Application of ACTS to CEE TPC detector -- `CeeActs` prototype

Zequn Sun[1], Aiqiang Guo[1], Ze Song[2]

1. IMP, CAS

2. CCNU

Mini Workshop on Tracking at HIRFL-CSR

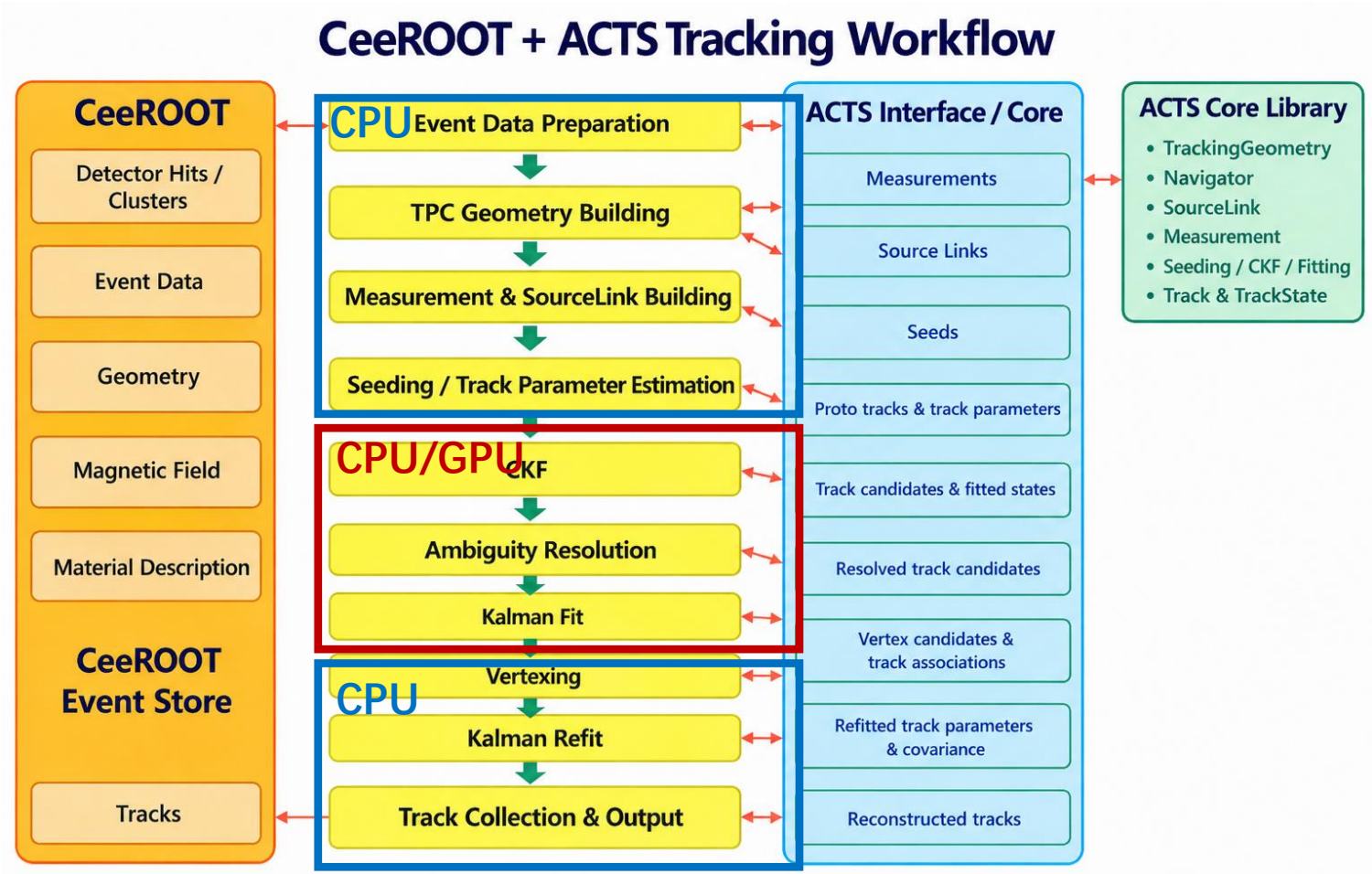
2026.11.21

Outline

- Benchmark
- Update
- ACTS library : CPU vs GPU
- Performance of CPU/GPU-based tracking with ACTS
- Performance with single-track events
- Summary

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 : v48.4.0
  - traccc: v1.7.0
- Datasets
  - 1000 events, 100 protons per event
  - By = 0.5 T, 0.0 < Pt < 0.5 GeV/c



Update

- ACTS version : v48.0.1 -> v48.4.0, fix bugs in Vertexing [1,2]
- CeeActs
  - Improve program performance
    1. Hits are indexed by surface once per event instead of calling `findSurface` in CKF
    2. `Stepper` in CKF : `HelixStepper` (low material and uniform B)
    3. `Stepper` in KalmanFit : `EigenStepper` : Runge-Kutta + Magnetic map + Material effect
    4. Tune some parameters used in CKF and KF
  - GPU-based CKF: end finding until 5 holes
    1. $\chi^2_{local} < 9$: as a good hit -> Kalman update
    2. $9 < \chi^2_{local} < 36$: errors of loc0/1 are multiplied by $\chi^2_{local}/8.5$ -> Kalman update with lower gain matrix
    3. $\chi^2_{local} > 36$: hole

