A Common Tracking Software(ACTS) and the preliminary integration to CEPC

Hongbo Zhu, Gang Li, Yubo Han , Jin Zhang from IHEP

On behave of the CEPC-ACTS group

Nanjing University, 2019-5-30

Outline

- Introduction to ACTS
- ACTS: General Models and Strategies
- Our contributions
- Preliminary CEPC Integration
- Conclusions and outlooks

A Common Tracking Software

- What we have: Experience from LHC tracking Run-1/2
  - Well tested high performance Codes
  - ATLAS common tracking software – Common tracking geometry

  -> ACTS

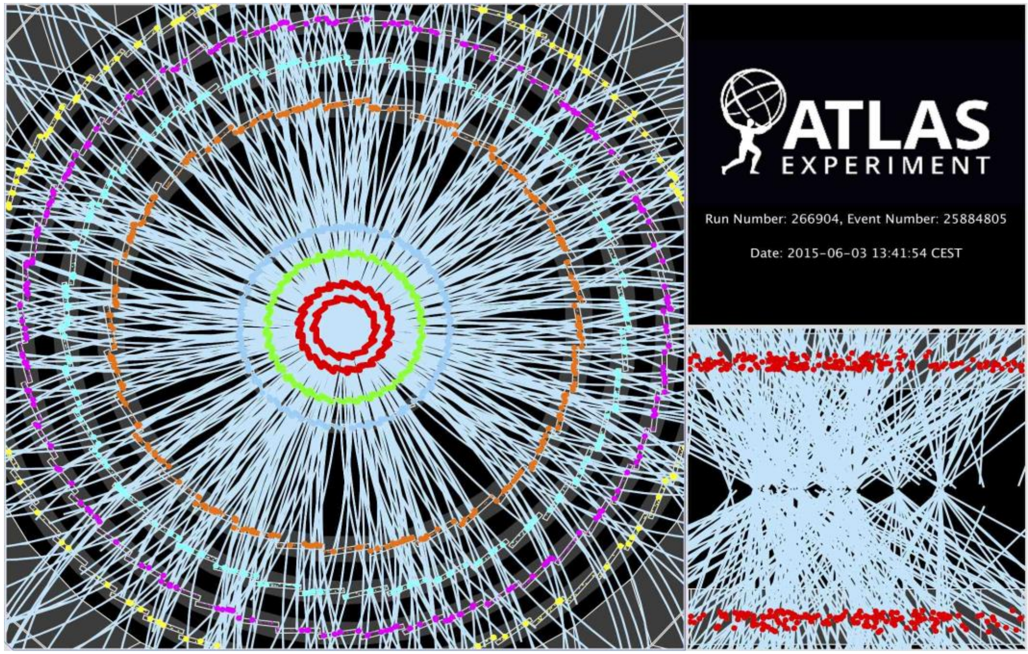
- Challenges
  - Increasingly growing pileup -> software campaigns
    - 40->1000 HL-LHC, FCC…
  - Tough cases from AthenaMT : 4 millions lines of C++!
  - Maintenance of Athena (ATLAS code)

A Common Tracking Software

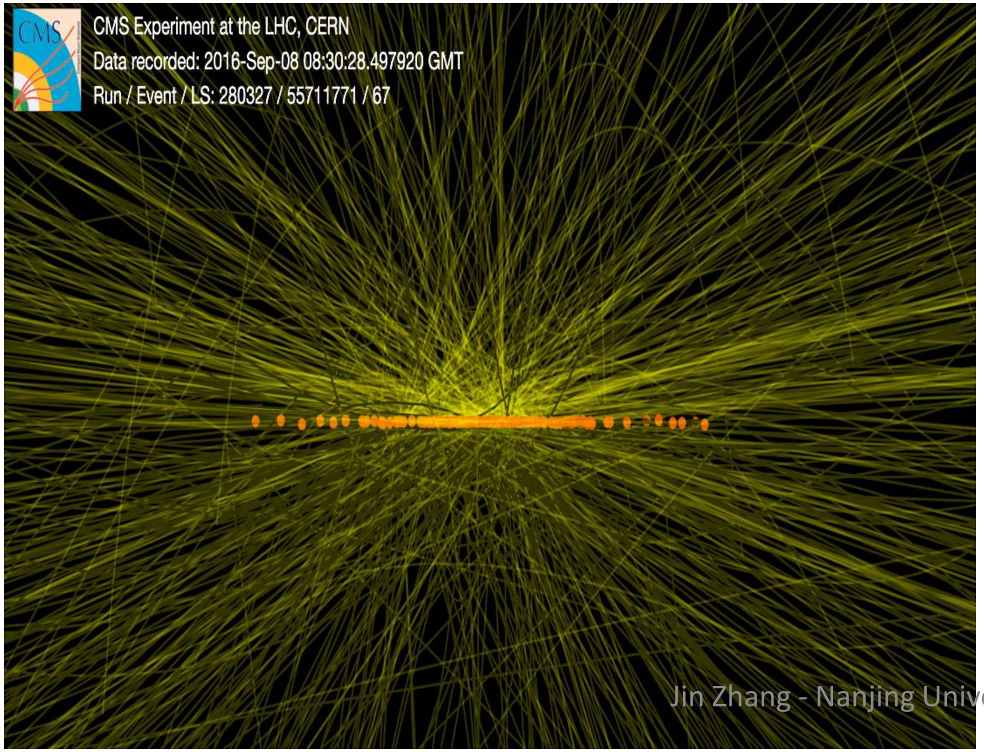
- Goal:
  - Encapsulate the current code from ATLAS(currently), or other experiment..
    - C++ 17 -> keep updating
    - Modernized code
  - Thread safety/long vectorization for modern CPU structure
  - General tracking models without detector specific design
  - Bring experience for the future High Energy Physics tracking community …
- Dependencies required from the acts-core
  - boost , Eigen, and a high C++ compiler..

The main part of code was taken from ATLAS. ACTS are seeking for experiences and validations from other experiments e.g. CMS, FCC,CEPC,BELL2, PANDA…

The challenge of computing resource in the future



A high pile up 13TeV LHC collision (86 vertices reconstructed)



A 13 TeV LHC collision (17 vertices)

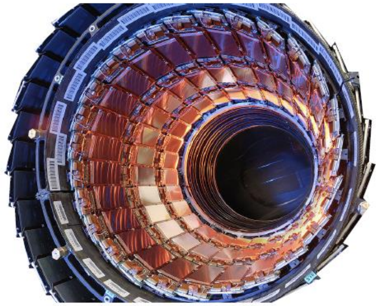
	LHC Run-1	LHC Run-2	LHC Run-4	FCC ~?
muon	21	40	150-200	1000
Tracks	~280	~600	~7-10k	~100k

For CEPC : our motivations

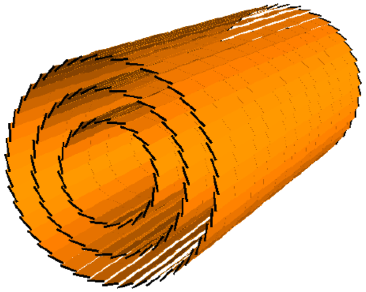
- Tracking is critical for detector studies benchmark
- *Contribute* - take part in the software development - gain experience
- *Bring back* - take it as a choice of tracking software as well as a performance validation tool for the CEPC detector layout design

Tracking Geometry

- First, we should have some tools for the tracking geometry that you can



real



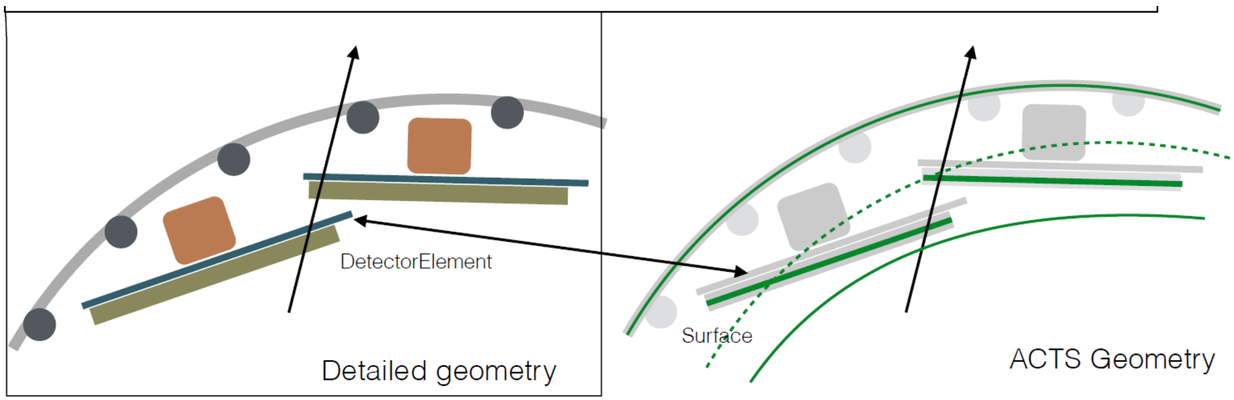
ideal

- General idea to construct a detector for tracking is to
  - Build them as light as possible <-> keep the measurement reliable
  - The detection module should be kept full detailed
  - The material should be simplified

Tracking Geometry = Simplified geometry + approximated material setup

Basic element - Surface

- Each ACTS Geometry object is based on Surface or extended from Surface
- Surface keeps the full details of the sensitive detector element
- Transform – position/rotation matrix

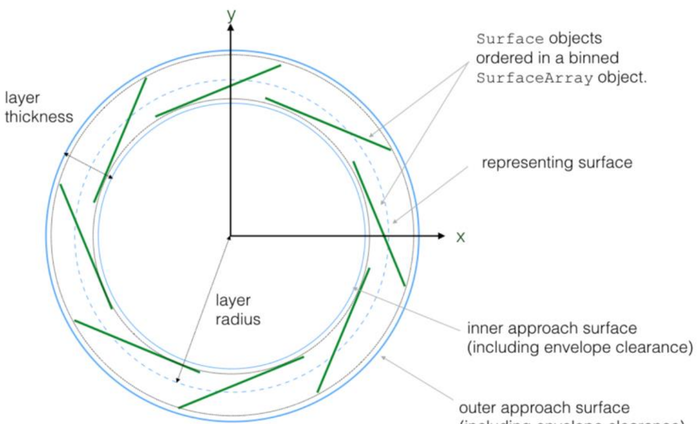


- Currently five Surface Type in ACTS

Surface Type	Local Coordinates	Bound Types available
ConeSurface	[rphi, z]	ConeBounds
CylinderSurface	[r, phi]	CylinderBounds
DiscSurface	[r, phi]	RadialBounds , DiscTrapezoidalBounds
PlaneSurface	[x, y]	RectangleBounds , TrapezoidalBounds , TriangleBounds , InfiniteBounds , EllipseBounds
PerigeeSurface , StrawSurface	[d, z]	CylinderBounds

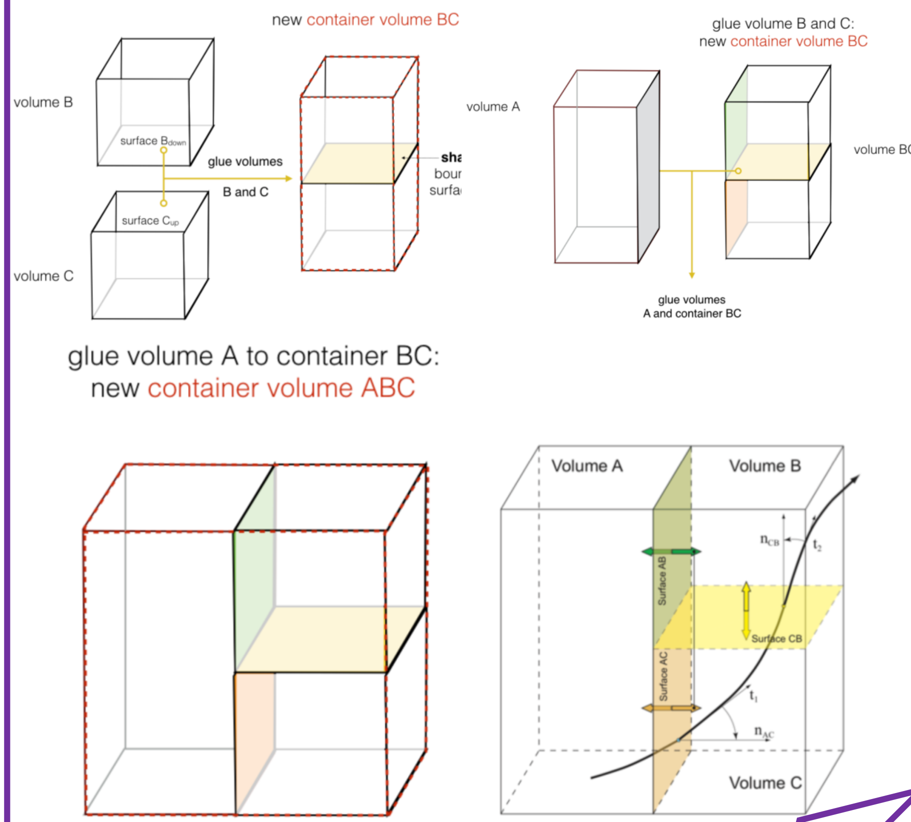
The full connected tracking Geometry

- Layers contains surface array
- Volume contains layer array
- Layers point to each neighbor layers
- Volume contained in other volume
- Boundary surface of volume point to outside/inside



x-y view of a CylinderLayer (with planar detection elements)

Glue and the full connected tracking geometry



attached boundary surfaces A_{right} to B_{left} and C_{left}

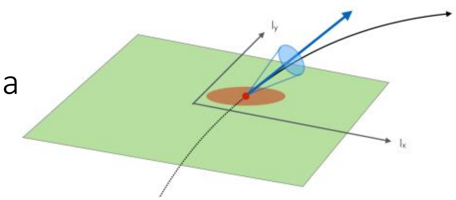
resulting navigation through the boundary portals

Tracking geometry created only one time – other technique to with condition data

Track Parameters

Dimension of the parameterization (time dimension will be included)

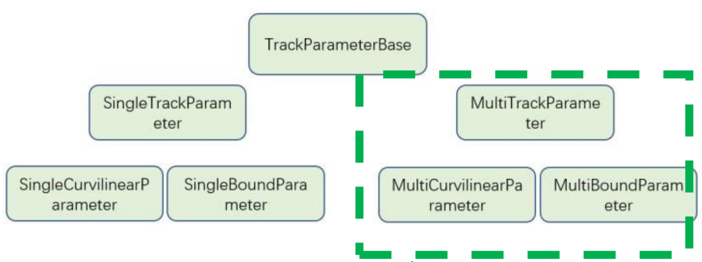
Track bound to a planer surface



Global Par : 7
- vector3D pos
- vector3D dir
- momentum

When bound to a surface

BoundPar : 5
- eLOC_0
- eLOC_1
- ePHI= 2
- eTHETA=3
- eQOP=4



For Gaussian Sum Filter
Behave like single component in Navigation, propagate as multi components in between

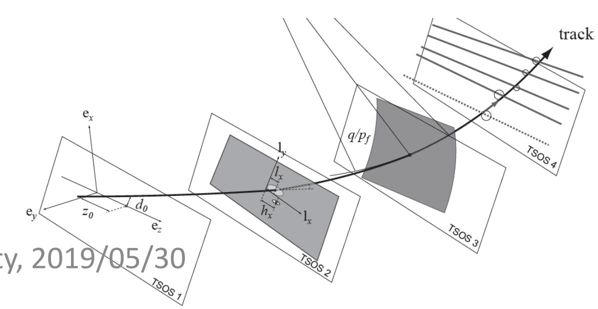
Measurement

no detector-restrict type of measurement - take all measurement as identical in a tracking/fitting algorithm

