ACTS GPU-based Track Reconstruction: `detray`

EP R&D Software Working Group Meeting

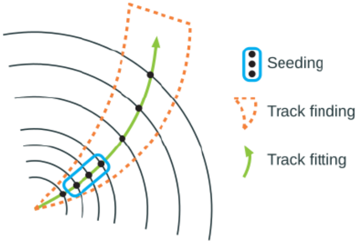
Joana Niermann
and the detray and traccc developers

11.12.2024

ACTS GPU R&D - Overview

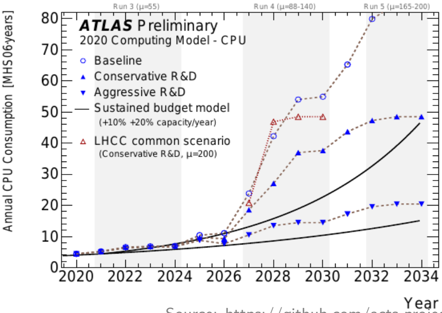
ACTS - A Common Tracking Software

- Track reconstruction toolkit,
- detector agnostic *tracking geometry*
- Clusterizsation, Seeding, Track Finding (CKF), Track Fitting (KF) and ambiguity resolution



Bringing Tracking Software to GPUs

- Tackle tracking combinatorics using massive parallelism
- Different GPU backends, e.g. CUDA or SYCL
- Polymorphic geometry is not GPU-friendly
- Vector-of-vector data structures difficult to move to device memory system

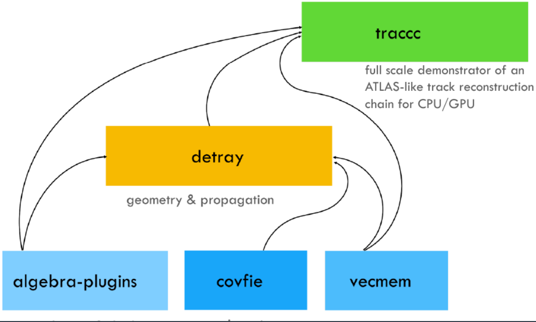


Source: https://github.com/acts-project/

ACTS GPU R&D - Overview

GPU sub-projects

- vecmem: Memory management between host and device for vector-like data structures (supports different backends, e.g. CUDA or SYCL).
- covfie: vector field description library, used for B-field
- traccc: Algorithmic chain demonstrator for track reconstruction (clusterization, seeding, CKF, KF, ambiguity resolution)
- detray: Implementation of geometry and track propagation in a parallelization friendly way.



The detray Project

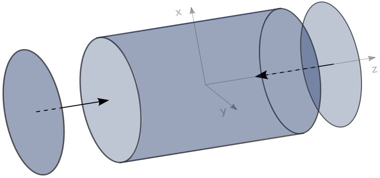
Project Outline

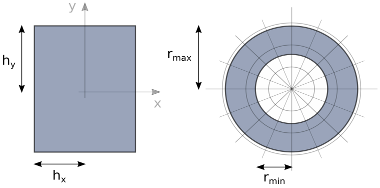
- Realistic tracking geometry description and propagation, without compromises in accuracy.
- Geometry classes without run-time polymorphism (in particular, no virtual function calls).
- Flat container structure with index based data linking.
- Implementation of core package equally usable in host and device code.

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

Geometry Description

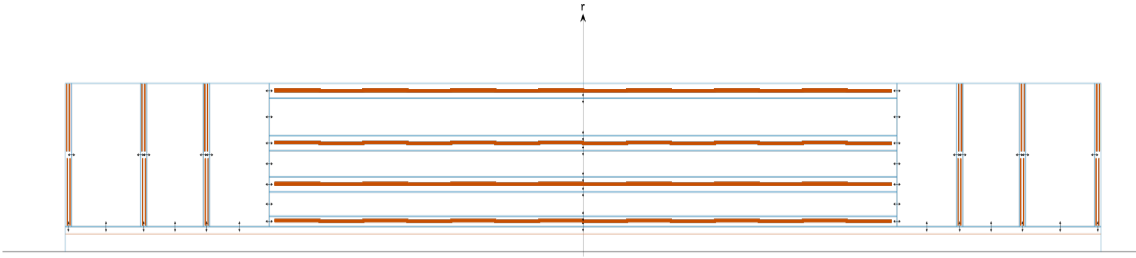
- Volumes: defined by their boundary surfaces
- Surfaces: Placed by affine transformations and defined by boundary masks
- Masks: Defined by a shape type.
  Specify local coordinates and extent of surfaces.
- Portals: Special surfaces that tie volumes together through index links.
- Material: Homogeneous *slabs* or *rods* of parametrized material or material maps (grids of slabs)





