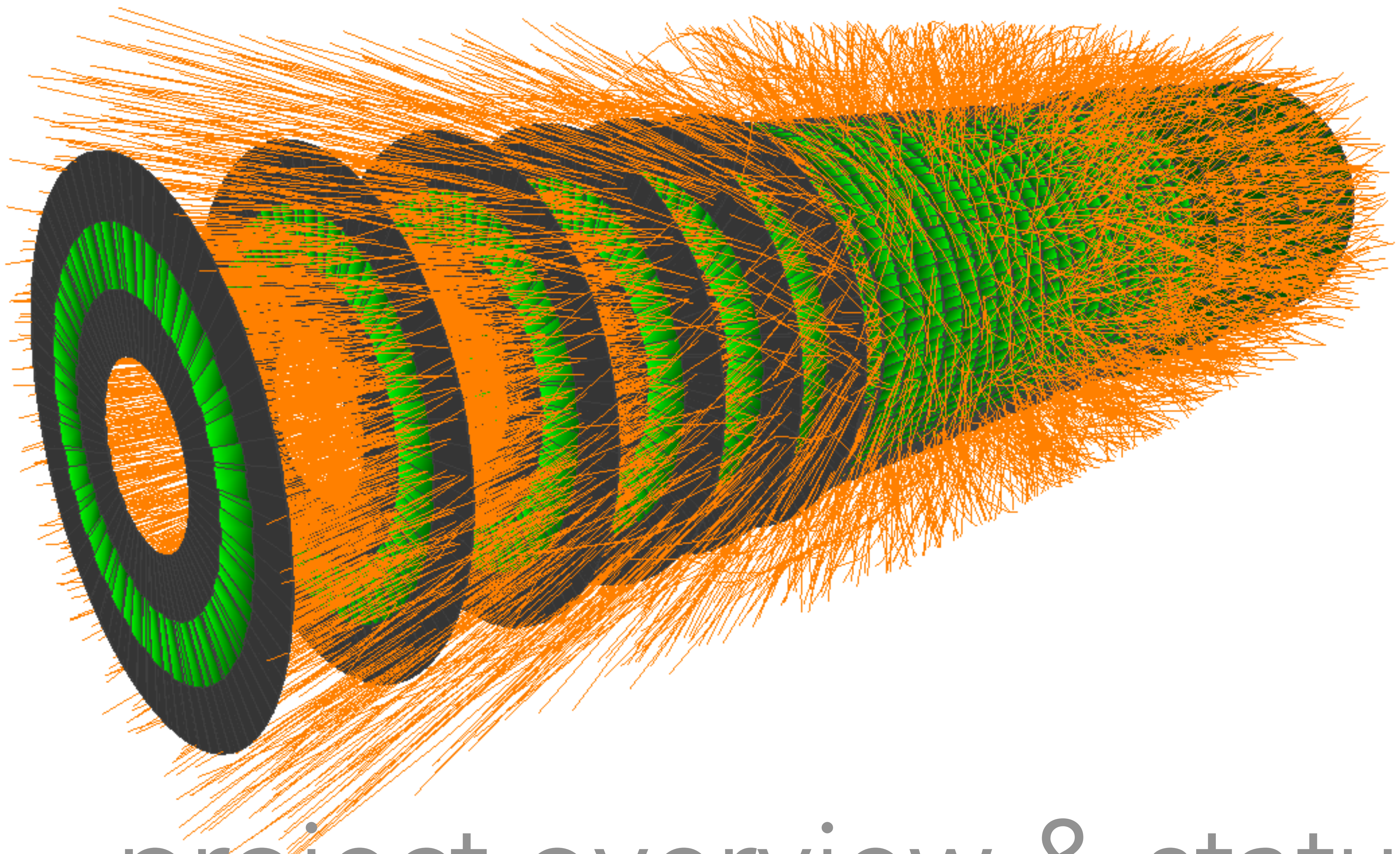
International Workshop on Future Tau Charm Facilities, Jan 14-18, 2024



acts project overview & status

supported by

cooperations

A. Salzburger (CERN) for the ACTS project

Development and R&D

Core

acts-developers@cern.ch

CPU multi-threaded library of
tracking reconstruction components

R&D1

acts-parallelization@cern.ch

CPU/GPU "single source" demonstrator
re-implementing the main Core chain

R&D2

acts-machinelearning@cern.ch

Machine learning and ML assisted
modules for track reconstruction

Core

acts-developers@cern.ch

CPU multi-threaded library of
tracking reconstruction components

Project has **5** years of age
More than **1500** merged PRs, more than **120** forks, ~**80** stars
~**50** different contributors

Core: the flagship project

Main target & language

- x86/ARM64 **multithreaded** architectures, GPU development moved to R&D1 line
- C++17 standard (we started moving towards C++20)
- minimal core dependencies:
  CMake, Eigen, BOOST **+ optional Plugins**

Component library structure

- track & vertex reconstruction components that allow for assembling of a track reconstruction applications for different experimental setups

- designed to be integrated into an experiment software that provides scheduling, configuration, IO, conditions data, etc.

Core concepts: multi threading and contextuatlity

Built-in parallelisation support

```
namespace Acts {
  class Module {
   /// @param gctx the geometry context (e.g. alignment)
   /// @param input the input data
   OutputData geometricOperation(const GeometryContext& gctx,const InputData& input) const;

  };
}
```


Allows parallel execution of this operation (without explicit technology binding, such as `tbb`) within and across events, nested **`State`** structs are used for necessary caching operations

```
namespace Acts {
  class Module {
   /// Nested State struct
   struct State { ActsScalar cache = 0.;};
   /// @param state is a cache for this operation
   /// @param input the input data
   OutputData operationWithCache(State& state, const InputData& input) const;

  };
}
```

Core concepts: multi threading and contextuatlity

Built-in parallelisation support and contextuality

```
namespace Acts {
   /// @param gctx the geometry context (e.g. alignment)
   /// @param input the input data
   OutputData geometricOperation(const GeometryContext& gctx,const InputData& input) const;

  };
}
```



```
using GeometryContext = std::any;
```

ACTS allows you to pack your own contextual data into the context objects (geometry, magnetic, field) and will carry it through the code base (untouched)

```
auto Experiment::applyCorrection(const GeometryContext& gctx, const InputData& input) const {

   const Experiment::Payload& payload = std::any_cast<const Experiment::Payload&>(gctx);
}
```

Core concepts: data driven, configuration & options

Design convention for data driven design, configuration and option

```
namespace Acts {
  /// doxygen documentation
  class Module {
   /// @struct Config for this module,
    struct Config {
       ActsScalar globalParameter; ///< configure this module
   };

   /// @struct Options for this module, changeable on call
    struct Options {
       ActsScalar callParameter; ///< how the horse feels today
   };

   /// @param cfg the configuration struct for this module
   Module(const Config& cfg) : m_config(cfg){};

   /// @param input the input data
   OutputData operation(const InputData& input, const Options& opt) const;

  };
}
```

Core concepts: configuration binding

Simple Config structs on ACTS side

```
namespace Acts {
  /// doxygen documentation
  class WorkHorse {
   /// @struct Config for To
    struct Config {
       ActsScalar coatColor; ///< configure the coat color
       ActsScalar maxPath;    ///< set the max path this horse can run
    };
  };
}
```

Connection to experiment framework, e.g. Gaudi/Athena

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

Selected Core modules

Geometry/Detector*
(Surface based geometry)

Plugins to DD4hep, TGeo, etc.

Event Data Model
target track reconstruction

backend separation
with different I/O models

Propagation
parameter + covariance
transport through
magnetic field

Seeding
Seed finding with
Triplet seeder,
OrthogonalSeedFinder

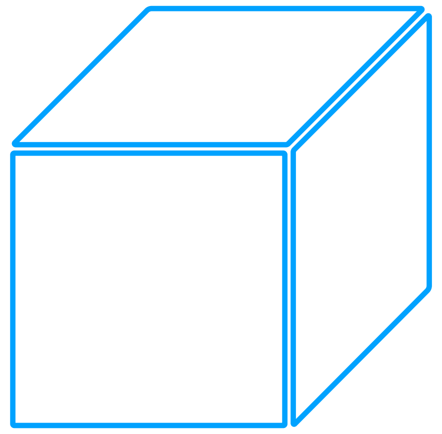
Track Fitting
parameter estimation
with Kalman Filter,
GSF, GX2F**

Combinatorial track finding
Combinatorial Kalman Filter
for track finding

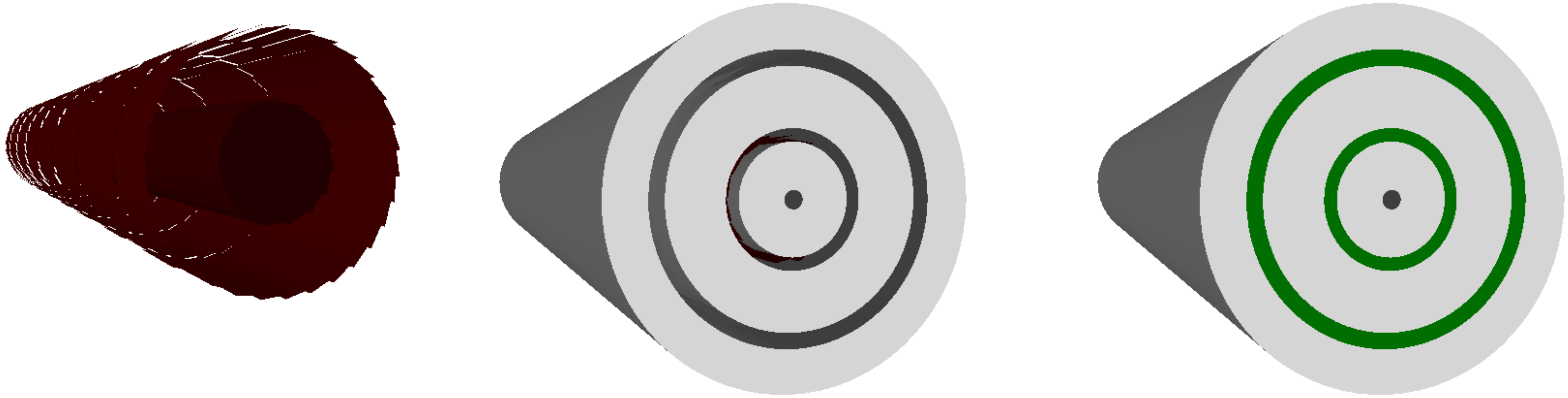
Vertex finding + fitting
Iterative, multi variant
primary vertex finders
and fitters

Detector alignment
KF based alignment
functionality

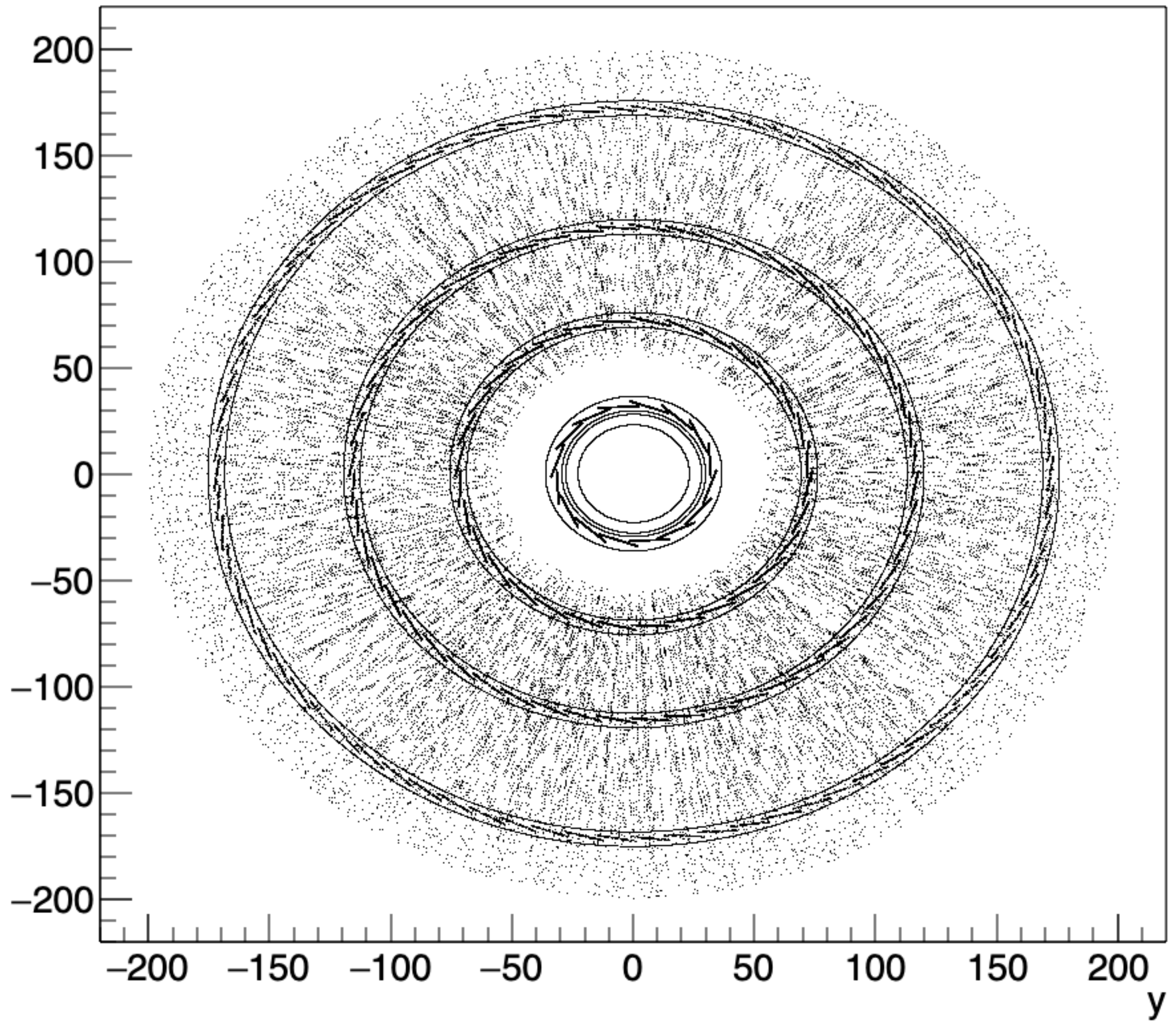
***new geometry model to be deployed in 2023, **GX2F prototype to come 2024**

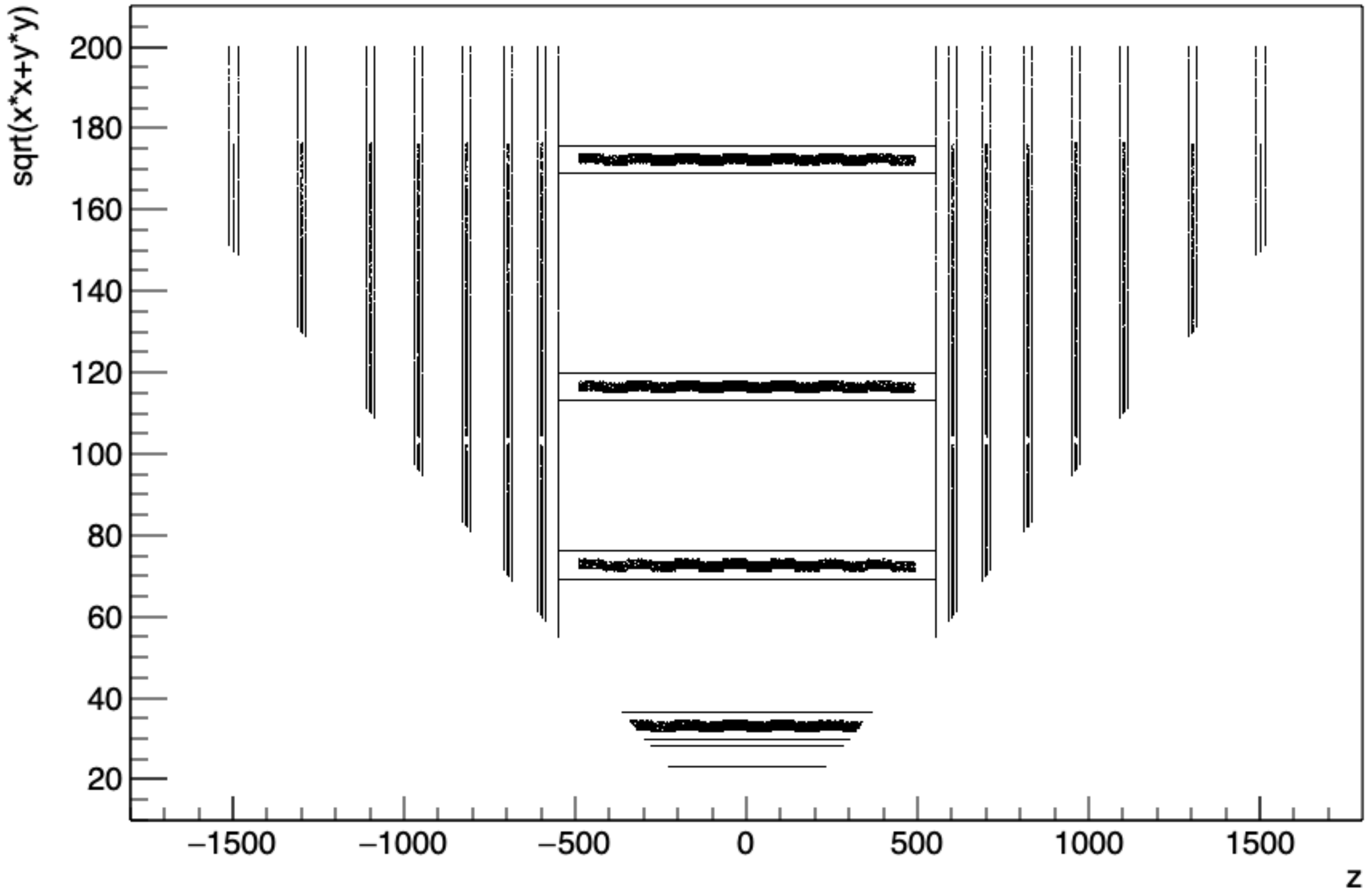


Geometry + Navigation



Surface based geometry model, optimised for reconstruction, can be **connected to standard HEP geometry modellers**





Geometry + Interfacing

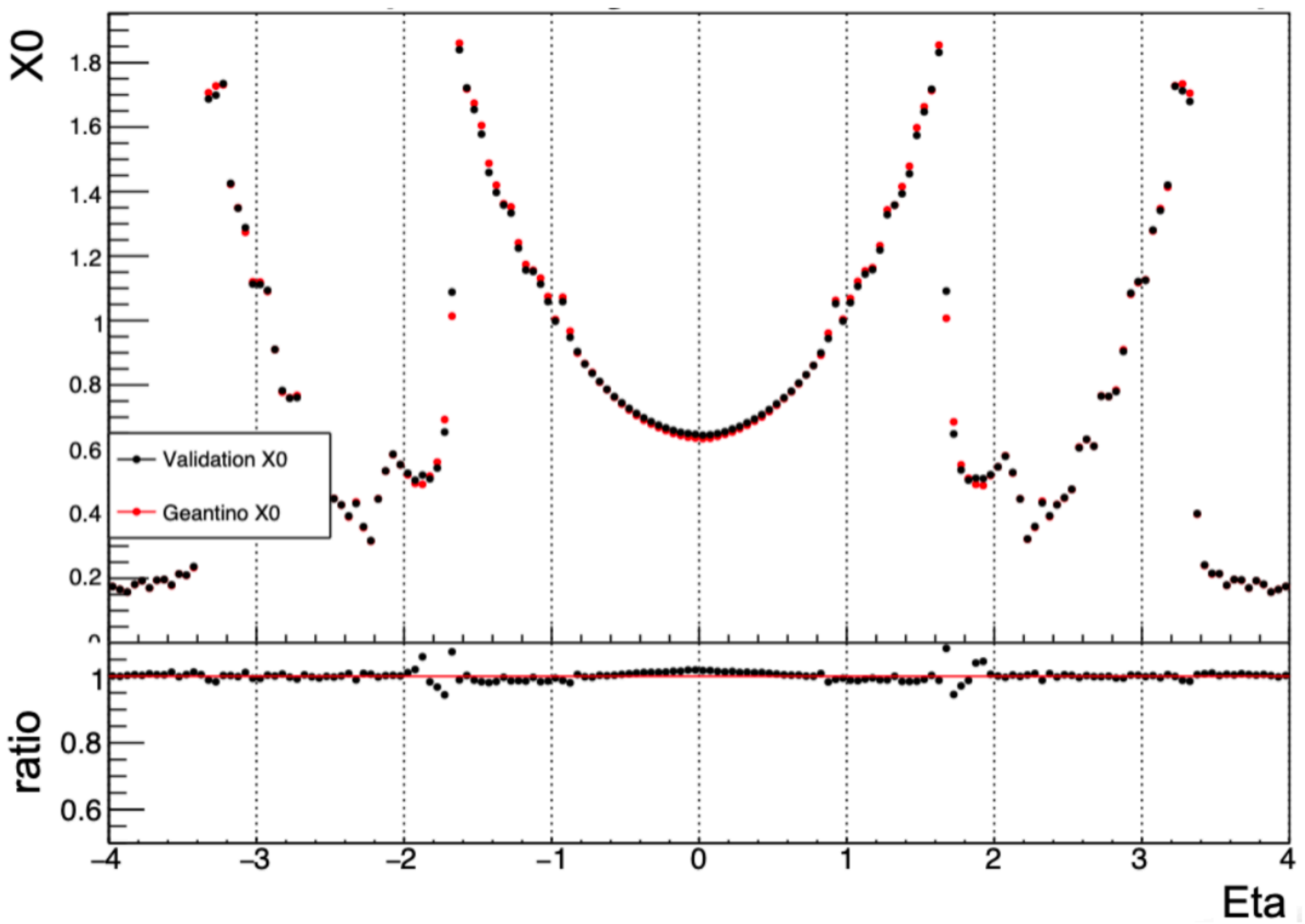
connected to standard HEP geometry modellers

Detailed geometry model,
e.g. DD4hep, TGeo, GeoModel, etc.

