TRACCC在CEPC顶点探测器中的应用

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

15th Aug 2024

Outline

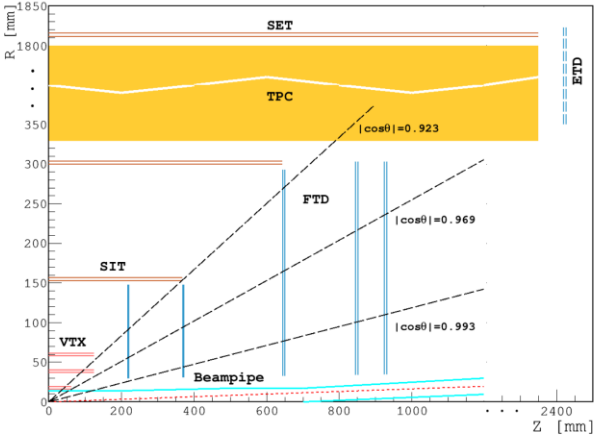
1 Introduction

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. And the Technical Design Report (TDR) is now being written.



Layout of the CEPC baseline tracker (CDR)

CEPC探测器重建面临的挑战:

- 多事例堆积
- 高束流本底（Z能区）
- 高数据采集率

ACTS Common Tracking Software

ACTS是一个用于高能物理和核物理实验带电粒子径迹重建，不依赖于特定的实验装置的软件包。

ACTS documents: https://acts.readthedocs.io/

TRACCC is one of ACTS R&D projects, providing full chain demonstrator for track reconstruction on accelerators.

This Contribution

We plan to apply ACTS' reconstruction tool in the reference detector of TDR, and to compare its performance with our origin reconstruction algorithm.

This Contribution will introduce the integration of ACTS & TRACCC with CEPC software (CEPCSW) environment.

Code working in progress:
https://code.ihep.ac.cn/zhangyz/cepcsw-acts/-/tree/master/Reconstruction/InDetActsTracking

1 Introduction

TRACCC & SYCL

使用SYCL编写的径迹重建算法，是TRACCC的主要开发方向之一。

SYCL is a high-level C++ programming model. An uniformed written code can run on a variety of platforms.
* High Portability and Programming Efficiency

TRACCC is developing track reconstruction algorithm using SYCL. Its track finding algorithm is not finished yet.

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

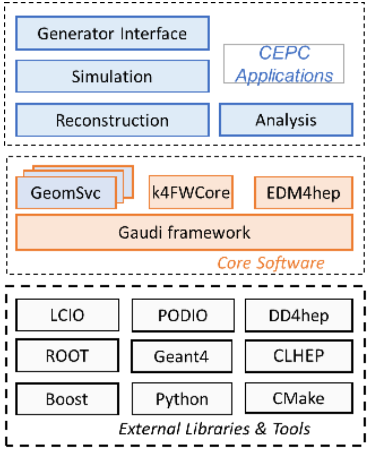
Status of TRACCC

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

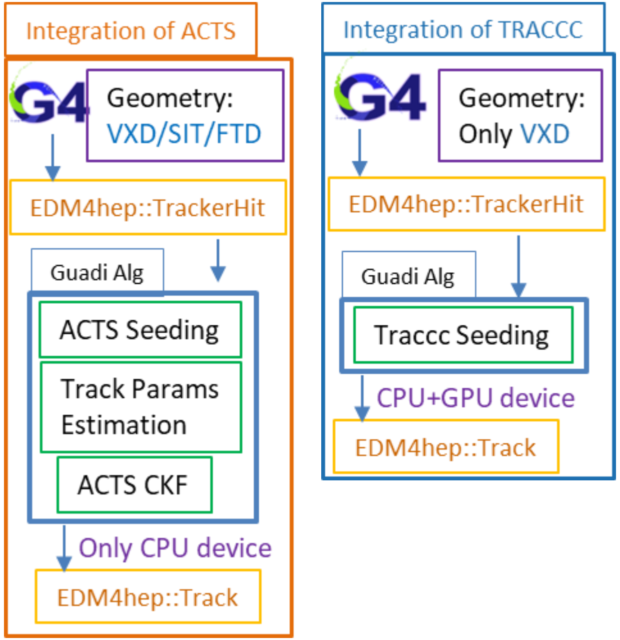


CEPCSW structure

Integration of TRACCC

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

1 Introduction



Integration of ACTS & TRACCC

Overview of This Contribution

- Integration of ACTS' full silicon track reconstruction algorithm & TRACCC's seeding algorithm on VTX detector with CEPCSW

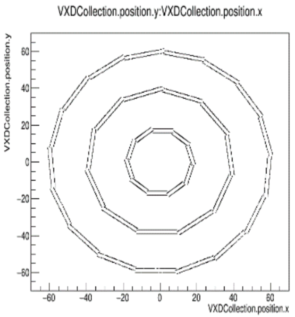
Steps:

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

CEPC VTX: three layers, both sides of which are mounted with silicon pixel sensors
ACTS & TRACCC: three layers with single-sided silicon pixel sensors

	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



The X–Y projection of the VTX

Outline

2 CEPC Geometry in ACTS

Geometry Conversion

① Convert the CEPC geometry file (in DD4hep format) to tgeo format.

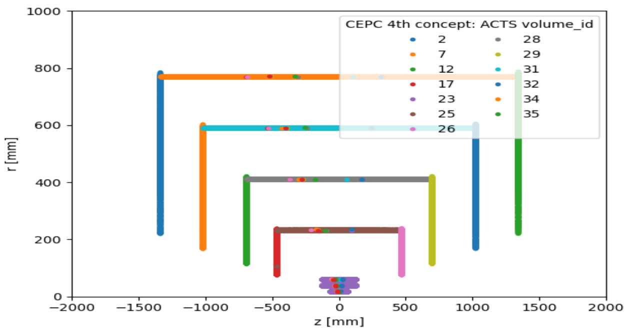
② Write the config file to specify the volumes (VXD, SIT, and FTD) that needs to be generated.

③ Use ACTS' tgeo reader to generate csv files.

