



ACAT 2022

Navigation, field integration and track parameter transport through detectors using GPUs and CPUs within the ACTS R&D project

Andreas Salzburger¹ Joana Niermann^{1,2} Beomki Yeo^{3,4} Stephen N. Swatman^{1,5} Attila Krasznahorkay¹

25.10.2022

¹CERN

²II. Physikalisches Institut, Georg-August-Universität Göttingen

³Department of Physics, University of California

⁴Lawrence Berkeley National Laboratory

⁵University of Amsterdam, Amsterdam, The Netherlands

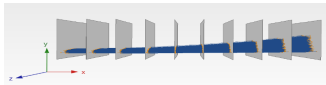
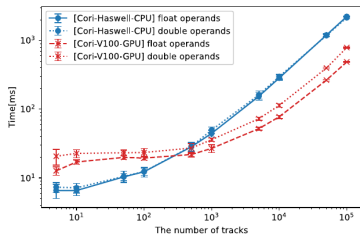
Table of Contents

1. The detray Project
2. Geometry Implementation
3. Track State Propagation
4. Summary and Outlook

Motivation

ACTS - A Common Tracking Software [1]–[3]

- Efficient, thread-safe implementation of track reconstruction tools.
- Detector agnostic *tracking geometry* description.
- For the ACTS Kalman Filter ported to GPU [4] a speedup of up to 4.6 towards a multithreaded CPU was found (events with $\gtrsim 1000$ tracks).
- ... but polymorphic geometry cannot be transferred to CUDA kernels (used telescope geometry instead).



ACTS Parallelization R&D Line [5]–[7]

- `vecmem`: Memory management between host and device for vector-like data structures (supports different backends, e.g. CUDA or SYCL).
- `tracc`: Algorithmic chain demonstrator for track reconstruction.
- `detray`: Implementation of geometry and track propagation in a parallelization friendly way.

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

Image: Performance plot and geometry setup [4]



General Considerations

- Realistic tracking geometry description, 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.

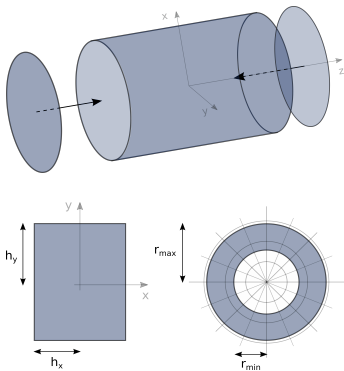
Heterogeneous Computing Model

- Goal: outsource many-track propagation to device.
- Need to handle host-device memory transfers.
- Core classes templated on STL vs. `vecmem` containers.
- Memory allocation strategy is determined by *vecmem* memory resources.
- The data structures are built host-side and then passed to the kernel via *data views*

The de~~tr~~ay Geometry Model

Building Blocks

- **Volumes:** logical containers for surfaces, defined by their boundary (portal) surfaces.
- **Surfaces:** Placed by transforms and defined by boundary masks in local coordinates.
- **Masks:** Define the shape types by specifying local coordinates and extent of surfaces.
- **Portals:** Surfaces that tie volumes together through links (No static difference to sensitive surfaces).
- **Material:** Added to a surface in same way as a mask (link + tuple unrolling). Many predefined materials available.



No abstract classes: Every type needs its own container. Solved by compile-time unrolling of tuple containers.

Detector Setup

A central Detector Implementation

- holds all geometry and magnetic field data,
- manages the container moves between host and device
- and provides an interface to all classes that need to access the tracking geometry data.

The current Testbed Geometry contains:

- A passive beampipe ($r = 19$ mm, beryllium)
- Up to four cylindrical barrel layers with between 224 and 1092 sensitive surfaces (silicon).
- Up to seven endcap layers on either side with 108 trapezoidal sensitive surfaces respectively.
- Gap volumes and portals for navigation.

