

Experiences & findings in developing common software



The **acts** project

@SaltyBurger



A. Salzburger (CERN) for the ACTS project

supported by



cooperations



Disclaimer

- **Main focus** of this talk
 - ▶ Lessons learnt from/by developing common software
 - ▶ **Concepts, benefits & costs** of doing so

- Recent talks on **ACTS status & performance**



[Ai X, Tracking with ACTS, CTD2020](#)



[Schlag B, ACTS Vertexing and Deep Learning Vertex Finding, CTD2020](#)

Superficial history of track reconstruction SW

1987

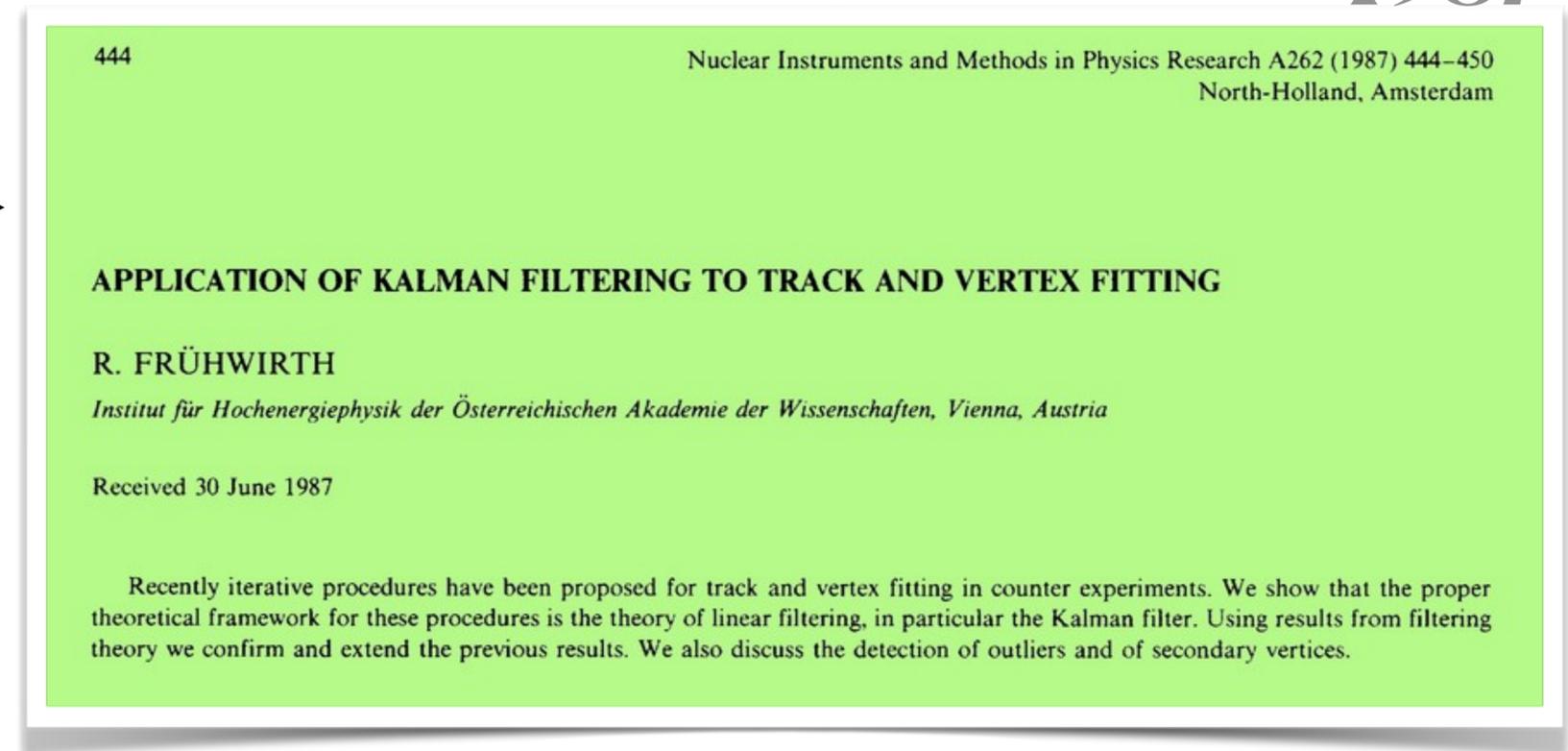
- Many of current algorithms have roots in **LEP era**:

- ▶ Code base usually in FORTRAN

- Move to C++ in early 2000's

- ▶ Common C++ based libraries

- ▶ Era of object oriented design



[R. Frühwirth, Application of Kalman Filtering To Track and Vertex Fitting \(1987\)](#)

“The Delphi detector was designed for the Kalman Filter”

Superficial history of track reconstruction SW

1991

- Many of current algorithms have roots in LEP era:

- ▶ Code base usually in **FORTRAN** →

- Move to C++ in early 2000's

- ▶ Common C++ based libraries
- ▶ Era of object oriented design

“For all its inelegance, and lack of safety features, it seems certain that FORTRAN will remain the main language for HEP code well into the 1990s....”

Computing at CERN in the 1990s

“FORTRAN is probably the only perennial standard which will never be questioned.”

Trends in Computing for HEP

“I don't know what the language of the year 2000 will look like but I know it will be called FORTRAN.”

Tony Hoare

Superficial history of track reconstruction SW

1994

- Many of current algorithms have roots in LEP era:
 - ▶ Code base usually in FORTRAN
- Move to C++ in early 2000's
 - ▶ Common **C++ based libraries**
 - ▶ Era of object oriented design

CLHEP - A Class Library for High Energy Physics

Shortcuts to: [Documentation](#) [Download](#) [CLHEP editors](#) [Mailing List](#) [CLHEP Workshops](#) [Bug Reports](#)

The **CLHEP project** was proposed by [Leif Lönnblad](#) at CHEP 92. It is intended to be a set of HEP-specific **foundation** and **utility** classes such as random generators, physics vectors, geometry and linear algebra. CLHEP is structured in a set of [packages](#) independent of any external package (interdependencies within CLHEP are allowed under certain [conditions](#)).

A large fraction of contributions (mainly to the Random, Vector, Geometry and Matrix packages) came from using CLHEP within (in alphabetical order):

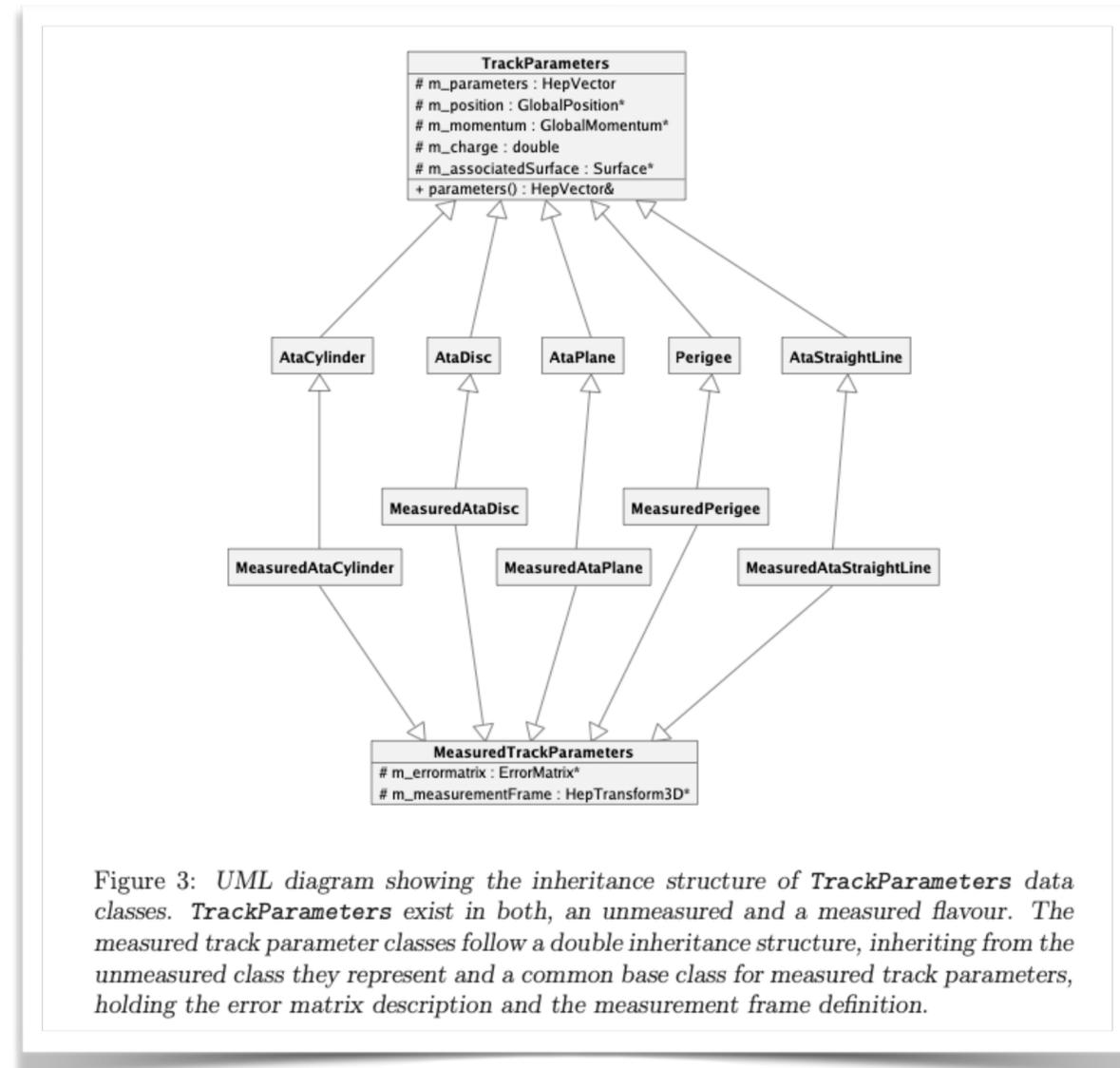
- the [BaBar experiment](#) @ [SLAC](#)
- the [Geant4](#) Collaboration
- the [ZOOM Project](#) @ [Fermilab](#)

[L. Lönnblad, CLHEP—a project for designing a C++ class library for high energy physics \(1994\)](#)

Superficial history of track reconstruction SW

2006

- Many of current algorithms have roots in LEP era:
 - ▶ Code base usually in FORTRAN
- Move to C++ in early 2000's
 - ▶ Common C++ based libraries
 - ▶ Era of object oriented (OO) design



[Akeson, P F et al, ATLAS Tracking Event Data Model \(2006\)](#)

Now we'd call it over-object oriented (OOO) ...

Legacy of the LHC era

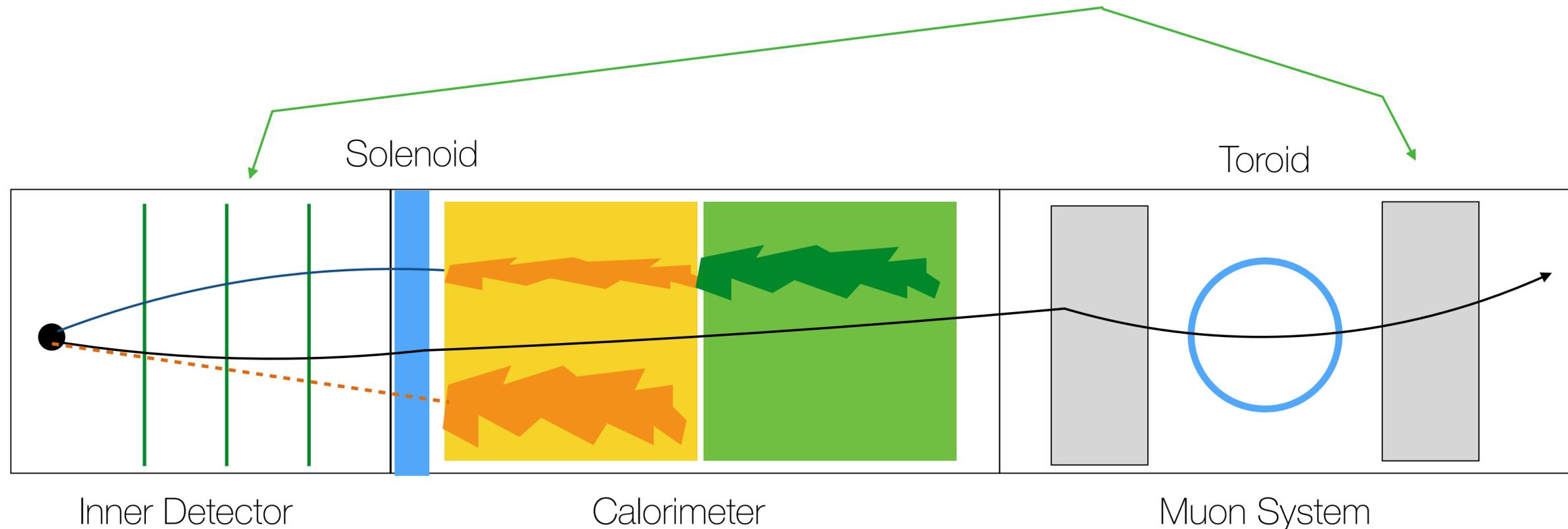
- Extremely ironed-out code
 - ▶ Many 10^{10} events processed (numerically validated)
- Extremely well performing track reconstruction algorithms
 - ▶ Technical track reconstruction (μ) efficiency $\sim 100\%$
- A great starting point!

Future challenges

- HL-LHC / FCC-hh studies
 - ▶ Way higher particle multiplicities (combinatorics)
- The rise of new technologies
 - ▶ C++20 has little to do with C++98
 - ▶ Heterogeneous computing
- Robots/AI want to take our jobs!

A great starting point

- ATLAS Track reconstruction SW
 - ▶ Designed/written with common (agnostic) top layer for **two** tracking devices



- ▶ Highly polymorphic Tracking reconstruction SW, with common abstraction layer

[Cornelissen T et al, Concepts, Design and Implementation of the ATLAS New Tracking, 2008](#)

A great starting point

- ATLAS Track reconstruction SW
 - ▶ Designed/written with common (agnostic) top layer for **two** tracking devices
 - ▶ Some very fast **algorithms & modules** in place

```
// projection of direction onto normal vector of reference frame
double PC = pVector[4] * C[0] + pVector[5] * C[1] + pVector[6] * C[2];
double Bn = 1. / PC;

double Bx2 = -A[2] * pVector[29];
double Bx3 = A[1] * pVector[38] - A[2] * pVector[37];

double By2 = A[2] * pVector[28];
double By3 = A[2] * pVector[36] - A[0] * pVector[38];

double Bz2 = A[0] * pVector[29] - A[1] * pVector[28];
double Bz3 = A[0] * pVector[37] - A[1] * pVector[36];

double B2 = B[0] * Bx2 + B[1] * By2 + B[2] * Bz2;
double B3 = B[0] * Bx3 + B[1] * By3 + B[2] * Bz3;
```

Some of this great CPU performance still roots in the FORTRAN style writing.

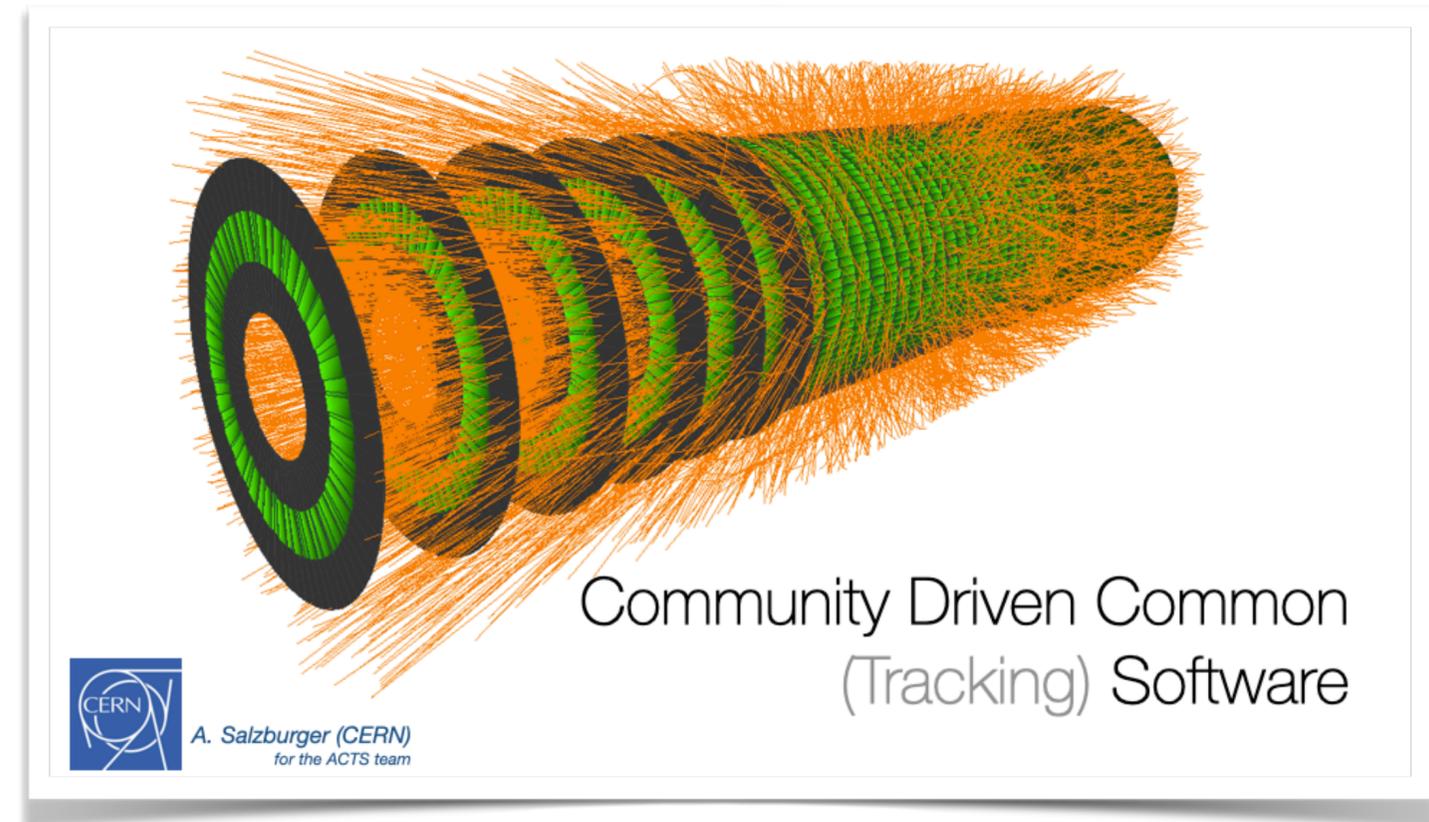
AtlasStepper, transcript in ACTS project

