SPONSORED BY THE
Federal Ministry of Education and Research
EP-R&D
Programme on Technologies for Future Experiments

ACAT 2021

Detray - A Compile Time Polymorphic Tracking Geometry Description

Andreas Salzburger[1]
Joana Niermann[1,2]
Beomki Yeo[3,4]
Attila Krasznahorkay[1]

01.12.2021

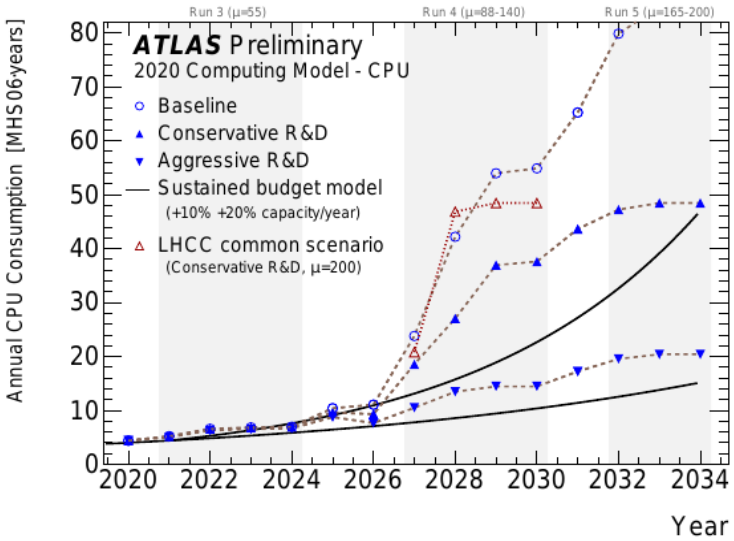
[1] CERN
[2] II. Physikalisches Institut, Georg-August-Universität Göttingen
[3] Department of Physics, University of California
[4] Lawrence Berkeley National Laboratory

Table of Contents

Motivation

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

- Detector agnostic C++ tracking toolkit.
- Efficient implementation of common tasks (e.g seed finding, track fitting . . .)
- Thread-safe implementation, including conditional data (e.g. alignment).
- Uses a simplified geometry description for tracking purposes.



Tracking on GPU

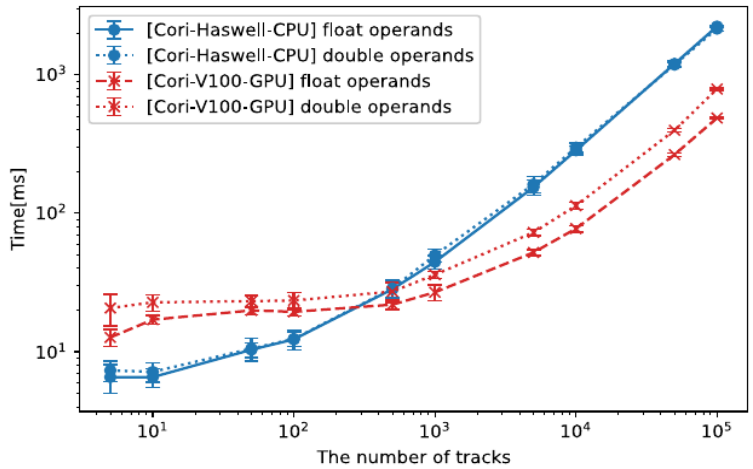
- To tackle the combinatorics in a high luminosity environment, investigate tracking on GPU.
- For this to succeed, define a suitable Event Data Model (EDM),
- develop a toolchain that supports e.g. CUDA kernels
- and provide GPU friendly implementations of the geometry and magnetic field.

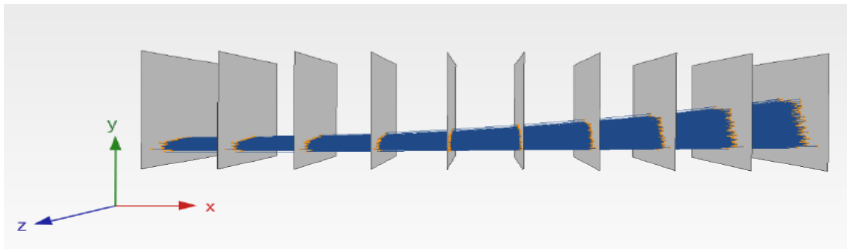
Image: Annual CPU Consumption [4]

The ACTS Kalman Filter on GPU

GPU Kalman Filter Design and Results [5]

- Sequential track fitting shows an almost linear rise with number of tracks.
- Investigation of intra- and inter-track parallelization in a telescope geometry.
- Speedup of up to 4.6 (fig on the right) towards multithreaded CPU, in events with more than 1000 tracks (using a custom matrix inversion algorithm).