ACTS geometry model
with builtin navigation

Geometry + Material

ACTS ships with a **material mapping module** that allows to describe the detector material with close-to Geant4 precision

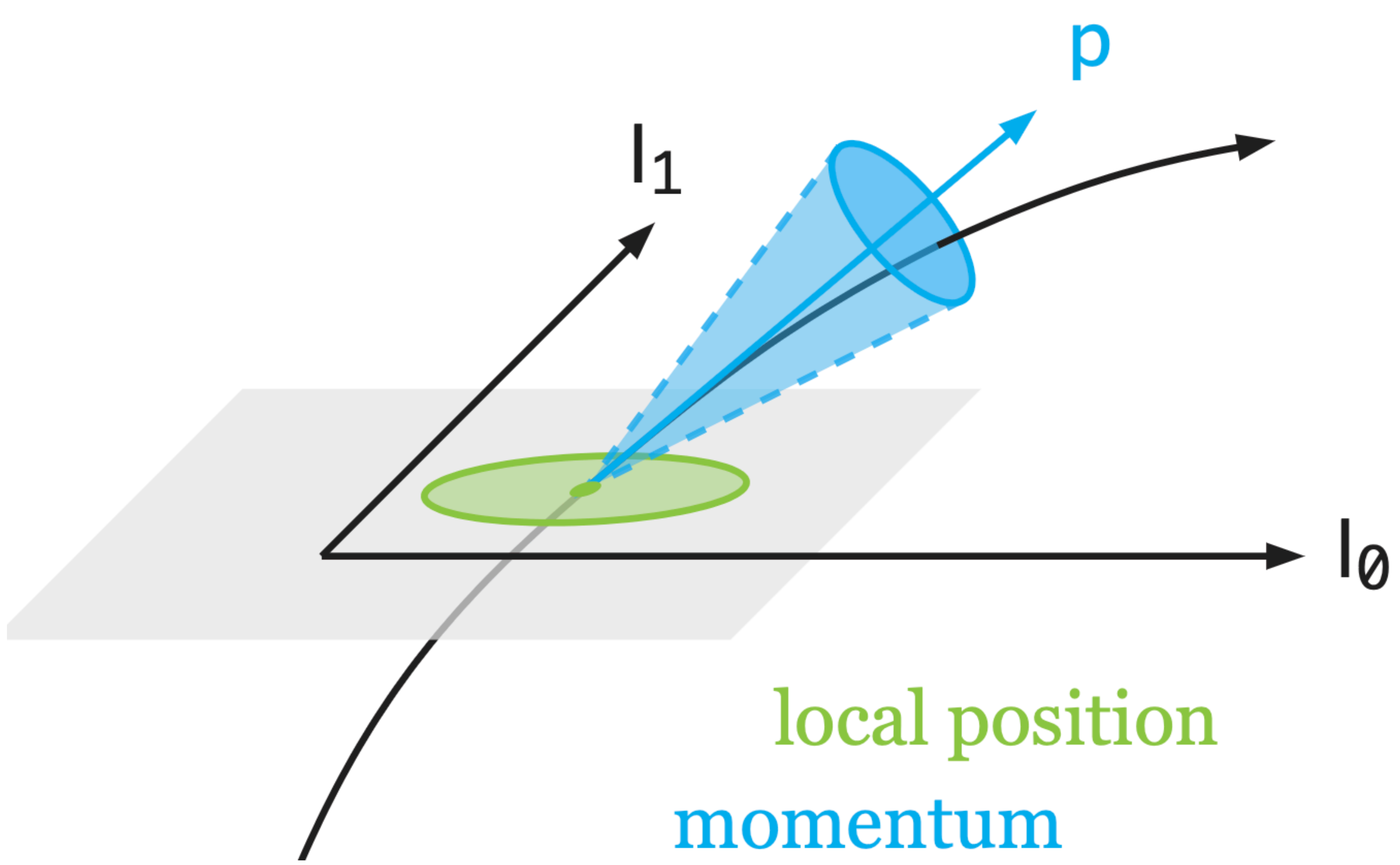


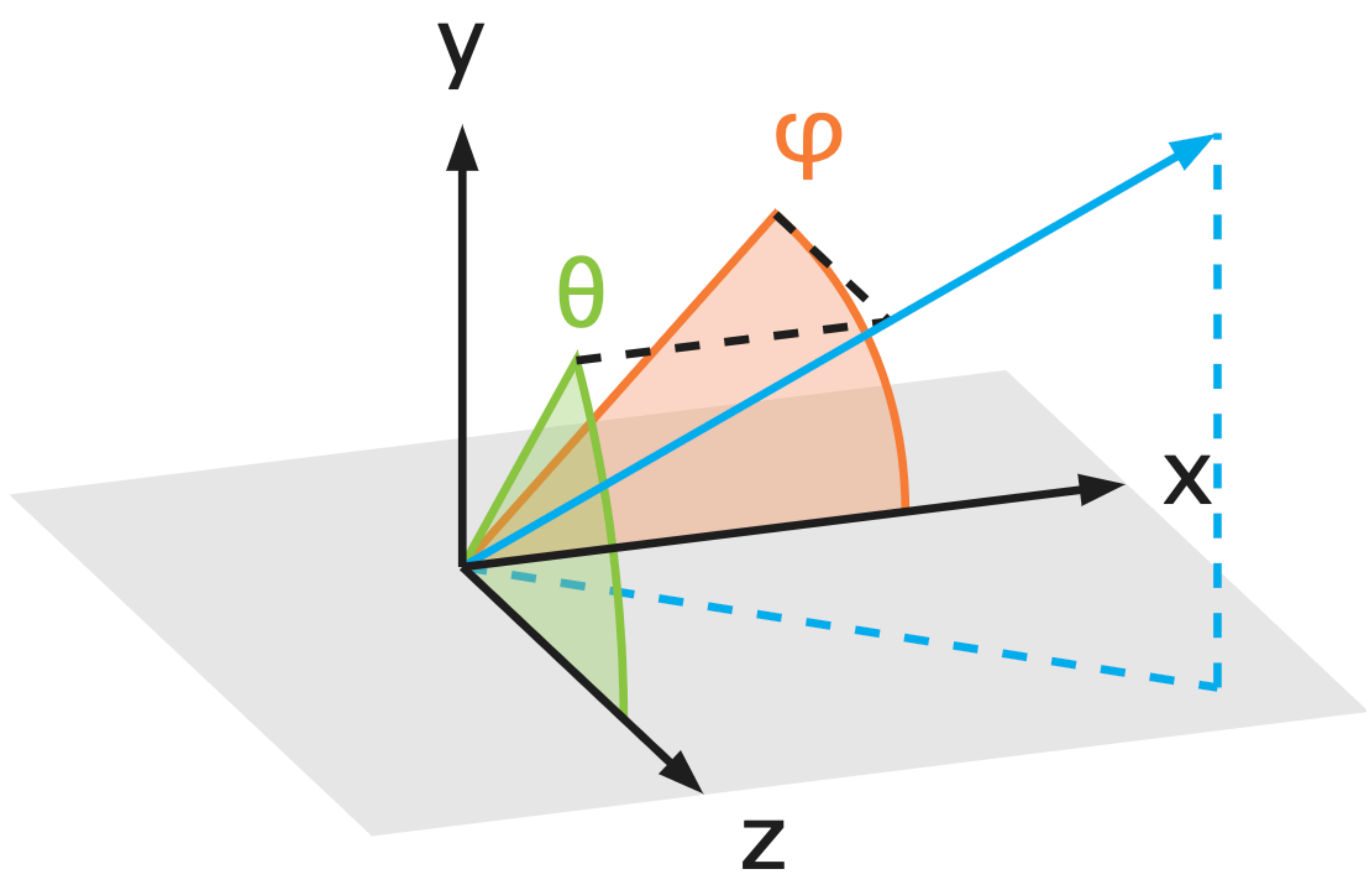
*** this can also be enhanced with a ML based material optimiser**

Event Data Model

(Bound) track parameterisation is defined:

local coordinates of the surface + global momentum





$$\vec{x} = (l_0, l_1, \phi, \theta, q/p, t)^T$$

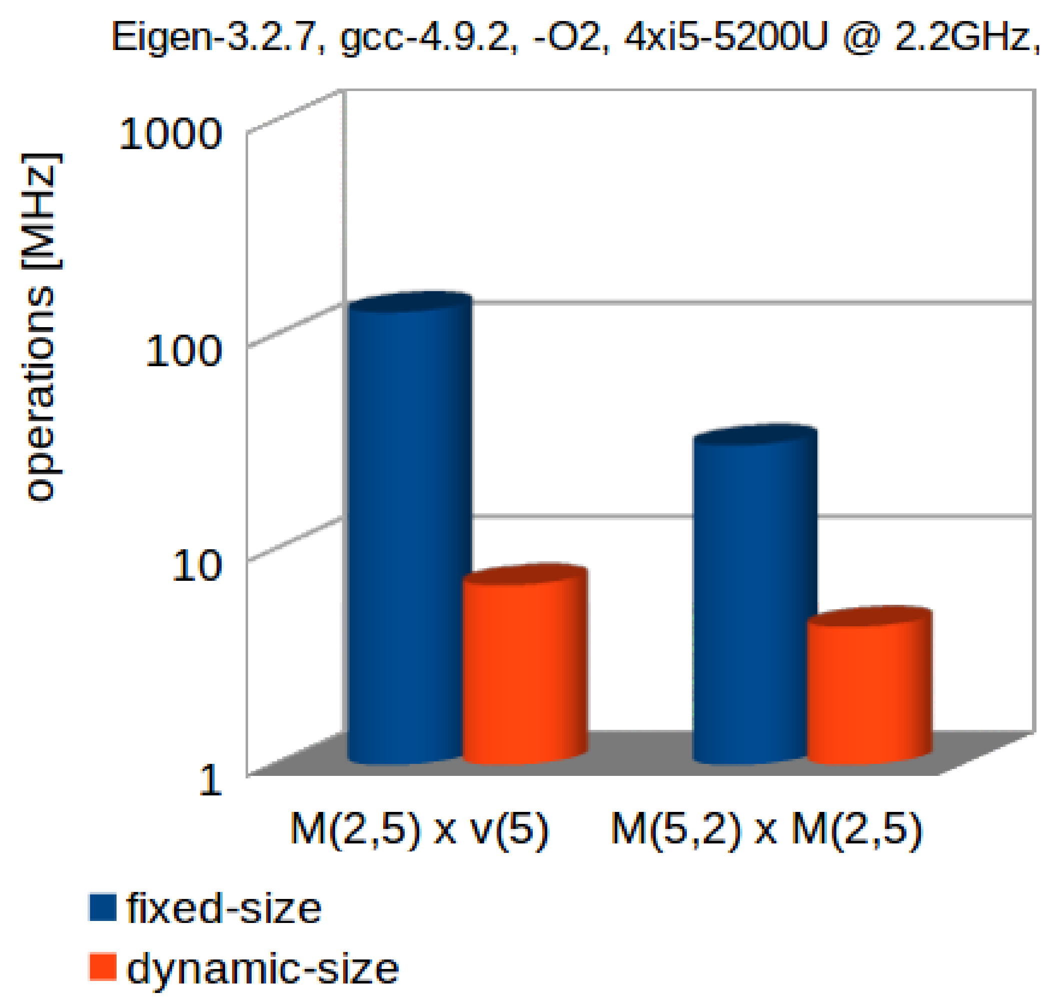
$$C = \begin{bmatrix} \sigma^2(l_0) & \mathrm{cov}(l_0,l_1) & \mathrm{cov}(l_0,\phi) & \mathrm{cov}(l_0,\theta) & \mathrm{cov}(l_0,q/p) \\ . & \sigma^2(l_1) & \mathrm{cov}(l_1,\phi) & \mathrm{cov}(l_1,\theta) & \mathrm{cov}(l_1,q/p) \\ . & . & \sigma^2(\phi) & \mathrm{cov}(\phi,\theta) & \mathrm{cov}(\phi,q/p) \\ . & . & . & \sigma^2(\theta) & \mathrm{cov}(\theta,q/p) \\ . & . & . & . & \sigma^2(q/p) \end{bmatrix}$$

Event Data Model

Parameter	l_0	l_1	phi	theta	q/p	t
Bound track parameters						
Pixel measurement						
Pixel measurement with time						
Strip measurement (along local x)						
Strip measurement (along local y)						
Drift time/circle measurement						
Track segment (straight line)						
...						

Measurements can be represented as subsets of the full bound parameter space.

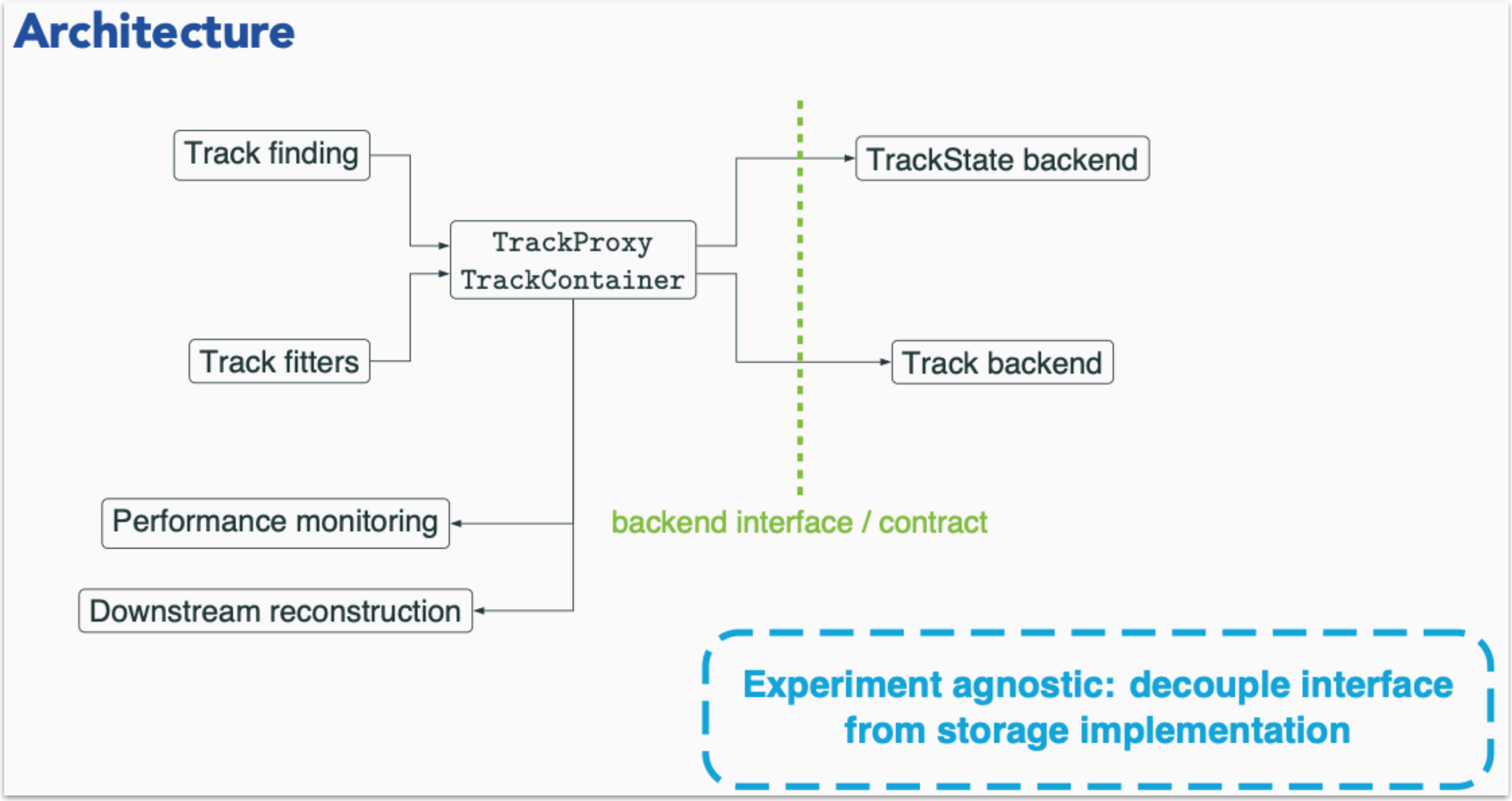
This is done at compile time to increase computing performance.



Event Data Model + I/O backend

ACTS has an internal EDM optimised for track reconstruction.

- recent work to separate transient model from I/O backend
- demonstrator with PODIO established
- Non-optimised EDM4Hep version also available



[Paul Gessinger-Befurt, CHEP2023, Parallel talk]

Propagation

Stepping loop
(until max number of steps)

`Navigator::status:`
On a current surface?

Walk through actor list

Walk through aborter list

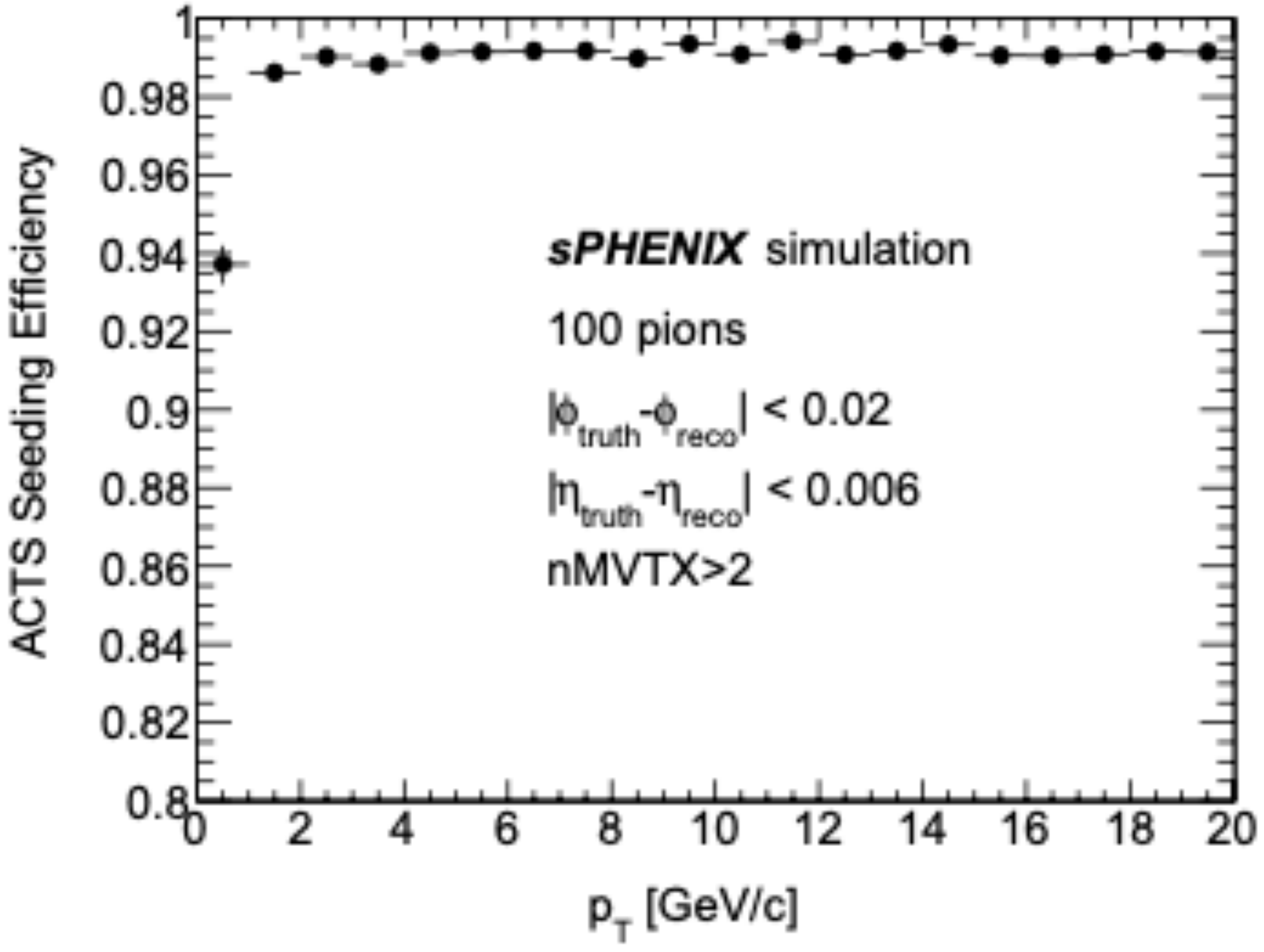
`Navigator::target`
Next candidate surface

```cpp
// Pre-Stepping: abort condition check
if (!state.options.abortList(result, state, m_stepper)) {
  // Pre-Stepping: target setting
  m_navigator.target(state, m_stepper);
  // Stepping loop
  ACTS_VERBOSE("Starting stepping loop.");

  terminatedNormally = false;  // priming error condition

  // Propagation loop : stepping
  for (; result.steps < state.options.maxSteps; ++result.steps) {
    // Perform a propagation step - it takes the propagation state
    Result<double> res = m_stepper.step(state);
    if (res.ok()) {
      // Accumulate the path length
      double s = *res;
      result.pathLength += s;
      ACTS_VERBOSE("Step with size = " << s << " performed");
    } else {
      ACTS_ERROR("Step failed with " << res.error() << ": "
                                     << res.error().message());
      // pass error to caller
      return res.error();
    }
    // Post-stepping:
    // navigator status call - action list - aborter list - target call
    m_navigator.status(state, m_stepper);
    state.options.actionList(state, m_stepper, result);
    if (state.options.abortList(result, state, m_stepper)) {
      terminatedNormally = true;
      break;
    }
    m_navigator.target(state, m_stepper);
  }
} else {
  ACTS_VERBOSE("Propagation terminated without going into stepping loop.");
}
```

Seeding

Triplet finding for initial track seeding, two alternative implementations:

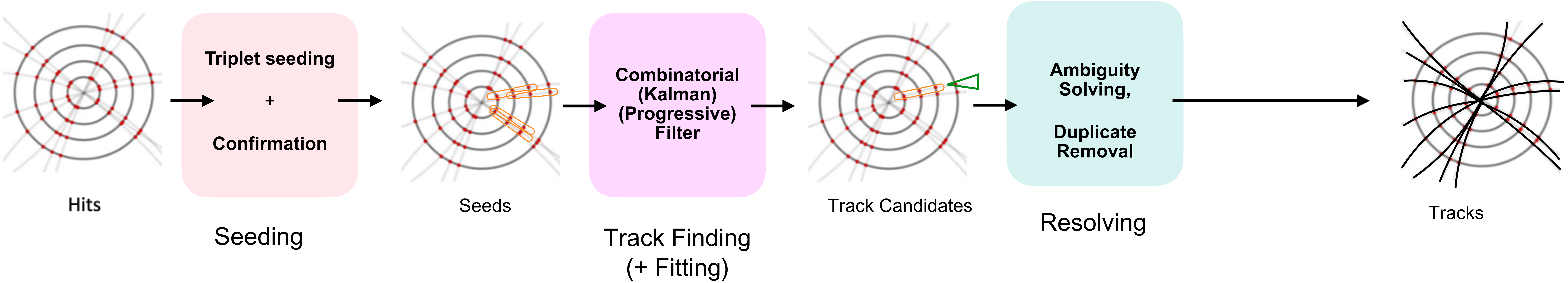
- ATLAS inspired mid-point seeding
- Orthogonal Seed finding



Output of seeding is input to the Combinatorial Track finding module.