A great starting point

- ATLAS Track reconstruction SW
 - ▶ Designed/written with common (agnostic) top layer for **two** tracking devices
 - ▶ Some very fast **algorithms & modules** in place
 - ▶ Undergone (like other experiment SW) **successful cleanup campaigns**
 - all LHC experiments reduced CPU time significantly since LHC startup
- However, time starts to show its scars
 - ▶ Several **developers left** the project/field
 - ▶ Some **technology advancements** missed/only profited in a limited manner

acts project - **Mission statement**

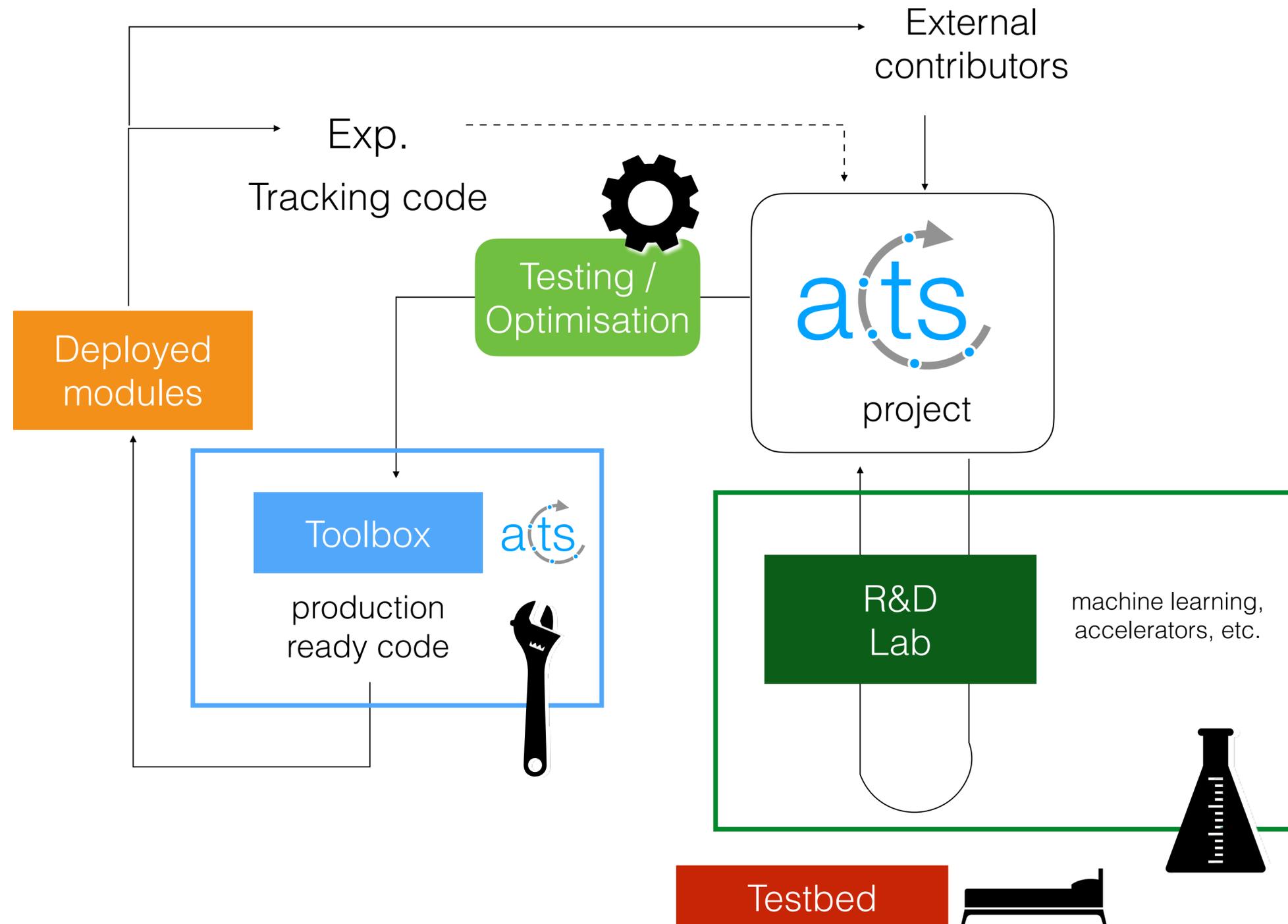
- **Preserve & advance** LHC state of the art track reconstruction software
- Develop & deploy **production ready software** for HL-LHC and beyond
- Establish **R&D testbed** for algorithms, technology advance (e.g. ML, GPU, detectors)



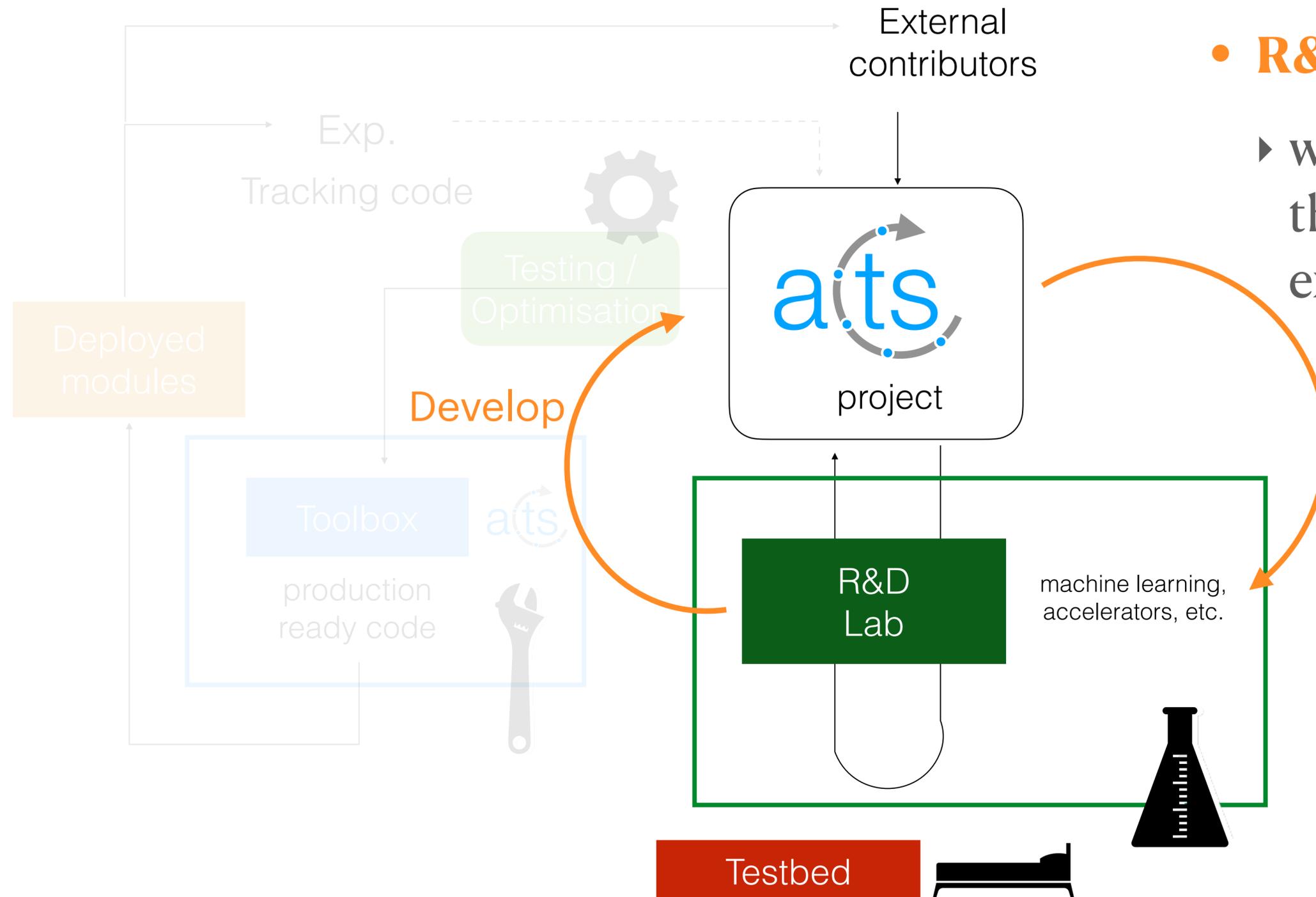
[AS, Community Driven Common Tracking Software, Plenary, CHEP2018](#)

- **Work & educate** in state of the art technology/workflows

ats project - Roadmap



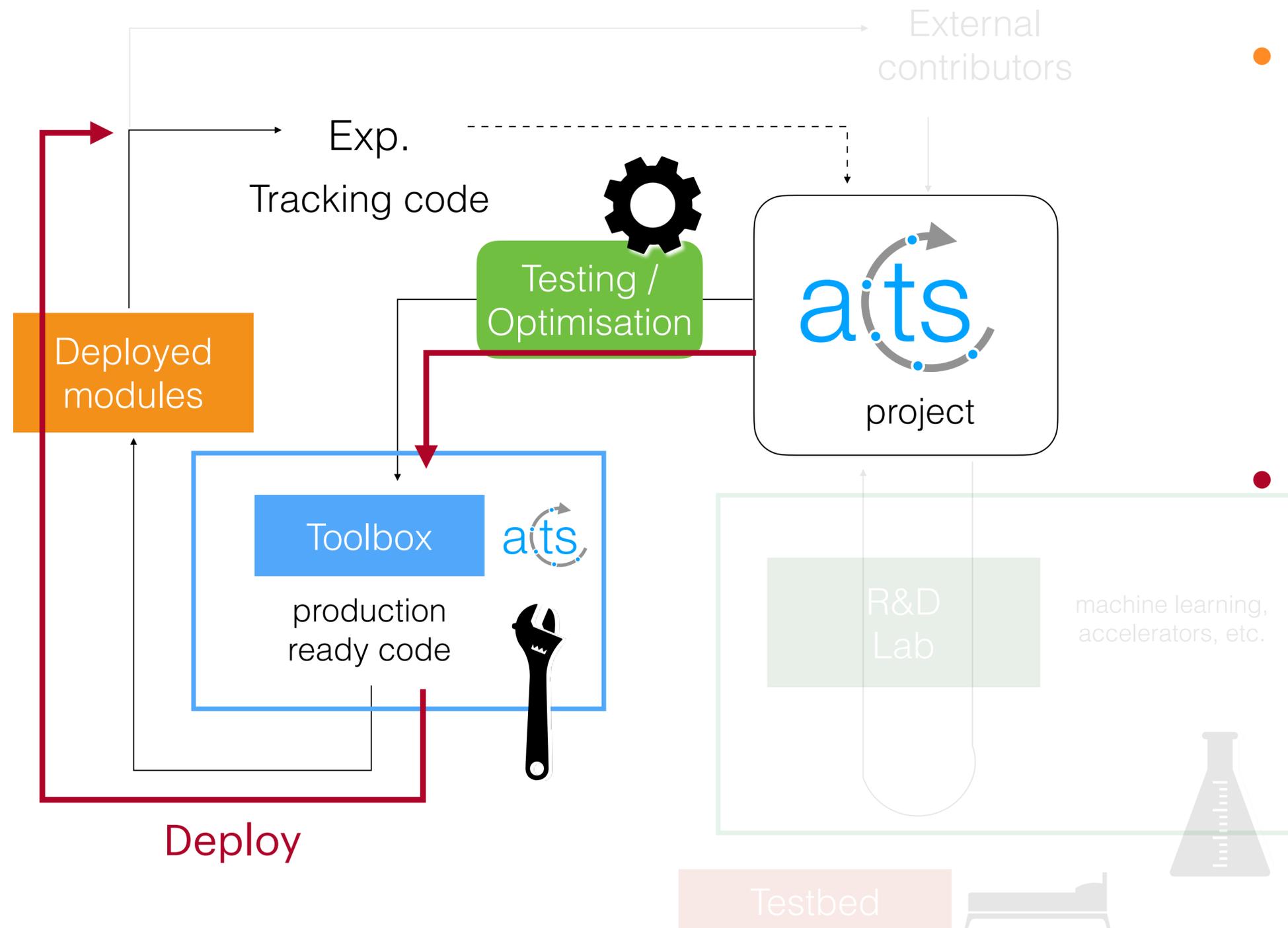
ats project - Roadmap



- **R&D loop**

- ▶ way faster/easier than in one experiment's context

ats project - Roadmap



- **R&D loop**

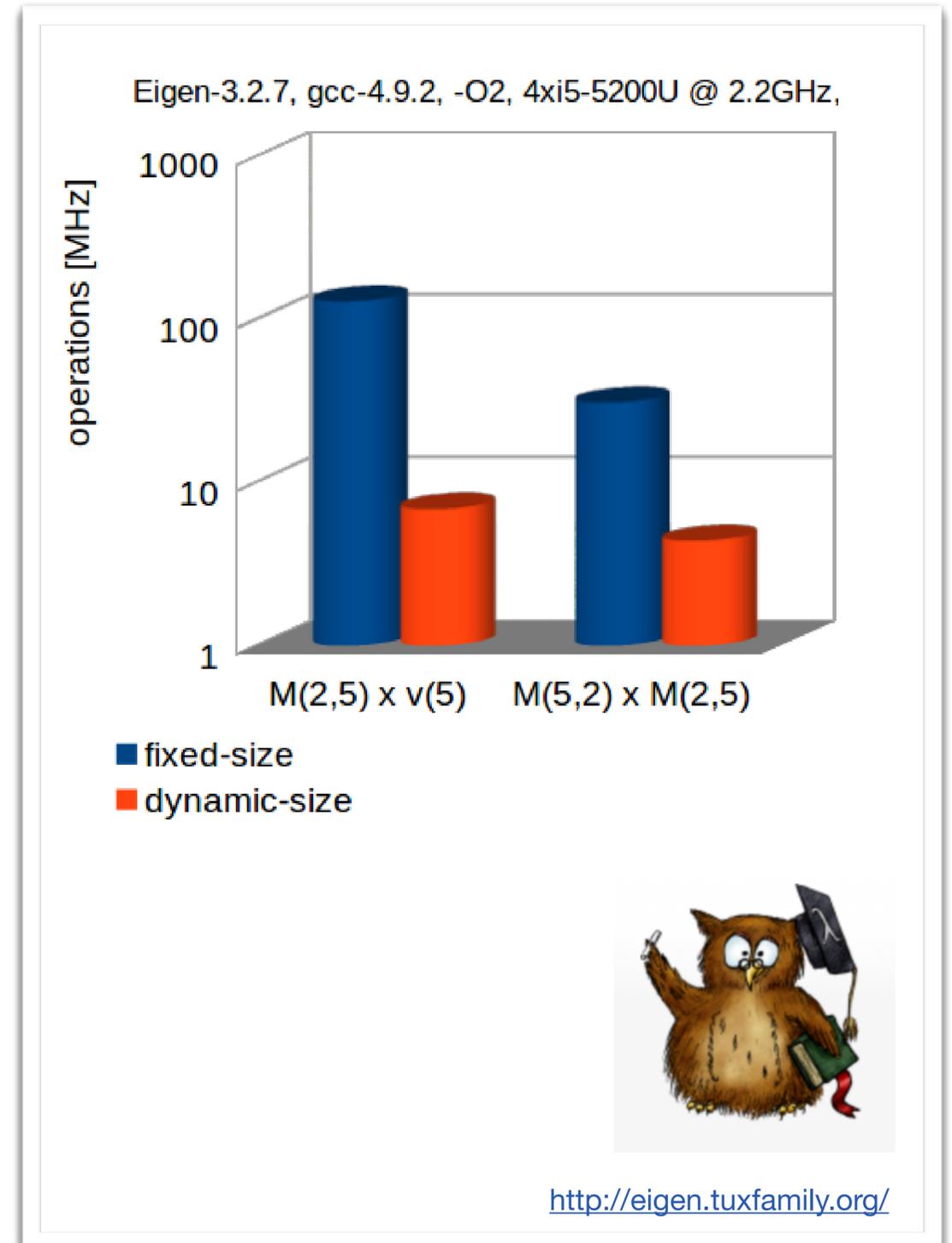
- ▶ way faster/easier than in one experiment's context

- **Deployment**

- ▶ Significantly slower/more involved than in one experiments' context

ats project - Preserve & advance

- **Extensive review of existing code**
 - ▶ Don't be shy, **take what's good!**
 - ▶ **Technical paradigm shift:**
 - Flatten data structures
 - Fixed size matrix multiplications
 - Lift runtime polymorphism to compile-time (C++ concepts replacing interfaces)
 - ▶ Modernize, simplify & streamline



