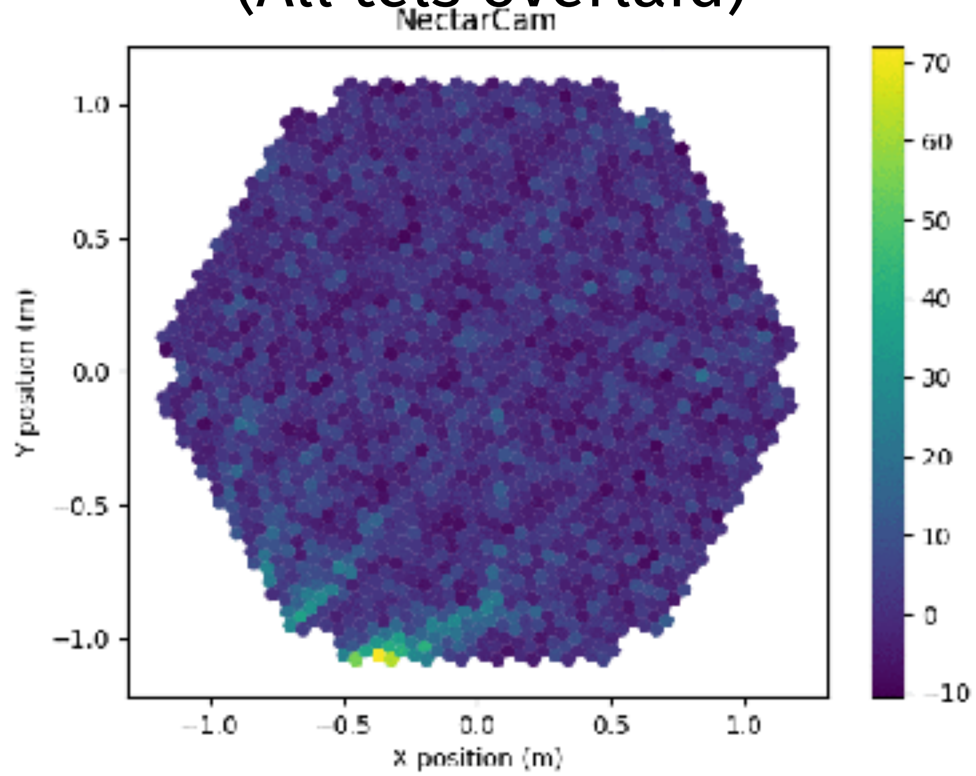
Outputs are: **Point-of-Origin** + **Energy** + Classification parameters



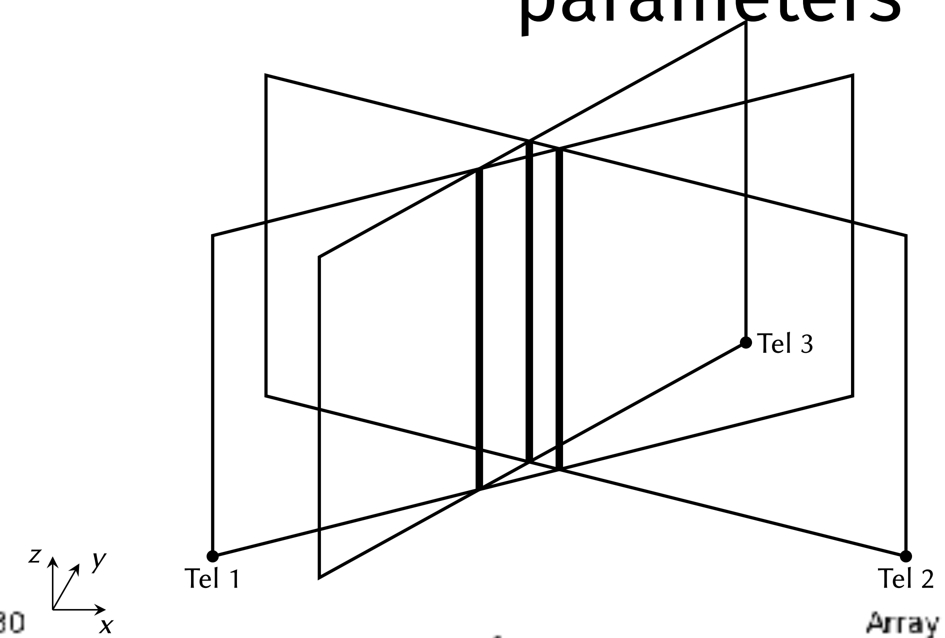


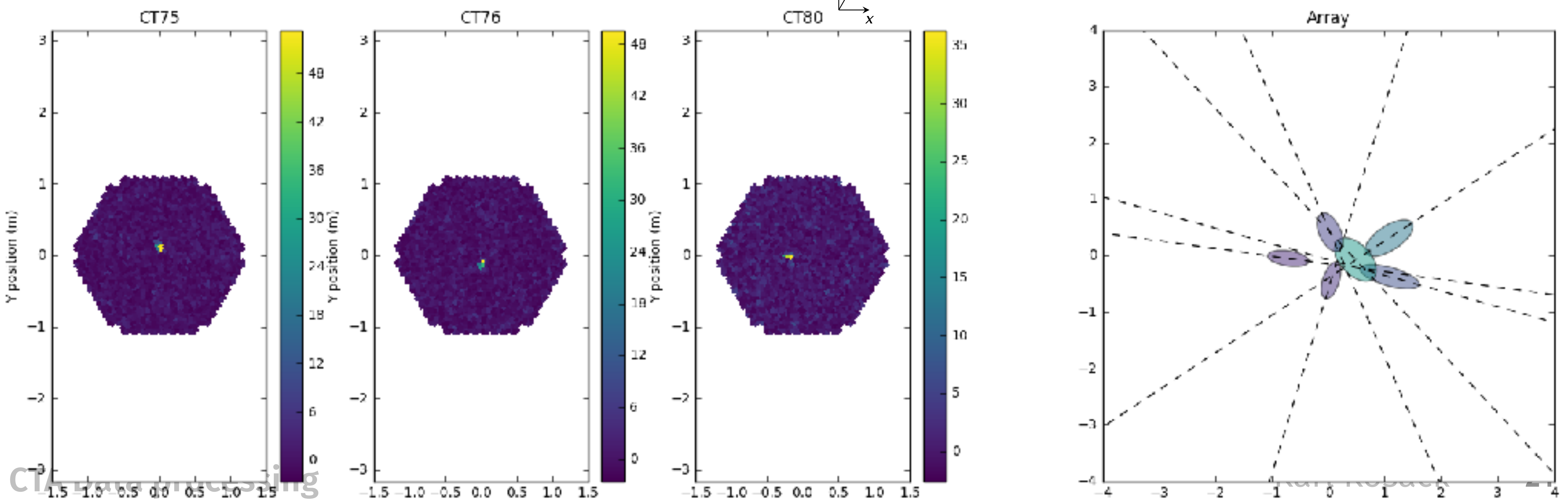
Stage 2: Reconstruction

(All tels overlaid)



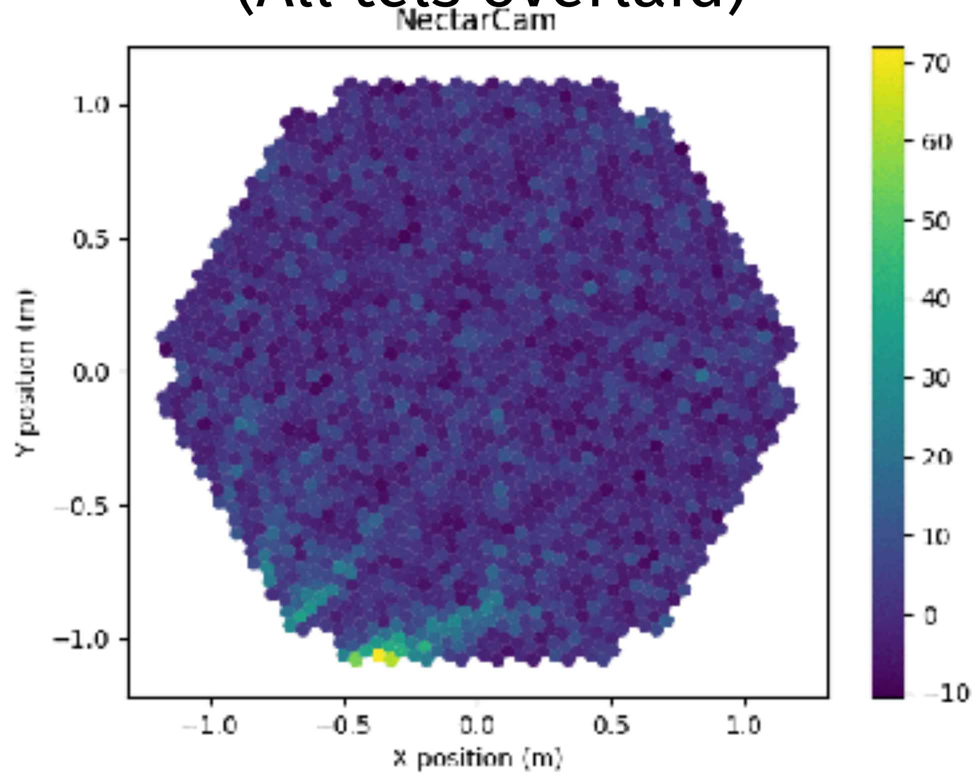
Outputs are: **Point-of-Origin** + **Energy** + Classification parameters



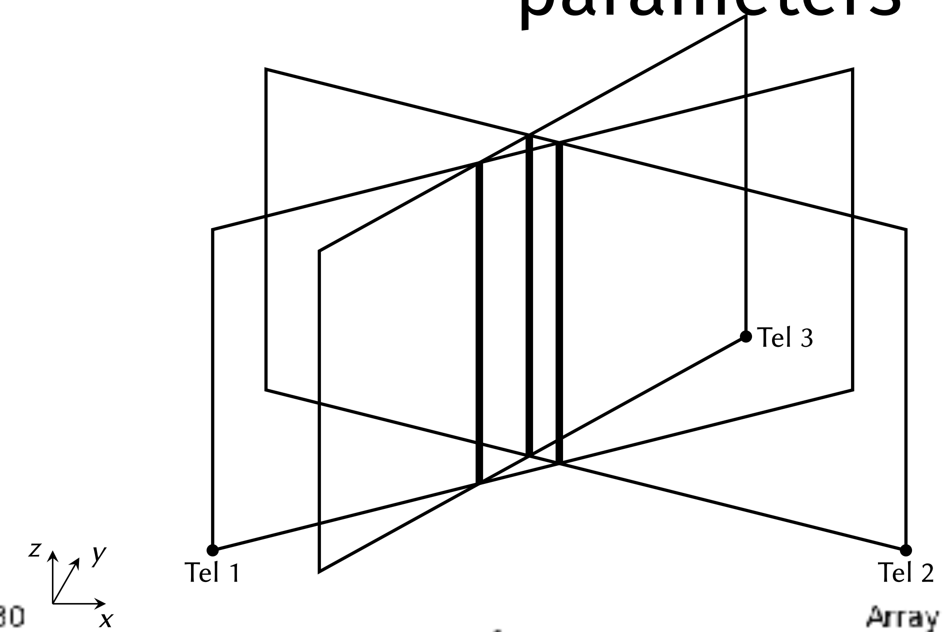


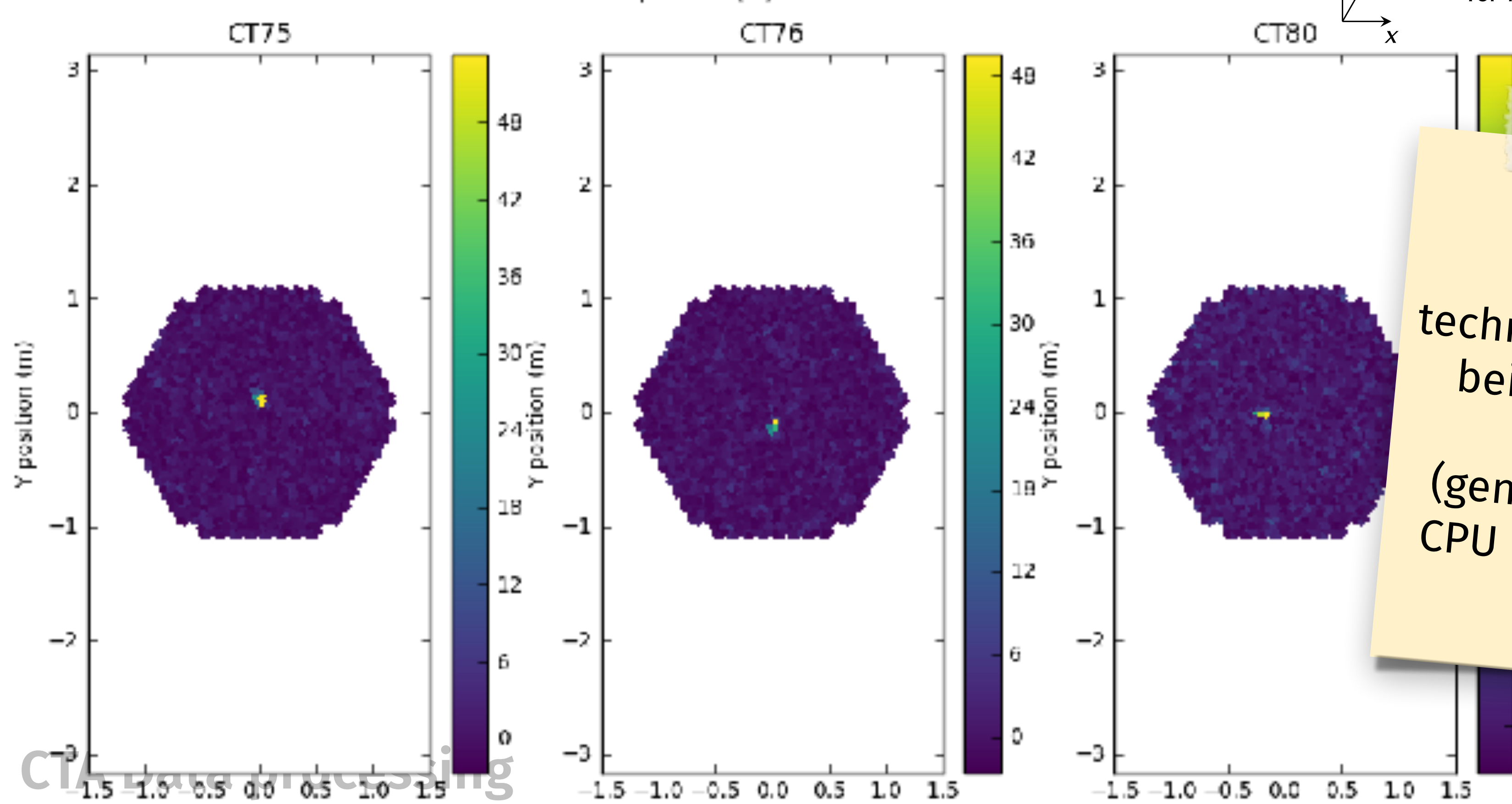
Stage 2: Reconstruction

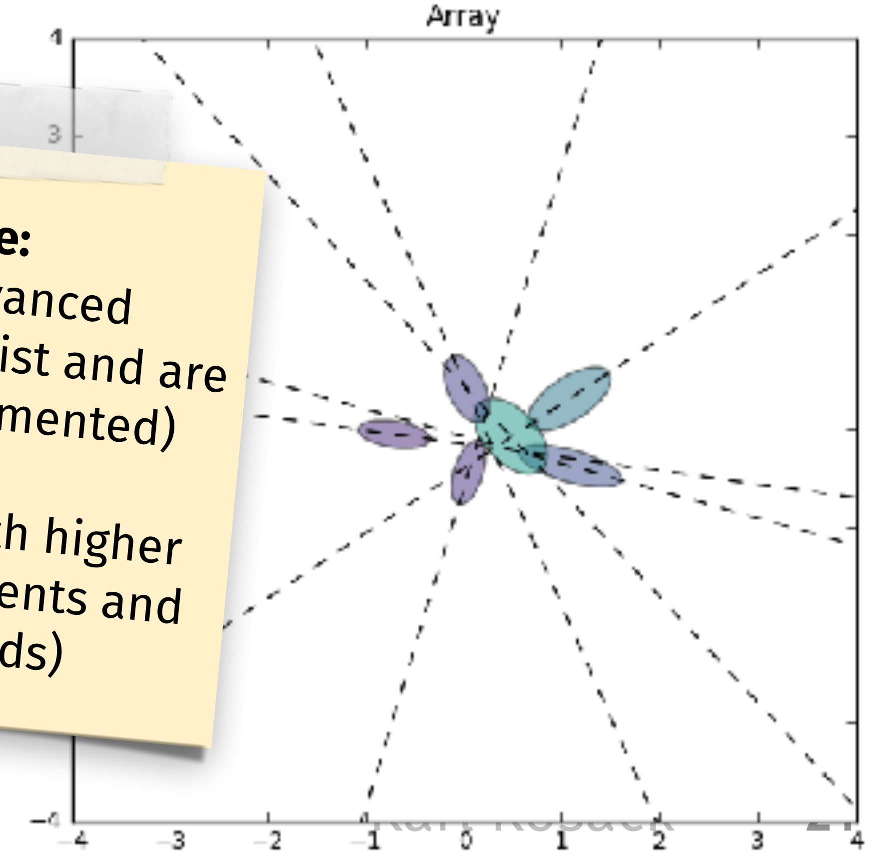
(All tels overlaid)



