Grant Agreement No: 101004761

AIDAinnova

Advancement and Innovation for Detectors at Accelerators
Horizon 2020 Research Infrastructures project AIDAINNOVA

MILESTONE REPORT

ACTS TRACKING ALGORITHM PROTOTYPES

MILESTONE: MS49

Document identifier:	AIDAinnova-MS49
Due date of milestone:	End of Month 23 (February 2023)
Report release date:	28/02/2023
Work package:	WP12: Software for Future Detectors
Lead beneficiary:	CNRS
Document status:	Final

Abstract:

The AIDAinnova project is expanding the capabilities of the Acts track reconstruction toolkit in order to make it the state-of-the-art experiment independent tracking toolkit and to develop further R&D directions. In the first 23 months of the project this investment has yielded a number of outcomes: software engineering foundations have been solidified, many state-of-the-art algorithms were integrated, new machine learning approaches were investigated, general usability from experiment frameworks was improved, and GPU R&D has progressed to the point where some of these ideas start feeding back into the Acts core.

AIDAinnova Consortium, 2023
For more information on AIDAinnova, its partners and contributors please see http://aidainnova.web.cern.ch/

The Advancement and Innovation for Detectors at Accelerators (AIDAinnova) project has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 101004761. AIDAinnova began in April 2021 and will run for 4 years.

Delivery Slip

	Name	Partner	Date
Authored by	Hadrien Grasland	CNRS	06/02/2023
Edited by	Frank Gaede [WP coordinator] Graeme Stewart [WP coordinator]	DESY CERN	20/02/2023
Reviewed by	Frank Gaede [WP coordinator] Graeme Stewart [WP coordinator]	DESY CERN	27/02/2023
Approved by	Paolo Giacomelli [Scientific coordinator]	INFN	28/02/2023

TABLE OF CONTENTS

Executive summary

During the first 23 months of the AIDAinnova project, the Acts tracking project has been improved in many different ways by contributing institutions.

Foundational components grew to sustain future growth, with the introduction of Python bindings, automated physics validation, and deeper build monitoring.

Several extra state-of-the-art tracking algorithms are now supported, and new approaches based on machine learning techniques have been tested.

Usability by experiments has been improved, in large part due to enhanced support for detector description toolkits and event data models in common use.

Finally, GPU R&D has made significant progress, reaching the point where some of its developments start being integrated back into the Acts core.

1. INTRODUCTION

Advances in detector technology call for matching improvements in the data processing software that is used to extract scientific meaning from detector output. One particularly important area of focus is track reconstruction or "tracking". This is an early data processing stage where the existence, trajectory, and charge/momentum ratio of charged particles are inferred from the energy deposits that they leave behind as they traverse a detector.

At the start of LHC Run 2, it had become clear that the ATLAS experiment's tracking code needed a major overhaul, acknowledging lessons learned during Run 1 and building on them to withstand the increased demands of upcoming ATLAS and LHC upgrades. However, instead of carrying out this rewrite in an experiment specific way, it was decided to develop Acts as an independent open-source library, with the intent that it could be more easily used and improved upon by other HEP experiments and contributors outside of HEP. Thus, the Acts tracking toolkit project [1] was started in 2016 and has succeeded in its goal of establishing a shared HEP tracking code development platform.

However, the development of Acts is far from over. Indeed, as the AIDAinnova project started, Acts lacked many well-known tracking algorithms in common use, and could not provide answers to emerging R&D questions such as the applicability of GPU hardware and machine learning techniques to tracking. During these past 23 months, it has been a focus of the WP12 Track Reconstruction task to address this situation.

Besides the changes carried out as part of AIDAinnova, that will be discussed further in the following, one notable "environmental" change which affects this milestone is that, in recent years, the TrackML detector has largely been superseded by the OpenDataDetector [2], a successor project which features several major improvements including correct handling of material scattering effects. Therefore, instead of being tested using the TrackML detector, as initially planned, new Acts algorithms developed during AIDAinnova are instead tested using the OpenDataDetector.