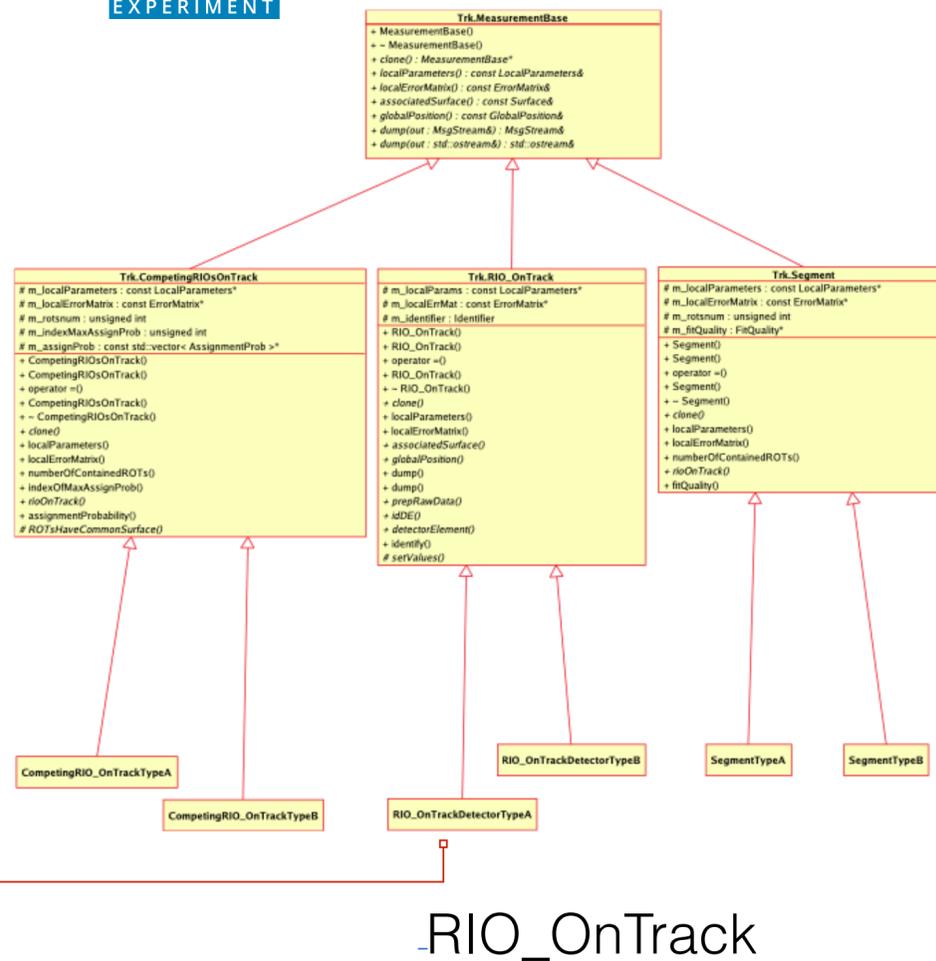
a:ts project - Simplify & Streamline



MeasurementBase



Measurement

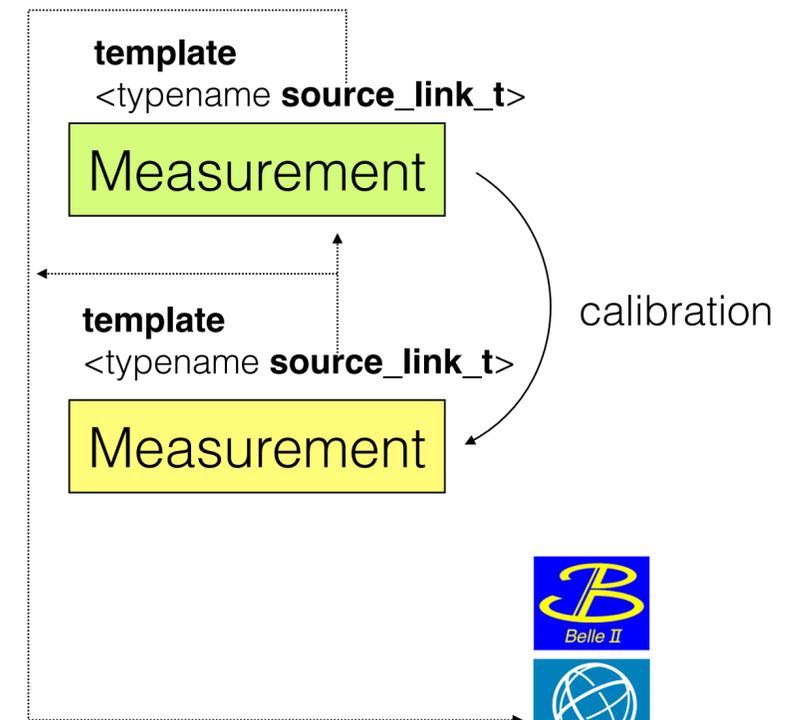


PrepRawData

PRD

RIO_OnTrack

calibration



acts project - Preserve & advance

- Example: modernize

```
// projection of direction onto normal vector of reference frame
double PC = pVector[4] * C[0] + pVector[5] * C[1] + pVector[6] * C[2];
double Bn = 1. / PC;

double Bx2 = -A[2] * pVector[29];
double Bx3 = A[1] * pVector[38] - A[2] * pVector[37];

double By2 = A[2] * pVector[28];
double By3 = A[2] * pVector[36] - A[0] * pVector[38];

double Bz2 = A[0] * pVector[29] - A[1] * pVector[28];
double Bz3 = A[0] * pVector[37] - A[1] * pVector[36];

double B2 = B[0] * Bx2 + B[1] * By2 + B[2] * Bz2;
double B3 = B[0] * Bx3 + B[1] * By3 + B[2] * Bz3;
```

AtlasStepper, transcript in ACTS project

```
void covarianceTransport(Covariance& covarianceMatrix, Jacobian& jacobian,
                       FreeMatrix& transportJacobian, FreeVector& derivatives,
                       BoundToFreeMatrix& jacobianLocalToGlobal,
                       const Vector3D& direction) {
    // Build the full jacobian
    jacobianLocalToGlobal = transportJacobian * jacobianLocalToGlobal;
    const FreeToBoundMatrix jacToLocal =
        surfaceDerivative(direction, jacobianLocalToGlobal, derivatives);
    const Jacobian jacFull = jacToLocal * jacobianLocalToGlobal;

    // Apply the actual covariance transport
    covarianceMatrix = jacFull * covarianceMatrix * jacFull.transpose();

    // Reinitialize jacobian components
    reinitializeJacobians(transportJacobian, derivatives, jacobianLocalToGlobal,
                          direction);

    // Store The global and bound jacobian (duplication for the moment)
    jacobian = jacFull;
}
```

EigenStepper, covariance transport

acts project - Production ready toolbox

Hosted on github.com



<https://github.com/acts-project/acts>

Synchronised gitlab read-only repository (former repo): <https://gitlab.cern.ch/acts>

Repository Structure:

- **Core:** Toolbox with production-ready code
- **Fatras:** Fast simulation extension
- **Tests:** Unit & Integration tests for
- **Plugins:** third party dependent plugins, e.g. Geant4, DD4Hep, ROOT, ...
- **Examples:** Test framework with examples (FCC-hh, ITk, sPhenix, ...)

Dependencies

- **Core:** Eigen, BOOST (for unit testing only)

Standards

- **C++17**

Licence



Mozilla Public Licence 2.0

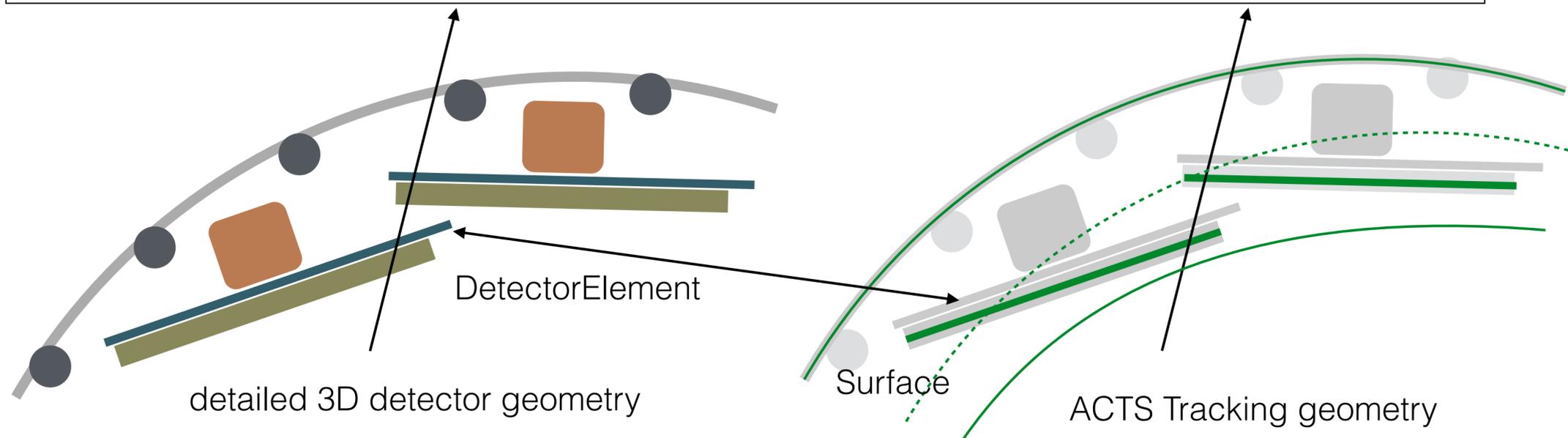
CERN officially recommends using an **outside provider** for development including non-CERN contributors ...

acts project - The toolbox

- Flexibility to interface with detector **SW & Plugin** mechanism

```
namespace Acts {  
  /// doxygen documentation  
  class DetectorElementBase {  
    /// the according represented surface  
    virtual const Surface& associatedSurface() const = 0;  
  };  
}
```

```
class MyDetectorElement {  
  /// @copydoc DetectorElementBase::associatedSurface  
  const PlaneSurface& associatedSurface() const;  
};
```

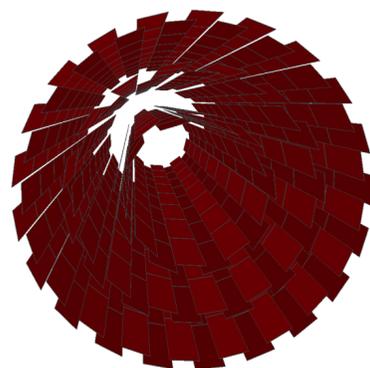


aits project - The toolbox

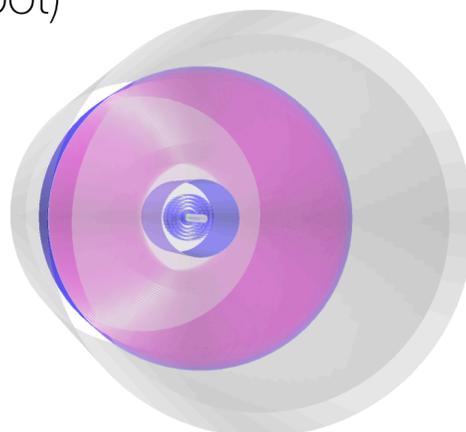
- Flexibility to interface with detector **SW & Plugin** mechanism

C++

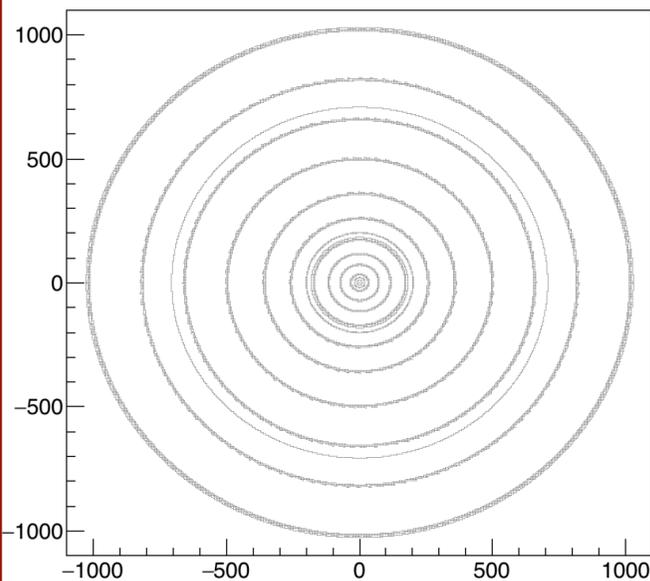
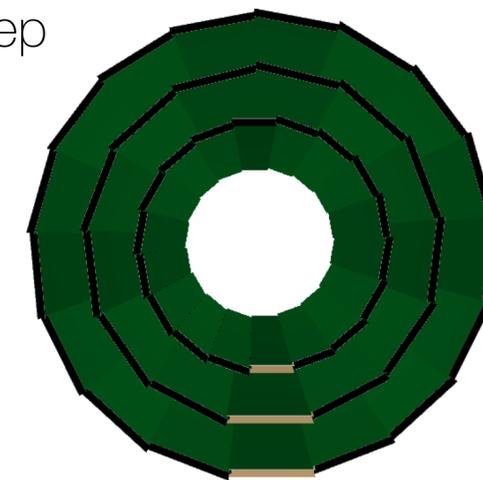
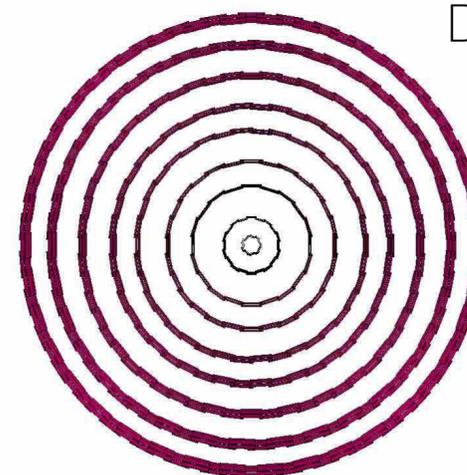
```
// BARREL :
// 4 pixel layers
// configure the central barrel
plbConfig.centralLayerBinMultipliers = {1, 1};
plbConfig.centralLayerRadii = {32., 72., 116., 172.};
plbConfig.centralLayerEnvelopes
= {pcEnvelope, pcEnvelope, pcEnvelope, pcEnvelope};
// material concentration always outside the modules
plbConfig.centralLayerMaterialConcentration = {1, 1, 1, 1};
plbConfig.centralLayerMaterialProperties
= {pcmbProperties, pcmbProperties, pcmbProperties, pcmbProperties};
plbConfig.centralModuleBinningSchema = {{16, 14}, {32, 14}, {52, 14}, {78, 14}};
plbConfig.centralModuleTiltPhi = {0.14, 0.14, 0.14, 0.14};
plbConfig.centralModuleHalfX = {0.4, 0.4, 0.4, 0.4};
plbConfig.centralModuleHalfY = {36., 36., 36., 36.};
plbConfig.centralModuleThickness = {0.15, 0.15, 0.15, 0.15};
plbConfig.centralModuleMaterial
= {pcMaterial, pcMaterial, pcMaterial, pcMaterial};
// pitch definitions
plbConfig.centralModuleReadoutBinsX = {336, 336, 336, 336};
plbConfig.centralModuleReadoutBinsY = {1280, 1280, 1280, 1280};
plbConfig.centralModuleReadoutSide = {-1, -1, -1, -1};
plbConfig.centralModuleLorentzAngle = {0.12, 0.12, 0.12, 0.12};
// no frontside/backside
plbConfig.centralModuleFrontsideStereo = {};
plbConfig.centralModuleBacksideStereo = {};
plbConfig.centralModuleBacksideGap = {};
```



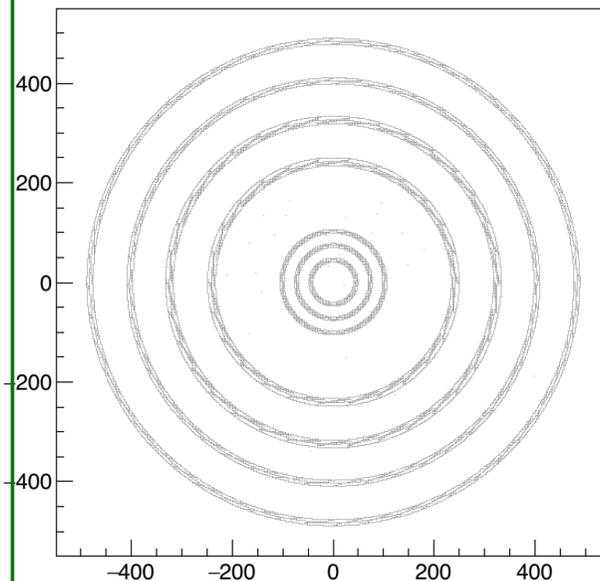
TGeo (Root)