Measurement is allowed from 1 dimension to 5 dimension – maximum of **BoundPar**

Measurements are interpreted as std::variant in compiling, call at fitting alg with std::visit

(Pixel cluster(2D), Strip cluster(1D), Segment(4D)…



Propagation

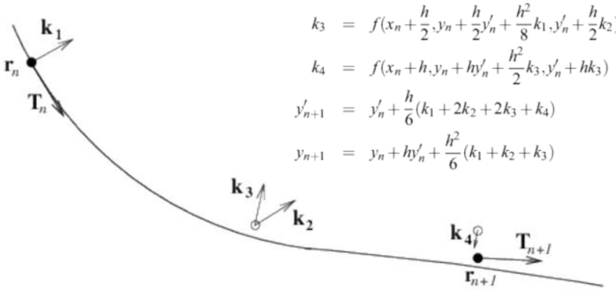
A general issue dealing with **particle motion**

In the magnetic field

Stepper

Runge-Kutta-Nystroem method

$$
\begin{aligned}
k_1 &= f(x_n, y_n, y'_n) \\
k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}y'_n + \frac{h^2}{8}k_1, y'_n + \frac{h}{2}k_1\right) \\
k_3 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}y'_n + \frac{h^2}{8}k_1, y'_n + \frac{h}{2}k_2\right) \\
k_4 &= f\left(x_n + h, y_n + hy'_n + \frac{h^2}{2}k_3, y'_n + hk_3\right) \\
y'_{n+1} &= y'_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\
y_{n+1} &= y_n + hy'_n + \frac{h^2}{6}(k_1 + k_2 + k_3)
\end{aligned}
$$



Track calculates and returns the error

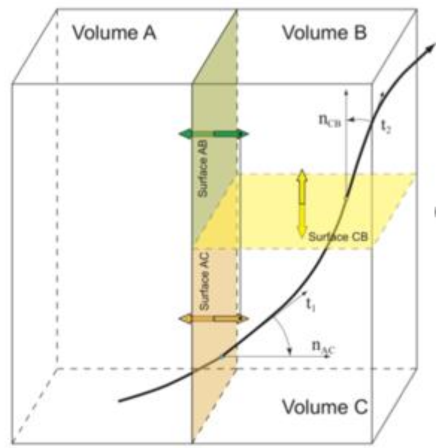
Adaptive of step size according to error

Vacuum/dense Environment(Calo) is chosen with a voting strategy

Reliable accuracy

integrated to the tracking geometry

Navigation



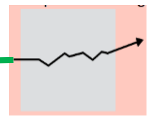
our fully connected tracking geometry

Come in convenient!

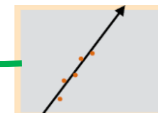
On surface

material effect, measurement update

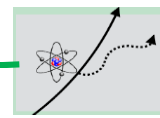
Actors



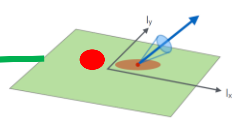
Multiple scattering



Ionization loss



Bremsstrahlung energy loss



Measurement update

......

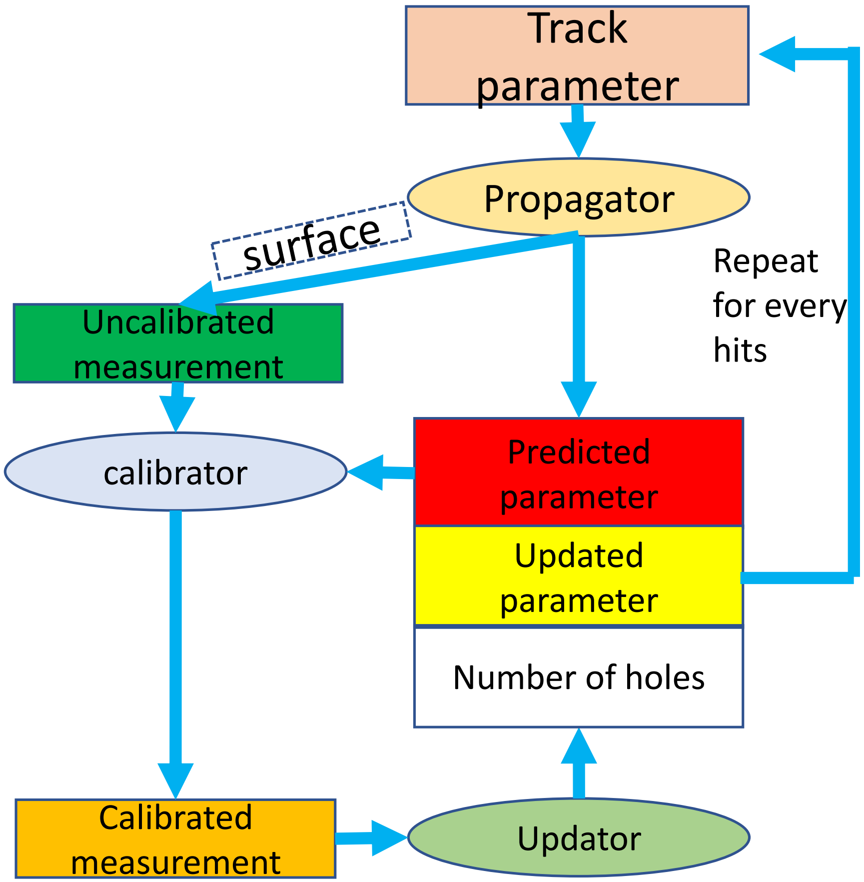
Aborters

- End of the world
- Path limit
- Target surface
- ......

Fitting

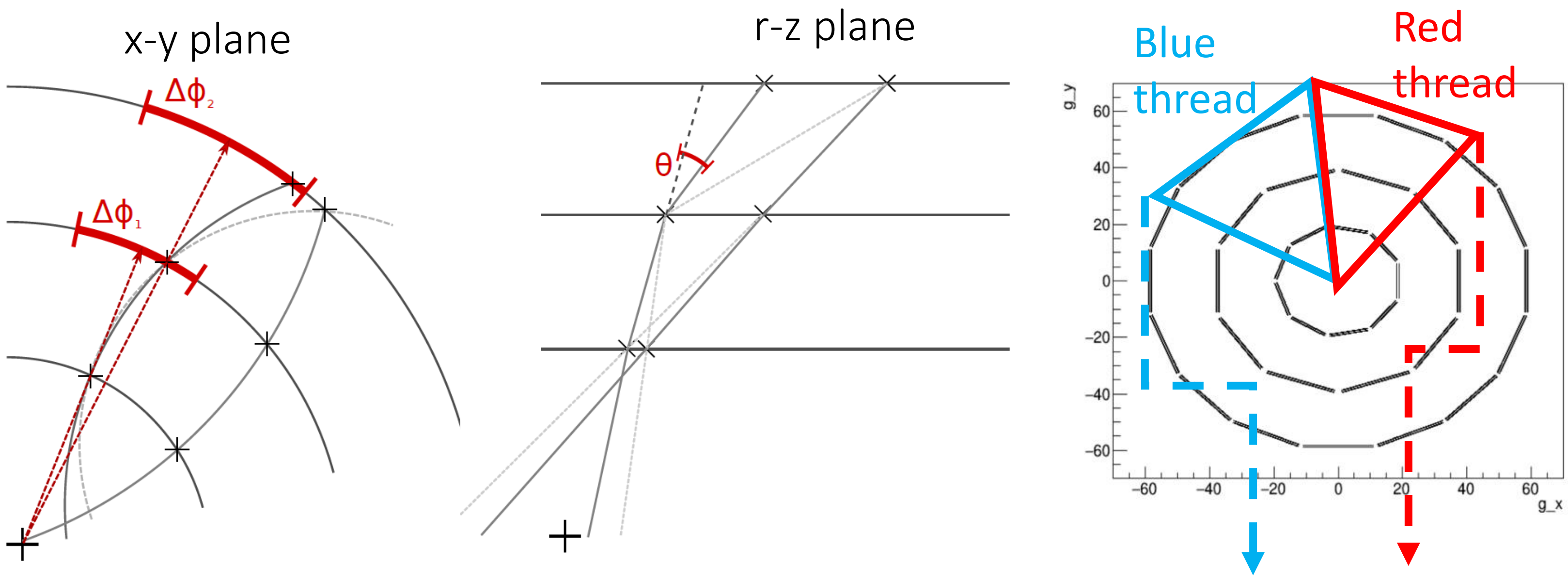
- The Kalman Fitter model was in the acts-core
  - Use Propagator as the prediction
  - Filter/Smoother both implemented
  - More details need to be implemented as a tool of CKF
- Gaussian sum filter task is being developed
  - Models are prepared and are on the merge request process
- DAF method will be implemented in the future

A brief Kalman work flow



Seeding

- A combinatorial seed finder – select 3-hit seeds
- Input of seed-finder : 3D Space Point
- Output : all combinations of 3-hit seeds
- The config option is set by users, e.g. minPt, measurement region

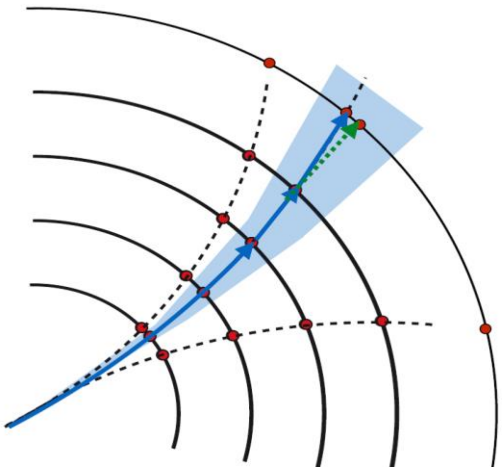


multi thread seed finder in different regions intra event is allowed

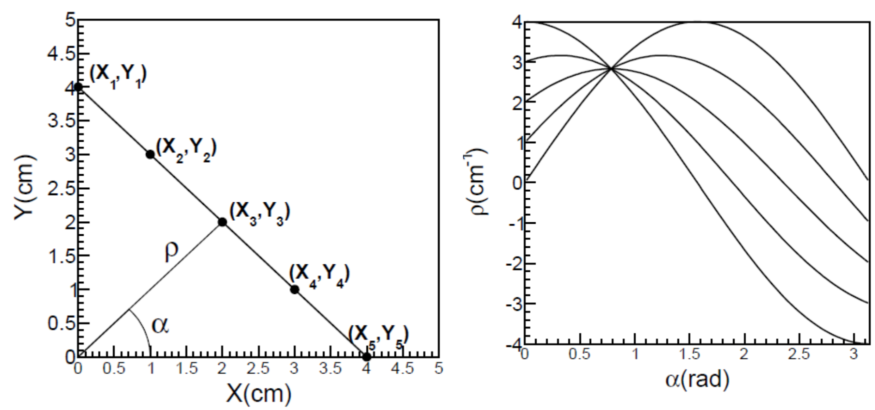
Basic requirement of Track Finding

- The track finding model is not implemented
- A track following method and a global method in expected

Combinatorial Kalman Filter



Hough transform



(a) X-Y plane

(b) parameter space

Other Models

- Vertex – Basic Vertex Finding and Fitting algorithms ready

□ Calibration : specific for detectors

  - A void Calibration in ACTS, you can implement your own detector version

□ Alignment : many nice algorithms online that are public

- The Calibration and Alignment and Magnetic field data (Condition data) are introduced with the Context object

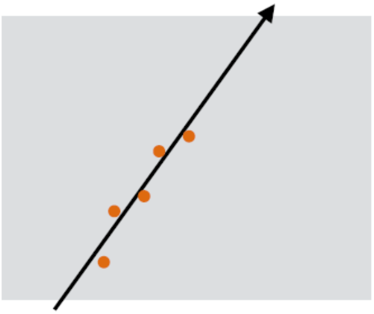
```
/// PropagatorOptions with context
///
/// @param gctx The goemetry context for this fit
/// @param mctx The magnetic context for this fit
/// @param cctx The calibration context for this fit
/// @param rSurface The reference surface for the fit to be expressed at
KalmanFitterOptions(std::reference_wrapper<const GeometryContext> gctx,
                    std::reference_wrapper<const MagneticFieldContext> mctx,
                    std::reference_wrapper<const CalibrationContext> cctx,
                    const Surface* rSurface = nullptr)
```

ACTS General Models and strategies

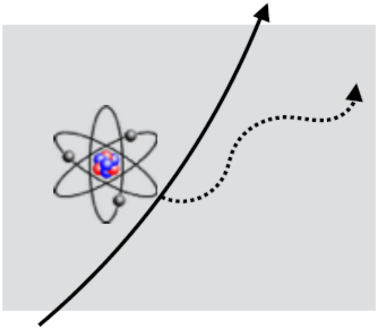
Gaussian Sum Filter

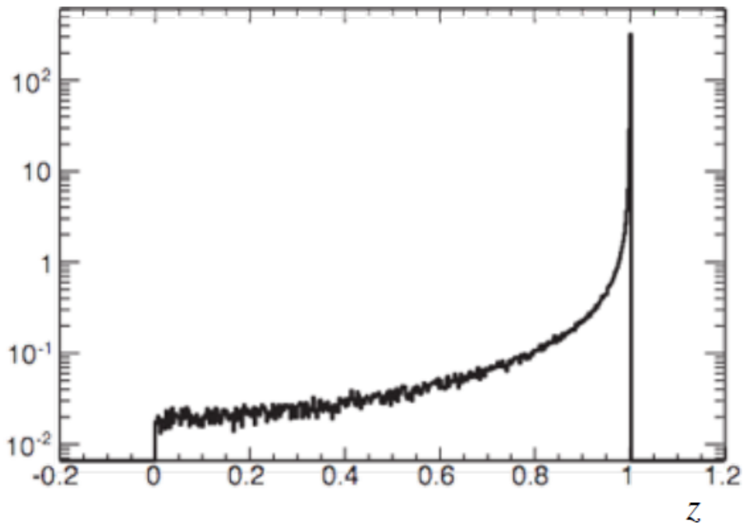
Kalman Filter : linearized filter allows all experiment noise is gaussian distributed

- Measurement errors – usually can be controlled
- Multiple scattering – a small gaussian tails
- Ionization loss – Landau distributed, fortunately dE<<E



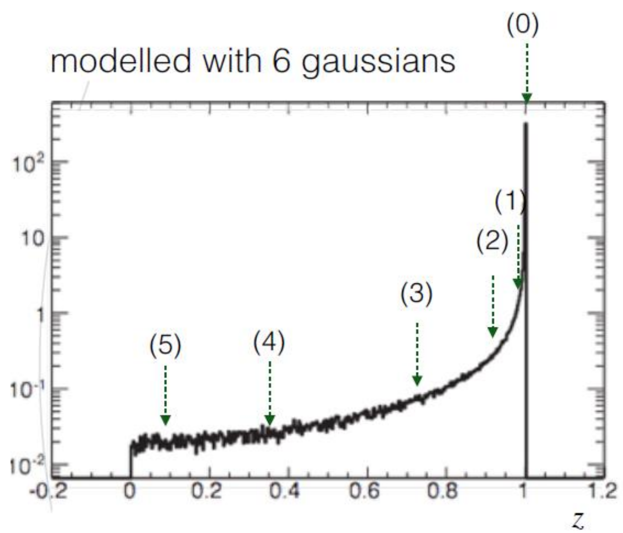
The electron reconstruction : The energy loss is a bremsstrahlung effect -> strongly non gaussian