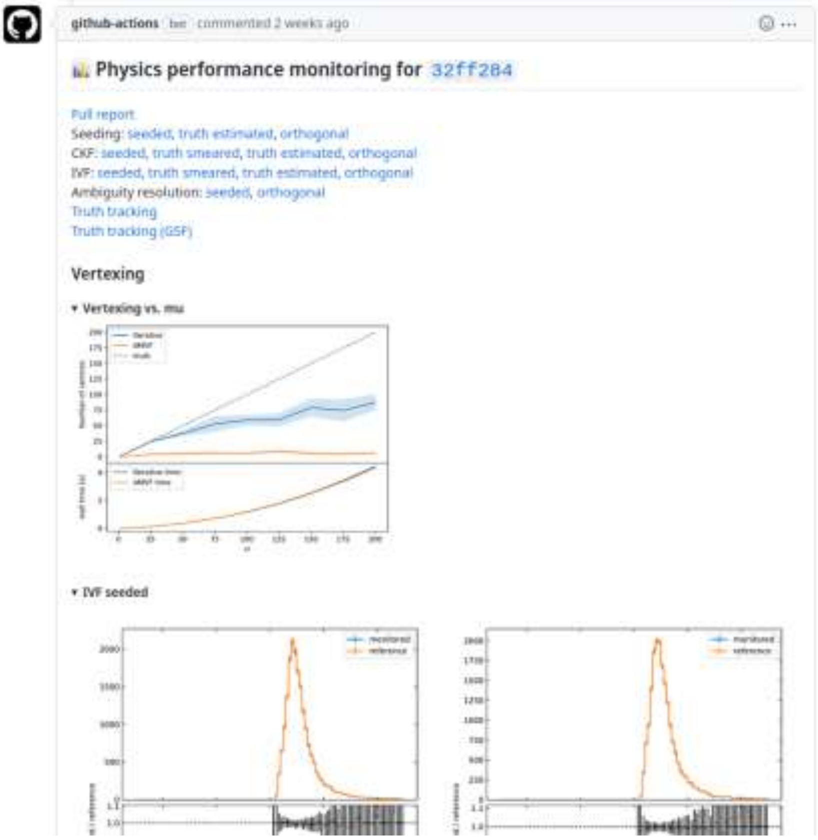
2. INFRASTRUCTURE

In addition to narrowly scoped unit and integration tests targeting individual components, Acts also provides more elaborate usage examples featuring semi-realistic physics use cases such as fast simulation or track reconstruction. These examples serve two purposes: they complement written documentation, by providing new users with a more hands-on introduction, and they can be used for physics validation by checking that the output of a well-known task matches expectations.

However, before AIDAinnova, these examples were also plagued by two issues. The first one was that they were initially implemented as C++ programs, configured through command-line options. This turned out to be a significant maintenance and ergonomics burden as the number of command-line options grew to accommodate more and more use cases. The problem was addressed during the first year of AIDAinnova by introducing **Python bindings**, which allowed a lot more user-side configuration and scripting without Acts team involvement. This also paved the way to using Acts inside Python-based machine learning frameworks, as discussed later in this report.

Another issue concerned physics validation. Before AIDAinnova started, it was a slow and manual process requiring significant expertise. As a result, it was rarely done, so issues introduced by new developments might only be noticed much later, making causal attribution difficult. This was resolved by integrating some **automated validation** into the project's Continuous Integration (CI) infrastructure (Fig. 1), ensuring that the impact of any new code submission on physics results would always receive at least some scrutiny during code review.