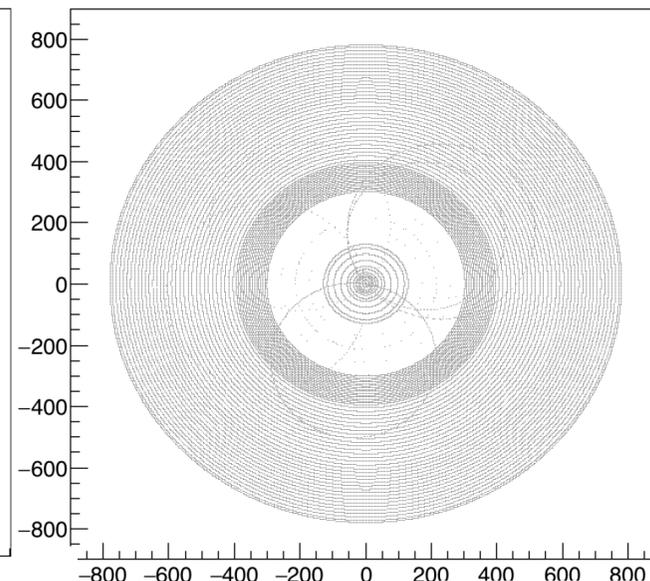
DD4Hep



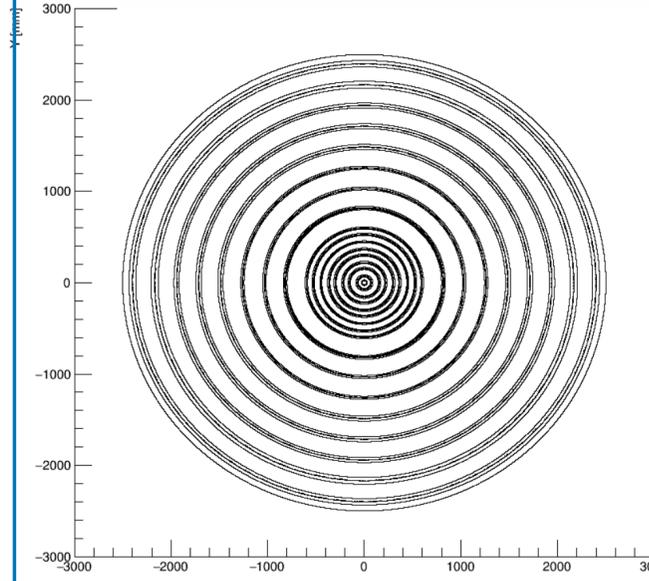
TrackML Detector



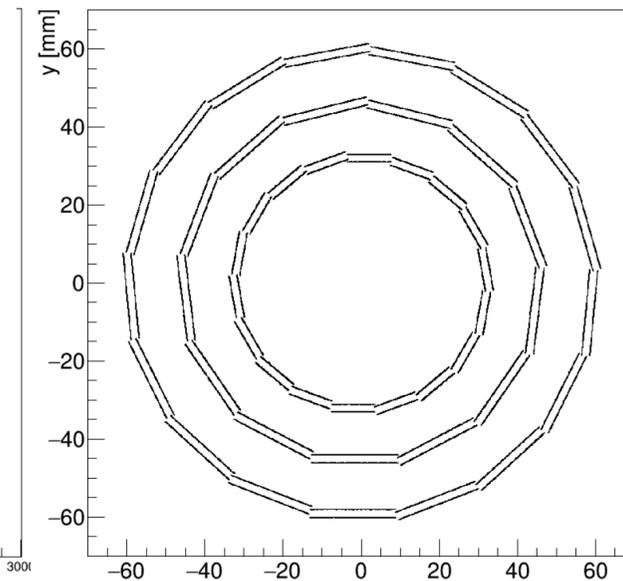
CMS Pixel & SVX



sPhenix



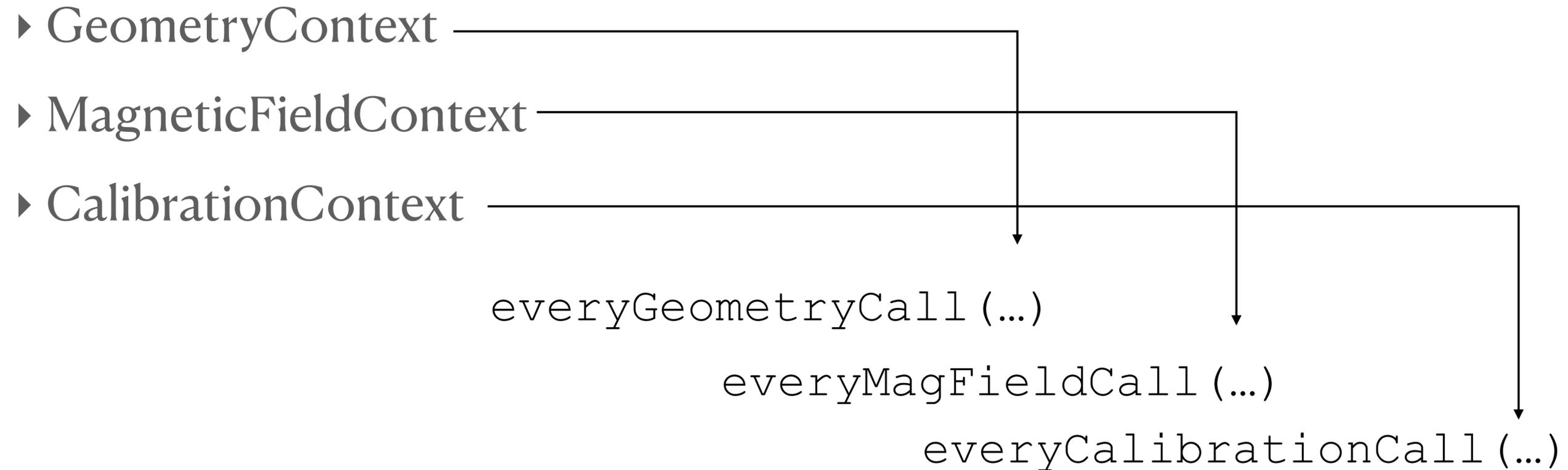
FCC-hh



CLIC Vertex

aits project - The toolbox

- **Contextual data** processing with **parallel** code execution:



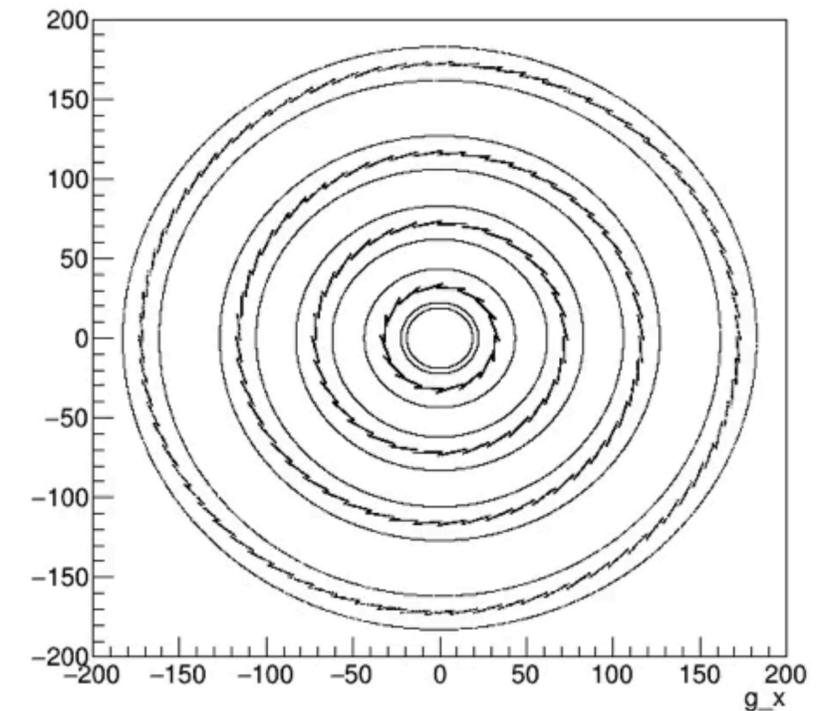
Quasi-free condition/contextual data handling.

aits project - The toolbox

- **Contextual data** processing with **parallel** code execution:
 - ▶ GeometryContext
 - ▶ MagneticFieldContext
 - ▶ CalibrationContext

GeometryContext In Action

Different alignment in every event

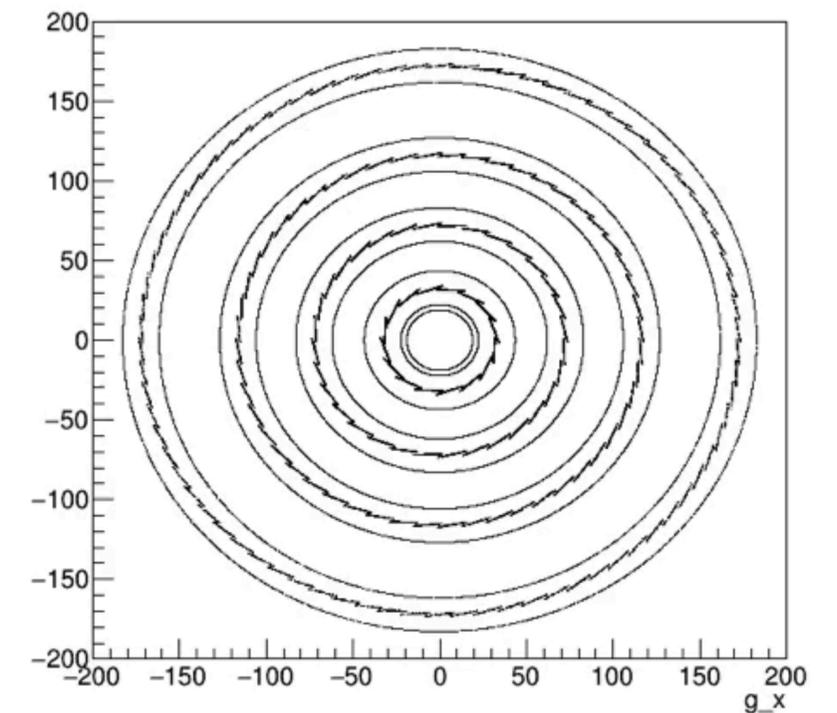


aits project - The toolbox

- **Contextual data** processing with **parallel** code execution:
 - ▶ GeometryContext
 - ▶ MagneticFieldContext
 - ▶ CalibrationContext

GeometryContext In Action

Different alignment in every event



acts project - The toolbox & integration

- **Plumbing & where it usually gets dirty:** integration into experiment code base
 - ▶ **Keep it simple**, e.g. configuration done with nested Config structs

```
namespace Acts {  
  /// doxygen documentation  
  class WorkHorse {  
    /// @struct Config for To  
    struct Config {  
      float coatColor; ///  
      float maxPath;    ///  
    };  
  };  
}
```

- ▶ Possible binding to **Gaudi(Hive):**

```
/// feed from Framework into ACTS configuration  
declareProperty("CoatColor", m_cfg.coatColor);  
declareProperty("MaxPath", m_cfg.maxPath);
```

acts project - The toolbox & integration

- **Plumbing & where it usually gets dirty:** integration into experiment code base
 - ▶ Any external code **needs to tie in** with experiment code base
 - may sound trivial, but e.g. log messaging needs to work

```
///  
/// This macro allows to use a locally defined logging object with the ACTS_*  
/// logging macros. The envisaged usage is the following:  
///
```

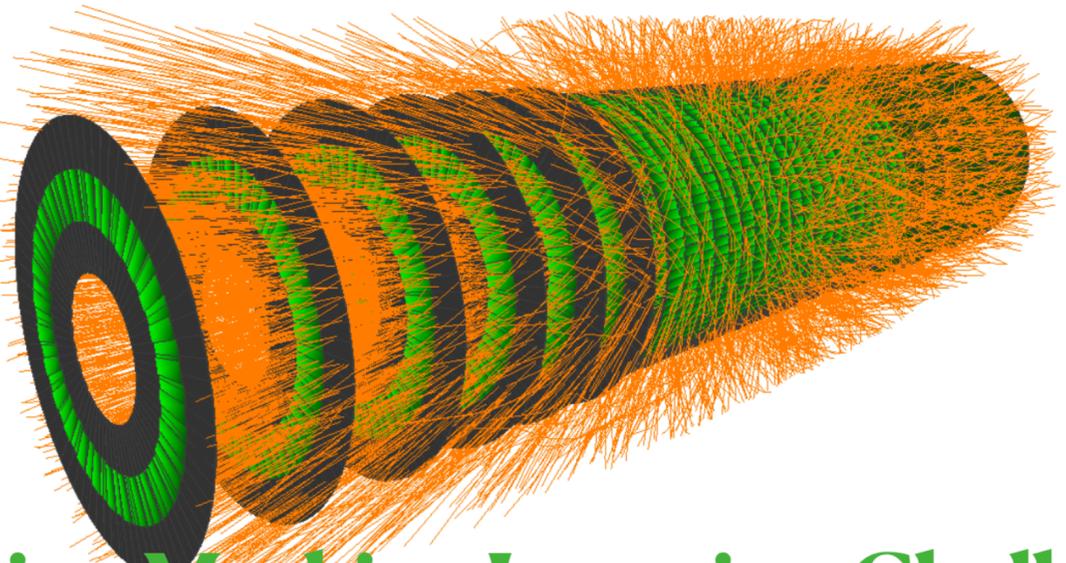
- **General guideline** (“fire brigade design”)
 - ▶ You **call us**, we never call you.
 - ▶ If you need something from your house, we **will bring** it to you.

project - R&D for algorithm research

- *Classical* algorithms are **not frozen**, there's room for algorithm R&D
- ACTS aims to provide a testbed for this research:
 - ▶ Testbed detectors available for testing
 - ▶ Examples (being worked on currently):
 - ray tracing inspired navigation through detector geometry
 - generalised track linearisation in vertex fitting
 - extension of KF into a global measurement formulation

aits project - R&D for ML

- Fast simulation extension was used for **Tracking Machine Challenge**
 - ▶ Very successful **kaggle** & **codalab** challenges
 - ▶ Many interesting ideas & solutions
 - ▶ Established a reference data set & format
 - still in use for much of current Tracking R&D



Tracking Machine Learning Challenge

A summary



@SaltyBurger



A. Salzburger (CERN) for the

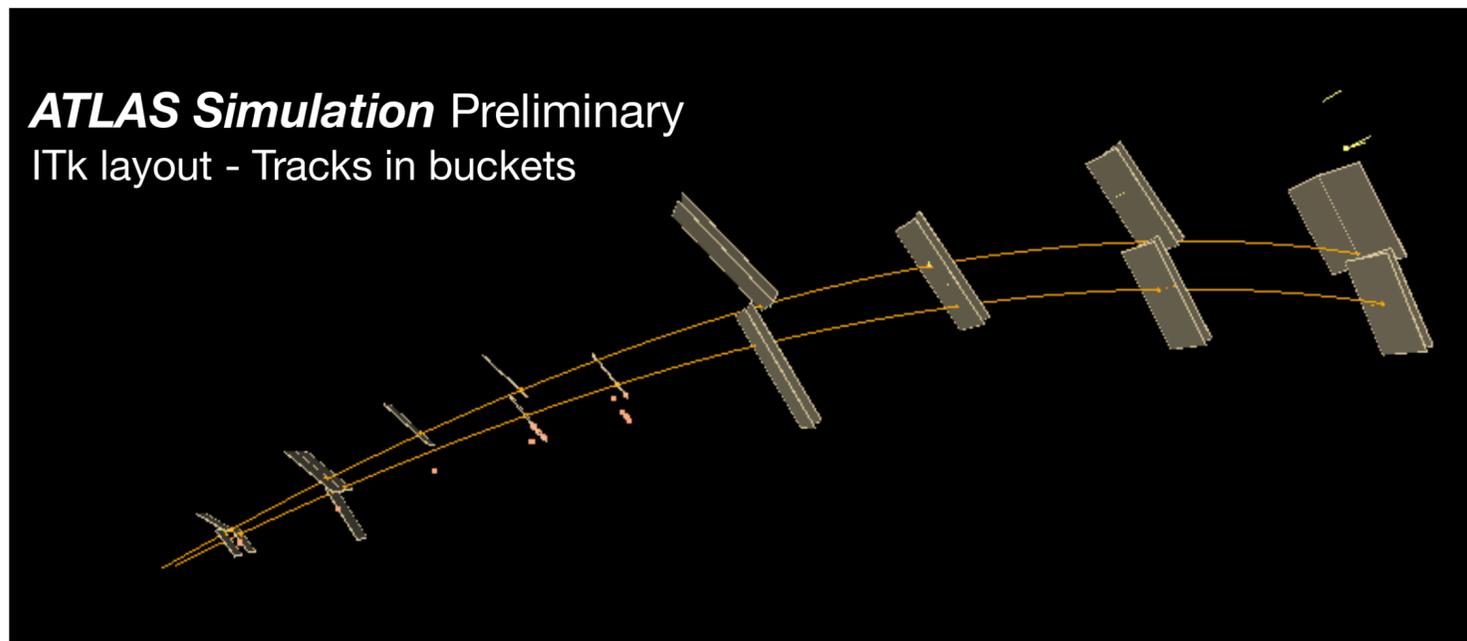


organisers

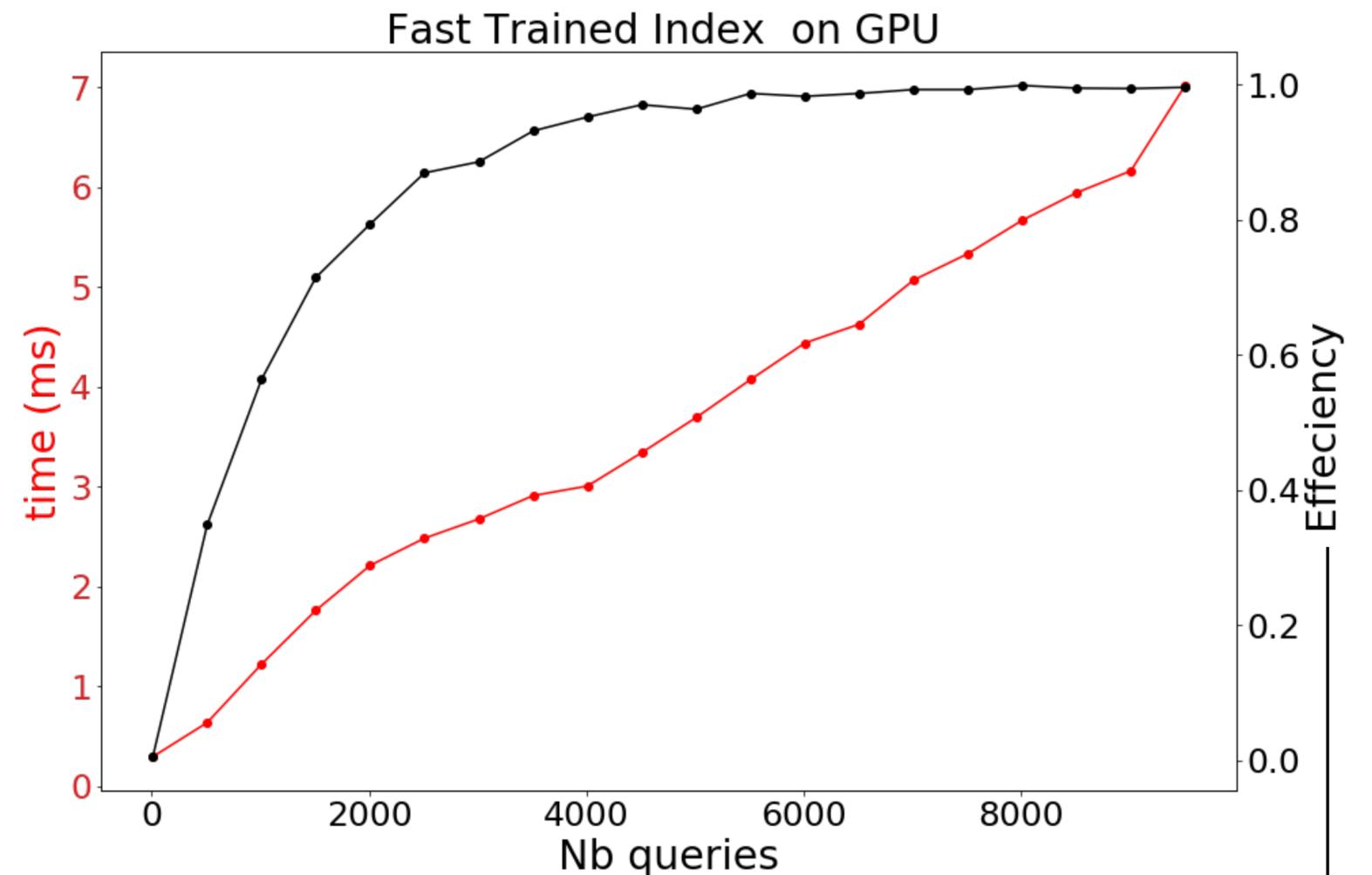
[AS, Conclusion of the TrackML Challenge, ICHEP2020](#)

aits project - R&D for ML

- Standalone examples allow for novel (ML) algorithm R&D
 - ▶ **Example:** track seeding using approximate nearest neighbourhood