Outputs are: **Point-of-Origin** + **Energy** + Classification parameters



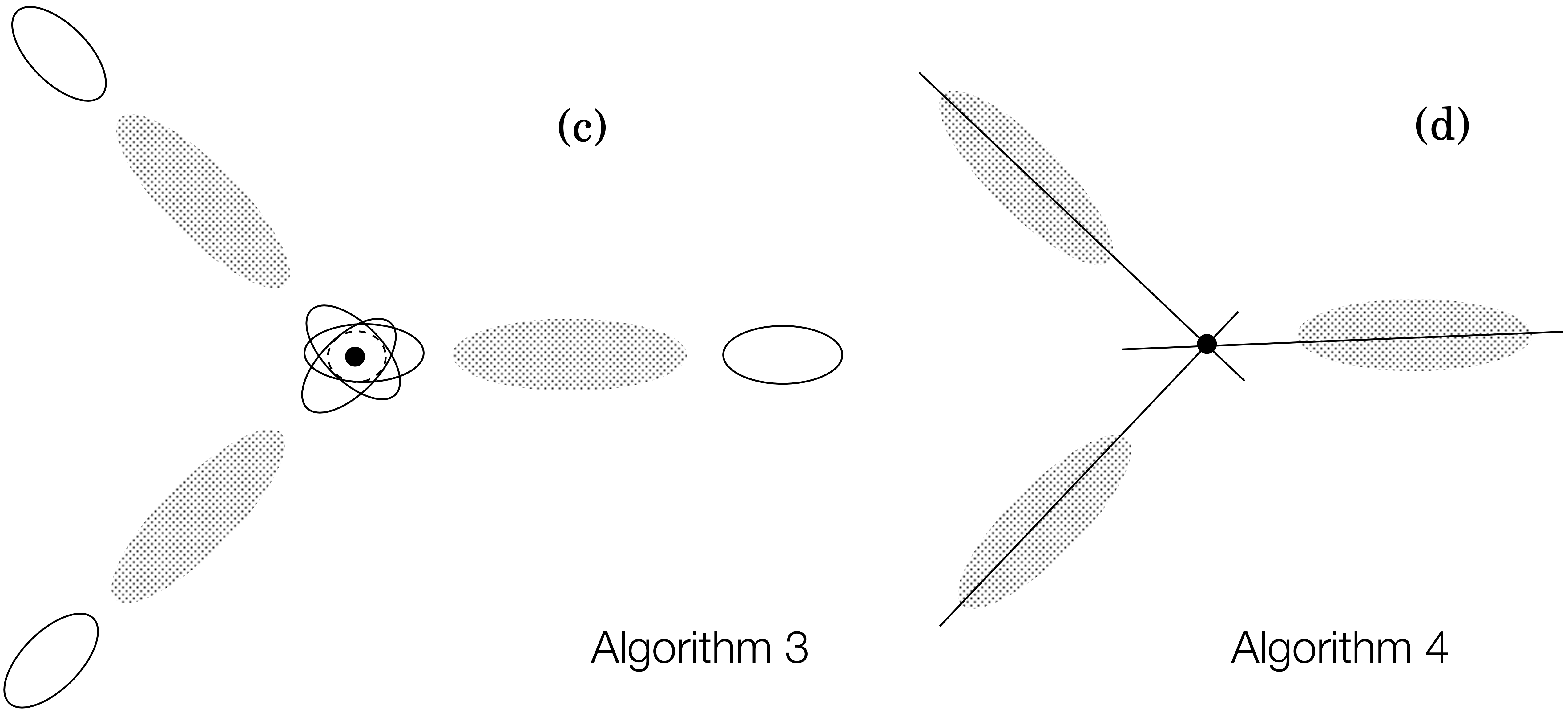




Note:
More advanced techniques exist and are being implemented)

(generally with higher CPU requirements and data needs)

Extensions to basic method



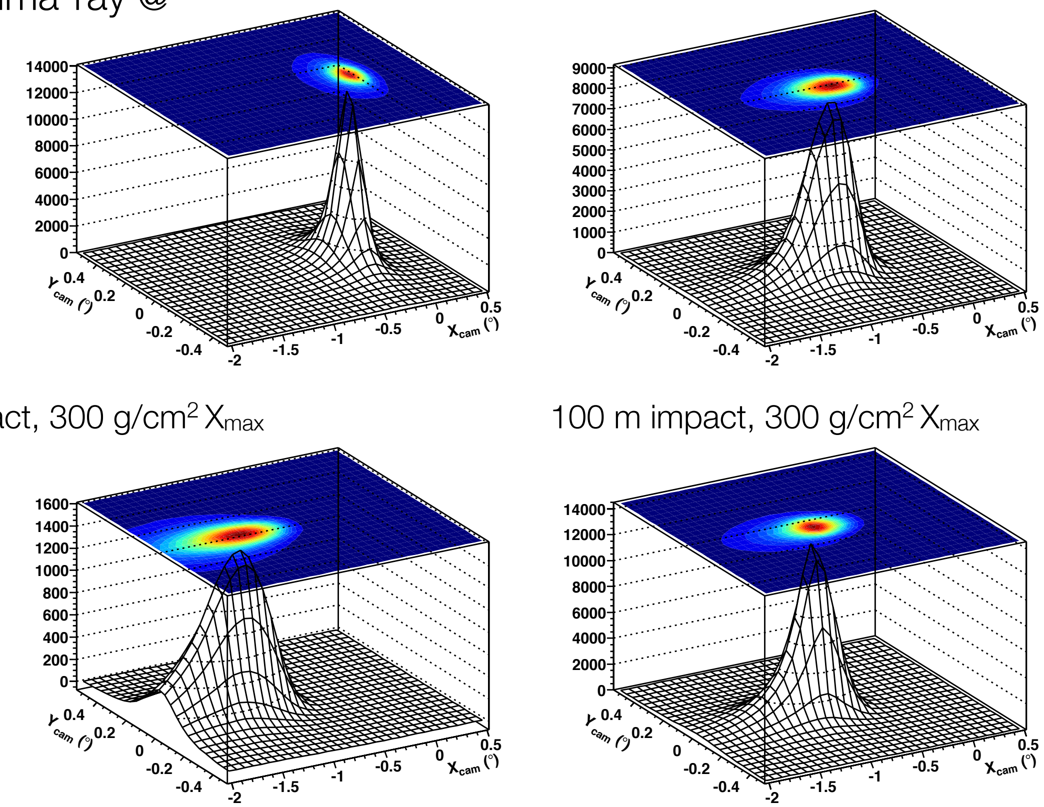
Use errors on single-telescope prediction

use global minimization
(current test show no significant improvement over a good weighting scheme)

Template Model Reconstruction

e.g. ImPACT, Model++

1 TeV Gamma-ray @



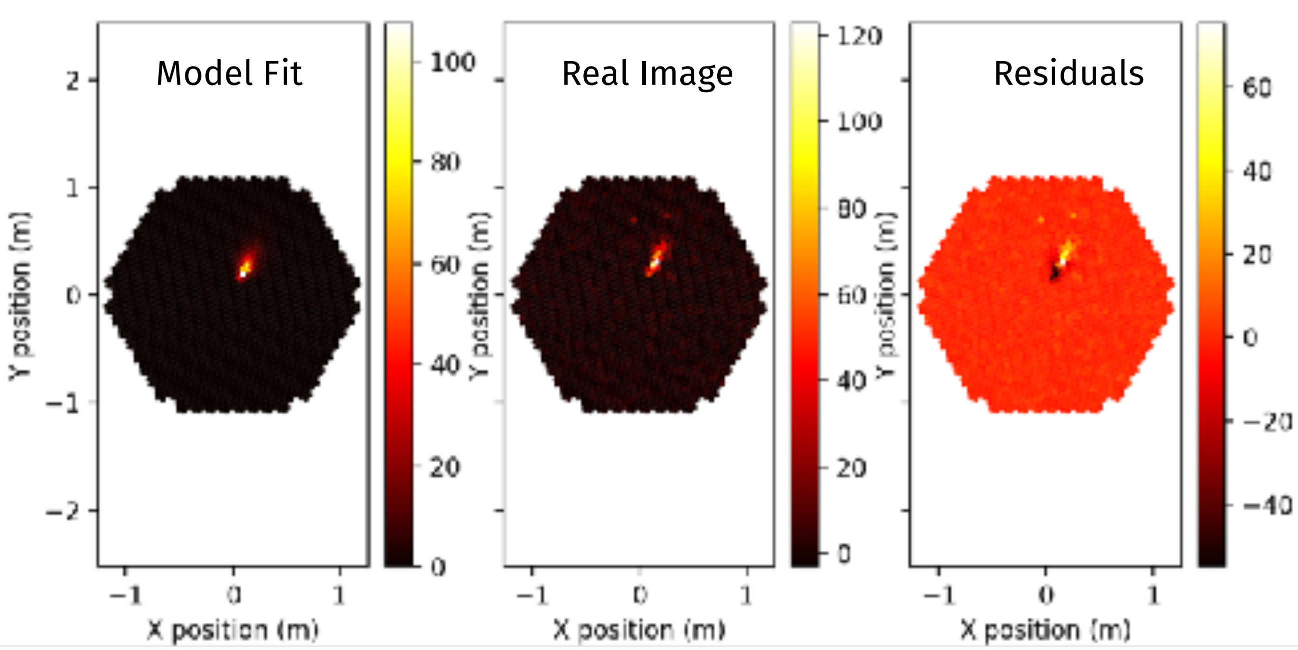
20 m impact, 300 g/cm^2 X_{max}

100 m impact, 300 g/cm^2 X_{max}

200 m impact, 300 g/cm^2 X_{max}

100 m impact, 400 g/cm^2 X_{max}

D. Parsons

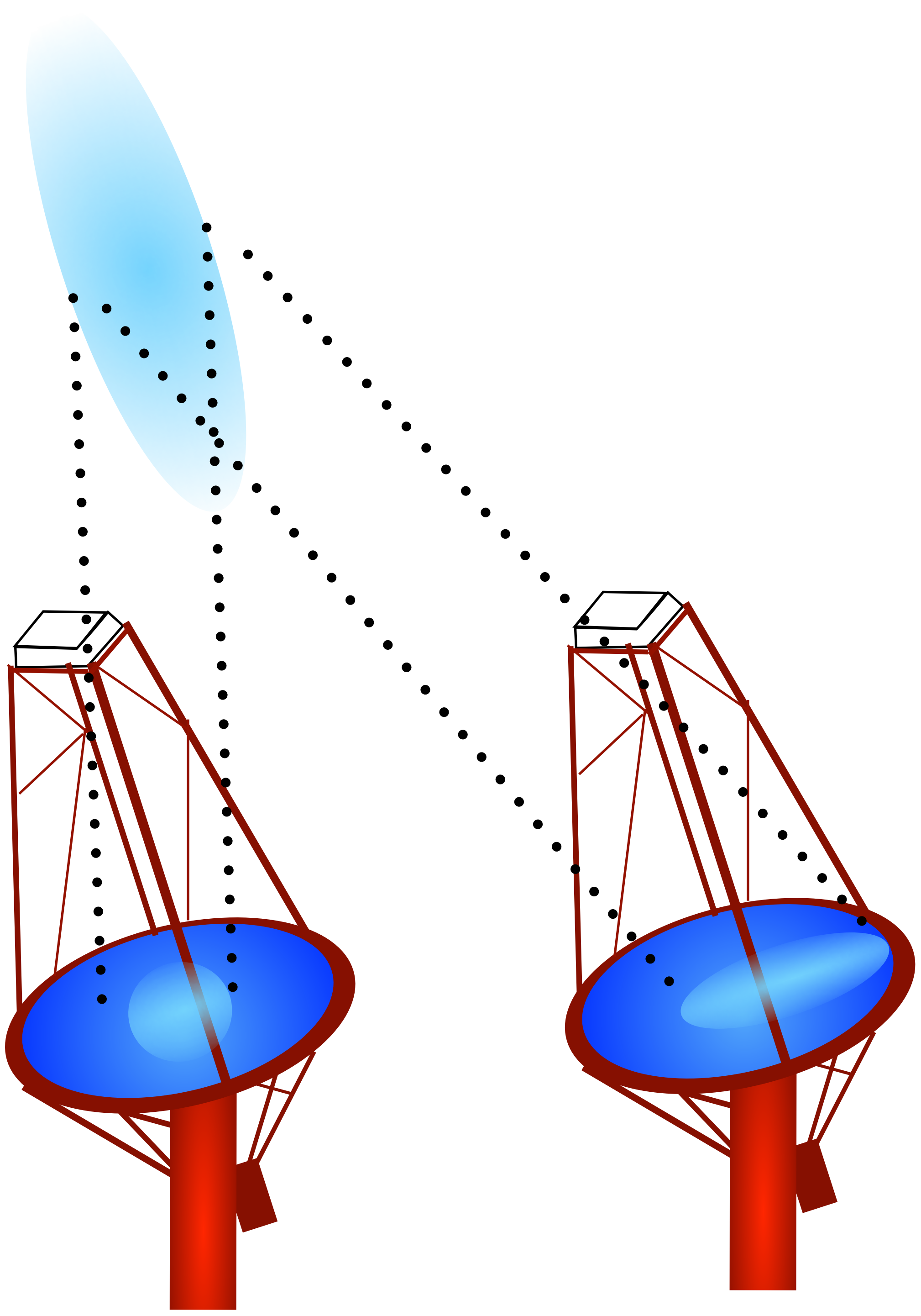


Build model from full monte-carlo simulations

- N-dimensional data-cube
- E, position, direction

Simultaneously fit all shower images to model

3D Model Reconstruction



Model shower as 3D object (e.g. Gaussian)

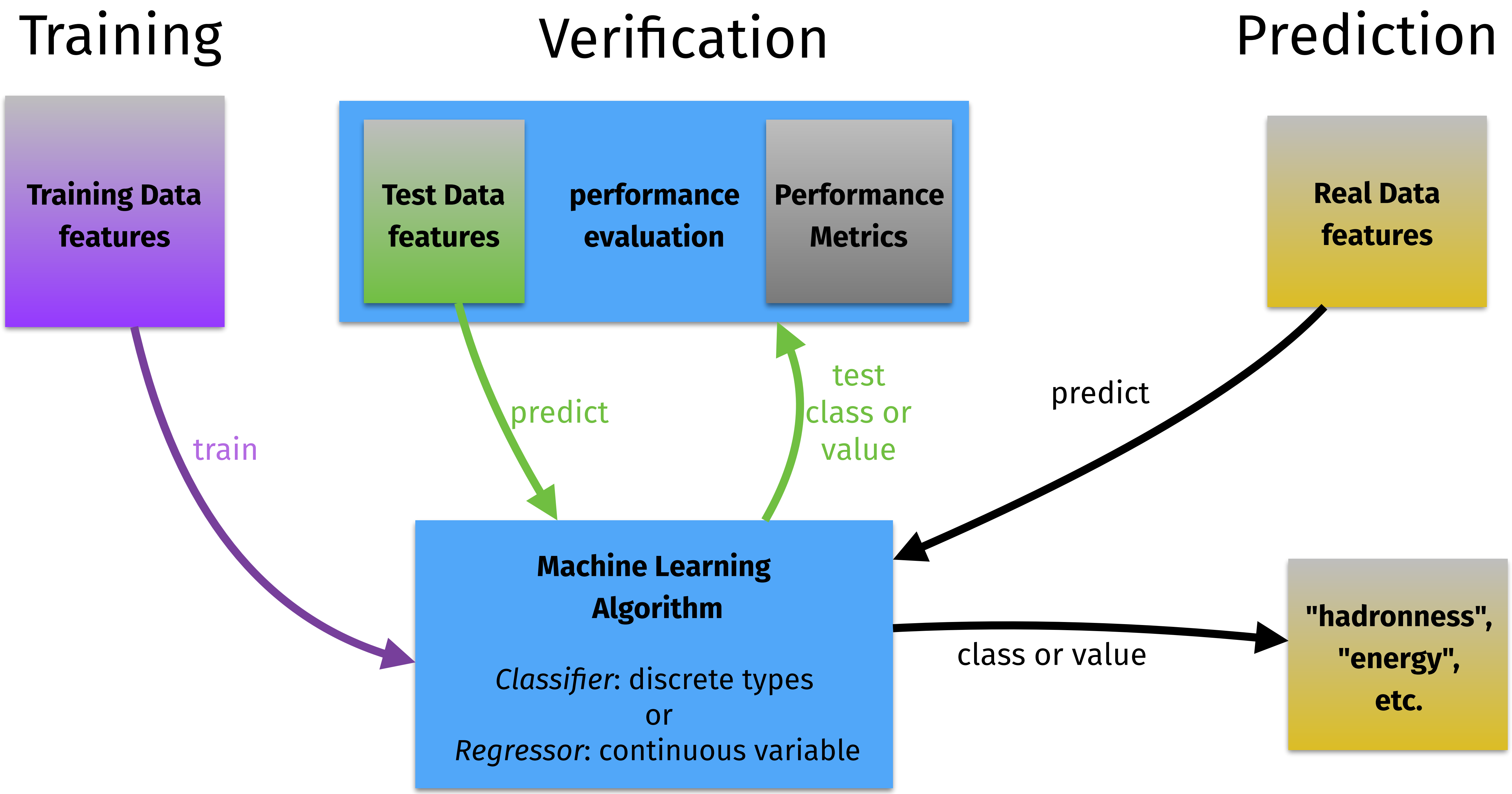
- position, direction, energy as parameters

Project shower image into cameras

Simultaneously fit real images to model

No model datacube needed, but model much more simplistic

Basic Machine Learning



Stage 3: Progenitor Discrimination

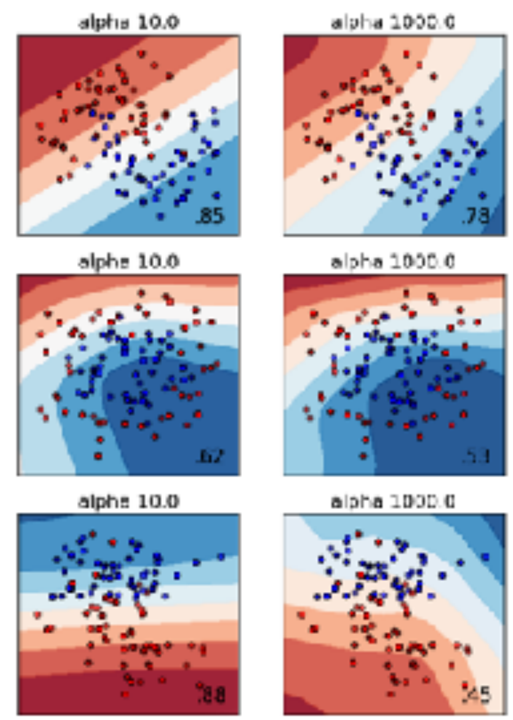
2-stage machine learning: **per-telescope prediction** → ***per-shower* prediction**

per-image parameter sets
(width, length, ...)