The `detray` Detector

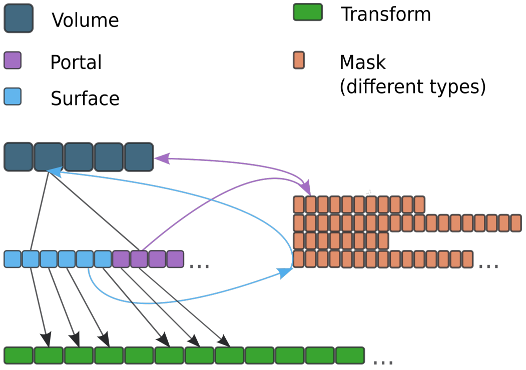
- Holds SoA data containers, indexed by surface/volume descriptors and other links (e.g. to material/grids/acceleration structures)
- Performs the container moves between host and device using `vecmem` (host detector → detector buffer → detector view)
- Provides object-oriented interface to the tracking geometry data
- Can be extended wit custom shapes, acceleration structures or material (compile-time)
- Geometry alignment needs to be updated per IoV: Load `traccc` kernel with new transform store (WIP)



Detector Container Structure

Linking by Index

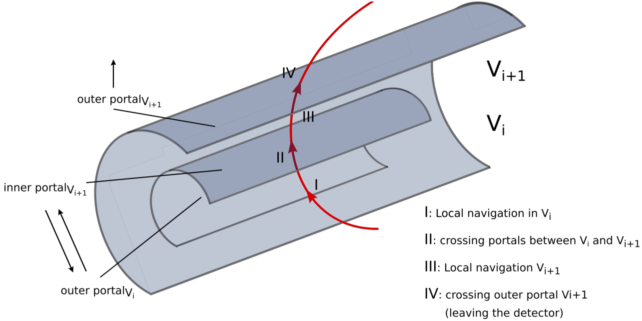
- Volumes keep a multi-index to the acceleration data structures.
- Surfaces/Portals keep indices into the transform, mask and material containers.
- Mask/material store: tuple of mask/material vectors.
- Transform store holds transformation matrices (contextual).
- Accelerator store: tuple of e.g grids.



⇒ memory layout results in differences to ACTS tracking geometry, e.g. duplicate portal descriptors.

Track State Propagation

- **Propagator:** runs the propagation loop: calls stepper, navigator and actors. Current focus of thread synchronization effort.
- **Navigator:** Moves between detector volumes and finds distance to next candidate surface.
- **Stepper:** Transports the track parameters and covariance matrix in B-field.
- **Actors/Aborters:** Extend propagation with custom functionality. Can be composed to model dependencies.



Navigation Implementation

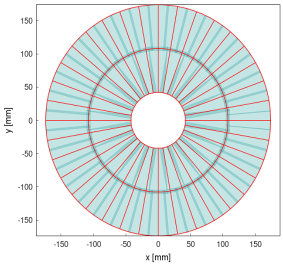
Trust-based candidate evaluation

- ...cache line-surface intersections. *trust levels* determine update method:
- ***Full trust***: Do nothing.
- ***High trust***: Only update the current next target surface.
- ***Fair trust***: Update all entries and sort again.
- ***No trust***: (Re-)initialize the entire (current) volume, i.e. fill cache and sort by distance.

$\Rightarrow$ Stepper/actors can lower trust level to influence navigation update policy.

Local Navigation in a Volume

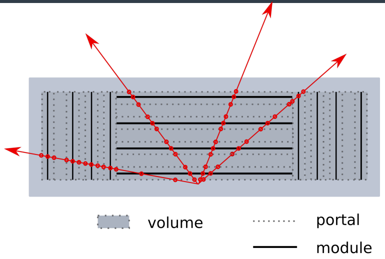
- Surface grids for local neighbourhood lookup (sensitives)
- Portals (and passives) are registered in a *brute-force* acceleration structure
- Different grid types, including dynamic bin capacity for "neighbourhood packing" and material maps

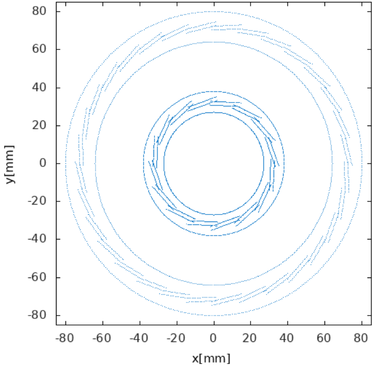


Navigation Validation

Ray Scan

- Shoot straight line rays through detector
- Record every intersection, together with associated volume index.
- Sort by distance and check for consistent crossing of adjacent portals.