Full chain prototype implementation

CKF chain

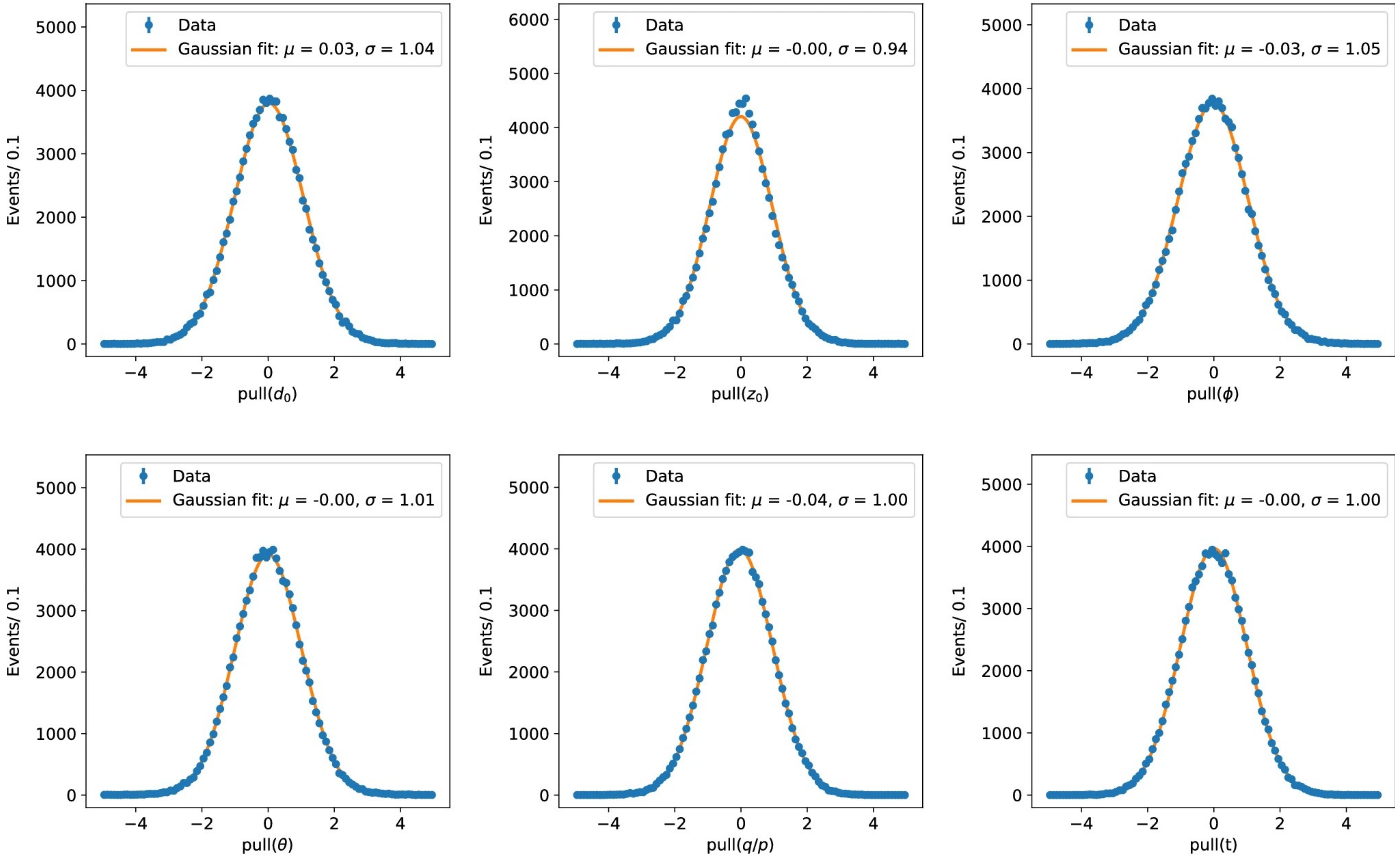


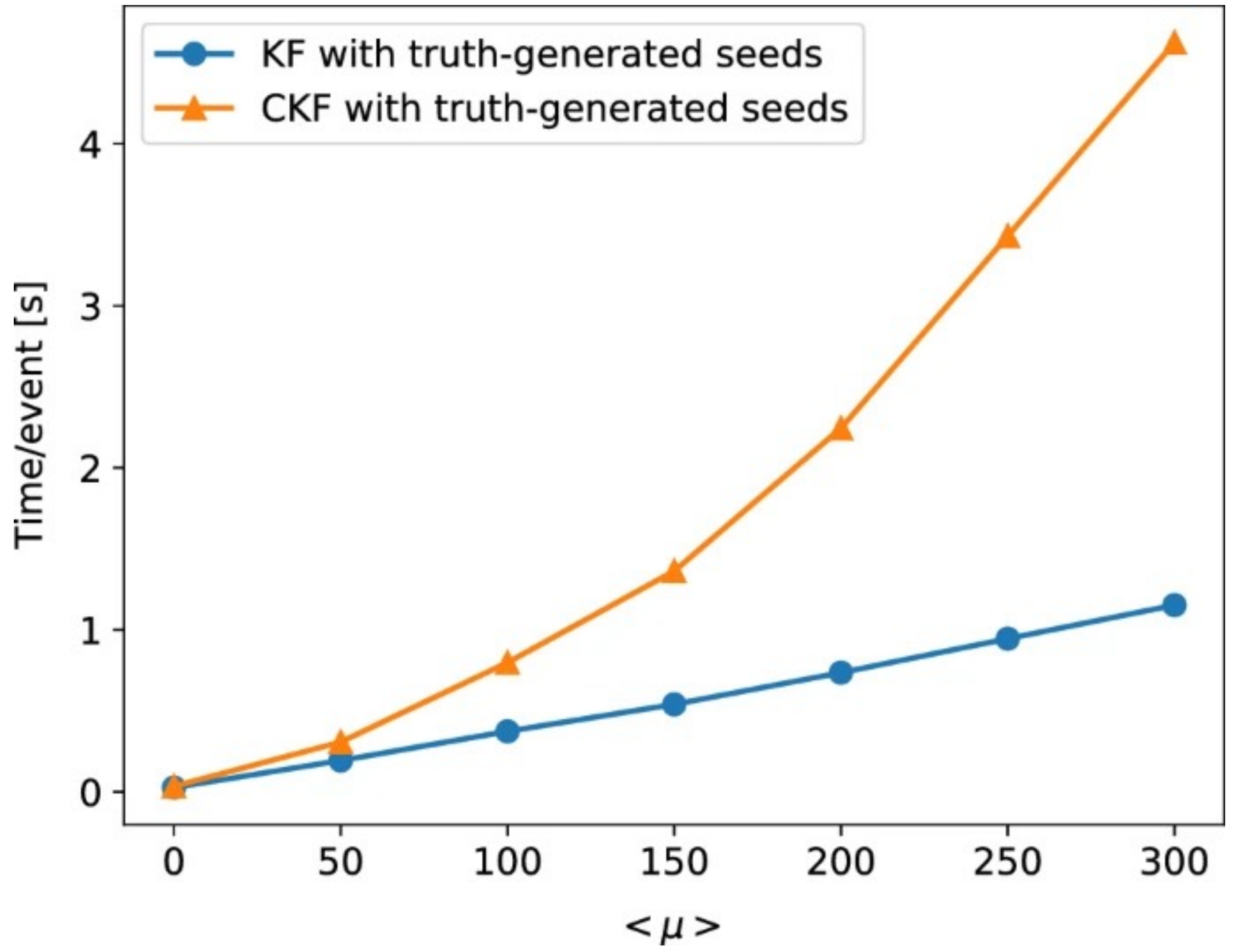
This resembles the ATLAS track reconstruction strategy

However, this is only one possible implementation put together by ACTS tools
(That's the spirit of a component library)

Full chain - with combinatorial track finding

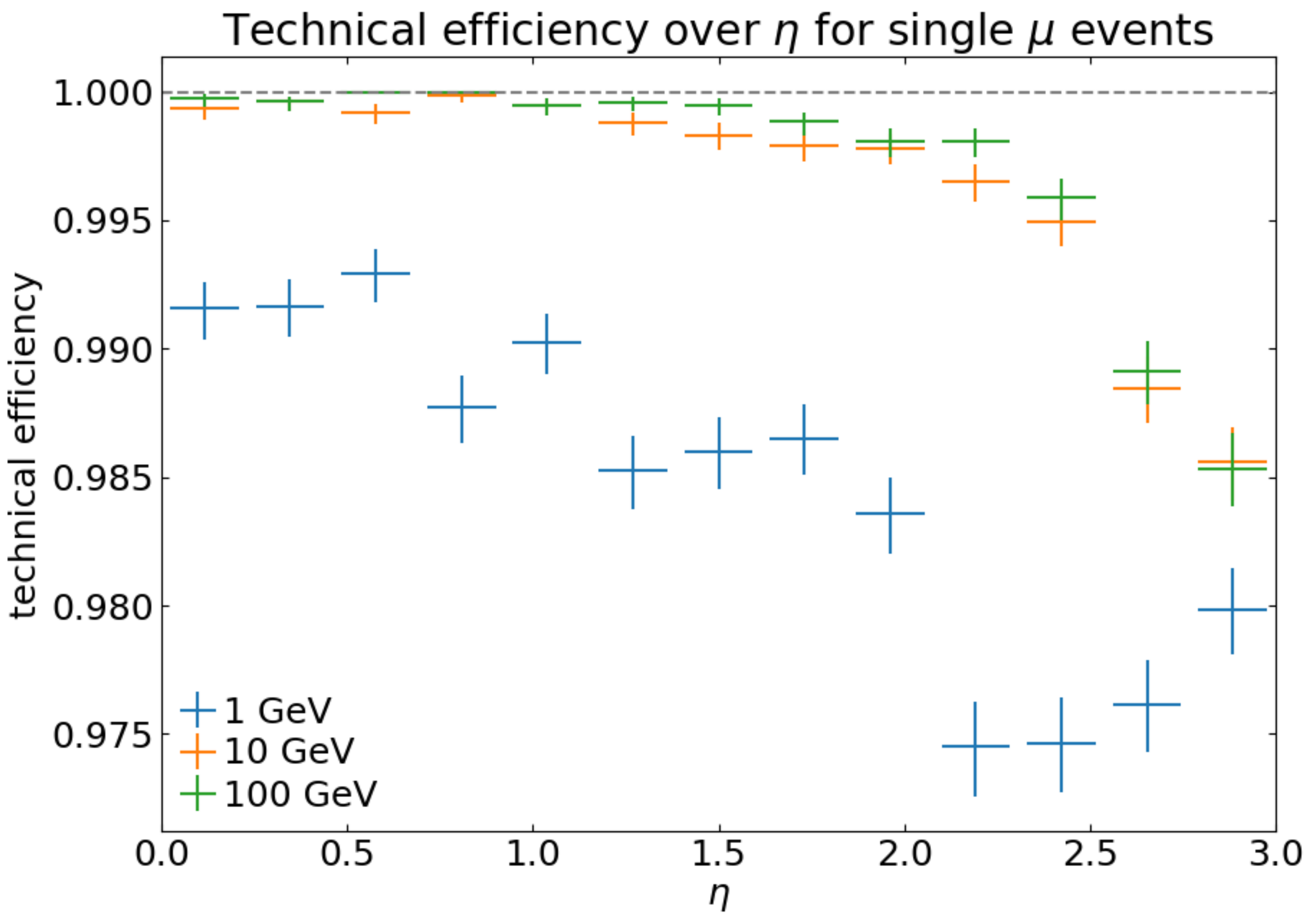
A first full chain documented using the Open Data Detector / TrackML detector in:
https://link.springer.com/article/10.1007/s41781-021-00078-8

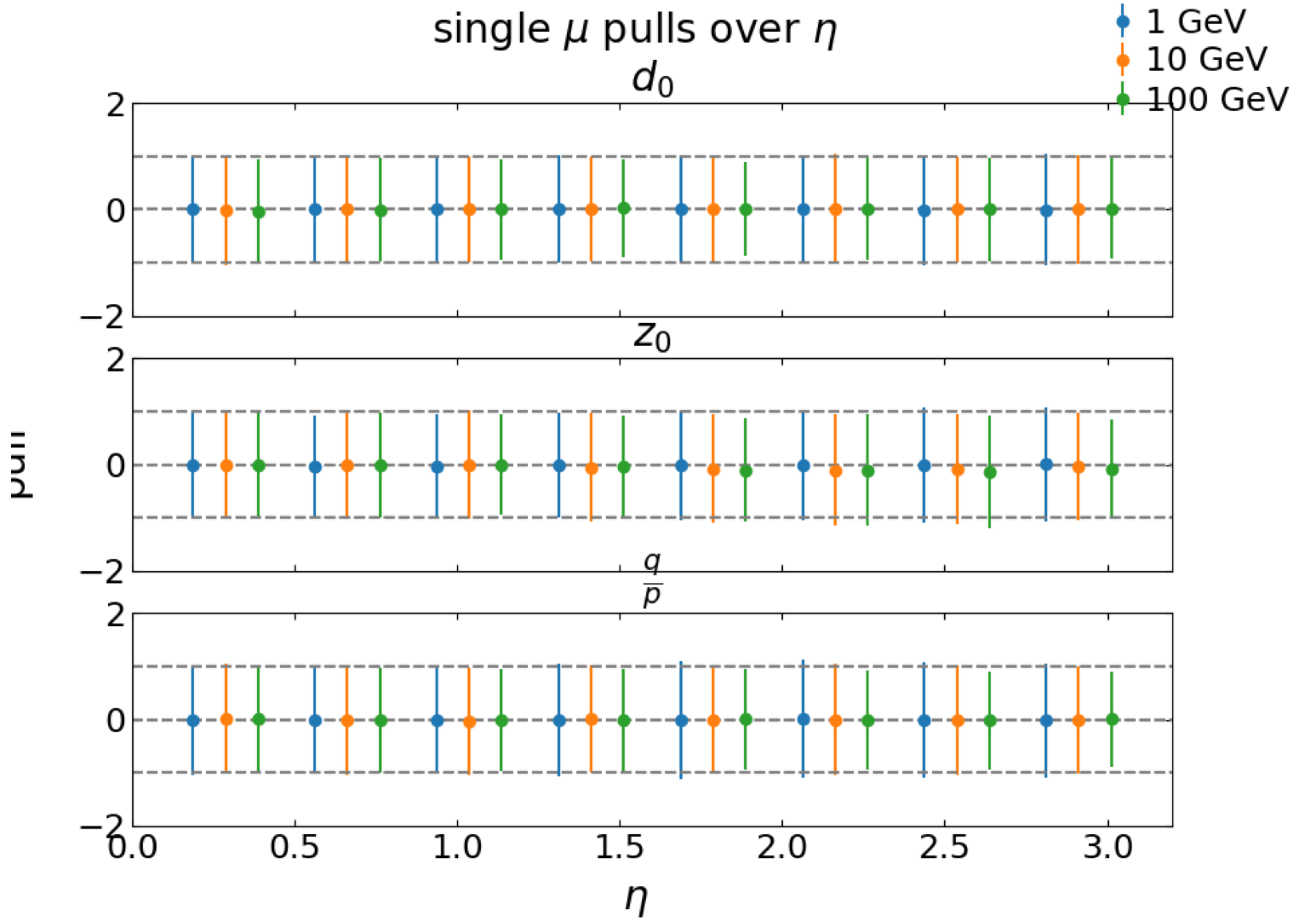




Full chain - with combinatorial track finding

Open Data Detector with Geant4 simulation





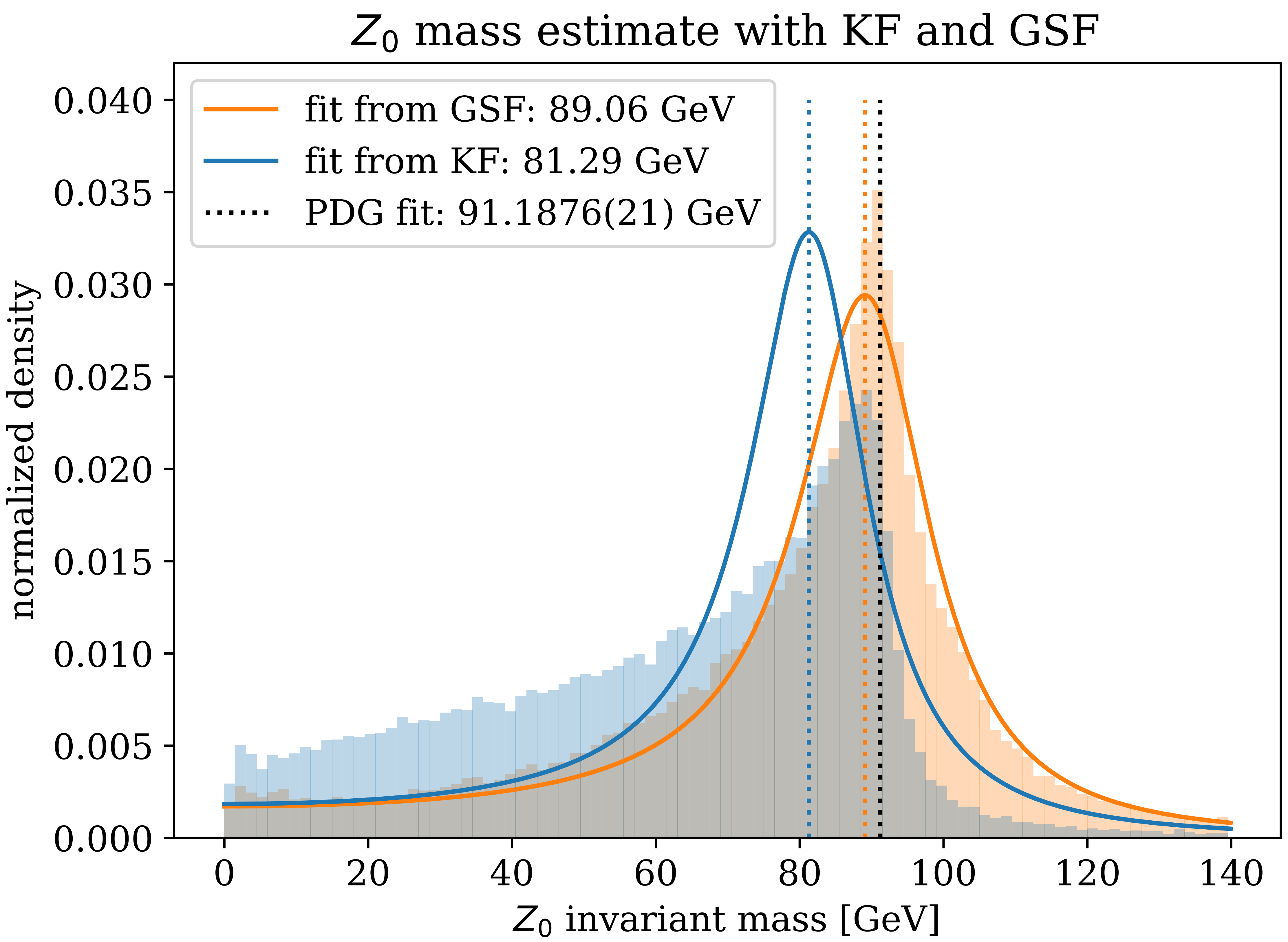
[Andreas Stefl, Connecting the Dots 2023]

Recently added functionality

First Gaussian Sum Filter (GSF) for electron reconstruction

The GSF is a multi-variant extension of the Kalman Filter to model non-Gaussian noise (i.e. Bremsstrahlung)

Global Chi2 Fitter currently in prototyping stage

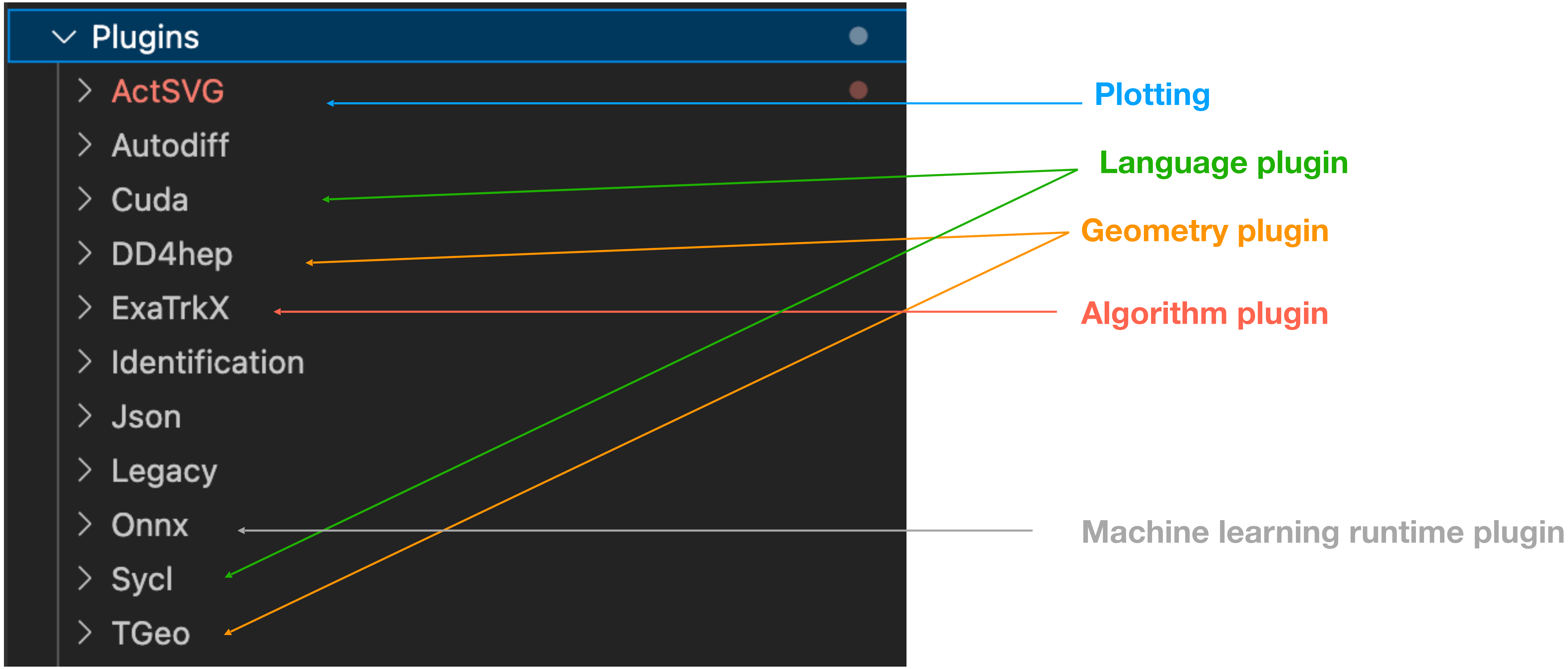


Aim is here to have interchangeable components.