per-shower parameters
(impact distance, energy, number of tels, ...)

as many "features" as possible

parameters



scikit-learn — machine learning in Python

particle class

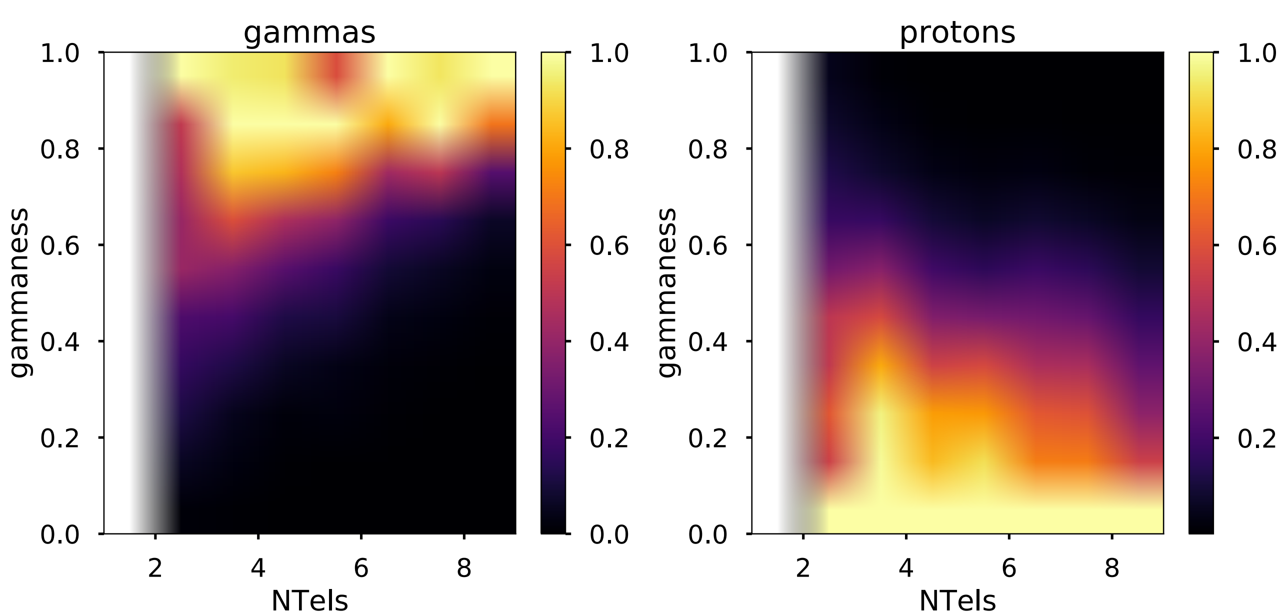
gamma-like

hadron-like

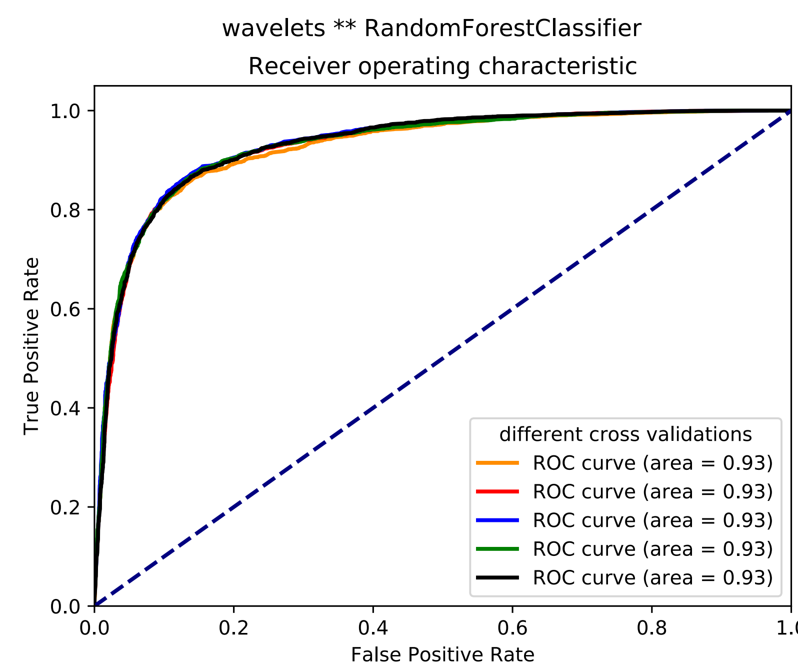
electron-like

Note: Also event quality classification, e.g. good PSF, good spectral resolution, sensitivity to unknown sources, ...

- using RandomForestClassifier implemented in *scikit-learn*
- data-mining approach: just throw all the data at it that we have
  - distance between telescope reconstructed impact position
  - error estimate on the impact position
  - Hillas parameters: width, length, skewness, kurtosis
  - total signal on camera
  - signal of the pixel with the highest count
  - total signal on all selected telescopes
  - number of selected telescopes



for now, cut on $NTels > 2$ & $gammaness > 0.75$

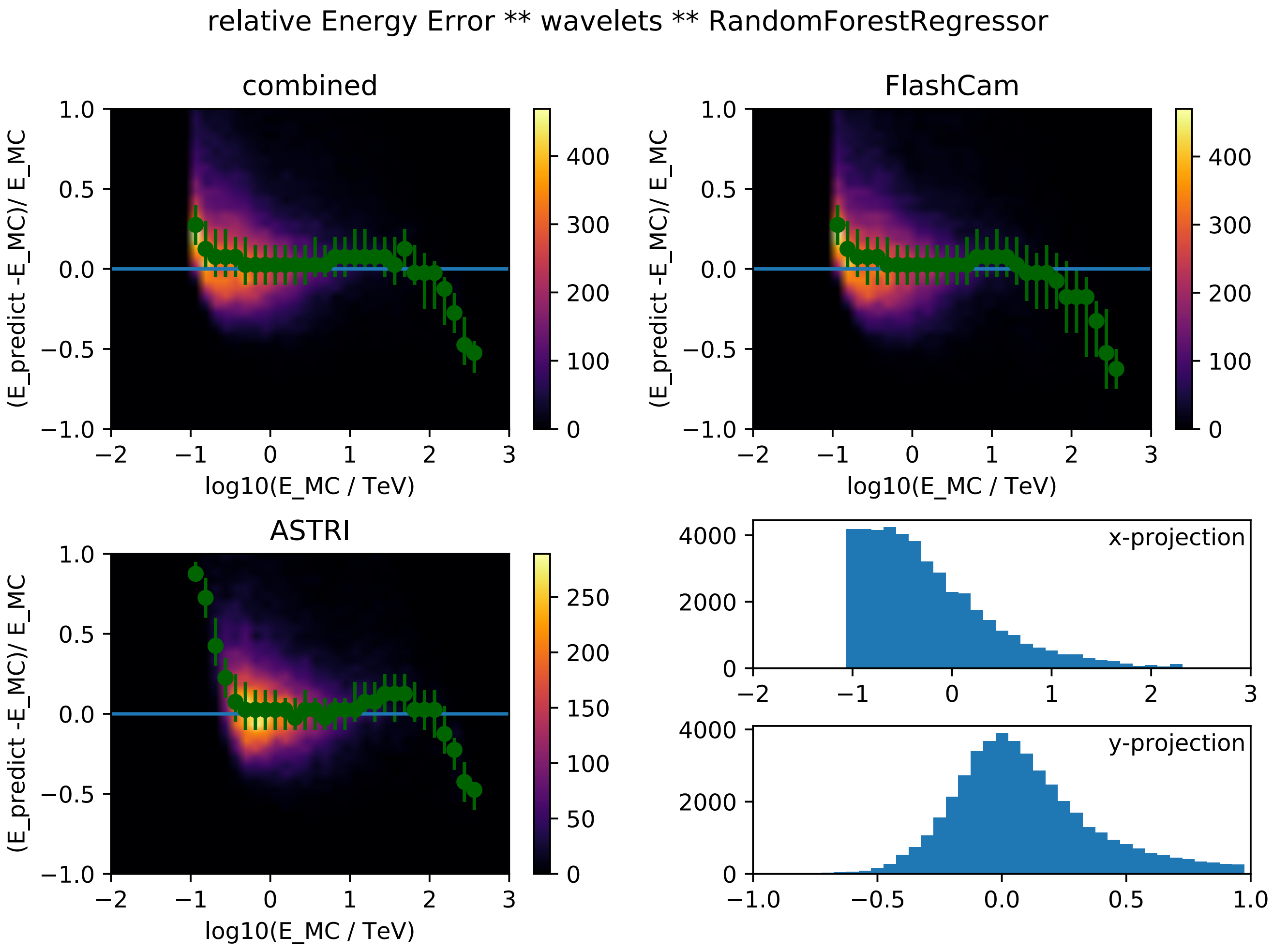


T. Michael

Stage 2: Energy Reconstruction

Several methods:

- Basic lookup-table as function of integrated signal + impact parameter
- **Machine-learning Regression** (currently used)
  - hillas parameters, reconstruction parameters, number of telescopes, etc
  - Predict per telescope and then combine



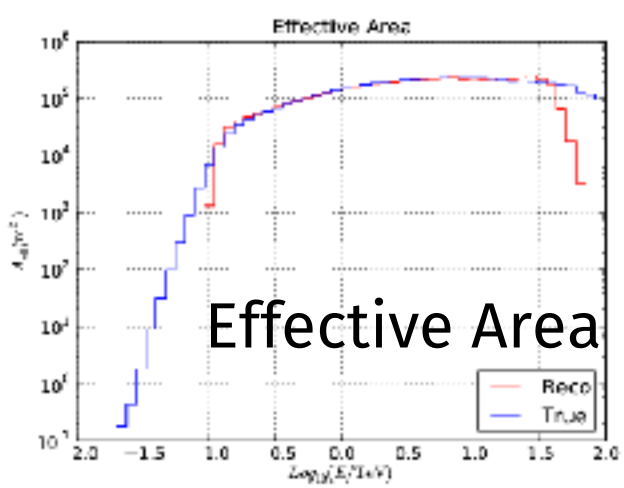
T. Michael

Output: Science Data

Event-List

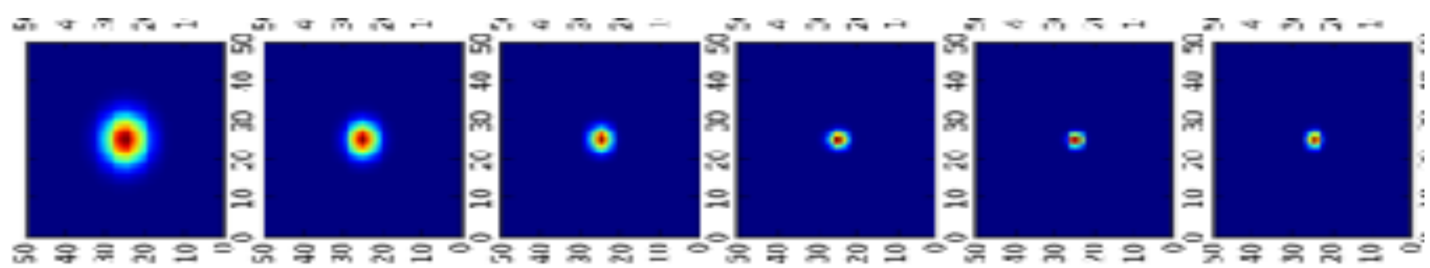
event_i	n_tels	RA	DEC	E	class	...
1	5	23.3	-40.1	0.01	g	
2	34	24.6	-40.5	20.0	g	
3	3	23.5	-41.12	0.45	g	
4	4	21.3	-38.2	1.03	h	

Instrumental Responses:

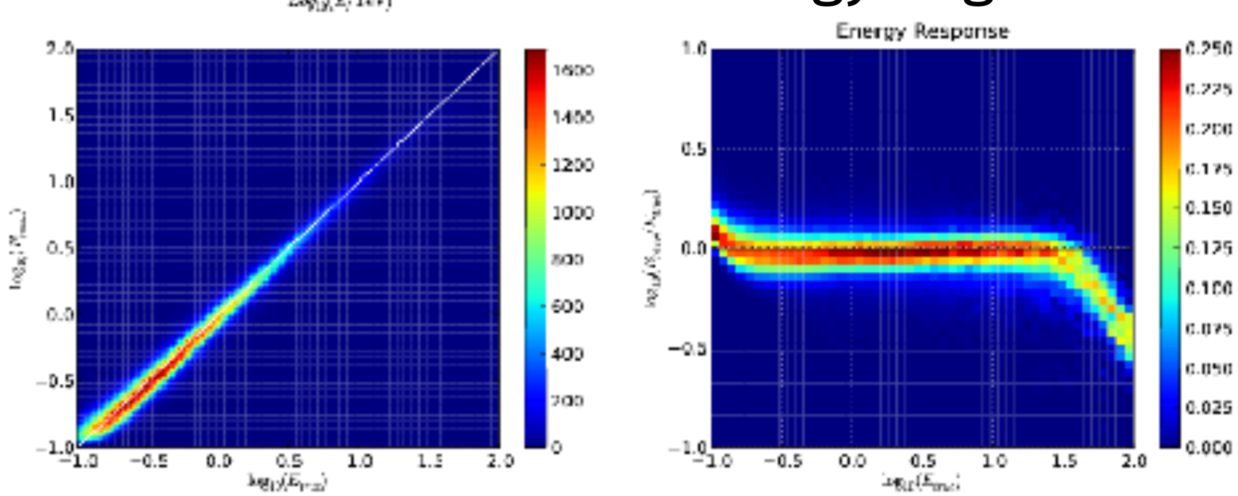


Effective Area

PSF



Energy Migration



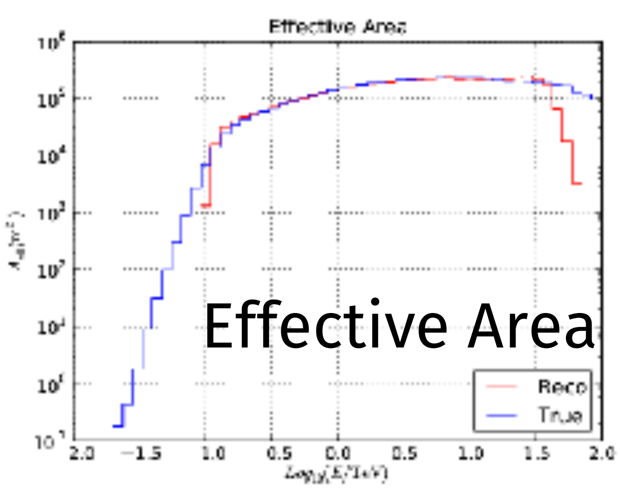
(note these are not CTA responses, just examples)

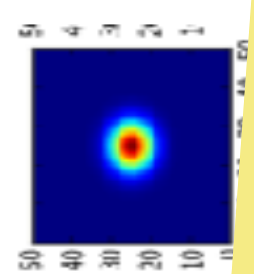
Output: Science Data

Event-List

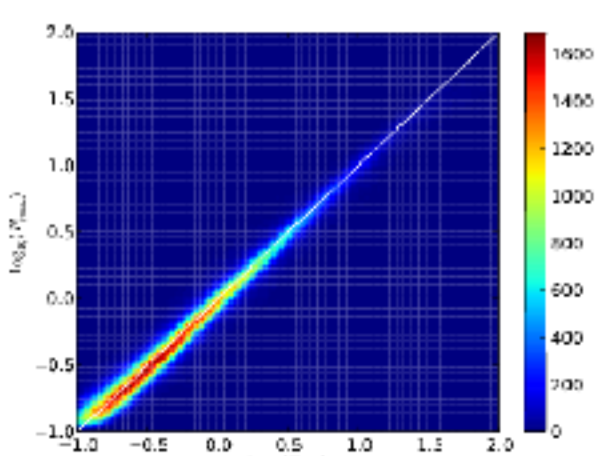
event_i	n_tels	RA	DEC	E	class	...
1	5	23.3	-40.1	0.01	g	
2	34	24.6	-40.5	20.0	g	
3	3	23.5	-			
4	4	21.3	-			

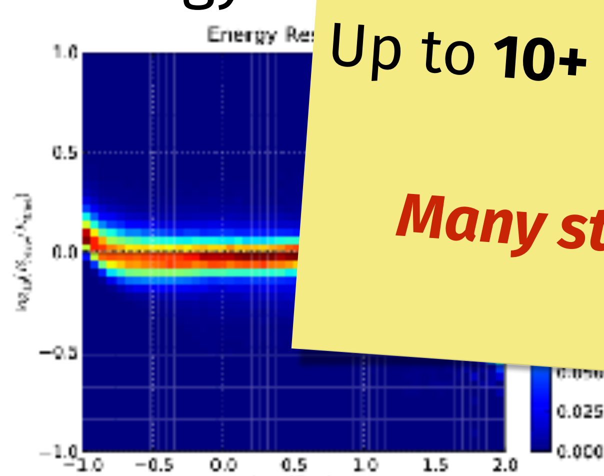
Instrument





Energy M





Response matrices are in general functions of:

Position in FOV,

Energy,

Observation Parameters (**Subarray** used , **Zenith** angle, **Azimuth** angle, **night-sky-brightness**, ...),

Analysis parameters (**algorithm** used, **event selection**)

Up to **10+ dimensions** if no assumptions are made

Many studies needed to understand the best factorization!

