Application of TRACCC seeding to the CEPC vertex detector

YiZhou Zhang, Xiaocong Ai, Tao Lin, WeiDong Li
zhangyz@ihep.ac.cn

18th May 2024
Zhengzhou University

Outline

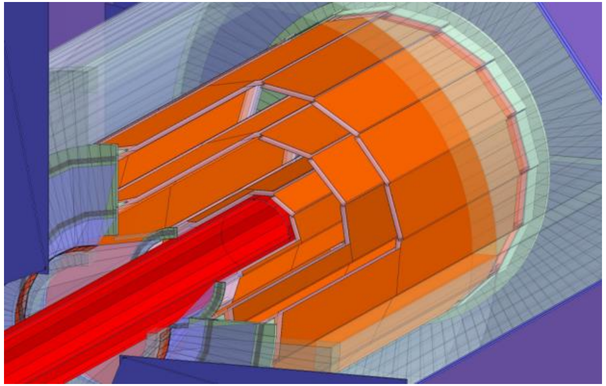
1 Introduction

Circular Electron Positron Collider (CEPC)

The CEPC is a future experiment mainly designed to precisely measure the Higgs boson's properties and search for new physics beyond the Standard Model.

- At 250 GeV: Higgs bosons are produced (4×10^6)
- At 160 GeV: W bosons are produced ($> 10^8$)
- At 90 GeV: Z bosons are produced ($> 4 \times 10^{12}$)

*The conceptual design report (CDR) has been completed in Oct. 2018. My work is all based on the design of CDR.



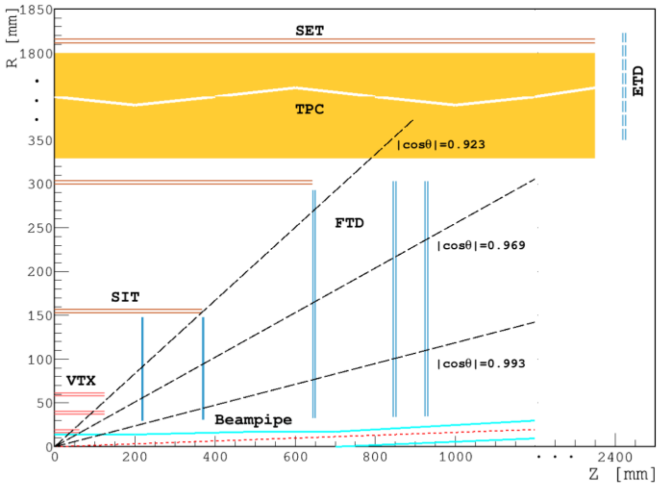
Schematic view of CEPC vertex detector

Only the silicon sensor sensitive region (in orange) is depicted.
The vertex detector surrounds the beam pipe (in red).

CEPC vertex detector

The VerTeX detector (VTX) is the innermost tracker playing a dominant role in determining the vertices of a collision event.

The baseline layout of the CEPC vertex detector consists of three concentric cylindrical double-sided layers of high spatial resolution silicon pixel sensors, providing six precise space-points to measure charged particles traversing the detector.



Layout of the CEPC baseline tracker

The VTX is located closest to the interaction point.

1 Introduction

TRACCC

TRACCC is one of the R&D lines aiming for developing the demonstrator for a full tracking chain for accelerators within the ACTS project.

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

✅: exists, 🟡: work started, ⚪: work not started yet

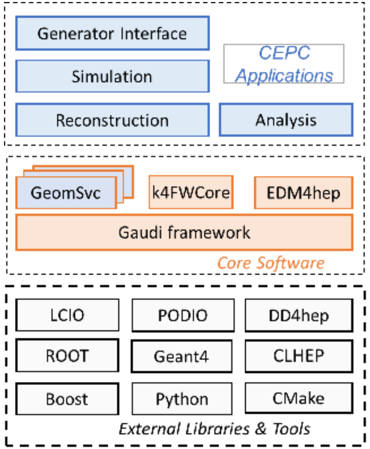
Status of TRACCC

CEPC software (CEPCSW) environment

Applications: simulation, reconstruction and analysis

Core software:

- framework: Gaudi
- detector description tool: DD4hep
- event data model: EDM4hep
- event data manager: k4FWCore
- Other CEPC-specific components



CEPCSW structure

This Contribution

Imply the seeding algorithm for the VTX detector based on TRACCC in the CEPCSW environment.

1 Introduction

Overview of This Contribution

Imply the seeding algorithm for the VTX detector based on TRACCC in the CEPCSW environment.

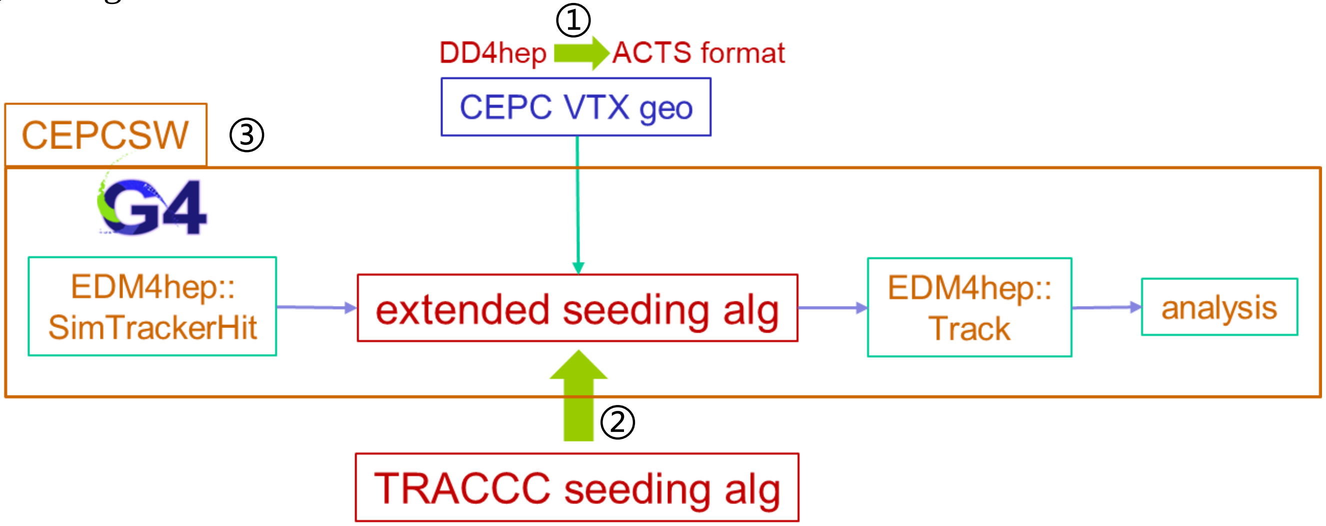
① Convert the CEPC VTX geometry to ACTS format
② Extend the seeding algorithm for CEPC VTX detector structure
③ Integration of TRACCC with CEPCSW