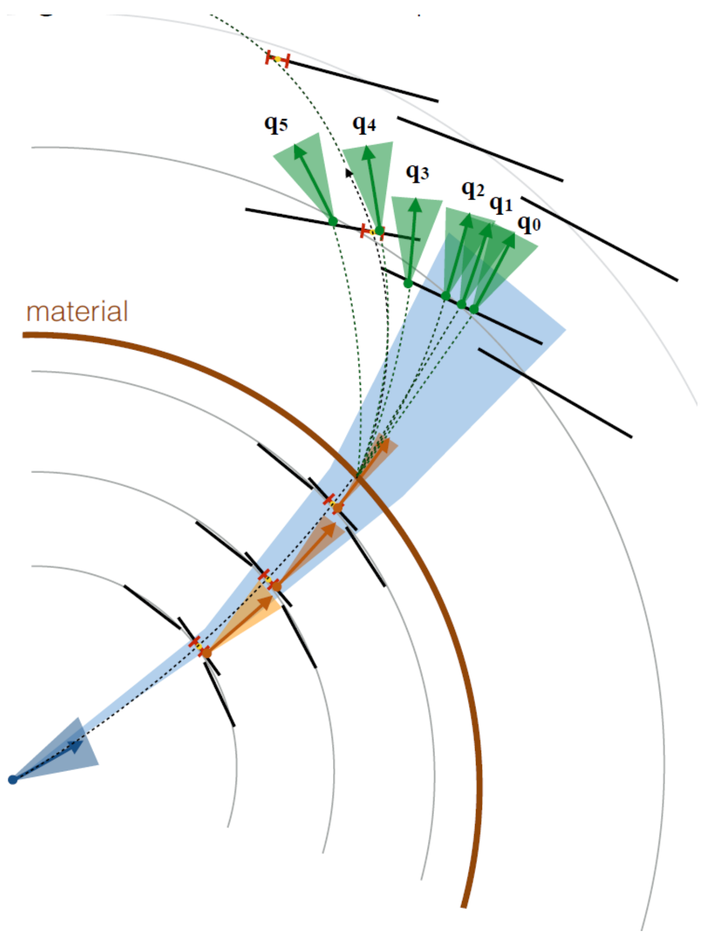




Gaussian Sum Filter

- The bremsstralung energy loss distribution can be approximated as a weighted sum of gaussian component
- Each component behaves like a Kalman component





One component splits into 6 components

The (mean/var/weight of) each component from ATLAS

GSF strategy

- In Propagate
  - Take the combination of components – behave like single in Navigation
  - Each component takes charge its own path
- Energy loss Effect
  - Bethe-Heitler – Currently take the ATLAS parameters to construct 6 components in each material effect
- Component Reduction
  - Iteration to combine closet components to a maximum number with the MultiTrackParameters class
- Measurement update part is similar with Kalman but modify the weight

ActionList<GsfActor, MultiMaterialInteractor, ComponentReductor>

Current Models and Status

Tracking

- Geometry
  - ✓ Layer
  - ✓ Volume
  - ✓ TrackingVolome
  - ✓ …
- Surface
- EDM
  - Measurement
  - Parameters
  - TrackState
- Material
- MagneticField
- Utilities

- Propagator
  - ✓ RKN Stepper
  - ✓ StraightLine Stepper
  - ✓ Navigator
- Fitter
  - ✓ Kalman Fitter
  - • Gaussian Sum Fitter
- Seeding
- Vertexing
- ☐ Track Finding
  - • CKF + HOUGH
- ☐ Calibration – no need
- ☐ Alignment – no need

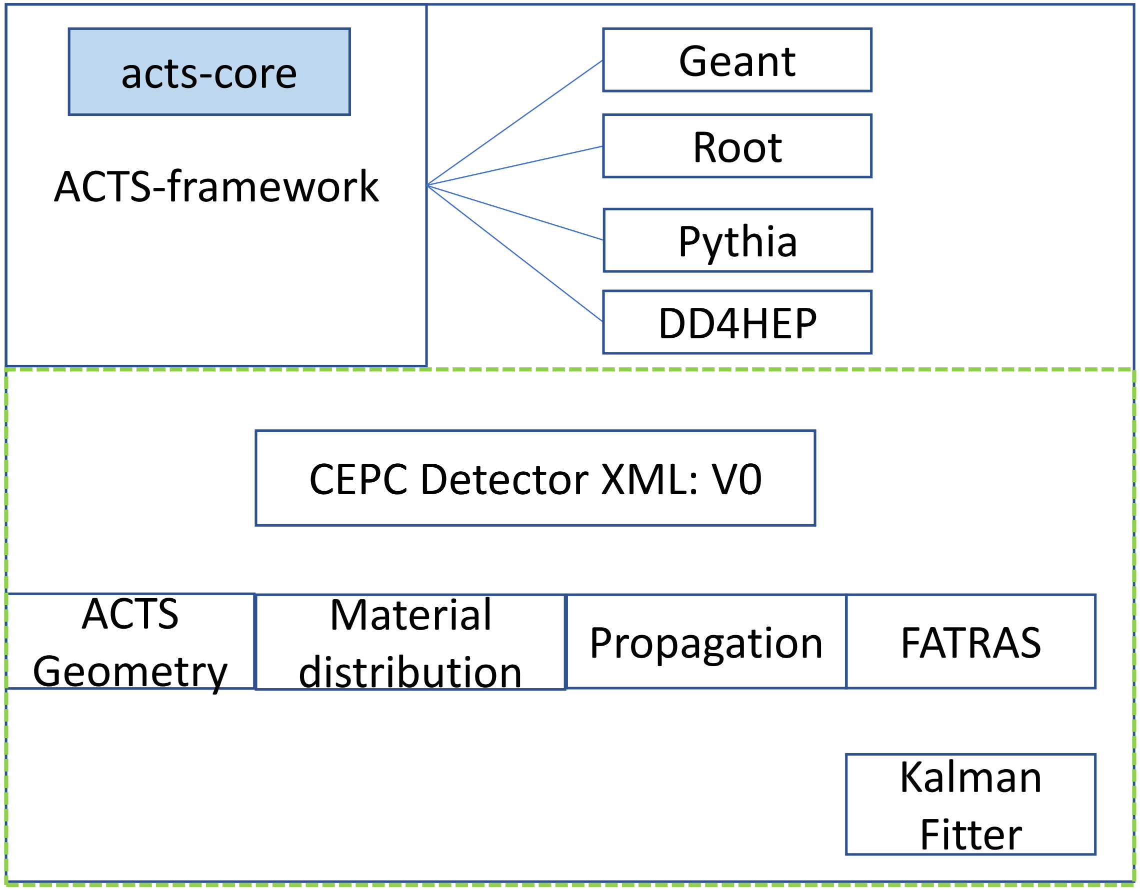
Plugins

- ✓ DD4HEP
- ✓ TGeo
- ✓ Geant
- ✓ Jason
- ✓ Digitization

…

We also have a minimal Gaudi Framework acts to the core codes

CEPC ACTS Integration Status



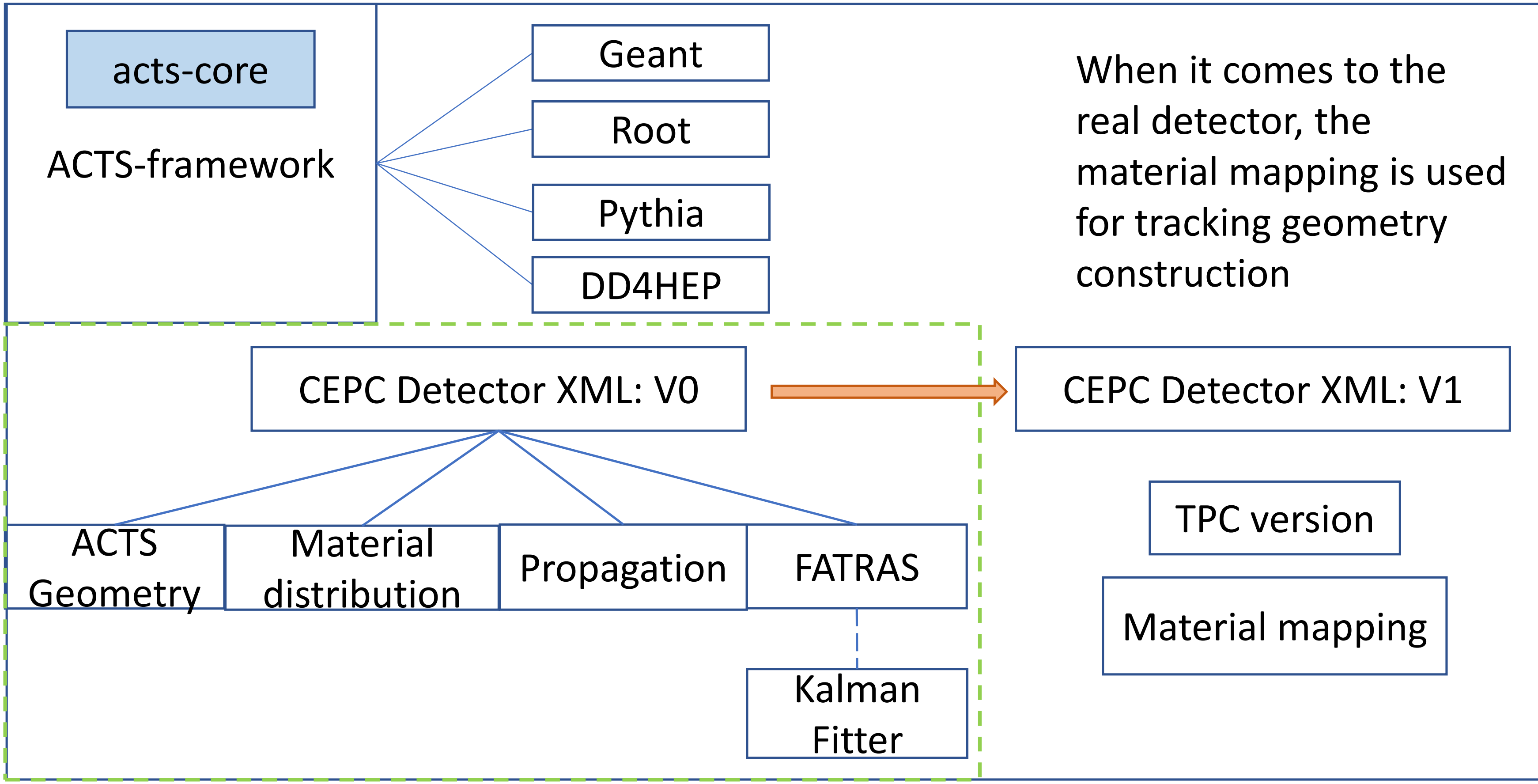
ACTS-framework

Minimal Gaudi framework

Test every model in the Acts-core

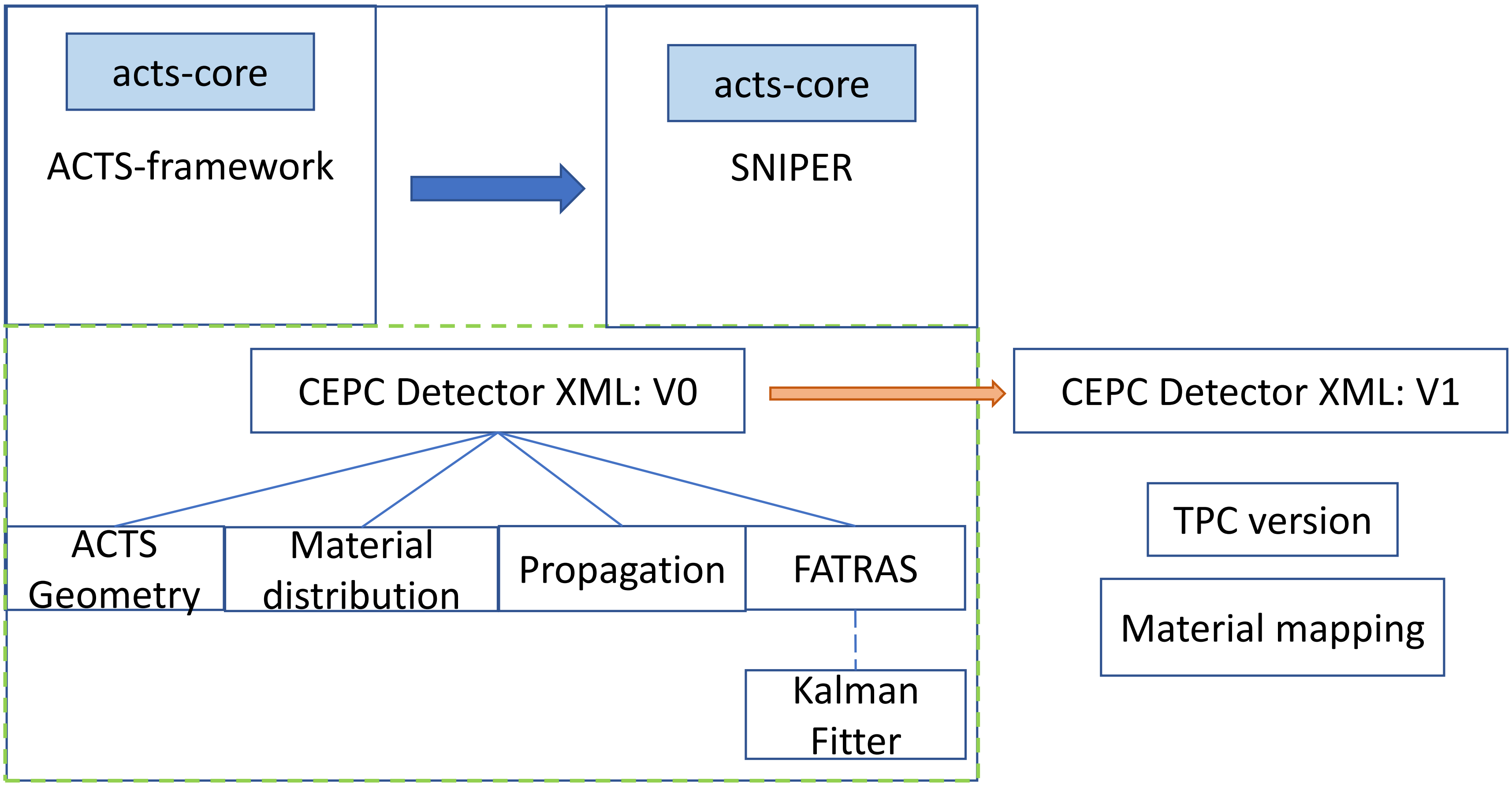
Support MT by tbb

CEPC ACTS Integration Status



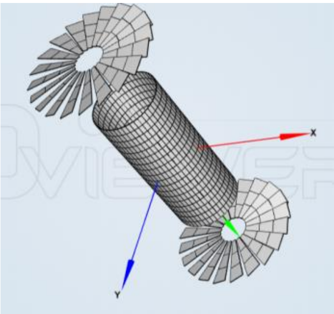
When it comes to the real detector, the material mapping is used for tracking geometry construction