These two changes brought Acts closer to the state-of-the-art of tracking software.



github-actions bot commented 2 weeks ago

Physics performance monitoring for 32ff284

Full report
Seeding: seeded, truth estimated, orthogonal
CKF: seeded, truth smeared, truth estimated, orthogonal
IVF: seeded, truth smeared, truth estimated, orthogonal
Ambiguity resolution: seeded, orthogonal
Truth tracking
Truth tracking (GSF)

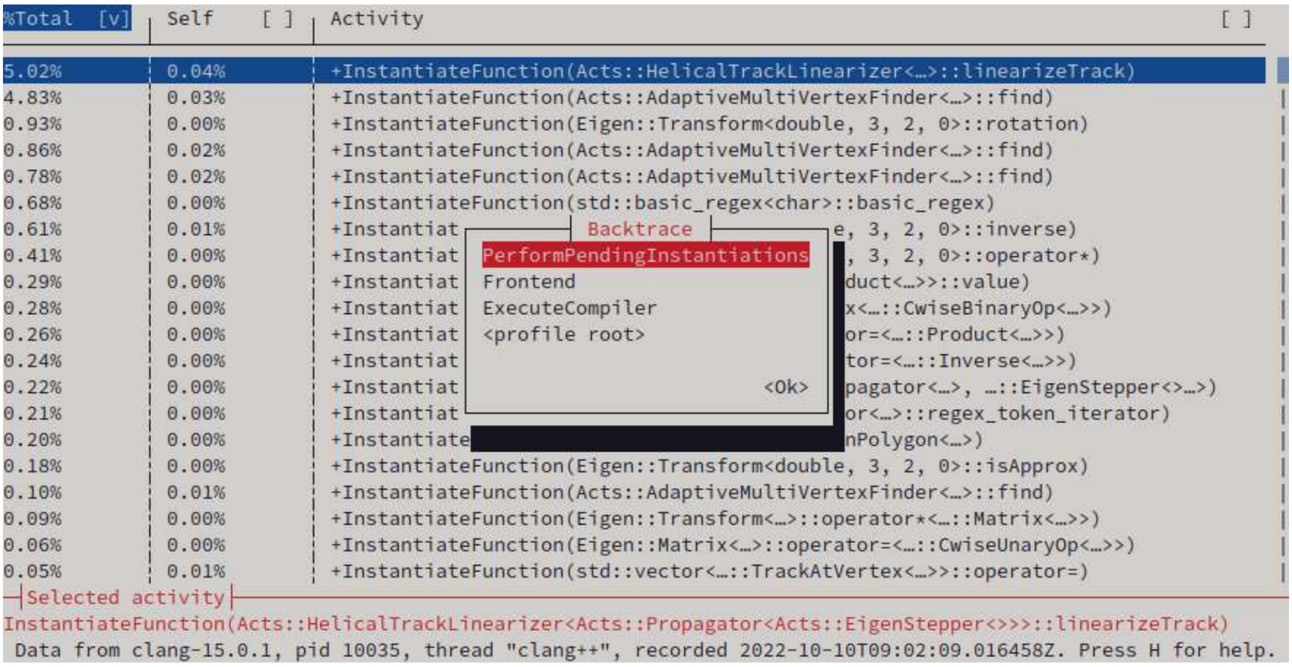
Vertexing

▾ Vertexing vs. mu

▾ IVF seeded

Fig. 1 Detailed output from automated physics validation in an Acts pull request

Another sustainability concern that was identified back in the AIDA-2020 era was the uncontrolled growth of Acts' **build requirements**, which made the project un-buildable on some systems, eventually including Github's CI nodes. While this was then resolved through ad-hoc efforts, that were further pursued during this reporting period, AIDAinnova also saw the introduction of a web dashboard [3] allowing the build CPU and memory consumption of the largest compilation units to be tracked over time, together with a local fine-grained build overhead analysis tool enabling anomalies to be easily understood [4]. We are not aware of any other HEP project expending comparable effort on this issue, making it an unexpected area of Acts innovation.



```
%Total  [v] | Self     [ ] | Activity                                                                        [ ]

5.02%       | 0.04%        | +InstantiateFunction(Acts::HelicalTrackLinearizer<…>::linearizeTrack)
4.83%       | 0.03%        | +InstantiateFunction(Acts::AdaptiveMultiVertexFinder<…>::find)
0.93%       | 0.00%        | +InstantiateFunction(Eigen::Transform<double, 3, 2, 0>::rotation)
0.86%       | 0.02%        | +InstantiateFunction(Acts::AdaptiveMultiVertexFinder<…>::find)
0.78%       | 0.02%        | +InstantiateFunction(Acts::AdaptiveMultiVertexFinder<…>::find)
0.68%       | 0.00%        | +InstantiateFunction(std::basic_regex<char>::basic_regex)
0.61%       | 0.01%        | +Instantiat ┌──────┤ Backtrace ├──────┐e, 3, 2, 0>::inverse)
0.41%       | 0.00%        | +Instantiat │PerformPendingInstantiations│, 3, 2, 0>::operator*)
0.29%       | 0.00%        | +Instantiat │Frontend                    │duct<…>>::value)
0.28%       | 0.00%        | +Instantiat │ExecuteCompiler             │x<…::CwiseBinaryOp<…>>)
0.26%       | 0.00%        | +Instantiat │<profile root>              │or=<…::Product<…>>)
0.24%       | 0.00%        | +Instantiat │                            │tor=<…::Inverse<…>>)