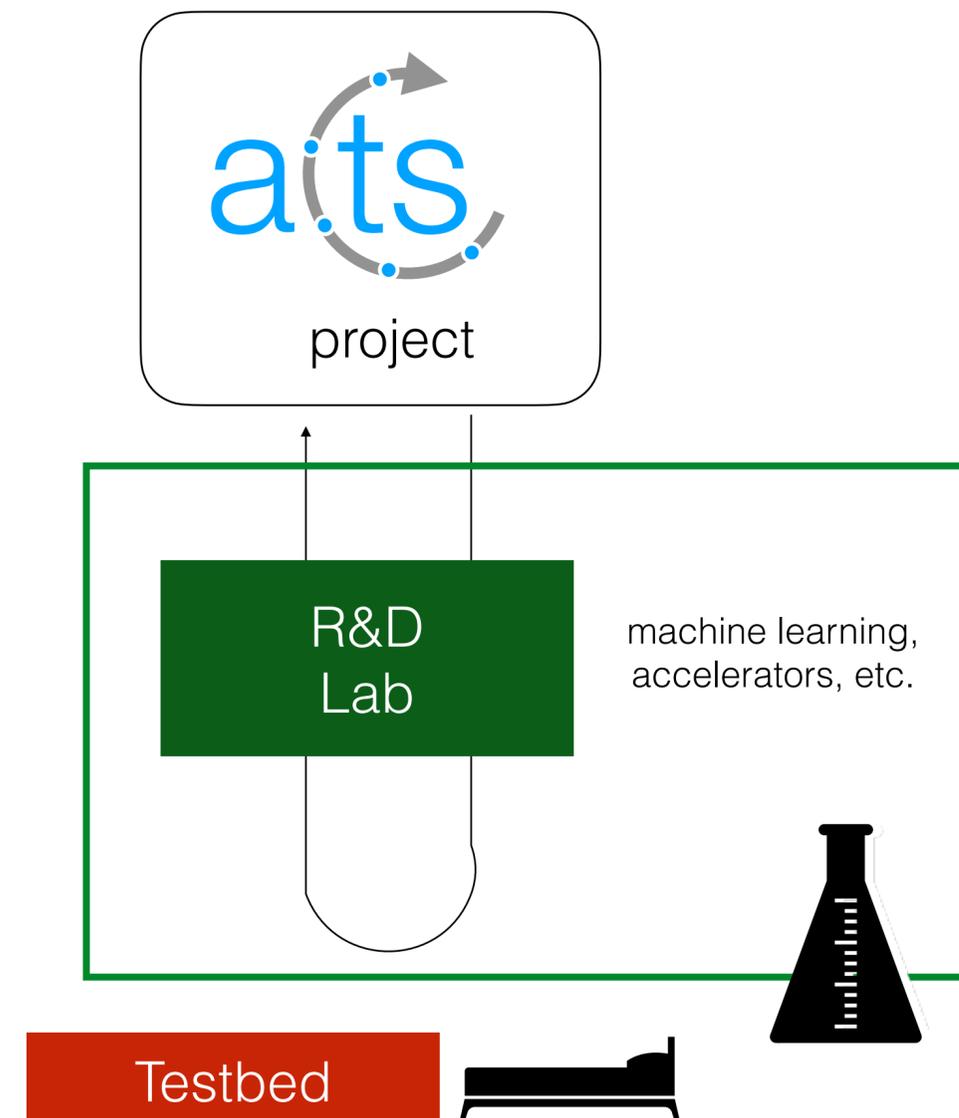
These tracks are brought to you by 



To find a bucket with at least 4/hits of the track contained (good enough for track seeding) ←

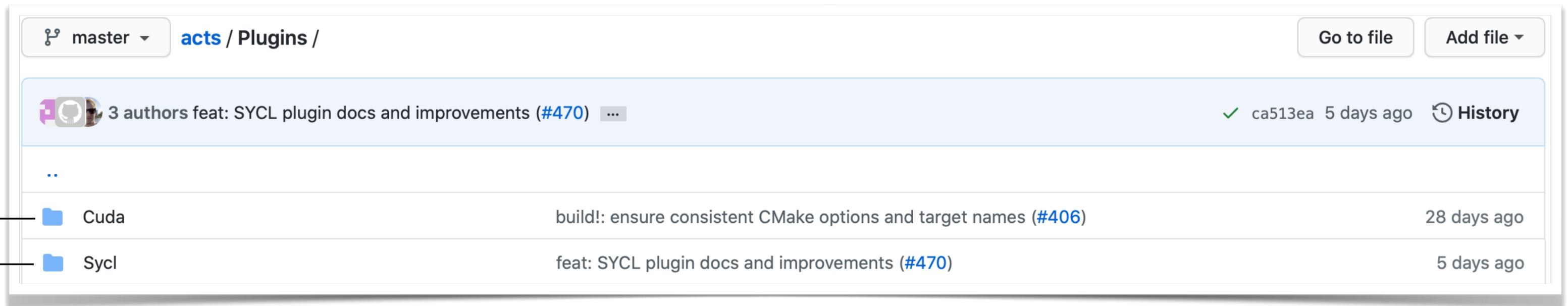
acts project - R&D for ML

- Standalone examples allow for novel (ML) algorithm R&D
 - ▶ **Example:** track seeding using approximate nearest neighbourhood
 - ▶ Aim to have a **path** way of these R&D projects **back** into the ACTS code base - **eventually** even production code
- **Learn** how to integrate external non-HEP software into HEP reconstruction code



acts project - R&D for accelerators

- Dedicated sub-group for parallelisation (acts-parallelization@cern.ch)
 - ▶ Evolving code to **run on different hardware**



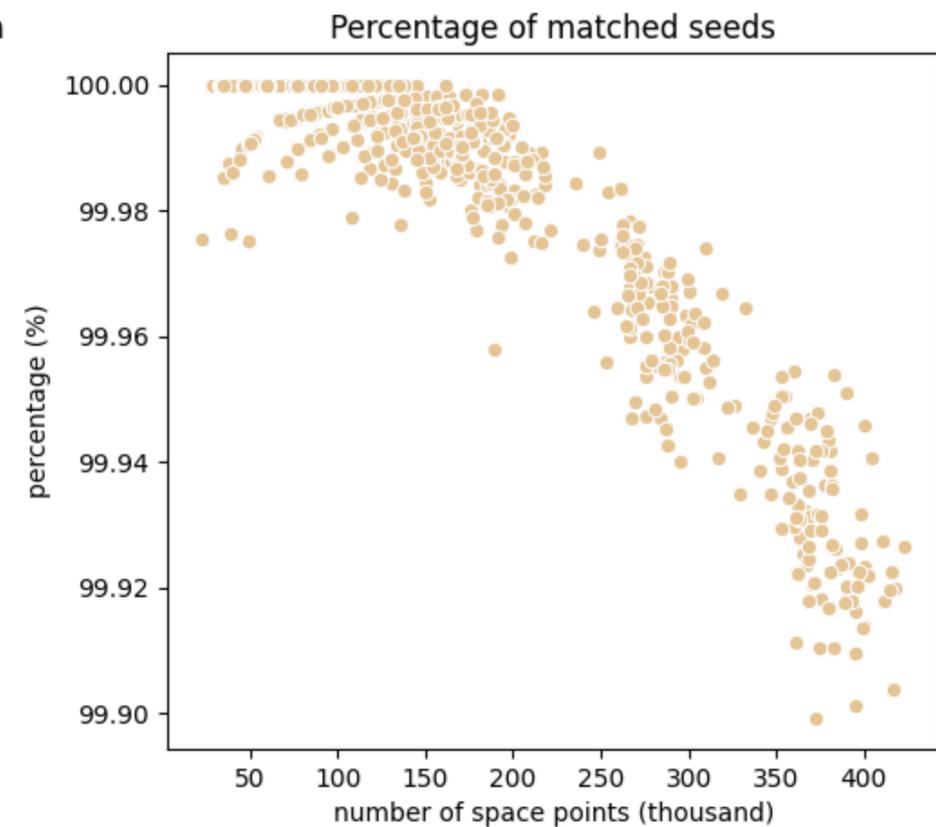
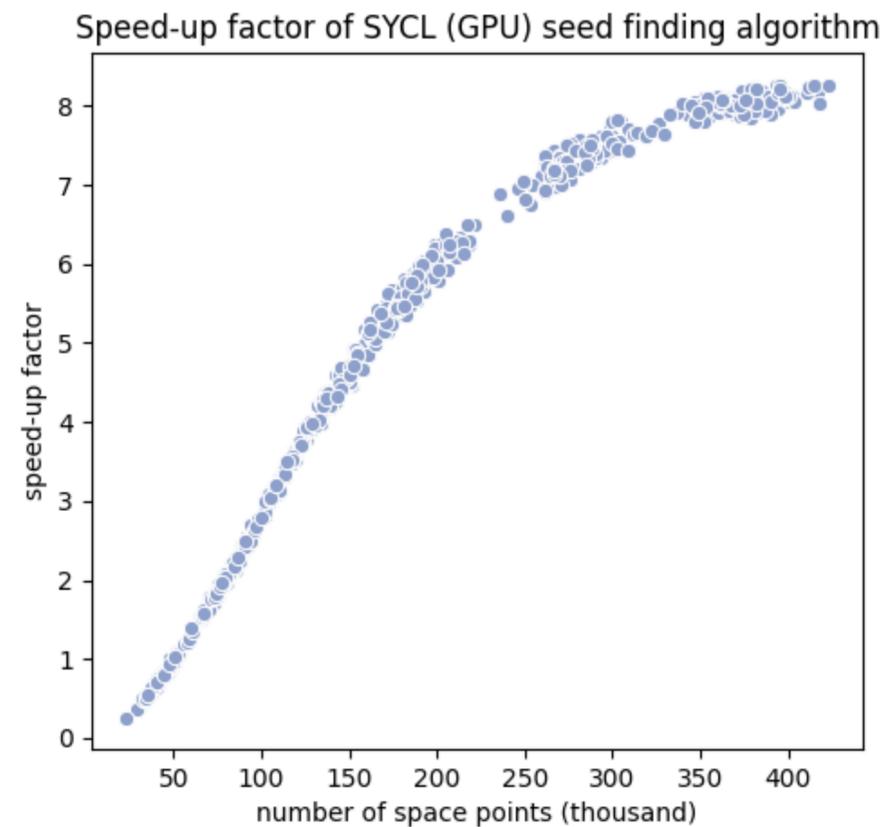
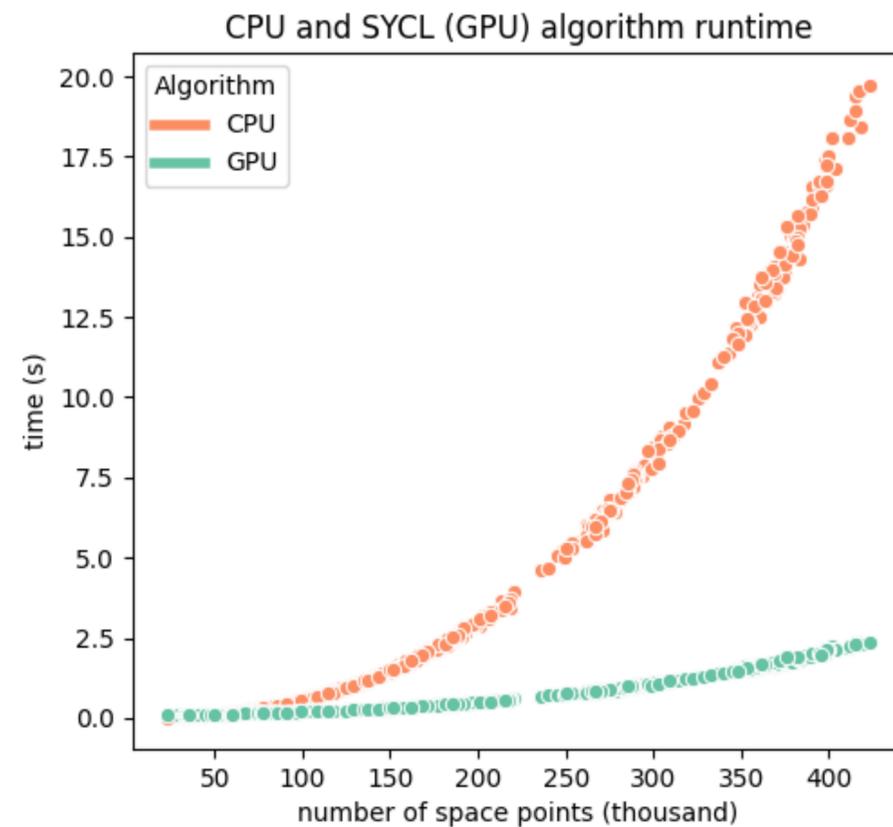
▶ Native CUDA modules (Seeding, Propagation, Fitting)

▶ SYCL/onAPI module (Seeding)



acts project - R&D for accelerators

- **Example:** SYCL/oneAPI implementation of track seeding algorithm
 - ▶ Reference to CPU implementation (ATLAS implementation)



- ▶ Valuable lessons/feedback to ACTS core project wrt EDM/algorithm design

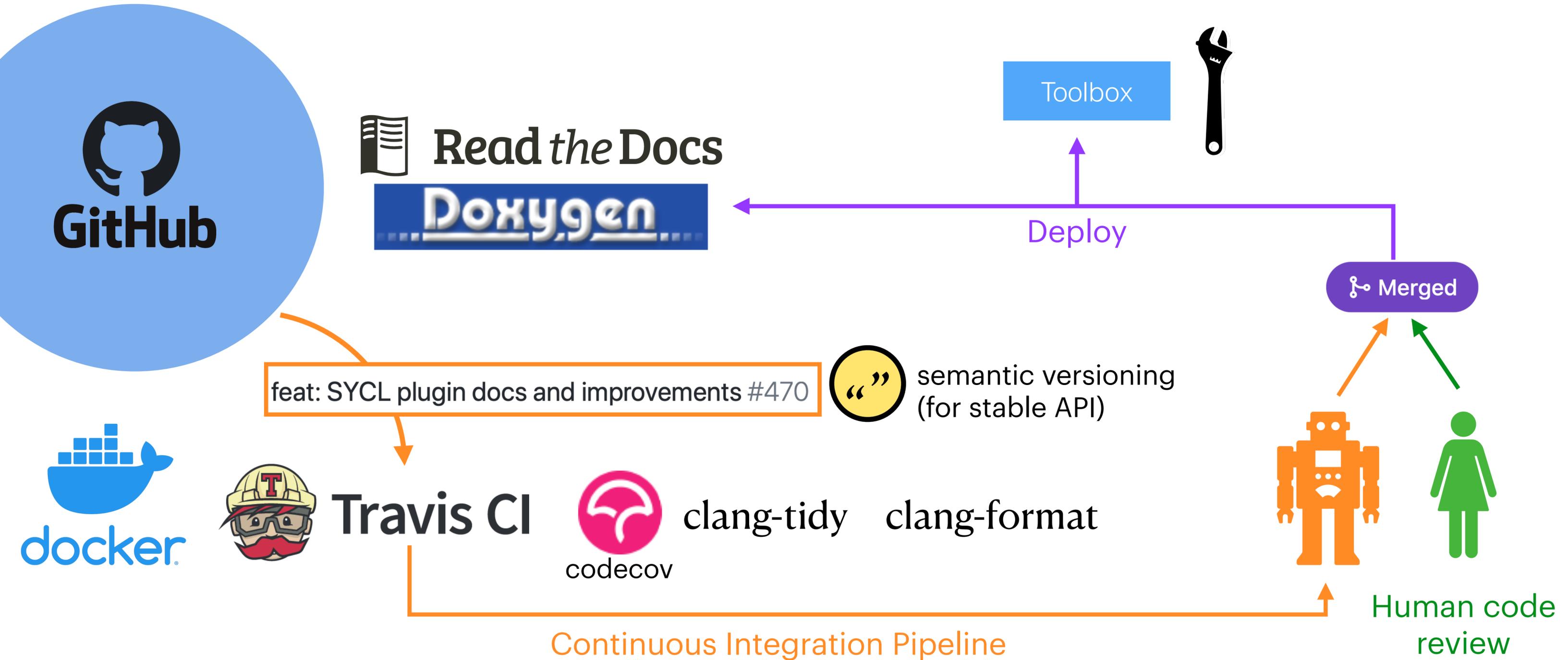
project - R&D for new SW technologies

- Stand-alone character of the repository eases R&D
 - ▶ Even larger scale inclusions, tests, expansions are simplified
 - ▶ Testbeds allow for rapid feedback on development
- Example: ACTS **auto-diff** within propagation
 - ▶ **auto-diff** is a rather recent development coming from the ML sector
 - relies on the fact that machine instructions are per definition differentiable
 - ▶ Test implementation in track propagation for derivative/jacobian calculation done