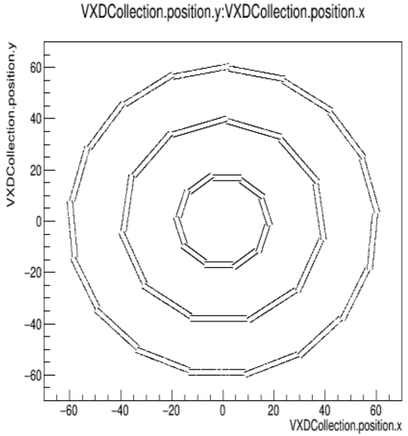
	R (mm)	$\|z\|$ (mm)	$\|\cos\theta\|$	σ (μm)
Layer 1	16	62.5	0.97	2.8
Layer 2	18	62.5	0.96	6
Layer 3	37	125.0	0.96	4
Layer 4	39	125.0	0.95	4
Layer 5	58	125.0	0.91	4
Layer 6	60	125.0	0.90	4

Layout of CEPC VTX detector

CEPC VTX: three layers, both sides of which are mounted with silicon pixel sensors

TRACCC: three layers with single-sided silicon pixel sensors





The X–Y projection of the VTX

Outline

2 CEPC Geometry in ACTS

Implementation of CEPC VTX geometry in ACTS format

① Convert the CEPC VTX geometry file (in DD4hep format) to tgeo format.
② Write the VTX config file to specify the volumes that needs to be generated.
③ Use ACTS' tgeo reader to generate CEPC VTX detector geometry in csv format.
④ Write the digitization config file to provide the segmentation information of each surface.
⑤ *Verification: Use Fast ATLAS Track Simulation (FATRAS) & ACTS' digitization tool to produce full simulation information and generate cells.

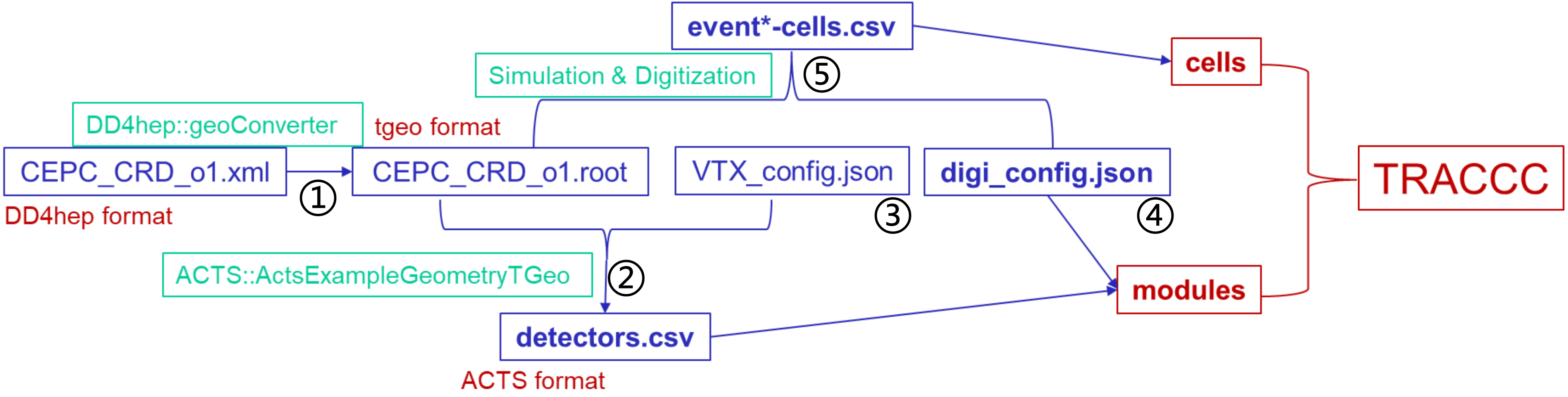
The correctness of the geometric transformation is verified.

②
```
{
  "Volumes": [
    {
      "Splitters": {
        "CylinderDisk": {
          "geo-tgeo-cyl-nphi-segs": 100,
          "geo-tgeo-cyl-nz-segs": 1,
          "geo-tgeo-disc-nphi-segs": 0,
          "geo-tgeo-disc-nr-segs": 0
        }
      },
      "geo-tgeo-binning0": {
        "central": [[-1,0]],
        "negative": [],
        "positive": []
      },
      "geo-tgeo-binning1": {
        "central": [[-1,0]],
        "negative": [],
        "positive": []
      },
      "geo-tgeo-cyl-disc-split": false,
      "geo-tgeo-itk-module-split": false,
      "geo-tgeo-dc-layer-split": false,
      "geo-tgeo-cepc-vxd-layer-split": true,
      "geo-tgeo-cepc-sit-layer-split": false,
      "geo-tgeo-cepc-ftd-layer-split": false,
      "geo-tgeo-urwell-layer-split": false,
      "geo-tgeo-layer-r-ranges": {
```

CEPC_VTX_config.json
Written by Xiaocong

④
```
{
  "acts-geometry-hierarchy-map": {
    "format-version": 0,
    "value-identifier": "digitization-configuration"
  },
  "entries": [
    {
      "layer": 2,
      "sensitive": 1,
      "value": {
        "geometric": {
          "digital": false,
          "indices": [
            0,
            1
          ],
          "segmentation": {
            "binningdata": [
              {
                "bins": 220,
                "max": 5.5,
                "min": -5.5,
                "option": "open",
                "type": "equidistant",
                "value": "binX"
              },
              {
                "bins": 2500,
                "max": 62.5,
                "min": -62.5,
                "option": "open",
                "type": "equidistant",
                "value": "binY"
              }
            ]
          },
          "thickness": 0.05,
          "threshold": 0.0
        }
      },
      "volume": 3
```

