



Application of ACTS to CEE TPC detector

-- `CeeActs` prototype

Zequn Sun, Aiqiang Guo
IMP, CAS

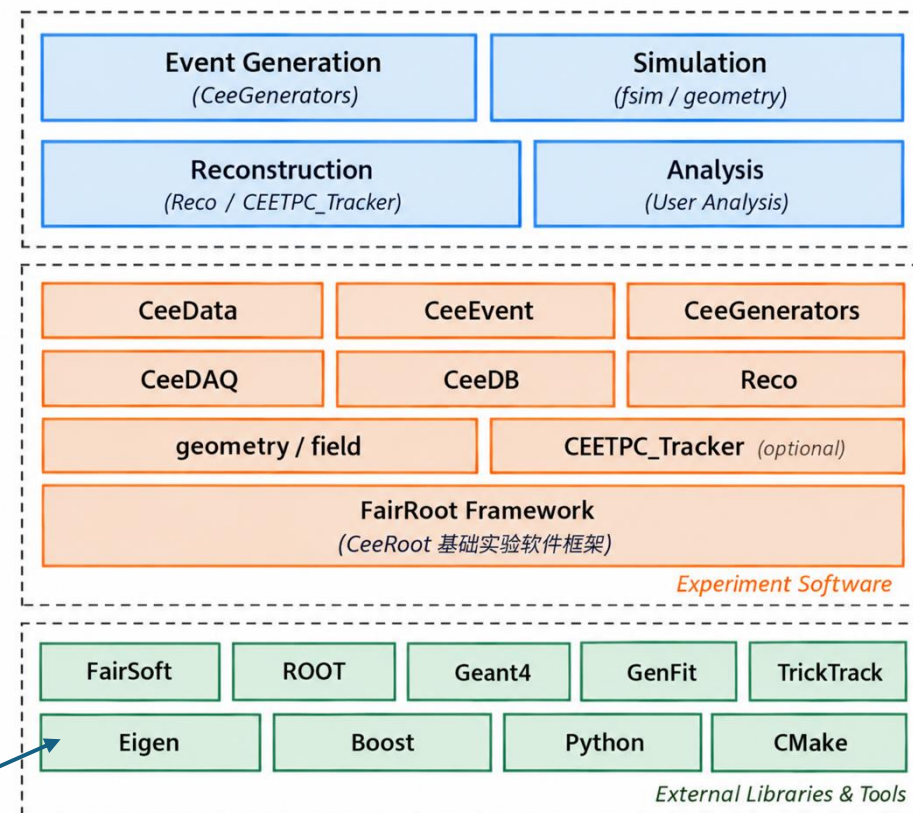
High-Energy Nuclear Physics and Tracking at HIRFL-CSR Workshop, Huizhou, China
2026.10.15

Outline

- Introduction
- Benchmark
- Integration of ACTS with CeeRoot
 - Geometry & EDM
 - Algorithm
- Preliminary performance of CPU/GPU-based tracking with ACTS
- Summary

CeeRoot

- **Simulation and reconstruction and analysis software for CEE : based on FairSoft and FairRoot**
- **Tracking Software development**
 - Hardware : multi-cores CPUs and accelerators (GPU, FPGA)
 - Software : Machine learning
- **This work extends CeeRoot to support ACTS CPU/GPU-based tracking**



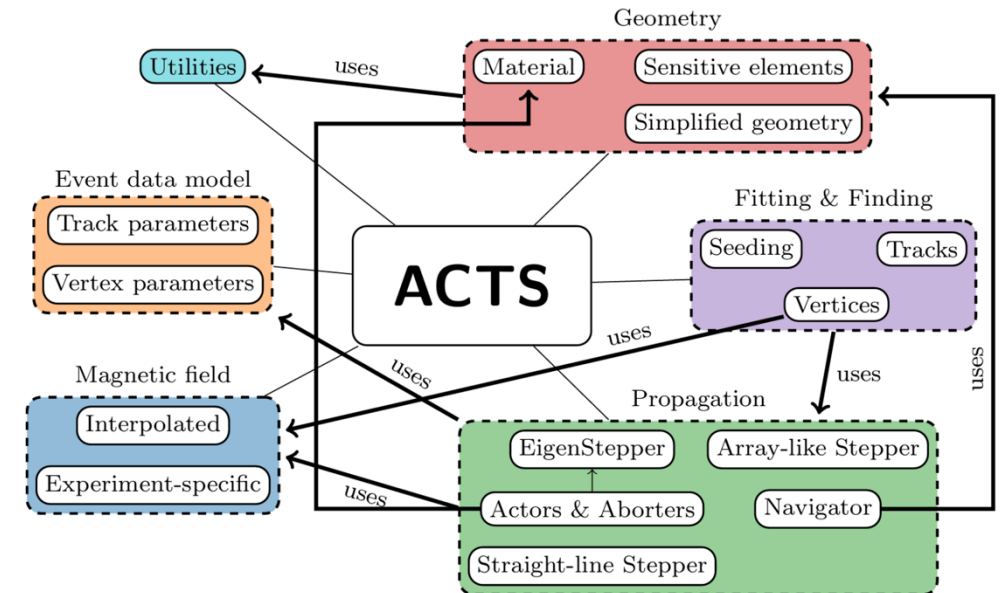
ACTS : A Common Tracking Software

➤ An open source and experiment independent tracking toolkit for HEP experiments [1,2]