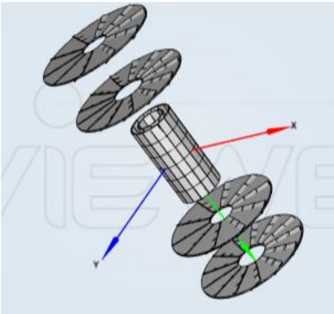
We expect to integrate the ACTS into SNIPER for CEPC



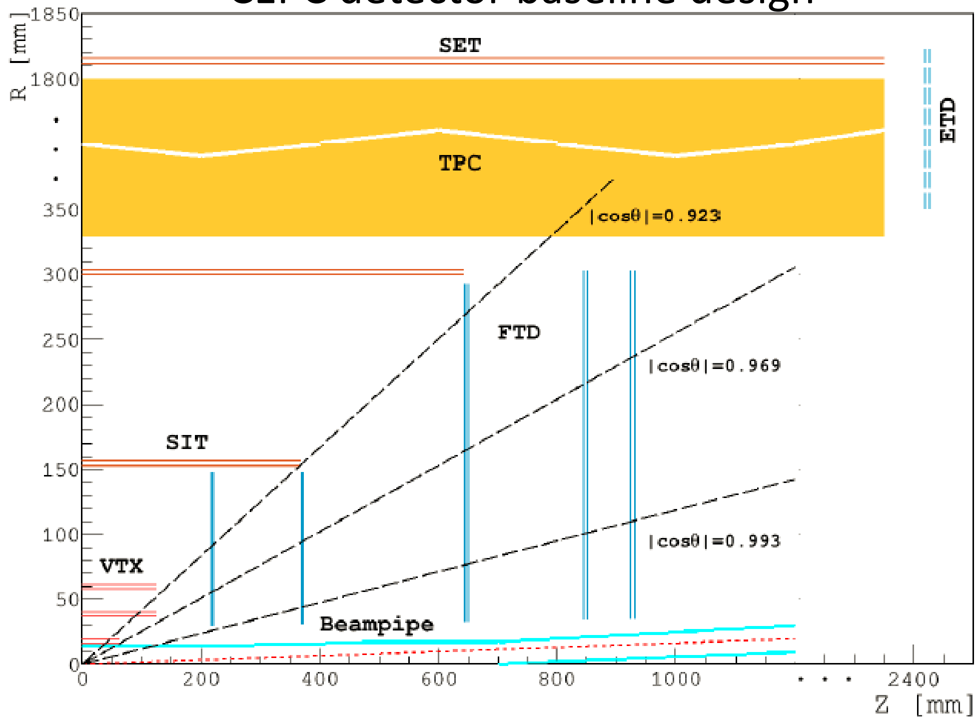
The integration does not start yet

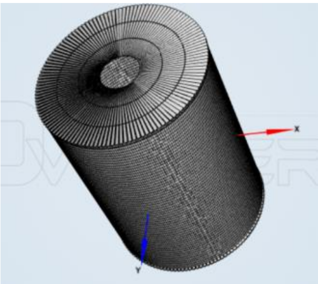
Geometry Example (DD4HEP constructor)

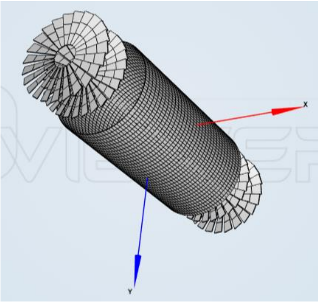




CEPC detector baseline design







VTX, SIT, SET

(0.075mm AluminumOxide

0.11mm SiliconOxide

0.1mm CarbonFiber)

FTD, ETD

(0.0625mm Silicon

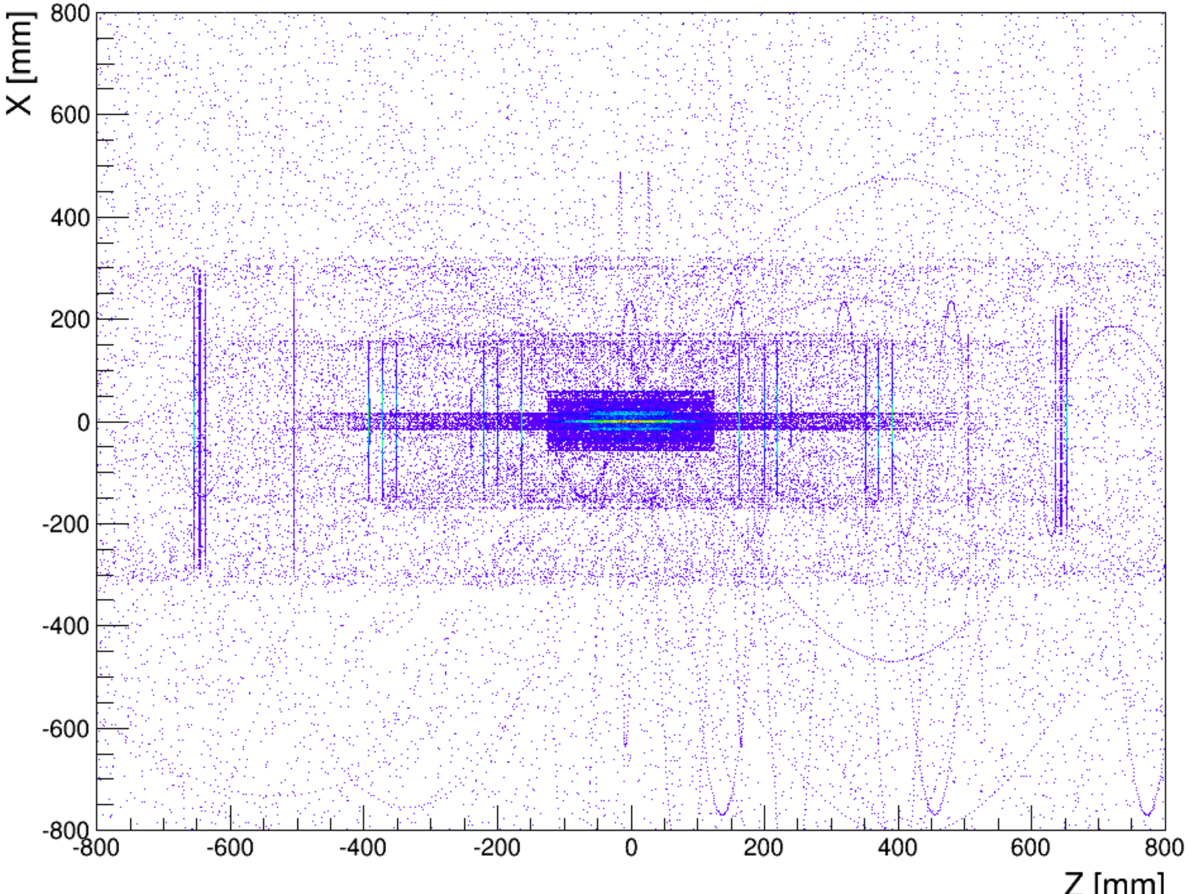
0.05mm Carbon

0.025 AluminumOxide)

TPC has not been included yet

Propagation Example

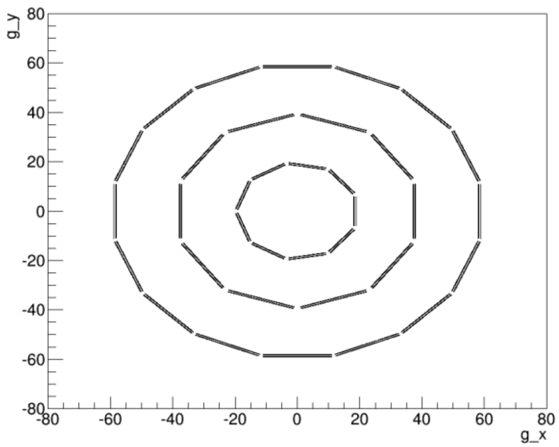
Propagate: generate tracks pass through detector, record all steps



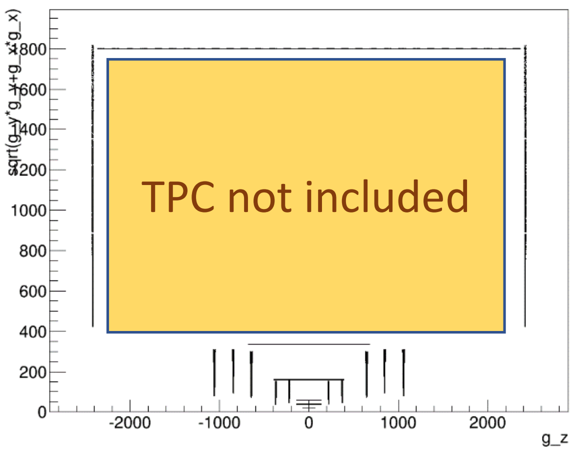
CEPC-ACTS Example

Geometry Validation with Propagate/Geantino recording

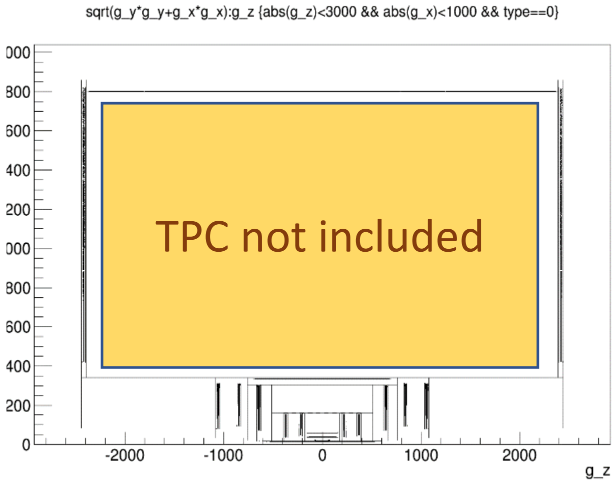
The vertex detector
x-y plane



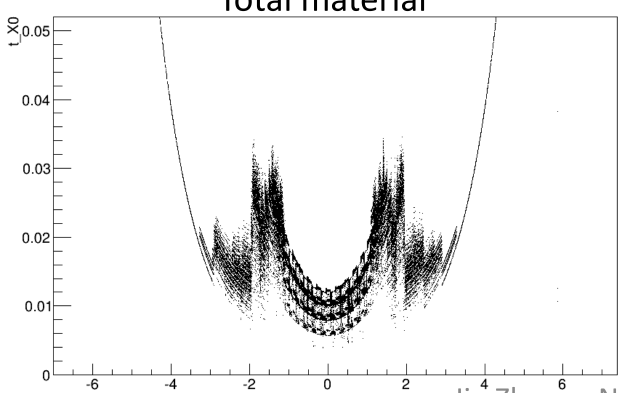
The sensitive detector recording



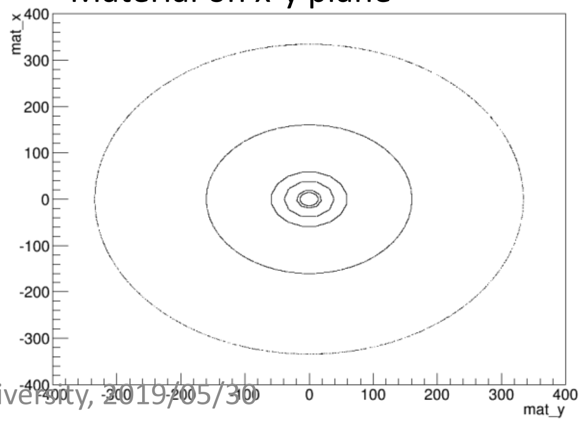
The navigation recording



Total material



Material on x-y plane



Conclusions

- ACTS aims to provide a modern software
  - Develop common tracking models
  - No specific detector design, independent from framework
- We are contributing to ACTS
  - Developing Gaussian Sum Filter and related models
  - Support a validation with CEPC detector
  - Other contributions maybe in the future (e.g. global track finding)
- CEPC integration
  - V0 Geometry is built
  - Could successfully run existing tests with the V0 CEPC Geometry

Outlooks

- Contribution to the acts core
- TPC geometry building
- Integrate ACTS to the SNIPER

谢谢!

Your interests and participation in CEPC-ACTS are welcomed!

https://gitlab.cern.ch/acts

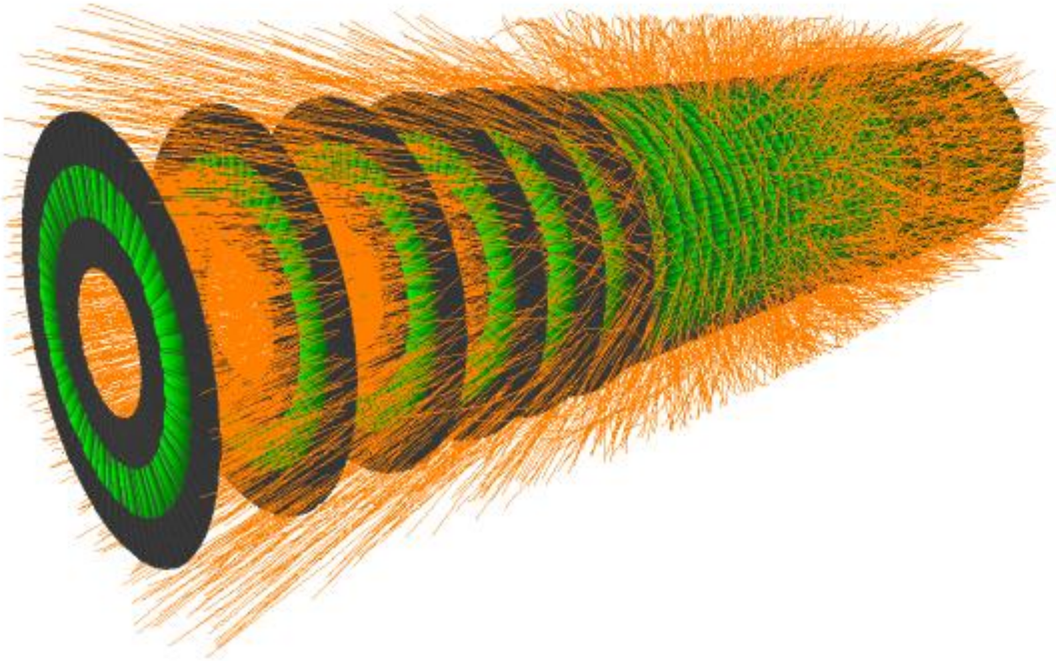
Our CEPC-ACTS metting https://indico.ihep.ac.cn/event/9795/

Backup

Machine learning Support

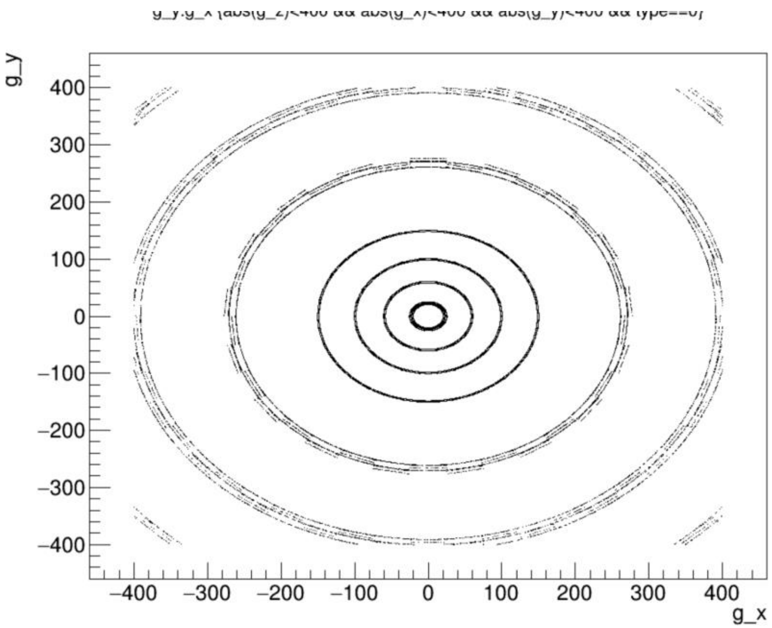
- The “machine learning challenge” detector is support by the ACTS general-ML detector