CEPC_digi_config.json



Details of Geometry conversion

Different geometry id

ACTS & TRACCC are using different geometry id:

- ACTS use lower 0-27 digit to store the surface id
- TRACCC use lower 8-47 digit to store the surface id

Therefore, the geometry id of the ACTS' geometries has been modified.

```
private:
 // (2^8)-1 = 255 volumes
 static constexpr Value kVolumeMask = 0xff00000000000000;
 // (2^8)-1 = 255 boundaries
 static constexpr Value kBoundaryMask = 0x00ff000000000000;
 // (2^12)-1 = 4096 layers
 static constexpr Value kLayerMask = 0x0000fff000000000;
 // (2^8)-1 = 255 approach surfaces
 static constexpr Value kApproachMask = 0x0000000ff0000000;
 // (2^28)-1 sensitive surfaces
 static constexpr Value kSensitiveMask = 0x000000000fffffff;
```

geometry id of ACTS (Acts::GeometryIdentifier)

```
private:
// (2^12)-1 = 4095 volumes
static constexpr value_t k_volume_mask = 0xfff0000000000000;
// (2^4)-1 = 15 surface categories
static constexpr value_t k_id_mask     = 0x000f000000000000;
// (2^40)-1 = 1099511627775 surfaces (per volume and id)
static constexpr value_t k_index_mask  = 0x0000ffffffffff00;
// (2^8)-1 = 255 extra values
static constexpr value_t k_extra_mask  = 0x00000000000000ff;
```

geometry id of TRACCC (detray::geometry::barcode)

Adapt cuts of the parameters

- TRACCC use some parameters to determine whether the space points can form a triplet. The cuts for some of the parameters are adapted to CEPC pixel geometry.

```
if (deltaCotTheta2 - error2 > 0) {
    deltaCotTheta = std::abs(deltaCotTheta);
    error = std::sqrt(error2);
    dCotThetaMinusError2 = deltaCotTheta2 + error2 -
                           static_cast<scalar>(2.) * deltaCotTheta * error;
    if (dCotThetaMinusError2 > scatteringInRegion2) {
        return false;
    }
}
```

modified

triplet_finding_helper::isCompatible

Modify the EDM

Add "track id" to the EDM of cells, so we can trace back from the found seeds to origin tracks:

- Seed → space point → cluster → cell

*for the evaluation of track efficiency.

```
struct cell {
    channel_id channel0 = 0;
    channel_id channel1 = 0;
    scalar activation = 0.;
    scalar time = 0.;

    using link_type = cell_module_collection_types::view::size_type;
    link_type module_link;

    uint64_t track_id = 0;
};
```

Add track id to EDM of TRACCC

Outline

3 Seeding algorithm for CEPC

6-layers seeds finding

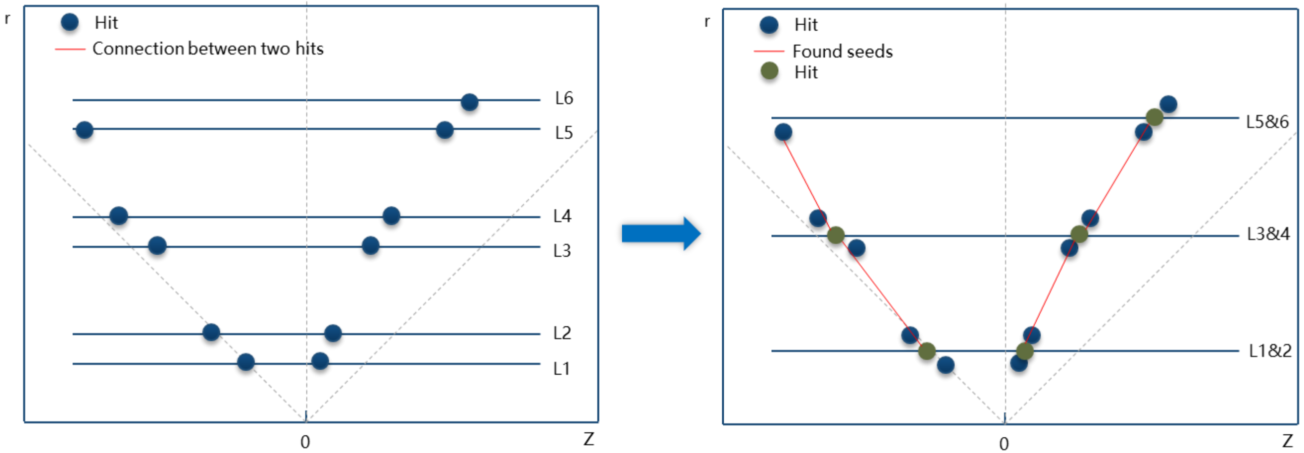
2 methods of modify the TRACCC algorithm to be suitable for 6-layers CEPC geometry.

1. Triplets Finding:
Before seeding, treating adjacent layers as one layer, and considering nearby space-points in two layers as one-space point.