- Support CPU multithreading
- Modular design
- Good scalability and support ML
- Supports various detector surface descriptions

➤ Several R&D projects within the ACTS ecosystem

- **traccc**: GPU tracking pipeline [3]
- **ML-based alg**: GNN based tracking, Seed Filter, Ambiguity Resolution and so on [2]



[1] Git : <https://github.com/acts-project/acts>

[2] Doc : <https://acts-project.github.io>

[3] <https://doi.org/10.1051/epjconf/202533701187>

`ACTS::traccc` : GPU tracking library in ACTS

- A fully GPU-based tracking pipeline for hep-ex within ACTS [1]
 - Compatible with NVIDIA, AMD and Intel
 - Algorithm and supporting libraries (detray, vecmem and covfie) are developed for GPU

OS & compilers:

Please note that due to the complexity of this software and its build system, it may be somewhat fragile in the face of compiler version changes. The following are general guidelines for getting *traccc* to compile:

- The C++ compiler must support C++20

In addition, the following requirements hold when CUDA is enabled:

- The CUDA Toolkit version must be greater than major version 12
- Ensure that the CUDA host compiler supports C++20 and is compatible with the `nvcc` compiler driver

The following table lists currently combinations of builds, compilers, and toolchains that are currently known to work (last updated 2022/01/24):

Build	OS	gcc	CUDA	comment
CUDA	Ubuntu 24.04	13.3.0	12.6	runs on CI

Dependencies

- [Boost](#): program_options
- [CMake](#)
- (Optional) [ROOT](#): RIO, Hist, Tree

Features

Category	Algorithms	CPU	CUDA	SYCL	Alpaka
Clusterization	CCL / FastSv / etc.	✓	✓	✓	✓
	Measurement creation	✓	✓	✓	✓
Seeding	Spacepoint formation	✓	✓	✓	✓
	Spacepoint binning	✓	✓	✓	✓
	Seed finding	✓	✓	✓	✓
	Track param estimation	✓	✓	✓	✓
Track finding	Combinatorial KF	✓	✓	✓	✓
Ambiguity resolution	Greedy resolver	✓	●	●	●
Track fitting	KF	✓	✓	●	✓

✓: exists, ●: work started, ○: work not started yet

[1] <https://indico.cern.ch/event/1499357/contributions/6621951/>

Benchmark

➤ Hardware :

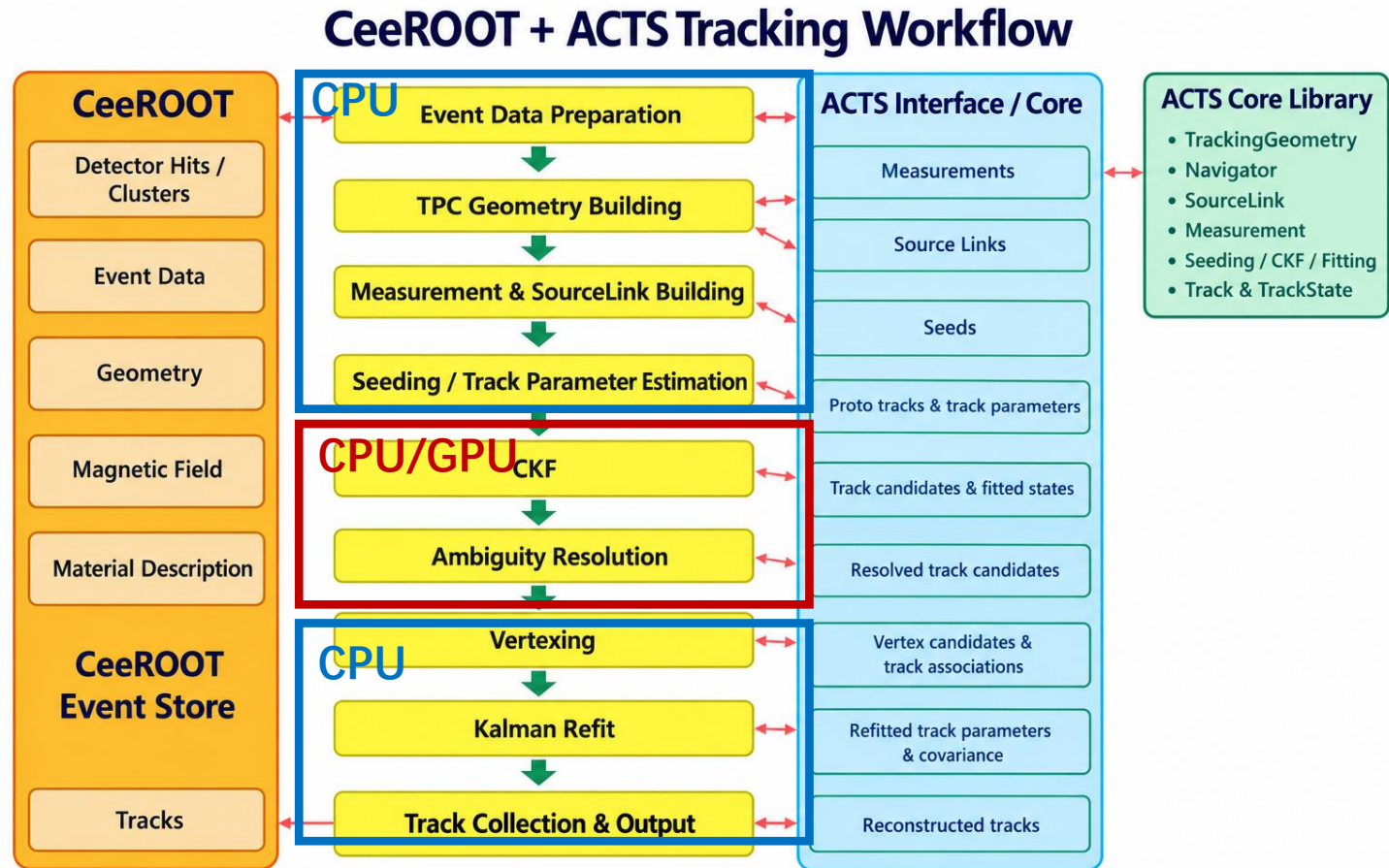
- CPU : Intel Ultra 9 275HX
(8P+16E, 3.07 GHz) + 1 thread
- GPU : NViDIA RTX 5070Ti
Laptop 12G (CUDA Version 13.4)