0.22%       | 0.00%        | +Instantiat │                        <Ok>│pagator<…>, …::EigenStepper<>…>)
0.21%       | 0.00%        | +Instantiat └────────────────────────────┘or<…>::regex_token_iterator)
0.20%       | 0.00%        | +Instantiate                              nPolygon<…>)
0.18%       | 0.00%        | +InstantiateFunction(Eigen::Transform<double, 3, 2, 0>::isApprox)
0.10%       | 0.01%        | +InstantiateFunction(Acts::AdaptiveMultiVertexFinder<…>::find)
0.09%       | 0.00%        | +InstantiateFunction(Eigen::Transform<…>::operator*<…::Matrix<…>>)
0.06%       | 0.00%        | +InstantiateFunction(Eigen::Matrix<…>::operator=<…::CwiseUnaryOp<…>>)
0.05%       | 0.01%        | +InstantiateFunction(std::vector<…::TrackAtVertex<…>>::operator=)
─┤Selected activity├──────────────────────────────────────────────────────────────────────────────
InstantiateFunction(Acts::HelicalTrackLinearizer<Acts::Propagator<Acts::EigenStepper<>>>::linearizeTrack)
 Data from clang-15.0.1, pid 10035, thread "clang++", recorded 2022-10-10T09:02:09.016458Z. Press H for help.
```

Fig. 2 Build overhead analysis tool developed as part of AIDAinnova

3. ALGORITHMS

At the start of AIDAinnova, Acts lacked support for several tracking algorithms in common use, and closing this gap with respect to the state-of-the-art has been a major focus of AIDAinnova. The most noteworthy advances in this respect are the integration of a Gaussian sum filter, a global chi-square fitter, a Hough transform, and an ambiguity resolver:

- The **Gaussian sum filter** is a generalization of the Kalman filter that uses a superposition of Gaussian probability laws instead of a single Gaussian spot model. This filter is used, e.g., when fitting electron tracks in the ATLAS experiment.
- The **Global chi-square fitter** is a non-iterative track fitter that can be used in circumstances where the set of measurements that forms a particle track are presumed to be already known and one just wants to compute optimally precise track parameters.
- The **Hough transform** is an alternative to the standard iterative HEP tracking pipeline that re-expresses particle position measurements as curves in the space of possible particle tracks, looking for crossing points between these curves. It is not a general-purpose tool, since it is sensitive to measurement noise and assumes that particle trajectories have a simple analytical formulation (typically a helix). However, it can be used in environments like high level triggers where some approximations are acceptable in exchange for faster computations.