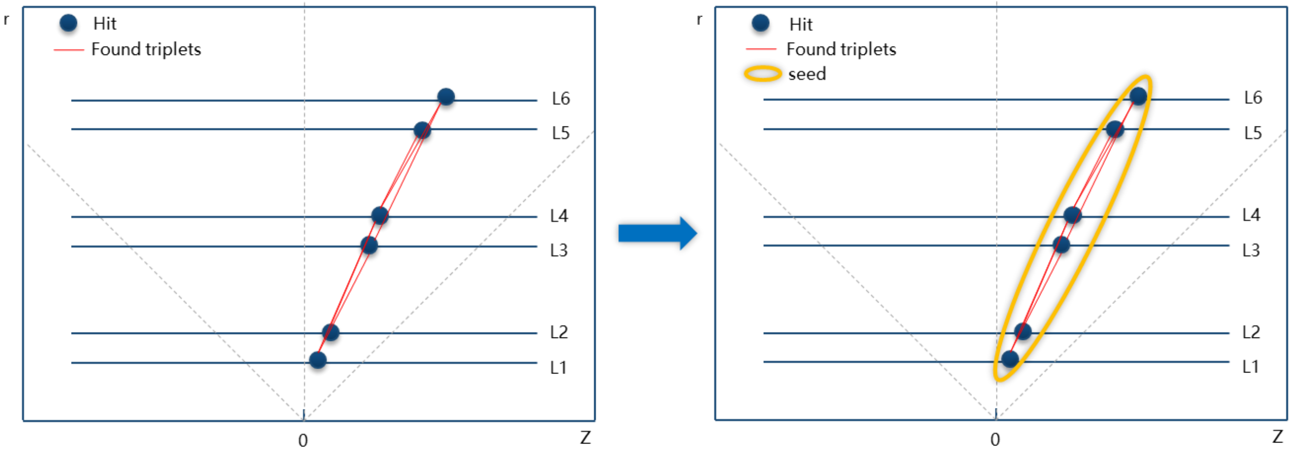
2. Seed Formation:
After seeding, combine the found triplets that sharing the same space-points into a "big" seed.

We have implied Seed Formation in TRACCC.

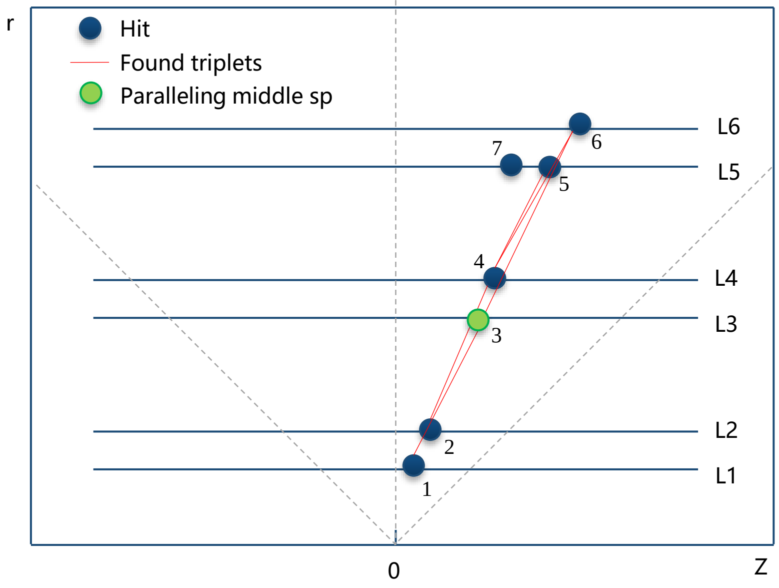
Triplets Finding



Seed Formation



3 Seeding algorithm for CEPC



In GPU

Example:
Paralleling for hit 3 ☞

	bottom	middle	top
	1	3	5
lowest d_0	1	3	6
	1	3	7
	2	3	5
	2	3	6
	2	3	7

Bot_inner	Bot_outer	Mid_inner	Mid_outer	Top_inner	Top_outer
1	2	3	3	5	6

{1, 2}
1.radius() < 2.radius()

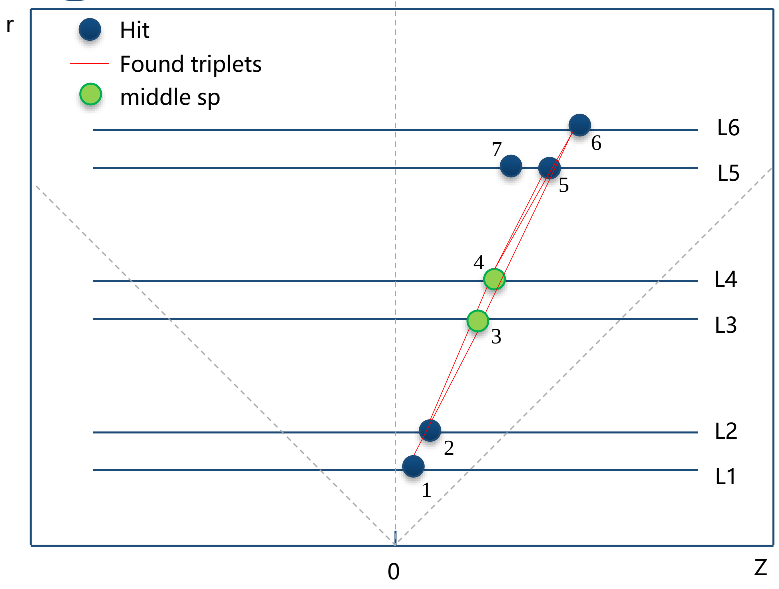
{5, 6} | {7, 6}
5.radius() < 6.radius()

6-layers seeds finding: Seed Formation steps in GPU

For each middle space point in parallel:

1. pick the triplet with the lowest impact params (d_0) among all the triplets where the middle sp is located
2. find the bottom sp & top sp that are closest to the bottom sp & top sp of the current triplet
3. form a new seed of 5 points and sort them according to their radius (distance to the origin of coordinates)

3 Seeding algorithm for CEPC



In CPU

For hit 3

Bot_inner	Bot_outer	Mid_inner	Mid_outer	Top_inner	Top_outer
1	2	3	3	5	6

For hit 4 +

Bot_inner	Bot_outer	Mid_inner	Mid_outer	Top_inner	Top_outer
1	2	4	4	5	6

Bot_inner	Bot_outer	Mid_inner	Mid_outer	Top_inner	Top_outer
1	2	3	4	5	6

{3， 4}
3.radius() < 4.radius()

6-layers seeds finding: Seed Formation step in CPU

Iterate through all new 5-points seeds:

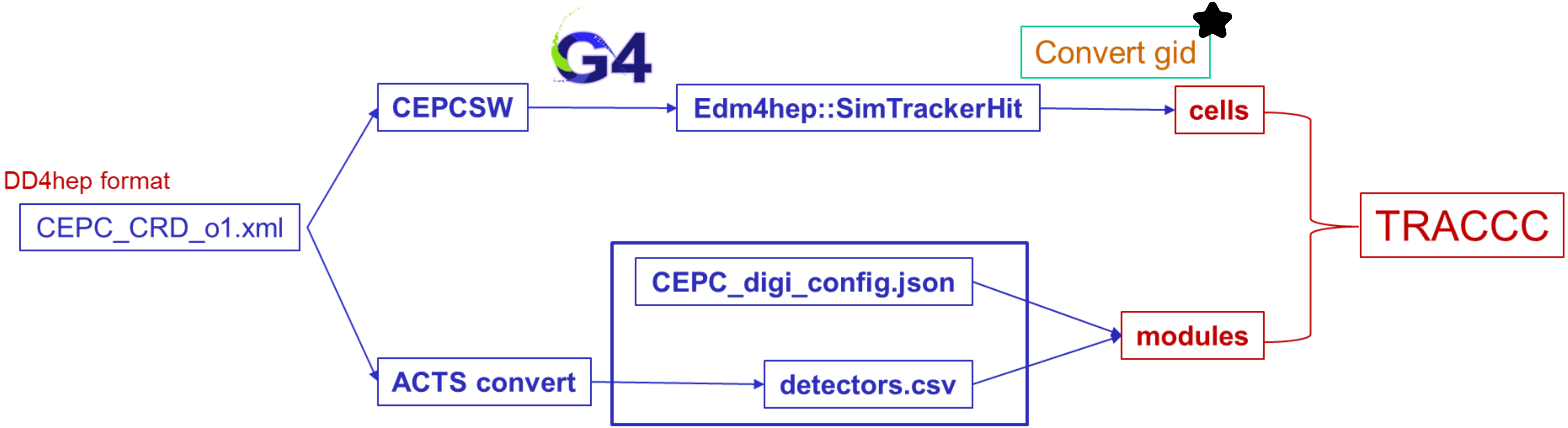
if two seeds have the same bottom sp & top sp, merge both into hexaplets (6-layers seeds)

Outline

4 Integration of TRACCC with CEPCSW

Two steps of the integration

- Use CEPCSW's Geant4 simulation as the input to be reconstructed in TRACCC.
- Package the extended seeding algorithm, and calling them in CEPCSW.



Layout of the deployment

The hits generated by the CEPCSW simulation are converted to the EDM of TRACCC before being reconstructed.

4 Integration of TRACCC with CEPCSW

Modify module order for Gid conversion

Each ladder is split into two parts based on $z < 0$ *or* $z > 0$.

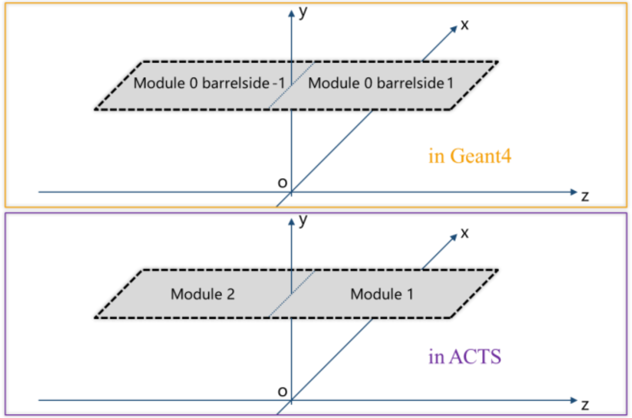
- In CEPCSW: barrelside = 1 for $z > 0$, barrelside = -1 for $z < 0$
- In ACTS: Odd #module for $z > 0$, Even #module for $z < 0$

Layer id	# of module	barrelside
0	10	±1
1	10	±1
2	11	±1
3	11	±1
4	17	±1
5	17	±1

Modules in CEPCSW

Layer id	# of module
2	20
4	20
6	22
8	22
10	34
12	34

Modules in ACTS



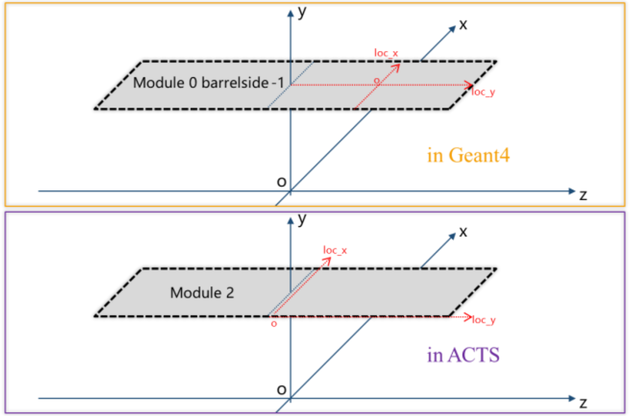
Different module order between Geant4 & ACTS

Adjust the local coordinates

- Geant4's local coordinates originate at the Centre of the module
- ACTS at the bottom-left corner of the module

```
<readout name="VXDCollection">
  <segmentation type="CartesianGridXY" grid_size_x="25*um" grid_size_y="25*um"/>
  <id>system:5,side:-2,layer:9,module:8,sensor:8,barrelside:-2,x:-11,y:-14</id>
</readout>
```

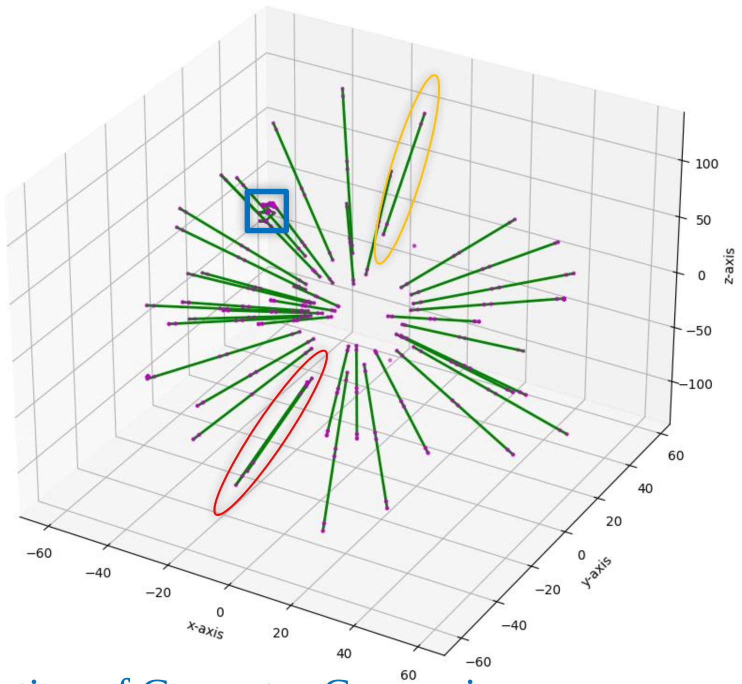
Modified the cell id of G4 simulated hits
for getting the local coordinates