A few Challenges Encountered

- Polymorphic geometry cannot be transferred to CUDA kernels. The "Curiously Recurring Template" pattern was used instead.
- Large amount of code revision toward ACTS. Code needs to be compiled with CUDA support.
- Telescope geometry and single surface type propagation are not very realistic.

Image: Performance plot and geometry setup [5]

See also: Poster: https://indico.cern.ch/event/855454/contributions/4596414/

The Current ACTS Parallelization R&D Landscape

vecmem [6]

- Memory management between host and device for vector-like data structures.
- Supports different backends (host, CUDA, SYCL, HIP).

traccc [7]

- Algorithmic chain demonstrator for track reconstruction.
- Benchmark performance in realistic detector and experimental setup.
- So far: EDM based on vecmem containers, track seeding and clusterization.

detray [8]

- Implement tracking geometry in a parallelization friendly way.
- Provide navigator(s) on that geometry.
- Implement parallelized stepping and propagation with its navigation.
- Allow for a material description.

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

See also: vecmem Talk: https://indico.cern.ch/event/855454/contributions/4605054/

The Detray Project

Initial Conception: Build a fast and simple **tracking** geometry and navigation that can be deployed on GPU.

General Considerations

- Geometry classes without runtime polymorphism (no virtual function calls).
- Flat container structure using vecmem, with index based data linking.
- Implementation of core package usable in host and device code.

Drop-In Navigation Backend for ACTS

- **Volumes** subdivide the detector into smaller areas.
- **Surfaces** as core building blocks that can either represent module surfaces or volume portals.
- A **Grid** as volume and surface finder accelerator (global searches and local navigation).
- Navigation between volumes via portals with subsequent local search for next surface.
- Read in files dumped from ACTS $\Rightarrow$ Detray can make use of existing tracking geometries.

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

The Detray Geometry Model

Structure

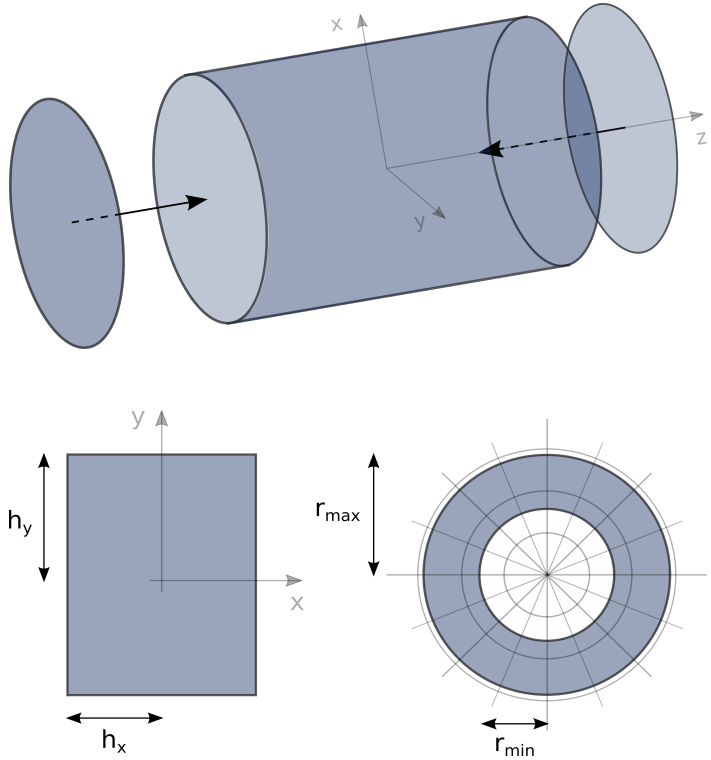
- **Detector** class as interface (builds and contains all data containers as well as the grid).
- **Navigator** moves through geometry by its links and feeds the data containers to intersection kernel.
- **Stepper** and **Propagator** classes steer navigation status and target calls.

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.
- **Portals**: Surfaces that tie volumes together through links.

Supported shapes: Rectangles, trapezoids, ring/discs, cylinders and an annulus shape.