Tracking ML detector with 1000 events
ACTS fast simulation

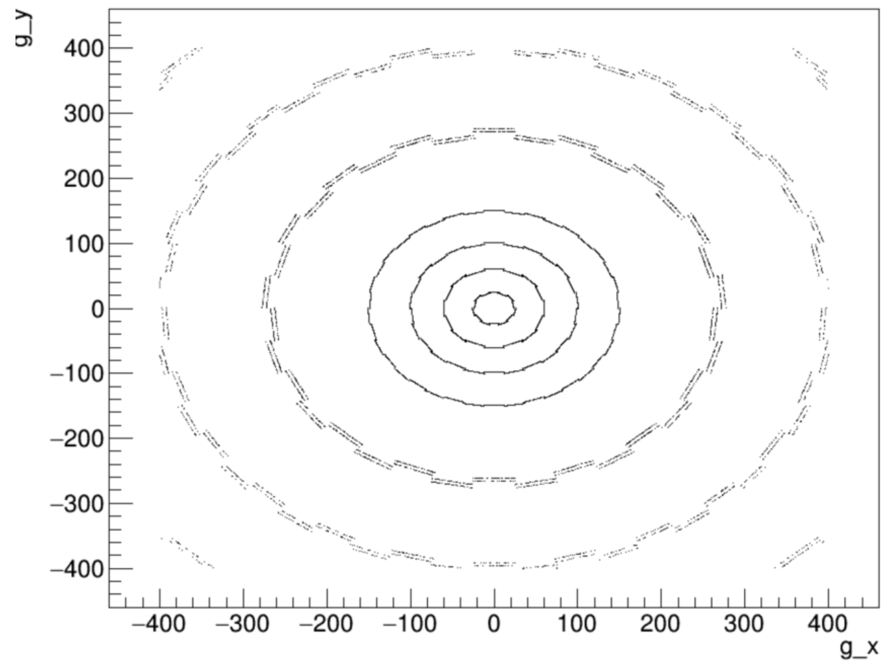


FCC detector Example

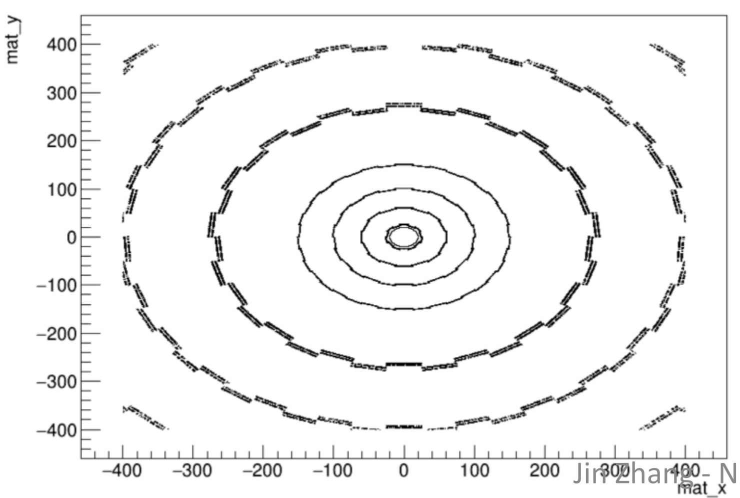
Navigation step

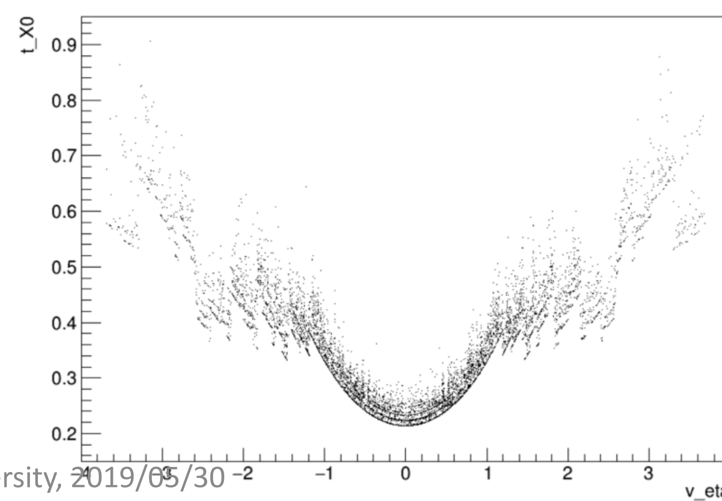


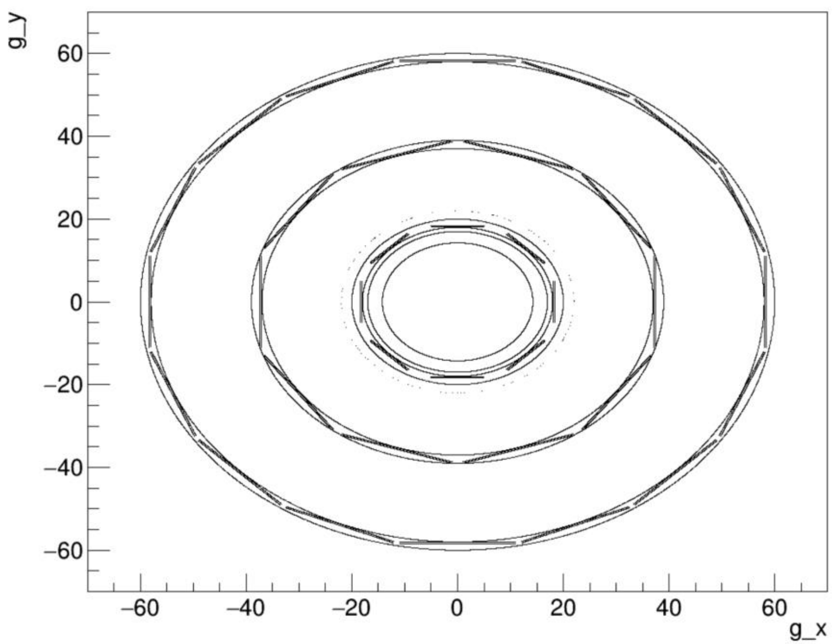
Sensitive

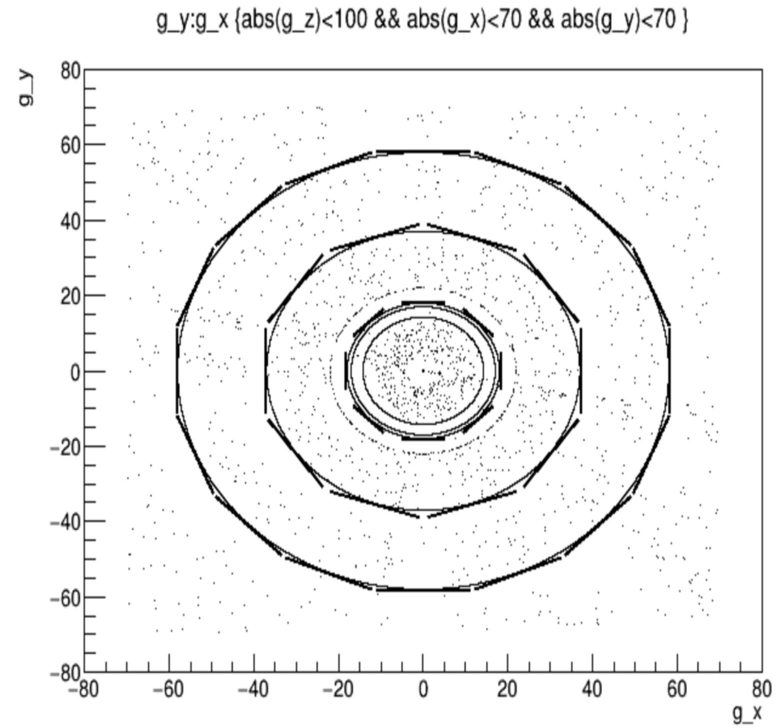


Material





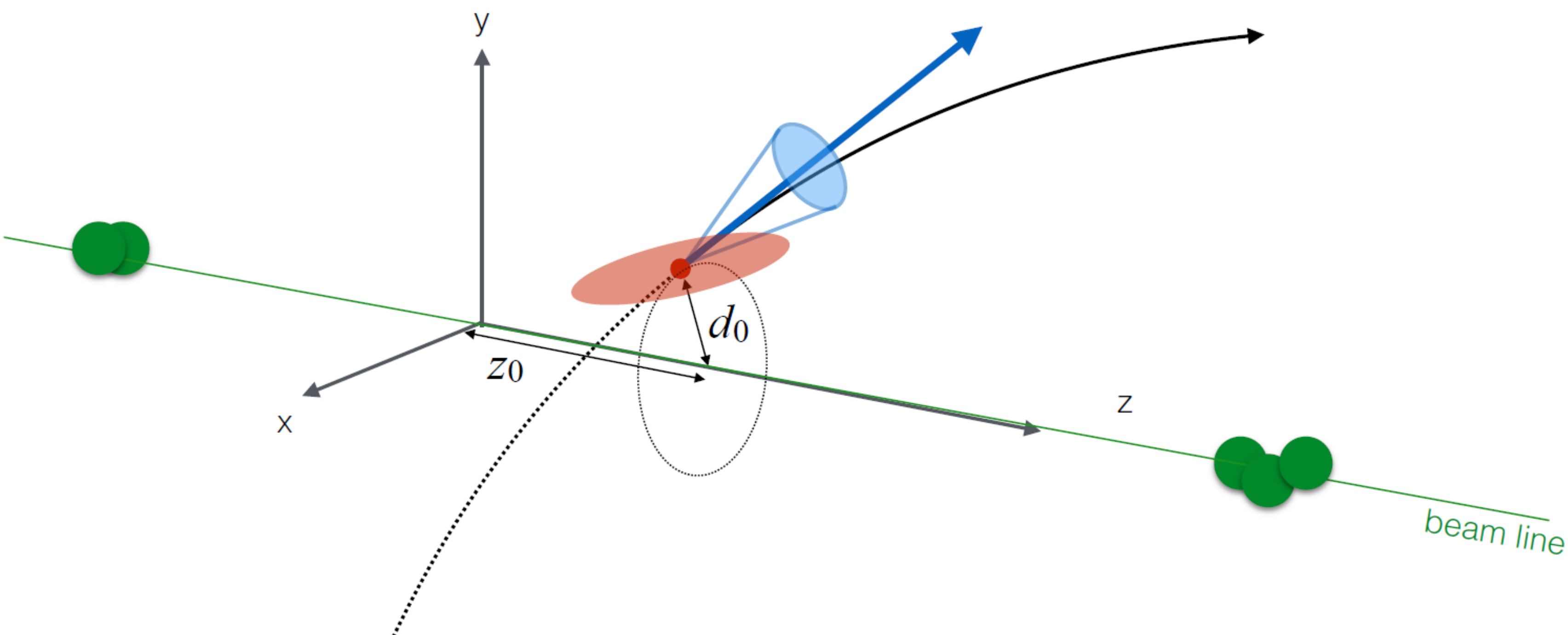

g_y
60
40
20
0
−20
−40
−60
−60
−40
−20
0
20
40
60
g_x


g_y:g_x {abs(g_z)<100 && abs(g_x)<70 && abs(g_y)<70 }
g_y
80
60
40
20
0
−20
−40
−60
−80
−80
−60
−40
−20
0
20
40
60
80
g_x

Jason file for material mapping

```json
{
    "volumes": {
        "10": {
            "layers": {
                "2": {
                    "representing": {
                        "type": "homogeneous"
                    }
                }
            },
            "name": "PST::Barrel"
        },
```

- Perigee representation

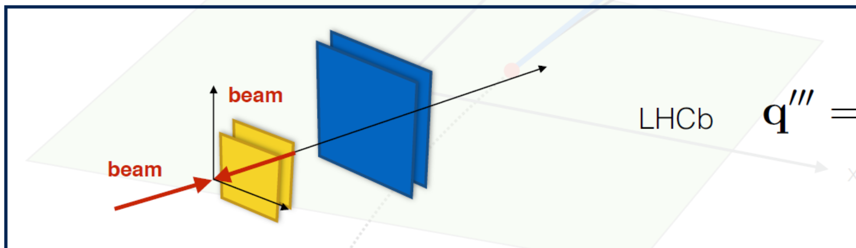


$$\mathbf{C} = \begin{pmatrix} \sigma^2(d_0) & cov(d_0, z_0) & cov(d_0, \phi) & cov(d_0, \theta) & cov(d_0, q/p) \\ . & \sigma^2(z_0) & cov(z_0, \phi) & cov(z_0, \theta) & cov(z_0, q/p) \\ . & . & \sigma^2(\phi) & cov(\phi, \theta) & cov(\phi, q/p) \\ . & . & . & \sigma^2(\theta) & cov(\theta, q/p) \\ . & . & . & . & \sigma^2(q/p) \end{pmatrix}$$

$$\mathbf{q} = (d_0, z_0, \phi, \theta, q/p)$$

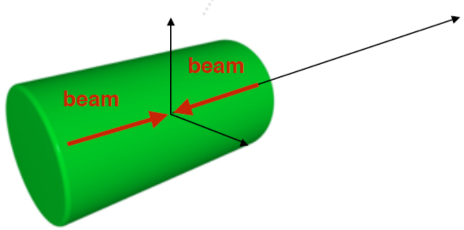
TrackParameterisation

In actual parameterization
General feature
(l1,l2, theta1,theta2,q/p)