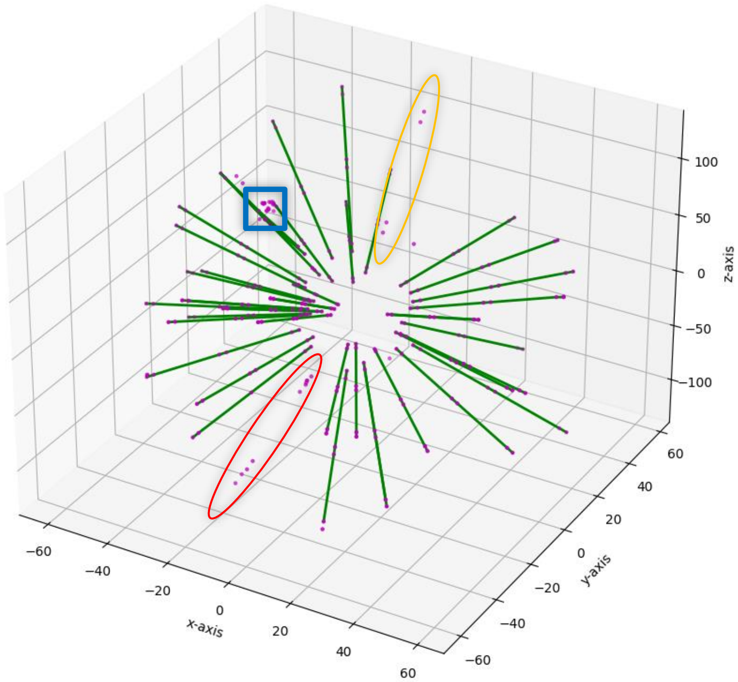
Different local coordinates between Geant4 & ACTS

4 Integration of TRACCC with CEPCSW

Geant4 simulation (1 event, 50 tracks)



TRACCC reconstruction



Verification of Geometry Conversion

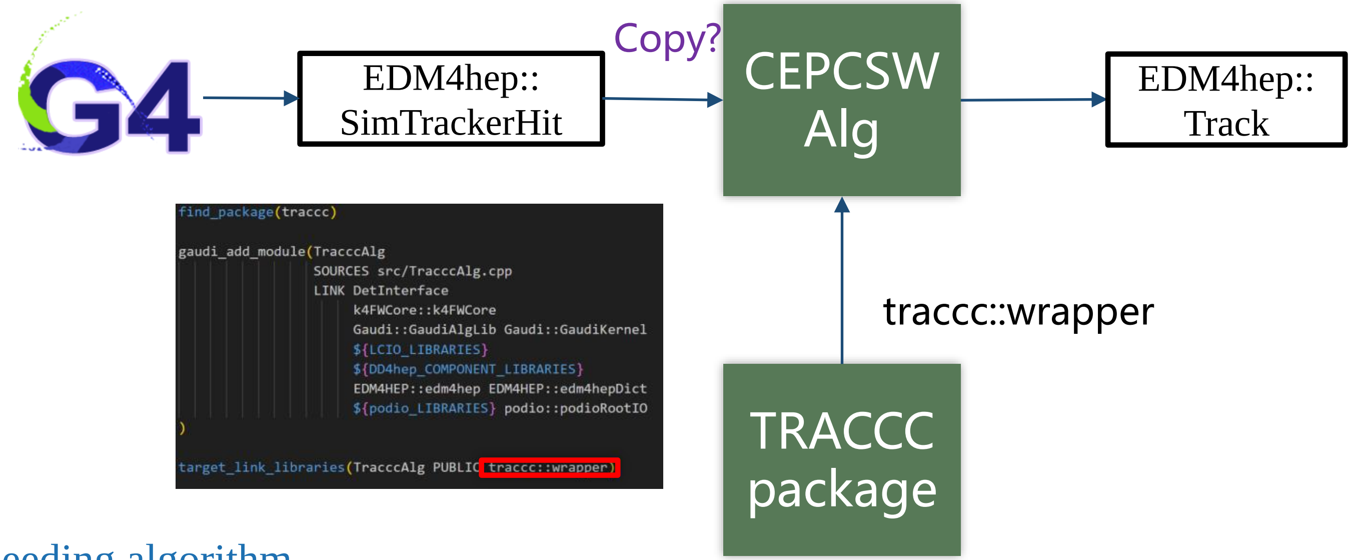
Simulated mu- of 100 Gev in Geant4, and reconstructed in TRACCC

*The yellow and red parts of the simulation do not have hits in the outermost layers (layer 4/5)

*The blue part G4 produced secondary particles (e-).

Modules are found correctly!

4 Integration of TRACCC with CEPCSW




```
find_package(traccc)

gaudi_add_module(TracccAlg
                 SOURCES src/TracccAlg.cpp
                 LINK DetInterface
                      k4FWCore::k4FWCore
                      Gaudi::GaudiAlgLib Gaudi::GaudiKernel
                      ${LCIO_LIBRARIES}
                      ${DD4hep_COMPONENT_LIBRARIES}
                      EDM4HEP::edm4hep EDM4HEP::edm4hepDict
                      ${podio_LIBRARIES} podio::podioRootIO
)

target_link_libraries(TracccAlg PUBLIC traccc::wrapper)
```