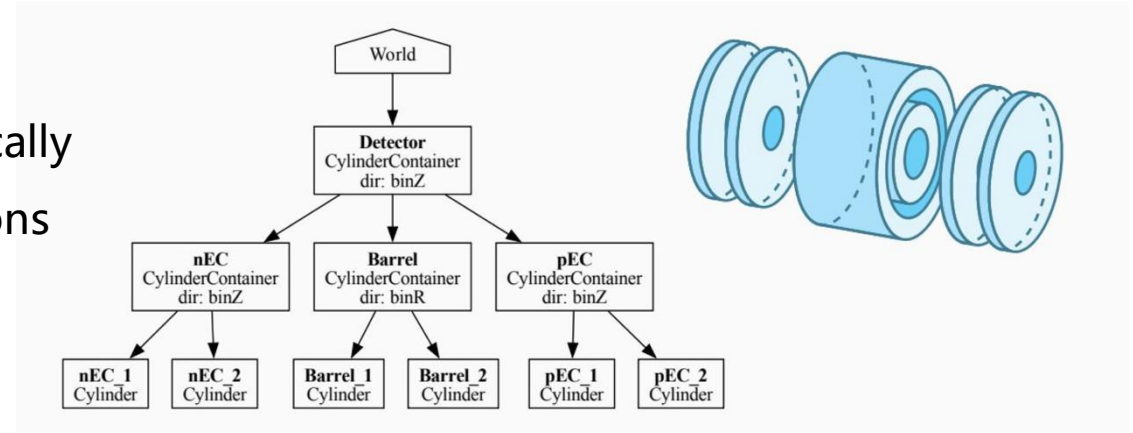
🐛 Bug Fixes

- Set particle hypothesis on fitter output tracks (#6352) by **@benjaminhuth**
- Correct the cone and point free/bound Jacobians (#6366) by **@Zequn-SUN**

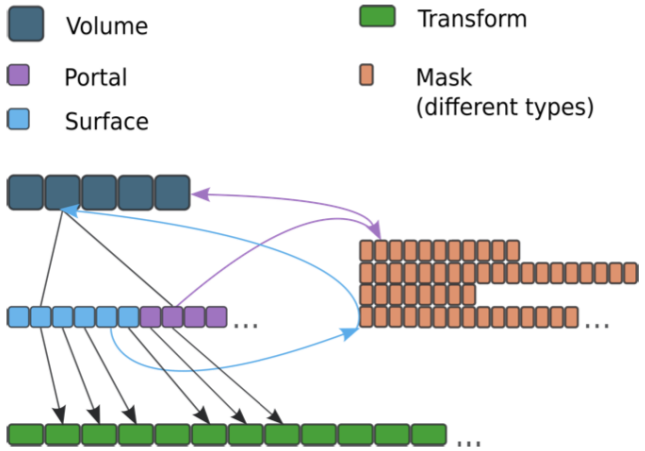
[1] https://github.com/acts-project/acts/releases/tag/v48.4.0
[2] https://github.com/acts-project/acts/pull/6366

ACTS library : CPU vs GPU, geometry

- ACTS::Gen3 geometry for CPU (**pointer-based**) [1]
  - Tree structure : volumes and surfaces built hierarchically
  - Portal link : connected via raw ptr and virtual functions
  - Not GPU-friendly
    1. Complex object hierarchy
    2. Fragmented memory layouts



- ACTS::detray geometry for GPU (**index-based**) [2]
  - Geometry are stored in large, contiguous arrays
  - Use `Masks` to control surface’ s shape
  - Index link rather than ptr



[1] Paul’s and Andi’s talk
[2] https://iopscience.iop.org/article/10.1088/1742-6596/2438/1/012026

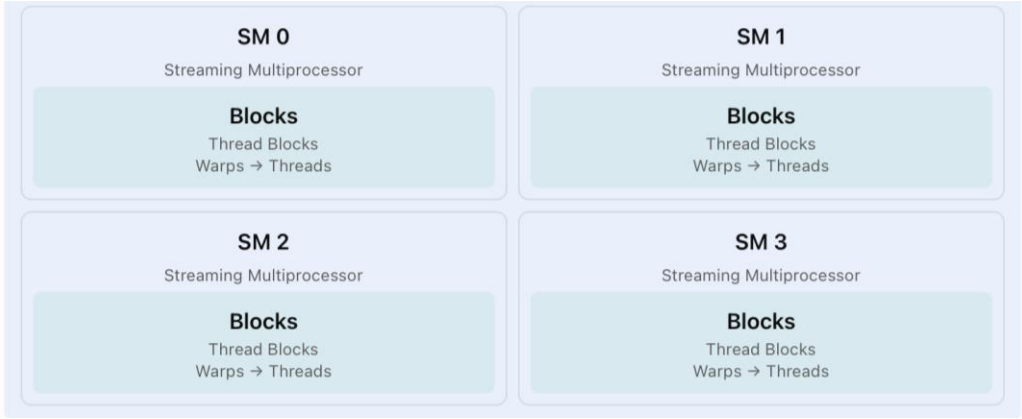
ACTS library : CPU vs GPU, algorithm

CKF	CPU	GPU
Finding	One seed at a time	All seeds simultaneously
Precision	FP64	FP32
χ^2	Good/outlier/hole	Good/hole
Smoother	No	Modified Bryson–Frazier, outward
Covariance	$C_f = (I - KH)\,C$	$C_f = (I - KH)\,C\,(I - KH)^T + KVK^T$

- Prediction covariance for GPU: Joseph form -> better numerical stability

KF	CPU	GPU
Smoother	Modified Bryson–Frazier, inward	Two-filter (both inward and outward)
χ^2	Filtered-χ^2	Backward-χ^2
Iteration	No	Yes