project - infrastructure & workflows

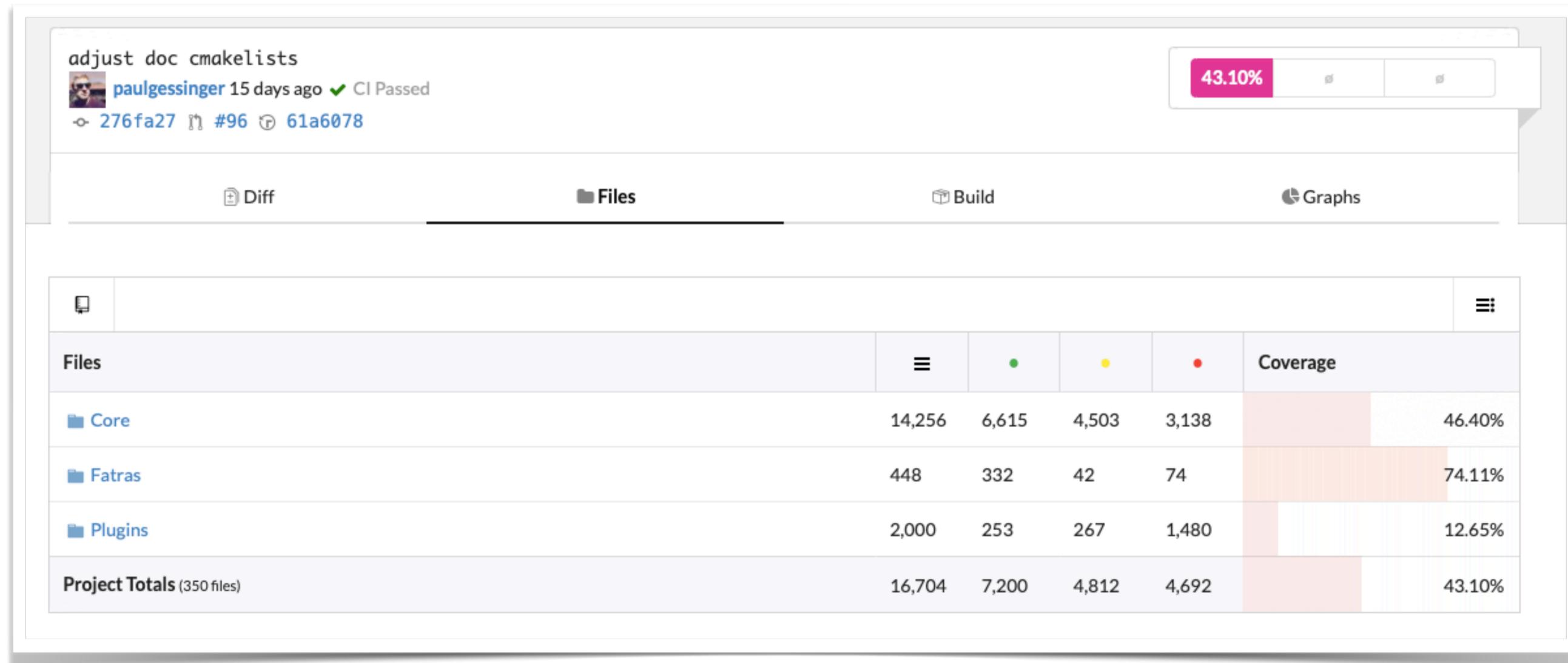
- HEP SW has/had sometimes a bad reputation in being “aged”
 - ▶ Same for workflows & technology
 - ▶ Some of it is rooted in the long lifetime & necessity of coherent data processing
- We need to stay attractive to the best technical students
 - ▶ Deploy **modern development workflows**
 - ▶ Be **open to move with technology**, new standards
 - ▶ A path to **publish the work** → **OpenDataDetector** project (backup)

aits project - workflow ecosystem



ats project - testing, code review & CI

- High level of **unit testing & CI** (coverage steadily increasing)

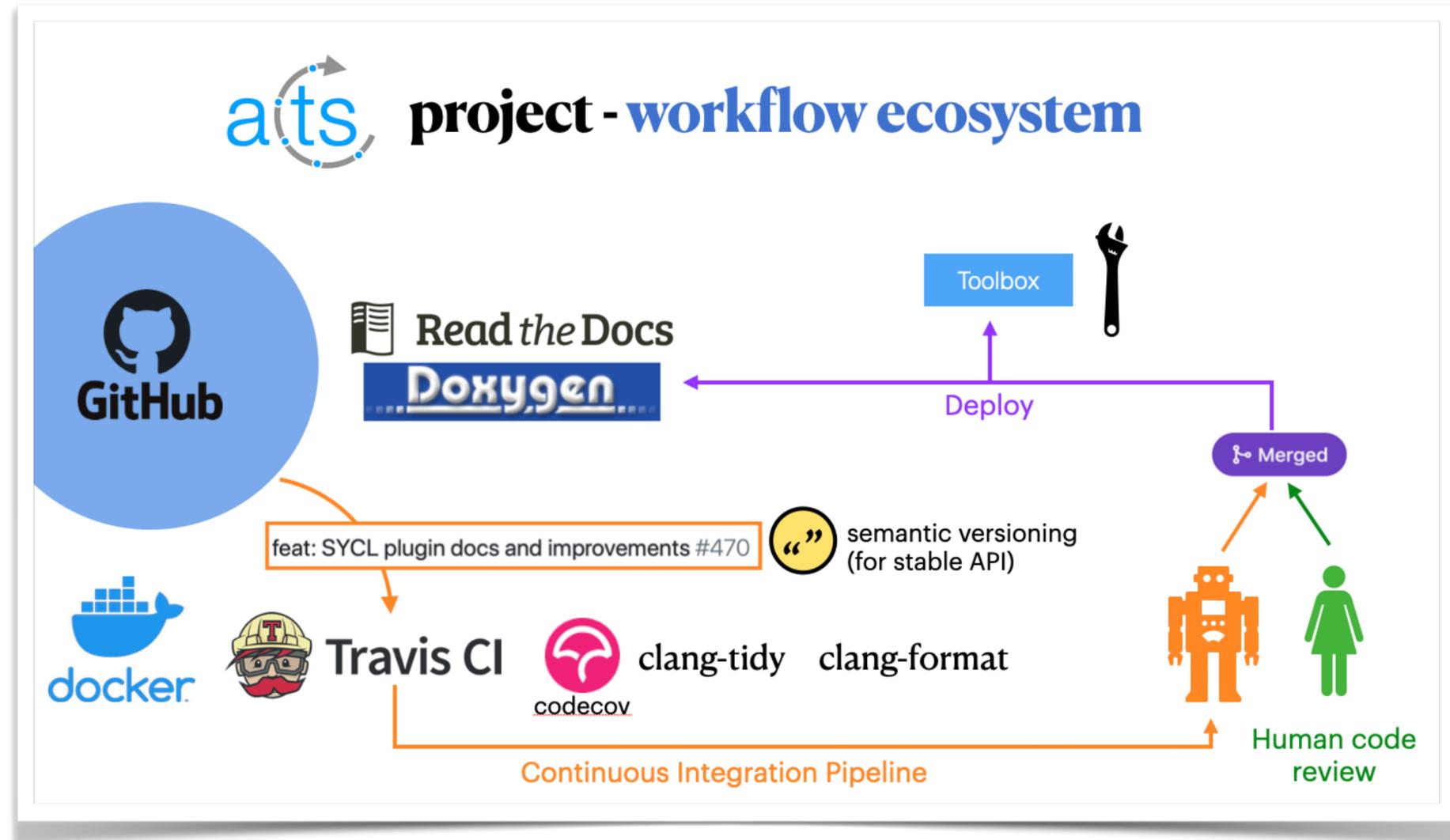


project - testing, code review & CI

- Directly benefits to involved experiments
 - ▶ **Highly tested code** deployed
 - ▶ **Experienced developers** that eventually co-authored the deployed code
- **ATLAS** example:
 - ▶ Migration to git/CI setup **pioneered** by ACTS developers
 - ▶ Vertexing speed-up by 40% being re-deployed to AthenaMT
 - ▶ ATLAS phase-2 ITK reconstruction planned with ACTS implementation

aits project - experienced workforce

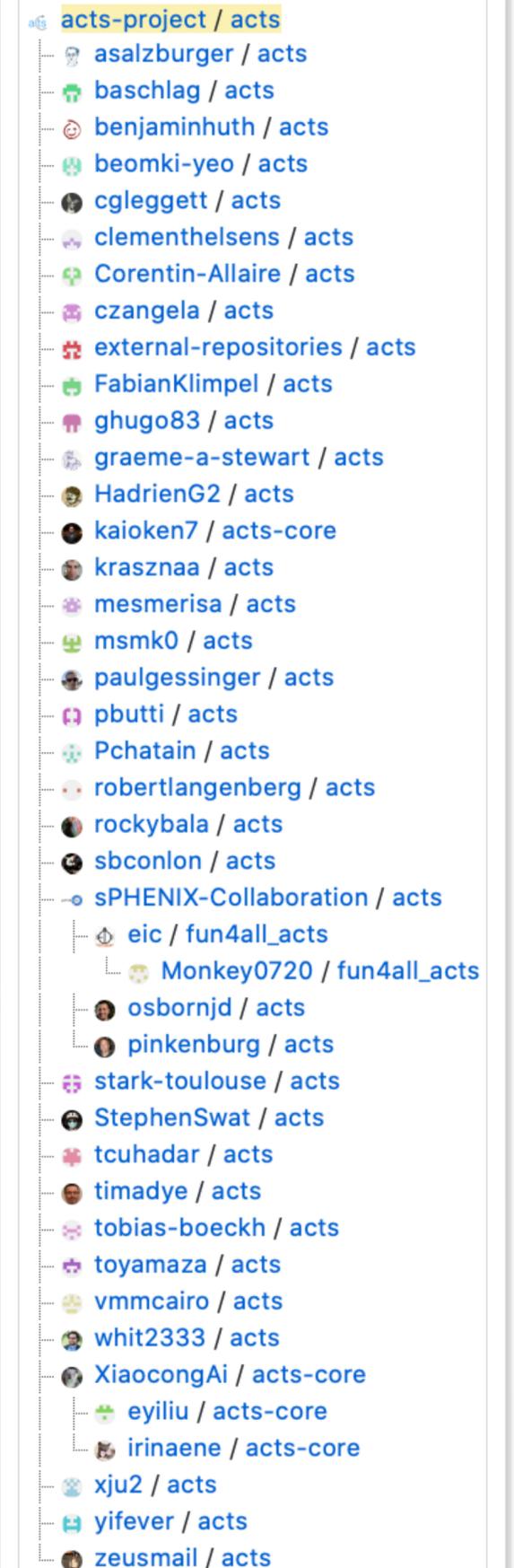
- Modern programming, language, conceptual skills & experience in industry standard workflows
 - ▶ Facilitates **change between experiments**
 - ▶ Facilitates careers outside the field
 - **open source access** make contribution **traceable**



That is an impressive skillset even/particularly outside HEP.

acts project - foster collaboration

- Currently **42** forks of the acts-project
 - ▶ Individuals (in HEP, students) or experiment forks (e.g. sPhenix)
- **10-15 active developers** on Core project
 - ▶ Still quite ATLAS heavy, though increasing contribution from **sPHENIX, Belle-2** or **non-associated individuals**

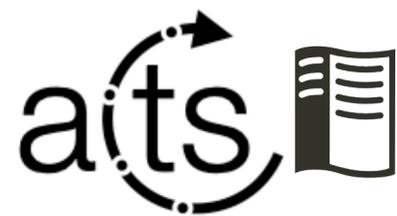


project - community, friend & family

- ACTS as one project in a new ecosystem of community driven SW
 - ▶ We **walk** and **we learn** together
 - interaction between SW projects **under the umbrella of HSF** is key
 - Example: ACTS **report** on Eigen compilation restrictions/issues in HSF WG#2
similar issues seen by other Eigen clients (CMS), follow up by HSF
 - ▶ We should **play** together
 - Encouragement to put modules together, build systems
 - Example: can ACTS run within on top of PataTrack, within ALLEN, etc...
 - ▶ Finally, we should **work** together

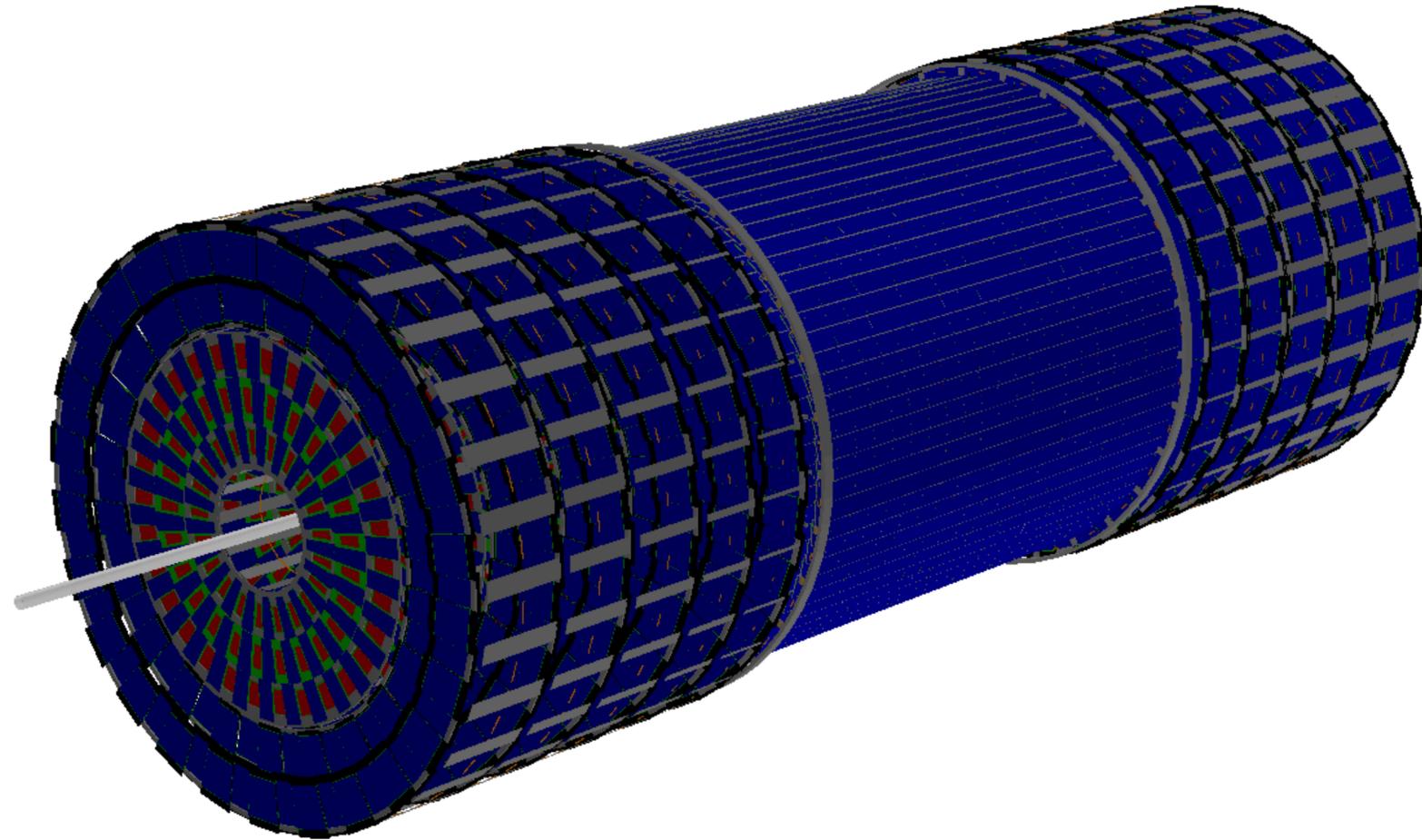
Summary

- **ACTS project** quite matured in the last 4 years
 - version v01.00.00 released in Sep 2020 with particle frozen API
- Encapsulation from experiment SW stack comes with a price tag
 - however, it did what we strived for: **enabled R&D in many areas**
 - starting to pay off (e.g. vertex reconstruction in ATLAS)



Read *the* Docs

- ✉ acts-developers@cern.ch
- ✉ acts-users@cern.ch
- ✉ acts-parallelization@cern.ch



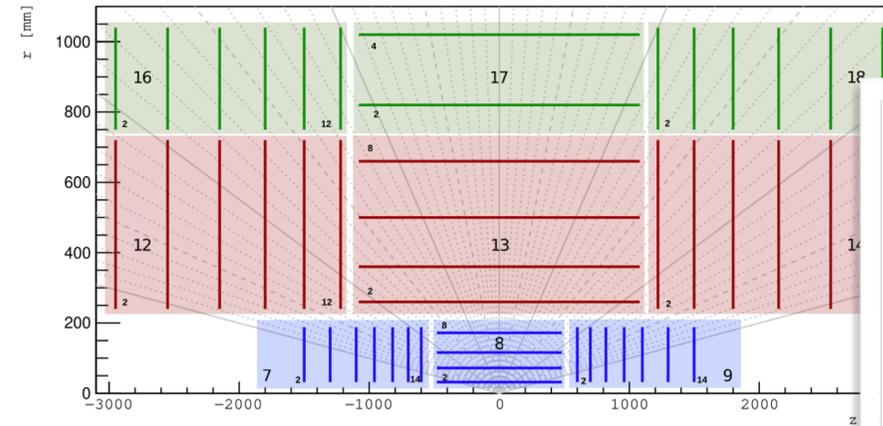
Open Data Detector - Status & Release



A. Salzburger (CERN)

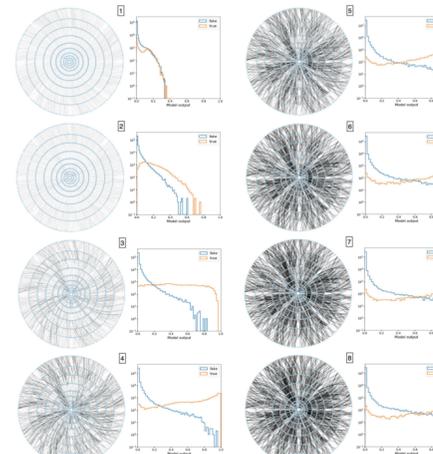
The dataset for current algorithm R&D

A virtual detector



20

Information Flow

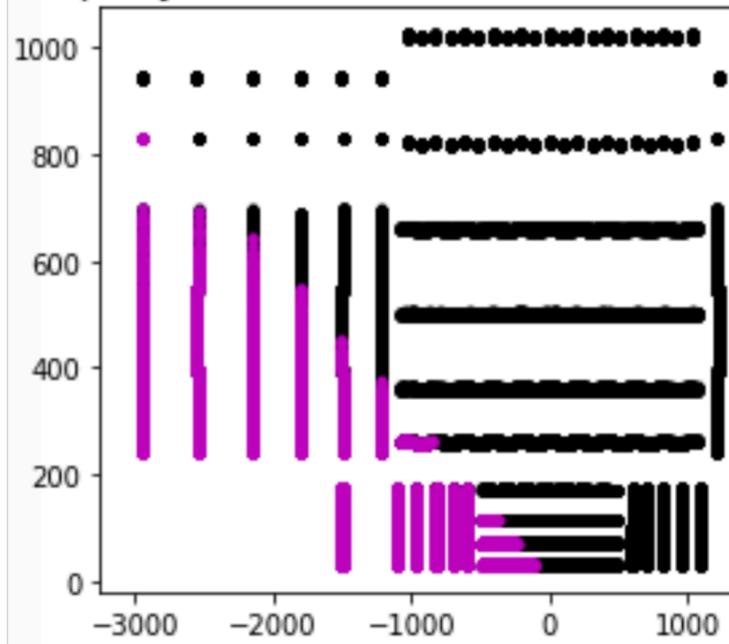


- Checking edge score after each step of graph network.
- Effective output of the model is in step 8.
- Full track hit assignment learned in last stages of the model.
- Tracklets learned in intermediate stages.