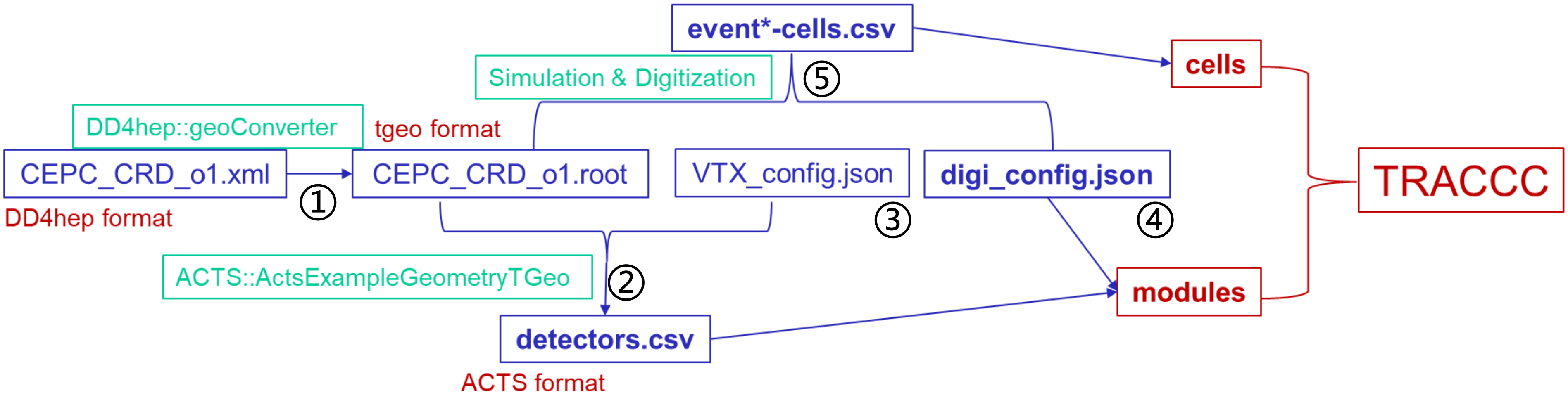
④ Write the digitization config file to provide the segmentation information of each surface.

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.



FATRAS generates hits in z-r plane
VXD + SIT + FTD (layer 0-3)



2 CEPC Geometry in ACTS

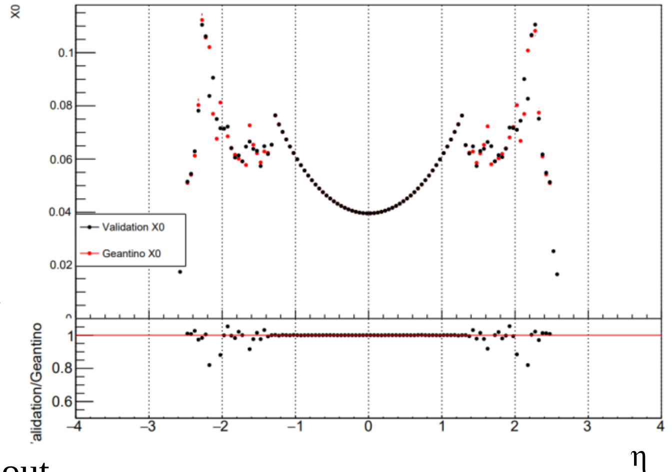
Material mapping

为了提高重建效率，ACTS在重建时使用简化的material。
将原本复杂的material被映射到tracking geometry的不同表面上。
ACTS' material mapping includes 3 steps：

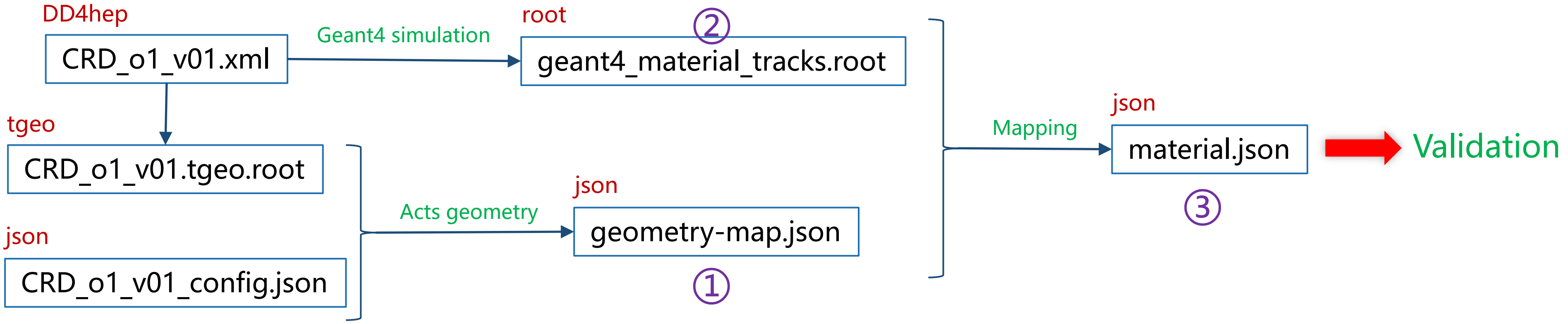
① Create a JSON file:
- configure which surface the material is mapped onto and with which binning.

② Geant4 simulation:
- collect the material inside the detector from the detailed geometry.

③ Mapping:
- all the steps are projected onto the closest surfaces (or volume) and averaged out over many events to create a map.



Validation of Material Mapping
VXD + SIT + FTD (layer 0-3)



2 CEPC Geometry in ACTS

Gid Conversion

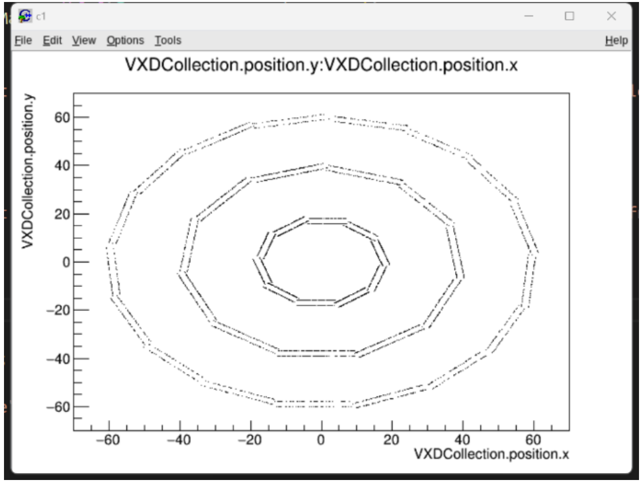
CEPCSW & ACTS 使用不同的geometry id。因此，为了在重建时获取正确的module信息，需要把CEPCSW的cellid转化为acts的gid.