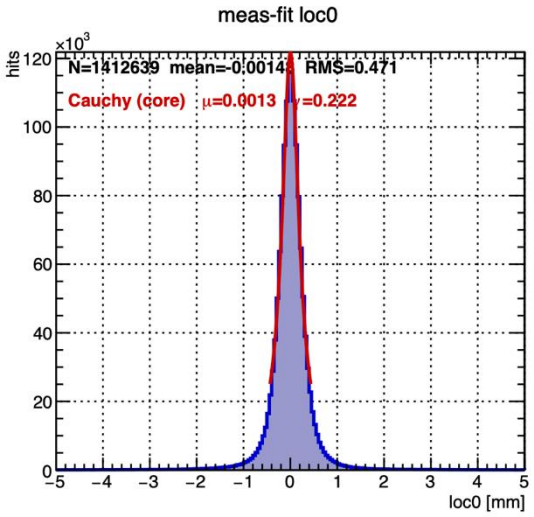
- NVIDIA 5070Ti Laptop : 5888 CUDA cores
  - 46 Streaming Multiprocessor
  - 32 Blocks with 48 Warp / SM
  - 32 Threads / Warp

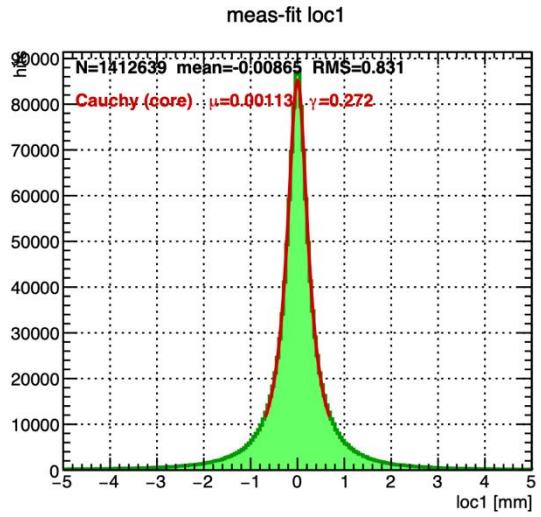


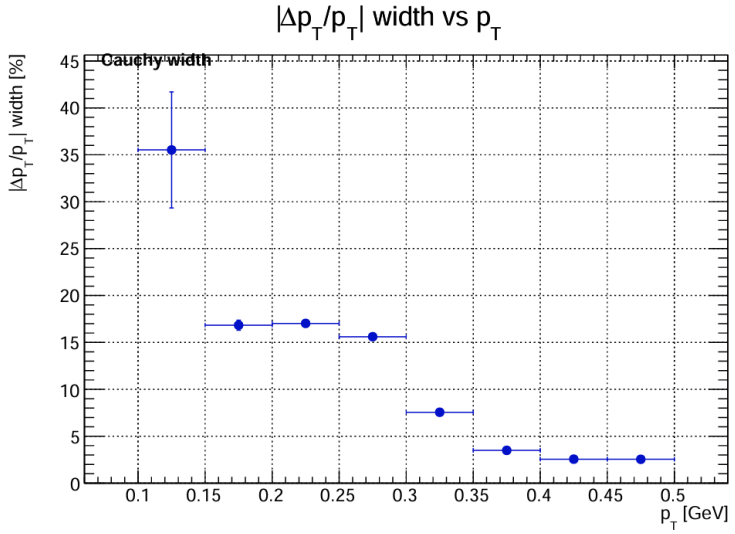
- KF based on previous KF parameters with the covariance * 1000

Performance : resolution and timing

- Resolution : good agreement between CPU and GPU workflows
  - **Hit residual ~ 0.3 mm**
  - **Pt residual $\sim 3\%$ for Pt > 0.35 GeV/c**







- Time cost : **GPU ~ 10x speedup over CPU**

CPU

CKF	197.405 s	(27.4 %)	mean 0.1974 s n=1000
AR	1.809 s	(0.3 %)	mean 0.0018 s n=1000
Track KF	339.352 s	(47.1 %)	mean 0.3394 s n=1000

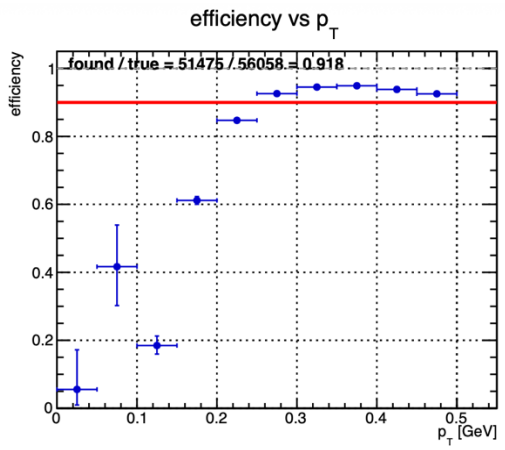
GPU

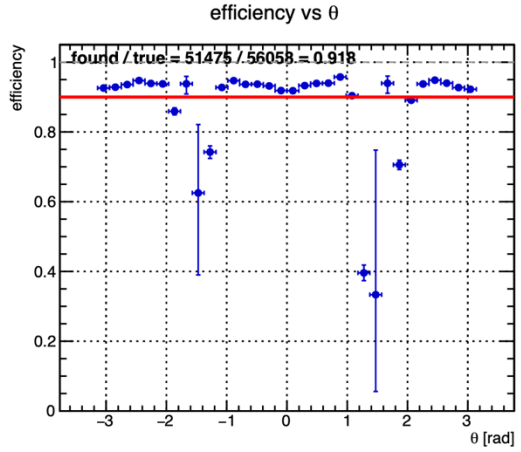
CKF	31.255 s	(12.9 %)	mean 0.0313 s n=1000
AR	5.264 s	(2.2 %)	mean 0.0053 s n=1000
Track KF	10.700 s	(4.4 %)	mean 0.0107 s n=1000