LHCb $\mathbf{q}''' = (x, y, t_x, t_y, q/p)$

$$t_{x(y)} = \frac{\partial p}{\partial x(y)}$$

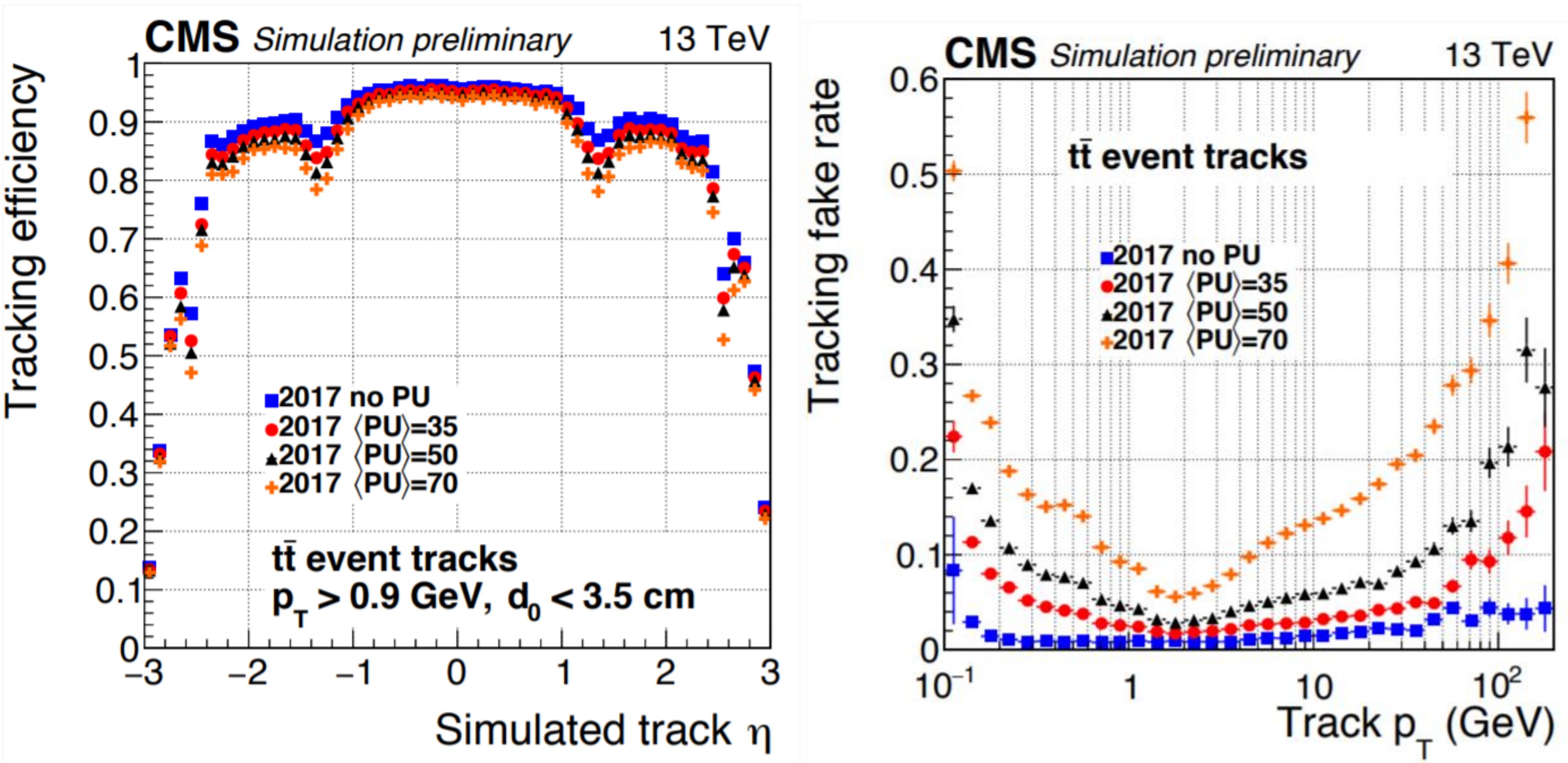


CDF $\mathbf{q}'' = (l_1, l_2, \phi, \cot(\theta), C)$

CMS $\mathbf{q}' = (l_1, l_2, \phi, \lambda, q/p)$

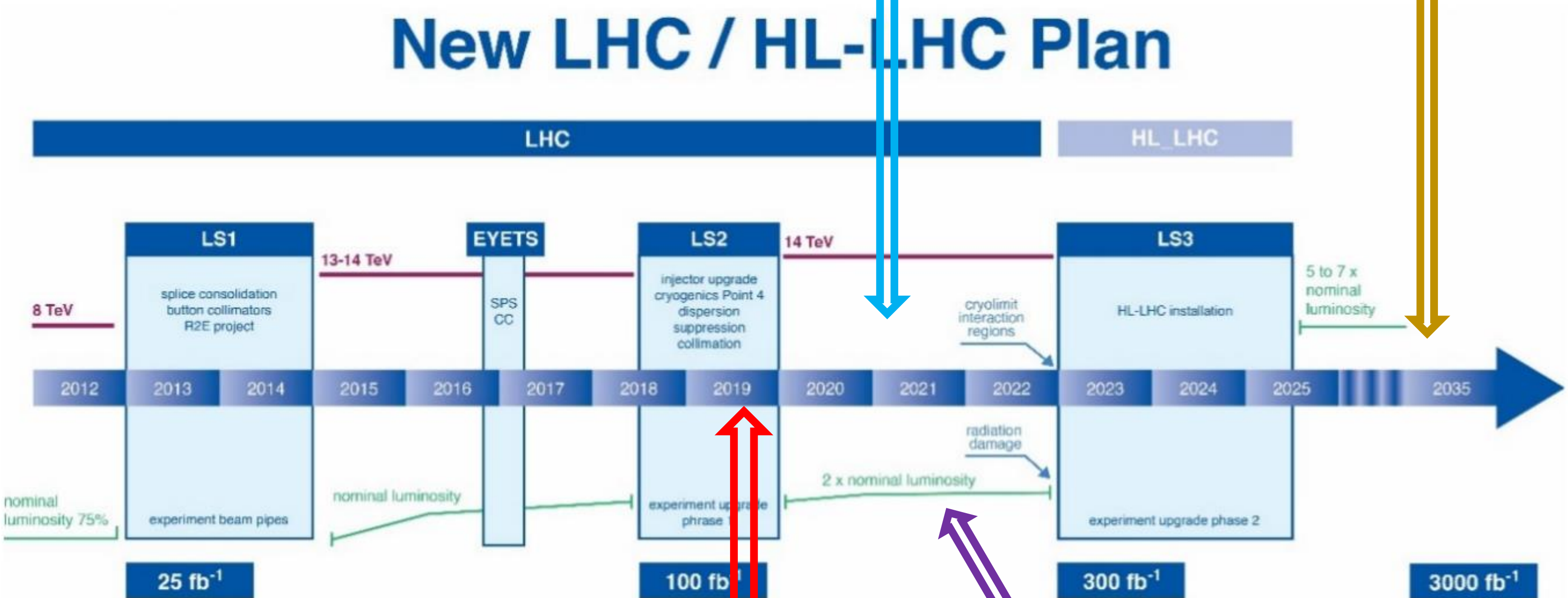
ATLAS $\mathbf{q} = (l_1, l_2, \phi, \theta, q/p)$

Pile up influence from CMS collaboration



Hope to use some ACTS models during Run3

Full MT ready tracking chain inter and intra-event parallelism

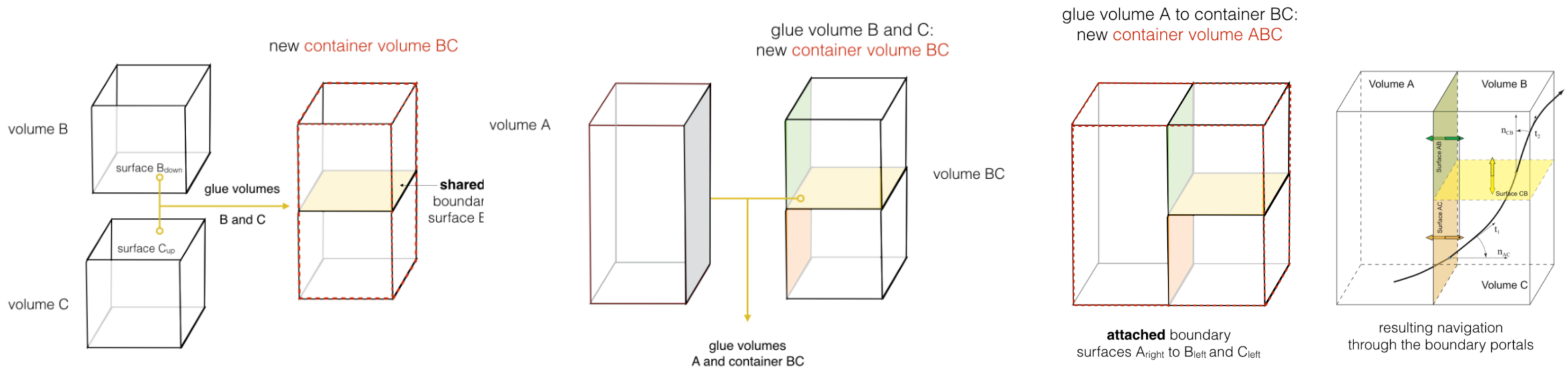


Now

Survive in an MP like environment

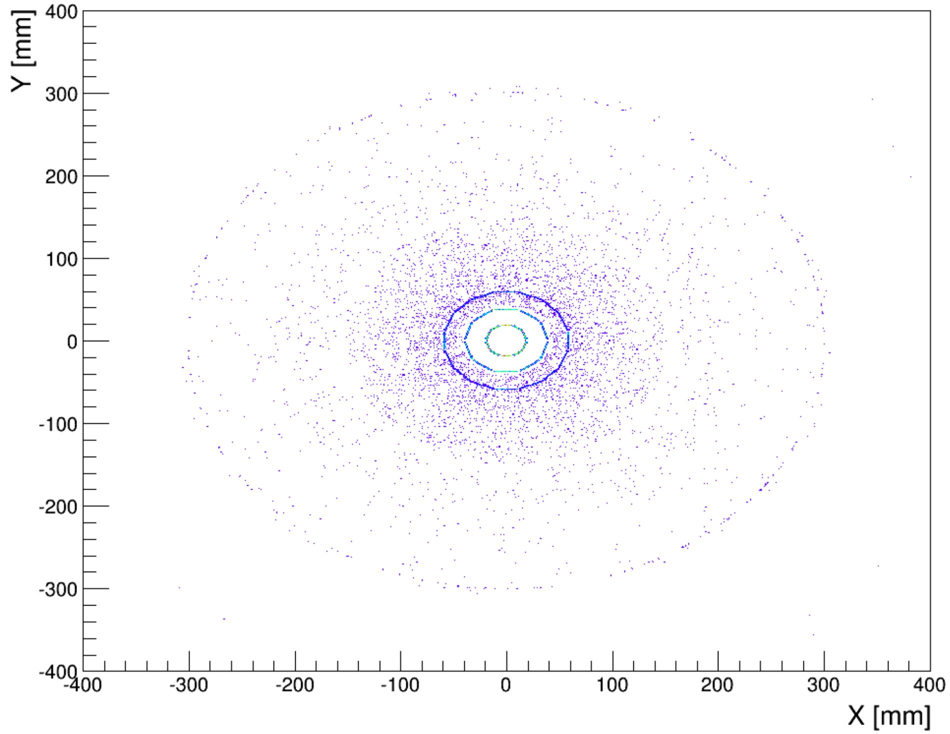
Build A Detector

- Surface->Layer->Volume->glue to tracking Volume
- Material mapping: on surface or volume



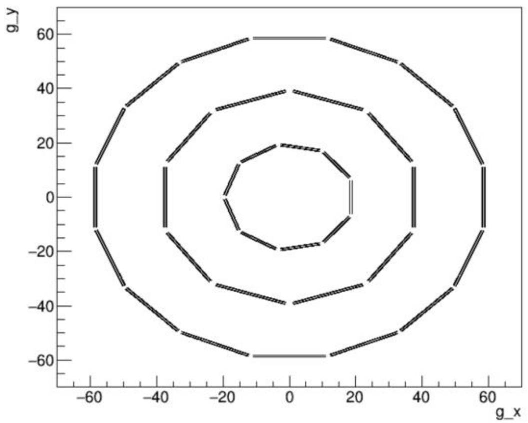
CEPC-ACTS Example
FATRAS && KalmanFitter

- FATRAS （Fast Atlas Tracking Simulation)

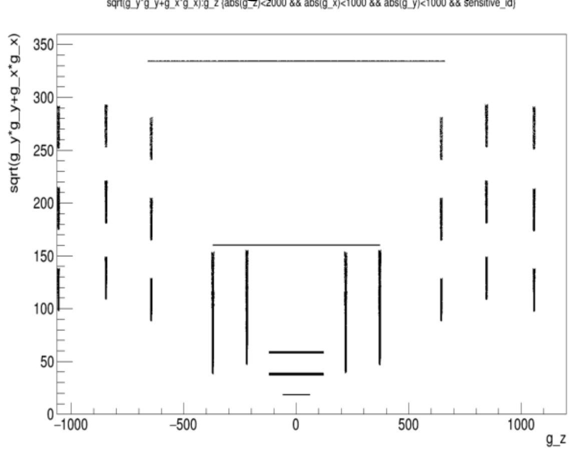


Geometry Validation with Propagate/Geantino recording

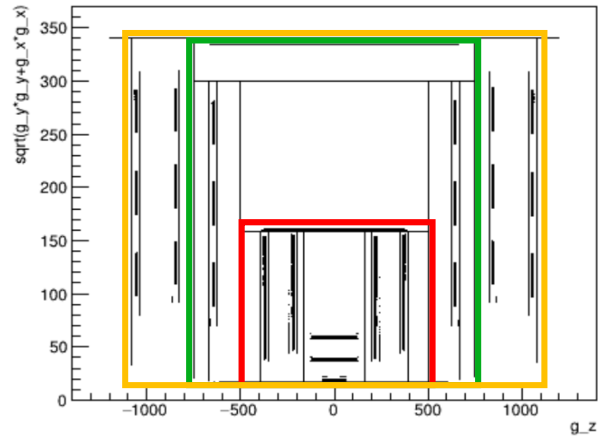
The vertex detector x-y plane



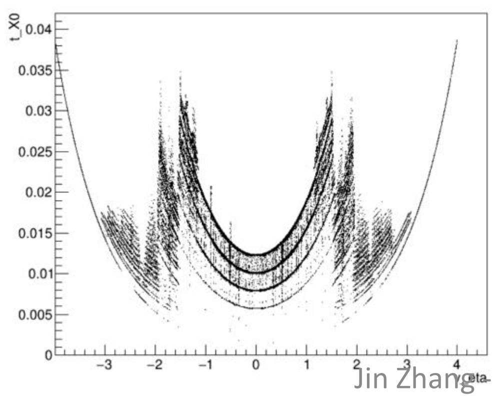
The sensitive detector recording(w/o ETD SET)



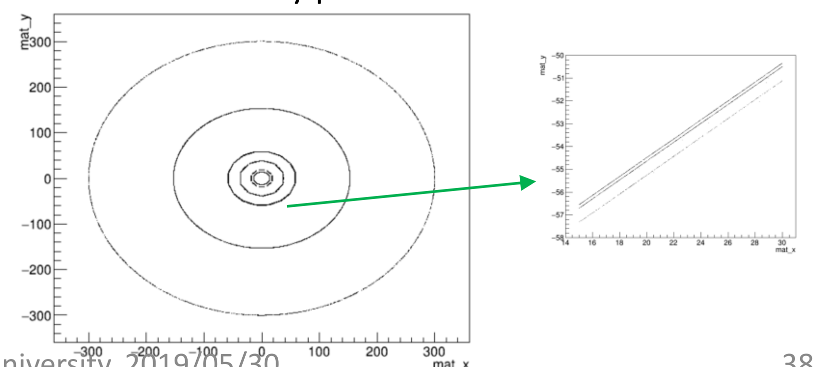
The navigation recording (w/o ETD SET)