➤ Software

- WSL Ubuntu 24.04
- ACTS : v48.0.1
- traccc: v1.6.0

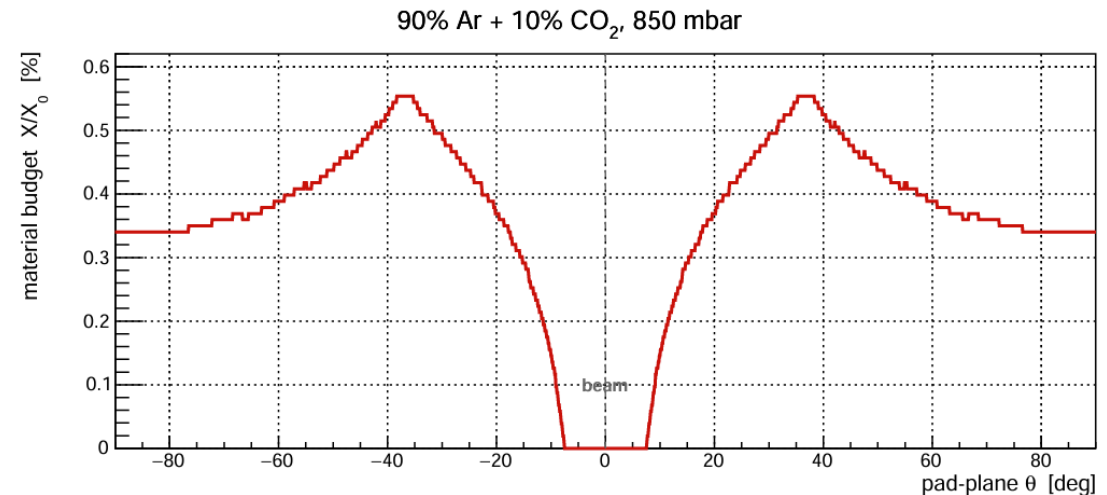
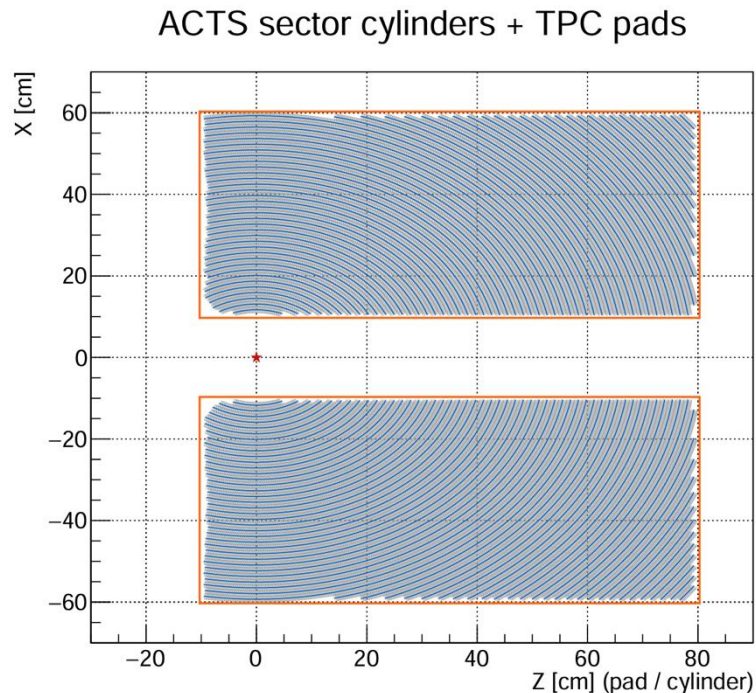
➤ Datasets