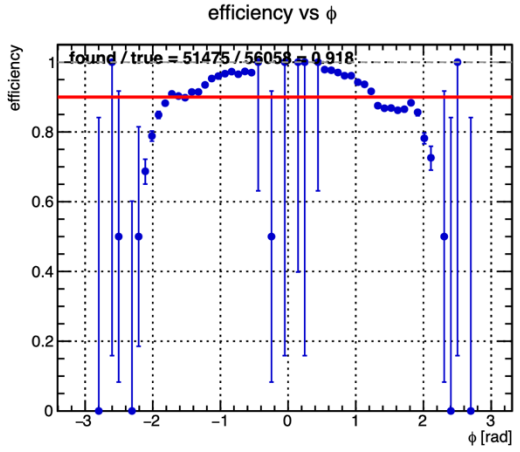
Performance : efficiency

- CPU : ~ 92 % (~ 95 % for tracks with Pt>0.3 GeV)
- GPU : ~ 90 % (~ 95 % for tracks with Pt>0.3 GeV)

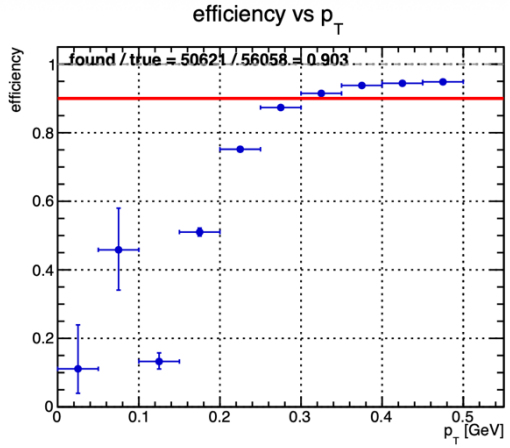
CPU

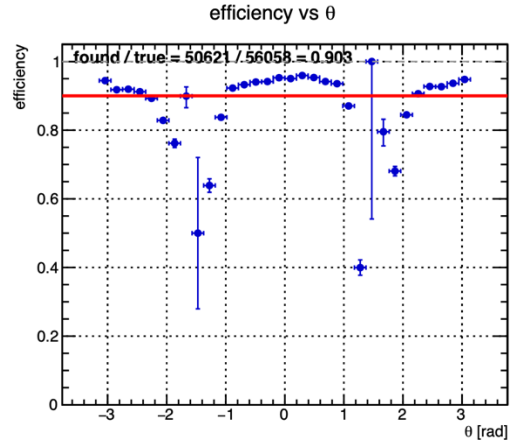


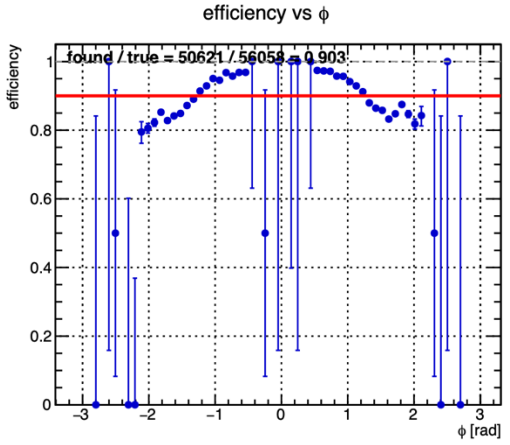




GPU

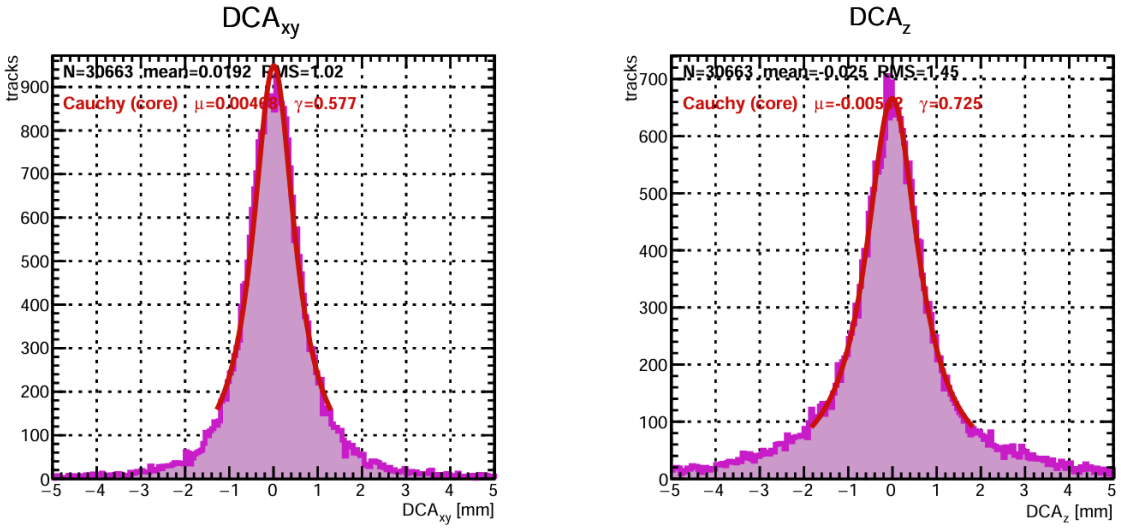


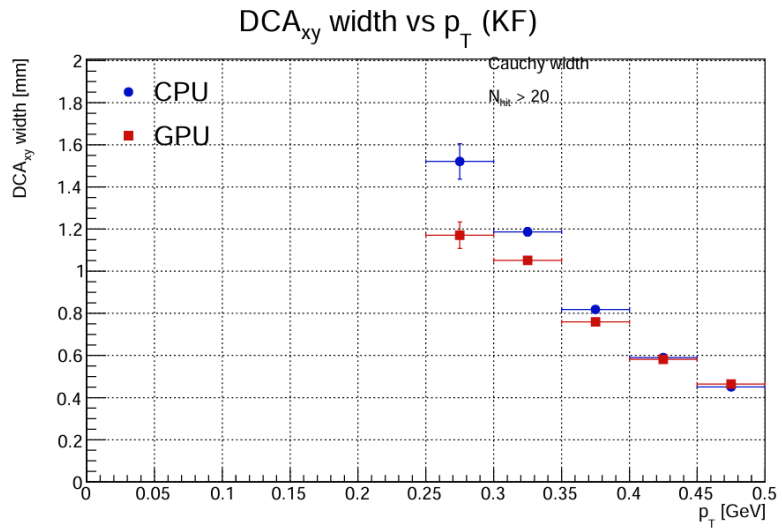