Package the seeding algorithm

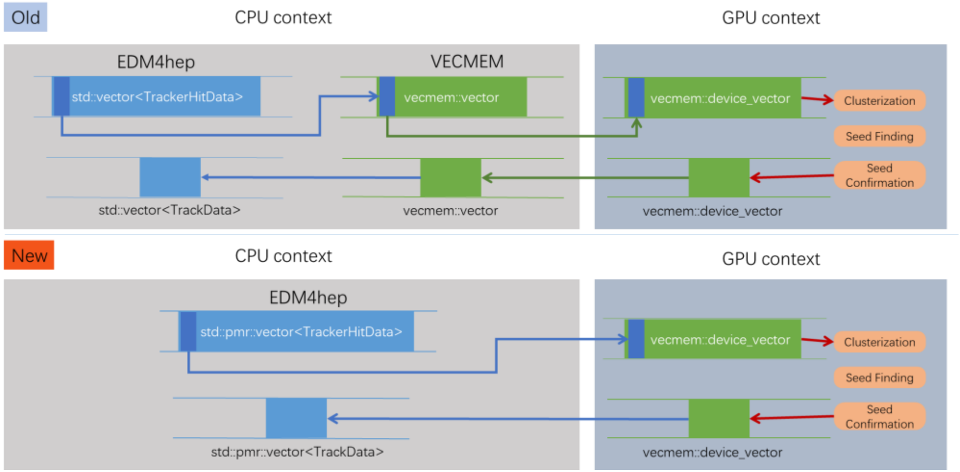
- Write a wrapper to wrap the seeding functions that CEPCSW needed.
- Calling the TRACCC package in CEPCSW alg.
- Pull request: https://github.com/cepc/CEPCSW/pull/270

Avoid the overhead from data copy

- We want TRACCC be able to use the hits data simulated by G4 directly !!
- EDM4hep and *VecMem may use the same memory.

* TRACCC uses VecMem as the vectorised data model across multiple device types.

4 Integration of TRACCC with CEPCSW

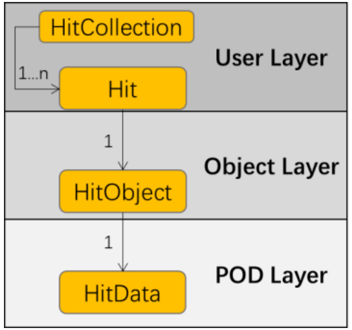


Modify the EDM4hep

We want EDM4hep & VecMem use the same storage format (std::pmr::vector),
So TRACCC can directly use the hit data with no data-copy.

Modify the data storage format of PODIO

EDM4hep is generated by PODIO,
so we modify the DataContainer of PODIO:



Layout of the PODIO storage format

Add Collection layer interfaces:

```
std::pmr::vector<{{ class.bare_type }}Data> data()
```

Add CollectionData layer interfaces:

```
{{ class.bare_type }}DataContainer getdata();
```

Modify the DataContainer storage format (vector → pmr::vector)

```
using {{ class.bare_type }}ObjPointerContainer = std::deque<{{ class.bare_type }}Obj*>;
using {{ class.bare_type }}DataContainer = std::pmr::vector<{{ class.bare_type }}Data>;
```

We add interfaces to get pmr::vector directly.

4 Integration of TRACCC with CEPCSW

Customized EDM4hep data collection

- Define a data collection whose member is totally the same with the EDM of TRACCC

- So we can directly use edm4hep::ACTSCells as the input of TRACCC.

```
#-------------  ACTSCells
edm4hep::ACTSCells:
  Description: "Cells for reconstruction in TRACCC Project"
  Author: "Yizhou Zhang, IHEP"
  Members:
    - uint32_t channel0          //channel0
    - uint32_t channel1          //channel1
    - float activation           //activation
    - float time                 //time
    - uint32_t module_link       //module_link
```

edm4hep.yaml

Verification

- Now TRACCC can directly read the simulated hits from Geant4 which is stored in EDM4hep format.
- No non-essential data-copy occurs.

In CEPCSW alg

In TRACCC alg

```
Running Seeding on device: Quadro RTX 8000
Initializing ...
EventLoopMgr         WARNING Unable to locate service "EventSelector"
EventLoopMgr         WARNING No events will be processed from external input.
ApplicationMgr          INFO Application Manager Initialized successfully
ApplicationMgr          INFO Application Manager Started successfully
TracccRun               INFO begin execute TracccRun
TracccRun               INFO reading hits from csv
TracccRun               INFO the size of the csv's cells vector: 199547
TracccRun               INFO creating edm4hep::ACTSCellsCollection
TracccRun               INFO the address of the cells vector: 3963cf0
TracccRun               INFO the size of the cells vector: 199547
TracccRun               INFO running traccc
the address of the cells vector: 3963cf0
the size of the cells vector: 199547
Traccc Success
==>Elapsed times...
         Clusterization (sycl)  5 ms
                Seeding (sycl)  4 ms
            Track params (sycl)  0 ms
                     Wall time  11 ms
TracccRun               INFO event 0 success
```

Running in CEPCSW

- The address of pmr::vector does not changed.
- No data copy occurs.

4 Integration of TRACCC with CEPCSW

Seeding efficiency evaluation

Geant4 simulated 50,000 tracks (50 tracks per event * 1,000 events)
Particle: mu- Energy: 5 Gev
A seed is "good" IFF all the space-points have the same track id.

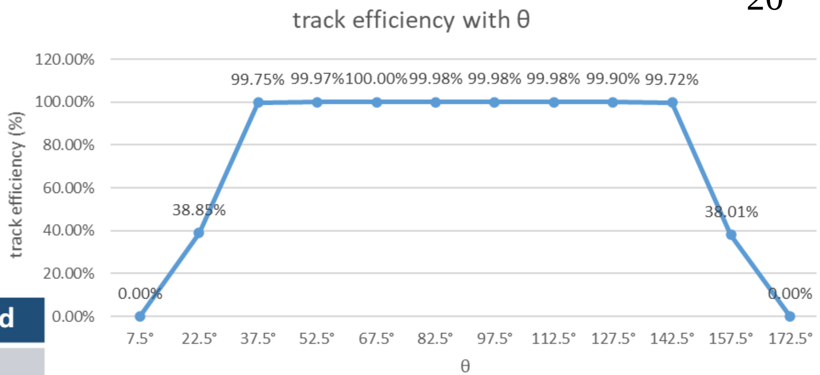
Seed #	Total	Good seed	1 hit err	2 hit err	3 hit err	Bad seed
6	25,000	24,401	493	94	3	9
5	1,858	1,635	209	7		7
4	7,864	7,851	8	5		0
3	5	5	0			0
sum	34,727	33,892(97.60%)	710	106	3	16(0.01%)