No runtime polymorphism



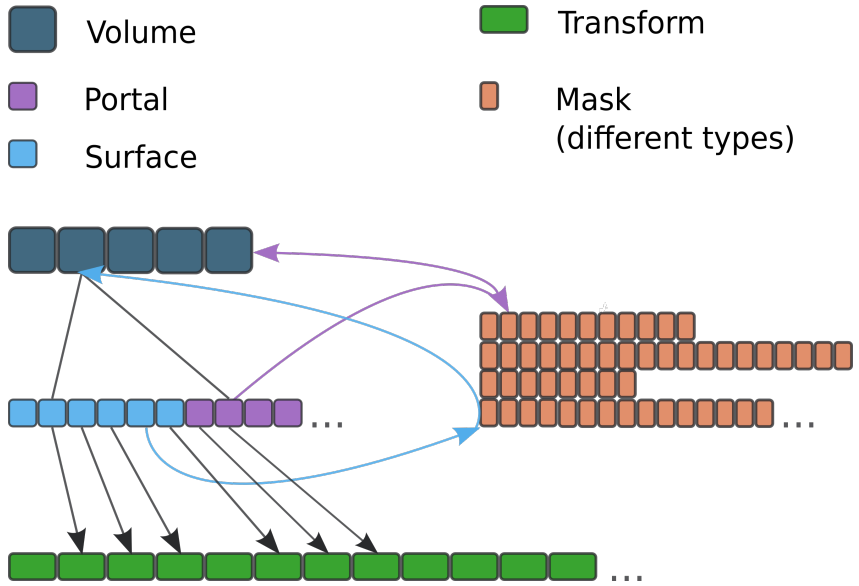
Geometry Implementation

All geometry containers are flat and serialized in memory (no vectors of vectors).

Geometry objects are uniquely identified by their respective index into the global containers.

Linking by Index

- Volumes keep index ranges into surface/portal containers.
- Surfaces/Portals keep indices into the transform and mask containers.
- Portals link to adjacent volume and next surfaces finder (local grid).
- Surfaces link back to mother volume.



⇒ Every type needs its own container: Compile time unrolling of mask container

⇒ Surface/portal links can be interpreted as a graph with volume nodes (navigation graph).

A Transparent Linear Algebra Implementation

Repository for implementation of linear algebra functionality for both detray and traccc [9]

Linear Algebra Backends

- Implements necessary functionality for particle transport, which can be switched at compile time.
- Backend (storage type), e.g.: `std::array`, `Eigen3`, `Vc::SimdArray`
- Frontend: Defines the mathematical interface (e.g. array-like access).