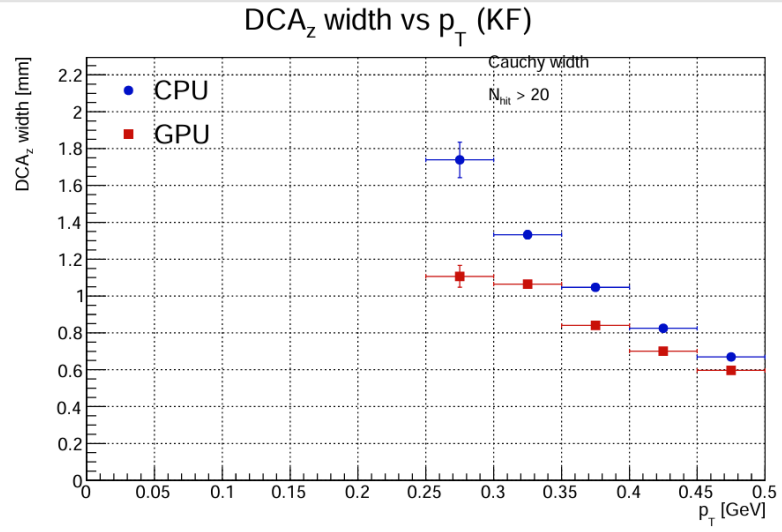


Performance : DCA

- Target (origin) v.s. Point of Closest Approach (**without vertex constrain**)
  - $\mathrm{DCA}_{xy} = \frac{x\mathrm{p}_y - y\mathrm{p}_x}{\sqrt{\mathrm{p}_x^2 + \mathrm{p}_y^2}}$
  - $\mathrm{DCA}_z = z + \mathrm{p}_z t, t = -\frac{x\mathrm{p}_x + y\mathrm{p}_y}{\mathrm{p}_x^2 + \mathrm{p}_y^2}$
- DCA : XY ~ 0.6 mm, Z ~ 0.8 mm for Pt > 0.3 GeV/c with nhit > 20