- 50 events, 100 protons per event, $B = 0.5$ T
- $0.0 < P_t < 0.5$ GeV/c



Application of ACTS to CeeRoot : TPC geometry

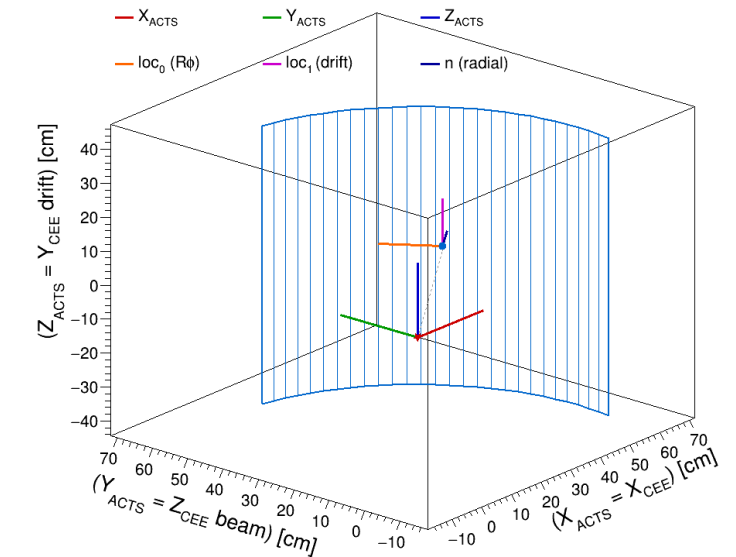
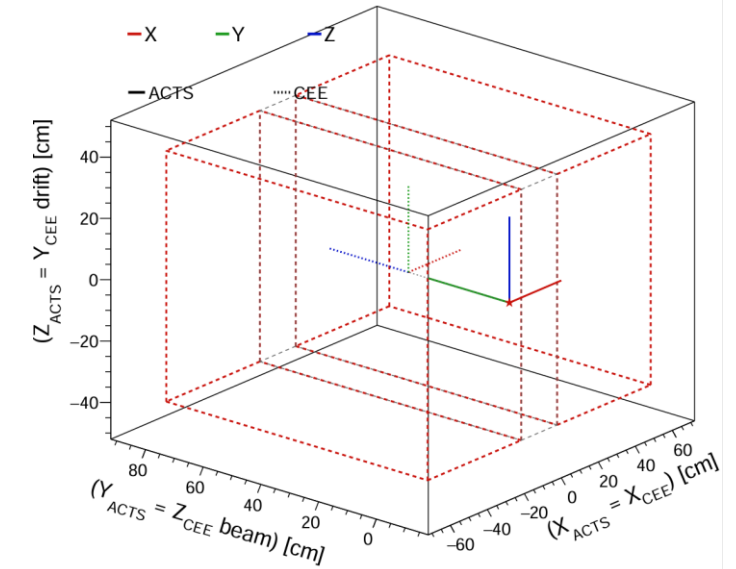
- ACTS Gen3 `TrackingGeometry` [1] : sensitive surface information (shape + location) + simplified material
- TPC pad information : `PadDigitizationOriginInfo\_NOGAP\_v1.2.txt`
- Pad-row at each TPC sector -> a fake `Acts::CylinderSurface`
- Material : 90% Ar + 10% CO₂, 850 mbar, width 1.4 cm



[1] <https://indico.cern.ch/event/1501989/contributions/6355820/>

Application of ACTS to CeeRoot : EDM

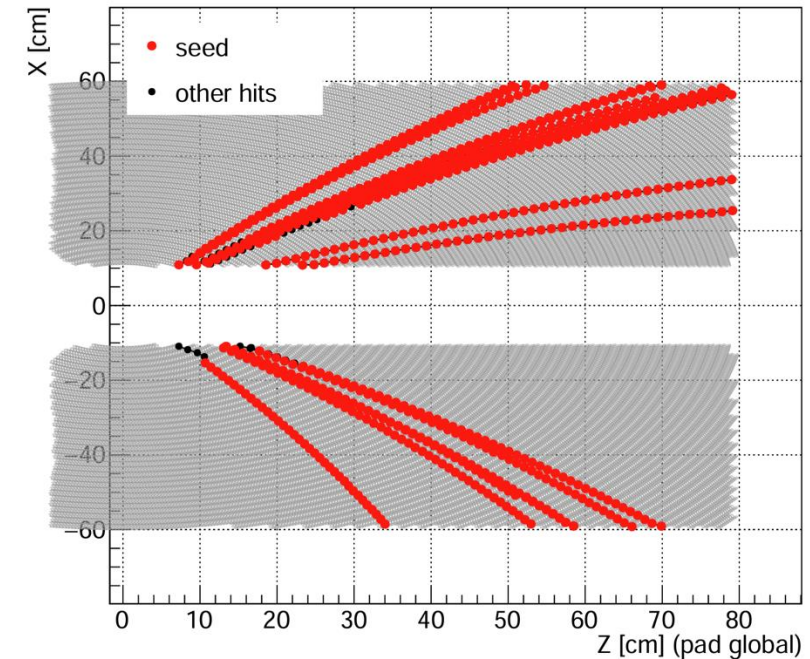
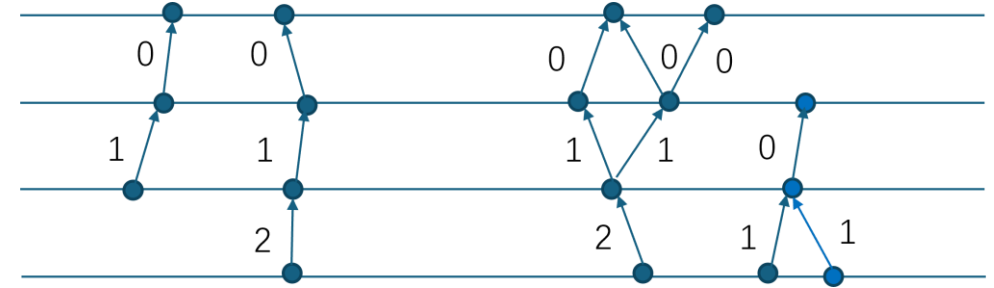
- Clustering (hit -> measurement) is implemented in CeeRoot
 - `CeeTpcHit` -> `Acts::Measurement` and `Acts::SourceLink`
 - Transformation
 1. Global frame: CEE TPC global vs ACTS global
$$(X, Y, Z)_{CEE} (cm) = (X, Z, Y - 35)_{ACTS} \times 10 (mm)$$
 2. ACTS global frame -> Surface local frame ($R\phi, z$)
$$\phi = \text{atan2}(Y_{ACTS}, X_{ACTS})$$
 3. Error : error propagation
 4. `Acts::SourceLink` : `Acts::Measurement` + `Acts::Surface::Identifier` + HitIndex