Core: Extensibility

Plugin mechanism

- Library is extendable in functionality with several plugins
- Usually also pull in additional third party dependencies



Core: testing

Comprehensive **Unit testing** is one of the main targets of our development model

- Best practise: write the code & tests together
- Small testable units/modules is key to this

Based on BOOST unit testing framework,

`Codecov` (as part of CI) checks covering

```
namespace Acts {

using namespace detail;

namespace Test {

BOOST_AUTO_TEST_CASE(grid_test_1d_equidistant) {
  using Point = std::array<double, 1>;
  using indices = std::array<size_t, 1>;
  EquidistantAxis a(0.0, 4.0, 4u);
  Grid<double, EquidistantAxis> g(std::make_tuple(std::move(a)));

  // test general properties
  BOOST_CHECK_EQUAL(g.size(), 6u);
  BOOST_CHECK_EQUAL(g.numLocalBins().at(0), 4u);

  // global bin index
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{-0.3}})), 0u);
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{-0.}})), 1u);
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{0.}})), 1u);
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{0.7}})), 1u);
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{1}})), 2u);
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{1.2}})), 2u);
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{2.}})), 3u);
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{2.7}})), 3u);
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{3.}})), 4u);
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{3.9999}})), 4u);
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{4.}})), 5u);
  BOOST_CHECK_EQUAL(g.globalBinFromPosition(Point({{4.98}})), 5u);
```

Codecov Report

Merging #1551 (`59d52ae`) into main (`f3b20f7`) will **decrease** coverage by `0.00%`.
The diff coverage is `0.00%`.

```
@@           Coverage Diff           @@
##             main    #1551   +/-   ##
=======================================
- Coverage   48.47%   48.46%   -0.01%
=======================================
  Files         381      381
  Lines       20699    20702      +3
  Branches     9503     9504      +1
=======================================
  Hits        10034    10034
- Misses       4112     4115      +3
  Partials     6553     6553
```

Core: collaboration & contributing

ACTS is Open Source and invites contributions, corrections, interactions

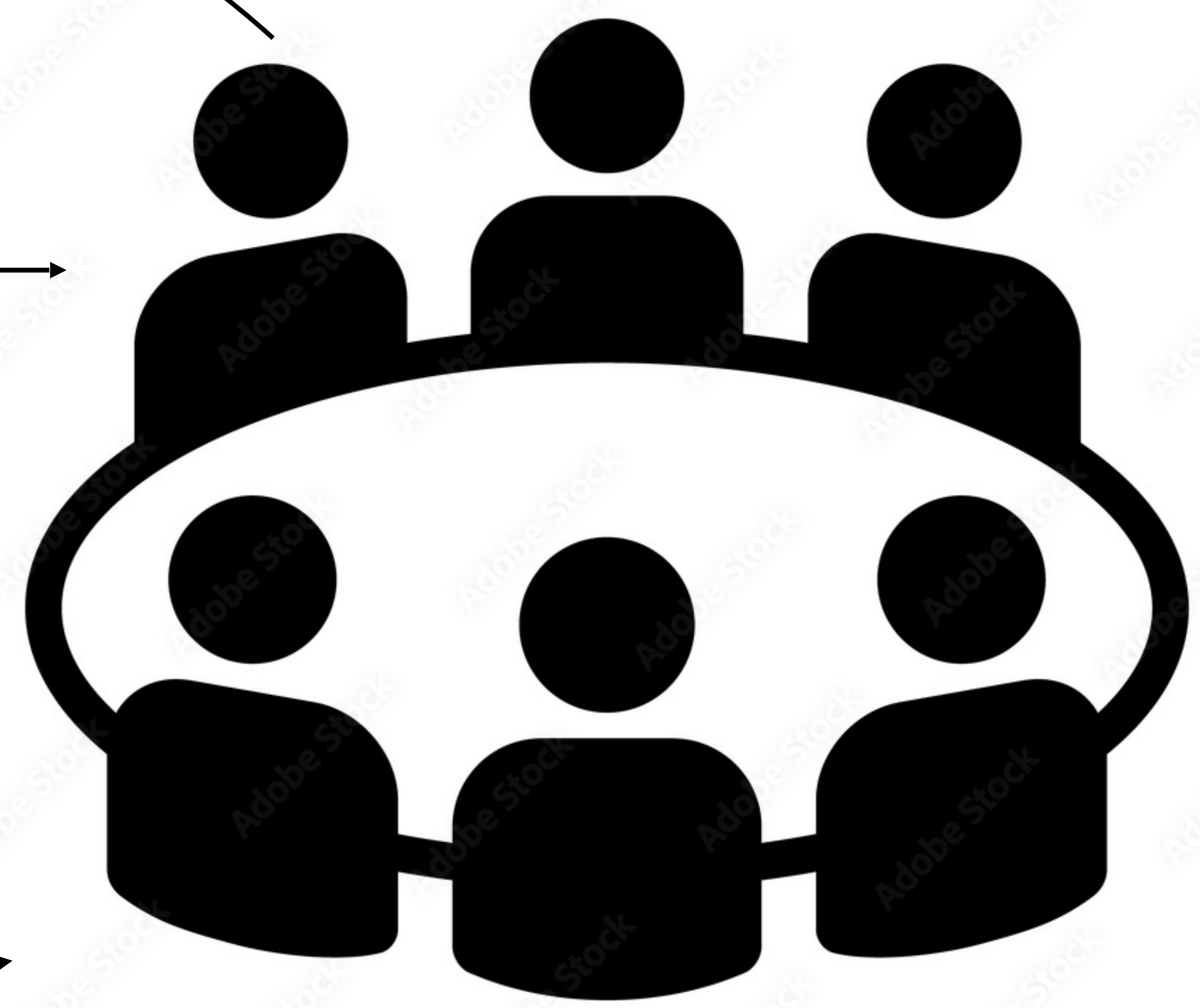
Development, Exchange with Experts, Collaboration, Code review, CI testing

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

Clone:

https://github.com/<username>/acts

Develop & Make a PR



Make an Issue:

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

Ask on mattermost:

https://mattermost.web.cern.ch/acts/channels/town-square

Discuss at the open develops meeting

https://indico.cern.ch/category/7968/

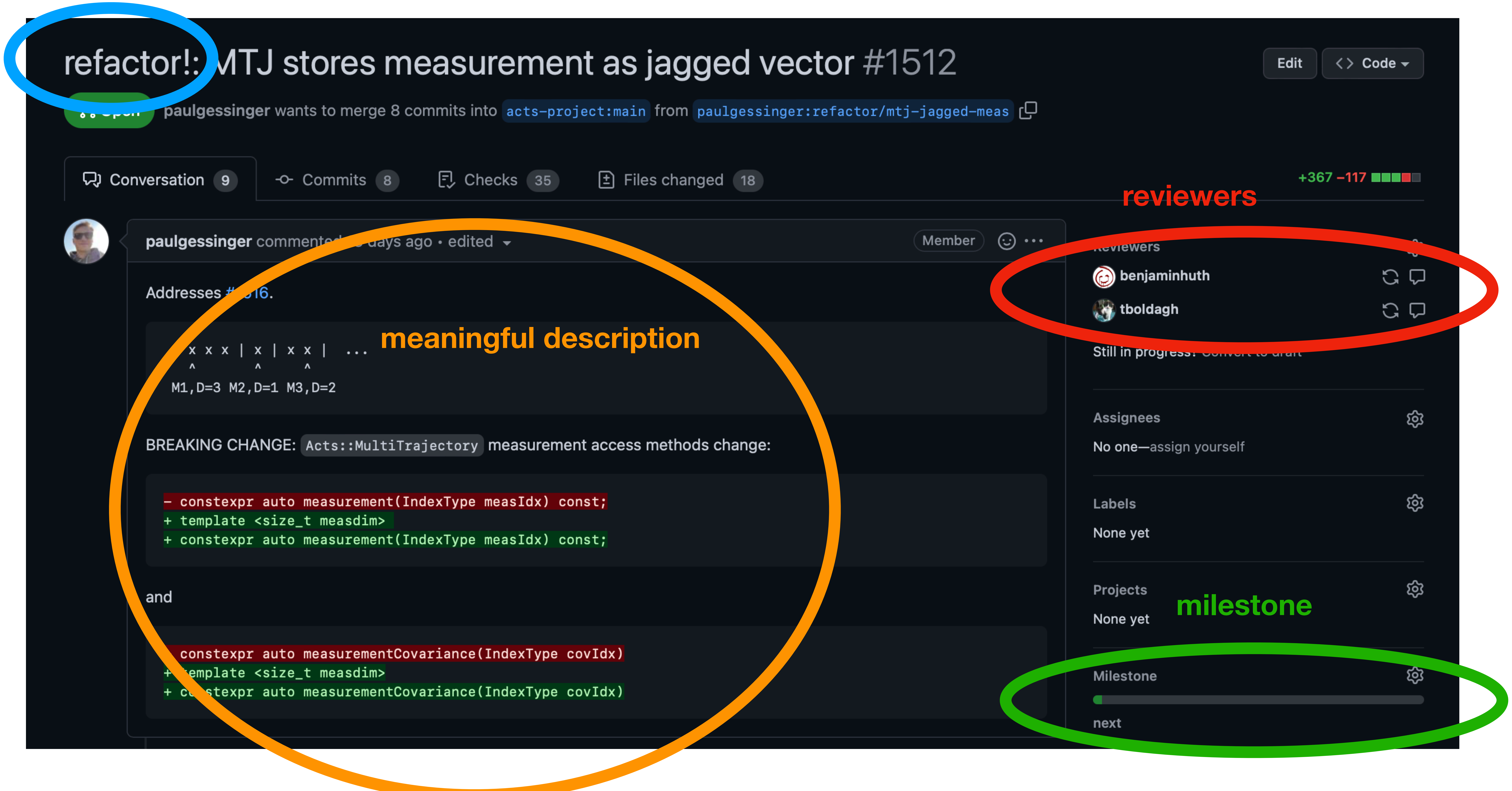
Tuesday 17:00, CE(S)T

new, periodic Asia-friendly slot 9:00 CE(S)T ~ 1/month

Core: contributing

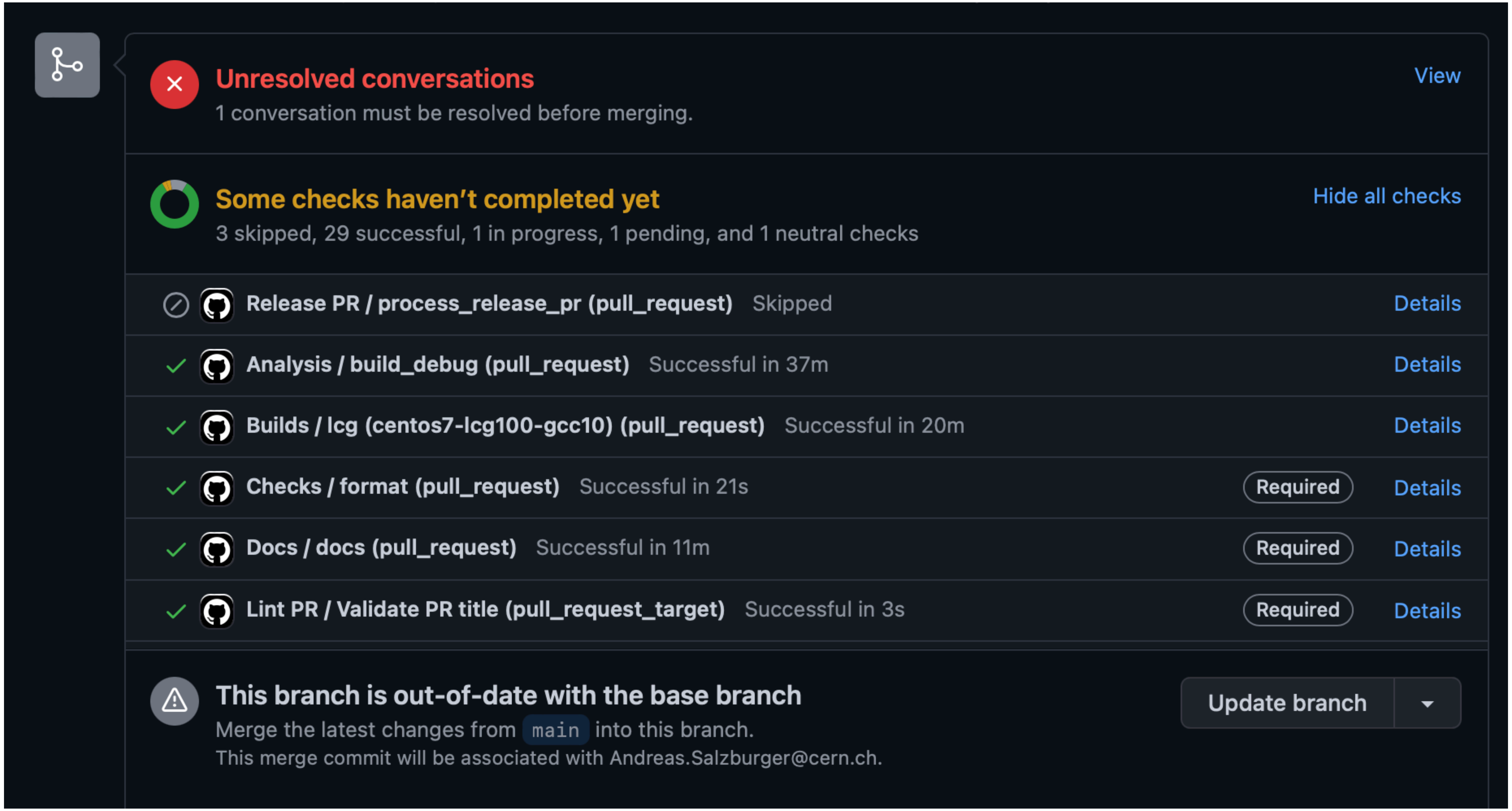
Pull requests come with a template that guides through a proper submission

semantic naming: feat, doc, refactor, fix

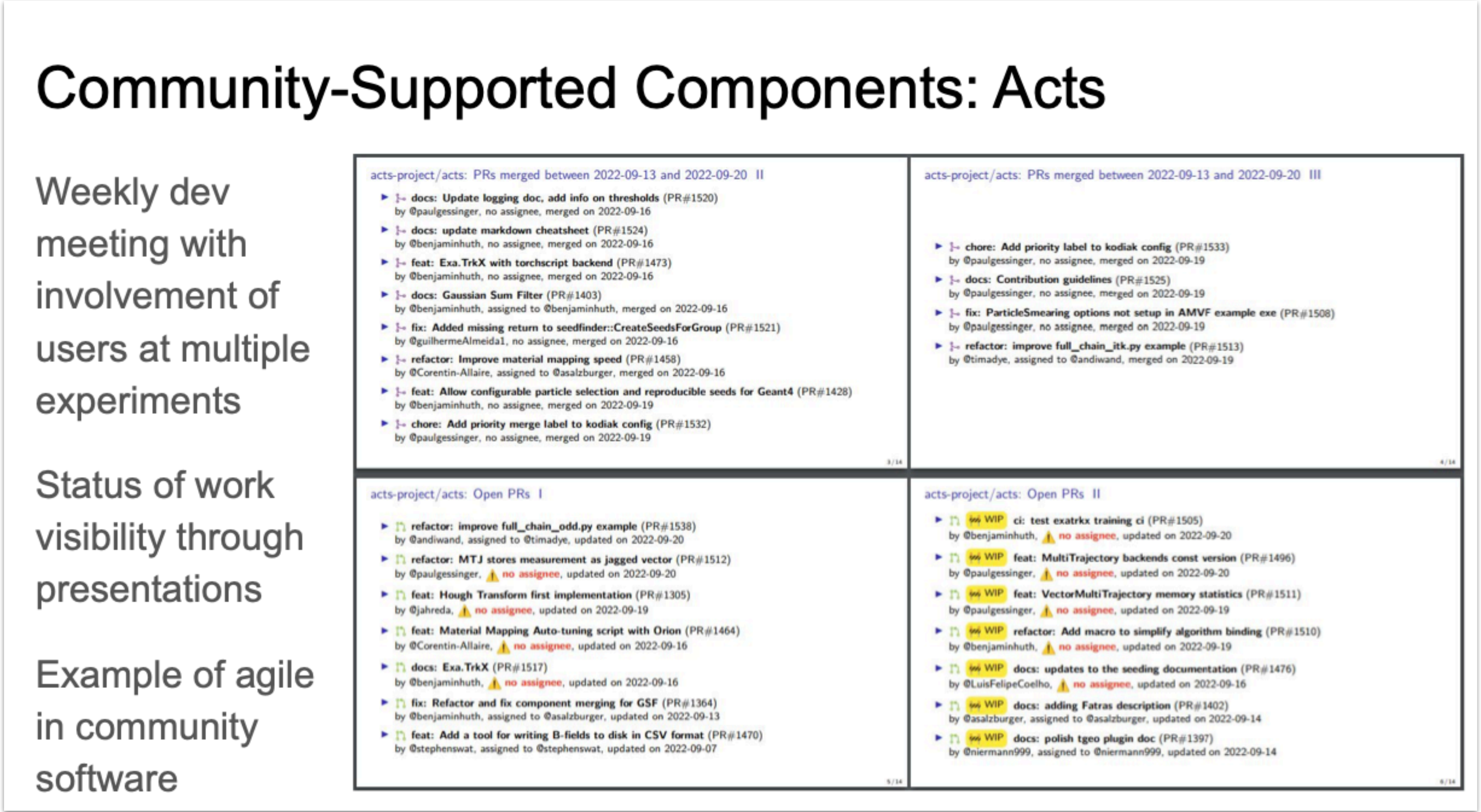


Core: contributing & testing

Pull requests run through a CI pipeline



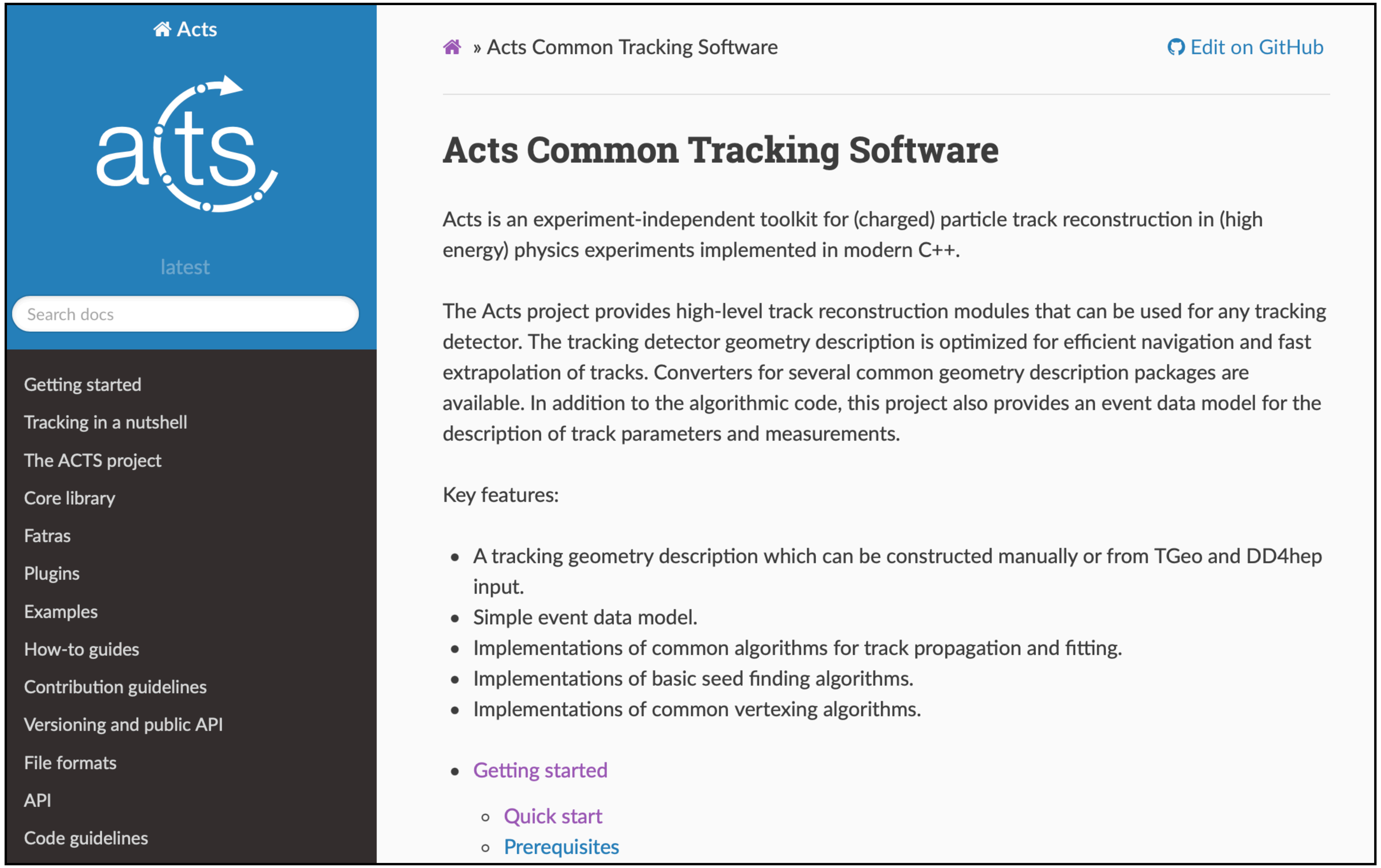
Core: collaboration & community



[Wouter Deconinck, CHEP2023, Plenary Contribution]