Total material

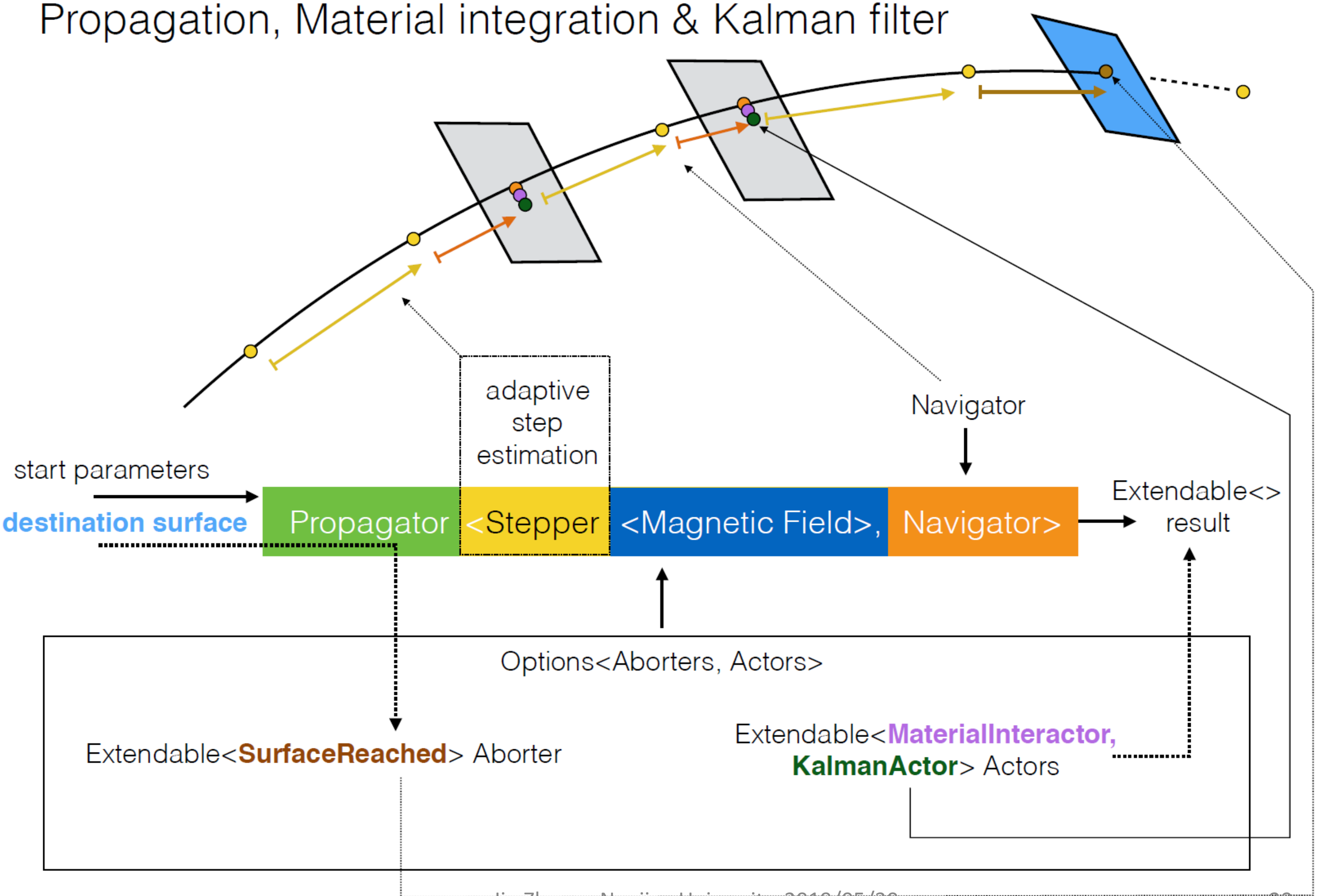


Material on x-y plane



Propagation

Propagation, Material integration & Kalman filter



Measurement

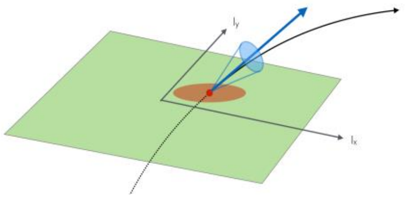
- The general idea is to no detector-related type of measurement, take all measurement as identical in a tracking/fitting algorithm
- Measurement is allowed from 1 dimension to 5 dimension – maximum of **BoundPar**

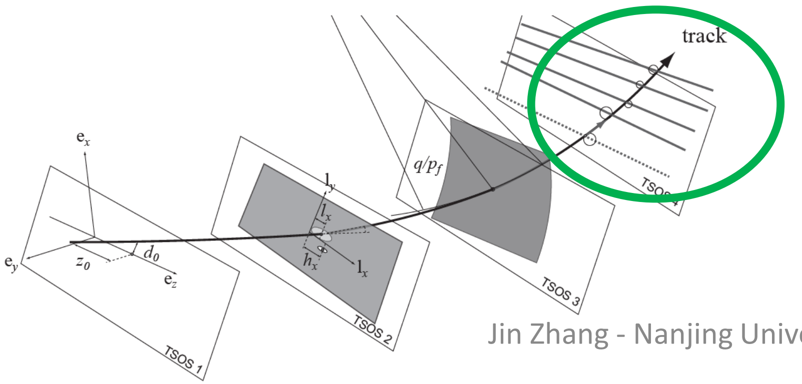
Then we can put different type measurement together in a container and fit them in a simple code
(Pixel cluster(2D), Strip cluster(1D), Segment(4D)…

BoundPar : 5

- ☐ eLOC_0
- ☐ eLOC_1
- ☐ ePHI= 2
- ☐ eTHETA=3
- ☐ eQOP=4

Track bound to a planer surface





Get track from Conformal/Hough transform as a measurement

Measurement in Fitting

Type interpreter to translate
Vector<Measurement> into Vector<FittableMeasurement>

Variant type

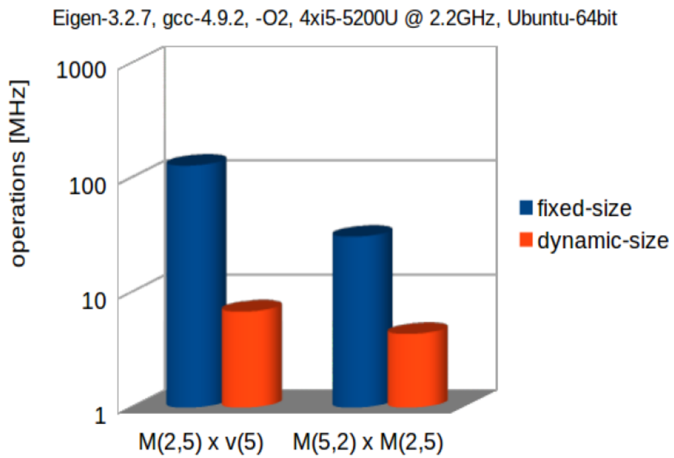
This is the idea of different description of measurement

```
using actual = detail::type_generator_t<meas_factory, 3>;
using expected = std::variant<
    Measurement<SourceLink, 0_p>, Measurement<SourceLink, 1_p>,
    Measurement<SourceLink, 0_p, 1_p>, Measurement<SourceLink, 2_p>,
    Measurement<SourceLink, 0_p, 2_p>, Measurement<SourceLink, 1_p, 2_p>,
    Measurement<SourceLink, 0_p, 1_p, 2_p>>;
static_assert(std::is_same<actual, expected>::value,
              "Variant is not identical");
```

In Fitting Algorithm (pseudocode)
Std::visit ([&] (const auto& calibrated) {…}

std::variant type keep performance benefit from using fixed size matrix

The projection matrix is used to project track parameter to the surface – same size with measurement



fixed size matrix calculation is faster

Contextual data

- As mentioned before, we only build tracking geometry at one time - we need a clean solution for the contextual data

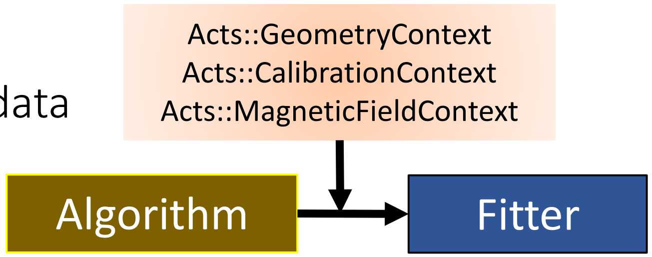
① Allow them to modify detector geometry, dangerous in MT env
② Create different Geometry
Pretty cost

Neither acceptable

- Alignment - Detector changing
- Conditions – Calibration changing
- Magnetic field changing – if necessary

Deal with Contextual data

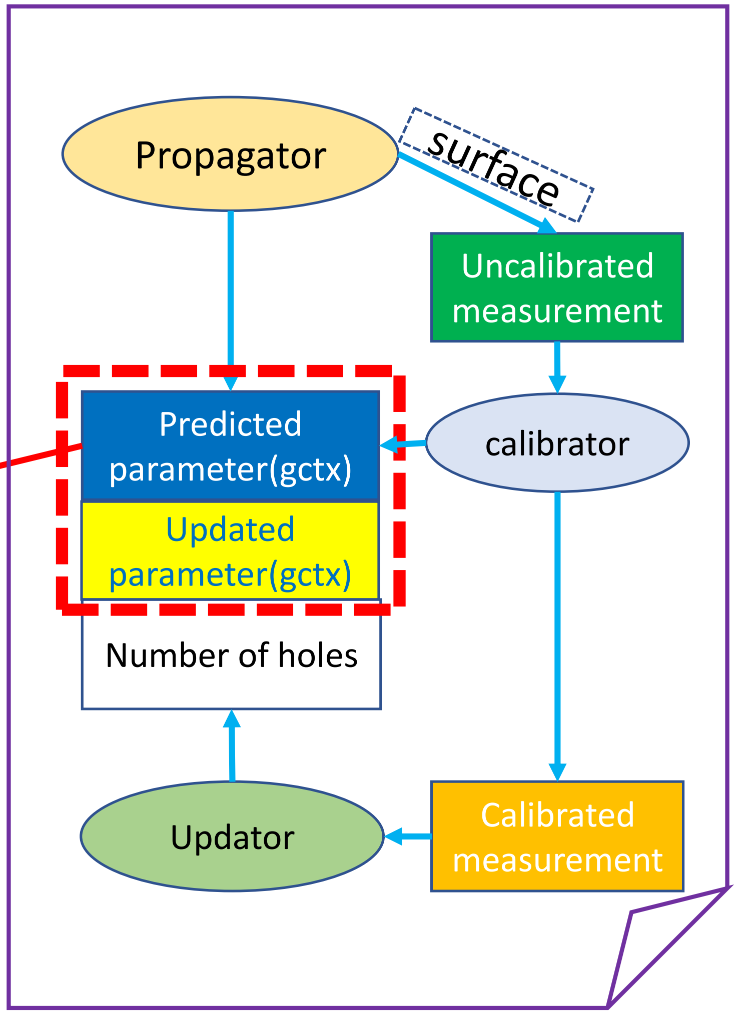
Each event carries
Event Contextual data



When project global parameter onto surface, gctx used local parameters

```
inline bool LineSurface::globalToLocal(const GeometryContext& gctx,
                                       const Vector3D& gpos,
                                       const Vector3D& mom,
                                       Vector2D& lpos) const {
  using VectorHelpers::perp;

  const auto& sTransform = transform(gctx);
```



Surface representation

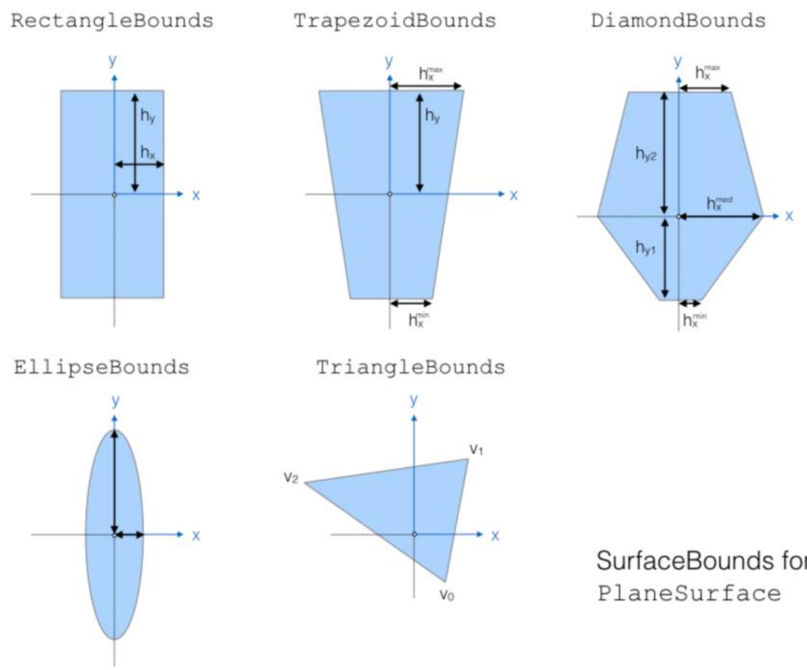
- Transform – position/rotation matrix
- Pointer to the Layer/Volume which contains it
- All surfaces can carry material

- Specific Surfaces has clear bounds and local<->global transform for the specific local coordinate

Surface owns globalToLocal() localToGlobal() method to transform the parameters presentation between local-global frame

```
inline bool LineSurface::globalToLocal(const GeometryContext& gctx,
                                       const Vector3D& gpos,
                                       const Vector3D& mom,
                                       Vector2D& lpos) const {
  using VectorHelpers::perp;

  const auto& sTransform = transform(gctx);
  const auto& tMatrix = sTransform.matrix();
  Vector3D lineDirection(tMatrix(0, 2), tMatrix(1, 2), tMatrix(2, 2));
  // Bring the global position into the local frame
  Vector3D loc3Dframe = sTransform.inverse() * gpos;
  // construct localPosition with sign*perp(candidate) and z.()
  lpos = Vector2D(perp(loc3Dframe), loc3Dframe.z());
```

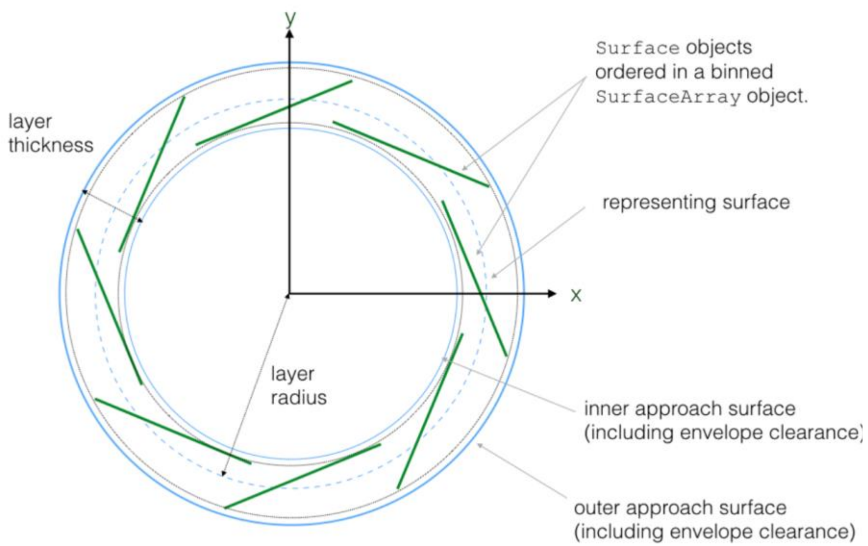


SurfaceBounds for PlaneSurface

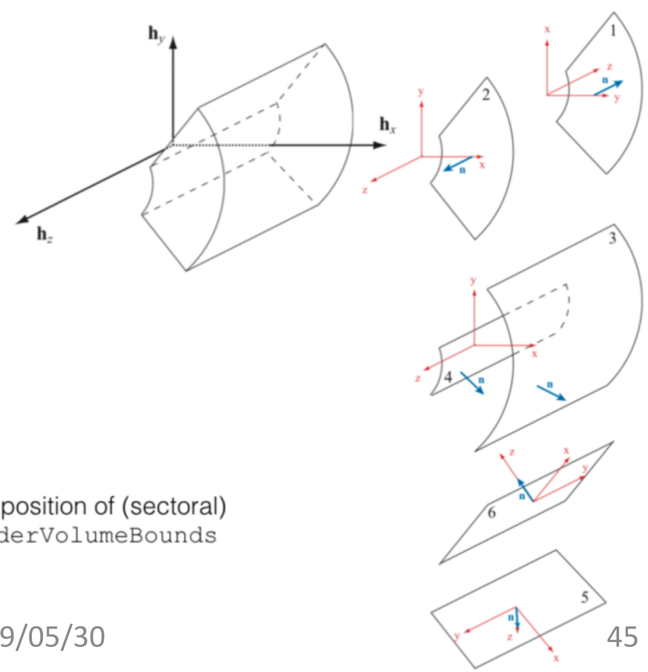
Based on the Surface Concept — Layer/Volume