Application of ACTS to CeeRoot : Seeding

➤ Seeding : Cellular Automata

1. Find doublet : $|\Delta\phi| < 0.3, |\Delta\rho| < 2.08 \text{ cm}, |\Delta\theta| < 0.3, \Delta x < 2.51 \text{ cm}, \Delta z < 2.13 \text{ cm}$
2. Evolution : $\theta_{xz} < 0.5, \theta_{Ry} < 0.4$
 - Loop all cells to find compatible neighboring cells in the next inner layer
 - Repeat until no further extension is possible
3. The best and longest one with root cell as seed -> estimate the sign of this track
 - Root cell : doublets formed by hit without neighbor hit in the outer layer



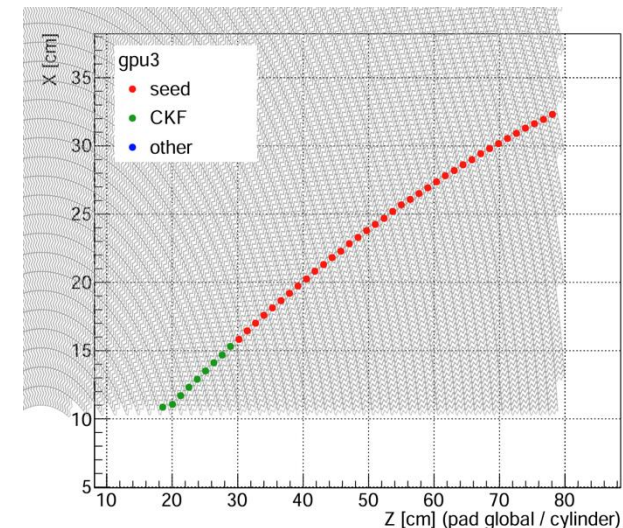
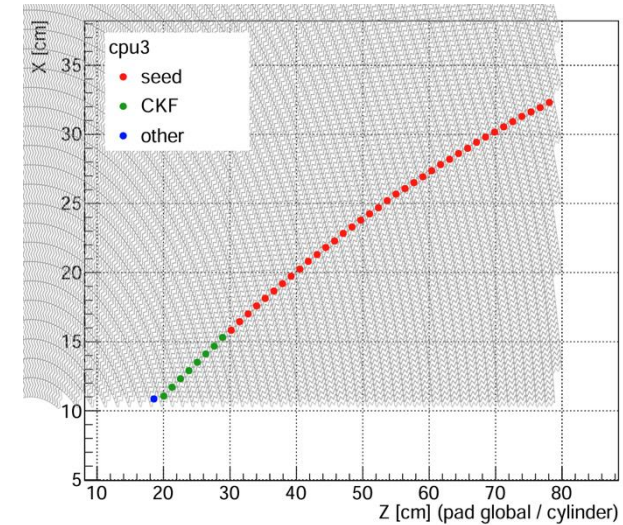
More in [Ze Song`s](#) and [Aiqiang`s](#) talks

Application of ACTS to CeeRoot : Finding

➤ Finding : Combinational Kalman Filter

- Chose at most 1 hit on this layer with minimum χ^2_{local} (KF predication vs measurement)
- Inward : look for hit from outer to inner layer
- Sign of charge : estimated with seed
- Particle hypothesis: proton, without material effect
- End :
 1. CPU mode : end : 3 holes ($\chi^2_{local} > 36$) or outliers ($9 < \chi^2_{local} < 36$)
 2. GPU mode : end : 3 holes ($\chi^2_{local} > 36$)

➤ Note : CPU and GPU have independent CKF code in ACTS



Ambiguity Resolution and Vertexing

- **Ambiguity resolution : each used hit on at most one track**
 - Rank remaining tracks with shared hits: more hits -> lower χ^2/ndf
- **Vertexing : `Acts::IterativeVertexFinder` + `Acts::FullBilloir` (preliminary)**
 - Good TPC track -> propagate to Zcee(Yacts)=0 -> VertexSeed
 - `Acts::FullBilloir` : calculate weight factor for each unused track with the same vertex position
 - `Acts::IterativeVertexFinder` : estimate vertex position with unused tracks -> FullBilloir -> estimate vertex position with lower weight tracks -> FullBilloir -> ...
- **Kalman fitter : `Acts::DirectNavigator`**
 - Vertex constrained
 - Inner -> outer
 - Sign of charge : same as CKF