VXD CEPCSW cellid:

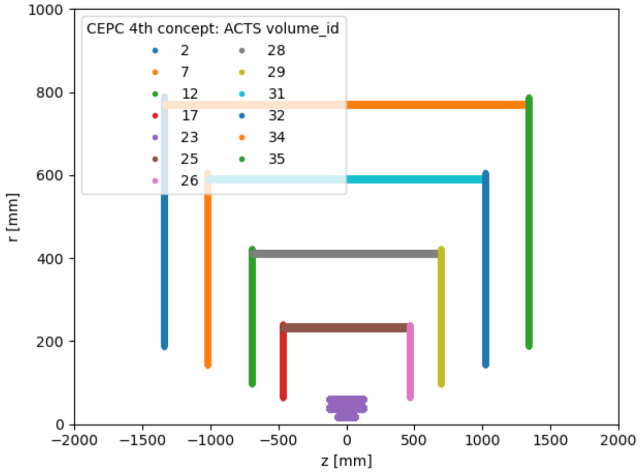
Layer: {0,1,2,3,4,5} # Indicate 6 layers from inside to outside
Module: { L0: 0-9, L1: 0-9,
L2: 0-10, L3: 0-10,
L4: 0-16, L5: 0-16} # Indicate ladders in the φ direction
Sensor: 0
Barrelside: 1 for z > 0 else -1 # one ladders has 2 sensors separated by z

VXD ACTS gid:

Volume: {23}
Boundary: 0
Layer: {2, 4, 6} # adjacent layers are too close, so being treated as the same layers
Approach: 0
Sensitive: {L2: 1-40, L4: 1-44, L6:1-68}
The sensitive counts from z>0 to z<0, then counts in φ direction (the order is same to CEPC), and then counts from inner to outer layers.



Generated hits of CEPC VXD by Geant4



ACTS volume ids of VXD + SIT + FTD

2 CEPC Geometry in ACTS

Gid Conversion

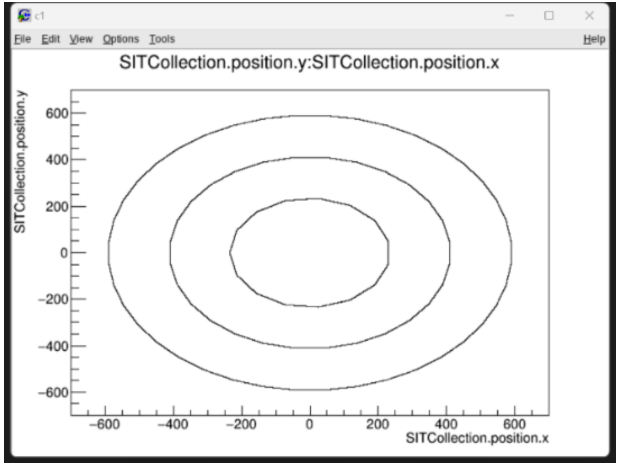
In CEPCSW, the outermost layer of SIT is considered in drift chamber.
We now only consider the inner 3 layers.

SIT CEPCSW cellid:

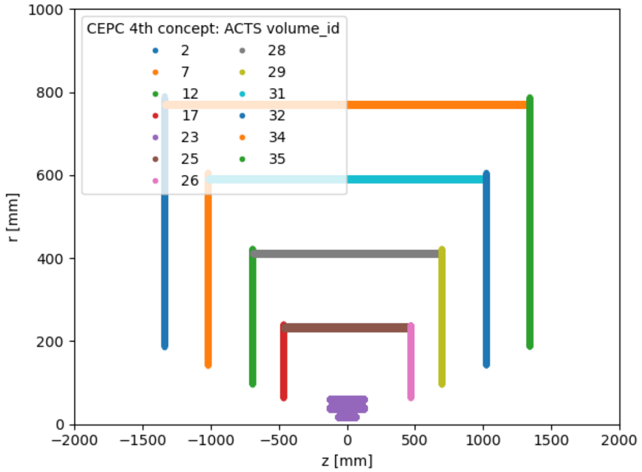
Layer: {0,1,2} # Indicate 3 layers from inside to outside
Module: {L0: 0-14, L1: 0-27, L2: 0-39} # Indicate ladders in the φ direction
Sensor: {L0: 0-9, L1: 0-14, L2:0-21} } # Indicate sensors in the z direction
Barrelside: 0

SIT ACTS gid:

volume: {25, 28, 31} # Indicate 3 layers from inside to outside
Boundary: 0
Layer: 2
Approach: 0
Sensitive: {vol25: 1-150, vol28: 1-420, vol31:1-880}
The sensitive counts in z direction (range z from large to small), then counts in φ direction (the order is same to CEPC), and then counts from inner to outer layers.



Generated hits of CEPC SIT by Geant4



ACTS volume ids of VXD + SIT + FTD

2 CEPC Geometry in ACTS

```
// uint64_t VXD_acts_volume_id = 23;
// std::vector<uint64_t> VXD_NumOfSensors_per_layer{20, 22, 34};
uint64_t barrel_sign = (m_barrelside == 1) ? 1 : 2;
uint64_t acts_volume = VXD_acts_volume_id;
uint64_t acts_boundary = 0;
uint64_t acts_layer = 2 * (m_layer >> 1) + 2;
uint64_t acts_approach = 0;
uint64_t acts_sensitive = VXD_NumOfSensors_per_layer[m_layer >> 1] * (m_layer & 1)
                          + 2 * m_module + barrel_sign;

// set acts geometry identifier
Acts::GeometryIdentifier moduleGeoId;
moduleGeoId.setVolume(acts_volume);
moduleGeoId.setBoundary(acts_boundary);
moduleGeoId.setLayer(acts_layer);
moduleGeoId.setApproach(acts_approach);
moduleGeoId.setSensitive(acts_sensitive);
```

Converter for VXD gid

```
// std::vector<uint64_t> SIT_acts_volume_ids{25, 28, 31};
// std::vector<uint64_t> SIT_NumOfSensors_per_module{10, 15, 22};
uint64_t acts_volume = SIT_acts_volume_ids[m_layer];
uint64_t acts_boundary = 0;
uint64_t acts_layer = 2;
uint64_t acts_approach = 0;
uint64_t acts_sensitive = SIT_NumOfSensors_per_module[m_layer] * m_module
                          + m_sensor + 1;

// set acts geometry identifier
Acts::GeometryIdentifier moduleGeoId;
moduleGeoId.setVolume(acts_volume);
moduleGeoId.setBoundary(acts_boundary);
moduleGeoId.setLayer(acts_layer);
moduleGeoId.setApproach(acts_approach);
moduleGeoId.setSensitive(acts_sensitive);
```

Converter for SIT gid

Validation of Gid Conversion

1. We get the global & local position of EDM4hep::TrackerHit.
2. Give the local position & converted Gid to Acts::Surface, if the gid conversion is correct, Acts::Surface can get the correct global position.