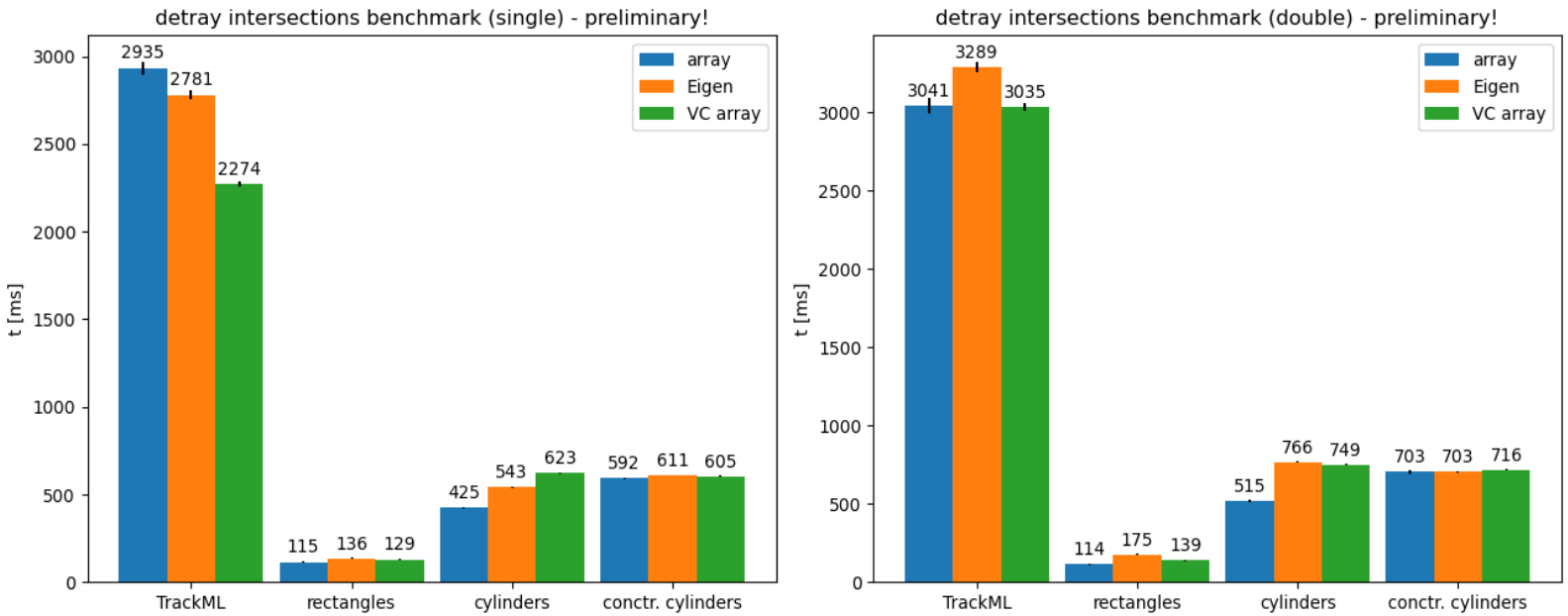
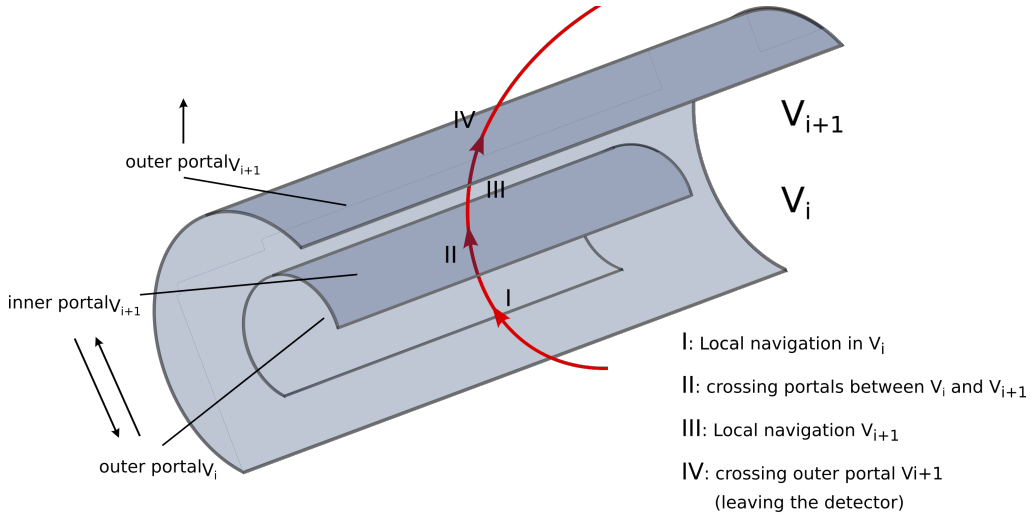


Figure 1: Intersection benchmarks with detray (v0.2.0) on an AMD Ryzen Threadripper 3970X in single and double precision for three different LA implementations.

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

Navigation Model

- **Propagator:** steers the workflow between the stepper, navigator and in future, compile-time configurable actions.
- **Stepper:** Advances the track-state through the geometry (currently straight-line stepper, Runge-Kutta to be added).
- **Navigator:** Provides the next candidate surface for the propagation.

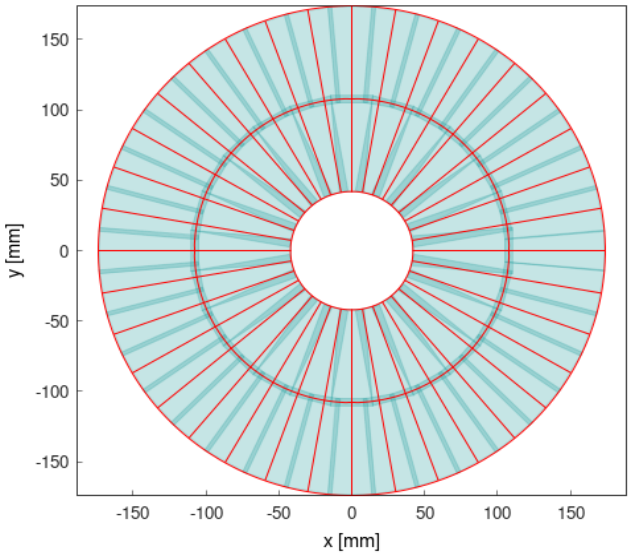


Trust-based Candidate Evaluation

- The trust state models the quality of the current candidate list.
- **No Trust**/Volume initialization: Gather all surfaces and portals of the volume for intersection. The resulting candidates are sorted by distance (shortest distance is returned to stepper).
- **High Trust**: After status update (stepping or by actor), only keep re-evaluating the next candidate until we find a hit.
- **Fair Trust**: All candidates are re-evaluated.