Core: documentation

Submitted code should have `doxygen` documentation and `readthedocs` resources


Acts

latest

Search docs

Getting started
Tracking in a nutshell
The ACTS project
Core library
Fatras
Plugins
Examples
How-to guides
Contribution guidelines
Versioning and public API
File formats
API
Code guidelines

» Acts Common Tracking Software

Edit on GitHub

Acts Common Tracking Software

Acts is an experiment-independent toolkit for (charged) particle track reconstruction in (high energy) physics experiments implemented in modern C++.

The Acts project provides high-level track reconstruction modules that can be used for any tracking detector. The tracking detector geometry description is optimized for efficient navigation and fast extrapolation of tracks. Converters for several common geometry description packages are available. In addition to the algorithmic code, this project also provides an event data model for the description of track parameters and measurements.

Key features:

- A tracking geometry description which can be constructed manually or from TGeo and DD4hep input.
- Simple event data model.
- Implementations of common algorithms for track propagation and fitting.
- Implementations of basic seed finding algorithms.
- Implementations of common vertexing algorithms.

- Getting started
  - Quick start
  - Prerequisites

https://acts.readthedocs.io/en/latest/

Core: some selected clients

ATLAS EXPERIMENT

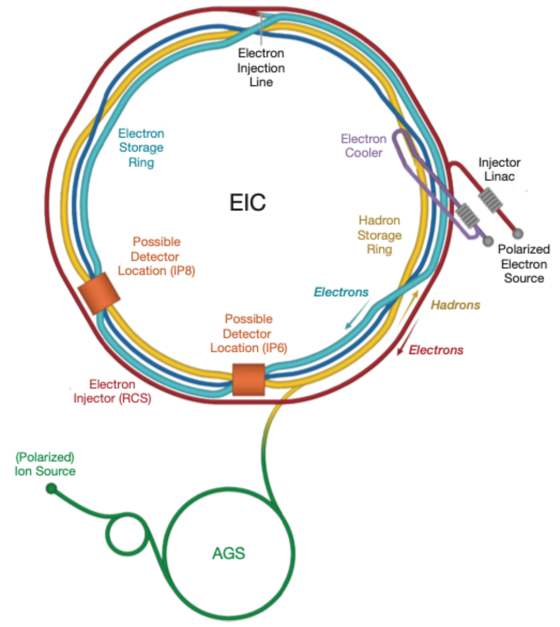
ACTS Vertex reconstruction in Run-3
full ACTS powered reconstruction for Phase-2

sPHENIX

full ACTS powered track reconstruction
[CSBS Article]

FASER

full ACTS powered track reconstruction
[FASER Preprint with ACTS mention]



Electron-Ion Collider (EIC) software stack, common track reconstruction software based on ACTS
[CHEP2023 keynote talk]

CEPC

Test implementation for CEPC design study
[CEPC Concept paper]

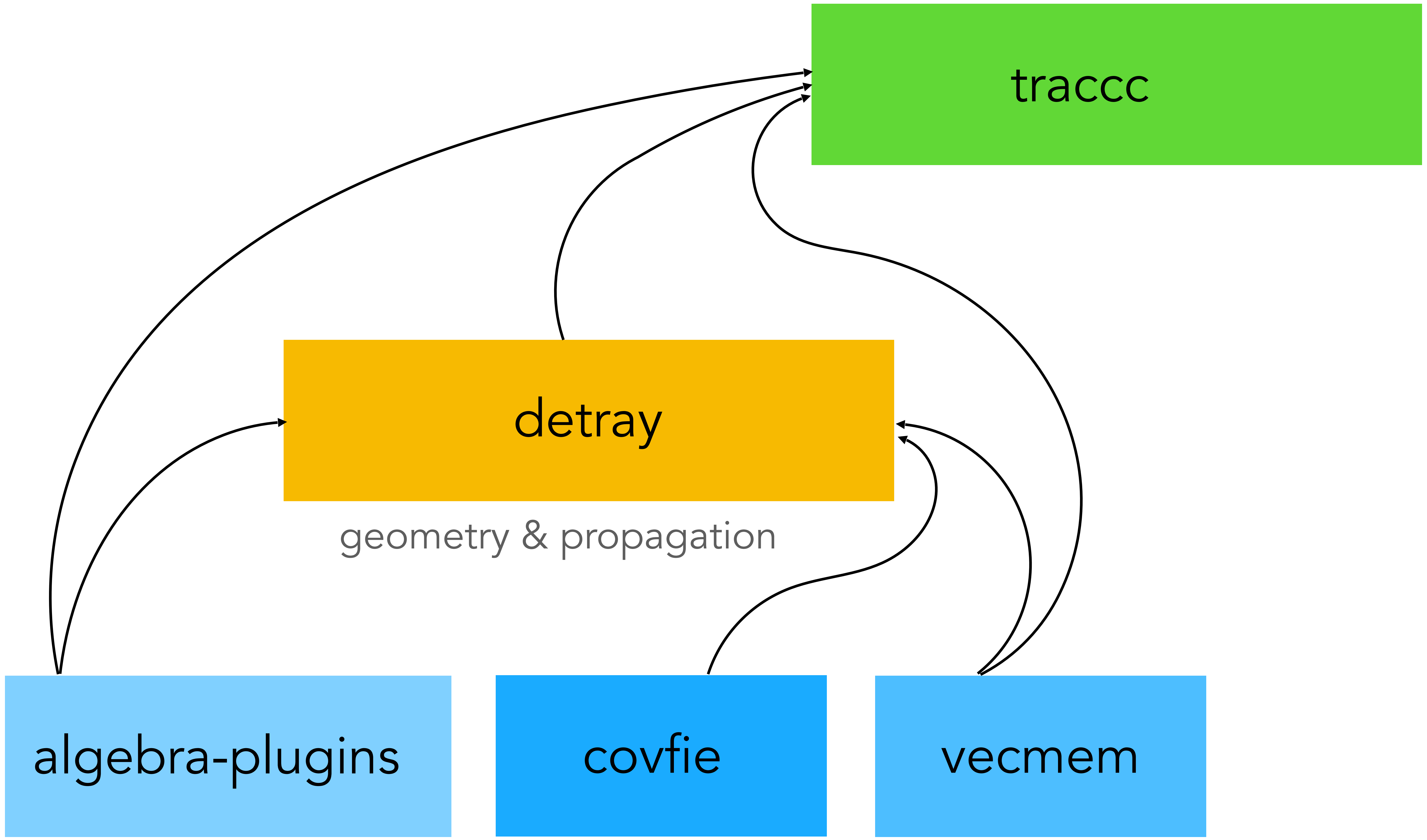
R&D1

acts-parallelization@cern.ch

CPU/GPU “single source” demonstrator
re-implementing the main Core chain

R&D1: the traccc project

full scale demonstrator of an ATLAS-like track reconstruction chain for CPU/GPU



encapsulation of algebra operation backend

generic vector field library

stl-style containers with managed memory

EP-R&D
Programme on Technologies for Future Experiments

Goal is to establish a track reconstruction chain without algorithmic compromises. CHEP2023 summary talk can be found [here].

R&D1: vecmem & algebra-plugins

vecmem: memory management

- use of `std::pmr::memory_resource` to customize the allocation scheme in the host side
- Supports CPU, CUDA, SYCL, and HIP
- Provides STL-like containers for host side for convenience of HEP developers

  `vecmem::vector`, `vecmem::jagged_vector (vector of vector)`, `vecmem::array`

algebra-plugins: encapsulation layer for algebra operations

- targeted at track reconstruction entirely
- dimensions up to 8 (needed for parameter propagation)
- supports device execution where possible and vecmem based backend
- can be used for algebra library benchmarking in realistic applications (instead of mockup benchmarks)

Backend	CPU	CUDA	SYCL
cmath	✅	✅	✅
Eigen	✅	✅	🟡
SMatrix	✅	⚪	⚪
VC	✅	⚪	⚪

✅: natively supported
🟡: natively supported, but not tested
⚪: no support

vecmem is a generic library which can be of use to a broader community use, **algebra-plugins** is not designed for production.

R&D1: track reconstruction: covfie

A generic vector field library based on composition design

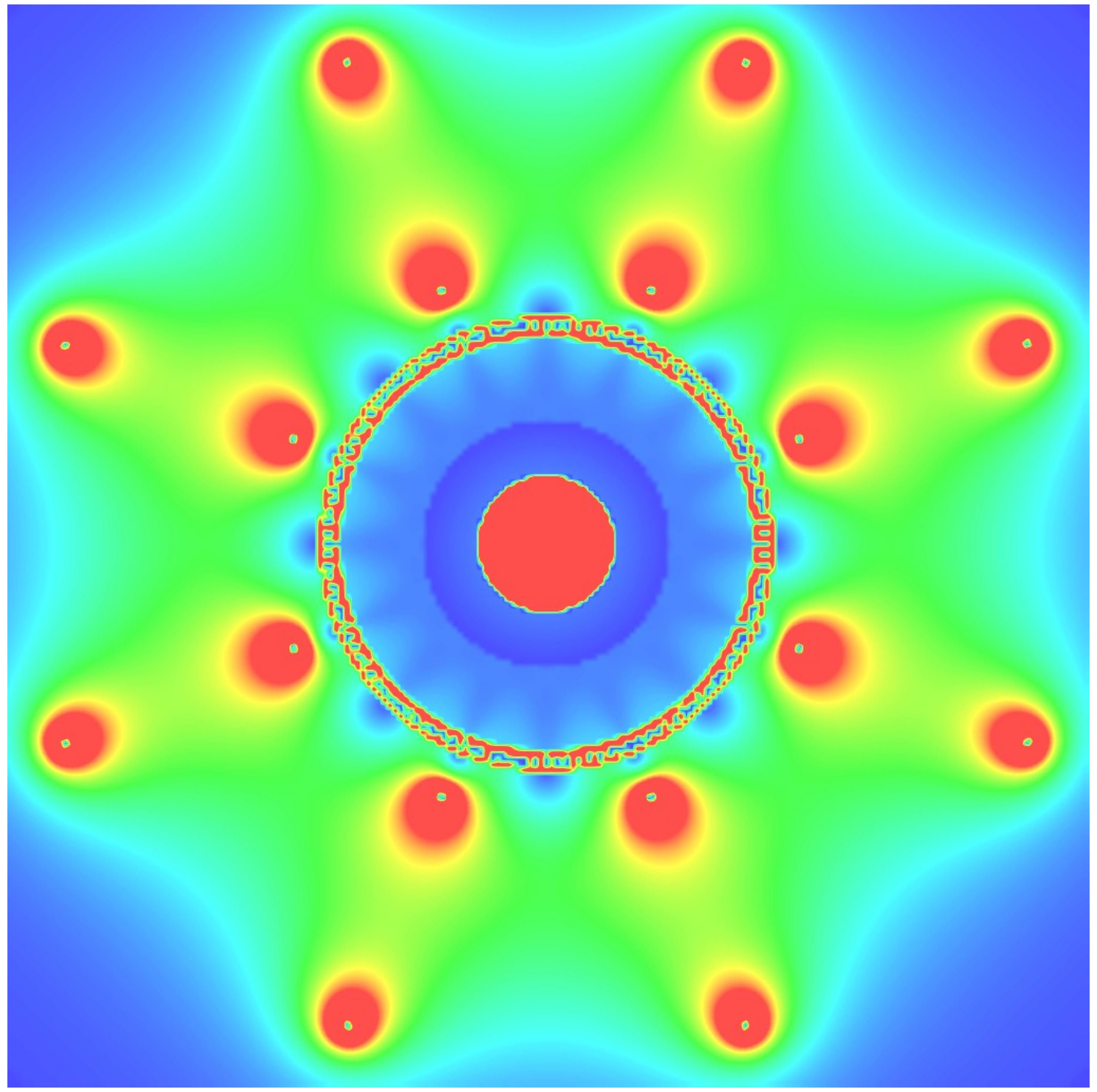
- format, coordinate transform and storage at compile time

```
using field_t =
    covfie::field<covfie::backend::transformer::interpolator::linear<
        covfie::backend::layout::strided<
            covfie::vector::ulong2,
            covfie::backend::storage::array<covfie::vector::float2>>>>;
```

possible field on CPU

```
using cuda_field_t = covfie::field<covfie::backend::transformer::affine<
    covfie::backend::transformer::interpolator::linear<
        covfie::backend::layout::strided<
            covfie::backend::vector::input::ulong3,
            covfie::backend::storage::cuda_device_array<
                covfie::backend::vector::output::float3>>>>>;
```

possible field on GPU



ATLAS magnetic field slice at z=0, entirely rendered on a GPU

	8192 X 8192 lookup time [ms]
CPU (Intel i5-7300U)	191719.2
GPU (GTX 1660 Ti)	90.4
GPU w/ texture memory	17.1

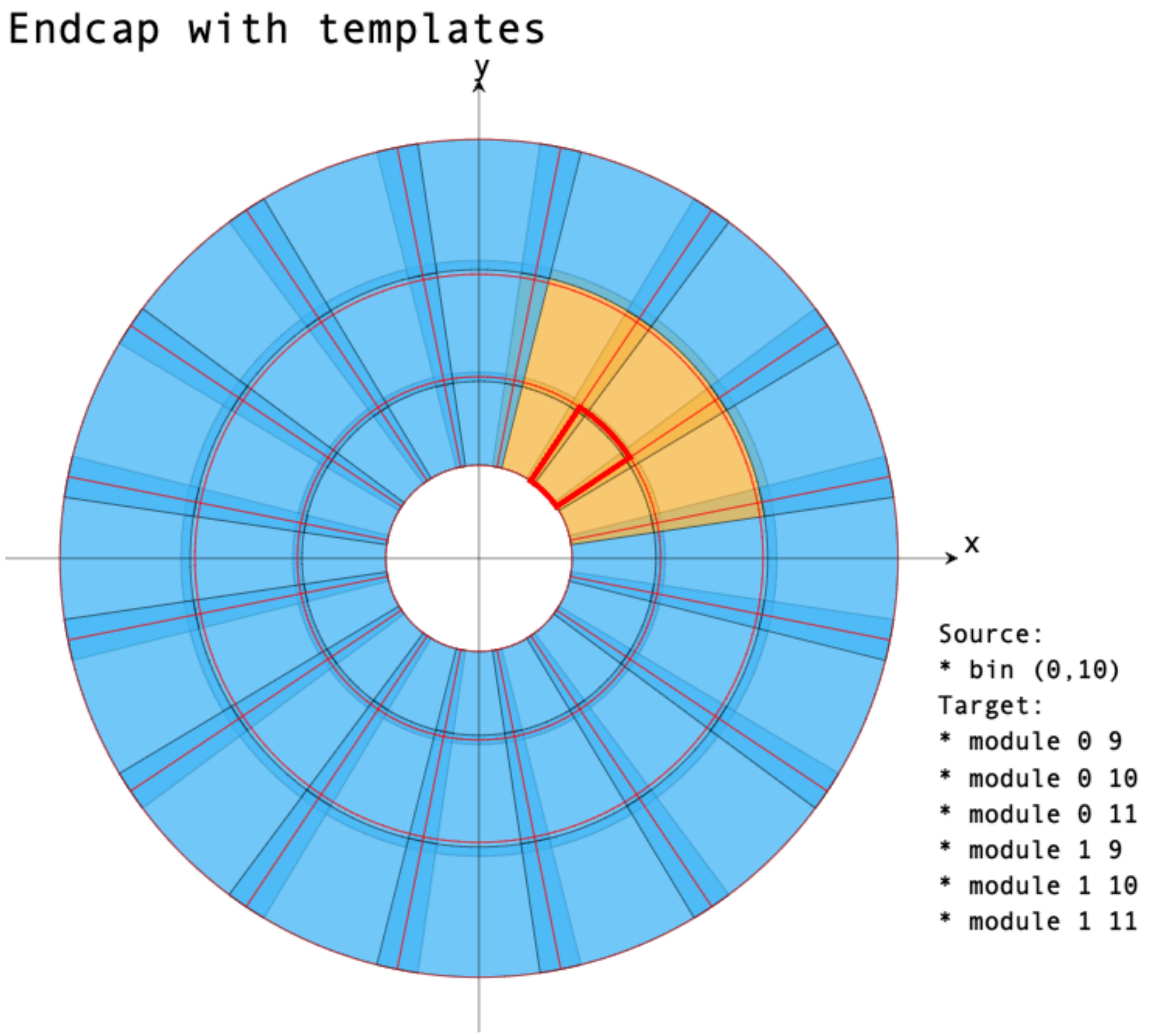
covfie is a generic library which can be of use to a broader community, e.g. full simulation on accelerators.