Navigation Validation

- Shoot ray/helix and follow with navigator
- Compare intersection trace with navigation trace
- Has been checked for different detectors (e.g. ODD, ITk) for CPU/CUDA

Material Validation

- Collect material along ray
- Compare to material recorded during navigation
- Some small discrepancy seen, different on host and device

Navigation Implementation

Recent Development

- Fix the size of the navigation cache at compile time, renavigate if cache is exhausted
- Reduced memory footprint down to 256 bytes per track
- Use smaller navigation cache in `traccc` CKF (currently only adapted CPU implementation)

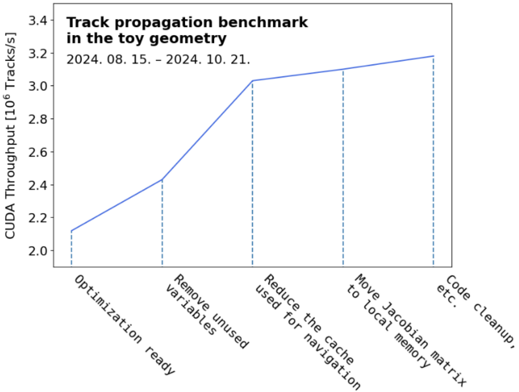
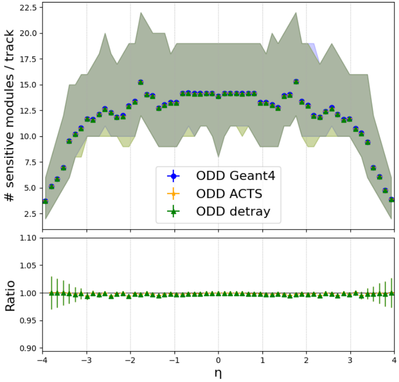


Image: https://indico.cern.ch/event/1338689/contributions/6010050/

Integration into ACTS

detray IO Library

- Readers and writers to and from an intermediate "payload" description
- All relevant detector components can be converted
- json-files that are currently used in traccc can now be dumped from ACTS



detray Plugin

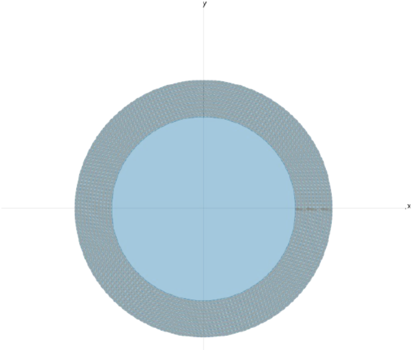
- Tracking geometry is built in ACTS (GeoModel/DD4hep)
- detray Plugin in ACTS: Convert ACTS Gen2 geometry to detray payloads
- Supports all relevant detector components (geometry, material, grids)
- Currently: comparing straight-line track state propagation

Image: https://indico.cern.ch/event/1338689/contributions/6010050/

Integration into ACTS

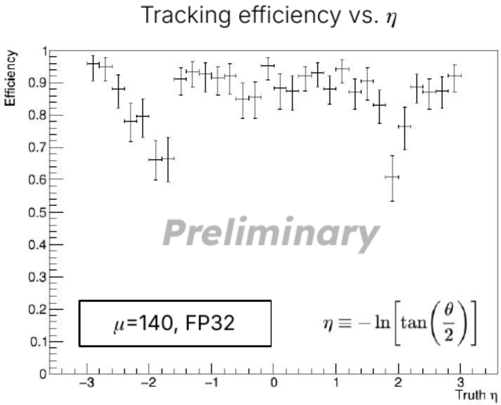
TODO

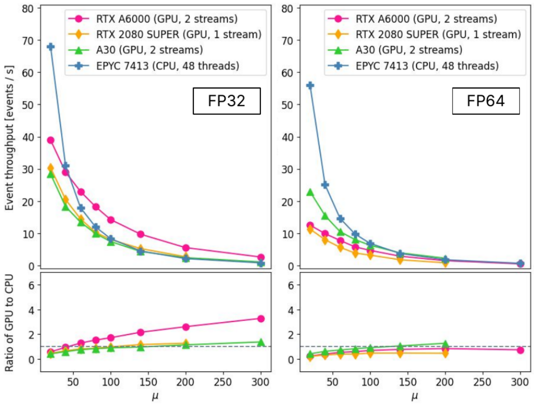
- Adapt to the new ACTS geometry building (Gen3)
- Using the ACTS DD4hep plugin, revive the FCC-hh/FCC-ee tracking geometries
- Straw tube demonstrator already exists in `detray` and has recently been ported to new `detray` IO implementation (`detector_builder`)
- Optimize detector data structure: Data deduplication and sorting



traccc Performance

traccc full chain benchmarks on the ACTS ODD:





⇒ Main performance driver: detray track propagation

Source: https://indico.cern.ch/event/1338689/contributions/6010050/

Status and Outlook

Status

- GPU-ready tracking geometry description and propagation available
- Transport of track parameters and covariance through (in-)homogeneous B-field
- Material map description for tracking (needs data deduplication)
- Linear algebra implementations using explicit vectorization (Vc) are fully available now: AoS layout shows poor performance so far, SoA layout will need adaptation of the `detray` detector
- Recently: Code cleanup and quality improvements. Code profiling and optimization have started.

Outlook

- IO optimizations: Deduplication, sorting
- Update `detray` plugin to Gen3 ACTS geometry. Export FCC detectors to `detray`
- Handle surfaces with two solutions correctly
- Backwards navigation for smoothing
- Continue performance study and optimize propagation throughput, e.g. thread synchronization