- The **ambiguity resolver** is a clean-up step that runs after particle track candidates have been enumerated. It gives track candidates a quality score and uses that score to prune “duplicate” track candidates that share common measurements and eliminate “fake” track candidates that are unlikely to correspond to real particles.

Existing algorithms also received significant quality and performance improvements, in part linked to the increasing use of Acts within the ATLAS inner tracker (ITk) reconstruction chain, which provides valuable validation feedback. A particularly notable improvement is that the Acts track propagation system has been improved to handle navigation through new detector layer types, such as calorimeters and muon systems, which most experiments require.

Alternate implementations of existing functionality, which provide a different physics and runtime performance trade-off, were also introduced. The most significant example during this reporting period was the integration of a **k-d tree-based seed finder**, which can provide a 6x speedup over the traditional triplet search algorithm in some scenarios and is unique to Acts.

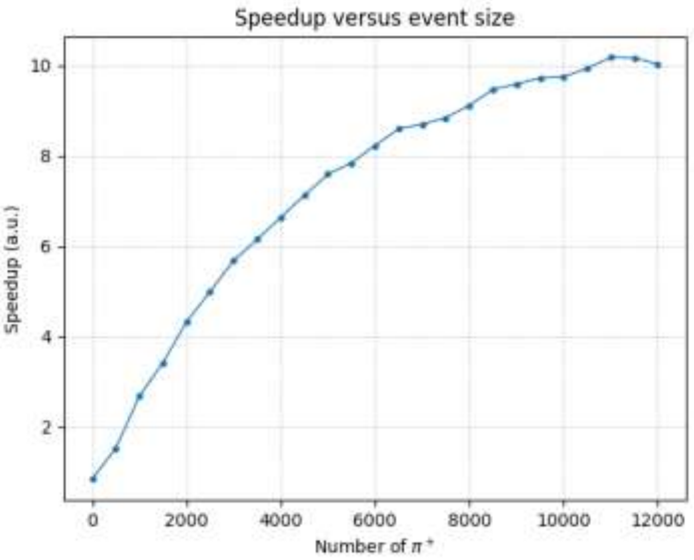


Fig. 3 Speedup achieved by the k-d tree seed finder over classical triplet search as a function of event complexity (number of simulated pions). Events with 4000 simulated pions match expected HL-LHC conditions.

Work is also ongoing to explore the applicability of machine learning approaches to track reconstruction. Two notable examples integrated to Acts during this reporting period are:

- A **graph neural network-**based track finder, developed by the Exa.TrkX project [5], has been integrated into the Acts’ codebase.
- It is now possible to **automatically tune the parameters** of Acts’ seeding and material mapping algorithms for any detector, provided that a suitable score function is provided. This work was presented at the ACAT 2022 conference [8].

4. EXPERIMENT SUPPORT

Acts geometry can be constructed from various established sources, including Geant4's GDML, ROOT's TGeo and DD4hep **detector descriptions**. However, doing so requires complementing the source geometry information with additional metadata describing which surfaces contain sensitive detectors, and how these surfaces are grouped into higher-level layered and binned structures. Before AIDAinnova, this was a rather manual and cumbersome process, involving the use of special-purpose classes, so it has been a focus of this project to improve this central part of the new user onboarding experience.

The Acts team has worked with the DD4hep team to improve the DDRec::Surface class, so that it can be used to provide all required information at the level of the DD4hep geometry description, without needing to resort to Acts-specific classes. Similarly, Acts' Geant4 geometry support has been improved to the point that simple GDML geometries may now be imported into Acts tracking geometries without needing extra user-side work.

Another issue that has recently begun receiving attention is that of importing and exporting measurement data in experiment-specific data formats. So far, Acts has mostly focused on defining the internal event data model used by its track reconstruction algorithms, but work has now started to enable interoperation with **event data models**, including ATLAS xAOD and the AIDAinnova-developed EDM4hep. A public interface to the output of Acts tracking on the experiment framework's side is also in the process of being introduced.