10,889 (21.78%) seeds are not reconstructible:
- missing hits on one of the ladders
- secondary particles generated

4,384 (8.77%) seeds are not reconstructed by TRACCC

Computing evaluation of TRACCC seeding

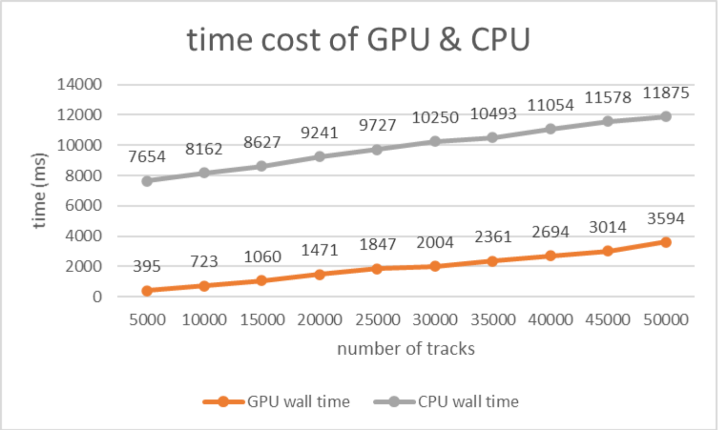
Run TRACCC in heterogeneous device:
- CPU: Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz
- GPU: NVIDIA Corporation TU102GL [Quadro RTX 8000]



Track efficiency with θ

$$track\ eff = \frac{N_{rec\ tracks}}{N_{total\ tracks}}$$

Material effects track efficiency at low θ angle.



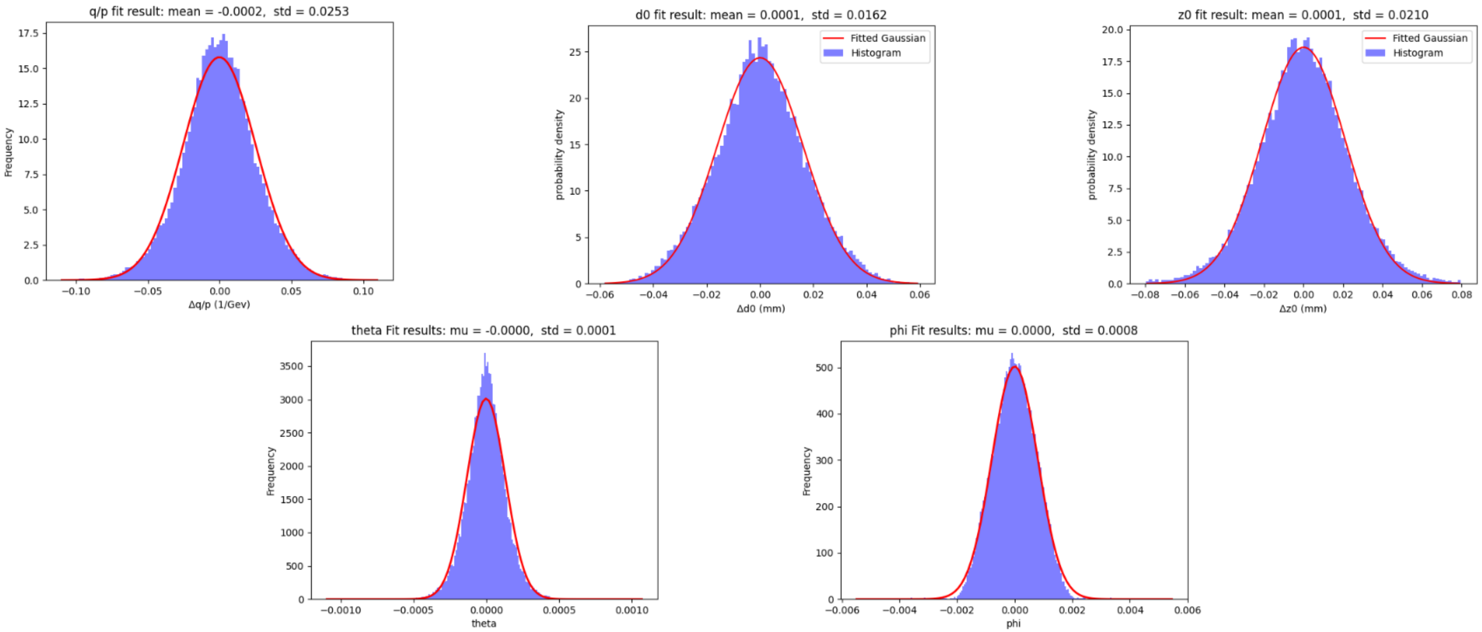
4 Integration of TRACCC with CEPCSW

Parameter Estimation

The param estimation of TRACCC is aimed to estimate 3 space-points seed.

The modified method for 6 sp seed is based on V. Karimaki NIM A305 (1991) 187-191:

https://doi.org/10.1016/0168-9002(91)90533-V



Difference between reconstruction and simulation (rec - sim) track param

Track parameters include qoverp, d0, z0, theta, phi

Particle: mu- Energy: 5 Gev

Summary & future work

Application of TRACCC seeding to the CEPC vertex detector

Summary

- Implement the CEPC pixel detector geometry (both 3-layers & 6-layers) in ACTS format
- The TRACCC has been successfully applied for GPU-based seeding for CEPC vertex detector
- Update the TRACCC algorithm to be suitable for 6-layers CEPC geometry
- Convert the G4 simulation hits to ACTS format as the input of TRACCC
- Use one common memory for both EDM4hep and VecMem to avoid the overhead from data copy
- Deploy the TRACCC as a plugin to CEPCSW

Future work

- Further analysis of seeding efficiency & computing performance

Thank you

YiZhou Zhang, Xiaocong Ai, Tao Lin, WeiDong Li
zhangyz@ihep.ac.cn

18th May 2024
Zhengzhou University

Backup