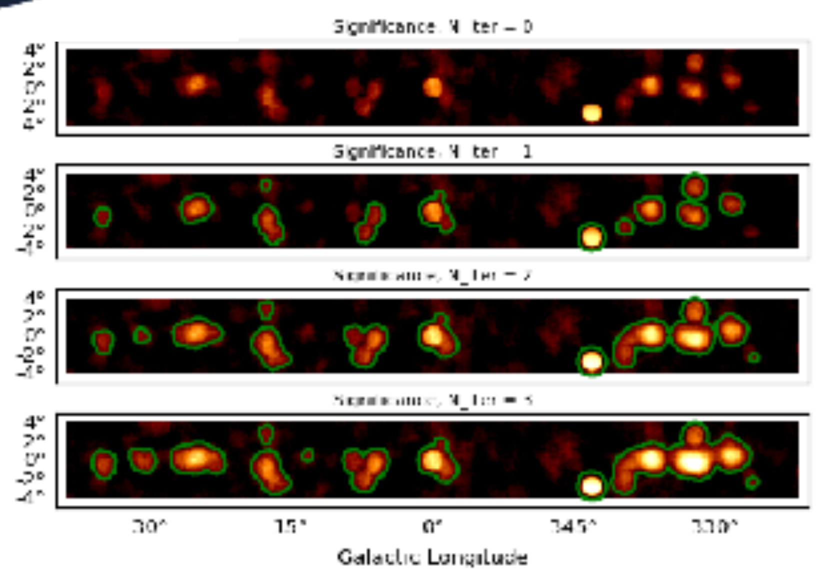
Stage 4: Automated Science

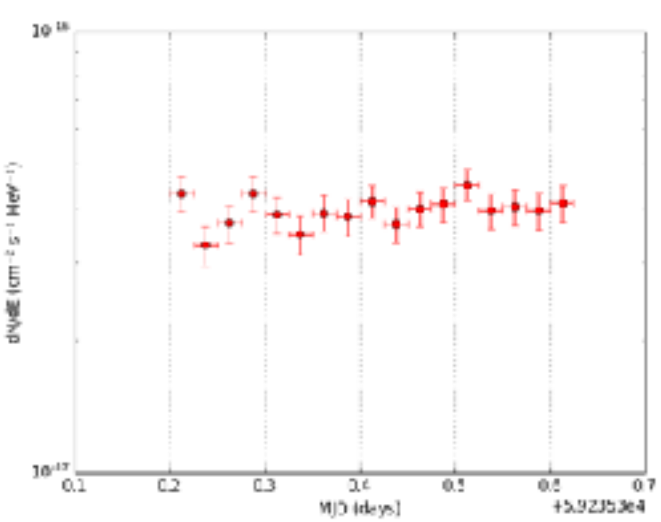
Produce preliminary results for proposal monitoring and alerts

Science Tools

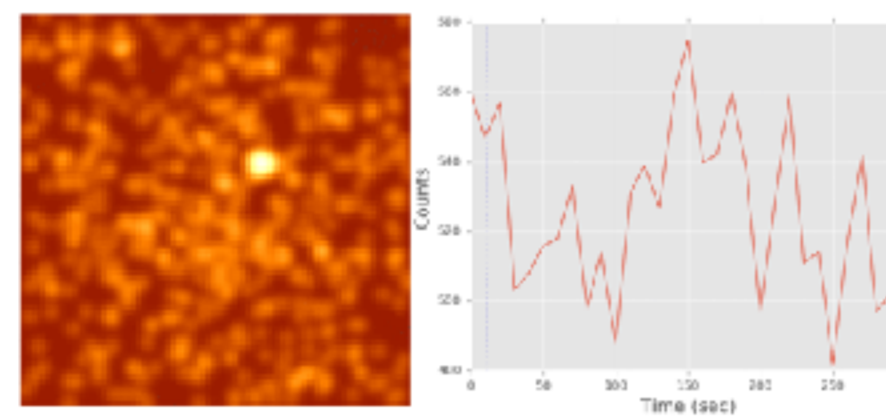
ctools cherenkov telescope array

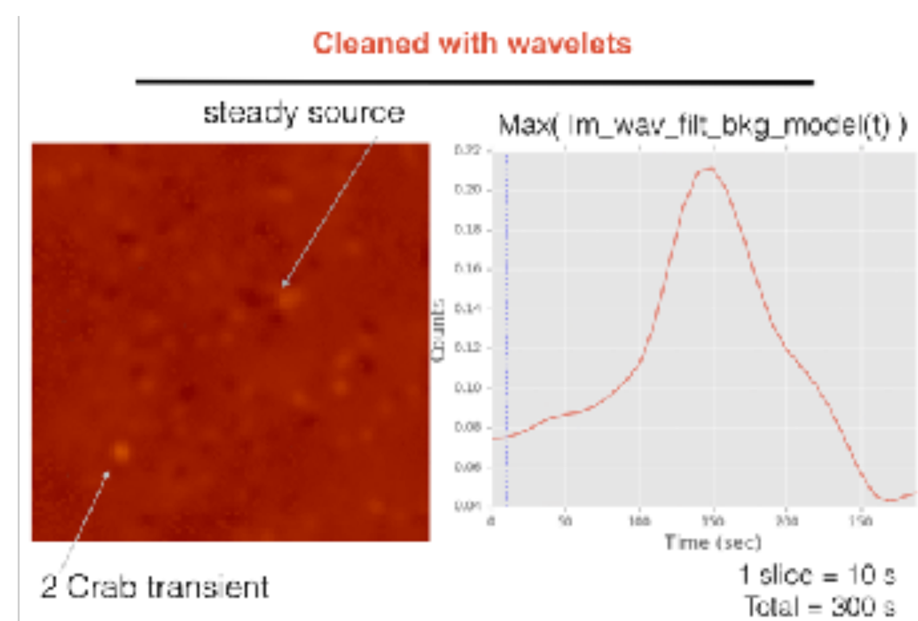
A Python package for gamma-ray astronomy





Custom Tools for Real-Time science and alert generation (on-site)





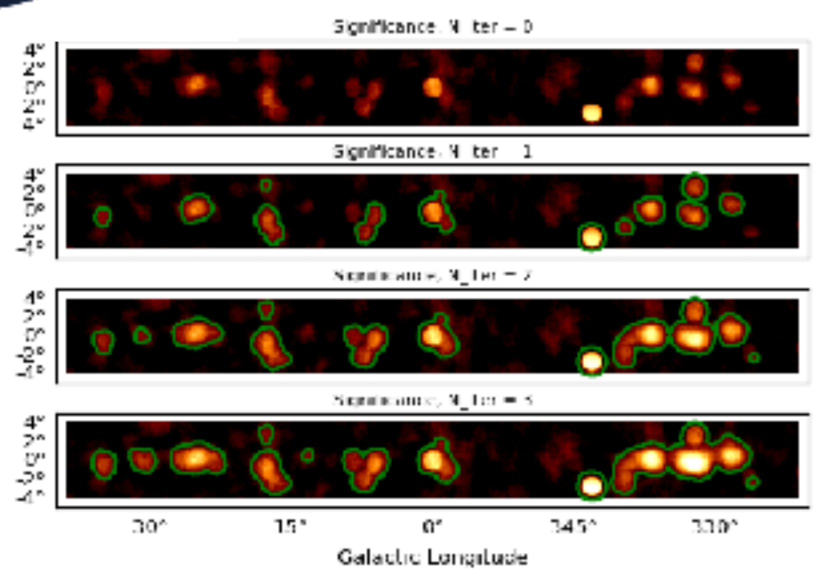
Stage 4: Automated Science

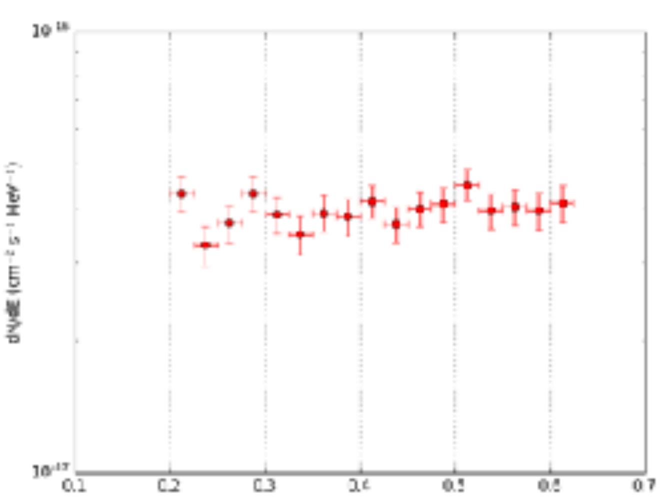
Produce preliminary results for proposal monitoring and alerts

Science Tools

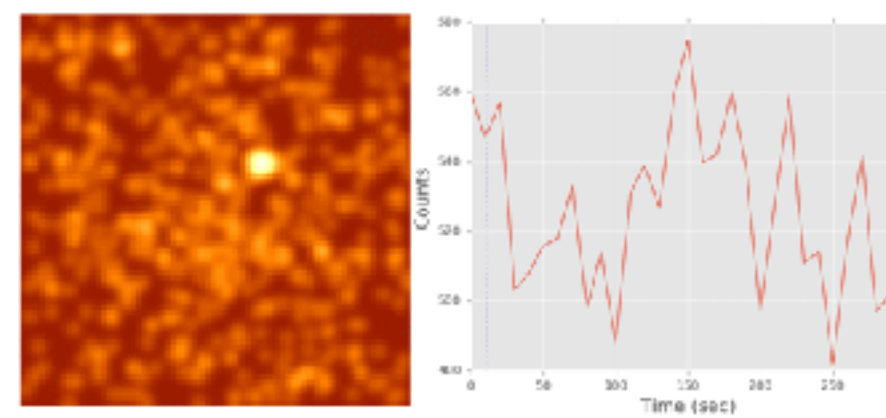
ctools
cherenkov telescope array

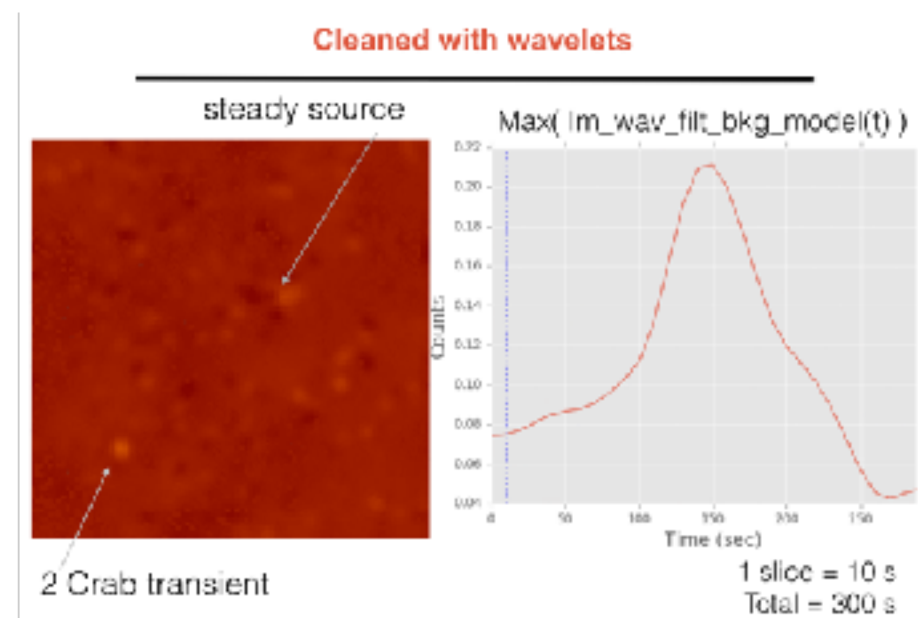
A Python package for gamma-ray astronomy





Custom Tools for Real-Time science and alert generation (on-site)





What will be done for CTA?

Multi-"pass" data production (like Fermi)

- All data re-processed with latest *validated/verified* techniques (reco, calibration) ≈ yearly
- New DL3 data disseminated,

Starting point:

- **basic Hillas Analysis** (good enough to reproduce all CTA requirements). First data release will likely be only this!

Future Improvements:

- Divide events into classes (with each their own IRF)
  - by which telescopes are triggered
  - by reco technique used (may be more than one)
  - etc.
- Fancier image cleaning
- Fancier reconstruction

Implementing a Data Processing Pipeline

Developers in CTA

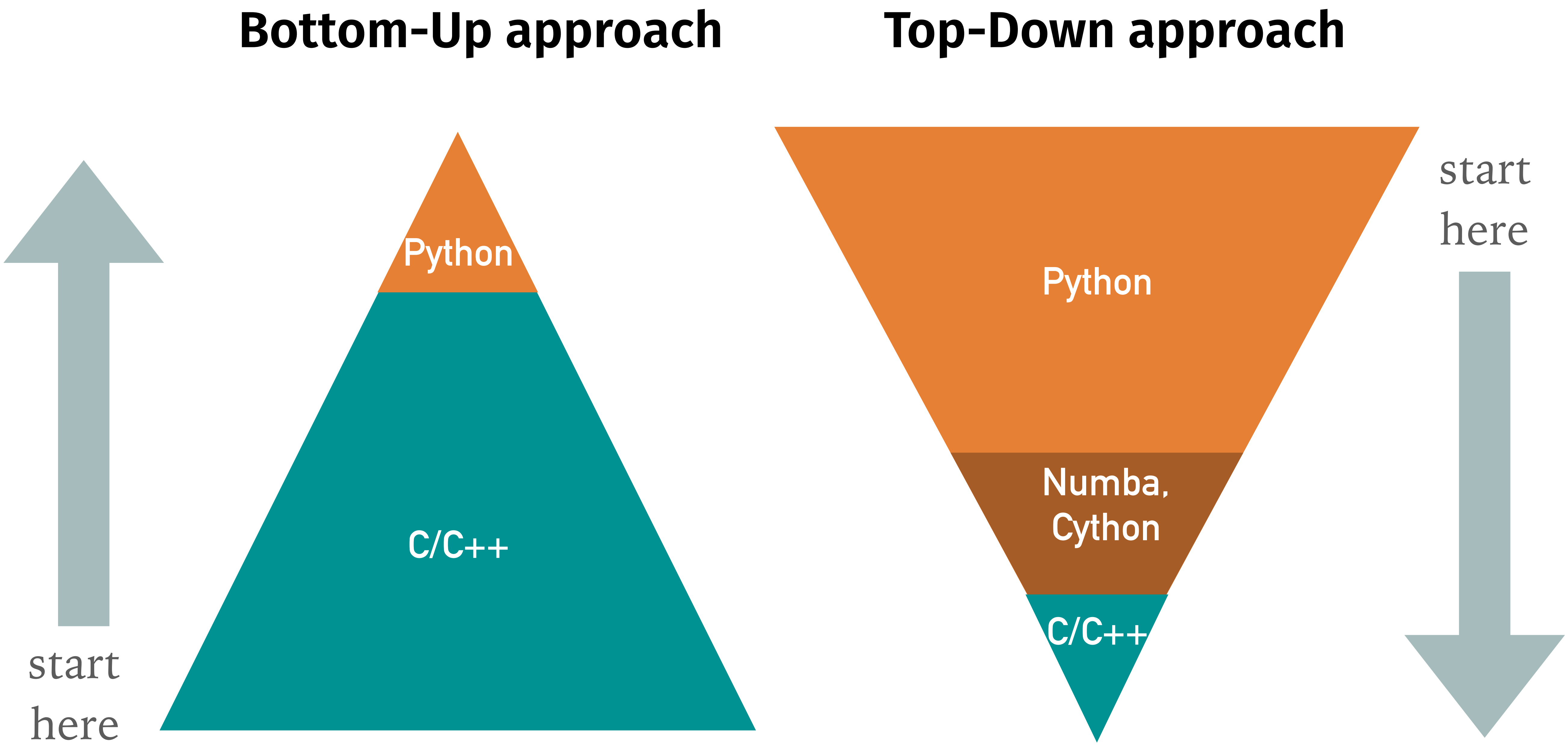
Developers:

- few in number (so far)
- diverse in skill and background, generally non-professionals
- located all around the world
- May pay some professionals for code review and guidance, however

Development constraints:

- want rapid development cycle
- Need strict versioning and release of product and related dependencies
- Need quality control and validation testing (automated as much as possible)

Building a Framework for the Pipeline



Most previous frameworks did it this way

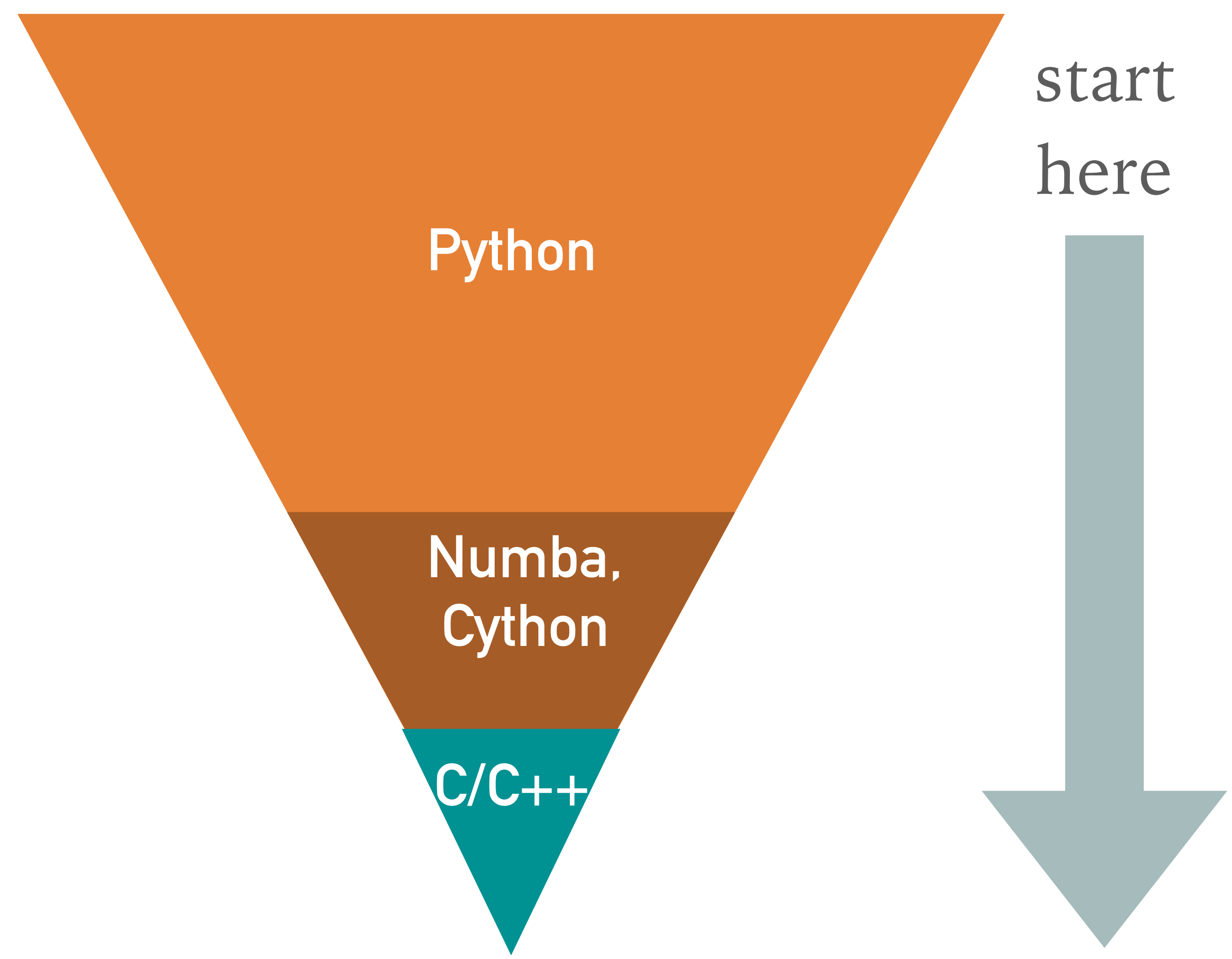
Our approach: start early with python and high-level API