Finally, the Acts Spack package has been kept up to date, in the same spirit of keeping Acts a good citizen of the Key4hep stack developed as part of AIDAinnova. Progress has also been made on making Acts usable inside of the Marlin ILC track reconstruction framework, while it was also experimentally applied to the ALICE upgrade detector.

5. GPU COMPUTING R&D

Graphics processing units (GPUs) provide about an order of magnitude more main memory throughput and floating-point processing power when compared to CPU platforms of comparable price or power consumption, at the cost of only being accessible over the comparatively slow PCI-express interconnect and using a more complex and less general-purpose programming model. For many important classes of computations, this is a good trade-off, hence a growing number of high-performance computing (HPC) systems are mainly GPU-based.

From a HEP perspective, this evolution can be seen as an opportunity and a threat. It is an opportunity, if it allows bridging some of the gap between projected HL-LHC and FCC computing power needs and hardware budgets. But it also threatens to shrink the pool of available computing resources over time if HEP software fails to adapt to this emerging hardware platform. Thus, many HEP software projects are actively exploring the viability of GPU computing for the work that they are doing, and Acts is no exception.

Early prototyping revealed that several core design decisions of Acts are not friendly to GPU computing, most prominently its inheritance-based geometry infrastructure. This precluded in-tree experimentation, so separate R&D projects were started to explore what a GPU-friendly Acts would look like, and how competitive it could be with the existing codebase:

- **vecmem** abstracts over the memory management primitives of multiple GPU APIs to provide a common framework for memory allocation and CPU/GPU data transfers.
- **algebra-plugins** abstracts over multiple algebra libraries to allow exploration of options besides Eigen, the library currently used in Acts, which has proven to have suboptimal runtime performance on CPU, problematic compile-time overhead, and low-quality GPU backends.
- **covfie** provides general-purpose vector field management facilities, and will be used specifically for magnetic field maps.
- **detray** investigates what a GPU-friendly detector geometry and track propagation infrastructure should look like.
- **traccc** puts all the previous projects together and complements them with tracking algorithms, with the aim of assembling a fully functional tracking chain.

During this AIDAinnova reporting period, several important developments were carried out within this experimental track reconstruction chain:

- vecmem received many correctness fixes and performance improvements, culminating in the recent introduction of support for **asynchronous data transfers**.
- algebra-plugins was used as a testing ground to evaluate several Eigen contenders, leading to the conclusion that **Fastor** [6] would be the most promising candidate on CPU.
- covfie was fully developed during AIDAinnova.
- detray matured to the point where it can be used in traccc for Kalman Filtering, and a new geometry implementation based on its design is in the process of being integrated back into Acts.
- traccc is close to implementing a complete tracking chain using both the SYCL and CUDA GPU APIs, with the main remaining work-in-progress block being Kalman filtering. Fair runtime performance comparisons between CPU and GPU algorithms are now possible for existing algorithms [7]. These algorithmic developments were almost fully carried out during the AIDAinnova timeline, taking advantage of support from Acts contributors outside AIDAinnova.