R&D1: detray (1)

Compile-time polymorphic geometry library

- bound surface type model and ACTS navigation
- polymorphism achieved by type unrolling
- device specialization through `vecmem`

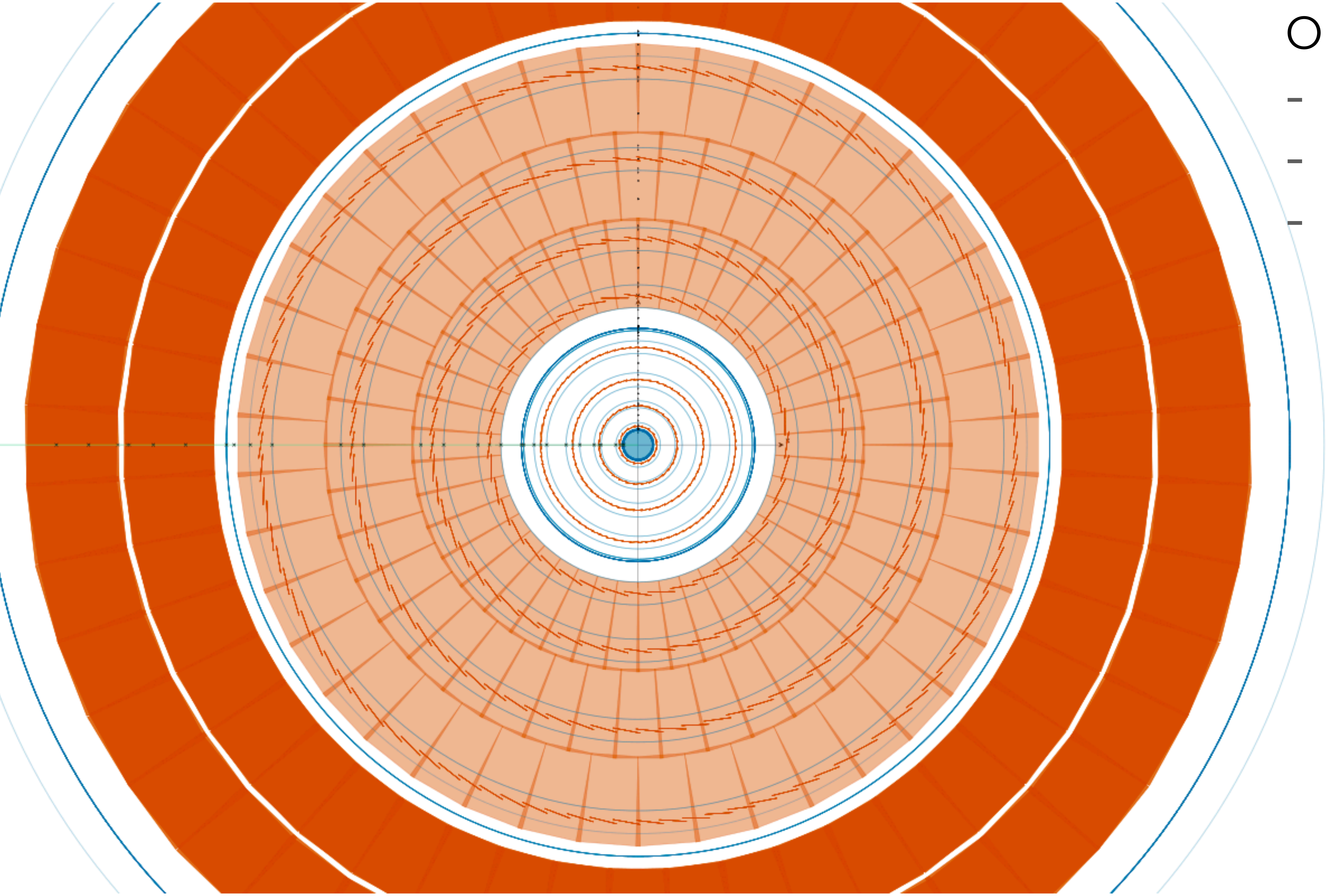
```
/** The detector definition.
 *
 * This class is a heavy templated detector definition class, that sets the
 * interface between geometry, navigator and grid.
 *
 * @tparam metadata helper that defines collection and link types centrally
 * @tparam array_type the type of the internal array, must have STL semantics
 * @tparam tuple_type the type of the internal tuple, must have STL semantics
 * @tparam vector_type the type of the internal array, must have STL semantics
 * @tparam source_link the surface source link
 */
template <typename metadata,
          template <typename, std::size_t> class array_t = darray,
          template <typename...> class tuple_t = dtuple,
          template <typename...> class vector_t = dvector,
          template <typename...> class jagged_vector_t = djagged_vector,
          typename source_link = dindex>
class detector {
```



Local navigation resolving by grid binning in detray

detray & VecGeom developers are already in contact and initial exchange, with plenty of room for more collaboration.

R&D1: detray (2)



Open Data Detector in detray

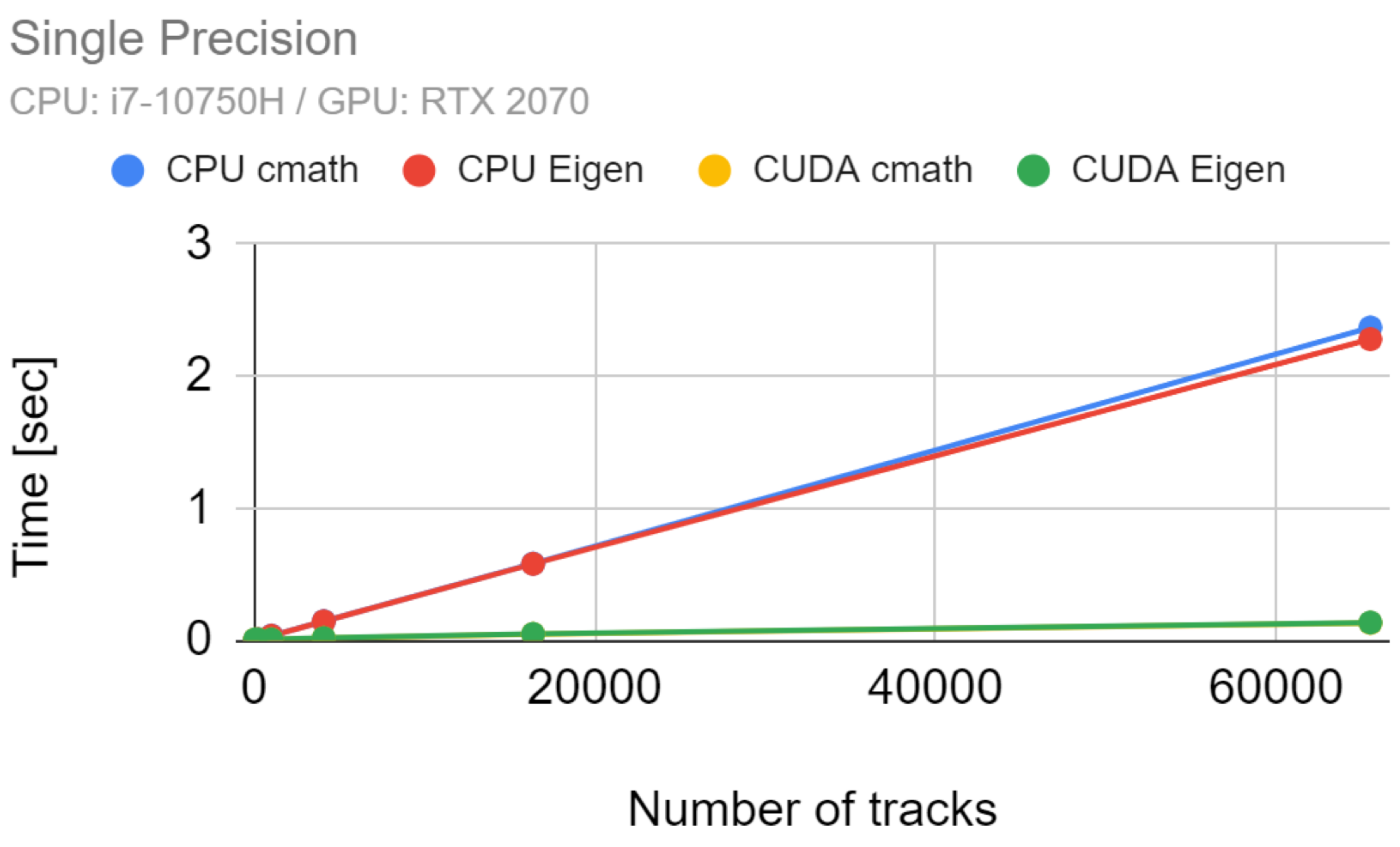
- Complex detector geometries
- Material description
- Propagation in inhomogeneous field

detray adaptive runge-kutta transport code is relatively self-contained and could serve more communities.

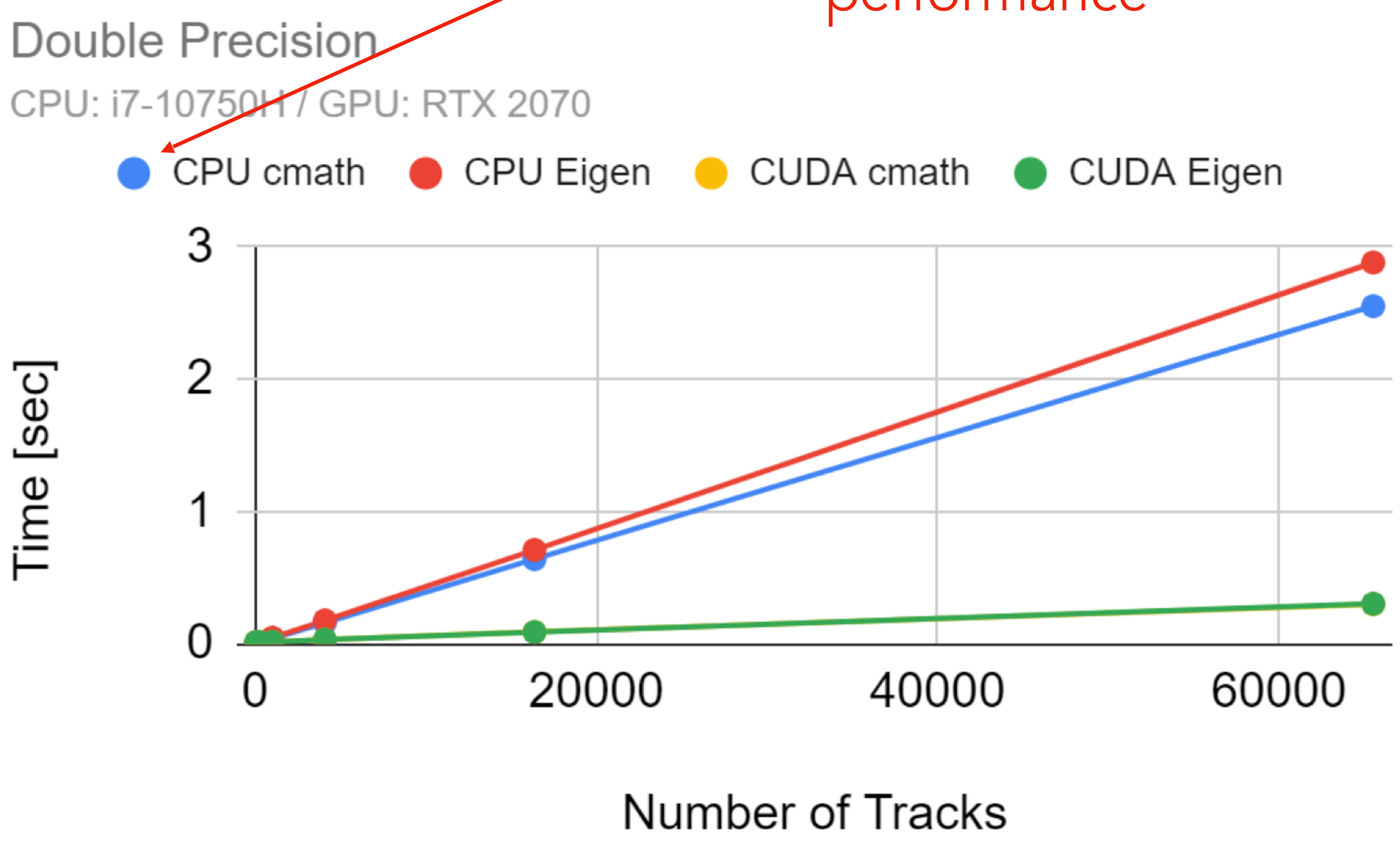
R&D1: detray (3)

Runge-Kutta propagation

- 4th order RKN propagator for parameters and covariances
- navigation and material effects integration using detray::geometry
- magnetic field access using covfie



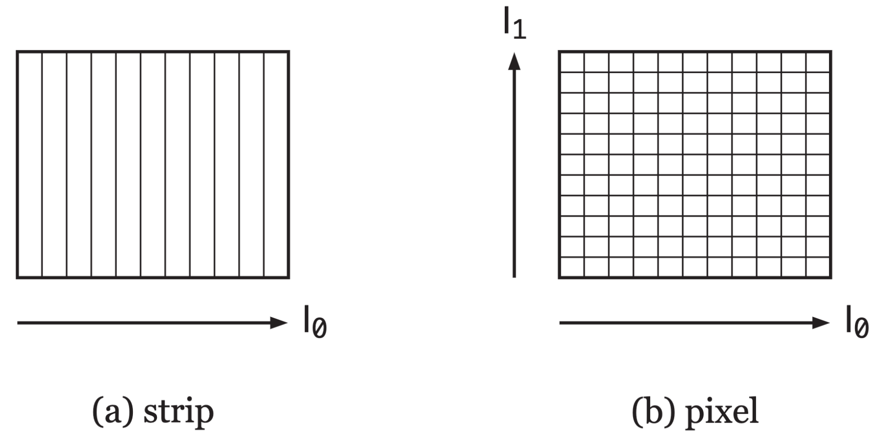
this is the **single core** performance



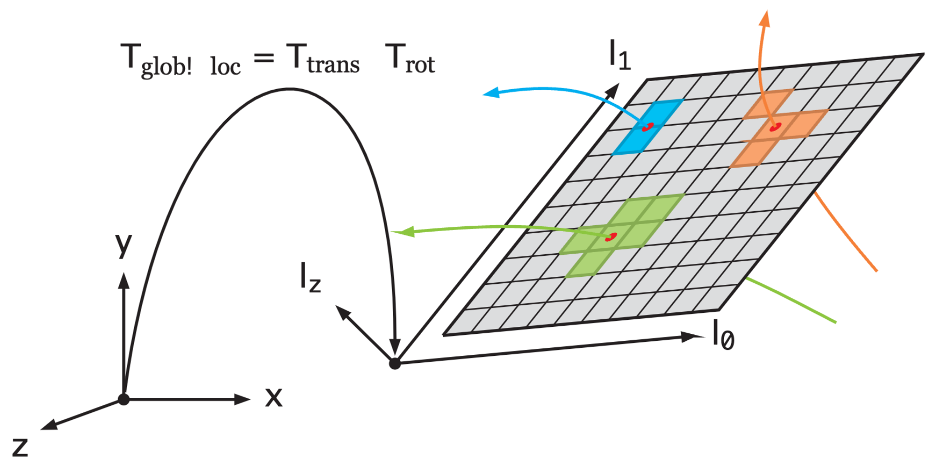
detray adaptive runge-kutta transport code is relatively self-contained and could serve more communities.

R&D1: traccc (1)

Full chain demonstrator for track reconstruction on CPU/GPU

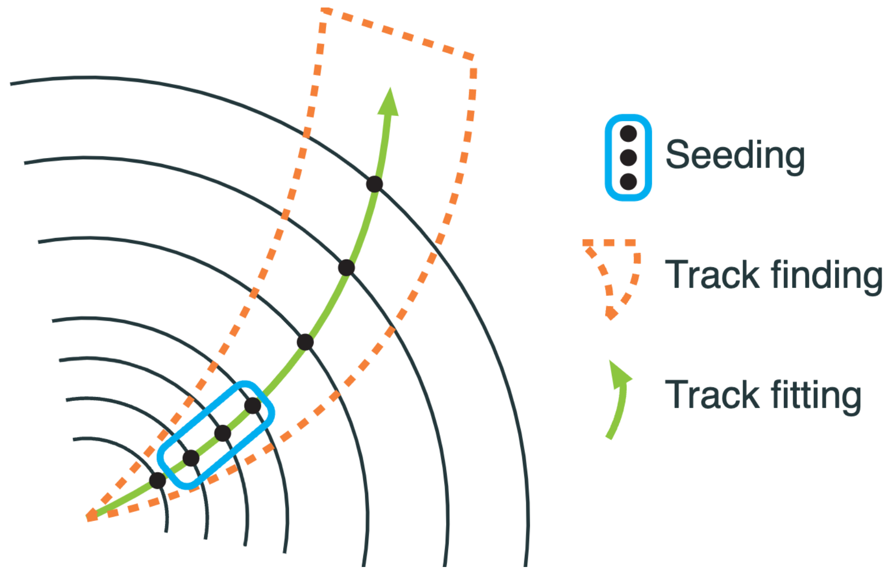


Clustering



Space Point formation

Consolidated code, testing also portability solutions CUDA/SYCL.

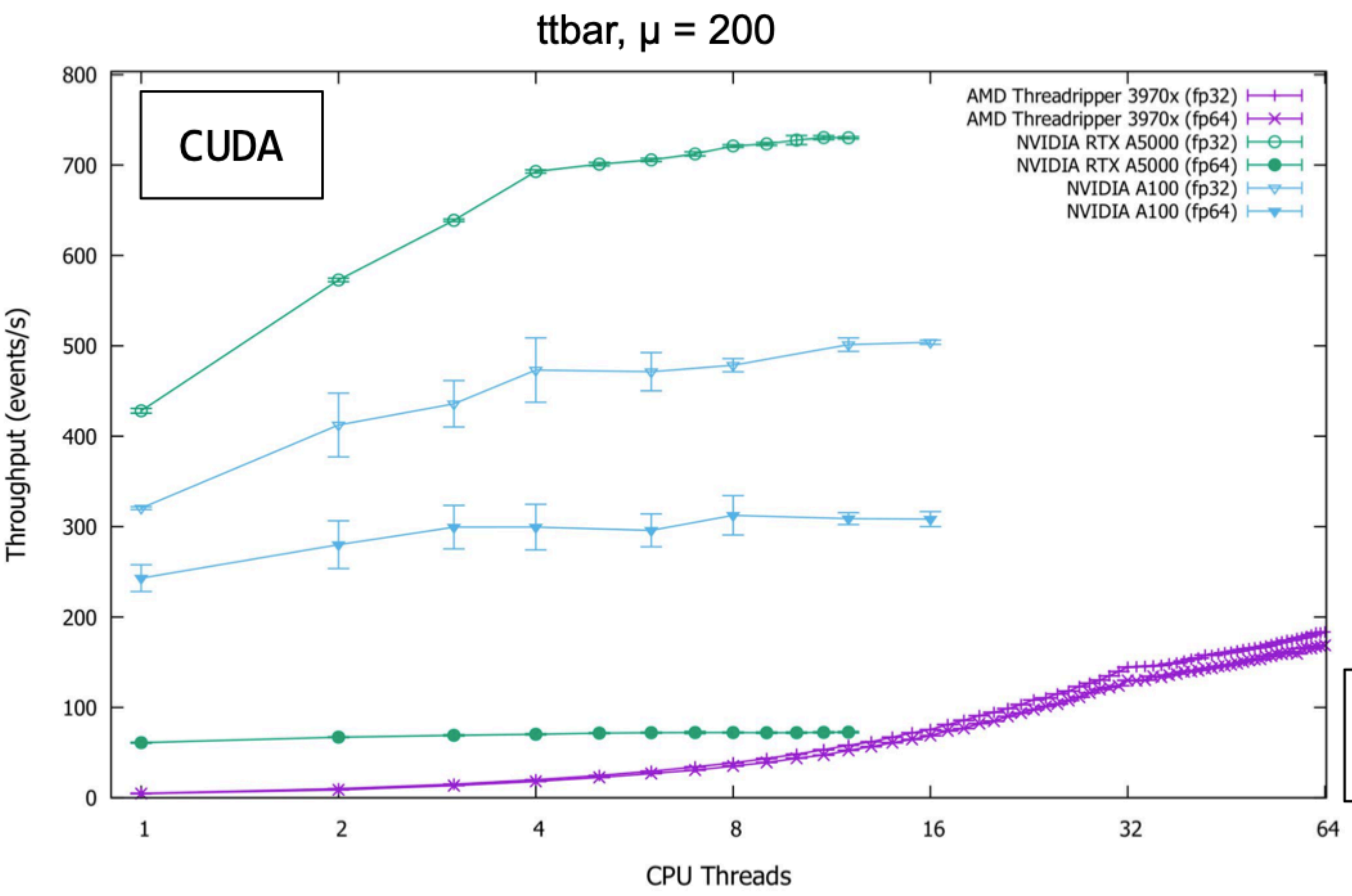


Track finding & fitting

traccc is an ideal playground to gather/share/exchange knowledge about code for heterogeneous systems.

R&D1n: traccc (2)

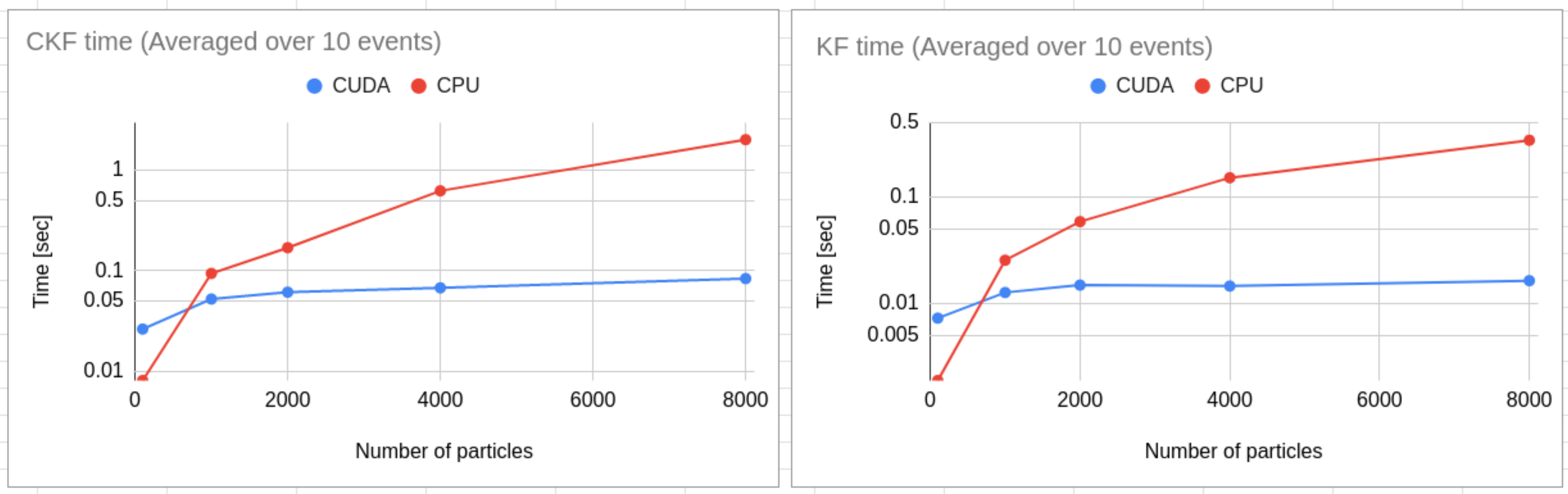
Full chain demonstrator for track reconstruction on CPU/GPU



Results partially generated on DAS-6.