Preliminary performance : timing

- CKF : ~5x
- AR : ~2x
- GPU memory :
~400 MB/event

----- CeeActs tracking timing summary -----

Geometry	0.040 s	(0.1 %)	mean 0.0402 s	n=1
Projection	0.042 s	(0.1 %)	mean 0.0008 s	n=50
Seeding	0.324 s	(0.9 %)	mean 0.0065 s	n=50
Tracking	23.079 s	(61.6 %)	mean 0.4616 s	n=50
CeeActsTpcAmbiguityTask	0.117 s	(0.3 %)	mean 0.0023 s	n=50
Vertex	13.877 s	(37.0 %)	mean 0.2775 s	n=50
TOTAL	37.479 s			

CPU

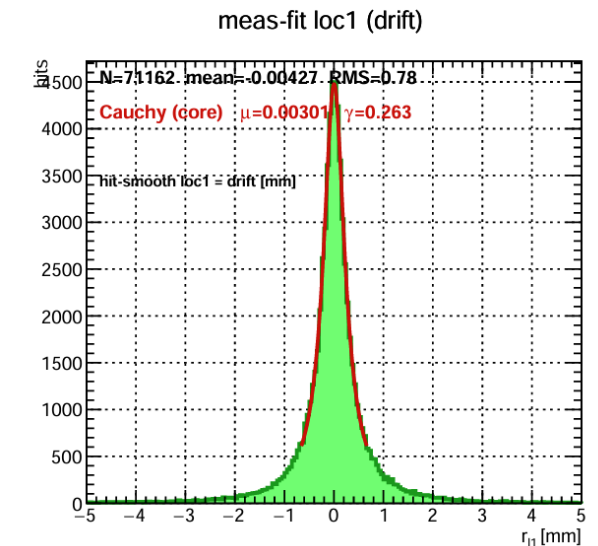
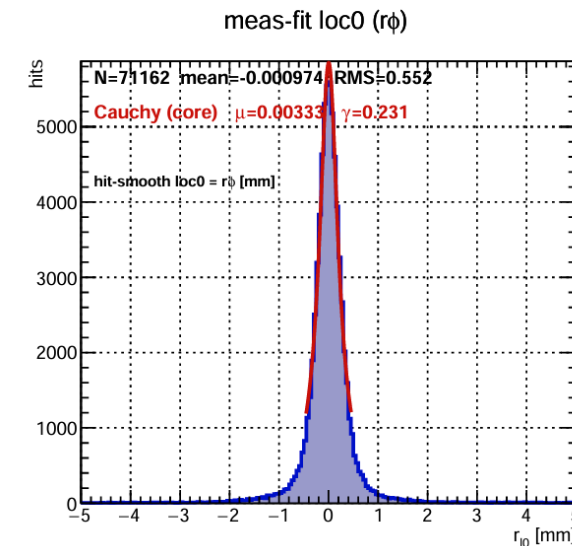
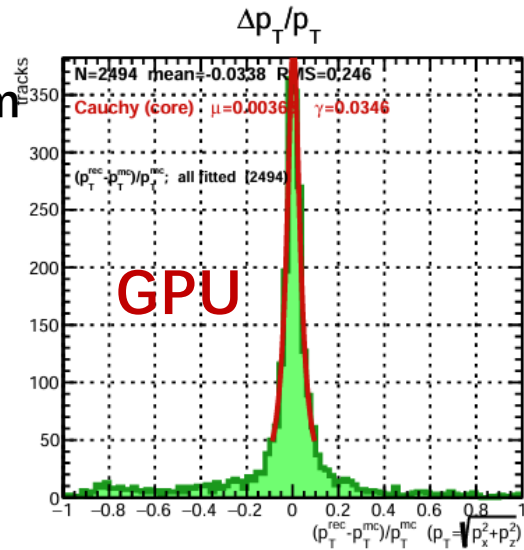
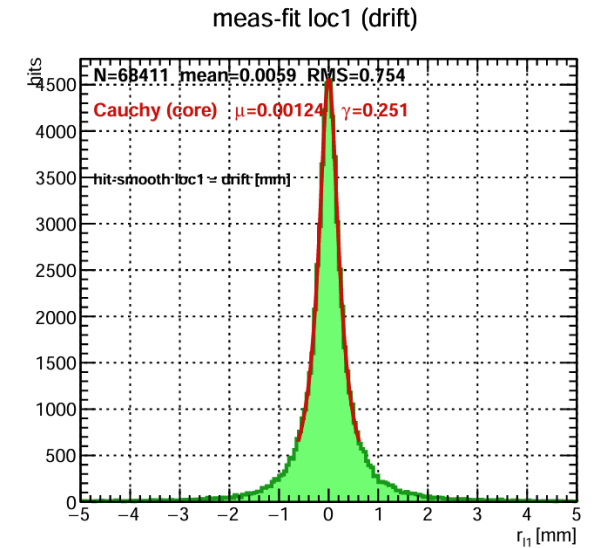
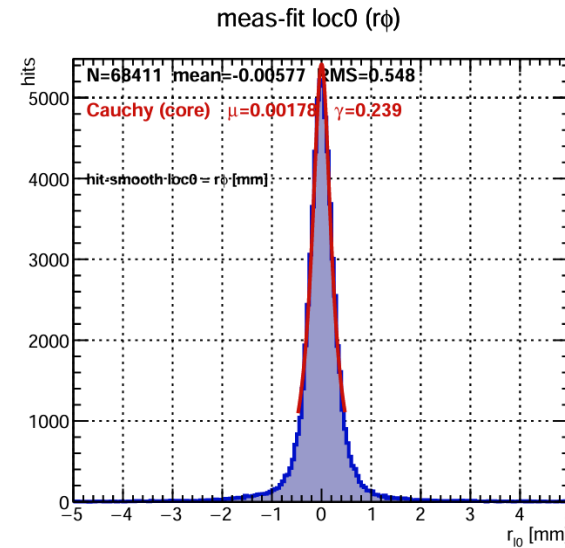
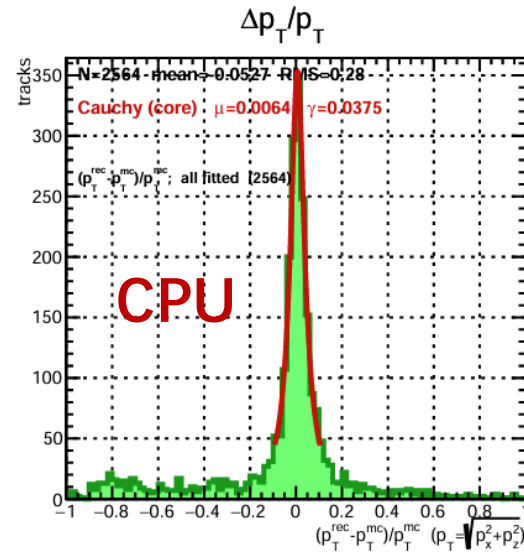
----- CeeActs tracking timing summary -----