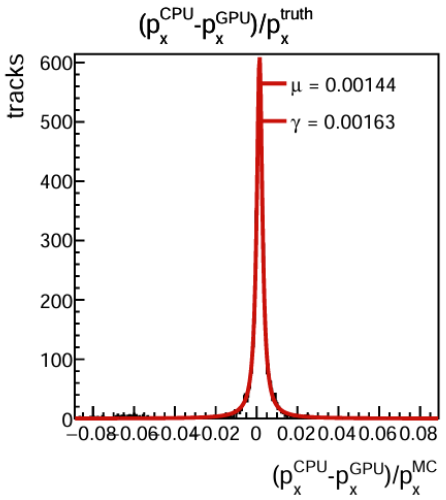


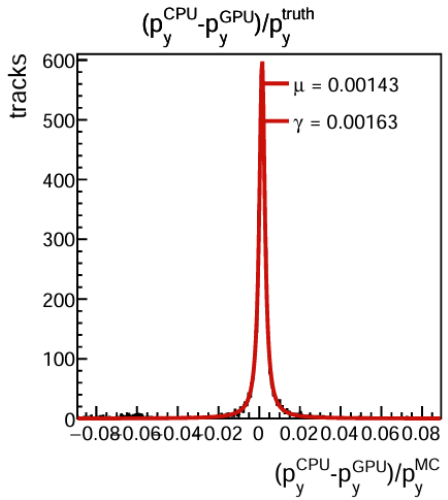


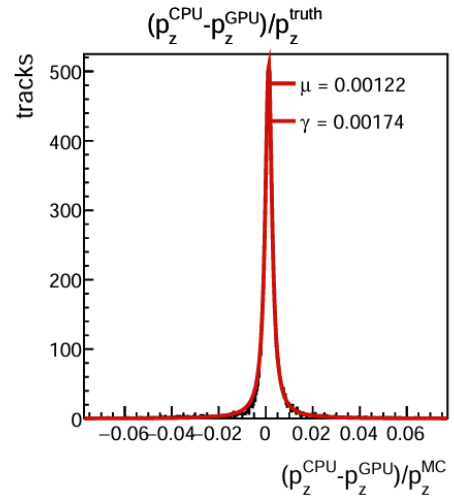


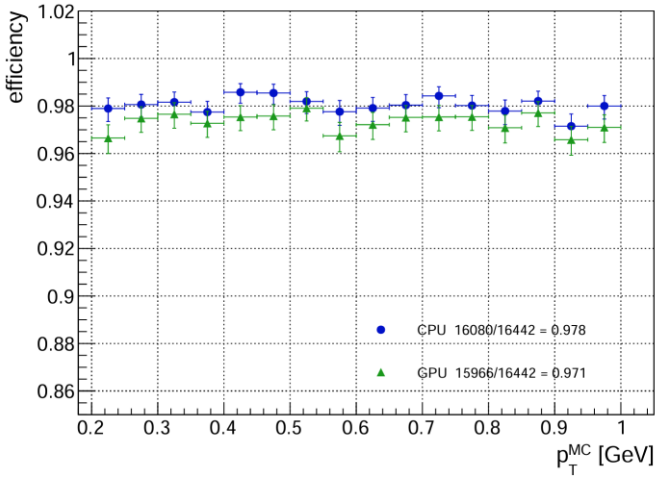
Performance with single-track events

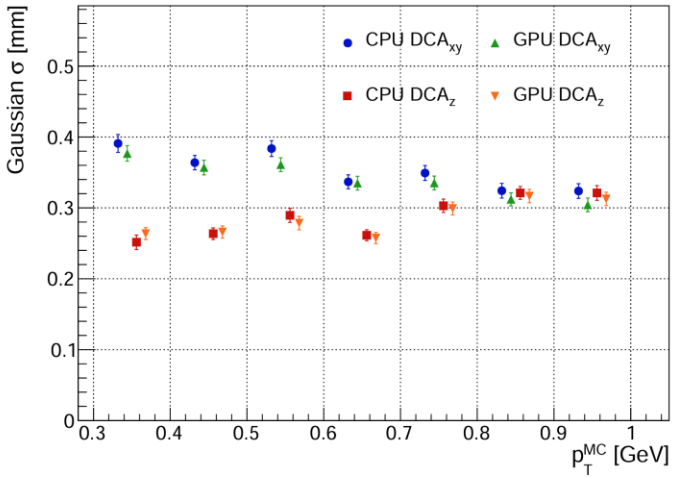
- Datasets
  - 30000 events, 1 proton per event
  - By = 0.5 T, 0.2 < Pt < 1.0 GeV/c
  - Vertex : target position without spatial resolution
- Performance
  - Track parameters : good agreement
  - Efficiency : difference ~ 1%
  - DCA : good agreement
  - **GPU-CPU data transfer takes significantly longer than CKF and KF execution**











Summary

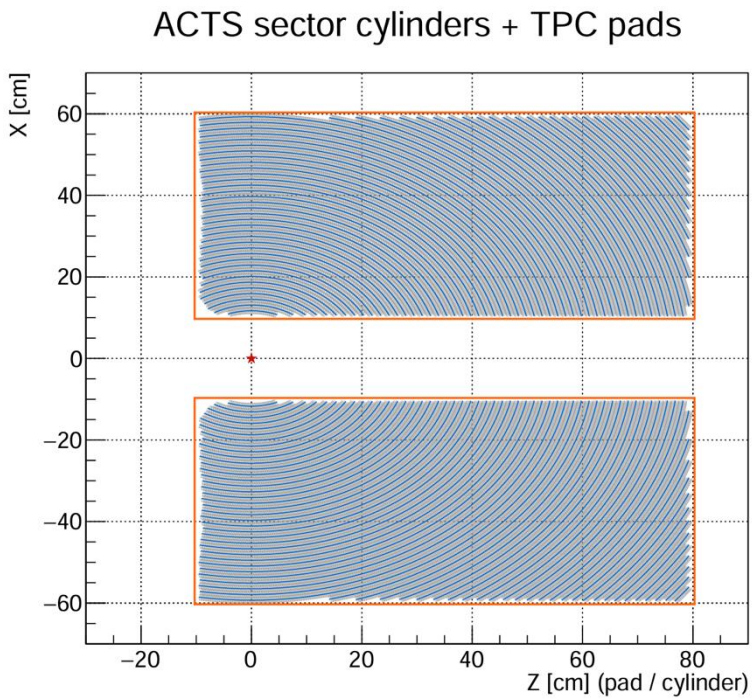
- Performance
  - $\varepsilon > 90\%$ for both CPU and GPU
  - Hit residual ~ 0.25 mm
  - Pt residual $\sim 4\%$ for Pt > 0.3 GeV/c with Nhits > 20
  - GPU ~ 10x speedup over CPU
- ToDo
  - GPU/CPU-based tracking
    1. **Improving the consistency of multi-track events**
    2. **Improve GPU-CPU data transfer efficiency**
  - Fix the alignment algorithm for each `Acts::Surface` [1,2] (ongoing)
  - GNN-based tracking