Building a Framework for the Pipeline

Bottom-Up approach

Most previous frameworks did it this way

Top-Down approach

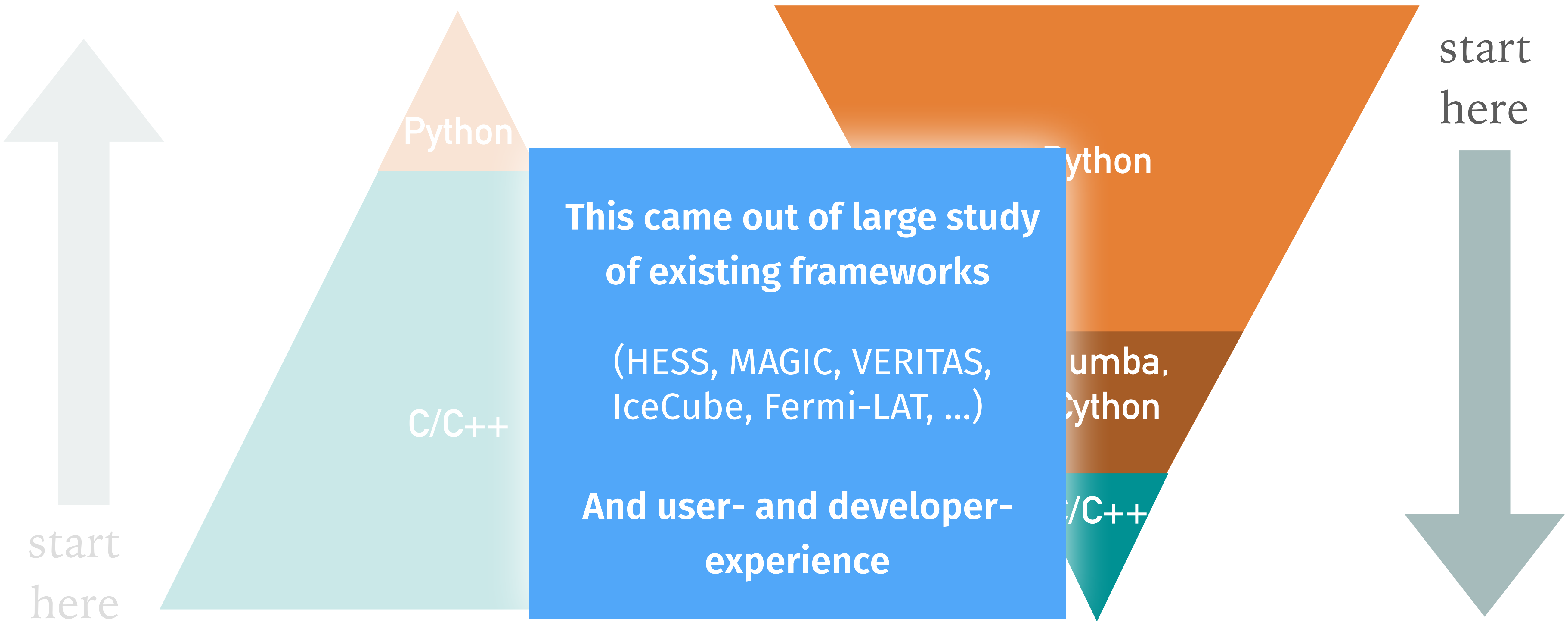


Our approach: start early with python and high-level API

Building a Framework for the Pipeline

Bottom-Up approach

Top-Down approach



This came out of large study of existing frameworks

(HESS, MAGIC, VERITAS, IceCube, Fermi-LAT, ...)