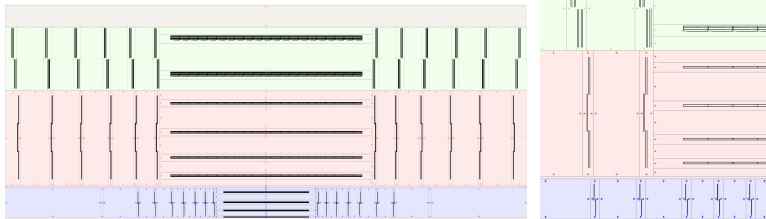
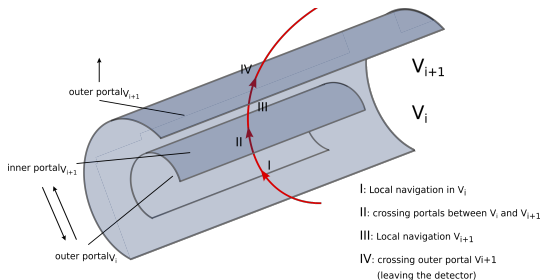


Image: The ACTS Open Data Detector implementation <https://github.com/acts-project/acts/pull/1039>

Track State Propagation

Main Participants

- **Propagator:** steers the workflow between the stepper, the navigator and the actors.
- **Navigator:** Moves between detector volumes and resolves next candidate surface.
- **Stepper:** Transports the track parameters through the geometry.
- **Actors/Aborters:** Perform various tasks during propagation and watch termination criteria.



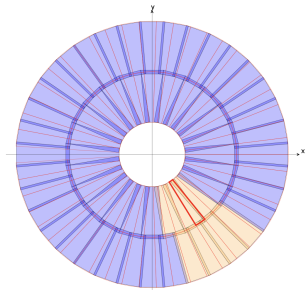
Trust-based candidate evaluation

- ... 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

- Accelerators provide neighbourhood lookups during navigation candidate search.
- Navigate local neighborhood, before reassuming volume navigation
- In principle: Any kind of accelerator possible, but currently readying the grid implementation.



The detrayer Actor Chain

```
// initialize the navigation
navigator.init(propagation);

// Run all registered actors/aborters after init
run_actors(propagation._actor_states, propagation);

// Run while there is a heartbeat
while (propagation.heartbeat) {

    // Take the step
    stepper.step(propagation);

    // And check the status
    navigator.update(propagation);

    // Run all registered actors
    run_actors(propagation.actor_states, propagation);
}
```

What is an actor in detrayer?

- Callable that performs a task after every step.
- Has a per track state, where e.g. results can be passed.
- Can be plugged in at compile time.
- In detrayer: Aborters are actors

Implementation

- Actors can 'observe' other actors, i.e. additionally act on their subject's state.
- Observing actors can be observed by other actors and so forth (resolved at compile time).
- Observer is being handed subject's state by actor chain
⇒ no need to know subject's state type and fetch it.
- Greater flexibility in testing different setups

Parameter Transport and Material Interaction

Use magnetic field map for interpolation: covfie library [10].

Field Integration [11]–[13]

- Inhomogeneous B-Field: No track solution in closed form, like a helix.
- Adaptive Runge-Kutta-Nyström algorithm for Integration.
- Gets the distance to next target surface and adjusts step-size according to integration error (B-field)
- Advances the track parameters and transport Jacobian along trajectory.

Material Interaction and Covariance Transport

- Called at every sensitive and passive material surface to add material effects to covariance (assumes thin scatterers).
- Takes energy loss effects and multiple scattering into account.
- Do Covariance transport, including transformations to/from local coordinates.

⇒ Both implemented as actors and can as such be freely composed in actor chain call.

Preliminary Benchmarks

Current status using the same code on host and device, without dedicated optimization campaign, yet:

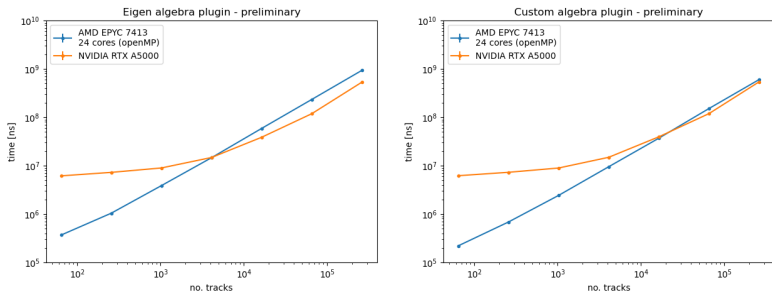


Figure 1: Benchmarks of the full propagation loop in single precision on an AMD EPYC 7413 24-Core host processor and an NVIDIA RTX A5000 device for two linear algebra implementations [14].