The conversion has been validated.

```
auto simcellid = simhit.getCellID();
Acts::GeometryIdentifier moduleGeoId = getVTXGid(simcellid);
const Acts::Vector2 local_position{loc0, loc1};
const auto& surface = trackingGeometry->findSurface(moduleGeoId);
auto global_postion = surface->localToGlobal(geoContext, local_position, globalFakeMom);
info() << "VXD global position(x,y,z): " << simhit.getPosition()[0] << ", "
                                         << simhit.getPosition()[1] << ", "
                                         << simhit.getPosition()[2]
debug() << "converted acts position(x,y,z): " << global_postion[0] << ", "
                                               << global_postion[1] << ", "
                                               << global_postion[2] << endmsg;
```

Code to check Gid Conversion

Outline

3 Seeding algorithm for CEPC

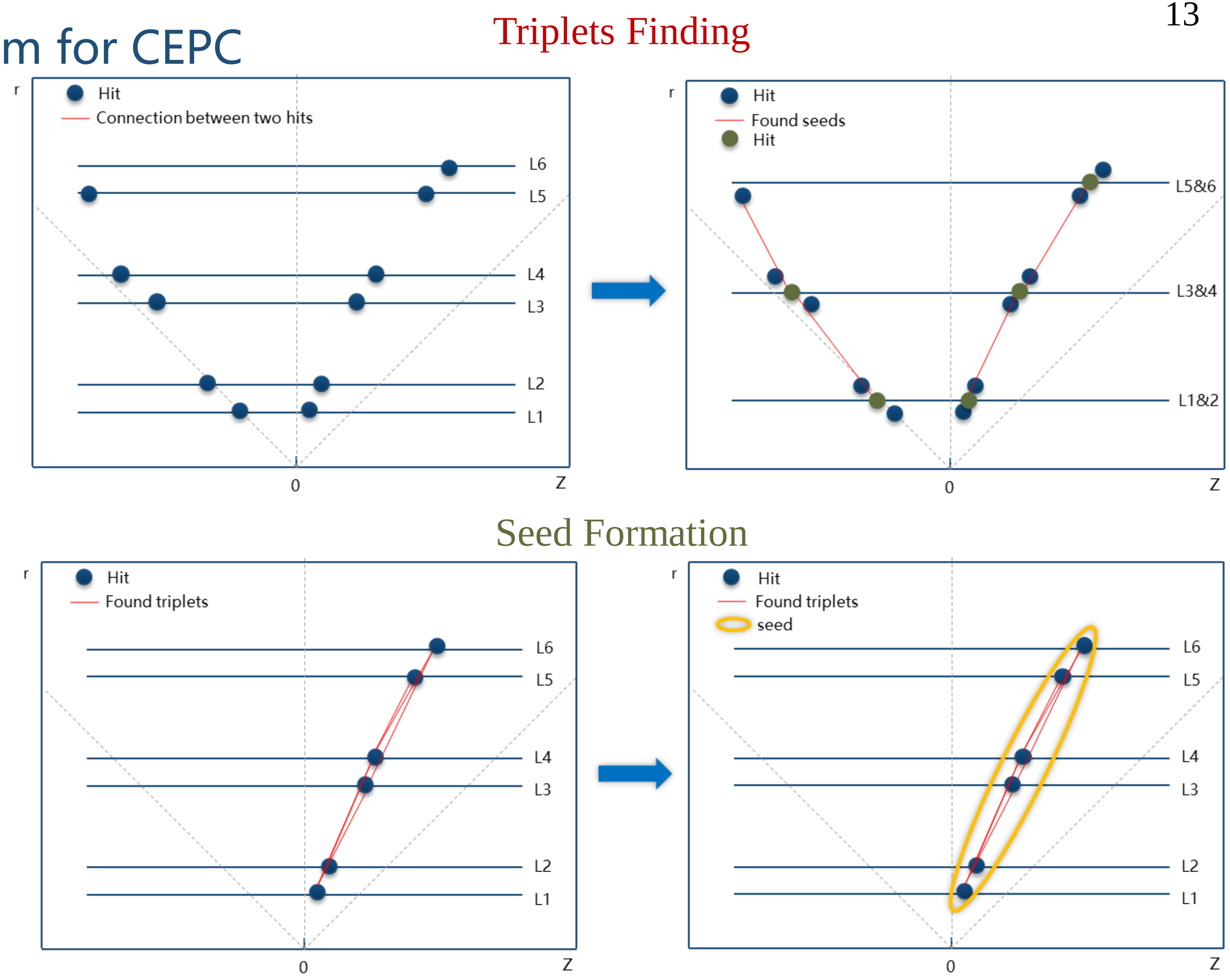
6-layers seeds finding

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

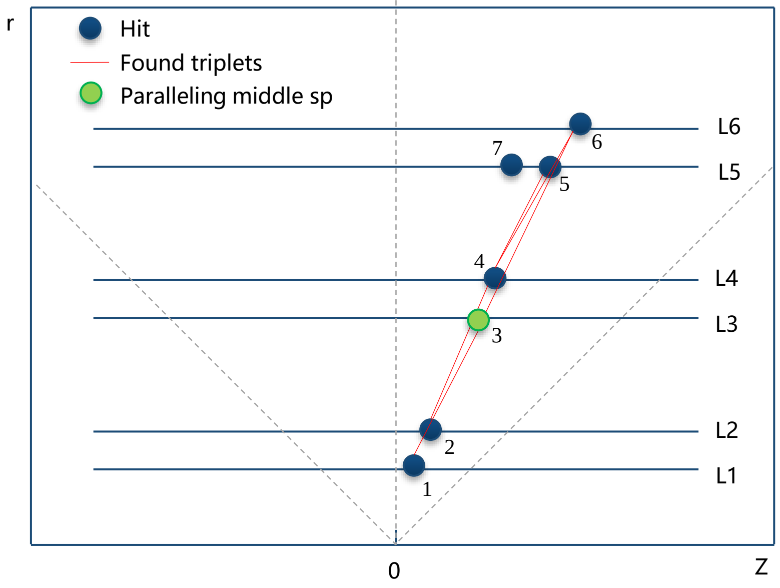
1. Triplets Finding:
Before seeding, treating adjacent layers as one layer, and considering nearby space-points in two layers as one-space point.