Accelerator Structure - The Grid

Volumes in outer layers can contain thousands of surfaces. Testing all of them during navigation is costly!



$\Rightarrow$ Model geometry object locality by binning the detector space and placing every object into the appropriate bin.

Local Navigation in a Volume

- A **Volume Grid** allows to query for volumes by position
- **Surface Finders** provide neighbourhood lookups during navigation candidate search.
- Results in a lower number of candidates to be evaluated.

Geometry Validation

Ray Scan

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

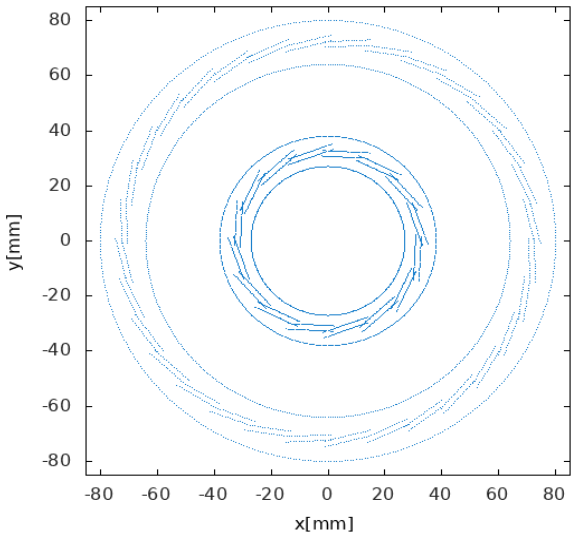
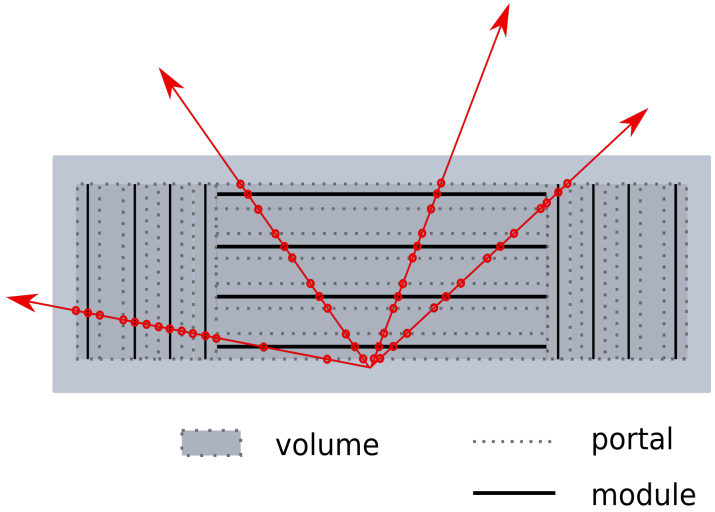


Figure 2: Intersections produced by ray scan. Two inner barrel layers modelled after ACTS Generic Detector (TrackML challenge).



Geometry Linking Validation

- Compare ray scan with geometry linking graph.
- Provides a coarse, but automated check of geometric setup.

Navigation Validation

- Shoot ray, but this time follow with navigator.
- Compare the entire intersection trace with the objects encountered by navigator.

See also: Fig2(TrackML) `https://sites.google.com/site/trackmlparticle/`

Heterogeneous Computing Model

GPU vs. CPU

- Huge number of processing cores allow for massive parallelization capability.
- Distinct computing model often results in separate code bases.
- E.g. more complicated memory management.

Implementation in Detray

- Goal: outsource many-track navigation to device.
- Core classes can be adapted for device without code duplication by virtue of templating and usage of vecmem containers.
- The geometry data structures are built host side and then made available to device by use of vecmem memory resources or in single large copy operations.