Graph Networks - Learning to Discover: Advanced Pattern Recognition, IPa Workshop 2019 | M.Kunze

24

query restricted to volume : 7, avg quality: 6.18



Results

dataset size: ~20%
1,637 particles, 11,030 hits

plotting error: too many
doublets 392529

392,529 doublets
 $p=0.26\%$, $r=99.15\%$

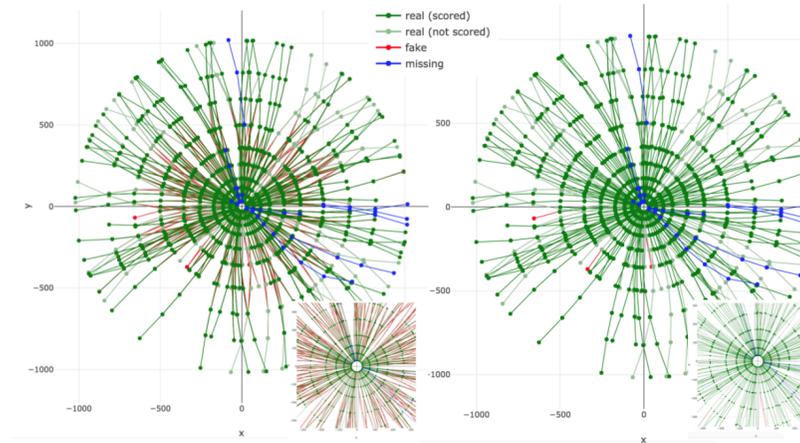
57.3s
build
QUBO

2,546 doublets
(2,964 triplets)
QUBO size: 14,345

17.1s
sample
QUBO
running on CPU

1,512 doublets
 $p=99.13\%$, $r=97.06\%$

trackml score **97.55%**



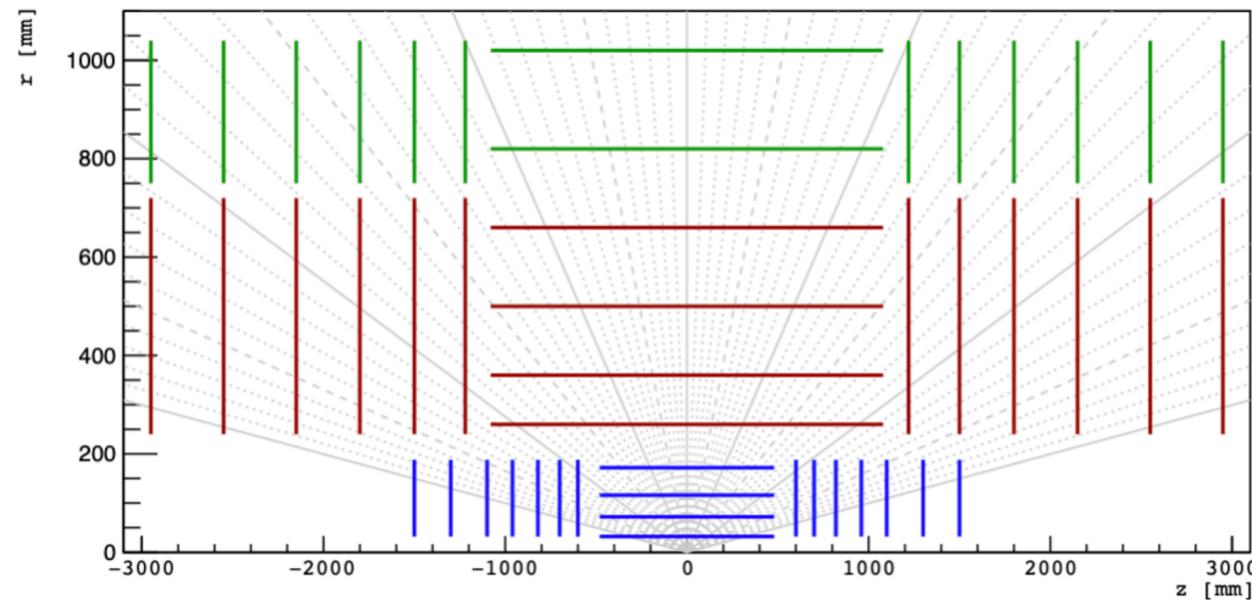
27

CHEP2018 - announcement

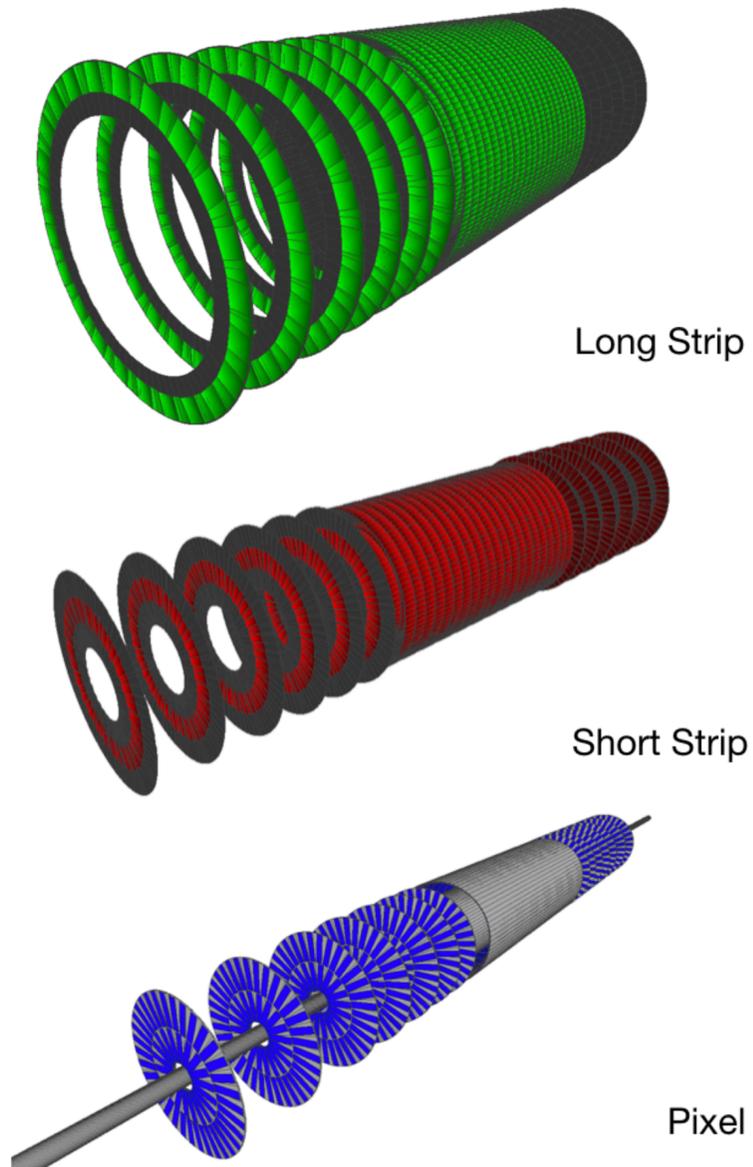
Reference detector & dataset

Common dataset for development within the community

- detector used for ~~TrackML~~
- dataset produced with ACTS fast simulation
- **proposal:** iron out the few features we discovered & dataset (LHC/HL-LHC), publish on [opendata](#) CERN



A. Salzburger - CHEP2018



39

Resynchronising master, add open data detector

!164 · opened 2 weeks ago by Andreas Salzburger

Improvement

New Feature

MERGED



1 of 2



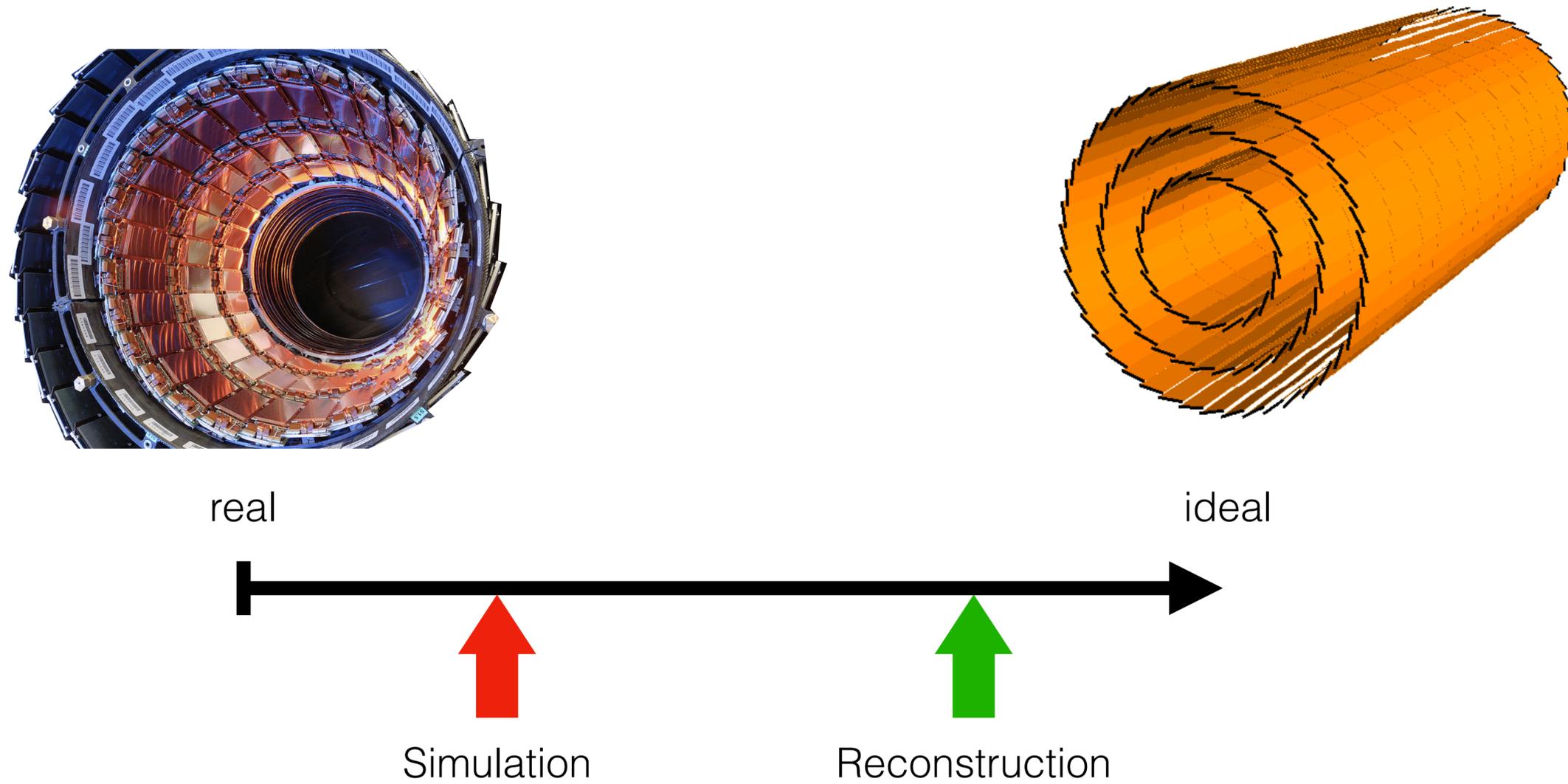
1

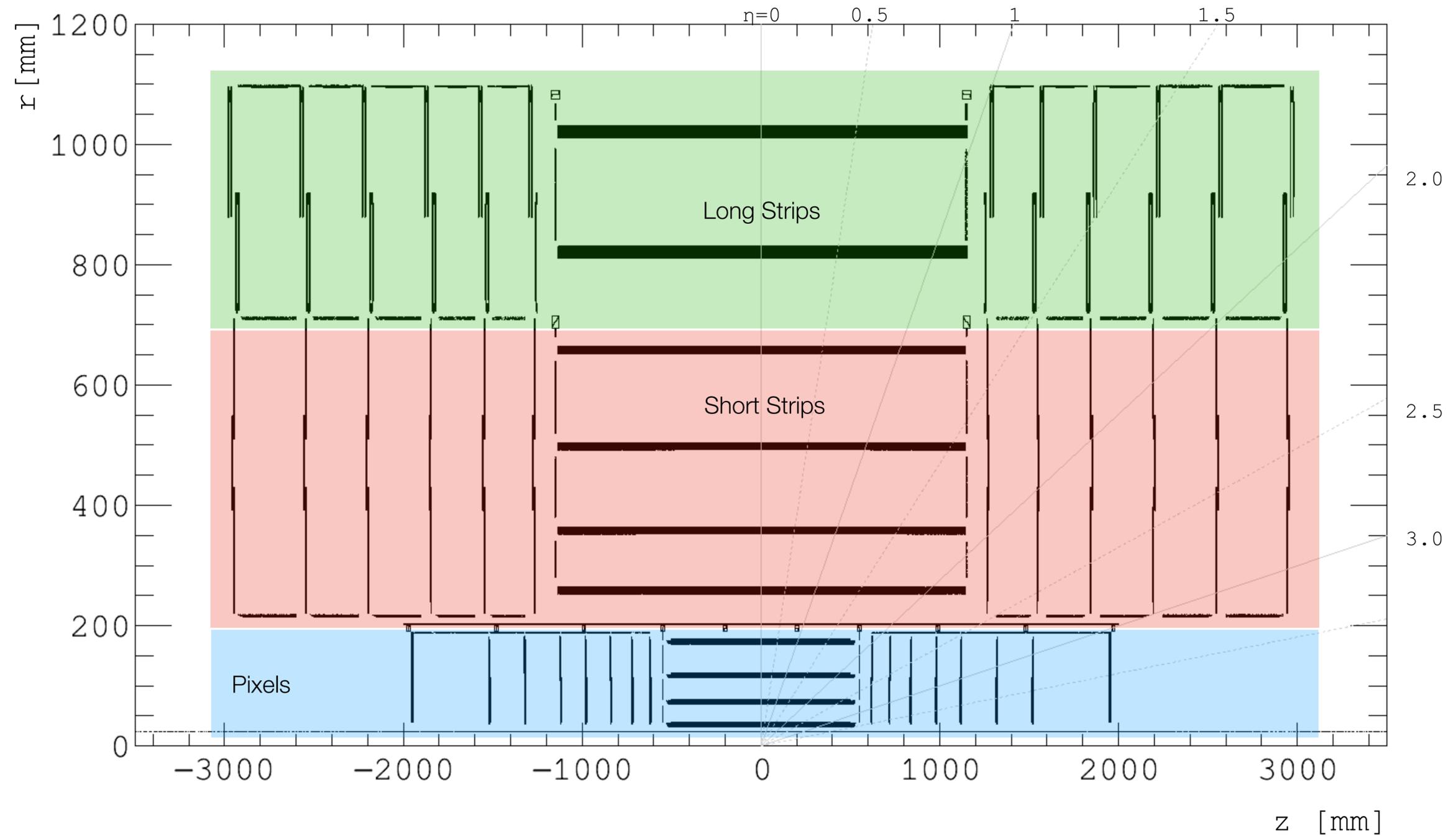


18

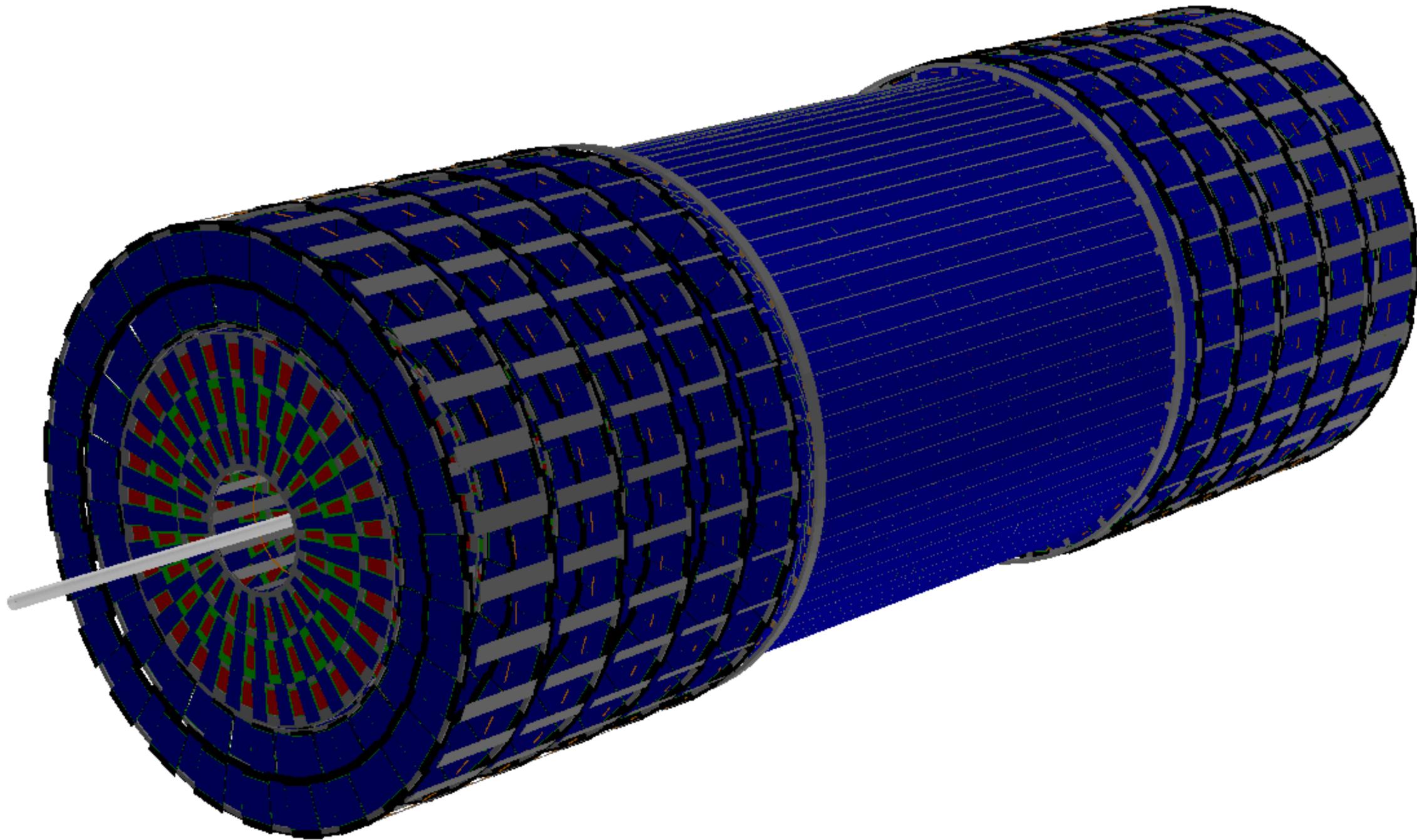
updated 3 days ago

Where's the right working point?