However, results are very preliminary at this point.

See also: algebra-plugins: <https://github.com/acts-project/algebra-plugins>

Summary and Outlook

Fully enabled host- and device-side:

- Realistic testbed detector, modelled after ACTS generic detector's pixel detector, including a simple material description.
- Integration of the covfie library for a full B-field description (currently homogeneous field).
- Adaptive Runge-Kutta-Nyström algorithm to integrate equation of motion.
- Transport of full track parametrization and covariance through (in-)homogeneous B-field and tracking geometry.
- Material Interaction actor.

Outlook

- Integration of navigation accelerators (grids are available, but not yet used).
- Interface to read existing ACTS tracking geometry implementations (almost ready).
- Extended backend support via `vecmem`, e.g. SYCL.
- Run Kalman Filter that will most likely be implemented in `traccc`.

Acknowledgements

This work benefited from support by the CERN Strategic R&D Programme on Technologies for Future Experiments (CERN-OPEN-2018-006)[18].

This work has been sponsored by the Wolfgang Gentner Programme of the German Federal Ministry of Education and Research (grant no. 05E18CHA).

This work was funded by the NSF under Cooperative Agreement OAC-1836650.

References

- [1] C. Gumpert *et al.*, "ACTS: from ATLAS software towards a common track reconstruction software," in *J. Phys.: Conf. Ser.*, vol. 898, IOP Publishing, Oct. 2017, p. 042011. DOI: 10.1088/1742-6596/898/4/042011.
- [2] A. Salzburger *et al.*, *ACTS- Project*, github, version v14.1.0, Oct. 2021. DOI: 10.5281/zenodo.5575151. [Online]. Available: <https://github.com/acts-project/acts>.
- [3] X. Ai, C. Allaire, N. Calace, *et al.*, "A common tracking software project," *Computing and Software for Big Science*, vol. 6, no. 1, pp. 1–23, 2022.
- [4] X. Ai *et al.*, "A gpu-based kalman filter for track fitting," *Computing and Software for Big Science*, vol. 5, no. 1, p. 20, Oct. 2021. DOI: 10.1007/s41781-021-00065-z.
- [5] A. Krasznahorkay *et al.*, *ACTS Project - vecmem*, 2020. [Online]. Available: <https://github.com/acts-project/vecmem> (visited on 10/24/2022).
- [6] B. Yeo *et al.*, *ACTS Project - tracc*, 2020. [Online]. Available: <https://github.com/acts-project/tracc> (visited on 10/24/2022).
- [7] A. Salzburger *et al.*, *ACTS Project - detrax: Test library for detector surface intersection*, github, 2020. [Online]. Available: <https://github.com/acts-project/detrax> (visited on 10/24/2022).
- [8] *CUDA C++ Programming Guide*, Oct. 2022. [Online]. Available: <https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html> (visited on 10/24/2022).

References

- [9] R. Reyes *et al.*, "SYCL: Single-source C++ accelerator programming," in *Parallel Computing: On the Road to Exascale*, IOS Press, 2016, pp. 673–682. DOI: 10.3233/978-1-61499-621-7-673.
- [10] S. N. Swatman, *covfie - A library for storing interpolatable vector fields on co-processors*, github, 2022. [Online]. Available: <https://github.com/acts-project/covfie> (visited on 10/24/2022).
- [11] J. Myrheim *et al.*, "A fast runge-kutta method for fitting tracks in a magnetic field," *Nucl. Instrum. Methods*, vol. 160, no. 1, pp. 43–48, 1979. DOI: [https://doi.org/10.1016/0029-554X\(79\)90163-0](https://doi.org/10.1016/0029-554X(79)90163-0).
- [12] L. Bugge and J. Myrheim, "Tracking and track fitting," *Nuclear Instruments and Methods*, vol. 179, no. 2, pp. 365–381, 1981, ISSN: 0029-554X. DOI: [https://doi.org/10.1016/0029-554X\(81\)90063-X](https://doi.org/10.1016/0029-554X(81)90063-X).
- [13] E. Lund, L. Bugge, I. Gavrilenko, *et al.*, "Track parameter propagation through the application of a new adaptive runge-kutta-nyström method in the atlas experiment," *Journal of Instrumentation*, vol. 4, no. 04, P04001, Apr. 2009. DOI: 10.1088/1748-0221/4/04/P04001.
- [14] A. Krasznahorkay *et al.*, *ACTS Project - algebra plugins*, 2020. [Online]. Available: <https://github.com/acts-project/algebra-plugins> (visited on 10/24/2022).
- [15] G. Guennebaud *et al.*, *Eigen v3*, <http://eigen.tuxfamily.org>, 2010. (visited on 11/16/2021).