And user- and developer-experience

Most previous frameworks did it this way

Our approach: start early with python and high-level API

ctapipe: data processing prototype

GitHub
repository
issue tracker

TravisCI
continuous integration

github.com/cta-observatory/ctapipe

ctapipe
package/framework

pytest

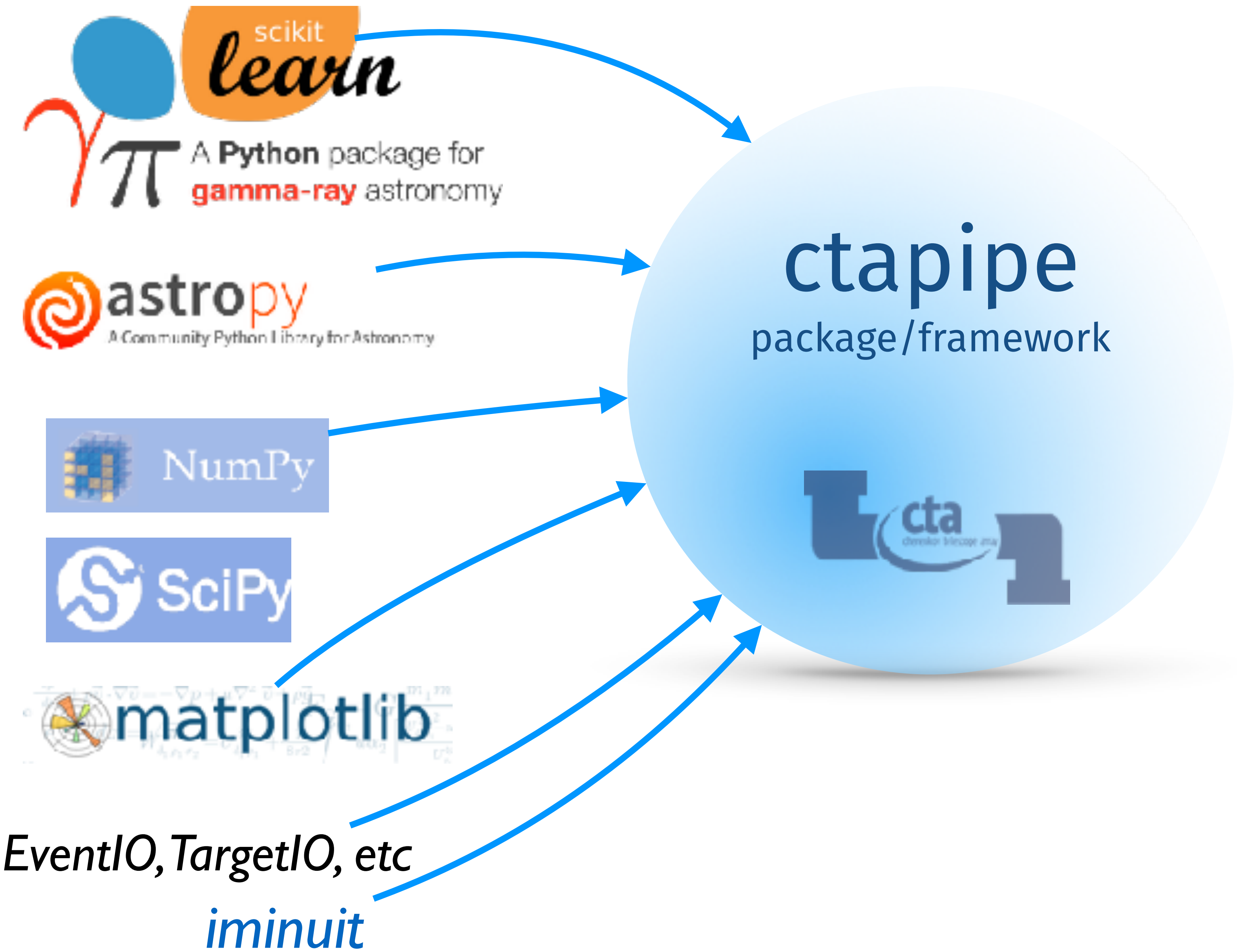
Sphinx
documention

ctapipe: data processing prototype

GitHub
repository
issue tracker

TravisCI
continuous integration

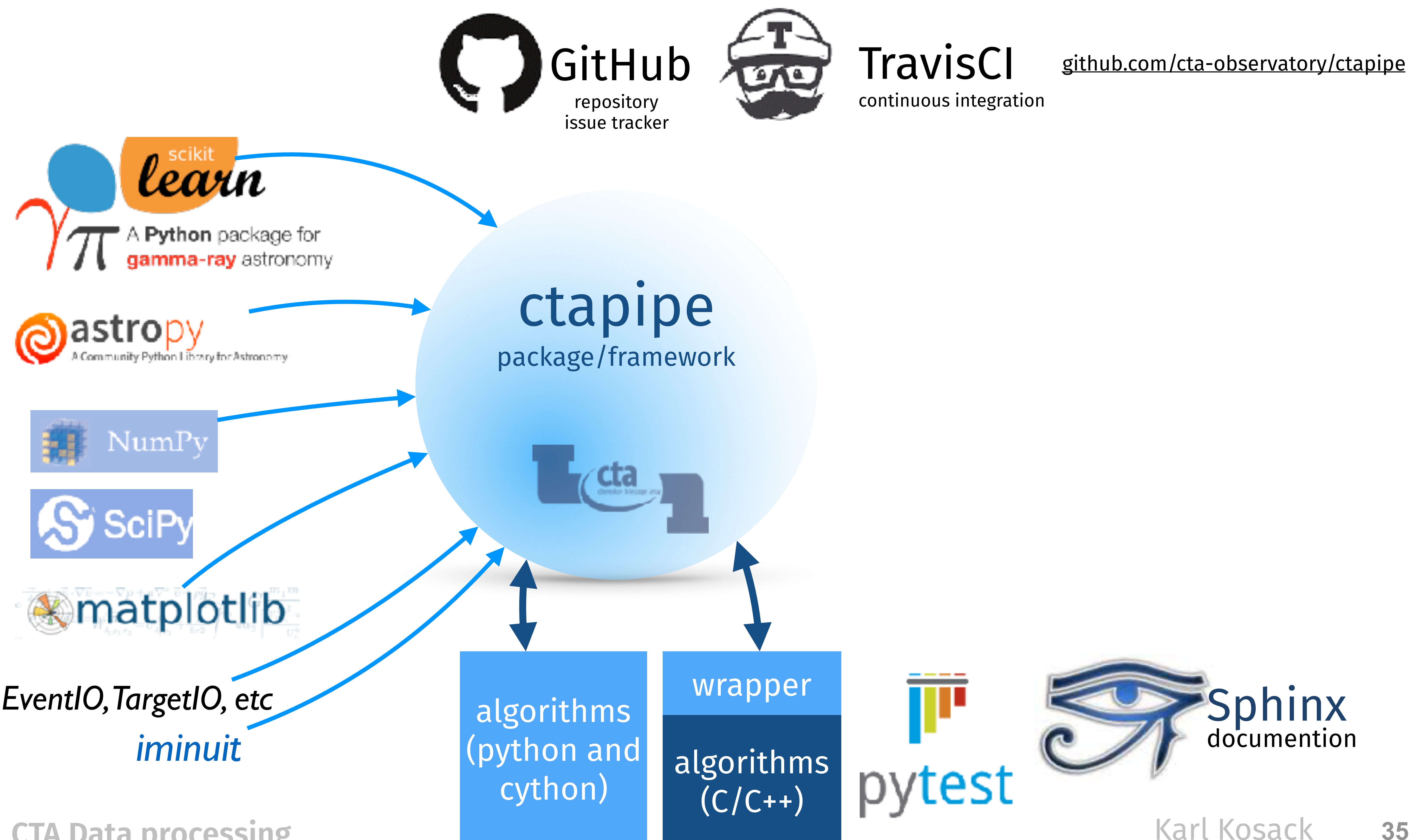
github.com/cta-observatory/ctapipe



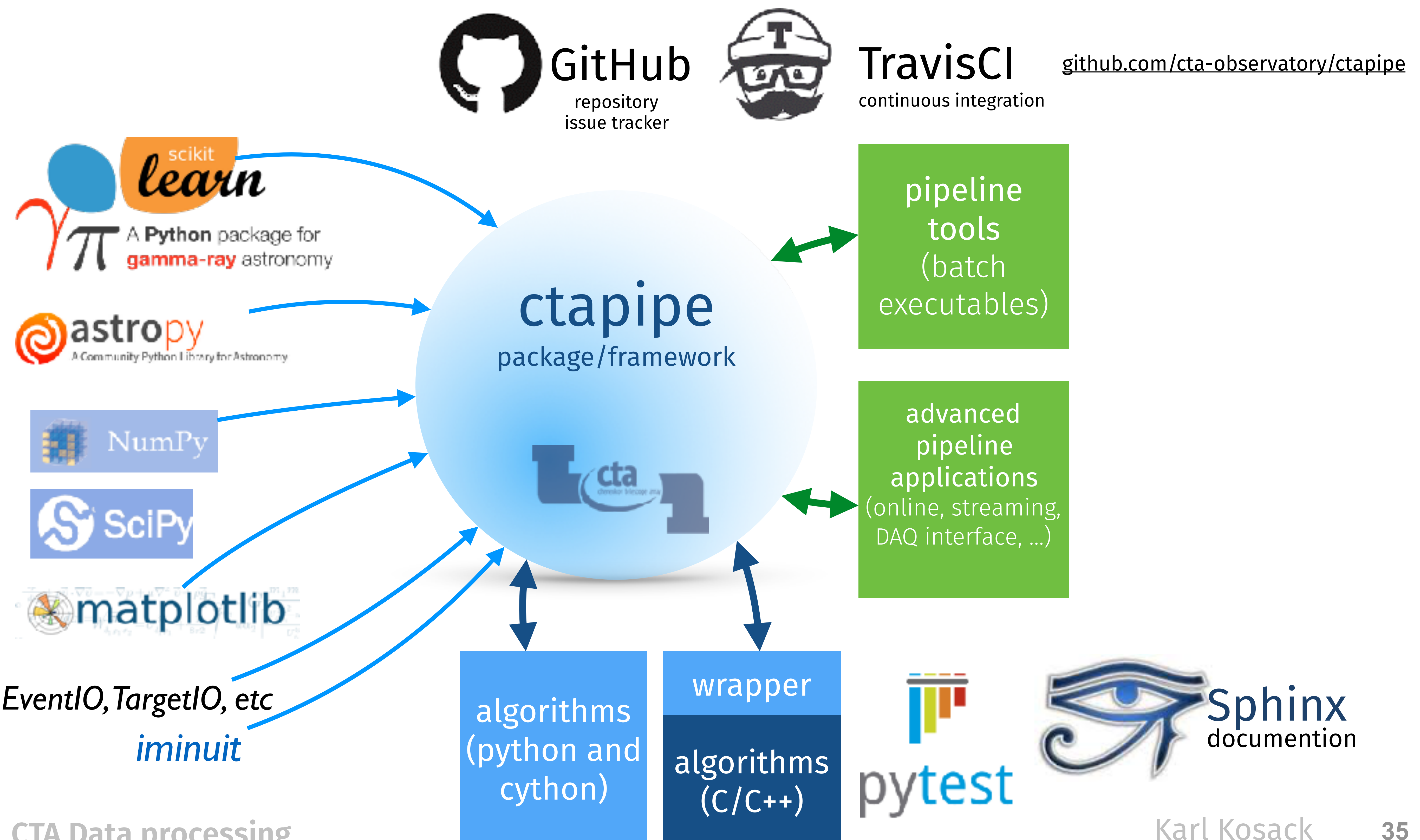
pytest

Sphinx
documention

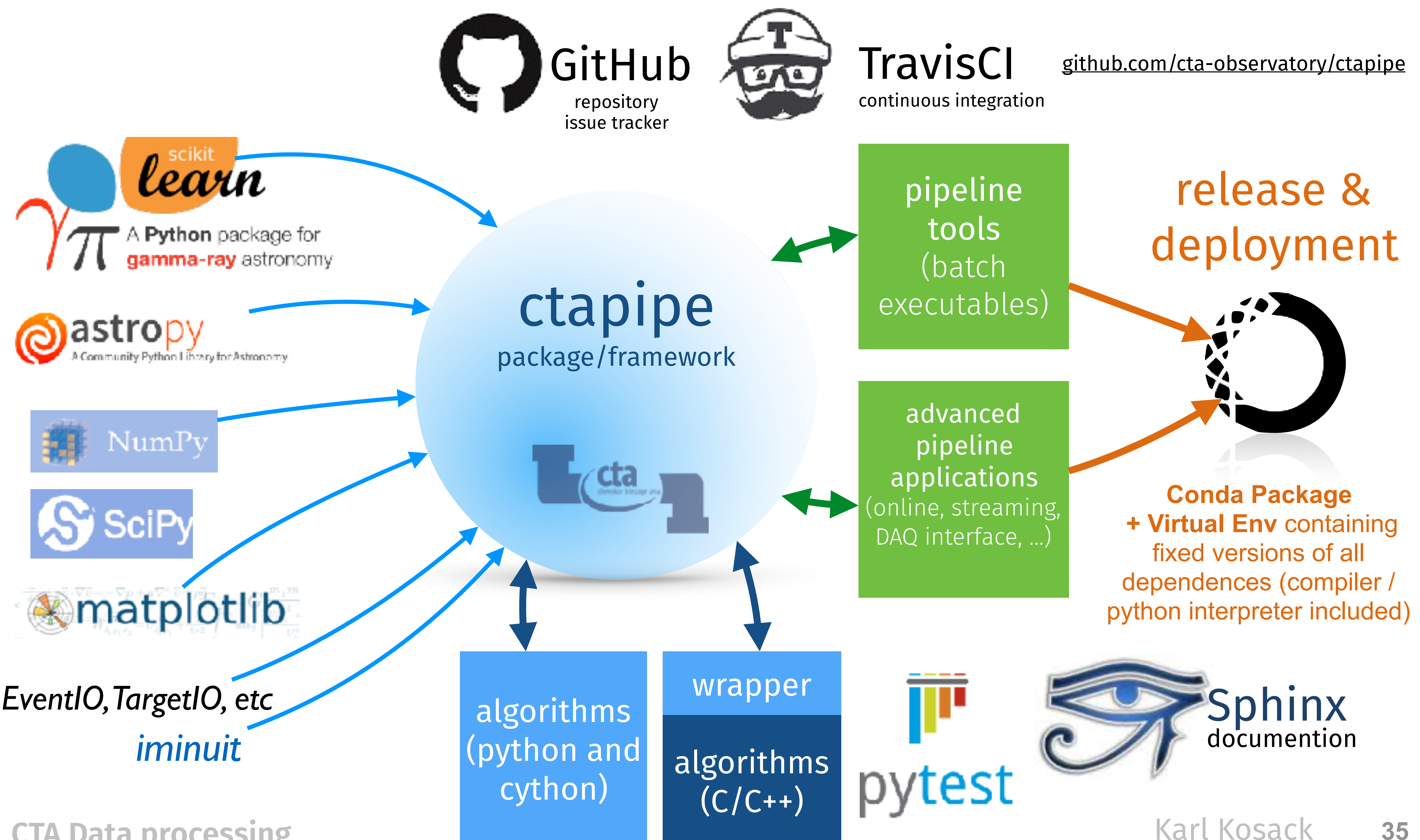
ctapipe: data processing prototype



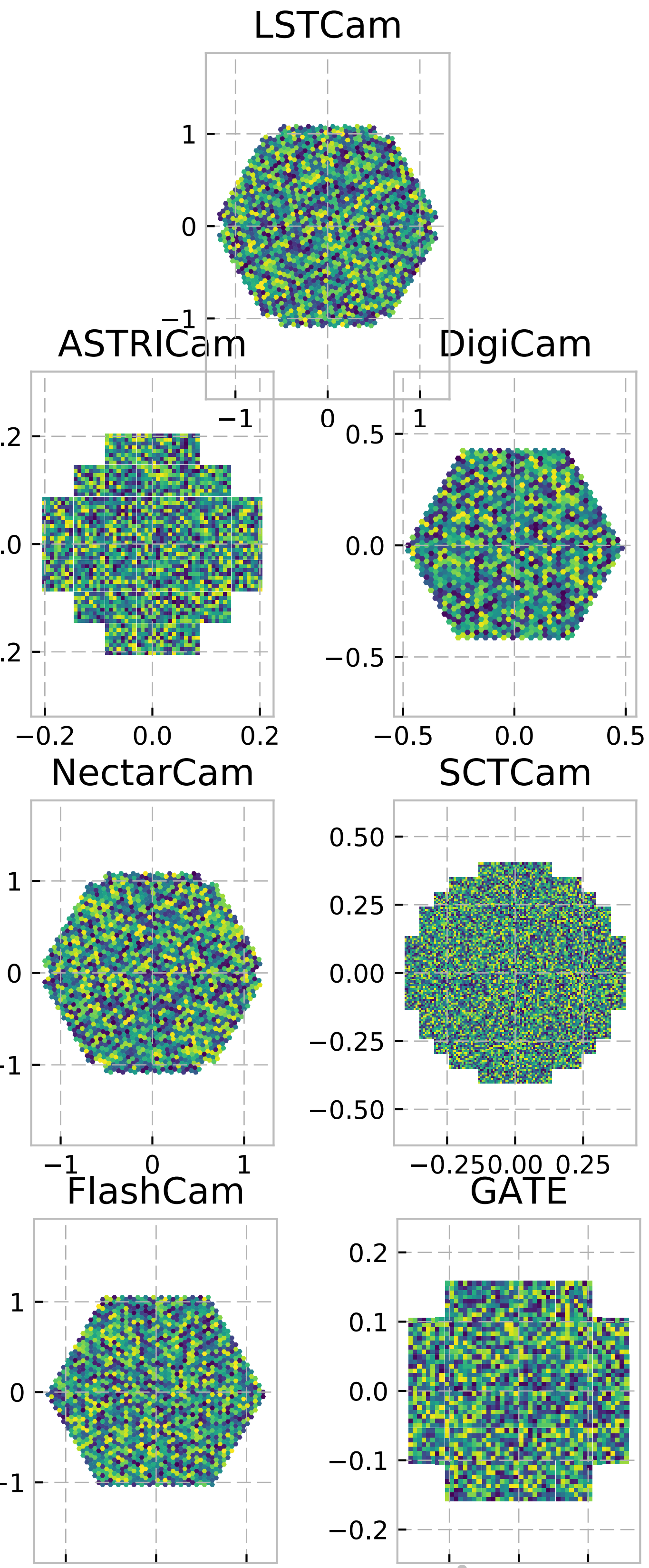
ctapipe: data processing prototype



ctapipe: data processing prototype



ctapipe status



ctapipe: python/C++ based data processing framework

What's there now:

- Basic **Calibration** & **Trace-Integration** for all cameras (several methods available)
- Cherenkov **Image Processing** for all camera geometries (Hillas + Wavelet so far), including Muon analysis
- **Stereo Reconstruction** (plane-intersection + ImPACT model so far)
- **Event Classification** & **Discrimination**
- **Sensitivity Curve Calculation**
- **Visualizations** at each level
- Basic data I/O and provenance tracking

Can be run on the CTA Grid (releases in virtual envs in CVMFS)

Will be phased in to replace existing MC studies this year

Also used by several camera groups for testbed (GATE, DigiCam)

See: github.com/cta-observatory/ctapipe for more info

Working successfully,

Smallish team (<20 developers, only ≈5 very active), and all basic features done after about 1 year of work

Still lots of development to go

Workflow Management

Pipeline must be executed on distributed computing systems (EGI Grid, etc).

- may be broken in to many steps and parallelized in various ways
- implies complex job management, job dependency tracking, monitoring
- may be simple batch system or more complex big-data solution

Current prototype:

- DIRAC middleware with custom CTA front-end
  - used for Monte-Carlo successfully
  - complex data-driven workflows are still experimental
- Some "big-data" systems (Spark, Storm, etc) being tested for possible use on-site

The challenge

Conceptually, data processing is well understood and we have several reference implementations...

What is more difficult is verification and quality:

- does the real instrument perform as simulated?
- Which algorithms are the most sensitive and least affected by systematics?
- How can be better understand/mitigate the effects of atmospheric variation?
- how to deal with data and hardware problems?
  - MCs have no non-uniform NSB, stars, non-working pixels
  - from experience: *lots of unexpected things to learn about the hardware, even 10 years after start*!
  - Bright sources (low hanging fruit) less affected, but start to see issues when we do deep observations (e.g. Key Science Projects!)

This will be the bulk of the data processing work!
Need lots of help.

Making a coherent system:

CTA Architecture Modeling

Why make an architectural model?

Connect Requirements to Design

- don't build something unnecessary
- don't forget something important

Define Common View of full system

- clear boundaries and responsibilities
- coherent interfaces
- define scoping and responsibilities (work packages)
- explore staging scenarios
- estimate effort and complexity

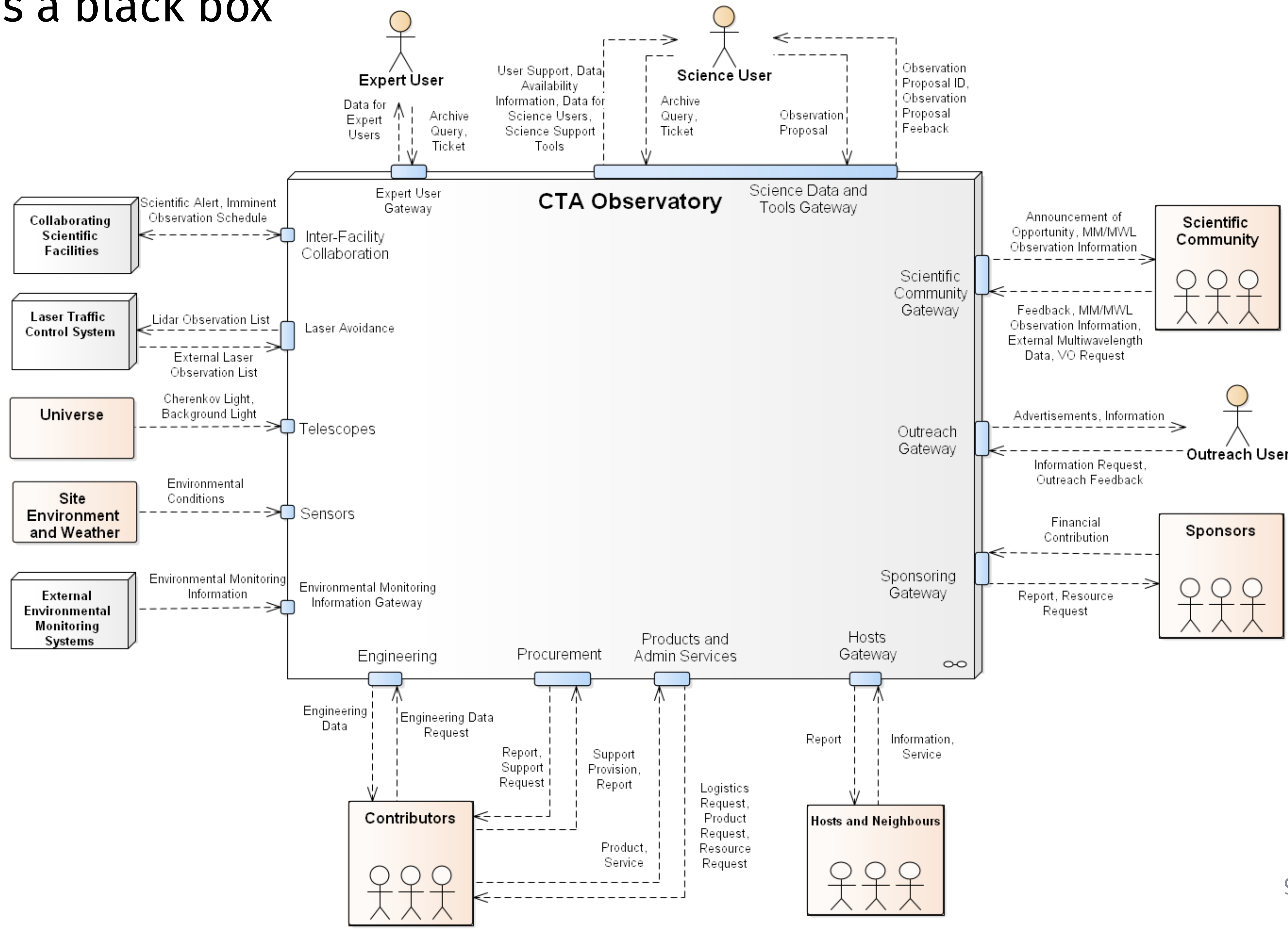
Architecture Team

Team formed by CTAO gGmbH following Architecture Plan

- **Team Leader:** M. Füßling (CTAO)
- **Primary Stakeholder Representative:** J. Hinton (MPIK, CTAO)
- **Architecture Consultant:** L. Hagge (DESY)
- **Modeling Expert:** I. Oya (DESY)
- **Additional CTA Experts:** K. Kosack (CEA), N. Neyroud (CNRS), G. Tosti (INAF)
- **CTAO Systems Engineers:** F. Dazzi and A. Mitchell.
- **External Consultants:** A. Morgenstern (IESE), D. Rost (IESE)

CTA System Context

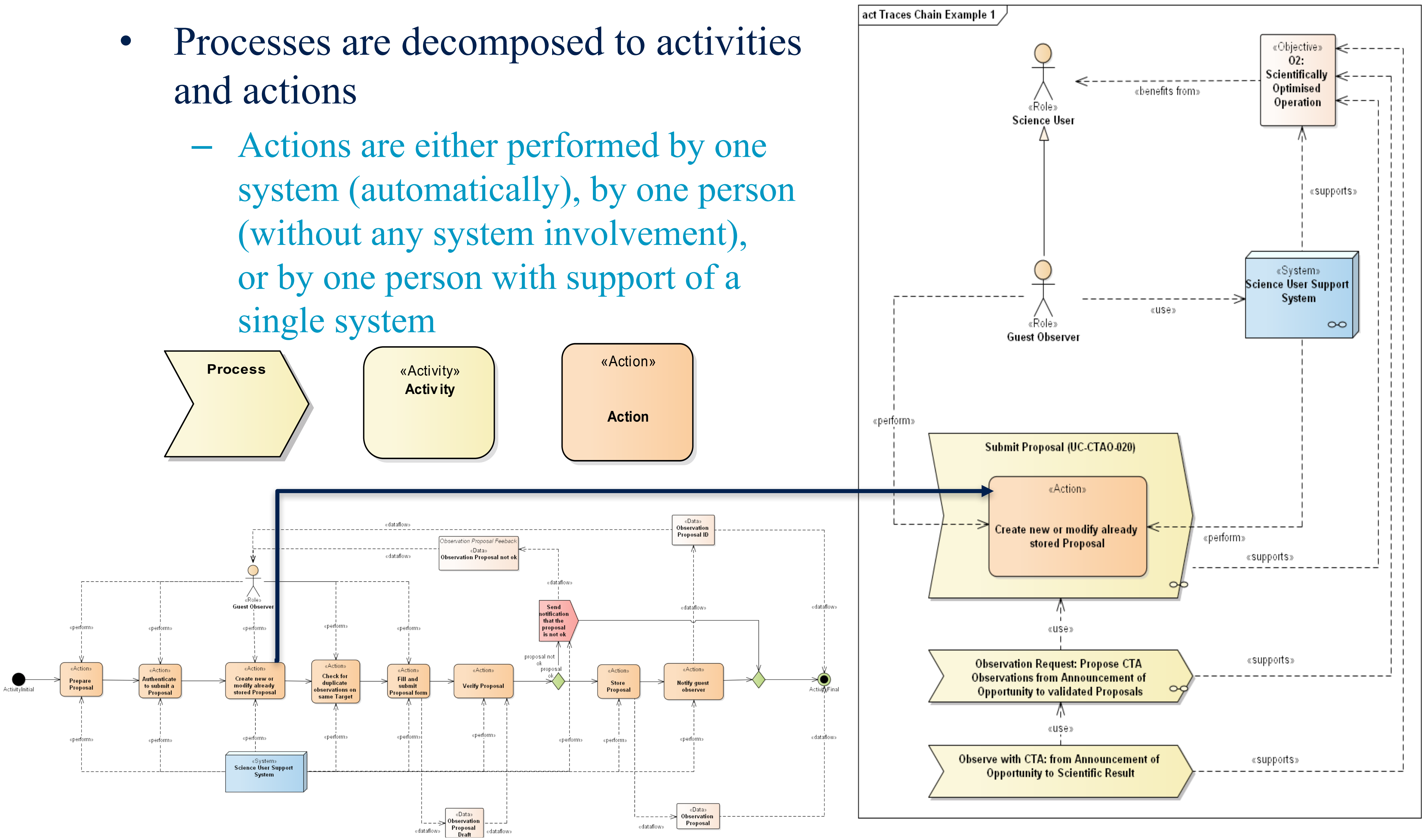
CTA as a black box



Source: Matthias Füßling, CTA architecture team

Process Modeling

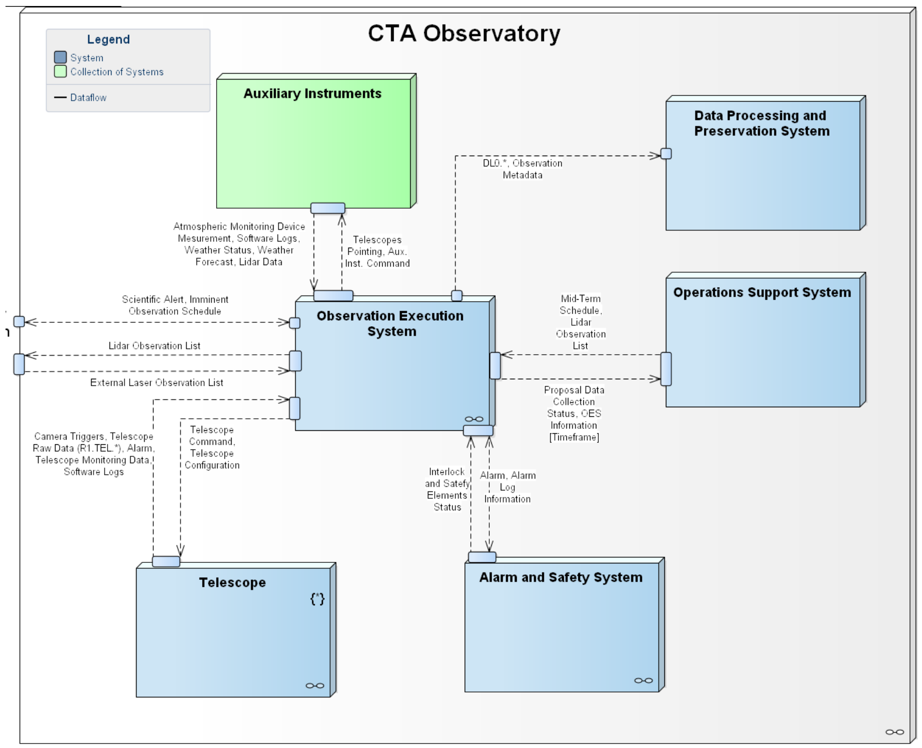
- Processes are decomposed to activities and actions
  - Actions are either performed by one system (automatically), by one person (without any system involvement), or by one person with support of a single system