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.



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()

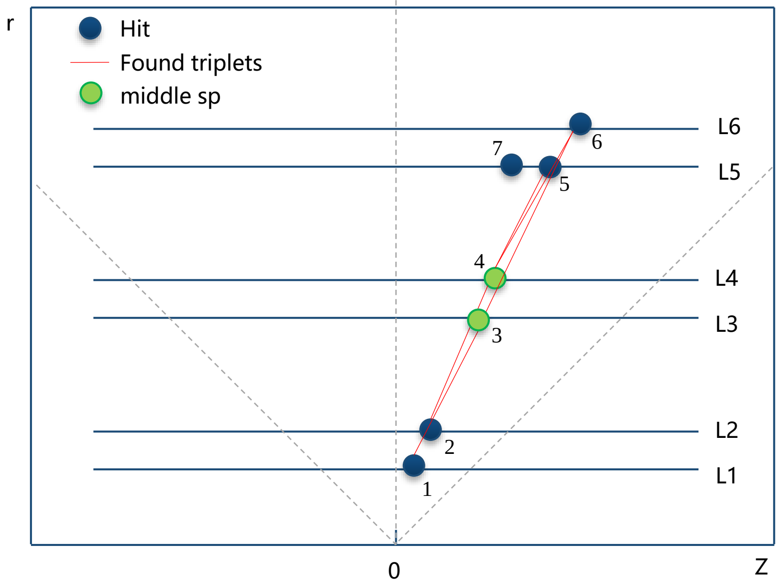
{5, 6} | {7, 6}
5.radius() < 6.radius()

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()

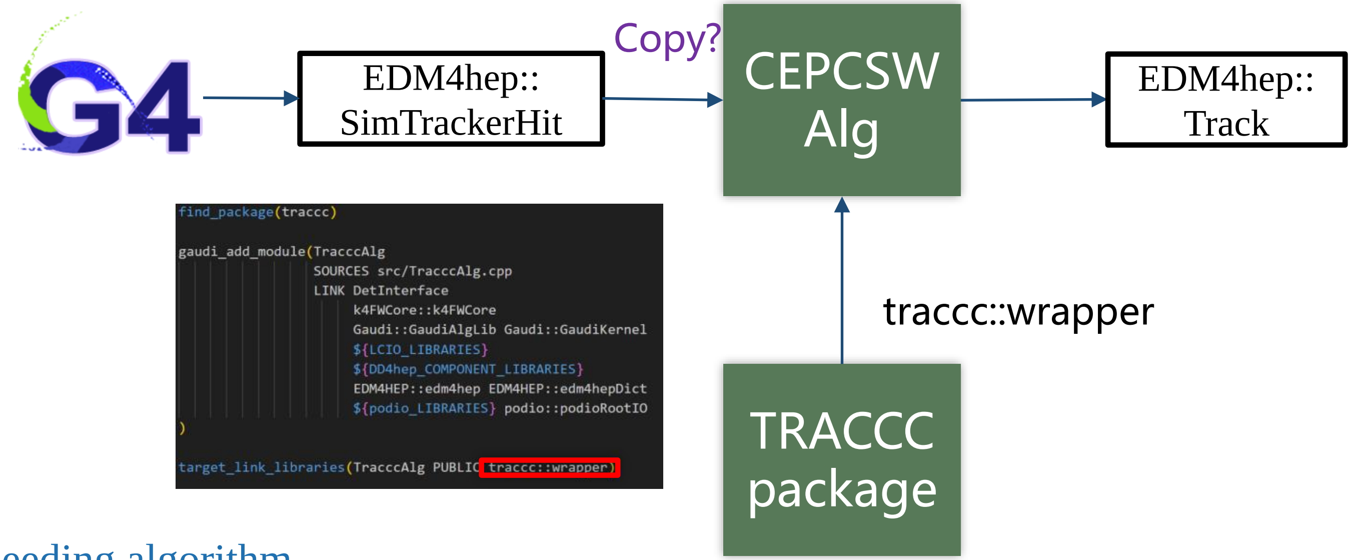
6-layers seeds finding: Seed Formation step in CPU

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




```
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

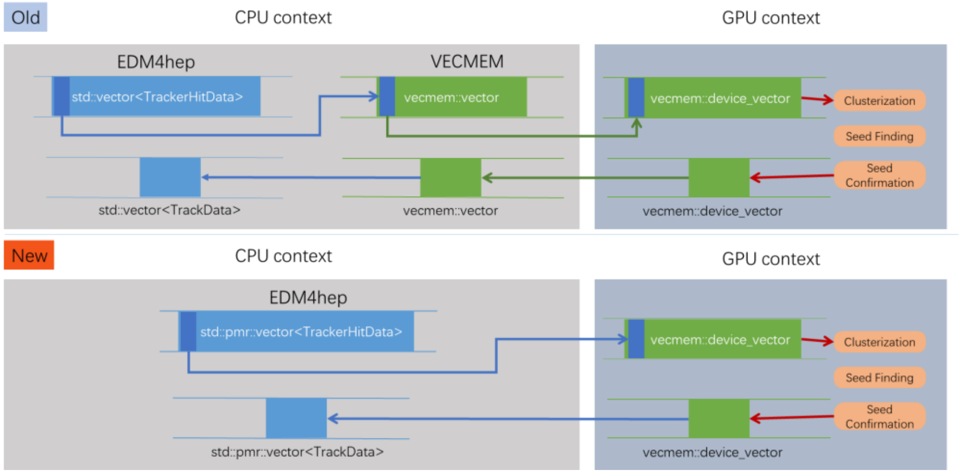
- Write a wrapper to wrap the seeding functions that CEPCSW needed.
- Calling the TRACCC package in CEPCSW alg.
- Pull request: https://github.com/cepc/CEPCSW/pull/270

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

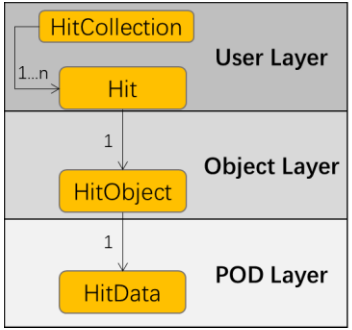


Modify the EDM4hep

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.