Geometry	0.038 s	(0.2 %)	mean 0.0384 s	n=1
Projection	0.041 s	(0.2 %)	mean 0.0008 s	n=50
Seeding	0.317 s	(1.7 %)	mean 0.0063 s	n=50
Tracking	4.483 s	(24.0 %)	mean 0.0897 s	n=50
CeeActsTpcAmbiguityTask	0.053 s	(0.3 %)	mean 0.0011 s	n=50
Vertex	13.729 s	(73.6 %)	mean 0.2746 s	n=50
TOTAL	18.662 s			

GPU

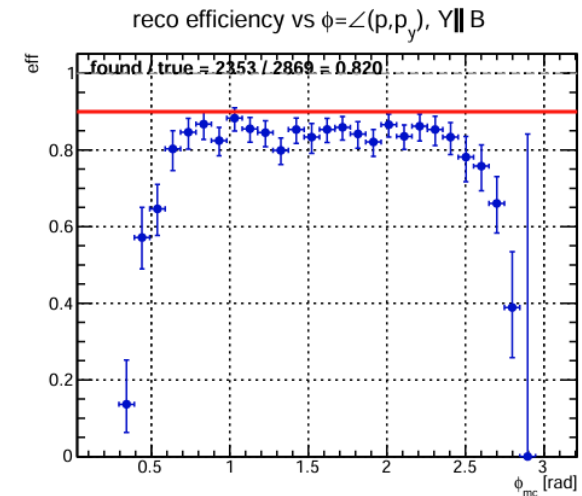
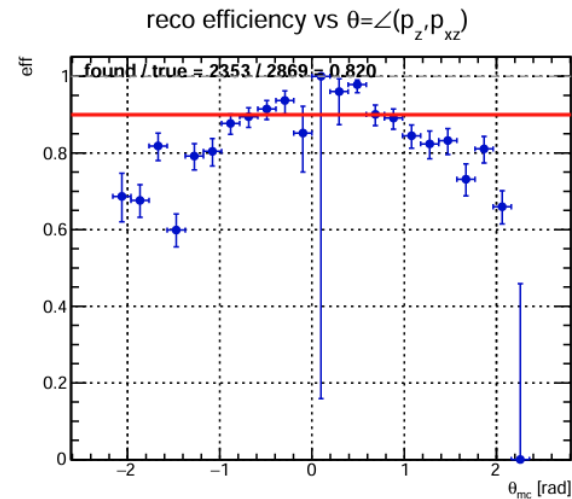
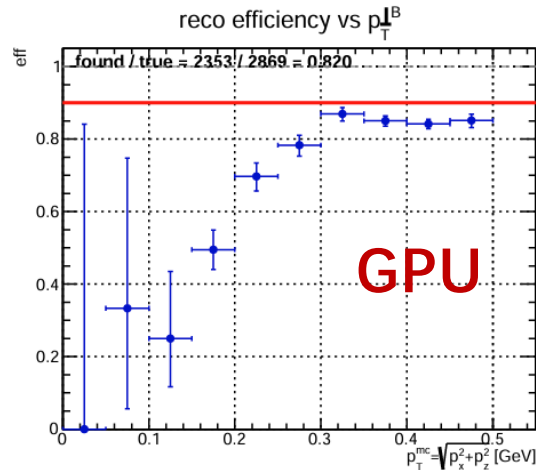
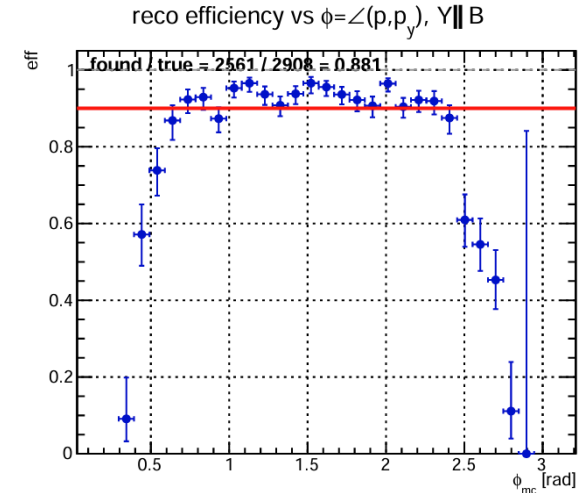
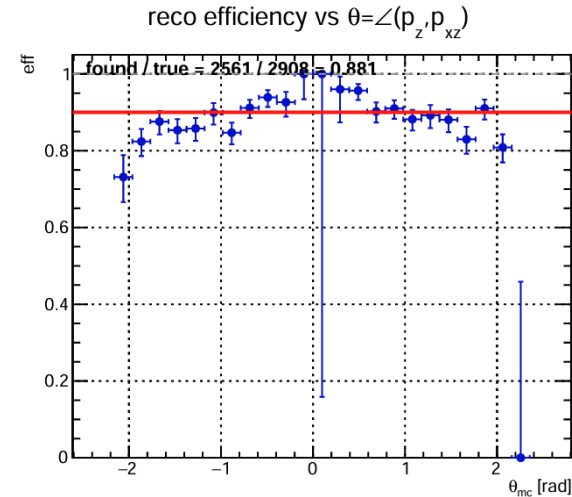
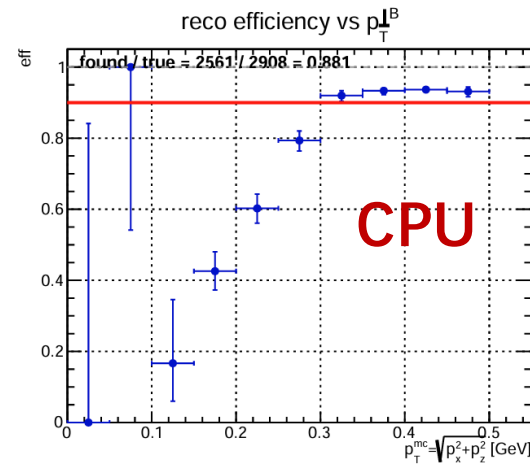
Preliminary performance : resolution