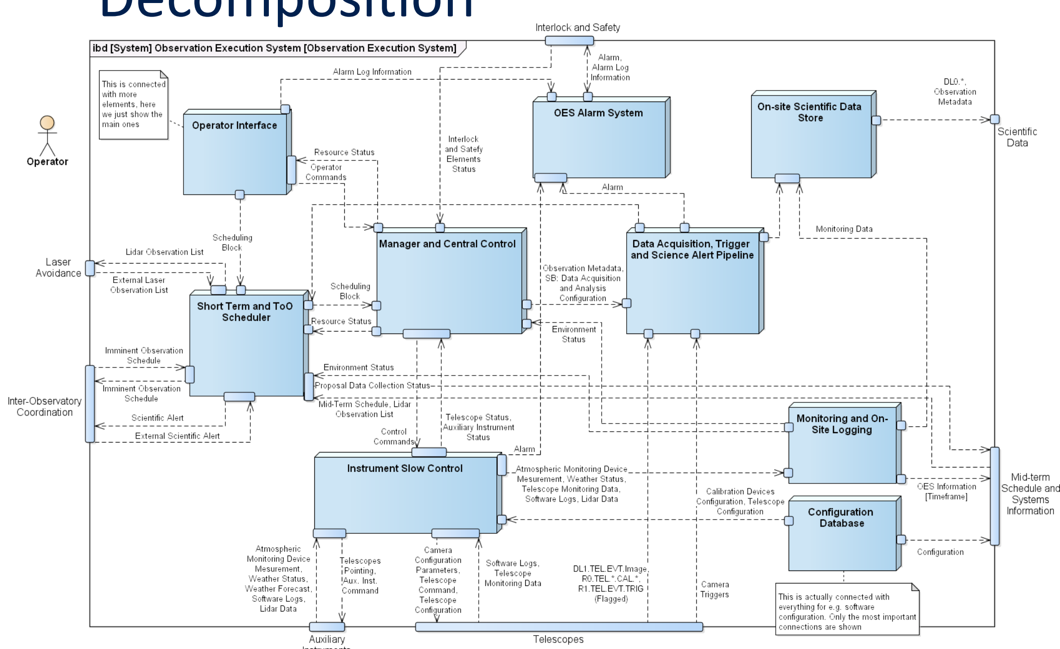
Source: Matthias Füßling, CTA architecture team

Defining Systems

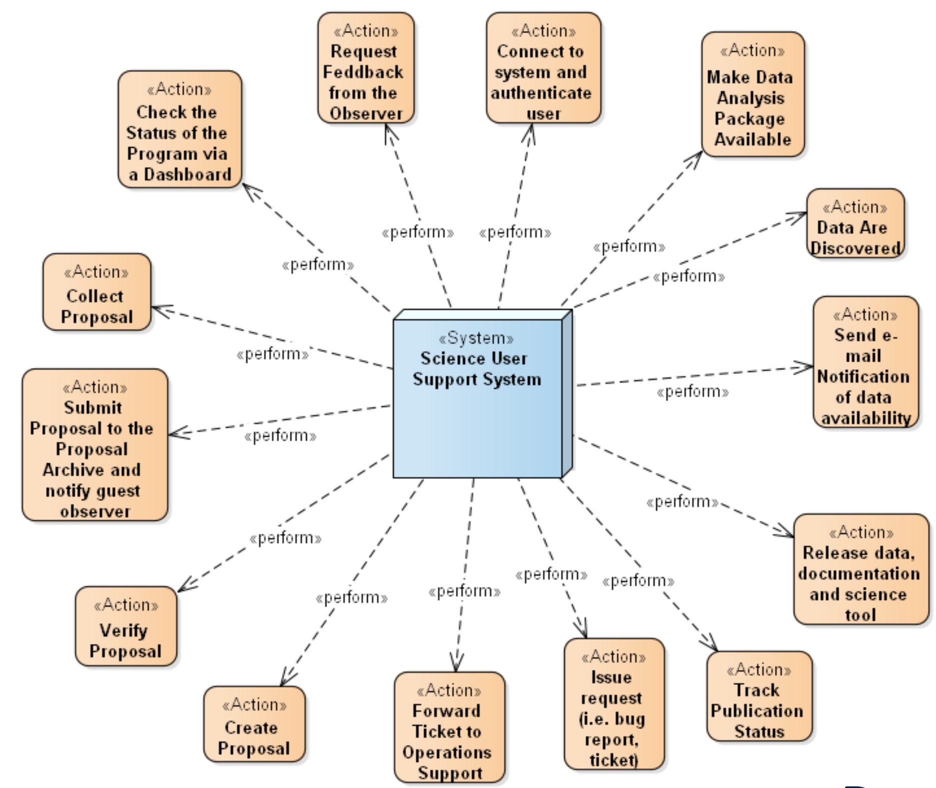
Context



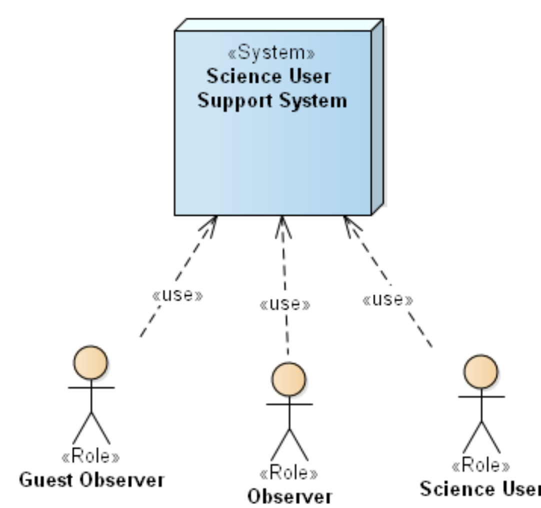
Decomposition



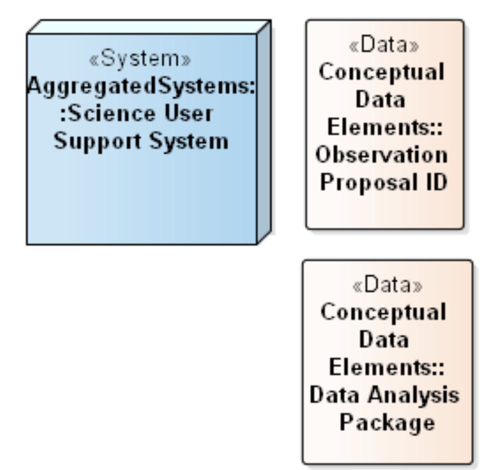
Functionality



Users

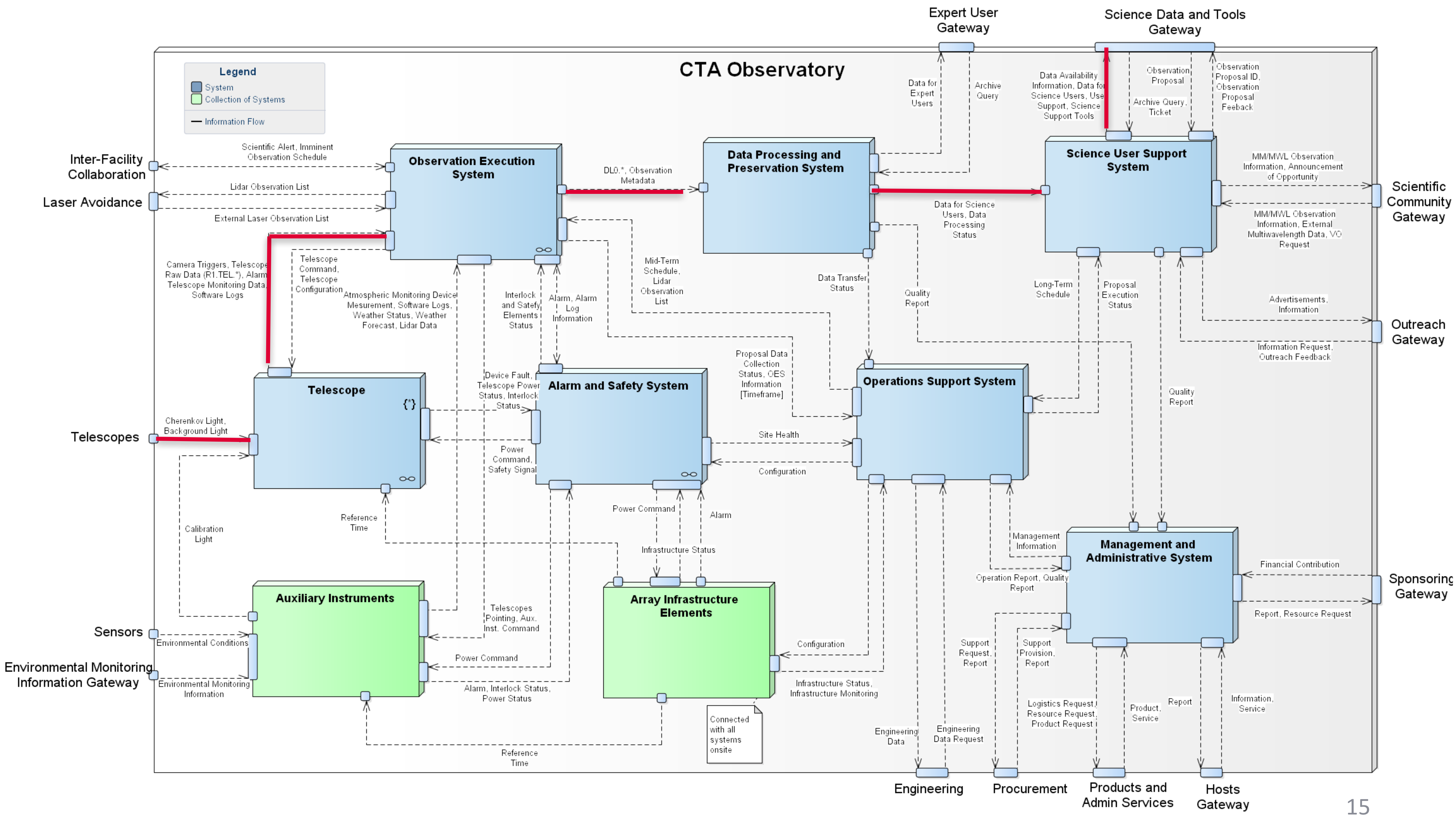


Data



Source: Matthias Füßling, CTA architecture team

Inside the box



Source: Matthias Füßling, CTA architecture team

Next steps before construction

- High-level systems will replace current working packages
- IKC agreements for systems and subsystems
- internal model for each **System**, by each work package
- Identify groups/institutes to produce each sub-system
- implementation!

Extra info, if there is time:

Modern Development Practices:

How to maintain and produce good code

CTA SYS group and PO are preparing

detailed guidelines for software in CTA

cherenkov telescope array

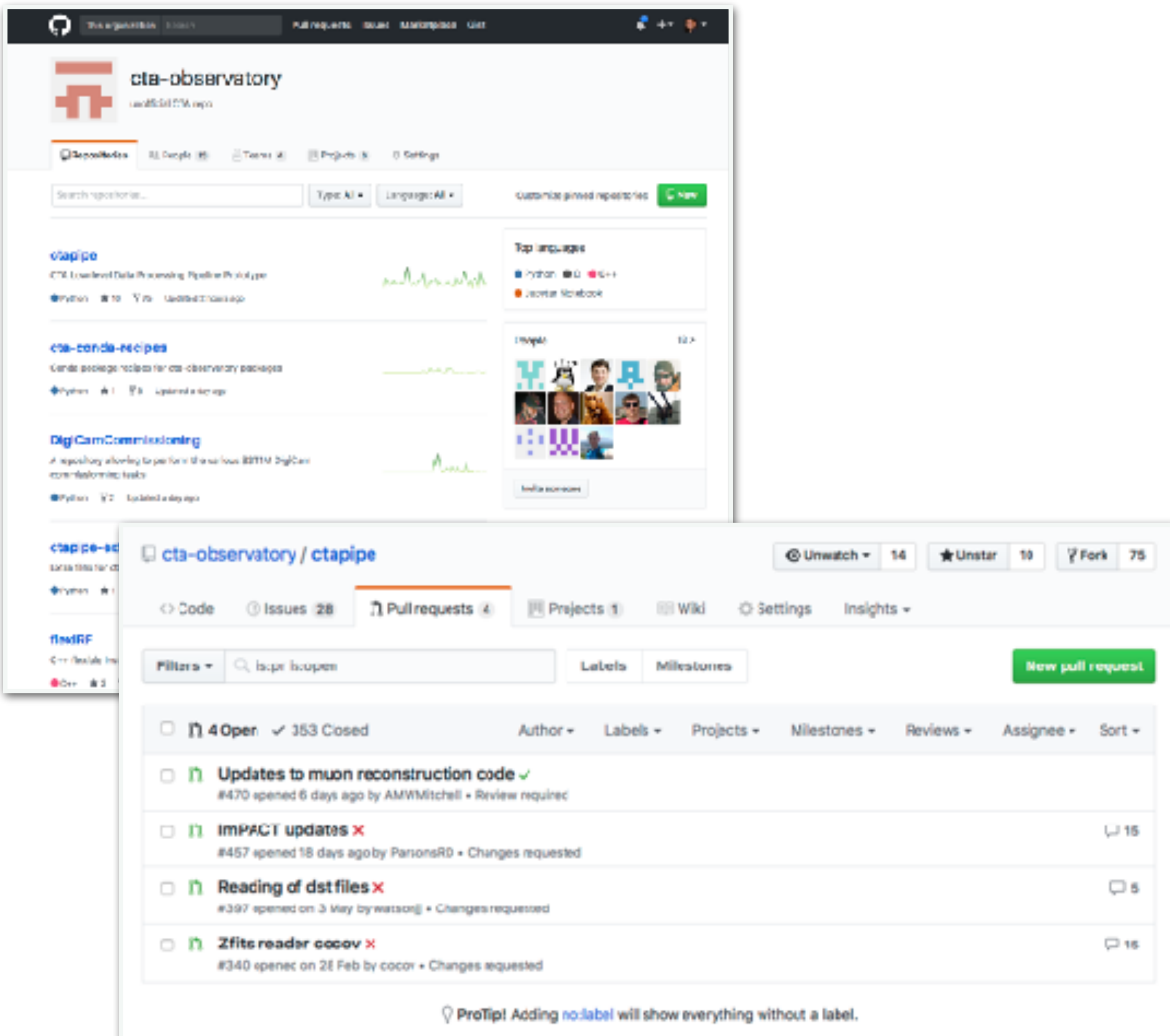
Programming Standards

cherenkov telescope array

Software Development Plan

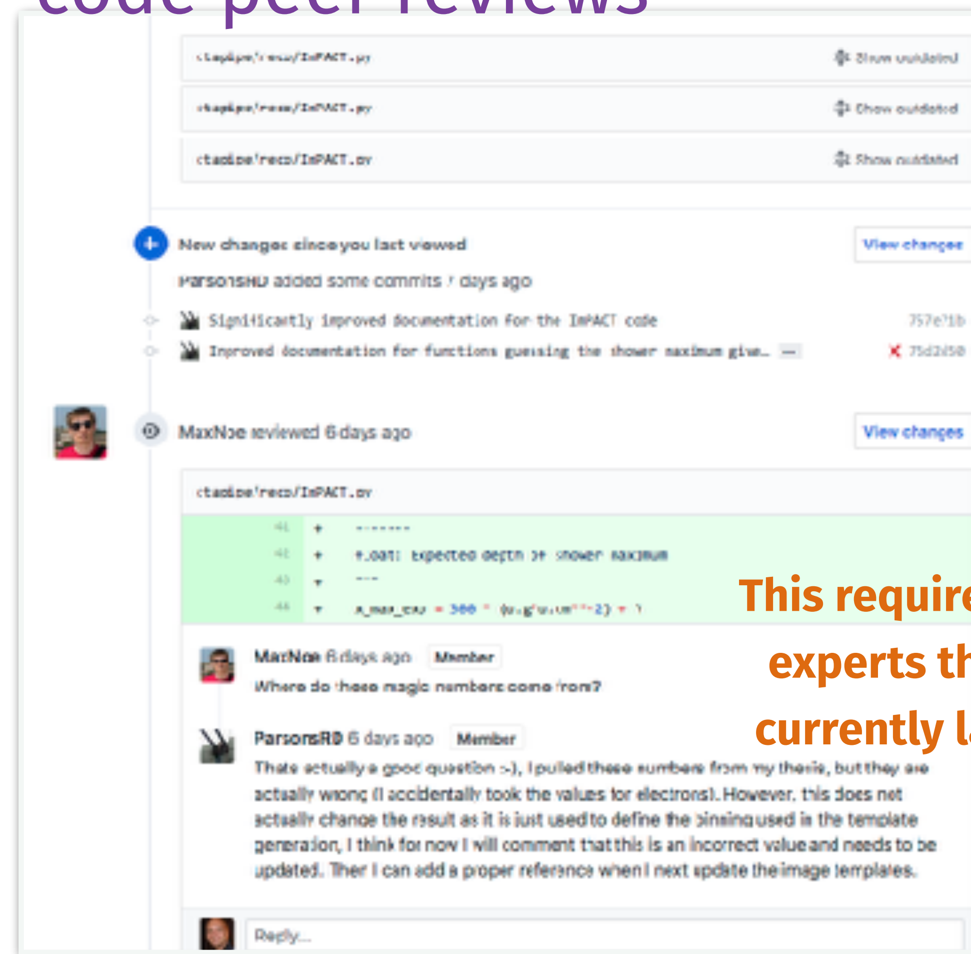
Code development

GitHub-Flow dev model



Issues and Pull-Requests from developer forks

code peer reviews



This requires many experts that are currently lacking!