Heterogeneous Implementation - An Example

Modified code example of the transform store:

```
#include <vecmem/containers/data/vector_view.hpp>

template <template <typename...> class vector_t>
class transform_store {
 public:
  // Constructor using vecmem memory resource
  transform_store(vecmem::memory_resource &mr)
    : _data(&mr) {}

  // Constructor using vecmem view objects
  DETRAY_DEVICE transform_store(store_view &sd)
    : _data(sd._view) {}

  // Do something (host and/or device)

 private:
  vector_t<transform3> _data;
};

// Provides the view object(s) to the kernel
struct store_view {
  // Constructor from transform store
  store_view(transform_store &ts)
    : _view(vecmem::get_data(ts.data())) {}

  // View object to be passed to device
  vecmem::data::vector_view<transform3> _view;
};
```

```
#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
}
```

```
#include <vecmem/containers/vector.hpp>
#include <vecmem/memory/cuda/managed_memory_resource.hp

// 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 = store_view(store);

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

Summary

Projects Status

- Complete implementation of index based tracking geometry without runtime polymorphism, using flat containers.
- Validation of the geometry implementation by various consistency checks.
- Successful straight-line propagation through multiple layers of volumes using the new tracking geometry description.
- Successful porting of most of the core implementation to the vecmem data management and to device, avoiding code duplication.

Outlook

- Performance optimization by local navigation on the grid.
- Go from straight-line propagation to Runge-Kutta stepper.
- Complete implementation of propagation demonstrator on device.
- Make full use of the detray navigation in the traccc project and study algorithmic performance in a fully featured geometric setup, including navigation.
- The successful setup of using only volumes is currently considered to be integrated into ACTS.

Acknowledgements

This work benefited from support by the CERN Strategic R&D Programme on Technologies for Future Experiments (CERN-OPEN-2018-006)[17].

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

References

[1] C. Gumpert, A. Salzburger, M. Kiehn, *et al.*, "ACTS: from ATLAS software towards a common track reconstruction software," in *J. Phys.: Conf. Ser.*, vol. 898, IOP Publishing, Oct. 2017, p. 042 011. DOI: 10.1088/1742-6596/898/4/042011.

[2] A. Salzburger, P. Gessinger, F. Klimpel, *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*, 2021. arXiv: 2106.13593 [physics.ins-det].