```yaml
#------------- 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

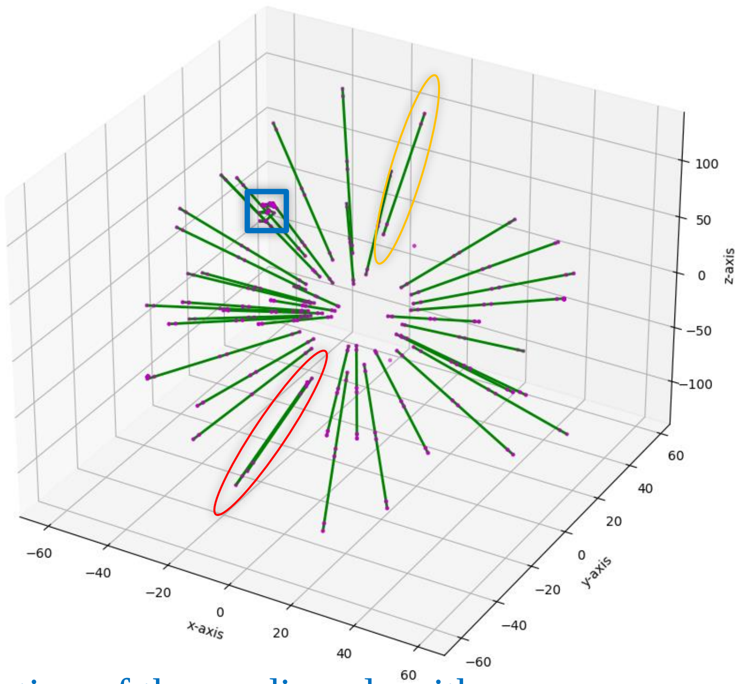
```
Running Seeding on device: Quadro RTX 8000
Initializing ...
EventLoopMgr      WARNING Unable to locate service "EventSelector"
EventLoopMgr      WARNING No events will be processed from external input.
ApplicationMgr       INFO Application Manager Initialized successfully
ApplicationMgr       INFO 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

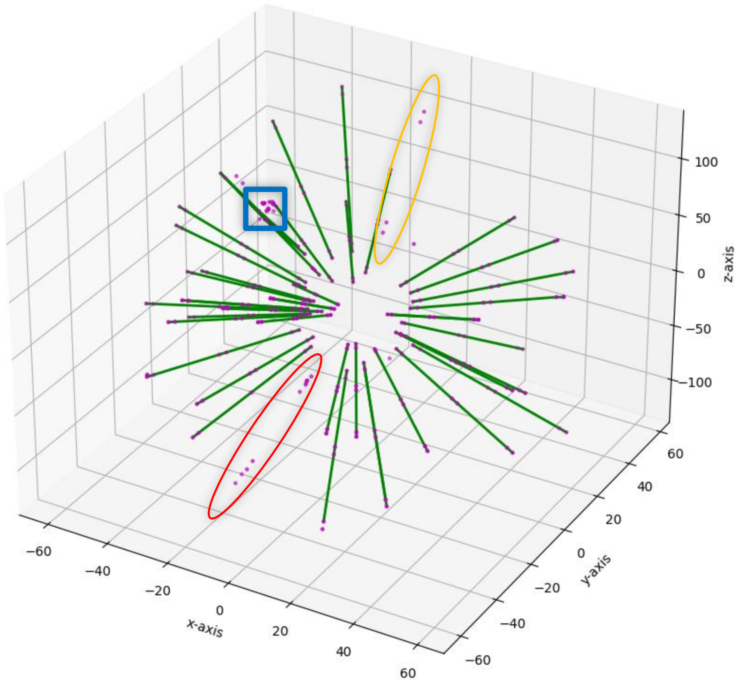
- The address of pmr::vector does not changed.
- No data copy occurs.

4 Integration of TRACCC with CEPCSW

Geant4 simulation (1 event, 50 tracks)



TRACCC reconstruction



Verification of the seeding algorithm

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)

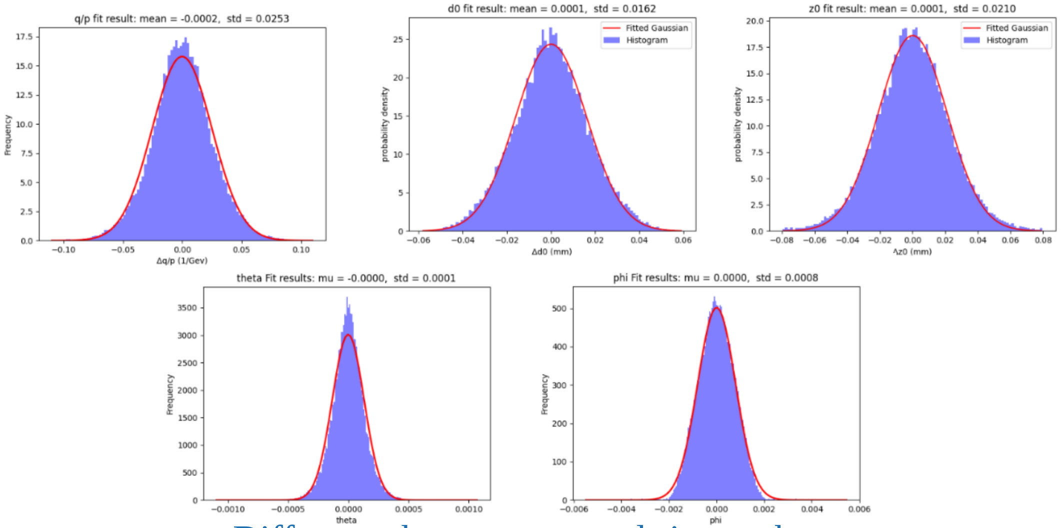
*The blue part G4 produced secondary particles (e-).

tracks are found correctly!

4 Integration of TRACCC with CEPCSW

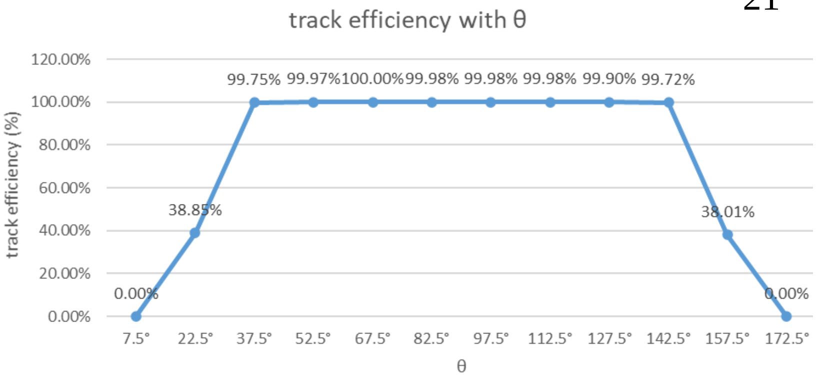
Seeding efficiency evaluation

Particle: mu-   Energy: 5 Gev



Difference between rec and sim track param

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



Track efficiency with θ

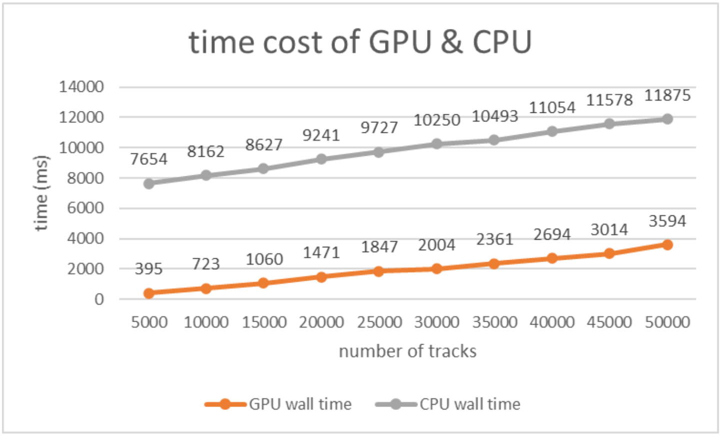
$$track\ eff = \frac{N_{rec\ tracks}}{N_{total\ tracks}}$$

Material effects track efficiency at low θ angle.

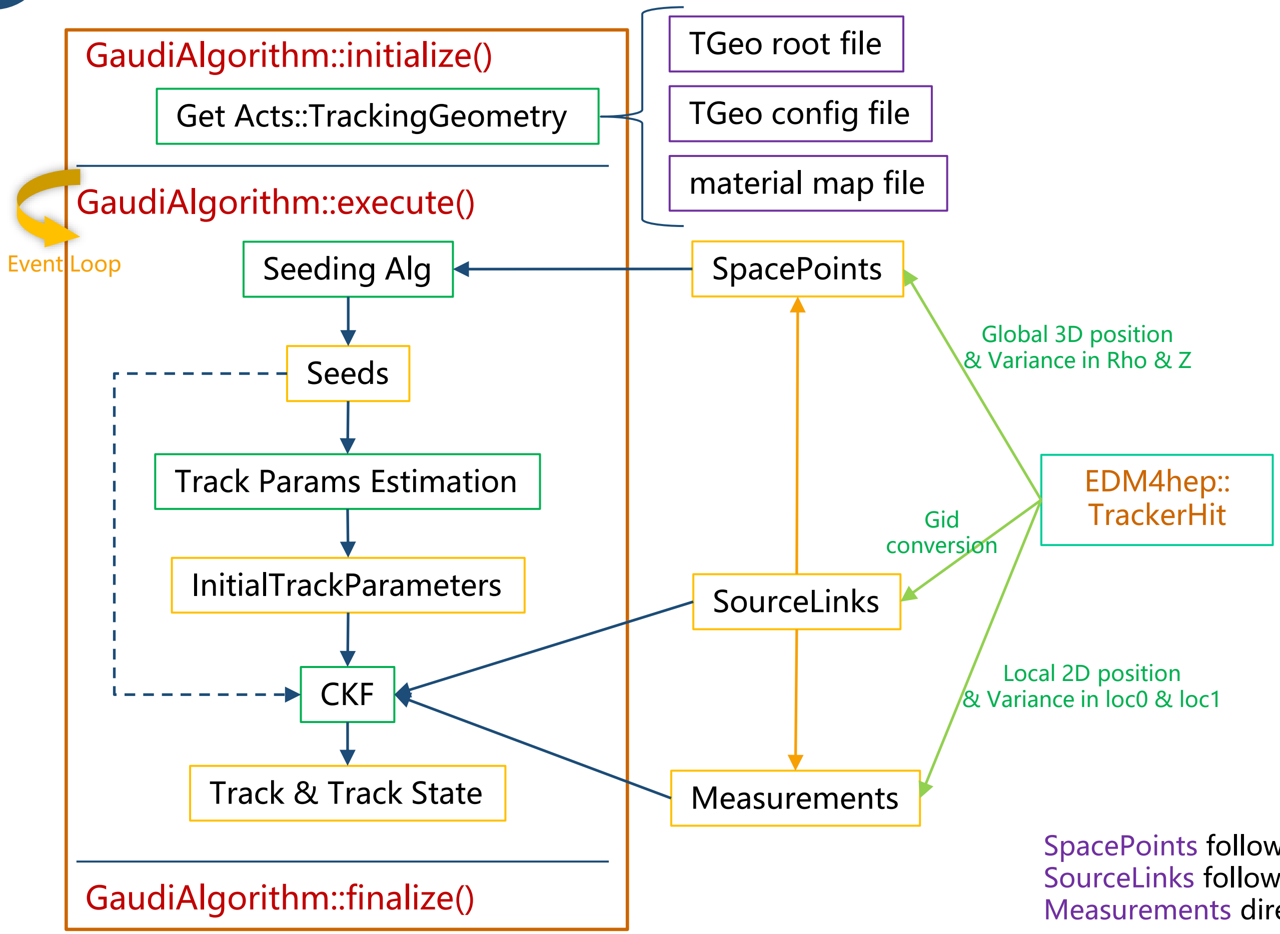
Computing evaluation of TRACCC seeding

Run TRACCC in heterogeneous device:

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



4 Integration of ACTS with CEPCSW



Gaudi Algorithm View

Initialize: read the CEPC geometry, get the Acts::TrackingGeometry.