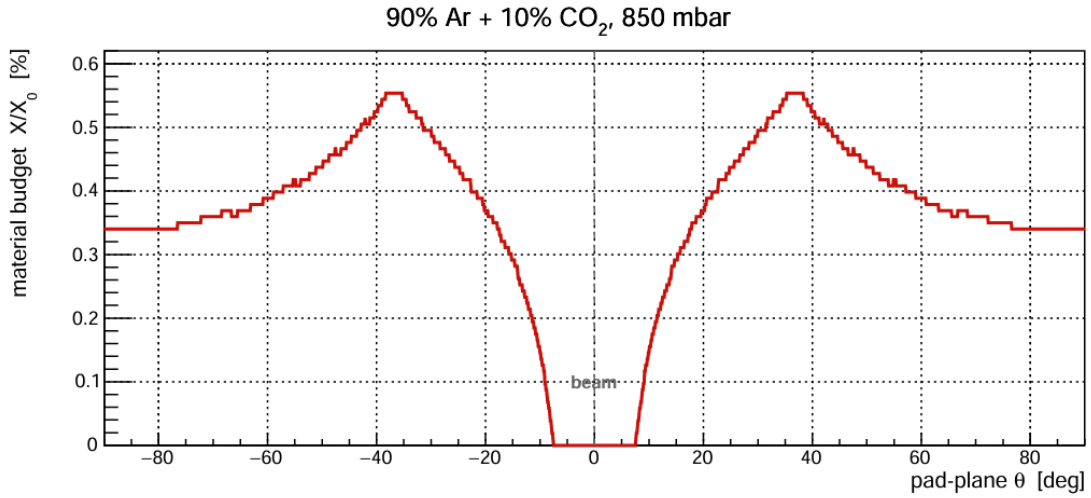
[1] https://github.com/acts-project/acts/pull/6299
[2] https://github.com/acts-project/acts/pull/6503

Backup

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% CO_2, 850 mbar, width 1.4 cm





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

Application of ACTS to CeeRoot : EDM

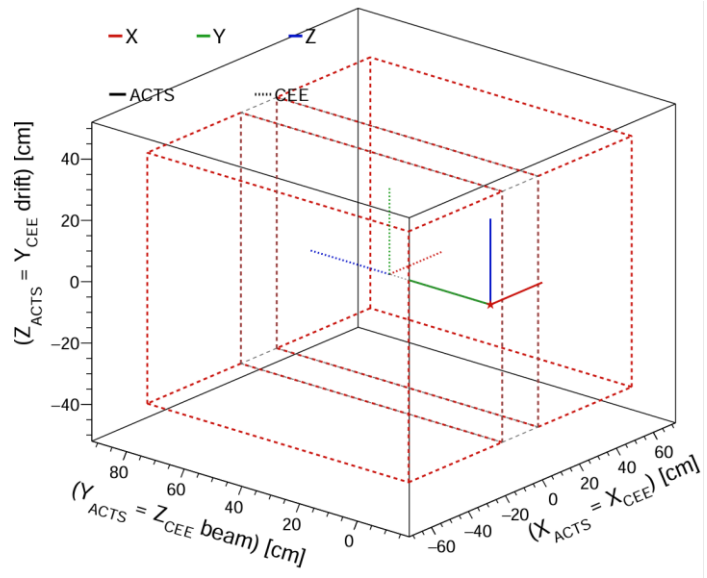
- Clustering (hit -> measurement) is implemented in CeeRoot
  - `CeeTpcHit` -> `Acts::Measurement` and `Acts::SourceLink`
  - Transformation
    1. Global frame: CEE TPC global vs ACTS global

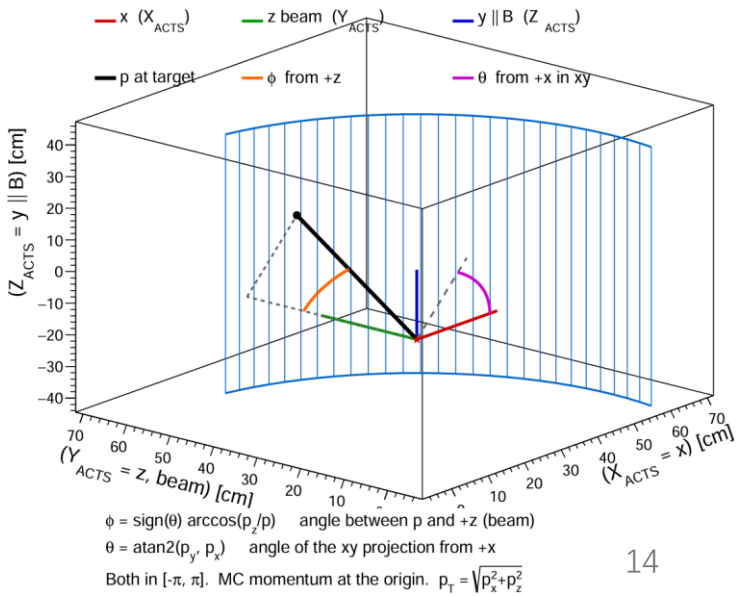
       $$(\boldsymbol{X, Y, Z})_{\boldsymbol{CEE}}\ (\boldsymbol{cm}) = (\boldsymbol{X, Z, Y - 35})_{\boldsymbol{ACTS}} \times \mathbf{10}\ (\boldsymbol{mm})$$