[4] P. Calafiura, J. Catmore, D. Costanzo, *et al.*, "ATLAS HL-LHC Computing Conceptual Design Report," CERN, Geneva, Tech. Rep., Sep. 2020, CERN-LHCC-2017-020, ATLAS-TDR-029. [Online]. Available: https://cds.cern.ch/record/2729668.

[5] X. Ai, G. Mania, H. M. Gray, *et al.*, "A GPU-based Kalman Filter for Track Fitting," *Computing and Software for Big Science*, May 2021. arXiv: 2105.01796 [physics.ins-det]. [Online]. Available: https://doi.org/10.1007/s41781-021-00065-z.

[6] A. Krasznahorkay, S. N. Swatman, P. Gessinger, *et al.*, *ACTS Project - vecmem: Vectorised data model base and helper classes*, github, 2020. [Online]. Available: https://github.com/acts-project/vecmem (visited on 11/16/2021).

[7] B. Yeo, S. N. Swatman, A. Salzburger, *et al.*, *ACTS Project - traccc: Demonstrator tracking chain for accelerators*, github, 2020. [Online]. Available: https://github.com/acts-project/traccc (visited on 11/16/2021).

References

[8] A. Salzburger, J. Niermann, B. Yeo, *et al.*, *ACTS Project - detray: Test library for detector surface intersection*, github, 2020. [Online]. Available: https://github.com/acts-project/detray (visited on 11/16/2021).

[9] J. Niermann, A. Krasznahorkay, B. Yeo, *et al.*, *ACTS Project - algebra plugins*, github, 2020. [Online]. Available: https://github.com/acts-project/algebra-plugins (visited on 11/16/2021).

[10] G. Guennebaud, B. Jacob, *et al.*, *Eigen v3*, http://eigen.tuxfamily.org, 2010. (visited on 11/16/2021).

[11] M. Kretz and V. Lindenstruth, "Vc: A C++ library for explicit vectorization," *Software Pract. Exper.*, vol. 42, 2012.

[12] M. Kretz, C. Engwer, and B. M. Gruber, *Vc: portable, zero-overhead C++ types for explicitly data-parallel programming*, github, 2021. [Online]. Available: https://github.com/VcDevel/Vc (visited on 11/16/2021).

[13] R. Brun and F. Rademakers, "ROOT - An object oriented data analysis framework," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 389, no. 1, pp. 81–86, 1997, ISSN: 0168-9002. DOI: 10.1016/S0168-9002(97)00048-X.

[14] *CUDA C++ Programming Guide*, Nov. 2021. [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html (visited on 11/29/2021).

References

[15] J. Nickolls, I. Buck, M. Garland, *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, ISSN: 1542-7730. DOI: 10.1145/1365490.1365500. [Online]. Available: https://doi.org/10.1145/1365490.1365500.

[16] N. Bell and J. Hoberock, "Thrust: A Productivity-Oriented Library for CUDA," in W.-m. W. Hwu, Ed., Jade Edition, ser. GPU Computing Gems. Dec. 2012, pp. 359–371, ISBN: 9780123859631. DOI: 10.1016/B978-0-12-385963-1.00026-5.

[17] M. Aleksa, J. Blomer, B. Cure, *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.