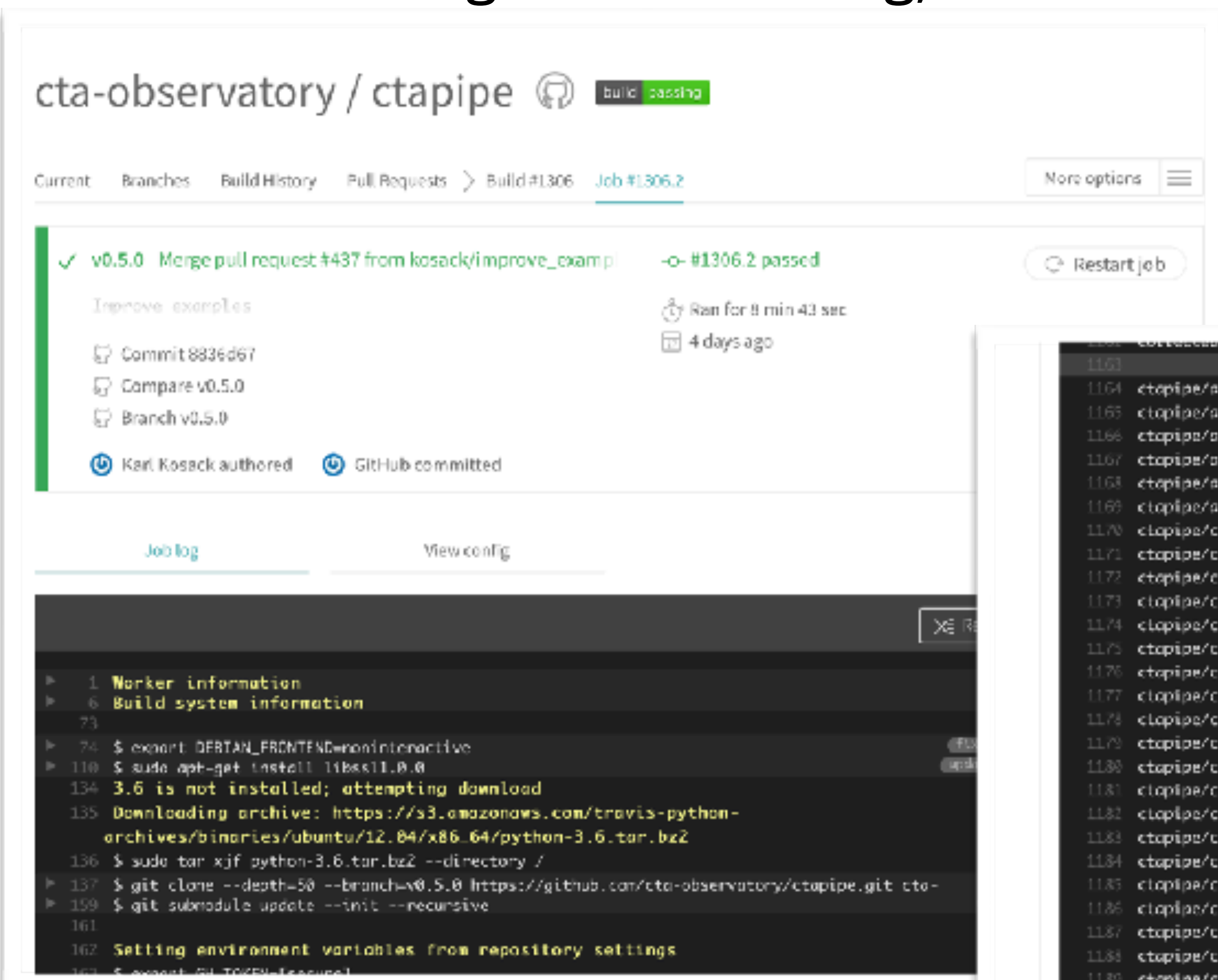
Required for Merging **Pull-Request into master:**

- **Code review** by somebody other than submitter
- all **tests pass**
- **code quality** stays same or increases

github.com/cta-observatory/ctapipe

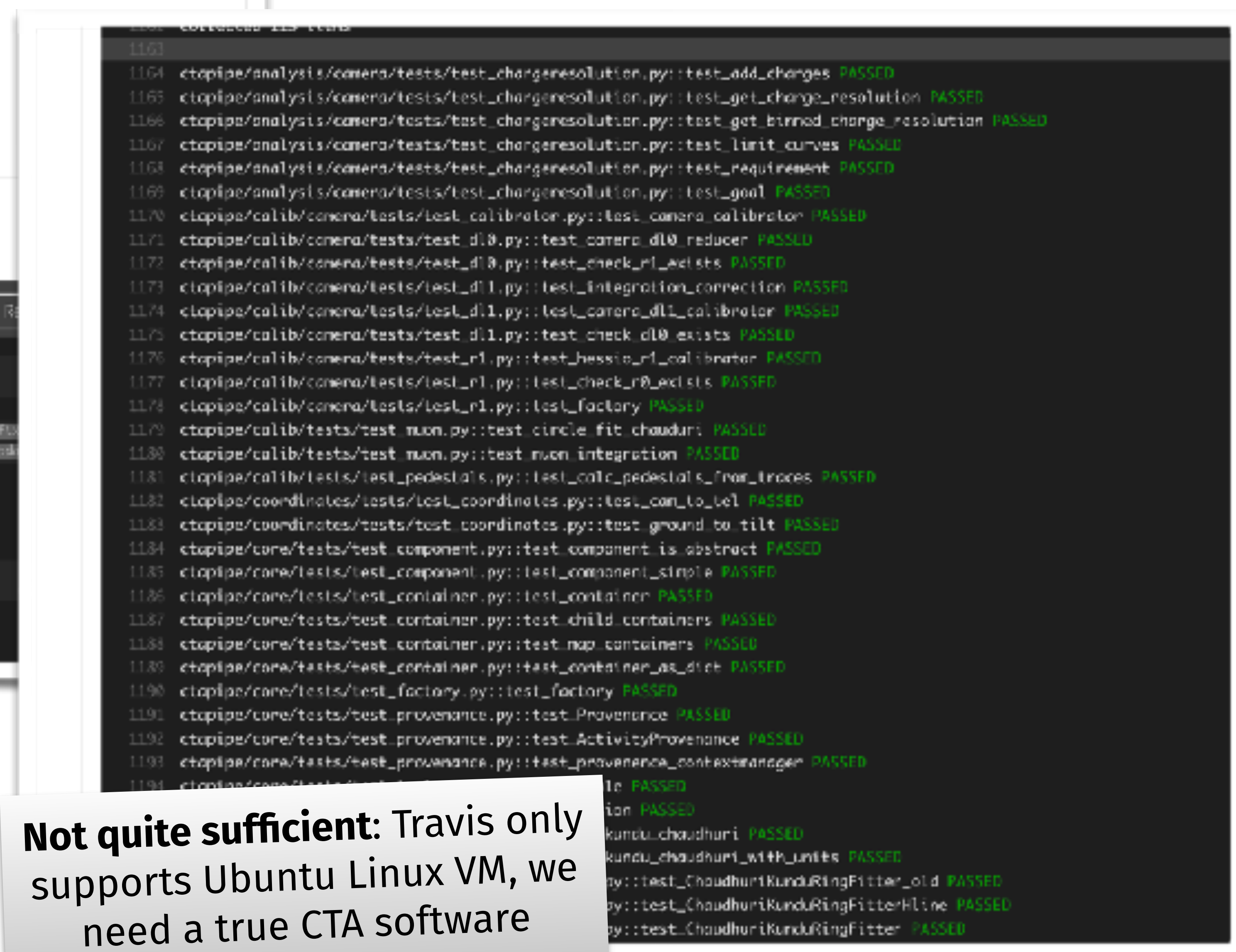
Continuous Integration

Unit and Regression Testing, Documentation deployment



Not just *"does it compile?"*, but *"does it (still) work?"*

note: a CTA software **requirement** to have this existing code *must* also add a unit test suite



pytest TravisCI

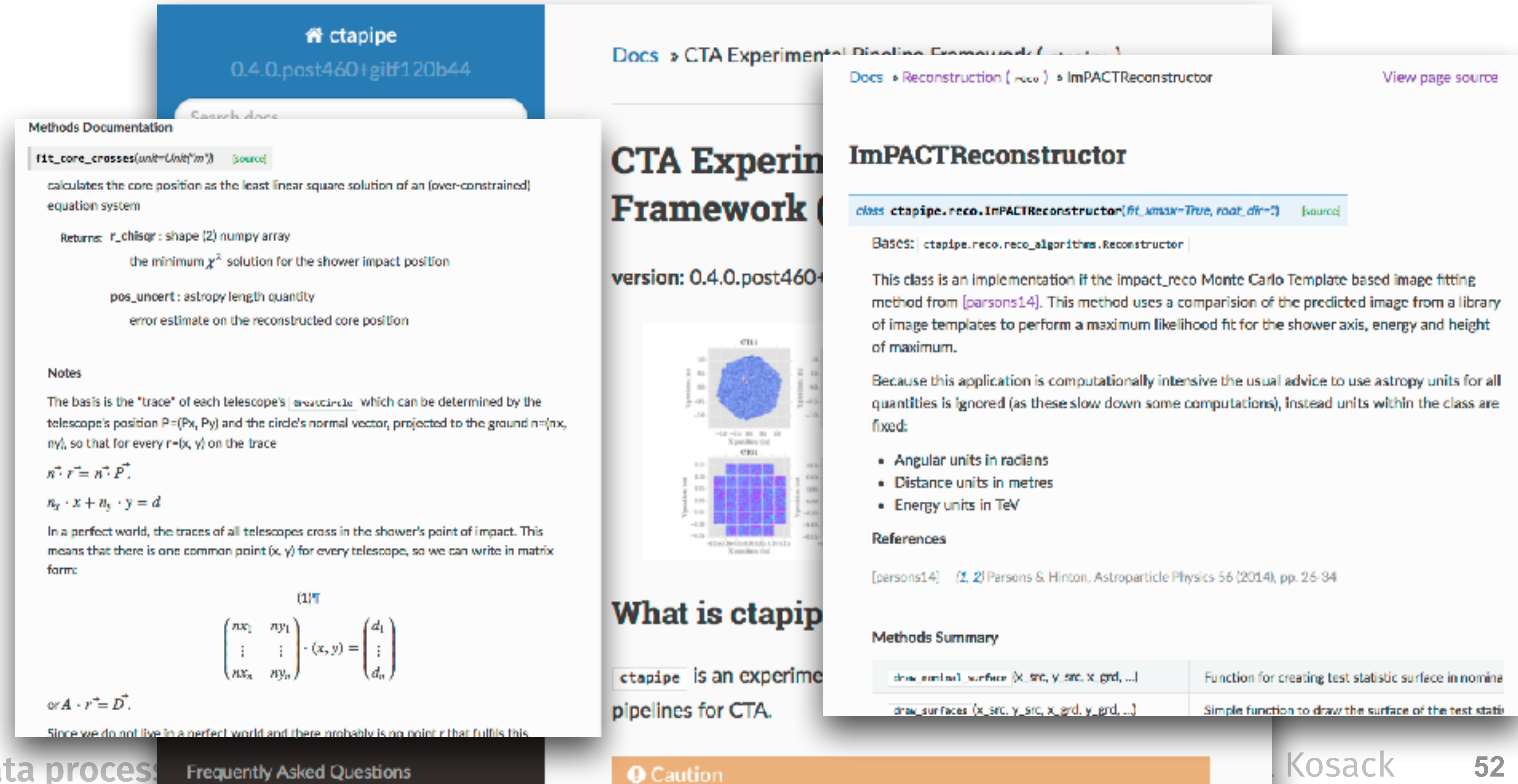
Not quite sufficient: Travis only supports Ubuntu Linux VM, we need a true CTA software integration system

Documentation

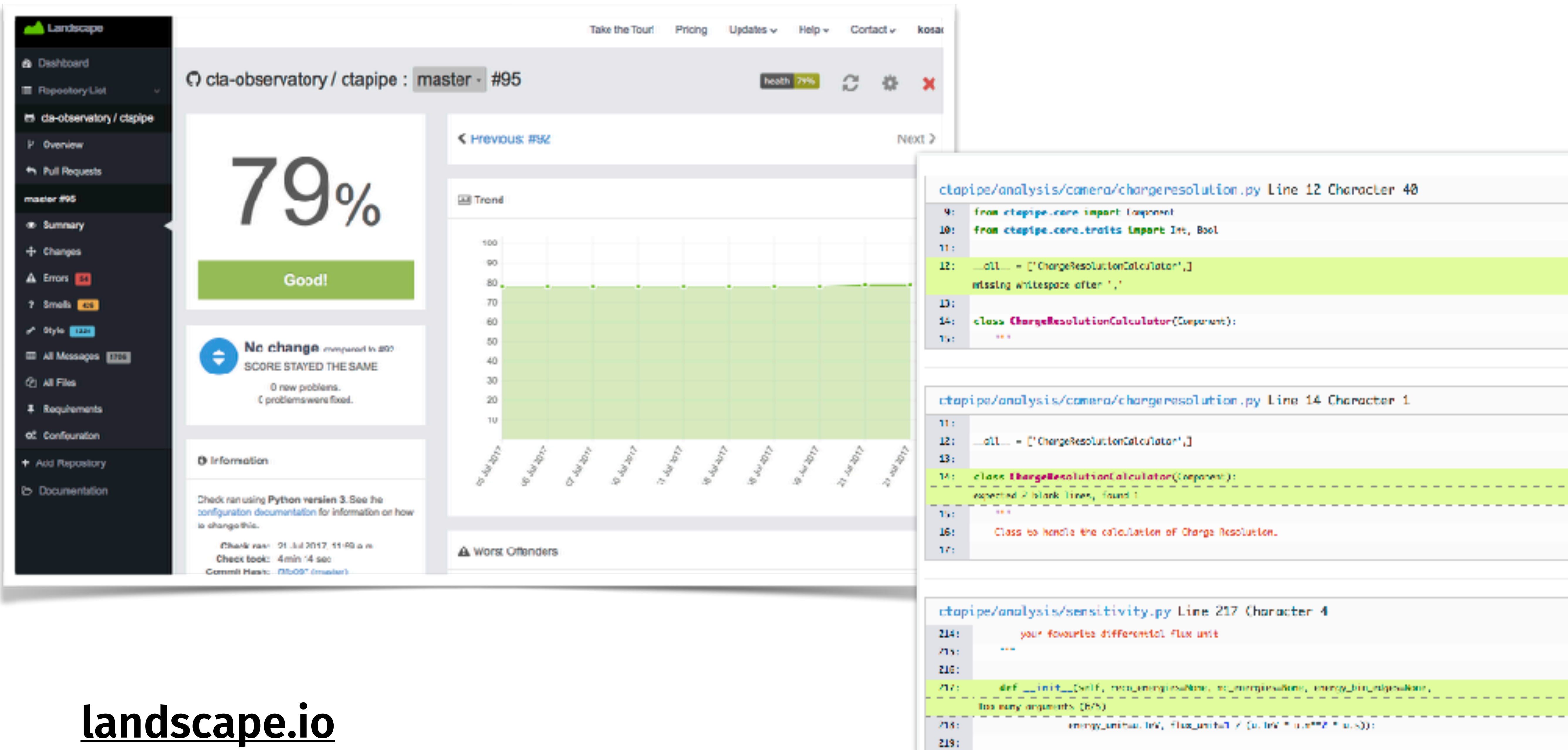
e.g. https://cta-observatory.github.io/ctapipe/

regenerated automatically on commits/merges to *upstream master*

→ Always up-to-date and in sync with code!



Code Quality Monitoring



landscape.io

- open-source, free cloud system for open-source projects
- **Integrates with GitHub** (or GitLab) and can be used to block bad code from being committed.

Deployment

Anaconda cross-platform package manager

- build "conda" packages for each module (mac + linux)
- handles dependencies and virtual-environments that are **self-contained for each release**
- locally-installed virtual envs on CTA Grid in CVMFS shared filesystem
- packages currently hosted on *Anaconda Cloud* in *cta-observatory* channel
  - again *only for public software*, but can host our own private channel locally (no resources for that currently!)

```
$ conda env create -n cta-v1 python=3.6
$ source activate cta-v1
$ conda install -c cta-observatory ctapipe=0.5
```

Deployment

Anaconda cross-platform package manager

- build "conda" packages for each module (mac + linux)
- handles dependencies and virtual-environments that are **self-contained for each release**
- locally-installed virtual envs on CTA Grid in CVMFS shared filesystem
- packages currently hosted on *Anaconda Cloud* in *cta-observatory* channel
  - again *only for public software*, but can host our own private channel locally (no resources for that currently!)

```
$ conda env create -n cta-v1 python=3.6
$ source activate cta-v1
$ conda install -c cta-observatory ctapipe=0.5
```

Final remarks

Lots of software under development and to be developed for CTA! Only touched on some here...

A few places where people are needed (probably lots more):

- Data Volume Reduction:
  - needs a dedicated team to study and implement
  - Some work started (e.g. LAPP)
- Data Pipeline:
  - algorithm study and development
  - good coders to help with code quality and design
  - develop data quality monitoring techniques
  - verification of algorithms, simulations, and calibration (a continuous effort)
  - parallelization and speed
  - study factorization of IRFs to improve science results (event classes, etc)
- Monte-Carlo:
  - configuration builder,
  - improve algorithms and software
  - implement "runwise" Monte-Carlo to simulate real observation conditions
- Science Tools:
  - verification and monitoring of science results
  - development of better algorithms
- Infrastructure:
  - Common database system and interface for instrumental configuration (used by pipeline, MCs, array control, hardware teams)