    2. ACTS global frame -> Surface local frame $(R\phi, z)$

       $$\boldsymbol{\phi} = \mathbf{atan2}(\boldsymbol{Y_{ACTS}, X_{ACTS}})$$

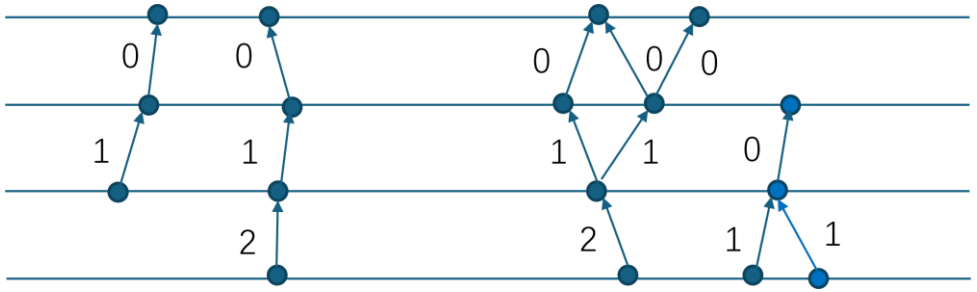
    3. Error : error propagation
    4. `Acts::SourceLink` : `Acts::Measurement` + `Acts::Surface::Identifier` + HitIndex

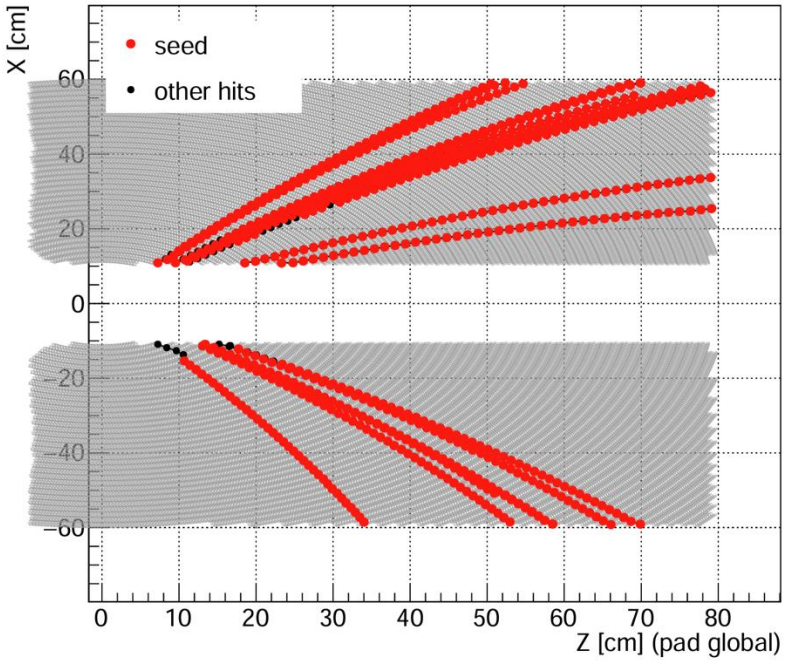




Application of ACTS to CeeRoot : Seeding

- **Seeding : Cellular Automata**
  1. Find doublet : $|\Delta\phi| < 0.3, |\Delta\rho| < 2.08\ cm,$ $|\Delta\theta| < 0.3, \Delta x < 2.51\ \text{cm}, \Delta z < 2.13\ \text{cm}$
  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

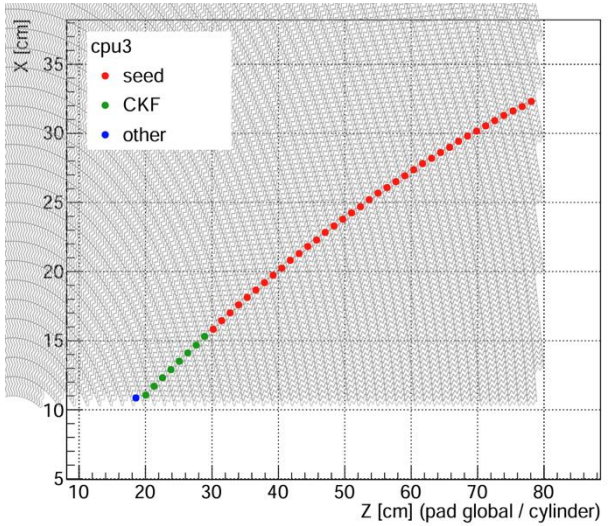


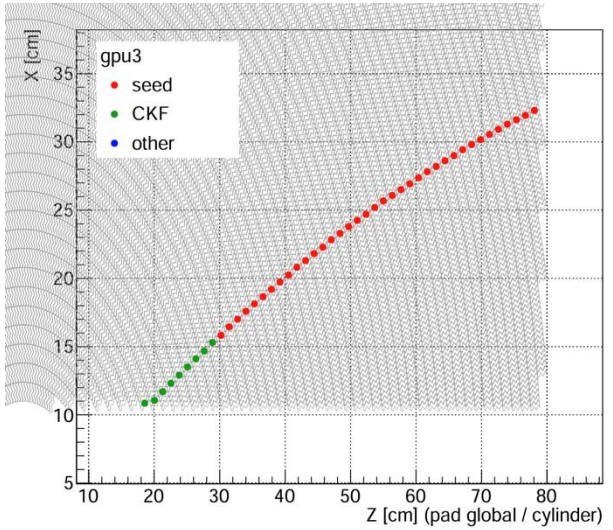


More in Ze Song`s and Aiqiang`s talks

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 : 5 holes ($\chi^2_{local} > 9$)
- **Note : CPU and GPU have independent CKF code in ACTS**





Ambiguity Resolution and Kalman refit

- **Ambiguity resolution : each used hit on at most one track**
  - Rank remaining tracks with shared hits: more hits -> lower χ^2/ndf
- **Kalman fitter : `Acts::DirectNavigator`**
  - **`EigenStepper` : Runge-Kutta + Magnetic map + Material effect**
  - Inner -> outer
  - Sign of charge : same as CKF
- **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 -> …