Execute: read the EDM4hep:: TrackerHit of current event, generate SpacePoints & measurements, and store the connection between ModuleGid & meas idx into SourceLinks.

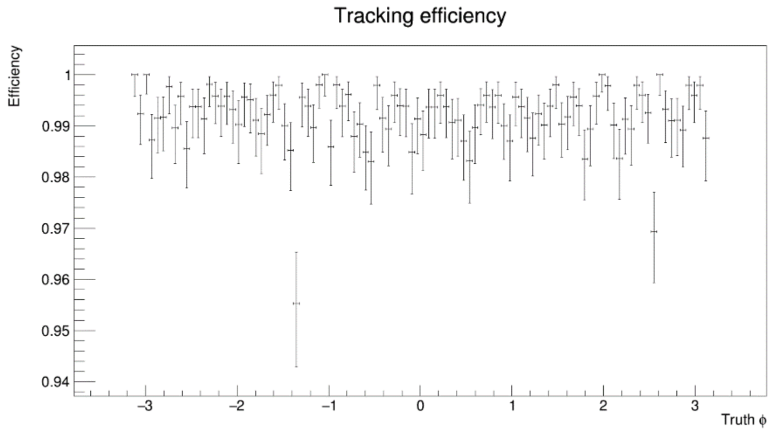
Follow the Acts::Example, we write the Seeding Alg & Track Params Estimation & CKF using Acts tools.
And finally get the track & track state stored in Acts::TrackContainer.

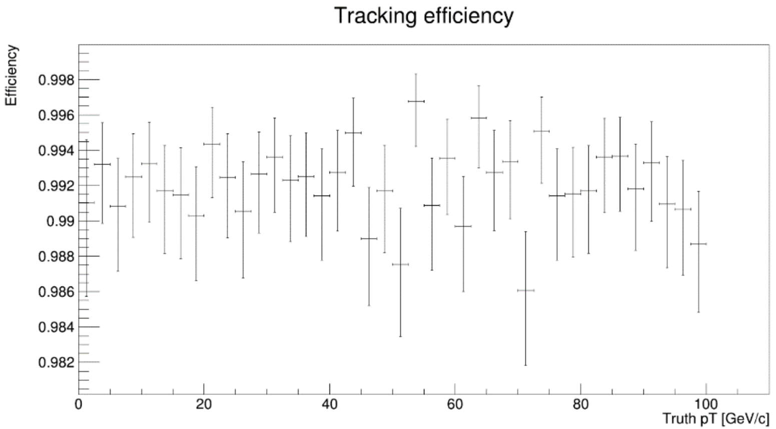
SpacePoints follows ActsExamples::SimSpacePoint
SourceLinks follows ActsExamples::IndexSourceLink
Measurements directly use Acts::BoundVariantMeasurement

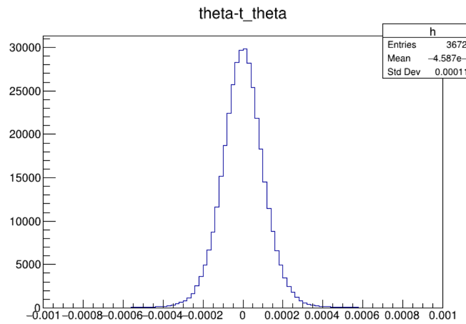
4 Integration of ACTS with CEPCSW

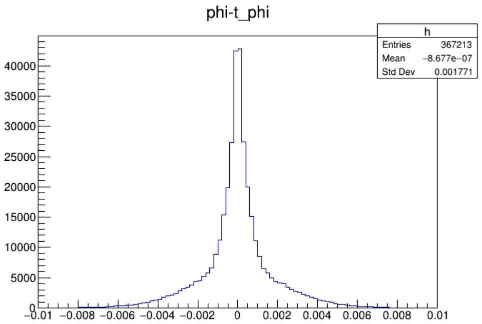
ACTS tracking efficiency evaluation

Particle: mu-   Energy: 1:100Gev









Distribution of Δφ & Δθ

Tracking efficiency with Φ & pT

Summary & future work

Application of TRACCC seeding to the CEPC vertex detector

Summary

- Implement the CEPC detector geometry 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
- Use one common memory for both EDM4hep and VecMem to avoid the overhead from data copy
- Integrate the ACTS & TRACCC in CEPCSW

Future work

- Further analysis of seeding efficiency & computing performance
- Comparison between ACTS reconstruction & origin reconstruction algorithm.

Thank you

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

15th Aug 2024

Backup

2 CEPC Geometry in ACTS

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

3 Integration of ACTS with CEPCSW

Get local position from EDM4hep

EDM4hep::TrackerHit do not directly provide the local position.

We set the segmentations in the cellid to get the local position.

The grid size of both VTX & SIT is set to 25um.