References

- [16] R. Brun *et al.*, "ROOT - An object oriented data analysis framework," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 389, no. 1, pp. 81–86, 1997. DOI: 10.1016/S0168-9002(97)00048-X.
- [17] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *IEEE Computational Science and Engineering*, vol. 5, no. 1, pp. 46–55, 1998. DOI: 10.1109/99.660313.
- [18] M. Aleksa *et al.*, "Strategic R&D Programme on Technologies for Future Experiments," CERN, Geneva, Tech. Rep., Dec. 2018, CERN-OPEN-2018-006. [Online]. Available: <https://cds.cern.ch/record/2649646>.
- [19] J. Nickolls *et al.*, "Scalable Parallel Programming with CUDA: Is CUDA the Parallel Programming Model That Application Developers Have Been Waiting For?" *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008. DOI: 10.1145/1365490.1365500. [Online]. Available: <https://doi.org/10.1145/1365490.1365500>.
- [20] N. Bell *et al.*, "Thrust: A productivity-oriented library for cuda," in *GPU Computing Gems Jade Edition*, ser. Applications of GPU Computing Series, Boston: Morgan Kaufmann, 2012, pp. 359–371. DOI: <https://doi.org/10.1016/B978-0-12-385963-1.00026-5>.
- [21] C. Allaire *et al.*, *OpenDataDetector*, gitlab, version v1, 2021. DOI: 10.5281/zenodo.4674402. [Online]. Available: <https://gitlab.cern.ch/acts/OpenDataDetector/>.

Heterogeneous Computing Model

Implementation in detray

- Goal: outsource many-track propagation to device.
- Need to handle host-device memory transfers.
- Core classes templated on STL vs. vecmem containers.
- The geometry data structures are built host side and memory allocation strategy is determined by *vecmem memory resources*.

```
#include <vecmem/containers/vector.hpp>

// Transform store using managed memory
vecmem::cuda::managed_memory_resource mng_mr;

// Build with host vector type
transform_store<vecmem::vector> store(mng_mr);

// Get store view object
auto sv = detray::get_data(store);

// Run the kernel
test_kernel<<<block_dim, thread_dim>>>(sv);
```

```
#include <vecmem/containers/device_vector.hpp>

// Kernel-side construction
__global__ void test_kernel(store_view sv) {
    // Build with device vector type
    transform_store<vecmem::device_vector> store(sv);

    // Do something
}
```


Open Data Detector - Overview

Kaggle TrackML challenge

- Generic Tracking Detector design
- Reduced physics list in fastsim
- But: afterwards dataset was used for further tracking R&D

⇒ Provide simplified generic, but more realistic dataset!

Next level: The Open Data Detector [21]

- More realistic detector description
- 4 layer Pixel detector
- Short- and Long-Strip detector
- Detector mounting, cables, cooling . . .

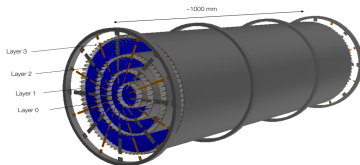
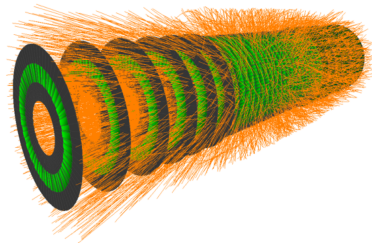


Image: (top) <https://sites.google.com/site/trackmlparticle/>