[18] S. N. Swatman, "Applications of Graphical Texture Features to Magnetic Fields," https://github.com/acts-projects/traccc/pull/94/files, 2021.

[19] C. Allaire, P. Gessinger, J. Hdrinka, *et al.*, *OpenDataDetector*, gitlab, version v1, 2021. DOI: 10.5281/zenodo.4674402. [Online]. Available: https://gitlab.cern.ch/acts/OpenDataDetector/.

Backup

A Magnetic Field Implementation in Texture Memory

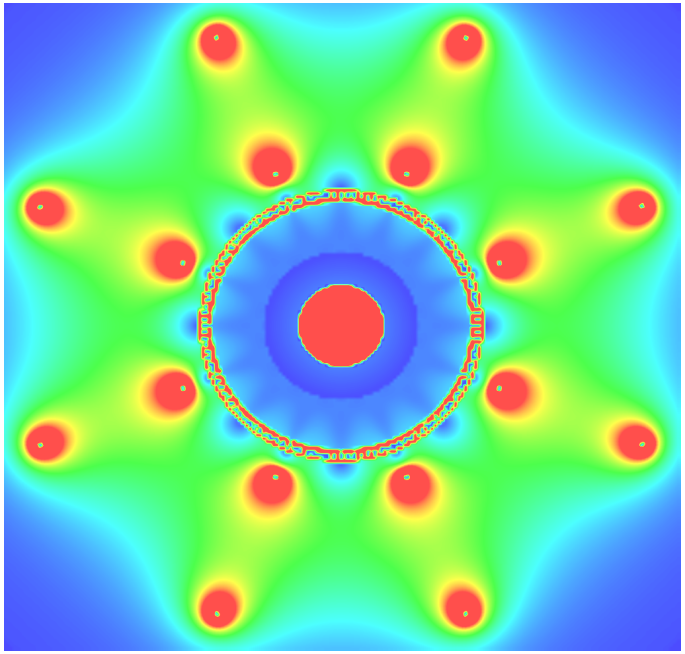


Figure 3: The ATLAS magnetic field at $z = 0\,\mathrm{mm}$ (1024 $\times$ 1024)

Textures as Magnetic Field Maps [18]

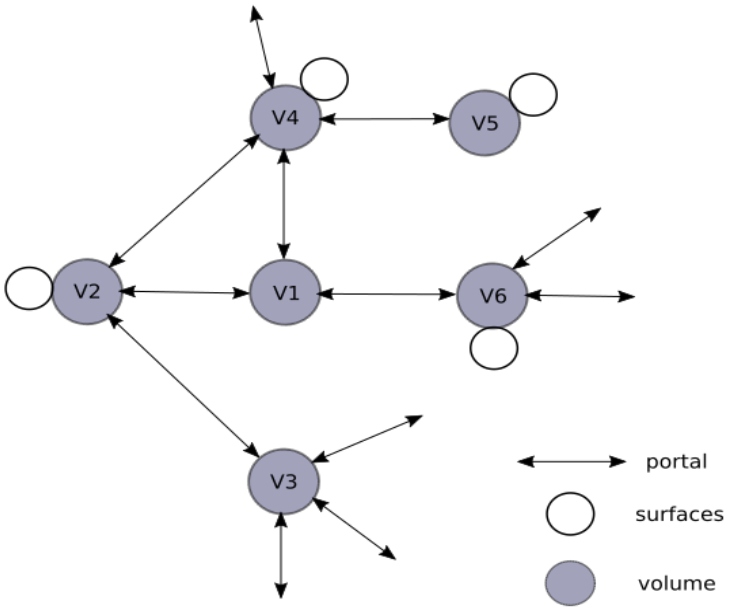
- Read in the field map from text file and facilitate lookups in kernel.
- Textures map positions in space to RGB colour space. This can be reinterpreted as vector field (RGB colour space encodes $\mathbb{R}^3$).
- Makes use of dedicated hardware and highly optimized vendor implementations (e.g. for interpolation)

Image: https://github.com/acts-project/traccc/pull/94

Navigation as a Directed Graph

Graph Data Structure

- Volume links carried by portals and surfaces can be displayed as graph data structure.
- The same graph is being followed in the Detray navigation model.
- It can be generated directly from the geometry implementation, or discovered during e.g. ray scan.



Benefits for Navigation

- No static distinction between portals and surfaces, just follow volume links.
- Less bookkeeping between portal and surface kernel structures.
- Better suited for parallelization, since all intersections are treated in the same data structure.

Image: Arbitrary graph structure, does not represent an actual geometry.

Toy Geometry - The TML Pixel Detector

Toy Geometry

- Implement a small geometry, independent from io module
- All links are manually checked for consistency

The toy geometry contains:

- A beampipe ($r = 27\,\mathrm{mm}$)
- An inner layer ($r_{\mathrm{min}} = 27\,\mathrm{mm}$, $r_{\mathrm{max}} = 38\,\mathrm{mm}$) with 224 pixel module surfaces
- A gap volume ($r_{\mathrm{min}} = 38\,\mathrm{mm}$, $r_{\mathrm{max}} = 64\,\mathrm{mm}$)
- An outer layer ($r_{\mathrm{min}} = 64\,\mathrm{mm}$, $r_{\mathrm{max}} = 80\,\mathrm{mm}$) with 448 pixel module surfaces
- Add grid will be added for local navigation.

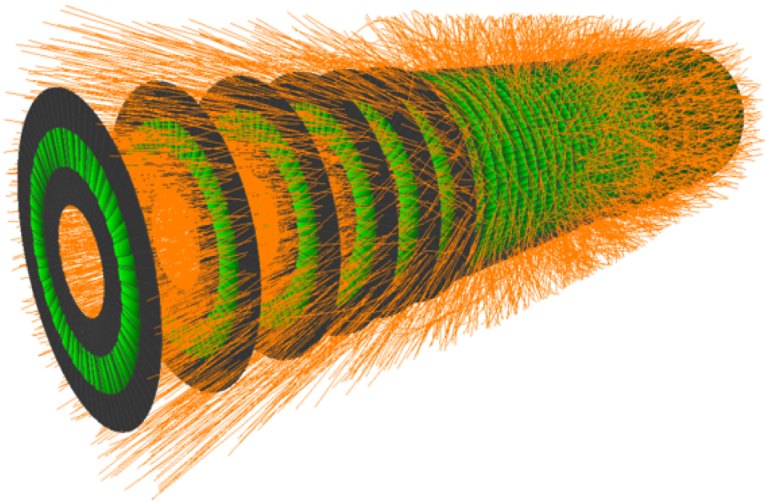
$\Rightarrow$ Provides a reliable, dynamically generated geometry that can be used for testing and rapid development.

$\Rightarrow$ More extended implementation (more layers, including endcaps) will be added soon.

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 [19]

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

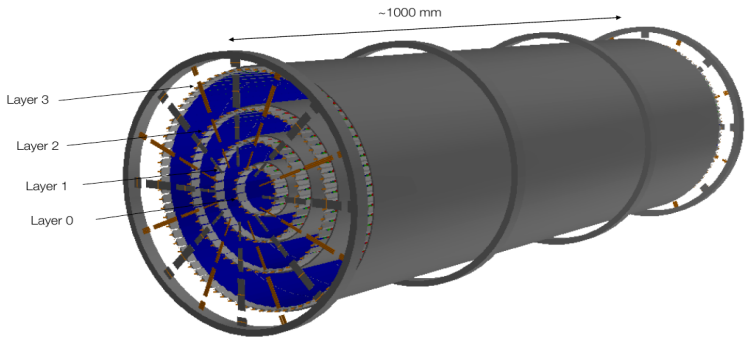


Image: (top) https://sites.google.com/site/trackmlparticle/