R&D1n: traccc (3)

First results on CKF with detray - on toy detector (TrackML pixels)



R&D2

acts-machinelearning@cern.ch

Machine learning and ML assisted modules for track reconstruction

R&D2: machine learning application and assistance

Diverse ML (assisted) applications

- ML module research:

  ML Ambiguity Solver [C. Allaire, CHEP2023, Parallel Tallk]

  ML Navigator

- integration of ML partial or end-to-end pipelines

  Exa.TrkX + ACTS

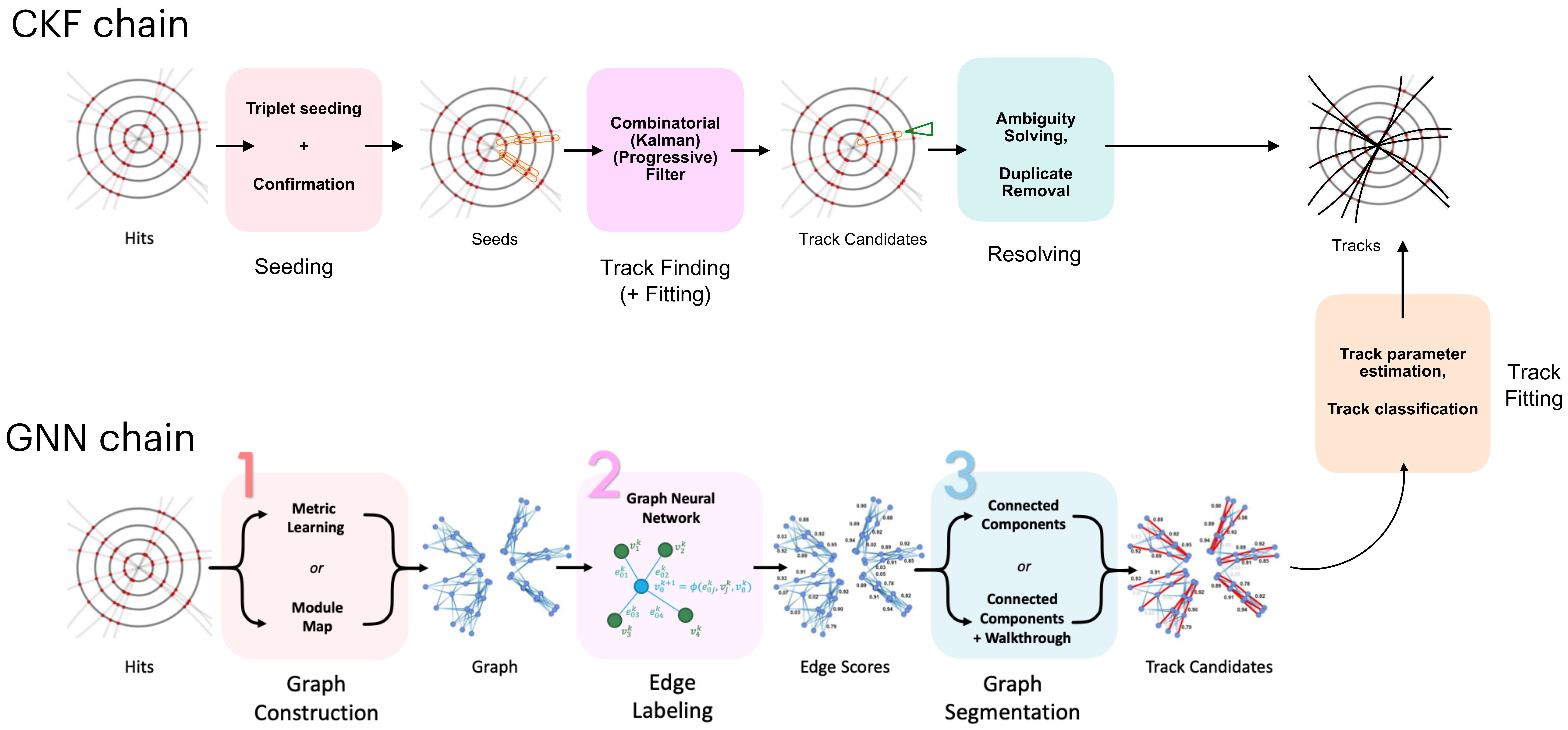
  Hashing + ACTS

- ML technology enhanced

  Parameter Tuning

  Auto-diff covariance transport

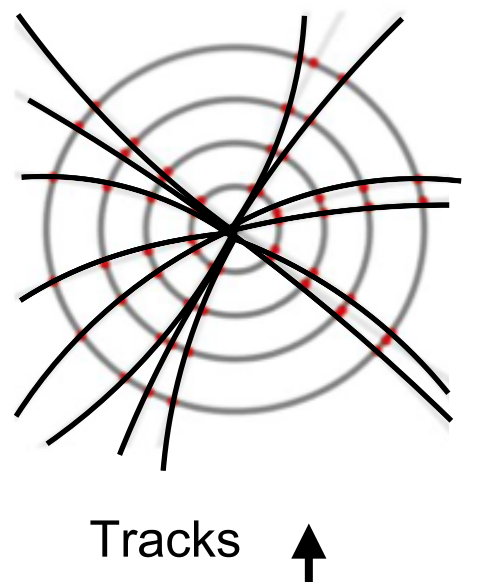
R&D2: GNN for track finding

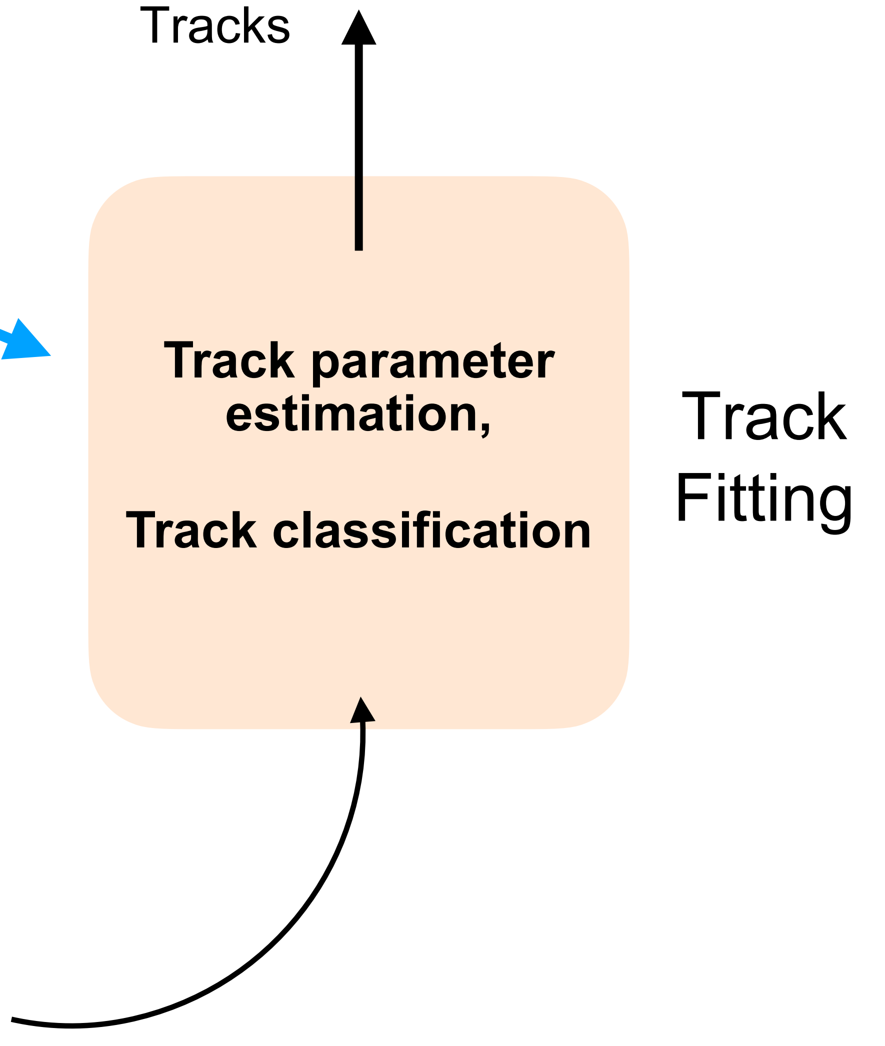


R&D2: GNN for track finding

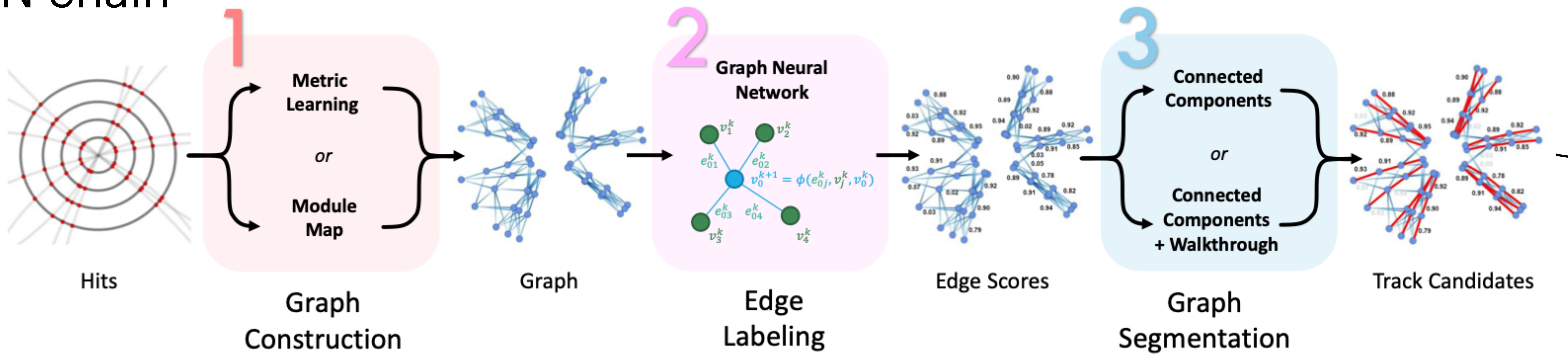
Track finding with using Graph Neural Networks has become popular in the last few years.

- strongly driven by TrackML challenge, Exa.TrkX collaboration, etc.
- ACTS used a toolkit to develop the principle pipelines
- ACTS offers a neat integration with track fitting on CPU/GPU

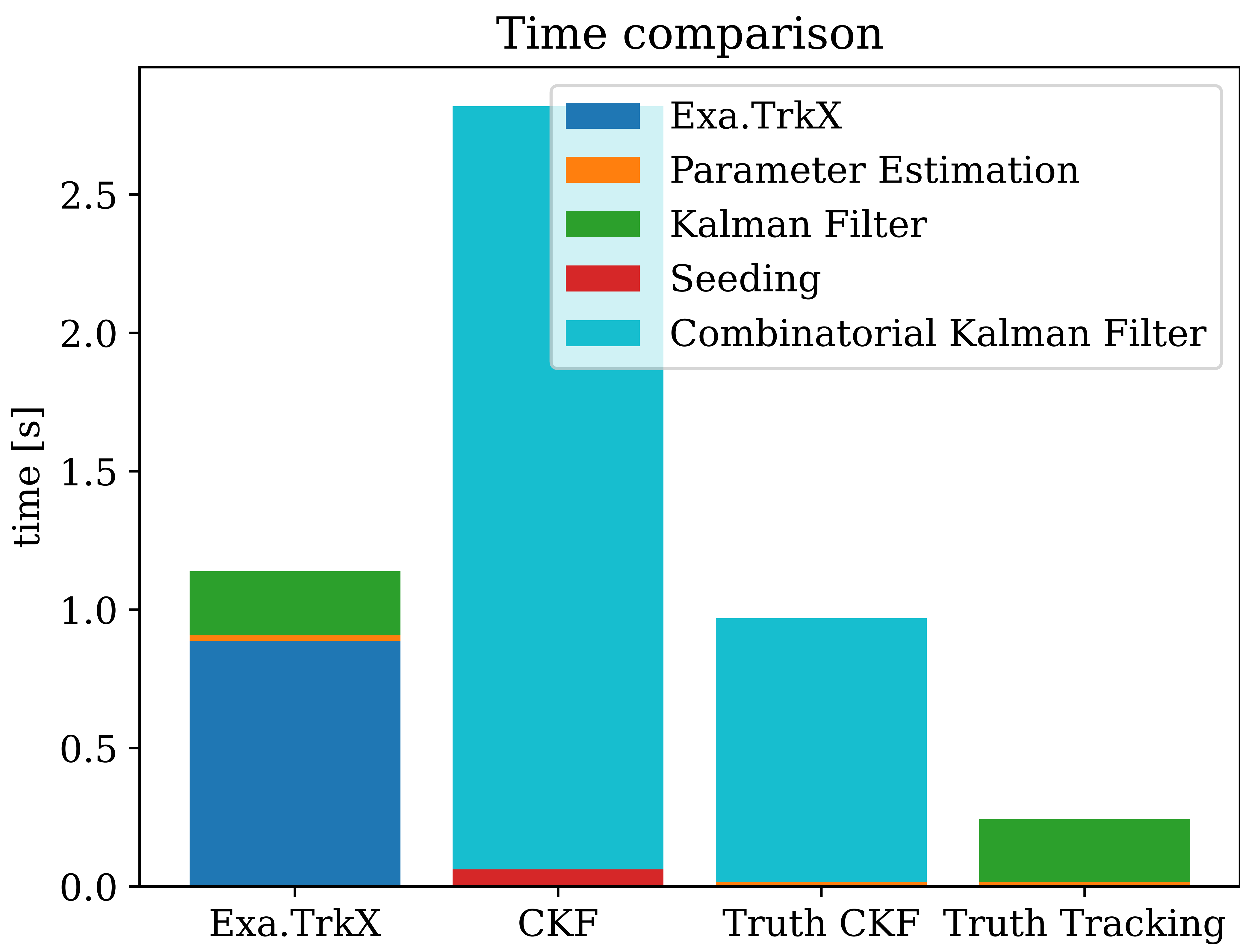


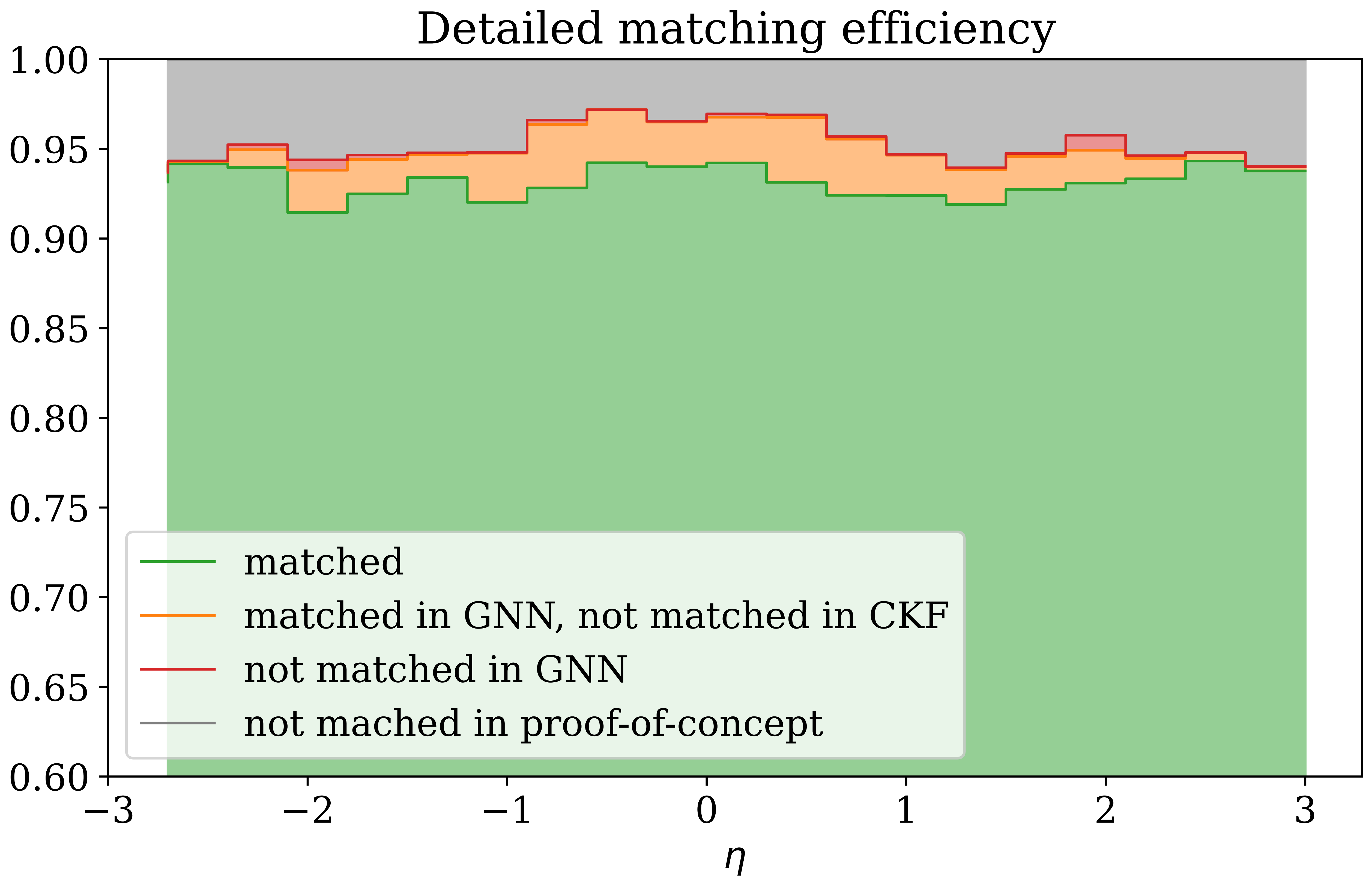


GNN chain



R&D2: GNN for track finding





High track reconstruction efficiency achieved in GNN + CKF setup, with very promising computing performance.

Study on Open Data Detector by B. Huth

Development and R&D, add-ons:

Core

acts-developers@cern.ch

CPU multi-threaded library of
tracking reconstruction components

Add-ons

OpenDataDetector
ActSVG

R&D1

acts-parallelization@cern.ch

CPU/GPU "single source" demonstrator
re-implementing the main Core chain

R&D2

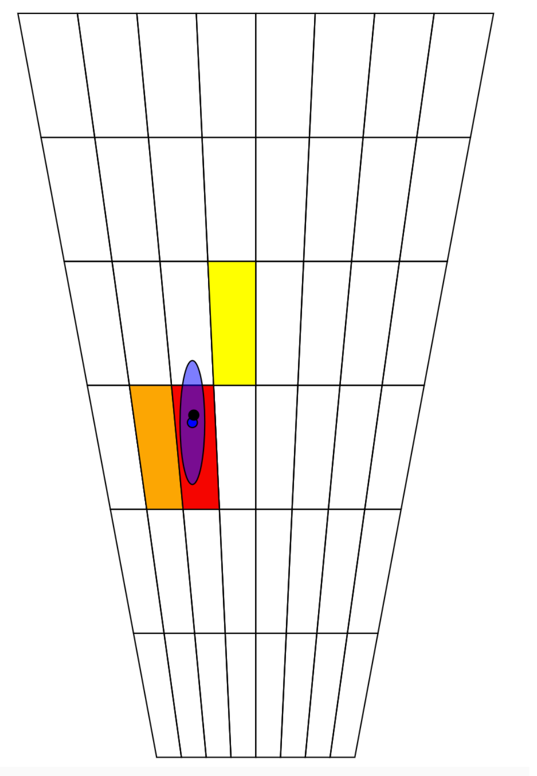
acts-machinelearning@cern.ch

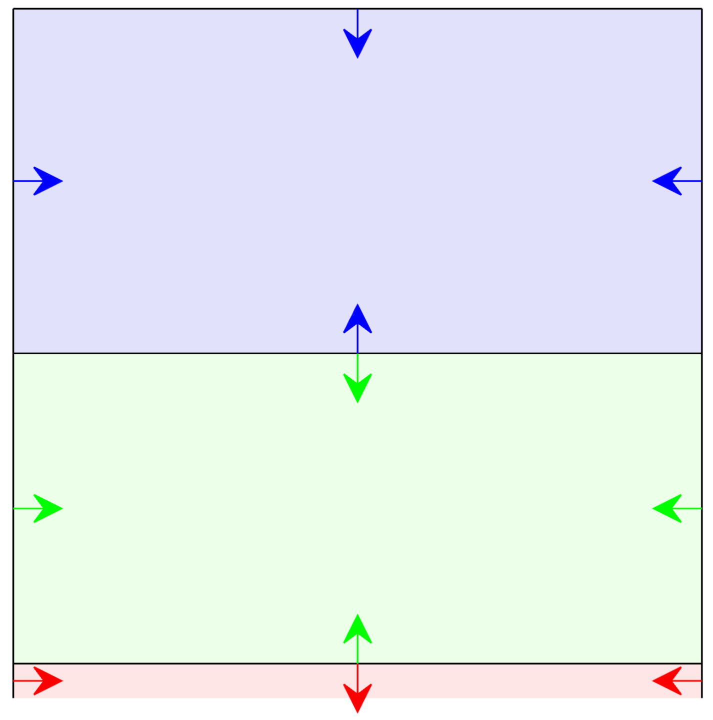
Machine learning and ML assisted
modules for track reconstruction