- Good agreement between these two workflows
- Pt residual : $\sigma \sim 4\%$ for $0.0 < Pt < 0.5$ GeV/c
- Hit residual : $\sigma \sim 0.3$ mm



Preliminary performance : efficiency

- CPU : ~ 88 % (> 90 % for tracks with $P_t > 0.3$ GeV)
- GPU : ~ 82 %
- Optimization is necessary
 - CPU and GPU have independent CKF code in ACTS
 - Need to check the intermediate results at each CKF step



Summary

➤ ACTS has been integrated into CeeRoot **preliminarily**

(not yet released)

✓ CPU-based tracking

✓ GPU-based tracking

➤ ToDo :

- **Optimization is necessary -> release ASAP**

- ❑ CKF strategy

- ❑ GPU/CPU resource scheduling optimization

- ❑ Material effect

- ❑ GPU-based vertexing implementation

- GNN-based tracking

```
CeeActs/
├─ CMakeLists.txt      → one library libCeeActs (gcc 13 for ACTS, gcc 8 for ROOT)
├─ Configs/
│   └─ CeeActsLinkDef.h → ROOT dictionary: FairTasks only
├─ Core/
│   └─ Geometry/      → Gen3 TrackingGeometry
│       ├── CeeActsTpcGeometry.{h,cxx}    2 CylinderSurfaces / pad row, TPC_L | TPC_BEAM | TPC_R
│       ├── CeeActsTpcGeometryRoot.cxx    fake TGeo for display
│       ├── CeeActsTpcGeometryId.h        unpack GeometryIdentifier
│       └─ CeeActsMagneticField.{h,cxx}    FairField → ACTS B, By → -Bz
│   └─ EventData/      → CeeTpcHit → measurement
│       ├── CeeActsTpcProjector.{h,cxx}    lab hit → (Rφ, drift) on the cylinder
│       ├── CeeActsTpcCalibrator.{h,cxx}    diagonal cov: 0.81xGetDx, 2.04xGetDy
│       └─ CeeActsTpcSourceLink.h          geometryId + hitIndex + measIndex
│   └─ Algorithm/
│       ├── CeeActsTpcCaSeeding.{h,cxx}    CA seed, doublet → triplet, default 5 hits
│       ├── CeeActsTpcCkfFitter.{h,cxx}    ACTS CKF, CPU, inward from the outer seed hit
│       ├── CeeActsTpcKalmanFitter.{h,cxx} Kalman on a fixed surface list
│       ├── CeeActsTpcVertexFitter.{h,cxx} Billoir vertex
│       └─ CeeActsTpcPerformance.{h,cxx}   purity, hit efficiency, residuals
│   └─ Traccc/
│       ├── CeeActsTpcDetray.{h,cxx}       Gen3 geometry → detray
│       └─ CeeActsTpcTracccFitter.{h,cxx}   CUDA CKF → AR → KF; flip p and q at export
└─ Task/      → FairRunAna order
    ├── CeeActsTpcGeometryTask
    ├── CeeActsTpcProjectionTask
    ├── CeeActsTpcSeedingTask
    ├── CeeActsTpcTrackingTask      fitMode 1 = CPU CKF, 3 = traccc
    ├── CeeActsTpcAmbiguityTask
    └─ CeeActsTpcVertexTask        Billoir + CPU Kalman refit
```

Backup