```
auto cellid = hit.getCellID();
double acts_loc0 = sit_decoder->get(cellid, "y") * grid_size;
double acts_loc1 = sit_decoder->get(cellid, "z") * grid_size;
```

```
// create and store the measurement
const std::array<Acts::BoundIndices, 2> indices{Acts::BoundIndices::eBoundLoc0, Acts::BoundIndices::eBoundLoc1};
Acts::ActsVector<2> par{acts_loc0, acts_loc1};
Acts::ActsSquareMatrix<2> cov = Acts::ActsSquareMatrix<2>::Zero();
measurements.emplace_back(Acts::Measurement<Acts::BoundIndices, 2>(std::move(sl), indices, par, cov));
```

Get and store the local position in measurement

```
<readout name="VXDCollection">
  <!-- <id>system:5,side:-2,layer:9,module:8,sensor:8,barrelside:-2</id> -->
  <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>
```

VXD

Module size (x-y × z direction)

Layer 0, 1:
- 11mm*62.5mm
- 880 * 5000 (25um/bin)

Layer 2, 3, 4, 5:
- 22mm*125mm
- 880 * 5000 (25um/bin)

```
<readout name="SITCollection">
  <!-- <id>system:5,side:-2,layer:9,module:8,sensor:8,barrelside:-2</id> -->
  <segmentation type="CartesianGridYZ" grid_size_y="25*um" grid_size_z="25*um"/>
  <id>system:5,side:-2,layer:9,module:8,sensor:8,barrelside:-2,y:-13,z:-13</id>
</readout>
```

SIT

Module size (x-y × z direction)

Layer 0, 1, 2:
- 97.55mm*91.85mm
- 3902 * 3674 (25um/bin)