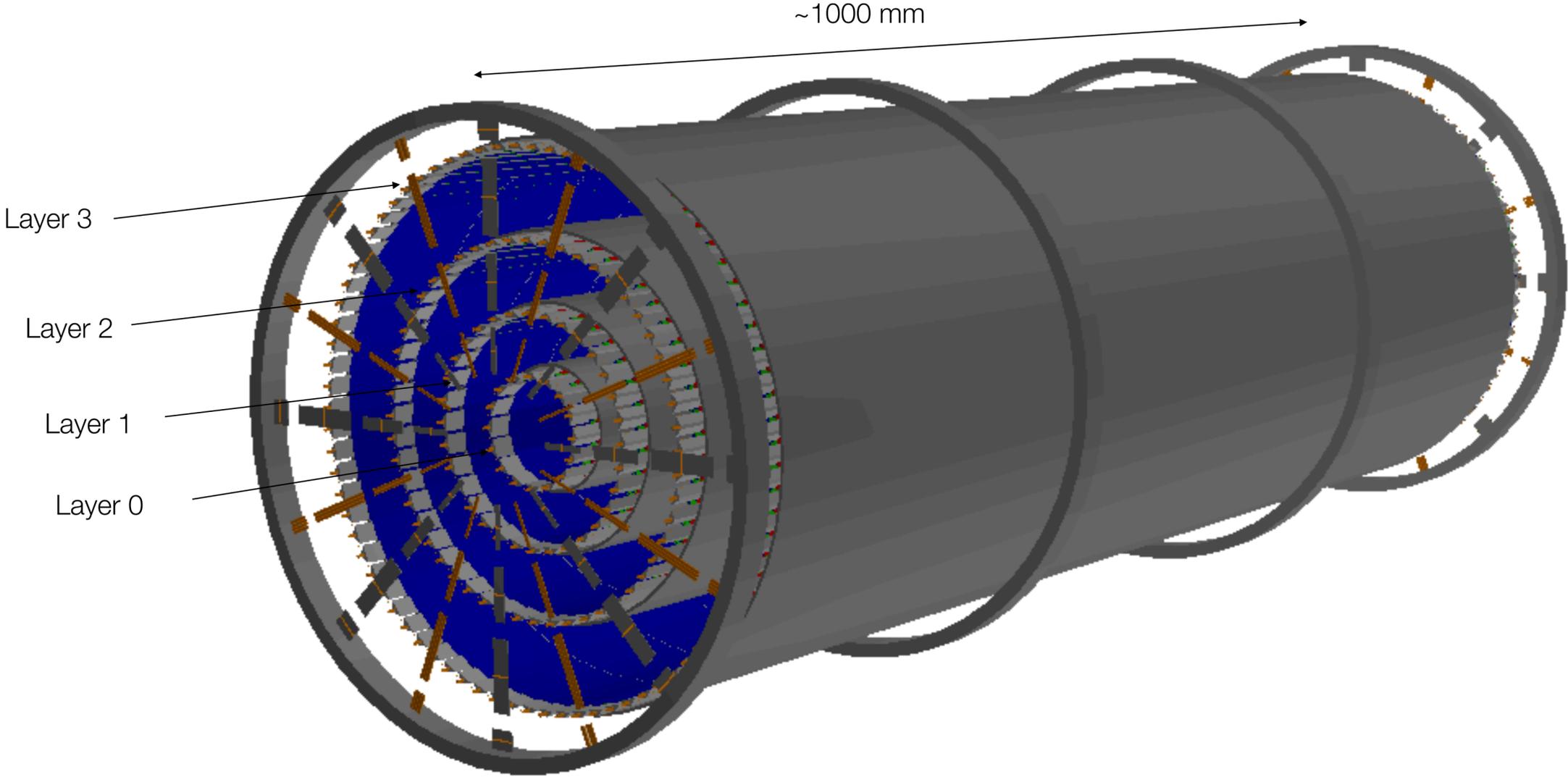




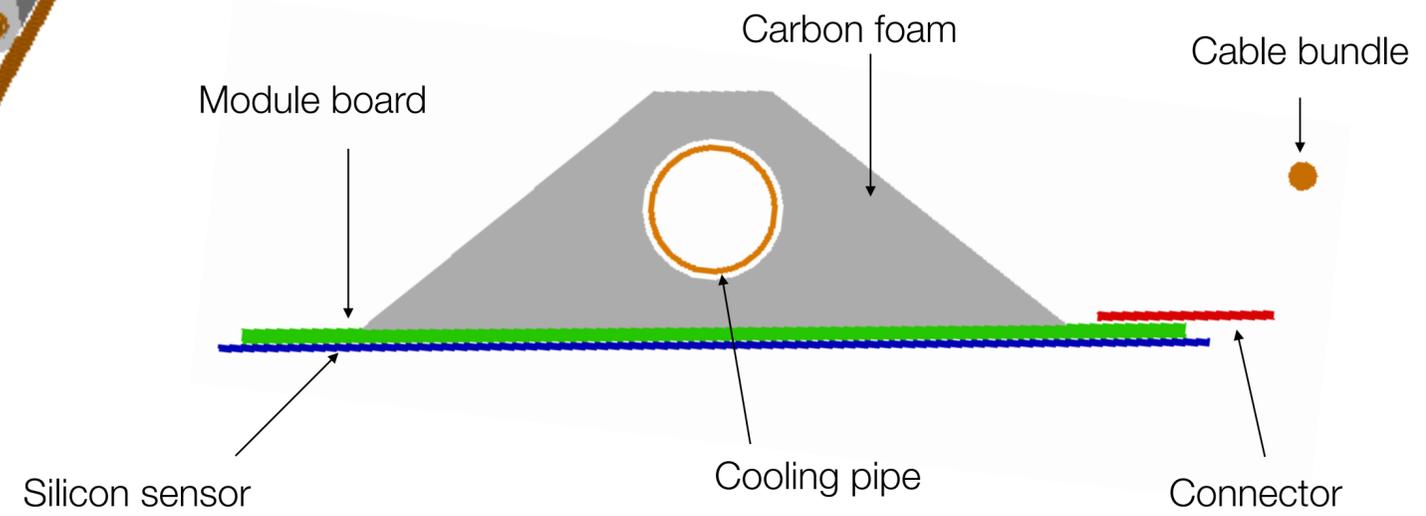
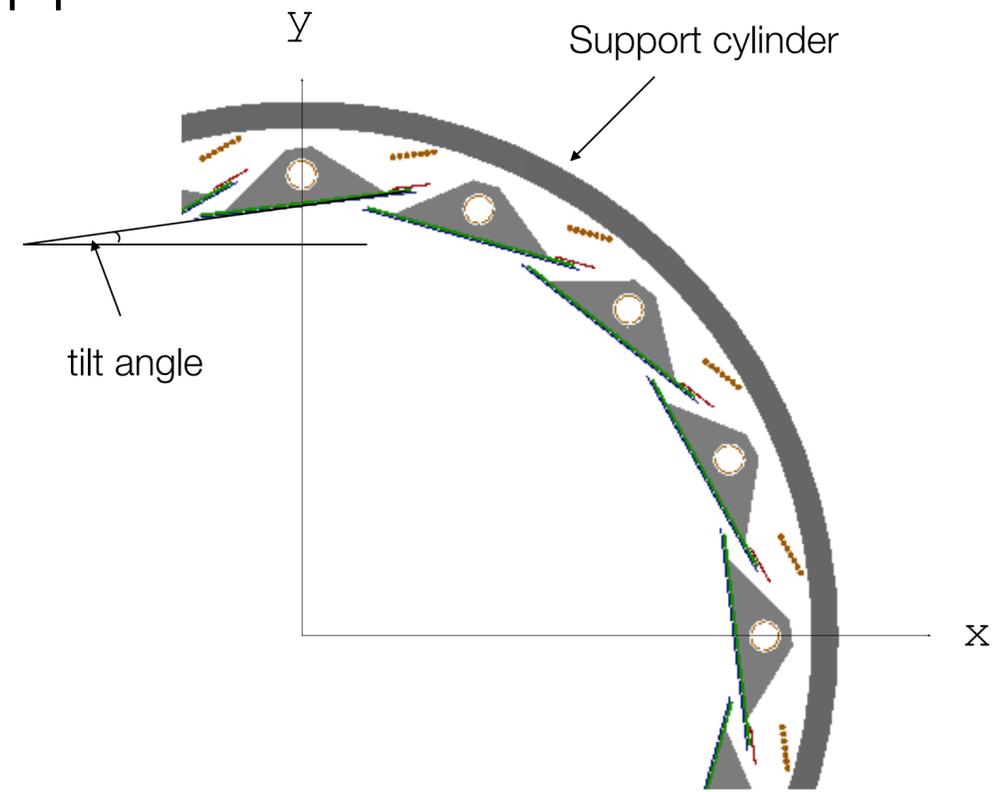
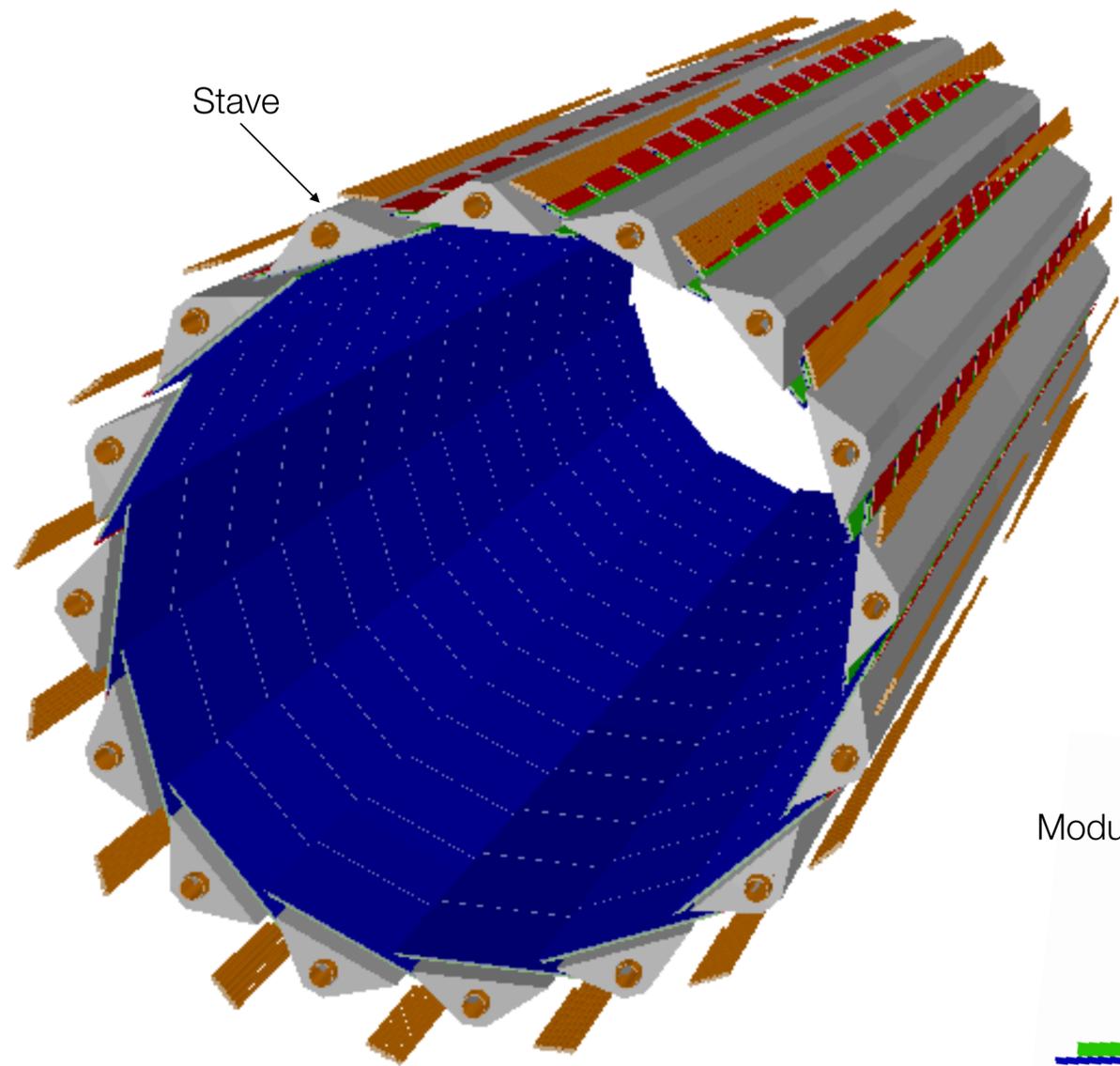
Full tracker view



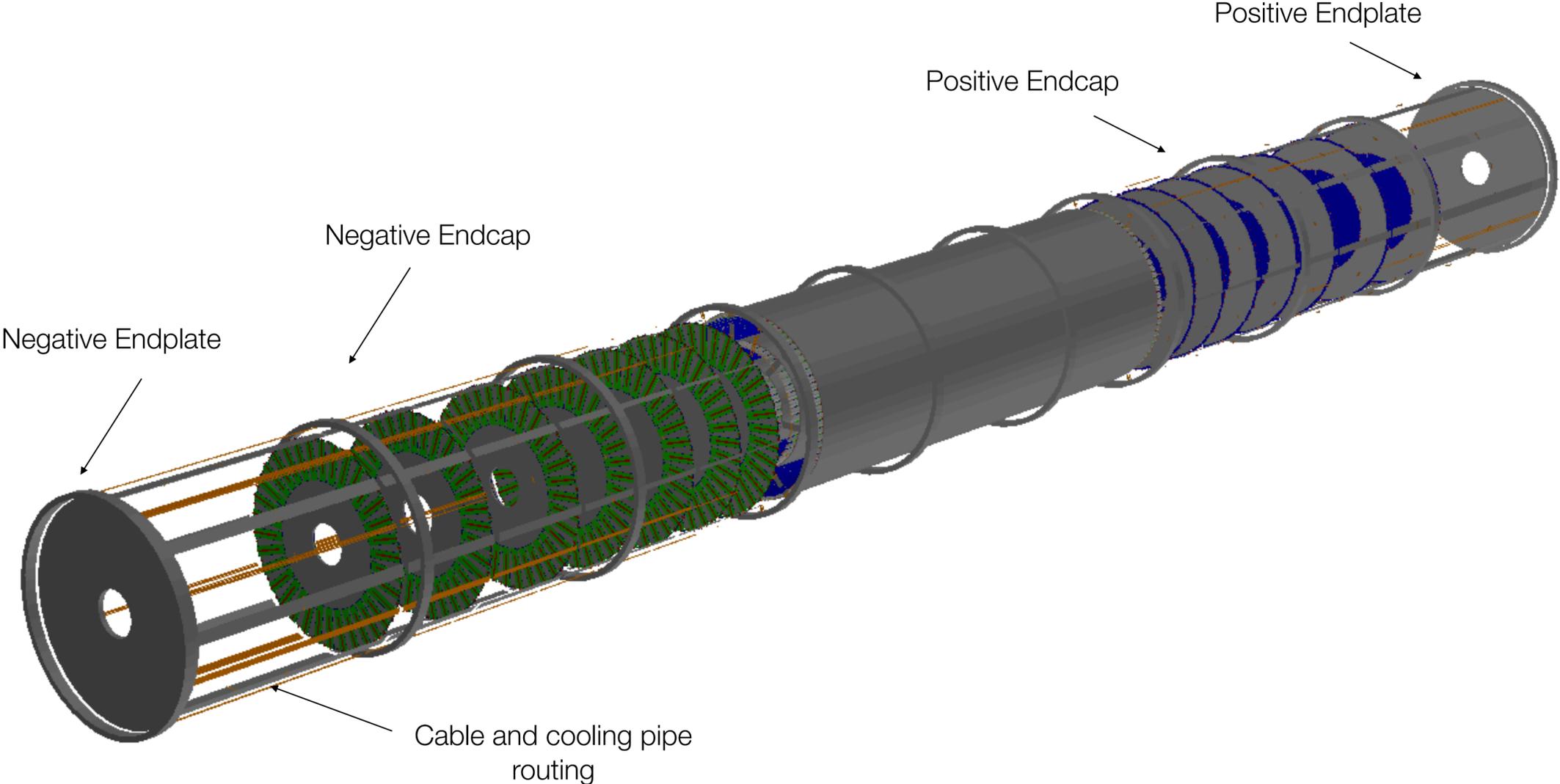
Pixel barrel system



Innermost layer definition



Full pixel system



Pixel end cap example