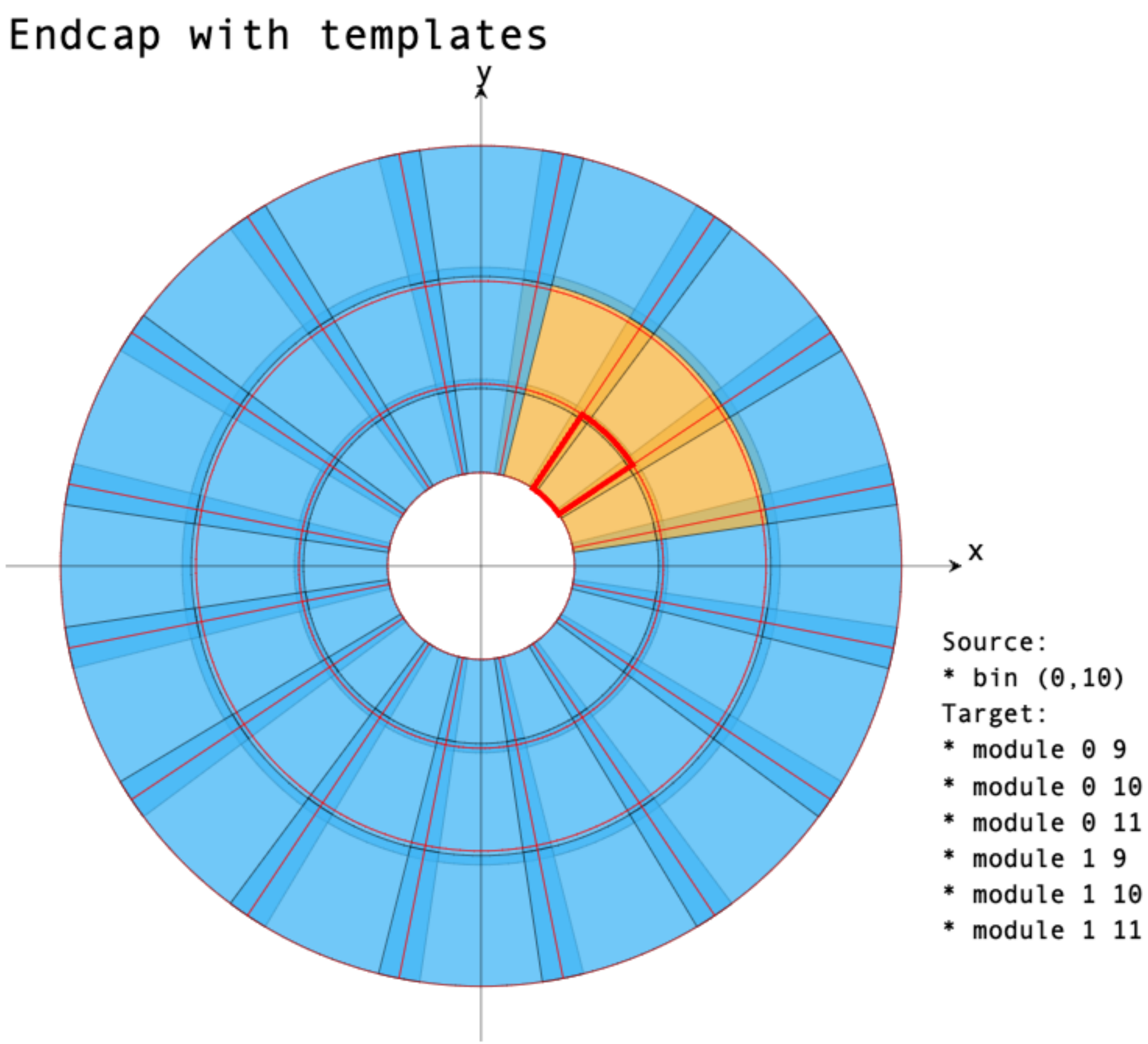
Plotting: actsvg

2D plotting library dedicated for tracking

- No dependencies, C++ header only, no ACTS dependency
  - ACTS and detray translate into `actsvg::meta` objects
- Plot geometry & geometric relations (on mouse over effects for debugging)
- Plot clusters & cluster information





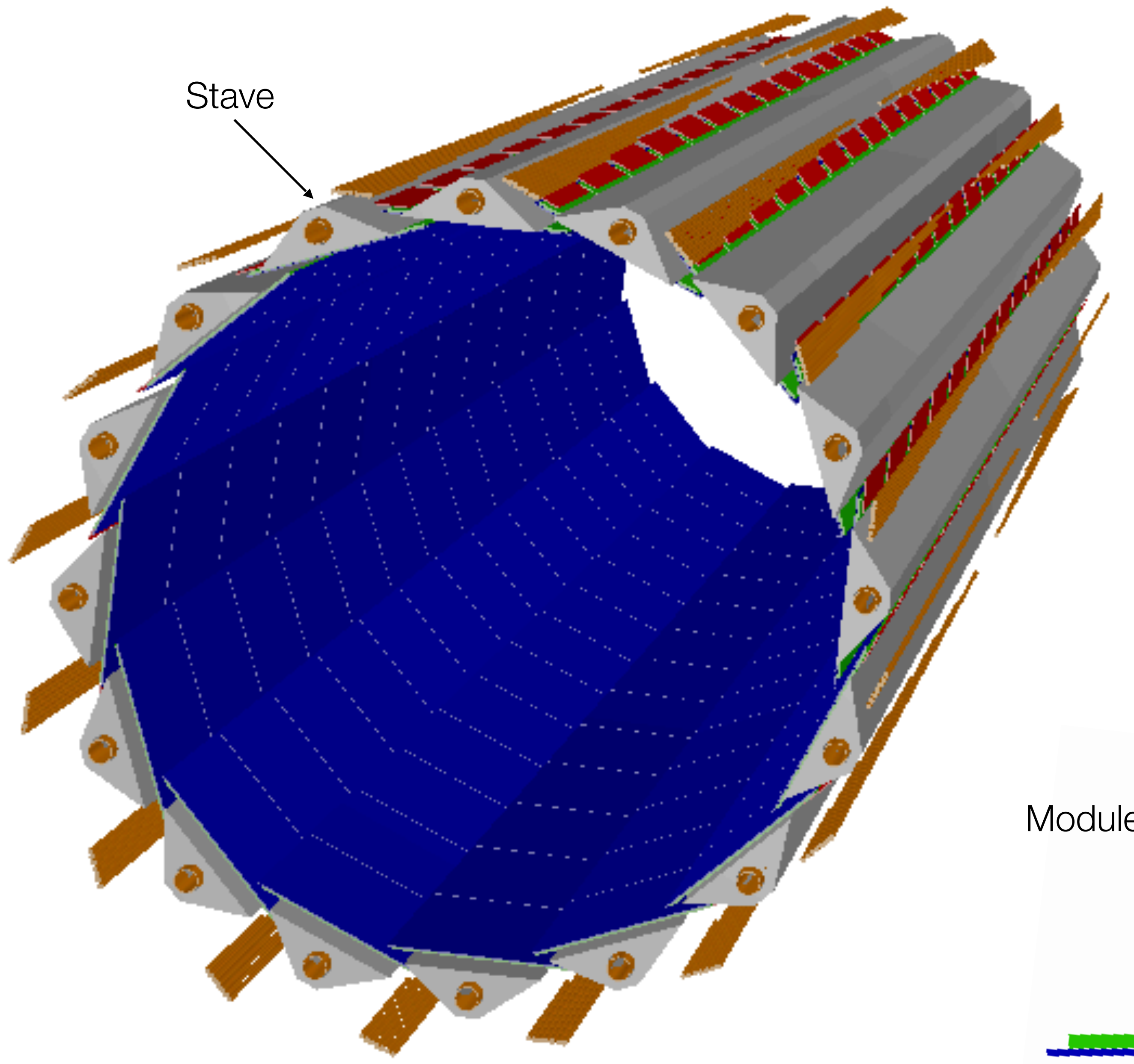


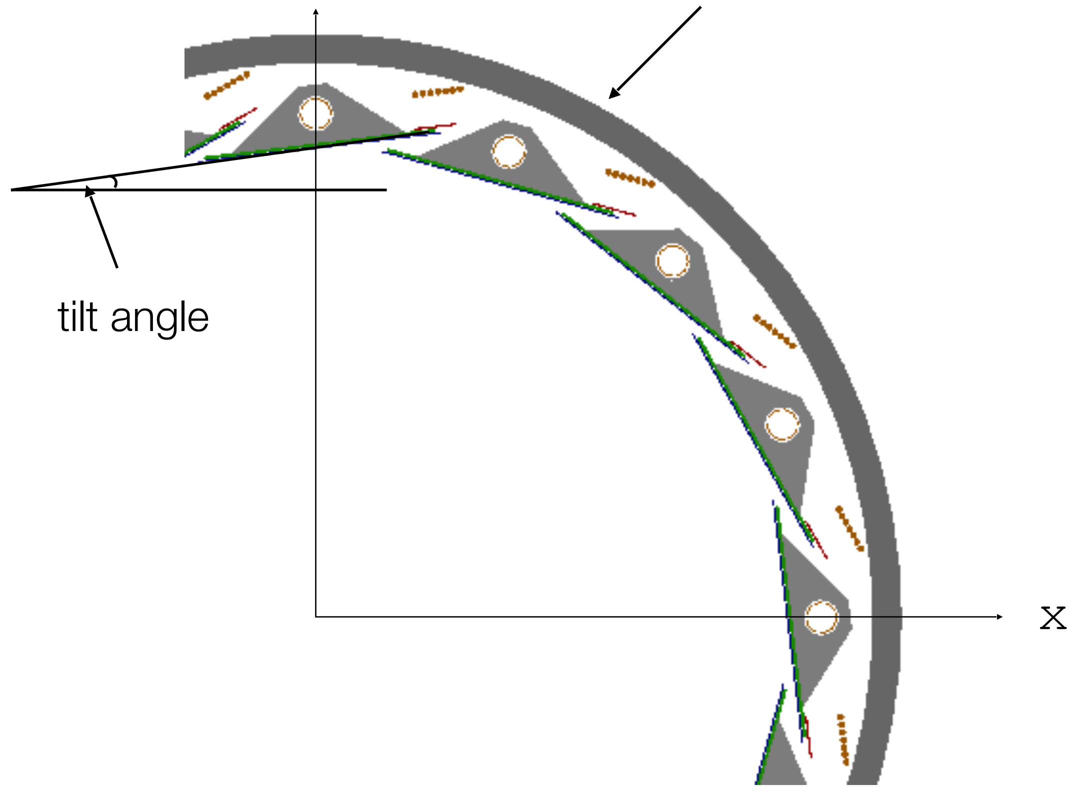
actsvg is a generic plotting library.

Community: Open Data Detector & key4hep

Evolution of TrackML detector

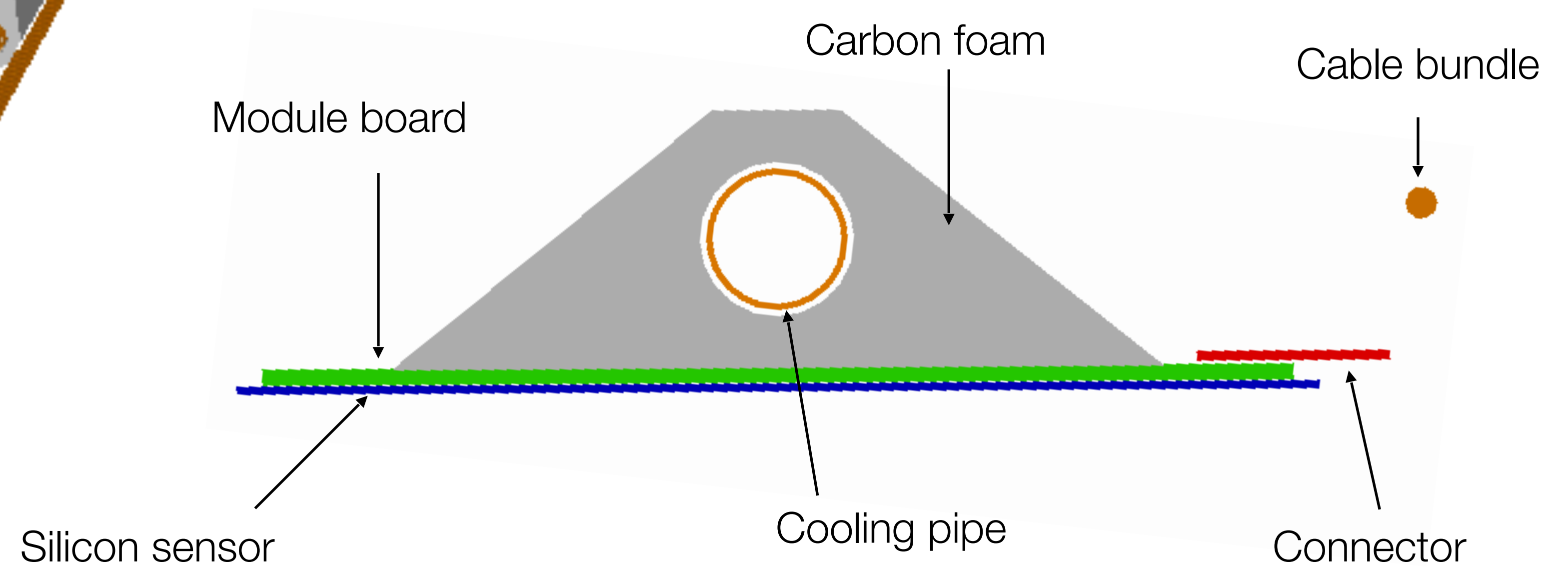
- Re-implemented in DD4Hep to enable full/fast simulation
- Quasi-realistic feedback to allow real-life scenario testing of algorithms
- Supports TrackML output format through ACTS binding (work ongoing to also support edm4hep)





ACTS integration into key4hep SW stack

- Codename: acts4hep
- Summer student project to make a ACTS Gaudi based demonstrator



[AS, CHEP2023 Parallel talk]

Ongoing activity to include first Calorimeter description, MS to follow.

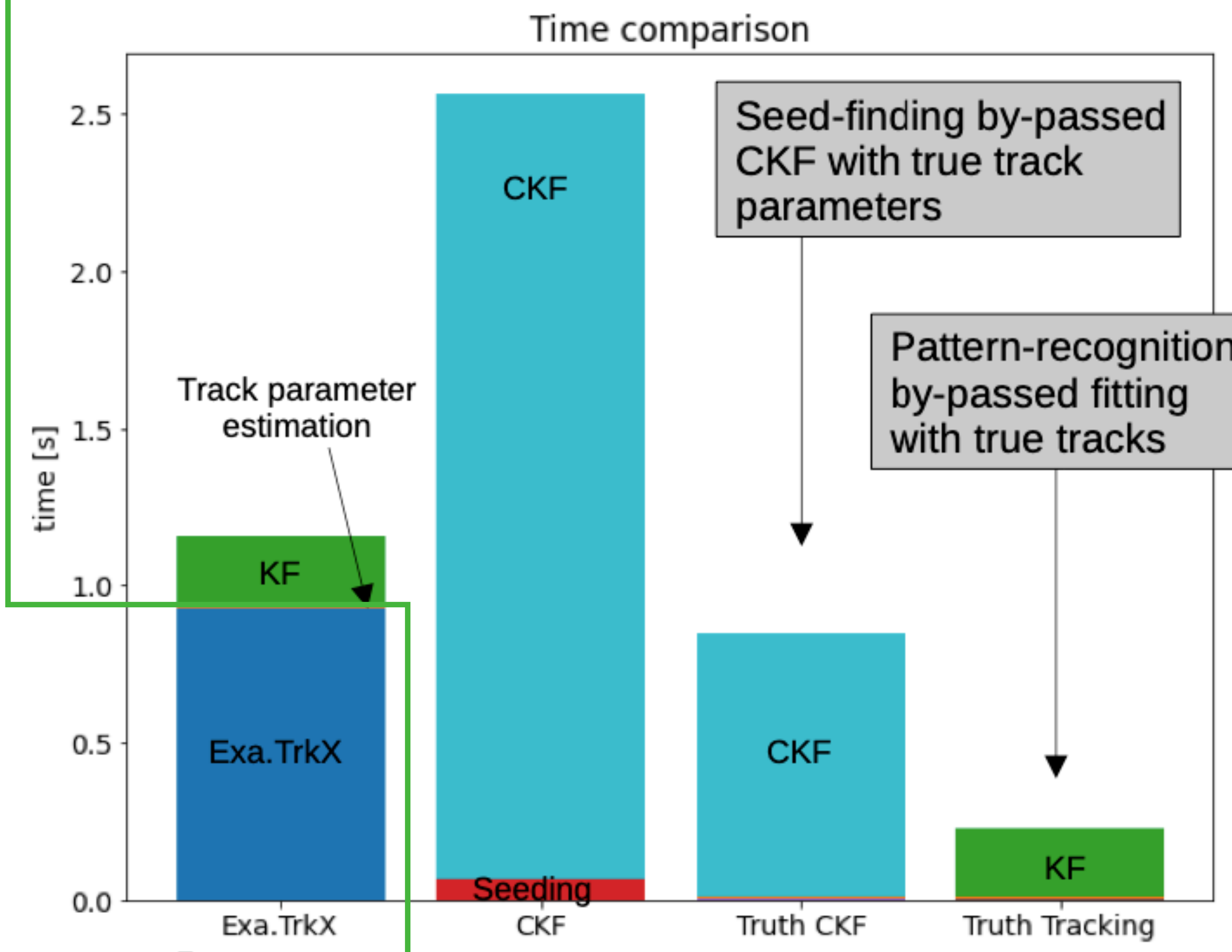
Concluding remarks

ACTS is a community driven software project

- idea is to develop a broadly carried, supported and sustainable component library

Join. Enhance. Profit. Give.

R&D1: technology BINGO



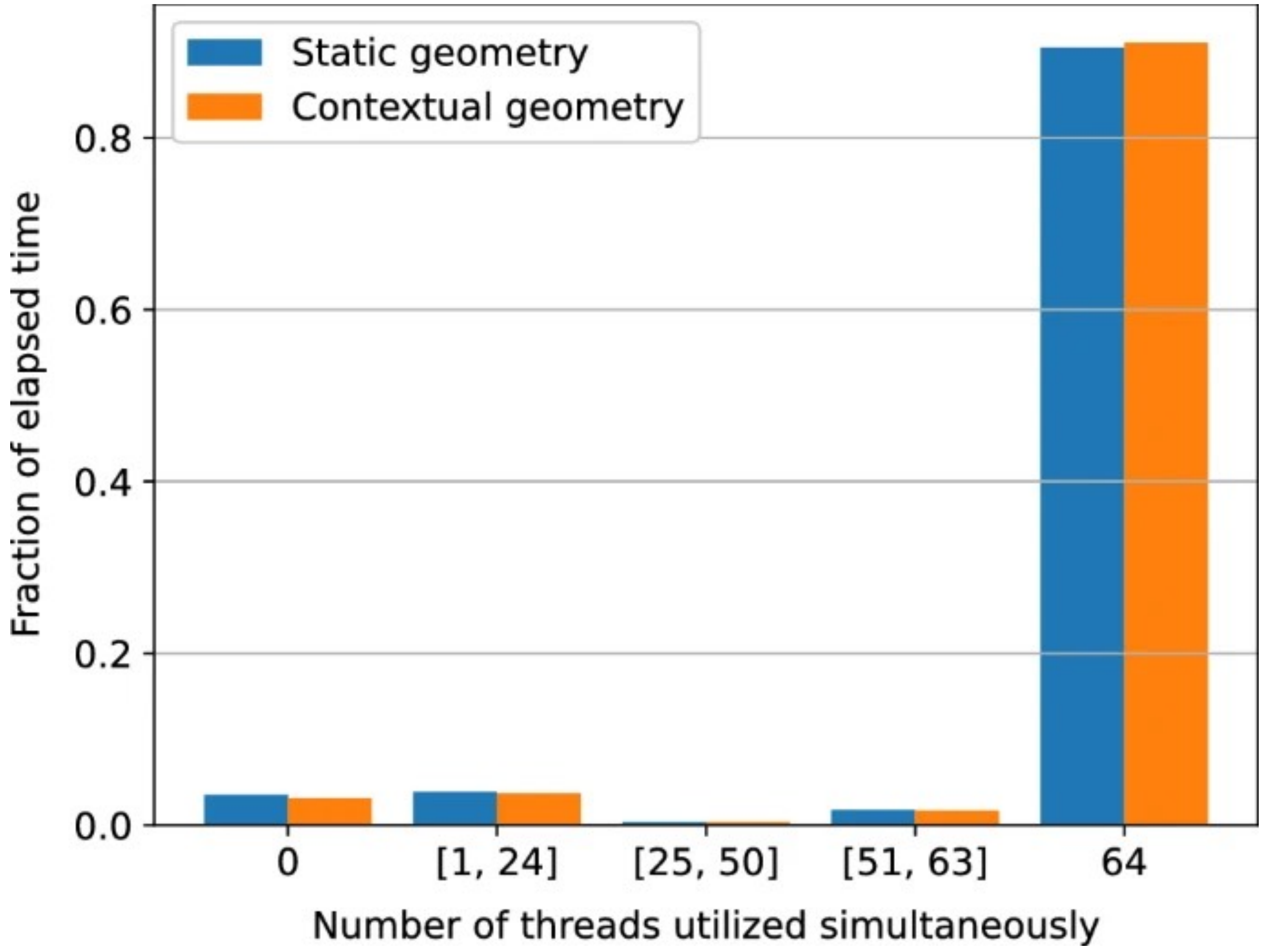
GPU

[Huth B, et al: ICHEP22](#)

CPU (single threaded)

Can we run large-scale multi threaded?

[acts-project/acts](#) designed specifically for MT application



Exa.TrkX project is the state of the art end-to-end Graph Neural Network based track finding library.

R&D1: technology BINGO

CPU GPU

acts-project/acts	x86	aarch64	oneAPI/SYCL	CUDA	acts-project/traccc acts-project/detray
Core Line	tested	tested	superseded	superseded	acts-project/covfie acts-project/vecmem
	tested (incomplete*)	tested (incomplete*)	tested (incomplete*)	tested (incomplete*)	R&D Line 1 "parallelization"
R&D Line 2 "machine learning"	tested	not tested	not implemented	tested w x86	

exatrkx & acts-project/acts