- Layer: an extend of Surface
  - Different type of internal Surface
    - Representing surface
    - Approach surface
    - Array of contained surface

- Volume : bounded by a set of boundary surface
  - Boundary surface : contains pointers to the attached Volume and inside/out normal vector



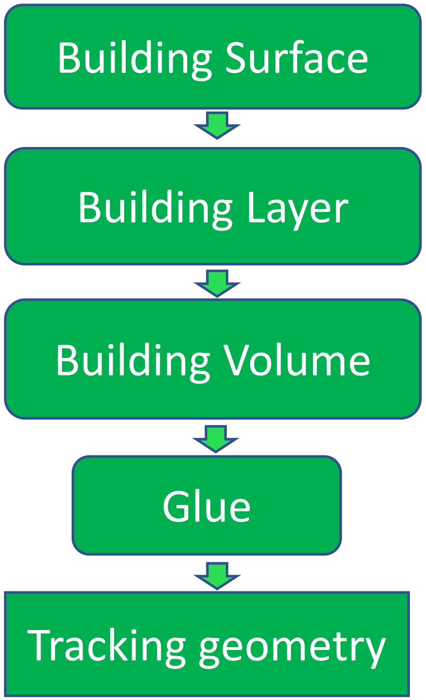
x-y view of a CylinderLayer (with planar detection elements)



Decomposition of (sectoral) CylinderVolumeBounds

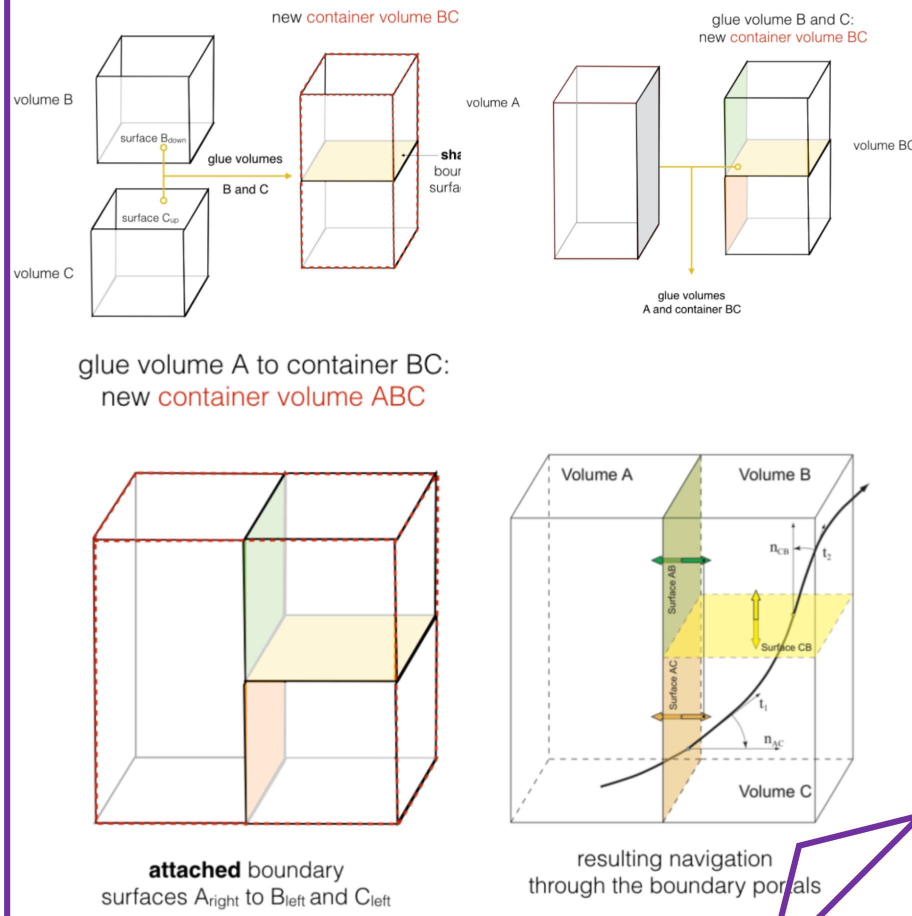
The full connected tracking Geometry

A very simplified geometry building sketch



- The real mechanism could be complicated, but automatically
- Tracking geometry created only one time – other technique to with condition data

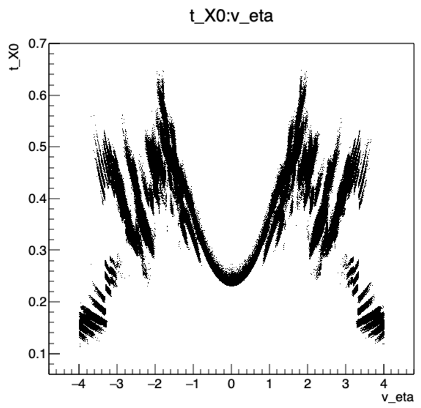
Glue and the full connected tracking geometry



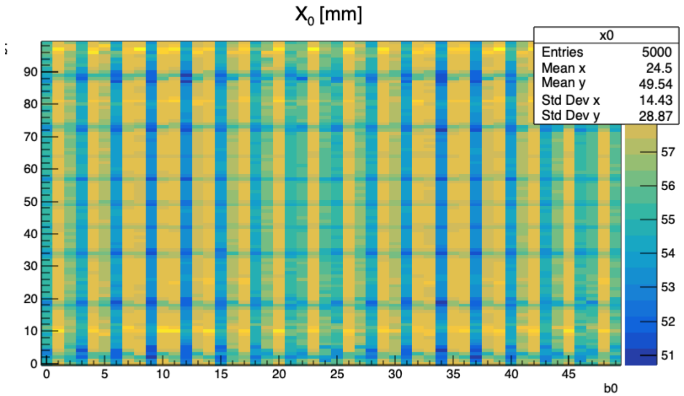
Material mapping

- Tracking needs an appropriate material description
  - Usually simplified version of full simulation description
  - Often used as material description for fast simulation

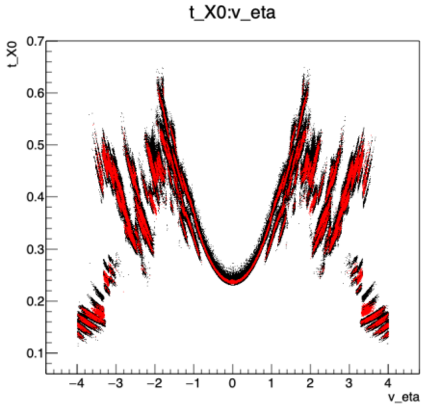
- ACTS provides plugin for Geant4, to record the full material
- A Jason file to record the tracking geometry and your binning
- The material mapping method to map the full material onto the chosen ACTS structure



INPUT

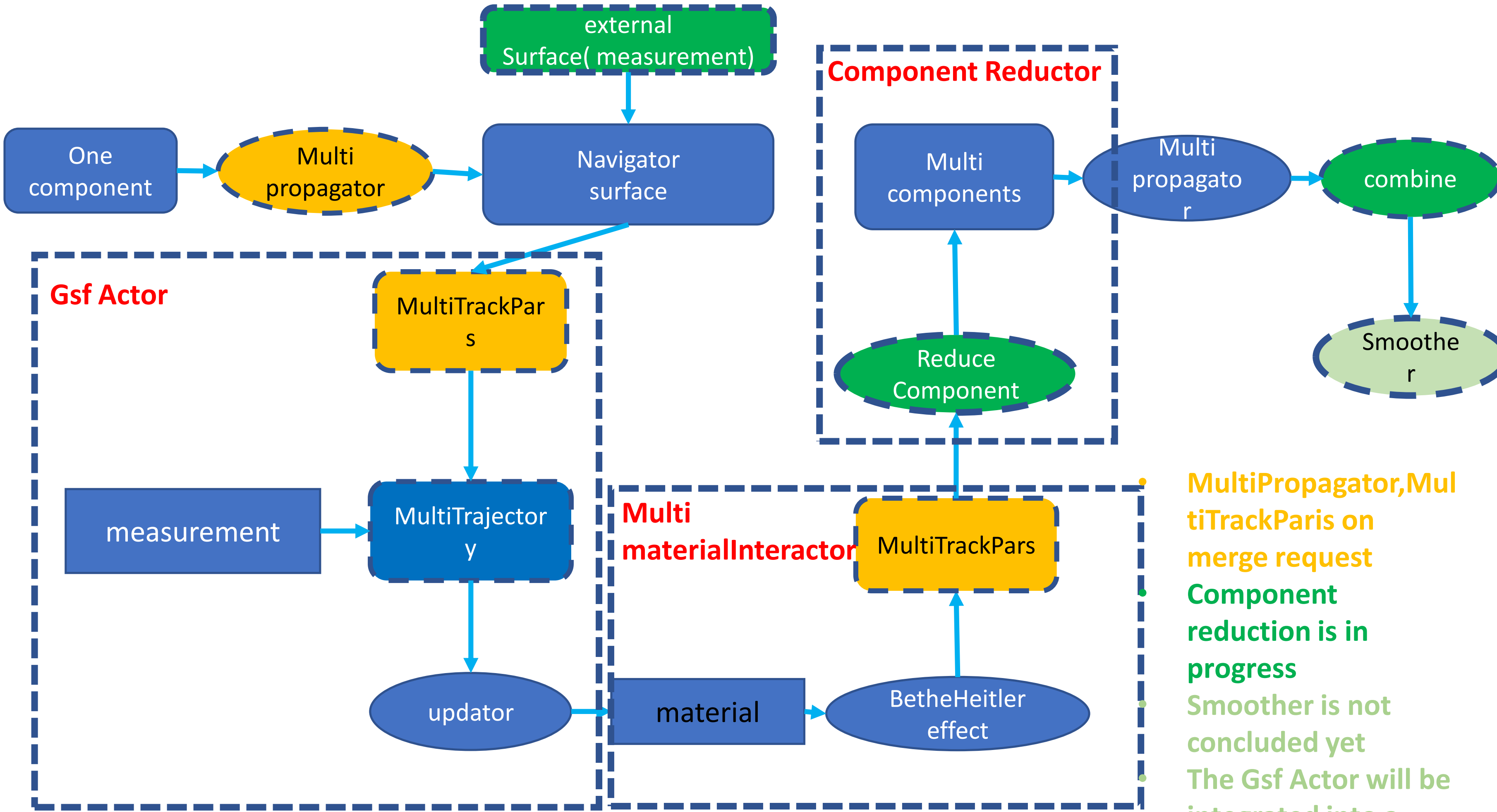


The material map of one surface after mapping with (50*100) bins



Output compares with Input

GSF workflow



- MultiPropagator,MultiTrackParis on merge request
- Component reduction is in progress
- Smoother is not concluded yet
- The Gsf Actor will be integrated into a new MultiTrajectory EDM

ActionList<GsfActor, MultiMaterialInteractor, ComponentReductor, >

Develop task in ACTS

ACTS : Software Propagator

```cpp
  auto& sd = state.stepping.stepData;
  auto dir = stepper.direction(state.stepping);

  D = FreeMatrix::Identity();

  double half_h = h * 0.5;
  // This sets the reference to the sub matrices
  // dFdx is already initialised as (3x3) idendity
  auto dFdT = D.block<3, 3>(0, 3);
  auto dFdL = D.block<3, 1>(0, 6);
  // dGdx is already initialised as (3x3) zero
  auto dGdT = D.block<3, 3>(3, 3);
  auto dGdL = D.block<3, 1>(3, 6);
```

```cpp
/// This is the check call on the a last of all conditions
template <typename last>
struct abort_list_impl<last> {
  template <typename T, typename result_t, typename propagator_state_t,
            typename stepper_t>
  static bool check(const T& conditions_tuple, const result_t& result,
                    propagator_state_t& state, const stepper_t& stepper) {
    // get the right helper for calling the abort condition
    constexpr bool has_result = condition_uses_result_type<last>::value;
    const auto& this_condition = std::get<last>(conditions_tuple);

    return condition_caller<has_result>::check(this_condition, result, state,
                                               stepper);
  }
};
```

```cpp
namespace concept {
  namespace Stepper {

  template <typename T>
  using state_t = typename T::State;

  template <typename T>
  using return_t = typename T::template return_parameter_type<void, void>;

  template <typename T>
  using jacobian_t = typename T::Jacobian;
  template <typename T>
  using covariance_t = typename T::Covariance;
  template <typename T>
  using bound_state_t = typename T::BoundState;
  template <typename T>
  using curvilinear_state_t = typename T::CurvilinearState;

  METHOD_TRAIT(get_field_t, getField);
  METHOD_TRAIT(position_t, position);
  METHOD_TRAIT(direction_t, direction);
  METHOD_TRAIT(momentum_t, momentum);
  METHOD_TRAIT(charge_t, charge);
  METHOD_TRAIT(surface_reached_t, surfaceReached);
  METHOD_TRAIT(bound_state_method_t, boundState);
  METHOD_TRAIT(curvilinear_state_method_t, curvilinearState);
  METHOD_TRAIT(update_t, update);
  METHOD_TRAIT(covariance_transport_t, covarianceTransport);
  METHOD_TRAIT(step_t, step);
  METHOD_TRAIT(corrector_t, corrector);
```

```cpp
/// @brief Propagate track parameters - User method
///
/// This function performs the propagation of the track parameters according
/// to the internal implementation object until at least one abort condition
/// is fulfilled, the destination surface is hit or the maximum number of
/// steps/path length as given in the propagation options is reached.
///
/// @tparam parameters_t Type of initial track parameters to propagate
/// @tparam surface_t Type of target surface
/// @tparam action_list_t Type list of actions
/// @tparam aborter_list_t Type list of abort conditions
/// @tparam propagator_options_t Type of the propagator options
///
/// @param [in] start Initial track parameters to propagate
/// @param [in] target Target surface of to propagate to
/// @param [in] options Propagation options
///
/// @return Propagation result containing the propagation status, final
///         track parameters, and output of actions (if they produce any)
template <typename parameters_t, typename surface_t, typename action_list_t,
          typename aborter_list_t,
          template <typename, typename> class propagator_options_t,
          typename target_aborter_t = detail::SurfaceReached,
          typename path_aborter_t = detail::PathLimitReached>
Result<
    action_list_t_result_t<typename stepper_t::template return_parameter_type<
                               parameters_t, surface_t>,
                           action_list_t>>
propagate(
    const parameters_t& start, const surface_t& target,
    const propagator_options_t<action_list_t, aborter_list_t>& options) const;
```

ACTS hopes to write “modern code”