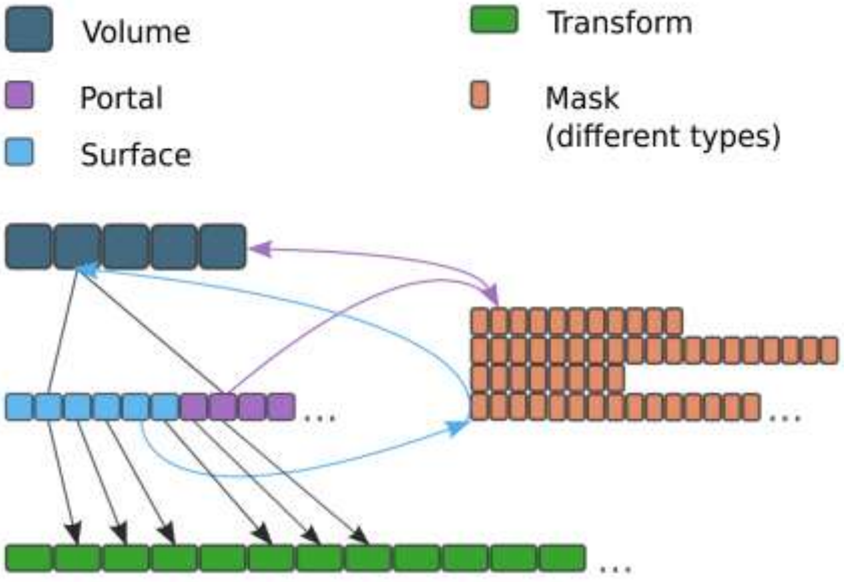


Fig. 4 GPU-friendly detector geometry data structure, developed by the detray R&D project and now being integrated into the core Acts library [10].

Another byproduct of traccc is that it could be used to study the performance characteristics of the various memory management schemes provided by the SYCL GPU computing API on a realistic tracking use case (measurement clustering, a form of sparse connected-component labelling). This work was presented at ACAT 2021 [9].

Finally, owing to the growing use of GPU computing as part of this R&D, as well as machine learning algorithm R&D, the continuous integration infrastructure of Acts was extended to enable testing projects on nodes with GPU hardware at CERN.

6. REFERENCES

JOURNAL ARTICLES:

[1] Ai, X. et al (2021) A Common Tracking Software Project, submitted to *Computing and Software for Big Science*, current preprint available at https://arxiv.org/pdf/2106.13593.pdf

[6] Poya, R. et al (2017) A high performance data parallel tensor contraction framework: Application to coupled electro-mechanics, *Computer Physics Communications*, 216, 35-52.

CONFERENCE PAPERS

[9] Joube, S. et al (accepted and pending publication) *Comparing SyCL data transfer strategies for tracking use cases.* In: *ACAT 2021 proceeding*, Virtual and IBS Science Culture Center, Daejeon, South Korea, from 2021-11-29 to 2021-12-03.

WORLD WIDE WEB DOCUMENTS

[2] Allaire, C. et al (2022) *OpenDataDetector* [online]. Available from: https://doi.org/10.5281/zenodo.4674401 [Accessed 20 January 2023]

[3] Gessinger, P. (2022) *Acts build metrics* [online]. Available from: https://acts-project.github.io/metrics/ [Accessed 20 January 2023]

[4] Grasland, H. (2022) *crofiler: Easier C++ build profiling* [online]. Available from: https://github.com/HadrienG2/crofiler [Accessed 20 January 2023]

[5] Spiropulu, M. et al (2022) *HEP advanced tracking algorithms at the exascale (Project Exa.TrkX)* [online]. Available from : https://exatrkx.github.io/ [Accessed 23 January 2023]

[7] Rita De Almeida, G. M. (2022) *Plot showcase: GPU vs CPU throughput results* [online]

[8] Allaire, C. et al (2022) *Auto-tuning capabilities of the ACTS track reconstruction suite* [online]. Available from: https://indico.cern.ch/event/1106990/contributions/4998102/ [Accessed 23 January 2023]

[10] Salzburger, A. et al (2022) *Navigation, field integration and track parameter transport through detectors using GPUs and CPUs within the ACTS R&D project* [online]. Available from: https://indico.cern.ch/event/1106990/contributions/4998085/ [Accessed 2 February 2023]

ANNEX: GLOSSARY

Acronym	**Definition**
CI	Continuous Integration
CPU	Central Processing Unit
FCC	Future Circular Collider
GPU	Graphics Processing Unit
HEP	High Energy Physics
HL-LHC	High-Luminosity LHC
HPC	High Performance Computing
LHC	Large Hadron Collider