ACTS Workshop 2024

GPU Tracking Geometry and Navigation - Lessons learned

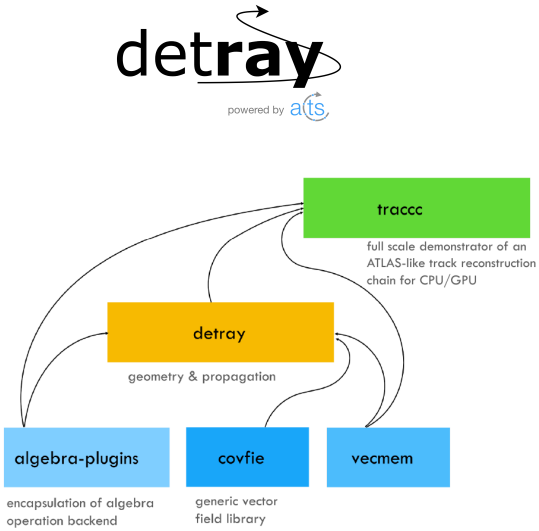
Joana Niermann
on behalf of many people

19.11.2024

The detray Project

Project Outline

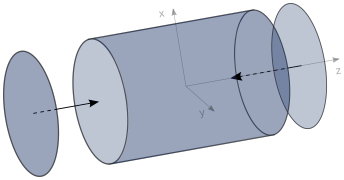
- Realistic tracking geometry description and propagation, without compromises in accuracy.
- Geometry classes without run-time polymorphism (in particular, no virtual function calls).
- Flat container structure with index based data linking.
- Implementation of core package equally usable in host and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

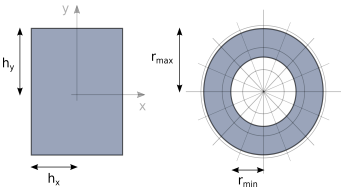


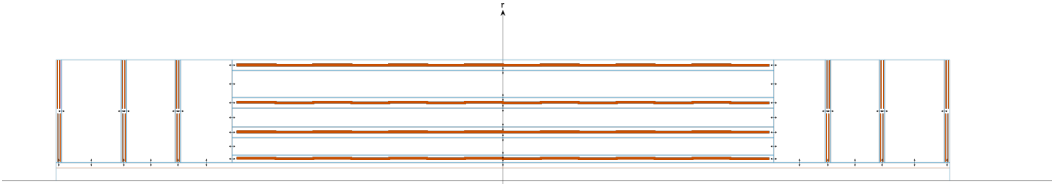
Source: https://github.com/acts-project/detray

Geometry Description

- **Surfaces**, **volumes**, **material maps**: Conceptually same as ACTS
- **Masks**: Defined by a shape type. Specify extent in local coordinates.
- Surfaces and volumes are not addressed by pointers but descriptors.
- **Detector**: Holds SoA data containers, indexed by descriptors and other links (e.g. to material/grids/acceleration structures)
- Context objects allow to switch alignment IoV (recent work by Vakho Tsulaia, hands-on session)







Detector Conversion

`detray` Plugin

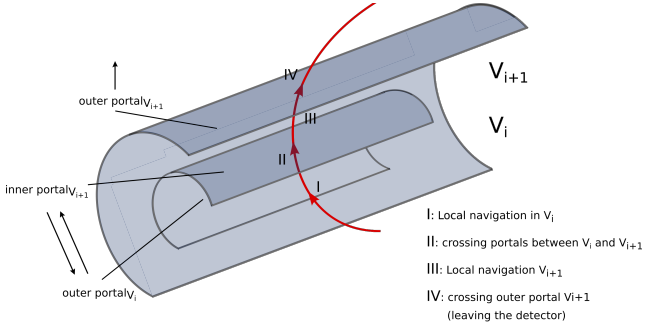
- Readers and writers to and from an intermediate "payload" description
- `detray` Plugin in ACTS: Convert ACTS Gen2 geometry to `detray` payloads (Eleni Xochelli, usage in EF-tracking ITk workflow by Neza Ribaric)
- All relevant detector components can be converted
- `json`-files that are currently used in `traccc` can now be dumped from ACTS

Open Issues

- Due to device-friendly detector data structures, portals are registered separately in every volume
- This also leads to complications with material maps on portals
- Data deduplication during IO

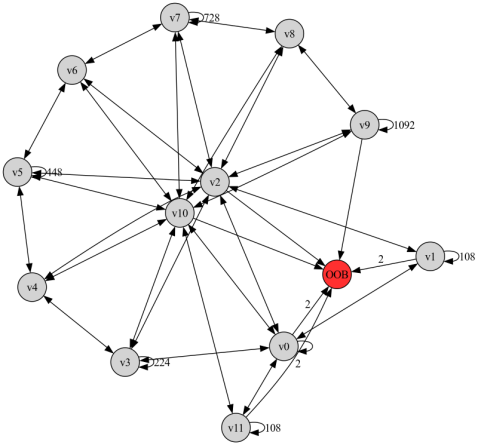
Track State Propagation

- **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 Workflow

- Volumes are nodes, linked by their portal boundary surfaces (edges).Sensitive and passive surfaces are loops.
- In each volume, multiple acceleration structures can be queried to customize navigation



⇒ Treat Portals and sensitive/passive surfaces the same: Always follow link to next volume

Navigation Implementation

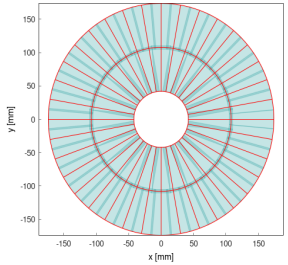
Trust-based candidate evaluation

- . . . 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.

⇒ 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



Lessons learned

- Simplify the geometry by removing layers $\Rightarrow$ eliminates layer resolution from navigation workflow
- Remove surface type resolution until surface is actually reached $\Rightarrow$ single navigation stream/cache
- Add flag to prevent re-navigation after actor update in case of bo change (trust level based cache update)
- Trust level based navigation update can be customized by navigation policies.
- Scalable mask tolerances to compensate for track curvature
- Decouple the navigator and stepper to able to reuse it outside of a `detray` ecosystem

Outlook

- Handle surfaces with two solutions and split surfaces correctly
- Backwards navigation for smoothing
- IO optimizations (Deduplication/sorting)